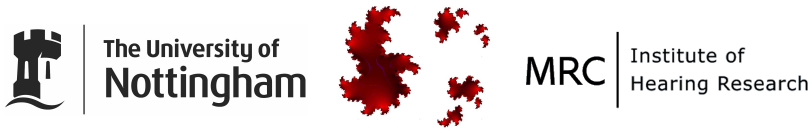
Machine Learning for Neural Coding of Sound Envelopes: Slithering from Sinusoids to Speech

Alban Levy

Christian J. Sumner	Institute of Hearing Research, MRC University of Nottingham, UK
Aristodemos Pnevmatikakis	Athens Information Technology Paiania, Marousi, Greece
Stephen Coombes	School of Mathematical Sciences University of Nottingham, UK

Thesis submitted to the University of Nottingham
for the degree of Doctor of Philosophy

January 2018



Abstract

Specific locations within the brain contain neurons which respond, by firing action potentials (spikes), when a sound is played in the ear of a person or animal. The number and timing of these spikes encodes information about the sound; this code is the basis for us perceiving and understanding the acoustic world around us. To understand how the brain processes sound, we must understand this code. The difficulty then lies in evaluating the unknown neural code. This thesis applies Machine Learning to evaluate auditory coding of dynamic sounds by spike trains, with datasets of varying complexity.

In the first part, a battery of Machine Learning (ML) algorithms are used to evaluate modulation frequency coding from the neural response to amplitude-modulated sinusoids in cat Cochlear Nucleus spike train data. It is found on this recognition task that, whilst absolute performance levels depend on the types of algorithms, their performance relative to each other is the same on different types of neurons. Thus a single powerful classification algorithm is sufficient for evaluating neural codes. Similarly, different performance measures are useful in understanding differences between ML algorithms, but they shed little light on different neural coding strategies. In contrast, the features used for classification are crucial; e.g. Vector Strength does not provide an accurate measure of the information contained in spike timing. Overall, different types of neurons do not encode the same amount of amplitude-modulation information. This emphasises the value of using powerful Machine Learning methods applied to raw spike timing information.

In the second part, a more ecological and heterogeneous set of sounds — speech — is used. The application of Hidden Markov Model based Automatic Speech Recognition (ASR) is tested within the constraints of an electrophysiological experiment. The findings suggest that a continuous digit recognition task is amenable to a physiology experiment: using only 10 minutes of simulated recording to train statistical models of phonemes, an accuracy of 70% could be achieved. This result jumps to about 85% when using 200 minutes worth of simulated data. Using a digit recognition framework is sufficient to examine the influence on the performance of different aspects of the size and nature of a neural population and the role of spike timing. Previous results suggest, however, that this accuracy would be reduced if experimental Inferior Colliculus data were used instead of a guinea-pig cochlear model. On the other hand, a fully-fledged continuous ASR task on a large vocabulary with many speakers may result in insufficient phoneme accuracy ($\sim$40%) to base an auditory coding-related investigation on. Overall this suggests that complex ML algorithms such as ASR can nevertheless be practically used to assess neural coding of speech, with careful selection of features.

Lay Summary

Researchers replace animal experiment by models as much as possible. However, current models of auditory processing are not good enough to reproduce the brain response to sounds. This work studies the feasibility of assessing, using current technologies, how the brain processes complex sounds.

For technical reasons — namely their spatial and temporal resolutions — multi-electrode arrays are the best technology to record data associated with this processing. Electrode arrays can be used to record the electrical activity of a small group of neurons deep in the brain of animals. This method has substantial drawbacks and it is in practice not possible to use it on humans for scientific investigations. Since it damages the brain area under study, these animal experiments cannot last too long, a few hours at best. As a first step towards studying speech processing in the central auditory pathway, the neural coding of simple envelopes is studied. To give a flavour of what a 'sound envelope' is, read out loud the right column of the following table, where capital letters are said loudly, lower case letters quietly, and dots represent short silences:

Fluctuation speed	Sample
None	`AAAAAAAAAAAAAAAAAAAAAAAAAAAA`
Slow	`aaaaaaaAAAAAAAaaaaaaa.......`
Normal	`aaaAAAaaa...aaaAAAaaa...aaaA`
Fast	`aAa.aAa.aAa.aAa.aAa.aAa.aAa.`

The modulation does not depend on the frequencies of the sound being spoken, as the sound '`a`' could be replaced by a sound containing a single frequency, for example 500 Herz. The first part of this thesis aims at recovering the modulation frequency (Slow, Normal or Fast in the given example) from cat spike train neural data, using a battery of algorithms.

The second part studies the neural response to complex and dynamic envelopes: human speech. Three datasets of increasing complexity are used as testbeds, the final one attempting to recognise complete sentences from simulated neural data. For example, here is the sentence recognised by the algorithm

```
Recognised:     CAUSEWAY AND   ABRUPTLY    TWO SURE
```

when the actual sentence that was given to the cochlear model was

```
Real:       THE CAUSEWAY ENDED ABRUPTLY AT THE SHORE
```

Or, to give a longer example:

```
Real: PLEASE TAKE THIS DIRTY TABLE CLOTH   TO     THE CLEANERS FOR    ME
Reco:                  EACH  PLACE CHANGES BURNED CAB EXOTIC   LONGER SURVEYING
```

Results suggest that such a complex dataset is not adapted for an experimental investigation of neural coding of speech, but that a continuous digit recognition task should be.

Acknowledgements

'A PhD is a funky exercise.'
Chris Sumner

The difficulties faced during this PhD were not those I expected, at all. Four years ago, I believed I could move to the UK, get crazy work done, and obtain a PhD by working all on my own without requiring an acknowledgement section. Oh, boy! have I ever been so wrong... It's not a section nor a chapter but a book! it would actually deserve. And I do not want to leave anyone out, be it from my fleeting memories or by extension from these pages: family, people around my research project, housemates, the Nottingham dancing scene, and.... from France.

Family comes first. Merci Florian, pour ton support constant, même dans tous ces moments en dehors de ton champ d'expertise ; les meilleures relations se basent sur un support mutuel, et qu'il est rare d'en autant trouver au sein d'une fratrie. Merci Aurélie, qui de cousine est devenue sœur, dans cette jeune fratrie à trois en quête d'aventures. Merci Catherine (M'man, pour les intimes), qui continue à donner même dans tous ces moments où le temps de recevoir se perd parmi les petites choses de la vie. Merci Philippe (aka Dad), pour un apport de stabilité et de sagesse qui ont permis à certains moments de virer de dynamique et de profondeur. Merci Jean-François, pour ce rappel constant que les privilèges des uns sont fait pour s'harmoniser avec les difficultés des autres, au milieu d'un univers où certains ignorent cette simple réalité. Merci Sophie, pour tes efforts continus des dernières années. Merci Léana, pour la surprenante proximité apparue après tant de temps. Of course, to the Hattori family: thank you for your support. Recent years have not been easy on your side and created a distance I deplore, but you all keep a warm place in my heart and in my prayers.

Europe, as a whole, is a beautiful concept, and watching the UK as it took a step away from it was a dramatic historical event I was able to witness firsthand... This research was funded as part of the NETT (Neural-Engineering Transformative Technologies) consortium by the European Commission, to which I am indebted. This PhD started by a 'No' that slowly became a consensual 'Yes' in a matter of months. For this and more, I am extremely thankful to Chris Sumner, who gave me this opportunity and often patiently sat and chatted with me, scientific and language barriers preventing efficient communication for a couple of years, and who observed time slowing down, full of anguish, as I was going in full gear

towards the seven heavens and back. Stephen Coombes' supervision during my Master's internship and the Gaussian processes investigations were accompanied by a solid teaching of the British way, that ought to guide and sharpen my words for the rest of my life. Aristodemos Pnevmatikakis, unfortunately suffering the consequences of the Greek economical struggle, was a pleasure to learn from and cycle with in the first months of the PhD, in Greece. As for the academic secondment, working at the Imperial College with Anil Barath, Kai Arulkumaran and Rob Holland was a fantastic experience, where I learnt more in 3 weeks than ever before. Finally, the both serious and facetious Sandra Winfield and Davidz Hawkerz, with whom a much closer relationship could have been made possible had my understanding of his multifarious jokes been less limited by accent barriers and language subtleties.

Among the NETT Fellows, three relationships became strong enough to turn into friendship. My relationship with MP Sid Visser (the Meeting Point) is a rare occasion to use regular expressions in a human context: foo*d; it was a delight to have lunch so often with him and stand next to such a tall mind. I mean a great mind. Nibiscus Bojah, my Serbian brother, pushed away to enjoy the Balkan smoking; hopefully we can finally get to work together soon. Nitzan Herzog, who did not exactly enjoy all of his PhD, but nonetheless reached the end of it as a man, as a father, as a scientist, as a friend; I wish you a great (Academic?) career that incorporates all the ideas you want to put together, and that's a lot. Overall, I am thankful to everyone in the project, and the Fellows for giving me the opportunity and honour to be their representative, whatever this may mean. More specifically, I am extremely grateful to all organisers of the ICSLANE 2015 conference in Barcelona: it took time, it was not an easy thing to do, and I shall never forget the experience of chairing the Neural Coding session and the final group discussion, as imperfect a chairman as I may have been. The stress prevented me to sleep or eat, and two Red Bulls may not be the healthiest diet, but in retrospect it was absolutely amazing and definitely worth the effort; the shoeless-conference trend ought to survive my absence. Hail to Nitz' spreading the word regarding money limitations. Spoiler alert: they were slightly incorrect, hence the adventure.

From the University of Nottingham, my thanks go to many more people. From the Institute of Hearing Research, besides Chris, the companionship of Mark Steadman, Toby Wells and Colin Horne in the beginning of my PhD was much enjoyed. With time I grew closer to Christian Füllgrabe, with whom I enjoyed one a many lunch and discussions over the innermost nature of the IHR. Eithne and Venessa: one of the most profound discussion I ever participated in, and you girls gotta dance. I am strongly indebted to Ning Ma, from the University of Sheffield, who showed me the HTK Lord of Light when the night was dark and full of terror. The same goes to Guy Brown, University of Sheffield, for a clear view that removed some of the remaining shadows from this thesis, and Reuben O'Dea, for a final push.

On the Maths' side, a good bunch of people circled over the years. Wilhelm Braun with whom I played Go long ago; can't remember who won. Kyle Wedgwood, who almost convinced me straight away to run away from Nottingham by crushing my brain with his English accent, now appreciated. Georgina Fenton, sweet touch of constant good mood in the School. Agne Tilunaite, our first discussion being about the vanity of small talk; as a gesture, our second discussion started with it, and silences turned awkward are now a fine delicacy when shared with her. Aytul Gökçe, who slowly understood that the amount of seriousness I would put in my sentences had a high variance, a negative mean, and more interestingly an almost infinite skewness, which could be scary. Mayte Bonilla Quintana, who will punch you if you do the wrong move on her; been there, done that, slowed down the dance. Elisa Tonello, both clever and ballsy as can be, admirable mixture. Luis y Patricia y Lemon, big souls in a big house. Kai Groh, a surprisingly nice philosopher for a German. Tupac Ibarra, too packed. Lia De Simon, a constant source of happiness and positivity; so much that she's close to the uncanny valley, where good characteristics become creepy. Andreas Z. Finke, who was pleasantly found in more social circles than anticipated, and in less still than wished. Dave Parkin, who turned up to be quite a cool IT guy, always reactive and efficient. Ian Dryden, always encouraging. By extension, Mister John, Beeston's Thai restaurant Sanchan's wiseman and music lover; keep the good sound going John, and see you soon for a Pad Thai Moo! And Beeston's Nero staff, who eventually knew what I wanted before I tell them, thus convincing me that witchcraft is real.

Regarding housing, lots of adventures were had, from the Chinese parents moving in to the retired lonely srilankese priest, from the cold silence of an old house to the nightly noises of a drug dealer and his dog. To Pimmie, a warm kitchenworm. To Costanza Bergo, welcome support in an awkward situation. To Josh Melton and Alexandru Dimitru: this shared house has been a bumpy road, but we all learned from those hard times. To Laura Vilkaite, who brought a touch of stability in this otherwise messy and awesome house. To Master Xiaofei Sun, whose grandiose evolution I was proud to observe as a bystander, witness or supporter over the course of two years, and in return was mine: you found yourself, and found yourself solid friendship roots, may they continue to strengthen, deepen and spread.

To Jason and Barbara, who showed me that a night full of terror could be an interesting frenzy. À Gustave, pour la fâble cachée de nos chats haut perchés.

To Kathryn, for whom I hopefully no longer am a... what was it again?! Ah, right... A conundrum... But anyways; these few meetings we had were each unique and amazing in incredibly different ways. They may not have been much in number or time, but their greatness made them dear to me.

Now, place for a good dancing crowd. Wow, there are so many people... So many great dances, laughs, game faces. Bachata used to be this foreign beauty I could merely dream about, and has now become a lifestyle, a life journey and a life goal. Recent events suggest that the latter is only true for my English self, but an acknowledgement section is no place for deep psychological analyses... Most centrally, it is good to start with the LTP family, Sol y Laura, Jackie Caspersz, Tamba Hissirou, Billy Homan, Felton Da Costa. You guys have managed to make of Nottingham a dancing nest in which I could evolve into the proud and happy social dancer I love being. Thank you so much for organising Bachakiz, can't wait to come back for the 2018 edition!

Among these dancing people, Giannis Kogias played many roles: a school comrade first, then a buddy, a housemate, a friend and finally a role model. An amazing friend and an even amazinger dancer, whose lessons in social calibration I have craftily not mastered... These late post-dancing-Thursday-night-chilling sessions in Tesco were an important and cherished ritual; since stress is your Achille's heel, I wish you a stress-free life journey! A few months back, Srishti Gupta and I said 'Let's be friends'; unfortunately it worked. Maria Fischesser, may we finish to watch 'Kubo and the Two Strings'. To Cato Rolea, a Romanian with more complexity and contrast than he dares to admit and enough to fear to show; may you find the perfect balance. In the meantime, give us some damn good remixes, you have that in you! To Maria Patsia and Cory Siddall, for being amazing in their openness, kindness and simplicity. To Paulina Ogrodzki, for many very different dances and for being my partner in the uni competition; that defeat was a valuable dancing lesson: Social and competition aren't the same thing. Uncanny... but still a great day! To Stella Georgiadou, Danai Galaziou and Gosia Weirdak, for allowing me to lead you, one step at a time, to a realm of clouds on which we could freely dance to satiety, or at least until the song finishes. To Hollie Mit, Bex Hawkins and Veronika Simonova, for enough dances to walk together towards a better dance understanding as the never-ending promise to even greater dances. To Björk Tyril, Rachel Melanie Parkes, Laura Rutty, Milena Iacovidou, Sanduni Senaratne, Wally's M. Kab, Jyoti Parmar, Jade Creber, Jodie Mallett for more smiles, patience and dances than I deserve. To Richard, very not Gendo Ikari like. To Natalie Williams, who has to continue working on her basics and straight up! To Lucy, towards whom an initial shyness prevented me from dancing enough with; I hope we dance and hike again, and code sometime. To the good-crazy Lisa London: we need to have a talk about religion before we go our way. To Ylenio Lg, for hosting me and being cool. To Cedric Henry, just for being cool; the 'Giannis of Kizomba', as was heard. For teaching me that Life is irony, to Thomas Michael Dichmont, whose advice to go explore the world while I was still free made me realise it was not what I wanted, which paradoxically shifted my life around since I now knew what I wanted, and which meta-paradoxically I may end up following. And of course lots of love to Korke y Judith: even though we have barely met, I proudly remain a Korke fanboy!

Distance truly is a terrible thing, more painful and necessary than it is reasonable to address here. Now then, the most delicate bit... France, and all the tied knots temporarily left behind. After hiding for so long, it's time to acknowledge and write some things in stone.

À Mélo et Raph, sans qui le non initial serait resté un non, prouvant qu'un encouragement peut changer le monde ; et mec, tu me dois du fric. À Louis Veyrat, car une invitation tardive vaut mieux que jamais ; tant pis pour toi, je suis quand même venu, nah. À Camille Manano, qui fait coucou de l'autre bout du globe. À Harry et Marie Cauderlier-Guy, vive les jeunes mariés ! À Moubi, jeune père dont la belle normalité me dépasse de bien des façons. À Jérémie Saives, qui naïvement souhaitait rester à la fac ; as-tu déjà changé d'avis ? À Lou Scotto Di Covella, petite bête qui monte, qui monte, qui monte. À Elisa Rebolini, mystérieusement hors radar. À Lucile Alexandre, envers qui les mots se perdent, en attente d'un contact qui les guidera. À Camille Niaufre, pour d'autres mots en attentes. À Leïla, par habitude plus que par raison ; sais-tu que le cœur est bien étrange, car renoncer à le saisir est le seul moyen de le comprendre, mélange d'humiliation et d'humilité pour l'intellect qui dépasse les perceptions. À Matti, car il n'y a que les relations auxquelles on tient le plus qu'on peut autant endommager, la peur ayant ce pouvoir hypnotique qui attire le papillon de nuit vers sa flamme fatale. Aux Rux, avec qui si peu de pizzas ont été échangées, pour tant de messages et d'émotions. À Charles-Pierre Astolfi, pour un potentiel encore latent. À David Montoya et Sara Lacroix, prepare to dance - le 22 juillet 2017, et tu sais que j'ai les jetons. À Emeline, pour une hospitalité paradoxalement constante et non-triviale. À Émile Contal, à qui je dois ni plus ni moins qu'une part de mon identité et de son sens ; qui a lu si loin dans mes contradictions et peines que je ne peux l'évoquer ici sans pleurer ; qui savait quand je ne savais plus rien. "Si ta vie à Nottingham nécessite un nouveau masque, porte-le au moins au début, enlève-le le soir, et organise un lever de rideau en douceur", juin 2013.

The story of these years in the UK is intertwined with another story, complex and bitter. To my ex. To Iuliz, towards whom writing current feelings would be one more nameless injustice. I dreamt a timeless life, where absence would be a dimensionless point between two instants full of presence. It is almost what this life between two lands has been, a two-geared life, without the poesy but with the pain caused by this spacetime distortion. Our constantly evolving relationship suffered from it, from this isolating and alienating distance, insufficient support to compensate for the accumulating instability, as a spring pulling our selves against the diverging flow of our respective lives. I pray that turning this PhD page will allow me to turn yours, as too much red ink has sunk the margins and my hands beg to let go of this page, rushed as they are to write on the next page yet another name: Liberty.

Merci. Et à bientôt.

Contents

Figures

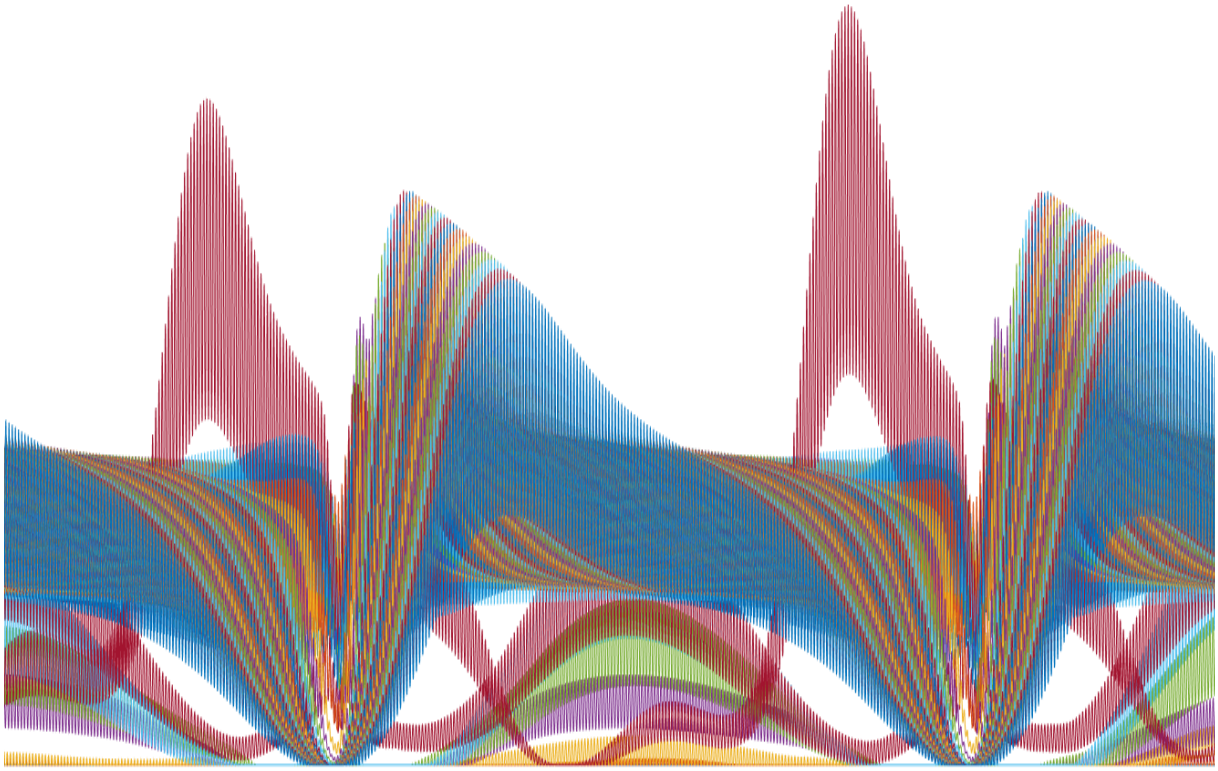
Tables

"**Nervous**: A condition of fear that causes one to be tense, awkward, jittery, and apprehensive, possibly with trembling or excitability, often caused by being uncertain of what dangers may be looming or how to handle them. For example, teenagers regularly feel nervous when they don't know what an attractive person of the opposite sex thinks of them."

Urban Dictionary

"One problem with being a large organism is the difficulty of passing messages between different parts of the body. The solution for all large animals is a **nervous** system."

Chris Plack, The Sense of Hearing



Glossary

$\mathbb{N}$	Integers
$\mathbb{N}^*$	Non-zero integers
$\mathbb{R}^p$	Real Euclidian space of dimension p
$[.,.]$ $].,.]$	Real intervals
$\{...\}$	Set
$\|\|.\|\|_p$	L^p norms
$(x_i)_{i...}$	Sequence indexed by i
CV	Coefficient of Variation
FFT	Fast Fourier Transform
Tr	Trace
f_c	Carrier frequency
f_m	Modulation frequency
$\mathbb{P}$	Probability
$\mathbb{E}$	Expectation
$\mathcal{N}(\mu, \sigma^2)$	Gaussian, mean μ, variance σ^2
IID	Independent, Identically Distributed
FOM	Figure of Merit
MAP	Maximum A Posteriori
pdf	Probability Density Function
VS	Vector Strength
ANN	Artificial Neural Network
ASR	Automatic Speech Recognition
DCT	Discrete Cosine Transform
DL	Deep Learning
DTW	Dynamic Time Warping
HMM	Hidden Markov Model
ML	Machine Learning
NCC	Nearest Cluster Centre
PCA	Principal Component Analysis
RBF	Radial Basis Function
SNN	Spiking Neural Network
SVM	Support Vector Machine
WER	Word Error Rate
AM	Amplitude Modulation
ANF	Auditory Nerve Fibre
AVCN	Anteroventral Cochlear Nucleus
BF	Best Frequency
BM	Basilar Membrane
CF	Characteristic Frequency
ChS	Sustained Chopper
ChT	Transient Chopper
CN	Cochlear Nucleus
DCN	Dorsal Cochlear Nucleus
ECoG	Electrocorticogram
EEG	Electroencephalogram
fMRI	functional Magnetic Resonance Imaging
HSR	High Spontaneous Rate
IC	Inferior Colicullus
IHC	Inner Hair Cell
ISI	Interspike Interval
LowF	Low Frequency
LSR	Low Spontaneous Rate
MEG	Magnetoencephalogram
OHC	Outer Hair Cell
On	Onset
PBU	Pause/Build-Up
PL	Primary-Like
PLN	Primary-Like Notch
PSTH	Post-Stimulus Time Histogram
PVCN	Posteroventral Cochlear Nucleus
RI	Rate by Inversion; Rate/Intensity
SBC	Spherical Bushy Cells
SO	Superior Olivary
TFS	Temporal Fine Structure
tMTF	temporal Modulation Transfer Function
VCV	Vowel-Consonant-Vowel

Introduction & Thesis Outline

Perception — or at least the mechanisms behind it — is a mysterious concept, entangled with consciousness. Its Latin origin *percipere* means 'seize, understand', which already includes the two concepts it is built upon: seizing bits of the external world through our senses, and making sense of it all.

This 'sense' to be made is obtained by association with internal representations, be they concepts or memories. For example, associating a sound to a word is a complex phenomenon that involves transforming the sound waveform into a neural signal, processing it and recognising some abstract features of the sound over a wide range of timescales and frequencies. At the same time, the auditory system is able to extract other useful information such as the voice intonation and associate it with a feeling experienced previously or even integrating visual stimuli (a facial expression) to form a more complete sensorial picture. Neuroscience aims at explaining such phenomena, empowering humans by developing their understanding of our condition and ultimately improving it.

Auditory Neuroscience, specifically, aims at understanding sound related percepts [Plack, 2005; Schnupp *et al.*, 2011]. Contemporary auditory neuroscience focuses on quite a few topics such as tinnitus, source localisation, hearing loss and hearing impairments, or the way the mammalian auditory system is able to segregate sounds. The latter, the so-called Cocktail Party Problem [Simon, 2017], attempts to explain the neurobiological mechanisms leading to the separation of a sum of sounds (what we hear) into different streams of information (what we experience), allowing us to focus on one specific source (what we perceive) such as a speaker in a noisy environment.

To study how the brain might segregate sounds, we need to be able to study its response to complex sounds [Neilans and Dent, 2015]. Hence, researchers need to be able to process neural data encoding complex sounds to tackle these open problems. While we will not work on the Cocktail Party Problem, this requirement of having to deal with data encoding complex and dynamical stimuli motivates the study of the processing of complex sounds such as speech.

Studying the processing of speech is made harder by the lack of appropriate data: obtaining fast brainstem activity with current technologies requires invasive procedures [Herff and Schultz, 2016]. Ethical and technological considerations limit the experiments that can be done in this direction to small mammals, with all the difficulties this entails: time constraints to collect the data, risks of damaging or missing important brain areas, a limited amount of neurons to record from and anaesthetics that change the neural processing. Any

prior information regarding the chances of success in an experiment is thus crucial to be able to design experiments tailored to the questions at hand.

Furthermore, our understanding of neural mechanisms for speech processing is hampered by the inability to create complex models that accurately describe and reflect the behaviours of real neural tissue such as Inferior Colliculus [Steadman, 2015]. Such models, to be useful and integrate a growing body of research and data, need flexibility and interpretability. With this in mind, it should be clear that classical linear models are a limitation that the growing trend of data-driven discoveries cannot keep using. The complexity behind manipulated neural representations is thus best suited to Machine Learning [Kotsiantis *et al.*, 2007] methodologies to assess the suitability of the coding for recognition, which is the topic investigated in this work.

This thesis studies the application of Machine Learning and Automatic Speech Recognition technologies [Young *et al.*, 2006] to spike train data, in an attempt to infer the complexity (such as the number of speakers, accents, phonemes) of the speech dataset on which it is reasonable to expect a good recognition accuracy under the constraints of an experimental procedure. This tool could be used to track down speech processing along the auditory pathway by comparing the data obtained at different functionally important places between the ear and the cortex. Popular animal models in use are the cat and the guinea-pig, both known to hear the range of frequencies used by humans to communicate [Fay and Wilber, 1989]. It is accepted that higher processing capabilities are used by humans in speech recognition tasks, and it is a legitimate question to ask whether animal models or data can provide insight about speech processing. In practice, however, little is known about the actual limitations such animal models entail regarding the peripheral and, even more so, the central auditory system, while many structural similarities motivate their use [Neilans and Dent, 2015].

To study the feasibility of such a tool, we (A) assess the relationship between Machine Learning algorithms and neural coding, and (B) proceed to testing a speech recognition framework on datasets of increasing complexity, as detailed below. These two aspects of the work are split into two parts, always studying spike train representations of sounds. First, the tools are presented and applied to the responses to sinusoidal modulation of pure tones, before moving to a complex dynamical modulation, namely speech. Experimental data is used wherever possible (cat data [Rhode and Greenberg, 1994] in chapter III, guinea-pig data [Steadman, 2015] in chapter VII), simulated data replacing it otherwise [Sumner *et al.*, 2003]. In the process, insight is obtained regarding neural coding of sounds at various places along the Auditory Pathway.

Part A starts with a review chapter on the auditory pathway (I), followed by chapter II that introduces the important Machine Learning algorithms and Data Mining concepts used in this thesis. It then focuses on comparing neural data and Supervised Learning algorithms in a simple sound discrimination task, using cat Cochlear Nucleus data: Chapter III uses and

compares a battery of classification algorithms to infer the modulation frequency from the neural response of amplitude-modulated tones. Chapter IV examines whether the measure of performance used affects the conclusion obtained in the previous chapter.

Part B starts with a review chapter (VI) on Speech Recognition technologies and their application to neural data, paving the way to the next three chapters. Chapters VII, VIII and IX evaluate a speech recognition framework on three speech datasets of increasing complexity (number of speakers, number of phonemes, speakers' accents, dictionary size). In each, a set of parameters is tested, either in the cochlear model used to simulate neural data, or in the processing applied to it, in order to improve the capabilities of the speech recognition framework.

A

Data Mining in the Cochlear Nucleus

I.

Auditory Pathway

This chapter presents the Auditory Pathway and introduces the basic neural mechanisms for demodulation, speech generation and speech perception. These neural mechanisms will be explored in parts A and B, focusing on extracting information from spike train data encoding Amplitude-Modulation and speech, respectively.

I.1 Overview of the Ascending Auditory System

I.1.1 Peripheral Auditory System

Sound is a pressure wave, alternating compressions and rarefactions propagating through an elastic medium, the air. To permit its percept, a biophysical system must transduce it into a neural spiking code amenable to cortical interpretation [Plack, 2005; Schnupp *et al.*, 2011; Pickles, 1988]. The ear, the complex system that performs this transduction shown in figure I.1, is classically divided into three functional parts:

- Outer ear: The sound arrives at the pinna and propagates into the ear canal;
- Middle ear: The eardrum vibrates with the incoming sound, inducing small movements that are transmitted by three tiny bones (Malleus, Incus, Stapes), the last one having a small surface of contact with the cochlea: the oval window;
- Inner ear: The vibrations on the oval window induce a wave on the basilar membrane within the cochlea, which allows mechanosensing organelles to transduce the sound into an electrical signal.

This electrical signal is transmitted to the Cochlear Nucleus (CN) through the Auditory Nerve (AN), body of neurons having long axons to transmit information across distances.

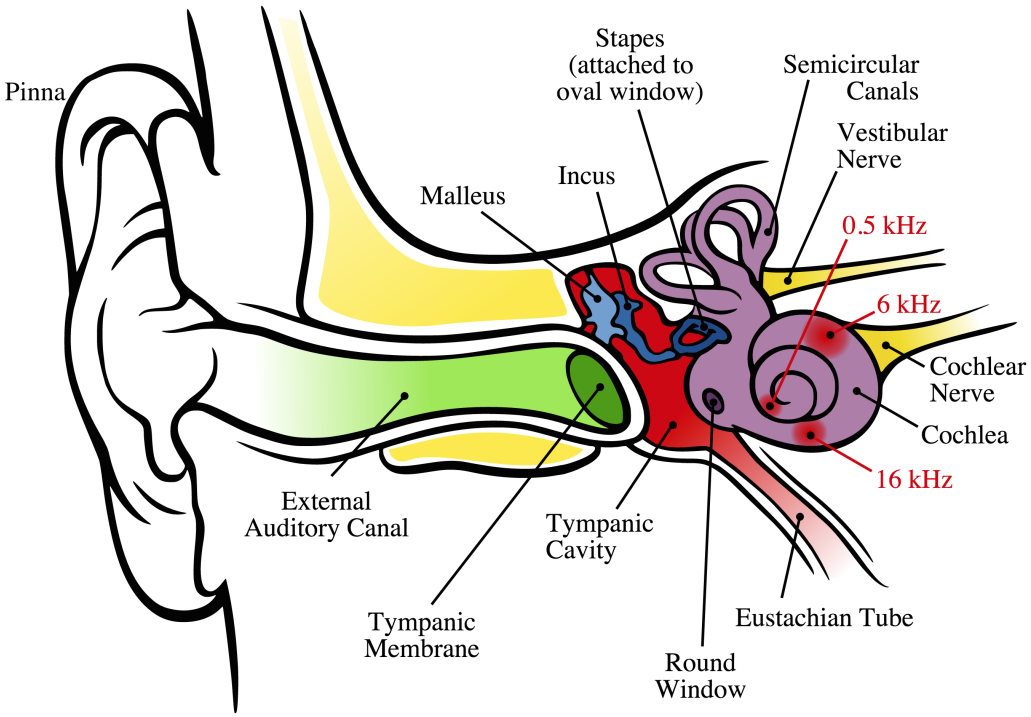


Fig. I.1 Anatomy of the human ear; adapted from [Chittka and Brockmann, 2005].

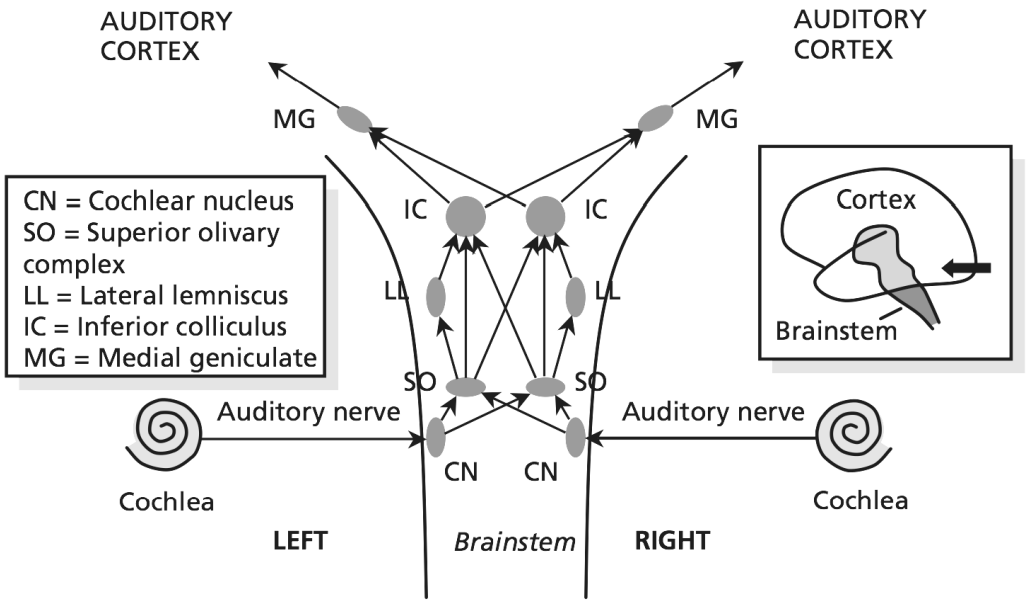


Fig. I.2 Main connections in the ascending auditory pathway, from the cochleae to the auditory cortices, adapted from [Plack, 2005]. The viewpoint is caudorostral, from the back looking towards the front

There, it is processed and sent to other nuclei such as the Superior Olivary Complex or the Inferior Colliculus (IC), as shown in figure I.2. The rest of this section presents in more details the elements of the ear, and the other components forming the Auditory Pathway.

I.1.1.1 Outer Ear: Sound Direction

The pinna is a reflector that efficiently leads sound to the ear canal. Due to its shape, it applies spatially dependent filtering to the sounds. Many animals, such as cats, are able to move theirs to focus more efficiently on listening to sounds coming from different directions without turning their heads, while humans only have weak muscles able to act on the pinna. The ear canal has resonance properties that amplifies sounds at middle frequencies and terminates at the eardrum — or tympanum — a thin membrane about 9 mm in diameter in adult humans [Kandel, 2013].

I.1.1.2 Middle Ear: Transmission

Behind the eardrum, the middle ear is an air-filled space extending from the pharynx through the Eustacian tube, which permits equalisation of the pressure on both sides of the eardrum. The middle ear's primary function is to allow an efficient transfer of acoustic energy from air to fluid by a high pressure-gain (at least 18 fold), mechanistically induced by the reduction of vibratory area, the oval window being about 14 fold smaller than the eardrum.

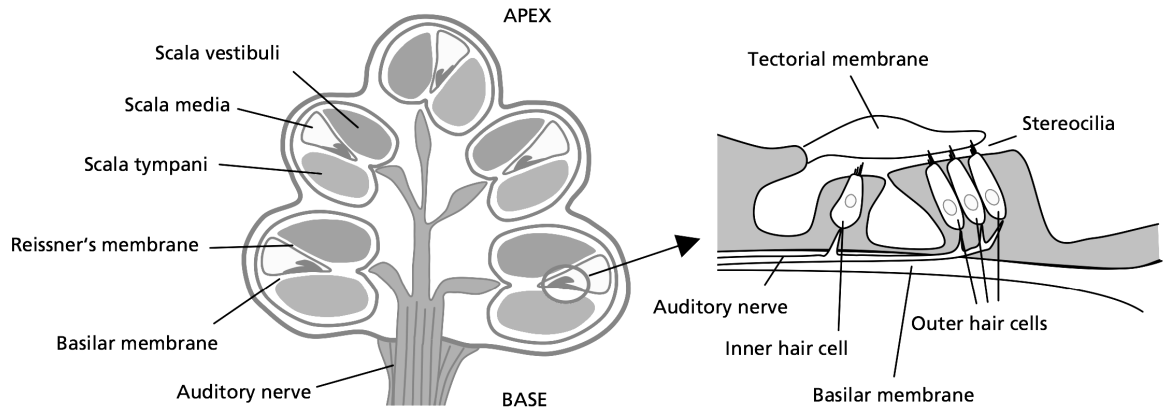


Fig. I.3 Schematic cochlea cross-section, with a close-up on the organ of Corti [Plack, 2005].

I.1.1.3 Inner Ear: Transduction

The cochlea is a complex spiral structure that contains a membrane called the basilar membrane (BM), as shown on figure I.3, and a structure called the organ of Corti attached to it. The motion of the stapes against the oval window induces a movement of the fluid in the cochlea that in turn induces a motion of the sensory-motor hair cells contained in the organ of Corti, attached to the BM. Those hair cells are split in two categories: the inner hair cells (IHC) and outer hair cells (OHC). IHCs are sensory units that sit closer to the centre of the cochlea and form a single row, transducing the sound into neural activity. On the other hand, OHCs are motor units able to amplify the movement of the basilar membrane.

I.1.1.4 Filtering action of the Basilar Membrane

The BM, thanks to the heterogeneity in width and stiffness along its length, vibrates maximally at different places depending on the spectrum of the incoming sound: near the base of the cochlea, where the membrane is narrow and stiff, high frequencies resonate, up to about 20 kHz in young humans [Plack, 2005]. Conversely, the apical end of the membrane is wide and floppy. Low frequencies resonate there, down to about 20 Hz. The frequency provoking a vibration with maximal amplitude at a given point of the membrane is called its Characteristic Frequency CF, or its Best Frequency BF.

Any given point on the BM has to move with its neighbours on a spatial scale characterised by the local stiffness of the membrane, preventing the cochlea from behaving like a perfect Fourier transform to the sound. The physical constraints of the BM lead us to view this forced motion of neighbouring points as being the bandwidth of a band-pass filter. The cochlea thus behaves like a filter-bank of band-pass filters having a fairly constant bandwidth in a logarithmic scale for decreasing BFs, following the decreasing stiffness of the membrane from base to apex. This is developed further in section VI.1 on Biophysical Auditory models.

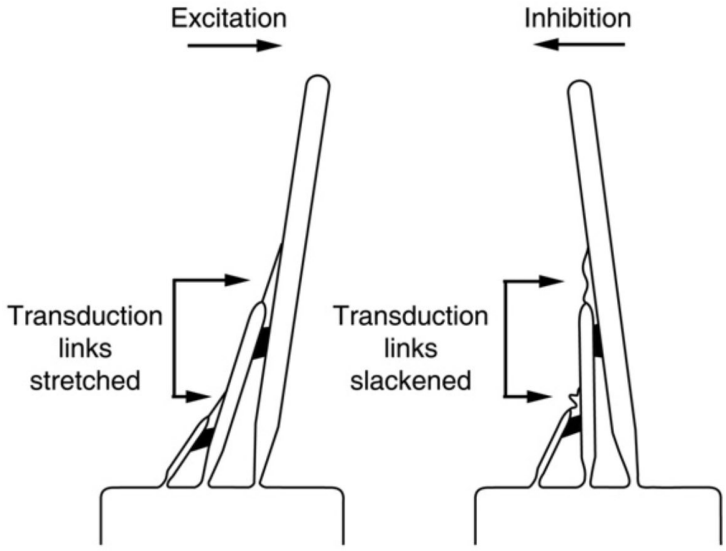


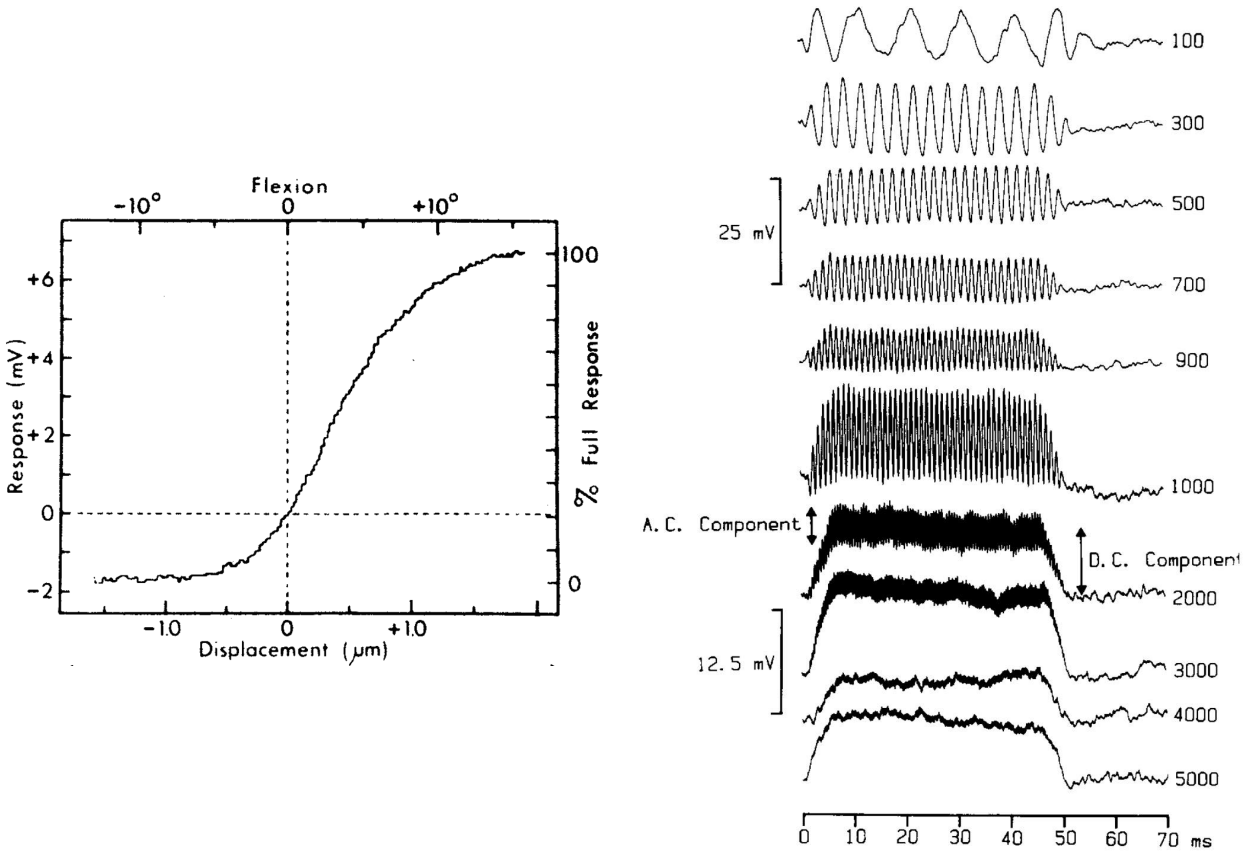
Fig. I.4 [Pickles, 1988] The tip link model for mechanotransduction. Deflection of the bundle in the direction of the tallest stereocilia, which is always the excitatory direction, applies tension to the links and pulls open the mechanotransducer channels. Deflection in the reverse direction takes tension off the links and allows the channels to close. For the purposes of illustration, the deflection of the stereocilia has been massively magnified from that expected in vivo.

I.1.1.5 Biophysics of Inner Hair Cells

The vibration of a specific part of the BM - associated with a specific frequency - induces a shearing movement between the tectorial and basilar membranes (figure I.3). The stereocilia, little organelles sticking out of the tops of IHCs attached to the BM presented in figure I.4, follow this movement. As the stereocilia bend towards the tallest cilia, fine protein fiber strands known as the tip links that connect the stereocilia are stretched and ion channels are opened on top of the IHC. Positively charged potassium ions (K^+) enter the cell, causing a depolarisation of the cell that locks to the sound waveform's phase. This in turn releases a neurotransmitter into the synaptic cleft between the hair cell and a neuron from the auditory nerve (spiral ganglion cells, also called cochlear nerve fiber), causing electrical activity in the form of a spike, also locked to a wave's phase. When the stereocilia are bent to the other direction, the tip links slacken and the channels close. This whole process makes the cochlea behave as a bank of overlapping band-pass filters that encodes precise temporal information into the spike-timing of nerve fibre, which will be vital in the analysis presented in chapter III.

I.1.1.6 Rectification and Low-Pass Filtering by Inner Hair Cells

IHCs have nonlinear responses with respect to the angle formed by their stereocilia when deflected, and a natural time-scale to return to their resting state. The consequences of these properties are naturally expressed in terms of signal processing elements:



A) B)

Fig. I.5 *A)* Membrane voltage of an inner hair cell, plotted as a function of the displacement of the cilia in the excitatory direction (positive displacement) or the inhibitory displacement (negative displacement) [Hudspeth and Corey, 1977]. *B)* Membrane potential of a single inner hair cell, recorded over 70 ms after a 80 dB SPL pure tone is played to the guinea-pig, at the indicated frequencies (in Hz) to the right of each trace. As the frequency increases, the sinusoidal component is converted into a steady depolarising component [Palmer and Russell, 1986].

- **Saturation**: The response is close to linear for small deflections of the cilia in the excitatory direction, and it saturates when the displacement is large (around 1.5 μm). This way, a broad range of deflection magnitudes can be mapped into a sensibly smaller range of membrane potential, as seen in the right part of figure I.5A.
- **Rectification**: In the inhibitory direction, a much smaller displacement (around 0.5 μm) reduces the current in the cell to its minimum. This asymmetry resembles the function $x \mapsto \max(x, 0)$, which is usually referred to as the half-wave rectifier, taking its name from the homonym electrical engine converting alternating currents (AC) into direct current (DC) by stopping the flow in one direction with a diode (left plot of figure I.5A).

- **Low-pass filtering**: When a pure tone is played to an animal, one would expect the membrane voltage in an IHC having this frequency as its best frequency to reflect the sinusoidal movement of the stereocilia; this is indeed the case at low frequencies. However, as the sound frequency increases into the kilohertz range, individual cycles of the vibration become less visible and are replaced by a continuous depolarisation that lasts as long as the stimulus. See figure I.5B.

The importance of this nonlinearity in the amplitude-modulation decoding ability of the cochlea is stressed in subsection I.2.1. Before focusing on the decoding properties of the Auditory Pathway, let us finish the presentation of its main components, now entering the CN and giving a fair account of the difference between the cell types that make it up.

I.1.2 Cochlear Nucleus Cytoarchitecture

I.1.2.1 Cochlear Nucleus Taxonomy

The study of the physiology of the Cochlear Nucleus started at least in the late fifties, where the morphology of CN neurons was related to their physiological response properties to specific features of sound [Rhode and Greenberg, 1992]. The CN is the first nucleus to process auditory information. Until then, the spatial arrangement of the IHC reflects the frequencies they encode, which can be rephrased by saying that sound is tonotopically transduced onto the Auditory Nerve Fiber (ANF). Each ANF branches as it enters the CN, towards the different components of the nucleus: The anteroventral, posteroventral and dorsal Cochlear Nucleus (respectively AVCN, PVCN, DCN) are three branches found in various species, each having specific subpopulations of cell types. The CN forms a network of neurons that project to other nuclei to perform different computations such as source localisation, as is shown in the next subsection.

Each CN cell type has a unique pattern of response to sound, consistent with the idea that they are involved in different aspects of the sound analysis. There are at least 4 ways to characterise a cell, each having its own name:

- Their pure tone response class, or Post-Stimulus Time Histogram (PSTH). See figure I.6 for example units (such as Primary-Like or Choppers, defined in this section);
- Their morphology: innervation and projection, cell shape, number and type of synapse (such as stellate or bushy cells);
- The shape of their receptive field (pure-tone tuning, classes I to VI);
- Their intrinsic currents: biophysical behaviour under current or voltage clamp and corresponding ion channels (type I or II with their subclasses).

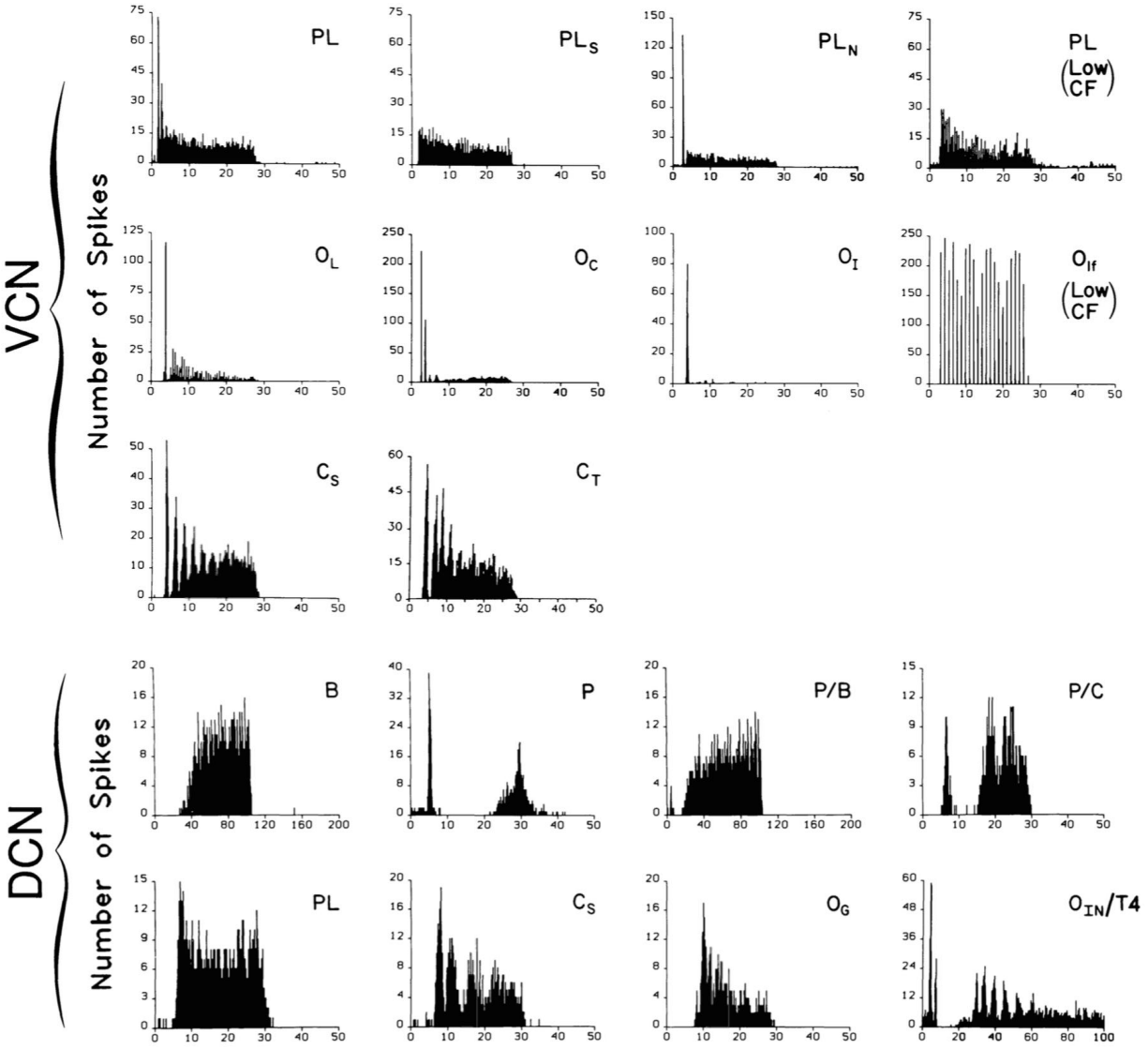


Fig. I.6 [Rhode and Greenberg, 1992] Response diversity of cochlear nucleus units in cats. Representative temporal response patterns (PSTH, or poststimulus-time histograms) for the major physiological unit classes in the ventral and dorsal cochlear nuclei of the cat. The PSTHs were computed from responses to brief tone bursts at the unit CF. For most cells the stimulus duration was 25 ms, with each stimulus repeated 250 times, once every 105 ms. For some DCN units the stimulus duration was lengthened to 100 ms due to the buildup (buildup and pauser/buildup units) or complexity (O_{IN}/T4, IN-inhibitory) of response. In these instances the stimuli were repeated once every 400 ms. Stimulus sound pressure level was generally 60 dB, approximately 30-40 dB above unit rate threshold. Histogram bin-width was micro-second. See list of abbreviations for key to unit type classification in I.1.2.2.

The elegant phrasing of [Joris and Smith, 2008] stresses how well neurons fall into categories: 'Biological taxonomy is always fraught with splitting vs. lumping difficulties. Kirsten Osen's morphological parcellation of the cochlear nucleus [Osen, 1969] was a landmark achievement because it hit exactly the right level along the splitter-lumper dimension.'

Thanks to the exceptional taxonomy obtained by studying the CN and relating morphology with other properties, the pure tone response is strongly predictive of the intrinsic currents that will be found in a cell, the intracellular responses to injected electrical currents, the morphology of the cell and the patterns of innervation of the cell by ANFs. Understanding how each mechanism contributes to the electrical activity of the cells is a major goal of contemporary auditory neuroscience. A complete discussion would be beyond the scope of this work, but providing a flavour of the differences mentioned and their consequences is informative:

- **Innervation**: The number of afferent Auditory Nerve Fibres and their spatial summation (the number of synapses and their proximity to the soma of the postsynaptic neurone) play a role in how much of the regularity of the afferent firing is kept to the output. On one side, a single massive synapse attached to the soma may produce a post-synaptic action potential for each spike train of the afferent ANF. Calyces of Held, for example, behave this way; these large synapses cover about half of the post-synaptic neuron's soma [Nakamura and Cramer, 2011]. Primary-like units have a small number of AN inputs. At the other extreme, if many smaller synapses coming from different ANFs are placed far from the soma, a more regular firing pattern is expected. Octopus cells, often thought of as being synchrony detectors, average in cats around 60 inputs [Oertel *et al.*, 2000]. Using the law of large numbers to understand this phenomenon by modelling these units as linear functions of their inputs, this conclusion seems natural: let us model the ANFs as independent random variables identically distributed, say according to some standard Gaussian distribution $\mathcal{N}(0,1)$ for simplicity, and model a unit i as applying a linear sum $f_i : (x_1, x_2, \dots) \mapsto \sum_k \alpha_{k,i} x_i$ over part of the ANFs (implicitly assuming only a few coefficients are positive, different for each unit). The previous descriptions then reflect how averaging independent random variables reduces the variance without affecting the mean. For example with the two extremes previously given, f_1 copying its first input and f_2 averaging its first n inputs, the reduction in variance is clear:

$$f_1(X_1, \dots) = X_1 \sim \mathcal{N}(0,1)$$

$$f_2(X_1, \dots) = \frac{1}{n}\sum_{k=1}^{n} X_k \sim \mathcal{N}(0, 1/n).$$

  A computational model [Rothman *et al.*, 1993] shows that, by varying the number of AN and their strength, the full range of response types seen in VCN bushy cells are reproduced, showing the importance of these two parameters in this nucleus' units.

- **Intrinsic currents**: Some of the main characteristics of ionic currents are their speed (the time-scale at which channels open or close, ranging from milliseconds (fast) to hundreds of milliseconds (slow)), their threshold (high if they require a strong difference between the cells' outside and inside ionic concentrations to open), and their inactivation capacity (whether or not a unit gets inactivated, and how fast it gets inactivated). In [Rothman and Manis, 2003b], the kinetic behaviour of ventral CN units is detailed, revealing the dynamics of three distinct macroscopic potassium currents: a fast transient current (I_A), a slow-inactivating low-threshold current (I_{LT}) and a non-inactivating high-threshold current (I_{HT}). A computational model [Rothman and Manis, 2003a] then shows that the different potassium currents found in CN units shape an important part of the cell's electrical activity, by setting time constants and the resulting firing thresholds. This simulation results indicate that those different potassium currents are responsible for different electrical properties of the cell: I_{HT} functions to repolarise the membrane during an action potential, I_A modulate the rate of repetitive firing, and I_{LT} is found to be responsible for the phasic discharge pattern observed in bushy cells. The temporal integration of stimuli — or temporal summation — reflects how a cell behaves as a capacitor, and is a function of the time-scale at which ion channels change state, with a dynamical impact well-explained by Bifurcation theory [Izhikevich, 2007].

We now present the unit types that are encountered in chapter III.

I.1.2.2 Cell Types in the Cochlear Nucleus

In the literature, different names sometimes refer to the same cell types, because of naming conventions that turned out to be equivalent thanks to the exquisite taxonomy of the CN mentioned earlier. Descriptions in this subsection are mainly based on [Shepherd, 2004], which is very rich in details and on [Pickles, 1988], but uses the classification from [Blackburn and Sachs, 1989] in order to facilitate the reading of chapter III. This classification is based on the different Post-Stimulus Time Histograms PSTH shapes obtained in response to pure tones and on the physiological response properties of the units, while other names refer to their anatomy. We add, when appropriate, a short description of the membrane properties and of the flow of ions when a cell is subject to *in vitro* injection of hyperpolarising and depolarising currents, since these properties form the fundamental differences between cell types that will make them react differently to stimuli.

PL (Primary-Like): Primary-like units take their name from their physiological resemblance with the responses of ANFs. The powerful end-bulb synapses that innervate bushy cells, mainly located on the soma, make it possible for a small number (sometimes only one) ANF to generate an action potential in the afferent neuron. Another name coming from their

morphology is bushy cells, due to their bush-like dendritic structure; they are thus often referred to as Spherical Bushy Cells (SBC), due to their soma's spherical shape. The electrical behaviour of primary-like units is produced by a low-threshold potassium conductance that is strongly activated by depolarisation, producing a membrane rectification, thus preventing temporal summation of inputs.

PBU (Pause/Build-Up): The DCN has a laminar structure, of which fusiform (or pyramidal) cells form the second layer. Most commonly, those cells show two types of response: a pauser response, where the initial burst of activity is followed by an inhibitory pause, or a build-up response where the response gradually increases during presentation of the stimulus. Both response types reflect a transient potassium conductance that makes the cell less likely to fire for about 10 ms. The pause is produced by activation of a transient potassium conductance that is normally inactivated at rest. If the cell is hyperpolarised, inactivation is removed, so that the transient conductance can be activated by a subsequent depolarisation. Unlike most CN units, these units are substantially influenced by inputs other than the cochlear, particularly local inhibition.

PLN (Primary-Like with Notch): PLN units show a fairly similar PSTH as PL units, except for a small notch after the initial burst that reflects the refractory period following a very sharp onset peak in the PSTH. Their anatomical name is Globular Bushy Cells (GBC). SBCs and GBCs differ in their location, their projections and the number of afferent ANFs (higher in PLNs, making them better at coincidence detection), and have anatomical differences that give bushy cells their names.

ChS (Sustained Chopper): Chopper units such as T-stellate cells tend to fire at regular intervals during a sustained tone burst at a rate unrelated to the period of the stimulus waveform, a pattern called chopping. They are also called T-Multipolar cells and project to the Trapezoidal Body. Their regularity arises from the integration of many inputs and from having longer time constants, since they have no low threshold potassium currents [Rothman and Manis, 2003a].

ChT (Transient Chopper): Sustained choppers have a fairly sustained local coefficient of variation CV, while ChT show an increase over time of this measure. Two hypotheses underlie this behaviour: a delayed inhibitory input will come from another neuron, or ChS have more inputs. Thus, a common technique to distinguish chopper subgroups is that of regularity analysis. If their coefficient of variation during the latter half of the response to a short (25-50 ms) tone burst is above a fixed threshold (0.3), they are said to be transient.

On (Onset): An onset response signals when an important change in the stimulus is observed. Octopus cells are large cells with a complex shape that receive a very large number of afferent ANFs and are driven only when a large number of the relatively small excita-

tory postsynaptic potentials arrive together, making them good coincidence detectors. They respond to tones over a very wide range of frequencies. D-multipolar (or D-stellate) are a different morphological cell type which also receive many ANFs and respond mainly at the onset of sounds and project to the Dorsal Acoustic Striae. Octopus cells also have a cocktail of intrinsic currents — I_{LT} and mixed cation current I_h, which lowers their effective input impedance. This further helps them fire very precisely and with a short latency at the onset of a stimulus.

LowF (Low Frequency): This category is used when the neuron's response is dominated by phase-locking properties, making it hard to evaluate in which category it falls. Some techniques based on randomised inputs allow further classification.

Unu & UNC (Unusual & Unclassified): Categories used for units that don't fit to any of the above. Comments are sometimes given regarding the results obtained for those units, although for readability they were removed from all figures in this thesis.

The classification of CN cells is mostly based on subjective criteria, but their diversity in response pattern and projection onto specific brainstems known to perform different tasks is a strong evidence for their functional diversity.

I.1.3 The Other Auditory Brainstem Nuclei

The brainstem nuclei are groups of neurons, mirrored in each hemisphere, that perform transmission and information processing along the auditory pathway. Connections between two nuclei are generally classified as belonging to either the ascending pathway or the descending pathway, depending on whether the information is going 'up' towards the cortex, or 'down' towards the CN. The ascending auditory pathway, whose main connections are schematically presented in figure I.2, contains nuclei that have been associated with different tasks such as calculating an interaural time difference (ITD), which requires binaural information. Evidence suggests that the descending pathway plays an important role in setting cochlear gain, thus impacting on the representation of sounds at the level of the cochlea [Jennings *et al.*, 2011] and having implications in speech recognition in noise [Clark *et al.*, 2012]. Secondary direct connections are missing from figure I.2, including a connection between cochlear nuclei, evidenced in [Davis, 2005] in cats and in [Arnott *et al.*, 2004; Shore *et al.*, 2003] in guinea-pigs.

The first processing station, as discussed, is the Cochlear Nucleus. The signal is then passed to the Superior Olivary (SO) complex, the first brainstem to receive binaural information and thought to be involved in sound localisation. This projects to the Lateral Lemniscus, and both project to the Inferior Colliculus (IC), known to receive inputs from multiple brainstem nuclei. Beside cat CN data discussed in chapter III, the other experimental data used in this thesis is guinea-pig IC data in chapter VII. The auditory system is

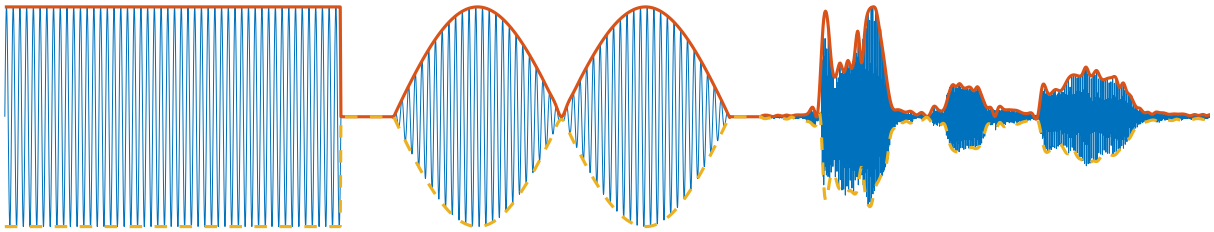


Fig. I.7 Representation of a pure tone (left), an amplitude-modulated tone (middle) and a speech sample (right) separated by short silences. Their respective higher and lower envelopes are represented in red and dashed yellow.

interconnected, with parallel cues transmitting different types of information. The information is passed to the auditory cortex via the medial geniculate body of the thalamus. The primary auditory cortex is located at the top of the temporal lobe, mostly hidden within the Sylvian fissure in humans.

I.2 Envelope Extraction & Speech Perception

Speech is a vocalised form of communication expressed through the articulation of sounds, allowing humans to express their thoughts and feelings using a large vocabulary, typically containing more than a thousand words. Speech sounds are characterised by the locations of spectral peaks and dips, fluctuating rapidly. We are able to detect fluctuations in the level of a sound up to rates of 1000 times a second[1] [Bacon and Viemeister, 1985]. This contrasts with the ∼100 Hz temporal resolution of the flicker-detection ability of the visual system [Plack, 2005].

The same way a pure tone is perceived as constant, most of the speech information is contained in the slow modulations of spectral envelopes of a speech waveform. See the example waveform's envelope in figure I.7, where the constant envelope of a pure tone contrasts with the temporal envelope of a sinusoid modulated in amplitude and a speech sample. Hence, Amplitude-Modulation (AM) is an important feature of most natural acoustic signals. The available evidence suggests that specialised neural mechanisms exist to extract AM information [Joris *et al.*, 2004]. In this section we present the neural emergence of demodulation in the auditory nerve fibre, going towards the processing of more complex envelope. The basics of speech generation are then introduced, finally ending with the neural basis of speech perception.

[1] Between approximately 18 Hz and 1000 Hz, a percept called 'roughness' is experienced, where fluctuations are too fast to be counted, but slow enough to be perceived [Joris *et al.*, 2004]. The fastest drum players are able to strike at 20 Hz, thus faster than most people could keep track of the number of strikes.

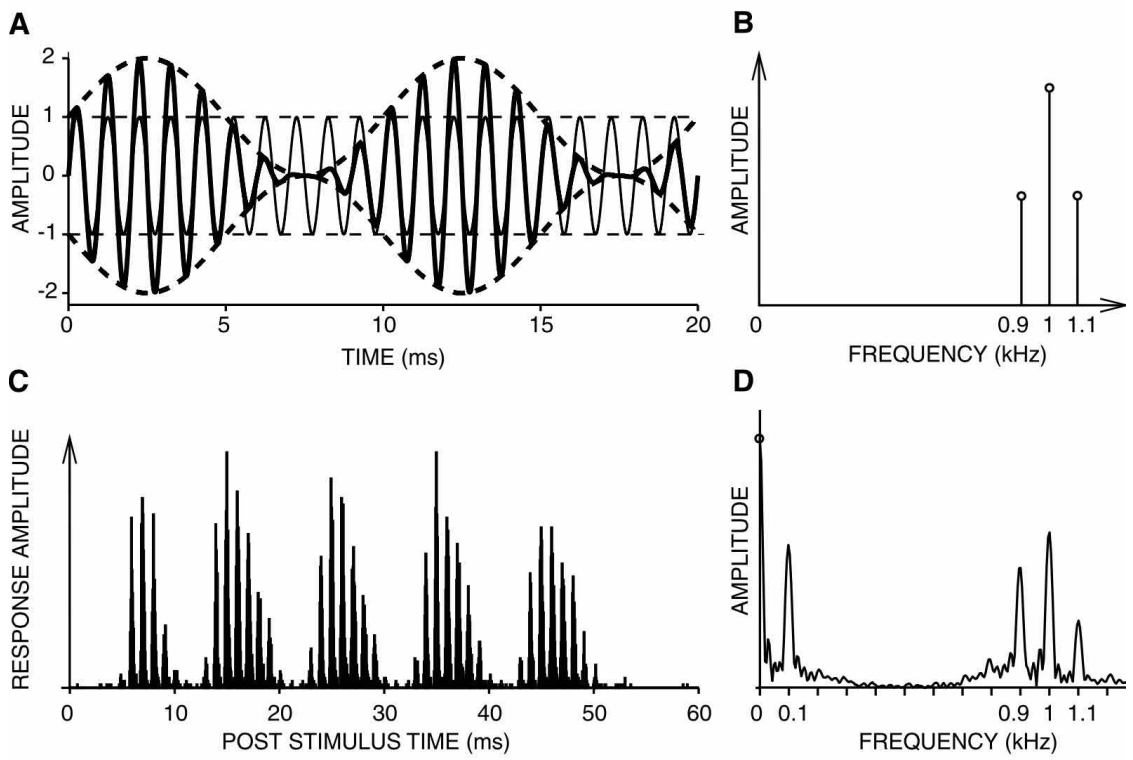


Fig. I.8 From [Joris *et al.*, 2004]. (A) superimposed waveforms of an unmodulated 1,000 Hz tone (thin line, $f_c = 1000$ Hz) and the same tone sinusoidally amplitude modulated (AM) (thick line, $f_m = 100$ Hz) at 100% with a modulation frequency of 100 Hz. Dashed lines indicate the envelope. The amplitude is referenced to the peak amplitude of the unmodulated tone. (B) idealised spectrum of the AM tone in A. At 100% modulation, the amplitude of the sidebands is half that of the carrier, i.e., a difference of 6 dB. (C) average response in the form of a post-stimulus time histogram (PSTH) of a nerve fiber to the signal shown in A (stimulus duration, 50 ms). (D) spectrum of the PSTH in C. The components at carrier frequency (f_c) and modulation frequency (f_m) indicate that there is phase-locking to the fine-structure of the stimulus waveform. The component at f_m is prominently present in the response but is absent in the stimulus (B), as a consequence of the nonlinearity of the response. The small circle on the ordinate indicates the average firing rate.

I.2.1 Amplitude-Modulation Decoding

I.2.1.1 Role of a Nonlinearity

The difficulty of understanding sound percepts already appears in the following paradoxically simple formula: for all $t \in \mathbb{R}$ and for two frequencies $f_m > 0$ and $f_c > 0$ with $f_c > f_m$,

$$[1+2m\sin(2\pi f_m t)]\sin(2\pi f_c t) = \sin(2\pi f_c t) - m\cos(2\pi(f_c+f_m)t) + m\cos(2\pi(f_c-f_m)t). \quad \text{(I.1)}$$

It is simple because it can easily be proven using classical trigonometric identities. It is paradoxical because, assuming $f_c \gg f_m$, the modulation is perceptually evident to our

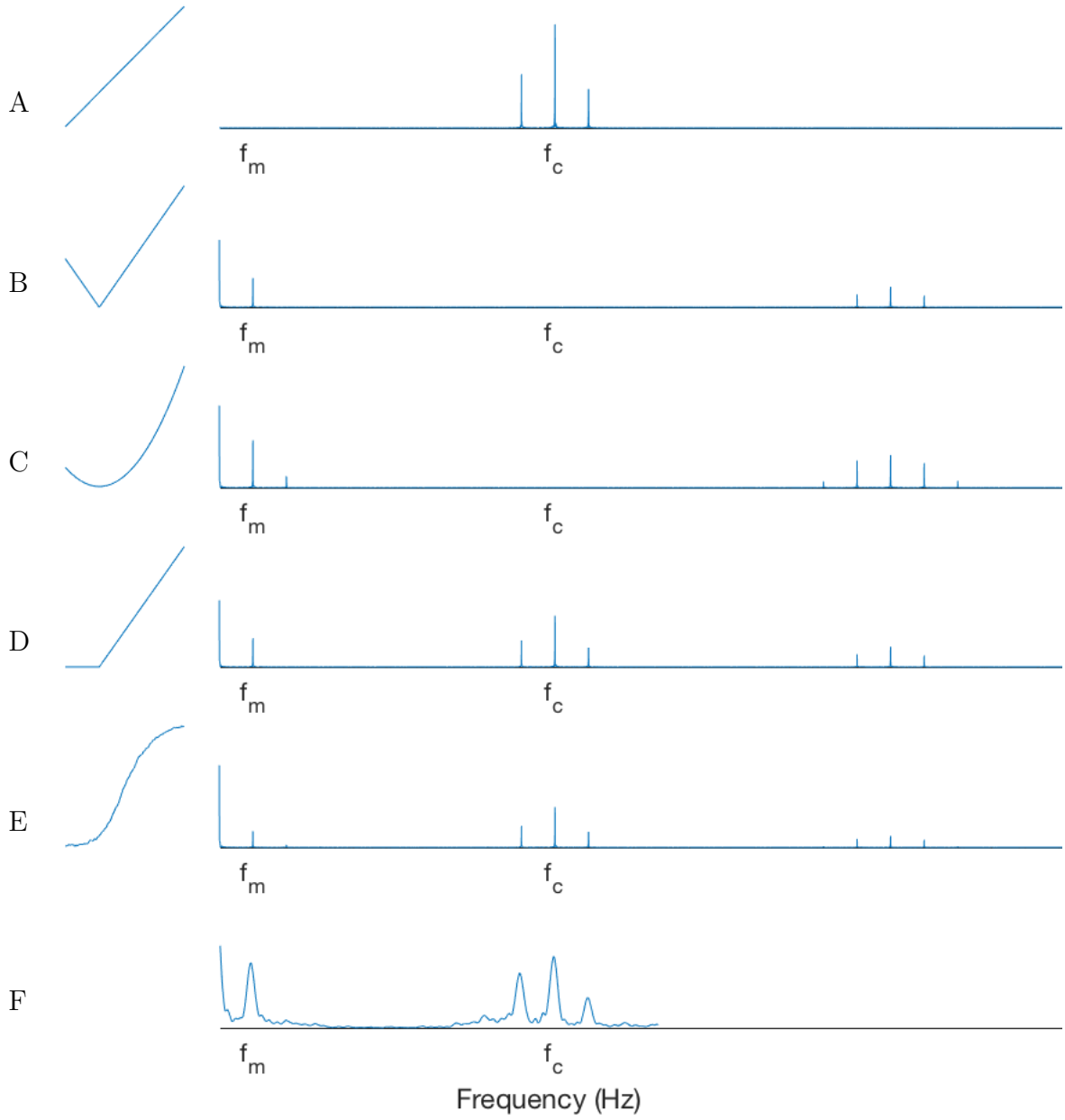


Fig. I.9 Absolute values of the FFT after applying different nonlinear functions f to the AM tone defined in equation I.1, with carrier frequency f_c and modulation frequency f_m, and comparison with figure I.8D. (A) $f(x) = x$, (B) $f(x) = |x|$, (C) $f(x) = x^2$, (D) $f(x) = \max(x, 0)$, (E) the nonlinear function figure I.5, extracted from the figure with Matlab. (F) the actual spectrum of PSTH of figure I.8C, extracted from the figure with Matlab. For readability, only f_m and f_c labels are provided on the x-axis. On the right side, the functions applied to the sinusoid varying between -2 and 2 are plotted, fox x values between -2 and 5.

ears when this function represents a sound waveform, and visually evident when represented on a graph (thick line on figures I.7 and I.8A), leading to a perceptually clear periodicity with frequency f_m. However, and this is evident from the right-hand side of formula I.1, there is no spectral energy at f_m; this percept arises from the two terms centred around f_c (figure I.8B). With this crucial observation in mind, it should be clear that our perception of sounds cannot be understood by directly reading the Fourier spectrum of the incoming waveform. The key to the AM extraction is in the nonlinearity of the transduction process.

Applying a nonlinearity brings up the modulation frequency f_m into the energy spectrum, in different ways for different non-linearities. See figure I.9A-E to visualise the effect of selected nonlinearities on the spectrum and compare it with I.9F, which was extracted from I.8D. The half-wave rectifier (figure I.9D) is a good analyser (in this case), because f_c and f_m both appear clearly, and because the spectrum it brings is remarkably similar to the spectrum obtained from some real ANF data (figures I.8D and I.9F). This is consistent with the fact that inner hair cells' non-linearity (I.9E, extracted from figure I.5) resemble half-wave rectification (I.9D); surprisingly, the spectrum obtained using a half-wave rectifier is even visually better than the result obtained using the function seen in figure I.5. This low-pass filtering from temporal integration has as a consequence, that as a signal ascends the auditory system, the ability to encode high frequencies decreases, which tends to increase, by redundancy, the available information on the demodulated modulation frequency but not the carriers.

Available evidence suggests there are neural mechanisms to extract AM information [Joris *et al.*, 2004]: ascending the auditory pathway, the neural code containing AM tuning changes from a temporal coding (i.e. synchronisation of spiking activity, the precise timing of neurons' activation being crucial) to a rate coding (i.e. only the average firing rate of a neuron contains information). After the auditory nerve fibres, the AM response is encoded differently by different unit types from the Cochlear Nucleus. This is presented there using the Vector Strength as measure, and is studied in chapter III in a more assumption-free approach.

I.2.1.2 Vector Strength

The Vector Strength (also referred to as synchronization coefficient [Rhode and Greenberg, 1994] or order parameter [Kuramoto, 1975]) is a common measure in Auditory Neuroscience, due to the well-known temporal precision of the discharge patterns in the auditory nerve fiber and higher nuclei. Let the spike train $\{t_1, \ldots, t_n\}$ be the response to an amplitude-modulated tone, with respective phase $\{\phi_1, \ldots, \phi_n\}$ with respect to the modulation. Its vector strength is defined as

$$\rho := \frac{1}{n} \left| \sum_{k=1}^{n} e^{i\phi_k} \right|, \tag{I.2}$$

which is independent of the initial phase θ_0 of the waveform. See figure I.10 for a representation of the vector strength computation on three short spike trains recorded from the same CN neuron under three different AM tones (with a modulation frequency of 150, 350 and 550 Hz, respectively). Vector Strength is fully described in [Rhode and Greenberg, 1994; Hemmen, 2013][2].

The Vector Strength makes an implicit assumption: it requires to know the actual modulation frequency being used, which is not directly available to the neurons. Hence, rather

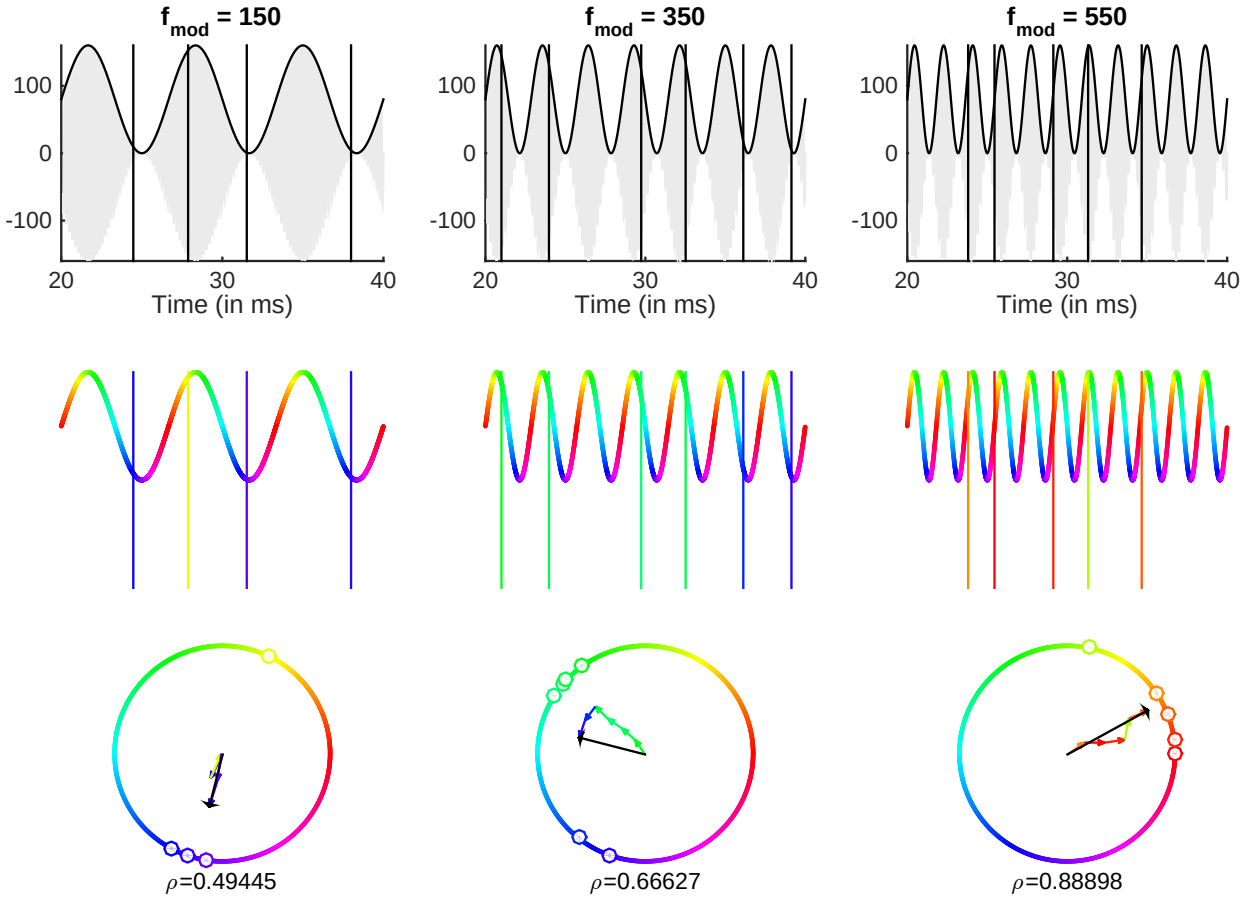


Fig. I.10 Representation of the vector strength calculation on 3 spike trains from an Onset unit, given the periodical input (grey AM tone, with modulation frequencies of 150, 350, 550 Hz): each spike is associated a phase ϕ_k (second row, phase-coloured), and the vector strength ρ is the length of the black arrow on the third row, using formula I.2.

[2]It seems that the first use in neurobiology of the Vector Strength is due to Goldberg and Brown [Goldberg and Brown, 1969], but the first mathematical description of the notion of Vector Strength dates back at least to von Mises' 1918 article 'On the integer-valuedness of atomic weights and related issues' [von Mises, 1928; Siegmund-Schultze, 2006]. See [Hemmen, 2013] for a recent mathematical depiction of the properties of the vector strength and [Siegmund-Schultze, 2006] for a discussion on the historical Pólya and von Mises controversy on the emergence of various important notions around the central limit theorem on the circle (law of (positive) atomic weights modulo 1) before modern Probability theory. In particular, on page 477 of [von Mises, 1928], the reader finds a visual representation of the Vector Strength.

than relying on a measure that assumes the knowledge of the actual modulation frequency used, a Data Mining approach frees itself from such information. This is done in chapter III.

I.2.1.3 Amplitude-Modulation in the Cochlear Nucleus

As the first locus of integrative processing in the auditory pathway, the Cochlear Nucleus is the birthplace of a rich diversification of response patterns beyond the auditory nerve. It seems natural that this diversity would be related to the functional roles played by each unit type. It is at least clear that, through an increased synchrony with the envelope many of them encode AM information, which is an important feature for sound percepts. This is usually evidenced by calculating a function[3] of the Vector Strength called the temporal Modulation Transfer Function (tMTF) of the neural response.

Based on vector strength calculations, neural Modulation Transfer Functions of sustained chopper units are low-pass for stimulus levels below 20 dB (above their threshold) and narrowly tuned band-pass functions for levels above this (see figure I.11). This holds true to different extents for most CN unit types, the sharpest peaks being found in chopper units responses [Frisina *et al.*, 1990; Hewitt *et al.*, 1992; Rhode and Greenberg, 1994]. In [Frisina *et al.*, 1990], the order obtained from the gain of CN units responses to AM sounds in gerbils is

$$\text{On} > \text{Ch} > \text{PLN} > \text{PL} > \text{ANF}$$

showing that regularly spiking units enhance modulation gain.

On top of this, it was shown that CN units had wider dynamic ranges than ANFs for AM encoding [Hewitt *et al.*, 1992]. This phenomenon was also observed on other types of CN units, both in quiet (at moderate to high sound levels) and in white noise with low signal-to-noise ratios [Rhode and Greenberg, 1994]. The rank they found using a synchronicity analysis for the units' ability to phase-lock to high modulation frequencies is the following:

$$\text{ANF} > \text{PLN} > \text{PL} = \text{On} > \text{Ch} > \text{PBU}$$

Thus, under many conditions, CN neurons preferentially synchronise and enhance a best modulation frequency from the sound signal, this peak changing with with sound level, but this enhancement is not given by a unit's phase-locking capability.

The way in which chopper cells temporally encode envelopes has been the subject of several modelling studies. A simple integrate-and-fire approach [Hewitt *et al.*, 1992] models the postsynaptic response as the linear summation of the many nerve fibres inputs, to which a low-pass first order Butterworth filter is applied, a time-varying threshold then generating action potentials. More biophysically complete Hudgkin-Huxley type models [Banks and Sachs, 1991; Wang and Sachs, 1995] also reproduce the linear subthreshold current-voltage curve of stellate cells, successfully replicating their responses to pure tone at Characteristic

[3] The modulation gain (in dB) at a given frequency is defined as $20\log_{10}(r_h/r_s)$ where r_h and r_s are the vector strengths of the response period histogram and the input stimulus, respectively. A Modulation Transfer Function is defined as the modulation gain for each frequency band defining the histograms in use.

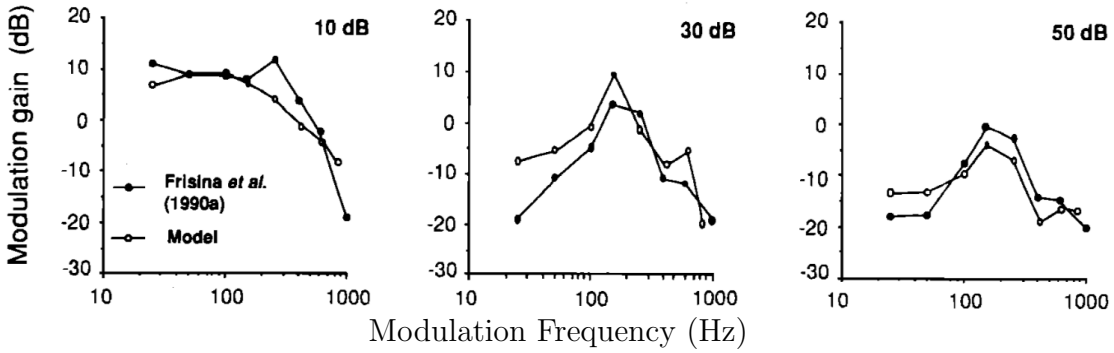


Fig. I.11 Model (open circles) and neural (closed circle) modulation transfer functions of a Sustained Chopper unit for three stimulus input (35% AM-signals) levels (10, 30 and 50 dB above unit threshold). Adapted from [Hewitt *et al.*, 1992].

Frequency, the Vector Strength in response to amplitude-modulated tones and the responses to other periodic stimuli. The theory of nonlinear dynamical systems also explains temporal patterns in the response of stellate cells as being stable regions in parameter space called Arnold tongues [Laudanski *et al.*, 2010].

I.2.2 Speech Generation

The possible shapes of the mouth, tongue and vocal tract give rise to a wide range of sounds an individual can make. Human speech is mostly produced with pulmonary pressure provided by the lungs which creates phonation in the glottis in the larynx that is then modified by the vocal tract into different vowels and consonants; see figure I.12 for two pictures showing different vocal folds configurations for speaking or breathing, respectively. When air is expelled from the lungs through the glottis, a pressure drop across the larynx occurs. When this drop becomes large enough, and if they are close enough, the vocal folds start to oscillate. It is common to think of speech generation as being the product of two phenomena: a 'source' and a 'filter', the production of quasi-periodic vibrations by the vocal folds making an original sound source, and the articulators (tongue, lips, teeth, velum, etc...) then modifying its spectrum as shown in figure I.13.

Much of the speech information is carried by the waveform's slow fluctuations, such as the waveform's envelope, the smooth curves shown in figure I.7 outlining the extremes of the rapidly fluctuating waveform. A simple signal processing experiment confirms this: removing the fine structure of a speech signal does not change much how understandable the message is[4] in a quiet environment. More accurately, the temporal envelopes extracted by the auditory system correspond to the slow fluctuations within each cochlear filter's narrow frequency band. On the other hand, the temporal fine structure, the rapid oscillations within

[4]Listen to https://auditoryneuroscience.com/topics/speech-modulated-signal to be convinced that speech is mostly conveyed by the envelopes.

the envelopes of each band, appear to contain useful information for recognition of speech in noise, where patterns of phase locking appear to be more robust than firing rates [Moon and Hong, 2014].

The same process gives birth to our perception of hearing a constant sound when a continuous pure tone is played, rather than perceiving the variations of the waveform. But the auditory system does not seem to proceed to a direct envelope extraction, as shown in I.2.3. Instead, it collects other information and processes them to accomplish different goals such as speech recognition in noise, source localisation or source segregation.

Many acoustic features are important for speech decoding and are understood as requiring separate parallel processing in the auditory system. Vowels and consonants, for example, are quite different: The prominent bands of frequency that determine the phonetic quality of a vowel, its formants, dominate both the acoustic properties of the sound and the neural responses, whereas consonants vary significantly more quickly, being characterised by either formant transitions, or silences bordered by transients in sound energy [Young, 2008]. To analyse the coding of speech, it is convenient to be able to parameterise the stimuli and study the responses under simple changes. Some important features of speech [Abrams, 2008] are:

- **Periodicity**: regular temporal fluctuations in the speech signal between 50 and 500 Hz;
- **Formant structure**: series of discrete peaks in the frequency spectrum of speech that are the result of an interaction between the frequency of vibration of the vocal folds and the resonances within a speaker's vocal tract;
- **Frequency transitions**: modulation of the fundamental frequency providing information about the intent and emotional state of the speaker;
- **Acoustic onset/offset**: spectral and temporal features present at the beginning and end (the initial or final ∼40 ms) of speech sounds;
- **Speech envelope**: temporal fluctuations in the speech signal between 2 and 50 Hz.

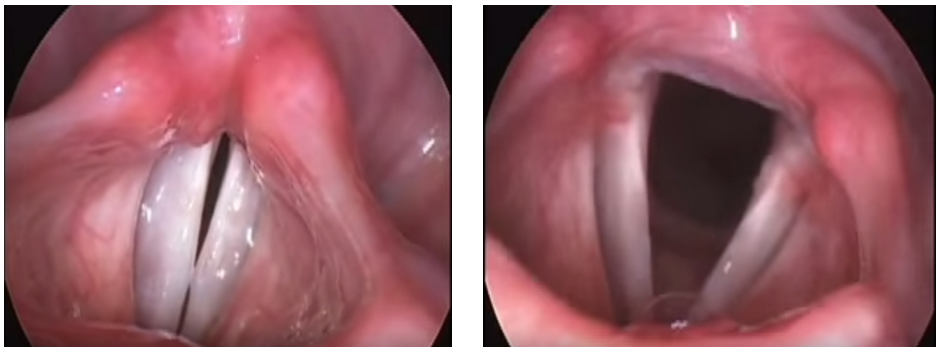


Fig. I.12 Vocal folds in action: oscillating during speech (left), opening during inhalation (right). From https://www.youtube.com/watch?v=v9Wdf-RwLcs

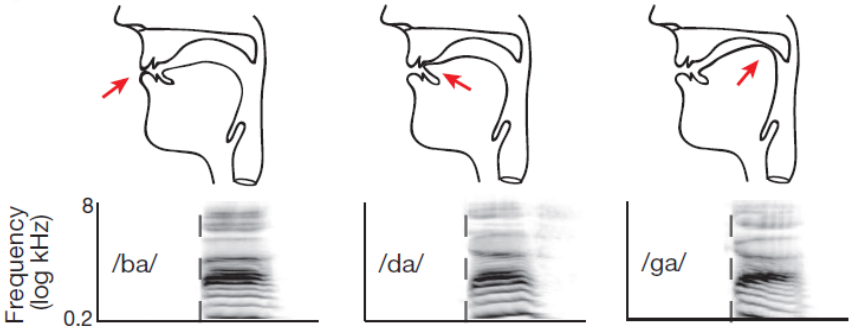


Fig. I.13 Schematic view of the articulation of three consonants (/ba/, /da/, /ga/) with respective spectrogram. Vocal folds are located at the top of the trachea, where the larynx stops in the drawings. Adapted from [Bouchard *et al.*, 2013].

This simplified presentation of the mechanisms of speech generation does not include the many physical elements that add up to produce speech. See for example `http://newt.phys.unsw.edu.au/jw/voice.html` for more details on speech generation, with illustrations.

I.2.3 Speech Processing in the Auditory Pathway

To understand the auditory responses to sounds, researchers tend to use either 'simple' acoustic stimuli, such as tones or clicks, or 'complex' acoustic stimuli such as natural noises or speech-like stimuli. Given the nonlinearity of the auditory system, an understanding of the neural mechanism of tone-processing is indeed bound to meet limitations when it is applied to more complex acoustic stimuli such as speech.

The limited dynamic range of ANFs means their rate can no longer increase after some threshold level is reached (between 40 and 80 dB SPL). At high sound levels, the peaks in the discharge rate profile of auditory nerves tend to disappear due to rate saturation. The instability of rate profiles with stimulus level in the ANF thus led to the idea that the way speech is encoded varies with sound level, from a rate-based representation at low sound levels to a representation based on temporal response patterns at high levels: a temporal code partly replaces the rate code when nerve fibres fire at a discharge rate near their maximal rate [Young and Sachs, 1979]. Modern approaches confirm that using temporal information improves the accuracy at high sound levels in speech recognition tasks, in quiet and even more so in noise [Holmberg *et al.*, 2007], suggesting that temporal coding improves robustness of speech recognition in challenging conditions but plays little role in encoding clear speech at low to medium volume.

Available experimental results at the cellular level of responses to speech are mostly from the CN and the ANF. Single unit recording is classically performed in the CN where both the matter of neural typing and the inputs received (from the nerve fibres) are best understood. Experimental results show that the encoding of characteristics of vowels is already partly

coded into the formant-to-trough rate differences observed in the VCN, where chopper units react more robustly at different sound levels or in noise than primary-like units or even the auditory nerve [May *et al.*, 1998]. Besides neuron typing, tuning curves are normally computed to evaluate how firing rates vary over time in response to different simple sounds such as tones, either reflecting changes in the envelopes or a more complex encoding.

Another way to measure speech responses is to use the electric or magnetic changes recorded at the brainstem or cortex level using external electroencephalogram (EEG) or magnetoencephalogram (MEG) devices to track the auditory response. The brainstem response, in particular, is reliable enough to be used clinically for auditory related disorders. Cortical responses, on the other hand, have higher inter- and intra-subject variability. The other technologies reviewed in [Abrams, 2008] are functional imaging techniques, measuring the changes in blood flows in the brain, which are known to be highly correlated with neural activity [Smith *et al.*, 2002]. Their advantage, relative to evoked potential or evoked fields responses, is their high-spatial resolution. They have, however, a much slower time-scale, averaging activity over the course of seconds, which is extremely slow given that speech tokens are of the order of 30 ms.

Within the peripheral auditory system, the distribution of the relative phases of synchronised activity between cues carries important response features reflecting the stimulus spectral parameters [Shamma, 1985]. This phase-locking property is used in the upstream auditory pathway, as timing features related to modulations are re-encoded into rate activity [Joris *et al.*, 2004]. Indeed, in a rare setting of invasive human recording of cortical activity, it was observed that reconstruction quality was highest for sound features most critical to speech intelligibility, and even allowed decoding of individual spoken words, hinting at neural encoding mechanisms of speech acoustic parameters in higher-order human auditory cortex [Pasley *et al.*, 2012].

All discussions specific to speech encoding are now suspended until the end of chapter VI. We now study the Cochlear Nucleus response to Amplitude-Modulated tones, using many classification algorithms and measures of performances. In order to use such tools, a chapter on Machine Learning is first needed. More precisely, since labelled data is used, only Supervised Learning methods are introduced.

II.

Supervised Learning & Data Mining

This chapter sets the foundations of Machine Learning and Data Mining used later in this thesis. The field of Machine Learning is first introduced in section II.1 and different model validation protocols are presented in section II.2. Classifiers are split into 4 groups: the ones based on Bayesian techniques (II.3), linear classifiers (II.4), nonlinear classifiers (II.5) and ensemble methods aggregating classifiers (II.6). As part of the Data Mining Swiss army knife, a section on dimensionality reduction is incorporated (II.7). Last comes the Weka data mining toolbox (II.8) and the implementation of a selection of its classifiers.

There are many ways to examine the unknown neural code. A way to assess the code is to see how well it supports a task - identifying a sound or discriminating two sounds. A relatively assumption-free approach is to employ a Machine Learning (ML) algorithm to see how well it performs a task given a set of spike trains. Since there are a world of different ML methods, how does one know what is an appropriate method? Here we review the basics of Machine Learning methods relevant to the work that follows in this thesis and give the definitions of classifiers and measures of performance output by the data mining toolbox Weka [Hall *et al.*, 2009].

II.1 Introduction to Machine Learning

Machine Learning, Pattern Recognition, Data Mining (DM) and Artificial Intelligence (AI) are related fields, often mistakenly referred to interchangeably due to their enormous overlap. The grand idea is to use algorithms that are able to learn from data by building statistical models and use these models to infer information from new datasets, hence to use information (raw data) to create knowledge, for example by inferring the neural code of neurons from their spike trains. The following informal definitions are used:

- Machine Learning: Set of algorithms and corresponding theory that aim at learning from data by building models with predictive power;
- Pattern Recognition: Branch of ML that focuses on the recognition of patterns and regularities in data;
- Data Mining: Practical use of ML tools and associated techniques to extract information from data;
- Artificial Intelligence: Branch of Computer Science attempting to create intelligent machines, traditionally split into weak AI and strong AI. The former defines a non-sentient type of intelligence focused on specific tasks, and the later a human-like intelligence with cognitive capability.

Machine Learning algorithms are usually split into two categories:

- Supervised learning algorithms: the algorithm is given the training dataset and the information (labels in the case of classification, numerical values in the case of regression) it is supposed to learn from it;
- Unsupervised learning algorithms: the algorithm has to find features in the data such that it can naturally cluster the data according to those features.

This chapter mostly deals with classification algorithms, or classifiers, to give a flavour of the tools used later on spike train data. Here we give the definitions of a selected subset of the classifiers implemented in Weka, a data mining toolbox used for the classification results in chapter III. A synopsis of all the classifiers used is given in appendix 11.

The classical way to assess a classification work is to split a dataset with its labels into two groups: a training set, used to train an algorithm, and a test set only used to test how accurate the model is. From the labels predicted on instances of the test set, is computed a number - a measure of performance - that sums up an aspect of all classifications. This necessity of testing the model on data that wasn't used during the training phase of the process is due to the classical problem of overfitting, which is equivalent to building a model that performs well on the training set but has very small predictive power. To circumvent this pitfall, there are various model validation methods generally accepted, with different advantages and drawbacks, presented below.

II.2 Model Validation

II.2.1 Training/Testing Paradigm

Resubstitution Method:
The same data is used to train an algorithm and to evaluate it. This unfair method provides optimistic evaluations of a model due to the clear dependance between the dataset used to train a classifier, and the dataset on which it is tested. It is avoided in practice because it leads to overfitting. One can use it during a data exploration phase to have an upper bound on the accuracy attainable later without overfitting.

Holdout Method:
The initial dataset is split into a training set and a test set. The training set is used to train the algorithm, which will be tested on the test set. This method evaluates the generalisation capacity of the model in a fair way, but opens the question of the size of both datasets, which is a non-trivial question. People use their own experience to judge what percentage of the data should go to each set, depending on the amount of data, the number of classes, the speed of training the algorithm, and the importance of having a well-trained classifier.

Leave-One-Out Method:
This strategy alleviates the problem of choosing a training and test set by picking one instance of the data, using all others to train the classifier, and testing it on this instance. This is performed as many times as there are data points, each time choosing a different test instance. This method has the advantage of testing all instances in a fair way and training the algorithm with as much data as is possible, but at a huge computational cost.

Stratified Cross-Validation Method:
Cross-validation is a widely used validation method. It is a fair method with a lower complexity than Leave-One-Out, while still using all points once at test instances. A number of folds is decided, for example 10, and the dataset is cut in 10 folds of approximately the same size. When splitting the data, one can make sure that the number of instances of each class is well-balanced among all folds, in which case, one talks about stratified cross-validation. Each fold will then be used once as test set, while the rest is used as training set. By default, Weka uses 10-fold stratified cross-validation.

II.2.2 Confusion Matrix & Accuracy

After training and testing in a classification task, each tested instance of class i is attributed a label j; for simplicity, let's assume the classes are 1, 2, ... N. This classification is of course correct if and only if $i = j$ and incorrect otherwise, and a very simple measure of the overall performance of the classifier on the test dataset is to count the percentage of correct guesses over the test set. A more informative way to understand what the algorithm did is to look at the confusion matrix, which is an integer-valued square matrix whose size is the number of classes, such that the value at the ith row and jth column counts the number of test instances of class i that were labelled as j. The percentage of correct guesses is then related to the confusion matrix M through its trace (sum of diagonal elements) and L^1 norm (sum of absolute value of all components) by the relationship

$$\%\text{correct} = \text{tr}(M)/||M||_1, \tag{II.1}$$

and is discussed in more detail (among other measures of performance) in chapter IV. This measure is often referred to as accuracy. These two terms are interchangeable when only one label is recognised per instance (a class), but have different meaning when there is an unknown number of labels to recognise per instance (a sequence of classes), as will be the case in the most of part B.

Now that we have defined the model validation methods applicable to all classifiers and the simplest model evaluation procedure, let us introduce the classifiers that are actually used. First we present Bayesian techniques used in Machine Learning, as this permits us to give the definitions of concepts used later while providing a rigorous mathematical framework to work in. We then define common linear and nonlinear classifiers, and lastly the class of ensemble learning algorithms.

II.3 Bayesian Techniques

Real data is noisy, both due to imperfect measurements and the intrinsic statistical variability in the phenomena under study. Imagine for example that everyday you take one of your M (for example, M = 3) different routes to go back home from work; you measure the time it took you (calling this value $x_1(i)$ for day i), and tell it your life partner. In general, it will not be possible to guess correctly each time which path you chose using only this information, because your average speed depends on many external parameters that one can't guess, such as the weather, your hunger or other needs. One would probably need a few more statistics in order to guess the path you took, say, the number of steps you walked this day $x_2(i)$, the mean temperature $x_3(i)$, or a measure of how hard it rained $x_4(i)$ guessed by a look at your clothes, how much you reek of perfume $x_5(i)$.

To rigorously tackle this uncertainty-related problem, scientists tend to use a probabilistic framework to infer the underlying information of interest: Let X_i be the feature vector computed on the ith day, a list of chosen statistics collected each time you walk back home. In the above example, $X_i = (x_1(i), x_2(i), x_3(i), x_4(i), x_5(i))$. The modeller's goal is to use the data you have collected during your first n walks back home X_1, X_2, ..., X_n (each of them containing 5 statistics in the current example), in order to correctly guess the path used on the next $n + 1^{\text{th}}$ walk. Let's call λ_i the path used on day i (path 1, 2 or 3); the goal is then to guess λ_{n+1} (1, 2 or 3), knowing $X_1, \ldots, X_n, X_{n+1}$ and $\lambda_1, \ldots, \lambda_n$. Definitions related to Probability theory are given in appendix 12.1.

Given a probability measure $\mathbb{P}$, the knowledge of a particular event Y that has positive probability modifies the chances of all the other events. This new knowledge transforms the probability $\mathbb{P}$ into a conditional probability $\mathbb{P}(\cdot|Y)$ defined for each event X that has a positive probability by

$$\mathbb{P}(X|Y) = \mathbb{P}(X\&Y)/\mathbb{P}(Y). \tag{II.2}$$

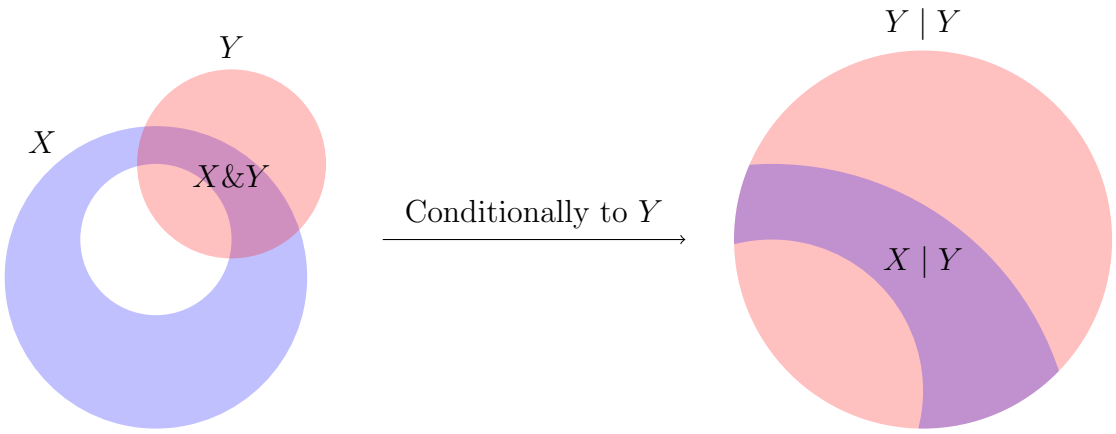


Fig. II.1 Geometrical interpretation of conditional probabilities (equation II.2).

Using geometry to develop an intuition of what this means (figure II.1), this division is a way to renormalise the areas where X also takes place (to sum to 1), only where the event Y also takes place. The separation between what is known and what is to be inferred is at the very heart of Bayes theorem, where they play dual roles:

$$\mathbb{P}(\lambda|X)\,\mathbb{P}(X) = \mathbb{P}(X|\lambda)\,\mathbb{P}(\lambda). \tag{II.3}$$

This formula is used to infer optimal decisions regarding the unknown λ. In this thesis, two natural estimations from Bayesian frameworks are used, respectively called the Maximum Likelihood where the probability measure is supposed to be known, and the Maximum a Posteriori (MAP) that learns from observations.

II.3.1 Maximum Likelihood

The maximum likelihood estimator is the value allowing maximal probability,

$$\hat{\lambda}_{ML} = \text{argmax}_{\lambda \in \text{paths}}\, \mathbb{P}(\lambda). \tag{II.4}$$

The Maximum Likelihood estimation has some very desirable properties under generally valid conditions [Papoulis, 1991], making it a natural statistic to use to infer a real (unknown) sampling parameter, such as being asymptotically unbiased[1], and converging in the mean square[2] sense. Uninformative variables would, ideally, not contribute to changes in the likelihood functions. In practice, however, noise will not be identical for all observations under different λ, thus making the 'optimal' decision (using the available data) not the real optimal (knowing the underlying distributions). Other classifiers may be better suited to deal with noise.

II.3.2 Maximum A Posteriori

After accumulating observations $X = \{X_1, \ldots, X_n\}$ and their associated values $\lambda_1, \ldots, \lambda_n$, Bayes Theorem allows us to define the posterior probability $\mathbb{P}(\lambda|X)$ as

$$\mathbb{P}(\lambda|X) = \frac{\mathbb{P}(X|\lambda)\,\mathbb{P}(\lambda)}{\mathbb{P}(X)}, \tag{II.5}$$

considering that the known paths $\lambda_1, \ldots, \lambda_n$ are implicitly used to define the probability measure, and that $\mathbb{P}(X)$ is not null. The Maximum a Posteriori (MAP) is the value achieving

[1] $\lim_{n\to\infty} \mathbb{E}[\hat{\lambda}_{ML}] = \lambda_0$, where λ_0 is the real value we try to assess.
[2] $\lim_{n\to\infty} \mathbb{E}[||\hat{\lambda}_{ML} - \lambda_0||^2] = 0$

maximal posterior probability[3], which means taking as a guess for the unknown value λ_0

$$\hat{\lambda}_{MAP} = \text{argmax}_{\lambda \in \text{paths}} \mathbb{P}(\lambda | X_1, \ldots, X_n, X_{n+1}, \lambda_1, \ldots, \lambda_n). \tag{II.6}$$

The probability function $\mathbb{P}$ has not been defined here; this choice is problem-dependent and different modellers ought to use different modelling approaches. Gaussian Processes are a common modelling tool thanks to their flexibility and their strong theoretical grounding. They are also used in Machine Learning and make an implicit use of MAP.

II.3.3 Gaussian Processes

Gaussian Processes are a family of stochastic processes[4] commonly used nowadays in Machine Learning [Rasmussen, 2006], thanks to their well-established theory and learning capability. The flexible probabilistic framework this family provides makes it a good candidate to build models from. Compared to Artificial Neural Networks where adaptive hidden units can learn important representations or features from the data, a central difficulty in using Gaussian Processes for Machine Learning is the choice or optimisation of the covariance function $k(s,t)$ that gives the correlation between the random variables X_s and X_t[5]. The other main drawback they share with many technologies is scaling-up, having by default an n^2 requirement in space, and n^3 in time, for a data sample of size n. Thus, developing scalable algorithms or approximations is an important aspect of Gaussian Processes when dealing with large datasets.

II.4 Linear Classifiers

This section focuses on the design of linear classifiers. This approach leads to fairly simple models to train and to interpret, and also to computationally cheap algorithms. It is first assumed that all feature vectors from the available classes are linearly separable.

II.4.1 Linear Separability

Linear separability in a 2-class dataset occurs when thresholding the output of a linear form is enough to correctly classify all data points. In geometrical terms, this property is equivalent to the existence of a hyperplane able to split the two classes, or to requiring that the respective convex hulls of the two sets are disjoints. Such a dataset is presented in figure II.2, where

[3] In other words, the MAP is the maximal likelihood on the updated probability.

[4] The exact definition of a stochastic process is not required in this work; a simple description is that of a group of Random Variables indexed by continuous or discrete labels (real- or integer-valued).

[5] Meaning that $\mathbb{E}[X_s X_t] = k(s,t)$, often assumed equal to $k(|s-t|)$ to simplify the choices. The next step is to parameterise this function to further reduce the choices. However, since this becomes an expert choice, some modelling difficulties remain.

$w \in \mathbb{R}^2$ and $b \in \mathbb{R}$ could be any vector and scalar such that $\{x \in \mathbb{R}^2 \,|\, w \cdot x = b\}$ represents the lines H_2 or H_3, for example. This way, the linear form $x \mapsto w \cdot x$ provides a simple way to split the dataset: all black points verify $w \cdot x > b$, and all white points $w \cdot x < b$ (or the opposite, depending on the chosen w).

II.4.2 Support Vector Machine

Support Vector Machines (SVMs) find an optimal hyperplane by solving a constrained optimisation problem: Let $(x_i)_{i=1,\dots,n}$ be the dataset, each point $x_i \in \mathbb{R}^p$ being associated a label $y_i \in \{1, -1\}$ (represented in black and white in figure II.2 with $p = 2$), and assume linear separability. An SVM will look for the optimal vector w and scalar b to define two hyperplanes, thus splitting the space in 3 domains. For all $x \in \mathbb{R}^p$,

- $(w \cdot x - b) \leq -1$ where there is no point labeled 1,
- $(w \cdot x - b) \in]-1, 1[$ where there is no point labeled -1 or 1,
- $(w \cdot x - b) \geq 1$ where there is no point labeled -1.

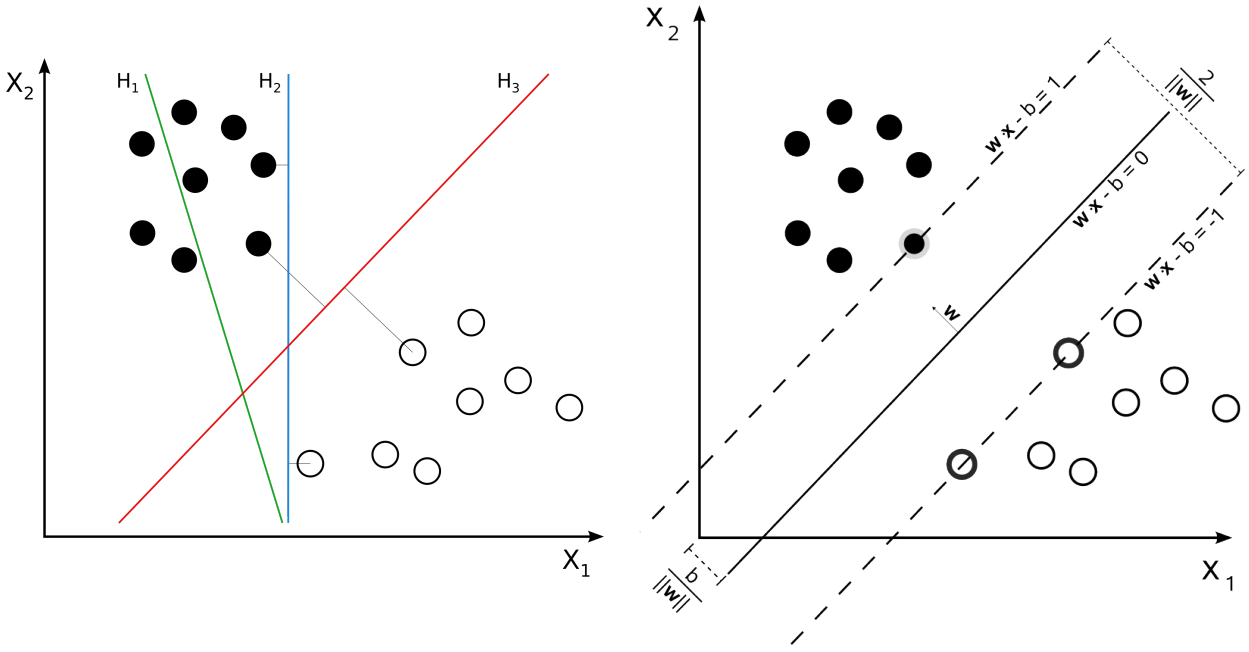


Fig. II.2 *(Left)* Plot showing how a support vector machine would choose a separating hyperplane for two classes of points in 2D. H_1 does not separate the classes. H_2 does, but only with a small margin. H_3 separates them with the maximum margin. *(Right)* Geometrical representation of the solution of an SVM applied to a linearly separable 2D dataset. A separating hyperplane is found, and the margin separating both classes $2/||w||$ is maximal. Samples on the margins are called support vectors. Images are within the Public domain, via Wikimedia Commons.

The distance between the hyperplanes[6] $(w \cdot x - b) = -1$ and $(w \cdot x - b) = 1$ is $2/||w||$. Trying to make this margin as wide as possible is equivalent to minimising $||w||$. The optimal w and b are thus chosen by solving this constrained optimisation problem:

$$\begin{aligned} (w, b) &= \text{argmin}_{w \in \mathbb{R}^p, b \in \mathbb{R}} ||w||_2, \\ \text{subject to} \qquad & \forall i, \quad y_i(w \cdot x_i - b) \geq 1. \end{aligned} \tag{II.7}$$

as is suggested by the right plot of figure II.2.

II.4.3 Single-Layer Perceptron Algorithm

The perceptron algorithm, also termed single-layer perceptron, is the simplest feedforward neural network. It is a 2-class linear classifier that learns from the data by updating weights until all data points are correctly split. Consequently, it requires linear separability. If this assumption is verified, the algorithm is guaranteed to converge, and there are upper bounds on the number of iterations necessary to find a separating hyperplane. If the data is not linearly separable, the algorithm doesn't terminate.

Formally, a Heaviside function is applied to the output of a linear form: $f(x) = 1$ if $w \cdot x + b > 0$ and 0 otherwise. After choosing a learning rate $0 < \alpha \leq 1$ and an initial weight vector $w = w(0)$ and scalar $b = b(0)$ (that could alternatively be included in the vector w), the learning rule is given by repeating the following steps, noting that each index i corresponds to both the i-th point x_i and its scalar weight w_i that will change whenever x_i is tested and incorrectly classified:

Algorithm 1 Perceptron Algorithm

procedure Perceptron Learning Rule
initialRandomWeights
    $w \leftarrow \text{rand}(n, 1)$
    $b \leftarrow \text{rand}(1, 1)$
loop:
    $i \leftarrow \text{rand } \{1, \ldots, n\}$
    $\hat{y}_i \leftarrow f(w \cdot x_i + b)$
    $w_i \leftarrow w_i + \alpha(y_i - \hat{y}_i) \cdot x_i$
    $b \leftarrow b + \alpha(y_i - \hat{y}_i)$
    if some training instances are incorrectly classified **then**
        goto *loop*
        fi

[6] A point x such that $(w \cdot x - b) = -1$ projects onto the second hyperplane to a point y with $(w \cdot y - b) = 1$, by a translation along the vector w, orthogonal to those planes, so there exists $\mu \in \mathbb{R}$ such that $y = x + \mu w$. This leads to $\mu = 2/||w||^2$ and thus to the distance between x and y being $||x - y|| = ||\mu w|| = 2/||w||$.

II.5 Nonlinear Classifiers

In the previous section were presented linear classifiers. However, most methods are non-linear. We now present a selection of them that will be encountered later on in this thesis.

II.5.1 k-Nearest Neighbours

The k-Nearest Neighbours (kNN) is a type of instance-based learning, or lazy learning. Given a fixed integer k, a class is chosen for a test sample by a majority vote on the classes of the k nearest neighbours in the training set. In the simplest case $k = 1$, the class attributed to a sample is the same class as the training point that is nearest under the distance metric being used. This algorithm is simple, commonly used and has interesting theoretical bounds on its error probability [Devroye *et al.*, 1997], but requires a brute-force search over the training set and does not provide much insight, as opposed to constructing generative models where one tries to summarise the important bits of information within a model.

II.5.2 Kernel Methods

When each dataset's sample is too big for a computer (large time series) or complex (graphs, texts, spike trains), one approach in Machine Learning involves extracting meaningful features from each sample and giving them to the classifier. This approach was the one used in the route-guessing example. Kernel methods, on the other hand, only require a similarity function over pairs of data points, applicable on their original representation. In a Machine Learning context, a kernel is the similarity function provided by a domain expert.

A common technique to reduce the computational cost that those approaches often involve is the kernel trick: instead of explicitly mapping the raw data into a high-dimensional representation to use a similarity function, one may manually construct a feature map such that the similarity function can be rewritten as the inner product in a convenient space - a real or complex inner product space that is also a complete metric space with respect to the distance function induced by the inner product: a Hilbert space. In this space, the inner product can often be computed efficiently, thanks to the inner product's linearity. For clarity, let's give an example on spike trains.

From two spike trains $T = \{t_1, \dots, t_n\}$ and $\tilde{T} = \{\tilde{t}_1, \dots, \tilde{t}_m\}$ that are represented as distributions $D(t) = \sum_i \delta(t - t_i)$ and $\tilde{D}(t) = \sum_j \delta(t - \tilde{t}_j)$ and convolved with a Gaussian function $g : x \mapsto \exp(-x^2/\sigma^2)$, one obtains two functions $f, \tilde{f} \in L^2(\mathbb{R})$, given by

$$f(t) = \sum_i \mathrm{e}^{-\frac{(t-t_i)^2}{\sigma^2}} \qquad \text{and} \qquad \tilde{f}(t) = \sum_j \mathrm{e}^{-\frac{(t-\tilde{t}_j)^2}{\sigma^2}},$$

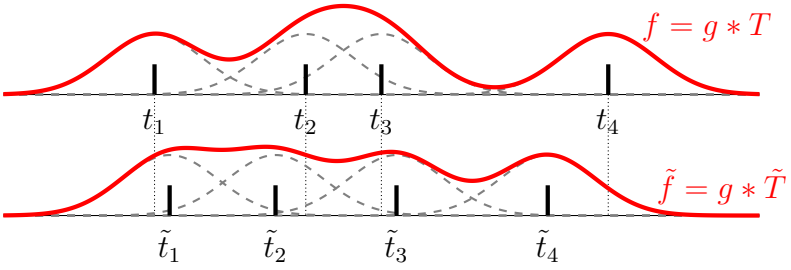


Fig. II.3 Convolution of two spike trains $T = \{t_1, \dots, t_4\}$ and $\tilde{T} = \{\tilde{t}_1, \dots, \tilde{t}_4\}$ with a Gaussian function g, to obtain two functions $f, \tilde{f} \in L^2(\mathbb{R})$, implicitly used for the Kernel Trick. For simplicity, spike trains are assimilated to their corresponding Dirac combs.

as exemplified in figure II.3, leading to the L^2-distance between f and $\tilde{f}$

$$d(f, \tilde{f}) = \sqrt{\int_{t \in \mathbb{R}} \left(\sum_i \mathrm{e}^{-\frac{(t-t_i)^2}{\sigma^2}} - \sum_j \mathrm{e}^{-\frac{(t-\tilde{t}_j)^2}{\sigma^2}} \right)^2 \mathrm{d}t}.$$

However, this distance being induced by the L^2-inner product can be rewritten using the L^2 scalar product $\langle f_1, f_2 \rangle = \int_{x \in \mathbb{R}} f_1(x) f_2(x) \, \mathrm{d}x$ as

$$d(f, \tilde{f})^2 = ||f - \tilde{f}||^2_{L^2} = \langle f - \tilde{f}, f - \tilde{f} \rangle = \langle f, f \rangle - 2\langle f, \tilde{f} \rangle + \langle \tilde{f}, \tilde{f} \rangle$$

which can then be very efficiently computed using the known value of the integral of Gaussian functions. This approach alleviates the storage of f and $\tilde{f}$ into memory and the numerical evaluation of integrals.

II.5.3 Artificial Neural Networks

Artificial Neural Networks (ANNs), or neural nets for brevity, are a family of statistical learning models inspired by biological neural networks. Any neural net having at least two hidden layers is called 'deep'; this adjective informally stresses that the network is tough to handle. Each nonlinear function is called a neuron or unit, and will perform computations from the linear combination (numeric weights) of the input it receives from a set of other neurons, and then applying a nonlinearity - a function called the activation function. Using enough neurons and layers, very complex functions can be learned by the network; theoretically, the Universal Approximation Theorem states that any bounded continuous function in $\mathbb{R}^n$ can be approximated [Cybenko, 1989].

One usually splits the neurons into 3 categories of layers, as shown in figure II.4: one input layer, an arbitrary number of hidden layers, and one output layer. The raw data is sent to the input layer, processed in the hidden layers, and the result is read out in the output layer. ANNs are usually separated into feedforward and recurrent neural nets: feedforward

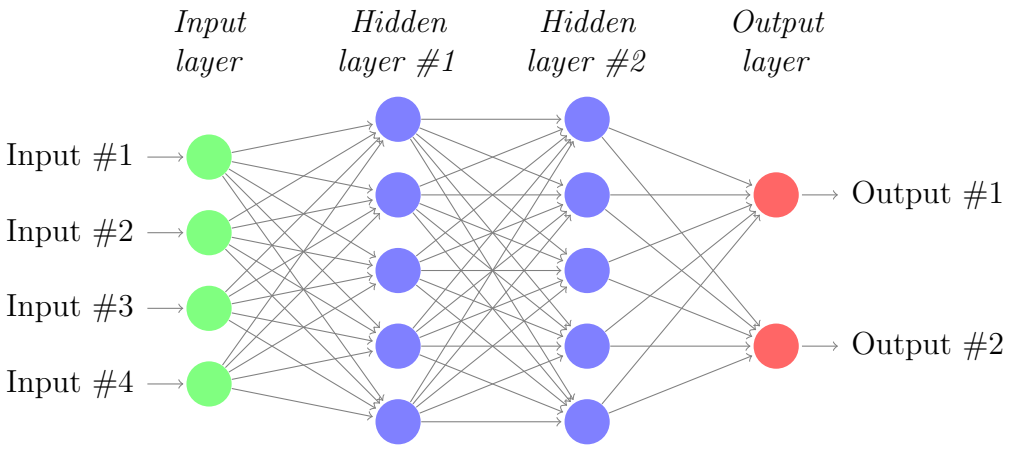


Fig. II.4 Fully connected feedforward ANN with two hidden layers.

nets' connections only go forward, from one layer to the next, whereas recurrent nets can have connections forming a directed cycle, or loop.

Recurrent neural networks have a particularly high potential for tasks using sequential data since their state is a function of all previous states. Their rich dynamics, when used with appropriate neural net architectures, led to great improvements in speech recognition tasks [Fernandez *et al.*, 2007; Graves *et al.*, 2013; Sak *et al.*, 2015]. A description of the gradient based method allowing these improvements, the long short-term memory (LSTM), is beyond the scope of this thesis [Hochreiter and Schmidhuber, 1997].

Neural networks have a long history [Bishop, 1995]. Their popularity decreased for a few decades, starting in the 70s, and support vector machines overtook neural nets in the machine learning community for some time, due to their simplicity. However, they received a renewed interest around 2012 [Le *et al.*, 2011] with Deep Learning: hardware technologies had evolved in the meantime, enough to handle large amounts of data and process it in a reasonable amount of time. Using big clusters of computers and GPU computing, neural nets came back and have since proven themselves in all fields of Machine Learning. New hardware continues to be developed to support the development and applications of Deep Learning, the latest being the Tensor Processing Unit (TPU) [Jouppi *et al.*, 2017] that was used to beat the world's Go champion in May 2017 [Silver *et al.*, 2016]. This victory from the algorithm, AlphaGo, had a big societal impact as it showed that progress in AI research was moving fast[7]. This effort continues, and a stronger algorithm was trained, this time without prior human knowledge, and defeated AlphaGo in October 2017 [Silver *et al.*, 2017].

[7]The impact in Europe and the United-States was clear nytimes.com/2017/05/23/business/google-deepmind-alphago-go-champion-defeat, but probably still small compared to that of South Korea that first feared AI newscientist.com/how-victory-for-googles-go-ai-is-stoking-fear-in-south-korea and as a result decided to invest in it much more nature.com/south-korea-trumpets-860-million-ai-fund-after-alphago-shock.

The most common algorithm used to train a feedforward neural net in a supervised framework is the backpropagation algorithm, presented now, along with two types of deep networks: The Multilayer Perceptron and the Radial Basis Function network.

II.5.4 Backpropagation Algorithm

ANNs have to learn their parameters from the training data. Typically there are many parameters. As often in Machine Learning, it is formalised as an optimisation problem: one must compute the weights minimising a chosen cost function that measures the error made using the current parameters, the most common being the quadratic error:

$$E = \frac{1}{2} \sum_j (t_j - y_j)^2 , \tag{II.8}$$

where t_j are the output targets, and y_j the ANN outputs for a given set of parameters w_{ij} and list of inputs x_j, for i crossing all layers and j the nodes within each layer.

The gradient descent method called backpropagation is the most popular one for this optimisation problem [Theodoridis and Koutroumbas, 2006]. Backpropagation updates each neuron's weights by propagating the gradient of a cost function, from the output layer back to the input layer, using the chain rule to derive the changes at every node. The gradient computed for each neuron's weight is multiplied by a learning constant and subtracted from the current value, as in all gradient descent methods. Calling $g(x)$ the activation function of a node, α the learning rate and h_j the output of this jth neuron, the target change to apply to the weight $w_{i,j}$ is

$$\Delta w_{ji} = \alpha(t_j - y_j)g'(h_j)x_i \tag{II.9}$$

where x_i is the ith input. This step is iteratively repeated after recomputing the neuron's output on the training set and the cost, until the cost function or its gradient are smaller than a chosen threshold.

This scheme has many variants, including adapting the learning constant, using noise to avoid convergence towards local minima, or updating weights using only batches of training data to speed-up the convergence or avoid overfitting.

II.5.5 Multilayer Perceptron

A multilayer perceptron is a mutliclass classifier based on a feedforward ANN such that each layer is fully connected to the next one. Except for the initial layer (input), each node is a neuron with a nonlinear activation function, using the backpropagation technique to train the network.

As in the single-layer case, there are theoretical guarantees of convergence if the training data is linearly separable. A 3-layer perceptron is also able to separate any union of polyhe-

dral regions in space, using enough neurons: Neurons in the first layer form the hyperplanes, those in the second layer form the regions, and neurons from the output layer form the classes [Theodoridis and Koutroumbas, 2006].

II.5.6 Radial Basis Function Network

A Radial Basis Function (RBF) is a real-valued function whose value depends only on the distance from the origin [Theodoridis and Koutroumbas, 2006]. Consequently, an RBF network is an Artificial Neural Network that uses RBFs as activation functions, typically with three layers. The output of a neuron from such a network, using the widely used Gaussian forms on the input vector x, is written as

$$g(x) = w_0 + \sum_{i=1}^{k} w_i \exp\left(-\frac{||x - c_i||^2}{2\sigma_i^2}\right),$$

where the parameters w_i, c_i, σ_i are learnt by training. The network training is usually split into two steps: choosing the c_i's and σ_i's using random sampling or (unsupervised) k-means clustering[8], and fit the w_i's using an objective function (supervised). A backpropagation step can be used as a third step to fine-tune all parameters.

Compared to multilayer perceptrons, RBF networks are of local nature, centred around a set of points. This difference has repercussions on the convergence speed and the generalisation performance: Multilayer perceptrons learn slower than their RBF counterpart but have improved generalisation properties.

II.6 Ensemble Learning

Ensemble methods are a class of multiple learning algorithms to improve the predictive performance of any single classifier it uses. Similarly, some classifiers are naturally defined for 2-class problems or designed for simple computations; these limitations can be bypassed by making different algorithms work together. In this section we describe some selected classical ensemble methods.

II.6.1 END

Weka's END classifier handles multi-class datasets with 2-class classifiers by building an Ensemble of Nested Dichotomies. Its default base classifier is the decision-tree generating algorithm J48.

[8] k-means clustering aims to partition n observations into k clusters in which each observation belongs to the cluster with the nearest mean, serving as a prototype of the cluster.

II.6.2 Bagging

Bootstrap aggregating - shortened into bagging - algorithms create an ensemble of subsets from the training data. Each subset consists of a chosen number of training points from the original dataset, sampled uniformly and with replacement. By sampling with replacement, an observation may be repeated in each subset. For classification, a model is trained on each subset, and a vote between their outputs gives the class. According to the original article [Breiman, 1996], the important element to allow bagging to outperform the base classifier is the instability of the prediction method. If perturbing the learning set can cause significant changes in the predictor constructed, then bagging can improve accuracy.

II.6.3 Boosting

Boosting algorithms improve a weak learner by adapting the weights attributed to the training data. Examples that are misclassified gain weight. Thus, the algorithm will focus on the examples previously misclassified. It was shown [Long and Servedio, 2010] that this method was not noise-tolerant, leading to a poor performance on real (noisy) data even for AdaBoost and LogitBoost, two common algorithms from this family.

II.6.4 Random Forest

The Random Forest algorithm combines the bagging idea presented above, and a random subset of the features (random subspace method, also referred to as attribute bagging), in order to construct a multitude of decision trees: each of these decision tree uses a specific (random) set of attributes, and the final decision is made by majority vote. This randomisation in data space and feature space is useful to correct for the decision trees' habit of overfitting, while providing a feature selection framework.

II.7 Dimensionality Reduction

Now that a representative amount of classifiers has been presented, this section covers the dimensionality reduction methods encountered in this thesis.

II.7.1 Principal Component Analysis

Principal Component Analysis (PCA), is a statistical procedure able to transform correlated observables into a set of uncorrelated variables called principal components. In practice, it is often used as a dimensionality reduction method, by projecting the observations onto a set of orthogonal vectors and only keeping the principal components associated with a high variance.

Let $X \in \mathcal{M}_{n,m}(\mathbb{R})$ be a matrix made of n observations of m different variables, such that for each variable, the mean observation is zero (column-wise zero empirical mean). The linear principal components of a PCA are obtained from the eigenvectors of the covariance matrix $\text{cov}(X)$, and give directions in which the data have maximal variance. A Matlab code to obtain the first k principal components of X is `[V, D] = eigs(cov(X), k);` and a new set of observations with 0-mean $Y \in \mathcal{M}_{\tilde{n},m}(\mathbb{R})$ can then be right-multiplied by $V \in \mathcal{M}_{m,k}$, which performs the projection onto the principal components.

The habit of keeping the components with large variance is rather old, as it was already criticised in the 80s, for example in [Jolliffe, 1982]:

> The original idea was to treat the principal components in the same way as ordinary regressor variables, and assess whether they should be included by computing their association with the dependent variable. However, in several recent publications the suggested rule for inclusion is simply based on the variance of the component, i.e. retain those components with large variances and reject those with small variances.

It is however fairly simple to construct an example where the information of interest is entirely contained in the component of smaller variance[9], and some real-world datasets present the same behaviour. Thus, selecting the components solely according to the size of their variance is known to be a flawed approach but is, in practice, still used by default. Alternative processing choices exist, such as renormalising the variables before running the PCA - which is equivalent to replacing the covariance function by a correlation function.

In a Machine Learning setting where applying a PCA is part of model-building, it is required to train the principal components on the training set only to avoid overfitting, and keep in mind that PCA maintains what is common in the data, not what differentiates them. There is no guarantee that the principal components are consistent with the classes in a classification problem. It is accepted that PCA is most useful when the variables behave like Gaussians [Theodoridis and Koutroumbas, 2006], since in this case, obtaining uncorrelated variables is equivalent to obtaining independent variables, which is generally not the case. Beside this separation between uncorrelated and independent, another problem arises regarding the scaling of the data, because PCA is sensitive to scaling. Yet another pitfall is linearity: if the relationship between the observed variables is nonlinear, a nonlinear dimensionality reduction method might be better suited.

To summarise, one needs to try this method to find out how useful it is. As such, PCA can be considered a data exploration tool rather than a reliable perfect features extraction method for classification[10].

[9] stats.stackexchange.com/what-can-cause-pca-to-worsen-results-of-a-classifier

[10] David MacKay said as a response to a commentary on the absence of PCA in his book [MacKay, 2003]: "Principal Component Analysis" is a dimensionally invalid method that gives people a delusion that they are doing something useful with their data.

II.8 The Weka Data Mining Toolbox

II.8.1 Weka

Weka [Hall *et al.*, 2009] is a collection of Machine Learning algorithms for Data Mining tasks. It contains tools for data pre-processing, classification, regression, clustering, association rules, and visualisation. As a Java open-source package, its methods can be called directly from Matlab. It contains an impressive list of tools, has a very active mailing-list, and tutorial videos freely available online. It is used extensively in chapter III. This section provides precise information regarding a few classifiers, selected for being widely used (k-nearest neighbours, logistic, SMO) or because a specific behaviour was experimentally observed (MultiClassClassifier, discussed in chapter IV). See appendix 11 for a synopsis of all algorithms.

II.8.2 Implementation of Weka's Selected Classifiers

Here we define a few selected classifiers from the Weka data mining toolbox: Nearest Neighbour and Logistic (because they are simple and have been studied extensively), SMO (which is an SVM) and MultiClassClassifier (because of the behaviour of a specific measure of performance, as presented in detail in chapter IV). The references for each algorithms' implementation are on the Weka website. However, the software documentation being difficult at times, users may need to dig in the code.

IBk: This class implements the k-Nearest Neighbours algorithm (kNN), and was used with the default option of $k = 1$, reducing the classifier to a nearest neighbour algorithm: Given n training instances, a test instance will be classified as belonging to the same class as the closest training point, like a vote among nearest neighbours from each class. To output class membership probabilities, a function of the minimal distance between a test point and every training instance of a class is computed: any real decreasing function would do; by default Weka compares instances using an Euclidian norm and computes class membership probabilities using an inverse function $x \mapsto (x + 0.001)^{-1}$ (the small value is added to avoid division by zero)[11].

Logistic: This classifier builds a multinomial logistic regression model using k classes for n instances with m attributes. For a given parameter matrix $B \in \mathcal{M}_{m,k-1}(\mathbb{R})$ whose columns are written as B_j, the probability for class $j \in \{1, \ldots, k-1\}$ with the exception of the last

[11] However, a simple test left some doubts about this: as a training set, the values 0 and 1 are used, belonging to class 0 and 1 respectively, and the value 0.1 is tested. Using $f = @(x)(1/(x+0.001))$, we should obtain that this point belongs to class 0 with probability $f(a)/(f(a) + f(b)) = 0.8992$ but Weka predicts 0.954. This implies the implementation in Weka is more complex than documented.

class is

$$\mathbb{P}(X_i \text{ belongs to class } j) = \mathbb{P}_j(X_i) = \frac{\exp(X_i B_j)}{1 + \sum_{j=1}^{k-1} \exp(X_i B_j)},$$

and the last class has probability

$$\mathbb{P}_k(X_i) = 1 - \sum_{j=1}^{k-1} \mathbb{P}_j(X_i).$$

The (negative) multinomial log-likelihood is:

$$L(B) = -\sum_{i=1}^{n} \sum_{j=1}^{k} \ln(\mathbb{P}_j(X_i)) + \lambda_r ||B||_2^2,$$

where λ_r is called the ridge parameter, used to increase the computational stability of the scheme and address the problem of multicollinearity (when predictor variables are highly correlated) [Dorugade, 2014]. In order to find the matrix B for which L is minimised (i.e. the likeliest parameters given the training instances), a Quasi-Newton Method is used to search for the optimised values of the $m * (k-1)$ variables. A test instance will then be classified according to the highest probability it belongs to a class.

MultiClassClassifier: This metaclassifier handles multi-class datasets with 2-class classifiers. By default - the options used here - the Logistic classifier is used for each binary classification and the evaluation is one-against-all, meaning that each class will be evaluated against all others together. We note that this algorithm is able to apply error-correcting output code (ECOC) [Dietterich and Bakiri, 1995] to increase the accuracy. Weka implements a randomised coding matrix, as suggested by [James and Hastie, 1997] where the ECOC approach is used.

Each base classifier will output a probability that a test instance belongs to its positive class (the negative class being all training instance of the other labels, in the multi-class case). The likeliest label is then chosen as the predicted class.

SMO: (Sequential Minimal Optimization) This metaclassifier solves multiclass problems using Hastie and Tibshirani's pairwise coupling method [Hastie and Tibshirani, 1998], where each base classifier is a support vector classifier. The default option, used in chapter IV, provides probabilities of 1 for guessed classes, and 0 for all others. Another option uses logistic regression fitting to output probabilities between 0 and 1, thus outputting a different K&B information score even though the confusion matrix would be the same.

Now that all the Machine Learning and Data Mining tools about to be used have been defined, it is time to apply them to a first task: decoding the modulation frequency in the neural response of an Amplitude-Modulated tone from Cochlear Nucleus data.

III.

Demodulation in the Cochlear Nucleus

Armed with a collection of classifiers and knowledge about the auditory system, it is now possible to investigate the encoding of Amplitude-Modulated tones by neurons in the Cochlear Nucleus. A general introduction to this problem is given in section III.1. The methodology is then provided in section III.2 followed by the results in (III.3). The chapter ends with a discussion on these results (III.4). The main contribution in this chapter is twofold: revisiting a classical Cochlear Nucleus dataset with a more modern Data Mining approach, and performing a massive comparison of Machine Learning classifiers on this dataset.

III.1 Introduction

Right after the Auditory Nerve, the Cochlear Nucleus (CN) processes the auditory information received from the inner hair cells in the cochlea, as described in chapter I. While this nucleus has been studied for decades in various animals, the functional roles of the units within it and their codes — the way they process information and the functional reasons behind this — is still subject to intensive research. Some neurons actually do not seem to transmit information through the average number of spikes they emit, but rather through the patterns of their spikes [Butts *et al.*, 2007]. The strategies of communication between neurons are the different neural codes they use [Quiroga and Panzeri, 2013]. In this chapter, our interest is focused on the neural code of CN neurons.

Given the exquisite temporal precision of spikes, a classical measure - namely the Vector Strength - has historically been used to evaluate the information contained in spike trains. Recent findings [Scholes *et al.*, 2015] suggest that while for some unit types this measure is appropriate, it is not universally good even within the Cochlear Nucleus and does not reflect the information contained in spike trains [Laudanski *et al.*, 2010]. It was shown in [Scholes *et al.*, 2015] that some complex spike trains are not suitably described by Vector Strength, since some neurons statistically behave like Poisson processes (formally defined in appendix 12) while others are more regular.

For auditory information, evidence suggests that the code changes from temporal coding to rate coding when the information is transmitted up to the auditory cortex, at least for amplitude-modulation encoding [Joris *et al.*, 2004]. Another type of analysis thus seems necessary to extract as much information as possible from spike trains and with less assumptions than those implicitly made when using Vector Strength, which is the motivation for the work we present. A computational architecture was developed to use modern tools of Data Mining, presented in chapter II, as a means to extract the modulation frequency of an Amplitude-Modulated tone from the patterns of spikes produced by cat CN neurons. We will use Blackburn and Sachs' [Blackburn and Sachs, 1989] classification of CN neurons into 13 different types of units, according to their physiology; see subsection I.1.2 for a presentation of the different types of cells.

Following the work of [Wohlgemuth and Ronacher, 2007; Scholes *et al.*, 2015; Geisler *et al.*, 1991] and having access to CN data [Rhode and Greenberg, 1994] described in III.2.1 from anaesthetised cats subject to amplitude-modulated (AM) tones, we will study the neural code of CN neurons under a single task. This task is to recover a specific feature of the sound — the modulation frequency of an AM tone — from neuronal responses, in a supervised learning framework. Even though this parameter is numerical in nature, the limitation of having a predefined discrete set of values this parameter can take motivated us to turn to a classification framework rather than a regression framework: the numerical aspect of the modulation frequencies to be inferred from the spike trains is thus lost because

those values are considered as labels. This should prove beneficial if the encoding of the modulation were to change a lot between the lower and the higher modulation frequencies. This decoding task provides neural coding insight in the following sense: the one able to perform the classification has decoded part of the message contained in a neuron's spike train. As such, being able to recover the label hidden behind a spike train is a generalisation of finding a tuning curve for primary neurons.

Artificial Intelligence aims at automatically extracting and processing meaningful information in a sentient way using some specific algorithms to learn from the data (Machine Learning), but this objective is not reached yet and advances are still made on a case by case basis [Bowling *et al.*, 2015]. Thus it makes sense to compare different Machine Learning techniques to understand which one is able to appropriately classify neuronal spike trains. This problem of classification algorithm selection has been computationally investigated on various datasets [Demar, 2006; Ali and Smith, 2006; Ng and Jordan, 2002; Boulesteix *et al.*, 2008; Sampson, 2012; Lehmann *et al.*, 2007]. In this classification framework, we will test an extensive list of classifiers from the Weka toolbox [Hall *et al.*, 2009], data preprocessing methods (representation of the spike trains), using the accuracy (II.2.2) as measure of performance.

The contribution of this chapter is two-fold: on one hand, it revisits a known dataset using more modern data mining tools to test if previous conclusions about encoding in the Cochlear Nucleus still stand. On the other hand, the results presented in this chapter are a practical contribution to the problem of classification algorithm selection. The methodology is now described in section III.2 followed by the result section III.3. This chapter ends with a discussion III.4.

III.2 Methodology

III.2.1 Dataset

The full dataset is a list of 1,534,375 spike trains, recorded by Rhode & Greenberg [Rhode and Greenberg, 1994] from 688 CN neurons in cats. The authors described their experiment this way:

> We investigated the neural temporal mechanisms in [...] the cochlear nuclei of the pentobarbital sodium-anesthesized cat associated with the neural coding of 100% amplitude modulated (AM) tones, both in quiet and in the presence of wideband, quasi-flat-spectrum noise. The AM carrier frequency was set to the neuron's characteristic frequency (CF) and the sound pressure level (SPL) of acoustic stimuli was varied over a wide dynamic range of intensities ($\leq$ 40 dB).

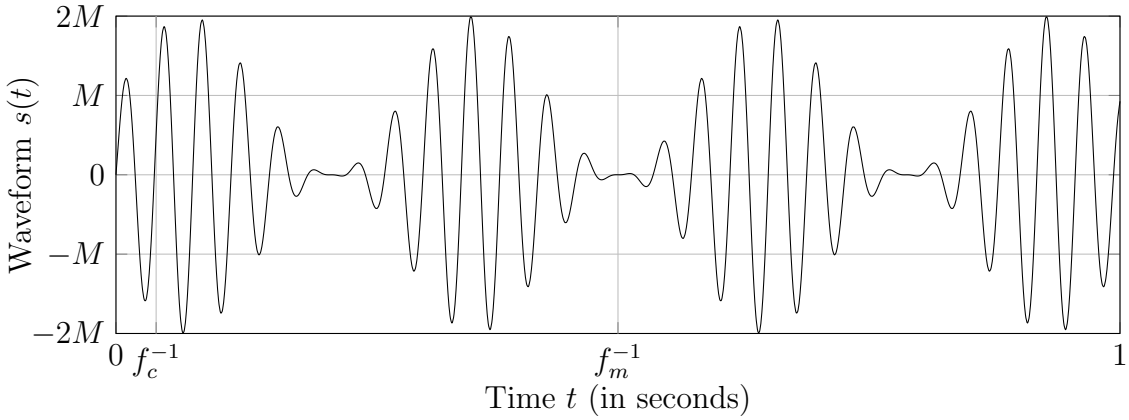


Fig. III.1 Plot of the 100% AM tone $s(t)$ defined by formula III.1, with $f_c \gg f_m$.

In this chapter, the following notations are used: f_m denotes the modulation frequency (50-2550 Hz) of a 100% AM tone, f_c its carrier frequency, and M sets the sound level to various sound levels, ranging from 10 to 110 dB SPL and mostly at 30, 50 or 70 dB SPL. This means that the recorded neural data are the time responses to waveforms of the form

$$s(t) = M\,(1 + \sin(2\pi f_m t))\sin(2\pi f_c t)), \tag{III.1}$$

assuming a null initial phase. See figure III.1 for a plot of such a function. From these responses, the authors computed the Vector Strength (defined in section I.2.1.2) of the neural responses to the signal's modulation as a measure of the AM-encoding capability of the neurons, and from then computed the temporal Modulation Transfer Function (tMTF) of each unit, which reads, in essence and as seen in section I.2.1.3, the average Vector Strength per modulation frequency.

See figure III.2 for a visual presentation of the discharge patterns from two different neurons to the same stimuli (25 presentations for each modulation frequency in {50, 150, ..., 1150}), and figure I.10 for a representation of the Vector Strength computation on three short spike trains recorded from the same CN neuron under three different AM tones (with a modulation frequency of 150, 350 and 550 Hz, respectively).

In this work, spike trains will be processed in different ways and results obtained with different classifiers will be compared. The different processing stages are summarised in figure III.3; the way the 463 small datasets were designed requires some explanation.

III.2.2 Data Selection

We first group spike trains corresponding to a single animal, neuron, modulation level (M) and carrier frequency (f_c), which allows us to split the spike trains dataset into 2118 smaller datasets, each containing responses from AM tones with various modulation frequencies

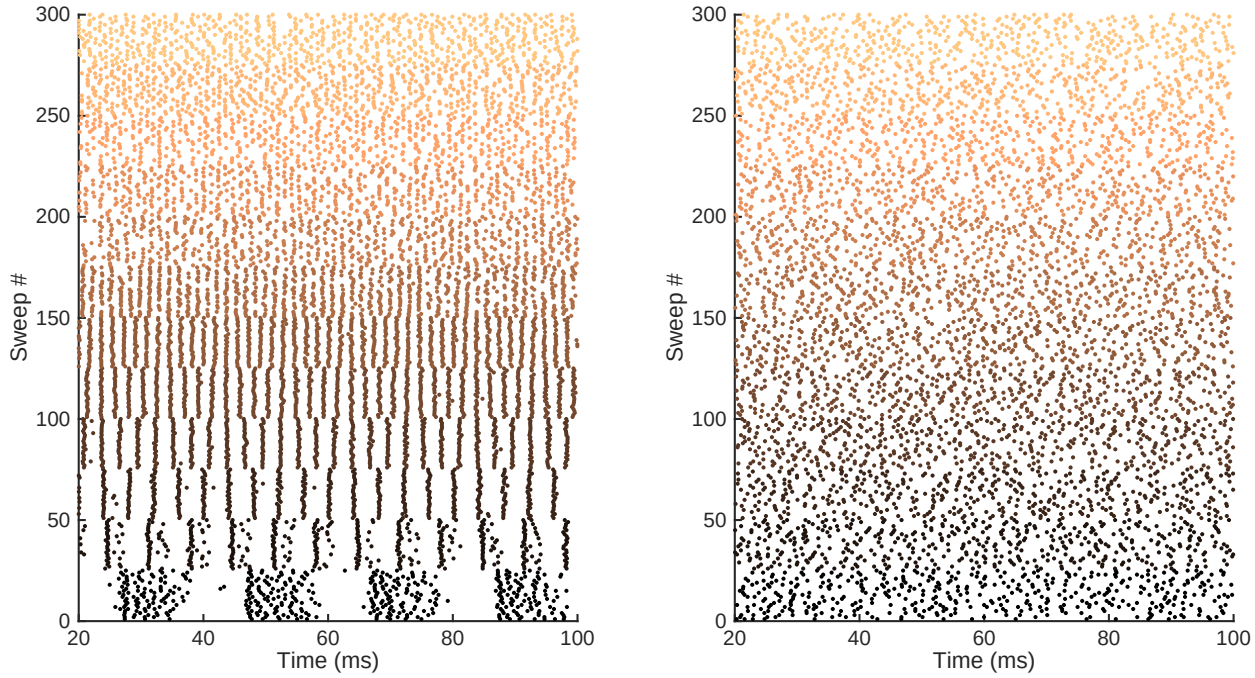


Fig. III.2 Raster plots of two spike trains datasets from two neurons for which were obtained the highest (left) and lowest (right) percentage of correct classification. Each horizontal line of dots corresponds to the spiking times of the recorded neuron; each raster plot contains 300 spike trains. Every 25 spike trains, the modulation frequency of the AM tone is incremented by 100 Hz steps (50 Hz for the bottom trains, 1150 Hz for the top ones). The right raster plot corresponds to a neuron that was apparently not even responding to sound.

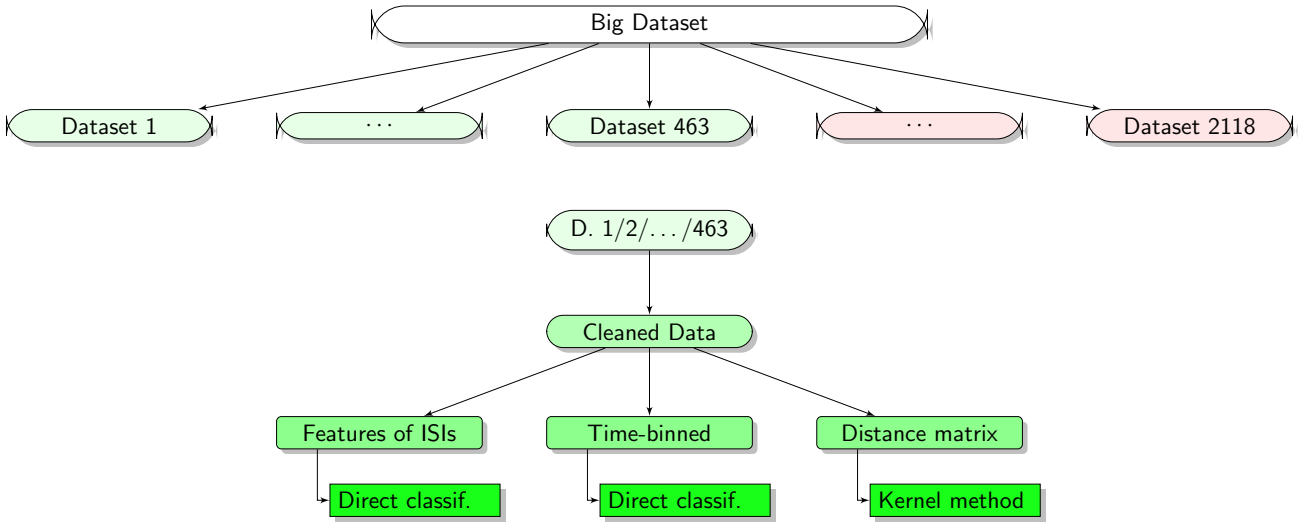


Fig. III.3 Representation of every step of data processing: Create 2118 smaller datasets and homogenise them by keeping, from the 463 ones satisfying additional conditions described in the text, only 300 spike trains (25 spike trains for each modulation frequency in 50, 150, . . . , 1150 Hz), then preprocessing and classifying each of them.

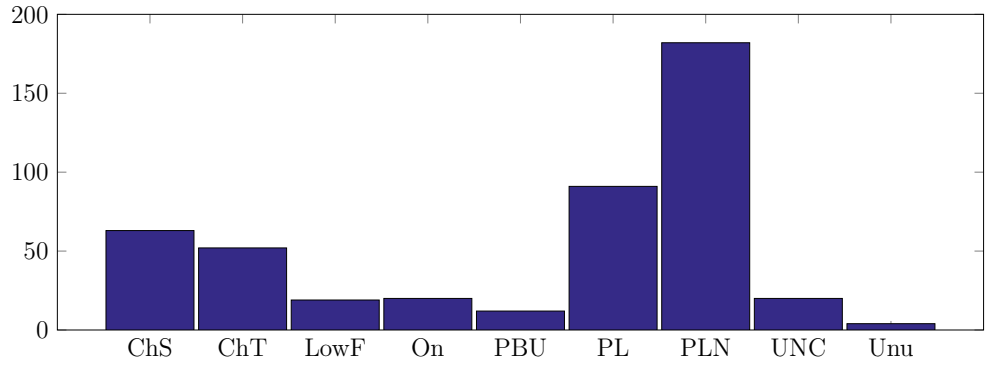


Fig. III.4 Histogram showing the number of small datasets corresponding to each unit type; 463 in total. The abbreviations stand for: Sustained Chopper, Transient Chopper, Low Frequency, Onset, Pause-Build-Up, Primary-Like, Primary-Like Notch, Unclassified, Unusual. Unit types were taken from [Blackburn and Sachs, 1989], explained in section I.1.2.1.

(f_m). Further analysis presented in appendix 11.1 leads to three more conditions:

- Each of the spike trains considered should contain at least 2 spikes between 20 ms and 100 ms, as only this part of the spike train is kept for analysis;
- For each modulation frequency, a dataset should contain at least 25 such spike trains corresponding to this modulation frequency;
- A dataset should contain 25 such responses corresponding to the following modulation frequencies: 50 Hz, 150 Hz, 250 Hz, ..., 1050 Hz, 1150 Hz.

In this setting, only 25 spike trains were kept for each modulation frequency given above. This provided a set of 463 datasets, each containing exactly 300 spike trains: 25 for each of the above list of modulation frequencies, all from a single animal, a single neuron, using a single modulation level and carrier frequency. Two of these datasets are presented in figure III.2 as raster plots. Finally, because there were only a few of each - not enough to consider each class individually, the three types of 'Onset' units were relabelled into one: Types 'OnL', 'OnI' and 'OnC' were relabelled 'On'. This provided a fairly homogeneous framework, in which to compare the results of running the same analysis on each dataset. Despite the number of datasets for each unit type not being homogenous, as seen in figure III.4, all this data were kept to provide the highest statistical power.

III.2.3 Preprocessing

Spike trains can be processed in many ways. All that is needed is a representation that the classification algorithms (see chapter II) can take as input. Since the Data Mining toolbox Weka (II.8) was used, there are two possibilities, regarding the form of the representation: a list of features (each having same length), or a kernel matrix. Three representations are used:

Interspike Interval Features This preprocessing is parameter-free. A fixed list of features is extracted from the list of interspike intervals (the list of times between two consecutive spikes), such as mean ISI, variance and coefficient of variation. Features chosen were 'standard' spike train or statistical measures; see the full list in appendix 11.2.2. Thus selecting useful features was left largely to the classifiers. Since the Vector Strength equals the magnitude at a particular frequency normalised by the mean firing rate [Joris *et al.*, 2004], statistics from the Fourier spectrum of the ISI encompass the Vector Strength. The complete list of features is given in appendix 11.

Time-binning Time-binning is the discretisation of the spike patterns into a sequence of integers, with a sample taking a value of one where a spike appears: a bin size is first chosen, which will be a parameter, and the number of spikes falling in the same bin are counted. For example, if the bin size is 1 ms, this binning makes 80 bins because all spikes over 80 ms were kept, most of them containing a 0, a certain number having a 1, and maybe some 2s if the bin size is bigger than the neuron's refractory period. If the bin size is small, there will be a lot of 0s and a 1 per spike, and this representation is somehow sensitive to the spike timing, whereas a much bigger bin size is no longer sensitive, as shown in figure III.5. Note that one can construct methods for finding an optimal bin-size [Shimazaki and Shinomoto, 2007] thus making the binning approach parameter-free.

Spike Metric Kernel methods represent a large part of Machine Learning since they allow computations to be performed without explicitly representing the data used in a high-dimensional setting, as shown in subsection II.5.2. These implicit computations are made possible thanks to the Kernel Trick: by computing distances between objects, one can already do many things, such as classifying data, thus allowing researchers to compare structures without projecting them first in a high-dimensional space. Support Vector Machines (SVMs) adopt this approach, which is the reason why a lot of theory has been be developed for SVMs. This approach applied to spike trains is now presented.

From a list of 300 spike trains $x_1, \ldots, x_{300}$ and a distance to compare two spike trains

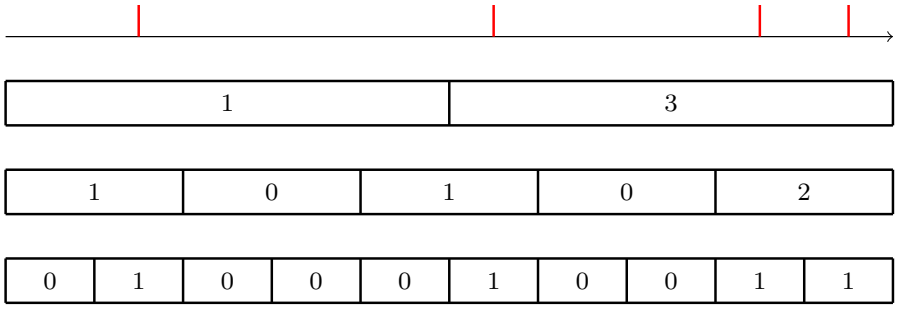


Fig. III.5 Time-binning of a spike train (in red) using three different time bins.

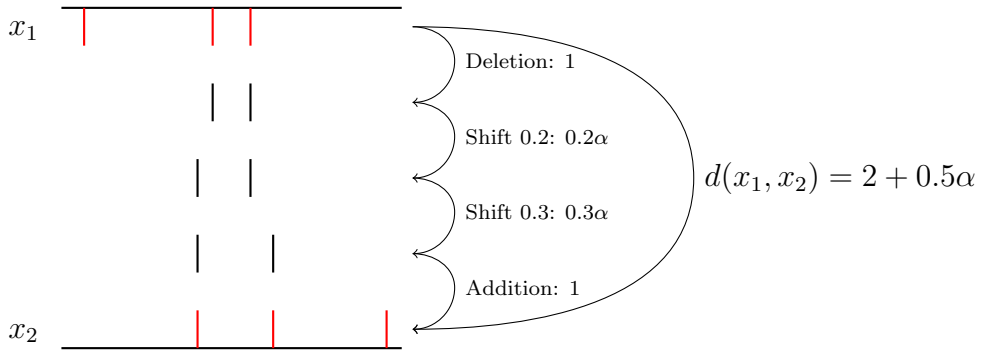


Fig. III.6 To compute the Victor and Purpura metric [Victor and Purpura, 1997] between two spike trains x_1 and x_2, one needs to find the transformation from one to the other having minimal cost. This cost is the sum of all atomic operations required for this transformation, with the following rules: Deleting or adding a spike costs 1, shifting a spike by δ costs $\delta\alpha$ (where α is a parameter). Here, $d(x_1, x_2) = 2 + 0.5\alpha$ if $\alpha \leq 8$, after which it's less costly to delete and add the two middle spikes than to shift them, in which case $d(x_1, x_2) = 6$.

d, one computes the distance matrix $D = (d(x_i, x_j))_{i,j}$. As a spike metric, the Victor and Purpura metric [Victor and Purpura, 1997] is used, using the mex files downloaded from the SPIKY software [Kreuz *et al.*, 2014], and a range of costs are tested, which is this metric's parameter. This metric is illustrated and discussed in figure III.6. Each cost gives us a distance matrix D, from which we obtain a kernel matrix $K = (k(d(x_i, x_j)))_{i,j}$ for a kernel function k. The matrix K is then the input to a classification algorithm. Among the algorithms used, only SMO was able to receive kernel matrices as input.

In order to limit this study to at most one parameter for each processing, the kernel function k used for the spike metric needs to be fixed, to vary the other parameter of interest: the metric's cost, which plays the role of the inverse of a time-scale by giving a bigger or smaller penalty when jittering the spikes. The functions we tested as kernel, given as functions of the variable x, are $\exp(-x^2)$, $\exp(-x/\gamma)$ for $\gamma \in \{1, 2, 5, 10, 20, 50, 100\}$, and the identity function. The result shown in figure III.7 was obtained on a single spike train dataset that previously allowed high accuracy of classification in [Scholes *et al.*, 2015]. The function $k(x) = \exp(-x/20)$ was selected for achieving maximal performance. Just like the SMO algorithm, it is implicitly used for all results involving the spike metric approach in the rest of this thesis.

III.2.4 Classification

As seen in chapter II, a natural output for a classifier is a confusion matrix, also called contingency table or error matrix. If a classifier is given 12 classes, the confusion matrix it outputs is an integer-valued 12×12 matrix, with each column corresponding to instances of predicted classes, while each row corresponds to the instances of an actual class. The

integer at line i and column j then represents the number of instances of the ith class that have been predicted as belonging to the jth class. This means that the number of correctly classified instances can be instantly read on the confusion matrix diagonal. All classification results in this chapter were obtained by stratified 10-fold cross-validation.

The raster plot of figure III.8, showing 25 instances for each of the 12 classes 50 Hz, 150 Hz, ..., 1150 Hz, is a typical confusion matrix for the datasets used: the spike trains at the top are quite easy to recognise, meaning that a good classifier should be able to get almost only 25s or so on the diagonal corresponding to low modulation frequencies. Indeed, up to 350 Hz, an almost perfect score is obtained (first four elements of the diagonal are 24, 24, 25 and 25). For higher frequencies, the data is more noisy, and one can read from the confusion matrix that the classifier is mostly trying to guess the right class among the high frequencies, which makes sense because their responses are very similar.

After preprocessing, 32 classification algorithms available in Weka were used, often with their default configuration. For a short description of each, see appendix 11.2.3. Only one was able to classify using the spike metric preprocessing. This algorithm — called SMO for Sequential Minimal Optimization — will be further discussed in the 'Results' section.

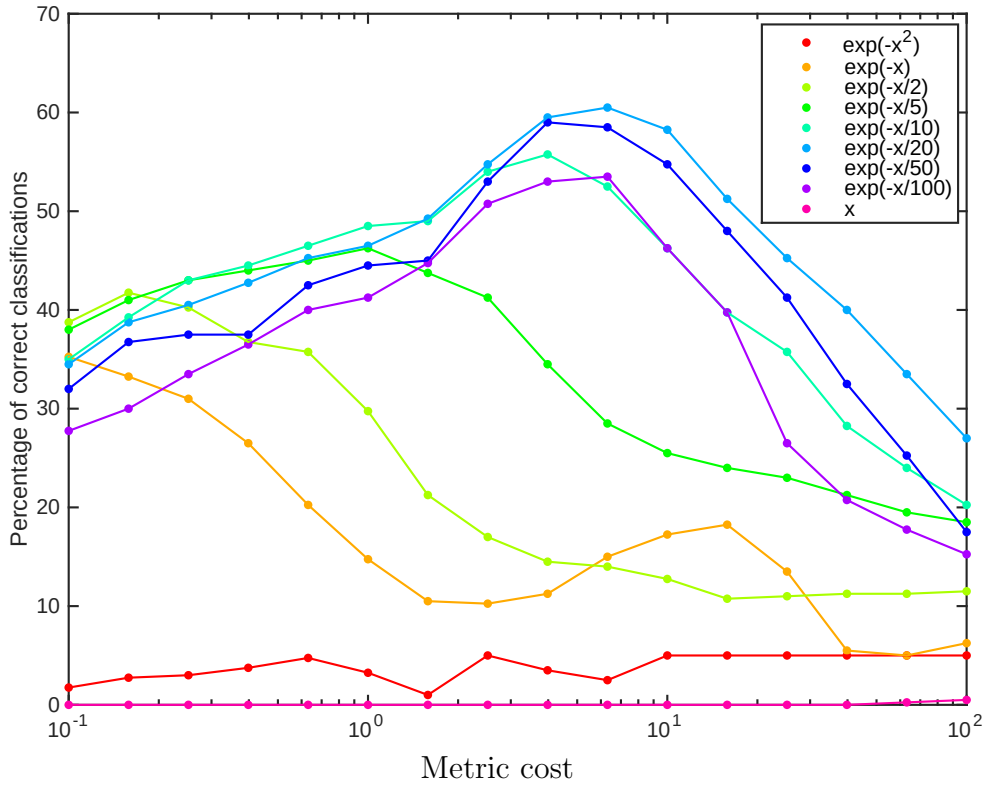


Fig. III.7 Results using SMO and the spike metric approach with different kernel functions, varying the Victor & Purpura metric's cost. For each colour, a kernel function: $k(t) = \exp(-t/\gamma)$ for $\gamma \in \{1, 2, 5, 10, 20, 50, 100\}, k(t) = \exp(-t^2), k = Id$ is applied, element-wise, to the distance matrix. Results obtained by cross-validation on a dataset of 400 spike trains.

a	b	c	d	e	f	g	h	i	j	k	l	<-- classified as
24	0	0	0	0	0	0	0	0	0	1	0	a = 50
0	24	0	0	0	0	0	0	0	0	1	0	b = 150
0	0	25	0	0	0	0	0	0	0	0	0	c = 250
0	0	0	25	0	0	0	0	0	0	0	0	d = 350
0	0	0	0	22	0	0	0	0	1	2	0	e = 450
0	0	0	0	0	22	0	0	1	0	1	1	f = 550
0	0	0	0	0	0	19	1	1	2	0	2	g = 650
0	0	0	0	0	0	1	18	1	1	2	2	h = 750
0	0	0	0	0	1	1	1	9	6	5	2	i = 850
0	0	0	0	1	0	2	0	1	11	6	4	j = 950
0	0	0	0	0	0	1	2	4	3	7	8	k = 1050
0	0	0	0	1	0	1	0	4	1	9	9	l = 1150

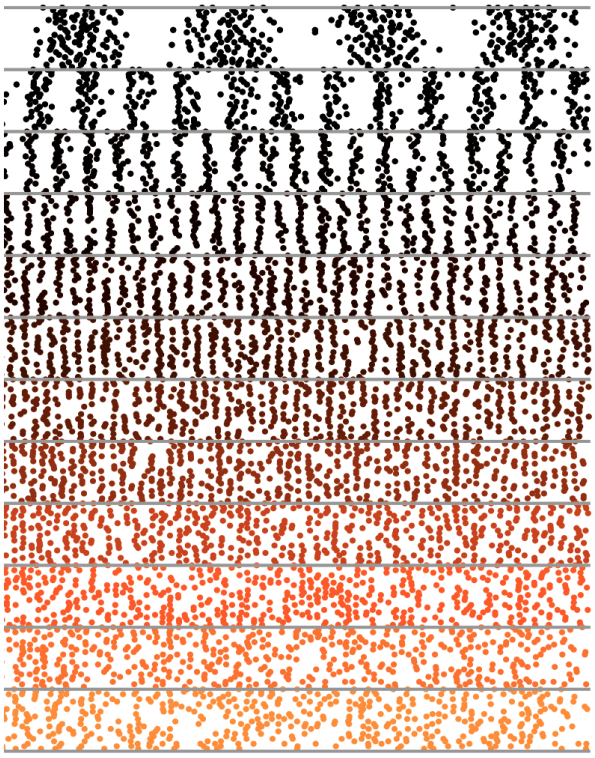


Fig. III.8 A typical confusion matrix for a classification on the spike train datasets, as output by Weka (left) and the raster plot of the data being classified (right), coloured by modulation frequency (labels for the classifiers). Each line of the matrix sums to the 25 instances for each of the 12 classes $\{50, 150, \ldots, 1150\}$, corresponding to the 300 spike trains seen on the right.

Since there are too many classifiers to discuss, it is more natural, in this chapter, to simply consider each algorithm as a black box.

From each classification we kept a few measures directly output by Weka; these are given in the next subsection and (partly) compared in the 'Results' section. Different measures may reflect different aspects of the classifications, leading to different behaviours. In this case, it would not make sense to talk about an overall best classifier. Chapter IV elaborates on the very matter of those different measures.

III.2.5 Measures of performance

A measure of performance for a supervised learning algorithm is any statistic that evaluates how well the algorithm performed on a dataset. We consider for each classification a small list of measures of performance output by Weka and tested some measures computed on the confusion matrices. We mainly focus on accuracy (proportion of correctly classified

instances) in this chapter, since it agrees with many of the other measures; see chapter IV for their definition, and their comparison:

- The percentage of correct classifications (accuracy, pctCorrect or P_{CG});
- The kappa statistic (kappa, κ);
- The area under the ROC curve (weightedAreaUnderROC, AUC);
- The Kononenko & Bratko Mean information score (K&BI, KBI, KBMeanInformation);
- The Mutual Information (MI).

III.3 Results

Here we present the results of numerous classifications (397,717 in total, with 859 classification for each of the 463 datasets; see figure III.4) by answering a list of questions that were made as independent as possible. Those questions aim at reflecting the coding differences between neuron types at the population level, and to separate what is learnt about the neural code from what is learnt about the classification algorithms. First we present a comparison of the classifiers' average scores, a comparison of the effect of the parameters on SMO average score, a correlation between the mean accuracy for each classifiers and neuron types, and an overview of the repartition of the scores across unit types. Lastly, we present an evaluation of the average cut-off frequency for each type of neuron and a correlate of different classifiers and preprocessing, the latter being the main result.

III.3.1 Ordering Classifiers

It is normally hard to define a 'best classifier' as it depends on the measure of performance and will not be universally best on all datasets. To answer this question in the current context, the average of each classifier's result is compared on all datasets and all processing. The only exception is SMO, since it is the only one containing results using the spike metric approach. As such, results from SMO are split into three groups, one for each processing (SMO.SpkM, SMO.ISI, SMO.TB) while only two groups (ending with .ISI or .TB, for example IBk.ISI and IBk.TB) give the averages for those two processing on the other classifiers. As a measure of performance, this chapter focuses on the accuracy. The only figure containing results obtained with other measures is the first one, figure III.9, kept to show how much the accuracy, the Mutual Information, the K&B Information and the area under the ROC curve are correlated, thus motivating further analysis. All other results related to the different measures of performance are given in chapter IV. These averages, presented in figure III.9, are summarised here. To improve readability, for each measure of performance, values were renormalised on the figure so that the minimal mean value would be 0, and the maximal 1.

A naive view of figure III.9 leads to the conclusion that the best classifier is SMO with the spike metric approach, second best is NaiveBayes, and in third comes SMO with the

time-binning. In fact, since the spike metric and the time-binning approaches used different parameters, an average of the values depend on the mean effect of a change in the parameters, preventing a direct comparison between SMO.SpkM and the other classifiers. There is a bias towards the spike metric as more parameters among the ones tested are close to the optimal one, as is investigated in more detail in figure III.11, where the focus is on results using SMO. The wide standard deviation observed in figure III.9 is due to both the vast range of parameters and the varying amount of AM information found in different spike train datasets. The only cases with low standard deviations are using ZeroR, which is the constant baseline, and MulticlassClassifier and SMO under the KBI score, which is investigated in chapter IV.

In practice, computational cost is often balanced with sensitivity to measure. Since NaiveBayes is very good for all tested measures, is faster than SMO and doesn't require computing a matrix of spike distances (which is quite resource demanding), it's a good place to start for a classification work similar to this one, whenever the strong conditions for applicability hold (independence of features).

III.3.2 Optimal Classifier per Neuron Type

This question of whether the appropriate choice of classifier would depend on the response characteristics of the neuron was one of the main motivations for this study. If the answer was yes, it would possibly enable one to build a dictionary to determine which algorithm to use for each unit type. This may give an interesting glimpse into different neural codes, affiliated to different classification strategies.

Each dot on the plot III.10 corresponds to a single unit type and a single classifier and processing (horizontal axis) over which is computed the mean percentage of correct classification across all classifications, represented on both the z-axis and the colormap. As previously, there are three SMO-related set of results, and two for all other classifiers. Both the unit types and the classifiers were ordered in increasing order of overall mean accuracy, for visualisation purposes. ZeroR is Weka's classifier for the baseline, classifying according to the prior probabilities only. Since the datasets are homogenous, each of the 12 modulation frequencies having 25 instances (spike trains) in each dataset, all confusion matrices and accuracies obtained by ZeroR on those datasets are the same.

Overall, there is no observed preference for classifiers to specific unit types. The only exception is that SMO when using the spike metric approach performs fairly badly on the PBU units, relative to the other unit types.

The result suggests that, on the whole, one can pick a classifier and, while the headline performance might vary, any conclusions or comparisons across different responses characteristics are likely to be valid. In figure III.10, the pattern is visually clear: some classifiers (horizontal x-axis) are better than others (warmer colours and position on the vertical axis), but the changes between two unit types (y-axis) are very similar between classifiers. Since

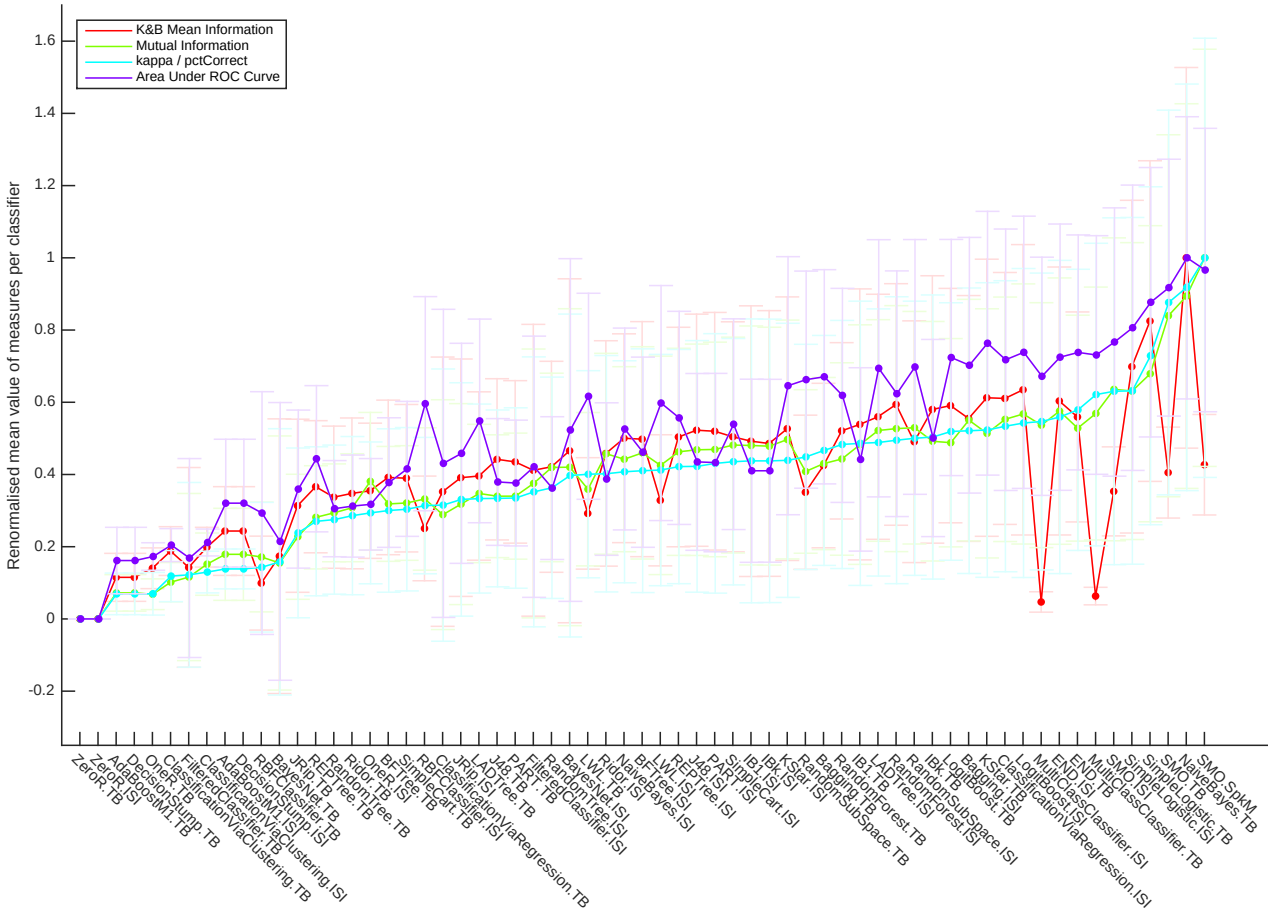


Fig. III.9 Overall mean performance of classifiers and processing methods ±1STD, sorted by increasing order of mean accuracy. Curves are renormalised between 0 and 1 to be able to compare them on a common scale. Accuracy means equivalently kappa or percentage of correct answers, since up to renormalisation they are equal. Four measures of performance associated to a colour each, are given for each classifier and preprocessing: the K&B Information, the Mutual Information, the accuracy and the area under the ROC curve. Classifiers, in abscissa, carry the preprocessing (ISI, TB or SpkM) after a dot. For TB (time-binning) or SpkM (spike metric), the result is the average over all parameters tested and all neural datasets. The ISI approach being parameter-free, the corresponding results are obtained by averaging over all datasets.

and we can use a kernel method with SMO and since it is able to extract more information from the spike trains than other algorithms, the focus is on this next.

III.3.3 Parameters with SMO

In figure III.9, SMO was split into SMO.SpkM, SMO.TB and SMO.ISI. The average difference between the spike metric approach and the two others (ISI and TB) is clear: the mean

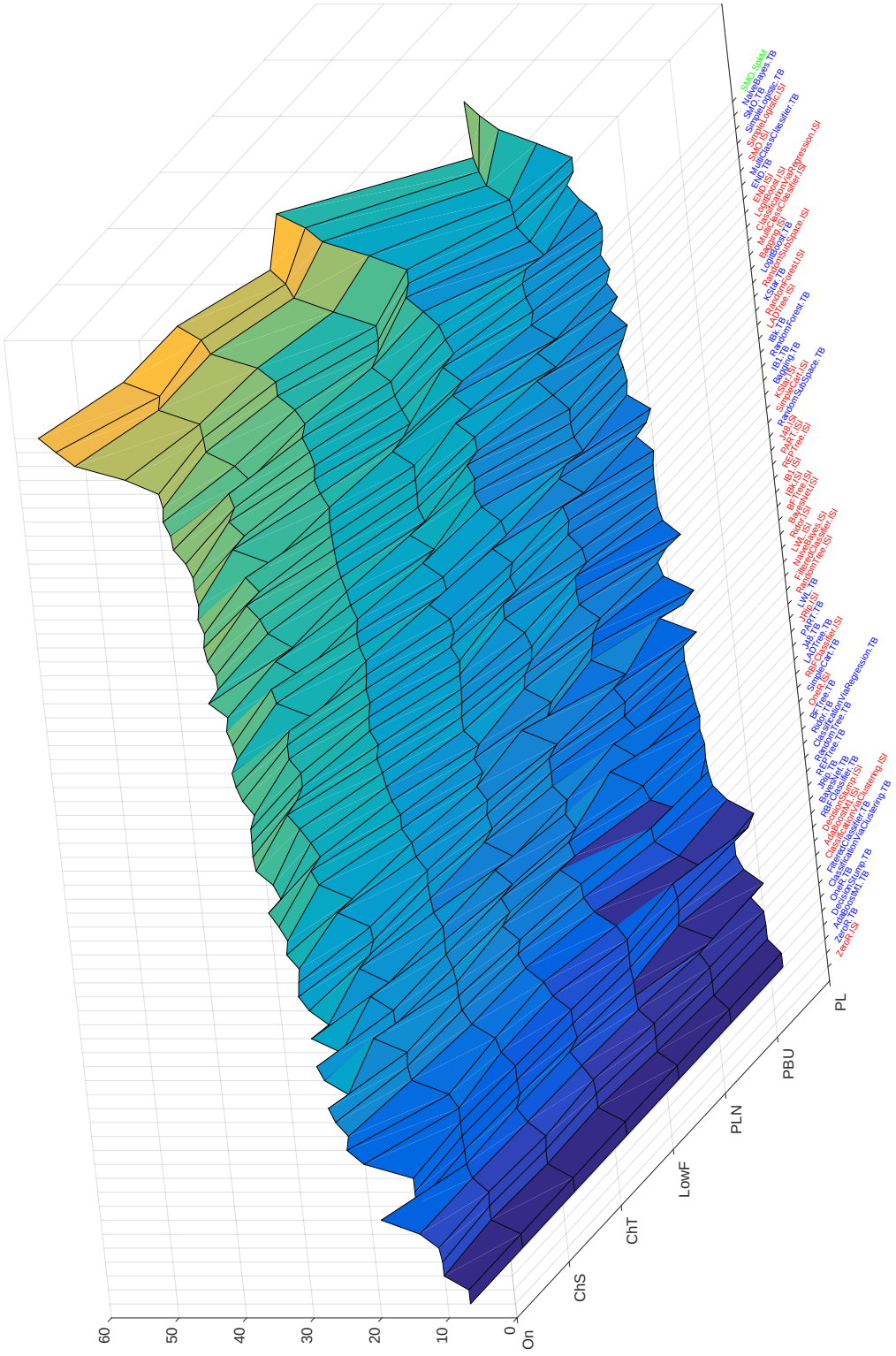


Fig. III.10 Mean accuracy (z-axis) of all classification algorithms and preprocessing (x-axis, names coloured by processing) and unit types (y-axis).

accuracy obtained with SMO.SpkM is notably above that of SMO.TB or SMO.ISI. However, this massive averaging across datasets and parameters blurs out the potential of the classifier, and reveals nothing on how sensitive it is when varying the parameters' values. As such, the effect of the different parameters on SMO are shown in figure III.11, each value giving the averaged accuracy over all datasets of a same unit type, for each preprocessing. The spike metric's score and the time-binning's time-step vary at different scales: metric cost is in ms^{-1} and binning time-step is in ms, thus by varying each on a linear scale, the two results end up looking different. In all cases, the distribution over the parameters is unimodal.

The highest score is almost consistently obtained by maximising the spike-metric's cost, even when it overtakes the best time-binning by only a small margin (LowF). Since these scores are compared against the ISI approach, it gives insight about the importance of the timing precision in the spike trains: for any unit type for which the ISI approach would obtain the highest score, units of this type would not not convey more time-precise information than through the statistics of their spike train[1]. On the other hand, for most unit types, a much higher accuracy can be obtained by the Victor and Purpora metric, stressing the role of their temporal coding.

The peak values in the time-binned approach should provide the optimal scale to analyse the spike trains, the scale at which the information is best decoded, hence possibly the scale at which information is encoded. In every unit type, the optimal time-step is, on average, under 1 ms. This supports the notion that all types are representing envelope information with sub-millisecond precision. For each unit type, a histogram of the best parameter on the different datasets was computed, plotted in grey in figure III.11, its height being renormalised on each box of the plot. The variability is fairly large, which may naively be linked with the necessity of the auditory system to code information across different time scales; this should only be the case if the neuron conveys enough information.

In order to make a stronger case that CN neurons process information at a sub-millisecond accuracy and verify how much information the units reaching a high optimal time-step convey, the optimal accuracy for each dataset is plotted next against its associated time-step. Figure III.12 shows that all units — with the exception of a chopper neuron — achieving more than 60% of accuracy are optimised for a time-step below 1 ms, which is not represented by the histograms of figure III.12.

In this dataset, a star neuron was found, allowing a recognition accuracy of 100% for a wide range of SMO parameters. A small number of such neurons may be enough for the brain to robustly decode AM information over a wide range of frequencies.

[1]The 'Unusual' units, for which results are not presented, did fall in this category.

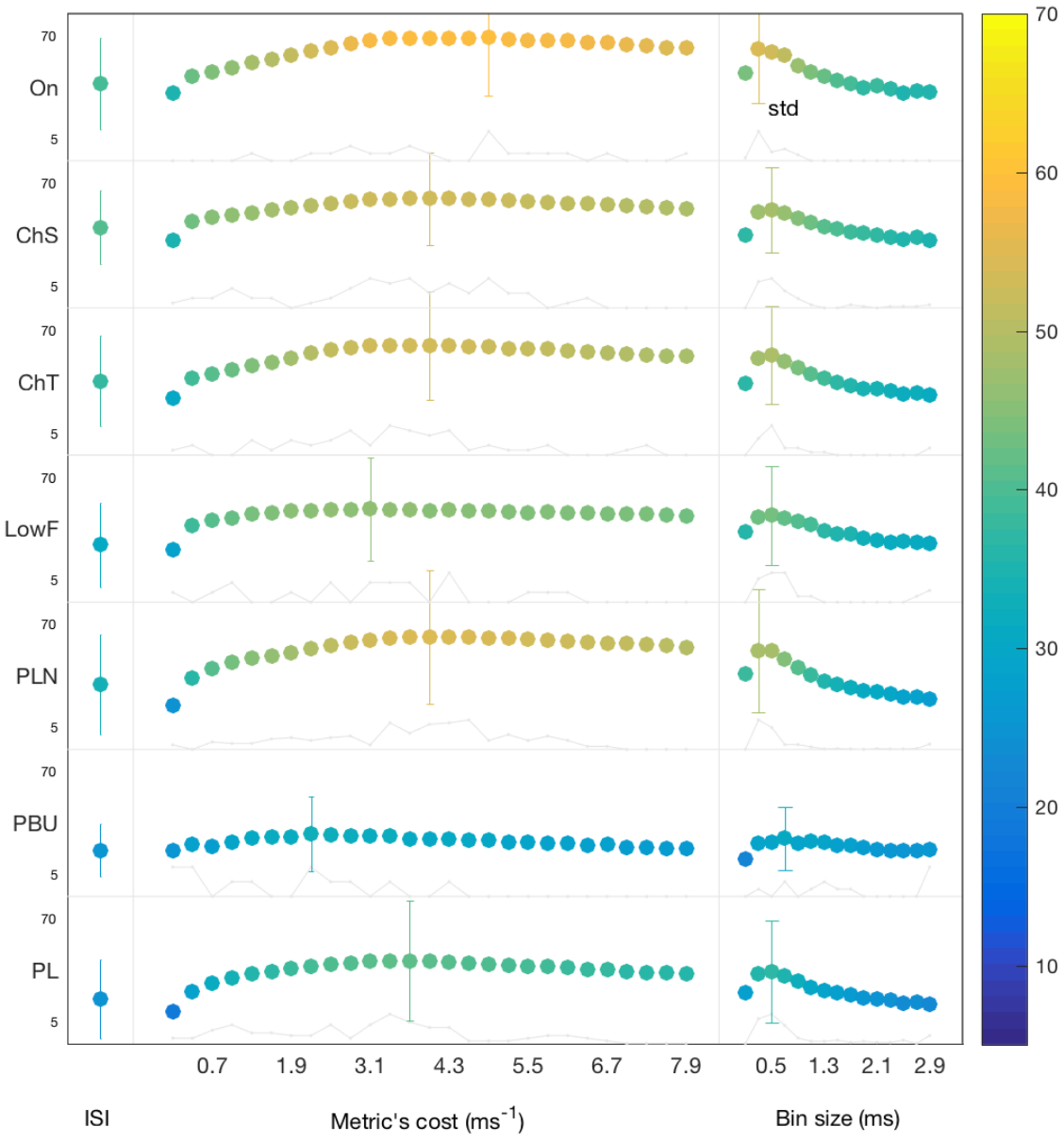


Fig. III.11 Mean accuracy using SMO as a classifier, with the three methods tested, arranged by parameter value (the ISI approach is parameter-free, and for the two others a range of parameters is tested) and neuron type. The standard deviation is shown around the highest mean value, for each unit type. Renormalised histograms of the best parameter across datasets are shown in grey within each box. Metric's cost unit is given in milliseconds^{-1}, and time-step in milliseconds. Each box shares a similar vertical axis and the same colour-code, from 5 to 70, with upper and lower margins for readability.

III.3.4 Classification Performance per Modulation Frequency

Until now, only one summary statistic was considered for each classification. However, any of these numbers is a summary of performance combined across multiple classes that are discrete samples of a continuous parameter. Thus for any unit type, the mean performance is expected to be a smooth function of the modulation frequency. This curve, called the Modulation Transfer Function[2] (MTF) is now evaluated. The present approach differs from [Rhode and Greenberg, 1994] in that confusion matrices are used instead of Vector Strength measures to derive temporal MTFs (tMTFs). These matrices act as a proxy for the evaluation of a robust AM encoding, by extraction of their diagonal values plotted on the y-axis, while the x-axis carries the modulation frequencies of the AM tone.

As exemplified by figure III.8, low modulation frequencies are usually well classified, and higher frequencies are almost randomly guessed in the cluster of high frequencies, behaving as low-pass filter MTFs. This hypothesis is tested using the best set of processing: SMO

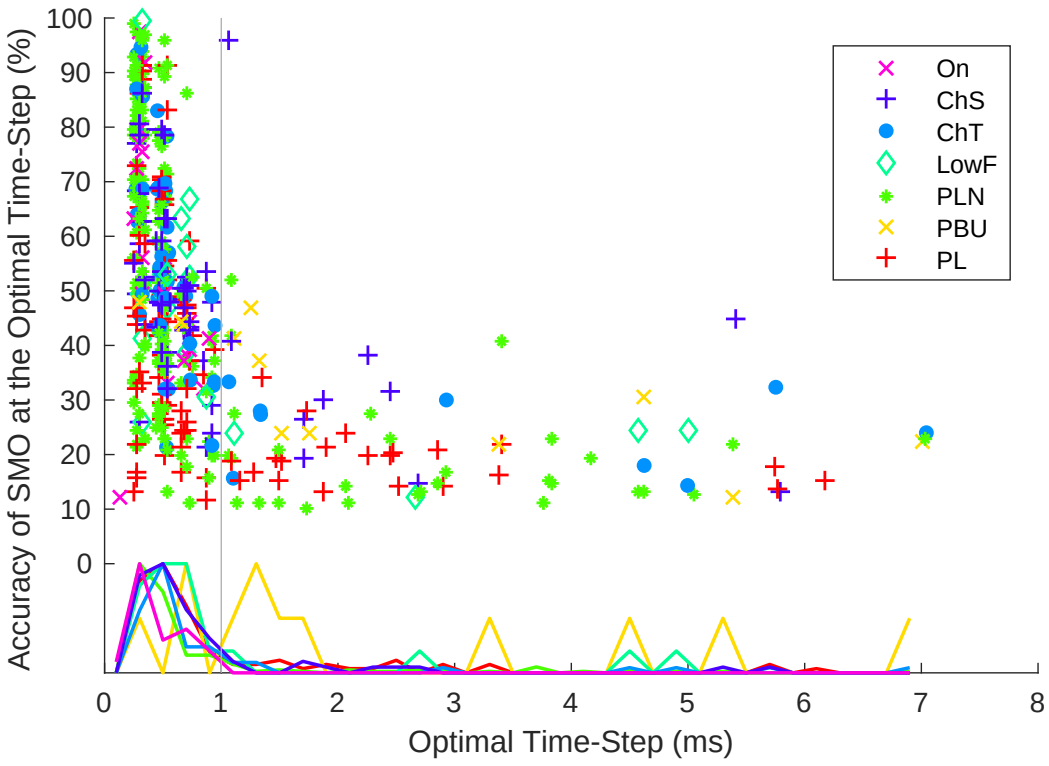


Fig. III.12 For each of the 463 datasets, the optimal accuracy for SMO.TB classifications is plotted against its associated time-step obtained for this classification, coloured by unit type. The lines represent the renormalised histograms for each unit type. Small horizontal jitter is added to help the markers' visual separation. A vertical line marks the 1 ms threshold.

[2] A CN neuron transmitting an AM tone can be viewed as a filter for the modulation frequency of the modulation. This analogy with classical signal processing motivates the use of MTF in Neuroscience.

Unit type	ChS	ChT	LowF	On	PBU	PL	PLN
Cut-off frequency (all costs)	460	480	433	485	360	449	661
Cut-off frequency (time-step in]0.1,1[)	464	498	431	529	364	500	657
Cut-off frequency (cost in]2,6[)	496	533	486	552	398	530	749
Accuracy (all costs; in %)	50	48.8	44.0	54.4	29.2	37.2	49.6

Tab. III.15 Cut-off frequencies, in Hz, obtained by averaging SMO results over a set of parameters specified in parenthesis. The last row gives the unit types' mean accuracies, multiplied by 4 to give a percentage.

with spike metric approach, and cost between 2 and 6 ms^{-1}. From these confusion matrices, the mean diagonal results are plotted in figure III.13, showing a trend of low-pass filtering. This is confirmed in figure III.14, where similar transfer functions are obtained for AM tones played at 30, 50 and 70 dB SPL, respectively. This result is not consistent with the findings in [Rhode and Greenberg, 1994] described in subsection I.2.1.3, where tMTFs have band-pass shapes at medium or high sound levels, mainly for chopper units. The right plot of figure III.13 shows that, except for PBU units, almost perfect scores can be achieved for any given modulation frequency, showing that almost all neuron types can provide the basis for discriminating modulation frequencies.

Shapes are not entirely equivalent between neural types: PBU units are the worst AM transmitters for the best classifiers, while Onset and LowF units have a relatively big increase for worse classifiers. Unit types differ in their peak performance, and in the slope of the decay with frequency. This difference is such that some unit types are best at low modulation frequencies (ChS) and others are relatively better at high modulation frequencies (PLN, On). There is great variation within unit type - most unit types have some neurons that can correctly classify any of the tested frequencies; the only exceptions being PBU and 'Unusual' units, not included in this work.

The small improvement observed in figure III.13 for each unit type at high modulation frequencies is probably a statistical bias due to the fact that spike trains corresponding to a modulation frequency of 1150 Hz are the most representative of the cluster of 'high-frequencies' responses. Thus, with its tendency to bet on 1150 Hz rather than on 1050 Hz or 950 Hz, the classifier increases its overall recognition accuracy, which it is built to do. This behaviour is also observed on many curves of figure III.14, in particular at 30 dB SPL for the Onset unit and the PBU units. A similar bias would explain why many mean accuracies are lower at 50 Hz than at 150 Hz.

As low-pass filters, the cut-off frequency ω of the mean MTF H of a given unit type is defined as

$$H(\omega) := ||H||_\infty/\sqrt{2}, \tag{III.2}$$

where the infinite norm $||\cdot||_\infty$ is calculated here as the maximum over the 12 different frequency values. Since all MTFs are monotonic in this region, cut-off frequencies were ob-

tained by piecewise-linear interpolation of the H functions. This results in the third row of table III.15, corresponding to the stars' abscissa on the top left of figure III.13.

Table III.15 shows that cut-off frequencies decrease when the set of confusion matrices considered to calculate it include classifications that had a lower accuracy. This decrease shows that, on average, the accuracy starts to decrease on the high-frequencies, confirming that classifications with an medium score tend to 'guess' classes from the high modulation frequencies rather than learning features to segregate them. The relative cut-off frequencies between unit types is, however, fairly robust.

III.4 Conclusions

In this chapter, a large set of classification algorithms was used and compared to assess how well modulation frequency is encoded in the responses of Cochlear Nucleus neurons to Amplitude Modulated tones, using three different preprocessing methods: summary statistics feature vector on the Interspike Intervals (ISIs), time-binned spike trains, and spike distance metric with a single classifier named SMO. The conclusions of this work can be split in two categories: methodological conclusions on the different preprocessing methods, classifiers and parameters, and scientific conclusions on the CN.

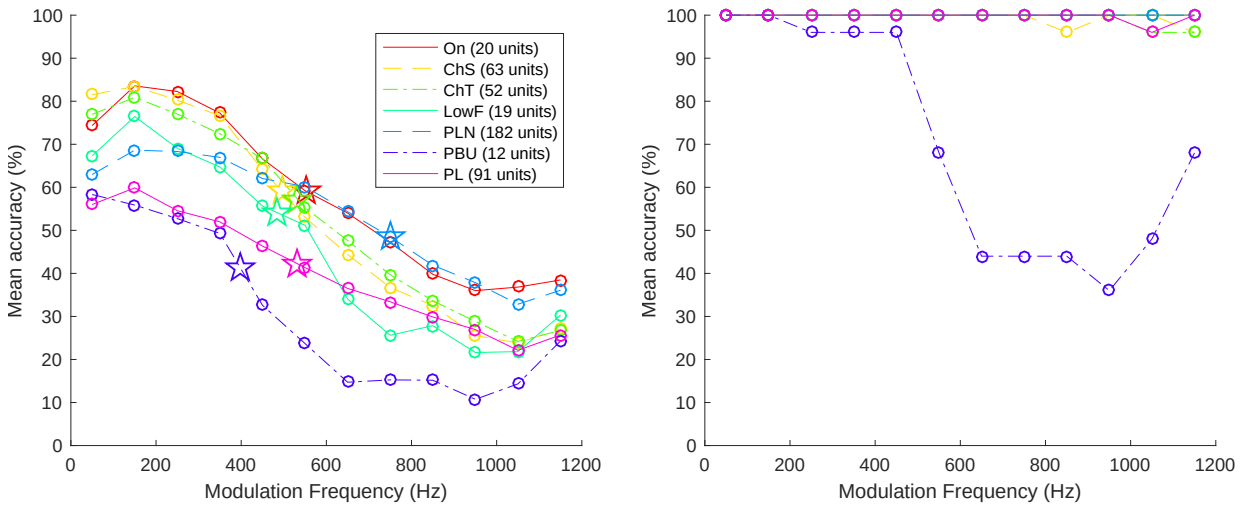
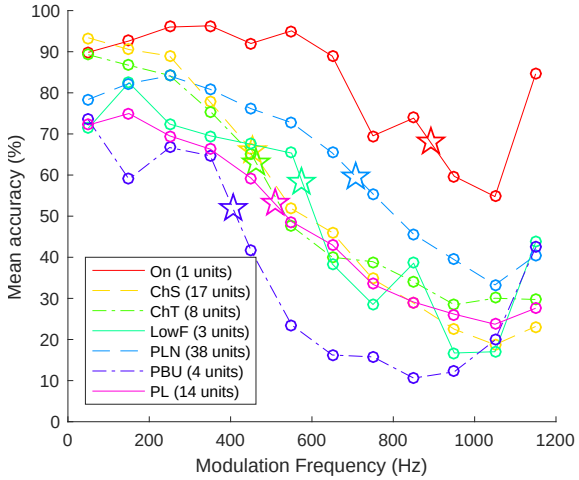
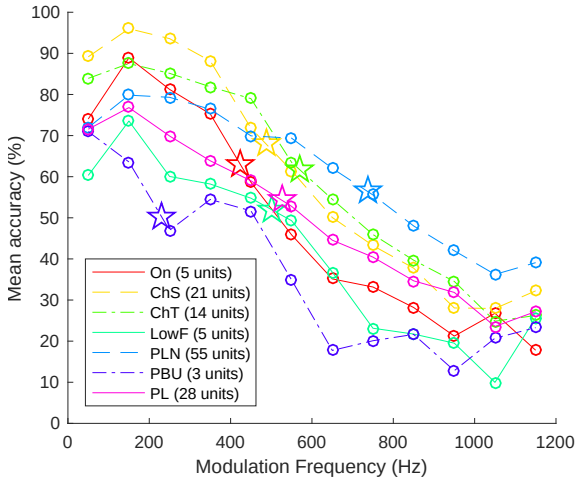


Fig. III.13 Mean (left) percentage of correct classification per modulation frequency for each unit type. Stars show the cut-off frequencies (equation III.2) All confusion matrices used here are only obtained using SMO with the spike-metric preprocessing - the algorithm giving the highest mean accuracy, and good parameters (cost between 2 and 6). Right plot shows the highest accuracy obtained for each modulation frequency and unit type.

30 dB SPL



50 dB SPL



70 dB SPL

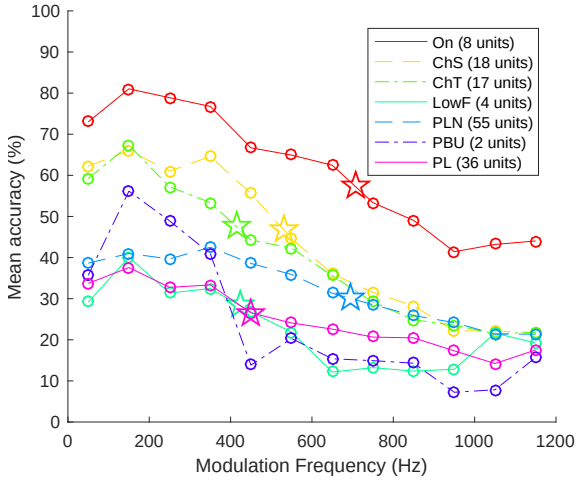


Fig. III.14 Mean percentage of correct classification per modulation frequency for each unit type. Stars show the cut-off frequencies (equation III.2) All confusion matrices used here are only obtained using SMO with the spike-metric preprocessing and good parameters (cost between 2 and 6), at the specified sound levels.

Scientific conclusions regarding the Cochlear Nucleus:
CN neurons differ reliably and systematically in their ability to signal AM frequency. Upon ranking the CN unit types, we obtain an order related to that of the rank order of [Rhode and Greenberg, 1994] for phase-locking capability to high modulation frequency, presented in table III.16, and more importantly this rank-ordering is fully compatible with the ranking of [Frisina *et al.*, 1990] since row (A) of the table implies row (D), thus stressing that the gain is an important feature in information transmission towards higher nuclei.

The Onset units, while only medium in their ability to phase-lock to high modulation frequencies, are the best AM transmitters in our setting, showing that they transmit information beyond phase-locking, using a neural code that the auditory system may be able to extract. Regular firing neurons are overall better at signalling AM frequencies. PL and PBUs are the worst unit types for AM transmission. The order was, to a reasonable extent, independent of the choice of the classifier and the performance metric used, avoiding the pathological cases where a disastrous combination of classification algorithm and performance metric would remove essential information (SMO and KBI).

This finding is robust across most measures and classification algorithms. The information that neurons encode regarding different AM frequencies is not a band-pass transfer function, as it was under synchrony analysis across virtually all CN response classes but the primary-like, this enhancement being most pronounced at moderate to high SPLs in Chopper units [Rhode and Greenberg, 1994; Frisina *et al.*, 1990]. Instead, as information processors, all CN units seem to behave as low-pass filters at all sound levels tested, and all CN neurons process information at sub-millisecond accuracy. This contrasts with the idea that chopper neurons perform some kind of temporal filtering, coding certain modulation frequencies preferentially; their increased synchrony at a preferred frequency at medium to high sound levels seems to be only part of the information these units transmit, leaving to other units — and possibly other brainstem nuclei — the role of extracting this piece of information [Joris *et al.*, 2004].

Different unit types differ in their maximal performance and their transfer function slope. Regular firing units (choppers) show good performance at low AM frequencies, but this falls off fairly rapidly with frequency. PLs are bad overall, but have a gentle low-pass slope. PLN

(A) :	On	>	ChS	>	ChT	>	LowF	>	PLN	>	PBU	>	PL
(B) :	On	>	PLN	>	ChS	>	ChT	>	LowF	>	PL	>	PBU
(C) :	ANF	>	PLN	>	PL	=	On	>	Ch	>	PBU		
(D) :	On	>	Ch	>	PLN	>	PL	>	ANF				

Tab. III.16 Rank order of CN units for AM transmission (A) on average using all classifiers, (B) based on SMO with the spike metric approach only, (C) rank order of [Rhode and Greenberg, 1994] for phase-locking to high modulation frequency, including ANF, and (D) rank order of [Frisina *et al.*, 1990] for gain modulation.

and Onset units are actually better at identifying high AM frequencies than choppers.

There is great variation within unit types. Most unit types have some examples that can perfectly identify any modulation frequency, especially PLNs. Overall, the previous findings that regular firing cells are good at encoding envelopes is confirmed [Rhode and Greenberg, 1994], but these transfer functions are low-pass when considered as information transmitters, and some irregular firing neurons do encode high frequency envelopes.

Methodological conclusions:
As a measure of synchrony, Vector Strength does not extract all the AM information output by CN neurons. Assumption-free approaches give a different picture, suggesting that more general approaches for AM decoding have the potential to be important; the rest depends on how the brain processes this information, which requires investigating.

Classifiers vary a lot in their overall performance, but in a robust way across unit types: regardless of the classifiers, the relative performance across different types of responding neuron is similar. By focusing on the results using the best classifier (SMO in the work presented, Naive Bayes coming second) it was possible to draw conclusions on the Neuroscience related questions and separate Neuroscience and Machine Learning issues. So when comparing neurons, the classifier choice is unlikely to influence conclusions — although it does strongly influence the estimate of the lower bound on how well a given neuron can do. This could be important when trying to quantitatively relate neural coding to perceptual performance.

Due to the importance of spike timing in the spike train dataset, the feature vector based on ISI derived summary statistics performed poorly, even though it contains the Vector Strength. The resolution of time-binning for algorithms is fairly crucial to maximise performance. The comparison between time-binning vs. spike distance based preprocessing showed that the kernel approach was marginally superior using a SVM.

Finally, all measures of performance are not equivalent on all classifiers, which may imply that different classifiers employ different functional strategies. This is examined in chapter IV.

IV.

Comparison of Performance Measures

This chapter studies the possible links that performance measures have with classification algorithms or with neuron types. Section IV.1 lays down the rigorous definitions of the performance metrics used and discusses some of them. These measures are then compared in section IV.2, revisiting the results of chapter III and further discussed. The main contribution presented there, is the comparison of different measures of classification obtained by applying different classifiers.

IV.1 Performance Measures for Classification Algorithms

This technical section lays down the definitions of traditional classification measurements output by Weka, discusses those definitions, and theoretically compares the measures to some extent.

IV.1.1 Definition of Classical Measures

Let $\mathcal{C}$ be a trained classification algorithm, given an abstract algorithm $\mathcal{A}$ and some training data $\{\tilde{x}_1, \ldots, \tilde{x}_{\tilde{n}}\}$ each having an associated label (or class) $\{\tilde{l}_1, \ldots, \tilde{l}_{\tilde{n}}\}$. The data to be classified is a new set of test points $\{x_1, \ldots, x_n\}$ labelled $\{l_1, \ldots, l_n\}$. For each test instance, it is assumed that the classifier outputs a probability distribution that this point belongs to the known classes; when a classifier should simply output the guessed label, it is rewriten as a probability of 1 of belonging to this class and 0 over other classes. From these n test points, working with n_l different labels, the list of output probability distributions is put in a so-called probability matrix $P \in \mathcal{M}_{n \times n_l}(\mathbb{R}^+)$ such that each row sums to 1, from which a confusion matrix $C \in \mathcal{M}_{n_l \times n_l}(\mathbb{N})$ is extracted. All classification performance measures defined below are based on C and P. The following notations are used:

- L_p is the natural entrywise norm on matrices, not the operator norm; only $p = 1$ and $p = 2$ are used. For $M \in \mathcal{M}_{n,m}(\mathbb{R})$ and $p \in \mathbb{N}^*$,

$$||M||_p := \left(\sum_{i,j} |M_{i,j}|^p \right)^{\frac{1}{p}};$$

- $\mathcal{C}$ denotes a classifier, already trained,
- $\log_2$ is the logarithm in base 2, such that $\log_2(2) = 1$;
- $\mathbb{L} = \{1, \ldots, n_l\}$ is the set of all labels used (n_l labels);
- $\tilde{x} = \{\tilde{x}_1, \ldots, \tilde{x}_{\tilde{n}}\}$ is the training set of data;
- $x = \{x_1, \ldots, x_n\}$ is the test set of data;
- $\tilde{L} = \{\tilde{l}_1, \ldots, \tilde{l}_{\tilde{n}}\}$ is the set of labels of training set;
- $L = \{l_1, \ldots, l_n\}$ is the set of labels of test set, the order being fixed for consistency of the discussion;
- C denotes a confusion matrix: $C(i, j)$ is the number of test instances of the ith label that are classified as belonging to the jth class; $||C||_1 = n$ is the number of test data points;

- P_C denotes the confusion matrix C after renormalisation: $P_C = C/||C||_1$;
- $P \in \mathcal{M}_{n\times n_l}(\mathbb{N})$ is a probability matrix. Its entry $p_{i,j} = P(i,j)$ is the probability that the ith data point x_i has the jth label;
- $\pi_r(j) = \pi_r(x_i \in \text{ class } j)$ is the prior probability distribution (of class j) that any test point belongs to class j. It is the observed proportion of instances of class j in the training set:
$$\pi_r(j) := \frac{\#\{i \in \{1,\dots,\tilde{n}\} \ / \ \tilde{l}_i = j\}}{\tilde{n}};$$
- $\mathbb{S}_1$ is the unit sphere for the L_1 norm on matrices: $\mathbb{S}_1 = \{M \in \mathcal{M}_{n_l}(\mathbb{R}), ||M||_1 = 1\}$;
- $\mathbb{S}_1^+$ is the subset of $\mathbb{S}_1$ restricted to matrices with positive elements:
$$\mathbb{S}_1^+ = \{M \in \mathcal{M}_{n_l}(\mathbb{R}^+), ||M||_1 = 1\}.$$

The classification is evaluated using various classical measures, presented below:

Percentage of correct guesses (P_{CG}): Percentage of correctly classified instances among n test points:

$$P_{CG} = \left\{ \begin{array}{ccc} \mathcal{M}_{n_l}(\mathbb{N}) & \to & \{0, n^{-1} \dots, 1\} \\ C & \mapsto & \text{trace}(P_C) = \sum_{k=1}^{n_l} P_C(k,k) = \frac{\text{trace}(C)}{||C||_1} = \frac{1}{n}\sum_{k=1}^{n_l} C(k,k) \end{array} \right. \tag{IV.1}$$

For notational convenience, values are not multiplied by 100, so that the elements of P_{CG} are between 0 and 1, and $P_{CG}^{100} = 100 * P_{CG}$ refers to the matrix whose elements are percentages. This measure simply looks at the percentage of correct guesses of the classes, regardless of how the classifier made mistakes. This very naive measure is the most commonly used in the literature, but does not make use of all the information present in the confusion matrix. This measure is also called accuracy.

kappa (κ): This measure is equivalent to the accuracy, up to a renormalisation and shift. It reflects the agreement of the prediction with prior classes, reducing the necessity to compare the results with a baseline:

$$\kappa = \left\{ \begin{array}{ccc} \mathcal{M}_{n_l}(\mathbb{N}) & \to & [-1,1] \\ C & \mapsto & \frac{P_{CG}(C) - P_r(P_C)}{1 - P_r(P_C)} \end{array} \right. \tag{IV.2}$$

where, using the column vector $\mathbf{1} = (1,\dots,1)^t \in \mathbb{R}^n$,

$$P_r = \begin{cases} \mathbb{S}_1^+ & \to & [0,1] \\ P_C & \mapsto & \mathbf{1}^t * P_C * P_C * \mathbf{1} = \sum_{k=1}^{n} \left[\left(\sum_{i=1}^{n} P_C(i,k) \right) * \left(\sum_{j=1}^{n} P_C(k,j) \right) \right]. \end{cases} \tag{IV.3}$$

This can be thought of as

$$\kappa = \frac{\text{observed agreement} - \text{chance agreement}}{1 - \text{chance agreement}},$$

because $P_{CG}(C)$ is the proportion of correctly classified instances ('observed probability that the classification was correct'), $\sum_{i=1}^{n} P_C(i,k)$ is the proportion of instances of the kth class ('observed probability that the class was k'), and $\sum_{j=1}^{n} P_C(k,j)$ is the proportion of instances classified as k ('observed probability probability that we classify an instance as k'), which means that the 'chance agreement' for a class k will be the observed probability that an instance[1] is of class k and is classified as being of class k is $\sum_{i=1}^{n} P_C(i,k) * \sum_{j=1}^{n} P_C(k,j)$. By additivity of probability on disjoint events - here the probability of belonging to a class - the total chance agreement is given by the sum over k of these terms, hence the definition of P_r.

Mutual Information (MI): As defined in [Cover and Thomas, 2006], given two random variables X and Y with a joint probability mass $p(x,y)$ and marginal probabilities mass functions $p(x)$ and $p(y)$, the mutual information $I(X;Y)$ is the relative entropy (or Kullback-Leibler divergence) between the joint distribution and the product $p(x)p(y)$:

$$I(X;Y) = \sum_{x \in \mathcal{X}} \sum_{y \in \mathcal{Y}} p(x,y) \log_2 \frac{p(x,y)}{p(x)p(y)} = \mathbb{E}_{p(x,y)} \log_2 \frac{p(X,Y)}{p(X)p(Y)}.$$

This definition can be extended to continuous densities, although this is not needed for this thesis. This definition of the mutual entropy when applied to a confusion matrix, following [Chase and Young, 2006] becomes:

$$MI = \begin{cases} \mathcal{M}_{n_l}(\mathbb{R}^+) & \to & \mathbb{R}^+ \\ P_C & \mapsto & \sum_{i=1}^{n_l} \sum_{j=1}^{n_l} P_C(i,j) \left[\log_2(P_C(i,j)) - \log_2(P_C(i)) - \log_2(P_C(j)) \right] \end{cases} \tag{IV.4}$$

where, for simplicity, $P_C(i) := \sum_{k=1}^{n_l} P_C(i,k)$ and similarly $P_C(j) := \sum_{k=1}^{n_l} P_C(k,j)$. The efficient Matlab implementation of the Mutual Information that was used also deals with infinities:

[1] This instance being randomly chosen and randomly classified, using independent draws. This implicit assumption always made when talking about the chance agreement.

```
P = C / sum(sum(C));
mP = bsxfun(@minus, bsxfun(@minus, log2(P), log2(sum(P,1))), log2(sum(P,2)));
mP(isinf(mP) | isnan(mP)) = 0;
MI = sum(sum(P .* mP));
```

K&B Information (KBI): The Kononenko & Bratko information score [Kononenko and Bratko, 1991] is a measure that aims at excluding the influence of prior probabilities (which may enable a classifier to easily obtain high accuracy) and deal with various types of imperfect or probabilistic answers[2]. The main idea behind this measure is that: '[T]he misclassification of a more probable class should count as a more serious mistake than misclassification of a less probable class since the former is not expected while the latter would not be surprising'. This logic led the authors to attribute a credit for correct classification and a penalty for misclassification. Both functions (for credit or penalty, respectively) are based on the same function from information theory that takes into account the prior probabilities of classes.

The way they do this is by saying that the entropy of an event j with prior probability $\pi_r(j)$ is $-\log_2 \pi_r(j)$ bits, which also gives the amount of information necessary to correctly classify an instance into class j. Analogously the amount of information necessary to correctly decide that an instance does not belong to class j is $-\log_2(1-\pi_r(j))$ bits. Let's consider a data point x_i labelled j: $l_i = j$ and look at $P(i,j)$, the (posterior) probability returned by the classifier $\mathcal{C}$ that this data point belongs to class j. Let's define the K&B information score of this answer (in bits) by considering the two cases:

- if $P(i,j) \geq \pi_r(j)$, it is a useful answer, leading to a positive score:

$$\text{score}(x_i) = -\log_2 \pi_r(j) + \log_2 P(i,j),$$

- if $P(i,j) < \pi_r(j)$, it is a misleading answer, leading to a negative score:

$$\text{score}(x_i) = -[-\log_2(1-\pi_r(j)) + \log_2(1-P(i,j))].$$

Thus, the mean K&B information score is defined on the test set x as

$$KBI = \frac{1}{n}\left(\sum_{i=1}^{n} \delta_{P(i,l_i)\geq\pi_r(l_i)}[\log_2 P(i,l_i) - \log_2 \pi_r(l_i)] + \sum_{i=1}^{n} \delta_{P(i,l_i)<\pi_r(l_i)}[\log_2(1-\pi_r(l_i)) - \log_2(1-P(i,l_i))]\right) \tag{IV.5}$$

where δ denotes the Kronecker delta function on 0: $\delta_y = 1$ if $y = 0$, $\delta_y = 0$ otherwise.

[2] We understand their use of 'imperfect' as referring to simple classifiers that only use prior information.

Area Under the ROC Curve (*AUC*): The Receiver Operating Curve (ROC) was originally used in signal detection, and its first use in Machine Learning seems to be in 1989 by Spackman [Spackman, 1989].

By assumption, the classifier $\mathcal{C}$ outputs probabilities for each test instance x_i to each class. In the binary case, these two classes are often referred to as 'positive' and 'negative'. To work with a common vocabulary, those terms are used as well to describe the AUC. For each threshold $t \in [0, 1]$, each instance belongs to one of the 4 following classes depending on whether the data point is actually negative or positive, and whether the output probability that it belongs to the positive class is above or below the threshold: true positives (TP), true negatives (TN), false positives (FP), false negatives (FN) (see figure IV.1 top). The proportions (or rates) of true positives and false positives are then, respectively, functions of this threshold t, as shown in figure IV.1 middle. By taking the cumulative distribution of these two parameterised functions, a parameterised curve is defined in the square $[0, 1]^2$, as shown in figure IV.1 bottom; each is an increasing function since, by definition, cumulative distributions are increasing. This function is called the ROC. A ROC curve is considered better the closer it gets to the top left corner of the square; this can be evaluated by looking at its integral, called the Area Under the ROC Curve (AUC), which is a value between 1 (highest value) and 1/2, which is interpreted as chance[3].

Assuming for notational simplicity that there are no ties in the estimated probabilities, let's describe how the ROC curve in figure IV.1 is plotted: Let n_p denote the number of positive test points and n_n the number of negative points. The ROC curve is a step function, starting at $(0, 0)$, moving $1/n_p$ units up when the threshold t becomes equal to the probability that a point with positive class actually belongs to this class, and moving $1/n_n$ to the right when this probability becomes equal to the probability of a point with negative class. After n_p such steps up and n_n steps to the right, the curve reaches the point $(1, 1)$.

An equivalent way to compute the AUC is presented by Hand and Till [Hand and Till, 2001], replacing the use of thresholds by ranking of the probabilities. This approach seems more computationally efficient, is not burdened by potential biases following the integral calculation of the AUC, is simple to implement, and has an immediate geometrical origin. In a nutshell, Hand and Till compute the integral of the step function (the AUC) as the sum of the area of each of the n_p rectangles of width $1/n_p$ units. After ranking the probabilities vector $p = (p_1^n, \ldots, p_{n_n}^n, p_1^p, \ldots, p_{n_p}^p)$ in increasing order (where p_i^c is the probability that the ith instance of the class c be positive), one obtains the vector $r = (r_1, \ldots, r_{n_n}, \ldots, r_{n_n+n_p})$[4], from which they compute the area of the ith rectangle as $\mathcal{A}_i = (r_i - i)/(n_p n_n)$. Summing over the indices of the positive points, Hand and Till obtain

[3]The chance level is the lowest practical value, as an AUC smaller than 1/2 urges you to switch instances labels to obtain superior performance.

[4]The vector r is a permutation of the vector $(1, 2, \ldots, n_n + n_p)$ such that $\forall i < j, p_{r_i} \leq p_{r_j}$, uniquely defined if there are no ties.

$$AUC = \frac{1}{n_p n_n} \sum_{i=1}^{n_p} r_i - i. \tag{IV.6}$$

This value is equal to the empirical probability that a randomly chosen negative point has a lower estimated probability of being positive than a randomly chosen positive point, this number being known as the Mann-Whitney-Wilcoxon statistic. They also give the standard error of this statistic.

However, this definition of the AUC does not naturally generalises to multi-class classifiers. Such a generalisation is proposed in [Hand and Till, 2001] by averaging pairwise comparisons of classes, which is a common approach in Machine Learning[5]. They generalise this formula to a c-classes problem by taking the average AUC measures for each sub-binary problem:

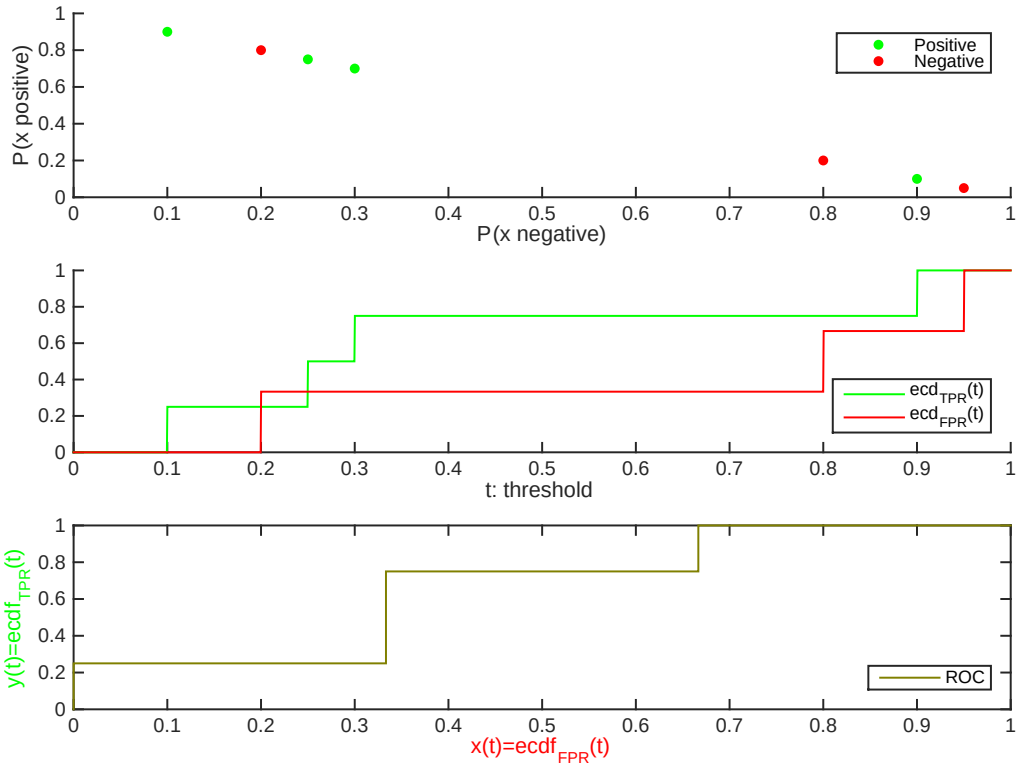


Fig. IV.1 Calculation of the ROC curve in a simple example, given the estimated probabilities on 7 data points output by a classifier. *Top*: Scatter plot of the probabilities for each test point, coloured by its real class. *Middle*: Empirical cumulative distributions of the true (green) and false (red) positive rates, as a function of the threshold t above which a test point would be classified as negative. *Bottom*: Graphical plot of the ROC as a parameterised function, which coordinates are given by the two cumulative distributions plotted above, axis coloured as the parameterised distributions.

[5]For example used in Weka's MultiClassClassifier.

$$AUC_c = \frac{1}{c(c-1)} \sum_{1 \leq i \neq j \leq c} AUC(i|j),$$

where the summation is over every coupling of distinct classes and $AUC(i|j)$ is the AUC measured on the binary classification task consisting only of points from class i (positive class) and j (negative class). From this formula, it is easy to generalise further to obtain a weighted Area Under the Curve by weighting the binary AUCs with positive weights summing to one:

$$wAUC_c = \frac{1}{c(c-1)} \sum_{1 \leq i \neq j \leq c} w_{i,j} AUC(i|j), \tag{IV.7}$$

which is in the spirit of Signal Detection Theory, where negative and positive options are not equivalent.

IV.1.2 Strategies of Measures of Performance

The present goal is to understand how measures of performance correlate when they are applied to the results of different classifiers, which requires us to understand how they evaluate a result. To give a flavour of the kind of difference, we provide a simple example, using for simplicity the 3 measures that take as input a confusion matrix (K&BI using the probabilities instead so cannot be computed from the confusion matrix alone), evaluating the three following confusion matrices:

$$C_1 = \begin{pmatrix} 10 & 0 & 0 \\ 0 & 10 & 0 \\ 0 & 0 & 10 \end{pmatrix} \qquad C_2 = \begin{pmatrix} 0 & 0 & 10 \\ 10 & 0 & 0 \\ 0 & 10 & 0 \end{pmatrix} \qquad C_3 = \begin{pmatrix} 3 & 3 & 3 \\ 3 & 3 & 3 \\ 3 & 3 & 3 \end{pmatrix}$$

In this case, the following results are obtained:

$$\begin{array}{rclrclrcl} P_{CG}(C_1) & = & 1 & P_{CG}(C_2) & = & 0 & P_{CG}(C_3) & = & 0.333 \\ \kappa(C_1) & = & 1 & \kappa(C_2) & = & -0.5 & \kappa(C_3) & = & 0 \\ MI(C_1) & = & 1.585 & MI(C_2) & = & 1.585 & MI(C_3) & = & 0 \end{array}$$

Those measures can be thought of as markers who have different strategies when marking a multiple-choice questionnaire:

- Marker 1 would only give points when the student gives the right answer (P_{CG});
- Marker 2 counts the number of correct answers, but wants the students who chose randomly to have a mark 0 in the end (κ), still considering systematic errors to be worse than the random ones;

- Marker 3 gives points if the student shows in any way that he has understood the problem, even if he gives only wrong answers - for example a stubborn student who decided to always give the solution to the right of the correct one (MI).

As shown in this elementary example, the mutual information of a confusion matrix does not have much to do with correctly or incorrectly classified instances, as its very definition is invariant under a switch of lines and columns in the confusion matrix. But since classifiers are engineered to have an optimised accuracy, it means that the 'cleverness' estimated by the mutual information should be strongly correlated with the 'correctness' of the classifier's prediction. This is indeed what is observed for classification of real neural data, as can be seen in raster plot IV.2 that shows the mutual information against the accuracy of almost 400,000 confusion matrices. These confusion matrices were obtained, through the work presented in chapter III, by preprocessing different spike train datasets in many ways and using various classifiers. An alternative view for the basis of comparing measures of performance, more philosophical than scientific, is given in appendix 12.3.

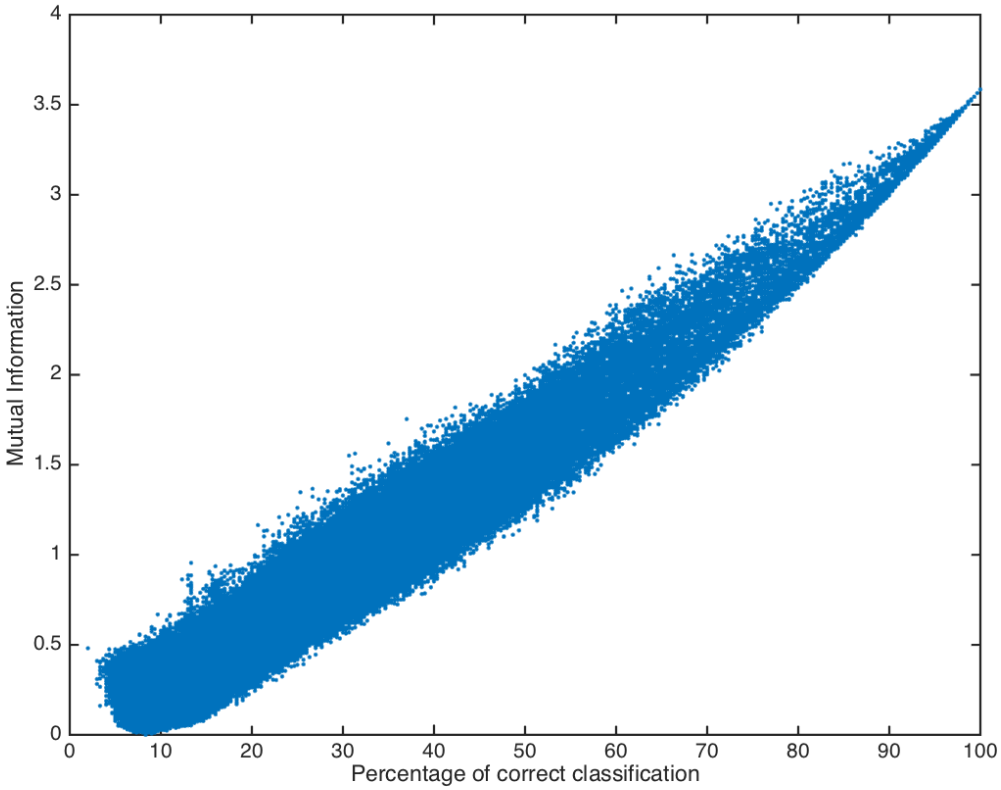


Fig. IV.2 Scatter plot of percentage of correct classification against mutual information, obtained over 397,717 confusion matrices (results detailed in chapter III).

IV.2 Results

Different measures of performance can be strongly correlated, as observed in figure IV.2 where accuracy and mutual information are used as coordinates for a raster plot of about 400,000 classification results. This was also discussed in chapter III, where figure III.9 shows the average performance for each classifier, under different measures. Mutual information and accuracy were already very close to each other, and both AUC and K&BI also followed the same trend, but were not as close, and K&BI gave very poor performances when either SMO or MultiClassClassifier were used as classifiers. The results presented here shed some light on these correlations.

Figure IV.3 shows two representations of the same information; only the colouring changes between the top 9 plots and the bottom 9 plots. Each collection of 6 raster plots and 3 histograms shows the result of each of the classifications run with one of 6 selected classifiers: SMO, END, LogitBoost, MultiClassClassifier, NaiveBayes and SimpleLogistic. Each non-diagonal subplot is the raster plot of two measures (one on the x-axis and the other on the y-axis) among the three selected: kappa, the K&BI, and the AUC. These classifiers and measures were selected to reduce the clutter: many other measures were tested in a similar manner, and would lead to much redundancy in the discussion. For example, a measure (namely the EMA[6]) turned out to be the square of another measure (MI), despite their apparent differences.

The top plots in figure IV.3 are coloured by classifier: each classifier is associated with a colour, and each point in the raster plots corresponds to a classification as described in the previous chapter, using any available dataset and preprocessing. Since each point plotted on the raster plots has an area, many plots mainly reveal the top colour, which is the SimpleLogistic classifier simply due to the order in which dots were plotted. The histograms, on the diagonal, compensate this by showing the distribution of values for each measure.

The histograms of kappa and K&BI have approximately the same shape. The main differences are that the first bin for K&BI is much higher, and the tail of kappa is heavier. The AUC histogram is more uniform, spreading more evenly this large number of measures, making it a better candidate measure to compare the results from many classifiers, while kappa and K&BI are more skewed.

On each raster plot, distinctive shapes emerge. On the bottom left plot, where kappa is plotted against AUC, all shapes seem to be on top of each other, with at least a slight shift of SimpleLogistic towards the top since other colours are only seen near the diagonal for values of kappa above 0.2. This is confirmed by how colours are spread on the kappa-histogram, where there is a high number of high values for SMO, due to the fact that there is a relatively higher number of points resulting from the spike metric approach, which had

[6] http://journals.plos.org/plosone/article?id=10.1371/journal.pone.0084217

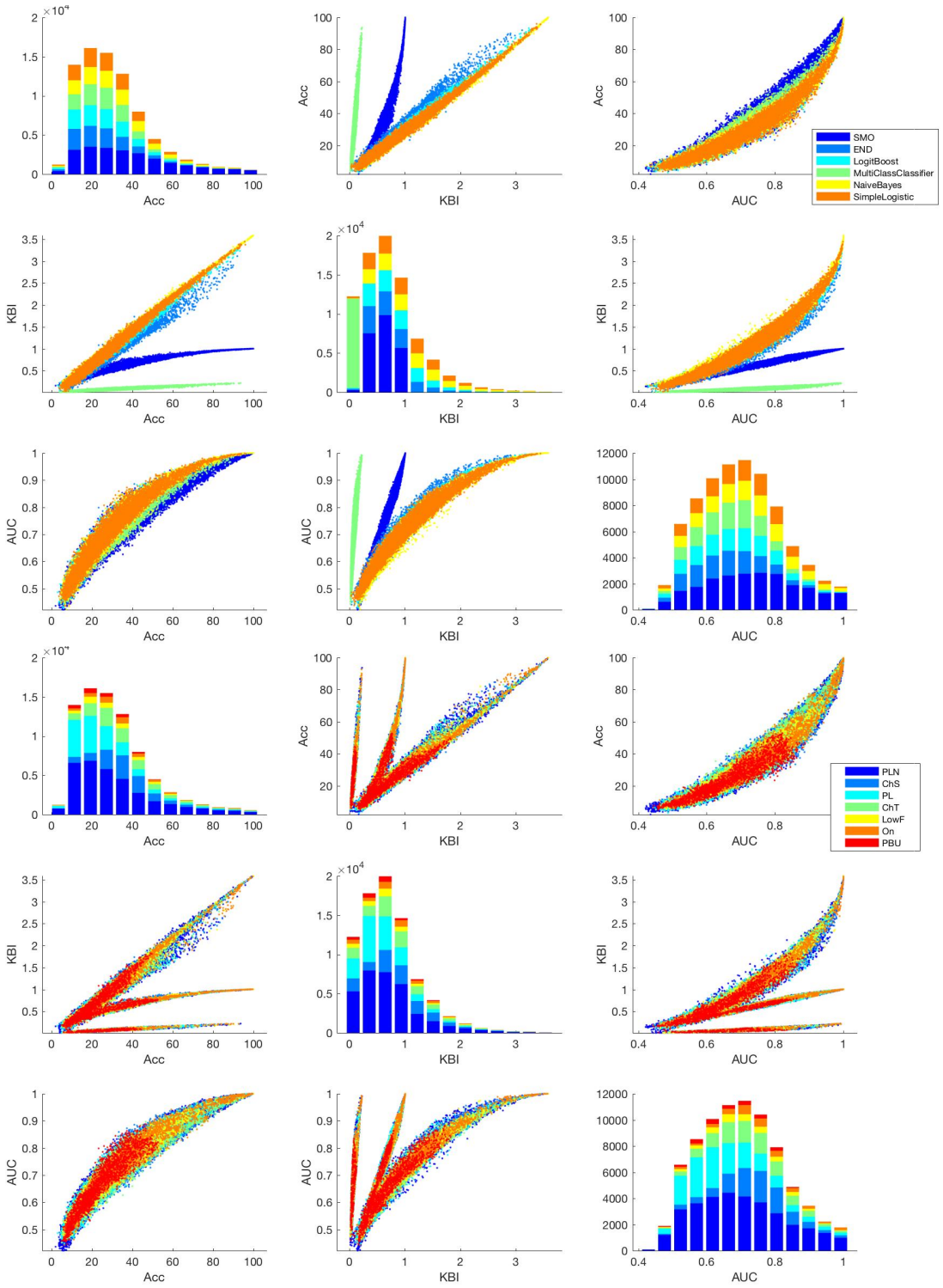


Fig. IV.3 Histograms and scatter plots of the 3 measures compared (κ, K&BI, AUC). On the diagonal, the histogram of the measures. Otherwise, axis are indicated on left and bottom. For visibility, these correspond to all classification using only the 6 classifiers in the legend (SMO to END). *Top 9*: Coloured by classifiers. *Bottom 9*: Coloured by unit type.

more parameters near the optimal parameter, as discussed for figure III.11.

The shapes appearing on the bottom-middle raster plot clearly discriminate the results coming from two classifiers: SMO and MultiClassClassifier. Those two are very close to a straight line, with MultiClassClassifier's slope being very steep. The K&BI results when using SMO are very close to zero, even for extremely good classifications (kappa near 1). This appears in the K&BI histogram since its first bin contains all values obtained using MultiClassClassifier, and relatively few of the other classifiers. Similarly, the following three bins contain most values obtained from SMO.

The bottom collection of plots in figure IV.3 presents the same data points, but coloured by type of neuron. No natural cluster emerges this time, the same classifier-based distributions are found across the different clusters defined previously as coming from specific algorithms. This fact is clearest on the plot showing κ versus K&BI: similar colouring is seen on data points coming from MultiClassClassifier, SMO, END, and also on the three other classifiers, as is verified in figure 11.3 of the appendix. As such, the clusters are unrelated to the neural origin of the data points but only to the classifiers, which was the initial motivation for comparing measures of performance on classification algorithms.

Are classifiers equivalent under different measures?

Different measures may evaluate classifications differently. This is of course not very important when the results are very similar, for example if a classifier is deemed equivalently good or equivalently bad under different measures, but whenever this is not the case the measures evaluate different aspects of the classification, which relates to the idea that they are based on different strategies.

Overall, a clear correlation between the averaged results for each classifier across measures can already be seen in figure III.9. After ordering the classifiers by increasing kappa score, the same trend appeared for the other two measures. A good classifier under one measure may be bad for another one, on a given dataset or possibly on all. The two most obvious examples are SMO and MultiClassClassifiers, as they both perform very well for both the kappa and the AUC measures, but are quite bad under the K&BI information score. This means that different classifiers perform differently for different measures, thus the 'best' classifier depends on the measure one cares most about. This behaviour was again observed in figure IV.3, where dots with the same colours correspond to evaluations from the same classifier, and the fact that SMO and MultiClassClassifier on this figure have their dots on lines with a steep slope is equivalent to these classifiers being good under one measure, and bad under the other one. This type of behaviour has been described in the literature, where trade-offs between classifiers are often described [Demar, 2006; Ali and Smith, 2006; Ng and Jordan, 2002; Boulesteix *et al.*, 2008; Sampson, 2012; Lehmann *et al.*, 2007], but no clear explanation seems to be given anywhere, and the clear alignment of points from a common classifier seems not to have been described.

Classifiers are built to optimise a specific function, uniquely defined by their algorithm in a way that is often hard to interpret; comparing measures of performance allows one to shed some light on this function.

IV.3 Discussion

The conclusions obtained in chapter III using the percentage of correct guesses would be mostly unchanged, had we used κ or AUC instead. Using K&BI, however, SMO and MultiClassClassifier would have provided poor results. In all cases, the relative results (between neuron types) would be unaltered.

To get a better understanding of the probabilities actually output by Weka, here is a very simple test: The training set corresponds to two instances, with value 0 and 1, respectively belonging to class 0 and 1. Then, we test feature vectors (numbers, here) between 0 and 1. Results are shown on figure IV.4. The conclusion of this hand-made test is that the MultiClassClassifier classifier outputs probabilities before thresholding, which is not the case for the other classifiers. Even IBk (k-nearest neighbours) would output only 1s, and IB1 (nearest neighbour) only 0.75s, even though they should be equivalent.

The understanding of this type of 'soft' probability could be made with the notion of risk: similarly to betting large or small amounts compared to the expected result, a measure would

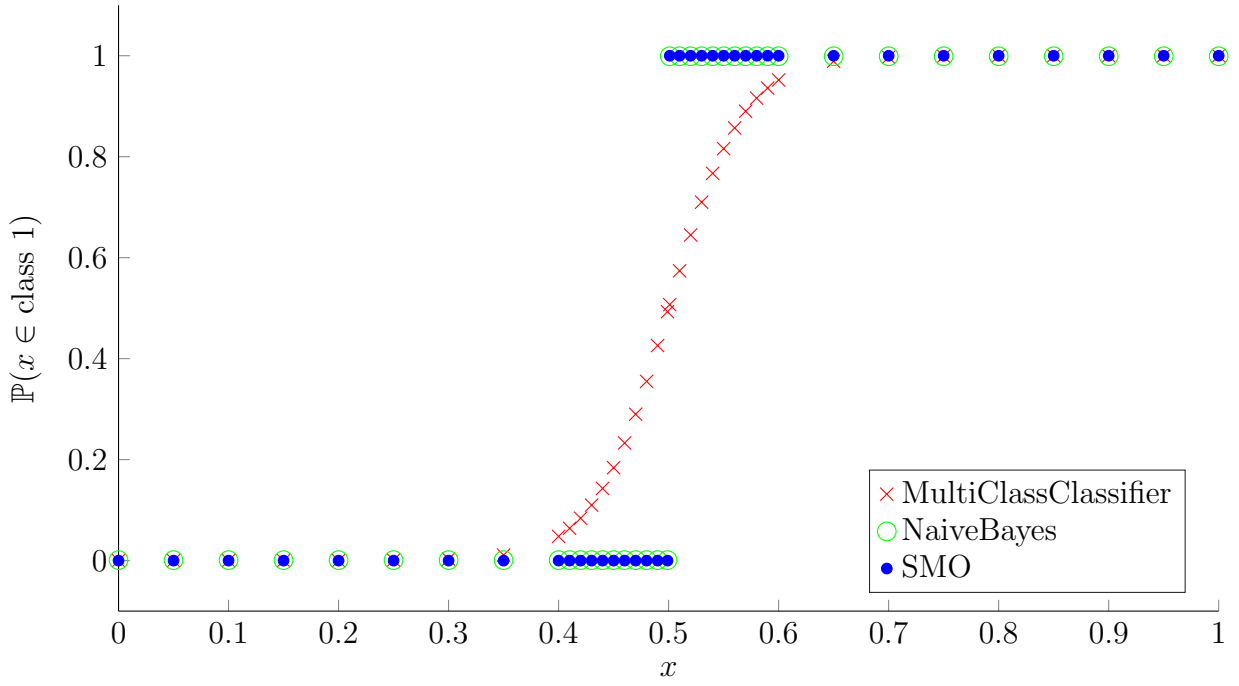


Fig. IV.4 Probabilities output by three classifiers that a value x belongs to class 1, tested on a subset of $[0, 1]$, when the training set was composed of two instances: the value 0 with class 0, and the value 1 with class 1.

gain more or less when correct, or lose more or less when incorrect. As seen in figure IV.4, MultiClassClassifier doesn't take much risk when it gives a probabilistic prediction, which is not the case for the others (NaiveBayes and SMO). As a measure for this risk assessment, K&BI information will add the scores, which are centred with respect to the prior distribution. Hence adding many small negative and positive values when the algorithm doesn't take too much risk (such as MultiClassClassifier) will lead to small overall gain (explaining the poor results with MultiClassClassifier under the K&BI measure), but summing bigger gains forces the K&BI information score to sum bigger values.

Because MultiClassClassifier uses an Error-Correcting Output Code method to aggregate the output of binary classifiers, it may be that for each base classifier, the prior probability of belonging to a class and the probability of not belonging to it are bigger than for the other classifiers (uniform priors in the previous case), and this imbalance between the two values would have a large negative impact on the K&BI information score, as observed. Since SMO is also an ensemble classifier, it may be subject to the same kind of imbalance on the prior, hence shifting the K&BI information score, compared to other non-ensemble classifiers.

The K&BI information is fairly different in nature because it is computed on the probabilities of belonging to the real class, whereas other measures (such as the accuracy) are computed only on the confusion matrix. As seen in figure IV.4, some algorithms output proper probabilities (continuous values between 0 and 1), but others output only 0 and 1. This leads to the interesting behaviour on the plot showing kappa versus K&BI for example.

Having proper probabilities is not enough to explain the peculiar behaviour of the green dots (MultiClassClassifier), since SMO[7] is not in this class but stands out too. This seems to relate to a more abstract property of these classifiers, which is that they use ensemble learning - see chapter II. The rigorous investigation of this issue is beyond the scope of this thesis, as it represents a problem in theoretical Machine Learning, studying the varieties that seem to implicitly define the results of trained algorithms, the different measures of performance then being different projections on $\mathbb{R}^+$. Just like the shadows of a complex 3D object are similar when light sources are near, the projections on $\mathbb{R}^+$ (the measures) give close and correlated results when they are calculated in a similar fashion; this phenomenon is not related to Neuroscience.

[7]Interestingly, SMO is categorised as 'functions' in Weka, which may be considered a mistake, and END is a meta-classifier, but has a much higher correlation between AUC and K&BI.

V.

Conclusion of part A

Envelope decoding starts early in the auditory pathway: the auditory nerve tonotopically encodes the input waveform, using a nonlinearity from the biophysics of inner hair cells that reveals slow fluctuations in each spectral band of the waveform. This information is sent to the Cochlear Nucleus, where many different types of neurons already output different pieces of information. Chopper units for example encode much more AM information than the primary-like units.

This conclusion was previously obtained using the Vector Strength, a metric that tests how much the spikes are in phase with the sound envelope. This measure was used to measure the outstanding capability of the auditory nerve fiber to lock to a specific phase of the envelope [Rhode and Greenberg, 1994], when responding to an amplitude-modulated tone. Using the same dataset, the authors' conclusions were verified using a data mining approach, free of any assumptions regarding the neuron's coding. This led to conclusions both in the realm of Neuroscience and in the field of the Machine Learning:

- Most conclusions from [Rhode and Greenberg, 1994] are supported by our results, regarding the difference in population-level coding of the modulation frequency. The one conclusion that is lost is the band-pass filtering ability of most neurones types at medium to high sound levels: we find using our data mining approach that transfer functions of all types of units and at all tested sound levels are low-pass, meaning that low modulation frequency is effectively contained in the spike trains, while the phase-locking capability is reduced at low frequencies and medium and high sound levels (most robustly in Chopper units);

- Classifiers, on average, can be well-ordered according to their results on various neuron types. This has useful consequences on engineering practices for neuroscientists, confirming that they should focus their efforts on tools that are either easy to use (time and skill constraints), or that bring some insight to the data (including meaningful biophysical quantities as features), or that simply beat other methods in terms of accuracy (as does Deep Learning given enough data and computational power);

- The common measures of performance of classification algorithms are strongly correlated, in a way that is classifier-dependant. Except for K&BI, the other measures (percentage correct, κ, AUC, MI) give very close relative results, with some variability that seem stochastic.

These results motivate us to use a single Machine Learning algorithm with a single measure of performance, now moving on to a much more dynamic and complex envelope: let's talk about speech!

B

Speech Recognition on Neural Data

VI.

Speech Recognition on Neural Data

In this review chapter, we present the field of speech recognition and the state-of-the-art of its application to neural data. The field of Biophysical Auditory Modelling VI.1 is first introduced, followed by an introduction to the field of Automatic Speech Recognition VI.2 with a presentation of Hidden Markov Models VI.3, used hereafter. The application of Automatic Speech Recognition (ASR) to neural data stems from two main motivations: to improve ASR systems by mimicking a system that already does it well - the mammalian brain - and as a tool to understanding the neural coding of complex sounds, since the nonlinearity of the auditory processing does not permit the extension of results from simple stimuli such as tones and clicks. In this latter regard, the chapter ends with a review in section VI.4 of the applications of speech recognition technologies to experimental and simulated electrophysiological neural data.

VI.1 Biophysical Auditory Models

VI.1.1 Introduction

The most common approach in Automatic Speech Recognition (ASR) is to compute features from sound waveforms, like in any other classification task [Rabiner and Juang, 1993]. Another approach is to model the physical responses to sound pressure using a biophysical model, mimicking the way the auditory system performs it. This approach initially had two main motivations: to develop better ASR systems, and to allow auditory neuroscientists to investigate the neural processing of sounds, relating their results to speech perception in a quiet or noisy background.

For a long time, it was considered plausible that the best speech analysis algorithms would be based on biophysical models of human audition [Lyon, 1982]. While this view pushed forward our drive to understand the auditory pathway, data driven discoveries enabled by the recent Deep Learning revolution have revealed that the strongest models may be built upon a sea of data, whenever enough data is available. Images and sounds galore are such examples, allowing incredible achievements in computer vision and speech application in just a few years that ought to soon match the human brain's capabilities[1] [Scharenborg, 2007; Barker *et al.*, 2013]. Unfortunately, a profusion of neural data is still not available due to technological and ethical limitations, forcing us to rely on auditory models and simulated data in our attempt to answer scientific questions about higher-order processing in the auditory system of the mammalian brain.

The models providing this data share many aspects, such as filtering or compression, as they all aim to reproduce the same biological system — up to the species-dependent specificities. They vary in their level of detail, in the tools used to implement auditory principles, the species to which the parameters are fitted, or whether and how they extract temporal information from the data in order to evaluate the model on a given task. For example, here is a description of the ways information is processed in three classical models:

- Seneff's auditory model [Seneff, 1988] consists of three stages: a bank of bandpass filters that model the frequency analysis of the cochlea, nonlinear rectification, short-term adaptation, lowpass filtering, and a rapid automatic gain control that models the transduction of the inner hair cells, and finally outputs two streams of information, one producing a rate coding in a fashion similar to the cochlea, and the other being a synchrony detector designed as a frequency-dependent periodicity detector that compares for each channel the mean rate of firing with a delayed version of it.

[1] Realistically, this goal could have been reached already. Industries' priorities often being on the applications that offer a financial return, speech applications are constantly improved and have now started to reach human parity [Xiong *et al.*, 2016] leaving only researchers with less data and computational power to study this matter, in order to understand why humans are, or were, better.

- Ghitza's EIH model (Ensemble Interval Histogram) [Ghitza, 1986] makes use of timing information to develop a spectral representation of the incoming sound. The ensemble histogram of the inverse of the interspike intervals from spike trains is calculated, this nerve fiber mechanism being modelled by a multi-level-crossing detector at the output of cochlear filters. This representation of speech is then used as front-end to a Dynamic Time Warping (DTW) recogniser.

- As in the two previous models, Lyon's model [Lyon, 1982] includes bandpass filtering, nonlinear rectification and compression, short-time adaptation, but adds a mechanism for lateral suppression: the presence of a second tone over a range of frequencies surrounding the Characteristic Frequency of an IHC may reduce its activity, compared to a single tone at its CF played alone.

To increase their interpretability, the models tend to include some processing that could be ascribed to the central auditory system, which is expected to extract useful features from the sounds. A review of the classical biophysically-inspired auditory representations used for ASR up to 2011 is [Stern, 2011]. Among these models, the most important for this thesis is the one used to simulate neural responses from the peripheral auditory system.

VI.1.2 Sumner's Model

Sumner's model [Sumner *et al.*, 2003] is a functional model of the auditory nerve response of the guinea-pig. While it shares common features with other models, it was fitted to experimental data, allowing one to test the capabilities of different types of nerve fibres in speech encoding, which is done in chapter VIII.

The middle-ear filtering is modelled by a cascade of Butterworth filters (VI.1.2.1). A dual-resonance nonlinear filter architecture is then used to reproduce the mechanical tuning of the basilar membrane (VI.1.2.2). Finally, transduction to the activity on the AN is accomplished with a model of the inner-hair-cell (VI.1.2.3). Details regarding the implementation of the model and the spike generation algorithm are given in appendix 12.

VI.1.2.1 Middle Ear

The response of the middle ear is modelled by a cascade of two linear band-pass Butterworth filters, in order to reproduce the thresholds found by [Evans, 1972]. One filter is second order with an upper cutoff of 25 kHz and a lower cutoff of 4 kHz. The other filter is second order with upper and lower cutoffs of 30 kHz and 700 Hz. In the original model, these filters were chosen to match data collected by Evans [Evans, 1992], and the second filter was third order. This was changed following [Steadman, 2015] to increase the response in the lower frequencies, known to be important for human speech [Plack, 2005].

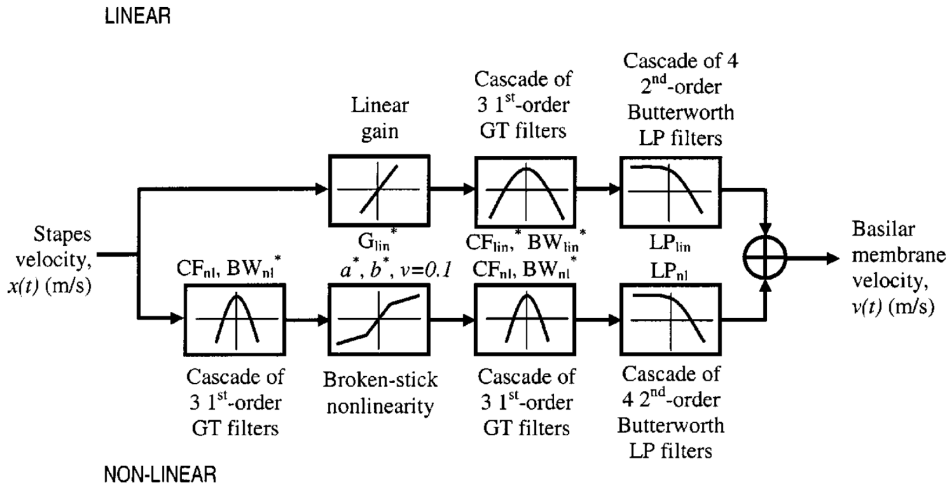


Fig. VI.1 Schematic of the DRNL filter architecture [Sumner *et al.*, 2003]. The filter output is a sum of a linear and a nonlinear pathway. The linear upper pathway is a gain followed by a gammatone filter and a low-pass filter. The nonlinear lower pathway consists of the following cascade; a gammatone filter, a compression function, a second gammatone filter, and a low-pass filter.

VI.1.2.2 Dual-Resonance Nonlinear Filter

The filtering of the BM is modelled with a dual-resonance-nonlinear (DRNL) filter architecture, shown in figure VI.1. This outputs the sum of two filters, one being linear and the second nonlinear. The broken-stick nonlinearity is described by

$$y[t] = \text{sign}(x[t]) \times \min(a|x[t]|, b|x[t]|^v) \tag{VI.1}$$

where a, b and $v = 0.1$ are parameters determining the exact behaviour. Due to the compression, the nonlinear pathway dominates the BF response only at low sound levels. At high levels, the linear pathway dominates the BF response, and at intermediate levels the output is a mix of the two.

It was shown in [Meddis *et al.*, 2001] that this model could be fit to BM laser-interferometry data for three different BFs (800 Hz, 9 kHz and 18 kHz) by varying the DRNL filter parameters.

VI.1.2.3 Model of the IHC

The first stage of IHC transduction is a simple biophysical model of the cilia transduction and receptor potential response. The second stage of transduction simulates the presynaptic calcium processes that lead to the release of neurotransmitter. Two calcium parameters at this stage determine the fiber type: G_{Ca}^{max}, the maximum calcium conductance in the vicinity of the synapse, and $[\text{Ca}]_{\text{thr}}$, the threshold concentration of calcium required for release.

The effects of these parameters on an ASR task are tested in chapter VIII. The third IHC stage models the manufacture, release, loss, and reuptake of neurotransmitter vesicles at the synapse. Finally, the refractory stage, explained in algorithm 2 of the appendix, then imposes an absolute and relative refractory period, reducing the probability that a vesicle will trigger an action potential.

VI.1.3 Front-Ends

As computational capabilities were increasing in the 90s, more features could be included in the models and two trends appeared in this scientific field: developing more complex biophysical models, or having simpler models that share the important features of the more complete ones. What is important to the modeller is that the model includes the relevant auditory phenomena. Such phenomena include the peripheral frequency selectivity, the saturating S-shaped rate-level response, the synchrony to low-frequency fine structure (replaced by the synchronisation to the envelopes at higher frequencies), an enhancement of sounds temporal contrast in the auditory-nerve fibres' transiency, or an enhancement of its spectral contrast by lateral suppression. To a certain extent, the presence of these principles can be assessed almost independently to each other.

Table VI.2 summarises the different processing blocks used in different front-end models, and important specificities of the model such as the species it was fitted with. In essence, all peripheral models share the use of a bank of narrow bandpass filters, a non-linearity and lowpass filtering; the low-pass filtering can be replaced by a threshold-passing paradigm [Gutig *et al.*, 2009]. Finer processing can include short-term and long-term adaptation by taking into consideration the release of neurotransmitters [Sumner *et al.*, 2003]. Subtle differences can be hard to categorise: for example, the amplification role of Outer Hair Cells is sometimes modelled explicitly [Zilany *et al.*, 2009], and sometimes implicitly contained within the nonlinearity of the basilar membrane filtering [Sumner *et al.*, 2003].

In order to fix the range of frequencies used in this thesis, table VI.2 contains the range of BFs used in all publications where they were defined, which is required when applying a model to an ASR task. The range of interest we kept was 100 Hz to 8 kHz. The number of different BFs varies greatly, from 34 in [Gutig *et al.*, 2009] to 512 in [Li *et al.*, 2000], while the number of nerve fibres per BF is typically varied within a publication. Results in [Holmberg *et al.*, 2007] where the total number of nerve fibres is varied from 15 to 10000 suggests that as few as 4% of the total number of ANFs would be sufficient to code speech information in a rate-place fashion.

Auditory nerve fibres have a wide range of spontaneous activity in all mammals. This spontaneous activity characterises the S-shaped rate-level function of the fibre, which in turn characterises the range of sound levels for which the fibre encodes information. A fibre's dynamical range is the sound intensity interval between which a variation in sound

	OME		BM	OH	IHC			Ada		AN						BFs		
Reference	T	A	B	Bl	R	G	L	SA	LA	L	C	LS	SG	SR	Sp	min	max	#Ch
[Lyon, 1982]			y		HW	y	y	y			y	y						u
[Ghitza, 1986]			y			y		y					y		?	200	3200	85
[Seneff, 1988]			y		HW	y		y		y	y				?	130	6400	40
[Yang *et al.*, 1992]			y		HW					y		y			?	200	5900	64
[Li *et al.*, 2000]	y		y			y								u		200	3500	512
[Zhang *et al.*, 2001]		y	y			y	y	y					y	H	Cat			u
[Sumner *et al.*, 2003]	y		y			y	y	y					y	LoMH	GP			u
[Holmberg *et al.*, 2007]	y		y			y	y	y					y	LoH	Hu	20	8000	91
[Gutig *et al.*, 2009]			y		Log	y					?		y			130	5400	34
[Zilany *et al.*, 2009]	y		y	y	y	y	y	y	y				y	?	Cat			u
[Brown *et al.*, 2010]	y	y	y			y	y	y						Lo	Cat	100	4500	30

Tab. VI.2 Summary of auditory front-ends. Common functional roles may be implemented differently in different models. Abbreviations: OME Outer/Middle Ear, BM Basilar Membrane, OH Outer Hair Cell, IHC Inner Hair Cell, Ada Adaptation, AN Auditory Nerve, BFs Best frequencies, Ref Reference, T Transfer Function, A Attenuation, B Bandpass Filterbank, Bl Block, R Rectification, G Gain, L Lowpass Filtering, SA/LA Short-Term/Long-Term Adaptation, C Automatic Gain Control, LS Lateral Suppression, SG Spike Generation, SR Spontaneous Rate, Sp Species, min/max Minimal/Maximal BF used, #Ch Number of Simulated Channels (uniformly distributed in log scale), y Yes, HW Half-Wave, u Undefined, H High, M Medium, Lo Low, GP Guinea-Pig, Hu Human. An empty box means the property is not part of a model; a question mark when the characteristics could not be clearly found

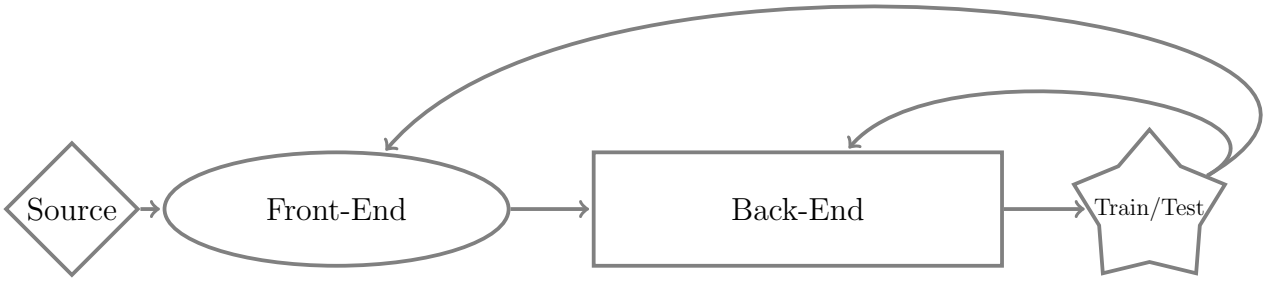


Fig. VI.3 Processing chain of ASR systems: a source is recorded and sequentially processed by two functional blocks, respectively called the front-end and modelled in the back-end, before being trained and tested. The front-end block directly processes the source data, typically a speech waveform. The back-end block may include additional processing. Training is achieved by propagating the errors to change the processing parameters (front-end or back-end) and the models being built (back-end).

level induces a variation in average spiking activity. Fibres are classified as Low Spontaneous Rate (LSR) when in the absence of stimulus their firing frequency is lower than 18 spikes per second and display a Rate-Intensity function with a shallow slope at Best Frequency. Medium Spontaneous Rate units (MSR) display a 'sloping saturation' type rate of growth at BF, with a low spontaneous activity (lower than 18 spikes per second). High Spontaneous Rate (HSR) units fire at a frequency higher than 18 spikes per second in the absence of any stimulus and have a narrow dynamic range [Sumner *et al.*, 2003; Winter and Palmer, 1991].

This ends the review of the auditory front-end blocks. The next section focuses on ASR systems and their back-end block, before reviewing their application to neural data, the front-ends being either the models presented in table VI.2 or experimental data.

VI.2 Automatic Speech Recognition

ASR is the automatised process of transcribing speech into text. Speech recognition is rarely accomplished by attempting to classify words from the raw waveform. The waveform is converted into an alternative representation which better represents the features of interest. Thus the first stage of processing, the Front-end block from figure VI.3, can be thought of as feature extraction.

ASR is a subfield of computational linguistics that is diverging into a subfield of Machine Learning, as the expertise used in the field is being replaced by parameters learned from the data itself. As in many scientific fields, Deep Learning has had a big impact on ASR systems. Typically, the front-end of a modern ASR system would be a neural network, rather than any fixed features (such as MFCCs, defined below in subsection VI.2.2), while the back-end

would still be a Hidden Markov Model (defined in subsection VI.2.3). By training with lots of data, big technology companies (Apple, Google, Microsoft, ...) now offer real-time speech recognition-based services such as Siri.

This data-driven learning is extremely powerful as major technology industries were realising back in 2012 [Hinton *et al.*, 2012]. However, when the amount of data is much more limited in quantity, as in the case of a physiological experiment, using hand-crafted features that extract useful information is still a valuable approach in order to understand the nature of the information being processed. This tradeoff motivates the choice to test different features and processing chains in the following chapters.

To give, later on, a fair comparison between speech-derived features and features calculated from neural data, classical auditory features will be used, developed by researchers over decades [Rabiner and Juang, 1993; Scharenborg, 2007]. The most important of these is introduced below, as well as the motivation behind their construction. First, a subsection defines the Hann and Hamming windows.

VI.2.1 Windows

In signal processing, a window function, abbreviated as window, are real-valued functions that are zero-valued outside of some interval. Pointwise multiplication of another function f by a window w provides a local view of f on the interval where w is non-zero. The Hann and Hamming windows are commonly used to smooth down time series, by convolution. Their definitions are very close: for a sample of length N, the coefficients for n in $\{0, \dots, N-1\}$ are defined as

$$w_{\text{Hann}}(n) = 0.5 - 0.5\cos\left(\frac{2\pi n}{N-1}\right) \qquad \text{and} \qquad w_{\text{Hamming}}(n) = 0.54 - 0.46\cos\left(\frac{2\pi n}{N-1}\right).$$

VI.2.2 Classical Features for ASR

For many years, auditory features were engineered by researchers, using their expertise in ASR and their knowledge in phonetics or psychoacoustics. The most common feature for speech recognition, the Mel Frequency Cepstral Coefficients (MFCCs) was introduced in 1980s [Davis and Mermelstein, 1980]. MFCCs have then been, for years, the industry standard speech feature used in the ASR community, and contain important processing steps also present in other classical feature extraction methods. The different steps leading to extracting MFCC features are presented on the left column of figure VI.4 and briefly explained here:

1. *Pre-Emphasis*: A first order high-pass filter is applied to the speech signal to improve the signal to noise ratio;

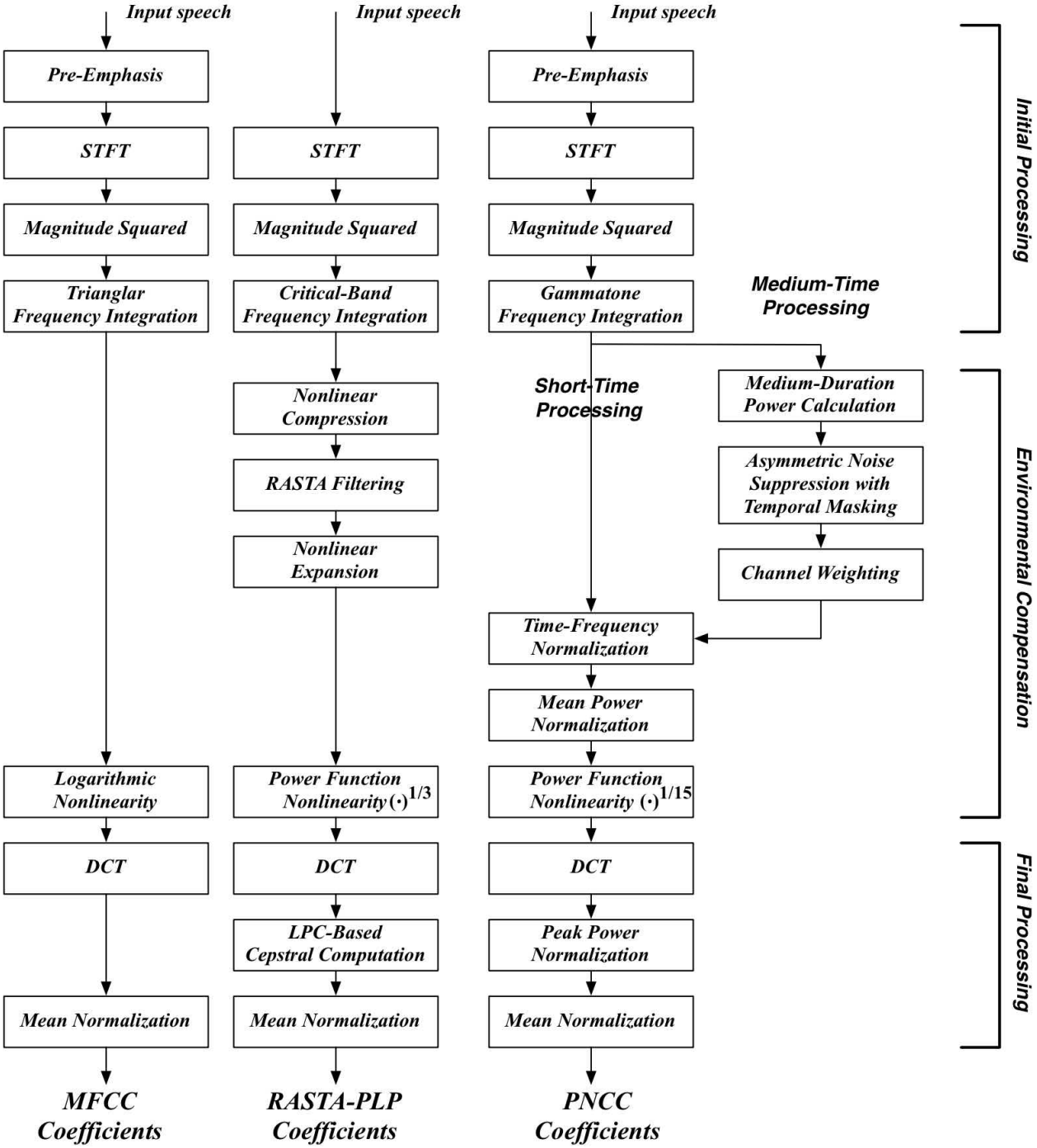


Fig. VI.4 Comparison of major functional blocks of the three classical features used in speech-related tasks: MFCC, RASTA-PLP and PNCC. Figure from [Stern, 2011].

2. *Short-Time (ST–)*: Speech timescales vary enormously depending on the task at hand, and the ASR community found that windows of 20 to 30 milliseconds contain the local speech information, and this varies at a rate of around 100 Hz. Thus, the signal is segmented into frames of about 25 ms, sliding the window by 10 ms steps (hence a 15 ms overlap between two consecutive frames). Each frame is then multiplied by a Hamming window to reduce the impact on the spectrum of abrupt changes in the signal, called spectral splatter;

3. *Fourier Transform (–FT)*: The spectral information of each frame is obtained by a Fourier transform;

4. *Magnitude Squared*: The magnitude is squared since the energy at different frequencies is related to the squared values of the Fourier coefficients;

5. *Triangular Frequency Integration*: Frequency-domain filtering is performed by applying a bandpass filter bank, their positions being equally spaced along the Mel-frequency, a logarithmic scale copying the best frequency placements of IHC in humans, reflecting similar effects in the human's aural perception and downsampling in frequency;

6. *Logarithmic Nonlinearity*: To differentiate between weak energy values, a nonlinearity is applied, mimicking Fechner's psychophysical transfer function for intensity [Stern, 2011];

7. *Discrete Cosine Transform (DCT)*: DCT is a Fourier-related transform that takes the features back into the time domain, thus providing a lowpass Fourier series representation of the frequency-warped log spectrum. This transform is widely used in image and audio processing as it tends to keep the most relevant features of speech;

8. *Cepstral Mean Normalisation*: Each channel is renormalised over time, to remove channel effects such as constant room noise[2].

Various arrangements of MFCCs have been subsequently made, mainly differing from the initial one in the approximation of the nonlinear pitch perception in humans, the filter bank design and the compression of the filter bank output. As examples, the processing chains of RASTA-Perceptual Linear Predictive and the Power-Normalized Cepstral Coefficients [Stern, 2011] are presented on figure VI.4. Other processing steps can be added to improve

[2]Assuming the recorded signal is a linear convolution of an input signal $x[n]$ with a channel impulse response $h[n]$, the process of applying a logarithm to our Fourier representation, separates them into a sum:

$$y[n] = x[n] * h[n] \quad \text{and} \quad \log(Y[n]) = \log(X[n]) + \log(H[n]).$$

Assuming a stationary impulse response, H is constant and by subtracting the average signal one removes H:

$$\log(Y[q]) - \text{mean}(\log(Y)) = \log(X[q]) - \text{mean}(\log(X[q])).$$

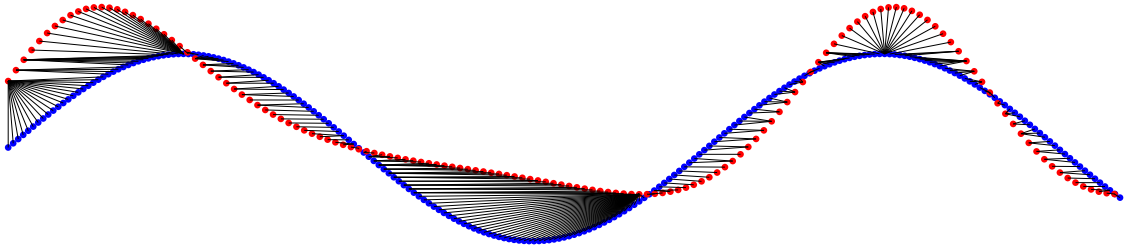


Fig. VI.5 Optimal mapping between two distorted sinusoids having different sampling frequencies, using Dynamic Time Warping, to show how DTW compensates for rate variations by mapping many points from the faster (blue) curve to the same point slower (red) curve.

the recognition using different features. For example, on the right column of figure VI.4, the 'Medium-Time Processing' steps were added to reduce the effects of reverberation.

VI.2.3 Models for Automatic Speech Recognition

After calculating features, a statistical model is still needed for the phonemes: the Back-end block of figure VI.3. At a short time-scale (around 10 ms), speech can be approximated as a stationary stochastic process: the sound generated in this time frame is a stochastic instantiation of a given structure, such as the vowel `/a/`. As such, speech can be approximated and modelled as a piecewise stationary process, making Hidden Markov Models (HMM, defined in section VI.3) a strong candidate as a back-end modelling tool for speech. id-01-07

Before HMMs become a standard, Dynamic Time Warping (DTW) was the best modelling tool for ASR systems. Statistical features were compared to each other and dynamically mapped so that differences in phoneme duration would be neglected. Based on the comparison of speech sequences (time series), one would be 'warped' onto the other in an optimal way, thus compensating for variations in speech rate between the two signals. DTW optimally maps one signal to another signal, efficiently removing temporal differences such as how long a vowel is spoken. Such a mapping is illustrated in figure VI.5, where the two signals being mapped are distorted sinusoids having different sampling frequencies; the optimal mapping is shown sample by sample using black lines.

VI.3 Hidden Markov Models

This section introduces the HMM technology, used in the next chapters, covering the mathematics of the Gaussian mixtures, the topology of the HMMs, the sub-word systems, a definition of the accuracy for a continuous recognition task, the training protocol used in the following chapters, and the way silences are dealt with using two models with different topologies.

VI.3.1 Introduction to HMMs

In probabilistic terms, a HMM is a dynamic Bayesian network: being given a series of observations and the symbols they encode, the goal is to train the network in order to recognise new (unobserved during training) instances of similar observations. The model built is a Markov process with unobserved (hidden) states that should be inferred.

First introduced and studied in the 60s and early 70s, the golden age of HMMs was in the 80s, where HMMs were successfully applied in many domains such as speech, handwriting, gesture recognition and bioinformatics [Rabiner and Juang, 1993]. Their flexibility in temporal pattern recognition made them the perfect candidate in all areas where researchers could extract useful features.

The first main difficulty in speech recognition is that the mapping from symbols (words) to sounds is not one-to-one: not only is there a huge variability in the various ways a word or phoneme can be pronounced, but also because similar sounds can encode different symbols. The second difficulty is that of the unknown word-boundaries, which differentiates this problem from a classification task.

HTK is a widespread C++ software implementing HMMs. It specialises in speech recognition, for which many additional tools were developed[3]. Following the HTK book [Young *et al.*, 2006], the mathematics behind a HMM in the case of isolated words (VI.3.2) are now described. The complementary algorithm that allows for decoding of series of symbols will be described next (VI.3.4). The rest of this chapter will be the application of HMMs to neural data as an ASR back-end modelling tool. Although the use of HMMs is more general, the focus here is on spoken words, to make the description less abstract[4].

VI.3.2 Isolated Word Recognition with HMMs

Let's assume that the sound waveform was encoded into a sequence of speech vectors or observations $\boldsymbol{O}$, typically into a sequence of vectors containing smoothed-spectral information by windows of 10 milli-seconds, defined as $\boldsymbol{O} = \{\boldsymbol{o}_1, \boldsymbol{o}_2, \dots, \boldsymbol{o}_T\}$ where $\boldsymbol{o}_t$ is the speech vector observed at time t. The isolated word recognition problem can then be expressed as finding

$$i_{opt} = \underset{i}{\operatorname{argmax}}\{\mathbb{P}(w_i|\boldsymbol{O})\}, \tag{VI.2}$$

where w_i is the i-th vocabulary word and $\mathbb{P}$ a probability defined below. Using Bayes theorem,

$$\mathbb{P}(w_i|\boldsymbol{O}) = \frac{\mathbb{P}(\boldsymbol{O}|w_i)\,\mathbb{P}(w_i)}{\mathbb{P}(\boldsymbol{O})}, \tag{VI.3}$$

[3] HTK is not as easy to use as other softwares offering similar modelling capabilities [Gaida *et al.*, 2014].

[4] Despite its relatively simple mathematical framework, HTK relies on various refinements to perform well, including feature projection, improved covariance modelling, discriminative parameter estimation, adaptation, normalisation, noise compensation and multi-pass system combination [Gales and Young, 2007].

and given the prior probabilities $\mathbb{P}(w_i)$, the most probable word only depends on the likelihood $\mathbb{P}(\boldsymbol{O}|w_i)$. Given the possibly huge dimensionality of the observation sequence $\boldsymbol{O}$, the direct estimation of the joint conditional probability $\mathbb{P}(\boldsymbol{o}_1, \ldots | w_i)$ is practically intractable. We are thus lead to make the simplifying assumption that the underlying model (that produces the sounds) is a Markov model.

The problem is now much simpler as it reduces to estimating the Markov model parameters. In a speech recognition task, the Markov model is a finite state machine which changes state at every time unit, and for each time t that a state j is entered, an observation $\boldsymbol{o}_t$ is sampled from the probability distribution $b_j(\boldsymbol{o}_t)$. The transition from state i to j is probabilistic and governed by the discrete probability $a_{ij} > 0$, with the property that $\sum_j a_{ij} = 1$.

Since each time t is associated to an unknown state x_t, the joint probability that the sequence of observations $\boldsymbol{O}$ be generated by the model M (which is an HMM, thus defined by the a's and b's parameters) moving through the state sequence X can be expressed as the product of the transition and output probabilities

$$\mathbb{P}(\boldsymbol{O}, X|M) = a_{x_1x_2} b_{x_2}(\boldsymbol{o}_1) a_{x_2x_3} b_{x_3}(\boldsymbol{o}_2) \ldots a_{x_t x_{t+1}} b_{x_{t+1}}(\boldsymbol{o}_t) \ldots . \tag{VI.4}$$

In practice, only the observation sequence $\boldsymbol{O}$ is known, the underlying state sequence X is hidden, hence the name 'Hidden Markov Model'. We can compute the joint likelihood by summing over all possible state sequences $X = x_1, \ldots, x_T$

$$\mathbb{P}(\boldsymbol{O}|M) = \sum_X a_{x_0x_1} \prod_{t=1}^{T} b_{x_t}(\boldsymbol{o}_t) a_{x_t x_{t+1}}, \tag{VI.5}$$

where x_0 is constrained to be the model entry state, and x_{T+1} the model exit state. Both are non-emitting in HTK (they are not associated with actual observations or models, see below), but allow the construction of composite models that concatenate predefined HMMs. Although direct calculation of equation VI.5 is not tractable, a simple recursive procedure to evaluate it efficiently is implemented in HTK.

A final assumption is needed to relate the models and their associated symbols: given a set of models M_i corresponding to words w_i, let's assume

$$\mathbb{P}(\boldsymbol{O}|w_i) = \mathbb{P}(\boldsymbol{O}|M_i), \tag{VI.6}$$

assuming that the parameters a_{ij}'s and b_i's are known. The training phase is now a matter of parameter estimation (the a_{ij} and b_j corresponding to each word model), and the recognition phase reduces to computing $\mathbb{P}(\boldsymbol{O}|M_i)$ for all i's and selecting the model with maximal probability.

As most continuous density HMM systems, HTK uses Gaussian Mixture Densities to

parameterise the densities b_j. For the rest of this thesis, only one data stream is considered, in order to simplify the models. The formula for expressing $b_j(\boldsymbol{o}_t)$ is then

$$b_j(\boldsymbol{o}_t) = \sum_{m=1}^{M_x} c_{jm}\,\mathcal{N}(\boldsymbol{o}_t; \boldsymbol{\mu}_{jm}, \boldsymbol{\Sigma}_{jm}), \tag{VI.7}$$

where c_{jm} is the weight of the m-th mixture component and $\mathcal{N}(\cdot; \boldsymbol{\mu}, \boldsymbol{\Sigma})$ is a multivariate Gaussian density with mean vector $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$, that is

$$\mathcal{N}(\boldsymbol{o}; \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{\sqrt{(2\pi)^n |\boldsymbol{\Sigma}|}} e^{-\frac{1}{2}(\boldsymbol{o}-\boldsymbol{\mu})' \boldsymbol{\Sigma}^{-1} (\boldsymbol{o}-\boldsymbol{\mu})}, \tag{VI.8}$$

with n being the dimensionality of $\boldsymbol{o}$. The above explanations of the algorithms are only rough; the interested reader is referred to the HTK book [Young *et al.*, 2006] for all details.

HTK will initialise $\boldsymbol{\mu}_j$ and $\boldsymbol{\Sigma}_j$ for all states j (using HTK's internal function HInit), then finds the maximum likelihood state sequence (by the Viterbi algorithm) and reassigns weighted means and variances to all states (Baum-Welch re-estimation) using weights calculated efficiently by a Forward-Backward algorithm. This process is repeated until convergence of the means, variances and transition parameters. At this point, the model can be tested on an unseen dataset.

VI.3.3 States & Mixtures

Non-emitting states are states defined in a HMM that are not trained or used for recognition. For engineering purposes, HTK's models include non-emitting entry and exit states, used to glue models together in continuous speech tasks, so that a HMM with 5 states actually only contains 3 sets of parameters (called emitting states) that describe the data it relates to, while the non-emitting first and the last states are only used by HTK to pass information when finding the optimal sequence of HMMs.

A HMM is defined by a given topology, that is, a fixed number of states, each having a fixed number of Gaussian mixtures that are iteratively trained on a dataset, and a transition matrix with a chosen set of non-zeros entries. Mixtures are alternative representations of the same state, so that the number of mixtures defines the number of ways a given state can be modelled. The different states reflect the stages that describe what the HMM is encoding; for example, if a 5 state-HMM was trained to recognise the sound /ABA/, the states 2 and 4 would probably (if correctly trained) encode the sound /A/ and state 3 would mostly contain a parametric description of the sound /B/. If a HMM were built to recognise the longest French word 'anticonstitutionnellement' (/ɑ̃.ti.kɔ̃s.ti.ty.sjɔ.nɛl.mɑ̃/), it would certainly need around 20 states, each describing a different part of the word that the HMM has to read, from left to right. Except in special cases, for example to recognise laughter 'ahahaha', it is

natural to consider that speech models should be read entirely, from left to right, which we describe as left-to-right without jump. To recognise a laughter, however, the amount of times the sound 'ah' is repeated varies, and this can be defined by returning to a previous states: the HMM recognising 'aha' can be changed to recognise 'aha', 'ahaha', 'ahahaha' and the like by changing the topology of the HMM and authorising the states to be read back; see the right graph of figure VI.8 for a visual explanation. All HMMs used are left-to-right without jump (topology shown in figure VI.7), exception made for silence models (figure VI.8).

Note that the use of a 3-state HMM in HTK practically requires only 1 emitting state for training and testing, because the initial and final states are non-emitting. Using 3 states is then close to using a nearest-neighbour classifier, because the model is no longer defined by a sequence of states, each mixture being a different template of the sound being modelled.

VI.3.4 Extension to Continuous Speech Recognition

A major difficulty in ASR systems is to handle the dynamical aspect of speech: the algorithm doesn't know when a word starts or ends. HTK uses the internal function HERest to perform embedded training, where all models are updated simultaneously after a composite HMM is trained on the whole utterance. Another tool (HVite) is used to find the optimal path (or optionally multiple alternatives), together with making the word boundary decisions and keeping a record of the words, using the Viterbi algorithm and the Token Passing algorithm [Young *et al.*, 2006].

With the option `-a`, HVite can use an utterance's word-level transcript and a dictionary to find the optimal phone-level transcript; this procedure is called forced alignment and depends on the training the HMMs received at that point. For each training pass, a first pass step of optimally identifying the phoneme transcripts given the current models is executed by forced alignment with HVite: while this is probably of little use for a small dictionary, it may be useful for a large-dictionary task by refining the labels and timing at each pass. The HMM parameters are then updated using the isolated-word algorithm previously presented.

VI.3.5 Networks, Word & Sub-Word Systems

HTK provides the recognition tool called HVite which uses the Token Passing algorithm to perform Viterbi-based speech recognition. HVite takes as input a network describing the allowable word sequences, a dictionary defining how each word is pronounced and a set of HMMs.

In chapters VIII and IX, a so-called word loop is used as grammar to allow any combination of words to be recognised. This simple network is sometimes called a trivial grammar. These 'words' are either proper words found in the dictionaries used, or the list of phonemes modelled with HMMs (leaving aside the silence models). These two types of recognition are

Word-Level Dictionary		Phone-Level Dictionary	
		aa	aa
		ah	ah
		b	b
		er	er
ONE	w ah n	n	n
AUBURN	aa b er n	w	w

Tab. VI.6 Small dictionaries at the word (left) and phone (right) levels.

called word-level and phone-level recognition. For phone-level recognition, the very same methodology applies, except that the dictionary used is trivial: each phoneme (playing the rôle of a dictionary entry) is decomposed as itself, where real words are decomposed into the sequence of phones they are made of, as shown in table VI.6.

After compilation from a word-level network, a dictionary and a set of HMMs, a word-level recognition network in HTK ultimately consists of HMM states connected by transitions to form paths having different log probabilities obtained by adding the log probabilities of the different emitting states generating the corresponding observations and the different transitions. These transitions are either within-HMM state transitions (defined by the transition matrix) or word transitions using language model likelihoods (defined by the network). By equiprobability of the words, the latter plays no rôle for trivial grammars.

The paths are efficiently found and compared using a Token Passing algorithm, whose description is beyond the scope of this thesis. The path with maximal likelihood is then selected by the algorithm as the recognised sequence of words.

VI.3.6 Accuracy For Continuous Speech Recognition

All results in the following chapters use the Accuracy metric of HTK. It should be noted that 'Accuracy' here has a different meaning to that in the chapter on Machine Learning. Continuous speech recognition is different because of its dynamical aspect: since the algorithm, when recognising, ignores the number of words it should find, there are 3 types of errors:

- Deletion errors: a word or more is missing in the recognition;
- Substitution errors: a word is mistakenly recognised as another one, as in classification;
- Insertion errors: a word is added where there isn't one.

HTK evaluates the number of errors - using HTK's internal tool HResults - by comparing a set of label files (output from a recognition tool, HVite in our case) with the corresponding reference transcription files. The evaluation algorithm is a Dynamic Programming-based string alignment procedure, using the standard US National Institute of Standards and Technology (NIST) Figure Of Merit (FOM) metric for the analysis of word-spotting output.

The optimal string match is calculated by minimising a score between the two strings with the following rules: identical labels match with score 0, a label insertion or deletion carries a score of 7, and a substitution carries a score of 10[5].

The accuracy figure in HTK takes into account the insertion errors (which can be understood as false positives), while the 'percentage correct' does not. Typically, a result with high percentage correct and very low accuracy would mean the algorithm recognised far too many words, including the correct ones. Hence, after the first plot, only the accuracy is considered as measure of performance. Formally, by defining H the number of correctly recognised labels (Hits), N the total number of labels in the defining transcription files, I the number of insertion errors, those two measures are defined by

$$\%\text{Correct} = \frac{H}{N} \times 100\% \qquad \text{and} \qquad \text{Accuracy} = \frac{H - I}{N} \times 100\%. \tag{VI.9}$$

The number of substitutions S denotes the remaining labels, incorrectly associated to a label.

VI.3.7 Training Protocol

In all chapters using HTK, phoneme-level HMMs are built. A so-called 'flat start' is used for all phonemes using HTK's internal function HCompV, thus giving all Gaussians a common mean and variance. The HMM's topology is 5 states left-to-right without jump, as represented in the figure VI.7. HMMs initially contain only one Gaussian mixture. Eight passes of embedded training are performed with HERest, without silences in the transcripts (except for an initial and a final one with SENT-START and SENT-END).

Our short-pause HMM is defined as a 3 states left-to-right HMM with a non-zero probability of jumping from state 1 to state 3 as shown in figure VI.8[6], its middle state being tied to state 2 of the long silence HMM named `sil`. At this point of the training, short-pauses are added to the end of each word to take care of any pauses introduced by the speaker. Then, 5 passes of retraining are performed with HERest. The number of mixtures for each HMM's

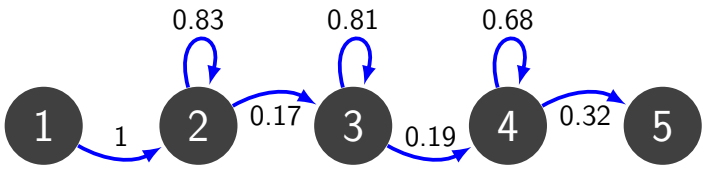


Fig. VI.7 Graphical model representing a left-to-right jump-free Markov chain with 5 states. Numbers are the non-zero probabilities of jumping between states.

[5]The substitution operation has to cost less than an insertion plus a deletion, otherwise the only thing the score would measure is the difference in length of the two sentences, as no substitution would be scored.

[6]This type of model is called tee-model. It is adapted for short pauses, as it can be skipped in the training phase if no feature vector corresponds to the HMM model trained so far — in this case, a short silence.

emitting state is finally sequentially increased to 5, retraining with 6 passes of HERest every time a mixture is added.

The optimal `-p` word insertion penalty, discussed next in VI.3.8, is chosen among two sets of values tested on the development set: from -200 to 0 by increments of 20 for phone-level recognition, -120 to 0 by increments of 40 for word-level recognition.

VI.3.8 Dealing with Silences

An unexpected source of degradation of accuracy using HTK models can be the amount of silence between words in speech files. HMMs deal with the dynamical properties of speech essentially using two sets of parameters:

Transition Matrix: The transition matrix $A = (a_{ij})$ that defines the probability of being in a state i, to jump into state j in the next frame as $a_{i,j}$. The topology of the phonemes' HMM is normally left-to-right, meaning that states can only increase, generally without skipping any states. For example for the phoneme `u`, the transition matrix could look like this after training:

$$A(\mathrm{u}) = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0.83 & 0.17 & 0 & 0 \\ 0 & 0 & 0.81 & 0.19 & 0 \\ 0 & 0 & 0 & 0.68 & 0.32 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

which is given as a graphical model in figure VI.7. Silence models, on the other hand, are treated differently. Two types of silences are trained: an HMM named `sil` for long silences and an HMM called `sp` for shorter ones, with different topologies (i.e. different kinds of

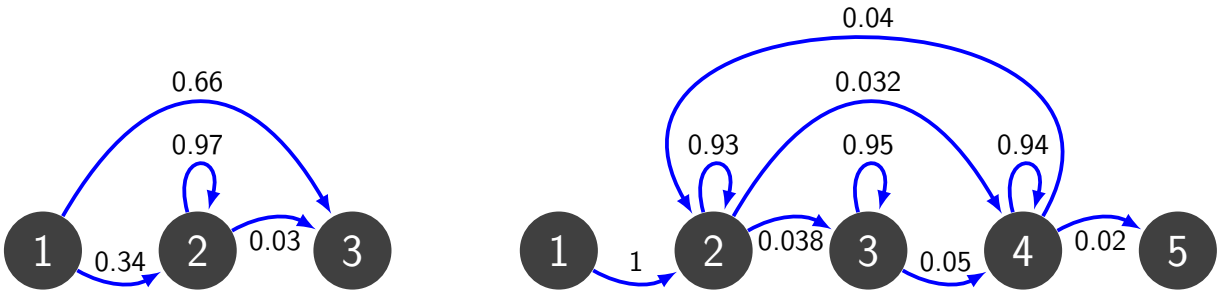


Fig. VI.8 Graphical models representing short ('sp', left) and long ('sil', right) silence HMM. Numbers are the non-zero probabilities of jumping between states.

transition matrices). For example:

$$A(\text{sp}) = \begin{pmatrix} 0 & 0.34 & 0.66 \\ 0 & 0.97 & 0.03 \\ 0 & 0 & 0 \end{pmatrix} \quad \text{and} \quad A(\text{sil}) = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0.93 & 0.038 & 0.032 & 0 \\ 0 & 0 & 0.95 & 0.05 & 0 \\ 0 & 0.04 & 0 & 0.94 & 0.02 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix},$$

where `sp` can jump from state 1 to state 3, both being non-emitting, and `sil` can jump from state 4 to state 2, which reads as the possibility to go through states 2, 3, 4 one more time. Both are represented as graphical models in figure VI.8. These transition parameters are trained like the other HMM parameters, including the HMMs sharing tied states. This implies that recognising a long silence and a short one do not lead to the same likelihood, which is what the algorithm calculates to recognise labels.

For sentences bearing unusually long silences between words (as might happen in chapter VIII), this effect distorted the recognition in a surprising way: many words are omitted in the testing phase. HTK will not recognise many words within those files, leading to a statistically high number of deletion errors in files having long silences. By checking the likelihood values in those files, it was possible to confirm that the amount of silence was the only source of problem.

Word Insertion Penalty

A word insertion penalty parameter can be fitted when using the function HVite for recognition with the option `-p`. This parameter is typically fitted on a development set that is separate from the training set. This parameter, however, is unique and gives a weight to fit the number of words to recognise. Fitting this parameter to the files containing long silences would solve the deletion rate bias for them but would increase dramatically the number of insertion errors in the files containing less silence between words. For this reason, development sets with normal silence durations were used, using the unusual deletion errors rate in the others as an additional measure of the quality of the HMMs: not only do they need to be well trained, they would also need to be recognised with very high likelihood to balance the overwhelming weight of `sil` in the files containing long silences.

VI.4 Speech Recognition on Neural Data

This section reviews the application of speech recognition technologies to neural data and explains the choice of some parameters fixed in the next chapters. The type of neural data found in the literature is first presented, with a focus on spike train data and the number of simulated units we ought to use. A simple model is then presented, as a first effort to

evaluate the number of animals to use for such a task, reflecting the various constraints of an electrophysiological experiment. This section ends with a discussion on the complexity of the speech datasets used in our literature review.

VI.4.1 Data Type

Since experimental evidence is at the heart of scientific progress but obviously requires the appropriate data to test hypotheses on, technology is a limiting factor for advances in Neuroscience. For example, relevant data for hypotheses-driven Auditory Neuroscience requires both high spatial and high temporal resolutions to deal with the fast processing of sounds our brain experiences.

Models are meant to reproduce specific features of the phenomenon under study, which is normally estimated by the predictive power of models. This predictive power always has limitations: as long as a phenomenon is not perfectly understood, the model won't be able to give good predictions when its input is modified. In the case of auditory modelling, since central auditory processing is imperfectly understood [Elhilali *et al.*, 2004], there is still a need for experimental data before auditory models can reproduce it. We thus quickly review the ASR capabilities of different brain technologies.

With current technology, electrophysiologic activity is best suited for ASR applications thanks to its speed. On the other hand, modalities based on metabolic processes such as functional Magnetic Resonance Imaging (fMRI) are too slow to capture the fast changes associated with speech processing [Herff and Schultz, 2016]. Electrodes can either measure ensemble of neurons and their synchrony (EEG, MEG, ECoG, defined below) or the localised activity of one or a few neurons (microarrays) up to our spike sorting capability. Each of these techniques offers a trade-off to the experimenter:

- Electroencephalograms (EEG) have a low signal-to-noise ratio, and capture many artefacts, mainly from head movement, making them highly impractical for ASR;
- Magnetoencephalograms (MEG) also suffer from artefact contamination, and the requirement of a large chamber for the MEG device makes it unsuited for future prosthetics devices;
- Microarrays offer unparalleled spatial and temporal resolutions, but are invasive, hence limited to academic investigation. Furthermore, their very limited spatial covering probably leads to missing crucial information processing, while possibly damaging parts of the area under investigation;
- Electrocorticography (ECoG) is a semi-invasive technology that provides high spatial and high temporal resolution, able to cover a wide brain array related to speech production or perception. The ECoG grid being on the brain surface, the recorded data

is unfiltered by the skin and skull and unaffected by glossokinetic artefacts (related to tongue movements).

As a result, ECoG is currently the most adapted technology to provide human data to an ASR back-end, allowing the application of both classification (ASR, [Leonard *et al.*, 2015]) and regression (sound reconstruction, [Pasley *et al.*, 2012]) techniques to recover neural information as speech. The results of our review are presented in table VI.9, where the focus is put on the different ways data is processed.

Microarrays on the other hand are the best technology to extract electrophysiological activity within deep areas of the brain, making it the technology most suited for learning about neural processing in the auditory pathway. As such, this thesis aims at developing an Automatic Speech Recognition back-end framework able to receive neural spike trains as its front-end. This framework and its results are presented in the following chapters.

VI.4.2 Spike Trains

Most spike train generation mechanisms found in the literature are based on two steps: first generating a cochleogram, which is to say, a two-dimensional representation of the information containing frequency-dependent fibres. In the second step, spike trains are generated from the cochleogram, thus reducing the information output by each simulated fibre. This approach is the most popular and follows the tradition of biophysically realistic models where each simulated fibre contains a sparse random and partial representation of the waveform, in a similar fashion to that presented in appendix 12.2. An example of how spike trains can be generated differently is given in [Gutig *et al.*, 2009], where spikes are equivalent to the activation of threshold detectors for frequency band energy (separating threshold onset or offset into two groups of neurons).

A few of the models tested are based on spike trains that are incomplete representations of a cochleogram or not simply used to reconstruct cochleograms. For example, [Schafer and Jin, 2014] recognise spike trains using a template based approach that recognises patterns between fibres: by associating a letter (or other symbol, but the letter case is more intuitive) to each fibre, a sequence of letters is obtained by chronologically keeping track of the activation of the fibres. A string metric is then used to find the closest template in the training set, in a way very close to how HTK matches a real sentence with a recognised sentence, enabling the calculation of the accuracy as presented in subsection VI.3.6.

The publications based on actual spike trains have a 'y' in the $\mathbb{SG}$ column of table VI.9. To avoid the reconstruction of the signal from the spike trains, most publications use cochleograms as output, rather than spike trains. The cochleogram-like reconstruction from spike train data — reversing the spike train generation — is called a neurogram. By default, in the next chapters, neurograms are obtained by convolving a Hann window with spike trains.

VI.4.3 Simulated Best Frequencies

Within a simulation, the BF is the only parameter that is varied between Auditory Nerve Fibres. Arrays of BFs uniformly spread in logarithmic scale between 100 and 8 kHz are used in upcoming chapters, following parameters used the literature and shown in table VI.2. The number of arrays used in the next chapters differ for practical reasons:

- In chapter VII, an array of 114 simulated units is used, to naturally compare results on simulated data with the results obtained on the available experimental data;
- In chapters VIII and IX, an array of 128 units is used. This number is dictated by the current maximal number of arrays available for a physiological experiment (2 multi-electrodes containing 64 channels each).

This ideal division of BFs is unlikely to happen with experimental data but is common when testing biophysical models. Conversely, electrodes might record and mix the responses of multiple neurons, thus carrying more information than that of a single ANF.

VI.4.4 Dataset Duration

An acute guinea-pig experiment depends on the state of the anaesthetised animal and the precise position of the electrodes. This observation leads to crucial considerations regarding the design of an electrophysiology experiment amenable to a speech recognition framework. This subsection introduces a model (equation VI.11) that aggregates all the constrains surrounding a physiology experiment. Said model is too incomplete yet to be used to define or refine an experiment as it stands but represents a first effort in constructing a formal framework to assess the feasibility of an experiment, lacking parameters that academic policymakers ought to be able to assess.

The experiment itself lasts for a duration D that can go beyond 20 hours, and depends on factors that are, to a large extent, beyond control: the animal's health, its reaction to the anaesthetics, the brain damages caused by the electrodes, the precision of the experimental procedure. As such, this duration can be considered random and independent from the other variables introduced here. For modelling purposes, this random variable is given a uniform distribution $D \sim \mathcal{U}([300, 1200])$ minutes. However, it is unreasonable to assume that the neurons the electrodes record from have not changed during this time. The electrode is likely to drift and record from different neurons that might appear, disappear or be damaged by the electrode. In the auditory nerve, a stable recording is often lost after a few minutes only. In the brain, this stability period tends to be longer, between 20 minutes and 4 hours.[7] It is possible to play the sounds dataset a certain number of times $N \in \mathbb{N}^*$ to the animal, but the longer the duration of the dataset d is, the more likely it is that the electrodes will

[7] Given numbers are estimates from supervisor's experience.

record from different populations of neurons over time. This constrains the duration d to be reasonably small. Note that d should include two components, mixed together to simplify the discussion: the duration of the dataset (deterministic), and the time required to get the electrode in place in between recordings (stochastic).

The sounds dataset's duration d is fixed for a defined protocol (dataset and brain areas to record from), constant through the experiments performed on n animals. For each experiment $i \in \{1, \dots, n\}$, this deterministic number and the number of repeats $N_i \in \mathbb{N}^*$ are related to the stochastic total recording duration D_i of experiment i by the formula

$$d * N_i \leq D_i,$$

which, after summation and expectation, reads

$$d * n\,\mathbb{E}[N_1] \leq n\,\mathbb{E}[D_1].$$

The stochasticity of neural data allows one to assume that, by accumulating the responses to the same dataset, the accuracy will increase. This means that, realistically, the recognition accuracy Acc[8] is an increasing function of the expected total number of repeats $n * \mathbb{E}[N_1]$ and, since $\mathbb{E}[N_1]$ is an unknown deterministic number, Acc is an increasing function of n, assuming Acc entails a robust speech recognition framework.

Using simulated data, one can assess the speech recognition accuracy that follows on from a given amount of data: for any $k \in \mathbb{N}^*$, $\mathrm{Acc}(k)$ is computable. Thus by fixing a threshold of acceptance for the recognition accuracy, arbitrarily let's say $\alpha = 80\%$, it is possible to reverse-engineer the amount of data required to build a speech recogniser that could be amenable to the accuracy α, thus implicitly defining the number of animals required for the experiment to be deemed 'successful':

$$n\,\mathbb{E}[N_1] \geq \mathrm{Acc}^{-1}(\alpha) = \min\{k \in \mathbb{N}^* \,/\, \mathrm{Acc}(k) \geq \alpha\}.$$

The value of an experiment could be modelled as

$$V(n, d, \mathbb{E}[N_1]) = \alpha_d\, d\,\mathbb{E}[N_1] - \alpha_n n, \tag{VI.10}$$

where the parameters α_d and α_n are defined by how important each aspect of the experiment is to the Home Office. Hence, since the Home Office regulation[9] requires that the data should be obtained in such a way as to maximise the expected value of using an animal (total recording time $\mathbb{E}[d \sum_i N_i] = d\, n\,\mathbb{E}[N_1]$) and minimise the number of animals used (n),

[8] This function is unknown, but could be assessed using former similar experiments, then updated as an experiment is being performed.

[9] https://www.gov.uk/guidance/research-and-testing-using-animals

an expected optimal successful experiment requires n animals with a dataset that lasts d minutes repeated $\mathbb{E}[N_1]$ times, where

$$\begin{array}{rcl} (n, d, \mathbb{E}[N_1]) & = & \text{argmax}_{n,d,\mathbb{E}[N_1]}(\alpha_d \, d \, \mathbb{E}[N_1] - \alpha_n n) \\ \text{subject to} & & \\ n * \mathbb{E}[N_1] & \geq & \text{Acc}^{-1}(\alpha) \\ d & \leq & 60 \text{ min} \\ d * \mathbb{E}[N_1] & \leq & \mathbb{E}[D_1] \\ D_1 & \sim & \mathcal{U}([300, 1200]) \text{ min.} \end{array} \tag{VI.11}$$

Other important parameters are missing from this simple model, such as the pain or distress inflicted to the subjects[10] or the statistical significance of the results.

VI.4.5 Recognition Task: Continuity, Complexity, Noise

This chapter ends with a section discussing the last important parameters that vary in the literature and are included in table VI.9: the continuity of speech segments, the complexity of the speech datasets (ranging from a single phoneme to everyday speech with multiple speakers) and the use of background noise.

Two types of recognition appear in the literature of biophysical auditory models: recognising one isolated sound at a time, or a sequence of them. The former is merely a classification task and confusion matrices give much information regarding the performance of the system. The latter, however, is harder to engineer and to assess, since a sequence can contain an arbitrary number of sounds to recognise, in theory at least. For this reason, the scientific approaches found in the literature either engineer a classifier adapted to the task at hand or tend to use HMMs to handle the dynamical aspects. This distinction is given in table VI.9 in column $\mathbb{C}$ which contains either C (for continuous speech, so that any number of symbols a priori can be recognised), I (for isolated words/phonemes), or a digit that gives the fixed number of symbols each sentence contains. About half of these entries use isolated sounds.

The publications listed in this table use datasets whose complexity matches the scientific questions studied within. Digit and phoneme recognition tasks are the most frequent, though a few other types appear such as letter recognition or a limited amount of words. This information is provided in the 'Task' column. Since consonants (C) and vowels (V) have important differences (their duration being the most noticeable), many phoneme recognition tasks are actually a grouping of phonemes such as 'CV', 'CVC' or 'VCV' (standing for Consonant-Vowel, Consonant-Vowel-Consonant and Vowel-Consonant-Vowel, respectively).

The complexity of training is also related to the size of the training set. Relevant information was added in the 'Training' column of table VI.9, including the number of speakers

[10] Current technologies permit no objective measure of pain, fear or distress, nor of the research's impact.

and their gender, or the number of times a dataset is played to the model or subject (animal or human). The name of the dataset is given in the 'Dataset' column whenever possible.

To compare the robustness of the models with that of humans, noise is often added to mask the speech waveforms with a varying signal-to-noise ratio. Different types of noise can be tested, in which case this information is given in the column 'Noise', with either the name of noise types or their number.

Ref	Front-End	𝕊𝔾	Back-End	Software	𝕃	ℂ	𝕄	𝕊	𝔻	Database	Task	Noise	Training
This thesis	[Sumner *et al.*, 2003]	y	h()_DCT_dd_HMM	HTK	P	C				VII, VIII, IX			
[Alam *et al.*, 2017]	[Zilany *et al.*, 2009]		Radon+SVM	LibSVM	P	C			I	TIMIT	18 phones	SSN, White	300 utterances per accent group
[Tavanaei and Maida, 2016]	SNN with Izhikevitch neurons, 2000 adaptive weights	y	SVM		W	I				Aurora	Digits	y	500 samples, (balanced M/F)
[Schaedler and Kollmeier, 2015]	Separated Gabor filters		HMM	HTK	W	5	1	4	I	Jepsen2008	50 words	IRCA5	
[Leonard *et al.*, 2015]	ECoG, 4 subjects, 256-electrode array, left hemisphere		Linear Regression + k-means	Matlab (k-means)	P	I				TIMIT	26 CVC		
[Schafer and Jin, 2014]	SNN with Izhikevich neurons, 35200 weights	y	VP metric + SVM		W	I			D?	Aurora 2	Digits	4 types	
[Mugler *et al.*, 2014]	ECoG, 4 subjects, frontal and temporal areas, speech production		LDA		P	I			?	Handmade	CVC		300 (read)
[Coath *et al.*, 2014]	Neuromorphic SNN	y	LPC	Matlab (Ellis 2005)	W	C			D	Handmade	FM, 7 monosyllabes	White	Self, 3 sweeps
[Bouchard and Chang, 2014]	ECoG, 2 subjects, 256 channels, single-trial, speech production, left hemisphere		Gamma bands + PCA + LPC		V	C				Handmade	CV (19C/3V)		15 to 100
[Yildiz *et al.*, 2013]	[Lyon, 1982]		Bayes+Dynamical model + DynamicEM	Matlab (Auditory)	W	I			D	Ti-46	Digits	White	1 (10 repetitions)
[Choi, 2013]	[Choi, 2013]		DCT+HMM		W	I	3	14	I	Aurora 2	Digits	10 types	55F/55M (8440 training utterances, 4004 testing)
[Pasley *et al.*, 2012]	ECoG, 15 subjects, left or right fronto-parietal regions, high-gamma band		Reconstruction by comparing SRTF with spectrograms	Matlab (STRFLab)	W	I			I	TIMIT	Listening, detecting or repeating		
[Clark *et al.*, 2012]	[Brown *et al.*, 2010]		h(25)_DCT_dd_HMM	HTK	W	3	7	16	I	TIDIGITS	9 Digits	Pink, babble	8440 utterances, half/half
[Brumberg *et al.*, 2011]	ECoG, 1 subject, intended speech production		LDA, SVM, FDA	R (MASS, mda, e1071)	C	I				Handmade	CV (39 C)		10 repetitions of each sound
[Brown *et al.*, 2010]	[Brown *et al.*, 2010]		h(25)_DCT_dd_HMM	HTK	W	C	7	16		Aurora 2	10 Digits	Pink	8440 utterances training
[Gutig *et al.*, 2009]	[Gutig *et al.*, 2009]	y	[Gutig *et al.*, 2009] + HMM	HTK	W	I	8	5		TIDIGITS, TI-46	7 Digits	White	10 repetitions training, 16 testing
[Flynn and Jones, 2008]	[Li *et al.*, 2000] + speech enhancement		DCT_dd_HMM	HTK	W	C	3	16	I	Aurora 2	Digits	8 types	8440 utterances to train, 200 dvpt, 1001 test (balanced F/M)
[Mesgarani *et al.*, 2008]	4 ferrets, A1 cortex, awake	y	PCA+SVM		P	C			I	TIMIT	Phonemes		90s, 15m/15F, repeat 5 times of speech
[Jeon and Juang, 2007]	[Yang *et al.*, 1992]+[Wang and Shamma, 1995]		h(25)_DCT_dd_g_HMM	HTK	P	C	40	2?	I	TIMIT (without SA)	48 Phonemes	White	
[Holmberg *et al.*, 2007]	[Holmberg *et al.*, 2007]	y	h(25)_DCT_g_dd_HMM, IPIH_DCT_g_dd_HMM	HTK	W	I	6	4	I	ISOLET	Letters	8 types, RSG-10	75F/75M
[Ghitza, 1986]	[Ghitza, 1986]		DTW	Rabiner's	W	I			D	Handmade	Alphabet, Digits, Controls	White	2F/2M

Tab. VI.9 Review of the application of ASR technologies to neural data, modelled or experimental. Abbreviations: 𝕊𝔾: Spike generation, 𝕃: Modelling level (W full word, P phoneme, V vowel, C consonnant), ℂ: Speech continuity (Continuous, Isolated, or having a fixed number of utterances), 𝕄: Number of Gaussian mixtures (non silence models), 𝕊: Number of emitting states in HMMs, 𝔻: Dependency on the speaker (Dependent or Independent). Front-ends include models presented in table VI.2, ECoG data, spike trains obtained by Spiking Neural Networks or Neuromorphic engineering. Back end chains use the following components: h(25) (Hann window with a 25 ms duration), DCT Discrete Cosine Transform, IPIH is a histogram feature similar to Ghitza's EIH, dd for the delta and delta-delta coefficients, g for a gaussianisation of all features (change their mean to 0 and variance to 1), LPC a Linear Predictive Coding, VP is the Victor and Purpura metric. Information regarding the tasks, training, database and types of noise (White, Pink, Speech-Shaped Noise or others) are given in the last columns.

VII.

Neural ASR on a Small-Sized Vocabulary

In this chapter, HMMs are trained on spike trains previously recorded as responses to a simple class of sounds collected by Mark Steadman [Steadman, 2015] or simulated with a cochlear model. A first simple set of processing blocks are then tested on this sparse data, as shown in figure VII.1.

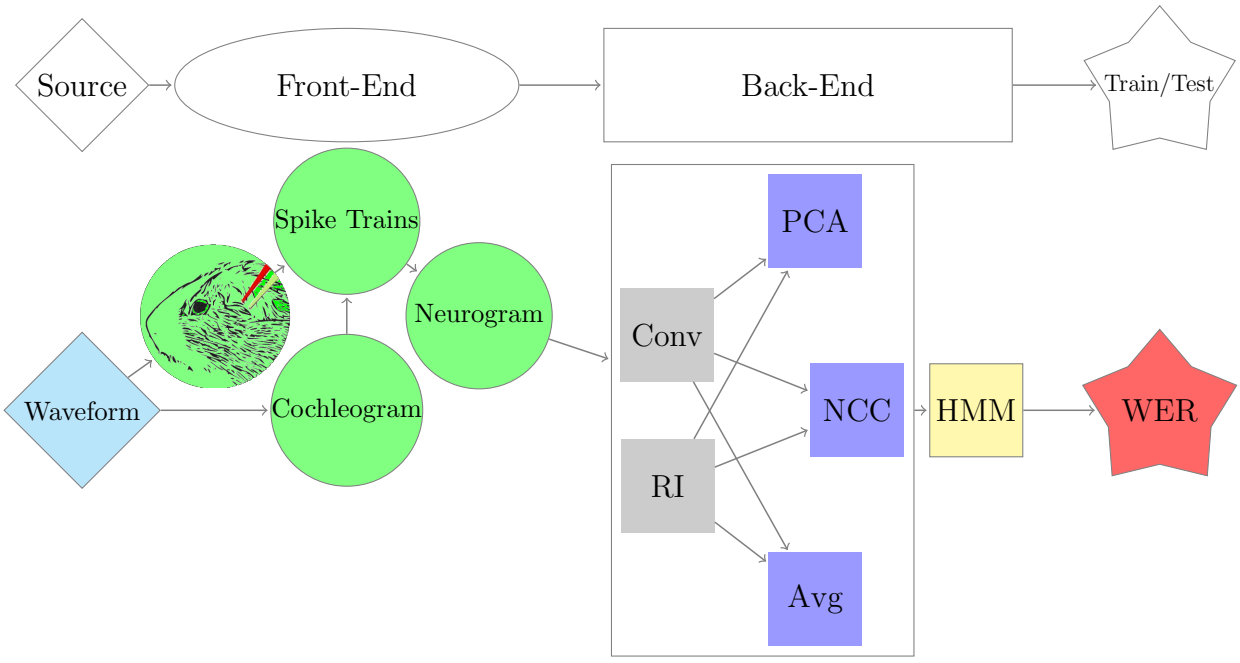


Fig. VII.1 Graph of data processing of this chapter. Spike trains come from simulation or experiment are processed using different methods, including a dimensionality reduction. Conv: Smoothing by convolution with a Hann function, RI: rate by inversion of interspike intervals and smoothing, NCC: Nearest Cluster Center, Avg: Averaging spike trains together.

VII.1 Introduction

This chapter tests on a simple task the computational architecture developed to perform speech recognition on neural data. This task is the classification of the spike train responses to short isolated sounds, either recorded from guinea-pigs Inferior Colliculus [Steadman, 2015] or simulated by a guinea-pig ear model [Sumner *et al.*, 2003]. Six processing methods are then evaluated and compared (two spike train smoothing times three dimensionality reduction methods). The research goal is to evaluate whether simulated AN data gives results comparable to experimental data, and whether this computational framework can cope with the task of recognising simple isolated sounds. With this in mind, the research questions are:

- Compare the simulated and experimental spike train data using simple processing, varying the number of spike trains used;
- Compare the generalisation ability of the models when testing the HMMs on a speaker removed from the training set and see whether the optimal time window of integration changes in this case.

Ideally, experimental IC data would have been compared with simulated IC data (and experimental ANF data with simulated ANF data). However, the comparison of physiological IC data and simulated AN data is motivated by the availability of guinea-pig physiological IC data as response to speech and the absence of satisfying IC models.

VII.2 Methodology

VII.2.1 SteadaCa Dataset

VII.2.1.1 Dataset Description

The spike trains used are responses to spoken sets of Vowel-Consonant-Vowel (VCV) such as /aba/ or /ada/, recorded over a period of 700 ms each. The 16 medial consonants used were /b, d, f, g, k, l, m, n, p, s, ʃ, t, ð, v, y/ and /z/, commensurate with the stimulus set used in [Shannon *et al.*, 1995] and these were always in an /a/-consonant-/a/ context, where /a/ is the open back unrounded vowel as in the word 'palm'. Each VCV is said by 3 English native male speakers, constituting a waveforms dataset of 48 spoken VCV. This dataset is referred to as the SteadaCa.

VII.2.1.2 Validation Protocol

HMMs are trained on neural responses to the SteadaCa dataset, using HTK version 3.4 [Young *et al.*, 2006], to classify the medial consonants produced by the speaker. The evalua-

tion is done by separating the data into training and testing sets. The way this split is done has two variants: either the responses of 8 repetitions of the same sound are used as training and the other 2 as test, or all instances from 2 speakers are used to train and responses to the third speaker are used as test.

VII.2.2 Front-End

VII.2.2.1 Experimental Data

The spike trains are recorded from the Inferior Colliculus of guinea-pigs as responses to the SteadaCa dataset, played 10 times per recording. The full set of responses contains 54720 spike trains, from 3 male speakers, 16 VCVs, 114 different neurons, with 10 sweeps per stimulus and neuron. See [Steadman, 2015] for additional information regarding the experimental methodology.

VII.2.2.2 Simulated Data

Simulated spike trains were obtained using the cochlear model [Sumner *et al.*, 2003] described in section VI.1.2 and appendix 12.2. The probability of firing is first obtained using a biophysical model with a sampling rate of 100 kHz. Spike trains are then generated as an inhomogeneous Poisson processes with an absolute refractory period of 3 ms. All computational details are given in appendix 12.

To proceed to a comparison of simulated and experimental data, and analyse the behaviour of HMMs on the SteadaCa dataset, important parameters of the cochlear model they are set in this chapter to the following values:

- BFs are uniformly spread in logarithmic scale between 100 Hz and 8 kHz;
- The calcium threshold $[\mathrm{Ca}^{2+}]_{\mathrm{thr}} = 1 * 10^{-15}$, unit free;
- The maximal calcium conductance $g_{Ca}^{\max} = 7 * 10^{-9}$ S.

They are discussed in chapter VIII.

VII.2.3 Back-End

A comparison of rate or rate+temporal decoding was performed in [Holmberg *et al.*, 2007] on simulated data, showing that a rate code carries all the speech information at normal sound levels (around 70 dB SPL) and in the absence of background noise. Hence, a rate coding for the spike trains is assumed. The front-end block outputs pictorial (2-dimensional, one for time, the other related to BFs) representations that contain sequences of feature vectors in time obtained by processing single spike trains and then reducing the dimensionality. Those two stages correspond to the two layers in the first 'back-end' block of figure VII.1. The following paragraphs detail this processing.

VII.2.3.1 Single Spike Train Processing

Following Steadman [Steadman, 2015], we create feature vectors from pictorial representations of neural activity, called neurograms, where each column corresponds to a time bin of 10 ms, each row corresponds to a group of processed single neuron spike trains or the grouping of spike trains, and the real value at each pixel is related to the spiking activity around that time, as exemplified in figure VII.2. Columns in these neurograms correspond to the overall activity within a time bin and will be the feature vectors for our HMMs.

There are many ways to regroup information from different spike trains into feature vectors. Since we work with HTK we aim at obtaining meaningful feature vectors for HMMs, which is in itself not a well-defined problem but is understood enough to be able to decide of important aspects of the processing. Previous testing guided the evaluation of the necessary trade-offs, such as:

- The more spike trains that are used to create a feature vector, the smaller the number of training elements in our datasets;
- The more features we add to improve the result, the more we tend to overfit; manually crafting features after testing on the same dataset is a form of overfitting;
- HMMs with too many features tend to behave poorly; feature vectors of size less than 50 are still acceptable, but hundreds are not;
- Since the HMMs have (an assumed) diagonal covariance matrix to simplify the training and testing, using correlated features may be detrimental;
- Extracting features after aggregating spike trains deteriorates temporal coding and improve rate coding.

In this chapter, only two single spike train processing are used:

- **Conv:** Convolution of spike trains by a Hann window of varying length. The most typical duration is 25 ms as seen in table VI.9, but testing other values provided valuable insight. This sliding window is applied every 10 ms;
- **RI:** estimates the spiking rate by inverting the ISI between two consecutive spikes[1]. The sequence is then smoothed down and down-sampled (from 100 kHz to 0.1 kHz) by a sliding Hann window with varying duration.

This time scale of 10 ms is common in ASR systems, as it achieves a balance between an appropriate time resolution for speech recognition and speed of the computations. With HMMs, this matter is related to the number of emitting states and the transition matrix.

[1] For a homogeneous Poisson spike train, this statistic is, on average, the rate. See details in appendix 12.7.

VII.2.3.2 Dimensionality Reduction

A final processing step of dimensionality reduction is used to reduce the complexity of the models, and increase the robustness of the features. For any number of neurons used in this chapter (3, 10, 15, 30, 60, or 114), the reduction of dimensionality was fixed so that the dimension would increase with the number of neurons (3, 5, 7, 10, 15 or 20, respectively). The different ways of regrouping features used are:

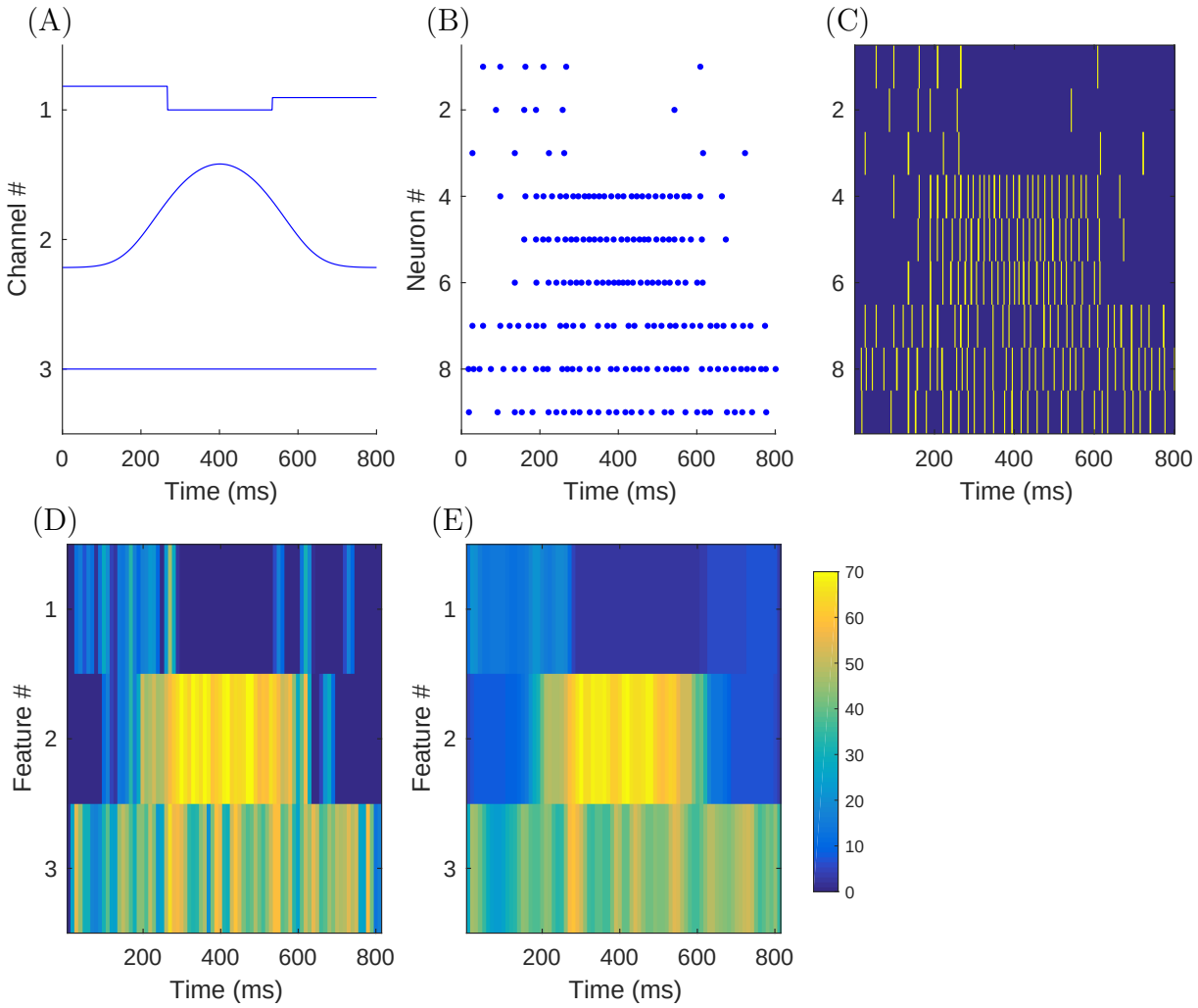


Fig. VII.2 Example of neurograms generation: (A) Shapes of spiking activity used to generate spike trains, renormalised for display (B) Raster plot of 9 simulated spike trains (3 per channel) (C) Neurogram (pictorial representation of size 9x800) as used in [Steadman, 2015] to calculate euclidean distances (D) Neurogram (pictorial representation of size 3x80) more typical in our calculations: spike trains are convolved with a renormalised Hann window of 40 ms to give a rate (number of spikes per second per spike train); the features are then averaged (mixing neurons 1, 2, 3 into feature 1, neurons 4, 5, 6 into feature 2, neurons 7, 8, 9 into feature 3). This processing is equivalent to applying 'Conv' followed by 'Avg' with a dimension reduction of 3 (E) Similar neurogram, with processing 'RI' followed by 'Avg'.

- **Avg:** Average some smoothed spike trains together. Spike trains are not ordered beforehand, so we expect to lose some useful information by mixing responses corresponding to different characteristic frequencies. For this reason, we do not expect this method to bring satisfying results; it is still tested to provide a baseline for dimensionality reduction methods. For simulated data, neurons are sorted by increasing BF;
- **PCA:** Run a Principal Component Analysis (II.7.1) on the neurograms. The PCA P_C is trained on the full neurograms training set. Reading the matrices' size gives:

$$\underset{70\times 114}{N} \quad \times \quad \underset{114\times 20}{P_C} \quad = \quad \underset{70\times 20}{\tilde{N}}$$

  for N a neurogram and its projection $\tilde{N}$, when using 114 neurons;
- **NCC:** To improve on our naive averaging, this approach creates clusters of relatively similar spike trains, and averages across neurons regrouped within a common cluster. A clustering algorithm (k-means) decides which neurons are grouped together, giving the algorithm the number of clusters we want as the number of features we wish to train HMMs with - which depends on the number of neurons used. Smoothed spike trains from neurons belonging to the same cluster will then be averaged together to reduce the number of variables. As for the PCA processing, the clusters are found using the whole training set.

With each processing, we are interested in the parameters and datasets allowing high recognition accuracy.

VII.2.4 HTK Parameters

In the following, unless stated otherwise, we use for each HMM 5 states (3 emitting) with 2 Gaussian mixtures per state. This number of states is not the one advised by the HTK book [Young *et al.*, 2006], where the number of recommended states per HMM is one plus 2 per #phonemes, which would give 7 states for VCV HMMs. By keeping 3 emitting states, we hope to train a more speaker-independent recogniser, as adding states might overtrain the /A/ sounds to specific spectral properties of a speaker's voice rather than to the more general attributes of /A/ sounds.

This choice proved to work well enough, as it was possible to obtain a score of 100% accuracy with simulated data, and almost perfect score with experimental data.

VII.3 Results

Our goal is to find a good balance between the processing applied to spike trains, the data aggregation and the amount of data required to obtain high classification performance.

VII.3.1 Number of Neurons

Neurons in the auditory pathway encode different pieces of information about the incoming sound, at any given moment. Thus, aggregating spike trains together improves the representation of the sound. Since we don't know initially which neurons encode which information, we test our decoding scheme using a varying number of neurons, and using the methods previously discussed for spike train processing and dimensionality reduction.

Figure VII.3 summarises our results, on both experimental and simulated spike trains. The x-axis gives the number of neurons tested (3, 10, 15, 30, 60, 114 neurons, respectively reduced to 3, 5, 7, 10, 15 and 20 features) and each curve represents a different spike train processing (marker type) and dimensionality reduction method (colour).

The full dataset is made of 114 spike trains. The simulated spike trains are randomly generated as Poisson processes with refractory periods, from firing probabilities generated by the model. Each training set was made of 192 neurograms (4 repeats * 16 VCVs * 3 speakers), and each test set of the remaining 48 ones. All results presented are obtained though 5-fold cross-validation.

The experimental data on which the left plot of figure VII.3 are based were not given any specific order prior to training and testing. From this dataset, all available data is used (114 neurons). Curves from the right plot are based on simulated cochlear data. High sound levels tended to saturate some auditory fibres, worsening the results, so only the results of

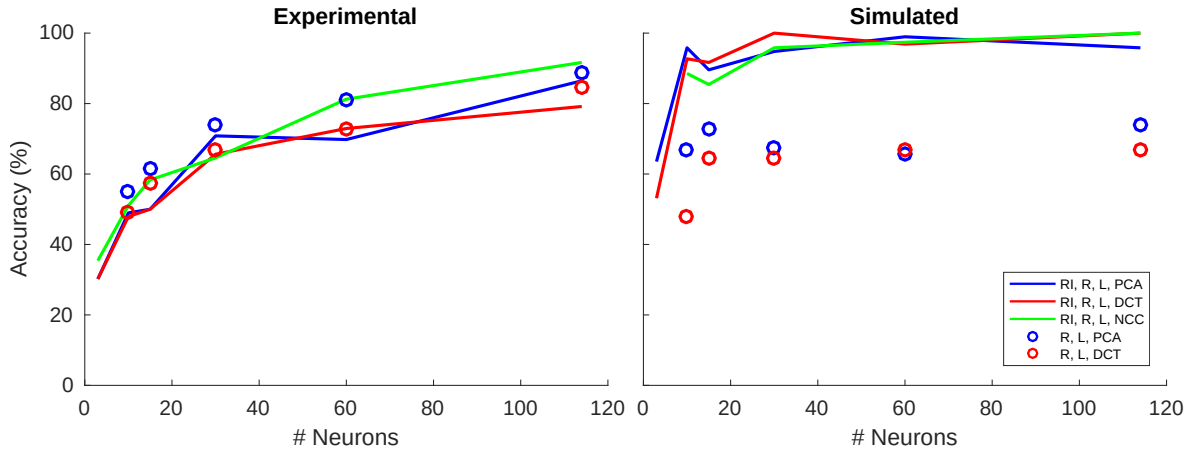


Fig. VII.3 Classification results on IC data (left) and simulated cochlear data at 70 dB (right). Two spike train processing are used: convolving with a Hann window of 40 ms and applying a logarithm ('R, L', dots) or rate inversion (RI, dashed line) followed by a local average (R, 40 ms) and logarithm (L). Three dimensionality reductions are used: Averaging a k-means cluster (NCC, green), train a PCA on the full training set (blue), using a DCT (red). The number of features for 3, 10, 15, 30, 60 and 114 neurons is respectively reduced to 3, 5, 7, 10, 15 and 20. Missing values resulted from errors in HTK. Two Gaussian mixtures and 3 emitting states were used by each letter's HMM.

sounds played at 70 dB SPL are presented. This improved the accuracy systematically, and allowed us to reach 96.25% of accuracy with simulated data.

Using the simulated data, the best results using 60 neurons were almost as good as when using 114, while they continue to improve for the experimental data. In both cases, the results go above 80% accuracy. As we work towards building a speech recogniser on a larger dictionary, more processing or parameters will be tested.

VII.3.2 Optimal Temporal Windows

One would expect that longer temporal integration would be required for an optimal recognition on the response from speakers not included in the training set, since the features shouldn't be as precise in time on an unseen pattern. To test this hypothesis, we vary the length of the Hann window used in each of the six processing (with or without ISI inversion, and three dimensionality reduction) between 10 ms and 480 ms; since signals last for 700 ms, the longest Hann windows essentially encode the rate. Results are shown in figure VII.3. We compare the optimal Hann duration when some speaker is excluded from the training set (blue, red and black curves), or when responses from all speakers are used both for training and testing (green curves).

As seen in figure VII.4, training and testing on all speakers allows for a much higher accuracy than testing on unseen speakers. On average, attributing a null value to missing values, and training on all 3 speakers leads to an accuracy of 77.71%, against 44.60% when training on two and testing on the third (47.01%, 45.48% and 41.30% when testing on speaker number 1, 2 and 3 respectively) on the experimental data, and on the simulated data, 74.19% versus 31.21% (36.09%, 25.09%, 32.46%).

The optimal Hann window is also strongly influenced by this separation of the speakers. On average in the experimental data, the mean optimal Hann window is 290 ms when training on the 3 speakers and 85.4 ms when training on 2 speakers (83.7, 87.7 and 84.9 ms when testing on speaker number 1, 2 and 3 respectively), keeping all values (to calculate the mean) when more than one Hann duration gives the highest accuracy for a given processing. These values are specified by the crosses on the x-axis in figure VII.4. For the simulated data, it is not as meaningful to average the optimal Hann windows when training with the 3 speakers (231 ms), since all processing allow us to reach a plateau at 100%; except for R-L-NCC that is close. The mean optimal window when separating speakers is 184 ms (218.7, 123.2 and 209.7 ms). This shift of an optimal time window (from 85.4 ms on the experimental data to 184 ms for the simulated data) goes with a reduction in accuracy. A possible explanation is that, as features provide relatively less speech information, a longer time window of integration is required to achieve maximal robustness.

Comparing accuracies might not be enough, and the systematic comparison of confusion matrices from simulated or experimental data may reveal patterns in the errors made. This,

however, does not seem to be the case: when summing all confusion matrices corresponding to a single speaker (1, 2 or 3) and a single type of data (experimental or simulated), we obtain six mean confusion matrices; note that only when both experimental data and simulated data provided a confusion matrix (rather than an HTK error) were the matrices used in this averaging. Let us now subtract the 3 mean confusion matrices obtained with simulated data from the respective 3 mean confusion matrices obtained with experimental data. This provides us with 3 square matrices with entries between 100 and -100, shown in figure VII.5. Any systematic difference in classification on experimental and simulated data should be a common pattern of these three matrices. The strongest claim these matrices suggest is that 'AFA' is robustly better recognised on experimental data. To verify there was no important pattern, table VII.6 shows for each of the 6 mean confusion matrices described above, the most frequent error for each of the 16 consonants. As expected, no pattern (couple of most likely error on simulated or experimental data) is the same for all 3 speakers.

The fact that, using simulated ANF data and appropriate processing, HMMs with 2 Gaussian mixtures per HMM can so easily get 100% of accuracy suggests that the task in this case is too simple for a complex and well-developed technology such as Hidden Markov

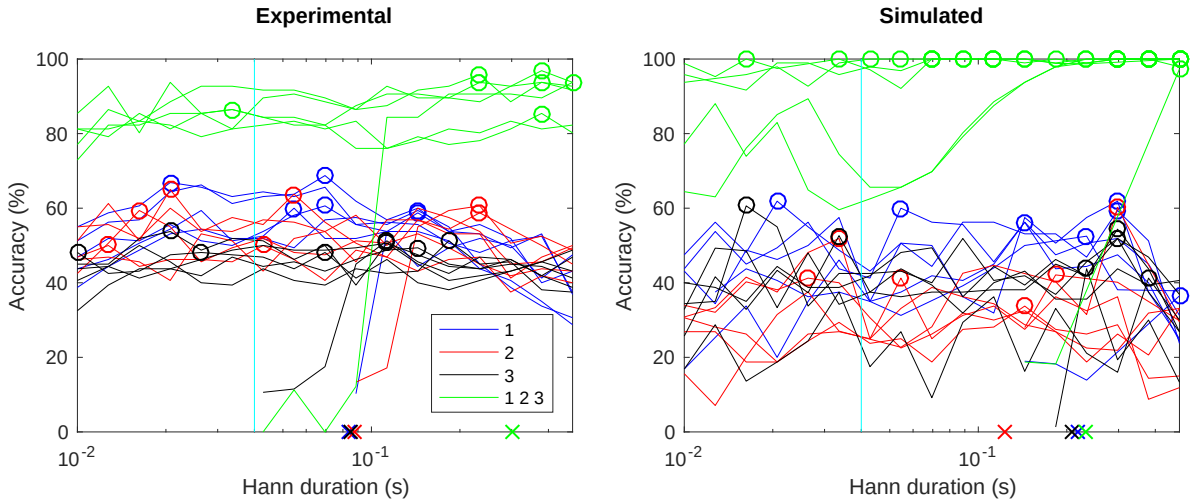


Fig. VII.4 Accuracy (y-axis) in our ASR task on experimental IC data (left) or simulated cochlear data (right), as a function of the Hann window duration (x-axis), when training and testing on all speakers or splitting them (training on two of them, testing on the third). Each colour defines a training and test set: the blue (1), red (2) and black (3) curves are obtained by training on 2 speakers and testing on the third one. The 10 repeats of each VCV are included. The green curves show the result of training on 8 repeats from the 3 speakers and testing on the 2 other repeats from all speakers. All 16 VCV sounds are used. For each colour, the 6 curves show the result of the 6 back-end processing presented in VII.3. Each curve's maximal values are highlighted by a circle. The mean optimal Hann window length is shown for each set of curves on the x-axis by a cross. The vertical cyan line shows the $x = 0.04$ s value, Hann duration used for the results shown in figure VII.3.

Models. Since the firing rate output by this nerve fibres model was fitted to guinea-pig data and the processing used in this chapter focuses on rate code, we would expect that the differences observed on experimental and simulated data are mostly due to the different places it comes from or aim to reproduce: Inferior Colliculus and Auditory Nerve, respectively. Let's also note that the fact that all useful BFs are present in the simulated AN data by design but may not be present in the experimental IC data plays, a priori, in favour of having higher results on the AN data.

A more complex task is required to test the recognition capability of such a system. We do so in the next chapter, moving to a digit recognition task.

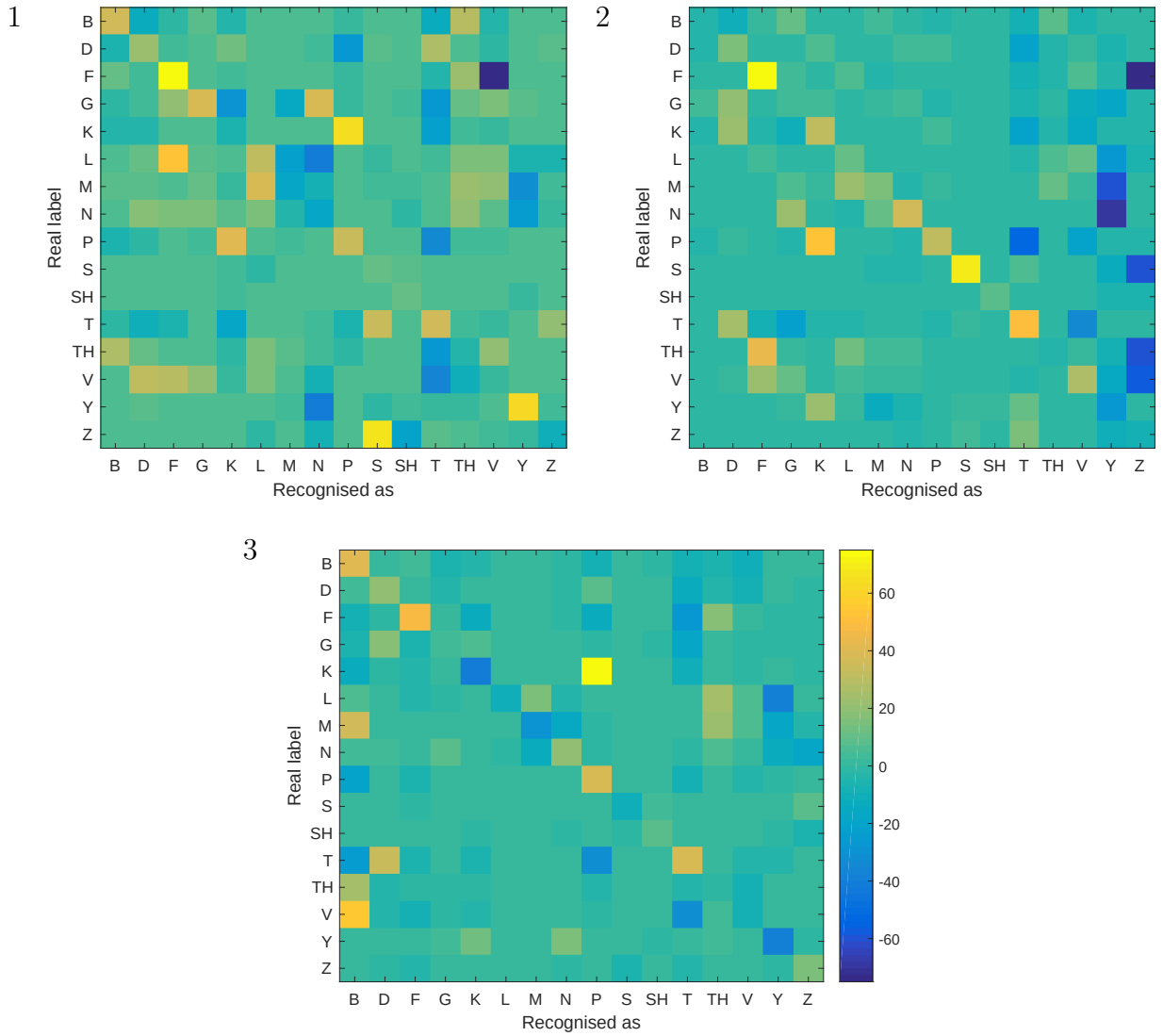


Fig. VII.5 For each speaker (1, 2 and 3), the difference of the mean confusion matrices obtained with experimental and with simulated data is shown. For clarity, only the medial consonant is given in the labels. See full description in the text.

	Speaker 1		Speaker 2		Speaker 3	
Class	exp	sim	exp	sim	exp	sim
B	TH 19.0	V 23.8	TH 35.4	V 32.2	F 6.0	V 11.6
D	T 37.0	T 24.3	V 18.9	T 19.8	B 48.5	B 46.0
F	TH 11.6	V 57.9	V 8.8	Z 73.9	TH 22.2	T 27.0
G	N 20.1	K 20.7	D 22.2	Y 18.4	B 32.3	B 37.7
K	P 38.9	T 17.3	D 22.2	G 24.9	P 84.4	B 13.7
L	F 27.7	N 35.8	V 10.8	Y 27.0	M 46.6	Y 39.7
M	L 24.7	N 25.7	L 35.0	Y 60.0	B 35.2	N 20.7
N	TH 11.4	Y 20.5	G 23.4	Y 71.5	M 17.0	M 29.8
P	K 23.3	T 22.8	K 54.9	T 53.0	K 11.8	B 20.8
S	SH 2.7	L 3.5	Z 9.7	Z 67.3	Z 17.1	Z 9.6
SH	S 0.1	Y 2.4	S 0.4	Y 4.6		Z 5.5
T	S 17.2	K 13.3	D 31.8	V 32.4	D 36.8	P 33.6
TH	V 14.4	T 19.1	F 48.8	Z 58.9	B 90.0	B 66.4
V	D 19.4	T 25.9	F 26.9	Z 56.5	B 86.6	B 31.8
Y	L 6.2	N 28.9	K 22.7	M 12.6	N 38.8	N 24.1
Z	S 87.0	S 50.1	T 19.9	Y 8.9	S 5.6	S 10.8

Tab. VII.6 Table showing the most common misclassification for each consonant, speaker and type of data (experimental or simulated). For each consonant (row), speaker (bi-column) and type of data (column), the red letter shows the most frequent error among the classifications used in figure VII.4, and the number next to it is the percentage of guesses this error represents in a row of the mean confusion matrix used. A set named by a speaker designates the set where this speaker was removed from the training set and used as test set. In case a class is always correctly classified in the confusions matrices kept for this table, no red letter and no percentage are given.

VII.4 Conclusion

This chapter tested a speech recognition framework using guinea-pig Inferior Colliculus responses to isolated VCV waveforms, using simple processing as a testbed. These results were compared with the results obtained similarly with simulated auditory nerve data.

The differences in accuracy when testing the HMM models on a speaker's responses left out from the training set hints at the integration of useful features for speech recognition in the IC that are not present in the auditory nerve. This motivates the further use of guinea-pigs as an animal model for speech processing research in humans.

The results obtained with simulated data allowed a perfect score, using a wide range of time-scale integration for different processing, showing that this dataset is too simple to test the recognition power of techniques with simulated data. To test further the abilities of the recognition system built, the next chapter studies the responses to continuous speech using a classical recognition task: digits.

VIII.

Neural ASR on a Medium-Sized Vocabulary

In this chapter, HMMs are used to build models for a digit recognition task in continuous speech to assess the suitability of this task for a physiology experiment. This represents an increase in complexity — more speakers, more phonemes, and continuous speech — since the VCV dataset was too simple to test the limits of a speech recogniser on neural data. In so doing, aspects of Auditory Nerve coding at different sound levels is investigated. A set of parameters given in red in the front-end block of figure VIII.1 is extensively studied, while the back-end's parameters are set.

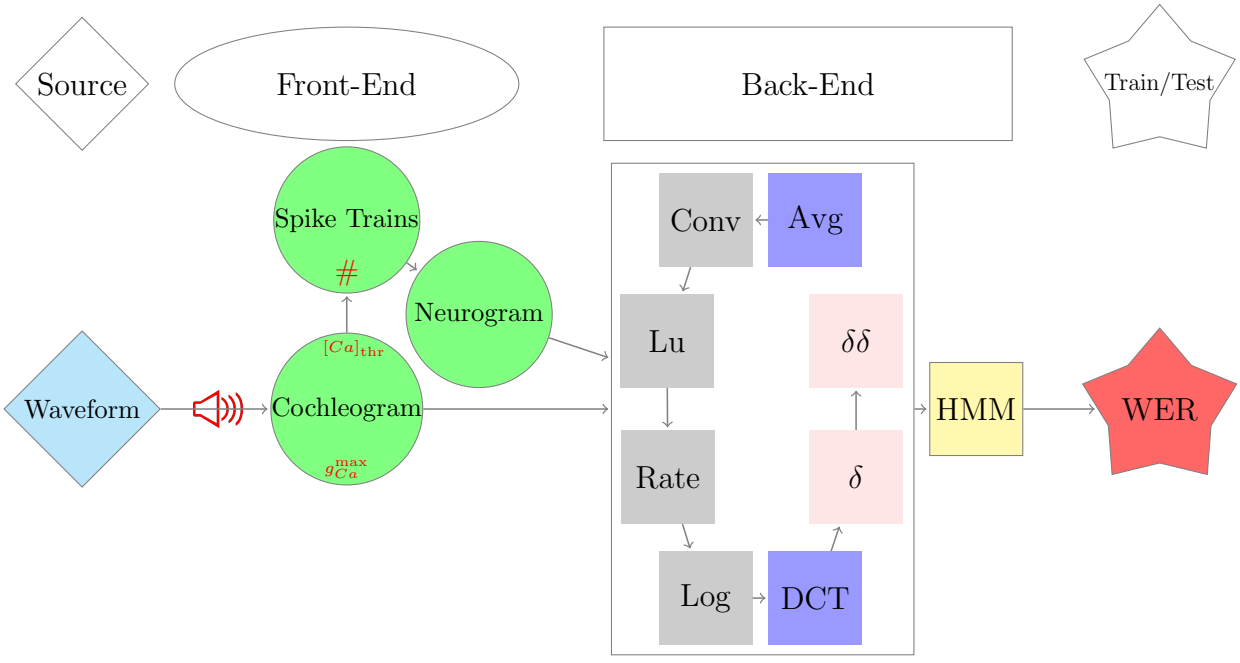


Fig. VIII.1 Graph of this chapter's data processing. In the front-end block, the parameters in red are the ones studied: sound level, maximal calcium conductance, calcium threshold for neurotransimitter release, and number of auditory nerve fibres per best frequency. The back-end block is fixed and explained in the chapter: dimensionality reduction nodes in light blue, processing in grey, data augmentation (speed and acceleration) in pink.

VIII.1 Introduction

In this chapter, HMMs are used to build models for a digit recognition task in continuous speech to assess the suitability of this task for a physiology experiment. This represents an increase in complexity — more speakers, more phonemes, and continuous speech — since the previous VCV SteadaCa dataset was too simple to test the limits of a speech recogniser on neural data.

The model's parameters might be a bottleneck for speech information transmission through the ANF. As such, the approach taken is to investigate whether the current dataset is suitable for an electrophysiological experiment, exploring what the model's parameters that allow good recognition accuracies are, and what these results teach us on the neural coding of speech in the ANF at different sound levels.

To compensate for the increase in complexity, the number of spike trains may need to be increased to extract more robust features. Contrary to physiological data, the number of spike trains is easily changed when simulating the data with a computational model. By pushing this number to the limit, the averaging of an infinity of spike trains would allow the recovery of the firing distribution probability used to generate the spike trains. Hence, results from a varying number of spike trains can be compared with results obtained similarly on cochleograms, being then a proxy for an infinity of (refractory period free) spike trains.

VIII.2 Methodology

This section presents the methodology used, and contrasts it with that of the previous chapter, regarding the dataset, the front-end and the back-end, before testing this setup on a continuous digit recognition task on simulated neural data.

VIII.2.1 sCUAVE Dataset & Validation Protocol

The CUAVE database [Patterson *et al.*, 2002] is an audio-visual speaker-independent corpus of over 7,000 digit utterances. It offers numerous advantages: it is free, contains both audio and video with labels, has different speed of dictions, includes male and female speakers, and has both connected and isolated digits — read in a continuous stream or separately, respectively, as discussed in subsection VI.4.5.

Following the constraints detailed in section VI.4.4, only a subset of the CUAVE dataset is used on a continuous digit recognition task. See table VIII.3 for statistics of this subset, named sCUAVE. All initial files were split into smaller ones after locating silences, in order to provide shorter waveforms for the cochlear model that obviate RAM memory limitations.

The sCUAVE dataset was separated into training, development and testing sets. While

a cross-validation protocol gives more statistically significant results, the additional computation time was crucial for the vast amount of calculations presented below.

Due to the silence issue presented in subsection VI.3.8, most files containing long silences were removed from the test and development sCUAVE dataset; some were left there, since this would still allow the comparison of different data processing in the front-end.

VIII.2.1.1 Word-Level & Phone-Level

Digit recognition tasks are normally evaluated on the word-level accuracy, using the principle presented in section VI.3.5 to infer a sequence of words using a dictionary and phonemes' statistical models. However, to compare results with other speech recognition results, a phoneme-level accuracy fits better to this thesis. To satisfy these requirements, both accuracies are given in all results. Using word-level HMMs is common when working on isolated sounds and a small dictionary. However, since this approach does not scale up well with the dictionary and is not in the spirit of this thesis, phoneme-level HMMs are built rather than word-level HMMs, for each result. The dictionary used is presented in table VIII.2.

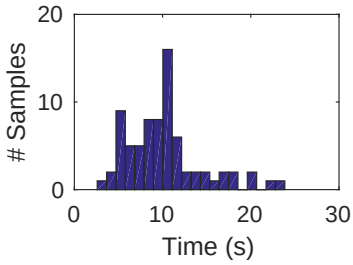
On top of the explanations provided in subsection VI.3.5, here is an example of the word-level output for a continuous digit task. In figure VIII.4, an example of HTK's output is given, where the number of hits H, substitutions S and total samples N is given at the sentence level[1] 'SENT' (full transcript of a file, N=25 meaning that the test set contains 25 files) and the word level 'WORD' which also contains the number of deletions and insertions. The confusion matrices include a column for the number of deletions of the real labels, and a row of insertions of recognised labels. A transcript is also provided in VIII.5 to illustrate

EIGHT	ey t
FIVE	f ay v
FOUR	f aa
FOUR	f aa r
NINE	n ay n
ONE	w ah n
SEVEN	s eh v n
SIX	s ih k s
THREE	th r iy
TWO	t uw
ZERO	z ia r ow

Tab. VIII.2 Dictionary used: each of the 20 phonemes has a corresponding HMM model, plus two for silences (explained in VI.3.8).

[1]The sentence level does not offer any interest in this thesis and as such was not discussed earlier. However, it can be useful for industrial standards, as a way to assess how close a speech recognition application is to being ready for delivery.

	Training	Testing	Development	Total
# Males	10	6	2	10
# Females	6	3	2	6
# Digits	546	260	66	872
# Files	50	25	6	81
Duration (s)	513	252	64	829



Tab. VIII.3 Statistics on the sCUAVE dataset. Samples from common speakers are spread in all sets. On the right, a histogram of the files' durations, with an average of about 10 digits per file.

the different types of errors: the real transcript (line starting with 'LAB') is aligned with the recognised one (starting with 'REC') and the three types of error appear: deletion, insertion and substitution.

VIII.2.2 Front-End

VIII.2.2.1 Front-End Processing

The data used for classification takes two forms: cochleograms or spike trains generated from a cochleogram. To be able to test both data types the same way, spike trains are processed in the front-end block to become a neurogram, which should be amenable to the same accuracy as with cochleograms, using enough fibres.

The simulated ANFs used are 128 units whose BFs are logarithmically spread between 100 Hz and 8 kHz, as discussed in subsection VI.4.3. To compare the results using fibres with different spontaneous activity, different values for the maximal conductance in the model are also tested; this is discussed in detail in the next subsection.

VIII.2.2.2 Sound Level & Neuronal Spontaneous Activity

As suggested in chapter VII, sound level has a major impact on recognition accuracy when using a cochlear model. This effect is well-known [Holmberg *et al.*, 2007; Young and Sachs, 1979] but not entirely understood, as it relates to the precise way speech is encoded at different sound levels. A simple way to present this is by looking at the rate/intensity (RI) function of the AN fibres.

```
LAB: SEVEN ONE TWO EIGHT THREE FIVE SIX ZERO FIVE NINE
REC: SEVEN     TWO EIGHT THREE FOUR SIX      FOUR NINE
```

Tab. VIII.5 Transcript for the digit recognition task: the real labels are on top and the aligned recognised labels below.

```
------------------- Overall Word-Level Results --------------------
SENT: %Correct=28.00 [H=7, S=18, N=25]
WORD: %Corr=80.77, Acc=73.85 [H=210, D=33, S=17, I=18, N=260]
------------------------ Confusion Matrix -------------------------
       E   F   F   N   O   S   S   T   T   Z
       I   I   O   I   N   E   I   H   W   E
       G   V   U   N   E   V   X   R   O   R
       H   E   R   E       E       E       O
       T                   N       E           Del
EIGH  23   0   0   0   0   1   0   0   1   0    1
FIVE   0  18   1   1   0   0   0   0   0   0    4
FOUR   0   0  22   0   0   0   0   0   0   0    4
NINE   0   0   0  18   5   0   0   0   0   0    4
 ONE   0   0   0   0  19   1   0   0   0   0    6
SEVE   0   0   0   0   0  23   0   0   0   0    3
 SIX   0   0   0   0   0   1  23   0   0   0    2
THRE   0   0   0   0   0   1   0  21   0   0    4
 TWO   0   0   0   0   0   0   1   0  23   1    2
ZERO   0   0   1   1   0   0   0   0   1  20    3
Ins    2   0   1  11   2   2   0   0   0   0

------------------ Overall Phone-Level Results --------------------
SENT: %Correct=0.00 [H=0, S=25, N=25]
WORD: %Corr=58.15, Acc=42.75 [H=453, D=217, S=109, I=120, N=779]
------------------------ Confusion Matrix -------------------------
       a   a   a   e   e   f   i   i   i   k   n   o   r   s   t   t   u   v   w   z
       a   h   y   h   y       a   h   y           w               h   w
                                                                               Del
  aa  20   0   0   0   0   0   1   0   0   0   0   0   0   0   0   0   1   0   1   0    3
  ah   0   6   0   0   0   0   0   0   2   0   0   0   0   0   0   0   1   3   0   0   14
  ay   0   0  28   0   0   0   0   0   1   0   1   0   0   1   2   0   0   0   3   1   14
  eh   0   0   0  20   0   1   0   1   0   0   0   0   0   0   0   0   1   0   0   0    3
  ey   0   0   0   0  12   1   0   0   1   0   1   0   0   0   1   0   0   0   0   1    9
   f   0   0   0   0   0  42   0   0   0   0   0   1   0   0   0   1   0   0   1   0    5
  ia   0   0   0   0   0   0  18   0   0   0   0   0   0   0   0   0   0   0   0   0    8
  ih   0   0   0   1   0   0   0  10   0   0   0   0   0   2   0   0   0   0   0   0   13
  iy   0   0   0   0   0   0   0   0  19   1   0   0   0   0   0   0   1   0   0   0    5
   k   0   0   0   0   0   1   0   0   0  21   0   0   0   0   0   0   0   0   0   0    4
   n   2   3   0   1   1   0   0   0   2   0  49   0   0   0   0   1   1   3   3   1   39
  ow   2   0   0   0   0   0   0   0   0   0   7   3   0   1   0   0   1   0   0   1   11
   r   0   0   0   0   0   1   0   0   0   1   0   0  23   0   0   0   0   0   0   0   27
   s   0   0   0   0   0   0   0   1   0   0   0   0   0  65   0   0   1   1   0   1    9
   t   0   0   0   0   0   0   2   0   1   1   1   0   0   0  26   2   0   1   0   6   13
  th   0   0   0   0   1   1   0   1   0   0   1   0   0   1   0  14   0   0   0   2    5
  uw   0   0   0   0   0   0   0   0   0   0   1   0   1   0   0   0   0  17   0   0    1
   v   0   0   0   0   0   0   0   0   0   0   1   5   0   0   0   0   0   0  34   2    0
   w   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0  18    0
   z   0   1   0   0   0   0   0   0   0   1   1   0   0   2   0   0   1   0   0   8   12
Ins    1   1  15   0   9   0   3   3   4   0  56   1   3   5   4   1   2   1   4   7
```

Fig. VIII.4 Typical outputs of HTK on sCUAVE. The only values used in this chapter are the accuracies — 73.85 and 42.75 here. See full description in the body of text.

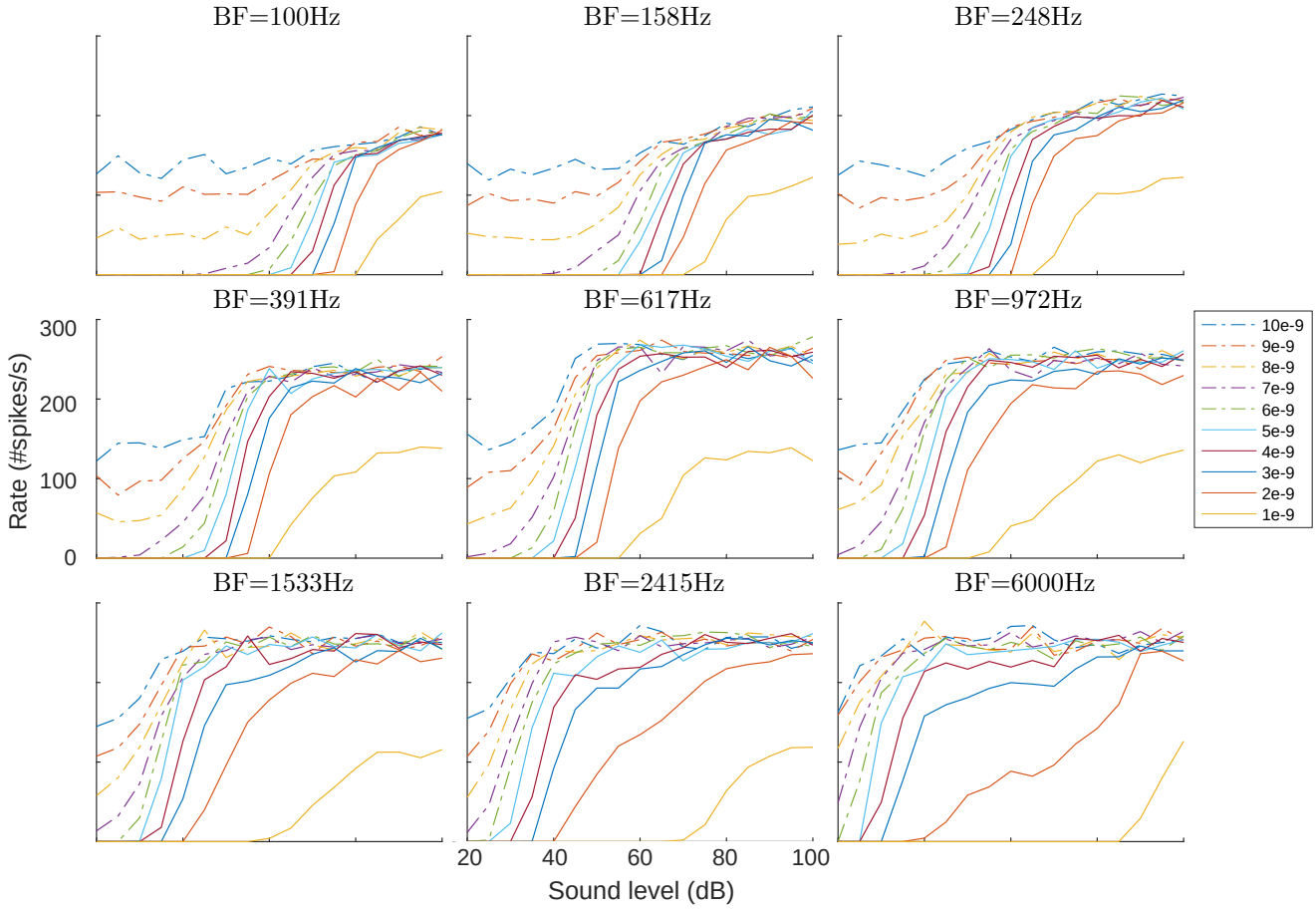


Fig. VIII.6 Intensity/Rate curves for AN fibres having different Best Frequencies (BF, given above each plot) and maximal conductance $g_{Ca}^{\max}$ (values given in legend, in nS). Each fiber is stimulated by a pure tone at its BF. The rate was obtained as the average of 20 simulated spike trains in stationary phase, the sound level varying from 20 to 100 dB SPL.

The spontaneous activity and dynamic range characterise an AN fibre. Three types are found in the guinea-pig: a 'saturating' fibre having high SR, low threshold, steep RI function and a small 20 dB dynamic range; 'sloping saturation' fibre which has less spontaneous activity, higher threshold, and does not saturate completely, but shows a sloping-saturation; and 'straight' fibres which have little or no SR, high thresholds and no steep part in their RI function, just a long slope [Sumner *et al.*, 2003]. Those behaviours are shown in figure VIII.6 by varying the maximal calcium conduction $g_{Ca}^{\max}$ in the model.

At high sound levels, the trough between formants disappear from the mean activity, as shown in figure VIII.7. In fact, as the sound level increases, many fibres attain their saturating point, then fire as much as they can. This leads to a loss of speech information in the rate, but precise timing replaces the rate coding: onsets, offsets and phase-locking to sounds leads to an increased synchrony of firing.

The classical discussion in Neuroscience of rate coding versus temporal coding is beau-

tifully represented by nerve fibres in this trade-off. Octopus neurons, for example, used as event (or synchrony) detector in [Holmberg, 2005], are innervated by enough fibres to give a precise signature of the onsets, offsets and phases.

Since we cannot expect the rate to provide all speech information at all sound levels, the cochlear model is tested using different sound levels, ranging from 50 dB to 90 dB. Furthermore, since the overall average level for a normal vocal effort in a quiet environment being 62.35 dB SPL one meter away from a speaker [ANSI, 1997], without removing pauses between words, one would expect speech processing to be optimal around 65 dB SPL.

To deal with silences in a fair way, the sound levels are enforced by renormalising only the speech part of each waveform to have an RMS of

$$\text{RMS} = 20 * 10^{dB/20} \tag{VIII.1}$$

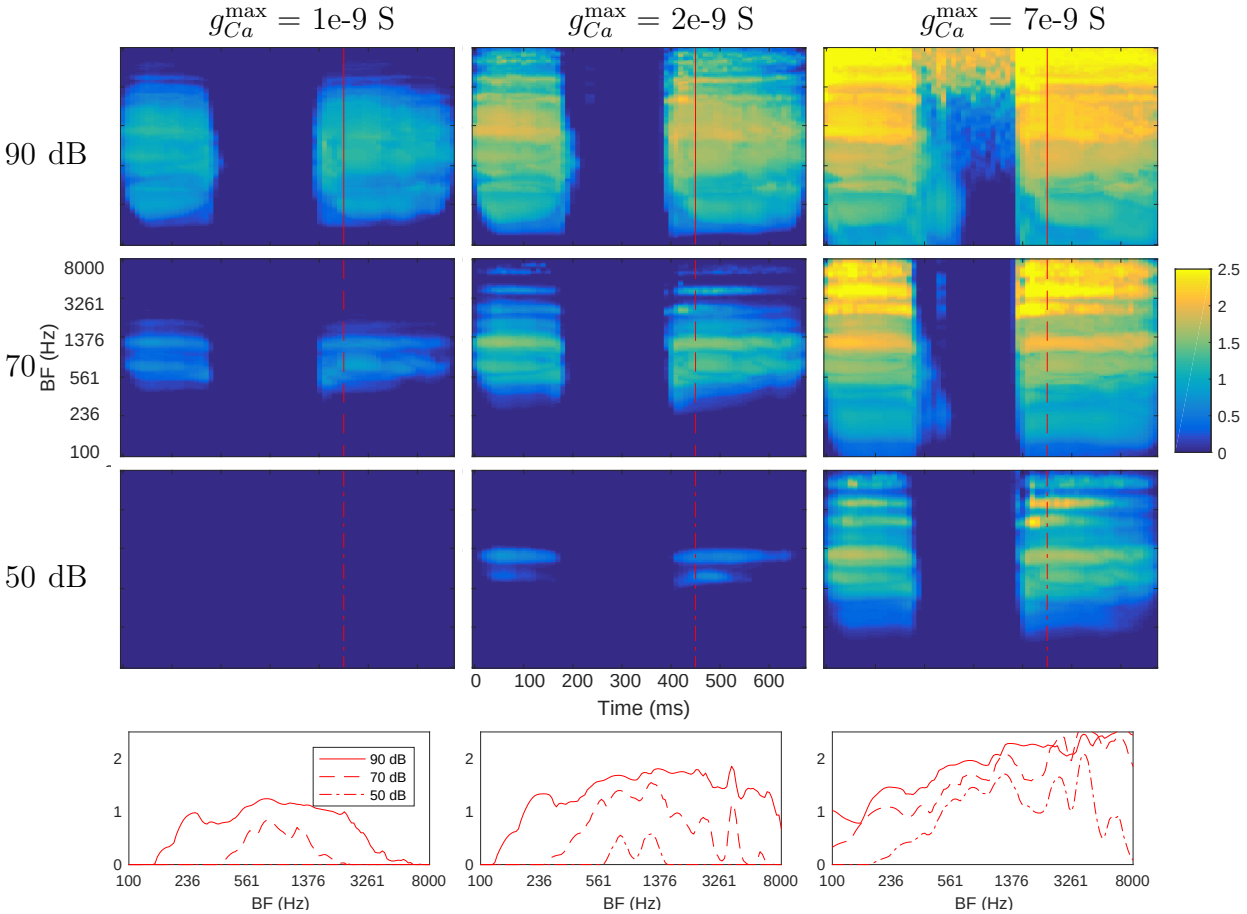


Fig. VIII.7 Cochleograms of the phone /*aba*/ said by a British male speaker, played to the cochlear model without refractoriness at three sound levels (50, 70, 90 dB) and 3 maximal AN fibres conductance (1, 2, 7 nS). The cochleograms are smoothed with a Hann window of 20 ms. The bottom plots show the neural activity at 450 ms (vertical slice from each picture).

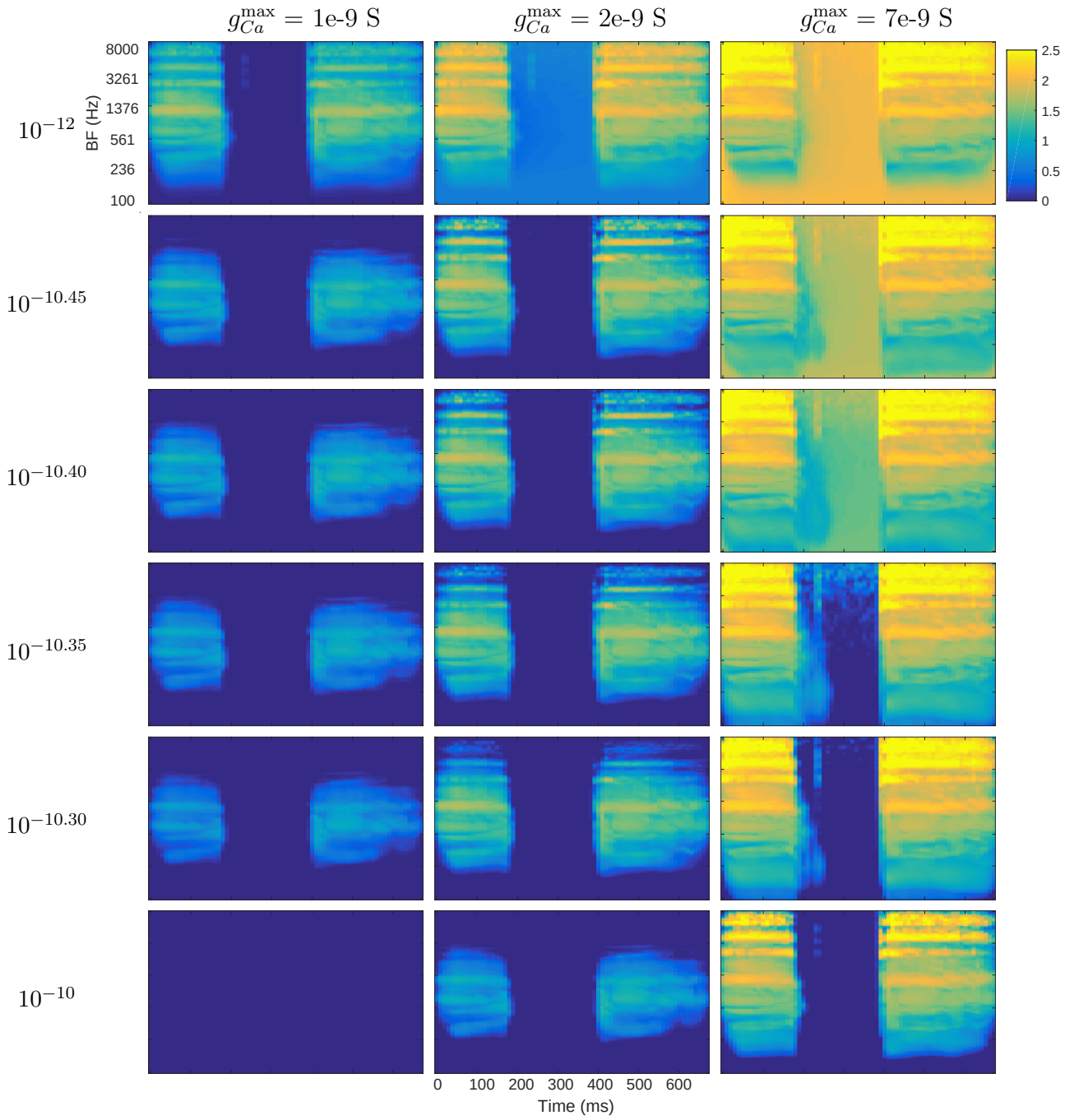


Fig. VIII.8 Cochleograms of the phone /*aba*/ said by a British male speaker, played to the cochlear model without refractoriness with different calcium thresholds between 10^{-10} and 10^{-12} and 3 maximal AN fibres conductance (1, 2, 7 nS). The cochleograms are smoothed with a Hann window of 20 ms. The sound level is fixed at 80 dB.

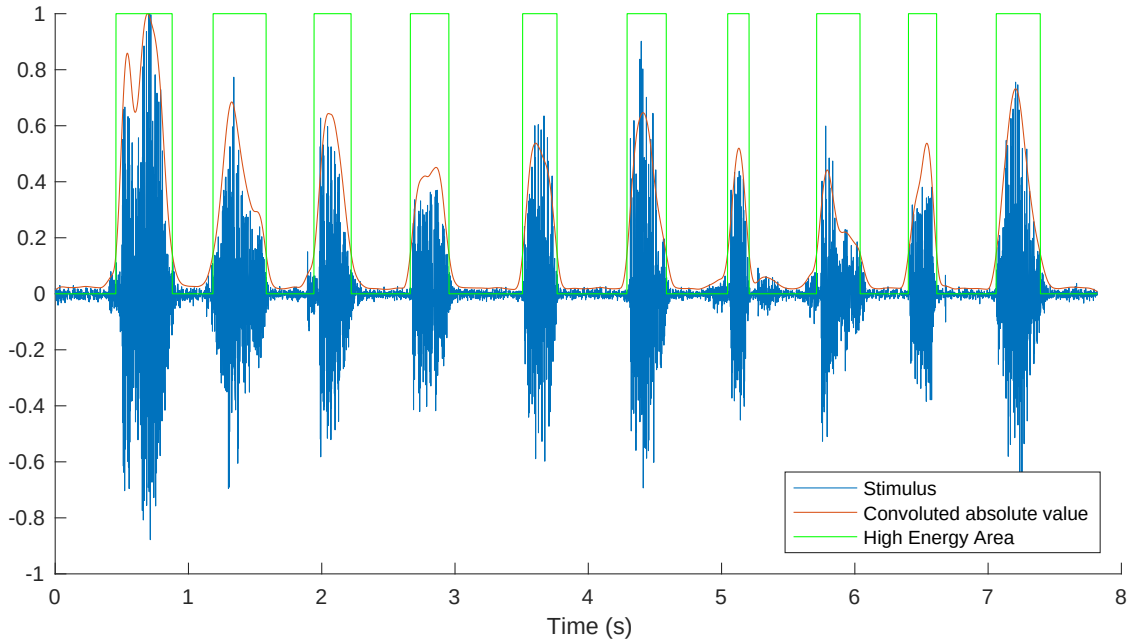


Fig. VIII.9 Extraction of the high-energy parts of a speech waveform as described in the body of text. All curves are renormalised for visualisation purposes.

where dB is the desired sound level in dB SPL. The value 20 refers to the lower threshold for hearing, set at 20 μPa. The RMS is estimated on the speech part, which is found using a simple approach: the absolute value of waveforms (resampled to 100 kHz) are convolved with a Hann window of 160 ms (and automatically re-centred thanks to the option `same` of the `conv` function), and only values beyond one seventh of the maximal value is kept. This last value (one seventh) was chosen after checking that the speech parts were appropriately found. We checked on ten different speech waveforms that the resulting sub-array did contain the speech areas, as is shown in figure VIII.9. Fortunately, this processing returns the entire numerical array when the input is a pure tone, allowing us to use the same processing when setting both speech or pure tones to a given sound level.

VIII.2.3 Back-End

VIII.2.3.1 Data Processing Chain

After obtaining either a neurogram or a cochleogram as described in VI.4.2 and VI.4.3, the pictorial representation is processed into 39 features amenable to speech recognition. This MFCC-like feature is used for the neural data, following the principles shown in figure VI.4, as opposed to chapter VII where simpler processing was used.

To obtain these 39 features, the number of channels is first reduced to 31, by averaging together the fibres four by four. Only the last feature is the average of the last eight fibres.

We then apply, as a nonlinearity, a logarithm in base 10. Values are thresholded to -5. We then add 5 to generate positive values.

Only rate-coding is used, averaging the activity in each of the 31 channels using Hann windows of varying length (from 10 ms to 1 s). Another nonlinearity (logarithm) is applied. A DCT is then used to decorrelate the channels, of which the first 13 coefficient are kept. We then add the delta and delta-delta coefficients, also renormalised, giving us a total of 39 features per 'time slice'. A cepstral normalisation finalises the processing.

VIII.3 Results

To evaluate whether a continuous digit recognition task is amenable to a physiology experiment, the simulated cochlear response to the sCUAVE dataset is tested in an ASR framework. The important model parameters introduced earlier are tested, to finally evaluate the accuracy obtainable under chosen circumstances for a varying number of ANFs.

VIII.3.1 Effect of Conductance on ASR

LSR fibres are often thought of as encoding more envelope [Joris and Yin, 1992] and, hence, speech information thanks to their wider dynamic range [Sumner *et al.*, 2003; Holmberg *et al.*, 2007]. Indeed, as seen in figure VIII.7, the maximal conductance in the fibre model is of paramount importance in the resulting spiking activity. However, the results presented in figure VIII.10, where the effect of varying the conductance for the digit recognition task on the sCUAVE dataset is tested, do not support this position.

As $g_{Ca}^{\max}$ increases from 2 nS to 9 nS, the overall recognition score increases, the maximal accuracy (across all sound levels) increases. Shorter Hann windows also give better results, as conductance increases. Still, as conductance increases, the optimal score is obtained for a decreasing sound level (90 dB to 70 dB, see black circles in figure VIII.10). Interestingly, recognition reaches an overall maximum for high levels of $g_{Ca}^{\max}$ at 70 dB SPL. Thus, it is true that LSR fibres encode high sound levels better than low sound levels, and HSR fibres encode lower (or medium) sound levels better than high sound levels, consistent with previous observations. However, overall recognition scores here are essentially better in HSR fibres than in LSR model fibres, at all signal levels.

At low sound levels, and low values of $g_{Ca}^{\max}$, performance is very poor for small Hann windows: longer windows increase performance. Whereas when sound levels are moderate or higher, and conductance is high, performance is less dependent on Hann window length, but if anything is slightly optimal for short Hann window lengths. The good recognition accuracy at long Hann windows suggests that High Spontaneous Rate (HSR) fibres encode more speech information than Low Spontaneous Rate (LSR) fibres, when forced to use a pure rate code.

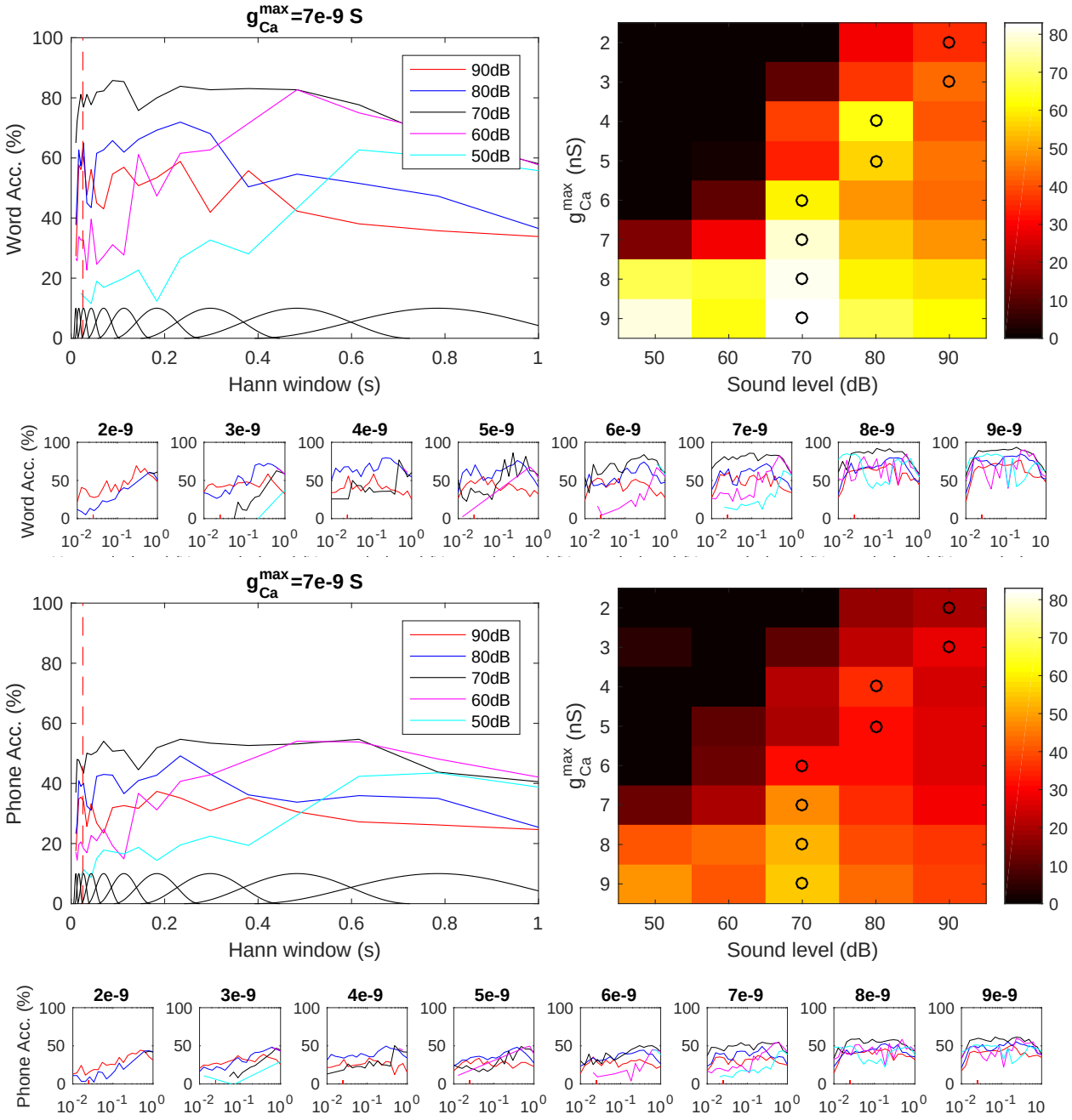


Fig. VIII.10 Word and phone accuracies using the MFCC-like feature on the sCUAVE dataset, on neurograms obtained with 50 fibres per BF, varying the length of the Hann window (x-axis). On top left plot, a linear scale is used for the x-axis and half of the Hann windows are displayed, centred at their duration. On bottom plots, a log-scale is used. At 0.025 s, a red mark shows the usual duration of the Hann window in ASR applications. Missing values are due to HTK not finishing calculations due to poor data. Top right plot shows the mean accuracy over the first 10 Hann durations for each sound level and maximum conductance. A circle shows the optimal sound level for each maximal conductance.

However, performance is often slightly better for shorter Hann windows (below 100 ms). suggesting that this additional information is available in the form of high frequency fluctuations in firing rate, either in the temporal fine structure or in high frequency envelope information. Since the human cochlea appears to have narrower band pass filters than that of other mammals [Shera *et al.*, 2002], one would expect — assuming there was not some other gain compensation for the reduced energy into the filters — higher recognition rates at high sound intensities if the parameters had been fitted to human physiological data, as the rate-place coding would increase in precision, while reducing the saturation of ANFs across a wide range of frequencies.

It also suggests that different types of neurons may encode information at different sound levels. At high sound levels, LSR neurons would provide an onset and offset response more distinguishable than the HSR, while at lower sound levels, the lower threshold of HSR fibres enable them to code more low sound level features of speech information than LSR. Indeed, as seen in figure VIII.7, features of the formants were missing in the cochleograms of LSR fibres since the sound intensity did not reach the higher threshold of LSR fibres. The lower threshold of HSR fibres, seen in rate-intensity functions of figure VIII.6, allows this potentially important information to appear at low to normal sound levels (70 dB SPL). The speech envelope would be expected to be better encoded by the LSR fibres as long as the features are present in the response, assuming their threshold of activation was reached.

Saturation of HSR fibres could be a limitation, as they have a narrower dynamic range, but this does not seem to be a limiting factor in the speech encoding ability of ANFs at a normal sound level. Even at high sound levels, both HSR and LSR low-BF fibres can still phase-lock to the fine structure of the incoming sound, which contains more speech-related information than high-BF fibres. Synchrony-based features would be necessary to extract more phase-related information at high sound levels, when the fibres saturate. This behaviour is indeed observed in [Holmberg *et al.*, 2007]: the recognition accuracy decreases much more at high sound levels when only using a rate-code, compared to using both a rate-code and a temporal code. For this result, they used both HSR and LSR fibres. Our result suggests that removing the LSR fibres from their model would lead to a further decrease in recognition accuracy at high sound levels.

To summarise, this discussion sheds light on the issue of thresholding versus saturation. This result suggests that, at normal sound levels, the lower threshold of HSR fibres is predominant in their higher ability to encode speech information using a rate code. A second but probably less important factor appears to be that at medium and high sound levels HSR fibres are able to make use of finer spike timing of features.

VIII.3.2 Effect of Calcium Thresholding on ASR

Up until now, the threshold concentration of calcium required for neurotransmitter release was fixed in the model to the default value $[Ca^{2+}]_{thr}$ = 4.48e-11, unit free. The conductance $[Ca^{2+}]_{thr}$ primarily affects the low intensity responses, thus the spontaneous rate and threshold of the unit. Since this default value is fairly high compared to the value fitted to real HSR neurons [Sumner *et al.*, 2003], the effect of this parameter on the speech recognition accuracy is investigated.

Figure VIII.8 shows that larger thresholds reduce the representation of the sound features in the cochleogram, particularly for smaller calcium maximal conductances. Having observed this dynamic, a maximal conductance of 2 nS is set, as well as a sound level of 80 dB. This choice is bound to limit the accuracy achievable, but is meant to avoid extreme values of spontaneous activity that would not be physiologically related to the thresholds tested, according to the link between those two parameters when fitted to experimental data [Sumner *et al.*, 2003].

As seen in figure VIII.10, the value $[Ca^{2+}]_{thr}$ = 4.48e-11 does not allow high accuracy. In figure VIII.11, results are shown on the left plot. For a fixed sound level of 80 dB and a fixed g_{Ca}^{max} of 2 nS, $[Ca^{2+}]_{thr}$ is varied between 1e-14 and 1e-10. The dashed blue curve is the same as the blue curve in the left plot of figure VIII.10.

In this setting, the accuracy tends to increase as $[Ca^{2+}]_{thr}$ decreases. Below 1e-11, the change is on average fairly negligible. This is shown in the right plot, where the averaged values for 3 Hann windows duration (the three closest to 40ms) is given for each value of $[Ca^{2+}]_{thr}$ that gave values (5e-11 or below; above, HTK finished on an error due to 'bad data'). This result is consistent with the idea that HSR neurons encode more speech information.

For small calcium thresholds ($\leq$ 3e-11), a trend appears that both small and large Hann windows perform better than medium ones (around 80 ms). On a larger scale, another trend is that the slope gets steeper with increasing threshold, as the values for large Hann windows are relatively constant for all values of $[Ca^{2+}]_{thr}$, but this uniformity is lost for small Hann windows. This hints at the usefulness of temporal fine structures being encoded at low calcium thresholds in the neural responses but slowly lost as this threshold increases, while a rate code for high temporal windows still represents the formants in a satisfying way.

VIII.3.3 Number of Fibres

Rather than reducing the value of $[Ca^{2+}]_{thr}$ to improve the accuracy, the default value is used once more. This allows for a better comparison of results by reducing the number of varying parameters.

As more spike trains drawn according to the same distribution are averaged, the PSTH, as a stochastic process, ought to converge to a distribution related to the probability of firing (see appendix 12). In figure VIII.12, it is shown that this property leads, by continuity, to

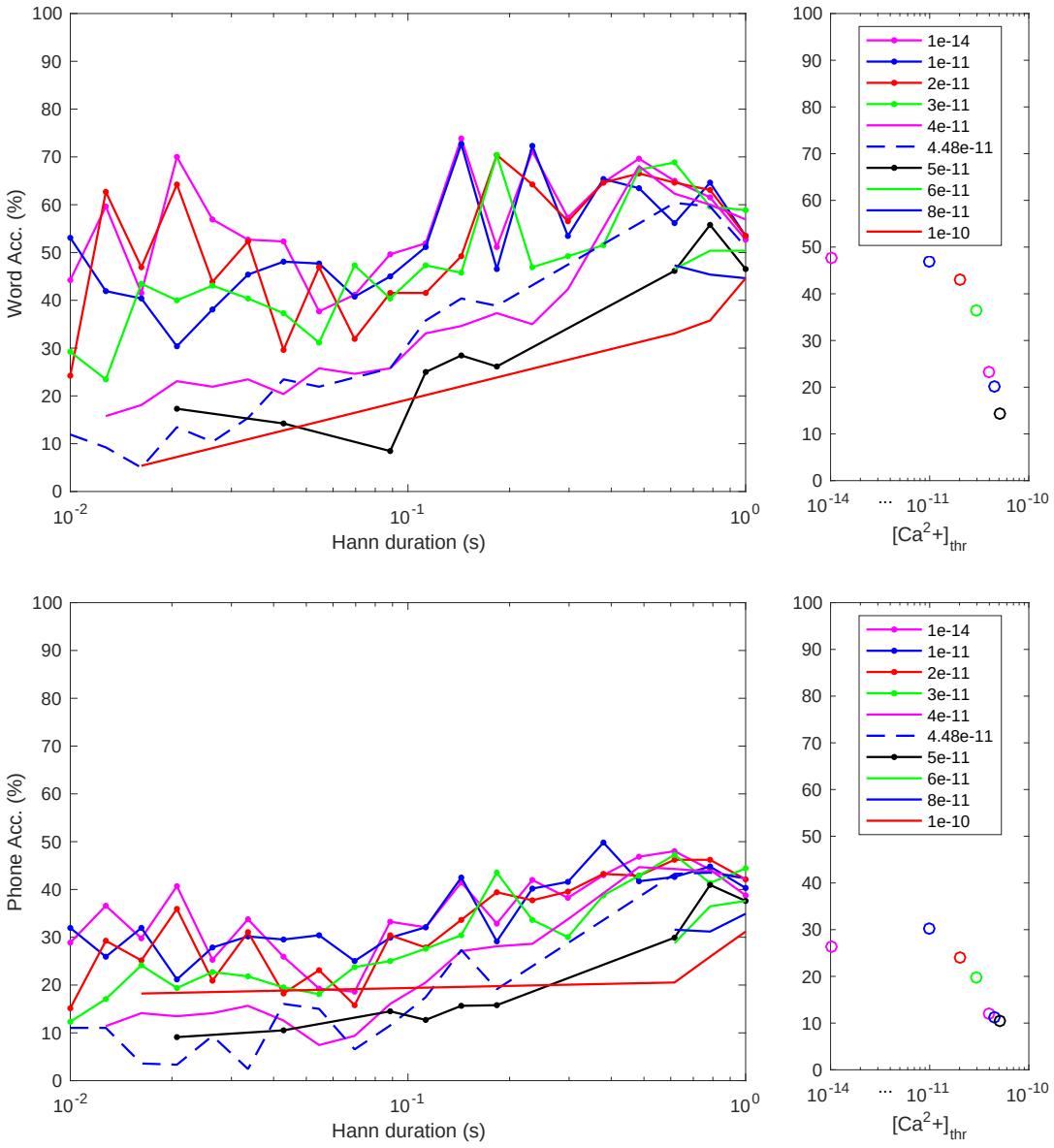


Fig. VIII.11 Word and phone accuracy using the MFCC-like feature on the CUAVE dataset, varying the length of the Hann window (x-axis), using 50 simulated fibres per BF. Sounds played at 80 dB to the cochlear model where $g_{Ca}^{\max}$ is fixed at 2 nS. The value of $[\mathrm{Ca}^{2+}]_{\mathrm{thr}}$ is varied (colours given in the legend). On the right plot, the mean accuracies obtained by averaging the accuracies corresponding to the 3 Hann windows closest to 40 ms (33.6, 42.8, and 54.6 ms) are given where a value was provided by HTK, to show the trends at this temporal scale.

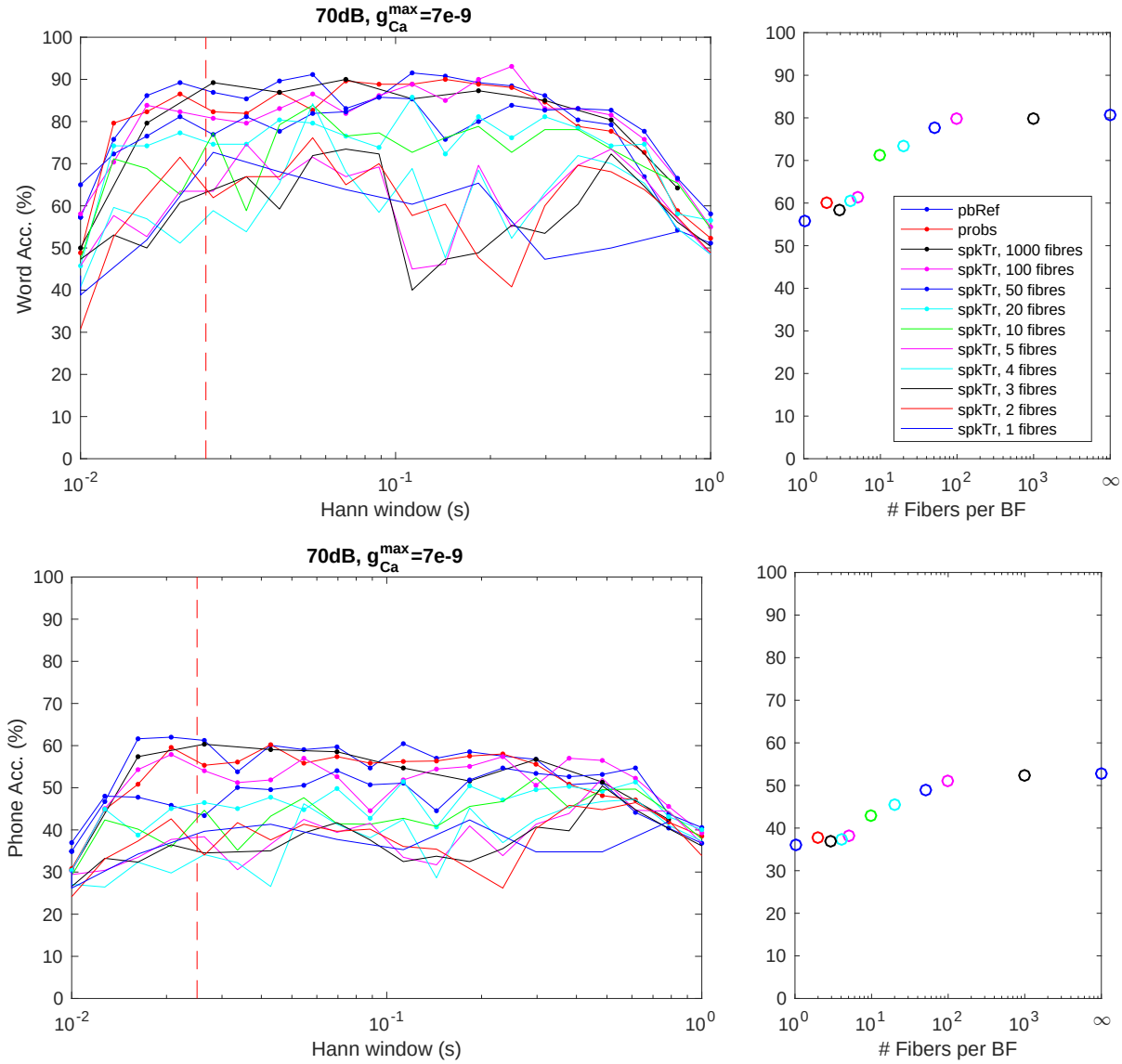


Fig. VIII.12 Accuracy using the MFCC-like feature on the sCUAVE dataset, on cochleograms - with (pbRef) or without (probs) refractoriness - or on neurograms obtained with certain number of fibres per BF, varying the length of the Hann window (x-axis). Sound level is fixed at 70 dB and $g_{Ca}^{\max}$ at 7 nS. The vertical red mark shows the 0.025 s value x on the x-axis, typical for ASR. On the right plot, the mean accuracy is given (y-axis) for each number of fibres per BF used (x-axis) the probability of firing with refractoriness, which is equivalent to an infinite number of fibres, as detailed in appendix 12.2.2.

a convergence of the recognition accuracy when the number of fibre per Best Frequency increases from 1 (a total of 128 fibres) to 200 (total of 200*128=25600 fibres, close to the human's 30000 fibres). In particular, on the right plot, it is clear that the mean value increases almost monotonically to the mean accuracy using the probability of firing including the refractoriness.

Due to the stochasticity of the spike generation and the nonlinear data processing in both front-end and back-end, this increase is not strictly respected for any fixed Hann window length. A cross-validation validation protocol might be robust enough to provide strictly increasing results.

Once more, the usual Hann window length (25 ms) is suboptimal, but by a small margin only.

VIII.4 Conclusion

The ASR framework developed allowed an accuracy of ∼50% at the phone-level and ∼80% at the word-level, on a number of fibres that fit the requirements for an electrophysiology experiment. As such, this simulated experiment can be considered a success. All conclusions obtained in this chapter are qualitatively the same at the phoneme-level and at the word-level, the latter implying a steady improvement in accuracy of about 30%. This improvement is simply due to the simplicity of the dictionary, thus helping the recognition by reducing the amount of correct guesses necessary to recognise a word, and reducing the amount of instances to recognise in each sequence of words.

In performing the analysis required to reach this conclusion, much was learnt regarding the impact of various important parameters from the cochlear model, that can henceforth be set:

- High Spontaneous Rate fibres encode more speech-related information than Low Spontaneous Rate ones in quiet and at normal sound levels. This result is counter-intuitive at first sight due to the narrower dynamic range of HSR fibres. However, their lower threshold of activation makes them able to encode information when LSR are still dormant. Furthermore, since the energy at any given frequency varies quickly in speech, formants can still be grossly described by the activation/inactivation patterns of the HSR fibres, in the absence of background noise;
- At medium and high sound levels, HSR fibres are able to make use of finer spike timing of features. LSR fibres are firing, but require longer windows of integration to encode less information than HSR, using a rate-code;
- Assuming a rate-code with small integration window, the optimal sound level decreases with an increase of the spontaneous activity, going from 90 dB SPL for fibres having a $g_{Ca}^{\max}$ of 2 nS, to 70 dB SPL for fibres having a $g_{Ca}^{\max}$ of 9 nS;

- A satisfying accuracy on this digit recognition task may be achievable with 40 minutes worth of recording. This result on simulated data is a strong indication that similar analyses on electrophysiological data should lead to meaningful results.

Our result suggests that HSR fibres encode more speech-related information in a quiet environment. On the other hand, noise-induced cochlear neuropathy was shown to be associated with a loss of LSR fibres [Furman *et al.*, 2013]. This type of hidden hearing loss, not revealed by audiograms, is hypothesised to mostly impact speech intelligibility in noisy environments. Hence, it is possible that the neural coding of speech in the auditory nerve evolves from being encoded by HSR fibres to a code based on LSR fibres that complement the HSR fibres more intense firing as background noise increases.

Now that the analysis of the optimisation of the cochlear's model's parameter on a continuous speech task is done and yields acceptable results, the complexity of the task is increased once more in the next chapter to challenge the ASR framework on a large-vocabulary.

IX.

Neural ASR on a Large-Sized Vocabulary

In this chapter, we consider a framework for a continuous speech recognition task on a large dictionary with many speakers. The same type of processing as in the previous chapter is applied, fixing the front-end parameters to their optimal values learnt from previous chapters, and testing more back-end processing, as shown in figure IX.1.

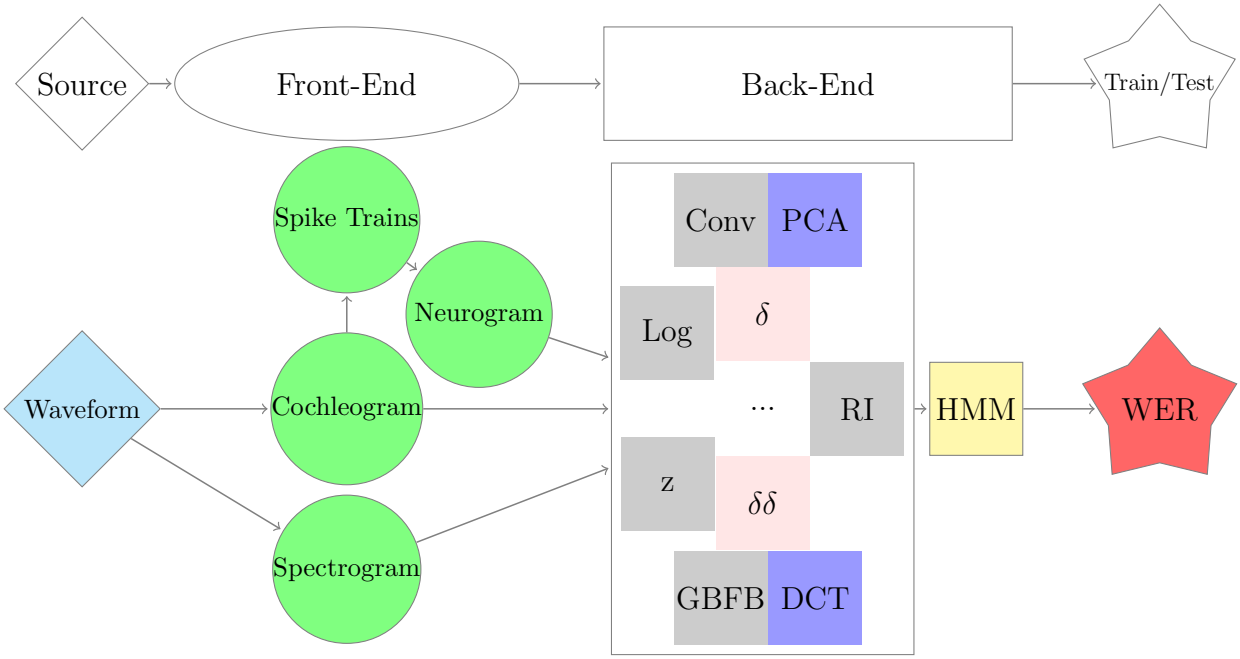


Fig. IX.1 Graph of data processing presented in this chapter. Front-ends are defined in previous chapters; their parameters are now entirely set. Back-end consists of many blocks defined within the methodology section; their number and order is changed.

IX.1 Introduction

In this chapter, we apply the neural speech recognition computational framework presented in previous chapters to a well-known speech dataset, TIMIT, to test the robustness of the HMM models on a fully-fledged continuous speech setting, based on a large dictionary and containing speech from many speakers. Since the full TIMIT dataset does not satisfy the time constraints for an electrophysiological experiment presented in section VI.4.4, a subset of this dataset is used, named sTIMIT, that lasts for an hour.

The front-end parameters are set using the parameters that gave the best results in chapter VIII. Many back-end processing are evaluated and compared, in order to see whether an acceptable recognition accuracy (set to 50%) is achievable on such a complex dataset.

IX.2 Methodology

IX.2.1 sTIMIT Dataset

The TIMIT dataset [Garofolo *et al.*, 1993] is widely used as a testbed in ASR tasks with biophysical models, as shown in table VI.9. It is fully annotated and has been used by the Speech Recognition communities for over 20 years. However, the setting of a physiology experiment is not constant in time, as the animal's physiological state changes, the electrode array may drift due to microscopic vibrations, or the state of the neurons surrounding the electrode may simply change due to the contact with the electrodes or brain damage. As such, due to its length (over five hours), this dataset is not fit for a physiological experiment. A subset of the TIMIT dataset was thus selected, called sTIMIT. This subsection describes the sTIMIT dataset; its main features are given in table IX.3. All information presented here about TIMIT either comes from direct evaluations on the dataset, or is given by the TIMIT documentation.

To be able to play the full dataset a few times to each animal, sTIMIT lasts for an hour, which should normally allow at least 3 sets of recording. About three quarters of the sentences are from the TRAIN dataset, the other quarter from the TEST set.

To reduce the pitch variability without reducing too much the number of speakers, only male speakers were kept, as they represent 70% of the complete dataset (438 out of 630 speakers). The sentences are regrouped within three categories in the corpus:

- Dialect (SA), meant to expose the dialectal variants of the speakers (only 2 sentences, read by all speakers);
- Compact (SX), designed to provide a good coverage of pairs of phones, with extra occurrences of phonetic contexts thought to be either difficult or of particular interest.

Dialect	Difficulty	Duration (males, SX)
New England	6	435 s
Northern	7	1046 s
North Midland	3	1105 s
South Midland	4	1037 s
Southern	4	915 s
NY City	4	438 s
Western	3	1056 s
Army Brat	5	299 s

Tab. IX.2 Difficulty attributed to each of the 8 dialects constituting the TIMIT dataset. The marks were between 1 (easy to understand) and 10 (very hard to understand), after listening 5 instances of each dialect by a single listener whose native language is French. The total duration of each dialect for the male speakers for SX-typed sentences is given on the right column.

Each of the 450 sentences in this set is read by 7 different speakers, and each speaker read 5 sentences;

- Diverse (SI), selected from existing text sources so as to add diversity in sentence types and phonetic contexts. Each of the 1890 sentences is read by only one speaker.

Only sentences of SX type are kept, the SA-type sentences being normally left aside [Lopes and Perdigao, 2011] because they introduce an unfair bias for certain phones [Lee and Hon, 1989].

To minimise the number of different dialects and select them, a level of difficulty was manually attributed to each dialect, by randomly listening to 5 sentences in each set and giving it a grade between 1 (very easy to understand) and 10 (very hard). The results are given in table IX.2.

All available sentences were then taken, alphabetically ordered in their folder, until the dataset would last for one hour. The sTIMIT dataset characteristics are summarised in table IX.3. The number of classes is reduced from 61 to 39, as proposed by [Lee and Hon, 1989; Lopes and Perdigao, 2011]. For simplicity, the number of classes is kept the same for both training and testing.

IX.2.2 Front-End

As shown in figure VIII.1, three types of data representation are used as front-end for this ASR task, with their code given in parenthesis:

• Spectrogram (**wav**): 31 channels encode a logarithmically scaled Mel-spectrogram between 64 and 8000 Hz;

Type	Options	Number of Sentences
Gender:	Males	1252
	Females	0
Sentence:	SA	0
	SX	1252
	SI	0
Dialect:	New England	0
	Northern	0
	North Midland	395
	South Midland	345
	Southern	142
	NY City	0
	Western	370
	Army Brat	0
Set:	Train	943
	Test	309

Tab. IX.3 Summary the sTIMIT dataset, given by total number of sentences for each possible category in the TIMIT dataset.

• Cochleogram (**probs**): 128 channels whose BFs are logarithmically spread between 100 and 8000 Hz, sounds played at 70 dB, g_{Ca}^{max} fixed at 8e-9 S and $[\,\mathrm{Ca}]_{\mathrm{thr}}$ at 0 to simulate High Spontaneous Fibres [Sumner *et al.*, 2003], Hann duration of 0.11 s, without considering the effect of refractoriness;

• Neurogram (**SpkTr**): up to 1280 spike trains are used, with 10 fibres per BF from cochleograms, to simulate 10 hours of recording using a one-hour long dataset. The description of the spike trains simulation, including the refractoriness, is given in appendix 12.

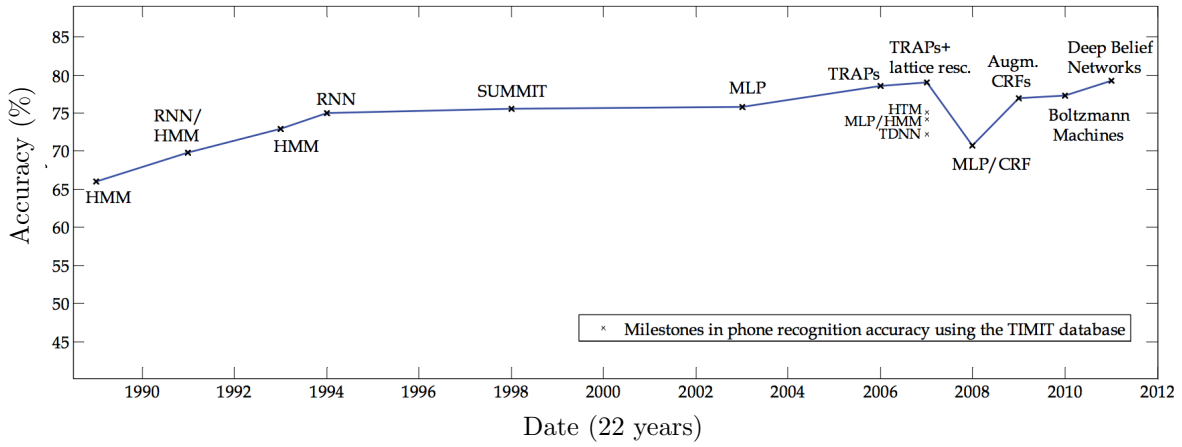


Fig. IX.4 Milestones of TIMIT phone recognition performance [Lopes and Perdigao, 2011]

All pictorial representations, as output by the front-end, have a sampling rate of 10 Hz as is classical for HMM-based modelling.

IX.2.3 Back-End

As in the previous chapter, the first back-end block processing reduces the dimensionality of the feature vector. This is achieved by regrouping spike trains (or channels). Various processing (given below) can then be applied. The different back-end processings used are each given a short name (in bold below), separated by underscores when used in a sequential order. This notation helps in easily comparing the effect of changing the sequence used in the results' section; For example, the processing chain used in chapter VIII on spike trains was spkTr_f_lu_r_l_d_dd_z.

Spike grouping:
Due to refractoriness and the dynamic range HSR units cover, rate coding per spike train has limitations that we get around by gathering spike trains together. This channel grouping is done in two ways:

- **f:** Prior knowledge of the best frequency of each channel is assumed, and channels are ordered by increasing BF. Neighbouring channels are then averaged together to obtain feature vectors of length 31. The 30 first features average the same number of channels; the last feature averages the remaining channels;
- **cA:** Channels are correlated with the frequency channels of the speech waveforms spectrograms. Spike train channels maximally correlating to the same frequency (among the 31 frequency channels of the spectrogram) are then averaged together.

Processing:

- **lu:** A logarithm in base 10 is applied, all values smaller than -5 are then thresholded to -5, and all values are finally increased by 5 to be positive or null. This non-linearity neglects the small values;
- **r:** A sliding Hann window is applied every 10 ms. Unless stated otherwise, in this chapter the Hann window duration is 0.11 s;
- **l:** Logarithm ($x \mapsto \log(1+x)$);
- **d:** DCT, of which we keep the first 13 coefficients;
- **dd:** Speed and acceleration are appended (respectively called δ and $\delta\delta$ coefficients);
- **z:** Cepstral normalisation per channel;

- **e:** Append squared energy per feature vector;
- **g:** Spectro-temporal Gabor filter bank features (GBFB), as implemented in [Schaedler and Kollmeier, 2015]
- **s:** Separable Gabor filter bank (SGBFB), as implemented in [Schaedler and Kollmeier, 2015];
- **lsp:** Logarithm applied to a spectrogram with log-space frequency output;
- **ISI:** Applied to a sparse array, this processing is constant between two consecutive spikes, its value being the time interval between those two spikes (InterSpike Interval).
- **ri:** This feature outputs the element-wise inverse of the 'ISI' feature. This naive estimation of the instantaneous rate by ISI inversion was used in chapter VII.

Dimensionality Reduction:
As a dimensionality reduction method, only the DCT and PCA are used. Despite the smaller accuracy is may bring compared to PCA (see figure VII.3), the significantly smaller computational cost of the DCT allows one to test and compare more processing.

IX.2.4 HTK Parameters

Transcript & Dictionary Even though the TIMIT dataset is provided with both word- and phoneme-level transcript, we only make use of the word-level transcript. A phone-level transcript is made at two stages of the processing: before any HMM training, using the BEEP[1] dictionary which contains the phonemic transcription of over 250,000 English words, and the function HLEd that selects the first pronunciation of each word present in the dictionary.

Mixtures & Training At the end of the training of a single mixture (8 passes with HERest), another phone-level transcript is created, this time using the function HVite to align the optimal pronunciation of each word with the word-level transcript, using the single mixtures HMMs trained. Another 6 passes of HERest training is made with this transcript

```
Word-Label:  DECEMBER AND JANUARY ARE    NICE MONTHS TO   SPEND IN MIAMI
Recognised:  DECEMBER     INTO    NEARER NICE        MUST STINGING MIAMI
Phone-Label: d ih s eh m b er ae n d jh ae n y uw eh r iy aa r  n ay s m ah [...]
Recognised:    uw s uw m   er    n   jh eh n   uw         w  er m ay s m ah [...]
```

Tab. IX.5 Example word-level and phoneme-level sentences recognised on the optimal value (phoneme-wise) of figure IX.7.

[1] http://svr-www.eng.cam.ac.uk/comp.speech/Section1/Lexical/beep.html

and single mixtures HMMs. The number of mixtures is then gradually increased, with 6 passes of HERest training after each increment. The timing information in the transcripts is never used.

Words & Phonemes The TIMIT database is a classical dataset for phone recognition, as the timeline in figure IX.4 shows, on which accuracies of almost 80% have been obtained [Lopes and Perdigao, 2011]. This motivates the use of sTIMIT on a phoneme recognition task. For comparison with the previous chapter, word-level accuracies are also provided, even though their scientific value is of limited value as the example given in table IX.5 shows: Too many context-dependent components play a role in the final result.

Phonemes & Triphones Considering the amount of words in a language (500k to a million in English), training a separate HMM for each word would be highly impractical. Instead, we keep on using phonemes, as about 40 phonemes are needed to represent all English words. This allows the system to recognise words never seen during the training stage, and to increase the number of words that can be recognised simply by extending the pronunciation dictionary.

Due to coarticulation, phoneme boundaries are often unclear. With HMMs, this issue is usually tackled using context-dependent models named triphones instead of phonemes, whereby each phone has a distinct HMM for any pair of left and right contexts. An example is given in table IX.6. The use of triphones allows significative phone recognition improvements [Lopes and Perdigao, 2011]. However, this modelling comes at a greater computational cost and a higher complexity. Since our goal in this work is not to optimise the accuracy at all costs but to compare the processing methods, we keep using phonemes-level HMMs as in the previous chapters.

Modelling Level	Transcript
Words	`sil this sp man sp...`
Phonemes	`sil th ih s sp m ae n sp ...`
Triphones (Cross-Word)	`sil th+ih th-ih+s ih-s+m s-m+ae m-ae+n ae-n+...`
Triphones (Word-Internal)	`sil th+ih th-ih+s ih-s sp m+ae m-ae+n ae-n sp ...`

Tab. IX.6 Modelling 'This man...' with word-, phones- and triphones-level HMM, using the usual HMMs 'sil' for long silences and 'sp' for short silences as word boundaries. Word-internal triphones facilitate decoding but cross-word triphones convey contexts through silences and are thus more expressive.

IX.3 Results

IX.3.1 Processing chains

Since word-level accuracies are consistently 10 to 15% below the phone-level accuracy, only the phone-level accuracies are discussed.

After trying only a few back-end processing choices in chapter VII and fixing it to a presumably good one in chapter VIII, the backend processing is now more thoroughly altered: processing is tried in diverse orders, removed or added. The results are shown in figure IX.7 as ranked by their phone-level accuracy. The three types of pictorial representations are used:

- Spectrograms refer to the short Fourier-transformed waveforms;
- Cochleograms contain the probabilities of firing per BF;
- Neurograms use either one or 10 fibres per BF (ending with sp1 or sp10).

By a margin, the best score is obtained using the spectrograms, where the use of PCA allows one to improve on the DCT if the dimension is large enough (50 instead of 20). As opposed to the conclusion in [Schaedler and Kollmeier, 2015], the separable Gabor filterbank ('s' processing) gives a lower accuracy than the simple GBFB ('g' processing).

Finishing the processing chain by a cepstral normalisation ('z') does not seem to have a positive influence, as it reduced the accuracy when added to the processing chain probs-f-l-d-dd. On the other hand, the δ and $\delta - \delta$ coefficients ('dd') following the DCT make a big difference of about 10%.

The Gabor filterbank ('g') is not adapted for use with DCT: since this filterbank generates many channels that are highly correlated, the channels must be decorrelated, either manually or using a transform, which is what PCA does, but not DCT. An alternative would be to manually extract the theoretically least correlated channels output by the GBFB filterbank [Schaedler and Kollmeier, 2015]. This was confirmed by various attempts that contain the sequence 'g-d'.

The use of PCA removes the improvement that the separable version of this filterbank ('s' compared to 'g') is meant to bring, as shows the $\sim$ 3% margin when applied on the spectrograms.

Surprisingly, using 10 fibres per BF was enough to obtain a score higher than using the probabilities (spkTr-f-lu-r-l-d-dd-z-sp10 / probs-f-l-d-dd-z), which in turn was better than with one spike train per BF (spkTr-f-lu-r-l-d-dd-z-sp1).

Replacing the knowledge of the BF with a mapping to the channels of the spectrograms (spkTr-cA-lu-r-l-d-dd-z-sp1) leads to a loss of 10% of accuracy. Other ways to aggregate spike trains without assuming the knowledge of the BFs could nonetheless reduce this gap.

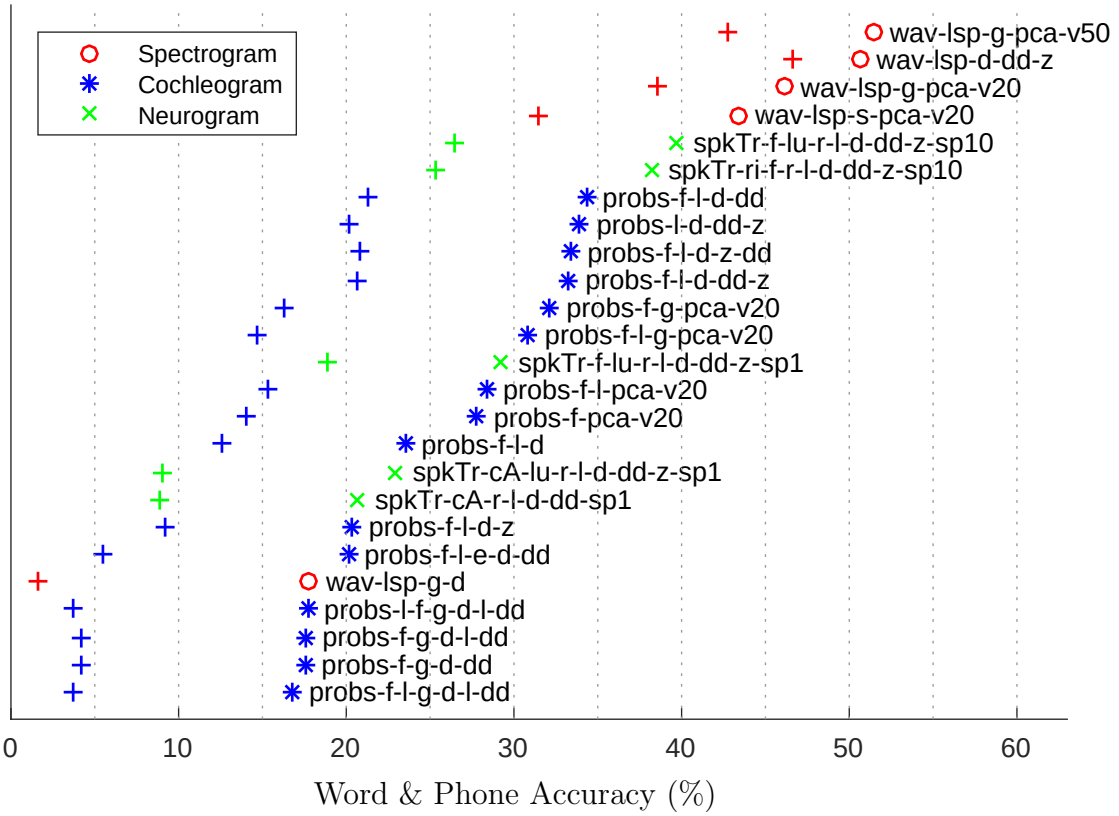


Fig. IX.7 Word (+ mark) and phone ($*, \times, \circ$ marks) accuracies over many processing. Range of back-end processing tested on the sTIMIT dataset, using waveforms (red circle), probabilities of firing (blue stars) or spike trains (green crosses) as output of the front-end block. Appended to the processing are two complementary parameters: for spike trains, the number of spike trains used per BF (either 1 or 10); for PCA, the vector dimension output by the PCA. All Hann windows used (in the 'r' processing, implicitly contained in 'probs') have a duration of 0.11 s.

Appending the energy ('e') before applying the DCT surprisingly drastically reduced the score (probs-f-l-e-d-dd, 19%).

Even the optimal score (51%) does not reach the accuracy obtained on the TIMIT dataset in the 1980s (66%, see figure IX.4). This difference could be reduced by many ways: increasing the size of the training set, increasing the number of Gaussian mixtures, or using triphones. Again, since our goal was not to optimise the overall accuracy but to compare the processing, a score of about 50% satisfies our needs.

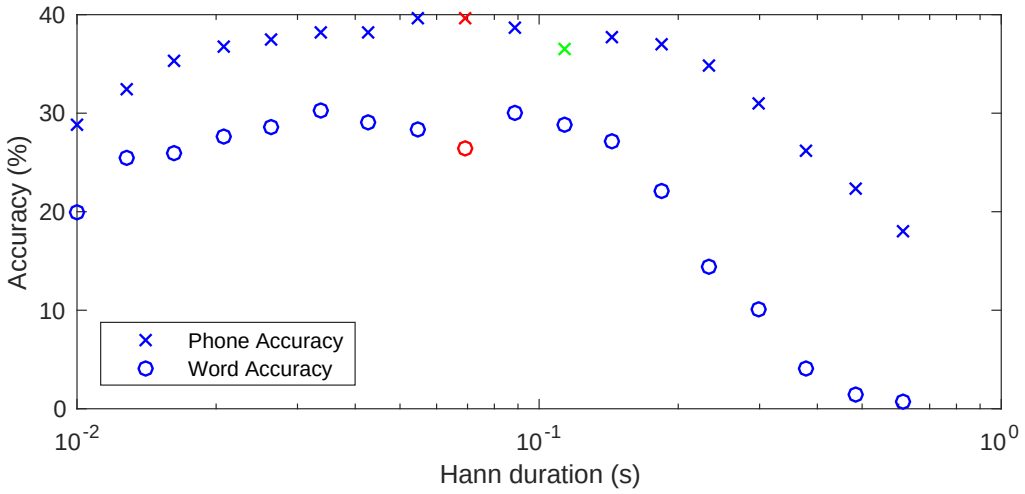


Fig. IX.8 Effect of varying the Hann window using the same processing chain as in previous chapters (spkTr_f_lu_r_l_d_dd_z), using 10 spike trains per channel. Both word-level and phoneme-level accuracies are given. The red markers highlight the result used in table IX.5; the green cross is the value appearing in figure IX.7

IX.3.2 Optimal Time Scale

As in the previous chapters, the optimal Hann window duration is evaluated, having fixed the back-end processing. Its optimal value for the phone-level accuracy, about 70ms, surprisingly corresponds to a local minimum for the word-level accuracy as shown in figure IX.8. While the word-level accuracy smoothly decays to 0 for very large Hann windows, the phone-level accuracy remains above 20%. For Hann windows larger than 0.6 s, HTK could not finish its training as was already observed in previous chapters where the data's quality would become too poor.

IX.4 Conclusion

This chapter concludes the research done on datasets of different sizes and complexity. The results we obtained on sTIMIT using spectrograms are significantly lower than the maximal accuracies found in the literature. However, one could expect that developing better front-end and back-end processing would lead to a satisfying improvement using spike train data.

The results obtained using simulated neural data are even lower than those obtained with spectrograms, and they are under the 50% phoneme-level threshold that had been arbitrarily decided on simulated neural data. Hence, we would not use this dataset for an electrophysiological experiment with the current computational framework but rather use the continuous digit recognition task.

Accuracies at the word-level were consistently smaller than those at the phoneme-level.

Beside the difficulty of recognising a word among a large amount of possible choices, this phenomenon is due to the word insertion penalty. This parameter, optimised by testing the accuracy on a small development set, has an impact on the ease with which HTK can insert more words in a recognised sentence. By studying the recognised sentences, short words such as articles ('a', 'the' and the like) were consistently omitted from the recognised sentences, leading to a systemic reduction in recognition accuracy. The solution adopted by big technology companies is a sentence-level recogniser ensuring the grammatical correctness of the final recognised sentences. This may be viewed as a top-down feedback mechanism adding a strong likelihood to grammatically acceptable sentences under a full language model, once more exhibiting the adaptability of likelihood-based recognition methods. Since we do not use such techniques, the word-level recognition accuracy is not a useful metric for our experiments.

X.

Conclusion of part B

In the previous three chapters, HMMs models were built and tested on spike train data, increasing the complexity of the speech recognition task from a dataset of isolated phonemes to one containing continuous speech. This, in turn, reduced the maximal accuracy in the phoneme recognition task.

While the VCV dataset was too simple for probing the limits of a recognition algorithm as powerful as HMMs, the sTIMIT dataset may prove itself to be too complex. Of course, the fact that the best accuracy obtained on the spectrograms was around 50% when researchers have been able to reach 80% on the full TIMIT dataset allows us to think that the full capabilities of HMMs have not been used. This means that, in order to draw accurate conclusions about neural processing, the computational framework should be optimised first. Until then, a dataset such as the CUAVE dataset is more adapted to investigate the neural processing of real speech in a physiology experiment. As such, this methodology should be amenable to studying how the brain processes complex sounds in the central Auditory Pathway.

The effect of important parameters of the cochlear model was studied during this investigation, providing insights into the functional role of different types of ANFs:

- High Spontaneous Rate fibres seem to provide more speech information at lower sound levels using a rate code;
- A low calcium concentration threshold is necessary to convey speech information at small temporal scales.

On the engineering side, many points of interest arose. All results bring qualitatively similar conclusions at the word-level and the phone-level, with either an improved (sCUAVE) or reduced (sTIMIT) accuracy depending on the complexity of the dictionary. The reasons for this are varied:

- A digit recognition task has more phonemes than words, while a large-dictionary based recognition task has more words than phonemes;
- The word insertion penalty leads to a positive bias in the digit recognition task where all recognised words are about the same length, but a harmful effect when recognising real sentences, since words have a significantly different duration;
- In the digit recognition task, many phonemes are uniquely found within a single word, leading to a context-dependency enabling recognising a word by accurately recognising simply one of its phones. This is evidently not true for the sTIMIT dataset.

Researchers tend to use a single time-scale to extract feature vectors. This is bound to bring limitations related to auditory coding. One would expect this issue to take even more importance higher up in the Auditory Pathway as the auditory system temporally works across many scales.

Final Conclusion & Discussion

The Automatic Continuous Speech Recognition technologies have reached a performance level enabling their use by technological companies in many real-life applications, slowly closing the gap between human and machine speech recognition. The same technology can be applied to other time-varying signals such as neural data, as long as enough information is contained within the features extracted from these signals. This thesis studied the application of an ASR technology to spike train data, on a large-size dictionary, with a view to applications to electrophysiology experiments. Indeed, an efficient neural decoder for dynamical stimuli as quick as sounds would be a helpful engineering tool to tackle Auditory Neuroscience open issues.

Among the sea of machine learning algorithms that exist, it is not clear what algorithm should be used or whether different algorithms would interpret data differently. This seemed particularly true for neural data, where the neuron assumption states that neurons communicate and process information using a neural code. Different neurons may use codes that would be best decoded by different machine learning algorithms. Hence, the coding of an amplitude-modulated tone from different Cochlear Nucleus neuron types was evaluated and compared using a big collection of classification algorithms. Part A of this thesis suggests that the type of neural coding under study does not influence the type of Machine Learning algorithm that should be used for this task. This result thus encouraged us to opt for the best machine learning algorithm fitting our needs.

The well-developed Hidden Markov Models algorithm was thus chosen to build statistical models of spoken phonemes in part B of this thesis, using the HTK toolbox to train statistical models of either experimental or simulated spike train responses to sounds. The interpretability of such generative modelling tools can indeed be crucial in understanding results. However, in this quantitative study, we rely on the recognition accuracy as an overall measure of performance, to evaluate the amount of information present in the spike trains in a rate-code fashion. The parameters and processing were developed and optimised while increasing the complexity of the given speech recognition task, starting with a simple sinusoid envelope discrimination task and culminating with a large-vocabulary-based dataset containing multiple speakers and accents.

While still imperfect, such sparse representation of speech was found to be amenable to

a speech recogniser for a physiology experiment, using simulated data that amounts to an electrophysiology experiment in a digit recognition task. On the other hand, this framework did not yield satisfying results for a more complex speech dataset, namely a subset of the TIMIT dataset that contains continuous speech at normal speed spoken by many speakers.

The ∼13% margin between the results obtained with simulated spike trains and these obtained with spectrograms is enough to hope that, by improving the recognition framework, a proportionally better large-vocabulary-based ASR on spike trains could be obtained and tested on experimental data in order to probe how the brain processes complex and dynamical sounds.

In so investigating, many intermediate conclusions were drawn regarding neural coding and the differences between different types of neurons, that form this work's scientific contributions.

First, in the Cochlear Nucleus, population level differences on the amount of Amplitude-Modulation information that could be extracted reflected the impressive time precision with which some types of neuron such as Onset cells encode information, and the low-pass filter role that all units play as information processors at all sound levels (30, 50, 70 dB SPL). The data-mining approach to compare auditory population coding with classification algorithms is novel, shifting the classical view of chopper units as enhancing specific AM frequencies to a low-pass information transmitter. Indeed, the classical band-pass at medium and high sound levels using a synchronicity analysis [Frisina *et al.*, 1990; Rhode and Greenberg, 1994] is replaced by a low-pass shape for all sound levels. This low-pass shape is actually universal among the CN unit types.

The guinea-pig cochlear model from [Sumner *et al.*, 2003] was tested for the first time on continuous speech datasets, used as it was intended as the front-end of an ASR system. The Inferior Colliculus data collected by [Steadman, 2015] was tested with HMMs models for the first time.

In the auditory nerve, the spontaneous rate of a fibre seems to be correlated with different roles played by the High Spontaneous Rate fibres and the Low Spontaneous Rate fibres at different sound levels. Contrary to the literature, our result supports the view that HSR neurons have a higher capability at speech encoding using a rate coding than LSR units, but require a lower sound level to work optimally. These conclusions ought to change when considering a temporal code, necessary at high sound levels to circumvent the auditory nerve saturation mechanism [Holmberg *et al.*, 2007]. It seems that this shifting in roles as sound level is increased is biophysically permitted by a combination of variation in their threshold of activity and their dynamic range: HSR fibres have a low threshold and and a small dynamic range, while LSR have a higher threshold and a larger dynamic range [Sumner *et al.*, 2003]. This result is counter-intuitive since units with larger dynamic range might be thought of as

encoding more information — assuming the sound level is high enough to activate them.

The differences in Amplitude-Modulation encoding between Inferior Colliculus neurons and simulated cochlear data are quite unclear: the latter provided better results when all speakers were present in the training set, but worse results when one speaker's responses was removed from the training and comprised the whole test set. This hints at a neural processing extracting useful speech-related features being extracted in the guinea-pig's auditory pathway.

These tests, made on the simulated response to speech datasets of increasing complexity, show that a continuous digit recognition task is in principle amenable to an electrophysiological experiment. Conditioned to a significant improvement of the framework, a fully-fledged continuous speech recogniser on neural data, suited to an electrophysiology experiment, is plausible.

In the past few years, modelling fields have gradually become more and more computational. The collection of always-expending amounts of data and the improvement of the technology to process it have opened a route to let algorithms learn enough parameters to replace what was once handcrafted by expert knowledge by automated methods. Following this trend, Deep Learning ought to become common practice in all learning tasks, replacing the drive to improve biophysical models to understand pending scientific questions in Auditory Neuroscience with an increased use of data-driven discoveries.

However, the *in viva* recording of neural activity at the cellular level deep inside the brain is still a huge challenge since it requires invasive methods that damage the area under study, and also because no technology currently allows to track the neural information processed by the millions of neurons reacting to any stimulus, hence limiting the questions scientists may ask.

Methods and protocols to aggregate data and partial knowledge, thus developing a global understanding of neural mechanisms through a shared data-driven simulated brain builder, is likely to be the key to answering most Neuroscience open issues in the close future.

Our study blends in this trend, as it challenges a well-spread technology, the Hidden Markov Models, well-suited to the amount of data we may gather from an electrophysiology experiment, on one of the most complex auditory stimuli: human speech.

A plethora of classifiers and parameters were tested in this thesis; still, many limitations pervade the presented work:

- A rate coding was assumed in all speech processing tasks. As concluded in [Holmberg *et al.*, 2007], a better use of temporal precision and the abundance of fibres are im-

portant to achieve high accuracies at sound levels other than 70-ish dB, and possibly reach the accuracy obtained with spectrograms;

- Following [Steadman, 2015], a modification of the original filters was already applied in order to make sure that frequencies down to 100 Hz were represented in the cochleograms. However, guinea-pigs have wider bandpass filters than humans, possibly limiting the accuracy achievable;

- When simulating data using the cochlear model from [Sumner *et al.*, 2003], a single type of ANF is ever being tested in any given task, instead of a mixture of different types of fibres as in real cochleas. They may however carry complementary information at any set sound level;

- A training and testing validation protocol was used to reduce the important and limiting calculation time. A cross-validation protocol would allow more robust statistics;

- The use of set features in the back-end block is motivated by the time constraints imposed by electrophysiology experiments. It is possible that it already represents enough data to allow Deep Learning methods to outperform any fixed set of features;

- Computational limitations were a bottleneck to test more processing and parameters;

- Real data has not been tested for the final tasks on continuous speech. It is unknown whether an experiment lasting so long could provide data amenable to HMM modelling, with current experimental electrophysiology technologies. Surely the digit task has high chances for delivering acceptable results, as the dataset lasts for ten minutes only;

- The number of Gaussian mixtures was kept small to reduce the number of times when, using what HTK sees as 'bad data', HTK could not end its calculations;

- Timing information from the transcripts was not used, letting HTK find the word boundaries by itself. This might have a deleterious effect on the HMM training;

- To avoid the computational burden that triphones bring, only phoneme-level HMMs were used. However, using context-dependent statistical models might be necessary to achieve an acceptable accuracy on complex speech dataset;

- While HMMs are powerful statistical models, the assumptions they rely on (such as Gaussianity, number of mixtures or states) might restrict their modelling power. For example, the word-level results on the sTIMIT dataset suffer a bias due to the necessity to share a common word insertion penalty. The optimisation of this parameter on the

development set leads to a significant absence of short words such as articles, thus decreasing the word-level accuracy. The use of a sentence-level recogniser ensuring the grammatical correctness of the recognised sentence, as used by big technology companies, should correct this.

Any change in these directions might lead to improvements in the accuracy. Naturally, simplifying and extending the computational framework would greatly benefit additional tests.

This work can be extended by incorporating models of the central auditory system and comparing the results with those obtained with experimental data coming from various nuclei along the auditory pathway. Speech, then used as the default dynamical stimulus, would allow the comparison of experimental and simulated data on highly complex and varied neural codes, thus delineating the limits of models on more complex signals than the ones they were built upon.

Another natural extension of this work is in the field of brain implant technologies, by replacing spike train data by ECoG data [Mesgarani and Chang, 2012]. This semi-invasive technology might become more well-spread in years to come, either to bypass speaking disorders [Brunner *et al.*, 2017] or to artificially enhance human mental abilities using electric stimulations in a closed-loop circuit [Sitaram *et al.*, 2017]. One could then train the models to recognise sounds and words in real time, in order to let mute, locked-in or curious people speak their mind [Pasley and Knight, 2013].

Regarding the ability of High Spontaneous Rate and Low Spontaneous Rate fibres to encode speech, a deeper investigation on the reasons why HSR fibres gave better results might shed light on the neural coding of speech in the Auditory Nerve Fibre at various sound levels and background noise levels, thus leading to a better understanding of the rate code and temporal code used by ANFs or CN units to deal with noise. Testing our framework on a human cochlear model and adding features related to temporal coding [Holmberg *et al.*, 2007] is a natural first step in this direction.

On the machine learning side, this work extracts set features before applying a machine learning algorithm. In both parts of this thesis, the front-end is based on set features. As neural nets improve, they ought to replace or complement set features on smaller datasets, thus guiding us to the most useful features present in the input.

Beyond speech recognition, relating trained models of phonemes to more abstract models of speech intelligibility and of psychoacoustic would serve all disciplines involve.

Finally, as originally intended, one could conceive an experiment using simultaneous speakers to probe the stream segregation mechanisms at different nuclei: using a technology similar to that used in this thesis (HMMs) but able to recognise two sentences spoken at the

same time, we would expect that running the same experiment at various places along the auditory pathway would shed light on where stream segregation takes place in the central auditory system, assuming more abstract features of the sounds (here, phonemes) would be extracted along the auditory pathway. This, in turn, should lead to improving our understanding of how brains solve cocktail party problems. It is unknown whether guinea-pigs would need to develop neural mechanisms for speech feature extraction, but since this holds true for dogs, monkeys and parrots, animals able to be trained to respond to calls, it may be a natural property of their auditory system to recognise phonemes, the same way we easily recognise so many different types of complex noises (wind blowing, water drop, rain, animal vocalisations).

Appendices

11.

Complements to Part A

11.1 Choice of Modulation Frequencies

This subsection gives in raw form, in figure 11.1 and table 11.2, the reason why the modulation frequencies $\{50, 150, \ldots, 1150\}$ Hz were chosen in chapter III, as a good compromise between a high number of smaller subsets satisfying the same conditions, and those conditions being rich enough in range and number of modulation frequencies.

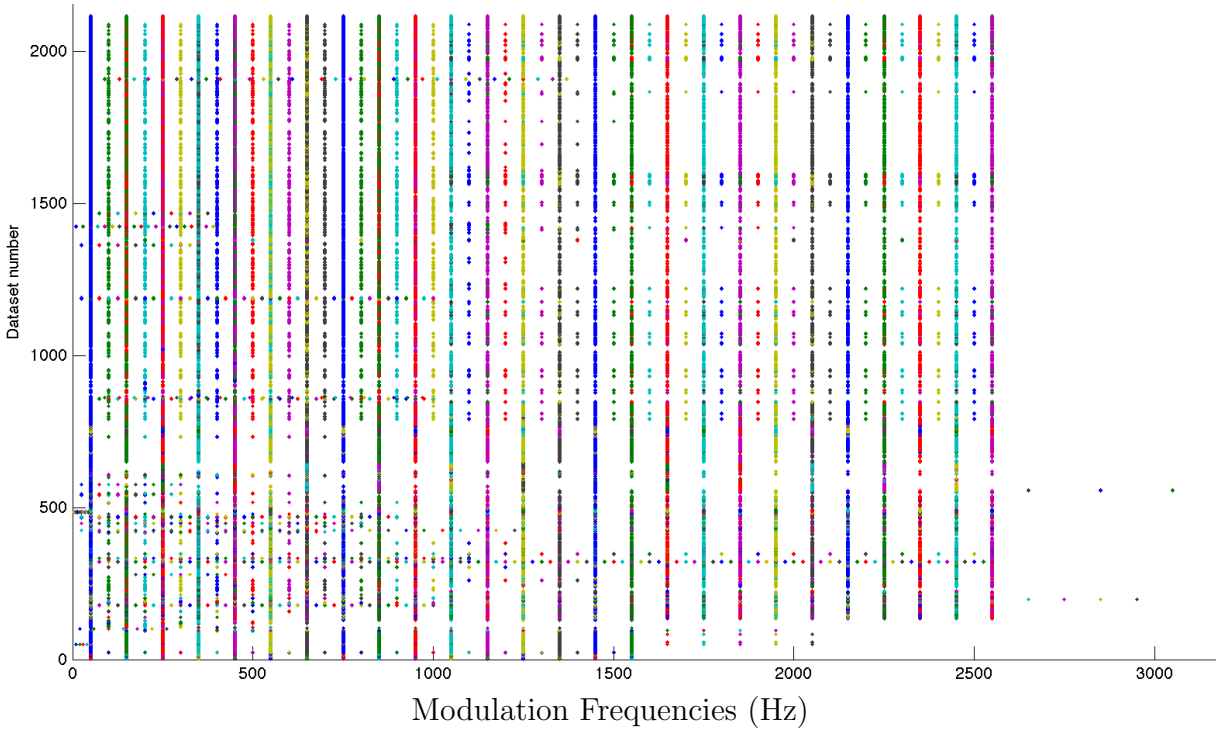


Fig. 11.1 Representation of all modulation frequencies (x-axis) present in the 2118 spike train datasets (y-axis; grouping of spike trains corresponding to a single animal, a single neuron, a single modulation level). Colouring is for illustration purposes only.

modFreqs	LowF	PL	PLN	ChT	ChS	OnL	OnI	OnC	PBU	Unu	UNC
[ ]	370 370	370 370	444 444	152 152	339 339	58 58	15 15	74 74	179 179	43 43	74 74
50	354 292	370 273	427 358	152 134	317 280	58 34	15 0	66 47	179 129	43 25	74 47
50,100	263 208	56 25	75 45	19 14	146 122	3 2	2 0	6 4	78 48	0 0	19 12
50:50:150	263 206	56 24	75 43	19 14	146 121	3 2	2 0	6 4	78 42	0 0	19 12
50:50:650	154 60	56 5	75 16	19 10	145 62	3 2	2 0	6 4	78 19	0 0	19 6
50:50:850	108 24	56 4	69 13	19 6	144 45	3 2	2 0	6 1	78 4	0 0	16 0
50:50:1050	30 5	24 1	32 11	9 4	79 16	3 0	2 0	3 1	27 2	0 0	0 0
50,150	348 274	309 207	420 339	150 126	317 273	55 28	15 0	66 46	176 111	43 20	68 41
50:100:250	342 246	309 199	420 329	150 121	317 264	55 25	15 0	66 39	176 103	43 20	68 39
50:100:350	327 219	309 184	420 321	150 119	317 252	55 22	15 0	66 37	176 102	43 19	68 33
50:100:550	272 148	309 167	420 287	150 106	316 229	55 17	15 0	66 31	176 87	43 18	68 32
50:100:750	217 80	309 156	417 268	150 96	316 190	55 16	15 0	66 30	176 61	43 13	65 23
50:100:950	171 52	309 136	414 238	150 85	313 132	55 13	15 0	66 25	173 40	43 7	61 21
50:100:1150	98 23	271 112	375 212	140 61	253 85	55 10	15 0	63 19	113 21	43 4	49 21
50:100:1350	72 1	262 90	360 177	134 37	247 49	52 10	15 0	63 11	108 10	43 2	49 18
50:100:1550	52 1	259 73	360 134	134 18	244 23	50 6	15 0	61 8	105 4	43 2	46 10
50:100:1750	50 1	234 50	353 96	129 9	218 15	46 2	15 0	59 3	87 0	40 0	45 8
50:100:1950	41 1	234 37	353 76	129 4	218 7	46 2	15 0	56 2	84 0	31 0	45 6
50:100:2150	36 0	231 22	350 41	127 0	215 3	46 2	15 0	53 0	81 0	29 0	45 6
50:100:2350	36 0	231 17	350 24	127 0	215 3	46 0	15 0	53 0	81 0	29 0	45 3
50:100:2550	36 0	231 9	350 12	127 0	215 3	46 0	15 0	53 0	81 0	29 0	45 2
50:100:2750	0 0	0 0	0 0	0 0	0 0	0 0	0 0	0 0	0 0	0 0	3 1
100	269 219	56 26	75 47	19 14	148 125	3 2	2 1	6 4	78 49	0 0	19 12
100,200	260 198	56 24	75 41	19 14	148 119	3 2	2 0	6 4	78 42	0 0	19 12
100:100:1000	90 20	56 4	69 13	19 4	138 27	3 0	2 0	6 1	75 2	0 0	12 0
100:100:2000	3 0	12 0	15 0	9 0	57 0	0 0	2 0	3 0	12 0	0 0	0 0

Tab. 11.2 This table contains the number of datasets (grouped spike trains coming from a single animal, a single neuron, and a single modulation frequency), per cell type, containing certain modulation frequencies (first column, in Matlab notation min:step:max). In black, the number of datasets of each cell type that contains responses of the modulation frequencies given in the left column. In grey, the number of datasets where each modulation frequency tested verifies certain conditions: that for each modulation frequency in the array of the left column, there be at least 25 spike trains each containing at least 2 spikes. The modulation frequencies used in the numerical experiment were selected based on the following trade-off: having enough modulation frequencies in each dataset, and having enough datasets. The list of modulation frequencies 50, 150, ..., 1150 Hz was chosen as a result. The vectors of modulation frequencies were chosen based on figure 11.1.

11.2 Synopsis of all Weka classifiers used

This section gives the implementation of the features used in chapter III, for each processing, referred to by the three keywords 'ISI', 'timeBined' and 'spikeMetric'.

11.2.1 Cleaning

The spike trains used are the first non-empty 25 spike trains of each dataset, keeping only the spikes occurring between 20 ms and 100 ms.

11.2.2 Preprocessing

The spike trains are preprocessed in 3 ways, and saved in ARFF files for later use by Weka.

- **ISI:** *(parameter-free)*
  - First, `@(spkTr)diff([0 spkTr])` is applied to each spike train;
  - From each ISI train, a list of features is extracted:

```
Mean=                @(x)mean(x);
secondElem=          @(x)x(2);
thirdElem=           @(x)x(3);
meanfunc1=           @(x)mean(1./x(2:end));
varISI=              @(x)log(1+var(x));
getRidFirstElement= @(x)(x(2:end));
accel=               @(x)(x-circshift(x,[1,1]));
varAccel=            @(x)var(accel(getRidFirstElement(x)));
numSpikes=           @(x)length(x);
CV=                  @(x)(mean(x))^2/var(x);
CX=                  @(x)log((var(x))^3/skewness(x)^2);
CY=                  @(x)log((var(x))^3/skewness(x)^2)/mad(x);
FourMRe=             @(x)mean(real(fft(x)));
FourVRe=             @(x)var(real(fft(x)));
FourMIm=             @(x)mean(imag(fft(x)));
FourVIm=             @(x)var(imag(fft(x)));
histo_short =        @(x)histc(x,0:0.2:4);
```

  - Print out these values in the ARFF file, preceded by the modulation frequency associated with the spike train.

- **timeBined** *(1 parameter: bin size)*
  - Get the maximal spike train duration (i.e. the maximal value among all our spike trains);
  - From that and the step size, infer the number of bins required (`ceil(maxTrainDur/timeStep);`)

  - Replace all spike trains by an array of integers every bin; each number being the number of spikes that were in this time window;
  - Print out this rectangular matrix, with their attribute to classify.

- **spikeMetric** *(2 parameters: metric with own parameters (cost for Victor & Purpura $d(\cdot,\cdot,c)$), transformation kernel k)*
  - The implementation of the Victor & Purpura metric (described in figure III.6) used comes from SPIKY [Kreuz *et al.*, 2014];
  - From the spike trains $x_1, \ldots, x_N$, is created the N^2 matrix $D = (d(x_i, x_j, c))_{i,j\in\{1\ldots,N\}}$;
  - The kernel k is applied to every element of D: $K = (k(d(x_i, x_j, c)))_{i,j\in\{1\ldots,N\}}$;
  - The matrix is printed out in a .matrix file, preceded by the size of the matrix;
  - A .arff file containing 2 attributes is saved: identifier and class.
    'identifier' is a nominal attribute with N instances (row1, row2, ..., rowN),
    'class' is the nominal attribute to classify,
    so that there are N lines of data: {row1, modFreq1},... {rowN, modFreqN}.

11.2.3 Classifiers

A high number of classification algorithms is run on the .arff files and statistics are extracted after 10-fold cross-validation; more measures based on the confusion matrices were tested but not included in this work to reduce redundancy. These are all given below.

- **Apply weka filters**
  - Put class as the last attribute.

- **Extract**
  - Percentage of correct classification;
  - Confusion matrix;
  - Number of instances;
  - Kappa statistic;
  - K&B information;
  - Weighted Area under ROC.

- **Run** [java notation: "**Class** (*package*)" otherwise called by "package.Class"]
  Synopsis based on `http://wiki.pentaho.com/display/DATAMINING/Data+Mining+Algorithms+and+Tools+in+Weka`.

- **AdaBoostM1** (*meta*) Class for boosting a nominal class classifier using the Adaboost M1 method. Only nominal class problems can be tackled. Often dramatically improves performance, but sometimes overfits.
- **BFTree** (*trees*) Class for building a best-first decision tree classifier. This class uses binary split for both nominal and numeric attributes. For missing values, the method of 'fractional' instances is used.
- **Bagging** (*meta*) Class for bagging a classifier to reduce variance. Can do classification and regression depending on the base learner (classifier or regression algorithm).
- **BayesNet** (*bayes*) Bayes Network learning using various search algorithms and quality measures. Base class for a Bayes Network classifier. Provides datastructures (network structure, conditional probability distributions, etc.) and facilities common to Bayes Network learning algorithms like K2 and B.
- **ClassificationViaClustering** (*meta*) A simple meta-classifier that uses a clusterer for classification. For cluster algorithms that use a fixed number of clusterers, like SimpleKMeans, the user has to make sure that the number of clusters to generate are the same as the number of class labels in the dataset in order to obtain a useful model.
- **ClassificationViaRegression** (*meta*) Class for doing classification using regression methods. Class is binarised and one regression model is built for each class value.
- **DecisionStump** (*trees*) Class for building and using a decision stump. Usually used in conjunction with a boosting algorithm. Does regression (based on mean-squared error) or classification (based on entropy). Missing is treated as a separate value.
- **END** (*meta*) A meta classifier for handling multi-class datasets with 2-class classifiers by building an ensemble of nested dichotomies.
- **FilteredClassifier** (*meta*) [Default filter used] Class for running an arbitrary classifier on data that has been passed through an arbitrary filter. Like the classifier, the structure of the filter is based exclusively on the training data and test instances will be processed by the filter without changing their structure.
- **IB1** (*lazy*) Nearest-neighbour classifier. Uses normalised Euclidean distance to find the training instance closest to the given test instance, and predicts the same class as this training instance. If multiple instances have the same (smallest) distance to the test instance, the first one found is used.
- **IBk** (*lazy*) K-nearest neighbours classifier. Can select appropriate value of K based on cross-validation. Can also do distance weighting.

- **J48** (*trees*) Class for generating a pruned or unpruned C4.5 decision tree.
- **JRip** (*rules*) This class implements a propositional rule learner, Repeated Incremental Pruning to Produce Error Reduction (RIPPER). See references on Weka website.
- **KStar** (*lazy*) K* is an instance-based classifier, that is the class of a test instance is based upon the class of those training instances similar to it, as determined by some similarity function. It differs from other instance-based learners in that it uses an entropy-based distance function.
- **LADTree** (*trees*) Class for generating a multi-class alternating decision tree using the LogitBoost strategy.
- **LWL** (*lazy*) Locally weighted learning. Uses an instance-based algorithm to assign instance weights which are then used by a specified WeightedInstancesHandler. Can do classification (e.g. using naive Bayes) or regression (e.g. using linear regression).
- **LogitBoost** (*meta*) Class for performing additive logistic regression. This class performs classification using a regression scheme as the base learner, and can handle multi-class problems.
- **MultiClassClassifier** (*meta*) A metaclassifier for handling multi-class datasets with 2-class classifiers. This classifier is also capable of applying error correcting output codes for increased accuracy.
- **Multischeme** (*meta*) Class for selecting a classifier from among several using cross validation on the training data or the performance on the training data. Performance is measured based on percent correct (classification) or mean-squared error (regression).
- **NaiveBayes** (*bayes*) Class for a Naive Bayes classifier using estimator classes. Numeric estimator precision values are chosen based on analysis of the training data. For this reason, the classifier is not an UpdateableClassifier (which in typical usage are initialized with zero training instances) – if you need the UpdateableClassifier functionality, use the NaiveBayesUpdateable classifier. The NaiveBayesUpdateable classifier will use a default precision of 0.1 for numeric attributes when buildClassifier is called with zero training instances.
- **OneR** (*rules*) Class for building and using a 1R classifier; in other words, uses the minimum-error attribute for prediction, discretizing numeric attributes.
- **PART** (*rules*) Class for generating a PART decision list. Uses separate-and-conquer. Builds a partial C4.5 decision tree in each iteration and makes the 'best' leaf into a rule.

- **REPTree** (*trees*) Fast decision tree learner. Builds a decision/regression tree using information gain/variance and prunes it using reduced-error pruning (with backfitting). Only sorts values for numeric attributes once. Missing values are dealt with by splitting the corresponding instances into pieces (i.e. as in C4.5).
- **RandomForest** (*trees*) Class for constructing a tree that considers K randomly chosen attributes at each node. Performs no pruning.
- **RandomSubSpace** (*meta*) This method constructs a decision tree based classifier that maintains highest accuracy on training data and improves on generalisation accuracy as it grows in complexity. The classifier consists of multiple trees constructed systematically by pseudorandomly selecting subsets of components of the feature vector, that is, trees constructed in randomly chosen subspaces.
- **RandomTree** (*trees*) Class for constructing a forest of random trees.
- **RBFClassifier** (*functions*) Class implementing radial basis function networks for classification, trained in a fully supervised manner using WEKA's Optimisation class by minimising squared error with the BFGS method. Note that all attributes are normalised into the [0,1] scale. The initial centres for the Gaussian radial basis functions are found using WEKA's SimpleKMeans. The initial sigma values are set to the maximum distance between any centre and its nearest neighbour in the set of centres.
- **Ridor** (*rules*) An implementation of a RIpple-DOwn Rule learner. It generates a default rule first and then the exceptions for the default rule with the least (weighted) error rate. Then it generates the 'best' exceptions for each exception and iterates until pure. Thus it performs a tree-like expansion of exceptions.The exceptions are a set of rules that predict classes other than the default. IREP is used to generate the exceptions.
- **SimpleCart** (*trees*) Class implementing minimal cost-complexity pruning. Note when dealing with missing values, use 'fractional instances' method instead of surrogate split method.
- **SimpleLogistic** (*functions*) Classifier for building linear logistic regression models. LogitBoost with simple regression functions as base learners is used for fitting the logistic models. The optimal number of LogitBoost iterations to perform is cross-validated, which leads to automatic attribute selection.
- **SMO** (*functions*) Sequential minimal optimisation algorithm for training a support vector classifier. This implementation globally replaces all missing values and transforms nominal attributes into binary ones. It also normalises all attributes by default. Multi-class problems are solved using pairwise classification (1-vs-1 and if logistic models are built pairwise coupling according to Hastie and Tibshirani,

1998). To obtain proper probability estimates, use the option that fits logistic regression models to the outputs of the support vector machine. In the multi-class case the predicted probabilities are coupled using Hastie and Tibshirani's pairwise coupling method.

- **ZeroR** (*rules*) Class for building and using a 0-R classifier. Predicts the mean (for a numeric class) or the mode (for a nominal class).
- The following classifiers were NOT used for being too slow on the previous tests: **Dagging** (*meta*), **MultilayerPerceptron** (*functions*), **NaiveBayesMultinomial** (*bayes*), **Logistic** (*functions*), **LMT** (*trees*).

11.3 Scatter Plots

Figure 11.3 allows us to check that for NaiveBayes, LogitBoost and SimpleLogistic classifiers, the distribution of measures with respect to the unit type is about the same on all scatter plots or histogram as it was on MultiClassClassifier, SMO and END. This is given in addition to figure IV.3, where the points issued from the NaiveBayes, LogitBoost and SimpleLogistic classifiers were on top of each other.

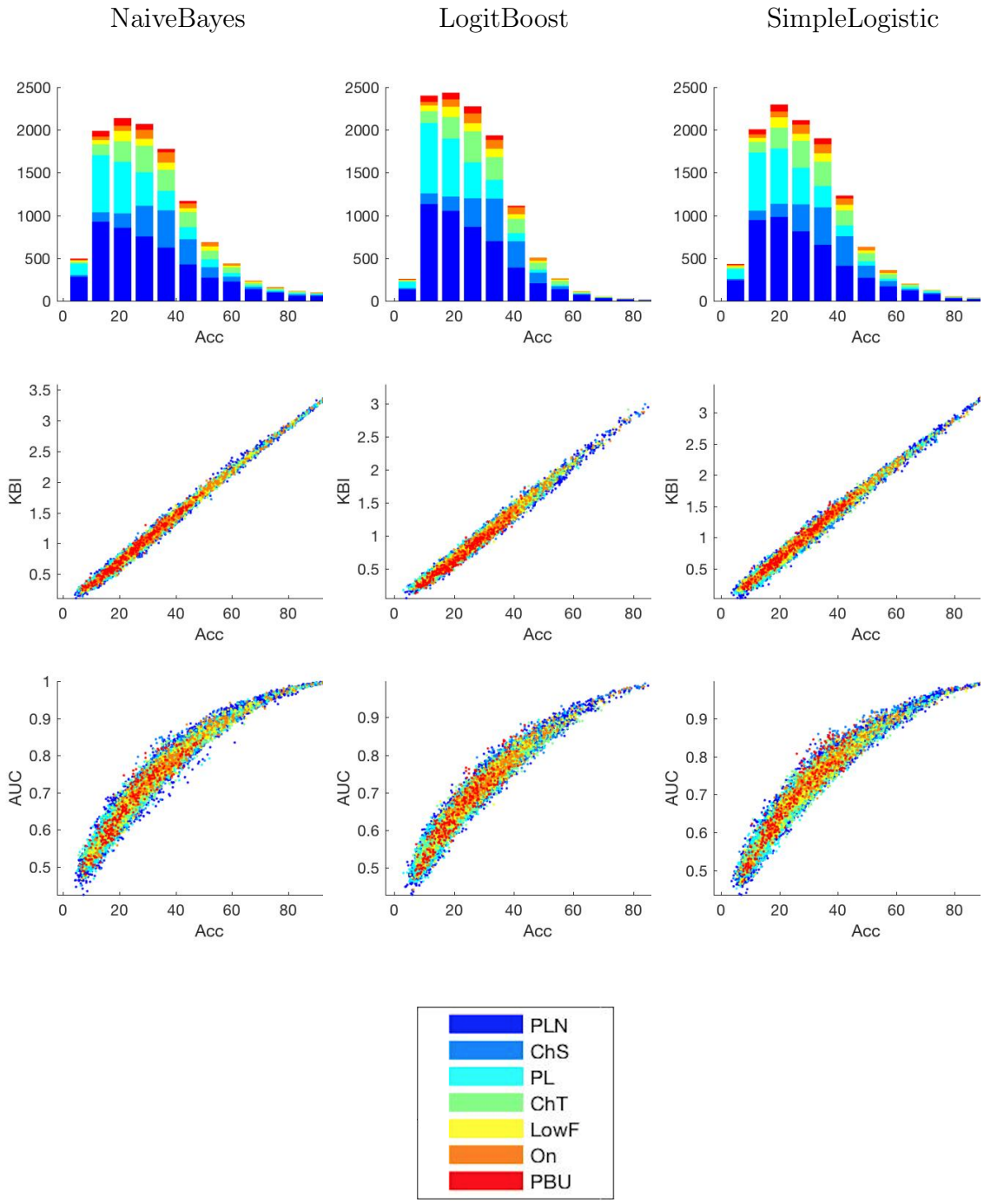


Fig. 11.3 On the first row are the accuracy histograms using all results obtained with NaiveBayes (left column), LogitBoost (middle column) and SimpleLogistic (right column) algorithms on all datasets of chapter III. On the second and third row are the scatter plots of accuracy VS KBI and AUC, respectively. Points are coloured by unit type; see legend.

12.

Complements to Part B

12.1 Probability Theory

12.1.1 Random Variables and Probability Density Functions

Modern Probability is based on notions from the mathematical fields of Topology and measure theory, such as measurability, down to the definition of its most basic concept, the random variable. See [Williams, 1991] for an accessible account of measure theory in Probability, and for all proofs of basic Probability properties asserted in the below.

Let Ω be a set - the field of real numbers $\mathbb{R}$ is the main focus here, together with the Euclidian spaces $\mathbb{R}^n$. Let's define a topology T on Ω as a family of subsets of Ω such that

- T contains Ω and the empty set,
- any union of elements of T is an element of T,
- any intersection of finitely many elements of T is an element of T.

The elements of a topological space (Ω, T) are called open sets, and their complementary sets are called closed. In this thesis, topological spaces are generated by open intervals in $\mathbb{R}$, or their cartesian product in $\mathbb{R}^n$, on which is assumed the existence of the Lebesgue measure, such that the measure of any interval $]a, a+\epsilon[$ is $\epsilon > 0$.

When an experiment is performed (such as rolling a dice, playing darts, submitting a grant proposal), there usually is a wide range of outputs Ω on which one can define a measure $\mathbb{P}$ such that $\mathbb{P}(\Omega) = 1$. This is a probability measure. A (real) random variable is defined as a function mapping each output $\zeta \in \Omega$ of the experiment to a number $X(\zeta) \in \mathbb{R} \cup \{\pm\infty\}$ with the requirements of being infinite with null probability (i.e. $\mathbb{P}(X \in \{\pm\infty\}) = 0$), and that of being measurable, which is a topological requirement that usually ensures one is using well-defined sets[1]. We systematically make this assumption, in order to be able to use sets of the form $\{X \leq a\} := \{\zeta \in \Omega / X(\Omega) \leq a\}$[2]. They form a useful subfamily of the larger family of measurable sets, usually called events in a probability context.

[1] The Banach-Tarsky paradox states that the axiom of choice allows for the construction of nonmeasurable sets used to cut a ball into 2 balls each having the original ball's volume. In light of such phenomena, working with measurable sets is a safety net.

[2] Note this notation does not explicitly use the probability space Ω; a common habit in probability.

For X a real random variable, the elements of Ω that are in the set $\{X \leq x\}$ change as x takes different values. This allows us to define the distribution function of the random variable X as

$$F_X(x) = \mathbb{P}(X \leq x). \tag{12.1}$$

This function is increasing and right-continuous, converges to 0 when x tends to $-\infty$ and to 1 when x tends to $+\infty$. It is continuous if and only if X has no atom, no single value $x \in \mathbb{R}$ such that $\mathbb{P}(X = x) > 0$. Whenever the function F_X can be written as

$$F_X(x) = \int_{-\infty}^{x} f_X(t)\,\mathrm{d}t, \tag{12.2}$$

for a positive function f_X of total mass 1, the function f_X is called the probability density function (pdf) of the random variable X, also called its law. This concept of pdf proved itself to be very useful in Probability and Statistics. The most common examples of pdfs are for Normal (Gaussian) random variables, exponential, gamma and uniform distributions, all appearing in this work.

Gaussian Distribution
We say that X is a Normal or Gaussian random variable with mean μ and variance σ^2 if its density function is given by

$$f_X(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right). \tag{12.3}$$

Exponential Distribution
We say that X has an exponential distribution with parameter $\lambda > 0$ if its distribution density is given by

$$f_X(x) = \begin{cases} \lambda \exp(-\lambda x) & x \geq 0, \\ 0 & \text{otherwise.} \end{cases} \tag{12.4}$$

Uniform Distribution
Let A be a Borel set with positive and finite Lebesgue measure $0 < L(A) < +\infty$. The uniform distribution over A is defined by the distribution

$$f_X(x) = \begin{cases} \frac{1}{L(A)} & x \in A, \\ 0 & \text{otherwise.} \end{cases} \tag{12.5}$$

12.1.2 Sampling from a Known Probability Density Function

Sampling from a continuous distribution f_X means that generating data in such a way that around every point $x \in \mathbb{R}$, the probability that the generated sample belongs to the

infinitesimal interval $[x, x + \mathrm{d}x]$ is $f_X(x)\,\mathrm{d}x$. For example, for a uniform distribution, any interval $[x, x + \mathrm{d}x]$ for $x \in A$ but not on its border has the same probability to contain the sampled value, for $\mathrm{d}x$ an infinitesimal quantity.

It is often important to be able to generate samples drawn from a chosen distribution, typically to test an hypothesis. Classical programming languages (Matlab, C, python, ...) have efficient random or pseudo-random number generators for the classical laws.

12.1.3 Estimating an Unknown Probability Density Functions

Once we have accumulated data points corresponding to different sampling distributions - in our example, the information corresponding to the walk - we model each feature vector X as being drawn from a pdf characterised by the unknown parameter λ that we want to estimate. Until we make further assumptions, we are facing a non-parametric problem, because each of these densities is unknown (thus belongs to an infinite dimensional space of admissible functions) and has to be guessed from a finite-size sample. The problem becomes simpler when one assumes these densities belong to a finite-dimensional set of parameters: for example, we can reduce the dimensionality to one, by assuming that the distribution belongs to the exponential family. This trade-off between richness of the model and easiness to fit a good model is always a modelling choice.

12.2 Spike Train Simulation

The cochlear model used for this thesis [Sumner *et al.*, 2003] outputs the firing rates at the required BFs at a sample rate of 100 kHz. The generated spike trains are modelled as inhomogeneous Poisson processes, on which we enforce a random refractory period. Imposing refractoriness actually modifies the law of probability of the final point process, so that the PSTH should converge to another law. This section presents the spike train modelling (see figure 12.1), the different algorithms used for spike generation, and how refractoriness impacts the spiking probability.

12.2.1 Generating Poisson Processes

12.2.1.1 Definition of Poisson Processes

An arrival process is a sequence of increasing random variables $0 < T_1 < T_2 < \dots$ called arrival times. A renewal process is an arrival process for which the sequence of inter-arrival times $(T_{i+1} - T_i)_{i\in\mathbb{N}}$ is a sequence of independent and identically distributed (IID) random variables. A Poisson process or homogeneous Poisson process is a renewal process in which the interarrival intervals have an exponential distribution function; i.e., for some real $\lambda > 0$ called rate or intensity, each $(T_{i+1} - T_i)$ has the density $f(x) = \lambda \exp(-\lambda x)\mathbb{1}_{\mathbb{R}^+}(x)$, where

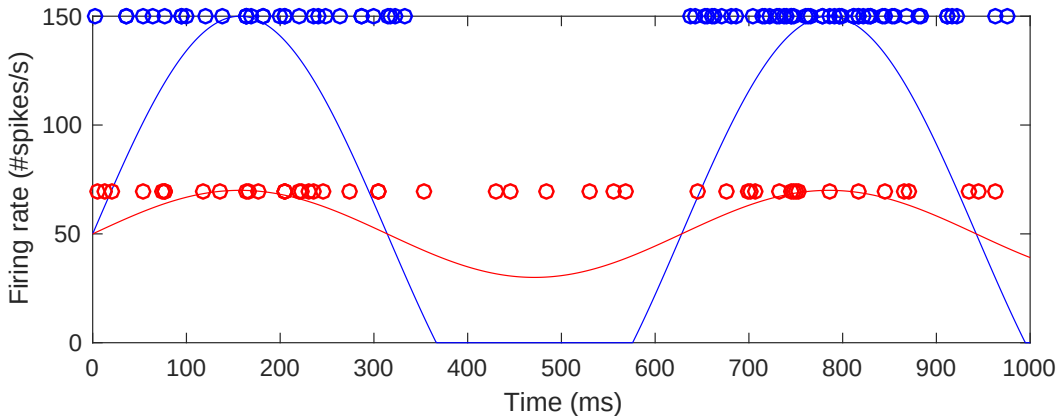


Fig. 12.1 Spike train generation with two different sinusoidal rates using the thinning algorithm, without refractoriness to emphasise the higher spiking activity where the rate is high. Spikes are shown as points along the horizontal line with ordinate the maximal rate (150 and 70 spikes per second, respectively).

$\mathbb{1}_{\mathbb{R}^+}$ denotes the Heaviside function.

Theorem: *For a Poisson process N of rate $\lambda > 0$, we have, for $0 < s < t$,*

$$\mathbb{P}(N_t - N_s = n) = \frac{(\lambda(t-s))^n \exp(-\lambda(t-s))}{n!}.$$

Alternatively, this property could be used to define a Poisson process. We will actually use it for the following definition: An inhomogeneous Poisson Process N is a counting process with a non-constant intensity function $\lambda(t) \geq 0, t \geq 0$ if

- $N(0) = 0$,
- For each $t > 0, N(t)$ has a Poisson distribution with mean $m(t) = \int_0^t \lambda(s)\,\mathrm{d}s$,
- The interarrival times are independent random variables.

Theorem: *For an inhomogeneous Poisson process N with rate $\lambda(t)$, we have, for $0 < s < t$*

$$\mathbb{P}(N_t - N_s = n) = \frac{\left(\int_s^t \lambda(u)du\right)^n \exp\left(-\int_s^t \lambda(u)du\right)}{n!}.$$

We now present two ways to simulate an inhomogeneous Poisson processes, with a short discussion on their complexity and efficiency. The take-home message being that the thinning method is recommended when working with a bounded rate, as it is exact, usually faster, and does not require any integration.

12.2.1.2 Simulation of Poisson Processes

Homogeneous Poisson Processes Simulation

A homogenous Poisson process with constant parameter $\bar{\lambda} > 0$ is defined by a sequence of independent random variables following an exponential distribution of parameter $\bar{\lambda}$. We can simulate each of these exponential variables as

$$E = -\ln(U)/\bar{\lambda}, \tag{12.6}$$

where U is a uniform variable $U \sim \mathcal{U}([0,1])$. Indeed, its cumulative function is for $y \in \mathbb{R}^+$,

$$\mathbb{P}(E \leq y) = \mathbb{P}(-\ln(\mathrm{U})/\bar{\lambda} \leq y) = \mathbb{P}(U \geq e^{-\bar{\lambda}y}) = 1 - e^{-\bar{\lambda}y}$$

and is 0 for $y < 0$, which is the cumulative function of an exponential distribution of parameter $\bar{\lambda}$ and cumulative distributions characterise the law of real random variables.

We now present two methods to simulate an inhomogeneous Poisson process with rate $\lambda(t) \geq 0$. The methods' names are not standardised.

Inhomogeneous Poisson Processes Simulation by the Binning Method

If we are working in a discrete time setting $t_i = i\delta/n$, we might want to simulate our Poisson process by choosing randomly in which bin a spike took place. This is referred to as the binning method. We know the probability of having a spike in the first bin:

$$\mathbb{P}(N_{\frac{\delta}{n}} = 1) = e^{-\int_0^{\frac{\delta}{n}} \lambda(t)\,\mathrm{d}t} \int_0^{\frac{\delta}{n}} \lambda(t)\,\mathrm{d}t,$$

and we will use the two approximations $e^s = 1+s+s^2/2+o(s^2)$, $\lambda(s) = \lambda(0)+\lambda'(0)s+o(s)$. Hence

$$\int_0^s \lambda(t)\,\mathrm{d}t = \int_0^s (\lambda(0) + \lambda'(0)t + o(t))\,\mathrm{d}t = \lambda(0)s + \lambda'(0)s^2/2 + o(s^2)$$

and

$$\begin{aligned}
\mathbb{P}(N_s = 1) &= \left(1 - \int_0^s \lambda(t)\,\mathrm{d}t + \frac{1}{2}\left(\int_0^s \lambda(t)\,\mathrm{d}t\right)^2 + o(s^2)\right) \int_0^s \lambda(t)\,\mathrm{d}t \\
&= \left(1 - (\lambda(0)s + \lambda'(0)s^2/2) + \frac{1}{2}\left(\lambda(0)s + \lambda'(0)s^2/2\right)^2 + o(s^2)\right)(\lambda(0)s + \lambda'(0)s^2/2 + o(s^2)) \\
&= \lambda(0)s + \left(\lambda'(0)/2 - \lambda(0)^2\right)s^2 + o(s^2),
\end{aligned}$$

leading to

$$\mathbb{P}(N_{(i+1)\delta/n} - N_{i\delta/n} = 1) = \lambda\left(\frac{i\delta}{n}\right)\frac{\delta}{n} + \left(\lambda'\left(\frac{i\delta}{n}\right)/2 - \lambda\left(\frac{i\delta}{n}\right)^2\right)\left(\frac{\delta}{n}\right)^2 + o(n^{-2}).$$

At the second order, we need to account for the probability of having two spikes in a bin:

$$\mathbb{P}(N_{(i+1)\delta/n} - N_{i\delta/n} = 2) = e^{-\int_{\frac{i\delta}{n}}^{\frac{(i+1)\delta}{n}}\lambda(t)dt}\left(\int_{\frac{i\delta}{n}}^{\frac{(i+1)\delta}{n}}\lambda(t)dt\right)^2 = \lambda\left(\frac{i\delta}{n}\right)^2\frac{\delta^2}{n^2} + o(n^{-2})$$

and no higher term

$$\mathbb{P}(N_{(i+1)\delta/n} - N_{i\delta/n} \geq 3) = o(n^{-2}).$$

Since we need to do this approximation n times for a time period of one second, this gives us an error of order $1/n$, counting the spikes we missed in the bins.

Expectation of the ISI for a Homogeneous Poisson Process

The expectation of an exponential variable E with parameter λ is readily calculated:

$$\mathbb{E}(E) = \int_0^{+\infty}\lambda t e^{-\lambda t}\,\mathrm{d}t = \frac{1}{\lambda}. \tag{12.7}$$

Expectation of the ISI distribution by binning

Having an ISI of k bins when simulatng a Poisson process of fixed parameter λ with $0 \leq \lambda \leq 1$ is equivalent to having $k-1$ consecutive 0's and a 1, drawing k times a Bernouilli distribution with parameter λ, which is to say that this law follows a geometric distribution of parameter λ, of which we calculate the expectation:

$$\mathbb{P}(\mathrm{ISI}_b = k) = (1-\lambda)^{k-1}\lambda$$
$$\mathbb{E}(\mathrm{ISI}_b) = \frac{1}{\lambda}.$$

For values of λ higher than 1, our algorithm still compares the 'probability' with a uniform variable smaller than one, so that

$$\forall\lambda > 0, \qquad f_b(\lambda) := \mathbb{E}(\mathrm{ISI}_b(\lambda)) = \max\left(1, \frac{1}{\lambda}\right), \tag{12.8}$$

which is the actual expectation for the ISI of a real Poisson process as long as $0 \leq \lambda \leq 1$ (see formula 12.7) and is numerically verified in figure 12.3.

Inhomogeneous Poisson Processes Simulation by the Thinning method

Let $(N_t)_{t\geq 0}$ be a Poisson($\bar{\lambda}$) process, with fixed positive and finite $\bar{\lambda} \geq \max_t(\lambda(t))$. We

simulate it by summing independent exponential variables $\mathrm{Exp}(\bar{\lambda})$. Once we have this spike train $(T_i)_{1\leq i\leq N}$, we keep each spike T_i with probability $\lambda(T_i)/\bar{\lambda}$. It gives us the counting process $(\tilde{N}_t)_{t\geq 0}$ with non-constant rate $\lambda(t) \geq 0$. We verify it is an inhomogeneous Poisson process:

$$\begin{aligned}\mathbb{P}(\tilde{N}_{t+\mathrm{d}t} - \tilde{N}_t = 1) = \mathbb{P}\left(N_{t+\mathrm{d}t} - N_t = 1; \text{this spike is kept}\right) \\ + \mathbb{P}(N_{t+\mathrm{d}t} - N_t \geq 2; \text{all spikes but one removed}).\end{aligned}$$

By independence of the thinning process with the counting process N and $0 \leq \lambda(t)/\bar{\lambda} \leq 1$,

$$\begin{aligned}\mathbb{P}\left(N_{t+\mathrm{d}t} - N_t = 1; \text{this spike is kept}\right) &= \mathbb{P}\left(N_{t+\mathrm{d}t} - N_t = 1\right) \times \mathbb{P}(\mathcal{U}([0;1]) \leq \lambda(t)/\bar{\lambda}) \\ &= \bar{\lambda}\,\mathrm{d}t \times \lambda(t)/\bar{\lambda},\end{aligned}$$

and the other term is negligible at this scale

$$\mathbb{P}(N_{t+\mathrm{d}t} - N_t \geq 2; \text{all spikes but one removed}) \leq \mathbb{P}(N_{t+\mathrm{d}t} - N_t \geq 2) = o(\mathrm{d}t).$$

As such, since a differential value is only the first order term,

$$\mathbb{P}(\tilde{N}_{t+\mathrm{d}t} - \tilde{N}_t = 1) = \lambda(t)\,\mathrm{d}t.$$

This, along with independence of the increments - a property of $(\tilde{N}_t)$ directly inherited from (N_t) - and $\mathbb{P}(\tilde{N}_{t+s} - \tilde{N}_t \geq 2) \leq \mathbb{P}(N_{t+s} - N_t \geq 2) = o(s)$ characterises a nonhomogeneous Poisson process [Pasupathy, 2011]. This semi-rigorous proof (lacking filtrations and such measure theory) can be made entirely rigorous and sample path-dependant. [Daley and Vere-Jones, 2003]. It is verified in figure 12.2 that the averaged histogram of the events converge to the rate.

Expectation of the ISI distribution by thinning

The thinning method for a constant Poisson parameter λ generates exponential variables with this parameter, and we obtain the binning by keeping its ceiling, or equivalently the integer part of the obtained real value and adding one (it is equivalent because an exponential variable has no atom). Assuming that E follows an exponential variable, we can calculate the probability that the ISI is $k \in \mathbb{N}^*$, and from it derive the expectation of the mean ISI:

$$\mathbb{P}(\mathrm{ISI}_t = k) = \mathbb{P}(E \in [k-1, k[) = \int_{k-1}^{k} \lambda \mathrm{e}^{-\lambda t}\,\mathrm{d}t = \mathrm{e}^{-\lambda(k-1)}(1 - \mathrm{e}^{-\lambda})$$

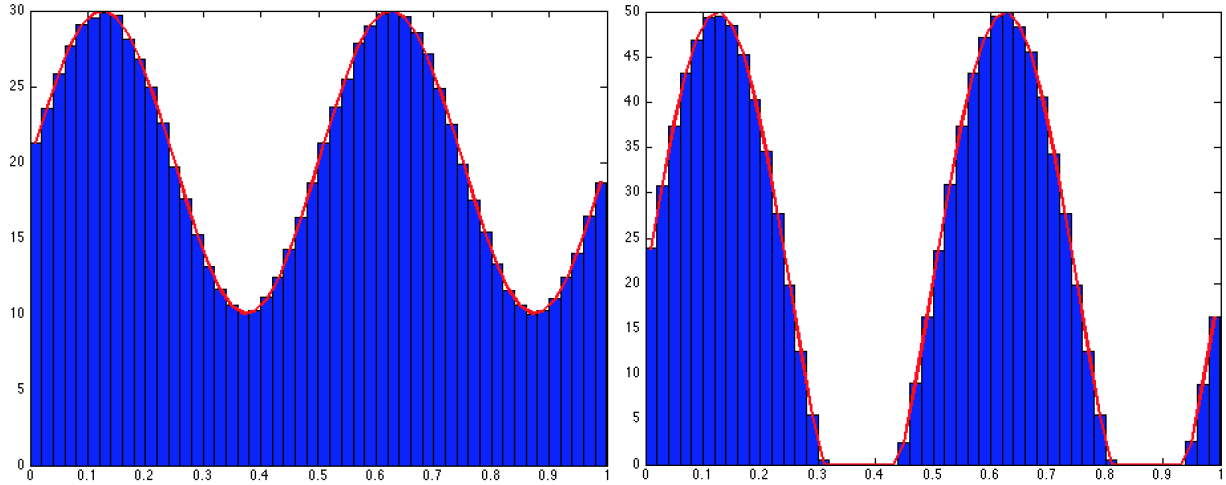


Fig. 12.2 Renormalised histograms for two chosen rates ($\lambda(t) = 20 + 10\sin(4\pi t)$ and $\lambda(t) = 20 + 30\sin(4\pi t)$, respectively) to verify over 50000 repetitions that the thinning algorithm was simulating the correct rate (red curve). The higher the rate, the faster the convergence of the averaged histogram towards the fluctuating rate.

for which we can calculate the expectation:

$$\forall \lambda > 0, \qquad f_t(\lambda) := \mathbb{E}(\mathrm{ISI}_t(\lambda)) = \frac{1}{1 - \mathrm{e}^{-\lambda}}, \tag{12.9}$$

which is numerically verified in figure 12.3. This value overshoots the real expectation λ^{-1} (12.7) because of the ceiling effect of this procedure: if multiple spikes were allowed within a bin, this effect would disappear. In practice however, we do not need to do anything: the refractoriness removes all issues that this overestimation might have. As seen on figure 12.3, when simulating the spike trains with a refractory effect (uniform between 0.75 ms and 1.5 ms), the binning method and the thinning method are in agreement.

12.2.1.3 Discussion of the methods

As shown in figure 12.3, the statistics for small rates are very similar, depart when the rate gets close to one, and of course is constant for the binning method for $\lambda \geq 1$ since a spike occurs in each bin while it converges nicely for the binning method. We recommend the thinning method, as it is exact and is faster for small values of the rate (it does not thin out too many points if the input verifies certain properties [Ross, 2006; Cox and Isham, 1980], as shown in figure 12.4).

12.2.2 Law of a Poisson Process with Random Refractoriness

Refractoriness refers to the latency between two consecutive action potentials. An absolute refractory period first occurs, during which a neuron simply won't emit another action po-

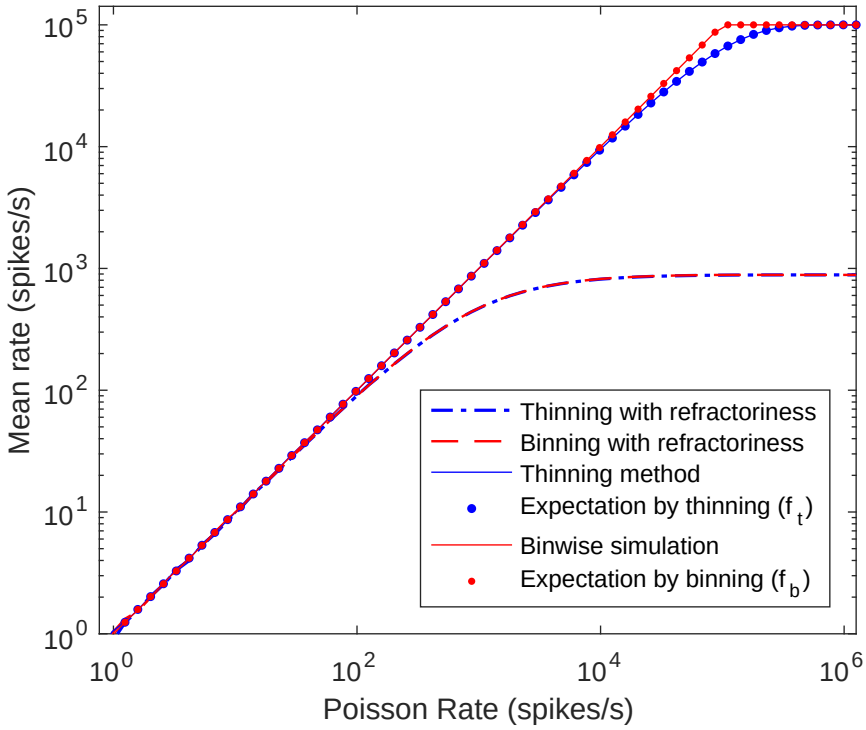


Fig. 12.3 Average rates (in number of spikes per second) when simulating a Poisson process with constant parameter λ (x-axis), and the associated theoretical expectations f_t and f_b (see formulae 12.8 and 12.9; they were inverted and multiplied by the sampling rate 1e-5). The mean rate obtained when applying the refractoriness ($\mathcal{U}([0.75, 1.5])$ ms) with each method is also given; both converge to 1e5/112.5 = 888.89 spikes per seconds, 112.5e-5 s being the mean refractory period - which would be the only cause for the ISI if the rate was infinite.

tential. After this, a relative refractory period happens, during which a neuron will require a stronger input than it normally does to be able to spike. After this time (3 ms at most), the neuron is back to its normal excitability.

This refractory period is modelled the following way: the refractory period is modelled as a uniform variable between 0 and 0.75 ms, adding up to an absolute refractory period of 0.75 ms. After generating the inhomogeneous Poisson process $(t_1, t_2, \dots)$ and reading the spike train in time, at each spike time t_i, we generate a random refractory period r_i (r_i between drawn as a uniform variable between 0.75 ms and 1.5 ms), and remove all spikes happening between time t_i and $t_i + r_i$. This is summarised in the pseudo-code 2.

Applying a refractoriness to this Poisson process has an effect on the average presence of spikes: instead of converging to the rate $\lambda(t)$, a renormalised histogram of the spike trains with refractoriness should converge to another law related to λ. The formula (3) from [Meddis and Hewitt, 1991] was corrected, which gives for discrete times $n\delta$, for $n = 1, 2, \dots,$

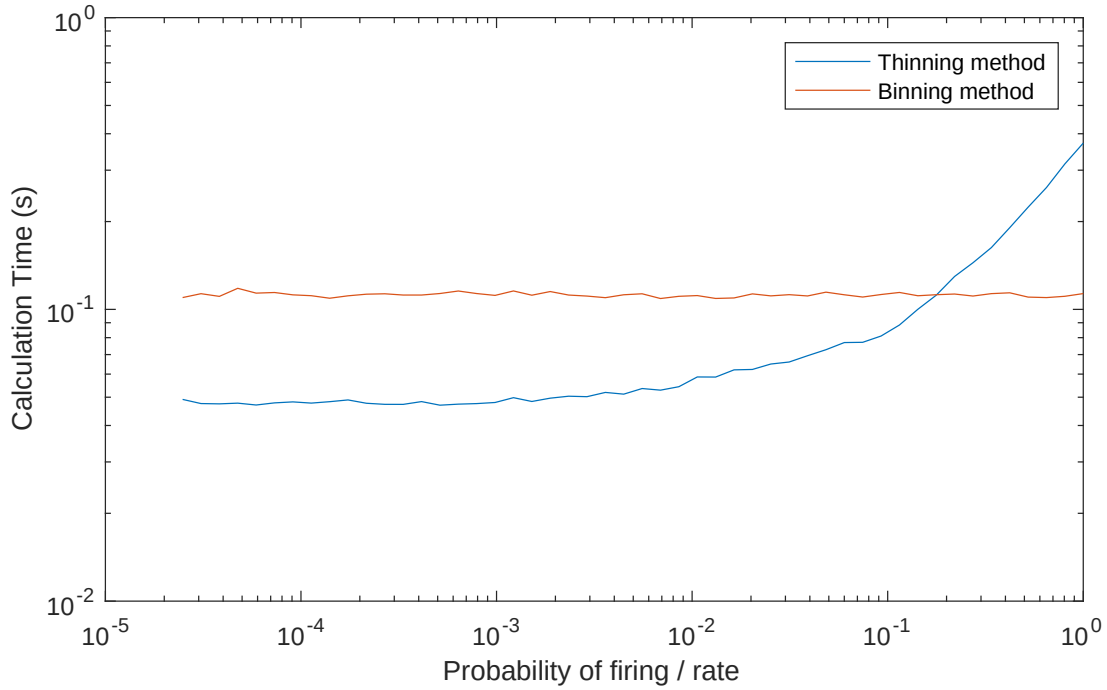


Fig. 12.4 Comparison of calculation time to simulate a Poisson process of fixed parameter (in abscissa) without refractoriness. In practice, times were averaged over 5 repetitions, generating a 10 millions-long array, using a mex file.

starting by initialising at the intial time by $\lambda_{ref}(\delta) = \lambda(\delta)$ and

$$\forall n \geq 1, \qquad \lambda_{ref}((n+1)\delta) = \lambda((n+1)\delta)\left(1 - \sum_{i=1}^{n} \lambda_{ref}((n-i)\delta)(1 - W(i\delta))\right), \quad (12.10)$$

where W is the cumulative distribution of the refractory period, which is in our case, in continuous time t (in ms)

$$\mathrm{W(t)} = \begin{cases} 0 & \text{if } < 0.75, \\ (t - 0.75)/0.75 & \text{if } 0.75 \leq t < 1.5, \\ 1 & \text{if } t > 1.5, \end{cases} \quad (12.11)$$

and we checked the convergence of a 10000 spike trains during a window of 0.01s; see figure 12.5.

Algorithm 2 Generating an Inhomogeneous Poisson Process with Uniform Refractoriness

procedure INHOMOGENEOUSPOISSONPROCESS
  $\bar{\lambda} \leftarrow \max(\lambda)$ % Assumed finite
  $T \leftarrow -\log(\text{rand}())/\bar{\lambda}$ % Generating an exponential($\bar{\lambda}$) random variable
  $n \leftarrow 0$
  while $T < T_{max}$ **do**
    if rand() $< \lambda(T)/\bar{\lambda}$ **then** % Thinning out spikes
      $n \leftarrow n + 1$
      $t_n \leftarrow T$ % Sequence of spike timing
    $T \leftarrow T - \log(\text{rand}())/\bar{\lambda}$ % Adding an exponential($\bar{\lambda}$) random variable
procedure RECFRACTORINESS
  $T \leftarrow 0$
  $\tilde{t}_1 \leftarrow t_1$
  $n \leftarrow 1$
  while Some spikes haven't been reached **do**
    $r \leftarrow \text{uniform}(0.75, 1.5)$ % Uniform variable between 0.75 and 1.5 ms
    $\tilde{t}_{n+1} \leftarrow \inf\{t_i | t_i > \tilde{t}_n + r\}$ % Sub-sequence of spike timing
    $n \leftarrow n + 1$

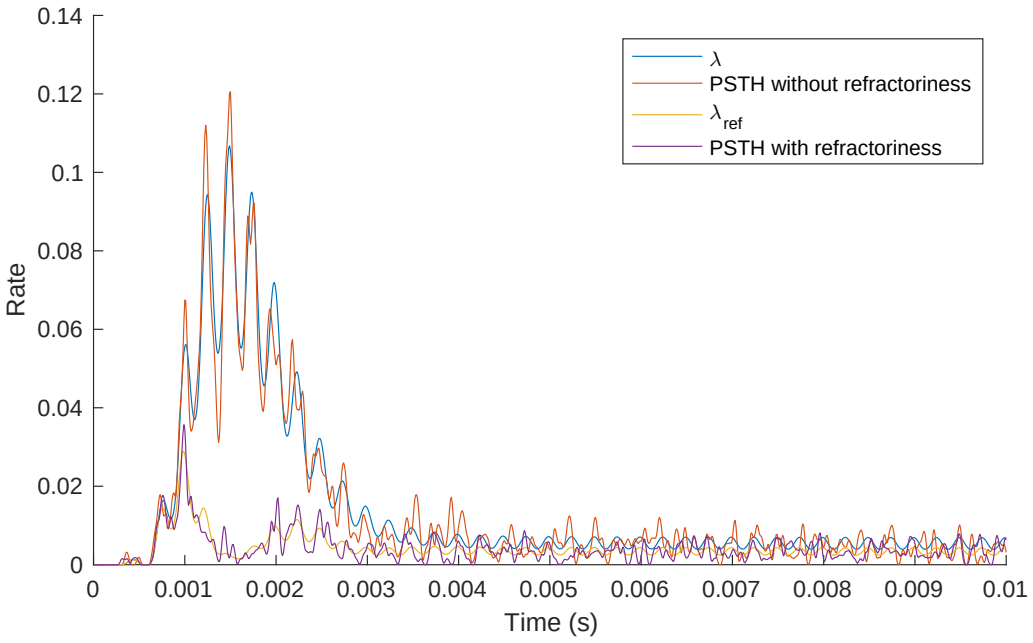


Fig. 12.5 Comparison of the rate λ and the PSTH of many spike trains generated from it, and the same with refractoriness, either encoded in the spike train simulation (using algorithm 2) or inferring the limit law from the initial probability (using formula 12.10).

12.2.3 Limitation Using the Thinning Method

In our implementation of the thinning algorithm, a Poisson process with fixed parameter $\bar{\lambda} > 0$ is first generated, simulating its interevent intervals as a sequence of exponential variables of parameter $\bar{\lambda}$, by using the formula 12.6. These increments are obtained as

$$E = -\ln(\text{rand}())/\bar{\lambda}, \tag{12.12}$$

where rand() is a sampled uniform variable between 0 and 1. To simulate the uniform distribution, we use the C function rand(), which outputs an integer uniformly distributed between 0 and `RAND_MAX`, this value being 2147483647 on our computer. To avoid the difficulties of dealing with $\log(0)$[3], the uniform distribution was simulated as

$$U = \frac{\text{rand}() + 1.0}{\text{RAND\_MAX} + 1.0},$$

giving us a uniform variable varying between 1/2147483647 and 1, thus giving exponential variables varying between 0 and $E_{max} = -\ln(1/2147483647)/\bar{\lambda}$. For $\bar{\lambda} = 0.1s^{-1}$, this approximates as $E_{max} \approx 214.89$ seconds. The probability to have higher ISIs is fairly small

$$\mathbb{P}(Exp(0.1) > 214.89) = e^{-0.1*214.89} \approx 4.65 * 10^{-10},$$

thus justifying this approximation.

12.3 An Alternative View of Supervised Learning

This subsection develops a view on evaluating Supervised Learning framework that reduces relying on given labels as ground truth.

Let's assume we are given the classes of a test set $\{x_1, \ldots, x_n\}$ guessed by two classifiers $\mathcal{C}_1$ and $\mathcal{C}_2$, where each x_i has a real labeled l_i, knowing that the labels belong to the set $\mathbb{L} = \{1, \ldots, n_l\}$. Let's add a third, abstract classifier $\mathcal{C}_T$ that outputs the real labels of the test instances, meaning that for all k, the classifier $\mathcal{C}_T$, when tested on x_k, would output l_k^T, with $l_k^T = l_k$, the real class of x_k. This means we have the table 12.6 of output labels from which we can compute the three following matrices, where i and j belong to $\{1, \ldots, n_l\}$[4]:

[3] The column values where a spike train occurs where obtained as

col = (int) expo with expo = (double) -log(U)/L,

the maximal λ being saved in L and U being a uniform sample between 0 and 1 detailed above. Allowing $U = 0$ and hence expo $= +\infty$ can be problematic since '(int)inf' is not defined in C and we'd obtain the negative number col $= -2147483648$.

[4] We use a logical writing similar to Matlab's to define the matrices' coefficients by convenience. Equivalently: $C_1(i,j) := \sum_{k=1}^{n} \delta(l_k^T - i) * \delta(l_k^1 - j)$.

- The confusion matrix for the classifier $\mathcal{C}_1$: $C_1(i,j) := \sum_{k=1}^{n}(l_k^T == i \ \& \ l_k^1 == j)$;
- The confusion matrix for the classifier $\mathcal{C}_2$: $C_2(i,j) := \sum_{k=1}^{n}(l_k^T == i \ \& \ l_k^2 == j)$;
- The agreement matrix between $\mathcal{C}_1$ and $\mathcal{C}_2$: $C_{1,2}(i,j) := \sum_{k=1}^{n}(l_k^2 == i \ \& \ l_k^1 == j)$.

The confusion matrices are computed with respect to the real labels of the data, whereas the agreement matrix is not. If we think of the real labels as the guesses by a classifier that we may call the true classifier that is always right (or, more weakly, that we completely trust), then the confusion matrices are simply the agreement matrices of each classifier against the true classifier. Thus any mathematical theory of the measures on confusion matrices should be thought of as working on agreement matrices for the following reasons:

- If we know the real classes, the theory applies to confusion matrices and we work in a supervised learning framework;
- If we do not know the classes, we can still run the evaluations and compare classifiers, in a semi-supervised learning framework[5].

This view simply explains that what we gain by measuring classifiers one against another is some generality in the framework, which is of utmost interest in general because most real data does not naturally fall in naturally separated classes.

Data	Estimated class by $\mathcal{C}_1$	Estimated class by $\mathcal{C}_2$	Estimated class by $\mathcal{C}_T$
x_1	l_1^1	l_1^2	l_1^T
x_2	l_2^1	l_2^2	l_2^T
x_3	l_3^1	l_3^2	l_3^T
$\vdots$	$\vdots$	$\vdots$	$\vdots$
x_n	l_n^1	l_n^2	l_n^T

Tab. 12.6 Labels output by our three classifiers $\mathcal{C}_1$, $\mathcal{C}_2$ and $\mathcal{C}_T$ when applied to the test set.

[5]This is not properly speaking a semi-supervised learning framework but can become so: If we train some classifiers on labeled data, we can accept some predictions as long as all classifiers agree, thus extending the labeled data as long as it seems legitimate to do so.

Index

Bibliography

Daniel Arthur Abrams. *Temporal features of speech in the auditory system.* PhD thesis, 2008.

M. S. Alam, M. S. A. Zilany, W. A. Jassim, and M. Y. Ahmad. Phoneme Classification Using the Auditory Neurogram. *IEEE Access*, 5:633–642, 2017.

Shawkat Ali and Kate A. Smith. On learning algorithm selection for classification. *Applied Soft Computing*, 6(2):119–138, January 2006.

ANSI. ANSI/ASA S3.5 : American National Standard Methods for Calculation of the Speech Intelligibility Index, 1997.

Robert H. Arnott, Mark N. Wallace, Trevor M. Shackleton, and Alan R. Palmer. Onset neurones in the anteroventral cochlear nucleus project to the dorsal cochlear nucleus. *Journal of the Association for Research in Otolaryngology*, 5(2):153–170, 2004.

S. P. Bacon and N. F. Viemeister. Temporal modulation transfer functions in normal-hearing and hearing-impaired listeners. *Audiology: Official Organ of the International Society of Audiology*, 24(2):117–134, 1985.

M. I. Banks and M. B. Sachs. Regularity analysis in a compartmental model of chopper units in the anteroventral cochlear nucleus. *Journal of Neurophysiology*, 65(3):606–629, March 1991.

Jon Barker, Emmanuel Vincent, Ning Ma, Heidi Christensen, and Phil Green. The PASCAL CHiME speech separation and recognition challenge. *Computer Speech & Language*, 27(3):621–633, May 2013.

Christopher M. Bishop. *Neural Networks for Pattern Recognition.* Clarendon Press, November 1995.

C. C. Blackburn and M. B. Sachs. Classification of unit types in the anteroventral cochlear nucleus: PST histograms and regularity analysis. *Journal of Neurophysiology*, 62(6):1303–1329, December 1989.

K. E. Bouchard and E. F. Chang. Neural decoding of spoken vowels from human sensory-motor cortex with high-density electrocorticography. In *2014 36th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pages 6782–6785, August 2014.

Kristofer E. Bouchard, Nima Mesgarani, Keith Johnson, and Edward F. Chang. Functional organization of human sensorimotor cortex for speech articulation. *Nature*, 495(7441):327–332, March 2013.

A.-L. Boulesteix, C. Strobl, T. Augustin, and M. Daumer. Evaluating Microarray-based Classifiers: An Overview. *Cancer Informatics*, 6:77–97, February 2008.

Michael Bowling, Neil Burch, Michael Johanson, and Oskari Tammelin. Heads-up limit holdem poker is solved. *Science*, 347(6218):145–149, January 2015.

Leo Breiman. Bagging Predictors. *Machine Learning*, 24(2):123–140, August 1996.

Guy J. Brown, Robert T. Ferry, and Ray Meddis. A computer model of auditory efferent suppression: Implications for the recognition of speech in noise. *The Journal of the Acoustical Society of America*, 127(2):943–954, February 2010.

Jonathan S. Brumberg, E. Joe Wright, Dinal S. Andreasen, Frank H. Guenther, and Philip R. Kennedy. Classification of intended phoneme production from chronic intracortical microelectrode recordings in speech-motor cortex. *Frontiers in Neuroscience*, 5:65, 2011.

Peter Brunner, Karen Dijkstra, William G. Coon, Jrgen Mellinger, Anthony L. Ritaccio, and Gerwin Schalk. An ECoG-Based BCI Based on Auditory Attention to Natural Speech. In *Brain-Computer Interface Research*, SpringerBriefs in Electrical and Computer Engineering, pages 7–19. Springer, Cham, 2017. DOI: 10.1007/978-3-319-57132-4_2.

Daniel A. Butts, Chong Weng, Jianzhong Jin, Chun-I. Yeh, Nicholas A. Lesica, Jose-Manuel Alonso, and Garrett B. Stanley. Temporal precision in the neural code and the timescales of natural vision. *Nature*, 449(7158):92–95, September 2007.

Steven M. Chase and Eric D. Young. Spike-Timing Codes Enhance the Representation of Multiple Simultaneous Sound-Localization Cues in the Inferior Colliculus. *The Journal of Neuroscience*, 26(15):3889–3898, April 2006.

Lars Chittka and Axel Brockmann. Perception SpaceThe Final Frontier. *PLoS Biol*, 3(4):e137, April 2005.

Yong-Sun Choi. Nonlinear spectro-temporal features based on a cochlear model for automatic speech recognition in a noisy situation. *ResearchGate*, 2013.

Nicholas R. Clark, Guy J. Brown, Tim Jrgens, and Ray Meddis. A frequency-selective feedback model of auditory efferent suppression and its implications for the recognition of speech in noise. *The Journal of the Acoustical Society of America*, 132(3):1535–1541, September 2012.

Martin Coath, Sadique Sheik, Elisabetta Chicca, Giacomo Indiveri, Susan Denham, and Thomas Wennekers. A robust sound perception model suitable for neuromorphic implementation. *Neuromorphic Engineering*, 7:278, 2014.

Thomas Cover and Joy Thomas. Elements of Information Theory, 2nd Edition, 2006.

D. R. Cox and Valerie Isham. *Point Processes*. CRC Press, July 1980. Google-Books-ID: KWF2xY6s3PoC.

G. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems*, 2(4):303–314, 1989.

Daryl J. Daley and D. (David) Vere-Jones. *An introduction to the theory of point processes / D.J. Daley, D. Vere-Jones*. Springer, New York, 2003. Includes bibliographical references and index.

S. Davis and P. Mermelstein. Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 28(4):357–366, August 1980.

Kevin A. Davis. Contralateral Effects and Binaural Interactions in Dorsal Cochlear Nucleus. *JARO: Journal of the Association for Research in Otolaryngology*, 6(3):280–296, September 2005.

Janez Demar. Statistical Comparisons of Classifiers over Multiple Data Sets. *J. Mach. Learn. Res.*, 7:1–30, December 2006.

Luc Devroye, Lszl Gyrfi, and Gabor Lugosi. *A Probabilistic Theory of Pattern Recognition*. Springer Science & Business Media, February 1997. Google-Books-ID: uDgXoRkyWqQC.

Thomas G. Dietterich and Ghulum Bakiri. Solving Multiclass Learning Problems via Error-correcting Output Codes. *J. Artif. Int. Res.*, 2(1):263–286, January 1995.

A. V. Dorugade. New ridge parameters for ridge regression. *Journal of the Association of Arab Universities for Basic and Applied Sciences*, 15:94–99, April 2014.

Mounya Elhilali, Jonathan B. Fritz, David J. Klein, Jonathan Z. Simon, and Shihab A. Shamma. Dynamics of Precise Spike Timing in Primary Auditory Cortex. *Journal of Neuroscience*, 24(5):1159–1172, February 2004.

E. F. Evans. The frequency response and other properties of single fibres in the guinea-pig cochlear nerve. *The Journal of Physiology*, 226(1):263–287, October 1972.

E. F. Evans. Auditory Processing of Complex Sounds: An Overview. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 336(1278):295–306, June 1992.

Richard R. Fay and Laura Ann Wilber. Hearing in Vertebrates: A Psychophysics Databook. *The Journal of the Acoustical Society of America*, 86(5):2044–2044, November 1989.

Santiago Fernandez, Alex Graves, and Jrgen Schmidhuber. An application of recurrent neural networks to discriminative keyword spotting. 2007.

Ronan Flynn and Edward Jones. Combined speech enhancement and auditory modelling for robust distributed speech recognition. *Speech Communication*, 50(10):797–809, October 2008.

Robert D Frisina, Robert L Smith, and Steven C Chamberlain. Encoding of amplitude modulation in the gerbil cochlear nucleus: I. A hierarchy of enhancement. *Hearing Research*, 44(2):99–122, March 1990.

Adam C. Furman, Sharon G. Kujawa, and M. Charles Liberman. Noise-induced cochlear neuropathy is selective for fibers with low spontaneous rates. *Journal of Neurophysiology*, 110(3):577–586, August 2013.

Christian Gaida, Patrick Lange, Rico Petrick, Patrick Proba, Ahmed Malatawy, and David Suendermann-Oeft. Comparing Open-Source Speech Recognition Toolkits. 2014.

Mark Gales and Steve Young. The Application of Hidden Markov Models in Speech Recognition. *Found. Trends Signal Process.*, 1(3):195–304, January 2007.

John Garofolo, Lori Lamel, William Fisher, Jonathan Fiscus, David Pallett, Nancy Dahlgren, and Victor Zue. TIMIT Acoustic-Phonetic Continuous Speech Corpus, 1993.

W. S. Geisler, D. G. Albrecht, R. J. Salvi, and S. S. Saunders. Discrimination performance of single neurons: rate and temporal-pattern information. *Journal of Neurophysiology*, 66(1):334–362, July 1991.

Oded Ghitza. Auditory nerve representation as a front-end for speech recognition in a noisy environment. *Computer Speech & Language*, 1(2):109–130, December 1986.

J. M. Goldberg and P. B. Brown. Response of binaural neurons of dog superior olivary complex to dichotic tonal stimuli: some physiological mechanisms of sound localization. *Journal of Neurophysiology*, 32(4):613–636, July 1969.

Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. Speech Recognition with Deep Recurrent Neural Networks. *arXiv:1303.5778 [cs]*, March 2013. arXiv: 1303.5778.

Robert Gutig and Haim Sompolinsky. Time-Warp-Invariant Neuronal Processing. *PLoS Biology*, 7(7), July 2009.

Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. The WEKA Data Mining Software: An Update. *SIGKDD Explor. Newsl.*, 11(1):10–18, November 2009.

David J. Hand and Robert J. Till. A Simple Generalisation of the Area Under the ROC Curve for Multiple Class Classification Problems. *Machine Learning*, 45(2):171–186, November 2001.

Trevor Hastie and Robert Tibshirani. Classification by Pairwise Coupling. In *Proceedings of the 1997 Conference on Advances in Neural Information Processing Systems 10*, NIPS '97, pages 507–513, Cambridge, MA, USA, 1998. MIT Press.

J. Leo van Hemmen. Vector strength after Goldberg, Brown, and von Mises: biological and mathematical perspectives. *Biological Cybernetics*, 107(4):385–396, August 2013.

Christian Herff and Tanja Schultz. Automatic Speech Recognition from Neural Signals: A Focused Review. *Frontiers in Neuroscience*, 10, September 2016.

M. J. Hewitt, R. Meddis, and T. M. Shackleton. A computer model of a cochlear-nucleus stellate cell: responses to amplitude-modulated and pure-tone stimuli. *The Journal of the Acoustical Society of America*, 91(4 Pt 1):2096–2109, April 1992.

G. Hinton, Li Deng, Dong Yu, G.E. Dahl, A. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T.N. Sainath, and B. Kingsbury. Deep Neural Networks for Acoustic Modeling in Speech Recognition: The Shared Views of Four Research Groups. *IEEE Signal Processing Magazine*, 29(6):82–97, November 2012.

Sepp Hochreiter and Jrgen Schmidhuber. Long Short-Term Memory. *Neural Comput.*, 9(8):1735–1780, November 1997.

Marcus Holmberg, David Gelbart, and Werner Hemmert. Speech encoding in a model of peripheral auditory processing: Quantitative assessment by means of automatic speech recognition. *Speech Communication*, 49(12):917–932, December 2007.

Marcus Holmberg. Automatic speech recognition with neural spike trains (PDF Download Available). In *ResearchGate*, 2005.

A. J. Hudspeth and D. P. Corey. Sensitivity, polarity, and conductance change in the response of vertebrate hair cells to controlled mechanical stimuli. *Proceedings of the National Academy of Sciences*, 74(6):2407–2411, June 1977.

Eugene M Izhikevich. *Dynamical systems in neuroscience: the geometry of excitability and bursting*. MIT Press, Cambridge, Mass., 2007. OCLC: 65400606.

Gareth James and Trevor Hastie. *Error Coding and PaCT's*. 1997.

Skyler G. Jennings, Michael G. Heinz, and Elizabeth A. Strickland. Evaluating Adaptation and Olivocochlear Efferent Feedback as Potential Explanations of Psychophysical Overshoot. *JARO: Journal of the Association for Research in Otolaryngology*, 12(3):345–360, June 2011.

Woojay Jeon and B.-H. Juang. Speech Analysis in a Model of the Central Auditory System. *IEEE Transactions on Audio, Speech, and Language Processing*, 15(6):1802–1817, August 2007.

Ian T. Jolliffe. A Note on the Use of Principal Components in Regression. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 31(3):300–303, 1982.

P. X. Joris and P. H. Smith. The volley theory and the spherical cell puzzle. *Neuroscience*, 154(1):65–76, June 2008.

P. X. Joris and T. C. Yin. Responses to amplitude-modulated tones in the auditory nerve of the cat. *The Journal of the Acoustical Society of America*, 91(1):215–232, January 1992.

P. X. Joris, C. E. Schreiner, and A. Rees. Neural processing of amplitude-modulated sounds. *Physiological Reviews*, 84(2):541–577, April 2004.

Norman P. Jouppi, Cliff Young, Nishant Patil, David Patterson, Gaurav Agrawal, Raminder Bajwa, Sarah Bates, Suresh Bhatia, Nan Boden, Al Borchers, Rick Boyle, Pierre-luc Cantin, Clifford Chao, Chris Clark, Jeremy Coriell, Mike Daley, Matt Dau, Jeffrey Dean, Ben Gelb, Tara Vazir Ghaemmaghami, Rajendra Gottipati, William Gulland, Robert Hagmann, C. Richard Ho, Doug Hogberg, John Hu, Robert Hundt, Dan Hurt, Julian Ibarz, Aaron Jaffey, Alek Jaworski, Alexander Kaplan, Harshit Khaitan, Andy Koch, Naveen Kumar, Steve Lacy, James Laudon, James Law, Diemthu Le, Chris Leary, Zhuyuan Liu, Kyle Lucke, Alan Lundin, Gordon MacKean, Adriana Maggiore, Maire Mahony, Kieran Miller, Rahul Nagarajan, Ravi Narayanaswami, Ray Ni, Kathy Nix, Thomas Norrie, Mark Omernick, Narayana Penukonda, Andy Phelps, Jonathan Ross, Matt Ross, Amir Salek, Emad Samadiani, Chris Severn, Gregory Sizikov, Matthew Snelham, Jed Souter, Dan Steinberg, Andy Swing, Mercedes Tan, Gregory Thorson, Bo Tian, Horia Toma, Erick Tuttle, Vijay Vasudevan, Richard Walter, Walter Wang, Eric Wilcox, and Doe Hyun Yoon. In-Datacenter Performance Analysis of a Tensor Processing Unit. *arXiv:1704.04760 [cs]*, April 2017. arXiv: 1704.04760.

Eric Kandel. *Principles of Neural Science, Fifth Edition*. McGraw Hill Professional, 2013.

Igor Kononenko and Ivan Bratko. Information-Based Evaluation Criterion for Classifier's Performance. *Machine Learning*, 6(1):67–80, January 1991.

S. B. Kotsiantis, I. D. Zaharakis, and P. E. Pintelas. Machine learning: a review of classification and combining techniques. *Artificial Intelligence Review*, 26(3):159–190, November 2007.

Thomas Kreuz, Mario Mulansky, and Nebojsa Bozanic. SPIKY: A graphical user interface for monitoring spike train synchrony. *arXiv:1410.6910 [physics, q-bio]*, October 2014. arXiv: 1410.6910.

Yoshiki Kuramoto. Self-entrainment of a population of coupled non-linear oscillators. In Prof Huzihiro Araki, editor, *International Symposium on Mathematical Problems in Theoretical Physics*, number 39 in Lecture Notes in Physics, pages 420–422. Springer Berlin Heidelberg, 1975.

Jonathan Laudanski, Stephen Coombes, Alan R. Palmer, and Christian J. Sumner. Mode-Locked Spike Trains in Responses of Ventral Cochlear Nucleus Chopper and Onset Neurons to Periodic Stimuli. *Journal of Neurophysiology*, 103(3):1226–1237, March 2010.

Quoc V. Le, Marc'Aurelio Ranzato, Rajat Monga, Matthieu Devin, Kai Chen, Greg S. Corrado, Jeff Dean, and Andrew Y. Ng. Building high-level features using large scale unsupervised learning. *arXiv:1112.6209 [cs]*, December 2011. arXiv: 1112.6209.

K. F. Lee and H. W. Hon. Speaker-independent phone recognition using hidden Markov models. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 37(11):1641–1648, November 1989.

Christoph Lehmann, Thomas Koenig, Vesna Jelic, Leslie Prichep, Roy E. John, Lars-Olof Wahlund, Yadolah Dodge, and Thomas Dierks. Application and comparison of classification algorithms for recognition of Alzheimer's disease in electrical brain activity (EEG). *Journal of Neuroscience Methods*, 161(2):342–350, April 2007.

Matthew K. Leonard, Kristofer E. Bouchard, Claire Tang, and Edward F. Chang. Dynamic Encoding of Speech Sequence Probability in Human Temporal Cortex. *The Journal of Neuroscience*, 35(18):7203–7214, May 2015.

Qi Li, Frank K. Soong, and Olivier Siohan. A High-Performance Auditory Feature For Robust Speech Recognition. *ResearchGate*, 2000.

Philip M. Long and Rocco A. Servedio. Random classification noise defeats all convex potential boosters. *Machine Learning*, 78(3):287–304, March 2010.

Carla Lopes and Fernando Perdigao. Phoneme Recognition on the TIMIT Database. 2011.

R. Lyon. A computational model of filtering, detection, and compression in the cochlea. pages 1282 – 1285, 1982.

David MacKay. Information Theory, Inference, and Learning Algorithms: Home, 2003.

B. J. May, G. S. Prell, and M. B. Sachs. Vowel representations in the ventral cochlear nucleus of the cat: effects of level, background noise, and behavioral state. *Journal of Neurophysiology*, 79(4):1755–1767, April 1998.

Ray Meddis and Michael J. Hewitt. Virtual pitch and phase sensitivity of a computer model of the auditory periphery. I: Pitch identification. *The Journal of the Acoustical Society of America*, 89(6):2866–2882, June 1991.

R. Meddis, L. P. O'Mard, and E. A. Lopez-Poveda. A computational algorithm for computing nonlinear auditory frequency selectivity. *The Journal of the Acoustical Society of America*, 109(6):2852–2861, June 2001.

Nima Mesgarani and Edward F. Chang. Selective cortical representation of attended speaker in multi-talker speech perception. *Nature*, 485(7397):233–236, May 2012.

Nima Mesgarani, Stephen V. David, Jonathan B. Fritz, and Shihab A. Shamma. Phoneme representation and classification in primary auditory cortex. *The Journal of the Acoustical Society of America*, 123(2):899–909, February 2008.

Il Joon Moon and Sung Hwa Hong. What Is Temporal Fine Structure and Why Is It Important? *Korean Journal of Audiology*, 18(1):1–7, April 2014.

Emily M. Mugler, James L. Patton, Robert D. Flint, Zachary A. Wright, Stephan U. Schuele, Joshua Rosenow, Jerry J. Shih, Dean J. Krusienski, and Marc W. Slutzky. Direct classification of all American English phonemes using signals from functional speech motor cortex. *Journal of Neural Engineering*, 11(3), 2014.

Paul A. Nakamura and Karina S. Cramer. Formation and maturation of the calyx of Held. *Hearing Research*, 276(1-2):70–78, June 2011.

Erikson G. Neilans and Micheal L. Dent. Temporal coherence for complex signals in budgerigars (Melopsittacus undulatus) and humans (Homo sapiens). *Journal of Comparative Psychology (Washington, D.C.: 1983)*, 129(2):174–180, May 2015.

Andrew Y. Ng and Michael I. Jordan. On Discriminative vs. Generative Classifiers: A comparison of logistic regression and naive Bayes. In T. G. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems 14*, pages 841–848. MIT Press, 2002.

Donata Oertel, Ramazan Bal, Stephanie M. Gardner, Philip H. Smith, and Philip X. Joris. Detection of synchrony in the activity of auditory nerve fibers by octopus cells of the mammalian cochlear nucleus. *Proceedings of the National Academy of Sciences*, 97(22):11773–11779, October 2000.

Kirsten Kjelsberg Osen. Cytoarchitecture of the cochlear nuclei in the cat. *The Journal of Comparative Neurology*, 136(4):453–483, August 1969.

A. R. Palmer and I. J. Russell. Phase-locking in the cochlear nerve of the guinea-pig and its relation to the receptor potential of inner hair-cells. *Hearing Research*, 24(1):1–15, 1986.

Athanasios Papoulis. *Probability, Random Variables, and Stochastic Processes*. McGraw-Hill, 1991. Google-Books-ID: 4IwQAQAAIAAJ.

Brian N. Pasley and Robert T. Knight. Decoding Speech for Understanding and Treating Aphasia. *Progress in brain research*, 207:435–456, 2013.

Brian N. Pasley, Stephen V. David, Nima Mesgarani, Adeen Flinker, Shihab A. Shamma, Nathan E. Crone, Robert T. Knight, and Edward F. Chang. Reconstructing Speech from Human Auditory Cortex. *PLoS Biol*, 10(1):e1001251, January 2012.

Raghu Pasupathy. *Generating Nonhomogeneous Poisson Processes*. 2011.

E. K. Patterson, S. Gurbuz, Z. Tufekci, and J. N. Gowdy. CUAVE: A new audio-visual database for multimodal human-computer interface research. In *In Proc. ICASSP*, pages 2017–2020, 2002.

James O. Pickles. *An Introduction to the Physiology of Hearing*. Academic Press, London ; San Diego, 2 edition edition, October 1988.

Christopher J. Plack. *The Sense of Hearing*. Lawrence Erlbaum Associates, 2005.

Rodrigo Quian Quiroga and Stefano Panzeri. *Principles of Neural Coding*. CRC Press, May 2013.

Lawrence R Rabiner and B. H Juang. *Fundamentals of speech recognition*. PTR Prentice Hall, Englewood Cliffs, N.J., 1993.

Carl Edward Rasmussen. Gaussian processes for machine learning. MIT Press, 2006.

William S. Rhode and Steven Greenberg. Physiology of the Cochlear Nuclei. In Arthur N. Popper and Richard R. Fay, editors, *The Mammalian Auditory Pathway: Neurophysiology*, number 2 in Springer Handbook of Auditory Research, pages 94–152. Springer New York, 1992.

W. S. Rhode and S. Greenberg. Encoding of amplitude modulation in the cochlear nucleus of the cat. *Journal of Neurophysiology*, 71(5):1797–1825, May 1994.

Sheldon M. Ross. *Simulation*. Academic Press, August 2006. Google-Books-ID: TMUt5OXVvY0C.

J. S. Rothman and P. B. Manis. The roles potassium currents play in regulating the electrical activity of ventral cochlear nucleus neurons. *Journal of Neurophysiology*, 89(6):3097–3113, June 2003.

Jason S. Rothman and Paul B. Manis. Kinetic Analyses of Three Distinct Potassium Conductances in Ventral Cochlear Nucleus Neurons. *Journal of Neurophysiology*, 89(6):3083–3096, June 2003.

J. S. Rothman, E. D. Young, and P. B. Manis. Convergence of auditory nerve fibers onto bushy cells in the ventral cochlear nucleus: implications of a computational model. *Journal of Neurophysiology*, 70(6):2562–2583, December 1993.

Haim Sak, Andrew Senior, Kanishka Rao, and Franoise Beaufays. Fast and Accurate Recurrent Neural Network Acoustic Models for Speech Recognition. *arXiv:1507.06947 [cs, stat]*, July 2015. arXiv: 1507.06947.

Adu-Poku Sampson. *Comparing classification algorithms in data mining :: CCSU Theses & Dissertations*. PhD thesis, 2012.

Marc Ren Schaedler and Birger Kollmeier. Separable spectro-temporal Gabor filter bank features: Reducing the complexity of robust features for automatic speech recognition. *The Journal of the Acoustical Society of America*, 137(4):2047–2059, April 2015.

P Schafer and D Jin. Noise-Robust Speech Recognition Through Auditory Feature Detection and Spike Sequence Decoding. *Neural Computation*, 26(3):523–556, March 2014.

Odette Scharenborg. Reaching over the gap: A review of efforts to link human and automatic speech recognition research. *Speech Communication*, 49(5):336–347, May 2007.

Jan Schnupp, Israel Nelken, and Andrew King. *Auditory Neuroscience: Making Sense of Sound*. MIT Press, 2011.

Chris Scholes, R.W. Mill, Alan R. Palmer, Stephen Coombes, W.S. Rhode, and Christian J. Sumner. Complex spike trains improve the fidelity of temporal coding of amplitude modulation in cochlear nucleus. *Submitted*, 2015.

S. Seneff. A Joint Synchrony/Mean-Rate Model of Auditory Speech Processing. *Proc. Journal of Phonetics*, 16:55–76, 1988.

Shihab A. Shamma. Speech processing in the auditory system I: The representation of speech sounds in the responses of the auditory nerve. *The Journal of the Acoustical Society of America*, 78(5):1612–1621, November 1985.

R. V. Shannon, F. G. Zeng, V. Kamath, J. Wygonski, and M. Ekelid. Speech recognition with primarily temporal cues. *Science (New York, N.Y.)*, 270(5234):303–304, October 1995.

Gordon M. Shepherd, editor. *The Synaptic Organization of the Brain*. OUP USA, fifth edition edition, January 2004.

Christopher A. Shera, John J. Guinan, and Andrew J. Oxenham. Revised estimates of human cochlear tuning from otoacoustic and behavioral measurements. *Proceedings of the National Academy of Sciences of the United States of America*, 99(5):3318–3323, March 2002.

H Shimazaki and S Shinomoto. A Method for Selecting the Bin Size of a Time Histogram. *Neural Computation*, 19(6):1503–1527, June 2007.

S. E. Shore, C. J. Sumner, S. C. Bledsoe, and J. Lu. Effects of contralateral sound stimulation on unit activity of ventral cochlear nucleus neurons. *Experimental Brain Research*, 153(4):427–435, December 2003.

Reinhard Siegmund-Schultze. Probability in 1919/20: the von Mises-Plya-Controversy. *Archive for History of Exact Sciences*, 60(5):431–515, August 2006.

David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587):484–489, January 2016.

David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, Yutian Chen, Timothy Lillicrap, Fan Hui, Laurent Sifre, George van den Driessche, Thore Graepel, and Demis Hassabis. Mastering the game of Go without human knowledge. *Nature*, 550(7676):nature24270, October 2017.

Jonathan Z. Simon. Human Auditory Neuroscience and the Cocktail Party Problem. In *The Auditory System at the Cocktail Party*, Springer Handbook of Auditory Research, pages 169–197. Springer, Cham, 2017. DOI: 10.1007/978-3-319-51662-2_7.

Ranganatha Sitaram, Tomas Ros, Luke Stoeckel, Sven Haller, Frank Scharnowski, Jarrod Lewis-Peacock, Nikolaus Weiskopf, Maria Laura Blefari, Mohit Rana, Ethan Oblak, Niels

Birbaumer, and James Sulzer. Closed-loop brain training: the science of neurofeedback. *Nature Reviews Neuroscience*, 18(2):86–100, February 2017.

A. J. Smith, H. Blumenfeld, K. L. Behar, D. L. Rothman, R. G. Shulman, and F. Hyder. Cerebral energetics and spiking frequency: The neurophysiological basis of fMRI. *Proceedings of the National Academy of Sciences*, 99(16):10765–10770, August 2002.

Kent A. Spackman. Signal Detection Theory: Valuable Tools for Evaluating Inductive Learning. In *Proceedings of the Sixth International Workshop on Machine Learning*, pages 160–163, San Francisco, CA, USA, 1989. Morgan Kaufmann Publishers Inc.

Mark Steadman. Investigating the neural code for dynamic speech and the effects of signal degradation. January 2015.

Richard M. Stern. Applying physiologically-motivated models of auditory processing to automatic speech recognition. 2011.

Christian J. Sumner, Lowel P. O'Mard, Enrique A. Lopez-Poveda, and Ray Meddis. A nonlinear filter-bank model of the guinea-pig cochlear nerve: rate responses. *The Journal of the Acoustical Society of America*, 113(6):3264–3274, June 2003.

Amirhossein Tavanaei and Anthony S. Maida. A Spiking Network that Learns to Extract Spike Signatures from Speech Signals. *arXiv:1606.00802 [cs]*, June 2016. arXiv: 1606.00802.

Sergios Theodoridis and Konstantinos Koutroumbas. *Pattern Recognition, Third Edition*. Academic Press, March 2006.

Jonathan D Victor and Keith P Purpura. Metric-space analysis of spike trains: theory, algorithms and application. *Network: Computation in Neural Systems*, 8(2):127–164, January 1997.

Richard von Mises. ber die 'Ganzzahligkeit' der Atomgewichte und verwandte Fragen. *Physikalische Zeitschrift*, 19:476–481, 1928.

X. Wang and M. B. Sachs. Transformation of temporal discharge patterns in a ventral cochlear nucleus stellate cell model: implications for physiological mechanisms. *Journal of Neurophysiology*, 73(4):1600–1616, April 1995.

Kuansan Wang and S. A. Shamma. Spectral shape analysis in the central auditory system. *IEEE Transactions on Speech and Audio Processing*, 3(5):382–395, September 1995.

David Williams. *Probability with Martingales (Cambridge Mathematical Textbooks)*. Cambridge University Press, February 1991.

I. M. Winter and A. R. Palmer. Intensity coding in low-frequency auditory-nerve fibers of the guinea pig. *The Journal of the Acoustical Society of America*, 90(4 Pt 1):1958–1967, October 1991.

Sandra Wohlgemuth and Bernhard Ronacher. Auditory discrimination of amplitude modulations based on metric distances of spike trains. *Journal of Neurophysiology*, 97(4):3082–3092, April 2007.

W. Xiong, J. Droppo, X. Huang, F. Seide, M. Seltzer, A. Stolcke, D. Yu, and G. Zweig. Achieving Human Parity in Conversational Speech Recognition. *arXiv:1610.05256 [cs]*, October 2016. arXiv: 1610.05256.

X. Yang, K. Wang, and S. A. Shamma. Auditory representations of acoustic signals. *IEEE Transactions on Information Theory*, 38(2):824–839, March 1992.

Izzet B. Yildiz, Katharina von Kriegstein, and Stefan J. Kiebel. From Birdsong to Human Speech Recognition: Bayesian Inference on a Hierarchy of Nonlinear Dynamical Systems. *PLoS Computational Biology*, 9(9), September 2013.

Eric D. Young and Murray B. Sachs. Representation of steadystate vowels in the temporal aspects of the discharge patterns of populations of auditorynerve fibers. *The Journal of the Acoustical Society of America*, 66(5):1381–1403, November 1979.

SJ Young, G Evermann, MJF Gales, T Hain, D Kershaw, G Moore, J Odell, D Ollason, D Povey, V Valtchev, and PC Woodland. *The HTK Book, version 3.4*. Cambridge University Engineering Department, 2006.

Eric D. Young. Neural representation of spectral and temporal information in speech. *Philosophical Transactions of the Royal Society of London. Series B, Biological Sciences*, 363(1493):923–945, March 2008.

X. Zhang, M. G. Heinz, I. C. Bruce, and L. H. Carney. A phenomenological model for the responses of auditory-nerve fibers: I. Nonlinear tuning with compression and suppression. *The Journal of the Acoustical Society of America*, 109(2):648–670, February 2001.

Muhammad S. A. Zilany, Ian C. Bruce, Paul C. Nelson, and Laurel H. Carney. A phenomenological model of the synapse between the inner hair cell and auditory nerve: long-term adaptation with power-law dynamics. *The Journal of the Acoustical Society of America*, 126(5):2390–2412, November 2009.