

MARGIN LEARNING IN SPIKING NEURAL NETWORKS

Dissertation
for the award of the degree
"Doctor rerum naturalium"
of the Georg-August-Universität Göttingen

within the doctoral program
Theoretical and Computational Neuroscience
of the Georg-August University School of Science (GAUSS)

submitted by
Rafael Brune

from Göttingen, Germany
Göttingen, 2017

THESIS COMMITTEE:

Dr. Robert Gütig
Dept. of Theoretical Neuroscience
Max-Planck-Institute for Experimental Medicine Göttingen

Prof. Dr. Theo Geisel
Nonlinear Dynamics
Max-Planck-Institute for Dynamics and Self-organization Göttingen

Prof. Dr. Fred Wolf
Theoretical Neurophysics
Max-Planck-Institute for Dynamics and Self-organization Göttingen

REFEREES:

Prof. Dr. Theo Geisel
Dr. Robert Gütig

ADDITIONAL EXAMINATION BOARD MEMBERS:

Prof. Dr. Tim Gollisch
Dept. of Ophthalmology
University Medical Center Göttingen

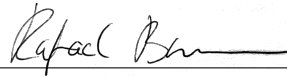
Prof. Dr. Alexander Gail
Sensorimotor Group
German Primate Center Göttingen

Prof. Dr. Tobias Moser
Institute for Auditory Neuroscience
University Medical Center Göttingen

DATE OF ORAL EXAM: December 15, 2017

DECLARATION

Hereby I declare that my doctoral thesis entitled "Margin learning in spiking neural networks" has been written independently with no other sources or aids than quoted.



Rafael Brune, Göttingen, October 2017

ABSTRACT

The ability to learn, generalize and reliably detect features embedded in continuous sensory input streams is a crucial function of the central nervous system. Sensory neurons process input from thousands of synapses and respond to short features embedded in the input spike stream.

Although supervised synaptic learning rules that allow neurons to learn and detect spatio-temporal structures in spike patterns have been developed and studied, it is unclear how neurons can learn to generalize when only a limited set of training examples embedded in high-dimensional input patterns are available. Current learning rules rely on the availability of many training patterns. The neurons generalization performance to previously unseen feature variations suffers from overfitting when the number of its synapses is too high and hence limiting their usefulness when studying neural processing of high-dimensional spatio-temporal input streams.

We introduce a novel definition of margin for spiking neuron models and a learning rule that extends the multi-spike tempotron with methods to increase said margin during training. We discover that this margin learning ensures high generalization ability even when only a small set of training patterns are available and the number of synapses is high. Using features embedded in Poisson patterns we demonstrate the improvement in performance even under noise. By successfully applying the introduced margin learning rules to human speech recognition tasks we show their potential for studying neural processing of high-dimensional inputs in spiking sensory neurons.

CONTENTS

1	INTRODUCTION	1
1.1	Classification Tasks and Neuron Model Framework . .	3
1.2	Thesis Structure	4
2	METHODS	7
2.1	Neuron Models	8
2.1.1	McCulloch-Pitz Model	8
2.1.2	Integrate-and-Fire Neuron	8
2.2	Perceptron Learning	11
2.3	Linear Support Vector Machines	12
2.3.1	Soft-Margin	13
2.4	Aggregate-Label Learning	15
2.4.1	Spike-Threshold-Surface	15
2.4.2	Multi-Spike Tempotron Learning Rule	16
2.4.3	ϑ^* Gradient	17
2.4.4	Neuron Parameters and Momentum Term	22
2.4.5	Pre-Training of the Neuron	22
2.5	Feature Detection Tasks	23
2.5.1	Synthetic Embedded Features Task	23
2.5.2	Phoneme Detection Task	25
2.6	Phoneme Recognition Test Evaluation	27
2.6.1	Proficiency	27
2.7	Parameter Optimization	29
3	LEARNING FROM SEGMENTED INPUT PATTERNS	31
3.1	Limitations of Existing Learning Rules	31
3.2	A Novel Learning Rule for Segmented Training Data . .	32
3.3	Voltage Gradient	36
4	TEMPOTRON LEARNING WITH MARGIN	41
4.1	Margin in the Spike-Threshold-Surface	42
4.2	Comparison with Stochastic Margin Learning	44
4.3	Noise Robustness	46
4.4	Weight Decay and Rescaling	52
4.5	Generalization Performance under Optimal Margin Learning Parameters	58
4.6	Margin Based Optimization	63
5	APPLICATION TO SPEECH RECOGNITION	67
5.1	Auditory Brain-Stem Model	67
5.1.1	Typical Responses of Auditory Neurons	67
5.1.2	Auditory Front-End	69
5.2	Phoneme Recognition Task	77
5.3	Generalization Performance and Front-End Dimension	82
5.4	General Performance Improvement	85

5.5	Weight Vector Regularization	86
6	DISCUSSION	91
6.1	Margin Learning for Spiking Neuron Models	91
6.1.1	Biological Plausibility	93
6.2	Limitations and Outlook	94
A	APPENDIX	101
	BIBLIOGRAPHY	105

LIST OF FIGURES

Figure 2.1	Linear classifier and margin example.	12
Figure 2.2	Spike-threshold-surface and multi-spike tempotron learning.	16
Figure 2.3	Synthetic embedded feature task example. . .	23
Figure 2.4	TIMIT example speech sentence.	26
Figure 3.1	Segmented learning progression example. . . .	33
Figure 3.2	Generalization performance of segmented learning.	34
Figure 3.3	LTP and LTD steps for the segmented learning algorithm.	37
Figure 3.4	Precision comparison of analytical and numerical gradient calculation.	38
Figure 4.1	Margin in the spike-threshold-surface.	42
Figure 4.2	Optimal learning step size for stochastic margin learning algorithms.	45
Figure 4.3	Convergence time and margin width of stochastic and gradient margin learning.	48
Figure 4.4	Mean minimal margin width for tempotron and margin learning.	49
Figure 4.5	Performance under noise and generalization performance of tempotron and margin learning.	50
Figure 4.6	Synaptic weight vector Euclidean norm for tempotron and margin learning.	51
Figure 4.7	Minimal margin and weight vector norm for margin learning with weight rescaling.	55
Figure 4.8	Noise robustness and generalization performance of margin learning with weight rescaling.	56
Figure 4.9	Relative position of output spike times inside the target feature for different learning rules. . .	57
Figure 4.10	Bimodal generalization error distribution. . . .	59
Figure 4.11	Parameter optimization for margin up learning with weight rescaling.	60
Figure 4.12	Generalization performance of margin learning rule variants.	62
Figure 4.13	Mean margin as a proxy for generalization performance.	64
Figure 4.14	Generalization performance of margin optimized parameters.	65

Figure 5.1	Auditory neuron responses from the inferior colliculus of mice.	68
Figure 5.2	Probe signal to test the auditory front-end spike generators.	70
Figure 5.3	Signal on- and off-set detector output.	71
Figure 5.4	Long- and short-pass detector output.	72
Figure 5.5	Band-pass detector output.	73
Figure 5.6	Spike pattern output for a probe signal.	74
Figure 5.7	Example spike pattern output of the full auditory front-end for a TIMIT sentence.	75
Figure 5.8	Example voltage traces for tempotron and margin learning.	78
Figure 5.9	Parameter optimization runs for phonemes AY and T.	79
Figure 5.10	Impact of limited training data on phoneme recognition performance.	83
Figure 5.11	Performance shift under increasing auditory frontend size.	84
Figure 5.12	Proficiency for tempotron and margin learning.	85
Figure 5.13	Euclidean norm of synaptic efficacies after learning.	86
Figure 5.14	Spike triggered spectrograms for S.	88
Figure 5.15	Synaptic efficacies for tempotron and margin learning for phoneme S.	89
Figure 6.1	Importance of phoneme context.	96

INTRODUCTION

When we look out the window and see a cat walking through the garden our brain is usually able to quickly recognize the animal as a cat and not as a dog. Even if we have never seen this exact cat in this specific context before it seems effortless for our brain to do this classification.

The problem that the brain faces is that rarely, if ever, we repeatedly encounter the same situation. It needs to be able to *generalize* from limited previous encounters with cats and transfer the knowledge to successfully identify new unseen instances. However, this ability to generalize must be balanced with the ability to *discriminate* among different stimuli. Inappropriately broad generalizations can result in confusing a dog with a cat.

This ability to learn and generalize is a crucial function of the central nervous system. Already on a single neuron level specificity and invariance can be observed. For example neurons in the brain of monkeys show specialized selectivity for faces and features present in faces (Perrett, Rolls, and Caan, 1982) while also being able to maintain selectivity over changes in stimulus size, position and viewing angle (Quiroga et al., 2005; Schwartz et al., 1983). These neurons process signals from thousands of synapses and are seemingly able to reliably detect their target feature embedded in their sensory input stream. It is still unclear how they learn, or are trained, to discriminate the target feature from others and are able to generalize from a limited number of encountered examples.

Although supervised synaptic learning rules that allow neurons to learn and detect spatio-temporal structures in spike patterns have been developed and studied (Florian, 2012; Ghosh-Dastidar and Adeli, 2009; Gütig, 2016; Gütig, Gollisch, et al., 2013; Gütig and Haim Sompolinsky, 2006, 2009; Ponulak and Kasiński, 2010; Ran Rubin, Monasson, and Haim Sompolinsky, 2010; Xu, Zeng, and Zhong, 2013), it is unclear how neurons can learn to generalize when only a limited set of training examples embedded in high-dimensional input patterns are available. Current learning rules rely on the availability of many training patterns. With limited training data, current approaches are at risk to overfit such that the neurons generalization performance on previously unseen feature instances suffers. The model learns ir-

relevant noise instead of extracting important aspects of the target feature.

This limits the ability of these learning rules to explain neural processing of high-dimensional spatio-temporal input streams in ethological situations.

To better understand how neurons might be able to implement measures to increase robustness and generalization performance we turn to the field of machine learning. At the beginning of the computer age collaboration between the disciplines of machine learning, neuroscience and psychology was highly productive (Churchland and Sejnowski, 1988; Hebb, 1949; G. E. Hinton, McClelland, and D. E. Rumelhart, 1986; J. J. Hopfield, 1982; McCulloch and Pitts, 1943; Rosenblatt, 1958).

A classical algorithm on the border between machine learning and neuroscience is the Perceptron learning rule (Rosenblatt, 1958). It trains a single neuron such that the linearly weighted sum of its inputs predicts a category based on whether or not it exceeds a fixed threshold. More intuitively this can be described as the neurons inputs being points in a high dimensional space and the neurons weights defining a hyperplane that separates the space into two classes. One of the important breakthroughs for the field of machine learning is Vapnik's work on support vector machines (SVM) (Cortes and V. Vapnik, 1995; V. N. Vapnik and A. J. Chervonenkis, 1974). It improves on the concept and performance of the Perceptron by using a *margin*.

This margin is defined as the distance between the closest points of both classes and the decision hyperplane. The goal of support vector machine learning is to maximize this margin. The intuition behind it being that a larger margin increases the probability that variants of the training data points end up on the same side of the hyperplane and are classified correctly. If the decision hyperplane is directly next to one of the training data points already small variations, e.g. through sensory noise, can lead to misclassification. This learning towards a larger margin allows support vector machines to find solutions that offer increased robustness and generalization performance (Cortes and V. Vapnik, 1995; V. N. Vapnik and A. J. Chervonenkis, 1974). Support vector machines are now a standard tool in machine learning and deliver state-of-the-art performance in applications like text categorization, hand-written character recognition and image classification (Cristianini and Shawe-Taylor, 2000).

So far the typical approach to increase the robustness of spiking neuron models is to apply different types of noise during training (Gütig, 2016; Gütig and Haim Sompolinsky, 2006; R. Rubin, L. F. Abbott, and H. Sompolinsky, 2017). While this research demonstrates

that continuously applying noise before training pattern presentation allows for robust neural selectivity it has the disadvantage of being a stochastic process. It only indirectly increases the robustness by generating artificial stimuli variations. Using this approach is also expected to be slower than a learning algorithm that increases robustness by directly operating with a deterministic gradient. Similar to the tempotron learning rules (Gütig, 2016; Gütig and Haim Sompolinsky, 2006) being dramatically faster than stochastic reinforcement learning schemes (Seung, 2003).

Recent research successfully transferred the margin concept from support vector machines to pools of spiking neurons (Le Mouel, Harris, and Yger, 2014). But their approach is based on using aggregate numbers of spikes from pools of neurons being lower or higher than a certain threshold to distinguish classes. This pooled binary classification is conceptually different from the single neuron as a feature detector model we are interested in.

Although the concept of margins can be naively transferred to binary neural classifiers by using the distance between firing threshold and voltage maximum, it is unclear how to meaningfully define and implement a margin in neuron models that use multiple output spikes for classification.

The goal of this thesis is to transfer the concept of margins from machine to synaptic learning rules. We introduce a definition for margins in spiking neuron models together with learning rules that extend the multi-spike tempotron (Gütig, 2016) with methods to increase the margin during training. We compare our margin learning algorithm with the approach of adding noise during training and show that our learning rule is both more effective and efficient. Using a synthetic task with features embedded in Poisson patterns we demonstrate the improvement in generalization performance under limited availability of training data and under noise. To show the potential for studying neural processing of realistic high-dimensional inputs in spiking sensory neurons we then apply the introduced margin learning rules to a phoneme recognition task based on human speech.

1.1 CLASSIFICATION TASKS AND NEURON MODEL FRAMEWORK

To define what we expect from the introduced margin learning rule we describe here the concept of the feature detection tasks and the general framework used to characterize the learning rule's advantages.

We use a synthetic embedded feature task to test the learning rules ability to train a neural classifiers to detect features embedded in a sensory input stream. The same task with the same parameters as used in Gütig, 2016. It consists of a set of different short spike patterns, features, inserted, with random counts and times, into patterns of background noise. Only one of those features will be used as the target, the others will serve as distractors. The task of the learning rules is to train integrate-and-fire neurons to elicit output spikes at times of target feature presence in the input spike pattern. If we imagine this target feature to correspond to an odor, or clue, about a food source then this clue would already be present in the sensory input stream before any reward signal about the successful acquisition of the food source arrives at the neuron. Meaning that the neuron must change its synaptic efficacies to detect the target feature without the knowledge about its appearance times. Gütig, 2016 introduced an aggregate label learning rule that solves this temporal credit-assignment problem. It is able to train neurons to fire for features embedded in the input pattern by only using the count of target features present in it as a teaching signal. We will be using this aggregate label learning rule throughout this thesis and extend it with algorithms for margin learning.

By limiting the availability of spike patterns during training and measuring the feature detection performance on test patterns we quantify the generalization performance of the resulting neural classifiers. To measure the robustness of the feature detection we also create noisy variations of the training patterns and check if they are still classified correctly.

To test the introduced margin learning rule with a more realistic feature detection task we apply it to human speech processing. Using an auditory front-end that converts sound input into spike patterns suitable for the integrate-and-fire neuron we train neurons to detect phonemes, distinct units of sound in human speech. We measure the generalization performance for the different learning rules for different amounts of training data and dimensionality of the auditory front-end.

1.2 THESIS STRUCTURE

In chapter 2 we review neuron models, learning rules and support vector machines. In this machine learning framework we will show how to construct a maximum margin hyperplane which is the main inspiration for the neural margin learning rule introduced in this thesis. It also introduces the multi-spike tempotron learning rule and

the concept of the spike-threshold-surface (both introduced in Gütig, 2016). Additionally we describe the embedded feature task and TIMIT speech corpus used as training data in our simulations. Chapter 2 does not represent any new work but is the foundation this research is build on.

Chapter 3 describes a new learning rule that in contrast to the aggregate label learning of the multi-spike tempotron makes use of exact timing of target features in the input spike patterns. This learning rule will later be used as a comparison baseline for the phoneme recognition application.

Our definition for margins in spiking neuron models is introduced in chapter 4. Here we demonstrate the effectiveness of the learning rules in comparison with a stochastic margin learning approach and show the increased noise robustness and generalization performance using the embedded feature task described in chapter 3.

In chapter 5 we develop a biologically inspired auditory front-end that converts sound waves into spike patterns suitable for use with the multi-spike tempotron. The front-end is based on previous works (Gütig, 2016; Gütig and Haim Sompolinsky, 2009) but we extend it with additional response types based on frequency, loudness and temporal structure of the input signal. The output of this auditory front-end is then used as training patterns in a phoneme recognition task to compare generalization performance of multi-spike tempotron learning and the new margin learning rule.

We summarize all results and discuss avenues for further research in chapter 6.

METHODS

In this chapter we will introduce two neuron models. The first, the McCulloch-Pitz model is a binary classifier that uses the weighted sum of inputs and compares it to a threshold to determine its binary output. Binary classifier means that given an input the classifier assigns it to one of two classes based on its internal decision rule.

The second neuron model is the more realistic leaky integrate-and-fire neuron. It integrates input spikes from different synapses over time and generates an explicit time dependent voltage trace. If this voltage exceeds a predefined firing threshold an output spike is generated. Since multiple output spikes can be generated over the time course of an input spike pattern this neuron model can be used as a multi-class classifier with different output spike counts corresponding to different classes. We will use the integrate-and-fire neuron as a feature detector and interpret the times of output spikes as the times where the neuron detected a target feature to be present.

To train these neuron models to solve a certain task we require learning rules that describe how their synaptic efficacies should be changed during training. Based on the McCulloch-Pitz model we describe the Perceptron learning rule, basics of support vector machines and the concept behind maximum margin classifiers that we use as the main inspiration for the later introduced margin learning concepts for spiking neural classifiers.

For the integrate-and-fire neuron we will describe the multi-spike tempotron learning rule and the concept of the spike-threshold-surface (both introduced in Gütig, 2016).

To study the different learning rules ability for feature detection we use synthetic embedded features and an English natural language speech corpus (TIMIT) for phoneme recognition.

2.1 NEURON MODELS

2.1.1 *McCulloch-Pitz Model*

The McCulloch-Pitz neuron model (Hertz, Krogh, and Palmer, 1991; McCulloch and Pitts, 1943) is one of the earliest and simplest mathematical descriptions of a biological neuron. It receives input from N synapses in the form of real values x_i , multiplies them with their respective synaptic efficacies or weights w_i and compares the result to a firing threshold ϑ .

$$y = \text{sgn}\left(\sum_i^N w_i x_i - \vartheta\right) \quad (2.1)$$

The sgn is the sign function results in an output of 1 if the sum is above ϑ and -1 otherwise.

A more generic description of the model is reached when the threshold ϑ is folded into the weight vector as an additional weight w_{N+1} with the corresponding input value x_{N+1} kept fixed to a constant for all input vectors.

$$y = \text{sgn}(\vec{x} \cdot \vec{w}) \quad (2.2)$$

This definition of the neuron models decision rule using the dot product makes a geometric interpretation apparent. The weight vector \vec{w} defines a decision surface in the N dimensional input space. This decision hyperplane splits the input space into two volumes and input vectors \vec{x} are assigned to one the two possible output classes based on which volume they are in.

2.1.2 *Integrate-and-Fire Neuron*

Biological neurons receive their input not in the form of an aggregated number but as a sequence of action potentials, or spikes, arriving at different times at its synapses. A more realistic model that includes the temporal dynamics of a real neuron is the integrate-and-fire neuron (Dayan and Laurence F Abbott, 2001).

In this model the neuron behaves like a parallel electric circuit consisting of a capacitor and a resistor.

$$\tau_m \frac{dV}{dt} = V_{\text{rest}} - V(t) + R_m I_e(t) \quad (2.3)$$

Where τ_m is the membrane time constant, R_m the total membrane resistance and $I_e(t)$ the externally applied current. This input current

can correspond to spiking inputs from other presynaptic neurons. Typically these induced input currents are modeled as an exponentially decaying kernel $e^{-(t/\tau_s)}$ with time constant τ_s . Using this exponentially decaying input current allows for solving the differential equation for a single input spike.

$$K(t) = V_{\text{norm}}(e^{-\frac{t}{\tau_m}} - e^{-\frac{t}{\tau_s}}) \quad \forall t \geq 0 \quad (2.4)$$

This double exponential kernel describes the time evolution of relative voltage change caused by a single input spike and is called the postsynaptic potential (PSP). V_{norm} is used to normalize the amplitude of the kernel to unit size.

Additionally the solution of the differential equation also yields an explicit representation of the voltage V_0 as a function of time t .

$$V_0(t) = \sum_{i=1}^N w_i \sum_{t_i^j < t} K(t - t_i^j) + V_{\text{rest}} \quad (2.5)$$

The parameter t_i^j describes the time of the j -th input spike at synapse i and w_i are the efficacies for each synapse.

The effect of spike reset is modeled by setting the voltage to the resting potential whenever the integrated membrane potential exceeds the given firing threshold ϑ . Assuming $V_{\text{rest}} = 0$

$$V(t) = V_0(t) - \vartheta \sum_{t_s^j < t} e^{-\frac{(t-t_s^j)}{\tau_m}} \quad (2.6)$$

With t_s^j is being the time points of reset and the exponential modeling their decaying influence.

Output spikes can have an impact on the voltage trace and timing of the following output spikes. This will be an important aspect to consider in gradient calculation for the multi-spike tempotron learning rule.

2.2 PERCEPTRON LEARNING

One important synaptic learning rule for the McCulloch-Pitz neuron model is the perceptron learning rule (Hertz, Krogh, and Palmer, 1991; Rosenblatt, 1958).

It adapts the synaptic weights \vec{w} of the neuron through repeated presentation of input vectors \vec{x}_i . With the goal of the learning rule being that the classification output y_i of the McCulloch-Pitz model

$$y_i = \text{sgn}(\vec{w} \cdot \vec{x}_i) \quad (2.7)$$

for each input vector \vec{x}_i is equivalent to their corresponding label l_i .

The perceptron learning algorithm is defined as follows: Given a set of input vectors \vec{x}_i , corresponding labels l_i , initial weights \vec{w} and a learning rate η

1. Iterate through all \vec{x}_i and perform the following steps:
 - Calculate the perceptron output $y_i = \text{sgn}(\vec{w} \cdot \vec{x}_i)$.
 - If the output y_i is not equal the desired label l_i update the weights:

$$\Delta \vec{w} = \eta l_i \vec{x}_i \quad (2.8)$$

2. Repeat from 1. until all input vectors are correctly classified.

This learning rule guarantees that the dot product $\vec{w} \cdot \vec{x}_i$ after a corrective learn step increases, or decreases depending on the label, towards the correct classification:

$$(\vec{w} + \eta l_i \vec{x}_i) \cdot \vec{x}_i = \vec{w} \cdot \vec{x}_i + \eta l_i \|\vec{x}_i\|^2 \quad (2.9)$$

Additionally Block, 1962; Novikoff, 1962 proved that if the input vectors of the two classes are linearly separable the perceptron learning rule converges to a solution within a finite number of steps.

In case of an linearly separable set of input vectors there are usually an infinite set of possible solutions and the convergence guarantees only to find one. These different solutions can be of varying quality with respect to the classifier performance on test data. The perceptron of optimal stability or linear support vector machine solves this issue.

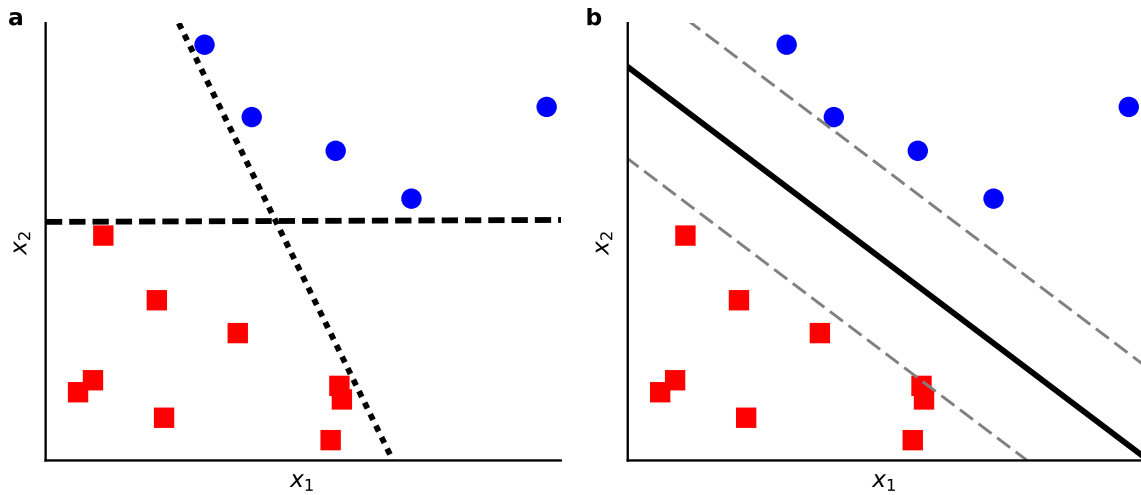


Figure 2.1: (a) Example of a linear classifier in two dimensions. Both lines, dashed and dotted, are solutions that correctly separate red and blue data points. (b) Example of a large margin classifier. The gray dashed lines illustrate the margins between the decision surface and the nearest data points.

2.3 LINEAR SUPPORT VECTOR MACHINES

As mentioned in the case of linearly separable training vector sets the perceptron algorithm is guaranteed to converge to a solution. While these solutions are valid their quality in terms of classification of input vectors that weren't part of the training set varies. The question is which out of all valid solutions is the optimal one?

An approach called *perceptron of optimal stability* or *linear support vector machine* solves this issue by defining an optimal valid solution, the *maximum margin hyperplane* (Cortes and V. Vapnik, 1995; V. N. Vapnik and A. J. Chervonenkis, 1974; V. Vapnik and A. Chervonenkis, 1964).

To find this maximum margin hyperplane we require a hyperplane

$$\vec{w} \cdot \vec{x} - b = 0 \quad (2.10)$$

that maximizes the distance between the hyperplane and the nearest points of each input data class. b is the bias, or threshold, as it was used in McCulloch-Pitz neuron model.

To do this we define two parallel hyperplanes that separate the input space correctly while maximizing the distance between them.

$$\begin{aligned} \vec{w} \cdot \vec{x}_i - b &= +1 \\ \vec{w} \cdot \vec{x}_i - b &= -1 \end{aligned} \quad (2.11)$$

We can now calculate the distance of a point to the decision plane.

$$p_i = \frac{y_i(\vec{w} \cdot \vec{x}_i - b)}{\|\vec{w}\|} \quad (2.12)$$

For points satisfying

$$y_i(\vec{w} \cdot \vec{x}_i - b) - 1 = 0 \quad (2.13)$$

which are points that lie directly on one of the parallel hyperplanes, we get a distance of

$$p = \frac{1}{\|\vec{w}\|} \quad (2.14)$$

The space between these two hyperplanes is called the *margin* and the hyperplane in the middle is the maximum margin hyperplane. The total distance between the two planes yields

$$d = \frac{2}{\|\vec{w}\|} \quad (2.15)$$

Hence to maximize the margin one has to find a solution that fulfills the criteria for correctly classifying the training data

$$l_i(\vec{w} \cdot \vec{x}_i - b) \geq 1 \quad \forall i \quad (2.16)$$

while minimizing the Euclidian weight vector norm

$$E = \|\vec{w}\| \quad (2.17)$$

The points \vec{x}_i on the two parallel hyperplanes are called *support vectors*.

In contrast to the perceptron learning rule which is an online learning algorithm, it iterates through all input vectors one after another, solutions for the support vector machine are typically calculated by solving the above optimization problem.

2.3.1 Soft-Margin

To extend this approach to non-linear separable training data sets Cortes and V. Vapnik, 1995 proposed the *soft-margin support vector machine*. They proposed use of the hinge loss function

$$p_i = \max(0, 1 - l_i(\vec{w} \cdot \vec{x}_i - b)) \quad (2.18)$$

that assigns a loss p_i to each misclassified input vector. This penalty is proportional to the distance from the corresponding decision hyperplane. Using the hinge loss function the minimization problem can be written as

$$E = \kappa \|\vec{w}\| + \frac{1}{n} \sum_i^n \max(0, 1 - l_i(\vec{w} \cdot \vec{x}_i - b)) \quad (2.19)$$

in which parameter κ is used to prioritize between weight vector regularization and correct classification of training samples.

2.4 AGGREGATE-LABEL LEARNING

We now take a look at a neuron that is processing sensory input streams, e.g. vision, hearing, smell, in which sensory clues about the environment are embedded. If this neurons task is to learn to detect a certain odor that for example is linked to a food source we can assume that the odor clue is already present in the sensory input stream before a reward signal about the successful acquisition of the food arrives at the neuron. The question is how can the neuron change its synaptic efficacies to learn to detect said odor clue in the sensory input?

In Gütig, 2016 a novel aggregate-label learning rule is proposed that solves this temporal credit-assignment problem. This multi-spike tempotron is the basis for the research of this thesis and the following sections will give an insight into the concept and implementation behind it. First we will introduce the spike-threshold-surface, then the multi-spike tempotron learning rule followed by a detailed calculation of the necessary gradient.

2.4.1 *Spike-Threshold-Surface*

If we imagine an integrate-and-fire neuron that elicits three output spikes based on the current input spike pattern and its synaptic efficacies and the goal is to change these efficacies such that the neuron generates one additional output spike it is unclear how to do this. One mathematical approach is to calculate the gradient of the output spike count k with respect to the efficacies \vec{w} . While k is a function of \vec{w} it can only take discrete values and the gradient would be zero everywhere besides at the undefined points at the discontinuous steps of k .

Solving this problem Gütig, 2016 introduced a continuous objective function by using a new method called the spike-threshold-surface that maps virtual threshold values to output spike counts given an input pattern and current synaptic efficacies. If the current input pattern would generate 3 output spikes and one would slowly decrease the natural threshold value away from $\vartheta = 1$ there will be a critical threshold value ϑ_4^* at which the output spike count jumps from 3 to 4. Following this ϑ_k^* is defined as the critical threshold value at which the output spike count switches from k to $k - 1$. Each ϑ_k^* corresponds to a specific voltage value of equation 2.6, if we set the firing threshold to $\vartheta = \vartheta_k^*$ then, by definition of ϑ_k^* , $V(t)$ will reach this new firing threshold exactly k times. Since each ϑ_k^* corresponds to a specific volt-

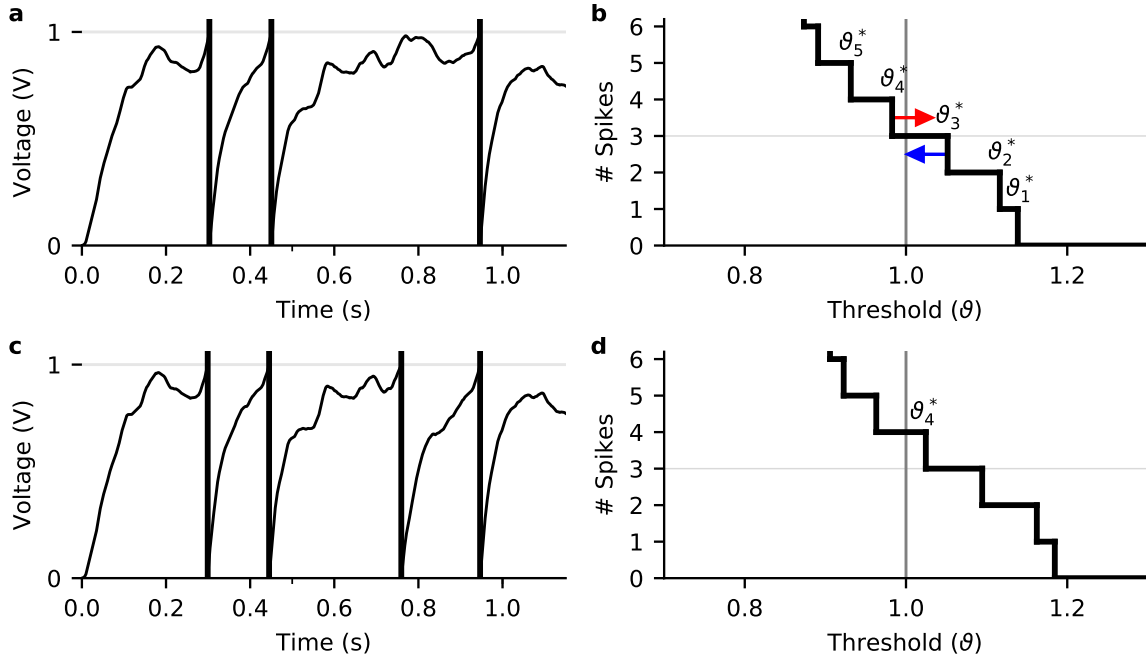


Figure 2.2: (a+b) Voltage trace of a leaky integrate-and-fire neuron for an example input pattern and its corresponding spike-threshold-surface. The current output spike count of 3 is marked with a horizontal line in the spike-threshold-surface. The θ_k^* threshold values mark the virtual threshold value at which the neuron would start to elicit k output spikes. If we would move the virtual threshold across θ_4^* from the left the neuron would go from 4 output spikes to three. By operating in this abstract space of continuous threshold variables a gradient based learning rule can be used (illustrated by red and blue arrows) to decrease the distance between the biological threshold and the desired output spike count. (c+d) Voltage trace and spike-threshold-surface after long-term-potential (LTP) learn steps along the gradient $\vec{\nabla}_{\bar{w}} \theta_4^*$, red arrow in (a), have been applied to change the synaptic efficacies until θ_4^* is above the natural threshold of $\theta = 1.0$.

age value they are a function of the neurons synaptic efficacies. Their gradients with respect to the efficacies are, in contrast to the gradient of the output spike count, meaningful.

Figure 2.2 (a) and (b) show the voltage trace and corresponding spike-threshold-surface for an example spike pattern and neuron efficacies.

2.4.2 Multi-Spike Tempotron Learning Rule

Following what is illustrated in figure 2.2 we can now do the following: if the neuron is supposed to elicit 4 instead of the current 3 output spikes we take a look at θ_4^* in the spike-threshold-surface. As expected the critical threshold value for 4 output spikes is lower than

the natural threshold of $\vartheta = 1$. To get the neuron to fire 4 times instead of 3 we need to somehow change the efficacies of the neuron in a way that ϑ_4^* is above the natural threshold. In contrast to the function of output spike counts, as mentioned in the previous section, ϑ_k^* is differentiable with respect to \vec{w} and we can calculate the gradient $\vec{\nabla}_{\vec{w}} \vartheta_{k+1}^*$ and move ϑ_4^* towards ϑ until it crosses it.

Generalizing this gives us the multi-spike tempotron learning rule:

1. If the desired spike count o is smaller than the current spike count k apply a long-term potentiation (LTP) learn step:

$$\Delta \vec{w} = \eta \vec{\nabla}_{\vec{w}} \vartheta_{k+1}^* \quad (2.20)$$

2. Otherwise if o is bigger than k apply a long-term depression (LTD) learn step instead:

$$\Delta \vec{w} = -\eta \vec{\nabla}_{\vec{w}} \vartheta_k^* \quad (2.21)$$

3. If o equals k the neuron already classifies this pattern correctly and no learning will be done.

The parameter η is the learn step size that is used to update the synaptic efficacies.

2.4.3 ϑ^* Gradient

Applying these updates to the neurons synaptic efficacies requires the calculation of the ϑ^* gradient. In this section we repeat the calculation described in Gütig, 2016 but go into more detail in some of the trickier parts. For this we assume that the exact value of the critical threshold ϑ^* for the output spike count required has already been determined (see Gütig, 2016 on how to numerically determine ϑ^*). Equation 2.6 can then be written as

$$V(t) = V_0(t) - \vartheta^* \sum_{t_s^j < t} e^{-\frac{(t-t_s^j)}{\tau_m}} \quad (2.22)$$

We expect that with well behaved input patterns there exists only a single t^* corresponding to the desired ϑ^* . By definition the voltage at this time point is equivalent to ϑ^* as well as to the time points of all previous output spikes t_s^j .

$$\vartheta^* = V(t^*) = V_0(t^*) - \vartheta^* \sum_{j=1}^m e^{-\frac{(t^*-t_s^j)}{\tau_m}} \quad (2.23)$$

$$\vartheta^* = V(t^*) = V(t_s^j) \quad \forall t_s^j < t^* \quad (2.24)$$

Applying the derivative $\frac{d}{dw_i}$ gives us the individual components $\vartheta_i^{*'}$ of the gradient and the following equivalence which will be useful later on.

$$\vartheta_i^{*'} = \frac{d}{dw_i} \vartheta^* = \frac{d}{dw_i} V(t^*) = \frac{d}{dw_i} V(t_s^j) \quad (2.25)$$

Due to the dependence of t^* and all t_s^j on w_i we have to include these indirect dependencies of $V(t^*)$ when calculating the derivative:

$$\vartheta_i^{*'} = \frac{d}{dw_i} V(t^*) \quad (2.26)$$

$$= \frac{\partial}{\partial w_i} V(t^*) + \sum_{j=1}^m \frac{\partial}{\partial t_s^j} V(t^*) \frac{d}{dw_i} t_s^j + \underbrace{\frac{\partial}{\partial t^*} V(t^*) \frac{d}{dw_i} t^*}_{=0} \quad (2.27)$$

The last term vanishes due to either t^* being a local maximum and hence $\frac{d}{dw_i} t^* = 0$ or t^* coincides with an inhibitory input spike and does not depend on w_i .

The derivatives at the output spike times can be calculated the same way:

$$\forall k \in 1 \dots m \quad \frac{d}{dw_i} V(t_s^k) = \frac{\partial}{\partial w_i} V(t_s^k) + \sum_{j=1}^k \frac{\partial}{\partial t_s^j} V(t_s^k) \frac{d}{dw_i} t_s^j \quad (2.28)$$

Using the equivalence with $\vartheta_i^{*'}$ and pulling out the term for $j = k$ from the sum lets us obtain

$$\vartheta_i^{*'} = \frac{d}{dw_i} V(t_s^k) \quad (2.29)$$

$$= \frac{\partial}{\partial w_i} V(t_s^k) + \sum_{j=1}^{k-1} \frac{\partial}{\partial t_s^j} V(t_s^k) \frac{d}{dw_i} t_s^j + \frac{\partial}{\partial t_s^k} V(t_s^k) \frac{d}{dw_i} t_s^k \quad (2.30)$$

$$\Leftrightarrow \frac{d}{dw_i} t_s^k = \frac{1}{\frac{\partial}{\partial t_s^k} V(t_s^k)} \left(\vartheta_i^{*'} - \frac{\partial}{\partial w_i} V(t_s^k) - \sum_{j=1}^{k-1} \frac{\partial}{\partial t_s^j} V(t_s^k) \frac{d}{dw_i} t_s^j \right) \quad (2.31)$$

To illustrate how we can refactor this equation we use the definition

$$\dot{V}(t_s^k) \equiv \frac{\partial}{\partial t} V(t) \Big|_{t=t_s^k-} \quad (2.32)$$

for the derivative at point t_s^k approaching from the left and as an example explicitly write out all necessary terms for $k = 3$:

$$\frac{d}{dw_i} t_s^1 = \frac{1}{\dot{V}(t_s^1)} \left(\vartheta_i^{*'} - \frac{\partial}{\partial w_i} V(t_s^1) \right) \quad (2.33)$$

$$\frac{d}{dw_i} t_s^2 = \frac{1}{\dot{V}(t_s^2)} \left(\vartheta_i^{*'} - \frac{\partial}{\partial w_i} V(t_s^2) - \frac{\partial}{\partial t_s^1} V(t_s^2) \frac{d}{dw_i} t_s^1 \right) \quad (2.34)$$

$$\frac{d}{dw_i} t_s^3 = \frac{1}{\dot{V}(t_s^3)} \left(\vartheta_i^{*'} - \frac{\partial}{\partial w_i} V(t_s^3) - \frac{\partial}{\partial t_s^1} V(t_s^3) \frac{d}{dw_i} t_s^1 - \frac{\partial}{\partial t_s^2} V(t_s^3) \frac{d}{dw_i} t_s^2 \right) \quad (2.35)$$

Inserting the terms for $k = 1$ and $k = 2$ results in

$$\begin{aligned} \frac{d}{dw_i} t_s^3 = \frac{1}{\dot{V}(t_s^3)} \left(\right. \\ & + \vartheta_i^{*'} \\ & - \frac{\partial}{\partial w_i} V(t_s^3) \\ & - \frac{\partial}{\partial t_s^1} V(t_s^3) \frac{1}{\dot{V}(t_s^1)} \vartheta_i^{*'} \\ & + \frac{\partial}{\partial t_s^1} V(t_s^3) \frac{1}{\dot{V}(t_s^1)} \frac{\partial}{\partial w_i} V(t_s^1) \\ & - \frac{\partial}{\partial t_s^2} V(t_s^3) \frac{1}{\dot{V}(t_s^2)} \vartheta_i^{*'} \\ & + \frac{\partial}{\partial t_s^2} V(t_s^3) \frac{1}{\dot{V}(t_s^2)} \frac{\partial}{\partial w_i} V(t_s^2) \\ & + \frac{\partial}{\partial t_s^2} V(t_s^3) \frac{1}{\dot{V}(t_s^2)} \frac{\partial}{\partial t_s^1} V(t_s^2) \frac{1}{\dot{V}(t_s^1)} \vartheta_i^{*'} \\ & \left. - \frac{\partial}{\partial t_s^2} V(t_s^3) \frac{1}{\dot{V}(t_s^2)} \frac{\partial}{\partial t_s^1} V(t_s^2) \frac{1}{\dot{V}(t_s^1)} \frac{\partial}{\partial w_i} V(t_s^1) \right) \end{aligned} \quad (2.36)$$

which allows us to group all terms containing $\vartheta_i^{*'}$ and all terms that do not together.

$$\begin{aligned} \frac{d}{dw_i} t_s^3 = \frac{1}{\dot{V}(t_s^3)} \left[\right. \\ & \vartheta_i^{*'} \left(1 - \frac{\partial}{\partial t_s^1} V(t_s^3) \frac{1}{\dot{V}(t_s^1)} \right. \\ & \quad \left. - \frac{\partial}{\partial t_s^2} V(t_s^3) \frac{1}{\dot{V}(t_s^2)} \left(1 - \frac{\partial}{\partial t_s^1} V(t_s^2) \frac{1}{\dot{V}(t_s^1)} \right) \right) \\ & + \left(- \frac{\partial}{\partial w_i} V(t_s^3) + \frac{\partial}{\partial t_s^1} V(t_s^3) \frac{1}{\dot{V}(t_s^1)} \frac{\partial}{\partial w_i} V(t_s^1) \right. \\ & \quad \left. - \frac{\partial}{\partial t_s^2} V(t_s^3) \frac{1}{\dot{V}(t_s^2)} \left(- \frac{\partial}{\partial w_i} V(t_s^2) + \frac{\partial}{\partial t_s^1} V(t_s^2) \frac{1}{\dot{V}(t_s^1)} \frac{\partial}{\partial w_i} V(t_s^1) \right) \right) \left. \right] \end{aligned} \quad (2.37)$$

This grouping makes it *easier* to spot a more useful definition of the derivative

$$\frac{d}{dw_i} t_s^k = \frac{1}{\dot{V}(t_s^k)} [\vartheta_i^{*'} A_k + B_k] \quad (2.38)$$

using two recursive coefficients A_k and B_k .

$$\begin{aligned} A_k &= 1 - \sum_{j=1}^{k-1} \frac{A_j}{\dot{V}(t_s^j)} \frac{\partial}{\partial t_s^j} V(t_s^k) \\ B_k &= -\frac{\partial}{\partial w_i} V(t_s^k) - \sum_{j=1}^{k-1} \frac{B_j}{\dot{V}(t_s^j)} \frac{\partial}{\partial t_s^j} V(t_s^k) \end{aligned} \quad (2.39)$$

Similarly we can write for t^*

$$\begin{aligned} A_* &= 1 - \sum_{j=1}^m \frac{A_j}{\dot{V}(t_s^j)} \frac{\partial}{\partial t_s^j} V(t^*) \\ B_* &= -\frac{\partial}{\partial w_i} V(t^*) - \sum_{j=1}^m \frac{B_j}{\dot{V}(t_s^j)} \frac{\partial}{\partial t_s^j} V(t^*) \end{aligned} \quad (2.40)$$

We can now insert the definition of $\frac{d}{dw_i} t_s^k$ into the derivative $\vartheta_i^{*'}$ from earlier

$$\begin{aligned} \vartheta_i^{*' } &= \frac{\partial}{\partial w_i} V(t^*) + \sum_{j=1}^m \frac{\partial}{\partial t_s^j} V(t^*) \frac{d}{dw_i} t_s^j \\ &= \frac{\partial}{\partial w_i} V(t^*) + \sum_{j=1}^m \frac{\partial}{\partial t_s^j} V(t^*) \frac{1}{\dot{V}(t_s^j)} [\vartheta_i^{*' } A_j + B_j] \end{aligned} \quad (2.41)$$

$$\begin{aligned} \Leftrightarrow \underbrace{\vartheta_i^{*' } - \vartheta_i^{*' } \sum_{j=1}^m \frac{\partial}{\partial t_s^j} V(t^*) \frac{A_j}{\dot{V}(t_s^j)}}_{\vartheta_i^{*' } A_*} &= \underbrace{\frac{\partial}{\partial w_i} V(t^*) + \sum_{j=1}^m \frac{\partial}{\partial t_s^j} V(t^*) \frac{B_j}{\dot{V}(t_s^j)}}_{B_*} \\ \Leftrightarrow \vartheta_i^{*' } &= -\frac{B_*}{A_*} \end{aligned} \quad (2.42)$$

To finally calculate A_* and B_* we need to explicitly write down the derivatives $\frac{\partial}{\partial w_i} V(t_x)$, $\frac{\partial}{\partial t_s^k} V(t_x)$ and $\dot{V}(t_x)$. At all time points t_x where the voltage reaches the threshold, $[t_s^1, t_s^2, \dots, t_s^m, t^*]$, the neuron model definition

$$V(t) = V_0(t) - \vartheta^* \sum_{t_s^j < t} e^{-\frac{(t-t_s^j)}{\tau_m}} \quad (2.43)$$

can be reduced to

$$\begin{aligned}
V(t_x) &= V_0(t_x) - V(t_x) \sum_{t_s^j < t_x} e^{-\frac{(t_x - t_s^j)}{\tau_m}} \\
\Leftrightarrow V(t_x) - V(t_x) \sum_{t_s^j < t_x} e^{-\frac{(t_x - t_s^j)}{\tau_m}} &= V_0(t_x) \\
\Leftrightarrow V(t_x) &= \frac{V_0(t_x)}{1 - \sum_{t_s^j < t_x} e^{-\frac{(t_x - t_s^j)}{\tau_m}}} \\
\Leftrightarrow V(t_x) &= \frac{V_0(t_x)}{C_{t_x}}
\end{aligned} \tag{2.44}$$

Where C_{t_x} is defined as

$$C_{t_x} \equiv 1 - \sum_{t_s^j < t_x} e^{-\frac{(t_x - t_s^j)}{\tau_m}} \tag{2.45}$$

This simpler definition of $V(t_x)$ allows us to calculate the missing derivatives

$$\begin{aligned}
\frac{\partial}{\partial w_i} V(t_x) &= \frac{1}{C_{t_x}} \frac{\partial}{\partial w_i} V_0(t_x) \\
&= \frac{1}{C_{t_x}} \sum_{t_i^j < t_x} K(t_x - t_i^j)
\end{aligned} \tag{2.46}$$

$$\begin{aligned}
\forall t_s^k < t_x \quad \frac{\partial}{\partial t_s^k} V(t_x) &= V_0(t_x) \frac{\partial}{\partial t_s^k} \frac{1}{C_{t_x}} \\
&= -\frac{V_0(t_x)}{C_{t_x}^2} \frac{e^{-\frac{(t_x - t_s^k)}{\tau_m}}}{\tau_m}
\end{aligned} \tag{2.47}$$

$$\begin{aligned}
\dot{V}(t_x) &= \frac{\partial}{\partial t_x} V(t_x) = \frac{\partial}{\partial t_x} \frac{V_0(t_x)}{C_{t_x}} \\
&= \frac{1}{C_{t_x}^2} \left[C_{t_x} \frac{\partial}{\partial t_x} V_0(t_x) + \frac{V_0(t_x)}{\tau_m} \sum_{t_s^j < t_x} e^{-\frac{(t_x - t_s^j)}{\tau_m}} \right]
\end{aligned} \tag{2.48}$$

With these three explicit derivatives one is able to fully calculate all parts of the recursive A and B definitions and the gradient $\vec{\nabla}_{\vec{w}} \theta^*$ necessary for the multi-spike tempotron rule.

A comparison of this analytical gradient calculation with a numerical approximation can be found in figure 3.4 in the section about a novel segmented learning rule.

2.4.4 Neuron Parameters and Momentum Term

If not mentioned otherwise we will use the following parameters for the neuron model: Membrane and synaptic current time constants of $\tau_m = 20\text{ms}$, $\tau_s = 5\text{ms}$. The learn step size is $\eta = 1e-5$.

As described in Gütig, 2016; Gütig and Haim Sompolinsky, 2006 and also common in machine learning (David E Rumelhart, Geoffrey E Hinton, Williams, et al., 1988) we use a momentum heuristic to accelerate learning. A linear combination of the current gradient and previous update is used to implement a decaying trace of former synaptic changes.

$$\Delta w_i^{\text{current}} = \Delta w_i + \mu \Delta w_i^{\text{previous}} \quad (2.49)$$

This update of w_i^{current} is only applied if the current synaptic change was not zero ($\Delta w_i \neq 0$). With the exception of the direct comparison with stochastic margin learning and all learning done with the segmented learning rule, where $\mu = 0$, we kept μ fixed to 0.99.

2.4.5 Pre-Training of the Neuron

We use two different versions of pre-training in this thesis. Both pre-training variants make sure the neuron is initialized such that it generates a $\approx 5\text{Hz}$ output spike rate when driven by 5Hz Poisson background activity.

The first one is as described in Gütig, 2016 and will be used if not mentioned otherwise. It randomly draws efficacies strengths from a Gaussian distribution with zero mean and 0.01 standard deviation. The learning step size is set to $\eta = 1e-3$ and the momentum to $\mu = 0$. The neuron is then trained on blocks of 100 spike patterns of 1s background activity each with labels drawn from a Poisson distribution with average 5. Pre-training stopped when the neuron generated more than 5Hz firing rate for a block of spike patterns.

The second method is meant to reduce a possible influence of the random initial weights. All efficacies are set to the same small value 0.01 and a 100 second long input pattern of background activity is used to determine the center of the plateau in the spike-threshold-surface that yields a 5Hz output rate. Using this center value we rescale the synaptic weights resulting in a weight vector with standard deviation of zero and an average background noise response rate of 5Hz.

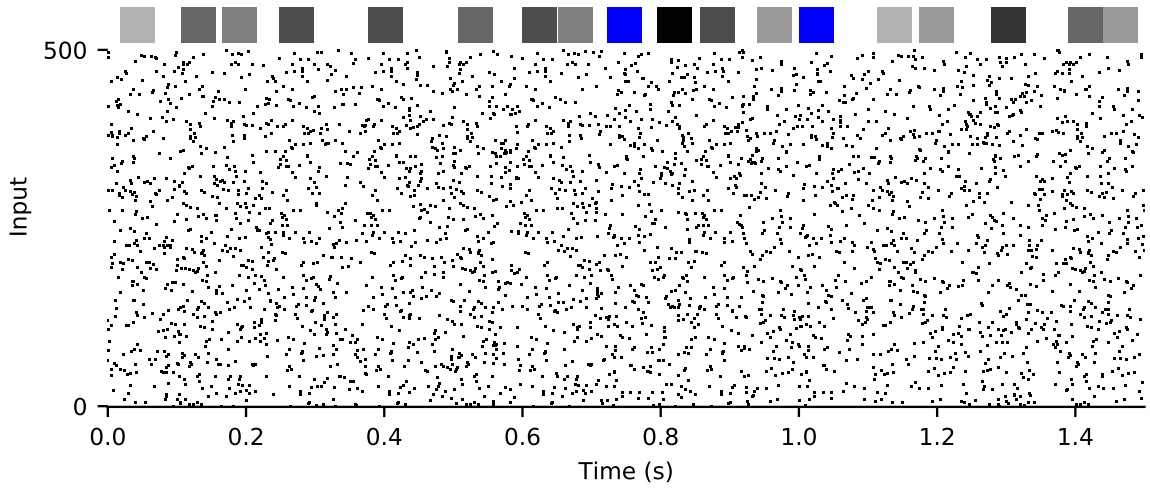


Figure 2.3: 1.5s excerpt from a synthetic embedded feature task spike pattern. Squares denote positions of embedded features. Blue is the target feature, squares of similar gray level correspond to the same distractor feature.

2.5 FEATURE DETECTION TASKS

2.5.1 *Synthetic Embedded Features Task*

To test a neural classifier's ability to detect features embedded in a sensory input stream we use a synthetic embedded feature task. The same task with the same parameters as used in (Gütig, 2016). It consists of a set of different short spike patterns, features, inserted, with random counts and times, into patterns of background noise. Only one of those features will be used as the target, the others will serve as distractors.

We generate spike pattern for a neuron with $N_{\text{synapses}} = 500$ afferents. First $N_{\text{feature}} = 10$ features with a length of $T_{\text{feature}} = 50\text{ms}$ and average spike rate of $r = 5\text{Hz}$ per afferent are generated. This is achieved by drawing for each synapse first from a Poisson distribution the number of spikes and then the corresponding spike times from a uniform distribution. Background noise of length $T_{\text{noise}} = 2.5\text{s}$ is generated using the same output rate and statistics.

To embed features into the background noise a feature count with average $n_{\text{feature}} = 5$ is drawn from a Poisson distribution for each feature. Corresponding feature times are drawn from a uniform distribution. Features are not allowed to overlap. To avoid this the insertion process iterates through the sorted feature times and after each insertion the following feature times are shifted by the feature length T_{feature} .

By generating new background noise, feature counts and feature times new patterns can be generated that contain the 10 features in different amounts, positions and in different background noise. On average a resulting spike pattern will have the length of $T_{\text{total}} = T_{\text{noise}} + n_{\text{feature}} * T_{\text{feature}} = 5\text{s}$.

Using these patterns together with the count of one of the features as a target spike count we can study how well a neural classifier is able to learn and detect this target feature.

2.5.1.1 Noisy Embedded Features

To measure generalization performance and noise robustness of different learning rules we also use this synthetic feature task but with added noise.

For e.g. 25% noise this is implemented through two variables $p_{del} = 0.25$ and $r_{add} = 1.25\text{Hz}$. p_{del} is the probability with which an input spike gets removed from a spike pattern and r_{add} gives the per synapse spike rate that is added as Poisson distributed noise. These parameters are balanced such that the average input spike rate per synapse of $r = 5\text{Hz}$ is kept ($r = (1 - p_{del})r + r_{add}$). The target features embedded in these noisy spike patterns are all different but still instances of the same template feature.

2.5.2 Phoneme Detection Task

For the phoneme recognition task in chapter 5.2 we use the DARPA TIMIT Acoustic-Phonetic Continuous Speech Corpus (TIMIT) (Garofolo et al., 1993). TIMIT stands for the joint effort of the Massachusetts Institute of Technology (MIT), Stanford Research Institute (SRI), and Texas Instruments (TI). It contains 6300 sentences spoken by 630 speakers with 10 sentences each. The speakers are 70% male and 30% female and come from 8 different major dialect regions of the United States. Sentences were designed and selected to be phonetically diverse and to contain a rich set of phone pairs.

Every sentence comes as a 16-bit, 16kHz wave file and orthographic transcription of the spoken sentence as well as phone boundary aligned segmentation into words and individual phonemes. The annotation on the phoneme level was done manually. Due to it being one of the few carefully annotated speech corpora it has become a wildly used dataset for comparison of phone recognition techniques (Lopes and Perdigao, 2011).

The TIMIT transcriptions use a phonetic alphabet called the ARPABET that consists of the following symbols

- Stops: b, d, g, p, t, k, dx, q
- Affricates: jh, ch
- Fricatives: s, sh, z, zh, f, th, v, dh
- Nasals: m, n, ng, em, en, eng, nx
- Semivowels and Glides: l, r, w, y, hh, hv, el
- Vowels: iy, ih, eh, ey, ae, aa, aw, ay, ah, ao, oy, ow, uh, uw, ux, er, ax, ix, axr, ax-h

For example usage of each symbol see the tables in the appendix.

TIMIT comes with a suggested split into training and test data. The core test data set consists of 24 speakers, two male one female from every dialect region. Since 2 sentences per speaker are always the same they are not included in the suggested test and training data sets. This results in a final number of 192 test sentences in the core test set and 3696 sentences in the training set. We will be using these suggested test and training sentences in our phoneme recognition task.

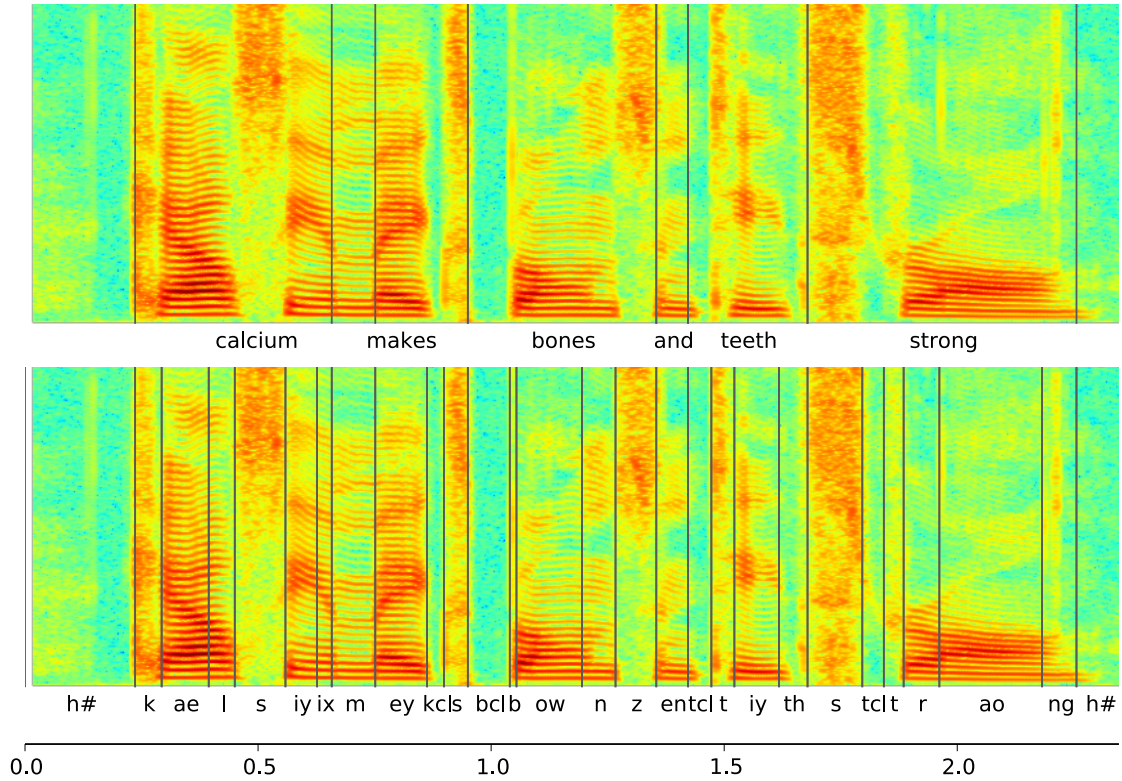


Figure 2.4: (a) Example sentence from the TIMIT test dataset. Borders between individual words have been added. Note how the word 'calcium' and 'makes' overlap. The phone 'm' only appears once and is part of both words. By just looking at the spectrogram it is not obvious how it can be easily segmented into words.

(b) The same example sentence with phone segmentation instead of words. The phones are described by the ARPABET used in the TIMIT dataset ('h#' denotes beginning and end of a sentence). As can be seen the transitions between different phones are smooth and often do not have obvious borders. This means the appearance of many phones in the spectrogram can vary depending on the phone context.

2.6 PHONEME RECOGNITION TEST EVALUATION

A common way to discuss the performance of classifiers is to count the amount of true and false positives as well as true and false negatives. These values can be used to calculate derived descriptors like the hit rate (or recall), false positive rate and precision

$$\text{hit rate} = \frac{\text{true positives}}{\text{true positives} + \text{false negatives}} \quad (2.50)$$

$$\text{precision} = \frac{\text{true positives}}{\text{true positives} + \text{false positives}} \quad (2.51)$$

$$\text{false positive rate} = \frac{\text{false positives}}{\text{false positives} + \text{true negatives}} \quad (2.52)$$

The hit rate describes what fraction of relevant targets have been detected. Precision gives us how many of the detected targets are actually relevant and the false positive rate how many non targets were wrongly classified as targets.

These three measures capture different aspects of the classifiers performance and are usually not discussed in isolation. Instead typically the value of one measure is compared for a fixed level of another measure or combined into a single new measure.

2.6.1 *Proficiency*

A disadvantage of these measures is that they are affected by the relative fractions of target and non-target classes. Different phonemes have wildly different average rates to appear in a TIMIT sentence, from ~ 0.03 for the phoneme 'ENG' to ~ 2 per sentence for 'IX'. Comparing hit- and false positive rates across phonemes is not directly possible since the average target feature rate as well as the amount of null-patterns influence what the classifier prioritizes. If there are 50% null-patterns it is easier to not fire for all of them instead of learning to elicit the correct output spike count for the other 50% of the patterns that contain targets.

Instead of the rates described before we use a different measure based on information entropy. The proficiency, also called uncertainty coefficient or Theil's U (Press, 2007; Theil, 1970) determines the degree of association between two variables. In this case the discrete variables are the true sequence of target and non-targets for a given phone X and the output sequence generated by the classifier Y . The

uncertainty coefficient then answers the question: what fraction of information about X does Y predict?

To calculate the proficiency we need the entropy $H(X)$, conditional entropy $H(Y|X)$ and mutual information $I(X;Y)$ as defined by (Shannon and Weaver, 1998).

$$H(X) = - \sum_{i=1}^n P(x_i) \log_b P(x_i) \quad (2.53)$$

$$H(X|Y) = - \sum_{x \in \mathcal{X}, y \in \mathcal{Y}} p(x, y) \log \frac{p(x, y)}{p(x)} \quad (2.54)$$

$$I(X;Y) = \sum_{y \in \mathcal{Y}} \sum_{x \in \mathcal{X}} p(x, y) \log \left(\frac{p(x, y)}{p(x) p(y)} \right) \quad (2.55)$$

Where $p(x)$ are the distributions and $p(x, y)$ the joint distributions of X and Y .

Putting this together gives us the proficiency as the normalized mutual information.

$$U(X, Y) = \frac{H(X) - H(X|Y)}{H(X)} = \frac{I(X;Y)}{H(X)} \quad (2.56)$$

It tells us what fraction of bits of X can Y predict.

To calculate the proficiency for a phoneme classifier we have to calculate the the distributions $p(x)$ and the joint distributions $p(x, y)$ of X and Y . For this we take the correct sequence of phonemes in a sentences and convert it into a binary sequence of targets and non-targets for each phoneme. We treat each phoneme that is not the target as an individual non-target in the binary sequence. If we would merge them into one big non-target that separates targets the result would be skewed due to multiple erroneous output spikes inside of this block only being treated as one error. This binary target sequence is X .

To construct Y we start with a sequence of non-targets with the same length as the correct one and use the output spike times of the neural classifier to mark entries in this sequence as targets.

By counting the targets and non-targets in both sequences and determining the agreement between both correct and classifier output sequence the distributions $p(x)$, $p(y)$ and $p(x, y)$ are determined.

2.7 PARAMETER OPTIMIZATION

For parameter optimization we employ a coordinate descent optimization algorithm (Wright, 2015). It iterates through each dimension and optimizes it independent of the others. The optimization of each dimension is done with the Brent optimization (Brent, 1971; Dekker, 1969) function from the NumPy Python package (Jones, Oliphant, Peterson, et al., 2001). After roughly bracketing the minima with a line-search it uses the bisection, secant and inverse quadratic interpolation methods to iteratively refine its guess for the minima position.

As an initial step size we supply a factor of three over the current parameter value and limit the amount of steps after minimum bracketing to three. After the final guess the current optimal value is used and optimization switches to the next parameter/dimension. Optimization stops if no improvement was made after a full iteration over all parameters.

LEARNING FROM SEGMENTED INPUT PATTERNS

In both the synthetic feature and phoneme detection tasks the training and test data provide access to the times of feature appearance in the input patterns. The aggregate label learning rule of the multi-spike tempotron is specifically developed under the assumption that this knowledge about feature times is not typically available to a learning neuron. To quantify the price of using aggregate labels instead of the timing information about target features we require a learning rule capable of incorporating this additional information.

Here we adapt an existing learning rule to the requirements of the feature detection task to use it as a comparison for the multi-spike tempotron and the margin learning rule.

3.1 LIMITATIONS OF EXISTING LEARNING RULES

In previous research (Florian, 2012; Memmesheimer et al., 2014; Ponulak and Kasiński, 2010) Perceptron like synaptic learning rules have been presented that can train a neuron using a given sequence of output spikes. Their goal is to study how neurons are able to generate precisely timed output spike sequences while we only require output spikes to be generated inside target feature appearance windows and not at specific time points.

We will concentrate on the learning rule proposed by Memmesheimer et al., 2014 discuss its shortcoming in regards to our application and propose a new learning rule based on similar concepts. Their learning rule is used to study feedforward networks and their capacity to generate desired spike sequences. The general concept is that when given a sequence of desired output spike times $[t_d^1 \dots t_d^n]$ with a given tolerance window $[t_d^x - \epsilon/2, t_d^x + \epsilon/2]$ of width ϵ the algorithm determines the first error in the current output spike sequence and applies weight potentiation or depression based on the error type. Learning from the first error ensures that learning steps are not based on erroneous output spike times that occurred earlier in

the spike output sequence. Weights are modified by a simple update rule

$$\Delta w_i = \pm \eta \sum_{t_i^j < t_{\text{err}}} K(t_{\text{err}} - t_i^j) \quad (3.1)$$

with η a learning rate, t_i^j the input spike times for synapse i and t_{err} the time of the error. If the error is an undesired spike t_{err} is set to the exact time of this spike. In case of a missed output spike in the target spike tolerance window t_{err} is set to the end of this time window.

With this finite precision learning rule one is able to study neuron capacity and the generation of precise output spike trains but when applied to the task of embedded feature detection the following limitations become clear.

- Weight potentiation is fixed to the end of the target output spike tolerance window. The optimal output spike position for a phoneme is unlikely to lie exactly on the end of the segment and most likely varies with different phonemes and phoneme contexts.
- Weight depression happens at the time of erroneous output spikes. Gütig and Haim Sompolinsky, 2006 suggest to use a gradient at the time of the unthresholded voltage maximum to reduce the distance between threshold and maximum instead.
- The learning rule uses an imprecise gradient approximation and ignores contributions of previous output spikes.

3.2 A NOVEL LEARNING RULE FOR SEGMENTED TRAINING DATA

Faced with the previously mentioned limitations we adapted the learning algorithm to allow for training with target feature segments of arbitrary length.

To address these issues we kept the general concept of the learning rule the same but changed how weight potentiation and depression are done. Instead of a sequence of desired output spike times we now have a sequence of target features $[[t_s^1, t_e^1], \dots, [t_s^n, t_e^n]]$ in which times $[t_s^x, t_e^x]$ describe the interval in which one output spike should be generated.

Learning is always done on the first (in time) occurring error in the output spike sequence. This ensures that every learning step minimizes the impact on previous already correct parts of the output sequence. Figure 3.1 shows the learning rule applied to a single training

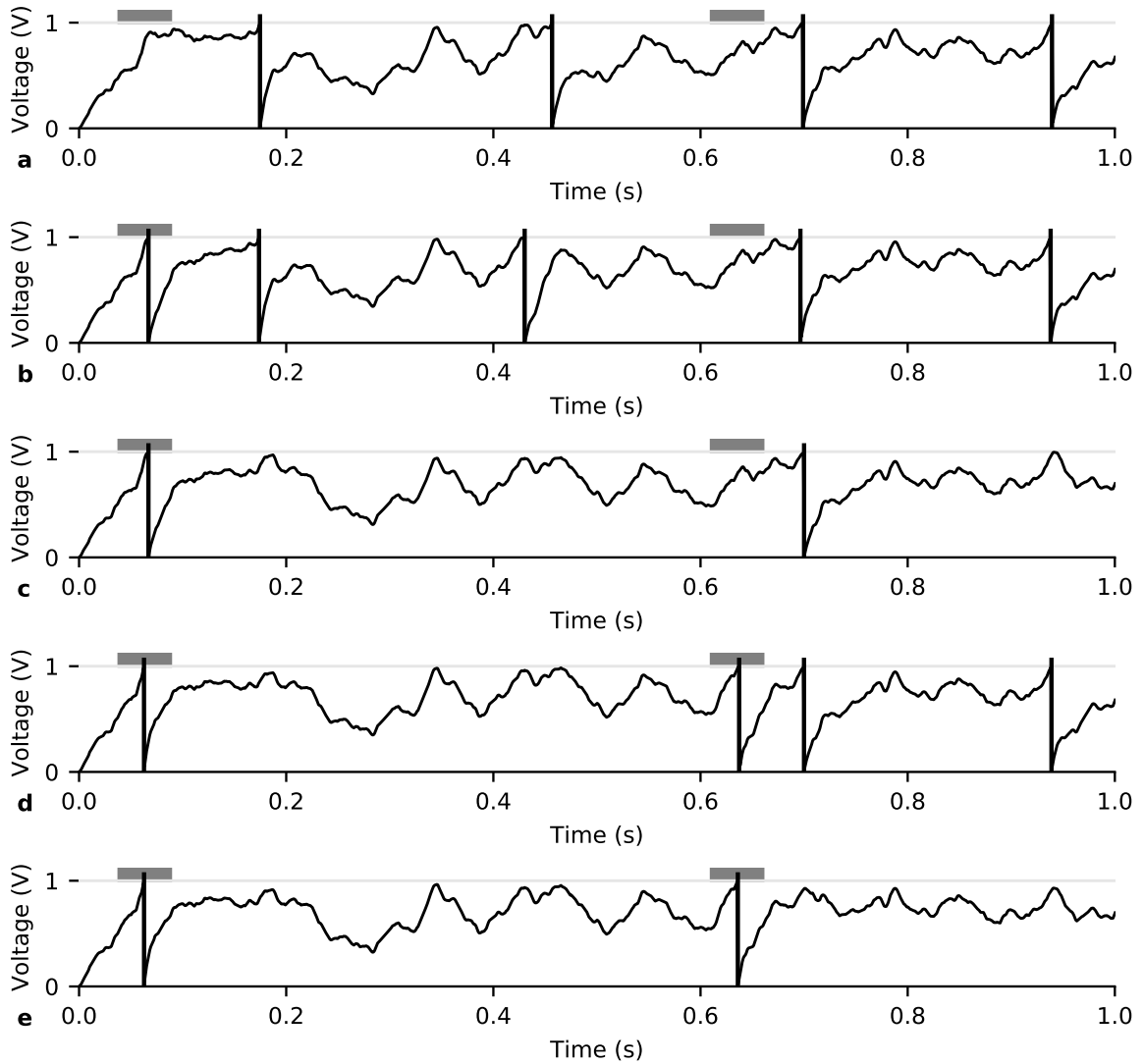


Figure 3.1: From (a) to (e) show the progression of the segmented learning algorithm. Learning always corrects the earliest error the neuron generates. In (a) an output spike should be generated in the first gray marked segment but is missing. Hence from (a) to (b) repeated LTP learning steps are applied to generate an additional output spike at the center of the target segment (the exact position inside the segment at which the LTP gradient is calculated can be chosen as a parameter and optimized for each task). In (b) two erroneous output spikes are generated between the first and second target segment. LTD learning steps are applied. Now in (c) an output spike corresponding to the second target segment is missing and is as in (a) learned by LTP steps calculated at the center of the interval. Lastly in (d) there are again erroneous output spikes which are de-learned by LTD steps.

pattern and illustrates the concept and working of the new learning algorithm.

In case of an error that consist of a missing output spike it is unclear where in the target feature interval $[t_s^x, t_e^x]$ a new output spike should be generated. Memmesheimer et al., 2014 fixed t_{err} to the end of their

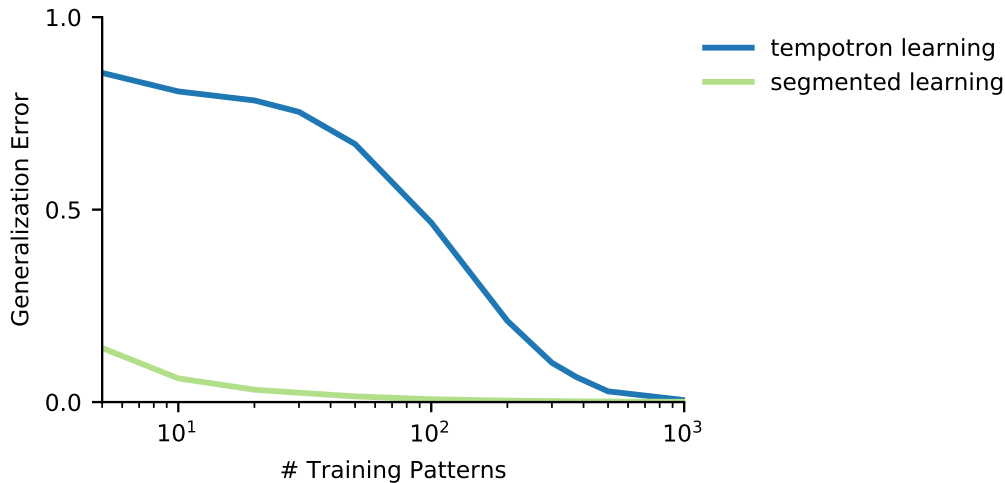


Figure 3.2: Embedded feature task generalization performance for multi-spike tempotron (blue) and segmented learning (green). The learning step sizes η have been optimized for best generalization performance and the generalization error is measured with a test set of patterns separate from the training pool. The momentum for the tempotron was set to $\mu = 0.99$ and training for both learning rules was allowed to last 500 cycles. The trainings error for both learning rules reached zero before reaching this limit. Generalization performance was measured on a test set unused during training.

Access to feature appearance interval information during training allows the segmented learning rule to achieve a low generalization error even when only a very small amount of training patterns are available. The aggregate-label based multi-spike tempotron learning rule does not have access to this timing information and is unable to reach a low generalization error when the training pool size is small: while it learns to elicit the correct amount of output spikes for all training patterns it doesn't necessarily learn to fire for the target feature.

Even though the segmented learning rule is forced to fire inside the target feature for all training patterns when the amount of training data is too small ($N < 100$) it starts to mis-classify some of the test patterns. The training data doesn't contain enough background noise to minimize the effects of background noise in the test pattern.

tolerance window but doing the same, fixing it to the end of the target feature interval t_e^x , is artificially limiting the applicability especially when the target feature interval is longer than τ_s and τ_m . With the average length of phonemes varying wildly (in the TIMIT dataset: 17.5ms for 'B' to 163.0ms for 'AW') learning could happen at a time point in the phone that is not representative or unique to it.

Using the spike-threshold-surface and selecting a ϑ_k^* for LTP gradient learning for which its corresponding t_k^* lies inside the target interval would be an approach following the concept of the tempotron and multi-spike tempotron. Similarly for the LTD step one can utilize the spike-threshold-surface to directly target the ϑ_k^* responsible for the erroneous output spike. Effectively this would implement a multi-spike tempotron learning rule limited to only operate inside a given time interval that is embedded inside a larger input spike

pattern. This spike-threshold-surface based segmented learning rule requires the calculation of many ϑ_k^*, t_k^* pairs to search for the appropriate critical thresholds - which might not even exist if the voltage inside the target segment stays below the resting potential. The computational complexity of this spike-threshold-surface based learning rule is prohibitive to its use in its current state and instead we chose to follow a simpler approach instead.

We introduce an additional parameter $\alpha \in [0, 1]$ that allows t_{err} to be positioned relative to the beginning and end of the target interval $[t_s^x, t_e^x]$.

$$t_{\text{err}} = t_s^x + \alpha(t_e^x - t_s^x) \quad (3.2)$$

This additional parameter can now be tuned for optimal placement of t_{err} inside the target feature.

Based on this error time an LTP step based on the voltage gradient $\nabla V(t_{\text{err}})$ will be used to change the synaptic weights. See Figure 3.3 a+b.

$$\Delta \vec{w} = \eta \vec{\nabla}_{\vec{w}} V(t_{\text{err}}) \quad (3.3)$$

Where η is the learning step size parameter.

In case of a wrongly elicited output spike, a spike outside a target interval or too many output spikes inside of it, we execute an LTD learning step towards de-learning the erroneous spike.

We follow Gütig and Haim Sompolinsky, 2006 to do such an LTD step and first shunt all synaptic inputs after the erroneous output spike t_{err} and then calculate the voltage gradient at the unthresholded voltage maximum t_{max} . Figure 3.3 c+d illustrate the process. The resulting LTD weight update can be written as

$$\Delta \vec{w} = -\eta \vec{\nabla}_{\vec{w}} V_{\text{shunted}}(t_{\text{max}}) \quad (3.4)$$

Where both V_{shunted} and t_{max} are based on the time of t_{err} .

Lastly we need to calculate the gradient. While all three previously mentioned papers, the binary tempotron (Gütig and Haim Sompolinsky, 2006), the multi-spike tempotron (Gütig, 2016) and Memmesheimer et al., 2014 use the same basic neuron model none of their gradients are directly transferable. The binary tempotron does not take into account multiple output spikes and voltage reset, the multi-spike tempotron calculates the gradient for borders in the spike-threshold-surface (see section 2.4), and lastly Memmesheimer ignores the contributions of previously voltage resets to the voltage gradient. In the next section we derive an analytical solution for the needed gradient, following the gradient calculation of the multi-spike tempotron

Gütig, 2016, which is also described in section 2.4, and validate its solution by comparing it to numerical differentiation (figure 3.4).

As mentioned in the introduction of this chapter this learning method can be used as a basis of comparison and to quantify the price of using aggregate labels as a teaching signal. Figure 3.2 shows the generalization performance of both the multi-spike tempotron learning rule and the here introduced learning rule that takes advantage of feature presence interval information. The parameters for both learning rules have been optimized for best generalization performance. Even for very small training pool sizes the segmented learning algorithm is able to learn the target feature and generalize to unseen test data. Without access to target feature timing information the tempotron learning rule needs an order of magnitude more training patterns to reliably detect the target feature in the test pattern.

3.3 VOLTAGE GRADIENT

The gradient calculation follows the approach described by Gütig, 2016 which can also be found here in section 2.4. The main difference is that to calculate the gradient for ϑ_k^* this critical threshold value is dependent on the synaptic efficacies \vec{w} while the firing threshold for the voltage gradient stays fixed at ϑ simplifying the calculation of required derivatives.

As previously described in the section about the multi-spike tempotron the voltage dynamics of the neuron model are described by the following equation:

$$V(t) = \sum_{i=1}^N w_i \sum_{t_i^j < t} K(t - t_i^j) - \vartheta \sum_{t_s^j < t} e^{-\frac{(t-t_s^j)}{\tau_m}} \quad (3.5)$$

With a postsynaptic potential kernel K given by

$$K(t - t_i^j) = V_{norm} \left(e^{-\frac{(t-t_i^j)}{\tau_m}} - e^{-\frac{(t-t_i^j)}{\tau_s}} \right) \quad \forall t \geq t_i^j \quad (3.6)$$

To build the gradient of the voltage V at time t' we need to calculate the derivative with respect to each synapse weight w_i . The reset term of our neuron model makes the voltage dependent on all previously generated output spikes t_s^j . Hence the derivative V_i' needs to take all contributions of previous input spikes into account.

$$V_i' = \frac{d}{dw_i} V(t') \quad (3.7)$$

$$= \frac{\partial}{\partial w_i} V(t') + \sum_{j=1}^m \frac{\partial}{\partial t_s^j} V(t') \frac{d}{dw_i} t_s^j \quad (3.8)$$

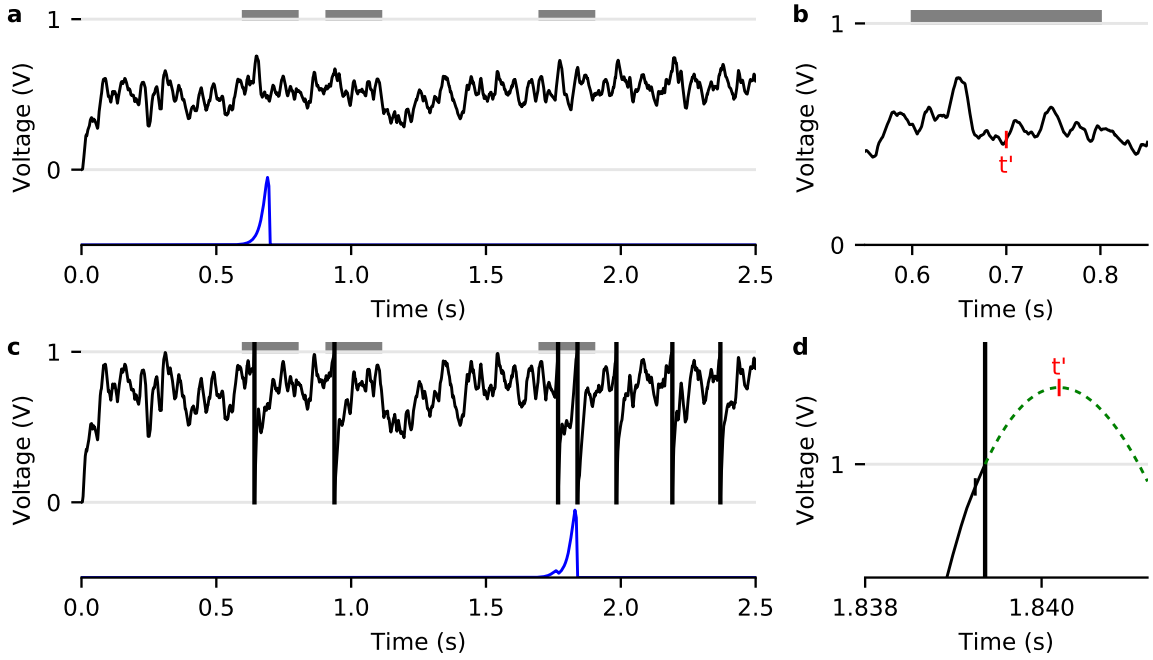


Figure 3.3: (a)+(b) Visualization of the LTP step in the segmented learning algorithm. To move towards weights that generate an additional output spike in the first grey marked interval we calculate the gradient for the voltage at a given t' inside the target segment (here t' is chosen to be at the center of the segment $\alpha = 0.5$). The resulting kernel is illustrated in blue.

(c)+(d) LTD steps are used to remove erroneous output spikes. This is done by first calculating the exact time point of the shunted voltage maximum after the erroneous output spike. (d) shows the original voltage trace in black, the output spike as the thick vertical black line and the shunted (all input spike contributions after the output spike are ignored) in dashed green. The resulting time t is used to calculate the voltage gradient based on the shunted spike inputs. The learning kernel is illustrated in blue. The smaller local maximum in the kernel is contributed by the output spike shortly before the one that is targeted.

To calculate the missing $\frac{d}{dw_i} t_s^j$ we take a look at the voltage derivative at times of output spike generation:

$$\forall k \in 1 \dots m \quad \frac{d}{dw_i} V(t_s^k) = \frac{\partial}{\partial w_i} V(t_s^k) + \sum_{j=1}^k \frac{\partial}{\partial t_s^j} V(t_s^k) \frac{d}{dw_i} t_s^j \quad (3.9)$$

By definition of our neuron model an output spike is generated when the voltage reaches the threshold ϑ . Accordingly a change in w_i may

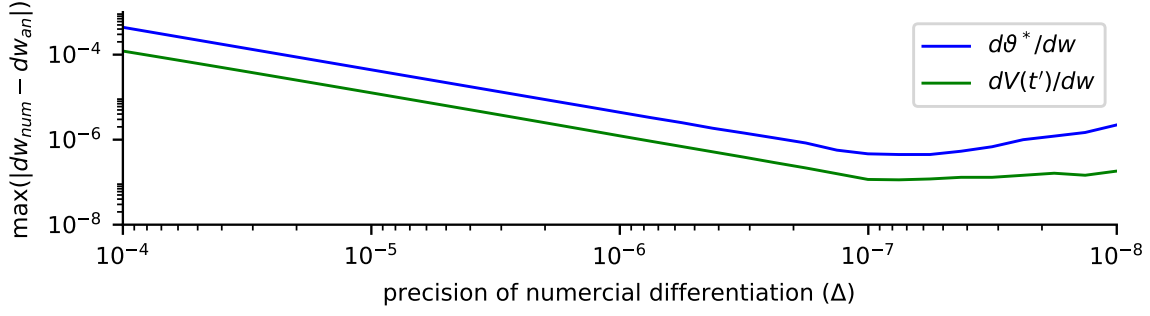


Figure 3.4: To validate the analytical derivation of the gradient we calculated the maximum difference between the analytical and a numerical difference quotient solutions dw vector. With increasing precision of the numeric Δ the maximum difference between the two methods converges. At a precision of about 10^{-7} the numerical solution starts to deprecate due to loss of significant digits during the subtraction of two very similar values and the subsequent division by the smaller and smaller Δ (due to limits of the double precision floating point calculations used). Blue shows the analysis for the multi-spike tempotron $\vec{\nabla}_{\vec{w}} \theta^*$ gradient and green for the segmented learning LTP gradient.

influence the time of t_s^j but $V(t_s^j)$ will always equal θ . The derivative therefore will be zero at all times.

$$0 = \frac{d}{dw_i} V(t_s^k) \quad (3.10)$$

$$= \frac{\partial}{\partial w_i} V(t_s^k) + \sum_{j=1}^{k-1} \frac{\partial}{\partial t_s^j} V(t_s^k) \frac{d}{dw_i} t_s^j + \frac{\partial}{\partial t_s^k} V(t_s^k) \frac{d}{dw_i} t_s^k \quad (3.11)$$

$$\Leftrightarrow \frac{d}{dw_i} t_s^k = \frac{1}{\frac{\partial}{\partial t_s^k} V(t_s^k)} \left(-\frac{\partial}{\partial w_i} V(t_s^k) - \sum_{j=1}^{k-1} \frac{\partial}{\partial t_s^j} V(t_s^k) \frac{d}{dw_i} t_s^j \right) \quad (3.12)$$

Reusing the notation introduced in the the θ^* gradient calculation of the multi-spike tempotron we can rewrite this recursive equation as

$$\frac{d}{dw_i} t_s^k = \frac{1}{\frac{\partial}{\partial t_s^k} V(t_s^k)} B_k \quad (3.13)$$

with B_k defined as

$$B_k = -\frac{\partial}{\partial w_i} V(t_s^k) - \sum_{j=1}^{k-1} \frac{B_j}{\frac{\partial}{\partial t_s^k} V(t_s^k)} \frac{\partial}{\partial t_s^j} V(t_s^k) \quad (3.14)$$

As a last step we need to explicitly build the three missing partial derivatives of equation 3.5 at time t_x used by V'_i and $\frac{d}{dw_i} t_s^k$. With t_x being a placeholder for all output spike times t_s^j and t' at which we calculate the gradient V' .

$$\frac{\partial}{\partial w_i} V(t_x) = \sum_{t_i^j < t_x} K(t_x - t_i^j) \quad (3.15)$$

$$\forall t_s^k < t_x \quad \frac{\partial}{\partial t_s^k} V(t_x) = -\frac{\vartheta}{\tau_m} e^{-\frac{(t_x - t_s^k)}{\tau_m}} \quad (3.16)$$

$$\begin{aligned} \frac{\partial}{\partial t_x} V(t_x) &= \sum_{i=1}^N w_i \sum_{t_i^j < t_x} V_{norm} \left(-\frac{1}{\tau_m} e^{-\frac{(t_x - t_i^j)}{\tau_m}} + \frac{1}{\tau_s} e^{-\frac{(t_i^j - t_x)}{\tau_s}} \right) \\ &\quad + \frac{\vartheta}{\tau_m} \sum_{t_s^j < t_x} e^{-\frac{(t_x - t_s^j)}{\tau_m}} \end{aligned} \quad (3.17)$$

With these explicit partial derivatives all recursive definitions can now be implemented in a computer simulation framework.

We use a mix of python and C code to achieve good usability and fast simulation execution for the multi-spike tempotron as well as the new learning rule introduced here.

TEMPOTRON LEARNING WITH MARGIN

So far the typical approach to increase robustness of spiking neuron models is to apply different types of noise during training. Three different approaches are apparent: removal and addition of random spikes from and into the input pattern (Gütig, 2016), jitter applied to the input spike times (Gütig, 2016; Gütig and Haim Sompolinsky, 2006) and noise on the firing threshold position (R. Rubin, L. F. Abbott, and H. Sompolinsky, 2017). These previous works demonstrate that continuously training with noisy patterns allows for robust neural selectivity.

But adding noise during training has the disadvantage of requiring careful choice of parameters: How to add the noise? What type of noise distribution should be used? How much noise is required? Moreover the presence of noise interferes with the learning of the task leading to slower learning. It is expected that a learning algorithm that increases robustness by directly operating with a gradient would be faster. Similar to the tempotron learning rules (Gütig, 2016; Gütig and Haim Sompolinsky, 2006) being dramatically faster than stochastic reinforcement learning schemes e.g. Seung, 2003.

The theory of support vector machines and the concept of maximum margin classifiers was an important breakthrough in machine learning. It allows the construction of highly robust classifiers with high generalization performance through mapping the input into a high dimensional feature space and maximizing the margin there. So far it was unclear how a meaningful margin can be defined and implemented for spiking neurons.

The concept of a margin can be naively transferred to a binary neural classifier by using the distance between firing threshold and voltage maximum - but when the neuron elicits multiple output spikes this definition loses all meaning as the voltage maximum is, by definition of the neuron model, at the firing threshold. If we disregard some area around each output spike and use the distance between voltage maximum outside this areas and the firing threshold this would give us a measure, albeit with a complex definition and imprecise, for how far away the neuron might be from eliciting an additional output spike. But how far away the neuron is from losing one output spike is not captured in this definition and would probably need an even

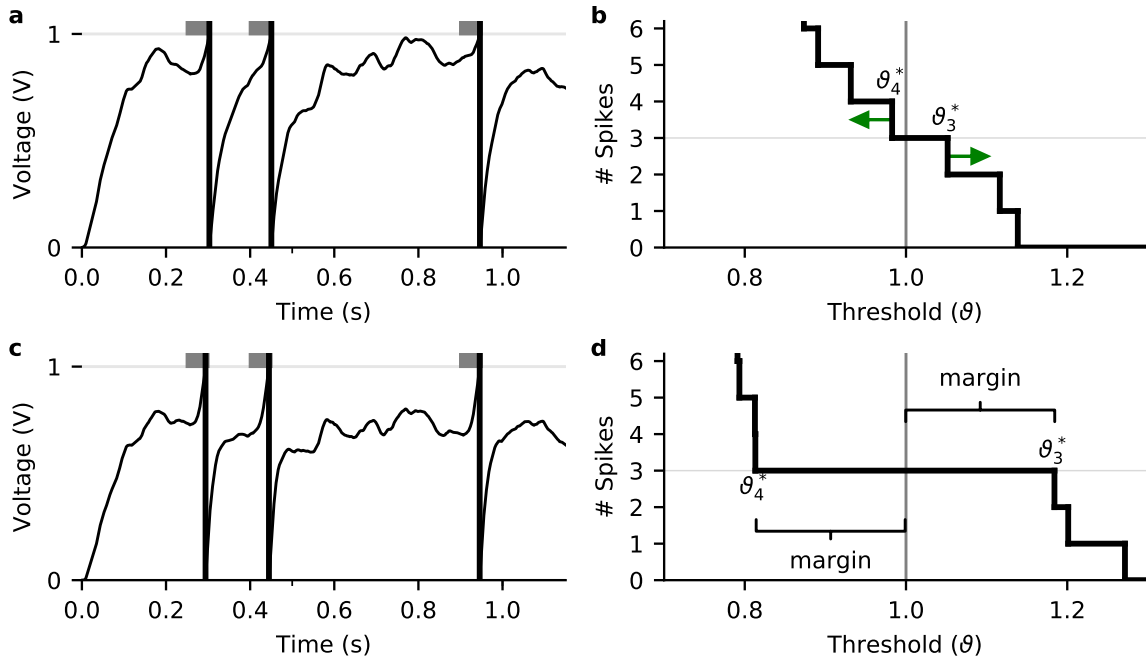


Figure 4.1: (a) For the current input our neural classifier elicits three output spikes. One at each target feature embedded in the input spike pattern (gray areas).

(b) In the corresponding spike-threshold-surface we can see that θ_4^* is close to the firing threshold. Already little amounts of noise in the input pattern could result in this additional 4th spike to be elicited. Our intuition is that if we use learning steps along the gradients $-\vec{\nabla}_{\vec{w}}\theta_{k+1}^*$ and $\vec{\nabla}_{\vec{w}}\theta_k^*$ (green arrows) we can move θ_4^* and θ_3^* further away from the firing threshold and increase the robustness of the neural classifier.

(c) Voltage trace after applying a series of these gradient learning steps with corresponding spike-threshold-surface shown in panel (d). Note how the sub-threshold voltage maxima are pushed away from the firing threshold.

(d) By using the spike-threshold-surface we define margins for a spiking neural classifier as the distances between the firing threshold θ and the neighboring threshold values θ_{k+1}^* and θ_k^* for a desired output spike count k .

more complex workaround by looking at the un-thresholded voltage trace. Also it remains unclear how one would use these definitions for a learning rule to increase the margins during training.

4.1 MARGIN IN THE SPIKE-THRESHOLD-SURFACE

Instead of trying to develop a margin description in the voltage over time representation we use a different approach that builds on the foundation of the multi-spike tempotron (Gütig, 2016) and use it to introduce a margin definition for spiking neurons and accompanying gradient based learning rules.

Figure 4.1 illustrates the concept on a single input pattern. By using the spike-threshold-surface, introduced in Gütig, 2016 and also

described here in section 2.4 we define margins as the distances between the firing threshold ϑ and the neighboring threshold values ϑ_{k+1}^* and ϑ_k^* for a desired output spike count k . We define the minimal margin for k output spikes as

$$\kappa_k = \min(\vartheta - \vartheta_{k+1}^*, \vartheta_k^* - \vartheta) \quad (4.1)$$

When κ_k is positive the neuron currently elicits the desired output spike count k and its magnitude describes the shortest distance to a neighboring threshold value in the spike-threshold-surface - the amount of change required, in input pattern spikes or synaptic efficacies, to change the current output spike count. If κ_k is negative the current output spike count does not equal k and multi-spike tempotron learning steps should be applied. The multi-spike tempotron learning rule can be described as a process to get κ to be positive for all training patterns at their respective desired output spike counts. Multi-spike tempotron learning stops the moment all trainings patterns make the neuron elicit the corresponding requested spike counts. To increase the margins we need to continue learning even when κ is already positive.

As illustrated in figure 4.1 we can use the same learning steps operating on the spike-threshold-surface as the multi-spike tempotron. Instead of moving the threshold values towards and over the firing threshold, we move the threshold values on both sides of the firing threshold further away to increase their margins. For the learning rule we introduce an additional parameter κ_{train} to control when increasing κ_k should stop.

We propose the following learning rule:

1. If the current input pattern does not generate the desired amount of output spikes the normal multi-spike tempotron learning rule is used.
2. When the neuron classifies the current spike pattern correctly we switch to margin learning:
 - a) If the current pattern has a label of zero and $\vartheta - \vartheta_{k+1}^* < \kappa_{\text{train}}$ an LTD step is used to move ϑ_{k+1}^* further away from the threshold.

$$\Delta \vec{w} = -\eta_{\text{margin}} \vec{\nabla}_{\vec{w}} \vartheta_{k+1}^* \quad (4.2)$$

- b) If the pattern has a non-zero label and the margin towards ϑ_k^* is smaller than the margin to ϑ_{k+1}^* and $\kappa_k < \kappa_{\text{train}}$ we

change the neurons synaptic efficacies with an LTP step to move ϑ_k^* further away.

$$\Delta\vec{w} = \eta_{\text{margin}} \vec{\nabla}_{\vec{w}} \vartheta_k^* \quad (4.3)$$

- c) Otherwise if the pattern has a non-zero label and the margin towards ϑ_{k+1}^* is smaller and $\kappa_k < \kappa_{\text{train}}$ we apply an LTD learning step instead.

$$\Delta\vec{w} = -\eta_{\text{margin}} \vec{\nabla}_{\vec{w}} \vartheta_{k+1}^* \quad (4.4)$$

The gradient $\vec{\nabla}_{\vec{w}} \vartheta^*$ is the same as used for the multi-spike tempotron as described in section 2.4. η_{margin} is a learning step size separate from the tempotron learning step size η and allows us to control how strong learning of the margin is in comparison to normal tempotron learning.

The definition 4.1 of κ uses both ϑ_{k+1}^* and ϑ_k^* . However, if the current spike has a label and output of $k = 0$ spikes, only ϑ_{k+1}^* is meaningful. For now we will define the margin κ for null-patterns as the distance between the threshold ϑ and ϑ_1^* . Accordingly, on null patterns we do LTD margin learn steps. In contrast to patterns with a label above zero, margin learning for null-patterns does not have both LTP and LTD efficacy updates which introduces a dependence on the statistics of the trainings data and might be a potential problem that requires further research.

The intuition behind the LTP and LTD steps is the following: by increasing ϑ_k^* we reinforce the synapses leading to the currently weakest output spike. The LTD step on the other hand weakens synapses that might lead to additional erroneous output spikes and as such weakens contributions from background noise or non-target features.

4.2 COMPARISON WITH STOCHASTIC MARGIN LEARNING

In this section we demonstrate that this gradient based margin learning rule is able to quickly and reliably increase the margins by comparing it to a stochastic learning rule based on firing threshold noise that also increases margins.

We chose a model with noise on the firing threshold for comparison since it allows us to use identical input patterns for both learning algorithms which would not be the case if we used a model with stochastic removal and addition of input spikes before training pattern presentation. From the perspective of the spike-threshold-surface

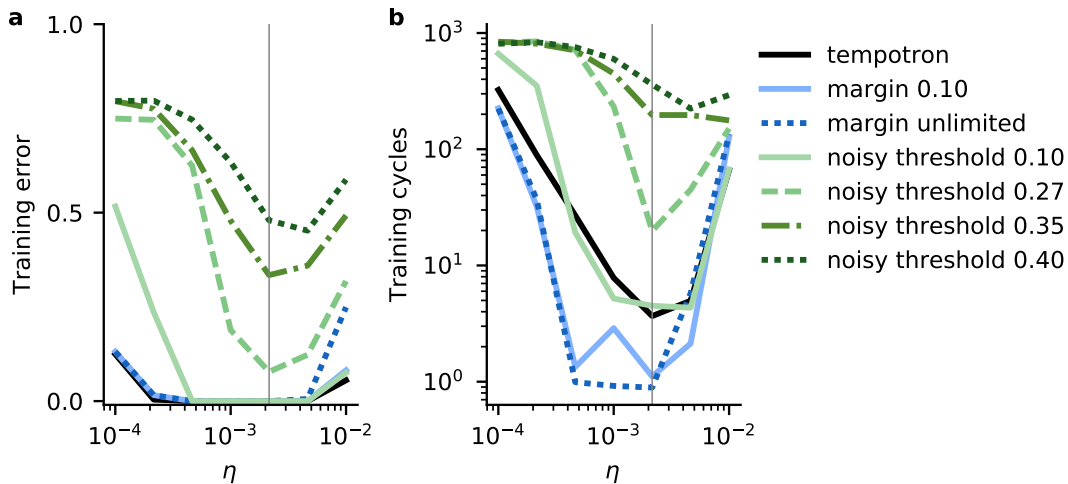


Figure 4.2: (a) Training error for different learn step sizes η . $\eta = 0.002154$, thin vertical line, was chosen for the following simulations as it minimizes the average training error reached within 1000 training cycles for the noisy threshold learning rule. (b) Average number of training cycles required to reach zero training error.

both the margin and the stochastic learning rule do the same learning steps to increase the margin. Both increase the margins by moving ϑ_{k+1}^* and ϑ_k^* further away from the threshold. We expect that the noise on the threshold interferes with the learning of the task. The stronger the noise on the threshold the higher the probability that a learning step is moving a critical threshold in the wrong direction. This should lead to reduced convergence speed and issues when the noise is too strong. By setting κ_{train} to different values for both margin learning algorithms we study their dependence on this parameter.

Noise on the firing threshold before pattern presentation was implemented by drawing a new firing threshold from a uniform distribution in the interval $[\vartheta - \kappa_{\text{train}}, \vartheta + \kappa_{\text{train}}]$. With κ_{train} being the requested minimal margin. This simple stochastic learning rule is not limited to the multi-spike tempotron as it does not use the spike-threshold-surface to determine the current margin or the next learning step. For instance R. Rubin, L. F. Abbott, and H. Sompolinsky, 2017 used a similar approach by drawing noise for the perceptron decision surface from a Gaussian distribution.

To compare the two methods for learning margins we use the synthetic embedded feature task and neuron parameters as described in section 2.5.1. The training pool consists of $N = 1000$ patterns generated without noise. In each learning cycle every pattern is presented once. All simulations are run for 100 different random seeds. To allow for comparability between convergence times of the two learning methods we set the momentum parameter to zero $\mu = 0.0$.

First we determined the optimal learning step size η for the noisy threshold algorithm with respect to the lowest training error reached within 1000 training cycles. The resulting parameter $\eta = 0.002154$ was then used for tempotron, stochastic margin and margin learning rules to track how the training error, the amount of misclassified training patterns, and the critical threshold positions ϑ_{k+1}^* and ϑ_k^* change over training time. η_{margin} for the margin learning rule was set to the same value as η for the tempotron correction learn steps.

Figure 4.3 shows the results for both margin learning methods trained for different κ_{train} . Initially we set $\kappa_{\text{train}} = 0.1$ for both algorithms to compare them at a value of κ_{train} that both are able to reach. For the margin learning rule we can set κ_{train} to infinity without it preventing the learning rule from solving the task. κ increases until it saturates at a maximum value for each training pattern. We measured the mean maximum margin for this task at around $\bar{\kappa} = 0.27$. We used this value as κ_{train} for the stochastic margin algorithm as well as values above it ($\kappa_{\text{train}} = 0.35$, $\kappa_{\text{train}} = 0.40$) to demonstrate the limits of the stochastic learning rule.

As expected the stochastic margin learning algorithm requires more training cycles to reach a zero tempotron error when κ_{train} is increased. Doing the same using the gradient based margin learning method does not negatively affect the mean amount of training cycles needed. Since every margin training step is designed to directly operate on the plateau in the spike-threshold-surface it requires less training cycles to reach the target margin. When setting κ_{train} to infinity the learning rule increases the margin until it saturates without negatively affecting the convergence time. But if the margin parameter for the stochastic algorithm is set too high its performance starts to degrade, it becomes orders of magnitudes slower and not all simulation runs converge to a solution within the maximal number of training cycles. To achieve a maximal margin with the stochastic learning algorithm the margin parameter needs to be carefully tuned since a too small parameter wastes potential and a too high parameter increases training time and the risk for unconverged simulation runs.

4.3 NOISE ROBUSTNESS

The mean κ value across all training patterns is not a good indicator for the robustness of the classifier. In maximal margin classifiers the optimal hyperplane is defined by the margins to the nearest data points in the training vectors. Similarly the smallest κ across all training patterns is a better indicator for the robustness of the neural classi-

fier. In figure 4.3 panel (c) it can be seen that the mean κ for tempotron learning without margin is relatively high (> 0.1) but individual patterns have a κ close to the firing threshold (< 0.01) while the margin learning rules are able to increase κ for all training patterns above 0.1.

To study how noise robustness and generalization performance of the tempotron with and without margin learning is impacted by the amount of available training data we measured the performance with different training pool sizes. Simulations were allowed to run until the trainings error reached zero and the average margin width, the distance between ϑ_{k+1}^* and ϑ_k^* , saturated ($< 1\%$ increase in the last 250 training cycles). Tempotron learn step size was set to $\eta = 1e-05$, momentum is set to $\mu = 0.99$, no noise was applied during training and test pool generation. We used three different learning step sizes for margin learning $\eta_{\text{margin}} = 1e-06$, $\eta_{\text{margin}} = 5e-06$ and $\eta_{\text{margin}} = 25e-06$. Results are shown in figures 4.4, 4.5 and 4.6.

The minimum margin for both sides, ϑ_{k+1}^* and ϑ_k^* , of the multi-spike tempotron, are very close to the threshold (figure 4.4 (b)). Already minimal threshold noise or additions and removal of input spikes can lead to misclassification of training patterns. This can be seen in figure 4.5 (a), it shows the classification error on training patterns under 5 and 10% spike noise (random addition and removal of input spikes). Margin learning increases the observable margins around the threshold of training patterns which leads to drastically improved noise robustness of the resulting solutions. Generalization performance is only slightly improved, see figure 4.5 (b).

These results show that this margin learning rule stabilizes the found solution regardless of its ability to detect the target feature correctly.

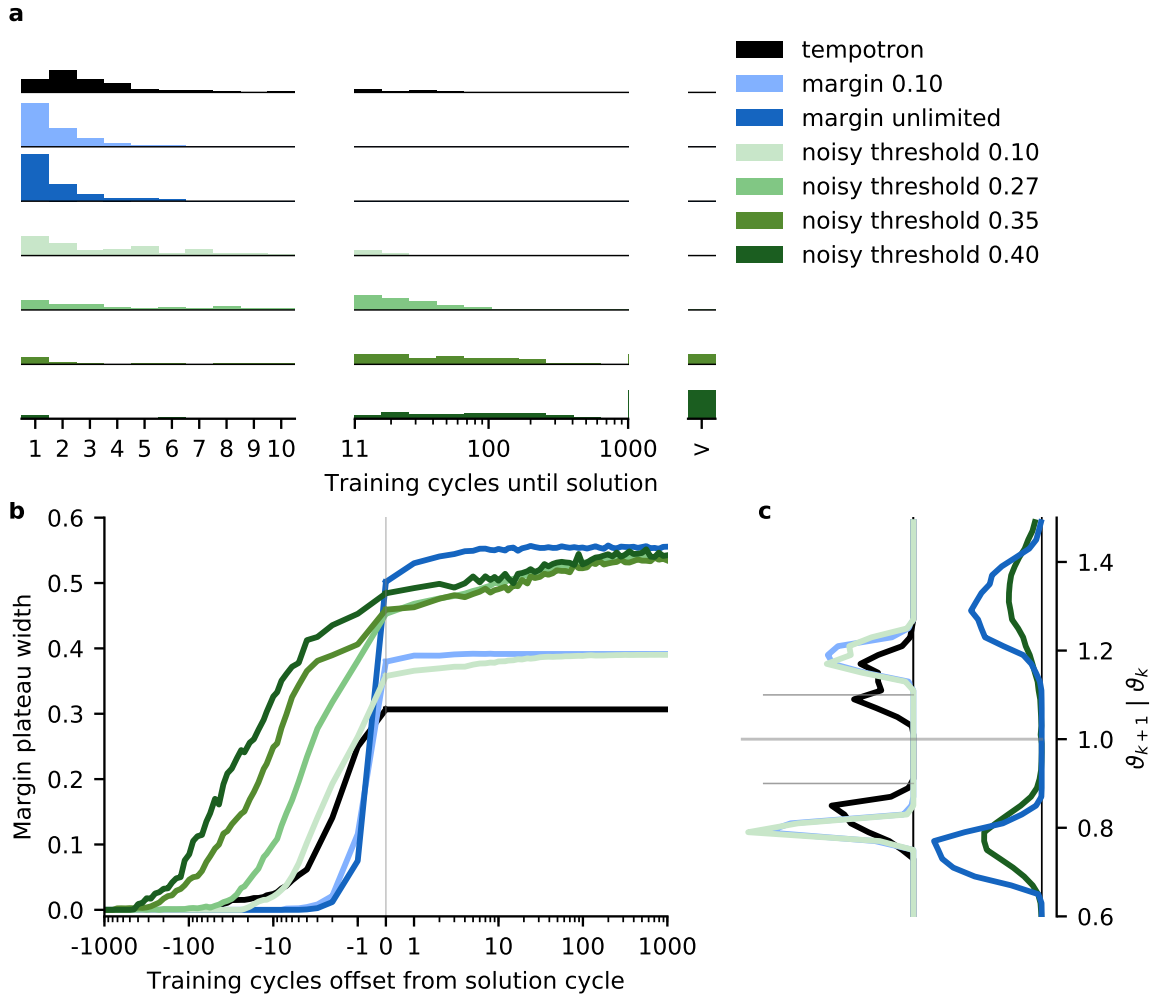


Figure 4.3: (a) Histograms of training cycles needed to reach zero training error for a trainings pool of 1000 training patterns. The graph is separated into linear- and log-scale distributed histogram bins. The bins on the right collect runs that did not reach zero training error during 1000 training cycles (17% for $\kappa_{\text{train}} = 0.35$ and 39% for $\kappa_{\text{train}} = 0.40$). (b) Progression of plateau width ($\vartheta_k^* - \vartheta_{k+1}^*$) in the spike-threshold-surface aligned on convergence time. (c) Histograms showing the distribution of ϑ_k^* and ϑ_{k+1}^* at the end of training for all simulation runs that reached zero training error. Small grey lines indicate κ_{train} .

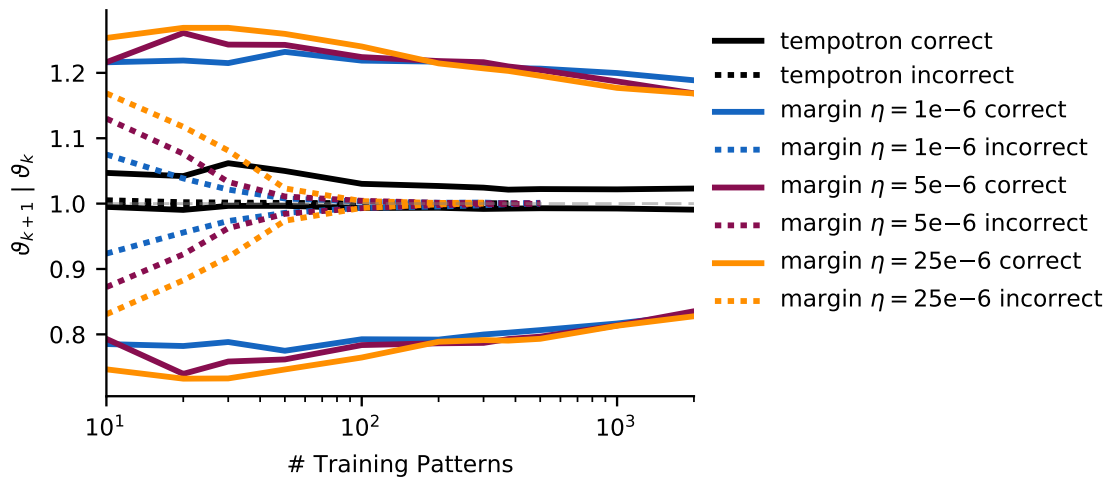


Figure 4.4: Average minimal margin for different training pool sizes. Results are presented split up based on generalization performance. Dotted lines are averages for simulation runs that did not learn to detect the target feature. Margin learning results are shown for three different margin learn step sizes η_{margin} . Since tempotron learning stops as soon as the correct amount of spikes are elicited for every training pattern the average minimal margin to both sides is small. A low amount of threshold noise, missing or additional spikes can already lead to errors in the output spike count.

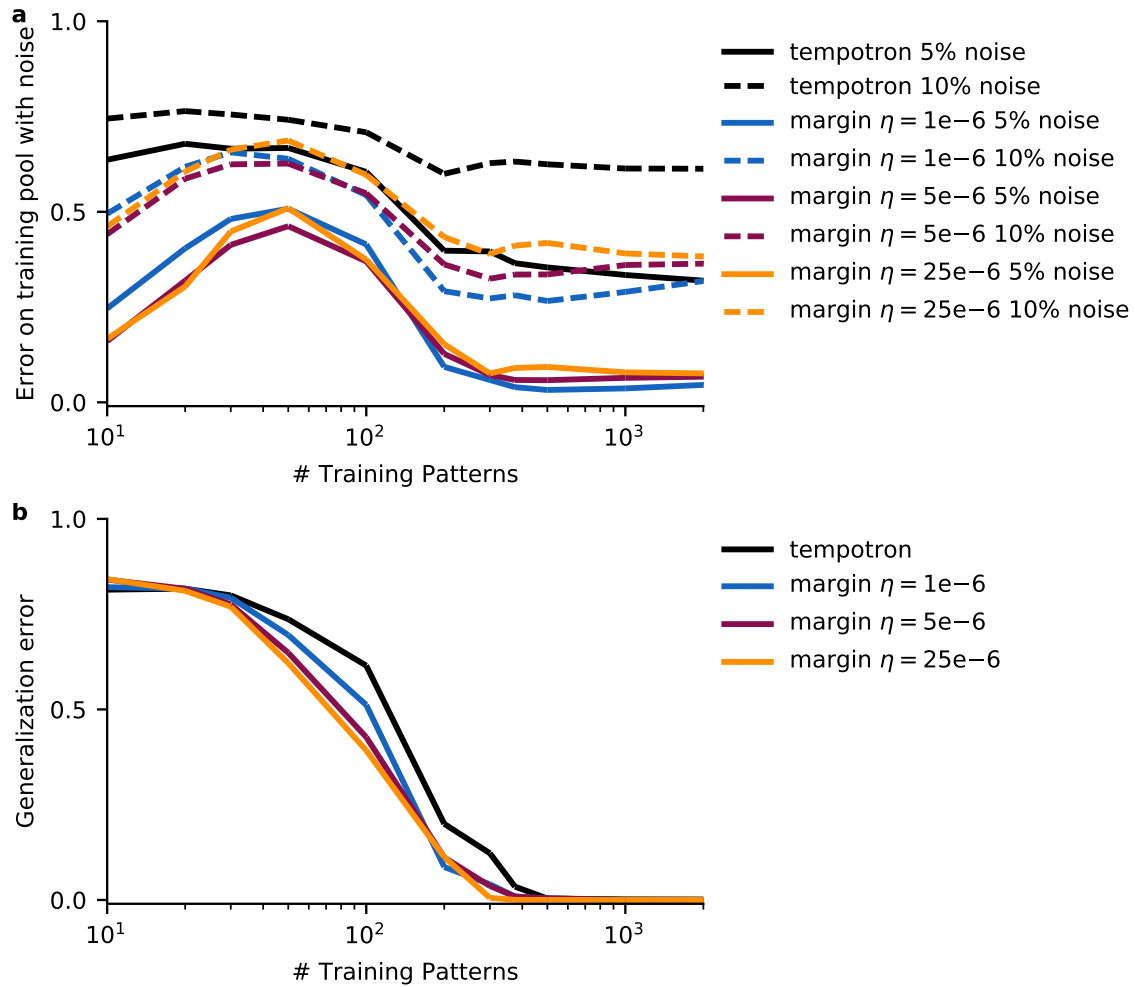


Figure 4.5: (a) Performance measured on the training pool with noise. After training without noise performance was measured on the same training pool but with noise (addition and removal of spikes). Full lines represent 5% spikes removed and randomly added, dashed lines are for 10% removal and addition. (b) Generalization performance measured on a test set of patterns that have not been used during training.

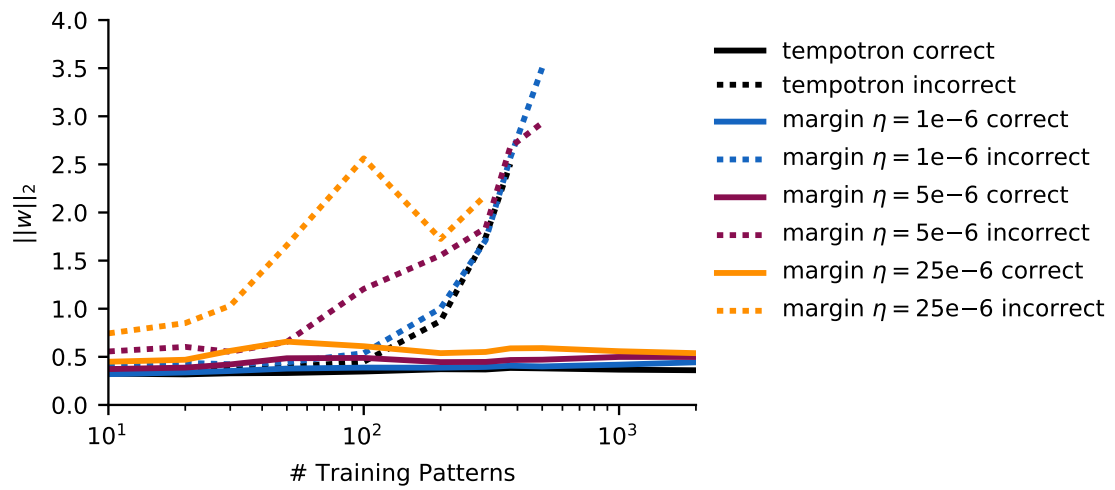


Figure 4.6: Euclidean (L2) norm of the synaptic weight vector separated by generalization performance.

4.4 WEIGHT DECAY AND RESCALING

The margin definition for a linear support vector machine (see section 2.1.1) is

$$d = \frac{2}{\|\vec{w}\|} \quad (4.5)$$

It describes the distance between two hyperplanes parallel to the decision surface that are defined by the closest training data points, the support vectors. To maximize the margin d one has to minimize the Euclidean norm of the weight vector \vec{w} while correctly classifying all training vectors.

So far the margin learning rule only includes the aspect of maximizing the margin in the spike-threshold-surface but disregards the weight vector minimization that comes naturally from the margin definition for support vector machines.

Limiting the total synaptic strength, or norm, (Gütig, 2016; R. Rubin, L. F. Abbott, and H. Sompolinsky, 2017) or introducing a weight decay (Gütig, 2016), in machine learning often called Tikhonov Regularization, has been shown to increase robustness and stability of neural classifiers. Based on these results and the motivation from machine learning we introduce two methods to combine margin learning with reduction of the Euclidean norm of the synaptic efficacies. The first one uses weight decay while the second one, weight rescaling, operates on the spike-threshold-surface.

Both methods extend the margin learning rule in a similar way. Only if a margin LTP synaptic update was applied for a pattern with k output spikes and its ϑ_k^* is further away from the natural threshold than ϑ_{k+1}^* , meaning the center of the k -th plateau in the spike-threshold-surface is above the firing threshold, a weight decay or rescaling step is executed.

The weight decay step is defined as an attenuation by factor λ

$$\vec{w} \mapsto \lambda \vec{w} \quad (4.6)$$

By only applying weight decay after LTP steps when the center is above the firing threshold ensures that, if the attenuation is not too strong, it moves the center of the plateau closer to the threshold and not away from it.

Weight rescaling directly uses the positions of ϑ_k^* and ϑ_{k+1}^* to center the margin plateau on the natural threshold. This is possible since

multiplying the synaptic weights with a factor α is equivalent to scaling the threshold and its resulting voltage trace.

$$V_\alpha(t) = \alpha \left(\sum_{i=1}^N w_i \sum_{t_i^j < t} K(t - t_i^j) - \vartheta \sum_{t_s^j < t} e^{-\frac{(t-t_s^j)}{\tau_m}} \right) \quad (4.7)$$

$$= \sum_{i=1}^N \alpha w_i \sum_{t_i^j < t} K(t - t_i^j) - \alpha \vartheta \sum_{t_s^j < t} e^{-\frac{(t-t_s^j)}{\tau_m}} \quad (4.8)$$

By multiplying the synaptic weights with α we can rescale the spike-threshold-surface of the current pattern. Based on this we define the weight rescaling step to center the current margin plateau on the natural threshold ϑ .

$$\vec{w} \mapsto \vec{w} \frac{\vartheta}{\vartheta_{k+1}^* + 0.5(\vartheta_k^* - \vartheta_{k+1}^*)} = \vec{w} \frac{2\vartheta}{\vartheta_{k+1}^* + \vartheta_k^*} \quad (4.9)$$

Combining weight decay and rescaling with our margin learning method gives us the following margin learning rule variants:

- **Margin:** margin learning without weight decay or rescaling.
- **Margin + decay:** margin learning with weight decay.
- **Margin + rescaling:** margin learning with weight rescaling.
- **Margin momentum + decay:** margin learning with weight decay where the margin weight updates are included in the tempotron momentum heuristic (see section 2.4.4).

Additionally we define a learning rule, **margin up with rescaling**, that only relies on LTP steps to increase the margin and uses weight rescaling to center the plateau of the current output spike count. Its learning rule is as follows:

1. If the current input pattern does not generate the desired amount of output spikes the normal multi-spike tempotron learning rule is used.
2. Only when the neuron classifies the current spike pattern correctly we switch to margin up learning:
 - a) If the current pattern has a label of zero and $\vartheta - \vartheta_{k+1}^* < \kappa_{\text{train}}$ an LTD step is used to move ϑ_{k+1}^* further away from the threshold.

$$\Delta \vec{w} = -\eta_{\text{margin}} \vec{\nabla}_{\vec{w}} \vartheta_{k+1}^* \quad (4.10)$$

- b) If the current pattern has a label above zero and $\vartheta_k^* - \vartheta < \kappa_{\text{train}}$ an LTP step is used to move ϑ_k^* further away from the threshold.

$$\Delta \vec{w} = \eta_{\text{margin}} \vec{\nabla}_{\vec{w}} \vartheta_k^* \quad (4.11)$$

- c) If an LTP step was done and the center of the current plateau is above the threshold $(\vartheta_{k+1}^* + \vartheta_k^*)/2 > \vartheta$ a weight rescaling step is done.

$$\vec{w} \mapsto \vec{w} \frac{2\vartheta}{\vartheta_{k+1}^* + \vartheta_k^*} \quad (4.12)$$

In the next section (4.5) we will optimize the learning parameters of these margin learning rules and take a look at the noise robustness and generalization performance. But first we will study the margin up and rescaling learning rule and apply it to the same task as the pure margin learning rule in figures 4.4, 4.5 and 4.6. The results are shown in the following figures 4.7 and 4.8. The margin learning rate was set to $\eta_{\text{margin}} = 25 * 10^{-6}$, the value with the best generalization performance for pure margin learning, for both margin learning rules.

The results show that the margin up with rescaling learning rule increases the average minimal margin over the multi-spike tempotron albeit not as strongly as for the pure margin learning. The Euclidean norm of the synaptic efficacies is also decreased when compared to tempotron and pure margin learning.

The noise robustness and generalization performance is also increased. When 5% of input spikes are randomly added and removed from the trainings data set the margin up with rescaling learning rule is still able to correctly classify all patterns even for very small trainings pool sizes ($N \geq 20$). Similarly the resulting synaptic efficacies of the neural classifier are able to correctly classify test patterns that were not used during training. For $N \geq 20$ the neuron successfully generalized without any errors. This is over an order of magnitude less training patterns than the pure tempotron or margin learning rule require to achieve the same low generalization error.

To minimize the impact of background noise in front of and directly after a feature in an input spike pattern the neurons response should learn to fire near the end of the feature but also not outside of it. The multi-spike tempotron learning rule as well as pure margin learning both have simulation runs in which the neuron learned to elicit spikes shortly after onset of the feature or directly after offset. Margin up with rescaling improves this, all runs learned to fire inside the target

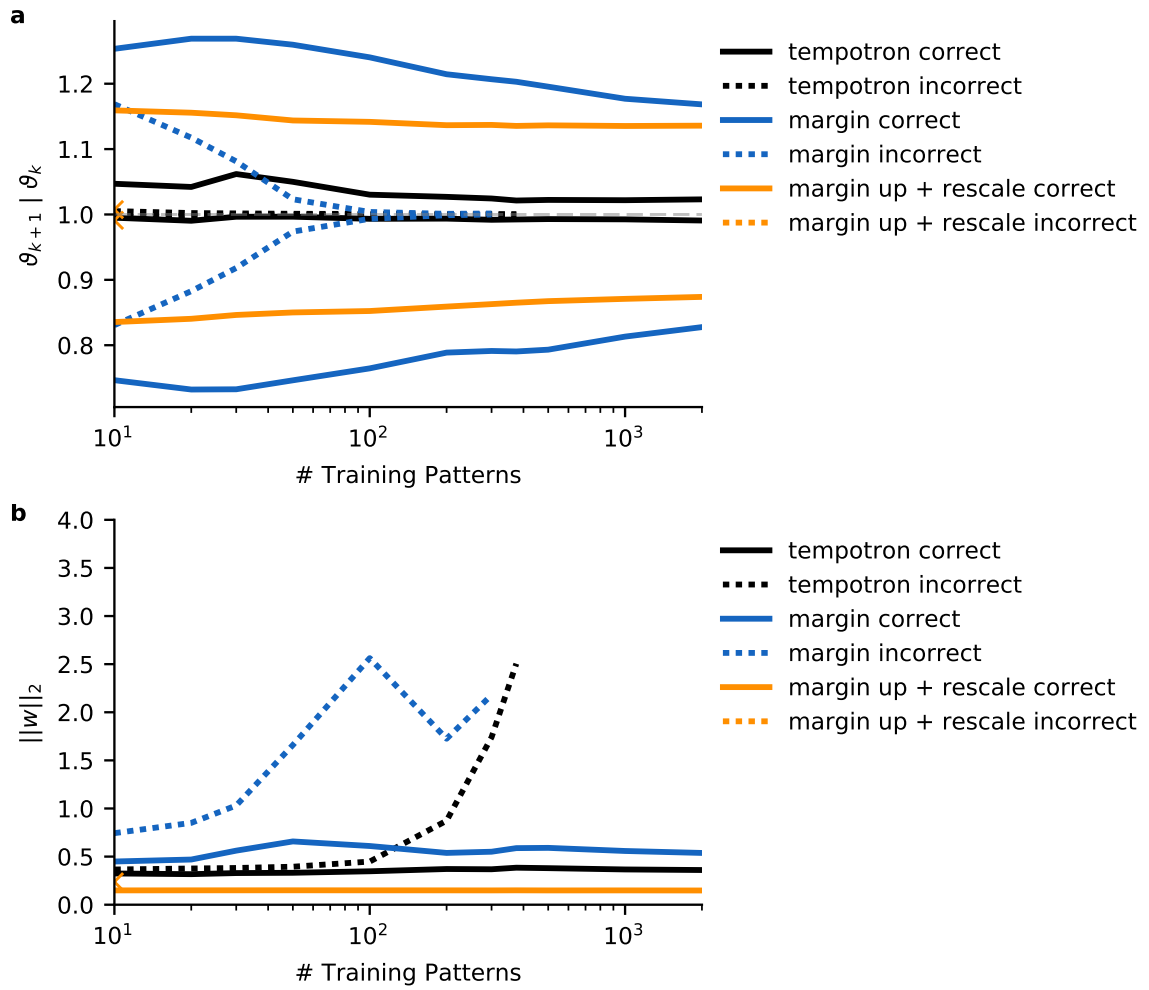


Figure 4.7: (a) Average minimal margin distance for tempotron, margin and margin up with rescaling learning rules. Dashed lines indicate the data points for runs that did not learn to detect the target feature. Margin up with rescaling only has such runs for 10 training patterns, its values are marked with 'x'.

(b) Euclidean norm of the synaptic efficacies for all three margin learning algorithms.

feature and on average further away from both feature on- and offset (see figure 4.9).

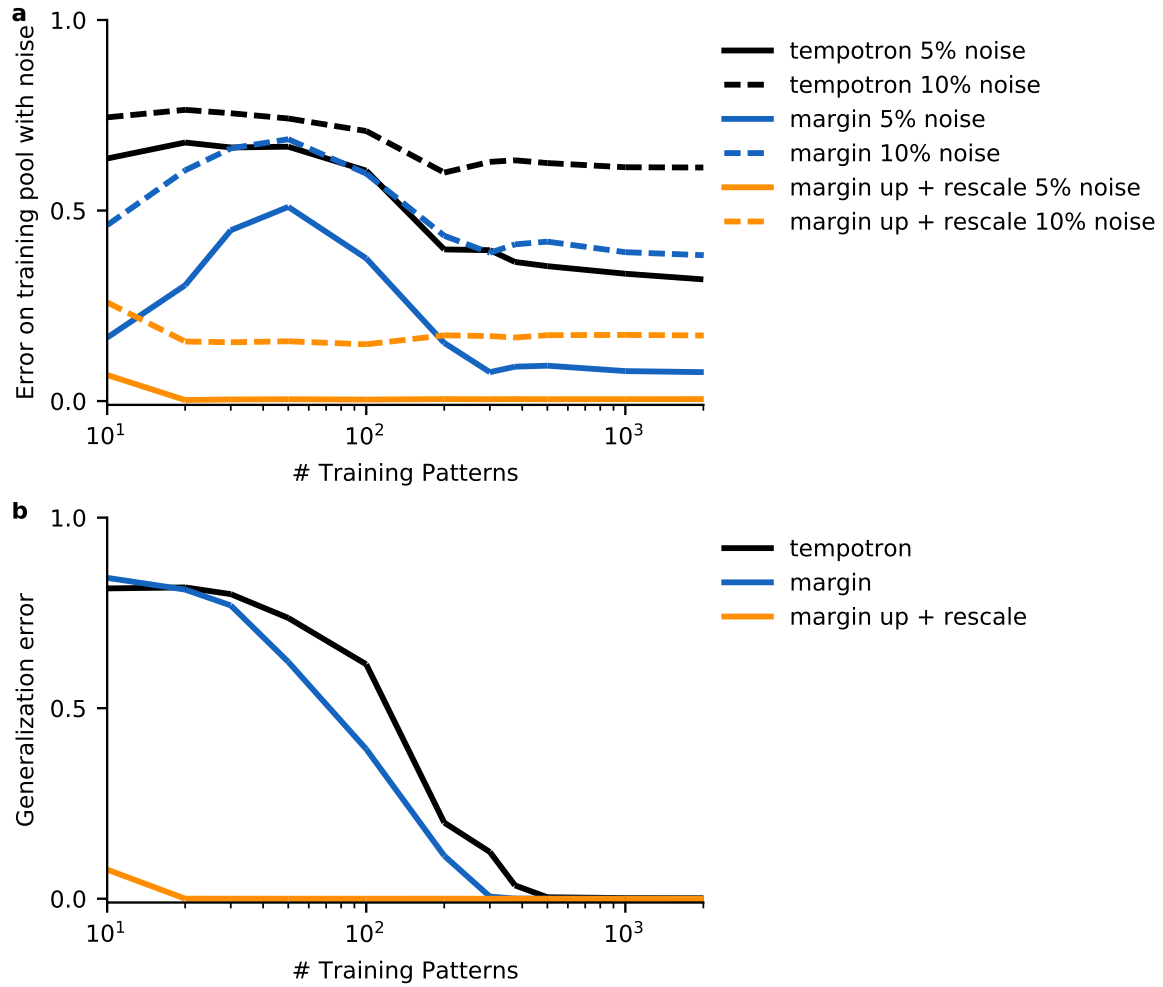


Figure 4.8: (a) Robustness to noise added to the training patterns after learning. We again measure performance under 5% and 10% removal and addition of spikes. (b) Generalization performance measured on test patterns which have not been used during training.

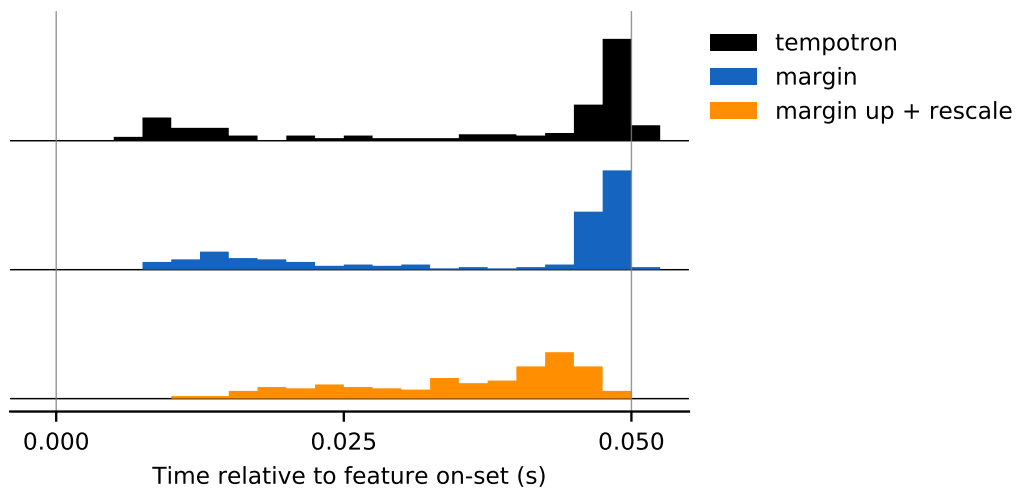


Figure 4.9: Relative position of output spike times inside the target feature for different learning rules. Histogram of mean output spike times relative to target feature for each simulation run. Target feature duration is 50ms, on- and off-set of target feature are marked with thin vertical lines.

4.5 GENERALIZATION PERFORMANCE UNDER OPTIMAL MARGIN LEARNING PARAMETERS

A crucial part of learning is the ability to generalize from a limited amount of stimuli to similar, yet not identical, stimuli. The simulations in the previous section use the embedded feature task and in both, training and test data, the pure feature, together with distractor features, were embedded in background noise. Every instance of the feature has the exact same pattern of input spikes. This means the generalization performance so far measured the ability to detect the feature and not how well the detector was able to generalize to new previously unseen feature instances.

To test the generalization performance of our margin learning algorithms to new feature instances we modified the synthetic embedded feature task by randomly removing and adding 25% of the input spikes (see section 2.5.1). This ensures that the target features embedded in patterns of the training pool and in the test set are all different but still instances of the same template feature. The generalization performance now measures how well the neuron generalizes from the training feature instances to new instances of the template feature.

Additionally we change how the neuron models efficacies are initialized before training. To reduce the influence of the initial random weights we use a different pre-training algorithm based on rescaling the synaptic efficacies to a 5Hz output rate while keeping a zero standard deviation of the weights (see section 2.4.5).

To compare the best case scenario for all learning rules we optimize the various parameters for each margin learning rule (tempotron learn step size, margin learn step size, decay factor) with respect to a generalization performance score. This score is calculated as the average generalization error across training pool sizes of $N = 10, 20, 30, 50, 100, 200$ patterns. Each parameter set and pool size combination is evaluated with 100 different initial seeds. Simulations were limited to maximum of 500 training cycles. Optimizing for these limited training pool sizes is a trade off as parameter sets that work well for small pool sizes might not work as well for bigger pool sizes outside the optimized range ($N > 200$) and vice versa. Since we are interested in the generalization performance that can be achieved for very small pool sizes and optimize for these, the results for larger pools are shown for completeness but should not be mistaken for optimal performance at these values of N .

Figure 4.11 (a), (b) and (c) show three optimization runs for the margin up and rescaling learning rule. We let the algorithm optimize

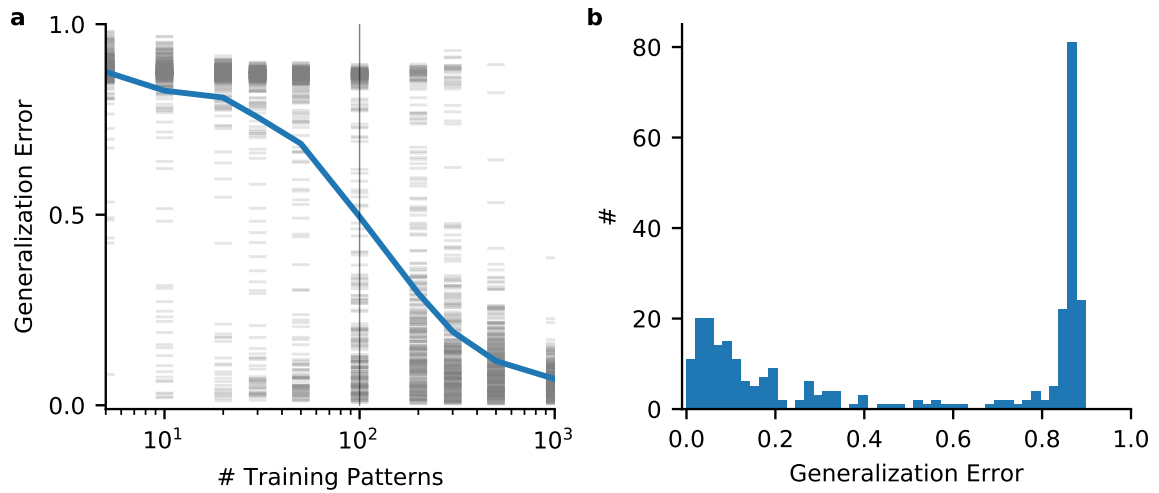


Figure 4.10: Bimodal distribution of generalization performance.

(a) Generalization error for individual simulation runs at different training pool sizes for the unmodified multi-spike tempotron learning rule. The vertical gray line indicates where the bi-modal distribution of raw data points shown in (b) is taken from.

learn step size η , margin learn step size η_{margin} and requested margin κ_{train} from three different initial conditions. The lines show the progression of the currently optimal parameter set while the dots show explored parameter sets that did not yield a performance improvement. The optimization runs resulted in the following parameter pairs. The standard error of the mean was calculated by treating the average results from one seed across all pool sizes as one score value and estimating based on this across all seeds.

run	η	η_{margin}	κ_{train}	score
1	$2.92\text{e-}05$	$11.87\text{e-}04$	0.129	0.206 ± 0.019
2	$3.11\text{e-}05$	$8.23\text{e-}04$	0.283	0.209 ± 0.021
3	$3.67\text{e-}05$	$9.39\text{e-}04$	0.452	0.213 ± 0.018

The raw results in figure 4.11 (a), (b) and (c) together with the end parameter sets in the table show that while they all reached a similar performance score their end values for κ_{train} varies. κ_{train} needs to be above ≈ 0.1 to reach a close to optimal generalization performance score but higher values of κ_{train} have a small detrimental effect. For all following simulations we will set κ_{train} to infinity to simplify the parameter optimizations.

Optimizing tempotron step size η , margin step size η_{margin} and attenuation factor $\lambda_{\text{attenuation}}$ for all margin learning variants results in the generalization performance and mean minimum margin shown in figure 4.12.

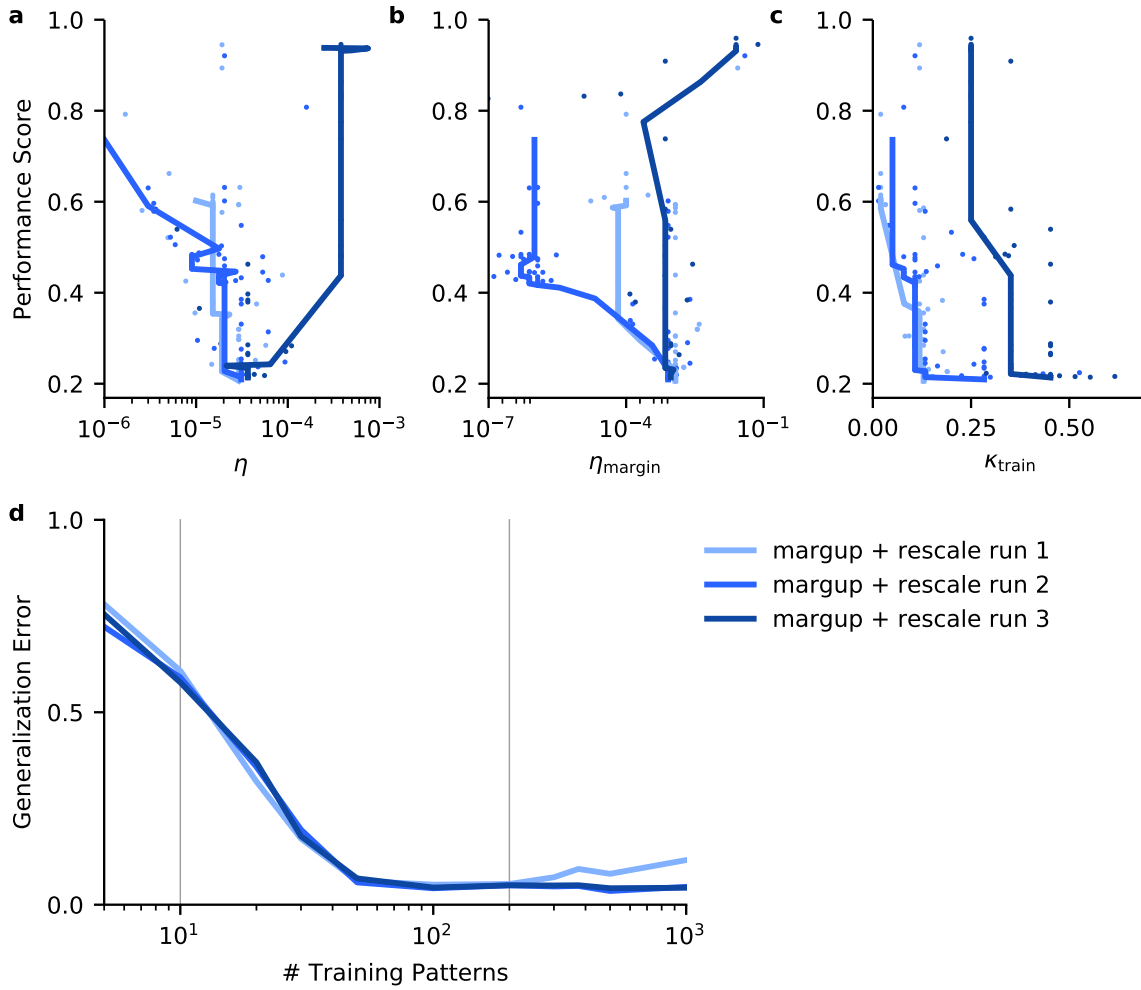


Figure 4.11: Parameter optimization runs for three different initial conditions.

(a+b+c) Lines show progression of current optimal parameter set. Dots show individual parameter sets that were explored during optimization but did not yield a better performance score. Low scores are better. η is the learn step size for multi-spike tempotron synaptic weight updates, η_{margin} is the same for margin weight updates and κ_{train} is the requested margin.

(d) Generalization performance for the final parameter sets at the end of their optimization runs. Area between thin vertical lines depicts which training pool sizes are used to calculate the performance score ($N = 10, 20, 30, 50, 100, 200$). As can be seen parameters optimized for this range are not necessarily optimal for training pool sizes outside of it.

The pure margin learning algorithm is unable to improve generalization performance over tempotron learning but increases the minimal margin reliably. If no margin learning steps are done and we only add weight rescaling to the learning rule the generalization performance is improved. Training patterns required for 50% generalization performance is decreased by a factor of three. As expected, without margin learning steps, the average minimal margin is only slightly above tempotron learning. Margin up with rescaling offers

the highest generalization performance for small amounts of training patterns, only needing above ten patterns for 50% generalization error instead of one hundred required by the tempotron. An improvement of nearly an order of magnitude. The margin learning rule with decay that includes the margin synaptic weight updates in the momentum heuristic also offers similar improvements in generalization performance and minimum margin.

The optimized parameters for all margin learning variants are listed in the following table.

run	η	η_{margin}	$\lambda_{\text{attenuation}}$
tempotron	2.13e-05		
margin	1.45e-05	5.07e-06	
rescaling	4.20e-05		
margin + decay	1.97e-05	2.70e-04	0.99109
margin + rescaling	2.63e-05	2.34e-04	
margin momentum + decay	1.25e-04	1.69e-06	0.98079
margin up + rescaling	4.73e-05	7.06e-04	

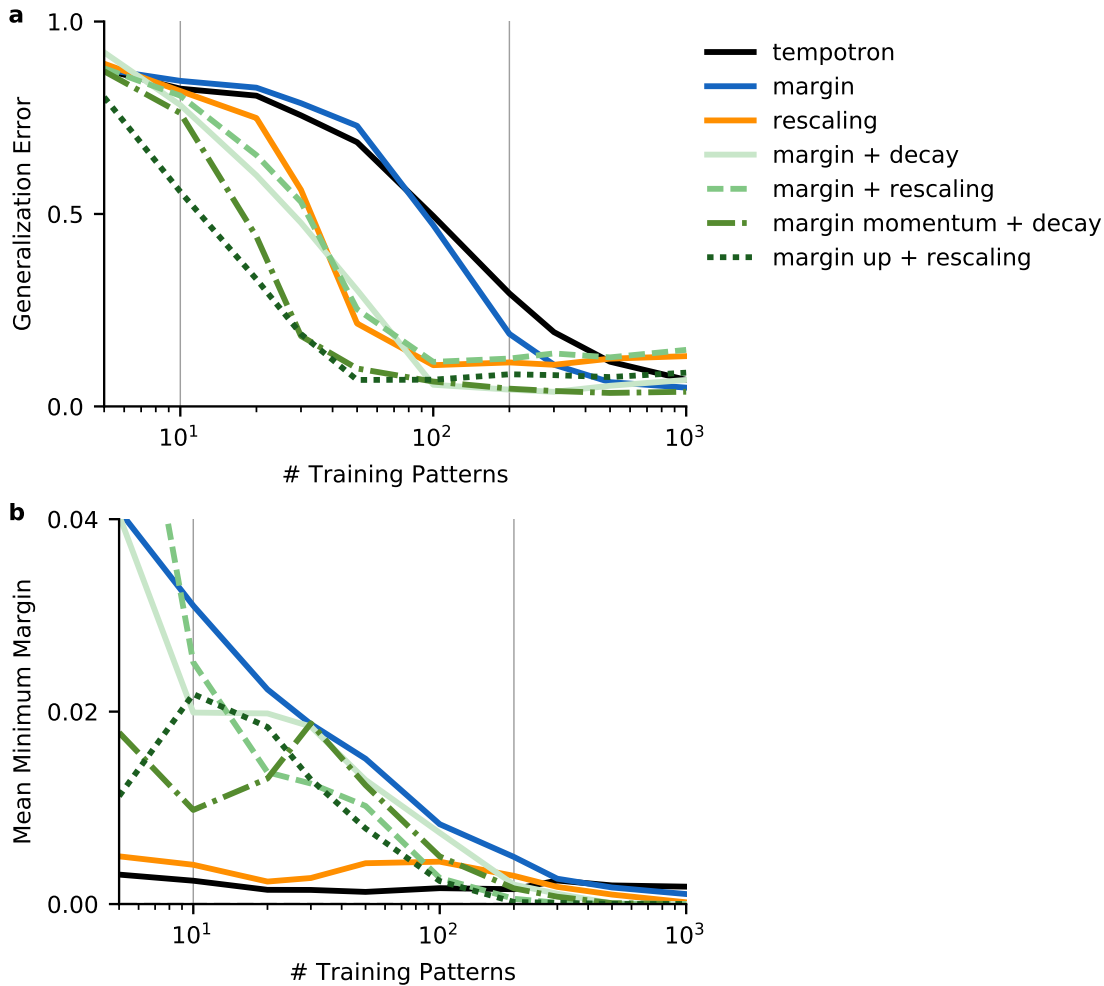


Figure 4.12: Mean generalization error and mean minimum margin for the multi-spike tempotron learning rule and margin learning variants. Shown are the results of optimized parameter sets. Area between thin vertical lines depicts training pool sizes for which the generalization performance was optimized.

(a) Generalization error for each simulation is measured by checking the classification correctness on a test data set of input patterns that were generated with the same process as training patterns but not used during training.

(b) Minimum margin on training batch averaged across simulation runs. With increasing training pool size the achievable minimal margin decreases due to the increasing amount of target feature instances and background noise limiting the possible solution space.

4.6 MARGIN BASED OPTIMIZATION

To optimize the parameters of the margin learning rules for optimal generalization performance we used the performance on the test data set directly as feedback for the optimization algorithm. It is unlikely that a neuron has access to test data to estimate its generalization performance.

An advantage of the margin learning rules is that they continue learning after the training error reached zero. When the training error is zero the multi-spike tempotron stops changing the synaptic efficacies while the margin learning rules continue to widen the margins on the training patterns. To study how access to the margins on training patterns can help with judging the classifiers generalization performance without using test data we limit the performance score for parameter optimization to training error and average margin.

For the simulations we use the synthetic embedded feature task with 25% noise. We calculate a performance score at the same training pool sizes ($N = 10, 20, 30, 50, 100, 200$) but instead of the average generalization error we use the average fraction of incorrectly classified training patterns for optimizing the pure multi-spike tempotron and the average margin for the margin learning rule with momentum and decay. The average margin is defined as the average κ across all training patterns and simulation runs. This includes negative values of κ to penalize for incorrectly classified training patterns. Not unlike the concept behind soft margin classifiers (Cortes and V. Vapnik, 1995) that penalizes incorrectly classified input vectors with the distance to the decision surface (see section 2.3.1). The score is calculated as $1 - x$ where x is the average margin since the optimization searches for a minimal score. After parameter optimization finishes we measure the generalization performance on a test data set as before.

To validate the use of this mean margin score as a proxy for generalization performance we tracked both scores during the margin learning optimization runs and calculated a Pearson correlation coefficient of 0.81. Figure 4.13 shows the raw data points for all parameter pairs explored.

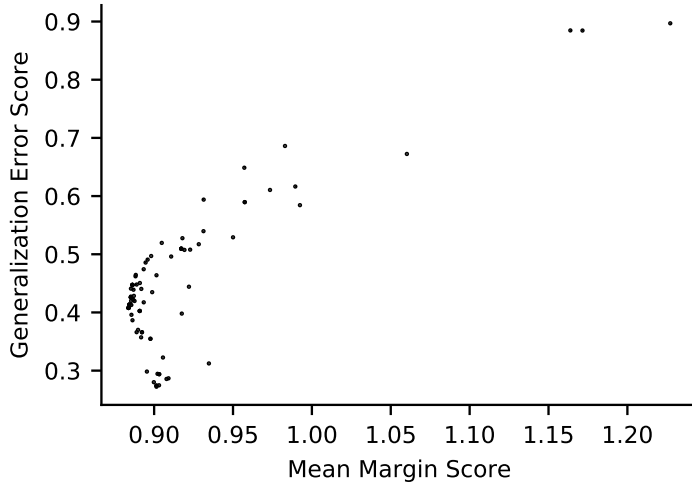


Figure 4.13: To validate the use of the mean margin as a proxy for generalization performance we measured both generalization error score, as used in the previous optimizations, and the average margin score ($1 - x$ with x being the average margin) for all parameter pairs encountered during optimization for margin learning. Lower scores are better. While the margin can not perfectly predict generalization performance lower margin scores do strongly correlate with lower generalization error scores resulting in a Pearson correlation coefficient of 0.81.

run	η	η_{margin}	score
tempotron 1	$3.38\text{e-}05$		0.00 ± 0.0
tempotron 2	$2.85\text{e-}05$		0.00 ± 0.0
tempotron 3	$4.11\text{e-}06$		0.00 ± 0.0
margin momentum + decay 1	$1.09\text{e-}04$	$9.00\text{e-}05$	0.8866 ± 0.0042
margin momentum + decay 2	$1.16\text{e-}04$	$5.99\text{e-}05$	0.8835 ± 0.0046
margin momentum + decay 3	$1.35\text{e-}04$	$3.54\text{e-}05$	0.8851 ± 0.0038

For the pure tempotron we optimized the learn step size η for three different initial conditions. Since the tempotron learns to correctly generate the output spikes for all training patterns over a broad range of η values its final parameter set also varies widely (figure 4.14 (a)). This also explains the varying generalization performance of the resulting parameter sets (figure 4.14 (c)).

Similarly we optimize the learning step size for tempotron η and margin steps η_{margin} for margin learning with momentum and decay using the minimal margin distance as performance measure. We fixed the decay attenuation factor $\lambda_{\text{attenuation}}$ to 0.98, the value obtained from optimization determined in the previous chapter. Leaving $\lambda_{\text{attenuation}}$ as a free parameter leads to it being decreased towards zero as the margin learning rule without any decay is able to achieve a higher average minimal margin (see figure 4.12). This is an obvious

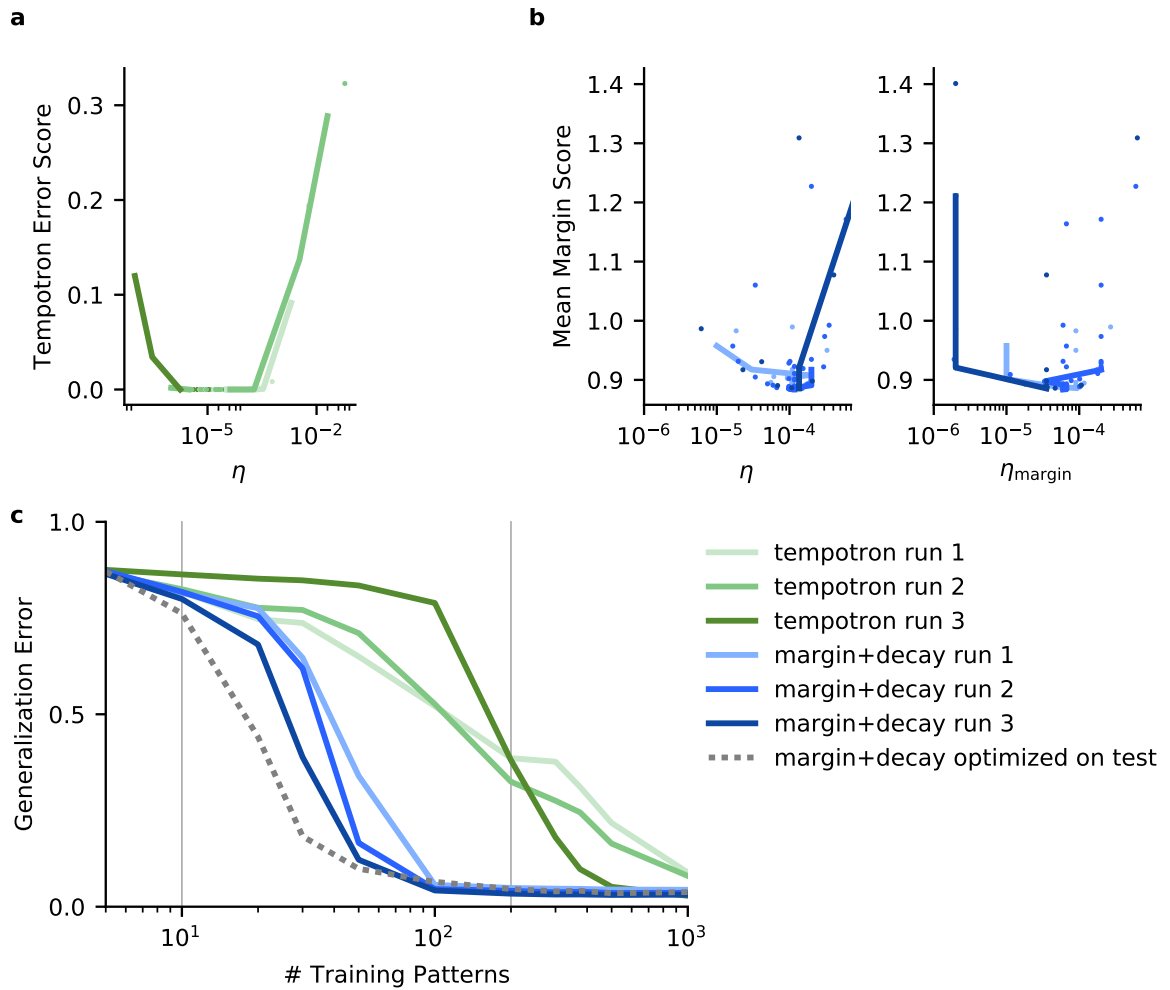


Figure 4.14: Tempotron error and average minimal margin based parameter optimization and resulting generalization performance.

(a) Optimization of learning step size η for three different initial conditions of the pure multi-spike tempotron. The performance score is based on the average fraction of incorrectly classified training patterns across multiple training pool sizes (area between thin vertical lines in (c), $N = 10, 20, 30, 50, 100, 200$). Over a wide range of η values the training error reaches zero for all simulation runs giving ambiguity to what is the optimal value for η (see (c)).

(b) Similar optimization for the margin momentum with decay learning rule. Performance score is based on the average margin across the same training pool sizes. Score is calculated as $1 - x$, this means 1.0 is equivalent to an average minimal margin of zero.

(c) Generalization performance of the resulting parameter sets. Gray dotted line represents the result from the parameter optimization based on generalization performance.

drawback of using the average margin as a performance measure and further research into other objective functions is required.

The optimized parameter sets for the margin learning rule are more clustered together and offer more stable and predictable generaliza-

tion performance. With only access to the average margin during training margin learning is able to successfully improve the generalization performance and is able to cross 50% average generalization error for a factor of 5 less training patterns when compared with multi-spike tempotron learning.

APPLICATION TO SPEECH RECOGNITION

5.1 AUDITORY BRAIN-STEM MODEL

To apply the neural learning rules to a task of phoneme recognition we developed an auditory brain-stem model to generate spike patterns from sound signals. This model is based on in-vivo measurements of neurons in the inferior colliculus of various mammals and mimics observed characteristic spike responses to varying sound stimuli. These responses vary in frequency tuning, loudness thresholds and required temporal structure of the sound signal.

Previously used auditory brain-stem models (Gütig, 2016; Gütig and Haim Sompolinsky, 2009; John J Hopfield and Brody, 2000) only implement a subset of neurons of the inferior colliculus described in the literature: neurons reacting to sound onset and offset. Since we do not know which features in the sound input are used for the discrimination of different human speech sounds we extended the brain-stem model to include additional more complex response types that cover frequency ranges and time spans possibly relevant for phoneme detection. These can be described as short-, long- and bandpass filter neurons as well as neurons that exhibit sustained firing behavior.

These different response types of neurons are mimicked by first separating the sound signal in the frequency domain using Fast-Fourier-Transform to allow for different frequency tuning. The resulting signal in each frequency channel can then be used to generate output spikes conditioned on a desired loudness threshold and temporal structure.

5.1.1 *Typical Responses of Auditory Neurons*

As a first step of neural auditory processing the cochlea executes a spectral decomposition of the incoming signal and splits it up into frequency specific neural channels, individual fibers of the auditory nerve (Harrison, 2001; Ruggero, 1992). This topographic representation of sound frequency, called tonotopy, then projects through the vestibulocochlear nerve and associated midbrain structures to the au-

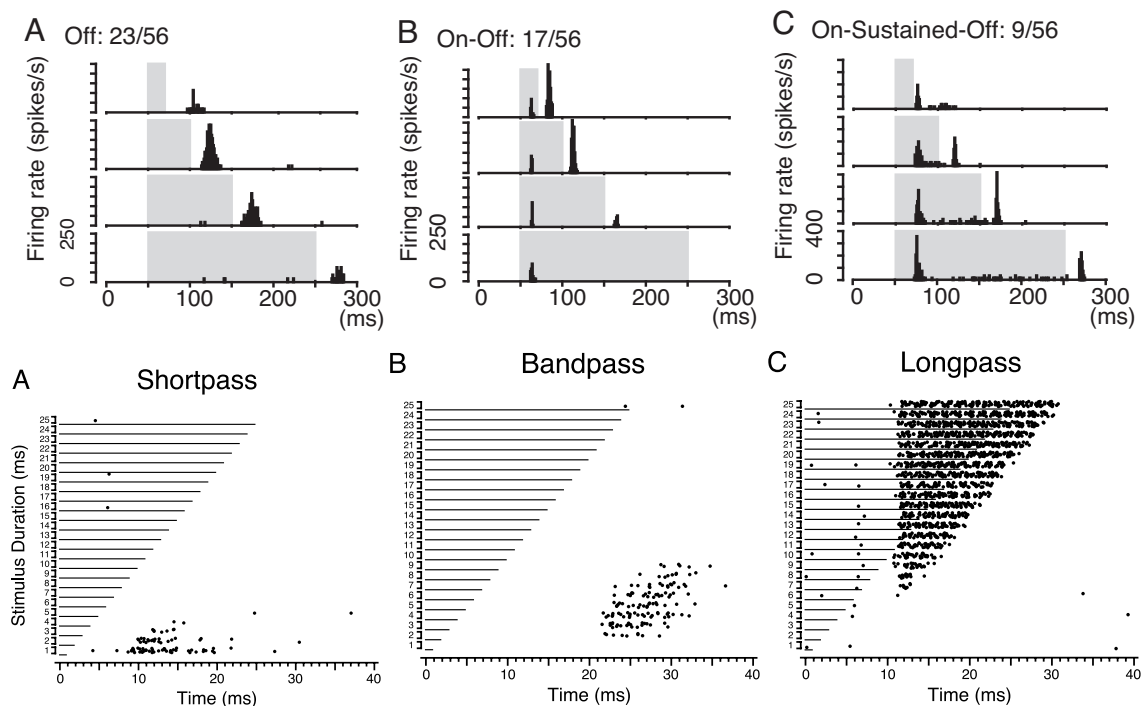


Figure 5.1: Different pure tone response types of auditory neurons measured in vivo in the inferior colliculus of mice. Different types of duration-tuning and response patterns can be observed. Top row from (Kasai, Ono, and Ohmori, 2012). Bottom row from (Faure et al., 2003).

auditory cortex. Along this pathway the inferior colliculus is the principal midbrain nucleus. Major ascending auditory pathways converge here and it appears to be an integrative station, is involved in the integration and routing of multi-modal sensory perception and might be responsible for detection of pitch (Shore, 2010).

Pure tone experiments show that typical neurons in the inferior colliculus have a best frequency at which they are most responsive (Phillips, S. Hall, and Boehnke, 2002). Additionally to the best frequency inferior colliculus neurons, measured in bats and other animals, also show tuning to different loudness thresholds (J. Casseday and Ellen Covey, 1992). And even more complexly shaped excitatory and inhibitory response areas to frequencies and sound pressure levels have been measured in mice (Egorova et al., 2001).

Duration tuned inferior colliculus neurons have been measured in many animals, in rats (Perez-Gonzalez et al., 2006), mouse (Brand, Urban, and Grothe, 2000), chinchilla (Chen, 1998) and many different bats (J. Casseday, Ehrlich, Covey, et al., 1994; Daphna Ehrlich, John H Casseday, and Ellen Covey, 1997; Z. Fuzessery and J. Hall, 1999; Zoltan M Fuzessery, 1994; P.-S. Jen and R. Feng, 1999; Pinheiro, Wu, and Philip H-S Jen, 1991).

These duration tuned neurons can be described as shortpass, longpass and bandpass filters of the input signal at the neurons best frequency and loudness level (see figure 5.1). Measurements show that duration tuned neurons can fulfill crucial roles for mating calls in frogs (Potter, 1965) or echo location calls of bats (Galazyuk and A. S. Feng, 1997).

The neurons response type to their preferred frequency, loudness and temporal structure can also vary (figure 5.1). Some neurons react to stimulus offset, some to onset and others are observed to elicit a sustained stream of output spikes during stimulus presence (Faure et al., 2003; Kasai, Ono, and Ohmori, 2012).

According to these observed preferences of inferior colliculus neurons we modeled our auditory front-end. It contains a multitude of spike output channels mimicking inferior colliculus neurons to cover a wide range of different frequencies, loudness thresholds and temporal structures that might be relevant to phoneme recognition.

5.1.2 Auditory Front-End

To generate spike patterns from sound files we extended a previously introduced auditory front-end (Gütig, 2016; Gütig and Haim Sompolinsky, 2009). The process is based on what is described in (Gütig, 2016) and is implemented as follows:

First the raw sound samples are converted to spectrograms using the `specgram` function included in the `matplotlib` library (Jones, Oliphant, Peterson, et al., 2001). Based on the sound data sampled with 16kHz these spectrograms are generated using a 512 samples wide overlapping sliding window with 1ms resolution. The resulting spectrogram is then mapped into 32 Mel-scale frequency channels between 0Hz and 8000Hz via triangular overlapping frequency filters (see figure 5.2 (a) small red lines).

The Mel-scale (Stevens, Volkman, and Newman, 1937) is a perceptual frequency scale for which an equal distance between frequencies in mel is perceived as to be of equal distance in pitch. We use the following formula to convert frequency in Hertz f into m in mels:

$$m = 1127 \ln\left(1 + \frac{f}{700}\right) \quad (5.1)$$

These filters have their peak at the corresponding 32 Mel-scale center frequencies (55Hz, 116Hz, 180Hz, 250Hz, 325Hz, 407Hz, 495Hz, 589Hz, 692Hz, 802Hz, 921Hz, 1050Hz, 1189Hz, 1339Hz, 1501Hz,

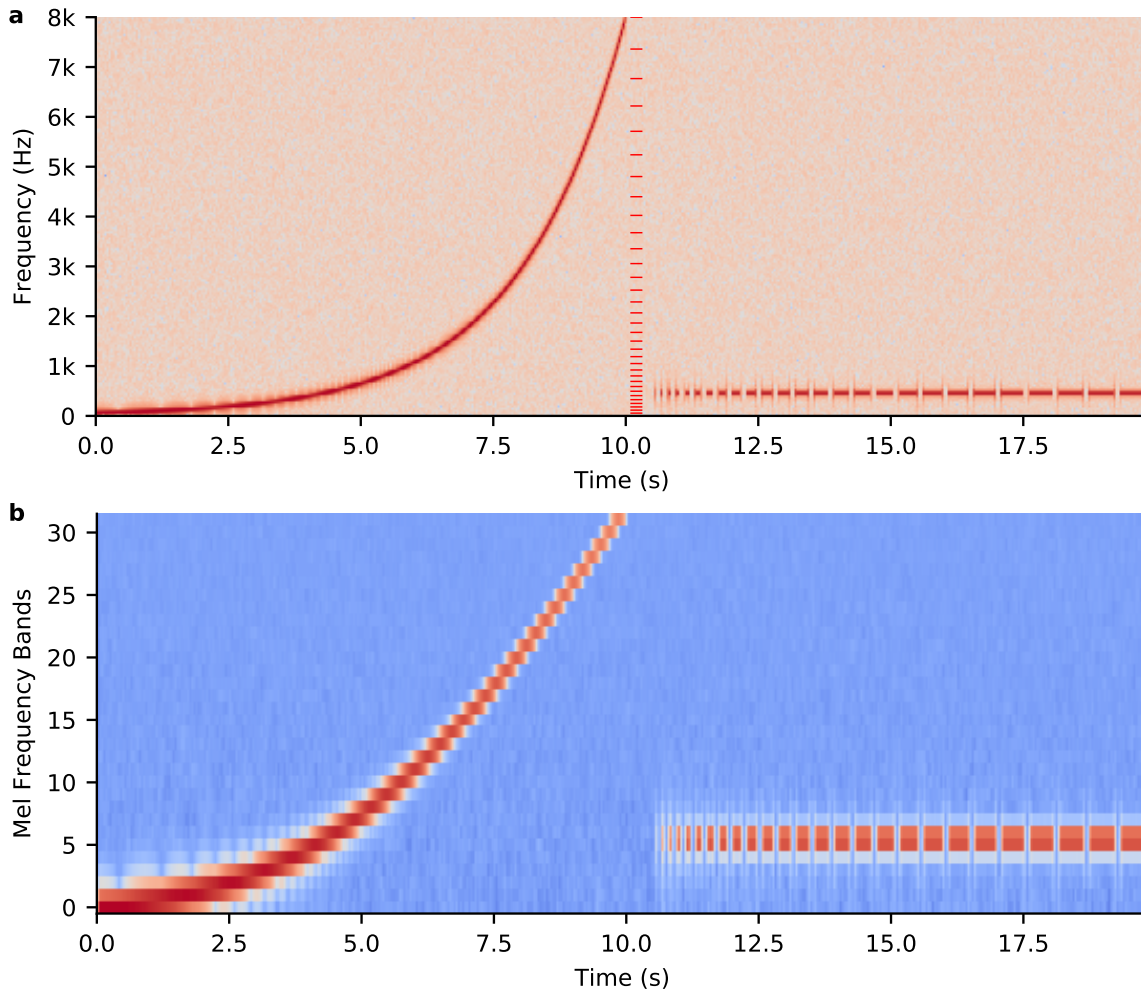


Figure 5.2: (a) Spectrogram generated by fast Fourier transform (FFT) of a probe signal to test the auditory front-end. Signal strength increases from blue to red. The first 10 seconds consist of a chirp with exponentially increasing frequency component starting at 0Hz going up to 8000Hz. The second half contains pure 440Hz tones of increasing length. On- and offset of the pure tones are smoothed out by a 10ms ramp to decrease the distortions in the FFT caused by sharp changes in the waveform derivative. The background consists of white noise with amplitude 0.005. Thin red lines mark the Mel-scale distributed frequency edges that are used to calculate the signal in the Mel-scale frequency bands.

(b) The smoothed and renormalized signal strength in the Mel-scale frequency filter bands generated from the FFT. The spike generators of the auditory front-end operate on the signal in each of these channels separately. Signal strength increases from blue to red.

1675Hz, 1864Hz, 2067Hz, 2287Hz, 2524Hz, 2780Hz, 3056Hz, 3354Hz, 3676Hz, 4023Hz, 4398Hz, 4802Hz, 5239Hz, 5710Hz, 6219Hz, 6768Hz, 7360Hz) and cut off at the corresponding neighboring higher and lower frequencies from 34 Mel-scale frequency edges (0Hz, 55Hz, 116Hz, 180Hz, 250Hz, 325Hz, 407Hz, 495Hz, 589Hz, 692Hz, 802Hz, 921Hz, 1050Hz, 1189Hz, 1339Hz, 1501Hz, 1675Hz, 1864Hz, 2067Hz,

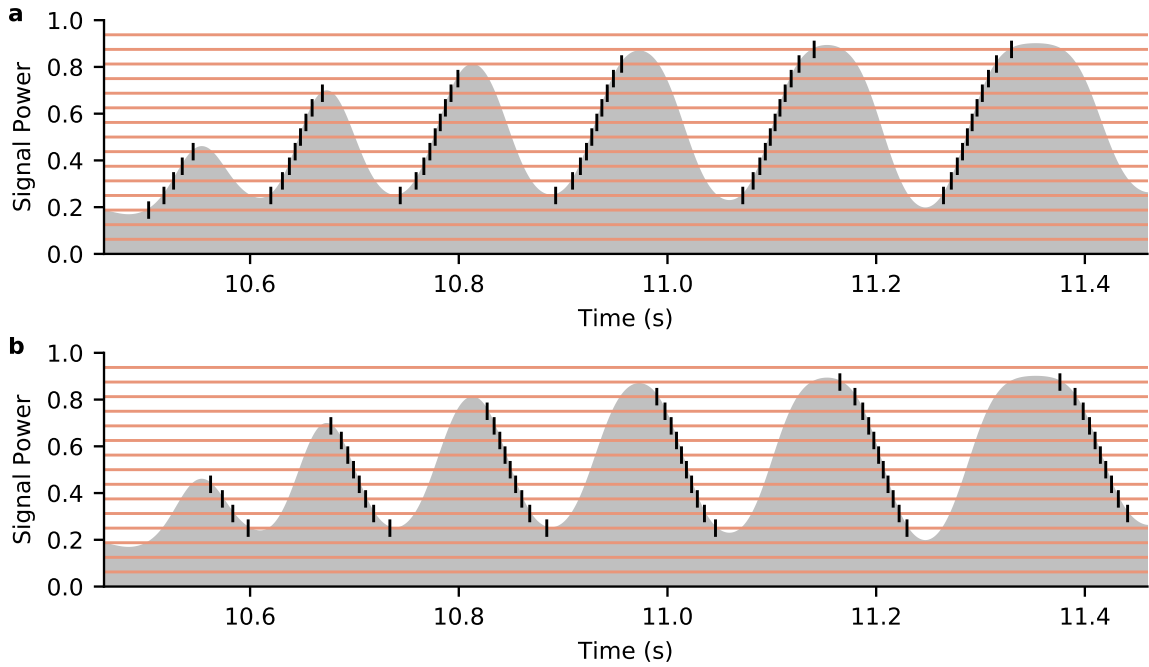


Figure 5.3: On-set (a) and off-set (b) detector output. The signal shown in gray corresponds to a Mel-scale filter band (center frequency 406.8Hz, low edge 325.4Hz, high edge 494.6Hz) that covers the 440Hz pure tone of the probe signal. In light red lines the signal threshold levels used for output spike generation are shown. On- and off-set detectors generate output spikes, shown as black vertical lines, whenever the signal level in the respective frequency band crosses its corresponding threshold level in the desired direction (from below/from above).

2287Hz, 2524Hz, 2780Hz, 3056Hz, 3354Hz, 3676Hz, 4023Hz, 4398Hz, 4802Hz, 5239Hz, 5710Hz, 6219Hz, 6768Hz, 7360Hz, 8000Hz).

The signal in each channel is first normalized by the global maximum then log-scaled ($\log(S + e) - \log(e)$, with $e = 1e-05$ being used to prevent $\log(0)$) and then again normalized by the global maximum. The resulting signals are smoothed using a Gaussian kernel with $\sigma = 1\text{ms}$, shifted by subtracting the global minimum and afterwards rescaled with the global maximum (the resulting signals can be seen in figure 5.2 (b)).

Inside the Mel frequency bands 15 different threshold levels (0.0625, 0.125, 0.1875, 0.25, 0.3125, 0.375, 0.4375, 0.5, 0.5625, 0.625, 0.6875, 0.75, 0.8125, 0.875, 0.9375) are used to determine when the signal crosses a threshold from below and above. To achieve better than 1ms resolution the threshold crossing times are estimated by linear interpolation. Based on these threshold crossings different spike generating response filters are implemented. Each of the different possible response type blocks contribute (32 frequencies \times 15 threshold levels) 480 afferents to the chosen front-end configuration.

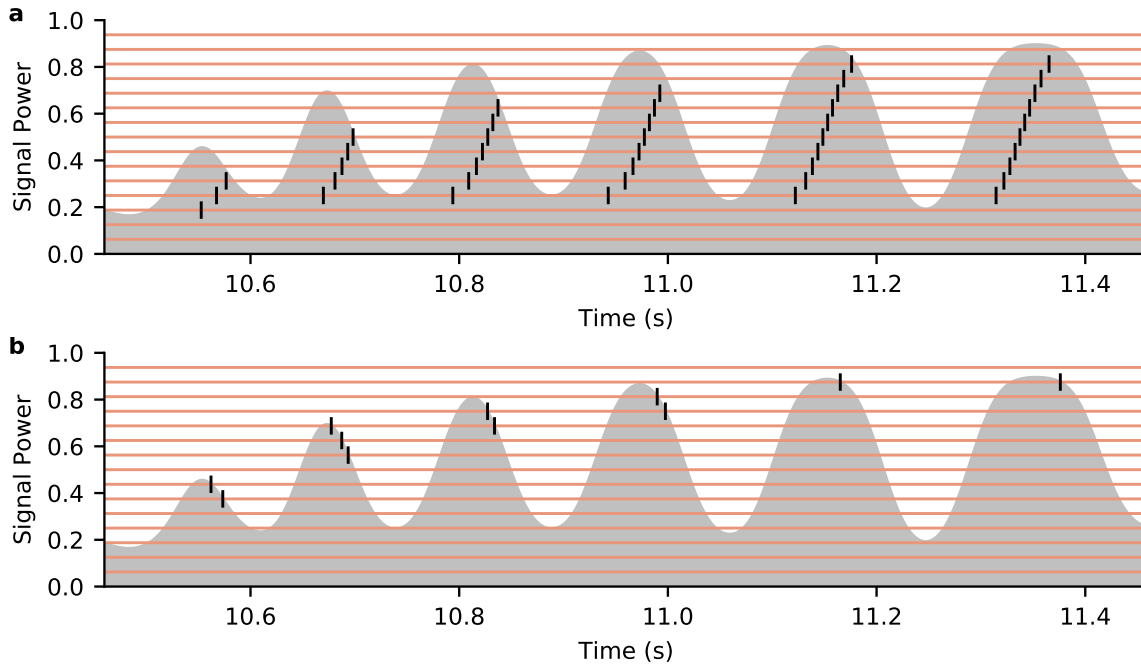


Figure 5.4: 50ms long-pass (a) and 50ms short-pass (b) detectors operate in the same way as on- and off-set detectors. Their output generation is based on threshold crossings but additionally have a temporal filter parameter that is used to only elicit output spikes when the temporal signal requirements are met. The long-pass filter only generates spikes when the signal has been above the respective threshold for a given time. Similarly for the short-pass filter the signal must have been above threshold shorter than the given parameter. The time of the output spikes is determined by causality - only at the moment where the temporal criterion is known to be fulfilled the output spikes are inserted into the output spike pattern.

Onset detectors are implemented such that each threshold crossing from below generates an output spike at its respective afferent channel corresponding to its frequency and threshold level. Offset detectors instead use the times of threshold crossings from above (see figure 5.3).

For short-, long- and bandpass response filters the above-threshold durations between signal on- and offset for a given threshold level are calculated and used as a conditional for output spike generation.

Shortpass response filters generate output spikes on threshold crossings from above only if the duration above-threshold is shorter than the given filter parameter (see figure 5.4).

Longpass filters generate a spike whenever the duration above-threshold is at least as long as the given filter duration parameter and an output spike is generated as soon as the requirement was fulfilled: at the time of threshold crossing from below shifted by the filter duration parameter (see figure 5.4).

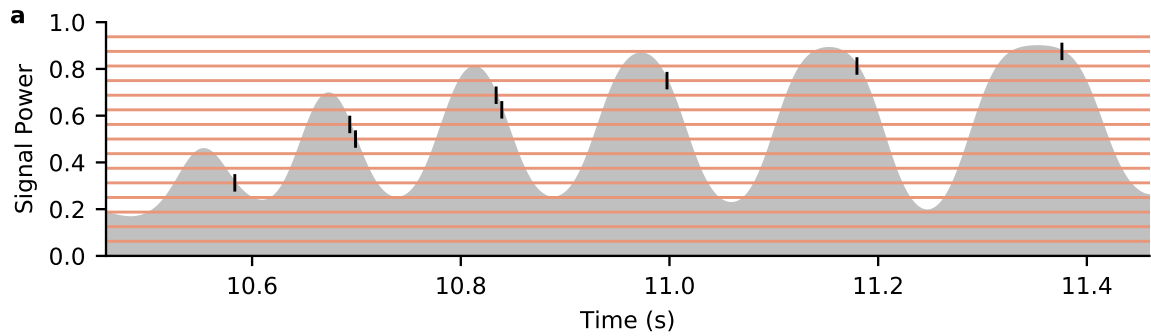


Figure 5.5: 40-60ms band-pass detector. The band-pass filter is a combination of short- and long-pass filter. It only generates output spikes when the time of the signal above the corresponding threshold lies in a certain interval.

Bandpass filters have two duration parameters and only generate output spikes if the signal stays above threshold for a duration between those two parameter values. Its output spikes are generated at the time of threshold crossing from above (see figure 5.5).

Figure 5.6 shows the Mel frequency signal strength for an example chirp sound and probe sounds of increasing duration. Panel (b) plots the raw spike output for the onset and offset detector front-end blocks. Note the synchronous line of spikes generated by the onset and offset detector at the beginning and end of the sound sample respectively. They are an artifact stemming from the fact that our sound sample must start and end at some point and is not an endless continuous stream. Accordingly these artifacts are removed before using the output spike pattern with a neural classifier.

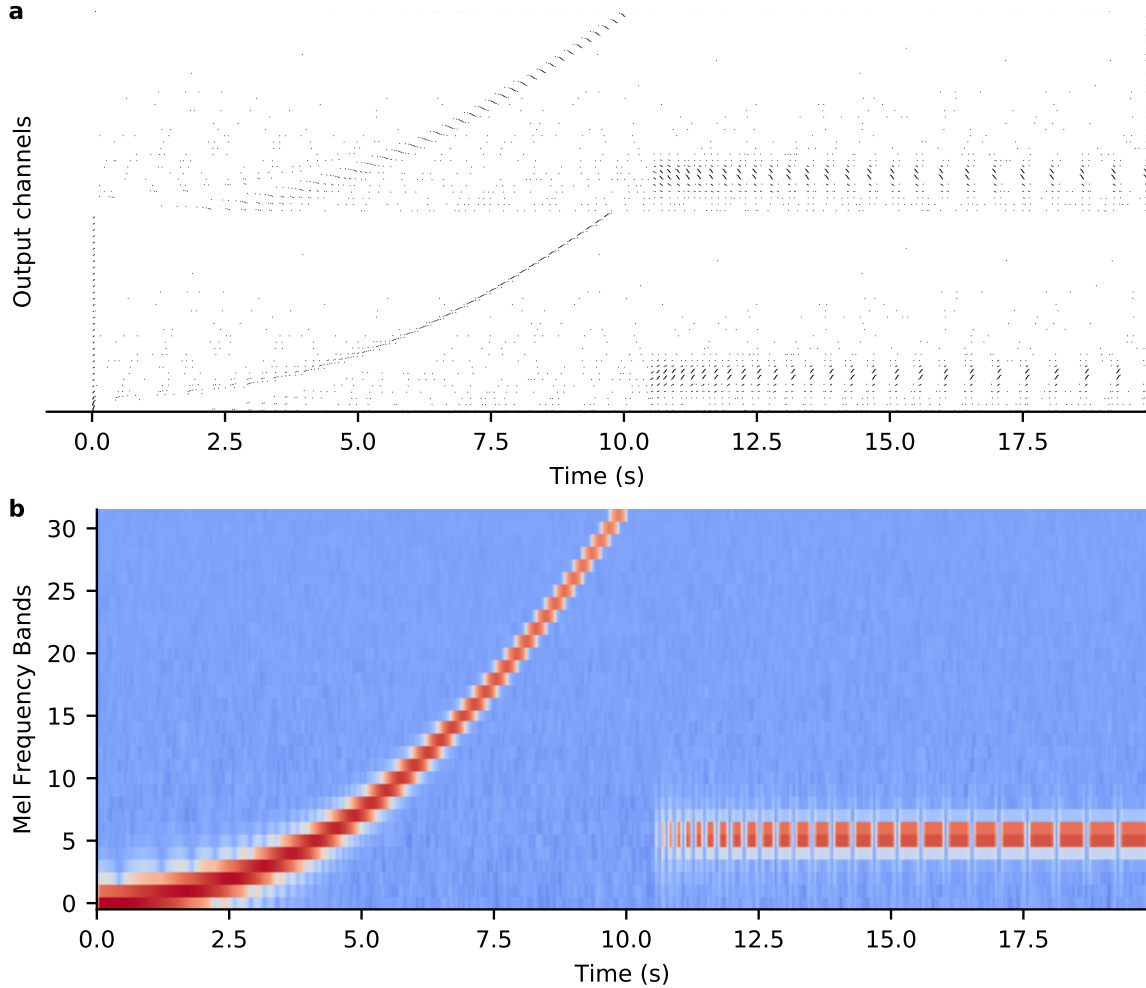


Figure 5.6: Full spike pattern output (a) for the probing signal (b) using an auditory front-end of only on-set (bottom) and off-set (top) detectors. The on- and off-set of the chirp and pure tones can be clearly seen. Due to higher noise in the lower Mel-scale filter bands the lower threshold level detectors generate more spurious spike responses. The synchronous line of spikes generated by the onset and offset detector at the beginning and end of the sound sample respectively are an artifact stemming from the fact that our sound sample must start and end at some point and is not an endless continuous stream. Accordingly these artifacts are removed before using the output spike pattern with a neural classifier.

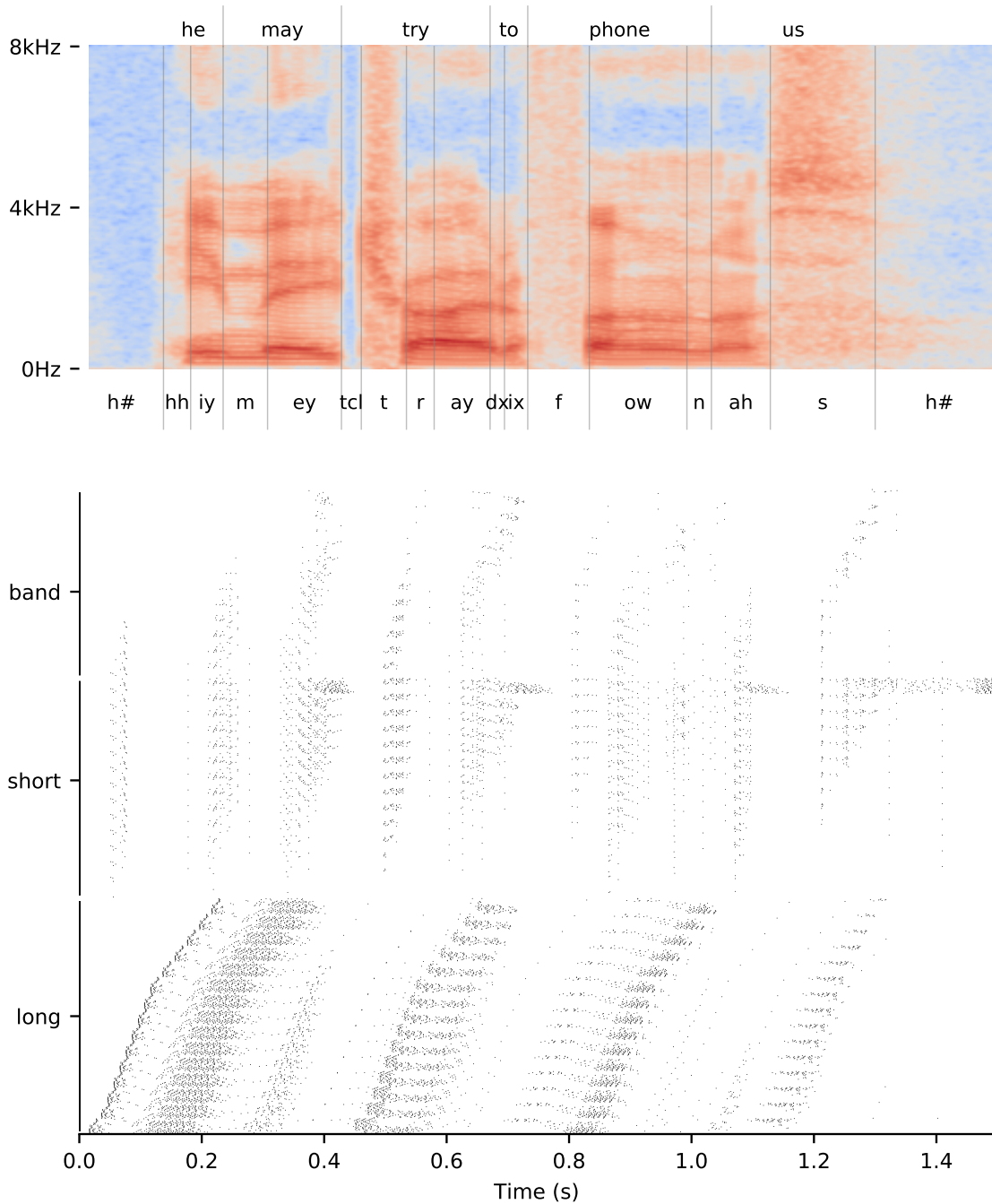


Figure 5.7: Example spike pattern generated by an auditory front-end configuration with 41 different active feature detectors, 32 frequency channels and 15 threshold levels. A TIMIT trainings sentence was used as input.

The front-end configuration consists of 41 output spike channel blocks covering time scales up to 200ms (from bottom to top): onset detector block, 14 long-pass filter blocks (20ms, 30ms, 40ms, 50ms, 60ms, 70ms, 80ms, 90ms, 100ms, 120ms, 140ms, 160ms, 180ms, 200ms), 13 short-pass filter blocks (10ms, 20ms, 30ms, 40ms, 50ms, 60ms, 80ms, 100ms, 120ms, 140ms, 160ms, 180ms, 200ms), offset detector block, 12 band-pass filter blocks (10-50ms, 20-60ms, 30-70ms, 40-80ms, 50-90ms, 60-100ms, 80-120ms, 100-140ms, 120-160ms, 140-180ms, 160-200ms, 180-220ms)

5.2 PHONEME RECOGNITION TASK

To show the potential of margin learning for studying neural processing of realistic feature detection tasks we use the auditory front-end to generate high dimensional spike patterns based on natural speech and train neurons as phoneme detectors. We use the TIMIT dataset that contains sentences of speech from American English speakers along with the provided information about the occurrences of phonemes.

To convert the raw sound input into spike patterns we configure the auditory front-end to contain a wide range of short-, long- and overlapping band-pass output spike channels. The auditory front-end consists of 41 output spike channel blocks covering time scales up to 200ms:

- 1 onset detector block
- 1 offset detector block
- 14 long-pass filter blocks (20ms, 30ms, 40ms, 50ms, 60ms, 70ms, 80ms, 90ms, 100ms, 120ms, 140ms, 160ms, 180ms, 200ms)
- 13 short-pass filter blocks (10ms, 20ms, 30ms, 40ms, 50ms, 60ms, 80ms, 100ms, 120ms, 140ms, 160ms, 180ms, 200ms)
- 12 band-pass filter blocks (10-50ms, 20-60ms, 30-70ms, 40-80ms, 50-90ms, 60-100ms, 80-120ms, 100-140ms, 120-160ms, 140-180ms, 160-200ms, 180-220ms)

We use 32 mel-scale frequency channels with 15 loudness thresholds each resulting in a total of $N_{\text{synapses}} = 41 * 15 * 32 = 19680$ synapses. To better accommodate the average length of phonemes (around 17.5ms for 'B' to 163.0ms 'AW') we increase the neuron models decay time constants for membrane integration τ_m to 40ms and synaptic currents τ_s to 10ms.

Training is done by presenting a TIMIT sentence from a training pool to the auditory front-end and using the generated spike pattern as input for the neuron. For the multi-spike tempotron and margin learning rule only the count of the target phone is provided as the teaching signal. If we train for the phoneme 'T' and the current training sentence contains 5 instances of 'T' we train the neuron with a desired output spike count of 5. No information about the exact timing of phonemes is used during training with the multi-spike tempotron and margin learning rules.

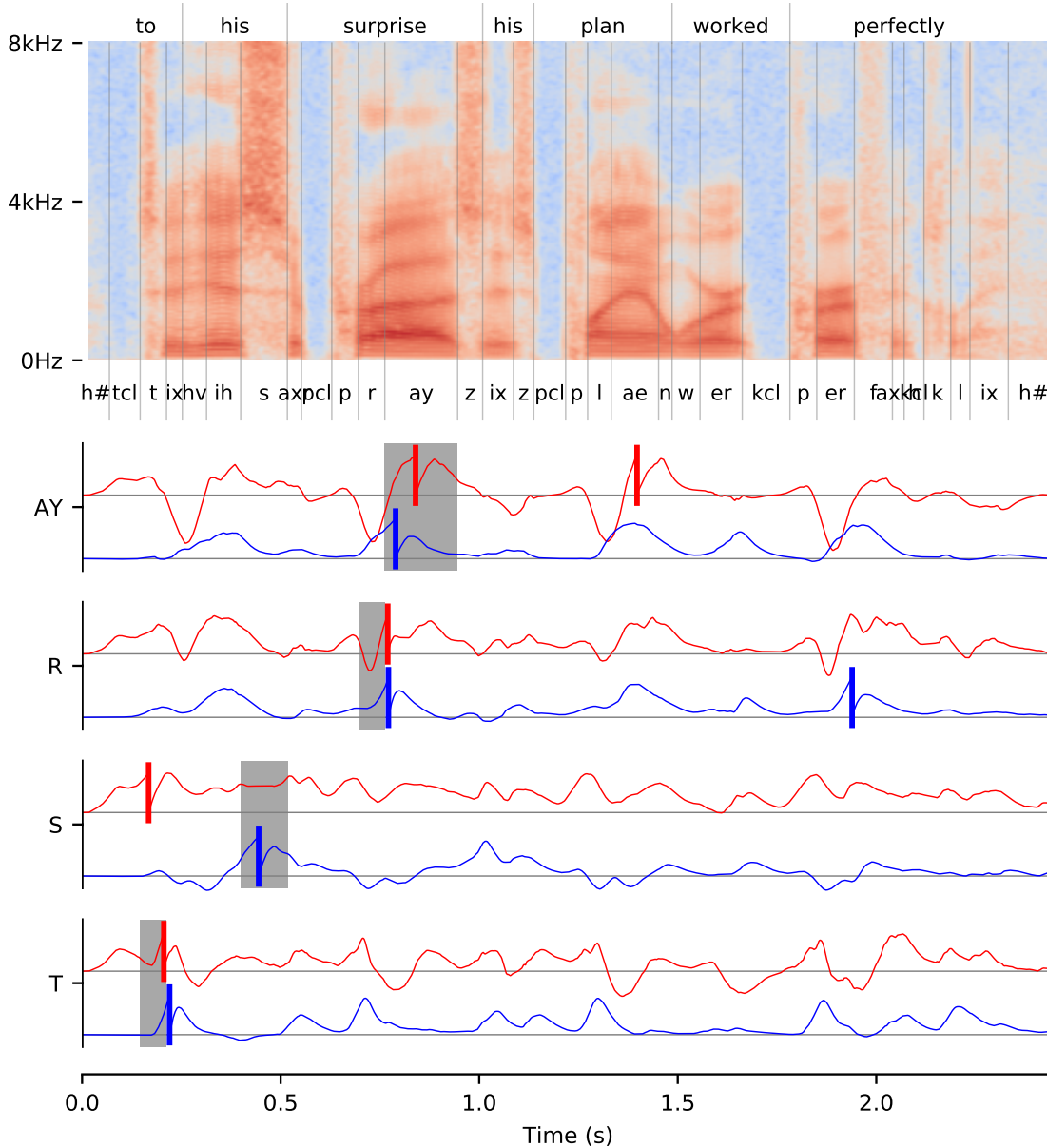


Figure 5.8: Voltage traces of neurons after tempotron (red) and margin learning (blue) on a TIMIT test sentence. Time intervals for the target phones AY, R, S and T are marked with gray areas.

Since traditional speech recognition machines using hidden-markov-models (HMMs) and deep neural networks are not spike based we cannot use them to test the performance achievable with our spike generating auditory front-end. We need a different approach to provide a baseline for comparison. For this we use the segmented learning rule to determine what performance can be reached with the current front-end if a neuron has access to the precise timing information of the target feature during training. We optimized learn

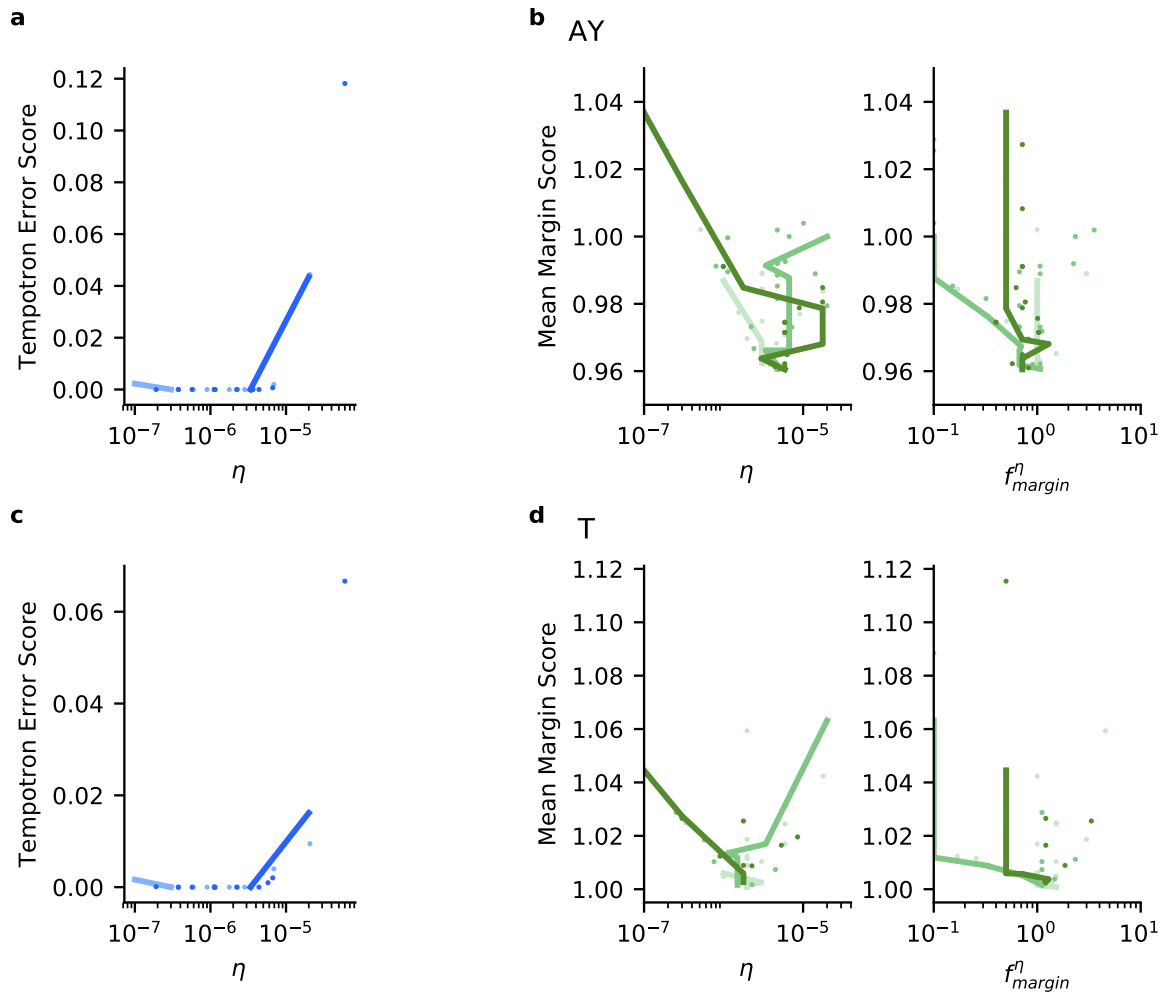


Figure 5.9: **(a+b)** Parameter optimization runs for phoneme AY using the training error for the tempotron learning rule **(a)** and the mean margin for the margin learning rule **(b)**. **(c+d)** Similar parameter optimization runs for phoneme T.

step size η and relative spike position in the target feature α for four different phonemes each from a different group of phonetic sounds: diphthongs 'AY', stops 'T', fricatives 'S' and liquids 'R'. Optimization of segmented learning was done with respect to optimal generalization performance on the TIMIT provided test sentences. This generalization performance is measured as proficiency, the ability to predict the correct sequence of target and non-target features (see section 2.6.1).

For the optimization of parameters for tempotron and margin learning we do not use the generalization performance on test data or the provided segmentation information of the trainings sentences. We proceed as in the section on margin based optimization 4.5 and restrict the optimization to information available during training. We optimize learn step size η for the tempotron based on the training

error (the fraction of training patterns for which it did not elicit the correct spike count). For margin learning with momentum and decay we use an attenuation of $\lambda = 0.995$ and optimized η and f_{margin}^{η} . The latter is a factor determining the strength of the margin learning step relative to η . As a scoring function for margin learning we use the average margin, same as in section 4.5. Due the fact that null-patterns can make up a large portion of the training data set (e.g. 50% for 'AY') and that it is unclear how to incorporate them in a balanced way in the margin learning rule we limited margin learn steps and average margin calculation to patterns with non-zero label.

We optimize the parameters at a small training pool size of $N = 500$ sentences, randomly drawn from the 3696 TIMIT train sentences for each simulation, to test the learning rules ability to generalize from a limited set of training examples. Each parameter pair was run for 100 random seeds over which the optimization score and generalization performance was averaged. Simulations were allowed to run for 500 training cycles.

Figure 5.9 shows the progression of the optimization runs for tempotron and margin learning. Initial parameters for tempotron learning were $\eta = 1e-7$ and $2e-5$. Margin learning started at $\eta = 1e-7$, $1e-6$ and $2e-5$ with corresponding values of $f_{\text{margin}}^{\eta} = 1.0, 0.5$ and 0.1 .

The table lists the resulting parameter sets and averaged across all runs: Training error, hit- and false-positive rates as well as the proficiency on the test dataset for the optimization results with the best performance. The hit-rate is the fraction of targets the neuron elicited spikes for while the false-positive rate is the fraction of non-targets the neuron erroneously spiked for. The last column lists the percentage of achieved proficiency with regards to the result from segmented learning.

To compare the proficiency values with a machine learning system we used the hidden-markov-model (HMM) based HTK speech recognition toolkit (Young, 1992) and calculated proficiency values for all TIMIT phonemes. First we used the standard approach with tri-phoneme context dependent HMMs that allow the system to use multiple HMMs per target phoneme to model context dependent phoneme variations. And secondly a modified version where we disabled tri-phoneme modeling to restrict each phoneme to a single HMM model more similar to our neural phoneme detectors that uses one neuron model per phoneme. But even restricted to single phoneme models the results are not fairly comparable as the hidden-markov-models are designed to output the most likely sequence of phonemes at once and hence each internal phoneme model competes

run	η	f_{margin}^{η}	train-error	hit-rate	fp-rate	proficiency	%
segmented AY			0.072	0.487	0.012	0.267 ± 0.006	100%
tempotron AY	$2.84e-06$		0.000	0.171	0.009	0.065 ± 0.005	24%
margin AY	$6.66e-06$	0.70	0.230	0.214	0.011	0.088 ± 0.009	33%
segmented R			0.355	0.365	0.031	0.139 ± 0.003	100%
tempotron R	$2.20e-06$		0.000	0.180	0.023	0.057 ± 0.006	41%
margin R	$2.24e-06$	0.33	0.490	0.269	0.024	0.096 ± 0.005	69%
segmented S			0.274	0.628	0.013	0.438 ± 0.004	100%
tempotron S	$4.21e-07$		0.000	0.273	0.030	0.098 ± 0.010	22%
margin S	$5.92e-06$	0.99	0.597	0.539	0.017	0.327 ± 0.016	75%
segmented T			0.383	0.382	0.023	0.158 ± 0.004	100%
tempotron T	$2.84e-06$		0.000	0.104	0.020	0.035 ± 0.011	22%
margin T	$1.97e-06$	1.10	0.421	0.282	0.019	0.112 ± 0.008	70%

HTK (single/tri-phoneme): AY 0.522/0.514, R 0.231/0.469, S 0.249/0.671, T 0.008/0.557

against another. Our neural classifiers work completely independent and only output singular time points of likely target phoneme appearance. Additionally during training of the hidden-markov-model the system directly uses the sequence of phonemes in the training sentences while the tempotron and margin learning algorithms only have access to the aggregate label. The proficiency results for the four target phonemes are listed below the table (also see figure 6.1 in the discussion for a comparison between tri-phoneme and single-phoneme performance).

For all four phonemes the segmented learning algorithm has a non zero training error. It is unable to learn to correctly fire for all targets in the training sentences. Since the training error already stopped improving before the simulations abortion criterion was reached the most likely explanation is that the auditory front-end is the limiting factor. This increases the difficulty of the task for tempotron and margin learning since there does not seem to be a solution to find that correctly elicits one spike for each target phoneme. The learning rules will have to cope with classification errors on the training data set.

Still tempotron learning achieves a zero training error for all four phonemes. Its low generalization performance means that the model overfitted the trainings data to decrease the training error to zero and did not learn to reliably detect the target feature.

The margin learning algorithm is only able to improve the generalization performance for AY over tempotron learning by a small amount but manages to increase proficiency for R, S and T more

significantly, tripling the proficiency for S and T, and recovering over 69% of the segmented learning performance for these three phonemes.

5.3 GENERALIZATION PERFORMANCE AND FRONT-END DIMENSION

To characterize how well the tempotron and margin learning rule are able to generalize under different conditions we measured their performance under changing trainings data availability and input space dimensionality.

We trained with 60, 125, 250, 500, 1000, 2000 and then the full 3700 training sentences and measured the average generalization performance for 100 initial random seeds. This random seed was used to randomly draw the subset of training patterns, or in the case of the full dataset to shuffle their order.

For 60 training patterns tempotron and margin learning result in similarly low proficiency, both unable to generalize from the limited amount of information available to them. To achieve about 50% of the segmented learning performance for T the margin learning rule requires only 500 training patterns while the tempotron rule requires the full dataset. For the phoneme S margin learning already reaches close to its peak performance at 500 patterns and even with the full trainings dataset tempotron learning is unable to reach the same performance. With the availability of all trainings sentences margin learning for the phoneme T is able to exceed the generalization performance of segmented learning. This is possible since the segmented learning rule does not incorporate margin learning. The difference between proficiency on training (0.30) to test (0.19) is likely to improve when margin learning is also applied to segmented learning. The proficiency drop for margin learning is small (0.25 on training to 0.24 on test).

To measure the effect of input space dimensionality we kept the size of the trainings pool fixed to 500 and changed the front-end configuration. The front-end configuration directly corresponds to the amount of input synapses of the neuron. We measured for 41, 21, 11 and 2 output spike generating blocks in the auditory front-end. With 32 channels and 15 threshold levels per block this results in 19680, 10080, 5280 and 960 input synapses. We used the following front-end configurations: 21 (2x all on-/offsets, 7x long-, 6x short- and 6x band-pass), 11 (2x all on-/offset, 3x long-, 3x short- and 3x band-pass), and 2 (2x all on-/offset). The individual blocks chosen as all being a strict

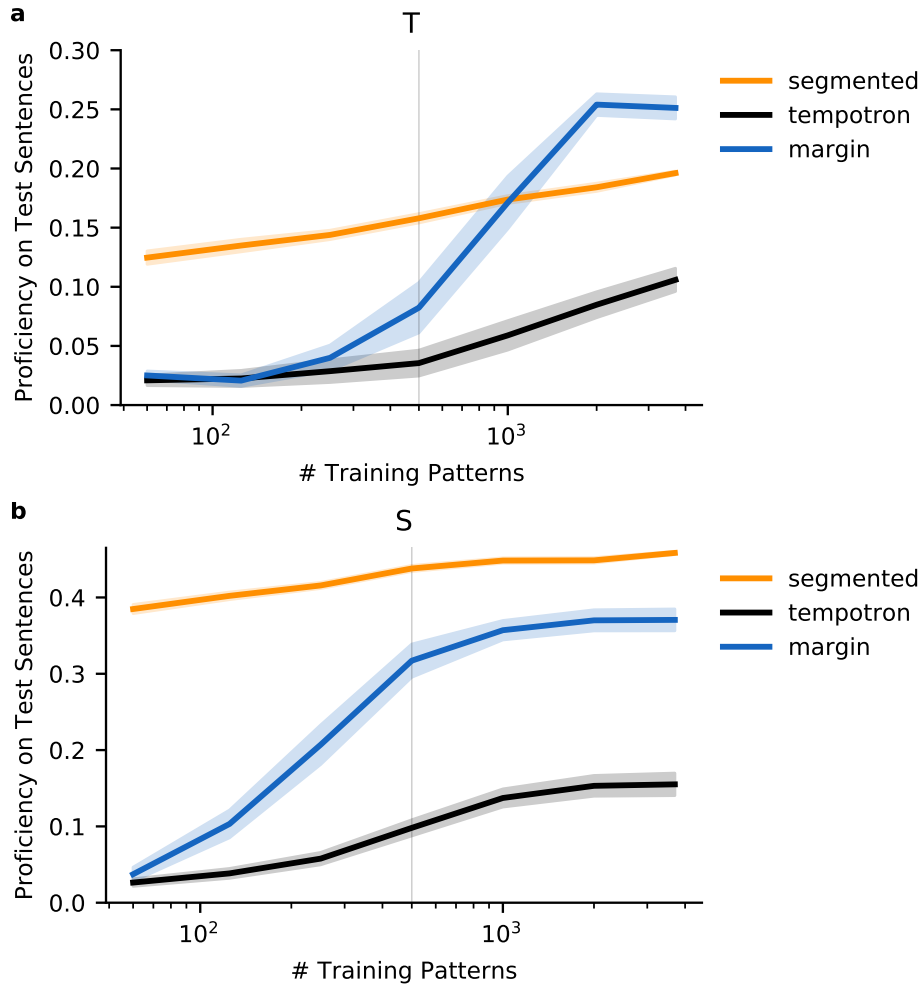


Figure 5.10: (a+b) Phoneme recognition performance under increasing availability of training data for the phonemes T and S. Vertical thin line represents 500 training patterns for which the parameters of the learning rules were optimized.

subset of the front-end with the next higher block count. Due to the change in synapse count we optimized the learning parameters for each front-end size individually.

Already with 5280 synapses the tempotron reaches a zero training error and the generalization performance starts to degrade (figure 5.11). The multi-spike tempotron learning rule is unable to use the additional synapses to improve generalization. Instead it uses the higher dimensionality to overfit the training data resulting in solutions that offer worse phoneme detection performance.

The margin learning rule is able to use the added information about temporal structures provided by the additional auditory front-end blocks to increase its phoneme detection performance.

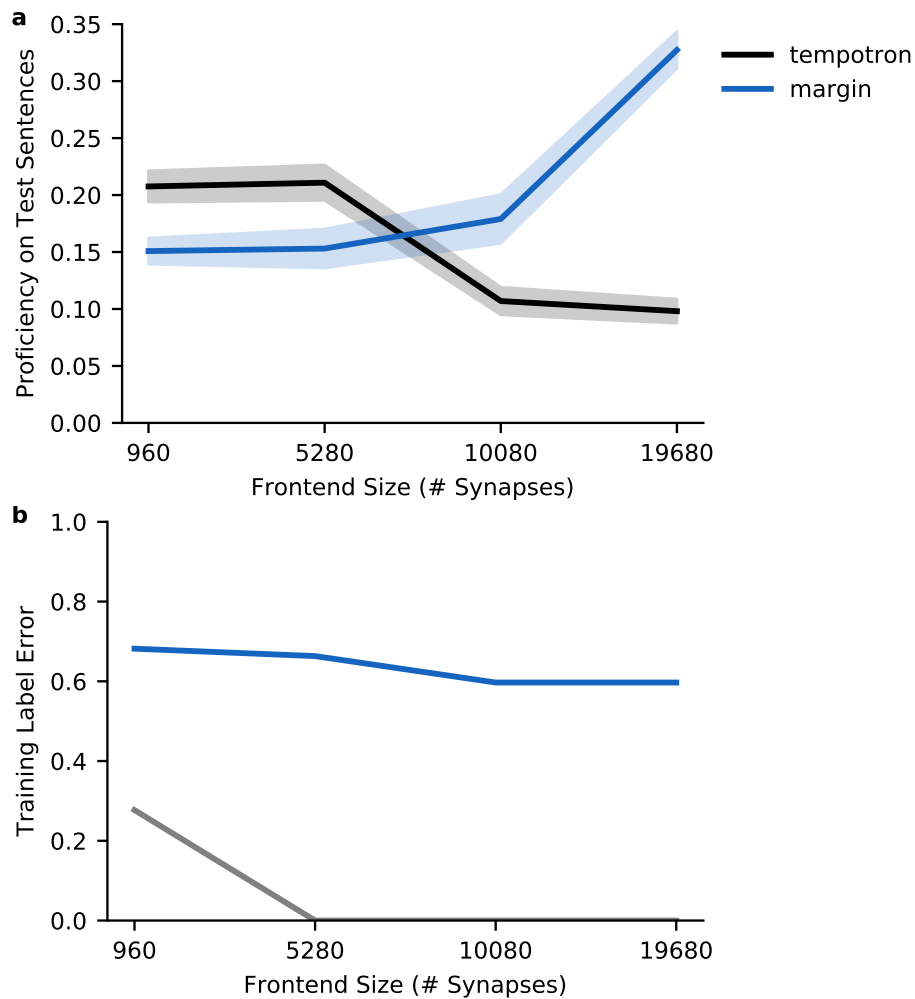


Figure 5.11: (a) Tempotron and margin learning performance for the phoneme 'S' for different auditory front-end sizes. (b) Corresponding fraction of misclassified training patterns.

The results for both experiments, reducing the available trainings data and increasing input space dimensionality, demonstrate that the tempotron learning rule with margin allows studying of complex feature detection tasks and is able to operate in high dimensional input spaces by preventing overfitting.

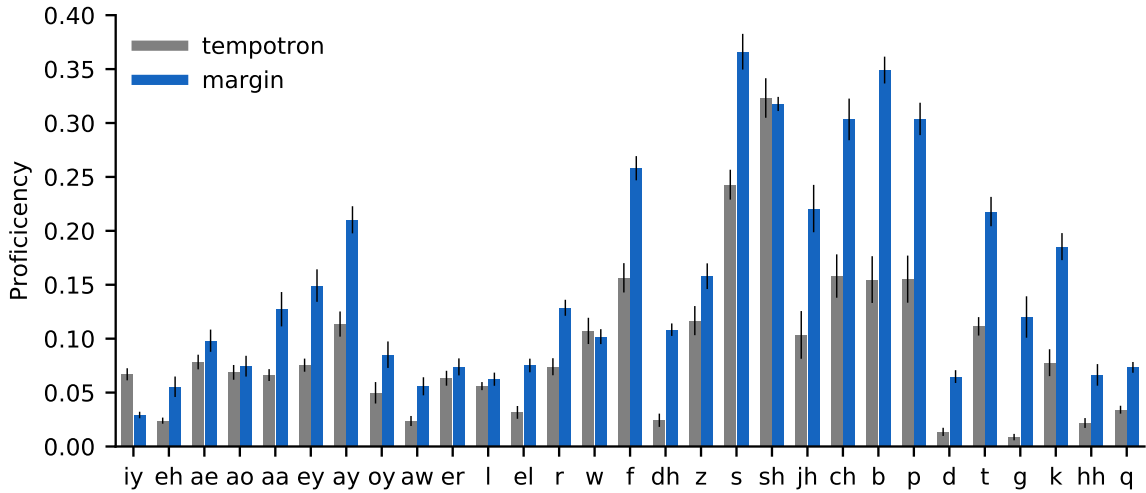


Figure 5.12: Proficiency on the TIMIT test sentences for tempotron and margin learning. Shown are all 29 phonemes for which one of the two learning rules exceeded a proficiency of 0.05. The average relative improvement in proficiency is 135%.

5.4 GENERAL PERFORMANCE IMPROVEMENT

Since the four phonemes AY, R, S, T may not be representative for the usefulness of margin learning and its performance improvement across all phonemes in the TIMIT data we trained phoneme detectors for all of them. We used all available training sentences, set η to $\eta = 3e-6$, a value near the optimized values for both tempotron and margin learning and $f_{\text{margin}}^{\eta} = 1.0$.

The average relative proficiency improvement of margin learning over tempotron learning is 102%. Since many of the phonemes have very low average proficiency for both margin and tempotron learning we also calculated the average improvement limited to phonemes for which one of the two learning rules exceeded a proficiency of 0.05. For these 29 phonemes the average proficiency improvement over tempotron learning is 135%. The hit-rate is increased by 84% while the false-positive rate also increases by an average of 10%. Figure 5.12 shows the proficiency of tempotron and margin learning for these 29 phonemes. The results for all phonemes can be found in the appendix.

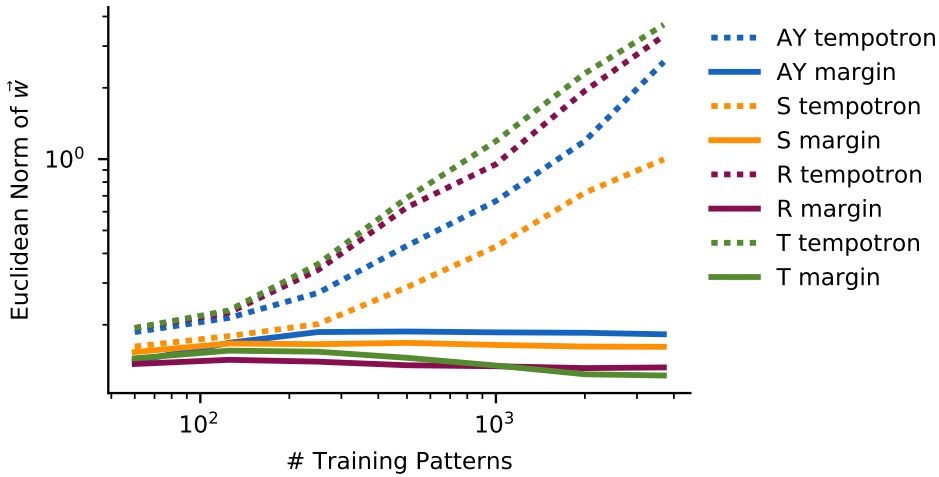


Figure 5.13: Euclidean norm of synaptic efficacies after learning. Compare with figure 4.7 which shows the Euclidean norm for the embedded feature task.

5.5 WEIGHT VECTOR REGULARIZATION

As demonstrated with the synthetic embedded feature task the addition of weight decay to reduce the weight vector norm is crucial. It suppresses irrelevant components of the weight vector and improves generalization performance. The components that are left with a significant strength are important for performing the feature detection task. This allows for interpretation of the functional role of these synapses.

Since we constructed the auditory front-end and know the exact functionality of every output channel we can look at the synaptic efficacies and get an understanding of which parts of the auditory front-end are used. Figure 5.15 shows the synaptic weights split up by front-end block, frequency channel and threshold level for tempotron and margin learning for the phoneme S. Figure 5.14 shows an example spectrogram for a TIMIT sentence as well as output spike triggered spectrograms for these weights. We used the weights of the best performing, with regard to proficiency on test data, random seed for each learning rule.

This visual representation of the synaptic efficacies for each block of output channels can be seen as a spectro-soundlevel receptive field of the neuron, similar to spectro-temporal receptive fields (STRF) but with the time domain being split up across the front-end block types.

Visually comparing the weights of tempotron and margin learning shows some apparent differences. The weights from tempotron learning look noisier, unstructured, strong efficacies can be found across

all types of front-end blocks. The weights from margin learning look more smooth and more sensitive to broad features in the frequency, threshold and time domain. It can be clearly seen how onsets in the high frequency longpass filters contribute excitatory while intermediate frequencies contribute inhibitory. This is augmented with excitatory input from the offset detectors for low frequencies. We can compare this with how the signal of an S phoneme is typically represented in the spectrogram of a sentence and with the output spike triggered spectrogram. The spike triggered spectrograms show that the neuron trained with the margin learning rule learned to elicit output spikes shortly after onset of signals in high frequency channels ($>4\text{kHz}$) and after offset of signals in the lower frequencies ($<2\text{kHz}$). The spike triggered spectrogram of the tempotron learning neuron is less clear about its preference. It seems to react to the end of the phoneme S where the high frequency noise turns off.

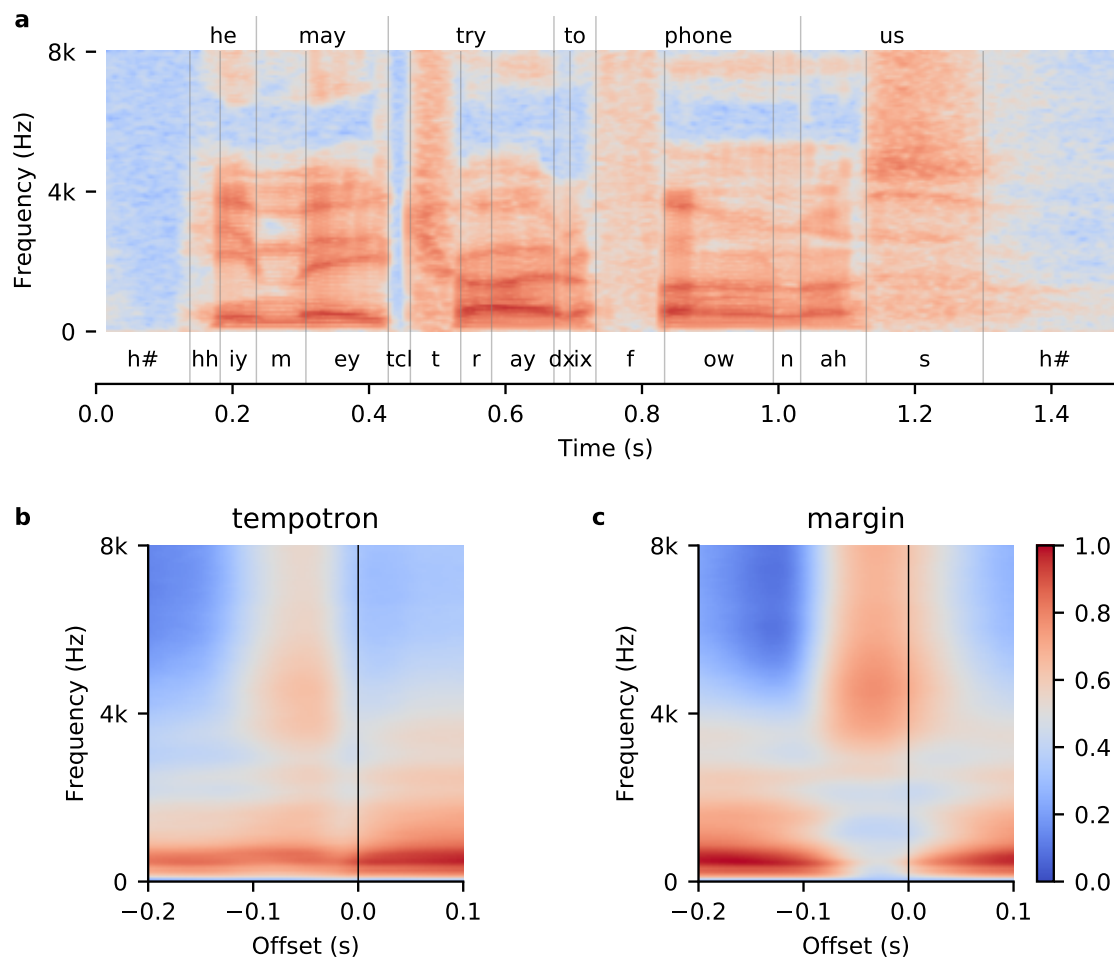


Figure 5.14: (a) The phoneme S has a characteristic high-pitched hissing sound. In the spectrogram we can see that the S consists of a high frequency noise portion that lasts for its duration while signals in lower frequencies that are part of phonemes before and after it are quiet. Signal intensity increases from blue to yellow to red. (b+c) Output spike triggered spectrograms of phoneme S for tempotron and margin learning. They are generated by calculating the average signal strength in the spectrogram around output spikes across all available TIMIT sentences.

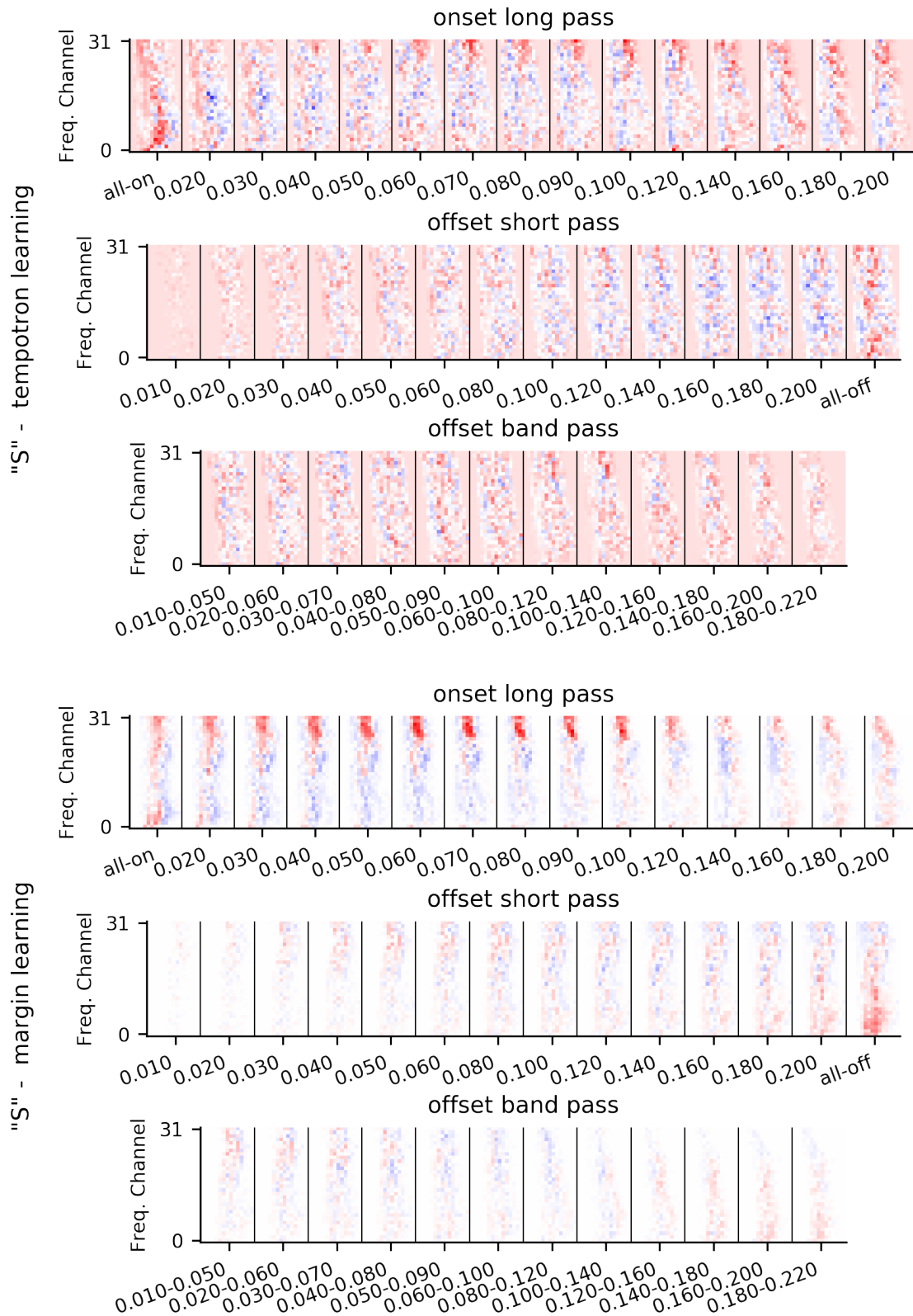


Figure 5.15: Synaptic efficacies for tempotron and margin learning split up by auditory front-end blocks that correspond to different temporal filters. The target phoneme is S. White equals zero, red is excitatory and blue inhibitory strength of the efficacy. The x-axis separates different spike generator blocks and inside of each the 15 threshold levels (increasing from the left).

DISCUSSION

6.1 MARGIN LEARNING FOR SPIKING NEURON MODELS

In this thesis we introduced the tempotron with margin, a learning rule for spiking neural classifiers that enables learning in high dimensional input spaces and offers robust generalization performance from limited training data. The algorithm is based on the multi-spike tempotron introduced by Gütig, 2016. He introduced a novel aggregate-label learning rule that solves the temporal credit-assignment problem using a continuous objective function called the spike-threshold-surface. It maps virtual threshold values to output spike counts given an input pattern and current synaptic efficacies. The margin learning rule extends this algorithm by continuing learning after a pattern is already correctly classified. Inspired by concepts from machine learning we introduce a margin definition based on the spike-threshold-surface and use gradient learning steps to widen these margins. This learning rule is complemented by adding weight decay that suppresses irrelevant components of the weight vector and improves generalization performance.

Using a synthetic task of features embedded in a sensory input stream we show that increasing the margins strengthens the robustness of the classifier against noise. Weight decay on its own leads to an improvement of generalization performance. We demonstrate that by combining both aspects, margin learning and weight decay, the resulting classifiers offer the best performance with regards to both generalization as well as robustness even when the amount of training data is limited.

By comparing the margin learning algorithm to a simple learning rule that uses noise on the firing threshold as a means to increase robustness we could also show that our gradient based approach is not only orders of magnitudes faster but also more robust. It has the advantage that margin parameter tuning is less critical. Stochastic margin learning requires tuning of the margin parameter as a trade off between margin width and convergence time.

We show that using the margin instead of the amount of correctly classified training patterns as a measure for learn progress allows for

parameter optimization and improved generalization performance even without access to test data.

Testing the margin learning rule with a phoneme detection task required the development of two additional tools. First we developed an auditory brain-stem model by extending an existing one with additional output channels sensitive to certain temporal structures in the sound input. This auditory front-end imitates responses to tones with different frequencies, loudness and temporal structures measured in neurons of the inferior colliculus. Its spike pattern output can then be used as input for neural classifiers.

To get a bound for the achievable performance of a neural classifier with this auditory front-end we introduced a learning rule, segmented learning, that uses knowledge about time intervals in the input pattern at which the target feature is present. This learning rule iteratively updates the synaptic efficacies using gradient learning steps until the neuron correctly elicits one spike inside of every target feature occurrence.

We applied segmented learning, multi-spike tempotron learning and margin learning to a phoneme detection task. Using sentences from the TIMIT speech corpus as input for the auditory front-end and the resulting spike patterns as training and test data for the neural classifiers. Only segmented learning had access to the times of target feature appearance while tempotron and margin learning only used the count of target features in the input sentence as their teaching signal. We show that margin learning improves the phoneme detection performance on average by a factor of two over multi-spike tempotron learning. With increasing size of the auditory front-end the performance of multi-spike tempotron learning decreases. It is unable to use the additional information available and overfits the training data. In contrast margin learning is able to improve phoneme detection performance when the size of the auditory front-end is increased.

Due to the margin learning rules property to suppress irrelevant synaptic efficacies and the reinforcement of important ones it allows for qualitative interpretation of the neural classifiers efficacies. This could prove helpful with understanding how neurons solve certain feature detection tasks and which input synapses play a crucial role for robust detection and generalization.

With these successful applications of the introduced margin learning rules to both a synthetic and a human speech recognition task we demonstrated their potential for studying neural processing of high-dimensional inputs with spiking neurons.

6.1.1 *Biological Plausibility*

While it is unlikely that a neuron has knowledge about the exact shape of its spike-threshold-surface it is possible that it has means to track, through some sort of eligibility trace, how close it came to eliciting an additional spike. Based on this information it could decrease the synaptic strength of the synapses leading to a local voltage maximum and increase the neurons robustness.

But even without knowledge about its current margins there is a process that could implement margin learning in neurons. Spike timing dependent plasticity (STDP) (Markram et al., 1997) learning could be a process to produce a margin learning like effect. By reinforcing synapses that contributed with input spikes directly before output spike generation and weakening synapses whose input spikes arrived immediately after an output spike it strengthens the correlation between output spikes and synapses important to elicit them. Similar to how our margin learning rule continues to change efficacies corresponding to the nearest critical threshold values after it already learned to elicit the correct output spike count. In this framework spike timing dependent plasticity could augment a neurons learning process by reinforcing already learned correlations.

The weakening of non-contributing input synapses in spike timing dependent plasticity can also be seen as a weight decay that slowly pushes their influence to zero. But research shows that spike timing dependent plasticity typically works in time frames of ± 40 ms around the output spikes (Bi and Poo, 1998). Synapses contributing to sub-threshold maxima outside this time window would be unaffected by it.

While long-term potentiation (LTP) of synaptic strength can last a long time (Bear, Connors, and Paradiso, 2007) its effect slowly decays. In combination with the spike timing dependent plasticity reinforcement of important efficacies and weakening of unimportant ones, such a slow general decay across all synapses should suffice to reduce the strength of irrelevant synaptic efficacies and improve the neurons generalization performance. Additionally, recent research suggests that LTP decay is regulated by active processes (Villarreal et al., 2002) so one could argue the possibility that the decay could be actively controlled by the neuron to improve its learning capabilities and robustness.

6.2 LIMITATIONS AND OUTLOOK

The segmented learning rule can be improved by combining it with margin learning. When segmented learning stops due to all output spikes being inside the given target features margin learning can be used to continue learning and increase the robustness of the found solution. This should increase the generalization performance in the phoneme recognition task even further and decrease the drop in proficiency between the training and test dataset (current proficiency for AY on training sentences 0.79, on test 0.27).

A current limitation of the segmented learning rule is that its LTP step requires a relative time point α inside the target feature interval that needs to be optimized for every task. The aggregate label learning rule of the multi-spike tempotron solves the problem of learning without knowledge about the optimal time point. Equivalently, using the spike-threshold-surface and selecting a ϑ_k^* for which its corresponding time point t_k^* lies inside the target interval would remove this additional parameter. Similarly the LTD step can be based on the spike-threshold-surface too and be used to directly target the ϑ_k^* responsible for an erroneous output spike. This implements a multi-spike tempotron learning rule limited to only operate inside a given time interval that is embedded inside a larger input spike pattern. This learning rule allows for defining overlapping target feature intervals and the possibility to study learning with no, partial or full information about target feature timing. Full information: each target interval corresponds to one target feature, partial information: target intervals are larger than the actual features and can overlap, no information: all target intervals are as long as the input spike pattern, the learning rule is then equivalent to multi-spike tempotron learning.

The auditory front-end used in the phoneme recognition task only includes output channels that elicit a single spike for every appearance of a specific temporal structure in the sound input. While some neurons react to stimulus offset or onset, others are observed to elicit a sustained stream of output spikes during stimulus presence (Faure et al., 2003; Kasai, Ono, and Ohmori, 2012). The current front-end is only sensitive to changes in the spectrogram and sustained firing channels would augment the output with information about the current signal strength across all frequencies and loudness levels. The inclusion of sustained firing output channels in the auditory front-end should improve detection performance for phonemes that include periods of constant signal strength in some frequency channels. The average output spike rates of these sustained channels can be up to an order of magnitude higher than for the on- and offset detectors, up to 50Hz (Brand, Urban, and Grothe, 2000; Faure et al., 2003; Kasai,

Ono, and Ohmori, 2012) instead of around 0.5-2Hz. While preliminary tests that incorporated sustained channels in the front-end were promising the stark differences in the input statistics of the synapses caused problems with tempotron learning as well as margin learning. The tempotron learning rules were historically designed for classification tasks of sparse spike patterns (Gütig, 2016; Gütig and Haim Sompolinsky, 2006). In the neuron model spikes that arrive at the same synapse shortly after another contribute to the membrane potential exactly the same as if they were arriving at different synapses. This linearity means that the effect of a synaptic efficacy change directly scales with the amount of input spikes arriving at a synapse. Different normalization schemes, ranging from per synapse learning step sizes to synaptic short term plasticity, could be used to counteract the effects of different input statistics.

In addition to the sustained firing output channels other improvements to the auditory front-end could be explored in future research. Measurements in the auditory cortex of awake marmosets show level invariant representations of sounds (Sadagopan and Wang, 2008). Including level as well as pitch invariant representations in the auditory front-end could result in advantages for speech recognition since vocal tracts of different sizes, mainly depending on age and sex, have different resonant frequencies that form the basis of many speech sounds (Hillenbrand et al., 1995).

Directly training phoneme detectors with the output of the auditory front-end might also not be a good model for studying human speech processing. Direct cortical surface recordings in humans revealed phonetic representations in the superior temporal gyrus (Mesgarani et al., 2014). Those recordings do not show local selectivity to single phonemes but instead the findings suggest a multi-dimensional feature space for encoding acoustic parameters of speech sounds. An intermediate layer that specializes on the detection of these acoustic speech sounds could be added to the model. This intermediate layer could be trained unsupervised, as in Gütig, 2016, to serve as a feature map for the phoneme detection in the following layer.

The introduction of tri-phone context modeling in hidden markov model based speech recognition machines lead to notable performance improvements (Lopes and Perdigao, 2011; Young, 1992, also see figure 6.1). Switching from single phonemes to training for transitions between phonemes might also lead to performance improvements in our model. Additionally, modern speech recognition systems incorporate grammar models and word context information (Lei et al., 2013) in their process, both aspects that our simple neural model lacks and would need to be implemented in a multi-layer

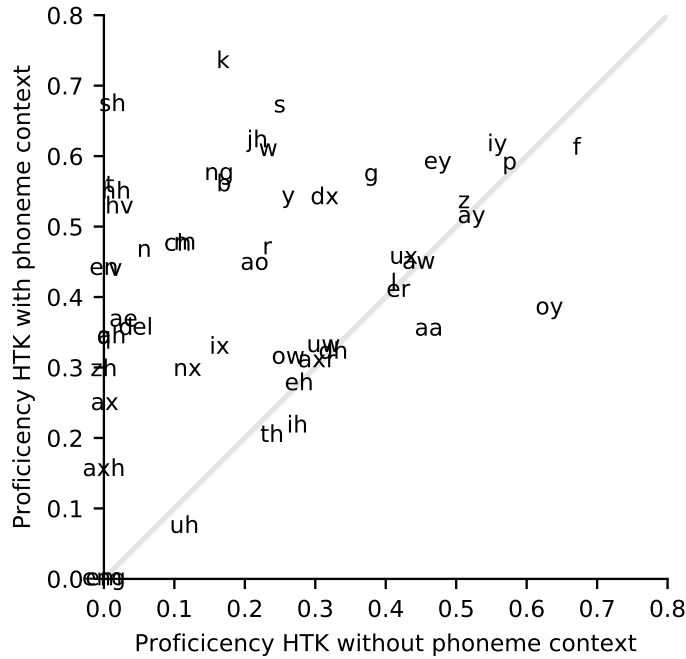


Figure 6.1: Proficiency results generated using the HTK speech recognition toolkit. We trained the system with and without modeling of tri-phoneme contexts in the underlying hidden markov models. The introduction of these context dependent HMMs lead to notable improvements of speech recognition systems.

neural network to construct a system that goes further than single phoneme detection.

Implementing spike timing dependent plasticity (STDP) as a means to increase the margin and comparing it with the margin learning rule could offer insight in the biological plausibility of the concept.

The margin learning rule uses only the nearest critical threshold values ϑ_k^* and ϑ_{k+1}^* in its gradient learn steps. The spike-threshold-surface allows for many more approaches that could increase robustness. One of them is the idea to not only widen the plateau of the current output spike count but also of its multiples i.e. ϑ_{2k}^* and ϑ_{2k+1}^* . The intuition being that those higher order plateaus in the spike-threshold-surface should, if the plateau around the current spike count is already wide, correspond to multiple spikes being elicited for the current target feature. Initial test with these learning rules motivate further research.

Margin learning has also prospect for unsupervised learning. Its capability to pick up on a feature even when only limited amounts of training examples are available and its property to increase the detection robustness for the found feature could be a building block for self-supervised networks. Making it a promising tool to study the development of feature maps for sensory input streams.

There are some remaining issues with the margin learning rule that require further research. How to properly include null-patterns in the margin definition and learning rule is an open question. Also directly using the margin distance to steer optimization does not lead towards the optimal parameter sets with regard to generalization performance. Penalizing correct and incorrectly classified training patterns differently might improve the reached performance. This could be important in situations where no solution exists that correctly classifies all training patterns. For example in the case of wrongly labeled input data or an auditory front-end that does not provide enough information to distinguish all phonemes from each other.

APPENDIX

This is a list of the ARPABET used in the DARPA TIMIT Acoustic-Phonetic Continuous Speech Corpus (TIMIT). The examples are from the documentation of dataset (Garofolo et al., 1993).

Stops		
b	bee	BCL B iy
d	day	DCL D ey
g	gay	GCL G ey
p	pea	PCL P iy
t	tea	TCL T iy
k	key	KCL K iy
dx	muddy, dirty	m ah DX iy, dcl d er DX iy
q	bat	bcl b ae Q

Affricates		
jh	joke	DCL JH ow kcl k
ch	choke	TCL CH ow kcl k

Fricatives		
s	sea	S iy
sh	she	SH iy
z	zone	Z ow n
zh	azure	ae ZH er
f	fin	F ih n
th	thin	TH ih n
v	van	V ae n
dh	then	DH e n

Nasals		
m	mom	M aa M
n	noon	N uw N
ng	sing	s ih NG
em	bottom	b aa tcl t EM
en	button	b ah q EN
eng	washington	w aa sh ENG tcl t ax n
nx	winner	w ih NX axr

Semivowels and Glides		
l	lay	L ey
r	ray	R ey
w	way	W ey
y	yacht	Y aa tcl t
hh	hay	HH ey
hv	ahead	ax HV eh dcl d
el	bottle	bcl b aa tcl t EL

Vowels		
iy	beet	bcl b IY tcl t
ih	bit	bcl b IH tcl t
eh	bet	bcl b EH tcl t
ey	bait	bcl b EY tcl t
ae	bat	bcl b AE tcl t
aa	bott	bcl b AA tcl t
aw	bout	bcl b AW tcl t
ay	bite	bcl b AY tcl t
ah	but	bcl b AH tcl t
ao	bought	bcl b AO tcl t
oy	boy	bcl b OY
ow	boat	bcl b OW tcl t
uh	book	bcl b UH kcl k
uw	boot	bcl b UW tcl t
ux	toot	tcl t UX tcl t
er	bird	bcl b ER dcl d
ax	about	AX bcl b aw tcl t
ix	debit	dcl d eh bcl b IX tcl t
axr	butter	bcl b ah dx AXR
ax-h	suspect	s AX-H s pcl p eh kcl k tcl t

phoneme	hit-/fp-rate tempotron	hit-/fp-rate margin	proficiency tempotron	proficiency margin	%
IY	0.177/0.022	0.139/0.029	0.067 ± 0.005	0.029 ± 0.002	43%
IH	0.116/0.024	0.138/0.022	0.026 ± 0.002	0.034 ± 0.003	132%
IX	0.162/0.040	0.127/0.041	0.042 ± 0.002	0.020 ± 0.003	46%
EH	0.108/0.021	0.175/0.020	0.024 ± 0.002	0.055 ± 0.009	231%
AE	0.170/0.012	0.236/0.013	0.078 ± 0.006	0.098 ± 0.010	125%
AX	0.107/0.024	0.106/0.026	0.024 ± 0.002	0.019 ± 0.003	78%
AH	0.095/0.016	0.059/0.016	0.025 ± 0.003	0.011 ± 0.004	45%
AXH	0.027/0.002	0.053/0.003	0.011 ± 0.003	0.021 ± 0.006	200%
UW	0.042/0.003	0.042/0.004	0.015 ± 0.003	0.012 ± 0.003	81%
UX	0.090/0.008	0.046/0.014	0.028 ± 0.005	0.007 ± 0.003	26%
UH	0.012/0.003	0.042/0.004	0.003 ± 0.001	0.012 ± 0.002	356%
AO	0.161/0.010	0.193/0.011	0.069 ± 0.006	0.074 ± 0.009	108%
AA	0.152/0.010	0.258/0.010	0.066 ± 0.005	0.127 ± 0.015	192%
EY	0.181/0.012	0.306/0.013	0.075 ± 0.005	0.149 ± 0.014	197%
AY	0.224/0.009	0.410/0.012	0.114 ± 0.011	0.210 ± 0.012	185%
OY	0.093/0.001	0.154/0.001	0.050 ± 0.009	0.085 ± 0.012	170%
AW	0.065/0.004	0.145/0.005	0.024 ± 0.004	0.056 ± 0.008	236%
OW	0.097/0.011	0.110/0.011	0.029 ± 0.004	0.033 ± 0.005	113%
ER	0.159/0.010	0.191/0.011	0.063 ± 0.006	0.074 ± 0.007	116%
AXR	0.092/0.016	0.060/0.016	0.022 ± 0.002	0.012 ± 0.003	53%
L	0.159/0.022	0.204/0.024	0.056 ± 0.003	0.062 ± 0.005	111%
EL	0.084/0.005	0.162/0.005	0.032 ± 0.005	0.075 ± 0.006	237%
R	0.185/0.025	0.301/0.023	0.074 ± 0.007	0.129 ± 0.007	173%
W	0.221/0.011	0.218/0.009	0.107 ± 0.011	0.102 ± 0.006	95%
Y	0.032/0.006	0.118/0.008	0.009 ± 0.002	0.040 ± 0.006	448%
M	0.136/0.017	0.107/0.022	0.045 ± 0.003	0.022 ± 0.003	49%
EM	0.000/0.001	0.000/0.001	0.000 ± 0.000	0.000 ± 0.000	200%
N	0.163/0.037	0.120/0.035	0.043 ± 0.002	0.019 ± 0.003	43%
EN	0.017/0.003	0.029/0.006	0.005 ± 0.002	0.010 ± 0.004	198%
NX	0.041/0.004	0.068/0.004	0.012 ± 0.002	0.023 ± 0.004	193%
NG	0.007/0.008	0.007/0.009	0.001 ± 0.000	0.001 ± 0.000	82%
ENG	0.000/0.000	0.000/0.000	0.000 ± 0.000	0.000 ± 0.000	237%
V	0.073/0.013	0.038/0.014	0.016 ± 0.002	0.004 ± 0.001	27%
F	0.269/0.009	0.458/0.011	0.156 ± 0.013	0.258 ± 0.011	164%
DH	0.097/0.018	0.260/0.013	0.024 ± 0.005	0.108 ± 0.005	445%
TH	0.034/0.004	0.101/0.006	0.012 ± 0.004	0.036 ± 0.007	292%
Z	0.235/0.017	0.344/0.017	0.117 ± 0.013	0.158 ± 0.011	135%
S	0.364/0.017	0.577/0.014	0.243 ± 0.013	0.366 ± 0.016	150%
ZH	0.011/0.001	0.074/0.001	0.006 ± 0.004	0.035 ± 0.009	541%
SH	0.437/0.003	0.462/0.003	0.323 ± 0.018	0.318 ± 0.006	98%
JH	0.182/0.005	0.379/0.005	0.103 ± 0.021	0.221 ± 0.021	213%
CH	0.249/0.003	0.460/0.004	0.158 ± 0.019	0.303 ± 0.019	191%
B	0.268/0.011	0.542/0.008	0.155 ± 0.021	0.349 ± 0.012	225%
P	0.284/0.012	0.498/0.009	0.155 ± 0.021	0.304 ± 0.014	195%
D	0.069/0.016	0.212/0.020	0.014 ± 0.003	0.065 ± 0.005	478%
DX	0.072/0.012	0.033/0.012	0.016 ± 0.002	0.009 ± 0.003	53%
T	0.258/0.020	0.417/0.015	0.111 ± 0.008	0.218 ± 0.013	195%
G	0.039/0.008	0.234/0.008	0.009 ± 0.002	0.120 ± 0.019	1376%
K	0.198/0.020	0.390/0.018	0.078 ± 0.012	0.185 ± 0.012	238%
HH	0.066/0.005	0.154/0.006	0.022 ± 0.004	0.066 ± 0.009	304%
HV	0.064/0.004	0.101/0.005	0.023 ± 0.004	0.038 ± 0.006	167%
Q	0.126/0.018	0.229/0.021	0.034 ± 0.003	0.073 ± 0.004	215%

BIBLIOGRAPHY

- Bear, Mark F, Barry W Connors, and Michael A Paradiso (2007). *Neuroscience*. Vol. 2. Lippincott Williams & Wilkins.
- Bi, Guo-qiang and Mu-ming Poo (1998). "Synaptic modifications in cultured hippocampal neurons: dependence on spike timing, synaptic strength, and postsynaptic cell type." In: *Journal of neuroscience* 18.24, pp. 10464–10472.
- Block, Hans-Dieter (1962). "The perceptron: A model for brain functioning." In: *Reviews of Modern Physics* 34.1, p. 123.
- Brand, Antje, Reas Urban, and Benedikt Grothe (2000). "Duration tuning in the mouse auditory midbrain." In: *Journal of Neurophysiology* 84.4, pp. 1790–1799.
- Brent, Richard P. (1971). "An algorithm with guaranteed convergence for finding a zero of a function." In: *The Computer Journal* 14.4, pp. 422–425.
- Casseday, JH and Ellen Covey (1992). "Frequency tuning properties of neurons in the inferior colliculus of an FM bat." In: *Journal of Comparative Neurology* 319.1, pp. 34–50.
- Casseday, JH, D Ehrlich, E Covey, et al. (1994). "Neural tuning for sound duration: role of inhibitory mechanisms in the inferior colliculus." In: *Science* 264.5160, pp. 847–849.
- Chen, Guang-Di (1998). "Effects of stimulus duration on responses of neurons in the chinchilla inferior colliculus." In: *Hearing research* 122.1, pp. 142–150.
- Churchland, PS. and TJ. Sejnowski (1988). "Perspectives on cognitive neuroscience." In: *Science*.
- Cortes, Corinna and Vladimir Vapnik (1995). "Support-vector networks." In: *Machine learning* 20.3, pp. 273–297.
- Cristianini, Nello and John Shawe-Taylor (2000). *An introduction to support vector machines and other kernel-based learning methods*. Cambridge university press.
- Dayan, Peter and Laurence F Abbott (2001). *Theoretical neuroscience*. Vol. 806. Cambridge, MA: MIT Press.
- Dekker, TJ (1969). "Finding a zero by means of successive linear interpolation." In: *Constructive aspects of the fundamental theorem of algebra*, pp. 37–51.
- Egorova, Marina, Günter Ehret, Inna Vartanian, and Karl-Heinz Esser (2001). "Frequency response areas of neurons in the mouse inferior colliculus. I. Threshold and tuning characteristics." In: *Experimental brain research* 140.2, pp. 145–161.

- Ehrlich, Daphna, John H Casseday, and Ellen Covey (1997). "Neural tuning to sound duration in the inferior colliculus of the big brown bat, *Eptesicus fuscus*." In: *Journal of Neurophysiology* 77.5, pp. 2360–2372.
- Faure, Paul A, Thane Fremouw, John H Casseday, and Ellen Covey (2003). "Temporal masking reveals properties of sound-evoked inhibition in duration-tuned neurons of the inferior colliculus." In: *Journal of Neuroscience* 23.7, pp. 3052–3065.
- Florian, Răzvan V (2012). "The chronotron: a neuron that learns to fire temporally precise spike patterns." In: *PloS one* 7.8, e40233.
- Fuzessery, ZM and JC Hall (1999). "Sound duration selectivity in the pallid bat inferior colliculus." In: *Hearing research* 137.1, pp. 137–154.
- Fuzessery, Zoltan M (1994). "Response selectivity for multiple dimensions of frequency sweeps in the pallid bat inferior colliculus." In: *Journal of neurophysiology* 72.3, pp. 1061–1079.
- Galazyuk, Alexander V and Albert S Feng (1997). "Encoding of sound duration by neurons in the auditory cortex of the little brown bat, *Myotis lucifugus*." In: *Journal of Comparative Physiology A* 180.4, pp. 301–311.
- Garofolo, J. S., L. F. Lamel, W. M. Fisher, J. G. Fiscus, D. S. Pallett, and N. L. Dahlgren (1993). *DARPA TIMIT Acoustic Phonetic Continuous Speech Corpus CDROM*. URL: <http://www.ldc.upenn.edu/Catalog/LDC93S1.html>.
- Ghosh-Dastidar, Samanwoy and Hojjat Adeli (2009). "A new supervised learning algorithm for multiple spiking neural networks with application in epilepsy and seizure detection." In: *Neural networks* 22.10, pp. 1419–1431.
- Gütig, Robert (2016). "Spiking neurons can discover predictive features by aggregate-label learning." In: *Science* 351.6277. ISSN: 0036-8075. DOI: 10.1126/science.aab4113. eprint: <http://science.sciencemag.org/content/351/6277/aab4113.full.pdf>. URL: <http://science.sciencemag.org/content/351/6277/aab4113>.
- Gütig, Robert, Tim Gollisch, Haim Sompolinsky, and Markus Meister (2013). "Computing complex visual features with retinal spike times." In: *PLoS One* 8.1, e53063.
- Gütig, Robert and Haim Sompolinsky (2006). "The tempotron: a neuron that learns spike timing-based decisions." In: *Nature neuroscience* 9.3, p. 420.
- Gütig, Robert and Haim Sompolinsky (2009). "Time-warp-invariant neuronal processing." In: *PLoS biology* 7.7, e1000141.
- Harrison, Robert V (2001). "Age-related tonotopic map plasticity in the central auditory pathways." In: *Scandinavian Audiology* 30.2, pp. 8–14.
- Hebb, D.O. (1949). *The Organization of Behavior*. Wiley.

- Hertz, John A, Anders S Krogh, and Richard G Palmer (1991). *Introduction to the theory of neural computation*. Vol. 1. Basic Books.
- Hillenbrand, James, Laura A Getty, Michael J Clark, and Kimberlee Wheeler (1995). "Acoustic characteristics of American English vowels." In: *The Journal of the Acoustical society of America* 97.5, pp. 3099–3111.
- Hinton, G. E., J. L. McClelland, and D. E. Rumelhart (1986). "Distributed representations." In: *Parallel distributed processing: explorations in the microstructure of cognition, vol. 1* (MIT Press).
- Hopfield, J. J. (1982). "Neural networks and physical systems with emergent collective computational abilities." In: *Proceedings of the national academy of sciences* 79.8.
- Hopfield, John J and Carlos D Brody (2000). "What is a moment? "Cortical" sensory integration over a brief interval." In: *Proceedings of the National Academy of Sciences* 97.25, pp. 13919–13924.
- Jen, PH-S and RB Feng (1999). "Bicuculline application affects discharge pattern and pulse-duration tuning characteristics of bat inferior collicular neurons." In: *Journal of Comparative Physiology A* 184.2, pp. 185–194.
- Jones, Eric, Travis Oliphant, Pearu Peterson, et al. (2001). *SciPy: Open source scientific tools for Python*. URL: <http://www.scipy.org/>.
- Kasai, Masatoshi, Munenori Ono, and Harunori Ohmori (2012). "Distinct neural firing mechanisms to tonal stimuli offset in the inferior colliculus of mice in vivo." In: *Neuroscience research* 73.3, pp. 224–237.
- Le Mouel, Charlotte, Kenneth D Harris, and Pierre Yger (2014). "Supervised learning with decision margins in pools of spiking neurons." In: *Journal of computational neuroscience* 37.2, pp. 333–344.
- Lei, Xin, Andrew W Senior, Alexander Gruenstein, and Jeffrey Sorensen (2013). "Accurate and compact large vocabulary speech recognition on mobile devices." In: *Interspeech*. Vol. 1.
- Lopes, Carla and Fernando Perdigao (2011). "Phoneme recognition on the TIMIT database." In: *Speech Technologies*. InTech.
- Markram, Henry, Joachim Lübke, Michael Frotscher, and Bert Sakmann (1997). "Regulation of synaptic efficacy by coincidence of postsynaptic APs and EPSPs." In: *Science* 275.5297, pp. 213–215.
- McCulloch, Warren S and Walter Pitts (1943). "A logical calculus of the ideas immanent in nervous activity." In: *The bulletin of mathematical biophysics* 5.4, pp. 115–133.
- Memmesheimer, Raoul-Martin, Ran Rubin, Bence P Ölveczky, and Haim Sompolinsky (2014). "Learning precisely timed spikes." In: *Neuron* 82.4, pp. 925–938.
- Mesgarani, Nima, Connie Cheung, Keith Johnson, and Edward F Chang (2014). "Phonetic feature encoding in human superior temporal gyrus." In: *Science* 343.6174, pp. 1006–1010.

- Novikoff, A. B. J. (1962). "On convergence proofs for perceptrons." In: *Proceedings of the Symposium on the Mathematical Theory of Automata*, pp. 615–620.
- Perez-Gonzalez, David, Manuel S Malmierca, Jodan M Moore, Olga Hernandez, and Ellen Covey (2006). "Duration selective neurons in the inferior colliculus of the rat: topographic distribution and relation of duration sensitivity to other response properties." In: *Journal of neurophysiology* 95.2, pp. 823–836.
- Perrett, DI, Edmond T Rolls, and W Caan (1982). "Visual neurones responsive to faces in the monkey temporal cortex." In: *Experimental brain research* 47.3, pp. 329–342.
- Phillips, Dennis P, SE Hall, and SE Boehnke (2002). "Central auditory onset responses, and temporal asymmetries in auditory perception." In: *Hearing research* 167.1, pp. 192–205.
- Pinheiro, A Daniel, Min Wu, and Philip H-S Jen (1991). "Encoding repetition rate and duration in the inferior colliculus of the big brown bat, *Eptesicus fuscus*." In: *Journal of Comparative Physiology A* 169.1, pp. 69–85.
- Ponulak, Filip and Andrzej Kasiński (2010). "Supervised learning in spiking neural networks with ReSuMe: sequence learning, classification, and spike shifting." In: *Neural Computation* 22.2, pp. 467–510.
- Potter, H David (1965). "Patterns of acoustically evoked discharges of neurons in the mesencephalon of the bullfrog." In: *Journal of Neurophysiology* 28.6, pp. 1155–1184.
- Press, William H (2007). *Numerical recipes 3rd edition: The art of scientific computing*. Cambridge university press.
- Quiroga, R Quian, Leila Reddy, Gabriel Kreiman, Christof Koch, and Itzhak Fried (2005). "Invariant visual representation by single neurons in the human brain." In: *Nature* 435.7045, pp. 1102–1107.
- Rosenblatt, Frank (1958). "The perceptron: A probabilistic model for information storage and organization in the brain." In: *Psychological review* 65.6, p. 386.
- Rubin, R., L. F. Abbott, and H. Sompolinsky (May 2017). "Balanced Excitation and Inhibition are Required for High-Capacity, Noise-Robust Neuronal Selectivity." In: *ArXiv e-prints*. arXiv: 1705.01502 [q-bio.NC].
- Rubin, Ran, Rémi Monasson, and Haim Sompolinsky (2010). "Theory of spike timing-based neural classifiers." In: *Physical review letters* 105.21, p. 218102.
- Ruggero, Mario A (1992). "Physiology and coding of sound in the auditory nerve." In: *The mammalian auditory pathway: Neurophysiology*. Springer, pp. 34–93.
- Rumelhart, David E, Geoffrey E Hinton, Ronald J Williams, et al. (1988). "Learning representations by back-propagating errors." In: *Cognitive modeling* 5.3, p. 1.

- Sadagopan, Srivatsun and Xiaoqin Wang (2008). "Level invariant representation of sounds by populations of neurons in primary auditory cortex." In: *Journal of Neuroscience* 28.13, pp. 3415–3426.
- Schwartz, Eric L, Robert Desimone, Thomas D Albright, and Charles G Gross (1983). "Shape recognition and inferior temporal neurons." In: *Proceedings of the National Academy of Sciences* 80.18, pp. 5776–5778.
- Seung, H Sebastian (2003). "Learning in spiking neural networks by reinforcement of stochastic synaptic transmission." In: *Neuron* 40.6, pp. 1063–1073.
- Shannon, Claude E and Warren Weaver (1998). *The mathematical theory of communication*. University of Illinois press.
- Shore, S. E. (2010). "Auditory/Somatosensory Interactions." In: *Encyclopedia of Neuroscience*. Elsevier Ltd, pp. 691–699. ISBN: 9780080450469. DOI: 10.1016/B978-008045046-9.00268-0.
- Stevens, Stanley Smith, John Volkman, and Edwin B Newman (1937). "A scale for the measurement of the psychological magnitude pitch." In: *The Journal of the Acoustical Society of America* 8.3, pp. 185–190.
- Theil, Henri (1970). "On the estimation of relationships involving qualitative variables." In: *American Journal of Sociology* 76.1, pp. 103–154.
- Vapnik, Vladimir N and Alexey J Chervonenkis (1974). "Theory of pattern recognition." In: *Statistical Learning Problems*.
- Vapnik, Vladimir and Alexey Chervonenkis (1964). "A note on one class of perceptrons." In: *Automation and remote control* 25.1, p. 103.
- Villarreal, Desiree M, Viet Do, Evelyn Haddad, and Brian E Derrick (2002). "NMDA receptor antagonists sustain LTP and spatial memory: active processes mediate LTP decay." In: *Nature neuroscience* 5.1.
- Wright, Stephen J (2015). "Coordinate descent algorithms." In: *Mathematical Programming* 151.1, pp. 3–34.
- Xu, Yan, Xiaoqin Zeng, and Shuiming Zhong (2013). "A new supervised learning algorithm for spiking neurons." In: *Neural computation* 25.6, pp. 1472–1511.
- Young, Steve J (1992). "The general use of tying in phoneme-based HMM speech recognisers." In: *Acoustics, Speech, and Signal Processing, 1992. ICASSP-92., 1992 IEEE International Conference on*. Vol. 1. IEEE, pp. 569–572.

ACKNOWLEDGMENTS

Without the support and help from quite a lot of people this thesis would have not been possible. Here is my attempt at an incomplete list:

First of all I have to thank my supervisor Robert Gütig for taking me into his research group and always pushing me to strive for deeper understanding and more clarity.

Then my thesis committee members Theo Geisel and Fred Wolf. Both of which always offered constructive criticism and discussions. I especially want to thank Theo for supporting me through all these years and encouraging me to continue towards a PhD.

Christian and Denny for keeping the desktop computers and compute clusters alive and running while I tried my best to test their limits. Without their care for the infrastructure I would still be waiting for my simulations to finish.

Julia for making the research group more social and for the many needed breaks, discussions and non-research related activities. The chocolate and ice-coffee consumption severely dropped after you left - which is never a good sign.

Dennis, Manuel, Ali and Liam for countless lunch breaks, conversations, discussions and diverse other nerdy enterprises.

Jan, Sebastian and Peter for always having an open ear for me, giving me advice and offering distractions.

My family. Especially my parents, for always supporting, encouraging and believing in me. I would not be here without all of you.

And finally my girlfriend Nadine. Without you at my side, your mental support, survival packages and the prospect of spending more time with you all of this would have been much harder.

CURRICULUM VITAE

RAFAEL BRUNE

18.09.1983 Born in Göttingen, Germany

EDUCATION

8/2003–7/2008 Studies in Physics
at Georg-August-University Göttingen, Germany

7/2008 Graduated with Diplom in physics
'A Stochastic Model for Panic Behaviour in Disease Dynamics'
from Georg-August-University Göttingen, Germany

7/2008–3/2012 Research scholar in the group on complex systems (RoCS)
at Northwestern University, Evanston, IL, USA

4/2012–11/2013 Scientific programmer
with Dr. Robert Gütig
at MPI for Experimental Medicine

since 11/2013 PhD student
with Dr. Robert Gütig
in the program Theoretical and Computational Neuroscience
at Georg-August-University Göttingen, Germany

PUBLICATIONS

2010 "The structure of borders in a small world."
Thiemann, C., Theis, F., Grady, D., Brune, R. and Brockmann, D.
PloS one, 5(11), p.e15422

2012 "Modularity maximization and tree clustering: Novel ways to
determine effective geographic borders."
Grady, D., Brune, R., Thiemann, C., Theis, F. and Brockmann, D.
In Handbook of optimization in complex networks (pp. 169-208), Springer US