Master Thesis

Optimization of Neural Network

Author:
Martin Bulín MSc.

Supervisor:
Ing. Luboš Šmídl Ph.D.

A thesis submitted in fulfillment of the requirements for the degree of Engineer (Ing.)

in the

Department of Cybernetics

April 28, 2017

Declaration of Authorship

I, Martin Bulín MSc., declare that this thesis titled, "Optimization of Neural Network" and the work presented in it are my own. The main methods follow on my work presented in (Bulín, 2016), however the workload is different.

I confirm that:

- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.

Signed:

Date:

“*Look deep into nature, and then you will understand everything better.*”

A. Einstein

UNIVERSITY OF WEST BOHEMIA

Abstract

Faculty of Applied Sciences
Department of Cybernetics

Engineer (Ing.)

Optimization of Neural Network

by Martin Bulín MSc.

abstract text...

Acknowledgements

acknowledgements text...

Contents

List of Figures

List of Tables

List of Abbreviations

AI	**A**rtificial **I**ntelligence
ANN	**A**rtificial **N**eural **N**etwork
DCT	**D**iscrete **C**osine **T**ransform
DFT	**D**iscrete **F**ourier **T**ransform
HMMs	**H**idden **M**arkov **M**odels
MNIST	**M**odified **N**ational **I**nstitute of **S**tandards and **T**echnology
MFCCs	**M**el **F**requency **C**epstral **C**oefficients
MSE	**M**ean **S**quared **E**rror
NN	**N**eural **N**etwork
PA	**P**runing **A**lgorithm
RPE	**R**ule **P**lus **E**xception
UFI	**U**nbalanced **F**eature **I**nformation
XOR	e**X**clusive **OR**

Chapter 1

Introduction

Introduction text...

1.1 Background

Neuron Principle

FeedForward Network structure

Notation used in this work

Backpropagation learning algorithm

(Rosenblatt, 1958)

1.2 State of the Art

State of the art text... (Reed, 1993)

Karnin method principle

OBD method principle

1.3 Thesis Objectives

Thesis objectives text...

1.4 Thesis Outline

Thesis outline text...

Chapter 2

Methods

Methods intro text...

2.1 Network Pruning

Network pruning text...

Network Shrinking...

2.2 Feature Selection

Minimal network structure text...

Pathing in Net (Feature Selection)

Feature Energy computation

2.3 Network Visualization

Graphical user interface text...

2.4 Speech Data Gathering

The presented methods are tested on several examples (chapter 3) and one of them rests in classification of phonemes. By definition a phoneme is one of the units of sound that distinguish one word from another in a particular language (Wikipedia, 2004). We focus on Czech language and consider 40 phonemes listed in Table 2.1. This section describes the way of gathering such phonemes and building a dataset for classification.

sound	*phoneme*	*example*	*sound*	*phoneme*	*example*	*sound*	*phoneme*	*example*
a	a	mám**a**	ch	x	**ch**yba	ř	R	mo**ř**e
á	A	t**á**ta	i	i	p**i**vo	ř	Q	t**ř**i
au	Y	**au**to	í	I	v**í**no	s	s	o**s**el
b	b	**b**od	j	j	vo**j**ák	š	S	po**š**ta
c	c	o**c**el	k	k	o**k**o	t	t	o**t**ec
č	C	o**č**i	l	l	**l**oď	ť	T	ku**t**il
d	d	**d**ům	m	m	**m**ír	u	u	r**u**m
ď	D	**dě**ti	n	n	**n**os	ú (ů)	U	r**ů**že
e	e	p**e**s	n	N	ba**n**ka	v	v	**v**lak
é	E	l**é**pe	ň	J	la**ň**	z	z	ko**z**a
eu	F	**eu**nuch	o	o	b**o**k	ž	Z	**ž**ena
f	f	**f**acka	ou	y	p**ou**to		_sil_	(silence)
g	g	**g**uma	p	p	**p**rak			
h	h	**h**ad	r	r	**r**ak			

TABLE 2.1: Czech phonetic alphabet.

The generation of a speech dataset consists of the following steps, where the work done in steps $1-3$ is taken over from (Šmídl, 2017).

1. acquisition of real voice recordings;
2. feature extraction from the sound signals (parameterization);
3. labeling the data;
4. definition of one sample;
5. splitting samples into training/development/testing sets.

Acquisition of Voice Recordings

The phoneme dataset is made of real speech recordings from a car interior environment, provided by (Škoda, ?? ref). We are talking about simple voice instructions for a mobile phone or a navigation system, many of them are names of people, streets or towns only. In total 14523 recordings (`.wav` files) of various length (and so number of phonemes) were obtained.

Parameterization

The goal of parameterization is to represent each recording by a vector of features. A commonly known procedure based on MFCCs is used. A nice detailed explanation of this method can be found e.g. in (Lyons, 2009).

The idea behind MFCCs originates in the fact that a shape of human vocal tract (including tongue, teeth etc.) determines what sound comes out. The shape of the vocal tract manifests itself in the envelope of the short time power spectrum, and the job of MFCCs is to accurately represent this envelope.

The parameterization workflow is summarized by these steps:

1. *splitting the signal into short frames*;

   Fig. 2.1 illustrates how a sound signal is splitted into short frames. The parameters are

$$frame\_size = 0.025\,s \ = 25\,ms$$
$$frame\_shift = 0.01\,s \ = 10\,ms$$

   Using the sampling frequency $f_s = 8kHz$, we get frames of length 200.

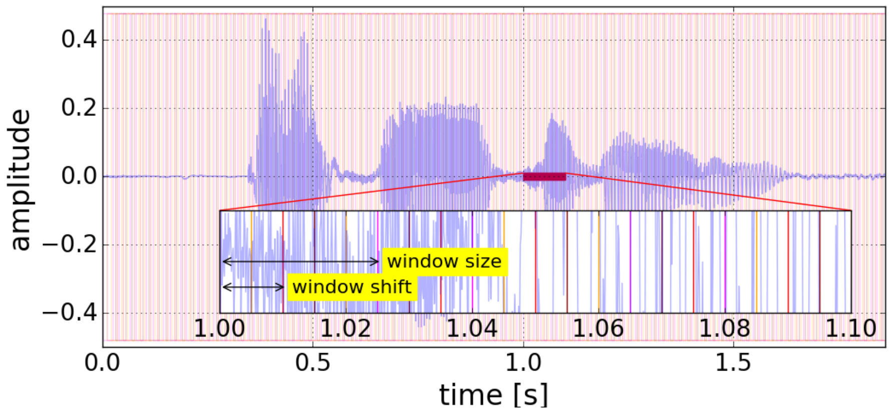


FIGURE 2.1: Framing a sound signal.

   We assume each frame captures one possible shape of the human vocal tract and therefore is capable of carrying one phoneme only. The next steps are applied for every single frame.

2. *calculation of the periodogram estimate of the power spectrum*;

   The aim is to identify which frequencies are present. In order to do so, we apply the Hamming window and perform the discrete Fourier Transform (DFT; Eq. (2.1)).

$$S(k) = \sum_{n=0}^{N-1} s_n \cdot e^{-2\pi i \frac{kn}{N}}, \qquad k = 0, ..., N-1 \tag{2.1}$$

   , where N (in this case $N = 200$) is the signal length. Then we take the absolute value $|S(k)|$.

3. *application of the mel filterbank to the power spectra, summation of the energy in each filter, taking a logarithm of the result*;

   Next, we use a filterbank of triangle filters (illustrated in Fig. 2.2) predefined on the transmitted band ($bw = \frac{f_s}{2} = 4kHz$) to get a single value per filter. We use 40 filters, therefore, each frame is now described by a vector of 40 numbers.

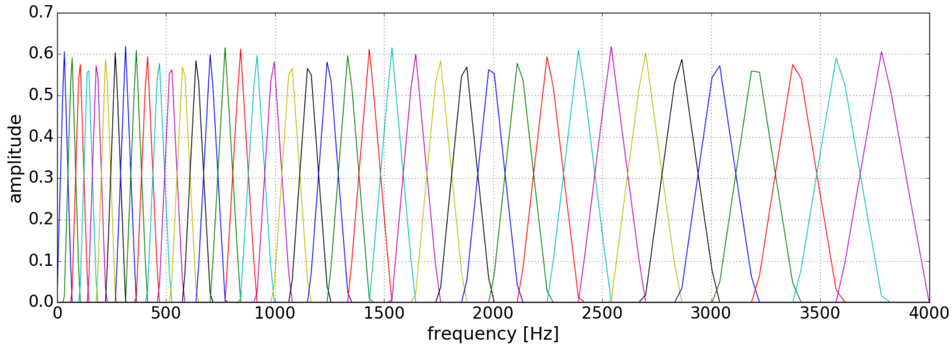


FIGURE 2.2: Mel Filterbank of 40 filters in Hertz-axis.

Finally, a logarithm of the result is taken and considered as a description of the frame (phoneme). Usually, a discrete Cosine Transform (DCT) is applied at the end, however, it is not done in this work. The result of a signal parameterization is a matrix shown in Eq. (2.2).

$$recording\_params = \begin{bmatrix} f_{11} & f_{12} & f_{13} & \cdots & f_{1F} \\ f_{21} & f_{22} & f_{23} & \cdots & f_{2F} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ f_{W1} & f_{W2} & f_{W3} & \cdots & f_{WF} \end{bmatrix} \tag{2.2}$$

, where $F = 40$ is the number of filters and W is the number of frames (windows) depending on the duration of a recording. Value f_{12} then belongs to the feature computed with the second filter in the first frame.

Data Labeling

We perform a supervised learning method, hence the data must be labeled. To the so a speech recognition method based on Hidden Markov Models (HMMs) from (Šmídl, 2017) is used. It labels the frames of each recording as shown on an example in Table 2.2.

recording_name		
frame_in	*frame_out*	*phoneme*
0	16	_sil_
16	25	a
25	32	n
32	44	o
44	65	_sil_

TABLE 2.2: Example of labeled recording.

It says that features extracted from this recording consist of 9 fourty-dimensional vectors representing phoneme `"a"`, 7 vectors of phoneme `"n"`, etc.

Forming a Sample

The 40 phonemes listed in Table 2.1 are naturally labels of classes, so we have a fourty-class classification problem. Having the information from previous section, one can match the extracted features with corresponding phonemes (classes). Now the task is to define the form of one sample.

Fig. 2.3 goes with the example in Table 2.2. The numbers in the first line are frame indices. The second line contains the known frame labels, where each frame is described by a vector of 40 features.

There is a possibility to take all frames labeled as `"a"` and consider the corresponding vectors directly as samples. However, as the labeling was not done manually and therefore cannot be considered as 100% correct, we introduce a parameter called `border_size`. Fig. 2.3 shows that we omit the frames on borders with another phoneme label and take only those in the middle.

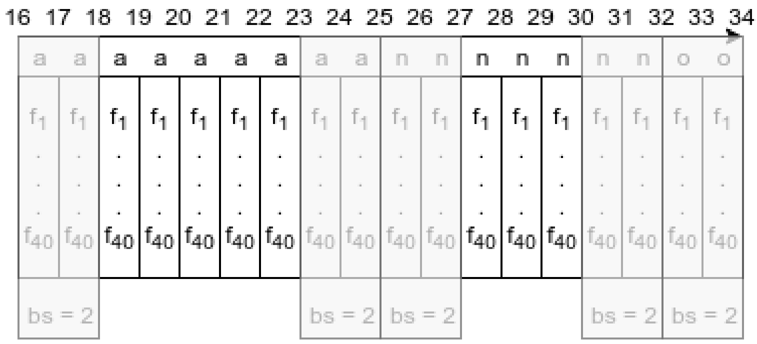


FIGURE 2.3: Forming a sample, illustration of parameter `border_size (bs)`.

Moreover, in Fig. 2.4 parameter `context_size` is introduced. The idea is to consider not only the information of one frame, but also of its context, into one sample.

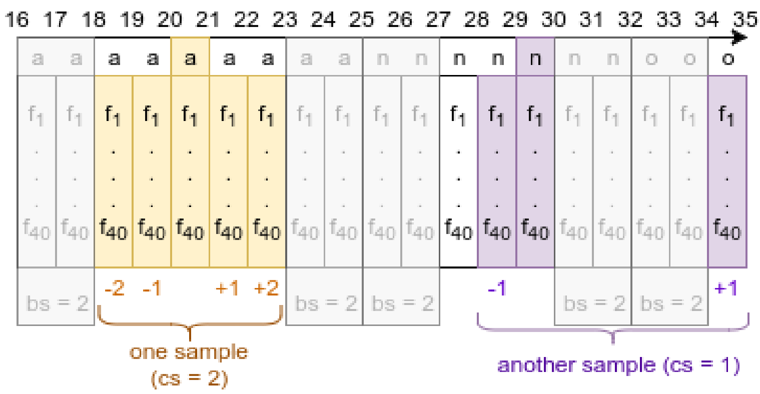


FIGURE 2.4: Forming a sample, illustration of parameter `context_size (cs)`.

Based on the chosen context size *cs* the previous and subsequent vectors are added one by one and forms one feature vector of length $40 \cdot (2cs + 1)$. An example for $cs = 2$ is illustrated in Fig. 2.5.

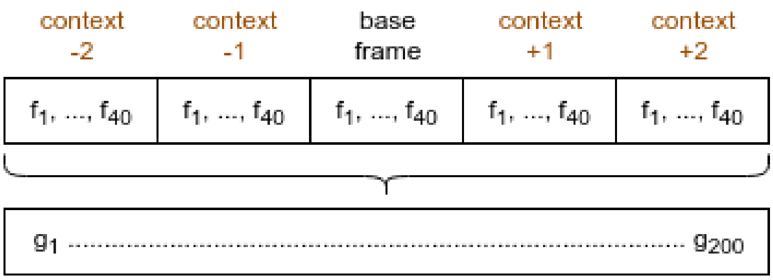


FIGURE 2.5: Example of building a feature vector with `context_size` $cs = 2$.

Talking about Fig. 2.5, features $g_1, g_2, ..., g_{200}$ gives the final feature vector of one sample, which takes the label of the base frame.

The last parameter of the speech dataset generation is the number of samples per class (`n_samples`). The rule of thumb is the more samples the better training results, however, getting best possible training results is not the objective of this work. Therefore we often use less samples to speed up the training process.

To summarize this section, we end up with three parameters of the speech dataset generation process:

- `border_size` (see Fig. 2.3)
- `context_size` (see Fig. 2.4)
- `n_samples` per class

Splitting data into three disjunctive sets

Fig. 2.6 shows a general approach of data splitting in machine learning. It is used for all classification problems in this work. The training data is used to set up model parameters. Development data is then used for testing during the training process, in order to adjust some learning parameters based on the test results. Finally, a trained model is tested on never-seen testing data. We use splitting: 80% training set; 10% development set; 10% testing set.

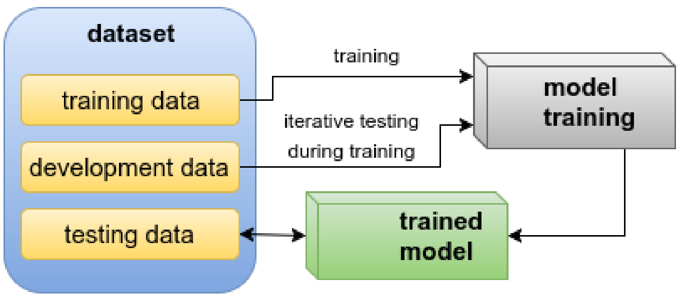


FIGURE 2.6: Using three disjunctive sets of data for a general machine learning process.

Chapter 3

Examples

The pruning algorithm is presented on several examples, where each of them has its purpose of being shown. The XOR problem (section 3.1) should verify the ability of finding an optimal network structure. Section 3.2 comes with another 2D problem, where one feature carries more information than the other one. The Rule-plus-Exception problem in section 3.3 deals with a minority of samples that has to be treated by a different net part than rule-based samples. The train problem (section 3.4) is a working example of the feature selection procedure. The MNIST database (section 3.5) is widely used in machine learning and can be regarded as commonly known, hence it is an ideal example to present new methods on. Finally, in section 3.6 the pruning algorithm is analysed on a large dataset of phonemes.

3.1 XOR Function

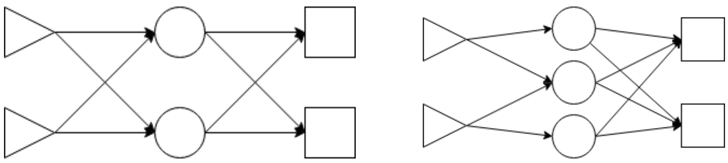
The standard Exclusive OR (XOR) function is defined by truth Table 3.1. Based on this function one can build a classification problem of two features and two classes.

x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	0

Table 3.1: XOR function.

This problem serves perfectly for a demonstration of network optimization methods, as two optimal architectural solutions producing the XOR function are already known (Fig. 3.1) [1].

[1] The known (e.g. from (Bradley, 2006)) minimal network architectures producing the XOR function $[2, 2, 1]$ and $[2, 3, 1]$ are adjusted to $[2, 2, 2]$ and $[2, 3, 2]$ in Fig. 3.1 in order to comply with the conventions introduced in chapter 2. The number of output neurons always equals the number of classes. The number of hidden-output synapses might not be optimized in this study.



(A) Solution A. (B) Solution B.

FIGURE 3.1: Optimal network architectures producing the XOR function.

With this knowledge we can prove that the pruning algorithm is (or is not) able to find the optimal solution. If the method is correct, it should end up with one of the shown architectures (Fig. 3.1a or Fig. 3.1b).

The truth Table 3.1 ruled the generation of a 2D dataset illustrated in Fig. 3.2. The two classes can be linearly separated by two lines (corresponding to two neurons, see Fig. 3.1a) and each class consists of 1000 samples. Each sample was randomly assigned to one of the two possible points belonging to its class (e.g. (0,0) or (1,1) for class 0) and then randomly placed in the surounding area within a specified range ($r = \frac{\sqrt{2}}{4}$).

The samples of each class were then splitted into three sets in the following manner: 80% to a training set, 10% to a validation set and 10% to a testing set.

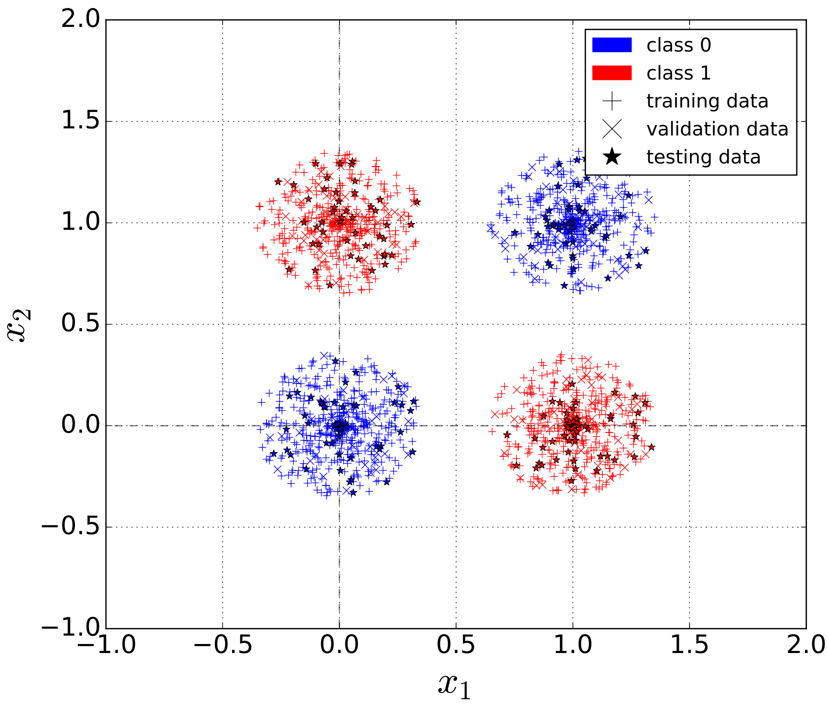


FIGURE 3.2: The XOR dataset.

The goal of this example is to show that the pruning algorithm finds one of the known minimal network structures (Fig. 3.1). An oversized network $[2, 50, 2]$ is used as the starting point. The following Table 3.2 shows all the experiment settings.

initial network		*learning parameters*		*pruning parameters*	
structure	[2, 50, 2]	learning rate	0.3	required accuracy	1.0
n synapses	200	number of epochs	50	retrain	True
transfer fcn	sigmoid	minibatch size	1	retraining epochs	50

TABLE 3.2: Experiment settings for the XOR example.

Results: XOR Function

Fig. 3.3 describes the pruning process. We can see the number of synapses, the network structure and the classification accuracy for single pruning steps (see [PA]). When the required accuracy (1.0) was not reached, the corresponding steps are transparent in the figure, indicating they were forgotten.

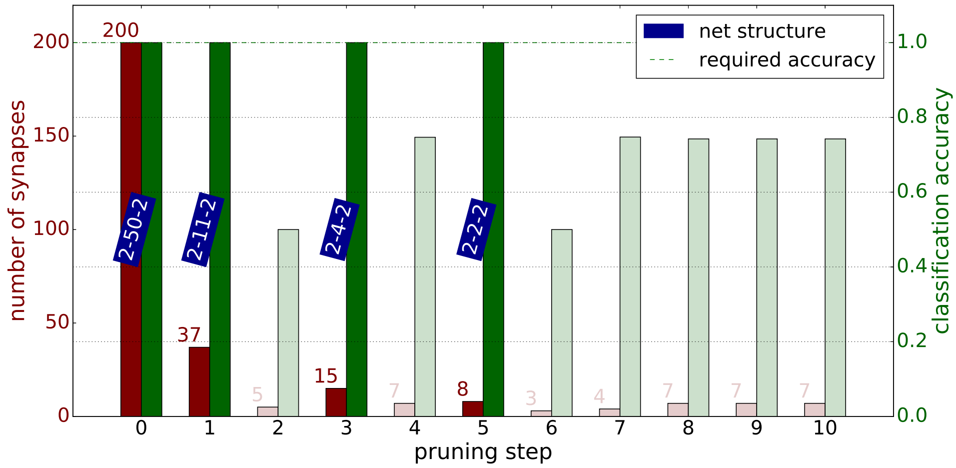
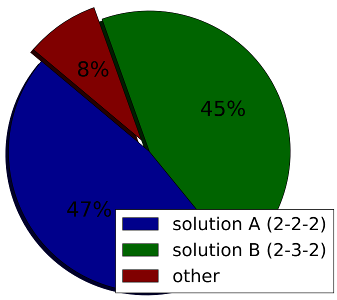
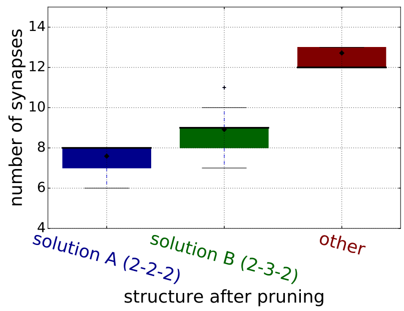


FIGURE 3.3: Illustration of the pruning procedure applied on XOR dataset (selected observation).

In Fig. 3.4 the hypothesis of this experiment is confirmed. We ran 100 observations of the experiment. In Fig. 3.4a we can see that in 47 out of 100 cases the pruning algorithm changed the network to $[2, 2, 2]$ architecture (Fig. 3.1a), in 45% of the cases it resulted with $[2, 3, 2]$ (Fig. 3.1b) and only in 8% it failed to find the optimal architecture. Fig. 3.4b gives statistics for the final number of synapses in these three cases.



(A) Network structure after pruning (100 observations).



(B) Number of synapses after pruning (100 observations).

FIGURE 3.4: Pruning results for XOR dataset.

3.2 Unbalanced Feature Information

This example is adopted from (Karnin, 1990). The problem is again two-dimensional having two non-overlapping classes as depicted in Fig. 3.5. The samples are uniformly distributed in $[-1, 1]$ x $[-1, 1]$ and the classes are equally probable, separated by two lines in 2D space ($x_1 = a$ and $x_2 = b$, where $a = 0.1$ and $b = \frac{2}{a+1} - 1 \approx 0.82$). Clearly, the problem can be solved by two neurons, similarly as the previous one.

What is interesting about this two-classes layout is that feature x_1 is much more important for the global classification accuracy than feature x_2. Having x_1 information, based on Fig. 3.5 one could potentially classify more than 90% of the samples. Opposite of that, we cannot say much with information from feature x_2 only. And this is something that also the pruning algorithm should find out.

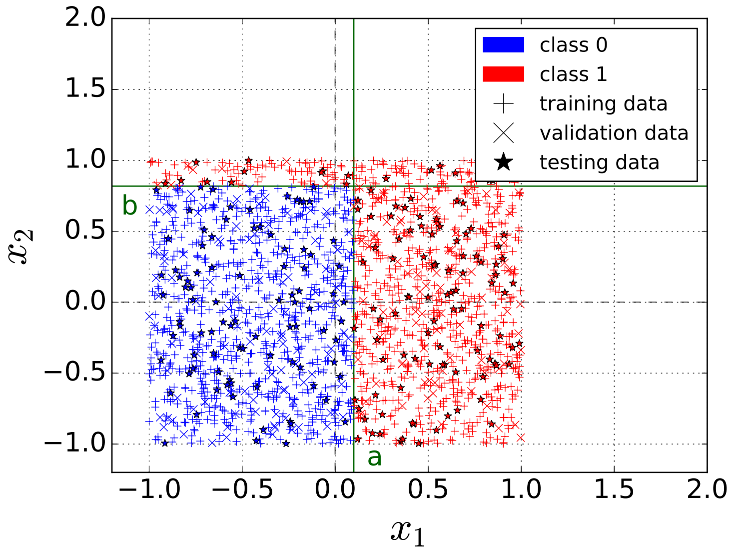
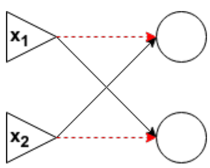
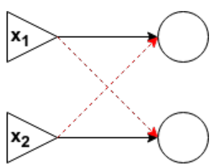


FIGURE 3.5: The UFI dataset.

Hence, we focus on synapses connecting the input and the hidden layer (shortly input-hidden synapses). We know the required network structure is $[2, 2, 2]$, as two lines are needed to separate the data in 2D space. Actually, we even know the lines must be parallel to coordinate axes, which means that each of the hidden units needs one of the features only. Therefore, the first hypothesis here is that pruning of input-hidden synapses should result in one of the cases in Fig. 3.6.



(A) Pruning result 1.



(B) Pruning result 2.

FIGURE 3.6: Expected pruning of input-hidden synapses (UFI problem).

To prove this behaviour, we ran an experiment with settings in Table 3.3.

initial network		*learning parameters*		*pruning parameters*	
structure	[2, 2, 2]	learning rate	0.7	required accuracy	0.98
n synapses	8	number of epochs	50	retrain	True
transfer fcn	sigmoid	minibatch size	1	retraining epochs	50

TABLE 3.3: Experiment settings for the UFI example.

The second hypothesis is that the synapse connected to the first feature (x_1) is more important and therefore, the other synapse (the one connected to feature x_2) should always be removed first.

Results: Unbalanced Feature Information

In Fig. 3.7 the first hypothesis is confirmed. We ran the experiment 100 times. In 48 cases, the pruning of input-hidden synapses finished with the result shown in Fig. 3.6a and it finished with the result shown in Fig. 3.6b in 44% of the cases.

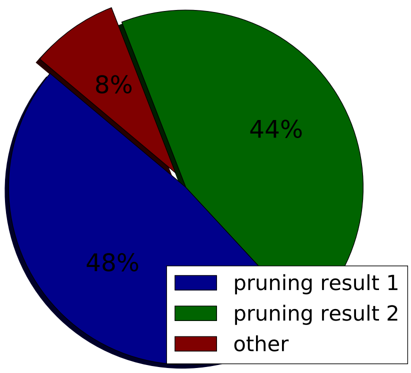


FIGURE 3.7: Results of pruning (see Fig. 3.6) input-hidden synapses (100 observations, UFI example).

In other words, with a probability of 92% the algorithm is able to find the axis-parallel lines and reveals that each of the lines needs the information from corresponding feature only. In the remaining 8% of the cases the pruning resulted with more than two input-hidden synapses.

The second hypothesis is confirmed in Fig. 3.8. The synapses's significance factor was always (100 observations) greater for the synapse coming from feature x_1 than for the synapse connected to x_2. By definition (see [PA]), the pruning method eliminates the synapses with low significance factors first, therefore the information coming from feature x_1 would live longer in the network than the x_2 information.

Let's try to explain this result. Consider w_{r1} to be the weight of the synapse connecting the x_1 feature and r^{th} hidden neuron (with bias b_r) and w_{s2} to be the weight of the synapse coming from feature x_2 to s^{th} hidden neuron (with bias b_s), then by neuron definition (Rosenblatt, 1958) we created two lines, perpendicular one to each other, as follows.

$$w_{r1} \cdot x_1 + 0 \cdot x_2 + b_r = 0 \tag{3.1}$$

$$x_1 = -\frac{b_r}{w_{r1}} \tag{3.2}$$

$$0 \cdot x_1 + w_{s2} \cdot x_2 + b_s = 0 \tag{3.3}$$

$$x_2 = -\frac{b_s}{w_{s2}} \tag{3.4}$$

In Fig. 3.5 we see that $a < b$. To generalize the problem (assuming normalised feature vectors, see chapter 2) we state $|a| < |b|$, meaning we want:

$$|-\frac{b_r}{w_{r1}}| < |-\frac{b_s}{w_{s2}}| \tag{3.5}$$

Hence we expect:

$$|w_{r1}| > |w_{s2}| \tag{3.6}$$

Out of this we expect a weight magnitude to be greater for more important synapses.

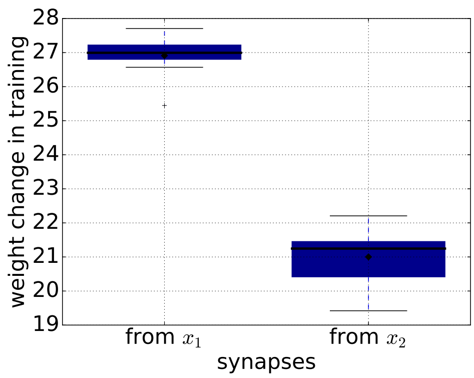


FIGURE 3.8: Weight change in training for the remaining input-hidden synapses (100 observations).

The weight magnitude seems to be a perfect measure to find feature significance factor. However, as we do not use a weight decay (see the learning approach in chapter 2), in general the more epochs we learn the greater weight magnitudes we get. Therefore small initial weight values do not affect the result significantly and so we can state:

$$|w_{ji}(t)| \approx |w_{ji}(t) - w_{ji}(0)| \tag{3.7}$$

where $w_{ji}(0) \in N(0, 1)$ is the initial value of weight w_{ji} and t is time. Summing it up we can say that the `kitt` measure [ref] based on weight change is equally good as the magnitude measure assuming enough training epochs (e.g. 50).

3.3 Rule-plus-Exception

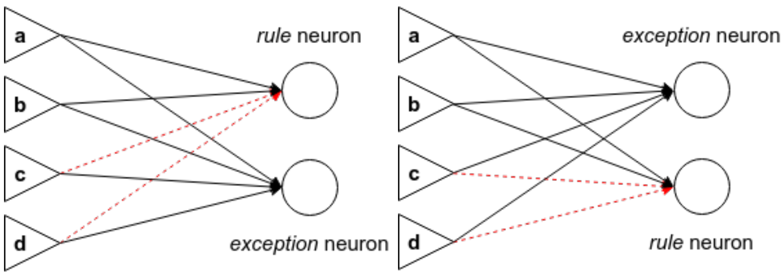
This four-dimensional problem is originally adopted from (Mozer and Smolensky, 1989) and is also used in (Karnin, 1990). The task is to learn another Boolean function: $AB + \overline{A}\,\overline{B}\,\overline{C}\,\overline{D}$. A single function output should be on (i.e. equals 1) when both A and B are on, which is the *rule*, and it should also be on when the *exception* $\overline{A}\,\overline{B}\,\overline{C}\,\overline{D}$ occurs.

Clearly, the *rule* occurs more often than the *exception*, therefore the samples corresponding with the *rule* should be more important for the global classification accuracy. The hypothesis is that the pruning method should suggest the part of network, which deals with the *exception*, to be eliminated first - before network elements dealing with the *rule*.

To test this hypothesis, a dataset of 10000 samples was generated. Each sample consists of four features: $[a, b, c, d]$. Each of these features (for every sample) was randomly set to be *on* (1) or *off* (0). Then, whenever the *rule* occured ($a = 1 \wedge b = 1$), the sample was assigned to class 1 (as a *rule* sample). If the *exception* occured ($a = 0 \wedge b = 0 \wedge c = 0 \wedge d = 0$), the sample was also labeled as 1 (as an *exception* sample). Otherwise, the sample was assigned to class 0. Thereby the generated dataset consisted of:

- 2511 *rule* samples (class 1);
- 649 *exception* samples (class 1);
- 6840 samples in class 0.

The function is expected to be learned by two hidden neurons, one dealing with the *rule* and the other one with the *exception*. We focus on input-hidden synapses again. Two possible expected pruning results are shown in Fig. 3.9.



(A) Pruning result 1. (B) Pruning result 2.

FIGURE 3.9: Expected pruning of input-hidden synapses (RPE problem).

We ran the learning-pruning procedure with the settings listed in Table 3.4.

initial network		*learning parameters*		*pruning parameters*	
structure	[2, 2, 2]	learning rate	1.0	required accuracy	1.0
n synapses	8	number of epochs	50	retrain	True
transfer fcn	sigmoid	minibatch size	1	retraining epochs	50

TABLE 3.4: Experiment settings for the RPE example.

Results: Rule-plus-Exception

Fig. 3.10 supports the hypothesis that one hidden neuron forms the *rule* and the other one the *exception*. With a probability of 97% the pruning finished with one of the structures in Fig. 3.9.

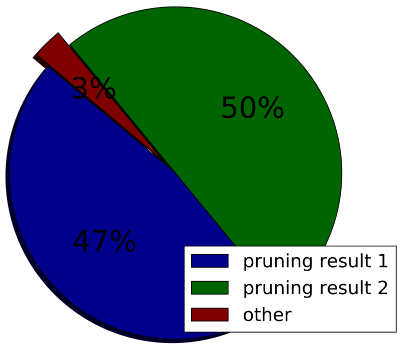


FIGURE 3.10: Results of pruning (see Fig. 3.9) input-hidden synapses (100 observations, RPE example).

The same thing is confirmed by Fig. 3.11. It shows weight change in training (`kitt` significance factor) for all 8 input-hidden synapses. We can see that synapses connecting the *rule* neuron with feature c (s_{rc}) and feature d (s_{rd}) were suggested as least important (resulted in structures in Fig. 3.9).

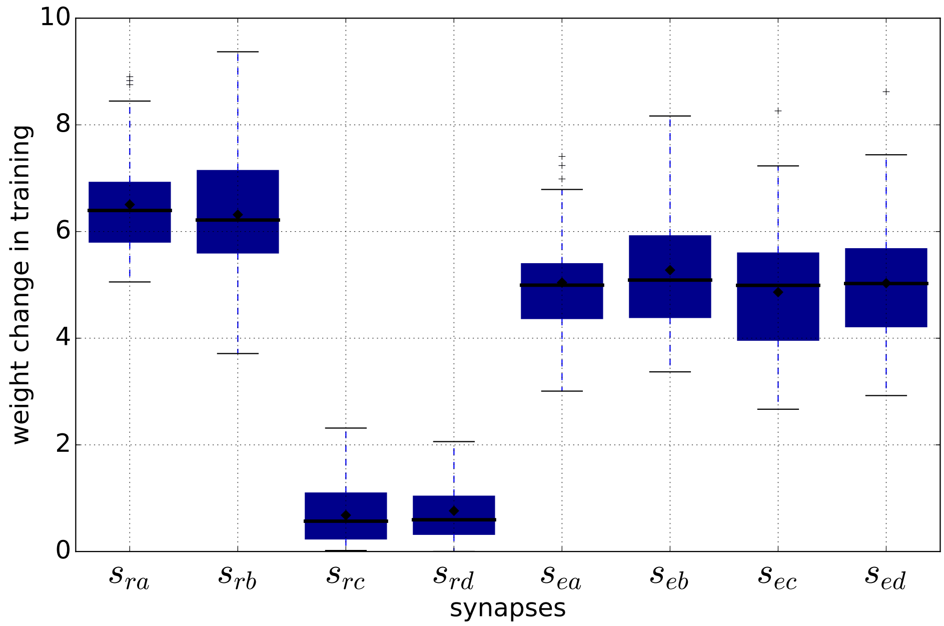


FIGURE 3.11: Weight change in training for input-hidden synapses (100 observations, RPE example).

Additionally, synapses responsible for *rule* (s_{ra} and s_{rb}) have a greater mean significance than the synapses connected with the *exception* neuron (s_{e*}).

3.4 Michalski's Trains

The train problem was originally introduced in (Larson and Michalski, 1977). The task was to determine concise decision rules distinguishing between two sets of trains (Eastbound and Westbound). In (Mozer and Smolensky, 1989), they presented a simplified version illustrated in Fig. 3.12.

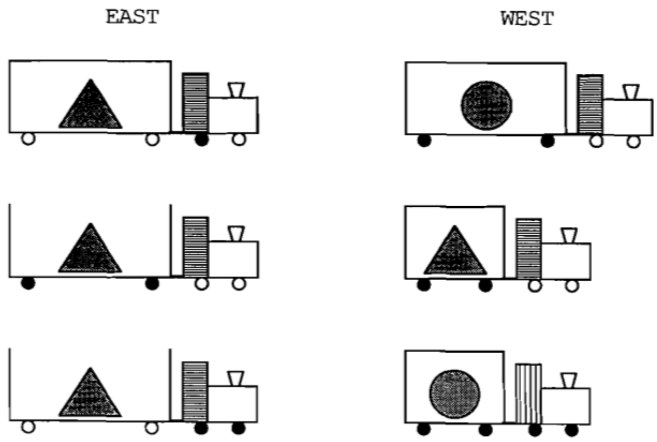


FIGURE 3.12: Michalski's train problem.

Each train is described by 7 binary features listed in Table 3.5.

	feature	*encoded as **0***	*encoded as **1***
0	car length	long	short
1	car type	open	closed
2	cabin pattern	vertical lines	horizontal lines
3	load shape	triangle	circle
4	color of trailer wheels	white	black
5	color of first car wheel	white	black
6	color of second car wheel	white	black

TABLE 3.5: Features describing a train.

Having Table 3.5 we can encode the trains shown in Fig. 3.12 into feature vectors as follows in Table 3.6.

class EAST		*class WEST*	
east 1	$[0,1,1,0,0,0,1]^T$	west 1	$[0,1,1,1,1,0,0]^T$
east 2	$[0,0,1,0,1,0,0]^T$	west 2	$[1,1,1,0,1,0,0]^T$
east 3	$[0,0,1,0,0,1,1]^T$	west 3	$[1,1,0,1,1,1,1]^T$

TABLE 3.6: Feature vectors for different train types.

The task is to determine the minimal number of input features capable of the east-west classification based on the six possible types in Table 3.6 (or in Fig. 3.12).

The hypothesis is that the pruning algorithm should select the needed features by eliminating unimportant input-hidden synapses. Looking at Fig. 3.12 one of the solutions could be keeping features $(0, 3)$, because the shape of the load together with the length of the car is enough to distinguish *west* trains from *east* trains. Another solution, for example, is keeping the car length, car type and color of the second car wheel - features $(0, 1, 6)$.

To test our pruning algorithm on this feature selection task, a dataset of 6000 samples (3000 west and 3000 east trains) was generated. The three possible train types for each class (Fig. 3.12) are equally distributed among the samples, meaning we have 1000 samples of each train type.

As shown in (Mozer and Smolensky, 1989), one hidden neuron is enough to learn this problem, hence we started with the network structure $[7, 1, 2]$. The experiment parameters are listed in Table 3.7.

initial network		*learning parameters*		*pruning parameters*	
structure	[7, 1, 2]	learning rate	0.3	required accuracy	1.0
n synapses	9	number of epochs	100	retrain	True
transfer fcn	sigmoid	minibatch size	1	retraining epochs	10

TABLE 3.7: Experiment settings for the train example.

We ran 100 observations of the experiment and considered the features that were not cut out, as a result of a single experiment.

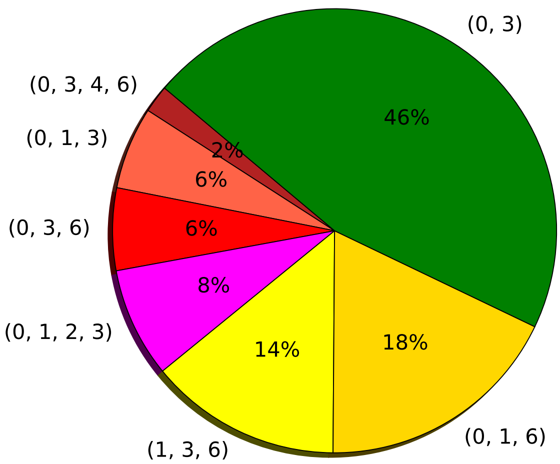


FIGURE 3.13: Results of feature selection by the pruning algorithm (train example). The labels corresponds with feature indices in Table 3.5.

The result pie in Fig. 3.13 shows that the pruning algorithm found the best possible solution ($(0, 3)$ - the car length and the load shape) in 46% of the cases. We can regard the $(0, 1, 6)$ and $(1, 3, 6)$ as another (not best but also good) solutions. The rest we consider as fail cases, as all of them include features $(0, 3)$ and the other features are redundant. To sum it up, we got a perfect solution: 46%; a good solution: 32%; a bad solution: 22%.

3.5 Handwritten Digits (MNIST)

The MNIST (Modified National Institute of Standards and Technology) database (Wikipedia, 2004) is a large database of handwritten digits that is widely used for training and testing methods in the field of machine learning.

The dataset was downloaded from (LeCun and Cortes, 1998). Some of the digits were written by employees of American Census Bureau (*United States Census Bureau* 2017) and some by students of an American high school. In total 70000 samples were collected. Examples are shown in Fig. 3.14.

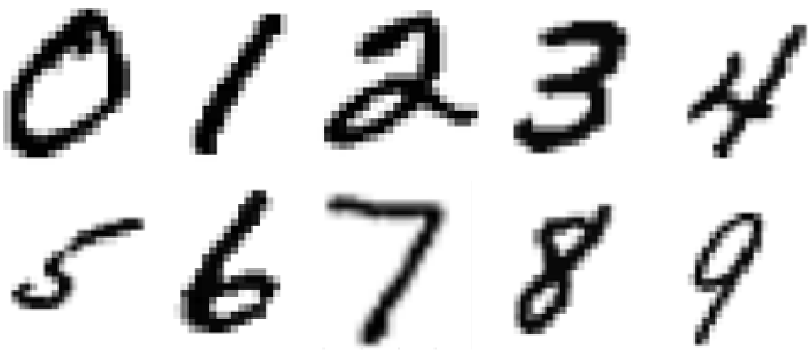


FIGURE 3.14: Examples of MNIST dataset.

Each sample is a grayscale image (normalised to $[0, 1]$) of size $28x28$ pixels. This gives row-by-row a vector of 784 features. The data was splitted into a training set of 50000 samples, a validation set of 10000 samples and a testing set of 10000 samples.

From (LeCun and Cortes, 1998) we know the problem can be learnt by a feedforward network with one hidden layer up to high accuracy ($98 - 99\%$). The first task is to achieve similar results with the implemented neural net framework. We tested the following learning settings (Table 3.8).

network parameters		*learning parameters*	
structure	[784, 20, 10]	learning rate	0.3
n synapses	15880	number of epochs	100
transfer function	sigmoid	batch size	10

TABLE 3.8: Settings for training a dense feedforward net on the MNIST dataset.

The training results are summarized in Table 3.9. A confusion matrix for the testing data is given in Fig. 3.15.

	accuracy	*MSE*
training data	97.2%	0.526
testing data	94.3%	1.025

TABLE 3.9: Training results on MNIST dataset.

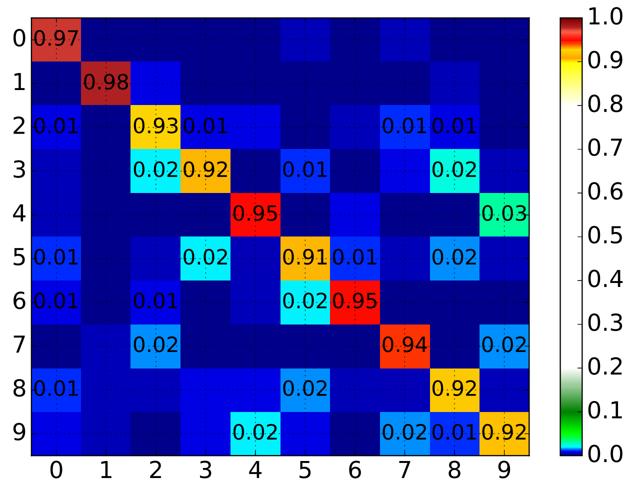


FIGURE 3.15: Confusion matrix (MNIST, testing data).

In the following, the pruning method is analysed on networks trained on the MNIST database. Parameters of the learning-pruning procedure are listed in Table 3.10.

initial network		*learning parameters*		*pruning parameters*	
structure	[784, 20, 10]	learning rate	0.3	required accuracy	0.97
n synapses	15800	number of epochs	30	retrain	True
transfer fcn	sigmoid	minibatch size	10	retraining epochs	10

TABLE 3.10: Experiment settings for the MNIST example.

The hypothesis is that the initial number of synapses in the network (15800) is redundant, as well as the number of features (784). In Fig. 3.16 we can see a selected observation of the pruning process. The number of synapses was reduced to 1259 and the number of used features to 465, while the classification accuracy was kept on 97%. The pruning procedure finished in 424 pruning steps (explained in [PA]).

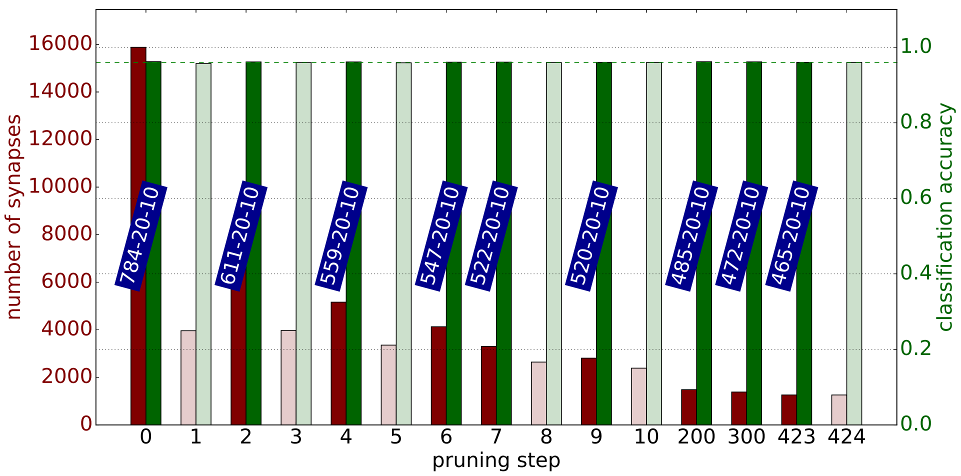


FIGURE 3.16: Illustration of the pruning procedure applied on MNIST dataset (selected observation). Required accuracy: 97%.

In Fig. 3.17, we can see a comparison of the evaluation time. We compare a fully-connected (initial) network to a pruned one. Bars are given for the three data groups (training: 50000 samples, validation: 10000 samples, testing: 10000 samples). The processing time was reduced by nearly half after the pruning, which led to a reduction of weight matrix dimension (see [SHRINK]).

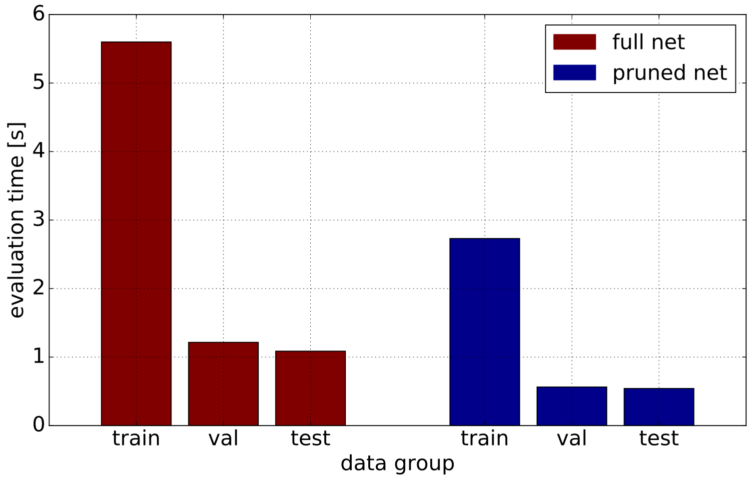


FIGURE 3.17: Evaluation (accuracy and error computation) time for all data groups (pruned vs. full net).

Fig. 3.18 gives the statistics by running 10 observations of the pruning procedure for several values of required classification accuracy. We observed the number of synapses (red axis) and used features after pruning (blue axis).

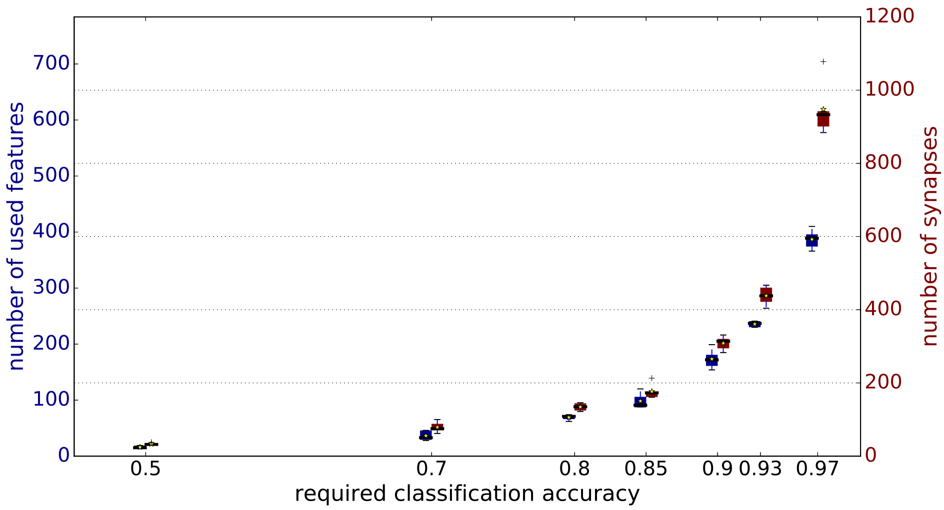


FIGURE 3.18: Minimal number of features and synapses to get required classification accuracy (MNIST data).

The results show that *less than a tenth* of the synapses and *about a half* of the features are needed to keep the maximal classification accuracy (97%). It is also worth saying that the MNIST dataset can be learnt to 50% using only 20 features and a network with 38 synapses. In the following, these two results are further analysed.

Minimal MNIST network

At first, we focus on a pruned network capable of MNIST classification with accuracy of 50% (Fig. 3.19). This example is simple enough to show the feature selection method described in [ref FS].

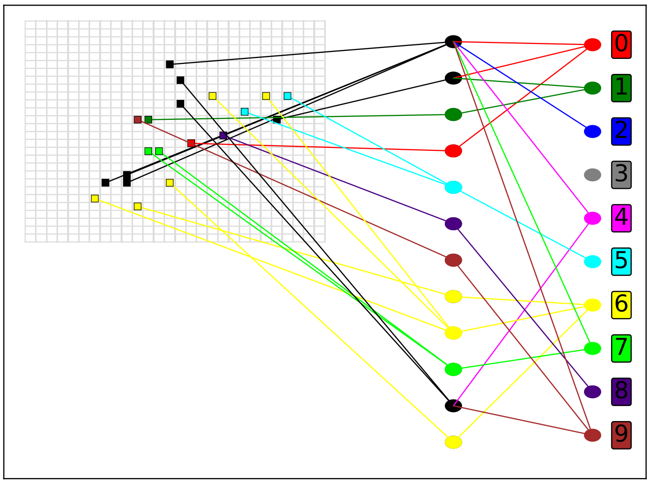


FIGURE 3.19: Result of network pruning and path tracking, MNIST data, accuracy: 50%.

Each class (digit in this case) has its color. If a hidden unit has one output connection only, it inherits the color of the class it is connected to. The features (pixels of the 28x28 image) are then colored in the same way. If a hidden unit influences more than one class, it is blacked. All features connected to a black hidden unit are then blacked as well, as they also affects more than one class.

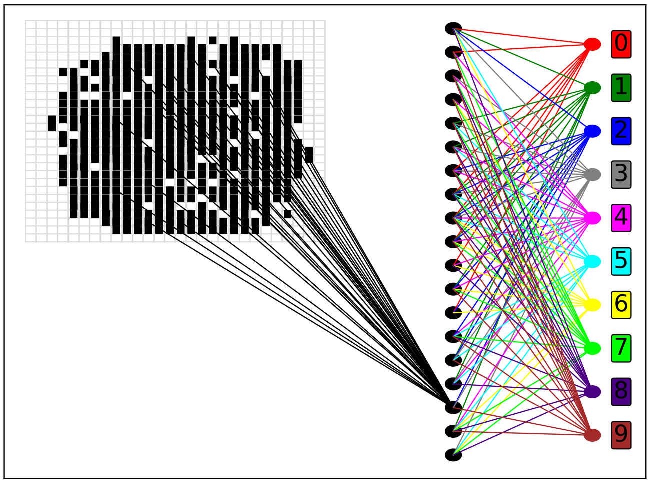


FIGURE 3.20: Result of network pruning and path tracking (shown 17^{th} hidden neuron only), MNIST data, accuracy: 97%.

A pruned network capable of 97% accurate classification is visualized in Fig. 3.20. To make the figure clearer, only synapses coming to the 17^{th} hidden unit are drawn between the input and hidden layer.

We can see that each of the features affects more than one class in this case. Therefore we better use the visualization in Fig. 3.21.

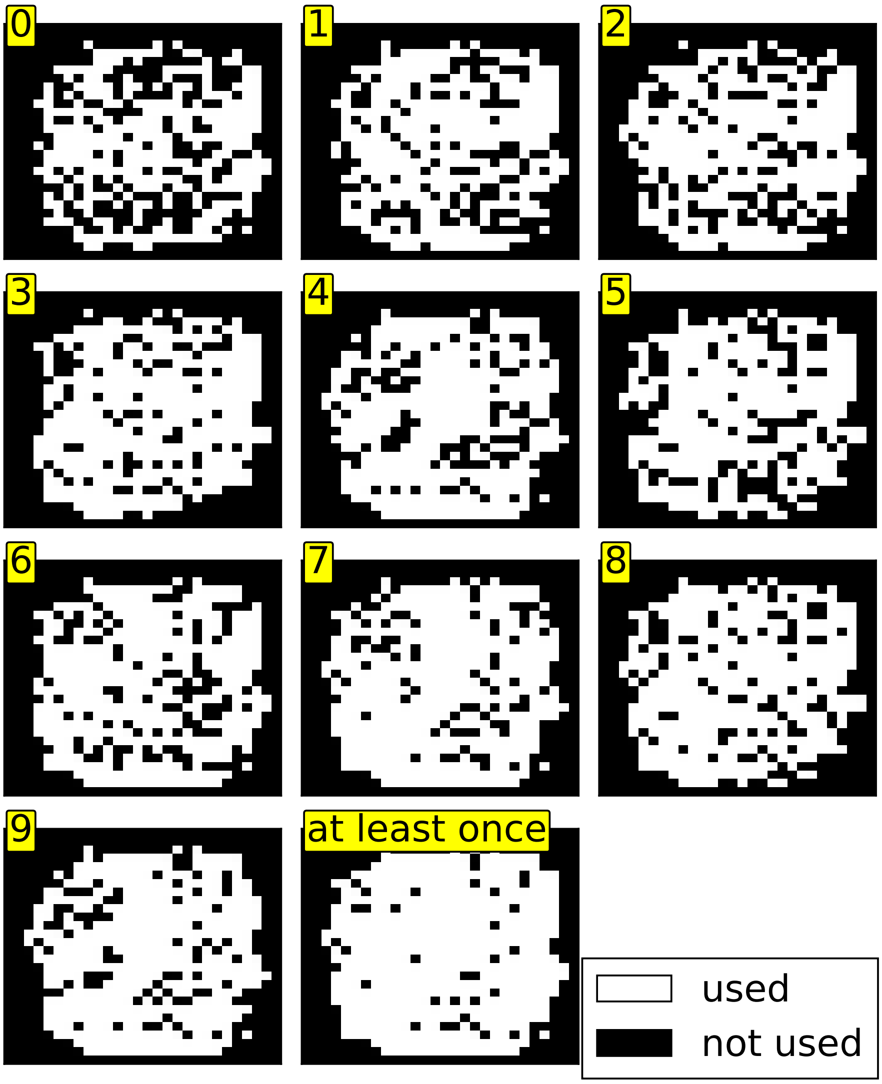


FIGURE 3.21: Used features for individual classes, MNIST data, accuracy: 97%.

Knowing all the remaining synapses are important for classification, we can track tha paths from individual classes to features. This way we distinguish features connected to a selected class from those that do not affect that class. Fig. 3.21 shows important features for each class (digit) separately. Note that the *at-least-once* subplot corresponds to the features shown in Fig. 3.20. It shows all features used by at least one class.

Some more ideas about path tracking in pruned networks are further discussed in section 4.3.

3.6 Phonemes (Speech Data)

The process of speech data gathering is described in section 2.4. The dataset generation process has three parameters: `border_size` (bs), `context_size` (cs) and `n_samples` (ns).

In this example, we first try to find optimal parameter settings, which would lead to a maximal trainability. The general rule is the more samples the better trainability, therefore we fix $ns = 1000$ and determine the other parameters first. See Table A1.1 for details of all generated datasets differing in bs and cs. It reveals that phoneme `"F"` does not have enough occurrences (less than $ns = 1000$) in the data, and of course the number of these occurrences decreases with growing bs. For $bs >= 6$ even more phonemes (`"D"`, `"F"`,`"N"`, `"Q"`, `"R"`, `"T"`) have less than 1000 occurrences.

Table 3.11 shows the experiment settings.

experiment settings		learning parameters	
n observations	5	learning rate	0.1
observed value	MSE' (Eq. X)	n epochs	50
network structure	$[40 \cdot (2cs + 1), 50, 40]$	batch size	10

TABLE 3.11: Speech dataset: experiment settings for determination of bs and cs.

We ran 5 observations of a simple network training for every combination of $bs \in [0, 5]$ and $cs \in [0, 9]$. Fig. 3.22 shows average MSE' (see Eq. X) values, complete results can be found in Table X.

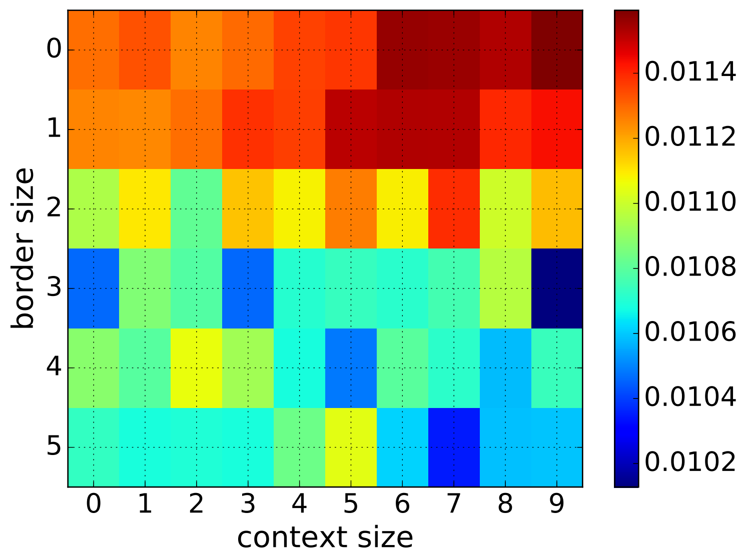


FIGURE 3.22: Test MSE' (Eq ref) for various parameters bs and cs ($ns = 1000$, 5 observations, see Table A1.1).

Based on Fig. 3.22 we can state that the `border_size` must be greater than two and we set $bs = 3$ for further experiments. The `context_size` was set to $cs = 3$.

Chapter 4

Discussion

Discussion text...

4.1 Methods Recapitulation

Methods recapitulation text...

4.2 Comparison of Pruning Methods

Comparison of results text...

Random

Magnitude

Karnin

OBD

Kitt

accuracy, convergence time, computation time, number of synapses/features (viz Tomas)

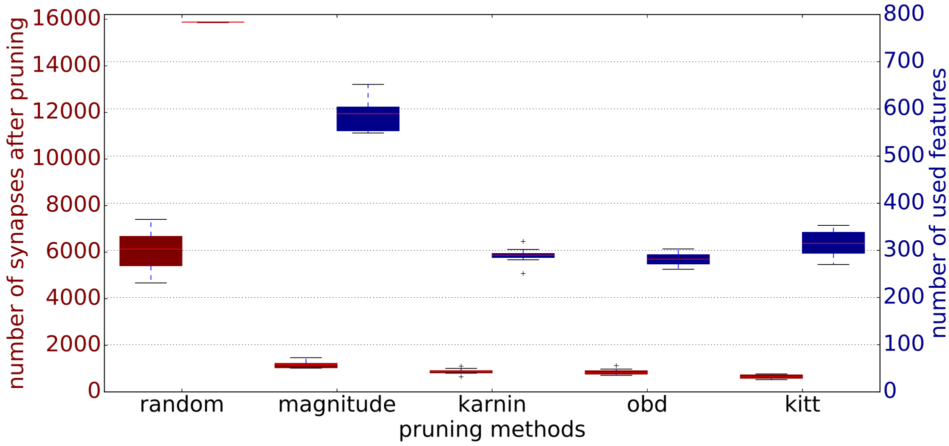


FIGURE 4.1: MNIST, req_acc = 0.95, retraining: 5 epochs

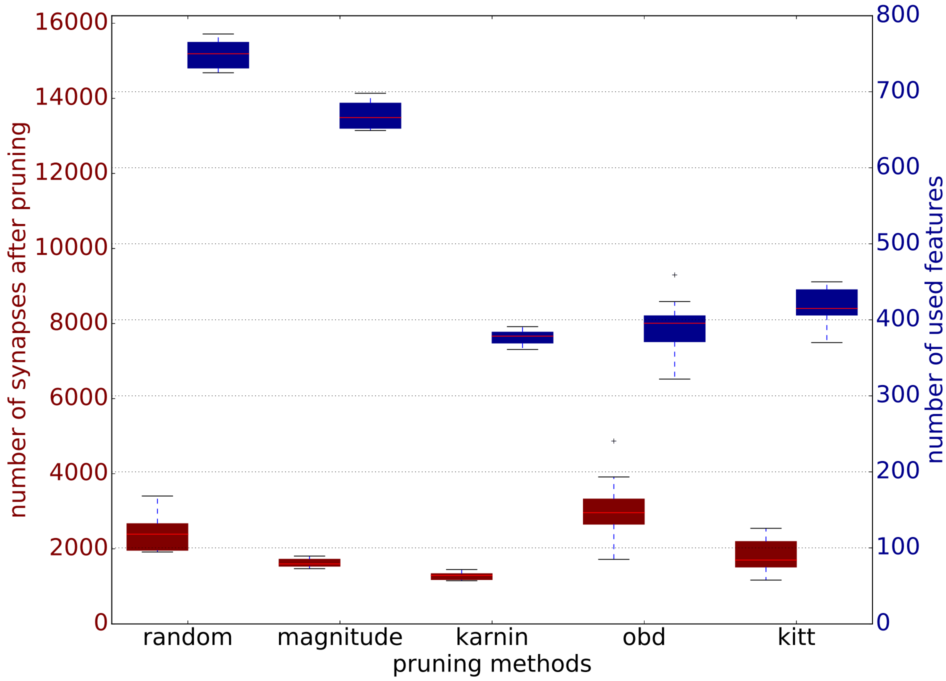


FIGURE 4.2: MNIST, req_acc = 0.95, no retraining

4.3 Future Work

Outlook...

Shrinking of layers

Tailoring

Building net from zero

Sphere neuron

Chapter 5

Conclusion

Conclusion text...

Outlook text... shrinking layers?

Bibliography

[1] Frank Rosenblatt. “The perceptron: A probabilistic model for information storage and organization in the brain”. In: *Psychological Review* 65 (1958), pp. 386–408.

[2] J. Larson and R. S. Michalski. “Inductive Inference of VL Decision Rules”. In: *ACM SIGART Bulletin*. New York, USA: ACM SIGAI, 1977, pp. 38–44.

[3] Michael C. Mozer and Paul Smolensky. “Skeletonization: A Technique for Trimming the Fat from a Network via Relevance Assessment”. In: *Boulder*. Institute of Cognitive Science, University of Colorado: CO 80309-0430, 1989, pp. 107–115.

[4] Ehud D. Karnin. “A Simple Procedure for Pruning Back-Propagation Trained Neural Networks”. In: *Letters*. IEEE Transaction on Neural Networks, 1990, pp. 239–242.

[5] R. Reed. “Pruning Algorithms - A Survey”. In: *IEEE Transactions on Neural Networks (Volume:4 , Issue: 5)* (Sept. 1993), pp. 740–747. URL: `http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=248452`.

[6] Yann LeCun and Corinna Cortes. *The MNIST database of handwritten digits*. 1998. URL: `http://yann.lecun.com/exdb/mnist/`.

[7] Wikipedia. *Plagiarism — Wikipedia, The Free Encyclopedia*. [Online; accessed 15-April-2017]. 2004. URL: `https://en.wikipedia.org/wiki/MNIST_database`.

[8] Peter Bradley. *The XOR Problem and Solution*. 2006. URL: `http://www.mind.ilstu.edu/curriculum/artificial_neural_net/xor_problem_and_solution.php`.

[9] James Lyons. *Mel Frequency Cepstral Coefficient (MFCC) tutorial*. 2009. URL: `http://practicalcryptography.com/miscellaneous/machine-learning/guide-mel-frequency-cepstral-coefficients-mfccs/`.

[10] Martin Bulín. “Classification of Terrain based on Proprioception and Tactile Sensing for Multi-legged Walking Robot”. MA thesis. Campusvej 55, 5230 Odense M: University of Southern Denmark, 2016.

[11] *United States Census Bureau*. 2017. URL: `https://www.census.gov/`.

[12] Luboš Šmídl. personal communication. supervision of the thesis. 2017.

Appendix A1

Supplementary Data

Generated Speech Datasets

id	*bs*	*cs*	*ns*	*incomplete classes*	total	*train*	*devel*	*test*
ds_00	0	0	1K	F (683)	39683	31747	3968	3968
ds_01	0	1	1K	F (683)	39683	31747	3968	3968
ds_02	0	2	1K	F (683)	39683	31747	3968	3968
ds_03	0	3	1K	F (683)	39683	31747	3968	3968
ds_04	0	4	1K	F (683)	39683	31747	3968	3968
ds_05	0	5	1K	F (683)	39683	31747	3968	3968
ds_06	0	6	1K	F (683)	39683	31747	3968	3968
ds_07	0	7	1K	F (683)	39683	31747	3968	3968
ds_08	0	8	1K	F (683)	39683	31747	3968	3968
ds_09	0	9	1K	F (683)	39683	31747	3968	3968
ds_10	1	0	1K	F (589)	39589	31672	3959	3958
ds_11	1	1	1K	F (589)	39589	31672	3959	3958
ds_12	1	2	1K	F (589)	39589	31672	3959	3958
ds_13	1	3	1K	F (589)	39589	31672	3959	3958
ds_14	1	4	1K	F (589)	39589	31672	3959	3958
ds_15	1	5	1K	F (589)	39589	31672	3959	3958
ds_16	1	6	1K	F (589)	39589	31672	3959	3958
ds_17	1	7	1K	F (589)	39589	31672	3959	3958
ds_18	1	8	1K	F (589)	39589	31672	3959	3958
ds_19	1	9	1K	F (589)	39589	31672	3959	3958
ds_20	2	0	1K	F (498)	39498	31599	3950	3949
ds_21	2	1	1K	F (498)	39498	31599	3950	3949
ds_22	2	2	1K	F (498)	39498	31599	3950	3949
ds_23	2	3	1K	F (498)	39498	31599	3950	3949
ds_24	2	4	1K	F (498)	39498	31599	3950	3949
ds_25	2	5	1K	F (498)	39498	31599	3950	3949
ds_26	2	6	1K	F (498)	39498	31599	3950	3949
ds_27	2	7	1K	F (498)	39498	31599	3950	3949
ds_28	2	8	1K	F (498)	39498	31599	3950	3949
ds_29	2	9	1K	F (498)	39498	31599	3950	3949
ds_30	3	0	1K	F (410)	39410	31528	3941	3941
ds_31	3	1	1K	F (410)	39410	31528	3941	3941
ds_32	3	2	1K	F (410)	39410	31528	3941	3941
ds_33	3	3	1K	F (410)	39410	31528	3941	3941
ds_34	3	4	1K	F (410)	39410	31528	3941	3941
ds_35	3	5	1K	F (410)	39410	31528	3941	3941
ds_36	3	6	1K	F (410)	39410	31528	3941	3941
ds_37	3	7	1K	F (410)	39410	31528	3941	3941
ds_38	3	8	1K	F (410)	39410	31528	3941	3941
ds_39	3	9	1K	F (410)	39410	31528	3941	3941
ds_40	4	0	1K	F (327)	39327	31462	3933	3932
ds_41	4	1	1K	F (327)	39327	31462	3933	3932

id	*bs*	*cs*	*ns*	*incomplete classes*	total	*train*	*devel*	*test*
ds_42	4	2	1K	F (327)	39327	31462	3933	3932
ds_43	4	3	1K	F (327)	39327	31462	3933	3932
ds_44	4	4	1K	F (327)	39327	31462	3933	3932
ds_45	4	5	1K	F (327)	39327	31462	3933	3932
ds_46	4	6	1K	F (327)	39327	31462	3933	3932
ds_47	4	7	1K	F (327)	39327	31462	3933	3932
ds_48	4	8	1K	F (327)	39327	31462	3933	3932
ds_49	4	9	1K	F (327)	39327	31462	3933	3932
ds_50	5	0	1K	F (253)	39253	31403	3925	3925
ds_60	6	0	1K	D, F, N, Q, R, T	38049	30441	3805	3803
ds_70	7	0	1K	D, F, N, Q, R, T, Z	36140	28914	3614	3612
ds_80	8	0	1K	D, F, N, Q, R, T, Z	34869	27899	3487	3483
ds_90	9	0	1K	D, F, N, Q, R, T, Z, b	33680	26947	3368	3365
ds_5K	3	5	5K	D, F, N, Q, R, T, Y, Z, g	184750	147803	18475	18472
ds_10K	3	5	10K	D, F, N, Q, R, T, U, Y, Z, g, x	335812	268654	33581	33577

TABLE A1.1: Datasets generated for the *Phonemes* example (section 3.6).

Results of Setting Speech Dataset Parameters

Appendix A2

Structure of the Workspace

Appendix A3

Implementation

Appendix A4

Code Documentation