UNIVERSITY
OF WEST BOHEMIA

Master Thesis

Optimization of Neural Network

Author:
Martin Bulín MSc.

Supervisor:
Ing. Luboš Šmídl Phd.

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

in the

DEPARTMENT OF
CYBERNETICS

April 15, 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. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- 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
PA	**P**runing **A**lgorithm
RPE	**R**ule **P**lus **E**xception

Chapter 1

Introduction

Introduction text...

1.1 State of the Art

State of the art text... (Rosenblatt, 1958) (Reed, 1993)

1.2 Thesis Objectives

Thesis objectives text...

1.3 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...

2.3 Network Visualization

Graphical user interface text...

2.4 Speech Data Gathering

Speech data classification text...

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 feature selection procedure. The MNIST database (section 3.5) is widely used in machine learning and can be regarded as commonly known. Therefore it is a good example for presentation of new methods. Finally, in section 3.6 the pruning algorithm is applied on a large dataset of phonemes.

3.1 2D-problem: XOR Function

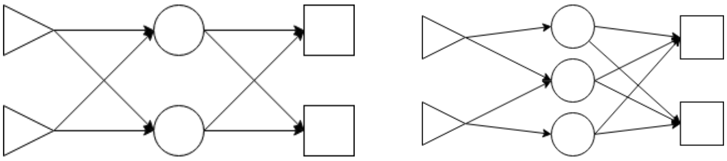
The standard Exclusive OR (XOR) function is defined by truth Table 3.1. Based on this function one can build a classification problem with two features and two classes.

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 demonstration of network optimization methods, as two optimal architectural solutions producing the XOR function are 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 synapses connected to the output layer is a subject to think about.



(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 (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 suronding 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.

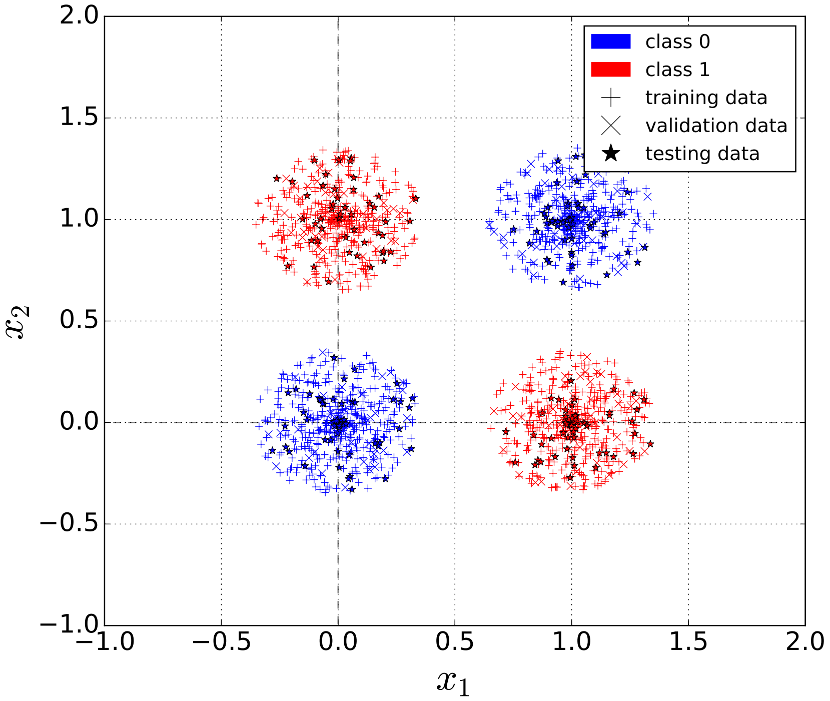


Figure 3.2: The XOR dataset.

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
transfer fcn	sigmoid	number of epochs	50	retrain	True
		minibatch size	1	retraining epochs	50

Table 3.2: Experiment settings for XOR dataset.

Results: XOR Function

Fig. 3.3 describes the pruning process. We can see the number of synapses (starting with 200 for fully-connected structure [2, 50, 2]), the network structure and classification accuracy at 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.

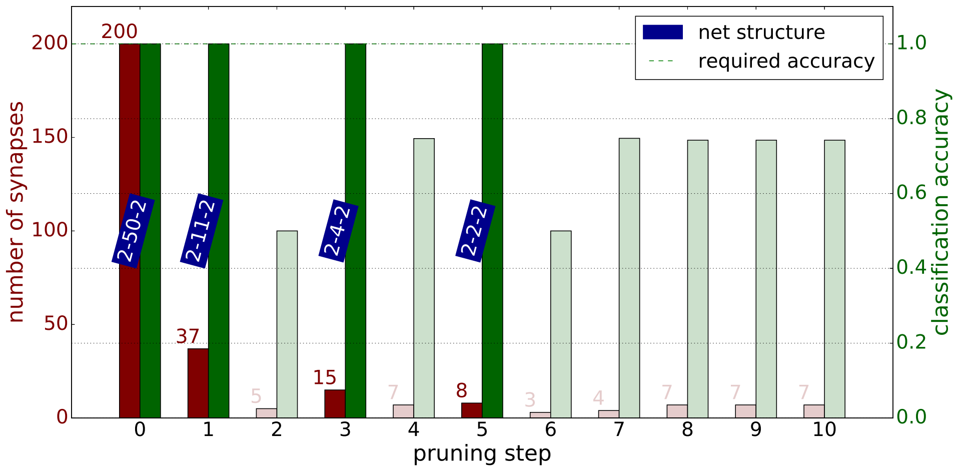
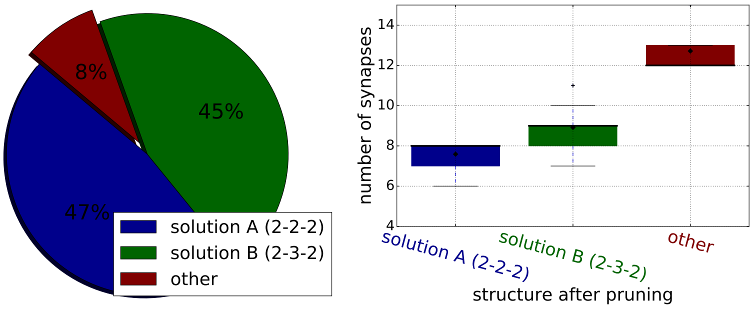


Figure 3.3: Illustration of the pruning procedure applied on XOR dataset (selected observation).

In Fig. 3.4 the hypothesis of this experiment is confirmed. 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 2D-problem: 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 what also the pruning algorithm should find out.

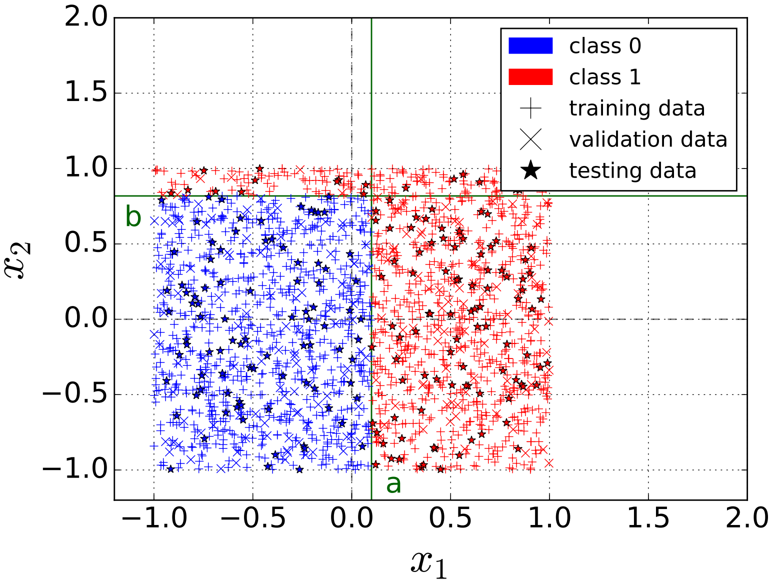
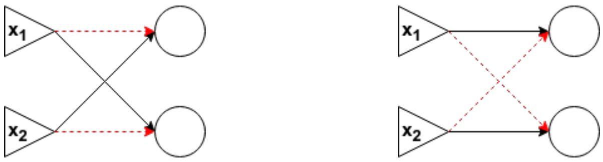


FIGURE 3.5: Dataset with unbalanced feature information.

Hence, we focus on synapses connecting the input and hidden layer. 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 (Unbalanced features 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
transfer fcn	sigmoid	number of epochs	50	retrain	True
		minibatch size	1	retraining epochs	50

TABLE 3.3: Experiment settings for dataset with unbalanced feature information.

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) will always be removed first.

Results: Unbalanced Feature Information

In Fig. 3.7 the first hypothesis is confirmed. The pruning of input-hidden synapses finished with the result shown in Fig. 3.6a in 48 out of 100 observations and with the result in Fig. 3.6b in 44% of the cases.

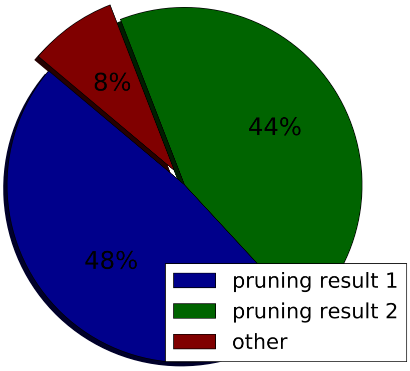


FIGURE 3.7: Results of pruning (see Fig. 3.6) in input-hidden layer (100 observations).

In other words, with a probability of 92% the algorithm is able to find the axes-parallel lines and reveals that each of the lines needs information from one 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. As we can see, the coefficient of synapses's significance 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 coefficients first, therefore information coming from feature x_1 would live longer in the network than x_2 information.

This result can be explained for such simple example. Assuming w_{r1} to be the weight of synapse connecting the x_1 feature and r^{th} hidden neuron (with bias b_r) and w_{s2} the weight of 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.

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

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

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

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

The feature vectors are normalised (described in chapter 2) and in Fig. 3.5 we see that $|a| < |b|$, hence we want:

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

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

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

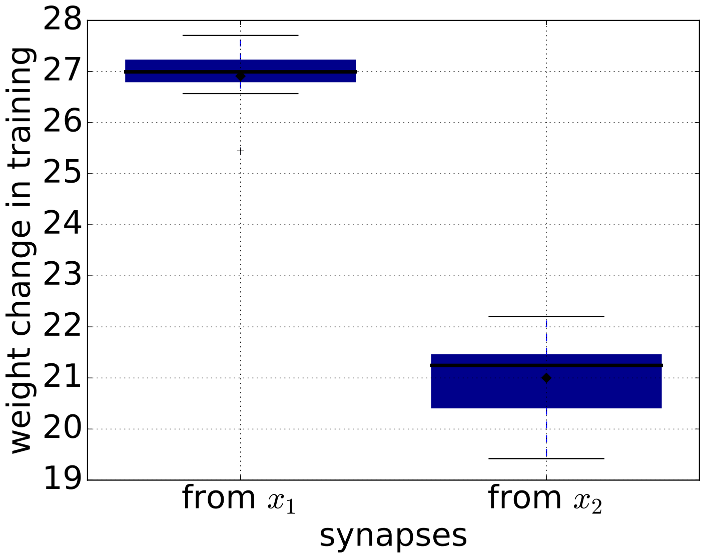


FIGURE 3.8: Weight change in training for the remaining synapses in the input-hidden layer (100 observations).

It seems like the magnitude is a perfect measure. 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 the weight magnitudes are. Therefore small initial weight values does not affect the result significantly and so we can state:

$$|w_{ji}| \approx |w_{ji} - w0_{ji}| \tag{3.7}$$

where $w0_{ji} \in [0, 1]$ is the initial value of weight w_{ji}. 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 The Rule-plus-Exception Problem

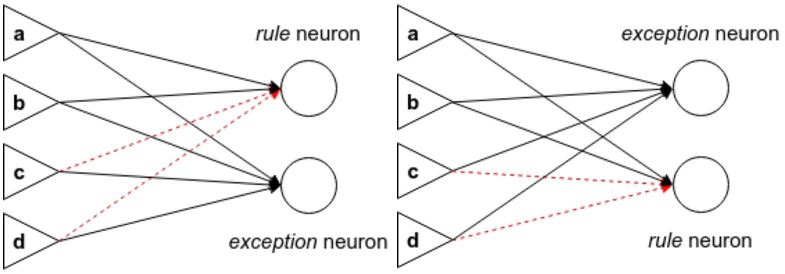
This four-dimensional problem is originally adopted from (Mozer and Smolensky, 1989) and is also used in (Karnin, 1990). The task is to learn another Boolean function: $AB + \overline{ABCD}$. A single function output should be on (i.e. equals 1) when both A and B are on, which is the *rule*, and it should also be on when the *exception* $\overline{ABCD}$ 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 will reveal the part of network, which deals with the *exception*, to be eliminated first - before the part of network dealing with the *rule*.

To test this example, 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 layer again. Two possibilities of expected pruning result 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
transfer fcn	sigmoid	number of epochs	50	retrain	True
		minibatch size	1	retraining epochs	50

TABLE 3.4: Experiment settings for dataset with unbalanced feature information.

Results: The 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.

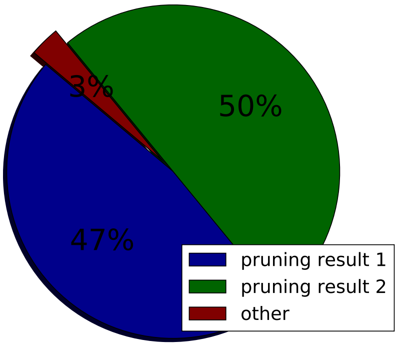


FIGURE 3.10: Results of pruning (see Fig. 3.9) in input-hidden layer (100 observations, RPE problem).

The same thing is confirmed by Fig. 3.11. It shows weight changes in training (coefficients of significance) for all 8 synapses in input-hidden layer. We can see that synapses connecting the *rule* neuron with feature c (s_{rc}) and d (s_{rd}) were eliminated first (resulting in structures in Fig. 3.9).

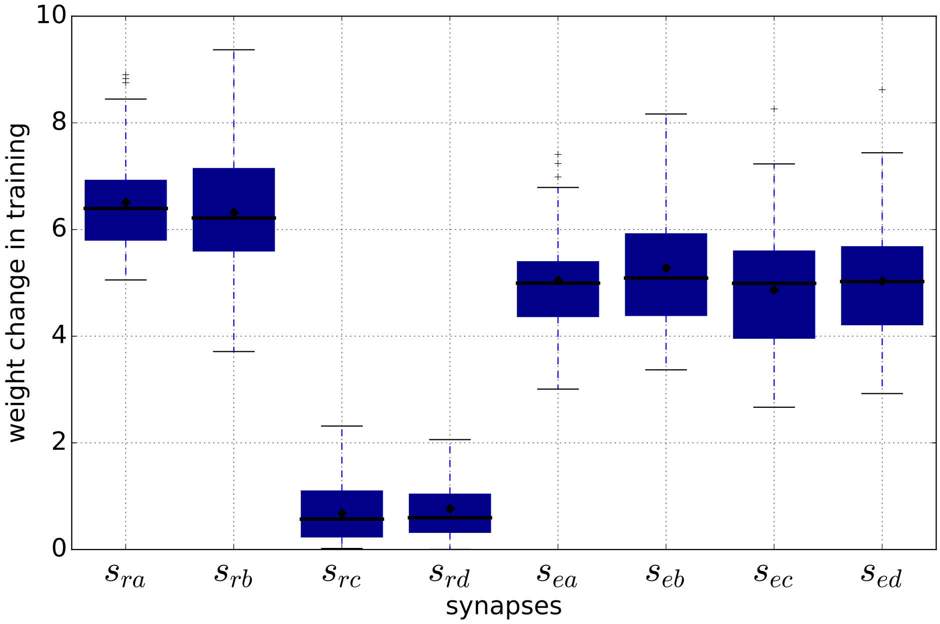


FIGURE 3.11: Weight change in training for synapses in input-hidden layer (100 observations, RPE problem).

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

3.4 Michalski's Train Problem

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.

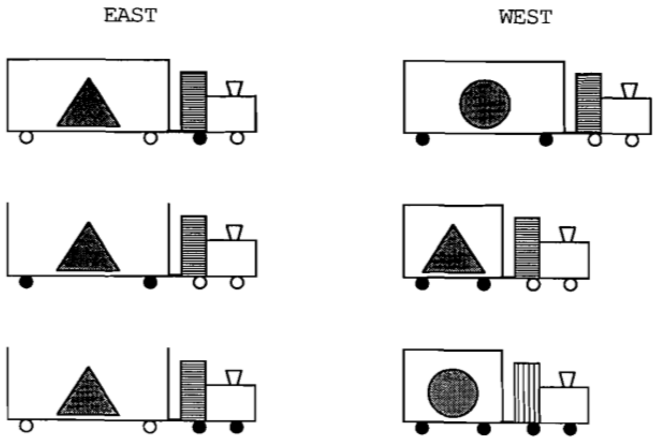


FIGURE 3.12: Michalski's train problem.

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 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 selects 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 used in (Mozer and Smolensky, 1989), one hidden neuron should be enough to learn this problem, hence we started with 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
transfer fcn	sigmoid	number of epochs	100	retrain	True
		minibatch size	1	retraining epochs	10

TABLE 3.7: Experiment settings for the train dataset.

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

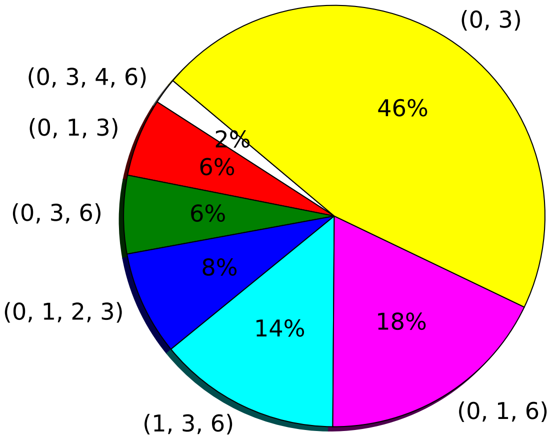


FIGURE 3.13: Results of feature selection by the pruning algorithm (train problem). The labels corresponds with feature indices in Table 3.5.

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 includes the $(0, 3)$ 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.

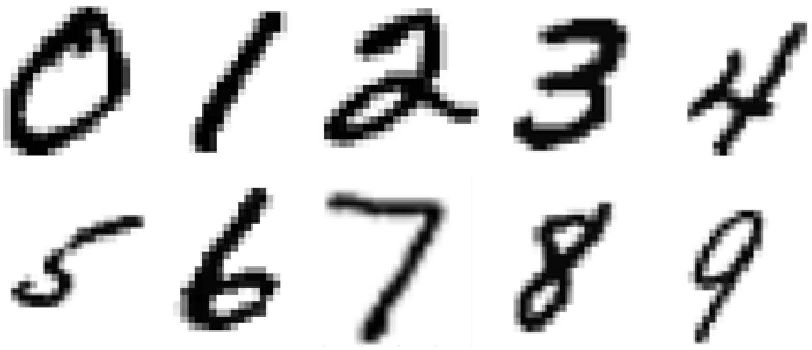


FIGURE 3.14: Examples of MNIST dataset.

Each sample is a grayscale image (normalised to $[0, 1]$) of size $28x28$ pixels. This gives row-by-row a vector of 784 features.

3.6 Phonemes (Speech Data)

PHONES data...

Chapter 4

Discussion

Discussion text...

4.1 Methods Recapitulation

Methods recapitulation text...

4.2 Comparison of Pruning Methods

Comparison of results text...

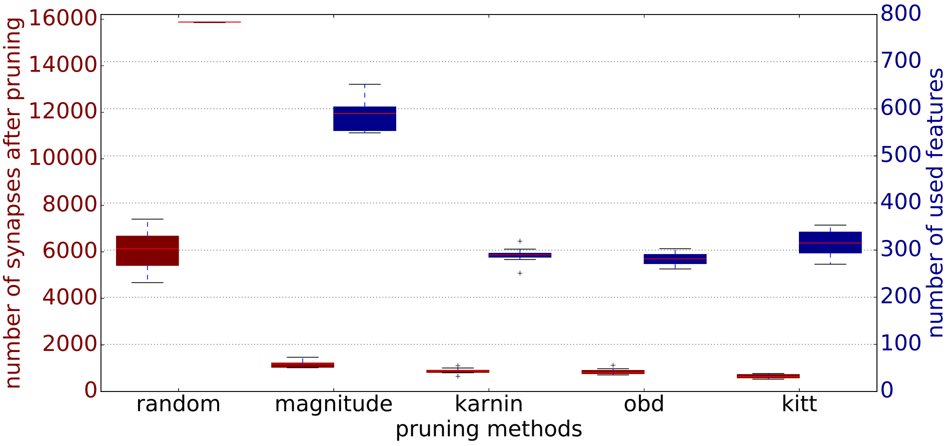


FIGURE 4.1: MNIST, req_acc = 0.95, retraining: 5 epochs

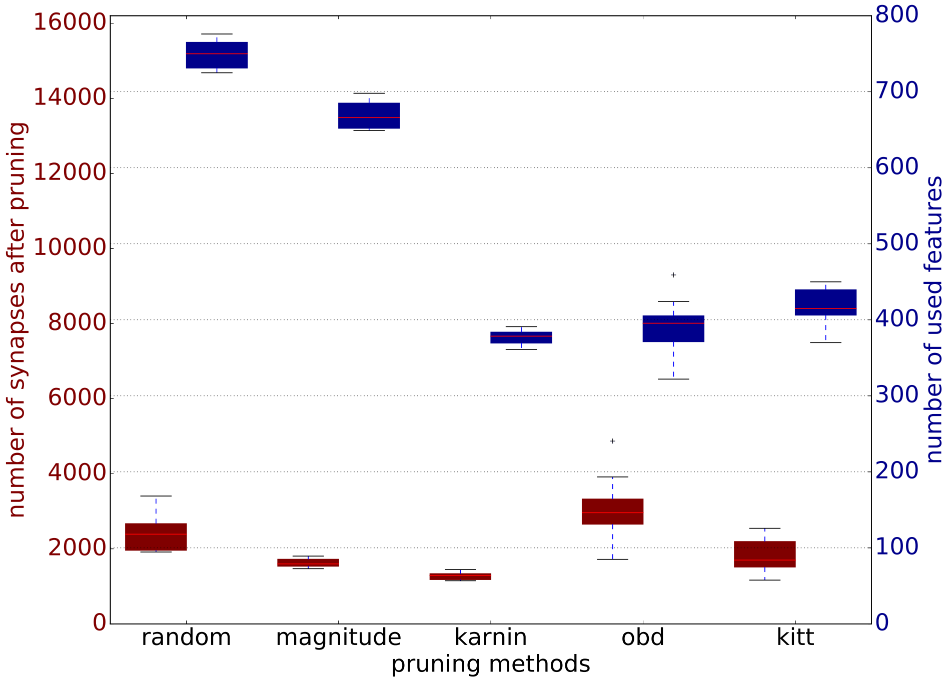


FIGURE 4.2: MNIST, req_acc = 0.95, no retraining

Chapter 5

Conclusion and Outlook

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

Appendix A1

Structure of the Workspace

Appendix A2

Implementation

Appendix A3

Code Documentation