Learning highly non-separable Boolean functions using Constructive Feedforward Neural Network

Marek Grochowski and Włodzisław Duch

Department of Informatics, Nicolaus Copernicus University,
Grudziądzka 5, Toruń, Poland,
grochu@is.umk.pl; Google: Duch

17th International Conference on Artificial Neural Networks

Introduction

Learning of problems with inherent non-separable Boolean logic is still a challenge (text categorization, medical data, natural language processing, bioinformatics, etc.)

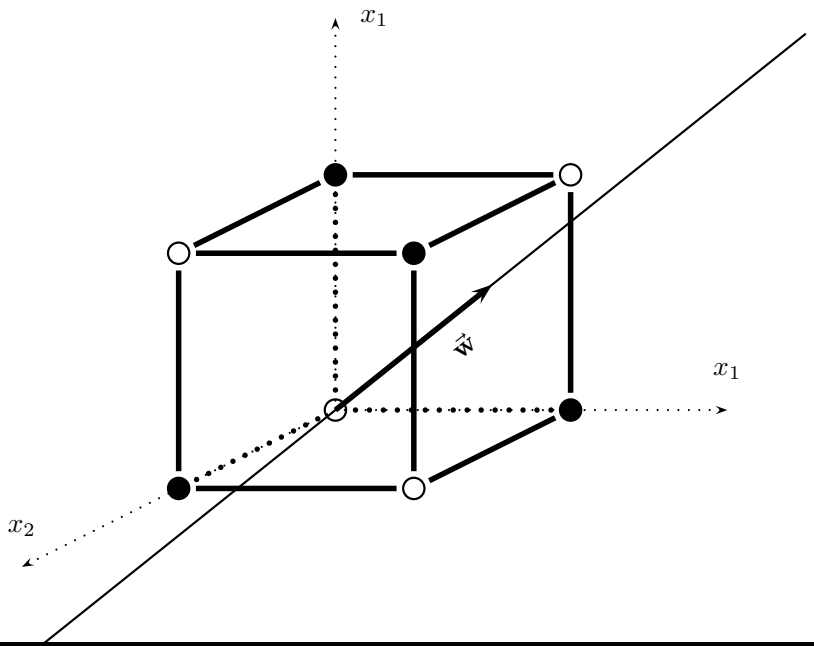
Many learning systems, such as SVMs, decision trees or similarity-based methods cannot deal with such problems - obtained solutions are too complex, generalization is poor

The goal is to find general algorithm capable of solving a large set of problems with the smallest possible number of parameters

N-bit parity problem

Examples of solving n-bit parity problem

- it is trivial with periodic function with a single parameter
- MLP need $O(n^2)$ parameters, learning is difficult
- most of neural network solutions proposed to solve n-bit parity problem will not work for other complex problems (E. Iyoda, H. Nobuhara, K. Hirota (2003); D. Stork, J. Allen (1992), B. Wilamowski, D. Hunter (2003))
- projection on a line with n threshold parameters (k-separable solution), Duch (2006)



Non-separable Boolean functions

The difficulty of learning Boolean functions grows quickly with the minimum k required to solve a given problem.

For many complicated problems often a simple linear mapping exists that leaves only trivial non-linearities that may be separated using window-like neurons.

The main idea is to create constructive network with neurons that realize window-like functions

Parity problem

Example of 10 dimensional parity problem ($2^{10} = 1024$ vectors)

Sequencion of labels of projection on random direction

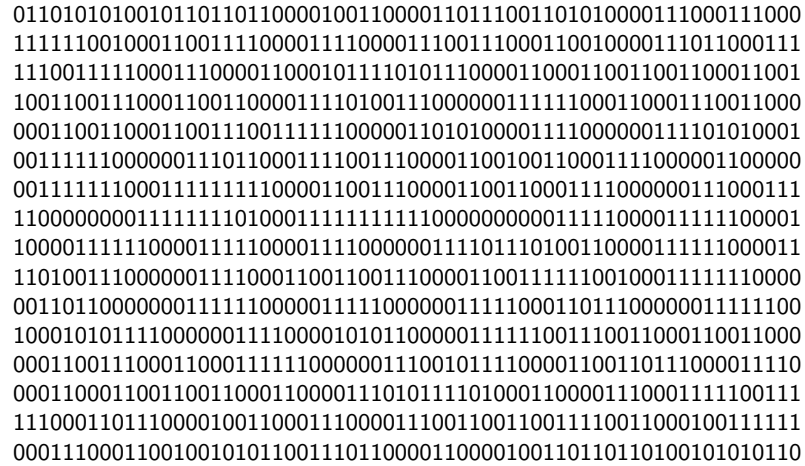
```
01101010100101101101100001001100001101110011010100001110001110000
11111100100011001111000011110000111001110001100100001110110001111
11100111110001110000110001011110101110000110001100110011001100011001
10011001110001100110000111101001110000001111110001100011100110000
00011001100011001110011111100000110101000011110000001111010100001
00111111000000111011000111100111000011001001100011110000011000000
00111111100011111111100001100111000011001100011110000001110001111
11000000001111111101000111111111110000000000111110000111111000001
10000111111000011111000011110000001111011010011000011111100001 1
11010011100000011110001100110011100001100111111001000111111100000
00110110000000011111000001111100000011111000110111000000111111100
10001010111100000011110000101011000001111110011100110001100110000
00011001110001100011111100000011100101111000011001101110000111110
00011000110011001100011000011101011110100011000011100011111100111
11100011011100001001100011100001110011001100111100110001001111111
00011100011001001010110011101100001100001001101101101001010101 10
```

Parity problem

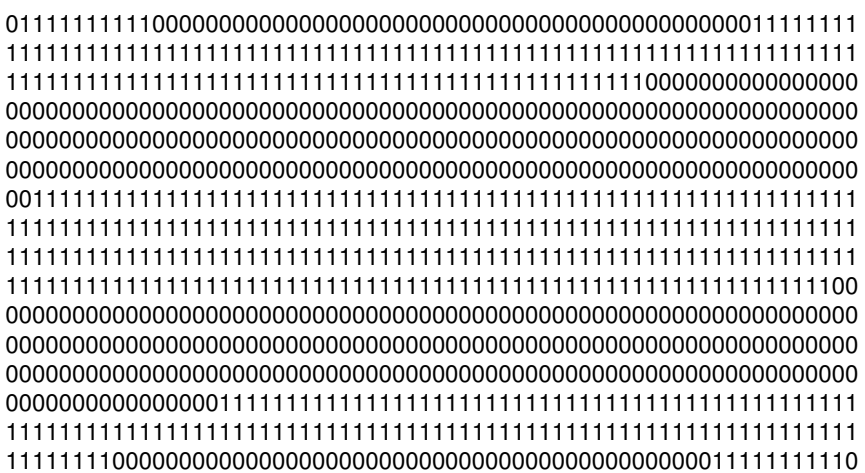
Example of 10 dimensional parity problem ($2^{10} = 1024$ vectors)

Sequencion of labels of projection on

random direction



diagonal direction



weights: 2.29 -2.27 2.29 -2.27 2.29 -2.27 -2.27 -2.27 2.29 -2.27

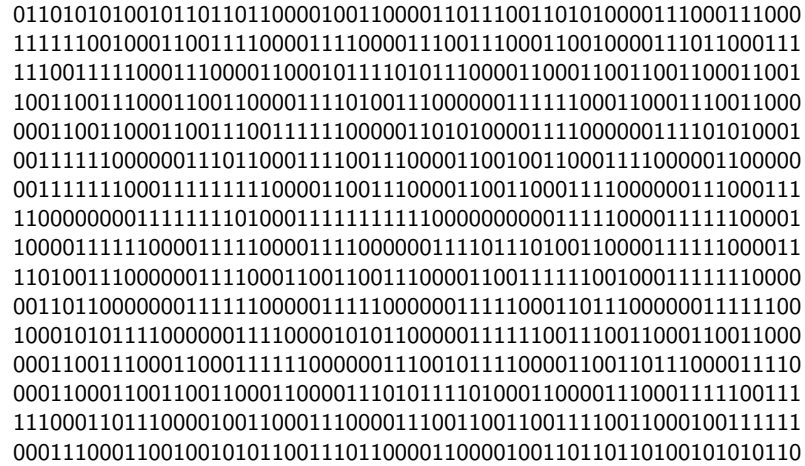
$k = 11$ (11-separability)

Parity problem

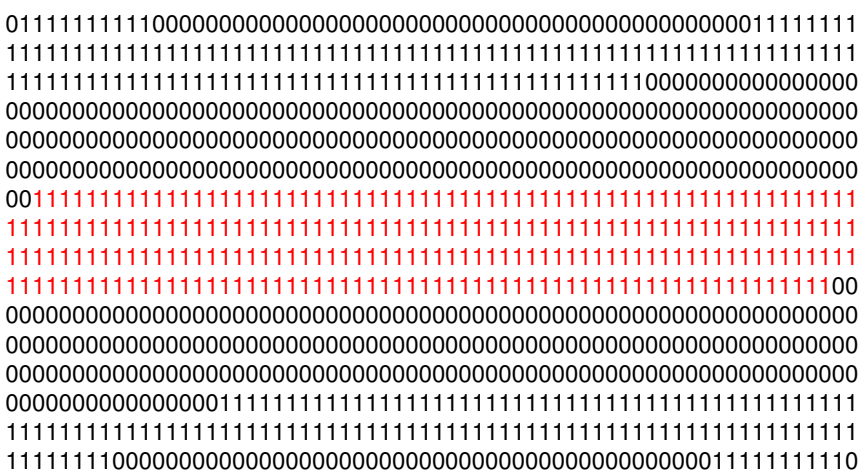
Example of 10 dimensional parity problem ($2^{10} = 1024$ vectors)

Sequencion of labels of projection on

random direction



diagonal direction



weights: 2.29 -2.27 2.29 -2.27 2.29 -2.27 -2.27 -2.27 2.29 -2.27

$k = 11$ (11-separability)

Error Function

Combination of projection and clustering

Hard-window transfer function

$$\tilde{M}_i(\vec{x};\vec{w},a,b) = \begin{cases} 1 & \text{if} \quad \vec{wx} \in [a,b] \\ 0 & \text{if} \quad \vec{wx} \notin [a,b] \end{cases}$$

Error measure

$$E(\vec{x};\Gamma) = E_{\vec{x}}||y(\vec{x};\Gamma) - c(\vec{x}))||$$

does not give any control over purity of clusters

Error function with purity term

$$E(\vec{x};\Gamma;a,b,\lambda) = E_{\vec{x}}||y(\vec{x};\Gamma) - c(\vec{x}))|| + \lambda E_{y\in[a,b]}||y(\vec{x};\Gamma) - c(\vec{x}))||$$

λ controls the tradeoff between the covering and the purity

Transfer Functions

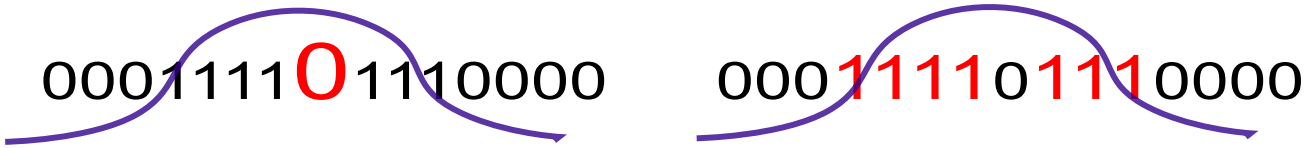
Soft window-like transfer functions

- $M(\vec{x};\vec{w},a,b,\beta) = \sigma(\beta(\vec{w}\vec{x}-a)) - \sigma(\beta(\vec{w}\vec{x}-b))$
- $M(\vec{x};\vec{w},t,a,\beta) = \sigma(\beta(\vec{w}\vec{x}-t-a))(1-\sigma(\beta(\vec{w}\vec{x}-t+a))$
- $M(\vec{x};\vec{w},a,b,\beta) = \frac{1}{2}\left(1-\tanh(\beta(\vec{w}\vec{x}-a))\tanh(\beta(\vec{w}\vec{x}-b))\right)$

slope parameter controls "softness" of transfer function $M(\vec{x};\Gamma_i) \overset{\beta\to\infty}{\longrightarrow} \tilde{M}(\vec{x};\Gamma_i)$

Error Function

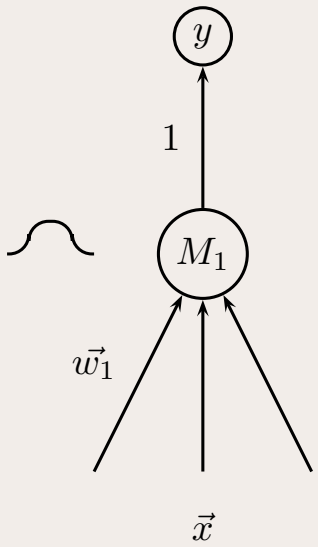
$$E(\vec{x};\Gamma,\lambda_1,\lambda_2) = \tfrac{1}{2}\sum_{\vec{x}}(y(\vec{x};\Gamma)-c(\vec{x}))^2 + \underbrace{+\lambda_1\sum_{\vec{x}}(1-c(\vec{x}))y(\vec{x};\Gamma)}_{\text{penalty}} \underbrace{-\lambda_2\sum_{\vec{x}}c(\vec{x})y(\vec{x};\Gamma)}_{\text{reward}}$$



Penalty factor λ_1 increases the total error for vectors x_i from class $c(x_i) = 0$ that falls into group of vectors from class 1 (it is a penalty for "unclean" clusters).
Reward factor λ_2 decreases the value of total error for every vector x_i from class 1 that was correctly placed inside created clusters (it is a reward for large clusters).

Learning

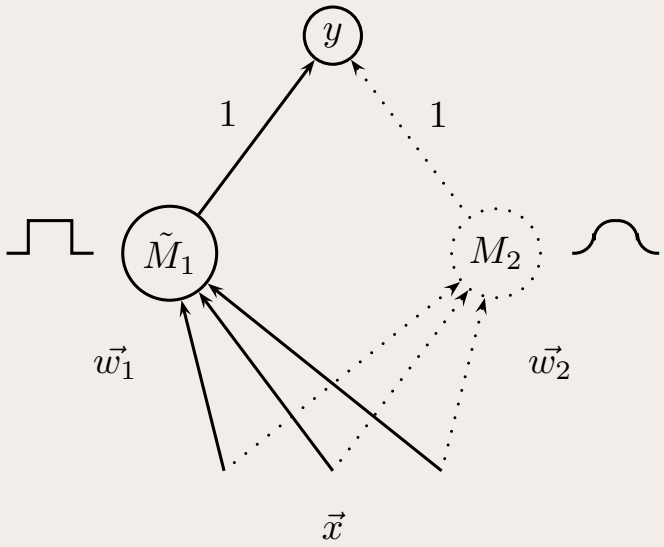
Constructive algorithm



Learning starts with an empty hidden layer and in every phase of the training one new unit is added, initialized and trained using the backpropagation algorithm

Learning

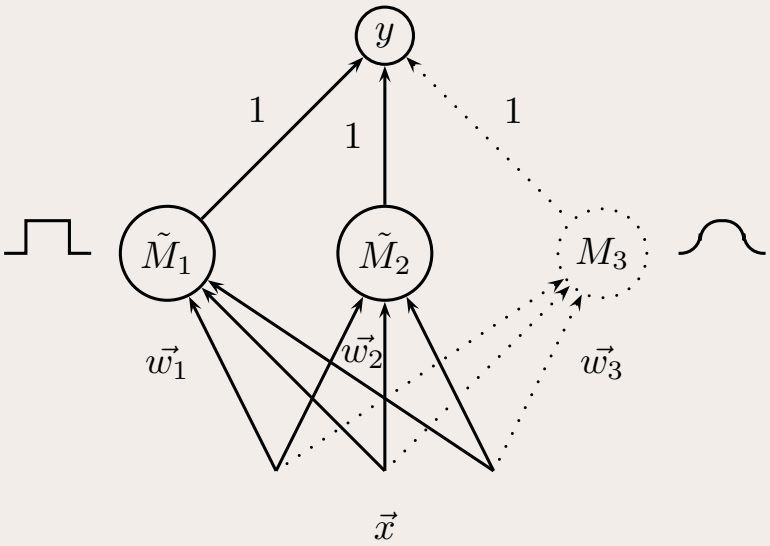
Constructive algorithm



The input vectors correctly handled by the first neuron do not contribute to the error, therefore the weights of this neuron are kept frozen during further learning. Slope β is set to large value to obtain hard boundaries. Next node is added and learning procedure is repeated on the remaining data

Learning

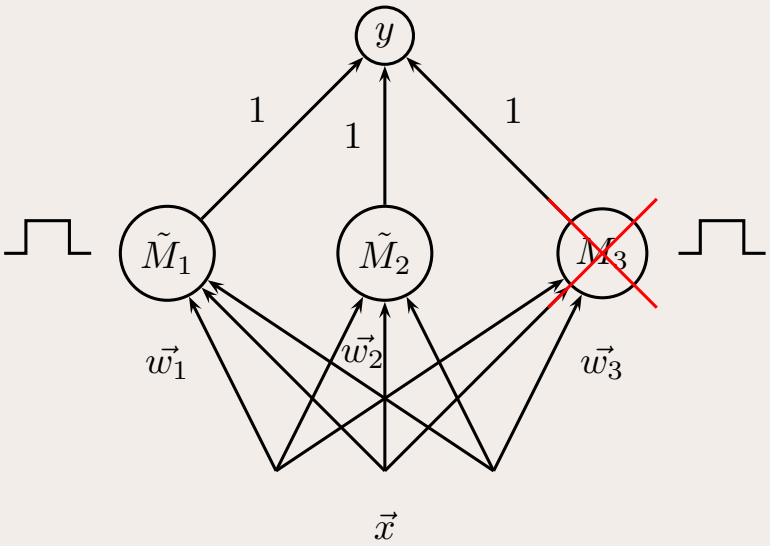
Constructive algorithm



Most nodes and connections are fixed and only weights of one node are modified at each training step

Learning

Constructive algorithm

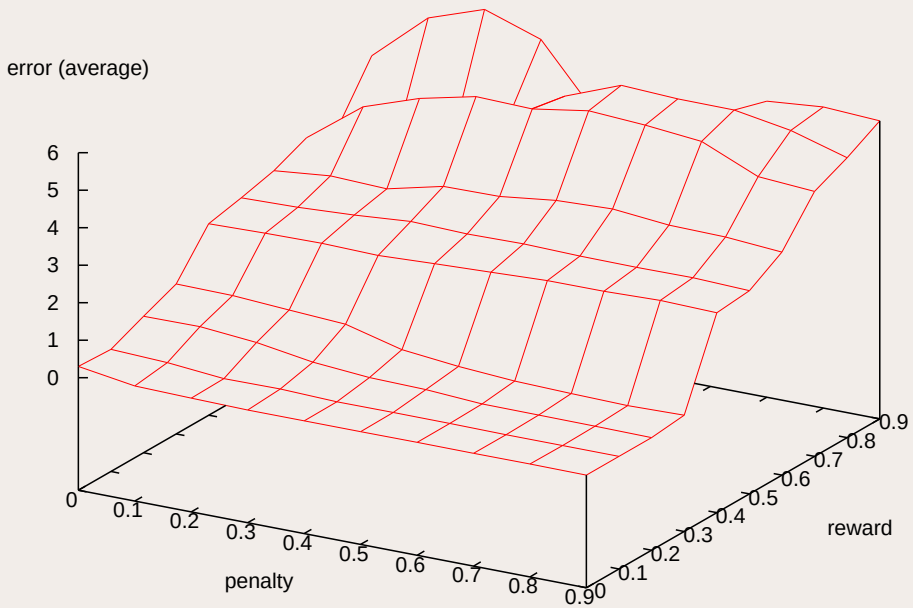


If number of cases correctly classified by a given new node drops below certain minimum the learning procedure stops and this node is removed from the network

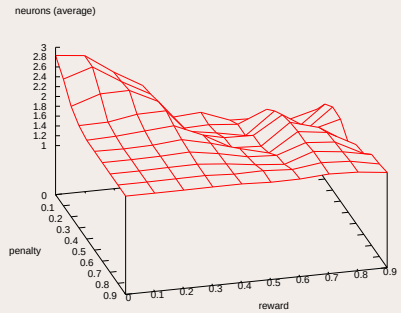
Influence of Penalty and Reward

Average over 192 6-separable 4 dimensional Boolean functions

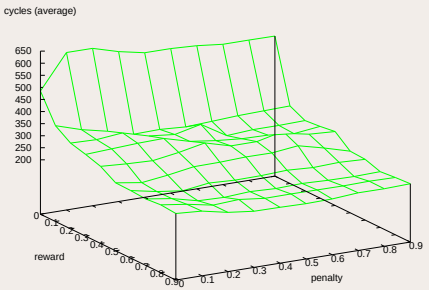
Error



Neurons



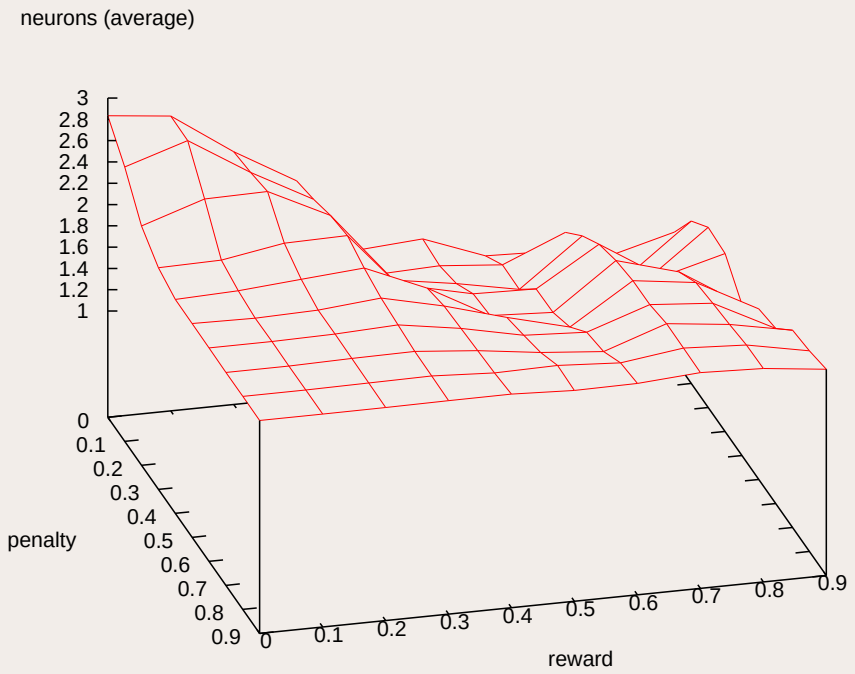
Cycles



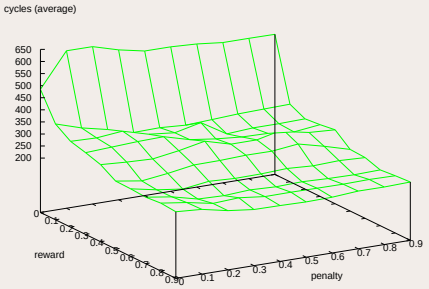
Influence of Penalty and Reward

Average over 192 6-separable 4 dimensional Boolean functions

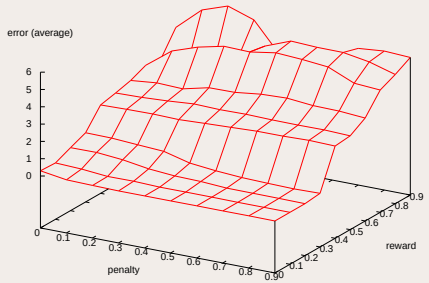
Neurons



Cycles



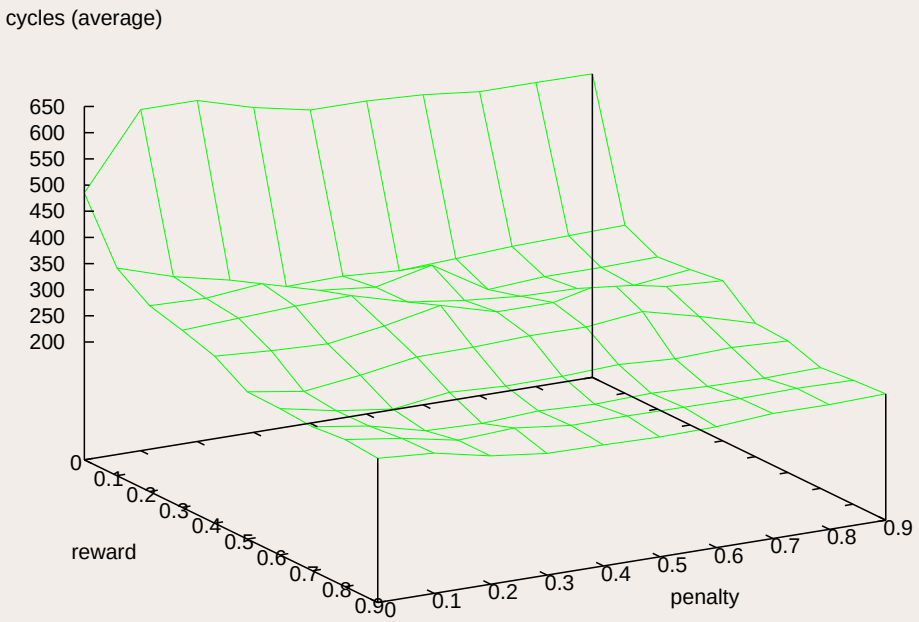
Error



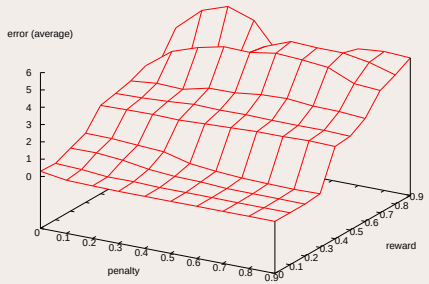
Influence of Penalty and Reward

Average over 192 6-separable 4 dimensional Boolean functions

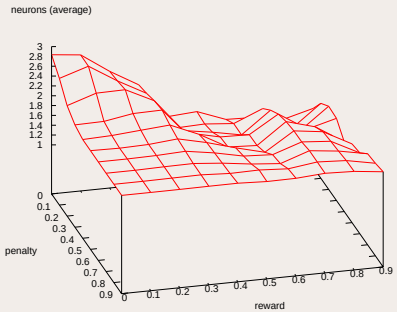
Cycles



Error



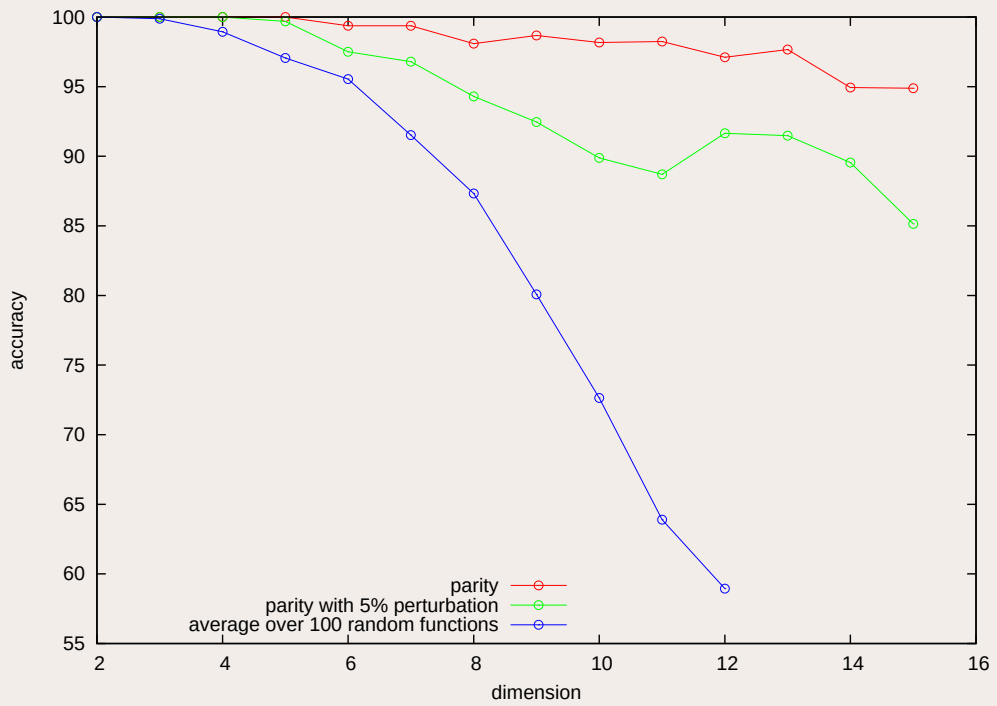
Neurons



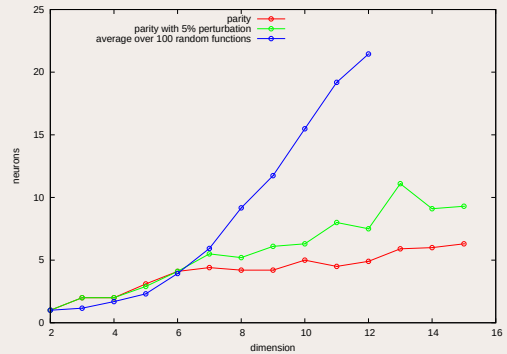
Boolean functions

parity
parity functions with small perturbation of labels
random Boolean functions

Average training accuracy



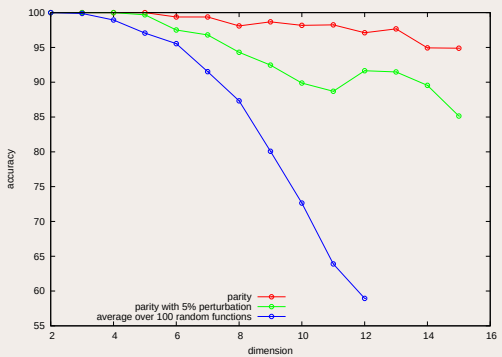
Average no. of neurons



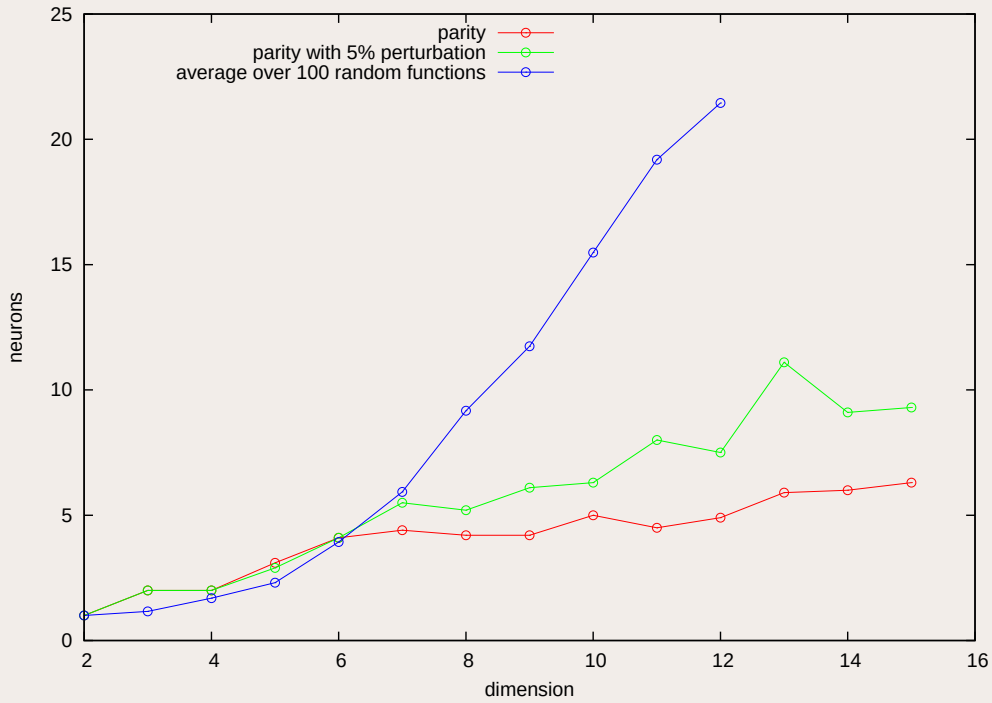
Boolean functions

parity
parity functions with small perturbation of labels
random Boolean functions

Average training accuracy



Average no. of neurons



Real World Data

Test for datasets from UCI repository

10x10 CV test accuracy

dataset	1-NN	Naive Bayes	SVM	c3sep
Appendicitis	81.3±1.5	85.3±1.0	86.5±0.3	85.3±1.0
Australian	78.0±0.2	80.0±0.3	85.0±0.1	86.3±0.6
Flag	50.1±1.1	41.1±1.1	51.1±1.1	53.6±1.8
Glass	68.2±1.7	47.4±1.7	66.7±0.9	61.1±1.3
Ionosphere	85.2±1.2	82.2±0.2	85.2±0.2	85.1±1.5
Iris	95.9±0.5	94.9±0.2	95.5±0.3	95.7±1.0
Pima-diabetes	70.5±0.5	75.2±0.5	70.3±1.0	76.3±0.4
Promoters	78.5±1.8	85.81±1.3	93.1±1.5	74.7±5.6
Sonar	86.8±1.8	67.8±1.2	84.2±1.1	77.9±2.4
Wine	95.1±0.8	98.1±0.3	95.1±0.2	97.1±0.8

Real World Data

Test for datasets from UCI repository

Comparison of complexity SVM vs. c3sep

	support vectors	neurons (total)	average number of neurons per class							
Appendicitis	32.1	4.2	2.2	2.0						
Australian	207.2	8.8	4.5	4.3						
Flag	315.2	26.7	5.1	6.1	4.0	2.1	2.7	3.6	1.1	1.6
Glass	295.8	14.0	1.4	3.9	1.0	2.8	1.9	2.8		
Ionosphere	63.9	7.9	3.0	4.9						
Iris	43.4	5.0	1.0	2.0	2.0					
Pima-diabetes	365.3	9.1	4.2	4.8						
Promotores	77.2	3.7	1.9	1.8						
Sonar	109.7	8.5	4.5	4.0						
Wine	63.3	4.0	1.0	2.0	1.0					

Summary and Further Work

Some conclusions

- First steps towards efficient learning of Boolean functions have been made here
- The approach presented here has been able to learn quite difficult Boolean functions using a very simple model
- Great advantage is small computational costs of the constructive network training

Further work

- Global minimization algorithms instead of backpropagation
- looking for k-separable solution -e.g. network with weight shering
- different error function
- additional transformation of input data
- Experiments with high dimensional real world problems
- Many more ideas ...