

WARSAW UNIVERSITY OF TECHNOLOGY
FACULTY OF MATHEMATICS AND INFORMATION SCIENCE


Neural Networks

Lecture 7

1

Summary of modelling of neural networks

2

Structures

3

Network structure

Main problems faced building a *feedforward* network (without feedback loops).

- how many layers is necessary for proper network's work?
- how many elements have to be in these layers?

4

Network structure

What is a network layer?

A layer - it is the part of network structure which contains active elements performing some operation. Usually the first layer elements (nodes) are passive and they propagate the signal through the network only.

5

Network structure

input layer first hidden layer last hidden layer output layer

6

Network structure

Linear network?

Linear network it is a network where input signals are multiplied by the weights, added, and next the result is send to the axon as the output signal of the neuron. Eventually some threshold can be used.

Typical examples of a linear network are a *simple perceptron* and an *Adeline network*.

7

Network structure

Or Nonlinear Network?

In a nonlinear network the output signal is calculated by a nonlinear function $f()$. The function $f()$ is called neuron *transfer function* and its operations have to be similar to the operations of a biological neuron.

Typical example of a nonlinear network is a *sigmoidal network*.

8

Network structure

Radial networks

For special purposes sometimes we are using the radial neurons i.e. neurons with *Radial Basis Function RBF*. They have non typical aggregation methods of input data, they uses non typical transfer function (Gauss function) and they learned by the special way.

Data aggregation consist on the calculation of a distance between an input signal (vector X), and established in a learning process centroid of a certain set T .

9

Network structure

Sigmoid and radial neuron

Sigmoidal neuron represents in the multidimensional space hyperplane separating space into two categories (Fig.A). Radial neuron represents hypersphere performing circular separation around the central point (Fig.B).

10

Network structure

Sigmoid and radial neuron

Sigmoid neuron builds its classification from hyperplane, defined by the weighted sum of input signal which is argument to nonlinear function (Fig.A), whereas the radial basis approach uses hyperellipsoids to partition the pattern space. (Fig.B).

11

Radial basis function network

Radial basis function network (RBF) uses radial basis functions as activation functions typically have three layers: an input layer, a hidden layer with a non-linear **RBF** activation function and a linear output layer.

Functions that depend only on the distance from a center vector are radially symmetric about that vector, hence the name *radial basis function*. In the basic form all inputs are connected to each hidden neuron. The norm is typically taken to be the Euclidean distance and the radial basis function is commonly taken to be Gaussian.

12

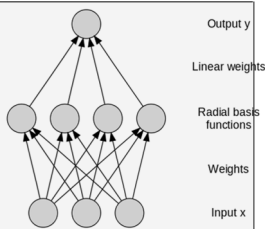
Radial basis function network

The output of the network is then a scalar function of the input vector, and is given by

$$\varphi(x) = \sum_{i=1}^N a_i \rho(\|x - c_i\|)$$

where N is the number of neurons in the hidden layer, c_i is the center vector for neuron i , and a_i is the weight of neuron i in the linear output neuron. Functions that depend only on the distance from a center vector are radially symmetric about that vector. The radial basis function is commonly taken to be Gaussian.

$$\rho(\|x - c_i\|) = \exp[-\beta \|x - c_i\|^2]$$



13

Network structure

How many layers?

The simplest *feedforward* network has at least two layers – an input and an output (nb. such a networks are called single layer networks – active neurons are located only in an output layer).

Usually between these layers there are multiple intermediate or hidden layers.

Hidden layers are very important they are considered to be categorizers or feature detectors. The output layer is considered a collector of the features detected and producer of the response

14

Network structure

The Input Layer

With respect to the number of neurons comprising this layer, this parameter is completely and uniquely determined once you know the shape of your training data. Specifically, *the number of neurons comprising that layer is equal to the number of features (columns) in your data*. Some neural networks configurations add one additional node for a bias term.

In addition very often the input data are *scaled* to be in the range of [0;1] or [-1;+1].

15

Network structure

The Output Layer

Like the input layer, every neural network has exactly one output layer. Determining its size (number of neurons) is simple; it is completely determined by the chosen model configuration.

The interesting solution is called „one out of N “. Unfortunately usually because of limited accuracy in network operation the non-zero signal can occur on each out elements. It is necessary to implement the special criteria for results *post-processing* and threshold of acceptance and rejection.

Often the number of network inputs is greater than the problem to be solved. For example it can be a result of splitting the output signal into some sub-ranges.

16

Network structure

How to built the network?

Too small network without hidden layer or to few neurons. Such a network is unable to solve a problem and even the very long learning time will not help.

Too big network will cheat a teacher. Too many hidden layers or too many elements in the hidden layers yields to the simplification of task. The network will learn whole set of the learning patterns. It learns very a fast and precisely but is completely useless for solving any similar problem.

* source: R.Tadeusiewicz. Odkrywanie właściwości sieci neuronowych, PAU Kraków 2007

17

Network structure

How many hidden layers?

Too many hidden layer yields to significant deterioration of learning. There is a consensus is the performance difference from adding additional hidden layers: the situations in which performance improves with a second (or third, etc.) hidden layer are very small. One hidden layer is sufficient for the large majority of problems.

Additional layer yields to an instability of gradient, and increase of the number of false minima.

Two hidden layer are necessary only if the learning refers the function with points of discontinuity.

* source: R.Tadeusiewicz. Odkrywanie właściwości sieci neuronowych, PAU Kraków 2007

18

Network structure

Size of the hidden layer - how many neurons?

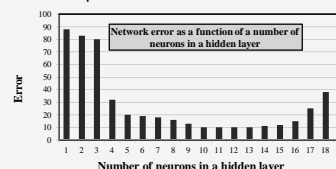
Too many neurons in the hidden layers may result in overfitting. Overfitting occurs when the neural network has so much information processing capacity that the limited amount of information contained in the training set is not enough to train all of the neurons in the hidden layers. A second problem can occur even when the training data is sufficient. An inordinately large number of neurons in the hidden layers can increase the time it takes to train the network.

* source: R.Tadeusiewicz *Odkrywanie właściwości sieci neuronowych*, PAU Kraków 2007 19

Network structure

Size of the hidden layer - how many neurons?

Using **too few neurons** in the hidden layers will result in something called underfitting. Underfitting occurs when there are too few neurons in the hidden layers to adequately detect the signals in a complicated data set.



* source: R.Tadeusiewicz *Odkrywanie właściwości sieci neuronowych*, PAU Kraków 2007 20

Network structure

Rough prerequisite for the number of hidden neurons (for most of typical problems) is the *rule of a geometric pyramid*. The number of neurons in a consecutive layers has a shape of a pyramid and decrease from the direction of input to the output. The numbers of neurons in a consecutive layers are forming geometric sequence.

* source: T.Masters *Practical Neural Networks*, Acad. Press, 1993. 21

Network structure

For example, for the network with one hidden layer, with n – neurons in the input layer and m – neurons in the output layer, in the hidden layer should be

$$\text{NHN} = \sqrt{n * m}.$$

For the network with two hidden layers

$$\text{NHN}_1 = m * r^2, \text{NHN}_2 = m * r \quad \text{where } r = \sqrt[3]{\frac{n}{m}}$$

Att: not for associative networks (numbers of input and output neurons are identical)

* source: T.Masters *Practical Neural Networks*, Acad. Press, 1993. 22

Network structure

For the network with one hidden layer, with n – elements in the input layer, k – elements in the hidden layer, the maximal number of separated classes M (equivalent to the number of output elements)

$$M = \sum_{i=1}^n \binom{k}{i} \quad \text{where for } k < i \quad \binom{k}{i} = 0$$

for $n \gg k$ we have $M = 2^k$.

* source: T.Masters *Practical Neural Networks*, Acad. Press, 1993. 23

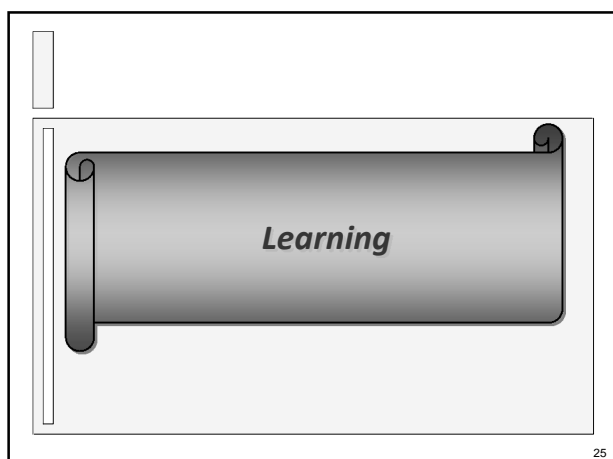
Network structure

It is possible to evaluate the number of elements in the hidden layer for $k < n$

$$k = \log_2 M$$

The number of hidden neurons should be between the size of the input layer and the size of the output layer or the number of neurons in that layer is the mean of the neurons in the input and output layers.

* source: T.Masters *Practical Neural Networks*, Acad. Press, 1993. 24




25

Net learning

One of the key elements of a neural network is its **ability to learn**. A neural network is not just a complex system, but a complex nonlinear adaptive system, meaning it can change its internal structure based on the information flowing through it.

Multi-layer networks use a variety of learning techniques.

A common method for measuring the discrepancy between the expected output and the actual output is using the squared error measure.



26

Net learning

There are three major learning paradigms:

Supervised Learning

The learning algorithm would fall under this category if the desired output for the network is also provided with the input while training the network. By providing the neural network with both an input and output pair it is possible to calculate an error based on its target output and actual output. It can then use that error to make corrections to the network by updating its weights.

27

Net learning

Unsupervised Learning

In this paradigm the neural network is only given a set of inputs and it's the neural network's responsibility to find some kind of pattern within the inputs provided without any external aid. This type of learning paradigm is often used in data mining and is also used by many recommendation algorithms due to their ability to predict a user's preferences based on the preferences of other similar users it has grouped together.

28

Net learning

Reinforcement Learning

Reinforcement learning is similar to supervised learning in that some feedback is given, however instead of providing a target output a reward is given based on how well the system performed. The aim of reinforcement learning is to maximize the reward the system receives through trial-and-error. This paradigm relates strongly with how learning works in nature, for example an animal might remember the actions it's previously taken which helped it to find food (the reward).

29

Net learning - summary

Typical learning algorithms:

- Backpropagation**, an abbreviation for "backward propagation of errors", is a common method of training artificial neural networks employing gradient descent method. The method calculates the gradient of a error function with respects to all the weights in the network. The gradient is fed to the optimization method which in turn uses it to update the weights, in an attempt to minimize the loss function.
Backpropagation requires a known, desired output for each input value in order to calculate the error function gradient. It is therefore usually considered to be a supervised learning method. The most important problem is that, under some circumstances, local minima appear in the error function which would not be there if the step function had been used.

30

Net learning - summary

2. To speed up the process of learning, the **momentum term** is added to the weight update rule. Introduction of the momentum rate allows the attenuation of oscillations in the gradient descent. Introduction of the momentum rate allows the attenuation of oscillations in the gradient descent. The momentum rate is used to prevent the system from converging to a local minimum or saddle point.

General idea:

$$W_i = W_i - \alpha \Delta W_i + \mu \Delta W_{i-1}$$

where α is the learning rate, and μ is the momentum term.

31

Net learning - summary

3. **Quasi-Newton** methods require only the gradient (like steepest descent) of the objective to be computed at each iterate. By successive measurements of the gradient, Quasi-Newton methods build a quadratic model of the objective function which is sufficiently good that superlinear convergence is achieved. Quasi-Newton methods are much faster than steepest descent (and coordinate descent) methods. Since second derivatives (the Hessian) are not required, quasi-Newton methods are sometimes more efficient.

32

Net learning - summary

4. **The conjugate gradient** method is one of the most popular and well known iterative techniques for learning networks with many connections. It is slower than Quasi-Newton methods but much faster than backpropagation algorithm. The computational efficiency is enhanced by adaptively modifying initial search direction by calculation of the gradient descent of error with respect to the weights and the determination of a new search direction.

33

Net learning - summary

5. Many modifications of backpropagation method have been proposed to increase the convergence rate of the standard algorithm, and **Quickprop** is one the most popular fast learning algorithms. The convergence rate of Quickprop is very fast; however, it is easily trapped into a local minimum and thus it cannot converge to the global minimum. Quickprop is also one of the better training algorithms and is loosely based on Newton's method.

34

Net learning - summary

6. The **RPROP (Resilient backPROPagation)** algorithm takes a very different approach to improving backpropagation as compared to Quickprop. Instead of making more use of gradient information for better weight updates, RPROP only uses the sign of the gradient, because its size can be a poor and noisy estimator of required weight updates. Furthermore, RPROP assumes that different weights need different step sizes for updates, which vary throughout the learning process.

35

Net learning - summary

7. The **Levenberg - Marquardt** algorithm is an iterative technique that locates the minimum of a multivariate function. It has become a standard technique for non-linear least-squares problems. It can be thought of as a combination of steepest descent and the Gauss-Newton method. When the current solution is far from the correct one, the algorithm behaves like a steepest descent method: slow, but guaranteed to converge. When the current solution is close to the correct solution, it becomes a Gauss-Newton method.

36

Net learning - summary

Supervised networks

Feedforward networks

1. Linear

- Hebb's model (*Hebb 1949, Fausett 1994*)
- Perceptron (*Rosenblatt 1969, Minsky, Pappert 1969/1988, Fausett 1994*)
- Adaline (*Widrow, Hoff 1960, Fausett 1994*)
- Higher order network (*Bishop 1995*)
- Functional link network (*Pao 1989*)

In parenthesis: authors' names and dates of invention/description

37

Net learning - summary

Supervised networks

Feedforward networks

2. MLP Multi-Layered Perceptron (*Bishop 1995, Fausett 1994*)
 - Back Propagation Network (*Rumelhart, Hinton, Williams 1986*)
 - Cascade Correlation Network (*Fausett 1994*)
 - Quickprop (*Fehlman 1989*)
 - RPROP Resilient Back PROPagation (*Riedmiller, Braun 1993*)

38

Net learning - summary

Supervised networks

Feedforward networks

3. Classification only networks

- LVQ Learning Vector Quantization (*Kohonen 1988, Fausett 1994*)
- PNN Probabilistic Neural Network (*Specht 1990, Masters 1993, Hand 1991, Fausett 1994*)

Recurrent Networks (*Hertz, Krogh, Palmer 1991*)

- BAM Bidirectional Associative Memory (*Kosko 1992, Fausett 1994*)
- Boltzman Machine (*Ackley 1985, Fausett 1994*)

39

Net learning - summary

Supervised networks

Competition Networks

- ARTMAP Adaptive Resonans Network (*Carpenter, Grossberg 1991*)
- CP Counterpropagation (*Hecht-Nielsen 1987, 1988, 1990, Fausett 1994*)
- Neocognitron (*Fukushima, Miyake, Ito 1983, Fukushima 1988, Fausett 1994*)

40

Net learning - summary

Unsupervised networks

Competition Networks

1. VQ Vector Quantization

- Grossberg's Network (*Grossberg 1976*)
- Kohonen's Network (*Kohonen 1984*)

2. SOM Self-Organizing Map (*Kohonen 1995, Fausett 1994*)

41

Net learning - summary

Unsupervised networks

Competition Networks

3. ART Adaptive Resonance Theory

- ART 1 (*Carpenter, Grossberg 1987, Moore 1988, Fausett 1994*)
- ART. 2 (*Carpenter, Grossberg 1987, Fausett 1994*)
- ART. 2A (*Carpenter, Grossberg, Rosen 1991*)
- Fuzzy ART (*Carpenter, Grossberg, Rosen 1991*)

42

Net learning - summary

Unsupervised networks

Autoassociative Networks

- Linear Autoassociative Network (*Anderson 1977, Fausett 1994*)
- Hopfield's (*Hopfield 1982, Fausett 1994*)

43

Literature

44

Net learning - summary

45

Summary of Backpropagation

46

Summary of backpropagation

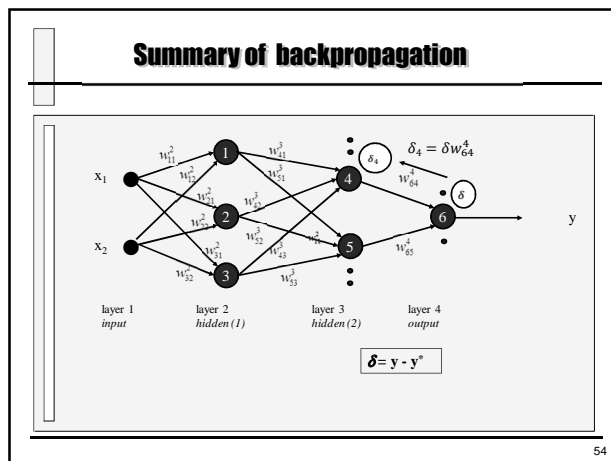
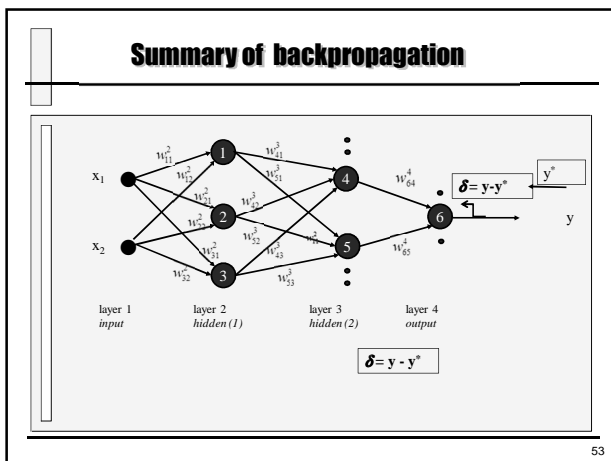
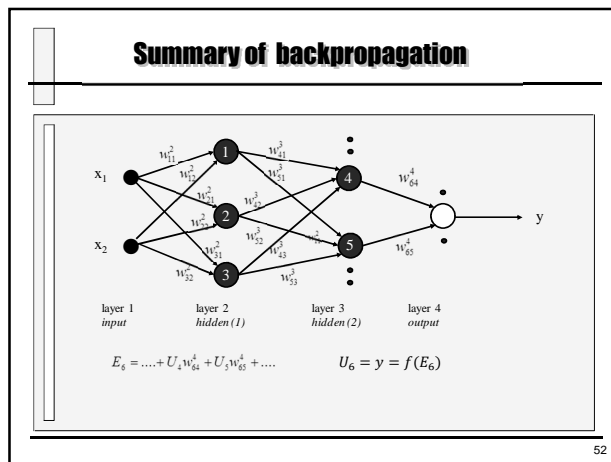
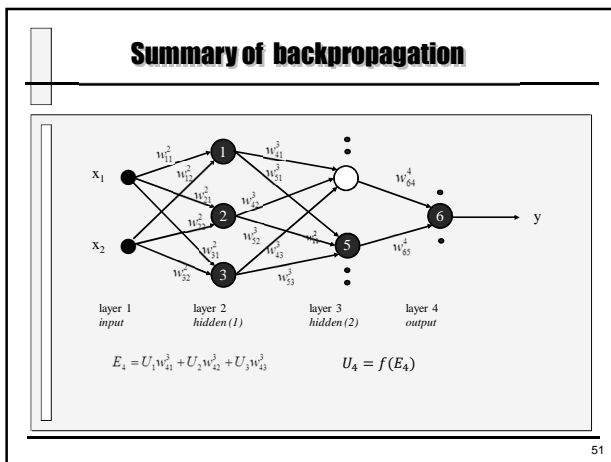
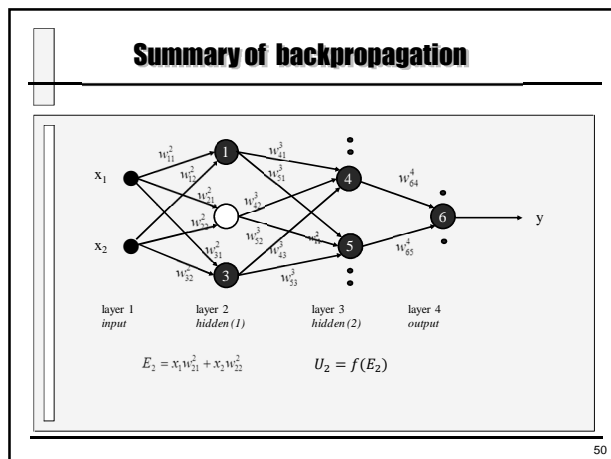
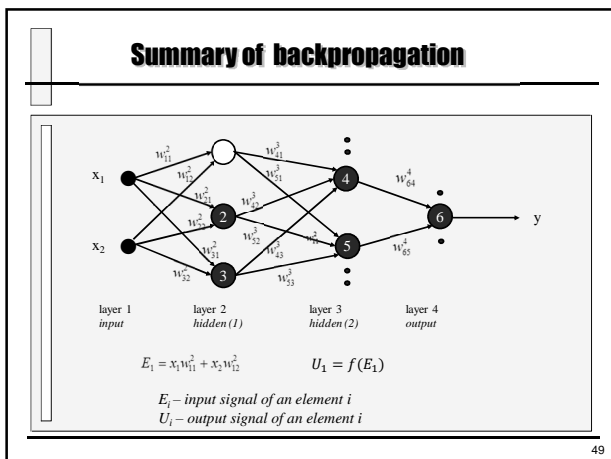
Algorithm (short version)

1. The network is initialized with small random values.
2. The input signal is presented at the net input.
3. The set the output values from the network is calculated.
4. The set the output from the network is compared to the desired output and the error is calculated.
5. The errors of hidden elements are calculated. To calculate the errors in the layer (s) the knowledge of the errors in the next layer (s+1) is necessary.
6. Weights are adjusted (modification).
7. Return to the point 2.

47

Summary of backpropagation

48



Summary of backpropagation

$\delta_1 = \dots + \delta_4 w_{41}^3 + \delta_5 w_{51}^3 + \dots$

layer 1 input layer 2 hidden(1) layer 3 hidden(2) layer 4 output

55

Summary of backpropagation

$$w_{11}^2 = w_{11}^2 + \eta \delta_1 \frac{du_1}{dE_1} x_1 = w_{11}^2 + \eta \delta_1 \frac{\partial u_1}{\partial w_{11}^2} x_1$$

$$w_{12}^2 = w_{12}^2 + \eta \delta_1 \frac{du_1}{dE_1} x_2 = w_{12}^2 + \eta \delta_1 \frac{\partial u_1}{\partial w_{12}^2} x_2$$

56

Summary of backpropagation

$$w_{21}^2 = w_{21}^2 + \eta \delta_2 \frac{du_2}{dE_2} x_1 = w_{21}^2 + \eta \delta_2 \frac{\partial u_2}{\partial w_{21}^2} x_1$$

$$w_{22}^2 = w_{22}^2 + \eta \delta_2 \frac{du_2}{dE_2} x_2 = w_{22}^2 + \eta \delta_2 \frac{\partial u_2}{\partial w_{22}^2} x_2$$

57

Summary of backpropagation

$$w_{41}^3 = w_{41}^3 + \eta \delta_4 \frac{du_4}{dE_4} U_1 = w_{41}^3 + \eta \delta_4 \frac{du_4}{dE_4} f(E_1) = w_{41}^3 + \eta \delta_4 \frac{du_4}{dE_4} f(x_1 w_{11}^2 + x_2 w_{12}^2)$$

$$w_{42}^3 = w_{42}^3 + \eta \delta_4 \frac{du_4}{dE_4} U_2 = w_{42}^3 + \eta \delta_4 \frac{du_4}{dE_4} f(E_2) = w_{42}^3 + \eta \delta_4 \frac{du_4}{dE_4} f(x_1 w_{21}^2 + x_2 w_{22}^2)$$

$$w_{43}^3 = w_{43}^3 + \eta \delta_4 \frac{du_4}{dE_4} U_3 = w_{43}^3 + \eta \delta_4 \frac{du_4}{dE_4} f(E_3) = w_{43}^3 + \eta \delta_4 \frac{du_4}{dE_4} f(x_1 w_{31}^2 + x_2 w_{32}^2)$$

58

Summary of backpropagation

$$w_{64}^4 = w_{64}^4 + \eta \delta_6 \frac{du_6}{dE_6} U_4 = w_{64}^4 + \eta \delta_6 \frac{du_6}{dE_6} f(E_4) = w_{64}^4 + \eta \delta_6 \frac{du_6}{dE_6} f(U_1 w_{41}^3 + U_2 w_{42}^3 + U_3 w_{43}^3)$$

$$w_{65}^4 = w_{65}^4 + \eta \delta_6 \frac{du_6}{dE_6} U_5 = w_{65}^4 + \eta \delta_6 \frac{du_6}{dE_6} f(E_5) = w_{65}^4 + \eta \delta_6 \frac{du_6}{dE_6} f(U_1 w_{51}^3 + U_2 w_{52}^3 + U_3 w_{53}^3)$$

59

Vector Notation of Neural Networks

Vector Notation of Neural Networks

60

Vector notation of neural network

This diagram illustrates how a multilayer neural network can be represented in a vector notation.

61

Vector notation of neural network

This diagram illustrates how a 2-D neural network can be represented in a vector notation.

62

Model mózgu

CCortex™ to program komputerowy opracowany przez Artificial Development (AD). Podstawowym zadaniem jest symulacja działania ludzkiego mózgu, a dokładniej kory mózgowej i jest współdziałania z obwodowym systemem nerwowym. Program symuluje działania 20 miliardów (10^9) neuronów i 20 bilionów (10^{12}) połączeń neuronowych.

63

Model mózgu

W roku 2013 naukowcy z Japonii i Niemiec wykorzystując superkomputer **K computer** (Fujitsu) w Instytucie Riken'a zasymulowali działanie jednej sekundy mózgu człowieka.

Użyli do tego 82,000 procesorów (K computer ma 705,024 procesorów i pamięć RAM 1,4 milion GB,

Naukowcy odtworzyli 1,73 miliarda komórek, z 10,4 bilionem synaps. Symulacja zajęła 40 minut czasu rzeczywistego symulując działanie jednej sekundy czasu „biologicznego”.

64