Lecture 10

Introduction to neural networks

Information Systems (Machine Learning)

Andrey Filchenkov

21.11.2017

Lecture plan

- History of neural networks
- Single layer neural network
- Completeness problem of neural networks
- Multilayer neural networks
- Backpropagation
- Heuristics for neural networks
- Modern neural networks

- The presentation is prepared with materials of the K.V. Vorontsov's course "Machine Leaning".

Lecture plan

- History of neural networks
- Completeness problem of neural networks
- Multilayer neural networks
- Backpropagation
- Heuristics for neural networks
- Modern neural networks

Early history

1943 Artificial neuron by McCulloch and Pitts

1949 Neuron learning rule by Hebb

1957 Perceptron by Rosenblatt

1960 Perceptron learning rule by Widrow and Hoff

1969 “Perceptrons” by Minski and Papert

1974 Back propagation algorithm by Webros and by Galushkin

Modern history

1980 Convolutional NN by Fukushima

1982 Recurrent NN by Hopfield

1991 “Vanishing gradient problem” was identified by Hochreiter

1997 Long short term memory network by Hochreiter and Schmidhuber

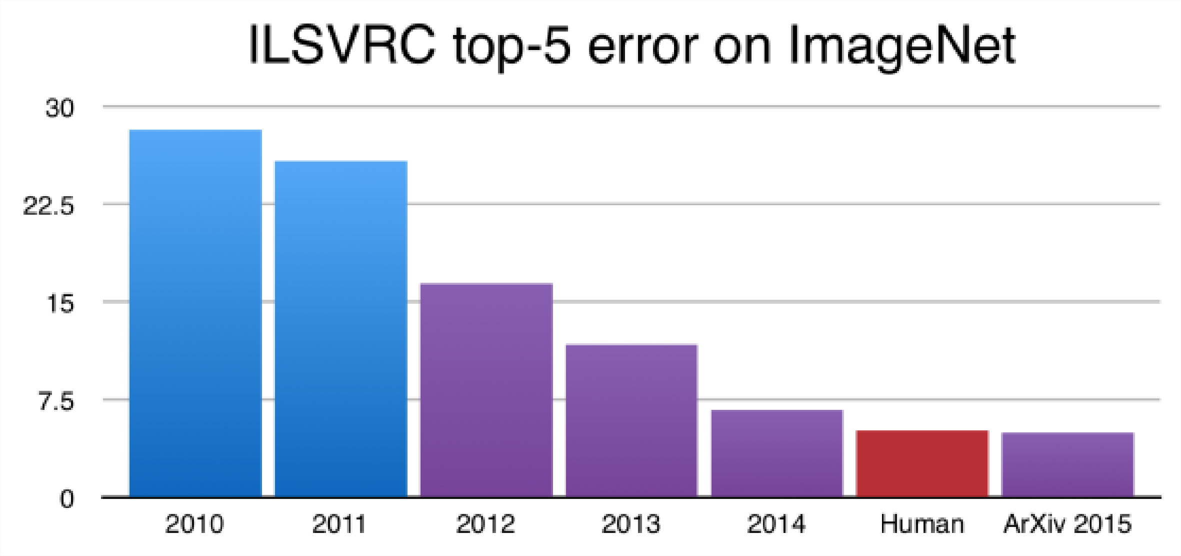
1998 Gradient descent for convolutional NN by LeCun et al.

2006 Deep model by Hinton, Osindero and Teh

Today history

2012 Hinton, Krizhevsky, and Sutskever suggest Dropout

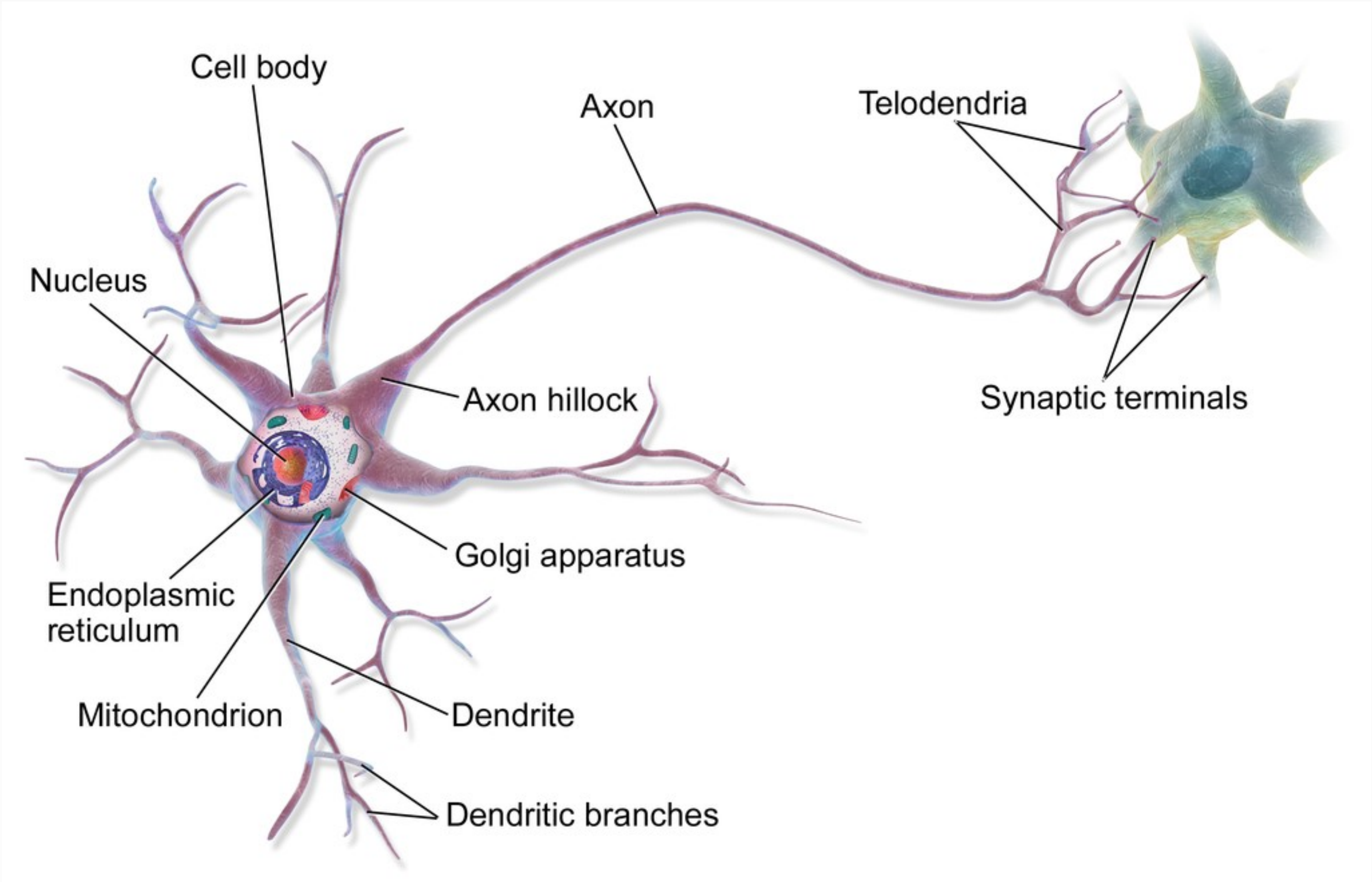
2012 They win ImageNet (and two less known competitions). Deep learning era begins.



Lecture plan

- History of neural networks
- Single layer neural network
- Completeness problem of neural networks
- Multilayer neural networks
- Backpropagation
- Heuristics for neural networks
- Modern neural networks

Biological intuition

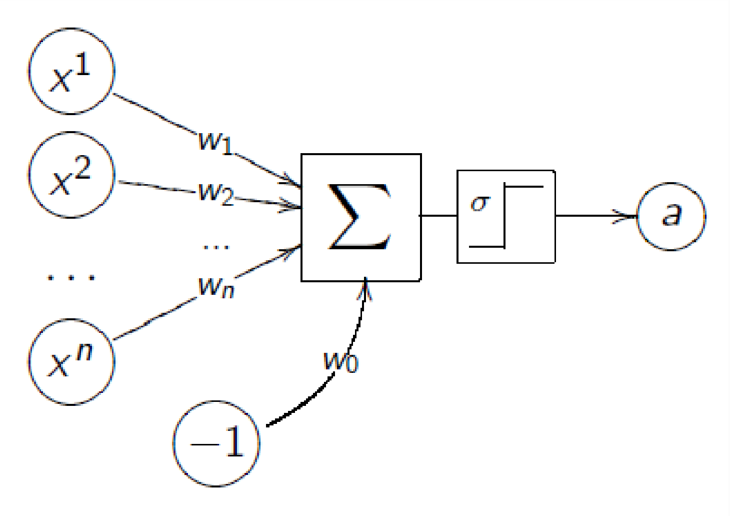


Perceptron

Rosenblatt's perceptron:

$$a_w(x, T^\ell) = \sigma\left(\sum_{i=1}^{n} w_i x^i - w_0\right),$$

where $\sigma(x) = 1$ if $x > 0$ and 0 otherwise.

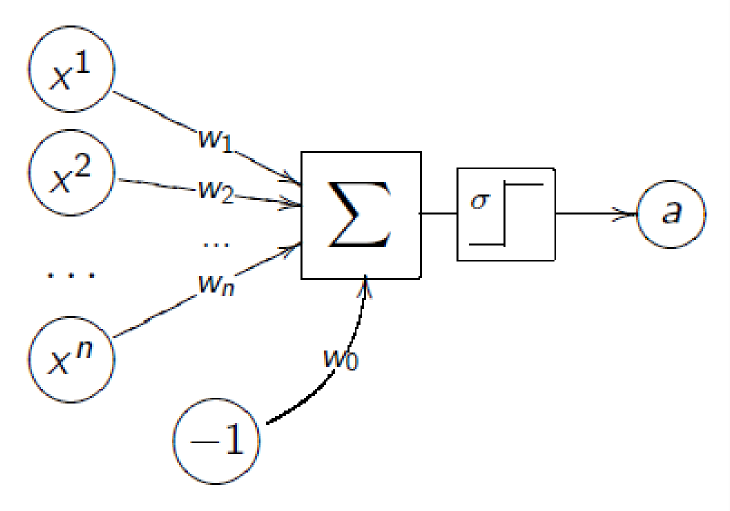


Neuron

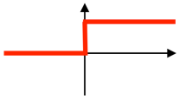
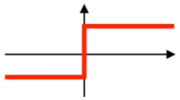
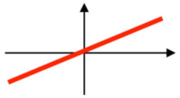
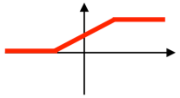
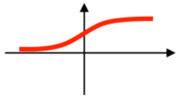
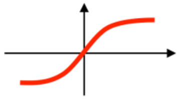
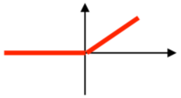
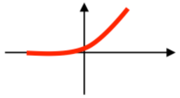
Generalized McCulloch-Pitts neuron:

$$a_w(x, T^\ell) = \sigma\left(\sum_{i=1}^{n} w_i f_i(x) - w_0\right),$$

where σ is an activation function.



Activation functions

Activation function	Equation	Example	1D Graph
Unit step (Heaviside)	$\phi(z) = \begin{cases} 0, & z < 0, \\ 0.5, & z = 0, \\ 1, & z > 0, \end{cases}$	Perceptron variant	
Sign (Signum)	$\phi(z) = \begin{cases} -1, & z < 0, \\ 0, & z = 0, \\ 1, & z > 0, \end{cases}$	Perceptron variant	
Linear	$\phi(z) = z$	Adaline, linear regression	
Piece-wise linear	$\phi(z) = \begin{cases} 1, & z \geq \frac{1}{2}, \\ z + \frac{1}{2}, & -\frac{1}{2} < z < \frac{1}{2}, \\ 0, & z \leq -\frac{1}{2}, \end{cases}$	Support vector machine	
Logistic (sigmoid)	$\phi(z) = \frac{1}{1 + e^{-z}}$	Logistic regression, Multi-layer NN	
Hyperbolic tangent	$\phi(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$	Multi-layer Neural Networks	
Rectifier, ReLU (Rectified Linear Unit)	$\phi(z) = max(0, z)$	Multi-layer Neural Networks	
Rectifier, softplus	$\phi(z) = \ln(1 + e^z)$	Multi-layer Neural Networks	

Scalar products in supervised learning

Classification:

$$Q(w, T^\ell) = \sum_{i=1}^{\ell} L(\langle w, x_i \rangle y_i) \to \min_w;$$

Regression:

$$Q(w, T^\ell) = \sum_{i=1}^{\ell} (\sigma(\langle w, x_i \rangle) - y_i)^2 \to \min_w.$$

Rosenblatt's rule and Hebb's rule

Rosenblatt's rule for $\{1; 0\}$ classification case for weight learning: for each object $x_{(k)}$ change the weight vector:

$$w^{[k+1]} := w^{[k]} - \eta\left(a_w\left(x_{(k)}\right) - y_{(k)}\right).$$

Hebb's rule for $\{1; -1\}$ classification case for weight learning: for each object $x_{(k)}$ change the weight vector:

if $\langle w^{[k]} x_{(k)} \rangle y_{(k)} < 0$ then $w^{[k+1]} := w^{[k]} + \eta x_{(k)} y_{(k)}$.

Delta rule

Let $L(a_w, x) = (\langle w, x \rangle - 1)^2$.

Delta-rule for weight learning: for each object $x_{(k)}$ change the weight vector:

$$w^{[k+1]} := w^{[k]} - \eta\big(\langle w, x_{(k)} \rangle - y_{(k)}\big).$$

Lecture plan

- History of neural networks
- Single layer neural network
- Completeness problem of neural networks
- Multilayer neural networks
- Backpropagation
- Heuristics for neural networks
- Modern neural networks

Completeness problem (for neuron)

Basic idea: synthesize combinations of neurons.

Completeness problem: how rich is the family of functions that can be represented with a neural network?

Start with a single neuron.

Logical functions as neural networks

Logical AND

$$x^1 \wedge x^2 = [x^1 + x^2 - 3/2 > 0]$$

Logical OR

$$x^1 \vee x^2 = [x^1 + x^2 - 1/2 > 0]$$

Logical NOT

$$\neg x^1 = [-x^1 + 1/2 > 0]$$

Two ways of making it more complex

Example (Minkovski):

$$x^1 \oplus x^2$$

Two ways of making it more complex

1. Use non-linear transformation:

$$x^1 \oplus x^2 = [x^1 + x^2 - 2x^1x^2 - 1/2 > 0]$$

2. Build superposition:

$$x^1 \oplus x^2 = [(x^1 \vee x^2) - (x^1 \wedge x^2) - 1/2 > 0]$$

Completeness problem (Boolean functions)

Completeness problem: how rich is the family of functions that can be represented with a neural network?

DNF Theorem:

Any particular Boolean function can be represented by one and only one full disjunctive normal form.

What is with all possible functions?

Gorban Theorem

Theorem (Gorban, 1998)

Let

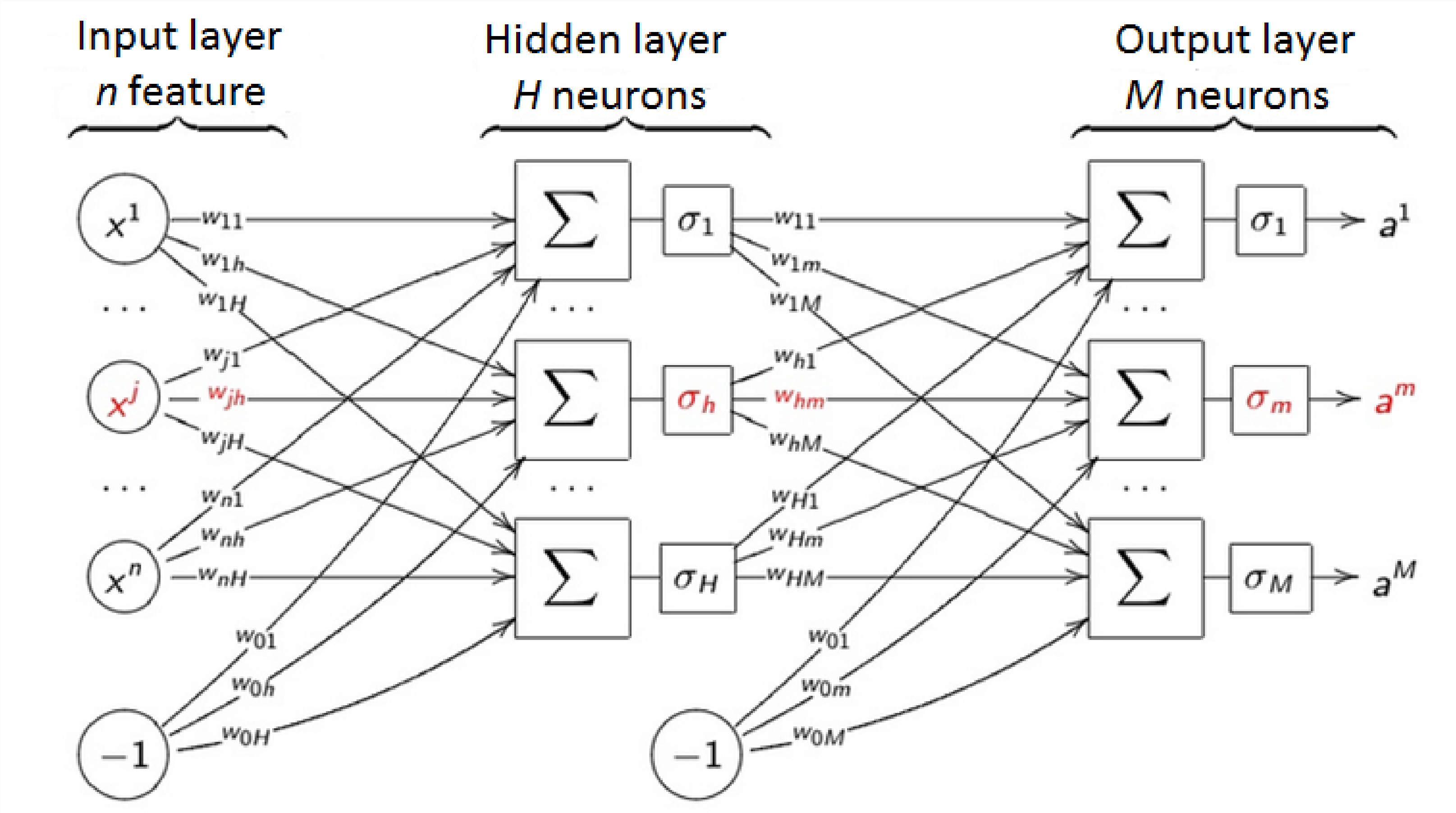
- X be a compact space,
- $C(X)$ be an algebra of continuous on X real-valued functions,
- F be linear subspace $C(X)$, closed with respect to a nonlinear continuous function ϕ and containing constant $(1 \in F)$,
- F separates points in X.

Then F is dense in $C(X)$.

Lecture plan

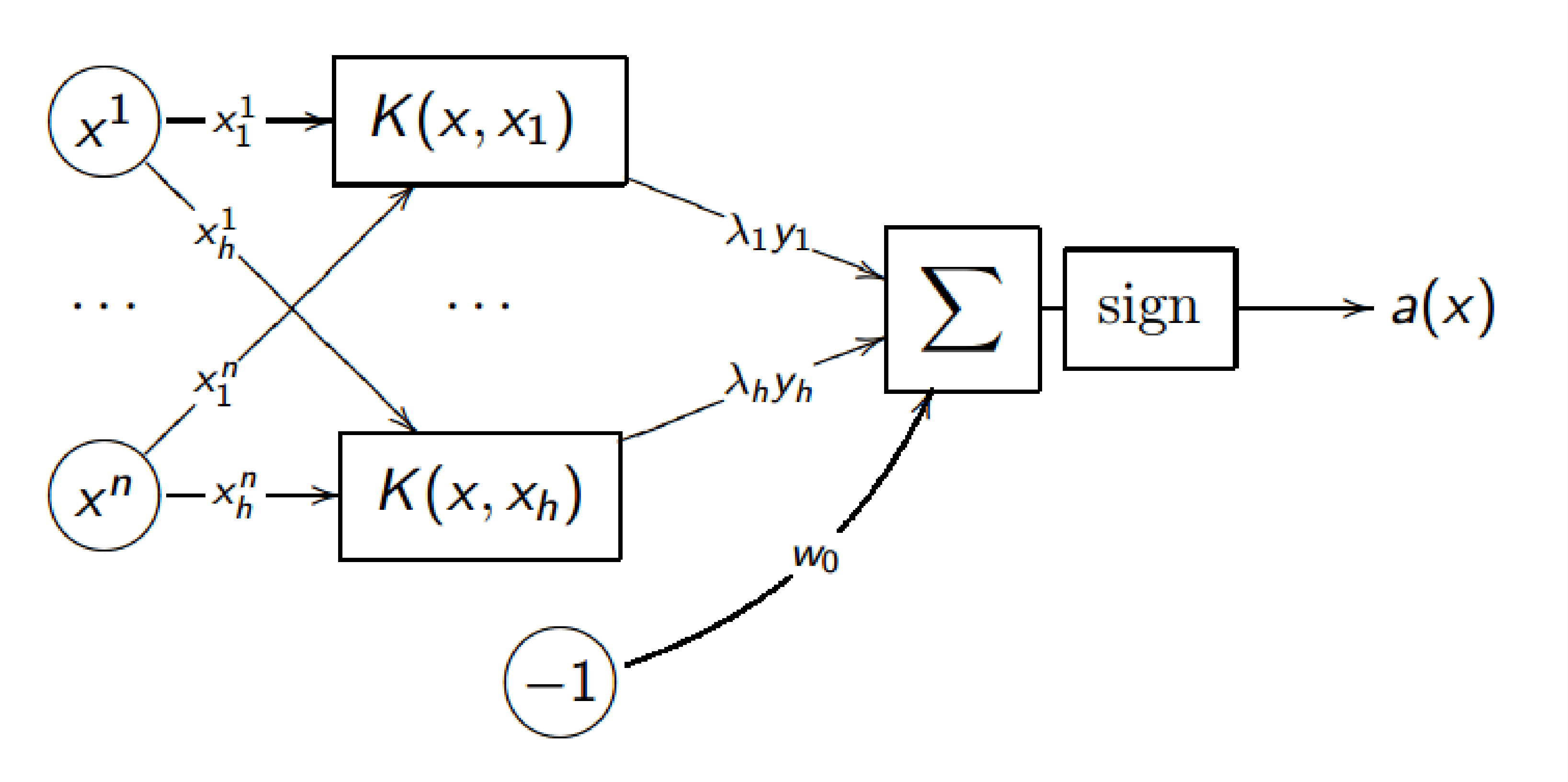
- History of neural networks
- Completeness problem of neural networks
- Single layer neural network
- Multilayer neural networks
- Backpropagation
- Heuristics for neural networks
- Modern neural networks

Multilayer neural network



Neural network for SVM

Order objects by margin:



Multilayer neural network

Any number of layers

Any number of neurons on each layer

Any number of ties between different layers

Weights adjusting

Use SGD to learn weights

$$w = (w_{jh}, w_{hm}) \in \mathbb{R}^{H(n+M-1)M}:$$

$$w^{[t+1]} = w^{[t]} - \eta \nabla L(w, x_i, y_i),$$

where $L(w, x_i, y_i)$ is a loss function (depends on the problem we are solving).

Lecture plan

- History of neural networks
- Completeness problem of neural networks
- Single layer neural network
- Multilayer neural networks
- Backpropagation
- Heuristics for neural networks
- Modern neural networks

Derivation of functions superposition

$$a^m(x_i) = \sigma_m\left(\sum_{h=0}^{H} w_{hm} u^h(x_i)\right);$$

$$u^h(x_i) = \sigma_h\left(\sum_{j=0}^{J} w_{jh} f_j(x_i)\right);$$

Let $L_i(w) = \frac{1}{2}\sum_{m=1}^{M}(a^m(x_i) - y_i^m)^2$.

Find partial derivatives

$$\frac{\partial L_i(w)}{\partial a^m}; \frac{\partial L_i(w)}{\partial u^h}.$$

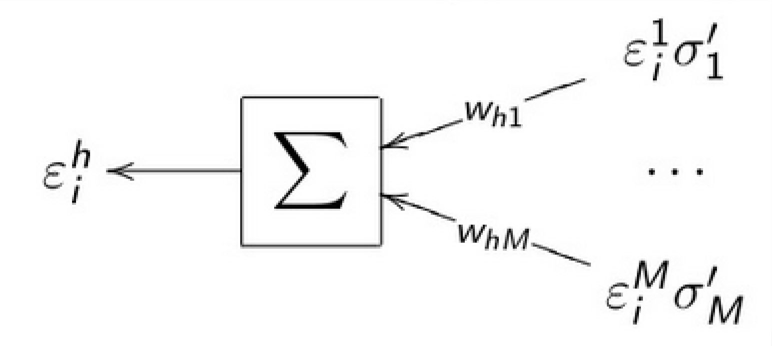
Errors on layers

$$\frac{\partial L_i(w)}{\partial a^m} = a^m(x_i) - y_i^m$$

$\varepsilon_i^m = a^m(x_i) - y_i^m$ is **error on output layer.**

$$\frac{\partial L_i(w)}{\partial u^h} = \sum_{i=1}^{M} (a^m(x_i) - y_i^m)\sigma'_m w_{hm} = \sum_{i=1}^{M} \varepsilon_i^m \sigma'_m w_{hm}$$

$\varepsilon_i^h = \sum_{i=1}^{M} \varepsilon_i^m \sigma'_m w_{hm}$ is **error on hidden layer.**



Backpropagation discussion (advantages)

Advantages:

- efficacy: gradient can be computed in a time, which is comparable to the time of the network processing;
- can be easily applied for any σ, L;
- can be applied in dynamical learning;
- not all the sample objects can be used;
- can be paralleled.

Backpropagation discussion (disadvantages)

Disadvantages:

- do not always converge;
- can stuck in local optima;
- number of neurons in the hidden layer should be fixed;
- the more ties, the probable overfitting is;
- “paralysis” of a single neuron and for network.

Lecture plan

- History of neural networks
- Completeness problem of neural networks
- Single layer neural network
- Multilayer neural networks
- Backpropagation
- Heuristics for neural networks
- Modern neural networks

Standard heuristics for gradient descent

- weights initialization;
- order of objects;
- optimization of gradient step;
- regularization (constraints for number of value of weights).

Acceleration of converge

1. Choose more accurate initial approximation. Neurons are tunes as algorithms

- on a random subsample;
- on a random input subset;
- on different initial approximations;

2. "Jogging off" weights.

3. Adaptive gradient step (steppest gradient descent).

Network structure selection

- Number of layers selection
- Number of hidden layer neuron selection
- Dynamical increasing of network
- Dynamical decreasing of network (brain damage)

Lecture plan

- History of neural networks
- Completeness problem of neural networks
- Single layer neural network
- Multilayer neural networks
- Backpropagation
- Heuristics for neural networks
- Modern neural networks

Plethora of neural networks

Tens or even hundreds different neural networks exist:

- self-organizing map
- radial basis function networks
- Bayesian neural networks
- modular neural networks
- echo state networks

… and deep neural networks

Tens or even hundreds different deep neural networks (deep learning networks) exist:

- convolutional neural networks
- recurrental neural networks (including long-short term memory)
- autoencoders
- deep Boltzman machines and deep Belief networks
- deep Q-networks
- …