Nizhny Novgorod State University
Institute of Information Technologies, Mathematics and Mechanics
Department of Computer software and supercomputer technologies

Educational course
«Introduction to deep learning
using the Intel® neon™ Framework»

Fully-connected neural networks

Supported by Intel

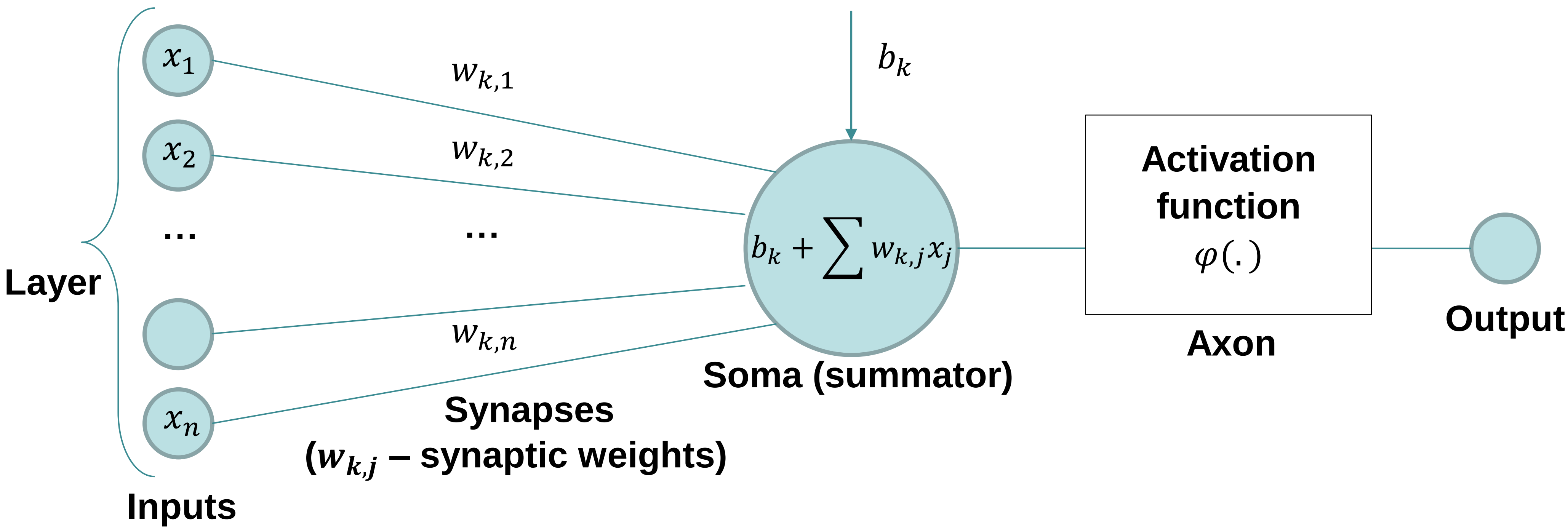
Valentina Kustikova,
Phd, lecturer, department of Computer software
and supercomputer technologies

Content

- Deterministic model of a neuron. Activation functions
- General structure of a multilayer fully-connected neural network
- Training problem of a multilayer fully-connected neural network. Optimization formulation of the training problem. Cost function
- Backward propagation of errors (backpropagation algorithm)
- Sequential and batch modes of training
- Heuristic recommendations for improving the performance of the backpropagation algorithm

DETERMINISTIC MODEL OF A NEURON. ACTIVATION FUNCTIONS

Deterministic model of a neuron (1)



Deterministic model of a neuron (2)

- The neuron model consists of three main components:
  - ***Synapses*** are input signals, each of which is characterized by its own weight
  - ***Summator*** is a component that adds input signals multiplied by synaptic weights
  - ***Activation function*** is a component that limits the amplitude of the output signal. Output of a neuron, as a rule, belongs the interval $[0,1]$ or $[-1,1]$

Deterministic model of a neuron (3)

- Mathematical model of a neuron:

$$u_k = \sum_{j=1}^{n} w_{k,j} x_j \,, \qquad y_k = \varphi(u_k + b_k) \qquad (1)$$

- Assuming $v_k = u_k + b_k$, the pair of equations can be written as follows:

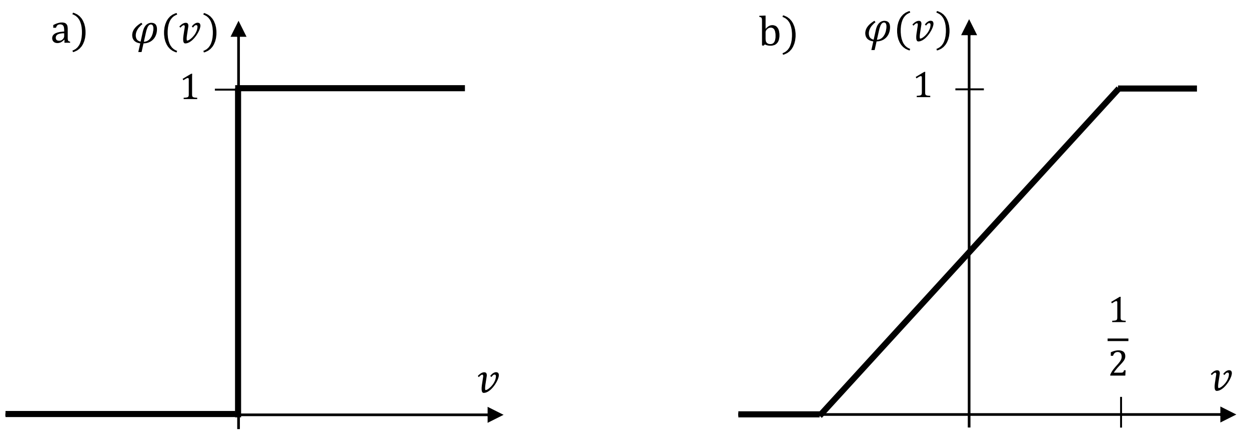
$$v_k = \sum_{j=0}^{n} w_{k,j} x_j \,, \qquad y_k = \varphi(v_k), \qquad (2)$$

where $x_0 = 1$ is a new synapse, $w_{k,0} = b_k$ is its weight

- Models (1) и (2) are equivalent neuron models

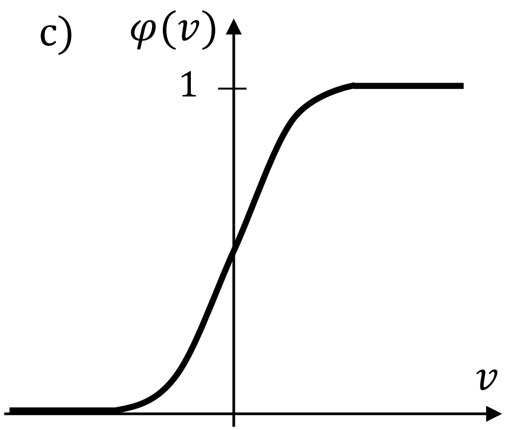
Activation functions (1)

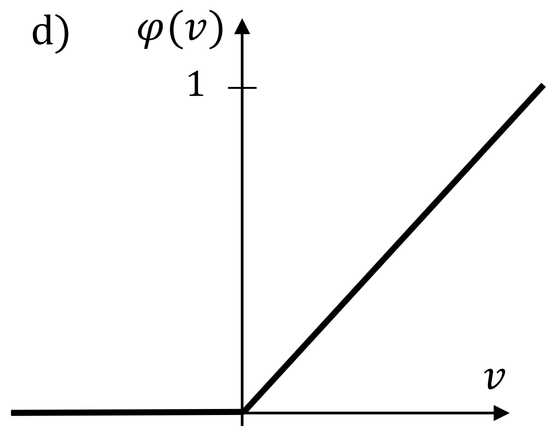
- ***Threshold function*** (a) describes an all-or-nothing principle. Applies to tasks that require a binary response
- ***Piecewise-linear function*** (b) can be considered as an approximation of a nonlinear amplifier



Activation functions (2)

- ***Sigmoid functions*** (c). Examples of such functions are the logistic function and the hyperbolic tangent
- ***Rectified Linear Unit*** (ReLU, d)

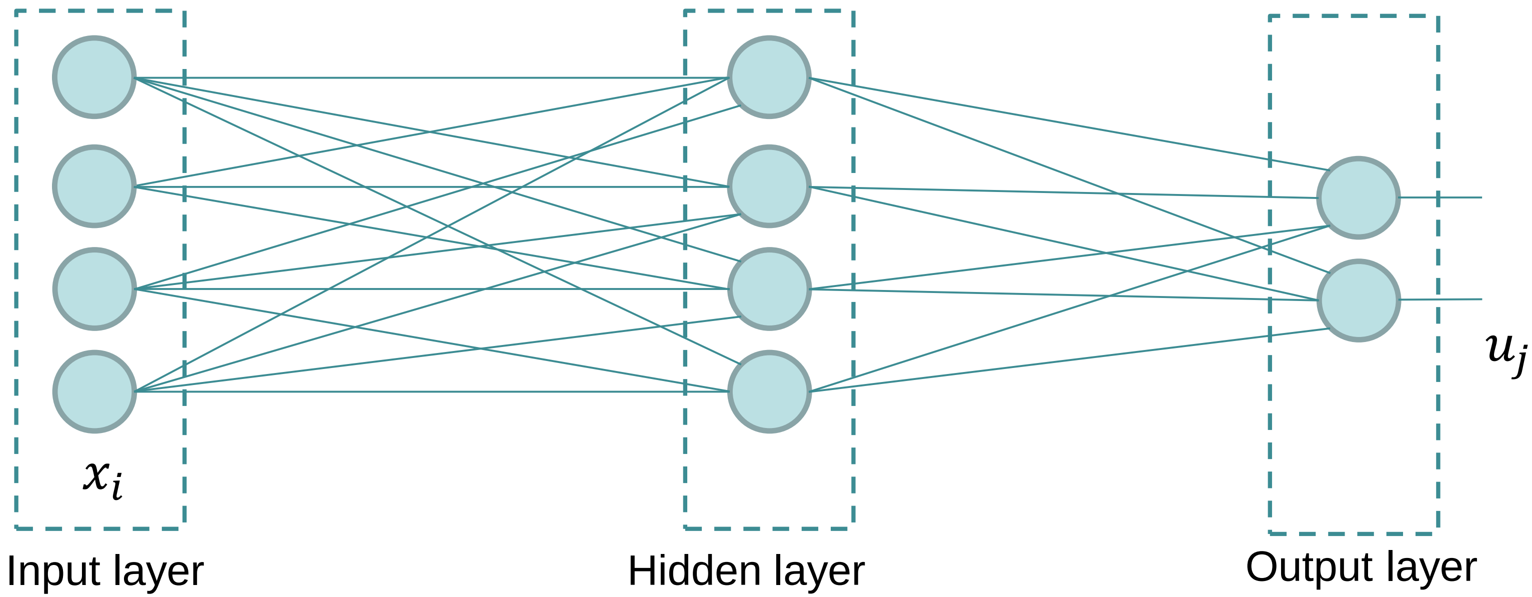




GENERAL STRUCTURE OF A MULTILAYER FULLY-CONNECTED NEURAL NETWORK

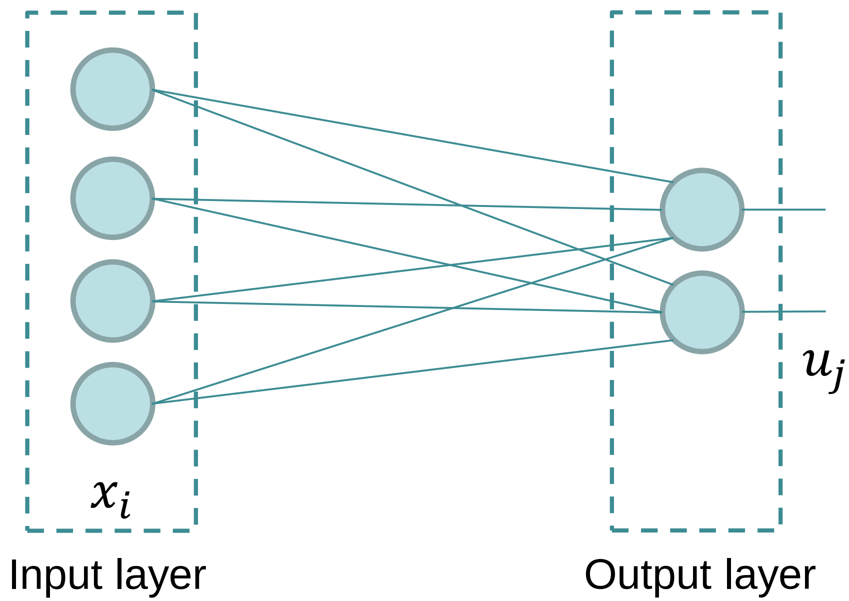
General structure of a multilayer fully-connected neural network (1)

- A multilayer fully-connected neural network contains neurons that are distributed across ***layers***



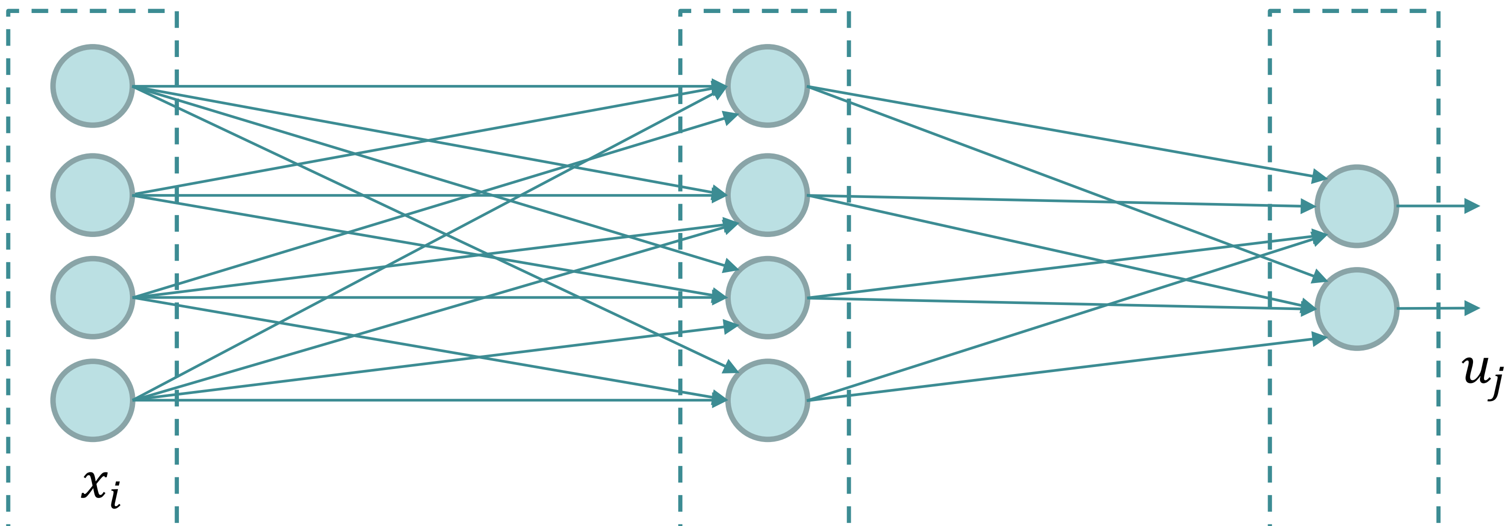
General structure of a multilayer fully-connected neural network (2)

- In the simplest case, the network has input and output layers, and the network is ***a single layer neural network***



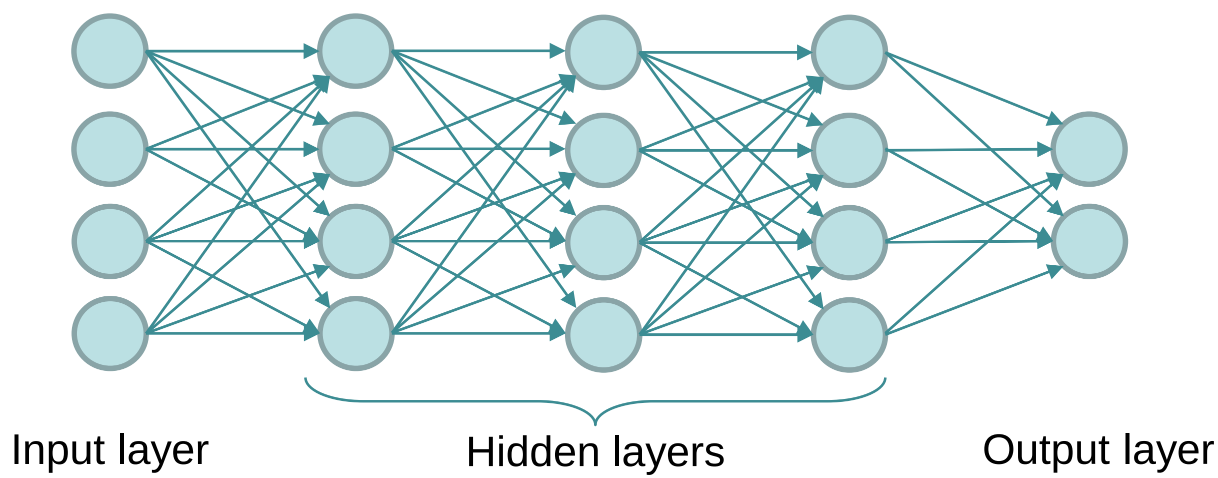
General structure of a multilayer fully-connected neural network (3)

- If the signal passes from the neurons of the input layer to the output neurons, then such a network is a ***feedforward network***



General structure of a multilayer fully-connected neural network (4)

- A network can contain many hidden layers. In this case the network is called ***a multilayer network***
- If all nodes of the layer are connected to the nodes of the next layer, then the layer is called ***fully-connected***
- If this condition is met for all layers of the network, the network is called ***fully-connected neural network*** (FCNN)



TRAINING PROBLEM OF A MULTILAYER FULLY-CONNECTED NEURAL NETWORK

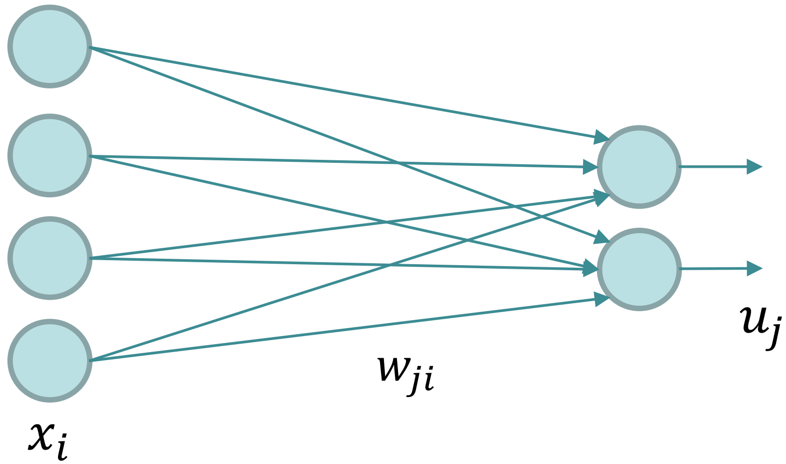
Training problem of a multilayer fully-connected neural network

- ***A training purpose*** is to adjust network weights

- ***A training task*** is a task of minimizing ***a cost function*** (an error function) reflecting the difference in the expected signal received at the network output and the actual signal corresponding to the current input, over the complete training set

Quadratic cost function for a single layer neural network (1)

- Consider a single layer neural network containing N input and M output neurons



- Train set $\langle X, Y \rangle$
  - L is a number of samples in the set,
  - X is a set of input signals (dimension equals N),
  - Y is a set of actual output signals (dimension equals M)

Quadratic cost function for a single layer neural network (2)

- The quadratic cost function:

$$E = \frac{1}{2}\sum_{k=1}^{L}\left\|y^k - u^k\right\|^2 = \frac{1}{2}\sum_{k=1}^{L}\sum_{j=1}^{M}\left(y_j^k - u_j^k\right)^2$$

$$= \frac{1}{2}\sum_{k=1}^{L}\sum_{j=1}^{M}\left(y_j^k - \varphi\left(\sum_{i=0}^{N} w_{ji}x_i^k\right)\right)^2,$$

where $y^k = \left(y_j^k\right)_{j=\overline{1,M}} \in Y$ is a train set, $u^k = \left(u_j^k\right)_{j=\overline{1,M}}$ is a network output for the input $x^k = \left(x_i^k\right)_{i=\overline{1,N}} \in X$

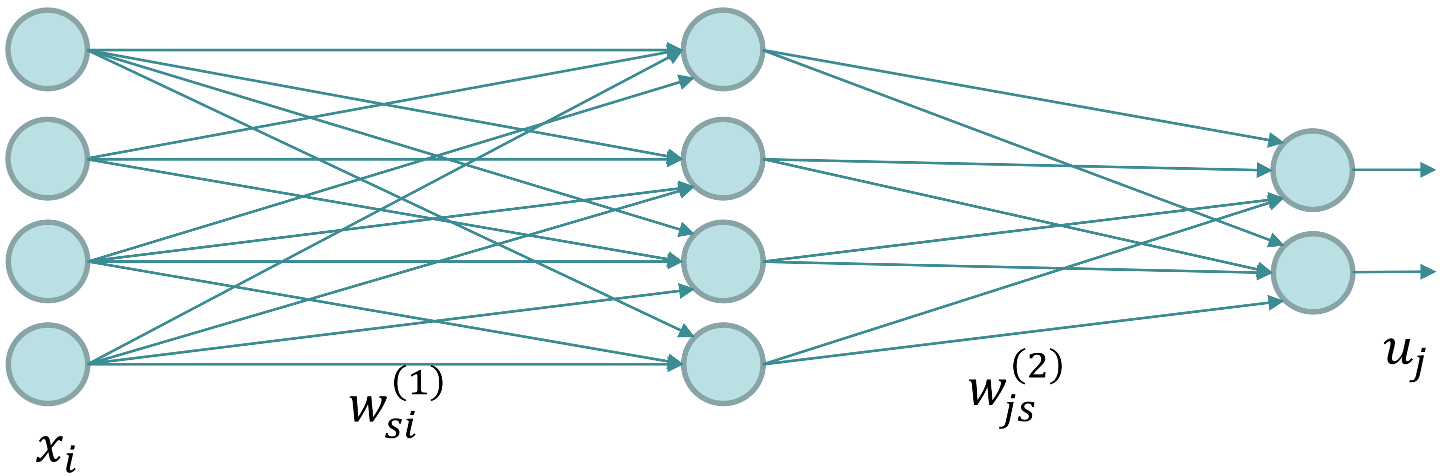
- Also, the normalization of the indicated metric is introduced by the number of training samples L

Quadratic cost function for a single layer neural network (3)

- What happens if we use two-layer network instead of a single layer?

Quadratic cost function for a two-layer neural network (1)

- Let us consider a two-layer neural network:

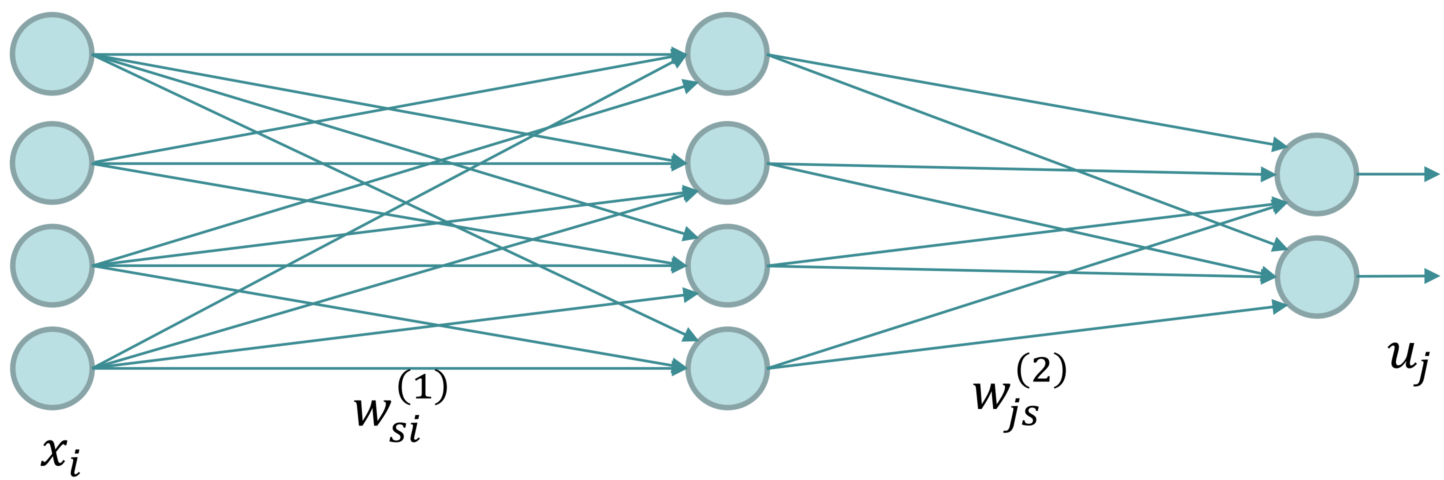


- $w_{si}^{(1)}, w_{js}^{(2)}$ are synaptic weights
- The output signal of the hidden layer neuron is described as follows:

  $v_s = \varphi^{(1)}\left(\sum_{i=0}^{N} w_{si}^{(1)} x_i\right), s = \overline{0, K}$, where K is a neuron number at the hidden layer

Quadratic cost function for a two-layer neural network (2)

- Let us consider a two-layer neural network:



- Signal of the output neuron j:

$$u_j = \varphi^{(2)}\left(\sum_{s=0}^{K} w_{js}^{(2)} v_s\right) = \varphi^{(2)}\left(\sum_{s=0}^{K} w_{js}^{(2)} \varphi^{(1)}\left(\sum_{i=0}^{N} w_{si}^{(1)} x_i\right)\right), j = \overline{1, M}.$$

Quadratic cost function for a two-layer neural network (3)

- Quadratic cost function for the training set:

$$E(w) = \frac{1}{2}\sum_{k=1}^{L}\sum_{j=1}^{M}\left(y_j^k - u_j^k\right)^2$$

$$= \frac{1}{2}\sum_{k=1}^{L}\sum_{j=1}^{M}\left(y_j^k - \varphi^{(2)}\left(\sum_{s=0}^{K} w_{js}^{(2)} \varphi^{(1)}\left(\sum_{i=0}^{N} w_{si}^{(1)} x_i^k\right)\right)\right)^2$$

Optimization formulation of the training problem with a quadratic cost function

- A general mathematical formulation of the training problem with a quadratic cost function:

$$\min_w E(w) = \min_w \left\{ \frac{1}{L} \sum_{k=1}^{L} \left\{ \frac{1}{2} \sum_{j=1}^{M} \left(y_j^k - u_j^k \right)^2 \right\} \right\}$$

- A quadratic cost function reflects the difference of the network output and the label
- A quadratic (Euclidean) cost function is applied when solving ***the regression problem***

Optimization formulation of the training problem with a cross-entropy error function

- For the classification problem the ***cross-entropy*** is chosen as a cost function. Let us consider the problem of training network:

$$\min_w E(w) = \min_w \left\{ -\frac{1}{L} \sum_{k=1}^{L} \sum_{j=1}^{M} y_j^k \ln u_j^k \right\}$$

  where $y_j^k = 1 \leftrightarrow x^k$ belongs the class j, otherwise $y_j^k = 0$

- Cross-entropy is a differentiable approximation of the cost function for the classification problem "0-1"
- As an activation function on the last layer, it is recommended to select softmax function:

$$\varphi(u_j) = \frac{e^{u_j}}{\sum_{i=1}^{M} e^{u_i}}$$

BACKPROPAGATION ALGORITHM

Backpropagation algorithm (1)

- Backpropagation algorithm determines the strategy of changing network parameters w during training using gradient optimization methods
- Gradient methods at each step refine the parameter values:

$$w(k+1) = w(k) + \eta p(w)$$

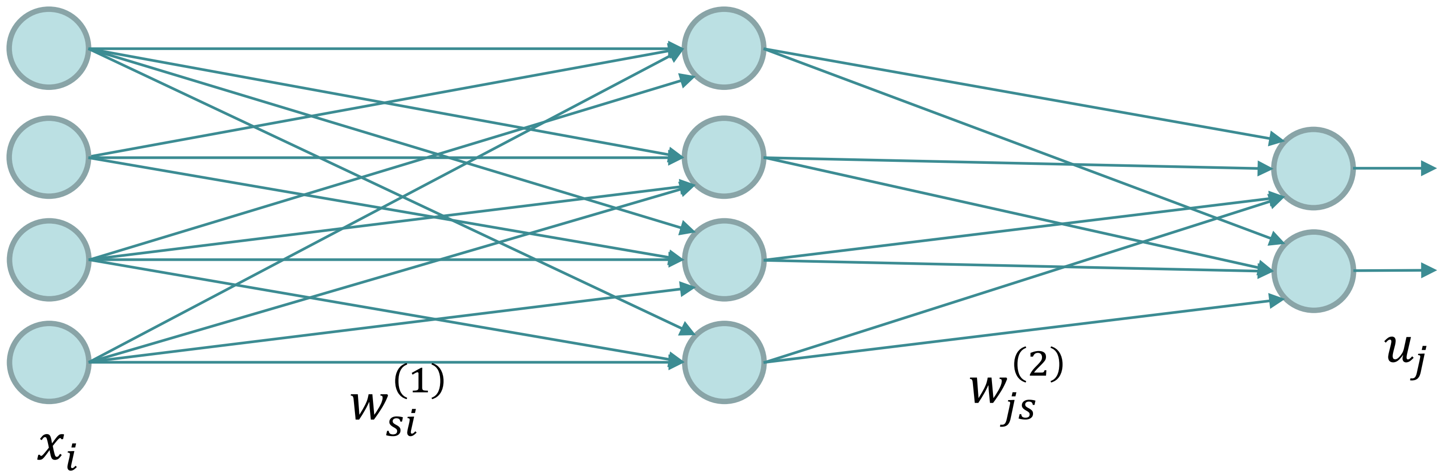
  - $\eta, 0 < \eta < 1$ is ***a learning rate*** (the “speed” of the movement in the direction of the function minimum),
  - $p(w)$ is a direction in a multidimensional space of network parameters
- In the classical backpropagation algorithm, the direction of motion coincides with the direction of the antigradient $p(w) = -\nabla E(w)$

Backpropagation algorithm (2)

- ***Initialization of the network synaptic weights*** (randomly from some distribution)
- Repeating the following steps for each sample of the training dataset
  1. ***Feed forward:***
     1. Calculating output values for all neurons
     2. Calculating derivatives of activation functions for each layer
  2. ***Backward:***
     1. Calculating cost function value and its derivative
     2. Refining synaptic weights
- ***Stopping criteria:*** the number of iterations (the number of passes along the entire training set), the cost function value

Backpropagation algorithm for two-layer network (1)

- Let us calculate the derivatives and the cost function values on the example of a two-layer neural network



- The cost function is described as follows:

$$E(w) = \frac{1}{2}\sum_{j=1}^{M}\left(y_j - \varphi^{(2)}\left(\sum_{s=0}^{K} w_{js}^{(2)}\,\varphi^{(1)}\left(\sum_{i=0}^{N} w_{si}^{(1)} x_i\right)\right)\right)^2$$

Backpropagation algorithm for two-layer network (2)

- The derivative of the cost function with respect to the parameters of the last network layer:

$$\frac{\partial E}{\partial w_{js}^{(2)}} = \frac{\partial \left(\boxed{\frac{1}{2}\sum_{j=1}^{M}\left(y_j - \overbrace{\varphi^{(2)}\left(\sum_{s=0}^{K} \boldsymbol{w}_{\boldsymbol{js}}^{(\boldsymbol{2})}\, \varphi^{(1)}\left(\sum_{i=0}^{N} w_{si}^{(1)} x_i\right)\right)}^{u_j}\right)^2} \right)}{\partial \boldsymbol{w}_{\boldsymbol{js}}^{(\boldsymbol{2})}}$$

$$= -\boxed{\left(y_j - u_j\right)}\frac{d\varphi^{(2)}\left(g_j\right)}{dg_j} v_s = \delta_j^{(2)} v_s,$$

$$g_j = \sum_{s=0}^{K} w_{js}^{(2)} v_s, \qquad \delta_j^{(2)} = \frac{\partial E(w)}{\partial g_j}$$

Backpropagation algorithm for two-layer network (2)

- The derivative of the cost function with respect to the parameters of the last network layer:

$$\frac{\partial E}{\partial w_{js}^{(2)}} = \frac{\partial \left(\frac{1}{2} \sum_{j=1}^{M} \left(y_j \boxed{- \varphi^{(2)} \left(\sum_{s=0}^{K} \boldsymbol{w}_{js}^{(2)} \overbrace{\varphi^{(1)} \left(\sum_{i=0}^{N} w_{si}^{(1)} x_i \right)}^{v_s} \right)} \right)^2 \right)}{\partial \boldsymbol{w}_{js}^{(2)}}$$

$$= \boxed{-} \left(y_j - u_j \right) \boxed{\frac{d\varphi^{(2)}(g_j)}{d g_j}} v_s = \delta_j^{(2)} v_s,$$

$$g_j = \sum_{s=0}^{K} w_{js}^{(2)} v_s, \qquad \delta_j^{(2)} = \frac{\partial E(w)}{\partial g_j}$$

Backpropagation algorithm for two-layer network (2)

- The derivative of the cost function with respect to the parameters of the last network layer:

$$\frac{\partial E}{\partial w_{js}^{(2)}} = \frac{\partial\left(\frac{1}{2}\sum_{j=1}^{M}\left(y_j - \varphi^{(2)}\left(\boxed{\sum_{s=0}^{K} \boldsymbol{w}_{js}^{(\mathbf{2})} \underbrace{\varphi^{(1)}\left(\sum_{i=0}^{N} w_{si}^{(1)} x_i\right)}_{v_s}}\right)\right)^2\right)}{\partial \boldsymbol{w}_{js}^{(\mathbf{2})}}$$

$$= -\left(y_j - u_j\right)\frac{d\varphi^{(2)}\left(g_j\right)}{d g_j}\boxed{v_s} = \delta_j^{(2)} v_s,$$

$$g_j = \sum_{s=0}^{K} w_{js}^{(2)} v_s, \qquad \delta_j^{(2)} = \frac{\partial E(w)}{\partial g_j}$$

Backpropagation algorithm for two-layer network (3)

- The derivative of the cost function with respect to the parameters of hidden layer:

$$\frac{\partial E}{\partial w_{si}^{(1)}} = \frac{\partial \left(\boxed{\frac{1}{2}\sum_{j=1}^{M}\left(y_j - \varphi^{(2)}\left(\sum_{s=0}^{K} w_{js}^{(2)}\, \varphi^{(1)}\left(\sum_{i=0}^{N} \boldsymbol{w_{si}^{(1)}} x_i\right)\right)\right)^2} \right)}{\partial \boldsymbol{w_{si}^{(1)}}}$$

$$= -\boxed{\sum_{j=1}^{M}(y_j - u_j)} \frac{d\varphi^{(2)}(g_j)}{dg_j}\frac{dg_j(v_s)}{dv_s}\frac{d\varphi^{(1)}(f_s)}{df_s} x_i =$$

$$= -\sum_{j=1}^{M}(y_j - u_j)\frac{d\varphi^{(2)}(g_j)}{dg_j} w_{js}^{(2)} \frac{d\varphi^{(1)}(f_s)}{df_s} x_i = \delta_s^{(1)} x_i,$$

$$f_s = \sum_{i=0}^{N} w_{si}^{(1)} x_i\,, \qquad \delta_s^{(1)} = \frac{\partial E(w)}{\partial f_s}$$

Backpropagation algorithm for two-layer network (3)

- The derivative of the cost function with respect to the parameters of hidden layer:

$$\frac{\partial E}{\partial w_{si}^{(1)}}=\frac{\partial\left(\frac{1}{2}\sum_{j=1}^{M}\left(y_j\boxed{-\varphi^{(2)}\left(\sum_{s=0}^{K}w_{js}^{(2)}\,\varphi^{(1)}\left(\sum_{i=0}^{N}\boldsymbol{w}_{\boldsymbol{si}}^{(\mathbf{1})}x_i\right)\right)}\right)^2\right)}{\partial \boldsymbol{w}_{\boldsymbol{si}}^{(\mathbf{1})}}$$

$$=\boxed{-}\sum_{j=1}^{M}(y_j-u_j)\boxed{\frac{d\varphi^{(2)}(g_j)}{dg_j}}\frac{dg_j(v_s)}{dv_s}\frac{d\varphi^{(1)}(f_s)}{df_s}x_i=$$

$$=-\sum_{j=1}^{M}(y_j-u_j)\frac{d\varphi^{(2)}(g_j)}{dg_j}w_{js}^{(2)}\frac{d\varphi^{(1)}(f_s)}{df_s}x_i=\delta_s^{(1)}x_i,$$

$$f_s=\sum_{i=0}^{N}w_{si}^{(1)}x_i\,,\qquad \delta_s^{(1)}=\frac{\partial E(w)}{\partial f_s}$$

Backpropagation algorithm for two-layer network (3)

- The derivative of the cost function with respect to the parameters of hidden layer:

$$\frac{\partial E}{\partial w_{si}^{(1)}} = \frac{\partial\left(\frac{1}{2}\sum_{j=1}^{M}\left(y_j - \varphi^{(2)}\left(\boxed{\sum_{s=0}^{K} w_{js}^{(2)}\,\varphi^{(1)}\left(\sum_{i=0}^{N} \boldsymbol{w}_{\boldsymbol{si}}^{(\mathbf{1})} x_i\right)}\right)\right)^2\right)}{\partial \boldsymbol{w}_{\boldsymbol{si}}^{(\mathbf{1})}}$$

$$= -\sum_{j=1}^{M}\left(y_j - u_j\right)\frac{d\varphi^{(2)}(g_j)}{dg_j}\boxed{\frac{dg_j(v_s)}{dv_s}}\frac{d\varphi^{(1)}(f_s)}{df_s}x_i =$$

$$= -\sum_{j=1}^{M}\left(y_j - u_j\right)\frac{d\varphi^{(2)}(g_j)}{dg_j}w_{js}^{(2)}\frac{d\varphi^{(1)}(f_s)}{df_s}x_i = \delta_s^{(1)}x_i,$$

$$f_s = \sum_{i=0}^{N} w_{si}^{(1)} x_i\,, \qquad \delta_s^{(1)} = \frac{\partial E(w)}{\partial f_s}$$

Backpropagation algorithm for two-layer network (3)

- The derivative of the cost function with respect to the parameters of hidden layer:

$$\frac{\partial E}{\partial w_{si}^{(1)}} = \frac{\partial \left(\frac{1}{2} \sum_{j=1}^{M} \left(y_j - \varphi^{(2)} \left(\sum_{s=0}^{K} w_{js}^{(2)} \boxed{\varphi^{(1)} \left(\sum_{i=0}^{N} \boldsymbol{w_{si}^{(1)}} x_i \right)} \right) \right)^2 \right)}{\partial \boldsymbol{w_{si}^{(1)}}}$$

$$= -\sum_{j=1}^{M} (y_j - u_j) \frac{d\varphi^{(2)}(g_j)}{dg_j} \frac{dg_j(v_s)}{dv_s} \boxed{\frac{d\varphi^{(1)}(f_s)}{df_s}} x_i =$$

$$= -\sum_{j=1}^{M} (y_j - u_j) \frac{d\varphi^{(2)}(g_j)}{dg_j} w_{js}^{(2)} \frac{d\varphi^{(1)}(f_s)}{df_s} x_i = \delta_s^{(1)} x_i,$$

$$f_s = \sum_{i=0}^{N} w_{si}^{(1)} x_i, \qquad \delta_s^{(1)} = \frac{\partial E(w)}{\partial f_s}$$

Backpropagation algorithm for two-layer network (3)

- The derivative of the cost function with respect to the parameters of hidden layer:

$$\frac{\partial E}{\partial w_{si}^{(1)}} = \frac{\partial\left(\frac{1}{2}\sum_{j=1}^{M}\left(y_j - \varphi^{(2)}\left(\sum_{s=0}^{K} w_{js}^{(2)}\,\varphi^{(1)}\left(\boxed{\sum_{i=0}^{N} \boldsymbol{w_{si}^{(1)}} x_i}\right)\right)\right)^2\right)}{\partial \boldsymbol{w_{si}^{(1)}}}$$

$$= -\sum_{j=1}^{M}(y_j - u_j)\frac{d\varphi^{(2)}(g_j)}{dg_j}\frac{dg_j(v_s)}{dv_s}\frac{d\varphi^{(1)}(f_s)}{df_s}\boxed{x_i} =$$

$$= -\sum_{j=1}^{M}(y_j - u_j)\frac{d\varphi^{(2)}(g_j)}{dg_j} w_{js}^{(2)} \frac{d\varphi^{(1)}(f_s)}{df_s} x_i = \delta_s^{(1)} x_i,$$

$$f_s = \sum_{i=0}^{N} w_{si}^{(1)} x_i\,, \qquad \delta_s^{(1)} = \frac{\partial E(w)}{\partial f_s}$$

Backpropagation algorithm for two-layer network (4)

- The derivative structures of the cost function with respect to the parameters of the output and hidden layer are identical:

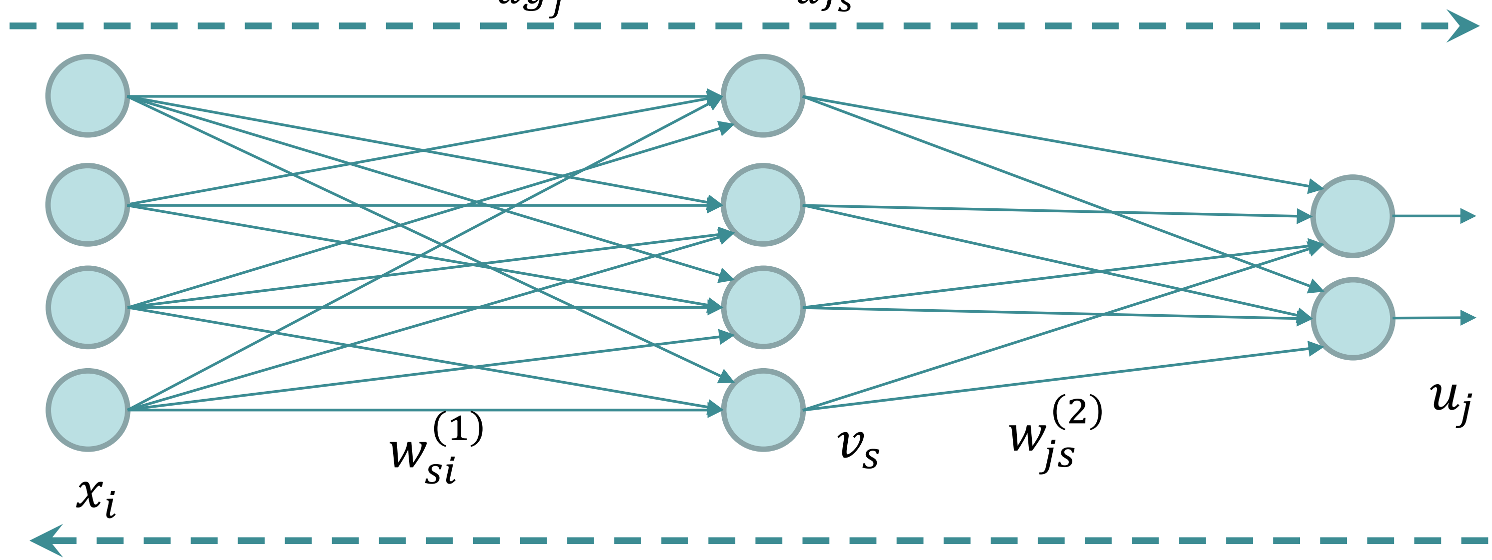
$$\frac{\partial E}{\partial w_{js}^{(2)}} = \delta_j^{(2)} v_s \qquad \frac{\partial E}{\partial w_{si}^{(1)}} = \delta_s^{(1)} x_i$$

Input neuron of the synaptic connection

The error transferred to the neuron with which this connection is established

Backpropagation algorithm for two-layer network (5)

1. ***Feed forward:***
   - Computing v_s and u_j
   - Computing $\frac{d\varphi^{(2)}(g_j)}{dg_j}$ and $\frac{d\varphi^{(1)}(f_s)}{df_s}$



2. ***Backward:***
   - Computing the cost function E and its derivatives $\frac{\partial E}{\partial w_{js}^{(2)}}, \frac{\partial E}{\partial w_{si}^{(1)}}$
   - Refining weights $w(k+1) = w(k) - \eta \nabla E(w)$

Convergence of backpropagation algorithm (1)

- There is no convergence proof of backpropagation algorithm
- A reasonable stopping condition: the Euclidean norm of the gradient has reached sufficiently small values
- To satisfy this condition, a large number of backpropagation iterations is required

Convergence of backpropagation algorithm (2)

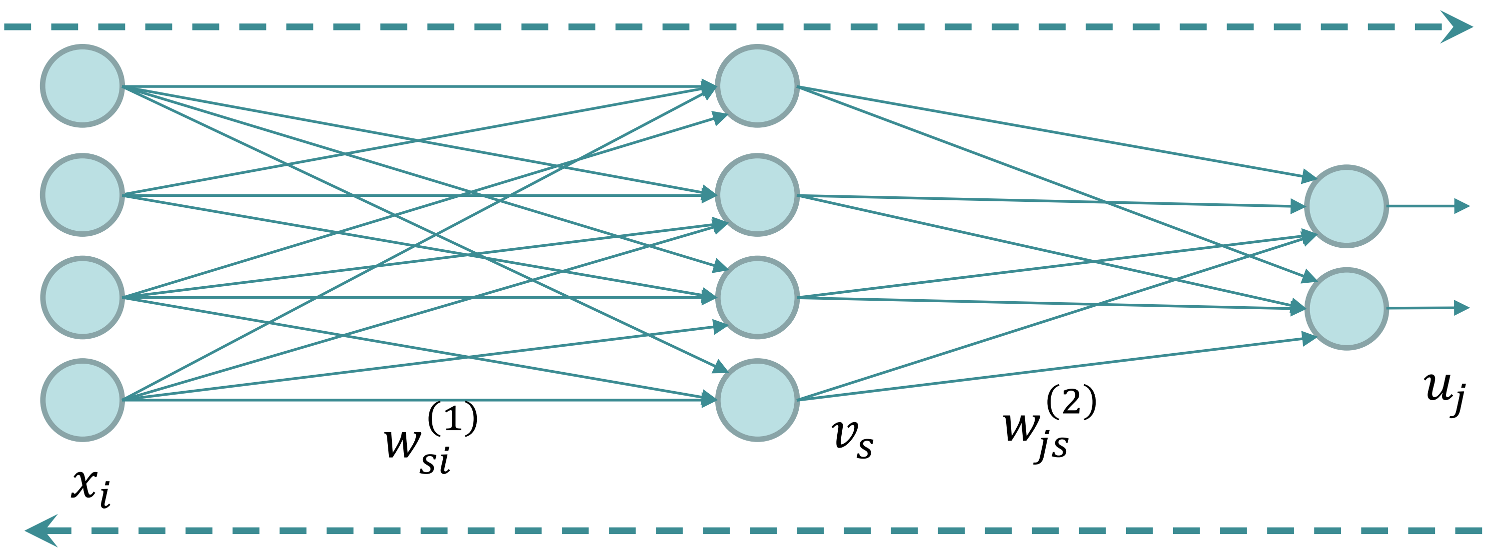
- Weaker stopping condition: during the full cycle of training samples presentation, the absolute value of the cost function change is rather small (in the range from 0.1 to 1%)
- This criteria does not guarantee the resulting network has good generalizing properties

Sequential and batch modes of training (1)

- Training implies that samples of the training set is repeatedly fed to the network input
- One complete cycle of presenting the complete training set is called the ***epoch***
  - During training, several such cycles can be performed until the synaptic weights are stabilized, or the minimum value of the cost function is achieved
  - In the implementation of epochs, it is advisable to change the order of the training samples, providing a stochastic search

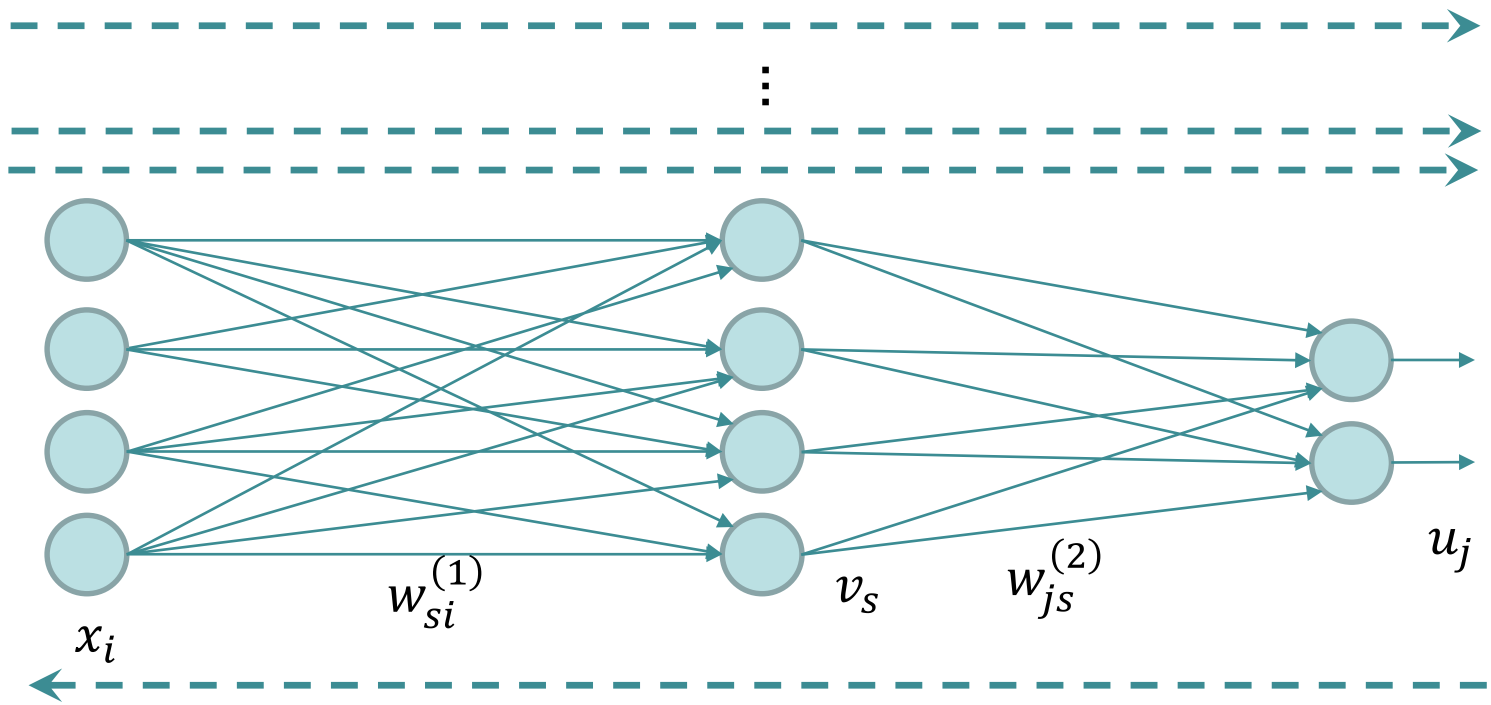
Sequential and batch modes of training (2)

- Implementation modes of backpropagation:
  - ***Sequential (or stochastic) mode***. In this mode weights are modified after each training sample



Sequential and batch modes of training (3)

- Implementation modes of backpropagation:
  - ***Batch mode.*** In this mode weights are modified after presenting all training samples of the epoch



Sequential and batch modes of training (4)

- The cost function for the complete training set normalized by the number of training samples is described as follows:

$$E(w) = \frac{1}{L}\sum_{k=1}^{L}\left\{\frac{1}{2}\sum_{j=1}^{M}\left(y_j^k - u_j^k\right)^2\right\}$$

- The weight changes are also carried out over the complete data set. For a two-layered fully connected neural network, the formulas can be written as follows:

$$\Delta w_{js}^{(2)} = -\frac{\eta}{L}\sum_{k=1}^{L}\left(y_j^k - u_j^k\right)\left.\frac{d\varphi^{(2)}(g_j)}{dg_j}\right|_{x^k} \left. v_s\right|_{x^k},$$

$$\Delta w_{si}^{(1)} = -\frac{\eta}{L}\sum_{k=1}^{L}\sum_{j=1}^{M}\left(y_j - u_j\right)\left.\frac{d\varphi^{(2)}(g_j)}{dg_j}\right|_{x^k} \left.\frac{dg_j(v_s)}{dv_s}\right|_{x^k} \left.\frac{d\varphi^{(1)}(f_s)}{df_s}\right|_{x^k} x_i^k$$

Sequential and batch modes of training (5)

- ***Sequential mode***
  - Slow

- ***Batch mode***
  - Fast and stable
  - Can “get stuck” in local minima

Sequential and batch modes of training (6)

- As a compromise one can use mini-batches
  - The training set is divided into mini-batches
  - A feed forward is performed for the complete set of samples from the mini-batch
  - The backward and weight correction is carried out after processing the mini-batch

Heuristic recommendations for improving performance of the backpropagation

- ***Informativeness maximization***
  - Presence of cardinally different training samples (not only visually perceived differences in appearance, but also differences in the cost function values)
- ***Activation function***
  - A fully-connected neural network trains faster if the activation function is antisymmetric $\varphi(-x) = -\varphi(x)$
- ***Input normalization***
  - All inputs should be preliminarily normalized throughout the training set

Conclusion

- The neuron model is introduced
- The general scheme of construction of fully-connected neural networks is considered
- A mathematical formulation of the training problem for the weights of a fully-connected network is introduced
- A general scheme of the back propagation method for training network parameters is considered
- Further, an example of using deep fully-connected networks to solve the computer vision problem with the Intel® neon™ Framework is considered

Literature

- Haykin S. Neural Networks: A Comprehensive Foundation. – Prentice Hall PTR Upper Saddle River, NJ, USA. – 1998.
- Osovsky S. Neural networks for information processing. – 2002.
- Goodfellow I., Bengio Y., Courville A. Deep Learning. – MIT Press. – 2016. – [http://www.deeplearningbook.org].

Authors

- **Kustikova Valentina Dmitrievna**
  Phd, lecturer, department of Computer software and supercomputer technologies, Institute of Information Technologies, Mathematics and Mechanics, Nizhny Novgorod State University
  valentina.kustikova@itmm.unn.ru
- **Zhiltsov Maxim Sergeevich**
  master of the 1st year training, Institute of Information Technology, Mathematics and Mechanics, Nizhny Novgorod State University
  zhiltsov.max35@gmail.com
- **Zolotykh Nikolai Yurievich**
  Dr., Prof., department of Algebra, geometry and discrete mathematics, Institute of Information Technologies, Mathematics and Mechanics, Nizhny Novgorod State University
  nikolai.zolotykh@itmm.unn.ru