Selected topics in Artificial Intelligence

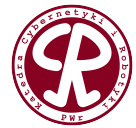
AI introduction

Wojciech Domski

Chair of Cybernetics and Robotics,
Wrocław University of Science and Technology

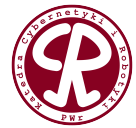
Presentation compiled for taking notes during lecture

Wrocław University
of Science and Technology



Outline

Wrocław University
of Science and Technology



Artificial intelligence (1/2)

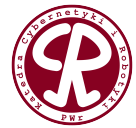
What is artificial intelligence?

Wrocław University
of Science and Technology

Artificial intelligence (2/2)

Artificial intelligence is an ability of a computer to act upon data which has not yet been seen.

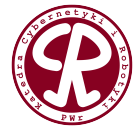
Wrocław University of Science and Technology



Example applications (1/1)

- predicting stock prices,
- recognizing people,
- automated medical diagnosis,
- creating paintings and art,
- speech synthesis,
- and more.

Wrocław University
of Science and Technology



Types of machine learning systems (1/3)

There are many techniques which allow to classify types of machine learning systems given the way they are trained.

Supervised learning

Supervised learning is about discovering a relation between given inputs (labels) *x* and outputs (e.g. classes) y. Through the series of iterations the algorithm learns a relationship between given inputs and outputs, thus for unknown inputs it should predict outputs. Generally, supervised learning can be divided into classification and regression problems where we can find following algorithms: Naive Bayes Classifier, SVM (Supported Vector Machine), Decision tree, Linear regression, Neural Networks.

Types of machine learning systems (2/3)

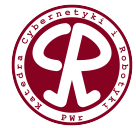
Unsupervised learning

A group of algorithms which operate on data sets without any additional information. The goal of unsupervised learning is to find relations between members of a given data set. For example, clustering or PCA (Principal Component Analysis).

Semi-supervised learning.

Is a combination of a supervised learning and unsupervised learning. Two types of data are given, labelled data as for supervised learning and unlabelled data as for unsupervised learning. Example of such technique are generative models.

Wrocław University of Science and Technology



Types of machine learning systems (3/3)

Reinforcement learning

In reinforcement learning emphasis is put upon action and result. The agent is performing an action in a given environment, as feedback the agent receives a reward. This way the agent can synthesise a policy which maximises the received reward. Example techniques are policy gradient or Q-networks.

Other types

- Batch learning
- Online learning
- Model-based learning

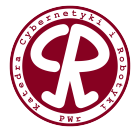
Machine learning problems

- Insufficient amount of training data.
- Unrepresentative data (wrong data proportion).
- Bad quality data.
- Redundant features.
- Over fitting of the trained model.
- Under fitting of the trained model.

Machine learning pipeline (1/13)

1. Goal analysis.
2. Data analysis.
3. Data acquisition.
4. Data visualization and verification.
5. Data augmentation.
6. Data adjustment.
7. Algorithm selection.
8. Learning process.
9. Validation.
10. Tuning.
11. Deployment.

Wrocław University of Science and Technology

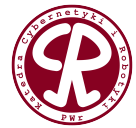


Machine learning pipeline (2/13)

Goal analysis

This is the first step in machine learning model creation. Also, this one is the most important one. The problem at hand has to be deeply analysed. The goal itself has to be defined and understood. During this process once can answer following question. Hasn't what we want to achieve been already solved in any other way. Usually, there already exist a solution or at least partial solution. Example for this could be transfer learning.

Wrocław University of Science and Technology

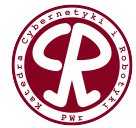


Machine learning pipeline (3/13)

Data analysis

Depending on a problem nature the data can already be present. In order to successfully start model training the provided data has to be carefully selected and analysed. It is required to learn the data we are going to work with. If the data is of numerical nature, maybe it is required to e.g. check if all numerical values are in the range they are expected to be.

Wrocław University
of Science and Technology

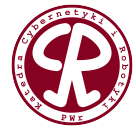


Machine learning pipeline (4/13)

Data acquisition

Sometimes only an idea about the data is available but not the samples themselves. In this case there are two paths to gather necessary samples. One way is to create a dataset by hand. This approach is time consuming, however depending on problem itself it can be an only solution. On the other hand, if the problem relates to a domain which is already researched, e.g. detection of a specific object open datasets are already available.

Wrocław University of Science and Technology

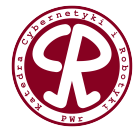


Machine learning pipeline (5/13)

Example of data sources

- https://www.kaggle.com/
- https://cocodataset.org

Wrocław University of Science and Technology

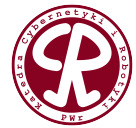


Machine learning pipeline (6/13)

Data visualization and verification

The goal of this step is to asses collected data and verify if it is suitable for the training provided that the data is of high quality. The aim of data visualization for numerical data type usually allows to discover some distinct correlation or immediately allows to find some relations thanks to which the task at hand can be simplified.

Wrocław University of Science and Technology

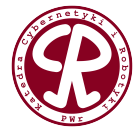


Machine learning pipeline (7/13)

Data augmentation

The process of training a model usually requires significant amount of data. If the amount of the data is not sufficient artificial techniques of enlarging data set are require. Adding noise to data that already exists or rotating an image are only examples how original dataset can be enriched. However, the technique has to be adequate allowing to create meaningful new data set.

Wrocław University of Science and Technology

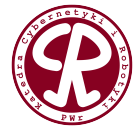


Machine learning pipeline (8/13)

Data adjustment

Final analysis of the dataset is required to be carried out. This allows to increase accuracy of trained model since corrupted or low quality data can be discarded before it is used in the pipeline.

Wrocław University of Science and Technology

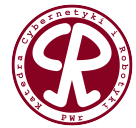


Machine learning pipeline (9/13)

Algorithm selection

Based on the data analysis additional information about it can be revealed. It might turn out that using a fairly simple approach would be efficient. If something like that is possible it allows to cut down on costs and save time.

Wrocław University of Science and Technology

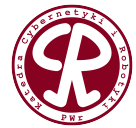


Machine learning pipeline (10/13)

Learning process

After method has been selected the training of a model can be started. Depending on the approach which has been chosen a variety of learning techniques can be applied.
For example, the same neural network can be trained using different optimizers allowing to reach different results.

Wrocław University
of Science and Technology

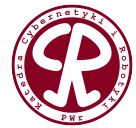


Machine learning pipeline (11/13)

Validation

This is the most crucial part of the process. It allows to check if the trained model can be used in real life. The purpose of machine learning or artificial intelligence methods is to draw conclusions based on previous experiences in a new situation. This can be extrapolated to the validation stage where the accuracy or reliability of the model has to be assessed based on data set which was not used during training stage.

Wrocław University of Science and Technology

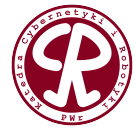


Machine learning pipeline (12/13)

Tuning

If overall performance of the model is promising there is always room for improvement. Tuning hyper parameters can increase performance without investing in enlarging dataset.

Wrocław University of Science and Technology

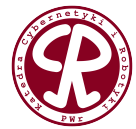


Machine learning pipeline (13/13)

Deployment

As the last stage deployment is considered to be semi-automatic. The model can be utilized in pipeline prepared specifically for a given task like detecting an object of interest in a given image.
Also, this stage might consider model optimization or model adjustment. The trained model might be used on a platform with limited resources then model translation is required to e.g. adjust data type.

Wrocław University of Science and Technology



Linear regression (1/2)

Let's consider model given by equation

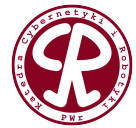
$$\hat{y} = w_0 + w_1 x_1 + w_2 x_2 + \ldots + w_k x_k \tag{1}$$

or equivalent matrix form

$$\hat{y} = W^T X. \tag{2}$$

X can be considered features while y is considered to be labels and W are weights.
To estimate how well the data fits the model MSE (mean squared error can be used).

Wrocław University of Science and Technology



Linear regression (2/2)

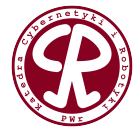
To calculate MSE following formula can be used

$$MSE = \frac{1}{m}\sum_{i=1}^{m}(\hat{y}_i - y_i) = \frac{1}{m}\sum_{i=1}^{m}(WX_i - y_i) \tag{3}$$

where m is a number of samples, thus dimension of features X. The goal is to find vector W which minimizes MSE. In order to minimize MSE in respect to W normal equation can be used

$$\hat{W} = (X^T X)^{-1} X^T y. \tag{4}$$

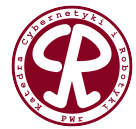
Wrocław University of Science and Technology



Gradient descent (1/2)

Gradient descent is a generic type of algorithm which allows to find an optimal solution for a given problem. Gradient descent is based on iterative computation of an approximated solution which step-by-step is being improved.
The solution starts with e.g. random initialization of approximated solution which is then being corrected with every step of the algorithm.
To compute direction in which to update solution gradient has to be calculated.

Wrocław University
of Science and Technology

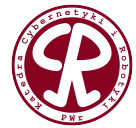


Gradient descent (2/2)

Possible drawbacks of straight forward implementation of a gradient descent algorithms are:

- slow convergence,
- over shooting of a solution,
- moving away from the optimal solution,
- getting stuck in local minima,
- prone to plateau.

Wrocław University of Science and Technology



Batch gradient descent (1/4)

To minimize MSE a gradient can be calculated in reference to W weights. Partial derivatives of cost function (MSE) in respect to weights can be computed as

$$\frac{\delta}{\delta W_i} = \frac{2}{m}\sum_{i=1}^{m}(W^T X^i - y^i)X_j^i. \tag{5}$$

Matrix equation can be written as

$$\bigtriangledown MSE = \frac{2}{m}\sum_{i=1}^{m} X^T(XW - y). \tag{6}$$

To determine the correction the new estimation of weights can be computed as

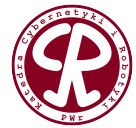
$$W_{i+1} = W_i - \eta \bigtriangledown MSE \tag{7}$$

Wrocław University of Science and Technology

Batch gradient descent (2/4)

where η is considered to be a learning rate which is usually kept small.
What is the drawback of Batch gradient descent?

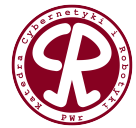
Wrocław University
of Science and Technology



Batch gradient descent (3/4)

In every step gradient is being evaluated for full set of data. This can be considered to be ineffective and time consuming when big data sets are used.

Wrocław University of Science and Technology



Batch gradient descent (4/4)

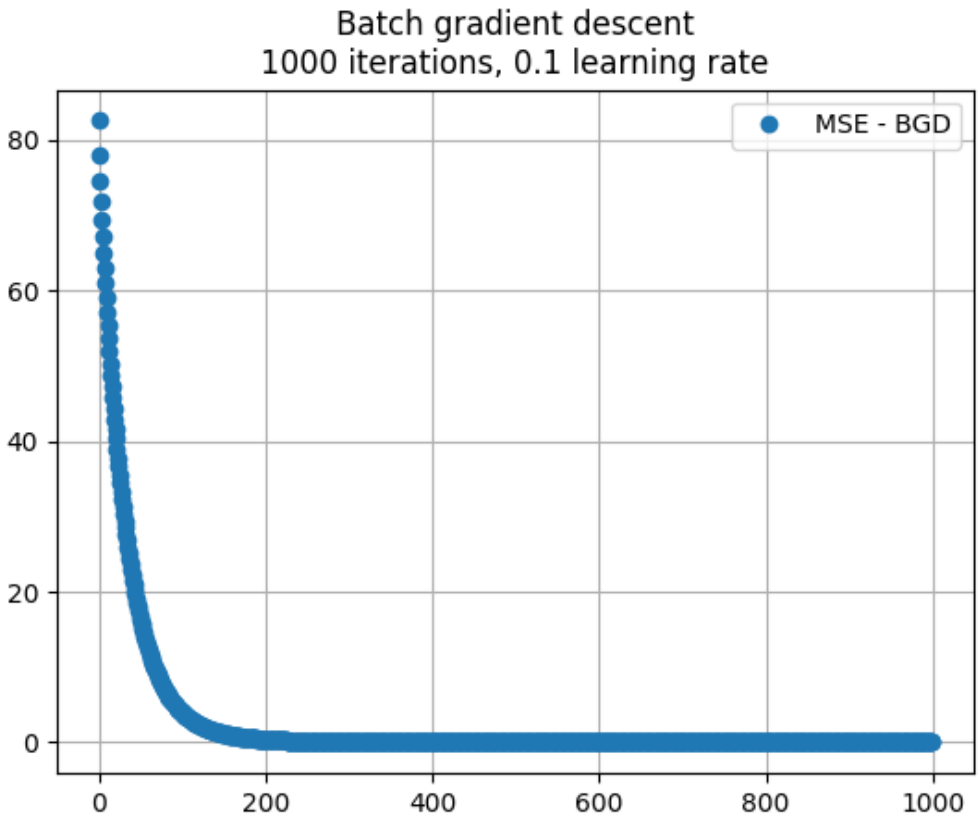


Figure: MSE per iteration

Stochastic gradient descent (1/3)

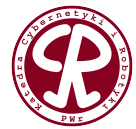
A variation to previously introduced Batch gradient descent is Stochastic gradient descent. It allows to calculate gradient, thus change, for each sample from the data set. Additionally, the samples which are used to calculate the gradient are drawn in random manner. The gradient can be calculated as

$$\bigtriangledown MSE = 2X^T(X_i W - y_i). \tag{8}$$

Estimation of new weights is identical to previous approach

$$W_{i+1} = W_i - \eta \bigtriangledown MSE. \tag{9}$$

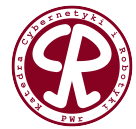
Wrocław University of Science and Technology



Stochastic gradient descent (2/3)

What is the drawback of Stochastic gradient descent?
In contrast to Batch gradient descent Stochastic gradient descent uses a random fetching from the data. The computation complexity is similar but instead of computing a single formula for a vector of a dimension m, m formulas for each sample are computed instead. Moreover, this approach is harder to parallelise given the dependency between calculation steps.

Wrocław University
of Science and Technology



Stochastic gradient descent (3/3)

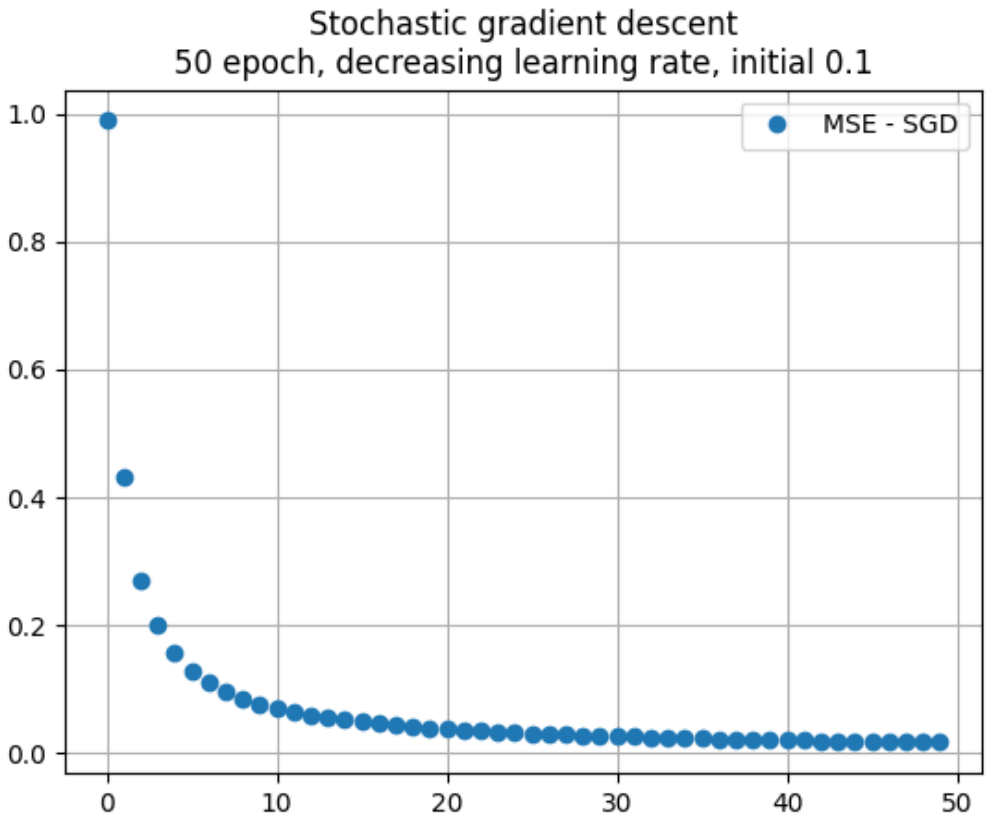


Figure: MSE per epoch

Mini-batch gradient descent (1/3)

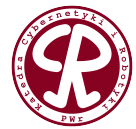
Mini-batch gradient descent is a middle ground between Batch gradient descent and Stochastic gradient descent.
The MSE gradient is calculated in respect to following relation

$$\bigtriangledown MSE = 2X^T(\tilde{X}_i^j W - \tilde{y}_i^j). \tag{10}$$

As it can be seen, now a subset of features $\tilde{X}$ as well as subset of labels $\tilde{y}$ is being taken.
Estimation of new weights is identical to previous approaches

$$W_{i+1} = W_i - \eta \bigtriangledown MSE. \tag{11}$$

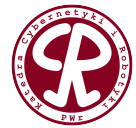
Wrocław University
of Science and Technology



Mini-batch gradient descent (2/3)

Mini-batch GD is similar to Stochastic GD through a randomized sequence of features and labels. The difference is that with each iteration of the MBGD not a single sample is drawn from set but a handful number of samples are computed in a single step. Therefore, MBGD combines BGD and SGD.

Wrocław University of Science and Technology



Mini-batch gradient descent (3/3)

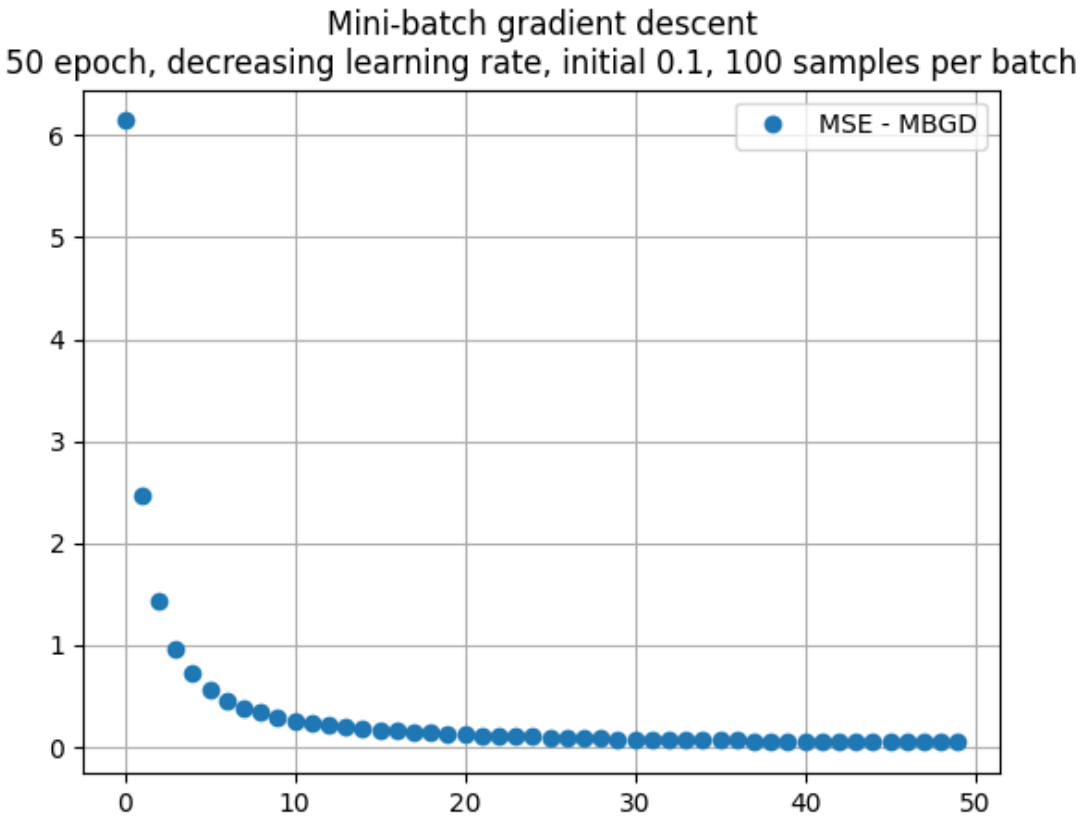


Figure: MSE per epoch

Gradient descent comparison

Calculations were performed for a set of 1000 samples.
Vector of weights was $W = [4, 5, 2, -8]^T$.

Method	MSE	$\hat{W}_0$	$\hat{W}_1$	$\hat{W}_2$	$\hat{W}_3$
		$e_{\hat{W}_0}$	$e_{\hat{W}_1}$	$e_{\hat{W}_2}$	$e_{\hat{W}_3}$
LR:	0.1234	4.1566	5.0747	1.8891	-8.2847
		-0.1566	-0.0747	0.1109	0.2847
BGD:	0.1233	4.1565	5.0747	1.8891	-8.2846
		-0.1565	-0.0747	0.1109	0.2846
SGD:	0.0731	4.1060	5.0827	1.9188	-8.2200
		-0.1060	-0.0827	0.0812	0.2200
MBGD:	0.0132	4.0206	5.0622	1.9744	-8.0910
		-0.0206	-0.0622	0.0256	0.0910

Wrocław University
of Science and Technology

Learning rate

How learning rate influences convergence rate?

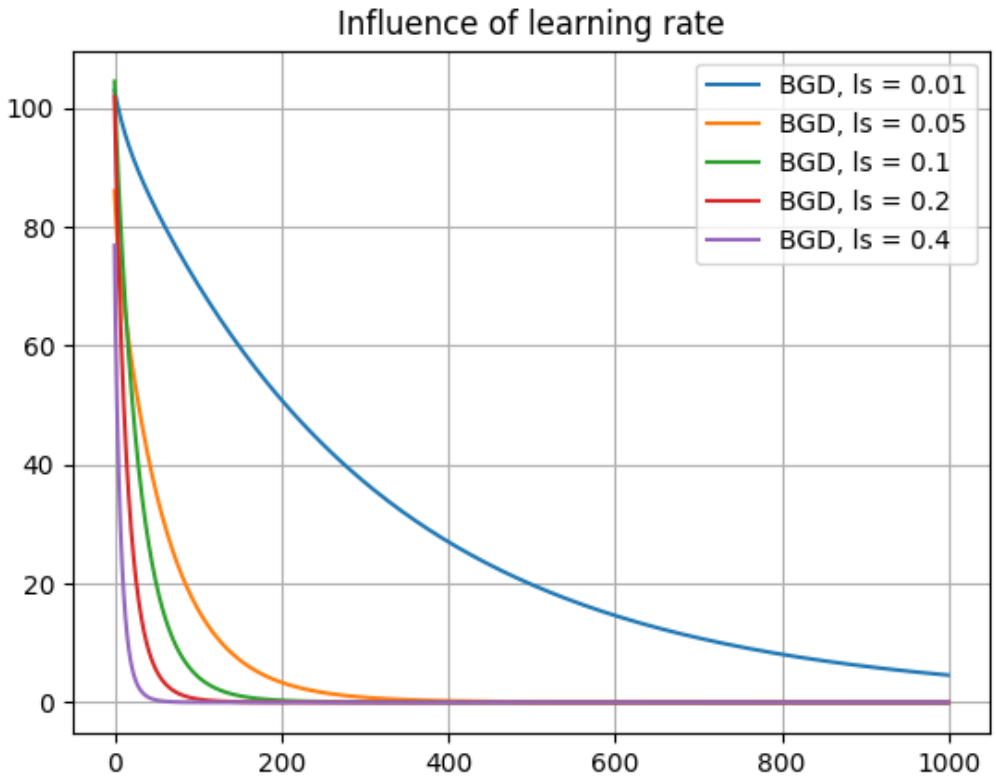
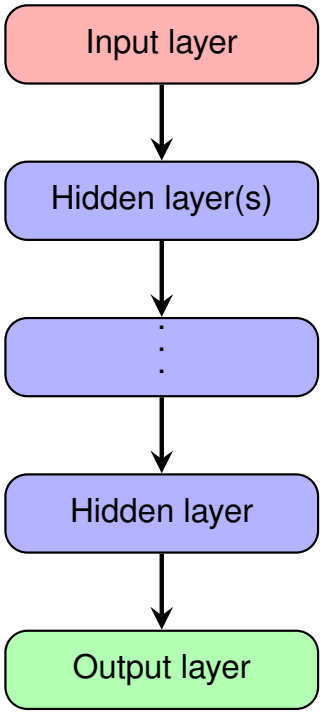


Figure: Learning rate influence on convergence

Neural network

A neural network consists of layers.

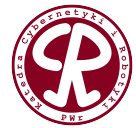


Input layer responsible for formatting and passing the data into internal layers.

One or more hidden layers. Responsible for processing the data.

Output layer which provides direct access to estimated prediction of a neural network.

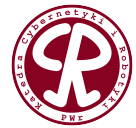
Wrocław University of Science and Technology



Forward propagation (1/2)

Forward propagation is a process of passing the computed values between layers in order from the input layer to the output layer. In other words, output of the first layer is the input of the second layer.
The output of the last layer (output layer) can be treated as the prediction of the neural network. Knowing the real value (label) and the prediction the difference between these two values can be calculated giving us the error.
Error is used as feedback to adjust weights in the layers.

Wrocław University
of Science and Technology



Forward propagation (2/2)

When forward propagation is being considered for a single layer it can be written as

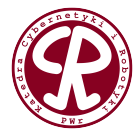
$$y_i = \sum_{i=1}^{n} x_i w_{ij} + b_i \tag{12}$$

where i is the number of inputs ($\dim X$) and j is the number of outputs ($\dim Y$). Above can be expressed as matrix equation

$$Y = XW + B. \tag{13}$$

X are features, W are weights, Y are predictions while B is bias. Without bias it would be impossible to output non-zero value for zero features.

Wrocław University of Science and Technology



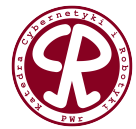
Optimizer

Optimizer is responsible for applying a slight change to the weights in order to minimize the error (difference between prediction and the real value).
To calculate correction, how to adjust weights in network, optimizer e.g. such as gradient descent can be used. It allows to apply a correction to the weights with already introduced formula

$$W_{i+1} = W_i - \eta \bigtriangledown \tag{14}$$

where $\bigtriangledown$ is a derivative of error E in reference to weights yielding $\bigtriangledown = \frac{\delta E}{\delta W}$

Wrocław University of Science and Technology



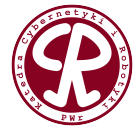
Backward propagation (1/10)

Backward propagation is a reverse process to forward propagation.

$$\frac{\delta E}{\delta Y} \rightarrow layer \rightarrow \frac{\delta E}{\delta X} \tag{15}$$

The order of the calculation is backward from the output to the input of a layer, thus the whole network. This relation describes that if one layer's input is second layer's output meaning that $\frac{\delta E}{\delta X}$ for one layer is $\frac{\delta E}{\delta Y}$ for previous layer.

Wrocław University of Science and Technology



Backward propagation (2/10)

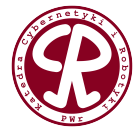
To calculate $\frac{\delta E}{\delta W}$ chain rule can be used. It is given as

$$\frac{\delta E}{\delta W} = \sum_{i=1}^{n} \frac{\delta E}{\delta y_i} \frac{\delta y_i}{\delta W} \tag{16}$$

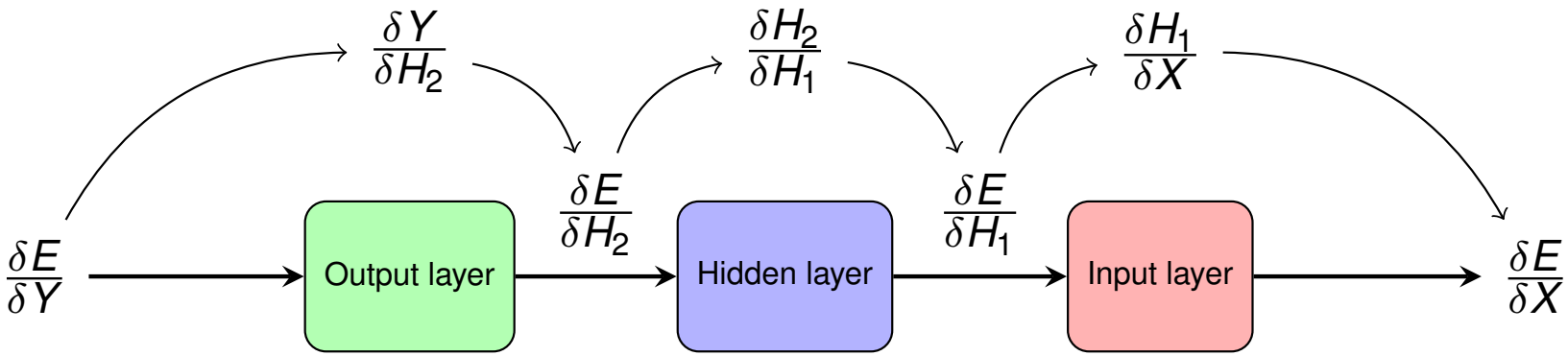
The derivative $\frac{\delta E}{\delta y_i}$ is know since it is the output of a given layer.

$$\frac{\delta E}{\delta X_i} = \sum_{i=1}^{n} \frac{\delta E}{\delta y_i} \frac{\delta y_i}{\delta X_i} \tag{17}$$

Wrocław University
of Science and Technology



Backward propagation (3/10)



Backward propagation (4/10)

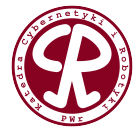
To compute back propagation it is needed to have information about error change in relation to the weights, thus $\frac{\delta E}{\delta W}$.

$$\frac{\delta E}{\delta W} = \begin{bmatrix} \frac{\delta E}{\delta w_{11}} & \cdots & \frac{\delta E}{\delta w_{1j}} \\ \vdots & \ddots & \vdots \\ \frac{\delta E}{\delta w_{i1}} & \cdots & \frac{\delta E}{\delta w_{ij}} \end{bmatrix} \tag{18}$$

Let's use chain rule given with (16).

$$\frac{\delta E}{\delta W} = \sum_{i=1}^{n} \frac{\delta E}{\delta y_i} \frac{\delta y_i}{\delta W} \tag{19}$$

Wrocław University of Science and Technology



Backward propagation (5/10)

then

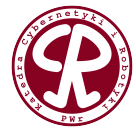
$$\frac{\delta E}{\delta w_{ij}} = \sum_{k=1}^{n} \frac{\delta E}{\delta y_k} \frac{\delta y_k}{\delta w_{ij}} \tag{20}$$

$$= \frac{\delta E}{\delta y_1} \frac{\delta y_1}{\delta w_{ij}} + \frac{\delta E}{\delta y_2} \frac{\delta y_2}{\delta w_{ij}} + \ldots + \frac{\delta E}{\delta y_n} \frac{\delta y_n}{\delta w_{ij}} \tag{21}$$

$$= \frac{\delta E}{\delta y_1} x_1 + \frac{\delta E}{\delta y_2} x_2 + \ldots + \frac{\delta E}{\delta y_n} x_i \tag{22}$$

$$= \frac{\delta E}{\delta y_i} x_i. \tag{23}$$

Wrocław University of Science and Technology



Backward propagation (6/10)

Further on it can be written as a matrix equation

$$\frac{\delta E}{\delta W} = \begin{bmatrix} \frac{\delta E}{\delta y_1} x_1 & \cdots & \frac{\delta E}{\delta y_j} x_1 \\ \vdots & \ddots & \vdots \\ \frac{\delta E}{\delta y_1} x_i & \cdots & \frac{\delta E}{\delta y_j} x_i \end{bmatrix} \tag{24}$$

$$= \begin{bmatrix} x_1 \\ \vdots \\ x_i \end{bmatrix} \begin{bmatrix} \frac{\delta E}{\delta y_1} & \cdots & \frac{\delta E}{\delta y_j} \end{bmatrix} \tag{25}$$

$$= X^T \frac{\delta E}{\delta Y} \tag{26}$$

To calculate derivative of error in relation to bias it can be written

$$\frac{\delta E}{\delta B} = \begin{bmatrix} \frac{\delta E}{\delta b_1} & \frac{\delta E}{\delta b_2} & \cdots & \frac{\delta E}{\delta b_j} \end{bmatrix}. \tag{27}$$

Wrocław University of Science and Technology

Backward propagation (7/10)

Let's consider above with relation to every element of bias vector, thus

$$\frac{\delta E}{\delta b_j} = \frac{\delta E}{\delta b_j}\frac{\delta y_j}{\delta y_j} = \frac{\delta E}{\delta y_j}\frac{\delta y_j}{\delta b_j} \tag{28}$$

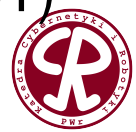
$$= \frac{\delta E}{\delta y_1}\frac{\delta y_1}{\delta b_1} + \ldots + \frac{\delta E}{\delta y_j}\frac{\delta y_j}{\delta b_j} \tag{29}$$

$$= \frac{\delta E}{\delta y_1} + \ldots + \frac{\delta E}{\delta y_j} = \frac{\delta E}{\delta y_j} \tag{30}$$

which gives

$$\frac{\delta E}{\delta B} = \left[\begin{array}{cccc} \frac{\delta E}{\delta y_1} & \frac{\delta E}{\delta y_2} & \cdots & \frac{\delta E}{\delta y_j} \end{array} \right] = \frac{\delta E}{\delta Y}. \tag{31}$$

Wrocław University of Science and Technology



Backward propagation (8/10)

Now, let's calculate derivative of error in reference to input.

$$\frac{\delta E}{\delta X} = \left[\begin{array}{cccc} \frac{\delta E}{\delta x_1} & \frac{\delta E}{\delta x_2} & \cdots & \frac{\delta E}{\delta x_i} \end{array} \right]. \tag{32}$$

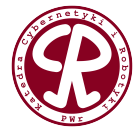
Once again we multiply the equation with $1 = \frac{y_j}{y_j}$ and we get

$$\frac{\delta E}{\delta X} = \frac{\delta E}{\delta x_i}\frac{\delta y_j}{\delta y_j} = \frac{\delta E}{\delta y_j}\frac{\delta y_j}{\delta x_j} \tag{33}$$

$$= \frac{\delta E}{\delta y_1}\frac{\delta y_1}{\delta x_1} + \ldots + \frac{\delta E}{\delta y_j}\frac{\delta y_j}{\delta x_i} \tag{34}$$

$$= \frac{\delta E}{\delta y_1} w_{i1} + \ldots + \frac{\delta E}{\delta y_j} w_{ij} = \frac{\delta E}{\delta y_j} w_{ij} \tag{35}$$

Wrocław University of Science and Technology



Backward propagation (9/10)

Let's expand $\frac{\delta E}{\delta X}$, thus

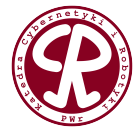
$$\frac{\delta E}{\delta X} = \left[\left(\frac{\delta E}{\delta y_1} w_{11} + \ldots + \frac{\delta E}{\delta y_j} w_{1j}\right) \quad \ldots \quad \left(\frac{\delta E}{\delta y_1} w_{i1} + \ldots + \frac{\delta E}{\delta y_j} w_{ij}\right) \right] \tag{36}$$

$$= \left[\frac{\delta E}{\delta y_1} \quad \ldots \quad \frac{\delta E}{\delta y_j} \right] \begin{bmatrix} w_{11} & \cdots & w_{i1} \\ \vdots & \ddots & \vdots \\ w_{1j} & \cdots & w_{ij} \end{bmatrix} \tag{37}$$

$$= \frac{\delta E}{\delta Y} W^T \tag{38}$$

With three formulas

Wrocław University of Science and Technology



Backward propagation (10/10)

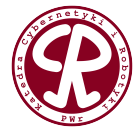
$$\frac{\delta E}{\delta X} = \frac{\delta E}{\delta Y} W^T \tag{39}$$

$$\frac{\delta E}{\delta W} = X^T \frac{\delta E}{\delta Y} \tag{40}$$

$$\frac{\delta E}{\delta B} = \frac{\delta E}{\delta Y} \tag{41}$$

back propagation for the whole network can be calculated. $\frac{\delta E}{\delta W}$ is used to update weights of a layer, $\frac{\delta E}{\delta B}$ is used to update bias. $\frac{\delta E}{\delta X}$ is used as $\frac{\delta E}{\delta Y}$ for previous layer. $\frac{\delta E}{\delta Y}$ is fed to the last layer (first in back propagation computation) as error computed at the end of forward propagation.

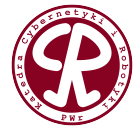
Wrocław University of Science and Technology



Activation layer (1/1)

The purpose of activation layer is to provide non-linearity to the model. Thanks to it the model can learn non-linear relationships and not only the linear ones.
As previously. Forward and backward propagation for activation layer has to be introduced.

Wrocław University
of Science and Technology



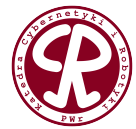
Forward propagation (1/1)

Forward propagation for a activation layer is nothing more the function (non-linear one) applied to the input, thus

$$Y = f_a(X). \tag{42}$$

The same function is applied to each input which is then propagated to the output of the activation layer. The dimension of inputs is equal to the dimension of outputs for a given activation layer.

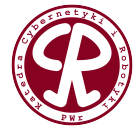
Wrocław University of Science and Technology



Backward propagation (1/2)

In order to calculate backward propagation the relation $\frac{\delta E}{\delta X}$, so the relation of the error to the input. As in previous case we are given $\frac{\delta E}{\delta Y}$.
As it can be seen the activation layer has no weights W, thus this relation $\frac{\delta E}{\delta W}$ is not present.

Wrocław University
of Science and Technology



Backward propagation (2/2)

$$
\begin{aligned}
\frac{\delta E}{\delta X} &= \left[\frac{\delta E}{\delta x_1} \cdots \frac{\delta E}{\delta x_i}\right] \\
&= \left[\frac{\delta E}{\delta y_1}\frac{\delta y_1}{\delta x_1} \cdots \frac{\delta E}{\delta y_i}\frac{\delta y_i}{\delta x_i}\right] \\
&= \left[\frac{\delta E}{\delta y_1} f_a'(x_1) \cdots \frac{\delta E}{\delta y_i} f_a'(x_i)\right] \\
&= \left[\frac{\delta E}{\delta y_1} \cdots \frac{\delta E}{\delta y_i}\right] \cdot \left[f_a'(x_1) \cdots f_a'(x_i)\right] \\
&= \frac{\delta E}{\delta Y} \cdot f_a'(X) \qquad (43)
\end{aligned}
$$

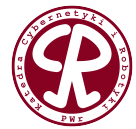
As it can be seen to calculate $\frac{\delta E}{\delta X}$ in addition to $\frac{\delta E}{\delta Y}$ derivative of $f_a'(X)$ has to be known. This derivative is given in analytic form

Loss function (1/2)

In course of backward propagation for the neural network it was assumed that the relation $\frac{\delta E}{\delta Y}$ is given. The error value in relation to the output (predictions) has to be calculated using a metric. One of commonly used metric is a MSE (Mean Squared Error) which is based on calculating square on the difference between what we inserted into the network Y^* and what was predicted Y.
This can be expressed with following relation

$$E = \frac{1}{n} \sum_{i=1}^{n} (y_i^* - y_i)^2 . \tag{44}$$

Wrocław University of Science and Technology



Loss function (2/2)

Given the loss function the $\frac{\delta E}{\delta Y}$ derivative can be calculated

$$\begin{aligned}
\frac{\delta E}{\delta Y} &= \left[\frac{\delta E}{\delta y_1} \cdots \frac{\delta E}{\delta y_i}\right] \\
&= \left[-\frac{2}{n}(y_1^* - y_1) \cdots -\frac{2}{n}(y_1^* - y_1)\right] \\
&= -\frac{2}{n}\left[(y_1^* - y_1) \cdots (y_1^* - y_1)\right] \\
&= -\frac{2}{n}(Y^* - Y) \\
&= \frac{2}{n}(Y - Y^*)
\end{aligned} \tag{45}$$

Wrocław University of Science and Technology