VSB TECHNICAL UNIVERSITY OF OSTRAVA | FACULTY OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE | DEPARTMENT OF COMPUTER SCIENCE

Deep Learning

Artificial Neural Networks

Jan Platoš, Radek Svoboda

February 23, 2022

Department of Computer Science
Faculty of Electrical Engineering and Computer Science
VŠB - Technical University of Ostrava

Artificial Neural Networks

Artificial Neural Networks

- Artificial Neural Networks (ANN) are models of the human nervous system.

Human nervous system

- The system is composed of cells, called neurons.
- The neurons are connected to other neurons using synapses.
- The strength of the synapses is affected by learning (external stimuli).

Artificial Neural Networks

- The system is composed of nodes, called neurons.
- These neurons are units of computation that:
  - Receive the inputs from other neurons.
  - Processes these inputs (computes).
  - Set its output.
- The computation process is affected by the input weights and activation function.
- The weights are analogous to the strength of the synapse.
- The weights are affected by the learning process.

Artificial Neural Networks

- The neural networks ability to learn is based on the architecture of the network.
  - Single-layer neural network.
  - Multi-layer neural network.
  - Recurrent neural networks.
  - Kohonen Maps (Self Organized Maps).
  - Convolution networks.
  - Deep neural networks.
  - ...
  -
- The learning is done by presenting the test instances to the network and correction of the output according to the expected output by weight adjusting.

Artificial Neural Networks - Single-layer Neural Network

- The basic architecture of neural network.
- The structure has two layers.
  - The input layer has one node for each input attribute.
  - The input node only transmit the input value to the output node.
  - The connection between input and output nodes are weighted.
  - The output layer consist of one output neuron.
  - The output neuron computes the output value.
- The class labels are from the set of $\{-1, +1\}$.

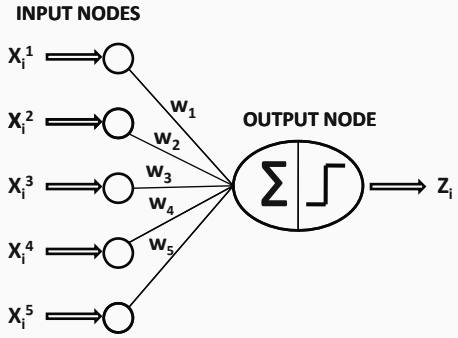


Figure 1: The Perceptron

Artificial Neural Networks - Single-layer Neural Network

- The weighted inputs are transformed into output value.
- The value in drawn from the set $\{-1, +1\}$.
- The value may be interpreted as the perceptron prediction of the class variable.
- The weights $W = \{w_1, \ldots, w_d\}$ are modified when the predicted output does not match expected value.

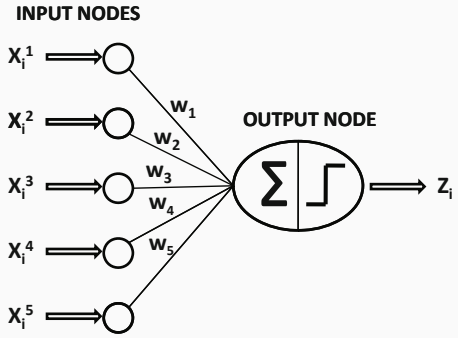


Figure 2: The Perceptron

Artificial Neural Networks - Single-layer Neural Network

- The function learned by the perceptron is referred as *activation function*.
- The function is usually signed linear function (e.g. weighted sum).
- The $W = \{w_1, \ldots, w_d\}$ are the weights for the connections of d different inputs to the output neuron.
- The d is also the dimensionality of the data.
- The b is the bias associated with the activation function.
- The output $z_i \in \{-1, +1\}$ is for the data record $\overline{X_i} = (x_i^1, \ldots, x_i^d)$ computed as follows:

$$z_i = sign\left\{\sum_{j=1}^{d} w_j x_i^j + b\right\} = sign\left\{\overline{W} \cdot \overline{X_i} + b\right\}$$

Artificial Neural Networks - Single-layer Neural Network

$$z_i = sign\left\{\sum_{j=1}^{d} w_j x_i^j + b\right\} = sign\left\{\overline{W} \cdot \overline{X_i} + b\right\}$$

- The difference between the prediction of the class value z_i and the real class value y_i is $(y_i - z_i) \in \{-2, 0, 2\}$.
- The result is 0 when the prediction and reality is the same.
- The weight vector $\overline{W}$ and bias b need to be updated, based on the error $(y_i - z_i)$.
- The learning process is iterative.
- The weight update rule for i-th input point $\overline{X_i}$ in t-th iteration is as follows:

$$\overline{W}^{t+1} = \overline{W}^{t} + \eta(y_i - z_i)\overline{X_i}$$

- The η is the learning rate that regulate the learning speed of the network.
- Each cycle per input points in the learning phase is referred as an *epoch*.

Artificial Neural Networks - Single-layer Neural Network

$$\overline{W}^{t+1} = \overline{W}^{t} + \eta(y_i - z_i)\overline{X_i}$$

- The incremental term $(y_i - z_i)\overline{X_i}$ is the approximation of the negative of the gradient of the least=squares prediction error $(y_i - z_i)^2 = (y_i - sign(\overline{W} \cdot \overline{X_i} - b))^2$
- The update is performed on a tuple-by-tuple basis not a global over whole dataset.
- The perceptron may be considered a modified version of a gradient descent method that minimizes the squared error of prediction.
- The size of the η affect the speed of the convergence and the quality of the solution.
  - The higher value of η means faster convergence, but suboptimal solution may be found.
  - Lower values of η results in higher-quality solutions with slow convergence.
- In practice, η is decreased systematically with increasing number of epochs performed.
- Higher values at the beginning allows bigger jumps in weight space and lower values later allows precise setting of the weights.

Artificial Neural Networks - Multi-layer Neural Network

- The perceptron, with only one computational neuron produces only a linear model.
- Multi-layer perceptron adds a hidden layer beside the input and output layer.
- The hidden layer itself may consist of different type of topologies (e.g. several layers).
- The output of nodes in one layer feed the inputs of the nodes in the next layer - this behavior is called *feed-forward network*.
- The nodes in one layer are fully connected to the neurons in the previous layer.

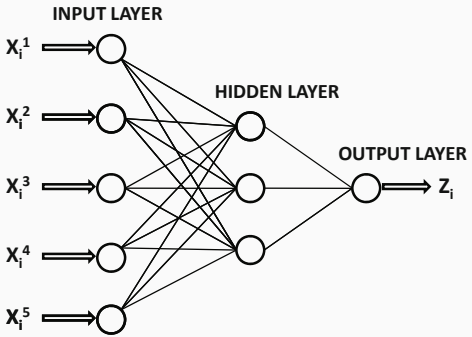


Figure 3: Multi-layer neural network

Artificial Neural Networks - Multi-layer Neural Network

- The topology of the multi-layer feed-forward network is determined automatically.
- The perceptron may be considered as a single-layer feed-forward neural network.
- The number of layers and the number of nodes in each layer have to be determined manually.
- Standard multi-layer network uses only one hidden layer, i.e. this is considered as a two-layer feed forward neural network.
- The activation function is not limited to linear signed weighted sum, other functions such as logistic, sigmoid or hyperbolic tangents are allowed.

Artificial Neural Networks - Multi-layer Neural Network

Sigmoid/Logistic function	$\sigma(x) = \frac{1}{1+e^{-x}}$
TanH	$\tanh(x) = \frac{(e^{x}-e^{-x})}{(e^{x}+e^{-x})}$
ReLU (Rectified linear unit)	$f(x) = \begin{cases} 0 & for x \leq 0 \\ x & for x \geq 0 \end{cases}$
Sinc	$f(x) = \begin{cases} 1 & for x = 0 \\ \frac{sin(x)}{x} & for x \neq 0 \end{cases}$
Gaussian	$f(x) = e^{x^2}$
Softmax	$\sigma(\mathbf{z})_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}}$

Artificial Neural Networks - Multi-layer Neural Network

Learning algorithm

- The learning phase is more complicated than the one in perceptron.
- The biggest problem is the get the error in the hidden layer, because the direct class label is not defined on this level.
- Some kind of *feedback* is required from the nodes in the forward layer to the nodes in earlier layers about the *expected* outputs and corresponding errors.
- This principle is realized in the *back-propagation* algorithm.

Back-propagation algorithm

- *Forward phase:*
  - The input is fed into input neurons.
  - The computed values are propagated using the current weights to the next layers.
  - The final predicted output is compared with the class label and the error is determined.
- *Backward phase:*
  - The main goal is to learn weights in the backward direction by providing the error estimation from later layers to the earlier layers.
  - The estimation in the hidden layer is computed as a function of the error estimate and weight is the layers ahead.
  - The error is estimated again using the gradient method.
  - The process is complicated by the using of non-linear functions n the inner nodes.

Artificial Neural Networks - Multi-layer Neural Network - Learning algorithm

Other learning algorithms:

- Gradient descent
- Stochastic Gradient Descent
  - Momentum
  - Averaging
  - AdaGrad
  - RMSProp
  - Adam
- Newton's method
- Conjugate gradient
- Quasi-Newton method
- Levenberg-Marquardt algorithm

Artificial Neural Networks - Multi-layer Neural Network

- The multi-layer neural network is more powerful than kernel SVM in its ability to capture arbitrary functions.
- It has ability not only to capture decision boundaries of arbitrary shapes, but also non-contiguous class distribution with different decision boundaries in different regions.
- With increasing number of nodes and layers, virtually any function may be approximated.
- **The neural networks are universal function approximators**.
- This generality brings several challenges that have to be dealt with:
  - The design of the topology presents many trade=off challenges for the analyst.
  - Higher number of nodes and layers provides greater generality but also the risk of over-fitting.
  - There is very little guidance provided from the data.
  - The neural network has poor interpretability associated with the classification process.
  - The learning process is very slow and sensitive to the noise.
  - Larger networks has very slow learning process.

Neural Network Model Properties

Neural Network Model Properties

- Topology
  - Number of layers.
  - Number of Neurons in each layer.
  - Type of layers.
- Number of parameters to be optimized.
- Batch size.
- Regularization - L1, L2.
- Dropout.
- Batch Normalization.
- Underfitting / Overfitting

Batch Size

(Stochastic) Gradient Descent - is the process of getting internal parameters of the model (neural network) to fit the data using the calculation of an error gradient or slope of error and ”descent” refers to the moving down along that slope towards some minimum level of error.

Batch - is a set of samples/rows of the data that are processed before the internal state (weights) are updated.

Batch size - is the number of samples/rows in a batch. It heavily affects the efficiency of the algorithm and creates a variants: Batch Gradient Descent, Stochastic Gradient Descent, Minibatch Gradient Descent.

Regularization L1 and L2

Regularization - modifies the target cost function with penalties to improve the ”shape” of the parameters.

L1 regularization also know as Lasso regularization (Least Absolute Shrinkage and Selection Operator). Adds an ”absolute value of magnitude” as a penalty.

$$\sum_{i=1}^{n}\left(y_i - \sum_{j=1}^{p} x_{ij}\beta_j\right)^2 + \lambda \sum_{j=1}^{p} |\beta_j|$$

L2 regularization also know as Ridge regularization. Adds an ”squared magnitude” as a penalty.

$$\sum_{i=1}^{n}\left(y_i - \sum_{j=1}^{p} x_{ij}\beta_j\right)^2 + \lambda \sum_{j=1}^{p} \beta_j^2$$

Dropout

Dropout - is an approach to regularization in neural networks which helps reducing interdependent learning amongst the neurons.

Training Phase - for each hidden layer, for each training sample, for each iteration, ignore (zero out) a random fraction, p, of nodes (and corresponding activations).

Testing Phase - use all activations, but reduce them by a factor p (to account for the missing activations during training).

Batch Normalization

- The output of the each activation functions and its distribution shifts during the training - it is called Internal Covariance Shift.
- Natural solution is to normalize the outputs - it is called standardization (transform it to standard Gaussian distribution).

$$x' = \frac{x - \overline{x}}{\sqrt{\sigma^2 + \epsilon}}$$

- **Batch Normalization** is the normalization of the layer output for each batch.

Overfitting/Underfitting

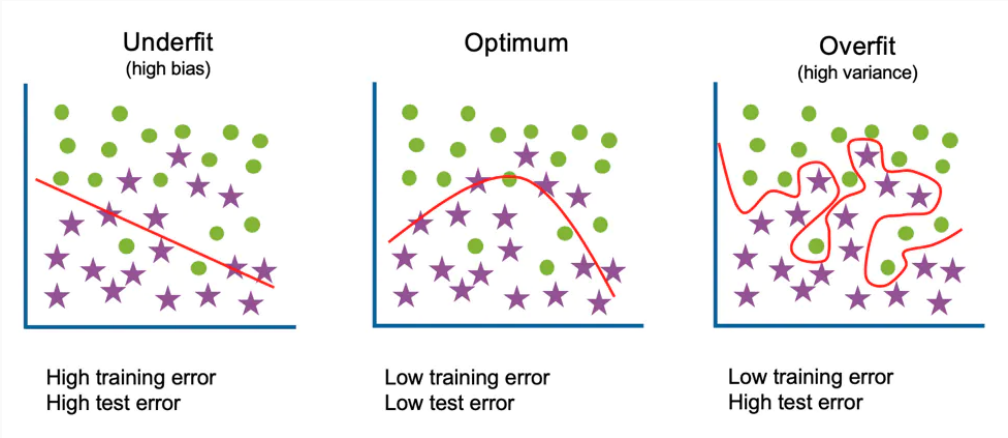
The bias error is an error from erroneous assumptions in the learning algorithm. High bias can cause an algorithm to miss the relevant relations between features and target outputs (underfitting).

The variance is an error from sensitivity to small fluctuations in the training set. High variance can cause an algorithm to model the random noise in the training data, rather than the intended outputs (overfitting).

Underfitting - is inability of the model to learn the properties of a training dataset.

Overfitting - occurs when a model is too specialized to traing dataset and is unable to generalize to the unseen data.

Overfitting/Underfitting



Overfitting/Underfitting - Underfitting avoidance

- Decrease Regularization
  - Regularization usually decrease variance.
  - Reducing regularization allow complexity and variance in the model.
- Extend training time
  - Training lead to acquiring knowledge from data.
  - Early stopping does not allow enough learning.
- Feature Selection
  - Use more complex features.
  - Increase a complexity of a model (more neurons, layers, etc.).

Questions?