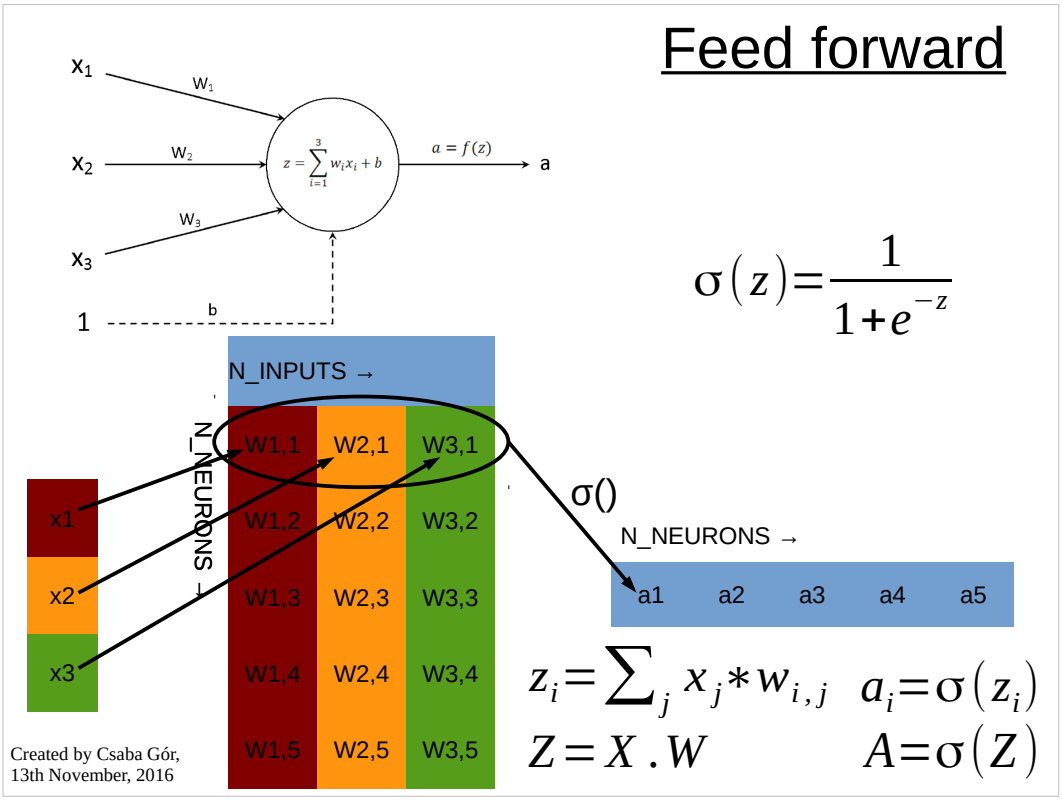
Artificial Neural Network Basics

Created by:
Csaba Gór
2016

This tutorial is quite maths-heavy, but may lacke accuracy in the use of mathematical notations...
(I am no mathematician)



Definition of the feed forward rules in a single neuron and in a single layer

upper-case letters denote matrices/vectors

lower-case letters with indices denote elements of matrices/vectors

X is the input vector

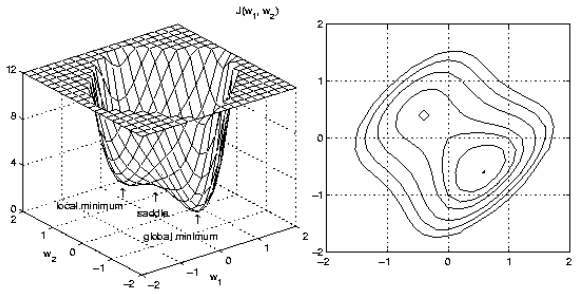
Z is the weighted sum of the inputs, aka. net input. It is basically the output vector before calling the sigmoid

A is actual the output vector

W is the weight matrix

In practice multiple input vectors are passed at once, forming a bacth of inputs, so X is a matrix as well as Z and A. However in theory they are vectors and they are indexed by only a single number. Because of this, uppercase matrix notation is used with only a single number for indexing!

$$MSE(A,Y)=\frac{1}{2N}\sum(A-Y)^2$$



0	0.3
0	0.2
0	0.4
0	0.3
0	0.1
1	0.8
0	0.2
0	0.3
0	0.1
0	0.2

MSE = 0.61 ⟶

Created by Csaba Gór,
13th November, 2016

A: network activations
Y: targets
N: number of examples
MSE: mean squared error cost function

This slide present the mean squared error function, a quadratic function which has a global minimum somewhere in the weight space. This global minimum represent a weight configuration, which produces minimal error on the learning examples.
The goal of the training is to find an optimal set of weights, by minimizing the error function.

Source of the error surface image:
http://neuron-ai.tuke.sk/hudecm/texty+dokumenty/spc/ai/NN_Fundamentals/www.rdt.monash.edu.au/%257Eapp/CSC437nn/Lnts/Figs/Galsg.gif

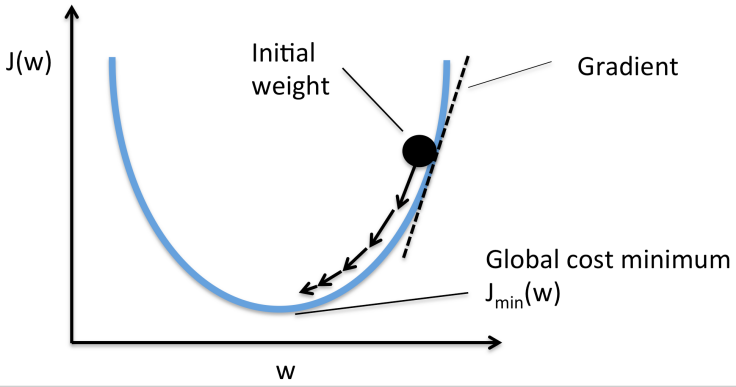
Gradient Descent

$$MSE(A,Y)=\frac{1}{2N}\sum(A-Y)^2$$

$$A=\sum X.W$$

$$W:=W-\frac{d\,MSE}{d\,W}*\eta$$

$$\frac{d\,MSE}{d\,W}=\frac{d\,MSE}{d\,A}\frac{d\,A}{d\,W}$$



Created by Csaba Gór, 13th November, 2016

W: weight matrix
η: eta, the learning rate
MSE: mean squared error cost function

MSE is plotted in a single-weight model, against the value of the weight. It can be seen that the gradient of MSE wrt. w, always points upwards, so if this gradient is subtracted from the weight, a downwards motion can always be achieved, thus minimizing the mean squared error.

This technique is called Gradient Descent. The portion of the gradient subtracted from w is determined by eta, the learning rate.

MSE is not directly a function of w, so the application of the chain rule is used to determine the gradient.

Source of Gradient Descent image:
http://sebastianraschka.com/images/faq/closed-form-vs-gd/ball.png
On the diagram, J() is the MSE cost function.

Output layer error and weight gradients

$$\frac{\partial MSE}{\partial w_{i,j}} = \frac{\partial MSE}{\partial a_i}\frac{\partial a_i}{\partial w_{i,j}}$$

$$\frac{\partial MSE}{\partial a_i} = \frac{\partial}{\partial a_i}\frac{1}{2\mathrm{N}}\sum_k (a_k - y_k)^2 = a_j - y_j = A - Y$$

$$\frac{\partial a_i}{\partial w_{i,j}} = \frac{\partial}{\partial w_{i,j}}\sigma\left(\sum_k x_k * w_{i,k}\right) \quad z_i := \sum_j x_j * w_{i,j}$$

$$\frac{\partial a_i}{\partial w_{i,j}} = \frac{\partial}{\partial z_i}\sigma(z_i) * \frac{\partial}{\partial w_{i,j}}\sum_k x_k * w_{i,k}$$

$$\frac{\partial a_i}{\partial w_{i,j}} = [\sigma(z_i) * (1 - \sigma(z_i))] * [x_k] = X^T . A * (1 - A)$$

Created by Csaba Gór,
13th November, 2016

In the case of multiple (usually at least thousands of) weights, the derivative of MSE must be calculated wrt. every single weight in the model.

Here the gradient is determined wrt. to one single weight by the application of the chain rule, and then the generalized form is presented in matrix-notation.

Backpropagation of errors

$$A_O = \sigma(A_N . W_O)$$

$$A_N = \sigma(A_M . W_N)$$

$$A_M = \sigma(A_L . W_M)$$

$$A_L = \sigma(X . W_L)$$

$$\frac{d\,MSE}{d\,A_O} = A_O - Y$$

$$\frac{d\,A_O}{d\,A_N} = [A_O * (1 - A_O)] . W_O^T$$

$$\frac{d\,A_N}{d\,A_M} = [A_N * (1 - A_N)] . W_N^T$$

$$\frac{d\,A_M}{d\,A_L} = [A_M * (1 - A_M)] . W_M^T$$

Created by Csaba Gór,
13th November, 2016

The output layer's gradient is somewhat more intuitive than the errors of deeper layers.

In order to get the gradient of a hidden layer, the error of the output (not the weight's gradient!) must be propagated back to each layer using the chain rule.

Keep in mind, that the derivatives determined above are members of a big chain-rule derivation (see next slide) and they are meaningless on their own.

Introducing the delta notation

Don't forget this, this is very importan! The previous derivatives are the MEMBERS of the 'big' chain-rule derivation!
They need to be multiplied together in order to get the gradient of MSE with respect to the parameters!
E.g. we are not interested in {dAM / dAL}, we are interested in {dMSE / dAL} which is obtained by the chain rule:

$$\frac{d\,MSE}{d\,A_L} = \frac{d\,MSE}{d\,A_O}\frac{d\,A_O}{d\,A_N}\frac{d\,A_N}{d\,A_M}\frac{d\,A_M}{d\,A_L}$$

$$\frac{d\,MSE}{d\,A_L} = \underbrace{A_O - Y * \sigma'_O}_{\delta_O} . \underbrace{W_O^T * \sigma'_N}_{\delta_N} . \underbrace{W_N^T * \sigma'_M}_{\delta_M} . W_M^T$$

$$\frac{d\,MSE}{d\,A_O} = (A_O - Y) * \sigma'_O \qquad \frac{d\,MSE}{d\,A_N} = (\delta_O . W_O) * \sigma'_N$$

$$\frac{d\,MSE}{d\,A_M} = (\delta_N . W_N) * \sigma'_M \qquad \frac{d\,MSE}{d\,A_L} = (\delta_M . W_M) * \sigma'_L$$

Created by Csaba Gór,
13th November, 2016

Here we introduce the delta notation.

If we think back to the first slides, we introduced the Z as the net input or weighted sum (the values before the application of the sigmoid).

The delta (the layer's error) is actually the gradient of MSE wrt. this weighted sum.

Since every delta is determined from the delta of the layer above, and the output layer's delta is calculated directly from the derivative of the cost function, all the deltas ultimately point to the MSE.

Calculating the weight gradients

The layer's delta is also called the layer's error (it is actually the gradient of MSE with respect to the net sum, Z.

It is not equivalent to the weights' error though!

Why do we need the delta notation? Because in every step of the backprop algorithm, besides the previous layer's delta, we also want to compute the gradient of MSE with respect to the weights, which is also determined using the current layer's delta. For example:

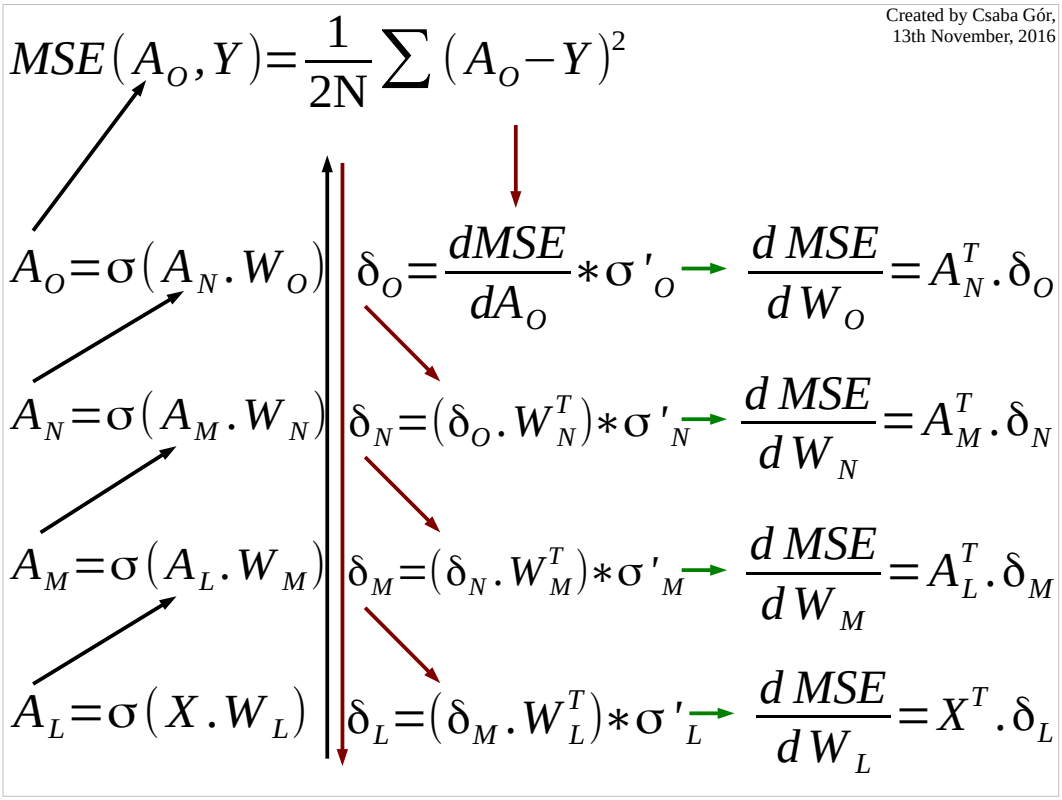
$$\frac{d\,MSE}{d\,W_O} = \frac{d\,MSE}{d\,A_O}\frac{d\,A_O}{d\,W_O} = \overline{A_N^T \cdot (\sigma'_O * \underline{(A_O - Y)})} = A_N^T \cdot \delta_O$$

$$\frac{d\,MSE}{d\,W_O} = A_N^T \cdot \delta_O \qquad \frac{d\,MSE}{d\,W_N} = A_M^T \cdot \delta_N$$

$$\frac{d\,MSE}{d\,W_M} = A_L^T \cdot \delta_M \qquad \frac{d\,MSE}{d\,W_L} = X^T \cdot \delta_L$$

Created by Csaba Gór,
13th November, 2016

Here we calculate the gradients of MSE wrt. the weight matrices, using the layer deltas and the outputs of the previous layers, which are the inputs of the current layer.



This slide shows the summary of the neural network learning process, with a 5-layer neural network (the input layer is not shown)

On the left, with black arrows, the forward pass can be seen.

The cost is determined by calculating the mean squared error of the network outputs (A_O) and the targets (Y).

In the middle column, the backwards pass can be seen, as the determination of the layer errors (the deltas).

On the right, the calculation of the weight gradients is shown.

See how the delta values are used two times during a backwards pass.

$$A_k = \sigma(A_{k-1} . W_k) \rightarrow \textit{forward pass}$$

$$\delta_k = (\delta_{k+1} . W_k^T) * \sigma'_k \rightarrow \textit{backward pass}$$

$$\frac{d\,MSE}{d\,W_k} = \nabla_w MSE = A_{k-1}^T . \delta_k \rightarrow \textit{weight gradients}$$

$$W_k := W_k - \nabla_w MSE * \frac{\eta}{m} \rightarrow \textit{SGD update rule}$$

$\delta_{k+1} = \frac{d\,MSE}{d\,A_O} * \sigma'_O$ at the output layer.

$A_{k-1} = X$ at the first hidden layer

Created by Csaba Gór,
13th November, 2016

This is a summary using generalized notations.

A_k is a neural network layer.
delta_k is the error of that layer, determined from the error and weight matrix of the layer above and the derivative of the sigmoid function wrt. the net output of the actual layer (Z_k, not shown).
The gradient of MSE wrt. the layer's weights is determined (the nabla notation is there to show the equivalence of it with the Leibnitz notation)
Finally, the update rule is shown.
m is the number of training examples in the mini-batch. The strenght of SGD is this averaging step.
m is incorporated into eta in practice, but the division is under the gradients in theory, not the learning rate. It makes no difference mathematically though.