Module: **Machine Learning and Data-Driven Materials**

Perceptron Neural Network – A practical Case

Batch and Online Learning

Week 5 – Session 2 (UK Time)

Institute for Materials Discovery (IMD)
University College London (UCL)
Dr. Adrian Rubio Solis, Prof. Adham Hashibon
Email: ucqsaru@ucl.ac.uk

Content: Week 7

Tuesday, Thursday :

Neural Networks Model:

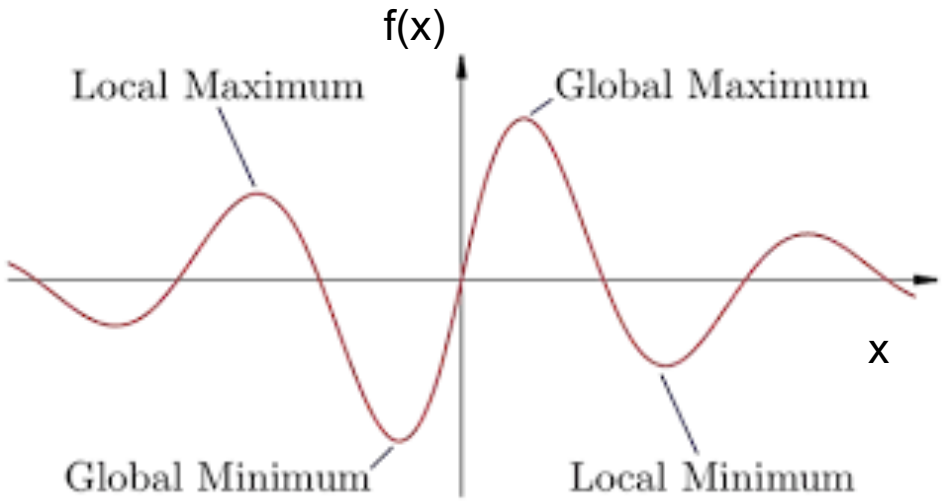
- Neurons and the Brain: Model Representation
- General Pipeline using Extreme Learning Machine – A Learning approach in neural networks
- Regularisation

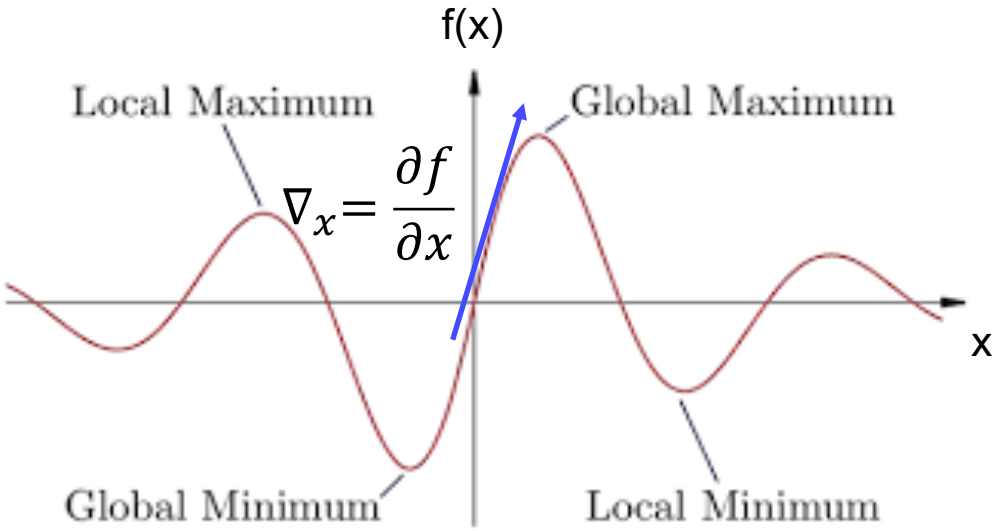
Gradient Descent Approach

- Perceptron Neural Network: A practical Case
- Example and intuitions: Batch vs on-line learning

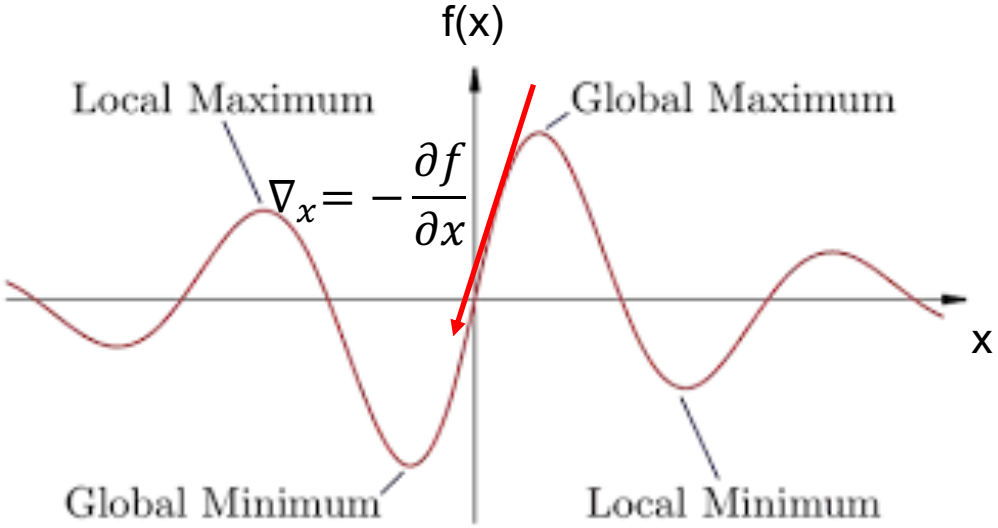
Online Learning

GRADIENT OF A FUNCTION

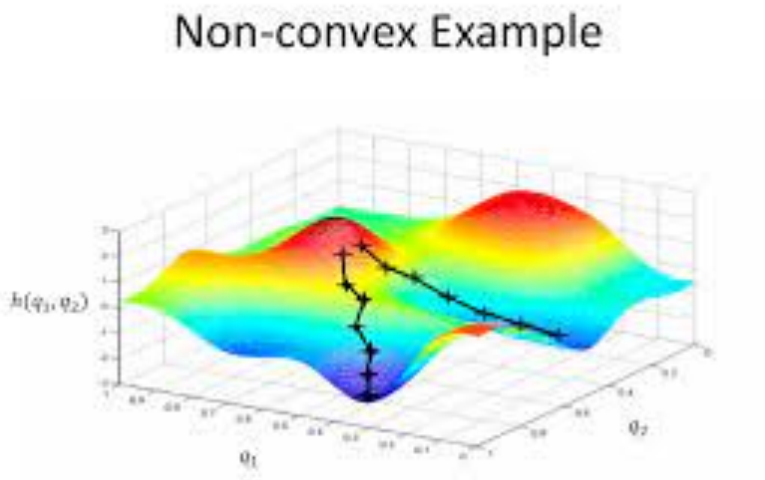


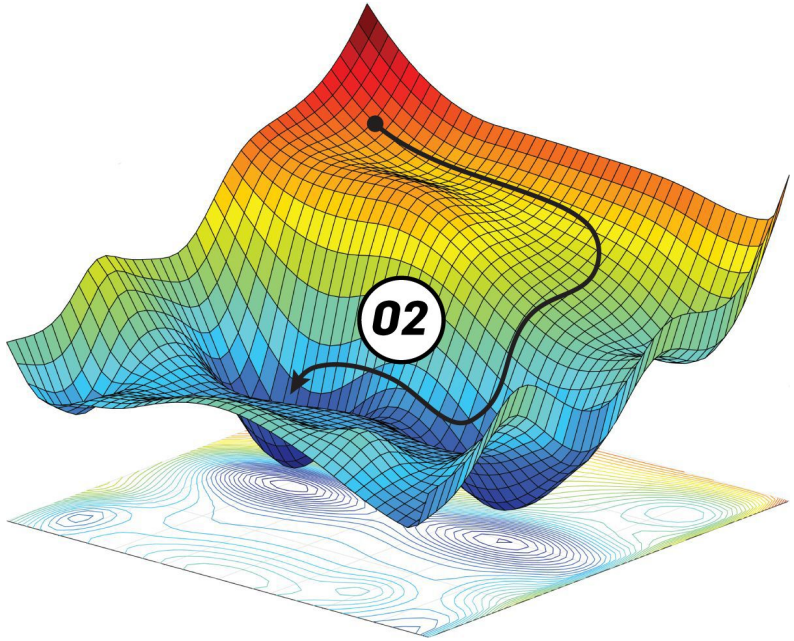


GRADIENT DESCENT APPROACH

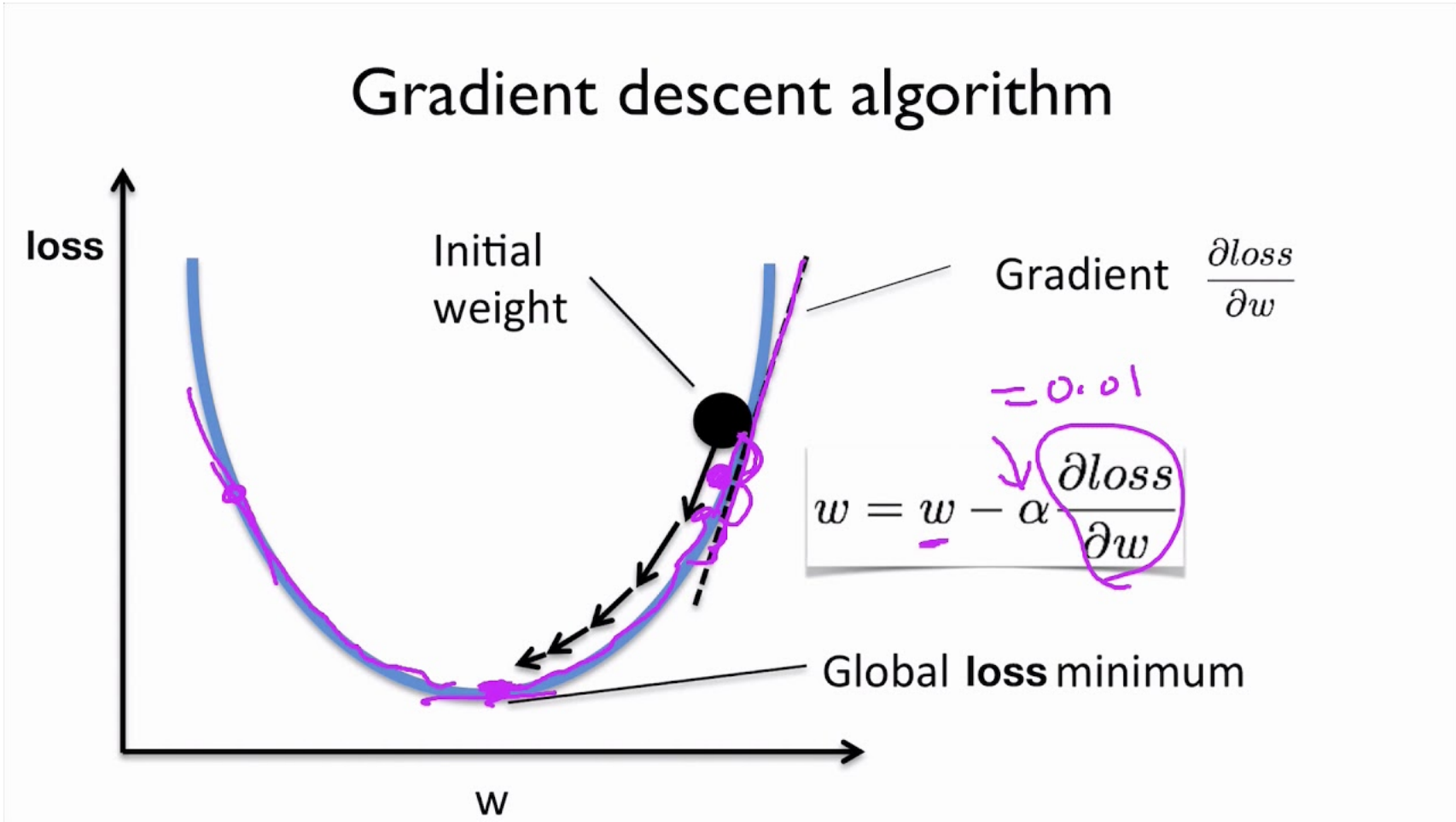


Non-Convex Function



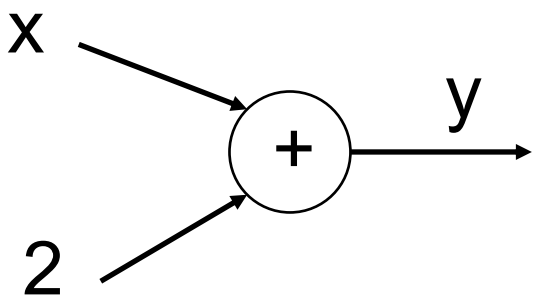


GRADIENT DESCENT APPROACH

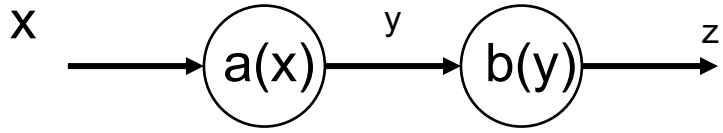


Backpropagation Algorithm

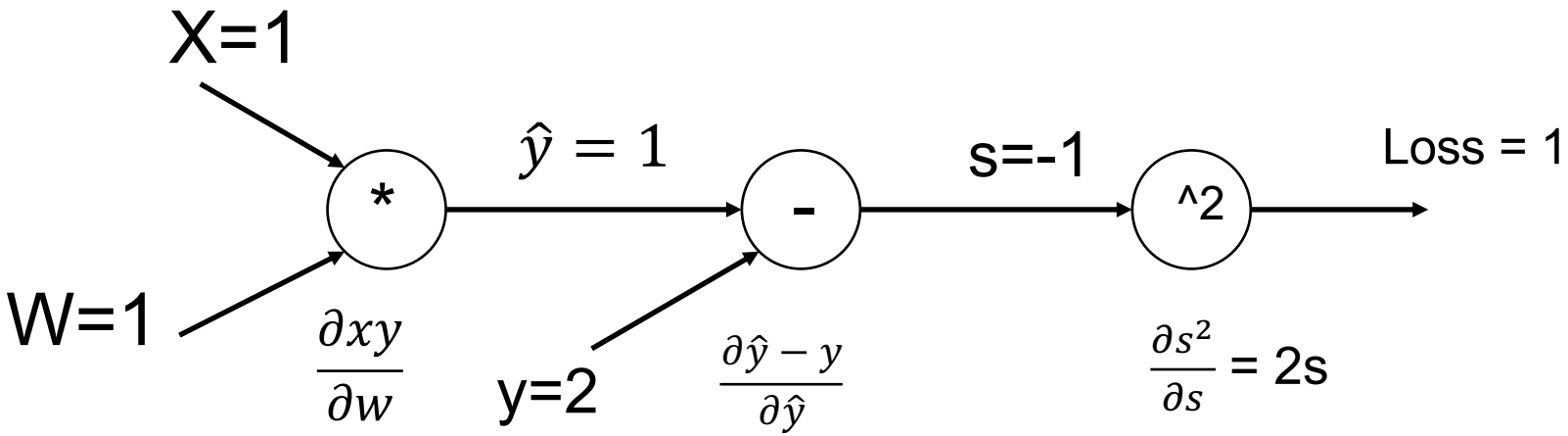
A **neural network** is a directed graph interconnected and usually called Single Layer Feedforward Network (SLFN)



Chain Rule



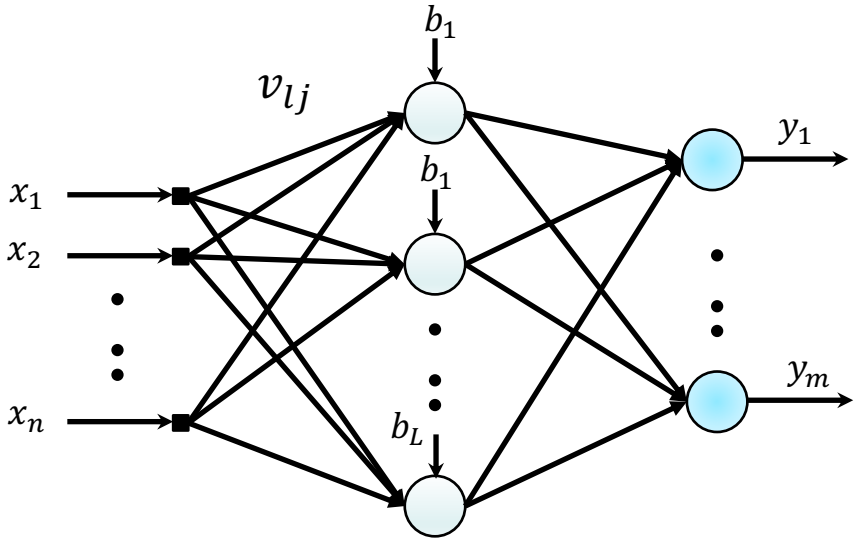
$$\frac{\partial z}{\partial x} = \frac{\partial z}{\partial y} \cdot \frac{\partial z}{\partial x}$$



$$\frac{\partial loss}{\partial w} = \frac{\partial loss}{\partial s}\frac{\partial s}{\partial \hat{y}} = -2*1 = -2$$

Backpropagation Algorithm

- **Regression:** Root-Mean-Square-Error (RMSE)



$$RMSE = \sqrt{\frac{1}{P}\sum_{p=1}^{P}\left(y_p - \hat{y}_p\right)^2}$$

The output of the two-layer NN is given by the equation:

$$y_i = \sigma\left(\sum_{l=1}^{L} w_{il}\,\sigma\left(\sum_{j=1}^{n} v_{lj}x_j + v_{l0}\right) + w_{i0}\right); i = 1,2,\ldots m \qquad (2.1)$$

Defining the hidden-layer outputs z_l allows one to write:

$$z_l = \sigma\left(\sum_{j=1}^{n} v_{lj}x_j + v_{l0}\right); l = 1,2,\ldots L \qquad (2.2)$$

Backpropagation Algorithm

The thresholds can be more easily to dealt with by defining $x_0 \equiv 1, z_0 \equiv 1$

$$y_i = \sigma\left(\sum_{\ell=0}^{L} w_{i\ell} z_\ell\right)$$
$$z_\ell = \sigma\left(\sum_{j=0}^{n} v_{\ell j} x_j\right).$$

Where the terms

$$u_i^2 = \sum_{\ell=0}^{L} w_{i\ell} z_\ell$$
$$u_\ell^1 = \sum_{j=0}^{n} v_{\ell j} x_j.$$

Thus, we have the Equations:

$$y_i = \sigma(u_i^2)$$
$$z_\ell = \sigma(u_\ell^1).$$

BACK PROPAGATION ALGORITHM:

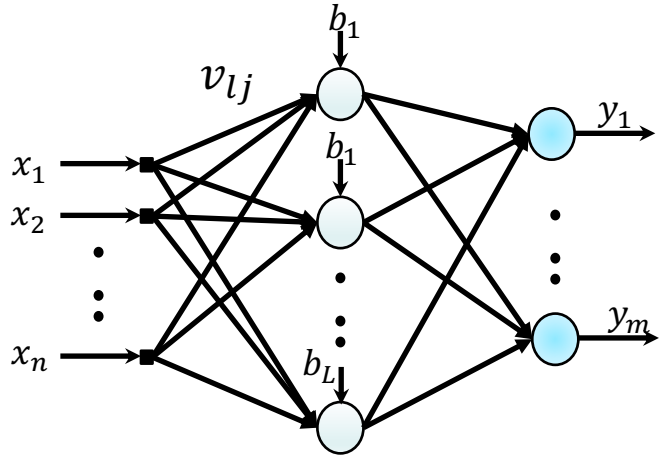
Weight tuning is a gradient descent approach, so the weights in layers two and one are updated by:

$$w_{i\ell} = w_{i\ell} - \eta \frac{\partial E}{\partial w_{i\ell}}$$
$$v_{\ell j} = v_{\ell j} - \eta \frac{\partial E}{\partial v_{\ell j}},$$

Where the **ERROR function** is defined as:

$$E = \frac{1}{2} e^T e = \frac{1}{2} \sum_{i=1}^{m} e_i^2$$
$$e_i = Y_i - y_i,$$

Where:



$$\frac{\partial y_i}{\partial w_{i\ell}} = \sigma'(u_i^2) z_\ell$$
$$\frac{\partial y_i}{\partial z_\ell} = \sigma'(u_i^2) w_{i\ell}$$
$$\frac{\partial z_\ell}{\partial v_{\ell j}} = \sigma'(u_\ell^1) x_j$$
$$\frac{\partial z_\ell}{\partial x_j} = \sigma'(u_\ell^1) v_{\ell j},$$

Backpropagation Algorithm

The required Gradients of the cost E with respect to the weights are now easily determined by:

$$\frac{\partial E}{\partial w_{i\ell}} = \frac{\partial E}{\partial u_i^2}\frac{\partial u_i^2}{\partial w_{i\ell}} = \left[\frac{\partial E}{\partial e_i}\frac{\partial e_i}{\partial y_i}\frac{\partial y_i}{\partial u_i^2}\right]\frac{\partial u_i^2}{\partial w_{i\ell}}$$

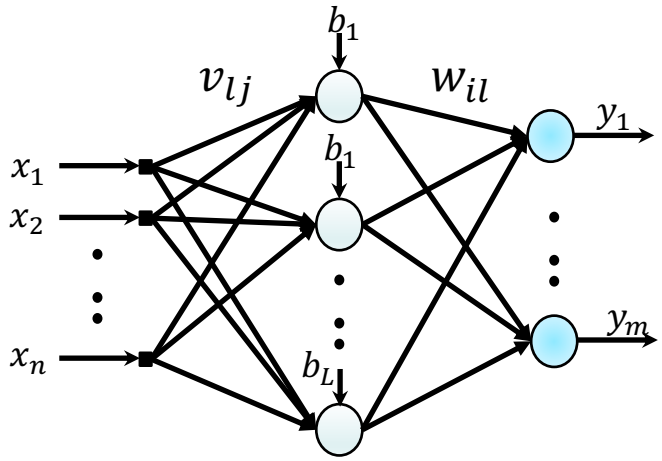
One using the above equalities:

$$\begin{aligned}\frac{\partial E}{\partial u_i^2} &= -\sigma'(u_i^2)e_i \\ \frac{\partial E}{\partial w_{i\ell}} &= -z_\ell\left[\sigma'(u_i^2)e_i\right].\end{aligned}$$

Similarly for the first layer weights:

$$\frac{\partial E}{\partial v_{\ell j}} = \frac{\partial E}{\partial u_\ell^1}\frac{\partial u_\ell^1}{\partial v_{\ell j}} = \left[\sum_{i=1}^{m}\frac{\partial E}{\partial u_i^2}\frac{\partial u_i^2}{\partial z_\ell}\frac{\partial z_\ell}{\partial u_\ell^1}\right]\frac{\partial u_\ell^1}{\partial v_{\ell j}}$$

$$\begin{aligned}\frac{\partial E}{\partial u_\ell^1} &= -\sigma'(u_\ell^1)\sum_{i=1}^{m} w_{i\ell}\left[\sigma'(u_i^2)e_i\right] \\ \frac{\partial E}{\partial v_{\ell j}} &= -X_j\left[\sigma'(u_\ell^1)\sum_{i=1}^{m} w_{i\ell}\left[\sigma'(u_i^2)e_i\right]\right].\end{aligned}$$



Using Sigmoid Activation Function

$$\sigma(s) = \frac{1}{1+e^{-s}}$$

Derivative of the Sigmoid Activation Function

$$\sigma'(s) = \sigma(s)\left(1-\sigma(s)\right)$$

Backpropagation Algorithm

These equations can be considerably simplified by introducing the notion of a back recursion through the network

$$
\begin{aligned}
w_{i\ell} &= w_{i\ell} - \eta \frac{\partial E}{\partial w_{i\ell}} \\
v_{\ell j} &= v_{\ell j} - \eta \frac{\partial E}{\partial v_{\ell j}},
\end{aligned}
$$

$$
\begin{aligned}
\delta_i^2 &\equiv -\frac{\partial E}{\partial u_i^2} = \sigma'(u_i^2) e_i \\
\delta_\ell^1 &\equiv -\frac{\partial E}{\partial u_\ell^1} = \sigma'(u_\ell^1) \sum_{i=1}^{m} w_{i\ell} \delta_i^2.
\end{aligned}
$$

Assuming the sigmoid function are used, the backpropagation errors can be computed as:

$$
\begin{aligned}
\delta_i^2 &= y_i(1 - y_i) e_i \\
\delta_\ell^1 &= z_\ell(1 - z_\ell) \sum_{i=1}^{m} w_{i\ell} \delta_i^2.
\end{aligned}
$$

Note the threshold updates are given by:

$$
\begin{aligned}
w_{i0} &= w_{i0} + \eta \delta_i^2 \\
v_{\ell 0} &= v_{\ell 0} + \eta \delta_\ell^1.
\end{aligned}
$$

Vector representation:

- **FORWARD** recursion:

$$
\begin{aligned}
z &= \sigma(V^T X) \\
y &= \sigma(W^T z),
\end{aligned}
$$

- **BACKWARD** recursion:

$$
\begin{aligned}
e &= Y - y \\
\delta^2 &= diag\{y\}\,(I - diag\{y\})\, e \\
\delta^1 &= diag\{z\}\,(I - diag\{z\})\, W\delta^2,
\end{aligned}
$$

Where, with ***y*** an m-vector, ***diag{y}*** is an ***m x m*** diagonal matrix having the entries $y_1, y_2, \ldots, y_m$. The weight and threshold updates are

$$
\begin{aligned}
W &= W + \eta z(\delta^2)^T \\
V &= V + \eta X(\delta^1)^T.
\end{aligned}
$$

Online Backpropagation Algorithm

Forward Recursion to Compute NN Output:
Present input pattern X to the NN and compute the NN output using:

$$z_\ell = \sigma\left(\sum_{j=0}^{n} v_{\ell j} X_j\right) \; ; \quad \ell = 1, 2, \ldots, L$$

$$y_i = \sigma\left(\sum_{\ell=0}^{L} w_{i\ell} z_\ell\right) \; ; \quad i = 1, 2, \ldots, m$$

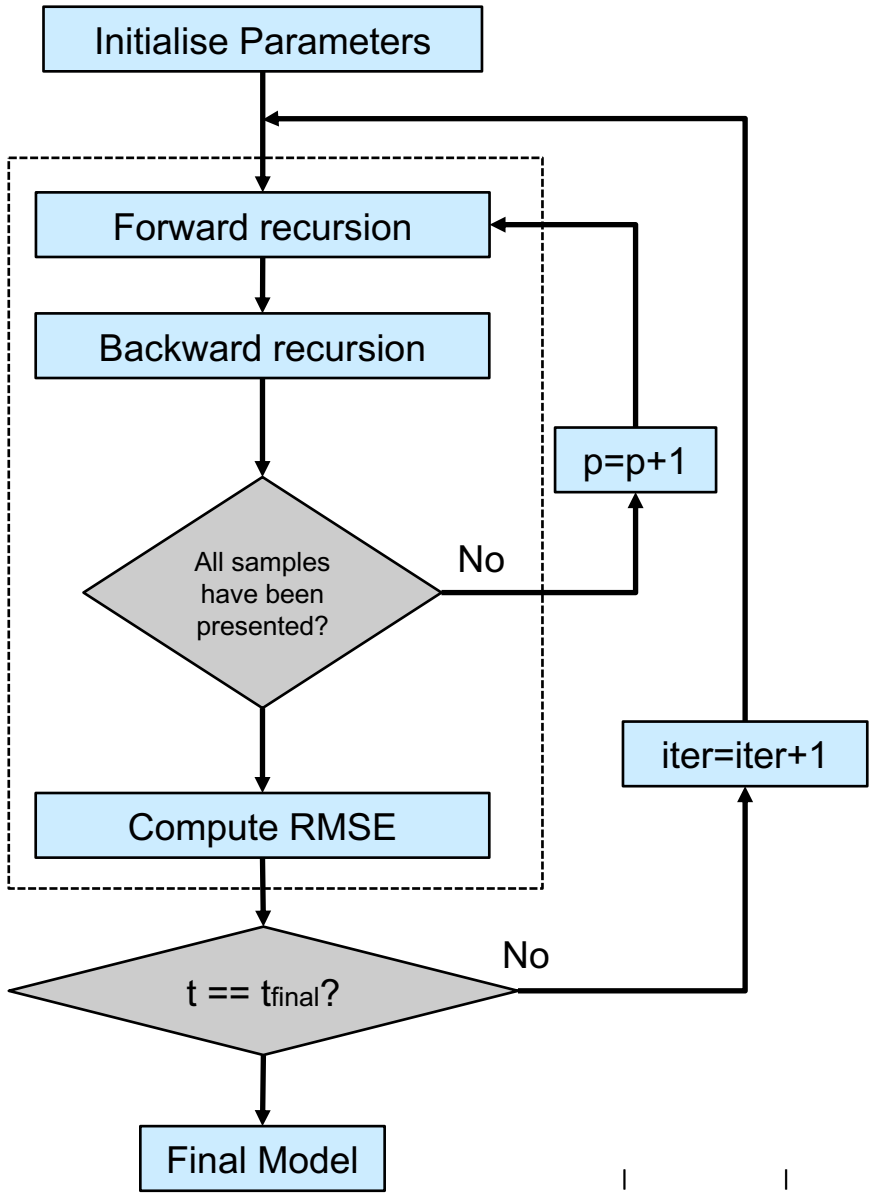
with $X_0 = 1$ and $z_0 = 1$, where Y is the desired output pattern.

Backward Recursion for Backpropagated Errors:

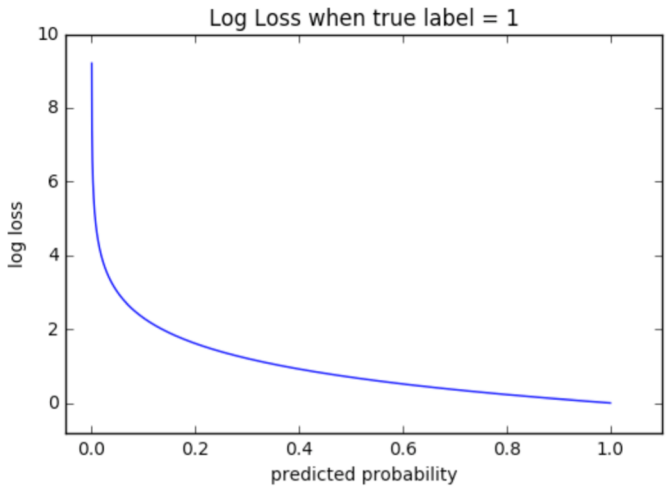
$$e_i = Y_i - y_i \; ; \quad i = 1, 2, \ldots, m$$
$$\delta_i^2 = y_i(1 - y_i)e_i \; ; \quad i = 1, 2, \ldots, m$$
$$\delta_\ell^1 = z_\ell(1 - z_\ell)\sum^{m} w_{i\ell}\delta_i^2 \; ; \quad \ell = 1, 2, \ldots, L$$

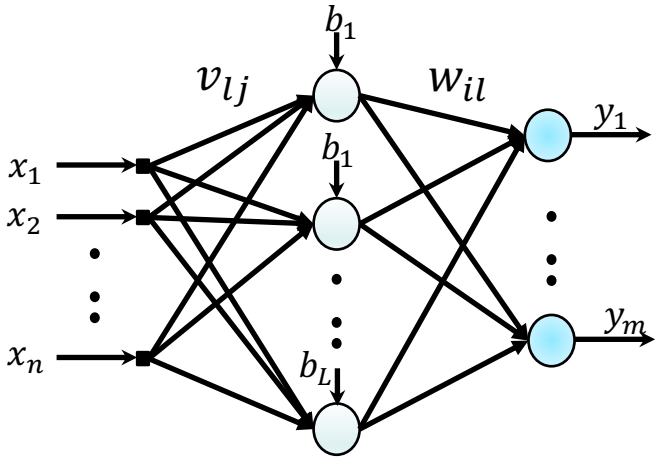
Computation of the NN Weight and Threshold Updates:

$$w_{i\ell} = w_{i\ell} + \eta z_\ell \delta_i^2 \; ; \quad i = 1, 2, \ldots, m; \; \ell = 0, 1, \ldots, L$$
$$v_{\ell j} = v_{\ell j} + \eta X_j \delta_\ell^1 \; ; \quad \ell = 1, 2, \ldots, L; \; j = 0, 1, \ldots, n$$



Cross Entropy: Binary vs Multiclass Classification





Math

In binary classification, where the number of classes M equals 2, cross-entropy can be calculated as:

$$-(y \log(p) + (1 - y) \log(1 - p))$$

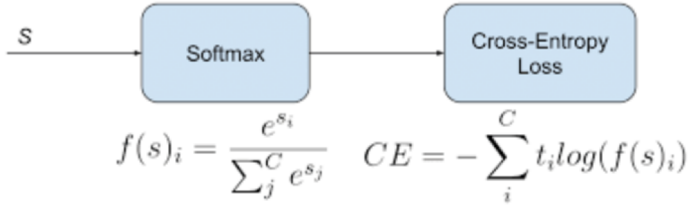
If $M > 2$ (i.e. multiclass classification), we calculate a separate loss for each class label per observation and sum the result.

Target Prediction

$$-\sum_{c=1}^{M} y_{o,c} \log(p_{o,c})$$

Note

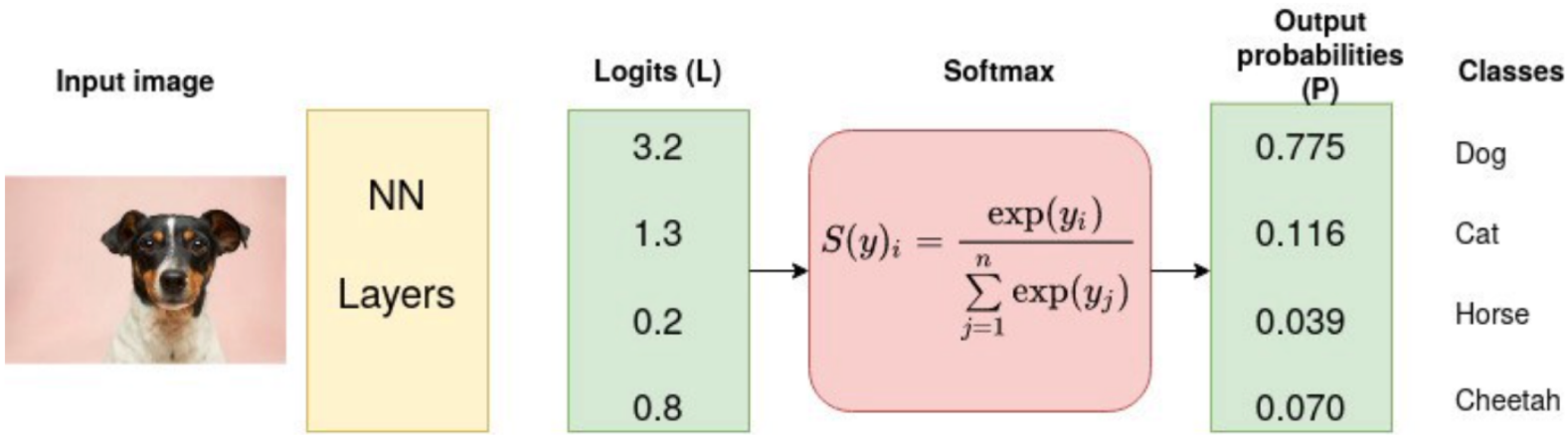
- M - number of classes (dog, cat, fish)
- log - the natural log
- y - binary indicator (0 or 1) if class label c is the correct classification for observation o
- p - predicted probability observation o is of class c



Some Loss Functions: https://ml-cheatsheet.readthedocs.io/en/latest/loss_functions.html#:~:text=Cross%2Dentropy%20loss%2C%20or%20log,diverges%20from%20the%20actual%20label.

Cross Entropy: Binary vs Multiclass Classification

What in fact Softmax and CE do is:



Softmax converts logits into probabilities. The purpose of the Cross-Entropy is to take the output probabilities (P) and measure the distance from the truth values

References:

[1] Huang G B, Zhu Q Y, Siew C K. Extreme learning machine: theory and applications[J]. Neurocomputing, 2006, 70(1-3): 489-501.

[2] Huang G B, Zhou H, Ding X, et al. Extreme learning machine for regression and multiclass classification[J]. IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), 2012, 42(2): 513-529.

[3] Deng, Wanyu, Qinghua Zheng, and Lin Chen. "Regularized extreme learning machine." *2009 IEEE symposium on computational intelligence and data mining*. IEEE, 2009.

Course Project

1. Select a data Base and apply four different techniques, i.e.
   a) Random Forest
   b) Support Vector Machine
   c) RBFNN, etc.

1. Select and model a data base using a Perceptron neural network. Please use the vector representation of the backpropagation algorithm
2. Select a data base and use Bayesian learning applied to a Neural Network


Thank you