Deep Learning for Optical Imaging

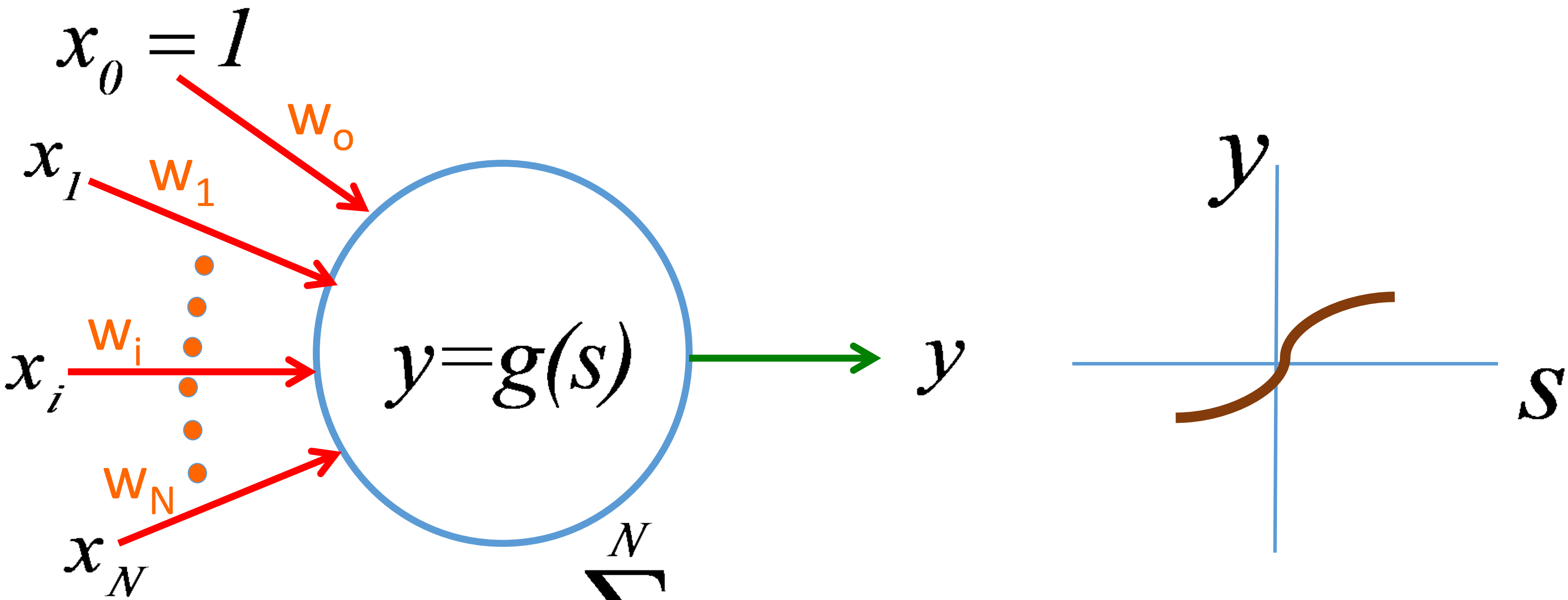
Lecture 3a

Pseudoinverse and Regression
Multliple classes

Outline

- Pseudoinverse
- Regression
- Multiple neurons
- Soft max

Single Neuron



$$s = \sum_{i=0}^{N} w_i x_i$$

Direct inversion

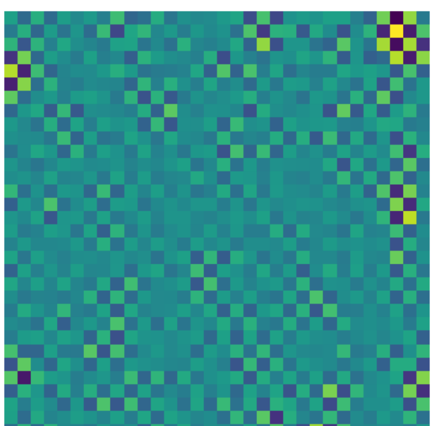
Weight calculation by matrix inversion

$$\underline{\underline{X}}\,\underline{w} = \underline{t} \Rightarrow \underline{w} = \underline{\underline{X}}^{-1}\underline{t}$$

$\underline{\underline{X}}$ is 1024 by 1024

Images used for weight calculation	100%
Test 2000 new images	52%

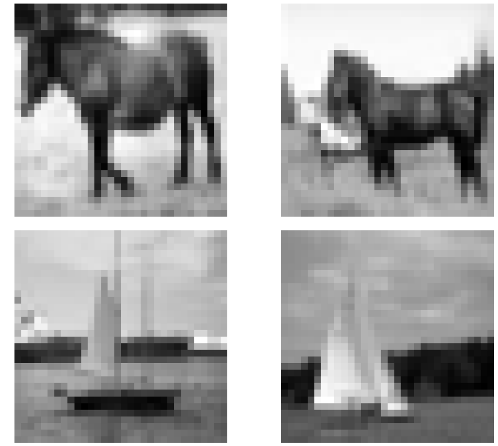
Weights ($\underline{w}$)



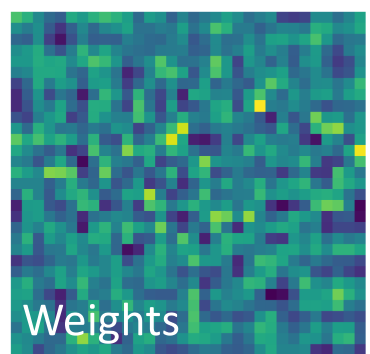
Database
1024 training images
2000 test images

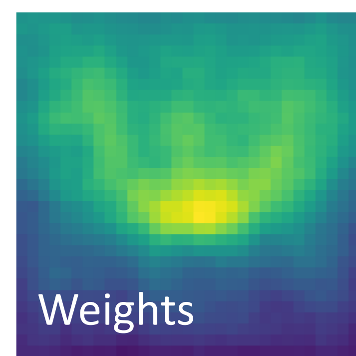
Class 1

Class 2

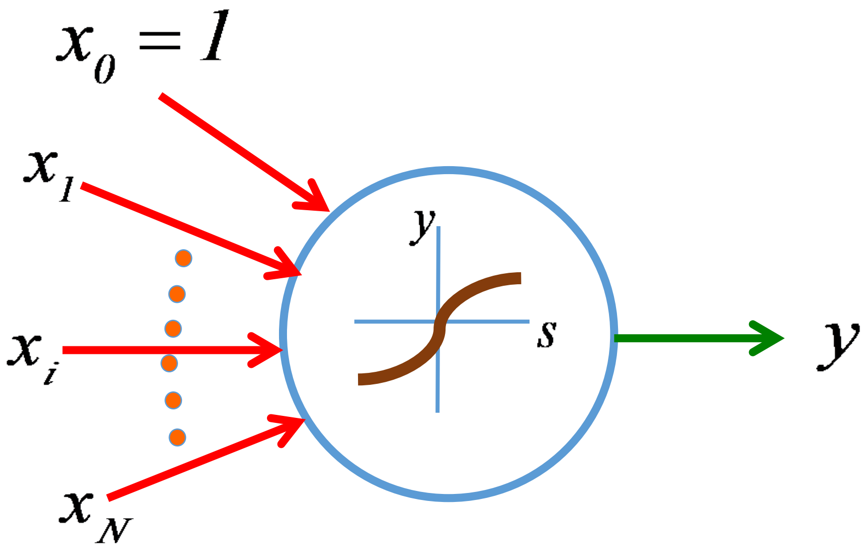


Perceptron	ADALINE
Training accuracy: 51.00%	Training accuracy: 73.44%
Test accuracy: 50.50%	Test accuracy: 71.50%
1000 iterations	Learning rate = 0.0001 2000 iterations





Adaline



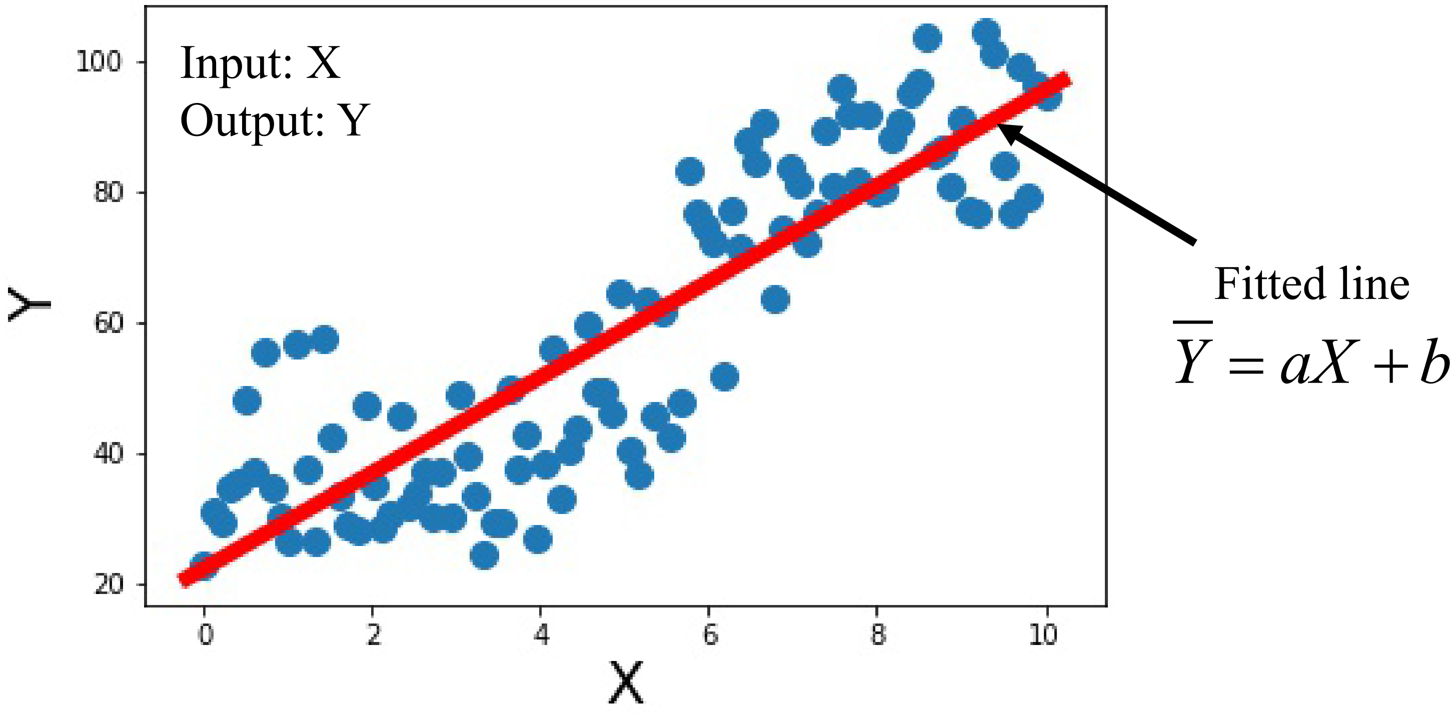
$$Training\ set: \left\{ x^m, y^m \right\} \quad m = 1, M \quad y^m = \pm 1$$

$$w_i^{t+1} = w_i^t + \Delta w_i^t$$

$$\Delta w_i^t = \alpha \frac{dg}{ds} \frac{ds}{dw_i} (y^m - y \ ) = \alpha \frac{dg}{ds} (y^m - y \ ) x_i^m$$

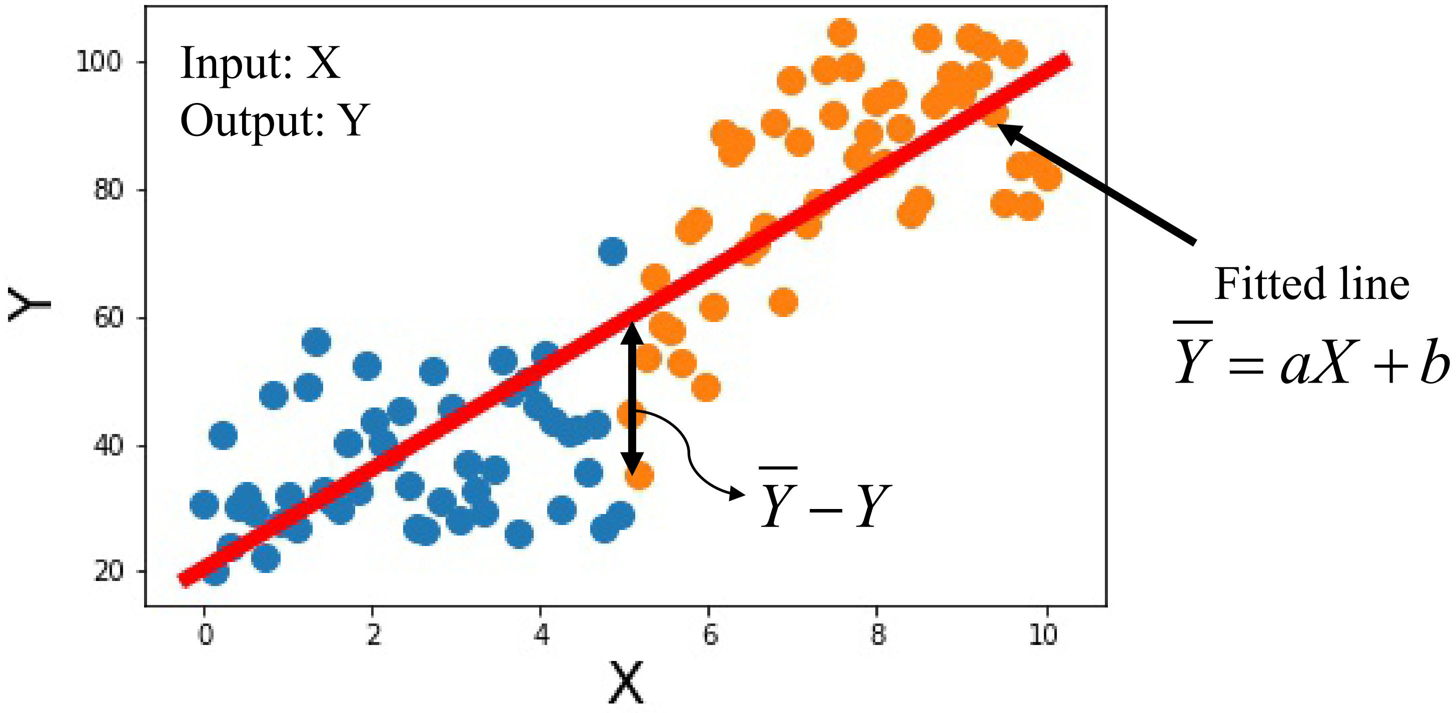
$$\alpha > 0$$

Regression



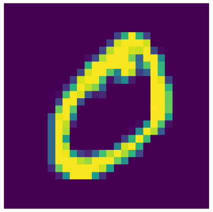
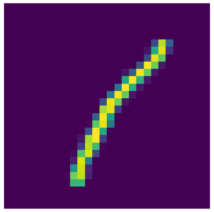
What is the difference between linear regression and ADALINE?

Regression

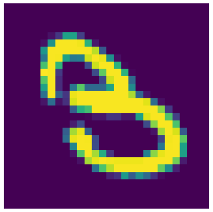
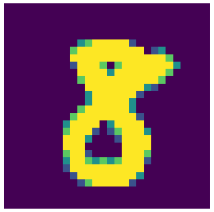


Input: X
Output: Y
Fitted line
$\overline{Y} = aX + b$
$\overline{Y} - Y$
Y
X
100
80
60
40
20
0
2
4
6
8
10

Regression vs. ADALINE: digit classification

ADALINE - Test	99.91%
Regression - Test	99.29%

ADALINE - Test	96.57%
Regression - Test	96.02%

ADALINE:

- Sigmoid activation function
- Sigmoid slope = 0.0001
- Learning rate = 0.0001
- Epoch = 200

A threshold function is used after the regression in order to classify the outputs.

Regression

Linear regression finds the best linear fit relationship between the input variables (x) and the single output (y).

$$y^{(m)} = \sum_{i}^{N} \beta_i x_i^{(m)} = \vec{\beta}.\overrightarrow{x^{(m)}}$$

The model parameters (β) can be calculated using least-squares estimation:

$$\vec{\beta} = \min\left(\sum_{m}^{M} (\vec{\beta}.\overrightarrow{x^{(m)}} - y_i)^2 \right)$$

We can put input and out variables in matrices X and Y.

$$\vec{\beta} = \min\left(\left(X\vec{\beta} - Y \right)^2 \right)$$

The optimum model parameter (β) lies at gradient zero:

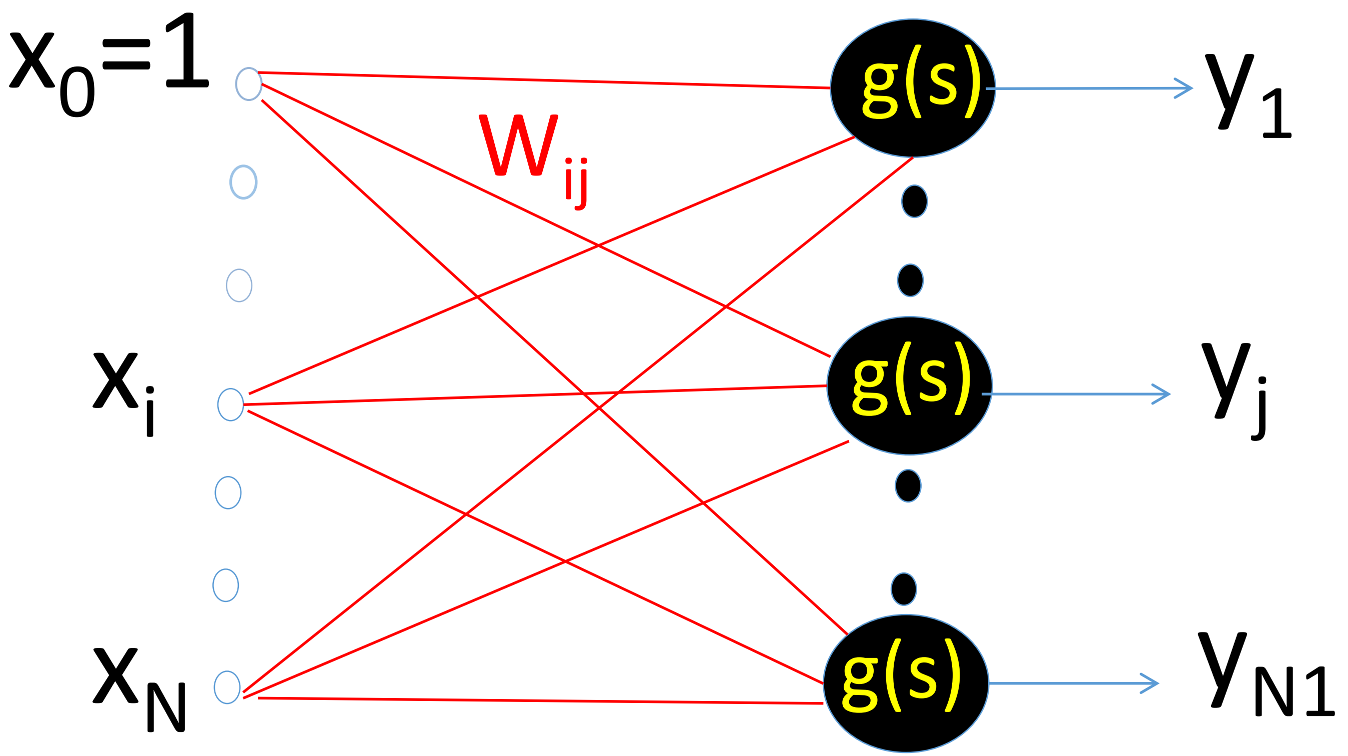
$$\frac{\partial\left[\left(X\vec{\beta} - Y \right)^2 \right]}{\partial \vec{\beta}} = 0 \rightarrow \vec{\beta} = (X^T X)^{-1} X^T Y$$

The index m is used for the samples. M is the total number of the samples. N is the dimension of the input.

Outline

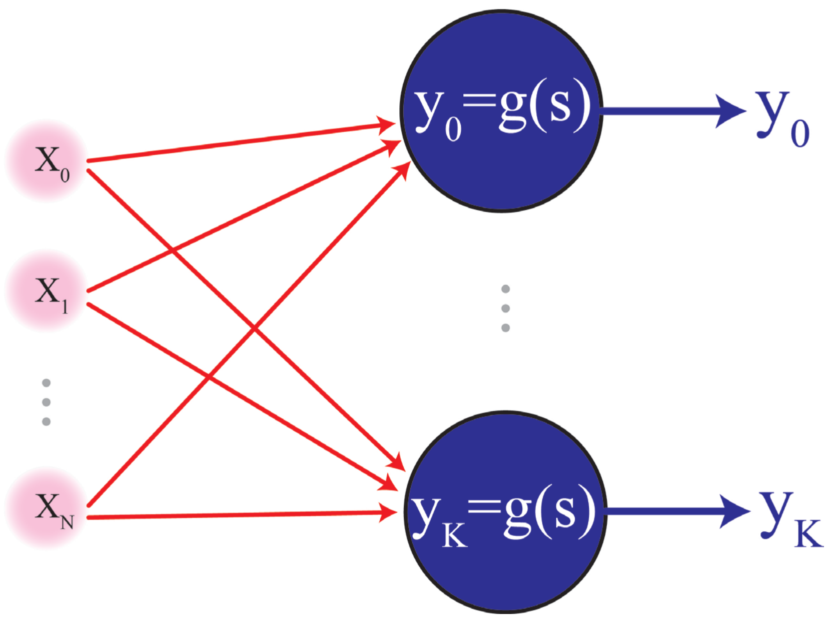
- Pseudoinverse
- Regression
- Multiple neurons
- Soft max

Multiple neurons



$$Y_j=g(\Sigma w_{ij}x_i+w_{0j})$$

Multi neurons



$$s_k = \sum_{i}^{N} x_i w_{ik}$$

$$y_k = g(\sum_{i}^{N} x_i w_{ik})$$

$$Labels : y_k^{(m)}$$

$$g(s_k) = \frac{1}{1+e^{-s_k}}$$

The index *i* determines the input.
The index *k* determines the output.
N is the total number of the inputs.
K is the total number of the outputs.
The index *m* determines the sample number

Multi neurons: Mean squared cost function

The mean squared cost function is: $$C = \sum_{m}^{M} \sum_{k}^{K} (y_k^{(m)} - y_k)^2$$

The index m is used for the training samples. M is the total number of the samples. The index k is used for the outputs. K is the total number of the outputs. $y^{(m)}_k$ denotes the label for the k^{th} neuron and y_k denotes the predicted output by the k^{th} neuron. Activation function is sigmoid.
$\frac{\partial C}{\partial w_i}$ calculation is as follows:

$$\frac{\partial C}{\partial w_{ik}} = \frac{\partial C}{\partial y_k} \frac{\partial y_k}{\partial s_k} \frac{\partial s_k}{\partial w_{ik}}$$

$$C = \sum_{m}^{M} \sum_{k}^{K} (y_k^{(m)} - y_k)^2 \rightarrow \frac{\partial C}{\partial y_k} = -\sum_{m}^{M} 2(y_k^{(m)} - y_k)$$

$$\Rightarrow \frac{\partial C}{\partial w_{ik}} = -\sum_{m}^{M} 2(y_k^{(m)} - y_k) y_k (1 - y_k) x_i^{(m)}$$

Multi neurons: Cross entropy cost function

The cross entropy cost function is: $C = -\sum_{m}^{M} \sum_{k}^{K} y_k^{(m)} \ln(y_k)$

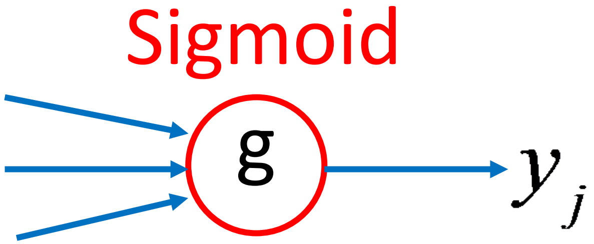
i is the index of the inputs
m is the index of the training samples
k is the index of the outputs
$y^{(m)}_k$ denotes the label for the k^th neuron and y_k denotes the predicted output by the k^th neuron. Activation function is sigmoid.
$\frac{\partial C}{\partial w_i}$ calculation is as follows:

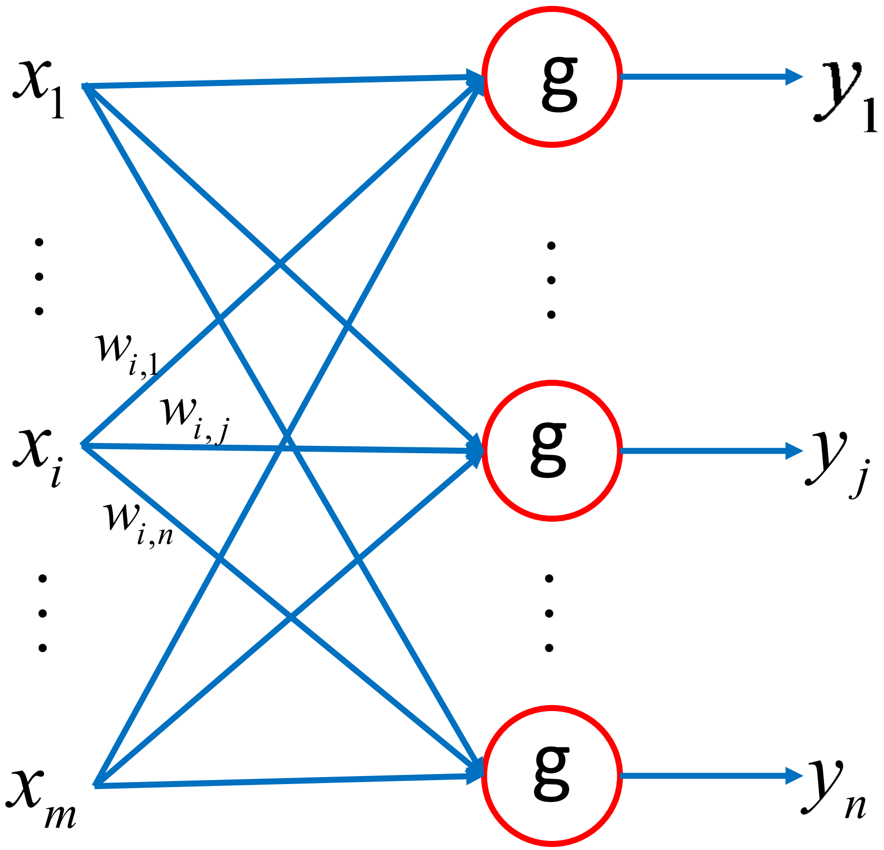
$$\frac{\partial C}{\partial w_{ik}} = \frac{\partial C}{\partial y_k} \frac{\partial y_k}{\partial s_k} \frac{\partial s_k}{\partial w_{ik}}$$

$$C = -\sum_{m}^{M} \sum_{k}^{K} y_k^{(m)} \ln(y_k) \rightarrow \frac{\partial C}{\partial y_k} = -\sum_{m}^{M} \frac{y_k^{(m)}}{y_k}$$

$$\Rightarrow \frac{\partial C}{\partial w_{ik}} = -\sum_{m}^{M} y_k^{(m)} (1 - y_k) x_i^{(m)}$$

Multiple neurons





Desired output

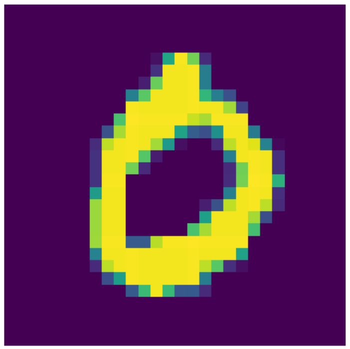
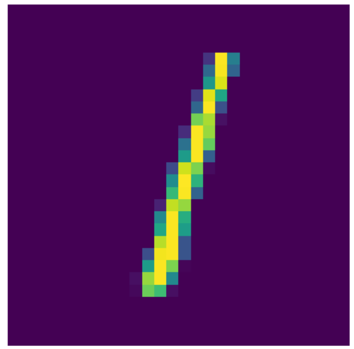
$$\text{class1}:\left[1,0,0,\ldots,0\right]$$

$$\text{class2}:\left[0,1,0,\ldots,0\right]$$

$$\text{class3}:\left[0,0,1,\ldots,0\right]$$

$$\vdots$$

Multiple neurons :
multiple digits classification (0,1, and 2)

Three neurons classification

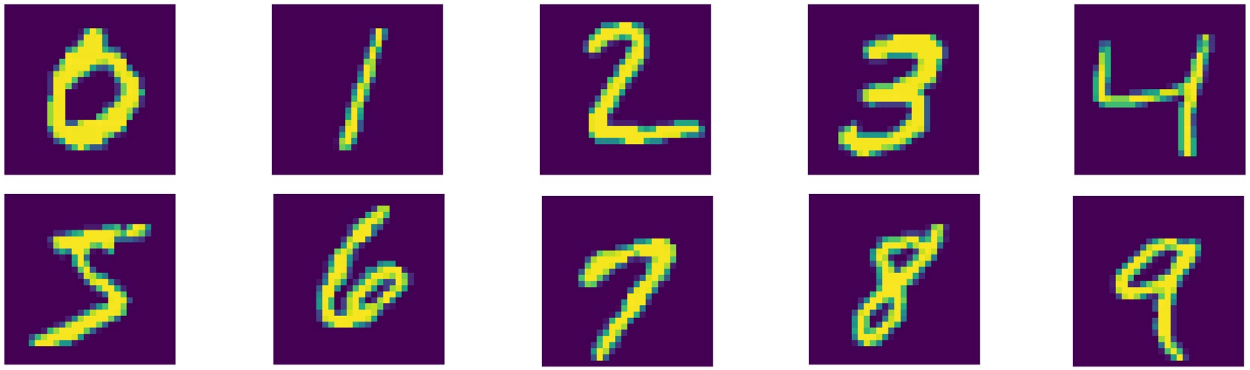
ADALINE – Training accuracy	96.71%
ADALINE – Test accuracy	97.01%

Digit 0 : Test accuracy	99.39%
Digit 1 : Test accuracy	98.50%
Digit 2 : Test accuracy	93.12%

18623 training samples
3147 test samples

Sigmoid activation function

Multiple neurons : all digits classification (0 to 9)



Accuracy of classification for different digits

0	91.60%
1	93.61%
2	80.42%
3	77.03%
4	82.45%
5	67.90%
6	89.64%
7	82.80%
8	59.75%
9	78.32%

Ten neurons classification

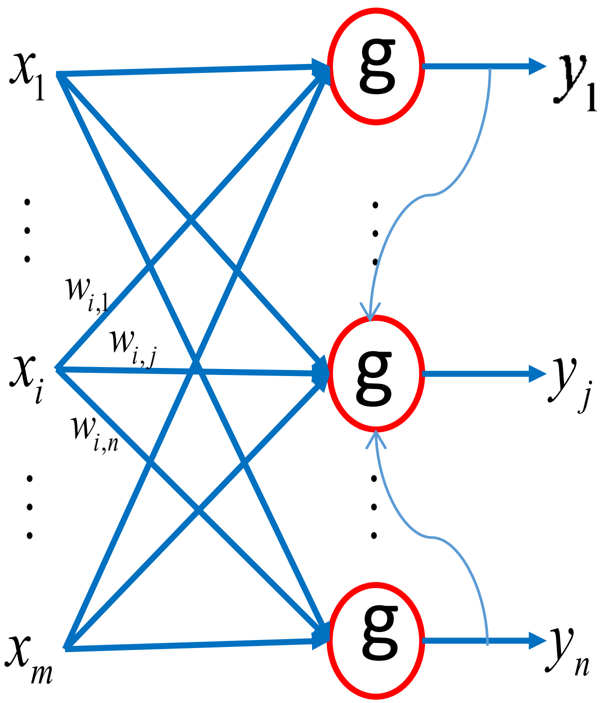
ADALINE – Training accuracy	85.00%
ADALINE – Test accuracy	80.83%

Higher accuracy can be reached with more iterations. However, the training is very slow that needs a lot of iterations.

10000 training samples
10000 test samples

Sigmoid activation function

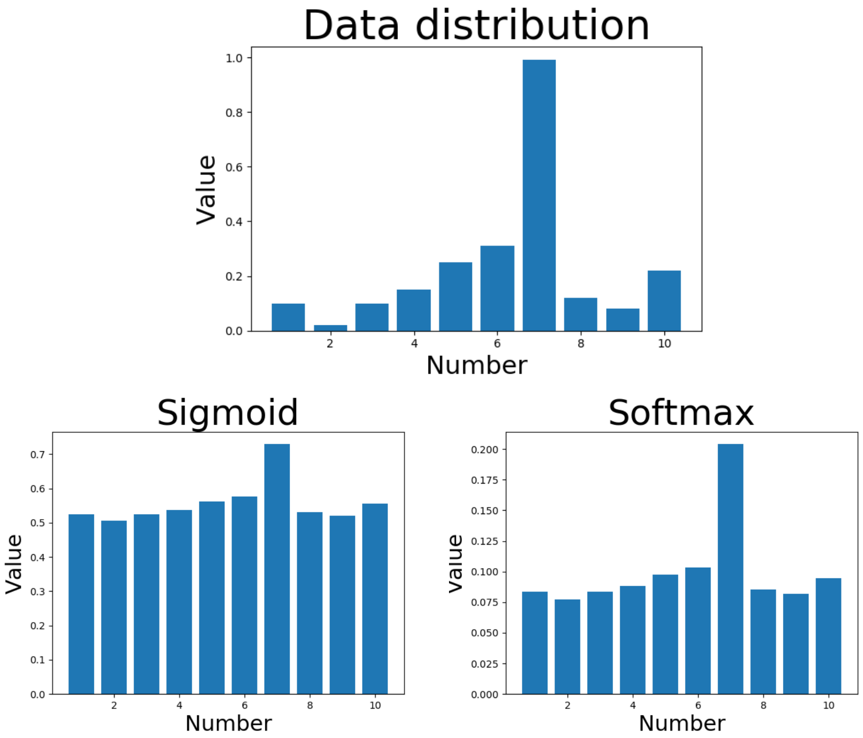
Soft Max

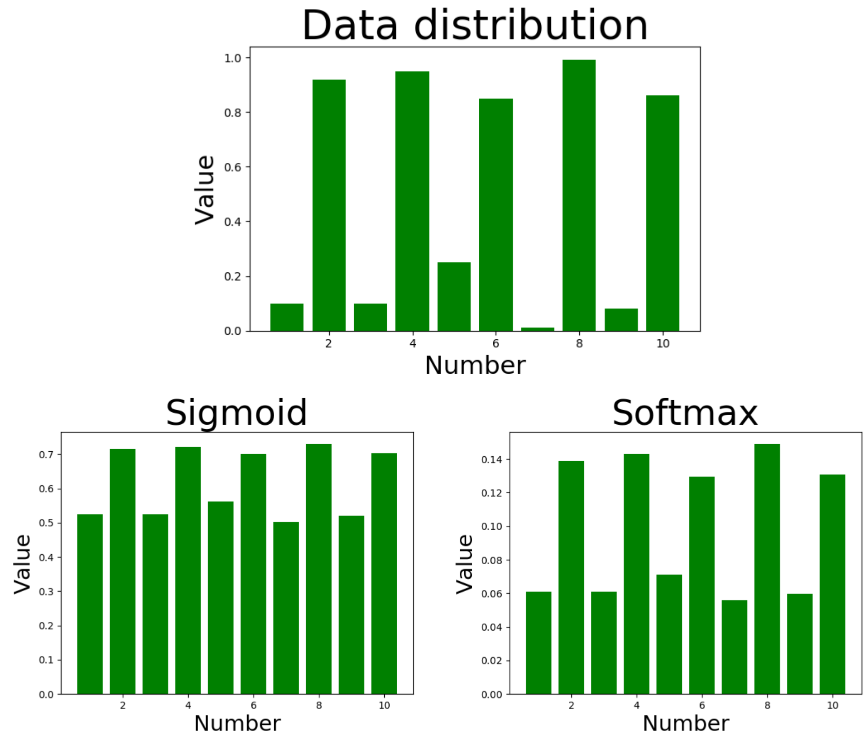


$$y_j = g(\Sigma w_{ij} x_i) = \frac{e^{\Sigma w_{ij} x_i}}{\sum_k e^{\Sigma w_{ik} x_i}}$$

$$\frac{dg}{dw_m} = \frac{e^{\Sigma w_{ij} x_i}}{\sum_k e^{\Sigma w_{ik} x_i}} \left(\delta_{mj} - \frac{e^{\Sigma w_{im} x_i}}{\sum_k e^{\Sigma w_{ik} x_i}}\right) x_i$$

Softmax and Sigmoid





Softmax vs Sigmoid

All digits classification (0 to 9)

Cross entropy error loss function
400 iterations

Sigmoid test accuracy: 89.99
Softmax test accuracy: 86.16