Deep Learning for Optical Imaging

Lecture 2b

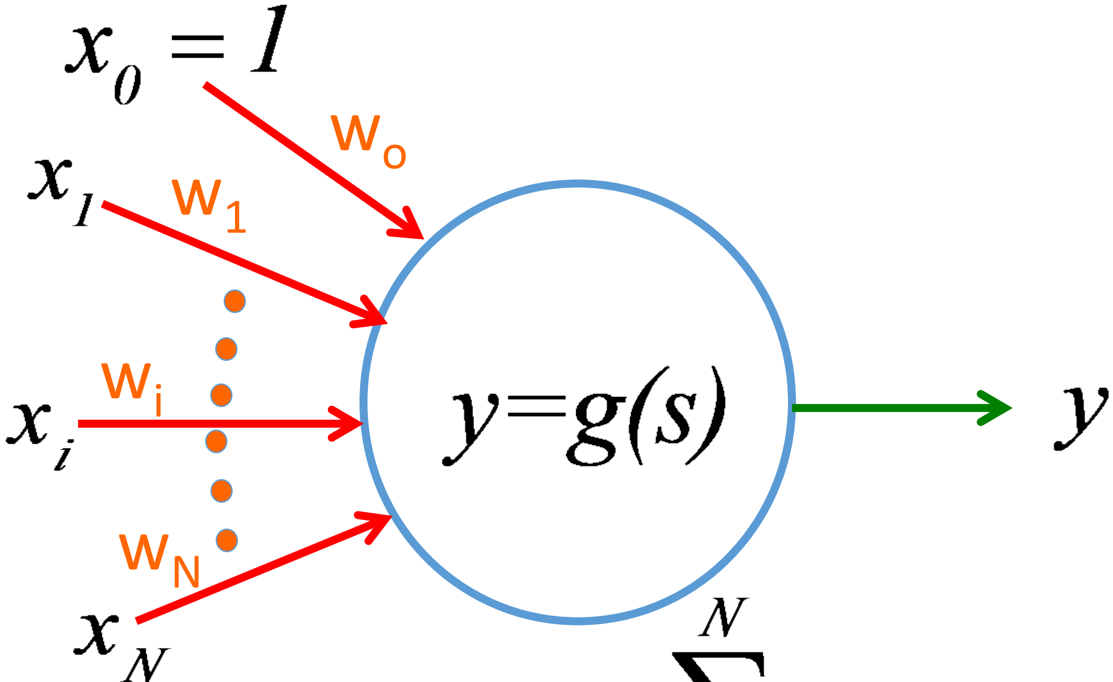
Single neuron

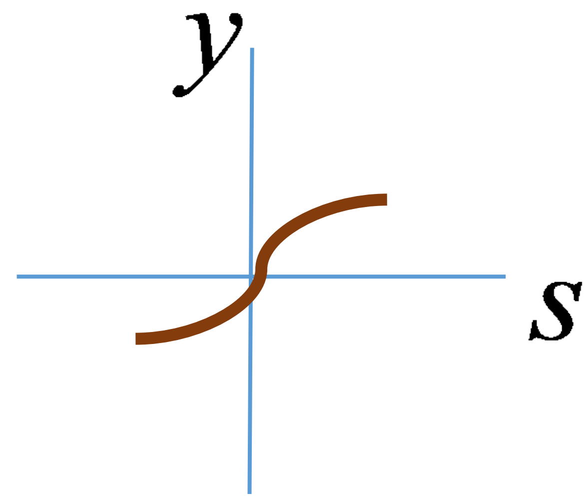
(continued)

Outline

1. Adaline continued
2. Regression
3. Cross-entropy

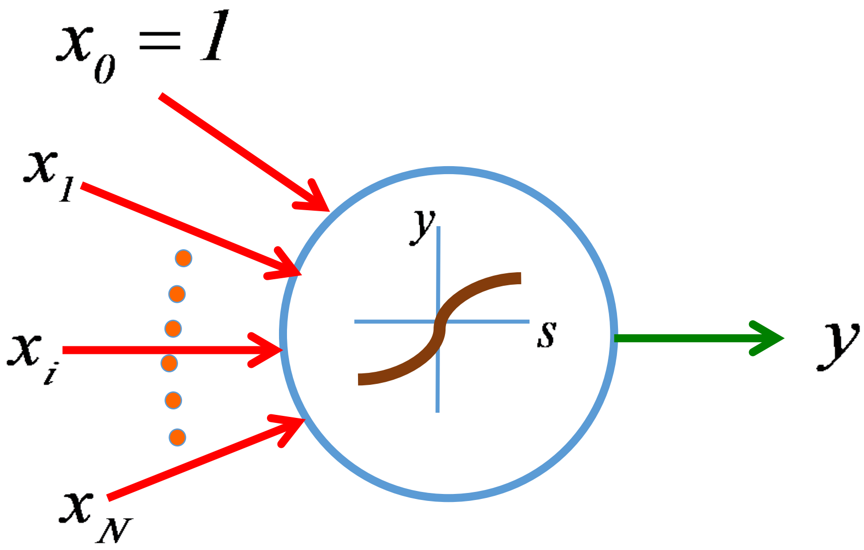
Single Neuron





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

Adaline



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

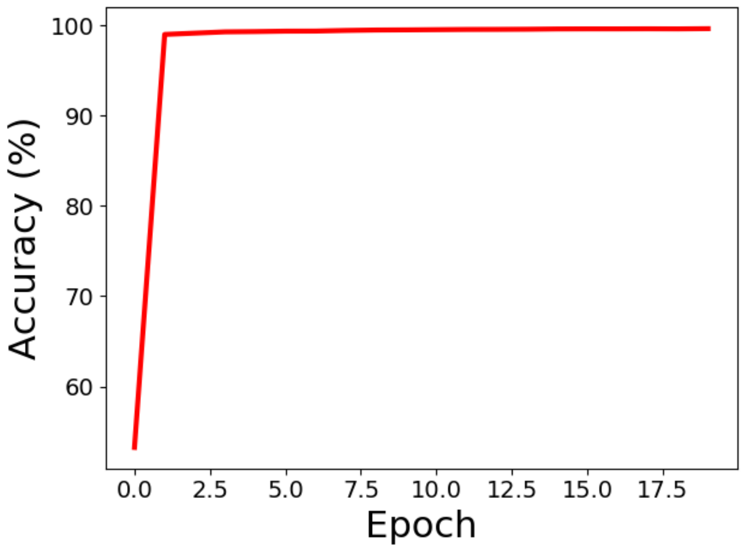
$$w_i^{t+1} = w_i^t + \Delta w_i^t$$

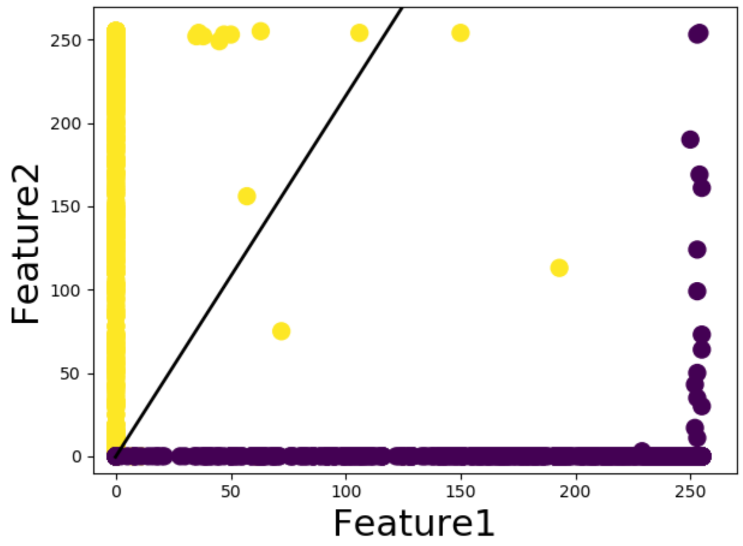
$$\Delta w_i^t = \alpha \frac{dg}{ds}\frac{ds}{dw_i}(y^m - y) = \alpha \frac{dg}{ds}(y^m - y)x_i^m$$

$$\alpha > 0$$

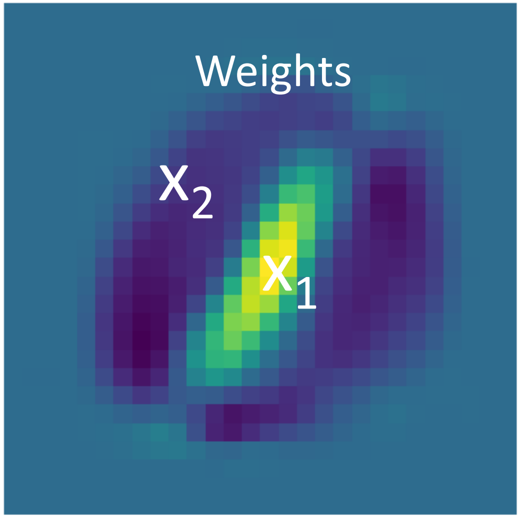
Binary classification of digits-ADALINE

Accuracy of classification





Training accuracy	99.66%
Test accuracy	99.95%

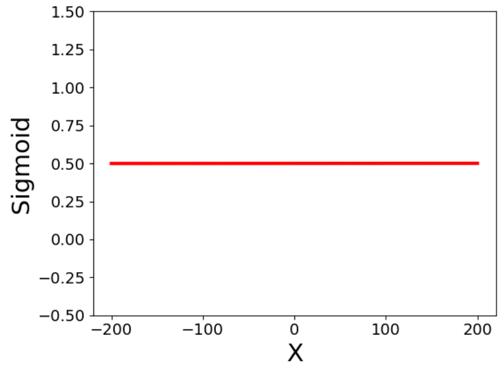


12665 Training samples
2115 Test samples
Sigmoid activation function

Face classification accuracy on the training set (Adaline)

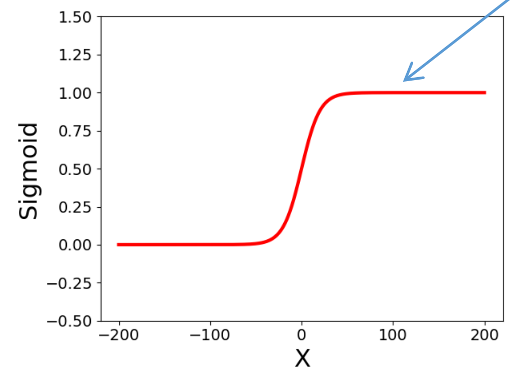
Sigmoid Slope = 0.00001

Accuracy = 100%



Sigmoid Slope = 0.1

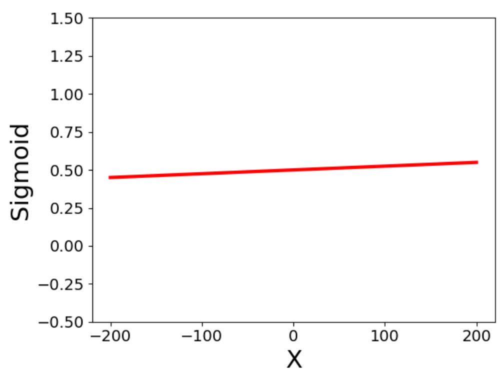
Accuracy = 50%



$$\frac{dg}{ds} \approx 0$$

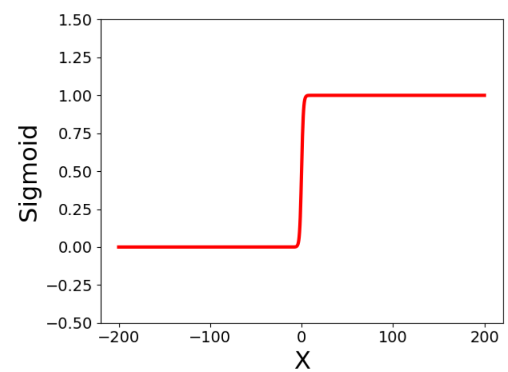
Sigmoid Slope = 0.001

Accuracy = 100%



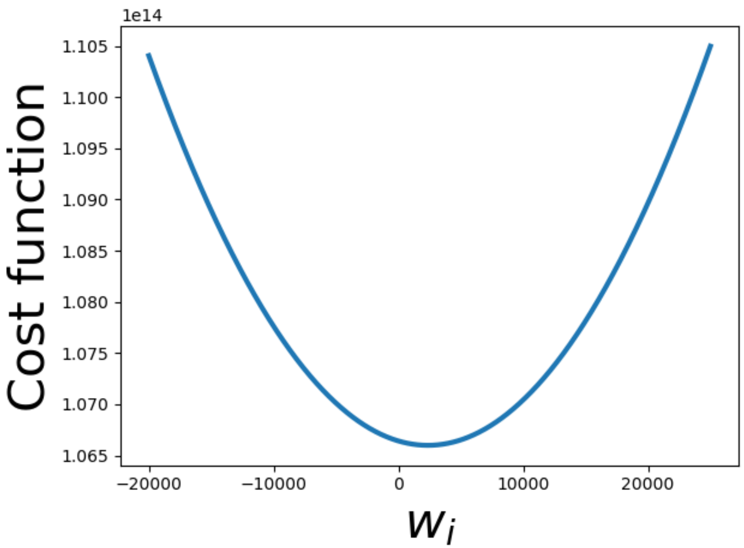
Sigmoid Slope = 1

Accuracy = 50%

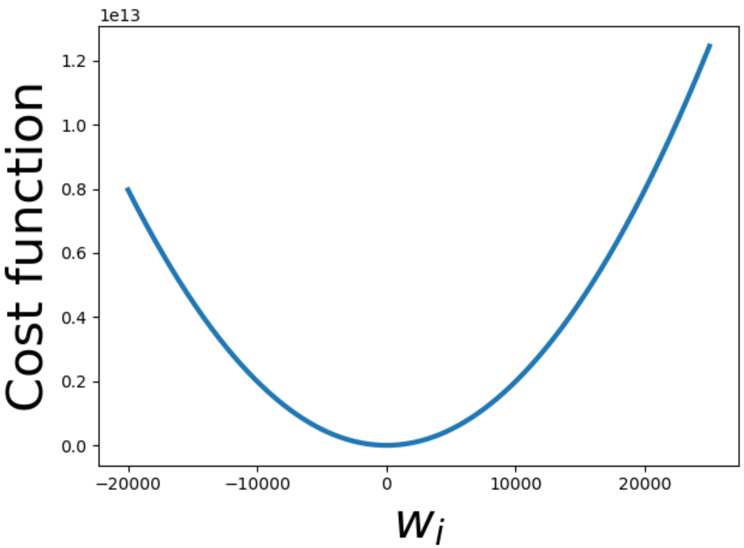


Cost function vs. w_i (Linear activation)

$$C(w) = \sum_{m}^{M} \left(y^{(m)} - y\right)^2 \qquad g(s) = s$$



Noisy weights

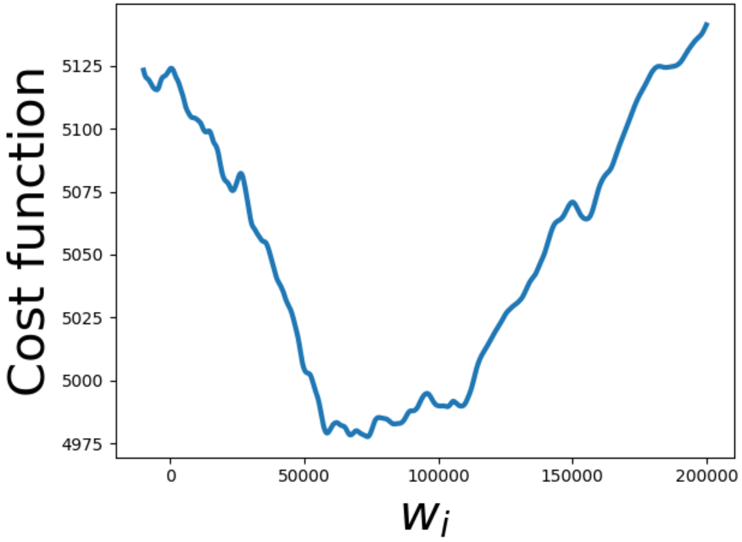


Trained weights

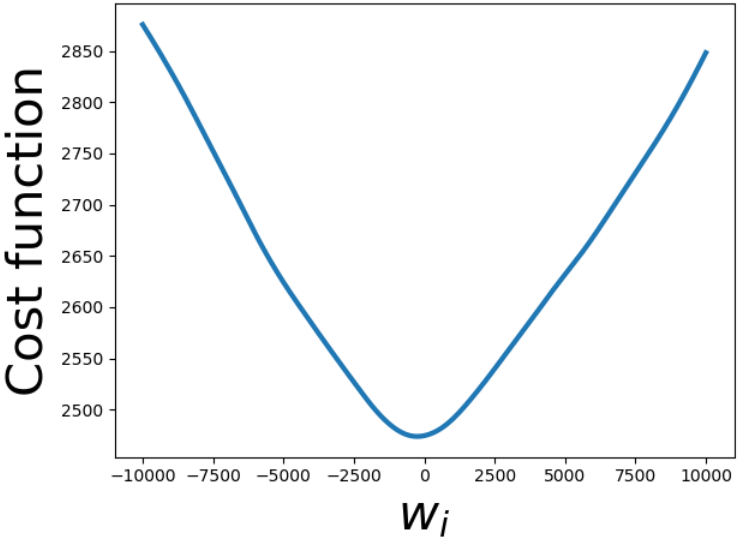
CIFAR10 database- Planes versus cars-32x32 pixels

Cost function vs. w_i (Sigmoid activation)

$$C(w) = \sum_{m}^{M} \left(y^{(m)} - y\right)^2 \qquad g(s) = \frac{1}{1+e^{-s}}$$



Noisy weights



Trained weights

CIFAR10 database- Planes versus cars-32x32 pixels

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.

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

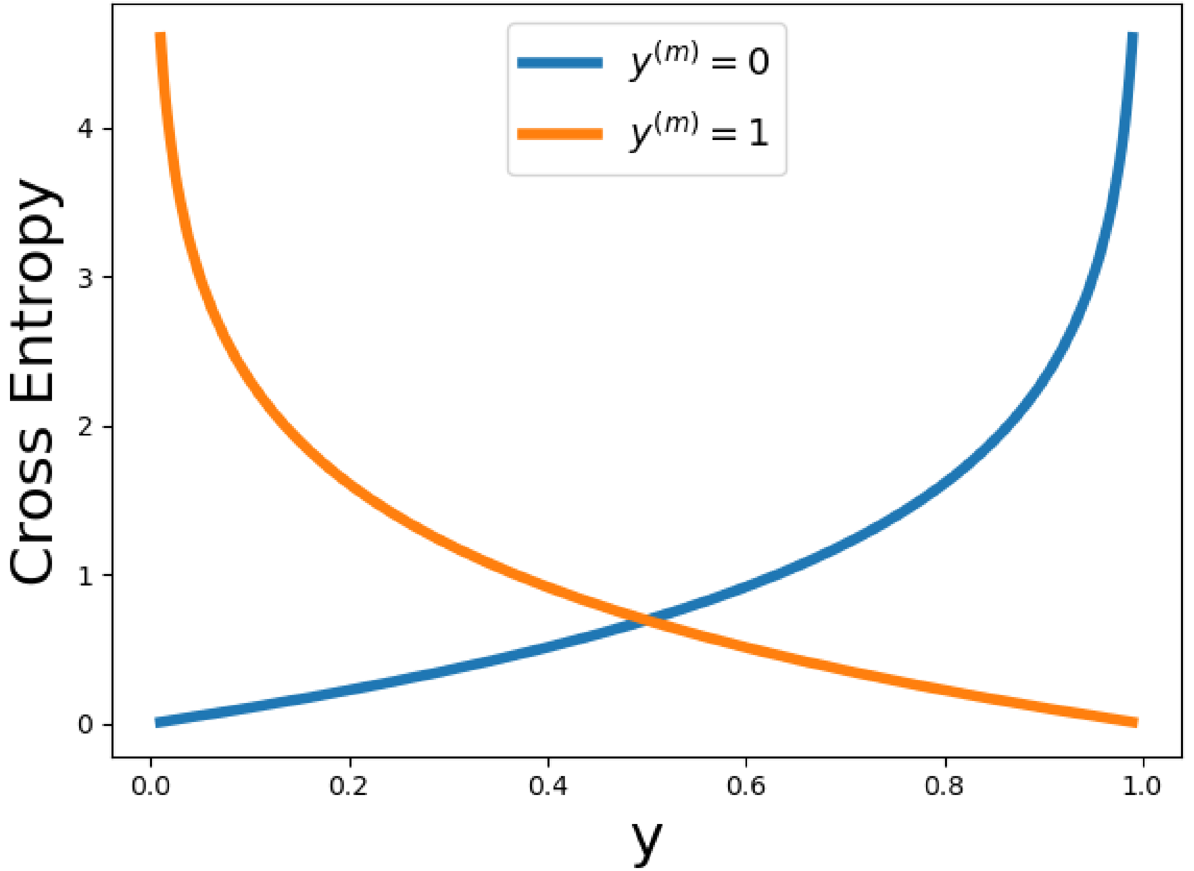
The optimum model parameter (β) lies at gradient zero:

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

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

One neuron: Cross entropy cost function

$$C = -\sum_m \left[y^{(m)} \ln(y) + (1 - y^{(m)}) \ln(1 - y) \right]$$



Question

One neuron: Cross entropy cost function

The cross entropy cost function for one neuron is:

$$C = -\sum_{m}^{M}\left[y^{(m)}\ln(y) + (1-y^{(m)})\ln(1-y)\right]$$

i is the index of the weights and m is the index of the training samples.
$y^{(m)}$ denotes the label and y denotes the predicted output by the neuron.

The weight w_i is updated using the following equation:

$$w_i^{new} = w_i^{old} - \alpha\frac{\partial C}{\partial w_i}$$

α is the learning rate

$\frac{\partial C}{\partial w_i}$ calculation is as follows:

$$\frac{\partial C}{\partial w_i} = \frac{\partial C}{\partial y}\frac{\partial y}{\partial s}\frac{\partial s}{\partial w_i}$$

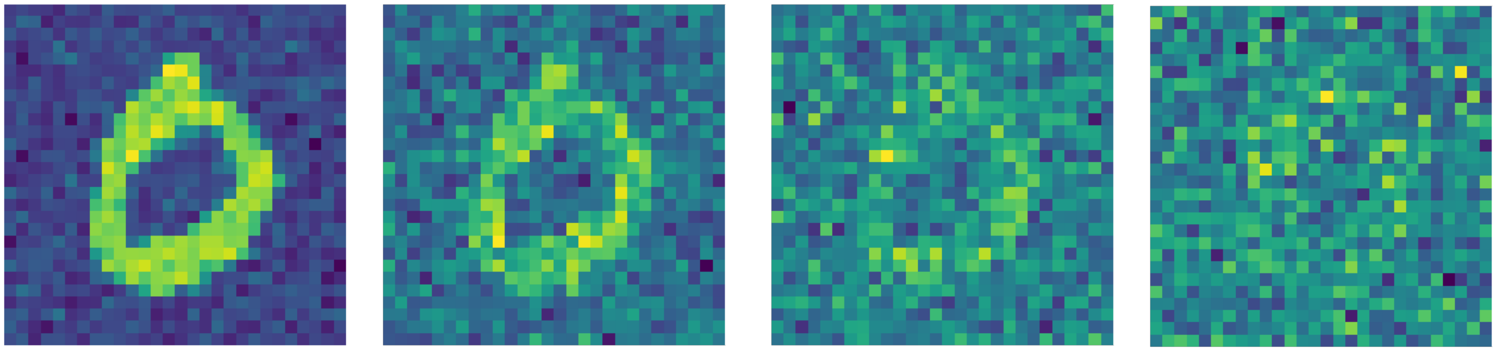
$$s = \sum_{i}^{N} w_i x_i \rightarrow \frac{\partial s}{\partial w_i} = x_i$$

$$y = g(s) = \frac{1}{1+e^{-s}} \rightarrow \frac{\partial y}{\partial s} = \frac{e^{-s}}{1+e^{-s}} = y(1-y)$$

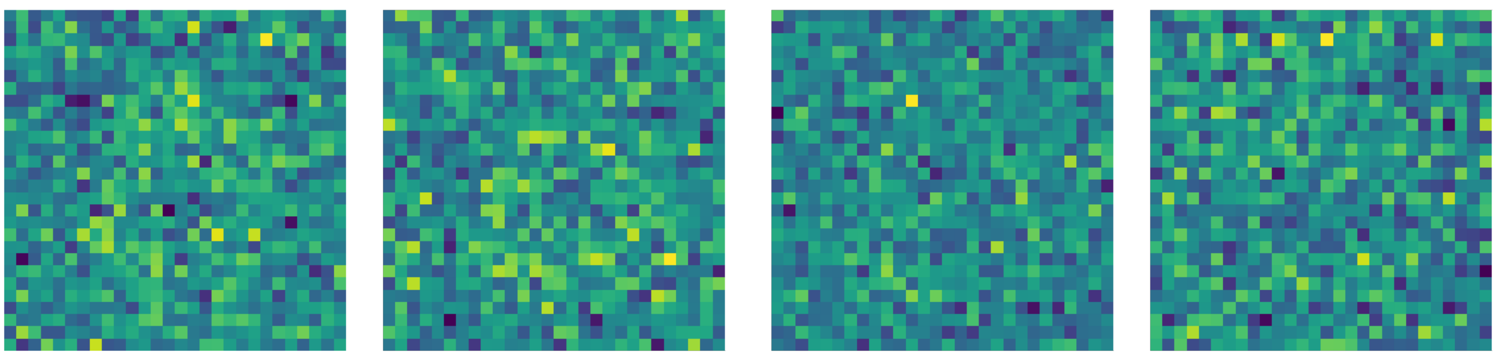
$$C = -\sum_{m}^{M}\left[y^{(m)}\ln(y) + (1-y^{(m)})\ln(1-y)\right] \rightarrow \frac{\partial C}{\partial y} = -\sum_{m}^{M}\left[\frac{y^{(m)}}{y} - \frac{(1-y^{(m)})}{1-y}\right] = -\sum_{m}^{M}\left[\frac{y^{(m)}-y}{y(1-y)}\right]$$

$$\Rightarrow \frac{\partial C}{\partial w_i} = -\sum_{m}^{M}(y^{(m)}-y)x_i$$

Digits: Gaussian noise



Perceptron	99.81%	99.76%	99.48%	98.30%
SNR	24.28	3.87	2.38	1.30

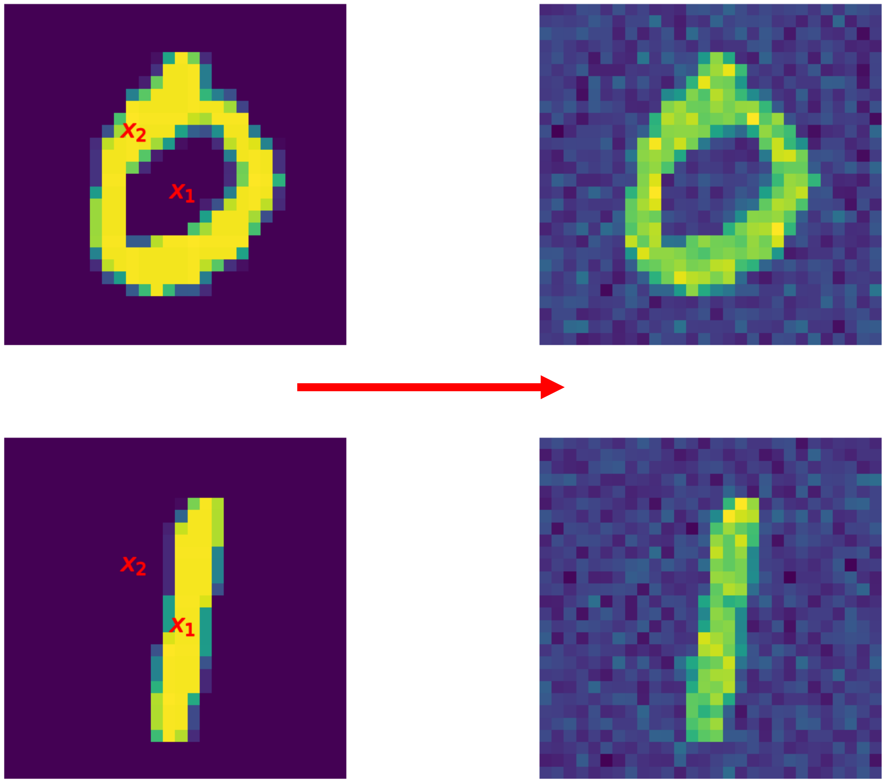


Perceptron	95.41%	90.07%	84.44%	76.88%
SNR	0.85	0.61	0.36	0.26

12665 Training samples

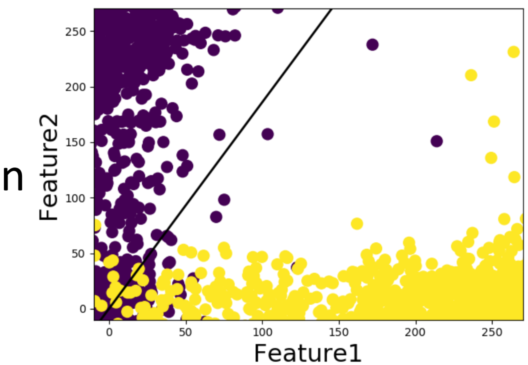
2115 Test samples

Digits: Gaussian noise

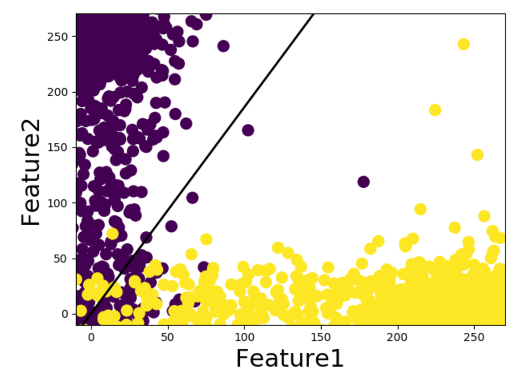


Perceptron - Test	99.78%
ADALINE - Test	99.81%

Perceptron



ADALINE



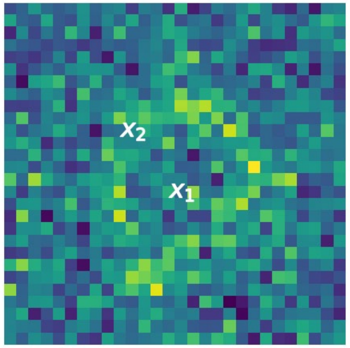
12665 Training samples

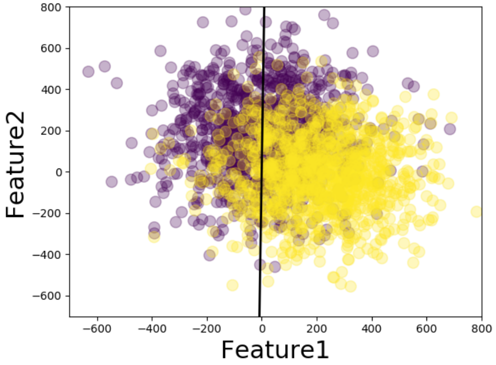
2115 Test samples

Sigmoid activation function is used for ADALINE

Digits: Gaussian noise (perceptron)

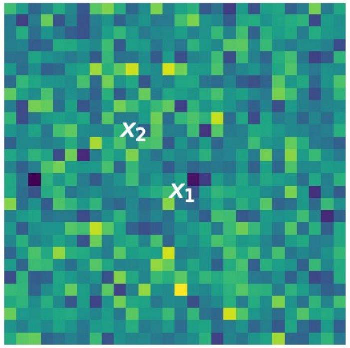
SNR 1.3

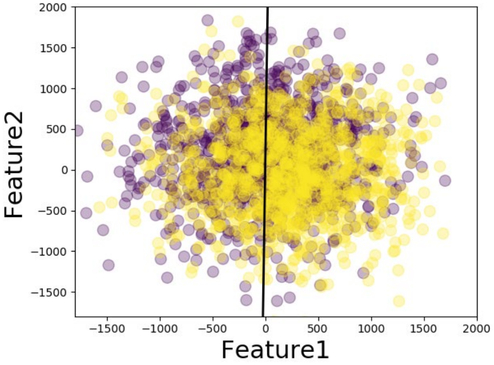




Test accuracy : 98.35

SNR 0.26



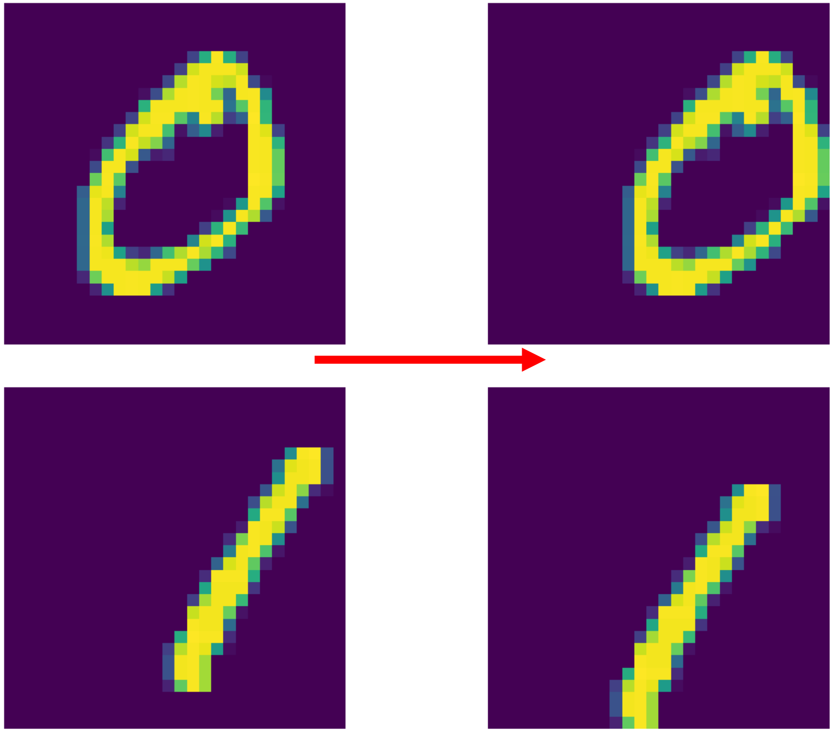


Test accuracy : 79.39

12665 Training samples
2115 Test samples

Digits: Shift

Images in test set are moved randomly in different locations



Perceptron - Test	61.70%
ADALINE - Test	59.95%

Including shifted images in the training

Perceptron - Test	85.72%
ADALINE - Test	88.56%

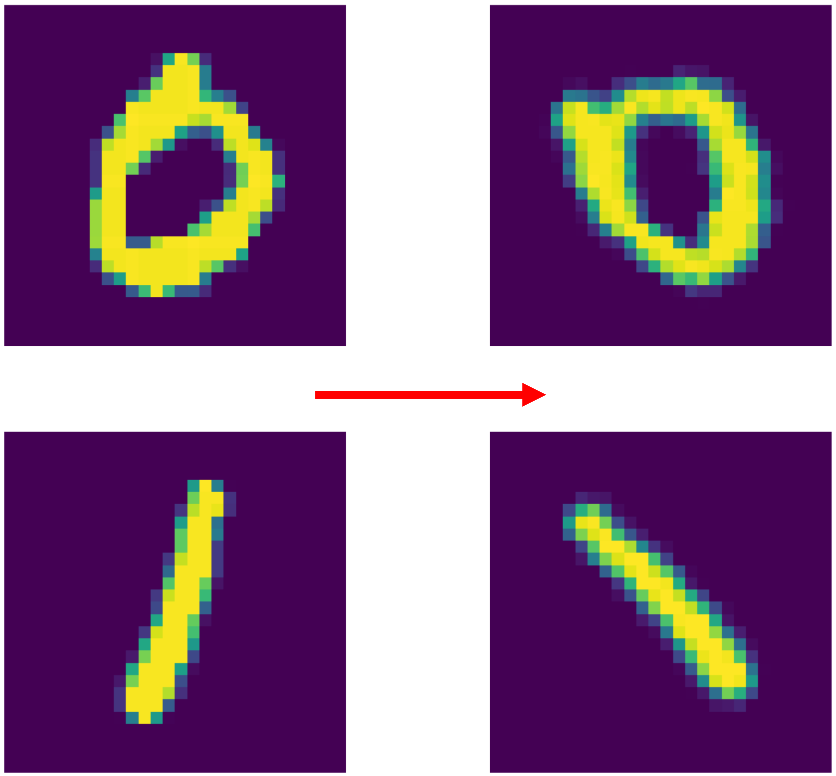
It needs more iteration when the shifted images are included in the training.

12665 Training samples
2115 Test samples
Sigmoid activation function is used for ADALINE

Digits: Rotation (in the test set or the training)

Images in test set are rotated randomly between 60° to 90°.

Perceptron - Test	51.63%
ADALINE - Test	51.54%



Including rotated images in the training

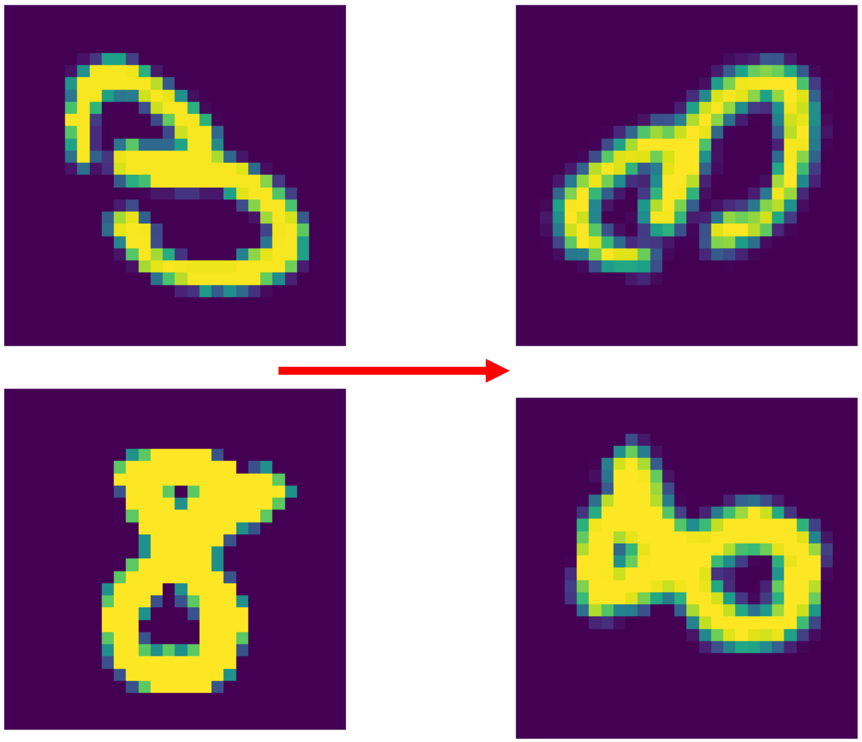
Perceptron - Test	99.67%
ADALINE - Test	99.86%

It needs more iteration when the rotated images are included in the training.

12665 Training samples
2115 Test samples
Sigmoid activation function is used for ADALINE

Digits: Rotation

Images in test set are rotated randomly between 60° to 90°.



Test with rotated images

Perceptron - Test	46.88%
ADALINE - Test	44.41%

Including rotated images in the training

Perceptron - Test	51.06%
ADALINE - Test	48.94%

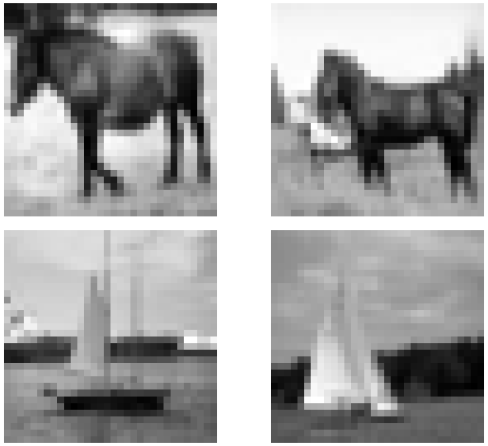
11198 Training samples
1984 Test samples
Sigmoid activation function is used for ADALINE

Direct inversion

Database
1024 training images
2000 test images

Class 1

Class 2



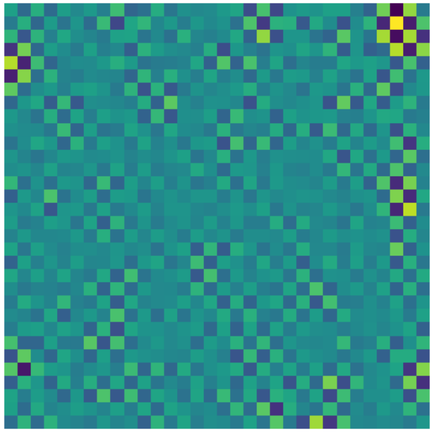
Weight calculation by matrix inversion

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

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

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

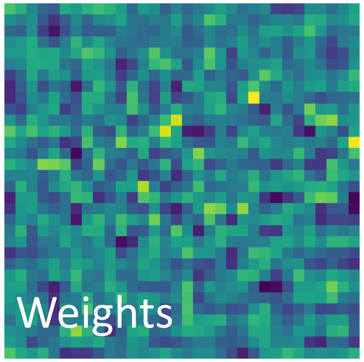
Weights ($\underline{w}$)



Perceptron

Training accuracy: 51.00%
Test accuracy: 50.50%

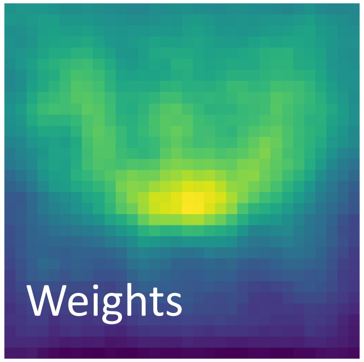
1000 iterations



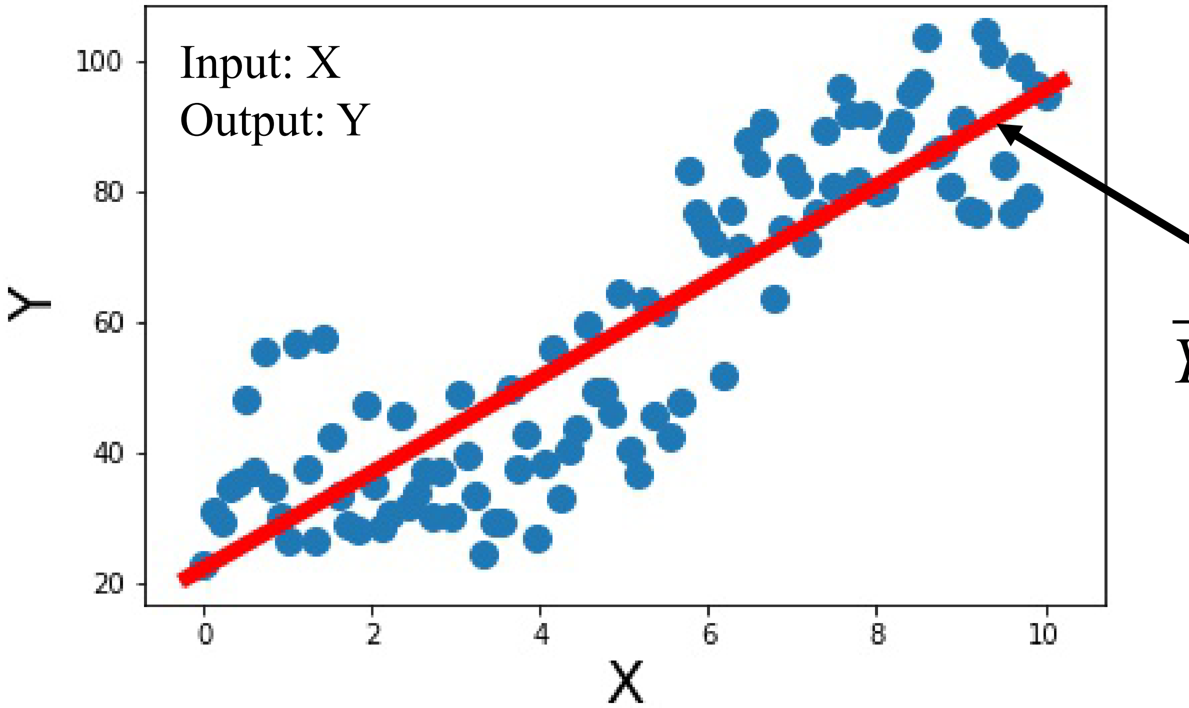
ADALINE

Training accuracy: 73.44%
Test accuracy: 71.50%

Learning rate = 0.0001
2000 iterations



Regression

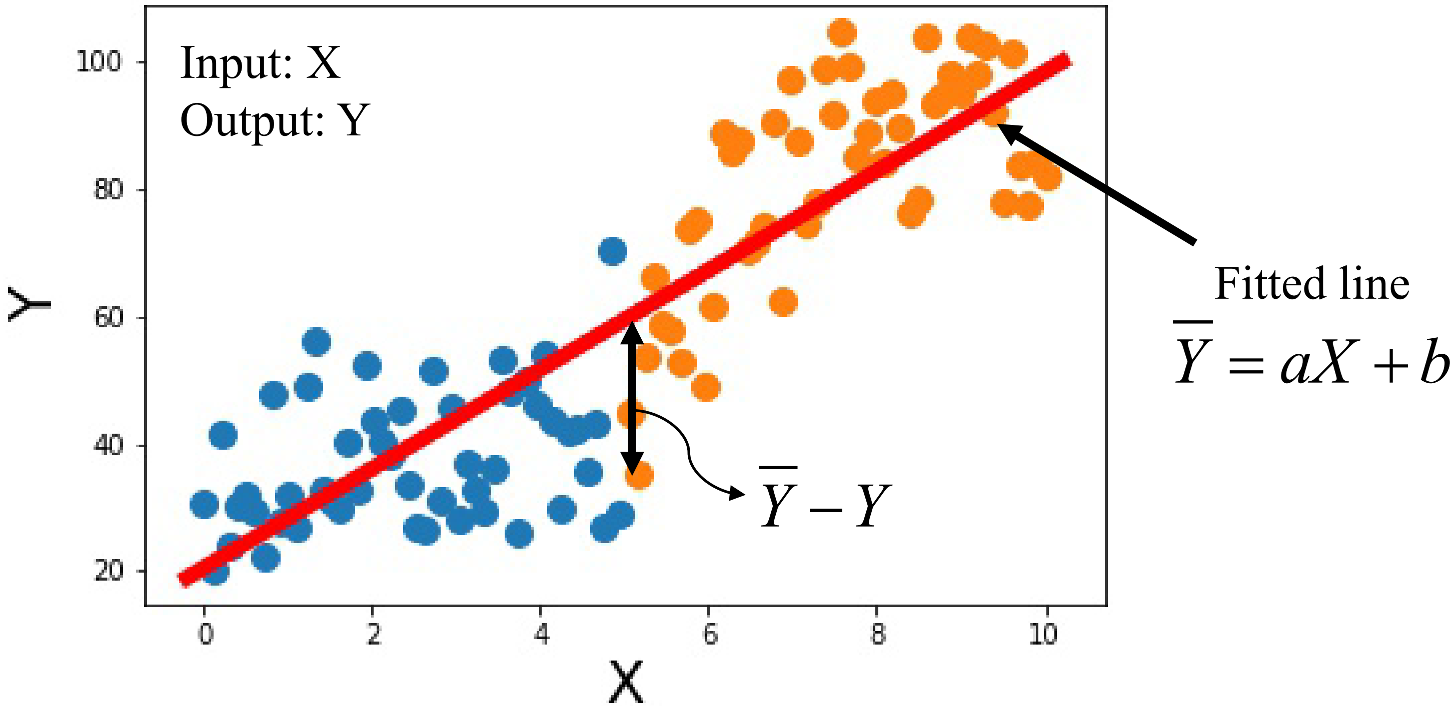


Fitted line

$$\overline{Y} = aX + b$$

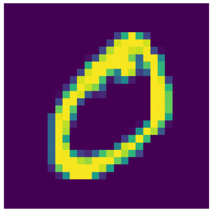
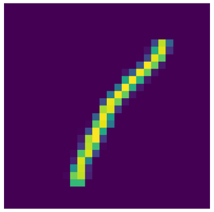
What is the difference between linear regression and ADALINE?

Regression

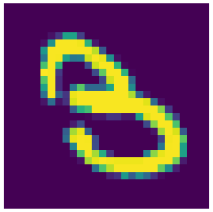
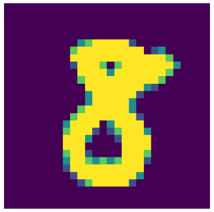


Input: X
Output: Y
Fitted line
$\overline{Y} = aX + b$
$\overline{Y} - Y$
Y
X
100
80
60
40
20
0
2
4
6
8
10

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.