Practical Machine Learning

Lecture 2
Linear models for classification and regression

Dr. Suyong Eum

OSAKA UNIVERSITY

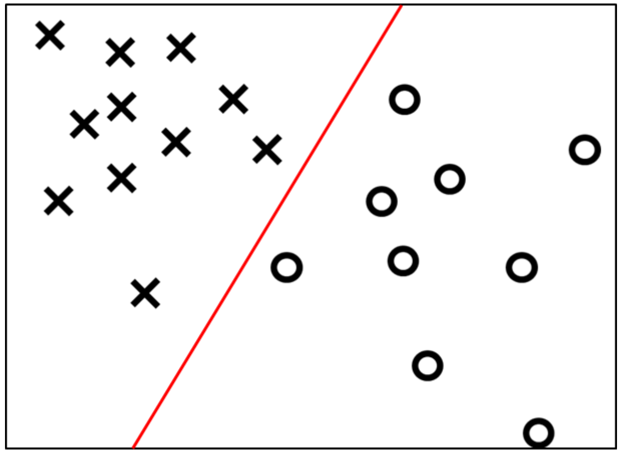
Lecture Outline

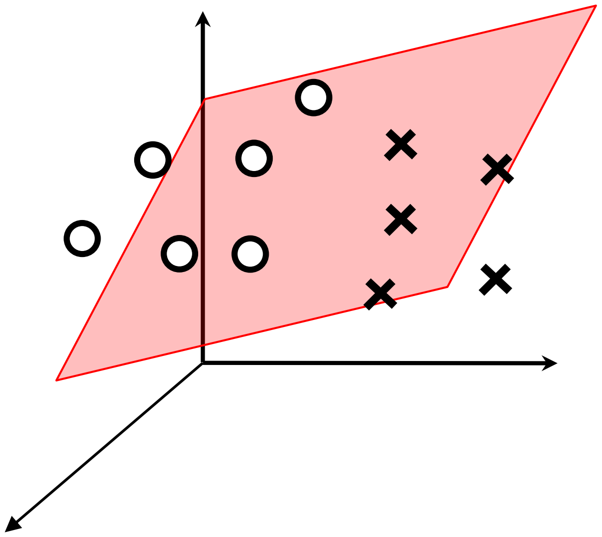
- Linear classification
  - Perceptron algorithm
- Linear regression
  - Mean Square Error (MSE)
  - Normal Equation
  - Gradient descent
- Logistic regression
  - Cross Entropy Error (CEE)

Classification

Terminology

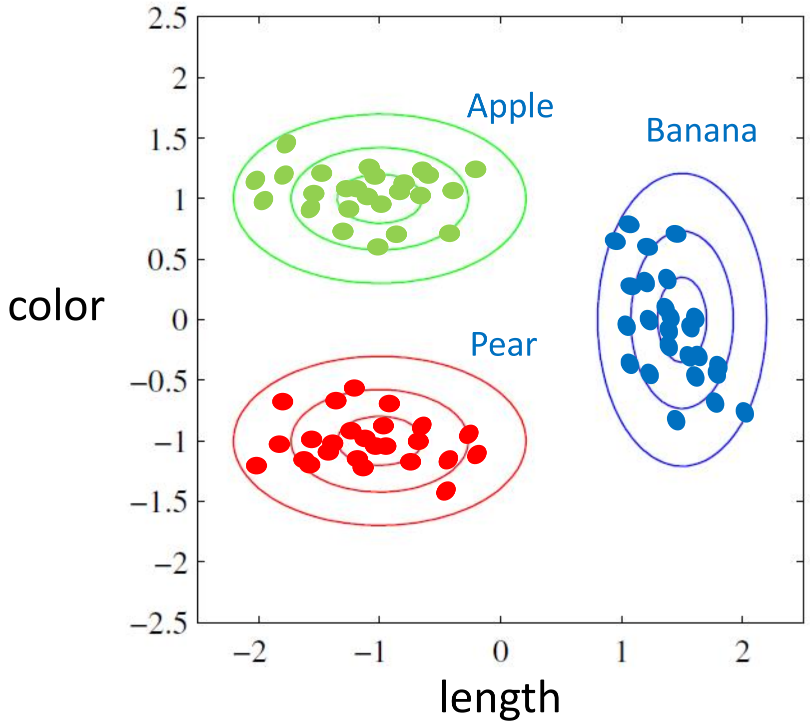
- Decision boundary (surfaces)
- Decision regions
- (D-1)-dimensional hyperplane within the D-dimensional input space

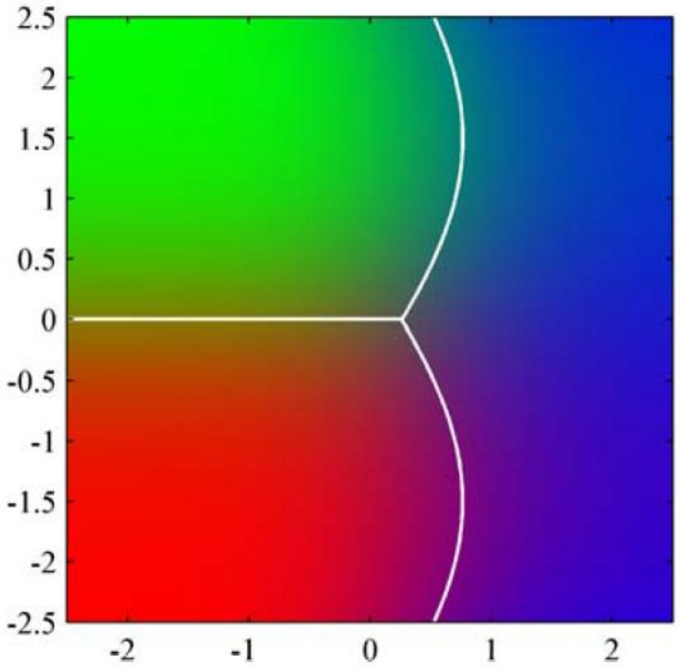




Two main approaches for classification

1) Probabilistic approach
   - Focus on the development of a probability model, e.g., cancer/non-cancer
2) Deterministic approach
   - Focus on the determination of a decision boundary





Christopher M. Bishop Pattern Recognition and Machine Learning, 2001

Perceptron neural network: history

- Rosenblatt (1962) introduced the perceptron algorithm.

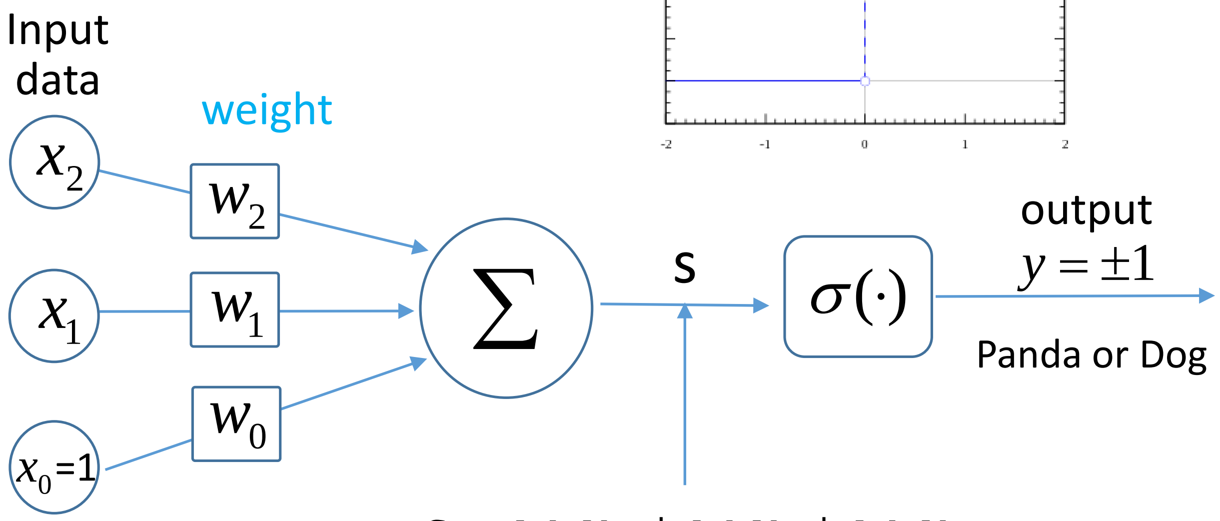
Frank Rosenblatt
1928–1969

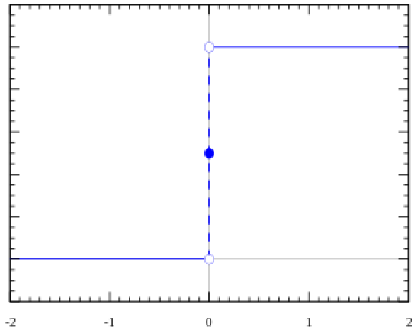
Rosenblatt's perceptron played an important role in the history of machine learning. Initially, Rosenblatt simulated the perceptron on an IBM 704 computer at Cornell in 1957, but by the early 1960s he had built special-purpose hardware that provided a direct, parallel implementation of perceptron learning. Many of his ideas were encapsulated in "Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms" published in 1962. Rosenblatt's work was criticized by Marvin Minksy, whose objections were published in the book "Perceptrons", co-authored with Seymour Papert. This book was widely misinterpreted at the time as showing that neural networks were fatally flawed and could only learn solutions for linearly separable problems. In fact, it only proved such limitations in the case of single-layer networks such as the perceptron and merely conjectured (incorrectly) that they applied to more general network models. Unfortunately, however, this book contributed to the substantial decline in research funding for neural computing, a situation that was not reversed until the mid-1980s. Today, there are many hundreds, if not thousands, of applications of neural networks in widespread use, with examples in areas such as handwriting recognition and information retrieval being used routinely by millions of people.

Christopher M. Bishop Pattern Recognition and Machine Learning, 2001

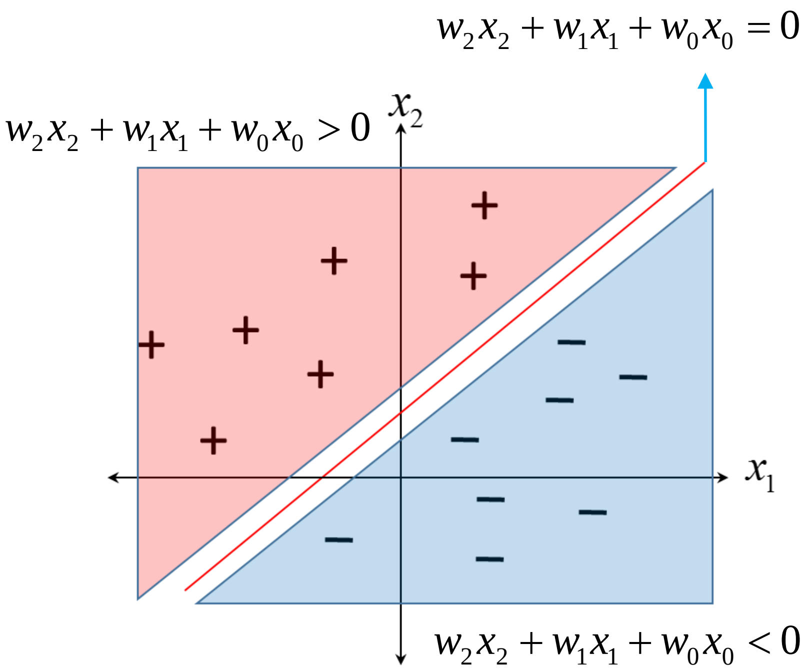
Perceptron neural network: architecture

- Simplest form of a neural network used for a linear classification.





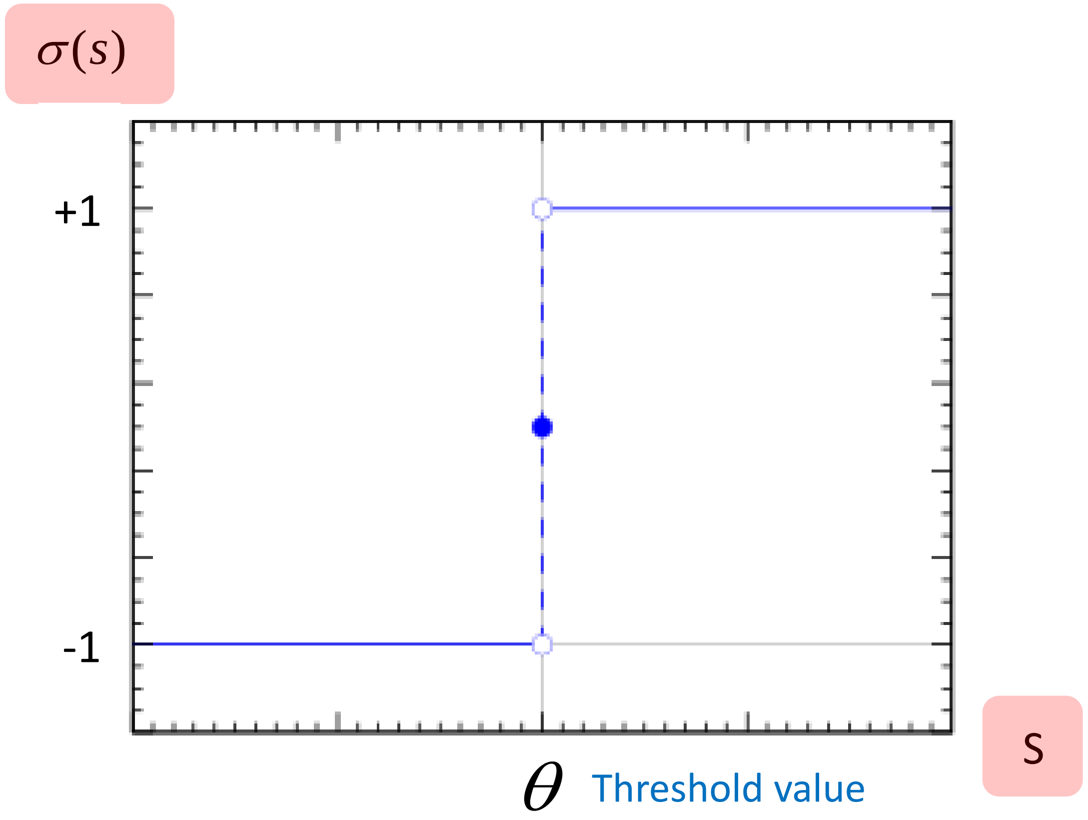
$$s = w_2 x_2 + w_1 x_1 + w_0 x_0$$



Perceptron neural network: architecture

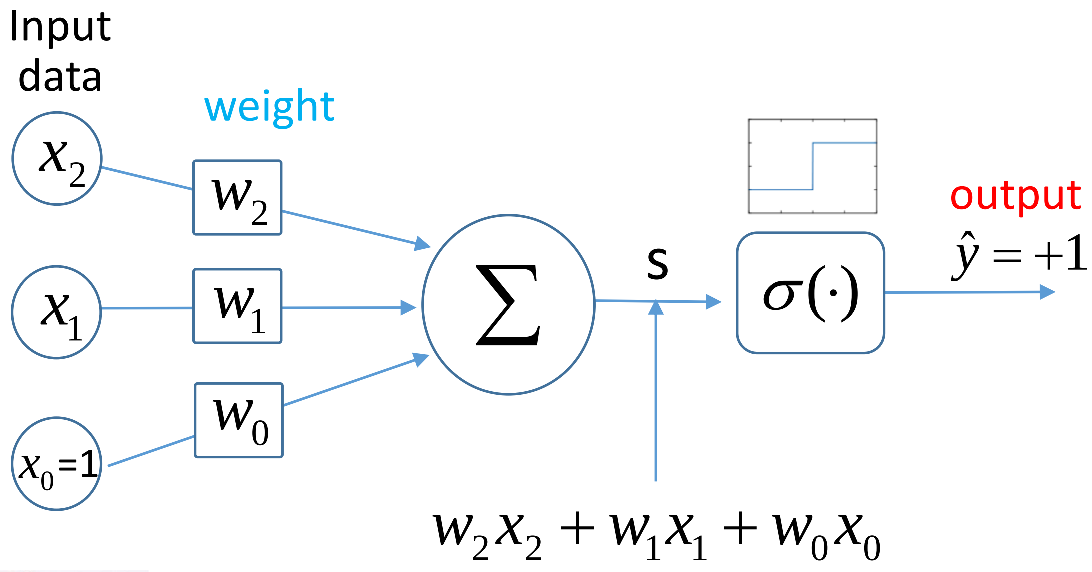
- It uses a step function as an activation function: Yes/No question.
- The step function is discontinuous: any problem?

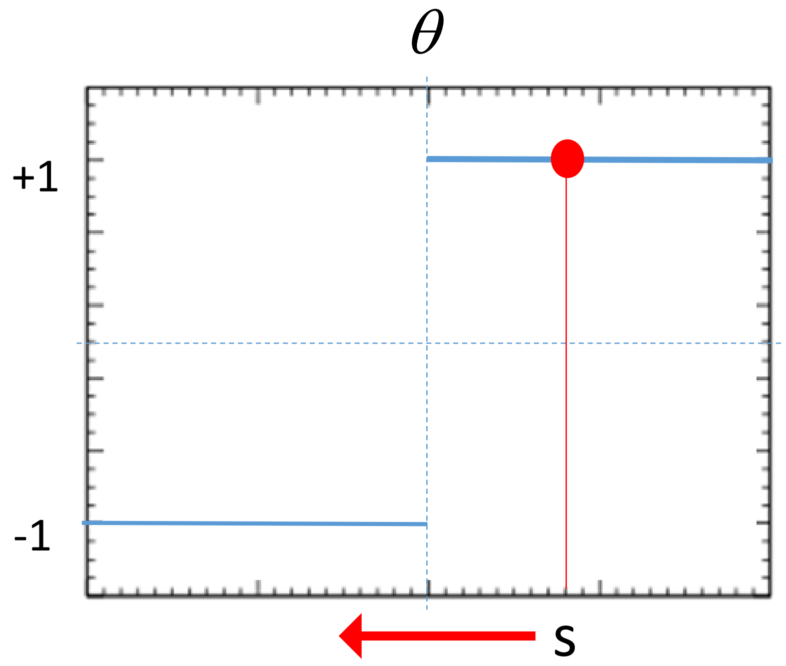
$$\sigma(s) = \begin{cases} +1 & if \;\; s > \theta \\ -1 & otherwise \end{cases}$$



Perceptron neural network: how to update parameters?

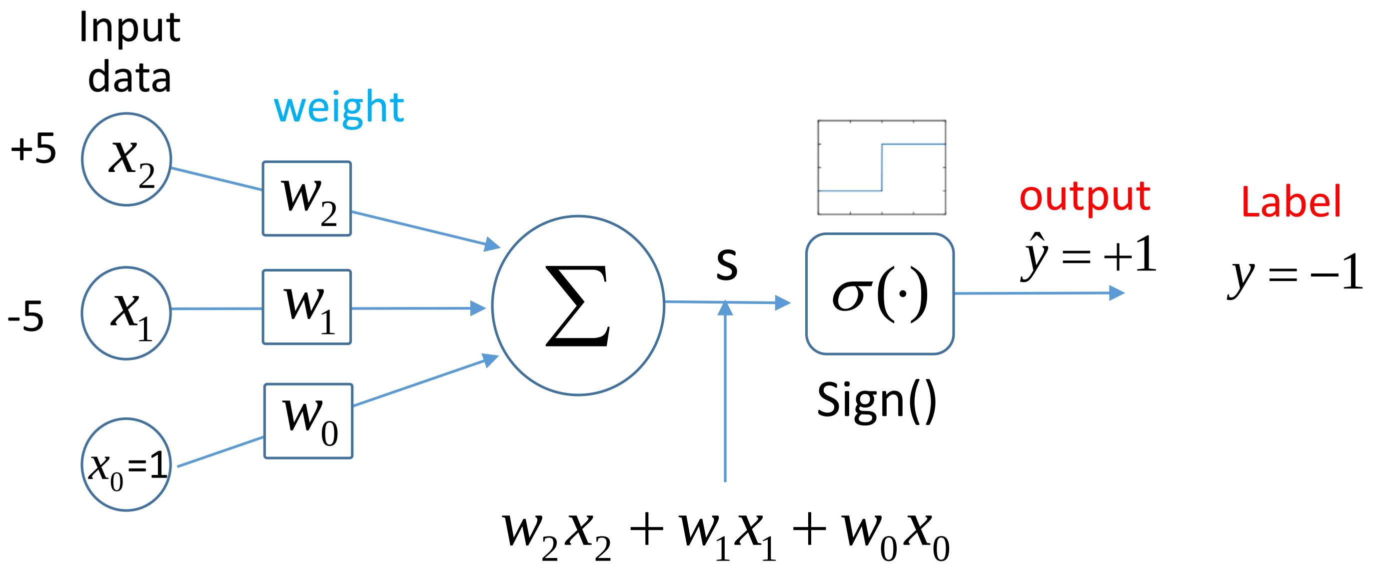
- Assuming that a data point X (*x1, x2*) has a label (-1)
- Assuming that it is misclassified to be (+1)
- Weight should be updated.

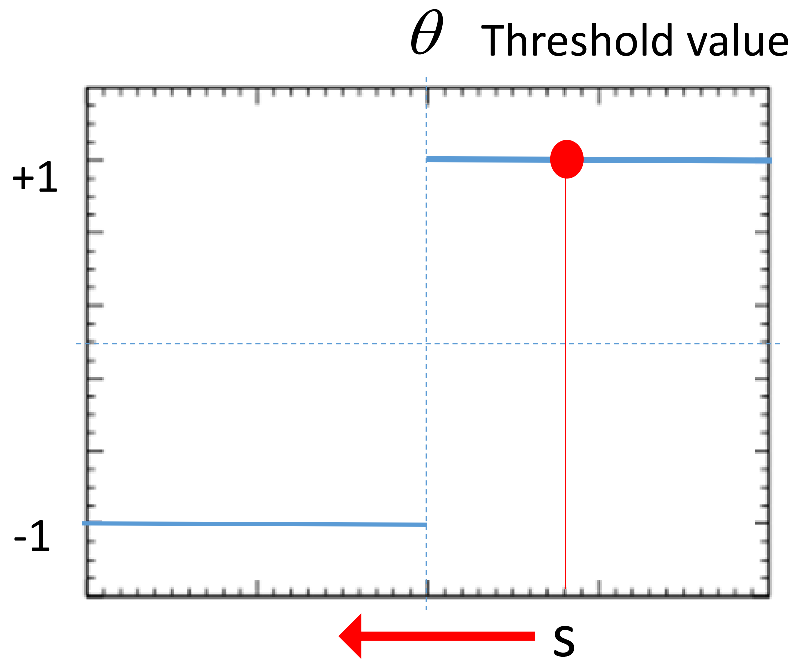




$$\mathrm{w}^{(\mathrm{new})} = \mathrm{w}^{(\mathrm{old})} + \gamma \cdot y_n \cdot \mathrm{x}_\mathrm{n}$$

Perceptron neural network: parameter update





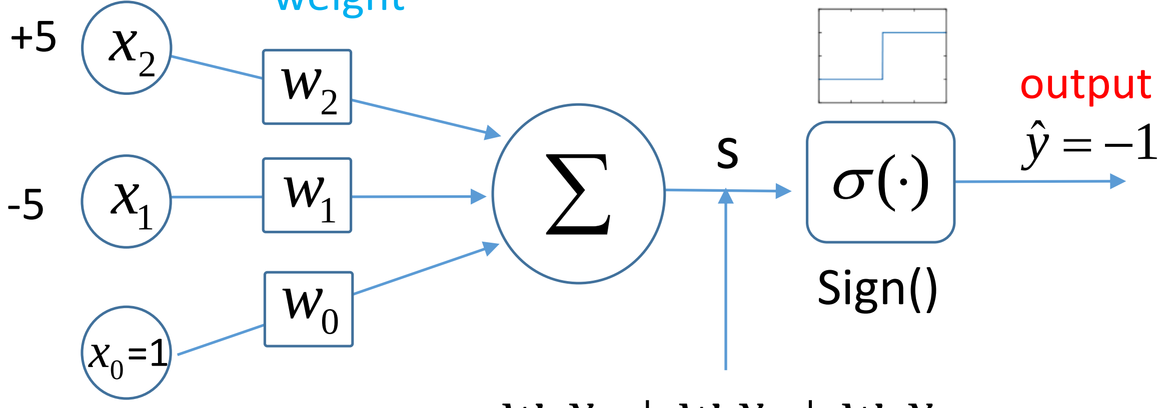
$$\mathrm{w}^{(\mathrm{new})} = \mathrm{w}^{(\mathrm{old})} + \gamma \cdot y_n \cdot \mathrm{x}_\mathrm{n}$$

$$w_2^{(\mathrm{new})} = w_2^{(\mathrm{old})} + \gamma \cdot (-1) \cdot (x_2 = 5)$$

$$w_1^{(\mathrm{new})} = w_1^{(\mathrm{old})} + \gamma \cdot (-1) \cdot (x_1 = -5)$$

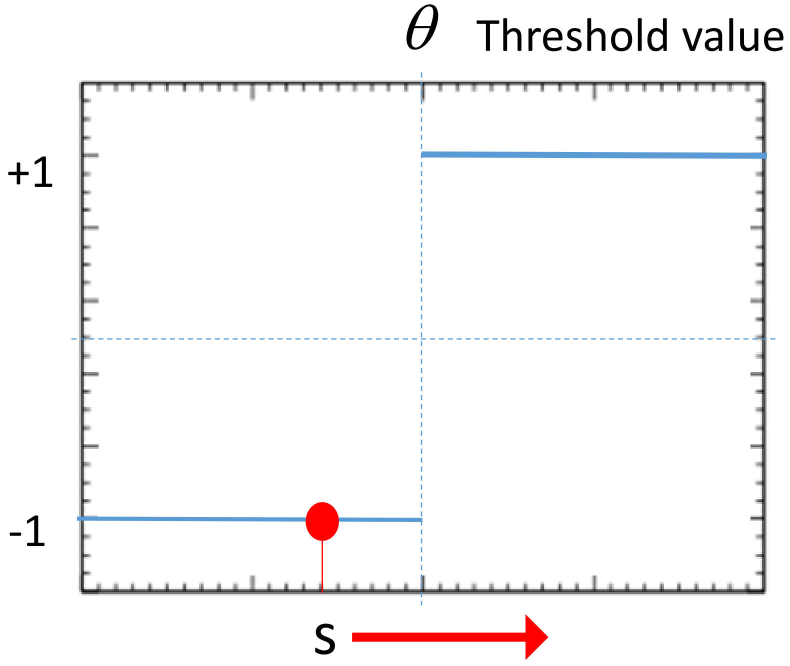
$$w_0^{(\mathrm{new})} = w_0^{(\mathrm{old})} + \gamma \cdot (-1) \cdot (x_0)$$

Perceptron neural network: parameter update



$\hat{y} = -1$ $\quad y = +1$

$$w_2 x_2 + w_1 x_1 + w_0 x_0$$



$$\mathrm{w}^{(\mathrm{new})} = \mathrm{w}^{(\mathrm{old})} + \gamma \cdot y_n \cdot \mathrm{x}_\mathrm{n}$$

$$w_2^{(\mathrm{new})} = w_2^{(\mathrm{old})} + \gamma \cdot (+1) \cdot (x_2 = 5)$$

$$w_1^{(\mathrm{new})} = w_1^{(\mathrm{old})} + \gamma \cdot (+1) \cdot (x_1 = -5)$$

$$w_0^{(\mathrm{new})} = w_0^{(\mathrm{old})} + \gamma \cdot (+1) \cdot (x_0)$$

Perceptron neural network: parameter update

- You will see another type of update rule in a literature, which is same.

$$\mathrm{w}^{(\mathrm{new})} = \mathrm{w}^{(\mathrm{old})} + \gamma \cdot (\hat{y}_n - y_n) \cdot \mathrm{x}_\mathrm{n}$$

$$\mathrm{n} \in A$$

Where “A” denotes the set of all data points

$$\mathrm{w}^{(\mathrm{new})} = \mathrm{w}^{(\mathrm{old})} + \gamma \cdot y_n \cdot \mathrm{x}_\mathrm{n}$$

$$\mathrm{n} \in \mathrm{M}$$

Where “M” denotes the set of all misclassified patterns

Perceptron neural network: algorithm procedure

1) Random initialization of parameters

$$\text{random} \sim w=\{w_0, w_1, w_2, \ldots, w_d\}$$

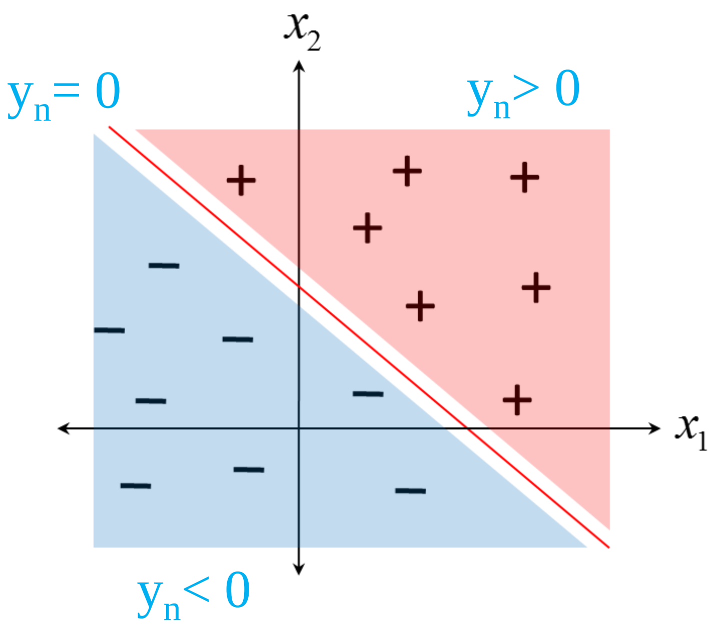
2) Searching misclassified data points using the decision boundary (y_n)

$$y_n = w_0x_0 + w_1x_1 + w_2x_2 + \ldots + w_dx_d$$

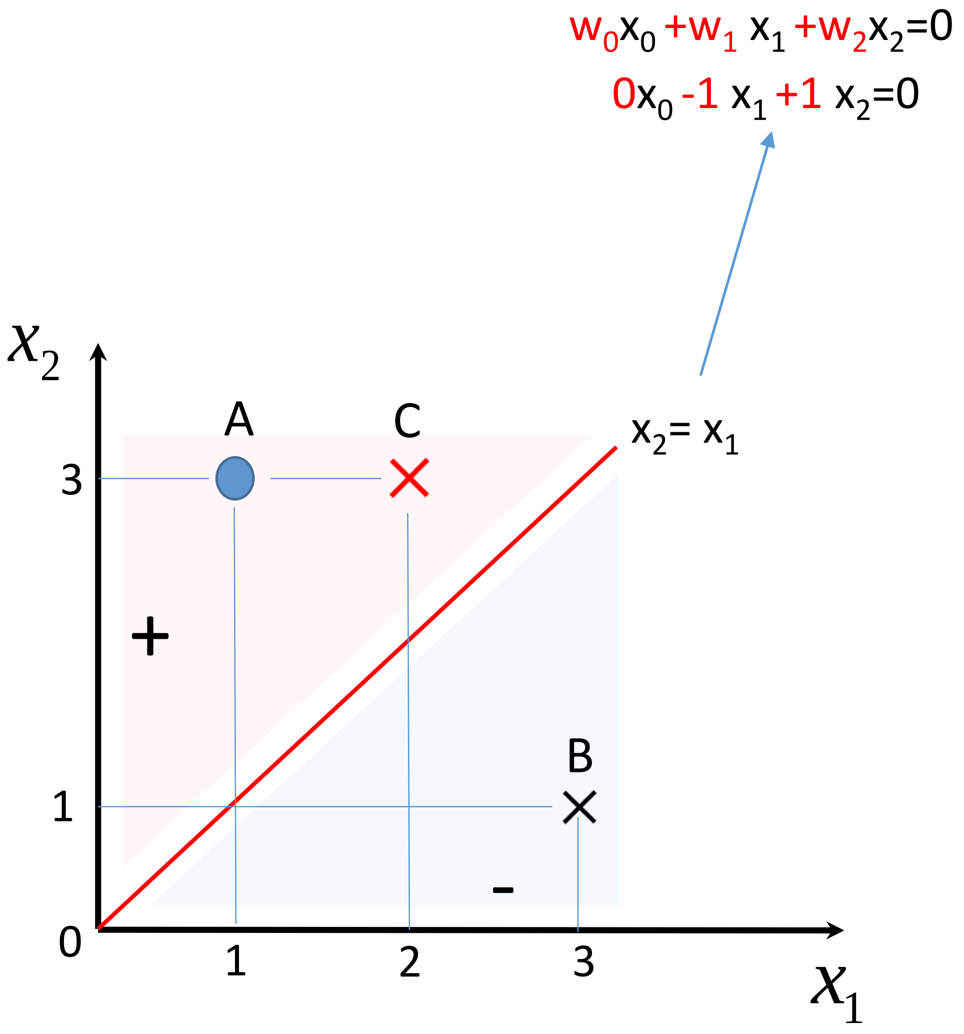
2-1) Updating the parameters using the misclassified data points

$$w^{(new)} = w^{(old)} + \gamma \cdot y_n \cdot x_n$$

2-2) Go to step 2)



Perceptron neural network: example



- Current decision boundary is determined with w=(0, -1, +1)
- Data point x_C (1, 2, 3) is misclassified
- Learning rate: $\gamma = 0.1$
- Let's update the parameter

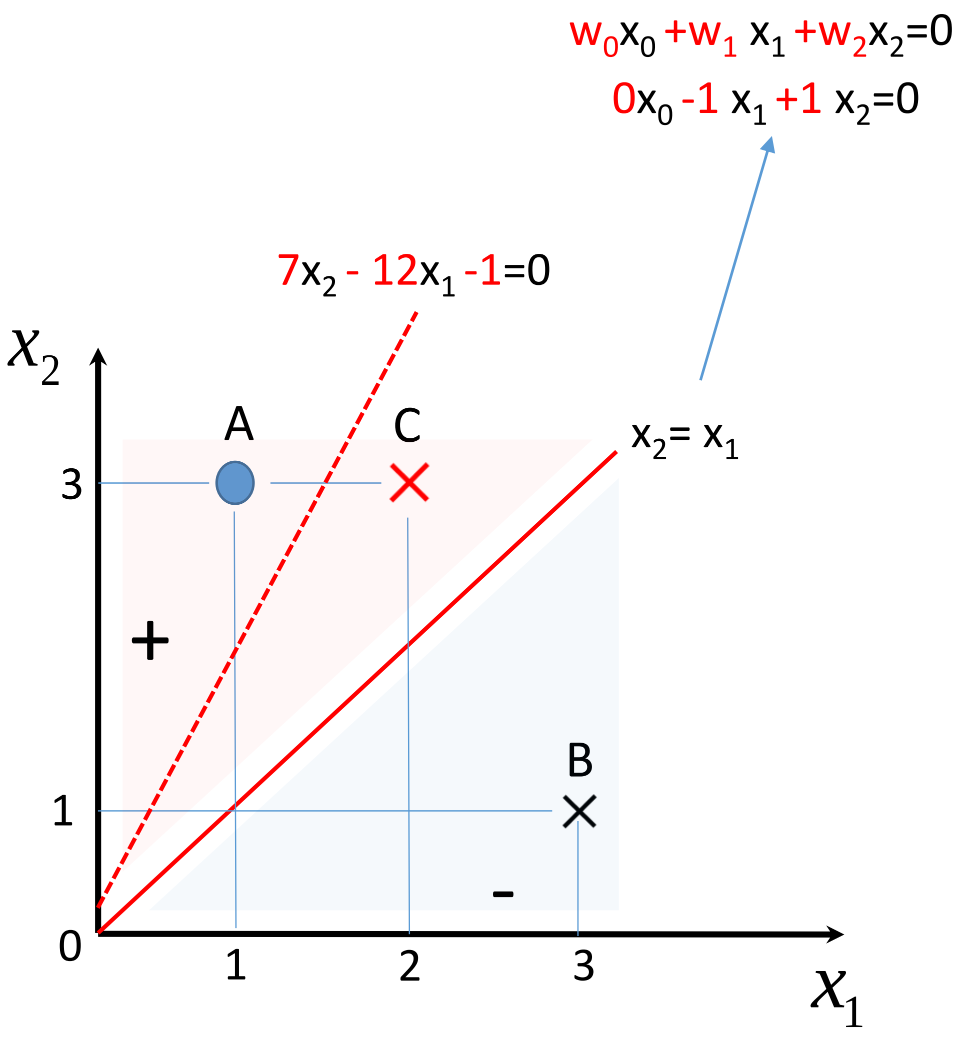
$w=(0, -1, +1)$ $\quad$ $x_c=(1, 2, 3)$

$$\mathrm{w}^{(\mathrm{new})} = \mathrm{w}^{(\mathrm{old})} + \gamma \cdot y_n \cdot \mathrm{x}_\mathrm{n}$$

$$= (0,-1,+1) + (0.1)(-1)(1,2,3)$$

$$= (-0.1, \quad -1.2, \quad 0.7)$$

Perceptron neural network: example



- Current decision boundary is determined with w=(0, -1, +1)
- Data point x_C (1, 2, 3) is misclassified
- Learning rate: γ = 0.1
- Let's update the parameter

w=(0, -1, +1) x_c=(1, 2, 3)

$$\mathrm{w}^{(\mathrm{new})} = \mathrm{w}^{(\mathrm{old})} + \gamma \cdot y_n \cdot \mathrm{x}_\mathrm{n}$$

$$= (0,-1,+1) + (0.1)(-1)(1,2,3)$$

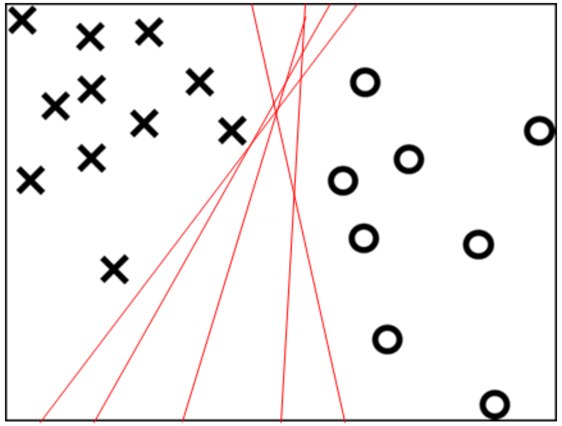
$$= (-0.1, \quad -1.2, \quad 0.7)$$

- New decision boundary is

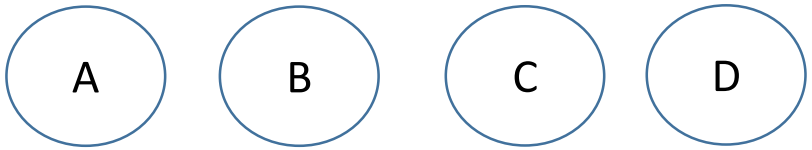
$7x_2 - 12x_1 - 1 = 0$

Perceptron neural network: convergence theorem

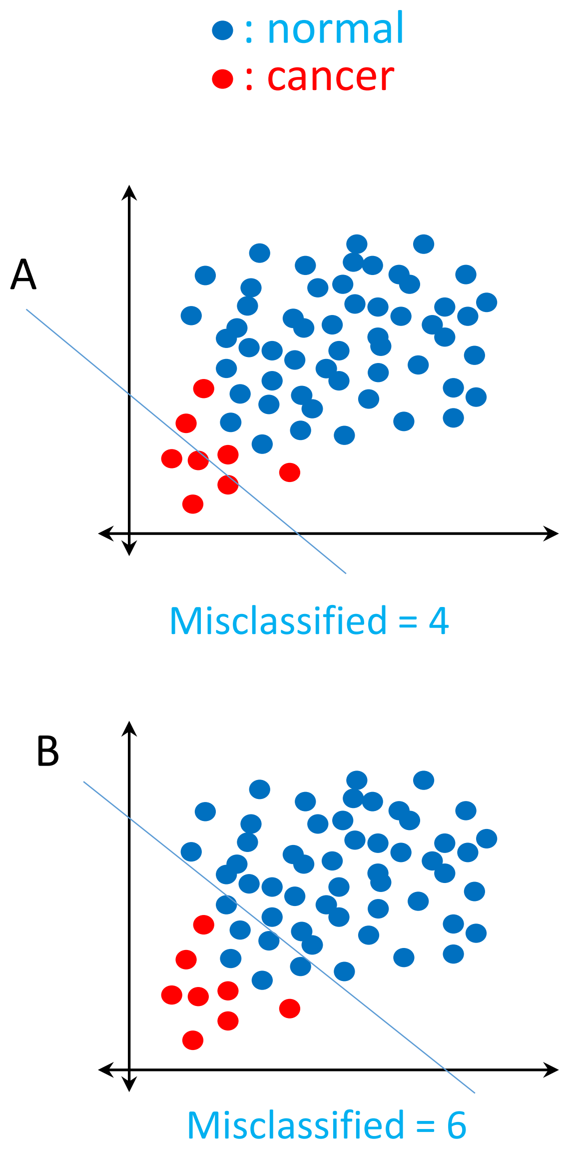
- The perceptron algorithm is guaranteed to find an exact solution within a finite number of iteration if given data set is linearly separable.
  - Slow convergence: cannot tell its feasibility until it's convergence
  - Does not converge if there is not any solution
  - Many solutions exist: converge to one depending on an initial and the order of data feeding



Perceptron neural network: multiclass classification



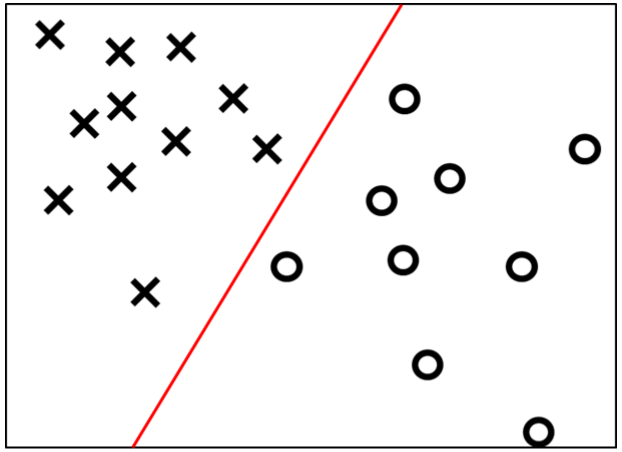
	One to One	One to rest
# of binary classifications to be carried out (when there is K classes)	❑ K(K-1)/2 - A to B - A to C - A to D - B to C - B to D - C to D	❑ K - A to B/C/D - B to A/C/D - C to A/B/D - D to A/B/C
Feature	❑ Complexity high	❑ Class imbalance problem

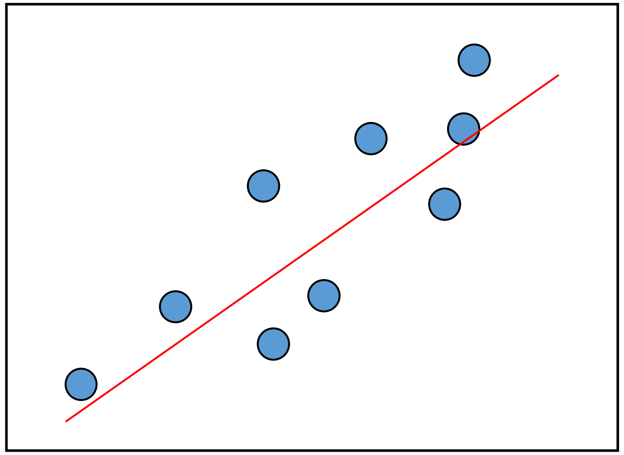


Regression

Regression

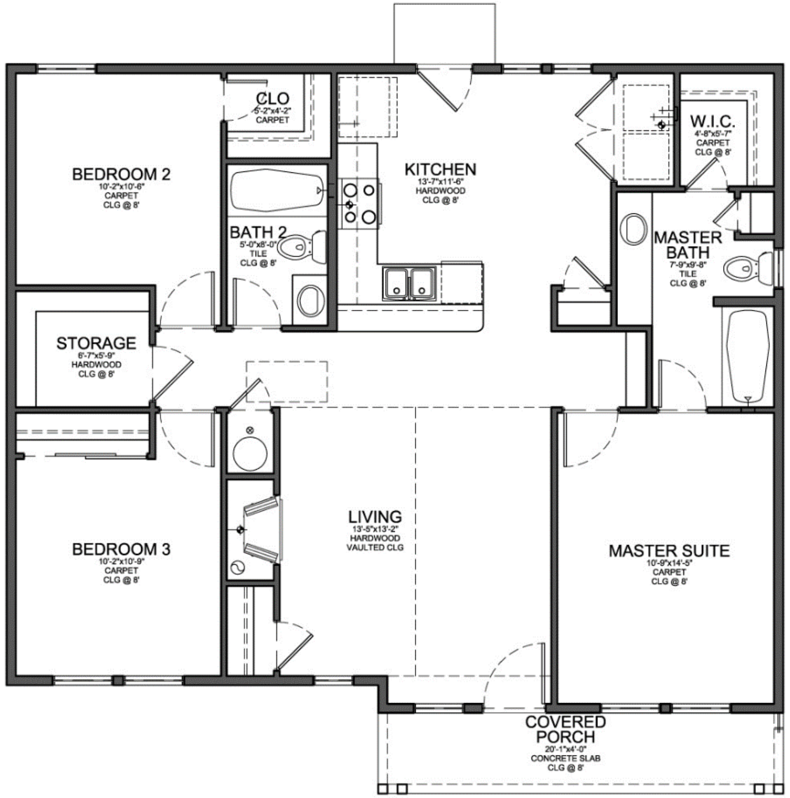
- Given data set, regression fits them to a certain function
- Classification vs Regression
  - Classification: given a data, its output is a discretized label
  - Regression: given a data, its output is a continuous value
  - Both belongs to supervised learning: input + label

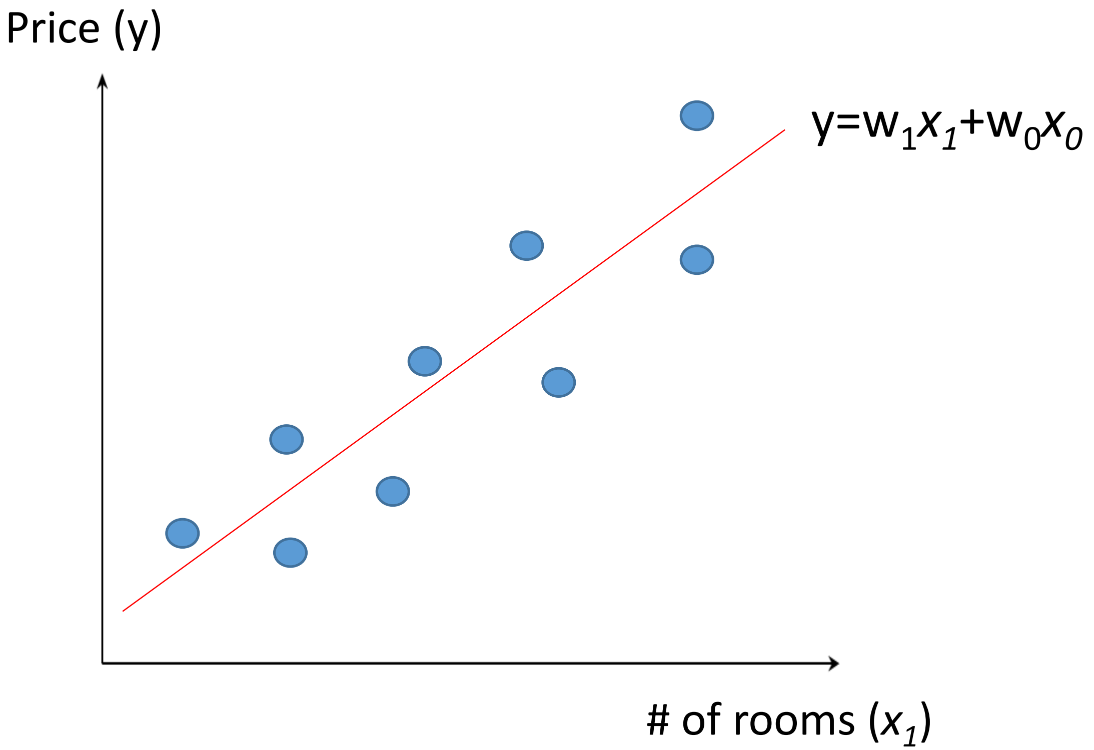




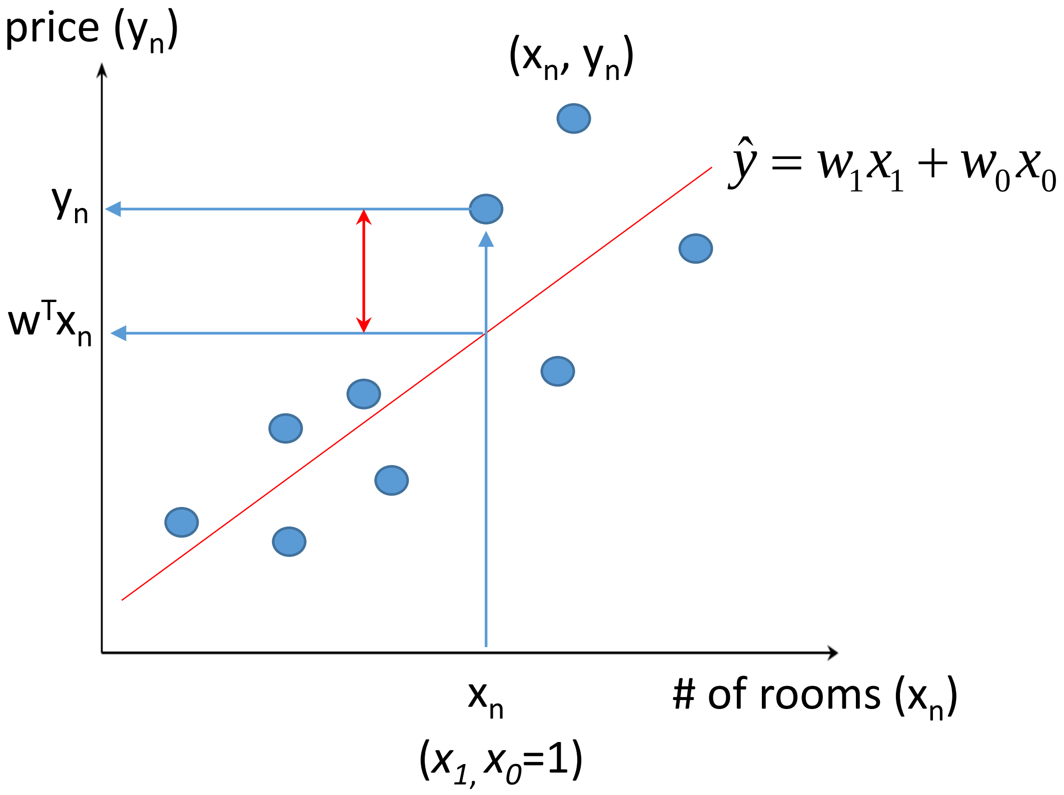
Linear regression

- How much is the house below?

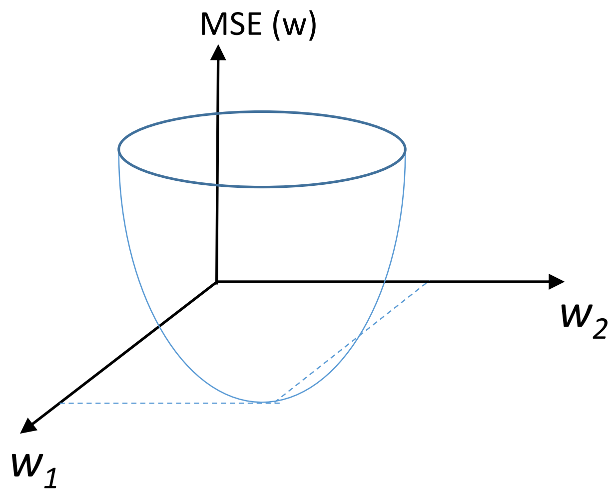




Linear regression: Mean Square Error (MSE) function



$$MSE(w) = \frac{1}{2}\sum_{n=1}^{N}(w^T x_n - y_n)^2$$



1) Analytical approach: normal equations
2) Systematical approach: gradient descent

Linear regression: 1) normal equation

$$MSE(w) = \frac{1}{2}(XW - Y)^T(XW - Y)$$

(1x1) = [(nxm)(mx1)-(nx1)]$^{-T}$ [(nxm)(mx1)-(nx1)]

$$= \frac{1}{2}((XW)^T - Y^T)(XW - Y)$$

$$= \frac{1}{2}(W^T X^T - Y^T)(XW - Y)$$

$$= \frac{1}{2}(W^T X^T XW - Y^T XW - W^T X^T Y + Y^T Y)$$

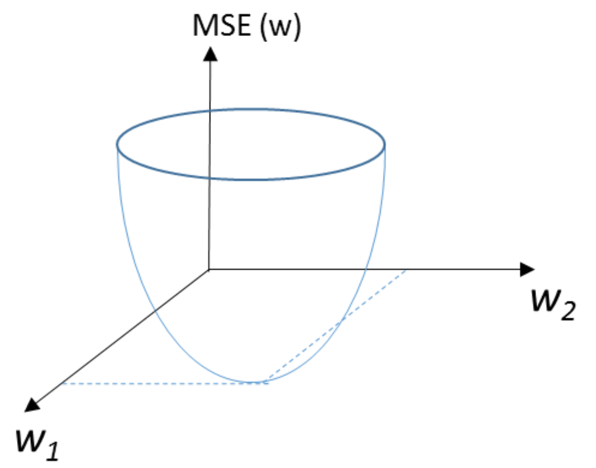
$Y^T XW = (W^T X^T Y)^T$

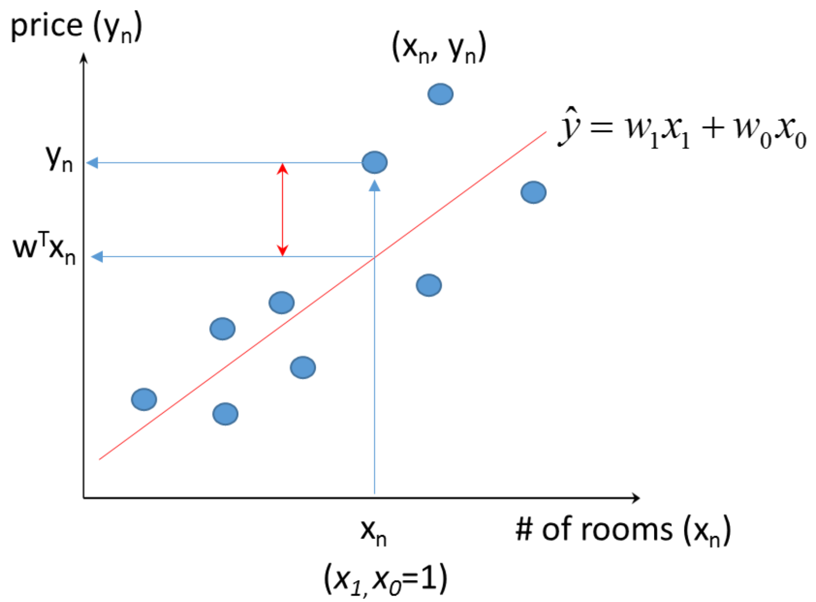
$$= \frac{1}{2}(W^T X^T XW - 2W^T X^T Y + Y^T Y)$$

$$\frac{dE}{dW^T} = \frac{1}{2}(2X^T XW - 2X^T Y) = 0$$

$$X^T XW = X^T Y \qquad W = (X^T X)^{-1} X^T Y$$

(mx1) [(mxn)(nxm)]$^{-1}$(mxn)(nx1)





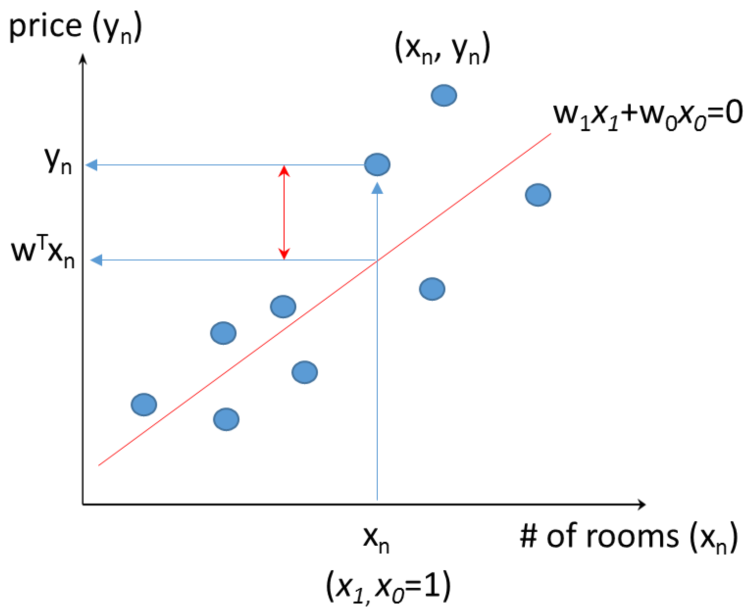
Linear regression: 1) normal equation - example

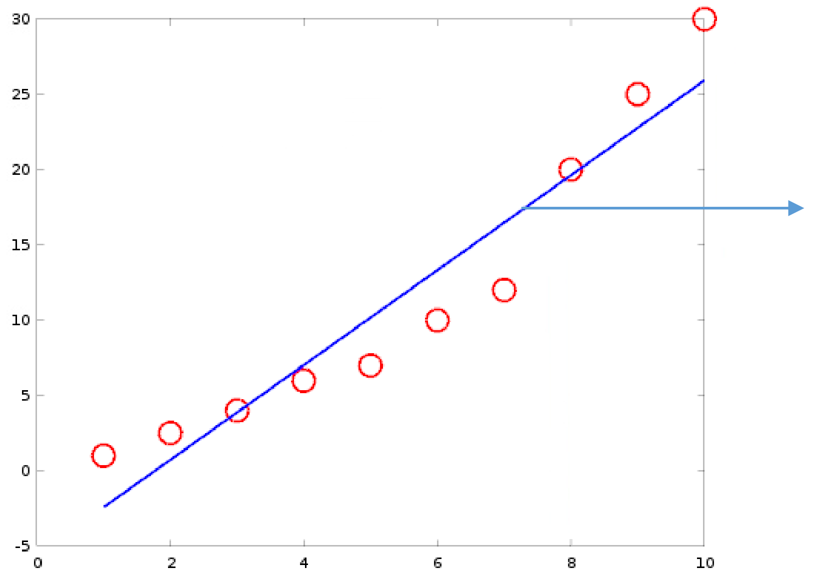
$$W = (X^T X)^{-1} X^T Y$$

(mx1) [(mxn)(nxm)]$^{-1}$(mxn)(nx1)

```
>> xt
xt =

    1     2     3     4     5     6     7     8     9    10
    1     1     1     1     1     1     1     1     1     1

>> yt
yt =

    1.0  2.5  4.0   6.0   7.0  10.0  12.0   20.0  25.0  30.0

>> w = inverse(xt*x)*xt*y
w =

   3.1485
  -5.5667
```

- ❑ transpose()
- ❑ Inverse()

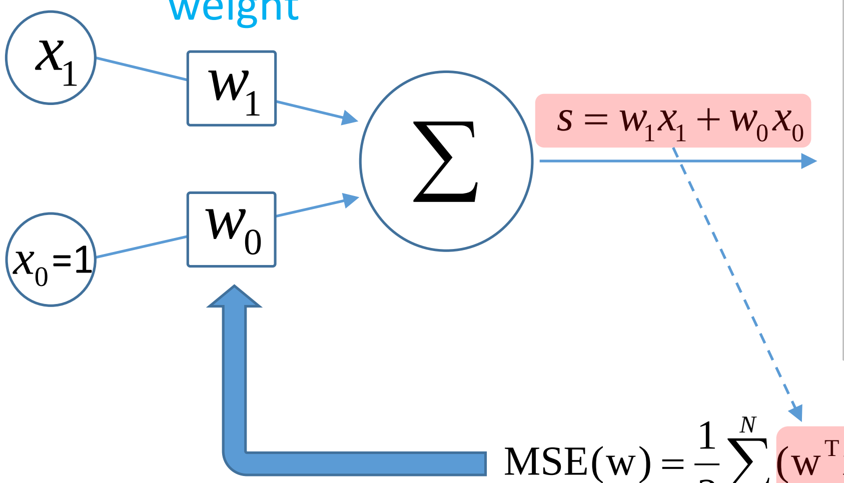




$3.1485\ x_1$ -5.5667

Input data

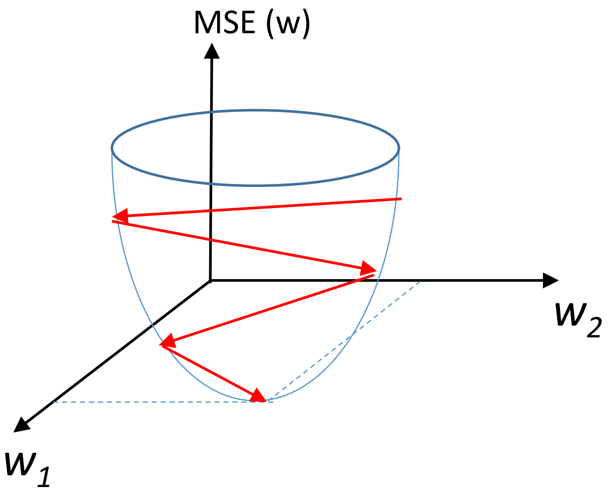
```
>> x
x =

     1     1
     2     1
     3     1
     4     1
     5     1
     6     1
     7     1
     8     1
     9     1
    10     1
```



Label: y_n

```
>> transpose(y)
ans =

    1.0000
    2.5000
    4.0000
    6.0000
    7.0000
   10.0000
   12.0000
   20.0000
   25.0000
   30.0000
```

$$\mathrm{MSE}(\mathrm{w}) = \frac{1}{2}\sum_{n=1}^{N}(\mathrm{w}^T \mathrm{x}_n - \mathrm{y}_n)^2$$



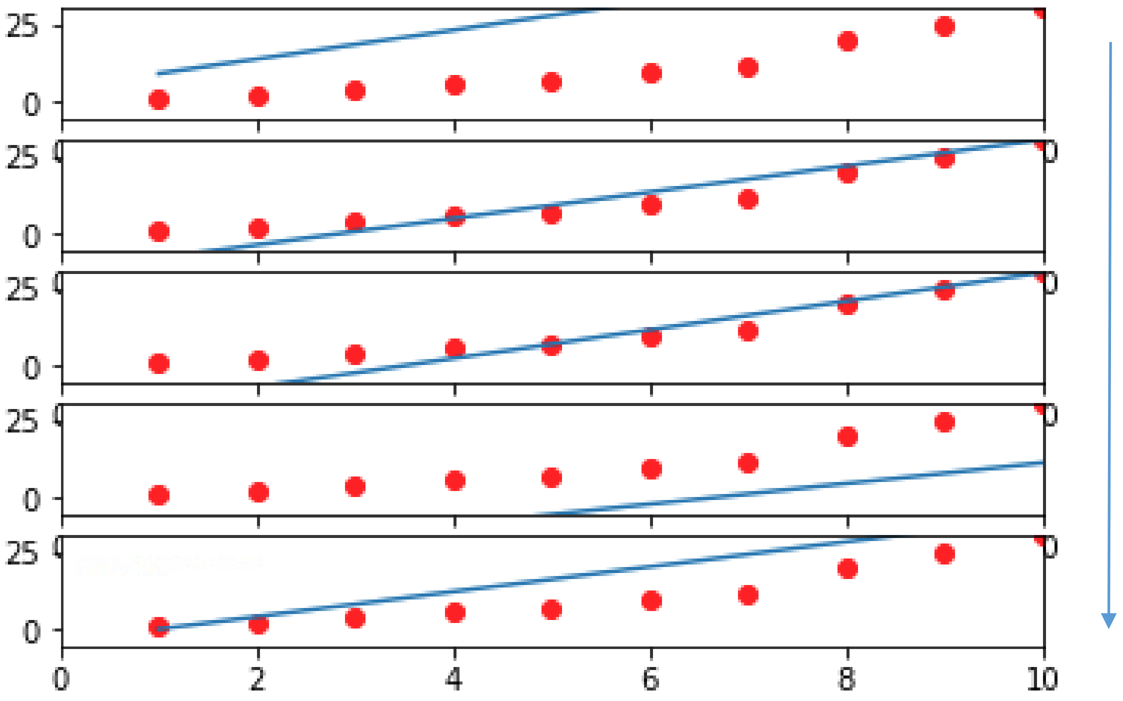
$$w_1 = w_1 + \gamma \frac{\partial E(\mathrm{w})}{\partial w_1}$$

$$w_2 = w_2 + \gamma \frac{\partial E(\mathrm{w})}{\partial w_2}$$

How to obtain?
backpropagation

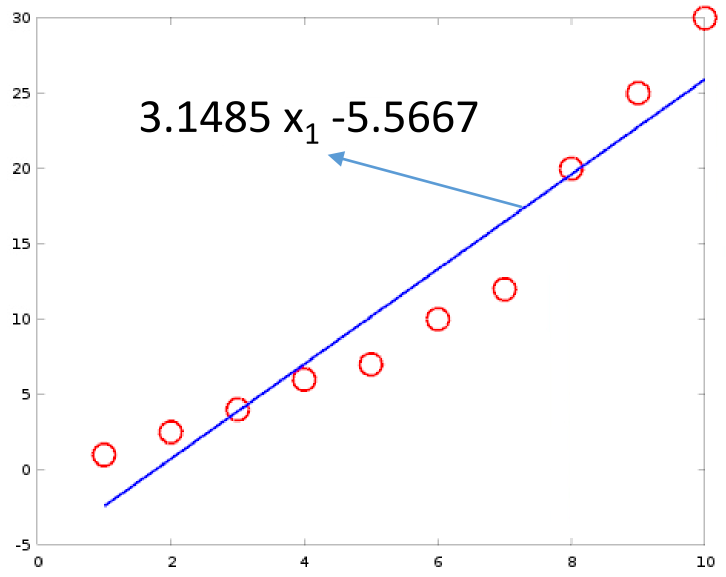
$$w_1 = w_1 + \gamma \frac{\partial E(\mathrm{w})}{\partial w_1} = w_1 + \gamma \left[\frac{\partial E(\mathrm{w})}{\partial s} \frac{\partial s}{\partial w_1} \right]$$

Linear regression: 2) gradient descent



iteration

Solution from
gradient descent



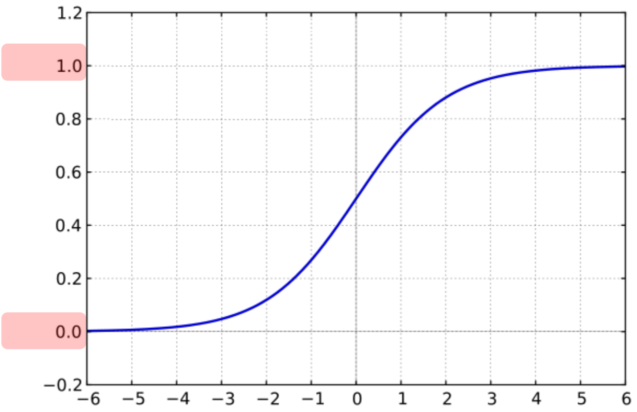
Solution from
normal equation

Logistic regression

- ❑ A logistic function is used to fit the given binary data set.
- ❑ A logistic function is a common "S" shape function.

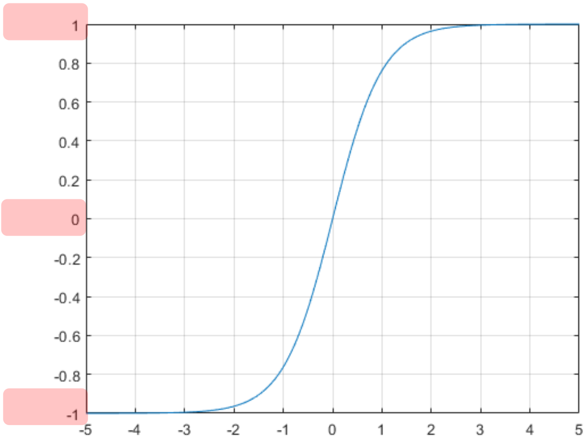
$$\sigma(x) = \frac{L}{1 + e^{-a(x+b)}}$$

- L: maximum value of the function
- a: steepness of the curve
- b: location of the midpoint



sigmoid function

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

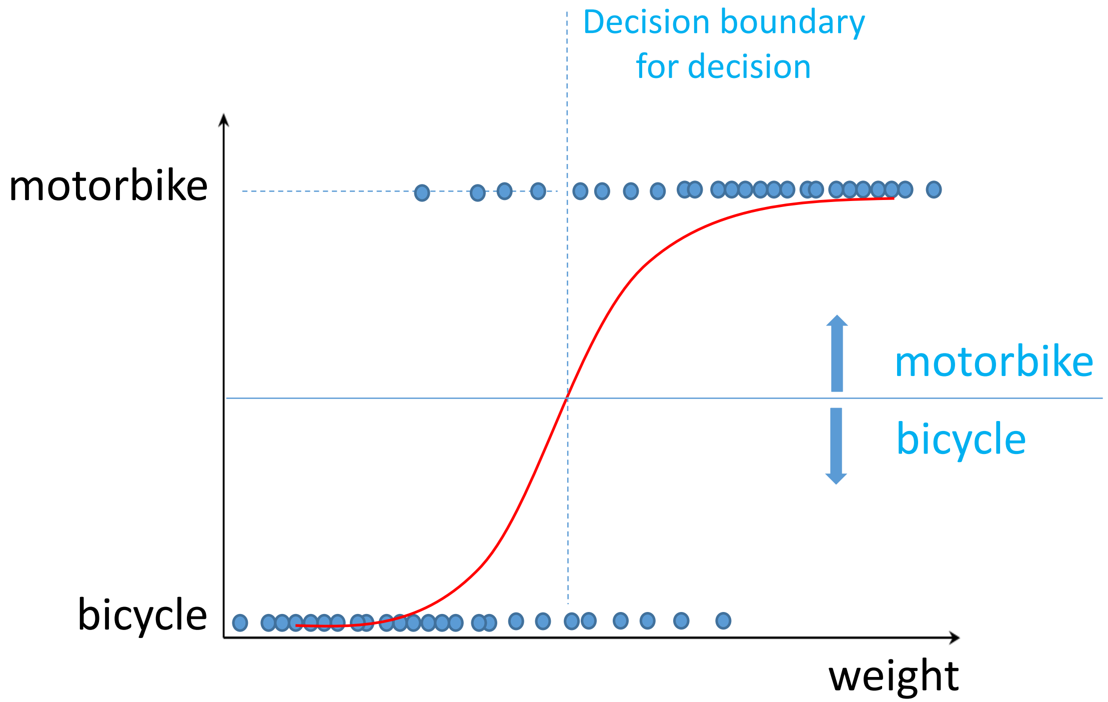


Hyper tangent function

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

Logistic regression

- ❑ Is it a bicycle or motorbike?



Logistic regression: cross entropy error function

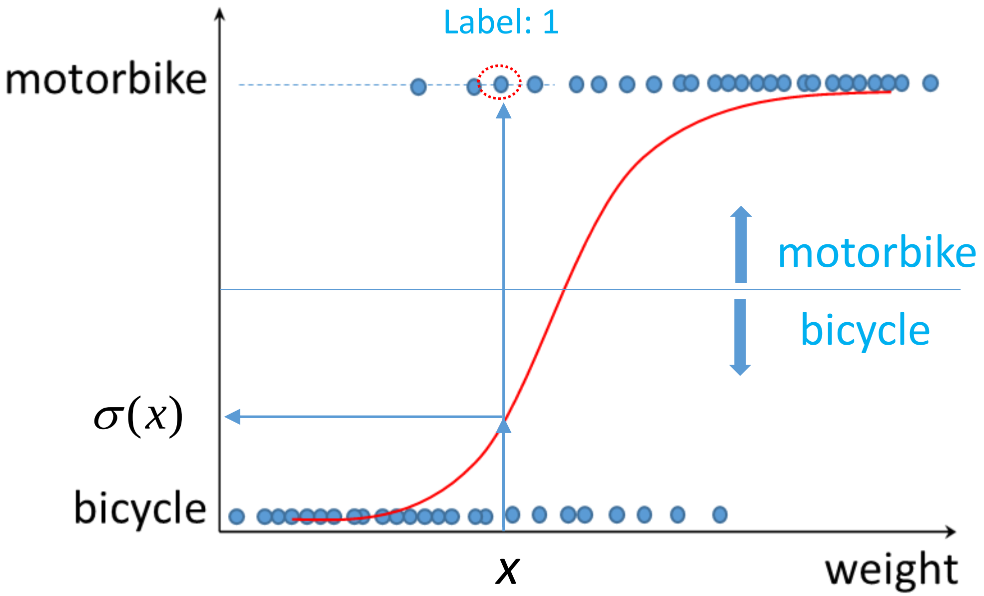
- How to calculate the error given below?
  - Label is one or zero
  - Outcome $\sigma(x)$ is a floating number between [0, 1]

$$\sigma(x) = \frac{1}{1+e^{-a(x+b)}}$$

Label : $y = (1, 0)$

prediction : $\hat{y} = (\sigma(x), 1-\sigma(x))$

$$\mathrm{CEE}(y, \hat{y}) = -\sum y \log \hat{y}$$



Logistic regression: cross entropy error function

$$CEE(y,\hat{y}) = -\sum y \log \hat{y}$$

prediction : $\hat{y} = (\sigma(x), 1-\sigma(x))$

Label : $y = (1,\ 0)$

When the prediction is correct, how correct is it?

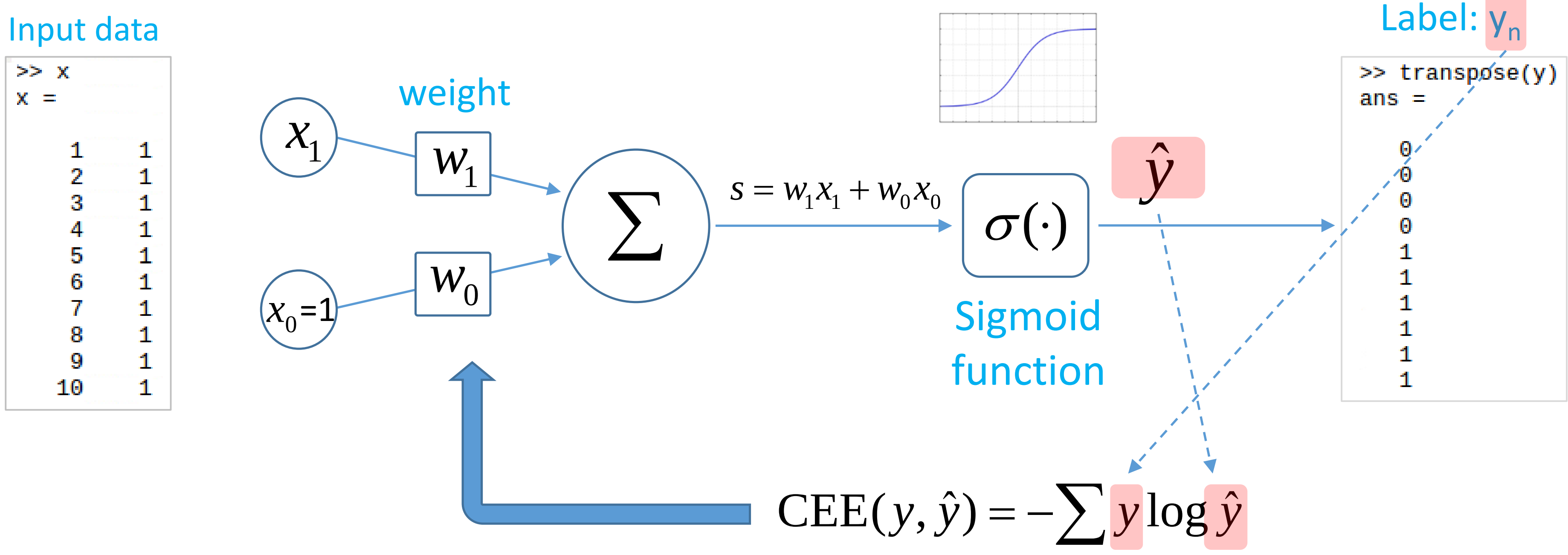
Prediction	Label	Cross Entropy (error)	MSE
0.1, 0.2, 0.7	0, 0, 1	-ln(0.1)*0-ln(0.2)*0-ln(0.7)*1 = **0.357**	$(0.1-0)^2+(0.2-0)^2+(0.7-1)^2$ = **0.14**
0.3, 0.3, 0.4	0, 0, 1	-ln(0.3)*0-ln(0.3)*0-ln(0.4)*1 = 0.916	$(0.3-0)^2+(0.3-0)^2+(0.4-1)^2$ = 0.54

When the prediction is wrong, how wrong is it?

Prediction	Label	Cross Entropy (error)	MSE
0.1, 0.2, 0.7	1, 0, 0	-ln(0.1)*1-ln(0.2)*0-ln(0.7)*0 = **2.303**	$(0.1-1)^2+(0.2-0)^2+(0.7-0)^2$ = **1.34**
0.3, 0.3, 0.4	1, 0, 0	-ln(0.3)*1-ln(0.3)*0-ln(0.4)*0 = 1.204	$(0.3-1)^2+(0.3-0)^2+(0.4-0)^2$ = 0.74

Logistic regression: how to estimate the parameters?

$$\sigma(x) = \frac{1}{1+e^{-s}} = \frac{1}{1+e^{-w_1 x_1 - w_0 x_0}}$$

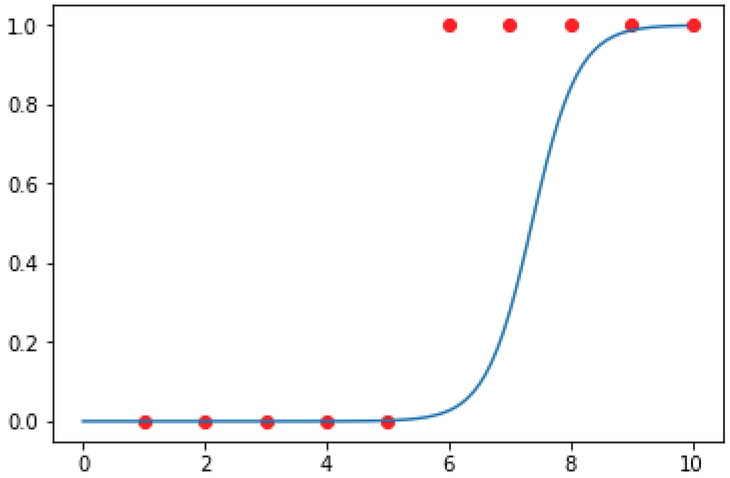


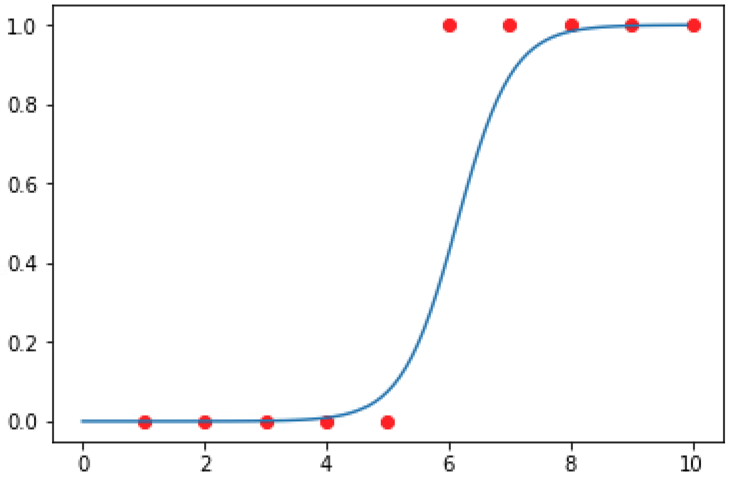
Same story
backpropagation

$$w_1 = w_1 + \gamma \frac{\partial E(\mathrm{w})}{\partial w_1} = w_1 + \gamma \left[\frac{\partial E(\mathrm{w})}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial s} \frac{\partial s}{\partial w_1} \right]$$

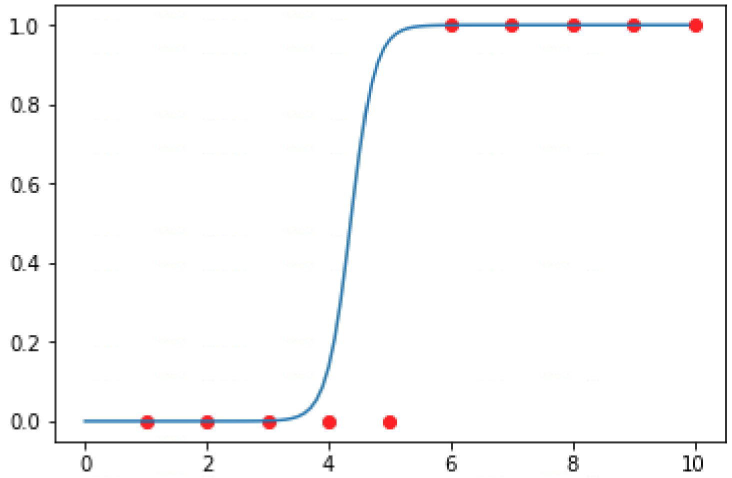
Logistic regression: how to estimate the parameters?

t=10

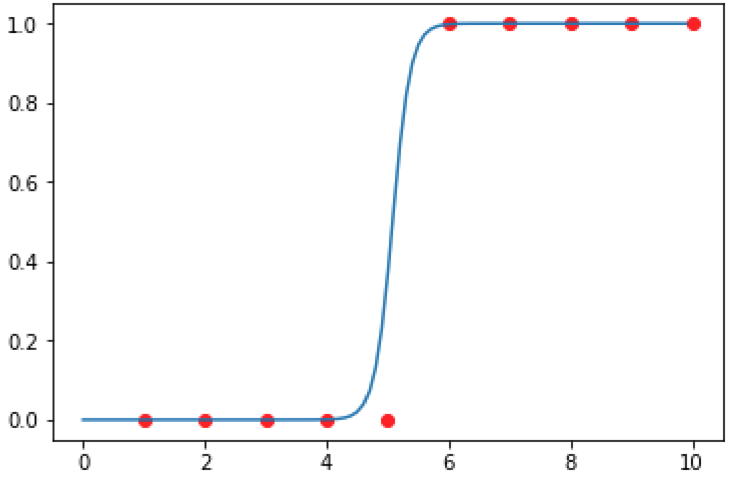


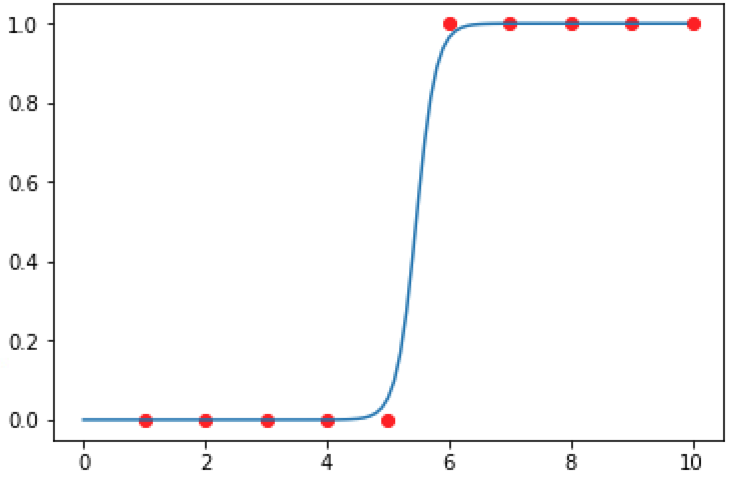


t=50



t=100





t=1000