Lecture 03
Regression

Prof. Ph.D. Woo Youn Kim

Contents

- Linear regression
- Multivariate linear regression
- Classification
- Logistic regression
- Softmax regression

모두를 위한 머신러닝/딥러닝 강의
https://hunkim.github.io/ml/

Andrew Ng lecture note@Standford

Supervised learning

- Several types of learning algorithms

- **Supervised learning**
  - Teach the computer how to do something, then let it use it’s new found knowledge to do it

- **Unsupervised learning**
  - Let the computer learn how to do something, and use this to determine structure and patterns in data

- **Reinforcement learning**

Linear regression

Regression

Let us start by talking about a few examples of supervised learning problems.

Living area ($feet^2$)	Price (1000$s)
2104	400
1600	330
2400	369
1416	232
3000	540
⋮	⋮

It can also be

Molecular structure	Solubility
	Energy
	HOMO-LUMO gap
	Photo Voltaic Efficiency
	Biological activity

How to fit the data?

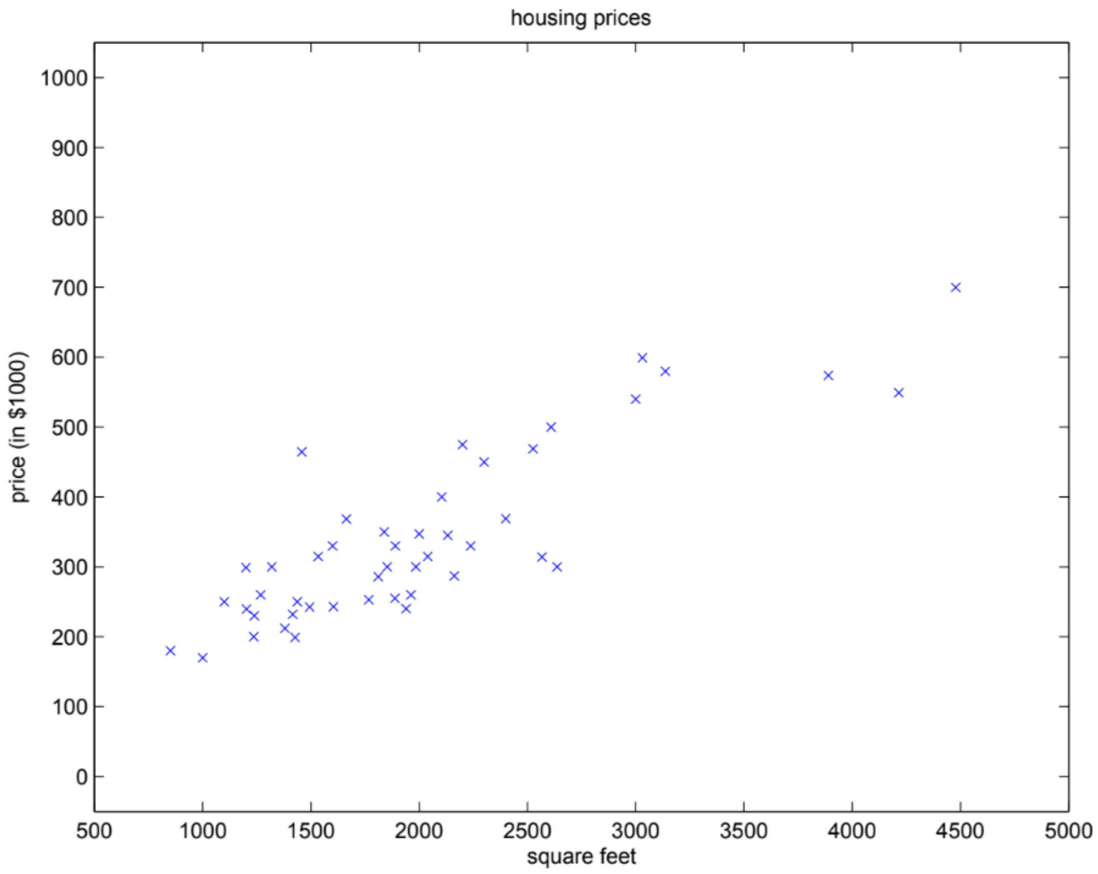
training set

Living area (feet^2)	Price (1000$s)
2104	400
1600	330
2400	369
1416	232
3000	540
⋮	⋮

Living area ($x^{(i)}$) = input features

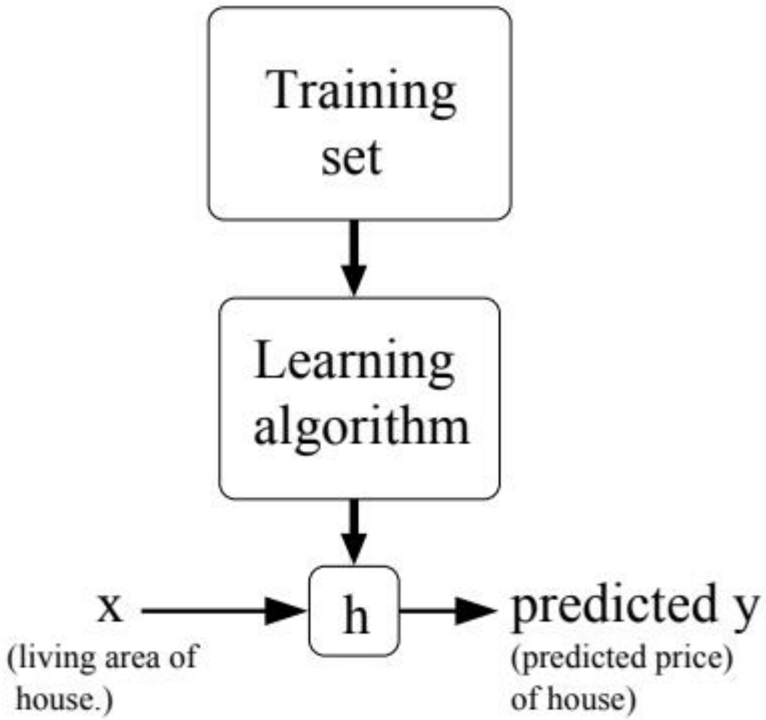
Price ($y^{(i)}$) = output or target

A pair ($x^{(i)}$, $y^{(i)}$): training example



KAIST

Hypothesis for best fit



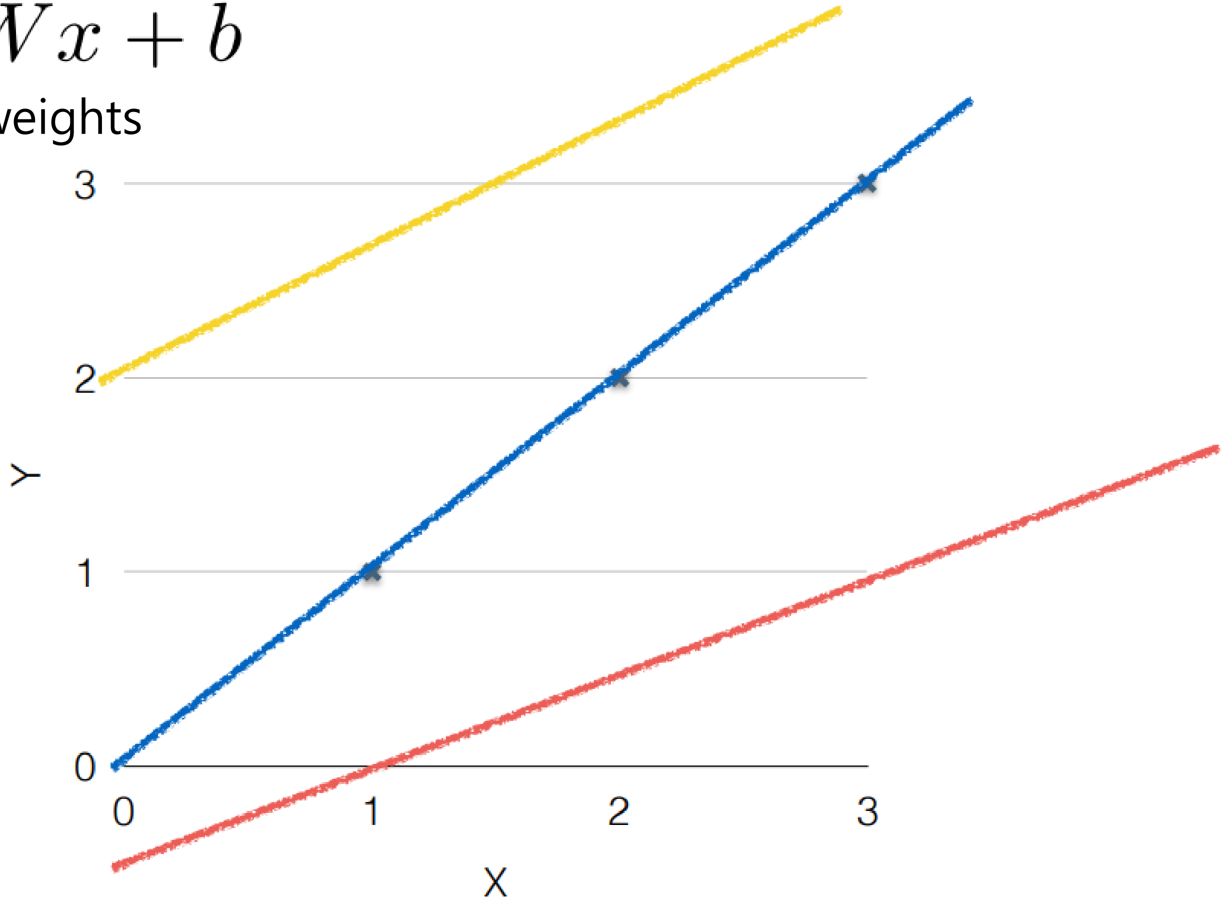
When the target variable that we're trying to predict is continuous, such as in our housing example, we call the learning problem a **regression** problem.

Regression (linear hypothesis)

x	y
1	1
2	2
3	3

$$H(x) = Wx + b$$

W: parameters or weights
b: bias



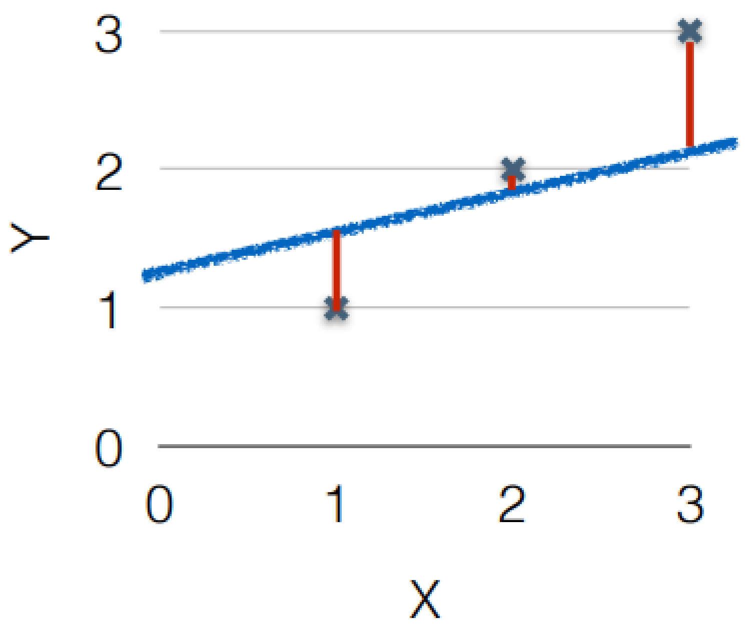
Cost function

How fit the line to our (training) data numerically?

$$\frac{(H(x^{(1)}) - y^{(1)})^2 + (H(x^{(2)}) - y^{(2)})^2 + (H(x^{(3)}) - y^{(3)})^2}{3}$$

$$cost = \frac{1}{m} \sum_{i=1}^{m} (H(x^{(i)}) - y^{(i)})^2$$

$$H(x) = Wx + b$$

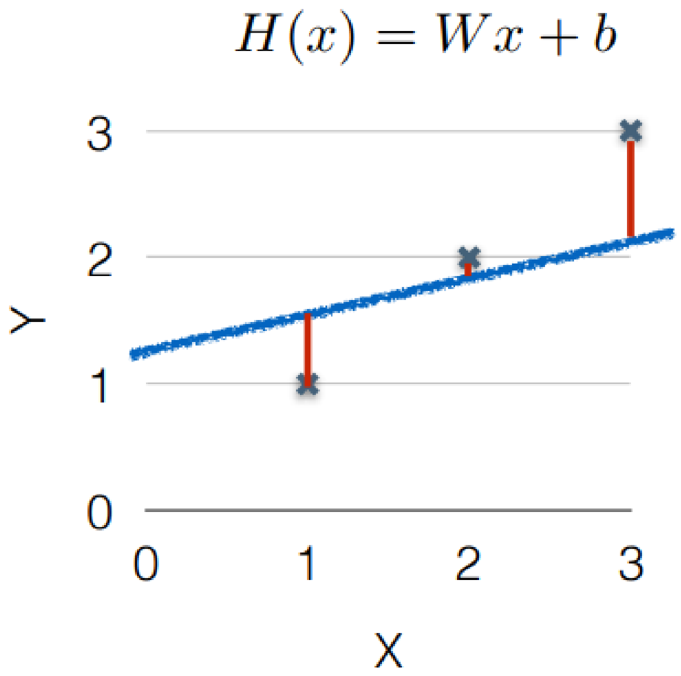


Cost minimization

$$H(x) = Wx + b$$

$$cost(W, b) = \frac{1}{m} \sum_{i=1}^{m} (H(x^{(i)}) - y^{(i)})^2$$

$$\underset{W,b}{\text{minimize}}\, cost(W, b)$$



Simplified hypothesis

$$H(x) = Wx + b$$

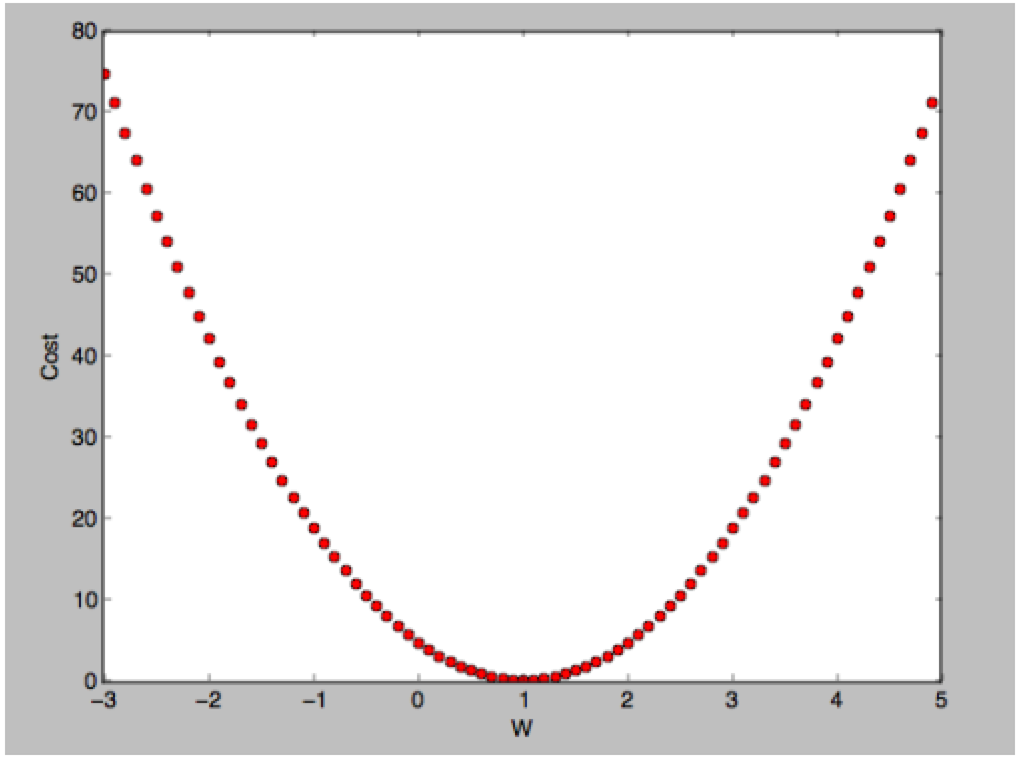
$$cost(W, b) = \frac{1}{m} \sum_{i=1}^{m} (H(x^{(i)}) - y^{(i)})^2$$

$$H(x) = Wx$$

$$cost(W) = \frac{1}{m} \sum_{i=1}^{m} (Wx^{(i)} - y^{(i)})^2$$

What cost looks like?

$$cost(W) = \frac{1}{m}\sum_{i=1}^{m}(Wx^{(i)} - y^{(i)})^2$$



- W=1, cost(W)=0

$$\frac{1}{3}((1*1-1)^2 + (1*2-2)^2 + (1*3-3)^2)$$

- W=0, cost(W)=4.67

$$\frac{1}{3}((0*1-1)^2 + (0*2-2)^2 + (0*3-3)^2)$$

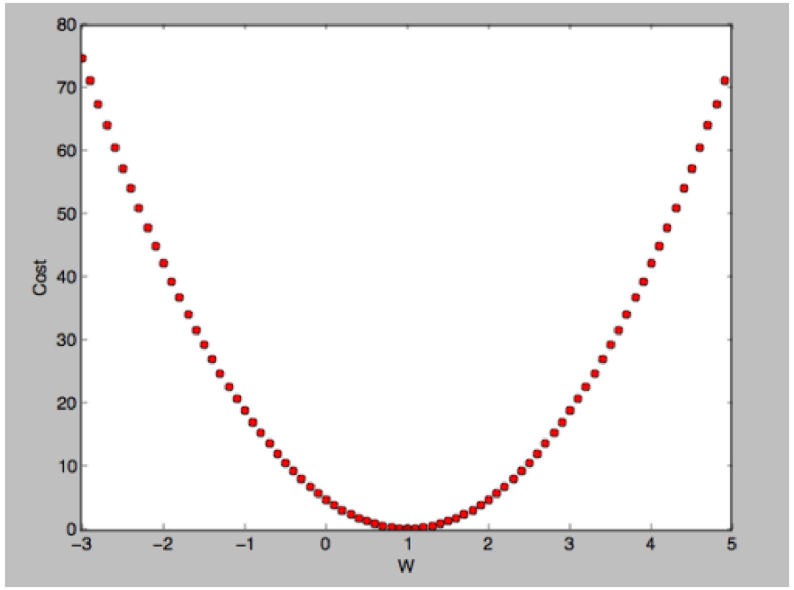
. . .

Gradient descent algorithm

- Minimize cost function using gradient descent algorithm

$$cost(W) = \frac{1}{m}\sum_{i=1}^{m}(Wx^{(i)} - y^{(i)})^2$$

- Start with initial guesses
  - Start at 0,0 (or any other value)
  - Keeping changing W and b a little bit to try and reduce cost(W, b)



- Each time you change the parameters, you select the gradient which reduces cost(W, b) the most possible

$$W := W - \alpha\frac{\partial}{\partial W}cost(W)$$

$\alpha\,(= \Delta W)$: updating rate (or learning rate)
- the larger, the faster
- but may cause ill-convergence

- Repeat until you converge to a local minimum (i.e., the gradient = 0)

Least Mean Squares (LMS)

$$cost(W) = \frac{1}{m}\sum_{i=1}^{m}(Wx^{(i)} - y^{(i)})^2$$

$$cost(W) = \frac{1}{2m}\sum_{i=1}^{m}(Wx^{(i)} - y^{(i)})^2$$

$$W := W - \alpha\frac{\partial}{\partial W}cost(W)$$

$$W := W - \alpha\frac{\partial}{\partial W}\frac{1}{2m}\sum_{i=1}^{m}(Wx^{(i)} - y^{(i)})^2$$

$$W := W - \alpha\frac{1}{2m}\sum_{i=1}^{m}2(Wx^{(i)} - y^{(i)})x^{(i)}$$

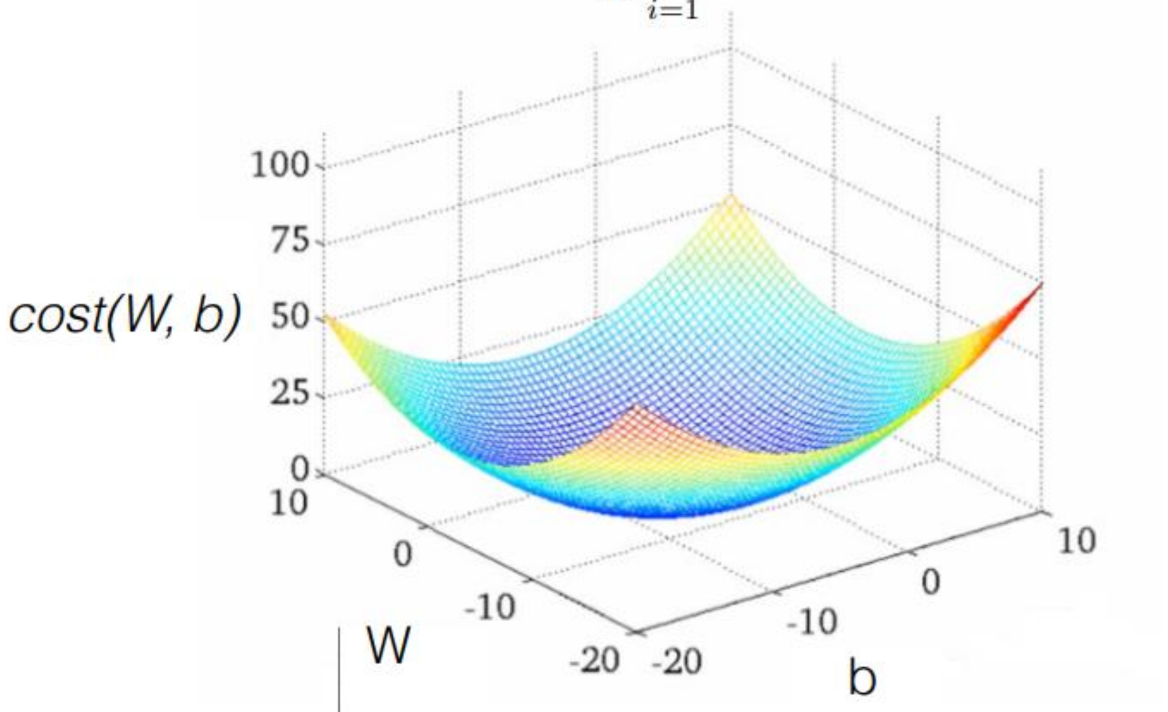
$$W := W - \alpha\frac{1}{m}\sum_{i=1}^{m}(Wx^{(i)} - y^{(i)})x^{(i)}$$

The resulting equation is called the LMS update rule (LMS stands for "least mean squares")

KAIST

Convex function

$$cost(W,b) = \frac{1}{m}\sum_{i=1}^{m}(H(x^{(i)}) - y^{(i)})^2$$



www.holehouse.org/mlclass/

Note that the optimization problem here has only one global, and no other local optima.

Thus gradient descent always converges assuming the learning rate α is not too large to the global minimum.

Gradient descent algorithm

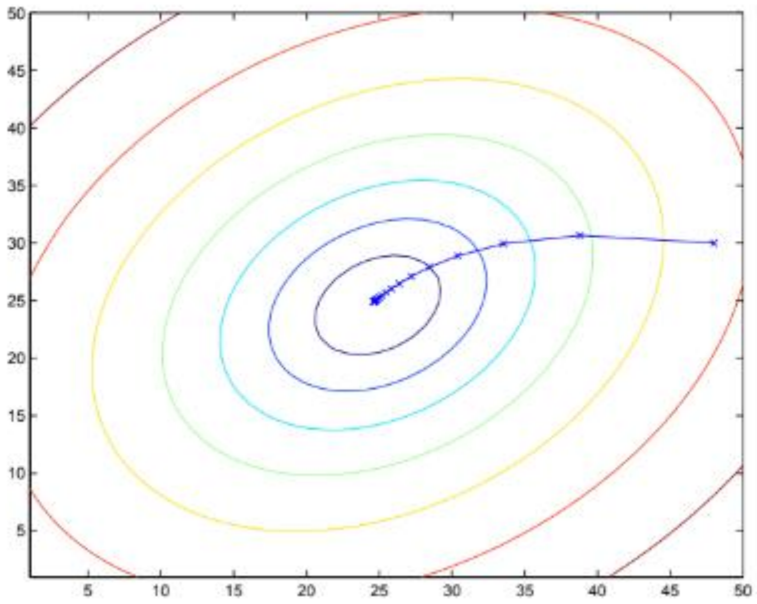
Batch gradient descent

Repeat until convergence {

$$\theta_j := \theta_j + \alpha \sum_{i=1}^{m} \left(y^{(i)} - h_\theta(x^{(i)}) \right) x_j^{(i)} \qquad \text{(for every } j\text{).}$$

}

scan through the entire training set before taking a single step



Stochastic gradient descent
(also incremental gradient descent)

Loop {

for i=1 to m, {

$$\theta_j := \theta_j + \alpha \left(y^{(i)} - h_\theta(x^{(i)}) \right) x_j^{(i)} \qquad \text{(for every } j\text{).}$$

}

}

start making progress right away, and continues to make progress with each example it looks at.

much faster convergence than batch gradient descent.

Multivariate linear regression

LMS with multivariables

multi-variable/feature

x^1 (quiz 1)	x^2 (quiz 2)	x^3 (midterm 1)	Y (final)
73	80	75	152
93	88	93	185
89	91	90	180
96	98	100	196
73	66	70	142

Hypothesis and cost function

$$H(x) = Wx + b$$

$$H(x_1, x_2, x_3) = w_1x_1 + w_2x_2 + w_3x_3 + b$$

$$cost(W, b) = \frac{1}{m}\sum_{I=1}^{m}(H(x_1^{(i)}, x_2^{(i)}, x_3^{(i)}) - y^{(i)})^2$$

KAIST

Hypothesis using matrix

$$H(x_1, x_2, x_3, ..., x_n) = w_1x_1 + w_2x_2 + w_3x_3 + ... + w_nx_n + b$$

$$\boxed{\begin{pmatrix} x_1 & x_2 & x_3 \end{pmatrix} \cdot \begin{pmatrix} w_1 \\ w_2 \\ w_3 \end{pmatrix} = \begin{pmatrix} x_1w_1 + x_2w_2 + x_3w_3 \end{pmatrix}}$$

$$H(X) = XW$$

vector, n
?

Matrix, m x n
?

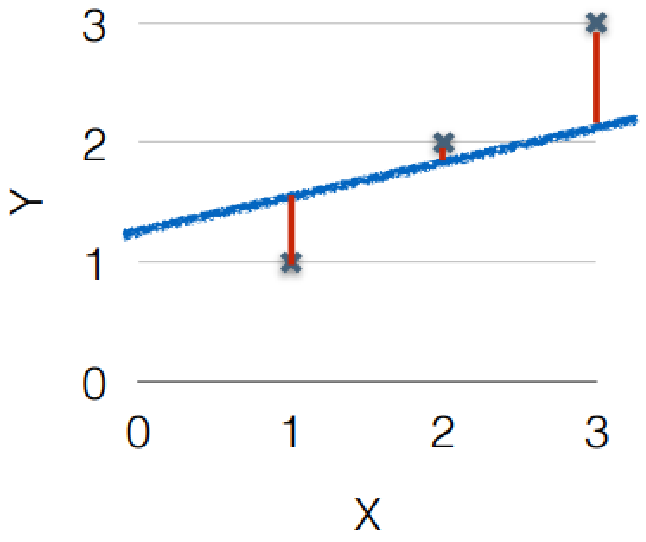
Probabilistic interpretation

Why is the least-squares cost function a reasonable choice?

$$\frac{(H(x^{(1)}) - y^{(1)})^2 + (H(x^{(2)}) - y^{(2)})^2 + (H(x^{(3)}) - y^{(3)})^2}{3}$$

$$cost = \frac{1}{m}\sum_{i=1}^{m}(H(x^{(i)}) - y^{(i)})^2$$

$$H(x) = Wx + b$$



Gaussian noise

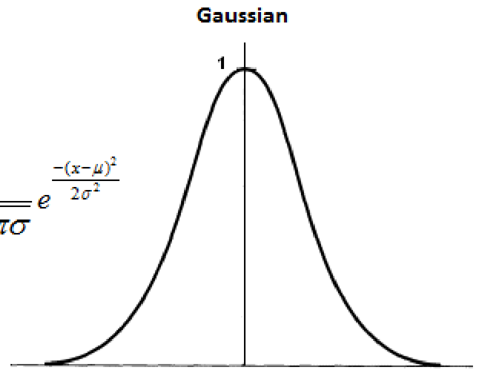
Suppose that the target variables and the inputs are related via the equation.

$$y^{(i)} = \theta^T x^{(i)} + \epsilon^{(i)}$$

where $\varepsilon(i)$ is an error term that captures either unmodeled effects or random noise.

If $\varepsilon(i)$ is independently and identically distributed according to a Gaussian distribution (also called a Normal distribution) with mean zero and some variance σ^2,

$$p(\epsilon^{(i)}) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(\epsilon^{(i)})^2}{2\sigma^2}\right)$$



$$f(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{\frac{-(x-\mu)^2}{2\sigma^2}}$$

Thus,

$$p(y^{(i)}|x^{(i)};\theta) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(y^{(i)} - \theta^T x^{(i)})^2}{2\sigma^2}\right)$$

The notation "$p(y^{(i)}|x^{(i)}; \theta)$" indicates that this is the distribution of $y^{(i)}$ given $x^{(i)}$ and parameterized by θ.

Likelihood

Given a data set, X ({x(i)}), and a ML model θ, what is the distribution of target, Y ({y(i)})?

The probability of the data is given by p(Y|X; θ), which is viewed a function of Y
for a fixed value of θ.

It can be instead called the likelihood function.

$$L(\theta) = L(\theta; X, \vec{y}) = p(\vec{y}|X;\theta)$$

$$\begin{aligned} L(\theta) &= \prod_{i=1}^{m} p(y^{(i)} \mid x^{(i)}; \theta) \\ &= \prod_{i=1}^{m} \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(y^{(i)} - \theta^T x^{(i)})^2}{2\sigma^2}\right) \end{aligned}$$

Lecture01-Likelihood

What is a reasonable way of choosing our best guess of the parameters θ or training the machine learning model?

Maximum likelihood

The principal of **maximum likelihood** says that we should choose θ so as to make the data as high probability as possible.

That is, we should choose θ to maximize L(θ).

$$
\begin{aligned}
L(\theta) &= \prod_{i=1}^{m} p(y^{(i)} \mid x^{(i)}; \theta) \\
&= \prod_{i=1}^{m} \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(y^{(i)} - \theta^T x^{(i)})^2}{2\sigma^2}\right)
\end{aligned}
$$

KAIST

Maximum likelihood

Instead of maximizing L(θ), we can also maximize any strictly increasing function of L(θ).

For example, maximize the log likelihood *l*(θ):

$$
\begin{aligned}
\ell(\theta) &= \log L(\theta) \\
&= \log \prod_{i=1}^{m} \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(y^{(i)} - \theta^T x^{(i)})^2}{2\sigma^2}\right) \\
&= \sum_{i=1}^{m} \log \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(y^{(i)} - \theta^T x^{(i)})^2}{2\sigma^2}\right) \\
&= m \log \frac{1}{\sqrt{2\pi}\sigma} - \frac{1}{\sigma^2} \cdot \frac{1}{2} \sum_{i=1}^{m} (y^{(i)} - \theta^T x^{(i)})^2
\end{aligned}
$$

Note that the final θ has no dependence on σ.

Maximizing *l* (θ) gives the same answer as minimizing

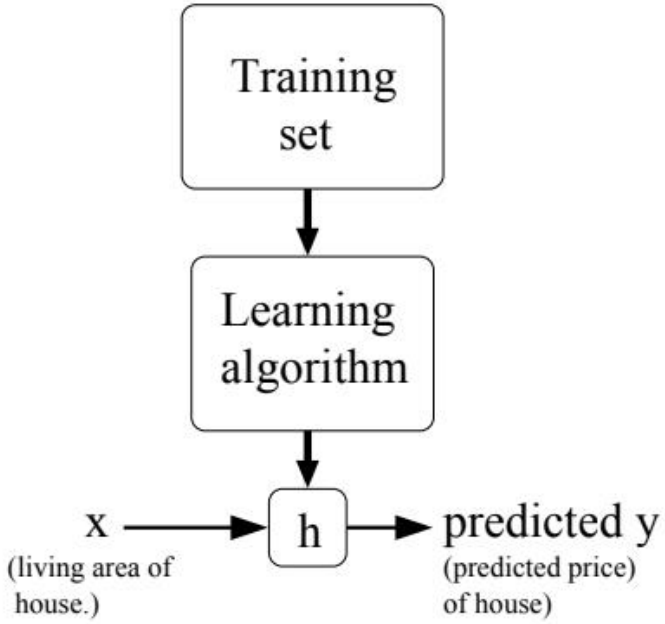
$$\frac{1}{2} \sum_{i=1}^{m} (y^{(i)} - \theta^T x^{(i)})^2$$

which is the original LMS cost function.

Summary

Under the previous probabilistic assumptions on the data, least-squares regression corresponds to finding the maximum likelihood estimate (MLE) of θ.

This is thus one set of assumptions under which least-squares regression can be justified as a very natural method that's just doing MLE.



Classification

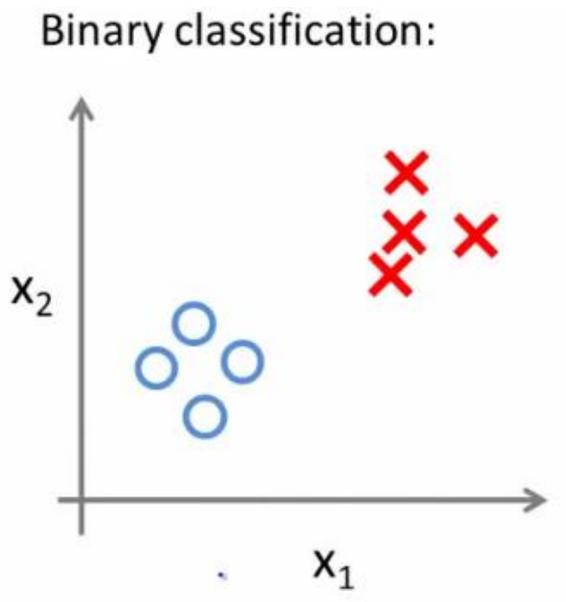
Classification

Classification problem: the values y we now want to predict take on only a small number of discrete values.

Binary classification problem: y can take on only two values, 0 and 1.

0 or - : the negative class,

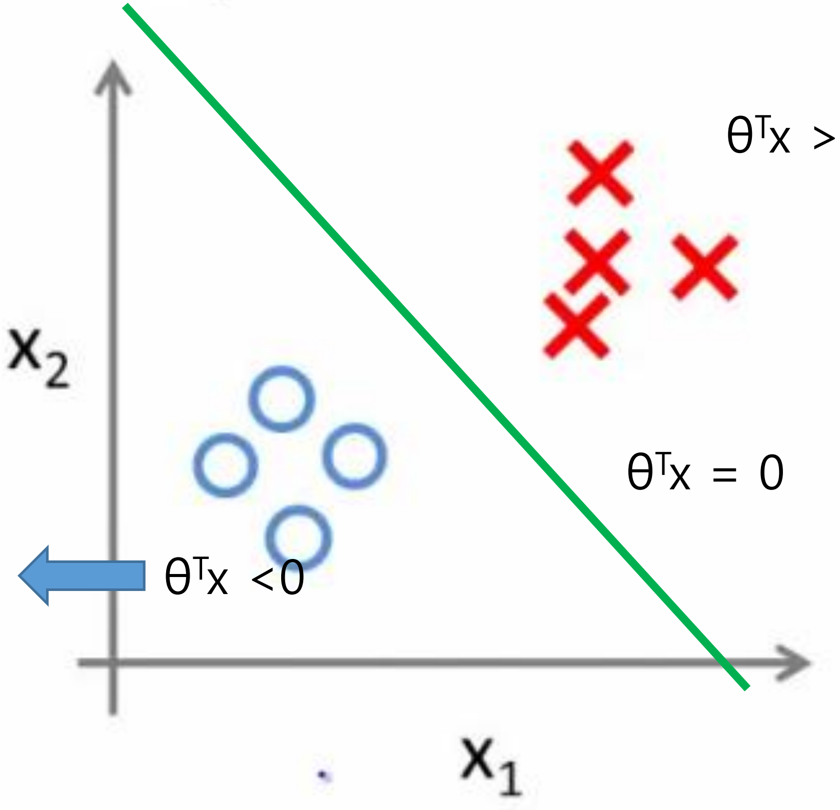
1 or + : the positive class,



KAIST

Decision boundary

Binary classification:



$\theta^T x > 0$ ⇨ $y = h(\theta^T x) = 1$

$\theta^T x = 0$

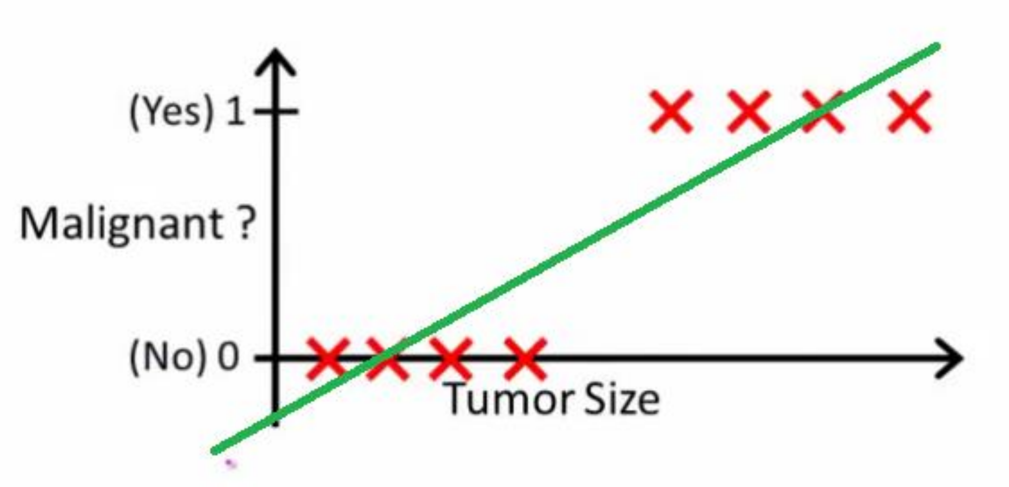
$y = h(\theta^T x) = 1$ ⇦ $\theta^T x < 0$

For 2D, the dimension of θ will be one.

Thus, the decision boundary will be a linear line.

Problem of linear regression

Let's try to predict y given z (= $\theta^T x$) using the linear regression algorithm



It performs very poorly!

Simply it doesn't make sense for $h_\theta(x)$ to take values larger than 1 or smaller than 0 when we know that $y \in \{0, 1\}$.

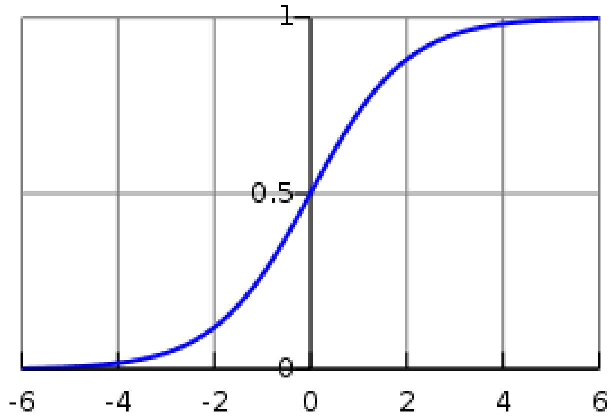
Logistic function

change $h_\theta(x)$ as

$$h(x) = \sum_{i=0}^{n} \theta_i x_i = \theta^T x \quad \Rightarrow \quad h_\theta(x) = g(\theta^T x) = \frac{1}{1 + e^{-\theta^T x}}$$

logistic function or the sigmoid function.

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

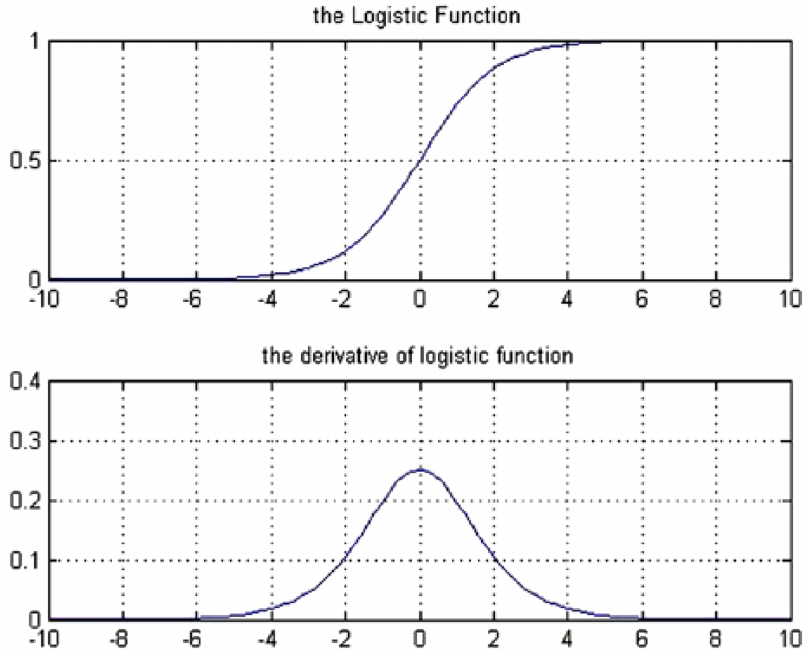


- ✓ It becomes 0.5 as a data point is on the decision boundary (z = 0).
- ✓ It goes to 1 as z → ∞ and to 0 as z → −∞.
- ✓ It is always bounded between 0 and 1.

Logistic function

A useful property of the derivative of the sigmoid function,

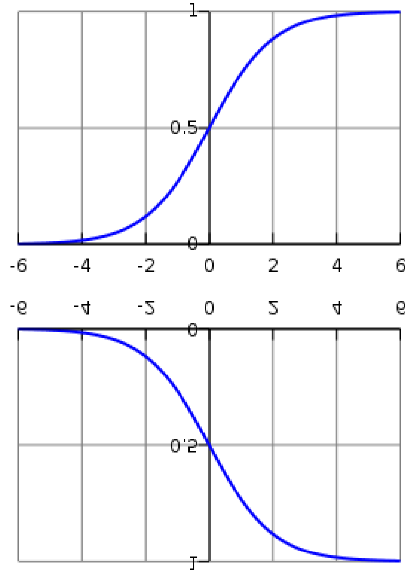
$$
\begin{aligned}
g'(z) &= \frac{d}{dz}\frac{1}{1+e^{-z}} \\
&= \frac{1}{(1+e^{-z})^2}\left(e^{-z}\right) \\
&= \frac{1}{(1+e^{-z})}\cdot\left(1-\frac{1}{(1+e^{-z})}\right) \\
&= g(z)(1-g(z)).
\end{aligned}
$$



Maximum likelihood

Assuming that

$$
\begin{aligned}
P(y = 1 \mid x; \theta) &= h_\theta(x) \\
P(y = 0 \mid x; \theta) &= 1 - h_\theta(x)
\end{aligned}
$$



More compactly

$$
p(y \mid x; \theta) = (h_\theta(x))^y \left(1 - h_\theta(x)\right)^{1-y}
$$

Maximum likelihood

For m training examples, the likelihood of the parameters

$$
\begin{aligned}
L(\theta) &= p(\vec{y} \mid X; \theta) \\
&= \prod_{i=1}^{m} p(y^{(i)} \mid x^{(i)}; \theta) \\
&= \prod_{i=1}^{m} \left(h_\theta(x^{(i)})\right)^{y^{(i)}} \left(1 - h_\theta(x^{(i)})\right)^{1-y^{(i)}}
\end{aligned}
$$

The corresponding log likelihood

$$
\begin{aligned}
\ell(\theta) &= \log L(\theta) \\
&= \sum_{i=1}^{m} y^{(i)} \log h(x^{(i)}) + (1 - y^{(i)}) \log(1 - h(x^{(i)}))
\end{aligned}
$$

Maximizing via the gradient ascent.

$$\theta := \theta + \alpha \nabla_\theta \ell(\theta)$$

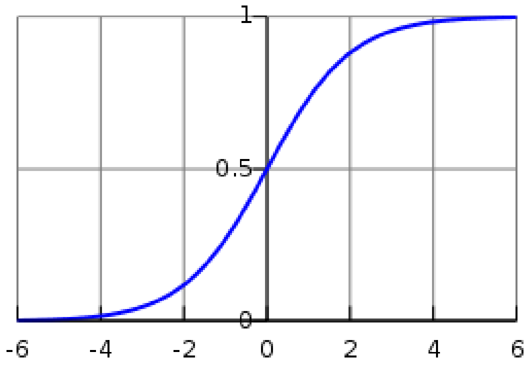
for the maximization

KAIST

Cost function

The log likelihood ➔ one has to maximize it

$$\begin{aligned}\ell(\theta) &= \log L(\theta) \\ &= \sum_{i=1}^{m} y^{(i)} \log h(x^{(i)}) + (1-y^{(i)}) \log(1-h(x^{(i)}))\end{aligned}$$



Cost function = -log likelihood ➔ one has to minimize it

$$Cost = -\sum_{i=1}^{m}\left[y^{(i)} \ln h\left(x^{(i)}\right) + (1-y^{(i)}) \ln(1-h\left(x^{(i)}\right))\right]$$

$y^{(i)}$: true value in the training data ➔ true probablity of being in class 1
$h\left(x^{(i)}\right)$: predicted value by the classifier ➔ predicted probability of being in class 1

$1-y^{(i)}$: true probablity of being in class 2
$1-h\left(x^{(i)}\right)$: predicted probability of being in class 2

Cost function

$$Cost = -\sum_{i=1}^{m}\left[y^{(i)} \ln h\left(x^{(i)}\right) + (1 - y^{(i)}) \ln(1 - h\left(x^{(i)}\right))\right]$$

Two important properties:

1. Non-negative because 0<= $prob$ <=1

2. For all training data (m), if $h\left(x^{(i)}\right)$ is close to $y^{(i)}$, cost goes to zero.

if $y^{(i)}$ = 0, $h\left(x^{(i)}\right)$→0: 1st term = 0 and ln1=0 in the 2nd term; therefore Cost →0

if $y^{(i)}$ = 1, $h\left(x^{(i)}\right)$→1: ln1=0 in the 1st term and 2nd term =0; therefore Cost →0

Indeed, it can play a role as a cost function.

Maximum likelihood

Let us take derivatives to derive the stochastic gradient ascent rule:

$$\begin{aligned}
\frac{\partial}{\partial \theta_j}\ell(\theta) &= \left(y\frac{1}{g(\theta^T x)} - (1-y)\frac{1}{1-g(\theta^T x)}\right)\frac{\partial}{\partial \theta_j} g(\theta^T x) \\
&= \left(y\frac{1}{g(\theta^T x)} - (1-y)\frac{1}{1-g(\theta^T x)}\right) g(\theta^T x)(1-g(\theta^T x)\frac{\partial}{\partial \theta_j}\theta^T x \\
&= \left(y(1-g(\theta^T x)) - (1-y)g(\theta^T x)\right) x_j \\
&= \left(y - h_\theta(x)\right) x_j
\end{aligned}$$

$$g'(z) = g(z)(1-g(z))$$

The stochastic gradient ascent rule

$$\theta_j := \theta_j + \alpha\left(y^{(i)} - h_\theta(x^{(i)})\right) x_j^{(i)}$$

Maximum likelihood

The stochastic gradient ascent rule

$$\theta_j := \theta_j + \alpha \underbrace{\left(y^{(i)} - h_\theta(x^{(i)})\right)}_{\text{Difference between the true and predicted values}} x_j^{(i)}$$

The derivative becomes zero as the difference becomes zero.

In other words, the classifier perfectly predicts the class of input data!

Cross entropy

$$Cost = -\sum_{i=1}^{m}\left[y^{(i)}\ln h\left(x^{(i)}\right) + (1-y^{(i)})\ln(1-h\left(x^{(i)}\right))\right]$$

$$\Downarrow$$

$$Cost = -\sum_{i=1}^{m} p_{true}\left(x^{(i)}\right)\ln p_{pred}\left(x^{(i)}\right)$$

It is also called the cross entropy

VS

Gibbs entropy formula

$$S = -k_{\mathrm{B}} \sum_{i} p_i \ln p_i$$

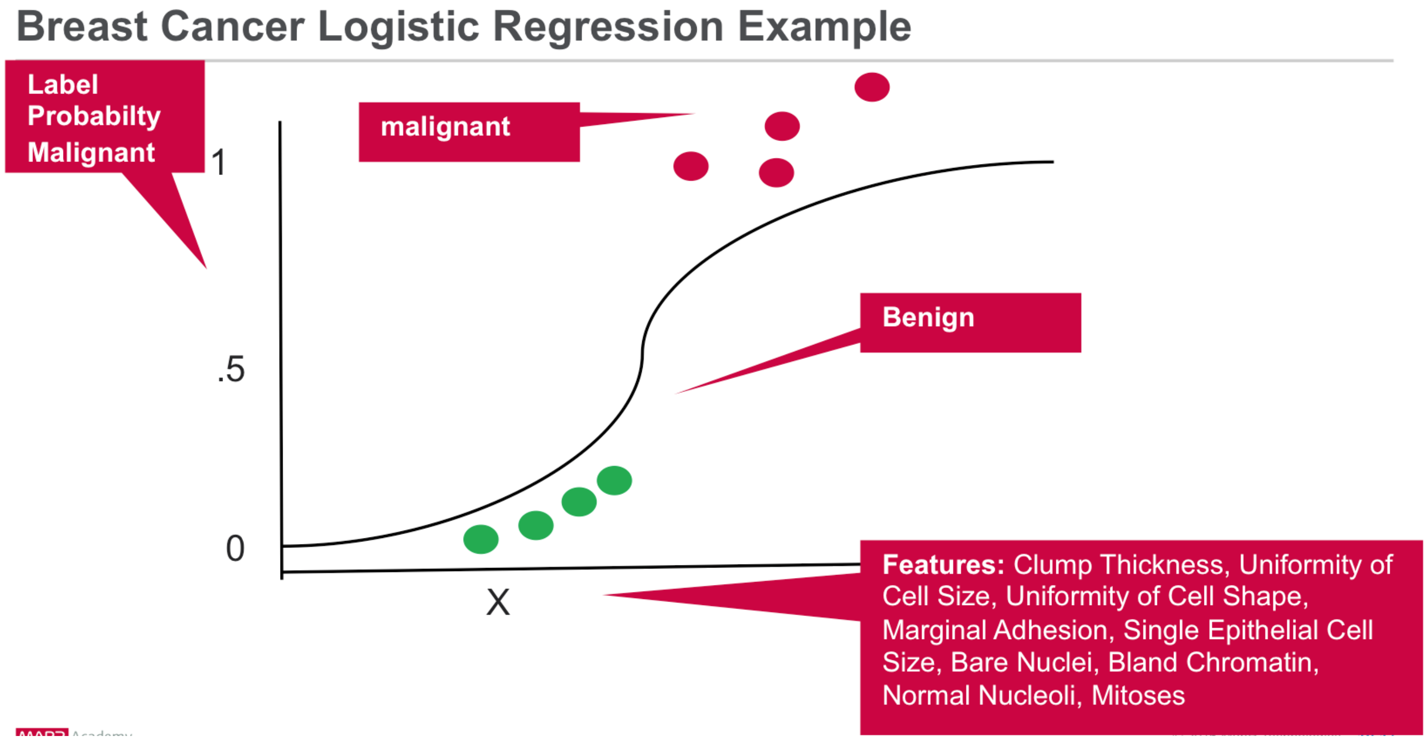
Entropy in information theory

$$S = -\sum_{i=1} p_i \ln p_i$$

KAIST

Example

Logistic regression measures the relationship between the Y "Label" and the X "Features" by estimating probabilities using a logistic function.

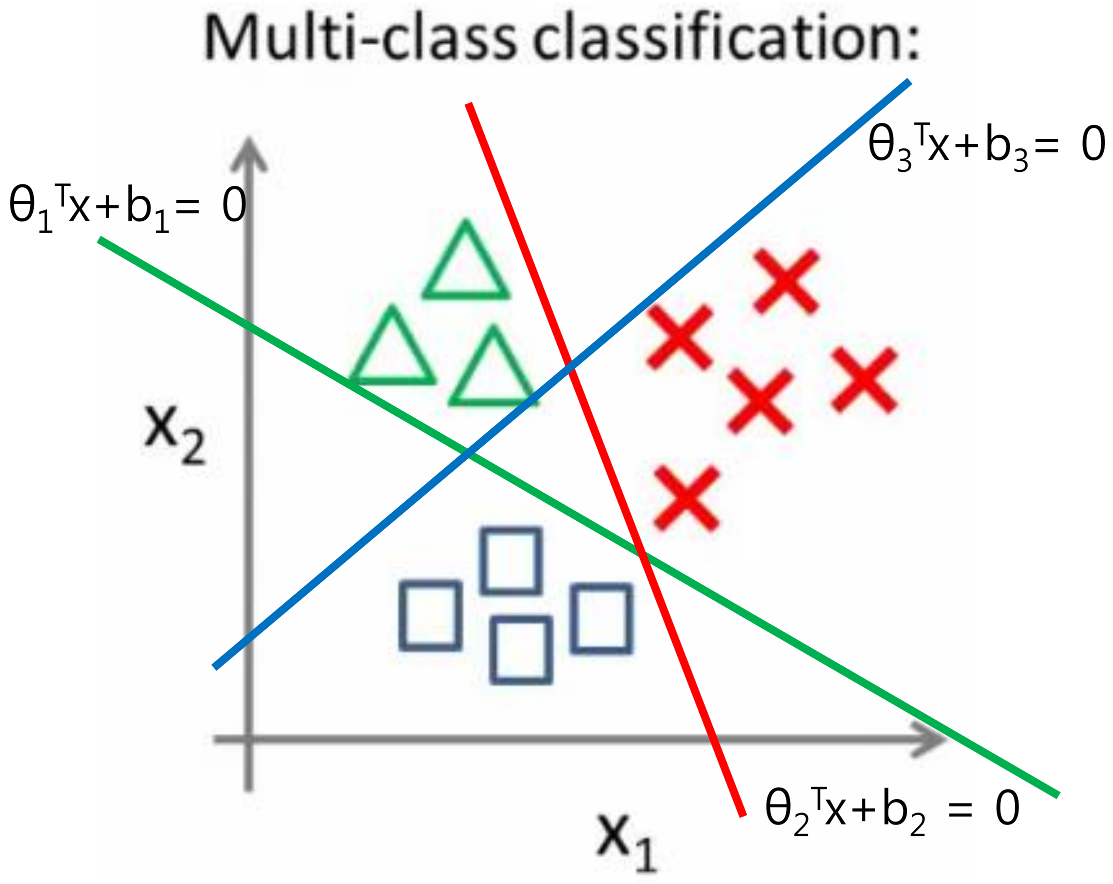


Wisconsin Diagnostic Breast Cancer (WDBC) Data Set which categorizes breast tumor cases as either benign or malignant based on 9 features to predict the diagnosis.

https://mapr.com/blog/predicting-breast-cancer-using-apache-spark-machine-learning-logistic-regression/

Softmax classification

Multinomial classification



We need 3 decision boundaries!

For 2D, 3 lines:

$\theta_1^T x + b_1 = 0$, $\theta_2^T x + b_2 = 0$, $\theta_3^T x + b_3 = 0$

Softmax function

The softmax function

$$\phi_i = \frac{e^{\eta_i}}{\sum_{j=1}^{k} e^{\eta_j}} \Rightarrow \sum_{i=1}^{k} \phi_i = 1$$

➔Probability of having the k-th label.

If k = 2 like the binary classification (y = 0 or 1),

$$\emptyset_1 = \frac{e^{z1}}{e^{z1} + e^{z2}} = \frac{1}{1 + e^{z2-z1}}$$ Prob. for k = 1

$$\emptyset_2 = 1 - \emptyset_1$$ Prob. for k = 2

logistic function!!

➔ Generalization of the logistic function for multinomial problems

Softmax classification

The softmax regression

$$
[\theta^T][x] = \begin{array}{c} \text{Dog}\ \ \text{Cat}\ \ \text{Rabbit} \\ \begin{bmatrix} x_{11} x_{12} x_{13} \\ x_{21} x_{22} x_{23} \\ x_{31} x_{32} x_{33} \end{bmatrix} \end{array} \begin{bmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \end{bmatrix} = \begin{bmatrix} z_1 \\ z_2 \\ z_3 \end{bmatrix} = \mathbf{z} \quad \Rightarrow \quad h(z) = \begin{bmatrix} p_1 \\ p_2 \\ p_3 \end{bmatrix}
$$

Probability of being in each class

The hypothesis function

$$
h_\theta(x) = \begin{bmatrix} \frac{\exp(\theta_1^T x)}{\sum_{j=1}^{k} \exp(\theta_j^T x)} \\ \frac{\exp(\theta_2^T x)}{\sum_{j=1}^{k} \exp(\theta_j^T x)} \\ \vdots \\ \frac{\exp(\theta_{k-1}^T x)}{\sum_{j=1}^{k} \exp(\theta_j^T x)} \end{bmatrix}
$$

One has to determine the parameters $\{\theta\}$ with a given training set ➔ cost function

One-hot encoding

$$h(x) = \begin{bmatrix} p_1 \\ p_2 \\ p_3 \end{bmatrix} = \begin{bmatrix} 0.8 \\ 0.2 \\ 0.1 \end{bmatrix} \rightarrow \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$$

The input data is labeled the class 1

$$\text{Dog} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$$

$$\text{Cat} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$$

$$\text{Rabbit} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

Cost function

As the generalization of the logistic regression, we use the cross entropy

$$Cost = -\sum_{i=1}^{m} y^{(i)} \ln h\left(x^{(i)}\right)$$

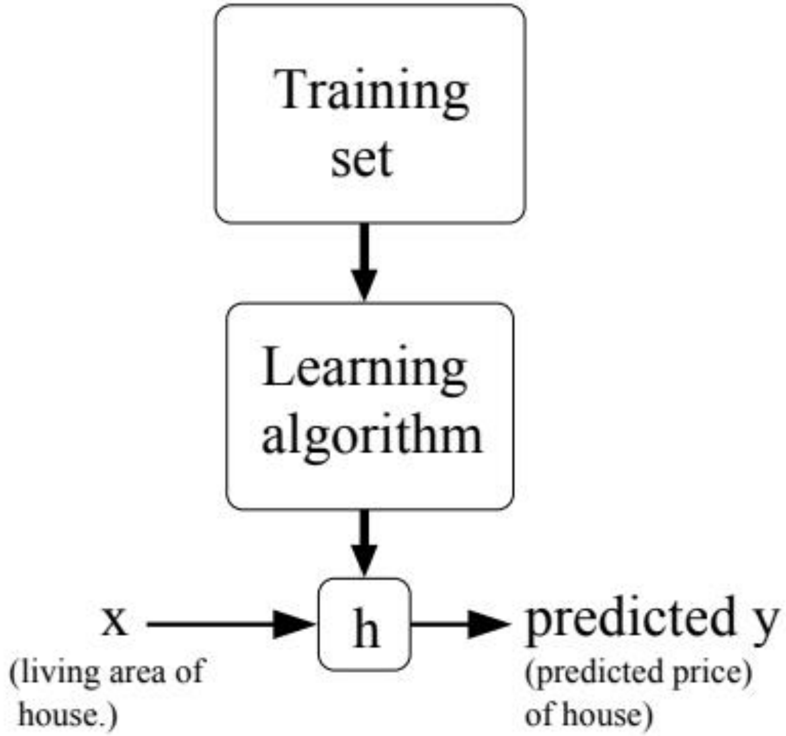
element-wise product

where

$$y^{(i)} = \begin{bmatrix} y_1^{(i)} \\ y_2^{(i)} \\ y_3^{(i)} \end{bmatrix} \qquad h\left(x^{(i)}\right) = \begin{bmatrix} \dfrac{\exp \theta_1^T\, x_1^{(i)}}{\sum_{k=1}^{3} \exp \theta_k^T\, x_k^{(i)}} \\ \dfrac{\exp \theta_2^T\, x_2^{(i)}}{\sum_{k=1}^{3} \exp \theta_k^T\, x_k^{(i)}} \\ \dfrac{\exp \theta_3^T\, x_1^{(i)}}{\sum_{k=1}^{3} \exp \theta_k^T\, x_3^{(i)}} \end{bmatrix}$$

$$y^{(i)} \ln h\left(x^{(i)}\right) = \begin{bmatrix} y_1^{(i)} \ln \dfrac{\exp \theta_1^T x_1^{(i)}}{\sum_{k=1}^{3} \exp \theta_k^T x_k^{(i)}} \\ y_2^{(i)} \ln \dfrac{\exp \theta_2^T x_2^{(i)}}{\sum_{k=1}^{3} \exp \theta_k^T x_k^{(i)}} \\ y_3^{(i)} \ln \dfrac{\exp \theta_3^T x_1^{(i)}}{\sum_{k=1}^{3} \exp \theta_k^T x_3^{(i)}} \end{bmatrix}$$

Summary



The principal of **maximum likelihood** says that we should choose θ so as to make the data as high probability as possible.

New terms

- Supervised learning
- Unsupervised learning
- Reinforcement learning
- Hypothesis
- Least Mean Square (LMS)
- Maximum Likelihood Estimate (MLE)
- Regression
- Gradient descent
- Classification
- Decision boundary
- Logistic or sigmoid function
- Cost function
- Softmax classification
- Cross entropy
- One-hot encoding

Likelihood

Bayes' Rule

$$P(\theta|x) = \frac{P(x|\theta)P(\theta)}{P(x)} = \frac{P(x|\theta)P(\theta)}{\sum_\theta P(x|\theta)P(\theta)}$$

$$\text{Posterior probability} = \frac{(\text{likelihood}) \times (\text{prior probability})}{(\text{evidence})}$$

Prior probability (사전확률) $P(\theta)$: probability of a parameter set θ.

Posterior probability (사후확률) $P(\theta|x)$: $P(\theta)$ given an observation x.

Evidence (증거) $P(x)$: probability of the observation.

Likelihood (가능도) $L(\theta|x) = P(x|\theta)$: $P(x)$ given the parameters θ

θ = disease

x = test result (T or F)

$P(x)$: probability of T and F

$P(\theta)$: probability of having a disease

$P(\theta|x)$: probability of having the disease given the test result

$P(x)$: probability of T and F.

back

KAIST