CSE 4553
Machine Learning
Lecture 2: Regression analysis

Winter 2022

Hasan Mahmud | hasan@iut-dhaka.edu

Contents

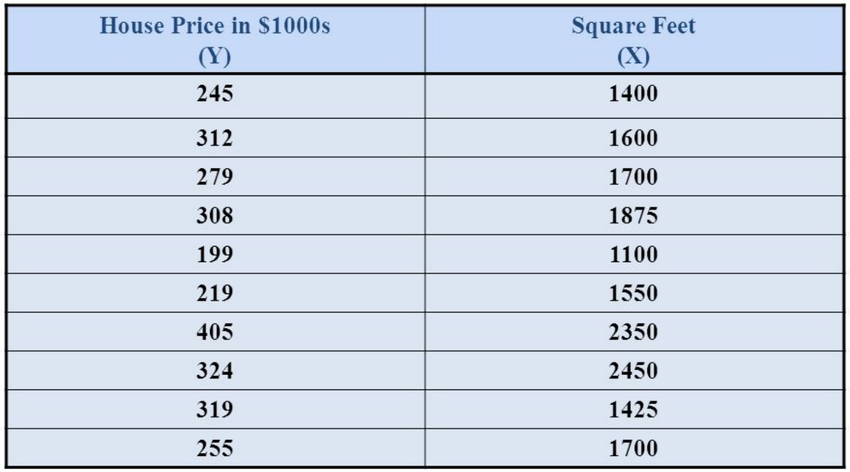
- Introduction
- What is regression?
- Linear regression
- Logistic regression

Introduction

- What we know today as regression was invented by the cousin of Charles Darwin, Francis Galton.
- Linear regression is used for predictive analysis and modeling.

What is regression?

- Regression is a parametric technique used to predict continuous (dependent) variable given a set of independent variables.
- Hypothesis, $y(x) = w_0 + w_1 x$
- Parameters: w_0, w_1
- Mathematically, regression uses a linear function to approximate (predict) the dependent variable.
- Predict 'real valued output‘
- Example



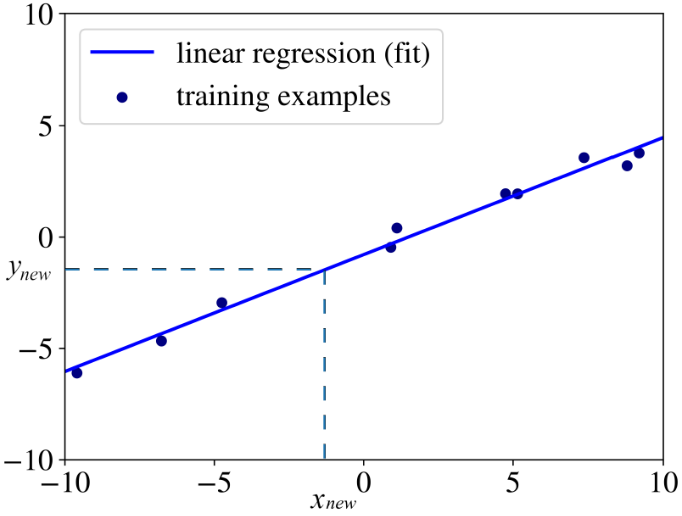
House Price in $1000s (Y)	Square Feet (X)
245	1400
312	1600
279	1700
308	1875
199	1100
219	1550
405	2350
324	2450
319	1425
255	1700

Linear regression

- For a collection of labelled/training examples, $\{x^n, y^n\}_{i=1}^{N}$ where $\{x^n, y^n\} \in \Re\}$, We want to build a model $g_{w,b}(x)$ as a linear combination of the features of the example x, i.e. $\boldsymbol{g_{w,b}(x) = wx + b}$
  - N is the number of instances, i indexes different examples in the set.
  - x_j^n is the input variables consists of d-dimensional feature vector, $x_1, x_2, \dots, x_d, where\ j\ =\ 1\ to\ d$
  - y^n is a real valued output or target variable. $y^n \in \Re, \Re$ is the set of all real numbers.
  - A pair (x^n, y^n) is called training example
  - The dataset that we will be using to learn a list of n training examples $\{x^n, y^n\}_{i=1}^{N}$ is called a training set.
  - $g_{w,b}$ means that the model g is parameterized by two values: w and b,
  - w is the d-dimensional weight vector $(w_1, w_2, \dots, w_d)$ and b is a real number.
  - In regression, there is noise added to the output, $g_{w,b}(x) = wx + b + \in$

Linear regression

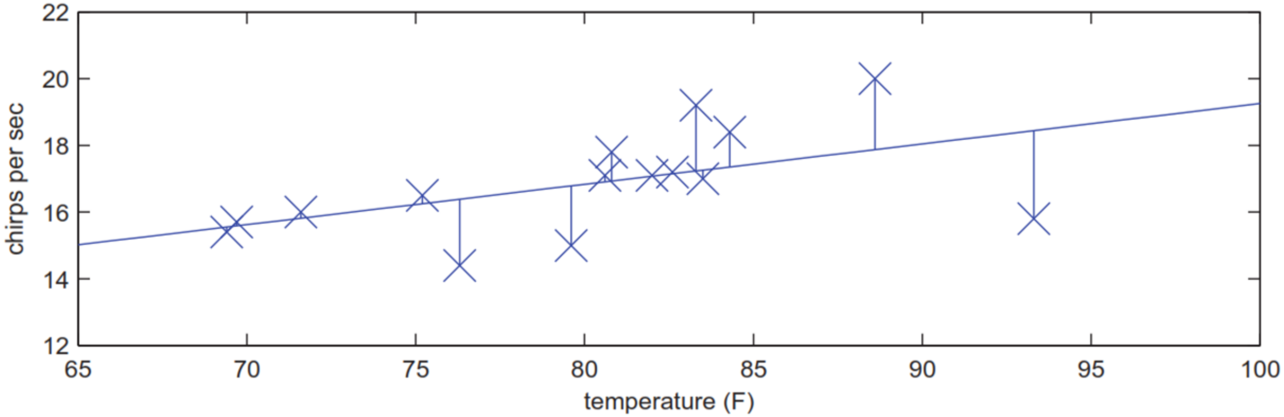
- To determine the optimal parameters, w, b for more accurate predictions/approximations of the regression model, $g(x)$, we measure the empirical error on the training set X as the difference between actual output, $y^n$and predicted output, $g(x^n)$. i.e.
  - $\boldsymbol{E(g|X) = \frac{1}{N}\sum_{i=1}^{N}[y^n - g(x^n)]^2 \Rightarrow \frac{1}{N}\sum_{i=1}^{N}[y^n - (wx^n + b)]^2}$
  - This is known as sum squared error or ordinary least squares and minimizes the average vertical projection of the points y to fitted line.



Linear regression

- Through numerical analysis
- Given training examples, $\{(x^n, y^n), n = 1, \ldots\ldots, N\}$ for real valued input x^n and real valued output y^n, a linear regression fit is: $\boldsymbol{y}(\boldsymbol{x}) = \boldsymbol{a} + \boldsymbol{b}\boldsymbol{x}$
- To determine the best parameters a, b, we measure the discrepancy between the observed outputs and the linear regression fit function such as the SSE (sum squared error). This is called ordinary least squares and minimizes the average vertical projects of the points y to the fitted line.

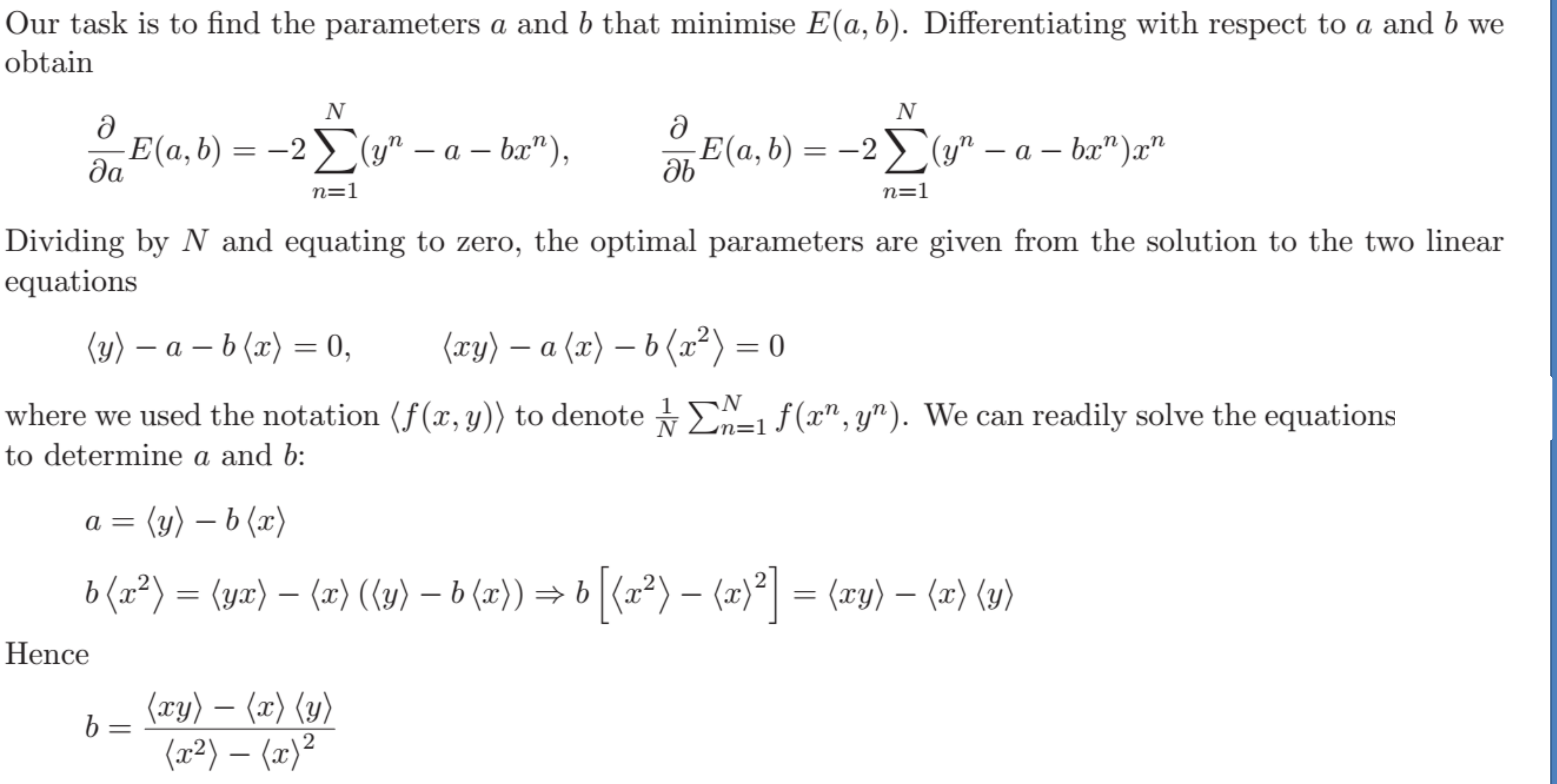
$$E(a, b) = \sum_{n=1}^{N} [y^n - y(x^n)]^2 = \sum_{n=1}^{N} (y^n - a - bx^n)^2$$



- Linear parameter model for regression

Linear regression

- Through numerical analysis



Our task is to find the parameters a and b that minimise $E(a,b)$. Differentiating with respect to a and b we obtain

$$\frac{\partial}{\partial a}E(a,b) = -2\sum_{n=1}^{N}(y^n - a - bx^n), \qquad \frac{\partial}{\partial b}E(a,b) = -2\sum_{n=1}^{N}(y^n - a - bx^n)x^n$$

Dividing by N and equating to zero, the optimal parameters are given from the solution to the two linear equations

$$\langle y\rangle - a - b\langle x\rangle = 0, \qquad \langle xy\rangle - a\langle x\rangle - b\left\langle x^2\right\rangle = 0$$

where we used the notation $\langle f(x,y)\rangle$ to denote $\frac{1}{N}\sum_{n=1}^{N} f(x^n, y^n)$. We can readily solve the equations to determine a and b:

$$a = \langle y\rangle - b\langle x\rangle$$

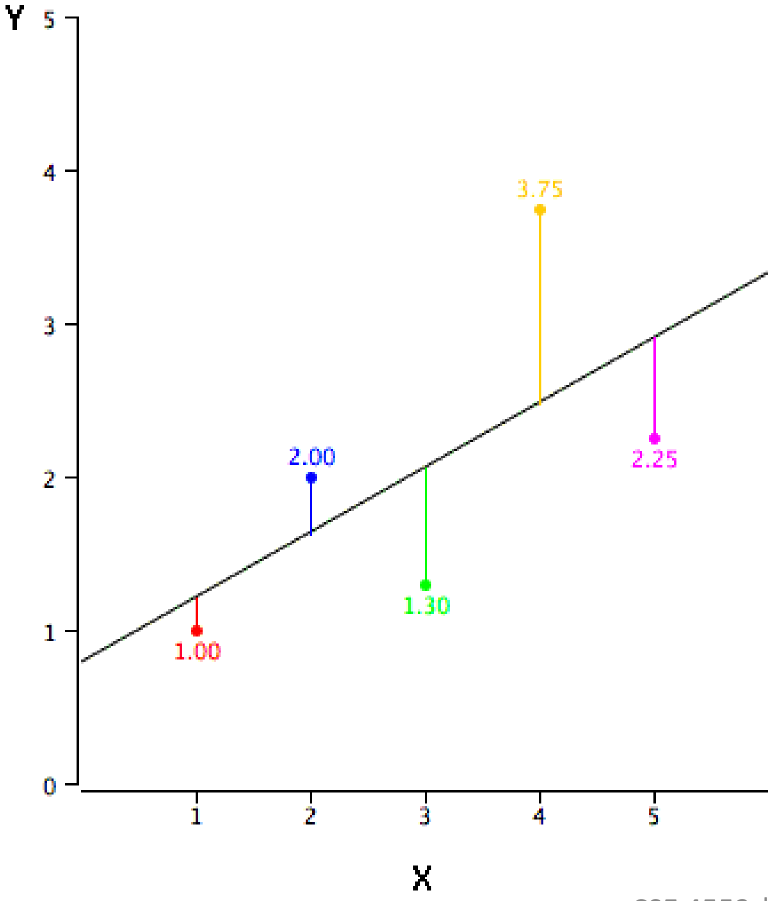
$$b\left\langle x^2\right\rangle = \langle yx\rangle - \langle x\rangle\left(\langle y\rangle - b\langle x\rangle\right) \Rightarrow b\left[\left\langle x^2\right\rangle - \langle x\rangle^2\right] = \langle xy\rangle - \langle x\rangle\langle y\rangle$$

Hence

$$b = \frac{\langle xy\rangle - \langle x\rangle\langle y\rangle}{\langle x^2\rangle - \langle x\rangle^2}$$

Linear regression

- Through linear algebra



Errors, $\epsilon_{\mathbf{i}} = \ y_{\mathbf{i}} + (\beta_0 + \beta_0 x_{\mathbf{i}})$

To minimize $\sum \epsilon^2$ is least squares

$$\beta_0 + X\vec{\beta} = \vec{y}$$

$$\vec{\beta} = (X^T X)^{-1} X^T \vec{y}$$

$$\begin{bmatrix} 1x_1 \\ 1x_2 \\ . \\ . \\ . \\ 1x_n \end{bmatrix} \text{X} \begin{bmatrix} \beta_0 \\ \beta_1 \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ . \\ . \\ . \\ y_n \end{bmatrix}$$

Linear regression

- Through linear algebra
  - So the least square matrix problem is:

$$\begin{bmatrix} \beta_0 + \beta_1 x_1 \\ \beta_0 + \beta_1 x_2 \\ . \\ . \\ . \\ \beta_0 + \beta_1 x_n \end{bmatrix} \textit{ is close to } \begin{bmatrix} y_1 \\ y_2 \\ . \\ . \\ . \\ y_n \end{bmatrix}$$

Linear regression

- Through linear algebra
  - Let us consider our initial equation: $\mathbf{x}\vec{\beta} = \vec{y}$
  - Multiplying both sides by X^T matrix: $\mathbf{x}^{\mathbf{T}}\mathbf{x}\vec{\beta} = \mathbf{x}^{\mathbf{T}}\vec{y}$
  - Where

$$X^T X = \begin{bmatrix} 1 & 1 & 1 & \dots & 1 \\ x_1 & x_2 & x_3 & \dots & x_n \end{bmatrix} x \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ & . \\ & . \\ & . \\ 1 & x_n \end{bmatrix} = \begin{bmatrix} N & \sum X_i \\ \sum X_i & \sum X_i^2 \end{bmatrix}$$

$$\vec{\beta} = (X^T X)^{-1} X^T \vec{y}$$

$$X^T \vec{y} = \begin{bmatrix} 1 & 1 & 1 & \dots & 1 \\ x_1 & x_2 & x_3 & \dots & x_n \end{bmatrix} x \begin{bmatrix} y_1 \\ y_2 \\ . \\ . \\ . \\ y_n \end{bmatrix} = \begin{bmatrix} \sum y \\ \sum X_i y_i \end{bmatrix}$$

$$\begin{bmatrix} \beta_0 \\ \beta_1 \end{bmatrix} = \begin{bmatrix} N & \sum X_i \\ \sum X_i & \sum x_i^2 \end{bmatrix}^{-1} \begin{bmatrix} \sum y \\ \sum X_i y_i \end{bmatrix}$$

Linear regression

- Through gradient descent
  - – Stochastic gradient descent
  - – Batch gradient descent

Univariate feature

- Hypothesis: $h_\theta(x) = \theta_0 + \theta_1 x^i$
- Parameters: θ_0 , θ_1
- Cost Function:

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta j} J(\theta)$$

Now,

$$\frac{\partial}{\partial \theta} J_\theta = \frac{\partial}{\partial \theta} \frac{1}{2m} \Sigma_{i=1}^{m} (h_\theta(x_i) - y_i)^2$$

$$\frac{\partial}{\partial \theta} J_\theta = \frac{1}{m} \Sigma_{i=1}^{m} (h_\theta(x_i) - y_i) . \frac{\partial}{\partial \theta j} (\theta x_i - y_i)$$

$$\frac{\partial}{\partial \theta} J_\theta = \frac{1}{m} \Sigma_{i=1}^{m} [(h_\theta(x_i) - y) . x_i]$$

Hypothesis: $h_\theta = \theta_0 + \theta_1 x$

Parameters: θ_0, θ_1

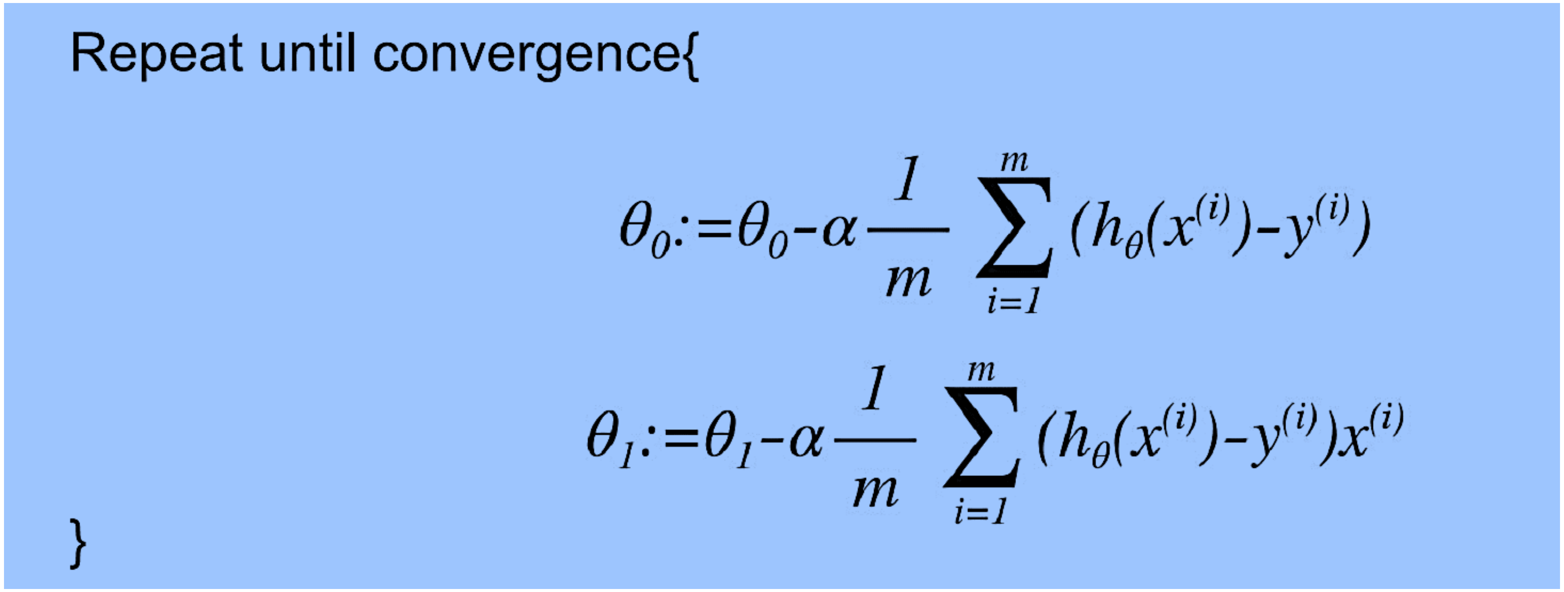
Cost Function: $J(\theta_0, \theta_1) = \dfrac{1}{2m} \displaystyle\sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2$

Goal: $\min\limits_{\theta_0, \theta_1} J(\theta_0, \theta_1)$?

Therefore,

$$\theta_j := \theta_j - \frac{\alpha}{m} \Sigma_{i=1}^{m} [(h_\theta(x_i) - y_i) . x_i]$$

Univariate feature



```
Repeat until convergence{
    $\theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})$
    $\theta_1 := \theta_1 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x^{(i)}$
}
```

Multivariate features

Area of site (1000 square feet)	Size of living place (1000 square feet)	Number of rooms	Ages in years	Selling price
3.472	0.998	7	42	25.9
3.531	1.500	7	62	29.5
$x^{(3)}$ 2.275	1.175	6	40	27.9
4.050	$x_2^{(4)}$ 1.232	6	54	25.9
...	...	...	...	...

- n : number of features
- $x^{(i)}$: input features of i-th training example
- $x_j^{(i)}$: value of feature j in i-th training example

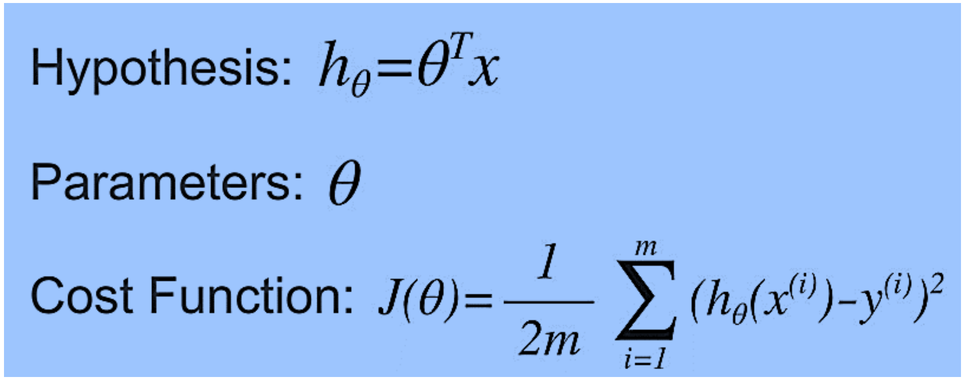
Multivariate features

- Hypothesis for multiple feature:

$$h_\theta(x) = \theta_0+\theta_1 x_1+\theta_2 x_2+\ldots+\theta_n x_n$$

- If we define $x_0 = I$, then we have,

$$X = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ . \\ . \\ . \\ x_n \end{bmatrix} \qquad \theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ . \\ . \\ . \\ \theta_n \end{bmatrix} \qquad h_\theta(x) = \theta^T x$$



Hypothesis: $h_\theta = \theta^T x$

Parameters: θ

Cost Function: $J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2$

Multivariate features

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta) \quad j=0,1,\ldots,n$$

$$\frac{\partial}{\partial \theta j} J(\theta) = \frac{\partial}{\partial \theta j} \frac{1}{2m} \Sigma_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2$$

$$= \frac{\partial}{\partial \theta j} \frac{1}{2m} \Sigma_{i=1}^{m} ((\theta_0 x_0^{(i)} + \ldots . + \theta_n x_n^{(i)}) - y^{(i)})^2$$

$$= \frac{1}{m} \Sigma_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2 x_j^{(i)}$$

Repeat{

$$\theta_j := \theta_j - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)} \quad j=0,1,\ldots,n$$

}

Stochastic, batch, mini-batch gradient descent

Linear Regression Cost Function,

$$J(\theta) = \frac{1}{2m}\Sigma_{i=1}^{m} (\hat{y}^{i}-y^{i})^{2} \quad [\text{m training data}]$$

$$\frac{\partial}{\partial\theta j}J(\theta) = \frac{1}{m}\Sigma_{i=1}^{m} (\hat{y}^{i}-y^{i})^{2} \,.\, x_{j}^{i}$$

Vanilla (Batch) GD,

$$\theta_{j} := \theta_{j} - \alpha\frac{\partial}{\partial\theta j}J(\theta)$$

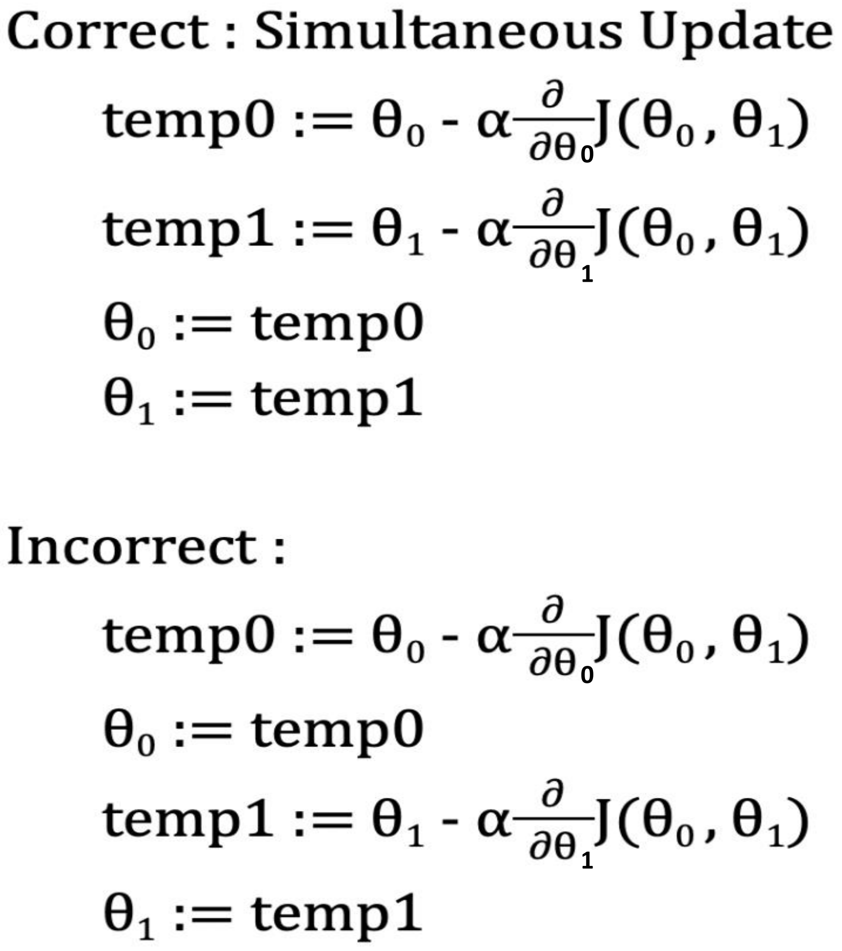
where, $\frac{\partial}{\partial\theta j}J(\theta) = \frac{1}{m}\Sigma_{i=1} (\hat{y}^{i}-y^{i})^{2} \,.\, x_{j}^{i}$

Statistic GD,

```
for i in range(m):
    $\theta_{j} := \theta_{j} - \alpha.(\hat{y}^{i}-y^{i})^{2} \,.\, x_{j}^{i}$
```

- In SGD, before for-looping, you need to randomly shuffle the training examples.
- In SGD, because it is using only one example at a time, its path to the minima is noisier (more random) than that of the batch gradient. But it’s ok as we are indifferent to the path, as long as it gives us the minimum AND the shorter training time.
- Mini-batch gradient descent uses n data points (instead of 1 sample in SGD) at each iteration.

Correct implementation



Correct : Simultaneous Update

$$\text{temp0} := \theta_0 - \alpha \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1)$$

$$\text{temp1} := \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1)$$

$$\theta_0 := \text{temp0}$$

$$\theta_1 := \text{temp1}$$

Incorrect :

$$\text{temp0} := \theta_0 - \alpha \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1)$$

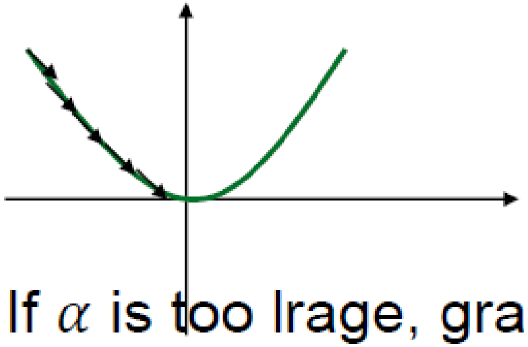
$$\theta_0 := \text{temp0}$$

$$\text{temp1} := \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1)$$

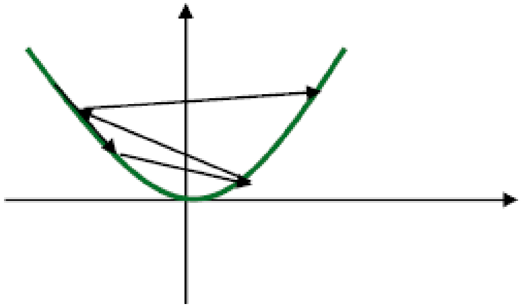
$$\theta_1 := \text{temp1}$$

Learning Rate

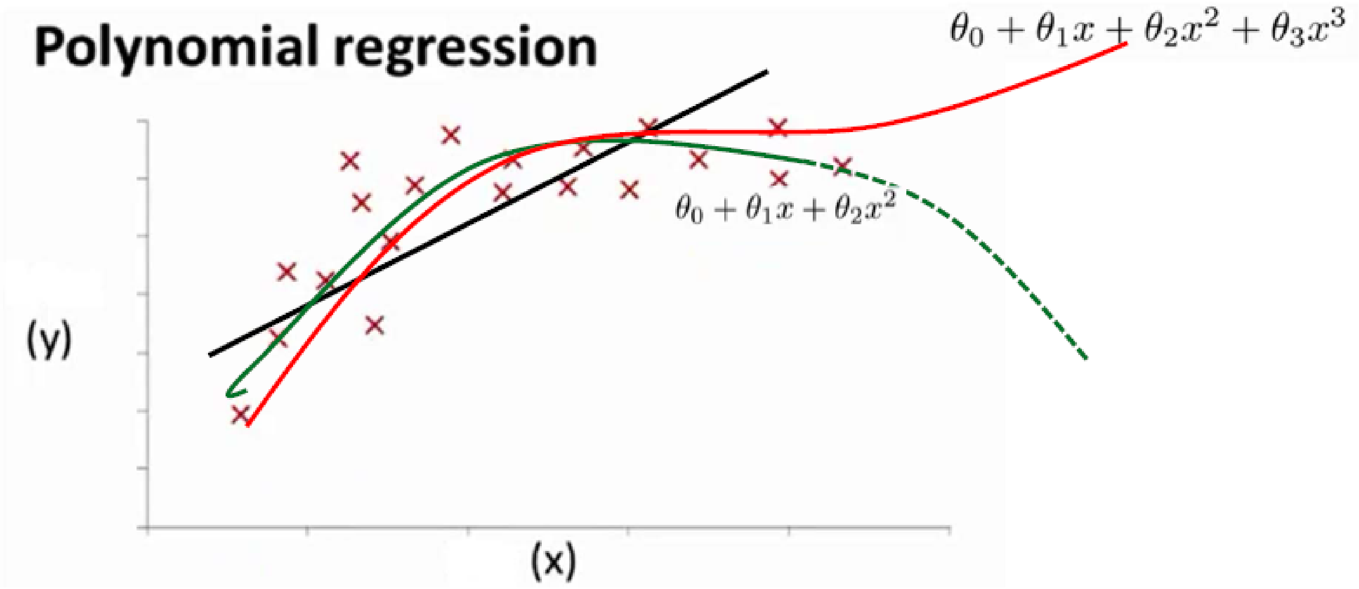
- If α is small, gradient descent can be slow



- If α is too lrage, gradient descent might overshoot the minimum

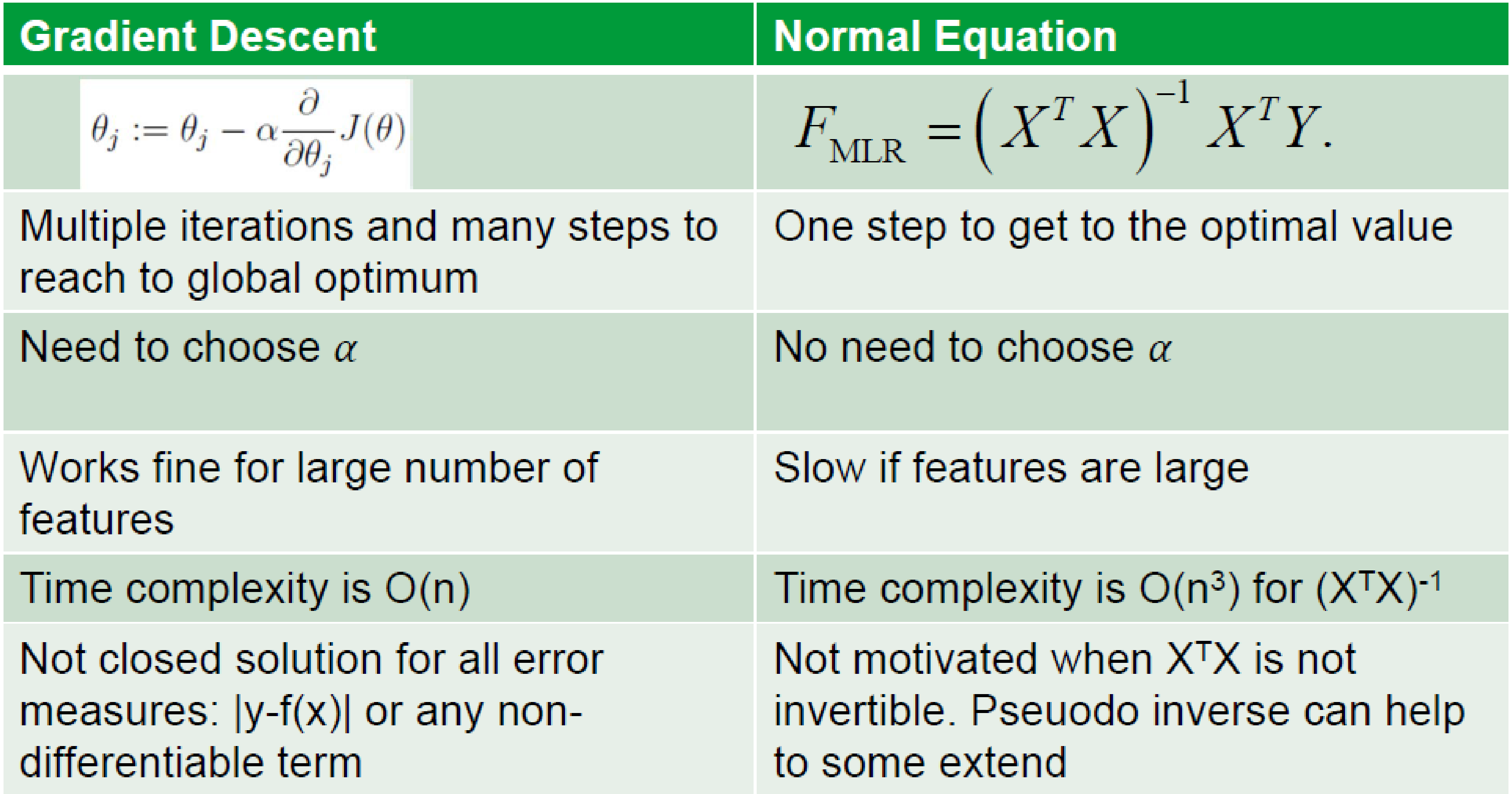


- Adding features to the model can make better fit



- Overfitting is problem
- Cross validation is one way to avoid overfitting

Gradient descent VS Normal equation



Gradient Descent	Normal Equation
$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$	$F_{\mathrm{MLR}} = \left(X^T X\right)^{-1} X^T Y.$
Multiple iterations and many steps to reach to global optimum	One step to get to the optimal value
Need to choose α	No need to choose α
Works fine for large number of features	Slow if features are large
Time complexity is O(n)	Time complexity is $O(n^3)$ for $(X^TX)^{-1}$
Not closed solution for all error measures: |y-f(x)| or any non-differentiable term	Not motivated when X^TX is not invertible. Pseuodo inverse can help to some extend

- Advantages of linear regression:
  - Simple to explain
  - Highly interpretable
  - Model training and prediction are fast
  - No tuning is required (excluding regularization)
  - Features don't need scaling
  - Can perform well with a small number of observations
  - Well-understood
- Disadvantages of linear regression:
  - Presumes a linear relationship between the features and the response
  - Performance is (generally) not competitive with the best supervised learning methods due to high bias
  - Can't automatically learn feature interactions

Introduction to Logistic Regression

- Limitations of linear regression
- Why logistic regression?

Logistic regression

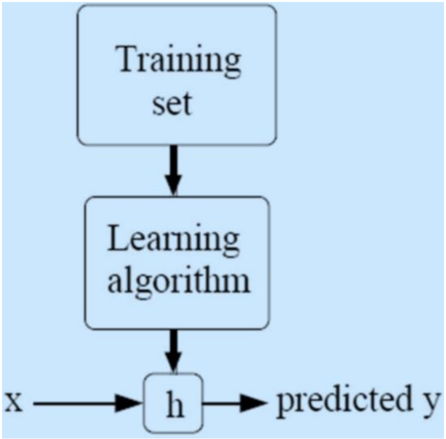
- Logistic regression is not a regression rather it is a classification algorithm

- Binary classification

$y \in \{0, 1\}$

0: “Negative Class” (e.g., benign tumor)

1: “Positive Class” (e.g., malignant tumor)

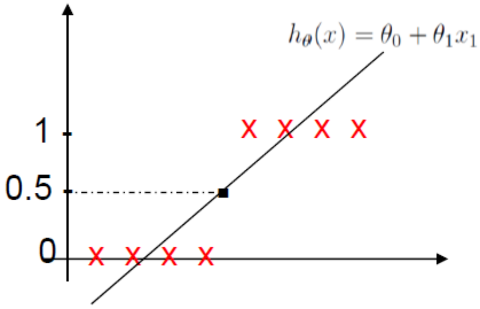


- Multi-class classification
  - y ∈ {0,1,2,3,…}

❑ A threshold is defined to classify

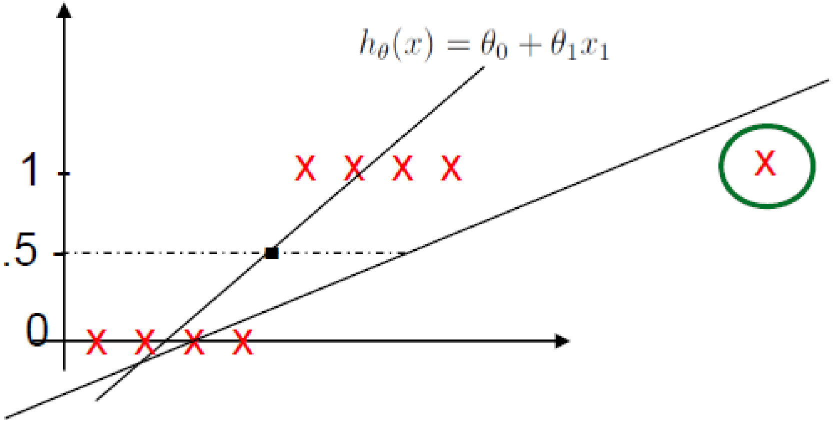
If $h_\theta(x) \geq 0.5$, predict “y = 1”

If $h_\theta(x) < 0.5$, predict “y = 0”



Logistic regression

- Applying linear regression for classification is often not useful



- $h_\theta(x)$ can be a large positive or negative value while y is 0 or 1
- Logistic regression : $0 \leq h_\theta(x) \leq 1$
  - A classification problem not regression despite the name

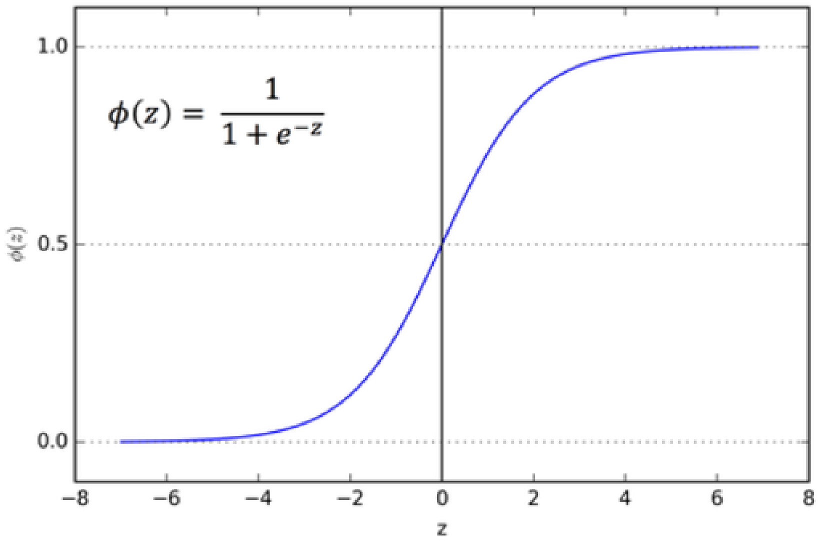
Hypothesis of logistic regression

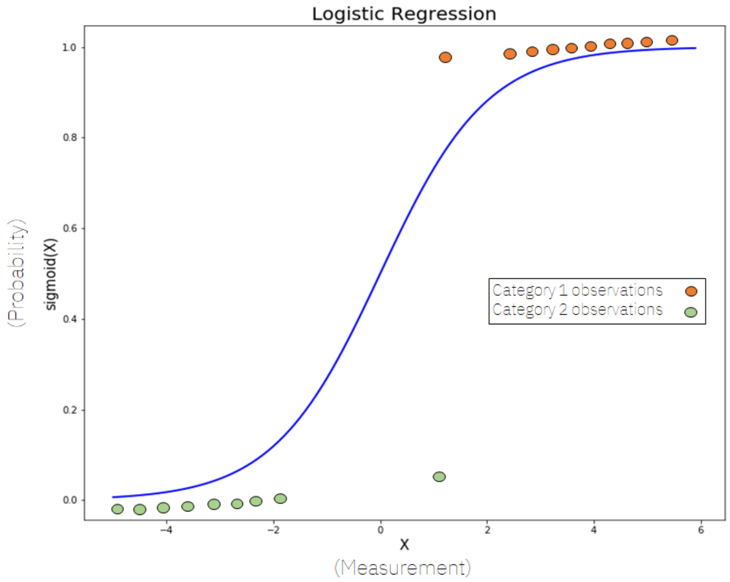
- If $h_\theta(x) \geq 0.5$, predict "y = 1"
- If $h_\theta(x) < 0.5$, predict "y = 0"

$$0 \leq h_\theta(x) \leq 1$$

- An explanation of logistic regression can begin with an explanation of the standard logistic function. The logistic function is a Sigmoid function, which takes any real value between zero and one. It is defined as:
- And if we plot it, the graph will be ***S*** curve,

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





Hypothesis of logistic regression

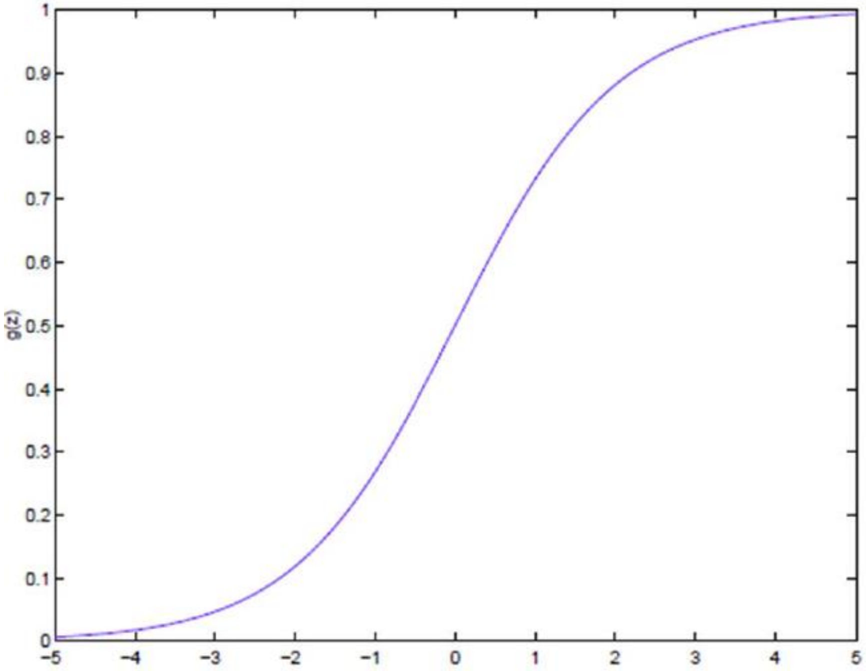
- Logistic or sigmoid function

$$h_\theta(x) = g(\theta^T x) = \frac{1}{1+e^{-\theta^T x}}$$

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

- Derivative of sigmoid:

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

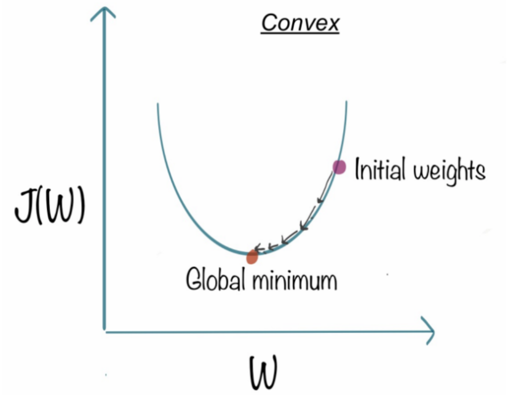


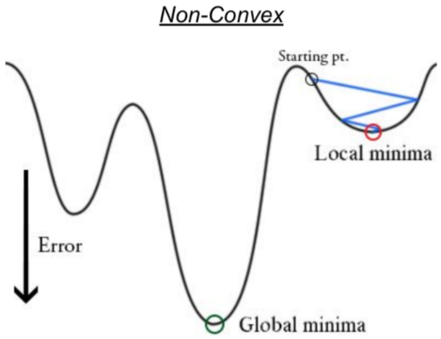
Cost Function of Logistic Regression

- Cost function of linear regression is convex:
- The same cost function for logistic regression is nonconvex because of nonlinear sigmoid function.
- If you plug in the linear regression cost function to sigmoid function then it becomes non-convex which may suffer from local minima problem.

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

$$h_\theta(x) = g(\theta^T x) = \frac{1}{1+e^{-\theta^T x}}$$





Cost Function of Logistic Regression

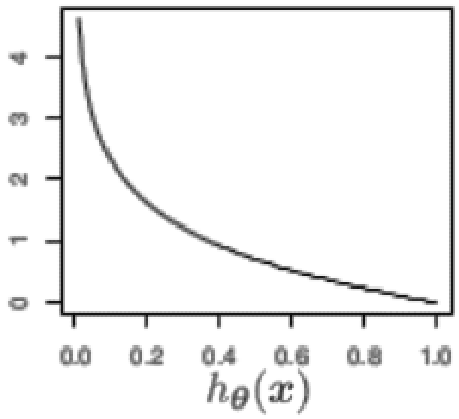
- We define the cost function of Logistic Regression as:

$$\text{Cost}(h_\theta(x), y) = \begin{cases} -\log(h_\theta(x)) & \text{if } y = 1 \\ -\log(1 - h_\theta(x)) & \text{if } y = 0 \end{cases}$$

$$J(\theta) = \frac{1}{m} \sum_{i=1}^{m} \text{Cost}(h_\theta(x^{(i)}), y^{(i)})$$

$$= -\frac{1}{m}[\sum_{i=1}^{m} y^{(i)} \log h_\theta(x^{(i)}) + (1 - y^{(i)}) \log (1 - h_\theta(x^{(i)}))]$$

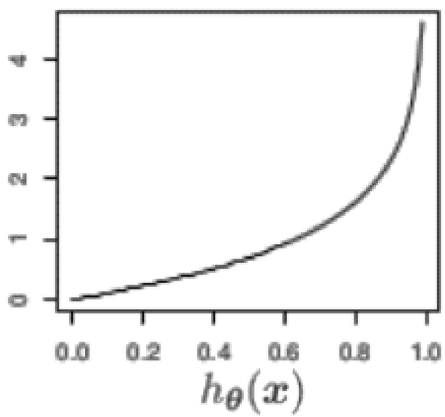
if $y = 1$



if $h_{\boldsymbol{\theta}}(\boldsymbol{x}) = 1$
then cost $= 0$

if $h_{\boldsymbol{\theta}}(\boldsymbol{x}) \to 0$
then cost $\to \infty$
predicted
$\text{prob}(y = 1 \mid \boldsymbol{x}; \boldsymbol{\theta}) = 0$
but $y = 1$

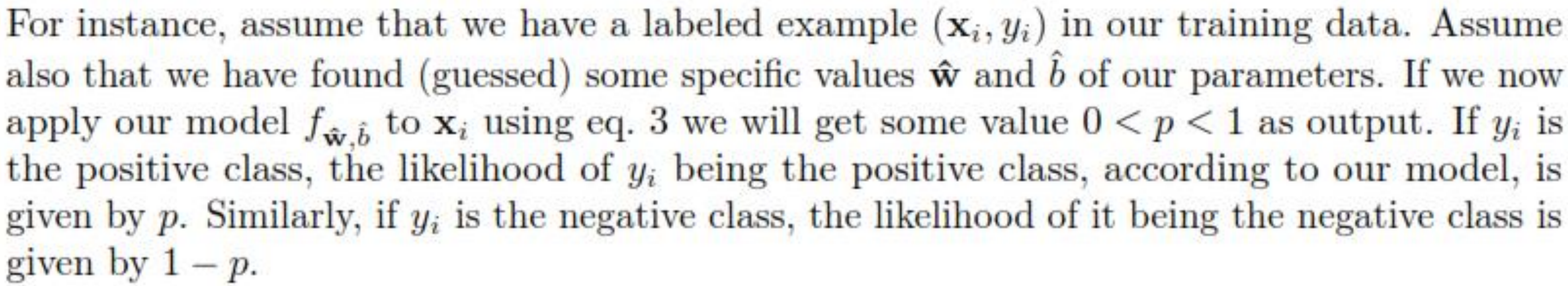
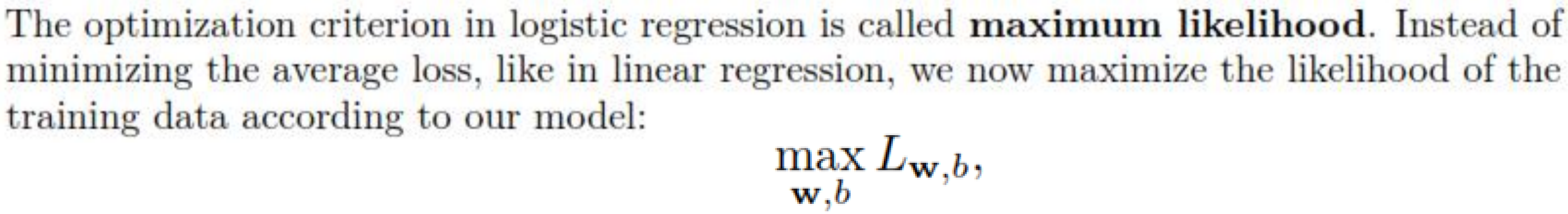
if $y = 0$



if $h_{\boldsymbol{\theta}}(\boldsymbol{x}) = 0$
then cost $= 0$

if $h_{\boldsymbol{\theta}}(\boldsymbol{x}) \to 1$
then cost $\to \infty$
predicted
$\text{prob}(y = 0 \mid \boldsymbol{x}; \boldsymbol{\theta}) = 0$
but $y = 0$

Cost function of logistic regression

- In logistic regression, instead of using a squared loss and trying to minimize the empirical risk, we maximize the likelihood of our training set according to the model.
- In statistics, the likelihood function defines how likely is the observation (an example) according to our model.
- 
  For instance, assume that we have a labeled example $(\mathbf{x}_i, y_i)$ in our training data. Assume also that we have found (guessed) some specific values $\hat{\mathbf{w}}$ and $\hat{b}$ of our parameters. If we now apply our model $f_{\hat{\mathbf{w}},\hat{b}}$ to $\mathbf{x}_i$ using eq. 3 we will get some value $0 < p < 1$ as output. If y_i is the positive class, the likelihood of y_i being the positive class, according to our model, is given by p. Similarly, if y_i is the negative class, the likelihood of it being the negative class is given by $1 - p$.
- 
  The optimization criterion in logistic regression is called **maximum likelihood**. Instead of minimizing the average loss, like in linear regression, we now maximize the likelihood of the training data according to our model:

$$\max_{\mathbf{w},b} L_{\mathbf{w},b},$$

Cost function of logistic regression

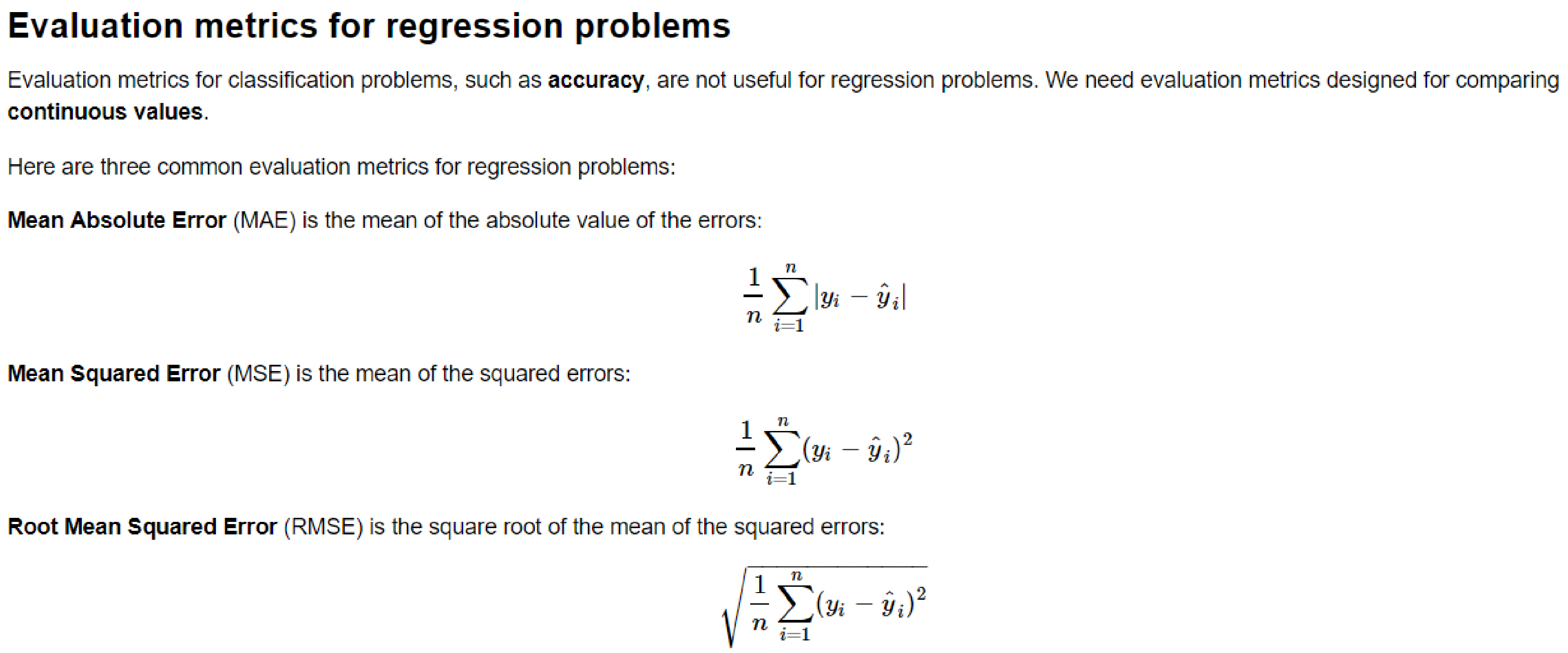
$$\begin{aligned} P(y=1 \mid x;\theta) &= h_\theta(x) \\ P(y=0 \mid x;\theta) &= 1 - h_\theta(x) \end{aligned}$$

- It can be written as : $p(y \mid x;\theta) = (h_\theta(x))^y \, (1 - h_\theta(x))^{1-y}$
- For m training example,the likelihood of the parameters:

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

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

- Advantages of logistic regression:
  - Highly interpretable (if you remember how)
  - Model training and prediction are fast
  - No tuning is required (excluding regularization)
  - Features don't need scaling
  - Can perform well with a small number of observations
  - Outputs well-calibrated predicted probabilities
- Disadvantages of logistic regression:
  - Presumes a linear relationship between the features and the log-odds of the response
  - Performance is (generally) not competitive with the best supervised learning methods
  - Can't automatically learn feature interactions



Evaluation metrics for regression problems

Evaluation metrics for classification problems, such as **accuracy**, are not useful for regression problems. We need evaluation metrics designed for comparing **continuous values**.

Here are three common evaluation metrics for regression problems:

Mean Absolute Error (MAE) is the mean of the absolute value of the errors:

$$\frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

Mean Squared Error (MSE) is the mean of the squared errors:

$$\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

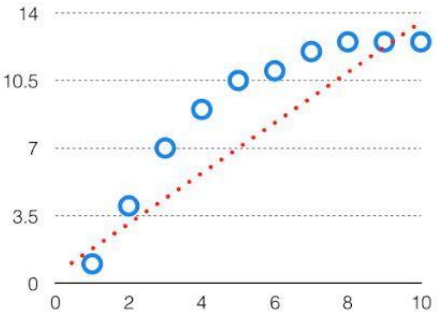
Root Mean Squared Error (RMSE) is the square root of the mean of the squared errors:

$$\sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

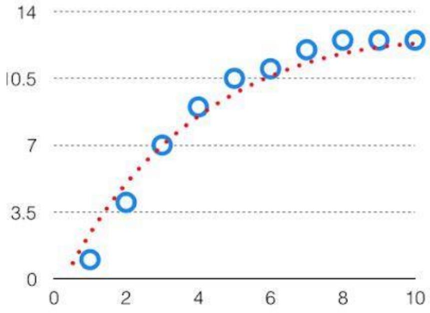
Bias-Variance

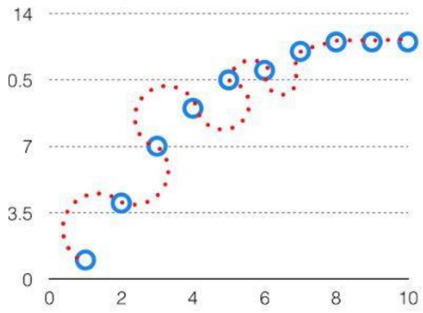
- **Bias:** The difference between the prediction of the values by the ML model and the correct value.
- Being high in biasing gives a large error in training as well as testing data. Its recommended that an algorithm should always be low biased to avoid the problem of underfitting.
- By high bias, the data predicted is in a straight line format, thus not fitting accurately in the data in the data set. Such fitting is known as **Underfitting** of Data.
- This happens when the hypothesis is too simpler or linear in nature

- **Variance:** The variability of model prediction for a given data point which tells us spread of our data is called the variance of the model.
- The model with high variance has a very complex fit to the training data and thus is not able to fit accurately on the data which it hasn't seen before. As a result, such models perform very well on training data but has high error rates on test data.
- When a model is high on variance, it is then said to as **Overfitting** of Data.
- Overfitting is fitting the training set accurately via complex curve and high order hypothesis but is not the solution as the error with unseen data is high.



$$h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2)$$

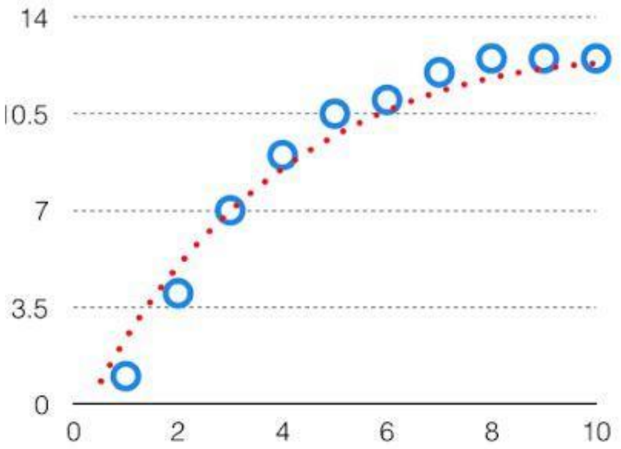


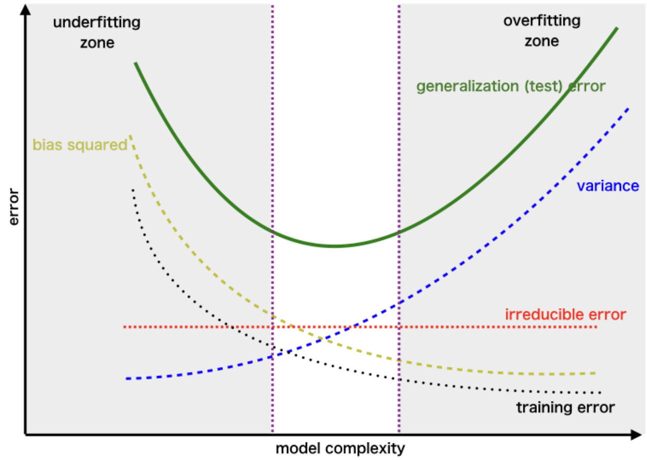


$$h_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2 \\ + \theta_3 x^3 + \theta_4 x^4$$

Bias-Variance trade-off

- If the algorithm is too simple (hypothesis with linear eq.) then it may be on high bias and low variance condition and thus is error-prone.
- If algorithms fit too complex (hypothesis with high degree eq.) then it may be on high variance and low bias. In this case, new entries will not perform well.
- There is something between both of these conditions, known as Trade-off or Bias Variance Trade-off.
- An algorithm can’t be more complex and less complex at the same time.





- This is referred to as the best point chosen for the training of the algorithm which gives low error in training as well as testing data.

Regularization

- Regularization (L1, L2)

L1 Regularization

$$\text{Cost} = \sum_{i=1} (y_i - \sum_{i=0} x_{ij} W_j)^2 + \lambda \sum_{j=0} |W_j|$$

L2 Regularization

$$\text{Cost} = \underbrace{\sum_{i=1} (y_i - \sum_{i=0} x_{ij} W_j)^2}_{\textbf{Loss Function}} + \underbrace{\lambda \sum_{j=0} (W_j)^2}_{\textbf{Regularization Term}}$$

Autoregression