Fundamentals of Machine Learning [DSE 2222]

Department of Data Science and Computer Applications

MIT, Manipal

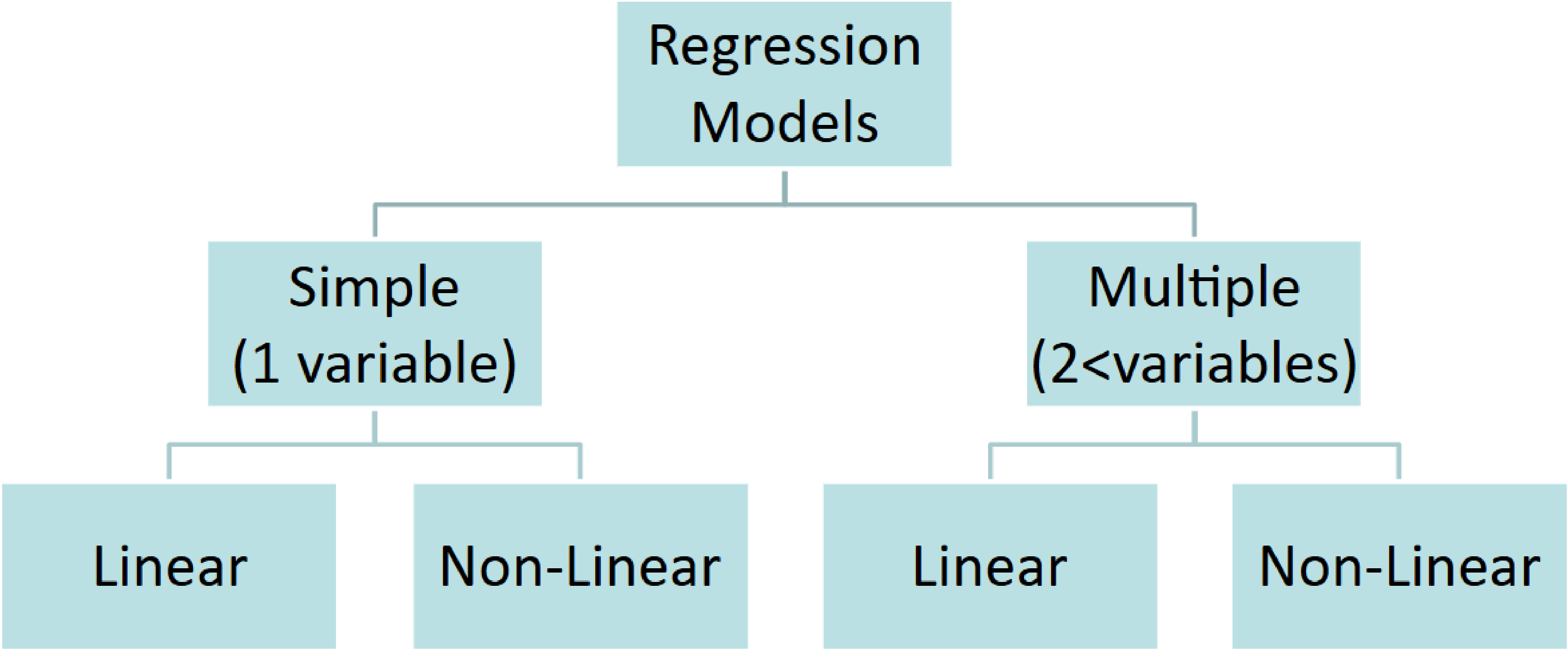
January 2025

Slide Set 2 – Regression Models

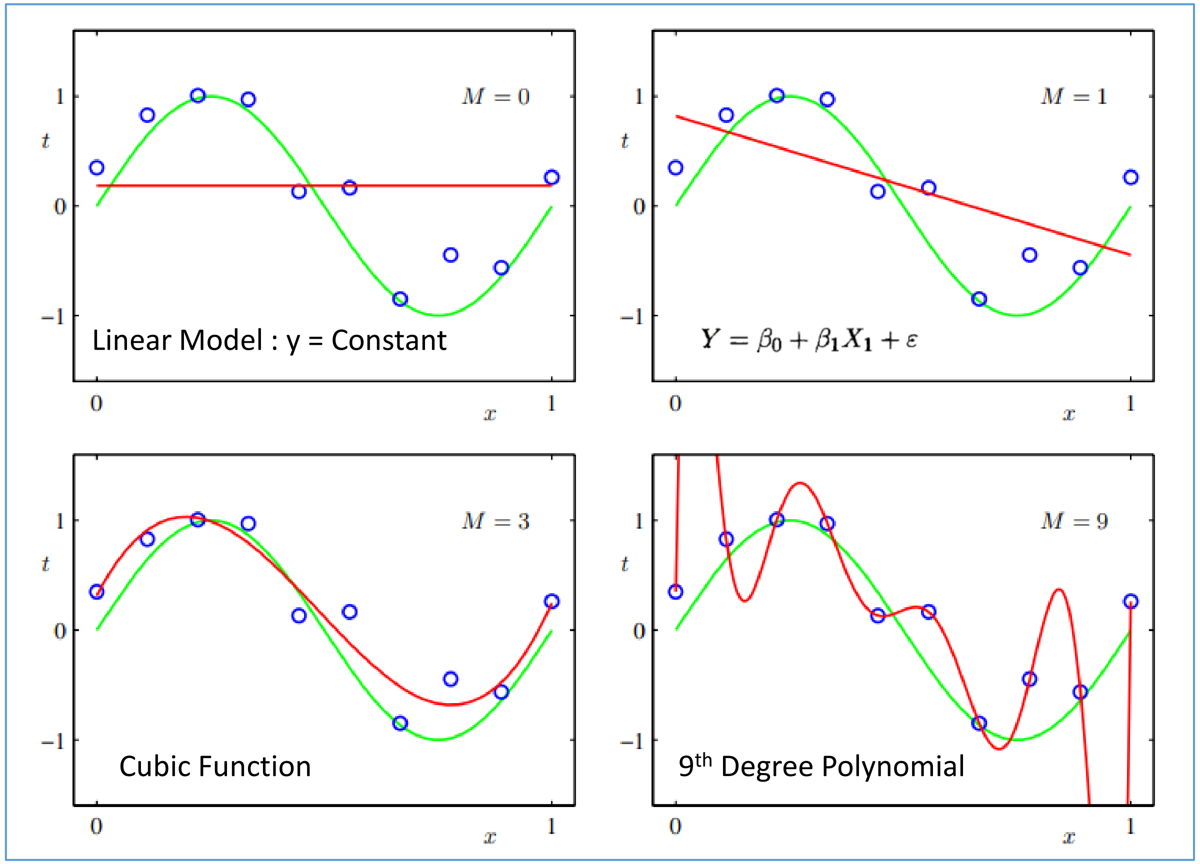
Regression Analysis

- Parametric Model
- Is a form of predictive modelling technique which investigates the relationship between
  - a **dependent** (target)
  - **independent variable (s)** (predictor).
- Used for forecasting, time series modelling and finding the causal effect relationship between the variables
- fit a curve / line to the data points, so that the differences between the distances of data points from the curve or line is minimized.

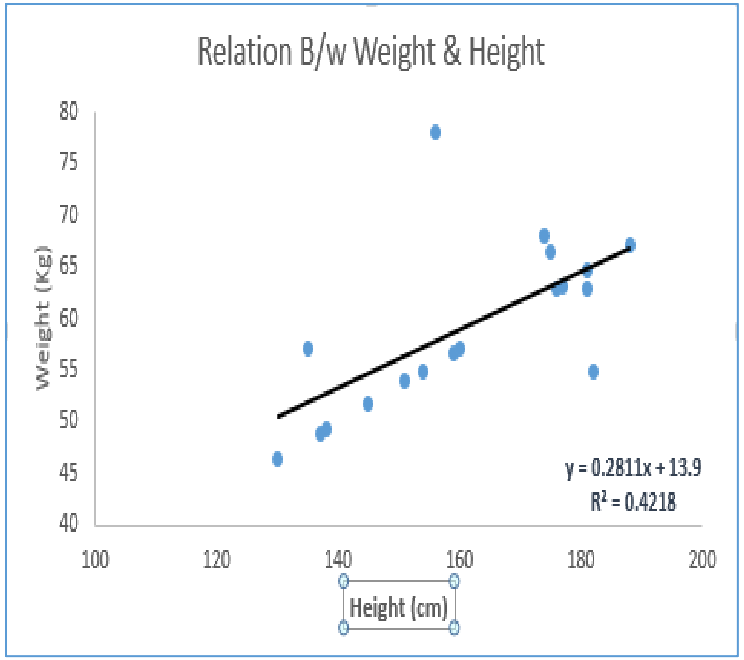
Type of Regression Models



Some fits to the data : which is best?



Linear Regression



$$Y=\overbrace{\beta_0}^{\text{Population Y-Intercept}} + \overbrace{\beta_1}^{\text{Population Slope}} x_1 + \overbrace{\epsilon}^{\text{Random Error}}$$

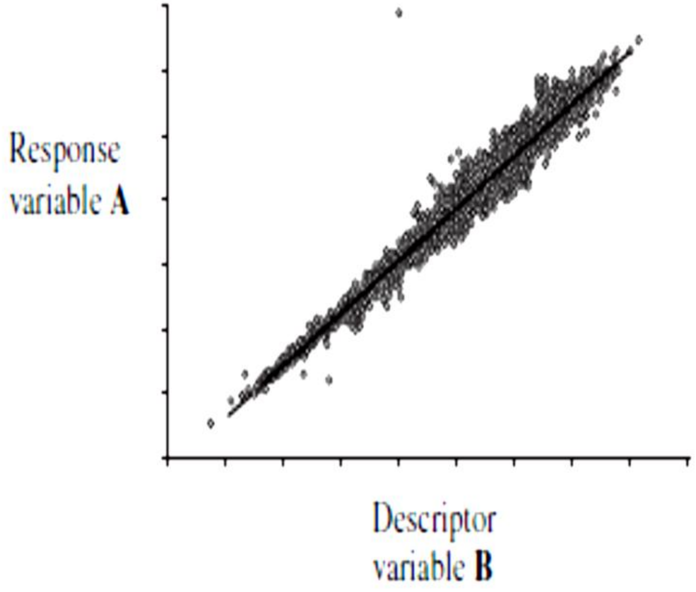
- There must be linear relationship between independent and dependent variables
- Relationship as a best fit line

$$Y = \beta_0 + \beta_1 X_1 + \varepsilon$$

- we can assume that there is some noise in the data which causes Random error Ɛ
- Error is normally distributed with mean 0, and standard deviations σ
- Called as Gaussian noise or white noise
- To get best fit line use **LEAST SQUARE METHOD**
  - It calculates the best-fit line for the observed data by minimizing the sum of the squares of the vertical deviations from each data point to the line.
- Linear Regression is very sensitive to Outliers.
- **Assumptions**
  - there is very little or no multi-collinearity in the data.
  - There is very little or no auto-correlation in the error terms.
  - The error terms must possess constant variance.

Simple Linear Regression

- Mathematical model that predicts continuous response variable
- Where there appears to be a linear relationship between two variables
- The prediction model is an equation of
  - y = a + b.x
- Method of Least squares



$$b = \frac{\sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n} (x_i - \bar{x})^2}$$

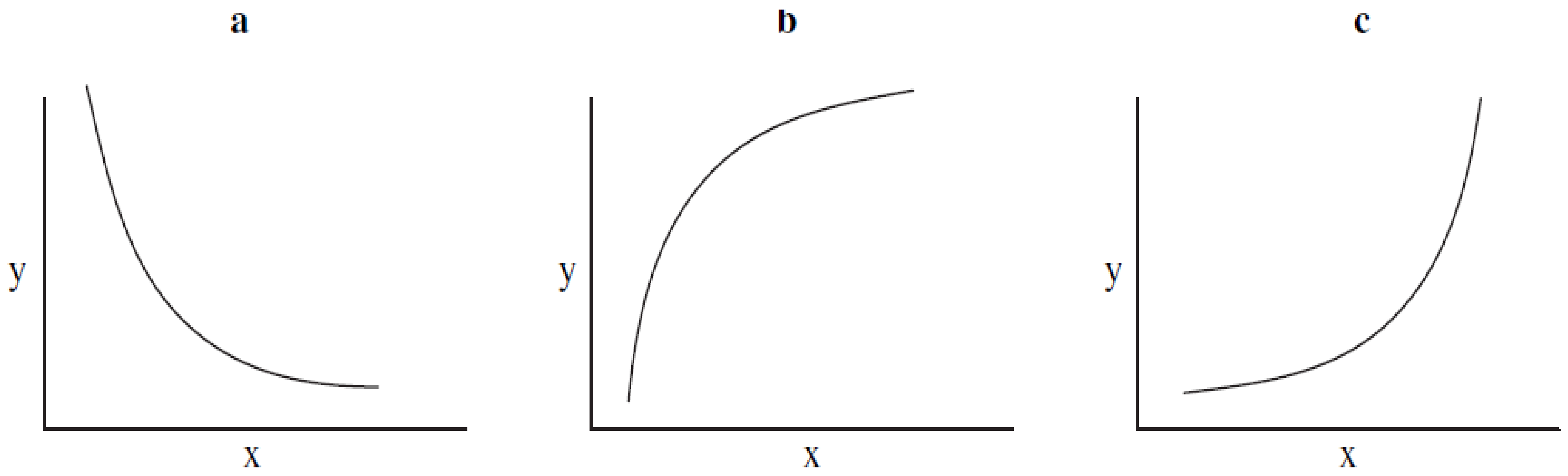
$$a = \bar{y} - b\bar{x}$$

Example 2

Xi	Yi	(h(x)-Yi)
2	69	
9	98	
5	82	
5	77	
3	71	
7	84	
1	55	
8	94	
6	84	
2	64	

Simple non-linear regression

- transform the nonlinear relationship to a linear relationship using a mathematical transformation
    - Situation a: Transformations on the x, y or both x and y variables such as log or square root.
    - Situation b: Transformation on the x variable such as square root, log or -1/x.
    - Situation c: Transformation on the y variable such as square root, log or -1/y.



Example 2 – Simple Non Linear Regression

x	y
3	4
6	5
9	7
8	6
10	8
11	10
12	12
13	14
13.5	16
14	18
14.5	22
15	28
15.2	35
15.3	42

Cost Function

- To measure accuracy of the hypothesis function we use a cost function. It’s an average difference of all the hypothesis results with inputs from x's and the actual output y's.
- $J(\theta_0, \theta_1) = \frac{1}{2m}\sum_{i=1}^{m}(\hat{y}_i - \mathrm{y}_i)^2 = \sum_{i=1}^{m}(h_\theta(xi) - yi)^2$
- To break it apart, it is $\frac{1}{2}\ \bar{x}$ where $\bar{x}$ is the mean of the squares of $(h_\theta(xi) - yi)$, or the difference between the predicted value and the actual value.
- This function is also as "Squared error function", or "Mean squared error"

Cost Function

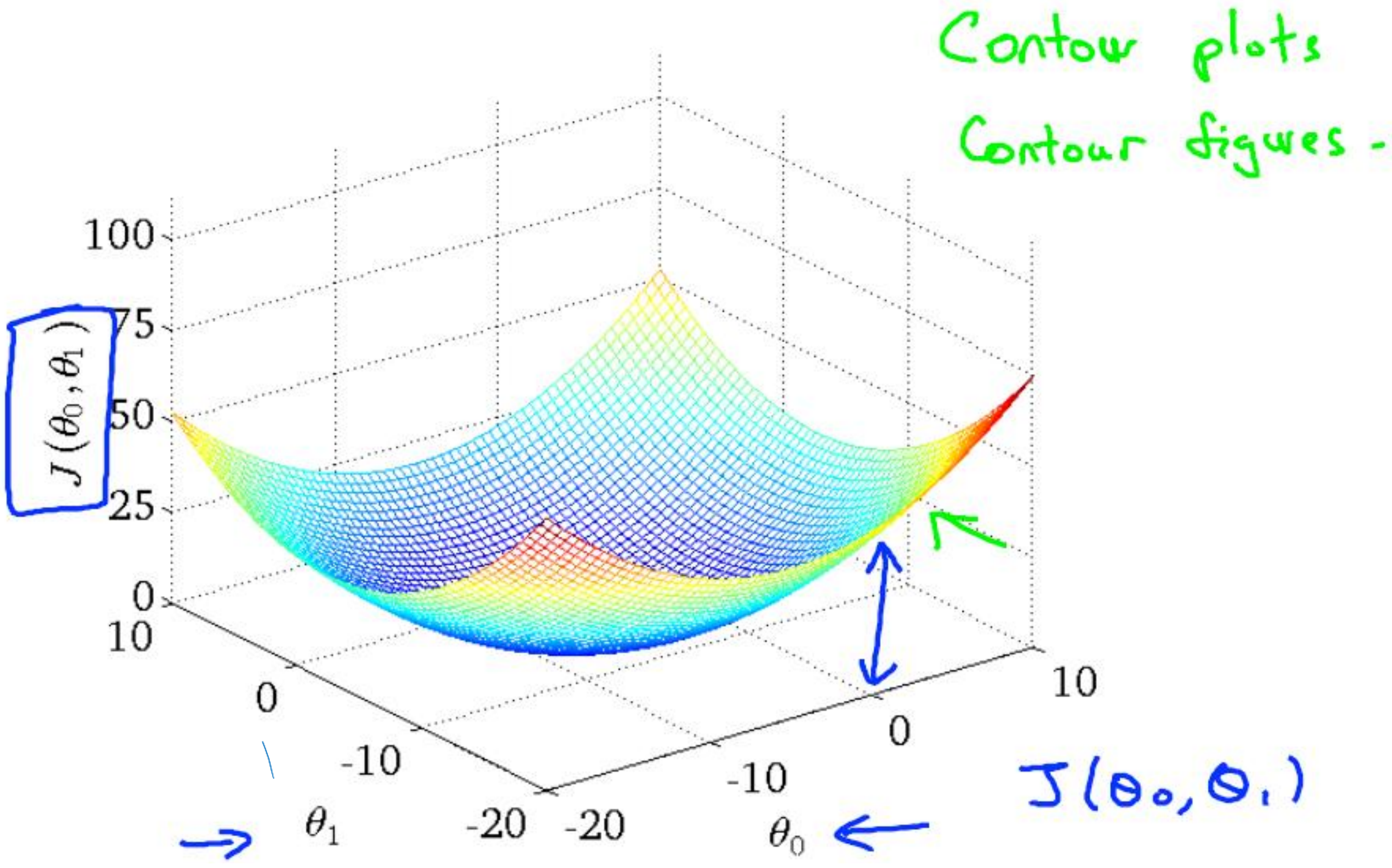
Hypothesis: $h_\theta(x) = \theta_0 + \theta_1 x$

Parameters: θ_0, θ_1

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

Goal: $\underset{\theta_0, \theta_1}{\text{minimize}}\ J(\theta_0, \theta_1)$

Contour Plot



Gradient Descent

- We want to choose θ so as to minimize *J(*θ*)*. To do that we use a search algorithm that
  - starts with some "initial guess" for θ.
  - and repeatedly changes θ to make *J(*θ*)* smaller, until converge to a value of θ that minimizes *J(*θ*)*.
- The **gradient descent algorithm**, which starts with some initial θ, and repeatedly performs the update θ. The algorithm can be represented as

Repeate until convergence

{

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

}

where

- j=0,1 represents the feature index number.

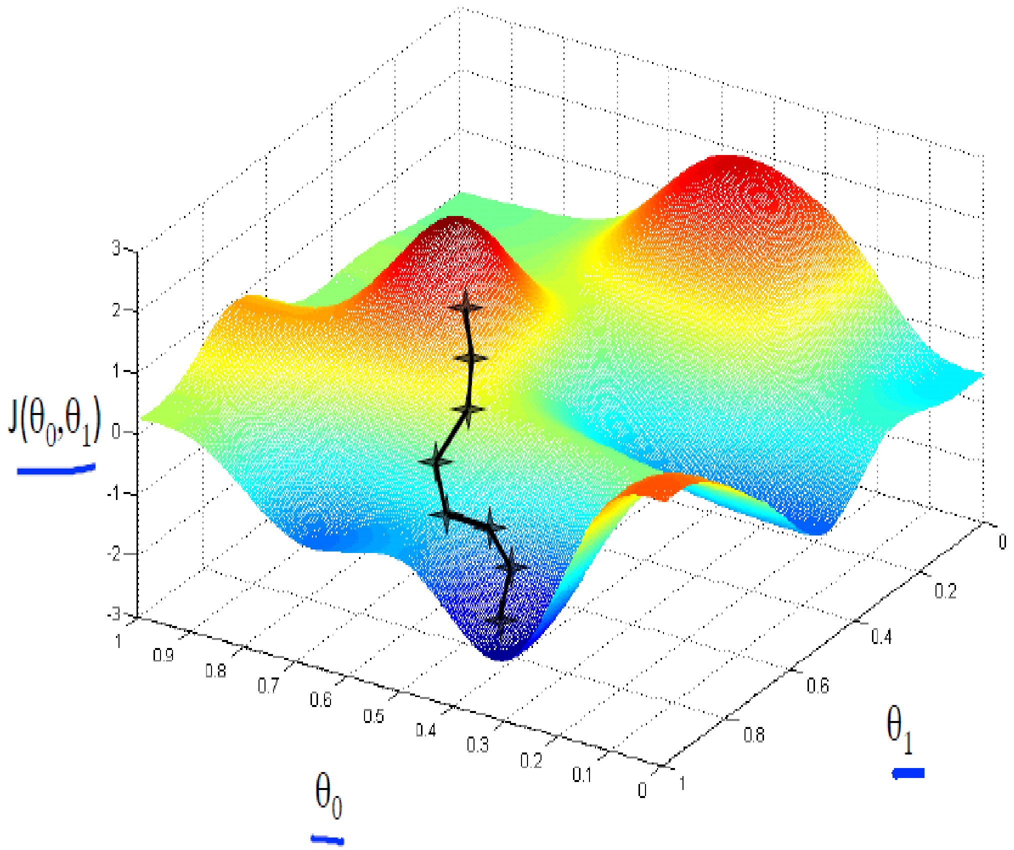
Gradient Descent

Have some function $J(\theta_0, \theta_1)$ $J(\theta_0, \theta_1, \theta_2, \dots, \theta_n)$

Want $\min_{\theta_0, \theta_1} J(\theta_0, \theta_1)$ $\min_{\theta_0, \dots, \theta_n} J(\theta_0, \dots, \theta_n)$

Outline:

- Start with some θ_0, θ_1 (say $\theta_0 = 0, \theta_1 = 0$)
- Keep changing θ_0, θ_1 to reduce $J(\theta_0, \theta_1)$ until we hopefully end up at a minimum



Gradient descent algorithm

$$
\begin{aligned}
&\text{repeat until convergence } \{ \\
&\quad \theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1) \\
&\qquad (\text{for } j = 1 \text{ and } j = 0) \\
&\}
\end{aligned}
$$

Linear Regression Model

$$h_\theta(x) = \theta_0 + \theta_1 x$$

$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right)^2$$

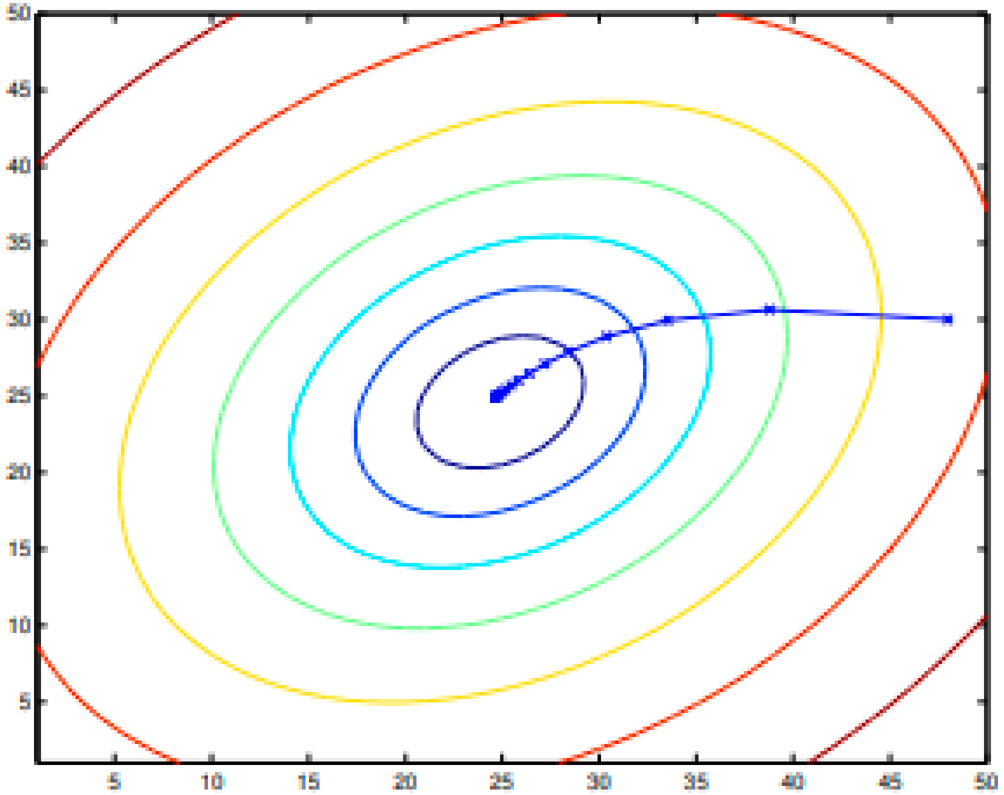
- This update is simultaneously performed for all values of j
- Here, α is called the learning rate.
- In order to implement this algorithm, we have to work out what is the partial derivative term on the right hand side.

$$
\begin{aligned}
\frac{\partial}{\partial \theta_j} J(\theta) &= \frac{\partial}{\partial \theta_j} \frac{1}{2} \left(h_\theta(x) - y\right)^2 \\
&= 2 \cdot \frac{1}{2} \left(h_\theta(x) - y\right) \cdot \frac{\partial}{\partial \theta_j} \left(h_\theta(x) - y\right) \\
&= \left(h_\theta(x) - y\right) \cdot \frac{\partial}{\partial \theta_j} \left(\sum_{i=0}^{d} \theta_i x_i - y \right) \\
&= \left(h_\theta(x) - y\right) x_j
\end{aligned}
$$

Batch gradient descent

- This method looks at every example in the entire training set on every step
- gradient descent can be susceptible to local minima in general, the optimization problem we have posed here for linear regression has only one global, and no other local optima.
- Gradient descent always converges to the global minimum. Indeed, J is a convex quadratic function.

- example



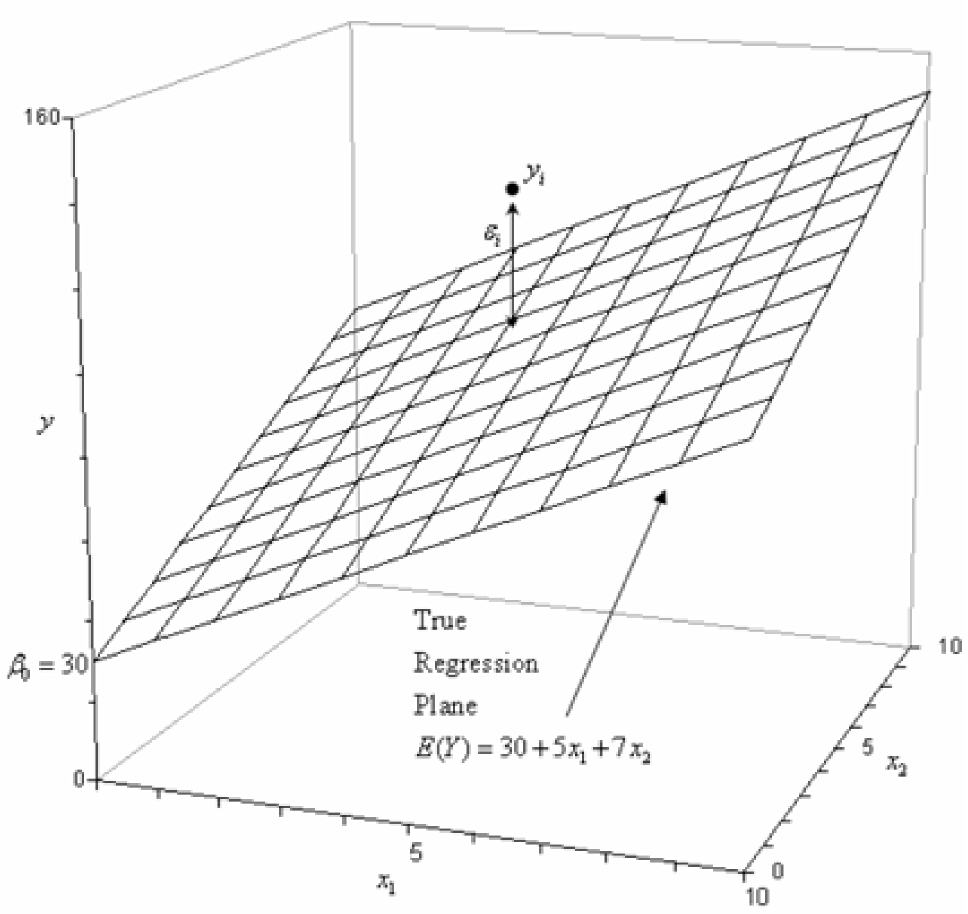
Stochastic gradient descent

- In this algorithm, we repeatedly run through the training set, and each time we encounter a training example, we update the parameters according to the gradient of the error with respect to that single training example only
- Batch gradient descent has to scan through the entire training set before taking a single step
- if n is large—stochastic gradient descent can start making progress right away, and continues to make progress with each example it looks at.

Relevant Terminology

- Multicollinearity
  - When the independent variables are highly correlated to each other, then variables are said to possess multicollinearity.
  - It makes task complex in selecting the important featured variables.
  - can increase the variance of the coefficient estimates and make the estimates very sensitive to minor changes in the model.
- Autocorrelation
  - Presence of correlation in error terms
  - refers to the degree of correlation between the values of the same variables across different observations in the data.
- Outliers
  - In every dataset, there must be some data points that have low or high value as compared to other data points
  - those data points don't relate to the population termed as outliers, an extreme value.
- Heteroscedasticity
  - systematic change in the spread of the residuals over the range of measured values.
  - The error terms must possess constant variance.
  - Absence of constant variance leads to **heteroskedestacity**.

Multiple Linear Regression

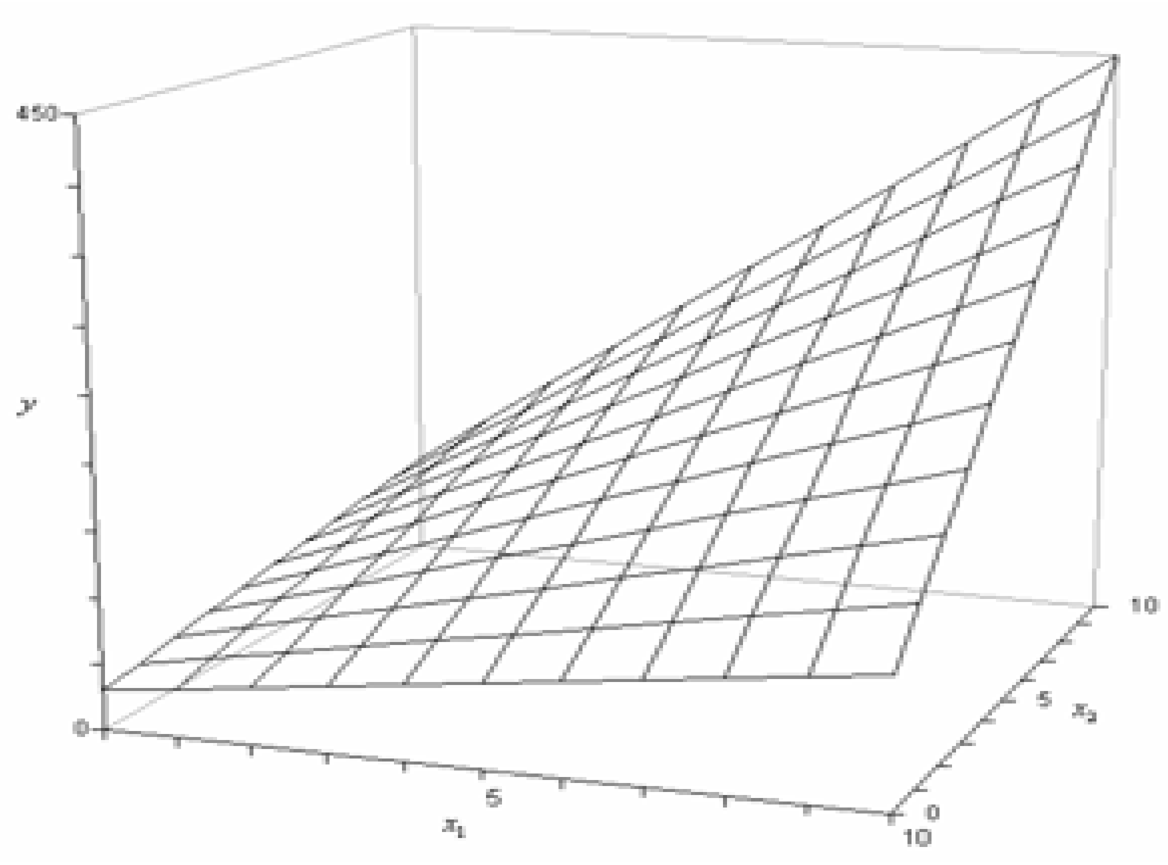


$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \varepsilon,$$

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_n X_n$$

- Suffers from
  - Multicollinearity, autocorrelation, heteroskedasticity.
- Multicollinearity can increase the variance of the coefficient estimates and make the estimates very sensitive to minor changes in the model.
- In case of multiple independent variables, we can go with **step wise approach** for selection of most significant independent variables.

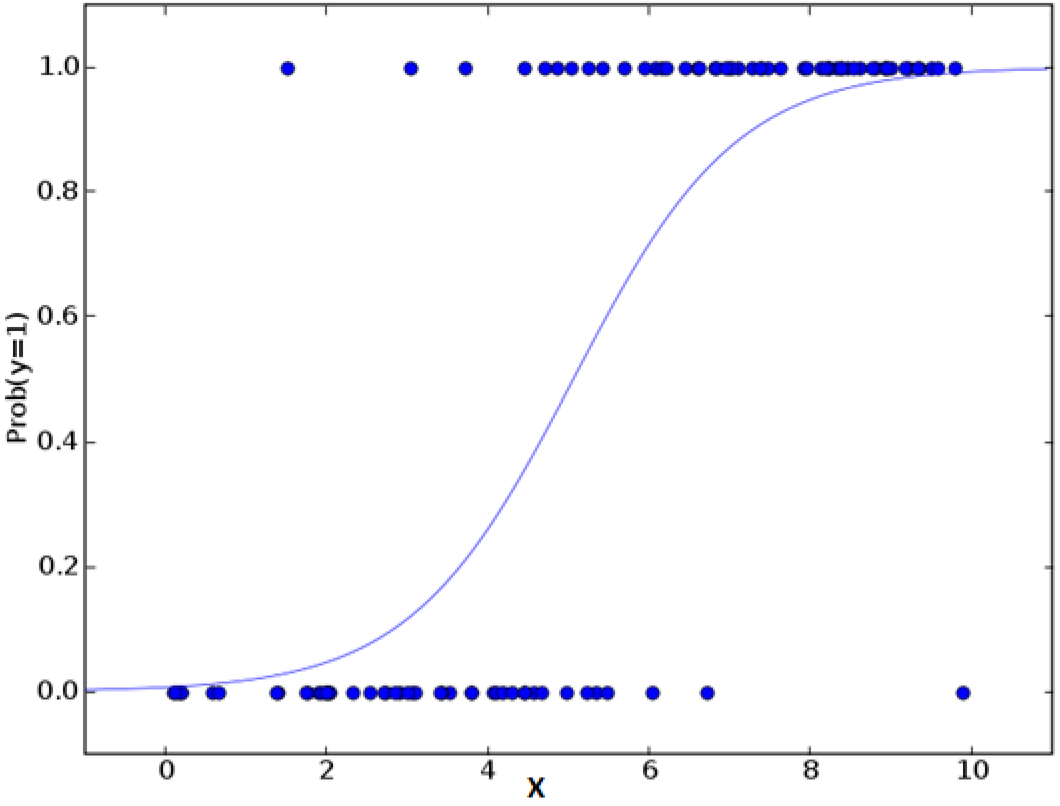
Polynomial Regression



$$Y = \beta_0 + \beta_1 x_1 + \beta_2 x_1^2 + \beta_3 x_1^3 + \epsilon$$

- the relationship between the independent variable *x* and the dependent variable *y* is modelled as an *n*th degree polynomial in *x*.
- is considered to be a special case of multiple linear regression.
- contain squared and higher order terms of the predictor variables making the response surface curvilinear.

Logistic Regression



- The nature of target or dependent variable is dichotomous, which means there would be only two possible classes.
- doesn’t require linear relationship between dependent and independent variables.
- Types include
  - Binomial
  - Multinomial
    - represent “Type A” or “Type B” or “Type C”.
  - Ordinal
    - represent “poor” or “good”, “very good”,
- Is used to find the probability of event=Success and event=Failure.
- **y = e^(b0 + b1*x) / (1 + e^(b0 + b1*x))**
- the parameters are chosen to maximize the likelihood of observing the sample values rather than minimizing the sum of squared errors
- Assumption:
  - The independent variables should not be correlated with each other i.e. **no multi collinearity**.

Stepwise Regression

- The aim is to maximize the prediction power with minimum number of predictor variables.
- While dealing with multiple independent variables, fits the regression model by adding/dropping co-variates one at a time based on a specified criterion.
- The selection of independent variables is done with the help of an automatic process, which involves *no* human intervention.
- This feat is achieved by observing statistical values like R-square, t-stats and AIC metric to discern significant variables.
- Stepwise regression methods are :
  - Forward selection starts with most significant predictor in the model and adds variable for each step.
  - Backward elimination starts with all predictors in the model and removes the least significant variable for each step.

Ridge Regression

- when the data suffers from multicollinearity (independent variables are highly correlated).
- we not only minimize the sum of squared residuals but also penalize the size of parameter estimates, in order to shrink them towards zero:
- By adding a degree of bias to the regression estimates, ridge regression reduces the standard errors.
- Above, we saw the equation for linear regression. :
    - y=a+b*x+e
    - [error term is the value needed to correct for a prediction error between the observed and predicted value]
    - => y= a+ b_1x_1+ b_2x_2+....+e, for multiple independent variables.
- Solves the multicollinearity problem through shrinkage parameter λ (lambda)
- The coefficients of correlated predictors are similar

$$= \underset{\beta\in\mathbb{R}^p}{\operatorname{argmin}} \underbrace{\|y - X\beta\|_2^2}_{\text{Loss}} + \lambda \underbrace{\|\beta\|_2^2}_{\text{Penalty}}$$

Lasso Regression

- Least Absolute Shrinkage and Selection Operator
- Penalizes the absolute size of the regression coefficients.
- If group of predictors are highly correlated, lasso picks only one of them and shrinks the others to zero which certainly helps in feature selection
- one of the correlated predictors has a larger coefficient, while the rest are (nearly) zeroed.
- it reduces the variability and improving the accuracy of linear regression models

$$L_{lasso}(\hat{\beta}) = \sum_{i=1}^{n}(y_i - x_i'\hat{\beta})^2 + \lambda \sum_{j=1}^{m}|\hat{\beta}_j|.$$

Elastic Net Regression

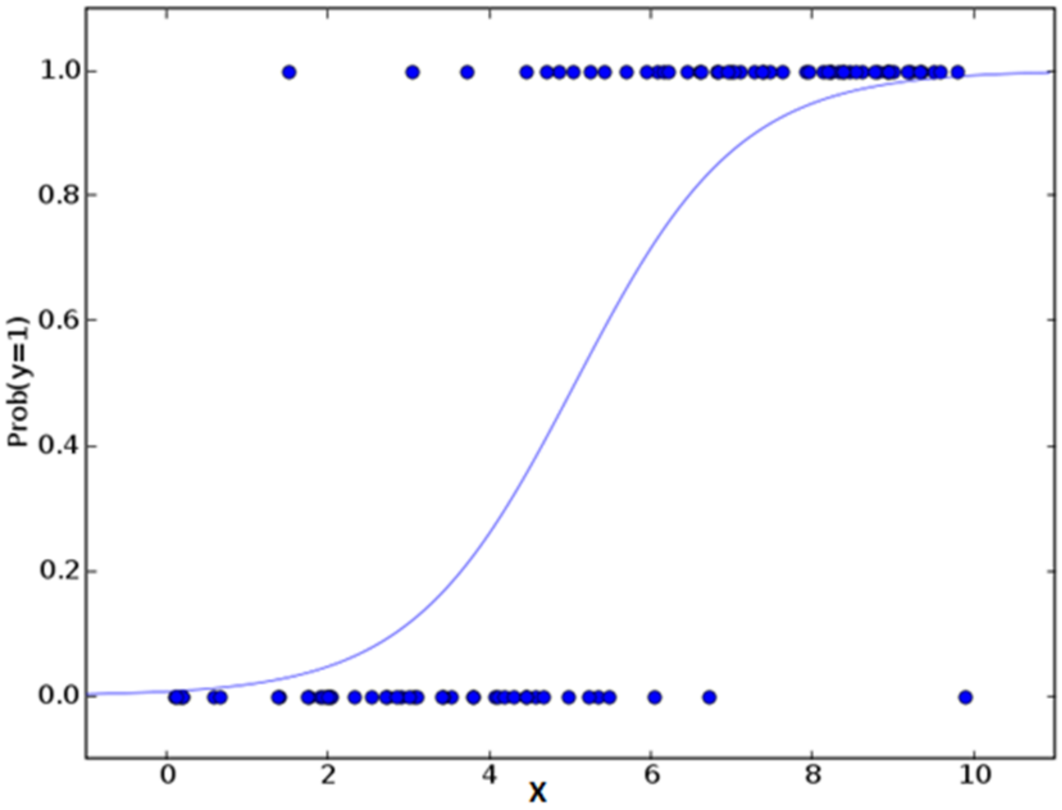
- hybrid of Lasso and Ridge Regression techniques.
- It is trained with L1 and L2 prior as regularizer.
- It is recommended to use when the number of predictors is very much higher than the number of observations.
- is useful when there are multiple features which are correlated.
- Lasso is likely to pick one of these at random, while elastic-net is likely to pick both.
- It encourages group effect in case of highly correlated variables
- There are no limitations on the number of selected variables

$$L_{enet}(\hat{\beta}) = \frac{\sum_{i=1}^{n}(y_i - x_i'\hat{\beta})^2}{2n} + \lambda\left(\frac{1-\alpha}{2}\sum_{j=1}^{m}\hat{\beta}_j^2 + \alpha\sum_{j=1}^{m}|\hat{\beta}_j|\right),$$

where α is the mixing parameter between ridge ($\alpha = 0$) and lasso ($\alpha = 1$).

Logistic Regression

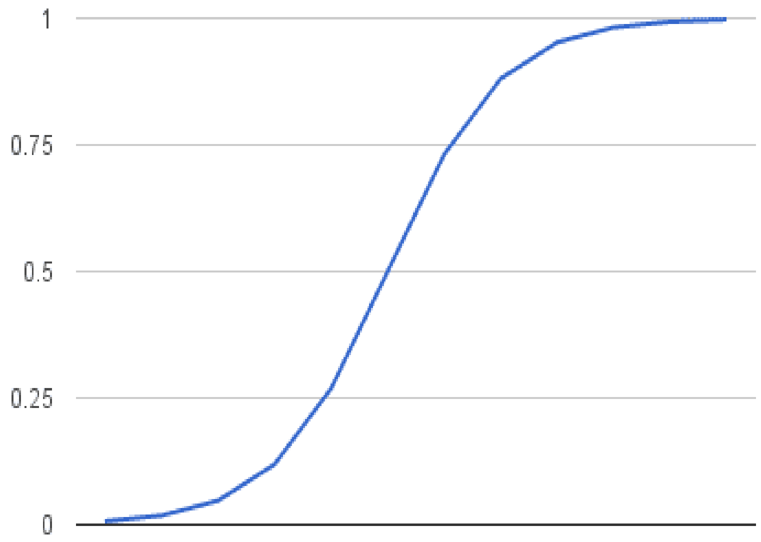
- Classification
  - Email – Spam/Not Spam
  - Tumor –is Malignant/Benign
- y Ɛ {1,0} - 1 is positive class and 0 is negative class
- Can be extended to y Ɛ {0,1,2,3}
- Threshold classifier output $h_\theta(x)$ at 0.5
  - If $h_\theta(x) >= 0.5$ then $y = 1$
  - If $h_\theta(x) < 0.5$ then $y = 0$
- Logistic Regression :
  - $0 <= h_\theta(x) <= 1$



Logistic Function

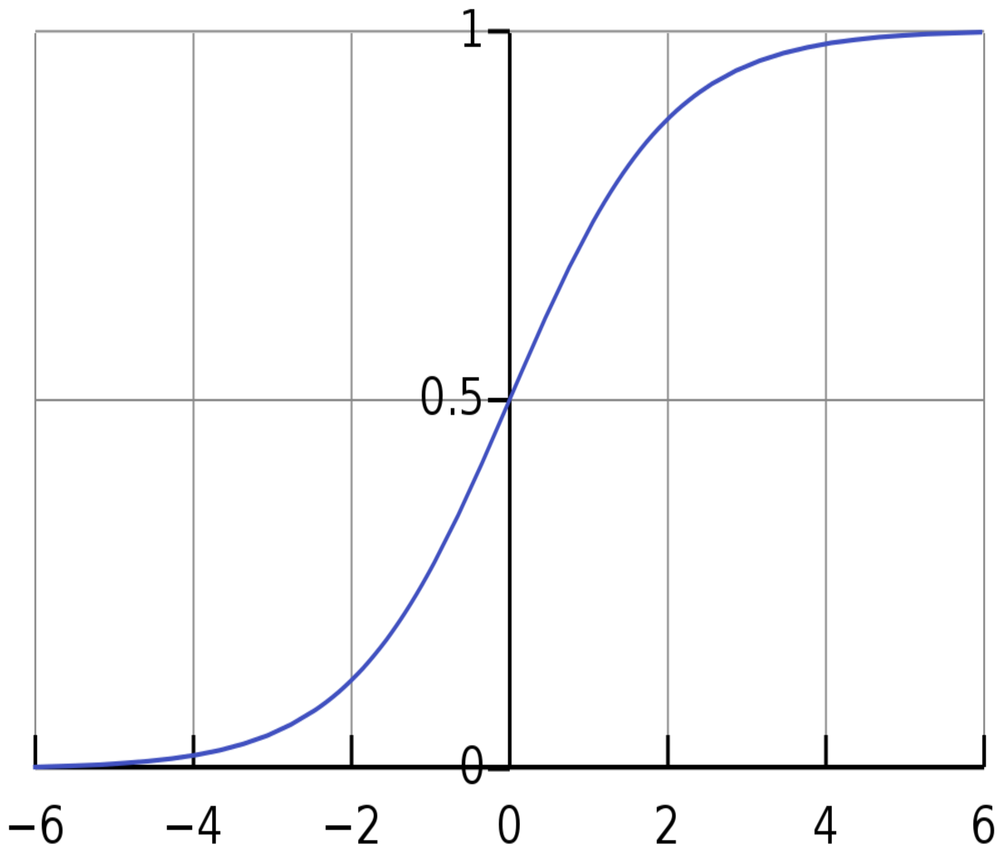
- The logistic function is defined as: transformed = 1 / (1 + e^-x)

X	Transformed
-5	0.006692850924
-4	0.01798620996
-3	0.04742587318
-2	0.119202922
-1	0.2689414214
0	0.5
1	0.7310585786
2	0.880797078
3	0.9525741268
4	0.98201379
5	0.9933071491



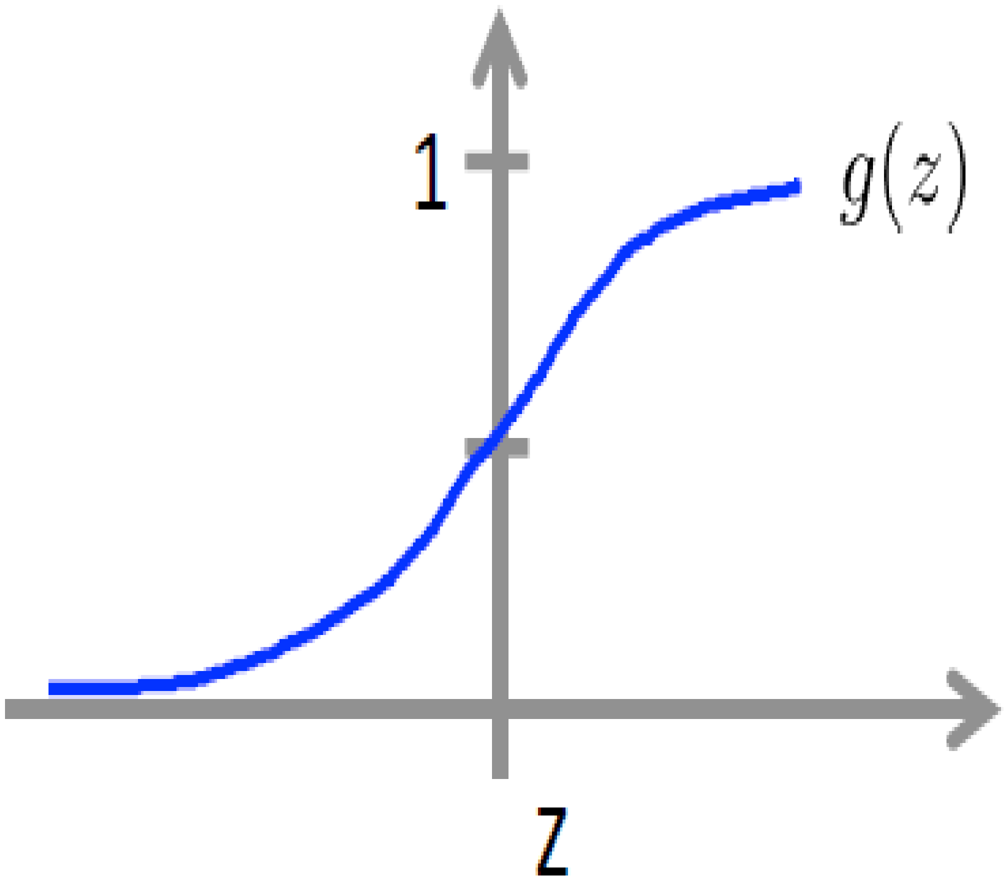
Logistic Regression Model

- Logistic Regression : $0 <= h_\Theta(x) <= 1$
- $h_\Theta(x) = g(\Theta^T x)$
- $g(z) = \frac{1}{1+e^{-\Theta T x}}$ => Sigmoid or Logistic Function
- => $h_\Theta(x) = \frac{1}{1+e^{-\Theta T x}}$
- Where e is the base of the natural logarithms (Euler's number)
- $h_\Theta(x)$ is estimated probability that y=1 on input x
- $h_\Theta(x) = P(y=1|x; \Theta)$
- Since $P(y=1|x; \Theta) + P(y=0|x; \Theta) = 1$



Logistic Regression – Decision Boundary

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



- Prob(y=1)
  - If $h_\Theta(x) >= 0.5$, $\Theta^T X >= 0$
- Prob(y=0)
  - If $h_\Theta(x) < 0.5$, $\Theta^T X < 0$

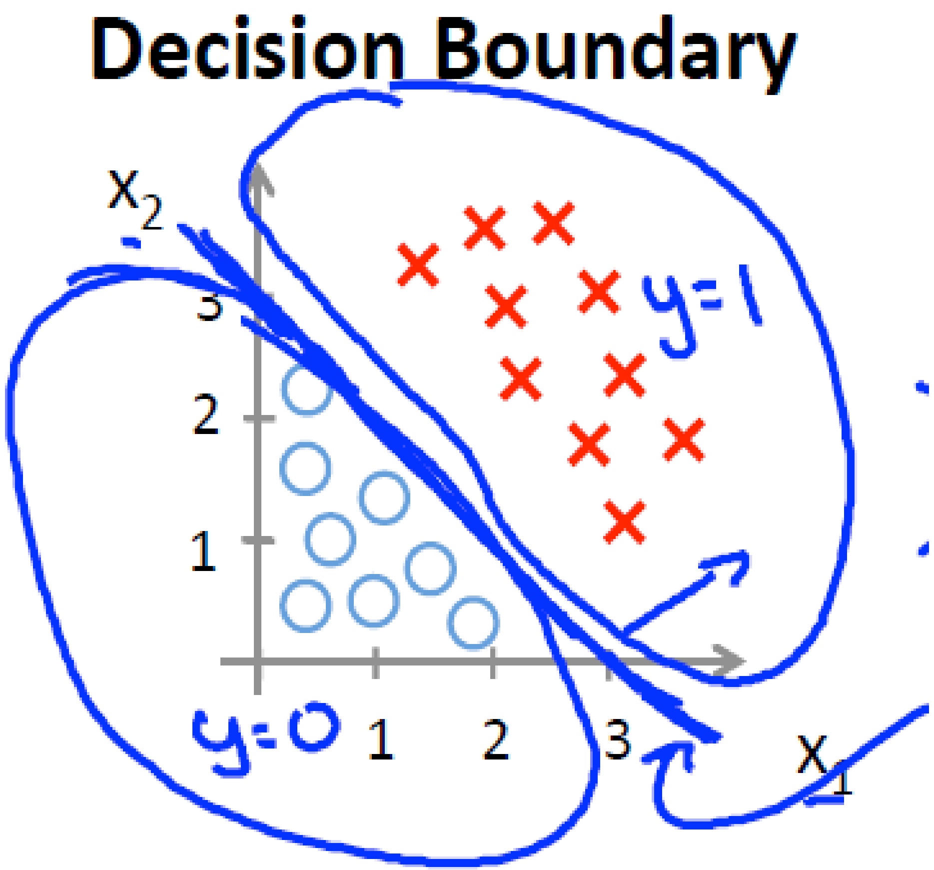
Logistic Regression – Decision Boundary

- If Θ^T is [-3 , 1, 1]

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

Predict "$y = 1$" if $-3 + x_1 + x_2 \geq 0$

Decision Boundary
$x_1 + x_2 >= 3$

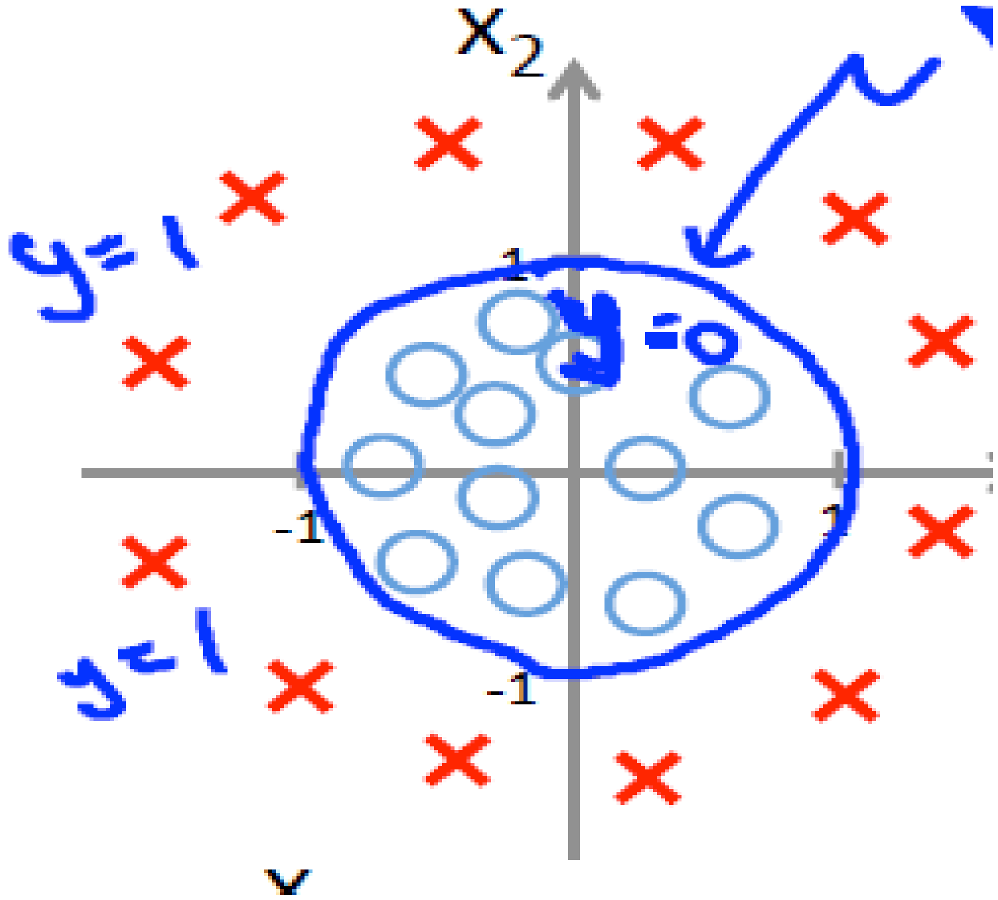


Logistic Regression – Non Linear Boundaries

$$h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 + \theta_4 x_2^2)$$

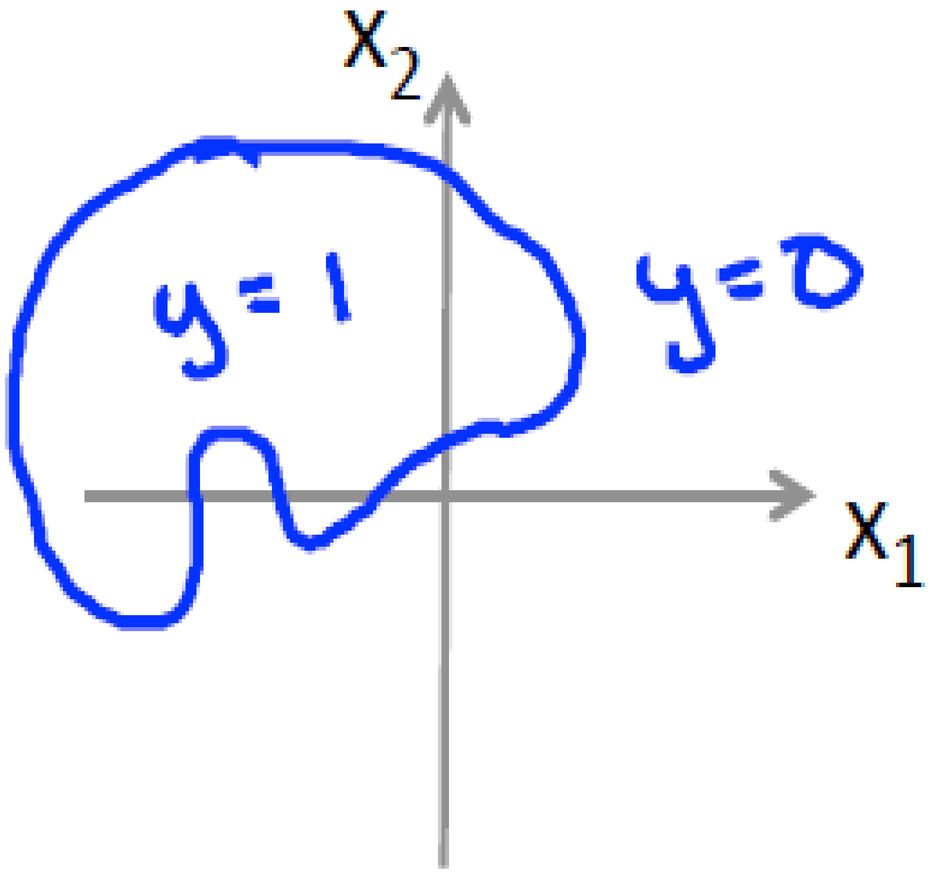
If Θ^T is [-1 , 0, 0, 1, 1]

Predict "$y = 1$" if $-1 + x_1^2 + x_2^2 \geq 0$



Logistic Regression - Non-linear boundaries

$$h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 \underline{x_1^2} + \theta_4 x_1^2 x_2 + \theta_5 x_1^2 x_2^2 + \theta_6 x_1^3 x_2 + \dots)$$



Training set: $\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \cdots, (x^{(m)}, y^{(m)})\}$

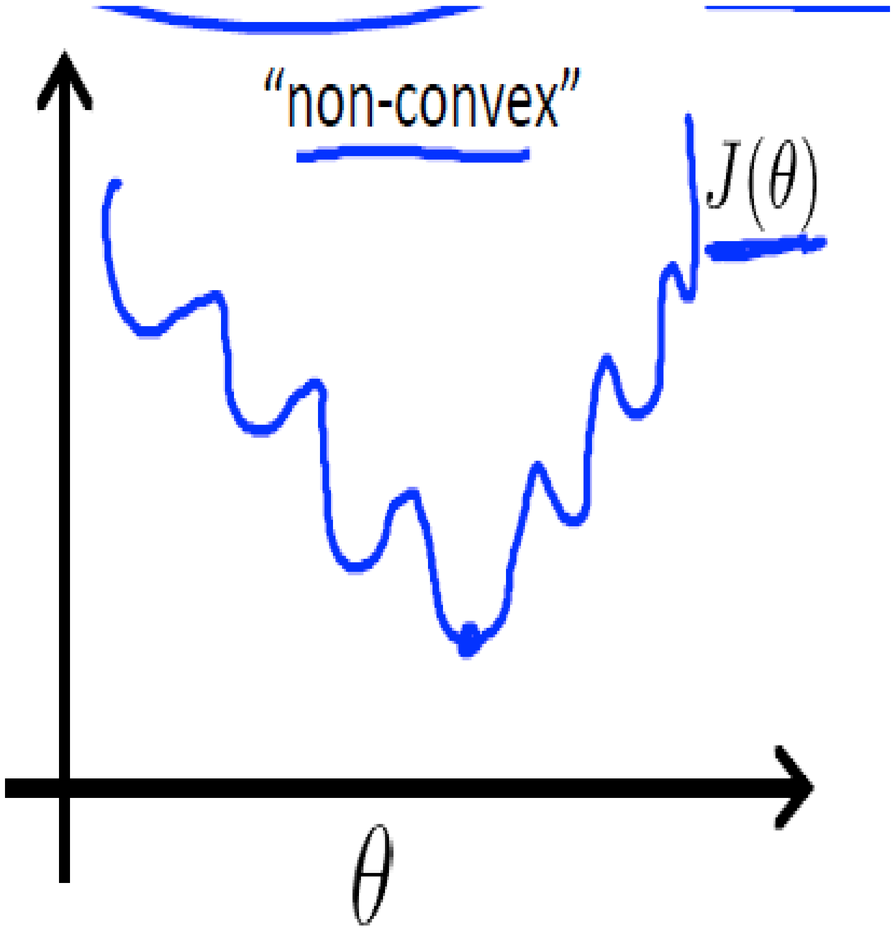
m examples

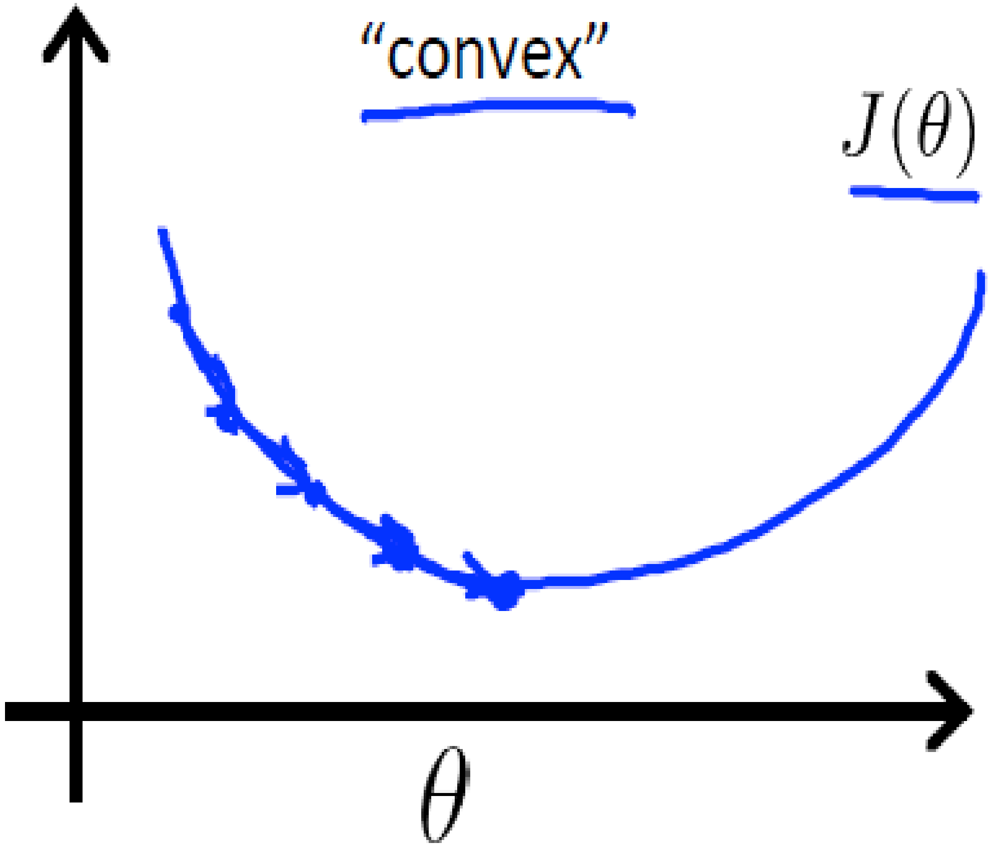
$$\underbrace{x \in \begin{bmatrix} x_0 \\ x_1 \\ \dots \\ x_n \end{bmatrix}}\ \mathbb{R}^{n+1} \qquad x_0 = 1, y \in \{0, 1\}$$

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

How to choose parameters θ ?

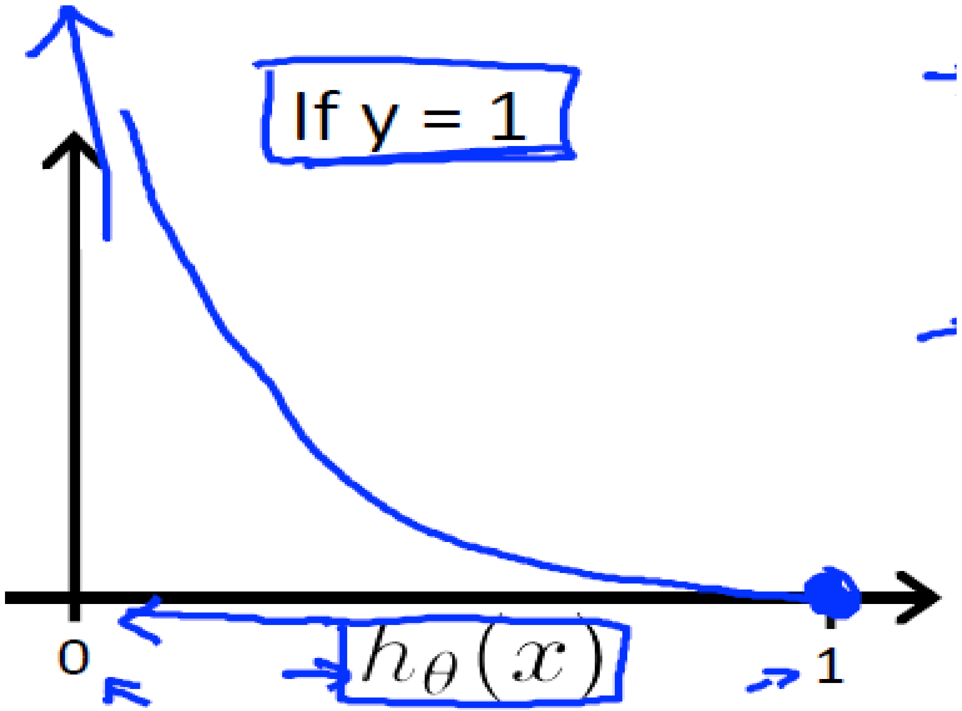
Logistic Regression – Cost function

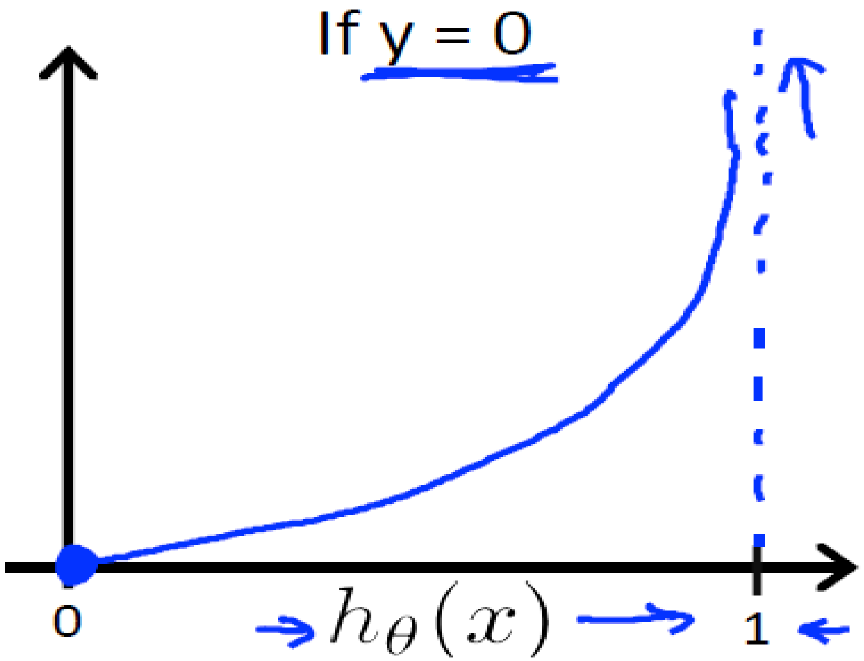




Logistic Regression – Cost function

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





Logistic Regression – Cost function

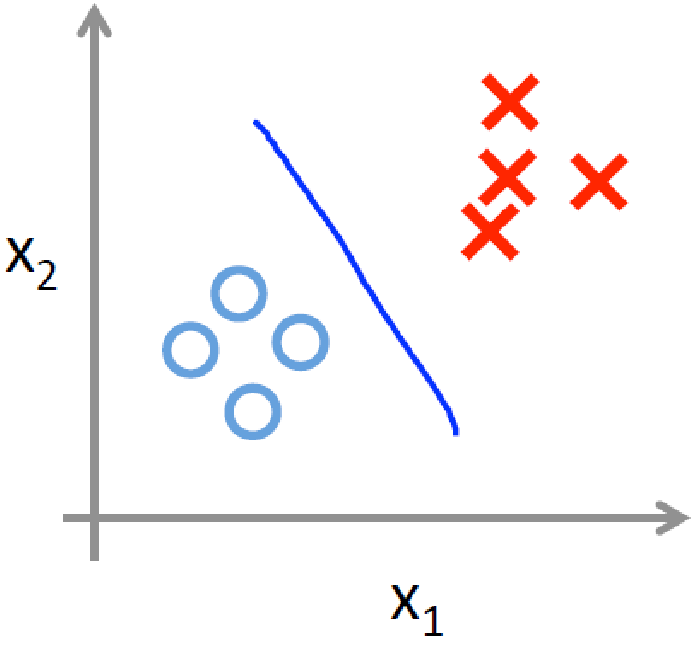
$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m} \mathrm{Cost}(h_\theta(x^{(i)}), y^{(i)})$$

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

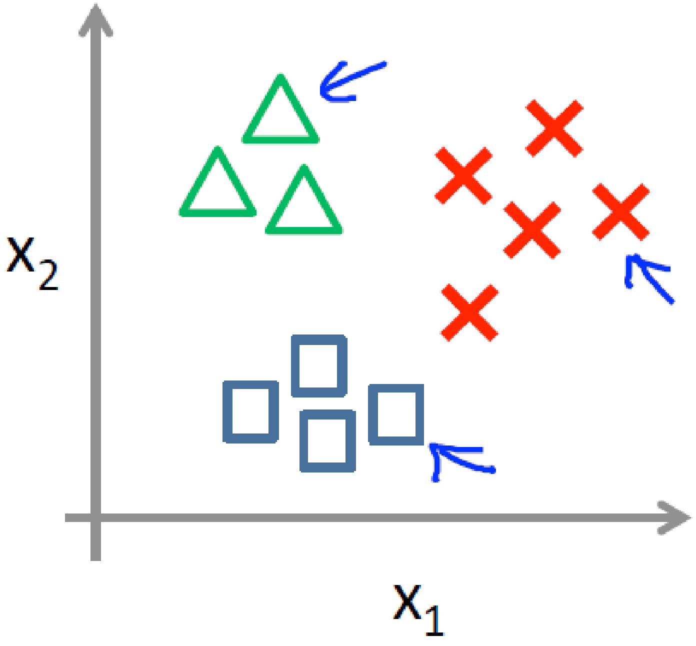
$$= -\frac{1}{m}\left[\sum_{i=1}^{m} y^{(i)} \log h_\theta(x^{(i)}) + (1 - y^{(i)}) \log (1 - h_\theta(x^{(i)}))\right]$$

Multiclass Classification

Binary classification:



Multi-class classification:



$$h_\theta^{(i)}(x) = P(y = i|x;\theta) \qquad (i = 1, 2, 3)$$

Multi class Classification

One-vs-all

Train a logistic regression classifier $h_\theta^{(i)}(x)$ for each class i to predict the probability that $y = i$.

On a new input x, to make a prediction, pick the class i that maximizes

$$\max_i h_\theta^{(i)}(x)$$

Logistic Regression - Prediction

X1	X2	Actual Y	Output (b0+b1*x 1 + b2* x2)	Predicted Y
2.7810	2.5505	0		
1.4654	2.3621	0		
3.3965	4.4002	0		
1.3880	1.8502	0		
3.0640	3.0053	0		
7.6275	2.7592	1		
5.3324	2.0886	1		
6.9225	1.7710	1		
8.6754	-0.2420	1		
7.6737	3.508	1		

- b0 = -0.4066054641
- b1 = 0.8525733164
- b2 = -1.104746259
- What is the
  - Accuracy?

Gradient Descent

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

Want $\min_\theta J(\theta)$:

Repeat $\{$

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

(simultaneously update all θ_j)

$\}$

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

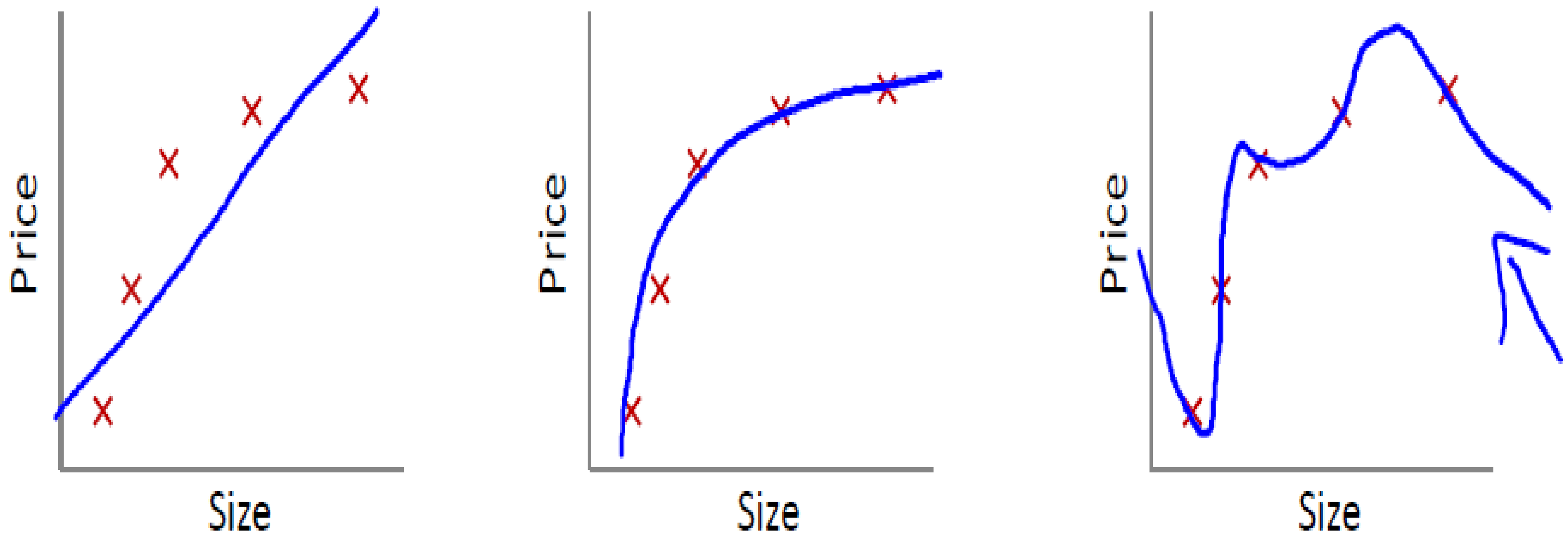
Logistic Regression by Stochastic Gradient Descent

- Given each training instance:
  - Calculate a prediction using the current values of the coefficients.
  - Calculate new coefficient values based on the error in the prediction.
- $\hat{y} = \frac{1}{1+e^{-(\theta 0 + \theta 1 * x1 + \theta 2 * x2)}}$
- $\theta = \theta + \alpha * (y - \hat{y}) * \hat{y} * (1 - \hat{y}) * x$
- If $\alpha = 0.3$
- $\theta_0 = -0.0375$
- $\theta_1 = -0.104290635$
- $\theta_2 = -0.09564513761$

X1	X2	Y
2.7810836	2.550537003	0
1.465489372	2.362125076	0
3.396561688	4.400293529	0
1.38807019	1.850220317	0
3.06407232	3.005305973	0
7.627531214	2.759262235	1
5.332441248	2.088626775	1
6.922596716	1.77106367	1
8.675418651	-0.2420686549	1
7.673756466	3.508563011	1

Over fitting in Linear Regression

Example: Linear regression (housing prices)

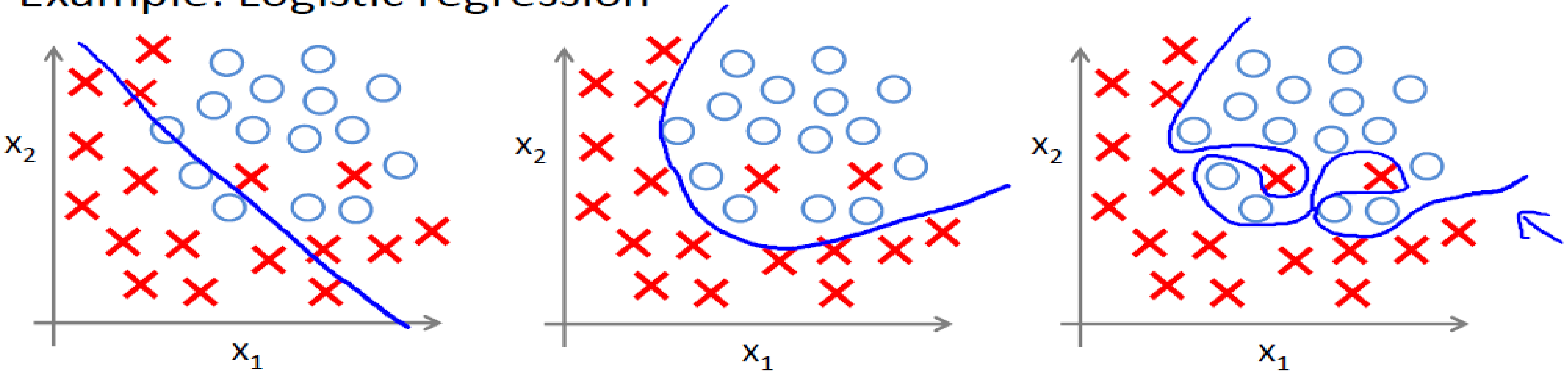


$\rightarrow \theta_0 + \theta_1 x$ $\rightarrow \theta_0 + \theta_1 x + \theta_2 x^2$ $\rightarrow \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$

Underfit – High Bias Just Right Overfit – High Variance

Over fitting in Logistic Regression

Example: Logistic regression



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

(g = sigmoid function)

$$g(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 + \theta_4 x_2^2 + \theta_5 x_1 x_2)$$

$$g(\theta_0 + \theta_1 x_1 + \theta_2 x_1^2 + \theta_3 x_1^2 x_2 + \theta_4 x_1^2 x_2^2 + \theta_5 x_1^2 x_2^3 + \theta_6 x_1^3 x_2 + \dots)$$

Underfit – High Bias

Just Right

Overfit – High Variance

Addressing Over fitting

Options

1. Reduce the number of features
   - Manually select which features to keep
   - Model Selection Algorithms
2. Regularization
   - Keep all features but reduce the magnitude/values of parameter ϴJ
   - Works well if many features all of which contribute a little to the predicting y

Regularization

- Linear Regression

$$J(\theta) = \frac{1}{2m}\left[\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2 + \lambda\sum_{j=1}^{n}\theta_j^2\right]$$

- Logistic Regression Cost Function

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

- Regularised Cost function

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m}\left[-y^{(i)}\log(h_\theta(x^{(i)})) - (1-y^{(i)})\log(1-h_\theta(x^{(i)}))\right] + \frac{\lambda}{2m}\sum_{j=1}^{n}\theta_j^2.$$

References

- Kevin P. Murphy, Machine Learning: A Probabilistic Perspective, MIT Press, 2012.
- Machine Learning by Andrew N G (Chapter 6)
  - https://www.youtube.com/watch?v=-la3q9d7AKQ