Machine Learning
DSECL ZG565

Dr. Monali Mavani

BITS Pilani
Pilani Campus

innovate achieve lead

Session Content

- Linear Regression – Geometric Approach
  - Gradient Descent Optimization
- Logistic regression
  - Gradient Descent Optimization

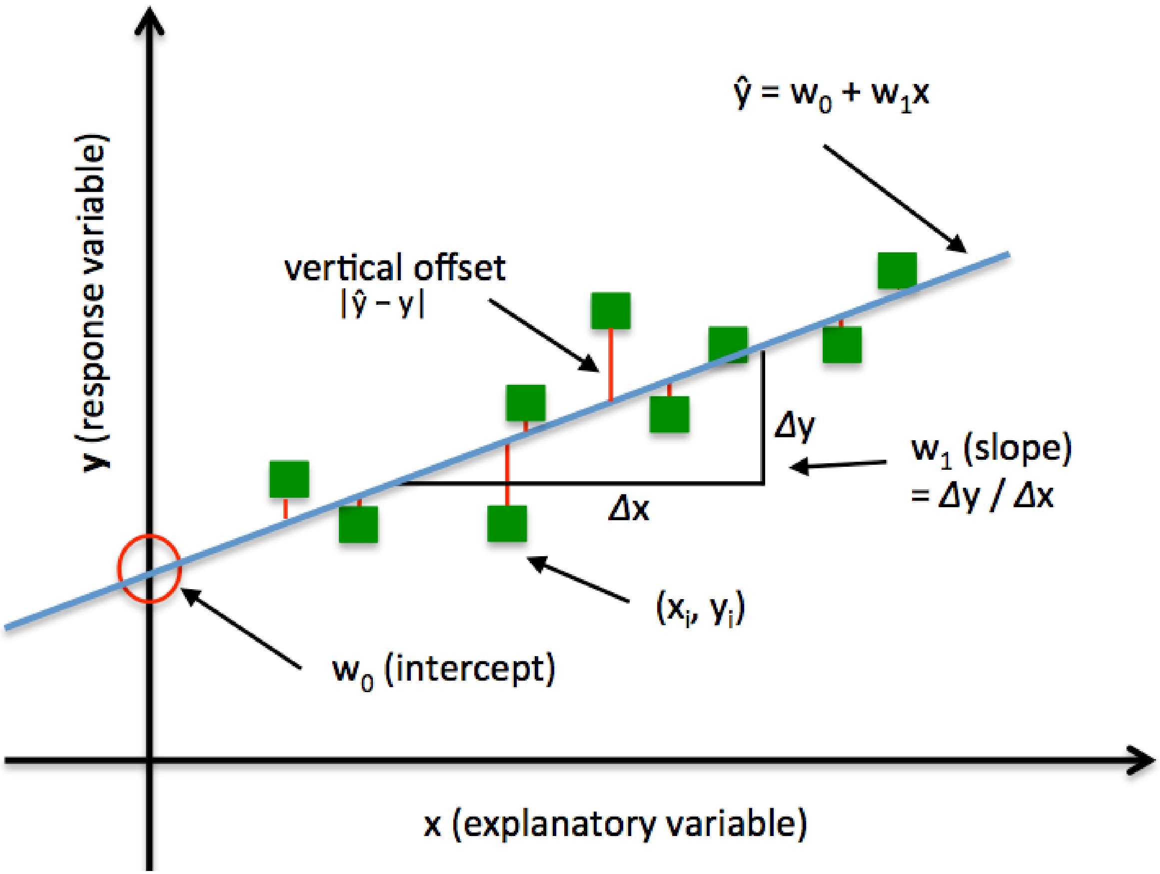
Linear Regression

Wish to learn f: X→ Y, where Y is real, given {<x^1,y^1>…<x^n,y^n>}

- **Geometric Approach**
  - **Least squares function fitting given {<x^1,y^1>…<x^n,y^n>}**

For each input observation $x^{(i)}$, a vector of features [x_1, x_2, …,x_n].
j^{th} feature of i^{th} training example: $x_j^{(i)}$

Geometric Approach



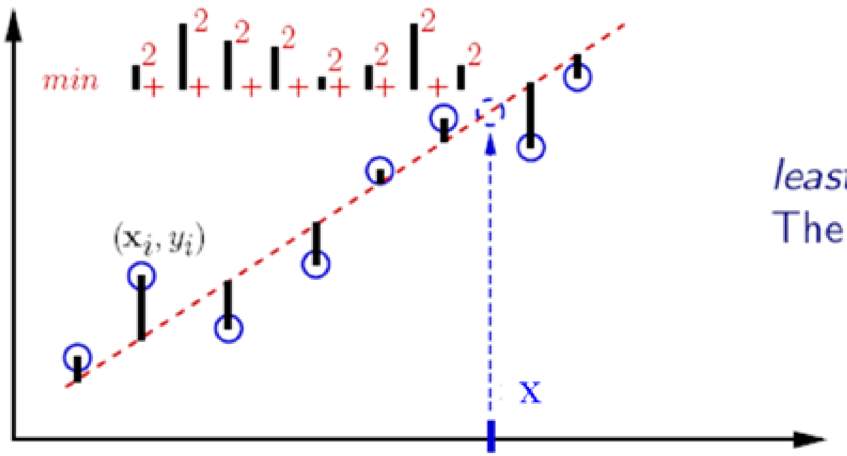
Given:

– Data $\boldsymbol{X} = \left\{ \boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(n)} \right\}$ where $\boldsymbol{x}^{(i)} \in \mathbb{R}^d$

– Corresponding labels $\boldsymbol{y} = \left\{ y^{(1)}, \ldots, y^{(n)} \right\}$ where $y^{(i)} \in \mathbb{R}$

Learning a *Line* via Minimizing MSE

- Fit model by minimizing sum of squared errors



least squares (LSQ)
The fitted line is used as a predictor

- **How to learn a line of equation y' = mx + b given a *labelled* dataset? : By minimizing *mean squared error***

$$\text{minimize} \frac{1}{2n} \sum_{i=1}^{n} \left(y'^{(i)} - y^{(i)}\right)^2 \quad \equiv \quad \underset{m,b}{\text{minimize}} \frac{1}{2n} \sum_{i=1}^{n} \left((mx+b)^{(i)} - y^{(i)}\right)^2$$

Linear Regression – Hypothesis function

- Hypothesis:

$$y = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \ldots + \theta_d x_d = \sum_{j=0}^{d} \theta_j x_j$$

Assume $x_0 = 1$

$$h(x) = \sum_{i=0}^{n} \theta_i x_i = \theta^T x,$$

Learning a *Linear Regression Model* via Minimizing a *Cost Function*

- $\boldsymbol{h_\theta(x) = \theta_0 + \theta_1 x}$
- $\boldsymbol{\theta}_0$ = b, $\boldsymbol{\theta}_1$= m, $\boldsymbol{h_\theta(x)}$= y', y' = mx + b
  - minimizing ***mean squared error***. That is:

$$\text{minimize}\frac{1}{2n}\sum_{i=1}^{n}\left(y'^{(i)} - y^{(i)}\right)^2$$

$$\equiv$$

$$\underset{\theta_0,\theta_1}{\text{minimize}}\boxed{\frac{1}{2n}\sum_{i=1}^{n}\left((\theta_0+\theta_1 x)^{(i)} - y^{(i)}\right)^2}$$

$$\equiv$$

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

Known as ***cost function*** $\boldsymbol{J(\theta_0,\theta_1)}$

This problem is referred to as an ***optimization problem*** with the objective of: $\underset{\theta_0,\theta_1}{\text{minimizing}}\,\boldsymbol{J(\theta_0,\theta_1)}$

Gradient Descent and cost function

Microsoft
werPoint Presentat

Logistic Regression

Linear Models for Classification

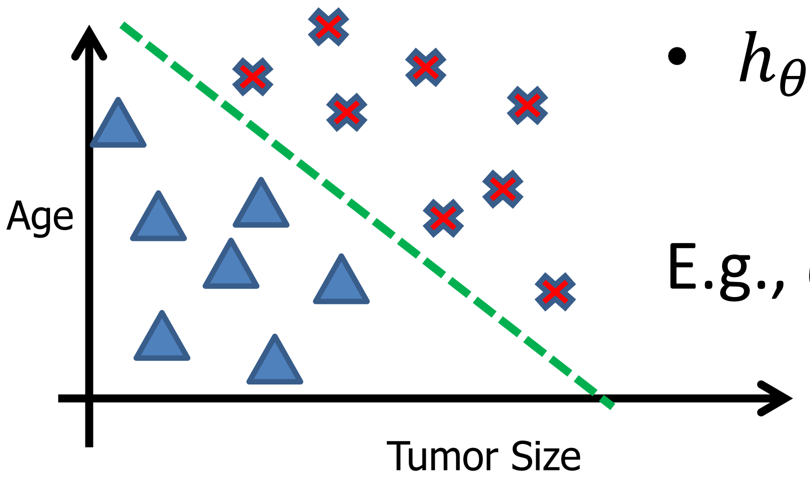
- In the most common scenario
  - the classes are taken to be disjoint
  - so that each input is assigned to one and only one class
- The input space is thereby divided into *decision regions* whose boundaries are called *decision boundaries* or *decision surfaces*
- We consider linear models for classification
  - **by which we mean that the decision surfaces are linear functions of the input vector x**

Two classes classification

- In two class classification, an input vector **x** is assigned to class C_1 if y(**x**) >= 0 and to class C_2 otherwise

- The corresponding decision boundary is therefore defined by the relation y(**x**) = 0

Decision boundary

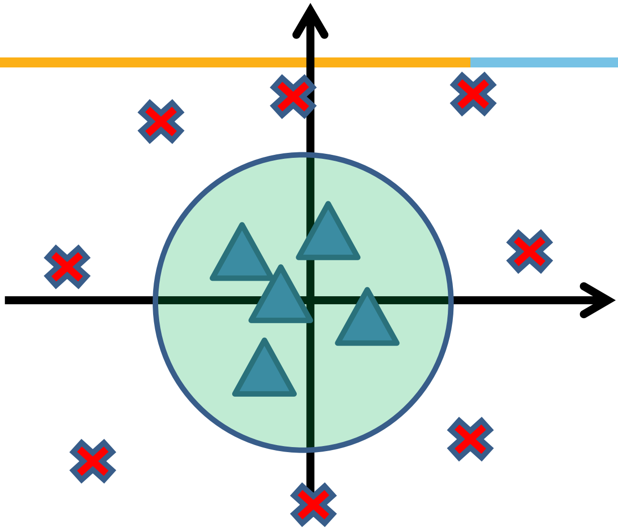


- $h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2)$

E.g., $\theta_0 = -3,\ \theta_1 = 1,\ \theta_2 = 1$

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

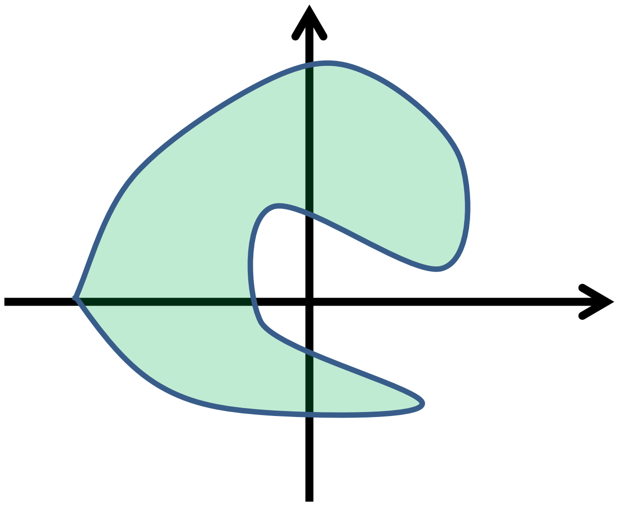
Slide credit: Andrew Ng



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

E.g., $\theta_0 = -1, \theta_1 = 0, \theta_2 = 0, \theta_3 = 1, \theta_4 = 1$

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



- $h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 + \theta_4 x_1^2 x_2 + \theta_5 x_1^2 x_2^2 + \theta_6 x_1^3 x_2 + \cdots)$

Slide credit: Andrew Ng

Logistic Regression vs Naïve Bayes

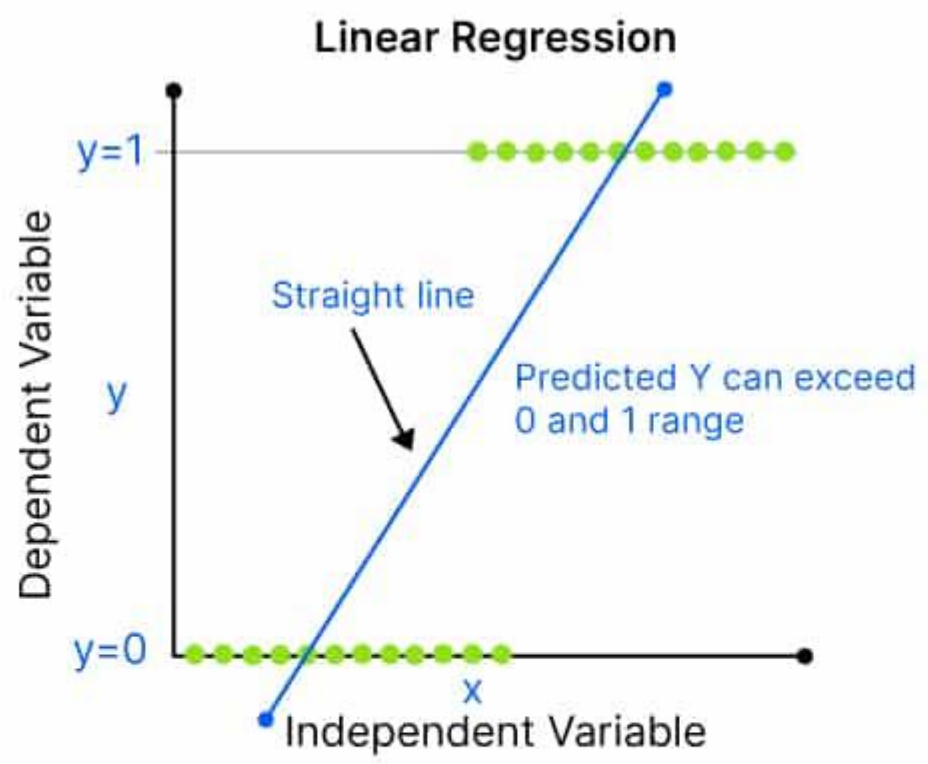
Idea:

- Naïve Bayes allows computing P(Y|X) by learning P(Y) and P(X|Y)
- Why not learn P(Y|X) directly?

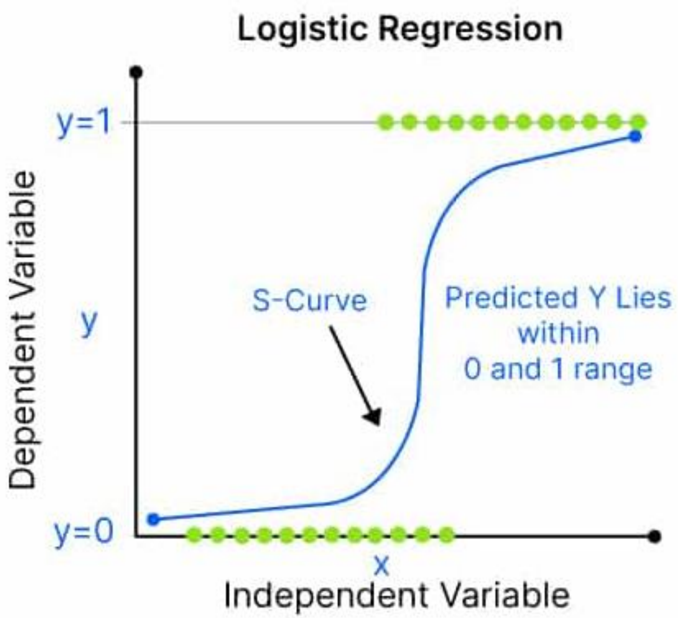
Linear Regression versus logistic regression

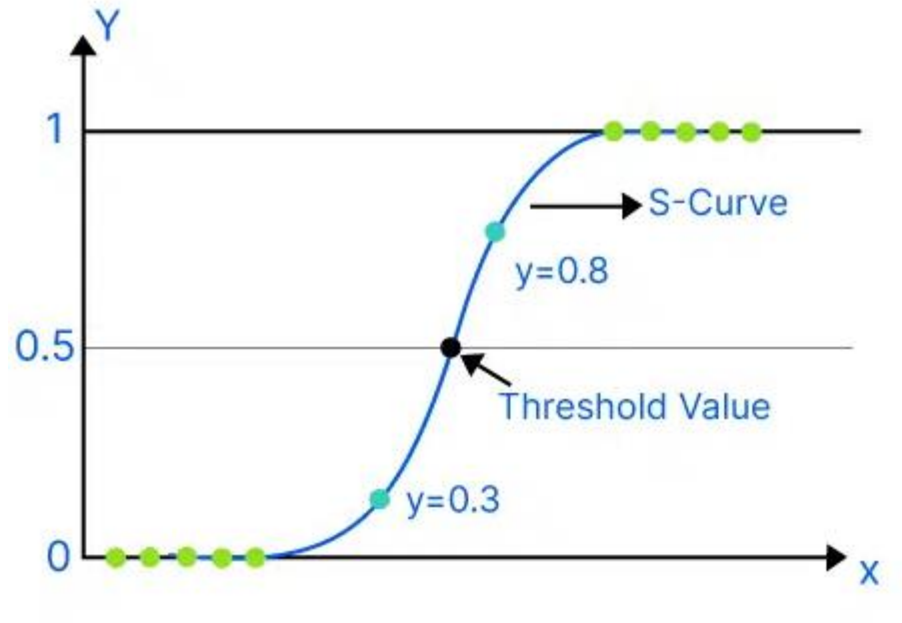
- **Linear Regression** could help us predict the student's test score on a scale of 0 - 100. Linear regression predictions are continuous (numbers in a range).
- **Logistic Regression** could help use predict whether the student passed or failed. Logistic regression predictions are discrete (only specific values or categories are allowed). We can also view probability scores underlying the model's classifications.

Linear Regression



Logistic Regression





Components

1. A feature representation of the input.
   - For each input observation $x^{(i)}$, a vector of features $[x_1, x_2, \ldots, x_n]$.
   - j^{th} feature of i^{th} training example: $x_j^{(i)}$
2. A classification/hypothesis function that computes y-hat, the estimated class, via p(y|x).
3. An objective function for learning, usually involving minimizing error on training examples : cross-entropy loss function
4. An algorithm for optimizing the objective function: stochastic gradient descent algorithm.

Sigmoid/Logistic Function

- Sigmoid/logistic function takes a real value as input and outputs another value between 0 and 1
- That framework is called logistic regression
  - Logistic: A special mathematical sigmoid function it uses
  - Regression: Combines a weight vector with observations to create an answer

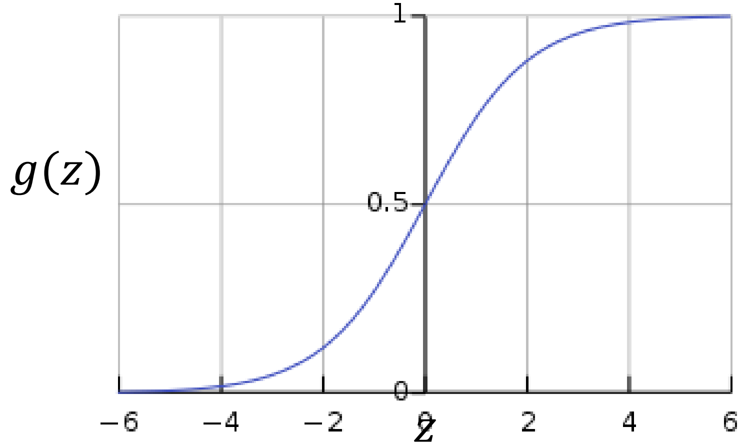
$$h_\theta(x) = g(\theta^T x)$$

Logistic regression Hypothesis representation

- ‘sigmoid’ means S-shaped.
  - A ‘squashing function’ because it maps the whole real axis into a finite interval.
- Want $0 \le h_\theta(x) \le 1$
- $h_\theta(x) = g(\theta^\top x)$,

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

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

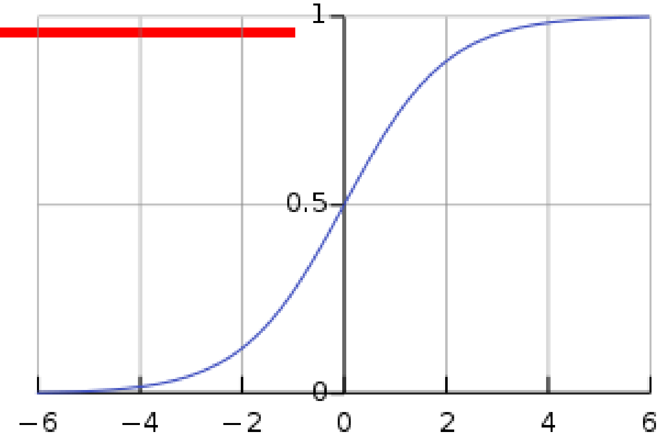


Interpretation of hypothesis output

$h_\theta(x) = g(\theta^\top x)$ $\quad z = \theta^\top x$

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

$g(z)$



$h_\theta(x) =$ estimated probability that $y = 1$ on input x

$h_\theta(x)$ =P(y=1| $x; \theta$) = 1- P(y=0| $x; \theta$)

Suppose predict "y = 1" if $h_\theta(x) \geq 0.5$

$z = \theta^\top x \geq 0$

predict "y = 0" if $h_\theta(x) < 0.5$

$\boldsymbol{z = \theta^\top x < 0}$

- σ(z) is nonlinear, however, the decision boundary is determined by $\theta^T x = 0$ which is a linear function in x

Slide credit: Andrew Ng

Interpretation of hypothesis output

- $h_\theta(x)$ = estimated probability that $y = 1$ on input x

- Example: If $x = \begin{bmatrix} x_0 \\ \mathrm{x}_1 \end{bmatrix} = \begin{bmatrix} 1 \\ \mathrm{tumorSize} \end{bmatrix}$
- $h_\theta(x) = 0.7$

- Tell patient that 70% chance of tumor being malignant

Slide credit: Andrew Ng

Learning model parameters

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

m examples $x \in \begin{bmatrix} x_0 \\ x_1 \\ \dots \\ x_n \end{bmatrix}$ $x_0 = 1, y \in \{0, 1\}$

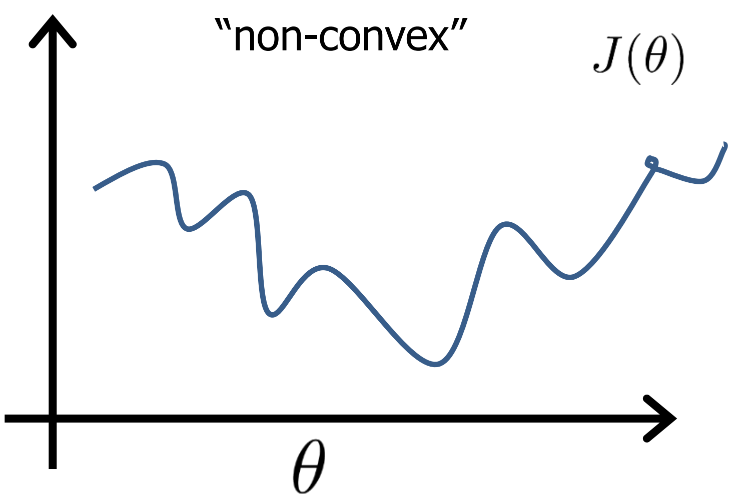
$$h_\theta(x) = \frac{1}{1 + e^{-\theta^T x}}$$

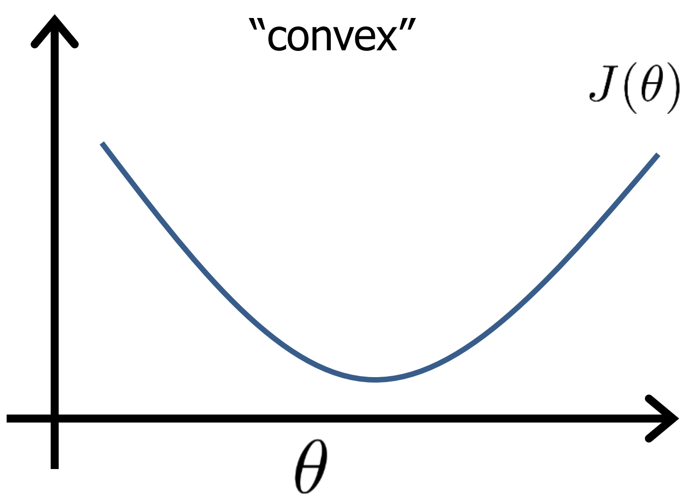
How to choose parameters (feature weights) θ ?

MSE Cost Function

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

$$\mathrm{Cost}(h_\theta(x^{(i)}), y^{(i)}) = \frac{1}{2} \left(h_\theta(x^{(i)}) - y^{(i)} \right)^2$$



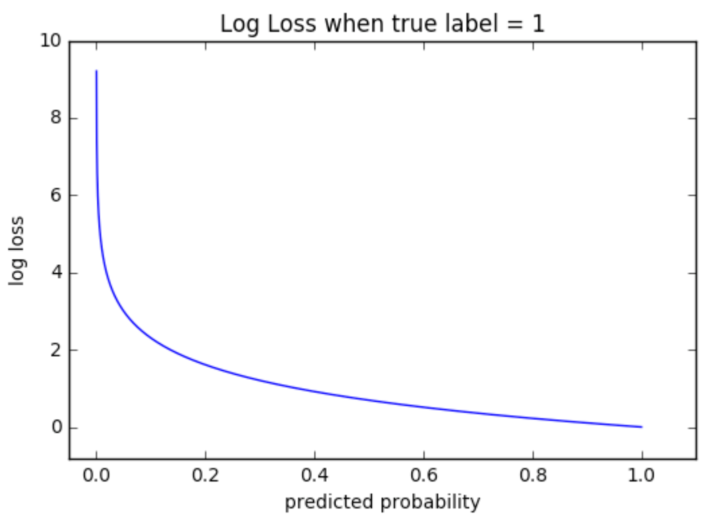


Error (Cost) Function

- Our prediction function is non-linear (due to sigmoid transform)
- Squaring this prediction as we do in MSE results in a non-convex function with many local minima.
- If our cost function has many local minimums, gradient descent may not find the optimal global minimum.
- So instead of Mean Squared Error, we use a error/ cost function called Cross-Entropy, also known as Log Loss.

Cross Entropy

- Uses conditional maximum likelihood estimation. Prefers the correct class labels of the training examples to be more likely.
- we choose the parameters w and b that maximize the log probability of the true y labels known as negative log likelihood loss or cross-entropy loss
- Cross-entropy loss increases as the predicted probability diverges from the actual label.
- So predicting a probability of .012 when the actual observation label is 1 would be bad and result in a high loss value.
- A perfect model would have a log loss of 0.

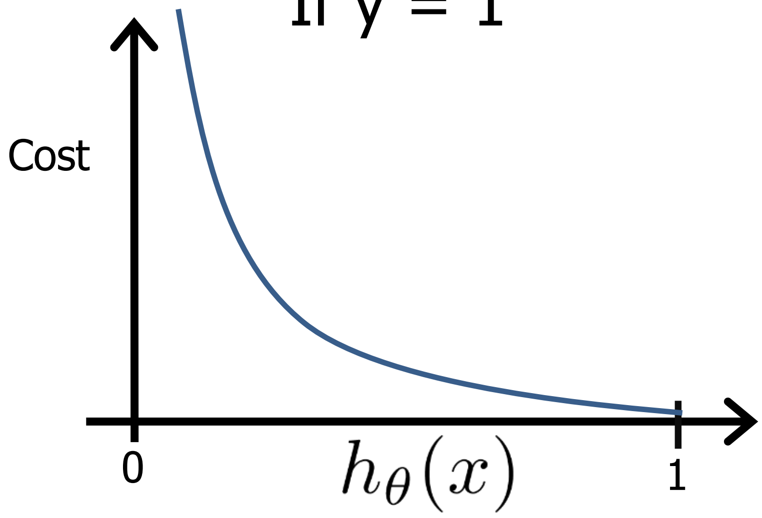


Logistic regression cost function (cross entropy)

Cross-entropy loss can be divided into two separate cost functions: one for y=1 and one for y=0

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

If y = 1



$\mathrm{Cost} = 0 \text{ if } y = 1, h_\theta(x) = 1$

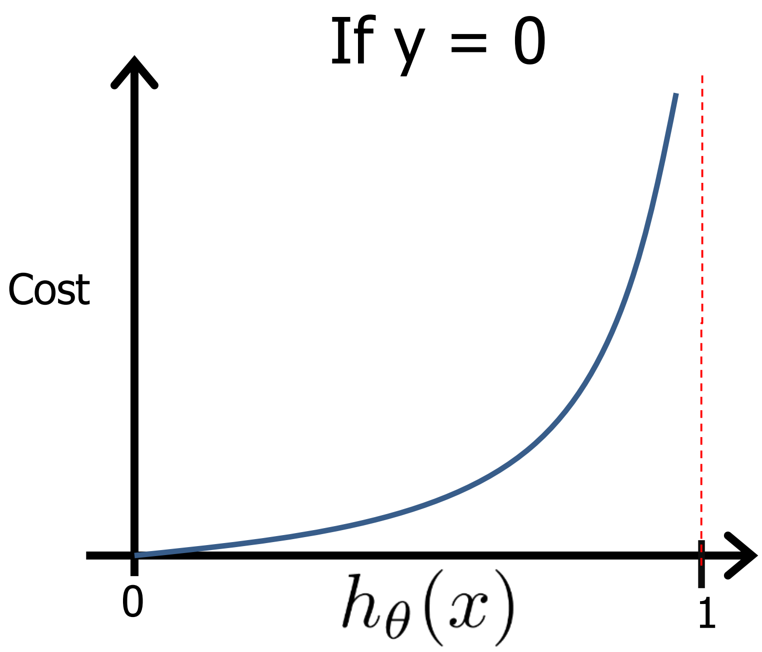
But as $h_\theta(x) \rightarrow 0$

$Cost \rightarrow \infty$

Captures intuition that if $h_\theta(x) = 0$, (predict $P(y = 1|x; \theta) = 0$), but $y = 1$, we'll penalize learning algorithm by a very large cost.

Logistic regression cost function

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



Cost=0; If y=0 and $h_\theta(x)=0$

Logistic regression cost function

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

$$\Downarrow$$

- $\mathrm{Cost}(h_\theta(x), y) = -y\ \log\big(\mathrm{h}_\theta(\mathrm{x})\big) - (1 - \mathrm{y})\log\big(1 - h_\theta(x)\big)$

- If $y = 1$: $\mathrm{Cost}(h_\theta(x), y) = -\log\big(h_\theta(x)\big)$
- If $y = 0$: $\mathrm{Cost}(h_\theta(x), y) = -\log\big(1 - h_\theta(x)\big)$

Cost function for m training examples

$$\mathbf{Cost}(\boldsymbol{h_\theta}(\boldsymbol{x}), \boldsymbol{y}) = -\boldsymbol{y}\ \mathbf{log}(\boldsymbol{h_\theta}(\boldsymbol{x})) - (\mathbf{1} - \mathbf{y})\ \mathbf{log}(\mathbf{1} - \boldsymbol{h_\theta}(\boldsymbol{x}))$$

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

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

To fit parameters θ : Apply Gradient Descent Algorithm

$$\min_\theta J(\theta)$$

To make a prediction given new x:

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

Gradient descent

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

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

Good news: Convex function!
Bad news: No analytical solution

Repeat {

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

}

(Simultaneously update all θ_j)

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

Slide credit: Andrew Ng

Gradient descent for Linear Regression

Repeat {

$$\theta_j := \theta_j - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta\left(x^{(i)}\right) - y^{(i)} \right) x_j^{(i)}$$

}

$$\boxed{h_\theta(x) = \theta^\top x}$$

Gradient descent for Logistic Regression

Repeat {

$$\theta_j := \theta_j - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta\left(x^{(i)}\right) - y^{(i)} \right) x_j^{(i)}$$

}

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

Slide credit: Andrew Ng

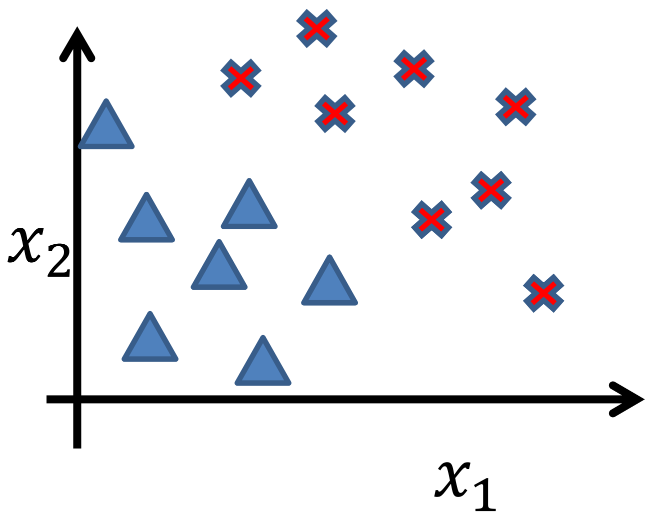
Multi-class classification

- Email foldering/tagging: Work, Friends, Family, Hobby
- Medical diagrams: Not ill, Cold, Flu
- Weather: Sunny, Cloudy, Rain, Snow

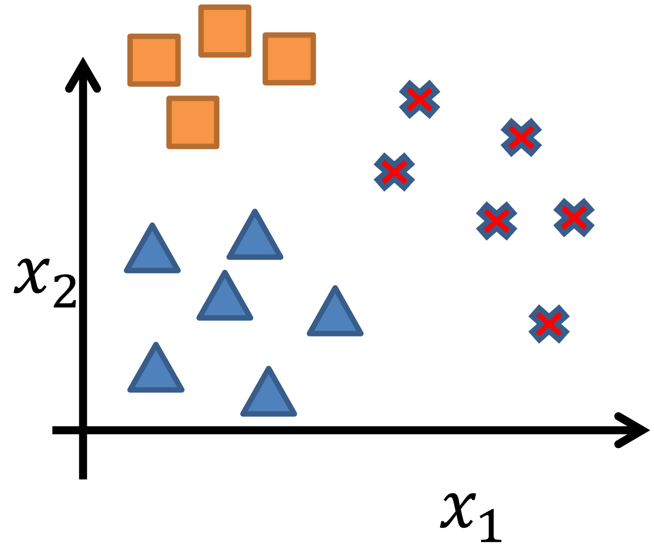
Slide credit: Andrew Ng

Multi-class classification

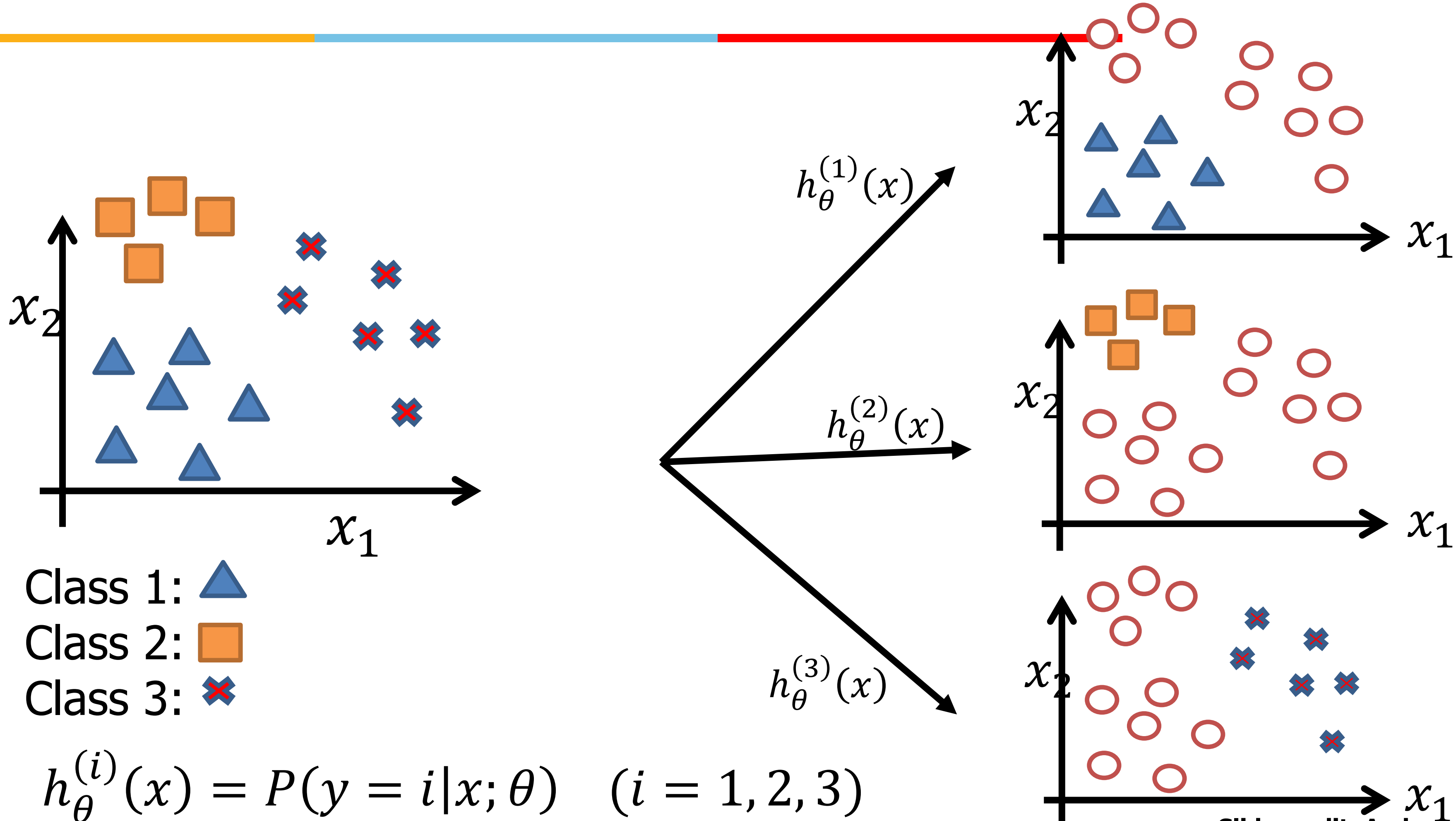
Binary classification



Multiclass classification



One-vs-all (one-vs-rest)



Class 1:
Class 2:
Class 3:

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

Slide credit: Andrew Ng

One-vs-all

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

- Given a new input x, pick the class i that maximizes

$$\max_{\mathrm{i}} h_\theta^{(i)}(x)$$

Slide credit: Andrew Ng

Logistic regression (Classification)

- **Model**

$$h_\theta(x) = P(Y = 1|X_1, X_2, \cdots, X_n) = \frac{1}{1+e^{-\theta^\top x}}$$

- **Cost function**

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

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

- **Learning**

Gradient descent: Repeat $\{\theta_j := \theta_j - \alpha \frac{1}{m}\sum_{i=1}^{m}\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right) x_j^{(i)}\}$

- **Inference**

$$\hat{Y} = h_\theta(x^{\text{test}}) = \frac{1}{1 + e^{-\theta^\top x^{\text{test}}}}$$

Logistic Regression and Gaussian Naïve Bayes Classifier

- Interestingly, the parametric form of P(Y|X) used by Logistic Regression is precisely the form implied by the assumptions of a Gaussian Naive Bayes classifier.

- Therefore, we can view Logistic Regression as a closely related alternative to GNB, though the two can produce different results in many cases

Parameter estimation of generic logistic regression

- Logistic Regression holds in many problem settings beyond the GNB problem
- General method required for estimating it in a more broad range of cases.
- In many cases we may suspect the GNB assumptions are not perfectly satisfied.
- We may wish to estimate the wi parameters directly from the data

References

- Tom M. Mitchell
  Generative and discriminative classifiers: Naïve Bayes and Logistic Regression
  http://www.cs.cmu.edu/~tom/mlbook/NBayesLogReg.pdf
- Andrew Ng, Michael Jordan
  On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes
  http://papers.nips.cc/paper/2020-on-discriminative-vs-generative-classifiers-a-comparison-of-logistic-regression-and-naive-bayes.pdf
- https://see.stanford.edu/materials/aimlcs229/cs229-notes1.pdf

References

- http://www.cs.cmu.edu/~tom/NewChapters.html
- http://ai.stanford.edu/~ang/papers/nips01-discriminativegenerative.pdf
- https://medium.com/@sangha_deb/naive-bayes-vs-logistic-regression-a319b07a5d4c
- https://www.youtube.com/watch?v=-la3q9d7AKQ
- http://www.datasciencesmachinelearning.com/2018/11/handling-outliers-in-python.html

l

Self Learning

Derivative of cost function for logistic regression

https://medium.com/analytics-vidhya/derivative-of-log-loss-function-for-logistic-regression-9b832f025c2d

Logistic regression sigmoid function - derivatives

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

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

$$\frac{d\big(\sigma(\mathrm{x})\big)}{dx} = \frac{0 * (1 + e^{-\mathrm{x}}) - (1) * \big(e^{-\mathrm{x}} * (-1)\big)}{(1 + e^{-\mathrm{x}})^2}$$

$$\frac{d\big(\sigma(\mathrm{x})\big)}{dx} = \frac{(e^{-\mathrm{x}})}{(1 + e^{-\mathrm{x}})^2} = \frac{1\text{-}1 + (e^{-\mathrm{x}})}{(1 + e^{-\mathrm{x}})^2} = \frac{1 + e^{-\mathrm{x}}}{(1 + e^{-\mathrm{x}})^2} - \frac{1}{(1 + e^{-\mathrm{x}})^2}$$

$$\frac{d\big(\sigma(\mathrm{x})\big)}{dx} = \frac{1}{1 + e^{-\mathrm{x}}} * \left(1 - \frac{1}{1 + e^{-\mathrm{x}}}\right) = \sigma(\mathrm{x})\big(1 - \sigma(\mathrm{x})\big)$$

Logistic regression cost function

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

- $\text{Cost}(h_\theta(x), y) = -y\ \log\big(\mathrm{h}_\theta(\mathrm{x})\big) - (1 - \mathrm{y})\log\big(1 - h_\theta(x)\big)$

- If $y = 1$: $\text{Cost}(h_\theta(x), y) = -\log\big(h_\theta(x)\big)$
- If $y = 0$: $\text{Cost}(h_\theta(x), y) = -\log\big(1 - h_\theta(x)\big)$

Step –I

Applying Chain rule and writing in terms of partial derivatives

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

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

$$\frac{\partial(J(\theta))}{\partial(\theta j)} = -\frac{1}{m} * \left(\sum_{i=1}^{m} \left[y^{(i)} * \frac{1}{h_\theta(x^{(i)})} * \sigma(z)\left(1 - \sigma(z)\right) * \frac{\partial(\theta^T x)}{\partial(\theta j)} \right]\right.$$

$$\left. + \sum_{i=1}^{m} \left[\left(1 - y^{(i)}\right) * \frac{1}{\left(1 - h_\theta(x^{(i)})\right)} * (-\sigma(z)\left(1 - \sigma(z)\right) * \frac{\partial(\theta^T x)}{\partial(\theta j)} \right] \right)$$

Step –II

- Evaluating the partial derivative using the pattern of the derivative of the sigmoid function.

$$\frac{\partial(J(\theta))}{\partial(\theta j)} = -\frac{1}{m} * \left(\sum_{i=1}^{m}\left[\mathrm{y}^{(\mathrm{i})} * \frac{1}{\mathrm{h}_\theta(\mathrm{x}^{(\mathrm{i})})} * \sigma(\mathrm{z})\left(1-\sigma(\mathrm{z})\right) * \frac{\partial(\theta^{\mathrm{T}}\mathrm{x})}{\partial(\theta \mathrm{j})}\right]\right.$$

$$\left. + \sum_{i=1}^{m}\left[\left(1-y^{(i)}\right) * \frac{1}{\left(1-h_\theta(x^{(i)})\right)} * \left(-\sigma(\mathrm{z})\left(1-\sigma(\mathrm{z})\right) * \frac{\partial(\theta^{\mathrm{T}}x)}{\partial(\theta j)}\right]\right)$$

$$\frac{\partial(J(\theta))}{\partial(\theta j)} = -\frac{1}{m} * \left(\sum_{i=1}^{m}\left[\mathrm{y}^{(\mathrm{i})}\frac{1}{\mathrm{h}_\theta(\mathrm{x}^{(\mathrm{i})})}\mathrm{h}_\theta(\mathrm{x}^{(\mathrm{i})})\left(1-h_\theta(x^{(i)})\right) * \mathrm{x}_\mathrm{j}^\mathrm{i}\right] + \right.$$

$$\left.\sum_{i=1}^{m}\left[\left(1-y^{(i)}\right) * \frac{1}{\left(1-h_\theta(x^{(i)})\right)} * \left(-h_\theta(x^{(i)})\left(1-h_\theta(x^{(i)})\right) * \mathrm{x}_\mathrm{j}^\mathrm{i}\right]\right)$$

Step –III

- Simplifying the terms by multiplication

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

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

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

How does logistic regression handle missing values?

- Replace missing values with column averages (i.e. replace missing values in feature 1 with the average for feature 1).
- Replace missing values with column medians.
- Impute missing values using the other features.
- Remove records that are missing features.
- Use a machine learning technique that uses classification trees, e.g. Decision tree

Class Imbalance Problem

- Find needle in haystack
- Lots of classification problems where the classes are skewed (more records from one class than another)
  - Credit card fraud
  - Intrusion detection
  - Defective products in manufacturing assembly line

Approaches to solve Class imbalance problem

- Up-sample minority class
  - randomly duplicating observations from a minority class
- Down-sample majority class
  - removing random observations.
- Generate Synthetic Samples
  - new samples based on the distances between the point and its nearest neighbors
- Change the performance metric
  - Use Recall, Precision or ROC curves instead of accuracy
- Try different algorithms
  - Some algorithms as Support Vector Machines and Tree-Based algorithms are better to work with imbalanced classes.