

# An Introduction to Machine Learning

Spencer Hanson, Samuel Reed

HackCU III, University Of Colorado Boulder

April 2017

# What is Classification?

- ▶ Email: Spam/Not Spam
- ▶ Sports Outcomes: Win/Loss
- ▶ Disease Prediction: Positive/Negative test results



# How do we Classify?

- ▶ Support Vector Machines (SVM)
- ▶ Neural Network
- ▶ **Logistic Regression**



Figure 1: Logistic Regression

# Introduction to Logistic Regression

- ▶ Hypothesis representation function,  $h_{\theta}(x)$
- ▶ Basic Binary Classification, 0 or 1. This is the output of the hypothesis function ( $y$ )
- ▶ Can be whether a trait occurs or not, absence or presence of what you want to predict
- ▶ Threshold value for deciding when to be positive or negative

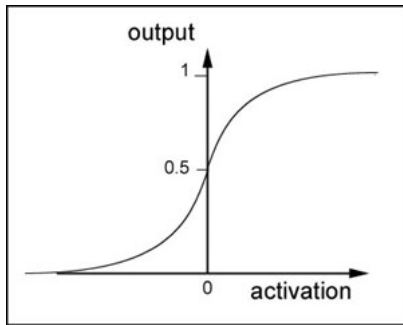


Figure 2: Sigmoid Activation Function

# Features and Data

- ▶ Features are specific traits about a given subject in the data.
- ▶ Examples being: size, color, shape, weight, etc..
- ▶ Subject is the item in question
- ▶ Example: A golf ball is 42.67 cm, white, spherical, and weighs 1.62 oz
- ▶ Two types features: Qualitative and Quantitative.
- ▶ Two types of data, training and testing

# Hypothesis Function

- ▶ Want  $0 \leq h_{\theta}(x) \leq 1$
- ▶ Threshold for classification: 0.5 (This is up to you and may change for different data sets)
- ▶ If  $h_{\theta}(x) \geq 0.5$ , predict "y=1"
- ▶ If  $h_{\theta}(x) < 0.5$ , predict "y=0"
- ▶  $h_{\theta}(x) = g(\theta^T x)$  where  $\theta$  is the matrix representing our coefficients, and  $x$  is the values of the features of our data.
- ▶  $h_{\theta}(x) = g(\theta_0 + \theta_1 x_1 + \dots + \theta_n x_n)$
- ▶  $g(z) = \frac{1}{1+e^{-z}}$

# Decision Boundary

- ▶  $h_{\theta}(x) = g(\theta_0 + \theta_1 x_1 + \dots + \theta_n x_n)$
- ▶ Predict "y=1" if  $\theta_0 + x_1 + x_2 + \dots + x_n \geq 0$
- ▶ Otherwise predict "y=0"

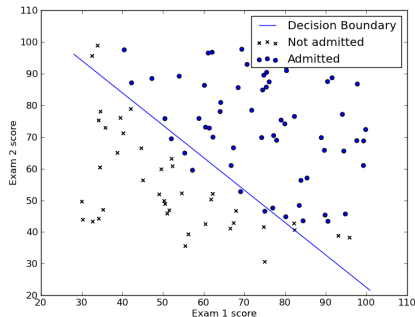


Figure 3: Decision Boundary Example

# Cost Function

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

- ▶ The cost function "punishes" the algorithm for incorrect predictions with very high costs (approaching infinity) and "rewards" the algorithm for correct predictions with low costs (approaching 0)
- ▶ If "y=1" and our predicted value from  $h_{\theta}(x)$  is "y=1" then our cost goes to 0, because the prediction was accurate
- ▶ If "y=1" and our predicted value from  $h_{\theta}(x)$  is "y=0" then our cost goes to infinity to teach the algorithm that this was an incorrect prediction
- ▶ Eventually we want our algorithm to only generate low costs



## Cost Function (Cont.)

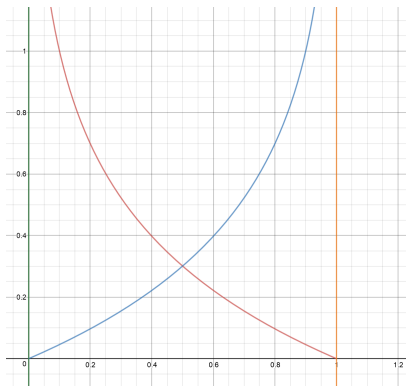


Figure 4: Cost Function for Classification

- ▶ Red is  $-\log(x)$ , for when  $y = 1$ , so if  $h_\theta = 1$ , cost = 0
- ▶ Blue is  $-\log(1 - x)$  for when  $y = 0$ , so if  $h_\theta = 0$ , cost = 0

## Combining the Cost Function

- ▶ Combined,  
 $cost((h_{\theta}(x)) = -y_i \log(h_{\theta}(x_i)) - (1 - y_i) \log(h_{\theta}(1 - x_i))$
- ▶ if  $y = 1$ ,  
 $cost((h_{\theta}(x), y) = -y_i \log(h_{\theta}(x_i)) - 0$
- ▶ if  $y = 0$ ,  
 $cost((h_{\theta}(x), y) = 0 - (1 - y_i) \log(h_{\theta}(1 - x_i))$
- ▶ So then,  
 $J(\theta) = \frac{1}{m} [\sum_{i=1}^m cost(h_{\theta}(x_i), y_i)]$
- ▶ Finally,  
 $J(\theta) = \frac{1}{m} [\sum_{i=1}^m -y_i \log(h_{\theta}(x_i)) - (1 - y_i) \log(h_{\theta}(1 - x_i))]$
- ▶ And therefore the derivative with respect to  $\theta_j$ ,  
 $\frac{\partial}{\partial \theta_j} J(\theta) = \sum_{i=1}^m [h_{\theta}(x_i) - y_i] (j x_i)$

# Minimizing the Cost Function with Gradient Descent

- Search for the set of coefficients that will minimize the next step in our cost function by calculating the slope of surrounding coefficients of the current data point. Chose the next data point with the smallest value.

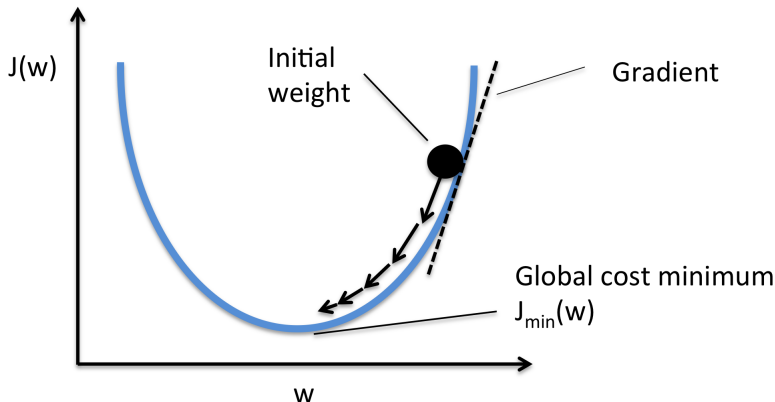


Figure 5: Minimizing the Cost Function with Gradient Descent

# Minimizing the Cost Function with Gradient Descent

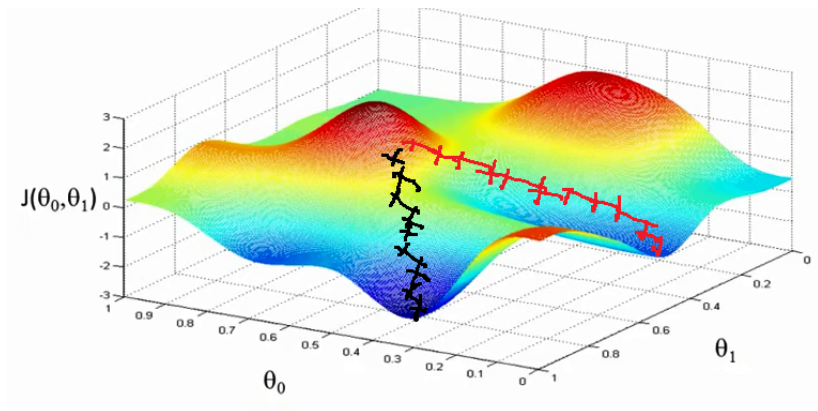


Figure 6: Gradient Descent Visualization

# Minimizing the Cost Function with Gradient Descent

- ▶ We want to minimize  $J(\theta)$
- ▶ Repeat:  $\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$  such that we are simultaneously updating all  $\theta_j$
- ▶  $\frac{\partial}{\partial \theta_j} J(\theta) = \sum_{i=1}^m [h_{\theta}(x_i) - y_i] (x_i^j)$
- ▶ Therefore our gradient descent will look like the following:  
Repeat:  $\theta_j := \theta_j - \alpha \sum_{i=1}^m [h_{\theta}(x_i) - y_i] (x_i^j)$

# Putting it All Together

- ▶ So we can classify positive and negative examples using a training set, then predict on "real" data.
- ▶ We train the hypothesis function with the cost function, and improve the cost using gradient descent.
- ▶ This creates a decision boundary, which we can then use to classify our data.
- ▶ By minimizing the cost function we train our hypothesis to create the best possible decision boundary.

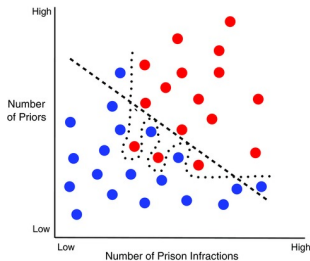


Figure 7: Decision Boundary

# Sources

- ▶ Stanford University Machine Learning Course Via Coursera
- ▶ Andrew Ng
- ▶ Kaggle Databases

# Lab! - Mushroom Classification



Figure 8: Mushroom Classification