# **Logistic Regression**

- Algorithm implemented in Octave
- —— Simple data were applied with the algorithm
- Same data were analyzed in R
- Same data were analyzed in Python

# **Algorithm**

Hypothesis  $h_{\theta}(x) = g(\theta^T x)$ 

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

Parameters:  $\theta_0, \theta_1, \dots, \theta_n$ 



cost function

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

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

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

Repeat {

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

$$\frac{\partial}{\partial \theta_{j}} J(\theta) = \frac{1}{2} \sum_{i=1}^{\infty} \left( \sum_{k=1}^{\infty} \left( \sum_{k=1}^{\infty$$

Advanced optimization

Optimization algorithms:

- Gradient descent
- Conjugate gradient
- **BFGS**
- L-BFGS

#### Advantages:

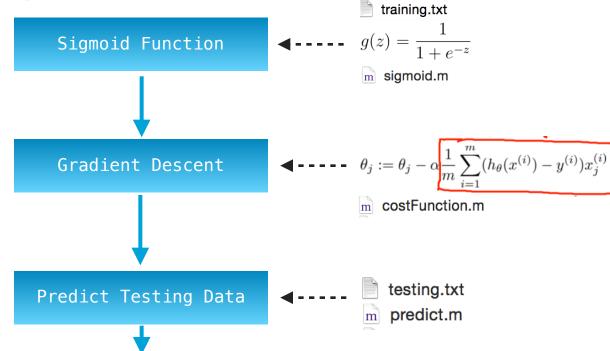
- No need to manually pick  $\alpha$
- Often faster than gradient descent.

#### Disadvantages:

More complex

## Implemented in Octave

m logistic regression.m



## Data Analyzed in R

#### File Name:

logistic\_regression.R

### Usage:

/usr/bin/Rscript logistic\_regression.R training.txt testing.txt prediction.txt

prediction.txt

### **Core Functions{Packages} Used:**

# build model glm{stats} # make prediction on testing data predict(stats)

## **Data Analyzed in Python**

#### File Name:

logistic\_regression.py

### Usage:

python logistic\_regression.py training.txt testing.txt prediction.txt

### **Core Functions Modules Used:**

# build model

LogisticRegression().fit(){sklearn} # make prediction on testing data LinearRegression().predict proba(){sklearn}