

Logistic Regression and Sentiment Classification

Chandresh Kumar Maurya
Post-Doc Fellow

IBM Research, Bangalore

October 27, 2018

What is machine learning?

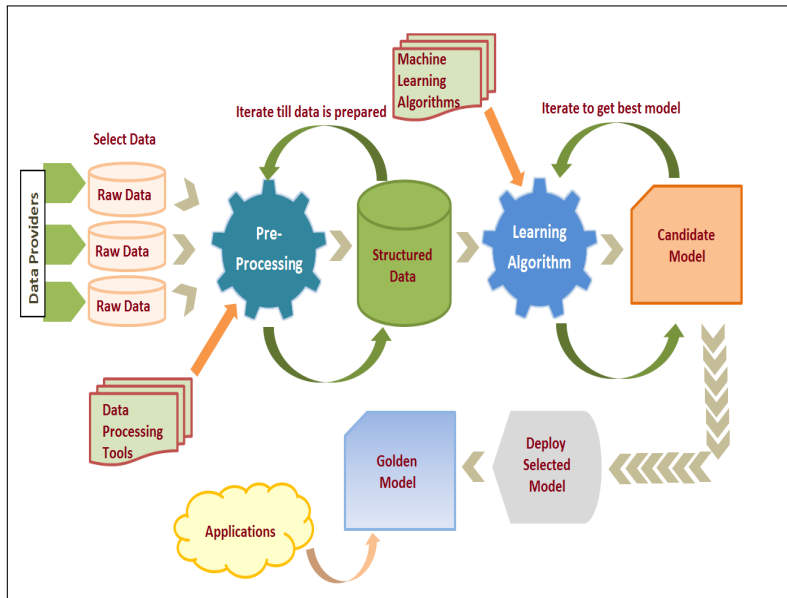
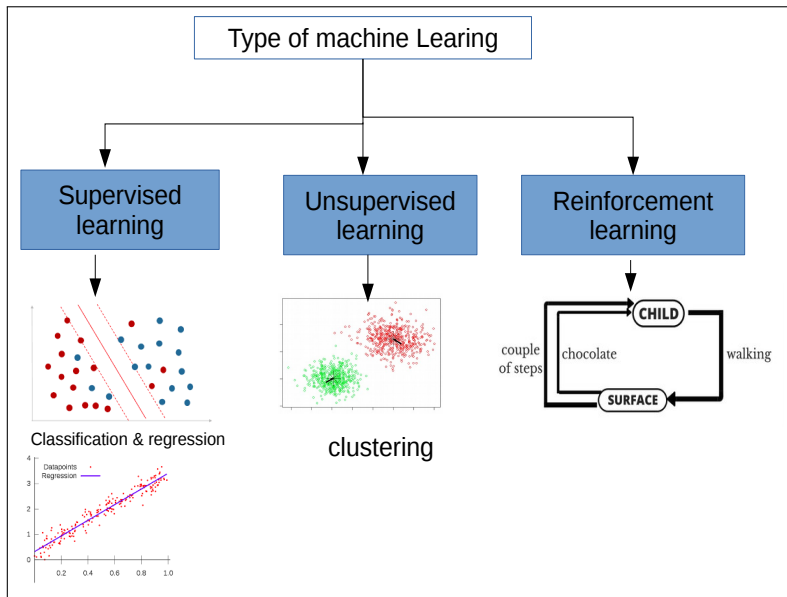


Figure: Steps in learning a machine

Types of machine learning?



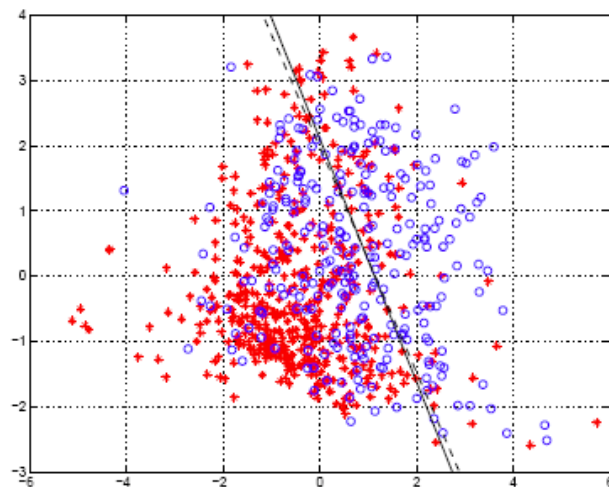
Classification Based on Probability

- Instead of just predicting the class, give the probability of the instance being that class
 - i.e., learn $p(y \mid \mathbf{x})$
- Comparison to perceptron:
 - Perceptron doesn't produce probability estimate

- Recall that:

$$0 \leq p(\text{event}) \leq 1$$

$$p(\text{event}) + p(\neg \text{event}) = 1$$



Logistic Regression

- Takes a probabilistic approach to learning discriminative functions (i.e., a classifier)

- $h_{\theta}(x)$ should give $p(y = 1 \mid x; \theta)$

– Want $0 \leq h_{\theta}(x) \leq 1$

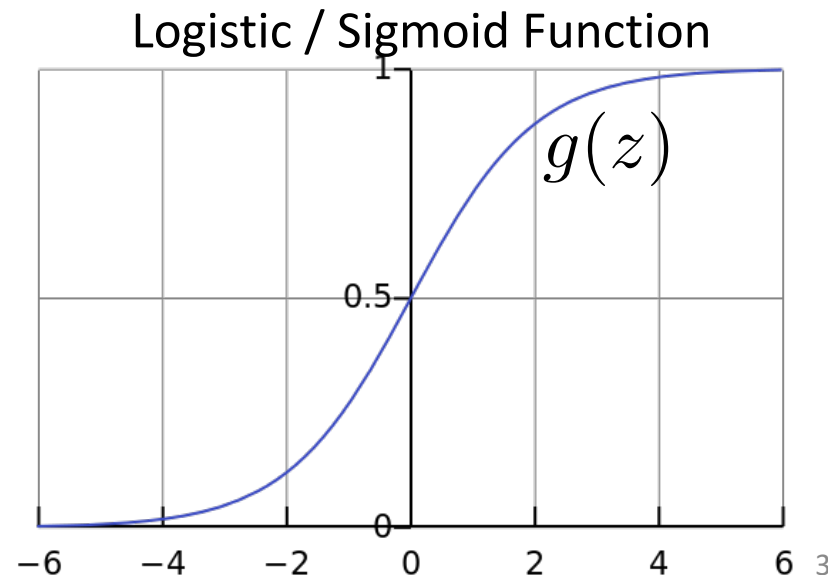
Can't just use linear regression with a threshold

- Logistic regression model:

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

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

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



Interpretation of Hypothesis Output

$$h_{\theta}(\mathbf{x}) = \text{estimated } p(y = 1 \mid \mathbf{x}; \theta)$$

Example: Cancer diagnosis from tumor size

$$\mathbf{x} = \begin{bmatrix} x_0 \\ x_1 \end{bmatrix} = \begin{bmatrix} 1 \\ \text{tumorSize} \end{bmatrix}$$

$$h_{\theta}(\mathbf{x}) = 0.7$$

→ Tell patient that 70% chance of tumor being malignant

Note that: $p(y = 0 \mid \mathbf{x}; \theta) + p(y = 1 \mid \mathbf{x}; \theta) = 1$

Therefore, $p(y = 0 \mid \mathbf{x}; \theta) = 1 - p(y = 1 \mid \mathbf{x}; \theta)$

Another Interpretation

- Equivalently, logistic regression assumes that

$$\log \frac{p(y = 1 \mid \mathbf{x}; \boldsymbol{\theta})}{p(y = 0 \mid \mathbf{x}; \boldsymbol{\theta})} = \theta_0 + \theta_1 x_1 + \dots + \theta_d x_d$$

odds of $y = 1$

Side Note: the odds in favor of an event is the quantity $p / (1 - p)$, where p is the probability of the event

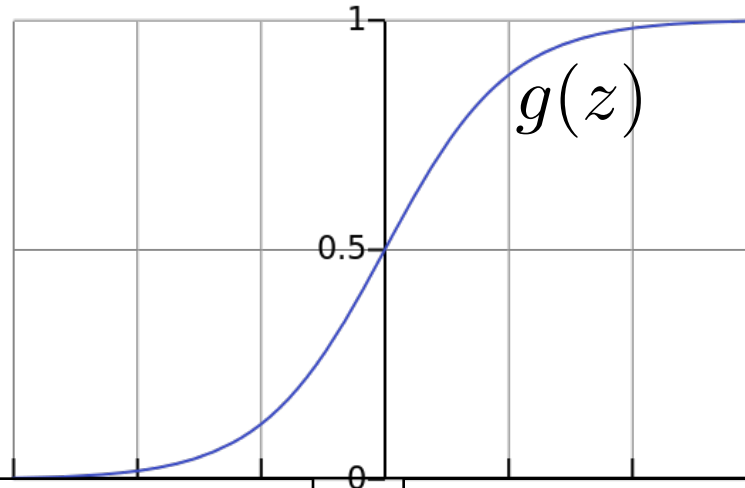
E.g., If I toss a fair dice, what are the odds that I will have a 6?

- In other words, logistic regression assumes that the log odds is a linear function of \mathbf{x}

Logistic Regression

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

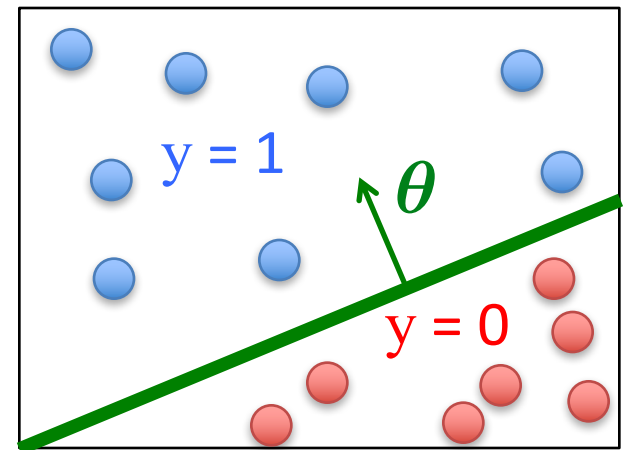
$$g(z) = \frac{1}{1 + e^{-z}}$$



$\theta^{\top} \mathbf{x}$ should be large negative values for negative instances

$\theta^{\top} \mathbf{x}$ should be large positive values for positive instances

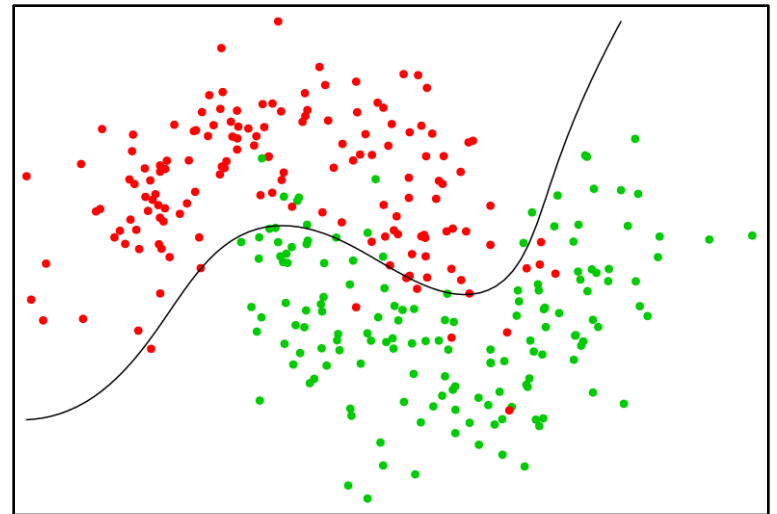
- Assume a threshold and...
 - Predict $y = 1$ if $h_{\theta}(\mathbf{x}) \geq 0.5$
 - Predict $y = 0$ if $h_{\theta}(\mathbf{x}) < 0.5$



Non-Linear Decision Boundary

- Can apply basis function expansion to features, same as with linear regression

$$\mathbf{x} = \begin{bmatrix} 1 \\ x_1 \\ x_2 \end{bmatrix} \rightarrow \begin{bmatrix} 1 \\ x_1 \\ x_2 \\ x_1 x_2 \\ x_1^2 \\ x_2^2 \\ x_1^2 x_2 \\ x_1 x_2^2 \\ \vdots \end{bmatrix}$$



Logistic Regression

- Given $\left\{ \left(\mathbf{x}^{(1)}, y^{(1)} \right), \left(\mathbf{x}^{(2)}, y^{(2)} \right), \dots, \left(\mathbf{x}^{(n)}, y^{(n)} \right) \right\}$
where $\mathbf{x}^{(i)} \in \mathbb{R}^d$, $y^{(i)} \in \{0, 1\}$

- Model: $h_{\boldsymbol{\theta}}(\mathbf{x}) = g(\boldsymbol{\theta}^T \mathbf{x})$

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

$$\boldsymbol{\theta} = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_d \end{bmatrix} \quad \mathbf{x}^T = \begin{bmatrix} 1 & x_1 & \dots & x_d \end{bmatrix}$$

Logistic Regression Objective Function

- Can't just use squared loss as in linear regression:

$$J(\boldsymbol{\theta}) = \frac{1}{2n} \sum_{i=1}^n \left(h_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)}) - y^{(i)} \right)^2$$

- Using the logistic regression model

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

results in a non-convex optimization

Deriving the Cost Function via Maximum Likelihood Estimation

- Likelihood of data is given by: $l(\boldsymbol{\theta}) = \prod_{i=1}^n p(y^{(i)} \mid \mathbf{x}^{(i)}; \boldsymbol{\theta})$
- So, looking for the $\boldsymbol{\theta}$ that maximizes the likelihood

$$\boldsymbol{\theta}_{\text{MLE}} = \arg \max_{\boldsymbol{\theta}} l(\boldsymbol{\theta}) = \arg \max_{\boldsymbol{\theta}} \prod_{i=1}^n p(y^{(i)} \mid \mathbf{x}^{(i)}; \boldsymbol{\theta})$$

- Can take the log without changing the solution:

$$\begin{aligned}\boldsymbol{\theta}_{\text{MLE}} &= \arg \max_{\boldsymbol{\theta}} \log \prod_{i=1}^n p(y^{(i)} \mid \mathbf{x}^{(i)}; \boldsymbol{\theta}) \\ &= \arg \max_{\boldsymbol{\theta}} \sum_{i=1}^n \log p(y^{(i)} \mid \mathbf{x}^{(i)}; \boldsymbol{\theta})\end{aligned}$$

Deriving the Cost Function via Maximum Likelihood Estimation

- Expand as follows:

$$\begin{aligned}\theta_{\text{MLE}} &= \arg \max_{\theta} \sum_{i=1}^n \log p(y^{(i)} \mid \mathbf{x}^{(i)}; \theta) \\ &= \arg \max_{\theta} \sum_{i=1}^n \left[y^{(i)} \log p(y^{(i)} = 1 \mid \mathbf{x}^{(i)}; \theta) + (1 - y^{(i)}) \log (1 - p(y^{(i)} = 1 \mid \mathbf{x}^{(i)}; \theta)) \right]\end{aligned}$$

- Substitute in model, and take negative to yield

Logistic regression objective:

$$\begin{aligned}\min_{\theta} J(\theta) \\ J(\theta) = - \sum_{i=1}^n \left[y^{(i)} \log h_{\theta}(\mathbf{x}^{(i)}) + (1 - y^{(i)}) \log (1 - h_{\theta}(\mathbf{x}^{(i)})) \right]\end{aligned}$$

Intuition Behind the Objective

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

- Cost of a single instance:

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

- Can re-write objective function as

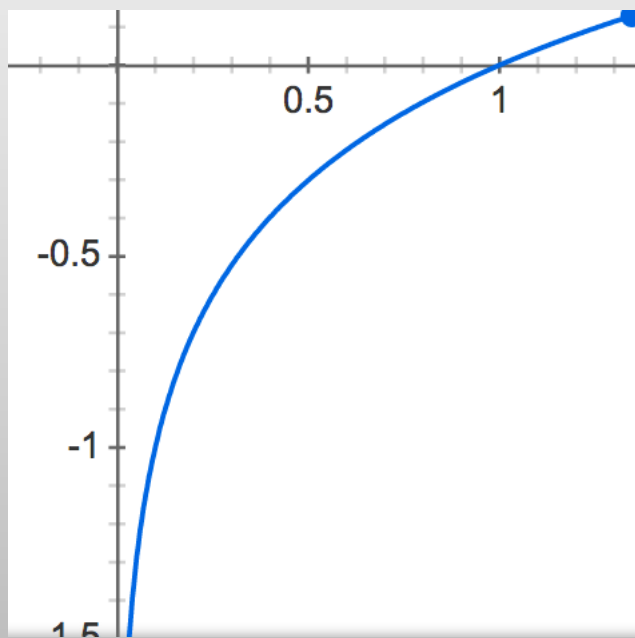
$$J(\boldsymbol{\theta}) = \sum_{i=1}^n \text{cost}(h_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}), y^{(i)})$$

Compare to linear regression: $J(\boldsymbol{\theta}) = \frac{1}{2n} \sum_{i=1}^n \left(h_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}) - y^{(i)} \right)^2$

Intuition Behind the Objective

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

Aside: Recall the plot of $\log(z)$

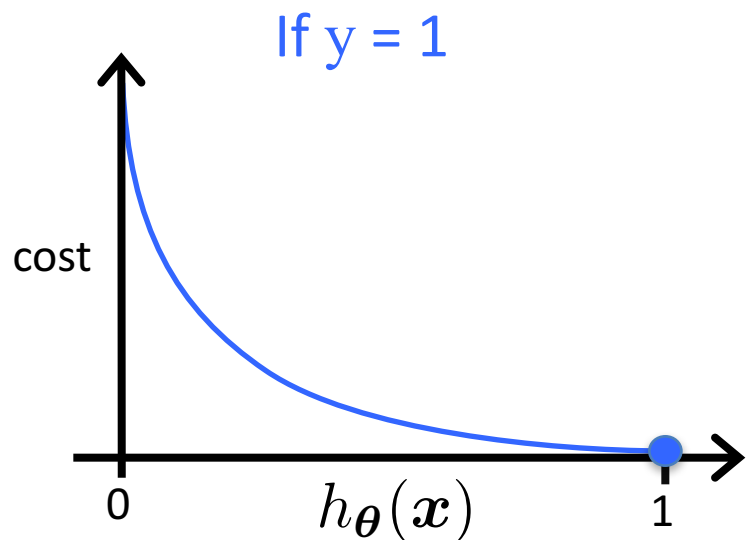


Intuition Behind the Objective

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

If $y = 1$

- Cost = 0 if prediction is correct
- As $h_{\theta}(\mathbf{x}) \rightarrow 0$, cost $\rightarrow \infty$
- Captures intuition that larger mistakes should get larger penalties
 - e.g., predict $h_{\theta}(\mathbf{x}) = 0$, but $y = 1$

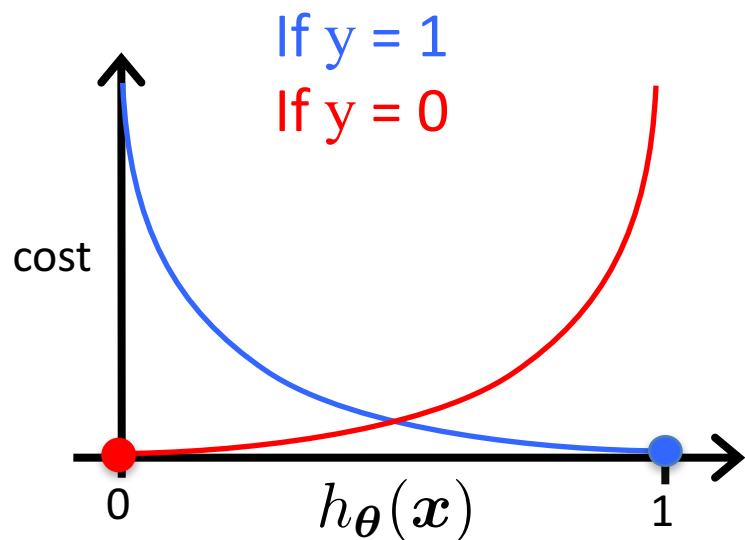


Intuition Behind the Objective

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

If $y = 0$

- Cost = 0 if prediction is correct
- As $(1 - h_{\theta}(\mathbf{x})) \rightarrow 0$, $\text{cost} \rightarrow \infty$
- Captures intuition that larger mistakes should get larger penalties



Regularized Logistic Regression

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

- We can regularize logistic regression exactly as before:

$$\begin{aligned} J_{\text{regularized}}(\boldsymbol{\theta}) &= J(\boldsymbol{\theta}) + \lambda \sum_{j=1}^d \theta_j^2 \\ &= J(\boldsymbol{\theta}) + \lambda \|\boldsymbol{\theta}_{[1:d]}\|_2^2 \end{aligned}$$

Gradient Descent for Logistic Regression

$$J_{\text{reg}}(\boldsymbol{\theta}) = - \sum_{i=1}^n \left[y^{(i)} \log h_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}) + (1 - y^{(i)}) \log (1 - h_{\boldsymbol{\theta}}(\mathbf{x}^{(i)})) \right] + \lambda \|\boldsymbol{\theta}_{[1:d]}\|_2^2$$

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

- Initialize $\boldsymbol{\theta}$
- Repeat until convergence

$$\theta_j \leftarrow \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\boldsymbol{\theta})$$

simultaneous update
for $j = 0 \dots d$

Use the natural logarithm ($\ln = \log_e$) to cancel with the $\exp()$ in $h_{\boldsymbol{\theta}}(\mathbf{x})$

Gradient Descent for Logistic Regression

$$J_{\text{reg}}(\boldsymbol{\theta}) = - \sum_{i=1}^n \left[y^{(i)} \log h_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}) + (1 - y^{(i)}) \log (1 - h_{\boldsymbol{\theta}}(\mathbf{x}^{(i)})) \right] + \lambda \|\boldsymbol{\theta}_{[1:d]}\|_2^2$$

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

- Initialize $\boldsymbol{\theta}$
- Repeat until convergence (simultaneous update for $j = 0 \dots d$)

$$\theta_0 \leftarrow \theta_0 - \alpha \sum_{i=1}^n \left(h_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}) - y^{(i)} \right)$$

$$\theta_j \leftarrow \theta_j - \alpha \left[\sum_{i=1}^n \left(h_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}) - y^{(i)} \right) x_j^{(i)} - \frac{\lambda}{n} \theta_j \right]$$

Gradient Descent for Logistic Regression

- Initialize θ
- Repeat until convergence (simultaneous update for $j = 0 \dots d$)

$$\theta_0 \leftarrow \theta_0 - \alpha \sum_{i=1}^n \left(h_{\theta} \left(\mathbf{x}^{(i)} \right) - y^{(i)} \right)$$

$$\theta_j \leftarrow \theta_j - \alpha \left[\sum_{i=1}^n \left(h_{\theta} \left(\mathbf{x}^{(i)} \right) - y^{(i)} \right) x_j^{(i)} - \frac{\lambda}{n} \theta_j \right]$$

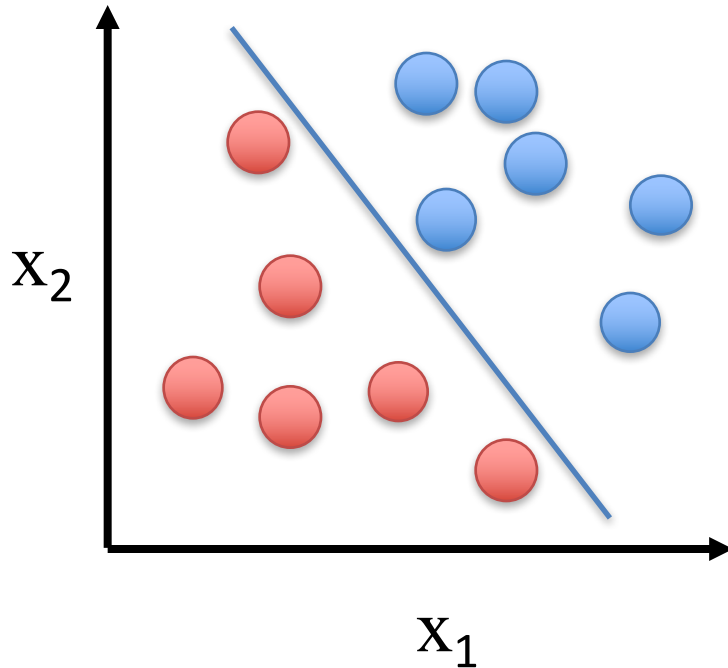
This looks IDENTICAL to linear regression!!!

- Ignoring the $1/n$ constant
- However, the form of the model is very different:

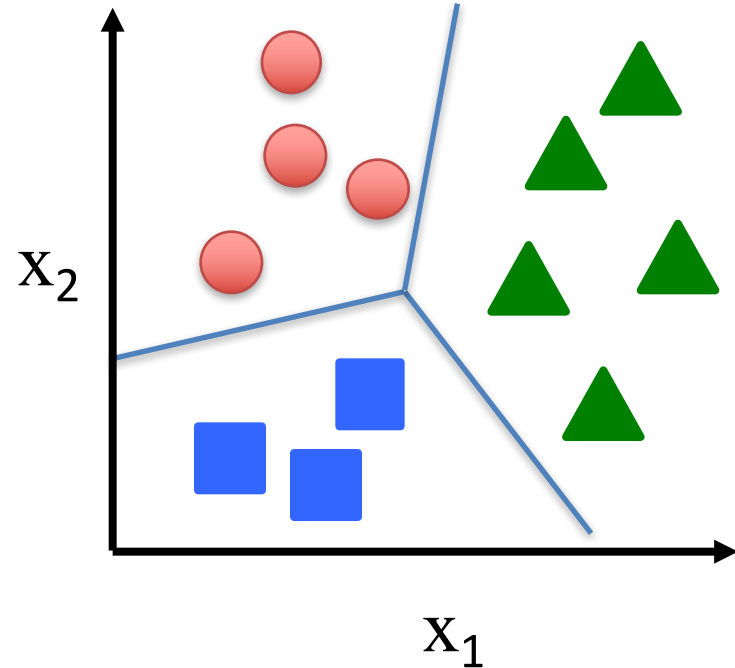
$$h_{\theta}(\mathbf{x}) = \frac{1}{1 + e^{-\theta^T \mathbf{x}}}$$

Multi-Class Classification

Binary classification:



Multi-class classification:



Disease diagnosis: healthy / cold / flu / pneumonia

Object classification: desk / chair / monitor / bookcase

Multi-Class Logistic Regression

- For 2 classes:

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

weight assigned to $y = 0$ weight assigned to $y = 1$

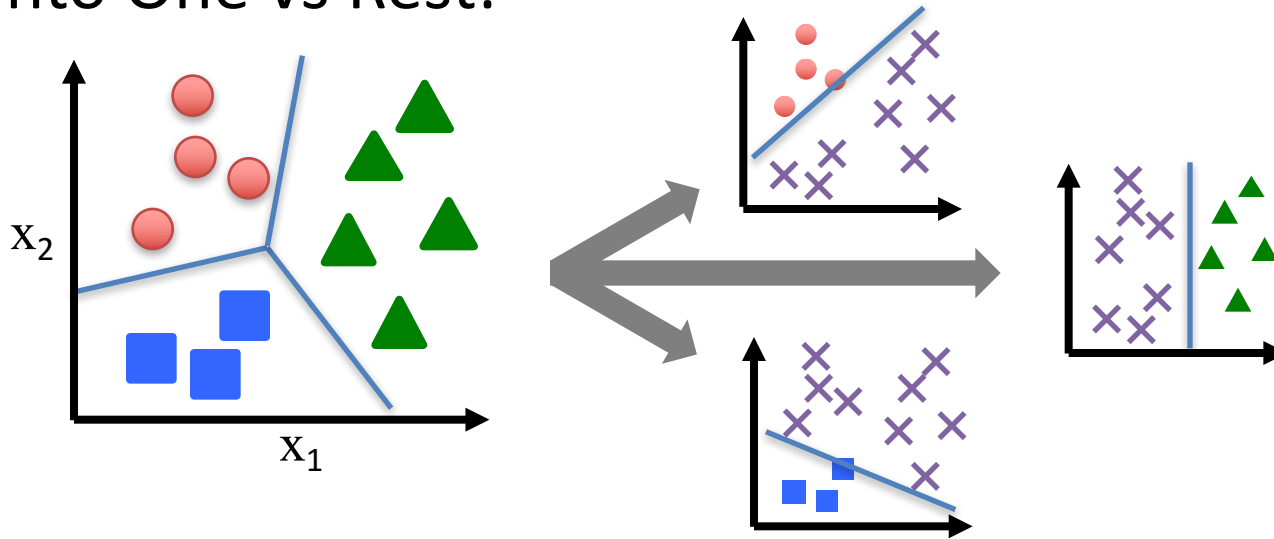
- For C classes $\{1, \dots, C\}$:

$$p(y = c \mid x; \theta_1, \dots, \theta_C) = \frac{\exp(\theta_c^T x)}{\sum_{c=1}^C \exp(\theta_c^T x)}$$

- Called the **softmax** function

Multi-Class Logistic Regression

Split into One vs Rest:



- Train a logistic regression classifier for each class i to predict the probability that $y = i$ with

$$h_c(\mathbf{x}) = \frac{\exp(\boldsymbol{\theta}_c^T \mathbf{x})}{\sum_{c=1}^C \exp(\boldsymbol{\theta}_c^T \mathbf{x})}$$

Implementing Multi-Class Logistic Regression

- Use $h_c(\mathbf{x}) = \frac{\exp(\boldsymbol{\theta}_c^\top \mathbf{x})}{\sum_{c=1}^C \exp(\boldsymbol{\theta}_c^\top \mathbf{x})}$ as the model for class c
- Gradient descent simultaneously updates all parameters for all models
 - Same derivative as before, just with the above $h_c(\mathbf{x})$
- Predict class label as the most probable label

$$\max_c h_c(\mathbf{x})$$