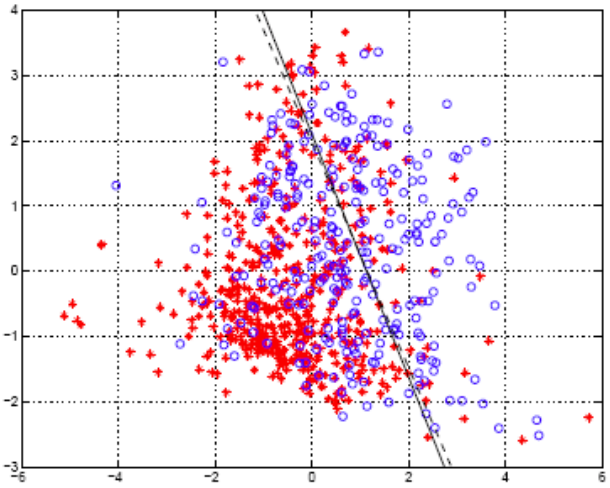
Logistic Regression

Classification Based on Probability

- Instead of just predicting the class, give the probability of the instance being that class
  - i.e., learn $p(y \mid \boldsymbol{x})$
- Comparison to perceptron:
  - Perceptron doesn't produce probability estimate
  - Perceptron (and other discriminative classifiers) are only interested in producing a discriminative model
- 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)
- $h_{\boldsymbol{\theta}}(\boldsymbol{x})$ should give $p(y = 1 \mid \boldsymbol{x}; \boldsymbol{\theta})$
  - Want $0 \leq h_{\boldsymbol{\theta}}(\boldsymbol{x}) \leq 1$

Can't just use linear regression with a threshold

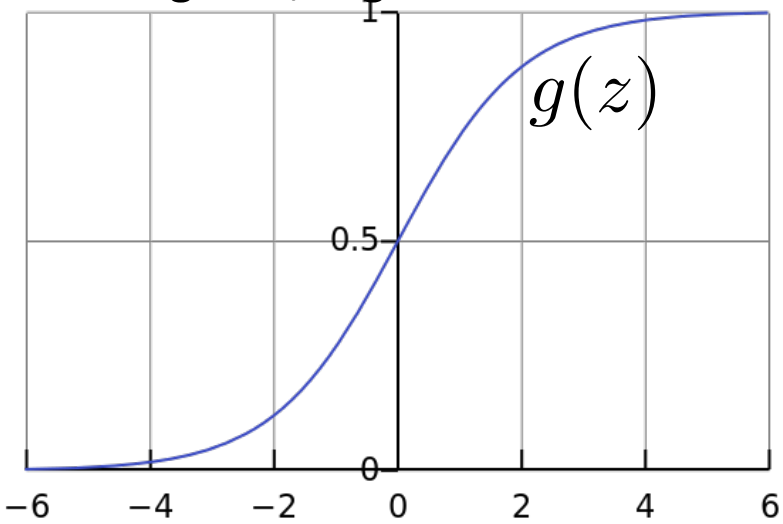
- Logistic regression model:

$$h_{\boldsymbol{\theta}}(\boldsymbol{x}) = g\left(\boldsymbol{\theta}^{\mathsf{T}}\boldsymbol{x}\right)$$

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

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

Logistic / Sigmoid Function



Interpretation of Hypothesis Output

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

Example: Cancer diagnosis from tumor size

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

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

→ Tell patient that 70% chance of tumor being malignant

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

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

Another Interpretation

- Equivalently, logistic regression assumes that

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

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

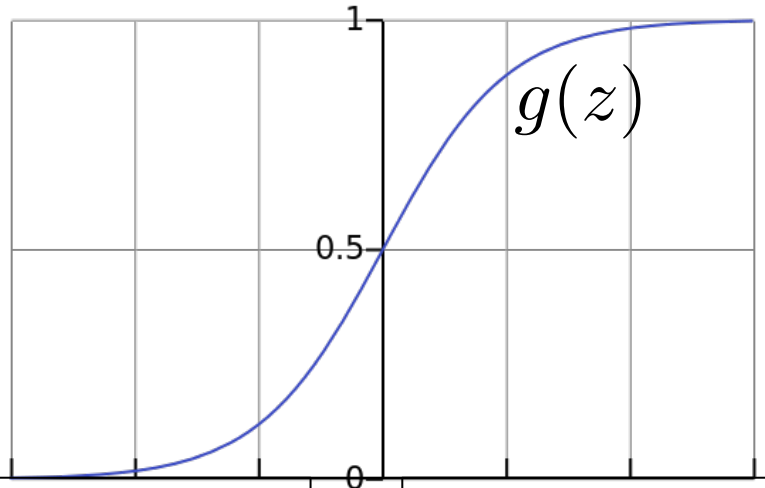
E.g., If I toss a fair dice, what are the odds that I will have a 6?

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

Logistic Regression

$$h_{\boldsymbol{\theta}}(\boldsymbol{x}) = g\left(\boldsymbol{\theta}^{\mathsf{T}}\boldsymbol{x}\right)$$

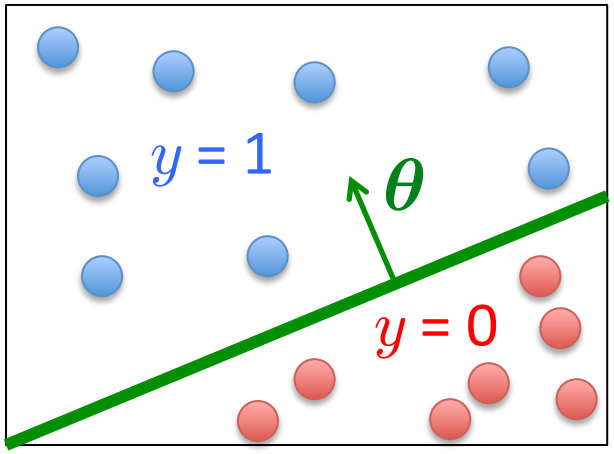
$$g(z) = \frac{1}{1+e^{-z}}$$



$\boldsymbol{\theta}^{\mathsf{T}}\boldsymbol{x}$ should be large <u>negative</u> values for negative instances

$\boldsymbol{\theta}^{\mathsf{T}}\boldsymbol{x}$ should be large <u>positive</u> values for positive instances

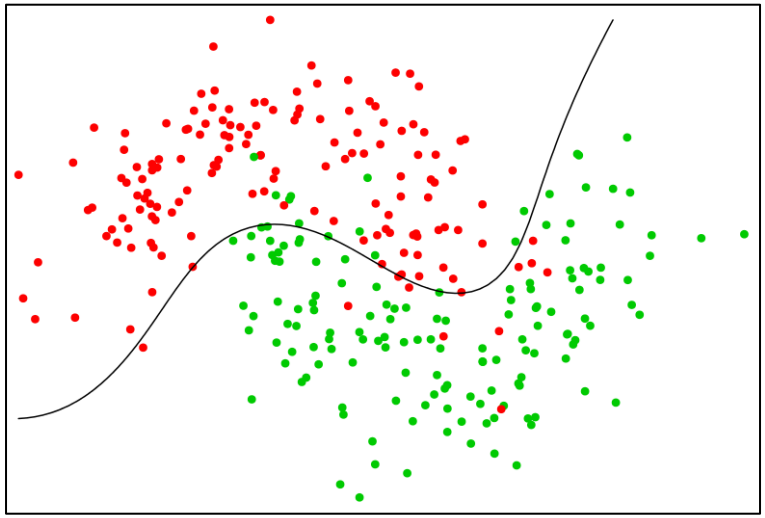
- Assume a threshold and...
  - Predict $y = 1$ if $h_{\boldsymbol{\theta}}(\boldsymbol{x}) \geq 0.5$
  - Predict $y = 0$ if $h_{\boldsymbol{\theta}}(\boldsymbol{x}) < 0.5$



Non-Linear Decision Boundary

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

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



Logistic Regression

- Given $\left\{ \left(\boldsymbol{x}^{(1)}, y^{(1)} \right), \left(\boldsymbol{x}^{(2)}, y^{(2)} \right), \ldots, \left(\boldsymbol{x}^{(n)}, y^{(n)} \right) \right\}$

  where $\boldsymbol{x}^{(i)} \in \mathbb{R}^d, \; y^{(i)} \in \{0, 1\}$

- Model: $h_{\boldsymbol{\theta}}(\boldsymbol{x}) = g\left(\boldsymbol{\theta}^\intercal \boldsymbol{x}\right)$

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

$$\boldsymbol{\theta} = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_d \end{bmatrix} \qquad \boldsymbol{x}^\intercal = \begin{bmatrix} 1 & x_1 & \ldots & x_d \end{bmatrix}$$

Logistic Regression Objective Function

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

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

  - Using the logistic regression model

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

    results in a non-convex optimization

Deriving the Cost Function via Maximum Likelihood Estimation

- Likelihood of data is given by: $l(\boldsymbol{\theta}) = \prod_{i=1}^{n} p(y^{(i)} \mid \boldsymbol{x}^{(i)}; \boldsymbol{\theta})$

- So, looking for the $\boldsymbol{\theta}$ that maximizes the likelihood

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

- Can take the log without changing the solution:

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

$$= \arg\max_{\boldsymbol{\theta}} \sum_{i=1}^{n} \log p(y^{(i)} \mid \boldsymbol{x}^{(i)}; \boldsymbol{\theta})$$

Deriving the Cost Function via Maximum Likelihood Estimation

- Expand as follows:

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

$$= \arg\max_{\boldsymbol{\theta}} \sum_{i=1}^{n} \left[y^{(i)} \log p(y^{(i)} = 1 \mid \boldsymbol{x}^{(i)}; \boldsymbol{\theta}) + \left(1 - y^{(i)}\right) \log \left(1 - p(y^{(i)} = 1 \mid \boldsymbol{x}^{(i)}; \boldsymbol{\theta})\right) \right]$$

- Substitute in model, and take negative to yield

Logistic regression objective:

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

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

Intuition Behind the Objective

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

- Cost of a single instance:

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

- Can re-write objective function as

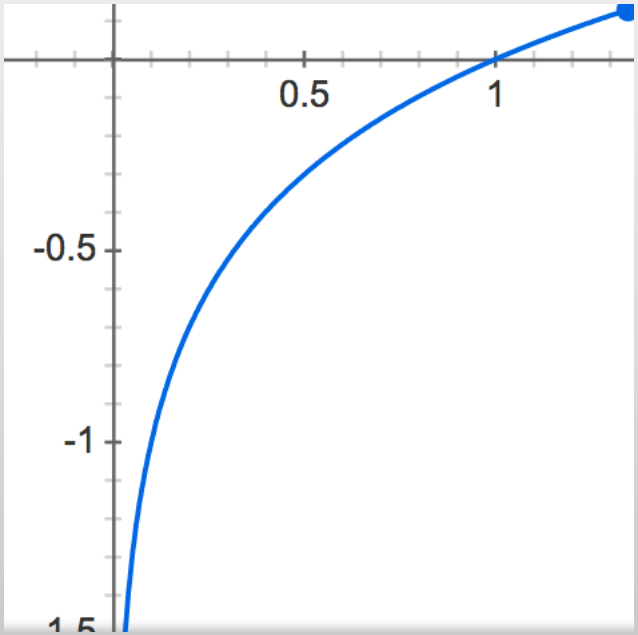
$$J(\boldsymbol{\theta}) = \sum_{i=1}^{n} \text{cost}\left(h_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)}), y^{(i)}\right)$$

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

Intuition Behind the Objective

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

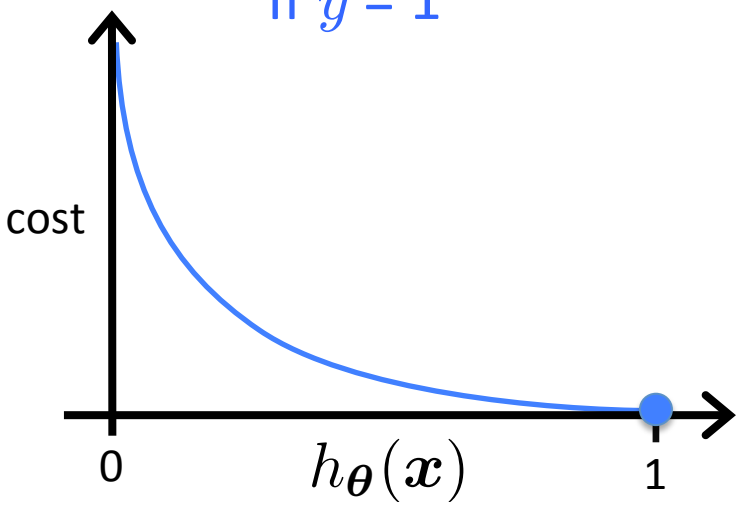
Aside: Recall the plot of $\log(z)$



Intuition Behind the Objective

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

If y = 1



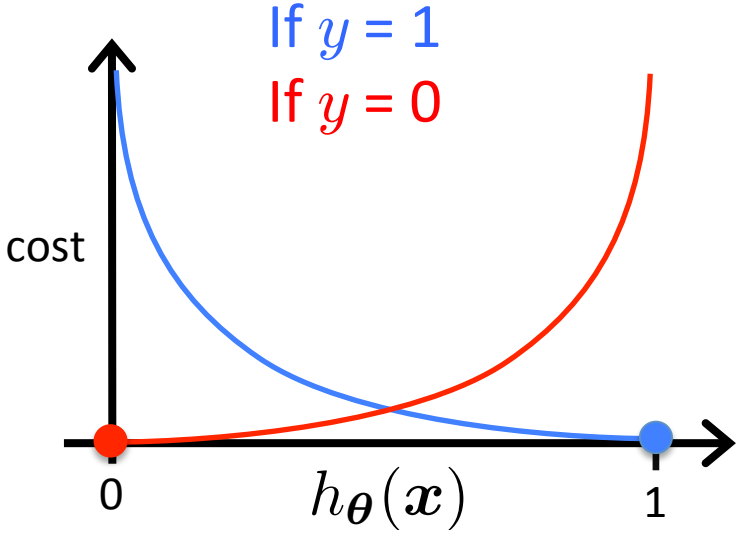
If y = 1

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

Based on example by Andrew Ng

Intuition Behind the Objective

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



If y = 0

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

Based on example by Andrew Ng

Regularized Logistic Regression

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

- 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

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

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 ... d

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

Gradient Descent for Logistic Regression

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

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

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

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

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

Gradient Descent for Logistic Regression

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

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

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

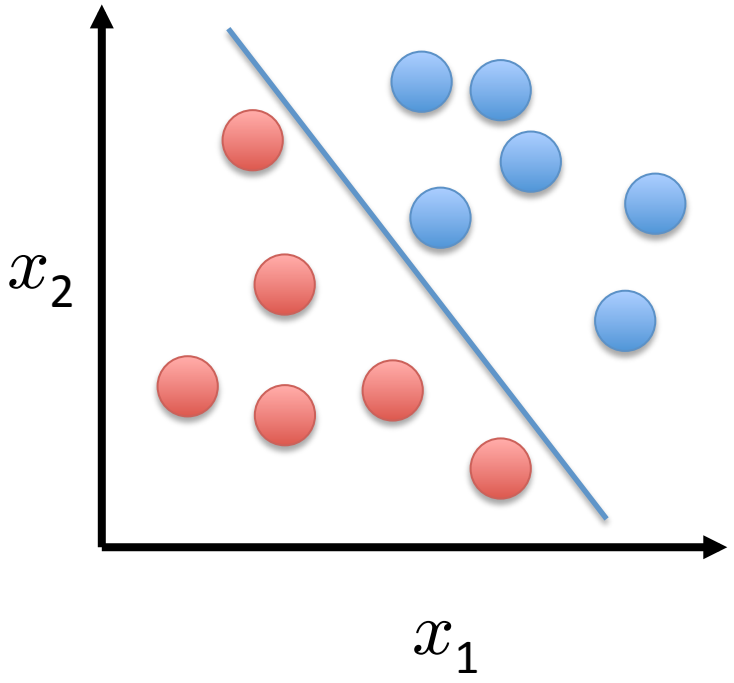
This looks IDENTICAL to linear regression!!!

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

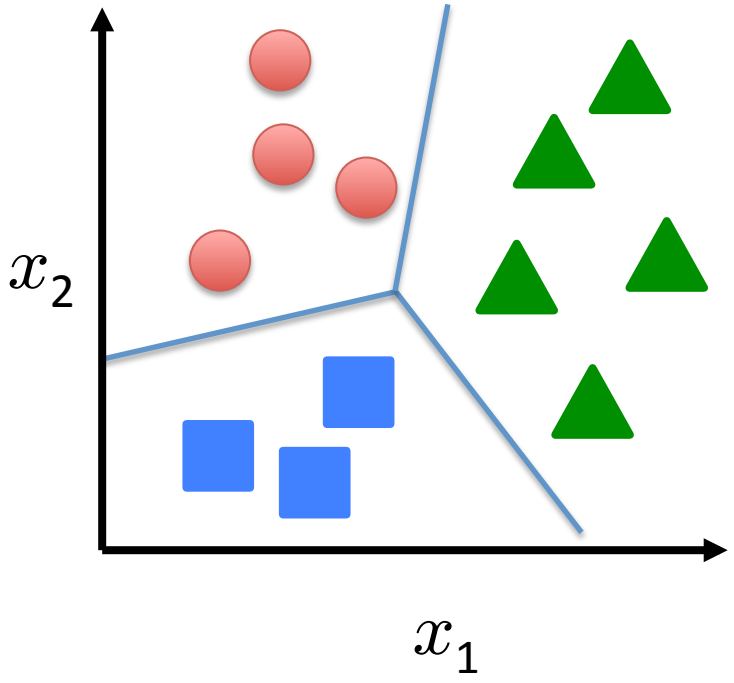
$$h_{\boldsymbol{\theta}}(\boldsymbol{x}) = \frac{1}{1 + e^{-\boldsymbol{\theta}^\top \boldsymbol{x}}}$$

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:

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

weight assigned to $y = 0$

weight assigned to $y = 1$

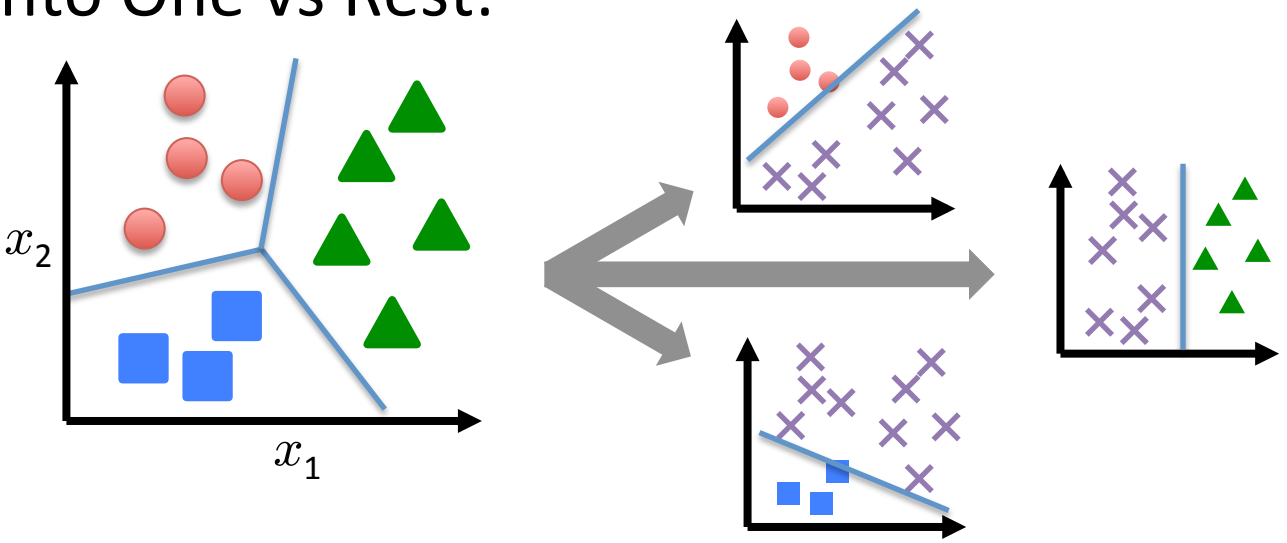
- For C classes {1, ..., C}:

$$p(y = c \mid \boldsymbol{x}; \boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_C) = \frac{\exp(\boldsymbol{\theta}_c^{\mathsf{T}}\boldsymbol{x})}{\sum_{c=1}^{C}\exp(\boldsymbol{\theta}_c^{\mathsf{T}}\boldsymbol{x})}$$

  - 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

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

Implementing Multi-Class Logistic Regression

- Use $h_c(\boldsymbol{x}) = \dfrac{\exp(\boldsymbol{\theta}_c^\mathsf{T}\boldsymbol{x})}{\sum_{c=1}^{C}\exp(\boldsymbol{\theta}_c^\mathsf{T}\boldsymbol{x})}$ as the model for class c

- Gradient descent simultaneously updates all parameters for all models
  - Same derivative as before, just with the above $h_c(\boldsymbol{x})$

- Predict class label as the most probable label

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