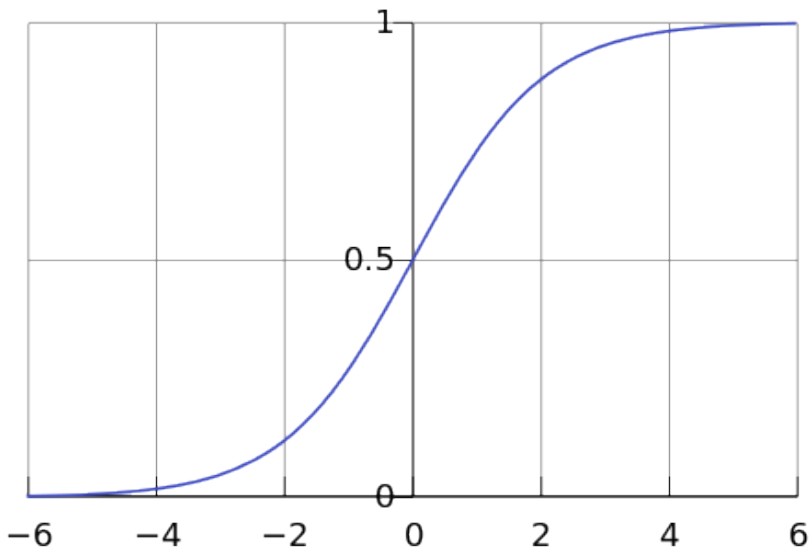


Image: Wikipedia

Logistic regression Classification

Data Mining:
Jay Urbain, PhD

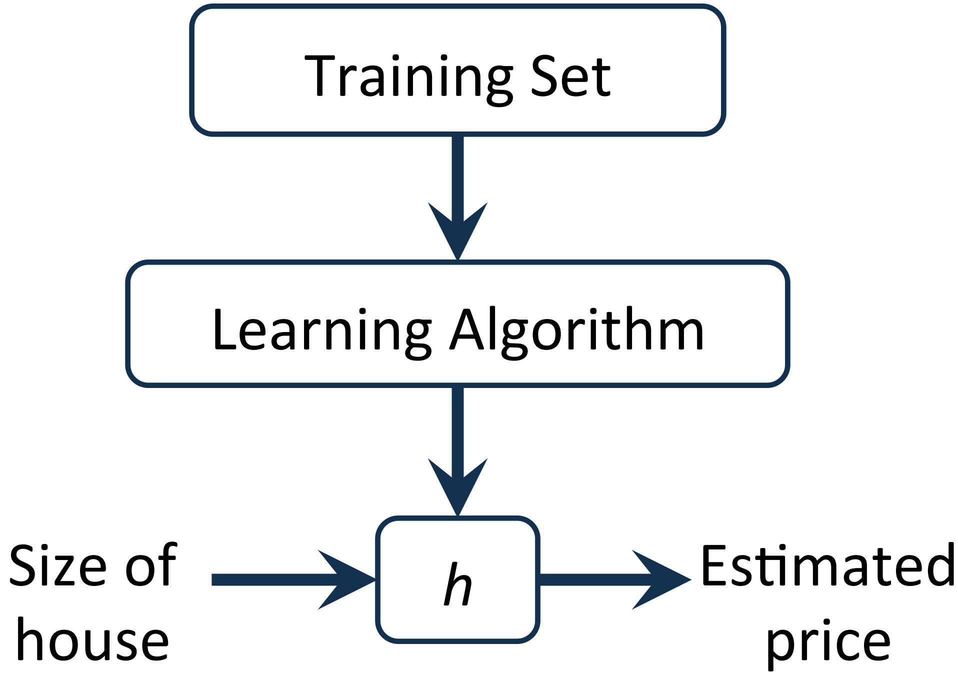
Credits: Andrew Ng, Stanford; Hastie and Tibshirani, Stanford

Numeric Prediction: Linear Regression

- When what we want to predict is *numeric* and when all attributes are *numeric*.
  - Note: If attributes are not numeric, can convert to *indicator variables*.
- Staple method in statistics.
- Express class as a linear combination of the attributes, with associated weights learned from training data.

$$y_i = \beta_1 x_{i1} + \cdots + \beta_p x_{ip} + \varepsilon_i = \mathbf{x}_i^{\mathrm{T}} \boldsymbol{\beta} + \varepsilon_i, \qquad i = 1, \ldots, n,$$

- Excellent method, simple, fast, *but* linear.
  - If the data exhibits a nonlinear dependency, the best fitting straight line will be found.



h – hypothesis to learn
h maps from x to y

Learn weights to *minimize* sum of squared error between *hypothesis* and ground truth *y*.

$$h_\theta(x) = \theta_0 + \theta_1 x$$

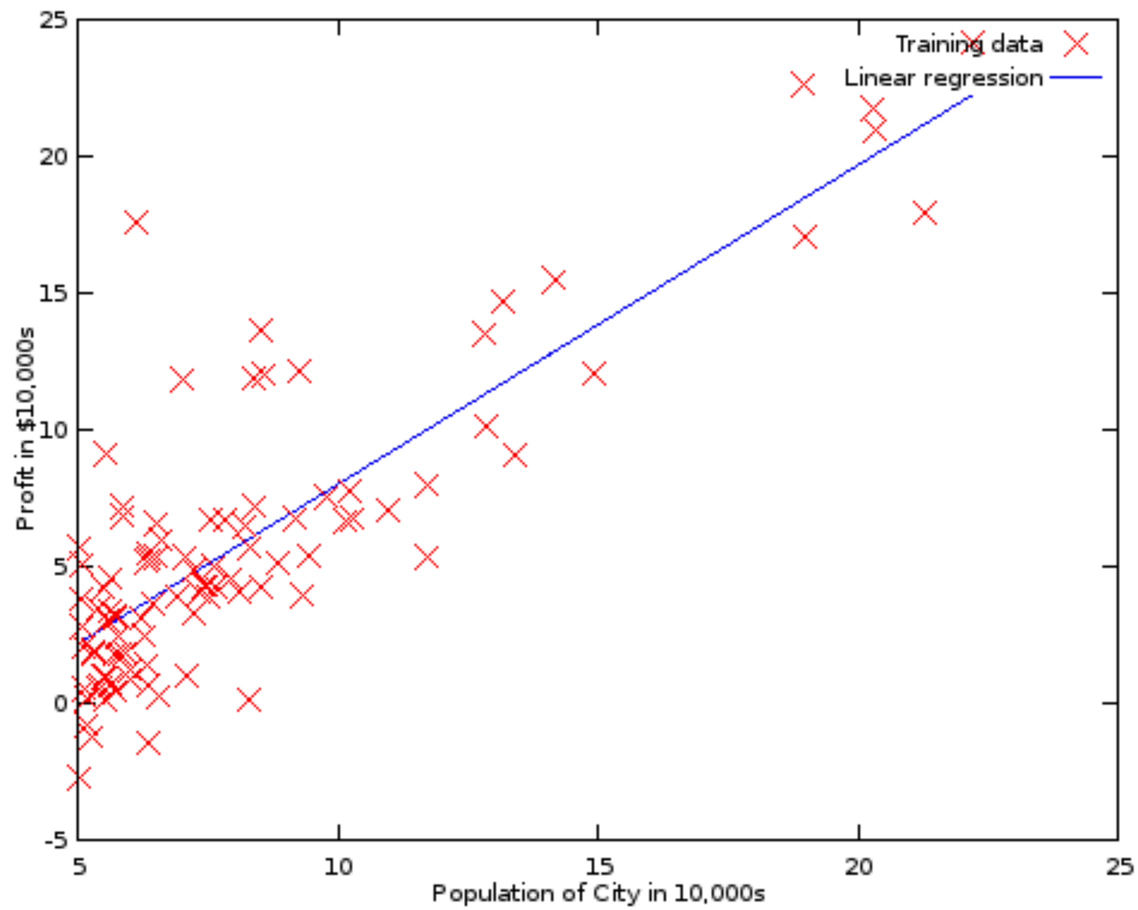
$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$$

Linear regression

Theta found by gradient descent: -3.630291 1.166362

For population = 35,000, we predict a profit of 4519.767868

For population = 70,000, we predict a profit of 45342.450129



Classification

Email: Spam / Not Spam?
Online Transactions: Fraudulent (Yes / No)?
Tumor: Malignant / Benign ?

$$y \in \{0, 1\}$$

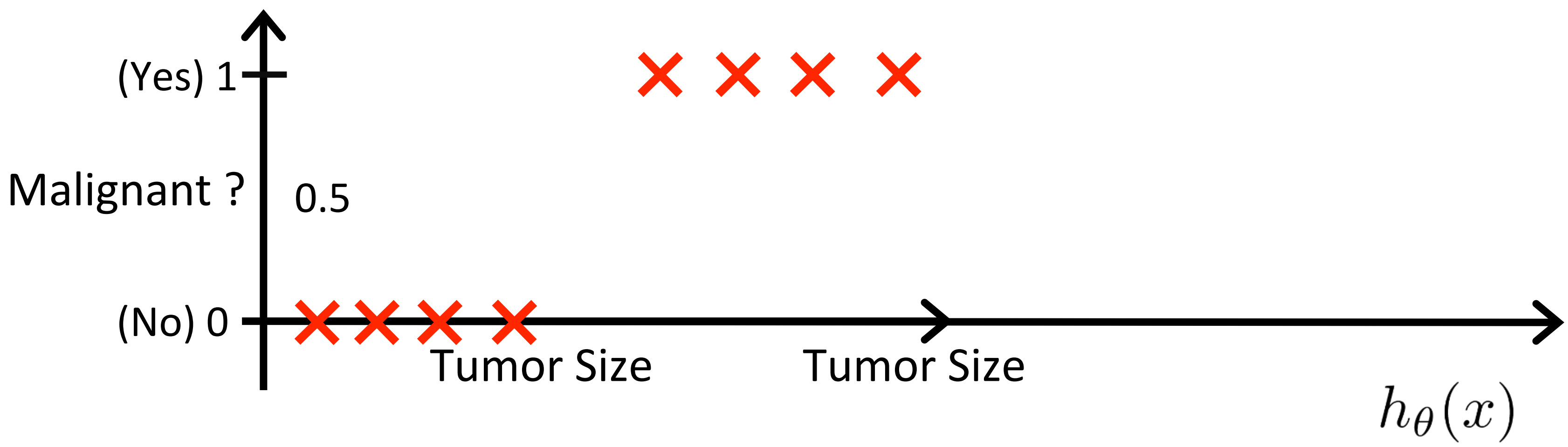
0: “Negative Class” (e.g., benign tumor)

1: “Positive Class” (e.g., malignant tumor)

Multinomial classification: y {0, 1, 2, 3, …}

Linear Regression for classification

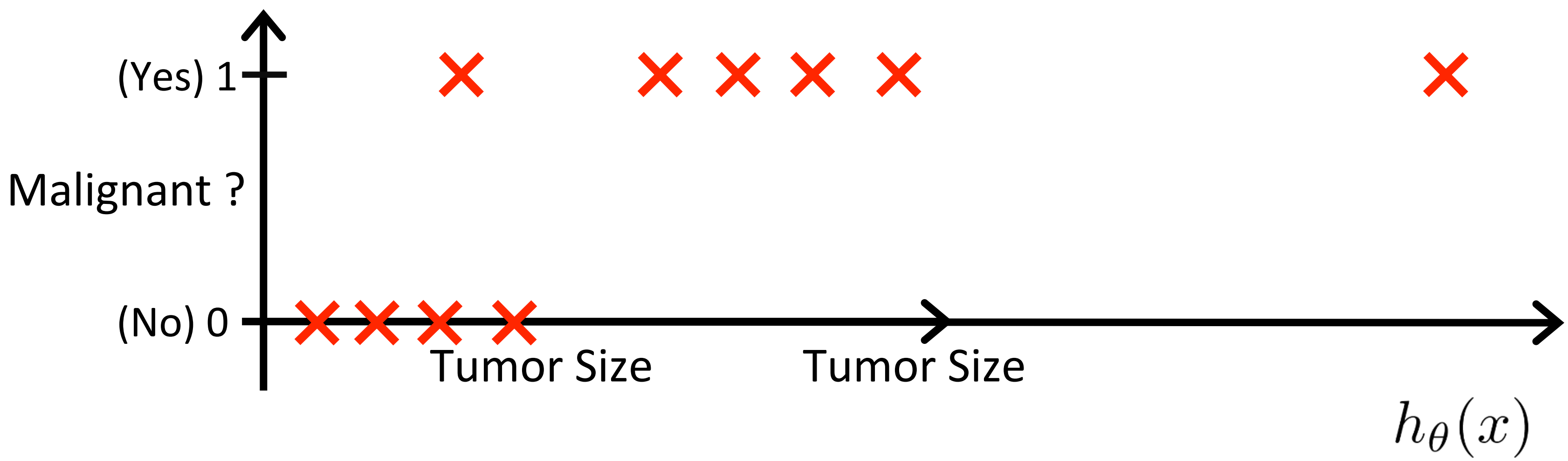
- Linear regression can be used for classification in domains with numeric attributes.
  - Perform a regression for each class, set output to 1 for instances that belong to the class, and 0 for those that do not.
  - The result is a linear expression for each class.
  - Then, given a test instance of an unknown class, calculate the value of each linear expression and choose the one that is **largest**.
  - *Called multinomial linear regression.*
  - Problems: output is not a proper probability, assumes errors are not statistically significant.



Threshold classifier output $h_\theta(x)$ at 0.5:

If $h_\theta(x) \geq 0.5$, predict "y = 1"

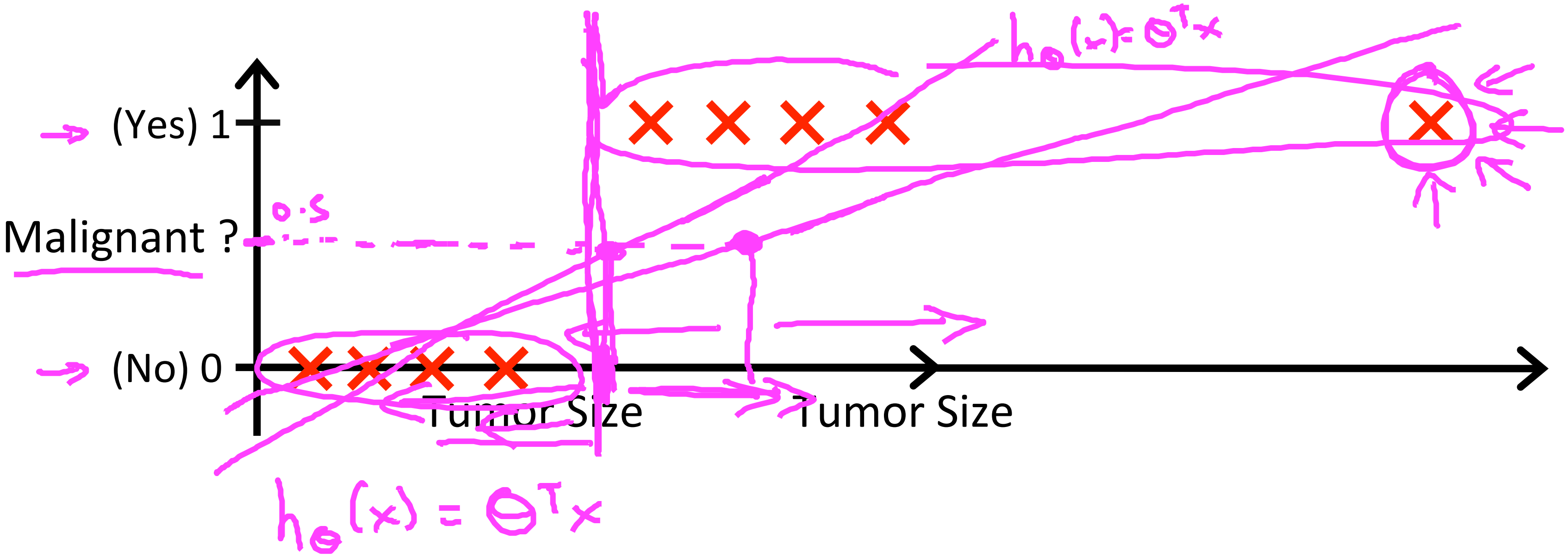
If $h_\theta(x) < 0.5$, predict "y = 0"



Threshold classifier output $h_\theta(x)$ at 0.5:

If $h_\theta(x) \geq 0.5$, predict "y = 1"

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



Threshold classifier output $h_\theta(x)$ at 0.5:

If $h_\theta(x) \geq 0.5$, predict "y = 1"

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

Classification: y = 0 or 1

$h_\theta(x)$ can be > 1 or < 0

With regression

Logistic *Regression*: $0 \leq h_\theta(x) \leq 1$

Want proper probability

Classification

Logistic Regression Model: hypothesis representation

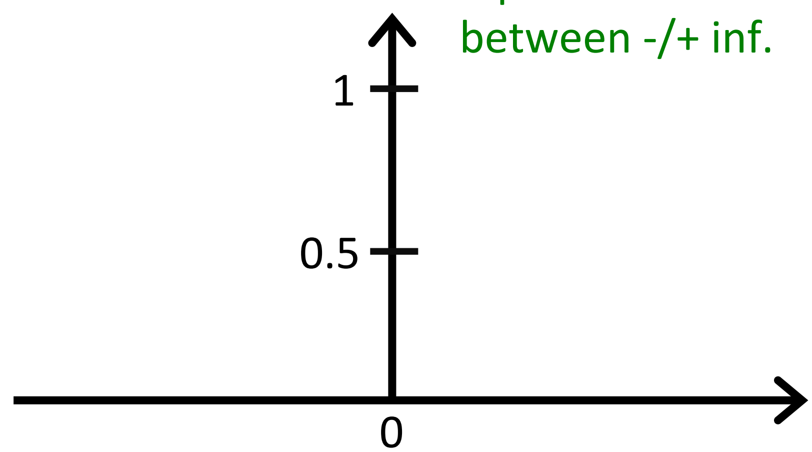
Want $0 \leq h_\theta(x) \leq 1$

Replaces original target variable which can not be approximated easily with a nonlinear function.

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

Input variables can range between -/+ inf.

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



Sigmoid function

==

Logistic function

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

Logistic Regression Model: hypothesis representation

Want $\boxed{0 \leq h_\theta(x) \leq 1}$

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

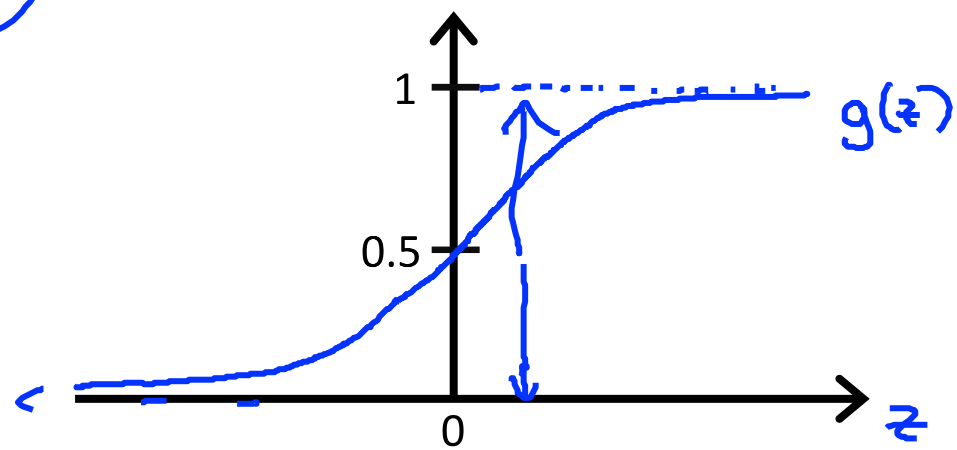
$$g(z) = \frac{1}{1+e^{-z}}$$

$\theta^T x$

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

$h_\theta(x)$

$g(z$



Sigmoid function

Logistic function

Parameters θ.

Interpretation of Hypothesis Output

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

Example: If $x = \begin{bmatrix} x_0 \\ x_1 \end{bmatrix} = \begin{bmatrix} 1 \\ \text{tumorSize} \end{bmatrix}$

$$h_\theta(x) = 0.7$$

Tell patient that 70% chance of tumor being malignant.

"probability that y = 1, given x, parameterized by θ"

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

Generates proper probability. Use threshold to select classification.

$h_\theta(x)$

Interpretation of Hypothesis Output

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

Example: If $x = \begin{bmatrix} x_0 \\ x_1 \end{bmatrix} = \begin{bmatrix} 1 \\ \text{tumorSize} \end{bmatrix}$

$$h_\theta(x) = 0.7$$

$y = 1$

Tell patient that 70% chance of tumor being malignant

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

$y = 0$ or 1

"probability that y = 1, given x, parameterized by θ"

$$\rightarrow P(y=0|\cancel{x;\theta}) + \boxed{P(y=1|\cancel{x;\theta})} = 1$$

$$\rightarrow P(y=0|x;\theta) = 1 - \boxed{P(y=1|x;\theta)}$$

Logistic regression decision boundary

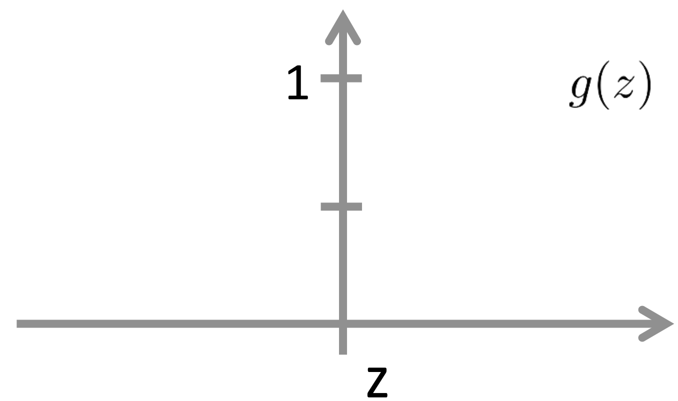
$$h_\theta(x) = g(\theta^T x)$$

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

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

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

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

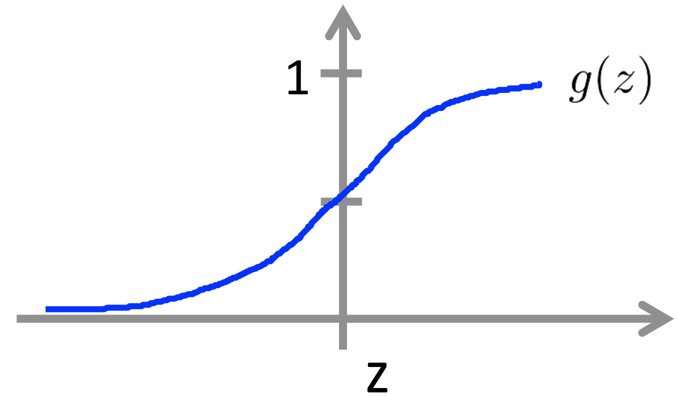


g(z) >= 0.5 when z >=0
g(z) < 0.5 when z < 0

1/(1 + EXP(-(0.0001)))=>0.500025

1/(1 + EXP(-(-0.0001)))=>0.499975

Logistic regression decision boundary



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

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

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

$\theta^T x \geq 0$

$g(z) \geq 0.5$
when $z \geq 0$

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

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

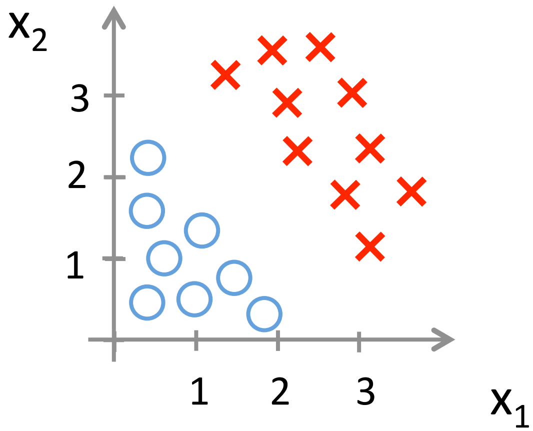
predict "$y = 0$" if $h_\theta(x) < 0.5$

$\theta^T x < 0$

$g(z) < 0.5$
when $z < 0$

>= 0

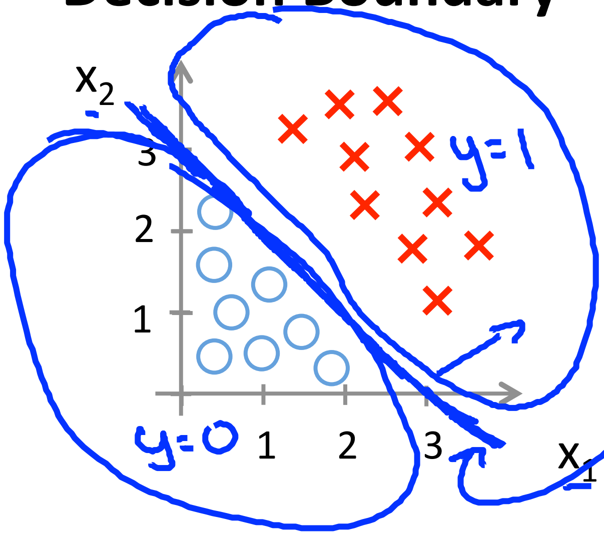
Decision Boundary



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

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

Decision Boundary



$$\theta = \begin{bmatrix} -3 \\ 1 \\ 1 \end{bmatrix}$$

$$\boxed{h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2)}$$

$\theta_0 = -3$, $\theta_1 = 1$, $\theta_2 = 1$

Decision boundary

Predict "$y = 1$" if $\overbrace{-3 + x_1 + x_2}^{} \geq 0$

$\theta^T x$

$\rightarrow x_1 + x_2 \geq 3$

x_1, x_2

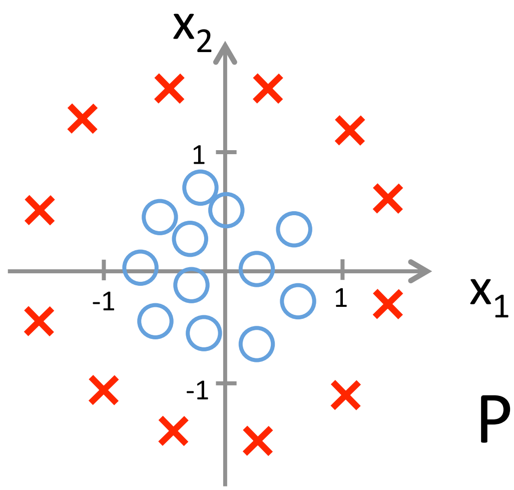
$h_\theta(x) = 0.5$

$\boxed{x_1 + x_2 = 3}$

$x_1 + x_2 < 3$

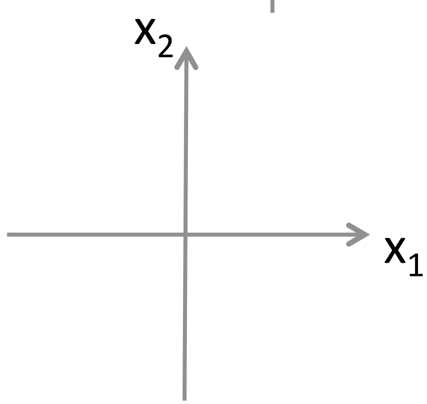
$y = 0$

Non-linear decision boundaries



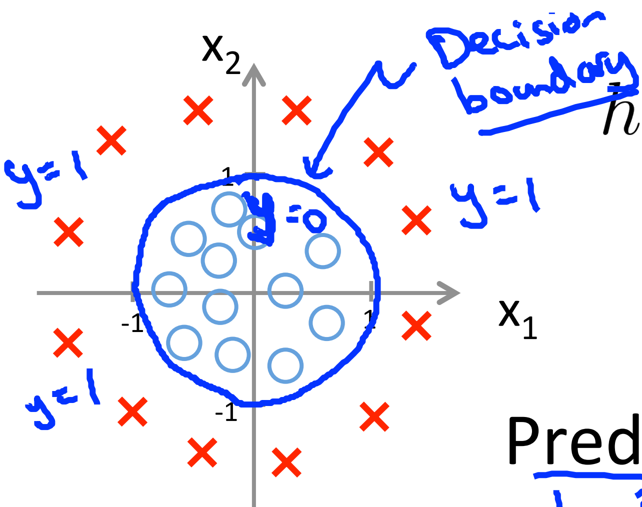
$$h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 + \theta_4 x_2^2)$$

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



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

Non-linear decision boundaries



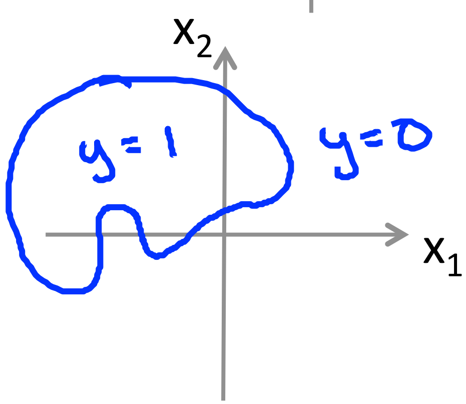
$$h_\theta(x) = g(\overset{=-1}{\theta_0} + \overset{=0}{\theta_1} x_1 + \overset{=0}{\theta_2} x_2 + \underset{=1}{\theta_3} x_1^2 + \underset{=1}{\theta_4} x_2^2)$$

$$\theta = \begin{bmatrix} -1 \\ 0 \\ 0 \\ 1 \\ 1 \end{bmatrix}$$

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

$$\boxed{x_1^2 + x_2^2 = 1}$$

$$x_1^2 + x_2^2 \geq 1$$



$$h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 \underline{x_1^2} + \theta_4 \underline{x_1^2 x_2} + \theta_5 \underline{x_1^2 x_2^2} + \theta_6 \underline{x_1^3 x_2} + \dots)$$

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

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

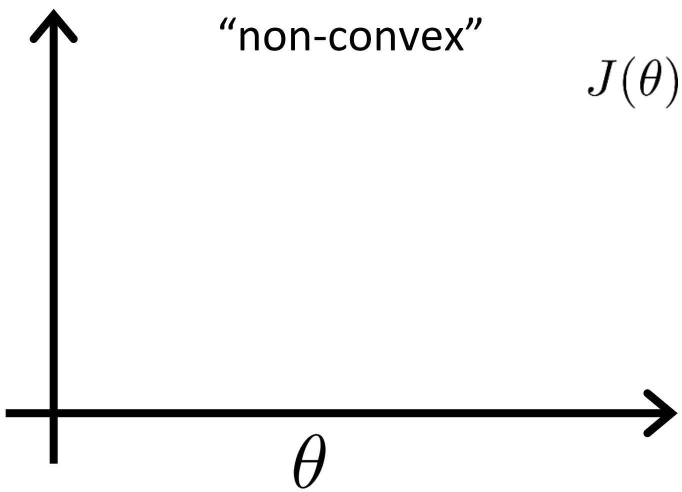
$$h_\theta(x) = \frac{1}{1 + e^{-\theta^T x}}$$

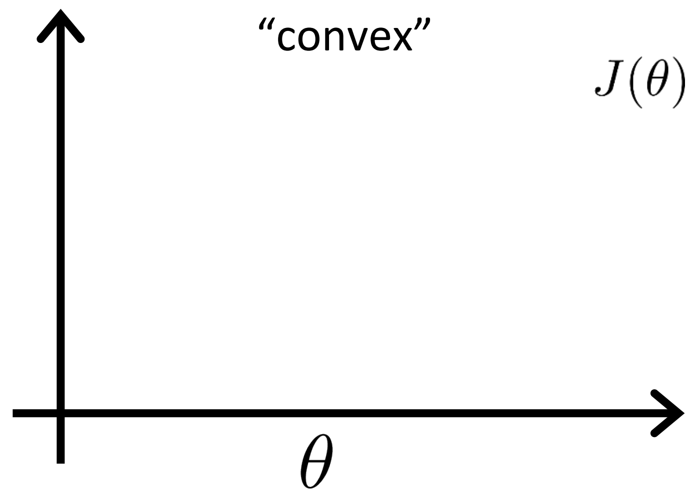
How to choose parameters θ ?

Cost function

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

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





Cost function

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

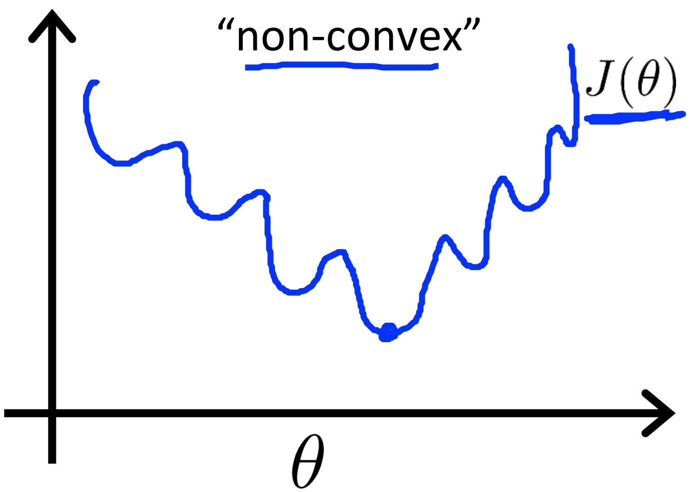
Logistic

$\text{cost}(h_\theta(x^{(i)}), y)$

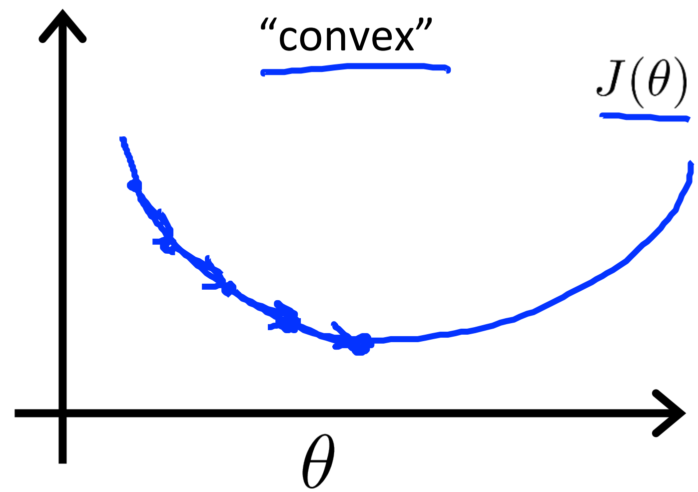
$$\text{Cost}(h_\theta(x^{(i)}), y^{(i)}) = \frac{1}{2}\left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$$

$\frac{1}{1+e^{-\theta^T x}}$

"non-convex"



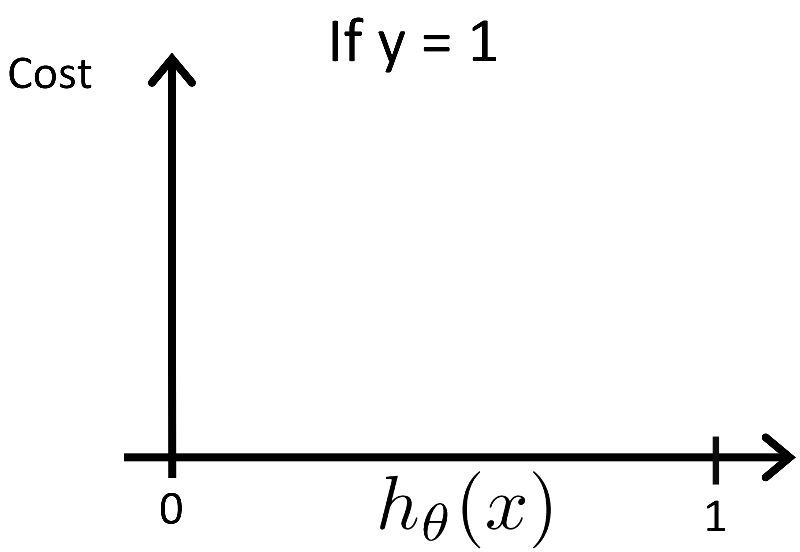
"convex"



Logistic regression cost function

```
>>> -math.log(0.0000001,2)
23.25349666421154
>>> -math.log(1,2)
-0.0
```

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



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

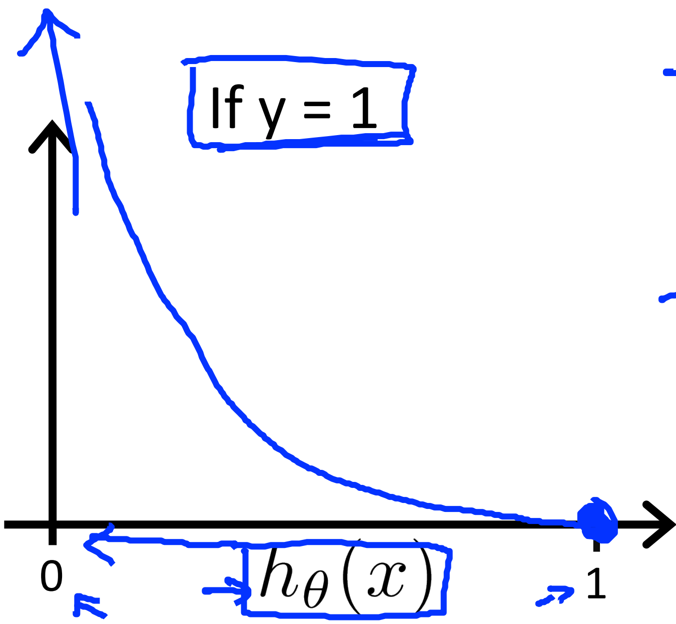
But as $h_\theta(x) \to 0$

$Cost \to \infty$

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

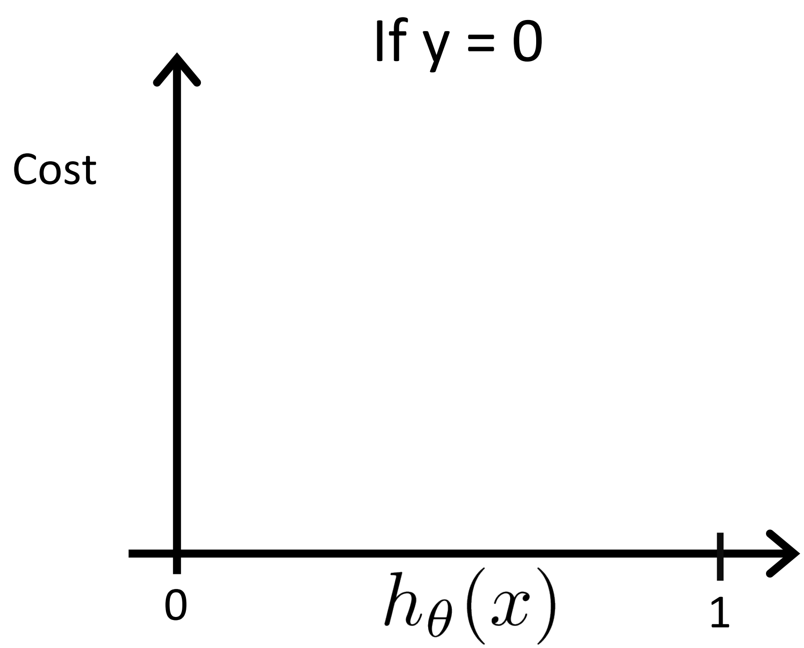
$$\mathrm{Cost}(h_\theta(x), y) = \begin{cases} \boxed{-\log(h_\theta(x))} & \text{if } y = 1 \\ -\log(1 - h_\theta(x)) & \text{if } y = 0 \end{cases}$$



- Cost $= 0$ if $y = 1, h_\theta(x) = 1$
  But as $h_\theta(x) \to 0$
  $Cost \to \infty$
- Captures intuition that if $h_\theta(x) = 0$, (predict $P(y = 1|x; \theta) = 0$), but $\boxed{y = 1}$, we'll penalize learning algorithm by a very large cost.

Logistic regression cost function

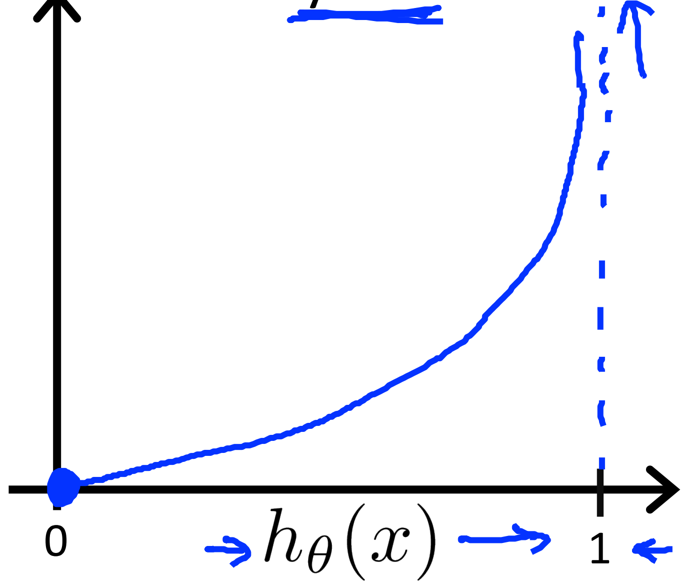
$$\mathrm{Cost}(h_\theta(x), y) = \begin{cases} -\log(h_\theta(x)) & \text{if } y = 1 \\ -\log(1 - h_\theta(x)) & \text{if } y = 0 \end{cases}$$

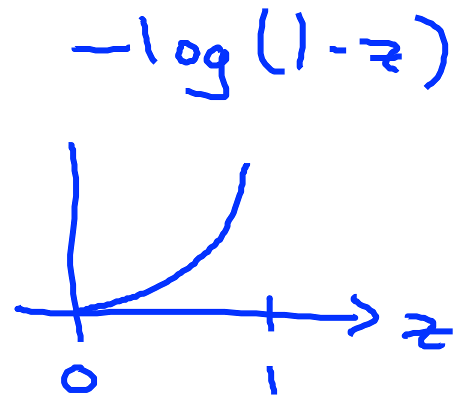


Logistic regression cost function

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

If y = 0





Logistic regression cost function with gradient Descent

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

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

Note: $y = 0$ or 1 always

Logistic regression cost function with gradient Descent

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

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

Note: $y = \boxed{0}$ or $\boxed{1}$ always

$$\rightarrow \text{Cost}(h_\theta(x), y) = -\underbrace{y}_{=1 \rightarrow =0} \log(h_\theta(x)) - \underbrace{(1-y)}_{=0 \rightarrow =1} \log(1 - h_\theta(x))$$

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

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

Logistic regression cost function

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

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

To fit parameters θ :

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

To make a prediction given new x

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

Logistic regression cost function

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

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

To fit parameters θ :

$$\boxed{\min_\theta J(\theta) \quad \text{Get } \theta}$$

To make a prediction given new x:

Output $h_\theta(x) = \frac{1}{1+e^{-\theta^T x}}$ $\qquad p(y=1 \mid x;\theta)$

Gradient Descent

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

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

Repeat $\{$

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

(simultaneously update all θ_j)

$\}$

Gradient Descent

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

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

Repeat $\{$

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

$\}$

(simultaneously update all θ_j)

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

Gradient Descent

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

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

Repeat $\{$

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

(simultaneously update all θ_j)

$\}$

Algorithm looks identical to linear regression!

Gradient Descent

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

$$\Theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \vdots \\ \theta_n \end{bmatrix} \leftarrow$$

for $i = 0$ to n

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

Repeat {

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

(simultaneously update all θ_j)

}

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

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

Algorithm looks identical to linear regression!

Optimization algorithm – Advanced optional material

Cost function $J(\theta)$. Want $\min_\theta J(\theta)$.

$$\theta$$

$$J(\theta)$$

$$\frac{\partial}{\partial\theta_j}J(\theta)$$ (for $j = 0, 1, \ldots, n$)

Gradient descent:

Repeat $\{$

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

$\}$

Optimization algorithm

Given θ, we have code that can compute

- $J(\theta)$
- $\frac{\partial}{\partial\theta_j}J(\theta)$ (for $j = 0, 1, \ldots, n$)

Optimization algorithms:

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

Advantages:

- No need to manually pick α
- Often faster than gradient descent.

Disadvantages:

- More complex

Example:

$$\theta = \begin{bmatrix} \theta_1 \\ \theta_2 \end{bmatrix}$$

$$J(\theta) = (\theta_1 - 5)^2 + (\theta_2 - 5)^2$$

$$\frac{\partial}{\partial \theta_1} J(\theta) = 2(\theta_1 - 5)$$

$$\frac{\partial}{\partial \theta_2} J(\theta) = 2(\theta_2 - 5)$$

```
function [jVal, gradient]
            = costFunction(theta)
  jVal = (theta(1)-5)^2 + ...
         (theta(2)-5)^2;
  gradient = zeros(2,1);
  gradient(1) = 2*(theta(1)-5);
  gradient(2) = 2*(theta(2)-5);
```

```
options = optimset('GradObj', 'on', 'MaxIter', '100');
initialTheta = zeros(2,1);
[optTheta, functionVal, exitFlag] ...
     = fminunc(@costFunction, initialTheta, options);
```

$$\texttt{theta} = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_n \end{bmatrix}$$

```
function [jVal, gradient] = costFunction(theta)
```

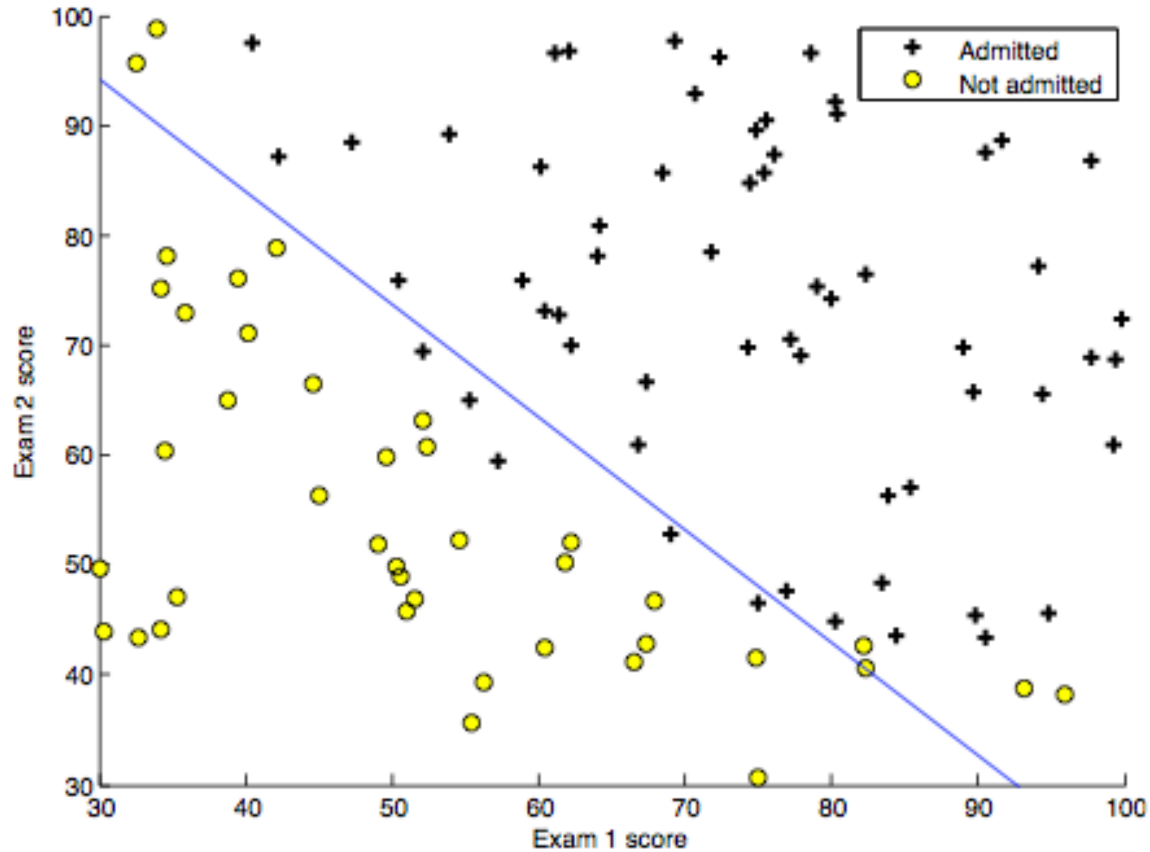
`jVal = [` code to compute $J(\theta)$ `];`

`gradient(1) = [` code to compute $\frac{\partial}{\partial\theta_0}J(\theta)$ `];`

`gradient(2) = [` code to compute $\frac{\partial}{\partial\theta_1}J(\theta)$ `];`

$\vdots$

`gradient(n+1) = [` code to compute $\frac{\partial}{\partial\theta_n}J(\theta)$ `];`

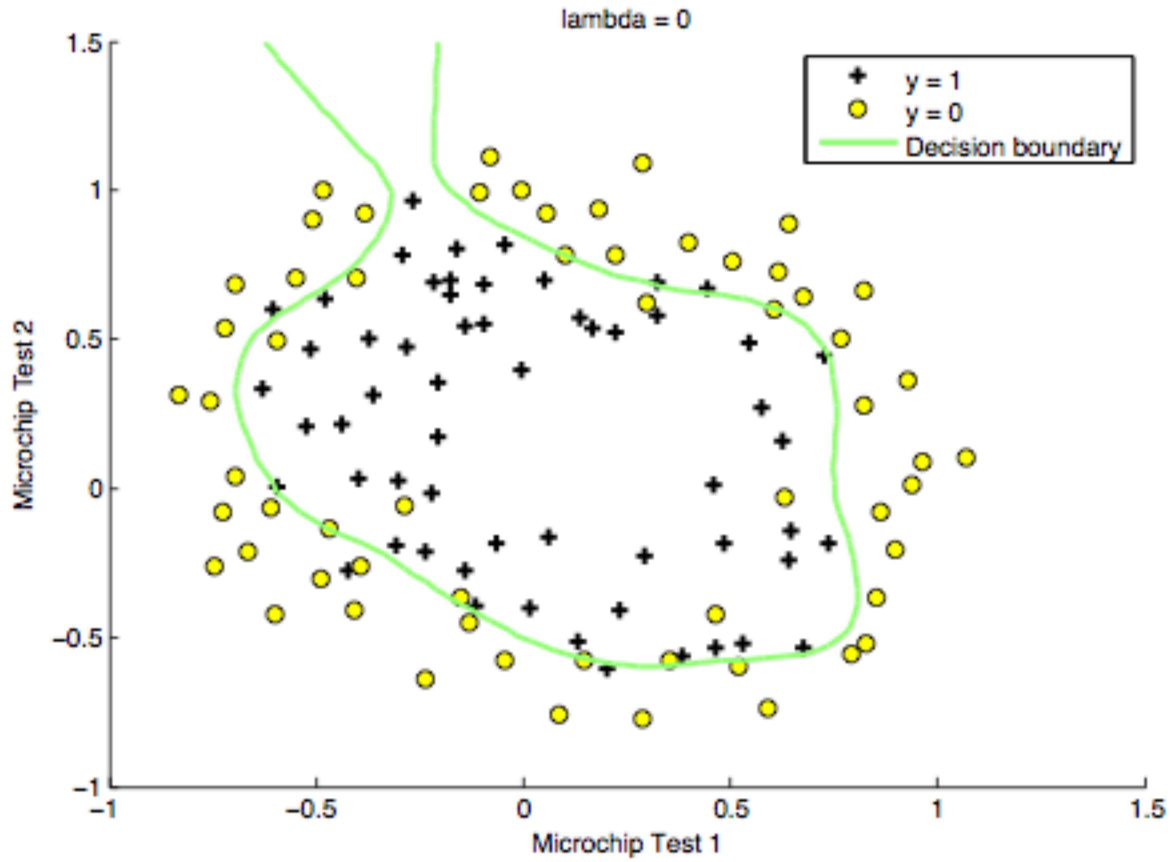

Admitted
Not admitted
Exam 2 score
Exam 1 score
100
90
80
70
60
50
40
30
30
40
50
60
70
80
90
100

Prevent *overfitting* with regularization parameter

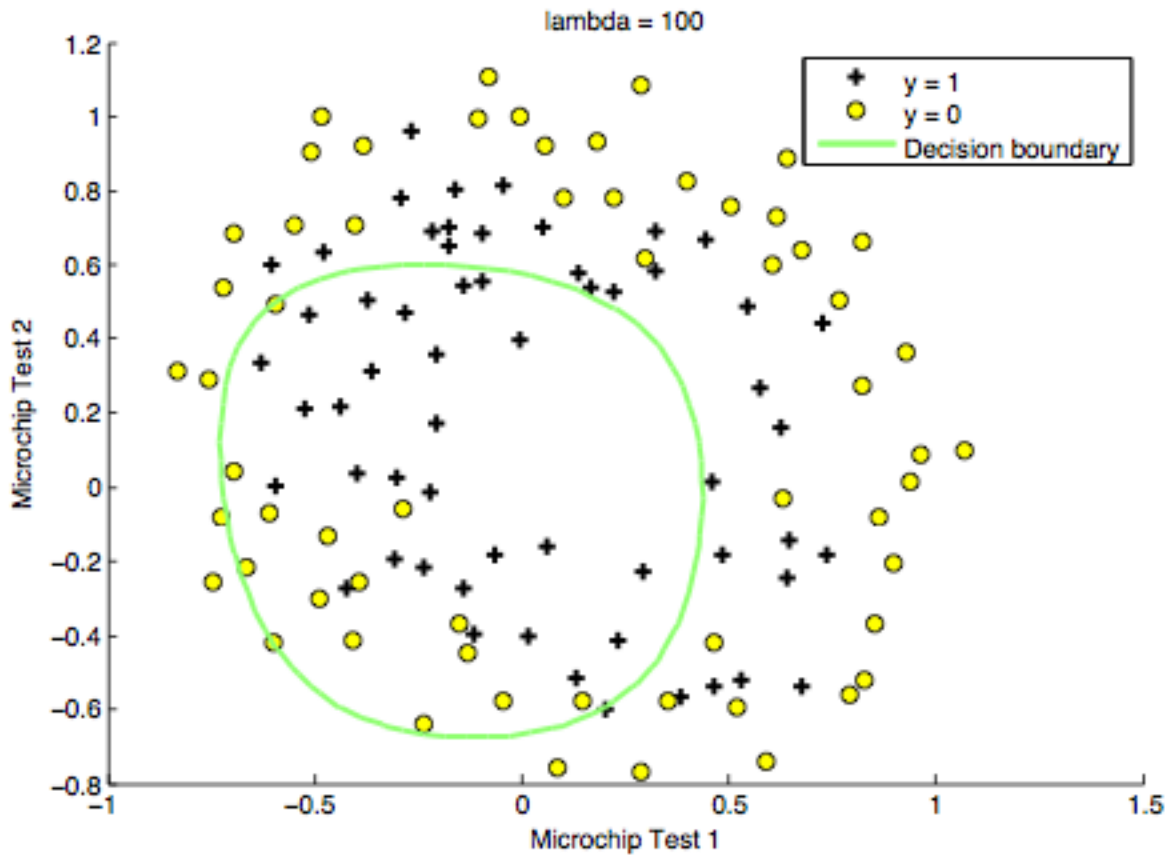
$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m}\left[-y^{(i)}\log(h_\theta(x^{(i)})) - (1-y^{(i)})\log(1-h_\theta(x^{(i)}))\right] + \frac{\lambda}{2m}\sum_{j=1}^{n}\theta_j^2.$$

$$\frac{\partial J(\theta)}{\partial \theta_0} = \frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_j^{(i)} \qquad \text{for } j = 0$$

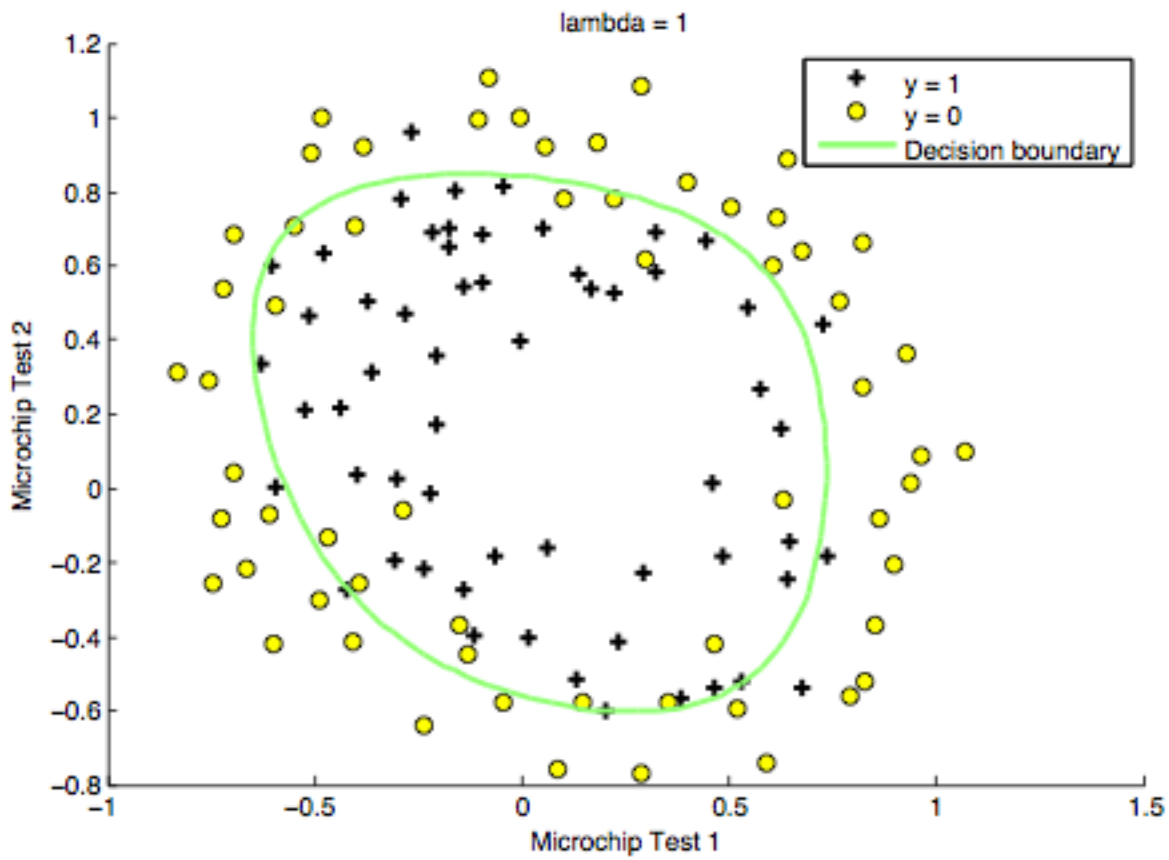
$$\frac{\partial J(\theta)}{\partial \theta_j} = \left(\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_j^{(i)}\right) + \frac{\lambda}{m}\theta_j \qquad \text{for } j \geq 1$$



Overfitting lamda = 0



Under fitting lamda = 100



Correctfitting lamda = 1

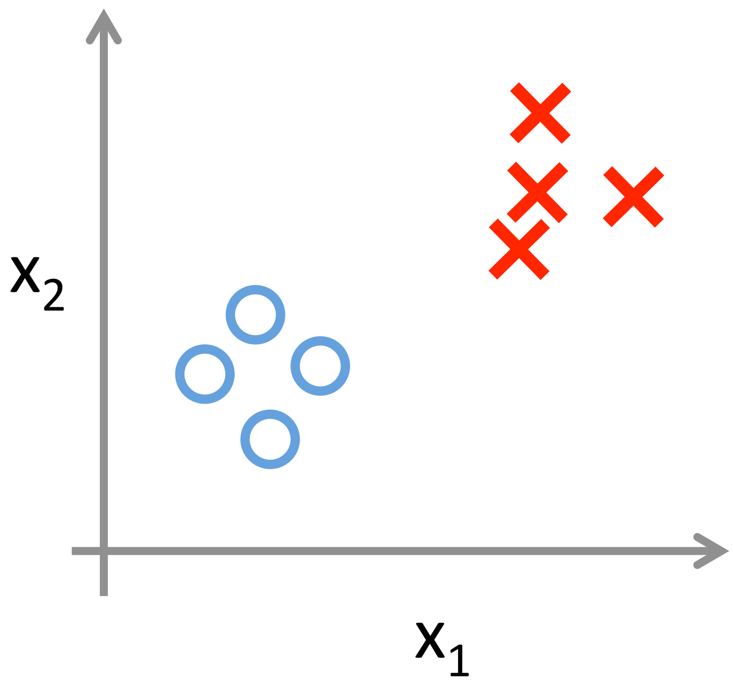
Multiclass classification

Email foldering/tagging: Work, Friends, Family, Hobby

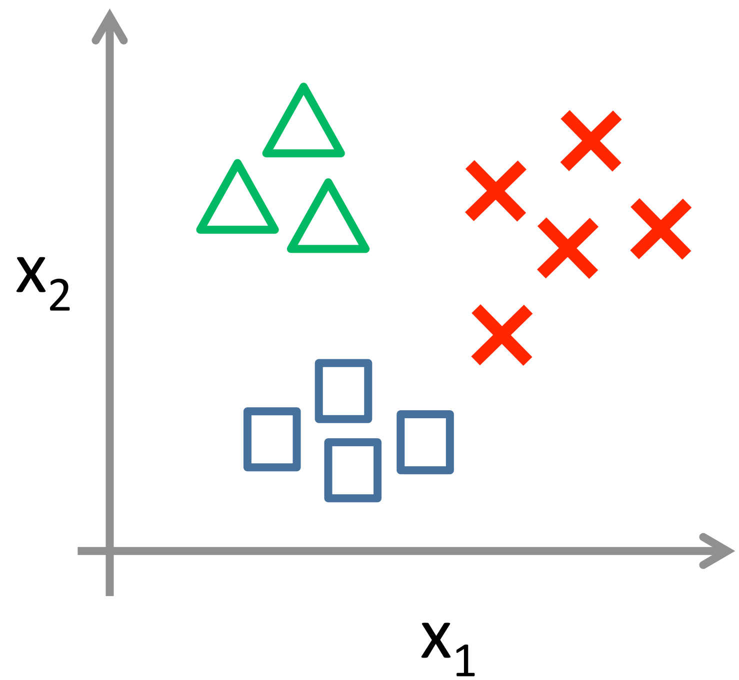
Medical diagrams: Not ill, Cold, Flu

Weather: Sunny, Cloudy, Rain, Snow

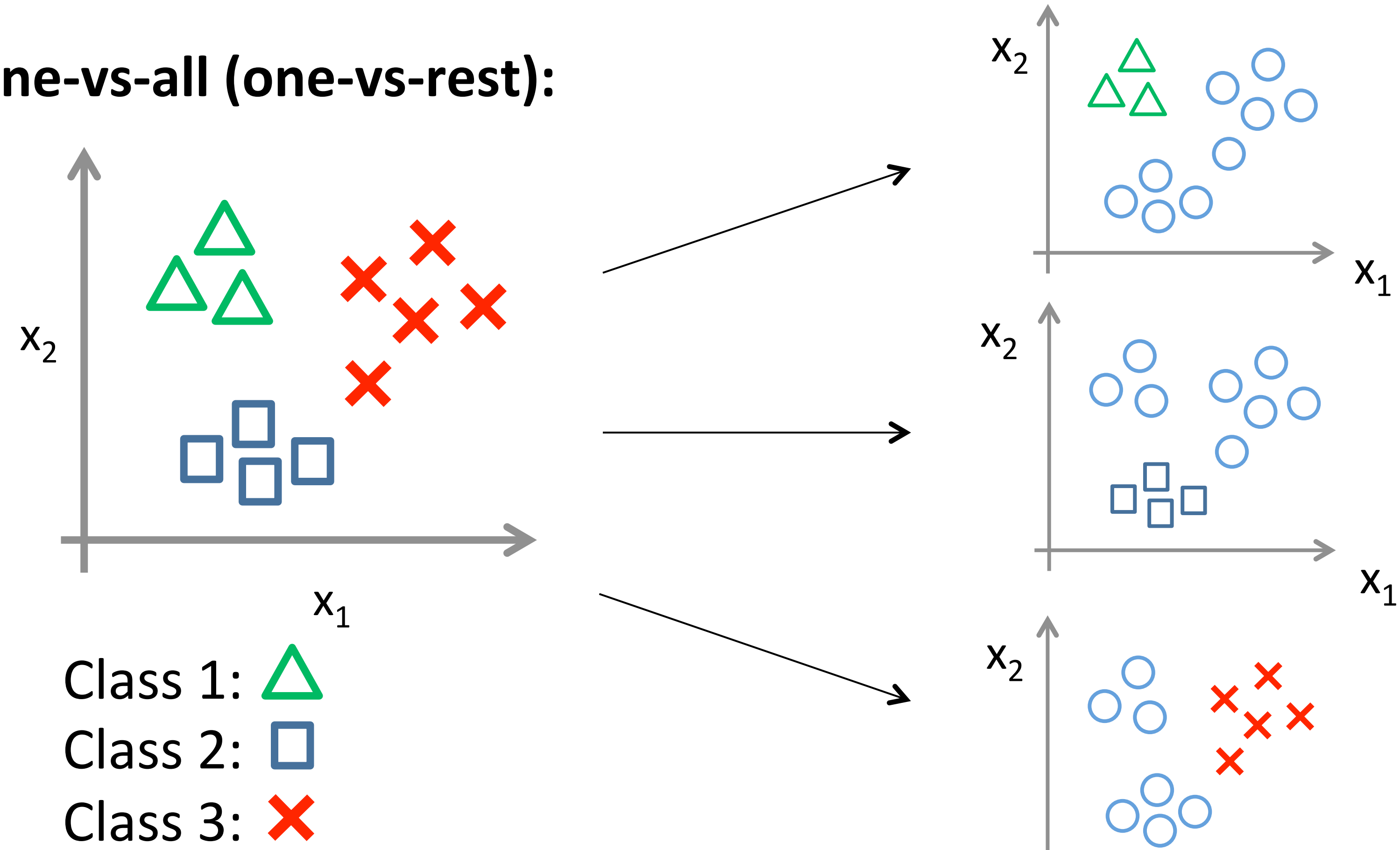
Binary classification:



Multi-class classification:

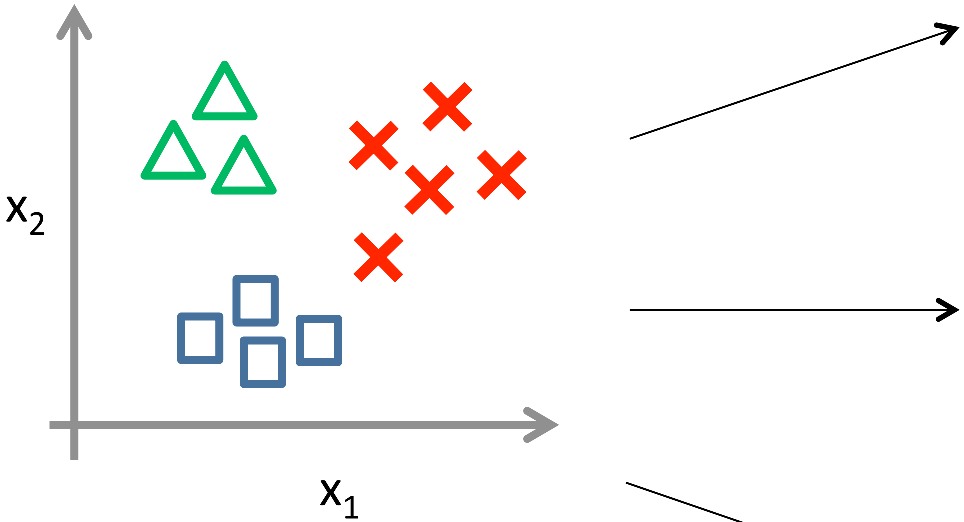


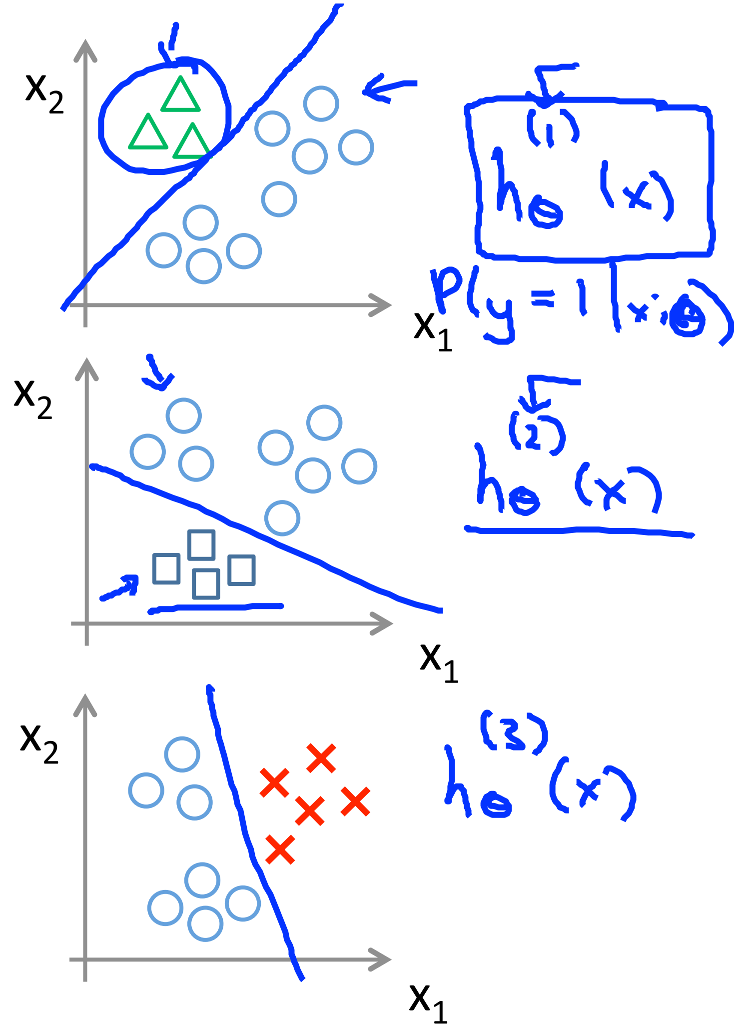
One-vs-all (one-vs-rest):



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

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





Class 1: △

Class 2: □

Class 3: ✖

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

One-vs-all

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

On a new input x, to make a prediction, pick the class i that maximizes

$$\max_i h_\theta^{(i)}(x)$$

Softmax

- Generalization of the logistic function that "squashes" a K-dimensional vector into a range [0,1]
- Softmax can be used in multiclass classification methods.
- In multinomial logistic regression, the input to the function is the result of K distinct linear functions, and the predicted probability for the j'th class given a sample vector x and a weighting vector w is:

$$P(y=j|\mathbf{x}) = \frac{e^{\mathbf{x}^\mathsf{T}\mathbf{w}_j}}{\sum_{k=1}^{K} e^{\mathbf{x}^\mathsf{T}\mathbf{w}_k}}$$