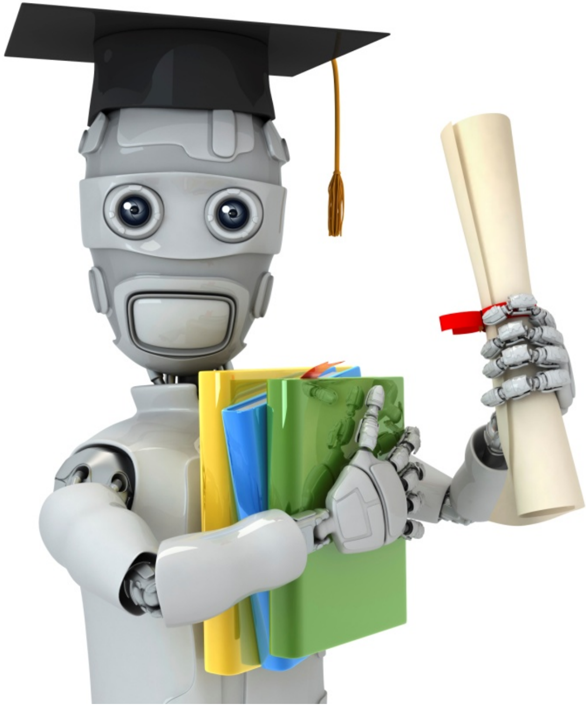


Machine Learning

Logistic Regression

Classification

Classification

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

Binary classification

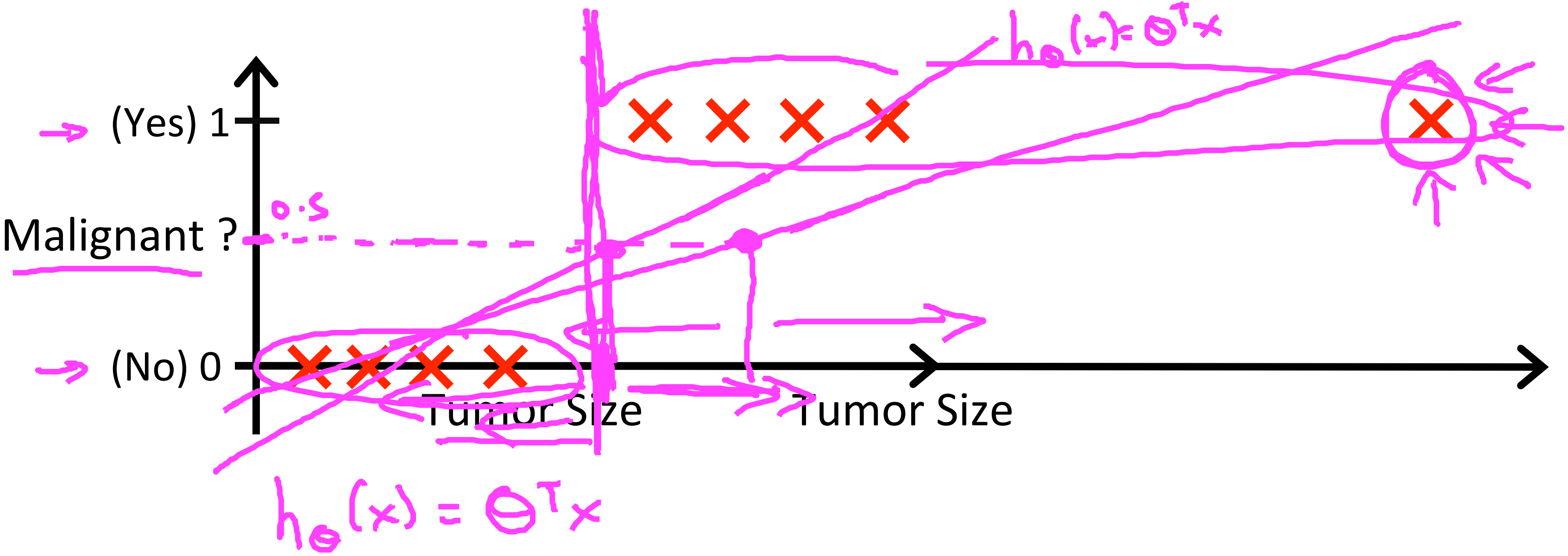
$y \in \{0, 1\}$

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

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

$y \in \{0, 1, 2, 3\}$

Multiclass classification



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

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

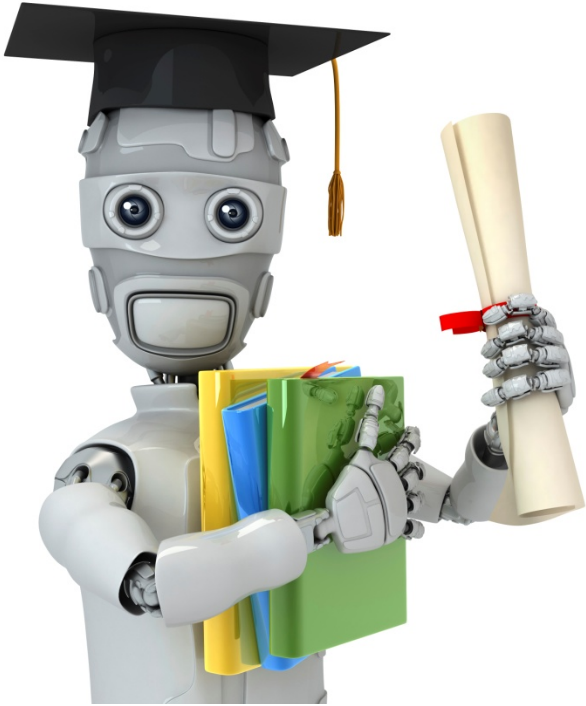
If $h_\theta(x) < 0.5$, predict "y = 0"

Classification: $\boxed{y = 0 \text{ or } 1}$

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

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

Classification



Machine Learning

Logistic Regression

Hypothesis Representation

Logistic Regression Model

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

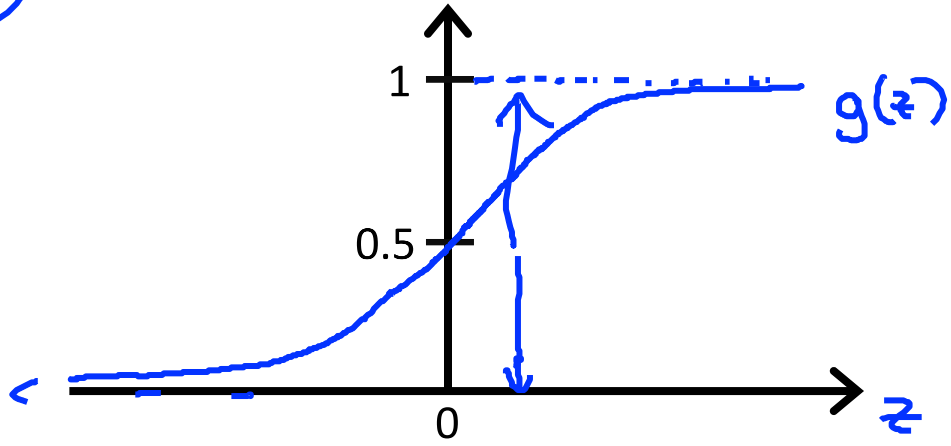
$$h_\theta(x) = g(\theta^T x)$$

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

z: real number

$\theta^T x$

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



Sigmoid function

Logistic function

Parameters θ.

$h_\theta(x)$

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

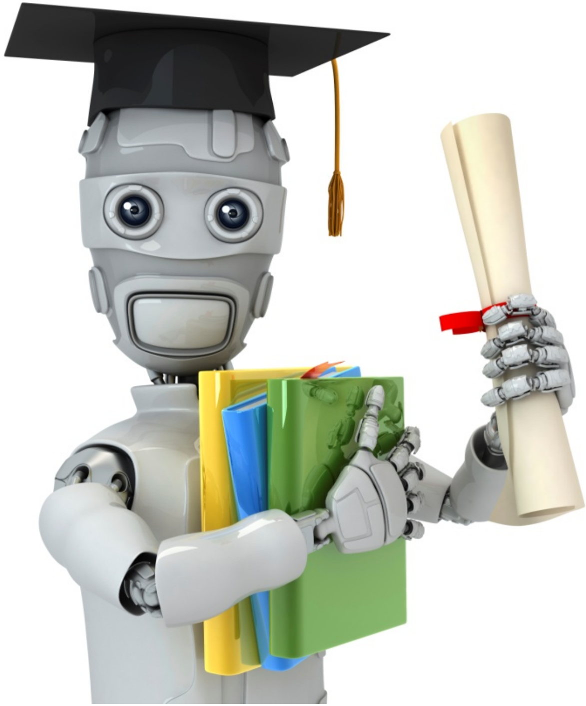
$h_\theta(x) = P(y=1|x;\theta)$

$y = 0$ or 1

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

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

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



Machine Learning

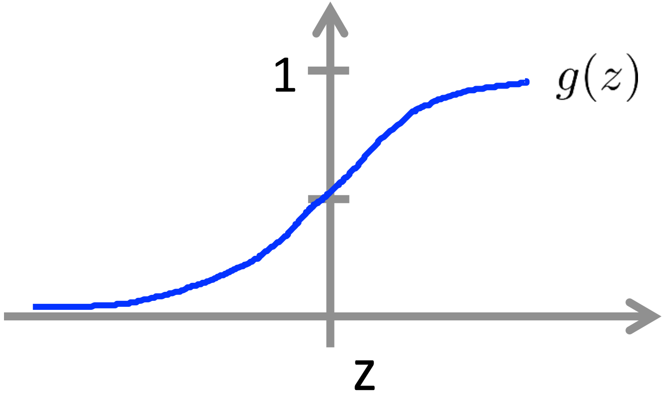
Logistic Regression

Decision boundary

Logistic regression

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

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



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

$\theta^T x \geq 0$

$g(z) \geq 0.5$
when $z \geq 0$
$h_\theta(x) = g(\theta^T x$

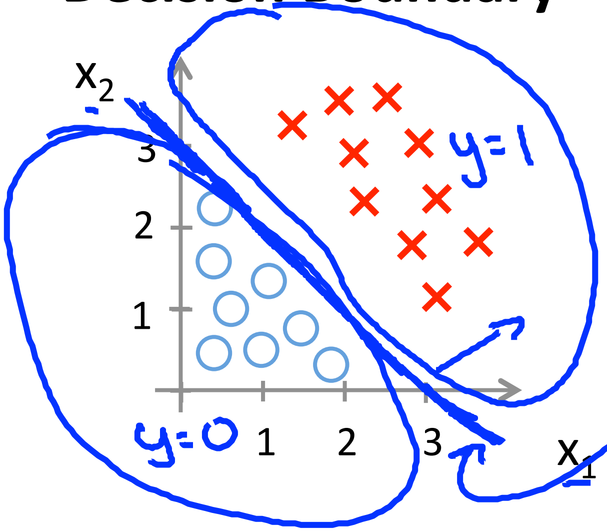
predict "$y = 0$" if $h_\theta(x) < 0.5$

$\theta^T x < 0$

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

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 $\underbrace{-3 + x_1 + x_2}_{\theta^T x} \geq 0$

$x_1 + x_2 \geq 3$

x_1, x_2

$h_\theta(x) = 0.5$

$\boxed{x_1 + x_2 = 3}$

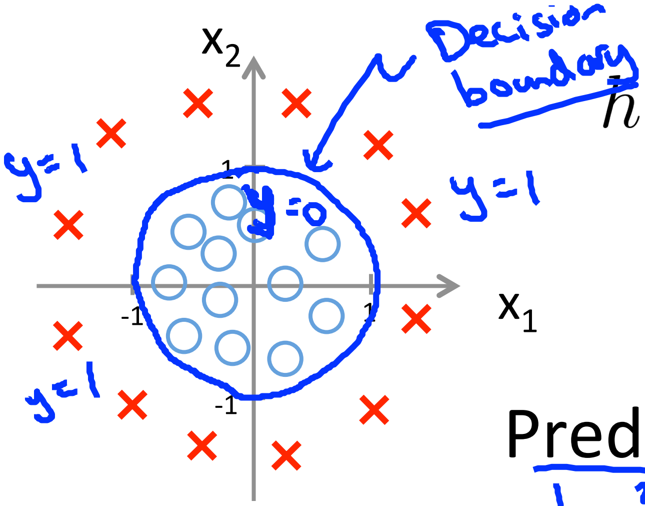
Straight line

$x_1 + x_2 < 3$

$y = 0$

The input to the sigmoid function g(z) doesn't need to be linear, and could be a function that describes a circle or more complex shape

Non-linear decision boundaries

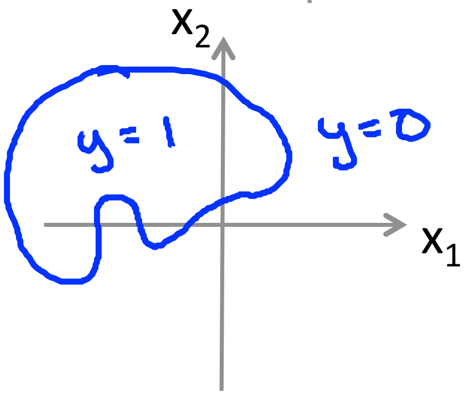


$$h_\theta(x) = g(\overset{=-1}{\theta_0} + \overset{=0}{\theta_1} x_1 + \overset{=0}{\theta_2} x_2 + \underset{=1}{\theta_3} x_1^2 + \underset{=1}{\theta_4} x_2^2)$$

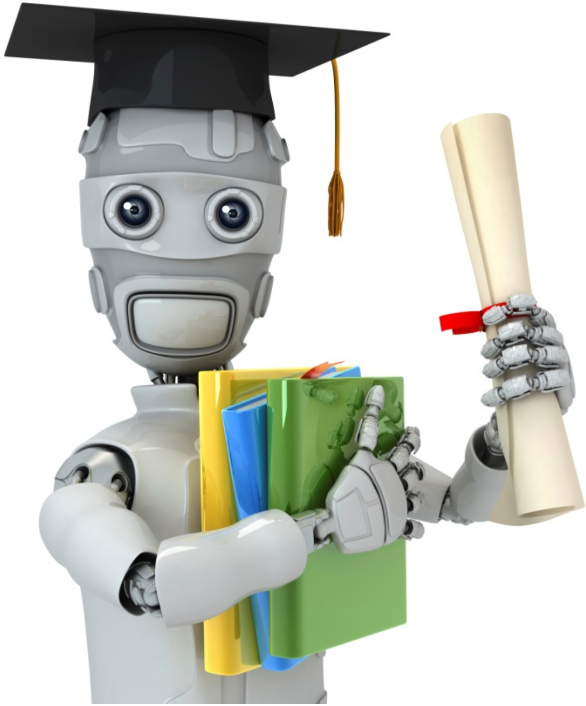
$$\theta = \begin{bmatrix}-1\\0\\0\\1\\1\end{bmatrix}$$

Predict "$y = 1$" if $\underbrace{-1 + x_1^2 + x_2^2}_{x_1^2 + x_2^2 \ge 1} \ge 0$

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



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



Machine Learning

Logistic Regression

Cost function

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

m examples

$$\underbrace{x \in \begin{bmatrix} x_0 \\ x_1 \\ \dots \\ x_n \end{bmatrix}} \quad \mathbb{R}^{n+1} \qquad \underline{x_0 = 1}, \underline{y \in \{0, 1\}}$$

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

How to choose parameters θ ?

Cost function

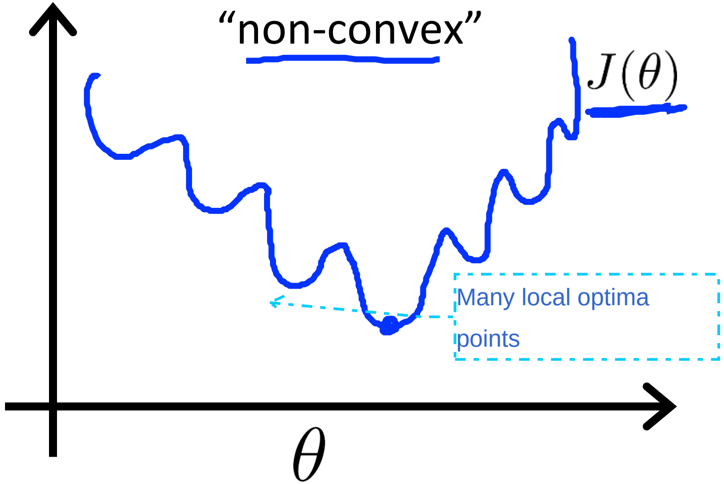
→ ~~Linear~~ regression: $J(\theta) = \frac{1}{m}\sum_{i=1}^{m} \cancel{\frac{1}{2}\left(h_\theta(x^{(i)}) - y^{(i)}\right)^2}$

Logistic

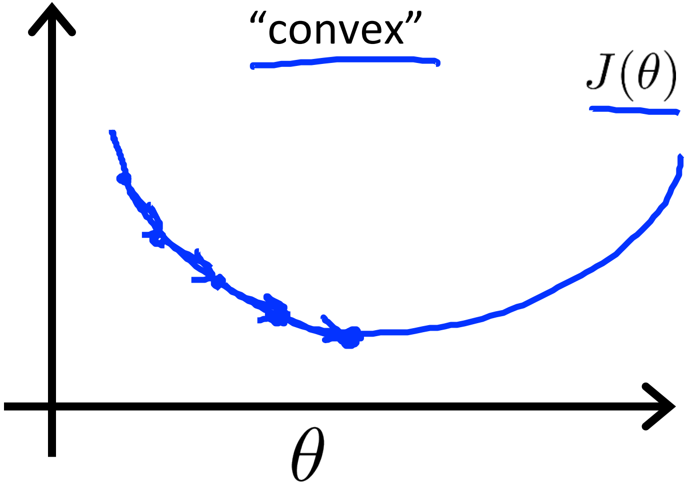
$\to \mathrm{cost}(h_\theta(x^{(i)}), y^{(i)})$

$$\mathrm{Cost}(h_\theta(x), y) = \frac{1}{2}\left(h_\theta(x) - y\right)^2 \leftarrow \boxed{\frac{1}{1+e^{-\theta^T x}}}$$

“non-convex”

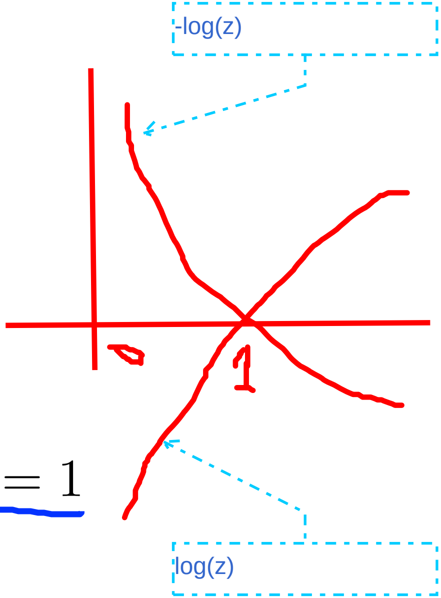


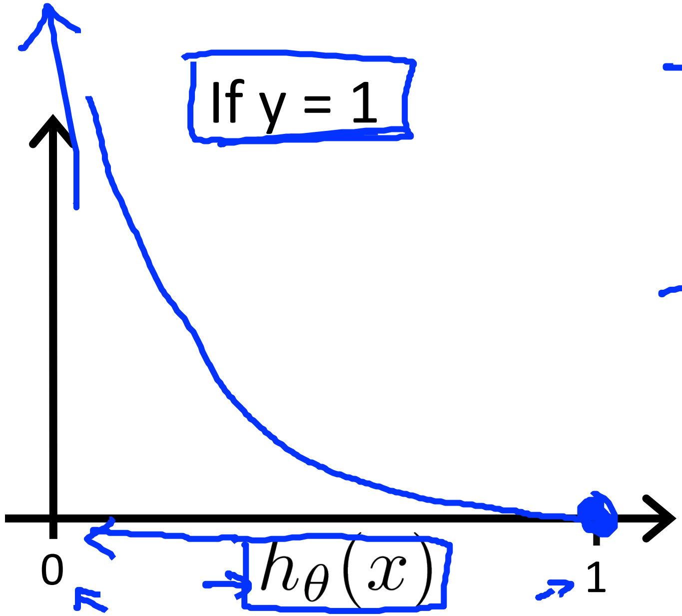
“convex”



Logistic regression cost function

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





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

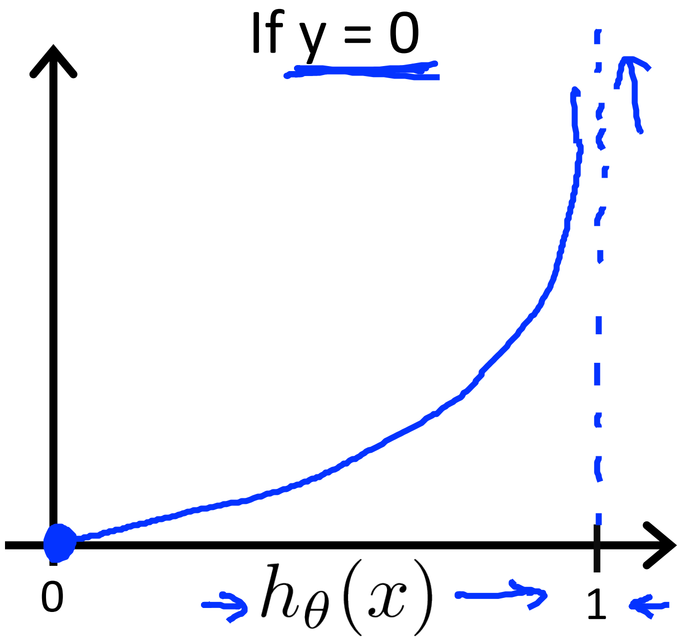
But as $h_\theta(x) \rightarrow 0$

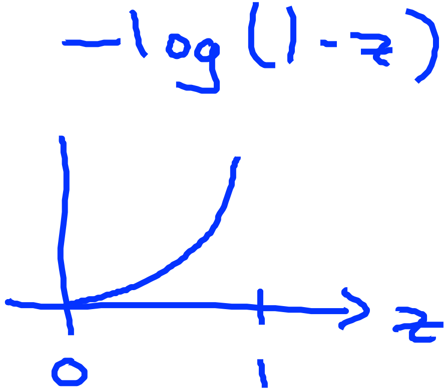
$Cost \rightarrow \infty$

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

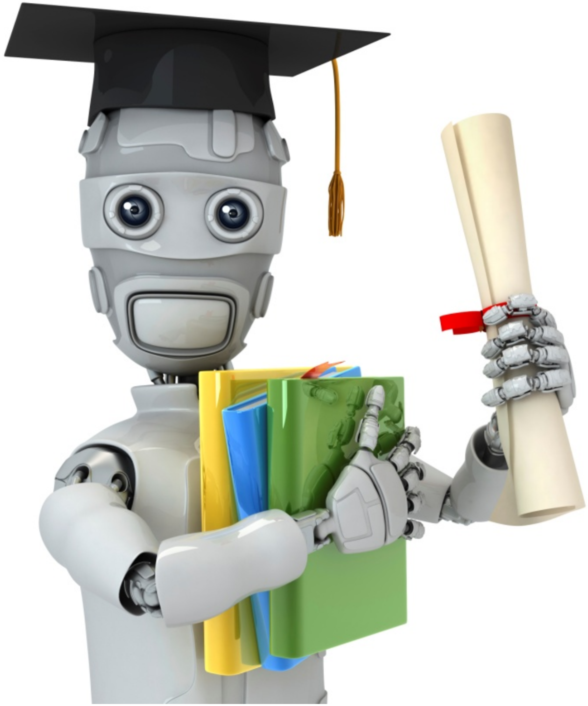
$$\mathrm{Cost}(h_\theta(x), y) = \begin{cases} -\log(h_\theta(x)) & \text{if } y = 1 \\ \boxed{-\log(1 - h_\theta(x))} & \text{if } y = 0 \end{cases}$$





If our correct answer 'y' is 0, then the cost function will be 0 if our hypothesis function also outputs 0. If our hypothesis approaches 1, then the cost function will approach infinity.

If our correct answer 'y' is 1, then the cost function will be 0 if our hypothesis function outputs 1. If our hypothesis approaches 0, then the cost function will approach infinity.



Logistic Regression

Simplified cost function and gradient descent

Machine Learning

Logistic regression cost function

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

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

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

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

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

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

Logistic regression cost function

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

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

Derived equation from statistics using principle of maximum likehood estimation for efficiently finding parameters

Also is convex

To fit parameters θ :

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

To make a prediction given new x :

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

$p(y=1 \mid x ; \theta)$

Gradient Descent

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

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

Repeat $\{$

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

$\}$ (simultaneously update all θ_j)

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

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

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

for $i = 0$ to n

Repeat {

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

(simultaneously update all θ_j)

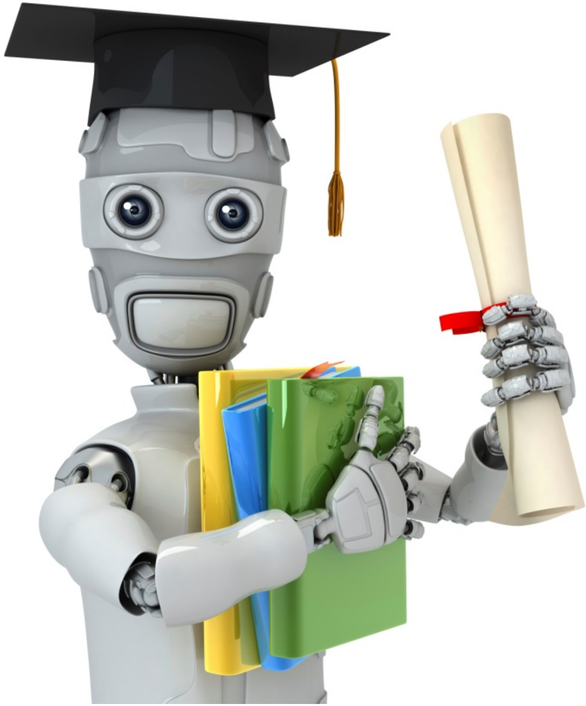
}

For linear regression

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

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

Algorithm looks identical to linear regression!



Machine Learning

Logistic Regression

Advanced optimization

Optimization algorithm

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

Given θ, we have code that can compute

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

Gradient descent:

Repeat $\{$

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

$\}$

Andrew Ng

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

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

$\theta_1 = 5, \theta_2 = 5.$

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

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

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

```
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);
```

Exit flag ==> whether the function converged or not

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

$\theta \in \mathbb{R}^d \quad d \geq 2.$

@ ==> Pointer to function

Andrew Ng

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

theta(1)
theta(2)
theta(n+1)

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

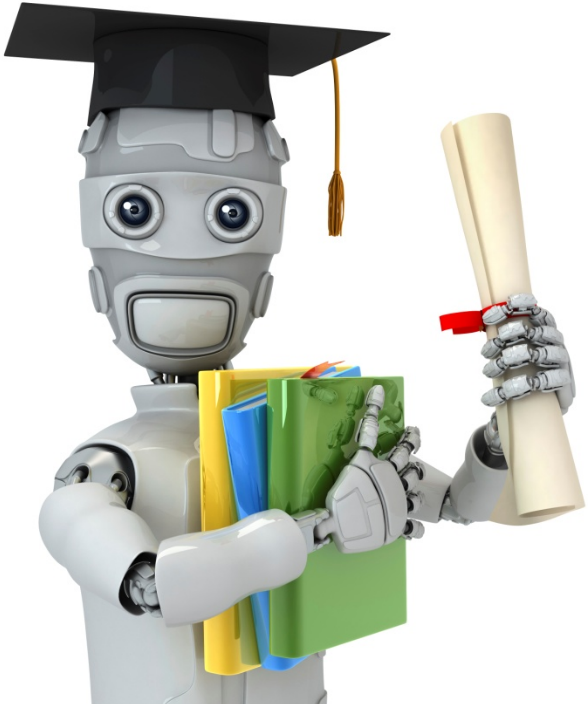
jVal = [$\boxed{\text{code to compute } J(\theta)}$];

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

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

$\vdots$

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



Machine Learning

Logistic Regression

Multi-class classification: One-vs-all

Multiclass classification

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

$y=1$ $\quad y=2$ $\quad y=3$ $\quad y=4$

Medical diagrams: Not ill, Cold, Flu

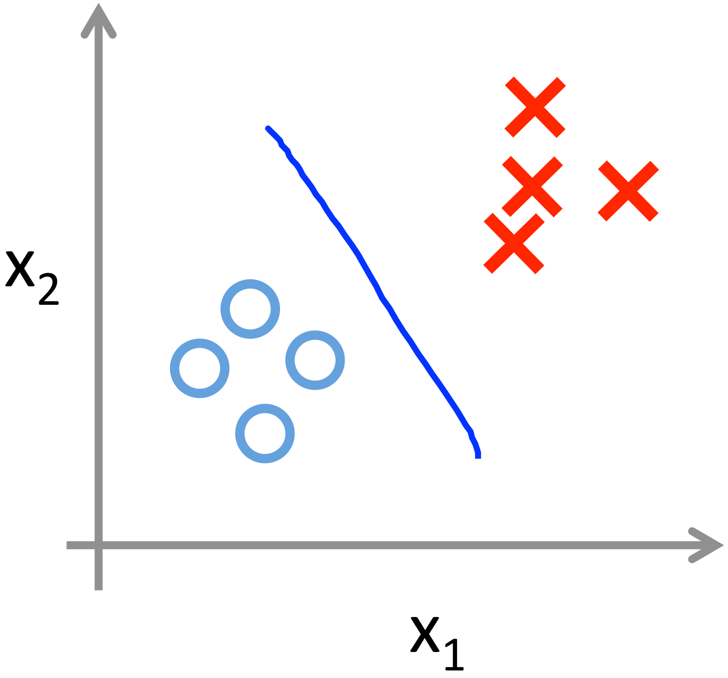
$y=1$ $\quad 2$ $\quad 3$

Weather: Sunny, Cloudy, Rain, Snow

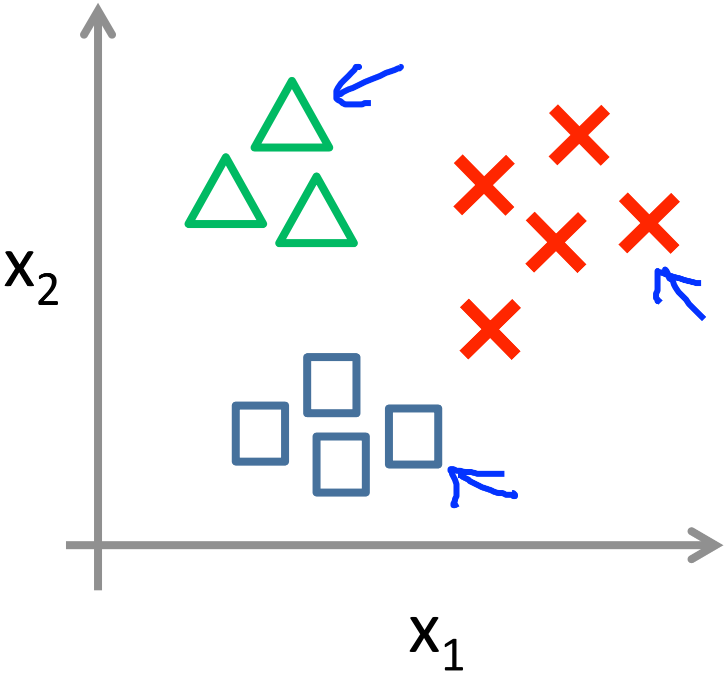
$y=1$ $\quad 2$ $\quad 3$ $\quad 4$

0 $\quad$ 1 $\quad$ 2 $\quad$ 3

Binary classification:

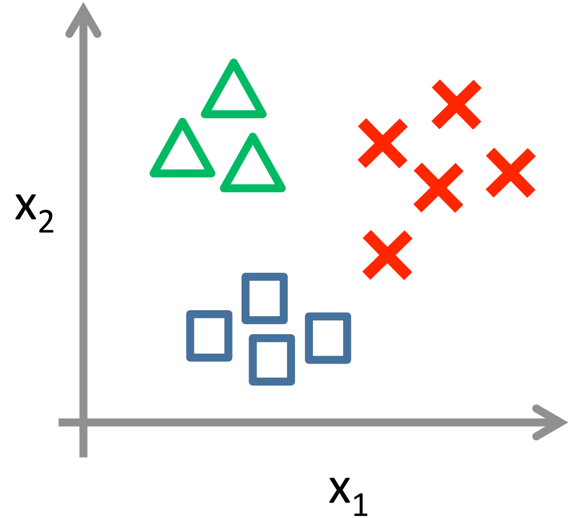


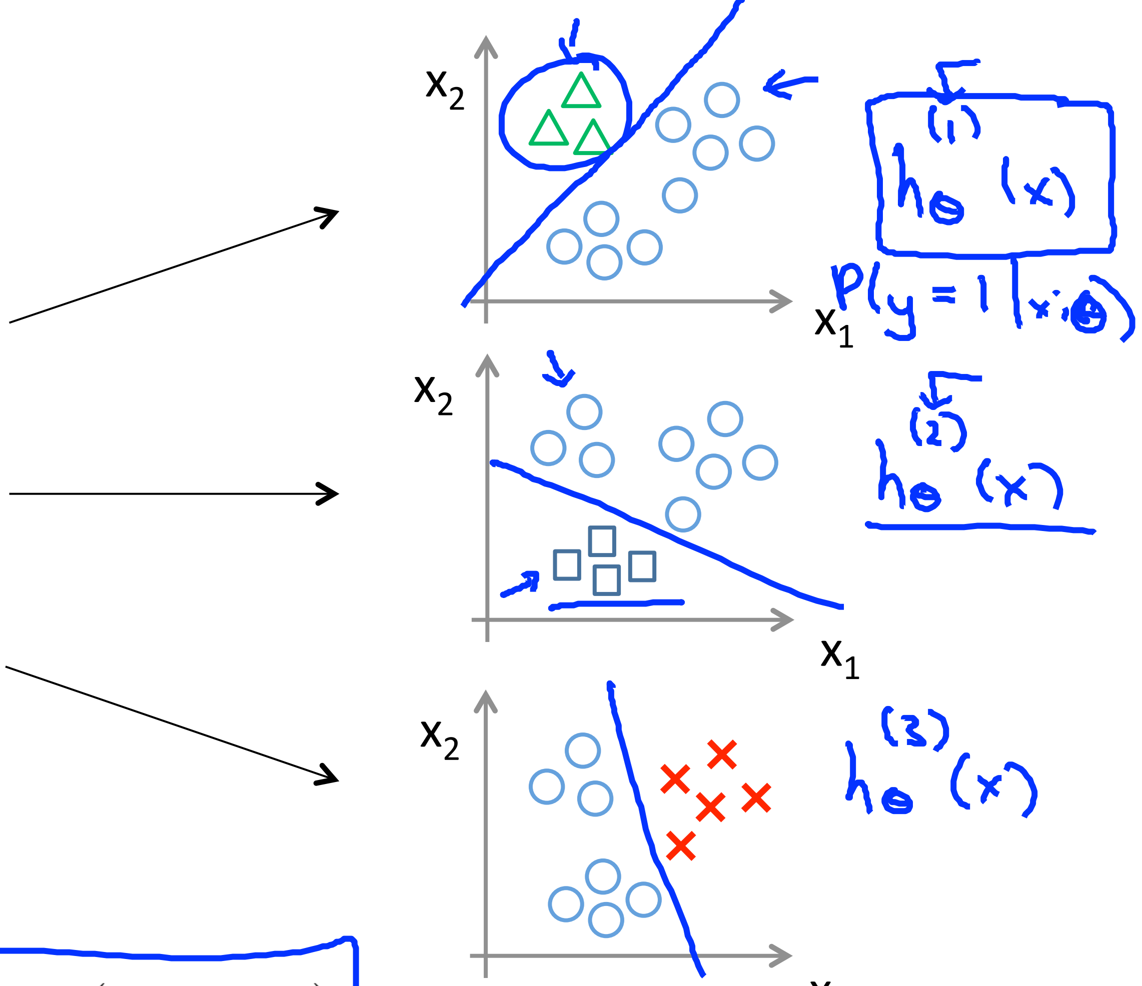
Multi-class classification:



Andrew Ng

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

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

We are basically choosing one class and then lumping all the others into a single second class. We do this repeatedly, applying binary logistic regression to each case, and then use the hypothesis that returned the highest value as our prediction.