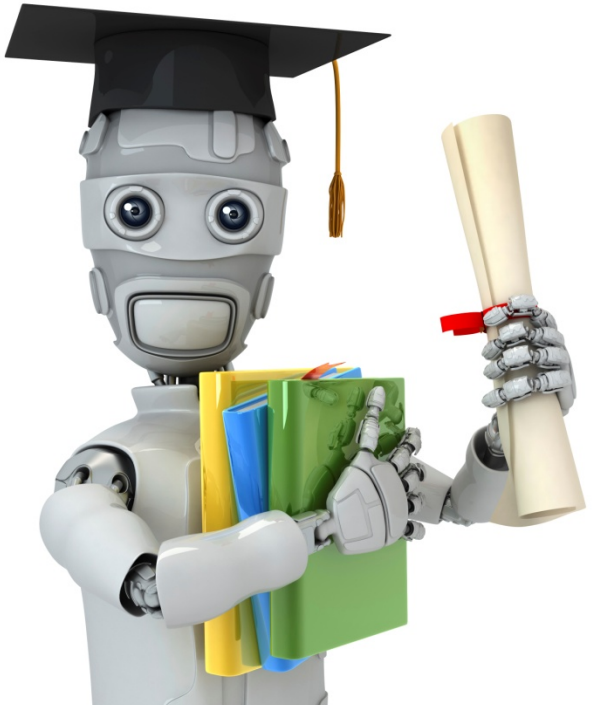


Machine Learning

Logistic Regression

one of the most popular and widely used learning algorithm today.

Classification

output y is discrete value

Classification examples

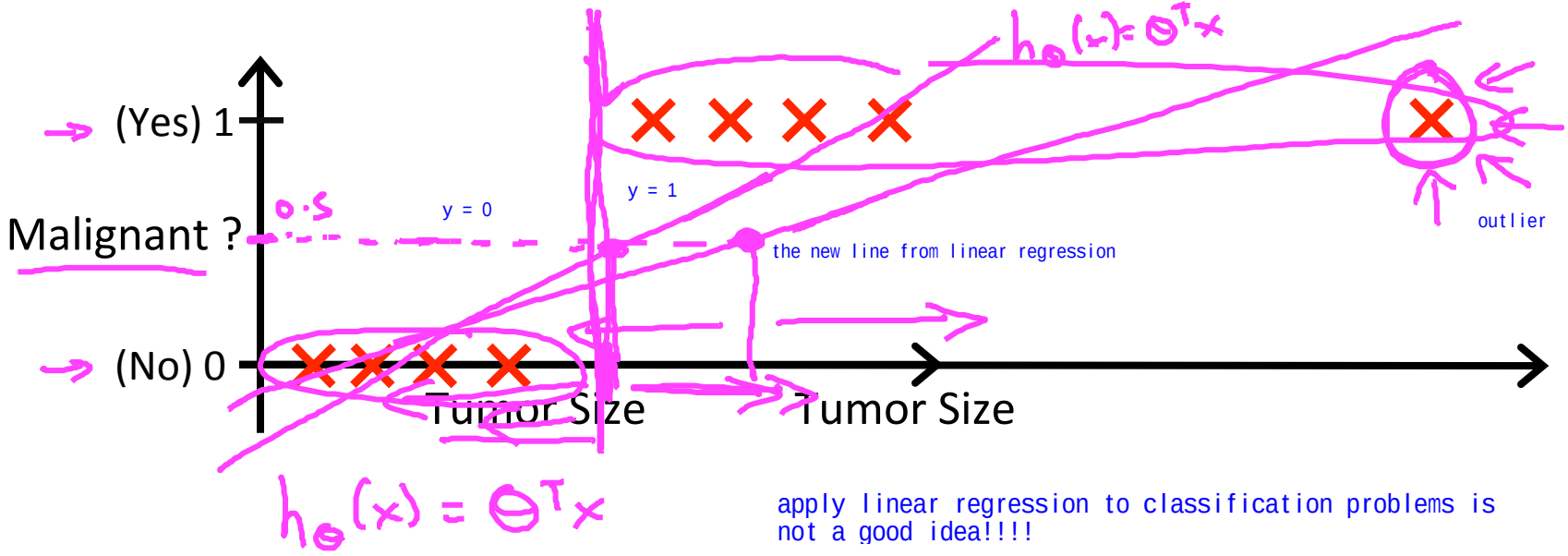
- Email: Spam / Not Spam?
- Online Transactions: Fraudulent (Yes / No)?
- Tumor: Malignant / Benign ?

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

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

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

$y \in \{0, 1, 2, 3\}$ later we will talk about multi-classification problem`



one method to do this using linear regression

- 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"

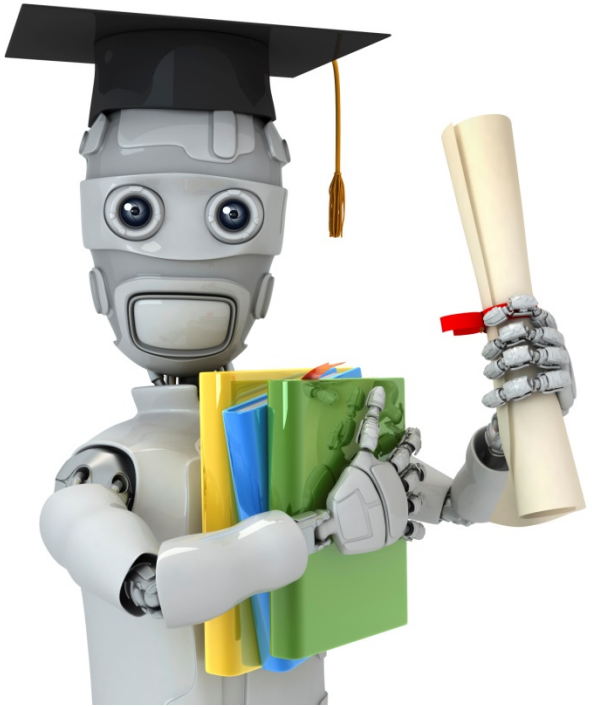
Classification: $y = 0$ or 1

$h_\theta(x)$ can be > 1 or < 0 if we use linear regression method

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

logistic regression has this property
output is bounded by [0,1]

Classification



Machine Learning

Logistic Regression

Hypothesis Representation

we want our function to satisfy this property

Logistic Regression Model

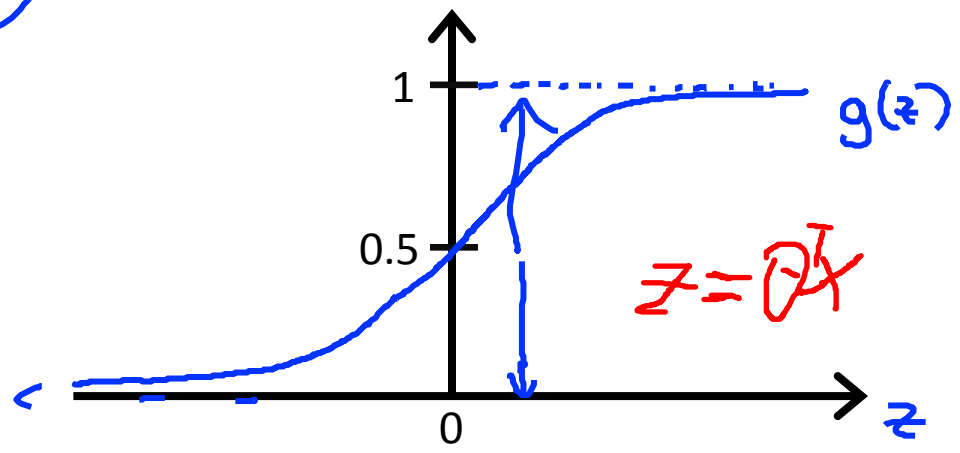
Want $0 \leq h_\theta(x) \leq 1$

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

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

$z = \theta^T x$

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



Sigmoid function

Logistic function

alternative names

Parameters θ - objective: fitting parameter theta

Interpretation of Hypothesis Output

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

hypothesis function

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

$h_\theta(x) = 0.7$

interpretation of the output value

Tell patient that 70% chance of tumor being malignant

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

written formally

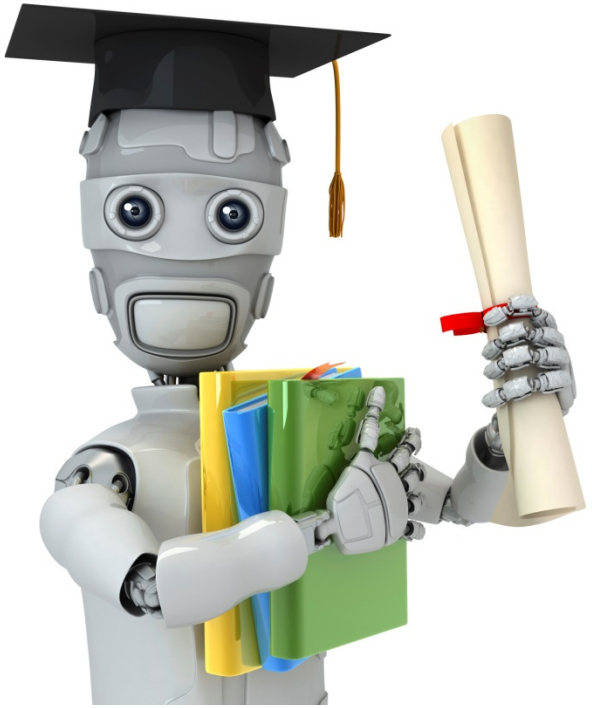
"probability that y = 1, given x, parameterized by θ"

y = 0 or 1

only value y can take is 1 or 0

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

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



Machine Learning

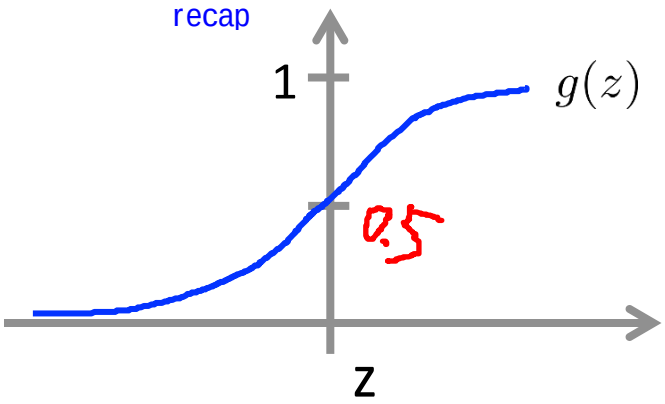
Logistic Regression

Decision boundary

Logistic regression

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

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



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

$\theta^T x \geq 0$

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

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

whenever theta'*x >=0

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

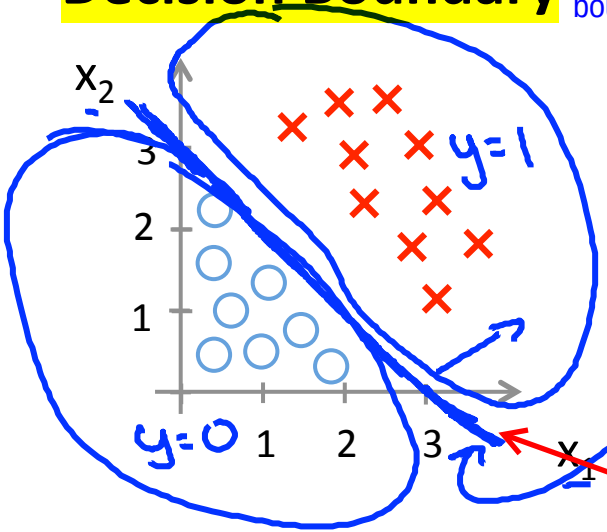
$\theta^T x < 0$

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

Decision Boundary

linear decision boundary

suppose we choose this parameter set $\theta = \begin{bmatrix} -3 \\ 1 \\ 1 \end{bmatrix}$



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

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

Decision boundary

This line is called the decision boundary

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

$\theta^T x$

$x_1 + x_2 \geq 3$

x_1, x_2

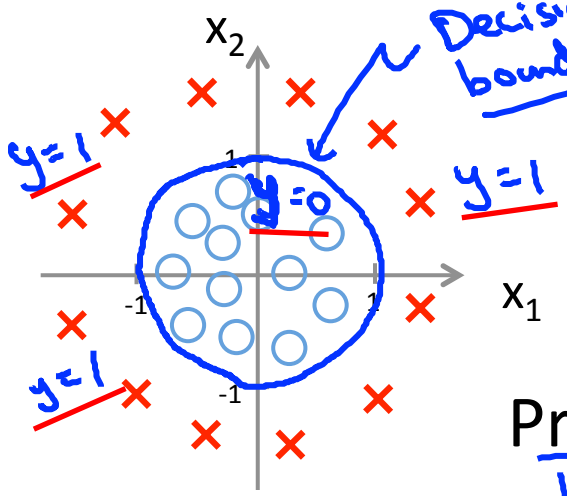
$h_\theta(x) = 0.5$

$x_1 + x_2 = 3$

$x_1 + x_2 < 3$

$y = 0$

Non-linear decision boundaries



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

same as the polynomial fit

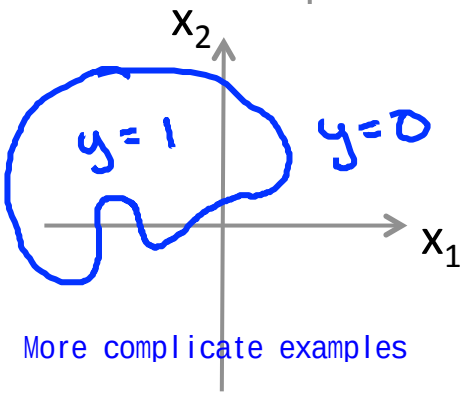
$\theta = \begin{bmatrix} -1 \\ 0 \\ 0 \\ 1 \\ 1 \end{bmatrix}$

suppose we have this data set

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

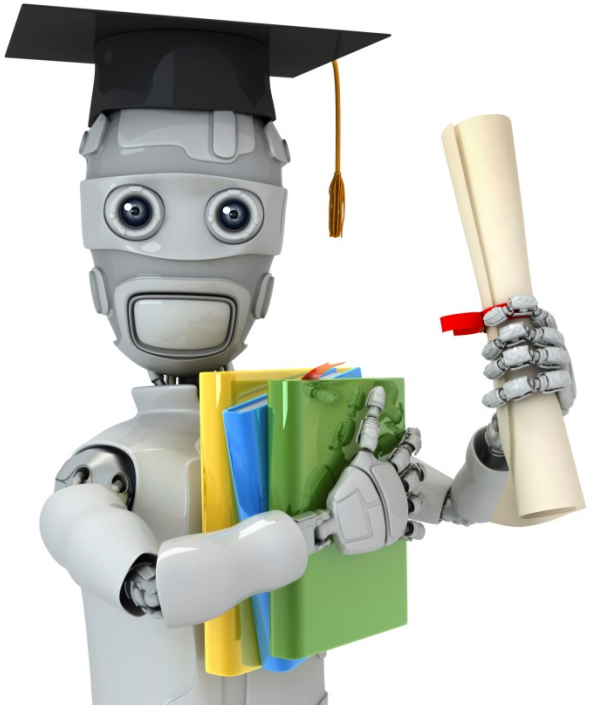
$x_1^2 + x_2^2 = 1$

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



More complicate examples

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



Machine Learning

Logistic Regression

Cost function

Training set:

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

m examples

$$x \in \begin{bmatrix} x_0 \\ x_1 \\ \cdots \\ x_n \end{bmatrix}$$

$\mathbb{R}^{n+1}$

n features

$x_0 = 1, y \in \{0, 1\}$

as usual

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

my hypothesis

How to choose parameters θ ?

Cost function

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

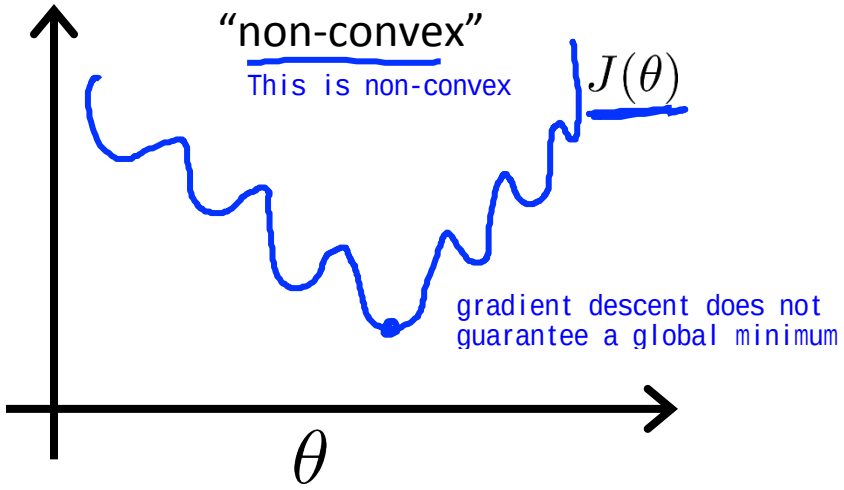
Logistic

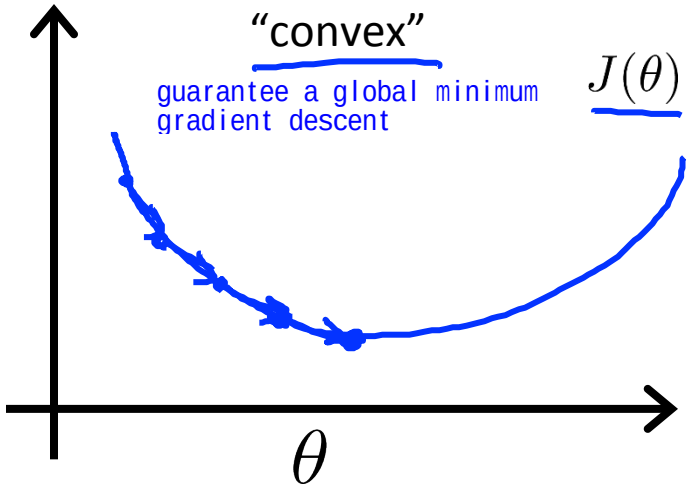
$\rightarrow \text{cost}(h_\theta(x^{(i)}), y)$

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

we want to minimize a cost function

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





Logistic regression cost function

cost function is linear regression for logistic regression is not convex, we have to define a new cost function so that it can be convex!

predicted value

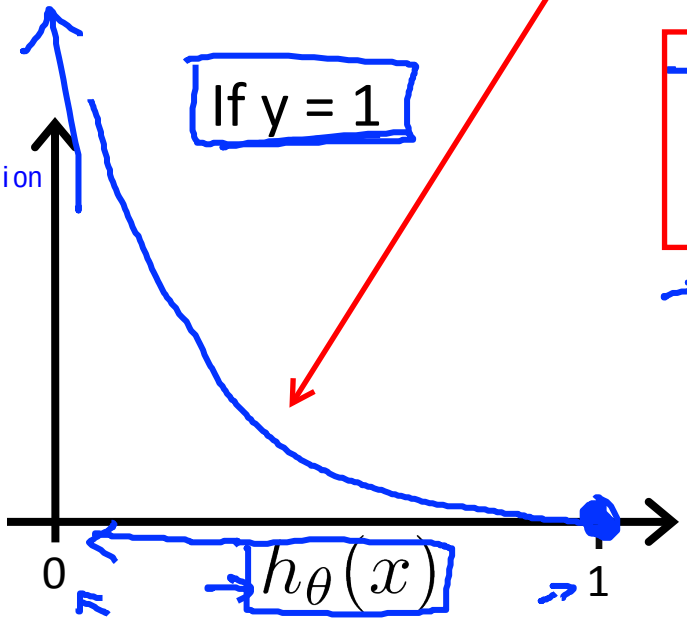
$$\text{Cost}(h_\theta(x), y) = \begin{cases} -\log(h_\theta(x)) & \text{if } y = 1 \\ -\log(1 - h_\theta(x)) & \text{if } y = 0 \end{cases}$$

actual value

when cost=0, if y=1, the predicted is 1 if the predicted approaches 0, the cost approaches infinity!!!

If y = 1

cost function y = 1



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

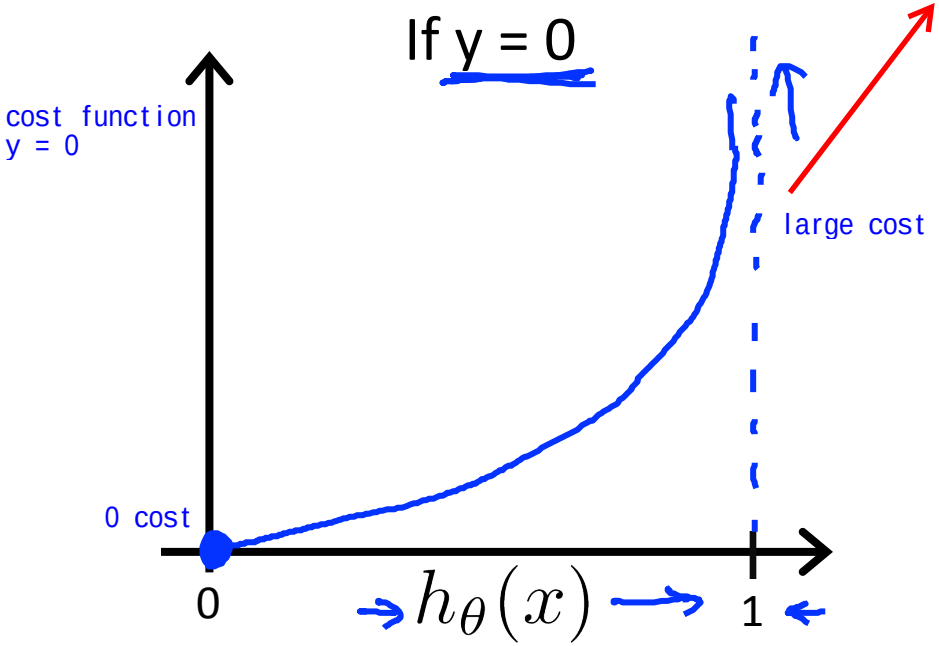
But as $h_\theta(x) \to 0$

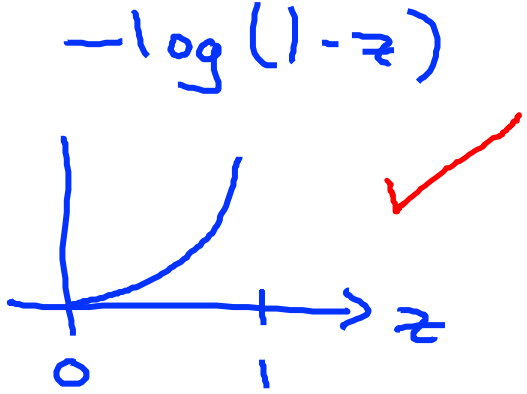
$Cost \to \infty$

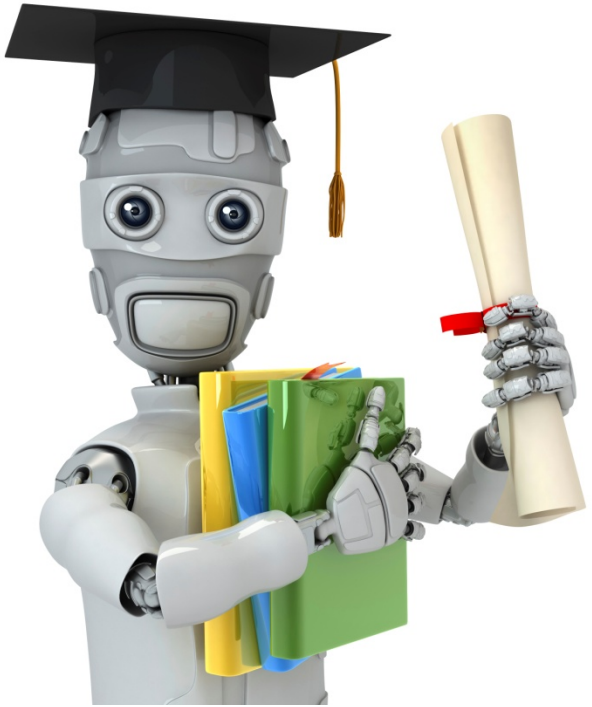
Captures intuition that if $h_\theta(x) = 0$, (predict $P(y = 1|x; \theta) = 0$), but $y = 1$, we'll penalize learning algorithm by a very large cost.

Logistic regression cost function

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







Machine Learning

Logistic Regression

Simplified cost function and gradient descent

Logistic regression cost function

overall cost function

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

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

Note: $y = 0$ or 1 always

combine them into a single equation

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

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

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

here y can either be 1 or 0,
therefore we can write in this way

Logistic regression cost function

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

then we can rewrite our cost function as follows

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

This is the cost function that everyone uses, and it is convex!!!

To fit parameters θ :

$$\min_\theta J(\theta)$$ Get θ

→ objective!

To make a prediction given new x:

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

output of hypothesis can be interpreted as the probability of y=1

Gradient Descent

We use gradient descent to minimize the cost function

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

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

Repeat $\{$

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

$\}$

same as usual!

(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

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

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

Repeat $\{$

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

(simultaneously update all θ_j)

$\}$

$\theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \vdots \\ \theta_n \end{bmatrix}$ for $i = 0$ to n

we would like to use vector notation to improve the efficiency!

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

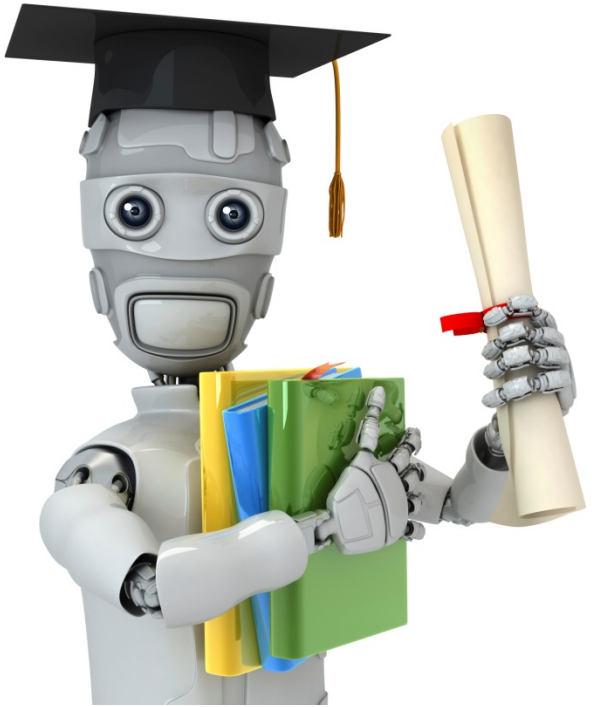
linear regression

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

logistic regression

Algorithm looks identical to linear regression!

feature scaling can also help your gradient descent algorithm to run faster!



Machine Learning

Logistic Regression

Advanced optimization

Optimization algorithm

gradient descent review

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

Given θ, we have code that can compute

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

supply code here!

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

more sophisticated algorithms beyond the scope of this course!

It is possible to implement (use software) them without understand them thoroughly!

Advantages: of those algorithms!

- No need to manually pick α
- Often faster than gradient descent. details are beyond the scope of this course!

Disadvantages:

- More complex

example of how to use these algorithms

Example:

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

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

exact solution!

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

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

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

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

compute cost function

gradient, 2x1 vector

```
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);
```

means you are going to provide the gradient

max iterations

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

we can then can this function!

unc = unconstrained

@ is a pointer of our defined function

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

we have to have at least 2 variables!

Apply in logistic regression

Note: These algorithms are faster than gradient descent but they are hard to debug. When you have a large ML problem,you should use these advanced optimization algorithms!

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

theta(1)

theta(2)

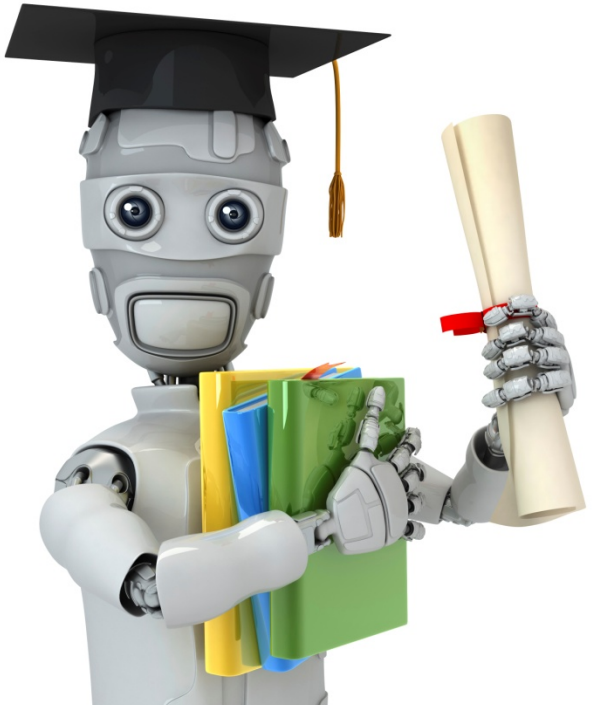
theta(n+1)

our function has to return the cost function and the gradients

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

This is the template

```
jVal = [code to compute J(theta)];
gradient(1) = [code to compute (∂/∂θ0) J(theta)];
gradient(2) = [code to compute (∂/∂θ1) J(theta)];
  :
gradient(n+1) = [code to compute (∂/∂θn) J(theta)];
```



Machine Learning

Logistic Regression

Multi-class classification:

One-vs-all

Multiclass classification

not just 0,1, but with multiple cases

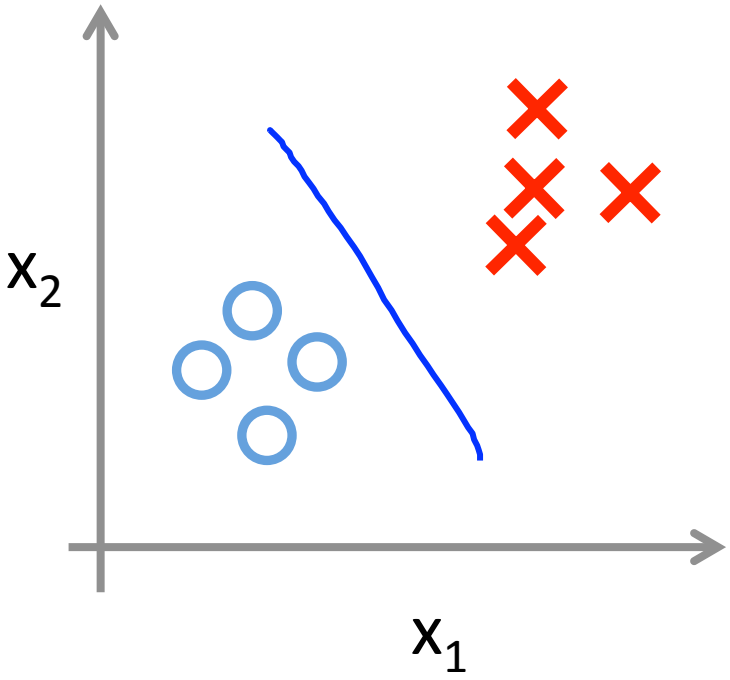
Examples!

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

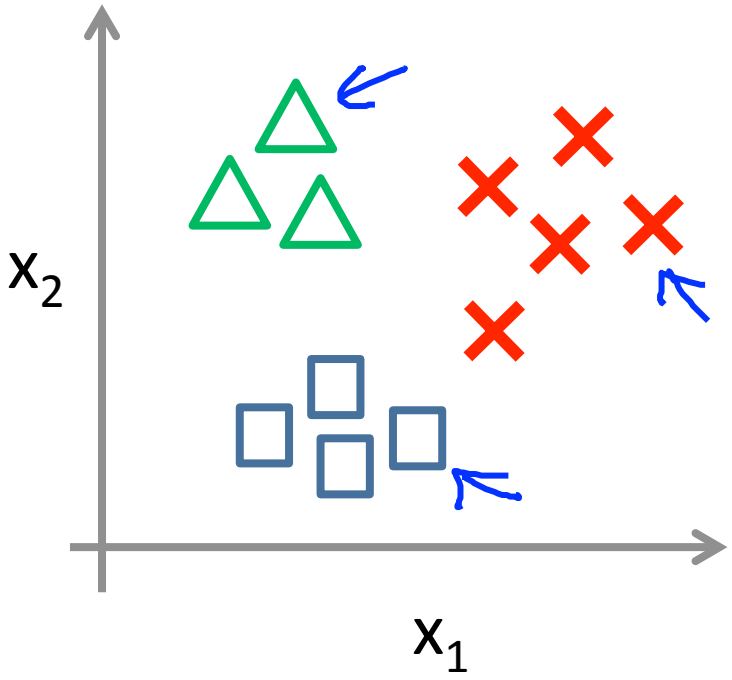
Medical diagrams: Not ill, Cold, Flu

Weather: Sunny, Cloudy, Rain, Snow

Binary classification:



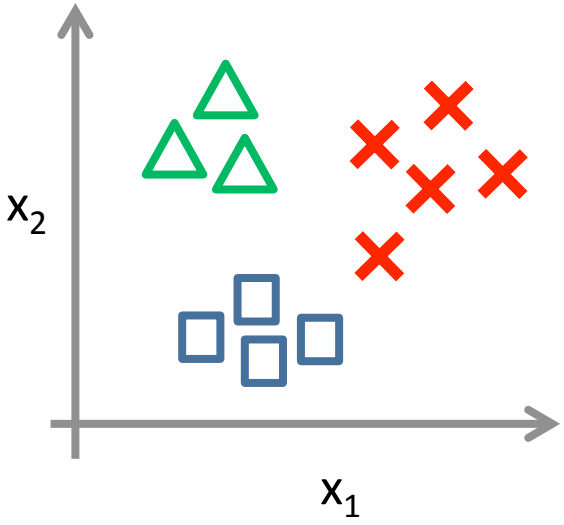
Multi-class classification:



the main idea is to turn the multi case to binary case

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

we create as a new sort of fake training set where classes two and three get assigned to the negative class



we do the same thing for class 2

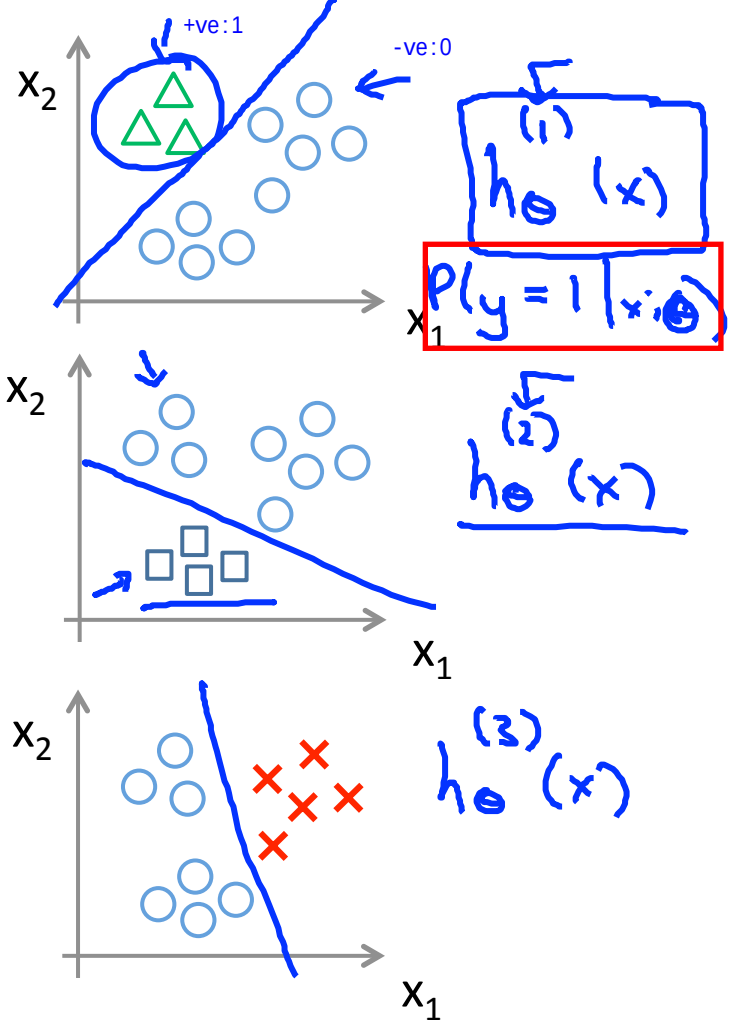
and so on so forth

Class 1: △ ←

Class 2: □ ←

Class 3: ✕ ←

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



Andrew Ng

One-vs-all

To summarize

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

To make a prediction

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

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

test h(x) for all classes and choose whichever the h gives us the largest value