6. Classification Algorithm – Logistic regression

Classification and Representation

SUPERVISED LEARNING:

- **REGRESSION** – y can take any value
- **CLASSIFICATION** - y can take only specified **discrete** value

Classification

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

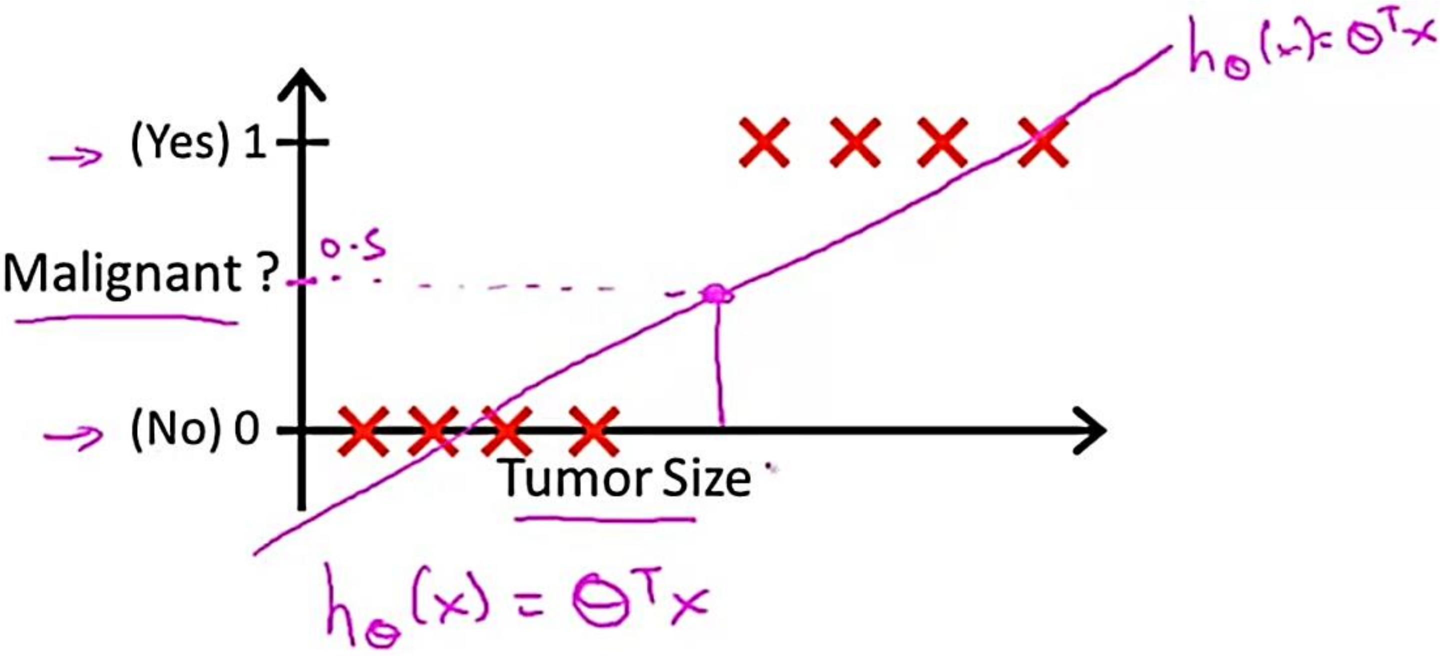
$y \in \{0, 1\}$ 0: "Negative Class"
1: "Positive Class"

Or

→ $y \in \{0, 1, 2, 3\}$

WHY REGRESSION ALGO **CAN'T** BE USED ON CLASSISFICATION PROBLEMS:

For a classification problem: if regression algo is applied:



To attempt classification, one method is to use linear regression and map all predictions greater than 0.5 as a 1 and all less than 0.5 as a 0. However, this method doesn't work well because classification is not actually a linear function.

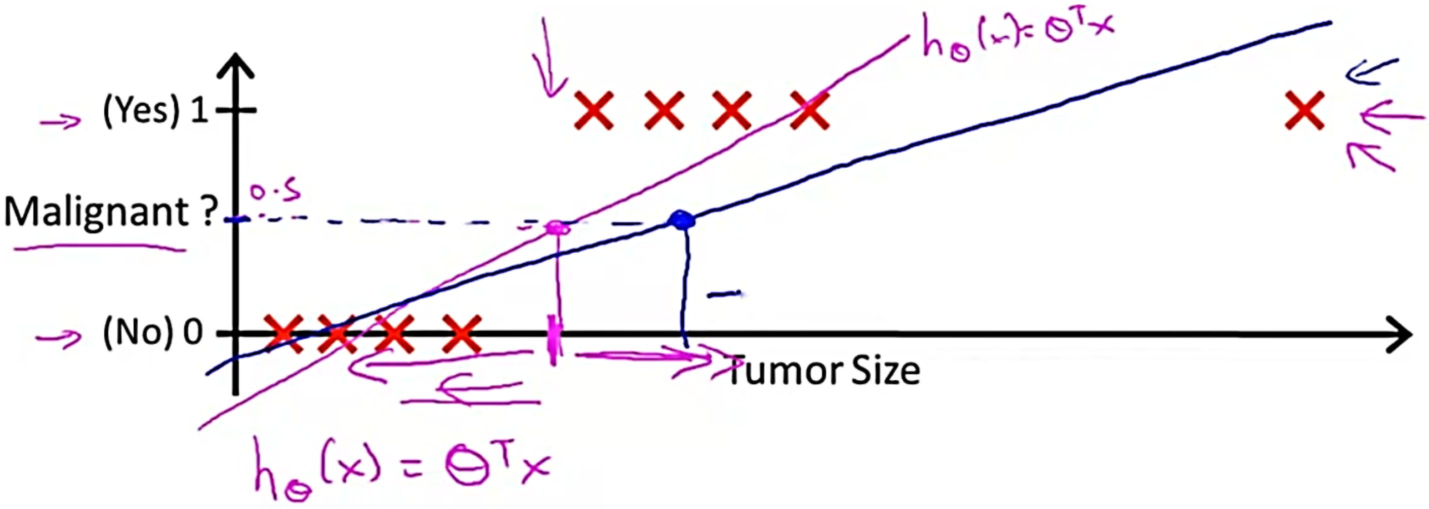
Threshold classifier output $h_\theta(x)$ at 0.5:

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

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

This seems to work fine for this example but:

If we add another data point:



Algo starts giving wrong hypothesis: Thus using reg algo on classification problem is bad

For a classification problem: **binary classification**

Classification: $y = 0$ or 1

h(x) can be:

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

So an algo is used :

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

Classification

Given x^i, the

Corresponding $\mathbf{y^i}$ is also called the **label** for the training example.

HYPOTHESIS REPRESENTATION:

Logistic Regression Model

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

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

$$z = \theta^T x$$

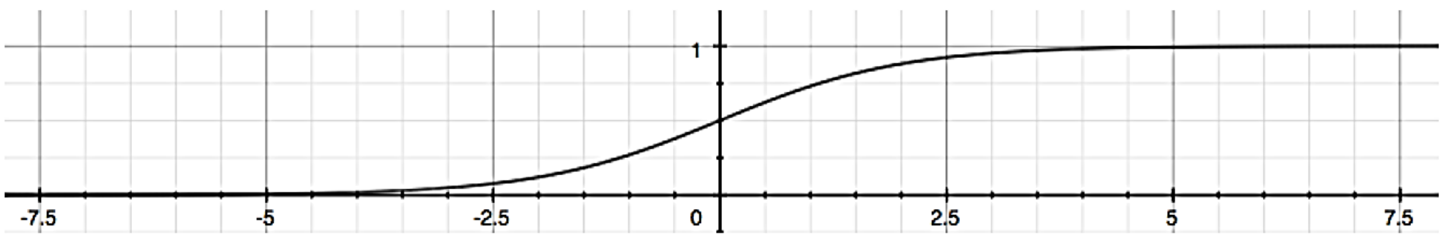
$$g(z) = \frac{1}{1+e^{-z}}$$

Here **g(z)** = **sigmoid** or logistic function

Sigmoid function

Logistic function

g(z) vs z: logistic fxn



Sigmoid fxn takes **asymptotes** at 0 (for -inf) and 1(for +inf)

We need to find best fitting parameters Θ for our hypothesis.

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

Interpretation of Hypothesis Output

h_θ

$h_\theta(x)$ = 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

$h_\theta(x) = P(y=1|x;\theta)$ "probability that y = 1, given x, parameterized by θ"

This means that **h(x)** – **is** just **the probability** of **y being equal to 1.**

Y == 0 or 1:

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

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

This gives the **prob. Of y=1** for an example having particular values of X and Θ.

i.e., prob of y==0 for a patient having values of x and prob of y=1 for the same patient adds up to 1 == 100%.

Our probability that our prediction is 0 is just the **complement** of our probability that it is 1 (e.g. if probability that it is 1 is 70%, then the probability that it is 0 is 30%).

DECESION BOUNDARY:

$$h_\theta(x) = g(\theta^T x) = P(y=1 \mid x; \theta)$$

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

$$\theta^T x \geq 0$$

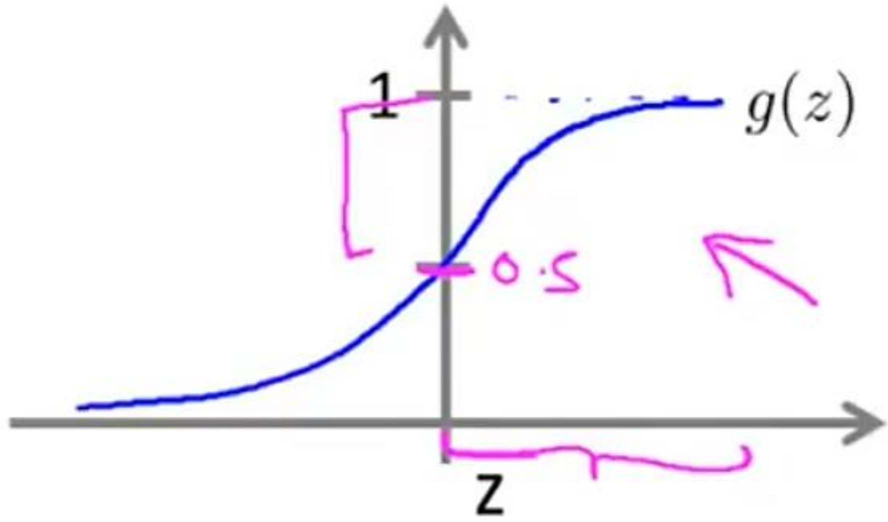
predict "$y = 0$" if $h_\theta(x) < 0.5$

$g(z) \geq 0.5$

when $z \geq 0$

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

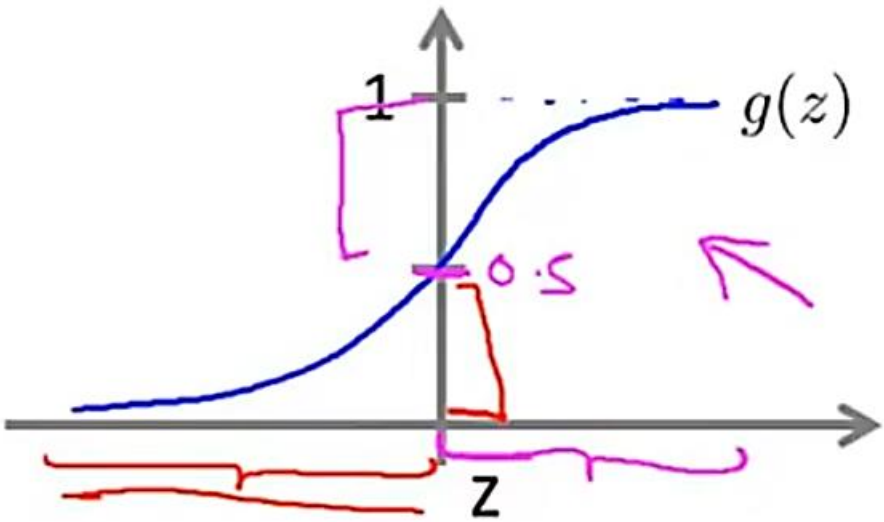
whenever $\underset{z}{\theta^T x} \geq 0$



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

$h_\theta(x) = g(\theta^T x)$ $\quad$ $g(z) < 0.5$

$\rightarrow \theta^T x < 0$



This means:

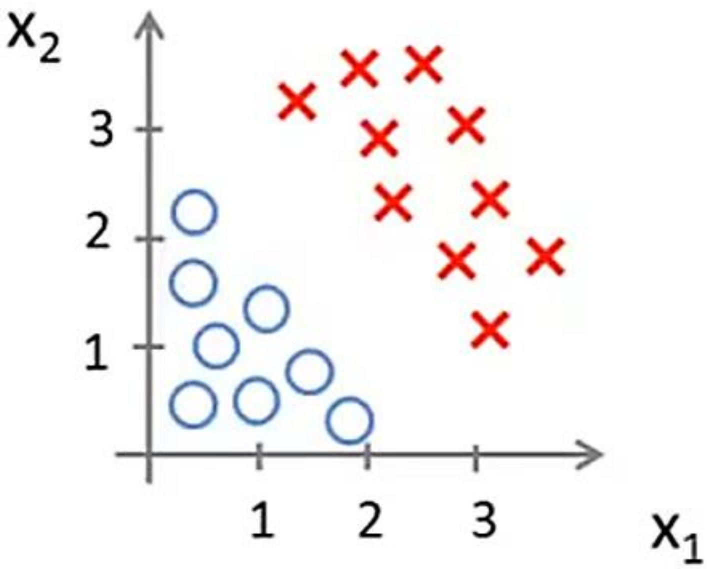
$g(z) \geq 0.5$
$when\ z \geq 0$

$\theta^T x \geq 0 \Rightarrow y = 1$
$\theta^T x < 0 \Rightarrow y = 0$

Decision Boundary: The decision boundary is the line that separates the area where y = 0 and where y = 1.

It is created by our hypothesis function.

Decision Boundary



Example: taking a **linear example**

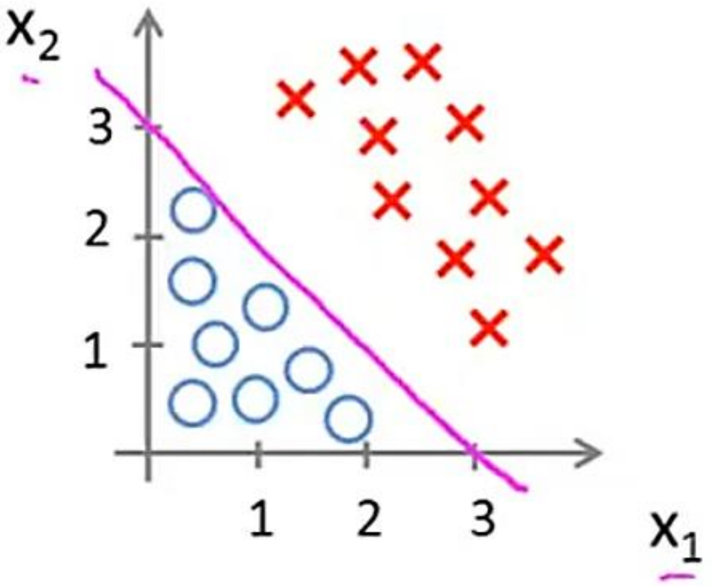
$$\theta = \begin{bmatrix} -3 \\ 1 \\ 1 \end{bmatrix}$$

$$h_\theta(x) = g(\underbrace{\theta_0}_{= -3} + \underbrace{\theta_1}_{= 1} x_1 + \underbrace{\theta_2}_{= 1} x_2)$$

Predict "$y = 1$" if $\underbrace{-3 + x_1 + x_2}_{\theta^T x} \geq 0$

$\rightarrow x_1 + x_2 \geq 3$

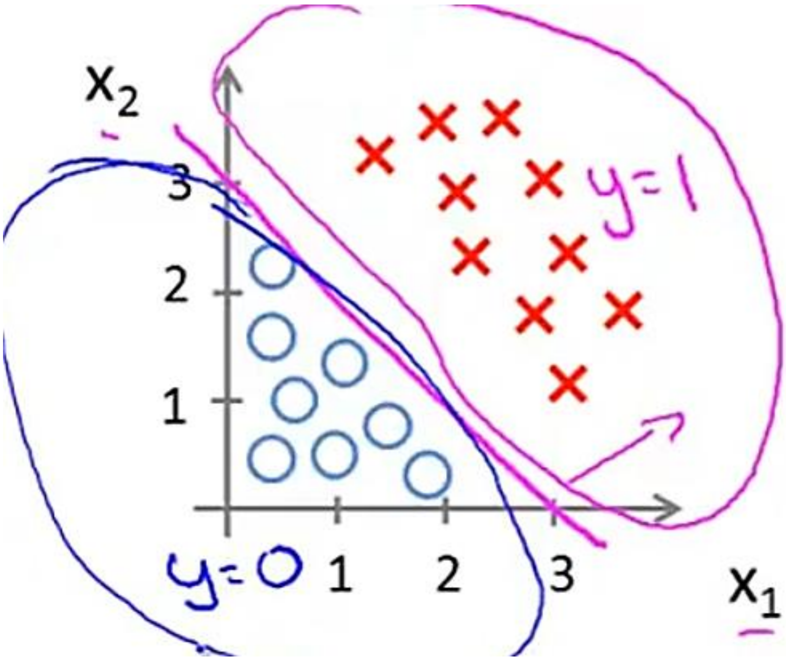
$x_1 + x_2 = 3$



this line is k/a decision boundary

All values to the right of this line correspond to y==1

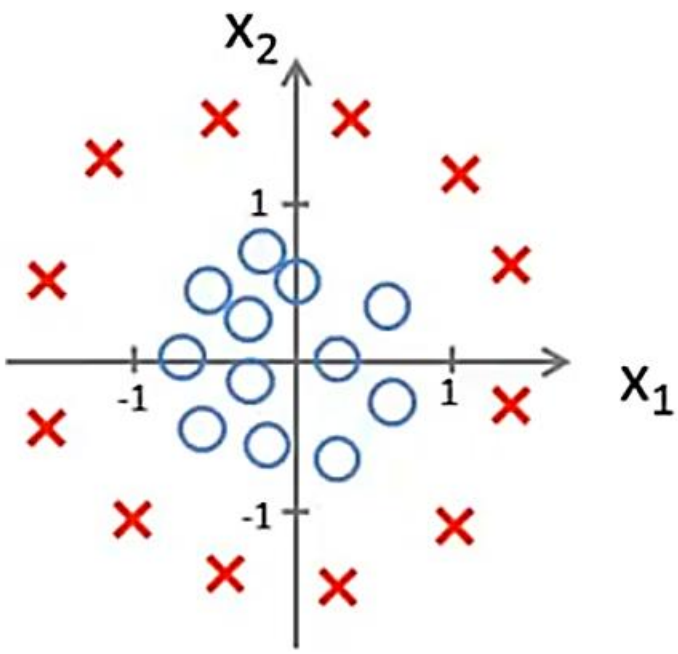
And all the values on left are for y==0



Non-linear example:

The hypothesis fxn can be super complex

Non-linear decision boundaries

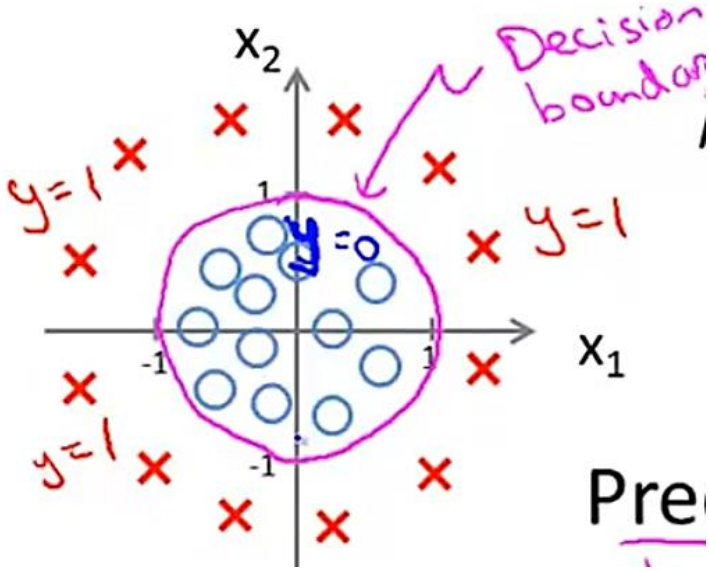


$$h_\theta(x) = g(\overset{=-1}{\theta_0} + \overset{=0}{\theta_1} x_1 + \overset{=0}{\theta_2} x_2 + \underset{=1}{\theta_3} x_1^2 + \underset{=1}{\theta_4} x_2^2)$$

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

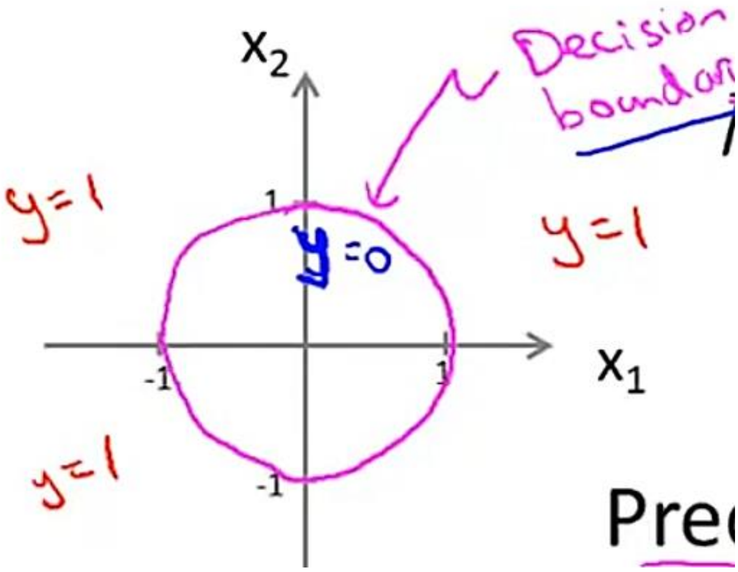
Predict "$y = 1$" if $\underbrace{-1 + x_1^2 + x_2^2 \geq 0}_{x_1^2 + x_2^2 \geq 1}$

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



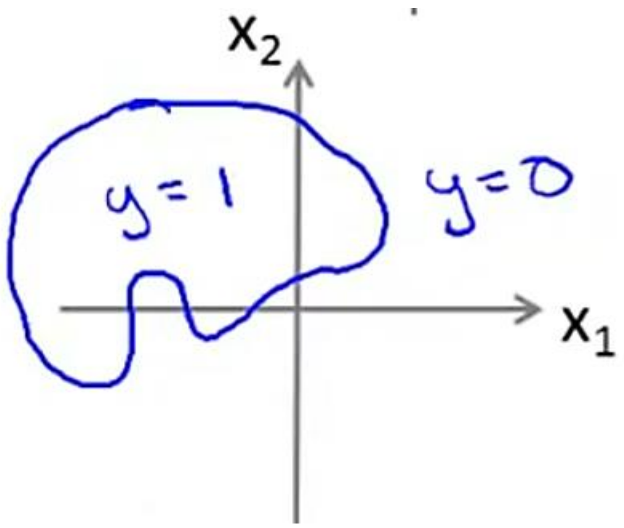
It can be noted that: the decision boundary depends only on the parameters(Θ) and not on the training set(values of x^i and y)

Thus:



The hypothesis fxn can be super complex and the decision boundary may have weird looking curves:

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



COST FUNCTION:

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

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

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

How to choose parameters θ ?

For a linear reg fxn:

Cost function

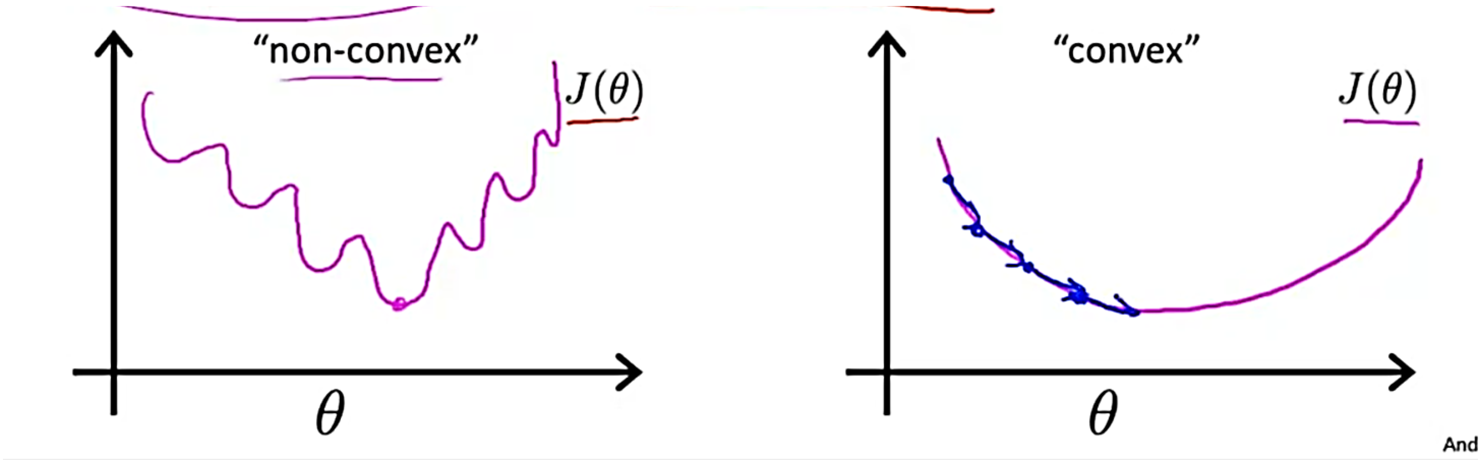
Linear regression: $J(\theta) = \frac{1}{m} \sum_{i=1}^{m} \cancel{\frac{1}{2} \left(h_\theta(x^{(i)}) - y^{(i)}\right)^2}$

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

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

But for logistic fxn:

This cost fxn is a non-convex fxn of Θ:



In a non-convex fxn there can be so many local optimas that it's hard to reach global minima.. unlike in convex fxn

In non-convex fxns, linear reg cost algo does not guarantee global mnima

So we choose a **different Cost fxn**:

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

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

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

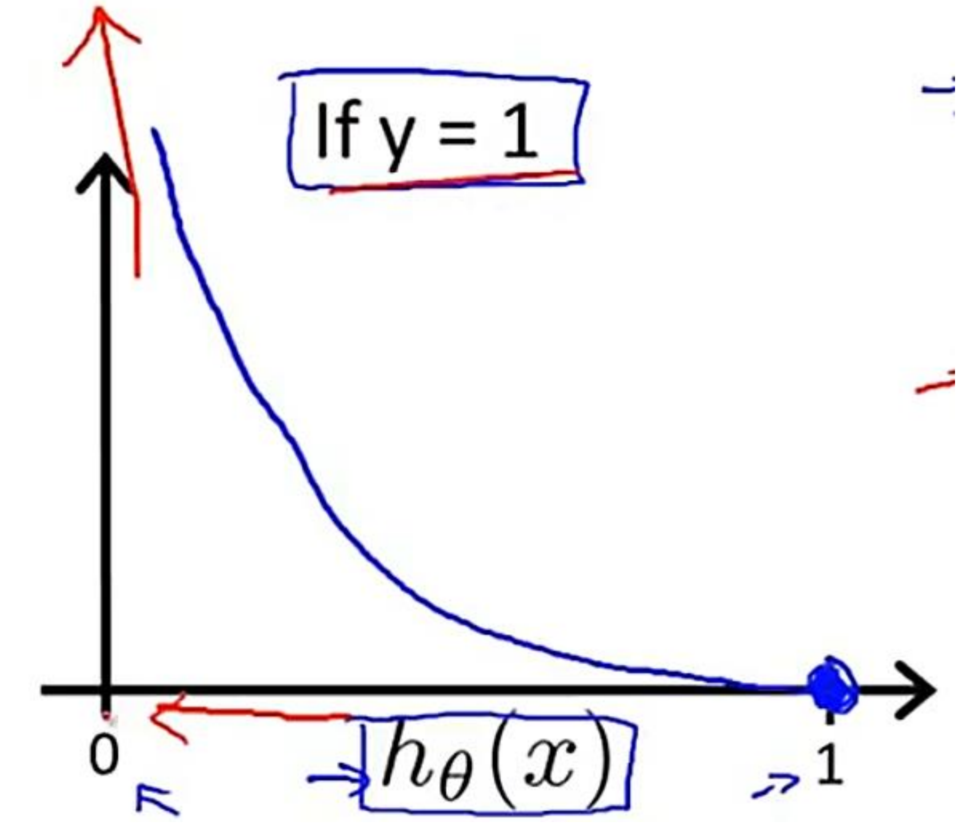
Note that writing the cost function in this way **guarantees** that **J(θ) is convex for logistic regression**

Logistic regression cost function

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

This is the cost of just one example. it will be needed to be summed for J(Θ)—error fxn

Cost vs h(x)



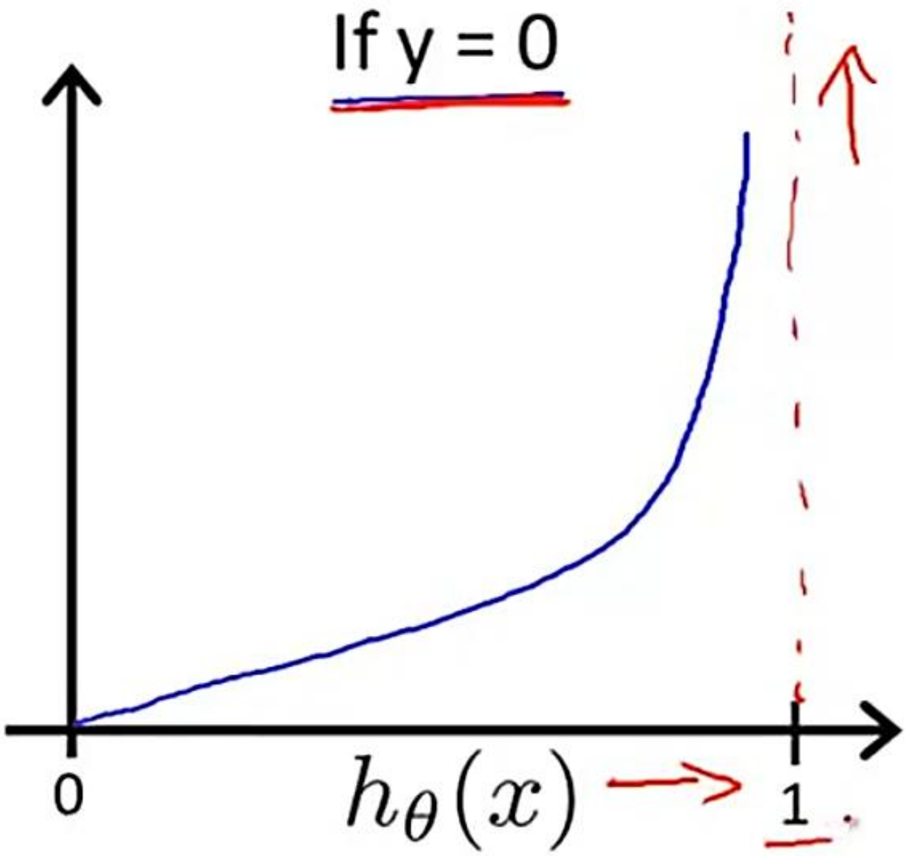
Cost $= 0$ if $y = 1, h_\theta(x) = 1$

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.

Cost vs h(x)



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

The above two graphs for **y==1** and **y==0** means that:

The cost of a given value of h(x) is the cost that is to be paid by the algorithm for that prediction.

It is the measure of how wrong the prediction is.

SIMPLIFIED COST FUNCTION: equivalent cost function:

Logistic regression cost function

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

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

Note: $y = 0$ or 1 always

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

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

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

Logistic regression cost function

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

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

VECTORIZED IMPLEMENTATION:

$$h = g(X\theta)$$

$$J(\theta) = \frac{1}{m} \cdot \left(-y^T \log(h) - (1-y)^T \log(1-h)\right)$$

To fit parameters θ:

$$\min_\theta J(\theta)$$ Get θ

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 {

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

} (simultaneously update all θ_j)

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

Repeat {

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

}

A vectorized implementation is:

$$\theta := \theta - \frac{\alpha}{m} X^T (g(X\theta) - \vec{y})$$

OPTIMIZATION ALGORITHMS:

- Conjugate gradient
- BFGS
- L-BFGS

Advantages:

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

Disadvantages:

- More complex

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

$$\theta = \begin{bmatrix} \theta_1 \\ \theta_2 \end{bmatrix} \qquad \theta_1 = 5, \theta_2 = 5.$$

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

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

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

```
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);
```

We can use octave's "**fminunc**()" optimization algorithm

For bigger example:

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

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

    jVal = [ code to compute J(θ)];

    gradient(1) = [code to compute ∂/∂θ0 J(θ)];

    gradient(2) = [code to compute ∂/∂θ1 J(θ)];
      ⋮

    gradient(n+1) = [code to compute ∂/∂θn J(θ)  ];
```

MULTICLASS CLASSIFICATION: One vs All Classification

When y can take more than 2 values: but they are discrete

Instead of y = {0,1} we will expand our definition so that y = {0,1...n}.

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

$y=1$ $y=2$ $y=3$ $y=4$

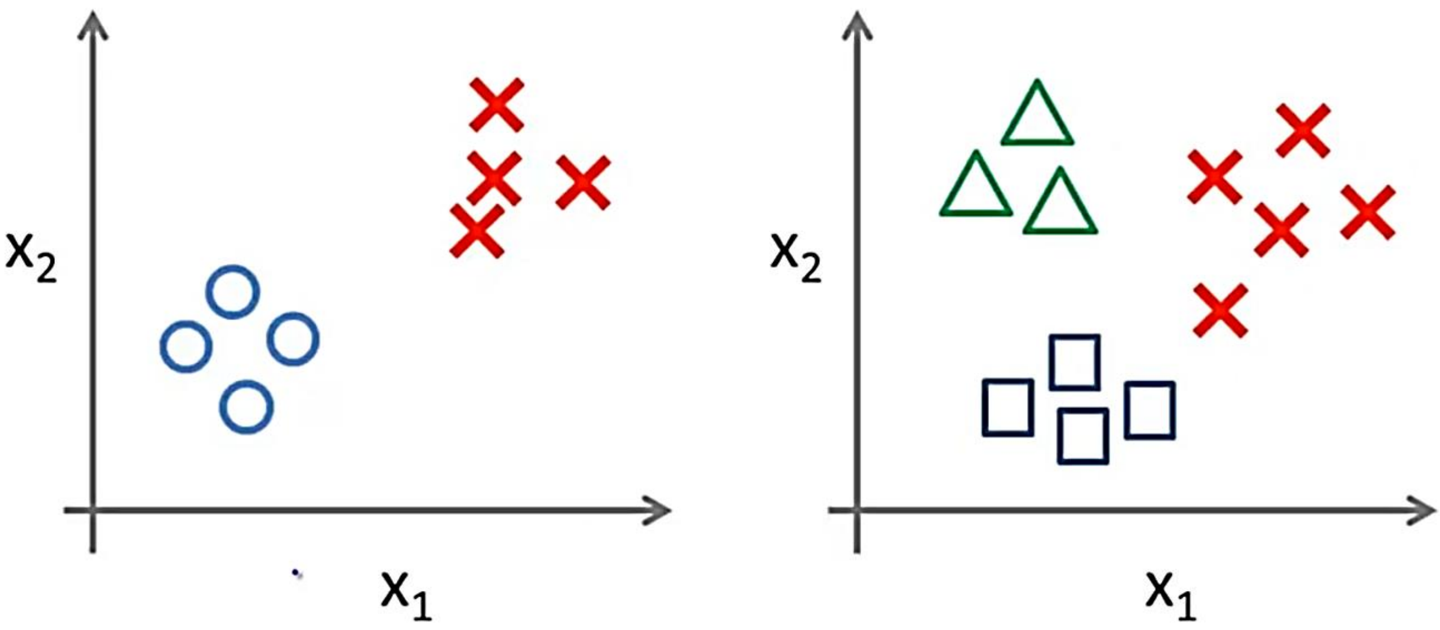
Medical diagrams: Not ill, Cold, Flu

$y=1$ 2 3

Weather: Sunny, Cloudy, Rain, Snow

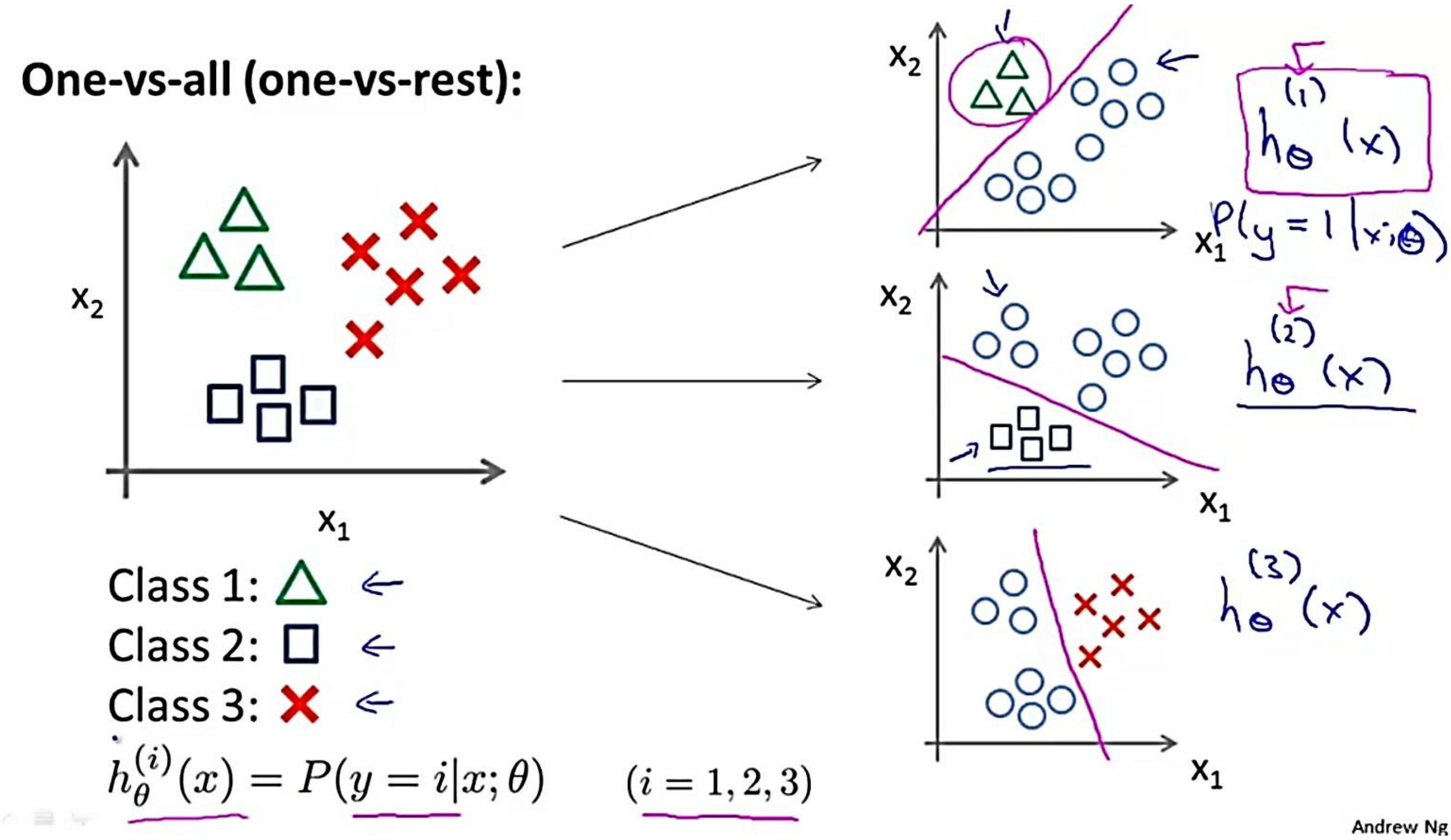
$y=1$ 2 3 4

Binary classification: Multi-class classification:



Since **y = {0,1...n},** we divide our problem into n+1 (+1 because the index starts at 0) binary classification problems; in each one, we predict the probability that 'y' is a member of one of our classes.

The multiclass problem is considered using 3 different binary classification problem



$h_\Theta^{(i)}$ (x) – here (i) denotes the class(y) – each value of y corresponds to a diff class

- It denotes the probability of y belonging to that class. That is it's the prob. Of y's value being that of i.
- Therefore, the values of all 3 h_Θ (x) are computed than the one corresponding to highest probability is selected.
- A different decision boundary exits for all classes

To make a prediction on a new x ➔ pick the class that maximizes $h_\Theta(x)$

$$y \in \{0, 1 \ldots n\}$$
$$h_\theta^{(0)}(x) = P(y = 0|x;\theta)$$
$$h_\theta^{(1)}(x) = P(y = 1|x;\theta)$$
$$\cdots$$
$$h_\theta^{(n)}(x) = P(y = n|x;\theta)$$
$$\text{prediction} = \max_i(h_\theta^{(i)}(x))$$

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

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