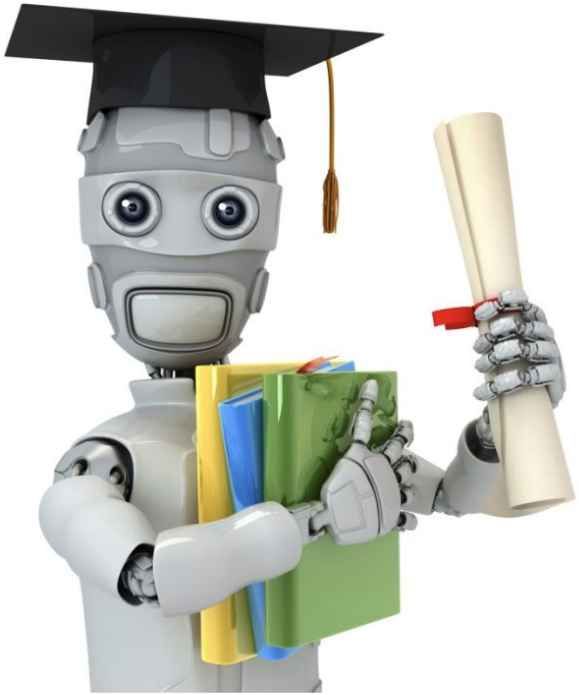


# Logistic Regression

# Classification

Machine Learning

**Week 8**

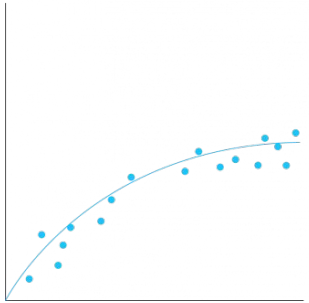
## Supervised Learning:

Predicting values. **Known** targets.
User inputs correct answers to learn from. Machine uses the information to guess new answers.

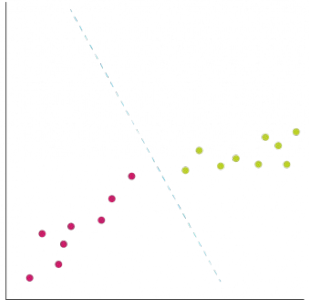
**REGRESSION:**
Estimate continuous values
(Real-valued output)

**CLASSIFICATION:**
Identify a unique class
(Discrete values, Boolean, Categories)

Regression



Classification



---

## Unsupervised Learning:

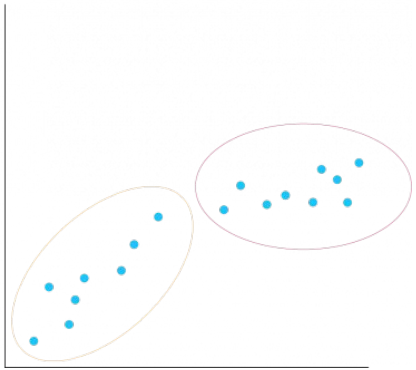
Search for structure in data. **Unknown** targets.
User inputs data with undefined answers. Machine finds useful information hidden in data.

**Cluster Analysis**
Group into sets

**Density Estimation**
Approximate distributions

**Dimension Reduction**
Select relevant variables

Clustering



## Classification

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

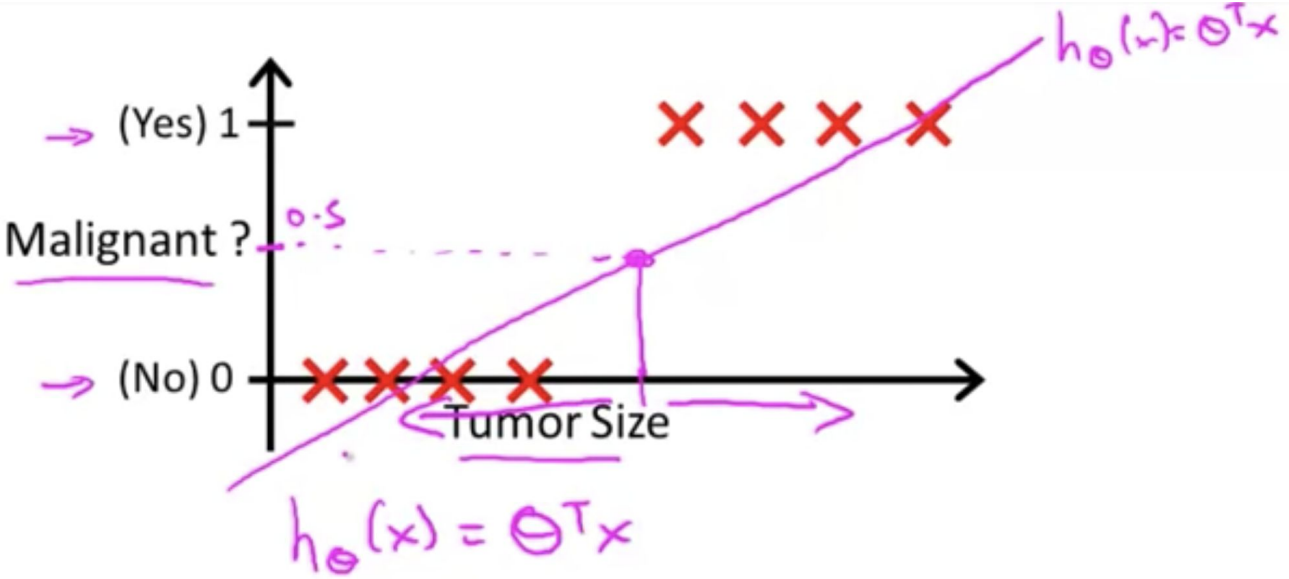
$y \in \{0, 1\}$

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

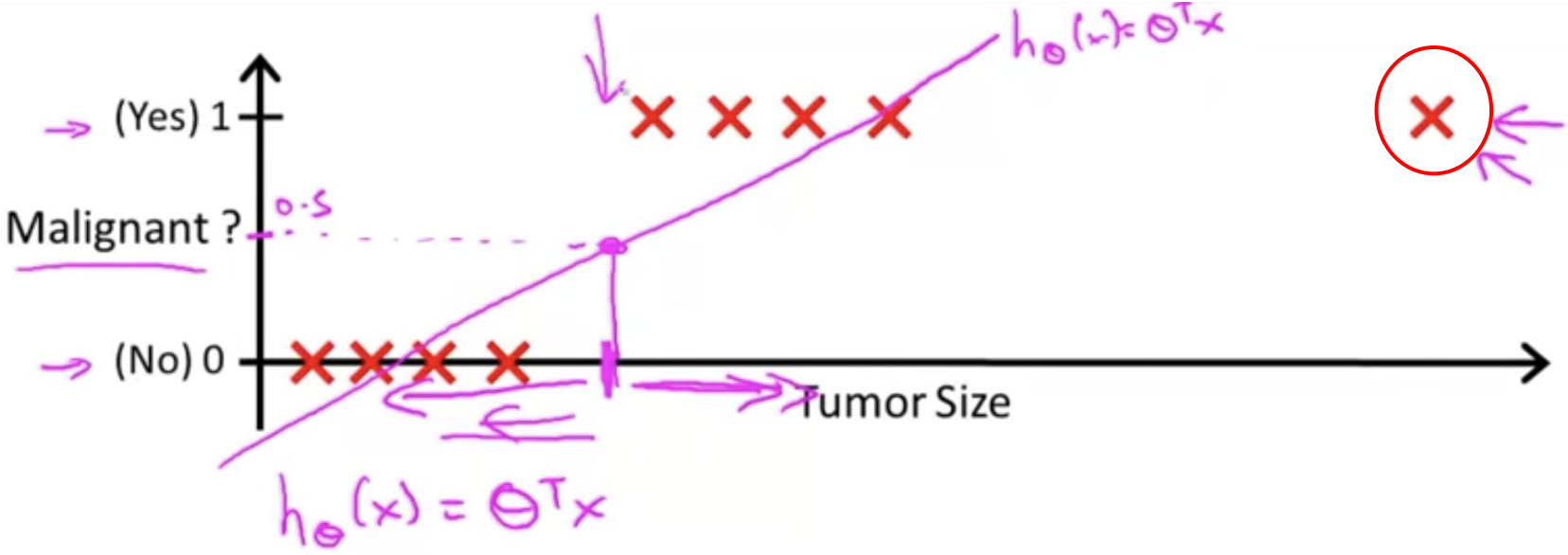
1: "Positive Class" (e.g., malignant tumor)

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

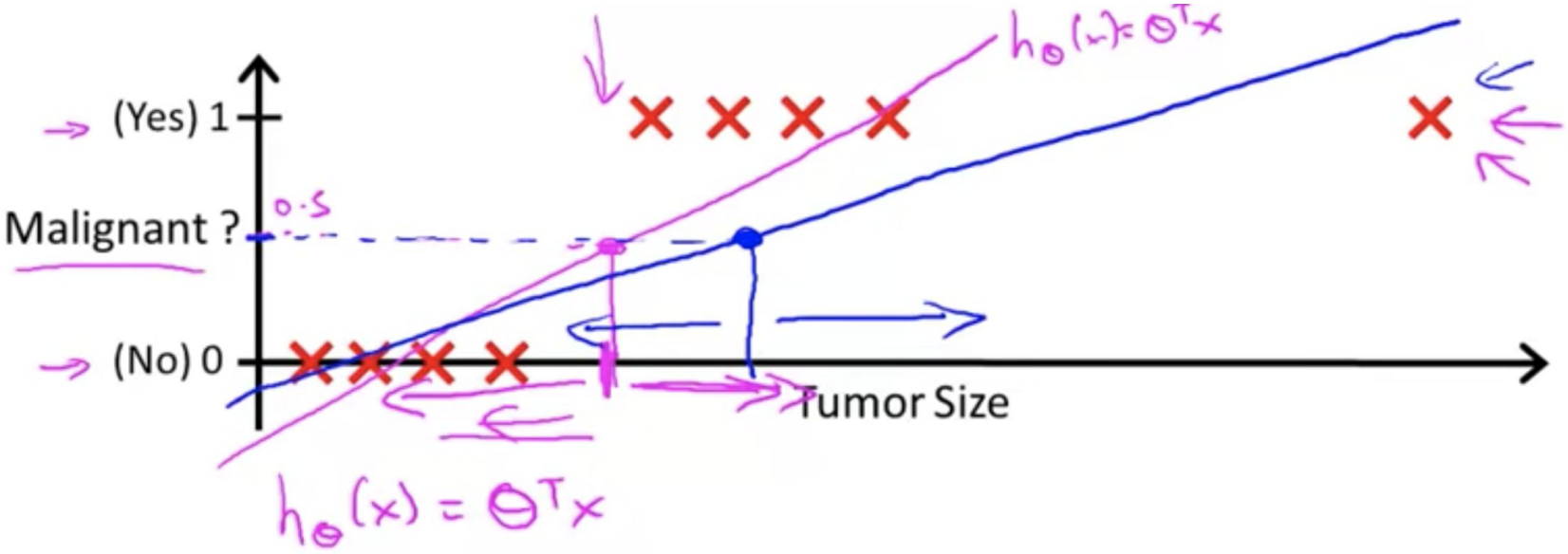
The definition of the labels are **arbitrary**




- 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"




Applying linear regression to classification problem is a bad idea!

Classification: $y = 0$ or $1$

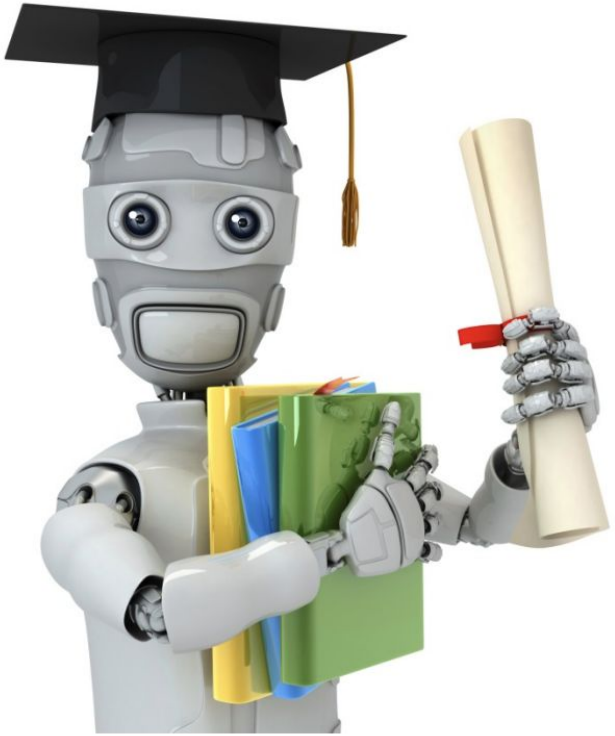
$h_\theta(x)$ can be $> 1$ or $< 0$

Why linear regression is a bad for classification.

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

Classification

What we want



Machine Learning

# Logistic Regression

## Hypothesis Representation

## Logistic Regression Model

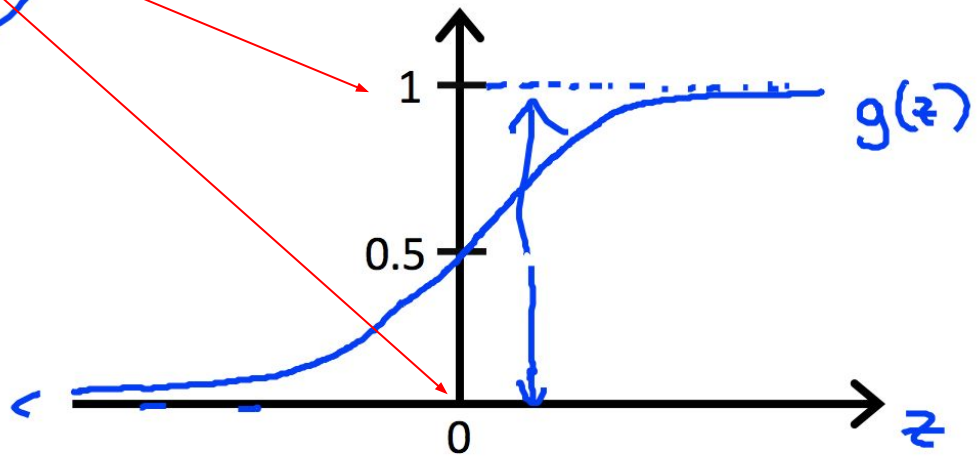
Want $0 \le h_\theta(x) \le 1$

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

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

$z = \theta^T x$

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




- Sigmoid function
- Logistic function

Parameters $\theta$.

## Interpretation of Hypothesis Output

$h_\theta(x)$

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

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

$h_\theta(x) = 0.7$ $\quad$ $y=1$

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 $\theta$"

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

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

The hypothesis is ranges between 0 and 1, and can be modelled as the probability that an output is 1.

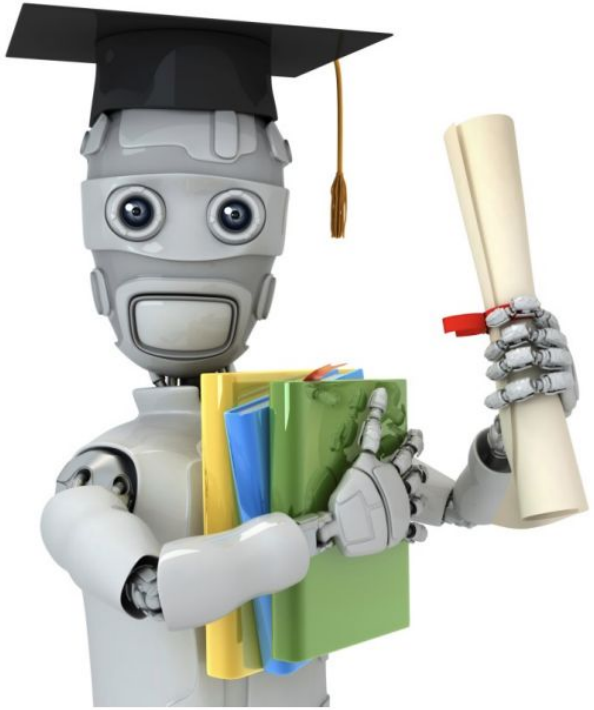
E.g, $h_\theta(x) = 0.7$ is a 70% chance that our output is 1.

Therefore,

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

And,

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



Machine Learning

# Logistic Regression

## Decision boundary

# Decision Boundary

The hypothesis gives values ranging from 0 to 1. In order to obtain the class of the prediction, we can establish a **threshold** in which values greater than the threshold are rounded up as 1 and lesser values are 0. We can choose 0.5

$$h_\theta(x) \geq 0.5 \rightarrow y = 1$$
$$h_\theta(x) < 0.5 \rightarrow y = 0$$

The logistic/sigmoid function has a very interesting behaviour. When its input is greater than or equal to 0, its output is greater than or equal to 0.5, and when its input is less than 0, its output is less than 0.5.

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

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

The **decision boundary** is a line that separates the areas y=1 and y=0

$$\theta = \begin{bmatrix} 5 \\ -1 \\ 0 \end{bmatrix}$$

$$y = 1 \; if \; 5 + (-1)x_1 + 0x_2 \geq 0$$

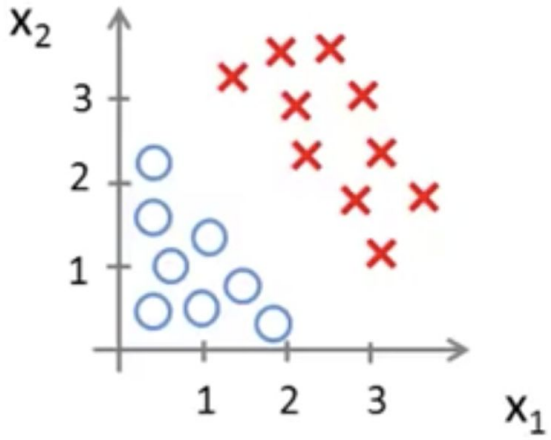
$$5 - x_1 \geq 0$$

$$-x_1 \geq -5$$

$$x_1 \leq 5$$

Find the decision boundary?

## Decision Boundary

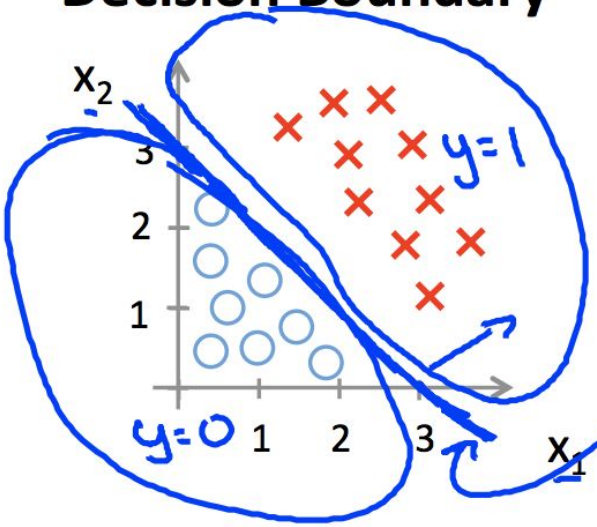



$$\theta = \begin{bmatrix} -3 \\ 1 \\ 1 \end{bmatrix}$$

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

$-3 \quad 1 \quad 1$

# Decision Boundary




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

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

-3 1 1

Decision boundary

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

$\theta^T x$

$x_1 + x_2 \geq 3$

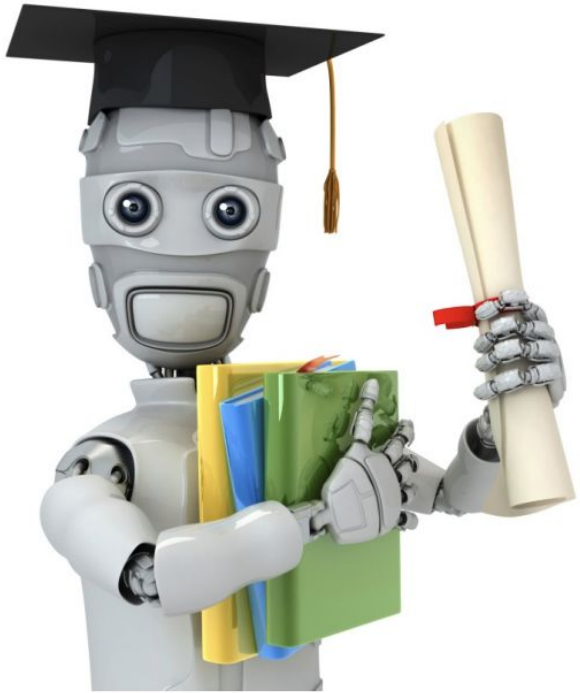
$x_1, x_2$

$h_\theta(x) = 0.5$

$x_1 + x_2 = 3$

$x_1 + x_2 < 3$

$y = 0$



Machine Learning

# Logistic Regression

## Cost function

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

m examples $x \in \begin{bmatrix} x_0 \\ x_1 \\ \cdots \\ x_n \end{bmatrix}$ $\mathbb{R}^{n+1}$ $x_0 = 1, y \in \{0, 1\}$

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

How to choose parameters $\theta$ ?

A cost function tells us “how good” our model is at making predictions for a given set of parameters. The cost function has its own curve and its own gradients. The slope of this curve tells us how to update our parameters to make the model more accurate.

Cost Function:

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

Questions:

1 - What cost function should we use ?

2 - Should we use **MSE** as in **Linear regression** ?

# Cost Function for Logistic Regression

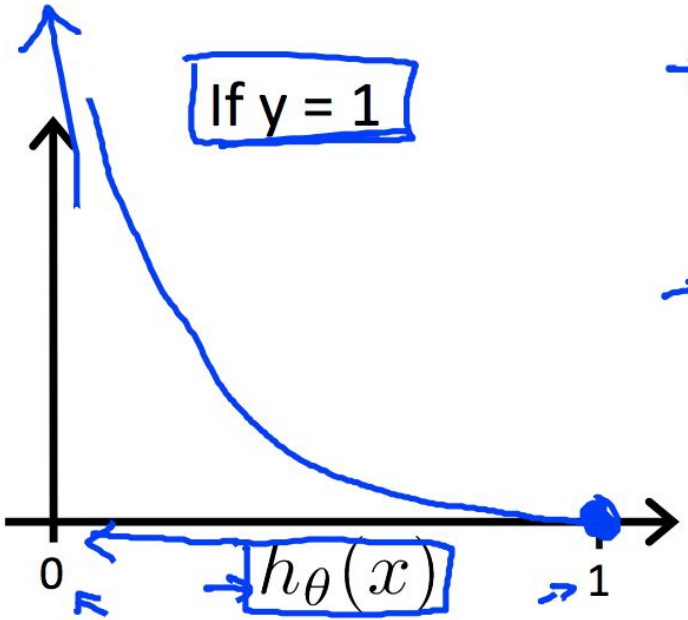
$$J(\theta) = \frac{1}{m} \sum_{i=1}^{m} \text{Cost}(h_\theta(x^{(i)}), y^{(i)})$$

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

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

# Logistic regression cost function

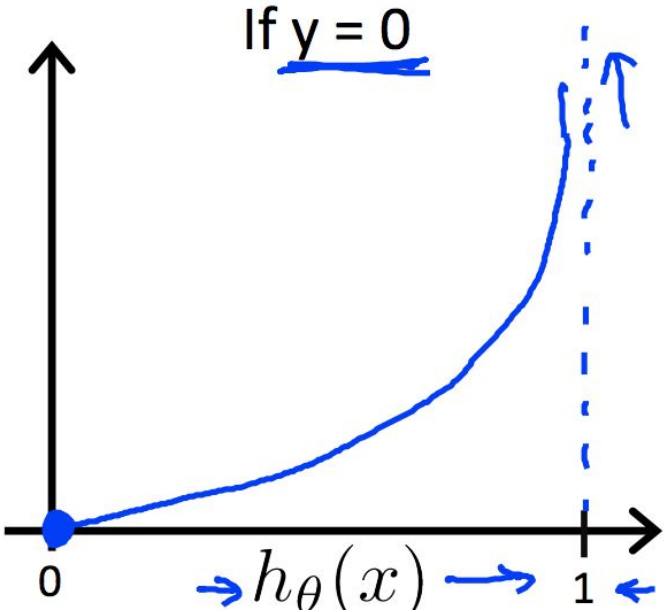
$$\text{Cost}(h_\theta(x), y) = \begin{cases} -\log(h_\theta(x)) & \text{if } y = 1 \\ -\log(1 - h_\theta(x)) & \text{if } y = 0 \end{cases}$$

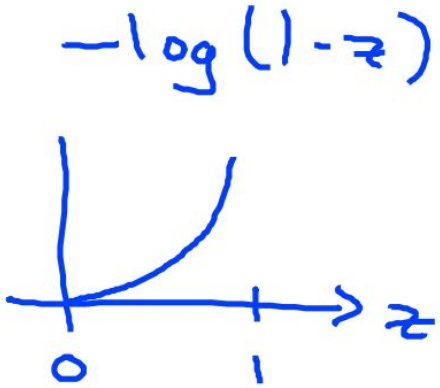



- $\text{Cost} = 0$ if $y = 1, h_\theta(x) = 1$
  But as $h_\theta(x) \to 0$
  $Cost \to \infty$
- Captures intuition that if $h_\theta(x) = 0$, (predict $P(y = 1|x; \theta) = 0$), but $y = 1$, we'll penalize learning algorithm by a very large cost.

# Logistic regression cost function

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

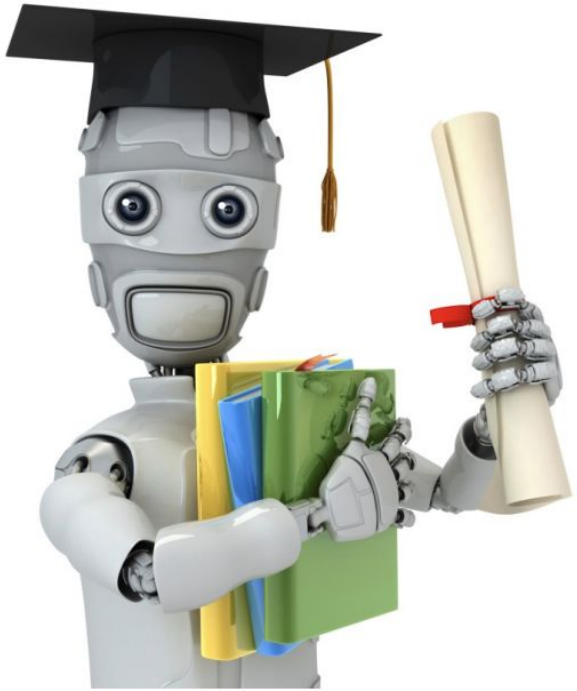


# Summary of cost function

If our hypothesis is far from y, the cost tends to infinity.
If our hypothesis is exactly the same as y, the cost is zero.

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



Machine Learning

# Logistic Regression

## Simplified cost function and gradient descent

# Simplified Cost Function and Gradient Descent

We can write the cost function's conditional cases in a more compact form

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

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

## 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 $\{$

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

(simultaneously update all $\theta_j$)

$\}$

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

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

Algorithm looks identical to linear regression!

## Optimization algorithm

Given $\theta$, we have code that can compute

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

(for $j = 0, 1, \ldots, n$ )

Optimization algorithms:

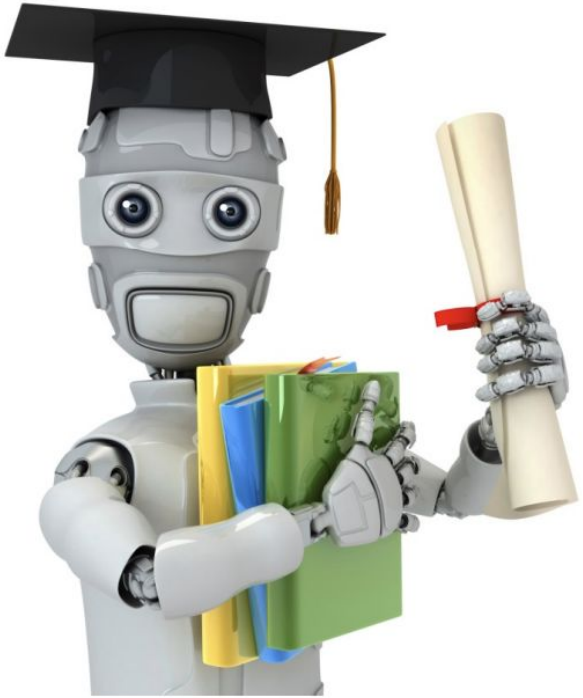
- Gradient descent
- Conjugate gradient
- BFGS
- L-BFGS

Advantages:

- No need to manually pick $\alpha$
- Often faster than gradient descent.

Disadvantages:

- More complex



Machine Learning

# Logistic Regression

## Multi-class classification: One-vs-all

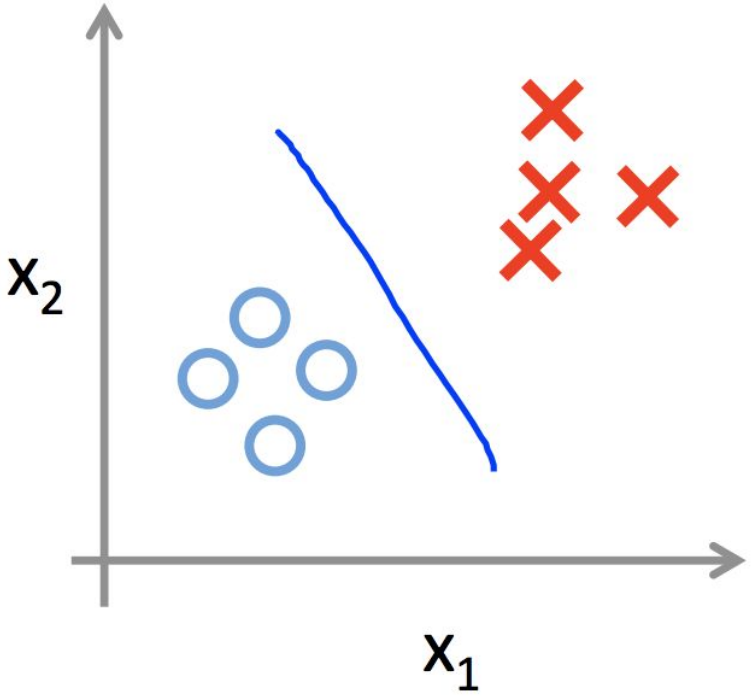
## Multiclass classification

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

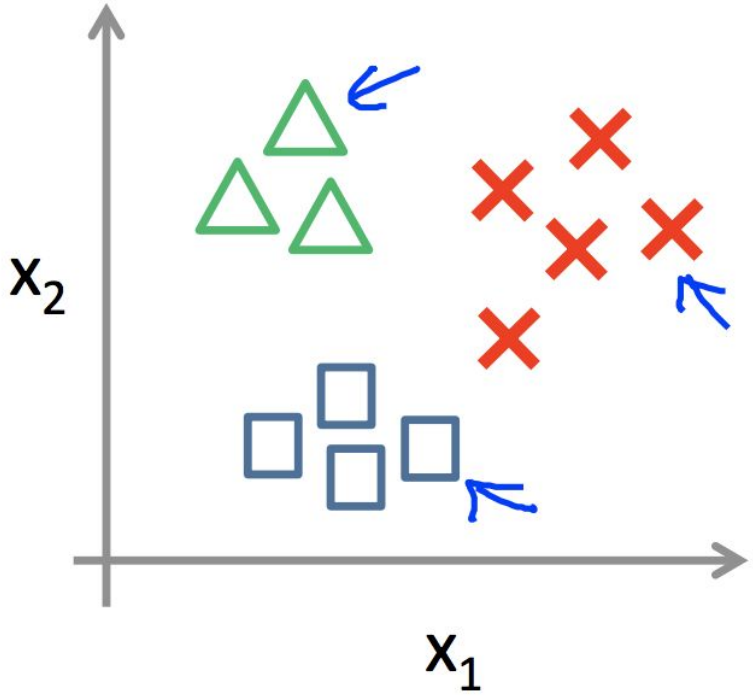

Medical diagrams: Not ill, Cold, Flu


Weather: Sunny, Cloudy, Rain, Snow

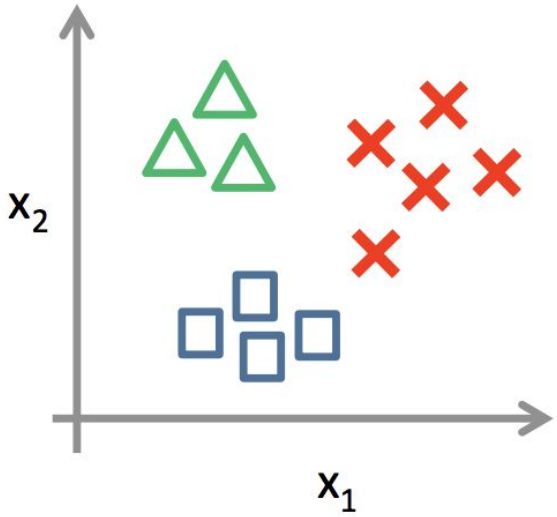
Binary classification:




Multi-class classification:



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

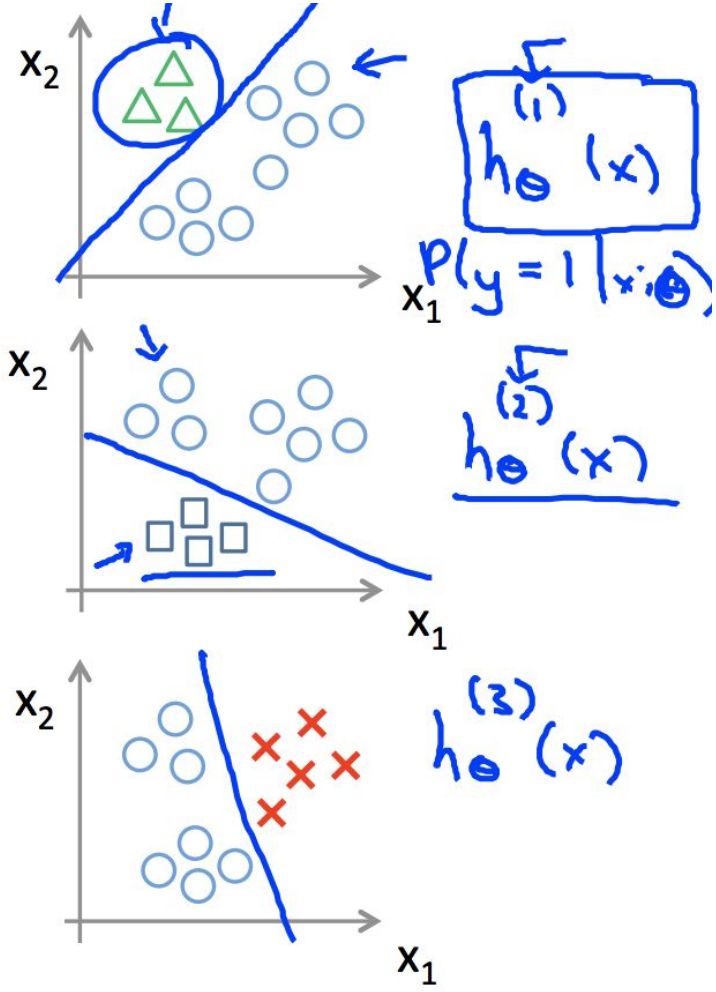



Class 1: △

Class 2: □

Class 3: ✕

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



# Multiclass Classification: One-vs-all

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

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

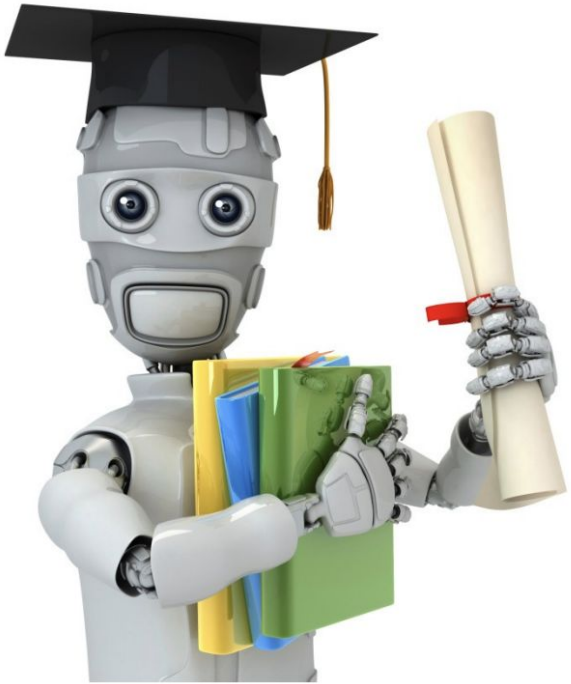
$$h_\theta^{(0)}(x) = P(y = 0 | x; \theta)$$

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

$$\dots$$

$$h_\theta^{(n)}(x) = P(y = n | x; \theta)$$

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

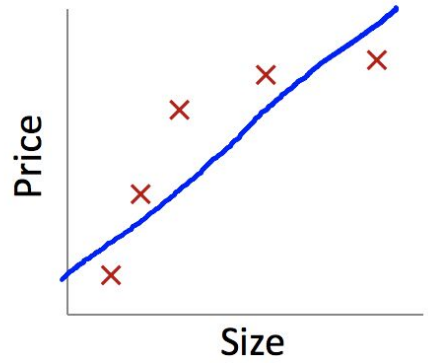


Machine Learning

# Regularization

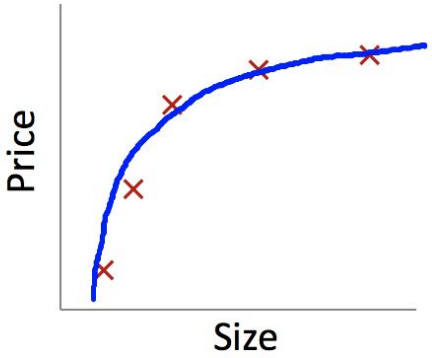
## The problem of overfitting

# Example: Linear regression (housing prices)



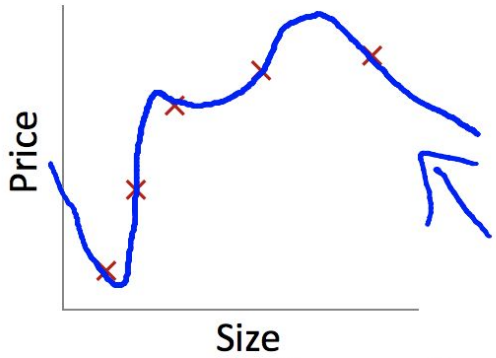

$\rightarrow \theta_0 + \theta_1 x$

"Underfit" "High bias"




$\rightarrow \theta_0 + \theta_1 x + \theta_2 x^2$

"Just right"

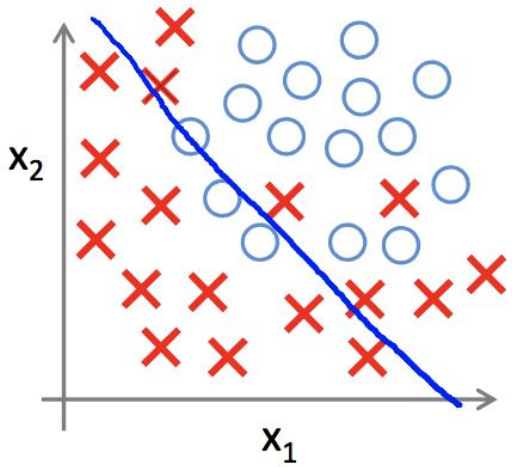



$\rightarrow \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$

"Overfit" "High variance"

**Overfitting:** If we have too many features, the learned hypothesis may fit the training set very well ($J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2 \approx 0$), but fail to generalize to new examples (predict prices on new examples).

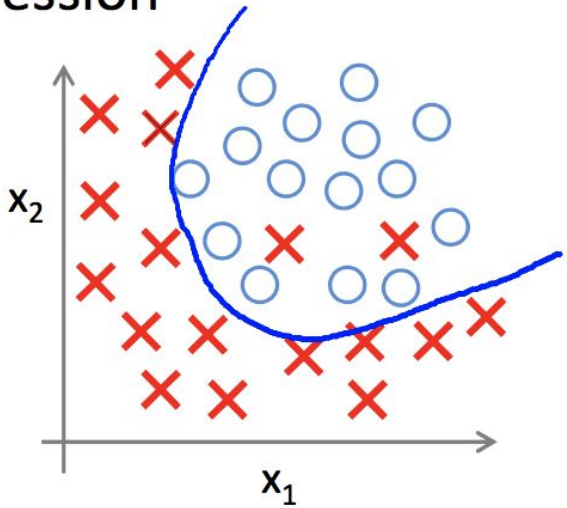
# Example: Logistic regression



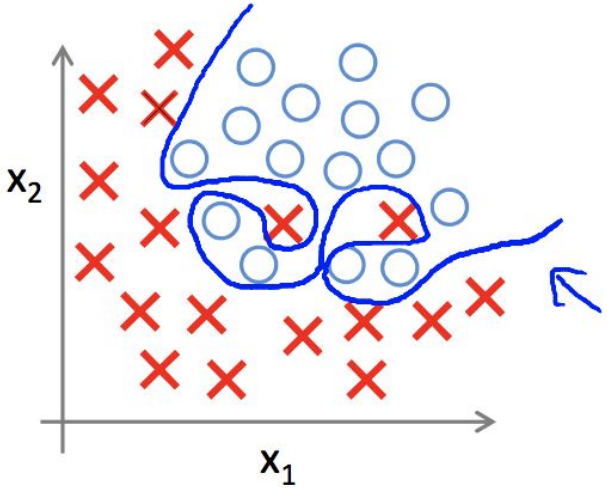

$h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2)$

($g$ = sigmoid function)

"Underfit"




$g(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 + \theta_4 x_2^2 + \theta_5 x_1 x_2)$

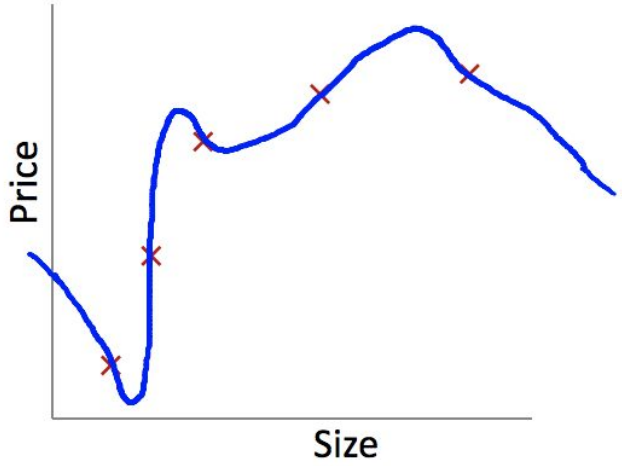



$g(\theta_0 + \theta_1 x_1 + \theta_2 x_1^2 + \theta_3 x_1^2 x_2 + \theta_4 x_1^2 x_2^2 + \theta_5 x_1^2 x_2^3 + \theta_6 x_1^3 x_2 + \dots)$

"Overfit"

## Addressing overfitting:

$x_1 =$ size of house
$x_2 =$ no. of bedrooms
$x_3 =$ no. of floors
$x_4 =$ age of house
$x_5 =$ average income in neighborhood
$x_6 =$ kitchen size
$\vdots$
$x_{100}$

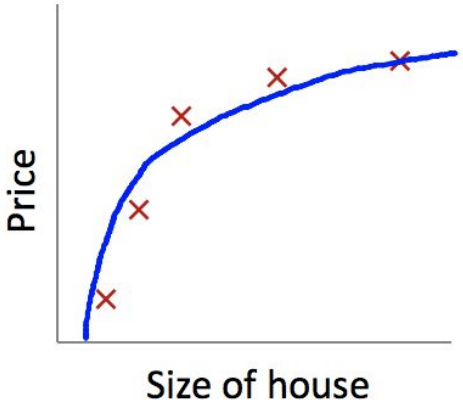


## Addressing overfitting:

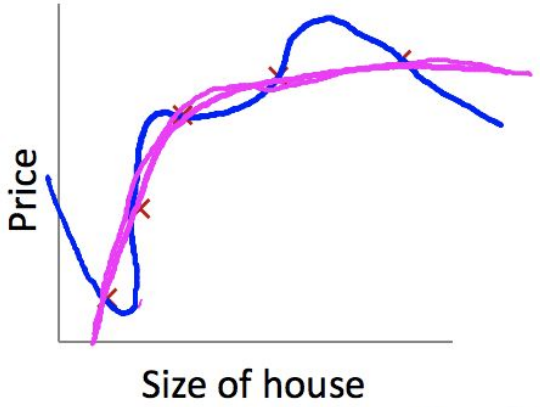
Options:

1. Reduce number of features.
   - — Manually select which features to keep.
   - — Model selection algorithm (later in course).
2. Regularization.
   - — Keep all the features, but reduce magnitude/values of parameters $\theta_j$.
   - — Works well when we have a lot of features, each of which contributes a bit to predicting $y$.

# Intuition




$$\theta_0 + \theta_1 x + \theta_2 x^2$$




$$\theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$$

Suppose we penalize and make $\theta_3, \theta_4$ really small.

$$\min_\theta \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2 + 1000\,\theta_3^2 + 1000\,\theta_4^2$$

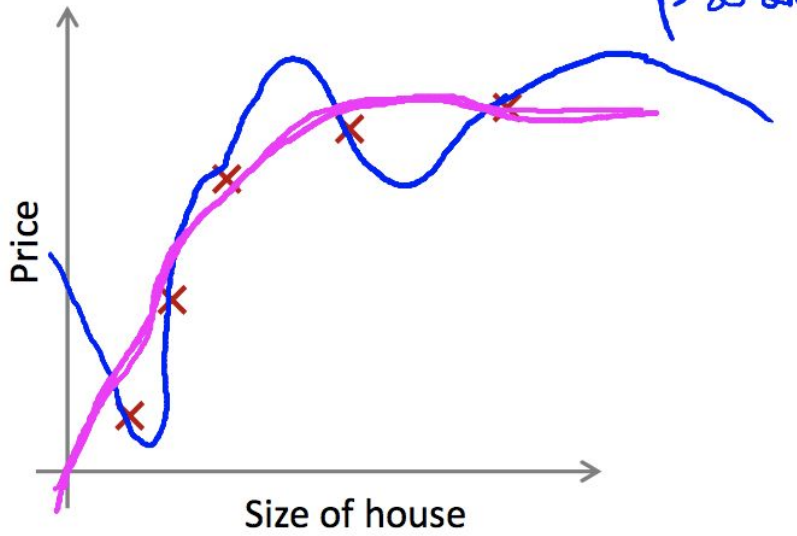
$\theta_3 \approx 0$ $\theta_4 \approx 0$

## Regularization.

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

regularization parameter

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



In regularized linear regression, we choose $\theta$ to minimize

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

What if $\lambda$ is set to an extremely large value (perhaps for too large for our problem, say $\lambda = 10^{10}$)?

- Algorithm works fine; setting $\lambda$ to be very large can't hurt it
- Algortihm fails to eliminate overfitting.
- Algorithm results in underfitting. (Fails to fit even training data well).
- Gradient descent will fail to converge.

**Addressing overfitting**:

1) Reduce the number of features:

a) Manually select which features to keep.

b) Use a model selection algorithm (studied later in the course).

2) Regularization

Keep all the features, but reduce the parameters $\theta_j$.