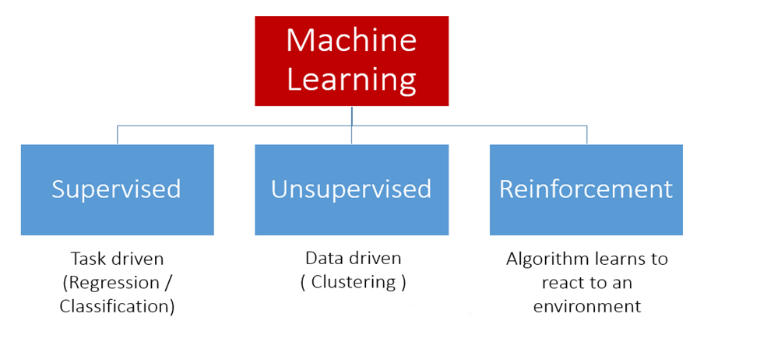
**Machine Learning**

Machine learning is an application of artificial intelligence (AI) that provides systems the ability to automatically learn and improve from experience without being explicitly programmed.

**Machine learning focuses on the development of computer programs** that can access data and use it learn for themselves.



For example, as a human,

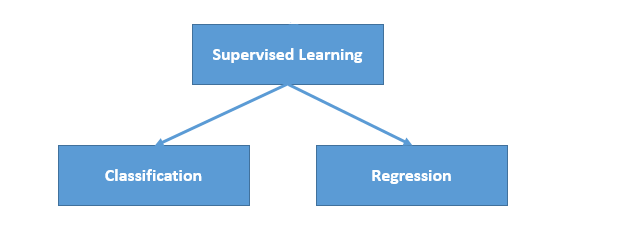
We can learn from classroom teaching (supervised learning),

From reading books (unsupervised learning),

From the environment based on various feedback such as reward, punishment (reinforcement learning).

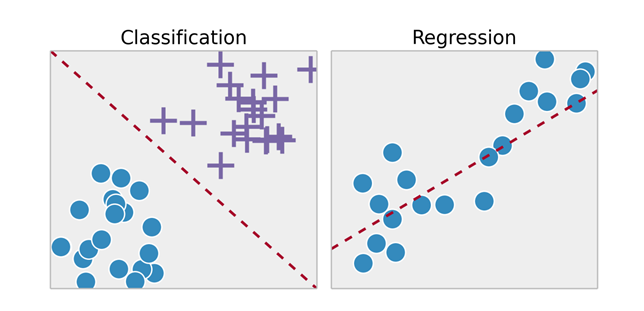
### ****Supervised Learning****

We are given a data set and already know what our correct output should look like, assuming that there is a relationship between the input and the output.



In a regression problem, we are trying to predict results within a continuous output. In a classification problem, we are instead trying to predict results in a discrete output.

Math is Fun on Continuous and Discrete Data.

.

**Example 1:**

Given data about the size of houses on the real estate market, try to predict their price. Price as a function of size is a continuous output, so this is a regression problem.

**Example 2**:

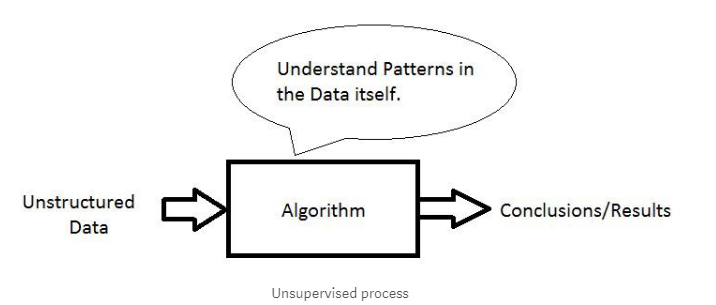
(a) Regression - Given a picture of Male/Female, We have to predict his/her age on the basis of given picture.

(b) Classification - Given a picture of Male/Female, We have to predict whether He/ She is of High school, College, Graduate age.

**Unsupervised Learning**

Like learning without a teacher.

Unsupervised learning, on the other hand, allows us to approach problems with little or no idea what our results should look like. We can derive structure from data where we don't necessarily know the effect of the variables.

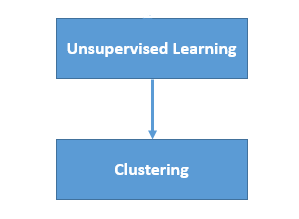


**Example:**

You have a bunch of photos of 6 people but without information about who is on which one and you want to divide this dataset into 6 piles, each with the photos of one individual.

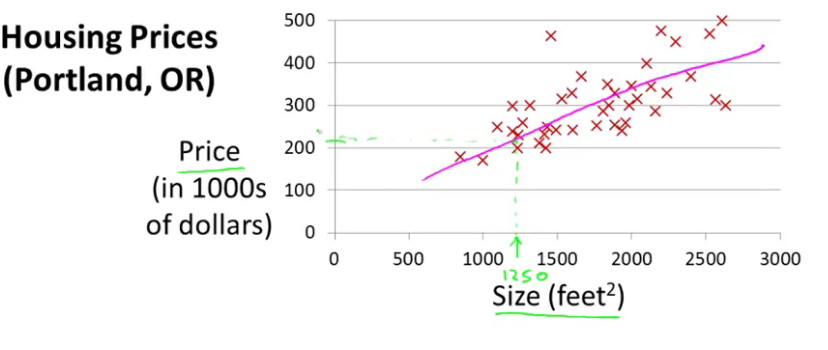
E.g. google photos

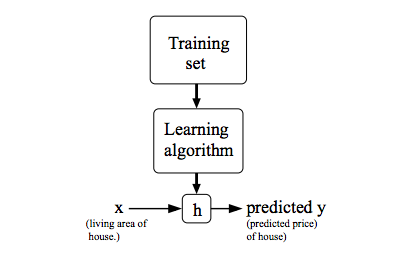
Clustering: Take a collection of 1000 essays written on the World Economy, and find a way to automatically group these essays into a small number that are somehow similar or related by different variables, such as word frequency, sentence length, page count, and so on.

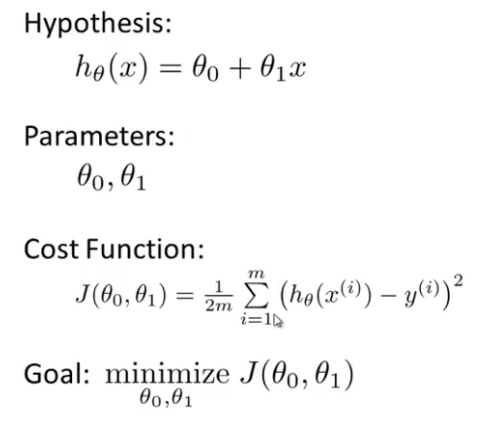


We can derive this structure by clustering the data based on relationships among the variables in the data.

***Example: Regression***

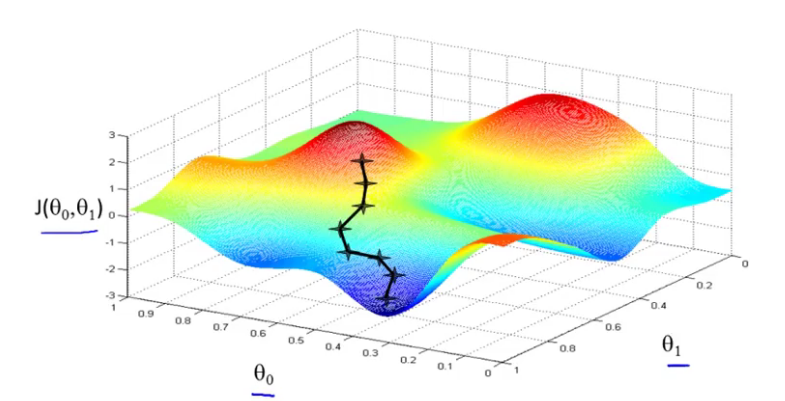


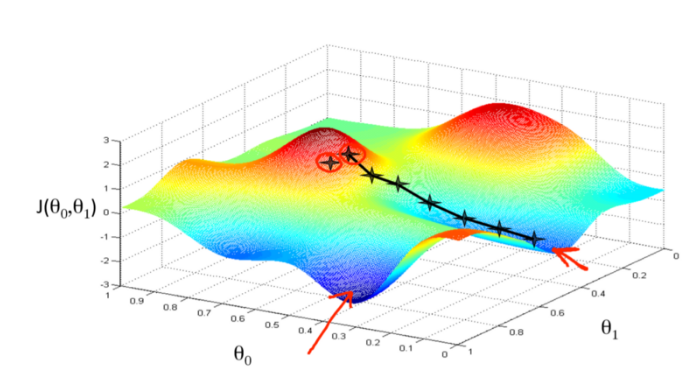




“Squared mean error value”



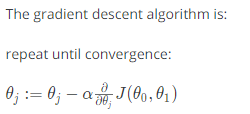




If we try to think of it in visual terms, our training data set is scattered on the x-y plane. We are trying to make straight line (defined by hθ(x)) which passes through this scattered set of data. Our objective is to get the best possible line. The best possible line will be such so that the average squared vertical distances of the scattered points from the line will be the least. In the best case, the line should pass through all the points of our training data set. In such a case the value of J(θ,θ1) will be 0.

**ML: Gradient Descent**

So we have our hypothesis function and we have a way of measuring how well it fits into the data. Now we need to estimate the parameters in hypothesis function. That's where gradient descent comes in.

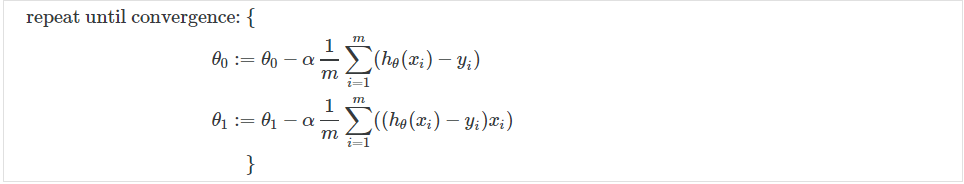


Where j=0, 1 represents the feature index number.

Alpha- Learning Rate

### ****Gradient Descent for Linear Regression****

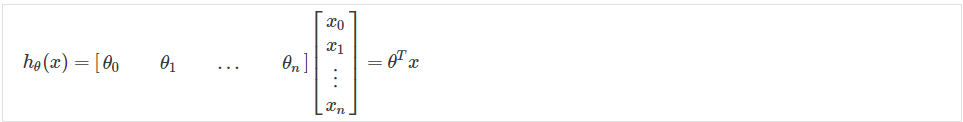
When specifically applied to the case of linear regression, a new form of the gradient descent equation can be derived. We can substitute our actual cost function and our actual hypothesis function and modify the equation to (the derivation of the formulas are out of the scope of this course, but a really great one can be found here):

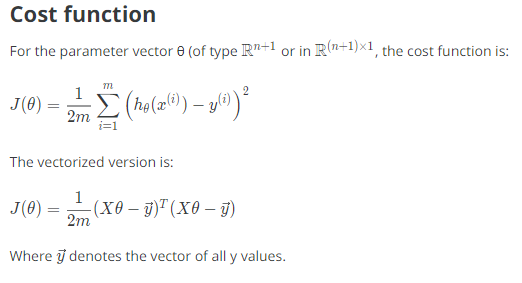


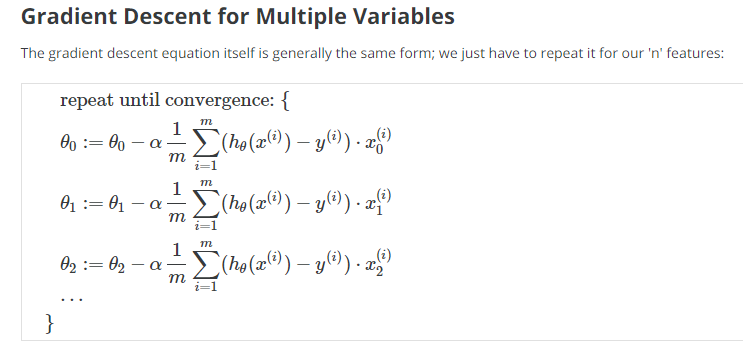
# Linear Regression with Multiple Variables:

# 









# Feature Normalization:

We can speed up gradient descent by having each of our input values in roughly the same range. The way to prevent this is to modify the ranges of our input variables so that they are all roughly the same. Ideally:

−1 ≤ *x* (*i*)​≤ 1 Or −0.5 ≤  *x*(*i*)​ ≤ 0.5

These aren't exact requirements; we are only trying to speed things up. The goal is to get all input variables into roughly one of these ranges, give or take a few.

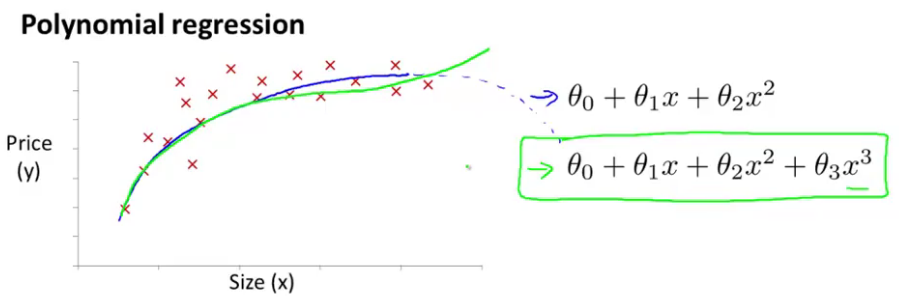
Two techniques to help with this are **feature scaling** and **mean normalization**.



Where *μi*​ is the **average** of all the values for feature (i) and *si*​ is the range of values (max - min), or *si*​ is the standard deviation.

# Polynomial Regression:

We can improve our features (with features normalization) and the form of our hypothesis function in a couple different ways.

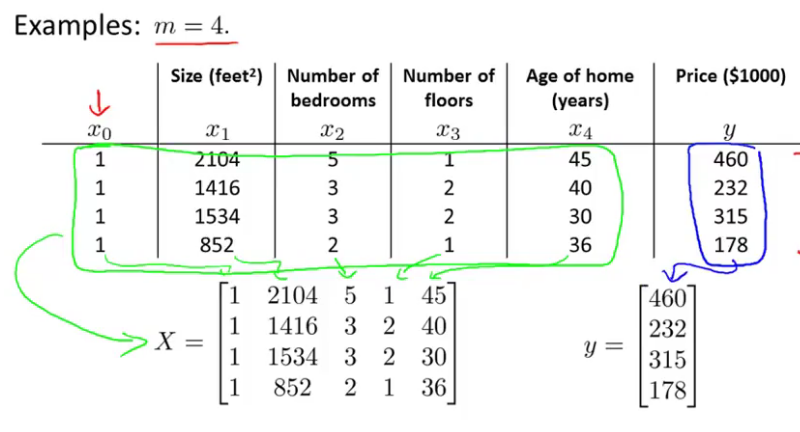


Our hypothesis function need not be linear (a straight line) if that does not fit the data well.

One important thing to keep in mind is, if you choose your features this way then feature scaling becomes very important.

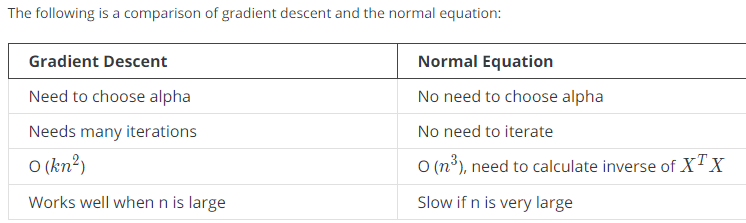
# Normal Equation:

The "Normal Equation" is a method of finding the optimum theta **without iteration.**





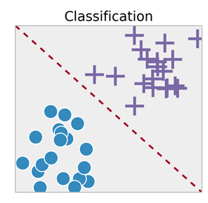
There is **no need** to do feature scaling with the normal equation.

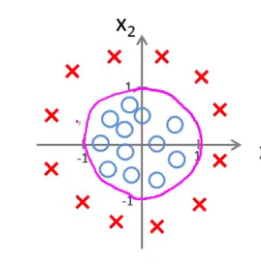


***Example: Classification***

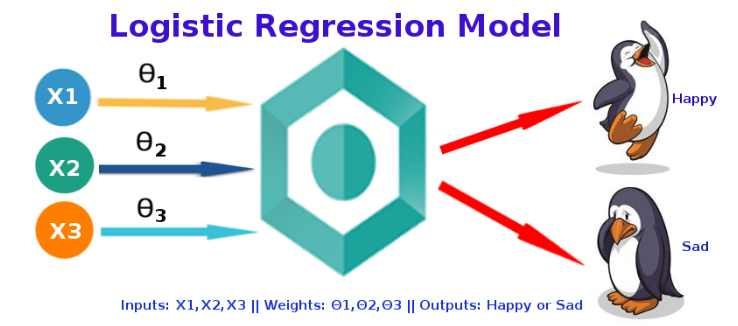
# ML: Logistic Regression

Now we are switching from regression problems to **classification problems**. Don't be confused by the name "Logistic Regression"; it is named that way for historical reasons and is actually an approach to classification problems, not regression problems.





**Binary Classification**



Instead of our output vector y being a continuous range of values, it will only be 0 or 1.



One-vs-All for multiclass classification:

Prominent applications of linear regression are:

1. Financial forecasting
2. Software cost prediction, software effort prediction and software quality assurance.
3. Restructuring of the budget : Organization or Country
4. Crime Data Mining: Predicting the crime rate of a states based on drug usage, number of gangs, human trafficking, and Killings.

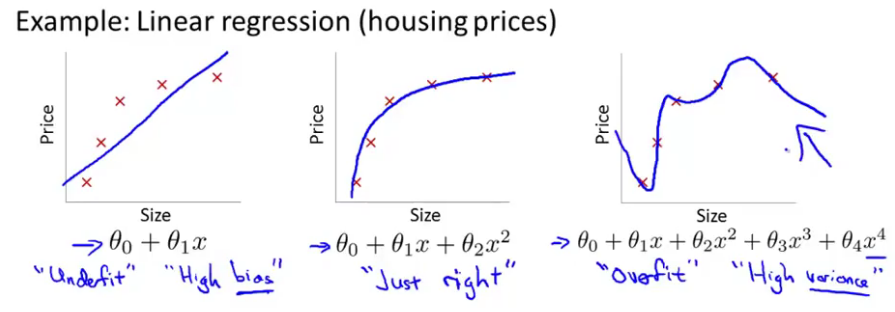
Applications of Logistic Regression:

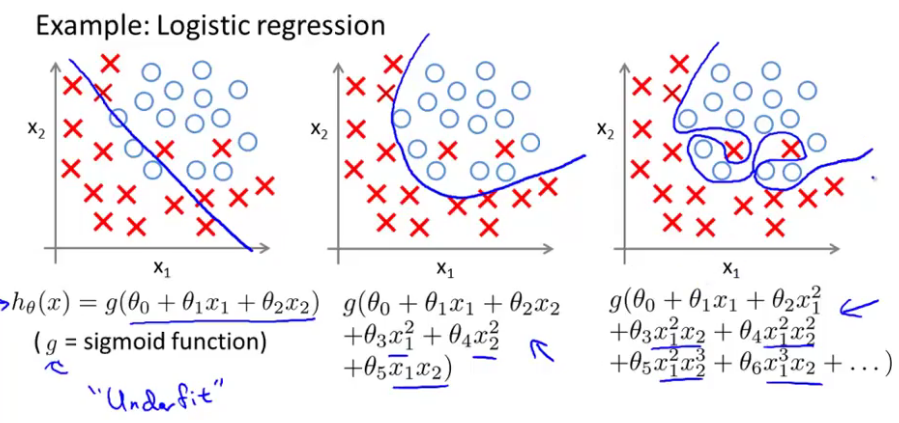
1. Image Segmentation and Categorization
2. Geographic Image Processing
3. Handwriting recognition
4. Prediction whether a person is depressed or not based on bag of words from the corpus seems to be conveniently solvable using logistic regression and SVM.

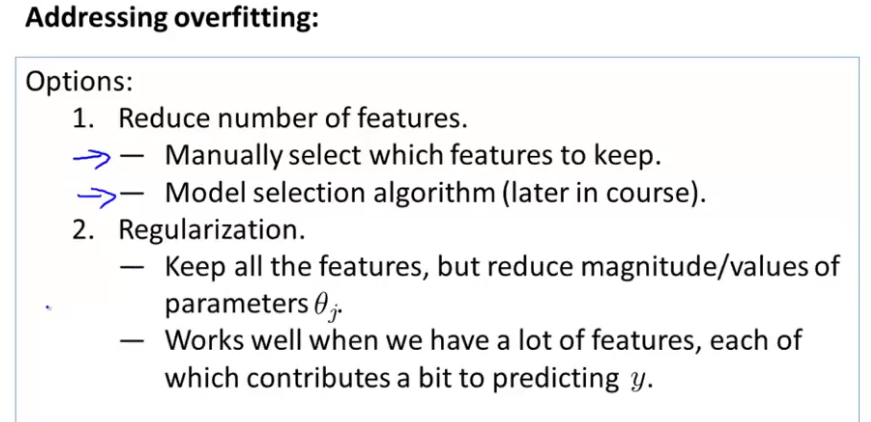
# ML: Regularization

**The Problem of Overfitting**

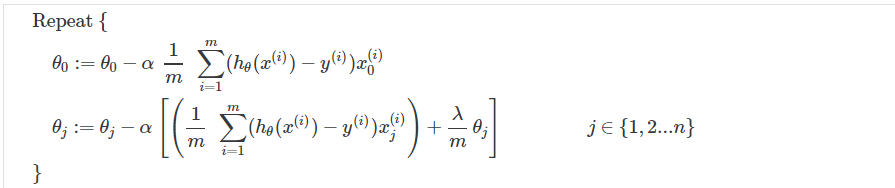
Regularization is designed to address the problem of overfitting.







# Regularized Linear Regression



# ML: Neural Networks: Representation

# Non-linear Hypotheses

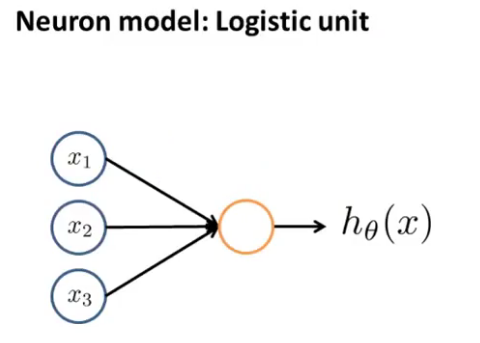
Neural networks offers an alternate way to perform machine learning when we have complex hypotheses with many features.

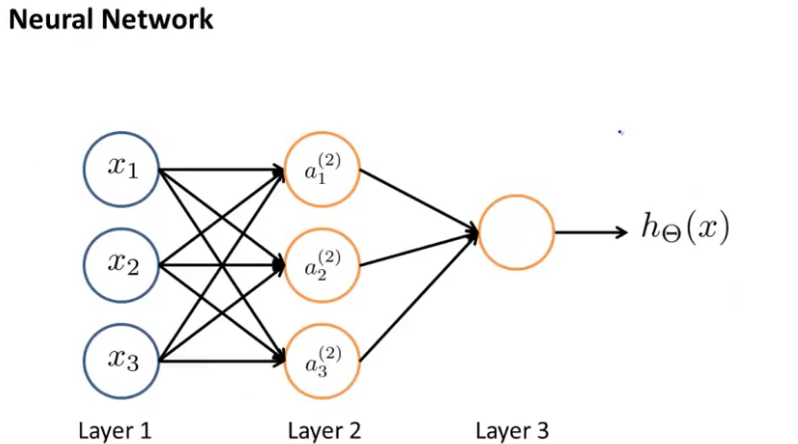
# Neurons and the Brain

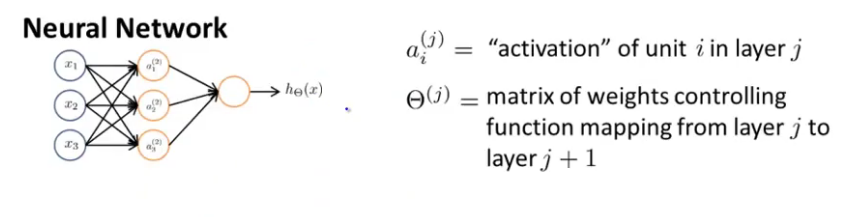
# 

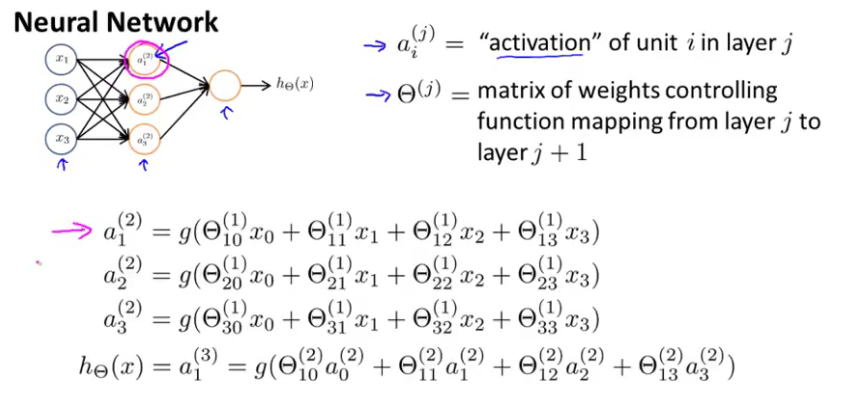
Neural networks are limited imitations of how our own brains work.

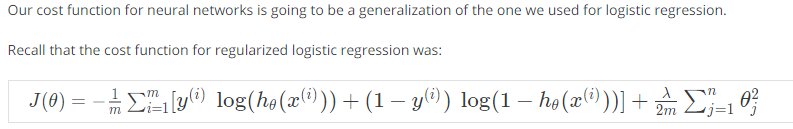
There is evidence that the brain uses only one "learning algorithm" for all its different functions. Scientists have tried cutting (in an animal brain) the connection between the ears and the auditory cortex and rewiring the optical nerve with the auditory cortex to find that the auditory cortex literally learns to see.

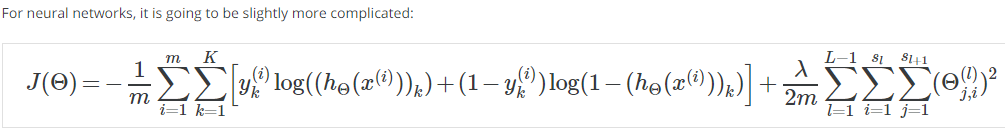












We have added a few nested summations to account for our multiple output nodes. In the first part of the equation, between the square brackets, we have an additional nested summation that loops through the number of output nodes.

.

# Backpropagation Algorithm

"Backpropagation" is neural-network terminology for minimizing our cost function, just like what we were doing with gradient descent in logistic and linear regression.

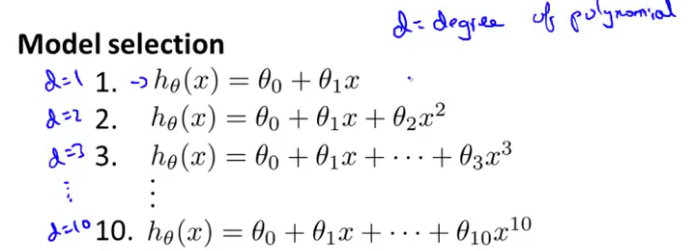
# ML: Advice for Applying Machine Learning

Errors in your predictions can be troubleshooted by:

* Getting more training examples
* Trying smaller sets of features
* Trying additional features
* Trying polynomial features
* Increasing or decreasing λ

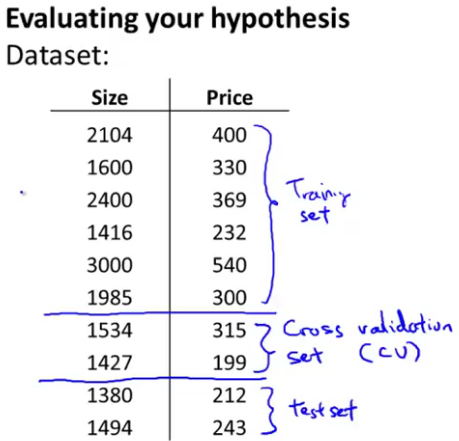
Don't just pick one of these avenues at random. We'll explore diagnostic techniques for choosing one of the above solutions in the following sections.

**Model Selection and Train/Validation/Test Sets:**



* Just because a learning algorithm fits a training set well, that does not mean it is a good hypothesis.
* The error of your hypothesis as measured on the data set with which you trained the parameters will be lower than any other data set.

In order to choose the model of your hypothesis, you can test each degree of polynomial and look at the error result.



One example way to break down our dataset into the three sets is:

* Training set: 60%
* Cross validation set: 20%
* Test set: 20%

We can now calculate three separate error values for the three different sets.

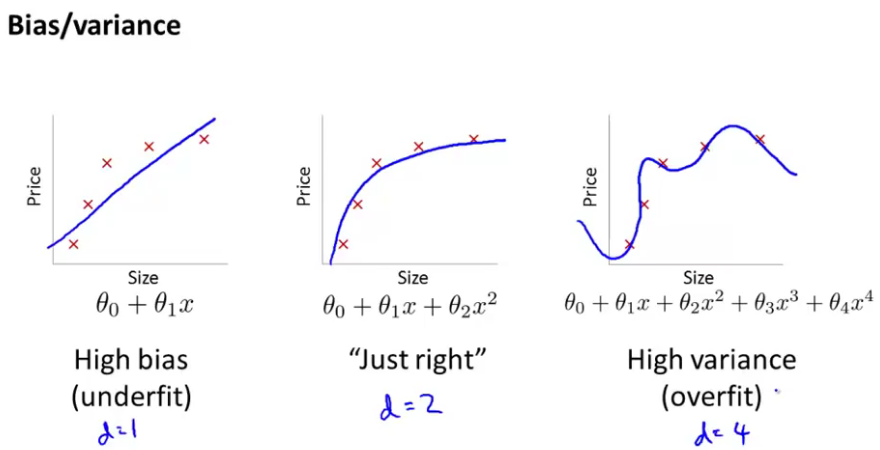
**With the Validation Set (note: this method presumes we do not use the CV set for regularization)**

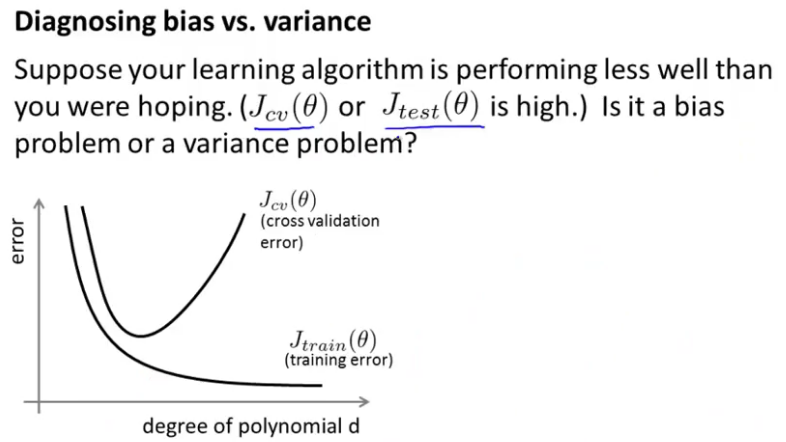
1. Optimize the parameters in Θ using the training set for each polynomial degree.
2. Find the polynomial degree d with the least error using the cross validation set.
3. Estimate the generalization error using the test set with *Jtest*​ (d = theta from polynomial with lower error)

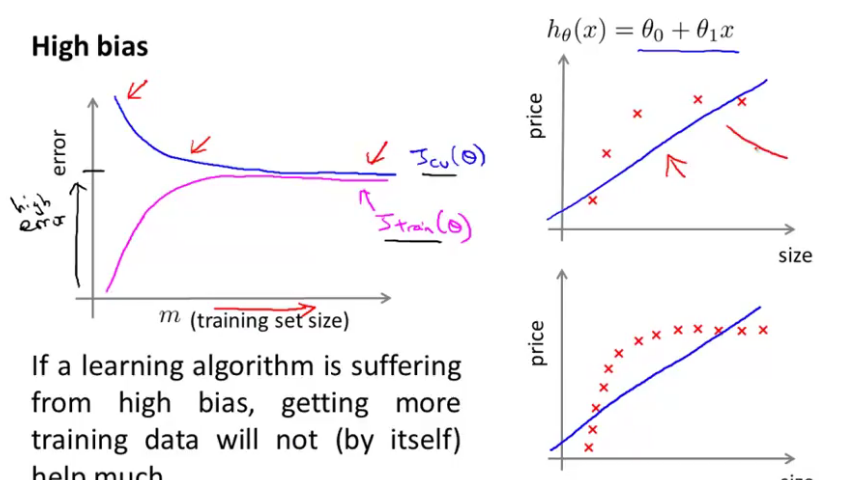
**Diagnosing Bias vs. Variance:**

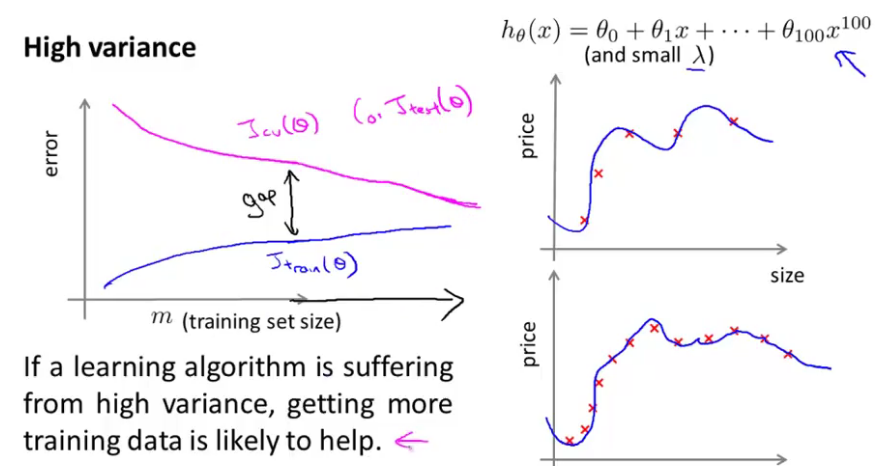
In this section we examine the relationship between the degree of the polynomial d and the underfitting or overfitting of our hypothesis.

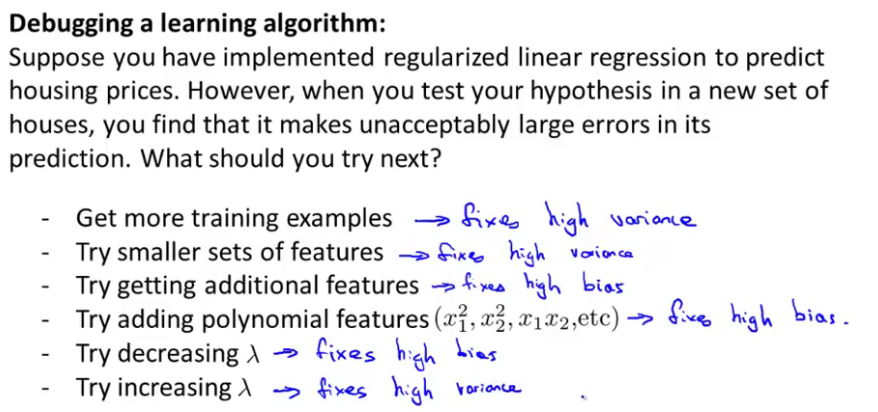
* We need to distinguish whether **bias** or **variance** is the problem contributing to bad predictions.
* High bias is underfitting and high variance is overfitting. We need to find a golden mean between these two.







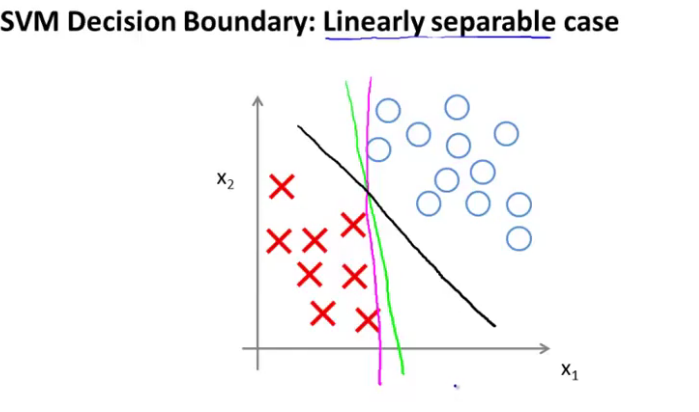


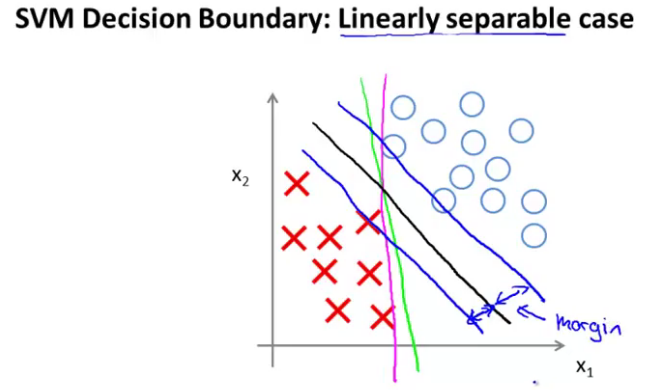


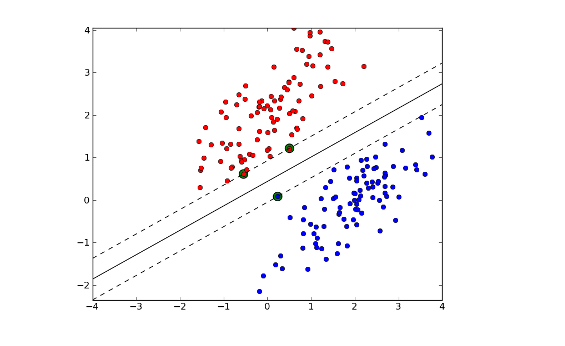
**Support Vector Machine (**Alternative view of logistic regression**)**

The **Support Vector Machine** (SVM) is yet another type of *supervised* machine learning algorithm. It is sometimes cleaner and more powerful. Used in classification problem.

A Support Vector Machine (SVM) is a discriminative classifier formally defined by a separating hyperplane.



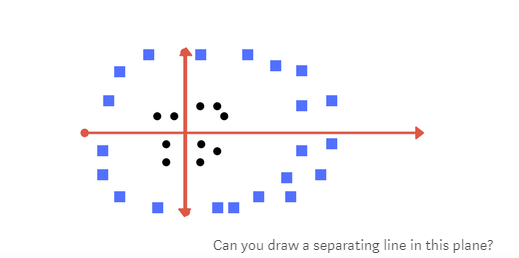




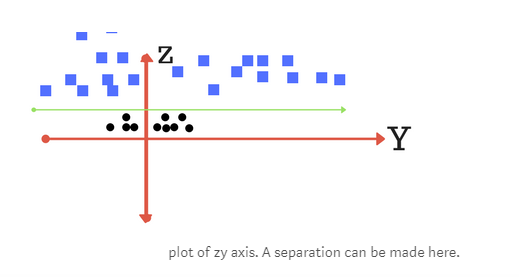
A useful way to think about Support Vector Machines is to think of them as Large Margin Classifiers.

The distance of the decision boundary to the nearest example is called the **margin**. Since SVMs maximize this margin, it is often called a *Large Margin Classifier*.

**If Non-Linear separable case**



So what do we do? We apply transformation and add one more dimension as we call it z-axis. Let’s assume value of points on z plane, w = x² + y². In this case we can manipulate it as distance of point from z-origin. Now if we plot in z-axis, a clear separation is visible and a line can be draw.



Thankfully, you don’t have to guess/ derive the transformation every time for your data set. The sklearn library's SVM implementation provides it inbuilt.

Polynomial and exponential kernels calculates separation line in higher dimension. This is called **kernel trick**

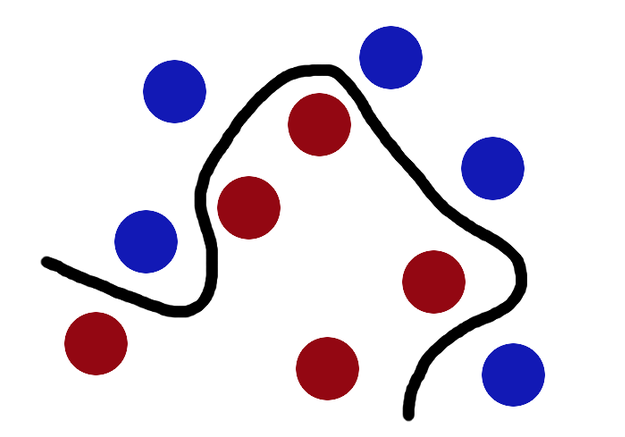
**Kernel:**

In its simplest form, the kernel trick means [transforming data](https://en.wikipedia.org/wiki/Data_transformation) into another dimension that has a clear dividing margin between classes of data.

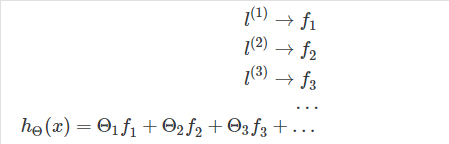
In machine learning, a “kernel” is usually used to refer to the kernel trick, a method of using a linear classifier to solve a non-linear problem.



**Why kernels?**  
Why kernels, as opposed to feature vectors? One big reason is that in many cases, computing the kernel is easy, but computing the feature vector corresponding to the kernel is really hard.



Gaussian kernels, mathematically, maps our data into an infinite dimension space (yes I did NOT misspell it). These were just out of the many kernels available. In this way, we get our solution from a higher dimensional space without even visiting it.



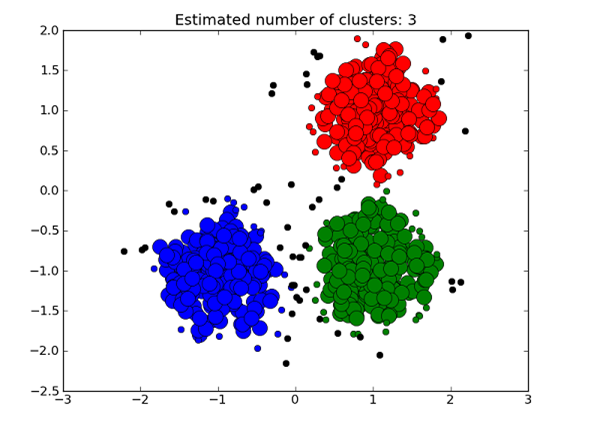
### ****Logistic Regression vs. SVMs****

1. If n is large (relative to m), then use logistic regression, or SVM without a kernel (the "linear kernel")
2. If n is small and m is intermediate, then use SVM with a Gaussian Kernel
3. If n is small and m is large, then manually create/add more features, then use logistic regression or SVM without a kernel.

**Note**: a neural network is likely to work well for any of these situations, but may be slower to train.

**Unsupervised Learning:**

**ML: Clustering**



Unsupervised learning is contrasted from supervised learning because it uses an **unlabeled** training set rather than a labeled one.

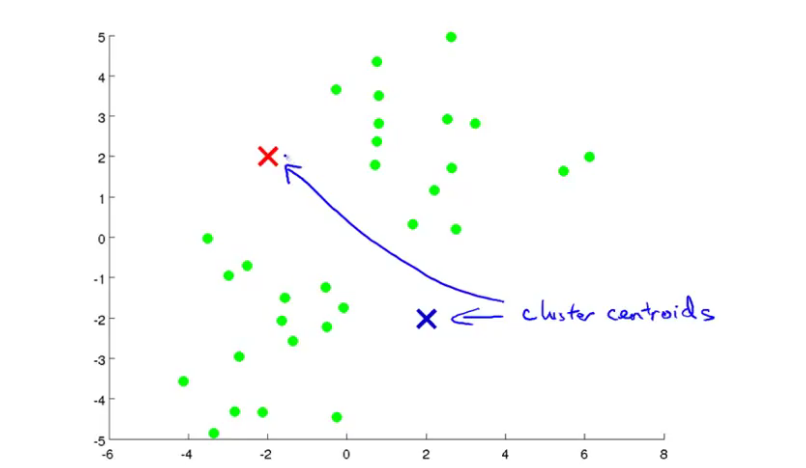
Clustering is good for:

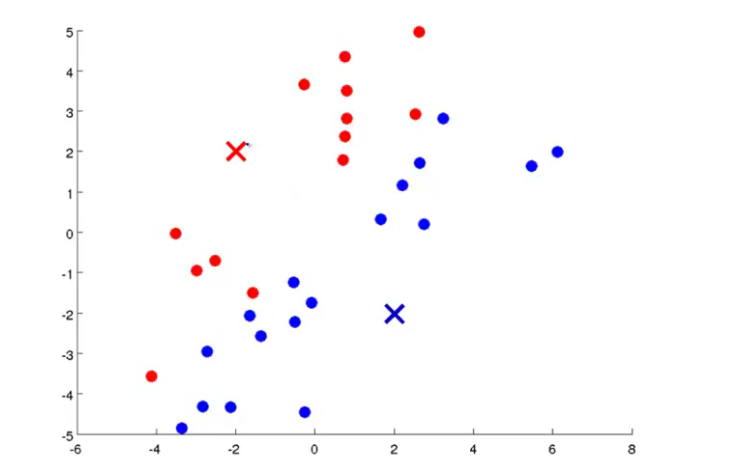
* Market segmentation
* Social network analysis
* Organizing computer clusters
* Astronomical data analysis

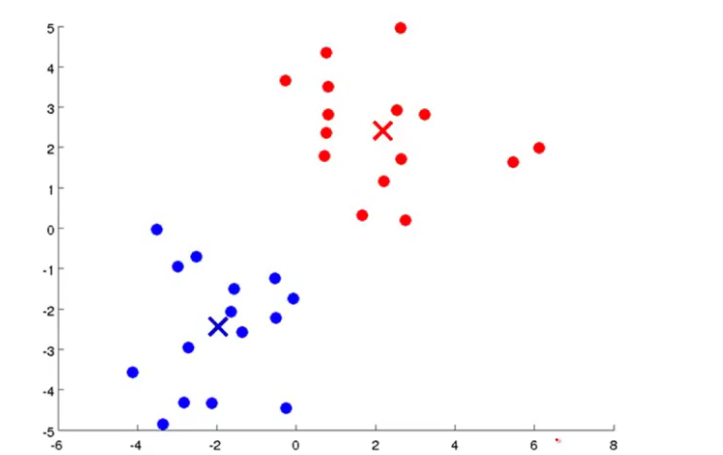
**K-Means Algorithm**

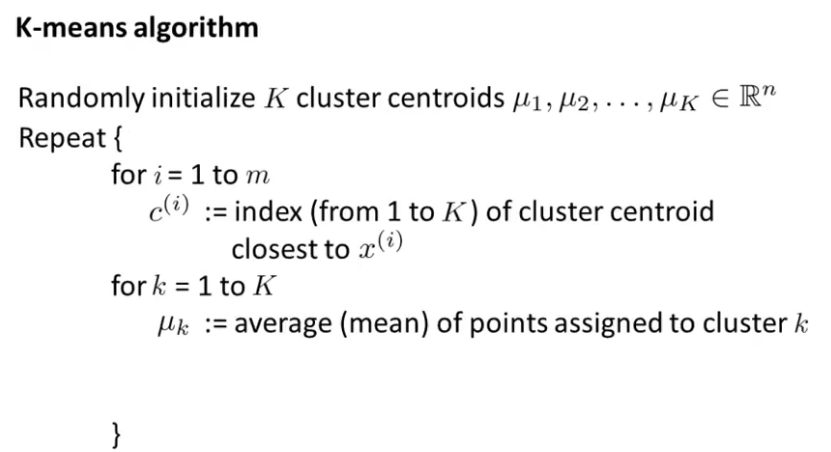
The K-Means Algorithm is the most popular and widely used algorithm for automatically grouping data into coherent subsets.

1. Randomly initialize two points in the dataset called the *cluster centroids*.
2. Cluster assignment: assign all examples into one of two groups based on which cluster centroid the example is closest to.
3. Move centroid: compute the averages for all the points inside each of the two cluster centroid groups, then move the cluster centroid points to those averages.
4. Re-run (2) and (3) until we have found our clusters.



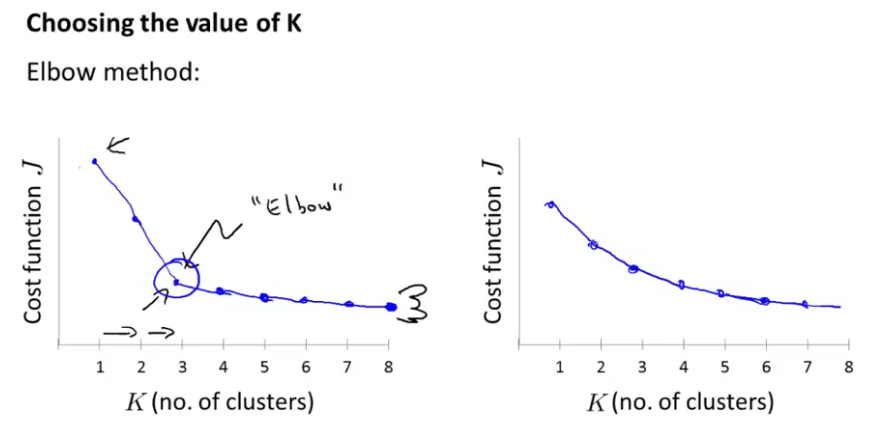


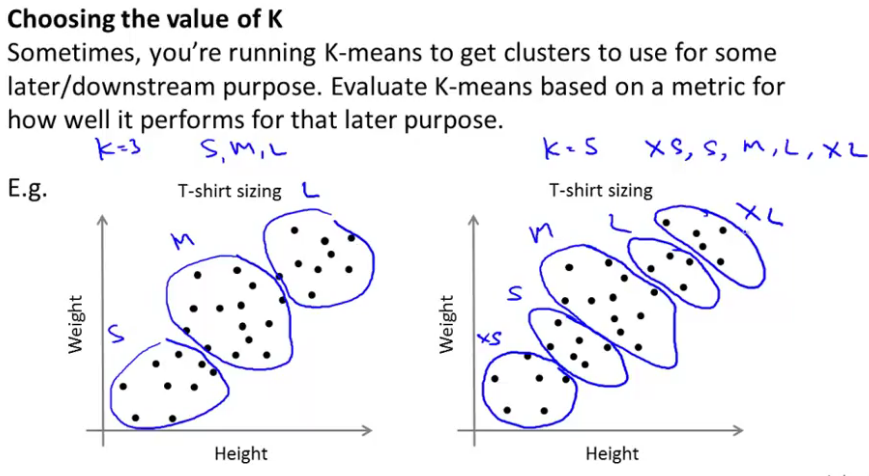




# Choosing the Number of Clusters

**The elbow method**: plot the cost J and the number of clusters K. The cost function should reduce as we increase the number of clusters, and then flatten out. Choose K at the point where the cost function starts to flatten out.





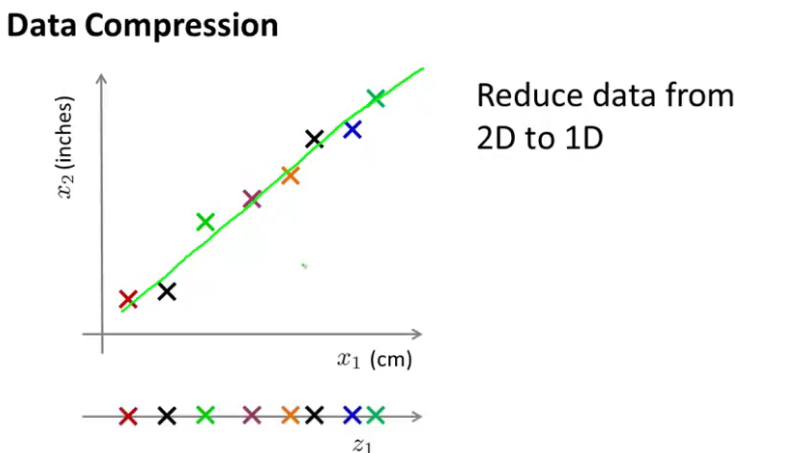
Mathematical expression are there to evaluate the number of cluster.

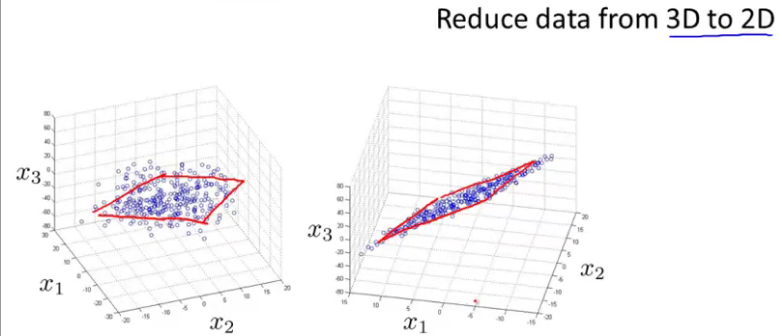
# ML: Dimensionality Reduction

**Data Compression**

* We may want to reduce the dimension of our features if we have a lot of redundant data.
* To do this, we find two highly correlated features, plot them, and make a new line that seems to describe both features accurately. We place all the new features on this single line.

Doing dimensionality reduction will reduce the total data we have to store in computer memory and will speed up our learning algorithm.

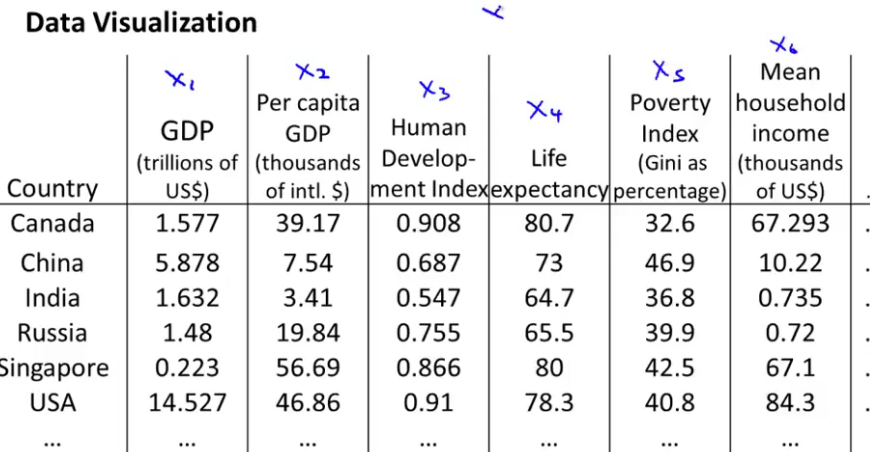


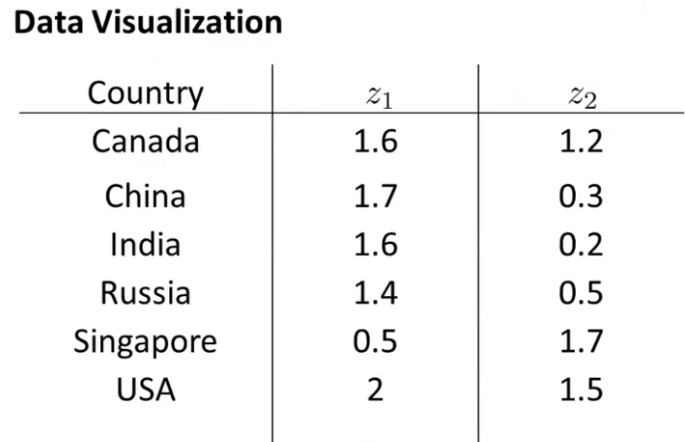


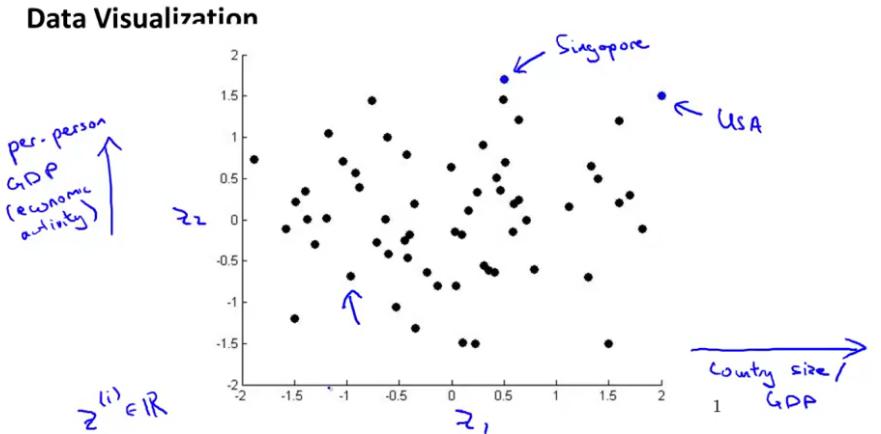
### ****Data Visualization****

It is not easy to visualize data that is more than three dimensions. We can reduce the dimensions of our data to 3 or less in order to plot it.

Example: hundreds of features related to a country's economic system may all be combined into one feature that you call "Economic Activity."

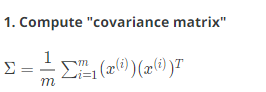






# Principal Component Analysis Algorithm

# 





[U,S,V] = svd(Sigma);

**3. Take the first k columns of the U matrix and compute z**

We'll assign the first k columns of U to a variable called 'Ureduce'. This will be an n×k matrix. We compute z with:



**Applications**

* Compressions : Reduce space of data, Speed up algorithm
* Visualization of data (Choose k = 2 or k = 3)

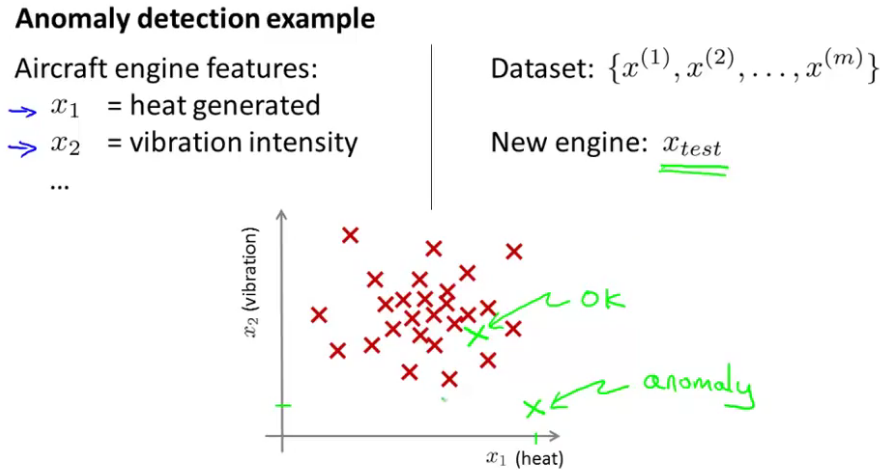
Don't assume you need to do PCA. **Try your full machine learning algorithm without PCA first.** Then use PCA if you find that you need it.

# ML: Anomaly Detection:

Just like in other learning problems, we are given a dataset

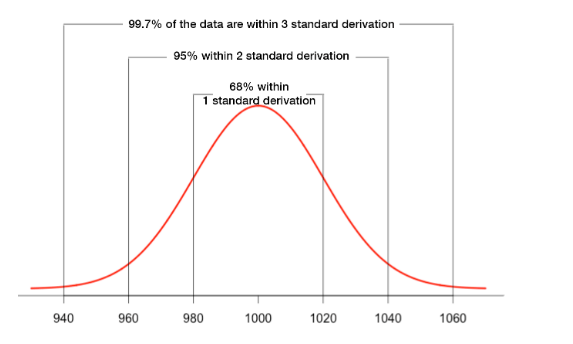
We are then given a new example, *xtest*​ and we want to know whether this new example is abnormal/anomalous.

We define a "model" p(x) that tells us the probability the example is not anomalous.



**The normal distribution** is the holy grail of anomaly detection. Normally distributed metrics follow a set of probabilistic rules. Values that follow those rules are recognized as being “normal” or “usual”, while values that break them are seen as being unusual, indicating anomalies.

**A normal distribution** is a very common probability distribution that approximates the behaviour of many natural phenomena. A data set is known as “normally distributed” when most of the data aggregates around its mean in a symmetric fashion. Values become less and less likely to occur the farther they are from the mean

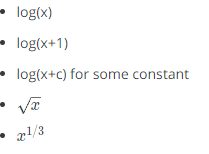


# Choosing What Features to Use

The features will greatly affect how well your anomaly detection algorithm works.

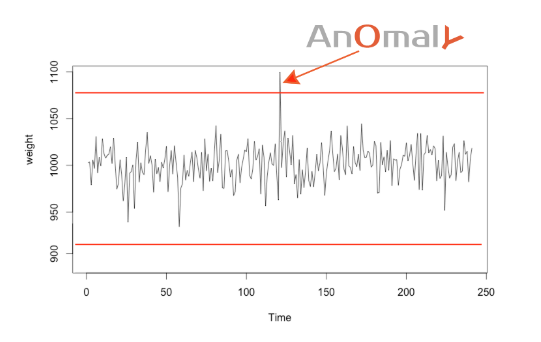
We can check that our features are Gaussian by plotting a histogram of our data and checking for the bell-shaped curve.

Some transforms we can try on an example feature x that does not have the bell-shaped curve are:

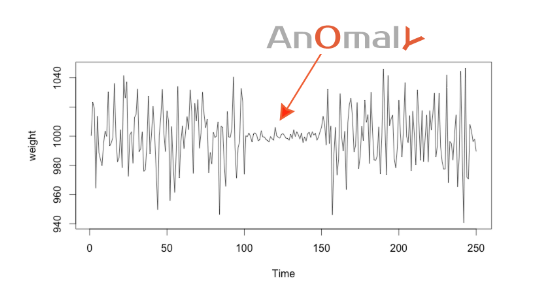


**When the Rules are Broken, It’s an Anomaly**

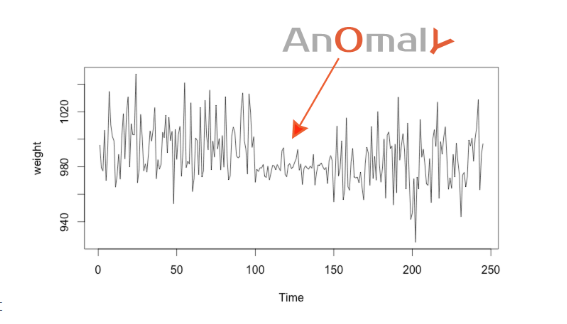
### Technique #1: Outlier Values

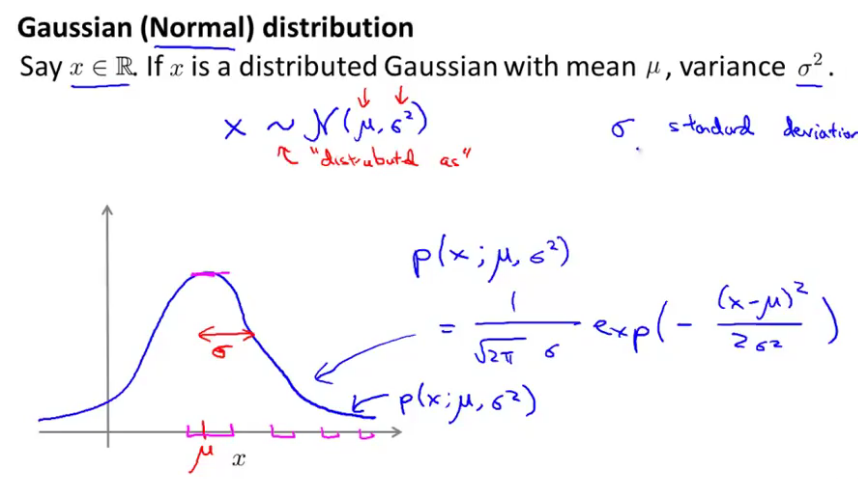


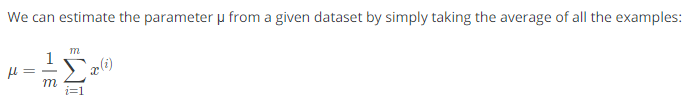
### Technique #2: Detecting Rare Distributions

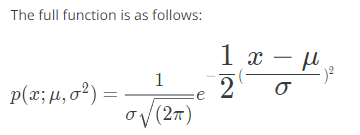


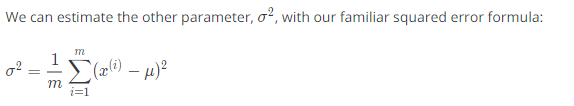
### Technique #3: Detecting a Change in the Normal Distribution

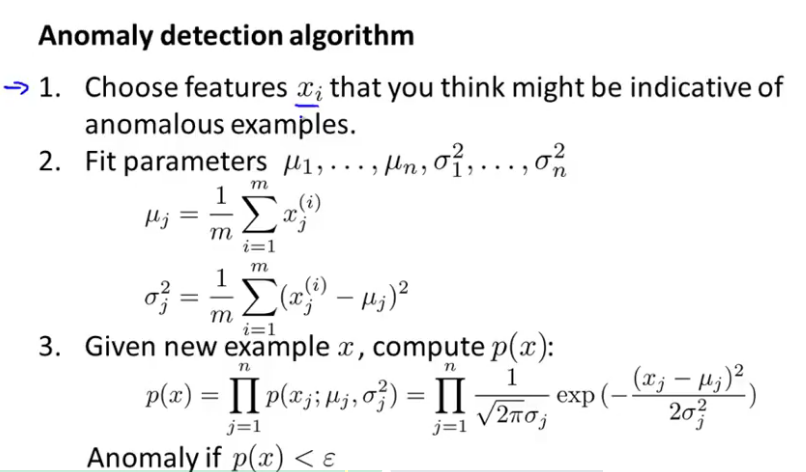


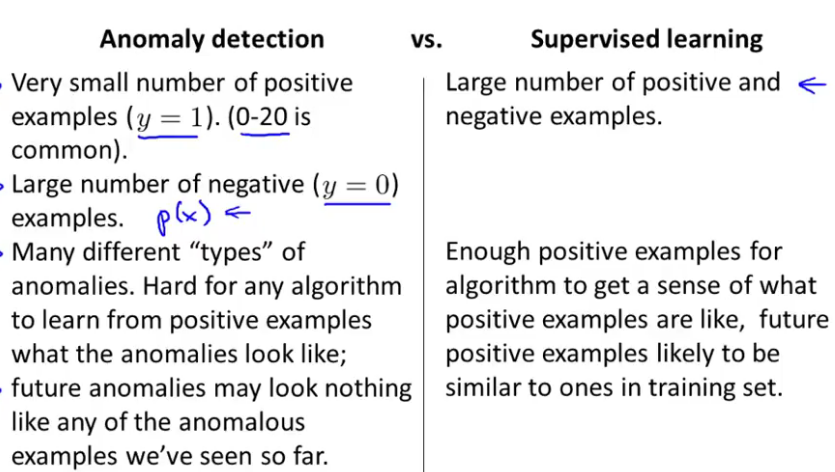


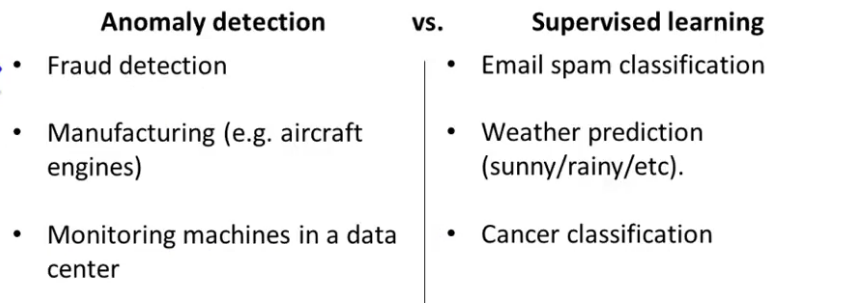










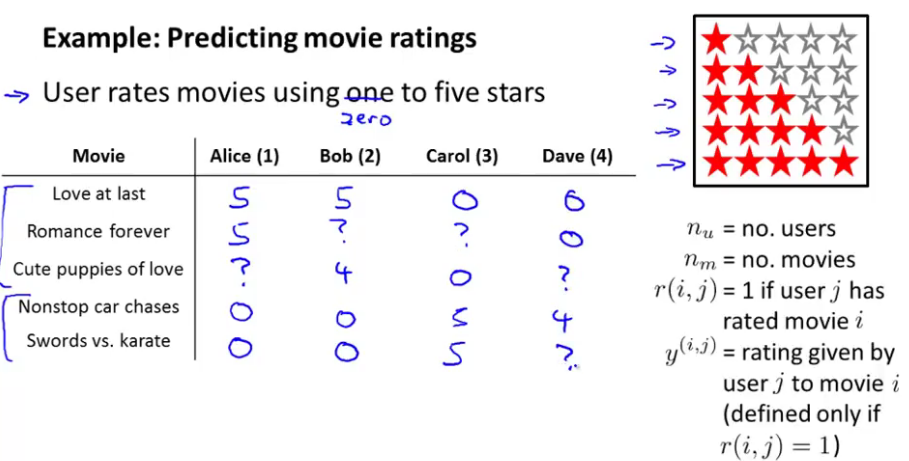


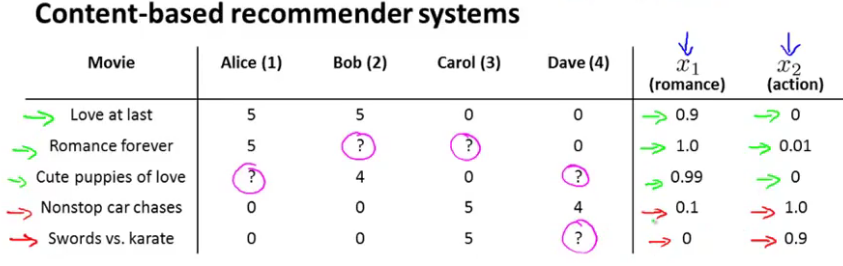
# ML: Recommender Systems:

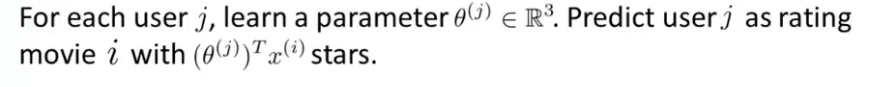
Recommendation is currently a very popular application of machine learning.

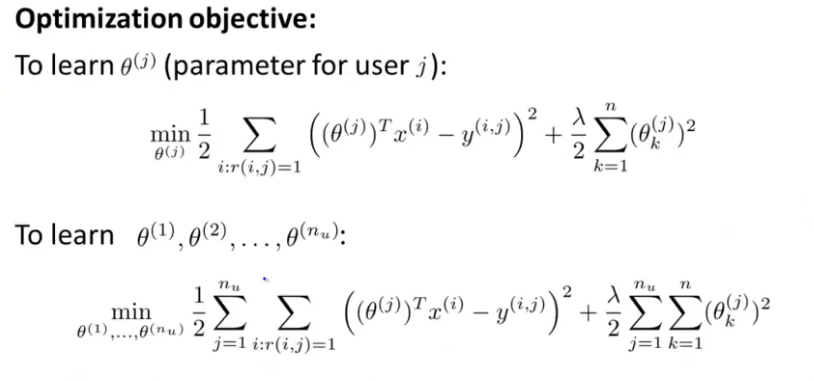
Say we are trying to recommend movies to customers. We can use the following definitions

# Content Based Recommendations









# Collaborative Filtering

# Vectorization: Low Rank Matrix Factorization

# Implementation Detail: Mean Normalization