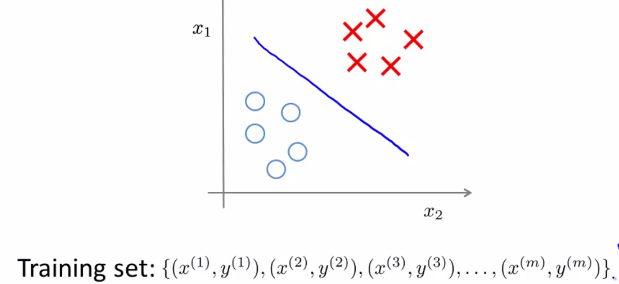
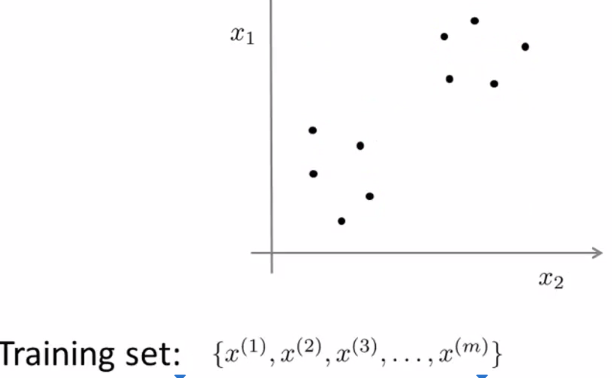
***Clustering***

**I. UNSUPERVISED LEARNING INTRO**

* Clustering will be our first unsupervised learning algorithm
* A typical supervised learning problem = *given* a LABELED training set w/ the goal of finding the decision boundary that separates  positive label + negative label examples.
* Given a set of labels to fit a hypothesis to it.



* In contrast, in an unsupervised learning problem, we're given data that does NOT have any labels associated w/ it. So, we're given data that looks like this.



* See a set of points x w/ no labels y (no colors or shapes)
* In unsupervised learning, we give this unlabeled training set to an algorithm + ask it to find some structure in the data for us.
* 1 type of structure we might have an algorithm find for this data set a grouping into 2 separate clusters + so an algorithm that finds clusters like the ones I've just circled is called a clustering algorithm
* **Clustering** is good for:
* **Market segmentation** 🡪 may have a database of customers + want to group them into different segments so you can sell to them separately or serve different segments better.
* **Social network analysis** 🡪 things like Facebook, Google+, or info about people you email the most frequently to find coherence in groups of people
* Who are the coherent groups of friends in the social network?
* **To organize computing clusters/data centers better** 🡪 If you know which CPUs in a data center tend to work together, you can use that to reorganize resources + how we lay out the network + design the data center communications more efficiently
* **To understand galaxy formation** + using that to understand astronomical data.

**II. K-MEANS ALGORITHM**

* **The K Means algorithm** is by far the most widely used clustering algorithm
* The K means clustering algorithm is best illustrated in pictures. Let's say I want to take an unlabeled data set like the one shown here, + I want to group the data into two clusters.
* 0:37
* If I run the K Means clustering algorithm, here is what I'm going to do. The first step is to randomly initialize two points, called the cluster centroids. So, these two crosses here, these are called the Cluster Centroids
* 0:53
* + I have two of them b/c I want to group my data into two clusters.
* 0:59
* K Means is an iterative algorithm + it does two things.
* 1:03
* First is a cluster assignment step, + second is a move centroid step. So, let me tell you what those things mean.
* 1:11
* The first of the two steps in the loop of K means, is this cluster assignment step. What that means is that, it's going through each of the examples, each of these green dots shown here + depending on whether it's closer to the red cluster centroid or the blue cluster centroid, it is going to assign each of the data points to one of the two cluster centroids.
* 1:32
* Specifically, what I mean by that, is to go through your data set + color each of the points either red or blue, depending on whether it is closer to the red cluster centroid or the blue cluster centroid, + I've done that in this diagram here.
* 1:46
* So, that was the cluster assignment step.
* 1:49
* The other part of K means, in the loop of K means, is the move centroid step, + what we are going to do is, we are going to take the two cluster centroids, that is, the red cross + the blue cross, + we are going to move them to the average of the points colored the same colour. So what we are going to do is look at all the red points + compute the average, really the mean of the location of all the red points, + we are going to move the red cluster centroid there. + the same things for the blue cluster centroid, look at all the blue dots + compute their mean, + then move the blue cluster centroid there. So, let me do that now. We're going to move the cluster centroids as follows
* 2:24
* + I've now moved them to their new means. The red one moved like that + the blue one moved like that + the red one moved like that. + then we go back to another cluster assignment step, so we're again going to look at all of my unlabeled examples + depending on whether it's closer the red or the blue cluster centroid, I'm going to color them either red or blue. I'm going to assign each point to one of the two cluster centroids, so let me do that now.
* 2:51
* + so the colors of some of the points just changed.
* 2:53
* + then I'm going to do another move centroid step. So I'm going to compute the average of all the blue points, compute the average of all the red points + move my cluster centroids like this, + so, let's do that again. Let me do one more cluster assignment step. So colour each point red or blue, based on what it's closer to + then do another move centroid step + we're done. + in fact if you keep running additional iterations of K means from here the cluster centroids will not change any further + the colours of the points will not change any further. + so, this is the, at this point, K means has converged + it's done a pretty good job finding
* 3:37
* the two clusters in this data. Let's write out the K means algorithm more formally.
* 3:42
* The K means algorithm takes two inputs. One is a parameter K, which is the number of clusters you want to find in the data. I'll later say how we might go about trying to choose k, but for now let's just say that we've decided we want a certain number of clusters + we're going to tell the algorithm how many clusters we think there are in the data set. + then K means also takes as input this sort of unlabeled training set of just the Xs + b/c this is unsupervised learning, we don't have the labels Y anymore. + for unsupervised learning of the K means I'm going to use the convention that XI is an RN dimensional vector. + that's why my training examples are now N dimensional rather N plus one dimensional vectors.
* 4:24
* This is what the K means algorithm does.
* 4:27
* The first step is that it randomly initializes k cluster centroids which we will call mu 1, mu 2, up to mu k. + so in the earlier diagram, the cluster centroids corresponded to the location of the red cross + the location of the blue cross. So there we had two cluster centroids, so maybe the red cross was mu 1 + the blue cross was mu 2, + more generally we would have k cluster centroids rather than just 2. Then the inner loop of k means does the following, we're going to repeatedly do the following.
* 5:00
* First for each of my training examples, I'm going to set this variable CI to be the index 1 through K of the cluster centroid closest to XI. So this was my cluster assignment step, where we took each of my examples + coloured it either red or blue, depending on which cluster centroid it was closest to. So CI is going to be a number from 1 to K that tells us, you know, is it closer to the red cross or is it closer to the blue cross,
* 5:32
* + another way of writing this is I'm going to, to compute Ci, I'm going to take my Ith example Xi + + I'm going to measure it's distance
* 5:43
* to each of my cluster centroids, this is mu + then lower-case k, right, so capital K is the total number centroids + I'm going to use lower case k here to index into the different centroids.
* 5:56
* But so, Ci is going to, I'm going to minimize over my values of k + find the value of K that minimizes this distance between Xi + the cluster centroid, + then, you know, the value of k that minimizes this, that's what gets set in Ci. So, here's another way of writing out what Ci is.
* 6:18
* If I write the norm between Xi minus Mu-k,
* 6:23
* then this is the distance between my ith training example Xi + the cluster centroid Mu subscript K, this is--this here, that's a lowercase K. So uppercase K is going to be used to denote the total number of cluster centroids, + this lowercase K's a number between one + capital K. I'm just using lower case K to index into my different cluster centroids.
* 6:47
* Next is lower case k. So
* 6:50
* that's the distance between the example + the cluster centroid + so what I'm going to do is find the value of K, of lower case k that minimizes this, + so the value of k that minimizes you know, that's what I'm going to set as Ci, + by convention here I've written the distance between Xi + the cluster centroid, by convention people actually tend to write this as the squared distance. So we think of Ci as picking the cluster centroid w/ the smallest squared distance to my training example Xi. But of course minimizing squared distance, + minimizing distance that should give you the same value of Ci, but we usually put in the square there, just as the convention that people use for K means. So that was the cluster assignment step.
* 7:33
* The other in the loop of K means does the move centroid step.
* 7:40
* + what that does is for each of my cluster centroids, so for lower case k equals 1 through K, it sets Mu-k equals to the average of the points assigned to cluster. So as a concrete example, let's say that one of my cluster centroids, let's say cluster centroid two, has training examples, you know, 1, 5, 6, + 10 assigned to it. + what this means is, really this means that C1 equals
* 8:06
* to C5 equals to
* 8:10
* C6 equals to + similarly well c10 equals, too, right?
* 8:14
* If we got that from the cluster assignment step, then that means examples 1,5,6 + 10 were assigned to the cluster centroid two.
* 8:24
* Then in this move centroid step, what I'm going to do is just compute the average of these four things.
* 8:31
* So X1 plus X5 plus X6 plus X10. + now I'm going to average them so here I have four points assigned to this cluster centroid, just take one quarter of that. + now Mu2 is going to be an n-dimensional vector. B/c each of these example x1, x5, x6, x10
* 8:52
* each of them were an n-dimensional vector, + I'm going to add up these things +, you know, divide by four b/c I have four points assigned to this cluster centroid, I end up w/ my move centroid step,
* 9:03
* for my cluster centroid mu-2. This has the effect of moving mu-2 to the average of the four points listed here.
* 9:12
* One thing that I've asked is, well here we said, let's let mu-k be the average of the points assigned to the cluster. But what if there is a cluster centroid no points w/ zero points assigned to it. In that case the more common thing to do is to just eliminate that cluster centroid. + if you do that, you end up w/ K minus one clusters
* 9:31
* instead of k clusters. Sometimes if you really need k clusters, then the other thing you can do if you have a cluster centroid w/ no points assigned to it is you can just randomly reinitialize that cluster centroid, but it's more common to just eliminate a cluster if somewhere during K means it w/ no points assigned to that cluster centroid, + that can happen, altthough in practice it happens not that often. So that's the K means Algorithm.
* 9:59
* Before wrapping up this video I just want to tell you about one other common application of K Means + that's to the problems w/ non well separated clusters.
* 10:08
* Here's what I mean. So far we've been picturing K Means + applying it to data sets like that shown here where we have three pretty well separated clusters, + we'd like an algorithm to find maybe the 3 clusters for us. But it turns out that very often K Means is also applied to data sets that look like this where there may not be several very well separated clusters. Here is an example application, to t-shirt sizing.
* 10:34
* Let's say you are a t-shirt manufacturer you've done is you've gone to the population that you want to sell t-shirts to, + you've collected a number of examples of the height + weight of these people in your population + so, well I guess height + weight tend to be positively highlighted so maybe you end up w/ a data set like this, you know, w/ a sample or set of examples of different peoples heights + weight. Let's say you want to size your t shirts. Let's say I want to design + sell t shirts of three sizes, small, medium + large. So how big should I make my small one? How big should I my medium? + how big should I make my large t-shirts.
* 11:10
* One way to do that would to be to run my k means clustering logarithm on this data set that I have shown on the right + maybe what K Means will do is group all of these points into one cluster + group all of these points into a second cluster + group all of those points into a third cluster. So, even though the data, you know, before hand it didn't seem like we had 3 well separated clusters, K Means will kind of separate out the data into multiple pluses for you. + what you can do is then look at this first population of people + look at them +, you know, look at the height + weight, + try to design a small t-shirt so that it kind of fits this first population of people well + then design a medium t-shirt + design a large t-shirt. + this is in fact kind of an example of market segmentation
* 12:01
* where you're using K Means to separate your market into 3 different segments. So you can design a product separately that is a small, medium, + large t-shirts,
* 12:09
* that tries to suit the needs of each of your 3 separate sub-populations well. So that's the K Means algorithm. + by now you should know how to implement the K Means Algorithm + kind of get it to work for some problems. But in the next few videos what I want to do is really get more deeply into the nuts + bolts of K means + to talk a bit about how to actually get this to work really well.

**III. OPTIMIZATION OBJECTIVE**

**IV. RANDOM INITIALIZATION**

**V. CHOOSING THE NUMBER OF CLUSTERS**