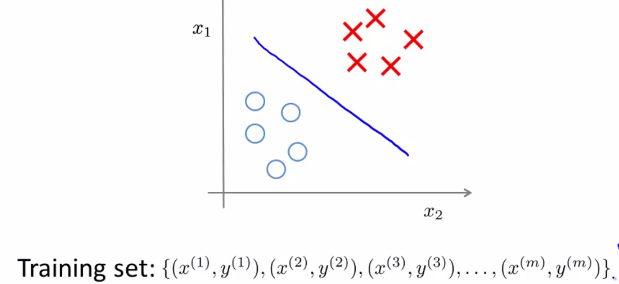
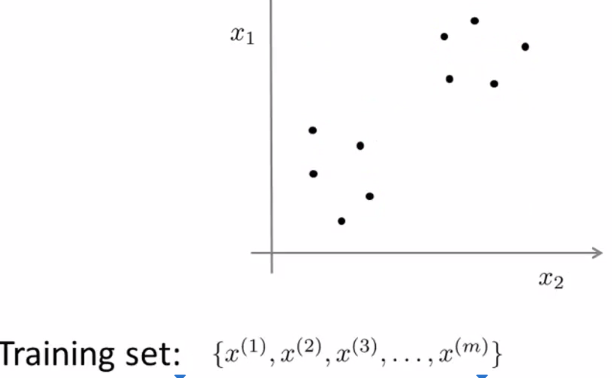
***Clustering***

**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 dec(i)sion boundary that separates  positive label + negative label examples.
* Given a set of labels to fit a hypothesis to it.



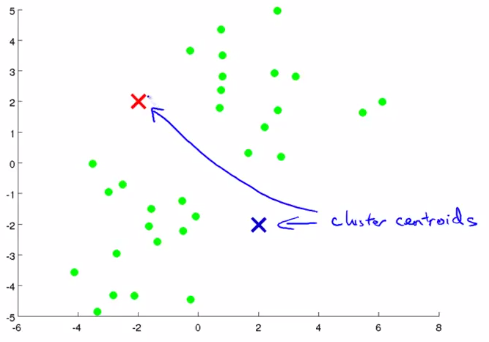
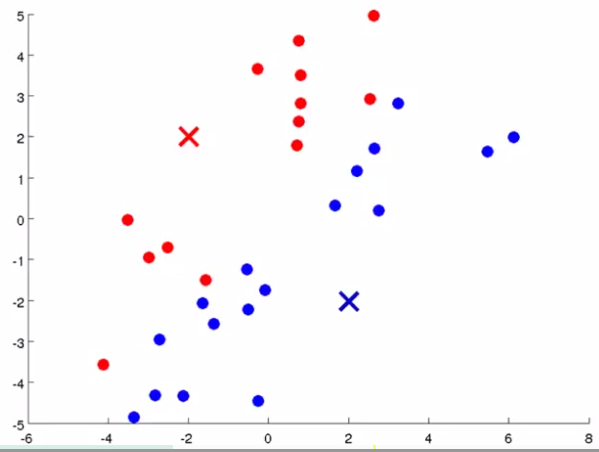
* In contrast, in an unsupervised learning problem, we're given data that does NOT have any labels assoc(i)ated w/ it. So, we're given data that looks like this.



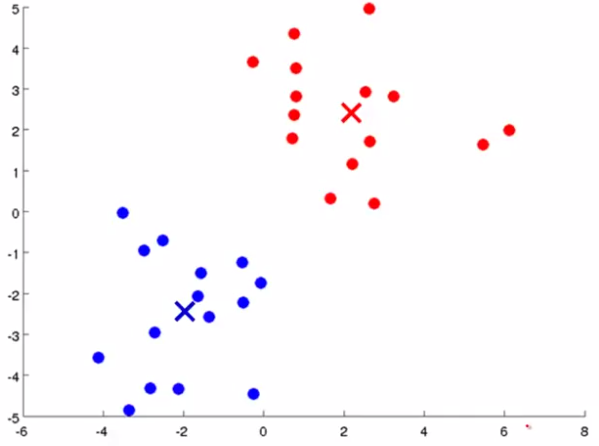
* See a set of points x w/ no labels y (no colors or shapes)
* In unsupervised learning, we give this unlabeled training set to an algorithm + ask it to find some structure in the data for us.
* 1 type of structure we might have an algorithm find for this data set a grouping into 2 separate clusters + so an algorithm that finds clusters like the ones I've just c(i)rcled 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.
* **Soc(i)al 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 soc(i)al 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 effic(i)ently
* **To understand galaxy formation** + using that to understand astronomical data.

**II. K-MEANS ALGORITHM**

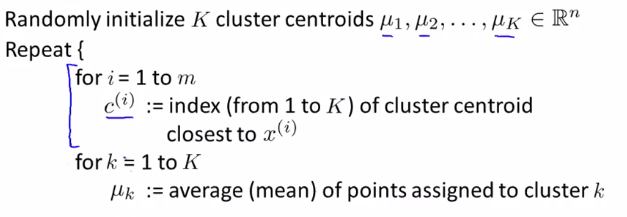
* **The K Means algorithm** is by far the most widely used clustering algorithm + is best illustrated in pictures
* Let's say I want to take an unlabeled data set + group the data into 2 clusters.
* 1st first step is to randomly initialize 2 points, the cluster **centroids,** b/c we want to group my data into 2 clusters.
* K Means is an **iterative algorithm** + it has 2 steps🡪 **Cluster assignment** + move centroid.
* In cluster assignment, the algorithm goes through each of the examples + depending on whether it's closer to the 1 cluster centroid or the other, it assigns the data point to 1 of the 2 centroids

* In move centroid step, we take the 2 centroids + move them to *the average of the points* w/in the cluster (move centroids to mean of the location of all red points + mean location of all blue points)
* Then we go back + do *another cluster assignment step* + then do *another* move centroid step.
* If you keep running additional iterations of K-means, the cluster centroids will eventually not change any further
* At this point, *K means has* ***converged*** + it's done a pretty good job finding the 2 clusters in this data



* Let's write out the K means algorithm more formally.

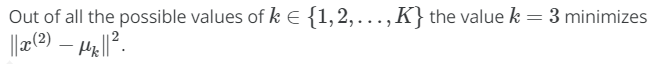


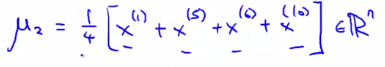
* It takes 2 inputs: parameter **K**, the number of clusters you want to find in the data, + an unlabeled training set of just X’s 🡪 {x1, …. x(m)}
* X(i) is an R(n) dimensional vector = training examples are n-dimensional rather n+1 dimensional



* The 1st step in K means is to randomly initialize k cluster centroids which we call μ1, μ2, up to μ(k)
* Then, the “inner loop” 🡪 For each training examples, set a variable c(i) to be the index (1-K) of the cluster centroid closest to x(i) 🡺 cluster assignment
* c(i) is going to be a number from 1 to K that tells us which cluster a point is closes to
* Another way of writing this: Compute c(i) by taking the ith example x(i) + measure it's distance to each of cluster centroid = *norm between x and μ* = **||x(i) – μ(k)||**



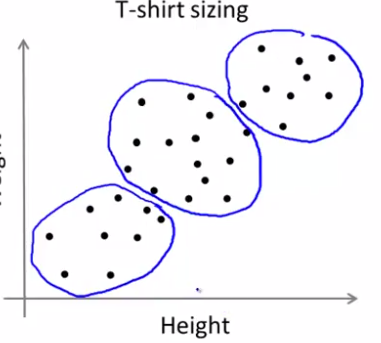
* capital K = the total number centroids lowercase k = index of different centroids.
* C(i) is going to find the value of k that minimizes this distance between X(i) + the cluster centroid
* By convention people tend to write this as the squared distance, so we can think of c(i) as picking the cluster centroid w/ the smallest squared distance to a training example x(i).
* But, of course, minimizing squared distance + minimizing distance should give the same value
* 
* 
* 
* 
* The other “inner loop” of K means does the move centroid step.
* For each cluster centroid (k = {1, …., K}), it sets μ(k) to the average of the points assigned to that cluster k
* As a concrete example, let's say cluster centroid 2 has training examples (x1, x5, x6, x10) assigned to it
* This also means c(1) = 2, c(5) = 2, c(6) = 2, c(10) = 2,
* The average of these points in cluster 2, μ2, is an n-dimensional vector (b/c each example was an n-dimensional vector



* This has the effect of moving μ2 to the average of theose
* But what if there’s a centroid w/ 0 points assigned to it?
* In that case, the more common thing to do is to just eliminate that centroid + end up w/ K-1 clusters
* Sometimes if you *really need* k clusters, the other thing you can do is randomly reinitialize that centroid
* But it's more common to just eliminate a cluster if, somewhere during K-means, it has no points assigned to its centroid (that can happen, but not that often)
* 1 other common application of K Means is to the problems w/ **non-separated clusters**.
* So far we've been picturing K Means + applying it to data sets w/ pretty well-separated clusters
* It turns out that very often K Means is also applied to data sets where there may not be several very well-separated clusters.
* Example application: Let's say you are a t-shirt manufacturer + you've gone to the population you want to sell shirts to + collected several examples of height + weight of these people



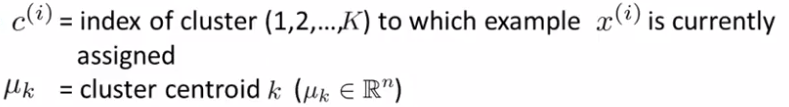
* Now you want to size your t shirts + design shirts of 3sizes, small, medium + large.
* So how big should I make small? How big should I make medium? How big should I make large?
* 1 way to do that would to be to run my K-means clustering on this data set
* Maybe what K Means will do is group these points into 3 clusters



* So, even though beforehand it didn't seem like we had 3 well-separated clusters, K-Means will separate out the data into multiple clusters for you
* What you can do is then look at this 1st population of people + their height + weight + try to design a small t-shirt so that it fits this population of people well
* Then design a medium t-shirt + design a large t-shirt
* This is kind of an example of market segmentation, where you're using K-Means to separate your market into 3 different segments to design a product separately that tries to suit the needs of each of your 3 separate sub-populations well.

**III. OPTIMIZATION OBJECTIVE**

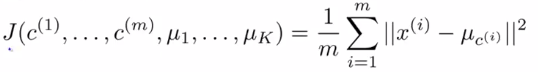
* Most of the supervised learning algorithms we've seen have an **optimization objective** or some **cost function** the algorithm was trying to *minimize*.
* It turns out K-means also has an optimization objective/cost function it's trying to minimize.
* Knowing what this is will be useful for 2 purposes:
* Knowing what the optimization objective of K-means is help one to debug the learning algorithm to make sure K-means is running correctly.
* We can use it to help K-means find better clusters + avoid the local optima.
* Reminder: While K-means is running, we keep track of 2 sets of variables:

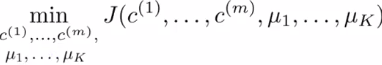


* First is the c(i)'s and that keeps track of the index or the number of the cluster, to which an example x(i) is currently assigned. And then the other set of variables we use is mu subscript k, which is the location of cluster centroid k. Again, for k-means we use capital K to denote the total number of clusters. And here lower case k is going to be an index into the cluster centroids and so, lower case k is going to be a number between one and capital K.
* 1:29
* We use **μ(c(i))** to denote the centroid of the cluster to which example x(i) has been assigned
* Let’s say that x(i) has been assigned to cluster 5, therefore c(i), the *index* of x(i), is equal to 5.
* So μ(c(i)) is = μ5 b/c c(i) = 5

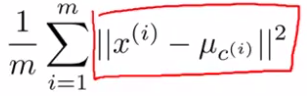


* The optimization objective of the k-means clustering algorithm:

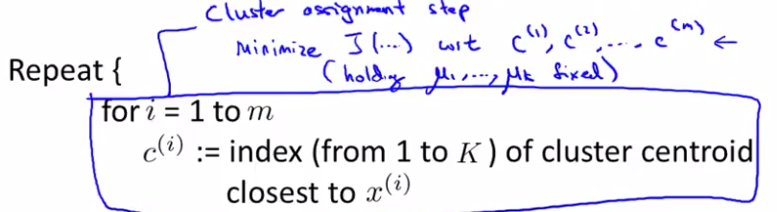


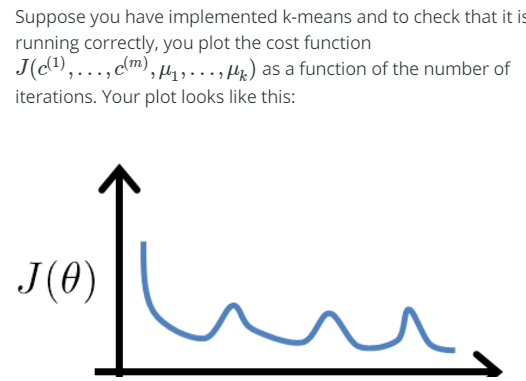


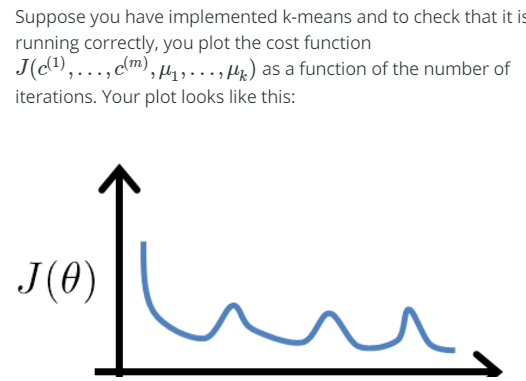
* The cost function K-means is minimizing is a function, J, of all the parameters [c1-c(m) and μ1 through μ(K)] that K-means is varying as the algorithm runs.
* The actual optimization objective is to the right 🡪 1/m \* sum of the squared distances between each example x(i) and the location of the cluster centroid to which x(i) has been assigned.



* What K-means is doing is trying to define parameters c(i) and μ(i) to try to minimize this cost function J, sometimes called the **distortion** **cost function**, or **distortion of the K-means algorithm**
* It's possible to show mathematically that the cluster assignment step is doing is minimizing J w/ respect to the variables c1, c2 and so on, up to c(m) *while holding the centroids μ1 up to μ(K) fixed*.
* So, the cluster assignment step doesn't change the centroids, but is picking the values of c1-c(m) that minimizes the cost/distortion function J.
* Intuitive meaning: “Assign each point to a centroid closest to it, b/c that's what minimizes the squared distance between the points + the centroid”



* It can also be shown mathematically that the move centroid step chooses the values of μ that minimizes J (w/respect to the locations of the centroids μ1-μ(K).
* So, K-means is this taking the 2 sets of variables + partitioning them into 2 halves, the set of c variables + the set of μ variables.
* It first minimizes J w/ respect to c + *then* minimizes J w/ respect to μ
* *Then it keeps on.*
* We can also use this optimization objective/cost function to try to debug other any algorithm + make sure our implementation of K-means is converging + running correctly
* 



* 

**IV. RANDOM INITIALIZATION**

* In this video, I'd like to talk about how to initialize
* 0:04
* K-means and more importantly, this will lead into a discussion of how to make K-means avoid local optima as well. Here's the K-means clustering algorithm that we talked about earlier.
* 0:15
* One step that we never really talked much about was this step of how you randomly initialize the cluster centroids. There are few different ways that one can imagine using to randomly initialize the cluster centroids. But, it turns out that there is one method that is much more recommended than most of the other options one might think about. So, let me tell you about that option since it's what often seems to work best.
* 0:39
* Here's how I usually initialize my cluster centroids.
* 0:43
* When running K-means, you should have the number of cluster centroids, K, set to be less than the number of training examples M. It would be really weird to run K-means with a number of cluster centroids that's, you know, equal or greater than the number of examples you have, right?
* 0:58
* So the way I usually initialize K-means is, I would randomly pick k training examples. So, and, what I do is then set Mu1 of MuK equal to these k examples.
* 1:10
* Let me show you a concrete example.
* 1:12
* Lets say that k is equal to 2 and so on this example on the right let's say I want to find two clusters.
* 1:21
* So, what I'm going to do in order to initialize my cluster centroids is, I'm going to randomly pick a couple examples. And let's say, I pick this one and I pick that one. And the way I'm going to initialize my cluster centroids is, I'm just going to initialize
* 1:36
* my cluster centroids to be right on top of those examples. So that's my first cluster centroid and that's my second cluster centroid, and that's one random initialization of K-means.
* 1:48
* The one I drew looks like a particularly good one. And sometimes I might get less lucky and maybe I'll end up picking that as my first random initial example, and that as my second one. And here I'm picking two examples because k equals 2. Some we have randomly picked two training examples and if I chose those two then I'll end up with, may be this as my first cluster centroid and that as my second initial location of the cluster centroid. So, that's how you can randomly initialize the cluster centroids. And so at initialization, your first cluster centroid Mu1 will be equal to x(i) for some randomly value of i and
* 2:26
* Mu2 will be equal to x(j)
* 2:29
* for some different randomly chosen value of j and so on, if you have more clusters and more cluster centroid.
* 2:35
* And sort of the side common. I should say that in the earlier video where I first illustrated K-means with the animation.
* 2:44
* In that set of slides. Only for the purpose of illustration. I actually used a different method of initialization for my cluster centroids. But the method described on this slide, this is really the recommended way. And the way that you should probably use, when you implement K-means.
* 3:00
* So, as they suggested perhaps by these two illustrations on the right. You might really guess that K-means can end up converging to different solutions depending on exactly how the clusters were initialized, and so, depending on the random initialization.
* 3:16
* K-means can end up at different solutions. And, in particular, K-means can actually end up at local optima.
* 3:23
* If you're given the data sale like this. Well, it looks like, you know, there are three clusters, and so, if you run K-means and if it ends up at a good local optima this might be really the global optima, you might end up with that cluster ring. But if you had a particularly unlucky, random initialization, K-means can also get stuck at different local optima. So, in this example on the left it looks like this blue cluster has captured a lot of points of the left and then the they were on the green clusters each is captioned on the relatively small number of points. And so, this corresponds to a bad local optima because it has basically taken these two clusters and used them into 1 and furthermore, has split the second cluster into two separate sub-clusters like so, and it has also taken the second cluster and split it into two separate sub-clusters like so, and so, both of these examples on the lower right correspond to different local optima of K-means and in fact, in this example here, the cluster, the red cluster has captured only a single optima example. And the term local optima, by the way, refers to local optima of this distortion function J, and what these solutions on the lower left, what these local optima correspond to is really solutions where K-means has gotten stuck to the local optima and it's not doing a very good job minimizing this distortion function J. So, if you're worried about K-means getting stuck in local optima, if you want to increase the odds of K-means finding the best possible clustering, like that shown on top here, what we can do, is try multiple, random initializations. So, instead of just initializing K-means once and hopping that that works, what we can do is, initialize K-means lots of times and run K-means lots of times, and use that to try to make sure we get as good a solution, as good a local or global optima as possible.
* 5:19
* Concretely, here's how you could go about doing that. Let's say, I decide to run K-meanss a hundred times so I'll execute this loop a hundred times and it's fairly typical a number of times when came to will be something from 50 up to may be 1000.
* 5:35
* So, let's say you decide to say K-means one hundred times.
* 5:38
* So what that means is that we would randomnly initialize K-means. And for each of these one hundred random intializations we would run K-means and that would give us a set of clusteringings, and a set of cluster centroids, and then we would then compute the distortion J, that is compute this cause function on
* 5:56
* the set of cluster assignments and cluster centroids that we got.
* 6:01
* Finally, having done this whole procedure a hundred times. You will have a hundred different ways of clustering the data and then finally what you do is all of these hundred ways you have found of clustering the data, just pick one, that gives us the lowest cost. That gives us the lowest distortion. And it turns out that if you are running K-means with a fairly small number of clusters , so you know if the number of clusters is anywhere from two up to maybe 10 - then doing multiple random initializations can often, can sometimes make sure that you find a better local optima. Make sure you find the better clustering data. But if K is very large, so, if K is much greater than 10, certainly if K were, you know, if you were trying to find hundreds of clusters, then,
* 6:45
* having multiple random initializations is less likely to make a huge difference and there is a much higher chance that your first random initialization will give you a pretty decent solution already
* 6:56
* and doing, doing multiple random initializations will probably give you a slightly better solution but, but maybe not that much. But it's really in the regime of where you have a relatively small number of clusters, especially if you have, maybe 2 or 3 or 4 clusters that random initialization could make a huge difference in terms of making sure you do a good job minimizing the distortion function and giving you a good clustering.
* 7:21
* So, that's K-means with random initialization.
* 7:24
* If you're trying to learn a clustering with a relatively small number of clusters, 2, 3, 4, 5, maybe, 6, 7, using
* 7:31
* multiple random initializations can sometimes, help you find much better clustering of the data. But, even if you are learning a large number of clusters, the initialization, the random initialization method that I describe here. That should give K-means a reasonable starting point to start from for finding a good set of clusters.

**V. CHOOSING THE NUMBER OF CLUSTERS**

* In this video I'd like to talk about one last detail of K-means clustering which is how to choose the number of clusters, or how to choose the value of the parameter
* 0:10
* capsule K. To be honest, there actually isn't a great way of answering this or doing this automatically and by far the most common way of choosing the number of clusters, is still choosing it manually by looking at visualizations or by looking at the output of the clustering algorithm or something else.
* 0:27
* But I do get asked this question quite a lot of how do you choose the number of clusters, and so I just want to tell you know what are peoples' current thinking on it although, the most common thing is actually to choose the number of clusters by hand.
* 0:42
* A large part of why it might not always be easy to choose the number of clusters is that it is often generally ambiguous how many clusters there are in the data.
* 0:52
* Looking at this data set some of you may see four clusters and that would suggest using K equals 4. Or some of you may see two clusters and that will suggest K equals 2 and now this may see three clusters.
* 1:08
* And so, looking at the data set like this, the true number of clusters, it actually seems genuinely ambiguous to me, and I don't think there is one right answer. And this is part of our supervised learning. We are aren't given labels, and so there isn't always a clear cut answer. And this is one of the things that makes it more difficult to say, have an automatic algorithm for choosing how many clusters to have.
* 1:32
* When people talk about ways of choosing the number of clusters, one method that people sometimes talk about is something called the Elbow Method. Let me just tell you a little bit about that, and then mention some of its advantages but also shortcomings. So the Elbow Method, what we're going to do is vary K, which is the total number of clusters. So, we're going to run K-means with one cluster, that means really, everything gets grouped into a single cluster and compute the cost function or compute the distortion J and plot that here. And then we're going to run K means with two clusters, maybe with multiple random initial agents, maybe not. But then, you know, with two clusters we should get, hopefully, a smaller distortion,
* 2:10
* and so plot that there. And then run K-means with three clusters, hopefully, you get even smaller distortion and plot that there. I'm gonna run K-means with four, five and so on. And so we end up with a curve showing how the distortion, you know, goes down as we increase the number of clusters. And so we get a curve that maybe looks like this.
* 2:31
* And if you look at this curve, what the Elbow Method does it says "Well, let's look at this plot. Looks like there's a clear elbow there". Right, this is, would be by analogy to the human arm where, you know, if you imagine that you reach out your arm, then, this is your shoulder joint, this is your elbow joint and I guess, your hand is at the end over here. And so this is the Elbow Method. Then you find this sort of pattern where the distortion goes down rapidly from 1 to 2, and 2 to 3, and then you reach an elbow at 3, and then the distortion goes down very slowly after that. And then it looks like, you know what, maybe using three clusters is the right number of clusters, because that's the elbow of this curve, right? That it goes down, distortion goes down rapidly until K equals 3, really goes down very slowly after that. So let's pick K equals 3.
* 3:23
* If you apply the Elbow Method, and if you get a plot that actually looks like this, then, that's pretty good, and this would be a reasonable way of choosing the number of clusters.
* 3:33
* It turns out the Elbow Method isn't used that often, and one reason is that, if you actually use this on a clustering problem, it turns out that fairly often, you know, you end up with a curve that looks much more ambiguous, maybe something like this. And if you look at this, I don't know, maybe there's no clear elbow, but it looks like distortion continuously goes down, maybe 3 is a good number, maybe 4 is a good number, maybe 5 is also not bad. And so, if you actually do this in a practice, you know, if your plot looks like the one on the left and that's great. It gives you a clear answer, but just as often, you end up with a plot that looks like the one on the right and is not clear where the ready location of the elbow is. It makes it harder to choose a number of clusters using this method. So maybe the quick summary of the Elbow Method is that is worth the shot but I wouldn't necessarily,
* 4:23
* you know, have a very high expectation of it working for any particular problem.
* 4:29
* Finally, here's one other way of how, thinking about how you choose the value of K, very often people are running K-means in order you get clusters for some later purpose, or for some sort of downstream purpose. Maybe you want to use K-means in order to do market segmentation, like in the T-shirt sizing example that we talked about. Maybe you want K-means to organize a computer cluster better, or maybe a learning cluster for some different purpose, and so, if that later, downstream purpose, such as market segmentation. If that gives you an evaluation metric, then often, a better way to determine the number of clusters, is to see how well different numbers of clusters serve that later downstream purpose.
* 5:11
* Let me step through a specific example.
* 5:14
* Let me go through the T-shirt size example again, and I'm trying to decide, do I want three T-shirt sizes? So, I choose K equals 3, then I might have small, medium and large T-shirts. Or maybe, I want to choose K equals 5, and then I might have, you know, extra small, small, medium, large and extra large T-shirt sizes. So, you can have like 3 T-shirt sizes or four or five T-shirt sizes. We could also have four T-shirt sizes, but I'm just showing three and five here, just to simplify this slide for now.
* 5:46
* So, if I run K-means with K equals 3, maybe I end up with, that's my small
* 5:53
* and that's my medium and that's my large.
* 5:58
* Whereas, if I run K-means with 5 clusters, maybe I end up with, those are my extra small T-shirts, these are my small, these are my medium, these are my large and these are my extra large.
* 6:19
* And the nice thing about this example is that, this then maybe gives us another way to choose whether we want 3 or 4 or 5 clusters,
* 6:28
* and in particular, what you can do is, you know, think about this from the perspective of the T-shirt business and ask: "Well if I have five segments, then how well will my T-shirts fit my customers and so, how many T-shirts can I sell? How happy will my customers be?" What really makes sense, from the perspective of the T-shirt business, in terms of whether, I want to have Goer T-shirt sizes so that my T-shirts fit my customers better. Or do I want to have fewer T-shirt sizes so that I make fewer sizes of T-shirts. And I can sell them to the customers more cheaply. And so, the t-shirt selling business, that might give you a way to decide, between three clusters versus five clusters.
* 7:10
* So, that gives you an example of how a later downstream purpose like the problem of deciding what T-shirts to manufacture, how that can give you an evaluation metric for choosing the number of clusters. For those of you that are doing the program exercises, if you look at this week's program exercise associative K-means, that's an example there of using K-means for image compression. And so if you were trying to choose how many clusters to use for that problem, you could also, again use the evaluation metric of image compression to choose the number of clusters, K? So, how good do you want the image to look versus, how much do you want to compress the file size of the image, and, you know, if you do the programming exercise, what I've just said will make more sense at that time.
* 7:53
* So, just summarize, for the most part, the number of customers K is still chosen by hand by human input or human insight. One way to try to do so is to use the Elbow Method, but I wouldn't always expect that to work well, but I think the better way to think about how to choose the number of clusters is to ask, for what purpose are you running K-means?
* 8:15
* And then to think, what is the number of clusters K that serves that, you know, whatever later purpose that you actually run the K-means for.