in this lecture i'm going to discuss unsupervised learning and in particular the k-means technique now unsupervised learning is different than what you may have seen in supervised learning where you try to learn a mapping from training data point x to label y where you're given examples of the xy mapping here and unsupervised learning we're just given access there's nobody gives us any correct labels and so we study the x's and we learn a mapping from x's to y's and the k in k means is the number of different y's that we're instructed to use in this mapping if you're used to thinking about supervised learning this seems quite mysterious how do we know which y's are supposed to be assigned to the x the predictor c's if we've never seen any y's at all in our training but when you take a somewhat different vantage point it starts to make sense because here might be the plot of your different x vectors now throughout this lecture i'll i'll mostly think about x as a vector but remember that we still can do future extraction first so what we really might have as the first stage of f is feature extraction and that's not learned that's something that you provide in designing the problem so really this could be a plot of the feature vectors we're going to be classifying vectors either x is a vector and we don't have any feature extraction or we extract features so here are different feature vectors and if we were instructed to assign two labels here then of course we would quite likely look at this and say well it would seem that this is one of the labels and this is the other one now we don't know which label belongs on which but but we do have some idea that these are similar and so whatever label we assign to one of these should probably be assigned to the others and these are what we call clusters so the problem of unsupervised learning can often be thought of as finding clusters in the data so here the elements that we end up giving the same label say y is one all throughout here are called a cluster and so sometimes unsupervised learning is called clustering or at least that's one major form of unsupervised learning unsupervised learning techniques have gotten some amazing results because note that without the need for supervision that is labeling from humans we can get vast amounts of data very very cheaply so we can take all the text that's available out there on on the internet and feed it in as sample text to the learning and uh of course we'll do some feature extraction on the text and then ask for clusters and here's an example from the langan static course of a result that was achieved by simply clustering no labeling on the data so we ask the

clustering of words and we provide a hundred million words from news articles and after feature extraction the vectors that are clustered together in cluster one are the word represent the words friday monday thursday wednesday tuesday saturday sunday weekends sundays saturdays clearly without any supervision the technique has discovered patterns in the data that cluster the days together the cluster of the months together substances adjectives about being large or actually scant it's a little interesting there anyways you see interesting patterns have been discovered man's names relatives so this this technique without any human labeling has in some sense understood a lot about the meaning of words it's a very striking result and a similar kind of striking result comes in analyzing images so on the left we see images of digits and without being told what that there there are digits there or what exactly to look for just look for grouping them in similar patterns we get mostly the digits groups it's the this the technique has discovered on its own that there are these different groupings with some errors you see that nines and fours are easily confused by this technique in particular and there's a five in the threes and a four yeah four and nine so it does very very well and likewise just taking a large collection of images we've discovered the planes this looks like dogs and birds that look like dogs it's interesting to start to ponder what's going on in in the groupings but this is without any human labeling that has to be remembered it's a pretty amazing result okay so we're not going to study the full glory of all the research that's been done to produce these results and in particular this is done with deep learning but we are going to learn a basic clustering technique in this lecture called k-means to understand k-means we're going to realize that it's a instantiation of a very general optimization idea so we're going to talk about optimization for a minute and so in this form of optimization very general view of optimization we're going to imagine we have some choices we need to make and choices of v is going to be the choices of values for this set of variables v so v is a bunch of variables and each one has its own domain so it doesn't have to be bits they could be each one could be a two-way choice maybe one of them is a choice of a real number but we have to choose values for all of these variables okay and then we're going to write choices of v as all the ways that we could set all the variables so a choice would be one way of setting all the variables each one to a value in its domain right so little c here is an element of choices of v it's a particular way of setting all the variables

so then in that very general view that we need to make all these choices we're going to say that the best way to make them is the one that achieves the lowest in some loss function now for any given problem we'll have to specify what that loss function is we want to lose as little as possible so we're optimizing in the min direction this function loss and c here is a way of choosing all the variables and this argument means consider all of the different ways of setting the variables c in all choices and of all the different ways we want to choose the one that minimizes the loss and that's the optimum and if we were to replace this argument with the expression min we would get the minimum value that could be obtained which would be the loss of c opt okay so that's that's the optimization framework the approach you want to explore is what if we can divide these variables into two groups such that we can optimize each group as long as we know the values of the other group so here's our idea is divide these variables into two disjoint groups so v1 and v2 are groups of variables when we take them together that's all the variables but they don't overlap so every variable is in either v1 or v2 and then we're going to write this use this semicolon to say if we made all the choices for v1 and all the choices for v2 we can combine them together with semicolon to get a way of making all the choices from v that's just a bit of a notation because now that we have that now that we have written have broken the variables into two groups and a notation for combining the choices we can talk about the best possible value that we could obtain by considering all the choices by by considering the choices in c1 choices in v1 and the choices in v2 in sequence we can consider all the choices in v2 and get the best possible one for any given c1 and then we can try each z1 do that for each c1 so for each c1 find the best possible c2 what does it score and the min of all that will in fact give us c the same value that c can obtain so if that's not clear it's worth you taking a moment to ponder what this expression is saying that we can divide these in into two groups and we can minimize one group inside minimizing the other group and we will obtain that the full minimum one way to think about it is whatever the minimizing choice he opt is out of all these choices and there should really be loss written right here whatever that min is part of it is minimizing c1 and part of it isn't minimizing c2 and when we plug those into these we will get that same min because c1 semicolon c2 will be that c opt all right so why did we divide them into two groups it's because sometimes in some specific settings including the k-mean

setting once we have these two groups we're able to optimize either of these mins as long as the other one is held constant so for a particular way of choosing the variables in v1 if we fix that we can we may have an efficient way to solve this problem and if we have solved this problem and found a particular c2 we may have an efficient way of solving this problem for that particular c2 taking out this min so basically if we fix c1 and take out the min we'll be able to solve the c2 problem and if we fix c2 and take out this min we'll be able to solve the c1 choice and if we have that situation we can use an iterative optimization technique so we're going to iteratively iteratively improve at first random choices of of the choices from variables 1 and the choices from variables 2. each time we make an improvement we'll improve the choice of c1 while holding c2 fixed or we'll improve the choice of c2 while holding c1 fixed so we'll start out with random choices for c1 and c2 so this is random choices of all the variables in v1 and this is random choices of all the variables in v2 and we'll just repeat this loop well maybe it goes around this way until our loss isn't getting any better and what does the loop say update c1 considering all the possible choices for c1 but notice that c2 is fixed here we're just living with that c2 we're not considering changing it just update c1 the best possible c1 and notice that if that changes c1 it could only make the loss c1 c2 get better because we chose it to minimize this if the origin if the previous c1 was better it would have come out of the min instead of some update the update that comes out has to be an improvement or no change and likewise we'll do an update of c2 holding c1 fixed c2 ranging over the choices from v2 so each of these can only improve the loss and if the loss doesn't improve we've arrived at a local optimum of c1 and c2 what i mean by local optimum is we'll have arrived at a choice of c1 and c2 where we cannot improve on the loss by changing just one side if we could one of these would do an update so in order to get out of that local optimal we might have to make simultaneous changes in the variables in v1 and then the variables in v2 but this method won't do that so it does not arrive at a global optimum necessarily but it will converge to a local optimum if one of these sets is finite it may converge anyways but for sure if one of these sets is finite we will eventually have explored at worst case all of that set right so let's just sketch that briefly if if the choices of v1 set is finite then it's at at some point we may have set c1 to each and every different value from from that set of choices in worst case we'll never set it to the same value again if

we leave that value because let's say we pick a particular c1 we're going to choose the best c2 for that one and if we keep iterating it's because we found a better loss than that which means that that c1 is no longer what we're using we already got right here the best c loss we could get with that particular c1 we're never going to revisit it every time we go around if c1 is changing it's changing to better and better and better loss lower and lower loss it can never revisit so if there's only finitely many choices it will converge so that convergence is essentially trivial if one of the choice sets is finite which tends what may often be the case if one of them is discrete okay so this is the paradigm we're going to use to do our clustering in addition to being useful for our clustering algorithm this same sort of iterative optimization one part of the choices while holding the other constant and then vice versa is used in hidden markov models and learning methods that we will encounter later in the course the unsupervised learning problem that we face in k means is we're given real vectors that we want to classify we'll think of them v1 to vm however typically they come as feature vectors for x1 to xn so just imagine that's already happened that's folded in we're not talking about feature extraction we've already done it we have vectors we want to classify except that we're not given classes so it's unsupervised and we are told k we told the number of groups that were defined so we want to find k different clusters so a cluster one of the k clusters is represented by a vector itself mu1 is an example also d dimensional same same space as these vectors that'll be the center of the cluster so we want to find k centers to say this is the k places where we think that the clusters are and an assignment that says which of the n vectors goes to which of the clusters so z is a function that when you apply it to a number between 1 and n you get a number between 1 and k that says which cluster v i is in so we assign v i to m sub z i z of i okay so once we've decided which points go in which clusters and where the centers of those clusters are we can see how how much the clusters are condensed around their centers so for each vector vi we can say how far is it from its center of mu sub z i and however far that is squared that'll be sort of the loss we experienced for that cluster because the vector wasn't really right at the center and we'll sum up that loss over all the vi's and we'll say that's how bad our choices were we want to find choices to minimize that so we find centers that are in the middle of a giant dense cluster of points really close to that center and we find k such centers and we cover everything if there's

any outliers we're going to pay a price for those outliers so the question is can we find the choice choices to minimize this loss now what are the choices we're making we don't get to choose the vi's that's the data came from feature extraction on our x's what do we get to choose we get to choose the muse we know how many of them there are that's provided we're going to choose the muse and we have to choose the mapping z which points go with which mu so just to get a visual look at this problem maybe these are the vi's and we are told k is 2 so we might choose to put a blue center and a red center here and here we might i mean i don't know how the algorithm has figured that out we haven't talked about that yet but somehow it's placed a blue center and a red center but we also have that's the muse we also have to choose z which is visually going to be represented by saying well which points are going to the red center and which are going to the blue center this is not the only choice this is the sensible choice but what we want to put together is an algorithm that picks both where the blue and red center go and which points are to be colored blue and which ones are to be colored red and this would be a sample conclusion that we might wish the algorithm would come up with so i want you to think about is can we divide the choices we need to make the choice of the muse and the choice of the z into two groups such that each group is easy to optimize if the other one isn't changing and here the obvious two groups are the muse we choose the muse that's one group of choices and we choose the z mapping that's the other group of choices so here there's k d dimensional vectors to choose and here there's n k way choices those are the two groups of choices so we think about if we hold one constant can we choose the other one and we could consider either one of those problems but let's imagine first that we've chosen which points are red and which points are blue can we choose where the red centroid should be and where the blue centroid should be well if we had these choices down here but we don't know where the centroids are anymore we're going to let that float we just know the red circles on the blue circles we clearly would like to put the red centroid at the mean of these locations that's what's going to minimize squared loss we've already seen that minimizing squared loss gives you the mean so we take the average of these vectors and we get some point in the middle that's where this android should be if the red circles are correctly colored and likewise the blue circle should be the blue dot should be the mean of the blue circles so even if we had not

placed these centroids if we had this z mapping that's drawn down here we would get centroids very similar to this out of that optimization and that's this step here set the muse to whatever mu gives you the lowest loss for the given z see z is not varying here and so we do that by setting each centroid to the mean of all the data assigned to that centroid and this is the math for that mean expression and that's going to minimize squared error now i want the converse the other iterative uh step where we've set the centroid so we don't know z that's what's pictured here we set the centroids but which point should go into which centroid and that's very easy we're just going to take say for this point we're going to measure the distance to red and the distance to blue now it's closer to red so let's put it with the red centroid and so that's setting z to be the z that gives the lowest loss by choosing z of i for the i v you know for v i is the centroid that's closest to vi so we measure mu j to vi for each j and take the j that minimizes that and that's this that's the centroid so here j is ranging from red and blue mu red and mu blue we take whichever one's closer and that's what z of i will be red or blue and that determines the coloring of the circles around these points and so we can note that we can repeat that you may feel that once we've chosen once we've done one step and then the other step we're done but no because each step here may change this may change z which may make a different mu new centroids new centroids may change which is closest so we get a new z then we get a new mu and we get a new z that can keep happening but at every step if z changes from what it was from what it was here to a new z here it's because the loss went down so each of these steps reduces the loss when it if we get one that doesn't reduce the loss there's no reason to change say it's tied don't change and if you don't change then these steps keep not changing and we're done or we could get so the loss isn't reducing very much as we change and we can just stop because loss is not going down very much this process is guaranteed to converge as i've said because the choices that we have for z are finite we have real vectors to choose from or floating point vectors for the centroids but there's only finitely many choices for the z mapping so it will converge um it doesn't necessarily converge fast although in practice it's pretty effective especially if we are smart about how we initialize how we make our initial choices so that's our last bit of discussion we noticed that our k-means algorithm just described it will converge to a local optimum um and if we don't like that optimal or suspicious of it or even if

we're not just to check we can restart and use a new initial set of centroids and go from there iterate from there we may get a different optimum and we get may get a better optimum and it's typical to run this many times and see what's the best optimum we get but there's also been developed a smart way to choose these initial centroids it's still random but it's a random process that's smarter and in that random process we're essentially going to take a random one of our vis is one of our centroids randomly selected but then we want to choose another vi another vj one of the data points um that's far away from that vi and then we want to choose one that's far away from both of those as long as we can keep doing that up to k of them so um that method that's the sketch of the motivation for the method but the actual method is a probabilistic method so it's not just going to choose the furthest away so this is called k-means plus plus and it involves k-means but with an initial selection of centroids done randomly according to a certain method so we'll select particular vi data points that were that we're clustering as centers but each selection is biased to be far away from the current selection and the bias is this following the probability of selecting these really could be v's the probability of selecting vi is proportional to the square of how far away it is from the nearest already selected center so we've got already selected some centers and we're looking at the x eyes that are far away and we're going to try to choose one that's far away by squaring the distance away to the nearest center and letting that be the used to define the probability of distribution so this takes a little bit of time but it pays off the iteration to uh convergence happens faster so in the in the end this saves time and it also provides you some guarantee on how bad the local optimum is so people have developed the theory of this you can be arbitrarily far from optimal loss if you don't initialize smart but if you initialize this way there's a guarantee on the ratio of your loss to the optimal loss it's not a great guarantee but it's a guarantee anyways so this is what we're doing you still there's a question that should be popping into your head is where did k come from how do you know how many clusters there are and typically you don't ahead of time know how many clusters your giant data set is going to fall into and that is a hyperparameter of this technique if we set k equal to n then we can just have zero loss we'll have a centroid for each data point done you should smell overfitting here that's overfitting we don't want k so large we can precisely capture the training data because that's probably not what the test data

will look like and in fact that should suggest to you what you can do okay is a hyper parameter of this technique and we get over fitting if it's too large so what can we do well we can use a validation set we can try different values of k and when the validation error starts going up we want to go back and use the value of k for the previous validation error so that's an example of how we could select k using some kind of search and using the validation set to rein in overfitting so with this lecture i will link a website where you can play with k-means we will visually generate some data for you and you can choose different ways of generating that data and choose different k's and watch it iterate you can click and watch the different phases of iteration happen visually highly recommend looking through that demonstration