Week 19
Video Transcripts

Video 1 (5:23): Clustering Image Dataset

Clustering algorithms are an example of unsupervised learning. The difference between supervised learning and unsupervised learning is very straightforward. In supervised learning, we give our training cases, and we say that these cases have a bunch of independent variables and a dependent variable. And, we know the value of the dependent variable, so we ask it to, sort of, use the independent variables to find a curve or something that fits to the dependent variable. In unsupervised learning, we do something slightly different. We say, "All right! Here's our space of independent variables. Now, try and see how many features that we have, and now try and see what values of these features will make these cases, make subsets of these cases closer together by some estimate", okay! How...how do they fit closer together? So, we have...we have-you can think of having a vast net or, you know, data points sitting all over the place, and you want to find planes that cut through the space that can group these data points together in a similar sort of way. Okay, that's the idea.

They're called clusters, and they use only value of the feature space. Of course, what we do is, we again work our way through a training and testing sample, and in the training sample we give it the independent variables and it groups them. And then, we want to measure how well that's done by looking whether these groups map on to something in the real world that makes sense to us, right! So, for example, you could take physical characteristics of people and say can we differentiate between these physical characteristics and put them into two groups. When we put them into two groups, maybe we can figure out, men versus women, you know, something like that. And then, when we don't say that this...this case corresponds to a man or this set of features correspond to women, we let the...the algorithm use the features to differentiate between the two groups of people. A popular algorithm for doing clustering is k-means clustering. What it does is, it partitions the data space into clusters and it minimizes the distance between the mean of a cluster and the data points. So, every data point is sitting in n-dimensional space where each dimension is a...a feature. And so, you can measure the distance between 1 data point and another data point, and we want to minimize the...the fine clusters with a mean distance between data points and each cluster is...is minimum, really, right! So, we want to minimize that distance. And, what you need to know in advance is, how many clusters you're going to have in your data set, in...in your domain, right!

So, you want to know how many clusters, like if you're doing men versus women, you have, 2 clusters you know that. You don't want to like say, "Hey! Find me the number of-" You can't say find me the number of clusters, in that case. You need to actually tell the k-means algorithm, how many clusters to use. So, to set that up before we get moving on this, you want to make sure you have your imports. So, the imports you want to do is, you're going to use a data set that comes with 'sklearn' called a digit recognition data set. So, I'll explain that in a second but let's just make sure we load it in. So, we have

NumPy, of course, matplotlib of course, we inline it for any graphs that we draw. We load the digits data set. So, that's the data set we're going to use. And we're going to import from 'sklearn' preprocessing. We're going to import a package called 'scale', which we use to rescale in this case. In our example, we're going to rescale data into a normally distributed mean, 0 standard deviation, 1 data set. So, once we've done that we should be in good shape. So, what we'll do is, we run this to load the actual digits, and run equal 'load_digits' and we can see what we get here. Oops, forgot to do that. And, we get this stuff over here. So, what we're getting is, a optical recognition of handwritten digital-digits data set. And, the goal here is that we have a set of data that is handwritten 1s, 0s, 2s, 3s, 4s, 5s, 6, 7, 8, 9s, and we want to use machine learning to be able to...to...able to recognize, given a new sample. Given a new- somebody else writes a one and gives it to our machine, we want to be able to recognize it as a one. A handwritten recognition problem, right, that's the goal here.

So, that's the data we're going to work with and that's the data here, right! So, what it really does is, it takes each digit and it pixilates it. So, you can think of roughly as something like this. You can take a...a digit, put a box around it, and put like, you know, depending on your resolution, put little pixels that point to this stuff. And if, you know, if there's a written part, then the pixel is lit up and, if there's a nonwritten part, it's not lit up. So, you can see which ones are lit up, which ones are not lit up, and that's what this is, sort of, representing over here with these numbers, this data and this thing you can see '1' '6'. So, '0's are you can think of unlit up parts and '1', '6', '10's are the lit-up parts in the-in that particular case. So, each one of these is a case, right, that's a case. Okay, so there are many many '0's and '1's here. So, that's the idea. So, our features are the pixels whether they're lit up or not and the categories we want to do are whether they're '0's, '1's, '2's, etc. And then, the last thing we can do is, we can scale the digits. So, we will run the scaler and what happens then is that we get our digits from 0s and 1s, instead we have scaled them and normally distributed between mean 0 and standard deviation 1. So, that's our data set.

Video 2 (4:19): K-Means Clustering 1_Image Dateset.

So, let's do some fun stuff, let's render the digits and their associated values. So, we're going to use- write a function here, called 'print_digits', and what 'print_digits' is going to do is it's going to take the...the actual cases and show them on the screen. So, we're going to get them like this, okay. I am going to run that and show it to you here. So, these are the digits we are printing. This is the data that we saw earlier, right! And, we're using that data to actually render the digit. And, we are doing that using matplotlib here. So, again what we do is, we set up a figure and we set up a bunch of subplots and we're going to adjust these so that the-because we have 2 sets of subplots. We have the actual images 0, 1, 2, 3, which are the images we are working with. And, we have the corresponding dependent variable value ‘0’, ‘1’, ‘2’, ‘3’, because we know in our data, we know that this is a ‘0’, this is a ‘1’.

In the data, we know that we just want...want- we're not going to use that for the clustering, but we know that, we're going to use that for testing. So, we have that and we want to make sure that the '0's and '1's are aligned with those things. So, the purpose of the 'subplot_adjust' is to just make sure that

they're aligned. If we don't do that, then we're going to get '0', '1', '2', '3', '4', '5', '6', '7', '8', '9', ending somewhere here, you know, ending where the four is, so that's not going to work very well. And then, we render-we work through this thing and add each image to the thing. We have the images here, and we create the stuff and add the text of the dependent variable value, the 4, 5, 6, 7, 8, 9 stuff, the actual values. And so, we have our images, we have our target values, and we know that there are 10, we're only going to show 10. We could show more because you know that, our data set is much larger, right! So, this is what we get and we get our image [inaudible]. This is kind of a neat way of showing this stuff. The next thing you want to do is break stuff into a training and testing sample, because that's what we always do.

So, that's again very straightforward. We take a 'test_size' of '25' percent and break our- use the 'train_test_split' function here and break it up into the-take the data and break it up and take the target and break it up into 2 different sets, and we're going to get our thing here. We can take a look at labels. So, the labels are-so the first case is a 5, the second case is a 2, third case is a 0, and that's what our y values are. And, if we want to look at the x values, they will be- 'x_train' is essentially our normalized data, the pixelated stuff normalized. So, we can look at the unique values of 'y_train' just to be on the safe side, and we find there are 10 of them which is right, '0', '1', '2', '3', '4', '5', '6', '7', '8', '9'. And, that gives us our basic structure here. And, the last thing we do, which we do always is always the same thing, is we import the 'cluster' algorithm-the 'cluster' library module and create a 'k-means' clustering algorithm from that. We tell it to initialize it by doing some initialization stuff. So, what it-'k-means' works better when you initialize it, so we initialize it initially, we run this pre-algorithm, you can think of to do the initialization. And, to start with a-but what k-means does is it starts by randomly allocating the..the digits, right!

So, it'll say, "All right, I need 10 categories and that's the number of clusters ‘10’, and it'll randomly assign the data to ten categories." But, if you can, sort of, intelligently start off when you have some, you know, some knowledge about the data, like maybe the means inside the values and all that kind of stuff, then you're better off. So, you can use this 'k-meansplusplus' to actually start the algorithm at a better point, and so that's the 'k-means'. And then, once we do that we fit it. So, fitting is going to be, of course, actually running the algorithm and we've got this stuff running. So, that's our stuff now. So now, we want to actually work with and figure out how well our clustering has worked, and try to understand the-what we can, whether we're getting a good result or not.

Video 3 (4:59): K-Means Clustering 2_Image Dataset

So now, what we've done is we've fitted our model, right! So, we fitted our model. So, our model now has predictions on, not predictions but really it's assigned cluster numbers to groups of-to each individual data point. So, we've told our model to take ten clusters. So, it's assigned ten cluster numbers, 0, 1 to 9, to each individual data point. And, those numbers are in this-clf is our model, in the-in this attribute called, labels underscore. So, what we're going to do now is we're going to take a look at our training data set, and look at the first 20 cases, and see what cluster numbers were assigned to each

case. And, we look at the actual image itself. So, we're going to eyeball. We're going to see what the image is and see what the cluster number is attached to it. And note that the cluster number does not have to correspond to the image. Because-for example, if it takes everyone and assigns it cluster seven, then that's great...great.

Then, we know cluster seven means a '1', right! Because this doesn't know that we're looking at...that we are looking at '1', '2', '3', '4', '0', '1' to '9'. The machine doesn't know that, right! All it knows is that we are looking to differentiate between different data points. So, let's take a look at this. We call our 'print_digits' function, the function we wrote before on 20 of these things. And, we look at this stuff here and try to figure out, what, you know, whether it's doing this accurately or not. So, we look at the first case is a '5' and it puts cluster ‘1’ there. So, let's see if there are any other '5's. Going further down, we see that further down, way down here, there is another '5' and it's put in cluster ‘8’. And then, if you go further down, there's another '5' in cluster ‘8’. So, we have two '8's for a '5' and one '1' for a '5'. So that's, you know, not great but we're only looking at 20 samples. We look at '2', the second item here, the number, the image is '2' and it's put in cluster ‘3’. So, let's look at if there are any more '2's. Yes, there's another '2' with ‘3’, another '2' with ‘3’.

And, we go further on and the last thing, another '2' with ‘3’. Now, that's great. All the 2s that we are seeing in our first 20, it could be messed up later on are all 3s. So, we know that '3' is likely to be cluster ‘2’. Sorry, cluster ‘3’ is likely to be the image '2' and so on and so forth. Right! So, we can look at, say, '0'. '0's is in cluster ‘2’ and we find '0' cluster ‘2’, and well ‘2’ '0's, they're both in cluster ‘2’. So, that's, you know, just eyeballing here. It's not really solid scientific way of looking. We just want to see what...what we're getting out of this. So, that's our basic structure here. So, we say, "Yes, there is some in our first 20. Anyway, there's some discrimination because '2's are in cluster ‘3’, ‘5’s are generally in cluster ‘8’, '3's are-'7's are in cluster ‘6’, looking at this '6's are in cluster, well I can't say much about '6's." Oh! What we can see from this is what we can see from this- so, what we want to do then is we want to use the test sample, now to generate predictions. So, we've got our...our training sample. We've trained our model to differentiate among these things by looking at data points.

Now, we give it the testing sample and it's going to use the feature values, and the trained clustering that we did before to predict which cluster things belong to. So, we do-we run that that and we get a bunch of predictions and, of course, they are just numbers we won't actually see that. So, what we'll do is we'll look at 15 of these values here. We're calling our 'print_cluster' function again. We look at 15 cluster...cluster values for every y prediction for every cluster. So, our...our y's are predicting ‘0’, ‘1’, ‘2’, ‘3’, ‘4’, ‘5’, ‘6’, ‘7’, ‘8’, ‘9’ as our clusters. So, for each of those clusters, we're going to get the images that are included in that cluster, right! So, if the cluster number is 0, these are the images that are included. We're looking at cluster 0, we see there a bunch of '1's, quite a few, but then there's a '2', '8', '8', '3', '8', '2', '8'. So, that's not so great. And we only have 15 values are printing. If we look at cluster ‘1’, they're all '4's. Now, so that's great. That looks good. Cluster ‘4’ seems to be getting accurately assigned to cluster ‘1’. Cluster ‘2’ is all '0's. Cluster ‘3’ has all '2's.

Possibly, there's some of them that look vaguely like '1's so that's not clear. And there's ‘1’ here that looks like a '3', the second-last one. So, [inaudible] we don't know...we don't know for sure, but pretty much over the board they're '2's. Cluster ‘4’ looks like they're almost all '6's. Cluster ‘5’ looks like they are mostly ‘5’. There is a- some, looks like a '7', not clear! People don't write very clearly, that's the bottom line. Cluster ‘6’ has mostly '7's. So, there is some sort of relationship here. But, our goal is to find and see whether these relationships are solid or not. So, that's what we want to do. We want to evaluate the model itself. So, let's see if you can do that.

Video 4 (7:21): K-Means Clustering 3_Evaluating the Model

For evaluating the model, we use something called an Adjusted rand index. It's a measure of the similarity between two groups, okay! So, the goal here is to-we have a group of actual test data that is the actual scores that we have, sorry, the actual...the actual numbers that we have, so we know that each case in our data set- we know what the number is, right! We know whether it's a 1 or a 2 or a 3, and we also have the predictions. That's what our model is predicting. And, we want to see what's the similarity between these 2 sets. You know, are they reasonably similar? We're not looking at actual values, but seeing whether the differentiation is similar or not. Okay, and you can look at that thing over there. So, if there's '0.0', then there's no similarity...no similarity at all and that's completely random. Any overlap in that- and ‘1.0’ indicates that the two groups are identical.

So, let's run that and we get '0.567' similarity. That's not so bad. It's a, you know, it's a-we've done a very basic model so it's kind of, it's not great...not great-it's not terrible. But, it's better than random because 0 would be random. So, we are pretty much not the way there [inaudible]. And again, since we have categories over here, we've got cluster numbers and all that kind of stuff, we can again draw a confusion matrix of sorts with this. So, this is going to be a little bit different, the confusion matrix. But, let's take a look at it and then, I'll explain it. So, it's a little bit confusing here. So, what this confusion matrix is doing is, it's- we have a cluster number and an actual prediction, right, so what this does is, each row in this corresponds to a number in the test sample. So, each row is-so, this is row ‘0’ in the test sample, okay! And each column is the cluster assigned to that case by the...by the model. So, what this is saying is that for our actual '0's, none of them were put in cluster ‘0’. None of them were put in cluster ‘1’. So, none of them into cluster ‘0’, None of them into cluster ‘1’, 43 were assigned to cluster ‘2’, and none of them into any other clusters. So, if you're looking at our '0's, we've done a great job of identifying them. If we get a cluster ‘2’, we are reasonably confident that it's a '0' or other, if our model is-let me rephrase that, if our model is seeing a '0', then it's going to put out inside cluster ‘2’.

So, that's a great number. Our testing sample, it's not the training sample. Can it...can it tell us confidently the other way? Well, almost! Because, if we find that we have cluster ‘2’, then the only case in cluster ‘2’ that's not identified as ‘0’ is this ‘1’ here which is '0', '1', '2' '3', ‘4 ‘, '5', '6'. In one case, it took a '6' that was actually a '6' and identified it as a, put it in cluster ‘2’ which is a '0', right! So with '0's, we've done a very good job. You know, so, given a '0', we can with, reasonable confidence and...and cluster number ‘2’, we can with extreme confidence actually, say, "Hey! That's a '0'." But, let's look at

something else. Let's look at ‘7’, right! So, that's '0', '1', '2', '3', '4', '5', '6', '7', no, not '7'. Let's look at this ‘1’. So that's-all right, let's look at this ‘1’ here. This '0', '1', '1', '3'...'3'. Let's look at a '3', right! So, the '3'- if this is a column, the row for '3's, so '3's get assigned to cluster ‘0’ in one case. '1', '2', '3'. Cluster ‘3’ in one case.

And so on and so forth. But cluster ‘8’ in 39 cases. So, in 39 cases, we're getting a 3 in cluster ‘8’. And, in 4 plus 1, 5 plus 1, 6 plus 1, 7 cases, we're getting a '3' in some other cluster. All right, so again that means that we have a precision essentially of 39 plus... 39 divided by 39 plus whatever the denominator is, right! 4, 5, 6, 7. However, if you look at the other way around, and we look at this, at the column, that is column number 8 over here, we find that in column 8, in cluster ‘8’, 39 of them are '3's, but ‘1’ is a '2', 16 are '5's, 11's are 8, and 40 are '9's. So, if given that we get a cluster value of ‘8’ for a figure, can we with confidence, say it's a '3'? Not really, right! Because, we have so many cases, where it could be a...a '9' or it could be a '8'. It could be-this thing [inaudible] in fact there are more ‘9’s than there are '3's in that cluster. So, that's...that's done, a very poor job of identifying the cluster correctly, right! So, this is what you want to do with the confusion matrix. You want to see how well it's actually figured out the clusters themselves. So, you look at this stuff and you can then, you know, decide for each digit, how well it's been doing, maybe try to get more data, or more or a higher pixel rate, or higher resolution, before you actually work and say, "Hey, we get a better [inaudible]."

But, we can see our 56 percent rand score is coming from the fact that we are, you know, doing well on some numbers. Like we are doing well on '0'. We are probably doing well on 0, 1, 2, 3, 4. Yup, we're doing well on 4, because 4 we have 0, 1, 2, 3, 4, right, '50' cases that are in cluster ‘1’, here, '50' cases in cluster ‘1’. And, only 2 plus 1, 3 plus 1, 4 cases that are 5 cases actually that are in other clusters. And even more important, cluster one doesn't contain anything except '4's. So, if you get a cluster ‘1’, you can say 'Hey, that's a 4, right!", even though you may not recognize every 4 correctly. So, that's how you sort of analyze the results of your clustering analysis and looking at this confusion matrix. Finally, what we can do is we can look at the graphical view of the clusters, kind of interesting. And, what we'll do here is, we'll run a principle component. We will take our- so the idea here is that a graph is two-dimensional because it fits on this page. Our data is n-dimensional because we have n features, right, as many features as we have.

So, what we can do is we can reduce our dimensions to 2 using the principle component analysis. So, we import this 'decomposition' from 'sklearn' and run 'pca' on that and say we want only 2 components from our training sample. And, once we've done that we can- let me draw this and show you, we get a really nice picture. It tells us-these are our data points reduced to 2 dimensions, and these are the clusters where they come in here. It will be nice when we look at this if you could see the boundaries were clearer, right! So we can see that some of the boundaries are reasonably clear, right! For example, there are very few on the boundary between the orange and the blue, and the orange and the purple, and the purple and the blue. But, if you look at the blue, the light blue and the green, and the light blue and the brown, and the light blue and the dark blue over there, you know and that area is very, very messy.

So, what you would like to see in a good- the result would be that very few points at the boundaries, you know so that the clusters are reasonably accurate, right! So, there are 10 clusters over here. So, that's the goal in clustering algorithm. A very brief introduction and so it should be up more about it.