Course Name: Machine Learning and Deep learning - Fundamentals and Applications

Professor Name: Prof. M. K. Bhuyan

Department Name: Electronics and Electrical Engineering

Institute Name: Indian Institute of Technology, Guwahati

Week-12

Lecture-44

Hello everyone. Welcome to this lecture on Machine Learning and Deep Learning Fundamentals and Applications. In this lecture, I am going to walk you through a few programming examples on machine learning and deep learning. The goal of this lecture is to get you acquainted with the practical implementations of the concepts that has been taught to you in this course. These examples are going to be on four machine learning algorithms and a custom made CNN that is Convolution Neural Network algorithm. Now this is going to be a basic demonstration.

You can definitely work on it and prepare your own complex course and get the results there as your journey progresses in the field of machine learning and deep learning. So let us start the class. So this is going to be the content of today's lecture. First we will be talking about why we are using Python as the environment for programming this machine learning and deep learning algorithms.

Then we will talk about the use of Colab which is basically used for accessing the free GPUs that has been provided to you by Google and we can definitely make use of this free GPUs that has been provided to us and after that I am showing the practical implementation of these four machine learning algorithms. They are linear regression, k-means clustering, KN classification and SVM classification and after that I will show you the custom CNN classification algorithm. So now why Python? So you see here in the middle of this slide I have written a code in C to print the statement hello world. Now you see here in order to print this statement I am writing three lines of codes. The first line is to initialize the header for library and then I am using this main function here and then I am writing the statement within this curly braces.

Now since this is a function I have to return this 0, right. Now see just to print this single statement I am writing four lines of codes. So this is becoming a cumbersome affair. Think what would be the scenario when we would be coding for ML algorithms and deep learning algorithms. But this is not the case for Python.

So when we are printing a statement using Python we just have to write print hello world. So you see here there is no specific requirements needed and most importantly the syntax of Python is similar to the English language and also it allows the coder or developer to write codes in fewer lines and on top of that we have a very strong support system on the net. So if we have any issue or if we encounter any error during coding if we write that issue or that error in the net in certain platforms we will instantly get the reply as a solution. So it makes life easier for our coders as well as the developers. So these are the reasons why we have chosen Python.

Now after this let us see how to use Colab. Now we are using Colab because Google provides us with a few resources. There is some GPUs where we can train our model. Now this makes life easier for us because it reduces the training time of our algorithm and one important thing is that in order to use Colab we need to have a Gmail account in Google. So before starting Colab just sign in to the Google or the Gmail account and after that type Colab in Google.

After that we see this page appearing in front of us. So here we just have to click on this link that is welcome to Colabratory and then this window pops up in front of you. Now in this window we have this button here that is new notebook. So if we click on this, this layout appears in front of us. So this is similar to the Jupyter notebook layout.

So here we see the cell. So we will be coding our algorithms in this location and then we can just change the file name here. Now initially we would not be provided the access to the Google servers. So in order to access the GPUs we first have to click on this runtime button and then in this drop down menu we just need to select change runtime type. After clicking on that this window pops up in front of us.

Now when we select this button here that is the T4 GPU and click on save we see that the T4 name appears on the top right corner of your notebook. Now this assures that you have selected the GPU. Now you just have to connect to the servers by clicking on this button. Before that I would like to draw attention to one of the icons here in this notebook. You see this icon here.

So this icon is basically used to access the content of the Colab. So this is how the content appears. So this space is provided by Colab where you can upload or down or save certain models or specific files to the Google Colab storage place temporarily. Now why this temporary? This because once you log out or log off from the session this content space deletes. So what I mean to say is that all the files that you have uploaded or you have saved in this content it would be deleted.

So if you want certain things to get saved or you want to retain something you either have to copy it to your Google drive or you need to download it. Alright. So this the

things that are stored here are temporary. Now there would be certain files that you would like to access from your Google drive. Mostly the datasets that you have created or you have downloaded.

So those datasets would be residing in Google drive. So in order to access those files you just have to click on this icon. So this icon basically mounts your Google drive. So when you click on that this window appears in your screen asking for your permission to access the Google drive files and after that you click on this button here. Once you do that you see here in the content space this drive folder appears and also the icon which you click to mount the Google drive is crossed out.

So this means that the drive has been accessed or you are able to access the drive now and it has been uploaded to your content space. And one more important thing that I would like to draw your attention. Once your notebook is connected to the Google servers you will see a green tick here with the RAM in this space shown in this portion of your notebook. Alright. So now the Google Colab is almost ready.

Now you just have to write your codes in this cells and if you want more cells you just have to click on this button here. So it will insert a new cell or also you can just hover your mouse pointer at this location and then this buttons would pop up. And when this button pops up you just click on this code button and a new cell would reappear. And the third option is you can just go to this insert button in this bar click on it then a drop down menu appears through which you can just insert a cell. So now you can start coding with Colab.

Now let us move on to the next slide where I am actually implementing a linear regression problem. So what I am doing here is that first I am importing a matplotlib.pyplot library. So this library basically helps us to plot different graphs or curves. So what we have here is a list that contains the ages of 13 cars which has been assigned to the variable x and then we have a second list which contains the respective speeds of these 13 cars when they passed through a toll gate.

And this list is assigned to the variable y. Now what we are doing we are using this 'plt' object that contains a matplotlib.pylot library and using the scatter plot function we are plotting the x and y and this is the diagram that we got. So you see here these are the dots that represents x and y. Now in order to fit the line I am taking help of the scipy library.

From this scipy library I am just importing this stats.linregress function. Now what it does is that it just fits a line to the points that has been assigned to us or that we have at

our disposal. So we are sending the x and the y coordinates of the points and giving it to stats.linregress which fits the line for us.

Now as a result or in return it sends 5 arguments or 5 parameters out of which we are going to use only 2. So these are slope and intercept. The remaining 3 we are just going to ignore that is why we are using underscore in their place. After that we are just defining a function here the name of the function is myfunct and it accepts 1 argument that is x and then it returns the value which is obtained by performing the function slope × x + intercept.

So this is basically mx + c. So this is the equation of the line. Now in the next line of code we are just assigning an empty list to y1 and then we are running a for loop to the entire length of x. After that each value in x is sent to the function myfunct and this value is calculated that is mx + c and it is stored in the variable value and after that we are just appending this value to y1. So at the end we will have a list y1 whose length is going to be 13. Now we are just plotting the original points which is given by xy and its color is kept blue and after that we are plotting the line that we have fitted or obtained through this linregress function.

So it is given by plt.plot (x , y1) and we see the regress line here. So line fitted alright. So we have obtained our result here. So this is the final outcome of the linear regression process. Now let us come to the unsupervised machine learning algorithm that is k-means clustering.

So in this case I am just using another library that is numpy. So it helps us to work with arrays and then we have this matplotlib library which help us to plot and then we have 2 variables which contains 2 list and now we are just scatter plotting this x and y and this is the output here. Subsequently what we are doing we are just zipping x and y so that we can form a coordinate. So that coordinate will contain one value from x and one value from y and we are just displaying it using print data.

So you can see the output here. So this is the output. So you see one value from x and the other value from y. Now we will do the k-means clustering on these points. So for k-means clustering I am taking help of the k-means function that has been already given to us by the library sklearn.cluster. Now what we are doing we are creating an object k-means using this k-means function and we are sending n cluster as argument. So n cluster will determine how many clusters we want as output. So in this case I am using 2 alright. But you can definitely play with this cluster value and see how it performs. Now one thing is that in this data you see we can make out that this is going to be one cluster and the other one is this one.

So it is easier for us by just visually looking at it. But in real world scenarios this is not going to be the case alright. But let us see this let us consider this because our goal is to

just learn how to use this algorithm. We are not looking into the complexities alright.

So okay. So now once this k-means object is created we are just fitting the data here and then we are plotting the xy values. But this time what we are going to do we are going to assign the points that belong to a particular cluster with a single color the points that belong to the other cluster with a different color. So that would be taken care of by c equal to k-means.levels alright. Now you see here the points belonging to cluster 1 are level or colored yellow and the points that belongs to the other cluster are leveled with blue color.

So this is the second cluster. Now I talked about this number right this cluster number. So this cluster number would not be easy to initialize at first. So if you want to get a proper method how to initialize this cluster you can look at this elbow method. So it is in the next slide.

So this is the elbow method. So what it does is that it finds out how many number of clusters are possible that is the minimum and the maximum value. So if we are having 10 points here so maximum value of the cluster is going to be 10 and the minimum is going to be 1 right. So that is why we are running a for loop from 1 to 10 alright and we are just calling in the x-means function here and sending n clusters is equal to 1. So we will be iterating it till it reaches 10 alright and for each object k-means we will be fitting the data and then we are just appending the value of k-means.inertia to inertias. So this variable is initialized here. So inertia was initially a empty list and after appending it or running it for 10 times we would be getting a list containing 10 elements or 10 scalar values alright. Now you might be wondering what this k-means.inertia is all about. This inertia is sum of the square distance of the samples to their closest cluster center alright and after we are done calculating this we are just simply plotting this inertia here in this line and this is the output we got.

So you see here clearly in this plot there is an elbow at this point. So this is the discontinuity because this part is linear and this part also we can consider almost linear right, but the discontinuity is occurring at this position. So we are choosing 2 as the optimal number of clusters alright. So this is a technique to determine how many clusters we should be picking up for the distribution of data that we have at our hand. One important thing that I forgot to mention is the use of this library here.

So we are importing this library that is warnings so that we can ignore the warnings that occurs while executing this code. Now in this case you see few of the warnings will appear, but those warnings are not detrimental to the execution of the program. Now since these are not detrimental so we can just simply ignore them and that is what we are doing in this line alright. So let us move on to the next algorithm here so that is KNN classification. So this is K nearest neighbor classification and for this we are importing 3

libraries one is numpy which help us to work with arrays then we have the matplotlib library which help us to plot and the other one is pandas.

So this library helps us to work with tabular data alright. Now we are taking help of the sklearn library and importing the dataset function and this helps us to load the iris dataset. So now this iris dataset consists of flowers basically the iris flowers so this contains 150 samples and contains 3 levels which means 3 classes alright. Now these 3 classes are basically the species of iris which are iris satosa, iris virginica and iris versicolor and each sample here has 4 features they are sepal length, sepal width, sepal length and peal width and this values are in centimeters alright. So once the entire dataset is loaded into this variable we are just segregating it into data and levels so it is shown in these 2 lines and after this we are just using this train test split to split it into training set and testing set alright.

So that what would be the ratio for dividing it would be determined by this. So we are considering 25% of the total training set as testing set and the remaining 75% would be considered as the training set and this is stored in this 4 variables alright. So, this variables contains x_train and x_test which contains the training and the testing data respectively and y_train and y_test contains the testing sorry training and the testing levels respectively. Now here I am using another function that is standard scalar which I have imported from sklearn.preprocessing. So this is basically used to standardize our data, standardize our data.

So what it does is that it makes mean of the distribution 0 and the standard deviation as 1 alright. So this is what is being done in these 2 lines. So we are just standardizing the x_train and the y_test data storing it in these 2 variables and after that we are using this k neighbor classifier function that is been already present in this library that is sklearn.neighbors and we are creating a classifier object here. So in this object we are just sending how many numbers of neighbors we are considering that is what is the value of k this going to be so that we have chosen as 5 and we are using metric Minkowski and p equal to 2. So this basically considers the distance as standard Euclidean distance.

So the metric that we will be using to calculate the 5 neighbors would be the Euclidean distance. So that is the meaning of this metric and p value alright. So once this classifier is constructed we are just fitting it with x_train and y_train just because it is a supervised learning technique alright and once the learning is done we are just predicting the x_test value which is being stored in this variable y_pred. Now the y_pred is printed here. So y_pred contains the levels of the test data and in order to show it in a more visually appealing manner or you can say more informative manner we are using the confusion matrix.

So confusion matrix is a function that is being imported from the sklearn.metrics and then we are sending these 2 as the arguments and this CM variable will contain the confusion matrix which being shown in this place and also we are calculating the accuracy score. So that is provided here. So we are getting an accuracy of 94.7%. So that is really a good accuracy here and you might be seeing here that in most of the algorithms or in this algorithm only we are using the sklearn library quite a lot.

So this sklearn library is very important when we talk about ML and DL algorithms okay. So you need to know about this library. After this I am just showing the real values and the predicted values in a tabular format. So I am using this pandas function to show it in a tabular format and you see here the green sorry the blue rectangles which shows the misclassified data. So now you see here this is the true data and this is the predicted data.

So this should not have been predicted as 1, it should have been predicted as 2 but yeah this is the misclassification that we got from this classifier. So there are 3 instances here and I am just showing you 23 samples here because the other samples I could not fit into this slide. So when you try it out on your own you will get a bigger table alright. So this is all about the k nearest neighbor algorithm. Now let us move on to the another supervised machine learning classification technique that uses SVM that is support vector machine.

So like I used in KNN here also I am using 3 libraries that is numpy, matplotlib, pandas and also I am using the sklearn library to load the iris dataset here. So this is being done here, the loading of iris dataset and then I am just splitting the train and the test using this line and then I am just standardizing the data using this code. So this is similar to the one that we did in KNN and after that I am importing this function SVC that is support vector classifier from sklearn.svm and creating the classifier object here. So here we are sending the kernel as linear, so we are considering linearly separable data that is why kernel is linear and random status 0 and then we are fitting the x_train and y_train data.

So the learning process is going to take place in this step. So once this is done we are predicting the test data, so it is done in this step and we are just assigning the values of this step to the variable ypad and this is shown in this portion. So this is the labels that we have predicted and to show it in a more readable manner or more informative manner we are using this confusion matrix here and also we are showing the accuracy score. So now if you look at this value here, accuracy score is 97.36% and this is the confusion matrix.

So you see here that the accuracy is more in case of svm compared to KNN. So svm is a better classifier for this data set that is what we can conclude and this is the tabular

form that I am showing. So for real values I am showing this column and for the predictive values I am showing this column and you see there is only one instance where we are having image misclassification alright. So these are the machine learning techniques that we are going to discuss in this class. Now we move on to the deep learning algorithm which is the convolution neural network.

So we are going to do a classification task in this case. So here also we are using this library numpy and then we are using a new library that is tensorflow. Now this tensorflow is a python friendly open source library for numerical computation that makes machine learning and developing neural networks faster and easier. So we are going to import a few functions from this tensor flow that is keras and another one is layers.

Now this keras.dataset.mnist is used to load the MNIST dataset. So MNIST dataset is a 10 class classification dataset which contains the images of the digits that is from 0 to 9. So now this dataset would be load into these variables. Now here we do not have to do the train test split because it is already has been splited by this function that is keras.dataset alright. Now after we have loaded it to this variables what we are doing we are taking the x_train and the x_test and we are just changing the data type of these values or these variables that is we are converting it into float 32 and float 32 here and after that we are dividing it by 255.

So this is basically the normalization step. So this is done so that the scale of the image comes between the range 0 and 1. So that we maintain a uniformity there and 255 because this is the highest possible value that we can obtain for a 8 bit image alright and after that we are just expanding the dimension of these images. Now the important thing is that when we consider grayscale image it always has height as well as width but no channel alright. So when we are working with CNNs or TensorFlow or the PyTorch library they always expects that we have a channel associated with a image. Now since there are no channel associated with grayscale image we are just including one here alright.

So making it a 3D tensor alright. In case of color images you always have h, w and channel. So this channel is going to be 3 because we have red, green and blue channels that comprises the color image. So therefore in order to get a channel there we are just expanding the dimension of the images by using np.expand_dims and we are using axis as -1.

So -1 means it will be expanding in the last dimension ok. So the similar thing is being done for the x_test as well alright. Now we are printing the x-ray in shape so you can see it over here. So it says 60000 , 28 , 28 , 1. So 60000 is the batch size batch 28 is the

height this 28 is the width and this 1 is the number of channels alright.

And before this I just forgot about this initialization part. So here I am initializing two variables so one is number of class which is assigned as 10 because we are doing a 10 class classification problem and the input shape of each image is going to be 28 , 28 , 1. So this 1 is for the channel and 28 and 28 are for height and width of that image alright. Now in the final steps of this cell what I am doing I am trying to categorize or convert the numerical data into categorical data. Now what actually it is? So if you consider numerical data and suppose say a label is or suppose say an image is labelled as 2.

So in numerical it has the label 2 alright. So in categorical what would be the result? The result would be 0 0 1 0 0 0 0 0 0 0. So you see here all the values that belongs to the other classes would be numbered as 0 and the place that belongs to 2 would be numbered as 1. Now you might be wondering why not consider this place because in python the counting always starts from 0 okay. So since this level was 2 so we are going to this third place that is 0 1 2 and in the third place we are just assigning 1 and the rest we would be having 0.

So this is the categorical form. So this helps in increasing the speed for algorithm. So that is the reason that is why we are considering categorical and it also has other benefits so which I am not going to cover in this lecture alright. So we have converted into a list which contains 0s and 1 and 1 is only for that place for which we have the label alright okay. So now we are forming the model in this portion. So we are considering a sequential model here which I have imported from the Keras function here which we have imported in our previous slide. So this sequential basically helps us to stack the layers one after the other in a sequential fashion.

So what we are doing here is we are importing this Keras.input layer and we are sending in the input shape here. So this is the first layer and after that we have a convolution layer here conv2d which contains 32 filters each of kernel size 3 x 3 and the activation function is relu. So the non-linearity function is going to be or being introduced by the relu function and after that I have the max pooling layer which has a pool size of 2 x 2. Now the important thing is that after its convolution we are introducing this max pooling that is because we want to reduce the spatial dimension of the incoming feature map alright and since we are reducing the spatial dimension in the next convolution layer we are increasing the number of filters so that we can encode more information alright.

So this is just increasing the receptive field of the input or the incoming feature map. So here we are having 64 filters with the kernel size of 3 x 3 and the activation is going to be the relu same as the previous case and then again we have another max pooling layer here alright and after that we are just flattening the incoming feature map. So this is like

vectorizing the incoming feature map and after that we are passing it through a dropout layer and then we are moving it to the dense layer which we can consider as the output layer. So in this case we are having 10 nodes because we have 10 classes and the activation this time is going to be the softmax because it gives us probability values and the class which is having the maximum probability would be assigned as accordingly. So the class with the maximum probability is going to be considered as the predicted class alright and this dropout layer is basically used for regularization. So that we do not have over fitting issues and in this case we are going to drop out 50% of the nodes alright and after creating this model you see here we are just printing the model of summary here and this is the entire model.

So we have the layers here so we see conv2d max pooling, conv2d max pooling, flatten dropout and finally the dense layer which is the output layer and then we have the output shape here that is the shape of the feature map, the shape of the outgoing feature map and then we have the parameters here alright and when we sum these parameters here we get the total parameters here. So total number of training parameters are 34826 alright. So this is fairly a very small model when you will be working with deep learning algorithms you will see that the models are having millions of parameters alright but this is just for the demo purpose so I am creating a small network here alright and you might be wondering what this none is all about. So this none is related to the batch size okay. So since now we have just made the skeleton of the model we are not providing any information related to the batch so that is why we are getting it as none alright okay.

Then now let us come to this so for batch size we have considered 128 and number of epochs we have considered 15. So what is this epoch? So epoch is one complete pass of the training dataset through the algorithm alright. Now we cannot just send in the entire dataset to the model it will just load the model right. So we are working in batches so in this case we are working on a batch of 128. Now let me explain it to you through an example suppose we have 10 samples so these are the 10 samples.

So now let us consider that we have a batch size of 2. So what it means is that we will be working on 2 samples at a time alright then we will continue working on it like this. So after 5 iterations the entire training set will be complete and this is called one epoch alright. So there would be 15 epochs as I have already shown here and after that what we are doing we are just configuring the model using model.compile and here we are using the loss function as categorical cross entropy and then we have the Adam optimizer for optimization and the metric that we would be monitoring in this case is the accuracy alright. After that we are just fitting the model here where we are sending in the x_train the y_train and the batch size here and also the epochs and after that we are doing the validation split which is 10% here.

So now this the entire training set will be split into 2 parts one containing the training set and another containing the validation set. Now this validation set is required so that we can fine tune the model at each epoch alright and since we are using 15 epochs you see the progress of the model for the 15 epochs here. So we have the training loss, training accuracy, validation loss, validation accuracy here alright so these are the values we see the loss is gradually decreasing here and the accuracy is gradually increasing here. So this decreasing and this is increasing same goes for validation loss as well it is decreasing gradually and it is increasing gradually for the validation accuracy and I have also shown it in a graphical plot. So you see here the training and the validation accuracy are consistently increasing and the loss is consistently decreasing.

Now when does the problem arise? Now suppose this validation loss starts to increase from this point like this. So this would be a case of overfitting which we do not want for our model alright. So if overfitting occurs we need to look or maybe change the model or maybe work with the number of layers or maybe introduce some regulation technique. So overfitting is something that we do not want for our model alright.

So we can say that after 8 epochs it is going to overfit if this is the case alright. Now since this is not the case for here so we can say that this is basically a perfect model that we are training here alright. So now here I am just evaluating the model by sending in the x_test and the y_test and I am just showing the test accuracy and the test loss here.

So the test loss is 2.54% and the test accuracy is 99.08% here. So it is a really good performing model and after that I am just predicting the model for the x_test data and we are showing it in this variable PRDT. Now initially since we have converted the levels into categorical form we need to convert it back into the numerical form so that we can see the confusion matrix for this data. So this is what we are doing here we are just finding the maximum position where it is occurring. So we are using np.argmax and then we are assigning it to the variable y_pred. Similarly for the ground truth data we are doing np.argmax for y_test and storing it in y_classes variable.

And then we are sending in the y_classes and y_pred which is the ground truth and the predicted value to the confusion matrix and we are obtaining the confusion matrix here. So you can see directly this is the confusion matrix here and we are also plotting the accuracy score sorry we are printing the accuracy score so this is the accuracy score here alright. So you can see that the CNN performs much better than the other machine learning algorithms though we are using separate data set but you see here in this case we use the 10 class data set in the previous case that is the machine learning case we use the 3 class data set and in spite of that we are getting a better performance alright. And in this case I am just trying to show the confusion matrix in a more visually appealing

manner by using this library this seaborn.heatmap alright so this I have imported here.

So you see what is the output of this seaborn.heatmap so this is what we get as the output. So you see here the diagonal elements are shown in light color to show that this has got the higher values as we can make it out from the color bar and the non-diagonal elements are having low values which is good for us and this is shown in the color bar here alright. So this is visually more appealing diagram for the confusion matrix so you can definitely use the seaborn packets to get things more visually appealing alright. So in this class we talked about Colab how to use it and how to access the GPUs that is being provided to us by Google and then we talked about 4 ML algorithms they are linear integration, k-means clustering, k-nearest neighbor and SVM that is support vector machine and out of this KNN and SVM are classification algorithms. After that we have also tried the CNN a custom made convolution neural network which perform very good for a 10 class classification problem. Now this is just a basic demonstration of this algorithms and I hope that you got an idea of how to practically implement this ML and DL algorithms and you can definitely explore the other complex algorithms in this field that is the ML algorithms and the deep learning algorithms and see how it performs for those cases.

I hope you learned something from this class and you enjoyed this class and with this note I conclude today's lecture. Thank you and have a great day.