

Deep Learning
Prof. Prabir Kumar Biswas
Department of Electronics and Electrical Communication Engineering
Indian Institute of Technology, Kharagpur

Lecture – 50
Training Trick, Regularization, Early Stopping

Hello, welcome back to the NPTEL online certification course on Deep Learning. In today's lecture we will talk about Training Tricks. In fact, if you think of what we are discussing for last few lectures more than 10 lectures or so we are discussing something on training tricks only right. Our purpose is how can we make the learning of a neural network more stable, error free and faster.

So, whatever we have discussed few classes on say momentum based techniques then accelerated gradient we have talked about (Refer Time: 01:18), we have talked about RMSprop and all that the purpose of all those different techniques was how can we make your training faster or the learning rate can be faster. Similarly, for last few lectures or say 3 4 lectures we have discussed about different normalization techniques.

The purpose of normalization techniques is to take care of the covariate shift of the training data both at the input layer as well as in the hidden layers. So, whether we talk about the batch normalization or you talk about layer normalization or you talk about instance normalization or you talk about group normalization. The purpose of all these normalization techniques is to normalize the distribution of the training data and as the distribution is normalized the neural network while training do not have to hop from one classifier to another classifier or one boundary to another boundary.

As a result the training of the neural network becomes faster and it is more efficient. And we have seen as reported in some of the literatures, that if you go for normalization techniques the performance of the network in terms of error rate both the test error rate, training error rate and the validation error rate reduces.

And out of all these we have found that possibly group normalization techniques performs better in the sense it is more stable in the sense that group normalization technique performance does not depend upon the batch size but, the batch normalization technique possibly performs better in terms of error rate.

So, whatever we have discussed for previous few classes are to improve the learning algorithm to improve the gradient or batch gradient descent algorithms. Now, when we talked about normalization one aspect we have missed that is during training you are computing the standard deviation and the mean for normalization of the training data.

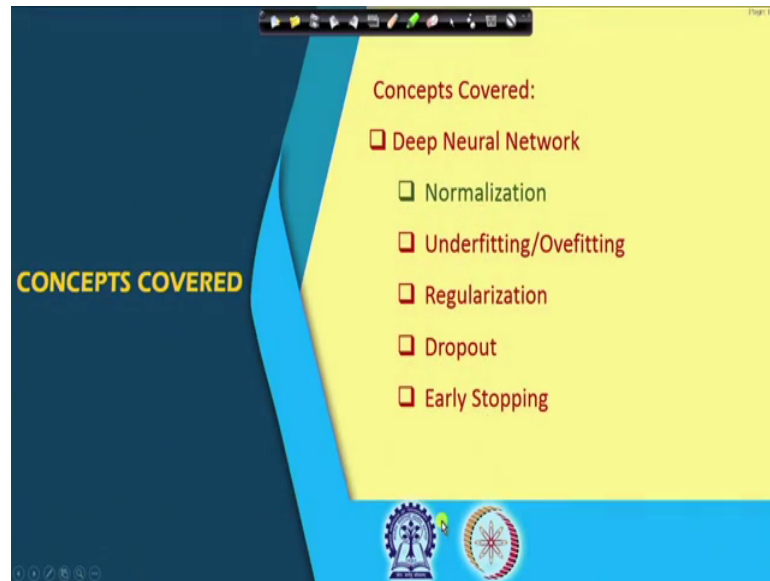
But during this time or during inferencing, you have to work on single data given an input data you have to find out that to which of the class that example the test data belongs and that time I do not have a group to compute what is the mean and standard deviation. So, how to compute mean and standard deviation at that time?

So, you compute you can compute the mean and standard deviation or you can use the mean and standard deviation as has been computed during the training time from different batches. So, there can be 2 ways you take consider all the training examples together and compute mean and variance. But that is a computationally intensive process a tedious process, but what can be done is as for every batch during the training time the mean and standard deviation was computed.

So, the final mean and standard deviation can be used as exponentially decaying average of mean and standard deviation has computed over different batches. And this exponentially decaying average of the mean and standard deviation can be used for normalization of the test data at the time of testing or normalization of the input unknown data at the time of inferencing. So, that is what can be done for normalization of the test data, or the normalization of the live data which will be used for inferencing.

Today we will talk about some other techniques of other tricks of training particularly to make your training more stable and while doing so we will talk about few problems like, what is an under fitting problem and what is an over fitting problem. To take care of this under fitting and what over fitting problem what are the different techniques that you use, the techniques are regularization one of the technique is dropout and the other technique is early stopping.

(Refer Slide Time: 05:32)



In fact, a regularization and top dropout these 2 different topics we have mentioned through our earlier discussions some time. But let us have a brief overview of what is regularization and what is dropout after we introduce what is an over fitting problem or what is an under fitting problem. So, while doing so one point should be made it clear that when you train your neural network or deep neural network there should be different sets of data.

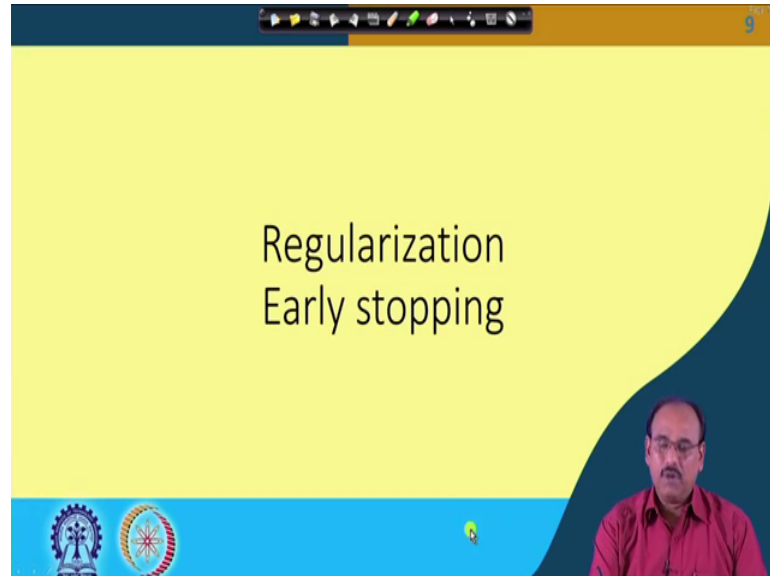
From the same data pool you have to have one partition of the data which is known as training data. So, training data is actually used for training of the neural network. Then I had to have another set of data which is again randomly chosen from the same pool which is known as validation data and there is a third pool which is known as test data.

So, when you train the neural network after every epoch of training with the help of training data, you have to validate the training with the help of validation data. And when the network is trained and validated satisfactorily, then you use the test data to check what is the performance of the network that is already tested and validated.

And it is to be noted that all these different partitions that is your test, data your validation data and the training data they should be mutually exclusive or they should be disjoint. In the sense that the data with which you validate or the sample with which you validate or the sample with which you test your trained neural network these samples should not be used while training the neural network.

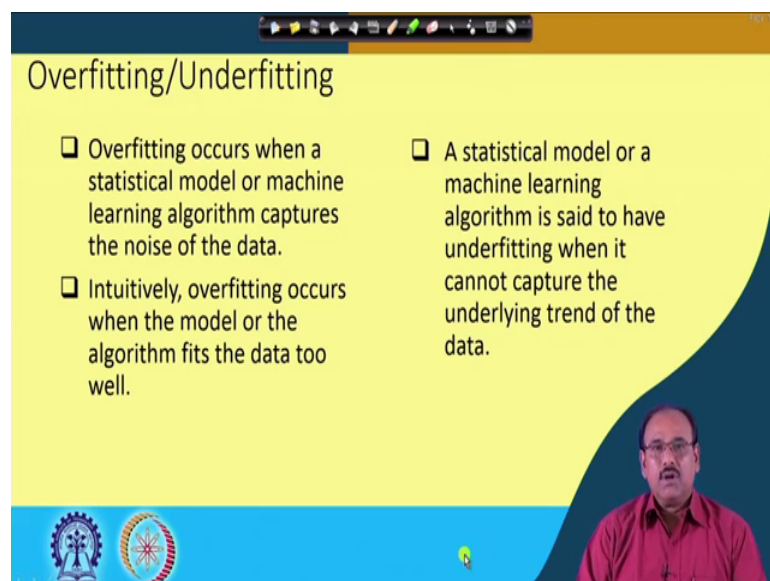
So, I had to have three distinct sets; one set is training data set, one data set is validation data set and another data set is testing data set ok.

(Refer Slide Time: 08:07)



So, with this now let us go to what is over fitting and under fitting problem and through that how regularization will take care of the overfitting and under fitting problem and subsequently how do we use early stopping to stop the training of the neural network at the proper point.

(Refer Slide Time: 08:28)

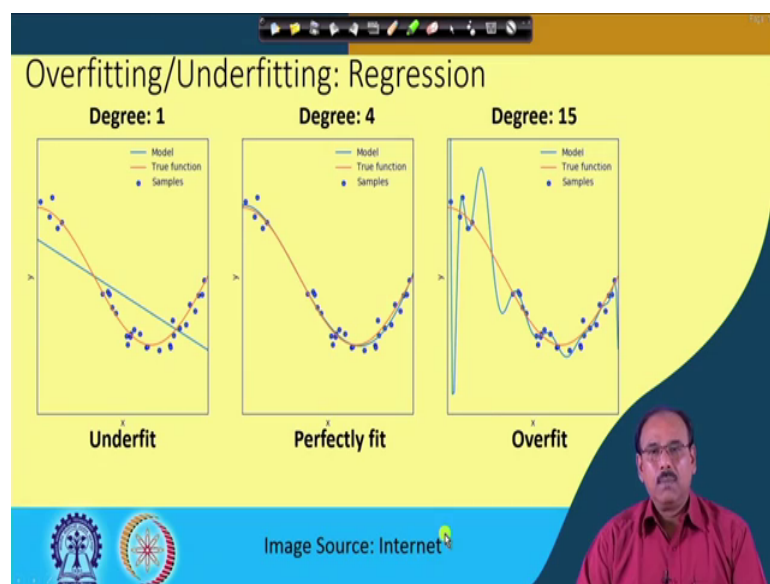


So, what is this overfitting under fitting problem? See overfitting occurs when a machine learning algorithms captures or fits the training data too well. I mean in a simpler term we can put it in this way that an overfitting occurs when a machine or a statistical model learns or captures the noise of the data, noise in the training data. That is it whatever training data is given it learns that training data or the distribution of the training data too well.

And as a result it may so happen that if your training data has got some noise; even that a noise will also be taken as a part of training data. And in the process the network because it learns the training data too well it may skip to understand or to learn what is the structure of the data. And as a result when you test or in the actual deployment your learning algorithm or the machine that has been trained that may not perform satisfactorily or it may not perform well.

So, this is the problem of overfitting. The underfitting problem is just the opposite that the under fitting problem in under fitting problem the machine cannot even capture the underlying trend of the data or it cannot capture the distribution of the data itself. So, these are two extremes. In case of under fitting the machine cannot capture the true distribution whereas, in case of overfitting even if there are finite minute details of the training data the machine captures that as well. So, both of this overfitting and underfitting are not desirable in case of machine learning algorithms.

(Refer Slide Time: 10:36)



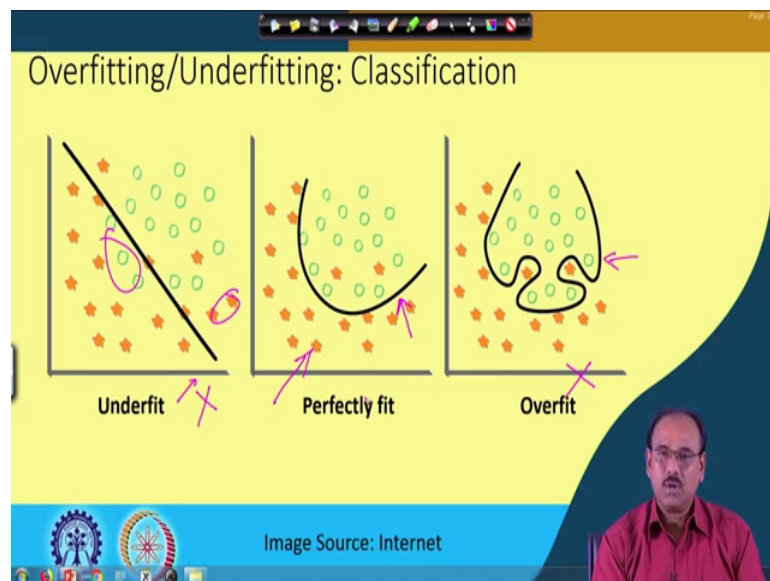
So, to put graphically you find that all these blue dots in this examples they are the training data and the red curve over here these red curve that represents the true model. So, this is a regression problem and we say that the underfitting occurs when the machine or the regression the regression regression model cannot capture the data structure of the data or the distribution of the data.

So, if the regression model is a linear model in that case you find that this set of data will be fitted by a straight line. So, naturally this is not the one which we want this is what is an under fitting problem. In case of an overfitting problem the regression the regression regression model this fits a polynomial offset degree 15 with this data.

So, as a result you find that there is no error or minimal error given by this fitted curve fitted polynomial features of degree 15, but in the process it has not understood what is the actual structure is or what is the actual distribution of the data. Whereas, in this case in the middle one this regression model which is fitted again a polynomial, but now it is of degree four and here this model has captured the underlying structure of the data and this is what is perfectly fit.

So, we have the over fitting, we have under fitting and we have perfect fitting. So, how does it work in case of our machine learning algorithms?

(Refer Slide Time: 12:36)

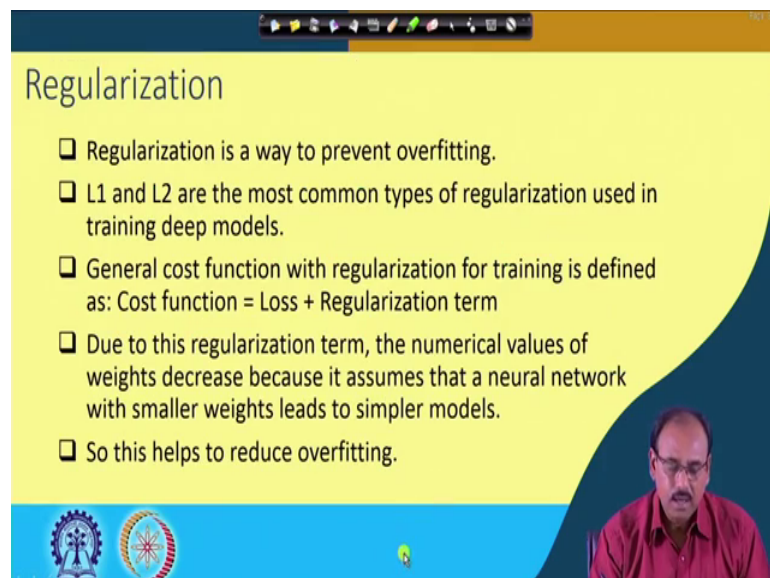


Say for example, over here you find that all the green circles they are the data set the training data set from one class and all the orange dots or orange stars they are the data samples from another class. So, a classifier has to learn the boundary between these two classes. On the left hand side you find that it is a case of under fitting, because what the classifier learns is a linear classifier and as a result there are lot of misclassifications. You find that on the right this is an overfitting case, because the classifier has learnt the training data too well.

And as a result it has missed to learn to understand what is the structure of the data, what is the inner structure of the data and that is what has been learnt in this middle case where we say that it is a perfect line. So, while training a network while training a machine learning algorithm, what we desire is to have a situation something like this; we do not want a situation which is an under fit situation neither we want an over fit situation.

So, this is what we have to get and that is where your regularization and early stopping comes into picture. So, let us see that what are these different regularization techniques that we can have.

(Refer Slide Time: 14:07)



The slide is titled "Regularization" and contains the following text:

- ❑ Regularization is a way to prevent overfitting.
- ❑ L1 and L2 are the most common types of regularization used in training deep models.
- ❑ General cost function with regularization for training is defined as: $\text{Cost function} = \text{Loss} + \text{Regularization term}$
- ❑ Due to this regularization term, the numerical values of weights decrease because it assumes that a neural network with smaller weights leads to simpler models.
- ❑ So this helps to reduce overfitting.

The slide also features a small video inset of a man in a red shirt speaking in the bottom right corner, and logos of institutions in the bottom left corner.

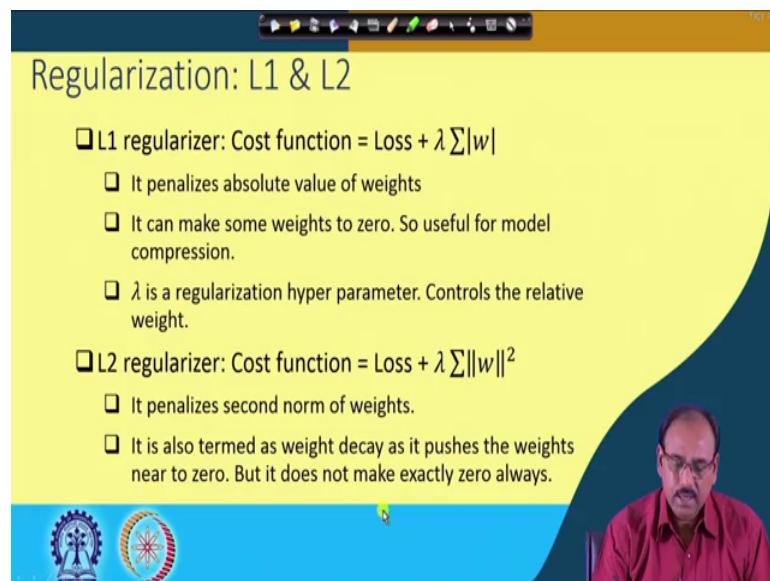
So, a regularization is a technique which tries to prevent the overfitting and while doing so we have discussed earlier. When we were talking about when we are formulating the different cost function or the loss function. That we had along with the data loss term we

also had a regularization term or regularization loss term. And there we have seen that we can have different types of regularization say L1 regularization L2 regularization and so on.

And your total cost function in general becomes a some of that data loss term, which is as given over here. So, this is the data loss term plus we have a regularization term. So, what this regularization tries to do is it tries to make the numerical values of the weight vectors, of the weight parameters to be small. Because it is believed that if your weight vectors are small then the numeric the neural network with smaller weight values will lead to a simpler model right. So, and it will not over fit the data.

So, this regularization term whether it is L1 regularization or it is L2 regularization. This helps in avoiding the overfitting problem. We can have other techniques as well regulation techniques.

(Refer Slide Time: 15:48)



Regularization: L1 & L2

- ❑ L1 regularizer: Cost function = Loss + $\lambda \sum |w|$
 - ❑ It penalizes absolute value of weights
 - ❑ It can make some weights to zero. So useful for model compression.
 - ❑ λ is a regularization hyper parameter. Controls the relative weight.
- ❑ L2 regularizer: Cost function = Loss + $\lambda \sum \|w\|^2$
 - ❑ It penalizes second norm of weights.
 - ❑ It is also termed as weight decay as it pushes the weights near to zero. But it does not make exactly zero always.

So, just what this L1 and L2 regularization does. In case of L1 regularization your total cost function becomes data close data loss plus some lambda times sum of absolute values of weight. So, this tries to make your weight parameters to be smaller. And lambda is our regularization is a it is a hyper parameter which tries to balance or controls the relative importance of your data loss term and the regularization term.

So, this hyper parameter lambda it has to be manually set. Similarly in case of L2 regularizer, the loss function simply becomes data loss plus lambda times modulus of w square. So, here again it tries to penalize the weight components or weight parameters which are too large. But because of both this data loss as well as regularization term both of them being optimized simultaneously, that weight vectors does not become 0. So, this is what is L1 regularization and L2 regularization.

(Refer Slide Time: 17:09)

Data Augmentation

- ❑ Increasing the size of training data is a way to prevent overfitting.
- ❑ It is difficult and costly to increase the training data.
- ❑ Data augmentation is a way to create a different image from one image while keeping the context same.
- ❑ There are a few ways of augmenting training data—rotating, flipping, scaling, shifting, contrast enhancement, brightness control, etc.

The other way in which the overfitting can be addressed is by data augmentation. So, we have seen when we talked about the de noising auto encoder is that while training you feed the network with the noisy data, where as your output is the actual data. So, your auto encoder tries to reconstruct in the output data from the regular from the noisy data which is given as input.

So, while doing so it tries to understand or tries to learn what is the inner structure of the data. And by doing that you are preventing the auto encoder to learn the data itself or to memorize the data itself, but rather the auto encoder tries to learn what is the inner structure inner structure of the data. And using this knowledge of the inner structure it can reconstruct the original data.

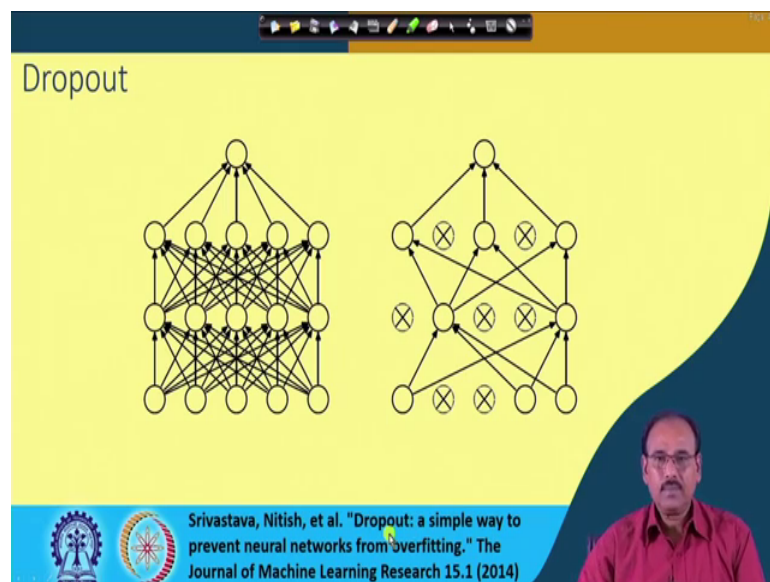
So, data augmentation is can be done many ways to avoid overfitting what we need to do is as well as under fitting what we need to do is we have to train your machine with large and large amount of data and that data should have a lot of variation. But in practice it is

extremely difficult to get the training data which is large in volume at the same time which will have wide variation. So, that one way to generate such data is to data augmentation.

So in case of data augmentation what you do is given the input data or a limited set of data using that limited set of data you create augmented data by you adding noise to it, by different types of transformations, applying to it by rotating to the data, by flipping the data, by scaling, by shifting, by contrast enhancement, by modifying the brightness and all these things. So, by all these different operations from a limited set of training data you generate a large volume of training data.

And that is what is your order data augmentation technique and you train your machine train the neural network using the augmented data. And because now you have large volume of training data with lot of variations while training the neural network will try to learn the inner structure of the data. It will not be able to memorize the data and that is what is required to avoid the overfitting problem. The other kind of technique of regularization or to avoid the overfitting problem is dropout.

(Refer Slide Time: 19:47)



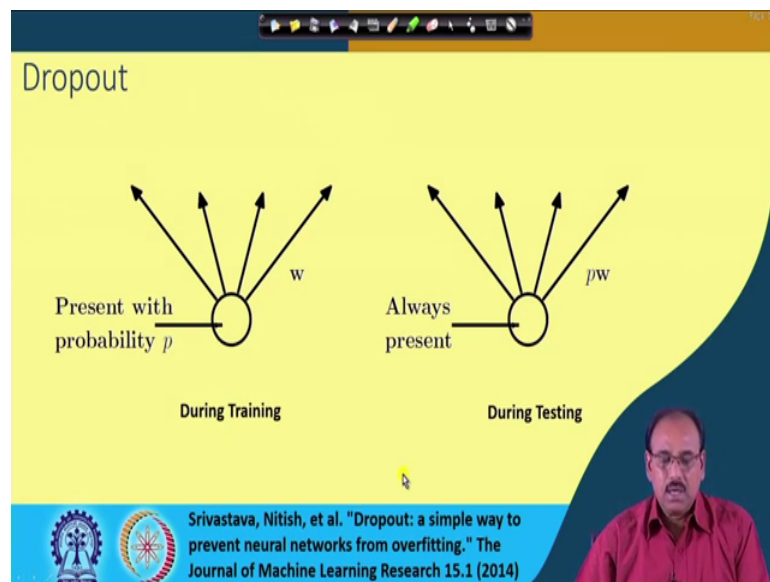
So, what is dropouts? Dropout is during the training operation, you choose the different nodes at random or with certain probability associated with it to be dropped out of the network. So, you say that so this is what has been shown over here so on the left what we have is the full network with full connection and on the right hand side what has been

shown is these are the nodes which have been dropped out of the network during training process and this is done at random. The purpose of dropout is that because of this dropout and different nodes are being dropped out at different training epochs.

It is not the same set of nodes which have been dropped out the network cannot memorize the training data at all rather when you are dropping out a node from a network this node does not participate in the forward pass by forwarding the output of its activation to the nodes in the subsequent layers nor it participates in the backward pass during the training operation by feeding the gradient term that it has received to the layer before it. As if this node is not present in the network at all.

So, as for the same set of training data at different training epochs different nodes are dropped out at random. So, as a result the network learns different representation of the same data ok. So, as it learns different representation of the same data it is not memorizing the training data. So, by doing this you can avoid both the overfitting as well as under fitting problem.

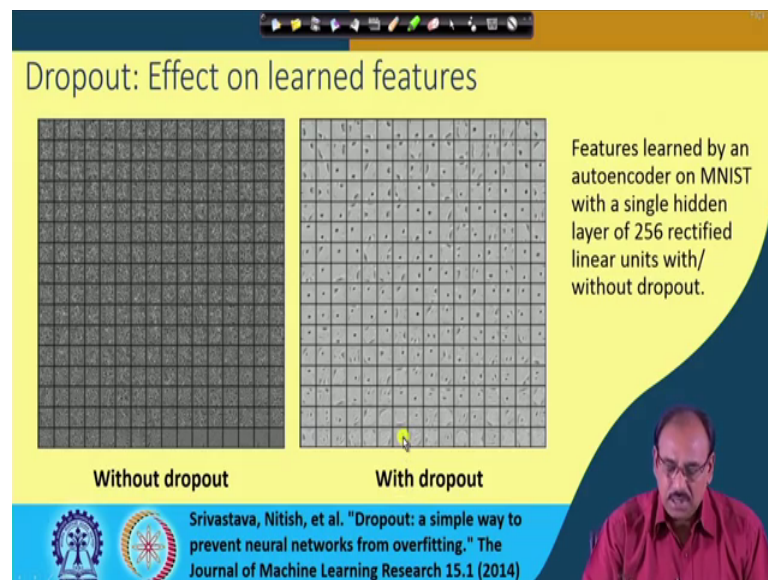
(Refer Slide Time: 21:56)



So, this slide shows that how this dropout mechanism is actually implemented. So, during training the different nodes are present in the network with certain probability p or it is dropped out with a probability one minus p , but during testing there is no drop out all the nodes are present in the network during testing time and during inference time right.

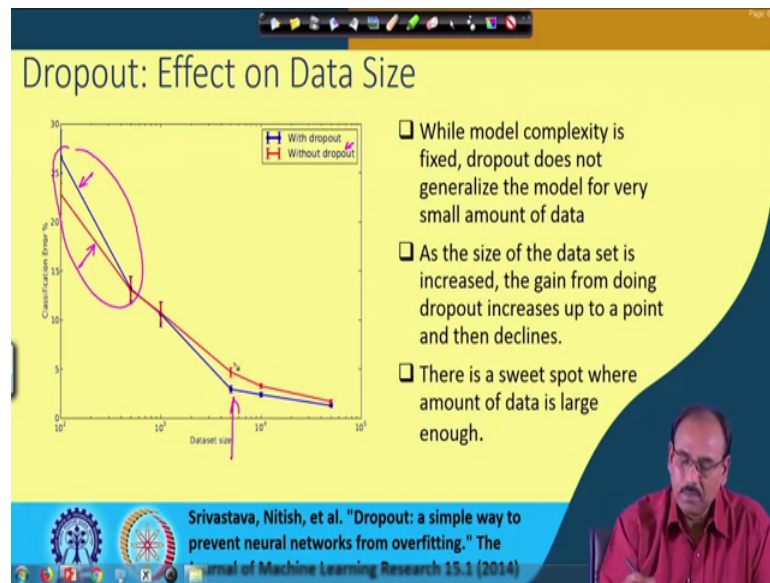
So, as the different nodes are dropped out during the training operation at random. So, the training becomes more robust and the network learns multiple representation of the same data by which your training process becomes more robust and it is more regularized in the sense that it will avoid the under fitting and over fitting problems right.

(Refer Slide Time: 22:51)



So, in addition to this so this slide simply shows that the features which are learnt without a dropout and with drop out. You find that when you are going for with the drop out the features learn certain structures within the data as given over here on the right hand side. So, when the dropout is employed the there are some structure which is which can be seen within the features which is learnt, but without drop out such structures are not visible within the features we just learnt. So, this is why with drop out your training becomes more robust.

(Refer Slide Time: 23:43)



And this slide again shows that drop out does not in general is the model when the data size is very very small. So, the performance of the drop out technique improves with the data size up to a certain level; up to a certain level after that say as shown over here, so as you increase the data set the classification error rate goes on reducing when you have this drop out in effect.

So, up to over here. So, this is the performance without dropout and this is the performance with drop out. And beyond this you find that, so as you increase the data set your performance of dropout goes on improving, but beyond certain point the performance is improvement in performance is not that prominent. So, that data size also plays a role to decide what is the performance of the top of technique right.

(Refer Slide Time: 25:00)

Early Stopping

- ❑ Hyperparameters need to be tuned for good performance while training neural networks.
- ❑ Number of iteration is a hyperparameter to be tuned. Lesser iteration may lead to underfit and more iteration may lead to overfit.
- ❑ Early stopping attempts to remove the need of manually setting this value.
- ❑ It can also be considered a type of regularization method.

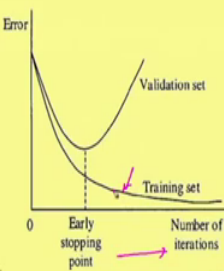


Image Source: Internet

In addition to this you find that we also have to decide that when to stop the training of the network. As we said earlier that we have to have three different data sets, we have to have the training data set, we have to have the validation data set and we have to have the testing data set.

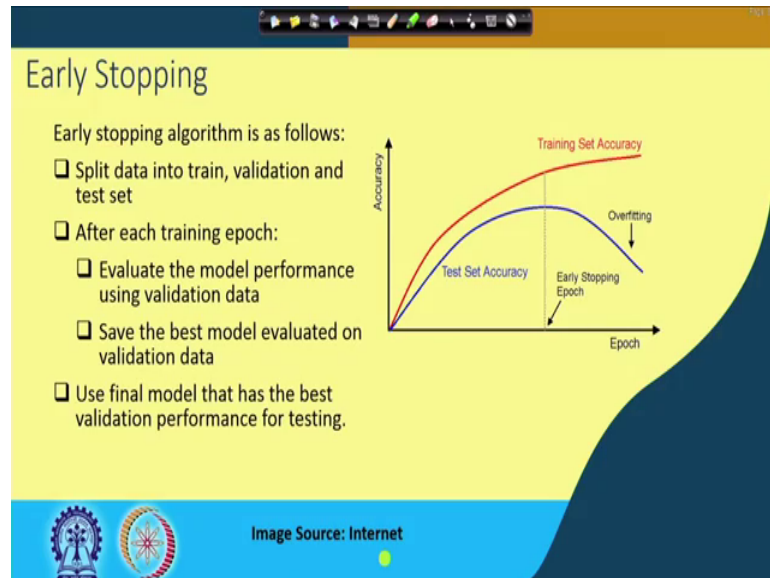
The purpose of validation is as the network is trained; after every epoch of training you have to use the validation data set to check what is the performance of the trained network at that point of time and it is believed that if the network is trained for a longer period of time then there is a risk of overfitting. In the sense the network tries to learn or fit too well to the training data as a result it skips that regularization part of it.

So, we have to have a proper mechanism to capture when the network is actually overfitting the data. And that is what is captured through the validation process. So, if you look at this particular diagram you find that with different training epochs as the number of iterations increases your error on the training data set that goes on reducing, but the error on the validation data set that reduces up to certain points and beyond that the error and validation on validation data set goes on increasing.

So, this is the point where we have to stop; we have to stop the training operation. Because beyond this as the training error goes on reducing. So, this is an indication that the overfitting has started occurring. And at the same time your error on the validation

data has started increasing. So, this is what is your early stopping point where we should stop training of the network.

(Refer Slide Time: 27:06)



So, the same is visible over here so what we do is you train the data train the network using the training data and after every training epoch you try to validate the network. So, during validation you use the validation data and find out what is the performance of the network on the validation data at that point of time. Every time you have improvement with the validation data.

So, whatever is the parameter of the network at that point of time you save it somewhere you save that parameter and then when you find that the performance of the network has started decreasing on the validation data what you do is in the final model that you use you use the parameters the best parameters that has been saved during the validation process. And you have to stop training the network at that particular time.

So, this is what is early stopping here during training, you do not only check the performance on the training data because the performance of the training data may go on improving because even if the model over fits the training data the performance error will be 0 right. So, the performance on the training data set will go on improving, but that is not what we want because in machine learning techniques you are training the day machine with the training data, but you want that it should perform well on the validation

data as well as in test cases. Our performance on the training data is not the only thing that we are looking for.

So, the performance of the training data even if it improves, but the moment the validation performance goes on decreasing you have to stop the training at that point and that is what is known as early stopping. Well, so far what we have discussed is the techniques or the training tricks by which your training is more efficient it is more stable and it is faster and now also seen that we have to stop the training at certain point of time.

So, that the machine or your neural network does not over fit the training data. So, that it performs well in the actual deployment give you less generalization error and that is where your network actually performs well. We will stop; we will stop here today and we will discuss about other topics in our coming lectures.

Thank you.