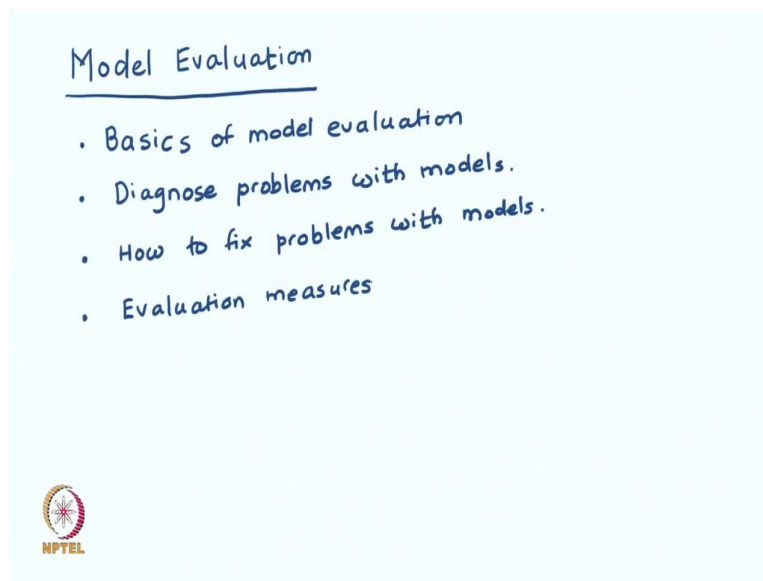


**Practical Machine Learning with TensorFlow**  
**Dr. Ashish Tendulkar Google**  
**Department of Computer Science and Engineering**  
**Indian Institute of Technology, Bombay**

**Lecture – 07**  
**Model Selection and Evaluation**

[FL]. Welcome to the next module of Practical Machine Learning course. In this module we will study the techniques around model evaluation.

(Refer Slide Time: 00:33)



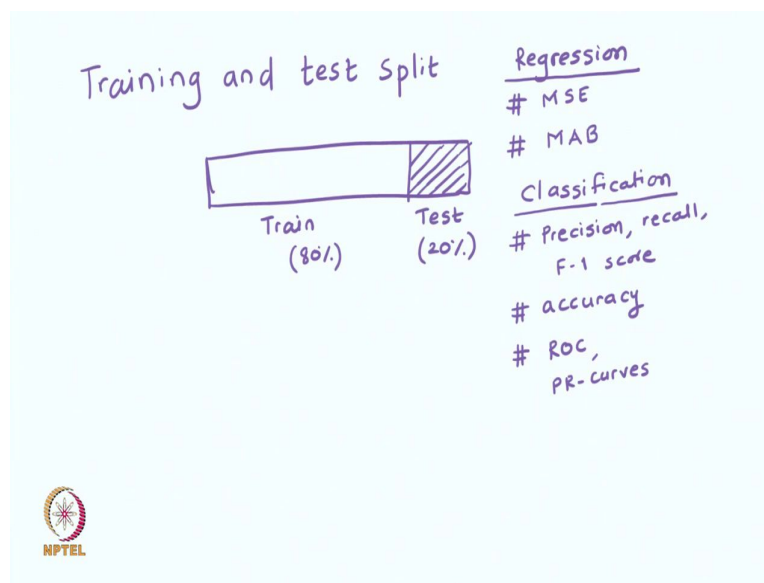
So, far we looked at other steps in machine learning, like data gathering, transformation, defining models, defining loss functions and training the model. The final step in the machine learning pipeline is really the model evaluation. In model evaluation, we are concerned with the question of how well the model will work on the unseen data. And, we are also interested in diagnosing the problems like underfitting and overfitting with the model and taking corrective actions.

Finally we will also study a bunch of evaluation measures that are established in practice, let us begin. So, as I said, we expect the model to work well on unseen data. Now, how do we really test the performance of the model on the unseen data? More than unseen data we really

want model to work well on the future data, we test we train the model on the past data, but we want to make it work on the future data and we do not have the future data in hand.

So, how do we really solve that particular problem? So, one way of solving that problem is taking the entire trained data and dividing it or partitioning it into two parts - one is a training and second is a test.

(Refer Slide Time: 01:57)



We use a large chunk of the data for training and hold out a small percentage of data for test. Usually, we use 80 percent of data for training and 20 percent data for testing. And, the model will be trained based on training data. Model does not get to see the test data.

If, by any chance your model gets to see the test data, then the performance that you report on test data, will not stand the scrutiny of the time in the sense that, you might get overestimation of the model performance.

How do we really measure the performance of the model? So, there are standard measures of performance. So, in the case of regression, we have mean squared error or mean absolute error. In mean squared error, we take the square of the error between the actual value and the predicted value. In mean absolute error, we take the absolute difference between the square and calculate mean of that. So, these are two measures that we use in case of regression.

In case of classification, you first compute the confusion matrix, and we calculate measures like precision, recall and F1 score to measure the accuracy or to measure the performance of the classification.

Sometimes we also use accuracy as a measure; accuracy as a measure is not stable it does not work well when your data set is imbalanced, when you have way too many examples from 1 class and too few examples from the other class, accuracy is not a good measure to use or it is not a good metric to use.

Apart from that we use ROC curve and PR curves to understand how our classification is working across different thresholds. So, we will go through each of these terms later in this module. Right now, it is enough to understand that we will be using one of these measures, and reporting performance on the test data. How do we believe that the model that worked well on the test data is also going to work well on the future data? Why do we really believe this?

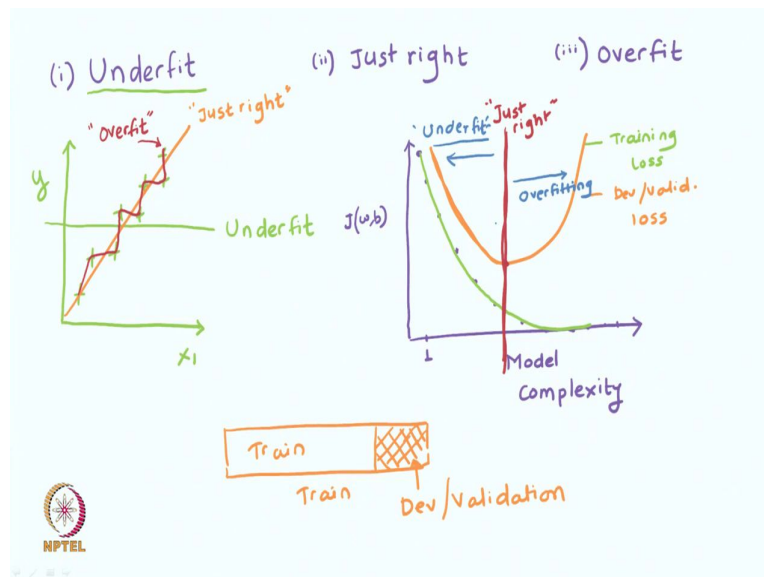
Here comes a very important caveat in machine learning - we assume that the training data and test data, are going to come from the same distribution. If, training data and test data is not coming from the same distribution we do not have much hope.

So, let us understand what do we mean by training and test data coming from the same distribution. Let us take an example of a kid prepare preparing for an exam, we tell the kid to prepare prepare for maths exam. And, the next day we give him questions from the maths, maths text book or maths syllabus. So, here since kid has already prepared for maths, he will be able to answer questions about maths. So, this is a nice example of training and test data coming from the same distribution.

And, imagine a case where we ask a kid to prepare for maths, but we ask questions in science. So, really kid would not be able to answer these questions, because he never trained on science. So, this is an example where training and test data are different. So, this is an important caveat to note, that machine learning machine learning makes an assumption that training and test data comes from the same distribution ok. Now hat we have learnt how to

train the model and evaluate it. Let us try to understand three possible scenarios that can happen to any model after training.

(Refer Slide Time: 06:57)



So, your model will either underfit, or it will be the right fit, or it will overfit, let us try to understand the 3 concepts through a diagram. Let us say your model that has a single feature and it is predicting the value  $y$  and let us say these are data points. So, let us say my model is a straight line; that means, it predicts some constant value. So, let us say my model is  $y$  is equal to 2. So, no matter what is the input it always predicts the value 2.

So, this is an example of under fitting, where the model is not taking into account a training data at all and it is going by a very strong bias that no matter what is the input, I am going to spit out the output which is a constant, which is which is a fixed number, which is two in this case.

So, the second thing that could happen to your model is that it will it will be doing just the right fit. So, let us use this particular color and this is an example of right fit. So, it appears that the points are; so the points are around a line. So, a linear model might be just the right fit for this particular model. So, this is the category of right fit.

In over fitting you end up achieving a 0 error on the training data, but your model may not generalize well on the future data. And, one of the important things that we care about in

machine learning is to have a model that generalizes well on the future data or that makes a better predictions in the future data.

So, one of the goals in machine learning is to build models that make good predictions on the future data rather than on the past data. So, that is why over fitting is bad for us. So, if you slightly perturb or if you select, if you select, any other point on the training data, you know the overfit end model would not be able to predict it. In over fitting the model is literally memorizing all the data points all the input output pairs.

So, we studied so there are under fitting and over fitting is our problem. And, we understood what is under fitting and over fitting. Now, what is now the next question is how do we really diagnose under fitting and over fitting?

So, in case of under fitting you are using a model with a single parameter which is bias whereas, in over fitting we are using a polynomial of 7 degree to get a perfect fit on the training data. So, these are problems of either there are too few parameters in case of under fitting or there are too many parameters in case of over fitting.

So, our job is to come up with the right set of parameters for getting just the right fit. And, let us define the complexity of the model or let us define a measure of complexity of the model, and one of the measures is the model complexity is measured by number of parameters that are used in the model.

So, in case of under fitted model here the model complexity is just one in num in case of number of parameters whereas, in case of this degree 7 polynomial, there will be far more parameters than the under fitted model. Whereas, just right model has 2 parameters; one parameter with respect to the bias and second parameter which is a coefficient for feature  $xy$ .

When you have two few parameters the model generally has very high loss and as we go on adding more parameters the model will become better at training error. So, model will have lesser and lesser training error. This is the training loss and now you have to understand, whether model is under fitting or over fitting. So, what we do is we have already divide training data into train and test part. So, we take this train part of that and we divide it further into two parts; one is called the train and the second is called dev, dev set or validation set.

So, we measure the training loss on the train set and you also measure the loss on validation. So, note here that validation data is not shown in training algorithm and a training time. So, whatever loss you are measuring when the dev set is similar to the loss that we measure on the test set. So, we will we will get the dev loss. So, what will happen is the dev loss or the validation loss will reduce initially as we increase the num as we increase the number of parameters in the model or as we increase the model complexity.

After a point it will shoot up so, no matter how many more parameters you increase, the training error will go down and keep going down, but the validation error keeps going up. So, if we look at this particular graph it seems to us that this is probably your sweet spot where the validation error is low enough and training error is also low.

Beyond this point validation error has short up. So, this is this point corresponds to just right fit ok. On the right hand side we have over fitting. What is happening here in case of over fitting? The validation error initially improved, but after a point it went up.

So, if you see this kind of pattern in the training error in case of you in your learning curves or in the model complexity versus a loss curve we can be sure that there is some kind of an over fitting happening whereas, this particular part to the left of just right is called under fitting zone so under fitting zone. So, we do not just we simply do not have enough capacity or we do not have enough parameters in the model to achieve a lower training and validation error in this particular part.

So, if your training error and validation error is high, we can we can assume that there is some kind of an under fitting happening. Whereas, if your validation error went down before going up, then we can say that we have an over fitting happening. In case of just right fit we have training and validation error both low and then we get a model which is to re just right fit for the data.

So, this is more like an exploratory exercise where in order to so, you will have to cons you will have to conduct multiple experiments starting with simpler model and adding more and more parameters to it, and checking out, the training loss and the validation loss.

The training loss is computed on the training set, dev or validation loss is constructed on Dev or validation set and we and we and we observe the patterns, or we observe the numbers, or we observe the loss, to figure out whether we have an over fitting situation or an under fitting situation. And, now know based on what we just learnt, what is really the problem with the model and how we can take corrective actions based on that. So, let us try to understand how we can fix under fitting.

(Refer Slide Time: 18:27)

"Fixing underfitting"

(i)  $x_1, x_2$   
 $x_1^2, x_2^2, x_1 x_2$

(ii) Reduce  $\lambda$  to fix underfitting

"Fixing Overfitting"

(i) Get more data

(ii) Reduce model complexity

"Regularization"

$= J(\omega, b) + \lambda$  "model complexity"

$L_2$ -regularization:  $\sum_{i=1}^m \|\omega_i\|_2 = \sum_{i=1}^m \omega_i^2$

$L_1$ -regularization:  $\sum_{i=1}^m \|\omega_i\|_1 = \sum_{i=1}^m |\omega_i|$

NPTEL

So, once you recognize that under fitting you have is your problem; that means, your model is not getting lower loss or validation in training set both the losses are high. We can infer that when a model does not have enough capacity or model does not have enough parameters to learn the training data or to learn the model on the training data.

So, what is the solution here? The solution here is to increase the complexity of the model, because your model simply does not have the capacity to learn patterns from the training data. So, here one of the one of the solutions that works well here is to add more parameters to the model. How do we add more parameters to the model? Either you get more features or we can do we can construct features by crossing the existing features or by constructing several polynomial features of different degrees.

So, for example, concretely you have two features in the original model, we can construct the features of polynomial sub degree 2 by raising the power of each of the features, and by taking the cross or by taking the feature cross or by taking the feature cross, which captures interaction between 2 features.

And, when we do this we are automatically increasing the capacity of the model. So, feature crossing is a standard way of increasing the complexity of the model. Now, how do we fix the over fitting? So, let us. Why does over fitting happens? Because we simply have too many parameters. So, one way of fixing over fitting is straight away getting more data.

If, you get more data we will we will have enough data to train for the complex model or model having large number of parameters. Second is we can reduce the model complexity. So, one way of doing this is through regularization. Here the idea is that we will penalize the model for being excessively complex.

So, you remember you must be remembering the loss this is a loss function, we have loss function and in loss function we are trying to minimize the loss function. So, we add a penalty which is lambda times the model complexity.

So, our new loss function becomes the loss plus the penalty for the complexity. Lambda is the factor that controls the controls the weight or controls how important, how much importance to give to the model complexity, it is called the regularization rate is lambda and there are various ways of where there are various ways of calculating model complexity.

So, one way to calculate model complexity is called L2 regularization or ridge regularization. In this case what we do is we calculate the second norm of the parameter values. So, essentially what happens is we calculate the second norm and we add the second norm as a as a penalty. The effect of this is we cannot have a model that minimizes the joint objective by assigning a very large rate on one of the one of the features. So, that would not be possible with L2 regularization.

The other measure is or the other way to do this is through L1 regularization, where instead of using the second norm, we use the first norm or the absolute value of the parameter of the parameter.




So, by adding these penalty terms, we are able to control the model complexity. So, if you are if you have the problem over fitting it is good, it will be a good idea to increase the value of the regularization rate or lambda. So, if you increase the value of regularization rate here, what will happen is that we will be giving more importance to the mod model complexity.

So, we will be we will be applying more penalty. So, if you have if you are suffering from over fitting increase the regularization rate. So, sometimes people also combine L1 and L2 regularization together which is called as elastic net regularization. So, those options are also available to you. If you are trying to fight under fitting and if you have regularization added one way of reducing under fitting is by we can reduce lambda to fix under fitting.

So, once you fit once you reduce lambda what will happen is that we apply lesser penalty for model complexity, and that gives us more root or that gives us more capacity to learn the training data, and that is how we can fix the under fitting problem. So, these are the broad ways in which you can fix up your under fitting and over fitting problem and in your training. So, whenever your training machine learning model you will often come you will often come across issues like under fitting and over fitting. And, these are some of the strategies in which you can fix those problems.

(Refer Slide Time: 26:37)

	More Data?	More Complexity?
Underfit	X	Decreasing "more complex"
Overfit	✓	Increase "less complex"



So, we have two levers here. So, we have problems which is under fitting and we have over fitting. So, we have two problems and what levers we have we have we say more data, more complexity. So, we say that more data does not help here, it definitely helps here, more complexity is by decreasing lambda and here we increase lambda and lambda helps us to so, increasing lambda leads to less complexity. And, here we get more complex model more complex in terms of number of parameters.

So, before we close this discussion let us quickly look at some of the evaluation measures. So, let us let us look at more of the classification measures like accuracy, precision, recall and so on. So, the first thing that we do is we build what is called as confusion matrix.

(Refer Slide Time: 29:11)


Confusion matrix

	Predicted	
	+1	-1
Actual	+1 TP	-1 FN
	-1 FP	TN

$$\text{Accuracy} = \frac{TP + TN}{TP + FN + FP + TN}$$

$$\text{Precision} = \frac{TP}{TP + FP}$$

$$\text{Recall} = \frac{TP}{TP + FN}$$

$$\text{F1-Score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$


So, confusion matrix is a key matrix from which we can derive all the matrix. So, this is let us say the actual and this is the predicted side. So, let us say actual is +1 and -1. If, actually is +1 and predicted is +1, we have what is called as true positives. If, actual is -1 predicted is -1, we have what is called as true negatives. If, actual is +1, but predicted is -1, we have what is called as false negative, and if actual is -1, but predicted is +1, it is false positive.

Now, we will use this confusion matrix. So, what you can do is you can essentially apply. So, these measures are mainly calculated for classification. And, once you get a classification you

can find out what are the true, you can first feed up the confusion matrix and then based on that we can calculate measures like accuracy.

So, we can calculate accuracy as so what is accurate which are accurate numbers which is true positives plus true negatives divided by true positives plus false negatives plus false positives plus true negatives. So, essentially it is a ratio of the diagonal element some of the diagonal element divided by all the sales in the matrix.

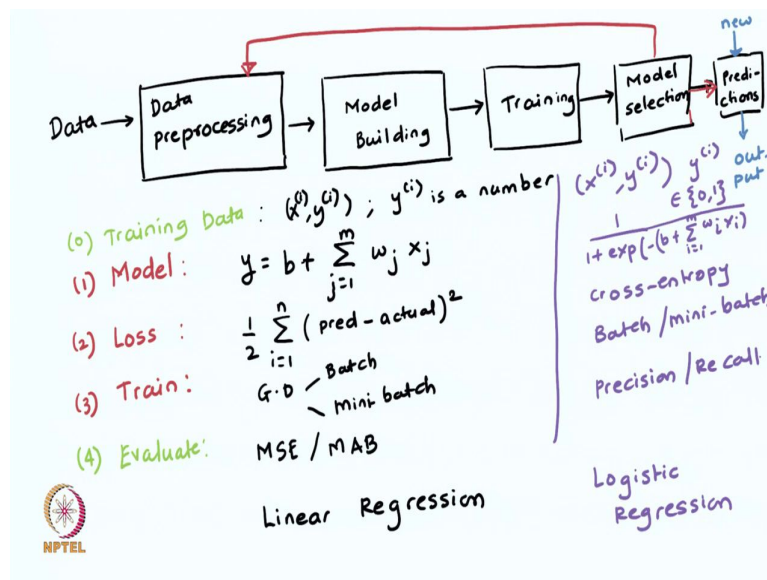
Then we have other measures called so, accuracy does not work on imbalance data set, where there are too many examples of 1 class and 2 few examples of the other class, but accuracy works with on a balance data set if you have roughly equal number of positive and negative example accuracy works in those cases.

So, in case of imbalance data sets or in general we calculate a measure called precision. So, the precision is the ratio of true positives divided by total predicted positives. So, what are the total predicted positives these are true positives plus false positives, these are total predicted positives. So, ratio of the actual positives to the total predicted positives is called precision. Recall on the other hand is the ratio of true positives to the total positives.

So, what are total positives, which is the total actual positives, which is true positives plus false negative. This is recalled and we combine precision and recall into a score called F 1 score, which is the harmonic mean of precision and recall and is calculated as 2 times precision into recall over precision plus recall.

So, these are the 4 measures that we mainly use to evaluate the models on a classification task. So, this brings us to the end of model evaluation discussion and this was the last piece in the machine learning pipeline. So, let us let us summarize or let us revisit the steps in machine learning pipeline.

(Refer Slide Time: 33:11)



So, we have we start with data. Data is the most essential component of machine learning pipeline, we first do data preprocessing. After data preprocessing we build model; model building involves specifying the architecture of the model or the model function itself, then we train the model training, and then we do model selection which you know a selecting model of right complexity. And, finally, we get the model on which we do predictions.

So, we do prediction on the new data and we get the output. Note that in a single go, we may not be able to get the final model. So, what we do is so there might be a loop over here. So, we do training we do model selection, we may not have good enough model, then we go back again we recheck the features see if we can add more features to the model. And, again repeat the process we do this until we are satisfied with the model quality. Once, we get the model of reasonable quality we deploy that model and use it to get predictions on the new data. So, this is the entire machine learning machine learning flow.

So, I would like to make a passing remark about the components in machine learning model. So, this is a very very important few of machine learning model. So, what happens is so if you if anybody tells you that they have invented a new machine learning algorithm you should ask them 3 questions.

So, the first question is what is your model? What is the mathematical function in your model? What is your loss function? How do you calculate the loss and third is how do you train the model? These are 3 core questions you should ask in addition to that you can ask a couple of more questions; one is what is your training data, and how do you evaluate a model, what are the metric that you use for evaluation?

So, if we ask these 5 questions we will be able to understand the machine learning model pretty well ah. So, this is a componentized way of machine learning model. Let us take the complete example and see how it goes. So, in the case of regression, the model that we use is bias plus the linear combination of features and the respective weights.

So, this is a model the loss we use squared loss, which was half we calculate loss across all the points we take the prediction minus the actual value we square it and this is how we get loss. And, then we use gradient descent either batch or mini batch to optimize the loss and find the parameters and we use mean squared error on mean absolute error as evaluation measures. This was for regression.

Let us try to get a similar view for logistic regression, which is a classification algorithm. In the case of logistic regression, we take linear combination and apply sigmoid function on the linear combination that you get is a model, we use cross entropy loss and we can use either batch, or mini batch gradient descent and you can evaluate the accuracy with precision recall.

So, if you come across any new machine learning algorithm. And, if you ask this 5 pointed questions and note down what is really and note down the answers, you will be able to pretty much understand what is really happening in this machine learning algorithm.

We are taking the training data, we are understanding what is the type of machine learning problem, we are solving whether it is a regression problem, or a classification problem, then we are selecting a suitable model, we are defining a loss function and optimization algorithm to find out the optimal parameters for which the loss is minimized. And, finally, we evaluate it on a suitable metric to report the performance on the test data, which we hope is a good measure of performance on the future data.

In this process, getting the features for the data, define loss functions or activities that are tightly linked to the domain. So, hope you understood the machine learning the whole end to end machine learning flow, you had a good understanding of componentized view of machine learning model. And, also steps involved in machine learning pipeline, we will take these learning's neural network in one of the follow up sessions.

After this session, we will go to neural network playground and understand each of this concepts visually, which I think is a very fun activity hope you enjoyed learning this part with us [FL].