Pattern Recognition

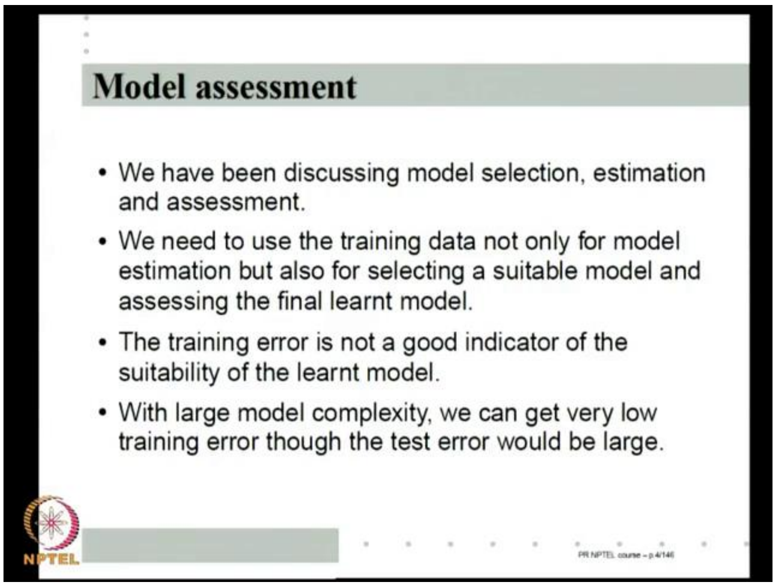
Prof. P. S. Sastry
Department of Electronics and Communication Engineering
Indian Institute of Science, Bangalore

Lecture - 40
Assessing Learnt classifiers; Cross Validation

Hello and welcome to this next lecture in pattern recognition. Last class we have been discussing issues about assessing a classifiers or regression function models in general any model that we have learnt from data. And we looked at some very generic issues of which are which are independent of any learning algorithm for example, no free lunch theorem and so on which tell you that there is no such thing as the best classifier learning algorithm. Essentially, each algorithm if it does well and certain kinds of pattern classification problems, it has to do badly on certain other classification problem. Several algorithm I have shown in this applications and that is nice. We also looked at this so called bias variance trade off in a, getting a good model estimated from data and after that we we were looking at in practise how does one assess (( )) models.

Apart from all those theoretical issues somebody gives you some data, and we have to learn a model from it, a classifier regression function whatever and then we have to ultimately decide by there what we learnt is good or not, what kind of estimate is directly likely to give. As we also seen learning a model involves two steps, a model selection and a model estimation.

(Refer Slide Time: 01:52)



Model assessment

- We have been discussing model selection, estimation and assessment.
- We need to use the training data not only for model estimation but also for selecting a suitable model and assessing the final learnt model.
- The training error is not a good indicator of the suitability of the learnt model.
- With large model complexity, we can get very low training error though the test error would be large.

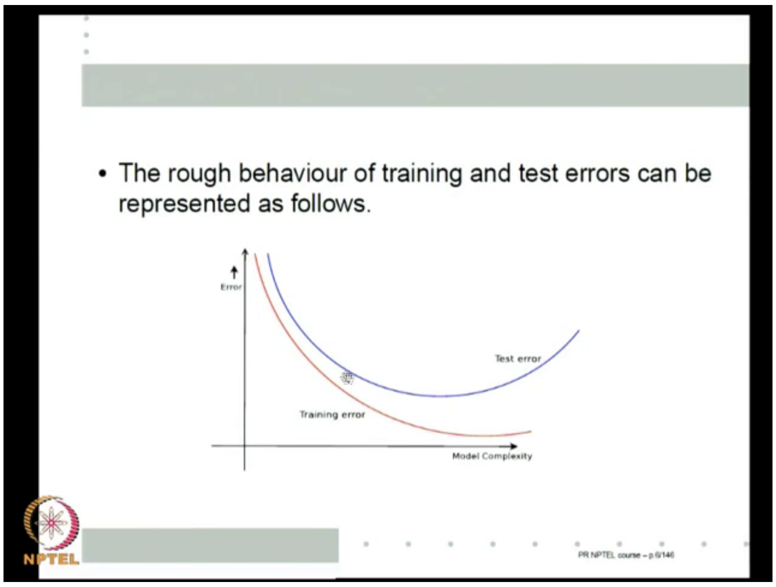
PR NPTEL course – p.4/146

So, essentially we need to use the training data for model estimation, model selection and model assessment. What does that mean? We have to first choose a class of models. A class of models would mean all the parameters of the models should be fixed. So, a neural networks is not a class of models, but neural networks with a single hidden layer with five hidden nodes is a class of models. So, we have to first choose the model parameters. Similarly, a Gaussian kernel with sigma is equal to 10 is a specific model class.

To select a suitable model class and then within that model class we have to learn a specific models. That is what we have, all the algorithms we have been considering so far are. Given some training data, how do we learn a particular model from the class of models that we have decided to learn from. And then we also have to assess how good the final learnt model will perform. So, the idea of model assessment is that we need to read the training data, not only for model estimation meaning all the learning algorithm that we have been discussing throughout the course, but also for selecting suitable model class and also for estimating the true risk of the final error that the learnt model is likely to have.

And the reason why this is an interesting or difficult problem is that the training error is not a good indicator of the suitability of the learnt model. Obviously, because all are algorithms are designed to minimize the training error. So, whatever you do a single (( )) or noise is there in our training data, our algorithms try to tune for that and hence we may get a very low training error, but that does not necessarily mean we will do well on unseen patterns. So, essentially when the model complexity is large it is possible that we can get very low training error, but the test error, that is error on new unseen data would be large. So, this is the basic issue.

(Refer Slide Time: 04:00)



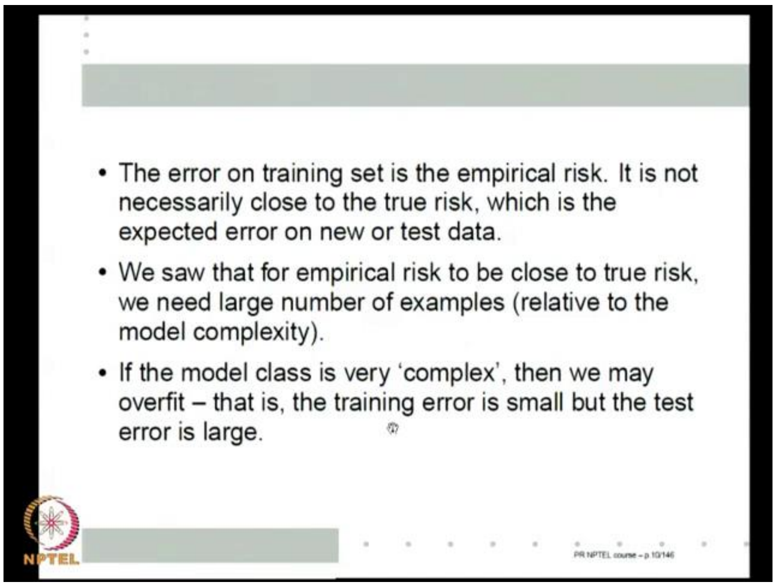
- The rough behaviour of training and test errors can be represented as follows.

We actually seen last class, but this graph is worth seeing many time. This is a very important graph for models assessment. Essentially the training and test errors behave like this for a for a given number of examples as I am increasing the model complexity. Model complexity is along x axis and error is on the y axis. So, the red is the training error. So, as I am increasing the model complexity at the same number of examples because more and more complicated models are there. So, I can certainly fit some model or the other better to the data.

So, my training error keeps coming down, but my test error initially when model complex is very small because of very high bias I get high test error, but if I make my model complexity very large. Once again because of high variance now I get high test error. So, somewhere in the middle I have the best bias variance trade off and that is what I need to get it. So for example, if you think of this as a least squares fitting of a polynomial and model complexity will be the degree of the polynomial.

As we know as we keep increasing the degree of the polynomial for a given number of examples, I have, I can reduce my training error as as small as I want. I eventually make it 0, but if I have 100 data samples and I fit a 100 degree polynomial, obviously even though my training data is 0 I am I am not going to say that this particular polynomial is what the data is generated from. So, the the issue is that we need to select the model with a right complexity. Of course, we do not know which is the right complexity model, but certainly just going by training error does not help us.

(Refer Slide Time: 05:56)



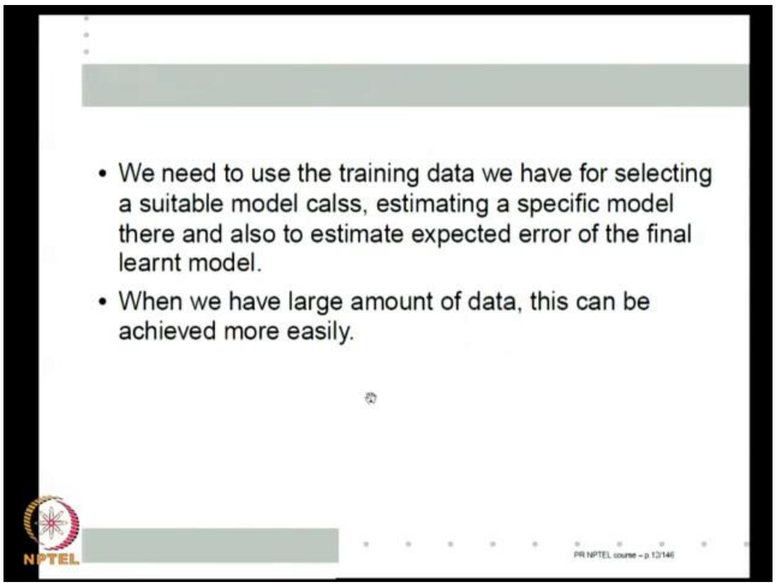
- The error on training set is the empirical risk. It is not necessarily close to the true risk, which is the expected error on new or test data.
- We saw that for empirical risk to be close to true risk, we need large number of examples (relative to the model complexity).
- If the model class is very 'complex', then we may overfit – that is, the training error is small but the test error is large.

PR NPTEL course – p.10/146

So, this we knew when we studied from our discussion on statistical learning theory. The error and training set is the empirical risk and it is not necessarily close to the true risk which is the expected error on new or test data. We saw that empirical risk will be close to true risk, only when you have large number of examples, large is relative to the model complexity, the v c dimension. So, otherwise and in general at any given number of examples we increase the complexity, the examples will become small relative to the complexity.

If the model class has very, is very complex then we will over fit, over fitting means we are essentially getting low training error, but large test error. It is like we have, it is like we have, it is like we have fitting a 100 degree polynomial to 100 points. It is it is over fitting because we get 0 training error, but it is not certainly a very good fit. So, this kind of over fitting is what we want to avoid.

(Refer Slide Time: 07:08)

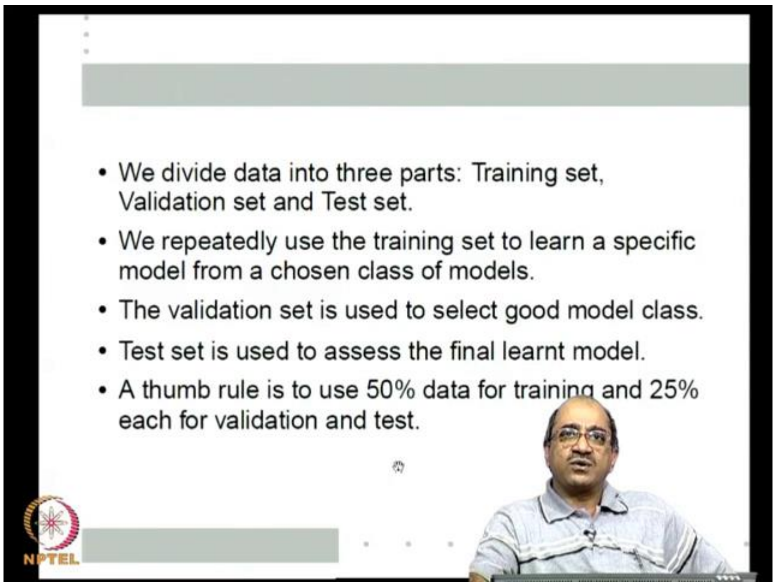


- We need to use the training data we have for selecting a suitable model calss, estimating a specific model there and also to estimate expected error of the final learnt model.
- When we have large amount of data, this can be achieved more easily.

PR NPTEL course – p.13/146

We need to use the training data for selecting a suitable model class. There is a model selection, estimating or learning a specific model from that class and also to get the final learnt model. As we briefly mentioned last time, if we have sufficient data this can be done very easily. If data is no problem then all this all this is not really problematic. What is the overall plan?

(Refer Slide Time: 07:35)



- We divide data into three parts: Training set, Validation set and Test set.
- We repeatedly use the training set to learn a specific model from a chosen class of models.
- The validation set is used to select good model class.
- Test set is used to assess the final learnt model.
- A thumb rule is to use 50% data for training and 25% each for validation and test.

If we have n of data then whatever data we have we split into three parts. A training set, a validation set and a test set. The training set is what we use like the training sets in all our algorithm, but we repeatedly use the training set to learn a specific model from a chosen class of models. So, if I chosen a neural networks with 1 hidden layer, 5 hidden nodes then we use the training set and a (( )) algorithm to learn the weights or if I chosen a SVM with a polynomial kernel with p is equal to 3, then for that model class we can of course, use and and a particular value of c then we will we will we will then use the SMO algorithm for example, with the training set to learn a specific SVM.

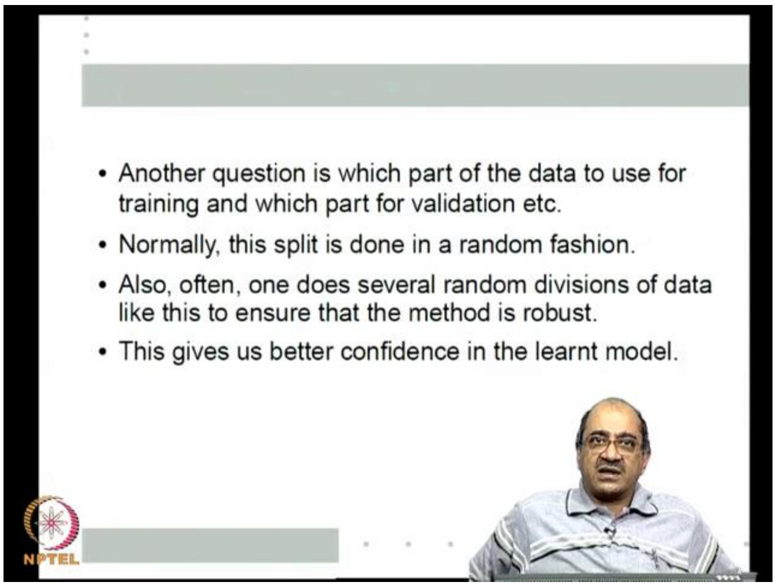
So, for a specific model class we use our training set to learn a specific model. What do you mean by repeatedly? What we are doing is we are using validation set to select good models. So, let us say I want to know what is a good c to use in my SVM algorithm. So, I will fix some value of c, use the training set to get my SVM, then I measure the error of that SVM on the validation set. Now, I choose some other value of c, once again use the training set to learn an SVM, measure the error of this SVM also using the validation set.

So, I will use the validation set to compare different models. That is how you do model selection. And finally, after doing a particular model, after fixing a value of c and other parameters of the kernel in the SVM, I will learn my final SVM and to get a idea what it, what its error rate is likely to be I use the test set. So, essentially a test set is not used till I come to the final classifier as I say test set should be locked up in a volt. So, then because this is certainly new random data which are not seen in anything so far, it is likely to give me the, a good estimate of the true risk.

Say for example, if I use the validation set to assess the utility of the final learnt model, it would not be good because I chose that particular model class only because on the validation set that particular model class is giving me no error. So, I am likely to think that what I learnt is very good even though it may not be that good. So, that is the reason indeed I also separate test set. So, basically if I have n of data, I repeatedly use the training set to learn specific model from a chosen model class. I compare different model classes from all selection using validation set.

And finally, I use the test set to find the expected error of the final learnt model. So, validation set is used to find the error rate of the classifier and different model classes may be index by a parameter alpha. Alpha could be for example, c of SVM. So, and then after choosing a particular model class and finally learning it, I use the test set to find its final error. A rough rule of thumb is to used about 50 percent data for training, 25 percent data for validation, 25 percent data for test. So, for 1000 samples use 500 for training, 250 for validation, 250 for test.

(Refer Slide Time: 11:02)

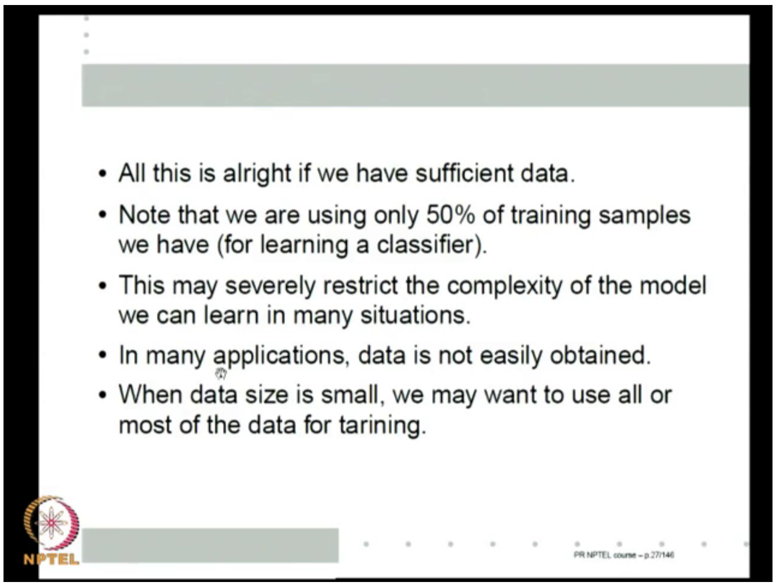


- Another question is which part of the data to use for training and which part for validation etc.
- Normally, this split is done in a random fashion.
- Also, often, one does several random divisions of data like this to ensure that the method is robust.
- This gives us better confidence in the learnt model.

What part to use for validation, which 250 for validation, which 250 for test, which 500 for training? Well, normally because you do not want any biases creeping in we will do this in a random fashion. You split the data randomly into three parts with this, in this proportion and then use one for training, one for validation, one for test. Very often one goes through this several times, several random divisions like this and do this just to see that the final classifier learnt does not change too much.

If the final classifier learnt does not change too much that means overall the learning is quite robust and that will also give me sufficient confidence that what I have learnt is correct. So, instead just doing one split of training validation test, one does many random splits of data like this and on each of them you do the same process. And compare the final learnt models to see if their vastly different from one another. If they are then maybe I am not learning well, if they are sufficiently robust then its fine. So, essentially if we have sufficient data this is what we can do.

(Refer Slide Time: 12:07)


- All this is alright if we have sufficient data.
- Note that we are using only 50% of training samples we have (for learning a classifier).
- This may severely restrict the complexity of the model we can learn in many situations.
- In many applications, data is not easily obtained.
- When data size is small, we may want to use all or most of the data for tarining.

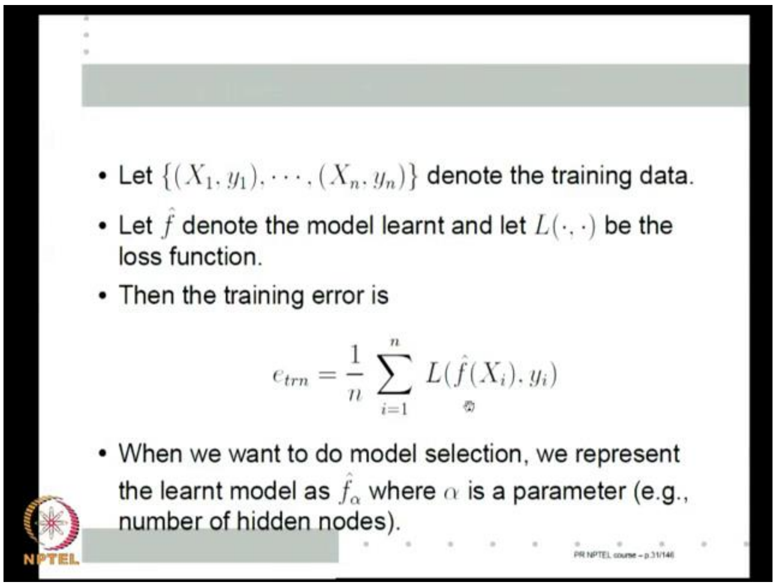
PR NPTEL course – p.27/146

What do you mean by sufficient data? See, what we have just now advocated is that 50 percent of the data should be set aside for validation and testing which means when I use my SVM algorithm or back propagation algorithm or linear least squares algorithm what have you, I am only using 50 percent of the data that means we are using only 50 percent in the training samples to actually learn a classifier. This can restrict the complexity of the model we can learn. So, if I have 1000 samples I may be able to learn neural networks with 20 hidden nodes, but because I have to learn only from 500 samples I can learn with only 10 hidden nodes for example.

Of course, things are not so linear, but in general when the number of training samples is less it restricts the complexity of model we can learn from. This does not matter if I can get training samples at will. In synthetic problems I can get as many training samples as I want because training samples are simply simulated. In some some kinds of problems for example, if I want web pages, then possibly I can think I can get as many web pages as I want by calling on the, on the net if web pages are my pattern. But in many many applications data is not easily obtained. So, obtaining data is a costly process and then finding a, in the training data every data should have its true classifier, true quote and unquote classification label should be there because you have to give data x i and y i.

So, I have to obtain the data x i and then classify it. Both are costly, obtaining data could be costly, also correctly classifying it could be costly. So, when data size is small we are we are we are rather thrifty and we do not want to permanently set aside the data for other purposes. We may want to use all the data for training. Only then we can learn a model of sufficient complexity. So, when data size is small doing what we advocated, having 50 percent for training, 25 percent for validation, 25 percent for test may not be feasible.

(Refer Slide Time: 14:31)



- Let $\{(X_1, y_1), \cdots, (X_n, y_n)\}$ denote the training data.
- Let $\hat{f}$ denote the model learnt and let $L(\cdot, \cdot)$ be the loss function.
- Then the training error is

$$e_{trn} = \frac{1}{n} \sum_{i=1}^{n} L(\hat{f}(X_i), y_i)$$

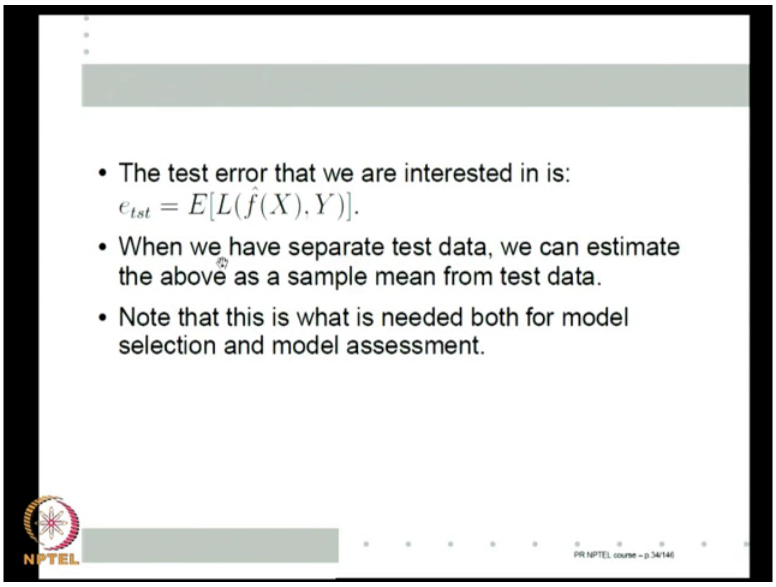
- When we want to do model selection, we represent the learnt model as $\hat{f}_\alpha$ where α is a parameter (e.g., number of hidden nodes).

PR NPTEL course – p.31/146

Lets us just formulize all this so that we know what we are talking about. Now, as earlier let us say X 1 y 1 X n y n is the training data. Let us say f hat is the model that we learnt and L is the usual loss function. Then what is the training error? Training error I will call it e trained is 1 by n summation i is equal to 1 to n L of f hat X i y i. This is your empirical risk; this is the training error we have been considering all learning algorithms. And when we want to do model selection obviously there will be, we have to also index different model classes.

So, we will represent it as f hat alpha where alpha is a parameter that denotes the model class. So, for example, number of hidden nodes or the value of c or whatever. So, when we when we are doing model selection and hence have to compare different model classes. This f hat may have a further subscript alpha. To test, test error that we are interested in is the true risk. I call it e test which is actual expectation of the loss function. This this is the true risk that we are interested in.

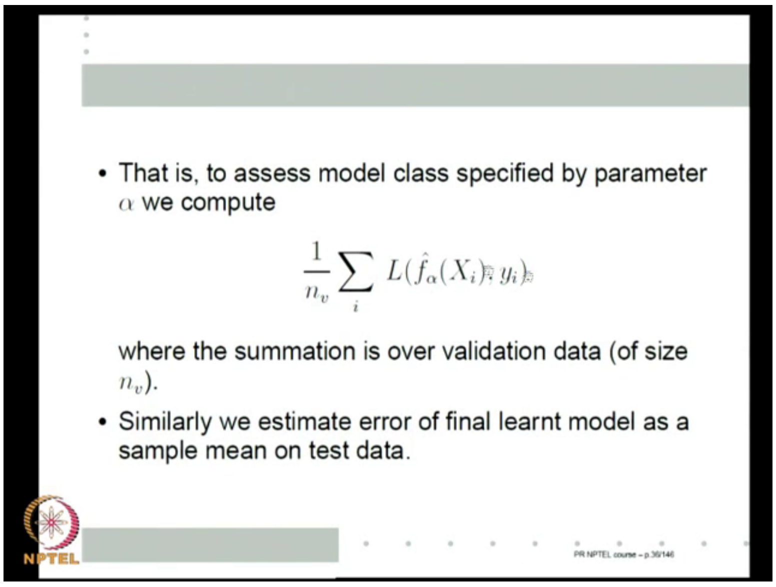
(Refer Slide Time: 15:36)


- The test error that we are interested in is: $e_{tst} = E[L(\hat{f}(X), Y)]$.
- When we have separate test data, we can estimate the above as a sample mean from test data.
- Note that this is what is needed both for model selection and model assessment.

PR NPTEL course – p.34/146

When we have separate test data we can easily estimate it as the sample mean form the test data. This is an expectation, if I separate test data I will simply estimate it as L f hat X i Y i is summation over i or 1 by the number of samples where the summation is over all the test data. Now, whether it is test or validation it is the same. In both cases I want to estimate this. One for different alpha, one for the final f hat, but essentially you want estimate this expectation as a sample mean on separate data.

(Refer Slide Time: 16:18)


- That is, to assess model class specified by parameter α we compute

$$\frac{1}{n_v} \sum_i L(\hat{f}_\alpha(X_i), y_i)$$

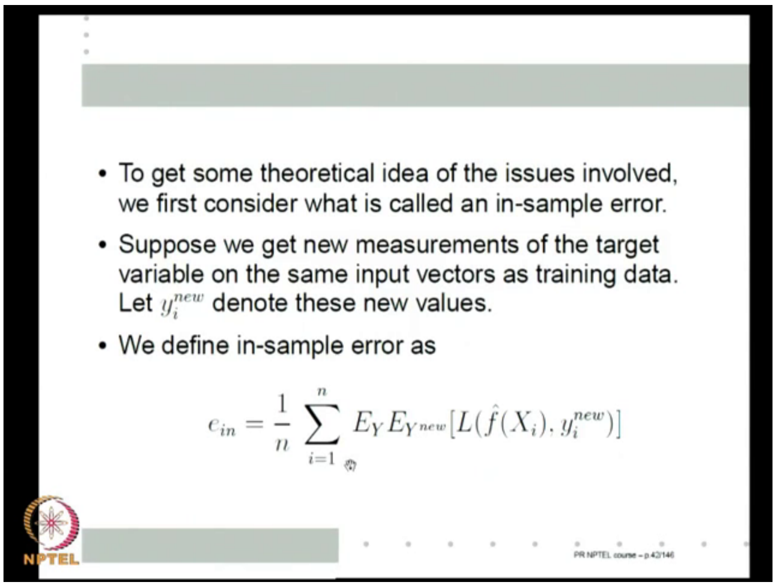
where the summation is over validation data (of size n_v).
- Similarly we estimate error of final learnt model as a sample mean on test data.

PR NPTEL course – p.35/146

That is what I am saying is suppose we want to assess some model class which is specified by parameter alpha as I said alpha could be value of c in SVM or number of hidden nodes in neural network and so on. So, if particular alpha we choose that model class, use our training data set to learn the right model. Let us say that is f hat alpha, then to asses that model class on the validation set I will do L f of hat alpha X i y i summed over i. This summation is not over your training data, but over the validation data.

So, if there are n v validation data points then it will be 1 by n v of the summation. Similarly, if I am doing test error, I do the same thing on the test. So, essentially the the the actual test error is a true expectation which I can certainly use approximate as a sample mean if I have a separate data. That is what we have been saying. Only if I do not have separate data then here in the training (( )) test error is not right because the f hat is specifically chosen to make the training error small. So, essentially I can do this, I can estimate the true risk as a sample mean like that If I have sufficient data.

(Refer Slide Time: 17:29)



- To get some theoretical idea of the issues involved, we first consider what is called an in-sample error.
- Suppose we get new measurements of the target variable on the same input vectors as training data. Let y_i^{new} denote these new values.
- We define in-sample error as

$$e_{in} = \frac{1}{n}\sum_{i=1}^{n} E_Y E_{Y^{new}}[L(\hat{f}(X_i), y_i^{new})]$$

PR NPTEL course – p.43/146

Then we can do model assessment, model selection everything like this. When the amount of data is small my problem is keeping it, keeping a part of it aside permanently for validation and testing. I do not want to do that. What can I do? So, we need some methods which can estimate the test error without having to permanently lock up data, part of the data. I want to use all data for learning, but at the same time I somehow want to estimate these test errors more accurately. So, let us first start by looking at what is called an in sample error. This gives you some theoretical ideas of what the problem is about.

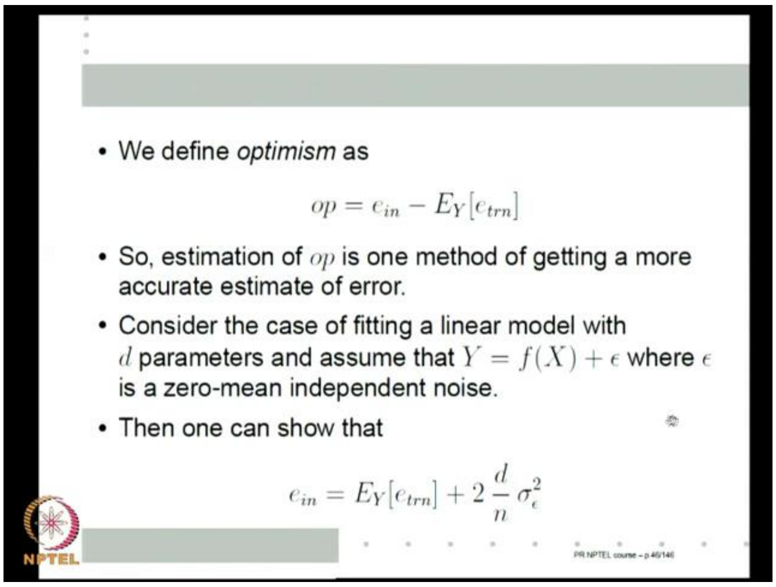
The in sample error is the following. Say we have a training data X i y i. So, let us assume that for this same X i we get one more measurement of y i. Normally, lot of times is the y i noise, that is really the noise in the training data. So, let us say, theoretically lets have (( )) pattern recognition (( )), what it means is that take the same X i go to some other expert and ask him to classify or if I am actually measuring some value of some function experimentally I set my data to X i again and re measure y i and so on. Let us call it y i new.

So, on the same X i we get new y i new and so this is a kind of in sample training data. X i's are same, but y i's are once again obtained randomly using the same underlined distribution. And then we define what is called the in sample error which is error f hat X i y i, y i new; this I take expectation with respect to the y new random variable with respect to which these y i news are obtained. See X i's are fixed. So, there is nothing random in X i.

That is why I am averaging over X i, but X i's are fixed, but y i new is random. So, I take an expectation over y new and also to take expectation over y because the f hat that I have learnt is a function of the old y i's. I have learnt f hat using x i y i. So, f hat is the function of the old y i's. So, this expectation averages over the f hat I could have learnt if I got some random data and then having learnt from there, this y this expectation tells me how I would have performed if the same x i, but different measurement noise was involved.

Very often if we think the the prediction variable is related to our our measurement vector or feature vector x by a deterministic function f x plus noise, then a good way of assessing how well we learn is to just ask at the same axis if I take a new measurement how well my prediction, predicting new measurement. This will tell me whether I am predicting the the the the deterministic part or unnecessarily modelling the stochastic variation in the original y i using which I have learnt f hat. That is why we want to define the in sample error both as the expectation with respect to Y new and expectation with respect to y.

(Refer Slide Time: 20:59)



- We define *optimism* as

$$op = e_{in} - E_Y[e_{trn}]$$

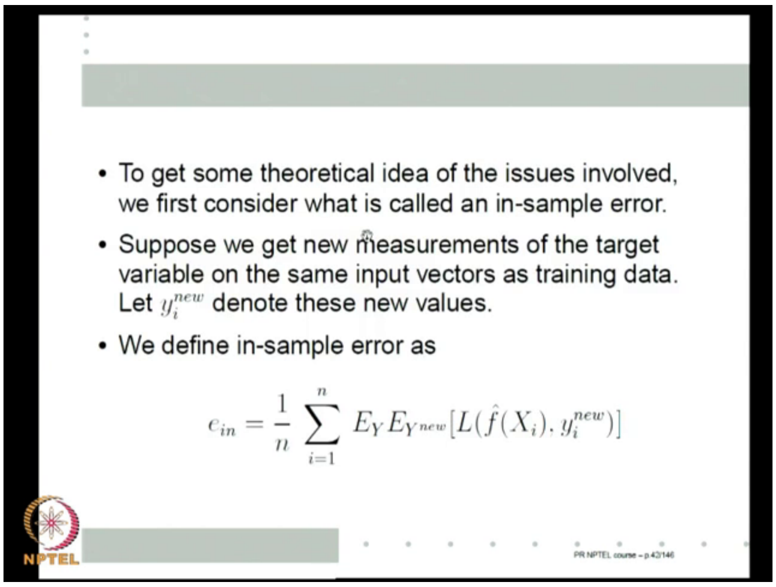
- So, estimation of op is one method of getting a more accurate estimate of error.
- Consider the case of fitting a linear model with d parameters and assume that $Y = f(X) + \epsilon$ where ϵ is a zero-mean independent noise.
- Then one can show that

$$e_{in} = E_Y[e_{trn}] + 2\frac{d}{n}\sigma_\epsilon^2$$

PR NPTEL course – p.46/146

Now, we define what is a (( )) optimism as the in sample error minus the training error.

(Refer Slide Time: 21:09)



- To get some theoretical idea of the issues involved, we first consider what is called an in-sample error.
- Suppose we get new measurements of the target variable on the same input vectors as training data. Let y_i^{new} denote these new values.
- We define in-sample error as

$$e_{in} = \frac{1}{n}\sum_{i=1}^{n} E_Y E_{Y^{new}}[L(\hat{f}(X_i), y_i^{new})]$$

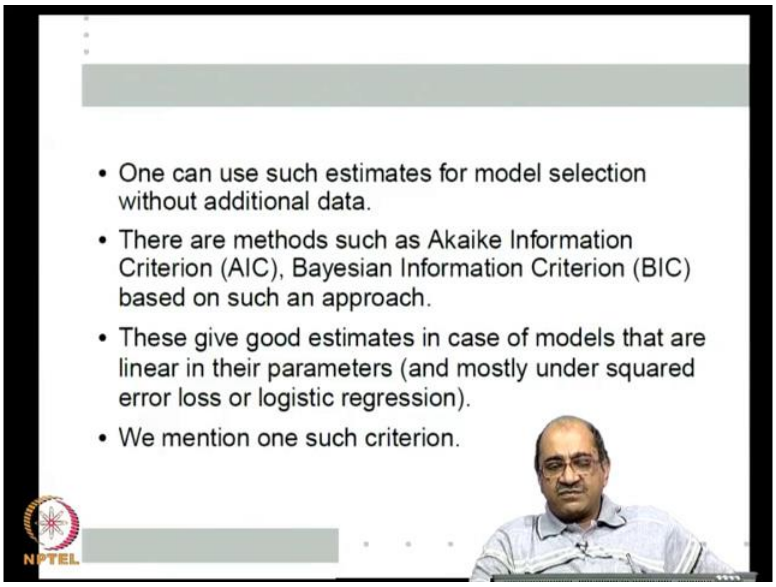
PR NPTEL course – p.43/146

See, essentially the in sample error would be higher than the training error. Training error chooses an f hat, we choose an f hat to minimize the training error. The same f hat and new measurements will not do very well, will not, will not do as well may do okay, but will not do just as well. And because this is averaged over all first measurements one expects in sample error to be higher than the training error. Now, essentially if it is small then it is all right. So, we call this is the optimism in sample error minus the training error.

So, if I can somehow estimate this o p then I can get at least a connection with the training error and in sample error in as much as in sample error is averaged over all possible measurement noise. If I can estimate that for my model or that itself gives me some some good sense of how well my model will perform in general. So, estimating the optimism is a method of getting more accurate error estimates. If we think of fitting a linear model with d parameters and we assume that the relationship between Y and X is f is equal to, Y is equal to f of X plus epsilon where epsilon is a 0 mean independent noise.

This is the standard linear least squares statistical model. Then one can show that the in sample error is the expectation of the training error plus 2 into d by n times sigma square epsilon. d is the number of parameters in the model, n is the number of data, sigma square epsilon is the variance of the noise. Now, it is a 0 mean. So, this this can be shown for for linear model learning. So, this gives us some some interesting ways of doing model selection.

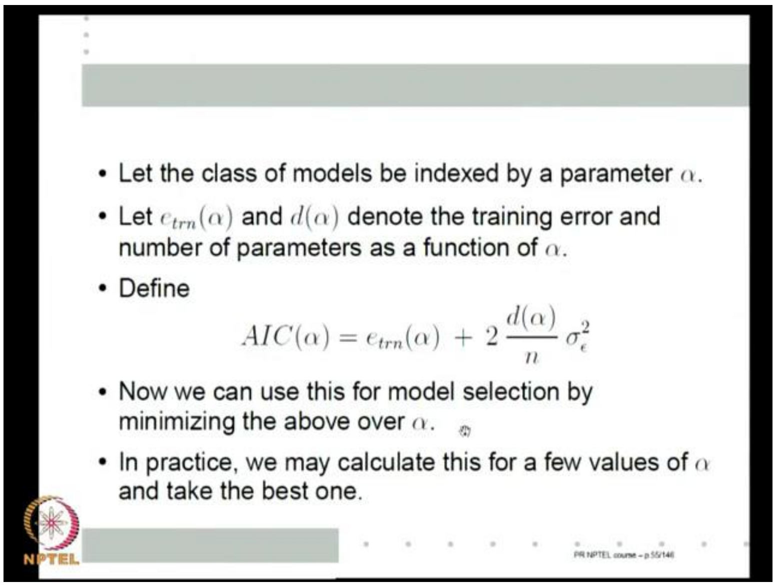
(Refer Slide Time: 23:05)


- One can use such estimates for model selection without additional data.
- There are methods such as Akaike Information Criterion (AIC), Bayesian Information Criterion (BIC) based on such an approach.
- These give good estimates in case of models that are linear in their parameters (and mostly under squared error loss or logistic regression).
- We mention one such criterion.

So, we can use estimates like this for doing model selection without needing separate validation data. There are many criteria, they are called Akaike information criterion, Bayesian information criterion so on which are, which essentially use such calculations to do a model selection without needing a validation data for model. This gives good estimates in case of models that are essentially linear in parameters, most of this work for linear models.

As we seen already linear models are essentially linear in parameters. So, the and we, as we saw when we learnt linear models the most often used loss functions or either the logistic loss or the squared error loss. So, all these estimates are for linear models and a squared error loss are for logistic regression. So, we we will just look at one one example of how one does that kind of model selection.

(Refer Slide Time: 24:08)



- Let the class of models be indexed by a parameter α.
- Let $e_{trn}(\alpha)$ and $d(\alpha)$ denote the training error and number of parameters as a function of α.
- Define

$$AIC(\alpha) = e_{trn}(\alpha) + 2\,\frac{d(\alpha)}{n}\,\sigma_\epsilon^2$$

- Now we can use this for model selection by minimizing the above over α.
- In practice, we may calculate this for a few values of α and take the best one.

PR NPTEL course – p.55/146

So, let us say I have a model class is indexed by parameter alpha. So, I want to learn the right value of alpha to select a model class when we are in model selection. So, what as I said alpha could be number of hidden nodes, alpha could be c value. As a matter of fact alpha could be not just as one parameter, it could be a vector parameter. So, alpha could consists of the c value as well as the parameters of the kernel function in an SVM. So, alpha could be a pair, an ordered pair c comma let us say p. p is the degree of the polynomial kernel. So, different values of alpha specify different model classes and I want to know which model class to choose.

In, within each model class I can find the training error because I have a training set. So, let us say e trained alpha, e trained is a function of alpha is the training error of the model I have learnt in the model class alpha. And let us say d of alpha denote the number of parameters to specify as model in in the class alpha. Because we are considering only linear models essentially as for linear class alpha is not like c f and SVM, for linear class alpha is simply equal to the the the number of parameters you use, the step size you use in the algorithm and things like that.

So, let us say d alpha denotes the the number of parameters in the model class alpha. Then we define AIC alpha as e trained alpha. This is the training error plus 2 d alpha by n sigma square epsilon. So, essentially we are saying this is a good model complexity term. As we already know the true error or true risk is empirical risk plus some model complexity term. In as much as my optimism estimate is given by this. We are saying in sample error is e trained plus this.

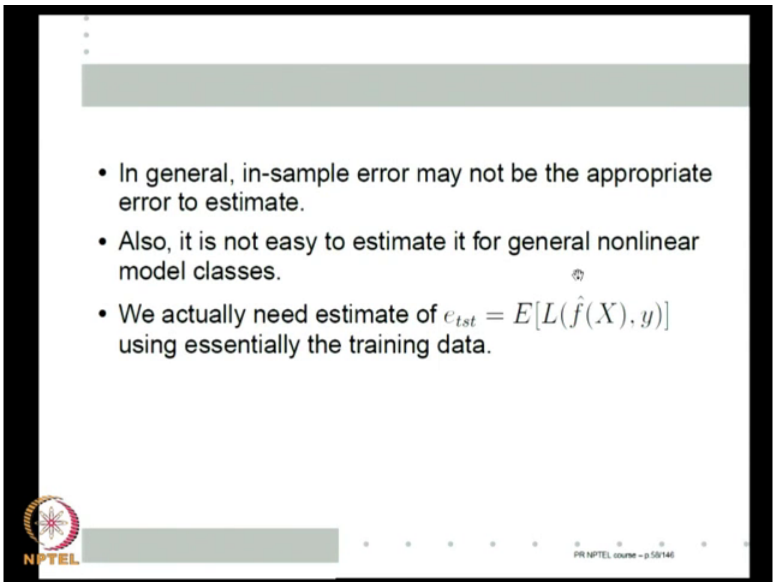
So, optimism estimate is this. So, this is a good estimate of the model complexity. So, we define AIC alpha as trained trained alpha plus 2 time d alpha by n into sigma square alpha. Of course, nobody gives you sigma square epsilon, either you assume some nice variance model or you make may be a few multiple measurements and estimate as a nice variance. To make multiple measurements I do not have to measure Y i for each of the X i is as much as I assume noise is independent at 1 X i if I measured many Y i many times and find the variance.

That variance I can use here. So, the idea of this is that now I I minimize AIC alpha rather than e trained. Normally, when I am doing a learning algorithm I am only minimizing training error. Now, instead of minimizing training error I minimize this complicated thing. I want to, so it is like I am not choosing a particular model. So, I am, I minimize this over alpha. Of course, in general alpha is a discrete parameter. So, one cannot minimize this.

So, but what it means is I choose different values of alpha and then on each alpha I find the training error and then calculate this. So, the the sum is the true reflection of that model class. Just because training error is small does not mean that the model class is good because for example, d alpha may be large. That is what we have seen earlier, how the model complexity comes. So, these are, these are criteria for properly accessing the model complexity.

So, I can use this compound thing for choosing a model. I I can evaluate this for a few alpha and choose the best alpha. What have I gained compared to what I am doing earlier? I do not need a validation set now. As I do not need a separate validation error I am using this estimate of optimism to to decide the model complexity term. So, I can use all the data I have for training as I do not need extra validation error. So, this this is how for example, I can do model selection. This is one, this is Akaike information criteria. This is one way of doing model selection.

(Refer Slide Time: 28:38)

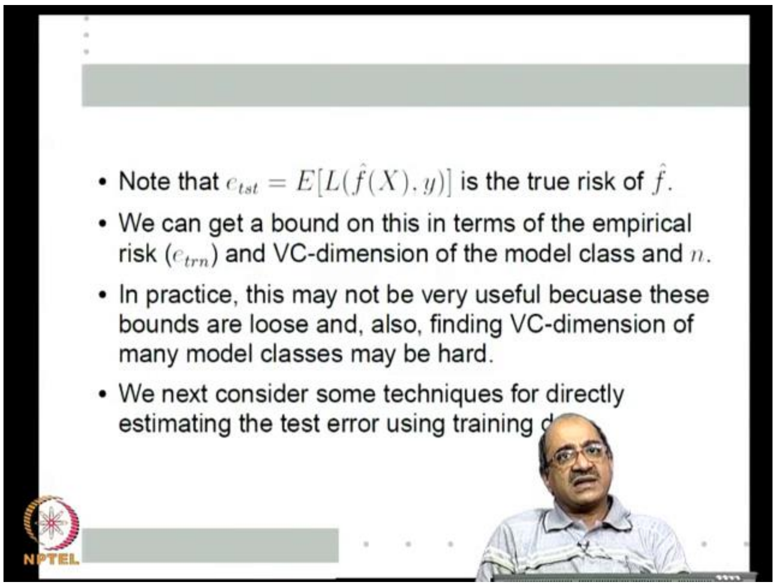


- In general, in-sample error may not be the appropriate error to estimate.
- Also, it is not easy to estimate it for general nonlinear model classes.
- We actually need estimate of $e_{tst} = E[L(\hat{f}(X), y)]$ using essentially the training data.

PR NPTEL course – p.58/146

Of course, in general in sample error may not be the appropriate error to estimate because that is the error that my learnt model will incur if I have the same X i, but do re measure or measurement for Y i. Apart from that we may not be able to get this kind of estimates in general. That particular estimate we have given is for linear models under squared error or logistic regression kind of loss function. So, in general getting these estimates is also not easy for non linear models. So, what we actually need is some way to reliably estimate the test error, the rue risk of a given model using the same thing at the training data. Of course, I use the training data to learn f hat, but somehow I want to get the true risk of f hat, somehow from the training data itself that is what we want to do.

(Refer Slide Time: 29:36)



- Note that $e_{tst} = E[L(\hat{f}(X), y)]$ is the true risk of $\hat{f}$.
- We can get a bound on this in terms of the empirical risk (e_{trn}) and VC-dimension of the model class and n.
- In practice, this may not be very useful becuase these bounds are loose and, also, finding VC-dimension of many model classes may be hard.
- We next consider some techniques for directly estimating the test error using training d

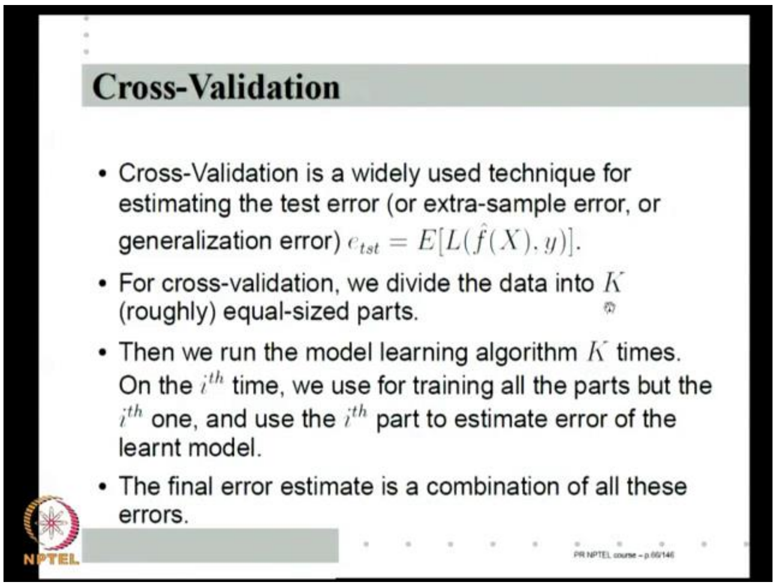
Now, from our discussion on statistical learning theory there are a few other possibilities one can think of doing this. For example, the test error is the true risk of f hat and in the VC theory we seen that the true risk of any model can be bounded above by an empirical risk and the term. And that term involves only VC dimension n and the training error. So, given the training error I can certainly get a bound on the true risk of any model. So, if I have learnt f hat from some training data and I know the training error f hat then knowing the VC dimension of that class, that model class, the number of examples and the training error I can get a bound on e test.

We have we have we have seen such expressions when we did our, when we discussed statistical learning theory. In practice these bounds may not be very useful for two reasons. As we said when we derived these bounds that these bounds are very loose. They have often used the so called union bound which essentially bounds the probability of union of sets by the some of the sets. That itself is loose bound. There are many other steps like that in the derivation because of which the bounds are loose.

The second reason, second problem with these bounds is we need to know the VC dimension of the model class. If you recall we had almost more than half a class of derivation to get the VC dimension of linear classifiers in (( )). So, in general when we have a specific model class, finding the VC dimension may not be that that easy. For example, for given neural network models one has only an estimate on VC dimension, an upper bound on VC dimension.

So, if you are using an upper bound on VC dimension and then an upper bound on the test both of which are loose, ultimately the bound we get on the test error may be loose. So, by loose what it means is that the true test error may be much smaller than the bound we get, but we unnecessarily think that our test error is very high. So, in practice this may not be very useful. So, what we need are techniques, statistical empirical techniques that allow me to estimate the true risk of the test error using some of the training data.

(Refer Slide Time: 32:22)



Cross-Validation

- Cross-Validation is a widely used technique for estimating the test error (or extra-sample error, or generalization error) $e_{tst} = E[L(\hat{f}(X), y)]$.
- For cross-validation, we divide the data into K (roughly) equal-sized parts.
- Then we run the model learning algorithm K times. On the i^{th} time, we use for training all the parts but the i^{th} one, and use the i^{th} part to estimate error of the learnt model.
- The final error estimate is a combination of all these errors.

PR NPTEL course – p.66/146

So, we will, we will start with one of the most popularly used techniques which is called cross validation. Cross validation is a very widely used statistical technique for estimating test error. In statistical (( )) the test error is also called generalization error, extra sample error, extra sample error to contrast it within sample error. In in sample we are using the same X i. The extra sample error is an estimate. So, essentially this expectation is like saying from the underlined distribution get new X and Y, many new X and Y and on that sample estimate the error.

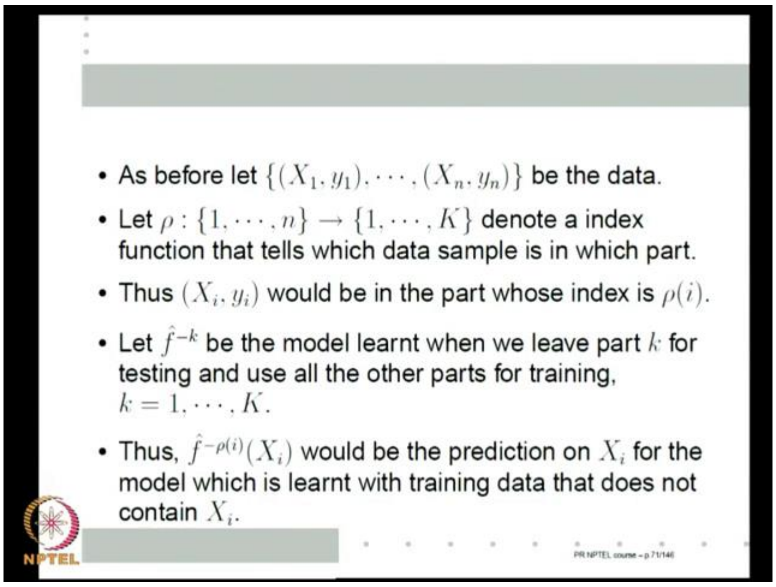
So, that is why it is called extra sample error, generalization error and test error, all these are same, essentially the true risk of f hat. So, cross validation is a technique for estimating the test error without separately setting aside some data for testing the validation. What does class validation do? Given the data we choose some parameter K, we will come back later to say how to choose K. K is some integer and we divide the data, the data of n points into K parts, roughly equal size, if you can cut them to be exactly equal size you will get exactly equal size.

Otherwise, may be one sample there will be, one data point this way that way. So, we divide the data into K parts. K is a parameter of the method. Once, we choose a K we will discuss that later, but so if if I want to do five parts, if I am, I may have 200 samples. So, I put 40 samples in each part. So, I make five parts. I I divide the data into five parts. Then what do I do? Then I run the model learning algorithm K times. So, if I, if I have divided my data into five parts I am going to learn f hat five times and the i th time we use for training all parts per the i th one and essentially use the i th part to estimate error of that learnt model.

So, if I have divided my data into five parts, the idea is I run the same learning algorithm on the, like this. First time I set aside my first part, use parts two, three, four, five for training. Then I will get some model. Then next time I use parts one, three, four, five for training, set aside two for testing for the second time. The third time I use parts one, two, four, five for training and set aside the third part for testing and so on. So, we run the model learning algorithm K times. On the i th time we use the, we use all parts, but the i th part that is one, two up to i minus 1 and i plus 1 to K for training and ultimately going to use the i th part for testing that model.

And my final error estimate is a combination of all these. So, the idea is that I make permanently setting aside any part of the data for testing, testing and validation. Essentially, using all all data both with training as well as testing, is an interesting finish of dividing like this and doing sequentially. Let us first set up some notation to actually write the cross validation estimate before talking about it.

(Refer Slide Time: 35:41)



- As before let $\{(X_1, y_1), \cdots, (X_n, y_n)\}$ be the data.
- Let $\rho : \{1, \cdots, n\} \rightarrow \{1, \cdots, K\}$ denote a index function that tells which data sample is in which part.
- Thus (X_i, y_i) would be in the part whose index is $\rho(i)$.
- Let $\hat{f}^{-k}$ be the model learnt when we leave part k for testing and use all the other parts for training, $k = 1, \cdots, K$.
- Thus, $\hat{f}^{-\rho(i)}(X_i)$ would be the prediction on X_i for the model which is learnt with training data that does not contain X_i.

PR NPTEL course – p.71/146

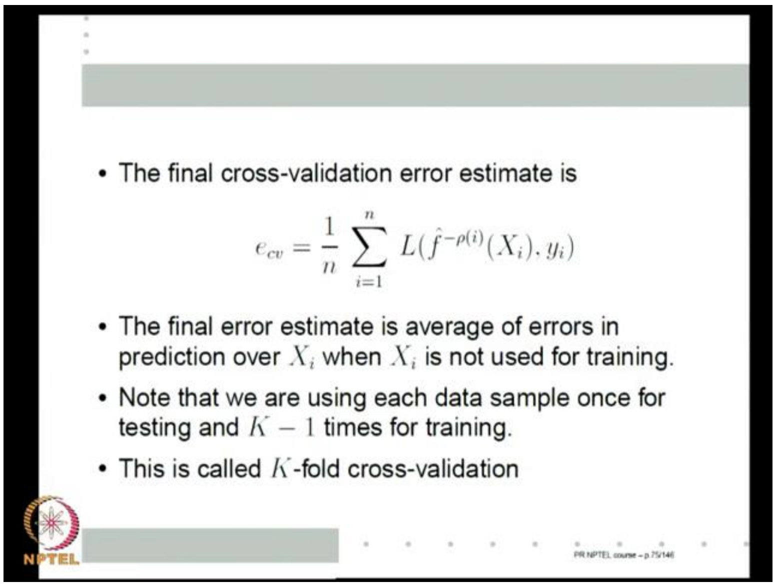
So, as earlier let us say X 1 y 1, X 2 y 2, X n y n with the data points. We are going to talk about these various parts. So, let us say we have, we are dividing the data into 1 to K parts, data is indexed by 1 to n. So, the division is represented by a function rho. So, rho is a function that maps the data end this is 1 to n to part and this is 1 to K. So, rho tells you which data sample is in which part. What about mean? So, X i y i the data index is i, so it will go into the part rho of i. That is how the rho is defined.

Rho is an index function. So, if I want to know which part X i y i would be, which of the K parts x i would be, I will just take rho of i and that part X i would be. That is the definition of rho. Now, f hat of minus K will be the model learnt when we leave a part K for testing and use all the other parts for training. So, f hat and minus 1 you will learn a use part two, three up to K for training, f hat of minus 2 is the model I learn when I use parts one, three, four, five so on for training and so on.

So, f hat of minus K is the model learnt when we leave part K. So, which means what is f hat of minus rho i? Rho i is the index which specifies which part X i X i was in. So, f hat of minus rho i is the model learnt using a training data set which does not contain X. So, if I say f hat of minus rho of x i, it is the prediction of the model learnt, the prediction on X i for a model for whose, which is learnt using training data that does not contain X i. f hat of minus rho i is a model learnt over a training data set that does not contain X i.

I need this notation because my final error estimate is for each X i I find its error on a model which is learnt without using it. Each X i is used exactly once for testing. So, I have to test each X i on a model that was learnt without using X i. And that can be represented under this notation as f hat of minus rho X i, as rho i is the index of the part to which X i belongs and f hat of minus K is the model learnt when we did not use part K, f hat of minus rho i is the model learnt when I did not use X i for training.

(Refer Slide Time: 38:37)



- The final cross-validation error estimate is

$$e_{cv} = \frac{1}{n} \sum_{i=1}^{n} L(\hat{f}^{-\rho(i)}(X_i), y_i)$$

- The final error estimate is average of errors in prediction over X_i when X_i is not used for training.
- Note that we are using each data sample once for testing and $K-1$ times for training.
- This is called K-fold cross-validation

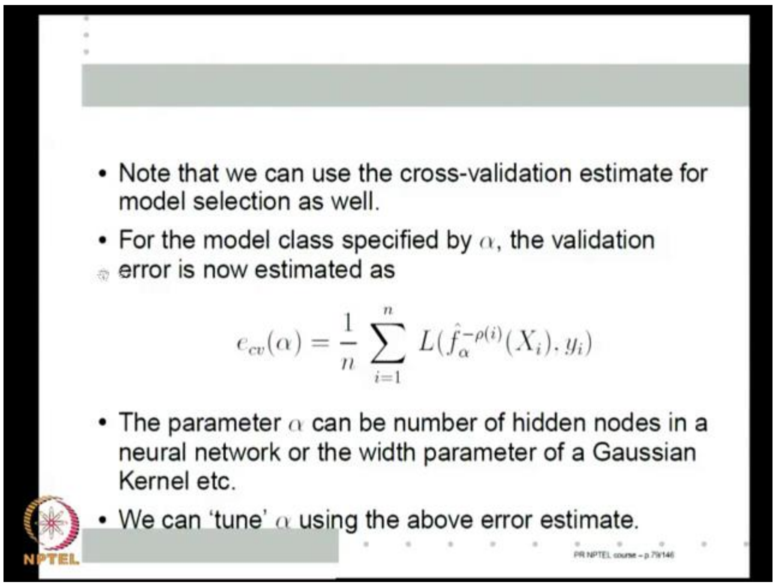
PR NPTEL course – p.75/146

So, with this notation the final cross validation error estimate which we call e f, e c v is 1 by n, i is equal to 1 to n, L f, f hat of minus rho i X i y i. Each X i is exactly once for testing, but the prediction for X i is taken from that model which did not use X i for training. So, this is the basic idea of cross validation. I use each data item, (( )) data point once for testing and K minus 1 times for training. I will, I will I will do it K times. So, each data point is used both for testing and training.

So, because I am not permanently locking up any part of the data and can get a fairly nice estimate for the error because this is different from training error because for, on each X i I am using the model that is used that is learnt without using X i and hence this is likely to give me a more accurate estimate of the error. So, the final error estimate is average of errors in predicting over X i where X i is not used for training. Note, that we are using each data sample, one is for testing and K minus 1 for training; that is the whole finesse of cross validation.

We are, we are not permanently setting aside any part of data for testing. However, to estimate the error I am using X i for testing with a model that was learnt without having X i in this training sample. That is the reason for believing that this will be a a good estimate. When we use K parts like this it is called K fold cross validation. So, the next, see we did not say whether this is validation error or test error as we already seen it does not matter.

(Refer Slide Time: 40:36)



- Note that we can use the cross-validation estimate for model selection as well.
- For the model class specified by α, the validation error is now estimated as

$$e_{cv}(\alpha) = \frac{1}{n} \sum_{i=1}^{n} L(\hat{f}_{\alpha}^{-\rho(i)}(X_i), y_i)$$

- The parameter α can be number of hidden nodes in a neural network or the width parameter of a Gaussian Kernel etc.
- We can 'tune' α using the above error estimate.

PR NPTEL course – p.79/146

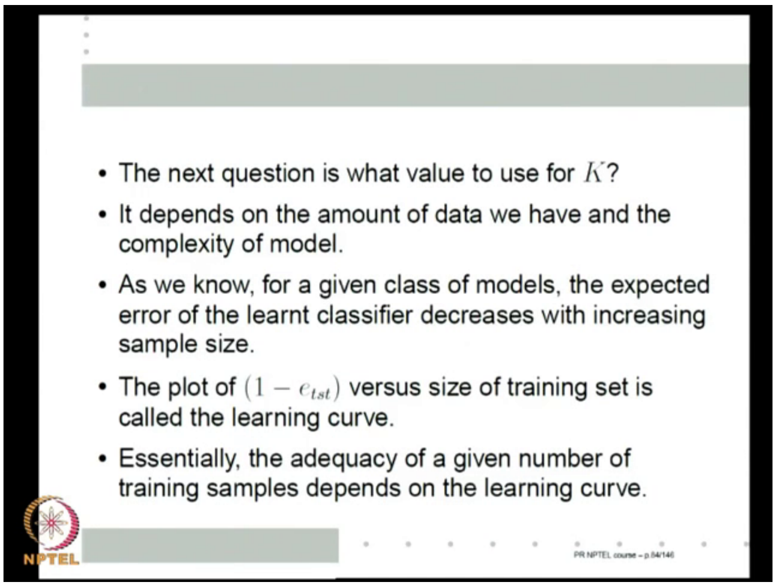
As a matter of fact one of the most one of the most often used applications of cross validation is model selection. So, cross validation is used for model selection in the following way. So, for the model class specified by alpha the validation error is called e c v alpha is given by this where f hat alpha is the model learnt from the model class alpha. And f hat alpha minus rho i is the model learnt with that particular one. So, this is how essentially I do cross validation.

If I want to find the best value of c in my SVM algorithm, I make my data into five parts. For example, for many different values of c, for each value of c I find the validation error of that value of c that alpha like this, with that value of c I learn five different classifiers. First I am using parts two, three, four, five for training; second time using parts one, three, four, five for training and so on. Then on each sample I use the prediction of of that model which did not use X i and that is how I get the error and that is the cross validation error for the, for that particular value of alpha.

This I do for different alpha and then choose choose the alpha that has the least value. So, essentially the alpha can be as as we have said n of number of times parameters of model class and we can essentially tune alpha using the above error estimate. So for example, in most cases whenever you are doing empirical investigation of any pattern recognition algorithm, this is how you choose the parameters of the algorithm. This is how we chose the model class parameters.

You use a some K for cross validation for some value of K fivefold, tenfold cross validation like this. So, essentially very often cross validation is actually used for doing model selection like this. It will also be used to estimate the final error after you learn the final classifier. After you choose a particular alpha, for that model class you once again run it once and that will give you the final cross validation error estimate for the final length classifier. So, the whole idea of cross validation is I am not permanently setting aside any part of data for the validation testing. I am using all parts of the data for training as well as all parts of data for testing. That is a very interesting finesse.

(Refer Slide Time: 43:03)



- The next question is what value to use for K?
- It depends on the amount of data we have and the complexity of model.
- As we know, for a given class of models, the expected error of the learnt classifier decreases with increasing sample size.
- The plot of $(1 - e_{tst})$ versus size of training set is called the learning curve.
- Essentially, the adequacy of a given number of training samples depends on the learning curve.

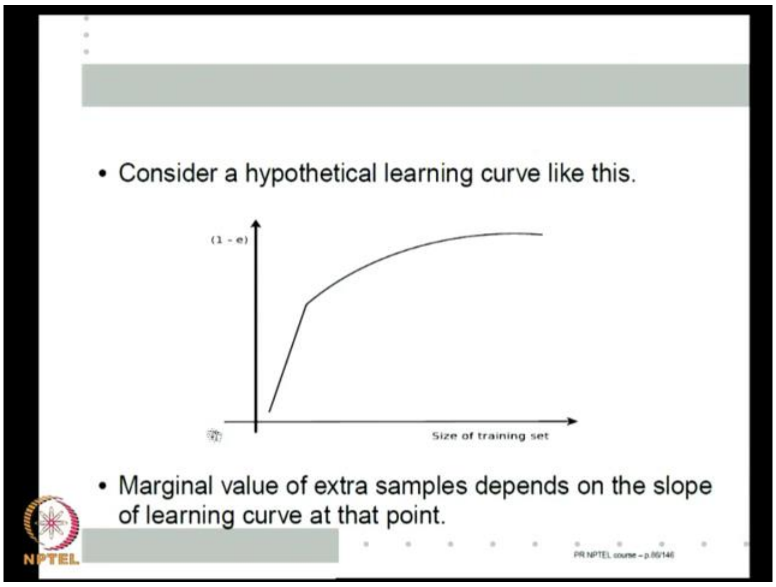
PR NPTEL course – p.84/146

Now, what value to use for K? Now, this is not very easy to say that this is not a theoretically derivable issue. It depends on the amount of data we have, the complexity of the model, the kind of learning algorithm using and so on. Basically, the reason is the following. Given any class of models and the model has specific complexity. So, if I have too few examples (( )) to the complexity, I would not learn well. If I have sufficiently many examples I learn well.

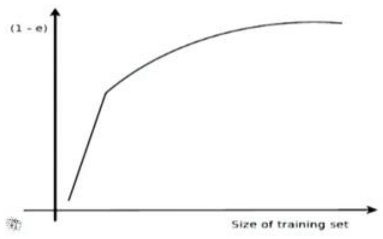
So, essentially for, for any given class of models, the expected error of the learnt classifier decreases with increasing sample size, as we know as the sample increases relative to VC dimension, empirical risk is a very good indicator of the true risk. So, we know that the expected error of the learnt classifier decreases with increasing sample size. So, if we plot 1 minus test error, that will increase with the training set. Because error decreases from, this is called the learning curve. So, if very few samples I get very high error, but as the samples go to infinity my error will be 0. So, this will go to 1. So, this is called a learning curve.

Of course, for a given problem, for a given algorithm we never have the learning curve with us; learning curve, but conceptually learning curve is a interesting thing to do. So, essentially whether or not a given number of examples is adequate to learn a from a particular model class; whether or not a given number of examples is learnt this particular SVM with this kernel function, this particular neural network with this many hidden nodes, this particular linear model with this many parameters. Essentially adequacy of a given number of training samples depends on this learning curve for that problem. I can, we can hypothetical learning most learning curves are like this.

(Refer Slide Time: 44:53)



- Consider a hypothetical learning curve like this.



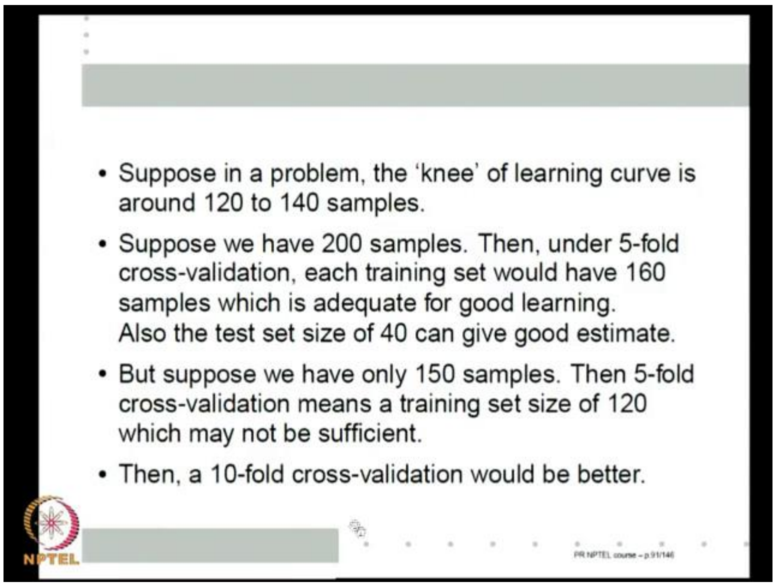
- Marginal value of extra samples depends on the slope of learning curve at that point.

PR NPTEL course – p.86/146

Of course, I put a king K not that learning curves have to have a king, but they can have a king. The idea is when examples are small, each additional example improves your learning. After a while slowly its effect tapers off. This is a general; this is general behaviour of all learning problems. In the, in the early stages when we have very few examples related to the complexity model every additional example is very useful in reducing the error.

After a while the effect of X i examples does not effect because for this model class you, this is already close n up to infinity. So, essentially the marginal value of extra samples depends on the slope of the learning curve at that point. At some in the initial part this slope is very high, at the large part slope is very low, in between it varies we do not know exactly how it varies, but there is roughly a knee like this. On all learning curves there will be a knee like this.

(Refer Slide Time: 46:03)



- Suppose in a problem, the 'knee' of learning curve is around 120 to 140 samples.
- Suppose we have 200 samples. Then, under 5-fold cross-validation, each training set would have 160 samples which is adequate for good learning. Also the test set size of 40 can give good estimate.
- But suppose we have only 150 samples. Then 5-fold cross-validation means a training set size of 120 which may not be sufficient.
- Then, a 10-fold cross-validation would be better.

PR NPTEL course – p.91/146

Now, the proper choice of K depend on how the learning curve behaves. Just to take one simple numerical example, suppose in a particular problem this knee of the learning curve is around 120 to 140 samples, somewhere around there. So, less than about 120 samples is not certainly adequate, but about 150 samples, 140 samples is always adequate at with 140 samples are fairly good confidence of learning. At 120 my confidence is small because that is the knee and there can be lot of difference between the confidence of at 120 and 140 because the knee is in between.

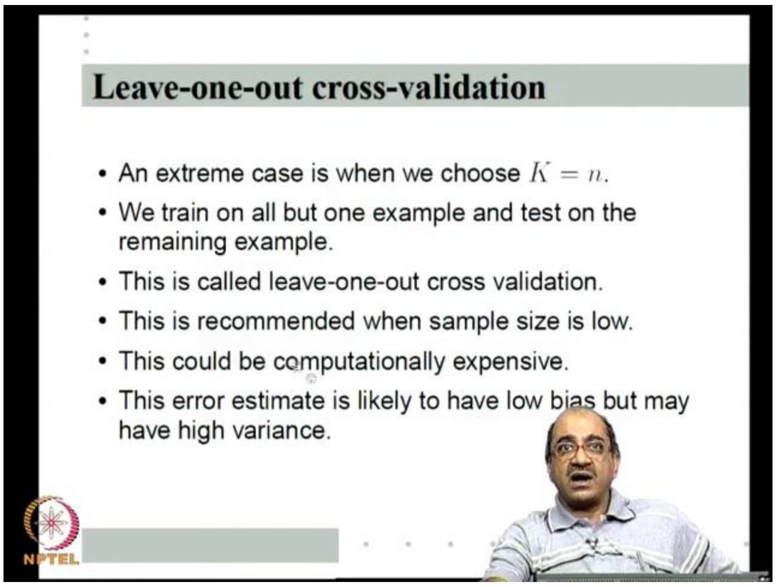
Let us say we have 200 samples, then if I do a fivefold cross validation, if I do fivefold each each part will contain forty samples. I use four four parts for training. So, every time my training set size will be 160. So, 160 is good enough to learn a good classifier. So, if I do a fivefold cross validation I certainly get, I certainly would, my training set sizes are 5 and also my test set size are 40 is also good, sufficient to give me good estimate. But suppose instead of 200 I have only 150 samples.

Then a fivefold cross validation means I am, each part will contain 30 samples. So, four parts will contain only 120 samples. So, it may not be a good enough training set size. On the other hand if I do tenfold cross validation then each part will contain 15 samples I use nine parts for training. So, in my training sets I will be 135 which is closer enough to 140. So, on the same problem if I had only 150 samples I may want to do tenfold cross validation. Now, what is the, what is the wrong in doing tenfold cross validation earlier. Then also exactly you have 180 samples for training.

That is not necessary cause what is happening is see the whole idea of cross validation because of learning with different training sets. I will learn sufficiently different (( )), so that the actual error I get is the true error. And different different f hat minus K's are learnt here on samples where 40 samples differ from training set to training set. Here only 15 set, if I did tenfold cross validation only 20 samples will differ from training set to training set.

So, you may, I may learn roughly the same classifier. So, I may not get a good estimate. So, if I can give sufficient variability in the training examples learnt by different classifiers I am likely to get a better estimate. So, essentially you know if I, if I do too large, too many parts I may get a high variance and (( )) if I do too small I may not get enough, I may have to add bad bias in my estimate. So, basically if I have 200 samples in this kind of problem I can do fivefold. If I have only 150 may be I want to do tenfold.

(Refer Slide Time: 49:16)



Leave-one-out cross-validation

- An extreme case is when we choose $K = n$.
- We train on all but one example and test on the remaining example.
- This is called leave-one-out cross validation.
- This is recommended when sample size is low.
- This could be computationally expensive.
- This error estimate is likely to have low bias but may have high variance.

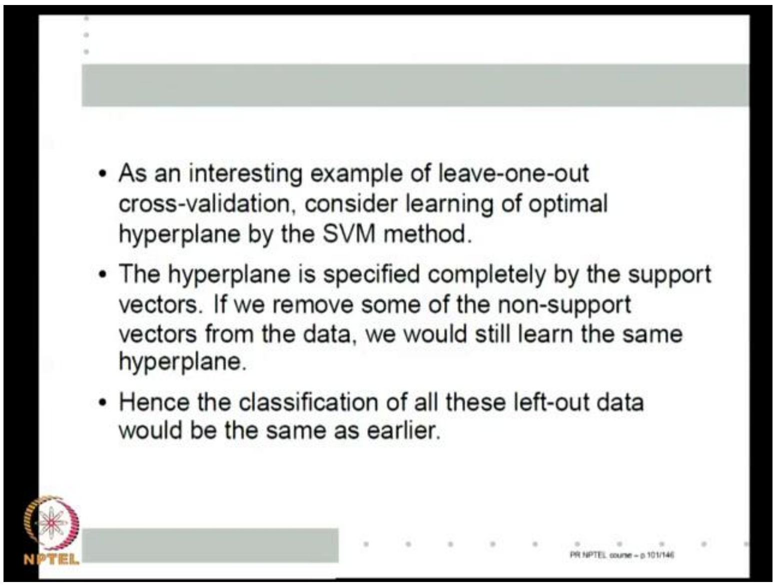
A rule of thumb most often one employs as a five or ten fold cross validation which is often good enough. An interesting special case of cross validation is what is called leave one out cross validation where we choose K is equal to n, where n is the number of examples. So, have as many parts as there example. That means these example is its own part. What does that mean? Because I use K minus 1 parts for training and one part for testing we train all, but one example and test on the remaining example.

So, for each example I leave one example out, train on the remaining examples and whatever I learnt on that I test on this example. Of course, my different training sets are very close to each other. So, but on the other hand I am using you know exactly, use example once for testing still on a model that is learnt without using it. Leave one out cross validation; this is called leave one out cross validation because every time we are leaving one example. It is recommended when sample size is really low.

As a matter of fact though its, it may not sound like much, if your sufficiently many examples leave one out cross validation in an expected sense gives you the true error. So, its bias is very low. But its variance can be large because you are you know we are going to leave one out, but its bias is small. So, the expectation of the leave one out cross validation error estimate would be close to the true error estimate true risk if sample size is sufficiently large.

But of course, this is also like (( )) sample size is low. Its main problem is it is also computationally very expensive. So, we do not want to go for it unless we really want to get this kind of an estimate because you have to run your classifier learning algorithm n times, where n is the number of examples you have. The essentially the error estimate is likely to have low bias, but may have high variance. But this is one very special kind of a cross validation estimates. It is called leave one out cross validation estimate. Very often even when data is large if we can do it one does this to get some estimate of the expected true risk. Also is often resorted to when we have very small sample sizes.

(Refer Slide Time: 51:33)



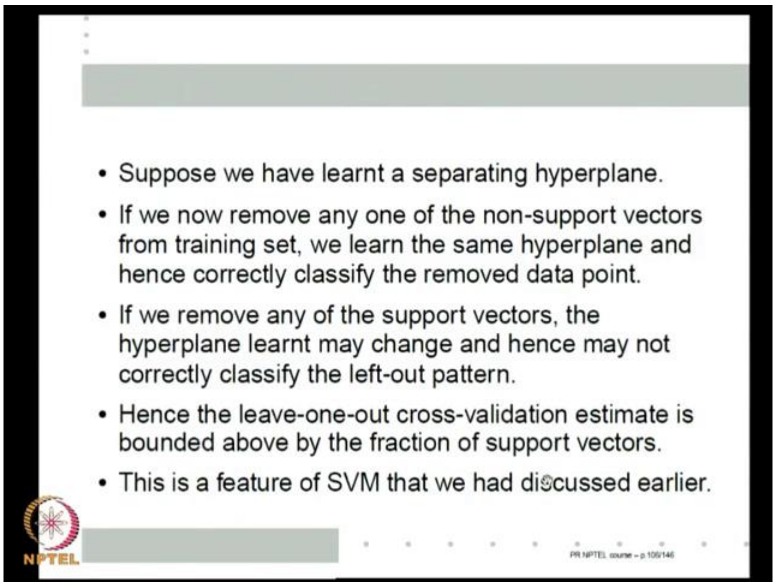
- As an interesting example of leave-one-out cross-validation, consider learning of optimal hyperplane by the SVM method.
- The hyperplane is specified completely by the support vectors. If we remove some of the non-support vectors from the data, we would still learn the same hyperplane.
- Hence the classification of all these left-out data would be the same as earlier.

PR NPTEL course – p.101/146

So, let us consider one interesting example of leave one out cross validation. Let us say we are learning an optimal hyper plane using the SVM method. A SVM SVM method I do not have to do this learning multiple times to know what, what happens when I do, when I leave one example. That is why we chosen SVM for this example. Do you know that in a, in the SVM the hyper plane is completely specified by the support vectors. So, if we remove some of the non support vectors from data you would still learn exactly same hyper plane by the nature of the optimization problem.

When we, when we did the theory of SVM we saw this that the hyper planes completely specified by the support vectors. And once having guaranteed, now if you keep this support vectors and throw away some of the non support vectors from the data and relearn resolve the optimization problem, we will get exactly the same solution which means if I throw away some of the non support vectors then we still learn the same hyper plane which means the classification of all these left out data would be same as earlier.

(Refer Slide Time: 52:34)



- Suppose we have learnt a separating hyperplane.
- If we now remove any one of the non-support vectors from training set, we learn the same hyperplane and hence correctly classify the removed data point.
- If we remove any of the support vectors, the hyperplane learnt may change and hence may not correctly classify the left-out pattern.
- Hence the leave-one-out cross-validation estimate is bounded above by the fraction of support vectors.
- This is a feature of SVM that we had discussed earlier.

PR NPTEL course – p.106/146

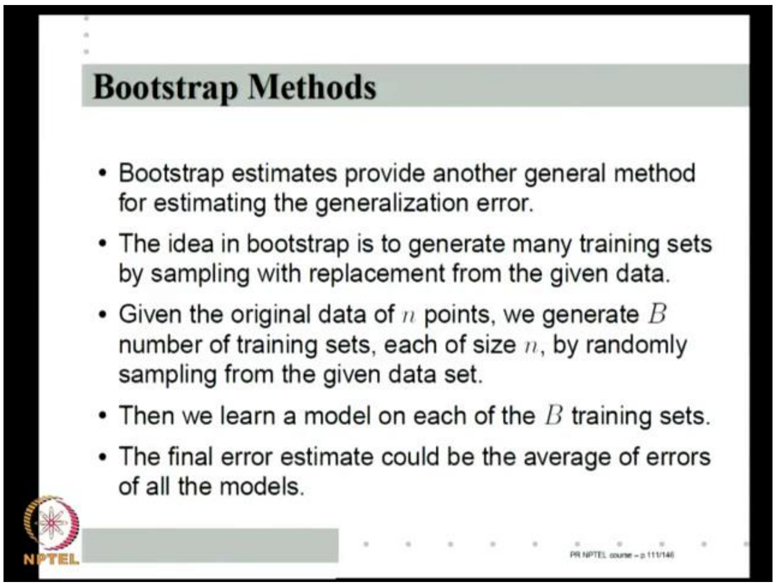
You can use this to actually calculate what will be the leave one out cross validation estimate. Suppose, you have learnt a separating hyper plane, so if we now remove any one of the non support vectors from the training data set we will learn exactly the same hyper plane which means we correctly classify the removed data point. We leave one out, we leave, removing 1 X learning f and only on that X you are testing and all those test errors are averaged.

So, whenever I remove a non support vector I get 0. Let us say we are doing 0 1 loss function. So, I get no error because I learn exactly the same hyper plane and hence it will still classify the earlier, earlier, the point as earlier. So, there is no error. Other than that if you remove any of the support vectors I do not know. I may or may not learn the same hyper plane and hence I may classify the left over point wrongly. So, what is the maximum error I can get?

Each when when I leave out each of the support vectors, the learnt one may not, may not classify the support vector correctly (( )) correctly, but whenever I leave out any of the non support vectors, I will I will learn the same hyper plane. So, my error will be 0. So, which means my leave one out cross validation estimate is bounded above by the fraction of support vectors. As a matter of fact when we did support vectors we showed, we stated that the expected probability of error is bounded above by the fraction of support vectors.

This is essentially how it comes about. So, the leave one out cross validation estimate is bounded above by the fraction of support vector. That is how fraction of support vectors is a very useful quantity in SVM because it is a leave one out cross validation estimate of the probability of error by the learnt classifier. This is the feature that we already discussed earlier, but this is interesting to see its connection with the leave one out cross validation estimate.

(Refer Slide Time: 54:28)



Bootstrap Methods

- Bootstrap estimates provide another general method for estimating the generalization error.
- The idea in bootstrap is to generate many training sets by sampling with replacement from the given data.
- Given the original data of n points, we generate B number of training sets, each of size n, by randomly sampling from the given data set.
- Then we learn a model on each of the B training sets.
- The final error estimate could be the average of errors of all the models.

PR NPTEL course – p.111/146

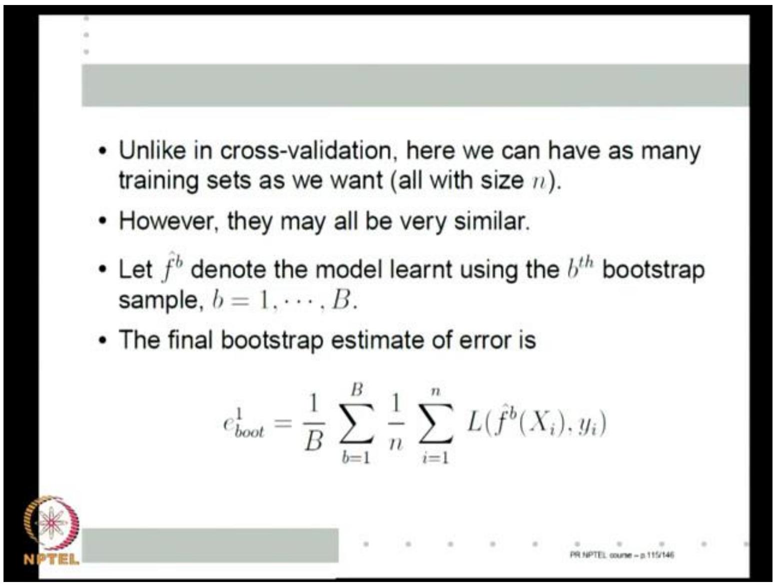
And another method of doing this similar thing of using your training data to get test error estimate is bootstrap. Bootstrap estimates provide another general method of estimating test error, generalization error, true risk whatever you call it. The idea in bootstrap is to generate many training sets from the given data by sampling with replacements. Sampling with replacement means I have X 1 to X n. I choose a particular index randomly between 1 and n, put that data point into my this training set.

We then generate one more index set which could be the same as what we said. So, every time I generate a number between 1 and n and that corresponding data I take into my training set. So, my training set may contain the same training point more than once, but by sampling with replacements like this I can generate many training sets of the same size. So, the idea is given the original data points, n points we generate B number of training sets. So, first I generate n random numbers each uniformly distributed between 1 and n that gives me one training data set.

Once again, I generate n random numbers, each uniformly distributed between 1 and n. That gives me one more training data set. Like this given the original data of n points, we can generate any number of training set. Let us say capital B is the number of training sets that we generate. The idea is then we learn a model on each of these B training sets and the final error could be average of error of all these models. So, this is another method, this is called bootstrap because it is essentially of course, this is a trained same training data set.

But using the same training data by repeatedly sampling we are we are generating many many training data sets and learning many different model and averaging out all their errors. So, the idea is just like the phase pulling one up by one's bootstraps. Using the same training data we are somehow getting at the test error. That is why it is called the bootstrap estimate.

(Refer Slide Time: 56:44)



- Unlike in cross-validation, here we can have as many training sets as we want (all with size n).
- However, they may all be very similar.
- Let $\hat{f}^b$ denote the model learnt using the b^{th} bootstrap sample, $b = 1, \cdots, B$.
- The final bootstrap estimate of error is

$$e^1_{boot} = \frac{1}{B} \sum_{b=1}^{B} \frac{1}{n} \sum_{i=1}^{n} L(\hat{f}^b(X_i), y_i)$$

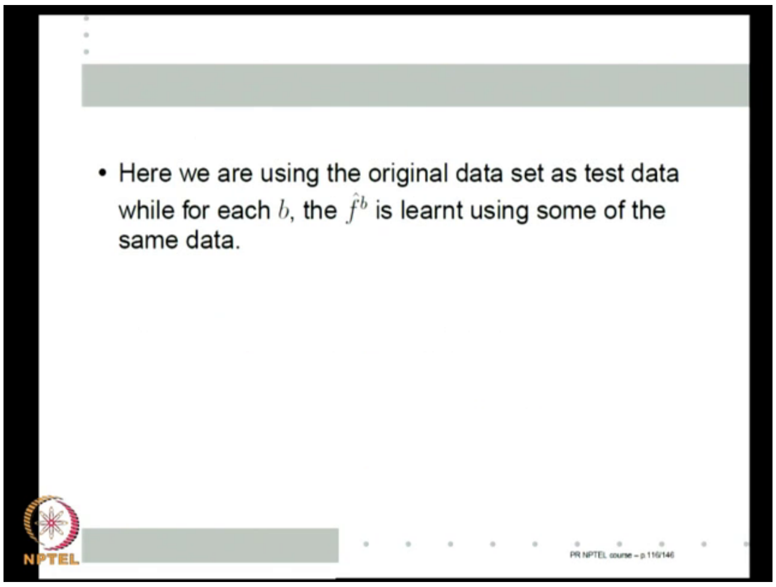
PR NPTEL course – p.115/146

The the main thing is unlike in cross validation we can have as many training sets as we want, all of size n because we are sampling with replacement we can keep generating as many as you want. B is at our control. See in the in the cross validation we cannot generate any more then n training data sets. I mean because I can at most put n parts. I cannot divide data into more than n parts, but here I can generate as many training data sets as possible, B can be anything.

However, they can be very similar that we will we will come to this later. We will later on see how similar they are. Once again, let us just t complete the the the formulism. So, like earlier let us say small b denotes the index of the bootstrap data set or bootstrap sample that we are generating. So, the capital B number of bootstrap samples, f hat small b denotes the model learnt using the b th bootstrap sample. Then the final bootstrap estimate of error is, I will, I will first call it e 1 boot is see this 1 by n i is equal to 1 to n L of f hat b X i comma y i.

If the training error or if the error on the original data set not… See, b is not learnt fully on the original data set. It is learnt on some sample version of the original data set. So, this is the the error or the estimate of true risk of the model learnt on the b th bootstrap data set. And average it over all the b bootstrap (( )). So, the idea is that this in an interesting error estimate. Of course, there are two problems with this, with at least one problem with this error estimate.

(Refer Slide Time: 58:39)



- Here we are using the original data set as test data while for each b, the $\hat{f}^b$ is learnt using some of the same data.

PR NPTEL course – p.116/146

Here we are using original data set as test data while for each b f hat b is learnt using some of the same data. So, essentially if I want to call this second summation as the the test error of f hat b then it is like I am using X 1 Y 1, X 2 Y 2, X n Y n as the test error, but some of these X's are used in training and hence learning f hat b. So, one does not expect this to become a very good estimate. So, this is not a good average. So, next class we will see a simple example of why this is not a good estimate and then see how we can improve it. And then moving on from there we have, we are learning yet many classifiers that can be learnt from the same training data set. We will look at how we can make such a method become very effective by learning more than one classifier.

Thank you.