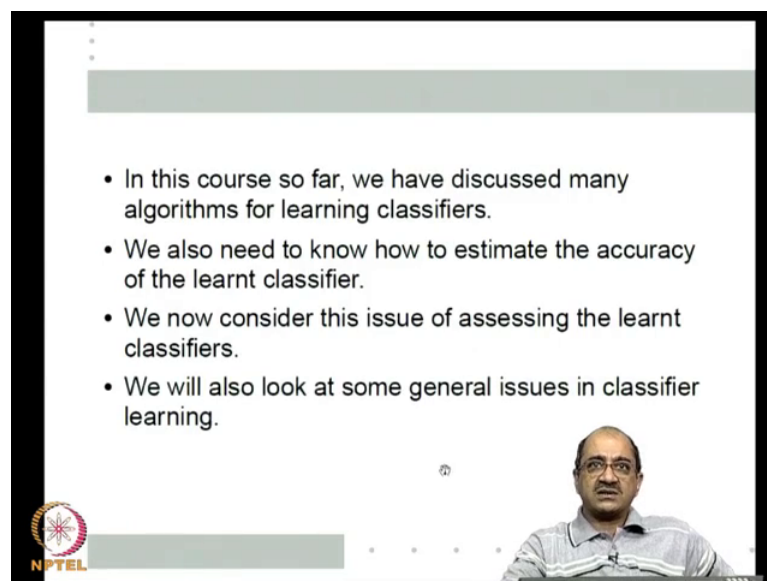


Pattern Recognition
Prof. P. S. Sastry
Department of Electronics and Communication Engineering
Indian Institute of Science, Bangalore

Lecture - 39
No Free Lunch Theorem; Model selection and
model estimation; Bias-variance trade-off

Hello and welcome to this next lecture in the course of pattern recognition, this talk will mostly discuss a few issues related to assessing what we learn correctly or not bother any specific algorithm for learning classifiers. So, let just step back and take a look at what have we done so far. In this course what we have done so far is that we discuss many different learning algorithms, various techniques starting from perceptron or starting from Bayes classifier with estimated densities. We had seen various learning algorithms various methods for learning classifiers.

(Refer Slide Time: 00:59)

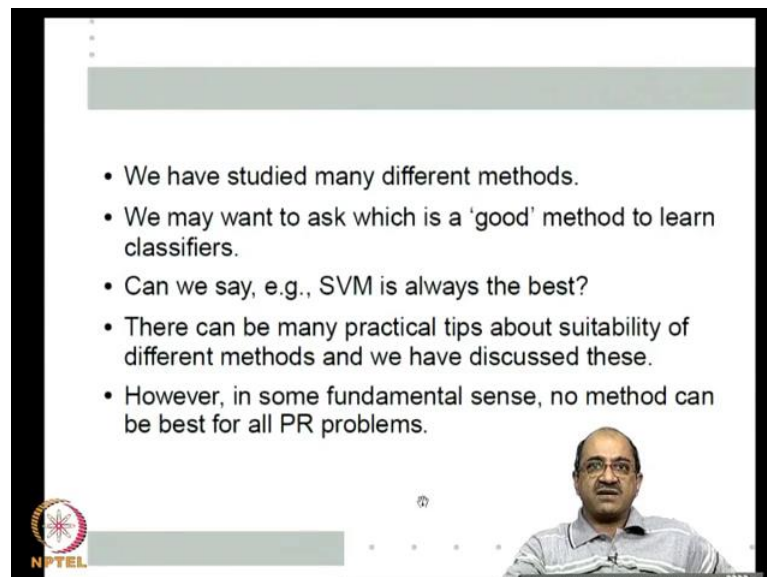


- In this course so far, we have discussed many algorithms for learning classifiers.
- We also need to know how to estimate the accuracy of the learnt classifier.
- We now consider this issue of assessing the learnt classifiers.
- We will also look at some general issues in classifier learning.

We given so many methods and let us say on a specific problem we got some examples and we use one of the methods to learn a classifier. Then one thing that is certainly needed is we need to know how to estimate the accuracy of the final learnt classifier. So, we learning from some finite data, so while may be many of the algorithms have some kind of optimality proof there is mostly asymptotic. So, we somehow need to estimate

whether what we learnt is good enough for our application or not.

(Refer Slide Time: 02:04)



The video frame displays a presentation slide with a list of bullet points. In the bottom right corner, there is a small inset video of a man speaking. The NPTEL logo is visible in the bottom left corner of the slide.

- We have studied many different methods.
- We may want to ask which is a 'good' method to learn classifiers.
- Can we say, e.g., SVM is always the best?
- There can be many practical tips about suitability of different methods and we have discussed these.
- However, in some fundamental sense, no method can be best for all PR problems.

Now, that is one aspect that we look at. How do we assess the correctness or otherwise or assess the level of performance of the learnt classifier? But before going there we look at some general issues in classifier learning that we nothing to do this specific method you are using no matter what method you are using there some generic issues. Now, the first question that we may want to ask along the general issues is what a good method is, we we studied many algorithm. We started with Bayes classifiers, then we said if you know the class conditional densities then Bayes classifier is the best you can get. But then we said of course, we do not know the class conditional densities who will give to us.

We said, but we have data, so we can estimate class conditional densities, we studies various method for estimation. Then we said, but the estimated densities may have lot of errors and do not know how those errors effect the goodness of Bayes classifier. Then we said can we directly estimate classifiers we will looked at discriminant function both for classification regression. We looked at linear techniques, we looked at perceptron LMS least square methods Fisher discriminant.

Then we went on to look at non-linear classifiers neural networks SVM's. Now, it the


quote a progression from simple methods to more difficult methods, the loss classifier algorithm we learnt is the best classifier algorithm, then what is the purpose of learning the others? So, in general you know, but having studied, so many different techniques before the quote is over we should ask, what is a good method? So, for example, as you saying can we say that SVM always learns the best classifier.

Now, of course whenever we considered different methods, we discussed many practical tips about when different methods are suitable? When would I use a know least squares Bayes linear regression? When do I use a non-linear regression method? When may I want to use SVM's? When may I want to use neural network. But these are some level of practical tips and we have discussed this enough through the course. But one thing would like to emphasis because you are teaching the n is that in a fundamental sense no method is best for all problems.

The reason we studied so many methods is that each method as if so niche where it will perform well. Well it is very difficult to completely characterize that niche and say for that if in this PR problem use this method. Of course, if you could characterize that there is no scoop for you know scientist for regression. But while each method each algorithm as its own niche in the abbreviation space.


In a very fundamental sense we can never say one method is the best or one method is definitely better than some other method and so on so for. And this in a very fundamental sense, so we first start from here, so that we will not think that you know there is a god given method or a (()) for all pattern recognition problems. There is no single algorithm that will do on all problems.

(Refer Slide Time: 05:07)



Inherent superiority of a classifier?

- We can say a classifier learning algorithm is inherently superior to another if the first method learns a better classifier on all (or most) PR problems.
- As it turns out, no algorithm is inherently superior like this.
- If a method does better on some PR problems then it would do worse than average on some others.
- We will now formalize this notion regarding relative merits of different learning algorithms.



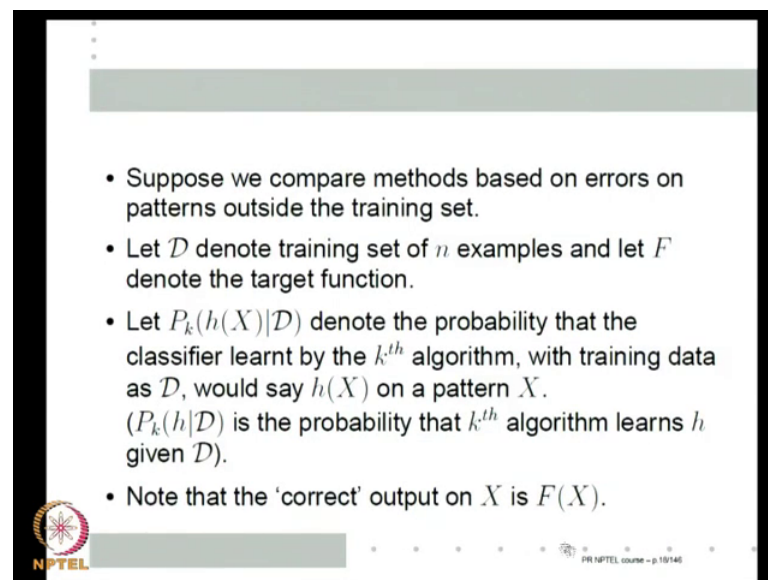
So, we can talk of this as inherent superiority of a method a method could be a, you know an SVM with the specific kernel function with specific values of parameters and so on. So, far we can say a classifier learning algorithm we will is inherently superior to another algorithm if the first one always learn a better classifier than the other. If you think always is a little to straw this a most of the time in most pattern recognition problems (()) for most pattern recognition problems if one method always learns superior classifiers to the others.

Then we can say this method is inherently superior to the other as it turns out there is no algorithm that is inherently superior like this. What is that mean, if an algorithm does better on some pattern recognition problems? Certain kinds of application problems, if it is doing very well compare to another method if it works much better it learns, but a classifiers. What did we mean is there would be other instant sub pattern recognition problems where it would do worse than average because even on most problems it cannot do better than any other method.

So, what we now going to say is why there is a there is a very formal way of saying this we will at least look at the statement of this formal theorem. We will not be able to prove this, but we look at the statement of the formal theorem that captures this thing that no

classifier is inherently superior. So, what we going to do is that there is no way I can say one classifier is inherently superior and we will say it in a very formal sense as a matter of fact we will say it as a theorem.

(Refer Slide Time: 07:07)



- Suppose we compare methods based on errors on patterns outside the training set.
- Let \mathcal{D} denote training set of n examples and let F denote the target function.
- Let $P_k(h(X)|\mathcal{D})$ denote the probability that the classifier learnt by the k^{th} algorithm, with training data as \mathcal{D} , would say $h(X)$ on a pattern X .
($P_k(h|\mathcal{D})$ is the probability that k^{th} algorithm learns h given \mathcal{D}).
- Note that the 'correct' output on X is $F(X)$.

NPTEL

PR NPTEL course - p.10140

So, now how do we compare methods we will agree to compare methods, so maybe we will give the same training data to two different methods and each method will all put a classifier. And then we can compare the methods we asking the classifiers output by the two method, how they perform on patterns outside the training set. This seems fair enough because both methods are given the same training set. Then they can view the training say training which way they want and then they will come out with a classifiers.

Once they come out with classifier were saying we will test the classifier on all patterns outside the training data training set and the that is how we will compare the a errors of classifiers learn by two different methods. So, let us put some notation let as earlier script \mathcal{D} denote the training set and let say n is the number of training examples we will keep n fix throughout the discussion. And let say F denotes the target function let the training example actually come classifier through F .

So, the F is the target, so we will think of it as classifier or regression it really does not

matter there is a target function that we will learning. Let $P_k(h|X)$ denote the probability that the classifier learnt by the k th algorithm using training data D would say h of X on a pattern X . When for the k th algorithm if you give training data D it will learnt some classifier on that classifier when X would say h right.

That is the probability of it happening essentially what it means is let say $P_k(h|D)$ is the probability that the k th algorithm learns the function in h given data D . So, if this is the probability of k th algorithm learning h than $P_k(h|X)$ will be the probability that will say h of X and X . Now, of course, given an X my learning algorithm will say h of X with this much probability where as the true output is $F(X)$

(Refer Slide Time: 09:27)


• Let $J_k(F, \mathcal{D})$ be the expected error when k^{th} algorithm learns target function F using training data \mathcal{D} .

• It is given by

$$J_k(F, \mathcal{D}) = \sum_h \sum_{X \notin \mathcal{D}} P(X)[1 - \delta(F(X), h(X))] P_k(h(X)|\mathcal{D})$$

where $\delta(a, b) = 1$ if $a = b$ and it is 0 otherwise.

• If we assume the learning algorithm to be deterministic, then, for a given \mathcal{D} , $P_k(h|\mathcal{D})$ is non zero only for one h . Then we can omit the first summation.

 PR NPTEL course - p.27946

So, how can I define the error so let us say $J_k(F, \mathcal{D})$ is for the k th algorithm the target function F is fixed that training data set \mathcal{D} is fixed. So, for a particular target function F given a particular training data said \mathcal{D} with the k th pattern recognition algorithm. Let us say $J_k(F, \mathcal{D})$ be the expected error of the classifier learnt by the k th algorithm using training data \mathcal{D} with target function F . Now, how, how can I give this so $J_k(F, \mathcal{D})$ on X and I said we already agreed to first letters consider the under summation.


We already agreed to measure the errors on X which are not in the training data. So, I sum over all X naught in D a particular X will come with probability P of X on X my algorithm will say h of X with probability $P_k h$ of X . And this tells you it is right or wrong δa comma b is one if a is equal to b 0 otherwise. So, if $F X$ is equal to $h X$ there is no error if $F X$ is equal to naught $h X$ there is error this is like a 0 or loss function. So, and I am summing over h because my k th algorithm I learnt different h with different probability.

Of course, if the learning algorithm is deterministic what all the learning algorithms will consist, so far or mostly deterministic then given a particular training data such the because the learning algorithm are just a computer program. Now given the same data set it always learn the same function which means given a D my what I called $P_k h$ given D that is probability of the my algorithm learning h is non 0 only for one h because given the same D it will always learn the same h .

So, $P_k h$ given D is non 0 for only h that means on a particular D a k th algorithm learns only one particular h . So, this h summation say can actually go but writing like this is much more general even if the algorithm with the same D with different probabilities its output different h 's is then also this expression is right. But because most of our algorithms are deterministic I do not need the summation with respect to h . This, then is enter prated as given D I used that h here in this in this expression, which will be learnt by the k th algorithm. To say that I used that h , which would be learnt by the k th algorithm is what this 0 on factor is $P_k h (())$ this is what deterministic algorithms.

So, any case k th algorithm learns h with probability P_k of h given D on with training data D so on an X the probability as at my be classifier learnt with k th algorithm we will say h of X is this. This is the error of saying h of X 1 minus $\delta F X$ for multiply weight by the probability of X that is my error summed over all X naught in the training set and I have to summed over h because of this probability. As I just, now said if the learning algorithm is deterministic for only h this is non 0. Hence effectively the summation the out a summation will reduce to only 1 h .

(Refer Slide Time: 12:52)



‘No Free Lunch’ Theorem

- The ‘No Free Lunch’ theorem is a formalization of the fact that no learning algorithm is inherently superior.
- The theorem says: for any two learning algorithms,

$$\sum_F \sum_D \text{Prob}[D|F] (J_1(F, D) - J_2(F, D)) = 0$$

(we take $|D| = n$ fixed)

- Averaged over all size- n training sets and uniformly averaged over all targets, F , the difference in expected errors of any two algorithms is zero.

PR NPTEL course – p.24/48

Now, the formalization of the fact that there is no inherently superior classifier is called no free lunch theorem. This is an American color (()) that no free lunch means, if somebody is offering you free lunch it certainly means that he wants some work from you. He or she wants some work from you, so there is no such thing as a free lunch you do not, you never get in this life you never get something for nothing. So, that is what this theorem says and once we complete the statement theorem we will come back and see why it is called no free lunch.


The no free lunch theorem is a formalization of the fact that no learning algorithm is inherently superior. What is the theorem? The theorem says the two partial theorem the first part is for any two learning algorithms we will call them 1 and 2 we will talk about k th algorithm. So, let us say any two learning algorithms 1 and 2 this quantity 0 what is this quantity? It is product of one time is probability D given F for a given target function.

What is the probability of generating a data set D into $J_1(F, D) - J_2(F, D)$. This is the difference in the error rate of the two algorithms, when given D weighted by getting D for getting D for training summed over all d right. This is the expected error overall possible training data sets that I may get if I fixed the target then I

summed over all F , which means uniformly over all F . If I assume that all possible target functions all possible pattern recognition problems are equally likely then this outer summation will have some whatever because every problem is at the same weight.

The weight does not matter, so summed over all possible F . This is 0, so let us say it words averaged we have as I said we always keeping my training set size fixed then all this discussion. So, as I noted here, so averaged over all size n training sets and uniformly averaged over all targets F this uniformly is very important because there is no weight for different F 's all F 's are the same weight. Averaged over all size n training sets and uniformly averaged over all targets the difference in the expected errors of any two algorithms is 0 no matter what is the two algorithms. If I averaged the expected error over all possible target functions uniformly averaged it uniformly then the expected error is 0 a matter of fact second part says that it is nothing to do with the training data set per say.

(Refer Slide Time: 15:45)



- The second part of theorem says that this is true, even if we fix \mathcal{D} :

$$\sum_F (J_1(F, \mathcal{D}) - J_2(F, \mathcal{D})) = 0$$

- If we uniformly average over all problems (i.e., target functions, F), all algorithms are the same!
- No inherently superior classifier design strategy exists.
- A specific learning technique may be good if we are interested in only certain subsets of PR problems!

PR NPTEL course - p.20/46

I can even keep \mathcal{D} fixed if I want for a particular training data set uniformly averaged over all possible target function meaning all possible pattern recognition problems the expected difference in the expected error of any two algorithms is 0. So, if we uniformly averaged over all problems all algorithms are same no one algorithm is any better than

other. That is what it means because this is the expected error of algorithm one. This is the expected error of algorithm two.

You take the difference and uniformly sum it over all F whether you can either fix the training data set or weigh the weight with respect to training data set like k . But in either case, if you uniformly averaged over all possible functions f the difference in the expected error of any two algorithms that really does not matter. What is the method? Of course, one of the algorithms may be arbitrary, right? It may always give you the same function irrespective to the data, I am not saying anything about what kind of method this is? No matter what the two methods are, the expected error that is the difference in error is 0 for uniformly average over all problems. All algorithms are same, no one algorithm is any better than the other.

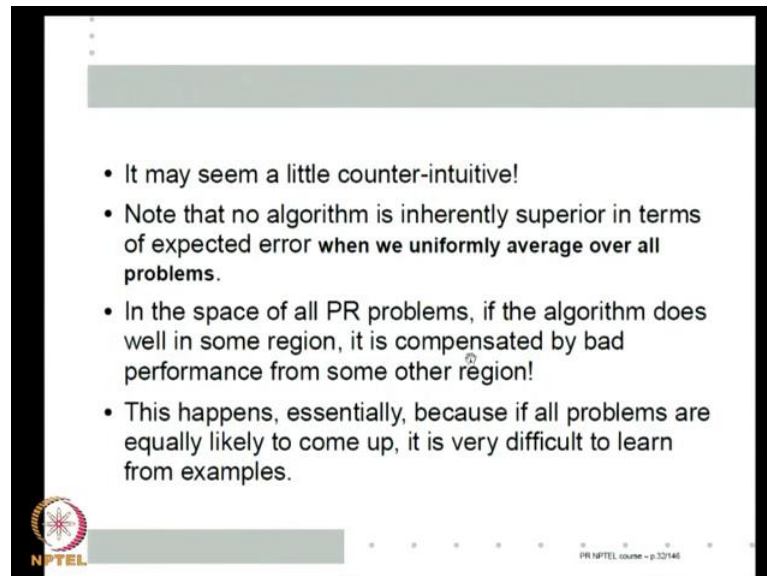
That means no inherently superior classifier design strategy exists, there is no method which allows you to learn a classifier. That is inherently superior to any other way of learning a classifier, any other meaning an arbitrary way of learning a classifier is not inferior. Not inferior in our inherently superior sense is not inherently inferior to any other algorithm that we already know or would have discovered. So, what does this mean? It means a specific learning technique may be good if we are interested only in certain subsets of PR problems.

If you think of a set of all possible PR problems if in some subset one algorithm does much better than others then it must be doing much worse than others in some other part of the space of all possible problems. The only way I can do better on certain class of problems is by doing worst on certain other problems. If all problems were equally then averaged over all possible problems the error is, one algorithm is no different from any other algorithm. So, any specific algorithm essentially as in space and it is exploiting the characteristic of that subset of pattern recognition problems.

To do better on them and the price we pay for it is that because you are, we are tuning the technique for some special property special structure. That that class of pattern recognition problems may have we would be doing badly in some other part of space of all possible pattern recognition problem. This is it in this sense is called a no free lunch

theorem if an algorithm is able to do better than other algorithms on certain problems it is able to do like that only by doing worse than another algorithms on some other problems. You cannot get good performance on certain problems without paying the price a bad performance on other problems that is why there is no free lunch.

(Refer Slide Time: 19:26)



- It may seem a little counter-intuitive!
- Note that no algorithm is inherently superior in terms of expected error when we uniformly average over all problems.
- In the space of all PR problems, if the algorithm does well in some region, it is compensated by bad performance from some other region!
- This happens, essentially, because if all problems are equally likely to come up, it is very difficult to learn from examples.

NPTEL

PR NPTEL course - p.32/146

As I said we conclude the theorem, but we can look at some, some comments on what it could mean and also we will look at it in a for example, setting of Boolean features. So, which you get a sense of why it happens at this stage it may look a little counter intuitive at the end of a tough pattern recognition course. If I tell, you, you know no algorithm is better than nearer algorithm not only that, no algorithm is better than near the algorithm all the algorithms I am teaching or no better than any arbitrary method anybody can think of.

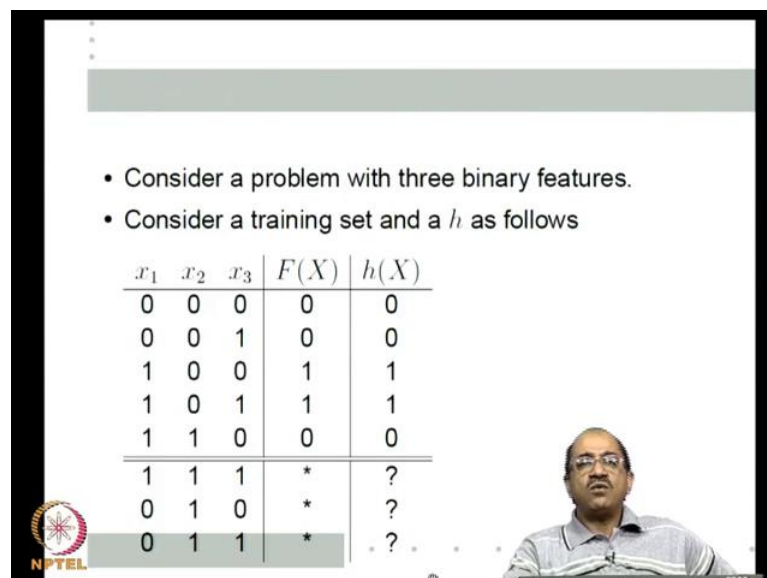
Then what is the purpose of all this? Well, remember that no algorithm is inherently superior when uniformly average over all problems; no algorithm is inherently superior in terms of expected error only when we uniformly averaged over all the problems. If every possible every conceivable pattern recognition problem is a equal importance to us only then no no algorithm is inherently superior space of all PR problems. If the algorithm does well in some region it is compensated by bad performance, some other

region that is all the theorem says does not say that an algorithm cannot do better than others in one particular region.

Now, when if you think every problem is equally important then obviously there is nothing to learn from examples. Let think of a two class problem, every pattern recognition problem is simply one particular subset of real line. If on different days I am asked to learn different subset of real line and every subset of real line is equally important to me, then I cannot have any general strategy of learning we already seen that.

Only for example, I am learning axis parallel rectangles then I can make a very nice efficient algorithm, but those good only if I know beforehand that I am interested only in axes parallel rectangles. On the other hand if I am interested in learning every conceivable subset of real line \mathbb{R}^2 then obviously it is very difficult to learn from examples. This is something you already know roughly but, in the no free lunch theorem nails it down completely that is in terms of expected error if we average over all possible PR problems no algorithm is better than any other algorithm.

(Refer Slide Time: 21:57)



• Consider a problem with three binary features.

• Consider a training set and a h as follows

x_1	x_2	x_3	$F(X)$	$h(X)$
0	0	0	0	0
0	0	1	0	0
1	0	0	1	1
1	0	1	1	1
1	1	0	0	0
1	1	1	*	?
0	1	0	*	?
0	1	1	*	?

The slide includes the NPTEL logo in the bottom left corner and a small video inset of a speaker in the bottom right corner.


Let us see what it means in a very simple setting then it does not look, so counter intuitive, let us say we consider a problem with just three binary features. The three

features $x_1 \times x_2 \times x_3$ a feature vector dimension three and all features are binary. So, essentially my space has only eight possible feature vectors. Let us say this is my training set, so I am shown some five patterns. So, these are 1 0 0 0 0 1 whatever these five patterns I am shown this is the values of the target function and let us say some h in my space had the same value, I have a h that is consistent on the training sample.


Now, obviously I am use some method to learn, of course I may output a function; that is consistent may output function consistent but I know the target function value. Let us assume there is no noise. Now, what is it I am interested in? I am interested in asking if I learn this h . How will it perform on unseen data? So, some pattern that is not in the training set. Of course, my $F(X)$ has some value what will my h say. I am not really bothered about values in training set, of course here I am, I may choose that they are same, but I am essentially interested in how will h perform on unseen patterns.

In this particular case that three unseen patterns, so five are in the training set. So, only three unseen patterns are there I am asking every h I am we are going to evaluate based on how well the predictions of h match the predictions of F . But it is not that were going to average this error or all possible F how many such possible F are there.

(Refer Slide Time: 23:50)



- Here there are a total of eight possible feature vectors.
- Since 5 are in training set, there would be $8(=2^3)$ boolean functions consistent with the training set.
- On any of the three patterns outside the training set, half of these Boolean functions would take '1' and the other half would take '0'.
- Hence, no matter what h is, averaged over all these F , its error is same!




Here there are a total of eight possible feature vectors five of them already in the training set. So, there are only three feature vectors outside the training set, so each of these three, so how many different F 's can I have. Now, the F has to choose these values for on the five so these are the possible F 's over which will be comparing all F 's that can take different values on these three. So, how many such F 's are there $2 \times 2 \times 2$ right, there are eight such F 's there. So, there are eight Boolean functions consistent with the training set those are the only ones and which I will be finding the errors because we are finding the error only on X not in the training set.


Now, as all of you know from such Boolean functions that if I look at all possible Boolean functions for this three row truth table then obviously on any one row of the truth table half the functions will be one half the functions will be 0 that is how the Boolean functions behave. So, on any of the three patterns outside the training set half these eight Boolean functions would take 1 and the other half would take 0, which means... Each of those patterns irrespective whether $h(X)$ is 1 or 0, see at all that the different learning algorithms can affect is whether it is that the corresponding $h(X)$ will be say 1 or 0 on these three patterns outside the training set. Irrespective of whether my $h(X)$ is 1 or 0 half the time is right half the time is wrong if I average over all possible F .

So, my algorithm can do nothing, all my algorithms are those given me a h which can only decide to say 1 or 0 on each of these patterns. Irrespective whether it says 1 or 0 on any one pattern its error when averaged over all f is always 0.5. Because half the patterns its prediction will be right, half the functions is for half the F its prediction will be correct for half the F its prediction is wrong. So, the fraction of F 1, which it will predict wrong is always half irrespective what h is? These essentially, what the no free (()) no matter what h is averaged over all these F its error is the same and hence no learning algorithm is better than other if we averaged over all F . And the other hand let say all F are not equally important to be.

(Refer Slide Time: 26:28)



- Now let us say, that we are only interested in Boolean functions that can be represented by a single term (conjunction of literals).
- It is easy to verify that the training data is consistent only with the function $x_1\bar{x}_2$.
- Thus, when we are interested only in this subset of PR problems, all learning algorithms are not the same.
- This is essentially the sense of No Free Lunch Theorem.



Let say when I start learning when I am given this problem I was also told that the target function is a single mean term right mean term as you remember from your brain algebra disc is a conjunction of literals. In general you write sum a products form, so each time its called a mean term. So, let say the Boolean function that I am interested in can be represented by single term, a single conjunction of literals. These are the... So I am only interested learning Boolean functions that can be represented as a single term.

Now, that a quit a structural constraint without going you did not tells of course looking at this table those of sure interested can verify, given this training data that the F X values. You can ask how many possible mean terms are? How many possible single mean term functions are consistent with this data? I wont go into the details is a very simple thing to do, but one can show that the training data consistent only with this one particular function $x_1\bar{x}_2$, which means if I am working in the space of function that can be represented by single mean term. That is the only niche of pattern recognition problem that been interested.

Now, all algorithms are not same because there is only one function, this consistent with all training data that is the function I must be having because I know before (()) that my my the functions are which I am learning are single mean terms. So, this is the only mean

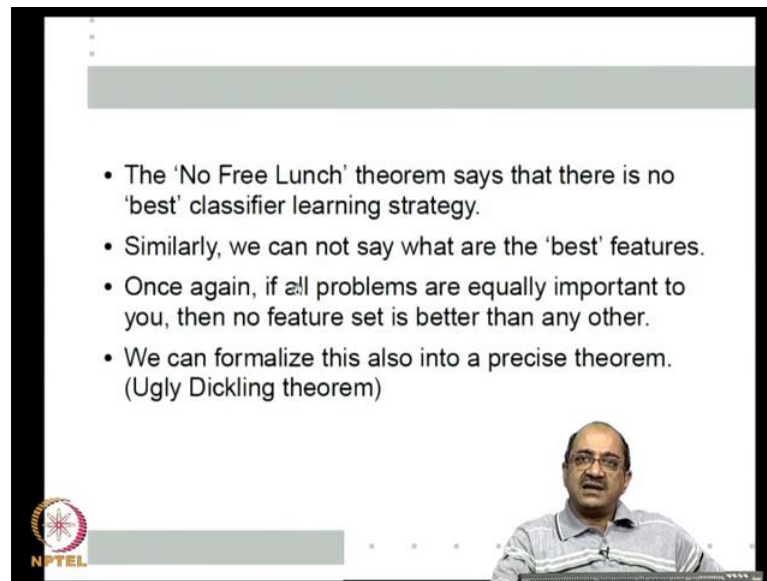
term that is, that could be generating data. So, if I average over all F , this is all F that satisfy this structural constraint is only one function. Hence, a h that predicts as per this function below 0 error every other h will not have 0 error.

Hence one method can be superior to other if I restrict my attention to learning only those target functions which can be represent the single mean term. This essentially is the sense of no free lunch theorem when we are interested only in this subset of PR problems a learning algorithm or not all learning algorithms are same any more. Otherwise where interested in all possible target functions obviously there is no algorithm can be superior to another.

So, this is the sense of no free lunch theorem, so they cannot be an algorithm that does well very well on some set let say cannot even have algorithm that can do very well on some subset and do average on the rest that is not possible. Because if they do very well on some subset of pattern recognition problems then what the no free lunch theorem assures as is that they do very badly not average they do very badly on some other subset of pattern recognition problem. Because uniformly averaged over all pattern recognition problems no algorithm is different from any other algorithm.

So, if there is an algorithm that has very well compare to another algorithm on some subset than this algorithm will do very badly compare to the other on some other subset pattern recognition problems. So, depending on the problem on hand we may have to choose an algorithm, but while this is in general true as a general tip for how one makes pattern recognition systems work is also true in this fundamental sense. That is not just because we do not know enough we do not have, we have not at found the ideal pattern recognition in any algorithm. There is no such a thing as a an ideal algorithm for learning classifiers that the such a thing does not exist that is what our no free lunch theorem guaranties.

(Refer Slide Time: 30:21)



- The 'No Free Lunch' theorem says that there is no 'best' classifier learning strategy.
- Similarly, we can not say what are the 'best' features.
- Once again, if all problems are equally important to you, then no feature set is better than any other.
- We can formalize this also into a precise theorem. (Ugly Duckling theorem)

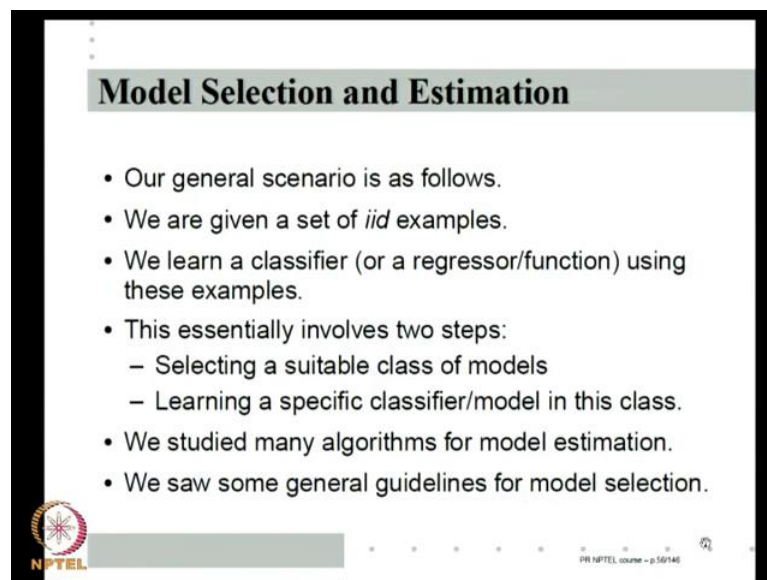
No free lunch theorem is to say that there is no best classifier learning strategy. So, that is nice and it is not as if we can be searching around for this one algorithm after which we can stop working on pattern recognition problem. So, depending on the kind of problems we have to find some algorithm that works best for the problems of interest to us and that application. In a same way there is no such thing that the best features there is no god given features for any problem, not for any problem there no god given features once again in a problem independence sense.

That is if all problems are equally important to you, then one feature set is as good as any other feature set, it really does not matter. For once again, uniformly I have raised your all problems, no feature set is better than any other this can also be proved I would not even state this because this is a little more complicated stated precisely this call ugly duckling theorem. The origin for the name is that all of us think the ducklings are very ugly as a matter of fact duckling in English language is used as the kind of symbol for ugliness, but obviously I am sure the ducklings mother would not agree the ducklings is ugly.

So, the ugliness is in the eye of the beholder, so that is what it says that there is no such thing at the right feature set. If all problems are equally important then no see features set

is better than any other, this theorem is called ugly duckling theorem any way I said we are not formalizing this. So, this is one generic issue about the classifier descent so we understand, now that there is no inherently superior classifier (()) strategy.

(Refer Slide Time: 32:04)



Model Selection and Estimation

- Our general scenario is as follows.
- We are given a set of *iid* examples.
- We learn a classifier (or a regressor/function) using these examples.
- This essentially involves two steps:
 - Selecting a suitable class of models
 - Learning a specific classifier/model in this class.
- We studied many algorithms for model estimation.
- We saw some general guidelines for model selection.

NPTEL

PR NPTEL course - p.56746

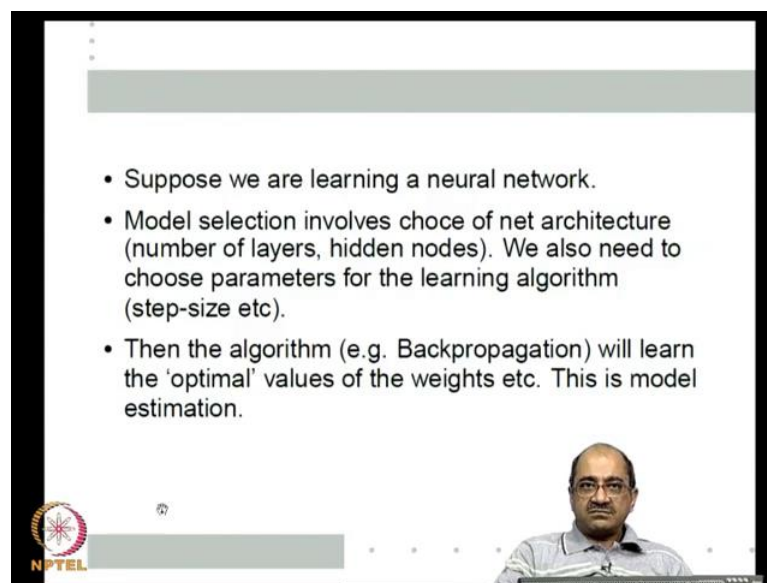
Now, a few others generic issues, so our, the entire idea of learning is the following where given a set of *i i d* examples. Then we are asked to learn a classifier or we learn a classifier or a regression function using this examples a linear regressor or a non-linear regressor or in general a function or a classifier we given training examples we learn a classifier this is the whole idea. This involves two steps learning a classifier or a function given training examples, as we seen through the course involves two steps.

What are the two steps? I mean to broad steps one is at a first select a suitable class of models, linear functions a neural network or you know non-linear function represent the kernel. What have you I choose a suitable class of models? Then I learned a specific element of that class a specific model specific classifier from that class. So, the two issues involves, one is I have to first decide or which class of models I will learn and then within that class I learn them.

So, the first one called model selection and the second one is called model estimation.

What we studied throughout the course are algorithms for model estimation. Back propagation is a method that if you fix if search is over a specific class of neural networks and gets you the one that has the least error right each of the algorithm that we considered are essentially model estimation algorithms. Of course, made is point we said how do I decide the model how do I decide how many hidden nodes to have my neural network, so that is the issue of model selection.

(Refer Slide Time: 34:04)



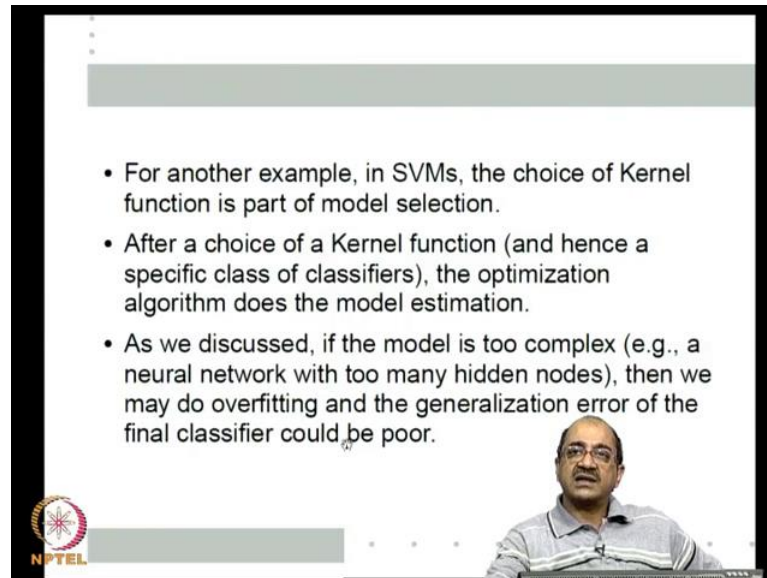
- Suppose we are learning a neural network.
- Model selection involves choice of net architecture (number of layers, hidden nodes). We also need to choose parameters for the learning algorithm (step-size etc).
- Then the algorithm (e.g. Backpropagation) will learn the 'optimal' values of the weights etc. This is model estimation.

So, for an example suppose you learning a neural network model. What is model selection involves? It involves choice of the network architecture that is let Us say I have to first chose the number of layers one hidden layer two hidden layer so on. Then I have to chose the number of hidden nodes I have to chose activation functions then we have to choose the parameters for the learning algorithm all this right. All this is part of the model selection, so choosing a set of hidden layers and hidden nodes in each hidden layer will give me one particular class of neural networks, one particular class of classifiers, one particular class of models.

Now, within that class within that family of classifiers we are minimizing mean square error we are minimizing empirical risk using squared error lost function, to learn a particular classifier. For example, using back propagation we learn and we learn optimal

values of weights etcetera, which means we are picking out one of neural networks of that structure right this is the model estimation.

(Refer Slide Time: 35:15)

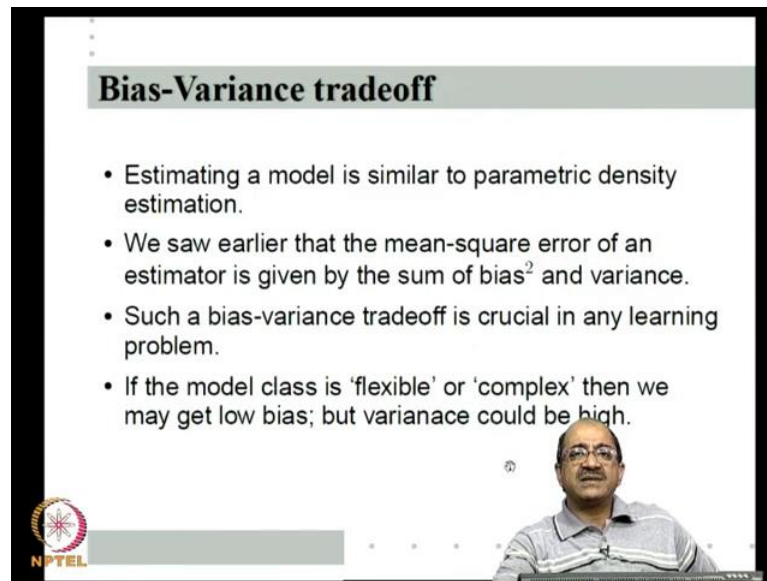


- For another example, in SVMs, the choice of Kernel function is part of model selection.
- After a choice of a Kernel function (and hence a specific class of classifiers), the optimization algorithm does the model estimation.
- As we discussed, if the model is too complex (e.g., a neural network with too many hidden nodes), then we may do overfitting and the generalization error of the final classifier could be poor.

For another example if I am using an S V M the choice of kernel function is part of the model selection I do decide whether I want let say polynomial kernel with P is equal to 3 or polynomial kernel with P is equal to 4 so on. So, that kind of decides which family of classifier which family of models I am searching over after I have choose that for example, after I choose the kernel function, then the optimization algorithm does the model estimation.

So, whether or not it is explicitly state a like this all are algorithms involve both model selection I mean essentially model estimation, but we have to also do a model selection right. Of course, we do not know what does the selected model is the right one we already discussed various stages that is the model is too complex. For example, neural network has too many hidden nodes if the v c dimension is very large then we may be doing over fitting. This generalization error of the learn classifier will be poor it will not do well on unseen data right. So, model selection is a very crucial issue because depending on the kind of function we want to learn depending on the amount of data we have, we have to choose the right model. How do I do model selection?

(Refer Slide Time: 36:34)



Bias-Variance tradeoff

- Estimating a model is similar to parametric density estimation.
- We saw earlier that the mean-square error of an estimator is given by the sum of bias² and variance.
- Such a bias-variance tradeoff is crucial in any learning problem.
- If the model class is 'flexible' or 'complex' then we may get low bias; but variance could be high.

NPTEL

A small inset video shows a man with glasses and a grey shirt speaking.

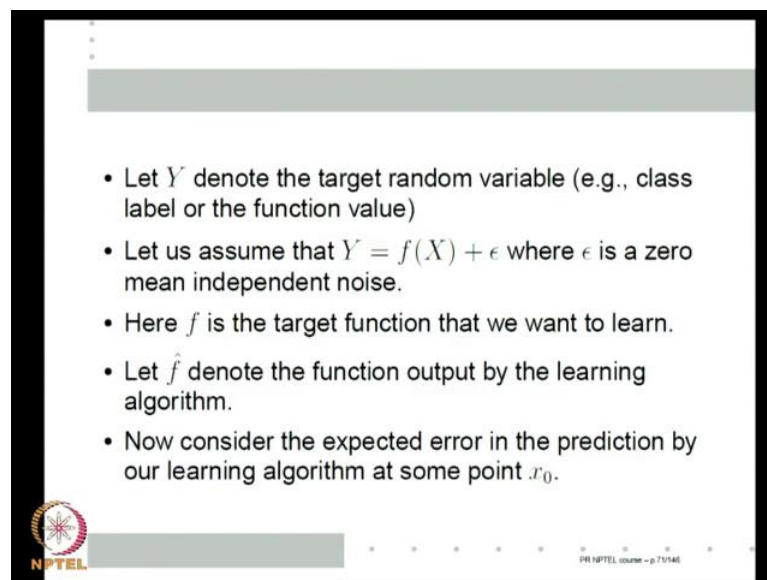
Some generic issues about what this model selection means this is often seen the many different ways of seeing it. In this class we will consider only one which is called the bias variance trade off. Estimating a model is very similar to parametric density estimation that we consider. When we discussed parametric density estimation we saw that the mean square error of an, of an estimator is given by the sum of its variance and bias square. We show that the mean square error of any estimator can witness bias square plus variance.

So, essentially very often there is a tradeoff between them. We may not be able to reduce the mean square error very much; the model selection may be trading of bias verses variance or in very nice scenario may reduce poor. So, this kind of bias variance decomposition is crucial in many learning problems and essentially what it means is the following as we seen this enough number of time this course if the class of models whoever which I am searching is too complex.

Then if I average over all possible training data I may get I would buy in large get the right function. Because you know the model class is very rich and the target function are very close approximation dude exist in my bag. So, that is essentially what low bias means, but because it is very complex on most realistic size date sets I may not be able to do a very good step of learning. Hence depending on the examples I have given I may

pick over different functions and hence variance would be very high. So let us just look at this in the little more detail what this bias variance decomposition is, once again to keep thing simple. We will look at it in a very, very simplistic scenario.

(Refer Slide Time: 38:43)



- Let Y denote the target random variable (e.g., class label or the function value)
- Let us assume that $Y = f(X) + \epsilon$ where ϵ is a zero mean independent noise.
- Here f is the target function that we want to learn.
- Let \hat{f} denote the function output by the learning algorithm.
- Now consider the expected error in the prediction by our learning algorithm at some point x_0 .

NPTEL P00 NPTEL course - p.715146

Let say Y denote the target random variable that you are predict that you want to predict class label or the function value in a regression problem. Lets also assume that the relationship between the target and feature vector X is Y is equal to f of X plus epsilon where epsilon is a 0 mean independent noise simplest of problems so f is essentially the target function. I want to learn the target function f given data training data which are X comma Y where Y is f of X plus epsilon. Let say \hat{f} at if the function output by the learning algorithm, now let say I want to ask what is the expected error in the prediction made by a our learning algorithm as prediction made a fact at specific point X naught let say.

(Refer Slide Time: 39:35)

We have

$$\begin{aligned}
 \text{error}(x_0) &= E \left[(Y - \hat{f}(x_0))^2 | X = x_0 \right] \\
 &= E \left[(f(x_0) + \epsilon - \hat{f}(x_0))^2 \right] \\
 &= E \epsilon^2 + E \left[(f(x_0) - \hat{f}(x_0))^2 \right] \\
 &= \sigma_\epsilon^2 + E \left[\{ (f(x_0) - E \hat{f}(x_0)) + \right. \\
 &\quad \left. (E \hat{f}(x_0) - \hat{f}(x_0)) \}^2 \right]
 \end{aligned}$$

NPTEL

PR NPTEL course - p.75148

So, the error I let us say we will take the mean square error. So, error at X naught is expected value of Y minus f at X naught whole squares where the Y is with respect to an X which is equal to X naught that is why I put condition X equal to X naught. So, we already know when X is equal to X naught Y is f of X naught less epsilon. So, I can write this f of X naught plus epsilon f at X naught whole square. Now, I get an epsilon square plus f X naught minus f at X naught whole square the cross term will go.


Because the cross term contain epsilon into f X naught minus f at X naught epsilon is independent of all this. So, expected value of the product will be expected value of epsilon into some term and expected value of epsilon is 0 because epsilon is 0 minimize. So, the cost term will go away because noise is 0 mean and independent. So, when expand the square I only get a specular epsilon square and specular of f X naught minus f at X naught whole square.

Now, if you remember what the same algebra that we did for a parametric point estimator I can rewrite this term as follows f X naught minus f at X naught whole square added and subtracted expected value of f at X naught. So, wrote it as f of X naught minus specular f at X naught plus specular f at X naught. Then grouped these separately why, now effectively square I get this square f X naught minus specular of f at X naught whole

square then this square expect value of f at X naught minus f at X naught whole square. A cost term a cost term is inside the expectation the first fact of the cost term $f X$ naught minus specular f at X naught is constant because $f X$ naught is a constant by definition this is a expectation of something.

So, this also constant that comes out with the expectation then the second term is specular f at X naught minus f at X naught and when I put expectation outside of that that will be 0. So, the cost term will be 0 that is giving me sigma square epsilon expected value of this square and expected value of this square, expected value of this square is $f X$ naught minus specular f at X naught whole squares. I do not need expectation because the constant the second term the expectation of this I have just wrote the other way, so as it looks familiar f at X naught minus specular f at X naught because these square anyway does not matter, so this is what I get finally.

(Refer Slide Time: 41:45)



$$\begin{aligned} \text{error}(x_0) &= \sigma_\epsilon^2 + (f(x_0) - E\hat{f}(x_0))^2 + E[(\hat{f}(x_0) - E\hat{f}(x_0))^2] \\ &= \sigma_\epsilon^2 + \text{bias}^2(\hat{f}(x_0)) + \text{Var}(\hat{f}(x_0)) \end{aligned}$$

- Bias is about how well, on the average, the learnt function captures the true underlying function.
- The variance is about how close would the predictions of the learnt function would be (over all training sets).

PR NPTEL course - p.79546

So, this is what we can write as this is the inherent error because of the noise variant this is noise variance this is noise variance because my data given is $f x$ plus epsilon. So, whatever is the variance in epsilon that the minimum error I have to make anyway then this is the bias this term is called bias squares. This is the variance this we already know f at x naught minus specular f at x naught whole square expectation is nothing but variance


of f at x naught this is called the bias of f at x naught.

What is bias? Tell us bias a difference between f of x naught the actual target function minus expected value of f of x naught what is the expectation with respect to obviously by training data sets. Different days I get different training data sets averaged over all training data sets. So, f at is a function of the training data, so f at is a random only because function of the training data. So, this expectation with respect to the training data distribution, so averaged over all possible training data set do a on the averaged make the right prediction.

So, bias is about how well on the average the learnt function captures the true underlying function here. Of course, we wrote written error at a particular point, but essentially were interested error all the other points, so essentially it is a matter of whether the expectation of f at is the same as the function f . So, bias is about how well on the average the learnt function captures the true underlying function. What about the variance? Variance tells me, so expected value of f of x naught is what I learnt averaged over all training data sets, where as this is what I learnt any one so to say so this tells you how much variance is there.

How close would be predictions of the learnt function would be? Over all training sets one day aligned with one training set. So, I learnt one function that make some prediction another day I learnt with another training data sets hence I learnt another function that may make a different prediction am asking how different are this prediction the predictions are close to the whether the variance will be small predictions are different. So the variance is about how close will be predictions of the learnt function over training sets because that is important because I will essentially learn with only one training set right I cannot averaged over all possible training sets.

(Refer Slide Time: 44:26)

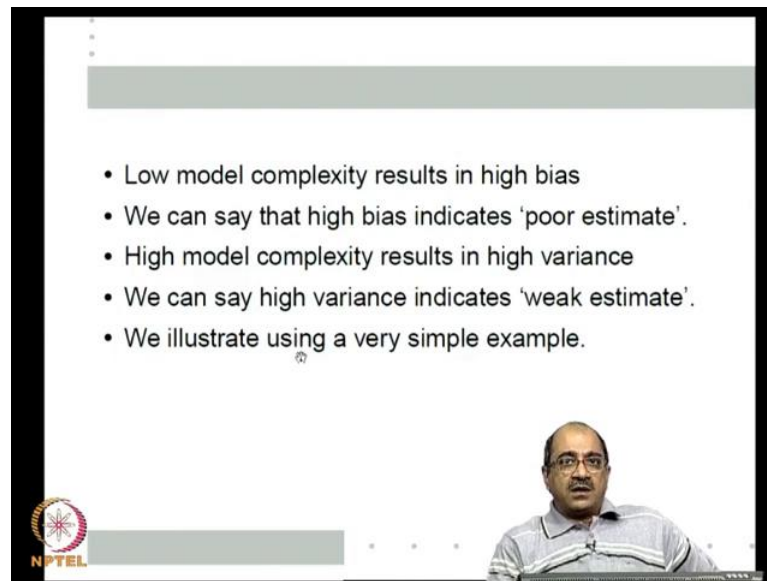


- If the class of functions over which we are searching is rich enough then it is likely to contain good approximators to the target function.
- Then, averaged over all training samples of a given size, we would be learning a function close to the target
- But if the model class is very rich, then, unless we have very large training sets, we may not learn well.
- Hence with different data sets we may learn very different functions thus giving a high variance.

PR NPTEL course - p.03146

The class of functions over which you are searching is rich enough there is likely to contain good approximation of the target function. Hence averaged over all training samples we may learn a function that is close to the target, so if the class is rich enough our bias is likely to be small, but the model class is rich unless you have very large training set we may not learn very well. In sense on any given day with the given training data I may not be able to learn the same function. So, on different data sets we may learn different function and that is high we make a high variance this is the whole idea of the bias variance decomposition in the final error we make.

(Refer Slide Time: 45:06)




The video frame displays a presentation slide with a white background and a black border. At the top, there is a grey header bar. The slide contains a bulleted list of five points. In the bottom right corner, there is a small inset video of a man with glasses and a mustache, wearing a light blue shirt, speaking. In the bottom left corner, there is a circular logo with a star and the text 'NPTEL' below it.

- Low model complexity results in high bias
- We can say that high bias indicates 'poor estimate'.
- High model complexity results in high variance
- We can say high variance indicates 'weak estimate'.
- We illustrate using a very simple example.

So, low model complexity results in high bias. If the model is very restricted I may not be able to find a function that is close to the target function. So, I may have lot of bias, so if I am learning you know tenth degree polynomial using linear models obviously even the best algorithm can only give me linear function. So, my model complexity is very small so they will be lot of bias. So, we can say high bias indicates poor estimate I am not chosen the right class to look for the function high model complexity and the other hand will have low bias, but will result in high variance. So, we can say high variance indicates a weak estimator, because we do not have the power to estimate from such a high model complexity unless you have very large number of examples.

(Refer Slide Time: 46:01)



- Suppose we use the k -nearest neighbour method.
- That is,

$$\hat{f}(x_0) = \frac{1}{k} \sum_{\ell=1}^k y_{(\ell)} = \frac{1}{k} \sum_{\ell=1}^k (f(x_{(\ell)}) + \epsilon)$$

where $x_{(1)}, \dots, x_{(k)}$ are the k nearest neighbours of x_0 (in the data set).


- We need to find $E[\hat{f}(x_0)]$ to calculate the variance (or the bias).

PR NPTEL course - p.55146

So, once again we look at this in a very, very simple example suppose my learning method is a k nearest neighbor method I am thinking of it is a regression. So, what it means is my \hat{f} at any given x naught is average of the k nearest neighbors I can find in the data set the values function values the k nearest neighbors. So, I represent nearest neighbors is x_1 to x_k . So, that corresponding function value is the data set is y_1 to y_k . So, if $y_{(l)}$ of within brackets l is the l th nearest neighbor value that is $x_{(l)}$ is the l th nearest neighbor to x naught then its value on the training data set is $f(x_{(l)}) + \epsilon_l$.

Because training data set I get only noise corrupt values so average of all this is what my predictor would be if I am using nearest neighbor algorithm, so this is my prediction. Now, I have to find expected value of \hat{f} of x naught right if I put an expectation here what are the random things here, of course ϵ is random and $x_{(l)}$ is also random because it depends on the training data set because to which of the closest to x .

(Refer Slide Time: 47:10)



- In the expression for $\hat{f}(x_0)$, the $x_{(\ell)}$ are also random.
- But let us ignore this for the sake of simplicity. Then


$$E[\hat{f}(x_0)] = E\left[\frac{1}{k} \sum_{\ell=1}^k (f(x_{(\ell)}) + \epsilon)\right] = \frac{1}{k} \sum_{\ell=1}^k f(x_{(\ell)})$$

- Now we can calculate the variance easily.

PR NPTEL course - p 54/148

So, do calculate calculating this expectation is very difficult because x is random let's ignore it for now. If I ignore it, so I put an expectation outside, but I am not treating x as a random then the epsilons will go away because epsilons are mean 0 I get this. That means between f at x naught, which is this and expect a value of f of x naught the only difference is epsilon right. Because we already seen that we are not worrying about average with respect to this, which you do not have reduce there enough the training example then maybe they distributed roughly around this same way. So, then epsilon is the only difference between the specular of f of x naught and f of x naught.

(Refer Slide Time: 47:57)



• We have

$$\begin{aligned}\text{var}(\hat{f}(x_0)) &= E \left[\frac{1}{k} \sum_{\ell=1}^k \epsilon \right]^2 \\ &= \frac{1}{k} \sigma_{\epsilon}^2\end{aligned}$$


• Thus the variance decreases monotonically with k .

PR NPTEL course - p 50/48


So, the variance is very simple, variance of $\hat{f}(x)$ is the expected value of $\hat{f}(x)$ squared minus the square of the expected value of $\hat{f}(x)$. The difference is only epsilon squared. So, this is $\frac{1}{k}$ because I am summing over k nearest neighbors and this square. Essentially it has to be $\frac{1}{k}$ in the sense it is the error random variable that corrupts the noise random variable that corrupts the k th training data. That particular training data there are anyway independent.

So, all the cross terms will go away and the any single epsilon squared term is nothing but the sigma squared epsilon. This square $\frac{1}{k^2}$ will make this $\frac{1}{k}$ as $\frac{1}{k^2} \times k = \frac{1}{k}$ square, so, ultimately I will get sigma squared epsilon by k . What it means is if I am essentially using a nearest neighbor estimator variance decreases monotonically with k .

(Refer Slide Time: 48:50)




- As k is increased, more smoothing is done in nearest neighbour estimate and hence variance is low. (This is true even considering the randomness of $x_{(\ell)}$ also).
- However, if we smooth too much the predicted value is likely to be far from target, giving high bias.
- We would get low bias if k is small. If $k = 1$, then we do not smooth at all.
- But low k means high variance.



This is intuitively very clear as k increases we are doing more and more smoothing in my estimate. I am taking average values of sufficiently many nearest neighbors of the current point as the estimate. Hence you know irrespective of what errors set I started with if I am doing average of enough of them and the data is you know distributed well then I roughly about get the same value. So, as k is increased I do more and more smoothing a matter of fact if k is equal to my data size I always give you the same value.

One can show this is true even considering the randomness of x_l in general however. If you smooth too much the predicted value is likely to be far from the target giving rise to high bias. So, if k is very large as I said if I do k nearest neighbor classifier and k is equal to the number of training data. No matter what the feature vector is I always call the same class right. So, if I smooth too much obviously predicted value can be far from the target giving rise to high bias we get low bias in this case if k is small if k is equal to one we are not doing smoothing at all. We get very low bias then, but then we get high variance right.

(Refer Slide Time: 50:10)




- Such bias-variance tradeoff is common in any learning method.
- In general, estimating bias and variance of a specific method is difficult.
- If we have large number of training samples then, usually, variance would be small.
- Also, with large sample size, we can choose a richer model class and reduce bias.
- In general, we need some knowledge of the target function and sufficient number of examples to reduce both bias and variance.

PR NPTEL course - p100148

So, this is essentially the issue or it always comes in various gadgets and different algorithms, so such a bias variance tradeoff is common in most learning methods. So, in general estimating bias and variance of a specific method is difficult. The generic issue is we have large number of training samples then usually variance would be small. So, variance can be reduced by having enough training data if you have large training data this generally no problem, because your variance will be small.


Also if I have a large training data then I can choose a richer model class and hence reduce bias. So, in general that is fine, but very often we do not get large training data. So, if you wanted I chose both variance and bias sufficiently we need to a sufficient number of examples also we need to a some knowledge of the target function. So, it be can intelligently chose the model class you want to chose the model class, so that we have we will have a function that is close to the target function. At the same time the class is not so rich that we will not be able to learn it fit the example that we have this often an art, but this is one way of looking at the tradeoff involve in designing a pattern recognition system.

(Refer Slide Time: 51:18)



Assessing a learnt model


- We use the training set to learn a model (or classifier).
- We are actually interested in how well the classifier would perform on new (unseen) data.
- This is the issue of so called generalization error.
- Though training data is *iid* according to same distribution, the error on training set is not necessarily a good indicator of the generalization error.



Now, let us finally come back to the issue of how do I assess a learnt model. We use a training set to learn a model right a classifier, but we actually interested in how well the classifier does on unseen data. Its performance on the training set is of no consequence even thou all my learning algorithms are essentially optimizing based on the error on the training set. We consult this enough number of time I in my first class I have been discussing this is the issue of generalization error how well nurse a classifier learnt from some data generalize, how well will it do on unseen data.

Training data is of course, also I had, but even intuitively the error and training set is not a good indicator the generalization error. Because I am picking my classifier to do well on the training data, so obviously it will do well on training data. Only some classifiers that does well on training data is what I will pickup. Hence the error on the training set, which is called the training error is not necessarily good indicator of the generalization error.

(Refer Slide Time: 52:24)



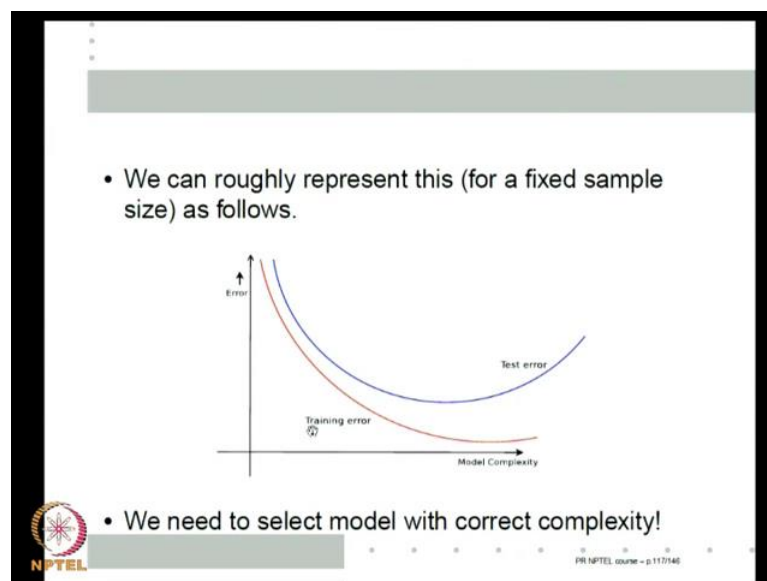
- The error on training set is the empirical risk. It is not necessarily close to the true risk, which is the expected error on new or test data.
- We saw that for empirical risk to be close to true risk, we need large number of examples (relative to the model complexity).
- If the model class is very 'complex', then we may overfit – that is, the training error is small but the test error is large.
- For example, using too many hidden nodes in a neural network results in overfitting.

PR NPTEL course - p114746

Of course, even in a stochastic (()) is it the error in training set is the empirical risk we already know empirical risk is not necessarily close to the true risk, which is the expected error on new test data. Now my expected error on new test data if I take data randomly is nothing but the true risk where the training set error is the empirical risk. Now, for empirical risk to be close to the true risk we need large number of examples relative to the model complexity.

The model is very complex then we may over fit that is that is basically when the training error is small, but the test error is large. Basically if the model is complex I may be able to find functions within the model, which can do very well on the training data, but you know they are really not learning anything. Now, this the phenomena that always is called the over fitting that is I get very low training errors when I see new data the errors will be large. One example is when I use too many hidden nodes in neural network.

(Refer Slide Time: 53:28)



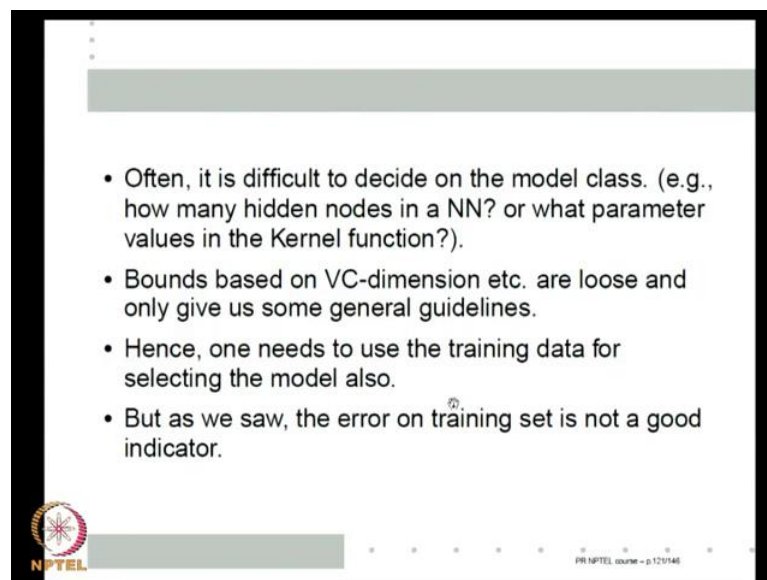
So, we can roughly represented like this, suppose we have a fixed samples size and we keep learning to the same algorithm with in more and more complex model classes. What can model complexity mean? Suppose I am suing neural networks with more and more hidden nodes given that I mean given sufficient complex in the problem what one can except to the training initially if I choose models of very low complexity. Training error is training error is the red curve the training error is large that the error on training set but, as I keep increasing the model complexity.

I can keep reducing my training error, so for example, if I am using r b f networks once I reached the number of hidden nodes, which is same as the training sample I can get 0 errors. So, while the shape of the curve specific shape of the curve is not important, the general shape of the curve that at low complexity I will have high training error, but training error can be monotonically decrease at the complexity decreases. But if I look at the test error the error of the learnt classifier at any given model complexity at a particular number of hidden nodes or whatever at a particular model complexity I learn a classifier and ask how will it do on unseen data. If I track that error that also initially decreases as my model complexity increasing.

It initially decrease, but after some model complexity its starts increasing, because given


your data that is the level of complex models are true (()). So, we need to select the model with the correct complexity only if you run out algorithm at a model complexity level of this. Are we likely to get a good final classifier? Otherwise we may get a bad final classifier here. So, this is a generic behavior of most learning algorithms for the same training sets size as I keep increasing model complexity till some level both training and test headers will decrease, but after that only training error will decrease, but generalization error will increase. So of course, there is no way of knowing, which is the right complexity.

(Refer Slide Time: 55:51)



Is very difficult to decide on what particular model class is correct from a problem. Right model class means how many hidden nodes in neural network what kernel do I use in SVM so on. So, far while we have seen some theory like the V C dimension, so on these are loose bounds they give while they give you general guidelines they cannot give you any specifics. Hence one needs to use the training data for doing model selection also deciding, which model class to use? How do I do this? But we saw training error is not a good indicator for the model class. I can keep increasing the complexity and keep reducing my training error. So, if we want to use the use the use the data that we have then you have to use it slightly differently.

(Refer Slide Time: 56:42)




- If we have sufficient data, then we can use it for both model selection and model estimation.
- We divide the data into Training set and Validation set.
- We select some model class, use training set to learn the classifier.
- Then find the error of the learnt model on the validation set.
- We do it for different model classes and choose the one that has least error on validation set.
- This allows us, e.g., to learn best number of hidden nodes in a neural network model.

PR NPTEL course - p127148

If we have sufficient data we can do this. We can do both model selection and model estimation this is how we do it, we divide the data into two parts; a training set and a validation set. We select some model class specific class on neural networks specific a kernel function with repressive parameters and so on. Then use the training set to learn the classifier whatever best classifier you can get within that model. We find the error of the learnt model on the validation set not on the training set, but the other training data that we have say called the validation set.

Now, I do it for different model class, so I start with let say a three hidden node network. I learn using my training data and the final network its error I measure on the validation set. Then I do it for four hidden nodes with the same training set why once again learn the best four hidden node network. Now, calculate its error on the validation set, now I can plot the errors on the validation set of models are different complexity. So, if you do this for models are different complexity we can chose the one that has the least error on the validation set. So, this allows us for example to use the training data to also learn let say the number of hidden nodes in neural network.

(Refer Slide Time: 58:09)



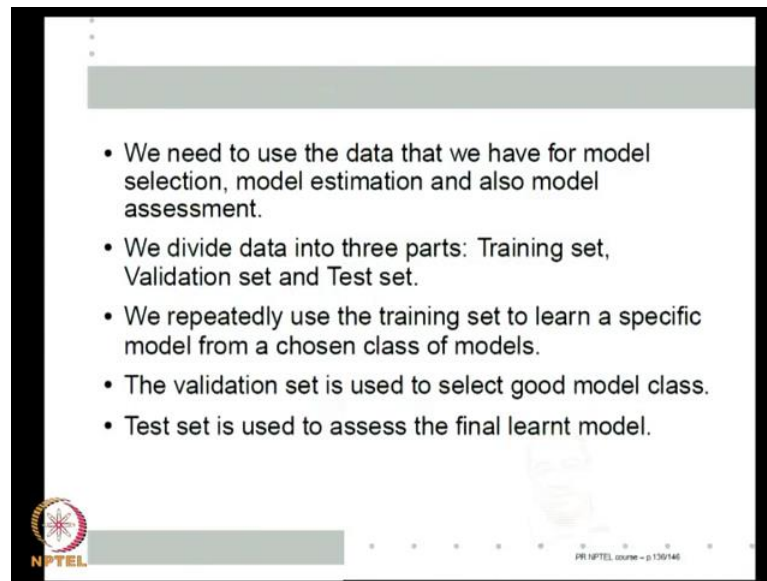
- Normally, we would also like to have an estimate of the generalization error (true risk) of the final learnt model.
- This is called model assessment.
- Using the validation data for this results in overly optimistic estimate.
- Hence one needs separate test data for final assessment.

PR NPTEL course - p131148

So, we divide the data we use one part as for training one part for validation the training is use for model estimation the validation is use for model selection. Normally, we would also like to have an estimate of the final generalization error of the final learnt model after doing all this I decide that five hidden nodes seem to the best and then learn the final five hidden node network. Now, if somebody ask me how well do thing perform what will I tell him, of course I can find, this is called model assessment.

I can I can tell him that it is error, so my final error I can say whatever if the error I got in the validation set may be my good estimate for generalization error. What is overly optimistic? I pick that particular network, because it is doing well learn validation set. So, by definition you know that network is tune to do well on the validation set so what one does is that we have separate test data also for final assessment.

(Refer Slide Time: 59:01)



- We need to use the data that we have for model selection, model estimation and also model assessment.
- 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.

NPTEL

PR NPTEL course - p130146

So, what does all this mean? We need to use the data that we have for model selection, model estimation and model assessment. We divide the data into three parts; training set validation set and test set. We repeatedly use the training set to learn specific model from a chosen class of models, the validation set is used to select good model class, and test set is used to assess the final learnt model. So this is how one does model assessment. Next class, we look at what to do. See of to do all this, I need lot of data I may or may not have enough data. So, we will look at some generic practices people use for doing model assessment in the next class.