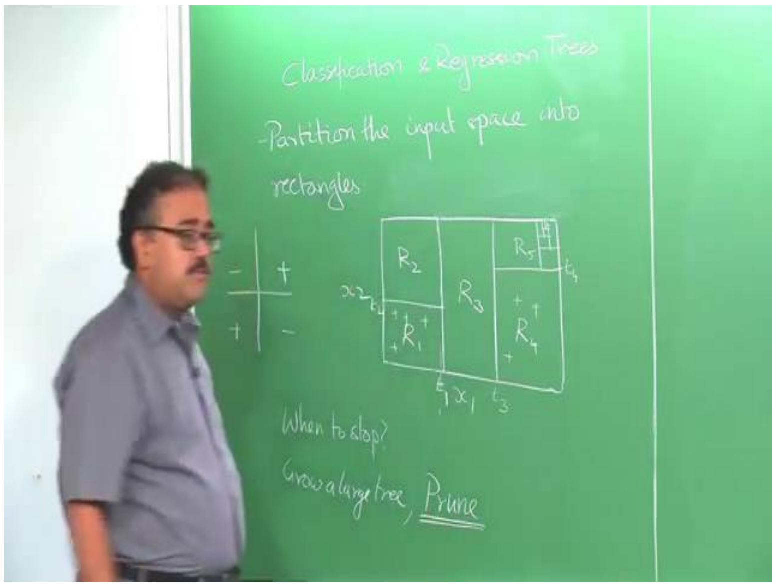
Introduction to Data Analytics

Prof. Nandan Sudarsanam and
Prof. B. Ravindran
Department of Management Studies and
Department of Computer Science and Engineering
Indian Institute of Technology, Madras

Module – 05
Lecture – 26
Classification and Regression Trees

Hi, so in this module will continue looking at Classification and Regression Trees.

(Refer Slide Time: 00:11)


Classification & Regression Trees
-Partition the input space into rectangles
R2
R3
R5
R1
R4
x2
t2
t1 x1 t3
t4
When to stop?
Grow a large tree, Prune

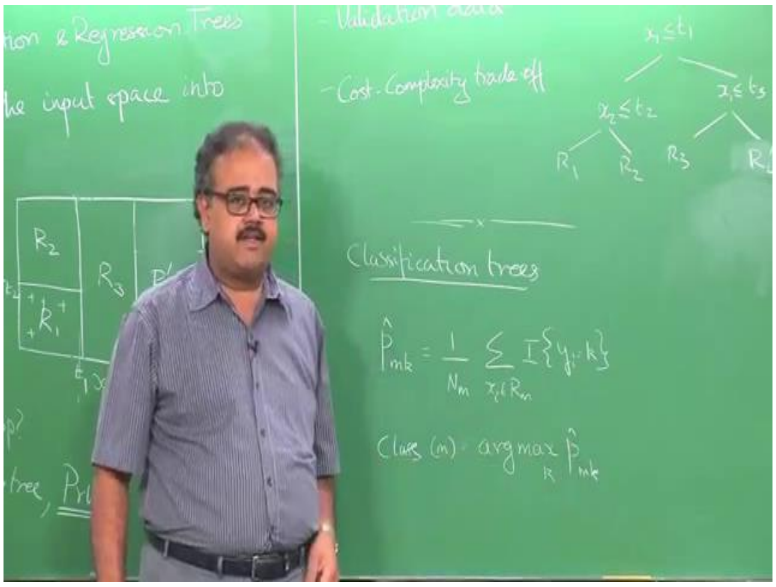
So, in the previous module we looked at how to build the tree, but we never looked at when to stop. So, if we continue growing a very large tree, so we might end up making a lot of decisions and essentially end up specializing your tree to individual data points. So, it will lead to over fitting of your tree to the data and that is not a good place to be in. On the other hand, if you stop growing your tree to soon, you might be sought on finding interesting patterns in the data.

So, for example, a very famous XOR case, so there is no single attribute on which you can split and get any kind of improvement in your performance, because whether you split on x 1 or whether you split on x 2. So, half your data is going to be positive, the other half is going to be negative. So, you really cannot hope to get any improvement by only splitting on one attribute. So, you just kind of stop there saying that hey there are no one individual attribute that gives me an improvement in performance and therefore let me stop.

So, you might lose out on interesting patterns if you are going to stop too early. So, you cannot leave a tree that is very, very specialized to small data points and you cannot stop early as well. So, what we do? So, you typically grow the tree until you come to a point, where there are few data points in the leaf. So, the leaf here would correspond to region. So, I keep going the tree until I come to a point where each region has only a few data points, typically you would like to have bought four or five data points at least per region.

So, for using some of the standard tools say something like weka. So, weka actually stops by default, when the number of data points in a region go down to two. So, it will stop only at that point, that is still a significant over fretting. So, you grow a large tree, where there are very few points per region and then you prune the tree, you basically start trying to collapse the internal nodes in the tree, such that you come up with the better tree. So, we have looked at how you need a validation sets. So, you have a training data and you have a validation data.

(Refer Slide Time: 03:04)



Classification & Regression Trees
the input space into
- Validation data
- Cost-Complexity trade off
$x_1 \le t_1$
$x_2 \le t_2$
$x_1 \le t_3$
R_1
R_2
R_3
Classification trees
$\hat{p}_{mk} = \frac{1}{N_m} \sum_{x_i \in R_m} I\{y_i = k\}$
Class (m) = $\arg\max_k \hat{p}_{mk}$

So, one way of looking at pruning which is to say that, I will look at the performance of the tree on the validation data. So, look at the performance of the tree on the validation data and then, I will start collapsing some of the internal nodes of the tree. So, what do I mean by that? Suppose that I have a tree that let us think of the tree that we had earlier. So, this is the tree that we had earlier. So, one way of pruning something like this could be to say that, I am going to collapse one of the internal nodes and replace it with a single region.

So, there I have pruned the tree, so this corresponds in the picture, this corresponds to doing something like, so that is our four prime. So, now, I can look at the performance of this pruned tree on the validation data versus the performance of the original tree on the validation data. So, now, if there is not a significant dip in the performance, when I have removed one of the internal nodes, then I will keep this new tree.

So, there are essentially tells me that whatever I did in terms of the previous test on t_4 or something that was peculiar to the training data. So, now, I look at it in the validation set, where you really does not look like I need to make that split. Or on the other hand, there might be a slight decrease in performance, when I go from the original tree to the prune tree. Now, the question you have to ask yourself is, given the fact that I have reduced the size of the tree is it to pay the penalty in terms of the slight reduction in performance.

So, there is trade off, so there is this what would call the cost complexity trade off. So, that is the cost that you incur in terms of the misclassification that or the misprediction in this case in that you are going to make in terms of the reduced reduction in the number of regions. And the complexity is essentially the number of test that you have to perform on the size of the decision tree that you are operating with. So, this is the cost complexity trade off and depending on which side of the cost complexity trade off that you want to be on you can get more complex or less complexities.

So, that many ways in which this kind of pruning techniques can be implemented and one simple way which is essentially to choose a validation set and then have a tradeoff between the prediction error of the pruned tree in the validation set verses some measure of complexity of the tree. So, one measure of complexity of the tree is the number of internal nodes that you have in the tree. So, you can basically trade off the two and try to come up with a good smaller tree.

So, far we have been talking about regression trees and we looked at how you would split regions and once having split the found the regions how do you fit, what is the best prediction that you have to make in each of those regions and how would be find the regions, we adopted a greedy approach by picking once variable at a time and then trying to find out which is the best point to split the variable.

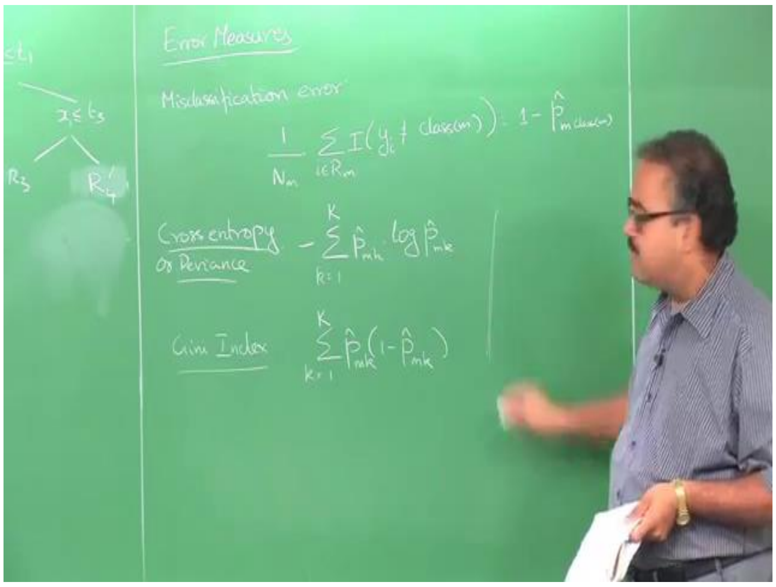
And we looked at the squared prediction error and essentially now if you want to look at classification trees, you only think we have to ask our self is, what is the appropriate measure that we will have to look at in the greedy such procedure. So, once we have decide on what the appropriate measure is, so the rest of the framework that we need to build classification trees are already in place. So, we can do the greedy algorithm and then once we know, what is the appropriate measure that we are going to use we can optimize that measure in the greedy algorithm and we can use the same setup that we have for pruning.

So, we have a validation data and then we have a cost complexity trade off, where the cost instead of being measured by this squared error is going to be measured by whatever metric we choose for a classification. So, I am going to say that I will build the classification tree where the prediction. So, which we will called as $\hat{p}$, so $\hat{p}$ here is the probability that a data point in region m belongs to class k the data point in region m. So, one of these regions, what is the probability that this data point belongs to class k. So, I am going to look at all the data point that is fall in the region R_m.

So, I am going to look at all the data points if fall in the region m and look at whether their class labels in the training data, whether it belongs to class k. So, if a point x_i in region m, if the label is k then this will be 1. So, the summation is essentially going to count the number of data points in region m that belongs to class one or number of data points in region one that belong to class two and so on and so forth this expression is for class k and then I am going to divide it by the total number of points that fall in region m.

So, N_m is the total number of points in region m. So, this is essentially gives me an empirical estimate of the probability of a data point falling in region m belonging to class k and if I am interested in actually returning that class label then... So, the class in region m is going to be the class that I will assign to data points in region is essentially that k that gives me the maximum $\hat{p}$ value that make sense.

(Refer Slide Time: 10:32)


Error Measures
Misclassification error
$\frac{1}{N_m}\sum_{i\in R_m} I(y_i \neq class(m)) = 1-\hat{p}_{m\,class(m)}$
Cross entropy or Deviance
$-\sum_{k=1}^{K}\hat{p}_{mk}\log\hat{p}_{mk}$
Gini Index
$\sum_{k=1}^{K}\hat{p}_{mk}(1-\hat{p}_{mk})$
$x_2 \leq t_3$
R_3
R_4

So, you look at the different error measures that we can use. So, what is the most popular error measure in classification or which is the most appropriate error measure in classification is a simple thing called a misclassification error, the misclassification error is given by... So, the number of times the actual label does not match the prediction that we make by our classifiers. So, class m is the prediction made by our classifier in the m^th region. So, for all the data points in the m^th region whenever the class label the true class label does not match the predictor class label.

So, I am going to sum it up divided by the total number of data points in that region. So, this gives me the misclassification error this for a specific region and I can do this over all regions and that gives me the total misclassification error. So, if you think about it this is essentially... So, the number of times this is should have been correct is given by $\hat{p}_m$ and the class that I am going to predict. So, this is the given this gives me the maximum from the previous expression that we had. So, the total misclassification error will be 1 - $\hat{p}_m$. So, I can use this as my error measure.

So, once I estimate $\hat{p}$ the once I estimate the arg max over k of $\hat{p}_{mk}$ then 1 minus that gives me the misclassification error for that region and I can sum this over all the regions. Whether there are a couple of slightly more common error measures that are used especially in the decision tree literature. So, one of these measures called cross entropy or deviance that leads us to something called information gained measure for looking at the splitting criteria.

So, the cross entropy is given by... So, this is $\hat{p}_{mk}$ is the label distribution that we have inferred is a label distribution that we have inferred from the data that was given to us, but we stop and think about it, if you have sufficient training data. So, that $\hat{p}_{mk}$ is actually the true output label distribution as well. So, this cross entropy term actually gives us in some sense the amount of information that we need to encode the true label given that you are in the region m. So, this gives raise to this measure called information gain that tells us that if I have not split into the following set of regions.

How much information would I need to represent the labels of the data verses having split into all these regions? How much less information do I need? If I do not know anything about where the data point lies on this entire plane then I have some amount of uncertainty about what the label is. Once I know that the data points lies in R_1 or R_2 verses the rest of the plane have less uncertainty about what the label should be and... So, given that when I have split the data into two regions I have gained some information about what the label should be and that leads to this notion of an information gain measure. And so it comes from this cross entropy term and we can use this also as a measure for a splitting the tree in the greedy growing stage.

Last one is a Gini index is given by the expression $\hat{p}_{mk} * (1 - \hat{p}_{mk})$ and it is actually a measure of the disparity in the population of distribution of variables. So, if you think about it, so this is the probability that the label k appears in the distribution in a particular region m and this a probability that the label k does not appear in the distribution m and this expression will be maximum when they are equally likely like the problem there label k appearing and the label k not appearing in the region m or equally likely. So, that is really a bad situation for us to be in.

So, in the some sense having a high value here essentially is a indicator that this is not a great way of splitting the data into different regions. So, when you could use one any of these three measures in terms of going your decision tree and they have their own advantages and disadvantages. And so the cross entropy and the Gini index or more sensitive to note probabilities than misclassification rate, but then quite often we find that these lead to much better trees than using the misclassification rate directly.

So, that brings us to the end of our discussion on classification and regression trees, but there are few points that I would like to make about the use of tree based classifiers or regressors. So, the first thing is I glossed over a little bit the problem of handling discrete valued attributes I said could choose red or not red, but then if you think about discrete valued attribute that has k possible values, you can see that there is a combinatorial many ways of dividing these attributes into two groups. So, you need to have clever way of handling this and then many of the decision tree packages do handle this in a meaningful way.

The other thing which we have to be very aware of when you are dealing with trees is that trees are notoriously unstable. So, what do I mean by unstable is that if there is a very small change in the training data that you give to the trees the tree that you build out of the data could be very different. So, you could delete a few data points and you might end up having a tree that looks very different. So, this is in contrast to some of the earlier thinks that we are looked at like logistic regression or support vector machines, where these things stable to deletion of one or two data points, but decision trees can be notoriously unstable.

So, one way that people get around this problem of instability in decision trees is to make sure that you build many, many different decision trees with slightly different views of the data and then combine the predictions made by all the decision trees into a single tree. So, that way we can actually end make sure that the overall classifier that you build is reasonably stable and another point that we have to look at suppose you have fitting decision trees using regression. So, you can see that for each region we will be outputting a specific constant value at no point or we worried about how the value will change from R_1 to R_2.

So, you are only worried about what is happening in R_1 and we are fitting a value in R_1 and therefore, there is no concern of smoothness in our faiths and therefore, the fits that given by decision tree can be pretty non smooth it is a smoothness is a criteria then we will have to either add some more regularizing factors into decision trees which makes it more complicated or will have to look at other forms of regressions. So, just keep in mind, so decision trees or very powerful classifiers, very powerful regressors they are wonderful in terms of interpretability, but they also come with their own caveats. See you later.