Machine Learning - Lecture 9: Generalized decision-tree learning

Chris Thornton

November 1, 2011

Use information gain

A key question in decision-tree learning is how to identify the 'best split' of data at each point.

Decision-tree methods often use information theory for this.

The frequency with which an output appears can be seen as the probability of it being correct for all members of the group.

Frequency counts then become probability distributions.

To find the best split, we take all the possible splits, and choose the one that gives the biggest reduction in uncertainty.

This is also the split that gives the biggest **gain** of information, i.e., the biggest improvement in the model.

Methods which work this way are said to use an **information-gain** heuristic.

Expected gains

One difficulty, here, is the fact that each split produces several subgroups, each of which has a distinct size and uncertainty.

How do we calculate an uncertainty for the split as a whole?

The solution is to calculate an **expected uncertainty** by weighting the uncertainty for each subgroup by its relative size (i.e., by the probability of it containing an arbitrary datapoint).

We then select the split that produces the biggest reduction in *expected* uncertainty.

Or equivalently, the biggest expected information gain.

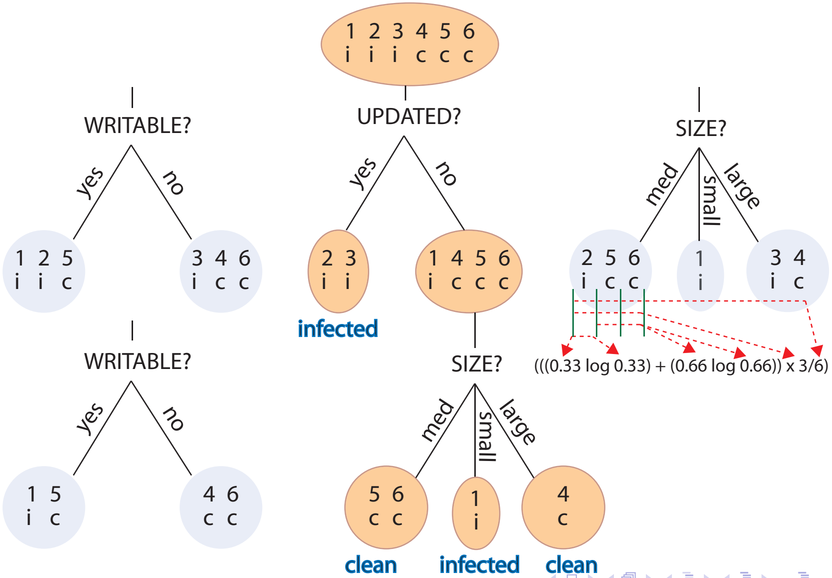
Expected entropy formula

The expected uncertainty for a split is calculated using this formula.

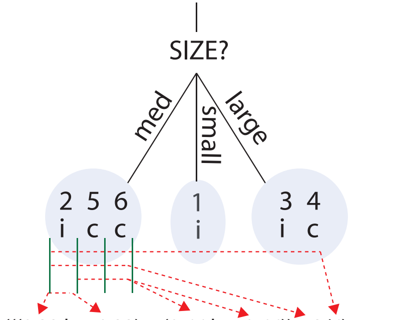
Here, the p_i values are the probabilities of classes in the j'th subgroup, and p_j is the probability of a datapoint appearing in the j'th group.

$$\sum_j p_j \left(- \sum_i p_i \log p_i \right)$$

Computation of entropy values



Expected entropy



(((0.33 log 0.33) + (0.66 log 0.66)) x 3/6)

+ ((1.0 log 1.0) x 1/6)

+ ((0.5 log 0.5) + (0.5 log 0.5)) x 2/6)

= expected entropy for SIZE split

Application to numeric data

The decision-tree algorithm is defined in terms of operations on categorical data.

Can it be used with numeric data?

In the standard algorithm, a split is constructed on a particular variable by creating one branch for each distinct value observed.

If we apply this to a numeric variable, we get one branch for each distinct *number*.

This may be fine with integer data.

Application to real-valued data

The problem is that with real-valued data, each numeric value is likely to be unique.

The result may be a split with a huge number of branches, each of which creates a subset of just one datapoint!

The algorithm terminates immediately, having generated a lookup table in the form of a single branch.

Worst-case generalisation ensues, and there's a good chance the tree will not even classify an unseen example.

Using single thresholds (C4.5 approach)

In order to handle real-valued data, we need splits to be made on the basis of threshold values.

A simple idea is to find the observed value which when treated as a theshold gives the best split.

The resulting tree defines ‘large’ generalisations in which each range of variable values is divided into two parts.

There is no risk of an unseen example being unclassified.

What about multiple thresholds?

Situations can occur where a single threshold seems inappropriate.

Imagine we have a variable representing age. The implicit structure may then be all to do with the four significant age groupings (0-20, 20-40, 40-60, 60+).

To deal with such situations, we really need the algorithm to put in a threshold wherever one is needed.

But how is this to be done?

Ideally, we should consider all possible ways of dividing the variable into subranges, and use the information heuristic to choose the best.

But the combinatorial costs are just too great.

The C4.5 approach

The most widely-used version of the decision-tree algorithm is Ross Quinlan's C4.5, an extended, public-domain version of his earlier ID3 method.

This adds a number of features to the standard decision-tree algorithm.

The C4.5 approach

The most widely-used version of the decision-tree algorithm is Ross Quinlan's C4.5, an extended, public-domain version of his earlier ID3 method.

This adds a number of features to the standard decision-tree algorithm.

- Maximising information encourages splitting on attributes with many values. The result is over-bushy trees. C4.5 corrects for this by maximising **gain ratio** instead. This is the normal gain divided by the gain *solely* attributable to the size of the split.

The C4.5 approach

The most widely-used version of the decision-tree algorithm is Ross Quinlan's C4.5, an extended, public-domain version of his earlier ID3 method.

This adds a number of features to the standard decision-tree algorithm.

- Maximising information encourages splitting on attributes with many values. The result is over-bushy trees. C4.5 corrects for this by maximising **gain ratio** instead. This is the normal gain divided by the gain *solely* attributable to the size of the split.
- C4.5 tries to correct over-complex trees through **pruning**. Working up from the leaf nodes, C4.5 gets rid of any split if it looks like prediction error before the split is as good as prediction after.

The C4.5 approach

The most widely-used version of the decision-tree algorithm is Ross Quinlan's C4.5, an extended, public-domain version of his earlier ID3 method.

This adds a number of features to the standard decision-tree algorithm.

- Maximising information encourages splitting on attributes with many values. The result is over-bushy trees. C4.5 corrects for this by maximising **gain ratio** instead. This is the normal gain divided by the gain *solely* attributable to the size of the split.
- C4.5 tries to correct over-complex trees through **pruning**. Working up from the leaf nodes, C4.5 gets rid of any split if it looks like prediction error before the split is as good as prediction after.
- C4.5 deals with numeric data using single thresholds.

The C4.5 approach

The most widely-used version of the decision-tree algorithm is Ross Quinlan's C4.5, an extended, public-domain version of his earlier ID3 method.

This adds a number of features to the standard decision-tree algorithm.

- Maximising information encourages splitting on attributes with many values. The result is over-bushy trees. C4.5 corrects for this by maximising **gain ratio** instead. This is the normal gain divided by the gain *solely* attributable to the size of the split.
- C4.5 tries to correct over-complex trees through **pruning**. Working up from the leaf nodes, C4.5 gets rid of any split if it looks like prediction error before the split is as good as prediction after.
- C4.5 deals with numeric data using single thresholds.

.

Summary

Summary

- Decision-tree learning applied naïvely to numeric data produces implicit lookup tables in which every branch relates to a unique real value.

Summary

- Decision-tree learning applied naïvely to numeric data produces implicit lookup tables in which every branch relates to a unique real value.
- To get around the problem, we need some way to divide variable values into ranges.

Summary

- Decision-tree learning applied naïvely to numeric data produces implicit lookup tables in which every branch relates to a unique real value.
- To get around the problem, we need some way to divide variable values into ranges.
- Single thresholds obtained using the information heuristic are the simplest approach.

Summary

- Decision-tree learning applied naïvely to numeric data produces implicit lookup tables in which every branch relates to a unique real value.
- To get around the problem, we need some way to divide variable values into ranges.
- Single thresholds obtained using the information heuristic are the simplest approach.
- Use of multiple thresholds is also possible.

Summary

- Decision-tree learning applied naïvely to numeric data produces implicit lookup tables in which every branch relates to a unique real value.
- To get around the problem, we need some way to divide variable values into ranges.
- Single thresholds obtained using the information heuristic are the simplest approach.
- Use of multiple thresholds is also possible.
- This approach is less strongly biased but over-training becomes a significant danger.

Summary

- Decision-tree learning applied naïvely to numeric data produces implicit lookup tables in which every branch relates to a unique real value.
- To get around the problem, we need some way to divide variable values into ranges.
- Single thresholds obtained using the information heuristic are the simplest approach.
- Use of multiple thresholds is also possible.
- This approach is less strongly biased but over-training becomes a significant danger.

Questions

Questions

- Deriving decision-trees for numeric data, there is the option of treating all numeric values as discrete, i.e., proceeding exactly as we do with categorical data. What problems may arise when we use a tree derived this way to classify an unseen example?

Questions

- Deriving decision-trees for numeric data, there is the option of treating all numeric values as discrete, i.e., proceeding exactly as we do with categorical data. What problems may arise when we use a tree derived this way to classify an unseen example?
- How does the problem of over-fitting relate to the problem of lookup tables?

Questions

- Deriving decision-trees for numeric data, there is the option of treating all numeric values as discrete, i.e., proceeding exactly as we do with categorical data. What problems may arise when we use a tree derived this way to classify an unseen example?
- How does the problem of over-fitting relate to the problem of lookup tables?
- Is there any way of detecting the point at which a learning algorithm is starting to over-fit the data?

Questions

- Deriving decision-trees for numeric data, there is the option of treating all numeric values as discrete, i.e., proceeding exactly as we do with categorical data. What problems may arise when we use a tree derived this way to classify an unseen example?
- How does the problem of over-fitting relate to the problem of lookup tables?
- Is there any way of detecting the point at which a learning algorithm is starting to over-fit the data?