Rule induction by bump hunting

1 Introduction

The objective of bump hunting is to find regions in the input (attribute/feature) space with relatively high (low) values for the target variable. The regions are described by simple rules of the type

if: condition-1 and ... and condition-n then: estimated target value.

There are many problems where finding such regions is of considerable practical interest. Often these are problems where a decision maker can in a sense choose or select the values of the input variables so as to optimize the value of the target variable.

Consider, for example, the problem of loan acceptance faced by a bank. Obviously, the bank would prefer to grant loans to people with a low risk of defaulting, and reject applicants with a high risk. It is assumed (and evidence shows) that the risk of defaulting depends on characteristics of the applicant, such as income, age, occupation, and so on. Now the bank may have collected data in the past concerning the characteristics of accepted applicants together with the outcome of the loan (defaulted or not). Such data may be used to find groups of applicants with a low probability of defaulting, which clearly is valuable information in deciding whether or not to accept future applicants.

Other applications are for example, the identification of interesting market segments and industrial process control.

2 Formal statement of the problem

In the introduction we gave an informal statement of the problem. In this section we introduce some notation to give a more precise statement. We have a collection of p input variables, denoted by $x_1, x_2, \ldots, x_p$, and a target variabe denoted by y. The input variables may be numeric (e.g. income of a loan applicant) or categorical (e.g. occupation of the applicant). The target variable may be numeric or a binary class label. In the latter case it is most convenient to code the two classes as 0 and 1, so that the average value of y may be interpreted as the probability that $y = 1$. In the credit scoring problem, we could code a defaulted loan by $y = 0$ and a non-defaulted loan by $y = 1$. In that case the mean of y can be interpreted as an estimate of the probability that an applicant will not default. If y is a class label, but there are more than two classes, then

the problem cannot be handled directly. It is possible however to reformulate it as a collection of binary problems. Just code one class with label 1 and the rest with label 0. Do this for each class in turn.

Let S_j denote the set of possible values (the domain) of x_j $(j = 1, \ldots, p)$. The input space is

$$S = S_1 \times S_2 \times \ldots \times S_p$$

The objective is to find subregions $R \subset S$ for which

$$\bar{y}_R \gg \bar{y},$$

where $\bar{y}$ is the global mean of y and and $\bar{y}_R$ the mean of y in R. The regions we are looking for must have "rectangular" shape, hence we call them boxes.

Let $s_i \subseteq S_i$. We define a box

$$B = s_1 \times s_2 \times \ldots \times s_p$$

where $\mathbf{x} \in B \equiv \bigcap_{j=1}^{p} (x_j \in s_j)$. When $s_i = S_i$, we leave x_i out of the box definition since it may take any value in its domain. Figure 1 shows an example of a box defined on two numeric variables, where $\mathbf{x} \in B \equiv x_1 \in [a, b] \cap x_2 \in [c, d]$.

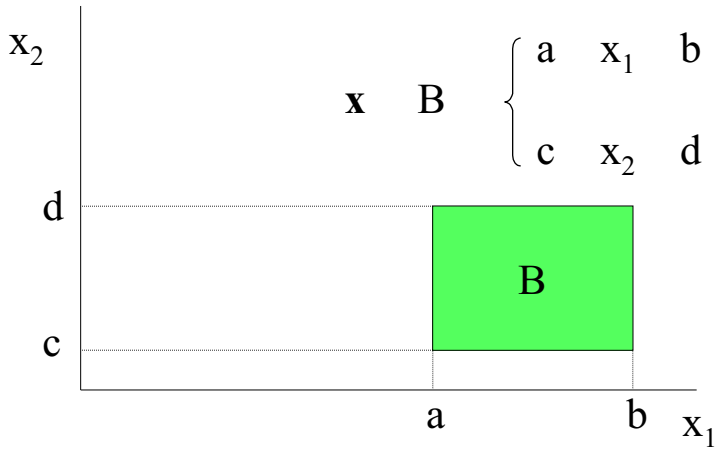

Figure 1: Example of a numeric box

Figure 2 shows an example of a categorical box where $\mathbf{x} \in B \equiv x_1 \in \{a, b\} \cap x_2 \in \{e, g\}$. Boxes may also be defined on combinations of numeric and categorical variables. An important property of a box is its *support*, which is the fraction of points from the data set that fall into the box.

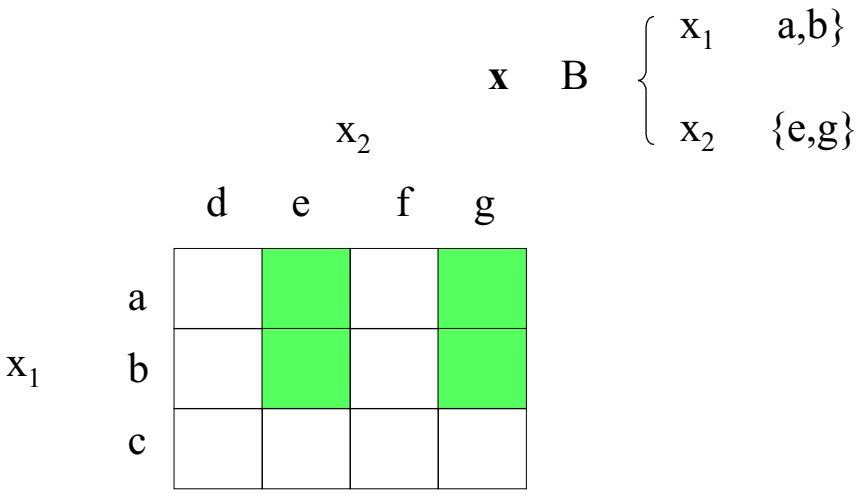

Figure 2: Example of a categorical box

3 Covering

In bump hunting it is customary to follow a so-called covering strategy. This means that the same box construction (rule induction) algorithm is applied sequentially to subsets of the data. The first box, B_1, is constructed on the entire data set. For the construction of the second box, B_2, we remove the data points that fall into B_1. In general, B_K is constructed on $\{y_i, \mathbf{x}_i | \mathbf{x}_i \notin \bigcup_{k=1}^{K-1} B_k\}$. In computing the *support* of B_K we count the data points that fall into B_K (but not into any of the previous boxes) and divide by number of observation of the entire data set (not just the data we used to construct B_K). Box construction continues until there is no box in the remaining data with sufficient support and sufficiently high target mean.

4 Box construction (rule induction)

Given the data (or a subset of the data), the goal is to produce a box B within which the target mean is as large as possible. It is not feasible to simply consider all possible boxes and pick the one with the highest target mean, so usually some kind of heuristic search is performed to find a good box. We discuss two approaches to this problem, and then review them to see where they differ.

4.1 Patient Rule Induction with PRIM

In this section we discuss a bump hunting algorithm called PRIM (for Patient Rule Induction Method), developped by Friedman and Fisher [FF99]. The box construction strategy of PRIM consists of two phases:

1. Patient successive top-down refinement, followed by
2. bottom-up recursive expansion.

4.1.1 Top-down peeling

Begin with a box B that covers all the (remaining) data. At each step a small subbox b within the current box B is removed. The particular subbox $b*$ chosen for removal is the one that yields the largest mean target value within $B - b*$. Each candidate subbox is defined on a single variable. Depending on the type of variable concerned the candidate subboxes are defined as follows:

a) Numeric variable x_j:

$$b_{j-} = \{\mathbf{x}|x_j < x_{j(\alpha)}\}$$
$$b_{j+} = \{\mathbf{x}|x_j > x_{j(1-\alpha)}\}$$

with $x_{j(\alpha)}$ the α-quantile of x_j in the current box, i.e. $P(x_j < x_{j(\alpha)}) = \alpha$.

b) Categorical variable x_j:

$$b_{jm} = \{\mathbf{x}|x_j = s_{jm}\}, \quad s_{jm} \in S_j$$

Typically $\alpha \leq 0.1$, so in each step only a small part of the data points in the current box is peeled of (hence the term *patient* rule induction). For the categorical variables it is obviously more difficult to control the fraction of data that is peeled of, e.g. it may be a binary variable with an equal distribution over the two categories.

Below we give pseudo-code for the top-down peeling algorithm:

Top-down peeling
Repeat
$C(b) \leftarrow$ set of candidates for removal
$b* \leftarrow \arg\max_{b \in C(b)} \bar{y}_{B-b}$
$B \leftarrow B - b*$
$\beta_B \leftarrow$ support of B
Until $\beta_B \leq \beta_0$
Return B

This is a so-called hill-climbing algorithm, because at each step in the search we peel of the single subbox that gives the most improvement of the target mean. The peeling sequence ends when the support of the current box has dropped below the user defined minimum support β_0.

Record	age	married?	own house	income	gender	y
1	22	no	no	28,000	male	0
2	46	no	yes	32,000	female	0
3	24	yes	yes	24,000	male	0
4	25	no	no	27,000	male	0
5	29	yes	yes	32,000	female	0
6	45	yes	yes	30,000	female	1
7	63	yes	yes	58,000	male	1
8	36	yes	no	52,000	male	1
9	23	no	yes	40,000	female	1
10	50	yes	yes	28,000	female	1

Table 1: Bank credit data

Example 1 *We consider once more the credit scoring data given in table 1. Note that the class label "bad" has been coded as 0 and "good" as 1.*

Suppose we are interested in finding groups with low risk, i.e. groups with a high mean for the class attribute y. We set the parameters $\alpha = 1/3$ and $\beta_0 = 0.4$. In the complete table $\bar{y} = 0.5$ and we are looking for boxes with higher mean than this overall average.

We can choose between a number of peeling actions. We start with the discrete variables. Possible peeling actions on discrete variables:

1. *If we peel off observations with* married=no*, the remaining tuples are: 3,5,6,7,8,10. For these tuples $\bar{y} = 4/6 = 2/3$.*

2. *If we peel off observations with* married=yes*, the remaining tuples are: 1,2,4,9. For these tuples $\bar{y} = 1/4$.*

3. *If we peel off observations with* own house=no*, the remaining tuples are: 2,3,5,6,7,9,10. For these tuples $\bar{y} = 4/7$.*

4. *If we peel off observations with* own house=yes*, the remaining tuples are: 1,4,8. For these tuples $\bar{y} = 1/3$.*

5. *If we peel off observations with* gender=female*, the remaining tuples are: 1,3,4,7,8. For these tuples $\bar{y} = 2/5$.*

6. *If we peel off observations with* gender=male*, the remaining tuples are: 2,5,6,9,10. For these tuples $\bar{y} = 3/5$.*

Now we still have to consider the possible peeling actions on the numeric variables. A glance at table 2 shows that peeling off the people with high income is pointless, because we would peel off the good risks. We can peel off people with income below 28 (2 examples). Then we would get $\bar{y} = 5/8$ for the remaining tuples. We cannot peel off all people with income below 30, because then we would peel off 4 tuples out of 10 which is higher than $\alpha = 1/3$.

Income	24	27	28	30	32	40	52	58
y	0	0	0,1	1	0,0	1	1	1

Table 2: Sorted income data with corresponding class labels

Age	22	23	24	25	29	36	45	46	50	63
y	0	1	0	0	0	1	1	0	1	1

Table 3: Sorted age data with corresponding class labels

Finally, we consider the possible peeling actions on age (see table 3). We could peel off everyone older than 45, giving a mean of 3/7 in the remaining tuples. We could also peel off everyone younger than 25, giving a mean of 4/7 in the remaining tuples.

The peeling action that leads to highest mean in the remaining box is to peel off married=no*, with a box mean of 2/3. The support of this box is 0.6, which is larger than* $\beta_0 = 0.4$*, so we continue peeling.*

Of the subsequent peeling possibilities there are two that lead to a box mean of 1, namely own house = yes *and* age < 36*. The former leads to a box with support 0.1, and the latter to a box with support 0.4, so we prefer the latter. The final box is*

if married=yes *and age* ≥ 36 *then* $\bar{y} = 1$ *(support = 0.4)*

4.1.2 Bottom-up pasting

After top-down peeling we have a sequence of boxes, each box in the sequence obtained by peeling of a part of its predecessor in the sequence. At each point in the search process we only look one step ahead: we choose the peel that leads to the best subbox. This means that box boundaries are determined without knowledge of later peels. Therefore the final box can sometimes be improved by readjusting its boundaries. In bottom-up pasting we enlarge te final box recursively until the next paste will cause $\bar{y}$ in the box to decrease.

4.1.3 Cross-validation

After peeling and pasting we still face the problem of which box from the sequence to select. An obvious choice would be to select the box with the highest target mean, but we also have to consider the well-known problem of overfitting: the box found may have a hight target mean on the data used to find it, but this may be due to the peculiarities of the sample. This danger increases as the number of observations in the box becomes smaller. Thus, we run the danger of selecting a small box with a very high target mean on the sample used to construct the sequence, but with a considerably lower target mean when applied to new data.

A well-known solution to this problem is to partition the available data in a training set and a test set. The peeling and pasting is appied to the training data with a small value for β_0. Subsequently, the test data is used to obtain an estimate of the target mean in each succesive box in the sequence. Since the estimates obtained in this manner are unbiased it is now a sensible strategy to simply select the box with the highest target mean on the test data.

4.1.4 Example: family expenditure

We give an illustration of PRIM using a cross section of 1519 households drawn from the 1980-1982 British Family expenditure survey. For each household we have data on the budget share spent on different expense categories (e.g. food, clothing, alcohol, and so on), as well as data on total household expenditure (totexp), total net household income (inc), age of household head (age), and number of children in the household (nk).

With bump hunting we may for example look for groups of households that spend a relatively large share of their budget on food. On average the households in the sample spend about 36% of their budget on food.

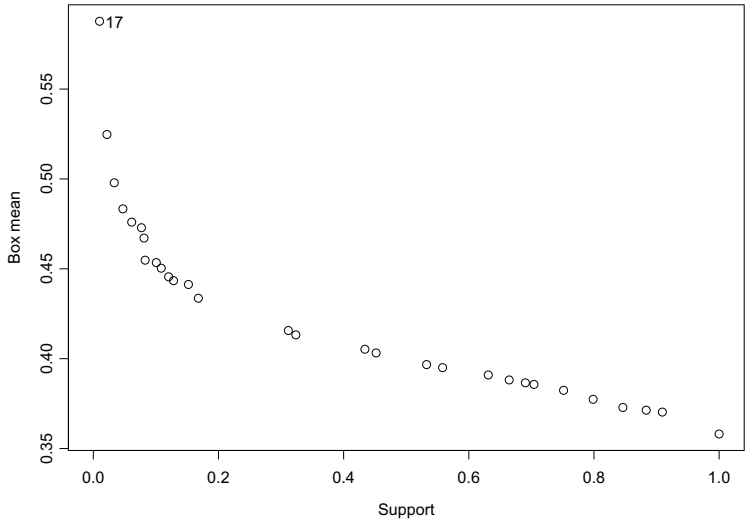

Figure 3: Peeling sequence of rule1

Figure 3 shows support and box mean (on the test set) of the sequence of boxes constructed on the entire training data set. The last (i.e. leftmost) point in the sequence, labeled "17" has the highest estimated target mean, and is therefore selected as the best rule. The rule is:

if totexp < 45 and age > 33 and inc < 135, then wfood $= 58\%$

where wfood denotes the budget share spent on food. This group has support of about 1% (the value of β_0), and the estimate of 58% was calculated on the test set.

Next the data points that match this rule are removed from that training data, and we proceed with the remaining data.

4.2 Rule induction with Data Surveyor

In this section we discuss an alternative bump hunting algorithm called Data Surveyor [HKS96, Sie95]. In the discussion we emphasize those aspects of Data Surveyor that differ substantially from PRIM.

Like in PRIM rule induction begins with a box B that covers all the data. At the first step a number of subboxes $B'_{(i)}$ $(i = 1, \ldots, w)$ within the initial box B are selected. The w subboxes are chosen as follows:

1. It is required that the mean value of the target variable in the subbox is *significantly* higher than the mean in the current box. When the respective confidence intervals for the means in the boxes are non-overlapping the difference is considered to be significant.

2. It is required that the subbox has support of at least β_0.

3. From the subboxes with a significantly higher mean than their "parent" box, and enough support, the w boxes with the largest target mean are chosen.

After the first step in the search we have (at most) w boxes with which the search continues. In the second step we apply the same procedure to find the best subboxes of those w boxes, and so on. Notice that at each level in the search we only consider a *total* of w subboxes of the boxes at the previous level, rather than the w best subboxes of *each* box at the previous level. In the latter case, the search process would explode very rapidly.

Like in PRIM each eligible sub-box is defined on a single variable, but in a different manner:

a) Numeric variable x_j:

$$B'_{jcd} = \{\mathbf{x} \in B | x_j \in [c, d]\}, \quad c, d \in S_j \text{ and } c < d.$$

b) Categorical variable x_j:

$$B'_{jm} = \{\mathbf{x} \in B | x_j = s_{jm}\}, \quad s_{jm} \in S_j.$$

Notice that the way the subboxes are constructed is potentially more greedy than in PRIM, in the sense that it may lead to a more rapid fragmentation of the data. Rather than peeling of a small part of the numeric variables, the algorithm goes directly for intervals with the highest target mean, provided of course that they have the minimum cover β_0 and the mean differs significantly from the current box. When in PRIM the value of α is set for example to 0.05,

then at each step only 5% of the data in the current box is removed. In Data Surveyor the best interval $[c, d]$ for x_j may lead to the removal of 50% or even more at early levels in the search. Likewise, for categorical variables PRIM peels of one value, whereas Data Surveyor selects one value, which clearly leads to faster fragmentation.

Below we give the pseudo-code for the Data Surveyor box induction algorithm. The parameter w determines the width of the beam search, d determines the depth of the search.

Beam Search
Beamset $\leftarrow$ {initial box}
Repeat
  all-subboxes $\leftarrow \emptyset$
  For each box B_i in Beamset do
    $C(B_i) \leftarrow$ set of candidate subboxes of B_i
    all-subboxes $\leftarrow$ all-subboxes $\cup\, C(B_i)$
  Beamset $\leftarrow$ best w subboxes from all-subboxes
Until no improvement possible or depth $= d$
Return Beamset

The important difference with PRIM regarding the search strategy is that PRIM uses a hill-climber, i.e. it only considers the *best* peeling action on the current box. The Data Surveying search algorithm on the other hand employs a beam search, i.e. at each level in the search the best w subboxes are considered. To put it differently, PRIM employs a beam search with $w = 1$. The advantage of taking $w > 1$ is that boxes that are suboptimal at the first level in the search may turn out to lead to better boxes at higher levels in the search process.

Another important difference between the two algorithms is the way in which the subboxes are constructed. In PRIM the subboxes are obtained by peeling off a typically small part of the current box. This strategy has the effect that a peeling sequence can become relatively long before the amount of data in the current box becomes too small. This means that the sequence has the opportunity to recover from unfortunate choices at early stages of the search.

Figure 4 shows the result of a beam search by Data Surveyor, with both the beam width and depth set to three, and minimum support set to 1%. The dataset contains data from car insurance policies where the target value indicates whether the insurant claimed or not. We are interested in high risk groups so we look for subgroups in the data that have an above average tendency to claim. The leftmost node in the figure represents the entire data set, which has exactly 50% claimers and 50% non-claimers (the data was selected to achieve this balance). The confidence interval for the probability of claiming is indicated between square brackets. The number of data points in a box is given next to the confidence interval. Note that at depth 3 in the search we only have two boxes, even though the beam width was set to three. This is simply because there was no other group with a non-overlapping confidence interval and support

higher than 1%. Also note the rapid fragmentation of the data: the best group at level 1 in the search (age in [19,24]) only contains about 14% of the data.

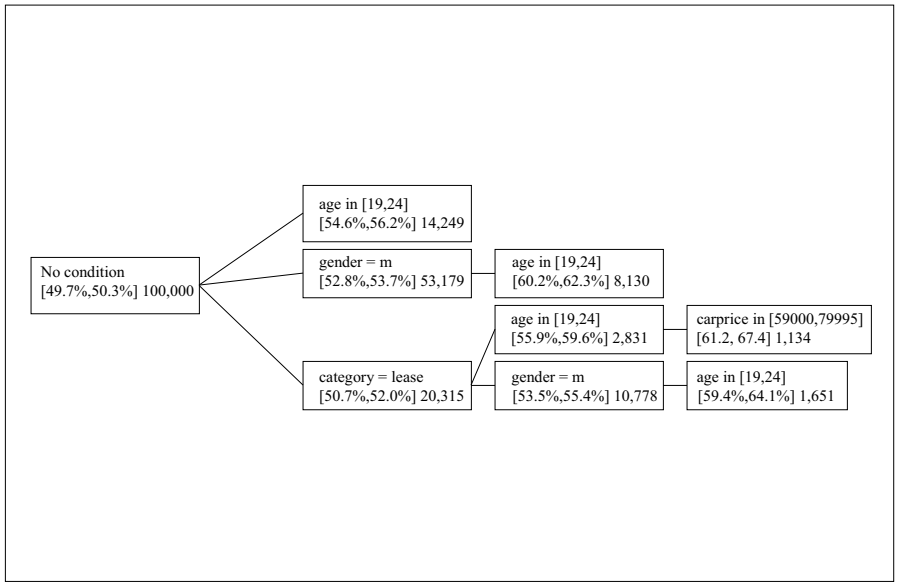

Figure 4: Example of beam search in Data Surveyor: beam width = 3, depth = 3 and minimum support = 1%

Note also that Data Surveyor uses a stopping rule: a node isn't expanded any further if we can find no subgroup with a significantly higher target mean. We saw that such a strategy is suboptimal for classification trees. PRIM on the other hand just continues peeling until the support of the current box becomes too small. Then by default the box that performs best on the test set is selected.

5 Diagnostics in PRIM

PRIM offers a number of tools to post-process or inspect the results of rule induction. We give a short discussion of three of them

1. Redundant variables.
2. Interbox dissimilarity.
3. Relative frequency ratio plots.

	Remove variable (+below)	
Definition	Mean	Support
totexp < 65.00	0.3558	0.9901
age > 36.50	0.4240	0.1578
$105.0 <$ inc < 135.0	0.4377	0.0375

Table 4: Analysis of the effect of removing variables

5.1 Redundant variables

After a box has been selected we can look at ways of simplifying the box by removing variables from its definition. As an example we look at the second rule constructed to find households that spend a relatively large part of their income on food. The rule is:

> if totexp < 65 and age > 36 and $105 <$ inc < 135, then wfood = 48%

where wfood denotes the budget share spent on food. This group has support of about 0.7%, and the estimate of 48% was calculated on the test set.

For each variable we consider the decrease in box mean when it is removed from the definition. The variable yielding the smallest decrease in target mean is provisionally removed, and we do the same thing for the remaining variables. In table 4 we see the result of such an analysis for the rule above. Apparently, the removal of income from the box definition yields the smallest decrease in target mean. From table 4 we can read that when it's dropped, the mean in the box becomes about 44%, and the support of the box goes up to about 4%. If we drop both income and age, the box mean decreases to about 42%, and the box support goes up to approximately 16%. Finally, if we drop all variables we simply end up with all data points not covered by the first rule. In that case the box mean becomes approximately 36%, and the support about 99%. The choice whether or not to remove any variables from the box definition is entirely up to the user.

5.2 Dissimilarity of boxes

The covering procedure produces a sequence of boxes that cover a subregion of the attribute space. These boxes can overlap or be disjoint, be close or far apart, depending on the nature of the target function. For example, if the target function has a single prominent mode, the covering procedure might produce a sequence of nested boxes, where each box in the sequence completely covers all of those induced before it. Alternatively, successive boxes might cover different "shoulders" of that mode, and produce a "cluster" of closely related boxes. If there are several different prominent modes, the boxes might divide into corresponding groups of nested/clustered sequences. By inspecting the

dissimilarities of pairs of boxes in the sequence, we can learn about the modal structure of the target function.

The dissimilarity $D(B_k, B_l)$ between boxes B_k and B_l is defined as the difference between the support of the smallest box $B_{k,l}$ that covers both of them, and the support of their union

$$D(B_k, B_l) = \beta(B_{k,l}) - \beta(B_k \cup B_l)$$

Here the support $\beta(B)$ of a box B, is defined as the fraction of observations *in the entire dataset* that it covers. The smallest box $B_{k,l}$ that covers both B_k and B_l is defined as follows:

1. If x_j is categorical and appears in the box definition of B_k with constraint $x_j \in s_{jk}$ and in the box definition of B_l with constraint $x_j \in s_{jl}$, then it appears in the box definition of $B_{k,l}$ with constraint $x_j \in s_{jk} \cup s_{jl}$.

2. If x_j is numeric and appears in the box definition of B_k with constraint $x_j \in [t^-_{jk}, t^+_{jk}]$, and in the box definition of B_l with constraint $x_j \in [t^-_{jl}, t^+_{jl}]$, then it appears in the box definition of $B_{k,l}$ with constraint $x_j \in [\min(t^-_{jk}, t^-_{jl}), \max(t^+_{jk}, t^+_{jl})]$.

Example 2 *If we have boxes* $B_1 = x_1 \in \{a, c\} \wedge x_3 \in [6, 11] \wedge x_{12} \in [80, 106]$ *and* $B_2 = x_1 \in \{b, c\} \wedge x_3 \in [15, 23] \wedge x_5 \in [2, 4]$ *then* $B_{1,2} = x_1 \in \{a, b, c\} \wedge x_3 \in [6, 23]$.

The dissimilarity between boxes assumes values in the interval $[0, 1)$. Nested boxes will have zero dissimilarity, as will "adjacent" boxes that have contiguous intervals on one numerical variable, and identical subsets on all other variables.

Notice that two boxes can be defined in terms of very different sets of variables and still be quite similar, if the two sets are highly correlated. A simple measure for this kind of similarity is box overlap

$$O(B_k, B_l) = \frac{\beta(B_k \cap B_l)}{\beta(B_k \cup B_l)}$$

Question 1 *Suppose two boxes cover disjoint sets of observations, what can you say about box overlap? Suppose furthermore that their box definitions have no variables in common. What can you say about their dissimilarity?*

5.3 Relative frequency ratio plots

From the perspective of interpretation, it is important to be aware of possible alternative definitions for each induced box. Other descriptions, based on different attributes, may lead to very similar boxes in terms of the actual data points that are covered. This is caused by correlation between the attributes.

We can compare the relative frequency distribution of values of each variable x_j within the box to that over the entire data sample by looking at their ratio

$$r_{jk} = \frac{p_j(x_j | \mathbf{x} \in B_k)}{p_j(x_j)}$$

A uniform distribution for $r_{jk}(x_j)$ implies that x_j is totally irrelevant for the definition of B_k; the relative frequency of its values is the same inside and outside the box. A highly peaked distribution for $r_{jk}(x_j)$ means that x_j is highly relevant for the definition of B_k whether or not it is one of the defining variables.

6 Using the results of bump hunting

We have discussed the *covering* strategy of bump hunting: find the best rule (box) on the complete dataset, remove the datapoints that fall into that box, and proceed to find the best rule on the remaining data. The end result is a list of rules where each rule specifies a box on the input variables and an estimated target mean in that box.

The most obvious use of such a list of rules is for the prediction of the target value of a new case with unknown target, or the selection of cases with high predicted target value. For example in credit scoring we may want to use the list of rules to select applicants with a low probability of defaulting. When a new applicant arrives, we collect the relevant data on the input variables, and proceed as follows. We look whether the applicant matches the first rule in the list. If so we look at the associated probability of defaulting, which is presumably very low for the first rule. The applicant would be accepted in that case. If he or she does not match the first rule, we proceed to the second rule, and so on. If the applicant does not match any of the rules, or we consider the probability of defaulting associated with the first matching rule to high, the applicant is rejected.

Of course this is only one example of how the results can be used; for other applications they may be used in a different way. We always have to be aware however that we are dealing with an ordered list of rules, and not with a collection of independent rules. The target mean in B_K was estimated on a dataset with the data points falling in $B_1, \ldots, B_{K-1}$ removed. The target mean in B_K computed on the *entire data set* is likely to be different, unless B_K does not overlap with $B_1, \ldots, B_{K-1}$. When there is overlap however, the estimated target mean is probably too low, since the target means in $B_1, \ldots, B_{K-1}$ are typically (though not necessarily) all higher than the mean in B_K.

In the past it has been quite customary to use different types of algorithms, e.g. classification or regression trees (like classification trees but with numeric target value), for finding groups in the data with a high value for the target variable. In that case the tree obtained is "post-processed" by selecting the leaf nodes with high target values. Friedman and Fisher [FF99] show, in an experimental comparison of CART and PRIM, that the latter tends to yield better results for this task. The explanation of this experimental observation is that CART fragments the data much faster than PRIM: since CART makes binary trees on average half of the data is removed at each step in the search. This means that it is more difficult to recover from unfortunate greedy steps in the search since we run out of data too fast.

7 Summary

Bump hunting algorithms are very suited when we are interested in finding groups in the data that have a particularly high (or low) value for the target variable. The groups are typically described by the conjunction of a number of simple conditions, each condition based on a single input variable. This has the advantage that the individual rules are easy to interpret.

Because the number of boxes that can be defined using the input variables is very large, we have to employ some heuristic search strategy to find good boxes. PRIM only continues the search with the best subbox of the current box (hill climbing), but the subboxes considered only remove a small amount of data from the current box. This gives PRIM the opportunity the recover from "unfortunate" choices before the algorithm runs out of data. Data Surveyor tends to remove more data from the current box at each step in the search, but it employs a beam search. Therefore it may find better boxes than a hill climber.

Experiments with PRIM suggest that bump hunting algorithms give better performance at the task considered in this chapter than post-processing the output of a tree-based algorithm such as CART.

References

[FF99] J.H. Friedman and N.I. Fisher. Bump-hunting in high-dimensional data. *Statistics and Computing*, 9:123–143, 1999.

[HKS96] M. Holsheimer, M. Kersten, and A. Siebes. Data surveyor: Searching the nuggets in parallel. In U. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and U. Uthurusamy, editors, *Advances in Knowledge Discovery and Data Mining*, pages 447–467. AAAI Press, 1996.

[Sie95] A. Siebes. Data surveying: Foundations of an inductive query language. In U. Fayyad and U. Uthurusamy, editors, *Proceedings of the first international conference on knowledge discovery and data mining*, pages 269–274. AAAI Press, 1995.