Generalized Branch and Bound Algorithm for Feature Subset Selection

M. Venkateswara Kumar and P. Viswanath

Department of Computer Science and Engineering, Indian Institute of Technology–Guwahati, Guwahati-781039, India, {v.mayla, viswanath}@iitg.ernet.in

Abstract. Branch and Bound Algorithm is a good method for feature selection which finds the optimal subset of features of a given cardinality when the criterion function satisfies the monotonicity property. It prunes-off the search space and hence is faster than doing the brute force exhaustive search. To find an optimal feature subset of a different cardinality the method needs to be applied from the beginning. Also the method cannot be used when we do not know the cardinality of the subset we are looking for. This paper presents a generalization over the Branch and Bound Algorithm which first finds optimal subsets of features of varying cardinalities in a single run. Then a method is given to find the best subset of features. The proposed method is experimentally verified using some standard data sets and the results are compared with that of the Branch and Bound Algorithm and with an other related feature subset selection method. The proposed generalized method can be a faster and a suitable one when one do not know the number of features in the best subset of features.

1 Introduction

Feature selection is to select a subset of features which for classification purposes can provide an “optimal” discriminative information. Large number of features can give rise to the “curse of dimensionality” problem which can reduce the performance of the classifier [1]. Also working with a smaller subset of features can reduce the design time and the classification time.

There are two broad approaches to do the feature selection, viz., filter approach and wrapper approach. These are based on how a subset’s quality is assessed. In the filter approach a criterion function like feature-class entropy, mutual information gain, correlation coefficient, etc., is used which is independent of the classifier (where the selected subset of features is intended to be used). Whereas in the wrapper approach the classifier is taken into consideration to evaluate a feature subset’s quality. The classification accuracy is used to measure the quality [3]. Hence the wrapper approach is more time consuming than the filter approach.

The complexity of the feature selection process depends on the cardinality of the given feature set and the cardinality of the required feature subset. In general, this is a combinatorial problem. Let the cardinality of the given feature set be \( n \), and let \( d \) be the cardinality of the feature subset we have to select, then the search space is of size \( \binom{n}{d} \). And if we do not know \( d \) then one has to find the best subset from the set of all
subsets whose size is $2^n$. Hence, it is difficult to do an exhaustive search. To reduce the search time various methods are proposed which are based on relaxing the “optimality” (so, one is satisfied with a sub-optimal solution) condition and by employing several heuristics [2].

Branch and Bound (BB) algorithm has the ability to reduce the search space without losing the optimal solution. It was first adopted by Narendra and Fukunaga [7], and subsequently it has been studied in more detail by others [2] [12]. BB algorithm prunes-off the search space provided the criterion function satisfies a property called the monotonicity property (this is explained in the subsequent section) which can reduce the search space considerably. But in the worst case still the BB algorithm is of exponential time complexity. BB algorithm is also a filter method. It will select the feature subset of the required cardinality without considering the application i.e., the classification task. This paper proposes an algorithm called the generalized branch and bound (GBB) algorithm, which for a given range of cardinality values can find the respective optimal subsets of features. So the output of the GBB algorithm is a set of subsets of features, in contrast to just a single subset of features as found by the BB algorithm. This can be achieved even by the BB method, but by repeating (re-executing) it several times by varying the required cardinality values and hence is a time consuming process. Also in the paper we present a method to choose the best subset of features among the set of subsets of features.

The paper is organized as follows. In Section 2 an introduction to the necessary mathematical preliminaries and a brief overview of the BB algorithm is given along with an improved version of the basic BB algorithm called improved branch and bound algorithm (IBB). Section 3 describes about the proposed generalized branch and bound algorithm (GBB). The results of the experimental studies in order to show the effectiveness of the GBB method are summarised in Section 4. Finally, some of the conclusions are given in Section 5.

2 Preliminaries

The Branch and Bound algorithm reduces the search space if the criterion function $Z$ satisfies the monotonicity property as described below. Let $F = \{f_1, f_2, \ldots, f_n\}$ be the set of features and $2^F$ be set of all subsets of $F$. Let $\mathbb{R}^+$ be the set of non-negative real numbers. Then, $Z : 2^F \to \mathbb{R}^+$ is the criterion function. We say, $Z$ satisfies the monotonicity property, whenever two arbitrary subsets $F_1$ and $F_2$ of $F$ satisfies $F_1 \subseteq F_2$, we have $Z(F_1) \leq Z(F_2)$. Some of the criterion functions which satisfies the monotonicity property are Bhattacharyya Distance, Divergence Measure, Patrick-Fischer Distance, etc., [22].

The selection of $d$ features from the set $F$ of $n$ features by using the BB algorithm is as follows. The algorithm constructs a search tree with root representing the set $F$ and each of the interior nodes representing a proper subset of $F$. Each leaf node of the tree represents a feature subset of cardinality $d$. If a node $p$ represents a feature subset $F'$, then a child $q$ of the parent $p$ represents a feature subset $F' \setminus \{f_i\}$ for some $f_i \in F'$. In this case, the arc between the nodes $p$ and $q$ is labeled by $f_i$. Each node has its associated criterion value. In general, by this process, a feature subset can be represented by more
than one node. For example, by first removing the feature \( f_i \) and then the feature \( f_j \), we get a feature subset. The same subset is obtained by removing first \( f_j \) followed by \( f_i \). To avoid this, the features assigned to an edge at level \( i \) are not considered to construct the tree in the subsequent levels, this is done with the help of a variable \( r \). At any level the value of \( r \) represents the number of available features for tree construction. At \( 0^{th} \) level the value of \( r \) is \( n \). The value of \( r \) is decreased while going from top to bottom of the tree and the value is increased while backtracking. The process of selecting 2 features from a set of 5 features which is represented as \( F = \{1, 2, 3, 4, 5\} \), using the \textit{BB} method with the criterion function \( Z(F) = \sum_{f_i \in F} f_i \) (a simple and trivial criterion satisfying the monotonicity property chosen for the purpose of better explanation. For example \( Z(\{1, 2, 3\}) = 1 + 2 + 3 = 6 \) is demonstrated in Fig. 1. The search tree begins with a root node representing \( F \). It is constructed in a depth-first order, and in Fig. 1 it is built from right to left. The number of child nodes for any node \( p \) in the tree depends upon the \( d \) and \( r \) values and is calculated by using the formula \( (r - (d - k - 1)) \). Here \( k \) represents the level of the node. By using this formula the number of child nodes at level 0 as shown in Fig. 1 is 3. In order to avoid generating duplicate nodes, as mentioned above, the selected features 3 are eliminated from further tree construction in the subsequent levels. So the available features for the tree construction at level 1 are \( 5 - 3 = 2 \) (i.e., \( r = 2 \)). This process of eliminating features continues till the \( r \) value becomes 0 at a leaf node. After reaching a leaf node the process backtracks from the leaf level (say \( i \)) to its parent level (i.e., \( i - 1 \)) if all of the nodes at level \( i \) are evaluated, otherwise the process continues with the exploration of the rightmost node among the available nodes in the corresponding level. While backtracking the removed features are collected and the \( r \) value is increased.

Some of the sub-trees can be pruned-off (i.e., can be avoided from being explored without losing the optimal solution) which can substantially decrease the time to find the best subset of features. Let \( Z^* \) be the largest criterion value among all feature subsets of cardinality \( d \) those are already explored. We call \( Z^* \) as the current-bound. For a node \( p \) which represents a subset \( F' \), if \( Z(F') \leq Z^* \), then the sub-tree starting from the node \( p \) can be pruned-off. Because of the monotonicity property of the criterion function, it is guaranteed that there will not be a node in the sub-tree starting from \( p \) for which the criterion is greater than \( Z^* \). The current-bound \( Z^* \) needs to be updated whenever a leaf node has its criterion greater than \( Z^* \).

In the \textit{BB} method the number of child nodes of a node are fixed. For example let this be \( a \). But among all possible child nodes \( a \) nodes are randomly chosen, and also the order in which these \( a \) nodes are explored is random. If the current-bound \( Z^* \) is a larger one, then many sub-trees can be pruned-off. This can be achieved by carefully choosing the child nodes and the order in which they are explored, and this improved version of the \textit{BB} method is called as the \textit{improved branch and bound (IBB) method}. The \textit{IBB} method orders the child nodes of a node based on a parameter called “criterion value decrease”. As shown in Fig.1 a node towards the left side has many child-nodes when compared to the right side nodes. So pruning-off a node in the left side leads to avoiding more computations compared to pruning-off a node in the right side. Let \( f_1 \) be a feature in the set \( F \), then the “criterion value decrease” corresponding to \( f_1 \) is \( Z(F) - Z(F \setminus \{f_1\}) \). A feature for which the “criterion value decrease” is large is
termed as a “good feature”, and relatively for which it is small is called a “bad feature”. So removing “bad feature” early can mostly lead to a better current-bound.

The IBB method for the example is shown in Fig. 2. Note that, the method, in this example, finds a better current-bound than the BB method (see Fig. 1) quickly and hence the number of pruned-off nodes are larger than the BB method. Algorithm 1 describes the IBB method.

As the search tree is constructed in a depth first manner, the Algorithm uses a stack (stack is a data-structure which maintains a list in such a way that the element entered (pushed) last will be the element retrieved (popped) first === [ ? ] ===.). A node is
popped from the stack to be expanded (means the child-nodes are generated), and its child-nodes are pushed into the stack in a particular order which is explained below.

Each node is represented by a tuple $⟨S, Z_S, \tau, l, r⟩$ where $S$ represents the feature subset corresponding to the node, $Z_S$ represents the criterion value $Z(S)$ associated with the node, $\tau$ represents the features available for tree construction, $l$ represents the level of the node in the tree, and $r$ represents the number of features available for exploring the node.

Sir in this algorithm we are calculating the criterion-decrease for every feature in $current-node.S$, this will lead us to duplicate testing. To avoid this we need to maintain another entry in the node format, which will give us the features available for further tree construction. For this I am adding entry $\tau$ to the node format.

Algorithm 1 Improved Branch and Bound Algorithm

{$F$ be the total set of features from which a subset of cardinality $d$ is required. $|F| = n$. $Z^*$ is the current-bound which is initialized to a least possible criterion value. If the criterion value is always positive then $Z^*$ can be initialized to 0. $F^*$ is the optimal subset that is given as the output. $F^*$ is initialized to emptyset.}

Create the root node with $S = F$, $Z_S = Z(S)$, $\tau = F$, $l = 0$, and $r = n$.

Push the root node into the stack.

while Stack is not empty do

$\text{current-node} =$ Pop a node from the stack.

if $\text{current-node}.Z > Z^*$ then

if $|\text{current-node}.S| = d$ then

$Z^* = \text{current-node}.Z$

$F^* = \text{current-node}.S$

else

$m = current-node.r - (n - d - current-node.l - 1)$ {m is the number of child nodes for the current-node.}

For each feature $f \in current-node.\tau$ find $\text{criterion-decrease}(f) = Z(current-node.S) - Z(current-node.S \setminus \{f\})$

Find $m$ features $f_1, f_2, \ldots, f_m$ from $current-node.\tau$ such that $\text{criterion-decrease}(f_1) \leq \text{criterion-decrease}(f_2) \leq \cdots \leq \text{criterion-decrease}(f_m)$ and there is no feature $f$ in $current-node.\tau$ such that $\text{criterion-decrease}(f) > \text{criterion-decrease}(f_m)$.

for each feature $f_i, i = 1$ to $m$ do

Create a new node $< S', Z', \tau', l', r' >$ where $S' = current-node.S \setminus \{f_i\}$, $Z' = Z(S'), \tau' = current-node.\tau \setminus \{f_1 \cdots f_i\}, l' = current-node.l + 1$, and $r' = current-node.r - i$.

Push the new node into stack.

end for

end if

end if

end while

Output $F^*$.
2.1 Drawbacks of the IBB method

1. The IBB method outputs the optimal feature subset of the given cardinality $d$, but the value of $d$ is normally unknown. The optimal feature subset could be of a different cardinality.

2. The IBB method needs to be re-run if one wants to modify the value of $d$, however small the modification may be.

3 Generalized branch and bound method

In order to overcome the disadvantages of the IBB method, the so-called Generalized Branch and Bound (GBB) method is proposed. The GBB method constructs a search tree, through which feature subsets of any cardinality within a range can be found in single execution. This method also uses the “Backward Elimination” as search direction, “Complete” as search strategy for generating the feature subsets, and independent criterion measures having monotonicity property for evaluating the optimality of the feature subsets. In BB and IBB methods the number of features to be selected $d$ is constant through out the construction of tree.

In the GBB method the search tree is constructed for finding the optimal feature subsets of different cardinality within a range, so the $d$ value is not constant. The number of child nodes($n_k$) for any node at level $k$ in the search tree depends upon the values of $d$ and $r$, and is calculated as $n_k = (r - (n - d - k - 1))$. The tree is explored in depth-first manner and from right to left. The tree is capable of generating all combinations of each cardinality $d$ within the given range.

The GBB method keeps track of a pair $\langle F^*_{\star i}, Z^*_{\star i} \rangle$ for each cardinality within the given range, where $F^*_{\star i}$ represents the feature subset of cardinality $i$ and $Z^*_{\star i}$: represents the criterion function value $Z(F^*_{\star i})$. GBB method finds these pairs in single execution instead of multiple executions taken by IBB method, thus reduction in time. GBB method uses the ordering heuristic similar to IBB method. Any node $p$ in the tree is explored to find the best feature subset of cardinality $d$, in GBB method this is selected as $i$ where $MIN \leq i \leq MAX$ and the criterion function value corresponds to the node $p Z(F') > Z^*_{\star i}$. $Z(F') > Z^*_{\star i}$ means there is a possibility to get best feature subset of cardinality $i$, so the node is explored for finding the feature subset of cardinality $i$. If the criterion value $Z(F') < Z^*_{\star i}$ for all $i$ within the range, then the node is pruned off from further exploration. This process generates all combinations corresponding to each cardinality in the range and prunes off the nodes.

The GBB method is best explained in Fig.3, which shows the search tree for selecting the best feature subsets of cardinality within the range 2-3 from a feature set of cardinality 5. At $0^{th}$ level only root node present, the criterion value corresponding to the total feature set $F Z(F) > Z^*_MAX$, so the root node is explored for finding the optimal feature set of cardinality $MAX$ by setting $d = MAX$ and $r = 5$, by using these two parameters the number of child nodes to the parent node at level 0 is 4. All these child nodes are ordered according to their criterion value decrease and exploration starts from right most node i.e., node corresponding to feature 2. The sub tree corresponding to node 2 is explored as in IBB method. When the leaf reached the best feature set
{3,4,5} of cardinality 3 and its criterion value 12 is stored in $F^*_3$, $Z^*_3$ respectively. After backtracking to the node 3 at level 1, this node is explored for the feature sets of cardinality 2 instead of 3, because the criterion value corresponding to the node 3 at level 1 is 12 which is less than or equal to the criterion value corresponding to the so far best feature set {3,4,5}, this way pruning is done. This process continues till the evaluation of total nodes in the tree. After the total evaluation of the tree we are left with the $(F^*_i, Z^*_i)$ pairs for each cardinality $i$ within the range. For selecting the best feature subset among these pairs, the following procedure is used.

### 3.1 Selection of Optimal Feature set within a Range

For selecting the best feature subset within the range, the following procedure is suggested.

1. Plot a graph by considering the cardinality $i$ of feature subset on one axis and corresponding $Z'$ value on another axis.
   
   $Z'_i = (Z^*_i - \theta \ast i)$ for all $i$ within the range.
   
   where $Z^*_i$ is from $(F^*_i, Z^*_i)$ pair.
   
   $\theta$: Constant Selected as 0.001 $\ast Z^*_{max}$

2. choose the feature subset of cardinality $i$ whose $Z'$ value is maximum.

In the process of comparison between the number of computations taken by GBB method and through multiple execution of IBB method for finding the best feature subsets of cardinality within 2-3 range, it has been observed that GBB method saves 3 computations. The redundant computations during the repeated execution of IBB method are eliminated through GBB method, hence the savage of computations. The savage of computations is more when the range is more and the number of features are more. Thus the time taken by GBB method is far far less than IBB method.

GBB method uses the same notations as IBB method along with MAX and MIN to hold the range of cardinality. The procedure of GBB method is best explained through FIG3 and the algorithm is in Algorithm 2.
Algorithm 2 Generalized Branch and Bound Algorithm

\{ $F$ be the total set of features from which a subset of cardinality $d$ is required. \mid |F'| = n, Z^* is the current-bound which is initialized to a least possible criterion value. If the criterion value is always positive then $Z^*$ can be initialized to 0. $F^*$ is the optimal subset that is given as the output. $F^*$ is initialized to emptyset. \}

Create the root node with $S = F, Z_S = Z(S), \tau = F, l = 0, \text{ and } r = n$.

Push the root node into the stack.

\textbf{while} Stack is not empty \textbf{do}

\quad \textit{current-node} = Pop a node from the stack.

\quad \textbf{for} $i = \text{MAX to MIN}$ \textbf{do}

\quad \quad \textbf{if} $\text{current-node.Z} > Z^*_i$ \textbf{then}

\quad \quad \quad $d = i$, break

\quad \quad \textbf{end if}

\quad \textbf{end for}

\quad \textbf{if} $d \geq \text{MIN}$ and $d \leq \text{MAX}$ \textbf{then}

\quad \quad $Z^*_d = \text{current-node.Z}$

\quad \quad $F^*_d = \text{current-node.S}$

\quad \textbf{else}

\quad \quad $m = \text{current-node.r} - (n - d - \text{current-node.l} - 1)$ \{ $m$ is the number of child nodes for the current-node. \}

\quad \quad \textbf{if} $m > 0$ \textbf{then}

\quad \quad \quad For each feature $f \in \text{current-node.}\tau$ find $\text{criterion-decrease}(f) = Z(\text{current-node.S}) - Z(\text{current-node.S} \setminus \{ f \})$

\quad \quad \quad Find $m$ features $f_1, f_2, \ldots, f_m$ from $\text{current-node.}\tau$ such that $\text{criterion-decrease}(f_1) \leq \text{criterion-decrease}(f_2) \leq \cdots \leq \text{criterion-decrease}(f_m)$ and there is no feature $f$ in $\text{current-node.}\tau$ such that $\text{criterion-decrease}(f) > \text{criterion-decrease}(f_m)$.

\quad \quad \quad \textbf{for} each feature $f_i, i = 1 \text{ to } m$ \textbf{do}

\quad \quad \quad \quad Create a new node $< S', Z', \tau', l', r' >$ where $S' = \text{current-node.S} \setminus \{ f_i \}$, $Z' = Z(S')$, $\tau' = \text{current-node.}\tau \setminus \{ f_1 \cdots f_i \}$, $l' = \text{current-node.l} + 1$, and $r' = \text{current-node.r} - i$.

\quad \quad \quad \quad Push the new node into stack.

\quad \quad \textbf{end for}

\quad \quad \textbf{end if}

\quad \textbf{end if}

\textbf{end while}

\textbf{for} $i = \text{MAX to MIN}$ \textbf{do}

\quad Output $(F^*_i, Z^*_i)$.

\textbf{end for}
The effectiveness of the GBB method is verified by conducting several experiments on some standard data sets obtained from the UCI machine learning repository [21] and are described below.

The three datasets used are

1. **WDBC** mammogram data has 569 instances divided into 2 classes. One is “benign” which contain 357 instances and the other is “malignant” containing 212 instances. The number of features are 30.

2. **WAVEFORM** data set contains 5000 instances, 3 classes. But, for the sake of convenience we have considered only the first 2 classes. class 1 contains 1692 instances and class 2 contains 1693 instances. There are 40 features.

3. **IONOSPHERE** data set contains totally of 351 instances. These 351 instances are divided into two classes. Each instance in the data set is described with help of 34 features. All these 3 data sets contains a class attribute.

The criterion measure used is the well known Bhattacharrya distance [22]. The optimal feature subsets obtained are compared with the feature subsets obtained using the Relief method. The Relief algorithm [13] assigns a “relvance” weight to each feature, which is meant to denote the relevance of the feature to the target concept. The Relief algorithm attempts to find all relevant features.

For each data set two graphs are plotted, graph (a) represents the graph plotted for the optimal feature sets of cardinality within a range. This is drawn by considering the cardinality of the feature set within the range on x-axis and it’s criterion function value on y-axis. graph (b) represents the graph plotted by considering teh feature sets of cardinality within the range and the $Z'$ value which is obtained thorough the procedure mentioned in section 3.

Due to monotonicity property the criterion function value is goes on increasing with respective increase in cardinality, but the amount of increment is goes on decreasing. Beyond certain value the increment in criterion value is less even with the adding of features. graph (b) gives us the cardinality of the feature set beyond which adding of features leads to the less increment in criterion function value.

Fig.4, Fig.5, and Fig.6 represents the results obtained by conducting the experiments on IONSPHERE, WDBC and WAVEFORM data sets respectively. Fig.5(a) represents the graph between the criterion function value $Z$ on Y-axis and cardinality of Feature subsets within range 10-25 on X-axis, Fig.5(b) represents the graph between criterion function value $Z'$ on Y-axis and cardinality of feature set on X-axis, from this graph the cardinality of the optimal feature subset within range 10-25 is of 16 and these are shown in table 1. Fig. 4(a) represents the graph between the criterion function values $Z$ on Y-axis and cardinality of feature subsets within the range 10-30 on X-axis, Fig.4(b) represents the graph between $Z'$ on Y-axis and cardinality of feature subsets within the range 10-30 on X-axis, from this graph the optimal feature subset selected within the range 10-30 is of 14 and these are shown in table 1. Fig.6(a) represents the graph between the criterion function values $Z$ on Y-axis and cardinality of feature subsets...
within the range 10-35 on X-axis, Fig.6(b) represents the graph between Z’ on Y-axis and cardinality of feature sets within range 10-35 on X-axis, from this graph the cardinality of the optimal feature set selected from the optimal feature sets whose cardinality within range 10-35 is of 16 and these are shown in table1. Table1 shows the time comparison between IBB and GBB, and the comparison between the optimal feature subsets obtained with GBB and standard filter algorithm RELIEF [13]. From Table1 it has been observed that GBB is more efficient than IBB.

Table 1. Comparison Table

<table>
<thead>
<tr>
<th>Data Set</th>
<th>Time(GBB) %IBB</th>
<th>Feature Subset(GBB)</th>
<th>Feature Subset(RELIEF)</th>
</tr>
</thead>
<tbody>
<tr>
<td>WDBC</td>
<td>25 % IBB</td>
<td>{1,2,3,5,9,10,11,14,19,21,22,23,25,26,28,29}</td>
<td>{1,2,3,5,6,9,10,14,19,21,22,23,25,26,28,29}</td>
</tr>
<tr>
<td>IONOSPHERE</td>
<td>25 % IBB</td>
<td>{2,4,7,9,10,11,12,13,14,16,18,19,20,22}</td>
<td>{2,4,7,9,10,11,12,13,14,16,18,19,20,32}</td>
</tr>
<tr>
<td>WAVEFORM</td>
<td>20 % IBB</td>
<td>{3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18}</td>
<td>{4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19}</td>
</tr>
</tbody>
</table>

5 CONCLUSION

In this paper we have developed the GBB algorithm inorder to overcome the disadvantages of IBB algorithm which is described in detail in section 2. GBB gives the optimal feature subsets of cardinality within a range as output in single execution, compared to IBB which is to be executed repeatedly for finging the feature sets for each cardinality within the range. From this it has been observed that GBB is more suitable when we want optimal feature set of cardinality within a range. The time complexity and the optimal feature subset obtained with GBB algorithm on different real data sets is compared with standard IBB algorithm and RELIEF algorithm respectively as shown in table1.

References

Fig. 4. Results obtained on IONOSPHERE data set.

Fig. 5. Results obtained on WDBC mammogram data set.

Fig. 6. Results obtained on Waveform data set.