Pattern Recognition Letters 45 (2014) 62–70

Contents lists available at ScienceDirect

Pattern Recognition Letters

journal homepage: www.elsevier.com/locate/patrec

Suboptimal branch and bound algorithms for feature subset selection: A comparative study ☆

Songyot Nakariyakul *

Department of Electrical and Computer Engineering, Thammasat University, 99 Moo 18 Phaholyothin Rd., Khlongluang, Pathumthani 12120, Thailand

ARTICLE INFO

Article history:
Received 3 September 2013
Available online 20 March 2014

Keywords:
Branch and bound algorithm
Dimensionality reduction
Feature selection
Look-ahead search strategy
Suboptimal solution

ABSTRACT

The branch and bound algorithm is an optimal feature selection method that is well-known for its computational efficiency. However, when the dimensionality of the original feature space is large, the computational time of the branch and bound algorithm becomes very excessive. If the optimality of the solution is allowed to be compromised, one can further improve the search speed of the branch and bound algorithm; the look-ahead search strategy can be employed to eliminate many solutions deemed to be suboptimal early in the search. In this paper, a comparative study of the look-ahead scheme in terms of the computational cost and the solution quality on four major branch and bound algorithms is carried out on real data sets. We also explore the use of suboptimal branch and bound algorithms on a high-dimensional data set and compare its performance with other well-known suboptimal feature selection algorithms.

1. Introduction

In many pattern recognition applications, selecting a representative subset of features from an original feature set is an essential pre-processing step. This process is termed feature selection [7,10,11]. Not only does feature selection reduce the cost of collecting irrelevant and redundant features, but it also improves the prediction performance of the predictors. In several data sets such as gene expression data [9] and hyperspectral remote sensing images [20] that contain more than hundreds of features, use of feature selection is mandatory. Thus, developing effective feature selection algorithms has been the focus of much work in statistical pattern recognition.

A feature selection technique usually requires an effective search strategy for finding the best subset of d features from an original set of D features and a criterion function J for assessing the quality of the selected feature subset. In general, a larger value of criterion function J usually indicates a better feature subset. Based on the performance, feature selection techniques in the literature can be categorized into two basic strategies: suboptimal and optimal methods. Suboptimal feature selection techniques select a good subset with a large (not necessary the largest) J value, whereas optimal search algorithms find the best (optimal) subset that yields the largest criterion function J value. Suboptimal methods include sequential search [15,19] and randomized search [8,21] algorithms. These methods are relatively fast and suitable for high-dimensional problems, where the number of original features is over 50. Optimal feature selection algorithms include exhaustive search and the branch and bound (BB) algorithm [16] and its variants. Exhaustive search is a brute-force approach that evaluates a given criterion function J for all possible combinations of d-dimensional subsets and chooses the best subset. Exhaustive search is only practical for low-dimensional problems ($D \leqslant 30$). The BB algorithm, on the other hand, has received much more attention from data mining researchers, since it explores the search space more efficiently than an exhaustive search. The BB algorithm organizes the search to reject many subsets that are guaranteed to be suboptimal without computing their J values. It has been successfully employed in many tasks such as minimizing the total completion time in a job scheduling problem [17], achieving an optimal facility layout in a manufacturing environment [22], and optimizing a reliable optical network design [2].

Although many advanced versions [5,14,24,25] of the BB algorithm have been proposed to improve the search speed of the original BB algorithm, the computational time of the BB algorithm is still very excessive when the dimensionality D of the original feature space reaches a few dozens or more. However, if the bound of the BB algorithm is relaxed and the optimality of the solution is allowed to be compromised, the search speed can be greatly accelerated [16], which makes the BB algorithm feasible

☆ This paper has been recommended for acceptance by J. Yang.

* Tel.: +66 264 3001x3148; fax: +66 2564 3021.
E-mail address: nsongyot@engr.tu.ac.th

http://dx.doi.org/10.1016/j.patrec.2014.03.002
0167-8655/

for high-dimensional problems. In this work, we are interested in studying the performance of the suboptimal BB algorithm.

This paper has two main objectives. The first objective is to investigate the performance of the BB algorithm when the optimality of the algorithm is not retained. We explore a tradeoff between the optimal solutions of selected feature subsets and the reduced execution time. Narendra and Fukunaga [16] were the first group who studied the suboptimal solutions of the BB algorithm. Since then, many improved versions of the BB algorithm [14,24,25] have been proposed, but, to the best of our knowledge, only Nakariyakul [13] carried out a comparative study of the suboptimal solutions of the basic BB algorithm with those of only one of its recent improvements. We analyze the suboptimal solutions of four major BB algorithms in the literature and note the advantages and disadvantages of each version of the BB algorithm using experimental results for real data sets. The second objective of this paper is to explore the possibility of using the suboptimal BB algorithm on high-dimensional data, for which use of the optimal BB algorithm is prohibitive. We compare the performance of a suboptimal BB algorithm with other well-known suboptimal feature selection algorithms on two high-dimensional data sets.

This paper is organized into five sections. Section 2 presents an overview of the basic BB algorithm and prior improvements to it. We review the scheme to incorporate in the branch and bound algorithm for suboptimal solutions in Section 3. Experimental results for four data sets are given in Section 4, and conclusions are advanced in Section 5.

2. The branch and bound algorithm and its improvements

In this section, the BB algorithm and prior improvements to it are discussed. The BB algorithm was first proposed by Narendra and Fukunaga [16] and is referred to as the basic BB algorithm in this paper. It requires that the criterion function J satisfies monotonicity. That is, assume that $\{y_1,y_2,\ldots,y_i\}$ is a subset of $\{y_1,y_2,\ldots,y_j\}$ for $i<j$, the monotonicity property of the criterion function requires that the J values of the two sets, $Y\setminus\{y_1,y_2,\ldots,y_i\}$ and $Y\setminus\{y_1,y_2,\ldots,y_j\}$, fulfill

$$J(Y\setminus\{y_1,y_2,\ldots,y_i\}) \geqslant J(Y\setminus\{y_1,y_2,\ldots,y_j\}), \tag{1}$$

where $Y\setminus\{y_1,y_2,\ldots,y_j\}$ denotes removing the subset of j features $y_1,y_2,\ldots,y_j$ from the feature set Y. Note that a subset with a larger J value is better than one with a smaller J value.

To find the optimal subset of d features from an original set of D features, the basic BB algorithm selects $D-d$ features to be *discarded* by constructing a solution tree. An example of the solution tree corresponding to selecting the best $d=2$ features out of $D=5$ total features is shown in Fig. 1. The numbers in parenthesis next to each node in Fig. 1 refer to the set of features remaining at that node. The root of the tree (the top) corresponds to the set of all five features, and the leaves at the bottom of the tree correspond to all ten possible subsets of two features. The number on a branch refers to the number of the feature removed as the search traverses that branch. The level number denotes the total number of features that have been omitted from the root at that level. The problem is to find the best node (leaf) at the bottom of the tree (level $D-d$) with the largest J value by exploring the tree with the fewest number of J calculations. Here, successor nodes for a given node refer to all nodes below that node. For instance, the successor nodes for node (1, 3, 4, 5) in Fig. 1 are nodes (1, 4, 5) and (1, 3, 5) at level 2 and nodes (1, 5), (1, 4) and (1, 3) at level 3.

To locate the best leaf (solution) of the tree, the basic BB algorithm starts the search at the top of the tree (level 0), and all nodes at level-1 are analyzed. The successor nodes of the node with the largest J value are further explored, and the search continues to the leaves of the tree. The current best subset (leaf) is found with an initial bound B for the criteria function J. This bound B is a lower bound that provides the best J value of d features found so far in the search, i.e.,

$$B = J(Y\setminus\{y_1^*,y_2^*,\ldots,y_{D-d}^*\}), \tag{2}$$

where $\{y_1^*,y_2^*,\ldots,y_{D-d}^*\}$ is the set of $D-d$ features that, when removed from the original set Y of D features, gives the best J value found so far. The algorithm then *backtracks* to any unexplored nodes at previous levels. If the J value of a node at level k is less than B (i.e., $J(Y\setminus\{y_1,y_2,\ldots,y_k\})<B$, where $k<D-d$ and each variable y_i takes on values in $\{1,2,\ldots,D\}$), then due to the monotonicity property of J in Eq. (1),

$$J(Y\setminus\{y_1,\ldots,y_k,y_{k+1},\ldots,y_{D-d}\})<B \quad \text{for all possible } \{y_{k+1},\ldots,y_{D-d}\} \tag{3}$$

This means that if $J<B$ for a node, its successor nodes (leaves) at the bottom of the tree can be eliminated because they cannot be the optimal subset of d features. The BB algorithm can thus efficiently cut off many sub-trees. If $J>B$ for a node, its successor nodes are explored further (as long as their J values remain larger than B). If a new different full path with a $J>B$ is found, the bound B is updated with the new larger J value. The search and backtracking continues until all leaves in the tree are either explored or cut off from the tree; thus, the BB algorithm gives the optimal solution.

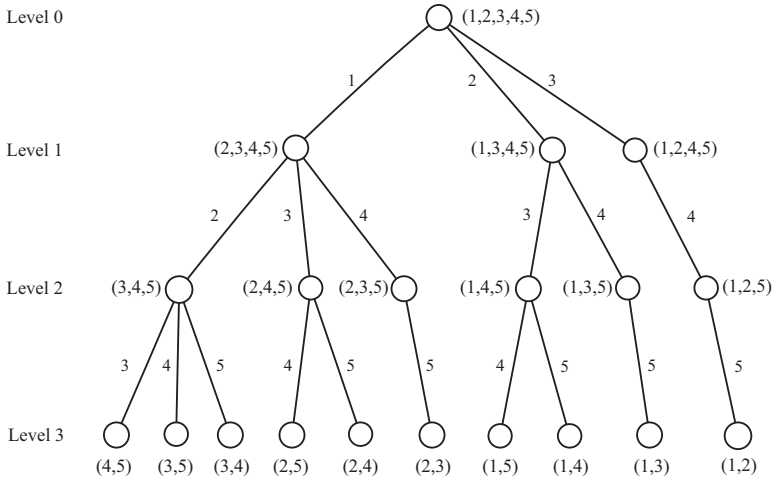

Fig. 1. The solution tree for the basic BB algorithm when $d=2$ and $D=5$.