A Scalable Feature Selection Algorithm for Large Datasets – Quick Branch & Bound Iterative (QBB-I)

Prema Nedungadi[1] and M.S. Remya[2]

[1] Amrita Create
[2] Department of Computer Science,
Amrita Vishwa Vidyapeetham
prema@amrita.edu

Abstract. Feature selection algorithms look to effectively and efficiently find an optimal subset of relevant features in the data. As the number of features and the data size increases, new methods of reducing the complexity while maintaining the goodness of the features selected are needed. We review popular feature selection algorithms such as the probabilistic search algorithm based Las Vegas Filter (LVF) and the complete search based Automatic Branch and Bound (ABB) that use the consistency measure. The hybrid Quick Branch and Bound (QBB) algorithm first runs LVF to find a smaller subset of valid features and then performs ABB with the reduced feature set. QBB is reasonably fast, robust and handles features which are interdependent, but does not work well with large data. In this paper, we propose an enhanced QBB algorithm called QBB Iterative (QBB-I).QBB-I partitions the dataset into two, and performs QBB on the first partition to find a possible feature subset. This feature subset is tested with the second partition using the consistency measure, and the inconsistent rows, if any, are added to the first partition and the process is repeated until we find the optimal feature set. Our tests with ASSISTments intelligent tutoring dataset using over 150,000 log data and other standard datasets show that QBB-I is significantly more efficient than QBB while selecting the same subset of features.

Keywords: Feature Selection, Search Organization, Evaluation Measure, Quick Branch & Bound, QBB-Iterative.

1 Introduction

In today's information age, it is easy to accumulate data and inexpensive to store it. The ability to understand and analyze large data requires us to more efficiently find the optimal subset of features. Feature selection is the task of searching for an optimal subset of features from all available features [15]. Feature selection aims to find the optimal subset of features by removing redundant and irrelevant features that contribute noise and to reduce the computational complexity.

Existing feature selection algorithms (FSA) for machine learning typically fall into two broad categories: wrappers and filters [13-15]. Filter methods choose the best

M.K. Kundu et al. (eds.), *Advanced Computing, Networking and Informatics - Volume 1*,
Smart Innovation, Systems and Technologies 27,
DOI: 10.1007/978-3-319-07353-8_15,

individual features, by first ranking the features by some informativeness criterion [21], such as their Pearson Correlation with the target and then selecting the top features. Finally, this subset of features is presented as input to the classification algorithm. Wrapper methods [21] use a search procedure in the space of possible feature subsets using some search strategy such as Sequential Forward Selection (SFS) or Sequential Backward Selection (SBS), and various subsets of features are generated and evaluated. There is also a third category named embedded methods [21], where the search for an optimal subset of features is built into the classifier construction. Features are selected as a part of the building the particular classifier, in contrast to the wrapper approach, where a classification model is used to evaluate a feature subset that is selected without using the classifier.

Within these categories, algorithms can be further differentiated by the precise nature of their evaluation function, and by how the space of feature subsets is explored. According to Dash *et al.*, evaluation measures can be divided into 5 categories, such as: distance (e.g. Euclidean distance), information (e.g. information gain), dependency (e.g. correlation coefficient), classifier error rate and consistency (e.g. inconsistency rate) [17]. By training and testing a definite classification model, the evaluation of a specific subset of features is performed. The search for the desired feature subset is wrapped around a specific classifier and training algorithm.

In this paper, we review a hybrid algorithm called Quick Branch and Bound (QBB) [16], which is a combination of two popular algorithms, the probabilistic search algorithm based Las Vegas Filter (LVF) and the complete search based Automatic Branch and Bound (ABB). QBB is reasonably fast, robust and handles features which are inter-dependent, but does not work well with large data [17]. Las Vegas Iterative (LVI) is a scalable version of LVF that first partitions the dataset into two partitions and uses the first partition to learn the features and verifies the goodness of the selected features with the second partition using the inconsistency measure [7]. We propose an enhanced version of QBB, called QBB Iterative (QBB-I) based on some of the concepts used in LVI and show that QBB-I is more efficient than QBB for larger datasets without sacrificing the quality of the features selected.

2 Feature Selection Algorithms

2.1 Feature Selection

A feature selection algorithm (FSA) is a computational solution that is motivated by a certain definition of relevance [19]. The FSAs can be classified according to the kind of output they yield; the algorithms that give a weighed linear order of features and the algorithms that find a subset of the original features [18]. Both types can be seen in a unified way by noting that in the latter case the weighting is binary. In this paper, we consider algorithms that find a subset of the original features. Fig. 1 describes the characteristics of FSA [19].

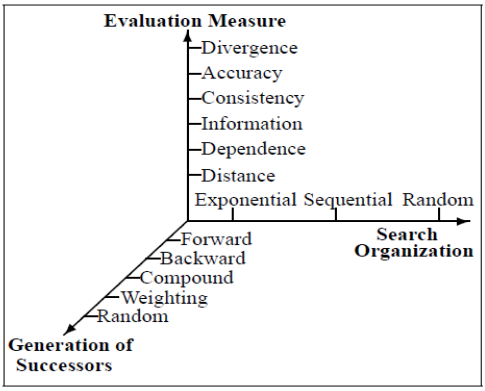

Fig. 1. Characterization of a FSA [19]

The two important aspects of a FSA are the search strategies to find the subset of features and the evaluation measure used to find the goodness of feature subsets. The search strategies employed by FSA include the exhaustive, complete, heuristic and random search. The search process may start with the empty set and increase by one feature at a time (forward search) or start with the full set and drop one feature at a time (backward search). The search may also start with a randomly generated feature set and change either probabilistically or deterministically. The evaluation measures include distance measures, information gain, correlation measures, consistency, and classifier error rate.

The algorithms used in this paper use the consistency measure. The inconsistency rate of a data is defined as follows [6]: (a) if the two patterns match all but their class labels, then they are considered inconsistent. (b) The inconsistency count is the total matching patterns minus largest number of patterns of different class label. For example, if there are n matching patterns, among them, c1 patterns belongs to label1, c2 to label2, c3 to label3 where $n = c1 + c2 + c3$. If c3 is largest among three, then inconsistency count = n - c3. (c) Inconsistency rate is equal to sum of all inconsistency counts divided by total number of patterns.

The QBB Algorithm [16] is a hybrid of two popular algorithms, the LVF [6] and the ABB [5] algorithms and exploits the advantage and disadvantage of both algorithms. LVF is probabilistic search algorithm which in the initial phase quickly reduces the number of features to a subset as initially many subsets can satisfy the consistency measure. However, after a certain point, fewer subsets can satisfy this and thus increase the computational complexity as it spends resource on randomly generating subsets that are obviously not good.

ABB starts with the full feature set and uses the inconsistency measure as the bound. It reduces the feature set by one feature at a time to generate subsets and is a complete search that guarantees optimal subset. However, its performance deteriorates when the difference between the total number of features and the optimal subset of features is large. QBB was proposed to exploit the advantages and overcome the disadvantages of both LVF and ABB. LVF quickly reduces the size of the initial

feature set to a consistent subset and then uses this smaller feature set as the input to ABB to find the optimal feature subset.

In a survey and experimental evaluation of feature selection algorithms [19], Molina *et al.* review different fundamental feature selection algorithms and assesses their performances. They evaluate and compare FSA such as LVF [6], ABB [5], QBB [16], FOCUS [1], RELIEF [11], and LVI [17] in order to understand the general behavior of FSAs on the particularities of relevance, irrelevance, redundancy and sample size of synthetic data sets and show that overall hybrid algorithms perform better than individual algorithms.

In [23], the authors present the activity recommendation and compare QBB with different algorithms and show that QBB has the best results. QBB is reasonably fast, robust and handle features which are inter-dependent but does not work well with large datasets [17].

2.2 Quick Branch and Bound Iterative (QBB-I) Algorithm

We propose the Quick Branch and Bound Iterative (QBB-I) algorithm so as to retain the advantages of QBB while reducing the time complexity. QBB-I divides the data to two parts, D0 (p %) and D1 (1-p %) and finds a subset of features for D1 using the consistency measure. It is designed such that the features selected from the reduced data will not generate more inconsistencies with the whole data.

Algorithm 1. Algorithm for QBB-iterative

```
1:  Procedure QBB-I (D, δ, S, p)
2:      δ = inConCal(S, D); T = S
3:      D0 = p% of Dataset; D1 = D - D0
4:      loop
5:          S′ = LV F(S, MaxTries, D0);
            /*All legitimate subsets from LVF are in S′*/
6:          MinSize=card(S);
7:          loop
8:              T′ = ABB (Sj, D0)
9:              if (MinSize > card (T′) then
10:                 MinSize = card (T′);
11:                 T = Sj;
12:             end if
13:         end loop
14:         if (checkIncon (subset, D1, inconData) ≤ δ) then
15:             Return T;
16:         else
17:             append (inconData, D0);
18:             remove (inconData, D1);
19:         end if
20:     end loop
21: end procedure
```

QBB-I finds a subset of features for D0 and checks this with D1 using the consistency measure. If the inconsistency rate exceeds the threshold, it appends the patterns from D1 which cause the inconsistency to D0 and deletes them from D1. This selection processes repeats until a solution is found. If no subset is found, the complete set of attributes is returned as a solution.

Table 1 describes the type of Search Organization, Generation of Successors and Evaluation Measure used by the four algorithms. A search algorithm takes responsibility for driving the feature selection process.

Table 1. Comparison of feature selection algorithms used

Characteristics	**LVF**	**LVI**	**QBB**	**QBB-I**
Search Organization	Random	Random	Random\ Exponential	Random\ Exponential
Generation of Successor	Random	Random	Random	Random
Evaluation Measures	Consistency	Consistency	Consistency	Consistency
Learning Speed	High	Medium	Medium	Medium
Data Size	Small	Large	Small	Large
Execution Time	High	Medium	High	Low

The time complexity of the feature selection algorithms is based on two parameters, the number of instances (N) and number of attributes (S). As N is much larger than S, we can significantly improve the performance with QBB-I over QBB by partitioning the data and starting with a smaller $N' << N$**.**

3 Experimental Results of QBB-I

We want to verify a) QBB-I run faster than QBB on large datasets, b) the feature set returned by QBB-I is identical to QBB, c) QBB-I is faster than LVF, ABB, and QBB in general and d) QBB-I is particularly suitable for huge datasets.

Our experiments compare QBB-I with other baseline existing algorithms such as LVF, LVI and QBB in terms of execution time and selected features. All algorithms use the consistency measure with the inconsistency rate set to 0 as there is no prior knowledge about the data. Using eight different datasets, we find that QBB-I takes less execution time compared to other algorithms. Our tests with ASSISTments intelligent tutoring dataset using 150,000 log data and other standard datasets show that QBB-I is significantly more efficient than QBB while selecting the same set of optimal features.

3.1 Dataset Used for Feature Selection

For testing the proposed approached, we use the Cognitive Tutor dataset, from the Knowledge Discovery and Data Mining Challenge 2010 KDD Cup 2010 [12] from two different tutoring systems from multiple schools over multiple school years. The dataset contains 19 attributes. We used the Challenge dataset for this work. These datasets are quite large which contains 3,310 students with 9,426,966 steps [12]. There are many technical challenges such as the data matrix is sparse, there is a strong temporal dimension to the data and the problem a given student sees is determined in part by student choices or past success history.

A total of seven datasets, both artificial and real, are chosen for the experiments from the UC Irvine data repository [4]. The Lymphography dataset [3] contains 148 instances and the test set is selected randomly with 50 instances. It contains 19 attributes. The lung-cancer dataset [8] describes two types of pathological lung cancers and contains 32 instances and 57 attributes. Mushroom dataset [22] contains 8124 instances and the dataset has 22 discrete attributes. Parity5+5 dataset [10], the concept is the parity of five bits. The dataset contains 10 features, of which five are uniformly random ones.

Splice junctions [20] are points on a DNA sequence at which superfluous DNA is removed during the process of protein creation in higher organisms. Led24 dataset [2] has a total of 3200 instances, of which 3000 are selected for testing. It contains 24 attributes. All attribute values are either 0 or 1, according to whether the corresponding light is on or not for the decimal digit.

3.2 Experimental Setup

QBB and QBB-I were implemented using the C language. In all algorithms, the inconsistency rate is taken as zero. And, the partition percentage used in QBB-I is 27% and that of LVI is 24%. The experiment is done with over 150,000 log data from the ASSISTments dataset.

3.3 Results

A total of seven datasets, both artificial and real, are chosen for the experiments from the UC Irvine data repository to check the effectiveness of QBB-I. These datasets are either commonly used for testing feature selection algorithms or artificially designed so that relevant attributes are known. Fig. 2 shows the graphical representation for the comparison of LVF, QBB and QBB-I for different datasets. For the first three dataset, there is no much reduction in the execution time due to small set of data. But for parity and KDD datasets, it is much clearer. So from this analysis, we can conclude that QBB-I algorithm is faster than the other algorithms for larger datasets and less beneficial for smaller datasets.

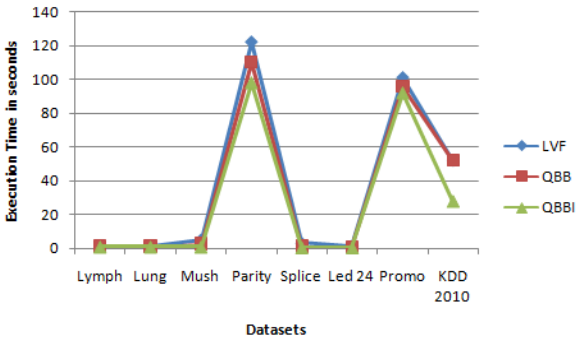

Fig. 2. Comparison of LVF, QBB and QBB-I in different datasets

We then compare the features selected by LVF, QBB and QBB-I. This experiment is mainly focused on checking the accuracy of our Enhanced algorithm. Table 2 shows the features selected by three algorithms. Table 2 shows that the features selected by QBB-I are identical to that of QBB while the features selected by LVI is identical to that of LVF.

Table 2. Features Selected By Three Algorithms

LVF	LVI	QBB	QBB-I
Row	Row		
StudentId	StudentId	StudentId	StudentId
Problem Hierarchy	Problem Hierarchy	Problem Hierarchy	Problem Hierarchy
Problem Name	Problem Name	Problem Name	Problem Name
Problem View	Problem View	Problem View	Problem View
Step Name	Step Name	Step Name	Step Name
CFA	CFA	CFA	CFA
InCorrects	InCorrects		
Hints	Hints		
Corrects	Corrects		
Opportunity	Opportunity	Opportunity	Opportunity
KC	KC	KC	KC

We also compare the execution time of the four algorithms (Fig. 3).The result shows that the QBB-I takes less execution time compared to all other algorithms.

Among the three existing algorithms, LVI was the fastest while QBB gave the better feature set. Our QBB-I algorithm, which uses the scalability concepts from LVI to scale the QBB algorithm, returns the same feature set as QBB and is faster than LVI.

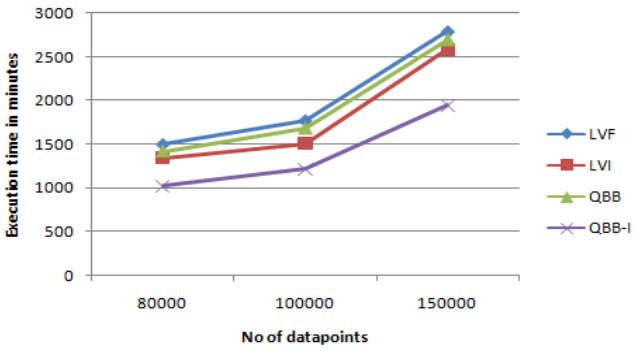

Fig. 3. Comparison of Execution time of different algorithms

For the ASSISTments dataset, Yu. *Et al*[9] manually identify useful combinations of features and then experimentally shows that some of the feature combinations effectively improve the Root Mean Squared Error (RMSE). While the previous experiments were with individual features, we also compare the features selected by QBB and QBB-I with the following combinations of features suggested by this paper. (Student Name, Unit Name), (Unit Name, Section Name), (Section Name, Problem Name), (problem Name, step Name), (Student Name, Unit Name, Section Name), (Section Name, Problem Name, Step Name), (Student Name, Unit Name, Section Name, Problem Name) and (Unit Name, Section Name, Problem Name, Step Name).

Both QBB and QBB-I selected the last four combinations with a subset of the ASSISTments dataset consisting of 150,000 log records, which have previously been shown to be the best combinations [9]. Fig. 4 shows the comparison of the time taken by QBB and QBB-I for selecting the combinations for 80,000, 100,000 and 150,000 log records. The result shows that QBB-I takes less time for execution compared to QBB.

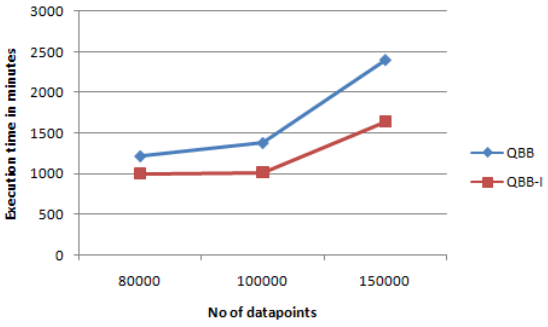

Fig. 4. Comparison of time for selecting combinations

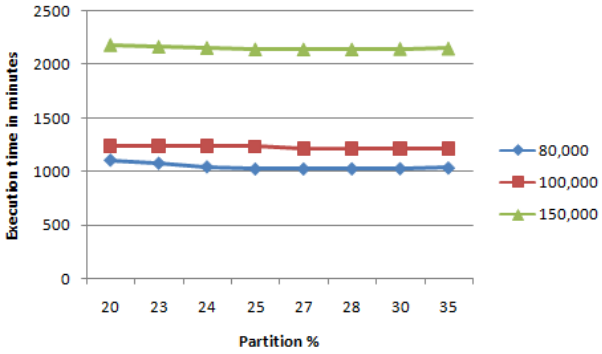

Fig. 5. Experiment to find out the optimum partition size (%)

Finally we vary the partition size (Refer Fig. 5) as we expect that as the partition size increases; the execution time reduces and then becomes a constant. We find that for 80000 log records of the ASSISTments dataset, this is reached with a partition size of around 25% while with 100,000 and 150,000 log records of the same dataset, this is reached with a partition size of 27%.

4 Conclusion

We proposed an algorithm (QBB-I) to feature selection which is a scaled version of QBB and show that it is more efficient than QBB while selecting the same feature set. We increase the efficiency of QBB-I by partitioning the dataset such that the proportion of the first partition is small with respect to the entire dataset. As inconsistent entries are moved to the first partition, QBB-I learns and avoids checking inconsistent entries in future iterations.

QBB-I improves the execution time of QBB algorithm when the dataset is large. When the dataset is small the performance improvement is negligible as the inconsistency checking is of order (N) and the time saving is not much if N is not large. Similar to LVI, the second issue is the size of the partition to start QBB-I. If D0 is too small, QBB-I may select features where majority of the data in D1 are inconsistent. In the worst case, D0 may become the entire dataset in the next iteration. If D0 is too large, the overheads of the additional loops may be larger and it may be better to just use QBB. The optimum partition size needs to be determined by testing with the data.

Using eight different datasets, we find that QBB-I takes less execution time compared to other algorithms. Our tests with ASSISTments intelligent tutoring dataset using over 150,000 log data and other standard datasets shows that QBB-I is significantly more efficient than QBB while selecting the same subset of optimal features. Future work involves testing this with larger datasets and the use of parallel feature selection algorithms.

Acknowledgments. This work derives direction and inspiration from the Chancellor of Amrita University, Sri Mata Amritanandamayi Devi. We thank Dr. M Ramachandra Kaimal, head of Computer Science Department, Amrita University for his valuable feedback.

References

1. Almuallim, H., Dietterich, T.G.: Learning with many irrelevant features. In: Proceedings of the 9th National Conference on Artificial Intelligence (1991)
2. Breiman, L., Friedman, J.H., Olshen, R.A., Stone, C.J.: Classification and Regression Trees. Wadsworth International Group, Belmont (1984)
3. Cestnik, G., Konenenko, I., Bratko, I.: Assistant-86: A Knowledge- Elicitation Tool for Sophisticated Users. In: Progress in Machine Learning, pp. 31–45. Sigma Press (1987)
4. Merz, C.J., Murphy, P.M.: UCI Repository of Machine Learning Batabases. University of California, Department of Information and Computer Science, Irvine (1996), http://www.ics.uci.edu/mlearn/MLRepository.html
5. Liu, H., Motoda, H., Dash, M.: A monotonic measure for Optimal Feature Selection. In: Proceedings of the European Conference on Machine Learning, pp. 101–106 (1998)
6. Liu, H., Setiono, R.: A probabilistic approach to feature selection: a Filter solution. In: Proceedings of the 13th International Conference on Machine Learning, pp. 319–327 (1996)
7. Liu, H., Setiono, R.: Scalable feature selection for large sized databases. In: Proceedings of the 4th World Congress on Expert System, p. 6875 (1998)
8. Hong, Z.Q., Yang, J.Y.: Optimal Discriminant Plane for a Small Number of Samples and Design Method of Classifier on the Plane. Pattern Recognition 24, 317–324 (1991)
9. Yu, H.-F., Lo, H.-Y., Hsieh, H.-P.: Feature Engineering and Classifier Ensemble for KDD Cup 2010. In: JMLR: Workshop and Conference Proceedings, vol. 1, pp. 1–16 (2010)
10. John, G.H., Kohavi, R., Pfleger, K.: Irrelevant features and the subset selection problem. In: Proceedings of the Eleventh International Conference in Machine Learning (1994)
11. Kira, K., Rendell, L.: A practical approach to feature selection. In: Proceedings of the 9th International Conference on Machine Learning, pp. 249–256 (1992)
12. Koedinger, K., Baker, R., Cunningham, K., Skogsholm, A., Leber, B., Stamper, J.: A data repository for the EDM community: the pslc datashop (2010)
13. Kohavi, K.: Wrappers for performance enhancement and oblivious decision graphs. Phd thesis, Stanford university (1995)
14. Kohavi, R., John, G.H.: Wrappers for feature subset selection. Artificial Intelligence 97(12), 273–324 (1996)
15. Dash, M., Liu, H.: Feature selection for classification. Intelligent Data Analysis 1(1-4), 131–156 (1997)
16. Dash, M., Liu, H.: Hybrid search of feature subsets. In: Lee, H.-Y., Motoda, H. (eds.) PRICAI 1998. LNCS, vol. 1531, pp. 238–249. Springer, Heidelberg (1998)
17. Dash, M., Liu, H., Motoda, H.: Feature Selection Using Consistency Measure. In: Arikawa, S., Nakata, I. (eds.) DS 1999. LNCS (LNAI), vol. 1721, pp. 319–320. Springer, Heidelberg (1999)
18. Kudo, M., Sklansky, J.: A Comparative Evaluation of medium and largescale Feature Selectors for Pattern Classifiers. In: Proceedings of the 1st International Workshop on Statistical Techniques in Pattern Recognition, pp. 91–96 (1997)

19. Molina, L.P., Belanche, L., Nebot, A.: Feature selection algorithms: a survey and experimental evaluation. Universitat Politcnica de catalunya. departament de llenguatges i sistemes informtics (2002)
20. Noordewier, M.O., Towell, G.G., Shavlik, J.W.: Training Knowledge-Based Neural Networks to Recognize Genes in DNA Sequences. In: Advances in Neural Information Processing Systems, vol. 3 (1991)
21. Saeys, Y., Inza, I., Larranaga, P.: A review of feature selection techniques in bioinformatics. Bioinformatics 23(19), 2507–2517 (2007)
22. Schlimmer, J.S.: Concept Acquisition Through Representational Adjustment (Technical Report 87-19). Doctoral disseration, Department of Information and Computer Science, University of California, Irvine (1987)
23. Nguyen, T.: A Group Activity Recommendation Scheme Based on Situation Distance and User Similarity Clustering. M. Thesis, Department of Computer Science KAIST (2012)