High-Dimensional Data Feature Subset Selection Using a Clustering-Based Algorithm: An Effective Implementation

K.Divya Kalyani, S.Venkata Lakshmi, P.Mounika
Assistant Professor, Dadi Institute of Engineering and Technology
divyakalyanik@diet.edu.in, venkatalakshmisalapu@gmail.com
reddymounika010593@gmail.com

Abstract:
Feature selection involves identifying a subset of the most useful features that produces compatible results as the original entire set of features. A feature selection algorithm may be evaluated from both the efficiency and effectiveness points of view. While the efficiency concerns the time required to find a subset of features, the effectiveness is related to the quality of the subset of features. Based on these criteria, a fast clustering-based feature selection algorithm (FAST) is proposed and experimentally evaluated in this paper. The FAST algorithm works in two steps. In the first step, features are divided into clusters by using graph-theoretic clustering methods. In the second step, the most representative feature that is strongly related to target classes is selected from each cluster to form a subset of features. Features in different clusters are relatively independent, the clustering-based strategy of FAST has a high probability of producing a subset of useful and independent features. To ensure the efficiency of FAST, we adopt the efficient minimum-spanning tree (MST) clustering method. The efficiency and effectiveness of the FAST algorithm are evaluated through an empirical study. Extensive experiments are carried out to compare FAST and several representative feature selection algorithms, namely, FCBF, ReliefF, CFS, Consist, and FOCUS-SF, with respect to four types of well-known classifiers, namely, the probability-based Naive Bayes, the tree-based C4.5, the instance-based IB1, and the rule-based RIPPER before and after feature selection. The results, on 35 publicly available real-world high-dimensional image, microarray, and text data, demonstrate that the FAST not only produces smaller subsets of features but also improves the performances of the four types of classifiers.

Introduction

With the aim of choosing a subset of good features with respect to the target concepts, feature subset selection is an effective way for reducing dimensionality, removing irrelevant data, increasing learning accuracy, and improving result comprehensibility. Many feature subset selection methods have been proposed and can be

divided into four broad categories: the Embedded, Wrapper, Filter, and Hybrid approaches.

The wrapper method are computationally expensive and tend to over fit on small training sets. The filter methods,in addition to their generality, are usually a good choice when the number of features is very large. Thus, we will focus on the filter method in this paper. With respect to the filter feature selection methods, the application of cluster analysis has been demonstrated to be more effective than traditional feature selection algorithms.

In cluster analysis, graph-theoretic methods have been well studied and used in many applications. Their results have, sometimes, the best agreement with human performance. The general graph-theoretic clustering is simple: compute a neighborhood graph of instances, then delete any edge in the graph that is much longer / shorter (according to some criterion) than its neighbors. The result is a forest and each tree in the forest represents a cluster. We apply graph-theoretic clustering methods to features. In particular, we adopt the minimum spanning tree (MST) based clustering algorithms, because they do not assume that data points are grouped around centers or separated by a regular geometric curve and have been widely used in practice. Based on the MST method, we propose a fast clustering based feature subset Selection algorithm (FAST). The FAST algorithm works in two steps. In the first step, features are divided into clusters by using graph-theoretic clustering methods. In the second step, the most representative feature that is strongly related to target classes is selected from each cluster to form the final subset of features. Features in different clusters are relatively independent; the clustering based strategy of FAST has a high probability of producing a subset of useful and independent features.

Proposed System

Feature subset selection can be viewed as the process of identifying and removing as many irrelevant and redundant features as possible. This is because irrelevant features do not contribute to the predictive accuracy and redundant features do not redound to getting a better predictor for that they provide mostly information which is already present in other features. Of the many feature subset selection algorithms, some can effectively eliminate irrelevant features but fail to handle redundant features yet some of others can eliminate the irrelevant while taking care of the redundant features. Our proposed FAST algorithm falls into the second group.

Traditionally, feature subset selection research has focused on searching for relevant features. A well-known example is Relief which weighs each feature according to its ability to discriminate instances under different targets based on distance-based criteria function. However, Relief is ineffective at removing redundant features as two predictive but highly correlated features are likely both to be highly weighted. Relief-F extends Relief, enabling this method to work with noisy and incomplete data sets and to deal with multiclass problems, but still cannot identify redundant features.

A. Advantages of Proposed System:

1. Good feature subsets contain features highly correlated with the class, yet uncorrelated with each other.
2. The efficiently and effectively deal with both irrelevant and redundant features, and obtain a good feature subset.
3. Generally all the six algorithms achieve significant reduction of dimensionality by selecting only a small portion of the original features.
4. The null hypothesis of the Friedman test is that all the feature selection algorithms are equivalent in terms of runtime.

B. Methodologies:

Various generic software development life cycle methodologies are available for executing software development projects. Although each methodology is designed for a specific purpose and has its own advantages and disadvantages, most methodologies divide the life cycle into phases and share tasks across these phases. This section briefly summarizes common methodologies used for software development and describes their relationship to testing.

C. Waterfall Model:

The waterfall model is one of the earliest structured models for software development. It consists of the following sequential phases through which the development life cycle progresses:

System Feasibility: In this phase, you consider the various aspects of the targeted business process, find out which aspects are worth incorporating into a system, and evaluate various approaches to building the required software.

Requirement Analysis: In this phase, you capture software requirements in such a way that they can be translated into actual use cases for the system. The requirements can derive from use cases, performance goals, target deployment, and so on.

System Design: In this phase, you identify the interacting components that make up the system. You define the exposed interfaces, the communication between the interfaces, key algorithms used, and the sequence of interaction. An architecture and design review is conducted at the end of this phase to ensure that the design conforms to the previously defined requirements.

Coding and Unit Testing: In this phase, you write code for the modules that make up the system. You also review the code and individually test the functionality of each module.

Integration and System Testing: In this phase, you integrate all of the modules in the system and test them as a single system for all of the use cases, making sure that the modules meet the requirements.

Deployment and Maintenance: In this phase, you deploy the software system in the production environment. You then correct any errors that are identified in this phase, and add or modify functionality based on the updated requirements.

D. Incremental or Iterative Development:

The incremental, or iterative, development model breaks the project into small parts. Each part is subjected to multiple iterations of the waterfall model. At the end of each iteration, a new module is completed or an existing one is improved on, the module is integrated into the structure, and the structure is then tested as a whole. The main advantage of the iterative development model is that corrective actions can be taken at the end of each iteration. The corrective actions can be changes to the specification because of incorrect interpretation of the requirements, changes to the requirements themselves, and other design or code-related changes based on the system testing conducted at the end of each cycle.

E. Prototyping Model:

The prototyping model assumes that you do not have clear requirements at the beginning of the project. Often, customers have a vague idea of the requirements in the form of objectives that they want the system to address. With the prototyping model, you build a simplified version of the system and seek feedback from the parties who have a stake in the project. The next iteration incorporates the feedback and improves on the requirements specification.

The prototyping model consists of the following steps:

Capture Requirements: This step involves collecting the requirements over a period of time as they become available.

Design the System: After capturing the requirements, a new design is made or an existing one is modified to address the new requirements.

Create or Modify The Prototype: A prototype is created or an existing prototype is modified based on the design from the previous step.

Assess Based on Feedback: The prototype is sent to the stakeholders for review. Based on their feedback, an impact analysis is conducted for the requirements, the design, and the prototype. The role of testing at this step is to ensure that customer feedback is incorporated in the next version of the prototype.

Refine The Prototype: The prototype is refined based on the impact analysis conducted in the previous step.

Implement The System: After the requirements are understood, the system is rewritten either from scratch or by reusing the prototypes. The testing effort consists of the following:

- Ensuring that the system meets the refined requirements
- Code review
- Unit testing
- System testing

Conclusion

We have presented a novel clustering- based feature subset selection algorithm for high dimensional data. The algorithm involves removing irrelevant features, constructing a minimum spanning tree from relative ones, and partitioning the MST and selecting representative features. In this algorithm, a cluster consists of features. Each cluster is treated as a single feature and thus dimensionality is drastically reduced. The result is a forest and each tree in the forest represents a cluster.

We apply graph-theoretic clustering methods to features. The most representative feature that is strongly related to target classes is selected from each cluster to form the final subset of features. Features in different clusters are relatively independent; the clustering based strategy of FAST has a high probability of producing a subset of useful and independent features.

References

[1]. H. Almuallim and T.G. Dietterich, "Algorithms for Identifying Relevant Features," Proc. Ninth Canadian Conf. Artificial Intelligence, pp. 38-45, 1992.
[2]. J. Biesiada and W. Duch, "Features Election for High-Dimensional data a Pearson Redundancy Based Filter," Advances in Soft Computing, vol. 45, pp. 242-249, 2008.
[3]. L. Yu and H. Liu, "Feature Selection for High-Dimensional Data: A Fast Correlation- Based Filter Solution," Proc. 20th Int'l Conf. Machine Leaning, vol. 20, no. 2, pp. 856-863, 2003.
[4]. L. Yu and H. Liu, "Efficiently Handling Feature Redundancy in High-Dimensional Data," Proc. Ninth ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining (KDD '03), pp. 685-690, 2003.
[5]. J. Demsar, "Statistical Comparison of Classifiers over Multiple Data Sets," J. Machine Learning Res., vol. 7, pp. 1-30, 2006.

