Markov Blanket Feature Selection for Support Vector Machines

Jianqiang Shen and Lida Li and Weng-Keen Wong
1148 Kelley Engineering Center, School of EECS
Oregon State University, Corvallis, OR 97331, U.S.A.
{shenj,lili,wong}@eecs.oregonstate.edu

Abstract
Based on Information Theory, optimal feature selection should be carried out by searching Markov blankets. In this paper, we formally analyze the current Markov blanket discovery approach for support vector machines and propose to discover Markov blankets by performing a fast heuristic Bayesian network structure learning. We give a sufficient condition that our approach will improve the performance. Two major factors that make it prohibitive for learning Bayesian networks from high-dimensional data sets are the large search space and the expensive cycle detection operations. We propose to restrict the search space by only considering the promising candidates and detect cycles using an online topological sorting method. Experimental results show that we can efficiently reduce the feature dimensionality while preserving a high degree of classification accuracy.

Introduction
Many machine learning problems require learning from high-dimensional data sets in which the number of instances is less than the number of features. A common strategy is to first run a feature selection step before applying a traditional learning algorithm. The main reasons for applying feature selection is to avoid overfitting by removing irrelevant features and to improve computational efficiency by decreasing the number of features the learning algorithm needs to consider. Many common-used feature selection methods such as mutual information and information gain (Yang & Pedersen 1997) are based on pairwise statistics. Although pairwise statistics can be computed quickly, they are unable to capture higher order interactions and can be quite unreliable when the training data is sparse. Also, the user usually has to tune a threshold to determine the optimal feature set. This paper presents an efficient and reliable feature selection approach without the need for such threshold.

Using Markov blankets for feature selection has been shown to be effective (Koller & Sahami 1996). The Markov blanket is defined as the set of variables \( M \) that renders a variable \( X \) conditionally independent of other variables given the variables in \( M \). Feature selection algorithms based on Markov blankets have attracted much attention recently (Aliferis, Tsamardinos, & Statnikov 2003; Bai et al. 2004). They all depend on some independence tests. In this paper, we formally analyze the performance of these test-based algorithms.

We propose to discover Markov blankets by first performing a fast heuristic Bayesian network structure learning step. Bayesian networks exploit conditional independence to produce a succinct representation of the joint probability distribution between a set of variables. Heuristic structure learning first defines a score that describes the fitness of each possible structure to the observed data, and then searches for a structure that maximizes the score. We analyze the sufficient condition that our approach improves the performance. Rather than learning the Bayesian network, we could try to learn only the Markov blanket for the class variable. However, as will be shown in our results, we observe much better accuracy when we first learn a Bayesian network because the Bayesian network captures a more accurate global view of how all the random variables interact with each other. Secondly, learning a network gives us more flexibility, and we can consider any node as the “class variable” once we have the Bayesian network. Finally, learning a network can still be quite efficient compared with only learning Markov blankets. Our method can learn a reasonably accurate Bayesian network from large text datasets within minutes.

Throughout this paper, we use linear support vector machines (SVMs) to perform classification after feature selection, since SVM is one of the state-of-the-art learning methods and has met success in numerous domains. This can be considered as a hybrid generative-discriminative learning approach (Raina et al. 2004). Although discriminative classifiers usually achieve higher prediction accuracy than generative classifiers, it has been shown that a generative classifier outperforms its discriminative counterpart when the amount of training data is limited (Ng & Jordan 2002).

In this paper we essentially apply the generative learner to identify which information is necessary for training the discriminative learner. Our hybrid approach significantly outperforms the pure SVM when the training size is small.

Searching Markov Blanket Features
Most common-used feature selection methods (such as mutual information, information gain, Chi square (Yang & Pedersen 1997)) are based on pairwise statistics which is unreliable when the training size is small. It is also difficult to de-
termine the optimal feature set size and the user has to tune some threshold. Furthermore, pairwise statistics have difficulty in removing redundant features and capturing higher-order interactions. Consider variables $x$ and $y$. It is possible that they will be both retained if they are highly correlated (and thus have close statistical measurements). It is also possible that they will be both removed if the class variable’s interaction with $x$ or $y$ alone is not significant, although the co-occurrence of $x$ and $y$ is very predictive of the class.

The Markov blanket criterion can address the above problems. The Markov blanket subsumes the information that a variable has about all of the other features. Koller and Sahami (1996) suggested that we should remove features for which we find a Markov blanket within the set of remaining features. Since finding Markov blankets is hard, they proposed an approximate algorithm, which is suboptimal and very expensive due to its simple approximation (Baker & McCallum 1998). There are several other algorithms based on Markov blankets (Aliferis, Tsamardinos, & Statnikov 2003; Bai et al. 2004). They all depend on independence tests and are sensitive to test failures. Furthermore, the user still needs to tune the threshold to determine the optimal feature set size. We analyze their error bounds and propose to select features by heuristically searching Bayesian networks.

Analysis of Test-based Algorithms

For simplicity, let’s concentrate on binary, complete data. We assume that the chi-square test is used for independence test and a linear SVM is used for classification. The results can be easily generalized to other situations.

Regarding variables $A$ and $B$, independence test $T$ is carried out by testing null hypothesis $H_0 : P_{AB} = P_A \cdot P_B$. Given the nominal level $\alpha$, it rejects $H_0$ if the chi-square statistic $\chi^2 > \chi^2_{1, \alpha}$, where $\chi^2_{1, \alpha}$ is the upper $\alpha$ percentage point of the cumulative chi-square distribution with 1 degree of freedom.

Proof (Sketch) We want to ensure type II error $P(\chi^2 \leq \chi^2_{1, \alpha}) \leq \epsilon$. Under the alternative hypothesis, $\chi^2$ follows the non-central chi-square distribution with 1 degree of freedom and non-centrality parameter $\lambda = n \cdot \Delta$ (Meng & Chapman 1966). Applying Pearson transformation and Fisher approximation (Johnson & Kotz 1970), we get $n > \frac{1}{\epsilon} (\sqrt{\chi^2_{1, \alpha} + \frac{1}{2} U_c^2} + U_c)^2$ after some operations.

Note $\chi^2_{1, \alpha}$ and $U_c$ grow extremely fast as $\alpha \to 0$ and $\epsilon \to 0$. A test-based feature selection algorithm usually involves $\Omega(m)$ independence tests given $m$ variables. Because $\Delta$ is usually very small, the probability that $T$ treats dependent variables as independent is high with limited samples. Thus, test-based feature selection algorithms are too aggressive in removing features, as also shown in the experiments.

Method Based on Bayesian Network Learning

We showed that test-based algorithms are very sensitive to $\alpha$ and even with infinite samples their error rates are still proportional to $\alpha$. We propose to search Markov blankets through learning a Bayesian network with some good heuristic strategy. Bayesian networks provide a descriptive summary of relationships between variables and an efficient representation of the joint probability distribution. In this paper, we use Bayesian Information Criterion (BIC) as the search score. We choose BIC because it can penalize the model complexity and lead to smaller feature sets. If the goal is to preserve the prediction accuracy as much as we can, then a Bayesian score such as BDeu(Heckerman, Geiger, & Chickering 1995) might be preferred.

Given the class variable $C$, let $\hat{M}$ be its minimal Markov blanket. $\hat{M}$ is the smallest variable set rendering $C$ conditionally independent of all other variables not in $\hat{M}$. We propose to search $\hat{M}$ and adopt $\hat{M}$ as the goal feature set. We approximate $\hat{M}$ by heuristically learning a Bayesian network structure. If the learned Bayesian network is reasonably accurate, $\hat{M}$ should be close to the set consisting of $C$’s parents, children, and children’s parents. The user does not need to specify any thresholds. The optimization approach is also much more robust compared with the independence-test approach. Finally, Bayesian network structure learning methods using the BIC or Bayesian score is asymptotically successful: with probability 1, it will converge to the right structure given sufficient samples (Geiger et al. 2001).

Definition 2 The graph dimension $|G| = \sum_{i=1}^{m} 2(\text{parent}(i))$ is the number of parameters needed to specify graph $G$, where $m$ is the number of variables.
Existing approaches to restrict the search space usually involve heuristics that are not always effective. We propose a new method that combines the strengths of both heuristic and exact search approaches. Our method, called the sparse candidate algorithm, is based on the observation that many real-world datasets can be modeled effectively using a sparse network structure. This property allows us to significantly reduce the search space by focusing on promising candidate edges.

The Sparse Candidate Algorithm

The sparse candidate algorithm is designed to address the challenges of searching for the optimal network structure in large datasets. It iteratively identifies promising candidate edges based on their impact on the network score. The algorithm is guided by a heuristic that estimates the potential improvement of adding an edge to the current network structure. This heuristic is used to prioritize edges for inclusion in the candidate set, ensuring that promising edges are explored first.

Efficient Bayesian Network Searching

Learning Bayesian network structure is a computationally intensive task, especially for large datasets. We propose a novel heuristic that efficiently searches for the optimal network structure. Our method leverages the fact that many real-world datasets can be modeled with a sparse network structure. By focusing on promising candidate edges, we can significantly reduce the search space and improve the efficiency of the learning process.

The Screen-based Algorithm

The screen-based algorithm is another effective method for restricting the search space. It iteratively screens out non-promising edges based on their contribution to the network score. This screening process is guided by a heuristic that estimates the potential improvement of adding an edge to the current network structure. The algorithm is designed to efficiently search for the optimal network structure by focusing on promising candidate edges.

Conclusion

In this paper, we have presented a new method for restricting the search space in Bayesian network learning. Our method, called the sparse candidate algorithm, is based on the observation that many real-world datasets can be modeled effectively using a sparse network structure. By combining the strengths of heuristic and exact search approaches, our method is able to efficiently search for the optimal network structure in large datasets. We believe that our method will have significant impact on the field of Bayesian network learning.
Cycle Detection Using Online Topological Sorting

During the heuristic search, we should only consider legal moves: Bayesian networks are directed acyclic graphs (DAGs), and we need to ensure that acyclicity is preserved for addition and reversal operations. Given a DAG \( G = (V, E) \) with \(|V|\) variables and \(|E|\) edges, Depth First Search (DFS) algorithm takes \( O(|V| + |E|) \) to detect cycles (Cormen et al. 2001; Witten & Frank 2005). Giudici and Castelo (2003) proposed a more efficient cycle detection algorithm which reduces the cost to \( O(1) \) for addition and \( O(|V|) \) for reversal. However, it needs to maintain an ancestor matrix and the update takes \( O(|V|^2) \). When \(|V|\) is large, the cost of cycle detection becomes a speed bottleneck.

In this paper, we detect cycles by maintaining an online topological ordering (Alpern et al. 1990). Let \( x \rightarrow y \) denote that there is an edge from \( x \) to \( y \), and \( x \rightarrow y \) denote that there is a path from \( x \) to \( y \). A topological ordering, \( ord \), of a DAG \( G = (V, E) \) maps each vertex variable to an integer between 1 and \(|V|\) such that, for any \( x \rightarrow y \in E \), \( ord(x) < ord(y) \). We can take extra work to maintain \( ord \) of the graph and detect cycles efficiently based on \( ord \). Edge removal is always legal and we need not to update \( ord \). Edge reversal equals to first removing the arc, and then adding it in the opposite direction. Thus, edge addition is our major concern. When we add an edge \( x \rightarrow y \), we have two cases. First, \( ord(x) < ord(y) \). We have the following lemma.

**Lemma 3** If we insert an edge \( x \rightarrow y \) to DAG \( G(V, E) \) in which \( ord(x) < ord(y) \), then the new graph is still a DAG.

**Proof** This can be proved by contradiction. We assume that the new inserted edge \( x \rightarrow y \) introduces a cycle. Then there must exist a path from \( y \) to \( x \) in \( G \). Assuming the path is \( y \rightarrow v_1 \rightarrow \ldots \rightarrow v_t \rightarrow x \), by the definition of topological ordering, we have \( ord(y) < ord(v_1) < \ldots < ord(v_t) < ord(x) \). This is contradicted with the fact \( ord(x) < ord(y) \).

Based on Lemma 3, it is always legal to add an edge \( x \rightarrow y \) if \( ord(x) < ord(y) \). We do not update the topological ordering in this case. Second, if \( ord(x) > ord(y) \) (including the case to reverse edge \( y \rightarrow x \)), we only need to check a small subgraph of \( G \) and update its topological ordering. We define \( \delta_{xy}^+ \) and \( \delta_{xy}^- \). (Pearce & Kelly 2006) and can use Depth First Search to find them:

**Definition 4** The Affect Region \( AR_{xy} \) of a DAG \( G \) is \( \{ k \in V | ord(y) \leq ord(k) \leq ord(x) \} \). We define a set \( \delta_{xy} \) as \( \delta_{xy}^+ \cup \delta_{xy}^- \), where \( \delta_{xy}^+ = \{ k \in AR_{xy} | y \rightarrow k \} \) and \( \delta_{xy}^- = \{ k \in AR_{xy} | k \rightarrow x \} \).

**Lemma 4** Adding edge \( x \rightarrow y \) where \( ord(x) > ord(y) \) will introduce a cycle if and only if \( x \in \delta_{xy}^- \).

The proof is similar to Lemma 3. Thus, to check the legality of adding an edge \( x \rightarrow y \) where \( ord(x) > ord(y) \), we only need to apply a Depth First Search algorithm to see if \( x \in \delta_{xy}^- \). The complexity is only \( O(|\delta_{xy}^+| + |E(\delta_{xy}^-)|) \) where \( E(\delta_{xy}^-) = \{ a \rightarrow b \in E | a \in \delta_{xy}^- \land b \in \delta_{xy}^- \} \).

We should notice three important things when updating \( ord \): 1) adding edge \( x \rightarrow y \) need not influence the order of any node outside \( \delta_{xy} \), 2) in the new graph, the order of any node in \( \delta_{xy}^+ \) should have a smaller order than the order of any node in \( \delta_{xy}^- \) for addition and reversal operations. Given a DAG \( G = (V, E) \) with \(|V|\) variables and \(|E|\) edges, Depth First Search (DFS) algorithm takes \( O(|V| + |E|) \) to detect cycles (Cormen et al. 2001; Witten & Frank 2005). Giudici and Castelo (2003) proposed a more efficient cycle detection algorithm which reduces the cost to \( O(1) \) for addition and \( O(|V|) \) for reversal. However, it needs to maintain an ancestor matrix and the update takes \( O(|V|^2) \). When \(|V|\) is large, the cost of cycle detection becomes a speed bottleneck.

| Dataset       | \(|V|\) | \(|S|\) | \(p\) | \(k\) | DFS time | PK time |
|---------------|--------|--------|------|------|----------|---------|
| Sonar         | 61     | 208    | 5    | 60   | 2.24     | 0.75    |
| Lung-cancer   | 57     | 32     | 5    | 56   | 8.63     | 2.89    |
| MFeat-pixel   | 241    | 2000   | 5    | 40   | 486.36   | 23.21   |
| Corn vs Wheat | 2590   | 393    | 5    | 10   | \(> 259200\) | 399     |
| Corn vs Crude | 4324   | 570    | 5    | 10   | \(> 259200\) | 1627    |

**Table 1:** Time (s) to learn a Bayesian network with different cycle detection methods (no post-processing)

**Experimental Results**

We apply our feature selection methods to complete, discrete data. We begin with an empty network. We adopt the greedy hill-climbing for the sparse candidate algorithm, and set \( \theta = 4 \) for the screen-based algorithm. The learned network structure is suboptimal due to the search space restriction. To partially overcome the problem, we adopt the random hill-climbing to post-process the learned structure (Goldenberg & Moore 2004). Specifically, we randomly perform an edge operation (addition with probability 0.8, removal with 0.1, and reversal with 0.1) and keep the operation if the score is improved. We try 5 million random operations by default.

We use the F1 value (Joachims 2001) which is \( 2 \times \text{precision} \times \text{recall}/(\text{precision} + \text{recall}) \) to measure performance. The SVM implementation we use is LibSVM (Chang & Lin 2001) with the linear kernel and \( C = 1 \) as suggested in (Joachims 2001). We expect that a good heuristic learning method is usually close to \( \pi \)-promising and can find good feature sets with large chance. It is also interesting to investigate how efficient our method is.

**The Time Performance of Cycle Detections**

To ensure candidate network structures are legal, cycle detection has to be run frequently and can become a computation bottleneck if it is not efficient. We apply the sparse candidate algorithm to three datasets (Sonar, Lung-Cancer and MFeat-pixel) from the UCI ML Data Repository to test our cycle detection algorithm. We also test on two subsets of the Reuters-21578 dataset (we will discuss how to get them later). We compare the computing time of learning with different cycle detection methods in Table 1, where \(|V|\) is the number of variables, \(|S|\) is the number of training instances, \(p\) is the maximal parent number, and \(k\) is the count of the candidate parents. The time shown as \(> 259200\) means that the program does not finish after running 72 hours. As we can see, using the online topological sorting to detect cycles is much more efficient than the Depth First Search, especially when there are many variables.
Experiments on Synthetic Data

We evaluate performance with two different Bayesian networks from the Norsys Bayes Net Library\(^1\). One network is *Boerlage92* which has 23 variables and describes a particular scenario of neighborhood events. Most child-parent relationships are “Noisy-OR” and can be easily captured using pairwise testing. We try to predict variable *TW* (whether Tom is currently fairly wealthy) based on the remaining variables. Another network is *Win95pts* which has 76 variables and describes printer troubleshooting in Windows95. Most child-parent relationships are “Noisy-And” and are difficult to be captured using pairwise testing. We try to predict variable *NetOK* (whether the net is OK).

We draw a given number of samples from the network and apply the sparse candidate method with \(p=k=15\) to select features. We then use SVMs to evaluate the F1 value on 10000 sampled instances. As a comparison, we also apply the Grow-Shrink (GS) algorithm (Margaritis & Thrun 1999) to search the Markov blanket. The GS algorithm relies on the chi-square tests and is the basis of several other Markov blanket feature selection methods (Aliferis, Tsamardinos, & Statnikov 2003; Bai et al. 2004). We set nominal level \(\alpha\) to 5\%. We also use the information gain criterion (Yang & Pedersen 1997) to select the same number of features with the sparse candidate algorithm. The result averaged over 1000 trials is plotted in Figure 1. Note that for this experiment, using BDeu (Heckerman, Geiger, & Chickering 1995) and the screen-based algorithm give similar results.

When the interactions between variables are mainly high-order (the *Win95pts* problem), information gain has difficulty in finding the right feature set because it mainly relies on pairwise statistical measurements. The GS algorithm is quite unreliable when the data is limited and it needs much more samples to converge to a reasonable performance than our method. Its performance even decreases with more training data. Using the Markov blanket features of the learned Bayesian network gives better performance than using all features when the data is limited. However, the difference in performance decreases when we have more training data.

Experiments on Benchmark Text Data

We also test on 7 binary text classification problems. They are Student vs Faculty from the WebKB set, Corn vs Wheat, Corn vs Crude, and Interest vs FX from the Reuters-21578 set, Mac vs PC, Auto vs Motorcycle, and Baseball vs Hockey from the 20Newsgroup set. We remove the header of each document and convert the remainder into a 0/1 vector without stemming or a stoplist.

The result is presented in Table 2. By capturing a more accurate global view of how all the random variables interact with each other, selecting features through learning a Bayesian network beats other feature selection methods in most cases, and can slightly improve the prediction accuracy. On the contrary, the GS algorithm is usually too aggressive in reducing the feature dimensionality and thus hurts the performances. The information gain method has difficulty in finding high-order interactions and thus shows suboptimal results. The screen-based method beats the sparse candidate method in most cases because it has no restriction on its parent set. For the 20Newsgroup data which is difficult to classify due to the discursive writing style of documents, the accuracy of the screen-based algorithm is significantly better than any other methods. One exception is the WebKB data, in which we can accurately predict the category using just several words. For example, “graduate” and “advisor” are very good indicators for “student”.

Such high-dimensional data sets pose a challenge to most Bayesian network learning algorithms. For example, Optimal Reinsertion (Moore & Wong 2003) ran out of memory on every dataset, even if we set \(p=3\) and ran it on a machine with 4G memory. Our methods can learn a Bayesian network from such massive data extremely fast. The screen-based method can learn a Bayesian network within minutes. Depending on the data size, the 5 million random operations of the post-processing take about 15 minutes to 1 hour. Thus, our feature selection method through heuristic Bayesian network structure learning is quite efficient.

Conclusion and Discussion

We analyze the Markov blanket feature selection algorithms based on the independence test and present an efficient feature selection method that uses a fast, heuristic technique to learn a Bayesian network over a high-dimensional data set. There is no threshold tuning and it can automatically find the optimal feature set. Also, since Bayesian networks provide the dependency explanations between variables, this feature selection method is easy to incorporate prior expert knowledge and explain the learned results. We analyze the sufficient condition when this method will improve the performance of classifiers. Experimental results show that our method can efficiently reduce the feature dimensionality without much loss of classification accuracy. To our best knowledge, there has been no attempt to heuristically learn a Bayesian network from massive text data. We show that by restricting the search space and improving the efficiency of cycle detection, heuristic search methods can efficiently learn a Bayesian network from high-dimensional data.

Our feature selection method can be considered as a hybrid generative-discriminative learning approach. Raina et al. (Raina et al. 2004) propose a hybrid model in which a high-dimensional subset of the parameters are trained to maximize generative likelihood, and another small subset of

\(^1\)Available at http://www.norsys.com/netlibrary/index.htm
parameters are trained to maximize conditional likelihood. We are interested in comparing our approach with this algorithm. With a large amount of real-world data being stored in relational form, we would also like to investigate learning the structure of probabilistic relational models (Friedman et al. 1999) and construct predictive feature sets for a class variable given the learned model.

References