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_i 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 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 determine 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 α, it rejects H_0 if the chi-square statistic $\widetilde{\chi}^2 > \chi^2_{1,\alpha}$, where $\chi^2_{1,\alpha}$ is the upper α percentage point of the cumulative chi-square distribution with 1 degree of freedom. Let $\mathbf{S}^n$ denote a training set with n samples generated from probability distribution $\mathcal{D}$, $\mathcal{F}(\mathbf{S}^n)$ denote the feature set returned by applying feature selection algorithm $\mathcal{F}$ to $\mathbf{S}^n$. We denote the expected error rate of $\mathcal{F}$ given n samples as $Err^n(\mathcal{F})$. There could be many different definitions of $Err^n(\mathcal{F})$, and one possibility is $Err^n(\mathcal{F}) = \int_{\mathbf{S}^n \in \mathcal{D}} P(\mathbf{S}^n) \frac{|\Phi \otimes \mathcal{F}(\mathbf{S}^n)|}{|\Phi \cup \mathcal{F}(\mathbf{S}^n)|} dS^n$, where Φ is the optimal feature set, $U \otimes V$ is the difference between set U and V. Most Markov-blanket algorithms are *test-monotonic*:

Definition 1 *We call feature selection algorithm $\mathcal{F}$* test-monotonic, *if $\mathcal{F}$ relies on independence test T, and $Err^n(\mathcal{F}) = h(Err^n(T))$. Here h is a monotonically increasing function in which $h(x) > 0$ if $x > 0$, and $Err^n(T)$ is the expected error rate of applying T to n samples.*

Lemma 1 *A test-monotonic feature selection algorithm with fixed nominal level α is not asymptotically consistent, i.e., $Err^n(\mathcal{F}) > 0$ as $n \to \infty$.*

Proof It has been shown that type I error of T is strictly no less than its nominal level α (Loh 1989). Thus $Err^n(T) = P \cdot E_{\text{I}} + (1 - P) \cdot E_{\text{II}} \geq P \cdot E_{\text{I}} > 0$ as $n \to \infty$. Here P is the probability that two variables are independent, E_{I} is the type I error and E_{II} is the type II error. Based on the definition of test-monotone, $Err^n(\mathcal{F}) > 0$ as $n \to \infty$. ∎

Thus, T will mistakenly treat two independent variables as dependent with probability no less than α. In practice, usually $P > 0.5$ and people set α to 5% to control the type I error. With small α, it is very likely that T treats dependent variable as independent, which is shown in the following:

Lemma 2 *Given nominal level α, if we want the probability that T identifies two dependent variables as independent to be no more than ϵ, then we need $n > \frac{1}{\Delta}(\sqrt{\chi^2_{1,\alpha} + \frac{1}{4}U^2_\epsilon} + U_\epsilon)^2$ samples, where Δ is a constant depending on the correlation between the variables, and U_ϵ is the upper ϵ percentage point of the cumulative normal distribution.*

Proof (Sketch) We want to ensure type II error $P(\widetilde{\chi}^2 \leq \chi^2_{1,\alpha}) \leq \epsilon$. Under the alternative hypothesis, $\widetilde{\chi}^2$ 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 $n > \frac{1}{\Delta}(\sqrt{\chi^2_{1,\alpha} + \frac{1}{4}U^2_\epsilon} + U_\epsilon)^2$ after some operations. ∎

Note $\chi^2_{1,\alpha}$ and U_ϵ 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 Δ 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 α and even with infinite samples their error rates are still proportional to α. 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 $|G| = \sum_{i=1}^{m} 2^{|Pa_G(i)|}$ *is the number of parameters needed to specify graph G, where*

$Pa_G(i)$ are the parents of node X_i. For a graph G we denote by $I(G)$ the set of all conditional independence relations in G. G is an I-map *for optimal graph G^* if $I(G) \subseteq I(G^*)$.*

Note $|G| \geq |G^*|$, $|G|$ can grow very large by adding a few edges. We are interested in *promising* learning approaches:

Definition 3 *Given m variables, a Bayesian network structure learning algorithm $\mathcal{A}$ is called* π-promising *if $\mathcal{A}$ only evaluates $O(m)$ candidates, while the target structure G^* is always in the candidate set and there are only $O(1)$ I-map graphs such as $|G| - |G^*| < \pi$ in the candidate set.*

In other words, a π-promising algorithm only evaluates a few promising candidates and most candidates are different from the target graph with some edges. A larger π leads to the better performance. We say a data set is *feature-challenging* if more than half variables are irrelevant to the classification. SVMs, especially those with a complex kernel have some ability to select features, at the cost of more samples. As also shown in the experiments, our feature selection method can improve the performance of SVMs:

Proposition 1 *Given a feature-challenging data set with m variables, the performance of a linear SVM will be improved by using the features learned from a π-promising algorithm if there are $n > m^{1/(\pi-1)}$ samples.*

Proof (Sketch) We first show that a linear SVM discriminates irrelevant features at a speed no faster than $O(\sqrt{\frac{m}{n}}\log(\frac{n}{m}))$. Meanwhile, the probability that a specific non I-map graph outscores G^* decays exponentially with n, and the probability for a I-map graph decays as a power of n (Zuk, Margel, & Domany 2006). The π-promising algorithm converges to the optimal graph at a speed faster than $O(m(\log n)^{\frac{1}{2}\pi-1}n^{-\frac{1}{2}\pi})$ by the union bound. When $n > m^{1/(\pi-1)}$, this rate is faster than SVMs. Its selected feature can improve the performance of SVMs. ∎

A π-promising algorithm converges to the optimality very fast, at a speed faster than the $\frac{1}{2}\pi$ power of the sample size. A good heuristic learning method only evaluates a limited number (generally $O(m)$) of candidate structures. Although we usually can not guarantee G^* is in the candidate set, at least some promising candidates close to G^* are in the set. Thus we can think they are roughly π-promising. Test-based methods are quite unstable when the samples are limited. Especially, the probability that they treat dependent variables as independent are high and they are too aggressive in removing features. On the contrary, searching Markov blankets through Bayesian network is asymptotically consistent. More importantly, it needs much fewer examples to learn a reasonable feature set with some good heuristics.

Efficient Bayesian Network Searching

Learning Bayesian network structure is usually done using heuristic methods since optimization is a hard problem. We approximate π-promising learning by employing some good heuristics. They are quite stable even with limited samples and lead to reasonably good feature sets in general. Since in practice feature selection usually needs to deal with massive datasets, it is necessary to restrict the search space and improve the efficiency of searching the optimal operation.

Restricting the Search Space

The search procedure usually spends a lot of time examining extremely unreasonable candidate structures. We experienced two approaches to restrict the search space: one tries to restrict the candidate parents for each node (Friedman, Nachman, & Peér 1999), another tries to restrict the candidate edges for the entire graph (Goldenberg & Moore 2004).

The Sparse Candidate Algorithm Traditional heuristic search such as hill-climbing usually restricts the search to Bayesian networks in which each variable has at most p parents. Thus, there are $O(\sum_{i=0}^{i=p}\binom{|V|-1}{i})$ possible parent sets for each variable. The search space is still prohibitively large when $|V|$ is large. The sparse candidate algorithm uses statistical cues from the training data to restrict the possible parents of each variable. Instead of having $|V| - 1$ possible parents for a variable, we only consider k possible parents, and usually $k \ll |V|$. Thus there are only $O(\sum_{i=0}^{i=p}\binom{k}{i})$ possible parent sets for each variable. We then attempt to maximize the score with respect to these restrictions. This "Restrict-Maximize" process is iterated until the score has converged or we reach the maximal iteration steps.

There are several different ways to choose the promising parent candidates (Friedman, Nachman, & Peér 1999). In this paper we choose the candidate based on the score if we were to add that candidate as a parent. To illustrate this, suppose we are given variable X_i and the current DAG $G = (V, E)$. We compute S_{ij} which is the score of DAG $G' = (V, E \cup \{X_j \to X_i\})$ for each variable X_j. We then choose variables with the highest S_{ij} values as the candidate parents for variable X_i. To enforce the monotonic improvement of the score, we always include the current parent set in the candidate parent set.

The Screen-based Algorithm The sparse candidate algorithm reduces the search space by restricting the candidate parent set for each node. The maximum number of parents and the size of the candidate parent set are the same for any variable. However, it is likely that some variables are directly influenced by many variables while some variables are influenced by only a few variables. Thus, the sparse candidate algorithm could still examine many unreasonable candidate structures while on the other hand, some significant interactions between variables could be ignored. We use a variation of the screen-based Bayesian net structure search algorithm (Goldenberg & Moore 2004) to reduce the search space and only consider promising candidate edges.

The screen-based algorithm begins by collecting the co-occurrences between variables. If variable X_i and X_j co-occurs at least θ times, we search the best scoring DAG with node set $\{X_i, X_j\}$ explaining the data. If that best DAG has an edge, we store the edge and the score into an edge dump. After we go through all co-occurrences, we sort the edges in the edge dump in decreasing order of their scores. For each edge in the edge dump, if it will improve the network score and will not introduce a cycle, then we add it to the network. The edge dump is usually small and thus the searching is quite fast. We found this screen-based algorithm works surprisingly well in practice with some extra post-processing.

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 $x \overset{*}{\rightarrow} y$ 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 $y \rightarrow v_1 \rightarrow \ldots v_\ell \rightarrow x$, by the definition of topological ordering, we have $ord(y) < ord(v_1) < \ldots < ord(v_\ell) < ord(x)$. 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 δ^+_{xy} and δ^-_{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 δ_{xy} as $\delta^+_{xy} \cup \delta^-_{xy}$, where $\delta^+_{xy} = \{k \in AR_{xy} | y \overset{*}{\rightarrow} k\}$ and $\delta^-_{xy} = \{k \in AR_{xy} | k \overset{*}{\rightarrow} x\}$.*

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 $E(\delta^+_{xy}) = \{a \rightarrow b \in E | a \in \delta^+_{xy} \vee b \in \delta^+_{xy}\}$.

We should notice three important things when updating ord: 1) adding edge $x \rightarrow y$ need not influence the order of any node outside δ_{xy}, 2) in the new graph, the order of any node in δ^-_{xy} should have a smaller order than the order of any node in δ^+_{xy}, 3) for any nodes $a, b \in \delta^+_{xy}$, if $ord(a) > ord(b)$ in the old ordering, then it holds in the new ordering (the statement is also true for any node pair in δ^-_{xy}). There have been several algorithms on maintaining ord (Alpern *et al.* 1990; Pearce & Kelly 2006). In this paper, we apply the PK algorithm (Pearce & Kelly 2006) to maintain ord, which takes $\Theta(|E(\delta_{xy})| + |\delta_{xy}| \log |\delta_{xy}|)$.

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

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

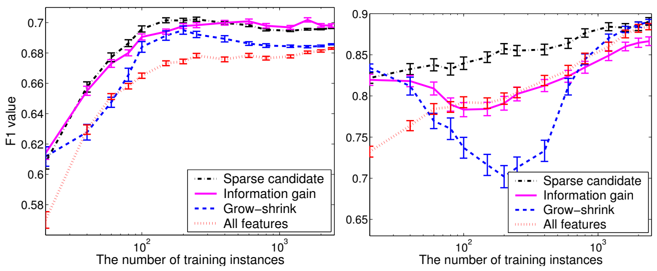
Experimental Results

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 θ=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*precision*recall/(precision+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 π-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 in Section). 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.



(a) Boerlage92 problem (b) Win95pts problem

Figure 1: Performance of feature selections given the training size, with 95% confidence intervals.

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 α 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 size. 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

Table 2: Performance on benchmark text datasets

(a) F1 values of different feature selections ($\times 100$)

Dataset	all features	GS	IG	SC(k:10, p:5)	SC(k:20, p:10)	Screen-based
WebKB	95.087	96.37	92.33	95.83	**96.42**	94.37
Corn vs Wheat	81.43	78.99	78.99	78.05	78.64	**81.82**
Corn vs Crude	99.73	98.95	99.73	99.47	99.73	**100**
Interest vs FX	78.69	76.24	**80.31**	76.53	77.72	79.47
Mac vs PC	83.82	76.69	84.19	83.29	84.14	**86.31**
Auto vs Motor	90.47	85.83	89.95	90.71	90.55	**91.34**
Baseball vs Hockey	90.97	83.68	91.94	90.92	92.62	**93.81**

(b) Number of selected features

Dataset	original	GS	SC 10	SC 20	Screen
WebKB	10850	8	98	109	601
C vs W	2589	5	6	6	24
C vs C	4323	5	48	59	421
I vs F	4191	5	9	9	62
M vs P	7212	7	55	61	322
A vs M	8859	7	74	80	411
B vs H	9483	7	119	128	849

(c) CPU time (without the post-processing)

Dataset	SC 10	SC 20	Screen
WebKB	85842	100472	431
C vs W	399	1056	23
C vs C	1627	3411	56
I vs F	1742	3306	201
M vs P	7965	12769	139
A vs M	13902	21549	281
B vs H	8189	12515	337

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

Aliferis, C. F.; Tsamardinos, I.; and Statnikov, A. 2003. HITON: A novel markov blanket algorithm for optimal variable selection. In *AMIA 2003 Symposium Proceedings*, 21–25.

Alpern, B.; Hoover, R.; Rosen, B. K.; Sweeney, P. F.; and Zadeck, F. K. 1990. Incremental evaluation of computational circuits. In *SODA-90*, 32–42.

Bai, X.; Glymour, C.; Padman, R.; Ramsey, J.; Spirtes, P.; and Wimberly, F. 2004. PCX: Markov blanket classification for large data sets with few cases. Report CMU-CALD-04-102, CMU.

Baker, L. D., and McCallum, A. K. 1998. Distributional clustering of words for text classification. In *SIGIR-98*, 96–103.

Chang, C.-C., and Lin, C.-J. 2001. *LIBSVM: a library for support vector machines*. Software available at http://www.csie.ntu.edu.tw/∼cjlin/libsvm.

Cormen, T. H.; Leiserson, C. E.; Rivest, R. L.; and Stein, C. 2001. *Introduction to Algorithms*. MIT Press, 2nd edition.

Friedman, N.; Getoor, L.; Koller, D.; and Pfeffer, A. 1999. Learning probabilistic relational models. In *IJCAI-99*, 1300–1309.

Friedman, N.; Nachman, I.; and Peér, D. 1999. Learning Bayesian network structure from massive datasets: The "sparse candidate" algorithm. In *Proc. of UAI-99*, 206–215.

Geiger, D.; Heckerman, D.; King, H.; and Meek, C. 2001. Stratified exponential families: graphical models and model selection. *The Annals of statistics* 29:505–529.

Giudici, P., and Castelo, R. 2003. Improving Markov Chain Monte Carlo model search for data mining. *Machine Learning* 50(1-2):127–158.

Goldenberg, A., and Moore, A. 2004. Tractable learning of large Bayes net structures from sparse data. In *ICML-04*, 345–352.

Heckerman, D.; Geiger, D.; and Chickering, D. M. 1995. Learning Bayesian networks: The combination of knowledge and statistical data. *Machine Learning* 20(3):197–243.

Joachims, T. 2001. *Learning to Classify Text Using Support Vector Machines*. Kluwer Publishers.

Johnson, N. I., and Kotz, S. 1970. *Distributions in Statistics. Continuous Distributions.* New York: Wiley.

Koller, D., and Sahami, M. 1996. Toward optimal feature selection. In *Proc. of ICML-96*, 284–292.

Loh, W.-Y. 1989. Bounds on the size of the chi-square-test of independence in a contingency table. *The Annals of statistics* 17(4):1709–1722.

Margaritis, D., and Thrun, S. 1999. Bayesian network induction via local neighborhoods. In *NIPS 12*, 505–511.

Meng, R. C., and Chapman, D. G. 1966. The power of chi square tests for contingency tables. *Journal of the American Statistical Association* 61:965–975.

Moore, A., and Wong, W.-K. 2003. Optimal reinsertion: A new search operator for accelerated and more accurate bayesian network structure learning. In *Proc. of ICML-03*, 552–559.

Ng, A. Y., and Jordan, M. I. 2002. On discriminative vs. generative classifiers: A comparison of logistic regression and naive Bayes. In *Advances in NIPS 14*.

Pearce, D. J., and Kelly, P. H. J. 2006. A dynamic topological sort algorithm for directed acyclic graphs. *J. Exp. Algorithmics* 11:1–24.

Raina, R.; Shen, Y.; Ng, A. Y.; and McCallum, A. 2004. Classification with hybrid generative/discriminative models. *NIPS 16*.

Witten, I. H., and Frank, E. 2005. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann.

Yang, Y., and Pedersen, J. O. 1997. A comparative study on feature selection in text categorization. In *ICML-97*, 412–420.

Zuk, O.; Margel, S.; and Domany, E. 2006. On the number of samples needed to learn the correct structure of a bayesian network. In *UAI*.