

# Bernoulli Mixture Models for Markov Blanket Filtering and Classification

**Mehreen Saeed**

MEHREEN.SAEED@NU.EDU.PK

*Department of Computer Science*

*National University of Computer and Emerging Sciences*

*Lahore Campus, Pakistan*

**Editors:** I. Guyon, C. Aliferis, G. Cooper, A. Elisseeff, J.-P. Pellet, P. Spirtes, and A. Statnikov

## Abstract

This paper presents the use of Bernoulli mixture models for Markov blanket filtering and classification of binary data. Bernoulli mixture models can be seen as a tool for partitioning an  $n$ -dimensional hypercube, identifying regions of high data density on the corners of the hypercube. Once Bernoulli mixture models are computed from a training dataset we use them for determining the Markov blanket of the target variable. An algorithm for Markov blanket filtering was proposed by Koller and Sahami (1996), which is a greedy search method for feature subset selection and it outputs an approximation to the optimal feature selection criterion. However, they use the entire training instances for computing the conditioning sets and have to limit the size of these sets for computational efficiency and avoiding data fragmentation. We have adapted their algorithm to use Bernoulli mixture models instead, hence, overcoming the short comings of their algorithm and increasing the efficiency of this algorithm considerably. Once a feature subset is identified we perform classification using these mixture models. We have applied this algorithm to the causality challenge datasets. Our prediction scores were ranked fourth on SIDO and our feature scores were ranked the best for test sets 1 and 2 of the same dataset.

**Keywords:** Markov blanket filtering, mixture models, feature selection

## 1. Introduction

The term Markov blanket was coined by Pearl (1988). The Markov blanket (MB) of a feature variable represents the set of features/attributes required to exactly predict the behavior of that variable. In a Bayesian network the Markov blanket consists of parents, children and spouses of the node representing that feature variable. The work described in this paper concentrates on identifying the MB of the target variable. Practically, it is not possible to identify the MB of the target variable exactly because of computational issues or lack of sufficient data. Koller and Sahami (1996) presented a feature selection algorithm called Markov Blanket filtering (MBF) which outputs an approximation to the MB of the target variable. Their algorithm can also output a sorted feature list according to the relevance of a feature with respect to the target variable. To run this algorithm we have to assume  $K$  which is the size of the conditioning sets. However, large  $K$  values are not possible due to computational issues.

In order to find the Markov blanket of target variable, we adapted the algorithm of Koller and Sahami to use Bernoulli mixtures so that large  $K$  values can be used. The use

of Bernoulli mixture models for classification is not new. The basic formula for a Bernoulli mixture model was first proposed by Duda and Hart (1973). They have been successfully used for OCR tasks by Juan and Vidal (2004) and Grim *et al.* (2000) and in supervised text classification tasks (Juan and Vidal, 2002). Bernoulli mixtures have also been used for supervised dimensionality reduction tasks (Sajama and Orlitsky, 2005).

Recently, we used Bernoulli mixtures for dimensionality reduction and showed how this transformed data can be used as input to a classifier giving rise to a hybrid model of learning (Saeed, 2008; Saeed and Babri, 2008). We used Bernoulli mixtures for mapping raw input data onto a new probability space. Such a transformation results in an immense reduction in the dimensionality of original data. It also achieves better classification results than individual models. The algorithm, described in this paper, calculates the entropy values from Bernoulli mixtures instead of calculating them from the actual training data. We employed this technique to the causality challenge datasets (WCCI, 2008a), i.e., SIDO and CINA datasets. Our results were amongst the top ranked entries in the competition.

The outline of this paper is as follows: In Section 2, Koller and Sahami’s MBF algorithm is presented. In Section 3, Bernoulli mixtures are briefly introduced and the Bernoulli mixtures based MBF algorithm is described. The simulation results on the causality challenge datasets are presented in Section 4 and finally the conclusions are given in Section 5.

## 2. Markov Blanket Filtering (MBF) By Koller and Sahami (1996)

Koller and Sahami (1996) presented an algorithm for Markov blanket filtering (MBF) which is a greedy search algorithm, for identifying the MB of the target variable, based upon an optimal feature selection criterion. We will briefly describe their algorithm in this section. Let  $\mathbf{F} = \{F_1, F_2, \dots, F_n\}$  be a set of features and  $\mathbf{f} = (f_1, f_2, \dots, f_n)$  be a corresponding assignment of values. Let  $\mathbf{G}$  be a subset of the feature set  $\mathbf{F}$  and  $\mathbf{f}_{\mathbf{G}}$  be a projection of  $\mathbf{f}$  onto the variables in  $\mathbf{G}$ . The MBF algorithm minimizes the divergence between  $P(C|\mathbf{F} = \mathbf{f})$  and  $P(C|\mathbf{G} = \mathbf{f}_{\mathbf{G}})$  using an expected conditional entropy measure given by:

$$\delta_{\mathbf{G}} = \sum_{\mathbf{f}} P(\mathbf{f}) D_{KL} \left( P(C|\mathbf{f}) \parallel P(C|\mathbf{f}_{\mathbf{G}}) \right)$$

where  $C$  is the class label and  $D_{KL}$  is the Kullback-Leibler (KL) divergence between the true distribution  $P(C|\mathbf{F} = \mathbf{f})$  and its estimated distribution  $P(C|\mathbf{G} = \mathbf{f}_{\mathbf{G}})$ . This divergence is given by  $D_{KL}(p \parallel q) = \sum_{x \in \Omega} p(x) \log \frac{p(x)}{q(x)}$ . The probability space  $\Omega$  is the set of all possible target labels. If we can find a feature set  $\mathbf{G}$  for which  $\delta_{\mathbf{G}}$  is very small then  $\mathbf{G}$  can be used as an approximation to the full feature set  $\mathbf{F}$  for predicting the target class.

Theoretically, Koller and Sahami have shown that a feature  $F_i$  can be omitted from a possible feature set  $\mathbf{G}$  by finding the Markov Blanket,  $\mathbf{M}$ , for  $F_i$ . If the MB for  $F_i$  can be identified then  $F_i$  can be safely removed from  $\mathbf{G}$  without an increase in the divergence from the true distribution. Practically, it is not possible to pinpoint exactly the MB of  $F_i$ , hence, heuristics have to be applied. The algorithm selects a candidate set  $\mathbf{M}_i$  for each feature  $F_i$  and estimates how close  $\mathbf{M}_i$  is to being the MB of  $F_i$ . The candidate set  $\mathbf{M}_i$  is composed of those features which have the highest correlation with  $F_i$ . The feature  $F_i$  for which  $\mathbf{M}_i$  is closest to being the MB is omitted. The approximation is based upon the

following expected cross entropy measure:

$$\delta_{\mathbf{G}}(F_i|\mathbf{M}_i) = \sum_{\mathbf{f}_{\mathbf{M}_i}, f_i} P(\mathbf{M}_i = \mathbf{f}_{\mathbf{M}_i}, F_i = f_i) D_{KL} \left( P(C|\mathbf{M} = \mathbf{f}_{\mathbf{M}}, F_i = f_i) \parallel P(C|\mathbf{M} = \mathbf{f}_{\mathbf{M}}) \right)$$

If  $\mathbf{M}_i$  is the MB for  $F_i$  then  $\delta_G(F_i|\mathbf{M}_i) = 0$  and this value will be small for an approximate MB. Algorithm 1 outlines the basic steps for selecting an approximate feature set for predicting the target variable. In this algorithm  $K$  determines the size of the conditioning set. We would like  $K$  to be as large as possible, however, practically larger values of  $K$  lead to fragmentation of the dataset and reduce the accuracy of the probability estimates used in estimating the cross entropy measure. Also, large values of  $K$  are not practical to implement as they require the counting of  $2^{(K+1)}$  combination of values to calculate the cross entropy measure. This algorithm can also be used to order variables, according to priority or relevance to target variable, based upon the cross entropy measure.

- 
- 
1. Let  $\mathbf{G} = \mathbf{F}$
  2. Repeat until desired number of features in  $\mathbf{G}$ 
    - a. For each feature  $F_i \in \mathbf{G}$ , let  $\mathbf{M}_i$  be the set of  $K$  features  $F_l \in \mathbf{G} - \{F_i\}$  which have the highest correlation with  $F_i$ .
    - b. Compute  $\delta_G(F_i|\mathbf{M}_i)$  for each  $i$ .
    - c. Choose the  $i$  for which this term is minimal and define  $\mathbf{G} = \mathbf{G} - \{F_i\}$
- 
- 

Algorithm 1: MBF Algorithm by Koller and Sahami (1996)

### 3. Multivariate Bernoulli Mixtures

Suppose we have a sample of training data,  $X = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_m\}$ , consisting of  $m$  input vectors. Each input vector  $\mathbf{x} \in R^n$ . If we want to estimate  $D$  mixture components from this data, then a finite mixture model is described by a probability (density) function given by  $p(\mathbf{x}) = \sum_{d=1}^D \pi_d p(\mathbf{x}|d)$ . Here,  $\pi_d$  is the prior of each mixture and  $p(\mathbf{x}|d)$  is its component-conditional probability (density) function.

A multivariate Bernoulli mixture model assumes that each component of the model is an  $n$ -dimensional multivariate Bernoulli probability distribution, each component or mixture having its own set of parameters. For a single binary vector  $\mathbf{x}_k \in \{0, 1\}^n$ , the form of this distribution, in the  $d^{th}$  mixture is given by (Bishop, 2006):

$$p(\mathbf{x}_k|d) = \prod_{i=1}^n (p_{di})^{x_{ki}} (1 - p_{di})^{1-x_{ki}} \quad \forall k, 1 \leq k \leq m, \forall d, 1 \leq d \leq D$$

Here  $p_{di} \in [0, 1]$  is the probability of success of the  $i^{th}$  component of vector  $\mathbf{x}_k$  for the  $d^{th}$  mixture, i.e.,  $p_{di} = p(x_{ki} = 1|d)$ . Also, we are assuming that the  $n$ -dimensional vector  $\mathbf{x}$  has  $n$  independent component attributes. The parameter  $\theta$  to be determined is the probability of success for each attribute of vector  $\mathbf{x}$ , i.e.,  $\theta = \mathbf{p}$  where  $\mathbf{p} \in [0, 1]^n$ .

We can use expectation maximization (EM) algorithm to find the parameters of each mixture component as described in Appendix A. Also, Appendix B explains the use of these mixtures for dimensionality reduction and classification. For details we refer the reader to our recent work in this area (Saeed, 2008; Saeed and Babri, 2008).

### 3.1 Adaptation of MBF algorithm Using Bernoulli Mixture Models

We found that the main problem with the MBF algorithm is in computing the expected cross entropy from training data. It is not possible to compute this measure using large values of  $K$  because of computational issues and problems with data fragmentation. However, we can compute an approximation to this measure via the use of Bernoulli mixture models. Bernoulli mixtures identify regions of high data density on the corners of a hypercube and we can exploit this fact to estimate the cross entropy measure for large  $K$  values.

Let's look at one Bernoulli mixture more closely. The parameters of the  $d^{th}$  Bernoulli mixture are completely specified by the probability vector  $\mathbf{p}_d$  and its prior  $\pi_d$ . Here, the  $i^{th}$  component of  $\mathbf{p}_d$ , i.e.,  $p_{di}$  represents the probability that the  $i^{th}$  binary feature is one in the  $d^{th}$  Bernoulli mixture. We can threshold these probability values to see which corner of the hypercube is represented by this mixture. A probability value greater than 0.5 can be taken as a one and zero otherwise.

As an example lets take a 3 feature case. The mixture (0.7, 0.9, 0.1) with prior  $\pi_d$  represents the corresponding feature vector (1, 1, 0). It shows us that within the dataset this mixture occurs  $\pi_d$  fraction of times and within this mixture, feature 1 is one with probability 0.7, feature 2 is 1 with probability 0.9 and feature 3 is 0 with probability 0.1. We can also estimate the probability of occurrence of this feature vector. If we make an optimistic estimate then we can say that feature 1 will be 1 when features 2 and 3 are 1 and 0 respectively at the most 70% of times. However, we can also say that feature 1 will be 1 at least 50% times when feature 2 and 3 are 1 and 0 respectively. So making an optimistic guess, we can estimate the probability of feature vector (1,1,0) to be at the most  $0.7 * \pi_d$ . This also tells us that (1, 1, 1) occurs at the most 10% times and similarly (0,1,0) occurs at the most 30% times.

We can formalize the above scheme and express the probability of feature vector,  $\mathbf{x}$ , when given the probability vector  $\mathbf{p}_d$  of the  $d^{th}$  mixture, as:

$$p(\mathbf{x}|d) = \pi_d * \prod_i p_{di}^{x_i} (1 - p_{di})^{1-x_i} \quad (1)$$

Suppose  $\mathcal{X}$  represents the set of all binary vectors in  $\{0, 1\}^n$ , hence,  $|\mathcal{X}| = 2^n$ . We will use the term 'main vector' for the feature vector that can be derived from a mixture density and has the highest probability of occurrence according to Eq. (1). Hence, 'main vector'  $\mathbf{v}$  is defined as:

$$\mathbf{v} = \arg \max_{\mathbf{x} \in \mathcal{X}} p(\mathbf{x}|d)$$

The main vector can also be derived by thresholding the probability values of a Bernoulli mixture with probability vector  $\mathbf{p}$ . The  $i^{th}$  feature of the main vector is assigned binary 1 value if its probability  $p_i \geq 0.5$  and binary 0 value otherwise.

The algorithm for finding the MBF is the same as Algorithm 1, except that Step 2b for calculating the cross entropy measure is replaced by Algorithm 2. Here, the cross entropy measure for a feature  $F_i$ ,  $1 \leq i \leq n$ , is approximated from the Bernoulli mixtures instead of training data. To run this algorithm, we first determine the Bernoulli mixtures, from the training data, for each class label separately.  $d_k^+$  and  $d_k^-$  are the  $k^{th}$  mixtures for the positive and negative classes ( $C^+$  and  $C^-$ ) respectively. The total mixtures,  $D^+$  and  $D^-$ , for positive and negative classes respectively, are specified as one of the initial parameters to the EM

algorithm and they don't have to be the same. Hence, the total mixtures generated from the entire training data is  $D = D^+ + D^-$ . Effectively, this algorithm finds the probability of each main vector within all Bernoulli mixtures of positive and negative classes. If the number of Bernoulli mixtures for each class is small then this can be done efficiently for large values of  $K$ .

---

Take all sub-vectors  $\mathbf{v}_j$ ,  $1 \leq j \leq D$ , of main vectors, in the mixture models found by EM so that  $\mathbf{v}_j \in \{0, 1\}^K$ , i.e.,  $\mathbf{v}_j$  is composed of all features that have the highest correlation with the target (corresponds to  $\mathbf{M}$  of Algorithm 1). As before,  $F_i$  is a feature and  $f_i$  is a corresponding assignment of value to  $F_i$ . In our case  $f_i \in \{0, 1\}$ . The cross entropy measure  $\delta'_G(F_i|\mathbf{M}_i)$  for each feature  $F_i$ ,  $1 \leq i \leq n$ , can be computed as:

1. Find the probability of  $\mathbf{v}_j$  in the positive class as  $p^+ = \sum_{k=1}^{D^+} \pi_k p(\mathbf{v}_j | d_k^+)$
2. Find the probability of  $\mathbf{v}_j$  in the negative class as  $p^- = \sum_{k=1}^{D^-} \pi_k p(\mathbf{v}_j | d_k^-)$
3. Calculate  $P(C^+|\mathbf{v}_j) = \frac{p^+}{p^+ + p^-}$ , assuming equal priors for both classes and similarly calculate  $P(C^-|\mathbf{v}_j)$
4. Repeat 1,2,3 to calculate  $P(C^+|\mathbf{v}_j, f_i = 0)$  and  $P(C^+|\mathbf{v}_j, f_i = 1)$
5. Sum over all sub-vectors and all values of  $f_i$  to calculate cross entropy measure,  $\delta'_G(F_i|\mathbf{M}_i)$ , approximated by:  $\sum_{l \in \{0,1\}} \sum_{j=1}^D D_{KL}(P(C|\mathbf{v}_j, f_i = l) || P(C|\mathbf{v}_j))$

---

Algorithm 2: Approximating the cross entropy measure using Bernoulli mixtures

**Time Complexity:** Section 4.1 shows that this algorithm has particular advantage when total training examples and features are very large in a dataset. If we don't count the initial step of computing correlation matrix and  $D$  Bernoulli mixtures then subsequent steps of the algorithm have roughly a time complexity of  $O(rnKD^2c)$ , where  $r$  is the number of features to eliminate,  $n$  is the total features and  $c$  is the total number of classes. In contrast, Koller and Sahami's algorithm for  $m$  training examples takes  $O(rnKm2^Kc)$  as the cross entropy measure is being computed from  $m \times K$  sized data and we need to look at  $2^K$  combination of values. However, we are computing our cross entropy measure from  $D \times K$  sized data, where we are only looking at  $D$  main vectors in each Bernoulli mixture, resulting in a dramatic reduction in time.

## 4. Simulations

To get an insight of MBF using Bernoulli mixtures we applied it to the SIDO and CINA datasets of the causality challenge. Each dataset has three types of test sets corresponding to the same training data. Test sets with subscript 0 are un-manipulated datasets and have the same distribution as the training data. Datasets with subscripts 1 and 2 have features that have been manipulated by some external agent and can have distracters or probes. Details can be found on the challenge's website (WCCI, 2008a) and are summarized in Table 1. The identity of each feature was not revealed to the challenge participants.

SIDO dataset is a pharmacology dataset that has only binary features. The features denote descriptor molecules which have been tested against AIDS HIV virus. The labels of each instance indicate molecular activity. Some of the features are actual molecular

descriptors and some of them are artificially generated probes. The identity of each feature was not revealed to the contestants of the challenge. CINA dataset is an econometrics dataset derived from UCI machine learning repository ‘Adult’ dataset. In this case also, the details of each feature were not revealed to us.

The evaluation of results was based upon the average accuracy, ‘Tscore’, of the three test sets for each dataset (Guyon et al., 2008). The contestants had the option of submitting a sorted nested feature subset list of  $r$  ( $r \leq n$ ) features, sorted according to the importance of each feature in predicting the target. Hence, multiple predictions could be made using the first  $s$  features, where  $s$  varies by powers of 2, i.e., 1, 2, 4, 8, ... features. An ‘Fscore’ was computed to indicate how well the predicted set of features matched with the actual MB of the target variable (known only to the organizers) (Guyon et al., 2008). For our submissions, we submitted a sorted list of features ranked according to a feature’s relevance in predicting the target, and a set of predictions corresponding to the nested feature subsets. Submitting a set of predictions gives us an added advantage over those participants who submitted only a single set of predictions.

During the challenge we made 29 complete entries on SIDO and 16 complete entries on CINA dataset. A complete entry consists of predictions on all three test files of a dataset (i.e., subscript 0, 1 and 2). Once we submitted an entry, we got immediate feedback on our performance, whether our entry was in the first, second, third or fourth quartile, as compared to the other participants. Only our last submitted entry was considered for final ranking. We used the quartile information and 2 fold cross validation accuracy to determine the final models for the last entry. Our entire source code was written in C++ and Matlab in the CLOP framework (Saffari and Guyon, 2006).

Dataset	Domain	Continuous Features	Binary Features	Positive Examples	Negative Examples	Total
SIDO	Pharmacology	0	4932	452	12226	12678
CINA	Econometrics	24	108	3939	12094	16033

Table 1: SIDO and CINA challenge datasets

#### 4.1 Results on SIDO, The Pharmacology Dataset

We generated Bernoulli mixtures on the SIDO dataset. The initial values specified to the algorithm for number of Bernoulli mixtures was 10 for both classes, however, we ended up with 4 and 7 Bernoulli mixtures for the positive and negative class respectively. There were 16,033 training examples in this set and we were unable to run Koller and Sahami’s algorithm on this set for  $K \geq 6$  due to limited computing resources available to us. However, we easily ran it for large values of  $K$  with the Bernoulli mixture version of MBF as there are only 11 mixtures and the combinations needed to compute cross entropy values is immensely smaller. Hence, we compute the MB from 11x4932 sized data instead of the full 16033x4932 sized data. The feature scores along with test scores on different values of  $K$  for the SIDO0, SIDO1 and SIDO2 datasets are given in Table 2.

For this dataset we returned our predictions on nested feature subsets, as described in Section 4. The challenge organizers chose the feature set for which the classification

K	SIDO0			SIDO1			SIDO2		
	Fnum	Fscore	Tscore	Fnum	Fscore	Tscore	Fnum	Fscore	Tscore
10	1024	0.4717	0.9332	1024	0.6895	0.7140	1024	0.6895	0.6145
15	1024	0.4857	0.9407	1024	0.6928	0.7464	128	0.6928	0.7155
20	512	0.4831	0.9457	2048	0.6635	0.6565	1024	0.6635	0.6134
25	512	0.4921	0.9381	1024	0.7335	0.7124	1024	0.7335	0.6325
30	1024	0.4592	0.9434	4096	0.7242	0.7246*	512	0.7242	0.6216
40	4096	0.4095	0.9391*	4096	0.694	0.7309*	4096	0.694	0.5788*

Table 2: Results on the SIDO dataset. Fnum is the number of features, Fscore is the score on the feature set (see Section 4), Tscore is the accuracy on the test set. The entries marked by ‘\*’ were classified using the naive Bayes’ classifier. For the rest of the entries the classifier used was a combination of Bernoulli mixtures and ensemble of neural networks.

accuracy was maximum and hence, this decides the ‘fnum’ value in Table 2. Depending upon the feature subset we used two different classifiers. Either the naive Bayes’ classifier or a combination of Bernoulli mixtures and ensemble of neural networks was used to classify data (see Appendix B). In an ensemble of neural networks, the overall prediction was made by combining outputs from individual neural networks (Saffari and Guyon, 2006). We used maximum of 10 neural network models, each trained with a different number of neurons in the hidden layer.

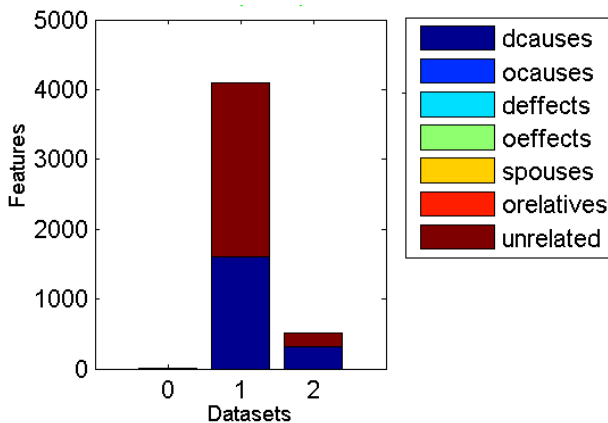


Figure 1: Profile of features selected for SIDO. Legend: dcause=direct cause, defect=direct effects, ocauses=other causes (indirect), oeffects=other effects (indirect), spouses=parent of a direct effect, orelatives=other relatives, unrelated=completely irrelevant (see WCCI, 2008b)

Our final entry for SIDO1 and SIDO2 was made using  $K = 30$ . It has the best Fscores amongst all ranked entries in the challenge on SIDO1 and SIDO2. Figure 1 shows the profile of features selected with  $K = 30$ . It is interesting to note that the algorithm finds only

the direct causes of target and irrelevant features in case of SIDO1 but the accuracy of the classifier is quite good even in the presence of the irrelevant features. However, in case of SIDO2, the algorithm is able to find out a higher fraction of direct causes as compared to irrelevant features and the classifier performs quite well.

The classification accuracy on SIDO0 is almost the same for all values of  $K$ . Interestingly, the performance of a simple naive Bayes' classifier and the hybrid model is almost identical on SIDO0 and SIDO1. The best classification results are obtained for  $K = 15$  for both SIDO1 and SIDO2 where the algorithm gives best results using 1024 and 128 features respectively and classification was done using a hybrid model of Bernoulli mixtures and an ensemble of neural networks.

## 4.2 Results on CINA, The Econometrics Dataset

No.	CINA0			CINA1			CINA2		
	Fnum	Fscore	Tscore	Fnum	Fscore	Tscore	Fnum	Fscore	Tscore
1	32	0.5069	0.9751	16	0.7858	0.8248	16	0.7858	0.6867
2	21	0.4875	0.9727	21	0.5196	0.8188	16	0.5196	0.6627

Table 3: Results on the CINA dataset. See Table 2 for explanation of Fnum, Fscore and Tscore

Table 3 shows the results obtained on two of our submitted entries for CINA dataset. The first row shows the feature scores and accuracy of the three test sets. Here, feature subset selection algorithm was used where forward selection was done based on the accuracy achieved by the Naive Bayes' classifier on the training set. The second row shows the results of feature selection when MBF using Bernoulli mixtures was used for selecting binary features and feature subset selection was used for selecting continuous features. Here, the total features used include 6 continuous features and the rest are binary. For this dataset classification was performed using an ensemble of neural networks on the selected feature

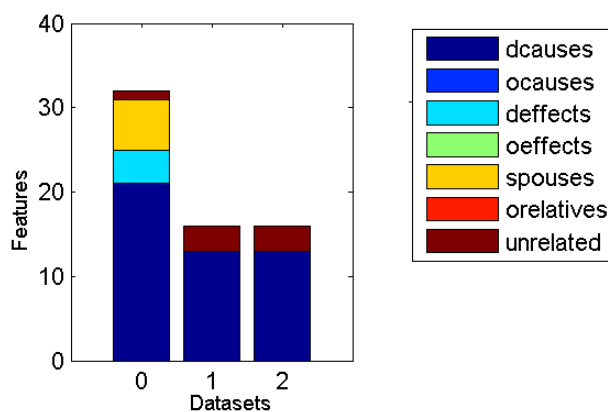


Figure 2: Profile of features selected for CINA. For legend see Figure 1 (WCCI, 2008b)



values after standardizing the data. We can see that the two methods have almost the same percentage accuracy on the three types of datasets, however, feature subset selection gives much better feature scores.

Figure 2 shows the profile of features selected for CINA on our last entry. It is interesting to see that our forward selection algorithm with naive Bayes’ is able to find many direct causes of the target variable for all three datasets. The percentage of irrelevant features identified is very small compared to the direct causes that have been identified. For dataset 0 this algorithm is able to identify many indirect causes of the target and also the parents of the direct causes.

### 4.3 Comparison and Discussion of Results

Dataset	K=15		Last Submission		Top ranking	Max Test Score
	Features	TScore	Features	Tscore		
SIDO0	1024	0.9407	8	0.9391	0.9443	0.9467
SIDO1	1024	0.7464	4096	0.7246	0.7532	0.7893
SIDO2	128	0.7155	512	0.6216	0.6684	0.7674
	K=3					
CINA0	21	0.9727	32	0.9751	0.9765	0.9788
CINA1	21	0.8188	16	0.8248	0.8691	0.8977
CINA2	16	0.6627	16	0.6867	0.8157	0.891

Table 4: Comparison of results. Top ranking is the Tscore of the winning participant. Max Test score is the highest achieved using actual knowledge of causal features.

Table 4 shows a comparison of results with the top ranking entries on both SIDO and CINA datasets. The table shows our results with  $K = 15$  on SIDO dataset, results with our last submitted entry and test scores of the winning participants. The column with the ‘Max Test Score’ indicates the best score reachable, as estimated by reference entries, using the knowledge of true causal relationships, not available to participants. Our last entry counted towards the ranking of participants and was ranked fourth. The average test score with  $K=15$  was the best accuracy rate amongst all participants, however, it did not count towards the final ranking as it was not our last submitted entry. The original dataset had 4932 features and we can see from the table that we were able to make quite good predictions using a very small set of features.

In case of CINA, we attained good results by using only 21 features on CINA0 and CINA1 and 16 features for CINA2 for  $K = 3$ . Also, our last entry was made using subset feature selection, with the forward selection algorithm with Naive Bayes’ algorithm (Alpaydin, 2005). Here the classifier used was an ensemble of neural networks. We can see that the two methods give almost the same accuracy, even though, they use a different feature set. Our test set accuracies are quite comparable with the top ranking participant for CINA0 and CINA1. However, our algorithm failed to produce good results for CINA2. Our last submitted entry was ranked sixth amongst all other entries.

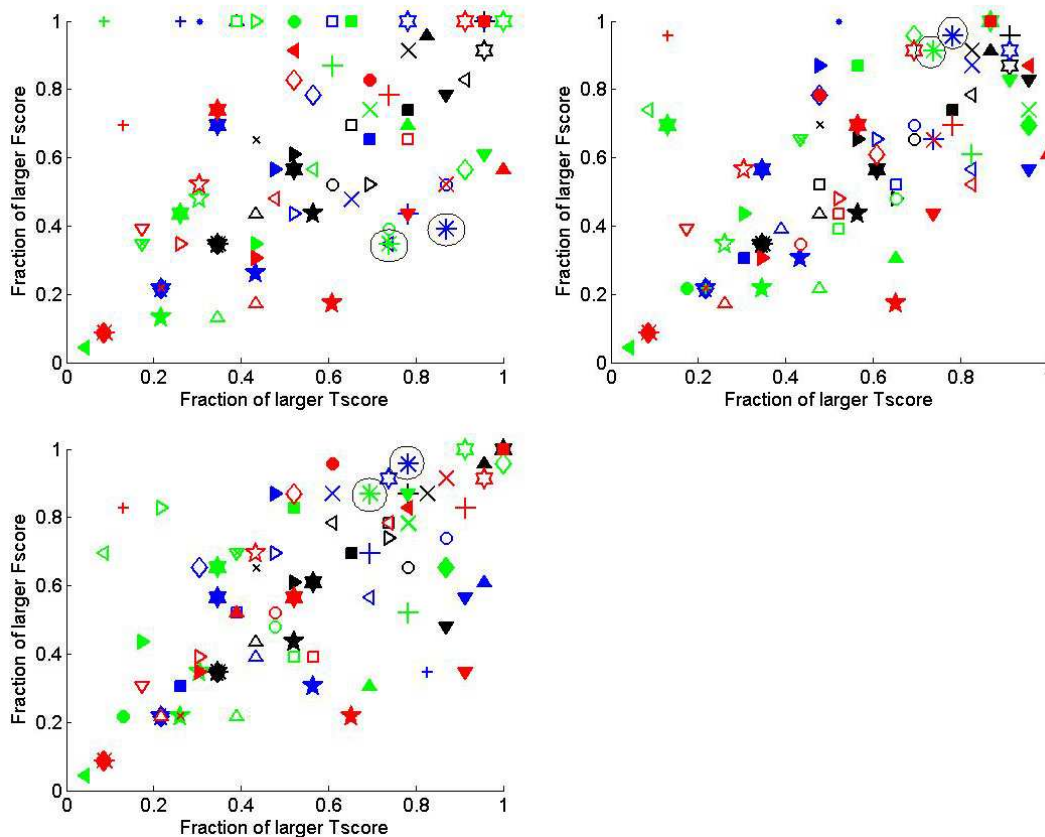


Figure 3: Pairwise performance of participants on test sets 0, 1 and 2. Green is SIDO and blue is CINA (see WCCI, 2008b). Our entry is encircled and denoted by ‘\*’.

Our feature scores (Fscores) were the best amongst all participants for test sets 1 and 2 of SIDO dataset and also for test sets 1 and 2 of the CINA dataset (Guyon et al., 2008). In order to further compare performances, the challenge organizers made pairwise comparisons between different challenge participants and counted the fraction of times our Tscore was better than other participants for a fixed number of features (Guyon et al., 2008). They did the same for Fscores also. Figure 3 shows the pairwise comparison. We have encircled our entries in black. Our feature scores are not very good for the test set 0, i.e., the data with no manipulations, but SIDO has a high Tscore as compared to the rest of the entries. Our feature scores for test sets 1 and 2 are significantly better than the rest of the participants on both SIDO and CINA datasets. Also, our Tscore is higher than the rest of the participants more than 70% of the times for test sets 1.

To further evaluate the results of the challenge, the organizers defined a new Fscore. The new Fscore was defined using the 3 versions of precision and recall, using as a set of features, the MB in set 1, MB and causes and effects in set 2 and all variables connected to the target in set 3 (Guyon et al., 2008). Figure 4 shows the plot of Tscore vs. the new Fscore of different participants for SIDO and CINA datasets. Again our entries are

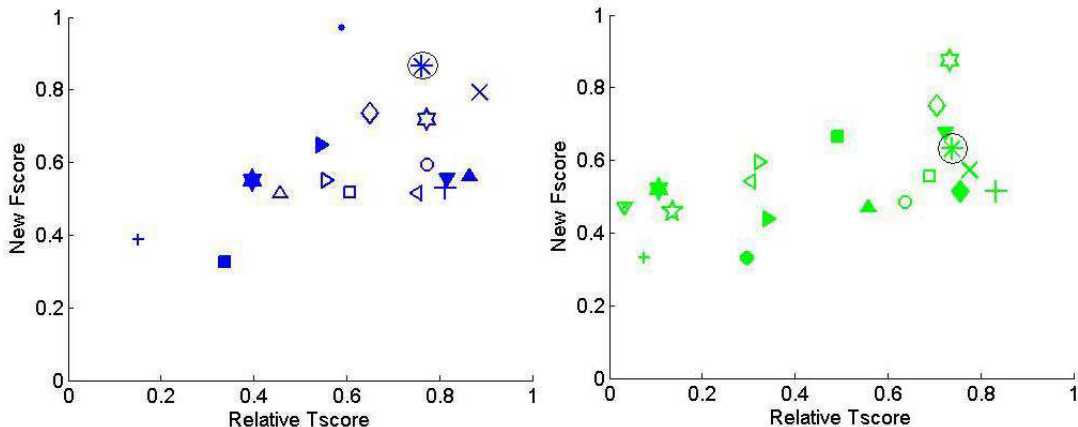


Figure 4: TScores Vs. new FScore for all participants for CINA (left) SIDO (right) datasets (see WCCI, 2008b). Our entry is encircled and denoted by ‘\*’.

encircled in black. It can be seen that we do quite well on CINA dataset as compared to the rest of the participants and our results are considerably better than other participants for the SIDO datasets.

## 5. Conclusions

In this paper we discussed the use of Bernoulli mixture models for Markov Blanket filtering by adapting the original algorithm proposed by Koller and Sahami in 1996. Instead of using the training data for estimating entropy values we have used Bernoulli mixtures to approximate these values. Hence, in this way we can reduce the computations required for approximating the cross entropy values and use larger conditioning sets. We tried our method on the SIDO and CINA datasets of the causality workbench challenge. Our method was ranked fourth on SIDO and sixth on CINA. Our feature scores were ranked best on the test sets 1 and 2 of both SIDO and CINA. Also, one of our entries, submitted during the challenge, had the best results on SIDO, out of all challenge entries, but did not count towards the final ranking as it was not our last submitted entry.

As part of future work we would like to extend our algorithm to do causal feature selection. Right now we determine only the MB of the target variable but do not pin point the causes or effects of the target variable. Also, we are working on developing methods that determine a suitable value of  $K$  for a particular dataset so that model selection techniques can be applied to datasets that have probes or noise added to them.

## Acknowledgments

Many thanks to Prof. Haroon Babri and Mr. Kashif Javed for their valuable comments and discussion during the challenge. The author would also like to thank Isabelle Guyon for her help and support during the causality challenge.

## Appendix A. The EM Algorithm for Learning Bernoulli Mixtures

A finite mixture model is described by a probability function given by  $p(\mathbf{x}) = \sum_{d=1}^D \pi_d p(\mathbf{x}|d)$ . Learning the parameters of a finite mixture model is a statistical parameter estimation problem and we can use expectation maximization (EM) algorithm to estimate these parameters from a sample of training data  $X = \{\mathbf{x}_k\}_{k=1}^m, \mathbf{x} \in R^n$ . The EM algorithm maximizes the log likelihood function of data given by:

$$\mathcal{L}(\Theta|X) = \sum_{k=1}^m \log \left( \sum_{d=1}^D \pi_d p(\mathbf{x}_k|d) \right) \quad (2)$$

Here  $\Theta$  denotes the unknown variables to be estimated and consists of the priors,  $\pi_d$ , of each mixture and the parameters,  $\theta_d$ , of each mixture distribution, i.e.,  $\Theta = \{\pi_d, \theta_d\}_{d=1}^D$ .

A Bernoulli mixture model assumes that each component of the model is an  $n$ -dimensional multivariate Bernoulli probability distribution, each component or mixture having its own set of parameters. The form of this distribution for a single vector  $\mathbf{x}_k \in \{0, 1\}^n$  in the  $d^{th}$  distribution is given by (Bishop, 2006):

$$p(\mathbf{x}_k|d) = \prod_{i=1}^n (p_{di})^{x_{ki}} (1 - p_{di})^{1-x_{ki}} \quad (\forall k, 1 \leq k \leq m, \forall d, 1 \leq d \leq D)$$

Here  $p_{di} \in [0, 1]$  is the probability of success of the  $i^{th}$  component of vector  $\mathbf{x}_k$  for the  $d^{th}$  mixture. Here the parameter  $\theta$  to be determined is the probability of success of each attribute of vector  $\mathbf{x}$ , i.e.,  $\theta = \mathbf{p}$  where  $\mathbf{p} \in [0, 1]^n$ .

The EM algorithm assumes that the observed data is incomplete and associates a vector of latent variables  $\mathbf{z}_k = \{z_{k1}, z_{k2}, \dots, z_{kD}\}$  with each data point. The latent variables are indicator variables, with  $z_{kd} = 1$  indicating that the  $d^{th}$  mixture component generated the  $k^{th}$  data point. The EM optimization takes place iteratively in two steps. In step 1, also called the expectation step (E-Step), we estimate the expected values of the hidden variables assuming that the model parameters  $\theta_d$  are known. In step 2, also called the maximization step (M-Step), we estimate the parameter values  $\theta_d$  to maximize the likelihood of data, given by Eq. (2), on the basis of the latent variables calculated in the E-step. This is done iteratively until the parameters converge to stable values. To start the EM algorithm we initialize the probabilities with random values.

The form of E-step is the same for more or less all distributions and it is given by:

$$z_{kd} = \frac{\pi_d p(\mathbf{x}_k|d)}{\sum_{j=1}^D \pi_j p(\mathbf{x}_k|j)} \quad (\forall d, 1 \leq d \leq D, \forall k, 1 \leq k \leq m) \quad (3)$$

The M-step determines the maximum likelihood estimate of the priors, of each distribution, as given below:

$$\pi_d = \frac{1}{m} \sum_{k=1}^m z_{kd} \quad (\forall d, 1 \leq d \leq D)$$

Also, in this step the parameters of the particular probability distribution are estimated. These parameters depend on the probability function being used. For a Bernoulli mixture

model, the M-step finds the maximum likelihood estimate of the probability of success of each vector component as given below:

$$\mathbf{p}_d = \frac{\sum_{k=1}^m z_{kd} \mathbf{x}_k}{\sum_{k=1}^m z_{kd}} \quad (\forall d, 1 \leq d \leq D)$$

The simulations, described in this paper, use the regularized version of EM algorithm described by Li *et al.* (2005). Also, Laplacian prior can be used to smooth the probability estimates, hence, the probability values are estimated as below ( $\gamma$  is the regularization constant):

$$\mathbf{p}_d = \frac{1 + \sum_{k=1}^m z_{kd}(1 + \gamma \ln z_{kd}) \mathbf{x}_k}{2 + \sum_{k=1}^m z_{kd}(1 + \gamma \ln z_{kd})}$$

The parameter  $D$ , i.e., the total mixtures to be determined is input to the algorithm by the user. We determine this number through cross-validation. Normally, we end up with fewer mixtures than originally specified, as the priors for many mixtures converge to zero and we can ignore those mixtures. This is especially true for sparse binary data with a high dimensional feature space. In such a case, no matter what the original value of  $D$ , specified to the regularized EM algorithm, we always end up with more or less the same number of mixtures with non-zero priors. Hence, the regularized version of EM algorithm is able to find an optimal number of clusters within data.

## Appendix B. Classification Using Bernoulli Mixtures

In this section we give a brief overview of how we use Bernoulli mixtures for feature transformation and classification of instances. For details, we refer the reader to our previous work (Saeed, 2008; Saeed and Babri, 2008). We extend our terminology to include the class labels assigned to each example point. Suppose we have a set of  $c$  labeled classes  $Q = \{q_1, q_2, \dots, q_c\}$ . We generate a total of  $D_i$  mixtures for class  $q_i$ . Let  $s_{id}$  represent the  $d^{\text{th}}$  mixture/cluster in the  $i^{\text{th}}$  class with prior for that mixture being given by  $\pi_{id}$  and  $\sum_{d=1}^{D_i} \pi_{id} = 1$ . Then the class conditional probability function, for the  $i^{\text{th}}$  class having  $D_i$  mixture components (given by  $\{s_{id}\}_{d=1}^{D_i}$ ), is given by:

$$p(\mathbf{x}|q_i) = \sum_{d=1}^{D_i} \pi_{id} p(\mathbf{x}|s_{id})$$

Given the original dataset  $X$ , we perform a transformation on  $X$  given by:  $T : X \rightarrow \Phi, X \in R^n, \Phi \in R^N$ . The  $l^{\text{th}}$  component of the new feature vector  $\phi_k$  is given by:

$$\phi_{kl} = \frac{p(\mathbf{x}_k|s_{id})\pi_{id}}{\sum_{i=1}^c \sum_{j=1}^{D_i} \pi_{ij} p(\mathbf{x}_k|s_{ij})} \quad (\forall k, 1 \leq k \leq m)$$

The index  $l$  has a value corresponding to every mixture of every class. For a sub-cluster  $s_{id}$ , the subscript  $l$  is given by  $l = \sum_{j=1}^{i-1} D_j + d$ . Here  $\phi_k \in R^N$ , where  $N = \sum_{i=1}^c D_i$ . We can see that the numerator in the above expression is the same as that of Eq. (3) which represents the expectation of the latent variable in a mixture. The term in the denominator is used to normalize the entire feature vector over all the classes. If  $N < n$  then our new

feature set will lie in a lower dimensional space as compared to the original feature space and hence, this method can be used for dimensionality reduction.

We use the expectation of the latent variables as the transformed set of features for input to a classification algorithm. The feature vector is normalized so that the sum of components is unity. We have shown in our previous work (Saeed, 2008; Saeed and Babri, 2008) that these set of transformed features lead to an immense reduction in the dimensionality of data, especially when the data is sparse. We can apply any classification algorithm to classify the data, e.g., SVM, neural networks, boosting models, etc. The work described in this paper uses an ensemble of neural networks.

## References

- Ehem Alpaydin. *Introduction to Machine Learning*. Prentice-Hall of India Private Limited, 2005.
- Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.
- Richard O. Duda and Peter E. Hart. *Pattern Classification and Scene Analysis*. Wiley, 1973.
- Jiri Grim, Pavel Pudil, and Petr Somol. Multivariate structural Bernoulli mixtures for recognition of handwritten numerals. In *Proceedings of International Conference on Pattern Recognition (ICPR'00)*, 2000.
- Isabelle Guyon, Constantin Aliferis, Greg Cooper, Andre Elissee, Jean-Philippe Pellet, Peter Spirtes, and Alexander Statnikov. Design and analysis of the causation and prediction challenge. In *Proceedings of Journal of Machine Learning Research*, 2008. to appear.
- Alfons Juan and Enrique Vidal. On the use of Bernoulli mixture models for text classification. *Pattern Recognition*, 35(12):2705–2710, December 2002.
- Alfons Juan and Enrique Vidal. Bernoulli mixture models for binary images. In *Proceedings of 17th International Conference on Pattern Recognition (ICPR04)*, 2004.
- Koller and Sahami. Toward optimal feature selection. In *Machine Learning: Proceedings of the 13<sup>th</sup> international conference*. Morgan Kaufman, 1996.
- Haifeng Li, Keshu Zhang, and Tao Jiang. The regularized EM algorithm. In *Proceedings of the 20th National Conference on Artificial Intelligence*, pages 807–812, 2005.
- Judea Pearl. *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufmann, 1988.
- Mehreen Saeed. *Hands-on pattern recognition challenges in data representation, model selection, and performance prediction*, chapter Hybrid learning using mixture models and artificial neural networks. 2008. To appear, see <http://www.clopinet.com/ChallengeBook.html>.
- Mehreen Saeed and Haroon Babri. Classifiers based on Bernoulli mixture models for text mining and handwriting recognition. In *Proceedings of International Joint Conference on Neural Networks, IEEE WCCI*, 2008.

Amir Saffari and Isabelle Guyon. *Quick start guide for CLOP*, May 2006. Available at <http://ymer.org/research/files/clop/QuickStartV1.0.pdf>.

Sajama and Alon Orlitsky. Supervised dimensionality reduction using mixture models. In *Proceedings of the 22nd international conference on machine learning*, pages 768–775, Bonn, Germany, 2005.

IEEE WCCI. Causality challenge #1: Causation and prediction, 2008a. See <http://www.causality.inf.ethz.ch/challenge.php>.

IEEE WCCI. Causation and prediction: Challenge analysis, 2008b. See <http://clonet.com/isabelle/Projects/WCCI2008/Analysis.html>.