JMLR Workshop and Conference Proceedings 6:225–236 NIPS 2008 workshop on causality

The Use of Bernoulli Mixture Models for Identifying Corners of a Hypercube and Extracting Boolean Rules From Data

Mehreen Saeed MEHREEN.SAEED@NU.EDU.PK
Department of Computer Science
National University of Computer and Emerging Sciences
Lahore Campus, Pakistan

Editor: Isabelle Guyon, Dominik Janzing and Bernhard Schölkopf

Abstract

This paper describes the use of Bernoulli mixture models for extracting boolean rules from data. Bernoulli mixtures identify high data density areas on the corners of a hypercube. One corner represents a conjunction of literals in a boolean clause and the set of all identified corners, of the hypercube, indicates disjuncts of clauses to form a rule. Further class labels can be used to select features or variables, in the individual conjuncts, that are relevant to the target variable. This method was applied to the SIGNET dataset of the causality workbench challenge. The dataset is derived from a biological signaling network with 21 time steps and 43 random boolean variables. Results indicate that Bernoulli mixtures are quite effective at extracting boolean rules from data.

Keywords: boolean networks, boolean rules, Bernoulli mixture models, feature selection

1. Introduction

Recently, the causality workbench team launched a challenge that involved many tasks related to causal discovery (see http://www.causality.inf.ethz.ch/pot-luck.php). One of the tasks, called 'SIGNET', was to discover boolean rules from raw data. The data was generated by the simulation of boolean rules, which describe the interaction of various variables in a boolean network representing a plant signaling network (Li et al., 2006). This is an interesting problem involving not only the modeling of time series data but also developing feature selection algorithms to explain the causes of a target variable.

The SIGNET data was derived by simulating an asynchronous boolean network. Boolean networks find important applications in many areas, especially for modeling biological systems such as plant signaling networks, genetic regulatory networks, signal transduction pathways, etc. In the past many scientists have modeled boolean networks using various techniques. A well established method was developed by Ideker et al. (2000)for inferring a genetic network from gene expression data. They used a method, called the predictor method, to infer a set of genetic networks. The genetic networks were derived using the concept of minimum set covering. A 'chooser' method, based upon entropy, was then used to discriminate between various networks, generated by the predictor. The predictor and chooser were used iteratively to derive the final gene network.

Other methods for modeling regulatory networks include graphical models like Bayesian networks (Friedman et al., 2000), which are directed graphs representing joint probabilities of the variables in the network and also captures the conditional independences between them. Neural networks have also been used for gene expression analysis and representing regulatory relationships between genes in a genetic network (Weaver et al., 1999). Such networks have also been constructed using the information theoretic measure, i.e., mutual information criterion between the input and output states (Liang et al., 1998).

This paper describes a novel technique for extracting boolean rules from the 'SIGNET' dataset by converting a probabilistic model of variable relationships into a rule based system. The probabilistic model is governed by a mixture of Bernoulli distributions. Mixture densities determine different groups or clusters, within data, based upon the various observational characteristics of data. They are ideal for estimating the overall probability distribution of data when the data is not uni-modal. The use of Bernoulli mixture models in machine learning and pattern 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). Prior to this work we used them for dimensionality reduction and feature selection (Saeed, 2008; Saeed and Babri, 2008) tasks.

Bernoulli mixture models identify areas of high data density in the form of probability vectors. We threshold these probabilities to obtain points that lie on the corners of a hypercube. Each corner of the hypercube represents a clause containing the conjunction of literals in a binary rule. Together the set of different corners specify the disjunction of various clauses to specify a complete rule. For feature selection within these rules we partition the data into 0 and 1 class labels and generate Bernoulli mixtures from these rules separately. We then discard features in the corresponding mixtures, based upon their probability values. The results obtained on the SIGNET dataset show that this method is quite effective in rule extraction from raw data.

The outline of this paper is as follows: Section 2 introduces the reader to the notion of Bernoulli mixture models and how they can be used for boolean rules extraction. Section 3 describes the results of applying this method to the 'SIGNET' task. Finally, the paper concludes via Section 4.

2. Multivariate Bernoulli Mixtures

A probability mixture model represents an overall probability distribution of data via a convex combination of various component probability distributions also called mixtures. Each mixture is a probability distribution over a discrete or continuous variable and has its own set of parameters. A mixture model with D components is described by a probability function given by $p(\mathbf{x}) = \sum_{d=1}^{D} \pi_d p(\mathbf{x}|d)$. Here, π_d is the prior or the mixing proportion of each mixture so that $\sum_d \pi_d = 1$ and $p(\mathbf{x}|d)$ is its component-conditional probability function.

A multivariate Bernoulli mixture model assumes that each component of the model is an n-dimensional multivariate Bernoulli probability distribution. Suppose we have data given by $X = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m\}$. 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):

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

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., $p_{di} = p(x_{ki} = 1|d)$. We assume that the n-dimensional vector $\mathbf{x}$ has n independent

component attributes. The parameter θ governing this distribution 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 any appropriate optimization algorithm to estimate the parameters of these mixtures. For our work we used expectation maximization (EM) algorithm to estimate these parameter values from raw data.

2.1 Bernoulli Mixtures For Identifying Edges of a Hypercube and Extracting Boolean Rules from Data

In essence, one Bernoulli mixture is a vector of probabilities, each component representing the probability/chances of success of an individual feature or attribute. A single Bernoulli distribution over the entire data does not tell us anything about the inter-relationship of variables with each other. However, a mixture of such distributions can be used to determine the covariances and hence the correlations between pairs of attributes. They can also be used to identify high data density areas on the corners of a hypercube by thresholding the probability vector. We extract a vector $\mathbf{v}_d$ from a probability vector $\mathbf{p}_d$ and call it the main vector. The value of an attribute in a main vector can be taken as a 1 (0) if its probability is greater (less) than a certain threshold. The probability values around 0.5 can be taken as don't cares. So mathematically,

$$v_{di} = \begin{cases} 1 & \text{if } p_{di} > \alpha \\ 0 & \text{if } p_{di} < 1-\alpha \\ \phi & \text{otherwise} \end{cases} \tag{1}$$

where ϕ represents a don't care value. As an example consider Figure 1. Here, we have two mixtures extracted from three dimensional data and two corresponding main vectors. The figure shows the corners of the hypercube represented by these vectors and the corresponding boolean rule extracted from the 2 mixtures.

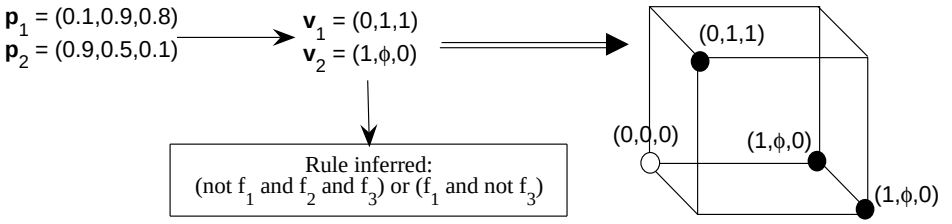

Figure 1: An example with two mixtures

Looking at Figure 1, we can see that it is easy to infer boolean rules from data once the corners of the hypercube are identified. Each corner represents a conjunct of boolean variables and together the set, of all the corners, forms a disjunction of rules to give us a disjunctive normal form of a boolean rule. The variables with a don't care value do not form a part of the rule.

2.2 Feature Selection in Rules

Once we have inferred the various disjuncts of a boolean rule we need to further identify the features that are irrelevant to our target variable. Suppose our training data is given by a set of m input vectors, $X = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m\}$, $\mathbf{x}_k \in \{0,1\}^n$. Each vector $\mathbf{x}_k$ is assigned a boolean label, $y_k \in \{0,1\}$, called the target variable. We partition the data into the two corresponding classes of 0 and 1 labels and generate D and D' Bernoulli mixture models from both these sets

separately. Let $M = \{\mathbf{p}_d; \pi_d\}_{d=1}^{D}$ be the mixtures extracted from instances with target variable equal to 1 and $M' = \{\mathbf{p}'_d; \pi'_{d'}\}_{d'=1}^{D'}$ be the mixtures estimated from data with 0 class labels.

Let $r = \{\mathbf{v}_d\}_{d=1}^{D}$ be the main vectors for mixtures in M and $r' = \{\mathbf{v}'_{d'}\}_{d'=1}^{D'}$ be the corresponding main vectors for mixtures in M'. A feature, which has don't care values in all vectors of r (r') is eliminated automatically for the rules for target variable = 1 (0). Also, if a feature's value does not change in both r and r' then it means that its value doesn't affect the target variable and hence this variable is irrelevant for its prediction. Hence, we can discard the literal/feature f from a rule if its value is the same in both the 0 and 1 labels. This variable is, therefore, assigned a don't care value, i.e.,

$$v_{df} = v'_{d'f} = \phi \quad \text{if } v_{df} = v_{d'f} \quad \forall d, \forall d'$$

Figure 2 illustrates the generation of rules for 0 and 1 class labels. Here we can see that the value of feature 2 does not vary in the main vector for both the classes and hence we can assign it a don't care value and eliminate it from the rules.

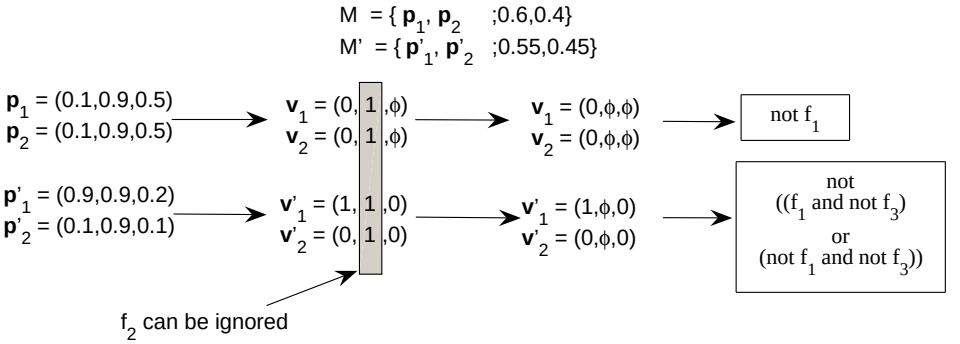

Figure 2: An example with two classes. The second rule is a negation as it is derived from 0 class labels.

The above mentioned procedure gives us rules for both 0 and 1 class labels for a target variable. Ideally, one rule should be the negation of the other. However, it is not the case as we treat the data with 0 and 1 labels separately. Out of the two rules, we choose the rule, which gives us better accuracy over the training set. A point to note here is that, our current model does not make any use of mixture priors. The mixture priors in a way represent the proportion of data being generated by a particular distribution. So even if part of the data (say a larger percentage, e.g., 80%) was generated by one mixture and a part of it (say a smaller percentage, e.g., 20%) by another mixture, we should still take both mixtures into account in the DNF of the rule for that feature as both the mixtures account for the generation of data and constitute the individual clauses in the rule.

Now we briefly outline the main steps of our algorithm:

1. For each variable i in the dataset repeat the following:
   i. Build the training data by using variable i as the target/class and the values of the rest of variables as input data.
   ii. Partition the data into 0 and 1 class labels
   iii. Generate mixtures M' and M for the corresponding 0 and 1 class labels
   iv. Generate the set of main vectors r' and r from both M' and M

v. Discard features, which have the same values in the main vectors for both class labels

vi. Generate two boolean rules from 0 and 1 class labels

vii. Check the training accuracy for both classes and output the rule with maximum training accuracy

3. Simulations

The method described in this paper was applied to the causality workbench's 'SIGNET' data describing the interaction of variables inside a plant signaling network (Li et al., 2006). The dataset includes 300 pseudo dynamic simulations of 43 boolean rules. Each simulation starts with a randomly generated boolean vector representing the initial state and is depicted by a 21x43 sized matrix. The next 20 vectors in a simulation are output by the boolean pseudo dynamic simulation using an asynchronous update scheme. Our task is to extract 43 boolean rules from the simulation data.

We applied the algorithm of Section 2.2 to extract boolean rules from the SIGNET data. Since all simulations result in stable values after a number of time steps, the dataset includes many repetitive training instances. To alleviate this problem, we removed duplicate entries. We emphasize that we are not using only stable states to generate the mixtures. All states present in the training data are used. Our data preparation only avoids biasing the data distribution in favor of stable states. For evaluation of the extracted rule, the organizers, of the challenge, suggested generating the truth tables for each true rule and computing a prediction error rate by comparing the output values of the extracted rule to the output value of the actual rule. The overall error rate is thus calculated by averaging over all the rules. The corresponding Matlab code for evaluating the system was also provided to us. In addition to calculating the overall error rate of prediction, we also calculated the training set error of the inferred rules. This was done by applying each rule to the individual boolean vectors of the simulation data and predicting the output value. The output value was compared with the actual value of simulation to get the overall training accuracy rate.

The main parameter of our model is the number of mixtures for positive and negative classes and the threshold α of Eq. (1). If the number of data points in a particular set were less than 100 then we generated only one mixture for this data and if the total data points were less than 350 then we generated only 2 mixtures from this set. The table and graph of Figure 3 show the training accuracy and overall accuracy of our method when generating different number of mixtures and using a threshold value of 0.8 on the original provided data. The results obtained after varying the threshold values are shown in Table 1 and discussed later.

When generating the rules we noticed that some rules cannot be inferred from data alone. For example, consider the following true rules:

MALATE = PEPC and not ABA and not AnionEM

ABA = 1

Since ABA is 1, hence, MALATE = 0, no matter what the values of the other variables are. However, when evaluating this rule, the system will generate the truth tables for three variables for MALATE and match the output of this rule with ours, resulting in very low accuracy. Hence, we replaced the constants ABA and AGB with 1 in the actual rules. In light of this, the organizers of the challenge generated a new dataset using our new rules. We repeated our experiments on the new dataset and the results obtained are illustrated in Figure 4.

Figure 3 shows that the best result on the original data was obtained by using only one mixture. This shows that generating different mixtures is not exactly necessary for inferring a rule based systems. However, as pointed out earlier the data is not consistent with the original set of rules. When we repeated our experiments with the new data generated according to the new set of rules we get the accuracies shown in Figure 4. Here, we can see that using 3 mixtures

Mixtures	% Training Accuracy	% Evaluation Accuracy
1	95.19	87.80
2	93.89	79.49
3	94.55	82.98
4	91.23	81.03
5	92.72	76.86
6	93.09	75.23
7	93.12	76.29
8	93.23	75.19
10	91.94	76.25

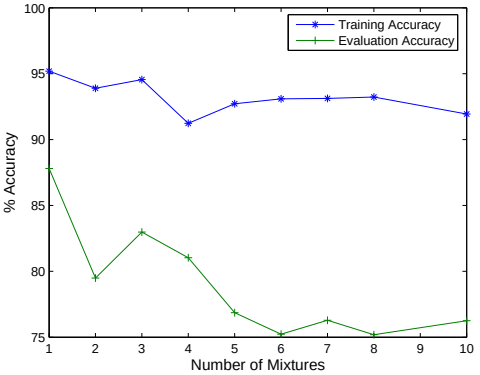

Figure 3: Results with different mixtures using original data (threshold = 0.8)

Mixtures	% Training Accuracy	% Evaluation Accuracy
1	95.41	86.97
2	95.60	85.61
3	95.88	87.61
4	95.37	81.72
5	95.44	83.68
6	94.65	81.04
7	94.63	82.32
8	94.83	79.2
10	94.42	80.87

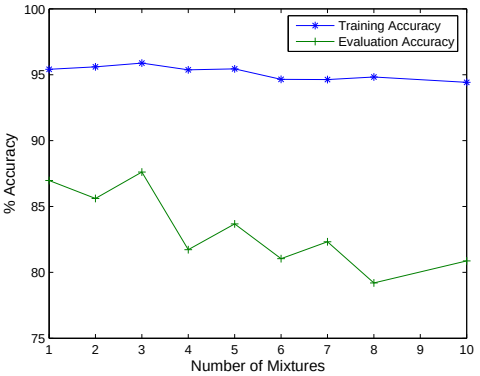

Figure 4: Results with different mixtures using new data (threshold = 0.8)

gives us the best results. As we increase the number of mixtures to 10 the accuracy deteriorates considerably owing to the generation of too many clauses for a rule.

To see the effect of threshold α on the quality of the generated rules, we repeated our experiments for different values of α using 1, 2 and 3 mixtures. The results are illustrated in Table 1. Here, we can see that the best overall results are obtained for a threshold value of 0.8. The results for a threshold of 0.7 are good for only a single mixture, but the accuracy for 2 and 3 mixtures is low as compared to the corresponding threshold of 0.8. When α is increased to 0.9 the accuracies deteriorate considerably. This is owing to the fact that too many values are being converted to don't care values for high threshold. In this case it is not possible to capture the actual rules governing the data.

Our method of converting a probabilistic model into a rule based system is quite intuitive and its usefulness is confirmed from experimental results on the SIGNET dataset. We can see that with a threshold value of 0.8 we get a good rule base with three mixtures, showing that most of the attributes can be modeled with rules that have three conjuncts inside them. Also, we get a good result with a single mixture at a threshold of 0.7.

All the results, demonstrated in this section, are, so far, based upon the algorithm described in Section 2.2. According to Step vi and vii of the algorithm, we generate two boolean rules for both the class 0 and class 1 labels. Out of these two rules we select the rule, which has

	threshold 0.7		threshold 0.8		threshold 0.9	
Mix-tures	**% Train Acc**	**% Evaluation Acc**	**% Train Acc**	**% Evaluation Acc**	**% Train Acc**	**% Evaluation Acc**
1	95.84	87.98	95.41	86.97	93.68	76.04
2	95.31	81.47	95.60	85.61	93.68	77.32
3	95.36	81.66	95.88	87.61	94.11	75.82

Table 1: Results with different threshold values on the new dataset

a higher training set accuracy. This may not be the best strategy as the training set data might overestimate the accuracy of a rule. In order to explore this further, we generated new data using the code provided by the organizers of the challenge. We varied the number of simulations in each experiment. For each simulation, we performed rule selection based upon training set accuracy, and then repeated the experiment by performing rule selection based upon validation set accuracy. For the later experiment, we set aside 20% simulations to constitute the validation set and generated the rules using the rest of 80% data. The results of the two methods, as a function of the number of simulations, are shown in Figure 5. Interestingly, for a smaller number of simulations, the method of selection based upon the validation set accuracy outperforms that of selection based upon the training set accuracy. As we increase the number of simulations, the method of selecting a rule based upon training set accuracy gives better results. This is because for smaller datasets, the rule is overfitting the data and hence gives poor performance. In general, we can see that the accuracy of the rule based system does not vary much with the increase in the number of simulations. It is drastically low for 20 simulations (74% using validation set) and then reaches the highest for 400 simulations (85% using validation set). The average accuracy (accuracy averaged over all the simulations) of rules selected via the training set and the validation set is the same, i.e., around 81%. Hence the validation set does not give us any added advantage over selection with the training set. The important thing to note here is that we can use rule selection based upon training set accuracy for large dataset sizes.

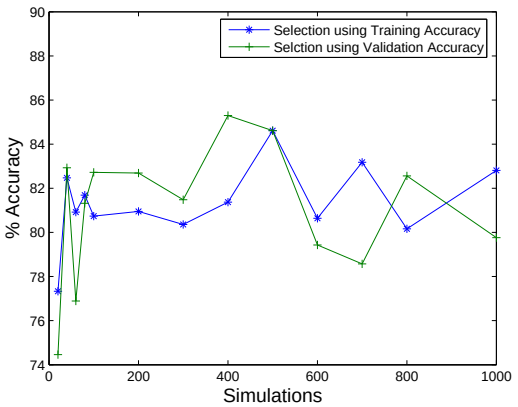

Figure 5: Results with different simulations using 3 mixtures (threshold = 0.8)

4. Discussion and Conclusions

Bernoulli mixtures provide a simple and intuitive method for inferring rules from raw data. The method, described in this paper, is simple and easy to implement. Our approach is novel as it shows how we can convert a probabilistic model into a rule based system. The concept applied here is different from the one adopted by Ideker et al. (2000). They are using only the stable values to infer boolean networks. Here, we are also making use of unstable values to deduce boolean rules. We use mixture models for feature selection and rule deduction. Ideker *et al.* , on the other hand, have used the concept of minimum set covering using greedy searches to determine various perturbations of genetic networks and entropy based measures to select the best network out of these networks.

Zheng and Geng (2008) were the other participants of the causality challenge who presented a solution for the SIGNET task. They have defined a methodology for identifying asynchronous networks. Their method is based upon the concept of finding the minimum explanatory set for a node, similar to Ideker et al. (2000)'s idea of finding minimum set covering. The log likelihood for a particular node is then maximized to determine its corresponding boolean function. Their method depends upon the number of assumed parent nodes and its complexity increases as the number of assumed parent nodes is increased. They achieved an impressive average accuracy of above 99% for a single parent node. However, this method might be too expensive to implement for networks where the in-degree of a node is more than one.

Currently we are converting a probabilistic model into a rule based system. The probabilistic model based on Bernoulli mixtures is a continuous model which is discretized to infer the boolean rules governing data. The priors of mixtures are completely ignored and not used. Another possibility could be to retain the continuous properties of our model and make predictions based on this model. The predictions can then be thresholded to attain truth tables and consequently the DNF of the individual rules.

x_1	x_2	x_1 **XOR** x_2
0	0	0
0	1	1
1	0	1
1	1	0

$\longrightarrow$

Rule from class label 0:
not((not x_1 and not x_2) or (x_1 and x_2))

Rule from class label 1:
((x_1 and not x_2) or (not x_1 and x_2))

Figure 6: The XOR problem and rule extraction

A critical parameter of our model is the actual number of mixtures to generate from data. If we can correctly identify this parameter, then it will increase our chances of inferring rules, from data, that have a higher degree of accuracy. As an example consider the truth table for the XOR problem in Figure 6. The corresponding rules are also shown in the figure. If we know in advance that our data can be modeled by two mixtures for each class then we can come up with accurate rules to represent the data. The rules extracted from both the zero and one class labels are also logically equivalent. However, for this problem one mixture will not be sufficient to infer the rules accurately. We need to come up with a good model selection technique that determines the actual number of mixtures to generate from data. There are several possibilities like the minimum message length (MML) criterion as suggested by Figueiredo and Jain (2002) is based upon concepts from information/coding theory. Minimum description length (MDL) and MML formally coincide with Bayesian inference criterion (BIC) (Schwarz, 1978), which is another possible approach to model selection.

In this paper, we have illustrated how a probabilistic model based upon Bernoulli mixtures can be converted into a rule based system. Two critical parameters of our method are the num-

ber of mixtures used to generate the rules and probability threshold value used to determine don't care values in rules. We evaluated the performance of our method by varying both these parameters and achieved good results when generating 3 mixtures with a threshold value of 0.8. We also evaluated the behavior of our method by varying the number of simulations or experiments to generate the raw data from which mixtures are learnt. We found that for very small number of simulations the performance of our rule based system is not very good. However, as we increase the number of simulations, the performance of the system also improves. Here, we have presented some preliminary work and showed how the basic concept of converting a mixture model into a rule based system can be used to give good performance. The method presented here is straightforward and intuitive and can be easily implemented.

Acknowledgments

I would like to thank Isabelle Guyon for her technical support, guidance and prompt replies to all my queries. I also thank Professor Haroon Babri and Mr. Kashif Javed of UET Lahore, for the useful discussions we had on this topic. Another thanks goes to the editors and the anonymous reviewers for their useful comments and feedback which helped me improve the quality of this paper

References

Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

Richard O. Duda and Peter E. Hart. *Pattern Classification and Scene Analysis*. Wiley, 1973.

Mario A.T. Figueiredo and Anil K. Jain. Unsupervised learning of finite mixture models. *IEEE transactions on pattern analysis and machine intelligence*, 24(3):381–396, March 2002.

Nir Friedman, Michal Linial, Iftach Nachman, and Dana Pe'er. Using bayesian networks to analyze expression data. In *Proceedings of the Fourth Annual International Conference on Computational Molecular Biology*, 2000.

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.

Trey E. Ideker, Vesteinn Thorsson, and Richard M. Karp. Discovery of regulatory interactions through perturbation: Inference and experimental design. *Pacific Symposium on Biocomputing*, 5:302–313, 2000.

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 (ICPR'04)*, 2004.

Song Li, Sarah M. Assmann1, and Re ka Albert. Predicting essential components of signal transduction networks: A dynamic model of guard cell abscisic acid signaling. *PLoS Biology*, 4(10):1732–1748, 2006.

Shoudan Liang, Stefanie Fuhrman, and Roland Somogyi. Reveal, a general reverse engineering algorithm for inference of genetic network architectures. *Pacific Symposium on Biocomputing*, 3:18–29, 1998.

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.

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.

Gideon Schwarz. Estimating the dimension of a model. *The Annals of Statistics*, 6(2):461–464, March 1978.

D.C. Weaver, Christopher T. Workman, and Gary D. Stormo. Modeling regulatory networks with weight matrices. *Pacific Symposium on Biocomputing*, 4:112–123, 1999.

Cheng Zheng and Zhi Geng. Reverse engineering of asynchronous boolean networks via minimum explanatory set and maximum likelihood. In *NIPS 2008 Workshop on Causality*, 2008. see http://clopinet.com/isabelle/Projects/NIPS2008/.

Appendix A. Pot-luck causality challenge: FACT SHEET (SIGNET)

Title: The Use of Bernoulli Mixture Models for Identifying Corners of a Hypercube and Extracting Boolean Rules From Data

Participant name, address, email and website: Mehreen Saeed, FAST National University of Computer and Emerging Sciences, Lahore Campus, Pakistan.

email: mehreen.saeed@nu.edu.pk

Task(s) solved: SIGNET

Reference: Not yet published

Method:

Profile of the method:

- **Preprocessing :** None
- **Causal discovery:** None
- **Feature selection:** Bernoulli mixtures were generated from data and the probabilities in the mixtures were thresholded to identify corners of the hypercube that represent the data. From there, only those features were selected which had varying values in class 1 and class 2.
- **Classification :** The task did not involve classification, only rule generation.

Mixtures	% Training Accuracy	% Evaluation Accuracy
1	95.41	86.97
2	95.60	85.61
3	95.88	87.61
4	95.37	81.72
5	95.44	83.68
6	94.65	81.04
7	94.63	82.32
8	94.83	79.2
10	94.42	80.87

Table 2: Results with different mixtures and threshold of 0.8)

- **Model selection/hyperparameter selection:** None

Results: Shown in Table 2

- **quantitative advantages:** Simplicity, computationally easy, intuitive
- **qualitative advantages:** Converts a probabilistic model into a rule based model which is novel.

Implementation: Code for Bernoulli mixtures was written in C++. The code for boolean rule generation was implemented in Matlab

Keywords:

- **Preprocessing or feature construction:** None
- **Causal discovery:** None
- **Feature selection:** Bernoulli mixtures
- **Classifier:** None
- **Hyper-parameter selection:** None
- **Other:** Boolean rule generation using Bernoulli mixtures