Approximating High Dimensional Probability Distributions

Stephan Altmueller and Robert M. Haralick
Department of Computer Science
Graduate Center, City University of New York
365 Fifth Avenue,
New York, NY, 10034
{saltmueller@,haralick@ptah.}gc.cuny.edu

Abstract

We present an approach to estimating high dimensional discrete probability distributions with decomposable graphical models. Starting with the independence assumption we add edges and thus gradually increase the complexity of our model. Bounded by the Minimum Description Length principle we are able to produce highly accurate models without overfitting. We discuss the properties and benefits of this approach in an experimental evaluation and compare it to the well studied Chow-Liu algorithm.

1 Introduction

In this paper, we discuss the problem of estimating high dimensional discrete probability distributions. A problem that arises in a wide variety of different research areas such as computer vision, text mining or information retrieval. These applications require to estimate high dimensional probability distributions from training data. It is generally not possible to estimate the joint distribution directly (even with a large training set), since the number of distinct observations grows exponentially with the number of dimensions. This problem can be solved by exploiting (conditional) independence relations within the population of interest. In recent years graphical models have been an active area of research. They combine graph and probability theory and allow to model multivariate domains. A graphical model consists of a graph and a list of probability tables. The graph encodes the (conditional) independences between the random variables while the probability tables store marginal probability distributions that are necessary to express the joint distribution (see section 2 for details). In this paper we discuss undirected discrete graphical models. To learn a graphical model in general the graph and the probabilities have to be learned from data. To learn the optimal structure of the graph is computationally intractable in general due to the high number of possible graphs. Thus greedy methods are widely used. One tractable subclass of graphical models are *dependence trees*, for which Chow and Liu [3] developed an optimal algorithm. The *Chow-Liu* algorithm has been extended in numerous ways (see for example [11], [5]). The remainder of the paper is organized as follows. In section 2 we discuss decomposable graphical models and describe their relationship to the *Chow-Liu* algorithm. In section 3 we describe the greedy forward selection procedure that we used in our experiments and a stopping criterion based on the Minimum Description Length principle. Section 4 describes experimental results and section 5 concludes with future research directions.

2 Decomposable Graphical Models

First, we give the notation that is used throughout the paper. $\mathbf{X} = \{X_1, \ldots, X_N\}$ denotes the vector of discrete random variables. $P(\mathbf{X})$ denotes the joint probability distribution of the variables in $\mathbf{X}$. $\mathbf{X}_A$, where $A \subseteq \{1, \ldots, N\}$, is a subset of $\mathbf{X}$ and $P(\mathbf{X}_A)$ denotes the marginal probability distribution of that subset. A graphical model $\mathcal{M} = (\mathcal{G}, \mathcal{L})$ consists of an undirected graph $\mathcal{G}$ and a list of marginal probability distributions $\mathcal{L}$. The graph $\mathcal{G} = (\mathbf{V}, \mathbf{E})$ is composed of a set of vertices $\mathbf{V}$ and an edge set $\mathbf{E}$. The vertices of the graph correspond to the random variables $\mathbf{X}$, thus we will use them interchangeably throughout the paper. Generally, we are given a dataset $\mathcal{D} = \langle d_1, \ldots, d_M \rangle$ drawn from an unknown population, for which we wish to estimate the joint probability distribution. The graph $\mathcal{G}$ encodes the dependences between the variables. The *Markov* properties form the theoretical foundation for this encoding (see [8] for a detailed discussion). Informally, they can be described as follows: If there is an edge between two vertices the two variables are dependent; if a there is path between two variables then the two variables are conditionally independent given the other variables. If there is no path between two variable they are independent. In this paper we limit our-

Proceedings of the 17th International Conference on Pattern Recognition (ICPR'04)
1051-4651/04 $ 20.00 IEEE

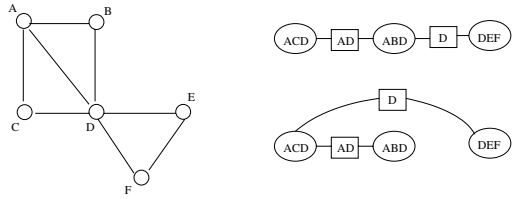

Figure 1. Dependence graphs and their junction trees.

selves to the subclass of decomposable graphical models [8]. A graphical model is decomposable if its graph is *triangulated*:

Definition 1 *A graph G = (V, E) is called triangulated iff all cycles of* $length \geq 4$ *have a chord. A chord is an edge in the graph that connects two non-consecutive edges of the cycle.*

If a graph is triangulated it permits the derivation of a secondary graphical structure [8]:

Definition 2 *Let* $\mathcal{C}$ *be the set of maximal cliques of the graph* $\mathcal{G}$ *and* $\mathcal{T}$ *be a tree with the elements of* $\mathcal{C}$ *as its vertices. Then* $\mathcal{T}$ *is called a junction tree iff the intersection* $C_1 \cap C_2$ *is contained in all nodes on the path between* C_1 *and* C_2*, for all possible pairs* $\{(C_1, C_2)|C_1, C_2 \in \mathcal{C}\}$.

Each edge (C_1, C_2) in the junction tree that connects the two cliques C_1 and C_2 has the intersection $S_{12} = C_1 \cap C_2$ associated with it. S_{12} is called a *separator*. The junction tree of a graphical model serves a number of important purposes. It provides the terms necessary to express the joint probability distribution as a function of lower order marginals, it is used to perform exact inference and it permits efficient sampling [6]. The list of separators of a junction tree are always unique, however the graphical structure of the tree is not. Figure 1 (taken from [8]) shows a graph and the two possible junction trees that can be derived from the graph.
The (conditional) independence assumptions that are encoded in the model graph $\mathcal{G}$ allow to express the joint probability as a product of lower order marginals:

$$P_{ap}(\mathbf{X}) = \frac{\prod_{c \in \mathcal{C}} P_c(\mathbf{X}_c)}{\prod_{s \in \mathcal{S}} P_s(\mathbf{X}_s)} \tag{1}$$

where $\mathcal{C}$ is list of cliques of the graph and $\mathcal{S}$ is the list of separators contained in the junction tree. The closed form expression for the joint probability only exists if the graph is decomposable (and therefore a junction tree exists). In the case of a non-decomposable graph a computationally expensive procedure called *iterative proportional fitting* has to be used to find such a factorization.

For a given dataset $\mathcal{D}$ we are generally interested in finding the model that includes all the (conditional) independences that govern the population from which $\mathcal{D}$ was drawn. Given a model we can estimate the marginal distributions (defined by the cliques and separators) from the dataset. The resulting joint distribution $P_{ap}(\mathbf{X})$ is an approximation of the joint distribution $P(\mathbf{X})$. In order to measure the goodness of the approximation the *Kullback-Leibler* (KL) divergence is used:

$$I(P, P_{ap}) = \sum_{x \in \mathbf{X}} P(x) \log(\frac{P(x)}{P_{ap}(x)}) \tag{2}$$

The KL divergence is not a metric (it does not satisfy the triangle inequality) and has the property that $I(P, P_{ap}) \geq 0$. It is only zero if the two probability distributions are equal. The *Chow-Liu* algorithm [3] is an optimal algorithm to find dependence trees, a subclass of graphical models where the graph $\mathcal{G}$ is a tree and therefore the joint distribution can be expressed as a product of second order marginals. It has been shown that the *Chow-Liu* algorithm guarantees to find the optimal tree with respect to the KL divergence. Dependence trees have been widely used since their introduction in the late 60's. See [3] and [11] for details and variations of the *Chow-Liu* algorithm. The complexity of the *Chow-Liu* algorithm is $O(N^2)$ where N is the number of variables.

3 Model Selection in High Dimensions

The objective of model selection is to find a model that minimizes the KL Divergence while maintaining a simple model; i.e. one with as few edges as possible. To find a optimal model one would have to search the entire model space. Since a brute force search is intractable, greedy algorithms are widely used. Deshpande *et al.* [4] give a theoretical description of an algorithm for efficient stepwise selection in decomposable models. The algorithm allows the efficient enumeration of all edges that can be added to a decomposable graph while maintaining decomposability. Based on this procedure an algorithm for efficient foward selection can be derived, which we will refer to as the *Efficient Forward Selection* (EFS) algorithm in the remaining part of the paper. For details on our implementation of the algorithm see [1]. Forward selection starts with the independence assumption (a graph with no edges) and repeatedly adds edges until the model fits the data according to a stopping criterion (see below). Malvestuto [10] shows that the KL divergence can be rewritten as:

$$I(P, P_{ap}) = -\mathcal{H}(P) + \mathcal{H}(\mathcal{M})$$

where $\mathcal{H}(P) = \sum_{x \in \mathbf{X}} - \log_2 P(x)$ is the entropy of P and $H(\mathcal{M})$ is the *model entropy* which is defined as:

$$\mathcal{H}(\mathcal{M}) = \sum_{c \in \mathcal{C}} \mathcal{H}(P_c) - \sum_{s \in \mathcal{S}} \mathcal{H}(P_s)$$

Like in (1) $\mathcal{C}$ denotes list of cliques of the graph and $\mathcal{S}$ is the list of separators. Thus, to minimize the KL divergence in our model search only the model entropy $H(\mathcal{M})$ needs to be minimized. The algorithm in [4] shows that the decrease in model entropy can be calculated individually for each edge that is eligible for addition. At each step in the algorithm we have a list of edges that can be added to the decomposable model and for each edge we know the quantity by which the model entropy will decrease if that edge was added. Our implementation of the EFS algorithms follows a greedy strategy by always choosing the edge, which will decreases the model entropy the most. The complexity of the EFS algorithm is $O(kN^2)$ where k is the total number of edges added by search procedure. To avoid overfitting we use a score based on the *minimum description length* (MDL) principle [7] as our stopping criterion. The description length associated with a model estimates the expected number of bits necessary to optimally encode the dataset given the model. The model itself also has to be encoded since the data could not be decoded without it. The MDL balances between model fit and model complexity. The description length is composed of the number of bits necessary to encode the model graph $\mathcal{G}$, the marginal probability distributions $\mathcal{L}$ associated with the cliques of the graph and the observed dataset $\mathcal{D}$. Since we are not interested in creating the actual code but merely the code length we can omit terms that are constant for all candidate models. To encode the graph it is sufficient to encode the cliques of the graph:

$$DL_{Graph} = \sum_{i=1}^{|C|} \log(N) + k_i \log(N)$$

where $\mid C \mid$ is the number of cliques, k_i is the size of the i^{th} clique. [6] shows that (1) can be rewritten in terms of conditional probabilities

$$\frac{\prod_{c \in \mathcal{C}} P_c(\mathbf{X}_c)}{\prod_{s \in \mathcal{S}} P_s(\mathbf{X}_s)} = P(X_{C_0}) \prod_{i=1}^{m} P(X_{C_i - S_i} | X_{S_i})$$

Thus, the description length of the probability tables is:

$$DL_{Prob} = d[(\|X_{C_0}\| - 1) + \sum_{i=1}^{C} \|X_{S_i}\|(\|X_{C_i - S_i}\| - 1)]$$

where d is the number of bits necessary to encode a single parameter and $\|X_A\|$ is the size of the subspace defined by the subset of features $\mathbf{X}_A$. To estimate the description length of the data we can use the model entropy:

$$DL_{Data} = M \cdot H(\mathcal{M})$$

The total description length is then the sum of DL_{Graph}, DL_{Prob} and DL_{Data}. We keep adding edges to our candidate model as long as the description length decreases. This definition of the description length was inspired by [7] and [6].

4 Experimental Evaluation

4.1 Synthetic Data

In the first round of experiments we sample a dataset from a known graphical model. The model is created at random with 60 vertices, 167 edges, an average clique size of 3.73 and a maximum clique size of 8. Figure 2 shows the percentage of edges that were present in the original graph but not found by the EFS algorithm. The error decreases as the size of the training sample increases. For each size of the training set we ran the learning procedure 10 times and averaged over the error.

4.2 Classifying Handwritten Digits

In another round of experiments we used the EFS algorithm to recognize handwritten digits from the MNIST database [9]. This database consists of 20×20 grayscale images. The database contains a training set with 60000 images and a test set with 10000 images. Since we are primarily interested in estimating high dimensional probability distributions we did minimal preprocessing: Binary images were created by thresholding using a global threshold and we derived two binary datasets: a $400-$dimensional dataset of the original images and a $256-$dimensional dataset of the images scaled to 16×16. For both datasets we estimated the class conditional probabilities using the EFS algorithm. During classification we picked the class that had the highest probability given the test image. We repeated the same procedure for the *Chow-Liu* algorithm. The lowest error rates were obtained when the threshold was high. Therefore we averaged over thresholds ranging from 220 to 254. The following table shows the average and best error rates for the different learning procedures and datasets:

Algorithm/Size	Best	Average
EFS/400	0.081	0.085
Chow Liu/400	0.278	0.289
EFS/256	0.079	0.082
Chow Liu/256	0.178	0.210

The error rate is somewhat larger than the ones reported in [2] and [9]. We attribute this to the fact that we did no preprocessing or feature extraction. Bach and Jordan show in [2] the insensitivity of graphical models to missing pixels. For future research we plan to investigate the effect of noise (randomly added pixels) on the performance of this approach.

4.3 Splice Dataset

The *Splice* Dataset [11] contains a dataset from the area of molecular biology. It contains a total of 64 ternary features and each observation belongs to one of three classes.

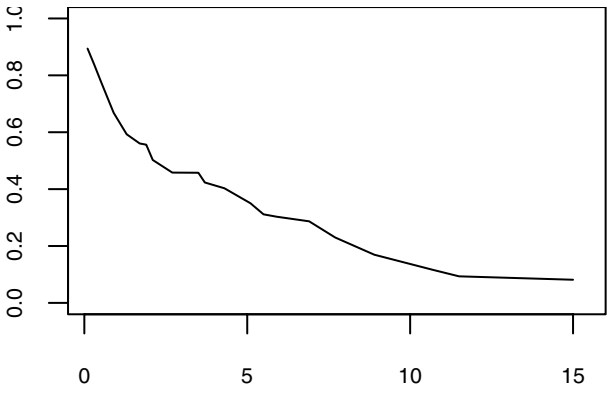

Figure 2. Percentage of unrecognized edges over training set size (in thousands)

As described in [11] only a small number of features are necessary for classification. To find these features we converted the class label into an additional variable and used this augmented dataset to learn a graphical model and a *Chow-Liu* tree. According to the Markov properties only the variables connected to the "class variable" are significant for classification. We extracted those features and the reduced dataset was classified. Figure 3 shows the error rate for classification with a graphical model, a *Chow-Liu* tree, after feature extraction using the graphical model and after feature extraction using the *Chow-Liu* algorithm. While the EFS and *Chow-Liu* algorithms perform almost equally well in the full space, there is a clear advantage of the graphical model over the *Chow-Liu* tree when it comes to automated feature selection. This is due to the fact that Chow-Liu algorithm is limited to finding trees and is therefore not able to detect all dependences simultanously.

5 Conclusions and Future Work

This paper makes a number of contributions. To our knowledge it is the first report on a practical implementation of the EFS algorithm described theoretically in [4]. We showed by experimental evaluation how this approach can be used to estimate high dimensional probability distributions quite accurately and how it can be used to perform automatic feature selection. The proposed stopping criterion based on the MDL guarantees a good balance between model complexity and accuracy. A desirable trait it shares with the *Chow-Liu* algorithm. The presented approach suggests a large number of possible extensions. Our approach does not account for uncertainty about the model itself. This can be addressed by using mixtures of graphical models (similar to the mixtures of trees in [11]). We are also planing to investigate the impact of different choices for the *MDL*, the sensitivity to noise and the effectiveness of search strategies other than greedy search.

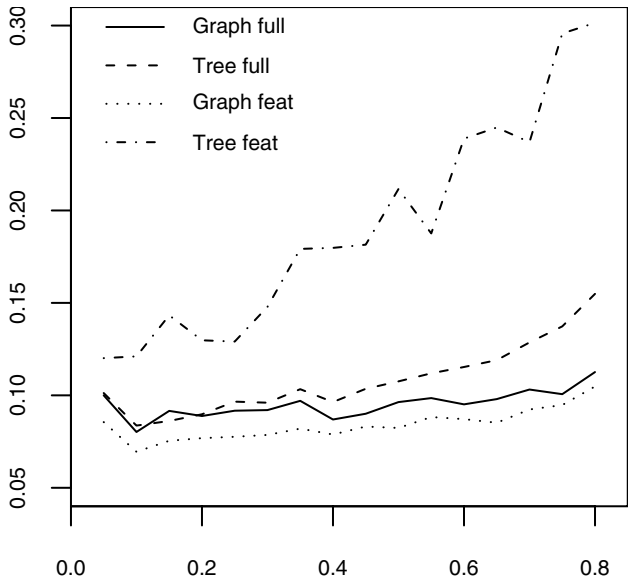

Figure 3. Error rate for the full space (Graph full, Tree full) and after feature extraction (Graph feat, Tree feat)

References

[1] S. Altmueller and R. Haralick. Forward selection in decomposable graphical models. Technical report, CUNY Graduate Center, 2004.

[2] F. R. Bach and M. I. Jordan. Thin junction trees. In *Advances in Neural Information Processing Systems 14*, pages 569–576, 2002.

[3] C. Chow and C. Liu. Approximating discrete probability distributions with dependence trees. *IEEE Trans. Information Theory*, 14:(11):462–467, November 1968.

[4] A. Deshpande, M. Garofalakis, and M. I. Jordan. Efficient stepwise selection in decomposable models. In *Uncertainty in Artificial Intelligence*, 2001.

[5] K. Huang, I. King, and M. Lyu. Constructing a large node chow-liu tree based on frequent itemsets. In *International Conference on Neural Information Processing*, 2002.

[6] M. I. Jordan, editor. *Learning in Graphical Models*. Kluwer Academic Publishers, 1998.

[7] W. Lam and F. Bacchus. Learning bayesian belief networks: An approach based on the mdl principle. *Computational Intelligence*, 10:269–293, 1994.

[8] S. Lauritzen. *Graphical Models*. Oxford University Press, 1996.

[9] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, November 1998.

[10] F. M. Malvestuto. Approximating discrete probability distributions with decomposable models. *IEEE Transactions on Systems, Man, and Cybernetics*, 21, NO. 5, 1991.

[11] M. Meila and M. I. Jordan. Learning with mixtures of trees. *Journal of Machine Learning Research*, 1:1–48, 2000.