ECAI 2010
H. Coelho et al. (Eds.)
IOS Press, 2010

doi:10.3233/978-1-60750-606-5-427

Kernel-Based Hybrid Random Fields for Nonparametric Density Estimation

Antonino Freno and **Edmondo Trentin** and **Marco Gori** [1]

Abstract. Hybrid random fields are a recently proposed graphical model for pseudo-likelihood estimation in discrete domains. In this paper, we develop a continuous version of the model for nonparametric density estimation. To this aim, Nadaraya-Watson kernel estimators are used to model the local conditional densities within hybrid random fields. First, we introduce a heuristic algorithm for tuning the kernel bandwidhts in the conditional density estimators. Second, we propose a novel method for initializing the structure learning algorithm originally employed for hybrid random fields, which was meant instead for discrete variables. In order to test the accuracy of the proposed technique, we use a number of synthetic pattern classification benchmarks, generated from random distributions featuring nonlinear correlations between the variables. As compared to state-of-the-art nonparametric and semiparametric learning techniques for probabilistic graphical models, kernel-based hybrid random fields regularly outperform each considered alternative in terms of recognition accuracy, while preserving the scalability properties (with respect to the number of variables) that originally motivated their introduction.

1 INTRODUCTION

In continuous domains, learning probabilistic graphical models from data is much more challenging than in discrete domains. While the multinomial distribution is a generally adequate choice for estimating conditional probabilities in discrete event spaces, choosing a suitable kind of estimator for (continuous) conditional density functions requires to make a decision as to whether to assume that the form of the modeled density is known (e.g. normal), which leads to parametric techniques, or to relax the parametric assumption, which leads to nonparametric techniques [4]. The parametric assumption is often limiting, because in real-world applications the true form of the density function is rarely known *a priori*. On the other hand, nonparametric techniques only make a much weaker assumption concerning the smoothness of the density function.

While a lot of research has been devoted to parametric graphical models in the machine learning community [19, 3], only a few efforts have been devoted to nonparametric (or semiparametric) models [11, 12, 1, 17, 16]. We introduce a nonparametric version of hybrid random fields (HRFs), which have been recently proposed for scalable pseudo-likelihood estimation in discrete domains [6, 7]. The model developed in this paper exploits kernel-based conditional density estimators. In Sec. 2 we review the basic concepts related to HRFs. Parameter and structure learning are addressed in Sec. 3 and Sec. 4 respectively, while Sec. 5 relates our work to some recent proposals. In Sec. 6, we provide an experimental evaluation of the prediction accuracy and computation time of kernel-based HRFs (KHRFs) on a number of pattern classification tasks, comparing our learning technique to other nonparametric, semiparametric, and parametric learning methods for graphical models. Finally, in Sec. 7 we summarize the main results of the presented work and we outline some directions for future research.

[1] Dipartimento di Ingegneria dell'Informazione, Università degli Studi di Siena (Siena, Italy). Email: {freno, trentin, marco}@dii.unisi.it.

2 HYBRID RANDOM FIELDS

HRFs are defined as follows [6, 7]:

Definition 1. *Let $\mathbf{X}$ be a set of random variables $X_1, \ldots, X_d$. A hybrid random field for $X_1, \ldots, X_d$ is a set of Bayesian networks $BN_1, \ldots, BN_d$ (with graphs $\mathcal{G}_1, \ldots, \mathcal{G}_d$) such that:*

i. Each BN_i contains X_i plus a subset $\mathcal{R}(X_i)$ of $\mathbf{X} \setminus \{X_i\}$;
ii. For each X_i, $p(X_i|\mathbf{X} \setminus \{X_i\}) = p(X_i|\mathcal{MB}_i(X_i))$, where $\mathcal{MB}_i(X_i)$ is the set containing the parents, the children, and the parents of the children of X_i in $\mathcal{G}_i$.

The set $\mathcal{MB}_i(X_i)$ is a Markov blanket (MB) of X_i within BN_i [21]. The elements of $\mathcal{R}(X_i)$ (i.e., all nodes appearing in graph $\mathcal{G}_i$ except X_i itself) are called 'relatives of X_i'. Condition ii in Definition 1 (the so-called 'modularity property') entails that $\mathcal{MB}_i(X_i)$ is a MB of X_i in $\mathbf{X}$.

HRFs provide a direct way of computing the pseudo-likelihood $p^*(\mathbf{x})$ of a given state $\mathbf{x}$ of the variables in $\mathbf{X}$ [2]:

$$p^*(\mathbf{X} = \mathbf{x}) = \prod_{i=1}^{d} p(X_i = x_i|\mathcal{MB}_i(X_i) = mb_i(X_i)) \quad (1)$$

where $mb_i(X_i)$ is the state of $\mathcal{MB}_i(X_i)$. Gibbs sampling techniques [9] need to be used when we want to extract a strict joint probability from a HRF.

It is known that the class of joint probability distributions representable by Bayesian networks (BNs) is strictly included in the class of pseudo-likelihood distributions representable by HRFs [7]. Since the theorems establishing this result do not rely on the assumption that the involved random variables are discrete, the result holds for continuous as well as for discrete event spaces.

3 PARAMETER LEARNING

In continuous HRFs, parameter learning consists in estimating the conditional density $p(X_i|\mathcal{MB}_i(X_i))$, for each node X_i belonging to the HRF. As for (unconditional) probability density function estimation, conditional density estimation can be addressed by either

parametric or nonparametric techniques. In order to make the model as widely applicable as possible (in particular to estimation tasks where no domain knowledge is available), we learn the parameters of continuous HRFs using a (kernel-based) nonparametric technique. Kernel-based conditional density estimation is addressed in Sec. 3.1, while in Sec. 3.2 we propose a data-driven bandwidth selection technique.

3.1 Kernel-based conditional density estimation

In order to estimate the conditional density $p(y|\mathbf{x})$, where y is the value of a random variable Y and $\mathbf{x}$ is the value of a random vector $\mathbf{X}$, we use the Nadaraya-Watson (NW) estimator [18, 25, 23]. Suppose we are given a dataset $\mathbf{D} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$. Then, the estimator takes the following form:

$$\hat{p}(y|\mathbf{x}) = \frac{\sum_{i=1}^{n} K_{h_1}(y - y_i) K_{h_2}(\mathbf{x} - \mathbf{x}_i)}{\sum_{i=1}^{n} K_{h_2}(\mathbf{x} - \mathbf{x}_i)} \tag{2}$$

In Eq. 2, each function K_h is defined as follows:

$$K_h(\mathbf{u}) = \frac{1}{h^d} K\left(\frac{\|\mathbf{u}\|}{h}\right) \tag{3}$$

where K is a kernel function, h is the bandwidth (or window width), i.e. a parameter determinining the width of the kernel function, and d is the dimensionality of $\mathbf{u}$. Our choice for K is the Epanechnikov kernel [5]:

$$K(x) = \frac{3}{4}(1 - x^2)\mathbf{1}_{\{|x| \leq 1\}} \tag{4}$$

where

$$\mathbf{1}_{\{|x| \leq 1\}} = \begin{cases} 1 & \text{if } |x| \leq 1 \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

We use the Epanechnikov kernel not only because it is known to be asymptotically optimal, but also because it offers a significant computational advantage (at least in the presence of large datasets) with respect to other optimal functions such as the Gaussian kernel [24].

3.2 Bandwidth selection

In order for the NW estimator to deliver accurate predictions, it is crucial to choose suitable values for the bandwidths h_1 and h_2. Our strategy for dealing with this task is based on the idea of finding the bandwidth values that maximize the cross-validated log-likelihood (CVLL) of the estimator with respect to dataset $\mathbf{D}$ [24, 13]. CVLL can be defined as follows:

$$CVLL(h_1, h_2) = \frac{1}{n} \sum_{i=1}^{n} \log(\hat{p}^{-i}(y_i|\mathbf{x}_i)\hat{p}^{-i}(\mathbf{x}_i)) \tag{6}$$

where

$$\hat{p}^{-i}(y_i|\mathbf{x}_i) = \frac{\sum_{j \neq i} K_{h_1}(y_i - y_j) K_{h_2}(\mathbf{x}_i - \mathbf{x}_j)}{\sum_{j \neq i} K_{h_2}(\mathbf{x}_i - \mathbf{x}_j)} \tag{7}$$

and

$$\hat{p}^{-i}(\mathbf{x}_i) = \frac{1}{n-1} \sum_{j \neq i} K_{h_2}(\mathbf{x}_i - \mathbf{x}_j) \tag{8}$$

Simplifying Eq. 6, we get:

$$CVLL(h_1, h_2) = \\ = \left(\frac{1}{n} \sum_{i=1}^{n} \log \sum_{j \neq i} K_{h_1}(y_i - y_j) K_{h_2}(\mathbf{x}_i - \mathbf{x}_j)\right) - \log(n-1) \tag{9}$$

The algorithm that we develop in order to maximize $CVLL(h_1, h_2)$ performs a double dichotomic search in a space of possible bandwidth pairs. Two ranges of values $(0, h_{max_1})$ and $(0, h_{max_2})$ are simultaneously explored by evaluating subregions of the intervals (according to the CVLL metric) and then narrowing down the search to smaller intervals in an iterative way. An iteration of the algorithm begins by splitting each interval $(0, h_{max_i})$ in two (equally large) regions $\mathcal{H}_{i_1}$ and $\mathcal{H}_{i_2}$. Then, each pair $(\mathcal{H}_{1_i}, \mathcal{H}_{2_j})$ such that $i, j \in \{1, 2\}$ is evaluated by choosing the median of each region as the value of the corresponding bandwidth. Finally, the pair of regions that maximizes the CVLL is selected as pair of (narrower) intervals for the following iteration. The algorithm returns the highest-scoring pair (h_1, h_2) found during the search. Pseudocode for the described technique is provided by Algorithm 1.

Algorithm 1 Bandwidth selection by double dichotomic search

Input: Limit points h_{max_1}, h_{max_2}; number s of iterations; dataset $\mathbf{D}$.

Output: Bandwidth pair (h_1, h_2).

```
1.  maxScore = −∞
2.  for(i = 1 to 2)
3.    min_i = 0
4.    max_i = h_{max_i}
5.    median_i = (1/2) max_i
6.  for(i = 1 to s)
7.    for(j = 1 to 2)
8.      ε_j = (1/4)(max_j − min_j)
9.      h_{j_1} = median_j − ε_j
10.     h_{j_2} = median_j + ε_j
11.   (k, k') = arg max_{k,k' ∈ {1,2}} CVLL(h_{1_k}, h_{2_{k'}})
12.   median_1 = h_{1_k}
13.   median_2 = h_{2_{k'}}
14.   for(j = 1 to 2)
15.     min_j = median_j − ε_j
16.     max_j = median_j + ε_j
17.   if(CVLL(h_{1_k}, h_{2_{k'}}) > maxScore)
18.     maxScore = CVLL(h_{1_k}, h_{2_{k'}})
19.     h_1 = h_{1_k}
20.     h_2 = h_{2_{k'}}
21. return (h_1, h_2)
```

Clearly, the complexity of computing the CVLL function is quadratic in the number of data points. This can be a serious limitation when dealing with very large datasets. However, a promising way of overcoming this issue is proposed in [13] based on dual-tree recursion [10]. Although the idea goes beyond the scope of the present work, we notice that incorporating the dual-tree fast approximation to the CVLL metric into the framework of kernel-based HRFs would be straightforward.

4 STRUCTURE LEARNING

Structure learning in HRFs consists of learning, for each variable X_i, which variables belong to $\mathcal{MB}_i(X_i)$. This reduces to learning what other variables appear in BN_i, and in particular what edges are contained in $\mathcal{G}_i$. Markov Blanket Merging (MBM) is the first structure learning algorithm that has been proposed thus far for HRFs [6, 7]. As compared to state-of-the-art learning algorithms for BNs and Markov random fields (MRFs), the main selling point of MBM is

a dramatic reduction of the computational cost of structure learning. MBM tries to maximize the model pseudo-likelihood given a dataset $\mathbf{D}$. To this aim, it starts from an initial assignment of relatives to the model variables, then it learns the local BNs and it iteratively refines the assignment in order to come up with MBs that increase the pseudo-likelihood with respect to previous assignments. The algorithm stops when no further refinement of the MBs increases the model pseudo-likelihood. Although the original version of MBM is designed for learning discrete HRFs, it can be used with little modification to learn continuous HRFs. Secs. 4.1–4.3 describe the way we modify MBM in order to adapt it to KHRFs.

4.1 Model initialization

One part of the algorithm that needs to be modified in a suitable way is the model initialization technique. In discrete HRFs, MBM produces an initial assignment by choosing an initial size k of the set of relatives, and then by selecting as relatives of each X_i the k variables that display the highest statistical dependence with respect to X_i, where the strength of the correlation is measured by the value of the χ^2 statistic. Since the χ^2 statistic naturally applies to discrete variables only, what we need is a way of measuring correlation for pairs of continuous variables in a direct way (i.e. without having to discretize the variables before applying the test).

Our choice is to measure the statistical correlation for any pair of continuous variables by the value of the correlation ratio [14] for that pair. Consider two random variables X_i and X_j that have been observed n times within a dataset $\mathbf{D}$. Moreover, define $\hat{\mu_i}$, $\hat{\mu_j}$, and $\hat{\mu}$ in the following way:

$$\hat{\mu_i} = \frac{1}{n}\sum_{k=1}^{n} x_{i_k} \tag{10}$$

$$\hat{\mu_j} = \frac{1}{n}\sum_{k=1}^{n} x_{j_k} \tag{11}$$

$$\hat{\mu} = \frac{1}{2}(\hat{\mu_i} + \hat{\mu_j}) \tag{12}$$

where x_{i_k} and x_{j_k} denote the values of the k-th observation of x_i and x_j in $\mathbf{D}$. Then, the correlation ratio statistic η for the pair (X_i, X_j) can be computed as follows:

$$\eta(X_i, X_j) = \sqrt{\frac{n(\hat{\mu_i} - \hat{\mu})^2 + n(\hat{\mu_j} - \hat{\mu})^2}{\sum_{k=1}^{n}(x_{i_k} - \hat{\mu})^2 + (x_{j_k} - \hat{\mu})^2}} \tag{13}$$

The correlation ratio is such that $0 \leq \eta \leq 1$, where lower values correspond to stronger degrees of correlation, while higher values mean weaker correlation. The reason for using the correlation ratio statistic is that it is a fairly general dependence test, capable of detecting not only linear dependencies but also non-linear ones. On the other hand, more standard dependence tests such as the correlation coefficient [15] can only capture linear dependencies, for example when the distribution is multivariate normal. Therefore, correlation ratio is a suitable choice for the initialization of MBM, given that our goal is to estimate densities without making assumptions on the nature of the modeled dependencies.

4.2 Local structure learning

In the BNs composing our nonparametric HRFs, the conditional density of each node given its parents is modeled by using NW estimators (as described in Sec. 3). For root nodes, the NW estimator clearly reduces to a standard (unconditional) Parzen window model [20]. An important point in MBM that needs to be addressed in a different way when dealing with continuous domains is the scoring function used to evaluate the structure of the local BNs. While the original version of MBM uses a heuristic function based on the minimum description length principle [6, 7], a natural evaluation function for kernel-based graphical models is the model CVLL with respect to the training dataset $\mathbf{D}$ [11, 12]. For a BN with graph $\mathcal{G}$ and nodes $X_1, \ldots, X_d$, if $\mathbf{D} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ and each $\mathbf{x}_j$ is a vector $(x_1, \ldots, x_d)$, the structure $\mathcal{G}$ is scored as follows:

$$CVLL(\mathcal{G}) = \sum_{j=1}^{n}\sum_{i=1}^{d} \log \hat{p}^{-j}(x_{i_j}|pa_j(X_i)) \tag{14}$$

where $pa_j(X_i)$ is the state of the parents of X_i in $\mathbf{x}_j$. Clearly, a CVLL-based strategy is much less prone to overfitting than a straight maximum-likelihood approach. The CVLL function is maximized (up to a local optimum) by heuristic search in the space of d-dimensional BN structures. To this aim, we use the greedy hill-climbing algorithm described in [6, 7].

4.3 Global structure learning

The last correction we introduce in the MBM algorithm concerns the evaluation function that we apply to the global structure of the HRF. Rather than maximizing straightly the model pseudo-log-likelihood with respect to the dataset, we optimize instead a cross-validated version of that function, consistently with the choices we made also for bandwidth selection and local structure learning. For a dataset $\mathbf{D}$ containing n d-dimensional patterns and a HRF with graphs $\mathcal{G}_1, \ldots, \mathcal{G}_d$, the cross-validated pseudo-log-likelihood (CVLL*) measure, denoted by $CVLL^*(\mathcal{G}_1, \ldots, \mathcal{G}_d)$, is defined by the following equation:

$$CVLL^*(\mathcal{G}_1, \ldots, \mathcal{G}_d) = \sum_{j=1}^{n}\sum_{i=1}^{d} \log \hat{p}^{-j}(x_{i_j}|mb_{i_j}(X_i)) \tag{15}$$

where $mb_{i_j}(X_i)$ is the state of the MB of X_i in pattern $\mathbf{x}_j$.

5 RELATED WORK

In directed and undirected graphical models, nonparametric conditional density estimators (based on Parzen windows) are used for the first time in [11, 12]. With respect to these models, in continuous HRFs we not only exploit double-kernel estimators (instead of single-kernel Parzen windows), but we also automate the task of bandwidth selection. A nonparametric technique for learning the structure of continuous BNs is also developed in [17]. However, that method is only aimed at inferring the conditional independencies from data, rather than at learning the overall density function. A semiparametric technique for learning undirected graphs, leading to so-called 'nonparanormal' MRFs (NPMRFs), is proposed in [16]. The nonparanormal approach consists in transforming the original data points (which are not assumed to satisfy any given distributional form) by mapping them onto a different set of points, which are assumed to follow a multivariate normal distribution. The graph is then estimated from the transformed dataset using the graphical lasso algorithm [8], which is both computationally efficient and theoretically sound for Gaussian distributions [22]. The idea of mapping the original dataset into a feature space where data are assumed to be normally distributed is also exploited in [1].

6 EXPERIMENTAL EVALUATION

The aim of this section is to evaluate the accuracy of KHRFs at modeling (multivariate) densities featuring nonlinear dependencies between the variables plus random noise (distributed in heterogeneous ways). To this aim, we sample a number of datasets from synthetic distributions, where the distributions are randomly generated in such a way as to make it unlikely that any particular parametric assumption may be satisfied. We then exploit the produced data for pattern classification, comparing the performance of our model to other probabilistic techniques. The data generation process is described in Sec. 6.1, while Sec. 6.2 illustrates the results of the experiments. The models and algorithms considered in the evaluation are implemented in the JProGraM software package, which is freely available at `http://www.dii.unisi.it/~freno/JProGraM.html` under an open-source license.

6.1 Random data generation

In order to generate datasets featuring nonlinear correlations between the variables, we exploit the idea of defining a random distribution based on a (randomly generated) directed acyclic graph (DAG), where each node corresponds to a random variable and each arc corresponds to a dependence of the child on the parent. Therefore, the data generation process is made up of three stages: first, we generate a random DAG with a specified number of nodes; second, we generate a random distribution from a specified DAG; third, we generate a random dataset from a specified (DAG-shaped) distribution.

6.1.1 Directed acyclic graph generation

Given a number d of nodes and a parameter p_{max} specifying the maximum number of parents allowed for each node, we generate a random DAG using Algorithm 2. We start by ordering the nodes from X_1 to X_n. Then, for each X_i, we randomly select p nodes from the set $\{X_1, \ldots, X_{i-1}\}$ (where p is a random integer in the interval $[0, \min\{i-1, p_{max}\}]$), and for each selected node X_j we introduce an edge from X_j to X_i. The resulting pair $(\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the set of vertices and $\mathcal{E}$ is the set of edges, is returned as output.

Algorithm 2 Random DAG generation

Input: Number d of nodes; integer p_{max}.
Output: DAG $\mathcal{G} = (\mathcal{V}, \mathcal{E})$.

```
1.  V = {X_1, ..., X_d}
2.  E = ∅
3.  for(i = 1 to d)
4.     p = random integer in [0, min{i − 1, p_max}]
5.     P = ∅
6.     while(|P| < p)
7.        j = random integer in [1, i − 1]
8.        P = P ∪ {(X_j, X_i)}
9.     E = E ∪ P
10. return (V, E)
```

6.1.2 Distribution generation

Algorithm 3 generates a random distribution from a DAG $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. The idea is that each edge (X_i, X_j) in the DAG represents a dependence of X_j on X_i, where the dependence is determined by a polynomial function of third degree $f_{j_i}(x) = a_1^{j_i} x^3 + a_2^{j_i} x^2 + a_3^{j_i} x + a_4^{j_i}$. The coefficients of each polynomial are selected randomly in the interval $[-a_{max}, a_{max}]$. Moreover, each node X_i is assigned a beta density function $beta_i(x)$, defined as follows (for $a < x < b$ and $\alpha_i, \beta_i > 0$):

$$beta_i(x) = \frac{\Gamma(\alpha_i + \beta_i)}{\Gamma(\alpha_i)\Gamma(\beta_i)(b-a)^{\alpha_i+\beta_i-1}}(x-a)^{\alpha_i-1}(b-x)^{\beta_i-1} \tag{16}$$

where

$$\Gamma(x) = \int_0^\infty t^{x-1} e^{-t}\, dt \tag{17}$$

The idea is that the values observed for variable X_i are subject to random noise, where the noise is distributed over the interval (a, b) according to a beta density with parameters α_i and β_i. For each $beta_i(x)$, the parameters α_i and β_i are randomly chosen in the intervals $(0, \alpha_{max}]$ and $(0, \beta_{max}]$ respectively, whereas a and b remain constant. Given the polynomials and the beta densities, the value of each X_i results from a linear combination of the related polynomial functions plus (beta-distributed) random noise. The output of Algorithm 3 is a pair $(\mathcal{F}_\mathcal{V}, \mathcal{F}_\mathcal{E})$ such that $\mathcal{F}_\mathcal{V} = \{beta_i(x) : X_i \in \mathcal{V}\}$ and $\mathcal{F}_\mathcal{E} = \{f_{i_j}(x) : (X_j, X_i) \in \mathcal{E}\}$.

Algorithm 3 Random distribution generation with DAG-shaped polynomial dependencies and beta-distributed noise

Input: DAG $\mathcal{G} = (\mathcal{V}, \mathcal{E})$; positive real numbers α_{max}, β_{max}, a_{max}; real numbers a, b such that $a < b$.
Output: DAG-shaped distribution $\mathcal{D}_\mathcal{G} = (\mathcal{F}_\mathcal{V}, \mathcal{F}_\mathcal{E})$.

```
1.  F_V = ∅
2.  for(i = 1 to |V|)
3.     α_i = random real in (0, α_max]
4.     β_i = random real in (0, β_max]
5.     beta_i(x) = beta(x; α_i, β_i, a, b)
6.     F_V = F_V ∪ {beta_i(x)}
7.  F_E = ∅
8.  for((X_i, X_j) ∈ E)
9.     for(k = 1 to 4)
10.       a_k^{j_i} = random real in [−a_max, a_max]
11.    f_{j_i}(x) = f(x; a_1^{j_i}, ..., a_4^{j_i})
12.    F_E = F_E ∪ {f_{j_i}(x)}
13. return (F_V, F_E)
```

6.1.3 Dataset generation

Given a distribution $\mathcal{D}_\mathcal{G} = (\mathcal{F}_\mathcal{V}, \mathcal{F}_\mathcal{E})$ organized in a DAG $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, Algorithm 4 generates patterns that are independent and identically distributed according to $\mathcal{D}_\mathcal{G}$. In order to produce a pattern $x_1, \ldots, x_d$, the algorithm determines the value of each variable X_i by first computing $\sum_{f_{i_j(x)} \in \mathcal{F}_\mathcal{E}} f_{i_j}(x_j)$, and then by adding to that sum a random value sampled from the density $beta_i(x)$, so as to introduce some noise. The ancestral ordering of the nodes $X_1, \ldots, X_d$ in $\mathcal{V}$ is followed so as to ensure that the argument of each function $f_{i_j}(x_j)$ has already been determined before computing the value of node X_i.

If one needs to generate data that are partitioned into several classes $\omega_1, \ldots, \omega_c$ (e.g. for the purposes of pattern classification),

the algorithm generates data for each ω_i (where $i > 1$) by deriving first a corresponding distribution $\mathcal{D}_{\mathcal{G}_i}$ from $\mathcal{D}_{\mathcal{G}_{i-1}}$ in the following way. For each polynomial $f_{j_k}(x)$ in $\mathcal{F}_{\mathcal{E}_{i-1}}$, the coefficients $a_1^{j_k}, \ldots, a_4^{j_k}$ are changed with probability P, where the change consists in multiplying each $a_l^{j_k}$ by a randomly selected real number in the interval $[-max_r, max_r]$. The resulting polynomial is used to replace $f_{j_k}(x)$ in $\mathcal{F}_{\mathcal{E}_i}$. Finally, the integers $n_1, \ldots, n_c$ specify the number of patterns to be generated for each class.

Algorithm 4 Random data generation from a DAG-shaped distribution

Input: DAG-shaped distribution $\mathcal{D}_{\mathcal{G}} = (\mathcal{F}_{\mathcal{V}}, \mathcal{F}_{\mathcal{E}})$; number c of classes; integers $n_1, \ldots, n_c$; real numbers P, max_r such that $0 < P \leq 1, max_r > 0$.
Output: Datasets $\mathbf{D}_1, \ldots, \mathbf{D}_c$.

```
for(i = 1 to c)
  D_i = ∅
  if(i > 1)
    distributionIsUnchanged = true
    while(distributionIsUnchanged)
      for(f_{j_k}(x) ∈ F_E)
        p = random real in [0, 1)
        if(p < P)
          for(l = 1 to 4)
            r = random real in [−max_r, max_r]
            a_l^{j_k} = r · a_l^{j_k}
          f_{j_k}(x) = f(x; a_1^{j_k}, ..., a_4^{j_k})
          distributionIsUnchanged = false
  for(j = 1 to n_i)
    for(k = 1 to d)
      x_{k_j} = random real sampled from beta_k(x)
      x_{k_j} = x_{k_j} + Σ_{f_{k_l(x)} ∈ F_E} f_{k_l}(x_{l_j})
    D_i = D_i ∪ {(x_{1_j}, ..., x_{d_j})}
return {D_1, ..., D_c}
```

6.2 Results

In order to test the accuracy of KHRFs at modeling joint densities (as learned by MBM), we apply them to a number of pattern classification tasks, where the datasets are generated using Algorithms 2–4. We consider eleven tasks, where each task is based on a different dataset $\mathbf{D}$ containing 500 patterns, and the patterns are equally divided in two classes ω_1 and ω_2. The data for each task are generated using each time a different (random) DAG. In particular, we choose a different number d of nodes for each DAG, where $5 \leq d \leq 25$. Then, we use the generated DAG as input for Algorithm 3. Here, we set $a_{max} = 2$ for the polynomial functions, while the beta densities are generated over the interval $[-2, 2]$, setting $\alpha_{max} = \beta_{max} = 2$. In a preliminary phase of the experiments, we found these parameters to be large enough to generate a suitably wide range of distributions. We use $c = 2$ and $n_1 = n_2 = 250$ as input values for Algorithm 4. Moreover, when changing the distribution from ω_1 to ω_2, we set $max_r = 2$ and $P = 0.1$. Our experience with preliminary results indicated that if the values of max_r and P (especially the latter) are too large (e.g. if $P \gtrsim 0.2$), the resulting classification tasks tend to be too easy to be dealt with, because patterns belonging to different classes are then distributed farther apart in the feature space. Before exploiting the datasets, we normalize the values of each feature X_i by transforming each x_{i_j} into $\frac{x_{i_j} - \min_k x_{i_k}}{\max_k x_{i_k} - \min_k x_{i_k}}$, where $1 \leq k \leq |\mathbf{D}|$.

We compare the performance of KHRFs to kernel-based BNs (KBNs), kernel-based MRFs (KMRFs), NPMRFs, and Gaussian MRFs (GMRFs). In kernel-based BNs and MRFs we estimate conditional densities in the same way as in KHRFs, while the model structure is learned using the algorithms proposed in [11] and [12] respectively. For the purposes of bandwidth selection, we always perform two iterations of Algorithm 1, whereas the limit points h_{max_1} and h_{max_2} are set differently for BNs, MRFs, and HRFs, based on preliminary validation on separate datasets. In particular, the used values are $h_{max_1} = 2$ and $h_{max_2} = 1$ for KBNs, $h_{max_1} = 1$ and $h_{max_2} = 2$ for KMRFs, and $h_{max_1} = 0.05$ and $h_{max_2} = 0.5$ for KHRFs. Structure learning in GMRFs and NPMRFs is performed as described in [8] and [16], using the graphical lasso technique, and conditional densities are then estimated within the resulting structures using Gaussian and nonparanormal conditional models respectively. To the best of our knowledge, the learning algorithms considered for KBNs, KMRFs, and NPMRFs are the state of the art in the literature on (continuous) nonparametric and semiparametric graphical models. On the other hand, GMRFs provide an authoritative term of comparison for evaluating the effect of relaxing the parametric assumption in density estimation.

In order to exploit the models for pattern classification, we partition the training data $\mathbf{D}$ for each task in two subsets $\mathbf{D}_1$ and $\mathbf{D}_2$, such that all patterns in $\mathbf{D}_i$ belong to class ω_i. For each model, we learn two class-specific versions, training each version on the respective dataset. Patterns in the test set are then classified as follows. For each class ω_i, we compute the posterior probability $P(\omega_i|x)$ that a pattern $\mathbf{x}$ belongs to class ω_i:

$$P(\omega_i|\mathbf{x}) = \frac{p(\mathbf{x}|\omega_i)P(\omega_i)}{p(\mathbf{x})} \tag{18}$$

where $p(\mathbf{x}|\omega_i)$ is the (pseudo-)likelihood of the model learned for class ω_i given $\mathbf{x}$, $P(\omega_i)$ is the prior probability of class ω_i (estimated as $\frac{|\mathbf{D}_i|}{|\mathbf{D}|}$), and $p(\mathbf{x}) = \sum_j p(\mathbf{x}|\omega_j)P(\omega_j)$. Given the posterior probability of each class, we attach to $\mathbf{x}$ the label with the highest probability, based on a maximum a posteriori strategy. The results of the experiments are reported in Table 1, where values are averaged by 5-fold cross-validation. For each model, we measure both recognition accuracy and training time (per class), where time was measured (in seconds) on a 2.34 GHz CPU.

Table 1 lends itself to the following interpretation. First, KHRFs are more accurate overall than the other models in terms of recognition rate. At the same time, learning KHRFs is much less expensive than learning kernel-based BNs and MRFs. Second, although GMRFs and NPMRFs are the most efficient models from the computational point of view, their advantage over KHRFs against the growth of the number of variables is not as significant as the advantage of GMRFs, NPMRFs, and KHRFs over KBNs and KMRFs. Third, the relatively low accuracy of GMRFs as compared to KHRFs, together with the fact that the improvement of NPMRFs over GMRFs is not as stable as one may wish, confirms that the distributions generating the datasets violate the parametric and semiparametric assumptions to a significant extent. Therefore, the considered experimental setting provides evidence not only that kernel-based HRFs are a very reasonable choice when no prior knowledge is available concerning the form of the distribution to be estimated, but also that KHRFs are the most promising option within the kernel-based family, both in terms of computational efficiency and prediction accuracy.

Table 1. Recognition accuracy (average ± standard deviation) and average training time (per class) measured by 5-fold cross-validation on synthetic datasets of growing dimensionality. For each dataset, d is the number of variables composing the data vectors. KHRFs are compared to other nonparametric (KBNs and KMRFs), semiparametric (NPMRFs), and parametric (GMRFs) graphical models. The highest accuracy value in each row is indicated in bold font.

	Recognition Accuracy (%) / Training Time (s)									
d	**KBN**		**GMRF**		**NPMRF**		**KMRF**		**KHRF**	
10	59.4 ± 4.4	31.5	61.2 ± 2.0	0.2	61.4 ± 3.8	0.2	59.6 ± 4.5	25.7	**62.6** ± 7.6	3.0
11	48.8 ± 4.6	33.8	51.6 ± 2.0	0.2	54.8 ± 4.5	0.2	49.2 ± 5.2	31.9	**62.4** ± 4.8	4.7
12	80.8 ± 2.8	57.6	70.6 ± 4.2	0.2	**87.2** ± 3.1	0.2	79.0 ± 2.6	27.1	82.2 ± 2.3	4.7
13	56.2 ± 3.9	63.9	53.2 ± 2.4	0.2	56.4 ± 33.6	0.2	56.4 ± 4.6	60.3	**65.0** ± 4.5	3.1
14	83.8 ± 4.7	93.0	87.2 ± 2.3	0.2	75.6 ± 7.7	0.3	85.0 ± 5.0	69.2	**88.4** ± 2.9	6.1
15	68.0 ± 2.6	90.4	58.8 ± 3.4	0.2	30.0 ± 24.4	0.3	68.4 ± 4.8	93.7	**73.8** ± 3.7	5.0
16	55.0 ± 6.3	146.7	76.0 ± 4.6	0.2	66.4 ± 7.3	0.3	63.2 ± 4.5	82.8	**81.4** ± 1.7	3.0
17	52.2 ± 2.1	141.3	50.6 ± 1.3	0.2	**63.4** ± 1.8	0.3	55.8 ± 3.9	102.8	56.4 ± 2.4	3.4
18	58.2 ± 7.3	152.1	53.6 ± 2.7	0.3	64.0 ± 3.2	0.3	62.2 ± 5.9	136.7	**75.2** ± 2.7	3.9
19	97.8 ± 0.7	196.2	97.6 ± 1.8	0.3	91.0 ± 1.7	0.3	98.4 ± 0.8	167.2	**98.8** ± 0.7	4.3
20	96.6 ± 2.5	270.0	**98.4** ± 1.0	0.3	78.8 ± 3.5	0.3	96.6 ± 1.4	212.4	97.4 ± 1.2	6.0

7 CONCLUSIONS AND FUTURE WORK

The main contribution of this work was to show that a continuous version of the HRF model can be built out of Nadaraya-Watson estimators, with very promising experimental results in terms of prediction accuracy. At the same time, the scalability properties of MBM with respect to the number of variables are preserved by KHRFs. Since a serious limitation of kernel-based estimators is that they do not scale well with respect to the number of data points, one direction for future research is to investigate the behavior of KHRFs when a dual-tree fast approximation to the CVLL* metric is used for bandwidth selection, as suggested in Sec. 3.2. On the other hand, although we believe that the methodology we used for generating synthetic benchmarks captures a fairly general class of distributions, it would be useful to evaluate the prediction accuracy of KHRFs not only on different synthetic distributions, considering alternative dependence relationships and density functions, but also in real-world applications.

ACKNOWLEDGEMENTS

The authors are grateful to Ilaria Castelli for reading and commenting a preliminary draft of this paper.

REFERENCES

[1] Francis R. Bach and Michael I. Jordan, 'Learning Graphical Models with Mercer Kernels', in *Advances in Neural Information Processing Systems*, pp. 1009–1016, (2002).

[2] Julian Besag, 'Statistical Analysis of Non-Lattice Data', *The Statistician*, **24**, 179–195, (1975).

[3] Christopher M. Bishop, *Pattern Recognition and Machine Learning*, Springer, New York (NY), 2006.

[4] Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*, John Wiley & Sons, New York (NY), second edn., 2001.

[5] V.A. Epanechnikov, 'Nonparametric Estimation of a Multidimensional Probability Density', *Theory of Probability and its Applications*, **14**, 153–158, (1969).

[6] Antonino Freno, Edmondo Trentin, and Marco Gori, 'A Hybrid Random Field Model for Scalable Statistical Learning', *Neural Networks*, **22**, 603–613, (2009).

[7] Antonino Freno, Edmondo Trentin, and Marco Gori, 'Scalable Pseudo-Likelihood Estimation in Hybrid Random Fields', in *Proceedings of the 15th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD 2009)*, eds., J.F. Elder, F. Fogelman-Souli, P. Flach, and M. Zaki, pp. 319–327. ACM, (2009).

[8] Jerome Friedman, Trevor Hastie, and Robert Tibshirani, 'Sparse Inverse Covariance Estimation with the Graphical Lasso', *Biostatistics*, **9**, 432–441, (2008).

[9] Walter R. Gilks, Sylvia Richardson, and David Spiegelhalter, *Markov Chain Monte Carlo in Practice*, Chapman & Hall/CRC, 1996.

[10] Alexander G. Gray and Andrew W. Moore, ''N-Body' Problems in Statistical Learning', in *Advances in Neural Information Processing Systems*, pp. 521–527, (2000).

[11] Reimar Hofmann and Volker Tresp, 'Discovering Structure in Continous Variables Using Bayesian Networks', in *Advances in Neural Information Processing Systems*, pp. 500–506, (1995).

[12] Reimar Hofmann and Volker Tresp, 'Nonlinear Markov Networks for Continous Variables', in *Advances in Neural Information Processing Systems*, (1997).

[13] Michael P. Holmes, Alexander G. Gray, and Charles L. Isbell, 'Fast Nonparametric Conditional Density Estimation', in *Proceedings of the 23rd Conference on Uncertainty in Artificial Intelligence (UAI '07)*, (2007).

[14] J.F. Kenney and E.S. Keeping, *Mathematics of Statistics. Part 2*, Van Nostrand, Princeton (NJ), second edn., 1951.

[15] J.F. Kenney and E.S. Keeping, *Mathematics of Statistics. Part 1*, Van Nostrand, Princeton (NJ), third edn., 1962.

[16] Han Liu, John Lafferty, and Larry Wasserman, 'The Nonparanormal: Semiparametric Estimation of High Dimensional Undirected Graphs', *Journal of Machine Learning Research*, **10**, 2295–2328, (2009).

[17] Dimitris Margaritis, 'Distribution-Free Learning of Bayesian Network Structure in Continuous Domains', in *AAAI*, pp. 825–830, (2005).

[18] Elizbar A. Nadaraya, 'On Estimating Regression', *Theory of Probability and its Applications*, **9**, 141–142, (1964).

[19] Richard E. Neapolitan, *Learning Bayesian Networks*, Prentice Hall, Upper Saddle River (NJ), 2004.

[20] Emanuel Parzen, 'On Estimation of a Probability Density Function and Mode', *Annals of Mathematical Statistics*, **33**, 1065–1076, (1962).

[21] Judea Pearl, *Probabilistic Reasoning in Intelligent Systems*, Morgan Kaufmann, San Francisco (CA), 1988.

[22] Pradeep Ravikumar, Garvesh Raskutti, Martin Wainwright, and Bin Yu, 'Model Selection in Gaussian Graphical Models: High-Dimensional Consistency of ℓ_1-regularized MLE', in *Advances in Neural Information Processing Systems*, pp. 1329–1336, (2008).

[23] M. Rosenblatt, 'Conditional Probability Density and Regression Estimators', in *Multivariate Analysis*, ed., P.R. Krishnaiah, volume II, 25–31, Academic Press, New York, (1969).

[24] B.W. Silverman, *Density Estimation for Statistics and Data Analysis*, Chapman and Hall, 1986.

[25] Geoffrey S. Watson, 'Smooth Regression Analysis', *Sankhyā: The Indian Journal of Statistics, Series A*, **26**, 359–372, (1964).