Online Estimation of Discrete Densities using Classifier Chains

Michael Geilke[1] and Eibe Frank[2] and Stefan Kramer[1]

[1] Johannes Gutenberg-Universtität Mainz, Germany
{geilke,kramer}@informatik.uni-mainz.de
[2] Department of Computer Science, University of Waikato, New Zealand
eibe@cs.waikato.ac.nz

Abstract. We propose an approach to estimate a discrete joint density online, that is, the algorithm is only provided the current example, its current estimate, and a limited amount of memory. To design an online estimator for discrete densities, we use classifier chains to model dependencies among features. Each classifier in the chain estimates the probability of one particular feature. Because a single chain may not provide a reliable estimate, we also consider ensembles of classifier chains. Our experiments on synthetic data show that the approach is feasible and the estimated densities approach the true, known distribution with increasing amounts of data.

1 Introduction

Whereas many data mining tasks have received considerable attention in the context of stream mining recently, only little is known about the estimation of joint densities in an online setting. Offline density estimation includes recent work based on decision trees [12], where the leaves contain piecewise constant estimators. A similar approach was pursued by Davies and Moore [2] as part of a conditional density estimator. Vapnik and Mukherjee [15] used SVMs to perform density estimation. Multi-variate densities are frequently estimated using kernel density estimators ([7, 14]). Kernel density estimation is also the predominant direction of the few online variants of density estimation so far, e.g., by Kristan *et al.* [8, 9] and Lambert *et al.* [10].

In this paper, we propose a different approach to online estimation of discrete joint densities (notice that we use discrete densities as a synonym for probability mass functions) based on so-called classifier chains ([13, 3]), which uses a set of probabilistic online classifiers to model a discrete joint probability distribution. In this way, one can build on existing work in online learning and hence take advantage of scalable, well-performing algorithms. The classifiers in a chain aim to model the class probabilities of a particular feature, and the overall chain aims to model the dependencies among the features. The individual estimates are combined using the product rule. Because a single classifier chain may not be sufficiently robust, we also provide a variant that uses ensembles of classifier chains. We evaluate our density estimators by using discrete joint densities that were generated with Bayesian networks.

The paper is organized as follows. In Section 2, we describe a method to perform online estimation of discrete joint densities using ensembles of classifier chains. This method is evaluated in Section 3, where we define the experimental set-up and present the results of our experiments. Section 4 concludes the paper.

2 Online Density Estimation

Let $X_1, X_2, \ldots, X_n$ be nominal features and $f(X_1, X_2, \ldots, X_n)$ be an unknown discrete joint density. We present algorithms that, given an infinite stream of data that is distributed according to f, determine a density estimate $\hat{f}$ for f. The data stream is processed in an online fashion, that is, an algorithm is only provided the current example, its current density estimate, and a limited amount of memory. After each example, the algorithm returns its updated estimate.

2.1 Classifier Chain

As a first step, we provide an algorithm that uses a classifier chain to determine a density estimate. Let $f(X_1, X_2, \ldots, X_n)$ be a discrete joint density. Then we can apply the product rule and obtain the following equality:

$$f(X_1, X_2, \ldots, X_n) = f_1(X_1) \cdot \prod_{i=2}^{n} f_i(X_i \mid X_1, X_2, \ldots, X_{i-1}). \tag{1}$$

In other words, in order to model the discrete joint density f, it is sufficient to model the density $f_1(X_1)$ and the conditional densities $f_i(X_i \mid X_1, X_2, \ldots, X_{i-1})$, $i \in \{2, \ldots, n\}$. The product over these estimates yields an estimate of the joint density f. To model the individual densities f_i, $1 \leq i \leq n$, we use classifiers that return class probability estimates. For $f_1(X_1)$, we use a majority class classifier and for $f_i(X_i \mid X_1, X_2, \ldots, X_{i-1})$, $i \in \{2, \ldots, n\}$, we use Hoeffding trees [4]. Both allow us to estimate the density in an online fashion. Note that we assume the underlying density stays fixed. In case the density changes over time, one could potentially use classifiers that are able to deal with concept drift such as Concept-adapting Very Fast Decision Tree learner [6], but this is not considered here.

Algorithm 1 describes the process of updating a density estimate when a new example arrives. It is based on the classifier chain implied by Equation 1 as described above. First, Algorithm 1 receives an example $(x_1, x_2, \ldots, x_n)$ from an instance stream. Then it produces n examples, where example i contains the features $X_1, X_2, \ldots, X_i$. The example (x_1) is forwarded to the classifier for $f(X_1)$ and the example $(x_1, x_2, \ldots, x_i)$ is forwarded to the classifier for $f(X_i \mid X_1, X_2, \ldots, X_{i-1})$, $i \in \{2, \ldots, n\}$. Afterwards each classifier processes its example and updates its current estimate.

Note that in applications where we need to draw an instance from our density estimate, we simply iterate over the classifiers from $f_1(X_1)$ to $f_n(X_n \mid$

Algorithm 1 Density estimation using a fixed classifier chain.

```
Require: $(x_1, x_2, \dots, x_n)$, a chain $cc$ of $n$ classifiers ($cc[1]$ is a majority class classifier
    and $cc[i]$ are Hoeffding trees for $i \in \{2, 3, \dots, n\}$)
for $i \in \{1, 2, \dots, n\}$ do
    train classifier $cc[i]$ on instance $(x_1, x_2, \dots, x_i)$
end for
```

$X_1, X_2, \dots, X_{n-1}$), draw an estimate from each classifier, sample a value based on the distribution obtained, and use the output as input for the next classifier. In a similar fashion, we can also compute the probability of a given instance.

2.2 Ensembles of Classifier Chains

The product on the right-hand side of Equation 1 is only one way to represent the discrete joint density – there are many other possibilities. Let $m : \{1, 2, \dots, n\} \to \{1, 2, \dots, n\}$ be a bijective mapping. Then

$$f(X_1, X_2, \dots, X_n) = f(X_{m(1)}) \cdot \prod_{i=2}^{n} f(X_{m(i)} \mid X_{m(1)}, X_{m(2)}, \dots, X_{m(i-1)}). \quad (2)$$

In other words, we simply use a different ordering of the features to represent the discrete joint density, which then results in a different classifier chain. Although all such products represent the same joint density assuming the true conditional density estimates are known, the ordering may be important for the performance of our classifiers: ideally, the ordering enables the classifiers to exploit conditional independence relationships so that some of the features can be ignored. Hence, to increase robustness, we consider another algorithm that generates several classifier chains and combines their estimates to a single density estimate. This algorithm, which generates ensembles of classifier chains, simply samples chains at random from the set of possible feature orderings and averages the density estimates obtained.

Note that although it is straightforward to obtain a density estimate for a particular instance from the ensemble, it is no longer straightforward to generate data samples based on the estimated density. The simple process that can be used in the case of a single chain no longer applies.

3 Experiments

In this section, we evaluate the algorithms presented in the previous section. Our online density estimator has been implemented in the MOA framework [1] using the classifiers `MajorityClass` and `HoeffdingTree`, where we use `MajorityClass` as leave classifiers for the Hoeffding trees and disabled pre-pruning. In order to compare the performance of the online density estimator with an offline variant, we implemented the same algorithm in the WEKA framework [5] using the corresponding offline classifiers `ZeroR` and `REPTree` with pruning disabled.

3.1 Experimental Set-up

To compare the performance of the algorithms, we used Bayesian networks to randomly generate discrete joint densities using the `BayesNetGenerator` of the WEKA framework. For each density, we specified the number of nodes and the number of possible node values. The number of arcs were set to 10. Using the Bayesian network, we then generated 10^3, 10^4, or 10^5 instances.

The resulting density estimates were evaluated using the Kullback-Leibler divergence (KL-divergence). Let f be the true density and $\hat{f}$ be an estimate. Then the Kullback-Leibler divergence is defined as follows:

$$\sum_i \hat{f}(i) \cdot \ln \frac{\hat{f}(i)}{f(i)},$$

where i are the possible values that the density can take. Unfortunately, computing the KL-divergence is computational expensive, since we have to consider every possible combination of feature values. For instance, if we have 8 nodes with 7 values each, then there are already $7^8 = 5764801$ different combinations, for each of which we have to compute the probability given by the estimator. This computation is very expensive so that we can only consider discrete joint densities with a small number of nodes and a small number of node values. Notice that, in order to avoid rounding errors, we computed the KL-divergence using logarithms [11].

3.2 Experiment 1

In the first experiment, we generated discrete joint densities using Bayesian networks with 10 arcs and 5 to 8 nodes each of which has cardinality 4. Then, we drew M instances from this joint density with M being one of the values of $\{10^3, 10^4, 10^5\}$. The instances were forwarded to our online and offline density estimators. In both cases, we consider two variants of the algorithms: one with a single random classifier chain, one with an ensemble of classifier chains. For the latter, we chose a number of chains that is linear in the number of features, which in this case is 1 times the number of features. In both cases, 10 independent runs were performed, but the data remained the same in each run.

The results are summarized in Table 1. We start the discussion by comparing the online variant of `CC` with the online variant of `ECC`. First of all, we observe that in all cases the average KL-divergence of `ECC` is lower than the average KL-divergence of `CC`. In particular, the average KL-divergence of `ECC` is, for $M = 10^3$, between 9.7% and 31.2% lower, for $M = 10^4$, between 31.3% and 60.9% lower, and, for $M = 10^5$, between 60.0% and 81.8% lower. For the minimal and maximal KL-divergence, similar observation can be made. `ECC` performs better in all cases. But, more importantly, in 11 out of the 12 cases, the difference between the minimal and maximal KL-divergence for `ECC` is smaller than the corresponding difference for `CC` – the only exception is $n = 7$ and $M = 10^3$. Whereas the maximal difference for `ECC` is 0.420 ($n = 6$ and $M = 10^3$), the

Table 1. n is the number of nodes of the Bayesian network, where each node has cardinality 4. M is the number of instances that we gave as input to the estimators. For a given n, we always use the same sequence of instances. *CC* refers to a variant where a single random classifier chain is used. *ECC* refers to an ensemble of classifier chains for which we use n random classifier chains. For all such combinations, we computed the average KL-divergence, the minimal KL-divergence, and the maximal KL-divergence for the online and offline density estimators. For each combination, the data was computed from 10 independent runs.

		Online						Offline					
		CC			ECC			CC			ECC		
n	M	avg	min	max	avg	min	max	avg	min	max	avg	min	max
5	10^3	1.101	0.997	1.162	0.982	0.916	1.060	0.296	0.296	0.297	0.296	0.296	0.296
	10^4	0.640	0.423	0.936	0.272	0.236	0.343	0.049	0.049	0.049	0.049	0.049	0.049
	10^5	0.053	0.029	0.094	0.010	0.007	0.012	0.005	0.005	0.005	0.005	0.005	0.005
6	10^3	1.602	1.361	1.889	1.218	1.032	1.451	0.620	0.619	0.620	0.619	0.619	0.619
	10^4	0.476	0.279	0.905	0.186	0.108	0.267	0.127	0.127	0.127	0.127	0.127	0.127
	10^5	0.061	0.013	0.204	0.022	0.012	0.034	0.018	0.018	0.018	0.018	0.018	0.018
7	10^3	1.263	1.233	1.284	1.141	1.031	1.223	1.489	1.489	1.489	1.489	1.489	1.489
	10^4	0.546	0.198	1.038	0.217	0.118	0.298	0.377	0.377	0.377	0.377	0.377	0.377
	10^5	0.079	0.011	0.252	0.017	0.009	0.036	0.064	0.064	0.064	0.064	0.064	0.064
8	10^3	0.832	0.652	0.959	0.572	0.519	0.624	1.902	1.901	1.903	1.902	1.901	1.902
	10^4	0.524	0.439	0.624	0.360	0.348	0.370	0.651	0.651	0.651	0.651	0.651	0.651
	10^5	0.198	0.180	0.231	0.079	0.077	0.081	0.147	0.147	0.147	0.147	0.147	0.147

maximal difference for CC is 0.840 ($n = 7$ and $M = 10^4$). So, compared to ECC, the performance of our density estimator using a single classifier chain depends more heavily on the chain that was used.

The situation in the offline setting is completely different. For the average KL-divergence, we observe no difference between CC and ECC. Almost the same holds for the minimal and maximal KL-divergence. Differences can only be observed in three cases ($n = 5$ and $M = 10^3$, $n = 6$ and $M = 10^3$, $n = 8$ and $M = 10^3$) all of which are negligible due to insignificant differences.

If we compare the online and offline density estimators, we find cases where the offline variant performs better and cases where the online variant performs better. For $n = 5$ and $n = 6$, the offline algorithms have a lower average KL-divergence than the online algorithms. However, if many instances are available, then the average KL-divergence of the online ECC is very close to the offline ECC. For $n \in \{7, 8\}$ and $M \in \{10^3, 10^4, 10^5\}$, the online ECC has a lower average KL-divergence than the offline algorithms, whereas the average KL-divergence of the online CC is only lower for the pairs ($n = 7$, $M = 10^3$), ($n = 8, M = 10^3$) and ($n = 8, M = 10^4$).

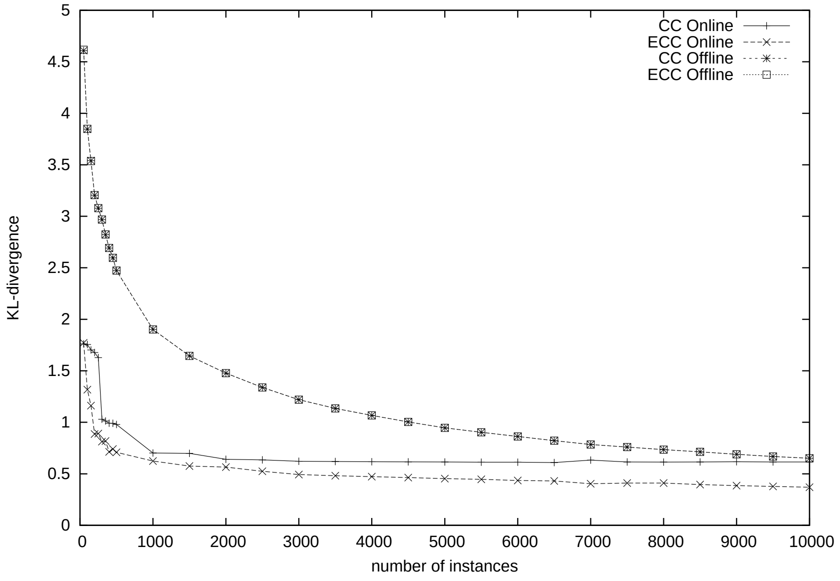

Fig. 1. The plot shows the KL-divergence with respect to the number of instances. It is based on a discrete joint density that was generated from a Bayesian network with 8 nodes where each node has cardinality 4 – the same that was used in the first experiment. The KL-divergence was measured for M instances with $M \in \{50, 100, 150, \ldots, 500\} \cup \{1000, 1500, 2000, \ldots, 10000\}$.

In order to provide a more detailed analysis of the algorithms' behavior, we extended the set M to include the values $\{50, 100, 150, 200 \ldots, 500\}$ and $\{1000, 1500, 2000, \ldots, 10000\}$. Then we used the same discrete joint density that we generated in the first experiment for the case $n = 8$, forwarded the instances to our algorithms, and measured the KL-divergence of the corresponding density estimators. The results are illustrated by Figure 1, which shows the KL-divergence of the density estimators with respect to the number of instances. The KL-divergence for the offline algorithms is indistinguishable. The online algorithms perform better than the offline algorithms, and, in particular, the density estimator ECC has a lower KL-divergence than any other density estimator. Moreover, the density estimator of the online ECC seems to converge faster than the one of the online CC. We also observe that the offline density estimators converge faster than the density estimator of the online CC, which results in almost the same KL-divergence when 10^4 instances are available. Figure 2 shows the continuation of the experiment, where the KL-divergence was measured for larger numbers of instances, namely $\{5 \cdot 10^3\}$, $\{1 \cdot 10^4, 2 \cdot 10^4, \ldots, 9 \cdot 10^4\}$, and

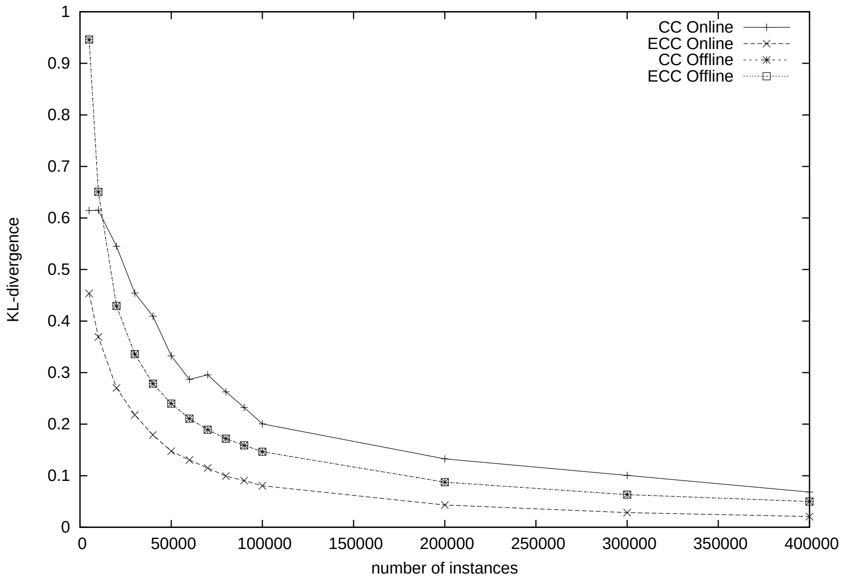

Fig. 2. The plot shows the KL-divergence with respect to the number of instances. It is based on a discrete joint density that was generated from a Bayesian network with 8 nodes where each node has cardinality 4 – the same that was used in the first experiment. The KL-divergence was measured for M instances with $M \in \{5 \cdot 10^3\} \cup \{1 \cdot 10^4, 2 \cdot 10^4, \ldots, 9 \cdot 10^4\} \cup \{1 \cdot 10^5, 2 \cdot 10^5, \ldots, 4 \cdot 10^5\}$.

$\{1 \cdot 10^5, 2 \cdot 10^5, \ldots, 4 \cdot 10^5\}$. The online `ECC` still performs better than any other algorithm, but the average performance of the online `CC` drops below the density estimators of the offline algorithms between 10000 and 20000 instances. This is due to faster rate of convergence, which we already observed in Figure 1.

Based on this first experiment, it seems that the online density estimator using an ensemble of classifier chains provides the best estimate if many instances are available and the discrete joint densities to be estimated have many features.

3.3 Experiment 2

In the first experiment, we observed that the performance of our online density estimator using a single classifier chain depended on the classifier chain that was used. In one case, the minimal KL-divergence was 0.198 and the maximal KL-divergence was 1.038. To investigate this matter further, we conducted another experiment. We assumed that a classifier chain that models the conditional independencies correctly performs better than a classifier chain that misses many

Table 2. n is the number of nodes of the Bayesian network, where each node has cardinality 4. M is the number of instances that we gave as input to the online estimators. *KL (Fixed)* is the KL-divergence for online estimator using a classifier chain that is sorted according to the topological order of the Bayesian network. *KL (min)* and *KL (max)* is based on 10 independent runs of online estimators with classifier chains having a random order. *KL (min)* is the minimal KL-divergence that has been observed and *KL (max)* is the maximal KL-divergence that has been observed. In both cases, *KL (Fixed)* has been excluded.

n	M	KL (Fixed)	KL (min)	KL (max)
5	10^3	1.154	0.997	1.162
	10^4	0.424	0.423	0.936
	10^5	0.093	0.029	0.094
6	10^3	1.198	1.361	1.889
	10^4	0.299	0.279	0.905
	10^5	0.022	0.013	0.204
7	10^3	1.148	1.233	1.284
	10^4	0.200	0.198	1.038
	10^5	0.011	0.011	0.252
8	10^3	0.708	0.652	0.959
	10^4	0.403	0.439	0.624
	10^5	0.059	0.180	0.231
9	10^3	2.726	2.566	4.356
	10^4	1.815	2.079	2.541
	10^5	1.832	1.807	2.086
10	10^3	2.658	2.640	3.258
	10^4	2.203	2.181	2.560
	10^5	1.949	1.930	2.143

conditional independencies due to a bad chain ordering. Therefore, we compared the performance of an online estimator using a classifier chain that is sorted according to the topological order of the Bayesian network with an online estimators using random classifier chains.

We generated discrete joint densities from Bayesian networks with 10 arcs and 5 to 10 nodes each of which has cardinality 4. Then, we drew M instances from these joint densities with M being one of the values of $\{10^3, 10^4, 10^5\}$. The instances were forwarded to 11 online density estimators all of which used a single classifier chain. One density estimators used a classifier chain that is sorted according to the topological order of the Bayesian network, the other density estimators used random classifier chains. The results are summarized in Table 2. `KL (Fixed)` is the KL-divergence of the online density estimator using a chain ordering that is sorted according to the topological order of the Bayesian network. `KL (min)` is the minimal KL-divergence of density estimators using a random chain ordering. `KL (max)` is the same as `KL (min)` except that density estimator with maximal KL-divergence is chosen.

In 5 out of 18 cases, `KL (Fixed)` is between 6.8% and 67.7% lower than `KL (min)`. In contrast to that, in 13 out of 18 cases, `KL (min)` is between 0.2% and 69.3% lower than `KL (Fixed)`. When comparing the KL-divergence of the estimator with fixed classifier chain and the worst density estimator with a random classifier chain, we observe a lower KL-divergence for the estimator with fixed classifier chain in all cases. The improvements range between 0.1% and 163.0%. So the order of the classifier chains does make a difference, but a classifier chain that is sorted according to the topological order of the Bayesian network does not necessarily provide the best estimate.

3.4 Experiment 3

In the first experiment, we observed performance improvements in the online setting when ensembles of classifier chains were used instead of a single classifier chain. There, we considered an ensemble having n classifier chains, which is 1 times the number of features. As a next step, we want to investigate whether larger ensembles can improve the performance even further. But since we are interested in fast online algorithms, we only consider linear numbers of classifier chains, that is, linear in the number of features.

We repeat the first experiment with online estimators using a single random classifier chain, and online estimators using an ensemble of $i \cdot n$ classifier chains, where n is the number of features and $i \in \{1, 2, 3\}$. The results are summarized in Table 3. We observe that with increasing numbers of classifier chains, the average KL-divergence decreases. Compared to a density estimator using a single classifier chain, the average KL-divergence improves, for `ECC1`, between 9.7% and 81.8%, for `ECC2`, between 19.0% and 86.3%, and, for `ECC3`, between 31.6% and 87.6%. Similary, the difference between the minimal and maximal KL-divergence decreases with increasing numbers of classifier chains. However, the larger the number of instances the lower the performance increase. For instance, for $n = 5$ and $M = 10^3$, the average KL-divergence decreases, for `ECC1`, by 10.8%, and, for `ECC3`, by 31.6%. In contrast to that, for $n = 5$ and $M = 10^5$, the average KL-divergence decreases, for `ECC1`, by 81.8%, and, for `ECC3`, by 87.6%. Considering that we expect many instances in an online setting, we assume that a smaller ensemble of classifier chains should be sufficient for most purposes.

4 Conclusions and Future Work

In this paper, we proposed two online algorithms to estimate discrete joint densities: one that uses a random classifier chain, one that uses an ensemble of random equally weighted classifier chains. We could show that the algorithms provide density estimates that approach the real density with increasing numbers of instances. When we considered the algorithms in an offline setting, the

Table 3. n is the number of nodes of the Bayesian network, where each node has cardinality 4. M is the number of instances that we gave as input to the estimators. For a given n, we always use the same sequence of instances. *CC* refers to an online estimator where a single random classifier chain is used. *ECC i* refers to an ensemble of classifier chains for which we use $i{\cdot}n$ random classifier chains. For all such combinations, we computed the average KL-divergence, the minimal KL-divergence, and the maximal KL-divergence for the online density estimators. For each combination, the data was computed from 10 independent runs.

		CC			ECC 1			ECC 2			ECC 3		
n	M	avg	min	max	avg	min	max	avg	min	max	avg	min	max
5	10^3	1.101	0.997	1.162	0.982	0.916	1.060	0.892	0.842	0.940	0.753	0.715	0.795
	10^4	0.640	0.423	0.936	0.272	0.236	0.343	0.168	0.134	0.205	0.128	0.109	0.146
	10^5	0.053	0.029	0.094	0.010	0.007	0.012	0.007	0.007	0.008	0.007	0.006	0.007
6	10^3	1.602	1.361	1.889	1.218	1.032	1.451	0.940	0.848	1.047	0.731	0.643	0.798
	10^4	0.476	0.279	0.905	0.186	0.108	0.267	0.128	0.084	0.167	0.096	0.072	0.116
	10^5	0.061	0.013	0.204	0.022	0.012	0.034	0.016	0.010	0.023	0.014	0.011	0.017
7	10^3	1.263	1.233	1.284	1.141	1.031	1.223	0.943	0.829	1.077	0.768	0.647	0.858
	10^4	0.546	0.198	1.038	0.217	0.118	0.298	0.121	0.079	0.150	0.091	0.061	0.121
	10^5	0.079	0.011	0.252	0.017	0.009	0.036	0.014	0.011	0.019	0.015	0.013	0.018
8	10^3	0.832	0.652	0.959	0.572	0.519	0.624	0.537	0.514	0.567	0.518	0.506	0.531
	10^4	0.524	0.439	0.624	0.360	0.348	0.370	0.300	0.292	0.311	0.258	0.254	0.264
	10^5	0.198	0.180	0.231	0.079	0.077	0.081	0.054	0.053	0.055	0.045	0.045	0.046

density estimator using a single random classifier chain performed as well as the density estimator using an ensemble of classifier chains. In contrast, in the online setting, an ensemble of classifier chains proved to be a useful tool to provide better density estimates. However, with increasing numbers of classifier chains, only minor improvements were observed when many instances are available.

Our online density estimator using an ensemble of classifier chains performed well on our synthetic data. Having an online method for estimating probability mass functions at our disposal should be useful for answering queries on the distribution of discrete random variables instantaneously and interactively, according to the requirements of human users.

In future work, we would like to experiment with large real-world data with drifting distributions and different underlying base classifiers, and develop a theory for ensembles of classifier chains for density estimation. Moreover, we would like to compare our new method with other approaches and consider a broader range of discrete joint densities in the experiments. Computing the Kullback-Leibler divergence is computationally expensive, and restricted densities could be considered. Furthermore, one could run those experiments on a cluster or on the cloud, thereby allowing consideration of a larger number of features and feature values. Finally, we would like to extend the work towards continuous joint densities and conditional densities. There are several offline algorithms that attempt to estimate such densities, but there appears to be little work in the online area.

References

1. Bifet, A., Holmes, G., Pfahringer, B., Kranen, P., Kremer, H., Jansen, T., Seidl, T.: Moa: Massive online analysis, a framework for stream classification and clustering. Journal of Machine Learning Research - Proceedings Track **11** (2010) 44–50
2. Davies, S., Moore, A.W.: Interpolating conditional density trees. In: Uncertainty in Artificial Intelligence. (2002) 119–127
3. Dembczynski, K., Cheng, W., Hüllermeier, E.: Bayes optimal multilabel classification via probabilistic classifier chains. In: International Conference on Machine Learning. (2010) 279–286
4. Domingos, P., Hulten, G.: Mining high-speed data streams. In: Knowledge Discovery and Data Mining. (2000) 71–80
5. Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I.H.: The weka data mining software: an update. SIGKDD Explorations **11**(1) (2009) 10–18
6. Hulten, G., Spencer, L., Domingos, P.: Mining time-changing data streams. In: Knowledge Discovery and Data Mining. (2001) 97–106
7. Hwang, J.N., Lay, S.R., Lippman, A.: Nonparametric multivariate density estimation: a comparative study. IEEE Transactions on Signal Processing **42**(10) (1994) 2795–2810
8. Kristan, M., Leonardis, A.: Online discriminative kernel density estimation. In: International Conference on Pattern Recognition. (2010) 581–584
9. Kristan, M., Leonardis, A., Skocaj, D.: Multivariate online kernel density estimation with gaussian kernels. Pattern Recognition **44**(10-11) (2011) 2630–2642
10. Lambert, C.G., Harrington, S.E., Harvey, C.R., Glodjo, A.: Efficient on-line nonparametric kernel density estimation. Algorithmica **25**(1) (1999) 37–57
11. Mann, T.P.: Numerically stable hidden Markov model implementation. An HMM scaling tutorial (2006) 1–8
12. Ram, P., Gray, A.G.: Density estimation trees. In: Knowledge Discovery and Data Mining. (2011) 627–635
13. Read, J., Pfahringer, B., Holmes, G., Frank, E.: Classifier chains for multi-label classification. Machine Learning **85**(3) (2011) 333–359
14. Scott, D.W., Sain, S.R. In: Multi-Dimensional Density Estimation. Elsevier, Amsterdam (2004) 229–263
15. Vapnik, V., Mukherjee, S.: Support vector method for multivariate density estimation. In: Neural Information Processing Systems. (1999) 659–665