Journal of Intelligent Information Systems (2019) 53:199–217
https://doi.org/10.1007/s10844-019-00549-w

Ensembles of density estimators for positive-unlabeled learning

T. M. A. Basile[1,2] · N. Di Mauro[3] · F. Esposito[3] · S. Ferilli[3] · A. Vergari[3]

Received: 28 July 2018 / Revised: 12 February 2019 / Accepted: 13 February 2019 /
Published online: 1 March 2019

Abstract
Positive-Unlabeled (PU) learning works by considering a set of positive samples, and a (usually larger) set of unlabeled ones. This challenging setting requires algorithms to cleverly exploit dependencies hidden in the unlabeled data in order to build models able to accurately discriminate between positive and negative samples. We propose to exploit probabilistic generative models to characterize the distribution of the positive samples, and to label as reliable negative samples those that are in the lowest density regions with respect to the positive ones. The overall framework is flexible enough to be applied to many domains by leveraging tools provided by years of research from the probabilistic generative model community. In addition, we show how to create mixtures of generative models by adopting a well-known bagging method from the discriminative framework as an effective and cheap alternative to the classical Expectation Maximization. Results on several benchmark datasets show the performance and flexibility of the proposed approach.

Keywords Positive-unlabeled learning · Density estimator

1 Introduction

The classical supervised setting of statistical machine learning (Hastie et al. 2009) aims at inducing models (classifiers) from training sets of labeled data in the form of samples $(\mathbf{x}^i, y^i)$ i.i.d. drawn from an unknown joint probability distribution $\mathsf{p}(\mathbf{X}, Y)$ over random variables (RVs) $\mathbf{X}$ and Y, where Y is the *label*.For binary classification, i.e., $Y \in \{0, 1\}$, labels y^i are assumed to be modeled by a Bernoulli distribution and are associated to *positive* and *negative* samples $\mathbf{x}^i$.

While nowadays gathering and storing all kinds of data is easier and easier, having all these data perfectly and reliably labeled is unrealistic for several reasons, which makes classical approaches to learning classifiers inapplicable. First, the exponential rate at which

✉ N. Di Mauro
nicola.dimauro@uniba.it

[1] Department of Physics, University of Bari "Aldo Moro", Bari, Italy

[2] National Institute for Nuclear Physics (INFN), Bari Division, Bari, Italy

[3] Department of Computer Science, University of Bari "Aldo Moro", Bari, Italy

data are produced contrasts the time required to produce high quality labels. Moreover, in many fields there are relatively few *labelers* effectively trained to produce reliable labels. Lastly, in many real-world domains it is sometimes unclear what should be considered as a negative sample, or the generation of negative samples is too expensive or just impossible. E.g., in process enactment, one would not waist time, money and resources to build a wrong item just for the purpose of showing how things are not to be done. Thus, the ability to learn predictive models in these scenarios may allow one to exploit the vast amount of data that are produced, saving precious time and resources.

In Positive-Unlabeled (PU) learning (De Comité et al. 1999; Liu et al. 2002), a set $\mathcal{P}$ of positive samples, and a set $\mathcal{U}$ of unlabeled samples—each of which may be positive or negative—are available at training time. So, discriminative information for the negative class must be found in unlabeled data. While a theoretical question may arise about how can one be sure that samples in $\mathcal{U}$ actually include negative ones, not much research in this direction has been done, and going deeper into this matter is beyond the scope of this paper. PU learning shares similarities with semi-supervised learning (du Plessis and Sugiyama 2014), one-class classification (Schölkopf et al. 2001), and outlier detection (Chandola et al. 2009). Differently from semi-supervised learning, no negative samples are available at training time and yet it is required to learn a discriminator between the two classes, in contrast with one-class classification. Additionally, PU learning is in opposition to outlier detection which is usually performed in a transductive way to label unlabeled training data only.

The interest for PU learning is supported by its successful application in several domains, such as document classification (Zhou et al. 2010), graph classification (Zhao et al. 2011), and fake review detection (Li et al. 2014). PU learning approaches can be roughly grouped into *two-staged*—extracting a set of reliable negative samples (RN) from $\mathcal{U}$ and then performing supervised learning—and *single-staged*—taking all samples in $\mathcal{U}$ as negative. The latter is clearly a simplification, because one knows by the very definition of PU-learning that $\mathcal{U}$ may in principle include both positive and negative samples. So, considering all samples in $\mathcal{U}$ as negative is likely to introduce a significant amount of noise. For the former, it becomes crucial to learn a metric that is able to discriminate among classes. However, each application domain needs a specific formulation for such a metric. Hence, ad-hoc algorithmic solutions are often required to cope with different data representations (Zhou et al. 2010; Ienco and Pensa 2016). Few approaches have been proposed to deal with categorical data (Calvo et al. 2007; Ienco and Pensa 2016) in PU learning, due to their being quite challenging because there is no natural distance for them (Ienco et al. 2012).

In this work we present *Generative Positive-Unlabeled* (GPU) learning, a novel two-staged approach to PU learning, introduced in Basile et al. (2017), aimed at being general enough to support very different application domains. In particular, the proposed GPU approach estimates the marginal distribution $\mathsf{p}_{\mathcal{P}}(\mathbf{X}|Y=1)$ of the positive samples in $\mathcal{P}$ via a generative model, and then performs inference on such a distribution to select a set of reliable negative samples from $\mathcal{U}$. The modeled probability density implicitly defines a metric space among samples, and we assume negative ones to be concentrated where positive ones are less likely. Performing inference on the generative model in GPU equals to computing the probability of a particular configuration of the RVs $\mathbf{X}$, which, in turn, can easily be exploited as a measure assessing how reliably an unlabeled sample can be considered as a negative one.

Generative models such as Probabilistic Graphical Models (PGMs) (Koller and Friedman 2009), extensively studied in the literature, offer a powerful formalism to deal with complex probability distributions over continuous, categorical, or even structured data

(Yang et al. 2014). Dealing with a particular domain translates into choosing a suitable PGM from a consolidated research field. As regards the subject of this paper, we exploit a PGM, learned as a density estimator in a certain domain, as a negative sample extractor for partially labeled data. A positive feature of this approach is that PGMs can natively handle categorical data. Since dealing with categorical data might be an issue for several existing PU learning methods, the adoption of PGMs in PGU automatically solves this issue. To stress this advantage, while GPU can deal with different data representations, here we focus specifically on categorical data. As already pointed out in Ienco and Pensa (2016), many PU learning approaches cannot be directly applied for categorical data since they employ metrics not suited for categorical data where there is no standard definition of distance. In (Ienco and Pensa 2016), the authors addressed the problem of classifying data described by categorical attributes proposing a distance-based classification method employing a distance metric learned directly from data. Here we tackle the same problem proposing a PGM based approach that can naturally deal with categorical data.

Differently from Basile et al. (2017), where the approach has been firstly presented, in this paper the GPU approach has been extended in order to deal with the PU learning problem adopting ensembles of PGMs (Antonucci et al. 2013) that generally try to improve the modeling capacity of a single PGM. We firstly present a method to aggregate many PGMs in an ensemble trying to avoid a costly expectation maximization approach. Then we propose two methods to combine the predictions of each model in the ensemble. The main idea is to represent the metric space for categorical data as a PGM that is able to encode the complex patterns arising in the data. We compared GPU, and its ensemble extension EGPU, on real data to several PU learners that have proven to be effective on categorical data. The proposed GPU approach can be applied to continuous data just by using a PGM able to deal with this kind of data such as Gaussian Bayesian networks whose variables are continuous and where all of the CPDs are linear Gaussian (Koller and Friedman 2009).

The paper is organized as follows. The next section provides a brief overview of the literature about PU learning. In Section 3 we introduce and discuss our GPU approach and its extension EGPU, while the experimental setting and the experiment results are presented in Section 4. Conclusions are drawn in Section 5.

2 Related works

PU learning has attracted a great deal of attention in the research on machine learning and data mining. An extensively adopted approach to PU learning is based on a two-staged strategy. First a negative set construction process identifies reliable negative samples from the unlabeled ones, and then traditional classification methods, such as Naive Bayes (NB) or Support Vector Machines (SVM), are directly applied to on the positive and identified negative instances to train predictive models. Alternative methods following this paradigm differ for how they implement these two steps.

Several proposals adopt distance-based approaches to identify negative samples, as the farthest unlabeled ones from positive samples. In Yang et al. (2012), after selecting features statistically related to positive samples, the unlabeled set is partitioned into four sets (reliable/likely/weak negative and likely positive) based on the Euclidean distance. Successively, a multi-level samples learning technique, weighted SVMs, is exploited to build a classifier. The same approach of first identifying, characterizing and discriminating features for positive samples is adopted in Pulce (Ienco and Pensa 2016), where a particular distance function previously designed by the authors is used to determine reliable negative samples; then, distance learning is applied twice—on the positive and reliable negative samples—and the resulting distances are used for k-NN classification.

After theoretically showing that, under suitable conditions, $\mathcal{P}$ and $\mathcal{U}$ provide sufficient information for learning, in Liu et al. (2002) PU learning is posed as a constrained optimization problem. In such a setting, the set of reliable negative samples is selected by using a Naive Bayes (NB) classifier and Expectation Maximization (EM). To the extreme, all the unlabeled samples are treated as negative samples in the NB classifier initially learned and successively used to extract the set of reliable negatives from unlabeled data (Liu et al. 2003). The dataset so obtained is finally exploited to learn a classifier using SVM. The same strategy is exploited in Li and Liu (2005) where a large set of irrelevant samples is generated and added to unlabeled data with the aim of reducing the noise in the data, represented by the presence of positive samples in $\mathcal{U}$. The obtained augmented set, as representative of negatives samples, is exploited, along with positive ones, to compute the parameters of the NB classifier devoted to the reliable identification of negative samples. Finally, an EM-based algorithm is exploited to learn the predictive model.

The Mapping-Convergence algorithm is used to learn a classifier using the positive samples and a set of *strong* negative samples extracted from the unlabeled samples by selecting items having features that rarely appear in the positive samples (Yu et al. 2002). SVM is iteratively applied for classifier building. SVM is also the final classifier learning approach used in (Li and Liu 2003) after an initial step of negative samples extraction carried out with different combinations of Rocchio, k-means clustering and SVMs.

A different policy is the weighted-based approach on unlabeled data exploited in Elkan and Noto (2008). The study shows that a classifier trained on positive and unlabeled samples is able to predict probabilities that differ by only a constant factor from the true conditional probabilities produced by a model trained on fully labeled positive and negative samples, provided that the labeled positive samples are chosen completely at random from all positive samples. This result is used in two different ways: learning from $\mathcal{P}$ versus $\mathcal{U}$ with adjustment of output probabilities finally assigned to unlabeled samples, and learning from $\mathcal{P}$ and $\mathcal{U}$ after double weighting of $\mathcal{U}$. The basic learning algorithm for each method is an SVM with a linear kernel whose outputs are post-processed into calibrated probabilities by fitting a one-dimensional logistic regression function.

Naive Bayes classifier is extensively adopted for categorical data in the four methods proposed in Calvo et al. (2007), namely: (Average) Positive Naive Bayes ((A)PNB), based on Naive Bayes, (Average) Positive TAN ((A)PTAN), and two variants of the Tree Augmented Naive Bayes model (Friedman et al. 1997) able to deal with positive and unlabeled samples. The difference lies in the way the prior probability for the negative class is estimated. For PNB and PTAN such a probability is derived directly from the whole set of unlabeled samples, while for APNB and APTAN the uncertainty is modeled by a Beta distribution.

The above survey shows that many works on PU learning (Liu et al. 2002, 2003; Calvo et al. 2007; Ienco and Pensa 2016) have adopted the text categorization perspective, which is quite peculiar. Indeed, features are intrinsically categorical, there is a huge number of features compared to other settings, the representation of samples is very sparse, and there is a heavy impact of text pre-processing in setting up the classification problem. Others have faced biomedical problems (Elkan and Noto 2008; Yang et al. 2012), where it is typical that databases specify which genes or proteins are related to some specific consequence, but this does not mean that all the others are unrelated to that consequence and, on the contrary, there is a strong interest in identifying which ones actually are (Elkan and Noto 2008).

As previously pointed out, although PU learning shares similarities to outlier detection (Chandola et al. 2009) and to one-class classification (Schölkopf et al. 2001), it differs from these settings in both the goal to pursue and the training set exploited, even when probability density estimation techniques are used as solving strategies (Riahi et al. 2014; Xu and Shelton 2010; Tax and Duin 2004; Hempstalk et al. 2008). Indeed, both methods aim at learning a model that can reject the new incoming samples using positive training data only. They do not require to learn a discriminator between the two classes and, hence, no effort to learn a model for the negative class is made. Intuitively, this type of approach is inferior because it ignores useful information that is present in the unlabeled samples.

As to outlier detection, the usual approach to identify negative samples is considering them as outliers with respect to the positive ones. Many works in this field proceed by learning a generative statistical model from the training data, in the form of some kind of Bayesian network, and then determining outliers based on the likelihood that another individual belongs to that population. Specifically, Riahi et al. (2014) extend unsupervised statistical outlier detection from the case of non-relational data to the case of relational data. Babbar and Chawla (2010) use the network as a background knowledge and derive from it two quantitative rules to uncover outliers and rank the instances based on joint probability distribution in the Bayesian network. The proposed approach also allows one to explain why an instance is an outlier. A critical analysis of distance-based techniques is also provided to show why there is a mismatch between outliers as entities "which are far away from their neighbors" and "real" outliers as identified using Bayesian Networks. Xu and Shelton (2010) perform anomaly detection on intrusion detection systems, based on patterns not conforming to a historic norm, using continuous time Bayesian networks.

Different types of approaches have been developed for one-class classification problems. Some methods use only positive samples to learn the target concept. They estimate the probability density function by fitting a statistical distribution, such as a Gaussian, to the target data, and predicting to be an outlier any instance that exhibits a low probability of appearing, as in Tax and Duin (2004). Other works are based on the application of a density estimator, used to form a reference distribution on positive data, for the generation of artificial data to set up a two-class classification problem, as reported in Hempstalk et al. (2008). To the extreme, some methods get ideas from PU learning approaches to solve the problem, as in Li et al. (2011), where the authors proposed a new PU learning algorithm for one-class classification of remote-sensing data.

3 Generative models for PU learning

In this section we introduce and discuss our GPU approach and its ensemble extension EGPU.

3.1 Notation

Let RVs be denoted by upper-case letters, e.g., X, and their values as the corresponding lower-case letters, e.g., $x \sim X$. We denote sets of RVs as $\mathbf{X}$, and their combined values as $\mathbf{x}$. For a set of RVs $\mathbf{X}$ we denote with $\mathbf{X}_{\setminus i}$ the set $\mathbf{X}$ deprived of X_i, and with $\mathbf{X}_{|\mathbf{Y}}$ the restriction of $\mathbf{X}$ to $\mathbf{Y} \subseteq \mathbf{X}$ (the same applies to assignments $\mathbf{x}$). W.l.o.g., here we deal with *discrete* RVs over finite domains. When we refer to a joint probability distribution $\mathsf{p}(\mathbf{X})$ over RVs $\mathbf{X}$, we are either considering the joint probability density function for continuous RVs, or the probability mass function for discrete RVs, or a hybrid combination of both in hybrid domains (Koller and Friedman 2009; Yang et al. 2014).

To denote a finite domain of a discrete RV X_j we introduce the following notation $\mathsf{Val}(X_j) = \{x_j^k\}_{k=1}^{K_j}$. If $\mathcal{D}$ is a set of samples over RVs $\mathbf{X}$, we indicate with $\mathsf{p}_{\mathcal{D}}(\mathbf{X})$ the real (unknown) probability distribution that generated the data, while if $\mathcal{M}$ indicates a generative model, $\mathsf{p}_{\mathcal{M}}(\mathbf{X})$ refers to the probability distribution estimated by such a model on finite sample sets. Disambiguation is provided by context. Generally one wants the estimate $\mathsf{p}_{\mathcal{M}}(\mathbf{X})$ to be as close as possible to $\mathsf{p}_{\mathcal{D}}(\mathbf{X})$. A common way to measure this closeness is via the *log-likelihood* function (Koller and Friedman 2009), or one of its variants, defined as:

$$\ell_{\mathcal{D}}(\mathcal{M}) = \sum_{\mathbf{x}^i \in \mathcal{D}} \log \mathsf{p}_{\mathcal{M}}(\mathbf{x}^i).$$

3.2 PU learning problem

In the classical PU learning setting, a training set $\mathcal{D} = \mathcal{P} \cup \mathcal{U}$ i.i.d. from $\mathsf{p}(\mathbf{X}, Y)$ is given, comprising $m_{\mathcal{P}} = |\mathcal{P}|$ positive samples and $m_{\mathcal{U}} = |\mathcal{U}|$ unlabeled samples. Samples in $\mathcal{P}$ are provided with a known positive class label, i.e.,

$$\mathcal{P} = \{(\mathbf{x}^i, 1)\}_{i=1}^{m_{\mathcal{P}}} \sim \mathsf{p}_{\mathcal{P}}(\mathbf{X}|Y = 1).$$

On the contrary, class information—labels—is not provided for samples in $\mathcal{U}$, i.e.,

$$\mathcal{U} = \{\mathbf{x}^i\}_{i=1}^{m_{\mathcal{U}}} \sim \mathsf{p}_{\mathcal{U}}(\mathbf{X}),$$

where $\mathsf{p}_{\mathcal{U}}(\mathbf{X})$ is the marginal probability distribution w.r.t. $\mathsf{p}_{\mathcal{U}}(\mathbf{X}, Y)$.

Let $\mathcal{D}_0$ (resp. $\mathcal{D}_1$) denote the subset of all negative (resp. positive) samples in $\mathcal{D}$. The aim of PU learning is to build a discriminator model $f : \mathbf{X} \rightarrow Y$ from $\mathcal{D}$ in order to make accurate predictions about the labels of unseen test data samples. Following Elkan and Noto (2008), we assume that samples in $\mathcal{P}$ are *selected completely at random* from all positive samples in $\mathcal{D}$, i.e.,

$$\mathsf{p}_{\mathcal{P}}(\mathbf{X}|Y = 1) = \mathsf{p}_{\mathcal{D}}(\mathbf{X}|Y = 1).$$

3.3 Generative models for PU learning

Our proposed approach, Generative PU learning (GPU), falls in the category of two-staged methods for PU learning. First it extracts a set of reliable negative samples $\mathcal{N}$ from $\mathcal{U}$, then $\mathcal{N}$ is employed to perform supervised learning. In the following we detail our contribution to the first step, discussing possible approaches for the second one.

As usual in statistical machine learning, we assume $\mathsf{p}_{\mathcal{D}}$ to be modeled as a mixture of probability distributions for the positive and negative class, i.e.,

$$\mathsf{p}_{\mathcal{D}} = \sum_{y \in \{0,1\}} \mathsf{p}(Y = y)\mathsf{p}(\mathbf{X}|Y = y) = w_{\mathcal{D}_0}\mathsf{p}_{\mathcal{D}_0}(\mathbf{X}) + w_{\mathcal{D}_1}\mathsf{p}_{\mathcal{D}_1}(\mathbf{X}),$$

where $w_{\mathcal{D}_0}$ (resp. $w_{\mathcal{D}_1}$) denotes the marginal probabilities of the label w.r.t. the negative (resp. positive) class and $\mathsf{p}_{\mathcal{D}_0}(\mathbf{X})$ (resp. $\mathsf{p}_{\mathcal{D}_1}(\mathbf{X})$) denotes the conditional probability of a sample w.r.t. the negative (resp. positive) class.

As it is common practice in PU learning (Elkan and Noto 2008), we assume that the positive samples in $\mathcal{P}$ are highly representative for all positive samples in $\mathcal{D}_1$, which in turn implies that $\mathsf{p}_{\mathcal{P}}$ is close to $\mathsf{p}_{\mathcal{D}_1}$. As an additional assumption, we consider the distribution generating $\mathcal{D}_0$ and $\mathcal{D}_1$ to be fairly *distinguishable* (Balasubramanian 2005). That is, we assume that high density regions of $\mathsf{p}_{\mathcal{D}_0}$ correspond to low density regions of $\mathsf{p}_{\mathcal{D}_1}$ and *vice versa*. While this assumption might seem too strict for real data, in practice, it is commonly adopted when performing unsupervised clustering (e.g., Gaussian densities must be separable in EM and K-means). As future research, we plan to investigate how to adapt GPU learning to more complex learning settings.

As already discussed in Ienco et al. (2012), distance metrics for categorical data are very difficult to assess, since it is impossible to quantify the difference between two values of a multivariate variable. The assumption adopted by Ienco et al. (2012) is that the distance between two values of a categorical attribute can be determined by the way in which the values of the other attributes are distributed in the data. In particular, the metric is data oriented and not fixed apriori. Here, the idea is to represent the metric space for categorical data as a PGM that is able to encode the complex patterns arising in the data.

The assumption that we make is that the distributions $\mathsf{p}_{\mathcal{P}}$ and $\mathsf{p}_{\mathcal{N}}$ generating, respectively, the positive and the negative samples are different. In particular, we assume that they may belong to the same family but having different modes. Having learned the density $\mathsf{p}_{\mathcal{P}}$ from the positive samples, we can label as negative the samples $\mathbf{x}$ in $\mathcal{U}$ having a low joint probability $\mathsf{p}_{\mathcal{P}}(\mathbf{x})$. This method should work well when there is no inconsistency between the positive and negative samples, i.e., any sample is never true labeled as both positive and negative.

The high level idea behind our approach is the following. By correctly modeling the probability distribution of positive samples over RVs $\mathbf{X}$, one can model discriminative patterns among samples in the form of *probabilistic dependencies* among their RVs. If this is done accurately, then a metric space is implicitly defined, associating low probability regions to negative samples and high probability ones to positive samples. Similar ideas have also been successfully investigated in applications for anomalous or outlier training samples (Xu and Shelton 2010; Riahi et al. 2014).

Algorithm 1 LearnGPU($\mathcal{P}, \mathcal{U}$).

```
1: Input: a set $\mathcal{P} = \{(\mathbf{x}^i, 1)\}_{i=1}^{m_\mathcal{P}}$ of positive samples, and a set $\mathcal{U} = \{\mathbf{x}^i\}_{i=1}^{m_\mathcal{U}}$ of unlabeled samples over RVs $\mathbf{X} \cup \{Y\}$, with $\mathsf{Val}(Y) = \{0, 1\}$.
2: Output: a trained discriminative model leaned on positive samples $\mathcal{P}$ and reliable negative samples $\mathcal{N}$ extracted from $\mathcal{U}$
3: $\mathcal{G} \leftarrow$ learnGenerativeModel($\mathcal{P}, \mathbf{X}$)    ▷ learn a generative model $\mathcal{G}$ from $\mathcal{P}$ encoding $\mathsf{p}_\mathcal{G}(\mathbf{X})$
4: $\mathcal{L} \leftarrow \{\log \mathsf{p}_\mathcal{G}(\mathbf{x}^i) | \mathbf{x}^i \in \mathcal{U}\}$
5: $\mathcal{N} \leftarrow$ reliableNegativeSamples($\mathcal{L}, \mathcal{P}, \mathcal{U}$)    ▷ first $n$ worst scored samples in $\mathcal{S}$
6: $f \leftarrow$ fitClassifier($\mathcal{P}, \mathcal{N}$)
7: return $f$
```

Algorithm 1 illustrates the general schema of our proposed GPU approach. In order to estimate $\mathsf{p}_{\mathcal{P}}$ we fit a generative model, $\mathcal{G}$, over the RVs $\mathbf{X}$ of the positive training set (line 3). The estimator can be obtained with any generative model, such as Bayes Networks (BNs) and Markov Networks (MNs), able to compute the joint probability $\mathsf{p}_{\mathcal{G}}(\mathbf{x})$ for a given state $\mathbf{x}$ of the RVs $\mathbf{X}$. We discuss the choice of such an estimator in Section 3.4. After fitting the estimator to the positive samples, we can use it to compute the joint probability for each state of the RVs appearing in the set of unlabeled samples in $\mathcal{U}$—we derive an empirical estimation of the less dense (i.e., less likely) regions by computing the point-wise log-likelihood of $\mathcal{G}$ over the samples in $\mathcal{U}$. Being $\mathcal{G}$ a generative model, this probability indicates the likelihood that a sample is generated by the model. The assumption is that samples having a low probability may be labeled as negative ones. Based on this information we build a set of reliable negative samples, denoted by $\mathcal{N}$ (line 5), to be exploited in the second stage of PU learning. Such a schema is general enough to be adapted to different data domains by leveraging different density estimators. Moreover, by specifying algorithmic variants to build $\mathcal{N}$ and the final discriminator f, one can improve its robustness and accuracy. We discuss such extensions in the following sections.

3.4 Bayesian Networks and mixtures of trees

A question arises on which generative model to employ. The main challenge in learning generative models is balancing the *representation expressiveness* of the learned models against the *cost of learning* and *performing inference* on them.

Probabilistic Graphical Models (PGMs), like Bayesian Networks (BNs) and Markov Networks (MNs), are able to model highly complex probability distributions and have been successfully employed as density estimators. However, exact inference with them is generally *intractable* (Cooper and Herskovits 1990). Since our GPU learning schema only requires the computation of complete evidence queries, employing BNs in GPU would lead to tractable inference to build $\mathcal{N}$.

Nevertheless, learning a complex model could still pose a challenge on very large datasets. Guaranteeing exact and tractable inference, a series of *tractable probabilistic models* (TPMs) have been recently proposed: either by restricting the expressiveness of PGMs by bounding their treewidth, or by exploiting local structures in a distribution. The limited expressive capabilities of TPMs, like mixtures of Bayesian trees (MTs) (Meila and Jordan 2000) and Cutset Networks (Di Mauro et al. 2017), or their ability to compile a high treewidth network into a deep probabilistic architecture, like Sum-Product Networks (Vergari et al. 2015), allow for more efficient learning schemes.

In this work we evaluate GPU by exploiting both BNs and MTs to investigate how the model expressiveness affects the estimation of $\mathsf{p}_{\mathcal{P}}$ and therefore ultimately the accuracy of the learned discriminator (see Section 4). In the following we briefly review both models.

BNs are a PGM encoding a probability distribution by means of a directed acyclic graph and a set of weights, where nodes correspond to RVs and edges to dependencies among RVs. Given a set of n RVs $\mathbf{X}$, for each variable $X_i \in \mathbf{X}$, Pa_i denotes the set of parents of node X_i in the DAG. The structure of the BN $\mathcal{G}$, induces a factorization of the joint distribution into local factors, that is:

$$\mathsf{p}_{\mathcal{G}}(\mathbf{X}) = \prod_{i=1}^{n} p(X_i|\mathsf{Pa}_i).$$

Learning a BN corresponds to learning both the structure and the conditional probability distribution corresponding to each local factor from the data. Classical structure learning algorithms search in the space of BNs guided by a scoring function. On the other hand, parameter learning is obtained by maximum likelihood estimation.

Concerning mixtures of generative models, a very competitive density estimation algorithm is MT (Meila and Jordan 2000). MT learns a mixture model $\mathcal{M}$ whose distribution factorizes according to

$$\mathsf{p}_{\mathcal{M}}(\mathbf{X}) = \sum_{i=1}^{k} \lambda_i \mathsf{p}_{\mathcal{T}_i}(\mathbf{X}),$$

where the distributions $\mathsf{p}_{\mathcal{T}_i}$, learned using the Chow-Liu algorithm (Chow and Liu 1968), are the mixture components and $\lambda_i \geq 0$, with $\sum_{i=1}^{k} \lambda_i = 1$ are their coefficients. The Chow-Liu algorithm learns BNs with lower treewidth—nodes have at most one parent in the network—thus leading to efficient learning and inference time. In Meila and Jordan (2000) the best components and weights are found as (local) likelihood maxima by using EM, with k fixed in advance.

3.5 Reliable negative sample elicitation

After learning a generative model $\mathcal{G}$, the density estimation information provided by $\mathcal{G}$ can be exploited in several ways. The most straightforward one would be to impose a threshold hyperparameter θ such that each sample in $\mathcal{U}$ whose loglikelihood $\log \mathsf{p}_{\mathcal{G}}$ falls under θ can be added to $\mathcal{N}$. However, determining the best value for θ would require to perform additional hyperparameter optimization. To alleviate this issue we propose to implicitly compute it by building $\mathcal{N}$ to comprise the $m_{\mathcal{P}}$ samples in $\mathcal{U}$ with the lowest log-likelihood score according to $\mathcal{G}$. In such a way we ensure that the resulting labeled set $\mathcal{P} \cup \mathcal{N}$ is balanced w.r.t. the positive and negative class. The risk of including positive samples into $\mathcal{P} \cup \mathcal{N}$ can be mitigated by adopting a robust classifier in the following supervised step, whose generalization ability on test data may also additionally benefit from the regularization capability of mis-specifying some sample labels. Lastly, we note how density information in the form of the finite set log-likelihoods can be directly incorporated into the construction of the classifier over $\mathcal{P} \cup \mathcal{N}$.

While we employ the likelihoods to select the most reliable negative samples from $\mathcal{U}$, they could also be used to select the most reliable positive samples instead. Adopting such a strategy, GPU can be turned into an iterative schema in which at each iteration $\mathcal{P}$ is augmented with the samples belonging to the most dense regions. After a stopping criterion is met, $\mathcal{N}$ can be built by collecting all the samples in $\mathcal{U}$ not added to $\mathcal{P}$.

3.6 Mixtures of generative models

To mitigate issues like the scarce accuracy of a single model and their tendency to overfit, PGMs could be employed as the components of a mixture of the form:

$$\mathsf{p}(\mathbf{X}) = \sum_{i=1}^{k} \lambda_i \mathsf{p}_i(\mathbf{X}),$$

being $\lambda_i \geq 0 : \sum_{i=1}^{k} \lambda_i = 1$ the mixture coefficients, and $\mathsf{p}_i(\mathbf{X})$ elementary PGM densities.

The first approach to learn such a mixture could be to employ EM to alternatively learn both the weights and the mixture components. However, with this approach, the learning time complexity grows at least of a factor of kt being t the number of iterations of EM. All the classic issues about convergence and instability of EM make this approach unpractical.

A more efficient method to learn the mixtures, could be to adopt bagging (Hastie et al. 2009) as a cheap and yet more effective way to only increase time complexity by a factor k. For bagged PGMs, mixture coefficients could be set equally probable and the mixture components can be learned independently on different bootstrapped data samples.

In particular, we draw k bootstrapped samples $\mathcal{P}_i$ from the dataset $\mathcal{P}$, sampling $|\mathcal{P}|$ samples with replacements, and on each of those we fit a generative model, thus leading to k models $\mathcal{G}_i$. The resulting bagged models could correspond to a weighted sum of all the learned models $\mathcal{G}_i$.

Differently from the explained elicitation process for a single model, now we have to choose how the models in the ensemble have to be combined. The first choice—*averaging*—is to consider the same elicitation process adopted for a single model with the likelihoods computed by the bagged model $\mathcal{G}$ as

$$\mathsf{p}_{\mathcal{G}}(\mathbf{X}) = \sum_{i=1}^{k} \lambda_i \mathsf{p}_{\mathcal{G}_i}(\mathbf{X}),$$

with uniform mixture coefficients, i.e., $\lambda_i = 1/k$.

As an alternative we can consider a *voting* (or a ranking) approach—as a combination approach, instead of considering the likelihood we can consider the votes each sample obtains from the models.

In particular, each model $\mathcal{G}_i$ is used to compute the likelihood $\mathsf{p}_{\mathcal{G}_i}(\mathbf{x})$ for each sample $\mathbf{x} \in \mathcal{U}$. Then, sorting the samples in $\mathcal{U}$ according to their likelihoods we obtain, for each model $\mathcal{G}_i$, a ranking

$$\pi_i : \{1, \dots, m_{\mathcal{U}}\} \rightarrow \{1, \dots, m_{\mathcal{U}}\},$$

i.e., a permutation of $\{1, \dots, m_{\mathcal{U}}\}$ denoted as a bijection mapping samples to ranks. Finally, the ranking of a sample $\mathbf{x}_j \in \mathcal{U}$ has been computed as the average ranking:

$$\pi(\mathbf{x}_j) = \frac{\sum_{i=1}^{k} \pi_i(j)}{k}.$$

Now, the elicitation process is performed considering averaged rankings instead of the averaged likelihoods.

3.7 Supervised classification step

In principle, every supervised classifier can be employed in GPU after the set $\mathcal{N}$ is constructed. In the empirical evaluation we provide in Section 4 we adopt the regular implementation of Support Vector Machines (SVMs) included in scikit-learn[1]. While the SVM works well in presence of noise like in the case of a single model, it overfits the training samples in the case of ensemble due to the improved model capacity. In this case we opted for a more basic learning approach like *Logistic Regression* (or *MaxEnt*) classifier with L1 regularization with primal formulation using the LIBLINEAR (Fan et al. 2008) library solver as implemented in scikit-learn.

Nevertheless, we now discuss other interesting variants for GPU learning. First, if one builds $\mathcal{N}$ to be unbalanced w.r.t. $\mathcal{P}$, it would be possible to adopt the more robust variant of *biased SVMs* (Hoi et al. 2004). Alternatively, if one focuses on iteratively augmenting the set $\mathcal{P}$ only with GPU, then 1-class SVMs (Schölkopf et al. 2001) could be employed to derive a max-margin hypersphere for the positive class. Additionally, the likelihoods associated to samples in $\mathcal{U}$ could be interpreted as sample confidence weights. Approaches like that of Zhou et al. (2012) could be adopted to learn a *weighted classifier* over $\mathcal{P} \cup \mathcal{U}$ without the need to build $\mathcal{N}$ either. Lastly, our probabilistic generative approach for the first stage can be plugged in an *unsupervised clustering* approach for the second stage, as done with the EM algorithm in Liu et al. (2002). A principled end-to-end probabilistic formulation would allow estimating both $\mathbf{p}_{\mathcal{D}_0}$ and $\mathbf{p}_{\mathcal{D}_1}$ iteratively and jointly.

[1]http://scikit-learn.org/

As regards the computational complexity of our approach it greatly depends on that of the adopted PGM used to learn the density distribution. Learning the structure and the parameters of a PGM strongly relies on the inference procedure. It is well known that exact inference in PGMs is NP-hard if no assumption is considered about the structure of the underlying graphical model (Cooper and Herskovits 1990), and still NP-hard even in the approximate case (Roth 1996). However, inference can be tractable in models in which the underlying graph has low treewidth or in recently proposed tractable probabilistic models like Sum-Product Networks (Vergari et al. 2015) and Cutset Networks (Di Mauro et al. 2017). Hence, the scalability of our proposed approach can be improved by integrating tractable models as density estimators.

4 Experiments

In this section we empirically evaluate the proposed GPU approach, and its ensemble extension EGPU, applying them to categorical data. We are interested in this kind of data because they are challenging for classical metric based approaches. Since there is no general consensus on how to build a metric to evaluate categorical data, ad-hoc solutions have been adopted on a domain-wise perspective (Ienco et al. 2012), and only recently PU learning schemes have been devised for it (Ienco and Pensa 2016). On the other hand, PGMs have been extensively investigated for categorical data and estimating a probability distribution over discrete RVs is a consolidated practice for extracting new representations in a domain-agnostic unsupervised way (Hinton and Salakhutdinov 2006; Bengio et al. 2012; Vergari et al. 2019). As stated in the previous sections, adapting GPU to other domains reduces to selecting an appropriate generative toolbox from the probabilistic model literature. Specifically, we aim at answering the following research questions: **Q1)** how does GPU and EGPU compare to state-of-the-art PU learning approaches? **Q2)** how does the quantity of available positive samples affect GPU and learning? **Q3)** how much does the choice of a generative model in estimating $\mathsf{p}_{\mathcal{P}}$ affect GPU's performance?

4.1 Experimental setting

We compared the proposed GPU and EGPU approaches to other state-of-the-art PU learning methods for categorical data. We took 10 datasets publicly available on the UCI machine learning repository[2], derived 3 experimental settings for each, and ran 10-fold cross validations exactly as in Ienco and Pensa (2016)[3]. The three settings were generated by putting in $\mathcal{P}$ 30%, 40%, and 50% labeled samples of the positive class respectively, and in $\mathcal{U}$ the remaining positive samples plus all the negative ones. When the dataset does not describe a binary classification problem, the two heavily populated classes were considered. In our experiments, all numerical attributes were discretized into 10 equal-width bins. Detailed dataset statistics are reported in Table 1.

We evaluate GPU by employing either BNs ($\mathsf{GPU}^{\mathsf{BN}}$) or MTs ($\mathsf{GPU}^{\mathsf{MT}}$) as generative models (see Section 3.4). BNs are learnt using the R package bnlearn[4] (release 4.1.1). To learn their structure we employed the simple score-based hill-climbing algorithm. In

[2]http://archive.ics.uci.edu/ml/

[3]The datasets and settings used in Ienco and Pensa (2016) were kindly provided by Dino Ienco.

[4]http://www.bnlearn.com/.

Table 1 Dataset statistics. #pos and #unl denote the number of positive and unlabeled samples respectively

dataset	#attributes	% pos						#test
		30		40		50		
		#pos	#unl	#pos	#unl	#pos	#unl	
audiology	69	15	79	20	74	26	68	11
breast-cancer	9	54	203	72	185	91	166	29
chess	36	451	2425	601	2275	751	2125	320
dermatology	34	30	136	40	126	50	116	19
hepatitis	19	9	130	12	127	15	124	16
lymph	18	17	111	22	106	28	100	14
nursery	8	1166	6562	1555	6173	1944	5784	859
pima	8	135	556	180	511	225	466	77
soybean	35	25	140	33	132	42	123	18
vote	16	72	319	96	295	120	271	44

#test denotes the number of positive and negative examples in the testing set

order to avoid overfitting of the network to the positive samples, the following $K2$ scoring function (Cooper and Herskovits 1992) was adopted[5]:

$$\mathsf{score}_{K2}(\mathcal{G} : \mathcal{P}) = \log p(\mathcal{G}) + \sum_{i=1}^{n} \sum_{j=1}^{q_i} \left(\log \left(\frac{(r_i - 1)!}{(N_{ij} + r_i - 1)!} \right) + \sum_{k=1}^{r_i} \log(N_{ijk}!) \right),$$

where $p(\mathcal{G})$ represents the prior probability of the network G over the n RVs X_i, r_i is the number of states of variable X_i, q_i is the number of possible configurations of the parent set Pa_i, N_{ijk} is the number of instances in the data where variable X_i takes value x_{ik} and the set of variables Pa_i takes value w_{ij}, N_{ij} is the number of instances in the data where the variables in Pa_i take their j-th configuration w_{ij}. Concerning parameter estimation, we set the imaginary sample size to 1. MTs are learnt using the Libra (Lowd and Rooshenas 2015) toolkit[6] (version 1.1.2). We imposed the number of components to be 10.

As the classifier for the supervised second stage, we adopt SVMs[7] with an RBF kernel as implemented in scikit-learn[8]. The penalty parameter C and the kernel coefficient γ have been optimized with a cross validation on the following grid $C \in \{0.001, 0.01, 0.1, 1, 10, 100, 1000\}$ and $\gamma \in \{0.001, 0.01, 0.1, 1, 10, 100, 1000\}$.

For the ensemble learning case, we evaluate EGPU by employing BNs as generative models, and a Logistic Regression, as implemented in scikit-learn, as classifier for the supervised task.

We compared GPU with Positive Naive Bayes (PNB), Average Positive Naive Bayes (APNB), Positive TAN (PTAN), Average Positive TAN (APTAN) (Calvo et al. 2007) and

[5]The same set of experiments has been conducted using the likelihood as scoring function, leading to overfitted models with an overall result worst than that obtained using the $K2$ score.

[6]http://libra.cs.uoregon.edu/.

[7]For this stage only, categorical data is one-hot encoded.

[8]http://scikit-learn.org/.

Pulce (Ienco and Pensa 2016) with $k = 7$ for the k-NN classifier. See Section 2 for a description of these methods.

Performance on the test set was evaluated using the F1-score, defined as $F = 2PR/(P+R) = 2tp/(2tp + fp + fn)$, where P and R are, respectively, the precision and the recall obtained by the algorithm, and tp, fp and fn are, respectively, the true positive, false positive and false negative samples. Since the number of positive samples is much larger than that of negative ones, as in Ienco and Pensa (2016) we directed the computation of P, R, and F1-score to the negative samples, differently from their classical setting, i.e., as $F = 2tn/(2tn + fp + fn)$. In particular, since no information for the negative class is provided, correctly predicting negative samples should be somehow harder than focusing on the positive counterparts.

Source code, in Python and R, of the proposed approach and scripts to reproduce the results are available at https://github.com/nicoladimauro/GPU.

4.2 Results and discussion

Overall results for the GPU approach are reported in Tables 2 and 3. We may note that PTAN and APTAN never won against the other approaches, while the two GPU approaches won 73.3% of the times (53.3% of the times GPU$^{\text{BN}}$ alone), and each GPU approach won more times than any competitor (GPU$^{\text{BN}}$ more than doubled the number of wins of each competitor). The worst-performing dataset for GPU approaches, and the only one where they perform neatly worse than all other competitors, is ‘hepatitis’. This may indicate that for such a dataset the distributions of the negative and positive class are hard to estimate as very different densities. Concerning question **Q1**, therefore, we can say that both GPU$^{\text{BN}}$ and GPU$^{\text{MT}}$ are competitive to the current state-of-the-art for categorical data. On datasets on which GPU$^{\text{BN}}$ does not win in all settings, it still performs comparably or better on settings with larger $\mathcal{P}$ sets. Overall, increasing the size of $\mathcal{P}$ improves the models’ accuracy in a consistent way. At the same time, on datasets where both GPU approaches are competitive, they improve over other methods even with only 30% positive samples available (**Q2**). Lastly, we observe that while GPU$^{\text{BN}}$ generally outperforms GPU$^{\text{MT}}$, the latter is still comparable to Pulce (see average ranks, Table 2) and overall more accurate than all other methods. To answer question **Q3**, we can state that the greater expressiveness of BNs, allowing better modeling of the probability distribution of the positive class, is fairly relevant for achieving better performances. Nevertheless, note that for both GPU$^{\text{BN}}$ and GPU$^{\text{MT}}$ we employed out-of-the-box PGMs and did not invest too much time optimizing the hyperparameters for their structure and weight learning algorithms. It is left for future work to explore how increasing a model complexity can degrade its performance, that is when too accurate probability distribution estimates can lead to overfitting.

As already said, in the reliable negative sample elicitation phase, the number of negative samples to be included in the set $\mathcal{N}$ was set to the same number of positive samples available in $\mathcal{P}$, i.e., $|\mathcal{N}| = |\mathcal{P}|$. However, the generated set $\mathcal{N}$ may contain some true positive samples that have been incorrectly predicted as negative ones. In order to quantify the accuracy of the proposed approach, Table 4 reports, for each dataset, the number of errors occurred in the negative elicitation step, when GPU$^{\text{BN}}$ has been used as a density estimator. As we can see, the percentage of errors decreases as the percentage of positive samples in $\mathcal{P}$ increases. For datasets like ‘breast-cancer’, the number of errors reaches 50% thus confirming the low accuracy in terms of F1-score obtained on this dataset. Anyway, also other competitors are not able to properly separate positive and negative samples on this dataset.

Table 2 F1-score results over the 30 datasets, comparing $\mathrm{GPU}^{\mathrm{BN}}_{\mathrm{SVM}}$ and $\mathrm{GPU}^{\mathrm{MT}}_{\mathrm{SVM}}$ against the competitors Pulce, PNB, APNB, PTAN, APTAN

dataset	%	$\mathrm{GPU}^{\mathrm{BN}}$	$\mathrm{GPU}^{\mathrm{MT}}$	Pulce	PNB	APNB	PTAN	APTAN
audiology	30	0.839	0.902	0.745	0.68	0.7	0.66	0.66
audiology	40	0.879	0.804	0.846	0.75	0.74	0.71	0.66
audiology	50	0.980	0.991	0.899	0.80	0.80	0.78	0.71
breast-cancer	30	0.475	0.450	0.534	0.40	0.39	0.43	0.43
breast-cancer	40	0.483	0.513	0.438	0.42	0.40	0.43	0.45
breast-cancer	50	0.517	0.535	0.443	0.42	0.41	0.44	0.44
chess	30	0.689	0.663	0.696	0.58	0.64	0.59	0.64
chess	40	0.691	0.665	0.688	0.58	0.64	0.60	0.64
chess	50	0.773	0.650	0.655	0.58	0.64	0.60	0.64
dermatology	30	1.000	0.834	0.992	0.57	0.57	0.57	0.56
dermatology	40	1.000	0.836	0.992	0.57	0.58	0.57	0.57
dermatology	50	0.992	0.951	0.992	0.59	0.60	0.57	0.58
hepatitis	30	0.822	0.665	0.843	0.87	0.87	0.85	0.86
hepatitis	40	0.778	0.654	0.873	0.88	0.88	0.85	0.85
hepatitis	50	0.764	0.742	0.855	0.88	0.88	0.86	0.85
lymph	30	0.827	0.782	0.851	0.84	0.85	0.79	0.84
lymph	40	0.825	0.795	0.827	0.84	0.83	0.79	0.81
lymph	50	0.824	0.835	0.814	0.86	0.87	0.81	0.82
nursery	30	0.809	0.761	0.739	0.65	0.65	0.56	0.50
nursery	40	1.000	0.762	0.773	0.69	0.69	0.61	0.56
nursery	50	0.960	0.779	0.807	0.69	0.70	0.74	0.44
pima	30	0.588	0.576	0.532	0.49	0.50	0.50	0.50
pima	40	0.568	0.593	0.547	0.49	0.50	0.50	0.51
pima	50	0.609	0.605	0.528	0.49	0.51	0.50	0.52
soybean	30	0.893	0.766	0.738	0.81	0.86	0.80	0.81
soybean	40	0.883	0.852	0.767	0.86	0.86	0.84	0.83
soybean	50	0.890	0.923	0.823	0.92	0.92	0.88	0.86
vote	30	0.826	0.799	0.679	0.62	0.62	0.56	0.55
vote	40	0.850	0.790	0.800	0.71	0.71	0.58	0.54
vote	50	0.844	0.829	0.829	0.77	0.77	0.61	0.56
# wins		16	6	3	5	6	0	0
Avg. F1-score		0.796	0.743	0.751	0.677	0.686	0.653	0.643
30%		0.777	0.720	0.735	0.651	0.665	0.631	0.635
40%		0.796	0.727	0.755	0.679	0.683	0.648	0.642
50%		0.815	0.784	0.764	0.700	0.710	0.679	0.652
Avg. ranking		2.16	3.23	3.2	4.57	3.95	5.47	5.42

The second column indicates the percentage of positive samples in $\mathcal{P}$

Table 3 Number of wins/ties among all the methods, the average of the number of wins for each method and its ranking in parenthesis

	GPU$^{\mathsf{BN}}$	GPU$^{\mathsf{MT}}$	Pulce	PNB	APNB	PTAN	APTAN	avg.
GPU$^{\mathsf{BN}}$	–	24/0	23/1	23/0	23/0	27/0	25/1	24.17 (1)
GPU$^{\mathsf{MT}}$	6/0	–-	13/0	22/0	22/0	25/0	24/0	18.67 (3)
Pulce	6/0	17/0	–	22/0	22/0	25/0	24/0	19.33 (2)
PNB	7/0	8/0	8/0	–	5/12	18/2	18/3	10.67 (5)
APNB	7/0	8/0	8/0	13/12	–	23/3	21/4	13.33 (4)
PTAN	3/0	5/0	5/0	10/2	4/3	–	12/6	6.50 (7)
APTAN	4/0	6/0	6/0	9/3	5/4	12/6	–	7.00 (6)

Another experiment has been done by varying the number of negative samples in $\mathcal{N}$ as a percentage of the number of samples in $\mathcal{P}$, i.e., by setting $|\mathcal{N}| = \alpha|\mathcal{P}|$. Table 5 reports the results adopting GPU$^{\mathsf{BN}}$ as a density estimator for $\alpha \in \{0.6, 0.7, 0.8, 0.9, 1.0, 1.1, 1.2, 1.3, 1.4\}$. It is possible to see that for $\alpha = 1.2$ the number of wins of the proposed approach over Pulce increases to 25.

As regards the ensemble approach, Table 6 reports the results when comparing EGPU with average (EGPU$^{10}_{\mathsf{avr}}$ and EGPU$^{20}_{\mathsf{avr}}$) with 10 and 20 components, EGPU with ranking (EGPU$^{10}_{\mathsf{rnk}}$ and EGPU$^{20}_{\mathsf{rnk}}$) with 10 and 20 components, to the best competitor Pulce. GPU$_{\mathsf{LR}}$ indicated a single BN model using logistic regression as classifier. The symbol • denotes an improvement over the base classifier GPU$_{\mathsf{LR}}$.

First of all we can note that the ensemble approaches always win against the base classifier (GPU$^{\mathsf{BN}}$ and GPU$_{\mathsf{LR}}$)—EGPU$^{20}_{\mathsf{avr}}$ seems to be the best performing. Furthermore there is an improvement against the best competitor Pulce w.r.t. the results obtained without ensemble (GPU$^{\mathsf{BN}}$). The improvement of the ensemble is evident for each dataset except for the `hepatitis` dataset where there are few positive samples in the training set making difficult to learn a good density estimator. The same effect is evident on the `lymph` dataset. On the contrary when the dataset contains a lot of samples the ensemble approach seems to perform better than the single model.

Table 4 Number of positive samples incorrectly predicted as negative ones in the negative sample elicitation phase

dataset	30%		40%		50%	
audiology	0.17	(2.6/15)	0.11	(2.3/20)	0.11	(2.8/26)
breast-cancer	0.48	(25.7/54)	0.46	(33.3/72)	0.43	(38.8/91)
chess	0.33	(148.6/451)	0.27	(162.4/601)	0.21	(159.7/751)
dermatology	0.00	(0.0/30)	0.00	(0.0/40)	0.00	(0.0/50)
hepatitis	0.19	(1.7/9)	0.00	(0.0/12)	0.11	(1.7/15)
lymph	0.16	(2.8/17)	0.15	(3.4/22)	0.14	(3.8/28)
nursery	0.00	(0.0/1166)	0.00	(0.0/1555)	0.03	(50.4/1944)
pima	0.33	(44.4/135)	0.30	(53.6/180)	0.28	(64.0/225)
soybean	0.14	(3.4/25)	0.15	(4.9/33)	0.10	(4.4/42)
vote	0.30	(22.9/72)	0.22	(21.0/96)	0.23	(27.9/120)

In parenthesis the average number of errors for each fold over the cardinality of both $\mathcal{P}$ and $\mathcal{N}$

Table 5 Detailed F1-score results over the 30 samples obtained by GPU^{BN} by varying the percentage of the reliable negative samples added to $\mathcal{N}$

dataset		GPU^{BN}							Pulce
		Negative percentage							
	%	70%	80%	90%	100%	110%	120%	130%	
audiology	30	0.832	0.812	0.834	0.839	0.627	0.857	0.857	0.745
audiology	40	0.949	0.940	0.958	0.879	0.938	0.940	0.923	0.846
audiology	50	0.991	0.989	0.971	0.980	0.989	0.966	0.957	0.899
breast-cancer	30	0.422	0.426	0.410	0.475	0.485	0.497	0.464	0.534
breast-cancer	40	0.416	0.409	0.470	0.483	0.494	0.492	0.500	0.438
breast-cancer	50	0.486	0.454	0.508	0.517	0.522	0.497	0.481	0.443
chess	30	0.629	0.644	0.668	0.689	0.693	0.693	0.700	0.696
chess	40	0.637	0.671	0.675	0.691	0.703	0.724	0.731	0.688
chess	50	0.645	0.701	0.739	0.773	0.777	0.788	0.786	0.655
dermatology	30	0.992	0.992	0.992	1.000	0.992	0.992	0.992	0.992
dermatology	40	0.992	0.992	0.992	1.000	1.000	1.000	1.000	0.992
dermatology	50	0.992	0.992	0.992	0.992	0.992	1.000	1.000	0.992
hepatitis	30	0.620	0.605	0.513	0.822	0.822	0.862	0.842	0.843
hepatitis	40	0.611	0.678	0.741	0.778	0.825	0.823	0.871	0.873
hepatitis	50	0.684	0.730	0.695	0.764	0.871	0.859	0.898	0.855
lymph	30	0.810	0.791	0.793	0.827	0.782	0.800	0.829	0.851
lymph	40	0.772	0.781	0.842	0.825	0.823	0.838	0.819	0.827
lymph	50	0.792	0.814	0.841	0.824	0.833	0.816	0.823	0.814
nursery	30	0.810	0.816	0.819	0.809	0.810	0.817	0.818	0.739
nursery	40	1.000	1.000	1.000	1.000	1.000	0.832	0.832	0.773
nursery	50	1.000	1.000	1.000	0.960	1.000	1.000	1.000	0.807
pima	30	0.515	0.535	0.545	0.588	0.543	0.562	0.565	0.532
pima	40	0.509	0.531	0.574	0.568	0.600	0.590	0.607	0.547
pima	50	0.532	0.552	0.603	0.609	0.620	0.613	0.623	0.528
soybean	30	0.834	0.816	0.902	0.893	0.914	0.902	0.915	0.738
soybean	40	0.846	0.893	0.902	0.883	0.915	0.924	0.925	0.767
soybean	50	0.886	0.864	0.854	0.890	0.922	0.916	0.935	0.823
vote	30	0.841	0.838	0.832	0.826	0.798	0.795	0.801	0.679
vote	40	0.850	0.861	0.864	0.850	0.848	0.831	0.814	0.800
vote	50	0.918	0.888	0.854	0.844	0.845	0.846	0.791	0.829
# wins		14	16	20	22	21	25	23	
Avg. F1-score		0.760	0.767	0.779	0.796	0.799	0.802	0.803	0.751

Last column reports Pulce results for comparison

Table 6 F1-score results over the 30 datasets, comparing EGPU and GPU against the competitor Pulce

dataset	%	$\text{EGPU}^{10}_{\text{avr}}$	$\text{EGPU}^{20}_{\text{avr}}$	$\text{EGPU}^{10}_{\text{rnk}}$	$\text{EGPU}^{20}_{\text{rnk}}$	GPU_{LR}	GPU^{BN}	Pulce
audiology	30	0.855	0.889●	0.853	0.875	0.879	0.839	0.745
audiology	40	0.835	0.906	0.921	0.921	0.940	0.879	0.846
audiology	50	0.980●	0.989●	0.980●	0.972●	0.960	0.980	0.899
breast-cancer	30	0.513●	0.490●	0.500●	0.494●	0.462	0.475	0.534
breast-cancer	40	0.506●	0.523●	0.536●	0.513●	0.444	0.483	0.438
breast-cancer	50	0.538●	0.557●	0.526●	0.535●	0.518	0.517	0.443
chess	30	0.640	0.636	0.646	0.646	0.651	0.689	0.696
chess	40	0.666	0.670●	0.667	0.664	0.668	0.691	0.688
chess	50	0.647●	0.651●	0.653●	0.639●	0.637	0.773	0.655
dermatology	30	0.992●	0.992●	0.992●	0.992●	0.992	1.000	0.992
dermatology	40	0.992●	0.992●	0.992●	0.992●	0.964	1.000	0.992
dermatology	50	0.992●	0.992●	0.992●	0.992●	0.992	0.992	0.992
hepatitis	30	0.712●	0.671●	0.652●	0.610●	0.598	0.822	0.843
hepatitis	40	0.722●	0.714●	0.693●	0.732●	0.674	0.778	0.873
hepatitis	50	0.755●	0.730●	0.686	0.752●	0.725	0.764	0.855
lymph	30	0.777●	0.779●	0.761●	0.763●	0.753	0.827	0.851
lymph	40	0.812●	0.783	0.812●	0.791	0.806	0.825	0.827
lymph	50	0.830●	0.815●	0.821●	0.791	0.802	0.824	0.814
nursery	30	0.839●	0.831●	0.841●	0.800●	0.771	0.809	0.739
nursery	40	1.000●	1.000●	1.000●	1.000●	1.000	1.000	0.773
nursery	50	1.000●	1.000●	1.000●	1.000●	1.000	0.960	0.807
pima	30	0.567●	0.576●	0.578●	0.583●	0.565	0.588	0.532
pima	40	0.591●	0.583●	0.591●	0.597●	0.571	0.568	0.547
pima	50	0.601	0.613●	0.610	0.595	0.613	0.609	0.528
soybean	30	0.881●	0.882●	0.875	0.867	0.876	0.893	0.738
soybean	40	0.900	0.889	0.894	0.901	0.931	0.883	0.767
soybean	50	0.909	0.904	0.895	0.893	0.912	0.890	0.823
vote	30	0.865	0.872	0.888●	0.889●	0.882	0.826	0.679
vote	40	0.902●	0.917●	0.911●	0.916●	0.887	0.850	0.800
vote	50	0.913●	0.906●	0.914●	0.902●	0.866	0.844	0.829
# wins/ties ag. Pulce		17/3	18/3	18/3	16/3			
# wins/ties ag. GPU^{BN}		13/3	15/3	14/3	12/2			
# ●		18	19	17	16			

The second column indicates the percentage of positive samples in $\mathcal{P}$

5 Conclusions

In Positive-Unlabeled (PU) learning only positive samples are labeled at training time. PU learning requires algorithms to cleverly exploit dependencies hidden in the data in order to build models able to discriminate between positive and negative samples. In this paper, we proposed to exploit probabilistic generative models for PU learning by characterizing the density distribution for the positive class. The overall GPU framework is flexible enough to be applied on many domains by leveraging tools provided by PGMs. We showed how to create mixtures of generative models by adopting a well-known bagging method from the discriminative framework as an effective and cheap alternative to the classical Expectation Maximization. Dealing with continuous or hybrid domains represent a future study. Results on several benchmark datasets empirically confirmed the validity of our new proposed approach.

References

Antonucci, A., Corani, G., Mauá, D.D., Gabaglio, S. (2013). An ensemble of bayesian networks for multilabel classification. In *Proceedings of the 23rd international joint conference on artificial intelligence, AAAI Press* (pp. 1220–1225).

Babbar, S., & Chawla, S. (2010). On bayesian network and outlier detection. In *Proceedings of the 16th international conference on management of data, Allied Publishers* (pp. 125–138).

Balasubramanian, V. (2005). MDL, Bayesian inference, and the geometry of the space of probability distributions. In *Advances in minimum description length: theory and applications, MIT Press* (pp. 81-98).

Basile, T.M.A., Di Mauro, N., Esposito, F., Ferilli, S., Vergari, A. (2017). Density estimators for positive-unlabeled learning. In *Proceedings of the 6th international workshop on new frontiers in mining complex patterns, Springer, LNCS*, (Vol. 10785 pp. 49-64).

Bengio, Y., Courville, A.C., Vincent, P. (2012). Unsupervised feature learning and deep learning: a review and new perspectives, CoRR arXiv:1206.5538.

Calvo, B., Naga, P.L., Lozano, J.A. (2007). Learning bayesian classifiers from positive and unlabeled examples. *Pattern Recognition Letters*, *28*(16), 2375–2384.

Chandola, V., Banerjee, A., Kumar, V. (2009). Anomaly detection: a survey. *ACM Computing Surveys*, *41*(3), 15:1–15:58.

Chow, C., & Liu, C. (1968). Approximating discrete probability distributions with dependence trees. *IEEE Transactions on Information Theory*, *14*(3), 462–467.

Cooper, G.F., & Herskovits, E. (1990). The computational complexity of probabilistic inference using bayesian belief networks. *Artificial Intelligence*, *42*, 393–405.

Cooper, G.F., & Herskovits, E. (1992). A bayesian method for the induction of probabilistic networks from data. *Machine Learning*, *9*(4), 309–347.

De Comité, F., Denis, F., Gilleron, R., Letouzey, F. (1999). Positive and unlabeled examples help learning. In *Proceedings of the 10th international conference on algorithmic learning theory, Springer, LNAI*, (Vol. 1720 pp. 219-230).

Di Mauro, N., Vergari, A., Basile, T.M.A., Esposito, F. (2017). Fast and accurate density estimation with extremely randomized Cutset networks. In *Proceedings of the European conference on machine learning and knowledge discovery in databases, Springer, LNAI*, (Vol. 10534 pp. 203-219).

Elkan, C., & Noto, K. (2008). Learning classifiers from only positive and unlabeled data. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining, ACM* (pp. 213–220).

Fan, R.E., Chang, K.W., Hsieh, C.J., Wang, X.R., Lin, C.J. (2008). Liblinear: a library for large linear classification. *Journal of Machine Learning Research*, *9*, 1871–1874.

Friedman, N., Geiger, D., Goldszmidt, M. (1997). Bayesian network classifiers. *Machine learning*, *29*(2-3), 131–163.

Hastie, T., Tibshirani, R., Friedman, J. (2009). *The elements of statistical learning*. Berlin: Springer.

Hempstalk, K., Frank, E., Witten, I.H. (2008). One-class classification by combining density and class probability estimation. In *Proceedings of the joint European conference on machine learning and knowledge discovery in databases, Springer, LNAI*, (Vol. 5211 pp. 505-519).

Hinton, G.E., & Salakhutdinov, R.R. (2006). Reducing the dimensionality of data with neural networks. *Science*, *313*(5786), 504–507.

Hoi, C.H., Chan, C.H., Huang, K., Lyu, M.R., King, I. (2004). Biased support vector machine for relevance feedback in image retrieval. In *Proceedings of the International joint conference on neural networks, IEEE* (pp. 3189–3194).

Ienco, D., & Pensa, R.G. (2016). Positive and unlabeled learning in categorical data. *Neurocomputing*, *196*, 113–124.

Ienco, D., Pensa, R.G., Meo, R. (2012). From context to distance: learning dissimilarity for categorical data clustering. *ACM Transactions on Knowledge Discovery from Data*, *6*(1), 1:1–1:25.

Koller, D., & Friedman, N. (2009). *Probabilistic graphical models. Principles and techniques*. Cambridge: MIT Press.

Li, H., Chen, Z., Liu, B., Wei, X., Shao, J. (2014). Spotting fake reviews via collective positive-unlabeled learning. In *Proceedings of the IEEE international conference on data mining, IEEE* (pp. 899–904).

Li, W., Guo, Q., Elkan, C. (2011). A positive and unlabeled learning algorithm for one-class classification of remote-sensing data. *IEEE Transactions on Geoscience and Remote Sensing*, *49*(2), 717–725.

Li, X., & Liu, B. (2003). Learning to classify texts using positive and unlabeled data. In *Proceedings of the 18th international joint conference on Artificial intelligence, Morgan Kaufmann* (pp. 587–592).

Li, X.L., & Liu, B. (2005). Learning from positive and unlabeled examples with different data distributions. In *Proceedings of the European Conference on Machine Learning, Springer, LNAI*, (Vol. 3720 pp. 218-229).

Liu, B., Lee, W.S., Yu, P.S., Li, X. (2002). Partially supervised classification of text documents. In *Proceedings of the 19th international conference on machine learning, Morgan Kaufmann* (pp. 387–394).

Liu, B., Dai, Y., Li, X., Lee, W.S., Yu, P.S. (2003). Building text classifiers using positive and unlabeled examples. In *Proceedings of the 3rd IEEE international conference on data mining, IEEE* (pp. 179–188).

Lowd, D., & Rooshenas, A. (2015). The libra toolkit for probabilistic models. *The Journal of Machine Learning Research*, *16*, 2459–2463.

Meila, M., & Jordan, M.I. (2000). Learning with mixtures of trees. *Journal of Machine Learning Research*, *1*, 1–48.

du Plessis, M.C., & Sugiyama, M. (2014). Semi-supervised learning of class balance under class-prior change by distribution matching. *Neural Networks*, *50*, 110–119.

Riahi, F., Schulte, O., Li, Q. (2014). A proposal for statistical outlier detection in relational structures. In *Proceedings of the 13th AAAI conference on statistical relational AI, AAAI Press* (pp. 93–99).

Roth, D. (1996). On the hardness of approximate reasoning. AI.

Schölkopf, B., Platt, J.C., Shawe-Taylor, J.C., Smola, A.J., Williamson, R.C. (2001). Estimating the support of a high-dimensional distribution. *Neural Computing*, *13*(7), 1443–1471.

Tax, D.M.J., & Duin, R.P.W. (2004). Support vector data description. *Machine Learning*, *54*(1), 45–66.

Vergari, A., Di Mauro, N., Esposito, F. (2015). Simplifying, regularizing and strengthening sum-product network structure learning. In *Proceedings of the international conference on machine learning and knowledge discovery in databases, Springer, LNAI*, (Vol. 9285 pp. 343-358).

Vergari, A., Di mauro, N., Esposito F (2019). Visualizing and understanding sum-product networks. Machine Learning.

Xu, J., & Shelton, C.R. (2010). Intrusion detection using continuous time bayesian networks. *Journal of Artificial Intellingence Research*, *39*(1), 745–774.

Yang, E., Baker, Y., Ravikumar, P., Allen, G., Liu, Z. (2014). Mixed graphical models via exponential families. In *Proceedings of Machine Learning Research*, (Vol. 33 pp. 1042-1050).

Yang, P., Li, X.L., Mei, J.P., Kwoh, C.K., Ng, S.K. (2012). Positive-unlabeled learning for disease gene identification. *Bioinformatics*, *28*, 2640–2647.

Yu, H., Han, J., Chang, K.C.C. (2002). PEBL: Positive example based learning for web page classification using svm. In *Proceedings of the 8th ACM SIGKDD international conference on Knowledge discovery and data mining, ACM* (pp. 239–248).

Zhao, Y., Kong, X., Philip, S.Y. (2011). Positive and unlabeled learning for graph classification. In *Proceedings of the 11th IEEE international conference on data mining, IEEE* (pp. 962–971).

Zhou, J., Pan, S., Mao, Q., Tsang, I. (2012). Multi-view positive and unlabeled learning. In *Proceedings of Machine Learning Research*, (Vol. 25 pp. 555-570).

Zhou, K., Gui-Rong, X., Yang, Q., Yu, Y. (2010). Learning with positive and unlabeled examples using topic-sensitive PLSA. *IEEE Transactions on Knowledge and Data Engineering*, *22*(1), 46–58.