Generative Probabilistic Models for Positive-Unlabeled Learning

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

[1] Department of Computer Science, University of Bari "Aldo Moro", Bari, Italy
[2] Department of Physics, University of Bari "Aldo Moro", Bari, Italy
[3] National Institute for Nuclear Physics (INFN), Bari Division, Bari, Italy

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 on many domains by leveraging tools provided by years of research from the probabilistic generative model community. Results on several benchmark datasets show the performance and flexibility of the proposed approach.

1 Introduction

The classical supervised setting of statistical machine learning [11] 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 ($Y \in \{0, 1\}$) labels 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 data is 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[1]. 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.

[1] 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.

In Positive-Unlabeled (PU) learning [6,19], 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. PU learning shares similarities to semi-supervised learning [22], one class classification [24], and outlier detection [4]. Differently from the first, no negative samples are available at training time and yet it is required to learn a discriminator between the two classes, in contrast with the second. Additionally, PU learning is in opposition to the last which is usually performed in a transductive way to label unlabeled training data only.

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-stage*, taking all samples in $\mathcal{U}$ as negative. For the former, it becomes crucial to learn a metric that is able to discriminate among classes. However, each application domain demands a particular formulation for such a metric. As a consequence, ad hoc algorithmic solutions are often required to cope with different data representations [31,15]. Especially challenging are categorical data, since there is no natural distance for them [16]. Indeed, few approaches have been proposed to deal with this kind of data [3,15] in PU learning.

This work introduces *Generative Positive-Unlabeled* (GPU) learning, a novel two-staged approach to PU learning that aims to be general enough to support very different application domains. It 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. Generative models such as Probabilistic Graphical Models (PGMs) [17] have been extensively studied in the literature and offer a powerful formalism to deal with complex probability distributions over continuous, categorical, or even structured data [28]. Dealing with a particular domain translates into choosing a suitable PGM from a consolidated research field. More generally, given a PGM learned as a density estimator in a certain domain, we exploit it as a negative sample extractor for partially labeled data. Albeit GPU can deal with different data representations, here we focus on categorical data, which are handled natively by PGMs. We compared GPU on real data to several PU learners that have proven to be effective on categorical data.

The paper is organized as follows: in the next Section we provide a brief review of the literature about PU learning; in Section 3 we introduce and discuss our GPU approach, 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 attentions in machine learning and data mining research. An extensively adopted approach to PU learning is based on a negative set construction process, that first identifies reliable negative samples from the unlabeled ones and, then, directly applies traditional classification methods. 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 [29], 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 [15], 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 having theoretically shown that, under appropriate conditions, $\mathcal{P}$ and $\mathcal{U}$ provide sufficient information for learning, in [19] 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 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 [18]. The dataset so obtained is finally exploited to learn a classifier using SVM. 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 reliable negative samples identification. Finally, an EM-based algorithm is exploited to learn the predictive model.

A different policy is the weighted-based approach on unlabeled data exploited in [9]. 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 randomly 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 is the classifier extensively adopted for categorical data in the four methods proposed in [3], namely (Average) Positive Naive Bayes ((A)PNB), based on Naive Bayes, and (Average) Positive TAN ((A)PTAN), two variants of the Tree Augmented Naive Bayes model [10] 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 this 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 ([19,18,3,15]) 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 ([9,29]), 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 [9].

As previously pointed out, although PU learning shares similarities to outlier detection [4] and to one-class classification [24], it shows difference from these settings in both the goal to fulfill and in the training set exploited, even in the case of probability density estimation techniques are used as solving strategies [23,27,25,12]. Indeed, both methods aim at learning a model able to 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 done. Intuitively, this type of approach is inferior because it ignores useful information that is present in the unlabeled samples.

3 Methodology

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 $\mathbf{x}$. 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 [17,28]. To denote a finite domain of a discrete RV X_j we introduce the following notation $\mathsf{Val}(X_j) = \{x_j^k\}_{k=1}^K$. If $\mathcal{D}$ is a set of samples over RVs $\mathbf{X}$, we indicate with $\mathsf{p}_{\mathcal{D}}(\mathbf{X})$ the real and 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 set. 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 done via the *log-likelihood* function, or one of its variants, defined as: $\ell_{\mathcal{D}}(\mathcal{M}) = \sum_{\mathbf{x}^i \in \mathcal{D}} \log \mathsf{p}_{\mathcal{M}}(\mathbf{x}^i)$ [17].

In the classical PU learning setting, one has a training set $\mathcal{D} = \mathcal{P} \cup \mathcal{U}$ i.i.d. from $\mathsf{p}(\mathbf{X}, Y)$. 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 other hand, class information, i.e., labels, is not provided for samples in $\mathcal{U}$, that is $\mathcal{U} = \{\mathbf{x}^i\}_{i=1}^{m_{\mathcal{U}}} \sim \mathsf{p}_{\mathcal{U}}(\mathbf{X})$, 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 prediction about the labels on unseen test data samples. Following [9], we assume that samples in $\mathcal{P}$ are *selected completely at random* from all positive samples in $\mathcal{D}$, i.e., $\mathsf{p}_{\mathcal{P}}(\mathbf{X}|Y=1) = \mathsf{p}_{\mathcal{D}}(\mathbf{X}|Y=1)$.

3.1 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 (RN) from $\mathcal{U}$, then RN is employed to perform supervised learning. In the following we detail our contribution as the first step, discussing possible approaches for the second one.

As a classical assumption 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 already said, as it is common practice in PU learning [9], we assume that the positive samples in $\mathcal{P}$ are highly representative for all positive samples in $\mathcal{D}_1$. As an additional assumption, we consider the distribution generating $\mathcal{D}_0$ and $\mathcal{D}_1$ to be fairly *distinguishable* [1]. 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 appear too strict for real data, in practice, it is commonly adopted while 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.

The high level idea behind our approach is the following. By correctly modeling the probability distribution of positive samples over RVs $\mathbf{X}$, one is able to modeling 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 for applications for anomalous or outlier training samples [27,23]. 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). We discuss the choice of such an estimator in Section 3.2. After that, 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 $\mathcal{U}$. Based on this information we build a set of reliable negative samples, denoted as $\mathcal{N}$ (line 7), we can exploit in the second stage of PU learning. As already stated, 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.

3.2 Bayesian Networks and mixtures of trees.

A question on which generative model to employ arises. 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.

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 $\mathcal{P}$
4: $\mathcal{L} \leftarrow \{\log \mathsf{p}_\mathcal{G}(\mathbf{x}^i) | \mathbf{x}^i \in \mathcal{U}\}$
5: $\mathcal{N} \leftarrow$ reliableNegativeSamples($\mathcal{L}, \mathcal{P}, \mathcal{U}$)
6: $f \leftarrow$ fitClassifier($\mathcal{P}, \mathcal{N}$)
7: **return** f

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*. 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 expressiveness capabilities of TPMs like mixtures of Bayesian trees (MT) [21] and Cutset Networks [8,7] allow for more efficient learning schemes. In this work we evaluate GPU by employing both BNs and MTs to investigate how the model expressiveness influences 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 (DAG) and a set of weights. In the DAG, 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 on the node X_i in the DAG. The structure of the DAG, corresponding to a BN $\mathcal{B}$, induces a factorization of the joint distribution into local factors, that is $\mathsf{p}_\mathcal{B}(\mathbf{X}) = \prod_{i=1}^n p(X_i | \mathsf{Pa}_i)$. Learning a BN corresponds to learn both the structure and the CPD corresponding to each local factor from the data. Classical structure learning algorithms search in the space of BNs guided by a scoring function. While, parameter learning is obtained by maximum likelihood estimation.

On the side of mixtures of generative models, a very competitive density estimator algorithm is MT [21]. MT learns a mixture model $\mathcal{M}$ whose a 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 with the Chow-Liu algorithm [5], 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 the lower treewidth (i.e., nodes have at most one parent in the network), thus leading to efficient learning and inference time. In [21] the best components and weights are found as (local) likelihood maxima by using EM, with k fixed in advance.

3.3 Reliable negative sample elicitation.

Once a generative model $\mathcal{G}$ has been learned one can exploit the density estimation information $\mathcal{G}$ provides 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 a test data may also additionally benefit from the regularization capability of misspecifying 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 $\mathcal{P} \cup \mathcal{N}$ (see next Section).

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. If one adopts 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 $\mathcal{P}$.

3.4 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). 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* [14]. Alternatively, if one focuses on iteratively augmenting the $\mathcal{P}$ set only with GPU, then 1-class SVMs [24] could be employed to derive a max-margin hypersphere for the positive class.

Additionally, the likelihood associated to samples in $\mathcal{U}$ could be interpreted as sample confidence weights. Approaches like that of [30] 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 it is done with the EM algorithm in [19]. 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.

4 Experiments

In this section we empirically evaluate the proposed GPU approach, applying it to categorical data. We are interested in this kind of data because it poses some challenges 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 [16], and only recently PU learning schemes have been devised for it [15]. 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 [13,2,26]. 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 compare to state-of-the-art PU learning approaches? **Q2)** how does the quantity of available positive examples affect GPU learning? **Q3)** how much does the choice of a generative model in estimating $\mathsf{p}_{\mathcal{P}}$ influence GPU performances?

4.1 Experimental setting

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 [15][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 3.2). BNs are learnt with the R package bnlearn[4] (release 4.1.1). To learn they structure we employed the simple score-based hill-climbing algorithm. Concerning parameter estimation, we set the imaginary sample size to 1. MTs are learnt using the Libra [20] toolkit[5] (version 1.1.2). We imposed the number of components to be 10.

As the classifier for the supervised second stage, we adopt the commonly used Support Vector Machines (SVMs)[6] with an RBF kernel as implemented in scikit-learn[7]. 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 $\gamma \in \{0.001, 0.01, 0.1, 1, 10, 100, 1000\}$.

[2] `http://archive.ics.uci.edu/ml/`.
[3] The datasets and settings used in [15] were kindly provided by Dino Ienco.
[4] `http://www.bnlearn.com/`.
[5] `http://libra.cs.uoregon.edu/`.
[6] For this stage only, categorical data is one-hot encoded.
[7] `http://scikit-learn.org/`.

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

We compared GPU with Positive Naive Bayes (PNB), Average Positive Naive Bayes (APNB), Positive TAN (PTAN) and Average Positive TAN (APTAN) [3] and Pulce [15] with $k = 7$. See Section 2 for a description of these methods.

4.2 Results and discussion

Performance on the test set was evaluated using the F-score, defined as $F = 2PR/(P + R)$, where P and R are, respectively, the precision and the recall obtained by the algorithm. Since the number of positive samples is much larger than that of negative ones, as in [15] we directed the computation of P, R, and F-score to the negative samples, differently from their classical setting.

Results are reported in Table 2. We may note that PTAN and APTAN never won against the other approaches, while the two GPU approaches won 55.3% of the times (36.8% of the times $\mathsf{GPU}^{\mathsf{BN}}$ alone), and each GPU approach won more times than any competitor ($\mathsf{GPU}^{\mathsf{BN}}$ more than doubled the number of wins of each competitor). GPU approaches never won on only 2 datasets out of 10; excluding ties, they lost in only 30% (9/30) of the times. 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 $\mathsf{GPU}^{\mathsf{BN}}$ and $\mathsf{GPU}^{\mathsf{MT}}$ are competitive to the current state-of-the-art for categorical data. On datasets on which $\mathsf{GPU}^{\mathsf{BN}}$ does not win in all settings, it still performs comparable or better on settings with larger $\mathcal{P}$ sets. Overall, increasing the size of $\mathcal{P}$ improves the model accuracies on 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 $\mathsf{GPU}^{\mathsf{BN}}$ generally outperforms $\mathsf{GPU}^{\mathsf{MT}}$, the latter is still comparable w.r.t. 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 the probability distribution of the positive class,

Table 2. F-score results over the 30 samples.

dataset	%	$\mathsf{GPU}^{\mathsf{BN}}$	$\mathsf{GPU}^{\mathsf{MT}}$	Pulce	PNB	APNB	PTAN	APTAN
audiology	30	0.841	**0.902**	0.745	0.68	0.7	0.66	0.66
audiology	40	**0.881**	0.804	0.846	0.75	0.74	0.71	0.66
audiology	50	0.955	**0.991**	0.899	0.80	0.80	0.78	0.71
breast-cancer	30	0.487	0.450	**0.534**	0.40	0.39	0.43	0.43
breast-cancer	40	0.487	**0.513**	0.438	0.42	0.40	0.43	0.45
breast-cancer	50	**0.536**	0.535	0.443	0.42	0.41	0.44	0.44
chess	30	0.671	0.663	**0.696**	0.58	0.64	0.59	0.64
chess	40	**0.715**	0.665	0.688	0.58	0.64	0.60	0.64
chess	50	**0.698**	0.650	0.655	0.58	0.64	0.60	0.64
dermatology	30	**0.992**	0.834	**0.992**	0.57	0.57	0.57	0.56
dermatology	40	**0.992**	0.836	**0.992**	0.57	0.58	0.57	0.57
dermatology	50	**1.000**	0.951	0.992	0.59	0.60	0.57	0.58
hepatitis	30	0.576	0.665	0.843	**0.87**	**0.87**	0.85	0.86
hepatitis	40	0.632	0.654	0.873	**0.88**	**0.88**	0.85	0.85
hepatitis	50	0.784	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	0.760	0.795	0.827	**0.84**	0.83	0.79	0.81
lymph	50	0.801	0.835	0.814	0.86	**0.87**	0.81	0.82
nursery	30	0.724	**0.761**	0.739	0.65	0.65	0.56	0.50
nursery	40	0.731	0.762	**0.773**	0.69	0.69	0.61	0.56
nursery	50	**0.949**	0.779	0.807	0.69	0.70	0.74	0.44
pima	30	0.552	**0.576**	0.532	0.49	0.50	0.50	0.50
pima	40	0.555	**0.593**	0.547	0.49	0.50	0.50	0.51
pima	50	0.547	**0.605**	0.528	0.49	0.51	0.50	0.52
soybean	30	**0.901**	0.766	0.738	0.81	0.86	0.80	0.81
soybean	40	**0.858**	0.852	0.767	**0.86**	**0.86**	0.84	0.83
soybean	50	**0.928**	0.923	0.823	0.92	0.92	0.88	0.86
vote	30	**0.849**	0.799	0.679	0.62	0.62	0.56	0.55
vote	40	**0.906**	0.790	0.800	0.71	0.71	0.58	0.54
vote	50	**0.873**	0.829	0.829	0.77	0.77	0.61	0.56
# wins		**14**	7	6	5	6	0	0
Avg. F1-score		**0.767**	0.743	0.751	0.677	0.686	0.653	0.643
30%		0.742	0.720	0.735	0.651	0.665	0.631	0.635
40%		0.752	0.727	0.755	0.679	0.683	0.648	0.642
50%		0.807	0.784	0.764	0.700	0.710	0.679	0.652
Avg. ranking		**2.60**	3.07	2.97	4.57	4.02	5.45	5.30

Table 3. Number of wins/ties among all the methods.

	GPU^{BN}	GPU^{MT}	Pulce	PNB	APNB	PTAN	APTAN	avg.
GPU^{BN}	—	18/0	18/2	23/0	23/0	25/0	24/0	**21.83**
GPU^{MT}	12/0	—	13/0	22/0	22/0	25/0	24/0	**19.67**
Pulce	10/0	17/0	—	22/0	22/0	25/0	24/0	**20**
PNB	7/0	8/0	8/0	—	5/12	18/2	18/3	**10.67**
APNB	7/0	8/0	8/0	13/12	—	23/3	21/4	**13.33**
PTAN	5/0	5/0	5/0	10/2	4/3	—	12/6	**6.83**
APTAN	6/0	6/0	6/0	9/3	5/4	12/6	—	**7.33**

is fairly relevant for achieving better performances. Nevertheless, note that for both GPU^{BN} and GPU^{MT} we employed out of the box PGMs and not invest too much time optimizing the hyperparameters for their structure and weight learning algorithms. It is a matter of future works to explore how increasing a model complexity can degrade its performances, that is when too accurate probability distribution estimates can lead to overfitting.

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. Results on several benchmark datasets empirically confirmed the validity of our new proposed approach.

References

1. Balasubramanian, V.: MDL, Bayesian inference, and the geometry of the space of probability distributions. Advances in minimum description length: Theory and applications pp. 81–98 (2005)
2. Bengio, Y., Courville, A.C., Vincent, P.: Unsupervised feature learning and deep learning: A review and new perspectives. CoRR abs/1206.5538 (2012)
3. Calvo, B., Larraaga, P., Lozano, J.A.: Learning bayesian classifiers from positive and unlabeled examples. Pattern Recognition Letters 28(16), 2375 – 2384 (2007)
4. Chandola, V., Banerjee, A., Kumar, V.: Anomaly detection: A survey. ACM Comput. Surv. 41(3), 15:1–15:58 (Jul 2009)
5. Chow, C., Liu, C.: Approximating discrete probability distributions with dependence trees. IEEE Transactions on Information Theory (1968)
6. De Comité, F., Denis, F., Gilleron, R., Letouzey, F.: Positive and unlabeled examples help learning. In: ALT. pp. 219–230. Springer (1999)
7. Di Mauro, N., Vergari, A., Basile, T.: Learning bayesian random cutset forests. In: ISMIS (2015)

8. Di Mauro, N., Vergari, A., Esposito, F.: Learning accurate cutset networks by exploiting decomposability. In: AIXIA (2015)
9. Elkan, C., Noto, K.: Learning classifiers from only positive and unlabeled data. In: KDD. pp. 213–220 (2008)
10. Friedman, N., Geiger, D., Goldszmidt, M.: Bayesian network classifiers. Machine learning 29(2-3), 131–163 (1997)
11. Hastie, T., Tibshirani, R., Friedman, J.: The Elements of Statistical Learning. Springer (2009)
12. Hempstalk, K., Frank, E., Witten, I.H.: One-class classification by combining density and class probability estimation. In: ECML/PKDD. pp. 505–519 (2008)
13. Hinton, G.E., Salakhutdinov, R.R.: Reducing the dimensionality of data with neural networks. Science (2006)
14. Hoi, C.H., Chan, C.H., Huang, K., Lyu, M.R., King, I.: Biased support vector machine for relevance feedback in image retrieval. In: IJCNN. pp. 3189–3194 (2004)
15. Ienco, D., Pensa, R.G.: Positive and unlabeled learning in categorical data. Neurocomputing 196, 113–124 (2016)
16. Ienco, D., Pensa, R.G., Meo, R.: From context to distance: Learning dissimilarity for categorical data clustering. TKDD 6(1), 1:1–1:25 (2012)
17. Koller, D., Friedman, N.: Probabilistic Graphical Models: Principles and Techniques. MIT Press (2009)
18. Liu, B., Dai, Y., Li, X., Lee, W.S., Yu, P.S.: Building text classifiers using positive and unlabeled examples. In: ICDM. pp. 179–188 (2003)
19. Liu, B., Lee, W.S., Yu, P.S., Li, X.: Partially supervised classification of text documents. In: ICML. pp. 387–394 (2002)
20. Lowd, D., Rooshenas, A.: The Libra Toolkit for Probabilistic Models. CoRR abs/1504.00110 (2015)
21. Meil, M., Jordan, M.I.: Learning with mixtures of trees. JMLR (2000)
22. du Plessis, M.C., Sugiyama, M.: Semi-supervised learning of class balance under class-prior change by distribution matching. Neural Networks 50, 110 – 119 (2014)
23. Riahi, F., Schulte, O., Li, Q.: A proposal for statistical outlier detection in relational structures. In: SRAI AAAI Workshop (2014)
24. Schölkopf, B., Platt, J.C., Shawe-Taylor, J.C., Smola, A.J., Williamson, R.C.: Estimating the support of a high-dimensional distribution. Neural Comput. 13(7), 1443–1471 (Jul 2001)
25. Tax, D.M.J., Duin, R.P.W.: Support vector data description. Machine Learning 54(1), 45–66 (2004)
26. Vergari, A., Di Mauro, N., Esposito, F.: Visualizing and understanding sum-product networks. CoRR (2016)
27. Xu, J., Shelton, C.R.: Intrusion detection using continuous time bayesian networks. J. Artif. Int. Res. 39(1), 745–774 (2010)
28. Yang, E., Baker, Y., Ravikumar, P., Allen, G., Liu, Z.: Mixed graphical models via exponential families. In: AISTATS. pp. 1042–1050 (2014)
29. Yang, P., Li, X.L., Mei, J.P., Kwoh, C.K., Ng, S.K.: Positive-unlabeled learning for disease gene identification. Bioinformatics 28, 2640–2647 (2012)
30. Zhou, J., Pan, S., Mao, Q., Tsang, I.: Multi-view positive and unlabeled learning. In: ACML. pp. 555–570 (2012)
31. Zhou, K., Gui-Rong, X., Yang, Q., Yu, Y.: Learning with positive and unlabeled examples using topic-sensitive plsa. TKDE 22(1), 46–58 (2010)