IE 506: Machine Learning - Principles and Techniques **Jan-May 2023**

End-term Project Report : Density Estimators for PU Learning

Team Name: The Finishers *Team Members: 22N0456 ; 22N0461*

Abstract

The main aim of Density Estimator for PU learning used to estimate the probability density function of the positive and negative instances in the unlabeled data. The estimated densities can be used to adjust the decision boundary of a classifier to better handle the presence of unlabeled negative instances. In past there had been many algorithm applied to get better density estimators like biased SVM, distance based approaches but their accuracy for that is not too good as proposed in this paper as well as our work on modification. So firstly we applied GPU approach which extracts a set of reliable negative samples from unlabeled ones and then it perform supervised learning. For supervised learning we applied SVM with RBF kernel. For modification we worked on MNIST dataset and for supervised learning we used Random forest instead of SVM with RBF kernel for both the data that was given in paper as well as for MNIST data.

As a result we obtained that the accuracy using Random Forest classifier is far better than SVM with RBF kernel for almost all of the dataset.

1 Introduction

Positive Unlabeled (PU) learning is a type of machine learning where a dataset consists of two types of samples: positive and unlabeled. Main challenge in PU learning is the lack of negative examples, which makes it difficult to estimate the true positive rate and the false positive rate. These negative instances can bias the decision boundary of a classifier towards the negative class, resulting in poor performance. PU learning is particularly useful in scenarios where labeling negative instances is difficult or expensive, such as in disease diagnosis, fraud detection, and spam filtering. This project introduces Generative positive unlabeled 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 of positive samples through generative model and then it performs inference on such a distribution to select a set of reliable negative samples from Unlabeled ones.

The main algorithm provided in this project is provided as follows firstly we take input as a set of positive samples and a set of unlabeled samples over considered random variables X of the positive training set. The given output is considered as a train discriminative model learn on positive samples and a reliable negative samples extracted from unlabeled samples. We evaluate GPU by exploiting both Bayesian Network and Mixture of trees to investigate how the model expressiveness affects the estimation of positive samples and therefore ultimately the accuracy of the learned discriminator. Through GPU we extract reliable negative samples and take the log likelihood of that samples. Then we fit a classifier either SVM or Random Forest to obtain the desired result.

Further in this report we provide a survey of existing literature in Section 2. Our proposal for the project is described in Section 3. We give details on experiments in Section 5. A description of future work is given in Section 7. We conclude with a short summary and pointers to forthcoming work in Section 8.

2 Literature Survey

In the previous work there have been proposed many work in this topic but we discussed distance based approaches and biased SVM. I provide these two past work in which distance based approach was proposed recently in the year 2013 and biased SVM was a older approach proposed in 2003.

2.1 Distance Based Approach

This paper was published in the journal Bioinformatics in 2012. The authors of the paper are Pengyi Yang, Xue Liang Li, Jian-Ping Mei, Chong K. Kwoh, and See-Kiong Ng. The distance-based approach is a common method for density estimation in positive-unlabeled (PU) learning. The aim is to identify negative samples, as the farthest unlabeled ones from positive samples.Based on the Euclidean distance the unlabeled set is partitioned into four sets (reliable/likely/weak negative and likely positive).

The main objective of this paper was to propose a positive-unlabeled (PU) learning approach for identifying disease genes, which can be considered a binary classification problem where the positive class corresponds to known disease genes and the negative class corresponds to non-disease genes.

To address the challenges of using only positive and unlabeled data for training a classifier, the authors proposed a PU learning algorithm that uses a support vector machine (SVM) classifier with a radial basis function (RBF) kernel. The algorithm first uses a pre-selection step to identify a set of potential disease genes based on their known functional annotations. It then trains the SVM classifier using the positive genes and a set of reliable negative genes, which are identified using the K-nearest neighbor (KNN) algorithm The authors evaluated the performance of their approach on three different disease gene datasets, including the Online Mendelian Inheritance in Man (OMIM) database. The results show that their PU learning algorithm outperforms several other state-of-the-art methods for disease gene identification, including a naive Bayes-based method and a supervised learning method using both positive and negative samples.

Overall, this study provides a novel approach for disease gene identification using PU learning, which can be applied to a wide range of disease datasets. The study highlights the potential of PU learning algorithms in overcoming the challenges of using only positive and unlabeled data for classification tasks, and it demonstrates the effectiveness of using SVM classifiers with RBF kernels in PU learning.

This algorithm was also better but not so better than the one that is given in paper. In our paper to extract reliable negative sample GPU approach was used which is far better than this approach.

2.2 Biased SVM

A modification of SVM to handle PU Learning by introducing a bias term.

The paper "Building classifiers using positive and unlabeled text examples" by Liu et al. (2003) proposed a novel approach to learning from positive and unlabeled data using a classifier called the Positive-Unlabeled (PU) learning algorithm. The authors showed that by assuming that the negative class is a mixture of some known positive and unknown negative examples, they could use this assumption to estimate the density of the negative class and train a binary classifier.

The paper evaluated the performance of their approach on several benchmark text classification datasets and compared it with other methods, including standard supervised learning, naive Bayes, and the Roc-

chio algorithm. The results showed that the proposed PU learning algorithm outperformed the other methods in terms of accuracy and F-measure. The authors also conducted experiments to demonstrate the robustness of their approach to different ratios of positive and negative examples and found that the PU learning algorithm could handle imbalanced data well.

In conclusion, the paper presents a promising approach to learning from positive and unlabeled data, which could have significant applications in areas such as text classification, information retrieval, and bioinformatics. The proposed PU learning algorithm is simple, efficient, and effective, and it could be extended to other domains beyond text classification.

The main Advantages of Biased SVM can improve the accuracy of PU Learning by accounting for the class imbalance and reducing the effect of false positives.Limitations of the accuracy of Biased SVM is highly dependent on the prior probability of the positive class and may perform poorly in extreme class imbalances.

3 Methods and Approaches

None of the past approaches give the best accuracy as given in our project. In this project we use Generative PU learning (GPU), which falls in the category of two-staged methods for PU learning.

3.1 Generative Models for PU learning

The main work of GPU is to extracts a set of reliable negative samples from Unlabeled, then Negative samples is employed to perform supervised learning. In statistical machine learning, we assume p_D to be modeled as a mixture of probability distributions for the positive and negative class i.e.,

$$p_D = \sum_{y \in 0,1} p(Y = y)p(X|Y = y) = w_{D0}p_{D0}(X) + w_{D1}p_{D1}(X),$$

where w_{D0} (resp. w_{D1}) denotes the marginal probabilities of the label with respect to the negative (resp. positive) class, and $p_{D0}(X)$ (resp. $p_{D1}(X)$) denotes the conditional probability of a sample with respect to the negative (resp. positive) class.

The idea behind our approach is to correctly modelling the probability distribution of positive samples over RVs X, one can model discriminative patterns among samples in the form of probabilistic dependencies among their RVs.

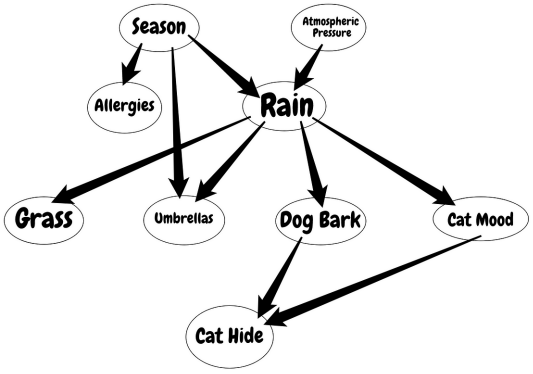
Algorithm 1:

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

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 $\mathsf{Val}(Y) = \{0, 1\}$.
Output: a trained discriminative model learned on positive samples $\mathcal{P}$ and reliable negative samples $\mathcal{N}$ extracted from $\mathcal{U}$
$\mathcal{G} \leftarrow \mathsf{learnGenerativeModel}(\mathcal{P}, \mathbf{X})$ ▷ learn a generative model $\mathcal{G}$ from $\mathcal{P}$
$\mathcal{L} \leftarrow \{\log \mathsf{p}_\mathcal{G}(\mathbf{x}^i) | \mathbf{x}^i \in \mathcal{U}\}$
$\mathcal{N} \leftarrow \mathsf{reliableNegativeSamples}(\mathcal{L}, \mathcal{P}, \mathcal{U})$
$f \leftarrow \mathsf{fitClassifier}(\mathcal{P}, \mathcal{N})$
return f

In this project we employed GPU by exploiting both BNs and MTs to investigate how the model expressiveness affects the estimation of positive samples and therefore ultimately the accuracy of learned discriminator. BNs are a PGM encoding a probability distribution by means of directed acyclic graph and a set of weights, where nodes corresponds to RVs and edges to dependencies among RVs.

Bayesian Network : Below shows a simple example of Bayesian Network



The structure of BN, induces a factorization of joint distribution into local factors, that is:

$$p_G(X) = \prod_{i=1}^{n} p(X_i | Pa_i)$$

Concerning mixture of generative models, a very competitive density estimation algorithm is Mixture of trees(MT). MT learns a mixture model M whose distribution factorizes according to

$$p_M(X) = \sum_{k=1}^{i} \lambda_i p(T_i(X)),$$

where $T_i(X)$ is the i^{th} decision tree.

The distribution p(Ti), learned using the Chow-Liu algorithm are the mixture components and $\lambda_i \geq 0$ with $\sum_{i=1}^{k} \lambda_i = 1$ are their coefficients.

3.2 Elicitation of Reliable Negative Sample

After learning a generative model G, we can use the density estimation information provided by G in several ways. One approach is to set a threshold hyperparameter 't', so that any sample in U whose log-likelihood score falls below t is added to the negative set N. However, determining the best value for t would require additional hyperparameter optimization, which is not ideal.To address this issue, we propose building the negative set N to comprise the mP = —P— samples in U with the lowest log-likelihood score according to G. This approach ensures that the resulting labeled set P u N is balanced w.r.t. the positive and negative class.

Lastly, we note that density information in the form of the finite set log-likelihoods can be directly incorporated into the construction of the classifier over P u N. While we use the likelihoods to select the most reliable negative samples from U, we could also use them to select the most reliable positive samples instead. After a stopping criterion is met, N can be built by collecting all the samples in U that were not added to P.

3.3 Classification Step

In the GPU approach, any supervised classifier can be used after constructing the set N. The empirical evaluation in Section 5 of the report uses a regular implementation of Support Vector Machines (SVMs). However, the authors discuss other interesting variants, such as using biased SVMs for an unbalanced set N or using 1-class SVMs for iteratively augmenting the set P only with GPU. The likelihoods associated with samples in U can also be used as sample confidence weights to learn a weighted classifier over P u U, without building N. Overall, the paper provides several options for the second stage of GPU, depending on the specific requirements of the problem at hand.

After mid-sem review we also worked through Random Forest classifier. We used this classifier for all datasets as we used for SVM with gaussian kernel. This classifier gives more superior result than SVM with gaussian kernel.

3.4 Work done before mid-term project review

Before mid-term report, we undertand the paper. There was little bit difficulties in understanding the paper in well manner though it was clear by some google search, you tube and chatgpt. After reading the paper we have the knowledge of motivation for publishing this paper, what is the main agenda of the paper, we understands the solution approach of the paper, understands the main algorithm and experiment it for three datasets and conclude the results for the same.

After understanding the paper we start implementing code. While implementing code we face many difficulties while implementing the code of author. The issues was corrected by asking doubts from sir and TAs. The main issue was in data reading which was corrected by importing some libraries like liac-arff, bnlearn, tarfile etc. Some part of the authors code was run in R language by using R command in python file. So this part also gave error in completion of code. After understanding the code of R command we successfully ran our code for three of the datasets that was given in paper.

3.5 Work done after mid-term project review

After mid-term review, we start implementing the code for rest of the datasets which was implemented before mid-term. Again we phase difficulties in running the code for rest of the datasets like some of the datasets takes too much time to implement the code like chess data take 2-3 hours to run the code in

Google Colaboratory so we move to IEOR server which has higher RAM so now our code take bit less time than previous one.

We implement the code for seven of the datasets using SVM with RBF kernel and Random Forest and observe the results. The results for Random Forest is better than SVM with RBF kernel.

After implementing the code for paper's dataset we start implementing code for mnist data. Mnist dataset takes too much in pre-processing and then we break this multilabel data into binary class and change into ratio of different positive samples. After doing this we implement the code for this dataset but partially succeeded and cannot able to analyze the result for this dataset.

4 Data set Details

The dataset was used in the paper is given in UCI machine learning repository. (http://archive.ics.uci.edu/ml/). We ran 10- fold cross validations as given in below table. The three settings were generated by putting in P 30We consider when dataset does not describe a binary classification problem, then we consider the two heavily populated class.In our experiments, we consider all numerical attributes were discretized into 10 equal-width bins. The data that was given by authors is already pre-processed and it was already converted into 30,40 and 50 percent samples. For unzipping and pre-processing the data we use many libraries. We firstly store the data in our google drive and then we import from their in our code. For unzipping line by line we use library named tarfile and then we converted into csv format and quote it.

We described our data in the following table:

Dataset	#Attributes	%pos 30,40,50						#test
		#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
Hepatitis	19	9	130	12	127	15	124	16
Nursery	8	1166	6562	1555	6173	1944	5784	859
Soybean	35	25	140	33	132	42	123	18
Vote	16	72	319	96	295	120	271	44

After mid-term review we also work on **mnist** dataset. But partially succeeded to get result from this data. Mnist dataset is a widely used dataset of handwritten digits. It contains 60,000 training images and 10,000 testing images of 28x28 pixels, each representing a grayscale image of a handwritten digit from 0 to 9. The dataset was created by Yann LeCun, Corinna Cortes, and Christopher Burges for the purpose of training and testing machine learning models on image recognition tasks. We import this from sklearn while implementing our (code.from sklearn.datasets import fetch openml).

For pre-processing of mnist data, though it is multilabel data we first convert it into binary class and then we change it into samples of 30,40,50 percent and rest of the part we considered as unlabeled. Basically we give one label to our 30,40,50 percent and called it as positive class and rest part we called it as unlabeled.

5 Experiments

5.1 Training for dataset given in paper

In this section, we empirically evaluate proposed GPU approach that was given in paper, which is applied to categorical data. The authors highlight that this type of data is challenging for classical metric-based approaches as 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, and only recently PU learning schemes have been devised for it. However, 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. In paper it was given state that adapting GPU to other domains reduces to selecting an appropriate generative toolbox from the probabilistic model literature. In paper they aim to answer the following research questions through their empirical evaluation:

(Q1) how does GPU compare to state-of-the-art PU learning approaches?

(Q2) how does the quantity of available positive samples affect GPU learning?

(Q3) how much does the choice of a generative model in estimating positive samples affect GPU's performance?

In our experiment we wanted to test how well our GPU approach works on categorical data. This kind of data is difficult to work with using traditional methods. We used two different kinds of generative models, called Bayesian networks (GPUBN) and mixture trees (GPUMT), to help us learn and understand the data. To avoid overfitting, we used a special scoring function called K2 when working with Bayesian networks given below:

$scoreK2(G:P) = \log p(G) + \sum_{i=1}^{n} \sum_{j=1}^{q_i} \log \frac{(r_i-1)!}{(N_{ij}+r_i-1)!} + \sum_{i=1}^{n} \sum_{k=1}^{r_i} \log(N_{ijk}!)$

We also used a commonly used machine learning algorithm called SVMs to help classify the data in the second stage. We compared our GPU approach to several other methods, including Positive Naive Bayes (PNB) and Pulce, and optimized our parameters through cross-validation. Overall, our approach showed promising results and could be a useful tool for analyzing categorical data in the future.

In the code that was explained during end-term presentation and also explained in code walk through video. Firstly we import sufficient libraries and pre-process our data explained in **4**th section then we give class lables to them, then for storing our result we create in log data file then for each samples we define a empty list for precision, recall, and error and F1 score and ran 10 fold cross validation for each samples. Then we train our positive and unalabeled data and GPU approach for bayesian network was running in R command that extract reliable negative samples using some threshold and then call our function for the same and then we use one hot encoder and then fit our to our SVM with RBF kernel classifier and print our result and print precision recall accuracy error for each of the samples and for each fold we consider.

Same work we do it for Random Forest classifier as given in above paragraph by only classifier here instead of SVM we use Random Forest.

Before mid-term we work on **Google Colabatory**.

After mid-term we move into **IEOR server** with 20 Core CPU: Intel(R) Xeon(R) CPU E5-2670 v2 @ 2.50GHz and 128 GB RAM. The reason behind moving into IEOR server is while running the code it takes very long time for pre-processing data and for training also.

6 Results

We evaluated performance on the test set using F1-score which measures the accuracy, defined as F = 2PR/(P +R) = 2tp/(2tp+f p+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 no information for negative class is provided, correctly predicting negative samples should be somehow harder than focusing on the positive counterparts.

Overall result was reported in below table 1 and table 2, where **table 1** shows the result of seven datasets that was given in paper using both classifier SVM with RBF kernel and also form Random Forest. This result is of the code that was implemented by us shown during presentation. The table shows as we increase the percentage of positive samples the accuracy increases for almost all of the datasets. **Table 1** also shows average F1 score for all of the datasets.

Table 2 shows the number of positive samples uncorrectly predicted as negative ones in the negative sample elicitation phase. In paranthesis the average number of errors for each fold over the cardinality of both positive and negative samples.

Also in this section we aim at answering the questions that were ask in experiment section, for Q1 we can say that both GPU for BN and GPU for MT are competitive to the current state-of-the-art for categorical data. In Q2 we observe that while GPU for BN generally outperforms GPU for MT and overall more accurate than all other methods. In Q3 we can state that the greater expressiveness of BNs, allowing better modeling the probability distribution of the positive class, is fairly relevant for achieving better performances.

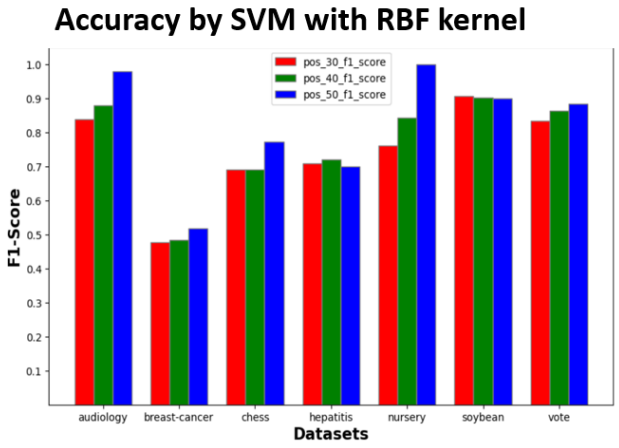
Table 1:

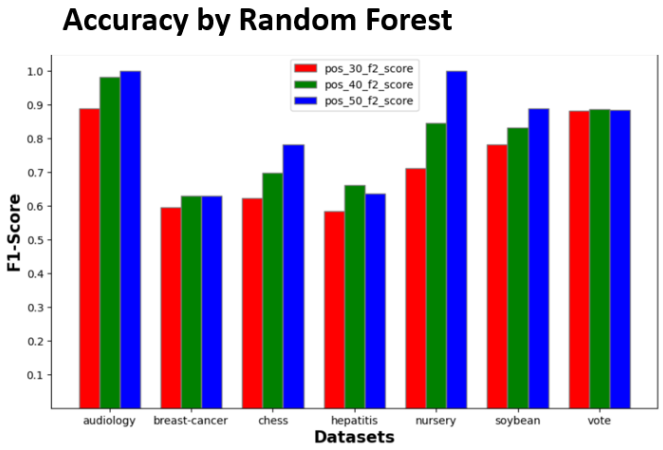
Dataset	Classifier	#Attributes	%pos 30,40,50						Avg. F1-Score
			#pos	F1-Score	#pos	F1-Score	#pos	F1-Score	
Audiology	SVM with RBF	69	15	0.8388	20	0.8789	26	0.9798	0.8981
Audiology	Random Forest	69	15	0.8272	20	0.9808	26	1.0	0.9360
Breast-cancer	SVM with RBF	9	54	0.4755	72	0.4828	91	0.5166	0.4916
Breast-cancer	Random Forest	9	54	0.5943	72	0.6282	91	0.6282	0.6169
Chess	SVM with RBF	36	451	0.6894	601	0.6911	751	0.7731	0.7178
Chess	Random Forest	36	451	0.6227	601	0.6975	751	0.7810	0.7004
Hepatitis	SVM with RBF	19	9	0.7096	12	0.6598	15	0.6984	0.6893
Hepatitis	Random Forest	19	9	0.5825	12	0.6598	15	0.6367	0.6264
Nursery	SVM with RBF	8	1166	0.7600	1555	0.8424	1944	1.0	0.8675
Nursery	Random Forest	8	1166	0.7099	1555	0.8448	1944	0.9989	0.8512
Soybean	SVM with RBF	35	25	0.9064	33	0.9007	42	0.8988	0.9020
Soybean	Random Forest	35	25	0.7817	33	0.8323	42	0.8879	0.8340
Vote	SVM with RBF	16	72	0.8325	96	0.8621	120	0.8828	0.8591
Vote	Random Forest	16	72	0.8810	96	0.8854	120	0.8826	0.8830

Table 2:

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)
Hepatitis	0.19	(1.7/9)	0.0	(0.0/12)	0.11	(1.7/15)
Nursery	0.00	(0.0/1166)	0.0	(0.0/15550)	0.03	(50.4/1944)
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)

Below shows a **BarPlot of F1-score vs datasets** using SVM with RBF kernel and using Random Forest. We can easily see that accuracy of Random forest is better than SVM with RBF in almost all the datasets.





7 Future Work

- Further research is needed to investigate the full potential of the proposed method and to explore its applicability to other related tasks.

- To investigate how to adapt GPU learning to more complex learning settings.

- To explore how increasing a model complexity can degrade its performance, that is when too accurate probability distribution estimates can lead to overfitting.

8 Conclusion

PU learning considers a set of positive samples and a larger set of unlabeled samples in which positive samples are labeled at training time. GPU acceleration can be used to speed up the density estimation

process. The framework is flexible and can be applied to many domains. We observe the results for different datasets using classifier SVM with RBF kernel and Random forest. Results on several benchmark datasets show the performance and flexibility of the proposed approach.

In conclusion we can say that this paper presents a density estimator for PU learning that leverages the power of GPUs to extract reliable negative samples. The use of GPU for negative sample selection enables the method to extract reliable negative samples with high precision, resulting in improved performance. The SVM with RBF kernel-based density estimation approach is able to effectively estimate the underlying positive distribution and classify new samples as positive. Random Forest-based density estimators provide a promising approach for addressing the challenges of density estimation in PU learning and have shown to be effective in practice.

References

1. Lowd, D., Rooshenas, A.: The Libra toolkit for probabilistic models. CoRR abs / 1504. 00110 (2015)

2. Basile, T., Mauro, N.D., Esposito, F., Ferilli, S., Vergari, A.: Generative probabilistic models for positive-unlabeled learning. In: Workshop on NFMCP Held with ECML/PKDD (2017)

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

4. Calvo, B., Larraaga, P., Lozano, J.A.: Learning bayesian classifiers from positive and unlabeled examples. Pattern Recogn. Lett. 28(16), 2375–2384 (2007)

5. Di Mauro, N., Vergari, A., Basile, T.M.A.: Learning Bayesian random cutset forests. In: Esposito, F., Pivert, O., Hacid, M.-S., Ra´s, Z.W., Ferilli, S. (eds.) ISMIS 2015. LNCS (LNAI), vol. 9384, pp. 122–132. Springer, Cham (2015). https://doi. org/10.1007/978-3-319-25252-0 13

6. Elkan, C., Noto, K.: Learning classifiers from only positive and unlabeled data. In: KDD, pp. 213–220 (2008)

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

8. D., Friedman, N.: Probabilistic Graphical Models: Principles and Techniques. MIT Press, Cambridge (2009)

9. B., Lee, W.S., Yu, P.S., Li, X.: Partially supervised classification of text documents. In: ICML, pp. 387–394 (2002)

10. Sch¨olkopf, 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 (2001)

11. Zhao, Y., Kong, X., Philip, S.Y.: Positive and unlabeled learning for graph classification. In: ICDM, pp. 962–971 (2011)

12. Zhou, J., Pan, S., Mao, Q., Tsang, I.: Multi-view positive and unlabeled learning. In: ACML, pp. 555–570 (2012)

13. 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)

14. import mnist784 data from sklearn.datasets import fetchopenml

15. http://archive.ics.uci.edu/ml/