Positive unlabeled learning with tensor networks

Bojan Žunkovič*
Faculty of computer and information science
University of ljubljana
Večna pot 113, Ljubljana, Slovenia
bojan.zunkovic@fri.uni-lj.si

Abstract

We apply the locally purified state tensor network to the positive unlabeled learning problem. We test the model on the image MNIST dataset and achieve state-of-the-art results even with very few labeled positive samples. We find that the agreement fraction between outputs of different models on unlabeled samples is a good indicator of the model's performance. This enables us to devise a 1-sample positive unlabeled learning strategy for unsupervised clustering. Our method can also generate new positive and negative samples, which we demonstrate on simple synthetic datasets.

1 Introduction

Positive unlabeled learning (PUL) is a binary classification problem with only some labeled positive samples and many more unlabeled positive and negative samples [1]. Typical PUL problems arise in medicine (e.g. disease gene identification [2], identifying protein complexes [3]), drug discovery [4], remote sensing [3], recommender systems [5], and more.

The main part of our model is a tensor network (TN) called a locally purified state [6, 7][2]. Tensor networks have already been applied for classification [8, 9, 10], generative modelling [11], image segmentation [12], anomaly detection[6]. Although TNs provide competitive results, state-of-the-art is only rarely achieved. A notable example is the anomaly detection with tensor networks [6], which is the basis of our approach.

Most approaches to positive unlabeled learning are applicable to either text data [3, 13, 14], or categorical data [15], or images [16, 17]. Instead our tensor-network approach can be applied in all domains. Here, we will show results on synthetic datasets and on the MNIST dataset. To best of our knowledge our results are first where a tensor-network model outperforms the state-of-the-art deep neural network (in this case, a generative adversarial network GAN) on image datasets. This is promising since we expect better results on categorical data [6].

2 Model

In this section, we shall discuss the main parts of the setup, i.e. the model architecture, the training loss, the batch generation procedure, and sampling. We will assume that the input vector x has size N with continuous elements belonging to the unit interval, i.e. $x_i \in [0, 1]$.

Architecture Our PUL tensor network model has two parts. The first part is the embedding layer. We define the embedding with a local vector transformation, which maps each input vector element

*https://qmltn.ai/

[2]https://github.com/qml-tn/pul

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

into a local Hilbert space. We use the following one-parameter Fourier embedding

$$\phi(x_i) = (1, \cos(x_i\pi), \cos(2x_i\pi), \ldots, \cos((d-1)x_i\pi)), \tag{1}$$

where the parameter d determines the dimension of the embedding Hilbert space. Instead of the cosine, we sometimes use the sine basis functions. An important property of the local embedding function is the element-wise orthonormality on the domain, i.e. $\int_0^1 \mathrm{d}u \phi_a(u)\phi_b(u) = \delta_{a,b}$, which will enable efficient sampling [8].

The second part of the model consists of two locally purified state (LPS) tensor networks. An LPS is a one-dimensional tensor network consisting of three-dimensional tensors $A^i \in \mathbb{R}^{D\times D\times d}$ and four-dimensional tensors $B^i \in \mathbb{R}^{D\times D\times d\times d}$. We calculate an output $\hat{y}$ associated with an LPS via the Born rule.

First, we construct matrices $\mathcal{A}^i$ and tensors $\mathcal{B}^i$ by contracting the local embeddings with A^i and B^i

$$\mathcal{A}^i_{a,b} = \sum_{\alpha=1}^{d} \phi_\alpha(x_i) A^i_{a,b,\alpha}, \quad \mathcal{B}^i_{a,b,\beta} = \sum_{\alpha=1}^{d} \phi_\alpha(x_i) B^i_{a,b,\alpha,\beta}. \tag{2}$$

We construct tensors $\mathcal{B}^i$ for positions $i \in \mathcal{I} = \{1, S+1, 2S+1, \ldots (\lfloor N/S \rfloor - 1)S + 1, N\}$. For the remaining positions, we construct the tensors $\mathcal{A}^i$. In other words, every Sth tensor is $\mathcal{B}^i$ (including the first and the last one), and the remaining are $\mathcal{A}^i$. Next, we calculate matrices $\mathcal{C}^i \in \mathbb{R}^{D^2\times D^2}$ which also depend on the position

$$\mathcal{C}^i_{(a_1,a_2),(b_1,b_2)} = \begin{cases} \sum_{\beta=1}^{d} \mathcal{B}^i_{a_1,b_1,\beta}\mathcal{B}^i_{a_2,b_2,\beta} & i \in \mathcal{I} \\ \mathcal{A}^i_{a_1,b_1}\mathcal{A}^i_{a_2,b_2} & \text{else} \end{cases}. \tag{3}$$

Finally, we calculate the output by taking the trace of the product of matrices $\mathcal{C}$

$$\hat{y}(x) = \operatorname{Tr} \mathcal{C}^1\mathcal{C}^2\ldots\mathcal{C}^N. \tag{4}$$

Our model consists of two LPS tensor networks, one associated with positive samples (producing the output $\hat{y}^{\mathrm{p}}(x)$), and one with negative samples (producing the output $\hat{y}^{\mathrm{n}}(x)$). We classify the sample as positive if $\hat{y}^{\mathrm{p}}(x) > \hat{y}^{\mathrm{n}}(x)$.

Since our model will also be a generative model, we have to calculate the normalisation of the positive LPS $\mathcal{N}^{\mathrm{p}}$ and negative LPS $\mathcal{N}^{\mathrm{n}}$ separately. Similarly, as for the outputs, we first calculate the matrices $\tilde{\mathcal{C}}^i$

$$\tilde{\mathcal{C}}^i_{(a_1,a_2),(b_1,b_2)} = \begin{cases} \sum_{\alpha,\beta=1}^{d} B^i_{a_1,b_1,\alpha,\beta}B^i_{a_2,b_2,\alpha,\beta} & i \in \mathcal{I} \\ \sum_{\alpha=1}^{d} A^i_{a_1,b_1,\alpha}A^i_{a_2,b_2,\alpha} & \text{else} \end{cases}. \tag{5}$$

We then calculate the normalisation by taking the trace of the product of $\tilde{\mathcal{C}}^i$

$$\mathcal{N} = \operatorname{Tr} \tilde{\mathcal{C}}^1\tilde{\mathcal{C}}^2\ldots\tilde{\mathcal{C}}^N. \tag{6}$$

Loss For positive samples x, we want the $\hat{y}^{\mathrm{p}}(x)$ to be large and $\hat{y}^{\mathrm{n}}(x)$ to be small and the opposite for negative samples. Since we will interpret the positive and negative LPS as probability distributions, we also want the 2-norm of the tensor networks to be close to one. Finally, to avoid the collapse of the network to one class, we want the sum of the outputs $\hat{y}^{\mathrm{p}} + \hat{y}^{\mathrm{n}}$ to be close to zero. Our loss has five terms

$$\mathcal{L} = \frac{1}{|\mathcal{D}^{\mathrm{l}}|}\sum_{x\in\mathcal{D}^{\mathrm{l}}} \mathcal{L}^1(x) + \frac{1}{|\mathcal{D}^{\mathrm{p}}|}\sum_{x\in\mathcal{D}^{\mathrm{p}}} \mathcal{L}^2(x) + \frac{1}{|\mathcal{D}^{\mathrm{n}}|}\sum_{x\in\mathcal{D}^{\mathrm{n}}} \mathcal{L}^3(x) + \frac{1}{|\mathcal{D}^{\mathrm{u}}|}\sum_{x\in\mathcal{D}^{\mathrm{u}}} \mathcal{L}^4(x) + \mathcal{L}^5, \tag{7}$$

where

$$\begin{aligned} \mathcal{L}^1(x) &= \lambda_1(\log(\hat{y}^{\mathrm{p}}(x)) - \mu_0)^2 + \lambda_2(\log(\hat{y}^{\mathrm{n}}(x)) - \mu_1)^2, \\ \mathcal{L}^2(x) &= \lambda_3(\log(\hat{y}^{\mathrm{p}}(x)) - \mu_0)^2 + \lambda_4(\log(\hat{y}^{\mathrm{n}}(x)) - \mu_1)^2, \\ \mathcal{L}^3(x) &= \lambda_5(\log(\hat{y}^{\mathrm{p}}(x)) - \mu_1)^2 + \lambda_6(\log(\hat{y}^{\mathrm{n}}(x)) - \mu_0)^2, \\ \mathcal{L}^4(x) &= \lambda_7(\log\hat{y}^{\mathrm{p}}(x) - \log\hat{y}^{\mathrm{n}}(x))^2, \\ \mathcal{L}^5 &= \lambda_8\left(|\log\mathcal{N}^{\mathrm{p}}| + |\log\mathcal{N}^{\mathrm{n}}| + |\log\mathcal{N}^{\mathrm{p}} - \log\mathcal{N}^{\mathrm{n}}|\right), \end{aligned} \tag{8}$$

and $\mathcal{D}^{\mathrm{l}}$ denotes positive samples with labels, $\mathcal{D}^{\mathrm{p,n}}$ denotes samples that are currently predicted to be positive/negative, $\mathcal{D}^{\mathrm{u}}$ denotes all unlabeled samples, and $|\mathcal{D}|$ denotes the number of samples in the set $\mathcal{D}$. The constants $\lambda_1, \lambda_2, \ldots \lambda_8$ are parameters of the model which have to be tuned, while the constants μ_0, μ_1 can be set to $\mu_0 = 5$, and $\mu_1 = -50$.

Batch generation In a typical PUL scenario, the number of labeled positive samples is much smaller than that of unlabeled samples. Hence, it is likely that many batches will not have any labeled samples during training. We solve this problem by adding a subsample (with repetition) of labeled positive samples. The number of added labeled samples equals the number of samples in the original batch, which contained randomly selected labeled and unlabeled samples.

Sampling A distinct feature of many tensor-network models is that they enable efficient sampling [8, 11]. We can efficiently sample also from a probability distribution encoded in an LPS state. In a tensor network, we typically sample sequentially by constructing a local probability density. We will assume that we have already sampled all positions until i. The values for the remaining positions $i, i+1, \dots N$ have not yet been determined. The probability density at the position i is then given by

$$p(x_i) = \operatorname{Tr} \mathcal{C}^1 \mathcal{C}^2 \dots \mathcal{C}^{i-1} \mathcal{P}(x_i) \tilde{\mathcal{C}}^{i+1} \tilde{\mathcal{C}}^N, \tag{9}$$

where

$$\mathcal{P}(x_i) = \begin{cases} \sum_{\alpha,\beta,\gamma=1}^{d} B^i_{a_1,b_1,\alpha,\gamma} \phi_\alpha(x_i) B^i_{a_2,b_2,\beta,\gamma} \phi_\beta(x_i) & i \in \mathcal{I} \\ \sum_{\alpha,\beta=1}^{d} A^i_{a_1,b_1,\alpha} \phi_\alpha(x_i) A^i_{a_2,b_2,\beta} \phi_\beta(x_i) & \text{else} \end{cases}. \tag{10}$$

Before sampling we have to normalise the probability density Eq. 9 to one. In a similar manner we can efficiently construct any marginal probability. Therefore, we can interpret the LPS as a compression of d^N coefficients of the estimated-probability series expansion (in the embedding basis functions). We can efficiently sample also from an LPS ensemble. At a given position, we first construct the probability densities corresponding to each LPS in the ensemble and then sample according to the average probability density. Finally we assign the sampled value to each LPS in the ensemble.

Our model is inspired by the tensor-network anomaly detection model [6] but has several key differences. First, we add a second tensor network that approximates the distribution of negative samples. Second, besides the loss for labeled samples, we add a loss for probably negative and probably positive samples as well as the "convergence" loss $\mathcal{L}^4$, which ensures that the model does not collapse to one class. Third, due to scarcity of labeled examples we have to modify the train batch construction to inject more labeled samples. Finally, as we will demonstrate in the next section, we can use the model to generate new positive and negative samples.

3 Results

We show results obtained on synthetic datasets and the MNIST dataset. The first we use to demonstrate the sampling and the second to demonstrate the accuracy of our model on an image dataset, which is typically challenging for tensor network models. In all cases we used the Adam optimizer and fixed the loss parameters to $\lambda_1 = \lambda_2 = \lambda_7 = 4$, $\lambda_4 = \lambda_5 = 2$, and $\lambda_3 = \lambda_6 = 1$. We fixed the loss hyperparameters after some manual tuning on the point datasets. The loss parameter λ_8 was changed dynamically in the range $\lambda_8 \in [0.5, 10]$. We increased λ_8 (up to the maximum value of 10) by a factor of 1.1 if all positive labeled samples were classified correctly. Similarly, we decreased λ_8 (up to the minimum value of 0.5) by a factor of 0.9 if less than 90% of the positive labeled samples were classified correctly.

Synthetic datasets We use three point datasets, namely the two moon dataset, the circles dataset and the blobs dataset available through the *scipy* API. In all cases, we use 1000 training samples where half of the samples are positive but only 100 are labeled. To obtain a more expressive model we first repeat the input nine times resulting in 18-dimensional feature vectors that are processed as discussed in Section 2. We use $S = 3$ and $D = d = 12$. We also randomly choose local basis functions between sine and cosine. We train the models with the Adam optimizer and a learning rate of 0.1. After we train the models, we sample from an ensemble by sequentially sampling from the average local probability density. Then we accept only samples for which all models predict a high probability that the sample is in the correct class, e.g. for positive samples we set the threshold to $\hat{y}^{\mathrm{p}} - \hat{y}^{\mathrm{n}} > 20$. Although the samples are already close to the original distribution, the thresholding further improves the sampled distribution by removing obvious outliers (see Fig. 1). By comparing the samples to the state-of-the-art GAN results [16, 18] it is visually clear that our method better reproduces the original distribution (especially after thresholding).

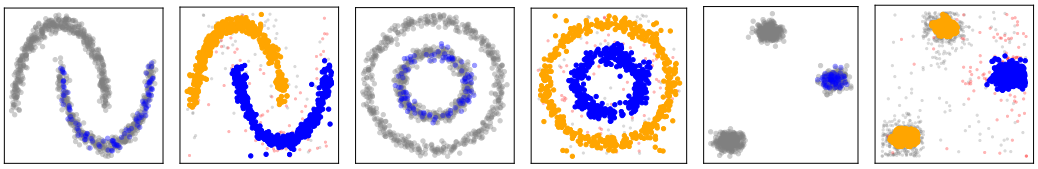

Figure 1: Samples generated by an ensemble of four models. We show the two moon dataset (first two panels), circles dataset (third and fourth panel), and blobs dataset (last two panels). First we show the training dataset, where gray denotes the unlabeled samples and blue the labeled positive samples. Then we show the generated samples, where orange circles are accepted positive samples, blue circles are accepted negative samples, gray and red dots are rejected negative and positive samples.

MNIST On the MNIST dataset, we performed the one-vs-one classification for each class pair and the one-vs-all classification for each of the ten classes in the dataset. We used the Adam optimizer with a learning rate of 0.01, cropped the images to size 20×20 pixels, and applied random rotation (for the angle 0.05π) and random zoom (with a factor in the range $[0.8, 1.2]$). For training and testing we used balanced datasets. We performed experiments with $N_{\mathrm{p}} = 100, 10, 1$ labeled positive samples. Finally, we fixed model parameters to $S = 10$, $d = 6$, and $D = 20$.

Before calculating the accuracy of the models, we perform a model selection step. After training several models we compare the predictions on the training dataset for each model pair and select the models which agree most. We find that the agreement fraction is a good estimate of the final test accuracy of the models (see Appendix A). We calculate the test accuracy only for selected models.

In the one-vs-one setting we obtain a mean accuracy of 0.986 ± 0.01 ($N_{\mathrm{p}} = 100$), 0.98 ± 0.04 ($N_{\mathrm{p}} = 10$), and 0.9 ± 0.2 ($N_{\mathrm{p}} = 1$). Which is significantly better as the GAN approaches which have the average test accuracy 0.89 ± 0.18 ($N_{\mathrm{p}} = 100$), 0.84 ± 0.18 ($N_{\mathrm{p}} = 10$), and 0.72 ± 0.23 ($N_{\mathrm{p}} = 1$) [18]. We report the results for the one-vs-rest task in Table 1. We find that the tensor network model has larger accuracy compared to the state-of-the-art GAN approaches [18].

Table 1: Average accuracy on one-vs-rest task on MNIST dataset. Largest accuracy is in bold.

	N_{p}	0	1	2	3	4	5	6	7	8	9	Avg.
TN	100	**0.99**	0.93	0.90	**0.90**	**0.93**	**0.91**	**0.98**	0.90	**0.91**	**0.88**	**0.92**
	10	0.98	0.95	**0.92**	0.90	0.91	0.83	**0.98**	0.92	0.80	0.87	0.91
	1	0.98	0.96	0.89	0.83	0.91	0.65	**0.98**	0.85	0.57	0.85	0.85
GAN	50	0.89	**0.97**	0.78	0.78	**0.93**	0.80	0.87	**0.93**	0.80	0.83	0.86

1-sample PUL clustering Since we obtain a good accuracy on most one-vs-rest classification tasks, we can utilise this to cluster a dataset in a completely unsupervised manner. First, we randomly select many (e.g. 100 in the MNIST case) samples, and then train a PUL model for each of the chosen samples. In the next step we calculate the agreement matrix between the labels of the models. This matrix provides the information on the number of classes and the best trained PUL models to construct a final classifier. Finally, we can cluster the dataset with the selected models.

4 Conclusion and broader impact

We propose a tensor-network approach to positive unlabeled learning problem and obtain state-of-the-art results on the MNIST image dataset. To date no tensor network approach outperformed best neural network methods on image datasets. However, to use the model on larger images, we would need to apply additional preprocessing, e.g. patch based embedding. Further, we expect tensor networks to achieve even better results on categorical datasets (work in progress) [6]. Finally, we propose a method to utilise 1-sample PUL models with inter-model label correlations as an unsupervised clustering method. Evaluation of this proposal is work in progress. Besides a broad applicability of the proposed model we believe that the 1-sample PUL clustering approach could be applicable also to other models and problems outside of the PUL setup.

Acknowledgments

The author acknowledges support from Sloveinan research agency (ARRS) project J1-2480. Computational resources were provided by SLING – Slovenian national supercomputing network. We thank Zoran Bosnić for reading the first version of the draft and providing useful comments.

References

[1] Jessa Bekker and Jesse Davis. Learning from positive and unlabeled data: A survey. *Machine Learning*, 109(4):719–760, 2020.

[2] Fantine Mordelet and Jean-Philippe Vert. Prodige: Prioritization of disease genes with multitask machine learning from positive and unlabeled examples. *BMC bioinformatics*, 12(1):1–15, 2011.

[3] Charles Elkan and Keith Noto. Learning classifiers from only positive and unlabeled data. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 213–220, 2008.

[4] Yashu Liu, Shuang Qiu, Ping Zhang, Pinghua Gong, Fei Wang, Guoliang Xue, and Jieping Ye. Computational drug discovery with dyadic positive-unlabeled learning. In *Proceedings of the 2017 SIAM international conference on data mining*, pages 45–53. SIAM, 2017.

[5] Yafeng Ren, Donghong Ji, and Hongbin Zhang. Positive unlabeled learning for deceptive reviews detection. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 488–498, 2014.

[6] Jinhui Wang, Chase Roberts, Guifre Vidal, and Stefan Leichenauer. Anomaly detection with tensor networks. *arXiv preprint arXiv:2006.02516*, 2020.

[7] Ivan Glasser, Ryan Sweke, Nicola Pancotti, Jens Eisert, and Ignacio Cirac. Expressive power of tensor-network factorizations for probabilistic modeling. *Advances in neural information processing systems*, 32, 2019.

[8] E Miles Stoudenmire and David J Schwab. Supervised learning with quantum-inspired tensor networks. *arXiv preprint arXiv:1605.05775*, 2016.

[9] Bojan Žunkovič. Deep tensor networks with matrix product operators. *Quantum Machine Intelligence volume*, 4(21), 2022.

[10] E Miles Stoudenmire. Learning relevant features of data with multi-scale tensor networks. *Quantum Science and Technology*, 3(3):034003, 2018.

[11] Zheng-Zhi Sun, Cheng Peng, Ding Liu, Shi-Ju Ran, and Gang Su. Generative tensor network classification model for supervised machine learning. *Physical Review B*, 101(7):075135, 2020.

[12] Raghavendra Selvan, Erik B Dam, Søren Alexander Flensborg, and Jens Petersen. Patch-based medical image segmentation using quantum tensor networks. *arXiv preprint arXiv:2109.07138*, 2021.

[13] Yanshan Xiao, Bo Liu, Jie Yin, Longbing Cao, Chengqi Zhang, and Zhifeng Hao. Similarity-based approach for positive and unlabelled learning. In *Twenty-second international joint conference on artificial intelligence*, 2011.

[14] Ke Zhou, Gui-Rong Xue, Qiang Yang, and Yong Yu. Learning with positive and unlabeled examples using topic-sensitive plsa. *IEEE Transactions on Knowledge and Data Engineering*, 22(1):46–58, 2009.

[15] Dino Ienco and Ruggero G Pensa. Positive and unlabeled learning in categorical data. *Neurocomputing*, 196:113–124, 2016.

[16] Ming Hou, Brahim Chaib-Draa, Chao Li, and Qibin Zhao. Generative adversarial positive-unlabelled learning. *arXiv preprint arXiv:1711.08054*, 2017.

[17] Florent Chiaroni, Mohamed-Cherif Rahal, Nicolas Hueber, and Frédéric Dufaux. Learning with a generative adversarial network from a positive unlabeled dataset for image classification. In *2018 25th IEEE International Conference on Image Processing (ICIP)*, pages 1368–1372. IEEE, 2018.

[18] Unpublished work under review. 2022.

A Accuracy estimate

In this appendix, we demonstrate that the fraction of matching labels between best-performing models is a reliable estimate of their accuracy. In Fig. 2, we show the difference between the estimated accuracy and actual test accuracy on the best models in the one-vs-one task. As expected, the larger the accuracy, the better the prediction. In fact, in most cases, we underestimate the actual test accuracy.

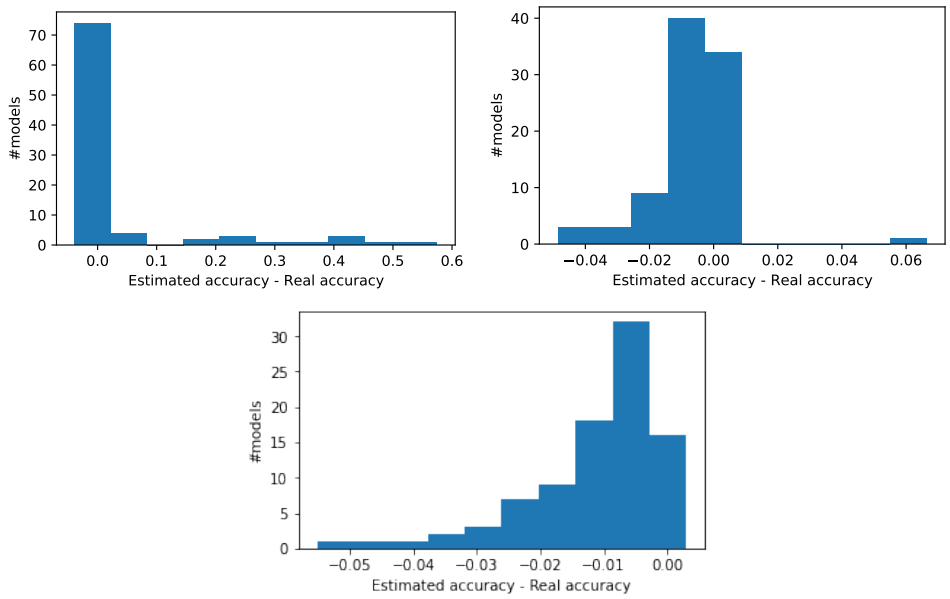

Figure 2: Histogram of differences between the estimated accuracy and the real test accuracy on the one-vs-one MNIST task. Top row from left to right we show results for $N_{\mathrm{p}} = 1, 10$, and in the bottom row for $N_{\mathrm{p}} = 100$.

B Compute Resources

All our experiments are performed on a compute cluster managed by Slurm Workload Manager. Each node has access to four NVidia A100 with 40 GB HBMI2. Estimated total compute time for presented experiments is 240 days.