Loopy Belief Propagation

David Iagaru[*1] DAVID.IAGARU@STUDENT-CS.FR

Jaonah Andriamampandry[†1] JAONAH.ANDRIAMAMPANDRY@ENS-PARIS-SACLAY.FR

[1] *ENS Paris-Saclay*

Editors: Under Review for the Graphical Models Discrete Inference course

Keywords: Belief Propagation, Graphical Models, Supervised Learning, Discrete Inference

1. Introduction

We implemented from scratch the Loopy Belief Propagation (Ihler et al., 2005) and the Supervised Belief Propagation algorithms described in (Yoo et al., 2017) applied to binary classification, using the iGraph library. We compared them on the Moovie-Lens dataset. It consists in a bipartite review network for movies from the users of MovieLens, whose edges contain integer ratings between 1 and 5. For the propagation, we focus on the sum-product algorithm and we operate on Markov Random Fields exclusively.

2. Methods

2.1. Loopy Belief Propagation (LBP)

The Loopy Belief Propagation is an approximate algorithm of the Belief Propagation, which works on cyclic graphs. The algorithm involves passing messages, and then computing beliefs to estimate the probability that a node is in a certain state. The algorithm is described in details in (Ihler et al., 2005; Yoo et al., 2017). (Yoo et al., 2017) describes it more consisely within the same frame work as our project.

2.2. Supervised Belief Propagation (SBP)

SBP applies on undirected graphs, when a part of the nodes are labelled for a classification. In order to reduce the complexity of the LBP, it is common to simplify the joint distribution as $\psi_{i,j}(x_i, x_j) = \epsilon \quad \text{if } x_i = x_j$ and $\psi_{i,j}(x_i, x_j) = 1 - \epsilon \quad \text{if } x_i \neq x_j$

Setting ϵ (named 'Propagation Strength') as an hyperparameter. Despite its simplicity, this approach does not provide an algorithmic way to determine the joint distrbution. SBP utilizes the information associated with each edge to determine a suitable propagation strength.

In the case of the classification within the Moovie-Lens dataset, a rating /5 is associated

[*] Contributed equally
[†] Contributed equally

with each edge (i, j). It is encoded in $\theta_{i,j} \in \{0, 1\}^6$ using the 1-hot encoding. The propagation strength is learned with $\epsilon_{i,j} = sigmoid(\theta_{i,j}^T w)$. Where $sigmoid(x) = \frac{1}{1+e^{-x}}$ and $w \in \mathbb{R}^6$ is the weight vector.

To use SBP, the nodes are first split into 2 disjoint sets : 'Training' nodes are used to optimize the weights and 'Observed nodes' will be used to perform propagation using the trained weights. Tests are performed over test nodes that are not directly linked to the ones used for the optimization of the weights. For those, the potential ϕ_i is set to $(0.5, 0.5)$ for every node i.

The whole algorithm is described in details in (Yoo et al., 2017)

3. Experiments

3.1. Implementation details

We implemented SBP on a 'training' graph graph and we tested it using the trained weights, in comparaison to LBP on a separate 'test' graph.
Since the nodes are not initially labelled in the original dataset, we first picked $k_2 = 20$ seed nodes. Starting with a seed node i, we assign labels to the remaining nodes as follows: items rated 5 by node i are labeled as positive, while items rated below 5 are labeled as negative as (Yoo et al., 2017) did. For the 'training' graph, we set the potential at 0.55 for labelled nodes and 0.5 for the others.
As for the hyperparameters, we set $\eta = 1 \quad d = 10^{-4} \quad \lambda = 0.5$.
The methods were performed on a a graph of varying sizes around 2000 nodes and 3000 edges.

3.2. Numerical considerations

In the SBP algorithm, it is necessary to divide by messages or beliefs. Without supplementary technique than described in the article, divisions by 0 can occur in the normalization of beliefs, messages and in the differentiation as well due to the limited machine precision. Before solving this problem, the computations crashed systematically for this reason.

- The first solution we implemented is to replace the division $\frac{a}{b}$ by $exp(log(a) - log(b))$, which accepts values closer to 0, since $log(10^{-100})$ is taken as $-100log(10)$ while the computer cannot divide directly by 10^{-100}. That decreased the crash-rate to 2/3

- The second modification is to reconsider the simplified formulas given in the paper. We considered that $\frac{b_j}{m_{i,j}}$ stands for $\frac{\prod_{l \in N(j) \setminus \{i\}} m_{l,j}}{b^*(s_n)+b^*(s_p)}$, analogously with $\frac{1-b_j}{1-m_{i,j}}$. We observed that it decreased the crash-rate from 1/2 to 1/3, which can be explained by the fact that the numerical errors don't come as often from the normalization. The bigger the graph, the more chances our implementation of the SBP has to crash, which limits the graph size that can be worked on.

These modifications made the implemented methods run with less errors despite an increased complexity of the algorithm, hence limiting the maximum size of used graphs for an acceptable inference time. Numerical overflows are still unavoidable, so it is hard to trust with 100% confidence, the results of the implemented SBP.

3.3. Weights Evolution

In order to monitor the learning process of our implementations, we looked into the evolution of the weights with respect to the number of iterations. The weight depicts which ratings are important and should be taken in account to propagate the message such that the beliefs corresponds to the labelled nodes.
According to the subsequent assignation of labelled nodes described in 3.1, we expect the weight associated with the rating 5 to be higher. On the other hand, one node connected to 2 seed node with a ratings of 5 and 1 respectively will be labelled positive as well, such that it will contribute to an increased weight associated to the rating 1. Considering the seed nodes are picked randomly, we expect a less predictable behaviour of the weights as well. Since the dataset should have some coherence (eg : 'bad' movies are rated worse in average, same for 'good' movies), there should the rating 5 should produce a bigger propagation strength. Within one execution, the convergence is fast and a clear order can be established between the ratings but the order often swaps due to the random initialization of seed nodes 1.

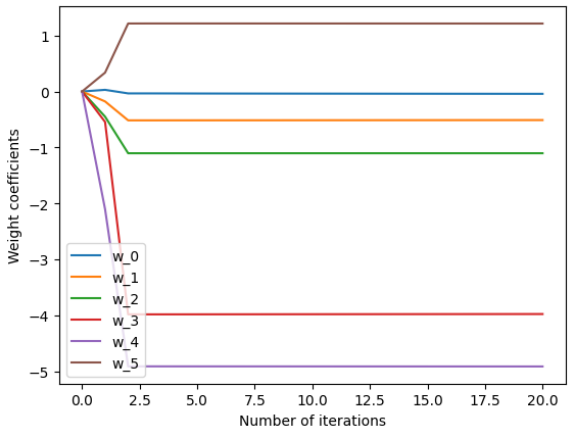

Figure 1: Weights evolution during the SBP training

3.4. Accuracy comparison

Here we show the AUC metric from our classification tests. The first one is a test of our LBP for several values of the propagation strength, which are $\{0.8, 0.7, 0.6, 0.51, 0.501, 0.5001\}$ in our case.

We can note that high propagation strengths gives us a worse accuracy in general, which suggests that we should take small propagation strength to have better results.

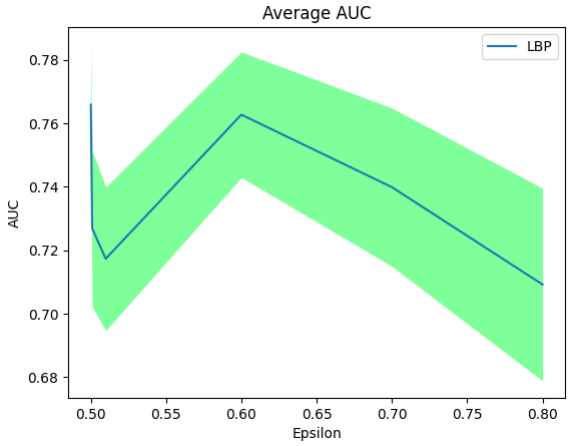

Figure 2: Average accuracy for different propagation strengths

To compare both models, we ran the models through the same seeds and we compute the accuracy of the LBP on the SBP train set everytime the SBP doesn't crash. In 20 iterations, SBP crashed 7 times, and we computed the average accuracy on the train set, along with it's incertitude computed using the standard deviation. We get :

$$\text{Accuracy of LBP} : 0.747\% \pm 0.041$$

$$\text{Accuracy of SBP} : 0.645\% \pm 0.043$$

Finally, we test the algorithms on a test set that is separated from the train data. However, we ran into some issues. Indeed, if the graph is too small, then finding a large enough test set that doesn't have an edge linking to the train data is rather hard. Also, if the test set is too small we won't have good results in terms of accuracy because the sample will be too small. The solution is to take a bigger graph, however this leads to a too high rate of crashes for SBP. We put the code that splits the graph into test set and train set in our Github repository.

4. Discussion

In conclusion, in the article we discovered an interesting algorithm. However, coding it from scratch proved to be difficult and the results are not exactly as stable as we wanted. For example, we would expect to have a SBP that performs better than LBP, but the average accuracy we found doesn't show this.

References

Alexander T. Ihler, John W. Fisher III, and Alan S. Willsky. Loopy belief propagation: Convergence and effects of message errors. *Journal of Machine Learning Research*, 6(31): 905–936, 2005. URL http://jmlr.org/papers/v6/ihler05a.html.

Jaemin Yoo, Saehan Jo, and U. Kang. Supervised belief propagation: Scalable supervised inference on attributed networks. pages 595–604, 2017. doi: 10.1109/ICDM.2017.69.