Regularization Methods for the Restricted Bayesian Network BESOM

Yuuji Ichisugi and Takashi Sano

Artificial Intelligence Research Center (AIRC),
National Institute of Advanced Industrial Science and Technology(AIST),
Tsukuba Central 1,Tsukuba,Ibaraki 305-8560, Japan

Abstract. We describe a method of regularization for the restricted Bayesian network BESOM, which possesses a network structure similar to that of Deep Learning. Two types of penalties are introduced to avoid overfitting and local minimum problems. The win-rate penalty ensures that each value in the nodes is used evenly; the lateral-inhibition penalty ensures that the nodes in the same layer are independent. Bayesian networks with these prior distributions can be converted into equivalent Bayesian networks without prior distributions, then the EM algorithm becomes easy to be executed.

1 Introduction

One of the remarkable hypotheses in the latest neuroscience is "the cerebral cortex is a kind of Bayesian network[4]." The cerebral cortex plays an important role in human intelligence. The cerebral cortex has many similarities to Bayesian networks[2], from the functional and structural point; this is suggested by a number of neuroscientific phenomena, well-simulated by the models involving Bayesian networks(For example [4–6, 8–10]).

Deep Learning, which stems from the Neocognitron[1], is garnering attention for its high recognitive performance. Neocognitron was designed to have the functionality of the visual cortex through the imitation of the hierarchical structure of ventral pathway of the cortex.

Combining the latest neuroscientific insights and the Deep Learning technology will lead to the better performing machine learning technology, which has the more human-like ability. With this goal in mind, we are developing a machine-learning algorithm called BESOM(BidirEctional Self-Organizing Map), a Bayesian network with a layer structure and each node has restricted CPT (Conditional Probability Table) model[6]. Though our BESOM algorithm is under development and lacks accuracy, it already has the ability to show the potential applications in engineering and as a possible computational model of the cerebral cortex[6, 8, 11].

Deep Learning using a Bayesian network is thought to be promising not only because of its similarity to the human brain but also from a technical viewpoint, particularly with respect to the following points:

Yuuji Ichisugi and Takashi Sano,
Regularization Methods for the Restricted Bayesian Network BESOM,
In Proc. of International Conference on Neural Information Processing (ICONIP 2016) Part I, LNCS 9947, pp.290--299, 2016.

The final publication is available at Springer via
https://dx.doi.org/10.1007/978-3-319-46687-3_32

- Inference in Bayesian networks can sometime be executed with low computational complexity.
- Because Bayesian networks have top-down information flow in addition to bottom-up, they may be more powerful than feed-forward neural networks.
- It is easy to build in prior knowledge about learning targets.

Despite these advantages, large-scale Bayesian networks like BESOM are not widely used, probably because of their large computational complexity, overfitting and local minima problems.

For the issue of computational complexity, efforts are being addressed by the use of restricted CPTs[3, 11].

The problems of overfitting and local minima are thought to arise from the high expressiveness of large-scale Bayesian networks. Assigning an adequate prior distribution to the parameters, this high expressiveness would be reasonably lowered to solve these problems.

In this study, we describe two types of prior distribution: the win-rate penalty and the lateral-inhibition penalty. We also introduce an approximate learning rule for use with these penalties. The two mechanisms can be applied simultaneously. They add biases to the recognition results: the win-rate penalty ensures that each value in the nodes is used evenly; the lateral-inhibition penalty ensures that the nodes in the same layer are independent.

2 The Architecture of BESOM

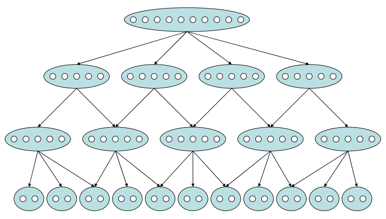

Fig. 1. An example of a BESOM network. Ovals are nodes (random variables) and the white circles inside are units (possible values for the random variables).

BESOM is a Bayesian network having a deep hierarchical structure similar to Deep Learning(Figure1). Like many Deep Learning architectures, it has connections between layers forming local receptive fields, while there are no connections in the same layer.

In BESOM, variables are called *nodes* and possible values for the variables are called *units*. In general, nodes are multinomial variables. If a black and white image is to be learned, the input pixel values are given as the observed binary variables in the bottom layer.

BESOM can be used in both unsupervised and supervised learning. When used for unsupervised learning, all variables in BESOM are hidden, except those in the bottom layer. In this case, acquired features are expressed in the upper layers. For supervised learning by BESOM, there are ways to assign the supervisory signal. One of the way, for example, is to assign the supervisory signal to a single node in the uppermost layer. In the test phase, the uppermost node becomes a hidden variable, whose inference value (the maximum posterior probability) is taken to be the recognition result.

Another significant feature of BESOM is the limitation placed on CPT, which will be explained in Section 4.

3 The Objective Function for Learning

Let $P(\mathbf{h}, \mathbf{i}|\theta)$ be a joint probability model with the a set of hidden variables $\mathbf{h}$ and the set of input variables $\mathbf{i}$, with given parameter θ. By $\mathbf{i}(t)$, we give the set of values of input variables at the time t. Under the assumption that the input data sequence forms i.i.d. (independent and identical distributions) for fixed parameter θ, the probability for the input data sequence $\mathbf{i}(1), \mathbf{i}(2), \cdots, \mathbf{i}(t)$ occurring under the parameter θ, which is likelihood of θ, is calculated like this:

$$P(\mathbf{i}(1), ..., \mathbf{i}(t) \mid \theta) = \prod_{i=1}^{t} P(\mathbf{i}(i) \mid \theta) = \prod_{i=1}^{t} \sum_{\mathbf{h}} P(\mathbf{h}, \mathbf{i}(i) \mid \theta)\,. \tag{1}$$

The objective of learning is to obtain MAP(maximum a posteriori) estimate of the parameter. In other words, the objective is to find maximizing parameter of θ, say θ^*:

$$\theta^* = \operatorname*{argmax}_{\theta} \left[\prod_{i=1}^{t} \sum_{\mathbf{h}} P(\mathbf{h}, \mathbf{i}(i) \mid \theta) \right] P(\theta). \tag{2}$$

To estimate parameter θ, the online EM (Expectation-Maximization) algorithm or its approximation is used. One method of approximation is given as follows.

The approximation algorithm here is combination of two steps, one for recognition and the other for learning. First, in the recognition step, based on current parameter $\theta(t)$ and given the input values $\mathbf{i}(t)$, the maximum posterior probability estimation values of the hidden variables $\hat{\mathbf{h}}(t)$ (i.e., MPE, Most Probable Explanation) are obtained as follows:

$$\begin{aligned} \hat{\mathbf{h}}(t) &= \operatorname*{argmax}_{\mathbf{h}} P(\mathbf{h}|\mathbf{i}(t), \theta(t)) = \operatorname*{argmax}_{\mathbf{h}} \frac{P(\mathbf{h}, \mathbf{i}(t)|\theta(t))}{P(\mathbf{i}(t))} \\ &= \operatorname*{argmax}_{\mathbf{h}} P(\mathbf{h}, \mathbf{i}(t)|\theta(t)). \end{aligned} \tag{3}$$

Approximate calculation of this formula can be efficiently executed by, for example, loopy belief revision algorithm[8].

Next, in the learning step, the marginalization of the hidden variables in Equation (2) is approximated using the estimated value $\hat{\mathbf{h}}(i)$ and the result is taken to be $\theta(t+1)$.

$$\theta(t+1) = \underset{\theta}{\operatorname{argmax}} \left[\prod_{i=1}^{t} P(\hat{\mathbf{h}}(i), \mathbf{i}(i)|\theta) \right] P(\theta). \tag{4}$$

The prior distribution for parameter θ is defined as the product of two factors, as follows:

$$P(\theta) = P^{\mathrm{WinRate}}(\theta)\ \ P^{\mathrm{Lateral}}(\theta). \tag{5}$$

The detailed explanation of the win-rate penalty $P^{\mathrm{WinRate}}(\theta)$ and and lateral-inhibition penalty $P^{\mathrm{Lateral}}(\theta)$, are given in Section 5 and Section 6, respectively.

4 The Conditional Probability Table Model

One important characteristic of BESOM is in its CPT model. (Note that the win-rate penalty and lateral-inhibition penalty mechanisms, which are the main subject of this paper, are thought to work with the other types of CPT models.)

In a Bayesian network an $O(2^m)$ number of parameters is generally needed with respect to m, the number of parent nodes, to express the CPT for each node. This causes an explosive increase in computational complexity and memory requirements as well as introducing the problem of overfitting and local minima.

To allow CPTs to be expressed with fewer parameters, we limited them in the following manner:

$$P(x|u_1, \cdots, u_m) = \frac{1}{m} \sum_{k=1}^{m} w(x, u_k). \tag{6}$$

As the simplest form of $w(x, u_k)$, we currently use the following:

$$w(x, u_k) = P(x|u_k). \tag{7}$$

In this case, the conditional probability $P(x|u_k)$ is expressed by a single parameter w_{xu_k} .

When this restrictions are introduced, the belief propagation algorithm can be optimized and computational complexity is dramatically reduced[11]. It has also been shown that the information flow between the nodes closely matches the characteristic anatomical structure of the cerebral cortex[6, 8].

5 Win-Rate Penalty

5.1 Purpose

If the BESOM network parameters are learned in a naive way, learning progresses for only a small set of units and the other units tend to stay at their initial values. In this case, learning is thought to fall into a local minimum.

An effective way to address this problem is to set the prior distribution for the parameter appropriately, and assign a bias so that the units in each node are used evenly.

Our approach uses the Kullback-Leibler (KL) divergence between the win-rate distribution that is targeted by each unit and the actual distribution. Penalties are imposed when the divergence is large. Here, the win-rate refers to the frequency with which units become the estimated values for the node.

In BESOM, the units are values of random variables, and the unit corresponding to the estimated value in one node (a random variable) is called the *winner unit.*

Using this penalties, it is expected that as learning progresses, the win-rate of each unit will approach the target value. The mechanism is called the *win-rate penalty* because units with larger win-rates are penalized.

5.2 The Problem of Complex Prior Distributions and its Solution

The maximum likelihood estimate for the parameters of a Bayesian network with hidden variables can be estimated using an EM algorithm which can be executed efficiently using the result of inference[7]. However, when the parameter has a complex prior distribution, it is not obvious to perform the EM algorithm efficiently.

If a Bayesian network with a prior distribution for its parameter can be converted into an equivalent Bayesian network without a prior distribution, then the EM algorithm will become easy to be executed.

Fortunately, a Bayesian network with a prior distribution describe below can be converted into an approximately equivalent Bayesian network with no prior distribution. In the converted Bayesian network, *restriction nodes* are added to give bias to the recognition result.

5.3 Defining a Prior Distribution, and Deriving an Equivalent Bayesian Network

The win-rate penalty $P^{\mathrm{WinRate}}(\theta)$ is defined as follows:

$$P^{\mathrm{WinRate}}(\theta) = \prod_{X \in \mathbf{X}} \exp(-C^{\mathrm{WinRate}} D_{KL}(Q(X) || P(X; \theta))). \tag{8}$$

where $\mathbf{X}$ is the set of all nodes and C^{WinRate} is a constant that determines the strength of the win-rate penalty.

$Q(X)$ is the distribution set as the target for the win-rate of node X and the network architect decides the shape of this distribution. For example, if the goal is to make the win-rate of the node units uniform, $Q(X)$ is defined for all units x_i $(i = 1, 2, \cdots, s)$ as

$$Q(X = x_i) = 1/s \tag{9}$$

where s represents the number of units in node X.

The Kullback-Leibler divergence between distributions $Q(X)$ and $P(X;\theta)$ is defined by the following equation:

$$D_{KL}(Q(X)||P(X;\theta)) = \sum_x Q(x) \log \frac{Q(x)}{P(x;\theta)}. \tag{10}$$

We define a function $R(x;\theta)$ as follows: (*)See Errata

$$R(x;\theta) = \frac{Q(x)}{P(x;\theta)} \log \frac{Q(x)}{P(x;\theta)}. \tag{11}$$

We can expect the following approximation holds because $x \sim P(x;\theta)$:

$$\sum_x f(x;\theta) \approx \sum_{i=1}^{t} \frac{1}{t} \frac{1}{P(x(i);\theta)} f(x(i);\theta). \tag{12}$$

Therefore,

$$\begin{aligned} D_{KL}(Q(X)||P(X;\theta)) &= \sum_x Q(x) \log \frac{Q(x)}{P(x;\theta)} \\ \approx \sum_{i=1}^{t} \frac{1}{t} \frac{1}{P(x(i);\theta)} Q(x(i)) \log \frac{Q(x(i))}{P(x(i);\theta)} &= \sum_{i=1}^{t} \frac{1}{t} R(x(i);\theta) \end{aligned} \tag{13}$$

holds. Then, $P^{\mathrm{WinRate}}(\theta)$ can then be rewritten as follows:

$$\begin{aligned} P^{\mathrm{WinRate}}(\theta) &= \prod_{X \in \mathbf{X}} \exp(-C^{\mathrm{WinRate}} D_{KL}(Q(X)||P(X;\theta))) \\ &\approx \prod_{X \in \mathbf{X}} \exp(-C^{\mathrm{WinRate}} \sum_{i=1}^{t} \frac{1}{t} R(x(i);\theta)) \\ &= \prod_{X \in \mathbf{X}} \prod_{i=1}^{t} \exp(-\frac{1}{t} C^{\mathrm{WinRate}} R(x(i);\theta)) \\ &= \prod_{i=1}^{t} \prod_{X \in \mathbf{X}} \exp(-\frac{1}{t} C^{\mathrm{WinRate}} R(x(i);\theta)). \end{aligned} \tag{14}$$

Equation (4), which MAP estimates parameter θ in the learning step, can be rewritten as follows (for simplicity, $P(\theta) = P^{\mathrm{WinRate}}(\theta)$ is assumed here):

$$\begin{aligned} \theta(t+1) &= \operatorname*{argmax}_{\theta} \left[\prod_{i=1}^{t} P(\hat{\mathbf{h}}(i), \mathbf{i}(i)|\theta) \right] P^{\mathrm{WinRate}}(\theta) \\ &= \operatorname*{argmax}_{\theta} \left[\prod_{i=1}^{t} \prod_{X \in \mathbf{X}} P(x(i)|\mathrm{pa}(x(i));\theta) \right] \left[\prod_{i=1}^{t} \prod_{X \in \mathbf{X}} \exp(-\frac{1}{t} C^{\mathrm{WinRate}} R(x(i);\theta)) \right] \\ &= \operatorname*{argmax}_{\theta} \prod_{i=1}^{t} \prod_{X \in \mathbf{X}} \left[P(x(i)|\mathrm{pa}(x(i));\theta) \exp(-\frac{1}{t} C^{\mathrm{WinRate}} R(x(i);\theta)) \right] \end{aligned} \tag{15}$$

where $\mathrm{pa}(x(i))$ represents the values of the parent nodes of node X at time i .

The above equation can be interpreted as that each node X has a corresponding restriction node R_X (Figure 2) whose conditional probability is defined as follows:

$$P(R_X = 1|X = x; \theta) = \exp(-\frac{1}{t} C^{\mathrm{WinRate}} R(x; \theta)). \tag{16}$$

(Node R_X always has a observed value 1 .) The Bayesian network with restriction nodes no longer has prior, therefore, it is possible to conduct parameter estimation using an EM algorithm.

Because C^{WinRate} is multiplied with the regularization parameter $1/t$, the penalty's influence decreases as time progresses when online learning.

The value $P(x; \theta)$ needed to calculate $R(x; \theta)$ can be simply estimated using statistics of values.

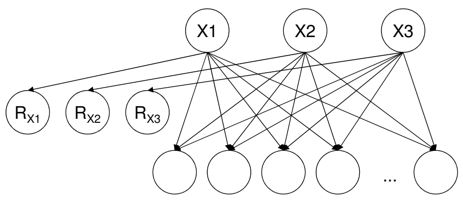

Fig. 2. Restriction nodes representing the win-rate penalty.

6 Lateral-Inhibition Penalty

6.1 Purpose

When the BESOM network parameters are learned in a naive way, nodes in the same layer that receive inputs from the same child nodes tend to represent similar feature. This phenomenon is also thought to relate to local minima or overfitting problems. Each of the hidden layers in BESOM, similar to those in Deep Learning, is expected to work as a feature extractor. When the same feature are redundantly expressed in many nodes, it is not preferable for recognition in the upper layers.

As in the previous section, this problem is addressed by defining a penalty as a prior distribution, and then, an equivalent Bayesian network without a prior is derived. In the prior distribution, a bias is applied in such a way that by assigning penalties to cases in which two nodes express similar values. Because this mechanism is thought to have a similar role to the lateral inhibition mechanism in the cerebral cortex, we name it the *lateral-inhibition penalty*.

6.2 Defining a Prior Distribution, and Deriving an Equivalent Bayesian Network

The prior $P^{\text{Lateral}}(\theta)$ corresponding to lateral inhibition is defined as

$$P^{\text{Lateral}}(\theta) = \prod_{(U,V)\in L} \exp(-C^{\text{Lateral}} I(U,V;\theta)) \tag{17}$$

where C^{Lateral} is the constant which determines the strength of the lateral-inhibition penalty and L are the set of pairs of nodes conducting lateral inhibition. Usually, each pair of nodes in the same layer that share child nodes should laterally inhibit each other.

$I(U,V;\theta)$ is the mutual information between nodes U and V and is defined as follows:

$$I(U,V;\theta) = \sum_u \sum_v P(u,v;\theta) \log \frac{P(u,v;\theta)}{P(u;\theta)P(v;\theta)}. \tag{18}$$

Here, we define a function $R(u,v;\theta)$ as follows(θ has been omitted):

$$R(u,v) = \frac{P(u,v)}{P(u)P(v)} \log \frac{P(u,v)}{P(u)P(v)} = (P(u|v)/P(u)) \log P(u|v)/P(u). \tag{19}$$

Given these definitions and the approximate equation (12), the following holds:

$$I(U,V;\theta) \approx \sum_{i=1}^{t} \frac{1}{t} R(u(i), v(i); \theta). \tag{20}$$

Following the same logic as in Section 5, we can derive an equivalent Bayesian network without the prior. In the network, for each node pair $(U,V) \in L$ that displays lateral inhibition, there is a shared binary-valued child node R_{UV} (Figure 3) whose conditional probability is defined as follows:

$$P(R_{UV} = 1|u,v;\theta) = \exp(-\frac{1}{t} C^{\text{Lateral}} R(u,v;\theta)). \tag{21}$$

Thus, the maximum likelihood value of a parameter can be easily estimated using an EM algorithm.

The values $P(u|v;\theta)$ and $P(u;\theta)$ required to calculate the value of $R(u,v;\theta)$ can be simply estimated using statistics of values.

7 Evaluation

We evaluated the effectiveness of the proposed method using recognition rates of an MNIST handwritten digit database[1].

[1] `http://yann.lecun.com/exdb/mnist/`

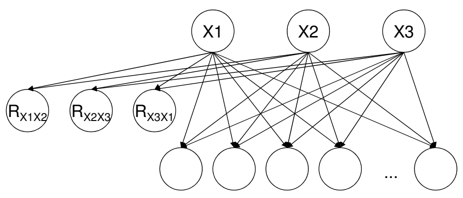

Fig. 3. Restriction nodes representing the lateral-inhibition penalty.

We used a four-layer BESOM network for this experiment. For the bottom layer, we created a 28 x 28 layout of two-unit nodes to take binary pixel values from a 28 x 28 input images. For the uppermost layer, we used a single 10-unit node to provide the supervisory signal. There were two hidden layers: for the layer immediately above the input layer we created a 5 x 5 array of 20-unit nodes and for the layer above that, a 3 x 3 layout of 30-unit nodes.

We evaluated the recognition rate by having the network first randomly learn 10,000 pieces of training data from a possible 60,000 pieces and then randomly recognize 1,000 pieces from 10,000 pieces of test data.

For learning, a very rough approximation of an online EM algorithm was used. First, an optimized loopy belief propagation algorithm[11] was applied. This was used to calculate the marginal posterior probabilities for each node. For each node, the value with maximum posterior was taken to be its estimated value, and the parameter was updated using the value.

Table 1 summarizes the results; each value is the average of 10 experiments. For both penalties, the recognition rate was higher than when no penalties were applied. This result also shows that two prior distribution can be applied simultaneously; however, it does not show the best accuracy in this case.

Table 1. Accuracy of recognition results of MNIST hand-written digits. (WR: Win-Rate penaltiy, LI: Lateral-Inhibition penaltiy)

	With WR	Without WR
With LI	80.6 %	81.8 %
Without LI	82.2 %	63.6 %

8 Conclusion and Future Work

Two regularization methods for parameter learning of layered Bayesian networks are proposed and an experiment shows that they are promising. We believe they alleviate both overfitting and local minima problems; however, more detailed evaluation and analysis may still be required.

BESOM is beginning to be used as a machine-learning algorithm; however, sufficient recognition precision has not been attained to enable its use in practical applications. The main reason for this is thought to be that the restrictions of CPTs described in Section 4 are too strong. To address this problem, it is necessary to develop a new CPT model and a suitable approximate belief propagation algorithm. This is what we are currently working on.

Acknowledgements

This paper is based on results obtained from a project commissioned by the New Energy and Industrial Technology Development Organization (NEDO).

References

1. K. Fukushima, Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. Biological Cybernetics, 36(4): 93-202, 1980.
2. J. Pearl , Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference, Morgan Kaufmann, 1988.
3. D. Heckerman, Causal independence for knowledge acquisition and inference. In Proc. of UAI 1993, pp. 122–127, 1993.
4. Lee, T.S., Mumford, D. , Hierarchical Bayesian inference in the visual cortex. Journal of Optical Society of America, A. . 20(7): 1434-1448, 2003.
5. George, D. Hawkins, J., A hierarchical Bayesian model of invariant pattern recognition in the visual cortex, In proc. of IJCNN 2005, vol. 3, pp.1812–1817, 2005.
6. Yuuji Ichisugi, The cerebral cortex model that self-organizes conditional probability tables and executes belief propagation, In Proc. of IJCNN 2007, pp.1065–1070, 2007.
7. Koller, D. and Friedman, N., Probabilistic Graphical Models: Principles and Techniques - Adaptive Computation and Machine Learning, The MIT Press, 2009.
8. Yuuji Ichisugi, "Recognition Model of Cerebral Cortex based on Approximate Belief Revision Algorithm", In Proc. of IJCNN 2011, pp.386–391, 2011.
9. Haruo Hosoya, Multinomial Bayesian Learning for Modeling Classical and Nonclassical Receptive Field Properties, Neural Computation,Vol. 24, No. 8, pp. 2119–2150, 2012.
10. Salvador Dura-Bernal, Thomas Wennekers, Susan L. Denham, Top-Down Feedback in an HMAX-Like Cortical Model of Object Perception Based on Hierarchical Bayesian Networks and Belief Propagation PLoS ONE, Vol.7, No.11., 2012.
11. Yuuji Ichisugi and Naoto Takahashi, An Efficient Recognition Algorithm for Restricted Bayesian Networks, In Proc. of IJCNN 2015, pp.1–6, 2015.

Errata (2017-02-15)

Equation (5): Normalization constant is required.

Equation (8),(11):

Our current implementation, which was used for the evaluation in Section 7, uses $D_{KL}(P(X;\theta)||Q(X))$ instead of $D_{KL}(Q(X)||P(X;\theta))$.

When we use $D_{KL}(P(X;\theta)||Q(X))$ for the definition of the equation (8), the equation (11) becomes

$$R(x;\theta) = \log \frac{P(x;\theta)}{Q(x)}$$

and it can be approximated as

$$R(x;\theta) \approx \frac{P(x;\theta)}{Q(x)} - 1.$$

The effect of this penalty is easy to understand intuitively. If $P(x;\theta)$ is larger than the target value Q(x), a larger penalty is given to the value x, making x less likely to be chosen as the inference result.

Conversely, if we use the definition of the penalty based on $D_{KL}(Q(X)||P(X;\theta))$ as written in this paper, the difference in the win-rate will become more intense. Because we do not understand why $D_{KL}(P(X;\theta)||Q(X))$ and $D_{KL}(Q(X)||P(X;\theta))$ cause different effect, we will continue to investigate this issue.

Equation (16),(21):

The values of these equations may exceed 1 because the values of $R(x;\theta)$ and $R(u,v;\theta)$ may become minus values. We can avoid this formal problem by just multiply the right hand sides of these equations by a sufficiently small constant value δ, for example:

$$P(R_X = 1|X = x;\theta) = \delta \exp\left(-\frac{1}{t} C^{WinRate} R(x;\theta)\right).$$

Because the δ will disappear when normalizing messages, we can ignore it when implementing this algorithm.