# The Serial Algorithm of Learning via Factor Graph

**Xiaowen Ding**
Computer Science Department
Carnegie Mellon University
xiaowend@andrew.cmu.edu

**Yu Zhao**
Computer Science Department
Carnegie Mellon University
yuzhao1@andrew.cmu.edu

## 1 Model Description

Here we denote the label vector of all nodes as $Y = \{y_1, y_2, \ldots, y_M\}$, where $y_i$ represents the label of node $i$.

In our factor graph, we have two kinds of factors: Attribute factors and Correlation factors.

- Attribute factor: $f(y_i, \mathbf{x}_i)$ represents the posterior probability of the label $y_i$ given the attribute(feature) vector $\mathbf{x}_i$. This is the factor of only one node.
- Correlation factor: $g(y_i, \text{adj}(y_i))$ denotes the correlation between the nodes, where $\text{adj}(y_i)$ is the set of nodes with edges to $y_i$. This is the factor between two nodes.

Given a factor graph $G = (V, E)$, we can define the joint distribution over $Y$ as

$$p(Y|G) = \prod_i f(y_i, \mathbf{x}_i) g(y_i, \text{adj}(y_i))$$

These two factors can be instantiated in different ways. Here we use exponential-linear functions. In particular, we define the attribute factor as

$$f(y_i, \mathbf{x}_i) = \frac{1}{Z_\alpha} \exp(\alpha^T \phi(y_i, \mathbf{x}_i))$$

where $\alpha$ is a weighting vector and $\phi$ is a vector of indicator feature functions. $Z_\alpha$ is a normalization factor. Similarly, we define the correlation factor as

$$g(y_i, \text{adj}(y_i)) = \frac{1}{Z_\beta} \exp(\sum_{y_j \in \text{adj}(y_i)} \beta^T \chi(y_i, y_j))$$

where $\beta$ is a weighting vector, $\chi$ can be defined as a vector of indicator functions and $Z_\beta$ is a normalization factor.

Combine these two factors, the total parameter configuration is $\theta = (\alpha, \beta)$. For presentation simplicity, we concatenate all factor functions for a node $i$ with label $y_i$ as $mathbfs(y_i) = (\phi(y_i, \mathbf{x}_i)^T, \sum_{y_j} \chi(y_i, y_j)^T)^T$. Then the joint probability can be written as

$$p(Y|G) = \frac{1}{Z} \prod_i \exp(\theta^T \mathbf{s}(y_i)) = \frac{1}{Z} \exp(\theta^T \sum_i \mathbf{s}(y_i)) = \frac{1}{Z} \exp(\theta^T \mathbf{S}(Y))$$

where $Z = Z_\alpha Z_\beta$ is the total normalization factor, $\mathbf{S}(Y)$ is the aggregation of factor functions over all nodes, i.e., $\mathbf{S}(Y) = \sum_i \mathbf{s}(y_i)$.

## 2 Learning Algorithm

We want to estate a parameter configuration $\theta$ so that the log-likelihood of observation information $\mathcal{O}(\theta) = \log p(Y^*|G)$ is maximized.

$$\begin{aligned}\mathcal{O}(\theta) = \log p(Y^*|G) &= \log\left(\frac{1}{Z}\exp(\theta^T\mathbf{S}(Y^*))\right) \\ &= \theta^T\mathbf{S}(Y^*) - \log Z \\ &= \theta^T\mathbf{S}(Y^*) - \log\sum_Y \exp(\theta^T\mathbf{S}(Y))\end{aligned}$$

Here we want to use the gradient descent method to optimize this objective function. Specifically, we need to calculate the gradient for parameter $\theta$:

$$\begin{aligned}\frac{\partial\mathcal{O}(\theta)}{\partial\theta} &= \frac{\partial\left(\theta^T\mathbf{S}(Y^*) - \log\sum_Y\exp(\theta^T\mathbf{S}(Y))\right)}{\partial\theta} \\ &= \mathbf{S}(Y^*) - \frac{\sum_Y \exp\theta^T\mathbf{S}(Y)\cdot\mathbf{S}(Y)}{\sum_Y\exp\theta^T\mathbf{S}(Y)} \\ &= \mathbf{S}(Y^*) - \mathbb{E}_{p_\theta(Y|G))}\mathbf{S}(Y)\end{aligned}$$

A challenge here is that the graphical structure in this factor graph can be arbitrary and may contain cycles, which makes it intractable to directly calculate the second expectation $\mathbb{E}_{p_\theta(Y|G))}\mathbf{S}(Y)$. A number of approximate algorithms have been proposed, such as Loopy Belief Propagation. Here we use LBP to approximate the marginal probabilities. We need to perform the LBP process in each iteration of the gradient descent. Finally with the gradient, we update each parameter with a learning rate $\eta$. The learning algorithm is summarized in Algorithm 1.

Initialize parameter $\theta$;
**repeat**
  Calculate $\mathbb{E}_{p_\theta(Y|G)}\mathbf{S}(Y)$ using Loopy Belief Propagation;
  Calculate the gradient of $\theta$:

$$\nabla_\theta = \mathbf{S}(Y^*) - \mathbb{E}_{p_\theta(Y|G)}\mathbf{S}(Y)$$

  Update parameter $\theta$ with the learning rate $\eta$:

$$\theta_{\text{new}} = \theta_{\text{old}} - \eta\cdot\nabla_\theta$$

**until** *convergence* ;

**Algorithm 1:** Learning Algorithm