Appendix

A. More Related Work

A.1. Comparison with Neural Networks on Graphs

Neural network is also a powerful tool on graph structured data. Scarselli et al. (2009) proposed a neural network which generates features by solving a heuristic nonlinear system iteratively, and is learned using Almeida-Pineda algorithm. To guarantee the existence of the solution to the nonlinear system, there are extra requirements for the features generating function. From this perspective, the model in (Li et al., 2015) can be considered as an extension of (Scarselli et al., 2009) where the gated recurrent unit is used for feature generation. Rather than these heuristic models, our model is based on the principled graphical model embedding framework, which results in flexible embedding functions for generating features. Meanwhile, the model can be learned efficiently by traditional stochastic gradient descent.

There are several work transferring locality concept of convolutional neural networks (CNN) from Euclidean domain to graph case, using hierarchical clustering, graph Laplacian (Bruna et al., 2013), or graph Fourier transform (Henaff et al., 2015). These models are still restricted to problems with the same graph structure, which is not suitable for learning with molecules. (Mou et al., 2016) proposed a convolution operation on trees, while the locality are defined based on parent-child relations. (Duvenaud et al., 2015) used CNN to learn the circulant fingerprints for graphs from end to end. The dictionary of fingerprints are maintained using softmax of subtree feature representations, in order to obtain a differentiable model. If we unroll the steps in Algorithm 3, it can also be viewed as an end to end learning system. However, the structures of the proposed model are deeply rooted in graphical model embedding, from mean field and loopy BP, respectively. Also, since the parameters will be shared across different unrolling steps, we would have more compact model. As will be shown in the experiment section, our model is easy to train, while yielding good generalization ability.

A.2. Comparison with Learning Message Estimator

By recognizing inference as computational expressions, inference machines (Ross et al., 2011) incorporate learning into the messages passing inference for CRFs. More recently, Hershey et al. (2014); Zheng et al. (2015); Lin et al. (2015) designed specific recurrent neural networks and convolutional neural networks for imitating the messages in CRFs. Although these methods share the similarity, *i.e.*, bypassing learning potential function, to the proposed framework, there are significant differences comparing to the proposed framework.

The most important difference lies in the learning setting. In these existing messages learning work (Hershey et al., 2014; Zheng et al., 2015; Lin et al., 2015; Chen et al., 2014), the learning task is still estimating the messages represented graphical models with designed function forms, *e.g.*, RNN or CNN, by maximizing loglikelihood. While in our work, we represented each structured data as a distribution, and the learning task is regression or classification over these distributions. Therefore, we treat the embedded models as samples, and learn the nonlinear mapping for embedding, and regressor or classifier, $f : \mathcal{P} \to \mathcal{Y}$, over these distributions jointly, with task-dependent user-specified loss functions.

Another difference is the way in constructing the messages forms, and thus, the neural networks architecture. In the existing work, the neural networks forms are constructed *strictly* follows the message updates forms (8) or (12). Due to such restriction, these works only focus on discrete variables with finite values, and is difficult to extend to continuous variables because of the integration. However, by exploiting the embedding point of view, we are able to build the messages with more *flexible* forms without losing the dependencies. Meanwhile, the difficulty in calculating integration for continuous variables is no longer a problem with the reasoning (3) and (4).

B. Derivation of the Fixed-Point Condition for Loopy BP

The derivation of the fixed-point condition for loopy BP can be found in (Yedidia et al., 2001b). However, to keep the paper self-contained, we provide the details here. The objective of loopy BP is

$$\begin{aligned}\min_{\{q_{ij}\}_{(i,j)\in\mathcal{E}}} \quad & -\sum_i (|\mathcal{N}(i)|-1)\int_{\mathcal{H}} q_i(h_i)\log\frac{q_i(h_i)}{\Phi(h_i,x_i)}dh_i + \sum_{i,j}\int_{\mathcal{H}^2} q_{ij}(h_i,h_j)\log\frac{q_{ij}(h_i,h_j)}{\Psi(h_i,h_j)\Phi(h_i,x_i)\Phi(h_j,x_j)}dh_i dh_j \\ \text{s.t.} \quad & \int_{\mathcal{H}} q_{ij}(h_i,h_j)dh_j = q_i(h_i), \quad \int_{\mathcal{H}} q_{ij}(h_i,h_j)dh_j = q_i(h_i), \quad \int_{\mathcal{H}} q_i(h_i)dh_i = 1.\end{aligned}$$

Denote $\lambda_{ij}(h_j)$ is the multiplier to marginalization constraints $\int_{\mathcal{H}} q_{ij}(h_i, h_j) dh_i - q_j(h_j) = 0$, the Lagrangian is formed as

$$\begin{aligned} L(\{q_{ij}\}, \{q_i\}, \{\lambda_{ij}\}, \{\lambda_{ji}\}) \quad &= -\sum_i (|\mathcal{N}(i)| - 1) \int_{\mathcal{H}} q_i(h_i) \log \frac{q_i(h_i)}{\Phi(h_i, x_i)} dh_i \\ &+ \sum_{i,j} \int_{\mathcal{H}^2} q_{ij}(h_i, h_j) \log \frac{q_{ij}(h_i, h_j)}{\Psi(h_i, h_j)\Phi(h_i, x_i)\Phi(h_j, x_j)} dh_i dh_j \\ &- \sum_{i,j} \int_{\mathcal{H}} \lambda_{ij}(h_j) \Big(\int_{\mathcal{H}} q_{ij}(h_i, h_j) dh_i - q_j(h_j) \Big) dh_j \\ &- \sum_{i,j} \int_{\mathcal{H}} \lambda_{ji}(h_i) \Big(\int_{\mathcal{H}} q_{ij}(h_i, h_j) dh_j - q_i(h_i) \Big) dh_i \end{aligned}$$

with normalization constraints $\int_{\mathcal{H}} q_i(h_i) dh_i = 1$. Take functional gradients of $L(\{q_{ij}\}, \{q_i\}, \{\lambda_{ij}\}, \{\lambda_{ji}\})$ with respect to $q_{ij}(h_i, h_j)$ and $q_i(h_i)$, and set them to zero, we have

$$\begin{aligned} q_{ij}(h_i, h_j) \quad &\propto \quad \Psi(h_i, h_j)\Phi(h_i, x_i)\Phi(h_j, x_j) \exp(\lambda_{ij}(h_j) + \lambda_{ji}(h_i)), \\ q_i(h_i) \quad &\propto \quad \Phi(h_i, x_i) \exp\left(\frac{\sum_{k \in \mathcal{N}(i)} \lambda_{ki}(h_i)}{|\mathcal{N}(i)| - 1} \right). \end{aligned}$$

We set $m_{ij}(h_j) = \frac{q_j(h_j)}{\Phi(h_i, x_i) \exp(\lambda_{ij}(h_j))}$, therefore,

$$\prod_{k \in \mathcal{N}(i)} m_{ki}(h_i) \propto \exp\left(\frac{\sum_{k \in \mathcal{N}(i)} \lambda_{ki}(h_i)}{|\mathcal{N}(i)| - 1} \right).$$

Plug it into $q_{ij}(h_i, h_j)$ and $q_i(h_i)$, we recover the loopy BP update for marginal belief and

$$\exp(\lambda_{ji}(h_i)) = \frac{q_i(h_i)}{\Phi(h_i, x_i) m_{ji}(h_i)} \propto \prod_{k \in N(i) \backslash j} m_{ki}(h_i).$$

The update rule for message $m_{ij}(h_j)$ can be recovered using the marginal consistency constraints,

$$\begin{aligned} m_{ij}(h_j) \quad &= \quad \frac{q_j(h_j)}{\Phi(h_i, x_i) \exp(\lambda_{ij}(h_j))} = \frac{\int_{\mathcal{H}} q_{ij}(h_i, h_j) dh_i}{\Phi(h_i, x_i) \exp(\lambda_{ij}(h_j))} \\ &= \quad \Phi(h_j, x_j) \exp(\lambda_{ij}(h_j)) \frac{\int_{\mathcal{H}} \Psi(h_i, h_j)\Phi(h_i, x_i) \exp(\lambda_{ji}(h_i)) dh_i}{\Phi(h_i, x_i) \exp(\lambda_{ij}(h_j))} \\ &\propto \quad \int_{\mathcal{H}} \Psi(h_i, h_j)\Phi(h_i, x_i) \prod_{k \in N(i) \backslash j} m_{ki}(h_i) dh_i. \end{aligned}$$

Moreover, we also obtain the other important relationship between $m_{ij}(h_j)$ and $\lambda_{ji}(h_i)$ by marginal consistency constraint and the definition of $m_{ij}(h_j)$,

$$m_{ij}(h_j) \propto \int \Psi(h_i, h_j)\Phi(h_j, x_j) \exp(\lambda_{ji}(h_i)) dh_i.$$

C. Embedding Other Variational Inference

The proposed embedding is a general algorithm and can be tailored to other variational inference methods with message passing paradigm. In this section, we discuss the embedding for several alternatives, which optimize the primal and dual Bethe free energy, its convexified version and Kikuchi free energy, respectively. We will parametrize the messages with the same function class, *i.e.*, neural networks with ReLU. More generally, we can also treat the weights as parameters and learn them together.

C.1. Double-Loop BP

Noticed the Bethe free energy can be decomposed into the summation of a convex function and a concave function, Yuille (2002) utilizes CCCP to minimize the Bethe free energy, resulting the double-loop algorithm. Take the gradient of

Lagrangian of the objective function, and set to zero, the primal variable can be represented in dual form,

$$q_{ij}(h_i,h_j) \propto \Psi(h_i,h_j)\Phi(h_i,x_i)\Phi(h_j,x_j)\exp(\lambda_{ij}(h_j)+\lambda_{ji}(h_i)),$$
$$q_i(h_i) \propto \Phi(h_i,x_i)\exp\Big(|\mathcal{N}(i)|\gamma_s(h_i) - \sum_{k\in\mathcal{N}(i)}\lambda_{ki}(h_i)\Big),$$

The algorithm updates γ and λ alternatively,

$$\begin{aligned} 2\lambda_{ij}^{new}(h_j) &= |\mathcal{N}(j)|\gamma_i(h_i) - \sum_{k\in\mathcal{N}(j)\backslash i}\lambda_{kj}(h_j) - \log\int_{\mathcal{H}}\Psi(h_i,h_j)\Phi(h_i,x_i)\lambda_{ji}(h_i)dh_i, \\ \gamma_i^{new}(h_i) &= |\mathcal{N}(i)|\gamma_i(h_i) - \sum_{k\in\mathcal{N}(i)}\lambda_{ki}(h_i). \end{aligned}$$

Consider the λ_{ij} as messages, with the injective embedding assumptions for corresponding messages, follow the same notation, we can express the messages in embedding view

$$\begin{aligned} \widetilde{\nu}_{ij} &= \widetilde{\mathcal{T}}_1\circ\Big(x_i,\{\widetilde{\nu}_{ki}\}_{k\in\mathcal{N}(i)\backslash j},\widetilde{\nu}_{ji},\widetilde{\mu}_i\Big), \quad \text{or} \quad \widetilde{\nu}_{ij} = \widetilde{\mathcal{T}}_1\circ\Big(x_i,\{\widetilde{\nu}_{ki}\}_{k\in\mathcal{N}(i)},\widetilde{\mu}_i\Big), \\ \widetilde{\mu}_i &= \widetilde{\mathcal{T}}_2\circ\Big(x_i,\{\widetilde{\nu}_{ki}\}_{k\in\mathcal{N}(i)},\widetilde{\mu}_i\Big). \end{aligned}$$

Therefore, we have the parametrization form as

$$\begin{aligned} \widetilde{\nu}_{ij} &= \sigma\left(W_1x_i + W_2\sum_{k\in\mathcal{N}(i)\backslash j}\widetilde{\nu}_{ki} + W_3\widetilde{\nu}_{ji} + W_4\widetilde{\mu}_i\right), \quad \text{or} \quad \widetilde{\nu}_{ij} = \sigma\left(W_1x_i + W_2\sum_{k\in\mathcal{N}(i)}\widetilde{\nu}_{ki} + W_3\widetilde{\mu}_i\right), \\ \widetilde{\mu}_i &= \sigma\left(W_5x_i + W_6\sum_{k\in\mathcal{N}(i)}\widetilde{\nu}_{ki} + W_7\widetilde{\mu}_i\right). \end{aligned}$$

where $\widetilde{\mu}_i\in\mathbb{R}^d$, $\widetilde{\nu}_{ij}\in\mathbb{R}^d$, and $\boldsymbol{W}=\{W_i\}$ are matrices with appropriate size.

C.2. Damped BP

Instead of the primal form of Bethe free energy, Minka (2001) investigates the duality of the optimization,

$$\begin{aligned} \min_{\gamma}\max_{\lambda} \quad & \sum_i\Big(|\mathcal{N}(i)|-1\Big)\log\int_{\mathcal{H}}\Phi(h_i,x_i)\exp(\gamma_i(h_i))dh_i \\ - & \sum_{(i,j)\in\mathcal{E}}\log\int_{\mathcal{H}^2}\Psi(h_i,h_j)\Phi(h_i,x_i)\Phi(h_j,x_j)\exp(\lambda_{ij}(h_j)+\lambda_{ji}(h_i))dh_jdh_i, \end{aligned}$$

subject to $\Big(|\mathcal{N}(i)|-1\Big)\gamma_i(h_i)=\sum_{k\in\mathcal{N}(i)}\lambda_{ki}(h_i)$. Define message as

$$m_{ij}(h_j) \propto \int_{\mathcal{H}}\Psi(h_i,h_j)\Phi(h_j,x_j)\exp(\lambda_{ji}(h_i))dh_i,$$

the messages updates are

$$\begin{aligned} m_{ij}(h_i) &\propto \int_{\mathcal{H}}\Phi_i(h_i,x_i)\Psi_{ij}(h_i,h_j)\exp\Big(\frac{|\mathcal{N}(i)|-1}{|\mathcal{N}(i)|}\gamma_i(h_i)\Big)\frac{\prod_{k\in\mathcal{N}(i)}m_{ki}^{\frac{1}{|\mathcal{N}(i)|}}(h_i)}{m_{ji}(h_i)}dh_i, \\ \gamma_i^{new}(h_i) &\propto \frac{|\mathcal{N}(i)|-1}{|\mathcal{N}(i)|}\gamma_i(h_i) + \sum_{k\in\mathcal{N}(i)}\frac{1}{|\mathcal{N}(i)|}\log m_{ki}(h_i). \end{aligned}$$

and the

$$q(x_i) \propto \Phi(h_i,x_i)\exp\left(\frac{|\mathcal{N}(i)|-1}{|\mathcal{N}(i)|}\gamma_i(h_i)\right)\prod_{k\in\mathcal{N}(i)}m_{ki}^{\frac{1}{|\mathcal{N}(i)|}}$$

which results the embedding follows the same injective assumption and notations,

$$\begin{aligned} \widetilde{\nu}_{ij} &= \widetilde{\mathcal{T}}_1\circ\Big(x_i,\{\widetilde{\nu}_{ki}\}_{k\in\mathcal{N}(i)},\widetilde{\nu}_{ji},\widetilde{\mu}_i\Big), \\ \widetilde{\mu}_i &= \widetilde{\mathcal{T}}_2\circ\Big(x_i,\widetilde{\mu}_i,\{\widetilde{\nu}_{ki}\}_{k\in\mathcal{N}(i)}\Big). \end{aligned}$$

and the parametrization,

$$\begin{aligned}
\widetilde{\nu}_{ij} &= \sigma\left(W_1 x_i + W_2 \sum_{k\in\mathcal{N}(i)} \widetilde{\nu}_{ki} + W_3 \widetilde{\nu}_{ji} + W_4 \widetilde{\mu}_i\right),\\
\widetilde{\mu}_i &= \sigma\left(W_5 x_i + W_6 \widetilde{\mu}_i + W_7 \sum_{k\in\mathcal{N}(i)} \widetilde{\nu}_{ki}\right).
\end{aligned}$$

It is interesting that after parametrization, the embeddings of double-loop BP and damped BP are essentially the same, which reveal the connection between double-loop BP and damped BP.

C.3. Tree-reweighted BP

Different from loopy BP and its variants which optimizing the Bethe free energy, the tree-reweighted BP (Wainwright et al., 2003) is optimizing a convexified Bethe energy,

$$\begin{aligned}
\min_{\{q_{ij}\}_{(i,j\in\mathcal{E})}} L = &\sum_i \int_{\mathcal{H}} q(h_i)\log q(h_i) dh_i + \sum_{i,j} v_{ij} \int_{\mathcal{H}^2} q_{ij}(h_i,h_j)\log\frac{q_{ij}(h_i,h_j)}{q_i(h_i)q_j(h_j)} dh_i dh_j\\
&-\sum_i \int_{\mathcal{H}} q(h_i)\log\Phi(h_i,x_i)dh_i - \sum_{i,j}\int_{\mathcal{H}^2} q_{ij}(h_i,h_j)\log\Psi(h_i,h_j)dh_i dh_j
\end{aligned}$$

subject to pairwise marginal consistency constraints: $\int_{\mathcal{H}} q_{ij}(h_i,h_j)dh_j = q_i(h_i)$, $\int_{\mathcal{H}} q_{ij}(h_i,h_j)dh_j = q_i(h_i)$, and $\int_{\mathcal{H}} q_i(h_i)dh_i = 1$. The $\{v_{ij}\}_{(i,j)\in\mathcal{E}}$ represents the probabilities that each edge appears in a spanning tree randomly chose from all spanning tree from $G = \{\mathcal{V},\mathcal{E}\}$ under some measure. Follow the same strategy as loopy BP update derivations, *i.e.*, take derivatives of the corresponding Lagrangian with respect to q_i and q_{ij} and set to zero, meanwhile, incorporate with the marginal consistency, we can arrive the messages updates,

$$\begin{aligned}
m_{ij}(h_j) &\propto \int_{\mathcal{H}} \Psi_{ij}^{\frac{1}{v_{ji}}}(h_i,h_j)\Phi_i(h_i,x_i)\frac{\prod_{k\in\mathcal{H}(i)\backslash j} m_{ki}^{v_{ki}}(h_i)}{m_{ji}^{1-v_{ij}}(h_i)} dh_i,\\
q_i(h_i) &\propto \Phi_i(h_i,x_i)\prod_{k\in\mathcal{N}(i)} m_{ki}^{v_{ki}}(h_i),\\
q_{ij}(h_i,h_j) &\propto \Psi_{ij}^{\frac{1}{v_{ji}}}(h_i,h_j)\Phi_i(h_i,x_i)\Phi_j(h_j,x_j)\frac{\prod_{k\in\mathcal{N}(i)\backslash j} m_{ki}^{v_{ki}}(h_i)}{m_{ji}^{1-v_{ij}}(h_i)}\frac{\prod_{k\in\mathcal{N}(j)\backslash i} m_{kj}^{v_{kj}}(h_j)}{m_{ij}^{1-v_{ji}}(h_j)}.
\end{aligned}$$

Similarly, the embedded messages and the marginals on nodes can be obtained as

$$\begin{aligned}
\widetilde{\nu}_{ij} &= \widetilde{\mathcal{T}}_1 \circ \left(x_i, \{\widetilde{\nu}_{ki}\}_{k\in\mathcal{N}(i)\backslash j}, \widetilde{\nu}_{ji}, \{v_{ki}\}_{k\in\mathcal{N}i\backslash j}, v_{ij}\right),\\
\widetilde{\mu}_i &= \widetilde{\mathcal{T}}_2 \circ \left(x_i, \{\widetilde{\nu}_{ki}, v_{ki}\}_{k\in\mathcal{N}(i)}\right).
\end{aligned}$$

Parametrize these message in the same way, we obtain,

$$\begin{aligned}
\widetilde{\nu}_{ij} &= \sigma\left(W_1 x_i + W_2 \sum_{k\in\mathcal{N}(i)\backslash j} \widetilde{v}_{ki}\nu_{ki} + W_3 \widetilde{v}_{ij}\nu_{ji}\right),\\
\widetilde{\mu}_i &= \sigma\left(W_4 x_i + W_5 \sum_{k\in\mathcal{N}(i)} \widetilde{v}_{ki}\nu_{ki}\right).
\end{aligned}$$

Notice the tree-weighted BP contains extra parameters $\{v_{ij}\}_{(i,j)\in\mathcal{E}}$ which is in the spanning tree polytope as (Wainwright et al., 2003).

C.4. Generalized Belief Propagation

The Kikuchi free energy is the generalization of the Bethe free energy by involving *high-order* interactions. More specifically, given the MRFs, we denote R to be a set of regions, *i.e.*, some basic clusters of nodes, their intersections, the intersections of the intersections, and so on. We denote the $sub(r)$ or $sup(r)$, *i.e.*, subregions or superregions of r, as the set of regions completely contained in r or containing r, respectively. Let h_r be the state of the nodes in region r, then, the Kikuchi free energy is

$$\sum_{r\in R} c_r \left(\int q(h_r) \log \frac{q(h_r)}{\prod_{i,j\in r} \Psi(h_i,h_j) \prod_{i\in r} \Phi(h_i,x_i)} \right),$$

where c_r is over-counting number of region r, defined by $c_r := 1 - \sum_{s\in sup(r)} c_s$ with $c_r = 1$ if r is the largest region in R. It is straightforward to verify that the Bethe free energy is a special case of the Kikuchi free energy by setting the basic cluster as pair of nodes. The generalized loopy BP (Yedidia et al., 2001b) is trying to seek the stationary points of the Kikuchi free energy under regional marginal consistency constraints and density validation constraints by following messages updates,

$$m_{r,s}(h_s) \propto \frac{\int \overline{\Psi}(h_r, x_{r\setminus s}) \bar{m}_{r\setminus s}(h_{r\setminus s}) dh_{r\setminus s}}{\bar{m}_{r,s}(h_s)},$$
$$q_r(h_r) \propto \prod_{i\in r} \Phi(h_i,x_i) \prod_{m_{r',s'}\in M(r)} m_{r',s'}(h_{s'}), \tag{19}$$

where

$$\bar{m}_{r\setminus s}(h_{r\setminus s}) = \prod_{\{r',s'\}\in M(r)\setminus M(s)} m_{r',s'}(h_{s'}),$$
$$\bar{m}_{r,s}(h_s) = \prod_{\{r',s'\}\in M(r,s)} m_{r',s'}(h_{s'}),$$
$$\overline{\Psi}(h_r, x_{r\setminus s}) = \prod_{i,j\in r} \Psi(h_i,h_j) \prod_{i\in r\setminus s} \Phi(h_i,x_i).$$

The $M(r)$ denotes the indices of messages $m_{r',s'}$ that $s' \in sub(r) \cup \{r\}$, and $r' \setminus s'$ is outside r. $M(r,s)$ is the set of indices of messages $m_{r',s'}$ where $r' \in sub(r) \setminus s$ and $s' \in sub(s) \cup \{s\}$.

With the injective embedding assumption for each message $\widetilde{\nu}_{r,s} = \int \phi(h_s) m_{r,s}(h_s) dh_s$ and $\widetilde{\mu}_r = \int \phi(h_r) q_r(h_r) dh_r$, following the reasoning (3) and (4), we can express the embeddings as

$$\widetilde{\nu}_{r,s} = \widetilde{\mathcal{T}}_1 \circ \left(x_{r\setminus s}, \{\widetilde{\nu}_{r',s'}\}_{M(r)\setminus M(s), M(r,s)} \right), \tag{20}$$

$$\widetilde{\mu}_r = \widetilde{\mathcal{T}}_2 \circ \left(x_r, \{\widetilde{\nu}_{r',s'}\}_{M(r)} \right). \tag{21}$$

Following the same parameterization in loopy BP, we represent the embeddings by neural network with rectified linear units,

$$\widetilde{\nu}_{r,s} = \sigma\Big(\sum_{i\in r} W_1^i x_r^i + W_2 \sum_{M(r)\setminus M(s)} \widetilde{\nu}_{r',s'} + W_3 \sum_{M(r,s)} \widetilde{\nu}_{r',s'} \Big) \tag{22}$$

$$\widetilde{\mu}_i = \sigma\Big(\sum_{i\in r} W_4^i x_i + W_5 \sum_{M(r)} \widetilde{\nu}_{r',s'} \Big) \tag{23}$$

where $\boldsymbol{W} = \{\{W_1^i\}, W_2, W_3, \{W_4^i\}, W_5\}$ are matrices with appropriate sizes. The generalized BP embedding updates will be almost the same as Algorithm 2 except the order of the iterations. We start from the messages into the smallest region first (Yedidia et al., 2001b).

Remark: The choice of basis clusters and the form of messages determine the dependency in the embedding. Please refer to Yedidia et al. (2005) for details about the principles to partition the graph structure, and several other generalized BP variants with different messages forms. The algorithms proposed for minimizing the Bethe free energy (Minka, 2001; Heskes, 2002; Yuille, 2002) can also be extended for Kikuchi free energy, resulting in different embedding forms.

D. Derivatives Computation in Algorithm 3

We can use the chain rule to obtain the derivatives with respect to $\boldsymbol{U}^T = \{\boldsymbol{W}^T, \boldsymbol{u}^T\}$. According to Equation 17 and Equation 18, the message passed to supervised label y_n for $n-$th sample can be represented as $m_y^n = \sum_{i\in\mathcal{V}} \tilde{\mu}_i{}^n$, and the corresponding derivative can be denoted as

$$\frac{\partial l}{\partial m_y^n} = \frac{\partial l}{\partial f} \frac{\partial f}{\partial \sigma(m_y^n)} \frac{\partial \sigma(m_y^n)}{\partial m_y^n}$$

The term $\frac{\partial l}{\partial f}$ depends on the supervised information and the loss function we used, and $\frac{\partial l}{\partial \sigma(m_y^n)} = \boldsymbol{u}^T \frac{\partial l}{\partial f}$. The last term

$\frac{\partial \sigma(m_y^n)}{\partial m_y^n}$ depends on the nonlinear function σ we used here.

The derivatives with respect to $\boldsymbol{u}$ for the current encountered sample $\{\chi_n, y_n\}$ SGD iteration are

$$\nabla_{\boldsymbol{u}} l(f(\widetilde{\mu}^n; \boldsymbol{U}), y_n) = \frac{\partial l}{\partial f} \sigma(m_y^n)^T \tag{24}$$

In order to update the embedding parameters $\boldsymbol{W}$, we need to obtain the derivatives with respect to the embedding of each hidden node, i.e., $\frac{\partial l}{\partial \tilde{\mu_i}^n} = \frac{\partial l}{\partial m_y^n}, \forall i \in \mathcal{V}$.

Embedded Mean Field

In mean field embedding, we unfold the fixed point equation by the iteration index $t = 1, 2, \ldots, T$. At $t-$th iteration, the partial derivative is denoted as $\frac{\partial l}{\partial \tilde{\mu_i}^{n(t)}}$. The partial derivative with respect to the embedding obtained by last round fixed point iteration is already defined above: $\frac{\partial l}{\partial \tilde{\mu_i}^{n(T)}} = \frac{\partial l}{\partial m_y^n}$

Then the derivatives can be obtained recursively: $\frac{\partial l}{\partial \tilde{\mu_i}^{n(t)}} = \sum_{j, i \in \mathcal{N}(j)} W_2^T \frac{\partial l}{\partial \tilde{\mu_j}^{n(t+1)}} \frac{\partial \sigma}{\partial (W_1 x_j + W_2 l_j^{(t)} + W_3 u_j)}$, $t = 1, 2, \ldots, T-1$. Similarly, the parameters $\boldsymbol{W}$ are also updated cumulatively as below.

$$\frac{\partial l}{\partial (W_1 x_i + W_2 l_i^{(t)} + W_3 u_i)} = \sum_{j, i \in \mathcal{N}(j)} \frac{\partial l}{\partial \tilde{\mu_j}^{n(t+1)}} \frac{\partial \sigma}{\partial (W_1 x_j + W_2 l_j^{(t)} + W_3 u_j)} \tag{25}$$

$$\nabla_{W_1} l(f(\widetilde{\mu}^n; \boldsymbol{U}), y_n) = \sum_{i \in \mathcal{V}_n} \sum_{t=1}^{T-1} \frac{\partial l}{\partial (W_1 x_i + W_2 l_i^{(t)} + W_3 u_i)} x_i^T \tag{26}$$

$$\nabla_{W_2} l(f(\widetilde{\mu}^n; \boldsymbol{U}), y_n) = \sum_{i \in \mathcal{V}_n} \sum_{t=1}^{T-1} \frac{\partial l}{\partial (W_1 x_i + W_2 l_i^{(t)} + W_3 u_i)} l_i^{(t)T} \tag{27}$$

$$\nabla_{W_3} l(f(\widetilde{\mu}^n; \boldsymbol{U}), y_n) = \sum_{i \in \mathcal{V}_n} \sum_{t=1}^{T-1} \frac{\partial l}{\partial (W_1 x_i + W_2 l_i^{(t)} + W_3 u_i)} u_i^T \tag{28}$$

Embedding Loopy BP

Similar as above case, we can first obtain the derivatives with respect to embeddings of hidden variables $\frac{\partial l}{\partial \tilde{\mu_i}^n} = \frac{\partial l}{\partial m_y^n}$. Since the last round of message passing only involves the edge-to-node operations, we can easily get the following derivatives.

$$\nabla_{W_3} l(f(\widetilde{\mu}^n; \boldsymbol{U}), y_n) = \sum_{i \in \mathcal{V}} \frac{\partial l}{\partial \tilde{\mu_i}^n} \frac{\partial \sigma}{\partial (W_3 x_i + W_4 \sum_{k \in \mathcal{N}(i)} \widetilde{\nu}_{ki}^{n(T)})} x_i^T \tag{29}$$

$$\nabla_{W_4} l(f(\widetilde{\mu}^n; \boldsymbol{U}), y_n) = \sum_{i \in \mathcal{V}} \frac{\partial l}{\partial \tilde{\mu_i}^n} \frac{\partial \sigma}{\partial (W_3 x_i + W_4 \sum_{k \in \mathcal{N}(i)} \widetilde{\nu}_{ki}^{n(T)})} (\sum_{k \in \mathcal{N}(i)} \widetilde{\nu}_{ki}^{n(T)})^T \tag{30}$$

$$\tag{31}$$

Now we consider the partial derivatives for the pairwise message embeddings for different t. Again, the top level one is trivial, which is given by $\frac{\partial l}{\partial \widetilde{\nu_{ij}}^{n(T)}} = W_4^T \frac{\partial l}{\partial \tilde{\mu_j}} \frac{\partial \sigma}{\partial (W_3 x_j + W_4 \sum_{k \in \mathcal{N}(j)} \widetilde{\nu}_{kj}^{(T)})}$. Using similar recursion trick, we can get the following chain rule for getting partial derivatives with respect to each pairwise message in each stage of fixed point iteration.

$$\frac{\partial l}{\partial \widetilde{\nu}_{ij}^{n(t)}} = \sum_{p \in \mathcal{N}(j) \backslash i} W_2^T \frac{\partial l}{\partial \widetilde{\nu}_{jp}^{n(t+1)}} \frac{\partial \sigma}{\partial (W_1 x_j + W_2 \sum_{k \in \mathcal{N}(j) \backslash p} [\widetilde{\nu}_{kj}^{n(t)}])} \tag{32}$$

Then, we can update the parameters W_1, W_2 using following gradients.

$$\nabla_{W_1} l(f(\widetilde{\mu}^n; \boldsymbol{U}), y_n) = \sum_{t=1}^{T-1} \sum_{(i,j)\in\mathcal{E}_n} \frac{\partial l}{\partial \widetilde{\nu}_{ij}^{n(t+1)}} \frac{\partial \sigma}{\partial (W_1 x_i + W_2 \sum_{k\in\mathcal{N}(i)\setminus j} [\widetilde{\nu}_{ki}^{n(t)}])} x_i^T \tag{33}$$

$$\nabla_{W_2} l(f(\widetilde{\mu}^n; \boldsymbol{U}), y_n) = \sum_{t=1}^{T-1} \sum_{(i,j)\in\mathcal{E}_n} \frac{\partial l}{\partial \widetilde{\nu}_{ij}^{n(t+1)}} \frac{\partial \sigma}{\partial (W_1 x_i + W_2 \sum_{k\in\mathcal{N}(i)\setminus j} [\widetilde{\nu}_{ki}^{n(t)}])} (\sum_{k\in\mathcal{N}(i)\setminus j} [\widetilde{\nu}_{ki}^{n(t)}])^T \tag{34}$$

$$\tag{35}$$

E. More Experiment Details

E.1. Graph Datasets

We show the detailed statistics of these graph datasets in Table 3.

	size	avg $\lvert V\rvert$	avg $\lvert E\rvert$	#labels
MUTAG	188	17.93	19.79	7
NCI1	4110	29.87	32.3	37
NCI109	4127	29.68	32.13	38
ENZYMES	600	32.63	62.14	3
D&D	1178	284.32	715.66	82

Table 3. Statistics (Sugiyama & Borgwardt, 2015) of graph benchmark datasets. $|V|$ is the # nodes while $|E|$ is the # edges in a graph. #labels equals to the number of different types of nodes.

E.2. Experiment Settings

For the small scaled string and graph datasets, we tune all the methods via cross validation. Specifically, for structured kernel methods, we tune the degree in $\{1, 2, \ldots, 10\}$ (for mismatch kernel, we also tune the maximum mismatch length in $\{1, 2, 3\}$) and train SVM classifier (Chang & Lin, 2001) on top, where the trade-off parameter C is also chosen in $\{0.01, 0.1, 1, 10\}$ by cross validation. For fisher kernel that using HMM as generative model, we also tune the number of hidden states assigned to HMM in $\{2, \ldots, 20\}$.

For our methods, we simply use one-hot vector (the vector representation of discrete node attribute) as the embedding for observed nodes, and use a two-layer neural network for the embedding (prediction) of target value. The hidden layer size $b \in \{16, 32, 64\}$ of neural network, the embedding dimension $d \in \{16, 32, 64\}$ of hidden variables and the number of iterations $t \in \{1, 2, 3, 4\}$ are tuned via cross validation. We keep the number of parameters small, and use early stopping (Giles, 2001) to avoid overfitting in these small datasets.