Appendix to Kernel Belief Propagation

May 12, 2011

Section 1 contains a review of Gaussian mixture BP and particle BP, as well as a detailed explanation of our strategy for learning edge potentials for these approaches from training data. Section 2 provides parameter settings and experiment details for particle BP and discrete BP, in the synthetic image denoising and depth reconstruction experiments. Section 3 contains a comparison of two different approximate feature sets: low rank approximation of the tensor features and low rank approximation of the individual features alone. Section 4 is an experiment on learning paper categories using citation networks. Sections 5 and 6 demonstrate the optimization objective of locally consistent BP updates, and provide a derivation of these updates in terms of the conditional expectation. Section 7 discusses the kernelization of Gaussian BP. Section 8 gives the error introduced by low rank approximation of the messages.

1 Gaussian Mixture and Particle BP

We describe two competing approaches for nonparametric belief propagation: Gaussian mixture BP, originally known as non-parametric BP (Sudderth et al., 2003), and particle BP (Ihler & McAllester, 2009). For these algorithms, the edge potentials $\Psi(x_s, x_t)$, self-potentials $\psi(x_t)$, and evidence potentials $\Psi(x_t, y_t)$ must be provided in advance by the user. Thus, we begin by describing how the edge potentials in Section 2 of the main document may be learned from training data, but in a form applicable to these inference algorithms: we express $\mathbb{P}(x_t|x_s)$ as a mixture of Gaussians. We then describe the inference algorithms themselves.

In learning the edge potentials, we turn to Sugiyama et al. (2010), who provide a least-squares estimate of a conditional density in the form of a mixture of Gaussians,

$$\mathbb{P}(v|u) = \sum_{i=1}^{b} \alpha_i \kappa_i(u, v) = \alpha^\top \boldsymbol{\kappa}_{u,v},$$

where $\kappa_i(u, v)$ is a Gaussian with diagonal covariance centred at[1] (q_i, r_i). Given a training set $\{(u^j, v^j)\}_{j=1}^m$, we obtain the coefficients

$$\alpha := \left[\left(\widehat{H} + \lambda I\right)^{-1} \hat{h}\right]_+,$$

where $\widehat{H} := \sum_{j=1}^m \int_{\mathcal{V}} \boldsymbol{\kappa}_{u^j,v} \boldsymbol{\kappa}_{u^j,v}^\top dv$, $\hat{h} := \sum_{j=1}^m \boldsymbol{\kappa}_{u^j,v^j}$, λ is a regularization coefficient, and $[\alpha]_+$ sets all negative entries of its argument to zero (the integral in $\widehat{H}$ can easily be computed in closed form). We emphasize that the Gaussian mixture representation takes a quite different form to the RKHS representation of the edge potentials. Finally, to introduce evidence, we propose to use kernel ridge regression to provide a mean value of the hidden variable x_t given the observation y_t, and to center a Gaussian at this value: again, the regression function is learned nonparametrically from training data.

We now describe how these edge potentials are incorporated into Gaussian mixture BP. Assuming the incoming messages are each a mixture of b Gaussians, the product of d_t such messages will contain b^{d_t} Gaussians, which causes an exponential blow-up in representation size and computational cost. In their original work, Sudderth et al. address this issue using an approximation scheme. First, they subsample from the incoming mixture of b^{d_t} Gaussians to draw b Gaussians, at a computational cost of $O(d_t \tau b^2)$ *for each node*, where τ is the number of iterations of the associated Gibbs sampler (see their Algorithm 1). The evidence introduced via kernel ridge regression is then incorporated, using a reweighting described by their Algorithm 2. Finally, in Algorithm 3, b samples $\left\{x_t^i\right\}_{i=1}^b$ are drawn from the reweighted mixture of b Gaussians, and for each of these, $\left\{x_s^i\right\}_{i=1}^b$ are drawn

[1] These centres may be selected at random from the training observations. We denote the mixture kernel by $\kappa(u, v)$ to distinguish it from the RKHS kernels used earlier.

from the conditional distribution $x_s|x_t^i$ arising from the edge potential $\psi(x_s, x_t)$ (which is itself a Gaussian mixture, learned via the approach of Sugiyama et al.). Gaussians are placed on each of the centres $\left\{x_s^i\right\}_{i=1}^b$, and the process is iterated.

In our implementation, we used the more efficient multiscale KD-tree sampling method of Ihler et al. (2003). We converted the Matlab Mex implementation of Ihler (2003) to C++, and used GraphLab to execute sampling in parallel with up to 16 cores. An input parameter to the sampling procedure is ϵ, the level of accuracy. We performed a line search to set ϵ for high accuracy, but limited the execution time to be at most 1000 times slower than KBP.

Finally, we describe the inference procedure performed by Particle BP. In this case, each node t is associated with a set of particles $\left\{x_t^i\right\}_{i=1}^b$, drawn i.i.d. from a distribution $W_t(x_t)$. Incoming messages m_{ut} are expressed as weights of the particles x_t^i. Unlike Gaussian mixture BP, the incoming messages all share the same set of particles, which removes the need for Parzen window smoothing. The outgoing message m_{ts} is computed by summing over the product of these weights and the edge and evidence potentials at the particles, yielding a set of weights over samples $\left\{x_s^i\right\}_{i=1}^b$ at node s; the procedure is then iterated (see Ihler & McAllester, 2009, eq. 8). We again implement this algorithm using edge potentials computed according to Sugiyama et al. Since an appropriate sample distribution W_t is hard to specify in advance, a resampling procedure must be carried out at each BP iteration, to refresh the set of samples at each node and ensure the samples cover an appropriate support (this is a common requirement in particle filtering). Thus, each iteration of Particle BP requires a Metropolis-Hastings chain to be run for every node, which incurs a substantial computational cost. That said, we found that in practice, the resampling could be conducted less often without an adverse impact on performance, but resulting in major improvements in runtime, as described in Section 2 below. See (Ihler & McAllester, 2009, Section 6) for more detail.

2 Settings for Discrete and Particle BP

2.1 Depth Reconstruction from 2-D Images

2.1.1 Discrete BP

The log-depth was discretized into 30 bins, and edge parameters were selected to achieve locally consistent Loopy BP marginals using the technique described in Wainwright et al. (2003). Empirically, finer discretizations did not improve resultant accuracy, but increased runtime significantly. We used the Splash scheduling of Gonzalez et al. (2009) since it provided the lowest runtime among all tested schedulings.

2.1.2 Particle BP

The particle belief propagation implementation was particularly difficult to tune due to its excessively high runtime. Theoretically, results comparable to the Kernel BP method were attainable. However in practice, the extremely high cost of the resampling phase on large models meant that only a small number of particles could be maintained if a reasonable runtime was to be achieved on our evaluation set of 274 images.

Ultimately, we decided to find a configuration which allowed us to complete the evaluation in about 2 machine-days on an 8-core Intel Nehalem machine; allowing inference on each evaluation image to take 10 minutes of parallel computation. For each image, we ran 100 iterations of a simple linear-sweep scheduling, using 20 particles per message, and resampling every 10 iterations. Each resampling phase ran MCMC for a maximum of 10 steps per particle. We also implemented acceleration tricks where low weight particles ($< 1E-7$ after normalization) were ignored during the message passing process. Empirically this decreased runtime without affecting the quality of results.

2.2 Synthetic Image Denoising

2.2.1 Discrete BP

To simplify evaluation, we permitted a certain degree of "oracle" information, by matching the discretization levels during inference with the color levels in the ground-truth image.

We evaluated combined gradient/IPF + BP methods here to learn the edge parameters. We found that gradient/IPF performed well when there were few colors in the image, but failed to converge when the number of colors increased into the hundreds. This is partly due to the instability of BP, as well as the large number of free parameters in the edge potential.

Therefore once again, edge potentials were selected using the technique described in Wainwright et al. (2003). This performed quite well empirically, as seen in Figure 1(c) (main document).

2.2.2 Particle BP

The high runtime of the Particle Belief Propagation again made accuracy evaluation difficult. As before, we tuned the particle BP parameters to complete inference on the evaluation set of 110 images in 2 machine days, allowing about 25 minutes per evaluation image. We ran 100 iterations of 30 particles per message, resampling every 15 iterations. Each resampling phase ran MCMC for a maximum of 10 steps per particle.

3 Effects of Approximate Message Updates

In this section, we study how different levels of feature approximation error ϵ affect the speed of kernel BP and the resulting performance. Our experimental setup was the image denoising experiment described in Section 5.1 of the main document. We note that the computational cost of our constant message update is $O(\ell^2 d_{\max})$ where ℓ is inversely related to the approximation error ϵ. This is a substantial runtime improvement over naively applying a low rank kernel matrix approximation, which only results in a linear time update with computational cost $O(\ell m d_{\max})$. In this experiment, we varied the feature approximation error ϵ over three levels, *i.e.* $10^{-1}, 10^{-2}, 10^{-3}$, and compared both speed and denoising performance of the constant time update to the linear time update.

From Figures 1 (a) and (c), we can see that for each approximation level, the constant time update achieves about the same denoising performance as the linear time update, while at the same time being orders of magnitude faster (by comparing Figures 1 (b) and (d)). Despite the fact that the constant time update algorithm makes an additional approximation to the tensor product features, its denoising performance is not affected. We hypothesize that the degradation in performance is largely caused by representing the messages in terms of a small number of basis functions, while the approximation to the tensor features introduces little additional degradation.

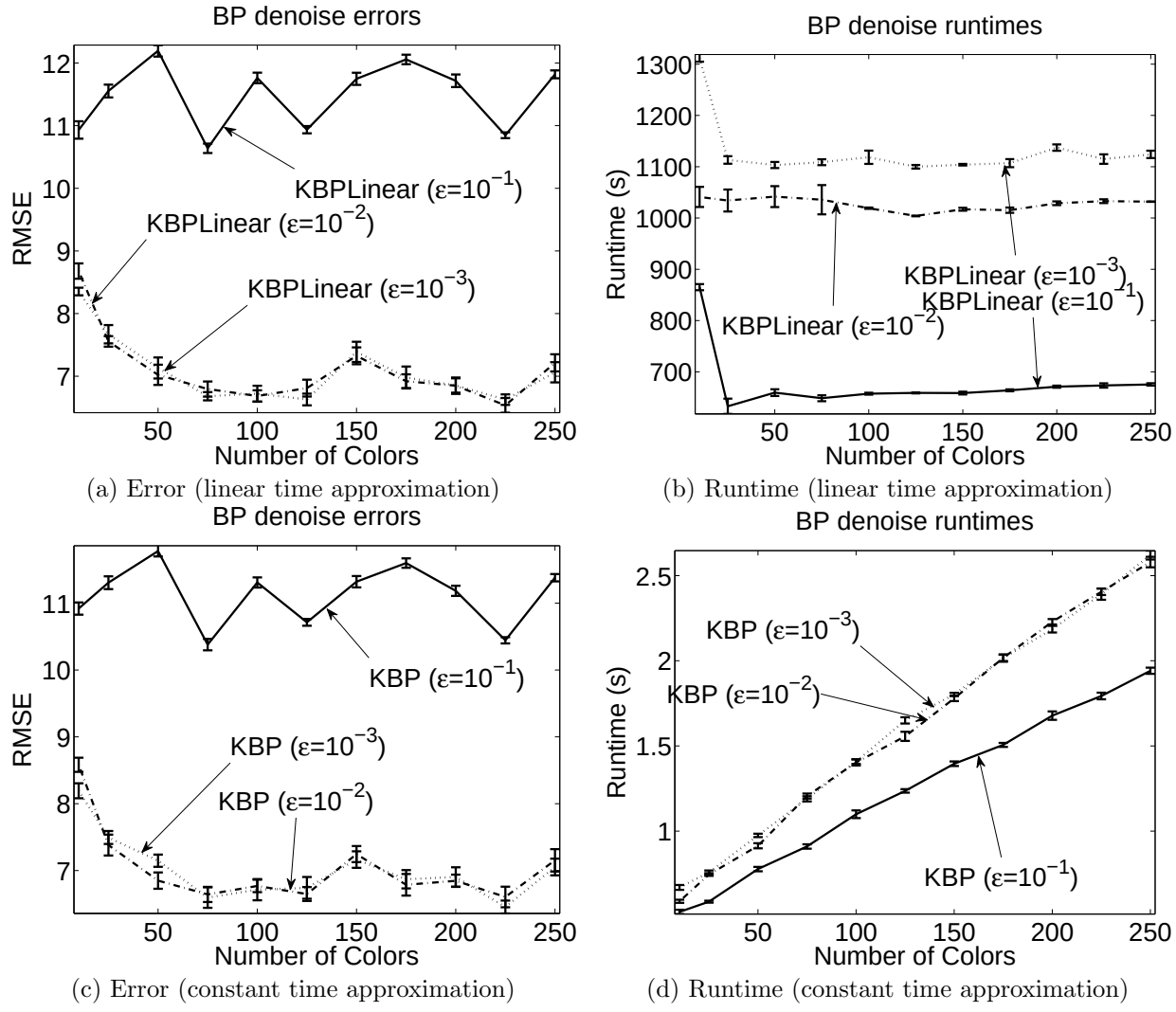
Another interesting observation from Figure 1 (d) is that the runtime of constant time kernel BP update increases as the number of colors in the image increases. This is mainly due to the increased number of test points as the color number increases; and also partially due to the increased rank needed for approximating the tensor features. In Figure 2, we plot the rank needed for kernel feature approximation and tensor feature approximation for different numbers of colors and different approximation errors ϵ. It can be seen that in general, as we use a smaller approximation error, the rank increases, leading to a slight increase in runtime.

Finally, we compare with kernel belief propagation in the absence of any low rank approximation (KBP Full). Since KBP Full is computationally expensive, we reduced the denoising problem to images of size 50×50 to allow KBP to finish in reasonable time. We only compared on 100 color images, again for reasons of cost. We varied the feature approximation error for the constant time and linear time approximation over three levels, 10^{-1}, 10^{-2}, 10^{-3}, and compared both speed and denoising performance of KBP Full versus the constant time and linear time updates.

The comparisons are shown in Figure 3. We can see from Figure 3(a) that the denoising errors for constant time and linear time approximations decrease as we decrease the approximation error ϵ. Although the denoising error of KBP Full is slightly lower than constant time approximations, it is a slight increase over the linear time approximation at $\epsilon = 10^{-3}$. One reason might be that the kernel approximation also serves as a means of regularization when learning the conditional embedding operator. This additional regularization may have slightly improved the generalization ability of the linear time approximation scheme. In terms of runtime (Figure 3(b)), constant time approximation is substantially faster than linear time approximation and KBP Full. In particular, it is nearly 100 times faster than the linear time algorithm, and 10000 times faster than KBP Full.

4 Predicting Paper Categories

In this experiment, we predict paper categories from a combination of the paper features and their citation network. Our data were obtained by crawling 143,086 paper abstracts from the ACM digital library, and extracting the citation networks linking these papers. Each paper was labeled with a variable number of categories, ranging from 1 to 10; there were a total of 367 distinct categories in our dataset. For simplicity, we ignored directions in the citation network, and treated it as an undirected graph (i.e, we did not distinguish "citing" and "being cited"). The citation network was sparse, with more than 85% of the papers having fewer than 10 links. The maximum number of links was 450.



(a) Error (linear time approximation)

(b) Runtime (linear time approximation)

(c) Error (constant time approximation)

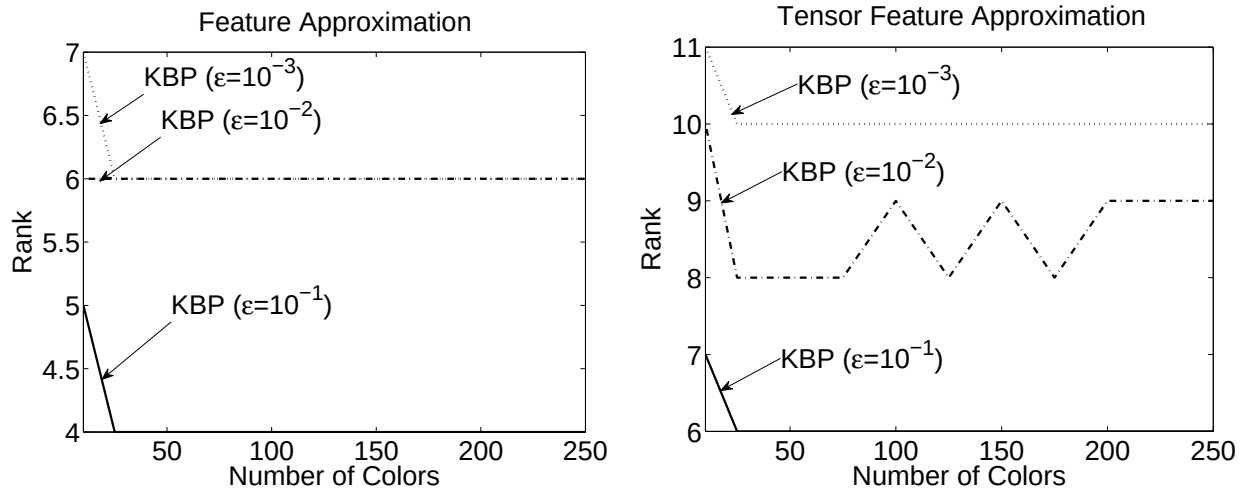
(d) Runtime (constant time approximation)

Figure 1: Average denoising error and runtime of linear time kernel BP versus constant time kernel BP, using different feature approximation errors, over 10 test images with a varying number of image colors.

Paper category prediction is a multi-label problem with a large output space. The output is very sparse: the label vectors have only a small number of nonzero entries. In this case, the simple one-against-all approach of learning a single predictor for each category can become prohibitively expensive, both in training and in testing. Recently, Hsu et al. (2009) proposed to solve this problem using compressed sensing techniques: high dimensional sparse category labels are first compressed to lower dimensional real vectors using a random projection, and regressors are learned for these real vectors. In the testing stage, high dimensional category labels are decoded from the predicted real vectors of the test data using orthogonal marching pursuit (OMP).

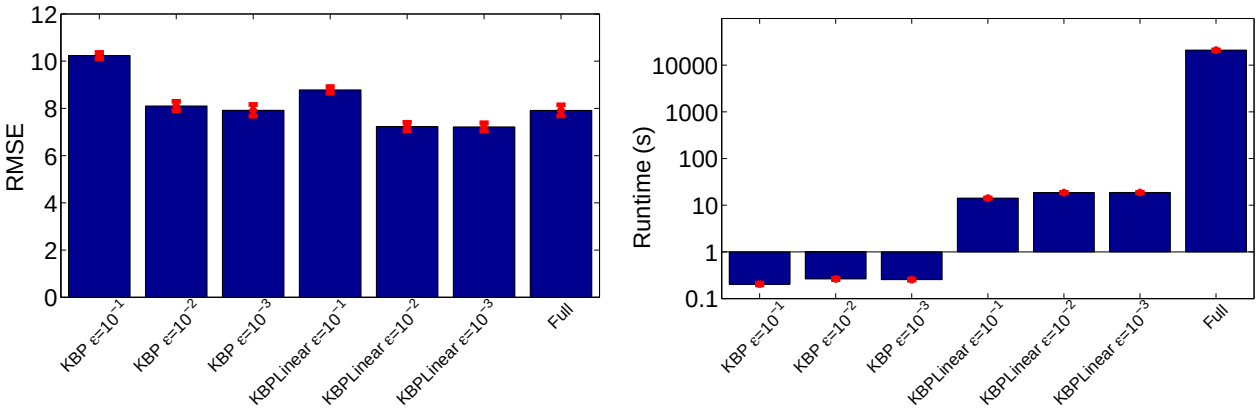
For the purposed of the present task, however, the compressed sensing approach ignores information from the citation network: papers that share categories tend to cite each other more often. Intuitively, taking into account category information from neighboring papers in the citation network should improve the performance of category prediction. This intuition can be formalized using undirected graphical models: each paper i contains a category variable $y_i \in \{0,1\}^{367}$, and these variables are connected according to the citation network; each category variable is also connected to a variable x_i corresponding to the abstract of the paper. In our experiment, we used 9700 stem words for the abstracts, and $x_i \in \mathbb{R}^{9700}$ was the tf-idf vector for paper i. The graphical model thus contains two types of edge potential, $\Psi(y_i, y_i)$ and $\Psi(y_i, y_k)$, where $k \in \mathcal{N}(j)$ is the neighbor of j according to the citation network.

It is difficult to learn this graphical model and perform inference on it, since the category variables y_i have high cardinality, making the marginalization step in BP prohibitively expensive. Inspired by the compressed sensing approach for multilabel prediction, we first employed random projection kernels, and then used our kernel BP



(a) Rank for feature approximation (b) Rank for tensor feature approximation

Figure 2: The rank obtained for kernel feature approximation and tensor feature approximation for different levels of approximation error ϵ, for the image denoising problem.



(a) Error (b) Runtime (vertical axis is in log scale)

Figure 3: (a) Denoising error for the constant time approximate update (KBP) and linear time approximate update (KBPLinear), over three levels of approximation error $\epsilon = \{10^{-1}, 10^{-2}, 10^{-3}\}$, versus kernel BP without low rank approximation (Full). (b) Runtime corresponding to different approximation schemes.

algorithm. Let $A \in \mathbb{R}^{d \times 367}$ be a random matrix containing *i.i.d.* Gaussian random variables of zero mean and variance $1/d$. We defined the random projection kernel for the category labels to be $k(y, y') = \langle Ay, Ay' \rangle = \langle \phi(y), \phi(y') \rangle$, and used a linear kernel for the abstract variables. We ran kernel BP for 5 iterations, since further iterations did not improve the performance. MAP assignment based on the belief was performed by finding a unit vector $\phi(\hat{y}) = A\hat{y}$ that maximized the belief. The sparse category labels $\hat{y}$ were decoded from $\phi(\hat{y})$ using OMP.

To measure experimental performance, we performed 10 random splits of the papers, where in each split we used 1/3 of the papers for training and the remaining 2/3 for testing. The random splits were controlled in such a way that high degree nodes (with degree > 10) in the citation networks always appeared in the training set. Such splitting reflects the data properties expected in real-world scenarios: important papers (high degree nodes which indicate either influential papers or survey papers) are usually labeled, whereas the majority of papers may not have a label; the automatic labeling is mainly needed for these less prominent papers. We used recall@k in evaluating the performance of our method on the test data. We compared against the regression technique of Hsu et al. for multilabel prediction, and a baseline prediction using the top k most frequent categories. For both our method and the method of Hsu et al., we used a random projection matrix with $d = 100$.

Results are shown in Figure 4. Kernel BP performs better than multilabel prediction via compressed sensing (i.e., the independent regression approach, which ignores graphical model structure) over a range of k values. In particular, for the top 10 and 20 predicted categories, kernel BP achieves recall scores of 0.419 and 0.476,

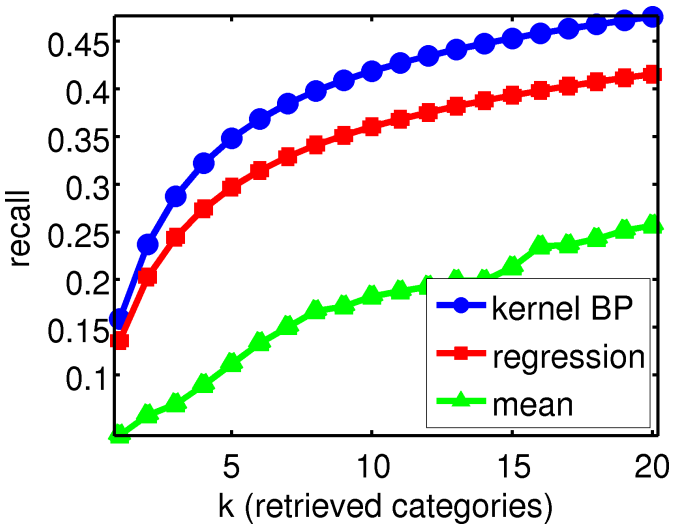


Figure 4: Comparison of kernel BP, multilabel prediction via compressed sensing (regression), and a baseline prediction using the top k most frequent categories (mean) for ACM paper category prediction.

respectively, as opposed to 0.362 and 0.417 for independent regression.

5 Local Marginal Consistency Condition When Learning With BP

In this section, we show that fixed points of BP satisfy particular marginal consistency conditions (14) and (15) below. As we will see, these arise from the fact that we are using a Bethe free energy approximation in fitting our model, and the form of the fixed point equations that define the minimum of the Bethe free energy. The material in this section draws from a number of references (for instance Yedidia et al., 2001, 2005; Wainwright & Jordan, 2008; Koller & Friedman, 2009), but is presented in a form specific to our case, since we are neither in a discrete domain nor using exponential families.

The parameters of a pairwise Markov random field (MRF) can be learned by maximizing the log-likelihood of the model $\mathbb{P}$ with respect to true underlying distribution $\mathbb{P}^\star$. Denote the model by

$$\mathbb{P}(\boldsymbol{X}) := \frac{1}{Z} \prod_{(s,t)\in\mathcal{E}} \Psi_{st}(X_s, X_t) \prod_{s\in\mathcal{V}} \Psi_s(X_s)$$

where $Z := \int_{\boldsymbol{X}} \prod_{(s,t)\in\mathcal{E}} \Psi_{st}(X_s, X_t) \prod_{s\in\mathcal{V}} \Psi_s(X_s)$ is the partition function that normalizes the distribution. The model parameters $\{\Psi_{st}(X_s, X_t), \Psi_s(X_s)\}$ can be estimated by maximizing

$$\begin{aligned}\mathcal{L} &= \mathbb{E}_{\boldsymbol{X}\sim\mathbb{P}^\star(\boldsymbol{X})} \left[\log \mathbb{P}(\boldsymbol{X})\right] \\ &= \mathbb{E}_{\boldsymbol{X}\sim\mathbb{P}^\star(\boldsymbol{X})} \left[\sum_{(s,t)\in\mathcal{E}} \log \Psi_{st}(X_s, X_t) + \sum_{s\in\mathcal{V}} \log \Psi_s(X_s) - \log Z\right]. \end{aligned} \tag{1}$$

Define $\widetilde{\Psi}_{st}(X_s, X_t) := \log \Psi_{st}(X_s, X_t)$ and $\widetilde{\Psi}_s(X_s) := \log \Psi_s(X_s)$. Setting the derivatives of $\mathcal{L}$ with respect to $\left\{\widetilde{\Psi}_{st}(X_s, X_t), \widetilde{\Psi}_s(X_s)\right\}$ to zero, we have

$$\frac{\partial \mathcal{L}}{\partial \widetilde{\Psi}_s(X_s)} = \mathbb{P}^\star(X_s, X_t) - \frac{\partial \log Z}{\partial \widetilde{\Psi}_{st}(X_s, X_t)} = 0, \tag{2}$$

$$\frac{\partial \mathcal{L}}{\partial \widetilde{\Psi}_s(X_s)} = \mathbb{P}^\star(X_s) - \frac{\partial \log Z}{\partial \widetilde{\Psi}_s(X_s)} = 0. \tag{3}$$

For a general pairwise MRF on a loopy graph, computing the log-partition function, $\log Z$, is intractable. Following e.g. Yedidia et al. (2001, 2005) and Koller & Friedman (2009, Ch. 11), $\log Z$ may be approximated as a minimum

of the Bethe free energy with respect to a new set of parameters $\{b_{st}, b_s\}$,

$$F(\{b_{st}, b_s\}) = \sum_{(s,t)\in\mathcal{E}} \int_{\mathcal{X}} \int_{\mathcal{X}} b_{st}(X_s, X_t) \left[\log b_{st}(X_s, X_t) - \log \Psi_{st}(X_s, X_t)\Psi_s(X_s)\Psi_t(X_t)\right] dX_s dX_t$$
$$- \sum_{s\in\mathcal{V}} (d_s - 1) \int_{\mathcal{X}} b_s(X_s) \left[\log b_s(X_s) - \log \Psi_s(X_s)\right] dX_s, \tag{4}$$

subject to normalization and marginalization constraints, $\int_{\mathcal{X}} b_s(X_s) dX_s = 1$, $\int_{\mathcal{X}} b_{st}(X_s, X_t) dX_s = b_t(X_t)$. Let $F^\star := \min_{\{b_{st}, b_s\}} F$ (we note that Bethe free energy is not convex, and hence there can be multiple local minima. Our reasoning does not require constructing a specific local minimum, and therefore we simply write $F^\star$). The zero gradient conditions on the partial derivatives of $\mathcal{L}$ are then approximated as

$$\frac{\partial \mathcal{L}}{\partial \widetilde{\Psi}_s(X_s)} \approx \mathbb{P}^\star(X_s, X_t) - \frac{\partial F^\star}{\partial \widetilde{\Psi}_{st}(X_s, X_t)} = 0, \tag{5}$$
$$\frac{\partial \mathcal{L}}{\partial \widetilde{\Psi}_s(X_s)} \approx \mathbb{P}^\star(X_s) - \frac{\partial F^\star}{\partial \widetilde{\Psi}_s(X_s)} = 0. \tag{6}$$

Since F is a linear function of $\left\{\widetilde{\Psi}_{st}(X_s, X_t), \widetilde{\Psi}_s(X_s)\right\}$ for every fixed $\{b_{st}, b_s\}$, Danskin's theorem (Bertsekas, 1999, p. 717) gives us a way to compute the partial derivatives of $F^\star$. These are

$$\frac{\partial F^\star}{\partial \widetilde{\Psi}_{st}(X_s, X_t)} = \frac{\partial F(\{b^\star_{st}, b^\star_s\})}{\partial \widetilde{\Psi}_{st}(X_s, X_t)} = b^\star_{st}(X_s, X_t), \tag{7}$$
$$\frac{\partial F^\star}{\partial \widetilde{\Psi}_s(X_s)} = \frac{\partial F(\{b^\star_{st}, b^\star_s\})}{\partial \widetilde{\Psi}_s(X_s)} = b^\star_s(X_s), \tag{8}$$

where $\{b^\star_{st}, b^\star_s\} := \operatorname{argmin}_{\{b_{st}, b_s\}} F$. Therefore, according to (5) and (6), learning a pairwise MRF using the Bethe energy variational approximation to the log partition function results in the following matching conditions,

$$\mathbb{P}^\star(X_s, X_t) = b^\star_{st}(X_s, X_t), \tag{9}$$
$$\mathbb{P}^\star(X_s) = b^\star_s(X_s). \tag{10}$$

We now introduce the notion of belief propagation as a means of finding the minima of the Bethe free energy. This will in turn lead to local marginal consistency conditions for learning with BP. Yedidia et al. (2001) showed that the fixed point of F (and therefore the global minimum $\{b^\star_{st}, b^\star_s\}$) must satisfy the relations

$$b^\star_{st}(X_s, X_t) = \alpha \Psi_{st}(X_s, X_t)\Psi_s(X_s)\Psi_t(X_t) \prod_{u\in\Gamma_s\setminus t} m^\star_{us}(X_s) \prod_{v\in\Gamma_t\setminus s} m^\star_{vt}(X_t), \tag{11}$$
$$b^\star_s(X_s) = \alpha \Psi_s(X_s) \prod_{u\in\Gamma_s} m^\star_{us}(X_s), \tag{12}$$

where α denotes a normalization constant and $\{m^\star_{ts}\}$ are the fixed point messages,

$$m^\star_{ts}(X_s) = \alpha \int_{\mathcal{X}} \Psi_{st}(X_s, X_t)\Psi_t(X_t) \prod_{u\in\Gamma_t\setminus s} m^\star_{ut}(X_t)\, dX_t. \tag{13}$$

Thus,

$$\mathbb{P}^\star(X_s, X_t) = \alpha \Psi_{st}(X_s, X_t)\Psi_s(X_s)\Psi_t(X_t) \prod_{u\in\Gamma_s\setminus t} m^\star_{us}(X_s) \prod_{v\in\Gamma_t\setminus s} m^\star_{vt}(X_t), \tag{14}$$
$$\mathbb{P}^\star(X_s) = \alpha \Psi_s(X_s) \prod_{u\in\Gamma_s} m^\star_{us}(X_s). \tag{15}$$

Combining these relations and assuming that $\mathbb{P}^\star(X_s)$ and $m^\star_{ts}(X_s)$ are strictly positive, we can also obtain the consistent relation for the local conditionals,

$$\mathbb{P}^\star(X_t|X_s) = \frac{\mathbb{P}^\star(X_s, X_t)}{\mathbb{P}^\star(X_s)} = \frac{\Psi_{st}(X_s, X_t)\Psi_s(X_t)\prod_{u\in\Gamma_t\setminus s} m^\star_{ut}(X_t)}{m^\star_{ts}(X_s)}. \tag{16}$$

6 BP Inference Using Learned Potentials

The inference problem in pairwise MRFs is to compute the marginals or the log partition function for the model with learned potentials. Belief propagation is an iterative algorithm for performing approximate inference in MRFs. BP can also be viewed as an iterative algorithm for minimizing the Bethe free energy approximation to the log partition function. The results of this algorithm are a set of beliefs which can be used for obtaining the MAP assignment of the corresponding variables.

The BP message update (with the learned potentials) is

$$m_{ts}(X_s) = \alpha \int_{\mathcal{X}} \Psi_{st}(X_s, X_t)\Psi_t(X_t) \prod_{u \in \Gamma_t \setminus s} m_{ut}(X_t)\ dX_t, \tag{17}$$

and at any iteration, the beliefs can be computed using the current messages,

$$\mathbb{B}_{st}(X_s, X_t) = \alpha \Psi_{st}(X_s, X_t)\Psi_s(X_s)\Psi_t(X_t) \prod_{u \in \Gamma_s \setminus t} m_{us}(X_s) \prod_{v \in \Gamma_t \setminus s} m_{vt}(X_t), \tag{18}$$

$$\mathbb{B}_s(X_s) = \alpha \Psi_s(X_s) \prod_{u \in \Gamma_s} m_{us}(X_s). \tag{19}$$

To see how the message update equation can be expressed using the true local conditional $\mathbb{P}^\star(X_t|X_s)$, we divide both size of (17) by the fixed point message $m^\star_{ts}(X_s)$ during BP learning stage, and introduce $1 = \frac{\prod_{u \in \Gamma_t \setminus s} m^\star_{ut}(X_t)}{\prod_{u \in \Gamma_t \setminus s} m^\star_{ut}(X_t)}$. The message update equation in (17) can then be re-written as

$$\frac{m_{ts}(X_s)}{m^\star_{ts}(X_s)} = \int_{\mathcal{X}} \frac{\Psi_{st}(X_s, X_t)\Psi_t(X_t) \prod_{u \in \Gamma_t \setminus s} m_{ut}(X_t)}{m^\star_{ts}(X_s)} \prod_{u \in \Gamma_t \setminus s} \frac{m_{ut}(X_t)}{m^\star_{ut}(X_t)}\ dX_t. \tag{20}$$

The belief at any iteration becomes

$$\mathbb{B}_s(X_s) = \alpha \Psi_s(X_s) \prod_{u \in \Gamma_s} m_{us}(X_s) = \left(\prod_{u \in \Gamma_s} \frac{m_{st}(X_s)}{m^\star_{st}(X_s)} \right) \left(\alpha \Psi_s(X_s) \prod_{u \in \Gamma_s} m^\star_{st}(X_s) \right). \tag{21}$$

We reparameterize the message as

$$m_{st}(X_t) \longleftarrow \frac{m_{st}(X_t)}{m^\star_{st}(X_t)}\ . \tag{22}$$

Since the potentials are learned via BP, we can use the relation in (16) to obtain

$$m_{ts}(X_s) = \int_{\mathcal{X}} \mathbb{P}^\star(X_t|X_s) \prod_{u \in \Gamma_t \setminus s} m_{ut}(X_t)\ dX_t. \tag{23}$$

Similarly, we obtain from (15) that

$$\mathbb{B}_s(X_s) = \left(\prod_{u \in \Gamma_s} m_{st}(X_s) \right) \mathbb{P}^\star(X_s). \tag{24}$$

Messages from Evidence Node Given evidence x_t at node X_t, the outgoing message from X_t to X_s is $m_{ts}(X_s) = \alpha \Psi_{st}(X_s, x_t)\Psi_t(x_t)$. Using similar reasoning to the case of an internal node, we have

$$\frac{m_{ts}(X_s)}{m^\star_{ts}(X_s)} = \frac{\Psi_{st}(X_s, x_t)\Psi_t(x_t)}{m^\star_{ts}(X_s)} \tag{25}$$

$$= \frac{\Psi_{st}(X_s, x_t)\Psi_t(x_t) \prod_{u \in \Gamma_t \setminus s} m^\star_{ut}(x_t)}{m^\star_{ts}(X_s)} \frac{1}{\prod_{u \in \Gamma_t \setminus s} m^\star_{ut}(x_t)} \tag{26}$$

$$= \mathbb{P}^\star(x_t|X_s) \frac{1}{\prod_{u \in \Gamma_t \setminus s} m^\star_{ut}(x_t)} \tag{27}$$

$$\propto \mathbb{P}^\star(x_t|X_s) \tag{28}$$

where $\prod_{u \in \Gamma_t \setminus s} m^\star_{ut}(x_t)$ is constant given a fixed value $X_t = x_t$. Reparametrizing the message $m_{ts}(X_s) \leftarrow \frac{m_{ts}(X_s)}{m^\star_{ts}(X_s)}$, the outgoing message from the evidence node is simply the true likelihood function evaluated at x_t.

7 A Note on Kernelization of Gaussian BP

In this section, we consider the problem of defining a joint Gaussian graphical model in the feature space induced by a kernel. We follow Bickson (2008) in our presentation of the original Gaussian BP setting. We will show that assuming a Gaussian in an infinite feature space leads to challenges in interpretation and estimation of the model.

Consider a pairwise MRF,

$$\mathbb{P}(\boldsymbol{X}) = \prod_{s \in \mathcal{V}} \Psi_s(X_s) \prod_{(s,t) \in \mathcal{E}} \Psi_{st}(X_s, X_t). \tag{29}$$

In the case of the Gaussian, the probability density function takes the form

$$\begin{aligned} \mathbb{P}(\boldsymbol{X}) &\propto \exp\left(-\tfrac{1}{2}(\mathbf{X}-\mu)^\top \mathbf{A}(\mathbf{X}-\mu)\right) \\ &\propto \exp\left(-\tfrac{1}{2}\mathbf{X}^\top \mathbf{A}\mathbf{X} - \mathbf{b}^\top \mathbf{X}\right), \end{aligned}$$

where $\mathbf{A} = \mathbf{C^{-1}}$ is the precision matrix, and

$$\mathbf{A}\mu = \mathbf{b}.$$

Putting this in the form (29), the node and edge potentials are written

$$\Psi_s(X_s) \triangleq \exp\left(-\tfrac{1}{2} X_s^\top A_{ss} X_s + b_s X_s\right) \tag{30}$$

and

$$\Psi_{st}(X_s, X_t) \triangleq \exp\left(-X_s^\top A_{st} X_t\right). \tag{31}$$

We now consider how these operations would appear in Hilbert space. In this case, we would have

$$\Psi_s(X_s) := \exp\left(-\tfrac{1}{2}\left\langle \phi(X_s), A_{ss}\phi(X_s)\right\rangle_{\mathcal{F}} + \left\langle b_s, \phi(X_s)\right\rangle_{\mathcal{F}}\right),$$

and

$$\Psi_{st}(X_s, X_t) := \exp\left(-\left\langle \phi(X_s), A_{st}\phi(X_t)\right\rangle_{\mathcal{F}}\right).$$

We call the A_{ss} and A_{st} *precision operators*, by analogy with the finite dimensional case. At this point, we already encounter a potential difficulty in kernelizing Gaussian BP: how do we learn the operators A_{ss}, b_s and A_{st} from data? We could in principle define a covariance operator $\mathbf{C}$ with (s,t)th block the pairwise covariance operator C_{st}, but A_{st} would then be the (s,t)th block of $\mathbf{C^{-1}}$, which is difficult to compute. As we shall see below, however, these operators appear in the BP message updates.

Next, we describe a message passing procedure for the Gaussian potentials in (30) and (31). The message from t to s is written

$$m_{ts}(X_s) = \int_{\mathcal{X}} \Psi_{st}(X_s, X_t) \underbrace{\Psi_t(X_t) \prod_{u \in \Gamma_t \setminus s} m_{ut}(X_t)}_{(a)} \, dX_t.$$

We first consider term (a) in the above. We will assume, with justification to follow, that $m_{ut}(X_t)$ takes the form

$$m_{ut}(X_t) \propto \exp\left(-\tfrac{1}{2} X_t^\top P_{ut} X_t + \mu_{ut}^\top X_t\right),$$

where the terms P_{ut} and μ_{ut} are defined by recursions specified below (we retain linear algebraic notation for simplicity). It follows that $\Psi_t(X_t)\prod_{u\in\Gamma_t\setminus s} m_{ut}(X_t)$ is proportional to a Gaussian,

$$\Psi_t(X_t) \prod_{u \in \Gamma_t \setminus s} m_{ut}(X_t) \propto \exp\left(-\tfrac{1}{2} X_t^\top P_{t\setminus s} X_t + \mu_{t\setminus s}^\top X_t\right),$$

where we define the intermediate operators

$$\mu_{t\setminus s} := \mu_t + \sum_{u \in \Gamma_t \setminus s} \mu_{ut}$$

and

$$P_{t\setminus s} := A_{ss} + \sum_{u \in \Gamma_t \setminus s} P_{ut}.$$

To compute the message $m_{ts}(X_s)$, we integrate

$$\begin{aligned} m_{ts}(X_s) &= \int_{\mathcal{X}} \Psi_{st}(X_s, X_t)\Psi_t(X_t) \prod_{u\in\Gamma_t\setminus s} m_{ut}(X_t)\, dX_t \\ &= \int_{\mathcal{X}} \exp\left(-X_t A_{ts} X_s\right) \exp\left(-\tfrac{1}{2} X_t^\top P_{t\setminus s} X_t + \mu_{t\setminus s}^\top X_t\right)\, dX_t \end{aligned} \tag{32}$$

Completing the square, we get the parameters of the message m_{ts} in the standard form,

$$P_{ts} = -A_{ts}^\top P_{t\setminus s}^{-1} A_{ts} \tag{33}$$

$$\mu_{ts} = -\mu_{t\setminus s}^\top P_{t\setminus s}^{-1} A_{ts}. \tag{34}$$

There are two main difficulties in implementing the above procedure in feature space. First, it is not clear how to learn the precision operators from the data. Second, we need to invert these precision operators. Thus, it remains a challenging open question to define Gaussian BP in feature space. The feature space Gaussian BP updates may be contrasted with the kernel BP updates we propose in the main text. The latter have regularized closed form empirical estimates, and they are very different from the Gaussian BP form in (32) and parameter updates in (33) and (34).

8 Message Error Incurred by the Additional Feature Approximation

We bound the difference between the estimated conditional embedding operator $\widehat{\mathcal{U}}_{X_t^\otimes|X_s}$ and its counterpart $\widetilde{\mathcal{U}}_{X_t^\otimes|X_s}$ after further feature approximation. Assume $\|\phi(x)\|_{\mathcal{F}} \le 1$, and define the tensor feature $\xi(x) := \bigotimes_{u\setminus s} \phi(x)$. Denote by $\tilde{\xi}(x)$ and $\tilde{\phi}(x)$ the respective approximations of $\xi(x)$ and ϕ. Furthermore, let the approximation error after the incomplete QR decomposition be $\epsilon = \max\left\{\max_{\mathcal{X}} \left\|\phi(x) - \tilde{\phi}(x)\right\|_{\mathcal{F}}, \ \max_{\mathcal{X}} \left\|\xi(X) - \tilde{\xi}(x)\right\|_{\mathcal{H}}\right\}$. It follows that

$$\left\|\widehat{\mathcal{U}}_{X_t^\otimes|X_s} - \widetilde{\mathcal{U}}_{X_t^\otimes|X_s}\right\|_{HS} \tag{35}$$

$$\le \left\|\widehat{\mathcal{C}}_{X_t^\otimes X_s}(\widehat{\mathcal{C}}_{X_sX_s} + \lambda_m I)^{-1} - \widetilde{\mathcal{C}}_{X_t^\otimes X_s}(\widetilde{\mathcal{C}}_{X_sX_s} + \lambda_m I)^{-1}\right\|_{HS} \tag{36}$$

$$\le \left\|(\widehat{\mathcal{C}}_{X_t^\otimes X_s} - \widetilde{\mathcal{C}}_{X_t^\otimes X_s})(\widehat{\mathcal{C}}_{X_sX_s} + \lambda_m I)^{-1}\right\|_{HS} + \left\|\widetilde{\mathcal{C}}_{X_t^\otimes X_s}\left[(\widetilde{\mathcal{C}}_{X_sX_s} + \lambda_m I)^{-1} - (\widehat{\mathcal{C}}_{X_sX_s} + \lambda_m I)^{-1}\right]\right\|_{HS} \tag{37}$$

$$\le \frac{1}{\lambda_m}\left\|\widehat{\mathcal{C}}_{X_t^\otimes X_s} - \widetilde{\mathcal{C}}_{X_t^\otimes X_s}\right\|_{HS} + \frac{1}{\lambda_m^{3/2}}\left\|\widehat{\mathcal{C}}_{X_sX_s} - \widetilde{\mathcal{C}}_{X_sX_s}\right\|_{HS}. \tag{38}$$

For the first term,

$$\frac{1}{\lambda_m}\left\|\widehat{\mathcal{C}}_{X_t^\otimes X_s} - \widetilde{\mathcal{C}}_{X_t^\otimes X_s}\right\|_{HS} \tag{39}$$

$$= \frac{1}{\lambda_m}\left\|\frac{1}{m}\sum_i \xi(x_s^i)\phi(x_s^i)^\top - \frac{1}{m}\sum_i \tilde{\xi}(x_s^i)\tilde{\phi}(x_s^i)^\top\right\|_{HS} \tag{40}$$

$$\le \frac{1}{\lambda_m}\frac{1}{m}\sum_i \left\|\xi(x_s^i)\phi(x_s^i)^\top - \tilde{\xi}(x_s^i)\tilde{\phi}(x_s^i)^\top\right\|_{HS} \tag{41}$$

$$\le \frac{1}{\lambda_m}\max_i \left\|\xi(x_s^i)\phi(x_s^i)^\top - \tilde{\xi}(x_s^i)\tilde{\phi}(x_s^i)^\top\right\|_{HS} \tag{42}$$

$$\le \frac{1}{\lambda_m}\max_i \left\{\left\|\xi(x_s^i)\phi(x_s^i)^\top - \tilde{\xi}(x_s^i)\phi(x_s^i)^\top\right\|_{HS} + \left\|\tilde{\xi}(x_s^i)\phi(x_s^i)^\top - \tilde{\xi}(x_s^i)\tilde{\phi}(x_s^i)^\top\right\|_{HS}\right\} \tag{43}$$

$$\le \frac{2\epsilon}{\lambda_m}. \tag{44}$$

Similarly, for the second term,

$$\frac{1}{\lambda_m^{3/2}}\left\|\widehat{\mathcal{C}}_{X_sX_s} - \widetilde{\mathcal{C}}_{X_sX_s}\right\|_{HS} \le \frac{2\epsilon}{\lambda_m^{3/2}}. \tag{45}$$

Combining the results, we obtain

$$\left\| \widehat{\mathcal{U}}_{X_t^{\otimes}|X_s} - \widetilde{\mathcal{U}}_{X_t^{\otimes}|X_s} \right\|_{HS} \leq 2\epsilon(\lambda_m^{-1} + \lambda_m^{-3/2}). \tag{46}$$

References

Bertsekas, D. P. (1999). *Nonlinear Programming*. Belmont, MA: Athena Scientific, second edn.

Bickson, D. (2008). *Gaussian Belief Propagation: Theory and Application*. Ph.D. thesis, The Hebrew University of Jerusalem.

Gonzalez, J., Low, Y., & Guestrin, C. (2009). Residual splash for optimally parallelizing belief propagation. In *In Artificial Intelligence and Statistics (AISTATS)*. Clearwater Beach, Florida.

Hsu, D., Kakade, S. M., Langford, J., & Zhang, T. (2009). Multi-label prediction via compressed sensing. In *Advances in Neural Information Processing Systems 21*.

Ihler, A. (2003). KDE Matlab ToolBox. `http://www.ics.uci.edu/ ihler/code/`.

Ihler, A., & McAllester, D. (2009). Particle belief propagation. In *AISTATS*, 256–263.

Ihler, E. T., Sudderth, E. B., Freeman, W. T., & Willsky, A. S. (2003). Efficient multiscale sampling from products of gaussian mixtures. In *Advances in Neural Information Processing Systems 17*.

Koller, D., & Friedman, N. (2009). *Probabilistic Graphical Models: Principles and Techniques*. MIT Press.

Sudderth, E., Ihler, A., Freeman, W., & Willsky, A. (2003). Nonparametric belief propagation. In *CVPR*.

Sugiyama, M., Takeuchi, I., Suzuki, T., Kanamori, T., Hachiya, H., & Okanohara, D. (2010). Conditional density estimation via least-squares density ratio estimation. In *AISTATS*, 781–788.

Wainwright, M., Jaakkola, T., & Willsky, A. (2003). Tree-reweighted belief propagation and approximate ML estimation by pseudo-moment matching. In *9th Workshop on Artificial Intelligence and Statistics*.

Wainwright, M. J., & Jordan, M. I. (2008). Graphical models, exponential families, and variational inference. *Foundations and Trends in Machine Learning*, *1*(1 – 2), 1 – 305.

Yedidia, J., Freeman, W., & Weiss, Y. (2005). Constructing free-energy approximations and generalized belief propagation algorithms. *IEEE Transactions on Information Theory*, *51*(7), 2282 – 2312.

Yedidia, J. S., Freeman, W. T., & Weiss, Y. (2001). Generalized belief propagation. In T. K. Leen, T. G. Dietterich, & V. Tresp, eds., *Advances in Neural Information Processing Systems 13*, 689–695. MIT Press.