Kernel Belief Propagation

Le Song,[1] **Arthur Gretton,**[1,2] **Danny Bickson,**[1] **Yucheng Low,**[1] **Carlos Guestrin**[1]
[1] School of Computer Science, CMU; [2]Gatsby Computational Neuroscience Unit & MPI for Biological Cybernetics

Abstract

We propose a nonparametric generalization of belief propagation, Kernel Belief Propagation (KBP), for pairwise Markov random fields. Messages are represented as functions in a reproducing kernel Hilbert space (RKHS), and message updates are simple linear operations in the RKHS. KBP makes none of the assumptions commonly required in classical BP algorithms: the variables need not arise from a finite domain or a Gaussian distribution, nor must their relations take any particular parametric form. Rather, the relations between variables are represented implicitly, and are learned nonparametrically from training data. KBP has the advantage that it may be used on any domain where kernels are defined ($\mathbb{R}^d$, strings, groups), even where explicit parametric models are not known, or closed form expressions for the BP updates do not exist. The computational cost of message updates in KBP is polynomial in the training data size. We also propose a constant time approximate message update procedure by representing messages using a small number of basis functions. In experiments, we apply KBP to image denoising, depth prediction from still images, and protein configuration prediction: KBP is faster than competing classical and nonparametric approaches (by orders of magnitude, in some cases), while providing significantly more accurate results.

1 Introduction

Belief propagation is an inference algorithm for graphical models that has been widely and successfully applied in a great variety of domains, including vision (Sudderth et al., 2003), protein folding (Yanover & Weiss, 2002), and turbo decoding (McEliece et al., 1998). In these applications, the variables are usually assumed either to be finite dimensional, or in continuous cases, to have a Gaussian distribution (Weiss & Freeman, 2001). In many applications of graphical models, however, the variables of interest are naturally specified by continuous, non-Gaussian distributions. For example, in constructing depth maps from 2D images, the depth is both continuous valued and has a multimodal distribution. Likewise, in protein folding, angles are modeled as continuous valued random variables, and are predicted from amino acid sequences. In general, multimodalities, skewness, and other non-Gaussian statistical features are present in a great many real-world problems. The corresponding inference procedures for parametric models typically involve integrals for which no closed form solutions exist, and are without computationally tractable exact message updates. Worse still, parametric models for the relations between the variables may not even be known, or may be prohibitively complex.

Our first contribution in this paper is a novel generalization of belief propagation for pairwise Markov random fields, *Kernel BP*, based on a reproducing kernel Hilbert space (RKHS) representation of the relations between random variables. This extends earlier work of Song et al. (2010) on inference for trees to the case of graphs with loops. The algorithm consists of two parts, both nonparametric: first, we *learn* RKHS representations of the relations between variables directly from training data, which removes the need for an explicit parametric model. Second, we propose a belief propagation algorithm for *inference* based on these learned relations, where each update is a linear operation in the RKHS (although the relations themselves may be highly nonlinear in the original space of the variables). Our approach applies not only to continuous-valued non-Gaussian variables, but also generalizes to strings and graphs (Schölkopf et al., 2004), groups (Fukumizu et al., 2009), compact manifolds (Wendland, 2005, Chapter 17), and other domains on which kernels may be defined.

A number of alternative approaches have been developed to perform inference in the continuous-valued non-Gaussian setting. Sudderth et al. (2003) proposed an approximate belief propagation algorithm for pairwise Markov random fields, where the parametric forms of the node and edge potentials are supplied in advance, and the messages are approximated as mixtures of Gaussians: we refer to this approach as *Gaussian Mixture BP* (this method was introduced as "nonparametric BP", but it is in fact a Gaussian mixture approach). Instead of mixtures of Gaussians, Ihler & McAllester (2009) used particles to approximate the messages, resulting in the *Particle BP* algorithm. Both Gaussian mixture BP and particle BP assume the potentials

---

Appearing in Proceedings of the 14^{th} International Conference on Artificial Intelligence and Statistics (AISTATS) 2011, Fort Lauderdale, FL, USA. Volume 15 of JMLR: W&CP 15.

to be pre-specified by the user: the methods described are purely approximate message update procedures, and do *not* learn the model from training data. By contrast, kernel BP learns the model, is computationally tractable even before approximations are made, and leads to an entirely different message update formula than the Gaussian Mixture and Particle representations.

A direct implementation of kernel BP has a reasonable computational cost: each message update costs $O(m^2 d_{\max})$ when computed exactly, where m is the number of training examples and $d_{\max}$ is the maximum degree of a node in the graphical model. For massive data sets and numbers of nodes, as occur in image processing, this cost might still be expensive. Our second contribution is a novel *constant time* approximate message update procedure, where we express the messages in terms of a small number $\ell \ll m$ of representative RKHS basis functions learned from training data. Following an initialization cost linear in m, the cost per message update is decreased to $O(\ell^2 d_{\max})$, independent of the number of training points m. Even without these approximate constant time updates, kernel BP is substantially faster than Gaussian mixture BP and particle BP. Indeed, an exact implementation of Gaussian mixture BP would have an exponentially increasing computational and storage cost with number of iterations. In practice, both Gaussian mixture and particle BP require a Monte Carlo resampling procedure at every node of the graphical model.

Our third contribution is a thorough evaluation of kernel BP against other nonparametric BP approaches. We apply both kernel BP and competing approaches to an image denoising problem, depth prediction from still images, protein configuration prediction, and paper topic inference from citation networks: these are all large-scale problems, with continuous-valued or structured random variables having complex underlying probability distributions. In all cases, kernel BP performs outstandingly, being orders of magnitude faster than both Gaussian mixture BP and particle BP, and returning more accurate results.

2 Markov Random Fields And Belief Propagation

We begin with a short introduction to pairwise Markov random fields (MRFs) and the belief propagation algorithm. A pairwise Markov random field (MRF) is defined on an undirected graph $\mathcal{G} := (\mathcal{V}, \mathcal{E})$ with nodes $\mathcal{V} := \{1, \ldots, n\}$ connected by edges in $\mathcal{E}$. Each node $s \in \mathcal{V}$ is associated with a random variable X_s on the domain$\mathcal{X}$ (we assume a common domain for ease of notation, but in practice the domains can be different), and $\Gamma_s := \{t|(s,t) \in \mathcal{E}\}$ is the set of neighbors of node s with size $d_s := |\Gamma_s|$. In a pairwise MRF, the joint distribution of the variables $\boldsymbol{X} := \{X_1, \ldots, X_{|\mathcal{V}|}\}$ is assumed to factorize according to a model $\mathbb{P}(\boldsymbol{X}) = \frac{1}{Z} \prod_{(s,t)\in\mathcal{E}} \Psi_{st}(X_s, X_t) \prod_{s\in\mathcal{V}} \Psi_s(X_s)$, where $\Psi_s(X_s)$ and $\Psi_{st}(X_s, X_t)$ are node and edge potentials respectively, and Z is the partition function that normalizes the distribution.

The inference problem in an MRF is defined as calculating the marginals $\mathbb{P}(X_s)$ for nodes $s \in \mathcal{V}$ and $\mathbb{P}(X_s, X_t)$ for edges $(s,t) \in \mathcal{E}$. The marginal $\mathbb{P}(X_s)$ not only provides a measure of uncertainty of X_s, but also leads to a point estimate $x_s^\star := \operatorname{argmax} \mathbb{P}(X_s)$. Belief Propagation (BP) is an iterative algorithm for performing inference in MRFs (Pearl, 1988). BP represents intermediate results of marginalization steps as messages passed between adjacent nodes: a message m_{ts} from t to s is calculated based on messages m_{ut} from all neighboring nodes u of t besides s, i.e.,

$$m_{ts}(X_s) = \int_{\mathcal{X}} \Psi_{st}(X_s, X_t)\Psi_t(X_t) \prod_{u\setminus s} m_{ut}(X_t) dX_t. \quad (1)$$

Note that we use $\prod_{u\setminus s}$ to denote $\prod_{u\in\Gamma_t\setminus s}$, where it is understood that the indices range over all neighbors u of t except s. This notation also applies to operations other than the product. The update in (1) is iterated across all nodes until a fixed point, $m_{ts}^\star$, for all messages is reached. The resulting node beliefs (estimates of node marginals) are given by $\mathbb{B}(X_s) \propto \Psi_s(X_s) \prod_{t\in\Gamma_s} m_{ts}^\star(X_s)$.

For acyclic or tree-structured graphs, BP results in node beliefs $\mathbb{B}(X_s)$ that converge to the node marginals $\mathbb{P}(X_s)$. This is generally not true for graphs with cycles. In many applications, however, the resulting loopy BP algorithm exhibits excellent empirical performance (Murphy et al., 1999). Several theoretical studies have also provided insight into the approximations made by loopy BP, partially justifying its application to graphs with cycles (Wainwright & Jordan, 2008; Yedidia et al., 2001).

The learning problem in MRFs is to estimate the node and edge potentials, which is often done by maximizing the expected log-likelihood $\mathbb{E}_{\boldsymbol{X}\sim\mathbb{P}^\star(\boldsymbol{X})}[\log \mathbb{P}(\boldsymbol{X})]$ of the model $\mathbb{P}(\boldsymbol{X})$ with respect to the true distribution $\mathbb{P}^\star(\boldsymbol{X})$. The resulting optimization problem usually requires solving a sequence of inference problems as an inner loop (Koller & Friedman, 2009); BP is often deployed for this purpose.

3 Properties of Belief Propagation

Our goal is to develop a nonparametric belief propagation algorithm, where the potentials are nonparametric functions learned from data, such that multimodal and other non-Gaussian statistical features can be captured. Most crucially, these potentials must be represented in such a way that the message update in (1) is computationally tractable. Before we go into the details of our kernel BP algorithm, we will first explain a key property of BP, which relates message updates to conditional expectations. When the messages are RKHS functions, these expectations can be evaluated efficiently.

Yedidia et al. (2001) showed BP to be an iterative algorithm for minimizing the Bethe free energy, which is a variational approximation to the log-partition function, $\log Z$, in the MRF model $\mathbb{P}(\boldsymbol{X})$. The beliefs are fixed points of BP algorithm if and only if they are zero gradient points of the Bethe free energy. In Section 5 of the Appendix, we show maximum likelihood learning of MRFs using BP results in the following equality, which relates the conditional of the true distribution, the learned potentials, and the fixed point messages,

$$\mathbb{P}^{\star}(X_t|X_s) = \frac{\Psi_{st}(X_s, X_t)\Psi_t(X_t)\prod_{u\backslash s} m^{\star}_{ut}(X_t)}{m^{\star}_{ts}(X_s)}, \quad (2)$$

where $\mathbb{P}^{\star}(X_s)$ and $m^{\star}_{ts}(X_s)$ are assumed strictly positive. Wainwright et al. (2003, Section 4) derived a similar relation, but for discrete variables under the exponential family setting. By contrast, we do not assume an exponential family model, and our reasoning applies to continuous variables. A further distinction is that Wainwright et al. specify the node potential $\Psi_s(X_s) = \mathbb{P}^{\star}(X_s)$ and edge potential $\Psi(X_s, X_t) = \mathbb{P}^{\star}(X_s, X_t)\mathbb{P}^{\star}(X_s)^{-1}\mathbb{P}^{\star}(X_t)^{-1}$, which represent just one possible choice among many that satisfies (2). Indeed, we next show that in order to run BP for subsequent inference, we do not need to commit to a particular choice for $\Psi_s(X_s)$ and $\Psi(X_s, X_t)$, nor do we need to optimize to learn $\Psi_s(X_s)$ and $\Psi(X_s, X_t)$.

We start by dividing both sides of (1) by $m^{\star}_{ts}(X_s)$, and introducing $1 = \prod_{u\backslash s} \frac{m^{\star}_{ut}(X_t)}{m^{\star}_{ut}(X_t)}$,

$$\frac{m_{ts}(X_s)}{m^{\star}_{ts}(X_s)} = \int_{\mathcal{X}} \frac{\Psi_{st}(X_s, X_t)\Psi_t(X_t)\prod_{u\backslash s} m^{\star}_{ut}(X_t)}{m^{\star}_{ts}(X_s)} \times \prod_{u\backslash s} \frac{m_{ut}(X_t)}{m^{\star}_{ut}(X_t)}\, dX_t. \quad (3)$$

We next substitute the BP fixed point relation (2) into (3), and reparametrize the messages $m_{ts}(X_s) \leftarrow \frac{m_{ts}(X_s)}{m^{\star}_{ts}(X_s)}$, to obtain the following property for BP updates (see Section 6 in the Appendix for details):

Property 1 *If we learn an MRF using BP and subsequently use the learned potentials for inference, BP updates can be viewed as conditional expectations,*

$$m_{ts}(X_s) = \int_{\mathcal{X}} \mathbb{P}^{\star}(X_t|X_s) \prod_{u\backslash s} m_{ut}(X_t)\, dX_t = \mathbb{E}_{X_t|X_s}\left[\prod_{u\backslash s} m_{ut}(X_t)\right]. \quad (4)$$

Using similar reasoning, the node beliefs on convergence of BP take the form $\mathbb{B}(X_s) \propto \mathbb{P}^{\star}(X_s)\prod_{t\in\Gamma_s} m^{\star}_{ts}(X_s)$. In the absence of external evidence, a fixed point occurs at the true node marginals, i.e., $\mathbb{B}(X_s) \propto \mathbb{P}^{\star}(X_s)$ for all $s \in \mathcal{V}$. Typically there can be many evidence variables, and the belief is then an estimate of the true conditional distribution given the evidence.

The above property of BP immediately suggests that if belief propagation is the inference algorithm of choice, then MRFs can be learned very simply: given training data drawn from $\mathbb{P}^{\star}(\boldsymbol{X})$, the empirical conditionals $\widehat{\mathbb{P}}(X_t|X_s)$ are estimated (either in parametric form, or nonparametrically), and the conditional expectations are evaluated using these estimates. Evidence can also be incorporated straightforwardly: if an observation x_t is made at node t, the message from t to its neighbor s is simply the empirical likelihood function $m_{ts}(X_s) \propto \widehat{\mathbb{P}}(x_t|X_s)$, where we use lowercase to denote observed variables with fixed values, and capitalize unobserved random variables.

With respect to kernel belief propagation, our key insight from Property 1, however, is that we need not explicitly recover the empirical conditionals $\widehat{\mathbb{P}}(X_t|X_s)$ as an intermediate step, as long as we can compute the conditional expectation directly. We will pursue this approach next.

4 Kernel Belief Propagation

We develop a novel kernelization of belief propagation, based on Hilbert space embeddings of conditional distributions (Song et al., 2009), which generalizes an earlier kernel algorithm for exact inference on trees (Song et al., 2010). As might be expected, the kernel implementation of the BP updates in (4) is nearly identical to the earlier tree algorithm, the main difference being that we now consider graphs with loops, and iterate until convergence (rather than obtaining an exact solution in a single pass). This difference turns out to have major implications for the implementation: the earlier solution of Song et al. is polynomial in the sample size, which was not an issue for the the smaller trees considered by Song et al., but becomes expensive for the large, loopy graphical models we address in our experiments. We defer the issue of efficient implementation to Section 5, where we present a novel approximation strategy for kernel BP which achieves constant time message updates.

In the present section, we will provide a detailed derivation of kernel BP in accordance with Song et al. (2010). While the immediate purpose is to make the paper self-contained, there are two further important reasons: to provide the background necessary in understanding our efficient kernel BP updates in Section 5, and to demonstrate how kernel BP differs from the competing Gaussian mixture and particle based BP approaches in Section 6 (which was not addressed in earlier work on kernel tree graphical models).

4.1 Message Representations

We begin with a description of the properties of a message $m_{ut}(x_t)$, given it is in the reproducing kernel Hilbert space (RKHS) $\mathcal{F}$ of functions on the separable metric space $\mathcal{X}$ (Aronszajn, 1950; Schölkopf & Smola, 2002). As we will see, the advantage of this assumption is that the update procedure can be expressed as a linear operation in the RKHS, and results in new messages that are likewise RKHS functions. The RKHS $\mathcal{F}$ is defined in terms of a unique positive definite kernel $k(x_s, x'_s)$ with the reproducing property $\langle m_{ts}(\cdot), k(x_s, \cdot)\rangle_{\mathcal{F}} = m_{ts}(x_s)$, where $k(x_s, \cdot)$ indicates that one argument of the kernel is fixed at x_s. Thus, we can view the evaluation of message m_{ts} at any point $x_s \in \mathcal{X}$ as a linear operation in $\mathcal{F}$: we call $k(x_s, \cdot)$ the *representer of evaluation* at x_s, and use the shorthand $k(x_s, \cdot) = \phi(x_s)$. Note that $k(x_s, x'_s) = \langle \phi(x_s), \phi(x'_s) \rangle_{\mathcal{F}}$; the kernel encodes the degree of similarity between x_s and x'_s. The restriction of messages to RKHS functions need not be onerous: on compact domains, universal kernels (in the sense of Steinwart, 2001) are dense in the space of bounded continuous functions (e.g., the Gaussian RBF kernel $k(x_s, x'_s) = \exp(-\sigma \|x_s - x'_s\|^2)$ is universal). Kernels may be defined when dealing with random variables on additional domains, such as strings, graphs, and groups.

4.2 Kernel BP Message Updates

We next define a representation for message updates, under the assumption that messages are RKHS functions. For simplicity, we first establish a result for a three node chain, where the middle node t incorporates an incoming message from u, and then generates an outgoing message to s (we will deal with multiple incoming messages later). In this case, the outgoing message $m_{ts}(x_s)$ evaluated at x_s simplifies to $m_{ts}(x_s) = \mathbb{E}_{X_t|x_s}[m_{ut}(X_t)]$. Under some regularity conditions for the integral, we can rewrite message updates as inner products, $m_{ts}(x_s) = \mathbb{E}_{X_t|x_s}[\langle m_{ut}, \phi(X_t) \rangle_{\mathcal{F}}] = \langle m_{ut}, \mathbb{E}_{X_t|x_s}[\phi(X_t)] \rangle_{\mathcal{F}}$ using the reproducing property of the RKHS. We refer to $\mu_{X_t|x_s} := \mathbb{E}_{X_t|x_s}[\phi(X_t)] \in \mathcal{F}$ as the feature space embedding of the conditional distribution $\mathbb{P}(X_t|x_s)$. If we can estimate this quantity directly from data, we can perform message updates via a simple inner product, avoiding a two-step procedure where the conditional distribution is first estimated and the expectation then taken.

An expression for the conditional distribution embedding was proposed by Song et al. (2009). We describe this expression by analogy with the conditioning operation for a Gaussian random vector $z \sim \mathcal{N}(0, C)$, where we partition $z = (z_1^\top, z_2^\top)^\top$ such that $z_1 \in \mathbb{R}^d$ and $z_2 \in \mathbb{R}^{d'}$. Given the covariances $C_{11} := \mathbb{E}[z_1 z_1^\top]$ and $C_{12} := \mathbb{E}[z_1 z_2^\top]$, we can write the conditional expectation $\mathbb{E}[Z_1|z_2] = C_{12} C_{22}^{-1} z_2$. We now generalize this notion to RKHSs. Following Fukumizu et al. (2004), we define the covariance operator $\mathcal{C}_{X_s X_t}$ which allows us to compute the expectation of the product of function $f(X_s)$ and $g(X_t)$, i.e. $\mathbb{E}_{X_s X_t}[f(X_s) g(X_t)]$, using linear operation in the RKHS. More formally, let $\mathcal{C}_{X_s X_t} : \mathcal{F} \mapsto \mathcal{F}$ such that for all $f, g, h \in \mathcal{F}$,

$$\begin{aligned} \mathbb{E}_{X_s X_t}[f(X_s) g(X_t)] &= \langle f, \mathbb{E}_{X_s X_t}[\phi(X_s) \otimes \phi(X_t)]\, g \rangle_{\mathcal{F}} \\ &= \langle f, \mathcal{C}_{X_s X_t} g \rangle_{\mathcal{F}} \end{aligned} \tag{5}$$

where we use tensor notation $(f \otimes g) h = f \langle g, h \rangle_{\mathcal{F}}$. This can be understood by analogy with the finite dimensional case: if $x, y, z \in \mathbb{R}^d$, then $(x y^\top) z = x (y^\top z)$; furthermore, $(x^\top x')(y^\top y')(z^\top z') = \langle x \otimes y \otimes z,\, x' \otimes y' \otimes z' \rangle_{\mathbb{R}^{d^3}}$ given $x, y, z, x', y', z' \in \mathbb{R}^d$. Based on covariance operators, Song et al. define a conditional embedding operator which allow us to compute conditional expectations $\mathbb{E}_{X_t|x_s}[f(X_t)]$ as linear operations in the RKHS. Let $\mathcal{U}_{X_t|X_s} := \mathcal{C}_{X_t X_s} \mathcal{C}_{X_s X_s}^{-1}$ such that for all $f \in \mathcal{F}$,

$$\begin{aligned} \mathbb{E}_{X_t|x_s}[f(X_t)] &= \langle f, \mathbb{E}_{X_t|x_s}[\phi(X_t)] \rangle_{\mathcal{F}} = \langle f, \mu_{X_t|x_s} \rangle_{\mathcal{F}} \\ &= \langle f, \mathcal{U}_{X_t|X_s} \phi(x_s) \rangle_{\mathcal{F}}. \end{aligned} \tag{6}$$

Although we used the intuition from the Gaussian case in understanding this formula, it is important to note that the conditional embedding operator allows us to compute the conditional expectation of *any* $f \in \mathcal{F}$, regardless of the distribution of the random variable in feature space (aside from the condition that $h(x_s) := \mathbb{E}_{X_t|x_s}[f(X_t)]$ is in the RKHS on x_s, as noted by Song et al.). In particular, we do not need to assume the random variables have a Gaussian distribution in feature space (the definition of feature space Gaussian BP remains a challenging open problem: see Appendix, Section 7).

We can thus express the message update as a linear operation in the feature space,

$$m_{ts}(x_s) = \langle m_{ut}, \mathcal{U}_{X_t|X_s} \phi(x_s) \rangle_{\mathcal{F}}.$$

For multiple incoming messages, the message updates follow the same reasoning as in the single message case, albeit with some additional notational complexity (see also Song et al., 2010). We begin by defining a tensor product reproducing kernel Hilbert space $\mathcal{H} := \otimes^{d_t - 1} \mathcal{F}$, under which the product of incoming messages can be written as a single inner product. For a node t with degree $d_t = |\Gamma_t|$, the product of incoming messages m_{ut} from all neighbors except s becomes an inner product in $\mathcal{H}$,

$$\begin{aligned} \prod_{u \backslash s} m_{ut}(X_t) &= \prod_{u \backslash s} \langle m_{ut}, \phi(X_t) \rangle_{\mathcal{F}} \\ &= \left\langle \bigotimes_{u \backslash s} m_{ut}, \xi(X_t) \right\rangle_{\mathcal{H}}, \end{aligned} \tag{7}$$

where $\xi(X_t) := \bigotimes_{u \backslash s} \phi(X_t)$. The message update (4) becomes

$$m_{ts}(x_s) = \left\langle \bigotimes_{u \backslash s} m_{ut}, \mathbb{E}_{X_t|x_s}[\xi(X_t)] \right\rangle_{\mathcal{H}}. \tag{8}$$

By analogy with (6), we can define the conditional embedding operator for the tensor product of features, such that $\mathcal{U}_{X_t^{\otimes}|X_s} : \mathcal{F} \to \mathcal{F}^{\otimes}$ satisfies

$$\mu_{X_t^{\otimes}|x_s} := \mathbb{E}_{X_t|x_s}[\xi(X_t)] = \mathcal{U}_{X_t^{\otimes}|x_s} \phi(x_s). \tag{9}$$

As in the single variable case, $\mathcal{U}_{X_t^{\otimes}|x_s}$ is defined in terms of a covariance operator $\mathcal{C}_{X_t^{\otimes} X_s} := \mathbb{E}_{X_t X_s}[\xi(X_t) \otimes \phi(X_s)]$ in the tensor space, and the operator $\mathcal{C}_{X_s X_s}$. The operator $\mathcal{U}_{X_t^{\otimes}|X_s}$ takes the feature map $\phi(x_s)$ of the point on which we condition, and outputs the conditional expectation of the tensor product feature $\xi(X_t)$. Consequently, we can express the message update as a linear operation, but in a tensor product feature space,

$$m_{ts}(x_s) = \left\langle \bigotimes_{u \backslash s} m_{ut}, \mathcal{U}_{X_t^{\otimes}|X_s} \phi(x_s) \right\rangle_{\mathcal{H}}. \tag{10}$$

The belief at a specific node s can be computed as $\mathbb{B}(X_s) = \mathbb{P}^\star(X_s) \prod_{u \in \Gamma_s} m_{us}(X_s)$ where the true marginal $\mathbb{P}^\star(X_r)$ can be estimated using Parzen windows. If this is undesirable (for instance, on domains where density estimation cannot be performed), the belief can instead be expressed as a conditional embedding operator (Song et al., 2010).

4.3 Learning Kernel Graphical Models

Given a training sample of m pairs $\{(x_t^i, x_s^i)\}_{i=1}^m$ drawn *i.i.d.* from $\mathbb{P}^\star(X_t, X_s)$, we can represents messages and their updates based purely on these training examples. We first define feature matrices $\Phi = (\phi(x_t^1), \ldots, \phi(x_t^m))$, $\Upsilon = (\phi(x_s^1), \ldots, \phi(x_s^m))$ and $\Phi^\otimes = (\xi(x_t^1), \ldots, \xi(x_t^m))$, and corresponding kernel matrices $K = \Phi^\top \Phi$ and $L = \Upsilon^\top \Upsilon$. The assumption that messages are RKHS functions means that messages can be represented as linear combinations of the training features Φ, i.e., $\widehat{m}_{ut} = \Phi \beta_{ut}$, where $\beta_{ut} \in \mathbb{R}^m$. On this basis, Song et al. (2009) propose a direct regularized estimate of the conditional embedding operators from the data. This approach avoids explicit conditional density estimation, and directly provides the tools needed for computing the RKHS message updates in (10). Following this approach, we first estimate the covariance operators $\widehat{\mathcal{C}}_{X_t X_s} = \frac{1}{m}\Phi\Upsilon^\top$, $\widehat{\mathcal{C}}_{X_t^\otimes X_s} = \frac{1}{m}\Phi^\otimes\Upsilon^\top$ and $\widehat{\mathcal{C}}_{X_s X_s} = \frac{1}{m}\Upsilon\Upsilon^\top$, and obtain an empirical estimate of the conditional embedding operator,

$$\widehat{\mathcal{U}}_{X_t^\otimes | X_s} = \Phi^\otimes (L^\top + \lambda m I)^{-1} \Upsilon^\top, \tag{11}$$

where λ is a regularization parameter. Note that we need not compute the feature space covariance operators explicitly: as we will see, all steps in kernel BP are carried out via operations on kernel matrices.

We now apply the empirical conditional embedding operator to obtain a finite sample message update for (10). Since the incoming messages $\widehat{m}_{ut}$ can be expressed as $\widehat{m}_{ut} = \Phi\beta_{ut}$, the outgoing message $\widehat{m}_{ts}$ at x_s is

$$\left\langle \bigotimes_{u\backslash s} \Phi\beta_{ut},\ \ \Phi^\otimes (L + \lambda m I)^{-1} \Upsilon^\top \phi(x_s) \right\rangle_{\mathcal{H}}$$
$$= \left(\bigodot_{u\backslash s} K\beta_{ut} \right)^\top (L + \lambda m I)^{-1} \Upsilon^\top \phi(x_s) \tag{12}$$

where $\odot$ is the elementwise vector product. If we define $\beta_{ts} = (L + \lambda m I)^{-1} (\bigodot_{u\backslash s} K\beta_{ut})$, then the outgoing message can be expressed as $\widehat{m}_{ts} = \Upsilon\beta_{ts}$. In other words, given incoming messages expressed as linear combinations of feature mapped training samples from X_t, the outgoing message will likewise be a weighted linear combination of feature mapped training samples from X_s. Importantly, only m mapped points will be used to express the outgoing message, regardless of the number of incoming messages or the number of points used to express each incoming message. Thus the complexity of message representation does not increase with BP iterations or degree of a node.

Although we have identified the model parameters with specific edges (s, t), our approach extends straightforwardly to a templatized model, where parameters are shared across multiple edges (this setting is often natural in image processing, for instance). Empirical estimates of the parameters are computed on the pooled observations.

The computational complexity of the finite sample BP update in (12) is polynomial in term of the number of training samples. Assuming a preprocessing step of cost $O(m^3)$ to compute the matrix inverses, the update for a *single* message costs $O(m^2 d_{\max})$ where $d_{\max}$ is the maximum degree of a node in the MRF. While this is reasonable in comparison with competing nonparametric approaches (see Section 6 and the experiments), and works well for smaller graphs and trees, a polynomial time update can be costly for very large m, and for graphical models with loops (where many iterations of the message updates are needed). In Section 5, we develop a message approximation strategy which reduces this cost substantially.

5 Constant Time Message Updates

In this section, we formulate a more computationally efficient alternative to the full rank update in (12). Our goal is to limit the computational cost of each update to $O(\ell^2 d_{\max})$ where $\ell \ll m$. We will require a one-off preprocessing step which is linear in m. This efficient message passing procedure makes kernel BP practical even for very large graphical models and/or training set sizes.

5.1 Approximating Feature Matrices

The key idea of the preprocessing step is to approximate messages in the RKHS with a few informative basis functions, and to estimate these basis functions in a data dependent way. This is achieved by approximating the feature matrix Φ as a weighted combination of a subset of its columns. That is, $\Phi \approx \Phi_{\mathcal{I}} W_t$, where $\mathcal{I} := \{i_1, \ldots, i_\ell\} \subseteq \{1, \ldots, m\}$, W_t has dimension $\ell \times m$, and $\Phi_{\mathcal{I}} = (\phi(x_t^{i_1}), \ldots, \phi(x_t^{i_\ell}))$ is a submatrix formed by taking the columns of Φ corresponding to the indices in $\mathcal{I}$. Likewise, we approximate $\Upsilon \approx \Upsilon_{\mathcal{J}} W_s$, assuming $|\mathcal{J}| = \ell$ for simplicity. We thus can approximate the kernel matrices as low rank factorizations, i.e., $K \approx W_t^\top K_{\mathcal{I}\mathcal{I}} W_t$ and $L = W_s^\top L_{\mathcal{J}\mathcal{J}} W_s$, where $K_{\mathcal{I}\mathcal{I}} := \Phi_{\mathcal{I}}^\top \Phi_{\mathcal{I}}$ and $L_{\mathcal{J}\mathcal{J}} = \Upsilon_{\mathcal{J}}^\top \Upsilon_{\mathcal{J}}$.

A common way to obtain the approximation $\Phi \approx \Phi_{\mathcal{I}} W_t$ is via a Gram-Schmidt orthogonalization procedure in feature space, where an incomplete set of ℓ orthonormal basis vectors $Q := (q_t^1, \ldots, q_t^\ell)$ is constructed from a greedily selected subset of the data, chosen to minimize the reconstruction error (Shawe-Taylor & Cristianini, 2004, p.126). The original feature matrix can be approximately expressed using this basis subset as $\Phi \approx QR$ where $R \in \mathbb{R}^{\ell \times m}$ are the coefficients under the new basis. There is a simple relation between Q and the chosen data points $\Phi_{\mathcal{I}}$, i.e., $Q = \Phi_{\mathcal{I}} R_{\mathcal{I}}^{-1}$, where $R_{\mathcal{I}}$ is the submatrix formed by taking the columns of R corresponding to $\mathcal{I}$. It follows that $W_t = R_{\mathcal{I}}^{-1} R$. All operations involved in Gram-Schmidt orthogonalization are linear in feature space, and the entries of R can be computed based solely on kernel values $k(x_t, x_t')$. The cost of performing this orthogonalization is $O(m\ell^2)$. The number ℓ of chosen basis vectors is inversely related to the approximation error or residual

$\epsilon = \max_i \|\phi(x_t^i) - \Phi_{\mathcal{I}} W_t^i\|_{\mathcal{F}}$ (W_t^i denotes column i of W_t). In many cases of interest (for instance, when a Gaussian RBF kernel is used), a small $\ell \ll m$ will be sufficient to obtain a small residual ϵ for the feature matrix, due to the fast decay of the eigenspectrum in feature space (Bach & Jordan, 2002, Appendix C).

5.2 Approximating Tensor Features

The approximations $\Phi \approx \Phi_{\mathcal{I}} W_t$ and $\Upsilon \approx \Upsilon_{J} W_s$, and associated low rank kernel approximations are insufficient for a constant time approximate algorithm, however. In fact, directly applying these results will only lead to a linear time approximate algorithm: this can be seen by replacing the kernel matrices in (12) by their low rank approximations.

To achieve a constant approximate update, our strategy is to go a step further: in addition to approximating the kernel matrices, we further approximate the tensor product feature matrix in equation (11), $\Phi^{\otimes} \approx \Phi^{\otimes}_{\mathcal{I}'} W_t^{\otimes}$ ($W_t^{\otimes} \in \mathbb{R}^{\ell' \times m}$). Crucially, the individual kernel matrix approximations neglect to account for the subsequent tensor product of these messages. By contrast, our proposed approach also approximates the tensor product directly. The computational advantage of a direct tensor approximation approach is substantial in practice (a comparison between exact kernel BP and its constant and linear time approximations can be found in Section 3 of the Appendix) .

The decomposition procedure for tensor $\Phi^{\otimes} \approx \Phi^{\otimes}_{\mathcal{I}'} W_t^{\otimes}$ follows exactly the same steps as in the original feature space, but using the kernel $k^{d_t-1}(x_t, x_t')$, and yielding an incomplete orthonormal basis in the tensor product space. In general the index sets $\mathcal{I}' \neq \mathcal{I}$, meaning they select different training points to construct the basis functions. Furthermore, the size ℓ' of $\mathcal{I}'$ is not equal to the size ℓ of $\mathcal{I}$ for a given approximation error ϵ. Typically $\ell' > \ell$, since the tensor product space has a slower decaying spectrum, however we will write ℓ in place of ℓ' to simplify notation.

5.3 Constant Time Approximate Updates

We now compute the message updates based on the various low rank approximations. The incoming messages and the conditional embedding operators become

$$\bigotimes_{u\backslash s} m_{ut} \approx \bigotimes_{u\backslash s} \Phi_{\mathcal{I}} W_t \beta_{ut}, \qquad (13)$$

$$\widetilde{\mathcal{U}}_{X_t^{\otimes}|X_s} \phi(x_s) \approx \Phi^{\otimes}_{\mathcal{I}'} W_{ts} \Upsilon_{\mathcal{J}}^{\top} \phi(x_s), \qquad (14)$$

where $W_{ts} := W_t^{\otimes}(W_s^{\top} L_{\mathcal{J}\mathcal{J}} W_s + \lambda m I)^{-1} W_s^{\top}$. If we reparametrize the messages m_{ut} as $m_{ut} = \Phi_{\mathcal{I}} \alpha_{ut}$ where $\alpha_{ut} := W_t \beta_{ut}$, we can express the message updates for $m_{ts}(x_s)$ as

$$m_{ts}(x_s) \approx \left(\bigodot_{u\backslash s} K_{\mathcal{I}'\mathcal{I}} \alpha_{ut} \right)^{\top} W_{st} \Upsilon_{\mathcal{J}}^{\top} \phi(x_s), \quad (15)$$

where $K_{\mathcal{I}'\mathcal{I}}$ denotes the submatrix of K with rows indexed $\mathcal{I}'$ and columns indexed $\mathcal{I}$. The outgoing message m_{ts} can also be reparametrized as a vector $\alpha_{ts} = W_{st}^{\top} \left(\bigodot_{u\backslash s} K_{\mathcal{I}'\mathcal{I}} \alpha_{ut} \right)$. In short, the message from t to s is a weighted linear combination of the ℓ vectors in $\Upsilon_{\mathcal{J}}$.

We note that W_{ts} can be computed efficiently prior to the message update step, since $W_t^{\otimes}(W_s^{\top} L_{\mathcal{J}\mathcal{J}} W_s + \lambda m I)^{-1} W_s^{\top} = W_t^{\otimes} W_s^{\top}(W_s W_s^{\top} + \lambda m L_{\mathcal{J}\mathcal{J}}^{-1})^{-1} L_{\mathcal{J}\mathcal{J}}^{-1}$ via the Woodbury expansion of the matrix inverse. In the latter form, matrix products $W_s W_s^{\top}$ and $W_t^{\otimes} W_s^{\top}$ cost $O(\ell^2 m)$; the remaining operations (size ℓ matrix products and inversions) are significantly less costly at $O(\ell^3)$. This initialization cost of $O(\ell^3 + \ell^2 m)$ need only be borne once.

The cost of updating a single message m_{ts} in (15) becomes $O(\ell^2 d_{\max})$ where $d_{\max}$ is the maximum degree of a node. This also means that our approximate message update scheme will be independent of the number of training examples. With these approximate messages, the evaluation of the belief $\widehat{\mathbb{B}}(x_r)$ of a node r at x_r can be carried out in time $O(\ell d_{\max})$.

Finally, approximating the tensor features introduces additional error into each message update. This is caused by the difference between the full rank conditional embedding operator $\widehat{\mathcal{U}}_{X_t^{\otimes}|X_s}$ in (11) and its low rank counterpart $\widetilde{\mathcal{U}}_{X_t^{\otimes}|X_s}$ in (14). Under suitable conditions, this difference is bounded by the feature approximation error ϵ, i.e., $\|\widehat{\mathcal{U}}_{X_t^{\otimes}|X_s} - \widetilde{\mathcal{U}}_{X_t^{\otimes}|X_s}\|_{HS} \leq 2\epsilon(\lambda^{-1} + \lambda^{-3/2})$ (see Section 8 of the Appendix for details).

6 Gaussian Mixture And Particle BP

We briefly review two state-of-the-art approaches to nonparametric belief propagation: Gaussian Mixture BP (Sudderth et al., 2003) and Particle BP (Ihler & McAllester, 2009). By contrast with our approach, we must provide these algorithms in advance with an estimate of the conditional density $\mathbb{P}^{\star}(X_t|X_s)$, to compute the conditional expectation in (4). For Gaussian Mixture BP, this conditional density must take the form of a mixture of Gaussians. We describe how we learn the conditional density from data, and then show how the two algorithms use it for inference.

A direct approach to estimating the conditional density $\mathbb{P}^{\star}(X_t|X_s)$ would be to take the ratio of the joint empirical density to the marginal empirical density. The ratio of mixtures of Gaussians is not itself a mixture of Gaussians, however, so this approach is not suitable for Gaussian Mixture BP (indeed, message updates using this ratio of mixtures would be non-trivial, and we are not aware of any such inference approach). We propose instead to learn $\mathbb{P}^{\star}(X_t|X_s)$ directly from training data following Sugiyama et al. (2010), who provide an estimate in the form of a mixture of Gaussians (see Section 1 of the Appendix for details). We emphasize that the updates bear no resemblance to our kernel updates in (12), which do not attempt density ratio estimation.

Given the estimated $\widehat{\mathbb{P}}(X_t|X_s)$ as input, each nonparametric inference method takes a different approach. Gaussian mixture BP assumes incoming messages to be a mixture of

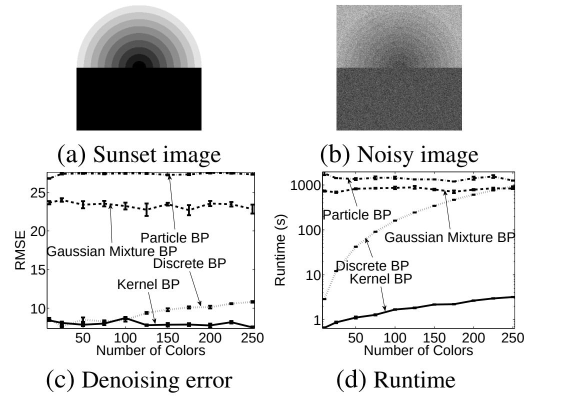


Figure 1: Average denoising error and runtime of kernel BP compared to discrete, Gaussian mixture and particle BP over 10 test images with varying numbers of rings. Runtimes are plotted on a logarithmic scale.

b Gaussians. The product of d_t incoming messages to node t then contains b^{d_t} Gaussians. This exponential blow-up is avoided by replacing the exact update with an approximation. An overview of approximation approaches can be found in Bickson et al. (2011); we used an efficient KD-tree method of Ihler et al. (2003) for performing the approximation step. Particle BP represents the incoming messages using a common set of particles.These particles must be re-drawn via Metropolis-Hastings at each node and BP iteration, which is costly (although in practice, it is sufficient to resample periodically, rather than strictly at every iteration). By contrast, our updates are simply matrix-vector products. See Appendix for further discussion.

7 Experiments

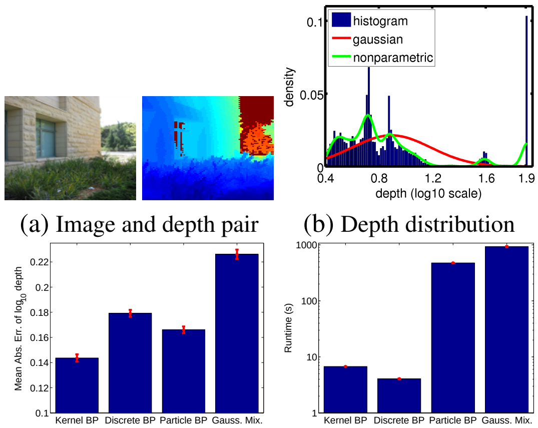
We performed four sets of experiments. The first two were image denoising and depth prediction problems, where we show that kernel BP is superior to discrete, Gaussian mixture and particle BP in both speed and accuracy, using a GraphLab implementation of each (Low et al., 2010). The remaining two experiments were protein structure and paper category prediction problems, where domain-specific kernels were crucial (for the latter see Appendix, Sec. 4).

Image denoising: In our first experiment, the data consisted of grayscale images of size 100×100, resembling a sunset (Figure 1(a)). The number of colors (gray levels) in the images ranged across 10, 25, 50, 75, 100, 125, 150, 175, 200, 225 and 250, with gray levels varying evenly from 0 to 255 from the innermost ring of the sunset to the outermost. As we increased the number of colors, the grayscale transition became increasingly smooth. Our goal was to recover the original images from noisy versions, to which we had added zero mean Gaussian noise with $\sigma = 30$. We compared the denoising performance and runtimes of discrete, Gaussian mixture, particle, and kernel BP.

The topology of our graphical model was a grid of hidden noise-free pixels with noisy observations made at each. The maximum degree of a node was 5 (four neighbours and an observation), and we used a template model where both the edge potentials and the likelihood functions were shared across all variables. We generated a pair of noise-free and noisy images as training data, at each color number. For kernel BP, we learned both the likelihood function and the embedding operators nonparametrically from the data. We used a Gaussian RBF kernel $k(x, x')$, with kernel bandwidth set at the median distance between training points, and residual $\epsilon = 10^{-3}$ as the stopping criterion for the feature approximation (see definition of ϵ in Section 5.1). For discrete, Gaussian mixture, and particle BP, we learned the edge potentials from data, but supplied the *true* likelihood of the observation given the hidden pixel (i.e., a Gaussian with standard deviation 30). This gave competing methods an important *a priori* advantage over kernel BP: in spite of this, kernel BP still outperformed competing approaches in speed and accuracy.

In Figure 1(c) and (d), we report the average denoising performance (RMSE: root mean square error) and runtime over 30 BP iterations, using 10 independently generated noisy test images. The RMSE of kernel BP is significantly lower than Gaussian mixture and particle BP for all numbers of colors. Although the RMSE of discrete BP is about the same as kernel BP when the number of colors is small, its performance becomes worse than kernel BP as the number of colors increases beyond 100 (despite discrete BP receiving the true observation likelihood in advance). In terms of speed, kernel BP has a considerable advantage over the alternatives: the runtime of KBP is barely affected by the number of colors. For discrete BP, the scaling is approximately square in the number of colors. For Gaussian mixture and particle BP, the runtimes are orders of magnitude longer than kernel BP, and are affected by the variability of the resampling algorithm.

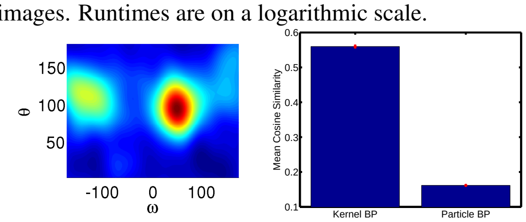
Predicting depth from 2D images: The prediction of 3D depth information from 2D image features is a difficult inference problem, as the depth may be ambiguous: similar features can occur at different depths. This creates a multimodal depth distribution given the image feature. Furthermore, the marginal distribution of the depth can itself be multimodal, which makes the Gaussian approximation a poor choice (see Figure 2(b)). To make a spatially consistent prediction of the depth map, we formulated the problem as an undirected graphical model, where a depth variable $y_i \in \mathbb{R}$ was associated with each patch of an image, and these variables were connected according to a 2D grid topology. Each hidden depth variable was linked to an image feature variable $x_i \in \mathbb{R}^{273}$ for the corresponding patch. This formulation resulted in a graphical model with $9,202 = 107 \times 86$ continuous depth variables, and a maximum node degree of 5. Due to the way the images were taken (upright), we used a templatized model where horizontal edges in a row shared the same potential, vertical edges at the same height shared the same potential, and patches at the same row shared the same likelihood



(a) Image and depth pair (b) Depth distribution

(c) Depth prediction error (d) Runtime

Figure 2: Average depth prediction error and runtime of kernel BP compared to discrete, Gaussian mixture and particle BP over 274 images. Runtimes are on a logarithmic scale.



(a) Angle distribution (b) Prediction accuracy

Figure 3: Average angle prediction accuracy of kernel versus particle BP in the protein folding problem.

function. Both the edge potentials between adjacent depth variables and the likelihood function between image feature and depth were unknown, and were learned from data.

We used a set of 274 images taken on the Stanford campus, including both indoor and outdoor scenes (Saxena et al., 2009). Images were divided into patches of size 107 by 86, with the corresponding depth map for each patch obtained using 3D laser scanners (*e.g.*, Figure 2(a)). Each patch was represented by a 273 dimensional feature vector, which contained both local features (such as color and texture) and relative features (features from adjacent patches). We took the logarithm of the depth map and performed learning and prediction in this space. The entire dataset contained more than 2 million data points ($107 \times 86 \times 274$). We applied a Gaussian RBF kernel on the depth information, with the bandwidth parameter set to the median distance between training depths, and an approximation residual of $\epsilon = 10^{-3}$. We used a linear kernel for the image features.

Our results were obtained by leave-one-out cross validation. For each test image, we ran discrete, Gaussian mixture, particle, and kernel BP for 10 BP iterations. The average prediction error (MAE: mean absolute error) and runtime are shown in Figures 2(c) and (d). Kernel BP produces the lowest error (MAE=0.145) by a significant margin, while having a similar runtime to discrete BP. Gaussian mixture and particle BP achieve better MAE than discrete BP, but their runtimes are two order of magnitude slower. We note that the error of kernel BP is slightly better than the results of pointwise MRF reported in Saxena et al. (2009).

Protein structure prediction: Our final experiment investigates the protein folding problem. The folded configuration of a protein of length n is roughly determined by a sequence of angle pairs $\{(\theta_i, \omega_i)\}_{i=1}^n$, each specific to an amino acid position. The goal is to predict the sequence of angle pairs given only the amino acid sequence. The two angles (θ_i, ω_i) have ranges $[0, 180]$ and $(-180, 180]$ respectively, such that they correspond to points on the unit sphere S^2. Kernels yield an immediate solution to inference on these data: Wendland (2005, Theorem 17.10) provides a sufficient condition for a function on S^2 to be positive definite, satisfied by $k(x, x') := \exp(\sigma \langle x, x' \rangle)$, where $\langle x, x' \rangle$ is the standard inner product between Euclidean coordinates. Given the data are continuous, multimodal, and on a non-Euclidean domain (Figure 3(a)), it is not obvious how Gaussian mixture or discrete BP might be applied. We therefore focus on comparing kernel and particle BP.

We obtained a collection of $1,400$ proteins with length larger than 100 from PDB. We first ran PSI-BLAST to generate the sequence profile (a 20 dimensional feature for each amino acid position), and then used this profile as features for predicting the folding structure (Jones, 1999). The graphical model was a chain of connected angle pairs, where each angle pair was associated with a 20 dimensional feature. We used a linear kernel on the sequence features. For the kernel between angles, the bandwidth parameter was set at the median inner product between training points, and we used the approximation residual $\epsilon = 10^{-3}$. For particle BP, we learned the nonparametric potentials using $\exp(\sigma \langle x, x' \rangle)$ as the basis functions.

In Figure 3(b), we report the average prediction accuracy (Mean Cosine Similarity between the true coordinate x and the predicted x', i.e., $\langle x, x' \rangle$) over a 10-fold cross-validation process. In this case, kernel BP achieves a significantly better result than particle BP while running much faster (runtimes not shown due to space constraints).

8 Conclusions and Further Work

We have introduced *kernel belief propagation*, where the messages are functions in an RKHS. Kernel BP performs learning and inference on challenging graphical models with structured and continuous random variables, and is more accurate and much faster than earlier nonparametric BP algorithms. A possible extension to this work would be to kernelize tree-reweighted belief propagation (Wainwright et al., 2003). The convergence of kernel BP is a further challenging topic for future work (Ihler et al., 2005).

Acknowledgements: We thank Alex Ihler for the Gaussian mixture BP codes and helpful discussions. LS is supported by a Stephenie and Ray Lane Fellowship. This research was also supported by ARO MURI W911NF0710287, ARO MURI W911NF0810242, NSF Mundo IIS-0803333, NSF Nets-NBD CNS-0721591 and ONR MURI N000140710747.

References

Aronszajn, N. (1950). Theory of reproducing kernels. *Trans. Amer. Math. Soc.*, *68*, 337–404.

Bach, F. R., & Jordan, M. I. (2002). Kernel independent component analysis. *J. Mach. Learn. Res.*, *3*, 1–48.

Bickson, D., Baron, D., Ihler, A., Avissar, H., & Dolev, D. (2011). Fault identification via non-parametric belief propagation. *IEEE Transactions on Signal Processing*. ISSN 1053-587X.

Fukumizu, K., Bach, F. R., & Jordan, M. I. (2004). Dimensionality reduction for supervised learning with reproducing kernel Hilbert spaces. *J. Mach. Learn. Res.*, *5*, 73–99.

Fukumizu, K., Sriperumbudur, B., Gretton, A., & Schoelkopf, B. (2009). Characteristic kernels on groups and semigroups. In *Advances in Neural Information Processing Systems 21*, 473–480. Red Hook, NY: Curran Associates Inc.

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

Ihler, A. T., Fisher III, J. W., & Willsky, A. S. (2005). Loopy belief propagation: Convergence and effects of message errors. *J. Mach. Learn. Res.*, *6*, 905–936.

Ihler, E. T., Sudderth, E. B., Freeman, W. T., & Willsky, A. S. (2003). Efficient multiscale sampling from products of gaussian mixtures. In *In NIPS 17*.

Jones, D. T. (1999). Protein secondary structure prediction based on position-specific scoring matrices. *J. Mol. Biol.*, *292*, 195–202.

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

Low, Y., Gonzalez, J., Kyrola, A., Bickson, D., Guestrin, C., & Hellerstein, J. M. (2010). GraphLab: A new parallel framework for machine learning. In *Conference on Uncertainty in Artificial Intelligence*.

McEliece, R., MacKay, D., & Cheng, J. (1998). Turbo decoding as an instance of Pearl's belief propagation algorithm. *J-SAC*.

Murphy, K. P., Weiss, Y., & Jordan, M. I. (1999). Loopy belief propagation for approximate inference: An empirical study. In *UAI*, 467–475.

Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufman.

Saxena, A., Sun, M., & Ng, A. Y. (2009). Make3d: Learning 3d scene structure from a single still image. *IEEE Trans. Pattern Anal. Mach. Intell.*, *31*(5), 824–840. ISSN 0162-8828. doi: http://dx.doi.org/10.1109/TPAMI.2008.132.

Schölkopf, B., & Smola, A. (2002). *Learning with Kernels*. Cambridge, MA: MIT Press.

Schölkopf, B., Tsuda, K., & Vert, J.-P. (2004). *Kernel Methods in Computational Biology*. Cambridge, MA: MIT Press.

Shawe-Taylor, J., & Cristianini, N. (2004). *Kernel Methods for Pattern Analysis*. Cambridge, UK: Cambridge University Press.

Song, L., Gretton, A., & Guestrin, C. (2010). Nonparametric tree graphical models. In *13th Workshop on Artificial Intelligence and Statistics*, vol. 9 of *JMLR workshop and conference proceedings*, 765–772.

Song, L., Huang, J., Smola, A., & Fukumizu, K. (2009). Hilbert space embeddings of conditional distributions. In *Proc. Intl. Conf. Machine Learning*.

Steinwart, I. (2001). On the influence of the kernel on the consistency of support vector machines. *J. Mach. Learn. Res.*, *2*, 67–93.

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. 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.

Weiss, Y., & Freeman, W. T. (2001). Correctness of belief propagation in Gaussian graphical models of arbitrary topology. *Neural Computation*, *13*, 2173–2200.

Wendland, H. (2005). *Scattered Data Approximation*. Cambridge, UK: Cambridge University Press.

Yanover, C., & Weiss, Y. (2002). Approximate inference and protein-folding. In *NIPS*, 1457–1464. MIT Press.

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.