Kernel-Based Just-In-Time Learning for Passing Expectation Propagation Messages

Abstract

We propose to learn a kernel-based message operator which takes as input all expectation propagation (EP) incoming messages to a factor node and produces an outgoing message. In ordinary EP, computing an outgoing message involves estimating a multivariate integral which may not have an analytic expression. Learning such an operator allows one to bypass the expensive computation of the integral during inference by directly mapping all incoming messages into an outgoing message. The operator can be learned from training data (examples of input and output messages) which allows automated inference to be made on any kind of factor that can be sampled.

This is the old abstract. Add online learning ?

AE: 1. State the problem. 2. Say why it's interesting. 3. Say what your solution achieves. 4. Say what follows from your solution. Message-passing is useful. Message-passing in general is slow. Our approach speeds it up. This allows us to use it on more complicated / larger problems.

1 Introduction

- In general: learning to infer is becoming a THING (variational community, inference networks for NN; probabilistic programming literature [kernel EP, stochastic inverses, Wingate stuff, etc.; can have additional properties beyond being accurate like better convergence properties (Eslami); all super-awesome and we need more research).
- Inference is hard (several problems: global [need to manipulate many interdependent variables]; unclear how to represent distributions [which, in the continuous case, can be infinite-dimensional objects], update rules are typically intractable [need to solve non-trivial integrals].
- Message passing reduces inference to local, deterministic update rules.
- VMP, EP solve the question of representation of distribution by mapping onto simple parametric forms, but the problem remains that we need to map / transform messages.
- remaining fundamental question: how to manipulate distributions or messages (distribution-to-distribution mappings).
- Barthelme, Heess, Eslami propose solutions, the first is asymptotically exact and principled but stochastic and therefore slow / high variance (this is especially bad as it is an inner loop to MP algorithm); Heess learns a deterministic mapping in a batch setup with NNs but does not produces uncertainty estimates and therefore blows up without warning, no factor refinement, need to do a lot of (potentially unnecessary) hard work up front; Eslami combines the two approaches by learning a deterministic mapping on the fly but uses several crude heuristics, computationally relatively expensive online updates and ultimately unprincipled uncertainty estimates.
- Here we present a kernel-based method for just-in-time learning of distribution-to-distribution mappings and apply it to learning message passing computations (KJIT). KJIT is fast, statistically efficient and its uncertainty estimates are principled.
- Our first contribution is to develop an N-independent kernel based method for regressing from distributions to distributions.
- Our second contribution is to apply this to the JIT setting.
- Experiments on a number of models demonstrate state-of-the-art speed and accuracy.

Existing approaches to automated probabilistic inference can be broadly divided into two categories [Heess et al., 2013]: uninformed and informed cases. In the uninformed case, the modeler has full freedom in expressing a probabilistic model without any constraint

on the set of usable factors. This high flexibility comes at a price during inference as less factor-specific information is available to the inference engine. Often MCMC-based sampling techniques are employed by treating the factors as black boxes. In the informed case [Stan Development Team, 2014, Minka et al., 2012], the modeler is required to build a model from constructs whose necessary computations are known to the inference engine. Although efficient during the inference, using an unsupported construct would require manual derivation and implementation of the relevant computations in the inference engine.

In this work, we focus on EP, a commonly used approximate inference method. Following Heess et al. [2013], Eslami et al. [2014], we propose to learn a kernel-based message operator for EP to capture the relationship between incoming messagesj to a factor and outgoing messages. The operator bridges the gap between the uninformed and informed cases by automatically deriving the relevant computations for any custom factor that can be sampled. This hybrid approach gives the modeler as much flexibility as in the uninformed case while offering efficient message updates as in the informed case. This approach supports fast inference as no expensive KL divergence minimization needs to be done during inference as in ordinary EP. In addition, a learned operator for a factor is reusable in other models in which the factor appears. As will be seen, to send an outgoing message with the kernel-based operator, it is sufficient to generate a feature vector for incoming messages and multiply with a pre-learned matrix. Unlike Heess et al. [2013] which considers a neural network, the kernel-based message operator we propose can be easily extended to allow online updates of the operator during inference.

TODO/TO add:

- Mention that online learning improves Nicolas's batch training approach.
- To connect to Ali's work, can mention that online learning of random forests is difficult. Online learning is straightforward and exact for Bayesian linear regression. May ask Balaji more on relevant citations.
- It was argued in Hutter [2009] (Section 11.2) that a random forest may have problems with exploring previously-unseen areas of the input domain.

Our paper is structured as follows: In section 2 we briefly review the notion of message passing for approximate inference and, in particular, the expectation propagation (EP) algorithm which is the basis of our work. We then describe how approximate EP can be performed using Monte Carlo estimates of the message, in the case when message updates cannot be computed analytically. In section **??** we describe how this approximation technique can be improved with an online learning framework that effectively remembers previously computed approximate updates and we propose a novel kernel-based regression technique for this approach.

2 Background

As is common in the message passing community we assume that distributions (or densities) over a set of variables $\mathbf{x} = (x_1, \dots x_D)$ of interest can be represented as factor graphs, i.e.

$$p(\mathbf{x}) = \frac{1}{Z} \prod_{j=1}^{J} \psi_j(\mathbf{x}_{\mathrm{ne}(\psi_j)}). \tag{1}$$

Here ψ_j, the "factors", are non-negative functions which are defined over subsets of the variables $\mathbf{x}_{\mathrm{ne}(\psi_j)}$, i.e. the neighbors of the factor node ψ_j in the factor graph (we use $\mathrm{ne}(\psi_j)$ to denote the set indices of the variables of factor ψ_j). Z is the normalization constant. In this work we focus on *directed* factors $\psi(\mathbf{x}_{\mathrm{out}}|\mathbf{x}_{\mathrm{in}})$ which specify a conditional distribution over variables $\mathbf{x}_{\mathrm{out}}$ given $\mathbf{x}_{\mathrm{in}}$ (and $\mathbf{x}_{\mathrm{ne}(\psi)} = (\mathbf{x}_{\mathrm{out}}, \mathbf{x}_{\mathrm{in}})$).

2.1 Message passing EP

The belief propagation - or sum-product algorithm - computes marginal distributions over subsets of variables by iteratively passing messages between variables and factors, ensuring consistency of the obtained marginals at convergence. Specifically, the messages sent from a factor ψ to variable x_i (where $i \in \mathrm{ne}(\psi)$) are computed as

$$m_{\psi \to i}(x_i) = \int \psi(\mathbf{x}_{\mathrm{ne}(\psi)}) \prod_{i' \in \mathrm{ne}(\psi) \setminus i} m_{i' \to \psi}(x_{i'}) \mathrm{d}\mathbf{x}_{\mathrm{ne}(\psi) \setminus i}, \tag{2}$$

where $m_{i' \to \psi}(\cdot)$ are the messages sent to factor ψ from its neighboring variables $x_{i'}$ other than x_i.

EP introduces an approximation in the case when the message $m_{\psi \to i}(\cdot)$ does not have a simple parametric form, by projecting the exact marginal $m_{i \to \psi}(x_i) m_{\psi \to i}(x_i)$ onto a member of some class of known parametric distributions

$$m_{\psi \to i}(x_i) = \frac{\mathrm{proj}\left[m_{i \to \psi}(x_i) \int \psi(\mathbf{x}_{\mathrm{ne}(\psi)}) \prod_{i' \in \mathrm{ne}(\psi) \setminus i} m_{i' \to \psi}(x_{i'}) \mathrm{d}\mathbf{x} \right]}{m_{i \to \psi}(x_i)} \tag{3}$$

where $\mathrm{proj}[p] = \mathrm{argmin}_{q \in \mathcal{Q}} \mathrm{KL}[p||q]$, and $\mathcal{Q}$ is typically in the exponential family, e.g. the set of Gaussian or Beta distributions.

2.2 Monte-Carlo message approximation

While EP introduces an approximation that allows message passing to proceed when the messages are not easily representable in closed form, computing *the numerator* of (3) is often a challenging problem in itself as it requires evaluating a high-dimensional integral as well as minimization of the Kullback-Leibler divergence to some non-standard distribution. For non-trivial factors this either requires a hand-crafted approximation (references!) or, for instance, the use of expensive numerical integration techniques, significantly limiting the ease with which EP can be applied and its scope in practice.

However, while deterministic approximations to (3) may be hard to come by, Barthelmé and Chopin [2011], Heess et al. [2013], Eslami et al. [2014] exploit an alternative, stochastic approximation. In general, the projection $\text{proj}\left[\cdot\right]$ of p onto some member of the exponential family corresponds to matching the relevant moments of p, e.g. its mean and variance, if q is required to be a Gaussian distribution. Thus, projecting p onto a general member of the exponential family, $q(x|\eta) = h(\theta)\exp\left(\eta^\top u(x) - A(\eta)\right)$ with a vector of sufficient statistics u and natural parameters η requires us to compute the expectation of $u(\cdot)$ under the numerator of 3).

A sample based approximation of this expectation can, in general, obtained via MCMC. One especially simple approach is importance sampling:

$$\mathbb{E}_{\mathbf{x}_{\text{ne}(\psi)}\sim b}\left[u(x)\right] \approx \frac{1}{N}\sum_{l=1}^{N} w(\mathbf{x}^l_{\text{ne}(\psi)})u(x^l_i), \qquad \mathbf{x}^s_{\text{ne}(\psi)} \sim \tilde{b} \tag{4}$$

where, on the left hand side, $b(\mathbf{x}_{\text{ne}(\psi)}) = \psi(\mathbf{x}_{\text{ne}(\psi)})\prod_{i\in\text{ne}(\psi)} m_{i\to\psi}(x_i)$. On the right hand side we draw samples $\mathbf{x}^l_{\text{ne}(\psi)}$ from some proposal distribution $\tilde{b}$ which we choose to be $\tilde{b}(\mathbf{x}_{\text{ne}(\psi)}) = r(\mathbf{x}_{\text{in}})\psi(\mathbf{x}_{\text{out}}|\mathbf{x}_{\text{in}})$ for some r with appropriate support, and compute importance weights $w(\mathbf{x}_{\text{ne}(\psi)}) = \frac{\prod_{i\in\text{ne}(\psi)} m_{i\to\psi}(x_i)}{r(\mathbf{x}_{\text{in}})}$.

The thus estimated expected sufficient statistics provide us with an estimate of the parameters η of the result q of the projection $\text{proj}\left[p\right]$, from which the message is readily computed.

2.3 Just-in-time learning of messages

Message approximations as in the previous section could be used directly when running the EP algorithm as in Barthelmé and Chopin [2011], but this approach can suffer from the unreliability of the importance sampling estimates when the number of samples N is small. On the other hand, for large N the computational cost of running EP with approximate messages can be very high, as the message approximations have to be re-computed in each iteration of EP. To obtain low-variance message approximations at lower computational cost Heess et al. [2013], Eslami et al. [2014] both amortize previously computed approximate messages by training a function approximator to directly map a tuple of incoming variable-to-factor messages $(m_{i'\to\psi})_{i'\in\text{ne}(\psi)}$ to an approximate factor to variable message $m_{\psi\to i}$, i.e. they learn a mapping

$$\hat{m}^\theta_{\psi\to i} : (m_{i'\to\psi})_{i'\in\text{ne}(\psi)} \mapsto m_{\psi\to i}, \tag{5}$$

where θ are the parameters of the function approximator. Note that for exponential family distribution this effectively reduces to a multi-variate regression problem as each message can be represented by a finite-dimensional parameter vector, and the message update (3) can thus be understood as a vector valued function.

While Heess et al. [2013] use neural networks and a very large, fixed training set to learn their approximate message operator prior to running EP with the intractable factor, Eslami et al. [2014] investigate an online learning approach in which the approximate message operator is updated on the fly, during inference, depending on whether the function approximator can be expected to produce a reliable message approximation. To this end, they endow their function approximator with an uncertainty estimate which

$$\hat{u}^\theta_{\psi\to i} : (m_{i'\to\psi})_{i'\in\text{ne}(\psi)} \mapsto u, \tag{6}$$

where u indicates the expected unreliability of the predicted, approximate message $m_{\psi\to i}$ returned by $\hat{m}^\theta_{\psi\to i}$. If $u = \hat{u}^\theta_{\psi\to i}\left((m_{i'\to\psi})_{i'\in\text{ne}(\psi)}\right)$ exceeds a predefined threshold, the required message is approximated via importance sampling (cf. eq. 4) and $\hat{m}^\theta_{\psi\to i}$ is re-trained on this new datapoint (leading to a new set of parameters θ' with $\hat{u}^{\theta'}_{\psi\to i}\left((m_{i'\to\psi})_{i'\in\text{ne}(\psi)}\right)) < \hat{u}^\theta_{\psi\to i}\left((m_{i'\to\psi})_{i'\in\text{ne}(\psi)}\right)$.

The just-in-time approach of Eslami et al. [2014] critically relies on two properties of the class of function approximators used to represent $\hat{m}^\theta_{\psi\to i}$: they need to provide (un)reliability estimates $\hat{u}^\theta_{\psi\to i}$ over their domain, and they need to allow for online learning. Eslami et al. [2014] use decision trees. [These do fulfill both criteria, however, only when resorting to unprincipled heuristics. In the next section we present a framework based on kernel mean embeddings which possesses the desired properties and achieves this in a principled manner.]

Expectation propagation [Minka, 2001, Bishop, 2006] is a message passing algorithm that extends BP in situations where As is common for message passing al-

gorithms we assume that the probabilistic models of interest are represent as factor graphs

Expectation propagation [Minka, 2001, Bishop, 2006] (EP) is a commonly used approximate inference method for inferring the posterior distribution of latent variables given observations. In a typical directed graphical model, the joint distribution of the data $X = \{X_1, \ldots, X_n\}$ and latent variables $\theta = \{\theta_1, \ldots, \theta_t\}$ takes the form of a product of factors, $p(X, \theta) = \prod_{i=1}^{m} f_i(X|\theta)$ where each factor f_i may depend on only a subset of X and θ. With X observed, EP approximates the posterior with $q(\theta) \propto \prod_{i=1}^{m} m_{f_i \to \theta}(\theta)$ where $m_{f_i \to \theta}$ is an approximate factor corresponding to f_i with the constraint that it has a chosen parametric form (e.g., Gaussian) in the exponential family (ExpFam). EP takes into account the fact that the final quantity of interest is the posterior $q(\theta)$ which is given by the product of all approximate factors. In finding the i^{th} approximate factor $m_{f_i \to \theta}$, EP uses other approximate factors $m_{\theta \to f_i}(\theta) := \prod_{j \neq i} m_{f_j \to \theta}(\theta)$ as a context to determine the plausible range of θ. EP iteratively refines $m_{f_i \to \theta}$ for each i with $m_{f_i \to \theta}(\theta) = \frac{\text{proj}\left[\int dX\, f(X|\theta) m_{X \to f_i}(X) m_{\theta \to f_i}(\theta)\right]}{m_{\theta \to f_i}(\theta)}$ where $\text{proj}[r] = \arg\min_{q \in \text{ExpFam}} \text{KL}[r \,\|\, q]$ and $m_{X \to f_i}(X) := \delta(X - X_0)$ if X is observed to be X_0. In the EP literature, $m_{\theta \to f_i}$ is known as a cavity distribution.

No need to mention this special-case msg.

AE: Too many inline equations.

The projection can be carried out by the following moment matching procedure. Assume an ExpFam distribution $q(\theta|\eta) = h(\theta) \exp\left(\eta^\top u(\theta) - A(\eta)\right)$ where $u(\theta)$ is the sufficient statistic of q, η is the natural parameter and $A(\eta) = \log \int d\theta\, h(\theta) \exp\left(\eta^\top u(\theta)\right)$ is the log-partition function. It can be shown that $q^* = \text{proj}[r]$ satisfies $\mathbb{E}_{q^*(\theta)}[u(\theta)] = \mathbb{E}_{r(\theta)}[u(\theta)]$. That is, the projection of r onto ExpFam is given by $q^* \in \text{ExpFam}$ that has the same moment parameters as the moments under r.

Should jump to this general msg from the beginning.

In general, under the approximation that each factor fully factorizes, an EP message from a factor f to a variable V takes the form

$$m_{f \to V}(v) = \frac{\text{proj}\left[\int d\mathcal{V} \backslash \{v\}\, f(\mathcal{V}) \prod_{V' \in \mathcal{V}} m_{V' \to f}(v')\right]}{m_{V \to f}(v)} := \frac{\text{proj}\left[r_{f \to V}(v)\right]}{m_{V \to f}(v)} := \frac{q_{f \to V}(v)}{m_{V \to f}(v)} \tag{7}$$

where $\mathcal{V} = \mathcal{V}(f)$ is the set of variables connected to f in the factor graph. In the previous case of $m_{f_i \to \theta}$, we have $\mathcal{V}(f) = \{X, \theta\}$ and V in Eq. 7 corresponds to θ. Typically, when the factor f is complicated, the integral defining $r_{f \to V}$ becomes intractable. Quadrature rules or other numerical integration techniques are often applied to approximate the integral.

AE: This is the main point we want to get accross. Ei-

3 Learning to Pass EP Messages

Our goal is to learn a message operator $C_{f \to V'}$ with signature $[m_{V \to f}]_{V \in \mathcal{V}(f)} \mapsto q_{f \to V'}$ which takes in all incoming messages $\{m_{V \to f} \mid V \in \mathcal{V}(f)\}$ and outputs $q_{f \to V'}(v')$ i.e., the numerator of Eq. 7. For inference, we require one such operator for each recipient variable $V' \in \mathcal{V}(f)$ i.e., in total $|\mathcal{V}(f)|$ operators need to be learned for f. Operator learning is cast as a distribution-to-distribution regression problem where the training set $S_{V'} := \{([m^n_{V \to f}]_{V \in \mathcal{V}(f)}, q^n_{f \to V'})\}_{n=1}^{N}$ containing N incoming-outgoing message pairs can be generated as in Heess et al. [2013], Eslami et al. [2014] by importance sampling to compute the mean parameters $\mathbb{E}_{r_{f \to V'}(v')}[u(v')]$ for moment matching. In principle, the importance sampling itself can be used in EP for computing outgoing messages [Barthelmé and Chopin, 2011]. The scheme is, however, expensive as we need to draw a large number of samples for each outgoing message to be sent. In our case, the importance sampling is used for data set generation which is done offline before the actual inference.

The assumptions needed for the generation of a training set are as follows. First, we assume the factor f takes the form of a conditional distribution $f(v_1|v_2, \ldots, v_{|\mathcal{V}(f)|})$. Second, given $v_2, \ldots, v_{|\mathcal{V}(f)|}$, v_1 can be sampled from $f(\cdot|v_2, \ldots, v_{|\mathcal{V}(f)|})$. The ability to evaluate f is not assumed. Finally we assume that a distribution on the natural parameters of all incoming messages $\{m_{V \to f}\}_{V \in \mathcal{V}(f)}$ is available. The distribution is used solely to give a rough idea of incoming messages the learned operator will encounter during the actual EP inference. In practice, we only need to ensure that the distribution sufficiently covers the relevant region in the space of incoming messages.

AE: Thi sentence needs re working that you only trai on messa observed the mod

In recent years, there have been a number of works on the regression task with distributional inputs, including Poczos et al. [2013], Szabo et al. [2014] which mainly focus on the non-parametric case and are operated under the assumption that the samples from the distributions are observed but not the distributions themselves. In our case, the distributions (messages) are directly observed. Moreover, since the distributions are in ExpFam, they can be characterized by a finite-dimensional natural parameter vector or expected sufficient statistic. Hence, we can simplify our task to distribution-to-vector regression where the output vector contains a finite number of moments sufficient to characterize $q_{f \to V'}$. As regression input distributions are in ExpFam, one can also treat the task as vector-to-vector regression. However, seeing the inputs as distributions allows one to use kernels on distributions which are invariant to parametrization.

Once the training set $S_{V'}$ is obtained, any distribution-to-vector regression function can be applied to learn a message operator $C_{f\to V'}$. Given incoming messages, the operator outputs $q_{f\to V'}$ from which the outgoing EP message is given by $m_{f\to V'} = q_{f\to V'}/m_{V'\to f}$ which can be computed analytically. We opt for kernel ridge regression [Schölkopf and Smola, 2002] as our message operator for its simplicity, its potential use in an online setting (i.e., incremental learning during inference), and rich supporting theory.

4 Proposed Method

4.1 Kernel Ridge Regression

AE: Too many inline equations in this section. Pull out the most important ones?

We consider here the problem of regressing smoothly from distribution-valued inputs to feature-valued outputs. We follow the regression framework of Micchelli and Pontil [2005], with convergence guarantees provided by Caponnetto and De Vito [2007]. Under smoothness constraints, this regression can be interpreted as computing the conditional expectation of the output features given the inputs [Grünewälder et al., 2012].

Let $\mathsf{X} = (\mathsf{x}_1|\cdots|\mathsf{x}_N)$ be the training regression inputs and $\mathsf{Y} = \left(\mathbb{E}_{q^1_{f\to V'}} u(v')|\cdots|\mathbb{E}_{q^N_{f\to V'}} u(v')\right) \in \mathbb{R}^{D_y\times N}$ be the regression outputs. The ridge regression in the primal form seeks $\mathsf{W}\in\mathbb{R}^{D_y\times D}$ for the regression function $g(\mathsf{x}) = \mathsf{W}\mathsf{x}$ which minimizes the squared-loss function $J(\mathsf{W}) = \sum_{i=1}^N \|\mathsf{y}_i - \mathsf{W}\mathsf{x}_i\|_2^2 + \lambda \operatorname{tr}\left(\mathsf{W}\mathsf{W}^\top\right)$ where λ is a regularization parameter and tr denotes a matrix trace.

It is well known that the solution is given by $\mathsf{W} = \mathsf{Y}\mathsf{X}^\top\left(\mathsf{X}\mathsf{X}^\top + \lambda I\right)^{-1}$ which has an equivalent dual solution $\mathsf{W} = \mathsf{Y}\left(K+\lambda I\right)^{-1}\mathsf{X}^\top = A\mathsf{X}^\top$. The dual formulation allows one to regress from any type of input objects if a kernel can be defined between them. All the inputs enter to the regression function through the gram matrix $K\in\mathbb{R}^{N\times N}$ where $(K)_{ij} = \kappa(\mathsf{x}_i,\mathsf{x}_j)$ yielding the regression function of the form $g(\mathsf{x}) = \sum_{i=1}^N a_i\kappa(\mathsf{x}_i,\mathsf{x})$ where $A := (a_1|\cdots|a_N)$. The dual formulation therefore allows one to straightforwardly regress from incoming messages to vectors of mean parameters.Although this property is appealing, the training size N in our setting can be chosen to be arbitrarily large, making computation of $g(\mathsf{x})$ expensive for a new unseen point x.

AE: Best not to use 'in our setting' until section 4 (proposed method). This section is still 'background', right?

To eliminate the dependency on N, we propose to apply random Fourier features [Rahimi and Recht, 2007] $\hat\phi(\mathsf{x})\in\mathbb{R}^D$ for $\mathsf{x} := [m_{V\to f}]_{V\in\mathcal{V}(f)}$ such that $\kappa(\mathsf{x},\mathsf{x}') \approx \hat\phi(\mathsf{x})^\top\hat\phi(\mathsf{x}')$ where D is the number of random features. The use of the random features allows us to go back to the primal form of which the regression function $g(\hat\phi(\mathsf{x})) = \mathsf{W}\hat\phi(\mathsf{x})$ can be computed efficiently. In effect, computing an EP outgoing message requires nothing more than a multiplication of a matrix W ($D_y\times D$) with the D-dimensional feature vector generated from the incoming messages.

AE: Best not to use 'we propose'

4.2 Kernels on Distributions

A number of kernels on distributions have been studied in the literature [Jebara and Kondor, 2003, Jebara et al., 2004]. Relevant to us are kernels whose random features can be efficiently computed. Due to the space constraint, we only give a few examples here.

Expected Product Kernel Let $\mu_{r^{(l)}} := \mathbb{E}_{r^{(l)}(a)}k(\cdot,a)$ be the mean embedding [Smola et al., 2007] of the distribution $r^{(l)}$ into RKHS $\mathcal{H}^{(l)}$ induced by the kernel k. Assume $k = k_{\text{gauss}}$ (Gaussian kernel) and assume there are c incoming messages $\mathsf{x} := (r^{(i)}(a^{(i)}))_{i=1}^c$ and $\mathsf{y} := (s^{(i)}(b^{(i)}))_{i=1}^c$. The expected product kernel κ_{pro} is defined as

$$\begin{aligned}\kappa_{\text{pro}}(\mathsf{x},\mathsf{y}) &:= \left\langle \bigotimes_{l=1}^c \mu_{r^{(l)}}, \bigotimes_{l=1}^c \mu_{s^{(l)}} \right\rangle_{\otimes_l\mathcal{H}^{(l)}} \\ &= \prod_{l=1}^c \mathbb{E}_{r^{(l)}(a)}\mathbb{E}_{s^{(l)}(b)} k^{(l)}_{\text{gauss}}(a,b) \approx \hat\phi(\mathsf{x})^\top\hat\phi(\mathsf{y})\end{aligned}$$

where $\hat\phi(\mathsf{x})^\top\hat\phi(\mathsf{y}) = \prod_{l=1}^c \hat\phi^{(l)}(r^{(l)})^\top\hat\phi^{(l)}(s^{(l)})$. The feature map $\hat\phi^{(l)}(r^{(l)})$ can be estimated by applying the random Fourier features to $k^{(l)}_{\text{gauss}}$ and taking the expectations $\mathbb{E}_{r^{(l)}(a)}\mathbb{E}_{s^{(l)}(b)}$. The final feature map is $\hat\phi(\mathsf{x}) = \hat\phi^{(1)}(r^{(1)}) \circledast \hat\phi^{(2)}(r^{(2)}) \circledast \cdots \circledast \hat\phi^{(c)}(r^{(c)}) \in \mathbb{R}^{d^c}$ where $\circledast$ denotes a Kronecker product and we assume that $\hat\phi^{(l)}\in\mathbb{R}^d$ for $l\in\{1,\ldots,c\}$.

Kernel on Joint Embeddings Another way to define a kernel on x,y is to mean-embed both joint distributions $\mathsf{r} = \prod_{i=1}^c r^{(i)}$ and $\mathsf{s} = \prod_{i=1}^c s^{(i)}$ and define the kernel to be $\kappa_{\text{joint}}(\mathsf{x},\mathsf{y}) := \langle\mu_{\mathsf{r}},\mu_{\mathsf{s}}\rangle_{\mathcal{G}}$ where $\mathcal{G}$ is an RKHS consisting of functions $g : \mathcal{X}^{(1)}\times\cdots\times\mathcal{X}^{(c)}\to\mathbb{R}$ and $\mathcal{X}^{(l)}$ denotes the domain of $r^{(l)}$ and $s^{(l)}$. This kernel is equivalent to the expected product kernel on the joint distributions.

TODO

- Add forest cost computed by Ali. Compare it with online kernel ridge regression.

5 Related Work

6 Experiments

Experiments to have

- Experiments on the logistic factor for comparing with Nicolas and Ali's results. Show prediction accuracy. Histogram of log KL. 4 plots (best, 2 median, worst) of predicted msgs vs. true.

- Plot the predictive variance as a test point (incoming messages) is far away from the training set. Say the mean of the means of Gaussian incoming messages in the training set is 0. Vary the mean of the Gaussian message from -10 to 10 on x-axis and plot predictive variance on y-axis. Expect a U-shaped curve where the minimum is at 0. This ensures the operator will query the oracle when there are incoming messages from an unexplored region. Perhaps plot one curve for one setting of the incoming Beta message (fixed).

AE: Need to talk about online training, concept of an oracle, concept of querying oracle when uncertain.

- For predictive variance on unseen region, we may need two plots where in one case the training samples are concentrated in one part of the space, and in the second case the training samples are concentrated in another part. This is to ensure that the uncertainty estimate does not depend on the absolute location.

- Use the operator for logistic in a binary logistic regression problem with a real dataset. Compare the inferred posterior of the coefficient vector to the posterior obtained by the handcrafted message operator, possibly with a histogram of log KL (of posteriors).

- Experiments on compound gamma factor whose handcrafted operator is slow. We can show that using a regression-based operator can be faster and possibly more accurate. A compound gamma can be used as a prior on a Gaussian precision in some model. Would be nice to have a real dataset (say a binary classification dataset from UCI repository).

  – Andrew Gelman [Gelman, 2006], Tom Minka: "Gamma prior on precision is not good". Cite them.

- Do we need kNN as a baseline regression function?

AE: Maybe enough to point to JIT paper and say KNN performs badly there?

- Choose the number of particles in the oracle importance sampler so that it takes roughly the same time as our operator. Compute the predictive quality of the two.

- As in Fig 3.c of Ali's paper, show that the inference error of sampling + JIT is better than sampling alone. This asserts that JIT interpolates in the message space in a useful way.

AE: This would be nice but is probably not critical.

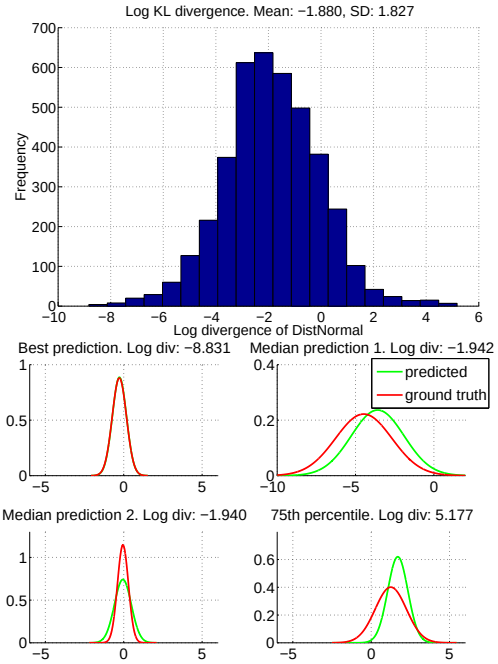

Figure 1: Log KL-divergence on a logistic factor test set using kernel on joint embeddings.

As a preliminary experiment, we consider the logistic factor $f(z|x) = \delta\left(z - \frac{1}{1+\exp(-x)}\right)$ which is deterministic when conditioning on x. The factor is used in many common models including binary logistic regression. The two incoming messages are $m_{X\to f}(x) = \mathcal{N}(x;\mu,\sigma^2)$ and $m_{Z\to f}(z) = \text{Beta}(z;\alpha,\beta)$. We randomly generate 2000 training input messages and learn a message operator using the kernel on joint embeddings. Kernel parameters are chosen by cross validation and the number of random features D is set to 2000. We report $\log KL[q\|\hat{q}]$ where $q = q_{f\to X}$ is the ground truth output message obtained by importance sampling and $\hat{q}$ is the message output from the operator. For better numerical scaling, regression outputs are set to $(\mathbb{E}_q[x], \log \mathbb{V}_q[x])$ instead of the expectations of the first two moments. The histogram of log KL errors is shown on the left of Fig. 1. The right figure shows sample output messages at different log KL errors. It can be seen that the operator is able to capture the relationship of incoming and outgoing messages. With higher training size, increased number of random features and well chosen kernel parameters, we expect to see a significant improvement in the operator's accuracy.

AE: I th at this p you nee properly fine the gistic re sion mo otherwis coming sages wi incompr hensible

AE: Wh 2000?

AE: No to menti log-scali

7 Conclusions and Future Work

We propose to learn to send EP messages with kernel ridge regression by casting the KL minimization problem as a supervised learning problem. With ran-

dom features, incoming messages to a learned operator are converted to a finite-dimensional vector. Computing an outgoing message amounts to computing the moment parameters by multiplying the vector with a matrix given by the solution of the primal ridge regression.

By virtue of the primal form, it is straightforward to derive an update equation for an online-active learning during EP inference: if the predictive variance (similar to a Gaussian process) on the current incoming messages is high, then we query the outgoing message from the importance sampler (oracle) and update the operator. Otherwise, the outgoing message is efficiently computed by the operator. Online learning of the operator lessens the need of the distribution on natural parameters of incoming messages used in training set generation. Determining an appropriate distribution was one of the unsolved problems in Heess et al. [2013].

AE: These paragraphs are very informative. Would be great to have them in some form towards the beginning of the paper too. 1. Say what you're going to say. 2. Say it. 3. Say what you just said. there

References

S. Barthelmé and N. Chopin. Abc-ep: Expectation propagation for likelihood-free bayesian computation. In L. Getoor and T. Scheffer, editors, *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, ICML '11, pages 289–296, New York, NY, USA, June 2011. ACM. ISBN 978-1-4503-0619-5.

C. M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006. ISBN 0387310738.

A. Caponnetto and E. De Vito. Optimal rates for the regularized least-squares algorithm. *Found. Comput. Math.*, 7(3):331–368, July 2007. ISSN 1615-3375. doi: 10.1007/s10208-006-0196-8. URL http://dx.doi.org/10.1007/s10208-006-0196-8.

S. M. A. Eslami, D. Tarlow, P. Kohli, and J. Winn. Just-In-Time Learning for Fast and Flexible Inference. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 154–162, 2014.

A. Gelman. Prior distributions for variance parameters in hierarchical models. *Bayesian Analysis*, 1:1–19, 2006.

S. Grünewälder, G. Lever, L. Baldassarre, S. Patterson, A. Gretton, and M. Pontil. Conditional mean embeddings as regressors. In J. Langford and J. Pineau, editors, *Proceedings of the 29th International Conference on Machine Learning*, pages 1823–1830, New York, NY, USA, 2012. Omnipress. URL http://icml.cc/2012/papers/898.pdf.

N. Heess, D. Tarlow, and J. Winn. Learning to pass expectation propagation messages. In C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 3219–3227. 2013.

F. Hutter. *Automated Configuration of Algorithms for Solving Hard Computational Problems*. PhD thesis, University of British Columbia, Department of Computer Science, Vancouver, Canada, October 2009.

T. Jebara and R. Kondor. Bhattacharyya and expected likelihood kernels. In *Conference on Learning Theory*. press, 2003.

T. Jebara, R. Kondor, and A. Howard. Probability product kernels. *J. Mach. Learn. Res.*, 5:819–844, Dec. 2004. ISSN 1532-4435. URL http://dl.acm.org/citation.cfm?id=1005332.1016786.

C. A. Micchelli and M. A. Pontil. On learning vector-valued functions. *Neural Comput.*, 17(1):177–204, Jan. 2005. ISSN 0899-7667. doi: 10.1162/0899766052530802. URL http://dx.doi.org/10.1162/0899766052530802.

T. P. Minka. *A Family of Algorithms for Approximate Bayesian Inference*. PhD thesis, Massachusetts Institute of Technology, 2001. AAI0803033.

B. Poczos, A. Rinaldo, A. Singh, and L. Wasserman. Distribution-free distribution regression. *arXiv:1302.0082 [cs, math, stat]*, Feb. 2013. URL http://arxiv.org/abs/1302.0082. arXiv: 1302.0082.

A. Rahimi and B. Recht. Random features for large-scale kernel machines. In *Neural Information Processing Systems*, 2007.

B. Schölkopf and A. J. Smola. *Learning with kernels : support vector machines, regularization, optimization, and beyond*. Adaptive computation and machine learning. MIT Press, 2002.

A. Smola, A. Gretton, L. Song, and B. Schölkopf. A hilbert space embedding for distributions. In *In Algorithmic Learning Theory: 18th International Conference*, pages 13–31. Springer-Verlag, 2007.

Stan Development Team. Stan: A c++ library for probability and sampling, version 2.4, 2014. URL http://mc-stan.org/.

Z. Szabo, A. Gretton, B. Poczos, and B. Sriperumbudur. Consistent, two-stage sampled distribution regression via mean embedding. Feb. 2014. URL http://arxiv-web3.library.cornell.edu/abs/1402.1754v2.