Kernel Belief Propagation

February 22, 2021

Motivation

- Difficult to derive belief propagation messages for continuous RVs with complex densities, which typically rely on easy to compute conditionals (ie conjugacy or discrete)
- Instead, rewrite messages using **nonparametric** representations of densities, i.e. sums of points in some space with no explicit parameters
- Approach extends to any domain on which kernels can be defined, such as strings and graphs

Why is this interesting?

- Background is relevant for spectral + kernel methods in latent variable models
- Keyword overlap with recent work in efficient attention

Preview: Comparison with Performer-style Inference

- Both do inference with kernels
- Have 3 terms in complexity in common: size of state space, feature dimension, number of samples
- Performer relies on explicit computation of inner products with RFF, so complexity is a function of feature dimension and size of state space
- KBP emphasizes approximate nonparametric inference, complexity is a function of number of samples. Feature dimension is always avoided with kernel trick. Approximations are made to reduce dependence on number of samples.

Learning in Markov Random Fields

- Pairwise MRF (typically parameterize log potentials)

$$\mathbb{P}(X) \propto \prod_{s,t \in \mathcal{E}} \Psi_{st}(X_s, X_t) \prod_{s \in \mathcal{V}} \Psi(X_s).$$

- Estimate gradients wrt log potentials by computing edge and node marginals via inference, ie the beliefs $\mathbb{B}(X_s, X_t)$ and $\mathbb{B}(X_s)$

- Belief propagation is an algorithm for performing inference

Belief Prop (BP)

- BP propagates messages from nodes to neighbours iteratively until convergence
- Messages from node t to s

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

- Belief at s

$$\mathbb{B}(X_s) = \Psi_s(X_s) \prod_{t \in \delta(s)} m^*_{ts}(X_s),$$

with fixed point messages m^*

BP

- The integrals in the messages may be difficult to compute
- Solution: Rewrite messages as an expectation (by dividing by fixed point messages)[1], then approximate conditional

$$
\begin{aligned}
m'_{ts}(X_s) &= \int_{\mathcal{X}} \mathbb{P}^*(X_t \mid X_s) \prod_{u \in \delta(t) \setminus \{s\}} m'_{ut}(X_t) dX_t \\
&= \mathbb{E}_{X_t | X_s} \left[\prod_{u \in \delta(t) \setminus \{s\}} m'_{ut}(X_t) \right]
\end{aligned}
$$

- Requires fully observed model, otherwise stuck with original integral
- *Take closer look at reparameterization (eqns 1-4), discuss how that affects algorithm

[1] Is this an existence statement about messages?

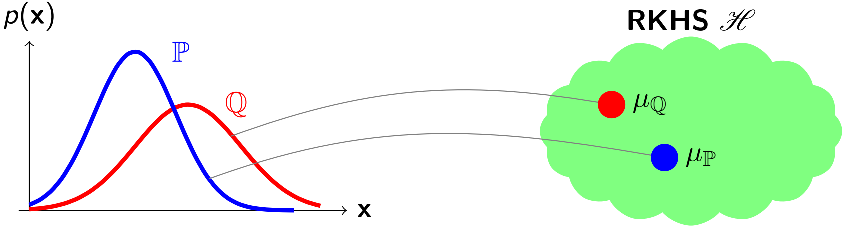
Issues with Nonparametric BP Baselines

- NPBP baselines are Gaussian Mixture BP (Sudderth et al, 2003) and Particle BP (Ihler and McAllester, 2009)
- They claim NPBP requires a 2-step process of estimating conditional $\mathbb{P}^*(X_t \mid X_s)$, then computing messages
- Kernel BP reduces this to a single step of matrix-vector products

High Level Overview of Kernel Belief Propagation

- Embed messages in RKHS
- Approximate expectations via observed samples
- Compute messages with inner products

Warmup: Kernel Mean Embedding



- Kernel mean embeddings map distributions into Hilbert spaces
- Can approximate embedding in RKHS via sampling

Kernel Mean Embedding

- Definition:
$$\mu_X(\cdot) = \mathbb{E}_X[\phi_X(\cdot)],$$
with feature map $\phi = k(x, \cdot) \in \mathscr{H}$

- Goal in this paper is to write everything as an inner product and then apply kernel trick

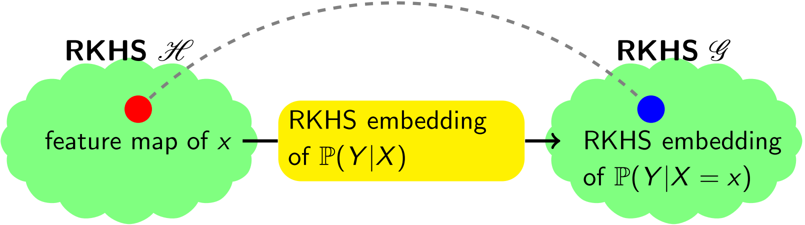
- Using the reproducing property and linearity:
$$\mathbb{E}_X[f(X)] = \mathbb{E}_X[\langle f, \phi_X \rangle] = \langle f, \mathbb{E}_X[\phi_X] \rangle = \langle f, \mu_X \rangle, \forall f \in \mathscr{H},$$
so we can write expected functions of RVs as inner products

- Consider categorical distribution with $\phi(\mathbf{x})$ one-hot: $\mu_X = P(X)$.

Kernel Mean Embedding: Empirical Estimation

- Some notation: With samples $\{X^n\}_{n=1}^N \sim \mathbb{P}(X)$, let $\Phi = [\phi(X^1), \cdots, \phi(X^N)] \in \mathbb{R}^{D \times N}$ be the feature matrices (assuming finite dim D feature space)
- We can approximate $\mu_X \approx \Phi^\top \mathbf{1}/m$
- Will use Φ again later, in the form of kernel matrix $K = \Phi^\top \Phi$

Conditional Mean Embedding



- Embed conditional probability function as an operator
- Conditioning operator is a family of functions of y indexed by x

Conditional Mean Embedding

- Going to need a series of definitions
- Goal is to write $\mathbb{E}_{Y|X}[g(Y)]$ as an inner product
  - Recall that messages were rewritten as conditional expectations wrt other messages
  - We consider the case when g is a message
- Will get there by defining the conditional mean embedding then applying similar derivation as marginal mean embedding

Conditional Mean Embedding

- Two Hilbert spaces $\mathscr{H}, \mathscr{G}$ for RVs X, Y
- Define uncentered cross-covariance operator

$$C_{YX} = \mathbb{E}_{YX}[\varphi_Y \otimes \phi_X],$$

  for $f \in \mathscr{H}, g \in \mathscr{G}$ and $C_{YX} : \mathscr{H} \to \mathscr{G}$
- Has property

$$\mathbb{E}_{YX}[g(Y)f(X)] = \langle g, C_{YX} f\rangle = \langle g \otimes f, C_{YX}\rangle,$$

  which extends the evaluation property to two variables. (Can have $Y = X$ for autocorrelation)
  Pf:

$$\begin{aligned}
\mathbb{E}_{YX}[g(Y)f(X)] &= \mathbb{E}_{YX}[\langle g, \phi_Y\rangle\langle f, \phi_X\rangle] \\
&= \mathbb{E}_{YX}[\langle g, \langle f, \phi_X\rangle\phi_Y\rangle] \\
&= \mathbb{E}_{YX}[\langle g, (\phi_X \otimes \phi_Y)f\rangle] \\
&= \langle g, \mathbb{E}_{YX}[\phi_X \otimes \phi_Y]f\rangle \\
&= \langle g, C_{YX} f\rangle
\end{aligned}$$

Conditional Mean Embedding

- Consider the property $C_{XX}\mathbb{E}_{Y|X}[g(Y) \mid X] = C_{XY}g$ (Fukumitsu 2004).
  Pf: $\forall f \in \mathscr{H}$,

$$\begin{aligned}
\langle f, C_{XX}\mathbb{E}_{Y|X}[g(Y) \mid X]\rangle &= \mathbb{E}_X[f(X)\mathbb{E}_{Y|X}[g(Y) \mid X]] \\
&= \mathbb{E}_{XY}[f(X)g(Y)] \\
&= \langle f, C_{XY}g\rangle
\end{aligned}$$

- Using the above property,

$$\begin{aligned}
\mathbb{E}_{Y|X=x}[g(Y) \mid X = x] &= \langle \mathbb{E}_{Y|X}[g(Y) \mid X], \phi_x\rangle \\
&= \langle C_{XX}^{-1}C_{XY}g, \phi_x\rangle \\
&= \langle g, C_{YX}C_{XX}^{-1}\phi_x\rangle
\end{aligned}$$

Conditional Mean Embedding: Empirical Estimation

- Samples $\{X^n\}_{n=1}^N \sim \mathbb{P}(X), \{Y^n\}_{n=1}^N \sim \mathbb{P}(Y)$,
- Feature matrices (assuming finite feature dim)

$$\Phi = [\phi(X^1), \cdots, \phi(X^N)] \in \mathbb{R}^{D \times N},$$

$$\Upsilon = [\varphi(Y^1), \cdots, \varphi(Y^N)] \in \mathbb{R}^{D \times N},$$

  with feature maps ϕ, φ for Hilbert spaces $\mathscr{H}, \mathscr{G}$ of functions from $\mathcal{X} \mapsto \mathbb{R}, \mathcal{Y} \mapsto \mathbb{R}$ respectively
- We can approximate cross-covariance operator with $C_{YX} \approx \Upsilon \Phi^\top / m$

Product Space Embeddings

- We have to deal with multiple neighbours and messages
- Consider product Hilbert space $\mathscr{H}^{\otimes} = \prod_i \mathscr{H}_i$ and product feature map ξ, so that $\xi(x) = \otimes_i \phi_i(x)$, the feature maps of the underlying spaces
- Recall tensor product $(f \otimes g)h = \langle g, h \rangle f$ where $f, g, h \in \mathscr{H}$
- For the case of message passing, generalizes to $\prod_i \langle f, \phi(x) \rangle = \langle \otimes_u f, \xi(x) \rangle$
- Not totally clear on the details, but can view $\xi(x)$ as just another function

Kernel Belief Propagation Messages

- Back to message passing:

$$\begin{aligned}
m_{ts}(x_s) &= \mathbb{E}_{X_t|x_s}[\prod_{u \in \delta(t) \backslash \{s\}} m_{ut}(X_t)] \\
&= \mathbb{E}_{X_t|x_s}[\prod_{u \in \delta(t) \backslash \{s\}} \langle m_{ut}, \phi_{X_t} \rangle] \\
&= \mathbb{E}_{X_t|x_s}[\langle \otimes_{u \backslash s} m_{ut}, \xi_{X_t} \rangle] \\
&= \langle \otimes_{u \backslash s} m_{ut}, \mathbb{E}_{X_t|x_s}[\xi_{X_t}] \rangle \\
&= \langle \otimes_{u \backslash s} m_{ut}, C_{X_t^{\otimes} X_s} C_{X_s X_s}^{-1} \phi_{x_s} \rangle
\end{aligned}$$

- And that's it! Now for empirical estimation and efficient computation

Empirical Estimation

- Key point: Avoids ever instantiating tensor products.[2]
- N data points, D feature dim
- Feature matrices $\Phi, \Upsilon, \Phi^{\otimes} : R^{D\times N}$ for $X_t, X_s, X_t^{\otimes}$ and kernels $K = \Phi^\top \Phi, L = \Upsilon^\top \Upsilon$

$$\begin{aligned} m_{ts}(x_s) &= \langle \otimes_{u \backslash s} m_{ut}, C_{X_t^{\otimes} X_s} C^{-1}_{X_s X_s} \phi_{x_s} \rangle \\ &\approx \langle \otimes_{u \backslash s} \Phi \beta_{ut}, \Phi^{\otimes} L^{-1} \phi_{x_s} \Upsilon^\top \rangle \\ &= (\odot_{u \backslash s} K \beta_{ut})^\top L^{-1} \Upsilon^\top \phi_{x_s}, \end{aligned}$$

  where $\beta_{ut} \in \mathbb{R}^N$ is some function of K, L and neighouring β
- Upfront $O(N^3)$ matrix inversion cost, $O(|\delta^*|N^2)$ cost per message
- Approximations use low-rank approximations of kernel matrices and tensor product to limit dependence on N and $|\delta^*|$

[2] This is what messed up the runtime in my version of kernelized inference. Not sure if this transfers to our setting, but likely does since it seems to be a property of tensor products.

Conclusion

- Approximations not applicable in Performer setting, would just subsample and reweight timesteps
- Avoid space blowup from tensor products by replacing with elementwise vector product
- Next: figure out how they train with latent variables

Definitions

- Domain $\mathcal{X}$
- Hilbert space $\mathscr{H}$ of functions on $\mathcal{X} \mapsto \mathbb{R}$ with inner product $\langle \cdot, \cdot \rangle$, kernel K, and feature map ϕ
- The point evaluation property, ie that function evaluation is an inner product,

$$\langle f, K(x, \cdot) \rangle = f(x),$$

implies the reproducing property:

$$\langle K(x, \cdot), K(y, \cdot) \rangle = K(x, y) = \langle \phi(x), \phi(y) \rangle$$

Theorem Notes

- Riesz representation theorem: If operator $\mathcal{A} : \mathscr{H} \to \mathbb{R}$ is bounded, then there exists a representer $g_{\mathcal{A}} \in \mathscr{H}$ st
$$A[f] = \langle f, g_{\mathcal{A}} \rangle, \forall f \in \mathscr{H}.$$
- Point evaluation property: In an RKHS, consider the evaluation functional $\mathcal{F}_{\mathbf{x}}(f) = f(\mathbf{x})$. Riesz representation theorem tells us there exists a representer $k_{\mathbf{x}} : \mathscr{H} \to \mathbb{R}$ st
$$\mathcal{F}_{\mathbf{x}}(f) = \langle f, k_{\mathbf{x}} \rangle = f(\mathbf{x}),$$
referred to as the reproducing kernel for the point $\mathbf{x}$.
- The reproducing property is a special case of the point evaluation property. Consider the kernel $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, and define $f(\mathbf{x}) = k(\mathbf{y}, \mathbf{x})$ for all $\mathbf{y} \in \mathcal{X}$. Applying the point evaluation property yields
$$f(\mathbf{x}) = \langle k(\mathbf{x}, \cdot), k(\mathbf{y}, \cdot) \rangle,$$
where $k(\mathbf{x}, \cdot)$ is the canonical feature map denoted by $\phi : \mathcal{X} \to \mathscr{H}$.
- Alternatively you can start by assuming the kernel is positive

Refs