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

Wittawat Jitkrittum,[1] Arthur Gretton,[1] Nicolas Heess, S. M. Ali Eslami
Balaji Lakshminarayanan,[1] Dino Sejdinovic[2] and Zoltán Szabó[1]
Gatsby Unit, University College London[1] University of Oxford[2]

Introduction

EP is a widely used message passing based inference algorithm.

- **Problem**: Expensive to compute outgoing from incoming messages.
- **Goal**: Speed up computation by a cheap regression function (message operator):

incoming messages $\mapsto$ outgoing message.

Merits:

- Efficient online update of the operator during inference.
- Uncertainty monitored to invoke new training examples when needed.
- Automatic random feature representation of incoming messages.

Expectation Propagation (EP)

Under an approximation that each factor fully factorizes, an outgoing EP message $m_{f\to V_i}$ takes the form

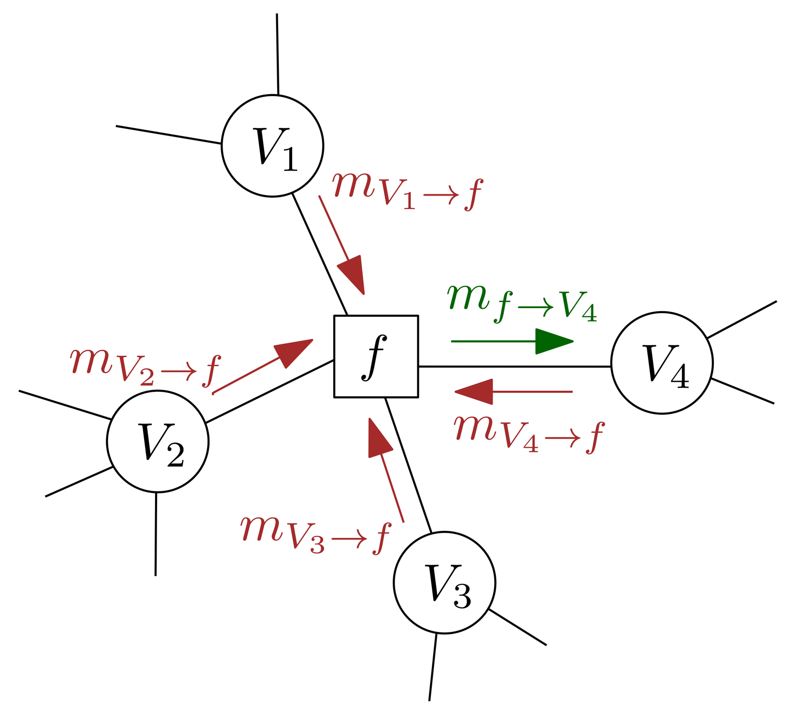
set of c variables connected to f

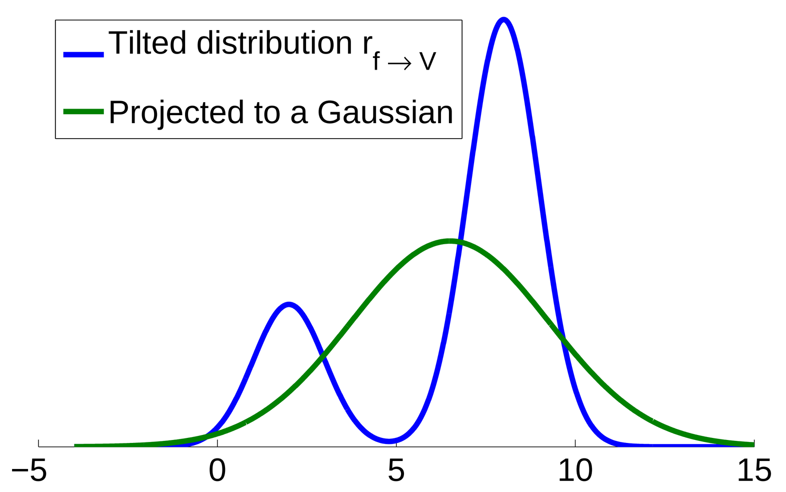
projected message

$$m_{f\to V_i}(v_i) = \frac{\mathrm{proj}\left[\int f(\mathcal{V}) \prod_{j=1}^{c} m_{V_j\to f}(v_j)\, \mathrm{d}\mathcal{V}\backslash\{v_i\}\right]}{m_{V_i\to f}(v_i)} := \frac{q_{f\to V_i}(v_i)}{m_{V_i\to f}(v_i)}$$

$\mathrm{proj}[r_{f\to V_i}] := \arg\min_{q\in\mathrm{ExpFam}} \mathrm{KL}\left[r_{f\to V_i} \,\|\, q\right]$
(projection onto exponential family)

incoming message from V_j





Projected message:

- $q_{f\to V}(v) = \mathrm{proj}\left[r_{f\to V}(v)\right] \in \mathrm{ExpFam}$ with sufficient statistic $u(v)$.
- Compute $q_{f\to V}(v)$ by moment matching: $\mathbb{E}_{q_{f\to V}}[u(v)] = \mathbb{E}_{r_{f\to V}}[u(v)]$.

Kernel on Incoming Messages

Propose to incrementally learn during inference a kernel-based EP message operator (distribution-to-distribution regression)

$$\left[m_{V_j\to f}\right]_{j=1}^{c} \mapsto q_{f\to V_i},$$

for any factor f that can be sampled.

- Product distribution of c incoming messages: $\mathsf{r} := \times_{l=1}^{c} r_l, \quad \mathsf{s} := \times_{l=1}^{c} s_l$.
- Mean embedding of r: $\mu_{\mathsf{r}} := \mathbb{E}_{a\sim \mathsf{r}} k(\cdot, a)$.
- Gaussian kernel on (product) distributions. Two-staged random feature approx.:

$$\kappa(\mathsf{r},\mathsf{s}) = \exp\left(-\frac{\|\mu_{\mathsf{r}} - \mu_{\mathsf{s}}\|_{\mathcal{H}}^2}{2\gamma^2}\right) \overset{1^{st}}{\approx} \exp\left(-\frac{\|\hat{\phi}(\mathsf{r}) - \hat{\phi}(\mathsf{s})\|_{D_{\mathrm{in}}}^2}{2\gamma^2}\right) \overset{2^{nd}}{\approx} \hat{\psi}(\mathsf{r})^\top \hat{\psi}(\mathsf{s}).$$

Message Operator: Bayesian Linear Regression

- **Input:** $\mathsf{X} = (\mathsf{x}_1|\cdots|\mathsf{x}_N)$: N training incoming messages represented as random feature vectors.
- **Output:** $\mathsf{Y} = \left(\mathbb{E}_{r^1_{f\to V}} u(v)|\cdots|\mathbb{E}_{r^N_{f\to V}} u(v)\right) \in \mathbb{R}^{D_y\times N}$: expected sufficient statistics of outgoing messages.
- Inexpensive online updates of posterior mean and covariance.
- Bayesian regression gives prediction and predictive variance.
- If predictive variance $>$ threshold, query the importance sampling oracle.

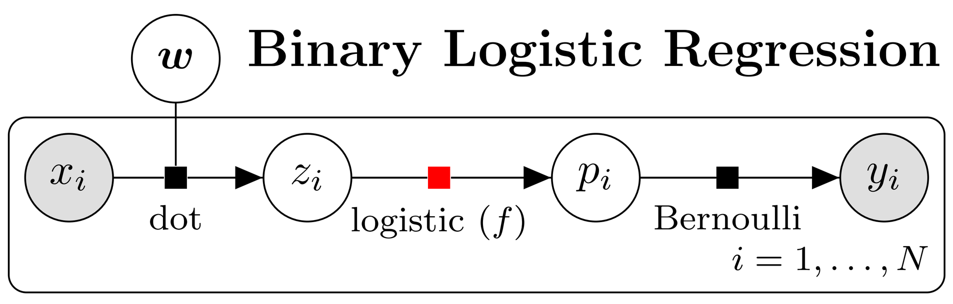
Two-Staged Random Features

In: $\mathcal{F}(k)$: Fourier transform of k, D_{in}: #inner features, D_{out}: #outer features, k_{gauss}: Gaussian kernel on $\mathbb{R}^{D_{\mathrm{in}}}$

Out: Random features $\hat{\psi}(\mathsf{r}) \in \mathbb{R}^{D_{\mathrm{out}}}$

1. Sample $\{\omega_i\}_{i=1}^{D_{\mathrm{in}}} \overset{i.i.d}{\sim} \mathcal{F}(k)$, $\quad \{b_i\}_{i=1}^{D_{\mathrm{in}}} \overset{i.i.d}{\sim} U[0, 2\pi]$.
2. $\hat{\phi}(\mathsf{r}) = \sqrt{\frac{2}{D_{\mathrm{in}}}}\left(\mathbb{E}_{x\sim \mathsf{r}} \cos(\omega_i^\top x + b_i)\right)_{i=1}^{D_{\mathrm{in}}} \in \mathbb{R}^{D_{\mathrm{in}}}$
3. Sample $\{\nu_i\}_{i=1}^{D_{\mathrm{out}}} \overset{i.i.d}{\sim} \mathcal{F}(k_{\mathrm{gauss}}(\gamma^2))$, $\quad \{c_i\}_{i=1}^{D_{\mathrm{out}}} \overset{i.i.d}{\sim} U[0, 2\pi]$.
4. $\hat{\psi}(\mathsf{r}) = \sqrt{\frac{2}{D_{\mathrm{out}}}}\left(\cos(\nu_i^\top \hat{\phi}(\mathsf{r}) + c_i)\right)_{i=1}^{D_{\mathrm{out}}} \in \mathbb{R}^{D_{\mathrm{out}}}$

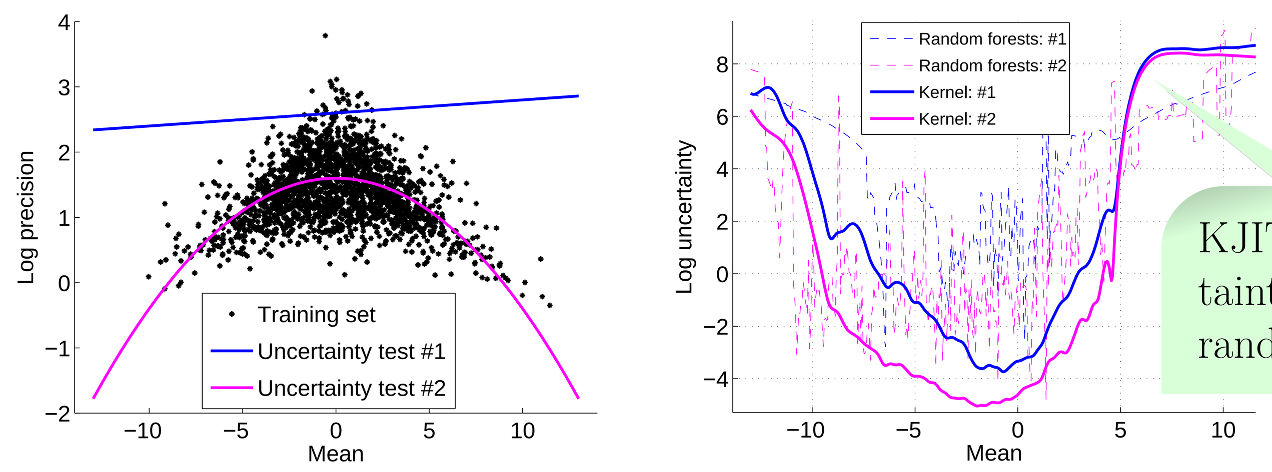
Experiment 1: Uncertainty Estimates



- Approx. $f(p|z) = \delta\left(p - \frac{1}{1+\exp(-z)}\right)$.
- Incoming messages:

$$m_{z_i\to f} = \mathcal{N}(z_i; \mu, \sigma^2),$$
$$m_{p_i\to f} = \mathrm{Beta}(p_i; \alpha, \beta).$$

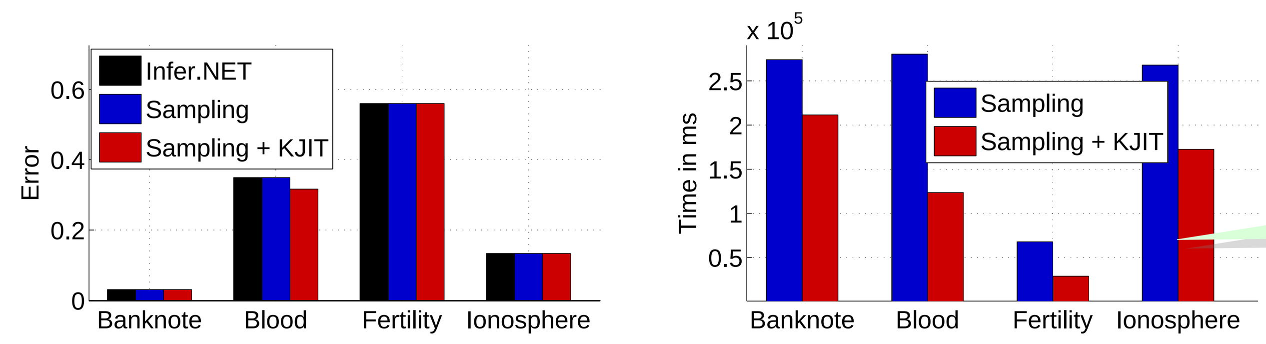
- Training messages collected from 20 EP runs on toy data.
- #Random features: $D_{in} = 300$ and $D_{out} = 500$.



KJIT gives smoother uncertainty estimates compared to random forests.

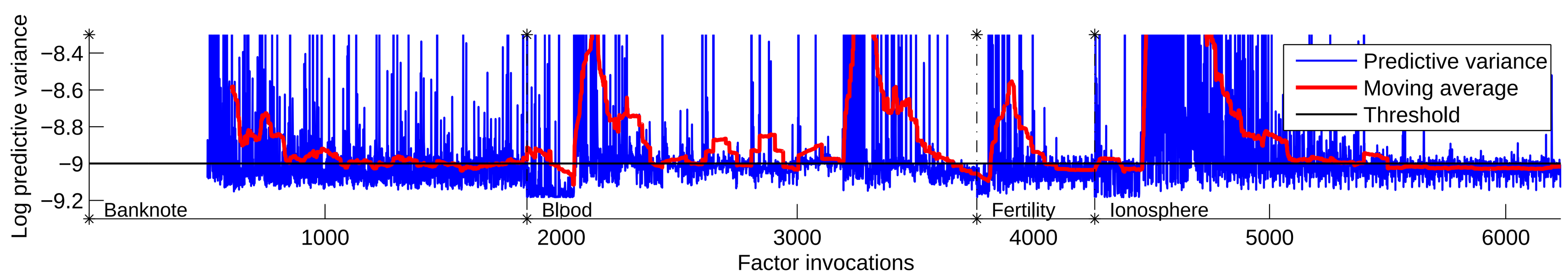
Experiment 2: Real Data

- Binary logistic regression. Sequentially present 4 real datasets to the operator.
- Diverse distributions of incoming messages.



Much faster with same classification errors as obtained by importance sampling.

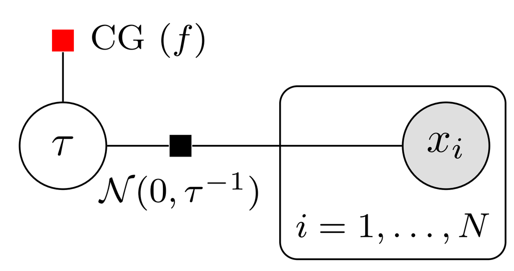
- **Sampling + KJIT** = proposed KJIT with an importance sampling oracle.
- KJIT operator can adapt to the change of input message distributions.

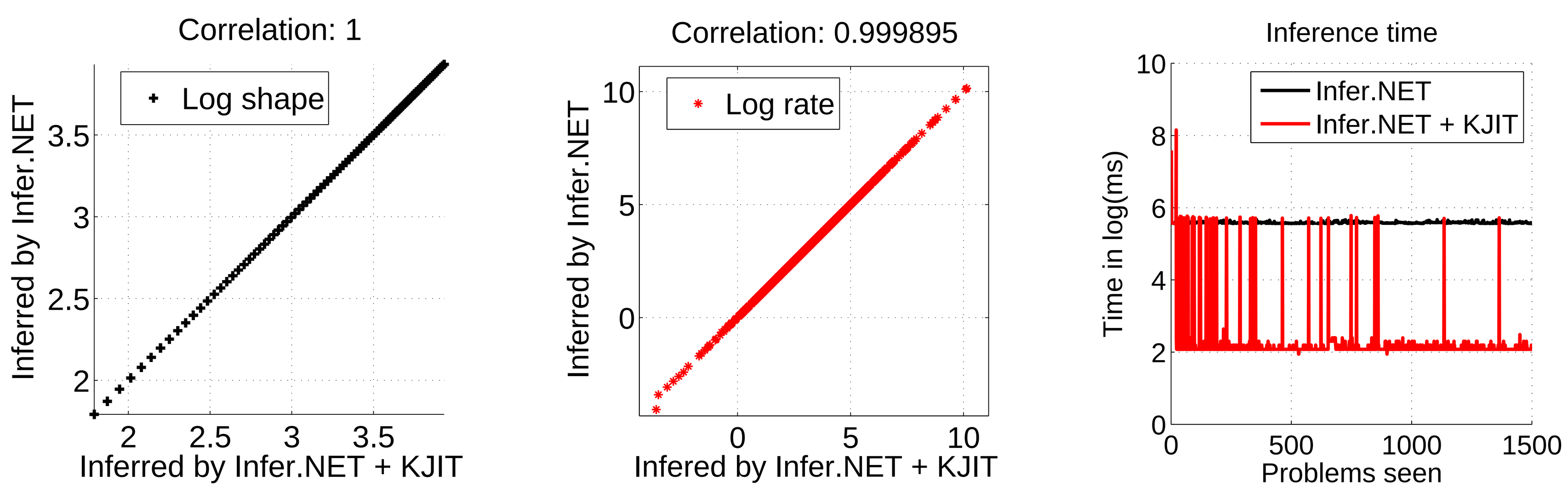


Experiment 3: Compound Gamma Factor

Infer posterior of the precision τ of $x \sim \mathcal{N}(x; 0, \tau^{-1})$ from observations $\{x_i\}_{i=1}^{N}$:

$$r_2 \sim \mathrm{Gamma}(r_2; s_1, r_1)$$
$$\tau \sim \mathrm{Gamma}(\tau; s_2, r_2)$$
$$(s_1, r_1, s_2) = (1, 1, 1).$$





- **Infer.NET + KJIT** = proposed KJIT with a hand-crafted factor as oracle.
- **Inference quality**: as good as hand-crafted factor; much faster.

Contact: `wittawat@gatsby.ucl.ac.uk`.

Code download: `http://github.com/wittawatj/kernel-ep`.

We thank the Gatsby Charitable Foundation for the financial support.