Just-In-Time Learning for Fast and Flexible Inference

S. M. Ali Eslami, Daniel Tarlow, Pushmeet Kohli and John Winn

{alie,dtarlow,pkohli,jwinn}@microsoft.com

Microsoft Research

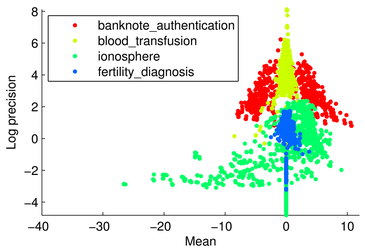
1. Goals

To combine the **flexibility** of sampling-based inference with the **speed** of optimized message-passing.

To allow practitioners to define any probabilistic model, press a button, and get accurate inference results within a matter of seconds.

2. Key observation

General algorithms appear to be solving problems that are harder than they need to be: in most **specific** inference problems, we only need to perform a small subset of all possible computations.



3. The JIT learning approach

1. Initially use highly general algorithms for inference, *e.g.* by computing messages in a message-passing algorithm via sampling.
2. Gradually learn to increase the speed of these computations by regressing from input to output messages at run-time.

4. JIT message computation

Learn a mapping from variable-to-factor messages $\{m_{k\to\psi}\}$ to a factor-to-variable message $m_{\psi\to i}$:

$$\overline{m}_{\psi\to i}(x_i) \equiv f(\{m_{k\to\psi}\}_{k\in\mathrm{ne}(\psi)}|\boldsymbol{\theta}). \tag{1}$$

Along with each prediction, produce a measure $\overline{u}$ of the regressor's uncertainty:

$$\overline{u}_{\psi\to i} \equiv u(\{m_{k\to\psi}\}_{k\in\mathrm{ne}(\psi)}|\boldsymbol{\theta}). \tag{2}$$

Use uncertainties at run-time to choose between regressor estimates and slow 'oracle' computations:

$$m_{\psi\to i}(x_i) = \begin{cases} \overline{m}_{\psi\to i}(x_i) & \overline{u}_{\psi\to i} < u^{\max} \\ m^{\mathrm{oracle}}_{\psi\to i}(x_i) & \text{otherwise} \end{cases} \tag{3}$$

where $u^{\max}$ is the maximum tolerated uncertainty for a prediction. Set $u^{\max}$ such that no held out prediction has an error above a user-specified maximum tolerated value $D^{\max}$.

Oracle messages can be computed using generally-applicable Monte Carlo (Barthelme *et al.*, 2011; Heess *et al.*, 2013), or via hand-implemented operators (*e.g.* those in Infer.NET, Minka *et al.*, 2012).

5. Random forests for JIT learning

Aim. Learn a mapping from incoming messages $\{m_{k\to\psi}\}_{k\in\mathrm{ne}(\psi)}$ to the outgoing message $m_{\psi\to i}$.

Requirements. The regressor must: 1) train and predict efficiently, 2) model arbitrarily complex mappings, 3) adapt dynamically, and 4) produce uncertainty estimates.

Representation. Let m_{out} be the outgoing message and $\mathbf{m}_{\mathrm{in}}$ the set of incoming messages.

- Represent the outgoing message m_{out} by a vector of real valued numbers $\mathbf{r}_{\mathrm{out}}$.
- Each set of incoming messages $\mathbf{m}_{\mathrm{in}}$ is represented in two ways:
  - *Regression parameterization* denoted by $\mathbf{r}_{\mathrm{in}}$ (*e.g.* concatenation of message parameters),
  - *Tree parameterization* denoted by $\mathbf{t}_{\mathrm{in}}$ (*e.g.* moments and ψ evaluated at the mode of $\mathbf{m}_{\mathrm{in}}$).

Split and prediction model. $\mathbf{t}_{\mathrm{in}}$ is used to traverse message sets down to leaves, and $\mathbf{r}_{\mathrm{in}}$ is used by a polynomial regressor to predict $\mathbf{r}_{\mathrm{out}}$:

$$\mathbf{r}_{\mathrm{out}} = \mathbf{W}\cdot\boldsymbol{\phi}^n(\mathbf{r}_{\mathrm{in}}) + \epsilon. \tag{4}$$

Training objective function. At each node j, depending on the subset of the incoming training set $\mathcal{S}_j$ we learn the function that 'best' splits $\mathcal{S}_j$ into the training sets corresponding to each child, $\mathcal{S}_j^{\mathrm{L}}$ and $\mathcal{S}_j^{\mathrm{R}}$, *i.e.* $\boldsymbol{\tau}_j = \mathrm{argmax}_{\boldsymbol{\tau}\in\mathcal{T}_j} I(\mathcal{S}_j,\boldsymbol{\tau})$. The objective function I is:

$$I(\mathcal{S}_j,\boldsymbol{\tau}) = -E(\mathcal{S}_j^{\mathrm{L}},\mathbf{W}^{\mathrm{L}}) - E(\mathcal{S}_j^{\mathrm{R}},\mathbf{W}^{\mathrm{R}}), \tag{5}$$

where $\mathbf{W}^{\mathrm{L}}$ and $\mathbf{W}^{\mathrm{R}}$ are the parameters of the polynomial regression models corresponding to the left and right training sets $\mathcal{S}_j^{\mathrm{L}}$ and $\mathcal{S}_j^{\mathrm{R}}$, and the 'fit residual' E is:

$$E(\mathcal{S},\mathbf{W}) = \frac{1}{2}\sum_{\mathbf{m}_{\mathrm{in}}\in\mathcal{S}} D^{\mathrm{mar}}_{\mathrm{KL}}(\overline{m}^{\mathbf{W}}_{\mathbf{m}_{\mathrm{in}}}\|m^{\mathrm{oracle}}_{\mathbf{m}_{\mathrm{in}}}) + D^{\mathrm{mar}}_{\mathrm{KL}}(m^{\mathrm{oracle}}_{\mathbf{m}_{\mathrm{in}}}\|\overline{m}^{\mathbf{W}}_{\mathbf{m}_{\mathrm{in}}}). \tag{6}$$

This objective function splits the training data at each node in a way that the relationship between the incoming and outgoing messages is well captured by the polynomial regression in each child.

Mean squared error of message parameters is sensitive to parameterization. Instead, calculate the marginals $\overline{b}_i$ and b_i^{oracle} induced on the target variable, and compute their KL:

$$D^{\mathrm{mar}}_{\mathrm{KL}}(\overline{m}_{\psi\to i}\|m^{\mathrm{oracle}}_{\psi\to i}) \equiv D_{\mathrm{KL}}(\overline{b}_i\|b_i^{\mathrm{oracle}}), \tag{7}$$

$D^{\mathrm{mar}}_{\mathrm{KL}}$ is the *marginal KL* and is used throughout the JIT framework, as it encourages the system to focus efforts on the quantity that is ultimately of interest: the accuracy of the posterior marginals.

Ensemble model. Compute the *moment average* $\overline{m}_{\mathrm{out}}$ of the distributions $\{\overline{m}^t_{\mathrm{out}}\}$ by averaging the first few moments of each predicted distribution across trees, and solving for the distribution parameters which match the averaged moments.

The moment average can be interpreted as minimizing an objective function:

$$\overline{m}_{\mathrm{out}} = \operatorname*{argmin}_m U(\{\overline{m}^t_{\mathrm{out}}\}, m), \tag{8}$$

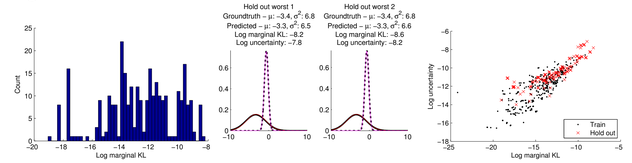
where

$$U(\{\overline{m}^t_{\mathrm{out}}\}, m) = \sum_t D_{\mathrm{KL}}(\overline{m}^t_{\mathrm{out}}\|m). \tag{9}$$

The level of agreement between the predictions of the different trees can be used as a proxy of the forest's uncertainty about that prediction. If all the trees in the forest predict the same output distribution, it means that their knowledge about the function f is similar *despite* the randomness in their structures. We therefore set $\overline{u}_{\mathrm{out}} \equiv U(\{\overline{m}^t_{\mathrm{out}}\}, \overline{m}_{\mathrm{out}})$.

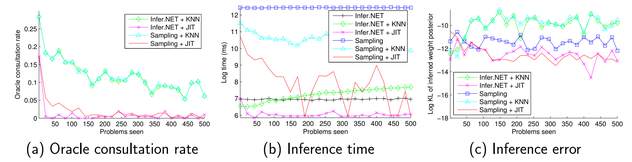
6. Experimental results

Logistic factor



(a) Inference error (b) Worst predicted messages (c) Awareness of uncertainty

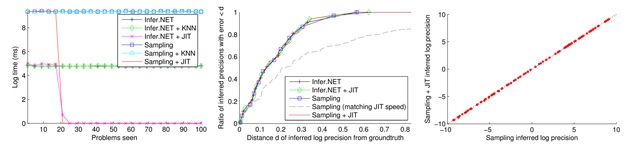
Figure 1: **Uncertainty aware regression.** Learning a logistic factor for use in logistic regression. (a) Histogram of marginal KLs of outgoing messages (Gaussian), which are typically very small. (b) The forest's most inaccurate predictions (black: m^{oracle}, red: $\overline{m}$, dashed black: b^{oracle}, purple: $\overline{b}$). (c) The regressor's uncertainty increases with marginal KL, *i.e.* it does not make confident but inaccurate predictions.



(a) Oracle consultation rate (b) Inference time (c) Inference error

Figure 2: (a) The factor consults the oracle for only a fraction of messages, (b) leading to significant savings in time, (c) whilst maintaining (or decreasing) error.

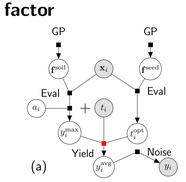
Compound gamma factor



(a) Inference time (b) Inference error (c) Accuracy

Figure 3: (a) JIT reduces inference time for sampling from ∼11 seconds to ∼1 ms. (b) JIT's posteriors agree highly with Infer.NET. Using fewer samples to match JIT speed leads to degradation of accuracy. (c) Speed comes at limited loss of accuracy.

Yield factor



(b)

	IS	JIT fresh		JIT continued	
	Time	FR	Speedup	FR	Speedup
11	451s	54%	195%	—	—
12	449s	54%	192%	60%	288%
13	451s	54%	191%	64%	318%

Figure 4: (a) The yield factor relates temperatures and yields recorded at farms to the optimal temperatures of their planted grain. JIT learning enables us to incorporate arbitrary factors with ease, whilst maintaining inference speed. (b) FR is fraction of regressions with no oracle consultation.