A Variational Inference Toolbox

Is it time for a community-driven VI toolbox?

Yarin Gal [yg279@cam.ac.uk], University of Cambridge

Variational inference is Fragmented

- Most advances in Deep Learning from the last few years are due to central code repositories exploiting model compositionality.
  - Vast number of published papers can be built from simpler building blocks, becoming themselves higher level building blocks.
  - Example: Simple deep networks → Recurrent Neural Networks → Neural Turing Machine → The Neural Queue.
- In contrast, VI has no central repository, or even an agreed-upon framework.
  - Instead we often re-implement existing work in VI, wasting weeks at a time.
  - Is it time for a community driven VI toolbox?

The time is Right

- Relying on recent advances in stochastic inference and sampling based variational inference (replacing integration with stochastic optimisation).
- Taking advantage of frameworks developed within the deep learning community: Theano, Torch, TensorFlow, etc.
- Allows us to design simple VI building blocks to compose together.
- Allows us to combine deep learning and VI seamlessly.

Vanilla variational inference

- Given data $X$, design initial probabilistic model,
  $$ p(x^*|X) = \int p(x^*|\omega)p(\omega|X)\,d\omega $$
  with some latent random variable $\omega$. The posterior $p(\omega|X)$ is intractable.
- Choose an approximating variational distribution $q(\omega)$ matching posterior properties.
- Evaluate divergence between approximating posterior and true posterior obtaining a lower bound,
  $$ \mathcal{L}(\theta) = \int q(\omega) \log p(X|\omega)\,d\omega - KL(q(\omega)||p(\omega)). $$
  And then...
- Spend weeks calculating and implementing derivatives, testing with finite differences, and optimising computations for performance and numerical stability.

We could do better.

If we had modular VI Building Blocks...

- Replace the last two steps in vanilla VI.
- Collect common VI building blocks into a central repository.
- Write down generative model in a symbolic language with existing VI blocks (creating new ones as necessary),
  $$ \text{var } \omega; \\
  f(\omega) = \text{Block}_{1}(\text{Block}_{2}(\text{Block}_{3}(\omega))) \\
  X = f(\omega); $$
- Simulate $T$ samples from the approximating posterior and propagate them down the generative model (forward pass),
  $$ \omega_j \sim q(\omega_j); \\
  X_j = f(\omega_j); $$
- Evaluate the objective with the output of the generative model,
  $$ \mathcal{L}(\theta) = \frac{1}{T} \sum_{t=1}^{T} \log p(X_j) - KL(q(\omega)||p(\omega)). $$
  Symbolically differentiate the objective:
  - evaluate derivatives with the same samples
  - obtaining a noisy but unbiased gradient estimate
  - this is a backward pass,
  - Optimise with a stochastic optimiser.

Example

```
import theano.tensor as T

m = T.matrix('m') # and other variational parameters
s = T.tensor4('s') # these are the generative model’s variables
U = m * T.dot(randn(N, K))
KL_U = T.kl_div(T.dmatrix('U'), m) + m.shape[0]
KL_X = T.kl_div(T.dmatrix('X'), m) + m.shape[0]
KL = KL_U + KL_X

LS_func = theano.function([m], KL)
```

Example Python code using the new pipeline. Here, $m, s, u$, and $L$ are the variational parameters, and the generative model $s$ (the probabilities of the discrete variables) is a function of latents $X, U,$ and $F$. Our objective is $LS$.

Emerging Challenges

- Existing tools lack...
  - good support for many operations used in VI (matrix inverses, matrix determinants, etc.),
  - “tricks-of-the-trade” used in VI to avoid problems of numerical instability and large matrix multiplications,
  - Would these lead to more efficient models, smaller, readable, and extendible code bases?
- Black-box variance reduction
  - Variance reduction forces model re-parametrisation → complicated inference and code,
  - Apply variance reduction automatically to the symbolic graph?
- Model compositionality
  - Speed up the innovation cycle allowing fast-evolving model complexity.
  - What are the basic VI building blocks?
  - Recent work casting deep learning tools as VI in Bayesian neural networks (see other paper) → already have many building blocks to start with!

A unified framework will make VI accessible to larger audiences.