UNIVERSITY OF CAMBRIDGE

Rapid Prototyping of Probabilistic Models using Stochastic Variational Inference

Yarin Gal

yg279@cam.ac.uk

October 1st, 2014

Rapid Prototyping

UNIVERSITY OF CAMBRIDGE

- In **data analysis** we often have to develop new models
- This can be a lengthy process
  - We need to derive appropriate inference
  - Often cumbersome implementation which changes regularly
- **Rapid prototyping** is used to answer similar problems in manufacturing
  - "Quick fabrication of scale models of a physical part"
  - Probabilistic programming can be used for rapid prototyping in machine learning

Rapid Prototyping

- In **data analysis** we often have to develop new models
- This can be a lengthy process
  - We need to derive appropriate inference
  - Often cumbersome implementation which changes regularly
- **Rapid prototyping** is used to answer similar problems in manufacturing
  - "Quick fabrication of scale models of a physical part"
  - Probabilistic programming can be used for rapid prototyping in machine learning

Rapid Prototyping

UNIVERSITY OF CAMBRIDGE

- In **data analysis** we often have to develop new models
- This can be a lengthy process
  - We need to derive appropriate inference
  - Often cumbersome implementation which changes regularly
- **Rapid prototyping** is used to answer similar problems in manufacturing
  - “Quick fabrication of scale models of a physical part”
  - Probabilistic programming can be used for rapid prototyping in machine learning

- In **data analysis** we often have to develop new models
- This can be a lengthy process
  - We need to derive appropriate inference
  - Often cumbersome implementation which changes regularly
- **Rapid prototyping** is used to answer similar problems in manufacturing
  - "Quick fabrication of scale models of a physical part"
  - Probabilistic programming can be used for rapid prototyping in machine learning

Today I'm going to argue that Stochastic Variational Inference (SVI) can be used for rapid prototyping as well, with several advantages over probabilistic programming.

Rapid Prototyping

UNIVERSITY OF CAMBRIDGE

- SVI is not usually considered as means of speeding-up development
- But this new inference technique allows us to **simplify the derivations** for a large class of models
  - With this we can take advantage of **effective symbolic differentiation**
  - Models are often mathematically **too cumbersome otherwise**
- Similar principles have been used for **rapid model prototyping in deep learning** for NLP for quite some time [Socher, Ng, and Manning 2010, 2011, 2012]

What is SVI?

- SVI is simply **variational inference** used with **noisy gradients** – we thus replace the optimisation with stochastic optimisation
- Variational inference
  - We approximate the posterior of the latent variables with distributions from a tractable family ($q(X)$ for example)

Example model: $X \to Y$

$$\log P(Y) \geq \int q(X) \log \frac{P(Y|X)P(X)}{q(X)} = E_q[\log P(Y|X)] - KL(q||P)$$

What is SVI?

- Stochastic variational inference
  - Often used to speed-up inference using **mini-batches**

$$\log P(Y) \geq \frac{N}{|S|} \sum_{i \in S} E_q[\log P(Y_i | X_i)] - KL(q || P)$$

summing over random subsets of the data points

  - But can also be used to **approximate integrals** through Monte Carlo integration [Kingma and Welling 2014, Rezende et al. 2014, Titsias and Lazaro-Gredilla 2014]

$$E_q[\log P(Y|X)] \approx \frac{1}{K} \sum_{i=1}^{K} \log P(Y|X_i), \; X_i \sim q(X)$$

summing over samples from the approximating distribution

  - Optimising these objectives relies on non-deterministic gradients

What is SVI?

- Stochastic variational inference
  - Often used to speed-up inference using **mini-batches**

$$\log P(Y) \geq \frac{N}{|S|} \sum_{i \in S} E_q[\log P(Y_i|X_i)] - KL(q||P)$$

    summing over random subsets of the data points

  - But can also be used to **approximate integrals** through Monte Carlo integration [Kingma and Welling 2014, Rezende et al. 2014, Titsias and Lazaro-Gredilla 2014]

$$E_q[\log P(Y|X)] \approx \frac{1}{K} \sum_{i=1}^{K} \log P(Y|X_i), \ X_i \sim q(X)$$

    summing over samples from the approximating distribution

  - Optimising these objectives relies on non-deterministic gradients

What is SVI?

- Stochastic variational inference
  - Often used to speed-up inference using **mini-batches**

$$\log P(Y) \geq \frac{N}{|S|} \sum_{i \in S} E_q[\log P(Y_i|X_i)] - KL(q||P)$$

    summing over random subsets of the data points

  - But can also be used to **approximate integrals** through Monte Carlo integration [Kingma and Welling 2014, Rezende et al. 2014, Titsias and Lazaro-Gredilla 2014]

$$E_q[\log P(Y|X)] \approx \frac{1}{K} \sum_{i=1}^{K} \log P(Y|X_i), \; X_i \sim q(X)$$

    summing over samples from the approximating distribution

  - Optimising these objectives relies on non-deterministic gradients

Stochastic optimisation

- Using gradient descent with noisy gradients and decreasing learning-rates, we are guaranteed to converge to an optimum

$$\theta_{t+1} = \theta_t + \alpha f'(\theta_t)$$

- Learning-rates (α) are hard to tune...
  - Use learning-rate free optimisation (again, from deep learning)
  - AdaGrad [Duchi et. al 2011], AdaDelta [Zeiler 2012]
  - RMSPROP [Tieleman and Hinton 2012, Lecture 6.5, COURSERA: Neural Networks for Machine Learning]

$$\theta_{t+1} = \theta_t + \frac{\alpha}{\sqrt{r_t}} f'(\theta_t); \; r_t = (1-\gamma) f'(\theta)^2 + \gamma r_{t-1}$$

and increase α times $1+\epsilon$ if the last two grads' directions agree

  - These have been compared to each other and others *empirically* in a variety of settings in [Schaul 2014]

Stochastic optimisation

- Using gradient descent with noisy gradients and decreasing learning-rates, we are guaranteed to converge to an optimum

$$\theta_{t+1} = \theta_t + \alpha f'(\theta_t)$$

- Learning-rates (α) are hard to tune...
  - Use learning-rate free optimisation (again, from deep learning)
  - AdaGrad [Duchi et. al 2011], AdaDelta [Zeiler 2012]
  - RMSPROP [Tieleman and Hinton 2012, Lecture 6.5, COURSERA: Neural Networks for Machine Learning]

$$\theta_{t+1} = \theta_t + \frac{\alpha}{\sqrt{r_t}} f'(\theta_t); \;\; r_t = (1-\gamma) f'(\theta)^2 + \gamma r_{t-1}$$

    and increase α times $1+\epsilon$ if the last two grads' directions agree
  - These have been compared to each other and others *empirically* in a variety of settings in [Schaul 2014]

- Using gradient descent with noisy gradients and decreasing learning-rates, we are guaranteed to converge to an optimum

$$\theta_{t+1} = \theta_t + \alpha f'(\theta_t)$$

- Learning-rates (α) are hard to tune...
  - Use learning-rate free optimisation (again, from deep learning)
  - AdaGrad [Duchi et. al 2011], AdaDelta [Zeiler 2012]
  - RMSPROP [Tieleman and Hinton 2012, Lecture 6.5, COURSERA: Neural Networks for Machine Learning]

$$\theta_{t+1} = \theta_t + \frac{\alpha}{\sqrt{r_t}} f'(\theta_t); \; r_t = (1-\gamma) f'(\theta)^2 + \gamma r_{t-1}$$

    and increase α times $1 + \epsilon$ if the last two grads' directions agree
  - These have been compared to each other and others *empirically* in a variety of settings in [Schaul 2014]

Stochastic optimisation

- Using gradient descent with noisy gradients and decreasing learning-rates, we are guaranteed to converge to an optimum

$$\theta_{t+1} = \theta_t + \alpha f'(\theta_t)$$

- Learning-rates (α) are hard to tune...
  - Use learning-rate free optimisation (again, from deep learning)
  - AdaGrad [Duchi et. al 2011], AdaDelta [Zeiler 2012]
  - RMSPROP [Tieleman and Hinton 2012, Lecture 6.5, COURSERA: Neural Networks for Machine Learning]

$$\theta_{t+1} = \theta_t + \frac{\alpha}{\sqrt{r_t}} f'(\theta_t); \; r_t = (1-\gamma) f'(\theta)^2 + \gamma r_{t-1}$$

    and increase α times $1+\epsilon$ if the last two grads' directions agree

  - These have been compared to each other and others *empirically* in a variety of settings in [Schaul 2014]

Stochastic optimisation

- Using gradient descent with noisy gradients and decreasing learning-rates, we are guaranteed to converge to an optimum

$$\theta_{t+1} = \theta_t + \alpha f'(\theta_t)$$

- Learning-rates (α) are hard to tune...
  - Use learning-rate free optimisation (again, from deep learning)
  - AdaGrad [Duchi et. al 2011], AdaDelta [Zeiler 2012]
  - RMSPROP [Tieleman and Hinton 2012, Lecture 6.5, COURSERA: Neural Networks for Machine Learning]

$$\theta_{t+1} = \theta_t + \frac{\alpha}{\sqrt{r_t}} f'(\theta_t); \; r_t = (1-\gamma) f'(\theta)^2 + \gamma r_{t-1}$$

    and increase α times $1+\epsilon$ if the last two grads' directions agree

  - These have been compared to each other and others *empirically* in a variety of settings in [Schaul 2014]

Rapid Prototyping with SVI

With **Monte Carlo integration** we can greatly simplify model and inference description

Example model: X $\rightarrow$ Y

Lower bound:

1. Simulate $X_i \sim q(X)$ for $i \leq K$
2. Evaluate $P(Y|X_i)$
3. Return $\frac{1}{K}\sum_{i=1}^{K} \log P(Y|X_i) - KL(q||P)$

Objective:

$$q_{opt} = \arg\max_{q(X)} \frac{1}{K}\sum_{i=1}^{K} \log P(Y|X_i) - KL(q||P)$$

UNIVERSITY OF CAMBRIDGE

Example model: X → Y

Objective:

$$q_{opt} = \arg\max_{q(X)} \frac{1}{K} \sum_{i=1}^{K} \log P(Y|X_i) - KL(q||P)$$

Symbolic differentiation is straight-forward in this representation:

$$\frac{\partial}{\partial\theta} \log P(Y|X), \ \frac{\partial}{\partial\theta} KL$$

are easy to compute for a large class of models [Titsias and Lazaro-Gredilla 2014]

Examples: Bayesian logistic regression, variable selection, Gaussian process (GP) hyper-parameter estimation, and more [Titsias and Lazaro-Gredilla 2014]

Example: Bayesian logistic regression

Given dataset with $\mathbf{x}_i \in \mathbb{R}^d$ and $y_i \in \{0, 1\}$ for $n \leq N$, we define

$$P(Y|X, \eta) = \prod_{i=1}^{N} \sigma(y_i \mathbf{x}_i^T \eta)$$

for some vector of weights η with prior $P(\eta) = \mathcal{N}(0, I_d)$.
Define

$$q(\eta|\theta = \{\mu, C\}) = \mathcal{N}(\eta; \mu, CC^T)$$

Symbolically differentiate and optimise wrt

$$\frac{\partial}{\partial\theta} \log\big(\prod_{i=1}^{N} \sigma(y_i \mathbf{x}_i^T \eta)\big), \ \frac{\partial}{\partial\theta} KL$$

UNIVERSITY OF CAMBRIDGE

Non-linear density estimation of categorical data (work in progress with Yutian Chen)

Model (using sparse GP with M inducing inputs / outputs Z and U):

$$X \sim \mathcal{N}(0, I)$$
$$(F_K, U_K) \sim GP(X, Z)$$
$$Y \sim Softmax(F_1, ..., F_K)$$

Approximating distributions: $q(X, F, U) = q(X)q(U)p(F|X, U)$, defining $q(x_n) = \mathcal{N}(m_n, s_n^2)$ and $q(u_k) = \mathcal{N}(\mu_k, CC^T)$

We have (with $\epsilon. \sim \mathcal{N}(0, I)$):

$$x_n = m_n + s_n \epsilon_n$$
$$u_k = \mu_k + C\epsilon_k$$
$$f_{nk} = K_{nM}K_{MM}^{-1}u_k + \sqrt{K_{nn} - K_{nM}K_{MM}^{-1}K_{Mn}}\epsilon_{nk}$$
$$y_n = Softmax(f_{n1}, ..., f_{nK})$$

- Original approach took half a year to develop –
  - Deriving variational inference
  - Researching appropriate bound in the statistics literature
  - Derivations for the model

$$\mathcal{L} = -\sum_{n=1}^{N} \mathrm{KL}(q(\mathbf{x}_n) \| p(\mathbf{x}_n))$$

$$+ \sum_{d=1}^{D} \left\{ -\frac{1}{2} \mathbf{Tr}[(I_K \otimes \mathbf{K}_{d,MM}^{-1})(\mathbf{\Sigma}_d + \hat{\boldsymbol{\mu}}_d \hat{\boldsymbol{\mu}}_d^T)] + \frac{MK}{2} + \frac{1}{2} \log |\mathbf{\Sigma}_d| - \frac{1}{2} \log |\mathbf{K}_{d,MM}| \right\}$$

$$+ \sum_{d=1}^{D} \sum_{n=1}^{N} \left\{ -\frac{1}{2} \mathbf{Tr} \left[\left(\mathbf{A}_d \otimes (\mathbf{K}_{d,MM}^{-1} \langle \mathbf{K}_{d,Mn} \mathbf{K}_{d,nM} \rangle_{\mathbf{x}_n} \mathbf{K}_{d,MM}^{-1}) \right) (\mathbf{\Sigma}_d + \hat{\boldsymbol{\mu}}_d \hat{\boldsymbol{\mu}}_d^T) \right] \right.$$

$$+ [(\mathbf{y}_{nd} + \mathbf{b}_{nd})^T \otimes (\langle \mathbf{K}_{d,nM} \rangle_{\mathbf{x}_n} \mathbf{K}_{d,MM}^{-1})] \hat{\boldsymbol{\mu}}_d - c_{nd} - \frac{1}{2} \mathbf{Tr}[\mathbf{A}_d \langle \mathbf{K}_{d,nn} \rangle_{\mathbf{x}_n}] + \frac{1}{2} \mathbf{Tr}(\mathbf{A}_d) \mathbf{Tr}(\mathbf{K}$$

$$= -\sum_{n=1}^{N} \mathrm{KL}(q(\mathbf{x}_n) \| p(\mathbf{x}_n))$$

$$+ \sum^{D} \left\{ -\frac{1}{2} \mathbf{Tr}[(I_K \otimes \mathbf{K}_{MM}^{-1})(\mathbf{\Sigma} + \hat{\boldsymbol{\mu}} \hat{\boldsymbol{\mu}}^T)] + \frac{MK}{2} + \frac{1}{2} \log |\mathbf{\Sigma}| - \frac{1}{2} \log |\mathbf{K}_{MM}| \right.$$

  - Implementation (hundreds of lines of python code)

- New approach –
  - Derivations took a day
  - Programming took a day (15 lines of Python)

Concrete example

- Original approach took half a year to develop –
  - Deriving variational inference
  - Researching appropriate bound in the statistics literature
  - Derivations for the model
  - Implementation (hundreds of lines of python code)

- New approach –
  - Derivations took a day
  - Programming took a day (15 lines of Python)

```
print 'Building model...'
X = m + T.exp(s) * eps_NQ
U = mu + T.tril(L).dot(eps_MK)

dist_ZZ = T.sum(((T.reshape(Z, (M, 1, Q)) - Z) / ard)**2, 2)
dist_ZX = T.sum(((T.reshape(Z, (M, 1, Q)) - X) / ard)**2, 2)
Kmm = sf2 * T.exp(-dist_ZZ / 2.0)
Kmn = sf2 * T.exp(-dist_ZX / 2.0)
Knn = sf2

KmmInv = sT.matrix_inverse(Kmm)
A = KmmInv.dot(Kmn)
B = Knn - T.sum(Kmn * KmmInv.dot(Kmn), 0)

F = A.T.dot(U) + B[:, None]**0.5 * eps_NK
S = T.nnet.softmax(F)
```

Disadvantages of this approach

- Studying how symbolic differentiation works is important though –
  - Careless implementation can take long to run
  - But careful implementation (together with mini batches) can actually scale well!
- Only suitable when variational inference is; As usual in variational inference depends on the family of approximating distributions
- We can have large variance in the approximate integration
  - Either use more samples (slower to run),
  - Or use variance reduction techniques [Wang, Chen, Smola, and Xing 2013]

Disadvantages of this approach

UNIVERSITY OF CAMBRIDGE

- Studying how symbolic differentiation works is important though –
  - Careless implementation can take long to run
  - But careful implementation (together with mini batches) can actually scale well!
- Only suitable when variational inference is; As usual in variational inference depends on the family of approximating distributions
- We can have large variance in the approximate integration
  - Either use more samples (slower to run),
  - Or use variance reduction techniques [Wang, Chen, Smola, and Xing 2013]

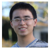
Disadvantages of this approach

UNIVERSITY OF CAMBRIDGE

- Studying how symbolic differentiation works is important though –
  - Careless implementation can take long to run
  - But careful implementation (together with mini batches) can actually scale well!

- Only suitable when variational inference is; As usual in variational inference depends on the family of approximating distributions

- We can have large variance in the approximate integration
  - Either use more samples (slower to run),
  - Or use variance reduction techniques [Wang, Chen, Smola, and Xing 2013]

Also thanks to Yutian Chen , Shakir Mohamed , and Richard Socher