Reparameterization trick: Revolution in stochastic computational graphs

Dmitry P. Vetrov
Research professor in HSE
Leading researcher at Yandex
Head of Bayesian methods research group

http://bayesgroup.ru
Outline

• Stochastic computational graphs
• Latent variable modeling
• Scalable variational inference
• Log-derivative trick and REINFORCE algorithm
• Reparameterization trick
Computational graphs

- Machine learning tries to find dependencies between observed variables $X$ and hidden variables $T$ in data.
- In standard ML algorithms the information is propagated deterministically from input.
- The output is used as prediction for the hidden variable(s).

Parameterized by the weights $\theta$
Automatic differentiation

- When given a **training sample** $\( (X_{tr}, T_{tr}) \) one may try to minimize the error on training sample

- For example in case of **probabilistic model** $\( p(T|X, \theta) \) we compute **maximum likelihood** estimate

\[
\theta_{ML} = \arg \max_{\theta} \ p(T_{tr}|X_{tr}, \theta)
\]

- Automatic differentiation allows us to propagate gradients and compute all derivatives w.r.t. $\theta$ efficiently via chain rule

![Diagram](attachment:image.png)

**Parameterized by the weights $\theta$**
Stochastic computational graphs

- In many popular models we inject random elements in our computational graph
- The ML model becomes more complex but obtains many useful properties
- Switching to stochastic computational graphs is current trend in modern ML

\[ X \xrightarrow{\text{Random variable } \xi} \Theta \xrightarrow{\text{Parameterized by the weights } \theta} T \]
Reinforcement learning

- States $s_t$ are defined as Markov decision process: $s_t \sim p(s_t|s_{t-1}, a_{t-1})$
- Actions are defined by current policy: $a_t \sim \pi(a_t|s_t)$
- Rewards are defined as a function of state and action: $r_t = r(s_t, a_t)$
Variational auto-encoders

- New way for dimension reduction (learning representations)
- High-dimensional data $X$ are generated from low-dimensional latent representations $T$ via deep neural network (decoder) that models $p(X|T, \theta)$
- Inference is made by another deep network (encoder) that returns a distribution $q(T|X, \phi)$ in the space of latent variables rather than a point estimate
Dropout

- Dropout is an heuristic regularization mechanism for deep neural networks
- Some neurons are randomly nullified during training
- It corresponds to training an ensemble of networks

Binary random variables $\xi$

Parameterized by the weights $\theta$
Artistic style

- Artistic style allows to transfer style from one image and apply it to another image
- The content is kept fixed
- Modern methods synthesize the image from scratch

![Diagram showing the process of artistic style transfer](image-url)
Attention mechanism

- New framework for detecting what computations should be made for a particular data
- Attention mechanism is modeled by an additional neural network that returns a distribution on the space of computations
- Surprisingly trainable scheme

\[X \xrightarrow{\text{Attention mechanism}} \text{Attention map} \xrightarrow{\text{Random sample } \xi} T\]

Parameterized by the weights \(\theta\)
Applications of attention

• Image2caption
• Machine translation
• Adaptive computation time
• Q&A systems
Bayesian regularization

- In Bayesian framework the weights $\theta$ are treated as random variables that have a distribution
  \[ p(T, \theta | X) = p(T | X, \theta) p(\theta) \]
- Prior $p(\theta)$ allows to **regularize** the model thus preventing overfitting on training set
- We need to take into account the randomness of $\theta$ during training

\[ X \rightarrow \text{Parameterized by random weights } \theta \rightarrow T \]
Latent variable models

- Consider stochastic computational graph with additional random variables $\xi$

- The probabilistic model turns to $p(T, \xi|X, \theta)$

- If we knew $\xi$ during training we could simply maximize likelihood by standard back-propagation

  $$\theta_{ML} = \arg \max_{\theta} \log p(T_{tr}, \xi|X_{tr}, \theta)$$

- The problem is we do not observe them in training sample $(X_{tr}, T_{tr})$ and hence have to use latent variable model

- In theory we simply need to integrate out $\xi$ and optimize marginal likelihood

  $$\log p(T_{tr}|X_{tr}, \theta) = \log \int p(T_{tr}, \xi|X_{tr}, \theta)d\xi \rightarrow \max$$
Incomplete likelihood decomposition

- The problem is $\log p(T_{tr} | X_{tr}, \theta)$ is generally intractable to compute not saying about optimizing it

- Make use of the following decomposition of log of incomplete likelihood that is true for any choice of $q(\xi)$

$$
\log p(T_{tr} | X_{tr}, \theta) = \int q(\xi) \log \frac{p(T_{tr}, \xi | X_{tr}, \theta)}{q(\xi)} d\xi + \int q(\xi) \log \frac{q(\xi)}{p(\xi | X_{tr}, T_{tr}, \theta)} d\xi
$$
Incomplete likelihood decomposition

• The problem is $\log p(T_{tr}|X_{tr}, \theta)$ is generally intractable to compute not saying about optimizing it

• Make use of the following decomposition of log of incomplete likelihood that is true for any choice of $q(\xi)$

$$
\log p(T_{tr}|X_{tr}, \theta) = \int q(\xi) \log \frac{p(T_{tr}, \xi|X_{tr}, \theta)}{q(\xi)} d\xi + \int q(\xi) \log \frac{q(\xi)}{p(\xi|X_{tr}, T_{tr}, \theta)} d\xi
$$

Variational lower bound  Always non-negative
Incomplete likelihood decomposition

- The problem is \( \log p(T_{tr} | X_{tr}, \theta) \) is generally intractable to compute not saying about optimizing it

- Make use of the following decomposition of log of incomplete likelihood that is true for any choice of \( q(\xi) \)

\[
\log p(T_{tr} | X_{tr}, \theta) = \int q(\xi) \log \frac{p(T_{tr}, \xi | X_{tr}, \theta)}{q(\xi)} d\xi + \int q(\xi) \log \frac{q(\xi)}{p(\xi | X_{tr}, T_{tr}, \theta)} d\xi
\]

Variational lower bound Always non-negative

- Instead of optimizing intractable \( \log p(T_{tr} | X_{tr}, \theta) \) we optimize its variational lower bound

\[
\mathcal{L}(q, \theta) = \int q(\xi) \log \frac{p(T_{tr}, \xi | X_{tr}, \theta)}{q(\xi)} d\xi
\]

with respect to \( q(\xi) \) and \( \theta \)

- Often a family of possible distributions \( q(\xi) \) is limited by some parametric family \( q(\xi) = q(\xi | \phi) \)

- Thus \( \mathcal{L}(q, \theta) \) becomes \( \mathcal{L}(\phi, \theta) \)
Optimizing ELBO

- This variational lower bound plays important role in probabilistic and Bayesian modeling and has special name - Evidence Lower BOund (ELBO)

- Key advantage of ELBO is that we may easily get its unbiased estimates using Monte Carlo methods

\[
\mathcal{L}(\phi, \theta) = \int q(\xi | \phi) \log \frac{p(T_{tr}, \xi | X_{tr}, \theta)}{q(\xi | \phi)} \, d\xi =
\]

\[
\sum_{i=1}^{n} \int q(\xi_i | \phi) \log \frac{p(t_i, \xi_i | x_i, \theta)}{q(\xi_i | \phi)} \, d\xi_i \approx \log \frac{p(t_k, \xi_k | x_k, \theta)}{q(\xi_k | \phi)},
\]

where \( k \sim \mathcal{U}\{1, \ldots, n\} \) and \( \xi_k \sim q(\xi_k | \phi) \)

- Extremely helpful for applying stochastic optimization framework
Stochastic optimization

- Extremely efficient technique for large-scale optimization of $f(x)$
- Uses unbiased estimates $g(x)$ instead of true gradients $\nabla f(x)$
- (Robbins, Monro, 1951) If $f(x)$ is differentiable, $\mathbb{E}g(x) = \nabla f(x)$, $\forall x$, and $\sum_k \alpha_k = +\infty$, $\sum_k \alpha_k^2 < +\infty$, $\alpha_k > 0$ then

$$x_{k+1} = x_k + \alpha_k g(x_k)$$

converges to stationary point of $f(x)$

- Convergence is sublinear (very slow!) and slows down with the increase of $\mathbb{D}g(x)$
Advanced techniques

- Modern stochastic optimization methods (SAG, Adam, SFO, IN2, etc.) use either momentum, memory, or unbiased estimates of Hessian to speed up the convergence.

- Variance reduction techniques (controlled variates, reparametrization, etc.) are also crucial.

- Linear and in some cases even superlinear convergence.
Stochastic gradients

<table>
<thead>
<tr>
<th>Function</th>
<th>Stochastic gradient</th>
</tr>
</thead>
<tbody>
<tr>
<td>$f(x) = \sum_{i=1}^{N} f_i(x)$</td>
<td>$N \nabla f_i(x)$</td>
</tr>
<tr>
<td>$f(x) = \mathbb{E}_y h(x,y) = \int p(y)h(x,y)dy$</td>
<td>( \frac{\partial}{\partial x} h(x,y_0), \ y_0 \sim p(y) )</td>
</tr>
<tr>
<td>$f(x) = \sum_{i=1}^{N} \mathbb{E}<em>y h(x_i, y) = \sum</em>{i=1}^{N} \int p(y)h_i(x,y)dy$</td>
<td>$N \frac{\partial}{\partial x} h_i(x,y_0), \ y_0 \sim p(y) )</td>
</tr>
<tr>
<td>$f(x) = \mathbb{E}_{y</td>
<td>x} h(x,y) = \int p(y</td>
</tr>
</tbody>
</table>

Last example has extremely large variance! 
Variance reduction is needed
Log-derivative trick

Consider differentiation of $\mathbb{E}_{y|x} h(x, y)$ in details

$$\frac{\partial}{\partial x} \int p(y|x) h(x, y) dy = \int \frac{\partial}{\partial x} (p(y|x) h(x, y)) dy =$$

$$\int \left( h(x, y) \frac{\partial}{\partial x} p(y|x) + p(y|x) \frac{\partial}{\partial x} h(x, y) \right) dy =$$

$$\int p(y|x) \frac{\partial}{\partial x} h(x, y) dy + \int h(x, y) \frac{\partial}{\partial x} p(y|x) dy$$
Log-derivative trick

Consider differentiation of $\mathbb{E}_{y|x} h(x, y)$ in details

$$\frac{\partial}{\partial x} \int p(y|x) h(x, y) \,dy = \int \frac{\partial}{\partial x} (p(y|x) h(x, y)) \,dy =$$

$$\int \left( h(x, y) \frac{\partial}{\partial x} p(y|x) + p(y|x) \frac{\partial}{\partial x} h(x, y) \right) \,dy =$$

$$\int p(y|x) \frac{\partial}{\partial x} h(x, y) \,dy + \int h(x, y) \frac{\partial}{\partial x} p(y|x) \,dy$$

The first term is ok since it can be replaced with a Monte Carlo estimate of expectation. To deal with second term we need to use log-derivative trick

$$\frac{\partial}{\partial x} p(y|x) = p(y|x) \frac{\partial}{\partial x} \log p(y|x)$$
Log-derivative trick

Consider differentiation of $\mathbb{E}_{y|x} h(x, y)$ in details

$$\frac{\partial}{\partial x} \int p(y|x) h(x, y) dy = \int \frac{\partial}{\partial x} (p(y|x) h(x, y)) dy =$$

$$\int \left( h(x, y) \frac{\partial}{\partial x} p(y|x) + p(y|x) \frac{\partial}{\partial x} h(x, y) \right) dy =$$

$$\int p(y|x) \frac{\partial}{\partial x} h(x, y) dy + \int h(x, y) \frac{\partial}{\partial x} p(y|x) dy$$

The first term is ok since it can be replaced with a Monte Carlo estimate of expectation. To deal with second term we need to use log-derivative trick

$$\frac{\partial}{\partial x} p(y|x) = p(y|x) \frac{\partial}{\partial x} \log p(y|x)$$

Applying the trick yields to

$$\frac{\partial}{\partial x} \int p(y|x) h(x, y) dy = \int p(y|x) \frac{\partial}{\partial x} h(x, y) dy + \int p(y|x) h(x, y) \frac{\partial}{\partial x} \log p(y|x) dy \approx$$

$$\frac{\partial}{\partial x} h(x, y_0) + h(x, y_0) \frac{\partial}{\partial x} \log p(y_0|x)$$
Log-derivative trick for ELBO

$$\frac{\partial}{\partial x} \int p(y|x)h(x, y)dy \approx \frac{\partial}{\partial x} h(x, y_0) + h(x, y_0) \frac{\partial}{\partial x} \log p(y_0|x)$$

- Now remember our ELBO

$$\mathcal{L}(\phi, \theta) = \int q(\xi|\phi) \log \frac{p(T_{tr}, \xi|X_{tr}, \theta)}{q(\xi|\phi)} d\xi$$

- Differentiation w.r.t. $\theta$ is okay

- Consider differentiation w.r.t. $\phi$ and for simplicity split integral into two parts and take the first term

$$\frac{\partial}{\partial \phi} \int q(\xi|\phi) \log p(T_{tr}, \xi|X_{tr}, \theta) d\xi \approx \log p(t_k, \xi_k|x_k, \theta) \frac{\partial}{\partial \phi} \log q(\xi|\phi),$$

where $k \sim \mathcal{U}\{1, \ldots, n\}$ and $\xi_k \sim q(\xi_k|\phi)$
Log-derivative trick for ELBO

\[
\frac{\partial}{\partial x} \int p(y|x)h(x, y)dy \approx \frac{\partial}{\partial x} h(x, y_0) + h(x, y_0) \frac{\partial}{\partial x} \log p(y_0|x)
\]

• Now remember our ELBO

\[
\mathcal{L}(\phi, \theta) = \int q(\xi|\phi) \log \frac{p(T_{tr}, \xi|X_{tr}, \theta)}{q(\xi|\phi)} d\xi
\]

• Differentiation w.r.t. \( \theta \) is okay

• Consider differentiation w.r.t. \( \phi \) and for simplicity split integral into two parts and take the first term

\[
\frac{\partial}{\partial \phi} \int q(\xi|\phi) \log p(T_{tr}, \xi|X_{tr}, \theta) d\xi \approx \log p(t_k, \xi_k|x_k, \theta) \frac{\partial}{\partial \phi} \log q(\xi|\phi),
\]

where \( k \sim \mathcal{U}\{1, \ldots, n\} \) and \( \xi_k \sim q(\xi_k|\phi) \)

ACHTUNG!
REINFORCE

\[ \frac{\partial}{\partial \phi} \int q(\xi | \phi) \log p(T_{tr}, \xi | X_{tr}, \theta) d\xi \approx \log p(t_k, \xi_k | x_k, \theta) \frac{\partial}{\partial \phi} \log q(\xi | \phi) \]

- Term \( \log p(t_k, \xi_k | x_k, \theta) \) can be arbitrary large negative that leads to very unstable stochastic gradients

- A partial solution is to use **baselines**

- Consider a function \( B(\xi, \phi) \) such that

\[
\int q(\xi | \phi) B(\xi, \phi) d\xi = 0
\]

- A good example is

\[
B(\xi, \phi) = \frac{\partial}{\partial \phi} \log q(\xi | \phi)
\]

the so-called **score function**
REINFORCE

Then

\[
\frac{\partial}{\partial \phi} \int q(\xi|\phi) \log p(T_{tr}, \xi|X_{tr}, \theta) d\xi = \frac{\partial}{\partial \phi} \int q(\xi|\phi)(\log p(T_{tr}, \xi|X_{tr}, \theta) - B(\xi, \phi)) d\xi \approx \\
(\log p(t_k, \xi_k|x_k, \theta) - B(\xi_k, \phi)) \frac{\partial}{\partial \phi} \log q(\xi|\phi),
\]

where \( k \sim \mathcal{U}\{1, \ldots, n\} \) and \( \xi_k \sim q(\xi|\phi) \)

- We can make baseline data-dependent \( B(\xi, \phi, x) \)

- If baseline correlates with \( \log p(t, \xi|x, \theta) \) then it may reduce the variance of stochastic gradient

- Unfortunately still impractical for high-dimensional \( \xi \)
Reparameterization trick

- Consider differentiation of complex expectation

\[ \frac{\partial}{\partial x} \int p(y|x)h(x, y)dy \]

- Express \( y \) as a deterministic function \( g(.) \) of random \( \epsilon \) and \( x \) and perform change-of-variables rule

\[ \int p(y|x)h(x, y)dx = \int p(\epsilon)h(x, g(\epsilon, x))d\epsilon \]

- Then stochastic differentiation is simply

\[ \frac{\partial}{\partial x} \int p(y|x)h(x, y)dx = \frac{\partial}{\partial y} \int p(\epsilon)h(x, g(\epsilon, x))d\epsilon \approx \frac{d}{dx} h(x, g(x, \hat{\epsilon})), \]

where \( \hat{\epsilon} \sim p(\epsilon) \)

- Such reparameterization trick allows to reduce variance of stochastic gradient by orders of magnitude!
Examples of reparameterization

| $p(x|y)$ | $p(\epsilon)$ | $g(\epsilon, y)$ |
|----------|----------------|------------------|
| $\mathcal{N}(x|\mu, \sigma^2)$ | $\mathcal{N}(\epsilon|0, 1)$ | $x = \sigma \epsilon + \mu$ |
| $\mathcal{G}(x|1, \beta)$ | $\mathcal{G}(\epsilon|1, 1)$ | $x = \beta \epsilon$ |
| $\mathcal{E}(x|\lambda)$ | $\mathcal{U}(\epsilon|0, 1)$ | $x = -\frac{\log \epsilon}{\lambda}$ |
| $\mathcal{N}(x|\mu, \Sigma)$ | $\mathcal{N}(\epsilon|0, I)$ | $x = A\epsilon + \mu$, where $AA^T = \Sigma$ |

- Not all continuous distributions can be effectively reparameterized
- Discrete distributions **cannot** be reparameterized
Reparameterization trick for ELBO

- Return to our ELBO

\[ \mathcal{L}(\phi, \theta) = \int q(\xi|\phi) \log \frac{p(T_{tr}, \xi|X_{tr}, \theta)}{q(\xi|\phi)} d\xi \]

- It is easy to get stochastic gradient w.r.t. \( \theta \)

\[ \frac{\partial}{\partial \theta} \mathcal{L}(\phi, \theta) \approx \frac{\partial}{\partial \theta} \log p(t_k, \hat{\xi}|x_k, \theta), \]

where \( k \sim \mathcal{U}\{1, \ldots, N\} \) and \( \hat{\xi} \sim q(\xi|\phi) \)
Reparameterization trick for ELBO

- Return to our ELBO

\[
\mathcal{L}(\phi, \theta) = \int q(\xi|\phi) \log \frac{p(T_{tr}, \xi|X_{tr}, \theta)}{q(\xi|\phi)} d\xi
\]

- It is easy to get stochastic gradient w.r.t. \( \theta \)

\[
\frac{\partial}{\partial \theta} \mathcal{L}(\phi, \theta) \approx \frac{\partial}{\partial \theta} \log p(t_k, \hat{\xi}|x_k, \theta),
\]

where \( k \sim \mathcal{U}\{1, \ldots, N\} \) and \( \hat{\xi} \sim q(\xi|\phi) \)

- Now perform reparameterization trick \( \xi = g(\epsilon, \phi) \)

- Then

\[
\frac{\partial}{\partial \phi} \mathcal{L}(\phi, \theta) \approx \frac{\partial}{\partial \phi} \log p(t_k, g(\hat{\epsilon}, \phi)|x_k, \theta) - \frac{\partial}{\partial \phi} \log q(g(\hat{\epsilon}, \phi)|\phi),
\]

where \( k \sim \mathcal{U}\{1, \ldots, N\} \) and \( \hat{\epsilon} \sim p(\epsilon) \)
Gradient propagation in CG

- Any computational graph defines **superposition** of functions to be computed.

- Remember how we propagate the gradients in computational graphs $t = f(h(g(x)))$.

- Using chain rule we have

  $\frac{\partial t}{\partial x} = \frac{\partial f(h)}{\partial h} \frac{\partial h(g)}{\partial g} \frac{\partial g(x)}{\partial x}$

- Also known as **back-propagation**.
Gradient propagation in Stochastic CG

- Now the situation has changed

\[ t = \mathbb{E} f(\xi) = \int p(\xi|g(x)) f(\xi) d\xi \]

- Using either log-derivative or reparameterization tricks we are still able to propagate the (stochastic) gradients
Gradient propagation in Stochastic CG

\[ g(\cdot) \quad p(\xi | g(x)) \quad f(\cdot) \]

- Now the situation has changed

\[ t = \mathbb{E} f(\xi) = \int p(\xi | g(x)) f(\xi) d\xi \]

- Log-derivative trick

\[
\frac{\partial t}{\partial x} = \frac{\partial}{\partial x} \int p(\xi | g(x)) f(\xi) d\xi = \int \frac{\partial p(\xi | g(x))}{\partial x} f(\xi) d\xi = \int p(\xi | g(x)) \frac{\partial \log p(\xi | g(x))}{\partial x} f(\xi) d\xi \approx \frac{\partial \log p(\hat{\xi} | g(x))}{\partial x} f(\hat{\xi}) = \frac{\partial \log p(\hat{\xi} | g(x))}{\partial g} \frac{\partial g(x)}{\partial x} f(\hat{\xi}),
\]

where \( \hat{\xi} \sim p(\xi | g(x)) \)
Gradient propagation in Stochastic CG

- Now the situation has changed

\[ t = \mathbb{E} f(\xi) = \int p(\xi | g(x)) f(\xi) d\xi \]

- Reparameterization trick. Let \( \xi = h(\epsilon, g(x)) \). Then

\[
\frac{\partial t}{\partial x} = \frac{\partial}{\partial x} \int p(\xi | g(x)) f(\xi) d\xi = \frac{\partial}{\partial x} \int p(\epsilon) f(h(\epsilon, g(x))) d\epsilon = \int p(\epsilon) \frac{\partial f(h(\epsilon, g(x)))}{\partial x} d\epsilon \approx \frac{\partial f(h(\hat{\epsilon}, g(x)))}{\partial x} = \frac{\partial f(h)}{\partial h} \frac{\partial h(g)}{\partial g} \frac{\partial g(x)}{\partial x},
\]

where \( \hat{\epsilon} \sim p(\epsilon) \)
Gradient propagation in Stochastic CG

\[ t = \mathbb{E} f(\xi) = \int p(\xi | g(x)) f(\xi) d\xi \]

- Now the situation has changed

\[ \frac{\partial t}{\partial x} = \frac{\partial}{\partial x} \int p(\xi | g(x)) f(\xi) d\xi = \frac{\partial}{\partial x} \int p(\epsilon) f(h(\epsilon, g(x))) d\epsilon = \int p(\epsilon) \frac{\partial f(h(\epsilon, g(x)))}{\partial x} d\epsilon \approx \frac{\partial f(h(\hat{\epsilon}, g(x)))}{\partial x} = \frac{\partial f(h)}{\partial h} \frac{\partial h(g)}{\partial g} \frac{\partial g(x)}{\partial x}, \]

where \( \hat{\epsilon} \sim p(\epsilon) \)

- Note that unlike previous slide here we use the information about gradient of \( f(\cdot) \)
Conclusion

- Stochastic computational graphs allow much more flexibility when modeling dependencies within the data
- Injecting simple stochasticity at middle levels yields to complex distributions at the end
- A way of performing Bayesian inference in complicated probabilistic models
- Most promising mean for generative models
- May process and generate “raw data” (texts, images, videos, graphs, etc.)
- OPEN PROBLEM: How to deal with discrete latent random variables?
Summer school on NeuroBayes

• We are planning to organize a summer school on neurobayesian methods in Aug, 2017
• We will show how deep learning can be combined with probabilistic modeling
• Visit our website http://DeepBayes.ru