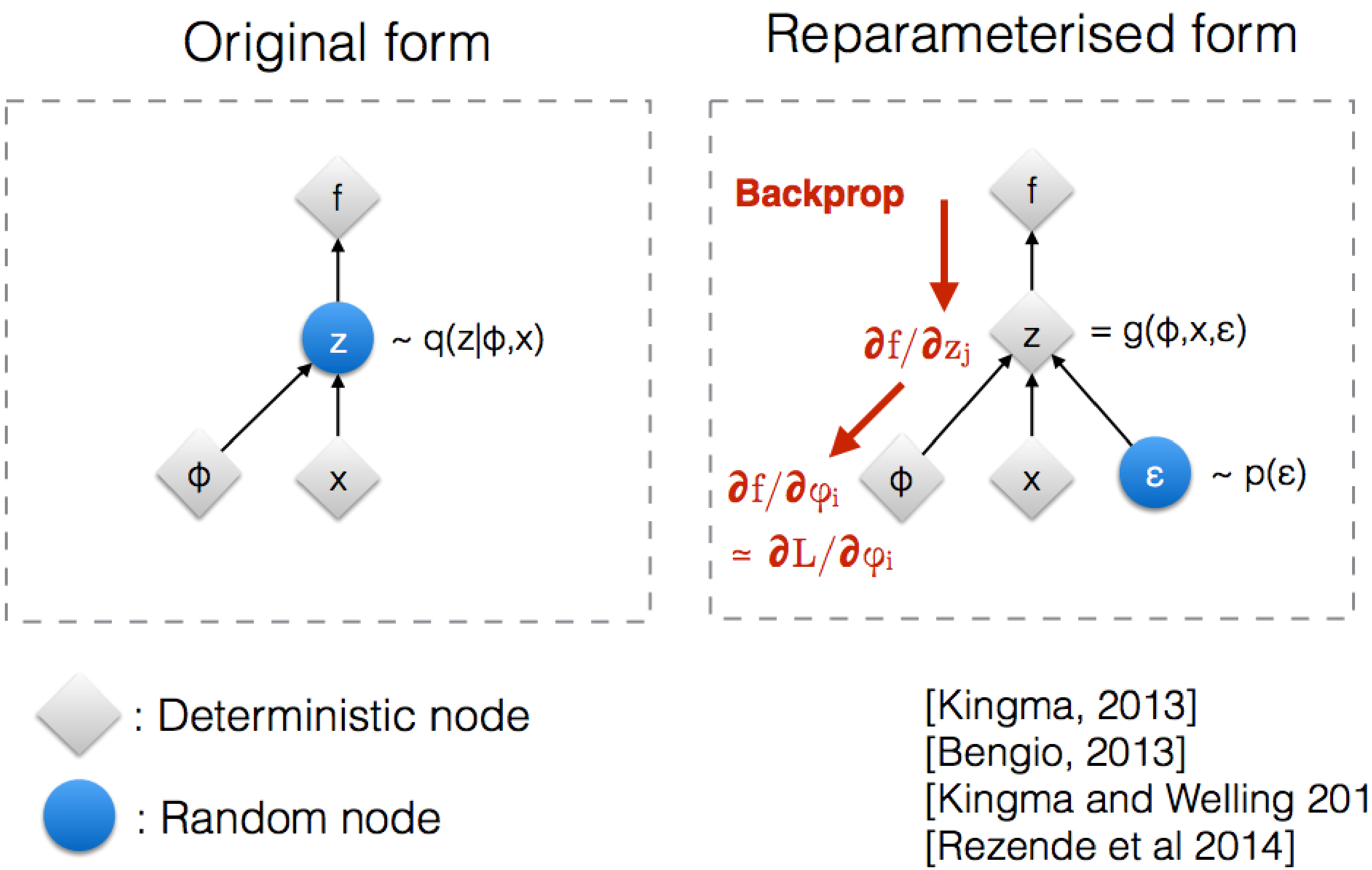
Reparameterization trick



[Kingma, 2013]
[Bengio, 2013]
[Kingma and Welling 2014]
[Rezende et al 2014]

Backpropagating VAE parameters $\boldsymbol{\varphi}, \boldsymbol{\theta}$

- Backpropagation → compute the gradients with respect to θ and φ

$$\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\varphi}) = \mathbb{E}_{\boldsymbol{z} \sim q_{\boldsymbol{\varphi}}(\boldsymbol{z}|\boldsymbol{x})}[\log p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})] - \mathrm{KL}(q_{\boldsymbol{\varphi}}(\boldsymbol{z}|\boldsymbol{x})||p(\boldsymbol{z}))$$

Backpropagating w.r.t. $\boldsymbol{\theta}$

- Backpropagation → compute the gradients with respect to θ and φ

$$\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\varphi}) = \mathbb{E}_{\boldsymbol{z}\sim q_{\boldsymbol{\varphi}}(\boldsymbol{z}|\boldsymbol{x})}[\log p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})] - \mathrm{KL}(q_{\boldsymbol{\varphi}}(\boldsymbol{z}|\boldsymbol{x})||p(\boldsymbol{z}))$$

- The expectation and sampling in $\mathbb{E}_{\boldsymbol{z}\sim q_{\boldsymbol{\varphi}}(\boldsymbol{z}|\boldsymbol{x})}$ do not depend on $\boldsymbol{\theta}$
  - → The gradient goes inside the expectation

$$\nabla_{\boldsymbol{\theta}}\mathcal{L} = \mathbb{E}_{\boldsymbol{z}\sim q_{\boldsymbol{\varphi}}(\boldsymbol{z}|\boldsymbol{x})}[\nabla_{\boldsymbol{\theta}} \log p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})]$$

- Also, the KL does not depend on $\boldsymbol{\theta}$, so no gradient from over there!
- Just Monte-Carlo integration with samples $\boldsymbol{z}$ drawn from $q_{\boldsymbol{\varphi}}(\boldsymbol{z}|\boldsymbol{x})$

Backpropagating w.r.t. $\boldsymbol{\varphi}$

$$\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\varphi}) = \mathbb{E}_{\boldsymbol{z} \sim q_{\boldsymbol{\varphi}}(\boldsymbol{z}|\boldsymbol{x})}[\log p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})] - \mathrm{KL}(q_{\boldsymbol{\varphi}}(\boldsymbol{z}|\boldsymbol{x}) \parallel p(\mathbf{z}))$$

- The sampling $\boldsymbol{z} \sim q_{\boldsymbol{\varphi}}(\boldsymbol{z}|\boldsymbol{x})$ depends on the parameters φ
  - And, sampling is not a differentiable operation
  - → No gradients
- Monte Carlo not even possible

$$\nabla_{\boldsymbol{\varphi}} \mathbb{E}_{\boldsymbol{z} \sim q_{\boldsymbol{\varphi}}(\boldsymbol{z}|\boldsymbol{x})}[\log p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})] = \int_{\boldsymbol{z}} \nabla_{\boldsymbol{\varphi}}[q_{\boldsymbol{\varphi}}(\boldsymbol{z}|\boldsymbol{x})] \log p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z}) d\boldsymbol{z}$$

  - → no density to sample from
  - $\nabla_{\varphi}[q_{\varphi}(z|x)]$ is the gradient of a density function
  - $\log p_{\theta}(x|z)$ is the logarithm of a density function
- How to turn the expression into Monte Carlo friendly?

Reparameterization trick

- Remember, we have a Gaussian output $z \sim N(\mu_Z, \sigma_Z)$
- For certain pdfs, including the Gaussian, we can rewrite their random variable $\boldsymbol{z}$ as deterministic transformations of an auxiliary and simpler random variable ε

$$\boldsymbol{z} \sim N(\boldsymbol{z}; \boldsymbol{\mu}_{\boldsymbol{z}}, \boldsymbol{\sigma}_{\boldsymbol{z}}) \Leftrightarrow \boldsymbol{z} = \boldsymbol{\mu} + \varepsilon \cdot \boldsymbol{\sigma}, \qquad \varepsilon \sim N(0, 1)$$

  - Because of change of variables: $q(z)dz = q(\varepsilon)d\varepsilon$
  - And, ε is an 'external' random variable
- Remember: $\boldsymbol{\mu}_{\boldsymbol{z}}, \boldsymbol{\sigma}_{\boldsymbol{z}}$ are deterministic (not random) values
  - The outputs of the encoder neural networks

What do we gain?

- We can rewrite our gradient

$$\nabla_{\boldsymbol{\varphi}} \mathbb{E}_{\mathbf{z}\sim q_{\varphi}(\mathbf{z}|\mathbf{x})}[\log p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})] = \nabla_{\boldsymbol{\varphi}} \int_{\boldsymbol{z}} \log p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z}) q_{\boldsymbol{\varphi}}(\boldsymbol{z}|\boldsymbol{x}) d\boldsymbol{z}$$

$$= \nabla_{\boldsymbol{\varphi}} \int_{\varepsilon} \log p_{\boldsymbol{\theta}}\left(\boldsymbol{x}\middle|\boldsymbol{\mu}_{\mathbf{z},\boldsymbol{\varphi}}, \boldsymbol{\sigma}_{\mathbf{z},\boldsymbol{\varphi}}, \varepsilon\right) q(\varepsilon) d\varepsilon$$

$$= \int_{\varepsilon} \nabla_{\boldsymbol{\varphi}} \log p_{\boldsymbol{\theta}}\left(\boldsymbol{x}\middle|\boldsymbol{\mu}_{\mathbf{z},\boldsymbol{\varphi}}, \boldsymbol{\sigma}_{\mathbf{z},\boldsymbol{\varphi}}, \varepsilon\right) q(\varepsilon) d\varepsilon$$

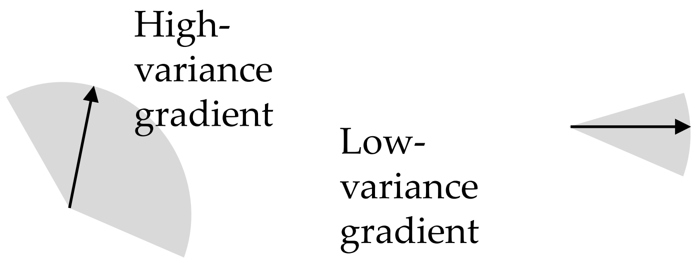
$$\approx \sum_{k} \nabla_{\boldsymbol{\varphi}} \log p_{\boldsymbol{\theta}}\left(\boldsymbol{x}\middle|\boldsymbol{\mu}_{\mathbf{z},\boldsymbol{\varphi}}, \boldsymbol{\sigma}_{\mathbf{z},\boldsymbol{\varphi}}, \varepsilon_k\right), \varepsilon_k \sim N(0,1)$$

Where $\boldsymbol{\varphi}$ are the parameters of the encoder networks $\boldsymbol{\mu}_{\mathbf{z}}, \boldsymbol{\sigma}_{\mathbf{z}}$

- The sampling in MC integration does not depend on $\boldsymbol{\varphi}$ anymore

Low variance estimator

- Sampling directly from $\varepsilon \sim N(0,1)$ leads to low-variance estimates compared to sampling directly from $z \sim N(\mu_Z, \sigma_Z)$
- Remember: we are sampling for z → we are also sampling gradients
  - Stochastic gradient estimator
- More distributions beyond Gaussian possible
  - Laplace, Student-t, Logistic, Cauchy, Rayleight, Pareto



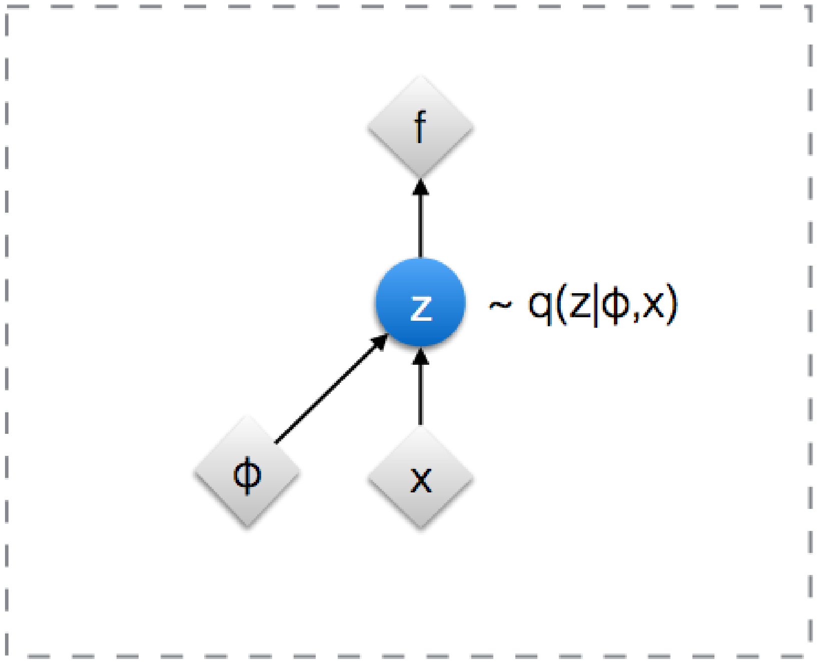
http://blog.shakirm.com/2015/10/machine-learning-trick-of-the-day-4-reparameterisation-tricks/

What exactly happened?

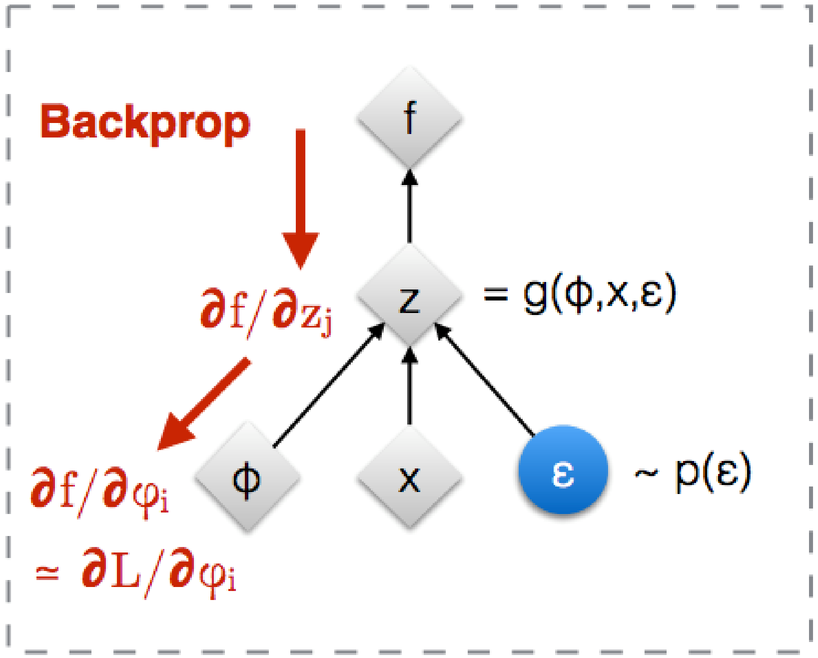
- Again, the latent variable is $\boldsymbol{z} = \boldsymbol{\mu_z} + \varepsilon \cdot \boldsymbol{\sigma_z}$
- $\boldsymbol{\mu_z}$ and $\boldsymbol{\sigma_z}$ are deterministic functions (the neural networks)
- ε is a random variable, which comes **externally**
  - The $\boldsymbol{z}$ as a result is itself a random variable, because of ε
- However, now the randomness is not associated with the neural network and its parameters that we have to learn
  - The randomness instead comes from the external ε
  - The gradients flow through $\boldsymbol{\mu_z}$ and $\boldsymbol{\sigma_z}$

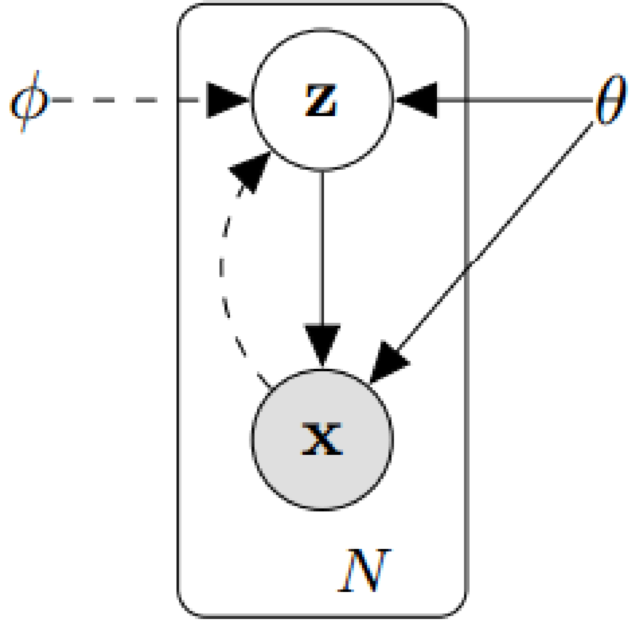
Reparameterization Trick (graphically)

Original form



Reparameterised form





: Deterministic node

: Random node

[Kingma, 2013]
[Bengio, 2013]
[Kingma and Welling 2014]
[Rezende et al 2014]

VAE Training Pseudocode

Data:

$\mathcal{D}$: Dataset

$q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$: Inference model

$p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z})$: Generative model

Result:

$\boldsymbol{\theta}, \boldsymbol{\phi}$: Learned parameters

$(\boldsymbol{\theta}, \boldsymbol{\phi}) \leftarrow$ Initialize parameters

while *SGD not converged* **do**

$\mathcal{M} \sim \mathcal{D}$ (Random minibatch of data)

$\boldsymbol{\epsilon} \sim p(\boldsymbol{\epsilon})$ (Random noise for every datapoint in $\mathcal{M}$)

Compute $\tilde{\mathcal{L}}_{\boldsymbol{\theta},\boldsymbol{\phi}}(\mathcal{M}, \boldsymbol{\epsilon})$ and its gradients $\nabla_{\boldsymbol{\theta},\boldsymbol{\phi}} \tilde{\mathcal{L}}_{\boldsymbol{\theta},\boldsymbol{\phi}}(\mathcal{M}, \boldsymbol{\epsilon})$

Update $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$ using SGD optimizer

end

The ELBO's gradients

Summary

- Latent variable models
- Autoencoders
- Variational inference
- Variational autoencoders
- Reparameterization trick

Reading material:

- All papers mentioned in the slides