Reparametrization Trick

Have a question? Ask or enter a search term here.

FAQ (.) / Reparametrization Trick

Backpropagating through continuous and discrete samples

Keywords: reparametrization trick, Gumbel max trick, Gumbel softmax, Concrete distribution, score function estimator, REINFORCE

Motivation

In the context of deep learning, we often want to **backpropagate a gradient through samples** $x \sim p_\theta(x)$, where $p_\theta(x)$ is a learned parametric distribution.

For example we might want to **train a variational autoencoder**. Conditioned on the input x, the latent representation is not a single value but a distribution $q_\phi(z|x)$, generally a Gaussian distribution $q_\phi(z|x) = \mathcal{N}(\mu_\phi(x), \Sigma_\phi(x))(z)$ which parameters are given by a (inference) neural network of parameters ϕ. When learning to maximize the likelihood of the data, we need to backpropagate the loss to the parameters ϕ of the inference network, across the distribution of z or across samples $z^s \sim p(z|x)$.

TODO: talk about REINFORCE

Goal

More specifically, we want to **minimize an expected cost**
$L(\theta, \phi) = \mathbf{E}_{x \sim p_\phi(x)}[f_\theta(x)]$
using gradient descent, which requires to compute the gradients $\nabla_\theta L(\theta, \phi)$ and $\nabla_\phi L(\theta, \phi)$.

Computing $\nabla_\theta \mathbf{E}_{x \sim p_\phi(x)}[f_\theta(x)]$

Under certain conditions, Leibniz's rule states that the gradient and expectation can be swapped, resulting in

$$\nabla_\theta L(\theta, \phi) = \nabla_\theta \mathbf{E}_{x \sim p_\phi(x)}[f_\theta(x)] = \mathbf{E}_{x \sim p_\phi(x)}[\nabla_\theta f_\theta(x)]$$

which **can be estimated using Monte-Carlo**:

$$\nabla_\theta L(\theta, \phi) \approx \frac{1}{|S|} \sum_{s=1}^{S} \nabla_\theta f_\theta(x^{(s)})$$

with iid samples $x^{(s)} \sim p_\theta(x)$.

So computing $\nabla_\theta \mathbf{E}_{x \sim p_\phi(x)}[f_\theta(x)]$ is fairly straightforward and requires only that:

- we can sample from $p_\theta(x)$
- f is differentiable w.r.t. θ

Computing $\nabla_\phi \mathbf{E}_{x \sim p_\phi(x)}[f_\theta(x)]$

Computing this gradient is much harder because ϕ parametrizes the expectation. Naturally we can rewrite the expectation as an integral over x, and use Leibniz's rule again

$$\nabla_\phi L(\theta, \phi) = \nabla_\phi \mathbf{E}_{x \sim p_\phi(x)}[f_\theta(x)] = \nabla_\phi \int_x f_\theta(x) p_\phi(x) dx$$
$$\nabla_\phi L(\theta, \phi) = \int_x f_\theta(x) \nabla_\phi p_\phi(x) dx$$

but now the integral **does not have the form of an expectation**, so we cannot use Monte-Carlo to estimate its value.

So computing $\nabla_\phi \mathbf{E}_{x\sim p_\phi(x)}[f_\theta(x)]$ is not straighforward. However notice that:

- we only need that the distribution $p_\phi(x)$ is differentiable w.r.t. ϕ
- there is not requirement that $f_\theta(x)$ be differentiable w.r.t x -- no need to backprop through it

> In the rest of the article we review a bunch of different tricks to compute the expectation $\nabla_\phi \mathbf{E}_{x\sim p_\phi(x)}[f_\theta(x)]$ depending on the particular application.

All methods

The table below sums up some ways to deal with samples in a computation graph. Everything in **bold** is either more powerful or less constraining. In the context of deep learning, the most important attributes are that the loss is differentiable w.r.t. ϕ, so that the parameters ϕ can be learned using gradient descent.

Method	**Continuous or Discrete**	**Backpropable Differentiable w.r.t ϕ**	**Follow exact distribution $p(x)$**	**$\frac{\partial f_\theta(x)}{\partial x}$ must exist**
Score function estimator	Continuous and discrete	**Yes**	Yes	**No**
Reparameterization trick	Continuous	**Yes**	Yes	Yes
Gumbel-max trick	Discrete	No	**Yes**	
Gumbel-softmax trick	Discrete	**Yes**	No (continuous relaxation)	Yes
ST-Gumbel esimator	Discrete	**Yes**	**Yes** on forward pass No on backward pass (continuous relaxation)	Yes
REBAR	Discrete	**Yes**	**Yes** on forward pass No on backward pass (continuous relaxation)	?

Score function estimator (trick)

The **score function estimator** (SF), also called **REINFORCE** when applied to reinforcement learning, and **likelihood-ratio estimator** transforms the integral into an expectation.

More specifically, using the property that $\nabla_\phi \log p_\phi(x) = \frac{\nabla_\phi p_\phi(x)}{p_\phi(x)}$ we can rewrite the gradient as an expectation

$$\nabla_\phi L(\theta,\phi) = \int_x f_\theta(x)\nabla_\phi p_\phi(x)dx = \int_x f_\theta(x)\nabla_\phi \log p_\phi(x) p_\phi(x)dx$$
$$\nabla_\phi L(\theta,\phi) = \mathbf{E}_{x\sim p_\phi(x)}[f_\theta(x)\nabla_\phi \log p_\phi(x)]$$

We can now use Monte-Carlo to estimate the gradient.

This estimator has been shown to have issues such as high variance. This problem can be alleviated by subtracting a **control variate** or **baseline** $b(x)$ to $f_\theta(x)$ and adding its mean $\mu_b = \mathbf{E}_{x\sim p_\phi}[b(x)]$ back:

$$\nabla_\phi L(\theta,\phi) = \mathbf{E}_{x\sim p_\phi(x)}[(f_\theta(x) - b(x))\nabla_\phi \log p_\phi(x)] + \mu_b$$

Applications:

- Extreme value theory
- Reinforcement learning (known as REINFORCE)

Reparameterization trick

Sometimes the random variable $x \sim p_\phi(x)$ can be **reparameterized** as a deterministic function g of ϕ and of a random variable $\epsilon \sim p(\epsilon)$, where $p(\epsilon)$ does not depend on ϕ:

$$x = g(\phi, \epsilon)$$

For instance the Gaussian variable $x \sim \mathcal{N}(\mu(\phi), \sigma^2(\phi))$ can be rewritten as a function of a standard Gaussian variable $\epsilon \sim \mathcal{N}(0, 1)$, such that $x = \mu(\phi) + \sigma^2(\phi) * \epsilon$.

In that case the gradient rewrites as

$$\nabla_\phi L(\theta, \phi) = \nabla_\phi \mathbf{E}_{x \sim p_\phi(x)}[f_\theta(x)] = \nabla_\phi \mathbf{E}_{\epsilon \sim p(\epsilon)}[f_\theta(g(\phi, \epsilon)] = \mathbf{E}_{\epsilon \sim p(\epsilon)}[\nabla_\phi f_\theta(g(\phi, \epsilon)]$$
$$\nabla_\phi L(\theta, \phi) = \mathbf{E}_{\epsilon \sim p(\epsilon)}[f'_\theta(g(\phi, \epsilon)) \nabla_\phi g(\phi, \epsilon)]$$

Requirements:

- $f_\theta(x)$ **must be differentiable** w.r.t x its input. This was not the case for the score function estimator.
- $g(\phi, \epsilon)$ **must exist and be differentiable** w.r.t. ϕ. This not obvious for discrete categorical variables $x \sim \mathcal{C}at(\pi_\phi)$. However, for discrete variables, we will see that:
  - the **Gumbel-max trick** does provide a g although it is nondifferentiable w.r.t. ϕ
  - the **Gumbel-softmax trick** is a relaxation of the Gumbel-max trick that provides

Applications:

- Training variational autoencoders (VAE) with continuous latent variables. See Kingma, Welling (2014) - Auto-Encoding Variational Bayes (https://arxiv.org/abs/1312.6114)

Links:

- Kingma (2013) Fast Gradient-Based Inference (https://arxiv.org/pdf/1306.0733.pdf)

Gumbel-max trick

> In the next sections we will interchangeably use the integer representation $x \in \{1, .., K\}$ and the one-hot representation $x \in \mathbb{R}^K$ for the same discrete categorical variable x.

The Gumbel-max trick was proposed by Gumbel, Julius, Lieblein (1954) - Statistical theory of extreme values[...] (http://library.wur.nl/WebQuery/clc/429411) to express a discrete categorical variable as a deterministic function of the class probabilities and independent random variables, called Gumbel variables.

Let $x \sim \mathcal{C}at(\pi_\phi)$ be a discrete categorical variable, which can take K values, and is parameterized by $\pi_\phi \in \Delta_{K-1} \subset \mathbb{R}^K$. The obvious way to sample x is to use its cumulated distribution function to invert a uniform random variable. However, we would like to use the reparametrization trick.

Another way is to define variables $\epsilon_k \sim \mathcal{G}umbel(0, 1)$ that follow a Gumbel distribution, which can be obtained as $\epsilon_k \sim -\log(-\log u_k))$ where $u_k \sim \mathcal{U}niform(0, 1)$. Then the random variable

$$x = \arg\max_k(\epsilon_k + \log \pi_k) \widehat{=} g(\phi, \epsilon)$$

follows the correct categorical distribution $x \sim \mathcal{C}at(\pi_\phi)$.

However we cannot apply the reparametrization trick because $g(\phi, \epsilon) \widehat{=} \arg\max_k(\epsilon_k + \log \pi_k)$ is non-differentiable w.r.t the parameters ϕ that we want to optimize. We now present the Gumbel-softmax trick which relaxes the Gumbel-max trick to make g differentiable.

Applications:

- Extreme-value theory?

Gumbel-softmax

The idea of replacing the $\arg\max$ of the Gumbel-max trick with a $\mathrm{softmax}$ was concurrently presented by Jang, Gu, Poole (2017) - Categorical reparameterization with Gumbel Softmax (https://arxiv.org/pdf/1611.01144) (under the name **Gumbel-softmax**) and Maddison, Mnih, Teh (2017) - The Concrete Distribution (https://arxiv.org/pdf/1611.00712) (under the name **Concrete distribution**). More precisely, define a *Gumbal Softmax* random variable

$$(x_k)_{1 \le k \le K} = \mathbf{softmax}((\epsilon_k + \log \pi_k)_k) \Leftrightarrow x_k = \frac{\exp((\log \pi_k + \epsilon_k)/\tau)}{\sum_j \exp((\log \pi_j + \epsilon_j)/\tau)}$$

where $\tau > 0$ is a temperature parameter, and $\epsilon_k \sim \mathcal{G}umbel(0, 1)$ as before. The references give an analytical expression for the distribution of the Gumbel-softmax.

> Note that the previous expression gives the **value of x** as a deterministic function of ϵ, *not* the **distribution p(x)**. So x is actually a **continuous value** supported on the simplex Δ^{K-1}.

The above authors show interesting properties of the Gumbel-softmax:

- When $\tau \longrightarrow 0$, the vector (x_k) becomes one-hot, and as expected, the hot component follows the categorical distribution π_ϕ.
- When $\tau \longrightarrow +\infty$, the vector (x_k) becomes uniform, and all samples look the same.
- $p(x_k = \max_i x_i) = \pi_k$, since the softmax keeps the relative ordering of the π_k
- When $\tau \leq (n-1)^{-1}$, the probability density $p(x)$ becomes convex.
    - when $p(x)$ is convex, the modes are concentrated on the corners of Δ^{K-1} which means samples x will tend to be one-hot.

Now we can write $x = g(\phi, \epsilon)$ and g is differentiable w.r.t. ϕ. **We can use the reparameterization trick!**

However, note that x does not exactly follow $\mathcal{C}at(\pi_\phi)$. There is a tradeoff between having accurate one-hot samples and badly conditioned gradient with high variance (using low temperature), and having smoother samples and smaller gradient variance (with higher temperatures) . In practice the authors start with a high temperature τ and anneal to small non-zero temperatures, so as to approach the categorical distribution in the limit.

Applications:

- Training stochastic binary networks (SBN). Raiko, Berglund, Alain, Dinh (2014) - Techniques for learning binary stochastic feedforward neural networks (https://arxiv.org/pdf/1406.2989.pdf).
- Semi-supervised learning with VAE having discrete latent variables. See Jang's paper.

ST-Gumbel-softmax

For non-zero temperatures, a Gumbel-softmax variable x does not exactly follow $\mathcal{C}at(\pi_\phi)$. If in the forward pass we replace x by its argmax, then we get a one-hot variable following exactly $\mathcal{C}at(\pi_\phi)$. However, in order to backpropagate the gradient, we can still keep the original, continuous x, in the backward pass.

This is called **Straight-Through-Gumbel-softmax** in Jang's paper, and builds on ideas from Bengio, Leonard, Courville (2013) - Estimating or Propagating Gradients (https://arxiv.org/pdf/1308.3432.pdf)

Questions

- Why not just sum over all discrete values?
- How does it actually work? ST vs Non-ST is there some kind of x-weighted sum?

Todo

- Talk about REINFORCE
- Read A* sampling (https://arxiv.org/abs/1411.0030)
- Read Magenta's post on REINFORCE
- Read REBAR (https://openreview.net/pdf?id=ryBDyehOl)

Useful links

- Openreview Jang+ 2017 (https://openreview.net/forum?id=rkE3y85ee&noteId=rkE3y85ee)
- Tutorial Eric Jang (http://blog.evjang.com/2016/11/tutorial-categorical-variational.html) allows to play with the Gumbel-softmax distribution. Code for discrete VAE on MNIST in Tensorflow.