ENM 540: Data-driven modeling and probabilistic scientific computing

Tricks of the trade

Paris Perdikaris
March 29, 2018

Penn
UNIVERSITY *of* PENNSYLVANIA

Tricks of the trade

- Variational bounds
- Density re-parametrizations
- Density ration estimation
- Variational optimization/evolution strategies
- Adversarial games

Variational bounds

Typical problem:

My loss function $f(\theta)$ is intractable to compute, typically because it involves intractable marginalization. I can't evaluate it let alone minimize it.

Solution:

Let's construct a family of - typically differentiable - upper-bounds:

$$f(\theta) \leq \inf_{\psi} g(\theta, \psi),$$

and solve the optimization problem

$$\theta^*, \psi^* \leftarrow \mathrm{argmin}_{\theta,\psi}\, g(\theta, \psi)$$

instead. Technically, once optimization is finished, you can discard the auxiliary parameter ψ^* - although often turns out to be meaningful and useful in itself, often for approximate inference such as the recognition model of VAEs.

Tricks of the trade:

Jensen's inequality: The mean value of a convex function is never lower than the value of the convex function applied to the mean. Generally appears in some variant of the standard evidence lower bound (ELBO) derivation below:

$$\begin{aligned} -\log p(x) &= -\log \int p(x, y) dy \\ &= -\log \int q(y|x) \frac{p(y, x)}{q(y|x)} dy \\ &\leq -\int q(y|x) \log \frac{p(y, x)}{q(y|x)} dy \end{aligned}$$

http://www.inference.vc/design-patterns/

The re-parametrization trick

One oft-encountered problem is computing the gradient of an expectation of a smooth function f:

$$\nabla_\theta \mathbb{E}_{p(z;\theta)}[f(z)] = \nabla_\theta \int p(z;\theta) f(z) dz$$

This is a recurring task in machine learning, needed for posterior computation in variational inference, value function and policy learning in reinforcement learning, derivative pricing in computational finance, and inventory control in operations research, amongst many others. This gradient is often difficult to compute because the integral is typically unknown and the parameters θ, with respect to which we are computing the gradient, are of the distribution $p(z;\theta)$. But where a random variable z appears we can try our random variable reparameterisation trick, which in this case allows us to compute the gradient in a more amenable way:

$$\nabla_\theta \mathbb{E}_{p(z;\theta)}[f(z)] = \mathbb{E}_{p(\epsilon)}[\nabla_\theta f(g(\epsilon, \theta))]$$

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

The re-parametrization trick

Let's derive this expression and explore the implications of it for our optimisation problem. One-liners give us a transformation from a distribution $p(\epsilon)$ to another $p(z)$, thus the differential area (mass of the distribution) is invariant under the change of variables. This property implies that:

$$p(z) = \left|\frac{d\epsilon}{dz}\right| p(\epsilon) \implies |p(z)dz| = |p(\epsilon)d\epsilon|$$

Re-expressing the troublesome stochastic optimisation problem using random variate reparameterisation, we find:

$$\nabla_\theta \mathbb{E}_{p(z;\theta)}[f(z)] = \nabla_\theta \int p(z;\theta) f(z) dz$$

$$= \nabla_\theta \int p(\epsilon) f(z) d\epsilon = \nabla_\theta \int p(\epsilon) f(g(\epsilon, \theta)) d\epsilon$$

$$= \nabla_\theta \mathbb{E}_{p(\epsilon)}[f(g(\epsilon, \theta))] = \mathbb{E}_{p(\epsilon)}[\nabla_\theta f(g(\epsilon, \theta))]$$

http://www.inference.vc/design-patterns/

The density ratio trick

The central task in the above five statistical quantities is to efficiently compute the ratio $r(x)$. In simple problems, we can compute the numerator and the denominator separately, and then compute their ratio. Direct estimation like this will not often be possible: each part of the ratio may itself involve intractable integrals; we will often deal with high-dimensional quantities; and we may only have samples drawn from the two distributions, not their analytical forms.

This is where the ***density ratio trick*** *or formally, density ratio estimation*, enters: it tells us to construct a binary classifier $S(x)$ that distinguishes between samples from the two distributions. We can then compute the density ratio using the probability given by this classifier:

$$r(x) = \frac{\rho(x)}{q(x)} = \frac{S(x)}{1 - S(x)}$$

To show this, imagine creating a data set of $2N$ elements consisting of pairs (data x, label y):

- N data points are drawn from the distribution ρ and assigned a label +1.
- The remaining N data points are drawn from distribution q and assigned label −1.

http://blog.shakirm.com/2018/01/machine-learning-trick-of-the-day-7-density-ratio-trick/

The density ratio trick

By this construction, we can write the probabilities ρ, q in a conditional form; we should also keep Bayes' theorem in mind.

$$\rho(x) = p(x|y=+1); \qquad q(x) = p(x|y=-1); \qquad p(y|x) = \frac{p(x|y)p(y)}{p(x)}$$

We can do the following manipulations:

$$r(x) = \frac{\rho(x)}{q(x)} = \frac{p(x|y=+1)}{p(x|y=-1)}$$

$$= \frac{p(y=+1|x)p(x)}{p(y=+1)} \bigg/ \frac{p(y=-1|x)p(x)}{p(y=-1)}$$

$$= \frac{p(y=+1|x)}{p(y=-1|x)} = \frac{p(y=+1|x)}{1-p(y=+1|x)} = \frac{S(x)}{1-S(x)}$$

http://blog.shakirm.com/2018/01/machine-learning-trick-of-the-day-7-density-ratio-trick/