Bayesian Neural Networks and Probalistic Programming

Adam Massmann and Laureline Josset

Water Center NN Meetings

Week 5: Oct. 17th, 2017

Disclaimer

Similar to the week 1 presentation:

- None of this content is original; it is all adapted from Sections 1.6, 9.4 and 10.1 in Bishop 2006, as well as extensively from Edward's documentation at edwardlib.org (Tran et al. 2016), but any mistakes or typos are certainly my own.
- This is only intended for use within CWC, just because there is a lot of stuff I don't fully understand so there might be some errors that need to be ironed out. I think of it as a starting point that will hopefully inspire some future discussions which will help us learn this stuff together.

Differences between Bayesian Neural Nets (BNN) and the Neural Nets we've looked at?

- BNN are essentially the same, except any weights or parameters in the neural network will be random variables with a prior distribution.

Advantages/disadvantages of BNN

- Advantages
  - Model will represent uncertainty.
  - In some applications, BNN result in better models for data-sparse problems than other methods (including non-Bayesian neural nets, see Xiong, Barash, and Frey 2011 for example).
- Disadvantages
  - Numerical methods can be more complicated and some problems are intractable.
  - Sensitivity to prior?
    - Edward Box advocates criticism of model (including choice of prior) which could overcome this (Box 1982).
  - Seems like BNN are a little less "plug and play." Hyperparameters, step sizes, etc. might need to be adjusted to get reasonable fits, and numerical methods for inference seem less universal (forum discourse.edward.org has some example issues users come across).

Pause

- We'll now look at a popular numerical method for inference in Bayesian problems, from the bigger picture fundamentals (taking an information theory perspective that was new to me) down to the numerical implementation.
- It's kind of dense so if it's more useful to skip to the practical exercises say the word!

Introduction - Information Theory (see section 1.6 in Bishop)

- Consider the amount of information gained/learned by an event or observation (we'll call x). We would receive more new information from a very surprising event than an event we expect (because we already know something about the expected event).
- So if we want to quantify this "amount of information" contained in an event we should use a function of the probability of the event ($p(x)$). The amount of information we'll call a function $h(\cdot)$, which will be a function of $p(x)$.

Guidance for the functional from of $h(\cdot)$

- If two events x and y are independent, then the amount of information gained by both events should be $h(x, y) = h(x) + h(y)$.
- We also know that the joint probability of x and y's occurrence would be: $p(x, y) = p(x)\ p(y)$.
- So the question is, what function $\hat{h}$ satisfies: $h(x, y) = \hat{h}(p(x)\ p(y)) = \hat{h}(p(x)) + \hat{h}(p(y))$?

Information Entropy

- $\hat{h}(\cdot) = \log(\cdot)$ satisfies
  $h(x, y) = \hat{h}(p(x)\, p(y)) = \hat{h}(p(x)) + \hat{h}(p(y))$, so
  $h(\cdot) = \log(p(\cdot))$.
- It's desirable for h to be positive, so because $0 \leq p \leq 1$, lets make it $h(\cdot) = -\log p(\cdot)$.
- Now say we have a bunch of random variables x for which we want to know the average amount of information (i.e. expectation of $h(x)$). This would be given by:

$$H[x] = -\sum_{x} p(x) \log p(x)$$

- This is known as the *entropy* of x.
- Extending this to continuous variables gives the *differential entropy*:

$$H[x] = -\int p(x)\, \log p(x)\, dx$$

[Bishop 2006, Shannon 1948]

So what does information entropy look like?

- From thermodynamics and statistical mechanics we have some idea of entropy as a measure of the disorder or randomness in a system. For information theory it is similar.[1]

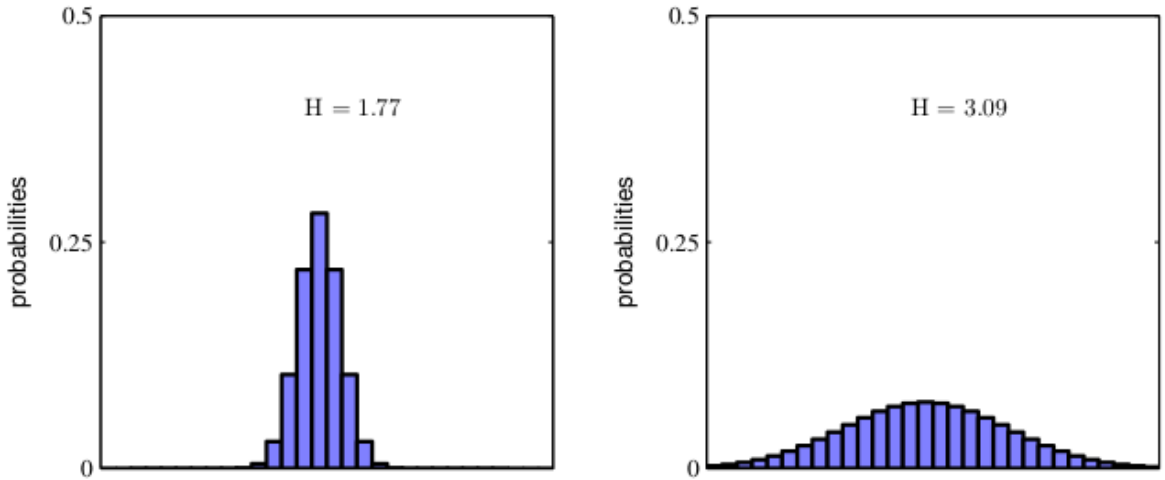

Figure 1: From Bishop 2006: Histograms of two probability distributions over thirty bins illustrating the higher value of entropy H for the broader distribution. The largest entropy would arise from a uniform distribution that would give $H = -\ln 1/30 = 3.40$

[1] von Neumann told Shannon he should also call it entropy because "nobody knows what entropy really is, so in any discussion you will always have an advantage." (Bishop 2006)

Kullback-Leibler divergence

- Why should we even care about information theory or entropy?
  - Because we can use entropy ideas to approximate inference on a probabilistic model, given data.
  - Say we have some phenomenon with a true probability distribution $p(x)$, which we are approximating with some [possibly parametric] distribution $q(x)$.
  - Then the additional necessary information required to communicate the value of x as consequence of using $q(x)$ would be:

$$\begin{aligned} KL(p\|q) &= -\int p(x) \ln q(x) dx - \left(-\int p(x) \ln p(x) dx\right) \\ &= -\int p(x) \ln \frac{q(x)}{p(x)} dx \end{aligned} \tag{1}$$

This is known as relative entropy or Kullback-Leibler (KL) divergence (Kullback and Leibler 1951).

Properties of Kullback-Leibler divergence

- Note that it is not a symmetrical quantity (e.g. $KL(p\|q) \neq KL(q\|p)$).
- Also, $KL(p\|q) \geq 0$,
- and $KL(p\|q) = 0$ only if p and q are identical (see Bishop 2006 for proof).
- So practically speaking KL-divergence is very useful as a cost function quantifying the similarity between two probability distributions.

Equivalence of KL-divergence and negative log likelihood

- Say we have N observations of data x_n from some unknown probability distribution $p(x)$.
- We want to try to approximate $p(x)$ with a parametric distribution $q(x|\theta)$, by minimizing the KL-divergence which can be approximated by:

$$KL(p\|q) \simeq \frac{1}{N}\sum_{n=1}^{N}[-\ln q(x_n|\theta) + In(p(x_n))] \tag{2}$$

- The second term is not a function of θ, and the first term is just the negative log likelihood. So for this example, minimizing KL-divergence is the same as minimizing the negative log likelihood, which we saw in Week 1!

KL-divergence for Bayesian problems (Tran et al. 2016)

- Say we have some group of latent variables z that define a hidden structure behind our data x. We can use Bayes' rule to define the distribution of z given our observed data x (the *posterior*):

$$p(z \mid x) = \frac{p(x, z)}{\int p(x, z) dz} = \frac{p(x|z)p(z)}{p(x)}$$

- The main computational problem in calculating the posterior is that the normalizing constant is usually intractable. So instead of calculating the posterior we will approximate the posterior.

Approximating the posterior

- We can approximate the posterior $p(z \mid x)$ with some probability distribution $q(z; \lambda)$, where q is a distribution of the latent variables z parameterized by λ.
- Fortunately, we have already defined a tool (KL-divergence) that is a measure of the difference between two probability distributions.
- So the problem then becomes:

$$\lambda^* = \text{argmin}_\lambda \text{divergence}(p(z|x), q(z; \lambda))$$

- Which we will frame as:

$$\lambda^* = \text{argmin}_\lambda \text{KL}(q(z; \lambda), p(z|x))$$

- But, we still have $p(z|x)$ so we need to get rid of that.

[Tran et al. 2016]

Minimizing KL-divergence

- To get rid of dependence of $p(z|x)$ we can use:

$$\log p(x) = KL(q(z;\lambda) \| p(z|x)) + \mathop{\mathbb{E}}_{q(z;\lambda)} [\log p(x,z) - \log q(z;\lambda)]$$

  - because $\log p(x)$ is invariant to λ, we can subtract this from $\mathrm{KL}(q(z;\lambda), p(z|x))$ without affecting λ^*.
  - in which case we are left with:

$$\lambda^* = \mathrm{argmin}_\lambda - \mathop{\mathbb{E}}_{q(z;\lambda)} [\log p(x,z) - \log q(z;\lambda)]$$

- So minimizing KL-divergence is the same as maximizing:

$$\mathrm{ELBO}(\lambda) = \mathop{\mathbb{E}}_{q(z;\lambda)} [\log p(x,z) - \log q(z;\lambda)]$$

  Where ELBO is an acronym standing for the "Evidence Lower Bound."

[Tran et al. 2016]

Maximizing ELBO using score function gradient

- Gradient ascent is used to maximize ELBO. We are then interested in:

$$\nabla_\lambda ELBO(\lambda) = \nabla_\lambda \mathop{\mathbb{E}}_{q(z;\lambda)} [\log p(x,z) - \log q(z;\lambda)]$$

- This can be rewritten:

$$\nabla_\lambda ELBO = \mathop{\mathbb{E}}_{q(z;\lambda)} \nabla_\lambda \log q(z;\lambda) \left(\log p(x,z) - \log q(z;\lambda)\right) \tag{3}$$

- Where so long as we know the score function $\nabla_\lambda \log q(z;\lambda)$ we can use Monte Carlo integration to estimate both ELBO and its gradient (Ranganath, Gerrish, and Blei 2014):
  1. Draw S samples z_s from $q(z;\lambda)$.
  2. evaluate the terms in equation 3 using z_s.
  3. compute the empirical mean across the samples of the evaluated terms from (2).

Gradient ascent for ELBO (Ranganath, Gerrish, and Blei 2014; Tran et al. 2016)

- From the previous slide: if we know the score function $\nabla_\lambda \log q(z;\lambda)$ we can use Monte Carlo integration to estimate both ELBO and its gradient:
  1. Draw S samples z_s from $q(z;\lambda)$.
  2. evaluate the terms in equation 3 using z_s.
  3. compute the empirical mean across the samples of the evaluated terms from (2).
- Which gives the following estimate of the gradient:

$$\nabla_\lambda ELBO(\lambda) \approx \frac{1}{S}\sum_{s=1}^{S}\left[\left(\log p(x,z_s) - \log q(z_s;\lambda)\right)\nabla_\lambda \log q(z_s;\lambda)\right]$$

Pseudocode for ELBO gradient ascent

$$\nabla_\lambda ELBO(\lambda) \approx \frac{1}{S}\sum_{s=1}^{S}\left[\left(\log p(x, z_s) - \log q(z_s;\lambda)\right)\nabla_\lambda \log q(z_s;\lambda)\right]$$

Algorithm 1 Black Box Variational Inference

Input: data x, joint distribution p, mean field variational family q.
Initialize λ randomly, $t = 1$.
repeat
  // **Draw** S **samples from** q
  for $s = 1$ **to** S **do**
    $z[s] \sim q$
  end for
  $\rho = t$th value of a Robbins Monro sequence
  $\lambda = \lambda + \rho \frac{1}{S}\sum_{s=1}^{S} \nabla_\lambda \log q(z[s] \mid \lambda)(\log p(x, z[s]) - \log q(z[s] \mid \lambda))$
  $t = t + 1$
until change of λ is less than 0.01.

Figure 2: Reproduced from Ranganath, Gerrish, and Blei 2014.

On to the practical exercise...

- Note I've only covered one numerical method, and the exercise focuses on the very basics of setting up and inferring a BNN in Edward.
- There are a lot more (and more sophisticated) techniques that leverage probabilistic programming, so I strongly encourage anyone to explore edwardlib.org for more examples if they are interested.
- The focus here was on neural networks because that's what we've been talking about, and I kept it relatively simple because this is all new for me.

References

See sections 1.6, 9.4, and 10.1 in Bishop, and edwardlib.org


Bishop, Christopher M (2006). *Pattern recognition and machine learning*. springer.


Box, George EP (1982). *An Apology for Ecumenism in Statistics.*. Tech. rep. WISCONSIN UNIV-MADISON MATHEMATICS RESEARCH CENTER.


Kullback, Solomon and Richard A Leibler (1951). "On information and sufficiency". In: *The annals of mathematical statistics* 22.1, pp. 79–86.


Ranganath, Rajesh, Sean Gerrish, and David Blei (2014). "Black box variational inference". In: *Artificial Intelligence and Statistics*, pp. 814–822.


Shannon, Claude E (1948). "A mathematical theory of communication, Part I, Part II". In: *Bell Syst. Tech. J.* 27, pp. 623–656.


Tran, Dustin et al. (2016). "Edward: A library for probabilistic modeling, inference, and criticism". In: *arXiv preprint arXiv:1610.09787*.


Xiong, Hui Yuan, Yoseph Barash, and Brendan J Frey (2011). "Bayesian prediction of tissue-regulated splicing using RNA sequence and cellular context". In: *Bioinformatics* 27.18, pp. 2554–2562.