EECS 182 Deep Neural Networks
Spring 2023 Anant Sahai

Discussion 13

1. Entropy, Cross-Entropy, Kullback - Leibler (KL)-divergence

Entropy is a measure of expected surprise. For a given discrete Random variable Y, we know that from Information Theory that a measure the surprise of observing that Y takes the value k by computing:

$$\log \frac{1}{p(Y=k)} = -\log[p(Y=k)]$$

As given:

- if $p(Y=k) \to 0$, the surprise of observing k approaches ∞
- if $p(Y=k) \to 1$, the surprise of observing k approaches 0

The Entropy of the distribution of Y is then the expected surprise given by:

$$H(Y) = E_Y\Big[-\log\big(p(Y=k)\big)\Big] = -\Sigma_k\Big[p(Y=k)\log[p(Y=k)]\Big]$$

On the other hand, Cross-entropy is a measure building upon entropy, generally calculating the difference between two probability distributions p and q. it is given by:

$$\begin{aligned} H(p,q) &= E_{p(x)}\Big[\frac{1}{\log\big(q(x)\big)}\Big] \\ &= \Sigma_x\Big[p(x)\log[\frac{1}{q(x)}]\Big] \end{aligned}$$

Relative Entropy also known as KL Divergence measures how much one distribution diverges from another. For two discrete probability distributions, p and q, it is defined as:

$$D_{KL}(p||q) = \Sigma_x\Big[p(x)\log[\frac{p(x)}{q(x)}]\Big]$$

(a) Let's define the following probability distributions given by:

$$p(x) = \begin{cases} 1 & \text{with probability 0.5} \\ -1 & \text{with probability 0.5} \end{cases}$$

$$q(x) = \begin{cases} 1 & \text{with probability 0.1} \\ -1 & \text{with probability 0.9} \end{cases}$$

Show that KL-divergence is not symmetric and hence does not satisfy some intuitive attributes of distances.

Solution:

To show this, we need to show that:

$$D_{KL}(p||q) \neq D_{KL}(q||p)$$

$$D_{KL}(p||q) = 0.5 \times \log[\frac{0.5}{0.1}] + 0.5 \times \log[\frac{0.5}{0.9}]$$
$$D_{KL}(q||p) = 0.1 \times \log[\frac{0.1}{0.5}] + 0.9 \times \log[\frac{0.9}{0.5}]$$

hence $D_{KL}(p||q) \neq D_{KL}(q||p)$

(b) Re-write $D_{KL}(p||q)$ in term of the Entropy $H(p)$ and the cross entropy $H(p,q)$.

Solution:

$$\begin{aligned}
D_{KL}(p||q) &= \Sigma_x \Big[p(x) \log[\frac{p(x)}{q(x)}]\Big] \\
&= \Sigma_x \Big[p(x)[\log\big(p(x)\big) - \log\big(q(x)\big)]\Big] \\
&= E_{p(x)}\Big[\log\big(p(x)\big)\Big] - E_{p(x)}\Big[\log\big(q(x)\big)\Big] \\
&= -E_{p(x)}\Big[\log\big(q(x)\big)\Big] + E_{p(x)}\Big[\log\big(p(x)\big)\Big] \\
&= E_{p(x)}\Big[\frac{1}{\log\big(q(x)\big)}\Big] - E_{p(x)}\Big[\frac{1}{\log\big(p(x)\big)}\Big] \\
&= H(p,q) - H(p)
\end{aligned}$$

2. Reparameterization Trick

Formally, a latent variable model p is a probability distribution over observed variables x and latent variables z (variables that are not directly observed but inferred), $p_\theta(x,z)$. Because we know z is unobserved, using learning methods learned in class (like supervised learning methods) is unsuitable. Indeed, our learning problem of maximizing the log-likelihood of the data turns from:

$$\theta \leftarrow arg\max_\theta \frac{1}{N}\Sigma_{i=1}^N \log[p_\theta(x_i)]$$

to:

$$\theta \leftarrow arg\max_{\theta} \frac{1}{N}\Sigma_{i=1}^{N}\log[\int p_\theta(x_i \mid z)p(z)dz]$$

where $p(x)$ has become $\int p_\theta(x_i \mid z)p(z)dz$.

(a) State **whether or not we could directly maximize the likelihood above and why?**

Solution: No, we can't because, in the integral, it is intractable to compute $p_\theta(x \mid z)$ for every z. On the other hand, if we look at the posterior density given by $p_\theta(z \mid x) = \frac{p_\theta(x|z)p_\theta(z)}{p_\theta(x)}$, we can see that $p_\theta(x)$ is also intractable.

(b) Instead of directly optimizing the likelihood of $p(x)$, we define the proxy likelihood as:

$$\mathcal{L}(x_i, \theta, \phi) = E_{z\sim q_\phi(z|x_i)}\Big[\log[p_\theta(x_i \mid z)]\Big] - D_{KL}\Big[q_\phi(z \mid x_i)||p(z)\Big]$$

This proxy term is a *lower bound* of the original likelihood. In order to optimize this variational lower bound, **which distribution do we sample from?**

Solution: We sample from $q_\phi(z \mid x_i)$

(c) How do we take gradients through samples? To do we, we need to show how sampling can be done as a deterministic and continuous function of the model parameters θ and the independent source of randomness (ie. the *prior*). Such an explicit representation of sampling is called **reparameterization**. Consider the case where the data x is sampled from a normal distribution with its mean parameterized by parameters θ and variance of 1, with our objective being a quadratic function of x:

$$\min_{\theta} E_q[x^2]$$

Write x as a function of ϵ, a vector sampled from a standard Normal $\mathcal{N}(0, 1)$, and compute the gradient of the expectation term above:

Solution: We can first make the stochastic element in q independent of θ, and rewrite x as:

$$x = \theta + \epsilon, \epsilon \sim \mathcal{N}(0, 1)$$

$$E_q[x^2] = E_\epsilon[(\theta + \epsilon)^2]$$

Hence we can write the derivative of $E_q[x^2]$ as:

$$\begin{aligned}\nabla_\theta E_q[x^2] &= \nabla_\theta E_\epsilon[(\theta+\epsilon)^2] \\ &= E_\epsilon[2(\theta+\epsilon)]\end{aligned}$$

(d) Now consider a more generic case where we would like to optimize

$$\min_\theta E_v[\mathcal{L}(x)]$$

where x is sampled from a learnt latent function $f_\theta(u, v)$ that is dependent on u the input data and v the independent randomness. **Show that the gradient $\nabla_\theta E_v[\mathcal{L}(x)]$ can be estimated by samples of $\nabla_\theta f_\theta(u, v)$.** (*Hint*: the process of this question is very similar to the previous part.)

Solution: The gradient of expectation can be expressed by the expectation of gradients, which can be sampled from an *independent* randomness that needs not to be a Gaussion or any fixed prior.

$$\begin{aligned}\nabla_\theta E_v[\mathcal{L}(x)] &= E_v[\nabla_\theta \mathcal{L}(f_\theta(u, v))] \\ &\approx \frac{1}{m}\sum_{i=1}^{m} \nabla_\theta \mathcal{L}(f_\theta(u, v))\end{aligned}$$

Hence this optimization problem can be handled simply by the samples from v.

3. Latent Variable Models

(a) **Describe step-by-step what happens during a forward pass in VAE training.** Use the notation from the variational lower bound term (the "proxy likelihood") in the previous question, namely $q_\phi(z \mid x), p_\theta(z \mid x_i), D_{KL}(\cdot)$ ... etc.

Solution: For a forward pass, through which we run our minibatch of input data,

i. We pass this through our Encoder network ($q_\phi(z \mid x)$). Note this is specifically optimized through the second term in our lower bound loss function (ELBO) i. e $D_{KL}(q_\phi(z \mid x_i)||p_\theta(z \mid x_i))$ whose only goal is to make an approximation of our posterior distribution.

ii. We then sample z from $z|x \sim \mathcal{N}(\mu_{z|x}, \Sigma_{z|x})$. These are the samples of latent factors that we can infer from x

iii. We pass the obtained z through our Decoder network ($p_\theta(x \mid z)$). We then sample $\hat{x}$ from $x|z \sim \mathcal{N}(\mu_{x|z}, \Sigma_{x|z})$. Note that is handled specifically by the first term is our loss i. e $E_{z\sim q_\phi(z|x_i)}\Big[\log p_\theta(x_i \mid z)\Big]$ whose only goal is to maximize the likelihood of the original input being reconstructed.

iv. Compute the loss, which is differentiable, then backpropagate and update parameters.

(b) **Describe what the encoder and decoder of the VAE are *respectively* doing** to capture and encode this information into a latent representation of space z. **Is the latent space dimension smaller that the input space? How is the information bottleneck created in VAE as opposed to Autoencoder.**

Solution:

i. **Encoder** - Encoder maps a high-dimensional input x (like the pixels of an image) and then (most often) outputs the parameters of a Gaussian distribution that specify the hidden variable z. In other words, they output $\mu_{z|x}$ and $\Sigma_{z|x}$. We will implement this as a deep neural network, parameterized by ϕ, which computes the probability $q_\phi(z|x)$. We could then sample from this distribution to get noisy values of the representation z.

ii. **Decoder** - Decoder maps the latent representation back to a high dimensional reconstruction, denoted as $\hat{x}$, and outputs the parameters to the probability distribution of the data. We will implement this as another neural network, parametrized by θ, which computes the probability $p_\theta(x|z)$. In the MNIST dataset example, if we represent each pixel as a 0 (black) or 1 (white), the probability distribution of a single pixel can be then represented using a Bernoulli distribution. Indeed, the decoder gets as input the latent representation of a digit z and outputs 784 Bernoulli parameters, one for each of the 784 pixels in the image.

(c) Once the VAE is trained, **how do we use it to generate a new fresh sample from the learned approximation of the data-generating distribution?**

Solution: We can now use only the Decoder network ($p_\theta(x \mid z)$). Here, instead of sampling z from the posterior that we had during training, we sample from our true generative process which is the prior that we had specified ($z \sim \mathcal{N}(0, I)$) and we proceed to use the network to sample $\hat{x}$ from there.

(d) In the previous question we have used a proxy likelihood:

$$\mathcal{L}(x_i, \theta, \phi) = E_{z \sim q_\phi(z|x_i)}\Big[\log[p_\theta(x_i \mid z)]\Big] - D_{KL}\Big[q_\phi(z \mid x_i)||p(z)\Big]$$

Please show that $\mathcal{L}(x_i, \theta, \phi)$ is always a lower bound to the true log likelihood for x_i.

Hint: You can show that something is a lower bound by showing that adding a non-negative term to it gives the original quantity — remember, the KL divergence is always non-negative.

Solution:

$$\begin{aligned}
\log p_\theta(x_i) &= E_{z\sim q_\phi(z|x_i)}\Big[\log p_\theta(x_i)\Big] \\
&= E_{z\sim q_\phi(z|x_i)}\Big[\log \frac{p_\theta(x_i \mid z)p_\theta(z)}{p_\theta(z \mid x_i)}\Big] \\
&= E_{z\sim q_\phi(z|x_i)}\Big[\log \frac{p_\theta(x_i \mid z)p_\theta(z)}{p_\theta(z \mid x_i)}\frac{q_\phi(z \mid x_i)}{q_\phi(z \mid x_i)}\Big] \\
&= E_{z\sim q_\phi(z|x_i)}\Big[\log p_\theta(x_i \mid z)\Big] - E_{z\sim q_\phi(z|x_i)}\Big[\log \frac{q_\phi(z \mid x_i)}{p_\theta(z)}\Big] + E_{z\sim q_\phi(z|x_i)}\Big[\log \frac{q_\phi(z \mid x_i)}{p_\theta(z \mid x_i)}\Big] \\
&= E_{z\sim q_\phi(z|x_i)}\Big[\log p_\theta(x_i \mid z)\Big] - D_{KL}(q_\phi(z \mid x_i)||p_\theta(z)) + D_{KL}(q_\phi(z \mid x_i)||p_\theta(z \mid x_i)) \\
&= \mathcal{L}(x_i, \theta, \phi) + D_{KL}(q_\phi(z \mid x_i)||p_\theta(z \mid x_i))
\end{aligned}$$

Because $D_{KL}(q_\phi(z \mid x_i)||p_\theta(z \mid x_i)) \geq 0$, and is not tractable due to $p_\theta(z \mid x_i)$ we can conclude that:
$\log p_\theta(x_i) \geq \mathcal{L}(x_i, \theta, \phi) = E_{z\sim q_\phi(z|x_i)}\Big[\log p_\theta(x_i \mid z)\Big] - D_{KL}(q_\phi(z \mid x_i)||p_\theta(z))$
Alternatively we could use Jensen's Inequality, which states, $\log E[X] \geq E[\log X]$ to show that:

$$\Sigma_{i=1}^N \log[p_\theta(x_i)] \geq \Sigma_{i=1}^N E_{q(z|x_i)}[\log\big(p_\theta(z)\big) - \log\big(p_q(z \mid x_i)\big) + \log\big(p_\theta(x_i \mid z)\big)]$$

That is:
We first write out the log-likelihood objective of a discrete latent variable model.

$$arg\max_\theta \frac{1}{N}\Sigma_{i=1}^N \log[p_\theta(x_i)] = arg\max_\theta \frac{1}{N}\Sigma_{i=1}^N log[\Sigma_z p_\theta(x_i \mid z)p_\theta(z)]$$

then,

$$\begin{aligned}
\Sigma_{i=1}^N \log[p_\theta(x_i)] &= \Sigma_{i=1}^N \Big(\Sigma_z \log[p_\theta(z)p_\theta(x_i \mid z)]\Big) \\
&= \Sigma_{i=1}^N \Big(\Sigma_z \log[\frac{q_\phi(z \mid x_i)}{q_\phi(z \mid x_i)}p_\theta(z)p_\theta(x_i \mid z)]\Big) \\
&= \Sigma_{i=1}^N \Big(\Sigma_z \log E_{q_\phi(z|x_i)}[\frac{1}{q_\phi(z \mid x_i)}p_\theta(z)p_\theta(x_i \mid z)]\Big) \\
\Sigma_{i=1}^N \log[p_\theta(x_i)] &\geq \Sigma_{i=1}^N E_{q(z|x_i)}[\log\big(p_\theta(z)\big) - \log\big(p_q(z \mid x_i)\big) + \log\big(p_\theta(x_i \mid z)\big)]
\end{aligned}$$

Contributors:

- Jerome Quenum.
- Anant Sahai.
- Kevin Li.
- Past CS282 Staff.

