TTIC 31230 Fundamentals of Deep Learning, 2020
Quiz 1

In all problems we assume that all probability distributions are discrete so that for a distribution P we have $\sum_x P(x) = 1$.

Problem 1 (20 pts) Consider a joint distribution $P(x, y)$ on discrete random variables x and y. We define the marginal distributions $P(x)$ and $P(y)$ as follows.

$$\begin{aligned} P(x) &= \sum_y P(x, y) \\ P(y) &= \sum_x P(x, y) \end{aligned}$$

We define the conditional entropies $H(y|x)$ and $H(x|y)$ as follows

$$\begin{aligned} H(y|x) &= E_{x\sim P(x)}\ E_{y\sim P(y|x)}\ -\ln P(y|x) \\ H(x|y) &= E_{y\sim P(y)}\ E_{x\sim P(x|y)}\ -\ln P(x|y) \end{aligned}$$

Let $Q(x, y)$ be defined to be the product of marginals.

$$Q(x, y) = P(x)P(y).$$

Derive the following equalities.

$$KL(P(x, y), Q(x, y)) = H(y) - H(y|x) = H(x) - H(x|y)$$

The above quantity is called the mutual information between x and y, written $I(x, y)$. Explain why this quantity is always non-negative.

Solution:

$$\begin{aligned} I(x,y) &= KL(P(x,y), Q(x,y)) \\ &= E_{(x,y)\sim P(x,y)}\ \ln \frac{P(x,y)}{P(x)P(y)} \\ &= E_{(x,y)\sim P(x,y)}\ \ln \frac{P(x)P(y|x)}{P(x)P(y)} \\ &= E_{(x,y)\sim P(x,y)}\ \ln \frac{P(y|x)}{P(y)} \\ &= E_{(x,y)\sim P(x,y)}\ (-\ln P(y) + \ln P(y|x)) \\ &= \big(E_{(x,y)\sim P(x,y)}\ (-\ln P(y))\big) - \big(E_{(x,y)\sim P(x,y)}\ -\ln P(y|x))\big) \\ &= H(y) - H(y|x) \end{aligned}$$

The derivation of $I(x,y) = H(x) - H(x|y)$ is similar.
I(x,y) is non-negative because KL divergence is always non-negative.

Problem 2 (20 pts): We consider a model distribution $Q_\Phi(z,y)$ with marginal distribution

$$Q_\Phi(y) = \sum_z Q_\Phi(z,y).$$

We are interested in minimizing the unconditional (or unsupervised) cross-entropy of this model.

$$\Phi^* = \operatorname*{argmin}_\Phi E_{y\sim\text{Train}} - \ln Q_\Phi(y)$$

For many models of interest $Q_\Phi(z,y)$ can be efficiently computed as $Q_\Phi(z)Q_\Phi(y|z)$ but $Q_\Phi(y)$ is intractable to compute. In a variational auto-encoder we train a second model $\tilde{Q}_\Psi(z|y)$ and use the following inequality

$$\begin{aligned} \ln Q_\Phi(y) &\geq \text{ELBO} \\ &= E_{z\sim\tilde{Q}(z|y)} \ln \frac{Q_\Phi(z,y)}{\tilde{Q}_\Psi(z|y)} \end{aligned}$$

Rather than minimize the cross entropy we can maximize the ELBO (the Evidence Lower BOund) which corresponds to minimizing an upper bound on the cross entropy. Maximization of the ELBO with respect to model parameters Φ and Ψ define a variational auto encoder (VAE). We will consider this in much more detail later in the class. For now we just consider the formal equations.

a. The ELBO can be written as

$$\text{ELBO} = E_{z\sim\tilde{Q}(z|y)} \ln \frac{Q_\Phi(y)Q_\Phi(z|y)}{\tilde{Q}_\Psi(z|y)}.$$

Here we have that the ELBO is the expectation of a log of a product of three terms. Separate all three terms and express the terms other than $\ln Q_\Phi(y)$ as entropies or cross entropies.

Solution:

$$\begin{aligned} ELBO &= E_{z\sim\tilde{Q}_\Psi(z|y)} \ln \frac{Q_\Phi(y)Q_\Phi(z|y)}{\tilde{Q}_\Psi(z|y)} \\ &= \left(E_{z\sim\tilde{Q}_\Psi(z|y)} \ln Q_\Phi(y)\right) + \left(E_{z\sim\tilde{Q}_\Psi(z|y)} \ln Q_\Phi(z|y)\right) + \left(E_{z\sim\tilde{Q}_\Psi(z|y)} \ln \frac{1}{\tilde{Q}_\Psi(z|y)}\right) \\ &= \ln Q_\Phi(y) - H(\tilde{Q}_\Psi(z|y), Q_\Phi(z|y)) + H(\tilde{Q}(z|y)) \end{aligned}$$

b. Now rewrite the ELBO by separating it into one the term for $P_\Phi(y)$ and one term for the other two combined and write the combined term as a KL

divergence. Explain why your expression implies that the ELBO is a lower bound on $\ln Q_\Phi(y)$.

Solution:

$$\begin{aligned} ELBO &= E_{z\sim\tilde{Q}_\Psi(z|y)} \ln \frac{Q_\Phi(y)Q_\Phi(z|y)}{\tilde{Q}_\Psi(z|y)} \\ &= \left(E_{z\sim\tilde{Q}_\Psi(z|y)} \ln Q_\Phi(y)\right) + \left(E_{z\sim\tilde{Q}_\Psi(z|y)} \ln \frac{Q_\Phi(z|y)}{\tilde{Q}_\Psi(z|y)}\right) \\ &= \ln Q_\Phi(y) - KL(\tilde{Q}_\Psi(z|y), Q_\Phi(z|y)) \end{aligned}$$

The lower bound property follows from the fact that KL divergence is non-negative.

Problem 3. (35 pts) Consider the following set of `+=` statements defining batch normalization where all computed tensors are initialized to zero.

For b, j $\mu[j]$ `+=` $\frac{1}{B}\ x[b,j]$

For b, j $s[j]$ `+=` $\frac{1}{B-1}\ (x[b,j]-\mu[j])^2$

For b, j $x'[b,j]$ `+=` $\frac{x[b,j]-\mu[j]}{\sqrt{s[j]}}$

Give backpropagation `+=` (or `-=`) loops for computing $x.\text{grad}[b,j]$, $\mu.\text{grad}[j]$, and $s.\text{grad}[j]$ from $x'.\text{grad}[b,j]$. The loops should be given in the order they are to be executed.

Solution:

For b, j $x.grad[b,j]$ `+=` $\frac{x'.\text{grad}[b,j]}{\sqrt{s[j]}}$

For b, j $\mu.grad[j]$ `-=` $\frac{x'.\text{grad}[b,j]}{\sqrt{s[j]}}$

For b, j $s.grad[j]$ `-=` $\frac{1}{2}(x[b,j]-\mu[j])s[j]^{-3/2}\ x'.grad[b,j]$

For b, j $x.grad[b,j]$ `+=` $\frac{2}{B-1}\ (x[b,j]-\mu[j])s.\text{grad}[j]$

For b, j $\mu.\text{grad}[j]$ `-=` $\frac{2}{B-1}\ (x[b,j]-\mu[j])s.\text{grad}[j]$

For b, j $x.grad[b,j]$ `+=` $\frac{1}{B}\ \mu.\text{grad}[j]$

Problem 4. (25 pts) Consider a function $c : R^d \times R^s \to R^s$, in other words a function that takes a vector of dimension d and a vector of dimension s and

yields a vector of dimension s. Given a sequence of vectors x_0, x_2, ..., x_T with $x_t \in R^d$ we can define a sequence of vectors h_0, h_1, ..., h_T by the equations

$$\begin{aligned} h_0 &= c(x_0, 0) \\ h_t &= c(x_t, h_{t-1}) \text{ for } 1 \leq t \leq T \end{aligned}$$

When the function c is defined by a neural network the resulting network mapping x_1, ..., x_T to h_0, ..., h_T is called a recurrent neural network (RNN).
a. In the educational framework EDF we work with objects where each object has a value attribute and a gradient attribute each of which have tensor values where the value tensor and the gradient tensor are the same shape. Each object is assigned a value in a forward pass and assigned a gradient in a backward pass. Suppose that we are given an EDF procedure `CELL` which takes as arguments a parameter object `Phi` and two EDF objects `X` and `H` where the value attribute of the object `X` is a d-dimensional vector and the value attribute of the object `H` is an s-dimensional vector. A call to the procedure `CELL(Phi,X,H)` returns an EDF object whose value attribute is computed in a forward pass in some possibly complex way from the value attributes of `Phi`, `X` and `H`. Given a sequence `X[]` of EDF objects whose value attributes are d-dimensional vectors, and an EDF object `ZERO` representing the constant s-dimensional zero vector, write a procedure for constructing the sequence of EDF objects representing h_1, h_2, ..., h_T as defined by the above RNN equations. Your solution can be in Python or informal high level pseudo code.

Solution: We can use the equations given as the definition of the computation graph if we replace c in the equations with the function `CELL`.

```
X = list()
H = list()
H[0] = CELL(Phi,X[0],ZERO)
for t in range(1,T)
    H[t] = CELL(Phi,X[t],H[t-1])
```

b. Deep learning systems generally make extensive use of parallel computation for training. How does the parallel running time of an RNN computation graph scale with the length T?

Solution: The parallel running time is proportional to T. RNNS are fundamentally serial and this is a problem. RNNs have recently been largely replaced by the transformer architecture.