TTIC 31230 Fundamentals of Deep Learning
Quiz 4

Problem 1: Upper Bounding $H(y)$

We consider a population distribution Pop on a set of observable values y and a stochastic encoder defining a conditional distribution $P_\Psi(z|y)$. We assume that we can sample from $P_\Psi(z|y)$ and that for any given z and y we can compute $P_\Psi(z|y)$. The population and the encoder define a joint distribution $P_{\text{Pop},\Psi}(y,z)$ where y is drawn from the population and z is drawn from $P_\Psi(z|y)$. All probabilities and information-theoretic quantities in this problem refer to this joint distribution.
We will use the fact that mutual information satisfies

$$I(y,z) = H(y) - H(y|z) = H(z) - H(z|y)$$

which implies

$$H(y) = H(z) - H(z|y) + H(y|z) \tag{1}$$

(a) Rewrite (1) in terms of expectations over $y \sim \text{Pop}$ and $z \sim P_\Psi(z|y)$ of quantities defined on $\text{Pop}(y)$, $P_{\text{Pop},\Psi}(z)$, $P_{\text{Pop},\Psi}(z|y)$ and $P_{\text{Pop},\Psi}(y|z)$.

Solution:

$$E_{y\sim\text{Pop}} -\ln\ \text{Pop}(y) = E_{y\sim\text{Pop},\ z\sim P_\Psi(z|y)}\ -\ln P_\Psi(z) + \ln P_{\text{Pop},\Psi}(z|y) - \ln P_{\text{Pop},\Psi}(y|z)$$

(b) Which of the terms in (1) can be directly esimated by simply sampling $y \sim \text{Pop}$ and $z \sim P_\Psi(z|y)$.

Solution: Just the middle term $H(z|y)$.

(c) Recall that the cross-entropy $H(P,Q)$ is defined to be $E_{x\sim P}\ -\ln Q(x)$ and that $H(P) \le H(P,Q)$ for any Q. Let $P_\Phi(z)$ and $P_\Theta(y|z)$ be two additional models and consider the cross entropies $H(P_{\text{Pop},\Psi}(z), P_\Phi(z))$ and $H(P_{\text{Pop},\Psi}(y|z), P_\Theta(y|z))$. Using the fact that cross-entropies upper bound entropies give an upper bound on $H(y)$ derived from (1) by replacing entropies by cross-entropies to these models. Express your upper bound as an expectation over sampling.

Solution:

$$H(y) \le E_{y\sim\text{Pop},\ z\sim P_\Psi(z|y)}\ -\ln P_\Phi(z)\ +\ln P_\Psi(z|y)\ -\ln P_\Theta(y|z)$$

(d) Which terms in you solution to (c) can be estimated directly by sampling.

Solution: All

(e) Consider minimizing the upper bound on $H(y)$ given in your solution to (c). How is this related the VAE training objective?

Solution: It is exactly the same.

Problem 2. Training Vector Quantization

Vector quantization (VQ) can be interpreted as introducing symbols. It uses an embedding matrix $E[K, I]$ giving an embedding vector $E[k, I]$ for each of K discrete "symbols". To make the notation more compact we will write $e(k)$ for the embedding vector $E[k, I]$ of the symbol k. We define the quantization operation to map a vector to the symbol whose embedding is nearest to that vector.

$$\text{nearest}_E(x) = \underset{k}{\text{argmin}} \ ||x - e(k)||$$

We consider a VQ-VAE where the latent variable is a single symbol (from a possibly large collection of K symbols). In this case the VQ-VAE optimizes the following objective.

$$\Phi^* = \underset{\Phi}{\text{argmin}} \ E_{y \sim \text{Pop}} \ ||y - y_\Phi(e(\text{nearest}_E(x_\Psi(y))))||^2 \tag{2}$$

$$\Psi^* = \underset{\Psi}{\text{argmin}} \ E_{y \sim \text{Pop}} \begin{cases} & ||y - y_\Phi(e(\text{nearest}_E(x_\Psi(y))))||^2 \\ + & \beta||x_\Psi(y) - e(\text{nearest}_E(x_\Psi(y)))||^2 \end{cases} \tag{3}$$

$$E^* = \underset{E}{\text{argmin}} \ E_{y \sim \text{Pop}} \ ||x_\Psi(y) - e(\text{nearest}_E(x_\Psi(y)))||^2 \tag{4}$$

I have written this as a separate objective function for each component of the model. The objective for a component defines a gradient for that component. Multiple simultaneous objectives define a multi-player game. We hope to reach a Nash equilibrium where this is defined as a parameter setting where all the objectives have zero gradients — each "player" is doing a locally best (or at least stationary) response. Multiple objectives can be implemented by putting stop gradients (detachments) in each objective to prevent the optimization of one component from affecting the other components.

The objective (3) defines the gradient for Ψ. In VQ-VAE we compute a gradient for (3) using the "straight-through" gradient for back-propagation through vector quantization. The VQ straight-through gradient can be written as

$$\nabla_x f(e(\text{nearest}_E(x))) \approx \nabla_e f(e)|_{e=e(\text{nearest}_E(x))}$$

(a) Give an `+=` equation for incorporating $e(\text{nearest}_E\ x)$.grad into x.grad.

Solution:

x.grad `+=` $e(\text{nearest}_E(x))$.grad.

(b) Write the SGD update equation for gradient descent on (4) using learning rate η.

Solution:

$$e(\text{nearest}_E(x_\Psi(y))) \;\texttt{+=}\; 2\eta(x_\Psi(y) - e(\text{nearest}_E(x_\Psi(y))))$$

(c) Assuming $\eta < 1/2$, rewrite your solution to (b) in the form of a rolling average update on $e(k)$ showing that $e(k)$ is a rolling average of the vectors of the form $x_\Psi(y)$ satisfying $\text{nearest}_E(x_\Psi(y)) = k$.

Solution: For $\text{nearest}_E(x_\Psi(y)) = k$ we have

$$e(k) = (1 - 2\eta)e(k) + 2\eta x_\Psi(y)$$

Problem 3: How Advantage-Actor-Critic (A2C) Reduces Variance. This problem will consider a simple artificial example that demonstrates the power of the advantage actor-critic algorithm. We start with policy gradient theorems for the episodic case.

$$\begin{aligned}
\text{REINFORCE}: \quad \nabla_\Phi R(\pi) &= E_{s_0,a_0,\ldots,s_T,a_T\sim\pi_\Phi} \sum_{t=0}^{T}(\nabla_\Phi \ln \pi_\Phi(a_t|s_t)) \left(\sum_{t'=t}^{T} R(s_t,a_t)\right) \\
\text{A2C}: \quad \nabla_\Phi R(\pi) &= E_{s_0,a_0,\ldots,s_T,a_T\sim\pi_\Phi} \sum_{t=0}^{T}(\nabla_\Phi \ln \pi_\Phi(a_t|s_t)) \, (Q^\pi(s_t,a_t) - V^\pi(s_t)) \\
V^\pi(s) &= E_{s_0,a_0,\ldots,s_T,a_T\sim\pi_\Phi \,|s_0=s,} \sum_{t=0}^{T} R(s_t,a_t) \\
Q^\pi(s,a) &= E_{s_0,a_0,\ldots,s_T,a_T\sim\pi_\Phi \,|s_0=s,\, a_0=a} \sum_{t=0}^{T} R(s_t,a_t)
\end{aligned}$$

In practice we use approximators $V_\Psi(s)$ and $Q_\Theta(s,a)$ for $V^\pi(s)$ and $Q^\pi(s,a)$. But we will ignore that here and just consider $V^\pi(s)$ and $Q^\pi(s,a)$ as defined above for which the above equations are exactly true.

We consider an MDP where we want to get to a goal state as quickly as possible. We consider an MDP where we have two actions a_1 and a_2 and a policy is just a biased coin flip between a_1 and a_2 independent of the state. We also suppose that a_1 always fails to advance, that a_2 always advances by one, and that we reach a goal as soon as we have advances N times. When we reach the goal, but only when we reach the goal, we get reward $-T$ (or equivalently, cost T) where T is the number of actions taken to reach the goal. Of course the best policy is to always pick a_2 which gives N advancements in N actions getting reward $-N$ (cost N). We define a state s to be the pair (i,t) where t is the number of actions taken so far and i is the number of advancements made so far.

(a) Define the state s_{t+1} as a function of s_t and a_t. The state transition is deterministic so don't worry about formulating this as a probability. Just say what the next state is in terms of the previous state and the action.

Solution: Let (i,t) be the state s_t. If the action is a_1 then $s_{t+1} = (i, t+1)$ and if the action is a_2 and $s_{t+1} = (i+1, t+1)$.

(b) If the stochastic policy picks a_2 with probability λ then the expected number actions taken to advance 1 step is $\sum_{t=1}^{\infty} \lambda(1-\lambda)^{t-1}t = \frac{1}{\lambda}$. Use this fact to give an expression for $V^\pi(i,t)$. (Remember that the reward, given only at the end, is $-T$).

Solution: From state (i,t) reward occurs when we have made $N-i$ additional advances. The expected time for each advance is $\frac{1}{\lambda}$. So the expected value of T given state (i,t) is $t + \frac{N-i}{\lambda}$ and so $V^\pi(i,t) = -(t + (N-i)/\lambda)$

(c) It can be shown that $Q^\pi(s,a) = R(s,a) + E_{s'|s,a}V^\pi(s')$. Use this and your result from (b) to give an analytic expressions for $Q^\pi(s,a)$ and the advantage $A^\pi(s,a) = Q^\pi(s,a) - V^\pi(i,t)$ for $a = a_1$ and $a = a_2$.

Solution: In this example, $R(s,a) = 0$ except at the end where $R((N-1,t),a_2) = -(t+1)$. So in all but the last step we have $Q^\pi(s,a) = V^\pi(s')$ where s' is the next state.

$$
\begin{aligned}
Q^\pi((i,t)a_1) &= V^\pi(i,t+1) = -(t+1+(N-i)/\lambda) \\
Q^\pi((i,t),a_2) &= V^\pi(i+1,t+1) = -(t+1+(N-i-1)/\lambda) \\
V^\pi(i,t) &= -(t+(N-i)/\lambda) \\
A^\pi((i,t,),a_1) &= -1 \\
A^\pi((i,t,),a_1) &= \frac{1}{\lambda} - 1
\end{aligned}
$$

It turns out that these equations handle the last step as well.

(d) Policy gradient adjusts the probability λ of selecting a_2. It is possible that for some samples of runs the REINFORCE algorithm decreases λ — it moves the policy in the wrong direction. For example, this happens when $\lambda = 1/2$ and there happens to be are more occurrences of a_2 than a_1 (note that the reward is always negative). But the reward (or cost) is correlated with the number of occurrences of a_1 so the expected update is still correct.

Given your answer to (c), is it possible that the A2C update ever reduces λ? Explain your answer.

Solution: In this example A2C always increases λ. The update on λ is the sum of the updates for each time step. In this case each time step behaves independently because the advantage is determined by the action taken at that time. At each time step we have that if the action selected is a_1 then the advantage is negative which decreases the probability of a_1 and hence increases the probability of a_2. When the action selected is a_2 the advantage is positive and the probability of a_2 is again increased.