TTIC 31230 Fundamentals of Deep Learning, Winter 2019
Final Exam

Problem 1. 25 points. Consider a population distribution Pop and subset S of its support. Let Pop_S be the restriction of Pop to the set S.

$$\text{Pop}_S(y) \quad = \quad \frac{1}{\text{Pop}(S)} \begin{cases} \text{Pop}(y) & \text{for } y \in S \\ 0 & \text{otherwise} \end{cases}$$

We also consider the Jensen-Shannon divergence

$$\text{JS}(P,Q) = \frac{1}{2}\left(\text{KL}\left(P, \frac{P+Q}{2}\right) + \text{KL}\left(Q, \frac{P+Q}{2}\right)\right)$$

Where for distributions P and Q we define $P+Q$ by the equation $(P+Q)(y) = P(y) + Q(y)$.

(a) 10 points. Solve for

$$\text{KL}\left(\text{Pop}_S, \frac{\text{Pop} + \text{Pop}_S}{2}\right)$$

in terms of the probability mass $\text{Pop}(S)$ of the set S. Your solution should have the form $\ln 2 - \ln(1+\epsilon)$ where ϵ is a function of $P(S)$ with $0 \le \epsilon \le 1$.

Solution:

$$\begin{aligned}
\text{KL}\left(\text{Pop}_S, \frac{\text{Pop} + \text{Pop}_S}{2}\right) &= E_{y\sim\text{Pop}_S} \ln \frac{2\text{Pop}_S(y)}{\text{Pop}(y) + \text{Pop}_S(y)} \\
&= E_{y\sim\text{Pop}_S} \ln \frac{\frac{2\text{Pop}(y)}{\text{Pop}(S)}}{\text{Pop}(y) + \frac{\text{Pop}(y)}{\text{Pop}(S)}} \\
&= \ln \frac{2}{\text{Pop}(S)+1} \\
&= \ln 2 - \ln(1+P(S))
\end{aligned}$$

(b) 5 points. Solve for

$$\text{KL}\left(\text{Pop}, \frac{\text{Pop} + \text{Pop}_S}{2}\right)$$

in terms of the probability mass $\text{Pop}(S)$ of the set S. Your solution should have the form $\ln 2 - \epsilon \ln(\epsilon+1)/\epsilon$ for ϵ a function of $P(S)$ with $0 \le \epsilon \le 1$. Hint: Write the KL-divergence as $P(S)E_{y\sim P(y|y\in S)}[\ldots] + (1-P(S))E_{y\sim\text{Pop}(y|y\notin S)}[\ldots]$.

Solution:

$$\begin{aligned}
& \mathrm{KL}\left(\mathrm{Pop}, \frac{\mathrm{Pop}+\mathrm{Pop}_S}{2}\right) \\
= \; & E_{y\sim \mathrm{Pop}} \ln \frac{2\mathrm{Pop}(y)}{\mathrm{Pop}(y)+\mathrm{Pop}_S(y)} \\
= \; & \mathrm{Pop}(S) E_{y\sim \mathrm{Pop}(y|y\in S)} \ln \frac{2\mathrm{Pop}(y)}{\mathrm{Pop}(y)+\frac{\mathrm{Pop}(y)}{\mathrm{Pop}(S)}} + (1-\mathrm{Pop}(S)) E_{y\sim \mathrm{Pop}(y|y\notin S)} \ln \frac{2\mathrm{Pop}(y)}{\mathrm{Pop}(y)} \\
= \; & \mathrm{Pop}(S) \ln \frac{2\mathrm{Pop}(S)}{\mathrm{Pop}(S)+1} + (1-\mathrm{Pop}(S)) \ln 2 \\
= \; & \ln 2 - \mathrm{Pop}(S) \ln \frac{\mathrm{Pop}(S)+1}{\mathrm{Pop}(S)}
\end{aligned}$$

(c) 5 points. For $\epsilon << 1$ we have $\ln(1+\epsilon) \approx \epsilon$ and $\ln(1+\epsilon)/\epsilon \approx \ln(1/\epsilon)$. Apply these approximation to get an approximate value for $\mathrm{JS}(\mathrm{Pop}_S, \mathrm{Pop})$ for small values of $P(S)$.

Solution:

$$\begin{aligned}
\mathrm{JS}(\mathrm{Pop}_S, \mathrm{Pop}, Q) &= \frac{1}{2}\left(\mathrm{KL}\left(\mathrm{Pop}_S, \frac{\mathrm{Pop}_S+\mathrm{Pop}}{2}\right) + \mathrm{KL}\left(\mathrm{Pop}, \frac{\mathrm{Pop}_S+\mathrm{Pop}}{2}\right)\right) \\
&\approx \frac{1}{2}\left(\ln 2 - \epsilon + \ln 2 - \epsilon \ln \frac{1}{\epsilon}\right) \\
&= \ln 2 - \frac{\epsilon}{2}\left(1 + \ln \frac{1}{\epsilon}\right) \\
\epsilon &= \mathrm{Pop}(S)
\end{aligned}$$

(d) 5 points. Given Your answer to (c), does the Jensen-Shannon divergence analysis of GANs indicate that the generator will suffer from collapse to a low-probability mode S. Explain your answer.

Solution: The analysis in part (c) indicates that any serious mode collapse will lead to a nearly maximal JS-divergence. So mode collapse should not occur. But the JS analysis is highly suspect — copying the training data is a serious mode collapse and would fool the discriminator.

As the real situation is unclear and any attempt to answer this receives full credit for thinking about it.

Problem 2. 25 points. This problem is on pseudo-likelihood. Consider a semantic segmentation $\hat{y}[i]$ on pixels i with $\hat{y}[i]$ a semantic class label in $\{C_1, \ldots, C_K\}$. We also assume a scoring function s_Φ on semantic segmentations defining

$$P_\Phi(\hat{y}) = \underset{\hat{y}}{\text{softmax}}\ s_\Phi(\hat{y})$$

Recall that pseudo-likelihood is defined by

$$\begin{aligned}\tilde{P}_\Phi(\hat{y}) &= \prod_i P_\Phi(\hat{y}[i]|\hat{y}\backslash i) \\ &= \prod_i \frac{P_\Phi(\hat{y})}{\sum_c P_\Phi(\hat{y}[i := c])}\end{aligned}$$

where $\hat{y}\backslash i$ assigns a class to every pixel other than i, and $\hat{y}[i := c]$ is the semantic segmentation identical to $\hat{y}$ except that pixel i is labeled with semantic class c.

(a) 10 points. show

$$\frac{P_\Phi(\hat{y})}{\sum_c P_\Phi(\hat{y}[i := c])} = (\underset{c}{\text{softmax}}\, s_\Phi(\hat{y}[i := c])[\hat{y}[i]]$$

Solution:

$$\begin{aligned}\frac{P_\Phi(\hat{y})}{\sum_c P_\Phi(\hat{y}[i := c])} &= \frac{\frac{1}{Z} e^{s_\Phi(\hat{y})}}{\sum_c \frac{1}{Z} e^{s_\Phi(\hat{y}[i:=c])}} \\ &= \frac{e^{s_\Phi(\hat{y})}}{\sum_c e^{s_\Phi(\hat{y}[i:=c])}} \\ &= (\underset{c}{\text{softmax}}\ s_\Phi(\hat{y}[i := c]))[\hat{y}[i]]\end{aligned}$$

(b) 5 points. How many scores need to be computed in the worst case for computing $P_\Phi(\hat{y})$. Given the result of part (a), how many for computing $\tilde{P}_\Phi(\hat{y})$?

Solution: K^N for P_Φ and KN for $\tilde{P}_\Phi$.

(c) 5 points. Consider a distribution on semantic segmentations where for each pixel the class assigned to that pixel is determined by the other pixels. Can this distribution be defined by a softmax over scores? Explain your answer.

Solution: No. Since $e^s > 0$ for any finite s, all semantic segmentations must have nonzero probability.

(d) 5 points. If P_Φ is a distribution defined in some other way such that the class of each pixel is completely determined by the other pixels, given a simple expression for $\tilde{P}_\Phi(\hat{y})$ in the case where $\hat{y}$ has nonzero probability under P_Φ.

Solution: We have $P_\Phi(\hat{y}|\hat{y}\backslash i) = 1$ which implies $\tilde{P}(\hat{y}) = 1$.

Problem 3. 25 Points: This problem is on the transformer.

Background: The Transformer computes layers of sequences of vectors $M[\ell, t, j]$ where ℓ ranger over layers, t ranges over "time" (the index into the sequence), and j is the index of a particular dimension of the vector at layer ℓ and time t.

We also let h range over "heads" — each head is computing an attention for a different purpose.

We compute $M[\ell + 1, T, J]$ from $M[\ell, T, J]$ as follows:

$$\begin{aligned}
\text{Query}[\ell, h, t, k] &= \sum_j W^Q[\ell, h, k, j] M[\ell, t, j] \\
\text{Key}[\ell, h, t, k] &= \sum_j W^K[\ell, h, k, j] M[\ell, t, j] \\
\text{Value}[\ell, h, t, i] &= \sum_j W^V[\ell, h, i, j] M[\ell, t, j] \\
s[\ell, h, t, t'] &= \frac{1}{\sqrt{K}} \sum_k \text{Query}[\ell, h, t, k] \text{Key}[\ell, h, t', k] \\
\alpha[\ell, h, t, t'] &= \underset{t'}{\text{softmax}}\ s[\ell, h, t, t'] \quad \text{the attention} \\
V[\ell, h, t, i] &= \sum_{t'} \alpha[\ell, h, t, t'] \text{Value}[\ell, h, t', i]
\end{aligned}$$

$$M[\ell + 1, t, J] = V[\ell, 1, t, I]; V[\ell, 2, t, I]; \cdots; V[\ell, H, t, I] \quad J = HI$$

(a) 15 points. Just as CNNs can be done in two dimensions for vision and in one dimension for language, the Transformer can be done in two dimensions for vision — the so-called spatial transformer. Rewrite the above so as to define a spatial transformer on images with attentions computes between spatial locations.

Solution:

$$
\begin{aligned}
\text{Query}[\ell,h,x,y,k] &= \sum_j W^Q[\ell,h,k,j]M[\ell,x,y,j] \\
\text{Key}[\ell,h,x,y,k] &= \sum_j W^K[\ell,h,k,j]M[\ell,x,y,j] \\
\text{Value}[\ell,h,x,y,i] &= \sum_j W^V[\ell,h,i,j]M[\ell,t,j] \\
s[\ell,h,x,y,x',y'] &= \frac{1}{\sqrt{K}}\sum_k \text{Query}[\ell,h,x,y,k]\text{Key}[\ell,h,x',y',k] \\
\alpha[\ell,h,x,y,x',y'] &= \underset{x',y'}{\text{softmax}}\ s[\ell,h,x,y,x',y''] \quad \text{the attention} \\
V[\ell,h,x,y,i] &= \sum_{x',y'} \alpha[\ell,h,x,y,x',y']\text{Value}[\ell,h,x',y',i]
\end{aligned}
$$

$$
M[\ell+1,x,y,J] = V[\ell,1,x,y,I]; V[\ell,2,x,y,I]; \cdots; V[\ell,H,x,y,I] \quad J = HI
$$

(b) 10 points. Assuming that summations can be computed in logarithmic time in parallel, what is the parallel order of run time of the spatial transformer as a function of the dimensions L, I, J, K, X and Y.

Solution: $O(L \log JKXY)$

Problem 4. 25 points. This problem is on AlphaZero.

Backgound. A version of AlphaZero can be defined as follows.

To select a move in the game, first construct a search tree over possible moves to evaluate options. The tree is grown by running "simulations". In descending into the tree, simulations select the move $\text{argmax}_a\, U(s,a)$ where we have

$$
U(s,a) = \begin{cases} \lambda_u\ \pi_\Phi(s,a) & \text{if } N(s,a) = 0 \\ \hat{\mu}(s,a) + \lambda_u\ \pi_\Phi(s,a)/N(s,a) & \text{otherwise} \end{cases} \qquad (1)
$$

where $N(s,a)$ is the number of simulations that reached state s and then selected action a from there.

When the search is completed, we select an action from the root position. For this we use a post-search stochastic policy

$$
\pi_{s_{\text{root}}}(a) \propto N(s_{\text{root}},a)^\beta \qquad (2)
$$

where β is temperature hyperparameter.

The value function $V_\Phi(s)$ and the policy $\pi_\Phi(a|s)$ are trained from a replay buffer containing triples (s, π, R) where s is a position from which a tree search was done, π is the root probability distribution over actions computed as above from the tree search, and R is the final reward of the game in which this position occurred. The parameters Φ are trained by SGD on the following objective defined on the replay buffer.

$$\Phi^* \;=\; \underset{\Phi}{\text{argmin}}\; E_{(s,\pi,R)\sim\text{Replay},\; a\sim\pi} \begin{pmatrix} (V_\Phi(s) - R)^2 \\ -\lambda_\pi \;\log \pi_\Phi(a|s) \\ +\lambda_R \;||\Phi||^2 \end{pmatrix} \qquad (2)$$

(a) 20point. Rewrite (1) and (2) above to replace the policy $\pi_\Phi(a|s)$ with an advantage function $A_\Phi(s, a)$ so that the expected value of an action a in state s (the Q-value) is modeled by $V_\Phi(s) + A(s, a)$.

Solution:

$$U(s,a) = \begin{cases} \lambda_u \;(V_\Phi(s) + A_\Phi(s,a)) & \text{if } N(s,a) = 0 \\ \hat{\mu}(s,a) + \lambda_u \;(V_\Phi(s) + A_\Phi(s,a))/N(s,a) & \text{otherwise} \end{cases} \qquad (1)$$

or

$$U(s,a) = \begin{cases} V_\Phi(s) + A_\Phi(s,a) + \lambda_u & \text{if } N(s,a) = 0 \\ \hat{\mu}(s,a) + (V_\Phi(s) + A_\Phi(s,a) + \lambda_u)/N(s,a) & \text{otherwise} \end{cases} \qquad (1)$$

$$\Phi^* \;=\; \underset{\Phi}{\text{argmin}}\; E_{(s,\pi,R)\sim\text{Replay},\; a\sim\pi} \begin{pmatrix} (V_\Phi(s) - R)^2 \\ \lambda_A \;(A_\Phi(s,a) - (R - V_\Phi(s))^2 \\ +\lambda_R \;||\Phi||^2 \end{pmatrix} \qquad (2)$$

(b) 5 points: Will the training procedure you defined in (a) effectively train $A_\Phi(s, a)$ for off-policy actions a? If not, is that a problem and does this problem arise in the original version in terms of $\pi_\Phi(a|s)$? Explain your answer.

Solution: The solution to (a) does not train $A_\Phi(s, a)$ on off-policy actions. However, the original formulation in terms of the policy $\pi_\Phi(a|s)$ drives down the probability of off-policy actions so as to place large probability on the on-policy actions. This would seem to be more effective.

But any answer gets full credit for thinking about it.