1	2	3	4	Total

CS 726: Advanced Machine Learning, Spring 2019, End-Semester exam

May 1, 2019. 5:30 to 8:30pm

Roll: _____

Name: _____

This exam is open notes.

---

1. Here we will learn the parameters of a Bernoulli distributed random variable y with a single parameter θ denoting the probability of 1. Let reward of a y be $R(y) = y\alpha$ where $\alpha > 0$ and $y \in \{0, 1\}$. We draw M samples of y from each of the following three algorithms for RL training at a time t when $\theta = \theta^t = [0.3]$. Let S_1 denote the number of ones in these M samples. For each of the following methods write the expression for the expected value of S_1, the training objective for updated parameter in terms of S_1, and the updated value of parameters θ^{t+1} assuming M is very large.

   (a) Policy gradient

      i. Expected value of S_1 ..1 The M samples will be obtained in this case from the Bernoulli with probability as θ^t. The expected size of S_1 is 0.3M.

      ii. Training objective and the θ at which it is maximized ..2

      The training objective becomes $S_1 \alpha \log(\theta)$. This is maximized with $\theta = 1$.

   (b) Reward Augmented Maximum Likelihood with $\tau = 1$.

      i. Expected value of S_1 ..2 The M samples in this case will be obtained from a Bernoulli with probability of one as $\frac{e^\alpha}{1+e^\alpha}$ which makes expected size of S_1 as $\frac{e^\alpha}{1+e^\alpha} M$

      ii. Training objective and the θ at which it is maximized ..2

      The training objective becomes: $\frac{e^\alpha}{1+e^\alpha} \log \theta + \frac{1}{1+e^\alpha} \log(1-\theta)$ The optimal for this is to set $\theta = \frac{e^\alpha}{1+e^\alpha}$.

   (c) Softmax policy gradient

      i. Expected value of S_1 ..2 The M samples in this case will be obtained from $\exp^{R(y)/\tau} P_{\theta^0}(y)$ which evaluates to $\frac{e^\alpha 0.3}{e^\alpha 0.3 + 0.7}$. and S_1 is just M times this number.

      ii. Training objective and the θ at which it is maximized ..2 The training objective then becomes $\frac{e^\alpha 0.3}{e^\alpha 0.3+0.7} \log \theta + (1 - \frac{e^\alpha 0.3}{e^\alpha 0.3+0.7}) \log(1-\theta)$.
      The optimal solution for this is $\theta = \frac{e^\alpha 0.3}{e^\alpha 0.3+0.7}$

2. We are estimating a 1-D regression function $f(x)$ as a Gaussian Process $GP(m(x), \kappa(x, x'))$ where the kernel function $\kappa(x, x') = \sigma_f^2 \exp(-\frac{(x-x')^2}{2\tau})$, $m(x) = 0$. Each $y_i = f(x_i) + \epsilon_i$ where $\epsilon_i \sim \mathcal{N}(0, \sigma^2)$ where σ^2 denotes the variance of the noise ϵ_i. Let the training data be $D = \{(x_1, y_1), \ldots, (x_N, y_N)\}$. Express briefly in qualitative terms the shape of the mean of $f(x|D)$ posterior to seeing the training data D in terms of the following properties:

   (a) Noise σ^2 is increased from zero to a large value. ..2 When noise is zero, the f will go through the training points and when σ^2 is large and tending towards infinity the training points will have no effect, and the prior f will be the same as the posterior.

(b) Length scale τ increased from a small value (say 0.0001) to a large value (say 1000). ..3 When τ is small, inter point distances blows up causing their covariance to be small, when τ is increased inter-point distances collapse causing their co-variance to be large. This will make the f curve go from being jerky to being smooth.

3. In class we studied models for predicting the time of the next event using recurrent marked temporal point processes (RMTPP). The RMTPP uses a RNN to fit non-homogenous intensity for the time of the next event, conditioned on times $\mathcal{H} = t_1, \ldots t_I$ of the first I events

$$\lambda^*(t) = \exp(V.h(t_I) + w(t - t_I) + b) \qquad f(t|\mathcal{H}) = \lambda^*(t) \exp(-\int_{t_I}^{t} \lambda^*(t)) \tag{1}$$

where $h(t_I)$ denotes the RNN state after inputing times $t_1, \ldots t_I$ and $f(t|\mathcal{H})$ denotes the density function for the time t of the next event given history $\mathcal{H}$.

In this question we will extend the model to predict the missing time t_i of any sequence of n events given the time of the remaining events $\mathcal{H}_{-i} = t_1, \ldots t_{i-1}, t_{i+1}, \ldots, t_n$.

(a) What is wrong with using the density $f(t|\mathcal{H}_{i-1})$ for calculating the density of t_i. Note in this case the RNN is fed only $t_1, \ldots, t_{i-1}$? Illustrate with an example giving specific values for necessary model parameters and the event times. ..3
The density should be conditioned on both the events prior to i and after i, otherwise there is no guarantee that the expected time will be less than t_{i+1}. Let the values of V, w is zero and $b = 1$. Then we have a homogeneous poisson process with expect time of next event as $t_{i-1} + 1$. If $t_{i+1} = t_{i-1} + 0.5$ we have an invalid expected value.

(b) Give a different inference method of estimating an expected value for t_i using the above trained model? Your solution should not require any parameter retraining. ..3
We will have to calculate the conditional probability $f(t|\mathcal{H}_{-i})$ This can be written by applying Bayes rule as
$f(t|\mathcal{H}_{-i}) = \frac{f(t|\mathcal{H}_{i-1}) \prod_{j=i+1}^{n} f(t_j|\mathcal{H}_{t_1 \ldots t_{i-1}, t, t_{i+1}, \ldots, t_{j-1}})}{\int_{t'=t_{i-1}}^{t'=t_{i+1}} f(t'|\mathcal{H}_{i-1}) \prod_{j=i+1}^{n} f(t_j|\mathcal{H}_{t_1 \ldots t_{i-1}, t', t_{i+1}, \ldots, t_{j-1}})}$

(c) Now suppose you are allowed to train two sets of RNNs. How would you design them, so that such missing event times can be estimated accurately and efficiently? ..3 We have the usual forward RNN that is fed $t_1, \ldots, t_{i-1}$, but we use train a second backward RNN that is fed $t_n, \ldots, t_{i+1}$. Let $g(t_{i+1})$ denote the state of the backward RNN after that. We next design an intensity function as

$$\lambda_i^*(t) = \exp(V.h(t_{i-1}) + U.g(t_{i+1}) + w(t - t_{i-1}) + u(t_{i+1} - t) + b)$$

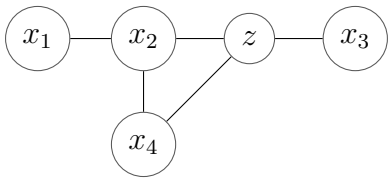
Thereafter the density is defined using the same method, but we add an additional normalizer so the range between $[t_{i-1}, t_{i+1}]$ integrates to 1.

4. We will use a MCMC sampler to do a specific kind of constrained Gibbs sampling. Let $P_G(x_1, \ldots, x_n, z)$ be a joint distribution over $n + 1$ variables implemented as a graphical model. You are given an initial sample $x_1^0, x_2^0, \ldots, x_n^0$ only over the first n variables, that is, value of z is hidden.

(a) Design a Gibbs sampling step $T(\mathbf{x}|\mathbf{x}')$ that can be used to generate new samples from the distribution $\sum_z P_G(z|\mathbf{x}^0) P_G(\mathbf{x}|z)$. ..2 In this case,

we just blindly apply the theory of designing a Gibbs sampling step for a distribution $P(\mathbf{x})$ which in this case happens to be $\sum_z P_G(z|\mathbf{x}^0)P_G(\mathbf{x}|z)$.
For an arbitrary $P(\mathbf{y})$, the Gibbs sampler under the case $\mathbf{y} \neq \mathbf{y}', \mathbf{y}_{-i} = \mathbf{y}'_{-i}$ is
$T(\mathbf{y}|\mathbf{y}') = \frac{1}{n}P(y_i|\mathbf{y}'_{-i}) = \frac{P(y_i,\mathbf{y}'_{-i})}{n\sum_y P(y,\mathbf{y}'_{-i})}$.
Substituting the formula for $P(\mathbf{y})$ in the above,
$T(\mathbf{x}|\mathbf{x}') = \frac{\sum_z P_G(z|\mathbf{x}^0)P_G(x_i,\mathbf{x}'_{-i},z)}{n\sum_z\sum_x P_G(z|\mathbf{x}^0)P_G(x,\mathbf{x}'_{-i},z)}$
Strictly speaking this solution is not complete because it does not cover the case where $\mathbf{x} = \mathbf{x}'$. But, we are awarding full marks to the above solution. In a similar vein one can write the solution for $\mathbf{x} = \mathbf{x}'$

(b) In the graphical model below, assume binary variables and potentials as $\psi_{1,2}(x_1, x_2)$, $\psi_{2.4}(x_2, z, x_4)$, $\psi_{3.}(x_3, z)$.



If we are doing MCMC sampling and the starting sample is $[x_1^0, \ldots, x_4^0] = [1, 1, 1, 1]$, calculate the value of $T(\mathbf{x}|\mathbf{x}')$ for transitioning from $\mathbf{x}' = [0, 0, 0, 0]$ to $\mathbf{x} = [0, 0, 0, 1]$ ..3

Here we just plug in the definition of $P(z|\mathbf{x})$ and $P(\mathbf{x}, z)$ in the above solution.
$P(z|\mathbf{x}^0) \propto \psi_{2.4}(1, z, 1)\psi_{3.}(1, z)$. $P_G(x, \mathbf{x}'_{-i}, z) \propto \psi_{1,2}(0, 0)\psi_{2.4}(0, z, x)\psi_{3.}(0, z)$. Plugging in these values we get: $T(\mathbf{x}|\mathbf{x}') = \frac{\sum_z \psi_{2.4}(1,z,1)\psi_{3.}(1,z)\psi_{1,2}(0,0)\psi_{2.4}(0,z,1)\psi_{3.}(0,z)}{4\sum_x\sum_z \psi_{2.4}(1,z,1)\psi_{3.}(1,z)\psi_{1,2}(0,0)\psi_{2.4}(0,z,x)\psi_{3.}(0,z)}$
The $\psi_{1,2}(0, 0)$ terms can be cancelled out.

5. Consider a graph with a single loop X_1—X_2—$\ldots X_n$—X_1.

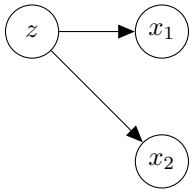
(a) Draw the junction tree for this graph. ..2 We need to triangulate the loop first. Thereafter, the JT is obvious.

(b) Suppose we have already computed messages on all edges on the JT, and now we want to compute $\Pr(X_i = 1|X_j = 0)$. Show what messages can be reused and what messages have to be computed afresh for the above query (assume $j > i$) ..2 Will depend on the specific JT drawn. But assuming the default linear JT where each node is a triangle, we will only need to send messages from the clique node C containing x_j to the clique node C' containing x_i. The other incoming messages onto C and C' can be reused.

(c) In the worst case, how much work do you have to redo for the above query? ..2 Worst case: the entire JT width has to be recomputed $O(n)$ where

(d) Give the smallest possible triangulated graph which has two different junction trees. ..3 The set of clique nodes will be unique. But different sets of edges are possible For example, we could have a star graph with a central x_1 to which x_2, x_3, x_4 are connected. The JT could be
x_1x_2 — x_1x_3 — x_1x_4 or
x_1x_2 — x_1x_4 — x_1x_3

6. In this question we will estimate the parameters of a hidden variable model using EM, VAE, and GANs. The Bayesian network denoting the dependence among three variables is shown below.



Joint distribution among variable pairs is:

$$P(x_1, z) = \mathcal{N}([\mu_1, \mu_z], \begin{bmatrix} 1 & \rho \\ \rho & 1 \end{bmatrix})$$

$$P(x_2, z) = \mathcal{N}([\mu_2, \mu_z], \begin{bmatrix} 1 & \rho \\ \rho & 1 \end{bmatrix})$$

For training the parameters $\theta = [\mu_1 \mu_2, \mu_z, \rho]$, we are given two types of data samples: $D_1 = \{x_1^1, \ldots, x_1^N\}$ where both x_2 and z are missing, and $D_2 = \{x_2^1, \ldots, x_2^N\}$ where x_1, z are missing.

(a) Write the formula for the following distributions: $P(z), P(x_1|z), P(z|x_1)$ [Hint: Use the formula for converting joint Gaussians to conditional Gaussians discussed during the Gaussian Processes lecture.] ..2

$$P([\mathbf{x}_A, \mathbf{x}_B]') = \mathcal{N}([\mu_A, \mu_B]', \begin{bmatrix} \Sigma_{AA} & \Sigma_{AB} \\ \Sigma_{BA} & \Sigma_{BB} \end{bmatrix})$$

$$P(\mathbf{x}_A|\mathbf{x}_B = x_b) = \mathcal{N}(\mu_A + \Sigma_{AB}\Sigma_{BB}^{-1}(x_b - \mu_b), \quad \Sigma_{AA} - \Sigma_{AB}\Sigma_{BB}^{-1}\Sigma_{BA})$$

This gives us that $P(z) = \mathcal{N}(\mu_z, 1), P(x_1|z) = \mathcal{N}(\mu_1 + \rho(z - \mu_z), 1 - \rho^2), P(z|x_1) = \mathcal{N}(\mu_z + \rho(x - \mu_1), 1 - \rho^2)$

(b) Now we apply EM to estimate θ. Let the initial values of parameters be $\theta^0 = [\mu_1^0, \mu_2^0, \mu_z^0, \rho^0]$. Estimate the next value μ_z^1 using the EM algorithm, and calculate the exact value in closed form for the case where $D_1 = \{0, -1, -2\}, D_2 = \{2, 3, 1\}$ and $\theta^0 = [-1, 3, 1, 0.2]$ ..5 Using the formula above we estimate $P(z|x_1^i, \theta^0)$ and $P(z|x_2^i, \theta^0)$ for $i = 1, \ldots, N$. That gives $2N$ Gaussian distributions for z in the E-step.

In the M-step, the expected sum of the means of these Gaussians is the updated estimate of $\mu_z^1 = \mu_z^0 + \rho^0/2(\sum_i x_1^i/N - \mu_1^0) + \rho^0/2(\sum_i x_2^i/N - \mu_2^0)$ We can substitute the values above to get the desired answer.

(c) Next, we use VAEs. In normal VAE discussed in class we assume that each data instance has both attributes x_1, x_2 which we jointly called $\mathbf{x}$ and used that to estimate the posterior $q(z|\mathbf{x})$. In this case, we have x_1 and x_2 in separate samples so we need to encode them separately as $q_1(z|x_1)$ and $q_2(z|x_2)$. Use $G_1(z)$ to denote the decoder network that generates x_1 and likewise $G_2(z)$ for x_2. Provide the training objective that can be used for this VAE. ..2

$\min_{G_1,G_2,q_1,q_1} \sum_{j=1}^{2} \sum_i E_{z\sim q_j(z|x_j^i)} \log G_J(x_j^i|z) - KL(q_j(z|x_j^i)||p(z))$

(d) Next we bring in GANs. We add to the above design two discriminators: $\mathcal{D}_1(x)$ that can discriminate between real x_1 samples and generated samples $G_1(z)$ and likewise $\mathcal{D}_2(x)$. Write the full objective function for augmenting your VAE above with adversarial training on the GANs. ..3

$$\min_{G_1,G_2,q_1,q_1} \Big(\sum_{j=1}^{2} \sum_i E_{z\sim q_j(z|x_j^i)} \log G_J(x_j^i|z) - KL(q_j(z|x_j^i)||p(z)) + \max_{\mathcal{D}_2} \sum_{i\in D_2} \log \mathcal{D}_2(x_2^i) + \sum_{i\in D_1} E_{z\sim q_1(z|x_1^i)} \log(1 - \mathcal{D}_2(G_2(z))) + \max_{\mathcal{D}_1} \sum_{i\in D_1} \log \mathcal{D}_1(x_1^i) + \sum_{i\in D_2} E_{z\sim q_2(z|x_2^i)} \log(1 - \mathcal{D}_1(G_1(z))) \Big)$$

7. Consider a neural translation model using the encoder-decoder network discussed in class. In this model, the probability of any output sequence $\mathbf{y}$ is factorized as: $\prod_{t=1}^{n} \Pr(y_t|\mathbf{x}, y_1, \ldots, y_{t-1})$ which is then computed using the decoder. Let the decoder be a transformer-based self-attention model. During inference our goal is find the sequence $\mathbf{y}^* = \text{argmax}_{\mathbf{y}} \prod_{t=1}^{n} \Pr(y_t|\mathbf{x}, y_1, \ldots, y_{t-1})$. Let m denote the size of the vocabulary of any y_t.

   (a) If you use the beam-search algorithm to find $\mathbf{y}^*$ with a beam-width (B) equal to the size of the vocabulary (m) of y_t, are you guaranteed to find the optimal $\mathbf{y}^*$. Justify with a brief proof or give a counter-example. ..3 No not guaranteed. Say $m = 2$. Consider sequences of length 3. Let the various probabilities be: $P(y_1 = 1) = 0.6, P(y_2 = 1|y_1 = 1) = 0.6, P(y_2 = 1|y_1 = 0) = 0.501, P(y_3 = 1|y_1 = 0, y_2 =?) = 1, P(y_3 = 1|y_1 = 1, y_2 =?) = 0.5$
   Optimal is $[y_1 = 0, y_2 = 1, y_3 = 1]$ with probability 0.4*0.501*1 whereas beam-search will find the two highest scoring prefixes of length 2 as $[y_1 = 1, y_2 = 1], [y_1 = 1, y_2 = 0]$. This best completion of these two is $[1, 1, 1]$ which has a probability of 0.6*0.6*0.5

   (b) Suppose we use self-attention to find at each time-step t, one previous variable $a_t \in \{1, \ldots, t-1\}$ such that $\Pr(y_t|\mathbf{x}, y_1, \ldots, y_{t-1}) = \Pr(y_t|\mathbf{x}, y_{a_t})$. Propose a more accurate alternative to Beam-search to find $\mathbf{y}^*$. [Express your idea in 2–4 sentences. Detailed algorithm not required.] ..3 The inference task becomes tractable in this case using graphical model message passing. We keep around for each j and each possible label y, the maximum probability prefix ending with label y and position t. Let m_{jy} denote that probability. At a future time t if it depends on position a_t we just need to find $\max_{y'} m_{a_t y'} \Pr(y_t = y|y_{a_t} = y')$ for each y_t and save m_{ty}. The time required is just nm^2.

   (c) Can you convert the $\mathbf{y}^* = \text{argmax}_{\mathbf{y}} \prod_{t=1}^{n} \Pr(y_t|\mathbf{x}, y_1, \ldots, y_{t-1})$ to a continuous optimization problem? [Hint: Recall the technique used in the SPEN paper.] ..3 For each y_t we will allocate m variables $l_1^t, \ldots, l_m^t$ using which we approximate a discrete $y_t = y$ as a continuous one using $\frac{e^{l_y^t}}{\sum_j e^{l_j^t}}$. Next, we convert the maximization problem as $\text{argmax}_{l_j^t : t=1,\ldots,n, j=1,\ldots,m} \prod_t \sum_j \Pr(y_t = j|\mathbf{x}, \mathbf{l}^1, \ldots, \mathbf{l}^{t-1}) l_j^t$ For conditioning on the $\mathbf{l}^t$ we feed to the decoder the averaged embeddings calculated as $\sum_j \mathbf{v}_j \frac{e^{l_y^t}}{\sum_j e^{l_j^t}}$ where $\mathbf{v}_j$ denotes the embedding of word j.

Total: 60