CS181 Midterm 2 Practice Solutions

1. **Convergence of K-Means**

Consider Lloyd's algorithm for finding a K-Means clustering of N data, i.e., minimizing the "distortion measure" objective function

$$J(\{\boldsymbol{r}_n\}_{n=1}^{N}, \{\boldsymbol{\mu}_k\}_{k=1}^{K}) = \sum_{n=1}^{N}\sum_{k=1}^{K} r_{nk}||\boldsymbol{x}_n - \boldsymbol{\mu}_k||_2^2.$$

Show that as a consequence of there being a finite number of possible assignments for the set of responsibilities $r_{n,k}$, and that for each such assignment there is a unique optimum for the means $\{\boldsymbol{\mu}_k\}_{k=1}^{K}$, the K-Means algorithm must converge after a finite number of iterations.

Since both the responsibility updates and the mean updates minimize the K-Means objective function, Lloyd's algorithm will never move to a new configuration of responsibility assignments unless the new configuration has a lower objective value. Since there are a finite number of possible assignments, each with a corresponding unique minimum with regard to the $\{\boldsymbol{\mu}_k\}_{k=1}^{K}$, the K-Means algorithm will converge after a finite number of steps, when no changes to the responsibilities will decrease the objective. When the responsibilities don't change in an iteration, the $\{\boldsymbol{\mu}_k\}_{k=1}^{K}$ also don't change.

2. **K-Means++**

One way to initialize Lloyd's algorithm for K-Means is to randomly select some of the data to be the first cluster centers. The easiest version of this would pick uniformly from among the data. K-Means++ biases this distribution so that it is not uniform. Explain in words how the distribution is non-uniform and why it should lead to better initializations.

The K-Means++ algorithm iteratively adds cluster centers, drawing them from a distribution over the data. This distribution is proportional to the squared distance of each datum from its nearest cluster center. Thus K-Means++ tends to favor points that are distant from the existing centers and produce a more diverse set of centers.

3. **Standardizing Input Data**

   Standardizing data helps ensure that distances makes sense and that the different properties of the items are balanced. Give an example of a kind of data for which standardization might be necessary to get good results from K-Means clustering.

   Any example with features that have different units:

   - Clustering galaxies by size and brightness.
   - Clustering fruit by color and weight.
   - Clustering houses by square footage and price.

4. **K-Medoids**

   $\boxed{\text{K-Medoids clustering is similar to K-Means, except that it requires the cluster centers to be data examples. Describe a situation in which this is desirable or necessary.}}$

   Sometimes we only have distances between points, and averages of points are not sensible or available. In such cases, we cannot describe a cluster by a mean of data and instead describe it by an “exemplar”.

5. **Value Iteration**

   Say an MDP has state space S and reward R where all rewards are positive. If you run value iteration, what is the largest k for which $V_k(s)$ is zero?

   $k = 0$. $V_0(s) = 0$ for all states s. Since $V_1(s) = max\{r_1, r_2...\}$ and all rewards are positive, all $V_k(s) > 0$ when $k > 0$.

6. **Infinite Horizon**

You are on a linear space and can move only right or left. Each position has reward r_i and $\gamma \approx 1$. Describe your optimal policy given any state i (don't forget about ties).

You always move towards state i that has $r_i(s) = max\{R\}$ and stay there forever. When there are ties, you move to the closest state i and then stay there. "Closest" is calculated as the state k with max $\sum r_j$ where $j \in [i, i+1, ..., k]$.

7. **Value Iteration vs Expectimax Search**

What is the running time of Expectimax Search and Value Iteration as a function of the horizon, T, the number of actions, M, the the number of transitions for any state and action, L, and the number of steps, N? Why don't we always choose the algorithm with better asymptotic running time?

Expectimax Search takes time proportional to the height of the game tree (see Figure 3 in the lecture notes on MDP's), so we see that the running time is $O((ML)^T)$. Value Iteration uses dynamic programming to ensure that it doesn't revisit the same state multiple times, and we get running time linear in the inputs $O(NMLT)$. Although value iteration is a linear-time algorithm, it visits states that might not actually be reachable, while Expectimax only runs computations on states that can be reached. For some problem domains, this results in higher efficiency.

8. **Setting Rewards to Compute Shortest Path**

Suppose we have a grid-world where each state is represented as a point in $\{(x, y) | 0 \leq x \leq 4, 0 \leq y \leq 4\}$. Suppose you have a robot that starts in the lower left corner and is given a goal point that is needs to reach. At each state, the robot can move one point to the left, right, up, or down, and each action has a 90% chance of success (otherwise you stay at the current point).

- What is the size of the state space in the resulting MDP?
- Suppose we want to minimize the length of path to the goal, but we have no preference for which path the robot should take (and all points are reachable). What rewards could we assign to each state so that we recover a shortest path to the goal as our optimal policy?

- Our grid is composed of each point in a 5×5 grid, so we have 25 possible states in the state-space.
- We can set the reward of each state, new state pair (that does not end with the goal node) to be the same negative value. Then, when we try to maximize our reward we will choose the shortest path, and since each transition has the same reward, we do not preferentially choose any paths with the same length.

9. **Bounds**

Show that as a consequence of the constraint $0 \leq p(x_n|\mu_k) \leq 1$ for the discrete variable x_n, the incomplete-data log likelihood function for a mixture of Bernoulli distributions is bounded above, and hence that there are no singularities for which the likelihood goes to infinity.

The result follows from (Bishop 9.51). The largest value that the argument to the logarithm on the r.h.s. of (9.51) can have is 1, since $\forall n, k : 0 \leq p(x_n|\mu_k) \leq 1$, $0 \leq \pi_k \leq 1$ and $\sum_k^K \pi_k = 1$. Thus the maximum value for $\ln p(X|\mu, \pi)$ equals 0.

10. **Gaussian**

Consider a Gaussian mixture model in which the marginal distribution $p(\boldsymbol{z})$ for the latent variable is given by

$$p(\boldsymbol{z}) = \prod_{k=1}^{K} \pi_k^{z_k},$$

and the conditional distribution $p(\boldsymbol{x} \mid \boldsymbol{z})$ for the observed variable is given by

$$p(\boldsymbol{x} \mid \boldsymbol{z}) = \prod_{k=1}^{K} \mathcal{N}(\boldsymbol{x} \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)^{z_k}.$$

Show that the marginal distribution $p(\boldsymbol{x})$, obtained by summing $p(\boldsymbol{z})p(\boldsymbol{x} \mid \boldsymbol{z})$ over all possible values of $\boldsymbol{z}$, is a Gaussian mixture of the form

$$p(\boldsymbol{x}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\boldsymbol{x} \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k).$$

From Bishop (9.10) and (9.11), we have

$$p(x) = \sum p(x|z)p(z) = \sum_z \prod_{k=1}^{K} (\pi_k N(x|\mu_k, \Sigma_k))^{z_k}$$

We use the 1-of-K representation for z:

$$= \sum_z \prod_{k=1}^{K} (\pi_k N(x|\mu_k, \Sigma_k))^{I_{kj}} = \sum_{j=1}^{K} \pi_j N(x|\mu_j, \Sigma_j)$$

Note: $I_{kj} = 1$ when $k = j$ and 0 otherwise.

11. **EM and Mixture Models**

Show that if we maximize

$$\mathbb{E}_z[\ln p(\boldsymbol{X}, \boldsymbol{Z} \mid \boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma})] = \sum_{n=1}^{N} \sum_{k=1}^{K} \gamma(z_{nk}) \left\{ \ln \pi_k + \ln \mathcal{N}(\boldsymbol{x}_n \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \right\}$$

with respect to $\boldsymbol{\mu}_k$ while keeping the responsibilities $\gamma(z_{nk})$ fixed, we obtain the closed form solution given by

$$\boldsymbol{\mu}_k = \frac{\sum_{n=1}^{N} \gamma(z_{nk}) \boldsymbol{x}_n}{\sum_{n=1}^{N} \gamma(z_{nk})} \,.$$

We write the r.h.s. of (9.40) as

$$-\frac{1}{2} \sum_{n=1}^{N} \sum_{j=1}^{K} \gamma(z_{nj})(x_n - \mu_j)^T \Sigma^{-1} (x_n - \mu_j) + c$$

c represents terms independent of mu_j. We take derivative wrt μ_k:

$$-\sum_{n=1}^{N} \gamma(z_{nk})(\Sigma^{-1}\mu_k - \Sigma^{-1} x_n)$$

Set this to zero and rearrange to get (9.17).

12. **Belief State Methods**

In belief state methods for POMDPs, the agent has to maintain a probability distribution over the current state of the world.

$$b(s_t) = P(s_t | o_1, , o_t, a_1, , a_{t-1})$$

Express $b(s_t)$ as a function of $b(s_{t-1})$.

$$\begin{aligned}
b(s_t) &= P(s_t \mid o_1, \ldots, o_t, a_1, \ldots, a_{t-1}) \\
&\propto P(o_t \mid o_1, \ldots, o_{t-1}, a_1, \ldots, a_{t-1}, s_t) P(s_t \mid o_1, \ldots, o_{t-1}, a_1, \ldots, a_{t-1}) \\
&= P(o_t \mid s_t) \sum_{s_{t-1}} P(s_t, s_{t-1} \mid o_1, \ldots, o_{t-1}, a_1, \ldots, a_{t-1}) \\
&= P(o_t \mid s_t) \sum_{s_{t-1}} P(s_t \mid s_{t-1}, a_{t-1}) P(s_{t-1} \mid o_1, \ldots, o_{t-1}, a_1, \ldots, a_{t-1}) \\
&= P(o_t \mid s_t) \sum_{s_{t-1}} P(s_t \mid s_{t-1}, a_{t-1}) b(s_{t-1})
\end{aligned}$$

13. **PCA on Circular Data (inspired by Barber, 15.2)**

Consider the following generative process for generating $2N$ data points along the circumference of a unit circle. (Recall that a unit circle satisfies $x^2 + y^2 = 1$.)

- Draw a point x_n along the x-axis $\sim \text{Unif}(-1, 1)$
- Given x_n, set $y_{n_1} = \sqrt{1 - x^2}$ and $y_{n_2} = -\sqrt{1 - x^2}$, and add both $\langle x_n, y_{n_1} \rangle$ and $\langle x_n, y_{n_2} \rangle$ to the dataset.

Suppose we generate N points x_n in this way, giving us a total of $2N$ datapoints.

(a) If N is very large, what do you expect $\bar{\boldsymbol{x}}$, the empirical mean of the data, to (approximately) be?

(b) Using the approximation of $\bar{\boldsymbol{x}}$ from part (a), write out the sample covariance matrix $\boldsymbol{S}$ of the data in terms of the x_n.

(c) What (numerical) values will the entries of $\boldsymbol{S}$ approach as N gets large? (Hint: You may want to use a rearrangement of the identity $Var[x] = E[x^2] - (E[x])^2$, and the fact that $x_n \sim \text{Unif}(-1, 1)$.)

(d) If we wanted to use PCA to reduce the datapoints to a single dimension, given your answer to part (c), what will the first principal component (i.e., eigenvector) and its corresponding eigenvalue approximately be?

(e) Is this surprising?

(a) Here, $\bar{\boldsymbol{x}} \approx \boldsymbol{0}$, since the x_n are uniformly distributed around 0 and since there is a negative y value for every positive y value.

(b) The sample covariance matrix is (occasionally) defined as $\frac{1}{N} \sum_{i=1}^{N} (\boldsymbol{x}_n - \bar{\boldsymbol{x}})(\boldsymbol{x}_n - \bar{\boldsymbol{x}})^\mathsf{T}$. Using part (a), we therefore have

$$\begin{aligned}\boldsymbol{S} &= \frac{1}{2N} \left[\sum_{n=1}^{N} \langle x_n, \sqrt{1 - x_n^2} \rangle \langle x_n, \sqrt{1 - x_n^2} \rangle^\mathsf{T} + \sum_{n=N+1}^{2N} \langle x_n, -\sqrt{1 - x_n^2} \rangle \langle x_n, -\sqrt{1 - x_n^2} \rangle^\mathsf{T} \right] \\ &= \frac{1}{2N} \begin{bmatrix} 2 \sum_n x_n^2 & 0 \\ 0 & 2 \sum_n (1 - x_n^2) \end{bmatrix},\end{aligned}$$

because the off-diagonal terms will cancel.

(c) We see that the top left diagonal entry will approach $E[x^2] = Var[x] + E[x]^2$. Using the fact that the x_n are distributed uniformly, the variance is $\frac{1}{12}(1 - -1)^2 = \frac{1}{3}$. Thus, $\boldsymbol{S}_{11} \approx \frac{1}{3}$ and $\boldsymbol{S}_{22} \approx \frac{2}{3}$.

(d) We can see from $\boldsymbol{S}$ that the principal eigenvector will approximately be $\langle 0, 1 \rangle$, with eigenvalue $\approx \frac{2}{3}$.

(e) If the data were generated uniformly on the circumference, it would not be the case that projecting onto the vertical axis would maximize variance more than, say, the horizontal axis. Our data generation process is responsible for this curiosity.

14. **High Dimensional Data (Bishop 12.1.4)**

Suppose we have a design matrix $\boldsymbol{X} \in \mathbb{R}^{N \times D}$ which has been centered, so the sample covariance matrix is $\boldsymbol{S} = \frac{1}{N}\boldsymbol{X}^\top \boldsymbol{X}$. Also, let $\boldsymbol{u}_d$, where $d = 1..D$, be the eigenvectors of $\boldsymbol{S}$.

(a) Show that the D vectors defined by $\boldsymbol{v}_d = \boldsymbol{X}\boldsymbol{u}_d$ are eigenvectors of $\frac{1}{N}\boldsymbol{X}\boldsymbol{X}^\top$, and that they have the same eigenvalues as their corresponding $\boldsymbol{u}_d$.

(b) Assuming we can recover the $\boldsymbol{u}_d$ from the $\boldsymbol{v}_d$ with reasonable time and memory, explain why calculating the $\boldsymbol{v}_d$ first might be useful if $N < D$.

(c) Show that the $\hat{\boldsymbol{u}}_d = \boldsymbol{X}^\top \boldsymbol{v}_d$ is, like $\boldsymbol{u}_d$, an eigenvector of $\boldsymbol{S}$.

(a) We have that $\frac{1}{N}\boldsymbol{X}^\top \boldsymbol{X}\boldsymbol{u}_d = \lambda_d \boldsymbol{u}_d$. Left-multiplying both sides by $\boldsymbol{X}$ gives the result.

(b) This way, we only need to explicitly represent and find the eigenvalues/vectors of an $N \times N$, rather than a $D \times D$ matrix.

(c) From part (a), we have that $\frac{1}{N}\boldsymbol{X}\boldsymbol{X}^\top \boldsymbol{v}_d = \lambda_d \boldsymbol{v}_d$. Left-multiplying both sides by $\boldsymbol{X}^\top$ gives the result. (As Bishop points out, $\hat{\boldsymbol{u}}_d$ is not necessarily normalized).

15. **Reconstruction Error in Matrix Form (inspired by Barber, 15.2.1)**

Suppose we have a set of centered data vectors $\boldsymbol{x}_n \in \mathbb{R}^D$, and we would like to project them onto a subspace of dimensionality K using a basis of K (not necessarily orthonormal) vectors $\boldsymbol{w}_k$. As in the notes, we define $\hat{\boldsymbol{x}}_n \equiv \sum_{k=1}^{K} \alpha_k^{(n)} \boldsymbol{w}_k$, and so the reconstruction error is

$$\begin{aligned} J(\{\boldsymbol{u}_k\}_{k=1}^{K}) &= \sum_{n=1}^{N} (\boldsymbol{x}_n - \hat{\boldsymbol{x}}_n)^\top (\boldsymbol{x}_n - \hat{\boldsymbol{x}}_n) \\ &= \sum_{n=1}^{N} (\boldsymbol{x}_n - \sum_{k=1}^{K} \alpha_k^{(n)} \boldsymbol{w}_k)^\top (\boldsymbol{x}_n - \sum_{k=1}^{K} \alpha_k^{(n)} \boldsymbol{w}_k). \end{aligned}$$

Now, let $\boldsymbol{X} \in \mathbb{R}^{N \times D}$ be the design matrix of data vectors, $\boldsymbol{W} \in \mathbb{R}^{D \times K}$ be a matrix of horizontally stacked $\boldsymbol{w}_k$ column vectors, and $\boldsymbol{Z} \in \mathbb{R}^{K \times N}$ be a matrix of horizontally stacked α column vectors. That is, $\boldsymbol{Z} = \begin{bmatrix} \alpha_1^{(1)} & \cdots & \alpha_1^{(N)} \\ \vdots & \ddots & \vdots \\ \alpha_K^{(1)} & \cdots & \alpha_K^{(N)} \end{bmatrix}$.

(a) Write the reconstruction error J only in terms of the matrices $\boldsymbol{X}$,$\boldsymbol{W}$, $\boldsymbol{Z}$, and matrix operations.

(b) Argue from the form of your answer to (a) that there are no unique $\boldsymbol{W}, \boldsymbol{Z}$ that minimize J.

(a) $J = \text{trace}((\boldsymbol{X}^\top - \boldsymbol{W}\boldsymbol{Z})^\top (\boldsymbol{X}^\top - \boldsymbol{W}\boldsymbol{Z}))$

(b) As Barber points out in section 15.2.1, J depends only on the product $\boldsymbol{W}\boldsymbol{Z}$, and there are clearly many matrices that can give this product.

16. **Q learning**

Suppose you are standing on a linear board: you can take action L or R (walk left or right), or sleep. If you walk, you have probability p_a that you actually walk to the next square, where $a \in L, R$. Otherwise, your cat distracted you and you are still on the same square. Staying a square gives you reward r_i. Your learning rate is $\alpha = .5$ and $\gamma = .5$.
You are on square 1, you choose $a = Left$ and receive $r = 4$. What is your updated value of $Q(1, L)$?

All Q values are initially 0. $Q(1, L) = 0 + .5(4 + .5 * 0 - 0) = 2$

17. **Q learning**

$\boxed{\text{In the next step from the previous problem, you are on square 0, you choose } a = R\text{, receive } r = 3 \text{ and end up in } s = 1\text{. What is your updated value of } Q(0, R)\text{?}}$

$Q(2, W) = 0 + .5(3 + .5 * 2 - 0) = 2$

18. **Model Based Reinforcement Learning**

In model-based RL, how might we compute the Maximum-Likelihood estimate of the expected reward from taking action a in state s?

We take the average of the rewards received from taking action a when we are in state s. We can keep track of $N(s,a)$, the number of times that we visit state s and perform a and $R^{\text{total}}(s,a)$, the sum of the rewards received from taking action a in state s. Then the ML estimate is given by

$$\frac{R^{\text{total}}(s,a)}{N(s,a)}$$

19. **ϵ-Greedy**

Why do we generally use an ϵ-Greedy algorithm when choosing the current action during the Q-Learning algorithm? Describe how you would change the value of ϵ over time to encourage early exploration/learning and good decision making after the state space was explored.

First we recall that in the ϵ-greedy algorithm we choose the best action, via

$$\max_{a' \in A} Q(s, a'),$$

with probability $1 - \epsilon$ and a random action otherwise. We use this because it encourages exploration of the state space by introducing randomness into the choice of action at each step, which we recall is necessary for RL to work with. In the beginning, we would choose a high value of ϵ, which would encourage a lot of random exploration of the state space, and as time went on, we could slowly decrease the value of ϵ to encourage taking the action that maximizes the Q value in the current state, which we expect to be a good action given that we've explored the state space sufficiently to closely approximate the true Q function.

20. **Convergence of Q-Learning**

Suppose we have a deterministic world where each state-action pair (s, a) is visited infinitely often. Consider an interval during which every such pair is visited. Suppose that the largest error in the approximation of $\hat{Q}_n$ after n iterations is e_n, i.e.

$$e_n = \max_{s,a} |\hat{Q}_n(s, a) - Q(s, a)|$$

Show that e_{n+1} is bounded above by γe_n, where γ is the usual discount factor.

$$\begin{aligned}
e_{n+1} &= \max_{s,a} |\hat{Q}_{n+1}(s, a) - Q(s, a)| \\
&= |(r + \gamma \max_{a'} \hat{Q}_n(s', a')) - (r + \gamma \max_{a'} Q(s', a'))| \\
&= \gamma | \max_{a'} \hat{Q}_n(s', a') - \gamma \max_{a'} Q(s', a')| \\
&\leq \gamma \max_{a'} |\hat{Q}_n(s', a') - Q(s', a')| \\
&\leq \gamma \max_{s'',a'} |\hat{Q}_n(s'', a) - Q(s'', a')| \\
&= \gamma e_n
\end{aligned}$$