Homework #5

ECE 461/661: Introduction to Machine Learning for Engineers
Prof. Carlee Joe-Wong and Prof. Gauri Joshi

Due: Monday, April 28, 2025
8:59 pm PT/11:59 pm ET/Tuesday, April 29, 2025 at 6:59 am CAT

Please remember to show your work for all problems and to write down the names of any students that you collaborate with. Although you are encouraged to work together, **you must write your own solutions, which should reflect your understanding of the problem.** When answering coding questions, you may use packages such as `numpy` for incidental operations like matrix multiplication. However, **you may not use built-in packages that directly implement the specified functions for you**, unless explicitly specified in the question. The full collaboration and grading policies are available on the course website: https://www.andrew.cmu.edu/course/18-661/. You are strongly encouraged (but not required) to use Latex to typeset your solutions.

Your solutions should be uploaded to Gradescope (https://www.gradescope.com/) in PDF format by the deadline. We will not accept hardcopies. **If you choose to hand-write your solutions, please make sure the uploaded copies are legible.** Gradescope will ask you to identify which page(s) contain your solutions to which problems, so make sure you leave enough time to finish this before the deadline. We will give you a 30-minute grace period to upload your solutions in case of technical problems.

1 Gaussian Mixture Models *[15 points]*

Consider an exponential mixture model for a 1-D dataset $\{x_n\}$ with the density function

$$p(x) = \sum_{k=1}^{K} \omega_k \text{Exp}(x|\mu_k),$$

where K is the number of mixture components, μ_k is the rate component and ω_k is the mixture weight corresponding to the k-th component. The exponential distribution is given by

$$\text{Exp}(x|\mu) = \mu \exp(-x\mu) \text{ for all } x \geq 0 \tag{1}$$

We would like to derive the model parameters (ω_k, μ_k) for all k using the EM algorithm. Consider the hidden labels $z_n \in \{1, \ldots, K\}$ and indicator variables r_{nk} that are 1 if $z_n = k$ and 0 otherwise. The complete log-likelihood (assuming base e for the log) is then written as

$$\sum_n \log p(\boldsymbol{x}_n, z_n) = \sum_n \sum_{z_n=k} \left[\log p(z_n = k) + \log p(x_n|z_n = k)\right].$$

(a) *[5 points]* Write down and simplify the expression for the complete log-likelihood for the exponential mixture model described above.

$$\sum_n \log p(x_n, z_n) = \sum_k \sum_n r_{nk} \left[\log \omega_k + \log \mu_k - x_n \mu_k\right]$$

(b) *[5 points]* Solve the M step of EM algorithm and find μ_k for $k = 1, \ldots, K$ that maximizes the complete log-likelihood.

(c) *[5 points]* Now perform the E step, and write the equation to update the soft labels $r_{nk} = P(z_n = k|x_n)$.

a. Plugging the definition of the exponential distribution here immediately gives

$$\begin{aligned}\sum_n \log p(x_n, z_n) &= \sum_k \sum_n r_{nk} \left[\log \omega_k + \log \text{Exp}(x_n|\mu_k)\right] \\ &= \sum_k \sum_n r_{nk} \left[\log \omega_k + \log \mu_k - x_n \mu_k\right]\end{aligned}$$

b. Taking the derivative of the log-likelihood with respect to μ_k and setting it to zero we have:

$$\frac{1}{\mu_k} \sum_n r_{nk} - \sum_n r_{nk} x_n = 0 \tag{2}$$

$$\mu_k = \frac{\sum_n r_{nk}}{\sum_n r_{nk} x_n} \tag{3}$$

c. Using the Bayes' rule:

$$\begin{aligned}r_{nk} &= \frac{P(x_n, z_n = k)}{P(x_n)} \\ &= \frac{\omega_k \mu_k \exp(-x_n \mu_k)}{\sum_k \omega_k \mu_k \exp(-x_n \mu_k)}.\end{aligned}$$

2 Eigenfaces *[35 points]*

Face recognition is an important task in computer vision and machine learning. In this question you will implement a classical approach called Eigenfaces. You will use face images from the `Yale Face Database B`, which contains face images from 10 people under 64 lighting conditions. **Please include your code in the final PDF you turn in for full credit.**

(a) **Dataset**. Download the data file `face_data.mat`. It contains three sets of variables:

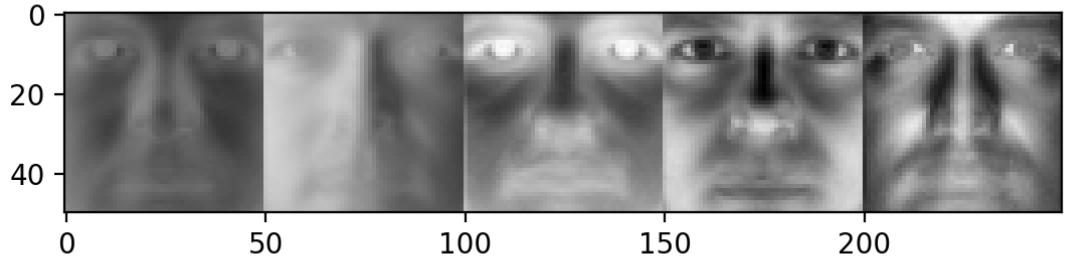
- `image`: each element is a face image (50×50 matrix). You can use `matplotlib.pyplot.imshow` to visualize the image. The data is stored in a cell array.
- `personID`: each element is the ID of the person, which takes values from 1 to 10.
- `subsetID`: each element is the ID of the subset which takes values from 1 to 5. Here the face images are divided into 5 subsets. Each subset contains face images from all people, but with different lighting conditions.

(b) *[10 points]* **Implement PCA**. Fill in the function `pca_fun` in the `pca.py` file. The function takes the data matrix (each row being a sample) and target dimensionality d (lower than or equal to the original dimensionality) as the input, and outputs the selected eigenvectors.

See the code.

(c) *[25 points]* **Compute Eigenfaces**. Take each 50×50 training image and vectorize it into a 2500-dimensional vector. Use your PCA implementation from part (b) to perform PCA on all vectorized face images, and retain the first $d = 200$ eigenvectors. These eigenvectors are called *eigenfaces* (when displayed as images). Please display the top 5 eigenfaces (use `imshow`) in your report.

Students should get eigenfaces like below (note: these are computed after centering the data. We will give full points for the correct results without centering, but the eigenfaces may look different.):



3 Thompson Sampling *[25 points]*

Consider the Thompson Sampling (TS) algorithm, a Bayesian approach to the multi-armed bandit problem. Consider a Bernoulli bandit with n arms, where each arm i at time-step $1 \leq t \leq T$ has Bernoulli i.i.d rewards $r_{i,t} \in \{0, 1\}$ with $\mathbb{E}[r_{i,t}] = \mu_i$. The TS algorithm starts with a prior distribution of μ_i for each arm i using the $P_{i,0} \sim Beta(1, 1)$ distribution and proceeds by selecting an arm based on the posterior distribution as follows. Note the prior distribution of μ_i at time t is denoted as $P_{i,t-1}$ and the posterior as $P_{i,t}$. Further, the posterior of the current time-step becomes the prior for the next time-step.

Algorithm 1 Thompson Sampling

for $t = 1, 2, \ldots, T$ **do**
  Sample $\hat{\mu}_{i,t} \sim P_{i,t-1}$ for each arm $i \in \{1, \ldots, n\}$
  Play arm $i_t = \arg\max_i \hat{\mu}_{i,t}$
  Observe reward $r_{i_t,t}$ and update posterior $P_{i,t}$
end for

Recall the probability density function of the Beta distribution, $Beta(\alpha, \beta)$, for any $x \in [0, 1]$ is

$$p(x) = \frac{(\alpha + \beta - 1)!}{(\alpha - 1)!(\beta - 1)!} x^{\alpha-1}(1 - x)^{\beta-1}.$$

We also know, for any $p, q \geq 1$,

$$\int_0^1 x^p (1 - x)^q dx = \frac{(p + q + 1)!}{(p - 1)!(q - 1)!}.$$

(a) *[15 points]* Until time-step t, suppose arm $i \in \{1, \ldots, n\}$ is pulled $N_{i,t}$ times and its total observed reward is $S_{i,t} := \sum_{u \leq t : i_u = i} r_{i,u}$ where i_t represents the arm chosen at time-step t. Find $P_{i,t}$ the posterior

distribution of μ_i, given the Beta prior as described above and observations on the rewards until time-step t. *(Hint: Compute the posterior for the first time-step. Use this recursively for the following time-steps.)*

Solution: Consider arm i is pulled at time-step t. We then compute the posterior distribution of $P_{i,t}$ as follows. Consider $\hat{\mu}_{i,t} \sim Beta(\alpha, \beta)$ and $r_{i,t} \sim Bern(\mu_i)$

$$\begin{aligned}
p(\hat{\mu}_{i,t+1} = \mu_i | r_{i,t}) &\propto p(r_{i,t} | \hat{\mu}_{i,t+1} = \mu_i) p(\hat{\mu}_{i,t+1} = \mu_i) \\
&= (\mu_i)^{r_{i,t}} (1-\mu_i)^{1-r_{i,t}} \frac{(\alpha+\beta-1)!}{(\alpha-1)!(\beta-1)!} (\mu_i)^{\alpha-1} (1-\mu_i)^{\beta-1} \\
&\propto (\mu_i)^{r_{i,t}+\alpha-1} (1-\mu_i)^{1-r_{i,t}+\beta-1} \\
p(\hat{\mu}_{i,t+1} = \mu_i | r_{i,t}) &= \frac{(\mu_i)^{r_{i,t}+\alpha-1} (1-\mu_i)^{1-r_{i,t}+\beta-1}}{\int_0^1 (\mu_i)^{r_{i,t}+\alpha-1} (1-\mu_i)^{1-r_{i,t}+\beta-1} d\mu_i} \\
&= \frac{(\alpha+\beta)!}{(r_{i,t}+\alpha-2)!(1-r_{i,t}+\beta-2)!} (\mu_i)^{r_{i,t}+\alpha-1} (1-\mu_i)^{1-r_{i,t}+\beta-1}
\end{aligned}$$

Hence recursively, the posterior $P_{i,t}$ is $\hat{\mu}_{i,t+1} \sim Beta(1 + S_{i,t}, 1 + (\sum_{u \le t} \mathbb{I}_{i_u=i}) - S_{i,t})$.

(b) *[5 points]* Compute the mean and variance of the posterior distribution of μ_i found in part (a).

Solution: Using mean and variance of Beta distribution,

$$\begin{aligned}
\mathbb{E}[\hat{\mu}_{i,t+1}] &= \frac{1 + S_{i,t}}{1 + S_{i,t} + 1 + (\sum_{u \le t} \mathbb{I}_{i_u=i}) - S_{i,t}} = \frac{1 + S_{i,t}}{2 + \sum_{u \le t} \mathbb{I}_{i_u=i}} \\
Var(\hat{\mu}_{i,t+1}) &= \frac{(1 + S_{i,t})(1 + \sum_{u \le t} \mathbb{I}_{i_u=i} - S_{i,t})}{(2 + \sum_{u \le t} \mathbb{I}_{i_u=i})^2 (3 + \sum_{u \le t} \mathbb{I}_{i_u=i})}
\end{aligned}$$

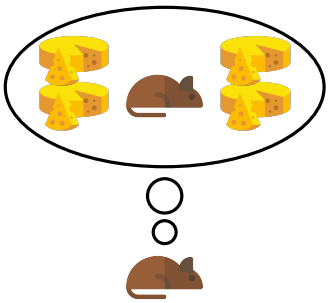
(c) *[5 points]* Using the computations in part (c), explain how TS balances exploration and exploitation.

Solution: In the initial rounds when the prior distribution is closer to $Beta(1,1)$, the probability mass is spread across the support range and has a higher variance. This allows samples $\hat{\mu}_{i,t}$ to be away from their true values. This randomizes their rank ordering and encourages exploration.

As the number of samples of reward observed from each arm increases, the posterior Beta distribution concentrates around the true mean of the Bernoulli reward arm, thus encouraging exploitation.

4 Gridworld *[25 points]*

Consider the following grid environment. Starting from any unshaded square, you can move up, down, left, or right. Actions are deterministic and always succeed (e.g. going left from state 16 goes to state 15) unless they will cause the agent to run into a wall. The thicker edges indicate walls, and attempting to move in the direction of a wall results in staying in the same square (e.g. going in any direction other than left from state 16 stays in 16). Taking any action from the target square with cheese (no. 11) earns a reward of r_g (so $r(11, a) = r_g \ \forall a$) and ends the episode. Taking any action from the square of cat (no. 6) earns a reward of r_r (so $r(6, a) = r_r \ \forall a$) and ends the episode. Otherwise, from every other square, taking any action is associated with a reward $r_s \in \{-1, 0, +1\}$ (even if the action results in the agent staying in the same square). Assume the discount factor $\gamma = 1$, $r_g = +10$, and $r_r = -1000$ unless otherwise specified.



1	2	3	4
5		7	8
9	10		12
13	14	15	16

(a) *[9 points]* Let $r_s = 0$. In the policy below, the arrow in each box denotes the (deterministic) action that should be taken in each state. Evaluate the following policy and show the corresponding value (i.e., cumulative reward) for every state (square).

→	→	↓	↓
→	→	↓	↓
↑	↑	↓	←
↑	↑	←	←

Solution:

10	10	10	10
-1000	-1000	10	10
-1000	-1000	10	10
-1000	-1000	-1000	-1000

(b) *[8 points]* Define the value of r_s that would cause the optimal policy to return the shortest path to the cheese square (no. 11); note that your value for r_s should lie in $\{-1, 0, +1\}$. Using this r_s, find the optimal value for each square and the optimal policy.

Solution: $r_s = -1$

6	7	8	7
5	-1000	9	8
4	3	10	9
3	2	1	0

→	→	↓	↓
↑	any	↓	↓
↑	←	any	←
↑	←/↑	←	←

(c) *[8 points]* Let's refer to the value function derived in (b) as $V_{old}^{\pi_g}$ and the policy as π_g. Suppose we are now in a new gridworld where all the rewards (r_s, r_g, and r_r) have +2 added to them. Consider still following the policy π_g, which is optimal for the original gridworld. What will the new values $V_{new}^{\pi_g}$ be in this second gridworld?

Solution:

16	15	14	15
17	-998	13	14
18	19	12	13
19	20	21	22

5 [Bonus] Markov Decision Process as Linear Program *[20 points]*

(This question is optional and counts towards extra credit.)

Consider a Markov Decision Process (MDP) $\mathcal{M} = \{\mathcal{S}, \mathcal{A}, P, r, \gamma\}$ where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, P is the transition kernel representing the probability of transition to state s' from state s when action a is taken as $P(s'|s,a)$, r is the reward function and γ is the discount factor. Consider the linear program

$$\min_{V \in \mathbb{R}^{|\mathcal{S}|}} \sum_{s \in \mathcal{S}} \rho(s) V(s)$$
$$\text{s.t.} \quad V(s) \geq r(s,a) + \gamma \sum_{s' \in \mathcal{S}} P(s'|s,a) V(s') \qquad \forall s \in \mathcal{S}, a \in \mathcal{A}$$

where $\rho(s) \in \mathbb{R}^+ \ \forall s \in \mathcal{S}$ and denote its solution as $V^\star \in \mathbb{R}^{|\mathcal{S}|}$.

(a) *[10 points]* Show that the dual formulation of the above linear program can be written as

$$\begin{aligned} &\max_{x \in \mathbb{R}^{|\mathcal{S}\times\mathcal{A}|}} \sum_{s\in\mathcal{S}, a\in\mathcal{A}} r(s,a)x(s,a) \\ \text{s.t.} \quad &\sum_{a\in\mathcal{A}} x(s,a) - \gamma \sum_{s'\in\mathcal{S}, a'\in\mathcal{A}} P(s|s',a')x(s',a') = \rho(s) \qquad \forall s \in \mathcal{S} \\ &x(s,a) \geq 0 \qquad \forall s \in \mathcal{S}, a \in \mathcal{A}. \end{aligned}$$

Solution: For an LP of the form

$$\min b^T y \quad \text{s.t.} \quad A^T y \geq c,$$

the dual is of the form

$$\max c^T x \quad \text{s.t.} \quad Ax = b, x \geq 0.$$

Observe that the primal LP has $|\mathcal{S} \times \mathcal{A}|$ constraints.

(b) *[10 points]* Denote the optimal solution to the dual problem as $x^\star \in \mathbb{R}^{|\mathcal{S}\times\mathcal{A}|}$. Due to strong duality and complementary slackness, we have

$$x^\star(s,a)\left(V^\star(s) - \gamma \sum_{s'\in\mathcal{S}} P(s'|s,a)V^\star(s') - r(s,a)\right) = 0 \qquad \forall s \in \mathcal{S}, a \in \mathcal{A}.$$

Now, show that the optimal policy $\pi^\star(\cdot|s)$, $\forall s \in \mathcal{S}$, can be derived as

$$\pi^\star(a|s) = \begin{cases} 1 & \text{if } a = \arg\max_{a\in\mathcal{A}} x^\star(s,a) \\ 0 & \text{else.} \end{cases}$$

Solution: To ensure the properties of complementary slackness are met, in every state, for the optimal action $x^\star(s,a^\star) > 0$ and $V^\star(s) - \gamma \sum_{s'\in\mathcal{S}} P(s'|s,a)V^\star(s') - r(s,a) = 0$ and for all other actions $x^\star(s, a \neq a^\star) = 0$. Hence the policy is constructed as $\pi^\star(s) = \arg\max_a x^\star(s,a)$ choosing the action(s) that corresponds to the optimal tight primal constraint and resulting $x^\star(s,a^\star) > 0$.