Homework 5

Name: Edward Ajayi

Andrew ID: eaajayi

Program: MS EAI

Institution: Carnegie Mellon University Africa

Semester: Spring 2025

Unit: 18-661 - Introduction to Machine Learning for Engineers

Faculty: Prof. Carlee Joe-Wong and Prof. Gauri Joshi

Date: April 29, 2025

Question 1: Gaussian Mixture Models

We are given a 1-D mixture model where each component is an exponential distribution:

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

where:

- ω_k: mixture weight for component k, with $\omega_k \geq 0$ and $\sum_{k=1}^{K} \omega_k = 1$
- μ_k: rate of the exponential distribution for component k
- $\text{Exp}(x \mid \mu) = \mu e^{-x\mu}$ for $x \geq 0$

Let $\{x_n\}_{n=1}^{N}$ be the observed data, and $z_n \in \{1, \ldots, K\}$ be the latent indicator of the component generating x_n. Let $r_{nk} = \mathbb{I}[z_n = k]$ be the binary indicator variable for $z_n = k$.

(a) Complete Log-Likelihood

We want to write down the complete data log-likelihood:

$$\log p(\{x_n, z_n\}_{n=1}^{N}) = \sum_{n=1}^{N} \log p(x_n, z_n)$$

Assuming $p(z_n = k) = \omega_k$ and $p(x_n \mid z_n = k) = \mu_k e^{-x_n \mu_k}$, we get:

$$p(x_n, z_n = k) = \omega_k \mu_k e^{-x_n \mu_k}$$

Therefore, the complete log-likelihood becomes:

$$\log p(\{x_n, z_n\}) = \sum_{n=1}^{N} \sum_{k=1}^{K} r_{nk} \log(\omega_k \mu_k e^{-x_n \mu_k}) = \sum_{n=1}^{N} \sum_{k=1}^{K} r_{nk} \left(\log \omega_k + \log \mu_k - x_n \mu_k\right)$$

$$\boxed{\log p(\{x_n, z_n\}) = \sum_{n=1}^{N} \sum_{k=1}^{K} r_{nk} \left(\log \omega_k + \log \mu_k - x_n \mu_k\right)}$$

(b) M-Step: Maximizing with respect to μ_k

We isolate the relevant terms of the log-likelihood for component k:

$$L(\mu_k) = \sum_{n=1}^{N} r_{nk} \left(\log \mu_k - x_n \mu_k \right)$$

Differentiate with respect to μ_k:

$$\frac{dL}{d\mu_k} = \sum_{n=1}^{N} r_{nk} \left(\frac{1}{\mu_k} - x_n \right)$$

Set the derivative to zero:

$$\sum_{n=1}^{N} r_{nk} \left(\frac{1}{\mu_k} - x_n \right) = 0 \Rightarrow \frac{1}{\mu_k} \sum_{n=1}^{N} r_{nk} = \sum_{n=1}^{N} r_{nk} x_n \Rightarrow \mu_k = \frac{\sum_{n=1}^{N} r_{nk}}{\sum_{n=1}^{N} r_{nk} x_n}$$

$$\boxed{\mu_k = \frac{\sum_{n=1}^{N} r_{nk}}{\sum_{n=1}^{N} r_{nk} x_n}}$$

(c) E-Step: Computing Soft Labels r_{nk}

We want to compute the posterior:

$$r_{nk} = P(z_n = k \mid x_n) = \frac{P(x_n \mid z_n = k) P(z_n = k)}{\sum_{j=1}^{K} P(x_n \mid z_n = j) P(z_n = j)}$$

Substitute in the exponential pdf and priors:

$$r_{nk} = \frac{\omega_k \mu_k e^{-x_n \mu_k}}{\sum_{j=1}^{K} \omega_j \mu_j e^{-x_n \mu_j}}$$

$$\boxed{r_{nk} = \frac{\omega_k \mu_k e^{-x_n \mu_k}}{\sum_{j=1}^{K} \omega_j \mu_j e^{-x_n \mu_j}}}$$

Question 2 - Eigenfaces

```
from scipy.io import loadmat
from matplotlib import pyplot as plt
import numpy as np

def pca_fun(input_data, target_d):

    # Center the data
    mean = np.mean(input_data, axis=0)
    centered_data = input_data - mean

    # Covariance matrix
    cov_matrix = np.cov(centered_data, rowvar=False)

    # Eigen decomposition
    eigvals, eigvecs = np.linalg.eigh(cov_matrix)

    # Sort eigenvectors by eigenvalues in descending order
    sorted_indices = np.argsort(eigvals)[::-1]
    top_eigvecs = eigvecs[:, sorted_indices[:target_d]]

    return top_eigvecs


### Data loading and plotting the image ###
import os
print(os.path.exists('face_data.mat'))
data = loadmat('face_data.mat')
images = data['image'][0]
person_id = data['personID'][0]
```

```
True
```

Computing Eigenfaces

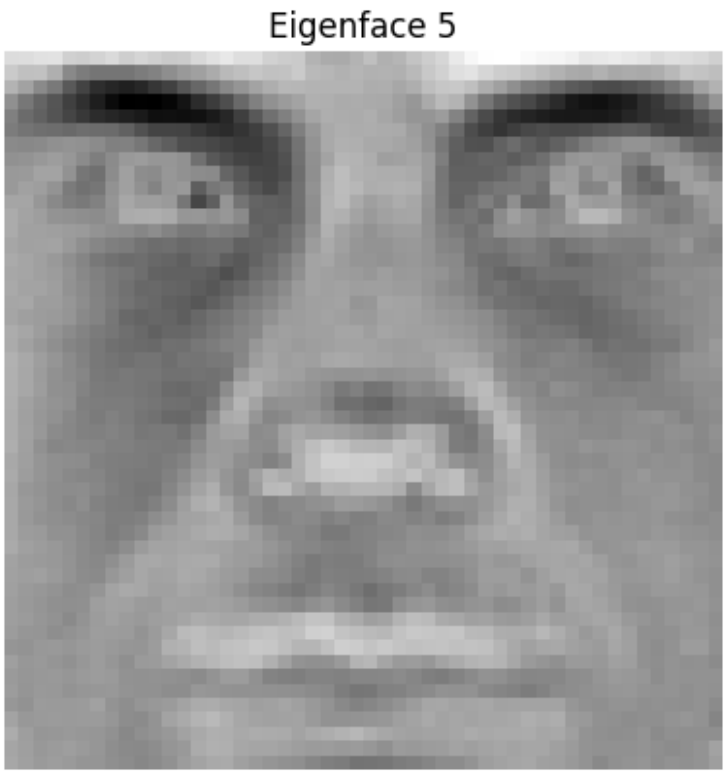
```
image_vecs = np.array([img.flatten() for img in images])

top_eigvecs = pca_fun(image_vecs, target_d=200)  # Shape: (2500, 200)

# Step 3: Display the top 5 eigenfaces
for i in range(5):
    eigenface = top_eigvecs[:, i].reshape(50, 50)
    plt.imshow(eigenface, cmap='gray')
    plt.title(f"Eigenface {i+1}")
    plt.axis('off')
    plt.show()
```

Eigenface 1

Eigenface 2

Eigenface 3

Eigenface 4




```
top_eigvecs.shape
```

```
(2500, 200)
```

Question 3: Thompson Sampling

We consider a Bernoulli bandit with n arms, where each arm i yields i.i.d. rewards $r_{i,t} \in \{0, 1\}$ with expectation μ_i. We begin with a prior distribution $\mu_i \sim \text{Beta}(1, 1)$ and apply Thompson Sampling.

Q. 3a

Let $N_{i,t}$ denote the number of times arm i has been pulled by time t, and $S_{i,t} := \sum_{u \leq t : i_u = i} r_{i,u}$ be the number of successes.

Given that the Beta distribution is conjugate prior to the Bernoulli likelihood, we update the posterior as follows:

$$\text{If} \quad \mu_i \sim \text{Beta}(\alpha_0, \beta_0)$$

and we observe s successes and f failures,

$$\text{then posterior:} \quad \mu_i \sim \text{Beta}(\alpha_0 + s, \beta_0 + f)$$

Starting with a prior $\text{Beta}(1, 1)$, and observing $S_{i,t}$ successes in $N_{i,t}$ trials:

$$\boxed{P_{i,t} = \text{Beta}(1 + S_{i,t}, 1 + N_{i,t} - S_{i,t})}$$

Q. 3b

For a Beta distribution $\text{Beta}(\alpha, \beta)$, the mean and variance are:

$$\mathbb{E}[\mu_i] = \frac{\alpha}{\alpha + \beta}$$
$$\text{Var}[\mu_i] = \frac{\alpha\beta}{(\alpha + \beta)^2(\alpha + \beta + 1)}$$

Using $\alpha = 1 + S_{i,t}$ and $\beta = 1 + N_{i,t} - S_{i,t}$:

$$\boxed{\mathbb{E}[\mu_i] = \frac{1 + S_{i,t}}{2 + N_{i,t}}, \quad \text{Var}[\mu_i] = \frac{(1 + S_{i,t})(1 + N_{i,t} - S_{i,t})}{(2 + N_{i,t})^2(3 + N_{i,t})}}$$

Q. 3c

Thompson Sampling naturally balances exploration and exploitation by sampling from the posterior distribution for each arm.

If an arm has a high observed average reward, its Beta posterior mean will be higher, so the sampled $\hat{\mu}_{i,t}$ will likely be high — leading to **exploitation** (playing the best-known arm).

On the other hand, if an arm has been pulled only a few times, its posterior distribution remains wide (high variance), meaning that there is still a significant probability of sampling a high $\hat{\mu}_{i,t}$, encouraging **exploration** (giving uncertain arms a chance).

Thus, Thompson Sampling automatically explores uncertain arms and exploits well-performing arms by using random samples from their posteriors.

Question 4

Q. 4a

To evaluate the given deterministic policy under the setting $r_s = 0$, we trace the path followed from each state to its terminal outcome (either the cheese at state 11 or the cat at state 6), count the number of steps taken, and compute the resulting value (cumulative reward). The results are summarized in Table 1.

Table 1: Policy Evaluation with $r_s = 0$

State	**Path (Following Given Policy)**	**Steps**	**Value**
1	$\rightarrow 2 \rightarrow 3 \downarrow 7 \downarrow 11$	4	10
2	$\rightarrow 3 \downarrow 7 \downarrow 11$	3	10
3	$\downarrow 7 \downarrow 11$	2	10
4	$\downarrow 8 \downarrow 12$ (stuck)	–	0
5	$\rightarrow 6$ (cat)	1	-1000
6	(Terminal: Cat)	0	-1000
7	$\downarrow 11$	1	10
8	$\downarrow 12$ (stuck)	–	0
9	$\uparrow 5 \rightarrow 6$ (cat)	2	-1000
10	$\uparrow 7 \downarrow 11$	2	10
11	(Terminal: Cheese)	0	10
12	(blocked, stuck)	–	0
13	$\uparrow 9 \uparrow 5 \rightarrow 6$ (cat)	3	-1000
14	$\uparrow 10 \uparrow 7 \downarrow 11$	3	10
15	$\leftarrow 14 \uparrow 10 \uparrow 7 \downarrow 11$	4	10
16	$\leftarrow 15 \leftarrow 14 \uparrow 10 \uparrow 7 \downarrow 11$	5	10

Q. 4b

To encourage the agent to reach the cheese as quickly as possible, we must select a step reward r_s that penalizes unnecessary movement.

- If we set $r_s = +1$, taking more steps would increase the cumulative reward, which would incentivize the agent to wander indefinitely rather than reaching the goal.
- If we set $r_s = 0$, the agent would have no incentive to minimize steps, and may follow arbitrary paths as long as it eventually reaches the cheese.

- However, setting $r_s = -1$ penalizes each additional step by decreasing the total reward by one unit per step. This motivates the agent to minimize the number of steps and thus encourages it to follow the shortest path to the cheese.

Therefore, to ensure that the optimal policy returns the shortest path to the cheese, we set: $r_s = -1$

Table 2: Optimal Policy Evaluation with $r_s = -1$

State	**Path (Following Given Policy)**	**Steps**	**Value**
1	$\rightarrow 2 \rightarrow 3 \downarrow 7 \downarrow 11$	4	6
2	$\rightarrow 3 \downarrow 7 \downarrow 11$	3	7
3	$\downarrow 7 \downarrow 11$	2	8
4	$\downarrow 8 \downarrow 12 \leftarrow 11$	3	7
5	$\rightarrow 6$ (then any action)	1	-1001
6	(Terminal: Cat)	0	-1000
7	$\downarrow 11$	1	9
8	$\downarrow 12 \leftarrow 11$	2	8
9	$\uparrow 5 \rightarrow 6$ (then any action)	2	-1002
10	$\uparrow 6$	1	-1001
11	(Terminal: Cheese)	0	10
12	$\leftarrow 11$	1	9
13	$\uparrow 9 \uparrow 5 \rightarrow 6$ (then any action)	3	-1003
14	$\uparrow 10 \uparrow 6$	2	-1002
15	$\leftarrow 14 \uparrow 10 \uparrow 6$	3	-1003
16	$\leftarrow 15 \leftarrow 14 \uparrow 10 \uparrow 6$	4	-1004

Q. 4c

Now, suppose that all rewards (r_s, r_g, and r_r) are increased by $+2$ units:

$$r_s = -1 + 2 = 1, \quad r_g = 10 + 2 = 12, \quad r_r = -1000 + 2 = -998$$

Since the optimal policy remains the same as in part (b), and the number of steps from each state to the terminal states does not change, the updated value function $V_{\text{new}}(s)$ can be computed by adjusting the old values $V_{\text{old}}(s)$ as follows:

$$V_{\text{new}}(s) = V_{\text{old}}(s) + 2 \times (\text{number of steps})$$

The final updated values for each state are summarized below.

Table 3: Updated Value Function after Adding +2 to All Rewards (Including Paths)

State	**Optimal Path**	**Steps to Terminal**	$V^{\pi_g}_{\textbf{old}}$	$V^{\pi_g}_{\textbf{new}}$
1	$\rightarrow 2 \rightarrow 3 \downarrow 7 \downarrow 11$	4	6	14
2	$\rightarrow 3 \downarrow 7 \downarrow 11$	3	7	13
3	$\downarrow 7 \downarrow 11$	2	8	12
4	$\downarrow 8 \rightarrow 12 \leftarrow 11$	3	7	13
5	$\rightarrow 6$ (then any action)	1	-1001	-999
6	(Terminal: Cat)	0	-1000	-1000
7	$\downarrow 11$	1	9	11
8	$\rightarrow 12 \leftarrow 11$	2	8	12
9	$\rightarrow 10 \uparrow 6$	2	-1002	-998
10	$\uparrow 6$	1	-1001	-999
11	(Terminal: Cheese)	0	10	10
12	$\leftarrow 11$	1	9	11
13	$\rightarrow 14 \uparrow 10 \uparrow 6$	3	-1003	-997
14	$\uparrow 10 \uparrow 6$	2	-1002	-998
15	$\leftarrow 14 \uparrow 10 \uparrow 6$	3	-1003	-997
16	$\leftarrow 15 \leftarrow 14 \uparrow 10 \uparrow 6$	4	-1004	-996

Question 5

Q. 5a

The primal linear program (LP) is:

$$\min_{V \in \mathbb{R}^{|S|}} \sum_{s \in S} \rho(s) V(s)$$

subject to

$$V(s) \geq r(s,a) + \gamma \sum_{s' \in S} P(s'|s,a) V(s') \quad \forall s \in S,\, a \in A$$

We form the Lagrangian by introducing dual variables $x(s,a) \geq 0$ for each constraint:

$$L(V,x) = \sum_{s \in S} \rho(s) V(s) + \sum_{s \in S, a \in A} x(s,a) \left(r(s,a) + \gamma \sum_{s' \in S} P(s'|s,a) V(s') - V(s) \right)$$

Expanding the Lagrangian:

$$L(V,x) = \sum_{s \in S} \rho(s) V(s) + \sum_{s \in S, a \in A} x(s,a) r(s,a) + \gamma \sum_{s,a,s'} x(s,a) P(s'|s,a) V(s') - \sum_{s,a} x(s,a) V(s)$$

Grouping the terms involving V:

$$L(V,x) = \sum_{s \in S, a \in A} x(s,a) r(s,a) + \sum_{s \in S} V(s) \left(\rho(s) + \gamma \sum_{s',a'} x(s',a') P(s|s',a') - \sum_{a \in A} x(s,a) \right)$$

To obtain the dual function $g(x) = \inf_V L(V,x)$, the coefficient of each $V(s)$ must be zero for the infimum to be finite. Thus, for each $s \in S$:

$$\sum_{a \in A} x(s,a) - \gamma \sum_{s' \in S, a' \in A} P(s|s',a') x(s',a') = \rho(s)$$

Therefore, the dual LP is:

$$\max_{x \geq 0} \sum_{s \in S, a \in A} r(s,a) x(s,a)$$

subject to

$$\sum_{a \in A} x(s,a) - \gamma \sum_{s' \in S, a' \in A} P(s|s',a') x(s',a') = \rho(s) \quad \forall s \in S$$

and

$$x(s,a) \geq 0 \quad \forall s \in S,\, a \in A$$

Q. 5b

From complementary slackness, the optimal solutions V^* and x^* satisfy:

$$x^*(s,a)\left(V^*(s) - \left(r(s,a) + \gamma \sum_{s' \in S} P(s'|s,a)V^*(s')\right)\right) = 0 \quad \forall s \in S, a \in A$$

Thus:

- If $x^*(s,a) > 0$, then

$$V^*(s) = r(s,a) + \gamma \sum_{s' \in S} P(s'|s,a)V^*(s')$$

- If $x^*(s,a) = 0$, then

$$V^*(s) \geq r(s,a) + \gamma \sum_{s' \in S} P(s'|s,a)V^*(s')$$

Therefore, the optimal action at state s corresponds to the action a that maximizes $x^*(s,a)$. The optimal policy is given by:

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