Homework 5 - Introduction to Machine Learning For Engineers

kipngeno koech - bkoech

April 23, 2025

1 Gaussian Mixture Models

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

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

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

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

We would like to derive the model parameters (ω_k, μ_k) for all k using the EM algorithm. Consider the hidden labels $z_n \in \{1, \dots, K\}$ 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

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

1. Write down and simplify the expression for the complete log-likelihood for the exponential mixture model described above. Plugging the definition of the exponential distribution here immediately gives

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

2. Solve the M step of the EM algorithm and find μ_k for $k = 1, \dots, K$ that maximizes the complete log-likelihood. Taking the derivative of the log-likelihood with respect to μ_k and setting it to zero, we have:

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

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

3. Perform the E step of the EM algorithm and write the equation to update the soft labels $r_{nk} = P(z_n = k | x_n)$. Using Bayes' rule, we have:

$$r_{nk} = \frac{P(x_n, z_n = k)}{P(x_n)} = \frac{\omega_k \mu_k \exp(-x_n \mu_k)}{\sum_{k'} \omega_{k'} \mu_{k'} \exp(-x_n \mu_{k'})}.$$

2 Eigen Faces

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.

1. **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.
2. [**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.
3. [**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.

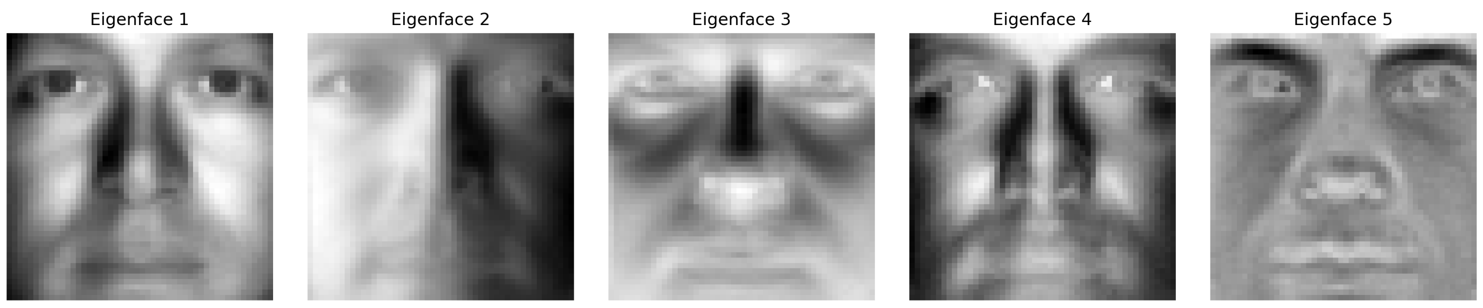

Figure 1: Top 5 Eigenfaces

3 Thompson Sampling

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 $P_{i,0} \sim \text{Beta}(1,1)$ 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

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

Recall the probability density function of the Beta distribution, $\text{Beta}(\alpha, \beta)$, 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 $p, q \geq 1$,

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

1. **[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_u represents the arm chosen at time-step u. 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.)

To find the posterior distribution $P_{i,t}$ of the unknown mean reward μ_i of arm i at time-step t, we begin with the prior and apply Bayesian updates.

Initially, the prior distribution of μ_i is:

$$P_{i,0} = \text{Beta}(1,1)$$

Suppose arm i has been pulled $N_{i,t}$ times by time t, and the total observed reward is:

$$S_{i,t} := \sum_{u \leq t:\ i_u = i} r_{i,u}$$

Each reward $r_{i,u} \sim \text{Bernoulli}(\mu_i)$, and since the Beta distribution is a conjugate prior for the Bernoulli likelihood, the posterior distribution after observing $S_{i,t}$ successes and $N_{i,t} - S_{i,t}$ failures becomes:

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

This result can be obtained recursively. After each new observation of reward $r_{i,t}$ at time-step t, the posterior is updated as:

$$P_{i,t} = \text{Beta}(\alpha_{i,t},\ \beta_{i,t})$$

where

$$\alpha_{i,t} = 1 + \sum_{u \leq t:\ i_u = i} r_{i,u} = 1 + S_{i,t}, \quad \beta_{i,t} = 1 + \sum_{u \leq t:\ i_u = i} (1 - r_{i,u}) = 1 + N_{i,t} - S_{i,t}$$

Hence, the posterior distribution $P_{i,t}$ at time-step t is:

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

2. **[5 points]** Compute the mean and variance of the posterior distribution of μ_i found in part (a).

Let $P_{i,t} = \text{Beta}(\alpha, \beta)$ where

$$\alpha = 1 + S_{i,t}, \quad \beta = 1 + N_{i,t} - S_{i,t}$$

Then the **mean** of μ_i is:

$$\mathbb{E}[\mu_i] = \frac{1 + S_{i,t}}{2 + N_{i,t}}$$

And the **variance** of μ_i is:

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

3. **[5 points]** Using the computations in part (b), explain how TS balances exploration and exploitation.

   Thompson Sampling (TS) naturally balances exploration and exploitation by using random samples drawn from the posterior distribution of each arm's expected reward.

   From part (b), we know that the posterior distribution of the mean reward μ_i for arm i at time t is:

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

   with mean:

$$\mathbb{E}[\mu_i] = \frac{1 + S_{i,t}}{2 + N_{i,t}}$$

   and variance:

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

   Exploitation: When an arm is pulled frequently (i.e., $N_{i,t}$ is large), the variance of its posterior distribution becomes small. As a result, the samples $\hat{\mu}_{i,t}$ drawn from this posterior concentrate near the true mean μ_i, leading TS to consistently select arms with higher estimated reward.

   Exploration: For arms that have been pulled fewer times (i.e., small $N_{i,t}$), the variance of their posterior remains large. This increases the probability of drawing a high sample $\hat{\mu}_{i,t}$ by chance, which may temporarily make that arm appear better than others. This leads TS to occasionally explore uncertain arms, even if their empirical average is lower.

4 Gridworld

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 the 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.

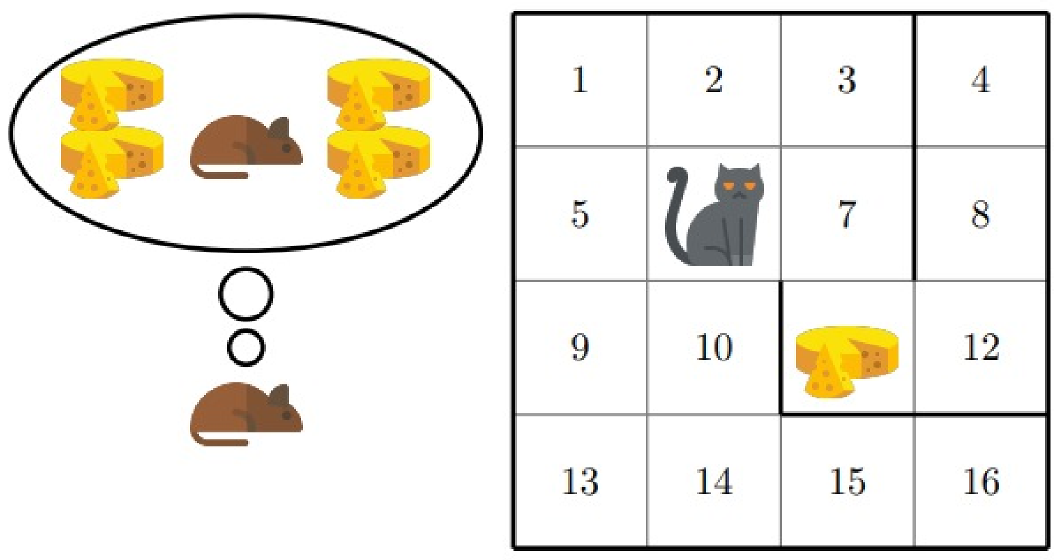

Figure 2: Gridworld Environment

1. 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).

$\rightarrow$	$\rightarrow$	$\downarrow$	$\downarrow$
$\rightarrow$	$\rightarrow$	$\downarrow$	$\downarrow$
$\uparrow$	$\uparrow$	$\downarrow$	$\leftarrow$
$\uparrow$	$\uparrow$	$\leftarrow$	$\leftarrow$

2. 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.

3. Let's refer to the value function derived in (b) as $V_{\text{old}}^{\pi_g}$ 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 $V_{\text{new}}^{\pi_g}$ be in this second gridworld?

5 Markov Decesion Process as Linear Program

Consider a Markov Decision Process (MDP) $M = \{S, A, P, r, \gamma\}$ where S is the state space, 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:

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

subject to

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

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

1. Show that the dual formulation of the above linear program can be written as

$$\max_{x\in\mathbb{R}^{|S|\times|A|}} \sum_{s\in S}\sum_{a\in A} r(s,a)x(s,a)$$

subject to

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

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

To derive the dual, rewrite the constraint in standard form:

$$V(s) - \gamma \sum_{s'\in\mathcal{S}} P(s' \mid s,a)V(s') \geq r(s,a).$$

Associate a dual variable $x(s,a) \geq 0$ with each constraint. The Lagrangian is:

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

Collect terms involving $V(s)$:

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

For the dual, set the coefficient of $V(s)$ to zero:

$$\rho(s) - \sum_{a\in\mathcal{A}} x(s,a) + \gamma\sum_{s'\in\mathcal{S}}\sum_{a'\in\mathcal{A}} P(s \mid s',a')x(s',a') = 0 \quad \forall s\in\mathcal{S}.$$

Thus, the dual constraint is:

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

The dual objective is to maximize the constant term:

$$\max \sum_{s\in\mathcal{S}}\sum_{a\in\mathcal{A}} x(s,a)r(s,a).$$

Therefore, the dual linear program is:

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

2. Denote the optimal solution to the dual problem as $x^\star \in \mathbb{R}^{|S\times A|}$. Due to strong duality and complementary slackness, we have

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

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

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

Given the optimal dual solution $x^*(s,a)$ and primal solution $V^*(s)$, complementary slackness gives:

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

If $x^*(s,a) > 0$, the constraint is tight:

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

Since $V^*(s)$ is the optimal value function, it satisfies:

$$V^*(s) = \max_{a' \in \mathcal{A}} \left\{ r(s,a') + \gamma \sum_{s' \in \mathcal{S}} P(s' \mid s,a') V^*(s') \right\}.$$

Thus, $x^*(s,a) > 0$ implies action a achieves the maximum, so it is optimal. Define the policy:

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

If $a^* = \arg\max_{a'} x^*(s,a')$ and $x^*(s,a^*) > 0$, then a^* is optimal, satisfying the Bellman equation. If $x^*(s,a) = 0$, action a is suboptimal or not selected. Since an optimal policy can be deterministic, choosing $a^* = \arg\max x^*(s,a)$ ensures optimality.

Thus, the optimal policy is:

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