Introduction to of Machine Learning for Engineers

Homework 5

Marion Kipsang

mkipsang@andrew.cmu.edu

April 27, 2025

1. Gaussian Mixture Models

a In expectation maximization, the complete data includes both the observed data x_n and hidden latent variables z_n. We define the indicator variables:

$$r_{nk} = \begin{cases} 1 & \textit{if } z_n = k \\ 0 & \textit{otherwise} \end{cases}$$

Thus, the complete log-likelihood is:

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

We recall the exponential density as:

$$Exp(x|\mu) = \mu e^{-x\mu} \implies \log Exp(x|\mu) = \log \mu - x\mu$$

Plugging this into the complete log-likelihood, we have:

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

b For the M-step, we maximize the complete log-likelihood with respect to μ_k, holding the soft assignments r_{nk} fixed.

For fixed k, the relevant components of the equation are:

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

Thus, for each component k, the objective to be maximized is:

$$\mathcal{Q}(\mu_k) = (\sum_{n=1}^{N} r_{nk}) \log \mu_k - (\sum_{n=1}^{N} r_{nk} x_n) \mu_k$$

Differentiating $\mathcal{Q}(\mu_k)$ with respect to μ_k:

- First term:
$$\frac{d}{d\mu_k} (\sum_{n=1}^{N} r_{nk}) \log \mu_k = \frac{\sum_{n=1}^{N} r_{nk}}{\mu_k}$$
- Second term:
$$\frac{d}{d\mu_k} (\sum_{n=1}^{N} r_{nk} x_n) \mu_k = \sum_{n=1}^{N} r_{nk} x_n$$

Thus, setting the derivative to zero:

$$\frac{d\mathcal{Q}}{d\mu_k} = \frac{\sum_{n=1}^{N} r_{nk}}{\mu_k} - \sum_{n=1}^{N} r_{nk} x_n = 0$$

Rearranging yields:

$$\frac{\sum_{n=1}^{N} r_{nk}}{\mu_k} = \sum_{n=1}^{N} r_{nk} x_n \implies \sum_{n=1}^{N} r_{nk} = \mu_k (\sum_{n=1}^{N} r_{nk} x_n)$$

Solving for μ_k:

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

This is the M-step update for μ_k.

c In the E-step of expectation maximization, given the current estimates of the parameters μ_k and ω_k, we compute the soft labels: $r_{nk} = P(z_n = k|x_n)$, the probability that a data point x_n belongs to component k.

We apply Bayes' rule:

$$r_{nk} = \frac{P(z_n = k)P(x_n|z_n = k)}{P(x_n)}$$

where:

- $P(z_n = k) = \omega_k$ is the mixture weight
- $P(x_n|z_n = k) = Exp(x_n|\mu_k) = \mu_k e^{-x_n\mu_k}$
- $P(x_n) = \sum_{j=1}^{K} \omega_j \mu_j e^{-x_n\mu_j}$ (total marginal density)

Thus we get:

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

This is the E-step update for r_{nk}.

3. Thompson Sampling

a Since we are dealing with a Bernoulli bandit setup, let's define:

- $\mu_i \in [0, 1]$ is the (unknown) probability of success, that is, the mean reward for arm i.
- At each time step $t = 1, 2, \ldots, T$, an arm i_t is chosen and its reward $r_{i,t} \in \{0, 1\}$ is observed.
- For each arm i, we initialize a prior:

$$\mu_i \sim Beta(\alpha = 1, \beta = 1)$$

  which is equivalent to a uniform distribution over $[0, 1]$

We let:

- $N_{i,t}$ be the total number of times arm i has been selected by time t
- $S_{i,t}$ be the total number of successes observed from arm i until time t:

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

To derive the posterior distribution, we first find the likelihood. Since rewards are i.i.d and Bernoulli:

- Each reward from arm i: $r \sim Bernoulli(\mu_i)$
- So the probability of $S_{i,t}$ successes and $N_{i,t} - S_{i,t}$ failures is:

$$p(D_t, \mu_i) = \mu_i^{S_{i,t}} (1 - \mu_i)^{N_{i,t} - S_{i,t}}$$

Therefore, the unnormalized posterior distribution is given by:

$$p(\mu_i | D_t) \propto \mu_i^{S_{i,t}} (1 - \mu_i)^{N_{i,t} - S_{i,t}} \cdot 1$$

This is exactly the kernel of a Beta distribution with parameters:

- $\alpha' = 1 + S_{i,t}$
- $\beta' = 1 + N_{i,t} - S_{i,t}$

Therefore, the final posterior distribution is given by:

$$P_{i,t} = \mu_i | D_t \sim Beta(1 + S_{i,t}, 1 + N_{i,t} - S_{i,t})$$

b For a random variable $X \sim Beta(\alpha, \beta)$:

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

Substituting for $\mu_i | D_t \sim Beta(1 + S_{i,t}, 1 + N_{i,t} - S_{i,t})$:

- The mean of the posterior is given by:

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

  This is the smoothed empirical average of the observed rewards for arm i.
- The variance of the posterior is given by:

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

c Instead of always choosing the arm with the highest mean (greedy), Thompson Sampling uses a random sample from the full posterior, allowing arms with high uncertainty to sometimes be selected.

Arms with higher uncertainty (higher posterior variance) have a higher probability of sampling high estimated values, promoting exploration. Arms with higher estimated means are more likely to be sampled with high values, promoting exploitation.

Thus, Thompson Sampling naturally trades off between exploring uncertain arms and exploiting arms with known good rewards, without requiring any manually tuned exploration bonuses.

4. Gridworld

a Give the deterministic policy shown by the arrows, we have:

- $r_s = 0$ (no penalty for movement)
- $r_g = +10$ (cheese at square 11)
- $r_r = -1000$ (cat at square 6)
- Discount factor $\gamma = 1$
- Episode ends upon reaching either square 6 or 11.

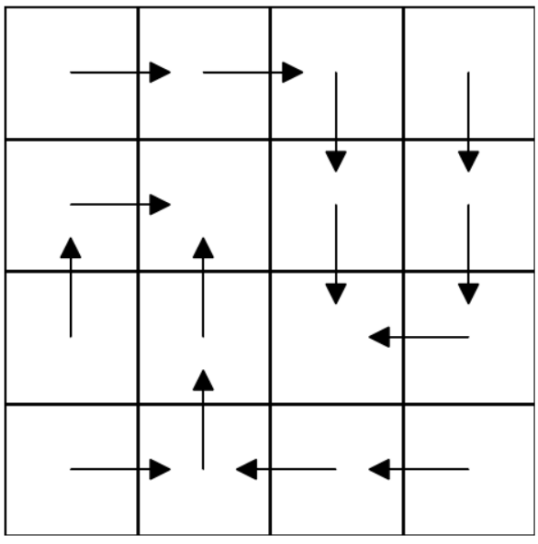
We compute the value of each state under the given policy. We use the following approach:

- Since $r_s = 0$, moving between squares does not affect the value.
- The only rewards occur upon reaching a terminal state (6 or 11).

State(square)	Path	Value
1	$1 \to 2 \to 3 \to 7 \to 11$	+10
2	$2 \to 3 \to 7 \to 11$	+10
3	$3 \to 7 \to 11$	+10
4	$4 \to 8 \to 12 \to 11$	+10
5	$5 \to 6$	-1000
6	terminal at 6	-1000
7	$7 \to 11$	+10
8	$8 \to 12 \to 11$	+10
9	$9 \to 5 \to 6$	-1000
10	$10 \to 6$	-1000
11	terminal at 11	+10
12	$12 \to 11$	+10
13	$13 \to 9 \to 5 \to 6$	-1000
14	$14 \to 10 \to 6$	-1000
15	$15 \to 14 \to 10 \to 6$	-1000
16	$16 \to 15 \to 14 \to 10 \to 6$	-1000

b When $r_s = -1$, every move is penalized encouraging short paths, when $r_s = 0$, there is no penalty for moving around, and when $r_s = +1$, every move is rewarded thus encouraging wandering. Therefore, to encourage shortest paths, we must penalize movement, thus set $r_s = -1$

With $r_s = -1$, we design optimal moves for each square as follows:



The optimal values are as follows:

State(square)	Path	Value
1	$1 \to 2 \to 3 \to 7 \to 11$	+6
2	$2 \to 3 \to 7 \to 11$	+7
3	$3 \to 7 \to 11$	+8
4	$4 \to 8 \to 12 \to 11$	+7
5	$5 \to 6$	-1001
6	terminal at 6	-1000
7	$7 \to 11$	+9
8	$8 \to 12 \to 11$	+8
9	$9 \to 5 \to 6$	-1002
10	$10 \to 6$	-1001
11	terminal at 11	+10
12	$12 \to 11$	+9
13	$13 \to 14 \to 10 \to 6$	-1003
14	$14 \to 10 \to 6$	-1002
15	$15 \to 14 \to 10 \to 6$	-1003
16	$16 \to 15 \to 14 \to 10 \to 6$	-1004

Therefore:

$$V(state) = sum\ of\ rewards\ along\ shortest\ path\ to\ cheese\ or\ cat$$

c The new rewards are as follows:

- $r_s = +1$
- $r_g = 12$
- $r_r = -998$

To compute new values:

- Count how many k steps to reach terminal steps.
- Compute:
$$V = (new\ step\ reward) \times k + (new\ terminal\ reward)$$

The new values are as follows:

State(square)	Path	Value
1	$1 \to 2 \to 3 \to 7 \to 11$	+16
2	$2 \to 3 \to 7 \to 11$	+15
3	$3 \to 7 \to 11$	+14
4	$4 \to 8 \to 12 \to 11$	+15
5	$5 \to 6$	-997
6	terminal at 6	-1000
7	$7 \to 11$	+13
8	$8 \to 12 \to 11$	+14
9	$9 \to 5 \to 6$	-996
10	$10 \to 6$	-997
11	terminal at 11	+10
12	$12 \to 11$	+13
13	$13 \to 14 \to 10 \to 6$	-995
14	$14 \to 10 \to 6$	-996
15	$15 \to 14 \to 10 \to 6$	-995
16	$16 \to 15 \to 14 \to 10 \to 6$	-994

5. [Bonus] Markov Decision Process as Linear Program

a Given the primal linear program:

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

subject to

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

As per linear programming duality basics, for a primal minimization linear program $\min_V c^T V$ subject to $AV \geq b$, its dual is given by $\max_x b^T x$ subject to $A^T x = c, x \geq 0$.

We then match the Markov decision process primal to the linear program form as follows:

- Given the objective $\min \sum_s \rho(s) V(s)$ and the constraints for each (s,a) as

$$V(s) - \gamma \sum_{s'} P(s'|s,a) V(s') \geq r(s,a)$$

- We notice that each constraint corresponds to a (state, action) pair. Thus, in matrix form:

$$AV \geq r$$

  where:
  - $V \in \mathbb{R}^{|S|}$
  - $r \in \mathbb{R}^{|S \times A|}$
  - $A \in \mathbb{R}^{|S \times A| \times |S|}$

  For constraint (s,a), the row is:
  - 1 for $V(s)$
  - $-\gamma P(s'|s,a)$ for $V(s')$

We now write the dual. According to LP duality, the dual variables $x(s,a)$ are associated with the primal constraints (one for each (s,a), therefore:

- The dual objective is:

$$\max \sum_{s,a} r(s,a) x(s,a)$$

- The dual constraints are:

$$A^T x = \rho$$

  plus non-negativity:

$$x(s,a) \geq 0$$

To expand $A^T x = \rho$, we recall that:

- Each row of A corresponds to (s,a)
- The corresponding entry for $V(s)$ is $+1$
- The entries for $V(s')$ are $-\gamma P(s'|s,a)$

Therefore, for each state s, the dual constraint is:

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

Therefore, the dual formulation is:

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

subject to:

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

$$x(s,a) \geq 0$$

b Given that:

- V^* is the optimal value function from the primal linear program.
- x^* is the optimal solution to the dual linear program.
- Strong duality and complementary slackness hold.

From linear programming duality theory, complementary slackness says that at optimality, for each constraint in the primal linear program:

$$(\textit{constraint}) \times (\textit{corresponding dual variable}) = 0$$

meaning that, for every state s and action a:

$$x^*(s,a)(V^*(s) - (r(s,a) + \gamma \sum_{s'} P(s'|s,a)V^*(s'))) = 0$$

This means that:

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

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

  that is, the action a must be optimal at state s
- if $x^*(s,a) = 0$, then:

$$\textit{No condition: } V^*(s) \geq r(s,a) + \gamma \sum_{s'} P(s'|s,a)V^*(s')$$

- Therefore:
  - Only the actions with positive $x(s,a)$ are optimal actions for state s.
  - If multiple $x(s,a) > 0$, any of them could be used.

Since we are to select the action with the highest $x(s,a) > 0$, we define the optimal policy as follows, for each state s:

- The optimal action is:

$$a^*(s) = \arg\max_{a' \in A} x^*(s,a')$$

- The optimal policy is:

$$\pi^*(a|s) = \begin{cases} 1 & \textit{if } a = a^*(s) \\ 0 & \textit{otherwise} \end{cases}$$