UNIVERSITY OF PENNSYLVANIA

ESE 650: LEARNING IN ROBOTICS

SPRING 2022

MIDTERM EXAM

03/15 TUE 9AM ET - 03/16 WED 8.59AM ET

DURATION: 180 MINUTES

MAXIMUM POINTS: 100

Read the following instructions carefully before you begin.

- You are allowed to use the class notes and any of the designated textbooks from the class syllabus. You are **not** allowed to use the internet or discuss with your peers.
- You will have 180 minutes from the time you start the exam to upload your solutions. Late submissions will only receive 50% of the credit. Make sure you keep enough buffer to upload and annotate solutions into Gradescope.
- You should use a pen and paper (do not use a pencil because it might not be readable when scanned) to write your solutions and click pictures and upload your solutions as a PDF (Dropbox app on your phone is great for scanning). Please make sure to write in legible handwriting. You can also use a tablet to write solutions.
- Begin each problem on a fresh page. Solutions that are not correctly annotated on Gradescope outline will not receive credit.
- Each question mentions how short/long we would like your responses to be. DO NOT write excessively verbose answers.
- None of the questions require long derivations. If you find yourself going through lots of equations reconsider your approach or consider moving on to the next question.
- If you are stuck, explain your answers and what you are trying to do clearly. We will give partial credit if you are on the right track.
- **This exam has 4 problems.** The questions are NOT arranged in order of difficulty. Try to attempt every question. You do not need to write code for any of the questions.

Problem 1 (30 points). Short answer questions.

(1) **(3 points)** Answer with a 2-3 sentence explanation. The Baum-Welch algorithm is used to learn a Hidden Markov Model (HMM) from an observation sequence $Y_1, ..., Y_t$. Given an HMM $\lambda = (\pi, T, M)$, it solves a maximum-likelihood estimation problem iteratively to find a new HMM $\lambda' = (\pi', T', M')$ such that $P(Y_{1:t}; \lambda') > P(Y_{1:t}; \lambda)$. When do we stop in our iterated application of Baum-Welch algorithm? Are we always guaranteed to find the same HMM λ' irrespective of our initial HMM λ?

Answer: We can decide to stop the iterative process when the updates to λ' are within a small margin. This can also be seen when $P(Y_{1:t}; \lambda') \approx P(Y_{1:t}; \lambda)$ which signifies that the model has converged. We are not always guaranteed to find the same λ' because different initializations of λ lead to different solutions.

(2) **(3 points)** Answer with a 3-4 sentence explanation. A robot tasked with estimating the 2D occupancy grid of an indoor environment is equipped with a LiDAR sensor (which is accurate and has a very focused field of view). We can update the state of each cell by estimating the log-odds ratio using the equation:

$$l(m_i \mid y_{1:k}, x_{1:k}) = l(m_i \mid y_k, x_k) + l(m_i \mid y_{1:k-1}, x_{1:k-1}) - l(m_i).$$

How does a LiDAR measurement affect the likelihood term in the equation? Assume that you can define the prior odds term $l(m_i)$ before you turn on the robot. What value (high/low) would you choose for the prior odds term in the following cases: a) the environment is dynamic and will have many people constantly moving around the robot, b) the indoor environment is expected to be static.

Answer: The LiDAR sensor is accurate so if a LiDAR detects an obstacle it is very likely that there is really an occupied cell at that location. We can therefore use a high value of the LiDAR likelihood $l(m_i \mid y_k, x_k)$ to quickly call a cell occupied. In the dynamic case we need a high value for the log odds prior, i.e., the prior probability of a cell being occupied should be large. This will diminish the log-odds of all cells which are not reported as obstacles by the latest/last-few LiDAR scan. In the static case we can use a low value or 0 for the log odds prior because if we detect a wall we want that to remain constant.

(3) **(3 points)** Answer with a 1-2 sentence explanation. Consider the following 2D transformation matrix:

$$T = \begin{bmatrix} \sin & \cos \\ \cos & -\sin \end{bmatrix}$$

Is T a rigid-body transformation? What is the determinant of T?
(Remember: $cos^2\theta + \sin^2\theta = 1$)
(**Bonus 1 point:** what type of transformation is T?)

Answer: T is a reflection around the horizontal axis and does not represent a rotation since its determinant is -1:

$$det(T) = -(cos^2\theta + \sin^2\theta) = -1$$

Rotation matrices belong to the special orthogonal group where the determinant of each element is $+1$. Following the general definition of rigid-body transformations that says they preserve the Euclidean distance between every pair of points, T is a rigid-body transformation. However, in robotics, rigid-body transformations typically refer to proper rigid transformations that include rotation and translation as they represent robot motion. In this stricter definition, T is not a rigid-body transformation.

(4) **(3 points)** Answer with a 1-2 sentence explanation. You are asked to design a Kalman Filter for a car with linear dynamics moving in a linear environment. The position of the car at time t is given by x_t. Its velocity is $\dot{x}$ and its acceleration is $\ddot{x}$. What is the minimal state vector for the Kalman filter such that the resulting system is Markovian? Explain when a system is Markovian (based on the definition of its state).

Answer: The state should include both position and velocity

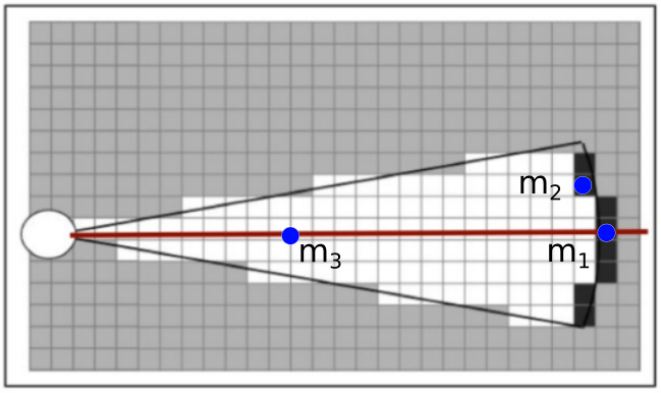
$$Y_t = \begin{bmatrix} x_t \\ \dot{x}_t \end{bmatrix}$$

The Markov property says that the next state is independent of past states given the current state. Therefore, the state of the system should contain all information necessary such that we can obtain its value at the following timestep.

(5) **(3 points)** Answer with a 2-3 sentence explanation. Suppose you are given a system with linear dynamics and linear observations but the noise in both the dynamics and observations is zero-mean but not Gaussian. Out of the four filtering algorithms that we know, namely the Kalman Filter, the Extended Kalman Filter, the Unscented Kalman Filter and the Particle Filter, which filtering algorithm will you pick?

Answer: We know that both the KF and EKF can only handle Gaussian noise. The UKF also handles only Gaussian noise because its update equations are based on the same equations as those of the Kalman filter (in particular, the expression of the Kalman gain in 3.35 is optimal only if the matrix noise in observations is Gaussian). The PF on the other hand can handle any kind of noise, we can simply add the non-Gaussian dynamics noise when we propagate the particles forward in the dynamics step and use the appropriate non-Gaussian likelihood to compute P$(y_{k+1} \mid x^{(i)}_{k+1|k})$.

(6) **(3 points)** Answer with a 2-3 sentence explanation. Consider a sonar sensor that gives a one real-valued reading corresponding to the distance measured along the optical (red) axis. Assume that the measurement was incorporated using a recursive Bayes filter to compute the occupancy probabilities of the map. Which of the cells m_1, m_2, and m_3 in the following figure is more likely to have the highest occupancy probability $P(m_1)$, $P(m_2)$, $P(m_3)$? Can you give approximate values to these probabilities? How would those values change if instead of a sonar we used a LiDAR sensor?



Answer: Cell m_1 is expected to have the highest occupancy probability. Possible occupancy probability values are $P(m_1) > 0.5$, $0.5 > P(m_2) > 0$,

and $P(m_3) \approx 0$. LiDAR has smaller FOV than sonar, hence it is more accurate, so that would increase $P(m_1)$, decrease $P(m_2)$, and decrease $P(m_3)$ even closer to 0.

(7) **(3 points)** Answer True or False with a 1-2 sentence explanation. The mean squared error of the Kalman filter can be improved if given access to future observations, i.e.,

$$\mathrm{E}\left[(\mu_k - x_k)^2 \mid y_0, \ldots, y_k\right] \geq \mathrm{E}\left[(\mu_k - x_k)^2 \mid y_0, \ldots, y_{k+1}\right];$$

here x_k is the true state of a linear dynamical system at time k, y_k is an observation linear in the state and μ_k is the mean of the Kalman filter's estimate of the state.

Answer: True. The Kalman filter estimate is the best estimate (smallest mean squared error estimate) given *all* past observations. This estimate can be improved if we know what occurred in the future, e.g., in the case of smoothing.

(8) **(3 points)** Answer with a 1-4 sentence explanation. According to the definition of the finite-horizon optimal control problem, is it possible to have a cost function that is independent of the control, i.e. a state-only dependent cost function?

Answer: Yes. The cost function can have a trivial dependency on the action, i.e. be the same for all actions. This is a special case of the state-action dependent cost function.

(9) **(3 points)** Answer with a 1-4 sentence explanation. You want your robot to hold a can of coke in the air for some time, such that the user can grab it whenever they want. You decide to use finite-horizon control with dynamic programming. You set the terminal cost to the distance between the coke position and the target; and runtime cost to actuator force. Is this control algorithm and this cost function a good choice for your problem?

Answer: No. This formulation will only ensure that the robot holds the coke in the air for one brief moment, not for a long period of time. The behavior the

robot will learn is to throw the coke into the direction of the target since that is easier to do than holding it there.

(10) **(3 points)** Answer with a 1-4 sentence explanation.

```
| 2 | 1 | 0 |   x - obstacle
| 3 | x | x |   3 - cost
| 4 | 3 | 1 |
```

You are given a gridworld with an obstacle and 7 states as above and you want to solve the discounted infinite-horizon control problem. The robot can move left, right, up, down, or stay. The cost is the same for every control and approximates the Euclidean distance to the goal; it is depicted in the figure above. Pick a discount factor $0 \leq \gamma < 1$ that you think is appropriate for this problem.

Answer: γ above 0.92 is correct. γ below 0.91 is incorrect.

Problem 2 (30 points). Hidden Markov Models.

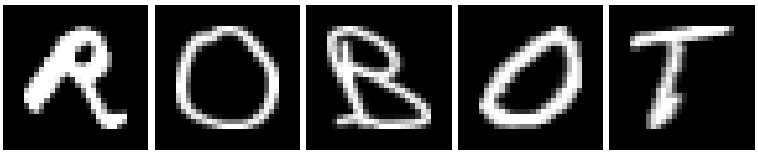
(1) **(10 points)** Consider an HMM with a finite number of states and a finite number of observations. The transition and observation matrices T, M are known, along with the initial distribution of the states π. We are given a sequence of observations $Y_1, ..., Y_k$ from this system.

(a) **(5 points)** How can we naively compute the likelihood $P(Y_1, ..., Y_k)$ of this sequence of observations? What is the computational complexity of this computation? Explain your answer.

(b) **(5 points)** Explain how we can simplify this computation. What is the resulting computational complexity of the new solution?

Answer: See Section 2.4.1 in the class notes. The computational complexity of the naive approach is N^K where N is the number of states and K is the number of timesteps. This is because we have to consider all possible state trajectories the HMM could have taken. Using the Forward algorithm we can reduce the computational complexity to N^2K.

(2) **(8 points)** You are asked to develop a handwriting recognition system using an HMM. Suppose you are given the following handwritten sequence of images:



The objective is to determine the letter represented in each image of the sequence. Assume that you have an imperfect character recognition model (e.g., a neural network) that classifies each image to one of the letters in the alphabet, where the alphabet has size $N = 26$ (assume only uppercase letters).

(a) **(6 points)** Draw the HMM diagram, clearly defining the state and observations. Describe in detail how you would define and populate the transition and observation matrices, and the initial state distribution.

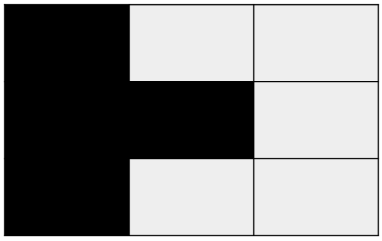
(b) **(2 points)** Out of the three algorithms we know for solving problems with HMMs, namely the Bayes Filtering, the Forward-Backward, and the Viterbi, which algorithms can you use and is there a particular one that you would expect to give the best result for this problem?

Answer:

(a) The state is a random variable $X \in \mathbb{R}^N$ defined over every letter in the alphabet. The observation z is the output of the character recognition model that provides a possibly wrong estimate of the letter represented in the image. The transition matrix is $T \in \mathbb{R}^{N \times N}$ that holds the probability of transitioning from each letter to every other letter between time-steps. These probabilities can be calculated from an existing corpus by computing the frequencies of these transitions. The observation matrix $M \in \mathbb{R}^{N \times N}$ holds the probabilities of observing a certain letter given any state. This matrix can be populated by measuring the efficacy of our character recognition model. We can simply count the outputs of the model given different images, i.e., how many times it predicted the letter a given images of a, the letter b given images of a etc. Initially the state can be uniformly distributed, i.e., we assume equal prior probability for every letter in the alphabet, or we can also compute these priors from an existing corpus.

(b) Technically we can use any of these algorithms to solve the problem. However, since we have access to the entire sequence of observations from the beginning, then we should expect the best result using the Viterbi algorithm since it estimates the most likely state trajectory.

(3) **(12 points)** Bayes filtering for localization. Consider a robot operating on a 2D grid world of 3 x 3 cells $c(i, j)$ where the cell with index $i = 1$ and $j = 1$ is in the upper left corner of the grid and i increases in the horizontal direction. The grid has obstacles in positions $(1, 1)$, $(1, 2)$, $(2, 2)$ and $(1, 3)$ shown in black color in the figure below:



The transition model for action *Move* says that the robot can equally likely end up at any neighboring square:

$$P(X_{t+1} = c_2 | X_t = c_1, U_t = Move) = \frac{1}{N(c_1)}$$

where $N(c_1)$ is the number of neighbors for state c_1. Initially the robot starts with uniform distribution $\frac{1}{5}$ over all free cells. The sensor value Z_t is a 4 bit sequence giving the presence or absence of an obstacle in that particular compass direction (NSEW for North, South, East, West). The boundaries of the grid are not considered obstacles. Suppose that the sensor has an error rate ϵ and that error occurs independently for the 4 sensor directions. So the probability of 4 measurements being right is $(1-\epsilon)^4$, and all of them wrong is ϵ^4. For example, the probability that a cell with obstacles North and South would produce reading NSE is $(1-\epsilon)^3\epsilon^1$. Show the distribution of $P(X_1|Z_1 = NSW)$ and $P(X_2|Z_1 = NSW, Z_2 = NS)$ assuming that $\epsilon = 0.2$.

Answer: We can compute the probability that the robot is located at each cell given measurement $Z_1 = NSW$ following the equation:

$$\begin{aligned} P(X_1|Z_1 = NSW) &= \eta P(Z_1 = NSW|X_1)P(X_1) \\ &= \eta P(Z_1 = NSW|X_1)\sum_{X_0} P(X_1|X_0)P(X_0) \end{aligned}$$

where η is the normalization factor. We know that the initial distribution is uniform and the robot is equally likely to transition to any neighbor, so the value of the summation is $\frac{1}{5}$. To estimate the distribution of $P(X_1|Z_1 = NSW)$:

$$P(X_1 = (2,1)|Z_1 = NSW) = \eta * (1-\epsilon)^3 * \epsilon^1 * \frac{1}{5} = \eta * 0.02048$$

$$P(X_1 = (3,1)|Z_1 = NSW) = \eta * (1-\epsilon)^1 * \epsilon^3 * \frac{1}{5} = \eta * 0.00128$$

$$P(X_1 = (3,2)|Z_1 = NSW) = \eta * (1-\epsilon)^2 * \epsilon^2 * \frac{1}{5} = \eta * 0.00512$$

$P(X_1 = (3,3)|Z_1 = NSW) =$ Same as (3,1)

$P(X_1 = (2,3)|Z_1 = NSW) =$ Same as (2,1)

The normalization factor can be estimated:

$$\eta = \frac{1}{\sum_{X_1} P(Z_1 = NSW|X_1)P(X_1)} = \frac{1}{0.04864}$$

So the actual probabilities are:

$$P(X_1 = (2,1)|Z_1 = NSW) = 0.421$$
$$P(X_1 = (3,1)|Z_1 = NSW) = 0.0263$$
$$P(X_1 = (3,2)|Z_1 = NSW) = 0.1052$$
$$P(X_1 = (3,3)|Z_1 = NSW) = 0.0263$$
$$P(X_1 = (2,3)|Z_1 = NSW) = 0.421$$

We can compute the distribution at the next time-step following the equation:

$$\begin{aligned} P(X_2|Z_1 = NSW, Z_2 = NS) &= \eta P(Z_2 = NS|X_2) P(X_2|Z_1 = NSW) \\ &= \eta P(Z_2 = NS|X_2) \sum_{X_1} P(X_2|X_1) P(X_1|Z_1 = NSW) \end{aligned}$$

For each cell:

$$P(X_2 = (2,1)|Z_1 = NSW, Z_2 = NS) = \eta * (1-\epsilon)^2 * \epsilon^2 * 0.0263$$
$$P(X_2 = (3,1)|Z_1 = NSW, Z_2 = NS) = \eta * (1-\epsilon)^2 * \epsilon^2 * (\frac{1}{2} * 0.421 + \frac{1}{2} * 0.1052)$$
$$P(X_2 = (3,2)|Z_1 = NSW, Z_2 = NS) = \eta * (1-\epsilon)^1 * \epsilon^3 * 0.0263$$
$$P(X_2 = (3,3)|Z_1 = NSW, Z_2 = NS) = \text{Same as (3,1)}$$
$$P(X_2 = (2,3)|Z_1 = NSW, Z_2 = NS) = \text{Same as (2,1)}$$

Estimate the normalization factor to get the final probabilities similarly to the first step.

Problem 3 (24 points). Kalman Filters.

(1) **(12 points)** Consider a linear system where the dynamics and observations of the states are given by

$$x_{k+1} = Ax_k + Bu_k$$
$$y_k = Cx_k + \nu_k$$

where the noise ν_k is correlated in time with correlations modeled as

$$\nu_k = D\nu_{k-1} + \rho_{k-1}$$

where $\rho_k \sim N(0, W)$ is zero-mean Gaussian noise with covariance W for all k. We would like to compute the estimate $\hat{x}_k$ of the state at time k such that this estimate is unbiased, i.e., $E[\hat{x}_k] = x_k$ and it minimizes the variance

$$\mathop{\mathrm{E}}_{\nu_0,\ldots,\nu_k}[(x_k - \hat{x}_k)^2]$$

Given that the noise ν_k is not white, how should we change the system such that we can use the Kalman Filter? Show the update equations of the Kalman Filter for this new system. Give a sketch of the solution, you do not need to derive the answer exactly. You might find the following typical update equations for the Kalman Filter useful.

$$\mu_{k+1|k} = A\mu_{k|k} + Bu_k$$
$$\Sigma_{k+1|k} = A\Sigma_{k|k}A^\top + R$$
$$K_{k+1} = \Sigma_{k+1|k}C^\top(C\Sigma_{k+1|k}C^\top + Q)^{-1}$$
$$\mu_{k+1|k+1} = \mu_{k+1|k} + K_{k+1}(y_{k+1} - C\mu_{k+1|k})$$
$$\Sigma_{k+1|k+1} = (I - K_{k+1}C)\Sigma_{k+1|k}$$

Answer: The observation noise ν_k is not white so we cannot use the Kalman filtering equations directly. However, we know how this noise changes in time so we can incorporate ν_k as a state of the system and define a new state

$$z_k = [x_k, \nu_k].$$

The new dynamics for z_k is given by

$$z_{k+1} = \underbrace{\begin{bmatrix} A & 0 \\ 0 & D \end{bmatrix}}_{A'} z_k + \underbrace{\begin{bmatrix} B \\ 0 \end{bmatrix}}_{B'} u_k + \begin{bmatrix} 0 \\ I \end{bmatrix} \rho_k.$$

The observations for this system are given by

$$y_k = \underbrace{\begin{bmatrix} C & I \end{bmatrix}}_{C'} z_k;$$

there is no noise in the observations, there is only noise in the dynamics. This is a linear dynamical system with additive Gaussian noise; the question asks for an unbiased estimator of the state x_k given past observations with the least variance which we know to be the result of the Kalman filter. We can now plug in the update equations of the Kalman filter in this setup.

$$\Sigma_{k+1|k} = \left(A' \Sigma_{k|k} \, A'^{\top} + \begin{bmatrix} 0 & 0 \\ 0 & W \end{bmatrix} \right) \quad \text{(note the covariance of dynamics noise)}$$

$$K_{k+1} = \Sigma_{k+1|k} C'^{\top} \left(C' \Sigma_{k+1|k} C'^{\top} + 0 \right)^{-1} \quad (Q \text{ is zero})$$

$$\hat{z}_{k+1|k+1} = (A' \hat{z}_{k|k} + B' u_k) + K_{k+1}(y_{k+1} - C'(A' \hat{z}_{k|k} + B' u_k))$$

$$\Sigma_{k+1|k+1} = \left(I - K_{k+1} C' \right) \, \Sigma_{k+1|k}$$

The estimate of $\hat{x}_k$ is finally given by

$$\hat{x}_k = \begin{bmatrix} I & 0 \end{bmatrix} \hat{z}_k.$$

(2) **(12 points)** We know that Kalman Filters (KF) are optimal for linear systems with Gaussian noise, which is not true for non-linear systems. The Extended Kalman Filter (EKF) is a modification of the KF to handle such situations.

(a) **(6 points)** Describe the basic idea and steps of the Extended Kalman Filter. How do the update equations differ from KF?

(b) **(4 points)** Explain in what situations EKF state estimates can be inaccurate. Optionally, you can answer this with an example.

(c) **(2 points)** After applying a non-linear transform $y = f(x)$ the resulting distribution y might not be Gaussian (even if x is Gaussian). In which case would the distribution y be Gaussian?

Answer:

(a) See Section 3.6.2 in the class notes. The main difference from the update equations of KF is that the matrices A and C are not deterministic anymore

(they represent Jacobians) and have to be estimated at every iteration because they depend on current estimates of the state.

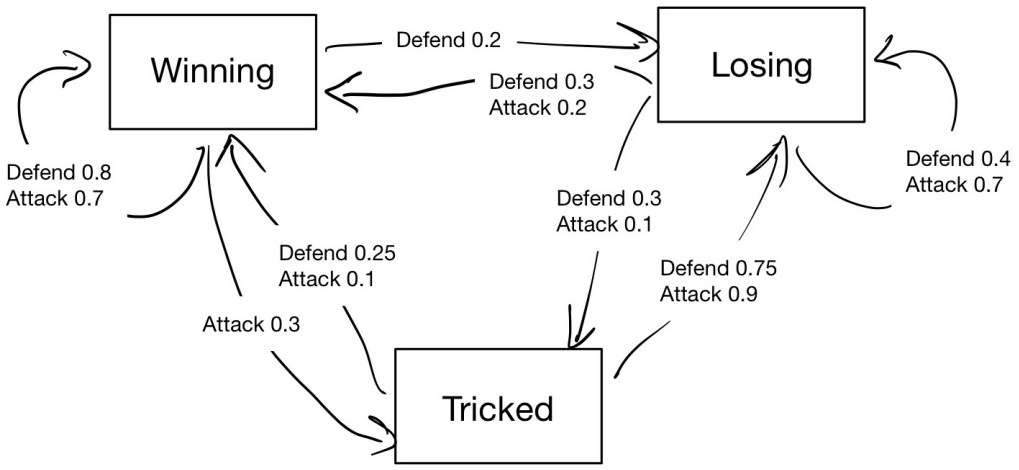
(b) EKF can produce bad estimates when the system goes into states where the linearized matrix A and the nonlinear dynamics $f(x_k, u_k)$ differ significantly. One example is the radar sensor with a non-linear observation model that we saw in class, which was accurate only in specific states when $x \approx 0$ or $h \approx 0$. Another example is that of the race car which can have abrupt changes in velocity.

(c) When $f(x)$ is linear so that we know the transformation matrix A.

Problem 4 (16 points). Consider the following Markov Decision Process. Garry Kasparov is playing chess with Deep Blue and must choose whether to defend or attack. There are three states:

(a) Winning (W) where Garry incurs a cost of 0,
(b) Losing (L) where Garry incurs a cost of 1, and
(c) Tricked (T) where Garry incurs a large cost of 5.

Garry begins in state W 40% of the time and in state L 60% of the time. The MDP including the transition probabilities for each action (Defend or Attack) is as follows. Note: the cost of taking an action u from state x is the cost of state x as denoted in the graph.



1. **(10 points)** Calculate the results of initializing the Q function as in the algorithm from the lectures, and one iteration of the Q Iteration algorithm for an infinite-horizon dynamic programming problem with a discount factor of $\gamma = 0.8$.
2. **(3 points)** Given Q^2, which policy would you select? Is your computation so far sufficient to prove this policy is optimal?
3. **(3 points)** The bellman equation for policy evaluation places a number of linear constraints on the return J^π of the policy from step 4.2. Write out that system of linear equations in the matrix form.

Answer:

(1) Note: the min operator for the first step does not have any effects for our MDP since both actions have the same Q-value at initialization.

$$Q^2(W, A) = 0.8 * 0.3 * 5 = 1.2$$
$$Q^2(W, D) = 0.8 * 0.2 * 0.1 = 0.16$$
$$Q^2(L, A) = 1 + 0.8 * (0.7 * 1 + 0.1 * 5) = 1.96$$
$$Q^2(L, D) = 1 + 0.8(0.4 * 1 + 0.3 * 5) = 2.52$$
$$Q^2(T, A) = 5 + 0.8 * 0.9 * 1 = 5.72$$
$$Q^2(T, D) = 5 + 0.8 * 0.75 * 1 = 5.6$$

Iter. k	$Q^k(W, A)$	$Q^k(W, D)$	$Q^k(L, A)$	$Q^k(L, D)$	$Q^k(T, A)$	$Q^k(T, D)$
1	0	0	1	1	5	5
2	1.2	0.16	1.96	2.52	5.72	5.6

(2) $\pi(W) =$ Defend, $\pi(L) =$ Attack, $\pi(T) =$ Defend. This is the not optimal policy because even if $\pi^2 = \pi^3$ for all states, the policies were from Q iteration and Q iteration need not have converged yet. If we were doing policy iteration, then we could have claimed that the policy was optimal.

(3)

$$\begin{bmatrix} J(W) \\ J(L) \\ J(T) \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 5 \end{bmatrix} + 0.8 \begin{bmatrix} 0.8 & 0.2 & 0 \\ 0.2 & 0.7 & 0.1 \\ 0.25 & 0.75 & 0 \end{bmatrix} \begin{bmatrix} J(W) \\ J(L) \\ J(T) \end{bmatrix} \quad (1)$$

END OF EXAM