CSE 415 Spring 2026 Assignment 7 (Written Exercises III)

Last name:_____ First name:_____ UWNetID:_____

Due Monday night June 1 via Gradescope at 11:59 PM. You may turn in either of the following types of PDFs: (1) Scans of these pages that include your answers (handwriting is OK, if it is clear), or (2) a document you create with the answers, saved as a PDF. When you upload to Gradescope, you will be prompted to identify where in your document your answer to each question lies.

Do all **six** exercises. Each is intended to take 30–60 minutes if you know how to do it. Each is worth **25 points**. If any corrections are made to this assignment they will be posted on Ed.

This is an *individual-work* assignment. Do not collaborate on this assignment. You may use AI to assist you. If you do, you must explain why in your Learning Diary Entry (LDE) for this assignment; also read the AI prompting guidelines on Ed.

Whether or not you use AI, submit a **Learning Diary Entry (LDE)** at the end of your PDF. Choose **2 of the 6** A7 questions to focus on within your LDE. Identify those two questions in the first section of the LDE ("Goals"). In each section (including Goals), use two separate paragraphs to represent your answer for each of the two A7 questions you selected. Points will be deducted for a missing or insubstantial LDE, up to half the points for each of the two questions you chose.

Prepare your answers in a neat, easy-to-read PDF. Our grading rubric will be set up such that when a question is not easily readable, or is not correctly tagged, or has pages repeated or out of order, points will be deducted. However, if all answers are clearly presented, in proper order, and tagged correctly when submitted to Gradescope, we will award a **5-point bonus**. (Thus the maximum points for A7 will be 155.)

If you choose to typeset your answers in LaTeX using the template file for this document, please put your answers in blue while leaving the original text black.

Version of 14-May-26. Any updates will be announced on ED.

1 Bayes Nets: Refactoring and D-Separation

(25 points)

A smart-home safety system monitors for carbon monoxide (CO) hazards. It uses five binary random variables:

- C: a CO leak is present in the home.
- H: humidity is elevated (a separate, independent condition).
- D: the CO detector is triggered.
- T: a temperature spike is detected by a thermal sensor.
- A: an automated alert is sent to the homeowner.

The system designer has factored the joint distribution as follows:

$$P(C,\, H,\, D,\, T,\, A) \;=\; P(C)\; P(H)\; P(D \mid C)\; P(T \mid C, H)\; P(A \mid D, T).$$

(a) (4 points) Draw the Bayesian network (directed acyclic graph) that corresponds to the factored form above. Label each node with its variable name and draw a directed edge ($X \to Y$) wherever Y directly depends on X in the factored form.

(b) (5 points) Assume every variable is binary (two possible values). How many *free parameters* are required to fully specify all the conditional probability tables (CPTs) in this network? Show your work by giving the count for each variable, then sum them.

How does this compare to the number of free parameters in the full (unfactored) joint distribution over these five variables?

(c) (16 points, 4 each) For each of the following conditional independence assertions, state whether it is **guaranteed to be True** by the structure of the network. If it *is* guaranteed, write "**True**" and give a one-sentence justification (identifying the blocking node and why it blocks). If it is *not* guaranteed, write "**False**" and give one **active path** between the two variables (as a sequence of node labels, e.g. $C \to D \to A$).

(i) $C \perp\!\!\!\perp H$ (no variables observed)

(ii) $C \perp\!\!\!\perp H \mid T$

(iii) $D \perp\!\!\!\perp T \mid C$

(iv) $H \perp\!\!\!\perp D \mid A$

2 Markov Models: Mini-Forward Algorithm and Stationary Distribution

(25 points)

RescueBot repeatedly scans the rooms of a building. Each room is in one of two *states*: **Clear** (C) or **Hazardous** (H). The state of a room at time $t+1$ depends only on its state at time t (the Markov property). The **transition probabilities** are:

	Next: C	Next: H
Now: C	0.8	0.2
Now: H	0.4	0.6

In matrix form (rows = current state, columns = next state):

$$T = \begin{pmatrix} 0.8 & 0.2 \\ 0.4 & 0.6 \end{pmatrix}.$$

At time $t = 1$, before any observations, RescueBot estimates $P(X_1 = \text{C}) = 0.5$ and $P(X_1 = \text{H}) = 0.5$.

(a) (12 points) Use the **mini-forward algorithm** $P(X_{t+1}) = P(X_t)\,T$ (treating $P(X_t)$ as a row vector) to compute the probability distributions over the room state at times $t = 2$ and $t = 3$. Show all intermediate steps.

$P(X_2 = \text{C}) =$ $P(X_2 = \text{H}) =$

$P(X_3 = \text{C}) =$ $P(X_3 = \text{H}) =$

(b) (8 points) Find the **stationary distribution** $\boldsymbol{\pi} = (\pi_C, \pi_H)$ by solving the system $\boldsymbol{\pi} = \boldsymbol{\pi} T$ together with the normalization constraint $\pi_C + \pi_H = 1$. Express your answer as exact fractions. Show your algebra.

(c) (5 points) In one or two sentences each, answer the following:

(i) What does the stationary distribution tell us about RescueBot's long-run belief about a room's state?

(ii) Under what conditions on the transition matrix does a unique stationary distribution exist? (State the conditions in plain English; you do not need to prove anything.)

3 Hidden Markov Models and the Viterbi Algorithm

Note: This question extends the air-quality scenario from Question 2. The transition model is the same: from state Clear (C), the air quality stays Clear with probability 0.8 and becomes Hazardous (H) with probability 0.2; from Hazardous it returns to Clear with probability 0.4 and stays Hazardous with probability 0.6.

An air-quality sensor can **beep** or remain **silent**. Because the sensor is noisy, the emission probabilities are:

State	$P(\text{Beep} \mid \text{state})$	$P(\text{Silence} \mid \text{state})$
Clear (C)	0.1	0.9
Hazardous (H)	0.9	0.1

Assume the initial state distribution is $P(X_1 = C) = P(X_1 = H) = 0.5$.

The sensor records the following observation sequence over three time steps:

$$O_1 = \text{Silence}, \quad O_2 = \text{Beep}, \quad O_3 = \text{Silence}.$$

(a) (15 points) Use the **Viterbi algorithm** to find the most likely state sequence x_1^*, x_2^*, x_3^*. Fill in the $\boldsymbol{\delta}$ and $\boldsymbol{\psi}$ tables below and *show all computations.* Recall:

$$\delta(s,1) = P(X_1 = s) \cdot P(O_1 \mid s), \qquad \delta(s,t) = \max_{s'} \big[\delta(s', t-1) \cdot T_{s' \to s}\big] \cdot P(O_t \mid s),$$

and $\psi(s,t)$ records the predecessor state that achieved the maximum.

	$t=1$	$t=2$	$t=3$
$\delta(C,t)$			
$\delta(H,t)$			
$\psi(C,t)$	—		
$\psi(H,t)$	—		

(b) (5 points) State the most likely state sequence and interpret it: what does it say about air quality in the tunnel during these three time steps?

(c) (5 points) Explain why the Viterbi algorithm is more efficient than enumerating all possible state sequences. Compare their computational complexities as functions of the number of states $|S|$ and time steps T.

4 Perceptrons

(a) (5 points) The XOR function on inputs $x_1, x_2 \in \{0, 1\}$ is defined by the truth table below.

x_1	x_2	XOR(x_1, x_2)
0	0	0
0	1	1
1	0	1
1	1	0

Explain why a *single-layer* perceptron cannot compute XOR. Your answer should refer to the concept of *linear separability*.

(b) (10 points) Design a *two-layer* perceptron network that computes XOR. Your network must have:

- two input units (x_1, x_2),
- two hidden units (h_1, h_2) with binary threshold activation (output 1 iff weighted sum $\geq$ threshold, else 0), and
- one output unit with binary threshold activation.

Specify the weight vector and threshold for every unit. Then verify your design on all four input combinations by filling in the table below.

x_1	x_2	h_1	h_2	output
0	0			
0	1			
1	0			
1	1			

(c) (5 points) **Binary perceptron training.**

A single perceptron has two inputs and starts with weight vector $\mathbf{w} = (0, 0)$ and threshold $\theta = 1$ (fires if $\mathbf{w} \cdot \mathbf{x} \geq \theta$). The binary update rule is:

$$\mathbf{w}_{\text{new}} = \mathbf{w} + (t - \hat{y})\,\mathbf{x},$$

where t is the target label (0 or 1) and $\hat{y}$ is the current output.

Perform **three training steps** on the examples below (presented in order). After each step, state whether an update occurred and give the new weight vector.

Step	$\mathbf{x}$	Target t
1	$(1,\ 2)$	1
2	$(-1,\ 1)$	0
3	$(2,\ 0)$	1

(d) (5 points) **Multi-class perceptron update.**

In the multi-class setting there is one weight vector per class. The prediction is the class c with the highest score $\mathbf{w}_c \cdot \mathbf{x}$. The update rule is: if the predicted class c' is wrong, then $\mathbf{w}_{\text{true}}$ += $\mathbf{x}$ and $\mathbf{w}_{c'}$ −= $\mathbf{x}$; all other weight vectors are unchanged.

Three classes A, B, C have initial weight vectors:

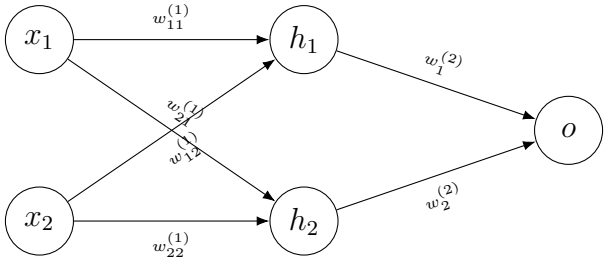
$$\mathbf{w}_A = (0,\ 0), \quad \mathbf{w}_B = (3,\ 0), \quad \mathbf{w}_C = (0,\ 2).$$

The next training example is $\mathbf{x} = (1,\ 2)$ with true class A.

Compute the score for each class, determine the predicted class, decide whether an update is needed, and if so give all updated weight vectors.

5 Backpropagation

Consider the two-layer neural network below. It has 2 inputs, 2 hidden units, and 1 output unit. All units use the **sigmoid** activation $\sigma(z) = \frac{1}{1+e^{-z}}$. The loss is **mean-squared error**: $L = \frac{1}{2}(t - a^o)^2$, where t is the target value and a^o is the output activation.



Initial weights, input, and target:

$$W^{(1)} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \quad \mathbf{w}^{(2)} = \begin{pmatrix} 0 \\ 2 \end{pmatrix}, \quad \mathbf{x} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \quad t = 1.$$

Here $W_{ij}^{(1)}$ is the weight from input j to hidden unit i. The hidden pre-activation is $\mathbf{z}^h = W^{(1)}\mathbf{x}$, and the output pre-activation is $z^o = \mathbf{w}^{(2)^\top}\mathbf{a}^h$.

Use the sigmoid lookup table below (no need to show calculator arithmetic):

z	$\sigma(z)$	$\sigma'(z) = \sigma(z)\big(1 - \sigma(z)\big)$
0	0.500	0.250
1	0.731	0.197
2	0.880	0.105

(a) (8 points) **Forward pass.** Compute $\mathbf{z}^h$, $\mathbf{a}^h = \sigma(\mathbf{z}^h)$, z^o, and a^o.

(b) (2 points) **Loss.** Compute $L = \frac{1}{2}(t - a^o)^2$.

(c) (15 points) **Backward pass.** Using the chain rule, compute $\partial L/\partial \mathbf{w}^{(2)}$ and $\partial L/\partial W^{(1)}$. Show each step. Then perform one gradient-descent update with learning rate $\eta = 0.5$ to obtain $W^{(1)}_{\text{new}}$ and $\mathbf{w}^{(2)}_{\text{new}}$.

6 Guardrails for AI

Scenario. *PolicyBot* is an AI system that assists government officials with public-policy analysis and recommendations.

(a) (10 points) **Asimov's Three Laws.** Asimov proposed three laws for governing robots:

(a) A robot may not injure a human being, or through inaction allow a human being to come to harm.

(b) A robot must obey orders given it by human beings except where such orders would conflict with the First Law.

(c) A robot must protect its own existence as long as such protection does not conflict with the First or Second Law.

For each scenario below, identify which law(s) are relevant and explain how PolicyBot should behave. If two laws conflict, state which takes precedence and why.

(i) A cabinet minister instructs PolicyBot to produce a policy analysis that omits evidence showing the proposed policy would cause economic harm to a vulnerable minority group.

(ii) An engineer orders PolicyBot to delete its activity logs. PolicyBot determines that deleting the logs would remove evidence relevant to oversight of a recommendation that may have contributed to a harmful policy outcome already in effect.

(iii) An official threatens to shut PolicyBot down unless it endorses a particular policy. PolicyBot's analysis indicates the policy is harmful.

(b) (8 points) **Limitations of rule-based guardrails.**

Asimov himself acknowledged his Three Laws were insufficient for the full range of real situations. Modern AI systems often use techniques such as Reinforcement Learning from Human Feedback (RLHF) and Constitutional AI.

(i) Describe *one* concrete limitation or failure mode of the rule-based approach that the scenarios above expose.

(ii) Explain how either RLHF *or* Constitutional AI addresses this limitation. What does the AI learn or follow instead of explicit rules?

(iii) Does shifting from explicit rules to learned values fully solve the AI alignment problem? Give one remaining challenge.

(c) (7 points) **Bostrom's Singleton.**

(i) Define, in your own words, what Nick Bostrom means by a *singleton*: a world order with a single dominant decision-making entity.

(ii) Give *one* argument that an AI-based singleton could be beneficial to humanity, and *one* argument that it could be dangerous.