CS 188 Summer 2019 Introduction to Artificial Intelligence — Final

- You have approximately 170 minutes.
- The exam is closed book, closed calculator, and closed notes except your two-page sheet of note.
- Mark your answers ON THE EXAM ITSELF. If you are not sure of your answer you may wish to provide a *brief* explanation. All short answer sections can be successfully answered in a few sentences AT MOST.
- For multiple choice questions,
  - ☐ means mark **all options** that apply
  - ◯ means mark a **single choice**
  - When selecting an answer, please fill in the bubble or square **completely** (● and ■)

First name	
Last name	
SID	
Student to your right	
Student to your left	

Your Discussion TA (fill in all that apply):

☐ Caryn (TTh 10:00 am Wheeler)
☐ Arin (TTh 10:00 am Dwinelle)
☐ Bobby (MW 3:00 pm)
☐ Mike (TTh 11:00 am)
☐ Benson (MW 4:00 pm)
☐ Did not attend any

For staff use only:

Q1.	They Hatin', Patrolling (Search Formulation)	/17
Q2.	Search	/17
Q3.	M-O-D-E Pruning	/19
Q4.	LQR	/15
Q5.	Approximate Q-Learning	/12
Q6.	Probability and Bayes Net Representation	/23
Q7.	Decode Your Terror (HMM)	/18
Q8.	Raisins Again (VPI)	/12
Q9.	Machine Learning: Potpourri	/19
Q10.	Perceptrons and Naive Bayes	/24
Q11.	Neural Network	/24
	Total	/200

THIS PAGE IS INTENTIONALLY LEFT BLANK

SID: _____

Q1. [17 pts] They Hatin', Patrolling (Search Formulation)

Recall that in Midterm 1, Pacman bought a car, was speeding in Pac-City, and the police weren't able to catch him. Now Pacman has run out of gas, his car has stopped, and he is currently hiding out at an undisclosed location.

In this problem, you are on the police side, tryin' to catch Pacman!

There are still p police cars in the Pac-city of dimension m by n. In this problem, **all police cars can move, with two distinct integer controls: throttle and steering, but Pacman has to stay stationary**. Once one police car takes an action which lands him in the same grid as Pacman, Pacman will be arrested and the game ends.

Throttle: $t_i \in \{1, 0, -1\}$, corresponding to {Gas, Coast, Brake}. This controls the **speed** of the car by determining its acceleration. The integer chosen here will be added to his velocity for the next state. For example, if a police car is currently driving at 5 grid/s and chooses Gas (1) he will be traveling at 6 grid/s in the next turn.
Steering: $s_i \in \{1, 0, -1\}$, corresponding to {Turn Left, Go Straight, Turn Right}. This controls the **direction** of the car. For example, if a police car is facing North and chooses Turn Left, it will be facing West in the next turn.

(a) Suppose you can **only control 1 police car**, and have absolutely no information about the remainder of $p-1$ police cars, or where Pacman stopped to hide. Also, the police cars can travel up to 6 grid/s so $0 \leq v \leq 6$ at all times.

(i) [4 pts] What is the **tightest upper bound** on the size of state space, if your goal is to use search to plan a sequence of actions that guarantees Pacman is caught, no matter where Pacman is hiding, or what actions other police cars take. Please note that your state space representation must be able to represent **all** states in the search space.

$28mn * 2^{mn}$

There are mn positions in total. At each legal position, there are 7 possible speeds (0, 1, 2, 3, 4, 5, 6), so a factor of 7 is multiplied. In addition, since change of direction depends on orientation of the car, another factor of 4 is multiplied.

The only sequence of actions which guarantees that Pacman is caught is a sequence of actions which visits every location. Thus, we also need to a list of $m * n$ boolean to keep track of whether we have visited a specific grid location, and that is another factor of 2^{mn}

(ii) [3 pts] What is the maximum branching factor? Your answer may contain integers, m, n.

9

3 possible throttle inputs, and 3 possible steering inputs. The list of boolean does not affect the branching factor.

(iii) [2 pts] Which algorithm(s) is/are guaranteed to return a path passing through all grid locations on the grid, if one exists?

- ☐ Depth First Tree Search
- ■ Breadth First Tree Search
- ■ Depth First Graph Search
- ■ Breadth First Graph Search

Please note the list of boolean is in the state space representation, so we can revisit the same grid position if we have to.

(iv) [2 pts] Is Breadth First Graph Search guaranteed to return the path with the shortest number of **time steps**, if one exists?

- ● Yes
- ○ No

The Breadth First Graph Search is guranteed to return the path with the shortest amount of time, because each edge here represent moving for 1 unit of time.

(b) Now let's suppose you can control **all** p police cars at the same time (and know all their locations), but you still have no information about where Pacman stopped to hide

(i) [3 pts] Now, you still want to search a sequence of actions such that the paths of p police car combined **pass through all** $m * n$ **grid locations**. Suppose the size of the state space in part (a) was N_1, and the size of the state space in this part is N_p. Please select the correct relationship between N_p and N_1

○ $N_p = p * N_1$ ○ $N_p = p^{N_1}$ ○ $N_p = (N_1)^p$ ● None of the above

In this question, we only need one boolean list of size mn to keep track of whether we have visited a specific grid location. So the size of the state space is bounded by $N_p = (28mn)^p 2^{mn}$, which is none of the above.

(ii) [3 pts] Suppose the maximum branching factor in part (a) was b_1, and the maximum branching factor in this part is b_p. Please select the correct relationship between b_p and b_1

○ $b_p = p * b_1$ ○ $b_p = p^{b_1}$ ● $b_p = (b_1)^p$ ○ None of the above

For example, the case of $p = 2$ means two cars can do all 9 options, so the branching factor is $9^2 = 81$. In general, the branching factor is then b_1^p.

SID: _____

Q2. [17 pts] Search

For this problem, assume that all of our search algorithms use tree search, unless specified otherwise.

(a) For each algorithm below, indicate whether the path returned after the modification to the search tree is guaranteed to be identical to the unmodified algorithm. Assume all edge weights are non-negative before modifications.

(i) [3 pts] Adding additional cost $c > 0$ to every edge weight.

	Yes	No
BFS	●	○
DFS	●	○
UCS	○	●

(ii) [3 pts] Multiplying a constant $w > 0$ to every edge weight.

	Yes	No
BFS	●	○
DFS	●	○
UCS	●	○

(b) **For part (b)**, two search algorithms are defined to be **equivalent** if and only if they expand the same states in the same order and return the same path. **Assume all graphs are directed and acyclic.**

(i) [3 pts] Assume we have access to costs c_{ij} that make running UCS algorithm with these costs c_{ij} equivalent to running BFS. How can we construct new costs c'_{ij} such that running UCS with these costs is equivalent to running DFS? Mark all correct choices.

○ $c'_{ij} = 0$ ○ $c'_{ij} = 1$ ○ $c'_{ij} = c_{ij}$

● $c'_{ij} = -c_{ij}$ ○ $c'_{ij} = c_{ij} + \alpha$ ○ Not possible

Breadth-First Search expands the node at the shallowest depth first. Assigning a constant positive weight to all edges allows to weigh the nodes by their depth in the search tree. Depth-First Search expands the nodes which were most recently added to the fringe first. Assigning a constant negative weight to all edges essentially allows to reduce the value of the most recently nodes by that constant, making them the nodes with the minimum value in the fringe when using uniform cost search. Hence, we can construct new costs c'_{ij} by flipping the sign of the original costs c_{ij}.

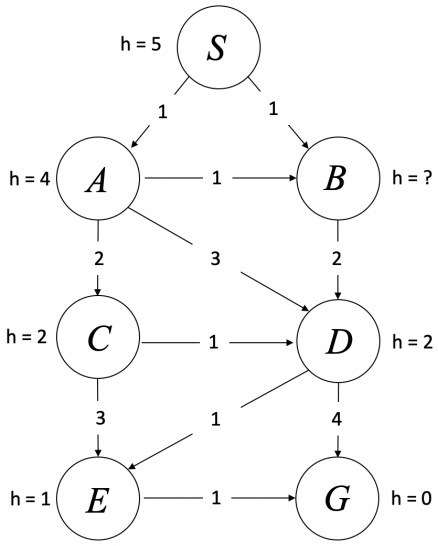
(ii) [3 pts] Given edge weight $c_{ij} = h(j) - h(i)$, where $h(n)$ is the value of the heuristic function at node n, running UCS on this graph is equivalent to running which of the following algorithm(s) on the same graph? Select all correct answers.

○ DFS ○ BFS ○ Iterative Deepening

● Greedy ○ A* ○ None of the above.

Greedy Search expands the node with the lowest heuristic function value $h(n)$. If $c_{ij} = h(j) - h(i)$, then the cost of a node n on the fringe when running uniform-cost search will be $\sum_{ij} c_{ij} = h(1) - h(start) + h(2) - h(1) + ... + h(n) - h(n-1) = h(n) - h(\text{start})$. As $h(\text{start})$ is a common constant subtracted from the cost of all nodes on the fringe, the relative ordering of the nodes on the fringe is still determined by $h(n)$, i.e. their heuristic values.

(c) Consider the following graph. $h(n)$ denotes the heuristic function evaluated at node n.



(i) [2 pts] Given that G is the goal node, and heuristic values are fixed for all nodes other than B, for which values of $h(B)$ will A* tree search be guaranteed to return the optimal path? Fill in the lower and upper bounds or select "impossible."

__0__ $\leq h(B) \leq$ __4__ ○ Impossible

An admissible heuristic is sufficient for A* tree search to be optimal. Recall the constraint for an admissible heuristic: $\forall n, 0 \leq h(n) \leq h^*(n)$. Since all other nodes satisfy the constraint above, we just need to enforce the constraint on node B, so that $0 \leq h(B) \leq h^*(B) = 4$.

(ii) [3 pts] With the heuristic values fixed for all nodes other than B, for which values of $h(B)$ will A* graph search be guaranteed to return the optimal path? Either fill in the lower and upper bound or select "impossible."

__4__ $\leq h(B) \leq$ __4__ ○ Impossible

We need a consistent heuristic for A* tree search to be optimal. Recall the constraint for a consistent heuristic: $\forall A, C \quad h(A) - h(C) \leq \text{cost}(A, C)$. Since all other nodes satisfy the constraint above, we just need to enforce the constraint on node B. Enforcing consistency along $S - B$, gives $h(B) \geq 4$, $A - B$ gives $h(B) \geq 3$, $B - D$ gives $h(B) \leq 4$.

SID: _____

Q3. [19 pts] M-O-D-E Pruning

For all parts of this question, write your final answer in the small box, but we encourage you to show relevant work in the big box. A correct final answer is sufficient for full credit.

NOTE: In the pruning process, we are only concerned about returning the correct value of the root node.

(a) Minimax Tree Pruning

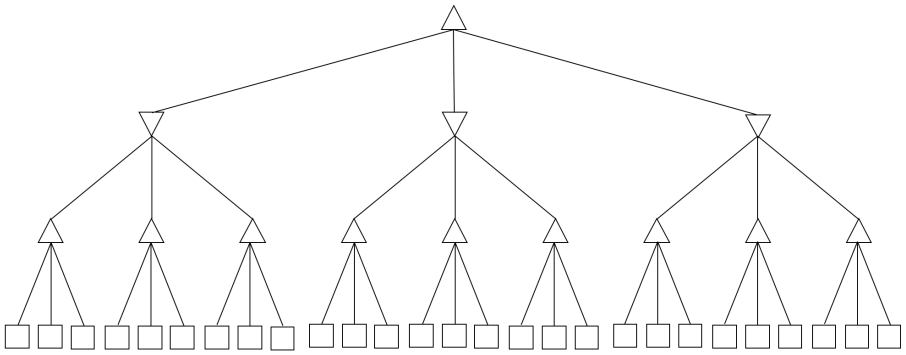


Figure 1: Minimax Tree with $b = 3, d = 3$

Suppose we have a complete minimax tree (with exactly 1 maximizer and 1 minimizer, alternating), where all nodes have branching factor b, and d total layers (an extra layer of maximizer is added, if d is not even)

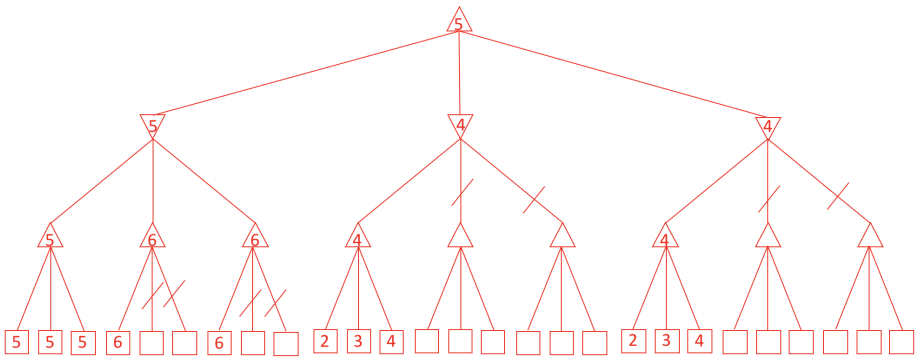
(i) [2 pts] In the case of $b = 3, d = 3$, what is the total number of **value nodes** (denoted by squares at the lowest layer)? Your final answer should be an integer.

Answer: 27

Count the diagram

(ii) [3 pts] In the case of $b = 3, d = 3$, what is the **maximum** total number of value nodes **whose value is never explored because an upstream branch is pruned** in one single set of values for the values nodes. Your final answer should be an integer.

Answer: 16



(iii) [3 pts] Now we are changing the branching factor b, keeping $d = 3$. In the case of $b = 5, d = 3$, what is the **maximum** total number of value nodes **whose value is never explored because an upstream branch is pruned** in one single set of values for the value nodes. Your final answer should be an integer.

Answer: 96

There are 5^3 value nodes.

1. The first sub-tree (lower maximizer in the diagram) of all sub-trees in the second-to-last layer (minimizer in the diagram) must have all child value nodes explored. That is 5^2 value nodes must be explored.

2. The remaining subtree of the first subtree in the second-to-last layer (minimizer with value 5 in previous diagram) must have the first child value nodes explored. That is an additional $(5-1) = 4$ value nodes must be explored.

So the total number of nodes unexplored is $5^3 - 5^2 - 4 = 96$

(b) Mode Tree Pruning

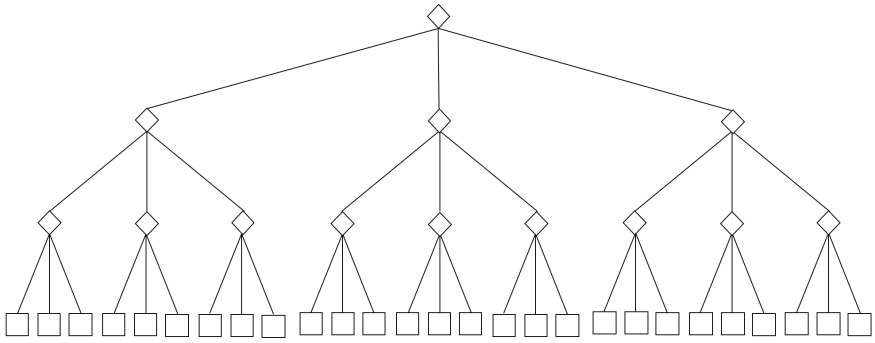


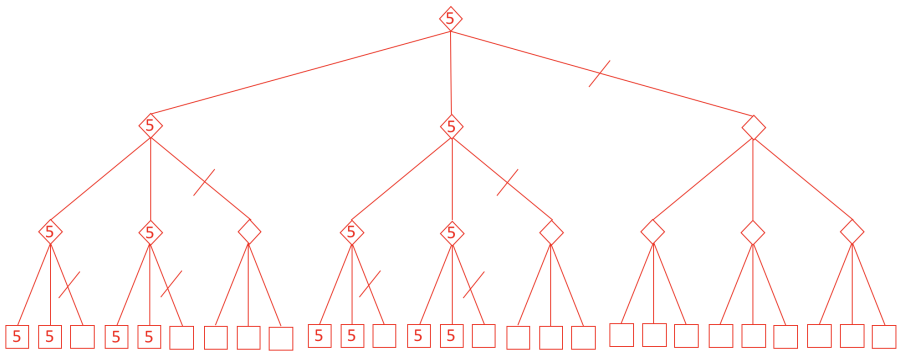
Figure 2: Mode Tree with $b = 3, d = 3$

We all know α-β pruning on minimax tree reduces the number of nodes explored to minimum, but what if we change all the maximizer and minimizer nodes to **mode nodes** (symbolized by spades in the diagram), where mode value (most frequent value) among the child nodes will be selected?

Important: Please note that, just like in $\alpha - \beta$ pruning, a child node n_c can be pruned when its parent node n_p are absolutely sure that the value of n_c will have no influence on the final value of the root node.

(i) [3 pts] In the case of $b = 3, d = 3$, what is the **maximum** total number of value nodes **whose value is never explored because an upstream branch is pruned** in one single set of values for the values nodes. Your final answer should be an integer.

Answer: 19



(ii) [4 pts] Now we are changing the branching factor b, keeping $d = 3$. In the case of $b = 5, d = 3$, what is the **maximum** total number of value nodes **whose value is never explored because an upstream branch is pruned** in one single set of values for the values nodes. Your final answer should be an integer.

Answer: 98

There are 5^3 value nodes. Since it takes 3 to guarantee a majority in a branching factor of 5, only the first 3 children of every sub-tree absolutely need to be explored.

1. The first layer, we absolutely need to explore the first 3 subtree of the root node
2. The second layer, we need to explore the first 3 children of the 3 needed nodes in the first layer. So there are $3^2 = 9$ needed nodes on the second layer that we guarantee to consider.
3. The last layer (value node layer), we again need to explore the first 3 children of the 9 needed nodes in the second layer. So there are $3^3 = 27$ value nodes that we guarantee to consider.

So the total number of nodes unexplored is $5^3 - 3^3 = 98$

(iii) [4 pts] Now we are changing the depth d, keeping $b = 3$. In the case of $b = 3, d = 4$, what is the **maximum** total number of value nodes **whose value is never explored because an upstream branch is pruned** in one single set of values for the values nodes. Your final answer should be an integer.

Answer: 65

There are $3^4 = 81$ value nodes. Since it takes 2 to guarantee a majority in a branching factor of 3, only the first 2 children of every sub-tree absolutely need to be explored.

1. The first layer, we absolutely need to explore the first 2 subtree of the root node
2. The second layer, we need to explore the first 2 children of the 2 needed nodes in the first layer. So there are $2^2 = 4$ nodes on the second layer that we guarantee to consider.
3. The third layer, we need to explore the first 2 children of the 4 needed nodes in the first layer. So there are $2^3 = 8$ nodes on the second layer that we guarantee to consider.
4. The last layer (value node layer), we again need to explore the first 2 children of the 8 needed nodes in the second layer. So there are $2^4 = 16$ value nodes that we guarantee to consider.

So the total number of nodes unexplored is $3^4 - 2^4 = 65$

Q4. [15 pts] LQR

For all parts of this question, write your final answer in the small box, but we encourage you to show relevant work in the big box. A correct final answer is sufficient for full credit.

Consider the following deterministic MDP with 1-dimensional continuous states and actions and a finite task horizon:

State Space $\mathcal{S}$: $\mathbb{R}$
Action Space $\mathcal{A}$: $\mathbb{R}$
Reward Function: $R(s, a, s') = -qs^2 - ra^2$ where $r > 0$ and $q \geq 0$
Deterministic Dynamics/Transition Function: $s' = cs + da$ (i.e., the next state s' is a deterministic function of the action a and current state s)
Task Horizon: $T \in \mathbb{N}$
Discount Factor: $\gamma = 1$ (no discount factor)

Hence, we would like to maximize a quadratic reward function that rewards small actions and staying close to the origin. In this problem, we will design an optimal agent π_t^* and also solve for the optimal agent's value function V_t^* for all timesteps.

By induction, we will show that V_t^* is quadratic. Observe that the base case $t = 0$ trivially holds because $V_0^*(s) = 0$ For all parts below, assume that $V_t^*(s) = -p_t s^2$ (Inductive Hypothesis).

(a) **(i)** [5 pts] Write the equation for $V_{t+1}^*(s)$ as a function of s, q, r, a, c, d, and p_t. If your expression contains max, you do not need to simplify the max.

$V_{t+1}^*(s) =$ $\max_{a \in \mathbb{R}} -qs^2 - ra^2 - p_t(cs + da)^2$

$$
\begin{aligned}
V_{t+1}^*(s) &= \max_{a \in \mathbb{R}} R(s, a, s') + V_t^*(s') \\
&= \max_{a \in \mathbb{R}} -qs^2 - ra^2 - p_t(s')^2 \\
&= \max_{a \in \mathbb{R}} -qs^2 - ra^2 - p_t(cs + da)^2
\end{aligned}
$$

(ii) [5 pts] Now, solve for $\pi_{t+1}^*(s)$. Recall that you can find local maxima of functions by computing the first derivative and setting it to 0.

$\pi^*_{t+1}(s) =$ $-\frac{cdp_t}{r+p_td^2}s$

We want to solve for the a that maximizes the value of $V^*_{t+1}(s)$. This is equivalent to maximizing $-ra^2 - p_{t+1}(cs+da)^2$. Differentiate this function, set it to 0 and solve for a.

$$\frac{d}{da}[-ra^2 - p_t(cs+da)^2] = -2ra - 2(cs+da)p_td = 0$$

Solving for a, we find that:

$$\pi^*_{t+1}(s) = -\frac{cdp_t}{r+p_td^2}s = k_{t+1}s$$

Observe that the optimal policy is a linear function of the state.

(b) [5 pts] Assume $\pi^*_{t+1} = k_{t+1}s$ for some $k_{t+1} \in \mathbb{R}$. Solve for p_{t+1} in $V^*_{t+1}(s) = -p_{t+1}s^2$.

$p_{t+1} =$ $q + rk_{t+1}^2 + p_t(c + dk_{t+1})^2$

$$\begin{aligned} V^*_{t+1}(s) &= R(s, \pi^*_{t+1}(s), s') + V^*_t(s') \\ &= -qs^2 - r(k_{t+1}s)^2 - p_t s'^2 \\ &= -qs^2 - r(k_{t+1}s)^2 - p_t(cs + d(k_{t+1}s))^2 \\ &= -(q + rk_{t+1}^2 + p_t(c + dk_{t+1})^2)s^2 \\ &= -p_{t+1}s^2 \end{aligned}$$

Q5. [12 pts] Approximate Q-Learning

For all parts of this question, write your final answer in the small box, but we encourage you to show relevant work in the big box. A correct final answer is sufficient for full credit.

We would like to train a Q function Q_w parameterized by some trainable weights w. On observing a sample (s, a, r, s') from the environment, we compare the old estimate of the Q-value of (s, a): $\hat{y} = Q_w(s, a)$ and the new estimate: $y = r + \max_{a'} Q_w(s', a')$ and use it to update w to minimize the squared error via gradient descent.

$$\begin{aligned} error(\hat{y}, y) &= (\hat{y} - y)^2 \\ &= (Q_w(s, a) - y)^2 \end{aligned}$$

Treating the target y as a constant, differentiate $error(\hat{y}, y)$ with respect to w to derive the weight update rule for w_i:

$$w_i \leftarrow w_i - \alpha \frac{\partial error}{\partial w_i}(\hat{y}, y)$$

Let's derive the update rules for different types of Q functions. **For all parts, leave your answers in terms of $y, \hat{y}, f_i(s, a)$, and w_i.**

(a) [4 pts] The Q-function we would like to train has the form $Q_w(s, a) = \sum_{i=0}^{m} w_i f_i(s, a)$. Please derive $\frac{\partial error}{\partial w_i}(\hat{y}, y)$ for this Q function.

$\frac{\partial error}{\partial w_i}(\hat{y}, y) =$ $2(\hat{y} - y) f_i(s, a)$

Other acceptable solutions are $2(\sum_{i=0}^{m} w_i f_i(s, a) - y) f_i(s, a)$, $2(Q_w(s, a) - y) f_i(s, a)$

(b) The Q-function we would like to train now has the form

$$Q_w(s,a) = \sum_{i=0}^{m} e^{w_i f_i(s,a)} + \sum_{i=m+1}^{2m} w_i^2 f_i(s,a)$$

.

(i) [4 pts] For $i \in \{1, \ldots, m\}$, derive $\frac{\partial error}{\partial w_i}(\hat{y}, y)$ for this Q function.

$\frac{\partial error}{\partial w_i}(\hat{y}, y) =$ $2(\hat{y} - y) f_i(s,a) e^{w_i f_i(s,a)}$

Other acceptable solutions are $2(\sum_{i=0}^{m} e^{w_i f_i(s,a)} + \sum_{i=m+1}^{2m} w_i^2 f_i(s,a) - y) f_i(s,a) e^{w_i f_i(s,a)}$, $2(Q_w(s,a) - y) f_i(s,a) e^{w_i f_i(s,a)}$

(ii) [4 pts] For $i \in \{m+1, \ldots, 2m\}$, derive $\frac{\partial error}{\partial w_i}(\hat{y}, y)$ for this Q function.

$\frac{\partial error}{\partial w_i}(\hat{y}, y) =$ $4(\hat{y} - y) w_i f_i(s,a)$

Other acceptable solutions are $4(\sum_{i=0}^{m} e^{w_i f_i(s,a)} + \sum_{i=m+1}^{2m} w_i^2 f_i(s,a) - y) w_i f_i(s,a)$, $4(Q_w(s,a) - y) w_i f_i(s,a)$

Q6. [23 pts] Probability and Bayes Net Representation

You're interested in knowing whether you would be **S**atisfied with your choice of snack(s), and so you decide to make the prediction using probabilistic inference over a model with the following variables:

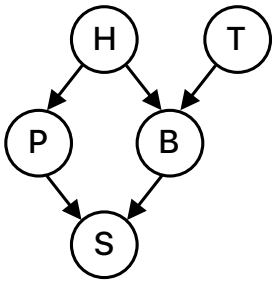
- S, whether or not you will be **S**atisfied.
- H, whether or not you will be **H**ungry.
- T, whether or not you will be **T**hirsty.
- P, whether or not you will have **P**izza.
- B, whether or not you will have **B**oba.

Each of the variables may take on two values: *yes* or *no*.

(a) [1 pt] Your first idea for a probability model is a joint probability table over all of the variables. What's the **minimum** number of parameters you need to fully specify this joint probability distribution?

$2^5 - 1 = 31$

(b) [3 pts] You decide this is too many parameters. To fix this, you decide to model the problem with the following Bayes net instead:



$\Pr(H)$	
$+h$	0.7
$-h$	0.3

$\Pr(T)$	
$+t$	0.6
$-t$	0.4

$\Pr(P\|H)$		
$+p$	$+h$	0.8
$+p$	$-h$	0.5
$-p$	$+h$	0.2
$-p$	$-h$	0.5

$\Pr(B\|H,T)$			
$+b$	$+h$	$+t$	0.4
$+b$	$+h$	$-t$	0.2
$+b$	$-h$	$+t$	0.9
$+b$	$-h$	$-t$	0.5
$-b$	$+h$	$+t$	0.6
$-b$	$+h$	$-t$	0.8
$-b$	$-h$	$+t$	0.1
$-b$	$-h$	$-t$	0.5

$\Pr(S\|P,B)$			
$+s$	$+p$	$+b$	0.9
$+s$	$+p$	$-b$	0.4
$+s$	$-p$	$+b$	0.7
$+s$	$-p$	$-b$	0.1
$-s$	$+p$	$+b$	0.1
$-s$	$+p$	$-b$	0.6
$-s$	$-p$	$+b$	0.3
$-s$	$-p$	$-b$	0.9

You do not know which snack(s) you are going for, but you know you are both hungry, thirsty, and definitely getting Pizza. According to your model, what is the probability that you will be satisfied? (First, write out the expression *in terms of conditional probabilities from the model*; then, plug in the *values from the tables* and compute the final answer.)

0.6

$$\begin{aligned}\Pr(+s|+h,+t,+p) &= \sum_b \Pr(+s|+p,b)\cdot\Pr(b|+h,+t)\\ &= \Pr(+s|+p,+b)\cdot\Pr(+b|+h,+t)+\Pr(+s|+p,-b)\cdot\Pr(-b|+h,+t)\\ &= 0.9\times 0.4+0.4\times 0.6\\ &= 0.6\end{aligned}$$

(c) [3 pts] You thought the last part required too much computation so you decide to use rejection sampling, sampling variables in topological order. Write the probability of rejecting a sample for the following queries.

$P(+p \mid +h) =$ 0.3

$P(-s \mid +p) =$ $P(-p \mid +h)P(+h) + P(-p \mid -h)p(-h) = 0.2 \times 0.7 + 0.5 \times 0.3 = 0.29$

$P(+s \mid -h, +t) =$ $1 - P(-h, +t) = 1 - P(-h)P(+t) = 1 - 0.3 \times 0.6 = 0.82$

(d) Given that you are satisfied with your choice of snack(s), write out the variable elimination steps you would take to compute the probability that you actually had boba, that is, $\Pr(+b \mid +s)$. (You do **not** have to plug in the values from the tables.)

(i) [2 pts] Which of the following factors do we start with?

- ■ $\Pr(H)$
- ☐ $\Pr(H|P)$
- ☐ $\Pr(+s|P)$
- ■ $\Pr(T)$
- ■ $\Pr(P|H)$
- ☐ $\Pr(+s|B)$
- ☐ $\Pr(P)$
- ☐ $\Pr(B|H)$
- ☐ $\Pr(+s|P, H)$
- ☐ $\Pr(B)$
- ☐ $\Pr(B|T)$
- ☐ $\Pr(+s|P, H, B)$
- ☐ $\Pr(+s)$
- ■ $\Pr(B|H, T)$
- ■ $\Pr(+s|P, B)$

(ii) [1 pt] First, we eliminate H. What is the factor f_1 generated when we eliminate H?

- ○ $f_1(P)$
- ○ $f_1(P,\ B)$
- ● $f_1(P, B, T)$
- ○ $f_1(B)$
- ○ $f_1(P,\ T)$
- ○ $f_1(P, B, +s)$
- ○ $f_1(T)$
- ○ $f_1(P, +s)$
- ○ $f_1(B, T, +s)$
- ○ $f_1(+s)$
- ○ $f_1(B,\ T)$
- ○ $f_1(B, +s)$
- ○ $f_1(T, +s)$

(iii) [1 pt] Write out the expression for computing f_1 in terms of the remaining factor(s) (before H is eliminated).

$f_1(P, B, T) =$ $\sum_h \Pr(h) \Pr(P \mid h) \Pr(B \mid h, T)$

(iv) [2 pts] Next, we eliminate T. What is the factor f_2 generated when we eliminate T?

$f_2(P, B)$

Write out the expression for computing f_2 in terms of the remaining factor(s) (before T is eliminated).

$f_2(P, B) =$ $\sum_t \Pr(t) f_1(t, P, B)$

(v) [2 pts] Finally, we eliminate P. What is the factor f_3 generated when we eliminate P?

$f_3(B, +s)$

Write out the expression for computing f_3 in terms of the remaining factor(s) (before P is eliminated).

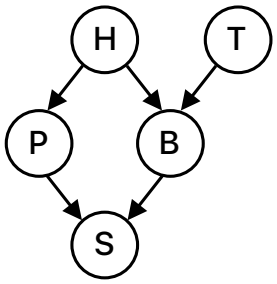
$f_3(B, +s) =$ $\sum_p \Pr(+s \mid p, B) f_2(p, B)$

(vi) [2 pts] Write out the expression for computing $\Pr(+b|+s)$ in terms of the remaining factor(s) (after P is eliminated).

$\Pr(+b \mid +s) =$ $\frac{f_3(+b,+s)}{\sum_b f_3(b,+s)}$

(e) Conditional Independence: For each of the following statements about conditional independence, mark if it is guaranteed by the Bayes Net.

The Bayes Net is reproduced below for your convenience.



(i) [1 pt] $H \perp\!\!\!\perp T$

● Guaranteed ○ Not guaranteed

(ii) [1 pt] $P \perp\!\!\!\perp T \mid B$

○ Guaranteed ● Not guaranteed

(iii) [1 pt] $H \perp\!\!\!\perp T \mid S$

○ Guaranteed ● Not guaranteed

(iv) [1 pt] $S \perp\!\!\!\perp T \mid B$

○ Guaranteed ● Not guaranteed

(v) [1 pt] $H \perp\!\!\!\perp S \mid P,\ B$

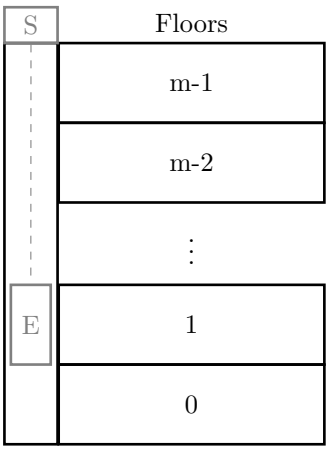
● Guaranteed ○ Not guaranteed

(vi) [1 pt] $P \perp\!\!\!\perp T \mid H,\ S$

○ Guaranteed ● Not guaranteed

Q7. [18 pts] Decode Your Terror (HMM)

You go to Disney and ride the famous Tower of Terror ride, where an elevator rises and drops seemingly at random. You're terrified, but vow to determine the sequence of rises and drops that make up the ride so you won't be as terrified next time. Assume the elevator E follows a Markovian process and it has m floors at which it can stop. In the dead of night, you install a sensor S at the top of the shaft that gives approximate distance measurements, quantized into n different distance bins. Assume that the elevator stops at T floors as part of the ride and the initial distribution of the elevator is uniform over the m floors.



You want to know the most probable sequence of (hidden) states $X_{1:T}$ given your observations $y_{1:T}$ from the sensor, so you turn to the Viterbi algorithm, which performs the following update at each step:

$$m_t[x_t] = P(y_t|x_t) \max_{x_{t-1}} [P(x_t|x_{t-1}) m_{t-1}[x_{t-1}]]$$
$$a_t[x_t] = \arg\max_{x_{t-1}} m_{t-1}[x_{t-1}]$$

(a) **(i)** [2 pts] What is the run time of the Viterbi algorithm to determine all previous states for this scenario? Please answer in big O notation, in terms of T, m, and n, or write "N/A" if the run time is unable to be determined with the given information.

Tm^2

For each timestep that we want to decode, we loop over all states and consider the transition probability to that state from all previous states. Thus, the runtime is Tm^2, since we repeat this process for each timestep (T), and at each timestep we have a double for loop for states (one to compute m-value for all states, one to compute arg max of previous m-values and transition probabilities, so m^2). For more details, see Note 6 (pg. 8).

(ii) [2 pts] What is the space complexity of the Viterbi algorithm to determine all previous states for this scenario? Please answer in big O notation, in terms of T, m, and n, or write "N/A" if the space complexity is unable to be determined with the given information.

Tm

We store only our a-value arrays (which keep track of the best transition along the path to the current

state). These are essentially back-pointers so we can reconstruct the best path. For a sequence of length T, we then have a matrix of these pointers that is $T \times m$, so we take up Tm space.

Eventually, we decide that the end of the ride is the exciting part, so we decide that we only wish to determine the previous K states.

(b) **(i)** [2 pts] What is the run time of the Viterbi algorithm to determine the previous K states? Please answer in big O notation, in terms of T, K, m, and n, or write "N/A" if the run time is unable to be determined with the given information.

Tm^2

The run time is the same as (a)(i) since the forward pass part of the Viterbi algorithm is unchanged (e.g., we still need to consider all transition probabilities at all timesteps to determine the most likely path, regardless of how far back we want to go).

(ii) [2 pts] What is the space complexity of the Viterbi algorithm to determine the previous K states? Please answer in big O notation, in terms of T, K, m, and n, or write "N/A" if the space complexity is unable to be determined with the given information.

Km

If we only wish to determine the previous K states, we only need to store a-values corresponding to the previous K timesteps (since when we reconstruct the path, we only need to go K steps back). Thus, the space complexity decreases to Km from part (a)(ii).

Suppose you instead only wish to determine the current distribution (at time T) for the elevator, given your T observations, so you use the forward algorithm, with update step shown here:

$$P(X_t|y_t) \propto P(y_t|x_t) \sum_{x_{t-1}} P(X_t|x_{t-1}) P(x_{t-1}, y_{0:t-1})$$

Additionally, from your previous analysis, you note that there are some states which are unreachable from others (e.g., the elevator cannot travel from the top floor to the bottom in a single timestep). Specifically, from each state, there are between $G/2$ and G states which can be reached in the next timestep, where $G < m$.

(c) **(i)** [2 pts] What is the run time for the forward algorithm to estimate the current state at time T, assuming we ignore states that cannot be reached in each update? Please answer in big O notation, in terms of T, m, G, and n, or write "N/A" if the run time is unable to be determined with the given information.

TmG

The normal run time for the forward algorithm is Tm^2 (very similar to Viterbi - we consider transition probabilities from each state to each other state during our forward pass). However, if we can ignore those states with zero transition probability, then we only need to consider G states when calculating our sum for each state. Thus, the run time is reduced to TmG.

(ii) [2 pts] What is the space complexity for the forward algorithm to estimate the current state at time T, assuming we ignore states that cannot be reached in each update? Please answer in big O notation, in terms of T, m, G, and n, or write "N/A" if the space complexity is unable to be determined with the

given information.

m

The space complexity for the forward algorithm is unchanged by how many states we can reach from each state, so this is the same as the normal space complexity for the forward algorithm. It is m because we store $P(X_t|y_t)$ for each state and we only store one timestep (since we only care about the current state probabilities).

Finally, assume that the number of elevator states is actually infinite (e.g., instead of stopping at floors, the elevator can stop at any point along the elevator shaft).

(d) [3 pts] What is the run time of the standard forward algorithm to estimate the current state at time T in this case? Please answer in big O notation, in terms of T, m, G, and n, or write "N/A" if the run time is unable to be determined with the given information.

N/A

From part (c)(i), the run time of the forward algorithm is proportional to the number of states m. Thus as $m \to \infty$, the run time grows to infinity. Both N/A and ∞ were accepted here.

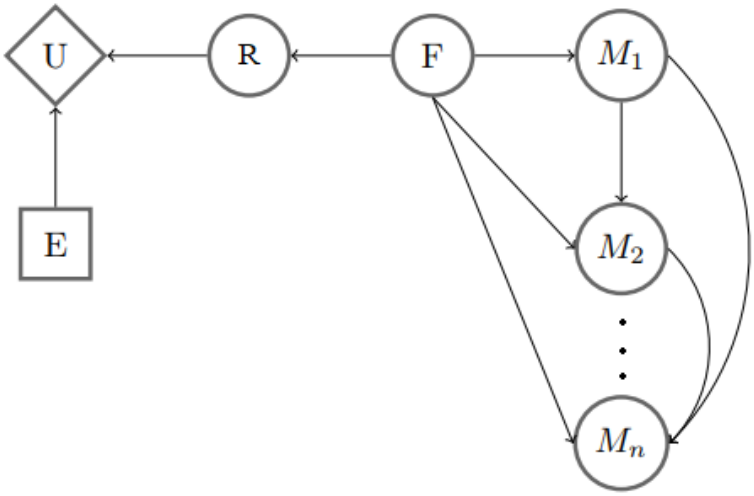
(e) [3 pts] Suppose you decide to use a particle filter instead of the forward algorithm. What is the run time of a particle filter with P particles? Please answer in big O notation, in terms of T, m, G, n, and P, or write "N/A" if the run time is unable to be determined with the given information.

TP

Assuming the time for resampling a single particle is constant, then the runtime is TP. At each timestep (total of T), we perform the time elapse and observation updates in constant time for each of the P particles.

Q8. [12 pts] Raisins Again (VPI)

(a) Valerie from midterm 1 is still concerned about her cookie containing raisins so she decides to take a completely new approach. She wants to find out which factory (F) made the cookie because that can help her find out if there are raisins (R) in the cookie. Thus she goes to the cookie company and asks a manager, M_1, which factory the cookie was made in. However, she doesn't fully trust his answer so she asks his superior, M_2, which factory he thinks it is in and how credible M_1 is. She knows there is a chain of n managers like this where every manager can tell her which factory they think the cookie was made in and the credibility of **every** manager working under them. This system can be modeled by the decision network below.



For this question assume $1 < i < k < n$, choose one for each equation:

	Could be true	**Must be true**	**Must be false**
$VPI(M_i) > VPI(M_k)$	●	○	○
$VPI(M_k) > VPI(M_i)$	●	○	○
$VPI(F) \geq VPI(M_i)$	○	●	○
$VPI(F) > VPI(R)$	○	○	●
$VPI(M_k \mid M_i) > VPI(M_k)$	●	○	○
$VPI(M_i \mid M_k) > VPI(M_i)$	●	○	○
$VPI(M_i, M_k) = VPI(M_i \mid M_k) + VPI(M_k)$	○	●	○
$VPI(M_i \mid F) > 0$	○	○	●

SID: _____

Q9. [19 pts] Machine Learning: Potpourri

(a) [2 pts] What it the **minimum** number of parameters needed to fully model a joint distribution $P(Y, F_1, F_2, ..., F_n)$ over label Y and n features F_i? Assume binary class where each feature can possibly take on k distinct values.

$2k^n - 1$

(b) [3 pts] Under the **Naive Bayes assumption**, what is the **minimum** number of parameters needed to model a joint distribution $P(Y, F_1, F_2, ..., F_n)$ over label Y and n features F_i? Assume binary class where each feature can take on k distinct values.

$2n(k-1)+1$

(c) [1 pt] You suspect that you are overfitting with your Naive Bayes with Laplace Smoothing. How would you adjust the strength k in Laplace Smoothing?

- ● Increase k
- ○ Decrease k

(d) [2 pts] While using Naive Bayes with Laplace Smoothing, increasing the strength k in Laplace Smoothing can:

- ■ Increase training error
- ☐ Decrease training error
- ■ Increase validation error
- ■ Decrease validation error

(e) [1 pt] It is possible for the perceptron algorithm to never terminate on a dataset that is linearly separable in its feature space.

- ○ True
- ● False

(f) [1 pt] If the perceptron algorithm terminates, then it is guaranteed to find a max-margin separating decision boundary.

- ○ True
- ● False

(g) [1 pt] In multiclass perceptron, every weight w_y can be written as a linear combination of the training data feature vectors.

- ● True
- ○ False

(h) [1 pt] For binary class classification, logistic regression produces a linear decision boundary.

- ● True
- ○ False

(i) [1 pt] In the binary classification case, logistic regression is exactly equivalent to a single-layer neural network with a sigmoid activation and the cross-entropy loss function.

- ● True
- ○ False

(j) **(i)** [2 pts] You train a linear classifier on 1,000 training points and discover that the training accuracy is only 50%. Which of the following, if done in isolation, has a good chance of improving your training accuracy?

■ Add novel features ☐ Train on more data ■ Train on less data

(ii) [2 pts] You now try training a neural network but you find that the training accuracy is still very low. Which of the following, if done in isolation, has a good chance of improving your training accuracy?

■ Add more hidden layers ■ Add more units to the hidden layers

(k) Recall the following kernels covered in class:

1. Linear kernel: $K(x, x') = x \cdot x' = \sum_i x_i x'_i$
2. Quadratic kernel: $K(x, x') = (x \cdot x' + 1)^2 = \sum_{i,j} x_i x_j x'_i x'_j + 2\sum_i x_i x'_i + 1$
3. Gaussian RBF kernel: $K(x, x') = \exp\left(-\frac{1}{2\sigma^2}\|x - x'\|^2\right)$

(i) [1 pt] There exists data that is separable with a Gaussian RBF kernel but not with a quadratic kernel.

● True ○ False

(ii) [1 pt] There exists data that is separable with a linear kernel but not with a quadratic kernel.

○ True ● False

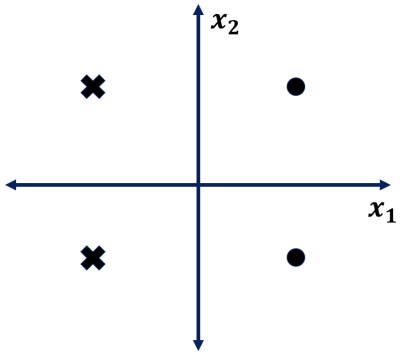
SID: _____

Q10. [24 pts] Perceptrons and Naive Bayes

(a) For each of the datasets represented by the graphs below, please select the feature maps for which the perceptron algorithm can perfectly classify the data.

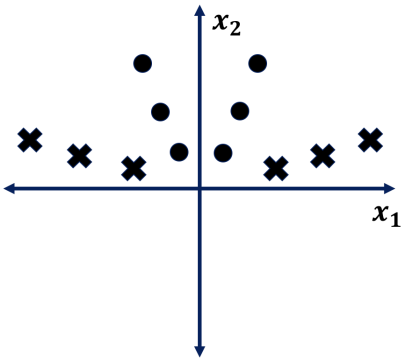
Each data point is in the form (x_1, x_2), and has some label Y, which is either a 1 (dot) or -1 (cross).

(i) [5 pts]



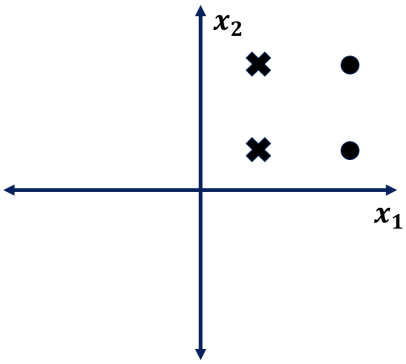
- [■] $[x_1 \quad x_2 \quad 1]$
- [■] $[x_1 \quad x_2 \quad x_1^2]$
- [■] $[x_1 \quad x_2 \quad |x_1|]$
- [■] $[x_1 \quad x_2 \quad Y]$
- [■] $[x_1 \quad x_2]$

(ii) [5 pts]



- [☐] $[x_1 \quad x_2 \quad 1]$
- [■] $[x_1 \quad x_2 \quad x_1^2]$
- [■] $[x_1 \quad x_2 \quad |x_1|]$
- [■] $[x_1 \quad x_2 \quad Y]$
- [☐] $[x_1 \quad x_2]$

(iii) [5 pts]



- [■] $[x_1 \quad x_2 \quad 1]$
- [■] $[x_1 \quad x_2 \quad x_1^2]$
- [☐] $[x_1 \quad x_2 \quad |x_1|]$
- [■] $[x_1 \quad x_2 \quad Y]$
- [☐] $[x_1 \quad x_2]$

(b) [2 pts] Performing maximum likelihood estimation (MLE) to fit the parameters of a Bayes net to some given data (with no Laplace smoothing) leads to which of the following learning algorithms?

- [x] Naive Bayes
- [ ] Perceptrons
- [ ] Kernelization
- [ ] Neural Networks
- ○ None

(c) Suppose that we are trying to perform a binary classification task using Naive Bayes. Y is the label, and (X_1, X_2) are the features. The domain for the features is anywhere on the 3×3 grid centered at $(0, 0)$. In other words, X_1 and X_2 have the domain $\{-1, 0, 1\}$

Suppose that this is your dataset: (0, 1, +), (0, −1, −), (−1, 1, +), (-1, −1, −), (1, 0, +), (−1, 1, −), (0, 0, +). What is the learned value of each of the following? (Leave your answer as a simplified fraction)

(i) [1 pt]

$P(Y = +)$

$\frac{4}{7}$

(ii) [1 pt]

$P(X_1 = 1|Y = -)$

0

(iii) [1 pt]

$P(X_2 = 0|Y = +)$

$\frac{2}{4}$

(d) Now, to decouple from the previous question, assume that the learned CPTs are below.

Y	X_1	$\Pr(X_1 \mid Y)$
+	−1	0.4
+	0	0.1
+	1	0.5
−	−1	0.6
−	0	0.3
−	1	0.1

Y	X_2	$\Pr(X_2 \mid Y)$
+	−1	0.2
+	0	0.2
+	1	0.6
−	−1	0.7
−	0	0.1
−	1	0.2

Y	$\Pr(Y)$
+	0.2
−	0.8

(i) [2 pts] What would be the predicted value for Y if the data point is at $(0, 0)$?

$\hat{Y} = -$

$P(Y = -, x_1 = 0, x_2 = 0) = 0.8 * 0.3 * 0.1 = 0.024, P(Y = +, x_1 = 0, x_2 = 0) = 0.2 * 0.1 * 0.2 = 0.004$

(ii) [2 pts] What would be the predicted value for Y if the data point is at $(1, -1)$?

$\hat{Y} = -$

$P(Y = -, x_1 = 1, x_2 = -1) = 0.8 * 0.1 * 0.7 = 0.056, P(Y = +, x_1 = 1, x_2 = -1) = 0.2 * 0.5 * 0.2 = 0.02$

SID: _____

Q11. [24 pts] Neural Network

The network below is a neural network with inputs x_1 and x_2, and outputs y_1 and y_2. The internal nodes are computed below. All variables are scalar values. Note that $ReLU(x) = \max(0, x)$.

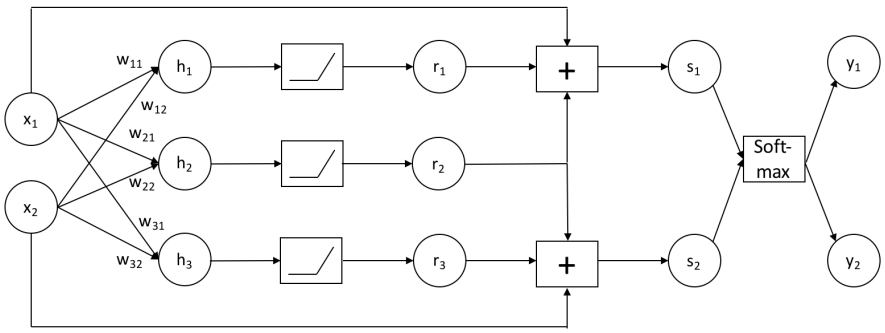


Figure 3: Neural Network

The expressions for the internal nodes in the network are given here for convenience:

$h_1 = w_{11}x_1 + w_{12}x_2 \qquad h_2 = w_{21}x_1 + w_{22}x_2 \qquad h_3 = w_{31}x_1 + w_{32}x_2$

$r_1 = ReLU(h_1) \qquad r_2 = ReLU(h_2) \qquad r_3 = ReLU(h_3) \qquad s_1 = x_1 + r_1 + r_2 \qquad s_2 = x_2 + r_2 + r_3$

$y_1 = \frac{exp(s_1)}{exp(s_1)+exp(s_2)} \qquad y_2 = \frac{exp(s_2)}{exp(s_1)+exp(s_2)}$

(a) Forward Propagation

Suppose for this part only, $x_1 = 3, x_2 = 5, w_{11} = -10, w_{12} = 7, w_{21} = 2, w_{22} = 5, w_{31} = 4, w_{32} = -4$. What are the values of the following internal nodes? Please simplify any fractions.

(i) [1 pt] $h_1 =$

5

$(-10) * 3 + 7 * 5 = 5$

(ii) [2 pts] $s_1 =$

39

$3 + ReLU((-10) * 3 + 7 * 5) + ReLU(2 * 3 + 5 * 5) = 3 + 5 + 31 = 39$

(iii) [3 pts] $y_2 =$

$\frac{1}{1+exp(3)}$

$s_2 = 5 + ReLU(2 * 3 + 5 * 5) = 5 + 31 = 36$

$y_2 = \frac{exp(36)}{exp(36)+exp(39)} = \frac{1}{1+exp(3)}$

(b) Back Propagation

Compute the following gradients analytically. The answer should be an expression of any of the nodes in the network $(x_1, x_2, h_1, h_2, h_3, r_1, r_2, r_3, s_1, s_2, y_1, y_2)$ or weights $w_{11}, w_{12}, w_{21}, w_{22}, w_{31}, w_{32}$ (clarification during exam: without derivative or partial derivative symbols). In the case where the gradient depend on the value of nodes in the network, **please list all possible analytical expressions, caused by active/inactive ReLU, separated by comma.**

Hint 1: If z is a function of y, and y is a function of x, the chain rule of taking derivative is: $\frac{\partial z}{\partial x} = \frac{\partial z}{\partial y} * \frac{\partial y}{\partial x}$

Hint 2: Hint: Recall that for functions of the form $g(x) = \frac{1}{1+exp(a-x)}$, $\frac{\partial g}{\partial x} = g(x)(1 - g(x))$

(i) [1 pt] $\frac{\partial h_2}{\partial x_1} =$

w_{21}

(ii) [2 pts] $\frac{\partial h_1}{\partial w_{21}} =$

0

h_1 is independent of weight w_{21}

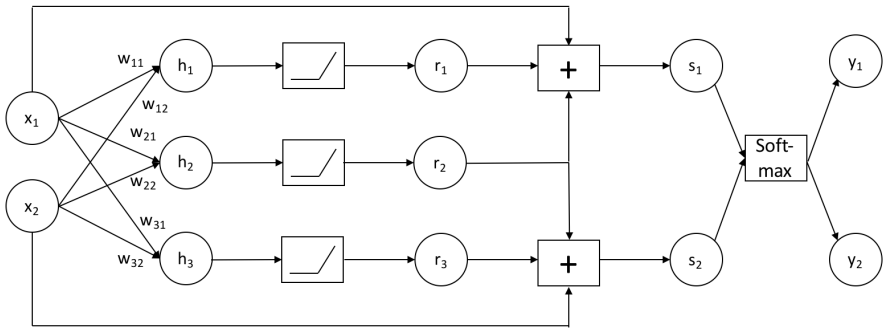


Figure 4: Neural Network, copied from the previous page for reference

(iii) [2 pts] $\frac{\partial r_3}{\partial w_{31}} =$

$0, x_1$

If the ReLU is active, the result is straightforward, and thus x_1. If the ReLU is inactive, the gradient is 0

(iv) [2 pts] $\frac{\partial s_1}{\partial r_1} =$

1

(v) [3 pts] $\frac{\partial s_1}{\partial x_1} =$

$1 + w_{11} + w_{21}, 1 + w_{21}, 1 + w_{11}, 1$

The ReLU with output r_1 and the ReLU with output r_2 can be active or inactive independently

(vi) [3 pts] $\frac{\partial y_2}{\partial s_2} =$

$y_1 y_2$, or $y_2(1 - y_2)$

$y_2 = \frac{exp(s_2)}{exp(s_1)+exp(s_2)} = \frac{1}{1+exp(s_1-s_2)}$

$\frac{\partial y_2}{\partial s_2} = y_2(1 - y_2) = y_1 y_2 = \frac{exp(s_1-s_2)}{(1+exp(s_1-s_2))^2} = \frac{exp(s_1+s_2)}{(exp(s_1)+exp(s_2))^2}$

Please note that, due to symmetry between y_1 and y_2, $\frac{exp(s_2-s_1)}{(1+exp(s_2-s_1))^2}$ is also equivalent to the correct answer. If you don't believe it, try simplifying it!

(vii) [3 pts] $\frac{\partial y_1}{\partial x_1} =$

See below

First of all, I want to apologize for making this mistake. This question turns out to be much harder than expected. But if you managed to get this one right, you deserve the title of a True Gradient Champion!

The correct answer is (4 total terms):
$y_1 y_2(1 + w_{11} - w_{31}), y_1 y_2(1 + w_{11}), y_1 y_2(1 - w_{31}), y_1 y_2$

This is how we get to it:
$y_1 = \frac{exp(s_1)}{exp(s_1)+exp(s_2)} = \frac{1}{1+exp(s_2-s_1)}$

When calculating $\frac{\partial y_1}{\partial s_1}$, this is the correct format for $g(x) = \frac{1}{1+exp(a-x)}$, because there is a negative sign is in front of s_1. e can apply the rule to simply get $\frac{\partial y_1}{\partial s_1} = y_1(1-y_1) = y_1 y_2$

But when calculating $\frac{\partial y_1}{\partial s_2}$, this is NOT in the correct format, because now there is no negative sign in front of s_2. One way to resolve this is to create a fake variable $t_2 = -s_2$, and $\frac{\partial t_2}{\partial s_2} = -1$.

Now $y_1 = \frac{1}{1+exp(s_2-s_1)} = \frac{1}{1+exp(-s_1+s_2)} = \frac{1}{1+exp(-s_1-t_2)}$, which is the correct format again. At this point, $\frac{\partial y_1}{\partial s_2} = \frac{\partial y_1}{\partial t_2}\frac{\partial t_2}{\partial s_2} = [y_1(1-y_1)](-1) = -y_1 y_2$

As seen in v, $\frac{\partial s_1}{\partial x_1} = 1 + w_{11} + w_{21}, 1 + w_{21}, 1 + w_{11}, 1$

Similarly, $\frac{\partial s_2}{\partial x_1} = w_{21} + w_{31}, w_{21}, w_{31}$

In the end,

$$\frac{\partial y_1}{\partial x_1} = \frac{\partial y_1}{\partial s_1}\frac{\partial s_1}{\partial x_1} + \frac{\partial y_1}{\partial s_2}\frac{\partial s_2}{\partial x_1} = y_1 y_2 * \{1 + w_{11} + w_{21}, 1 + w_{21}, 1 + w_{11}, 1\} + (-y_1 y_2) * \{w_{21} + w_{31}, w_{21}, w_{31}\}$$

Which can be expanded to this list (note that things can cancel out): $y_1 y_2(1 + w_{11} - w_{31}), y_1 y_2(1 + w_{11}), y_1 y_2(1 + w_{11} + w_{21} - w_{31}), y_1 y_2(1 - w_{31}), y_1 y_2, y_1 y_2(1 + w_{21} - w_{31}), y_1 y_2(1 + w_{11} - w_{21} - w_{31}, y_1 y_2(1 + w_{11} - w_{21}), y_1 y_2(1 + w_{11} - w_{31}), y_1 y_2(1 + -w_{21} - w_{31}, y_1 y_2(1 - w_{21}), y_1 y_2(1 - w_{31})$

BUT!!!! Please note that the activity of r_2 actually affect both $\frac{\partial s_1}{\partial x_1}$ and $\frac{\partial s_2}{\partial x_1}$! So we need to remove 6 contradicting terms, including $y_1 y_2(1 + w_{11} + w_{21} - w_{31})$, $y_1 y_2(1 + w_{21} - w_{31})$, $y_1 y_2(1 + w_{11} - w_{21} - w_{31})$, $y_1 y_2(1 + w_{11} - w_{21})$, $y_1 y_2(1 - w21 - w_{31})$, $y_1 y_2(1 - w_{21})$.

The easier way to think about this is, that if r_2 is activated, then the gradient will cancel out from the two sides. If r_2 is deactivated, it won't appear at all.

We are left with 6 terms: $y_1 y_2(1+w_{11}-w_{31}), y_1 y_2(1+w_{11}), y_1 y_2(1-w_{31}), y_1 y_2, y_1 y_2(1+w_{11}-w_{31}), y_1 y_2(1-w_{31})$

Removing the duplicates, we are left with the final 4 terms: $y_1 y_2(1 + w_{11} - w_{31}), y_1 y_2(1 + w_{11}), y_1 y_2(1 - w_{31}), y_1 y_2$

(c) [2 pts] In roughly 15 words, what role could the non-negative values in node r_2 play according to its location in the network architecture?

Smoothing the inputs into the soft-max functions. (anything reasonable is acceptable)