# CSCE 580: Artificial Intelligence
# Written Homework 5: Supervised and Reinforcement Learning
Due: 4/8/2024 at 11:58pm

You may either type your homework or upload a scanned copy of your written homework. However, your answers must be legible. **Answers that are not legible will be marked as incorrect.**

Turn in **PDF** (not Word or any other format) of your answers to Blackboard.

## 1 Neural Networks

### 1.1 Gradients (10 pts)

We have a function approximator $f(x, w_1, b, w_2) = (xw_1 + x^2w_1w_2 + b)^2w_1$.
Calculate $\frac{\partial f}{\partial w_2}$, $\frac{\partial f}{\partial w_1}$, and $\frac{\partial f}{\partial b}$. Show your work.

$\partial f/\partial w_2 = 2w_1 * (x^2w_1) * (xw_1 + x^2w_1w_2 + b)^{2w\_1 - 1}$

$= (x + 2x^2w_2 + b) * (2w_1) * (xw_1 + x^2w_1w_2 + b)^{2w\_1 - 1}$

$= 2w_1 * (xw_1 + x^2w_1w_2 + b)^{2w\_1 - 1}$

### 1.2 Activation Functions (10 pts)

Suppose we have a deep neural network of the same architecture described in question 1.4, what is the hypothesis space of the deep neural network if a linear activation function is used in all layers? Explain your answer.

The hypothesis space of a deep neural network with linear activation functions in all layers is equivalent to a single-layer linear model. This is because a linear activation function does not introduce non-linearity into the network.

### 1.3 Data Imbalance (5 pts)

Suppose you are doing supervised learning with a neural network on a binary classification task. What are the effects that an unbalanced dataset can have on the final learned model?

The model may become biased towards the majority class.

The model's ability to generate to unseen data may be compromised, especially for the minority class.

The model's sensitivity to the minority class may decrease, leading to a higher false negative rate.

Evaluation metrics like accuracy may not accurately reflect the model's performance, especially in cases where the majority class dominates.

## 1.4 Parameters (5 pts)

Suppose there is a fully-connected deep neural network with an input layer of size 324, followed by one hidden layer of size 3000, followed by 7 hidden layers of size 1000 and an output layer of size 1. How many parameters does this deep neural network have? Do not forget to include the bias terms. Show your work.

First hidden layer
Weights: 324 * 3000 = 972,000
Biases: 3000

Hidden layers 1-7
Weights: 3000 * 1000 * 7 = 21,000,000
Biases: 1000 * 7 = 7,000

Last hidden layer
Weights: 1000 * 1 = 1000
Biases: 1

Total # of parameters:
972,000 + 3,000 + 21,000,000 + 7,000 + 1,000 + 1 = 21,980,001

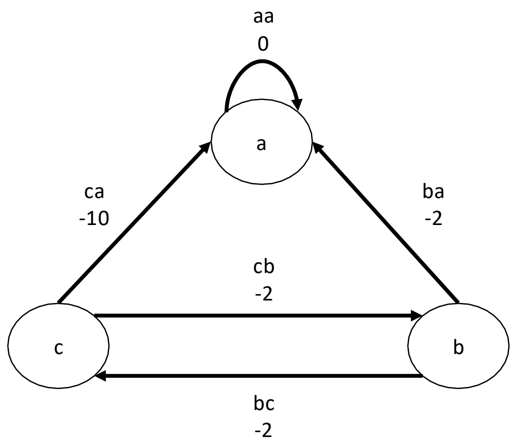
## 1.5 Matching (10 pts)

For each of the following items on the left, write in the letter corresponding to the best answer or the correct definition on the right. The first one is done for you as an example.

| | | | |
|---|---|---|---|
| B | ADAM | A | When a model is overly complex, causing it to make predictions using irrelevant patterns in the data |
| I | Overfitting | B | An optimization method with an adaptive learning rate for each parameter |
| E | Training set | C | Supervised learning with continuous labels |
| F | Validation set | D | Randomly sets the outputs of neurons to zero during training |
| C | Regression | E | Examples of input/output pairs that are used to train a model |
| D | Dropout | F | Examples of input/output pairs that are not seen during training and used to estimate how well the model generalizes |
| J | Classifcation | G | Can be used to allow deep neural networks to represent non-linear functions |
| K | SGD with Momentum | H | An algorithm used to compute the gradient |
| H | Backpropagation | I | The process of reducing overfitting |
| G | Activation Function | J | Supervised learning with categorical labels |
| A | Regularization | K | An optimization methods that uses a weighted average of past gradients to update parameters |

## 2 Dynamic Programming

A diagram of an MDP is shown in the figure below. There are three states: a, b, and c. The actions available in each state are shown on the edges of the arrows and the rewards are shown below the actions. The agent can take action aa in state a, actions ba or bc in state b, and actions ca or cb in state c. All transitions are deterministic. The discount factor $\gamma = 1$. The first iteration of value iteration has been performed and the values assigned to each state are currently $V(a) = 0$, $V(b) = -2$, and $V(c) = -6$.



### 2.1 (10 pts)

Given the current value function, perform one additional iteration of value iteration and write the new value of each state. Show your work.

Use Bellman optimality equation.

V(a) = 0
V(b) = -2
V(c) = -6

V(a) = max((0 + 1 * V(a), 0) = max(0,0) = 0
V(b) = max(-2,-8) = -2
V(c) = max(-10,-4) = -4

V(a) = 0
V(b) = -2
V(c) = -4

### 2.2 (10 pts)

What policy would be obtained by behaving greedily with respect to the value function in 2.1.

$\pi(aa|a)$ = 1____ $\pi(ba|b)$ = 1____ $\pi(bc|b)$ = 0____ $\pi(ca|c)$ = 0____ $\pi(cb|c)$ = 1____

## 2.3 (10 pts)

Is the value function in 2.1 optimal? Justify your answer based on theoretical properties of value iteration.

The value function obtained in 2.1 is not necessarily optimal. In value iteration, convergence to the optimal value function is guaranteed only if the discount factor is strictly less than 1. Depending on how the value stabilizes, the function is either optimal or not. If the value stabilizes, then the obtained value function is optimal. Otherwise, it's not.

## 3 Reinforcement Learning

### 3.1 Policy Iteration (10 pts)

For policy evaluation in the coding project, we started with a uniform random policy, a discount of $\gamma = 1$, and stopped policy evaluation only when $\Delta = 0$. If we started, instead, with a policy that gave the same action in every state (i.e. in every state, go left) we would never converge to $\Delta = 0$. Why is this? If we reduce $\gamma$, we would converge to $\Delta = 0$. Why is this?

In the starting policy, it would never converge because the agent is following a deterministic policy. Therefore, the agent doesn't explore different actions.

If we reduce the discount factor, we would converge to that since a smaller discount factor places less emphasis on future rewards. Reducing the discount factor can lead to faster convergance during policy evaluation.

### 3.2 Model-Free RL (10 pts)

In model-free control algorithms, such as Q-learning and SARSA, why do we use an action-value function, $Q(s, a)$, instead of a state-value function, $V(s)$?

In model-free control algorithms like Q-learning and SARSA, we use an action-value function instead of a state-value function because we need to estimate the value of taking specific actions in specific states rather than just the value of being in a particular state.

### 3.3 Matching (10 pts)

| | | | |
|---|---|---|---|
| C | State value function | A | On-policy model-free RL algorithm |
| F | Action value function | B | Off-policy model-free RL algorithm |
| H | Markov property | C | Expected future reward when in a given state |
| G | Discount factor | D | Used to find $v_\pi$ for a given policy $\pi$ |
| D | Policy Evaluation | E | Used to find $v_*$ using the Bellman optimality equation as an update rule |
| K | Value Iteration | F | Expected future reward when in a given state and taking a given action |
| I | Model-Free RL | G | Controls how far the agent looks into the future |
| J | Policy improvement | H | The joint distribution over the next states and rewards is conditionally independent of the history given the current state and action. |
| E | Policy iteration | I | Learns without knowing the dynamics of the MDP |
| A | SARSA | J | Uses $v_\pi$ to obtain new policy $\pi^J$ such that $\pi^J \geq \pi$. |
| B | Q-learning | K | Used to find $v_*$ by iterating between policy evaluation and policy improvement. |

## Extra Credit: Bellman Equation (25 pts)

In the lecture slides, we derived the Bellman equation from the definition of $v_\pi$. One crucial step was Equation 1. Show the derivation for Equation 1. Hint: you may need to use the definition of conditional expectations, marginalization (sum rule), the product rule, and the Markov property. Assume that all possible assignments to random variables are either finite or countably infinite, that is, sums are needed when taking expectations, but not integrals. Show your work.
**All of your work must be your own.**

$$\mathop{\mathbb{E}}_{a\sim\pi(a|s)}[G_{t+1}|S_t = s] = \sum_a \pi(a|s) \sum_{s'} p(s'|s,a)v_\pi(s') \tag{1}$$

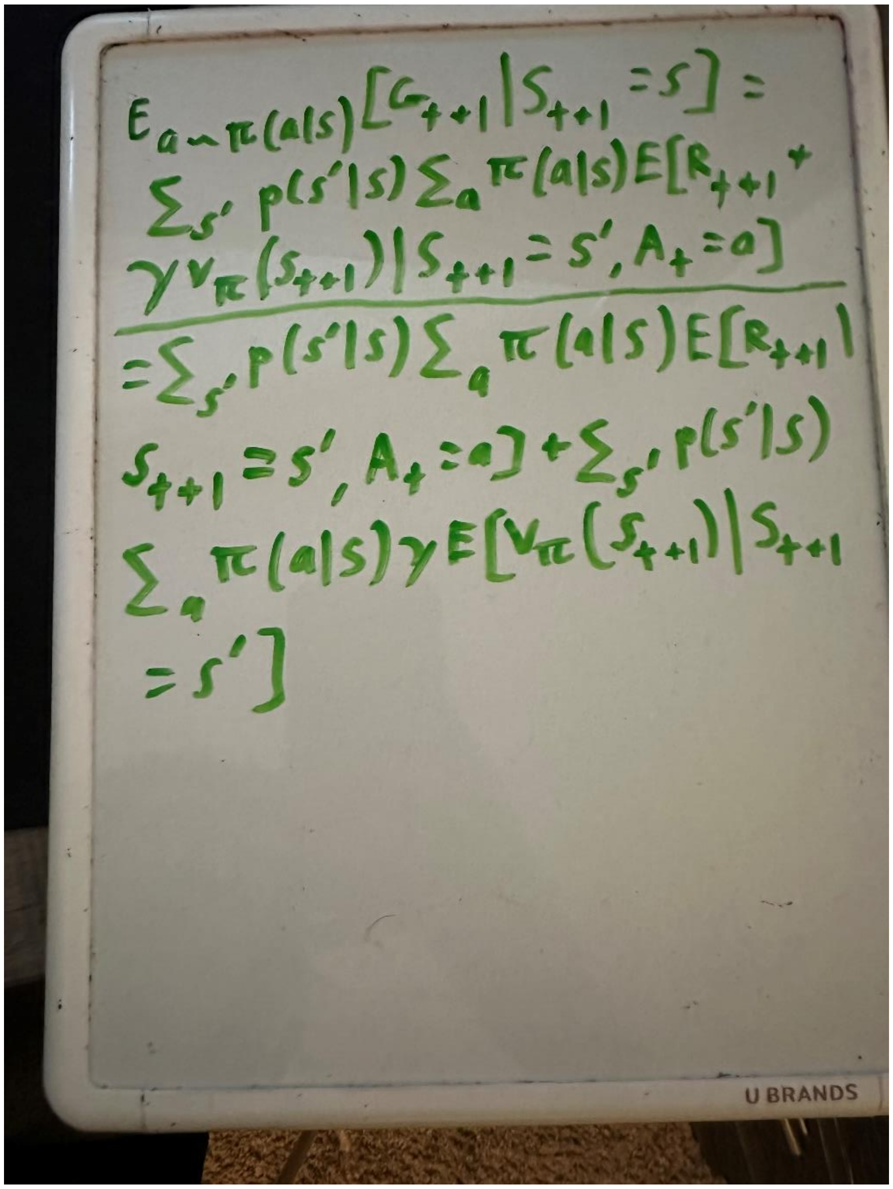
Background knowledge:

$\mathbb{E}_{a\sim\pi(a|s)}[G_{t+1}|S_{t+1} = s] = \sum_{g'} p(g'|s)g'$

$\mathbb{E}_{a\sim\pi(a|s)}[G_{t+1}|S_{t+1} = s'] = \sum_{g'} p(g'|s')g' = v_\pi(s')$

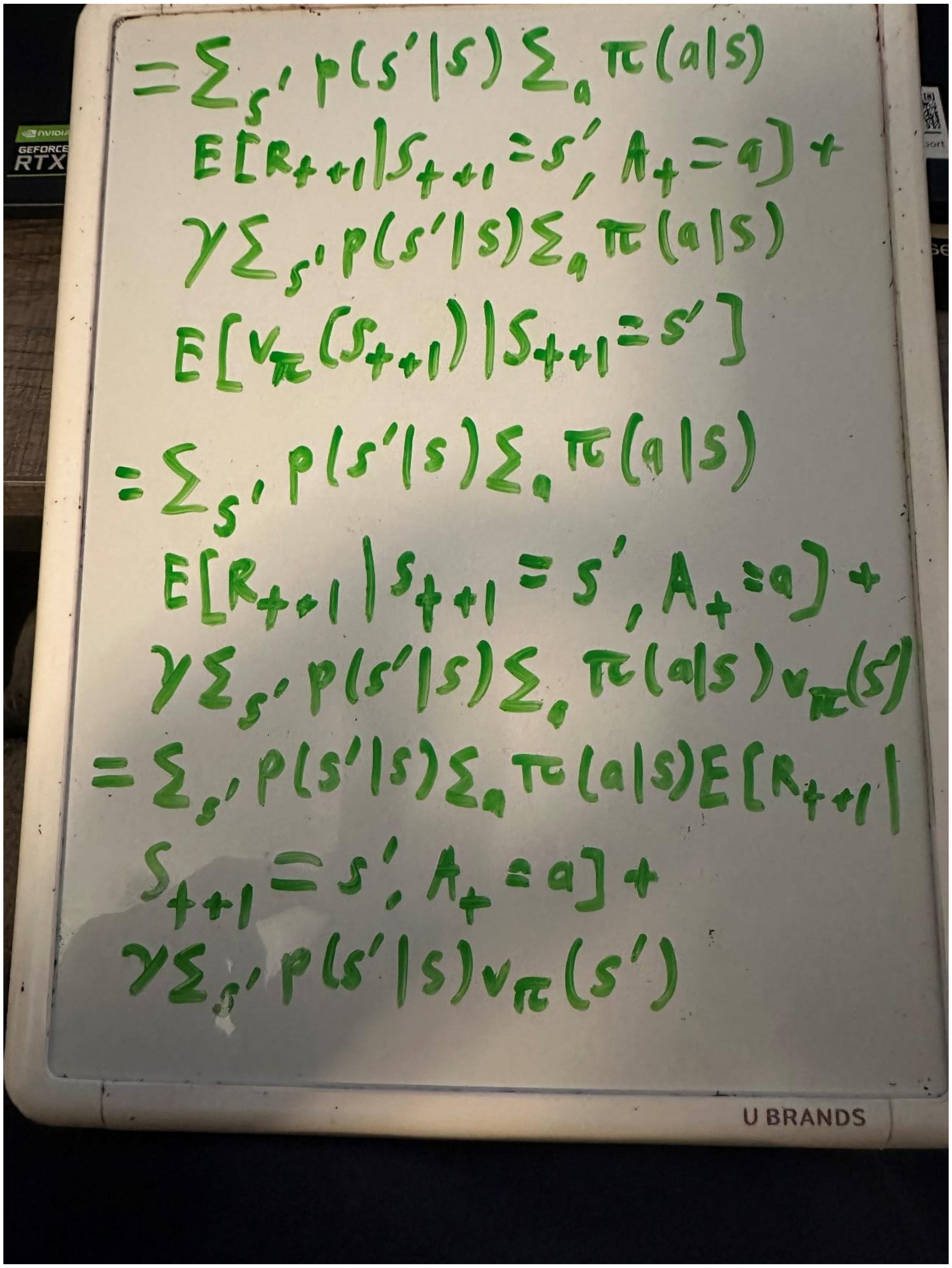
$p(g'|s', s) = p(g'|s')$ (Markov property)

Expand $G_{t+1}$:



$$E_{a\sim\pi(a|s)}[G_{t+1}|S_{t+1}=s] = \sum_{s'} p(s'|s)\sum_a \pi(a|s)E[R_{t+1} + \gamma v_\pi(S_{t+1})|S_{t+1}=s', A_t=a]$$

$$= \sum_{s'} p(s'|s)\sum_a \pi(a|s)E[R_{t+1}|S_{t+1}=s', A_t=a] + \sum_{s'} p(s'|s)\sum_a \pi(a|s)\gamma E[v_\pi(S_{t+1})|S_{t+1}=s']$$

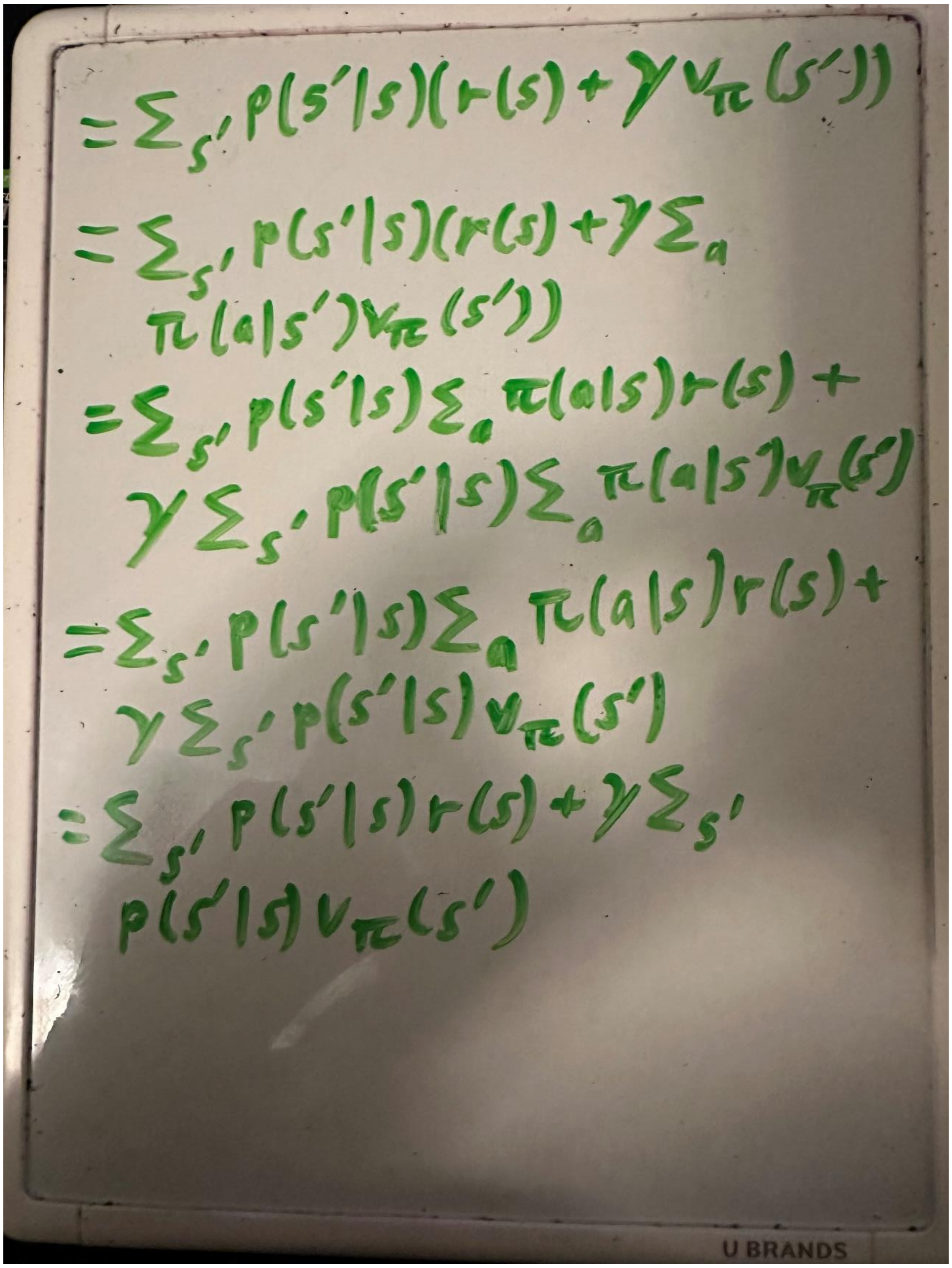


$$= \sum_{s'} p(s'|s) \sum_a \pi(a|s)$$
$$E[R_{t+1}|S_{t+1}=s', A_t=a] +$$
$$\gamma \sum_{s'} p(s'|s) \sum_a \pi(a|s)$$
$$E[v_\pi(S_{t+1})|S_{t+1}=s']$$

$$= \sum_{s'} p(s'|s) \sum_a \pi(a|s)$$
$$E[R_{t+1}|S_{t+1}=s', A_t=a] +$$
$$\gamma \sum_{s'} p(s'|s) \sum_a \pi(a|s) v_\pi(s')$$

$$= \sum_{s'} p(s'|s) \sum_a \pi(a|s) E[R_{t+1}|$$
$$S_{t+1}=s', A_t=a] +$$
$$\gamma \sum_{s'} p(s'|s) v_\pi(s')$$

Now use the definition of V$_{pi}$(s’) to simplify the expression:



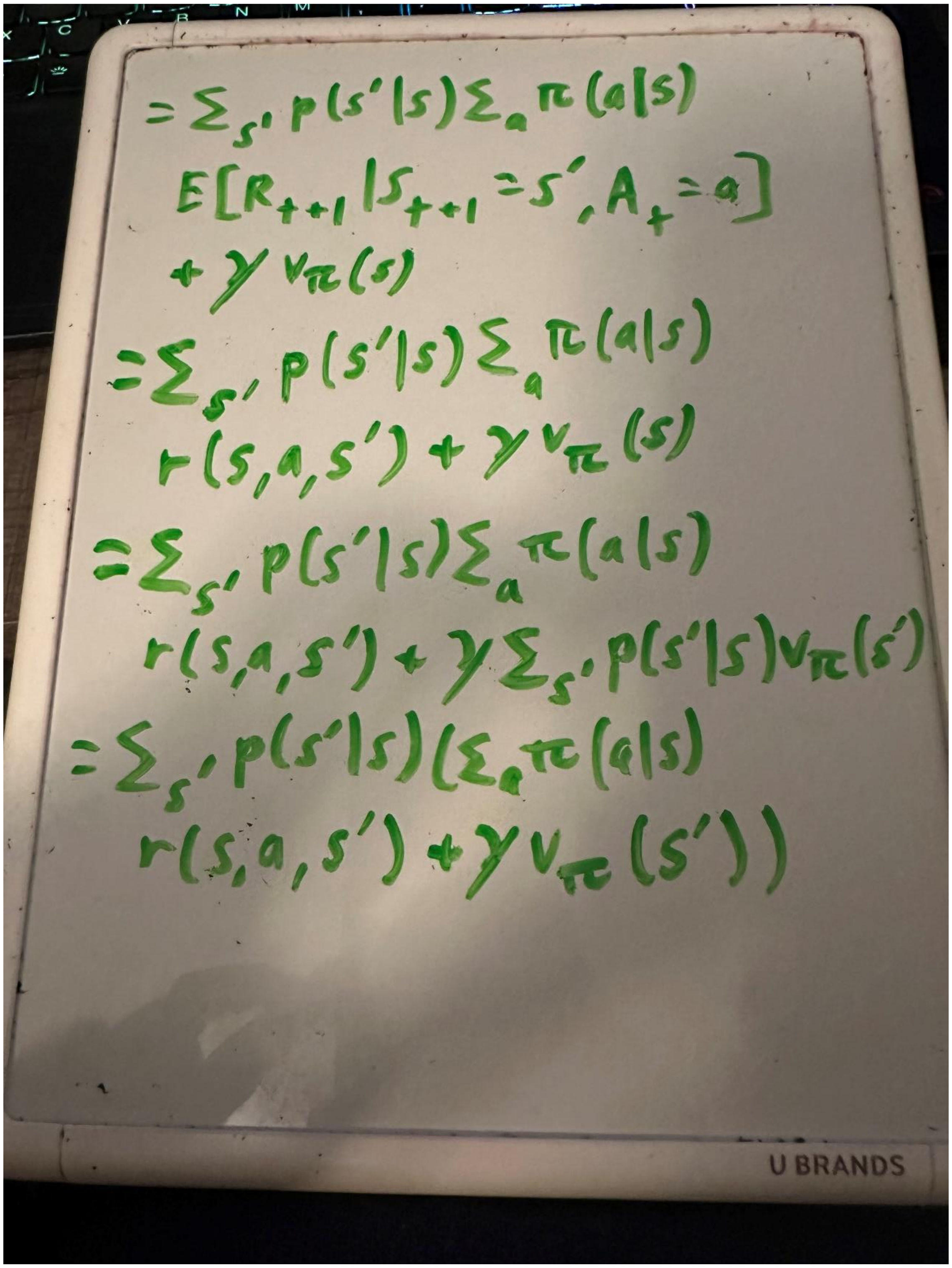
$$= \Sigma_{s'} p(s'|s)(r(s) + \gamma v_{\pi}(s'))$$

$$= \Sigma_{s'} p(s'|s)(r(s) + \gamma \Sigma_{a} \pi(a|s') v_{\pi}(s'))$$

$$= \Sigma_{s'} p(s'|s) \Sigma_{a} \pi(a|s) r(s) + \gamma \Sigma_{s'} p(s'|s) \Sigma_{a} \pi(a|s) v_{\pi}(s')$$

$$= \Sigma_{s'} p(s'|s) \Sigma_{a} \pi(a|s) r(s) + \gamma \Sigma_{s'} p(s'|s) v_{\pi}(s')$$

$$= \Sigma_{s'} p(s'|s) r(s) + \gamma \Sigma_{s'} p(s'|s) v_{\pi}(s')$$



$= \sum_{s'} p(s'|s) \sum_a \pi(a|s)$

$E[R_{t+1} | S_{t+1} = s', A_t = a]$

$+ \gamma v_\pi(s)$

$= \sum_{s'} p(s'|s) \sum_a \pi(a|s)$

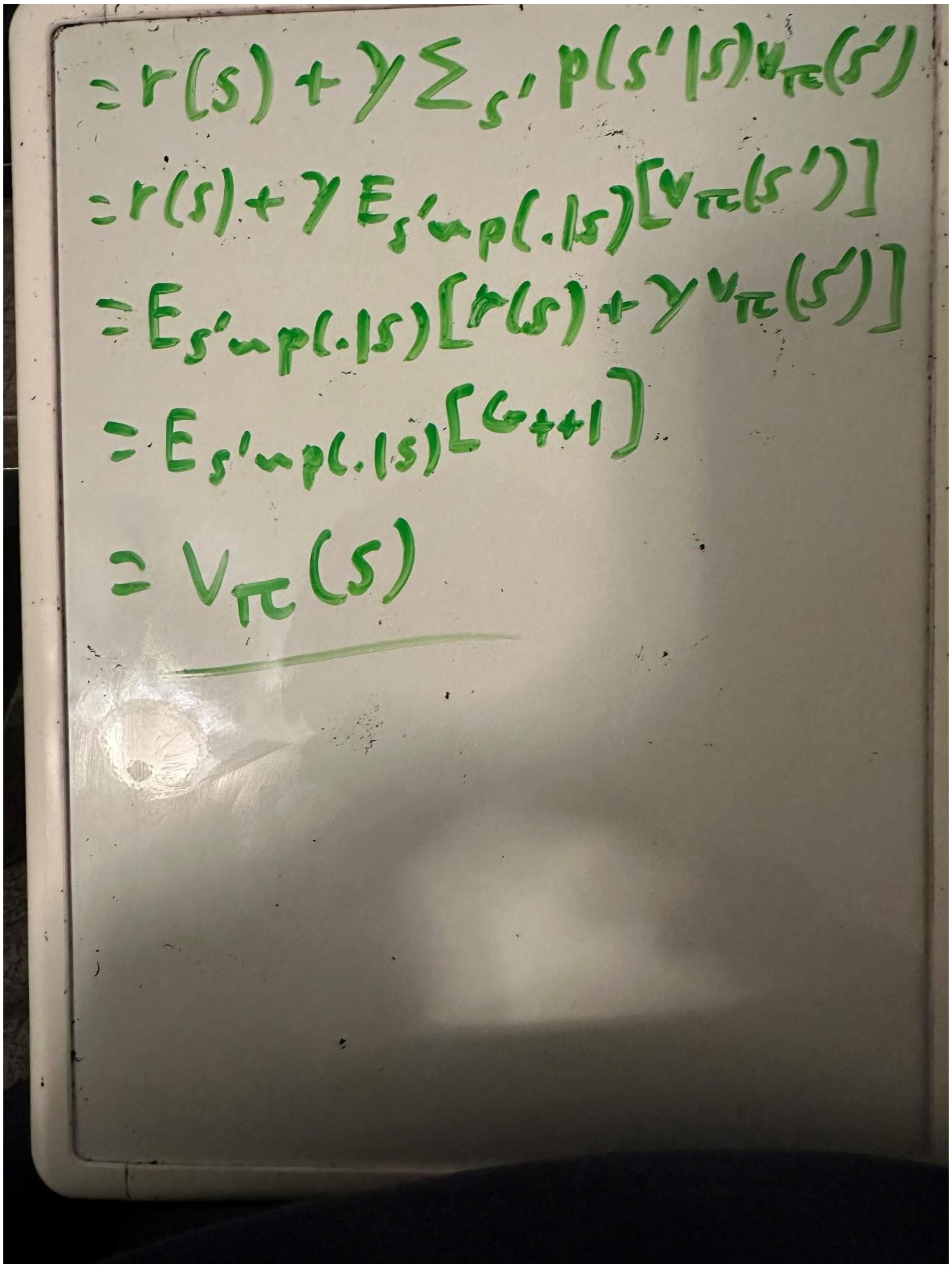
$r(s,a,s') + \gamma v_\pi(s)$

$= \sum_{s'} p(s'|s) \sum_a \pi(a|s)$

$r(s,a,s') + \gamma \sum_{s'} p(s'|s) v_\pi(s')$

$= \sum_{s'} p(s'|s) (\sum_a \pi(a|s)$

$r(s,a,s') + \gamma v_\pi(s'))$



$$= r(s) + \gamma \sum_{s'} p(s'|s) v_{\pi}(s')$$

$$= r(s) + \gamma E_{s' \sim p(.|s)}[v_{\pi}(s')]$$

$$= E_{s' \sim p(.|s)}[r(s) + \gamma v_{\pi}(s')]$$

$$= E_{s' \sim p(.|s)}[G_{t+1}]$$

$$= V_{\pi}(s)$$