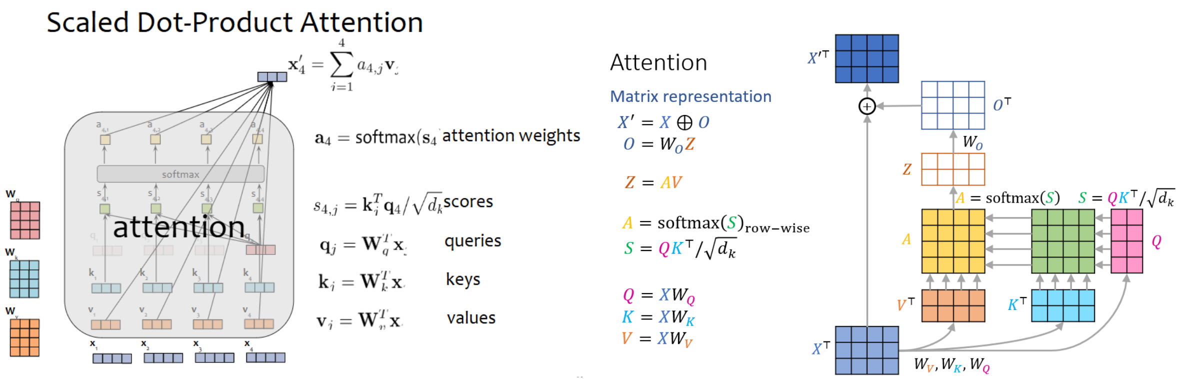
1 Definitions

1.1 Attention

Attention allows neural networks to focus on specific words of an input sequence when generating the output sequence. This means that the importance of different parts of the input are weighed dynamically.

1. Scaled Dot Product Attention: Computes attention scores using the formula Attention$(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$, where Q (query), K (key), and V (value) are matrices derived from the input, and d_k is the dimension of the keys.



2 MinGPT

MinGPT is a simple implementation of the GPT architecture. In following notebooks, pico version and femto version, we can explore how this model works. For some context, GPT utilizes transformers. A transformer is a type of neural network architecture that focuses on processing sequential data by emphasizing relationships between different parts of the sequence. It achieves this by employing a mechanism called "self-attention," which, in the context of large language models, allows the model to weigh the importance of each word in the prompt (context) when computing representations.

1. Look at the model architect of either pico or femto version of minGPT. Do you see anything familiar?

2. Compare minGPT pico and minGPT femto architectures. How many attention heads are their in each model? What are the dimension of the word embeddings?

   Note: Having multiple attention heads occurs when the attention mechanism is replicated several times in parallel, each time with different learned parameters. Think of it like having multiple channels in convolutions.

3. Now it's time to play around with the model. First, run the untrained pico model on the given context, "I do not like." Generate a sample from the model. What does the model predict?

4. Now, train the model for 100 iterations. Do the word embeddings look the same for the vocab? Does the word embedding look the same for the context?

5. Now, let's use the pretrained model on 10,000 iterations and run it with context = "not with in a house , not with a." Sample from this model. What does the model predict? What do the word embeddings for each attention head look like?

6. For the pretrained model run from the previous part, which attention head appears to be picking up on rhyming?

7. Now, compare the results from minGPT pico and minGPT femto. Which model is better?

Finally, in the notebooks, explore using either model how it generates new text!

3 Pay Attention to Dimensions!

1. Let's say our input sequence is $X \in \mathbb{R}^{T \times d_{model}}$ and our W_K, W_Q, W_V matrices are all in $\mathbb{R}^{d_{model} \times d_k}$.

   As a refresher, remember that $V = XW_V$, $Q = XW_Q$, and $K = XW_K$.

   (a) What would be the dimensions of $S = QK^T$?

   (b) How about $A = g_{sm}(S)$? g_{sm} applies the softmax function.

   (c) In the matrix version of attention, this softmax function is applied row or column wise?

2. Let's say that now $d_{model} = 2$, $T = 2$, and $d_k = 2$.

   (a) What would be the dimensions of $S = QK^T$?

   (b) How about $A = g_{sm}(S)$? g_{sm} applies the softmax function.

   (c) How about the W_K, W_Q, W_V matrices?

   (d) How about W, V, Q?

3. If $T = 3$, which of the following would change dimensions: $W_K, W_V, W_Q, K, V, Q, S, A, X$?

4 MDPs: Warm-Up

1. What is the Markov Property?

2. What are the Bellman equations, and when are they used?

3. What is a policy? What is an optimal policy?

4. How does the discount factor γ affect how the agent finds the optimal policy? Why do we restrict gamma $\gamma < 1$?

5. Fill in the following table explaining the effects of having different gamma values:

γ	**Effect on policy search:**
$\gamma = 0$	
$\gamma = 1$	

6. What are the two steps to Policy Iteration?

7. What is the relationship between $V^*(s)$ and $Q^*(s,a)$?

8. (MDP Notation Review) Draw a line connecting each term in the left column with its corresponding equation in the right column.

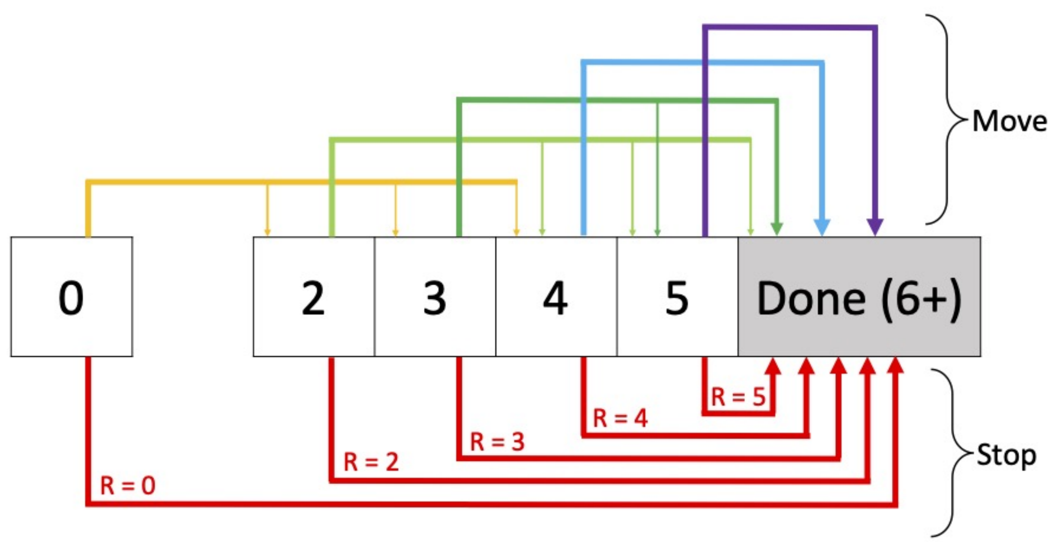
Term	*Equation*
Standard Expectimax	$\pi_V(s) = \text{argmax}_a \sum_{s'} P(s'\|s,a)[R(s,a,s')] + \gamma V(s')], \forall s$
Bellman Equation	$V_{k+1}(s) = \max_a \sum_{s'} P(s'\|s,a)[R(s,a,s')] + \gamma V_k(s')], \forall s$
Value Iteration	$V^*(s) = \max_a \sum_{s'} P(s'\|s,a)[R(s,a,s')] + \gamma V^*(s')]$
Q-Iteration	$Q_{k+1}(s,a) = \sum_{s'} P(s'\|s,a)[R(s,a,s')] + \gamma \max_{a'} Q_k(s',a')], \forall s,a$
Policy Extraction	$V_{k+1}^{\pi}(s) = \sum_{s'} P(s'\|s,\pi(s))[R(s,\pi(s),s')] + \gamma V_k^{\pi}(s')], \forall s$
Policy Evaluation	$\pi_{new}(s) = \text{argmax}_a \sum_{s'} P(s'\|s,a)[R(s,a,s')] + \gamma V^{\pi_{old}}(s')], \forall s$
Policy Improvement	$V(s) = \max_a \sum_{s'} P(s'\|s,a)V(s')$

5 MDPs: Racing

Consider a modification of the racing robot car example seen in lecture. In this game, the car repeatedly moves a random number of spaces that is equally likely to be 2, 3, or 4. The car can either Move or Stop if the total number of spaces moved is less than 6.

If the total spaces moved is 6 or higher, the game automatically ends, and the car receives a reward of 0. When the car Stops, the reward is equal to the total spaces moved (up to 5), and the game ends. There is no reward for the Move action.

Let's formulate this problem as an MDP with the states $\{0, 2, 3, 4, 5, Done\}$.



1. What is the transition function for this MDP? (You should specify discrete values for specific state/action inputs.)
2. What is the reward function for this MDP?

3. Recall the value iteration update equation:

$$V_{k+1}(s) = \max_a \sum_{s'} T(s,a,s')[R(s,a,s') + \gamma V_k(s')]$$

Perform value iteration for 4 iterations with $\gamma = 1$.

States	**0**	**2**	**3**	**4**	**5**	**Done**
V_0						0
V_1						0
V_2						0
V_3						0
V_4						0

4. You should have noticed that value iteration converged above. What is the optimal policy?

States	**0**	**2**	**3**	**4**	**5**
π^*					

5. How would our results change with $\gamma = 0.1$?

6. Now imagine we changed the rules so that moving to spaces after 5 does not immediately end the game, but rather wraps back around to the beginning states (this is the case where the states exist side by side in a loop).

   For example if you start in state 5, select the action Move, and move 2 spaces, you would end up in state 1 (this is a new state we would need introduce).

   To be clear, the states would now be states {0,1,2,3,4,5, *Done*}.

   How would this modification change our results? (Assume we again use γ=1)

6 MDPs: Policy Iteration

Recall the racing MDP from the prior problem. In this game, the car repeatedly moves a random number of spaces that is equally likely to be 2, 3, or 4. The car can either Move or Stop if the total number of spaces moved is less than 6.
If the total spaces moved is 6 or higher, the game automatically ends, and the car receives a reward of 0. When the car Stops, the reward is equal to the total spaces moved (up to 5), and the game ends. There is no reward for the Move action.
States: $\{0, 2, 3, 4, 5, Done\}$

Transition	Reward
$T(s, Stop, Done) = 1, \quad \forall s \neq Done$	$R(s, Stop, Done) = s, \quad \forall s \leq 5$
$T(0, Move, s') = \frac{1}{3}, \quad \forall s' \in \{2, 3, 4\}$	$R(s, a, s') = 0$ otherwise
$T(2, Move, s') = \frac{1}{3}, \quad \forall s' \in \{4, 5, Done\}$	
$T(3, Move, 5) = \frac{1}{3}$	
$T(3, Move, Done) = \frac{2}{3}$	
$T(4, Move, Done) = 1$	
$T(5, Move, Done) = 1$	
$T(s, a, s') = 0$ otherwise	

Now recall the policy evaluation and policy improvement equations, which together make up policy iteration.

$$\text{Policy Evaluation: } V^{\pi}_{k+1}(s) = \sum_{s'} T(s, \pi(s), s')[R(s, \pi(s), s') + \gamma V^{\pi}_k(s')]$$

$$\text{Policy Improvement: } \pi_{new}(s) \leftarrow \text{argmax}_a \sum_{s'} T(s, a, s')[R(s, a, s') + \gamma V^{\pi_{old}}(s')]$$

Perform two iterations of policy iteration for one step of this MDP, starting from the fixed policy below. Use the initial $\gamma = 1$.

States	0	2	3	4	5
π_0	Move	Stop	Move	Stop	Move
$V_0^{\pi_0}$					
$V_1^{\pi_0}$					
$V_2^{\pi_0}$					
$V_3^{\pi_0}$					
π_1					
$V_0^{\pi_1}$					
$V_1^{\pi_1}$					
$V_2^{\pi_1}$					
$V_3^{\pi_1}$					
π_2					

7 MDPs: Conceptual Questions

1. Which MDP methods are optimal (assuming sufficient convergence when iterating)? [Value Iteration, Policy Iteration, Both, Neither]

2. What are some limitations of value iteration? What are some limitations of policy iteration?

3. When does policy iteration end? Immediately after policy iteration ends (without performing additional computation), do we have the values of the optimal policy?

4. What changes if during policy iteration, you only run one iteration of Bellman update instead of running it until convergence? Do you still get an optimal policy?