Reinforcement Learning HW1

Yonatan Ariel Slutzky Eyal Grinberg

27 March 2023

1 Theoretical Questions

1.1 Question 1

We'll denote $\forall i \in [m], j \in [n]$ the value $T_{i,j}$ to be the length of the longest common sub-sequence of $X[:i] = x_1..x_i$ and $Y[:j] = y_1..y_j$. The obvious solution to the problem is $T_{m,n}$.

1.1.1 Algorithm:

1. $T_{1,1}$ will be 1 if and only if $x_1 = y_1$, and 0 otherwise.
2. $\forall i \in [m] \setminus \{1\}$. $T_{i,1}$ will be equal to

$$max\{T_{i-1,1}, \chi_{x_i=y_1}\}$$

3. $\forall j \in [n] \setminus \{1\}$. $T_{1,j}$ will be equal to

$$max\{T_{1,j-1}, \chi_{x_1=y_j}\}$$

4. $\forall i \in [m] \setminus \{1\}, j \in [n] \setminus \{1\}$. $T_{i,j}$ will be equal to

$$max\{T_{i-1,j}, T_{i,j-1}, T_{i-1,j-1} + \chi_{x_i=y_j}\}$$

1.1.2 Proof:

Given i, j, the longest common sub-sequence is one of the following:
- The longest common sub-sequence without the characters x_i, y_j - i.e. $T_{i-1,j-1}$. To this we can add the character x_i if and only if $x_i = y_j$
- The longest common sub-sequence without the character x_i - i.e. $T_{i-1,j}$
- The longest common sub-sequence without the character y_j - i.e. $T_{i,j-1}$

Our algorithm reflects this and thus it is correct.

1.2 Question 2

1.2.1 Section 1

We'll define the state space S to be the different possible rooms on the grid - $s_{i,j}$ represents the room on the i row and j column (starting from the south-most

west-most corner). Formally:

$$S := \{s_{i,j} | i \in [m], j \in [n]\}$$

We'll define the action space A to be the different possible movements on the grid - north represented by $\uparrow$, east represented by $\rightarrow$. Formally:

$$A := \{\uparrow, \rightarrow\}$$

We'll define $\forall i \in [m], j \in [n]$ the constant $X_{i,j}$ to be an indicator representing the absence of cheese in room i, j (1 if there's no cheese, 0 otherwise).
Next, we'll define $\forall t \in [T-1] \cup \{0\}$ the cost function c_t in the following manner:

$$c_t(s_{i,j}, \uparrow) := X_{i+1,j}$$

$$c_t(s_{i,j}, \rightarrow) := X_{i,j+1}$$

The above is defined only for actions possible in the grid - performing $\uparrow$ from $s_{m,j}$ isn't possible as is performing $\rightarrow$ from $s_{i,n}$.
Since there's only a single possible state in stage T - $s_{m,n}$, defining a cost function c_T isn't required (since it will add a constant to all possible trajectories).
The cumulative cost function is obviously the sum of the above functions (over t).

1.2.2 Section 2

Since the only possible operations are moving north or east, the total amount of operations is always $m - 1 + n - 1$ - that is the horizon of the problem.

1.2.3 Section 3

Any trajectory can be viewed as a possible realization of choosing the $m - 1$ indices of operations in which we move north, out of the total $m - 1 + n - 1$ operations (that any trajectory has).
Therefore, the total amount of trajectories is $\binom{m-1+n-1}{m-1}$.
For $m = 2$, the total amount of trajectories is $\binom{2-1+n-1}{2-1} = \binom{n}{1} = n$.
Thus, the total number is linear w.r.t n.
For $m = n$, the total amount of trajectories is

$$\binom{n-1+n-1}{n-1} = \binom{2n-2}{n-1} = \frac{(2n-2)!}{(n-1)!(n-1)!} = (*)$$

Under Stirling's approximation, $k! \sim \sqrt{2\pi k}(\frac{k}{e})^k$. Therefore

$$(*) \sim \frac{\sqrt{2\pi(2n-2)}(\frac{2n-2}{e})^{2n-2}}{(\sqrt{2\pi(n-1)}(\frac{n-1}{e})^{n-1})^2} = \frac{1}{\sqrt{2\pi}}\sqrt{\frac{2(n-1)}{(n-1)^2}}2^{2n-2}\frac{(n-1)^{2n-2}}{(n-1)^{2n-2}} =$$

$$= \frac{1}{\sqrt{2\pi}}\sqrt{\frac{2}{n-1}}2^{2n-2} = \theta(2^n)$$

1.2.4 Section 4

a. If there are multiple optimal trajectories, then the two mice could benefit from working optimally in parallel (in the case they don't pick the same trajectory). If there exists a single optimal trajectory, by definition the two mice will compete for the same cheese (and thus act sub-optimally).
b+c. In this case, a natural generalization for K mice will be to hold for each mice its room and for each mice its action. Formally:

$$S := \{(s_{i_1,j_1}, .., s_{i_K,j_K}) | i_1, .., i_K \in [m], j_1, .., j_K \in [n]\}$$

$$A := \{\uparrow, \rightarrow\}^K$$

For the general case,

$$|S| = (mn)^K, |A| = 2^K$$

For $K = 2$,

$$|S| = (mn)^2, |A| = 4$$

A subtle observation is that some of the states of S aren't actually possible. In the problem's setting, all of the mice are always in rooms whose sum of indices are equal. However, we decided not to treat this subtlety and leave this note instead.

1.3 Question 3

1.3.1 Section 1

$$P(\text{"}Bob\text{"}) = 0.25 \cdot 0.2 \cdot 0.325 = 0.01625$$

$$P(\text{"}Ok\text{"}) = 0$$

$$P(\text{"}B\text{"}) = 0.325$$

$$P(\text{"}Book\text{"}) = 0.25 \cdot 0.2 \cdot 0.2 \cdot 0.2 = 0.002$$

$$P(\text{"}Booook\text{"}) = 0.25 \cdot 0.2 \cdot 0.2 \cdot 0.2 \cdot 0.2 \cdot 0.2 = 0.00008$$

1.3.2 Section 2

a. We'll define $\forall t \in [K-1]$ the state space S_t to be the possible letters in a word. Formally

$$S_t := \{'B', 'O', 'K'\}$$

In addition, $S_0 := \{'B'\}, S_K := \{'-'\}$.
We'll define the action space to be the upcoming letter in the word (the letter we move to). Formally

$$\forall t \in [K-2] \cup \{0\}. A_t := \{'B', 'O', 'K'\}, A_{K-1} = \{'-'\}$$

An important note is that $\forall t \in [K-1] \cup \{0\}. S_{t+1} = A_t$.
Next, we'll define $\forall t \in [K-2] \cup \{0\}$ the cost function c_t in the following manner

- for each letter $s \in S_t$ and for each upcoming letter $a \in A_t$, we'll define the cost to be the probability to move from s to a:

$$c_t(s, a) := P(s \to a)$$

Since there's only a single possible state in stage K - $'-'$, defining a cost function c_K can be done by $c_K := -1$ - this is to make sure the largest probability has the smallest value.
The multiplicative cost function is obviously the product of the above functions (over t).
b. A naive bound on the complexity is $\theta(3^k)$, in which we try all possible word combinations.
A more tight bound on the complexity can be achieved after modifying the dynamic programming algorithm proposed for finite horizon with cumulative cost, in which we multiply the costs instead of summing. The bound will be $\theta(K|A|) = \theta(K)$.
c. A reduction to the additive cost function can be done by taking ln of the probabilities ($ln(c_K) := 0$ by convention) and taking minus the sum of them instead of multiplying - then we can apply the algorithm seen in class.
If we wish to retrieve the original probability, we can simply raise e to the the power of the received value.
This is possible because lnx is a monotone function and so is e^x.
d. We'll prefer the multiplicative variant when dealing with relatively small values of K. This is due to the fact that the products will be fractions that aren't of too little size, and thus their representations won't be hurt. In addition, this will prevent the use of costly functions such as lnx, e^x etc. We will prefer the additive variant in cases where K is large for the same reasons.
e. The most probable word of length 5 is "BKBKO" and its probability is 0.0067.

1.4 Question 4

1.4.1 Section 1

$$\forall v \in \{B, C, D, E\}.d_0(v) = \infty, d_0(A) = 0$$

$$\forall v \in \{A, C, D, E\}.d_1(v) = \infty, d_1(B) = -1$$

$$\forall v \in \{A, B\}.d_2(v) = \infty, d_2(C) = 2, d_2(D) = 1, d_2(E) = 1$$

$$d_3(A) = 6, d_3(B) = 2, d_3(C) = 6, d_3(D) = -2, d_3(E) = \infty$$

$$d_4(A) = 10, d_4(B) = -1, d_4(C) = 3, d_4(D) = 4, d_4(E) = 4$$

$$d_5(A) = 7, d_5(B) = 5, d_5(C) = 2, d_5(D) = 1, d_5(E) = -1$$

1.4.2 Section 2

Using Karp's theorem, the cost of the minimum mean cost cycle is

$$\mu^* = min_{v\in\{A,B,C,D,E\}}(max_{k\in[3]\cup\{0\}}\frac{d_5(v)-d_k(v)}{5-k}) =$$

$$min\{\frac{7-0}{5-0}, \frac{5-(-1)}{5-1}, \frac{2-2}{5-2}\frac{1-(-2)}{5-3}, \frac{1-1}{5-2}\} = 0$$

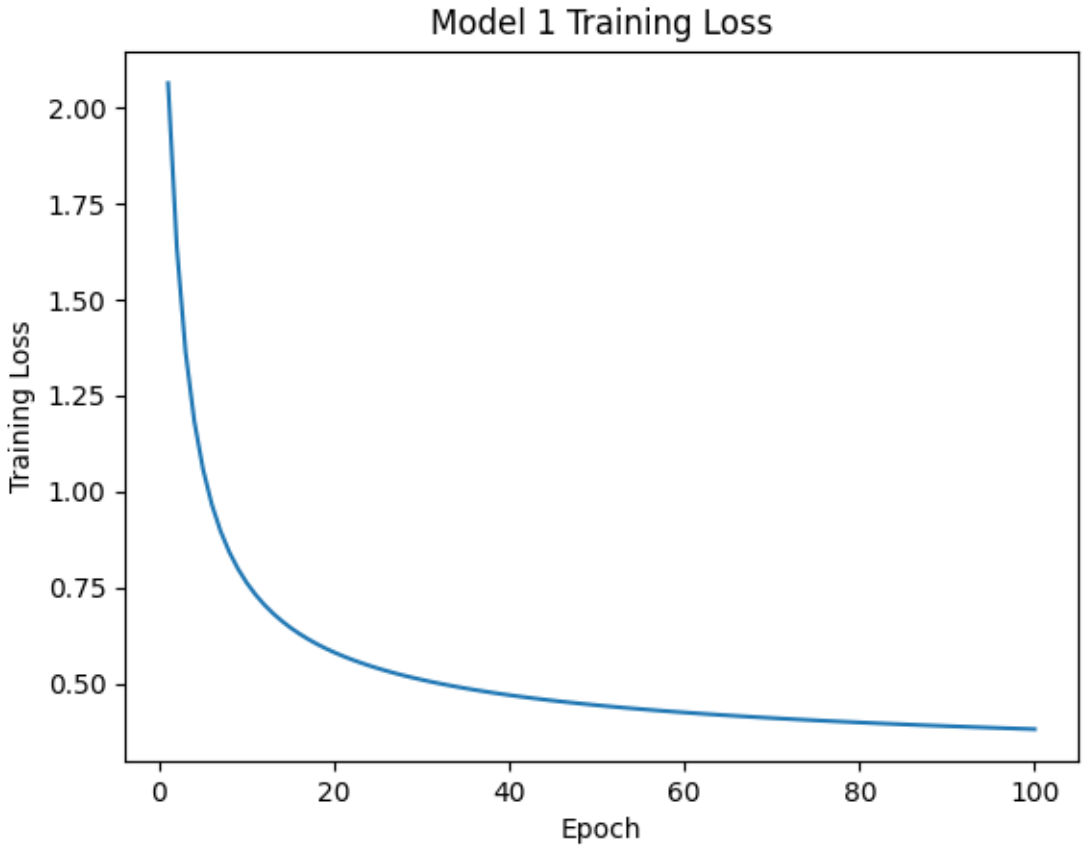
1.4.3 Section 3

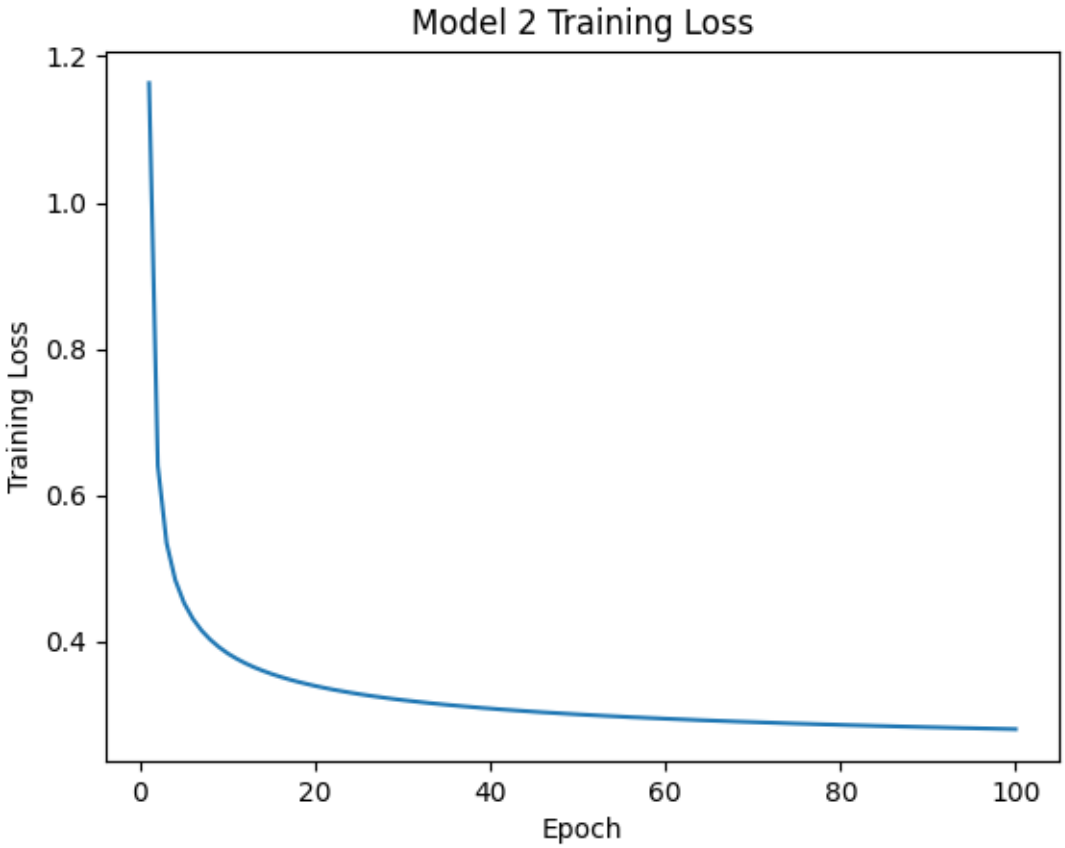
As seen in class, the optimal average cost is the same as the mean cost of the minimum mean cost cycle - 0.

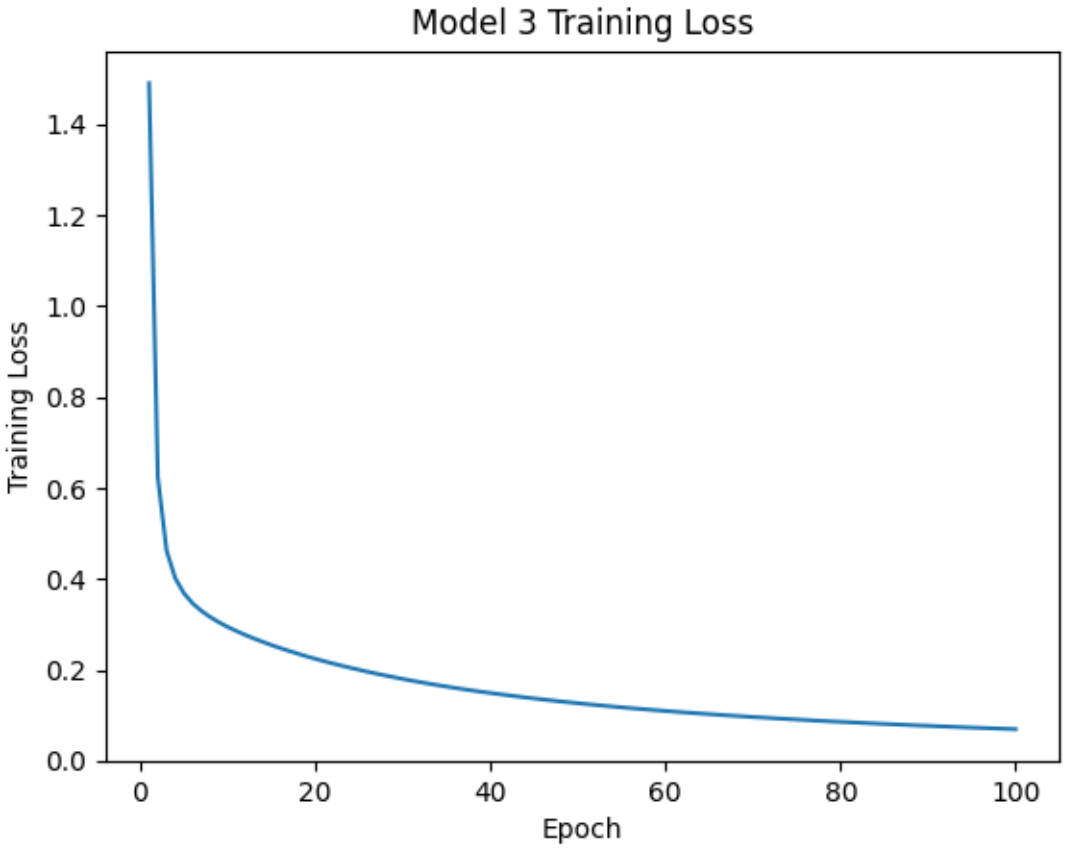
2 Practical Questions

2.1 Question 1

In the first model, we used the given configurations for 100 epochs. The test accuracy was 90%.
In the second model, we added the momentum hyper-parameter to the SGD optimizer, and we used it with size 0.9. The rest of the hyper-parameters are the same. The test accuracy was 92%.
In the third model, we added a ReLU hidden layer to the second model with a hidden width of 500. The test accuracy was 97%.
The following are the plots of the training loss of the different models:

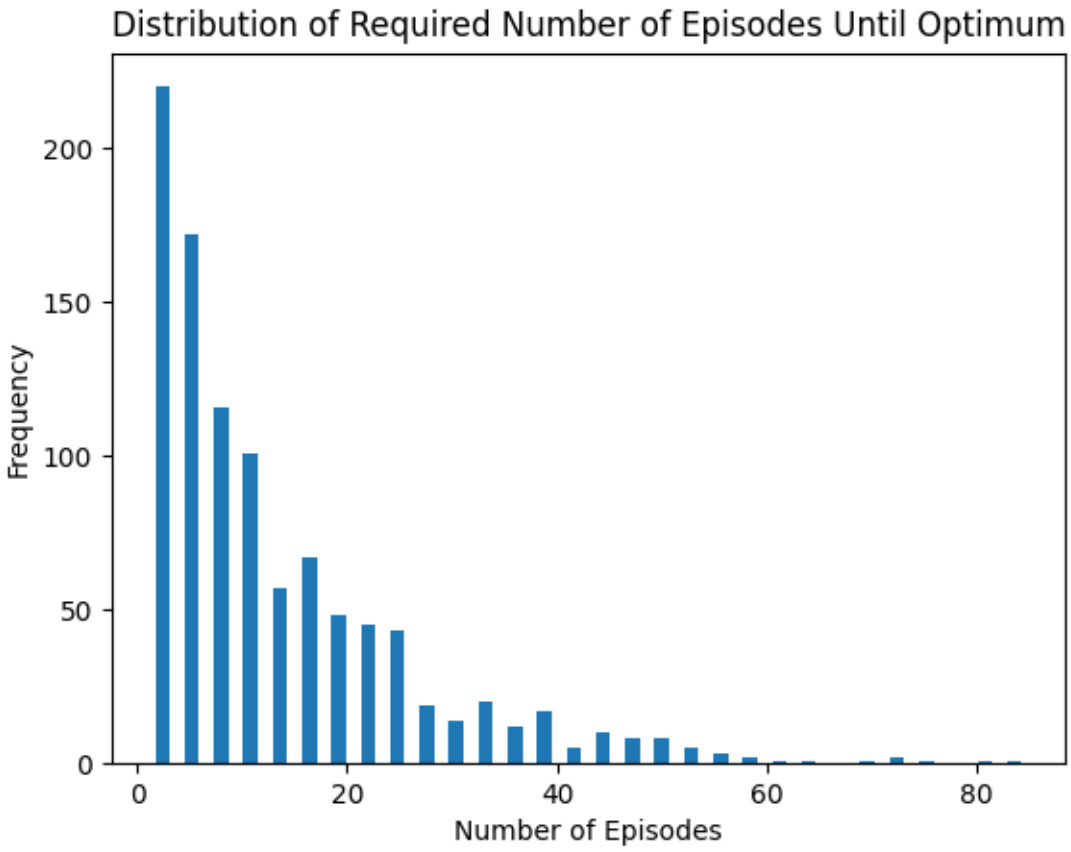

Model 1 Training Loss
Training Loss
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0
20
40
60
80
100
Epoch


Model 2 Training Loss
Training Loss
1.2
1.0
0.8
0.6
0.4
0
20
40
60
80
100
Epoch



2.2 Question 2

The average number of episodes required to reach the optimal score was 13.311.
Below is the histogram of the required episodes to reach the optimal score:


Distribution of Required Number of Episodes Until Optimum
Frequency
Number of Episodes
0
50
100
150
200
0
20
40
60
80