Reinforcement Learning Assignment 1

Question 1: Longest Common Subsequence

For strings $X = x_1 \dots x_n$ and $Y = y_1 \dots y_m$ of size n, m denote $lcs(X, Y)$ to be the size of the longest common subsequence. If $m = 0$ or $n = 0$ then $lcs(X, Y) = 0$. If $x_n = y_m$ then we can choose whether to take x_n and y_m to be part of the subsequence and continue with computing $lcs(x_1 \dots x_{n-1}, y_1 \dots y_{m-1})$, or we can choose to not take them as part of the sequence and calculate $lcs(X, y_1 \dots y_{m-1}), lcs(x_1 \dots x_{m-1}, Y)$. For the case where $x_n \neq y_m$ we can't take them to be part of the solution so we will calculate $lcs(X, y_1 \dots y_{m-1}), lcs(x_1 \dots x_{m-1}, Y)$.

This is concluded in the recursive formula:

$$lcs(X,Y) = \begin{cases} 0 & if\ n = 0\ or\ m = 0 \\ \max\{1 + lcs(x_1 \dots x_{n-1}, y_1 \dots y_{m-1}), lcs(X, y_1 \dots y_{m-1}), lcs(x_1 \dots x_{m-1}, Y)\} & if\ x_n = y_m \\ \max\{lcs(X, y_1 \dots y_{m-1}), lcs(x_1 \dots x_{m-1}, Y)\} & else \end{cases}$$

Define $M_{i,j} = lcs(x_1 \dots x_i, y_1 \dots y_j)$, the solution will be $M\_(n, m)$. From the formula above we can define an iterative algorithm:

Input: $X = x_1 \dots x_n, Y = y_1 \dots y_m$
output: $lcs(X, Y)$

define $\forall i \leq n, j \leq m \quad M_{i,j} = 0$
For $i = 1, \dots, n$ **do**:
  For $j = 1, \dots, m$ **do**:
    if $x_i = y_j$ **then** $M_{i,j} = \max\{1 + M_{i-1,j-1}, M_{i-1,j}, M_{i,j-1}\}$
    else $M_{i,j} = \max\{M_{i-1,j}, M_{i,j-1}\}$

return $M_{n,m}$

Since the algorithm makes n iterations with m steps the running time is $O(nm)$.

Question 2: Moses the mouse

1) State space: $S = \{s_{i,j} | 0 \leq i \leq N, 0 \leq j \leq M\}$

Action space: $A(s_{i,j}) = \begin{cases} \{s_{i+1,j}\}\ if\ j = M, i < N \\ \{s_{i,j+1}\}\ if\ i = N, j < M. \\ \{s_{i+1,j}, s_{i,j+1}\} else \end{cases}$ Meaning the action space is $\{north, east\}$ if there is no wall in the direction.

Cumulative cost function: First define the reward function

$$r_t(s_t, a_t) = \begin{cases} 1 & if\ there\ is\ a\ cheese\ in\ s_t \\ 0 & else \end{cases}$$

$$r_T(s_T) = \begin{cases} 1 & if\ there\ is\ a\ cheese\ in\ s_T \\ 0 & else \end{cases}$$

And $R(path) = \sum_{t=1}^{T-1} r_t(s_t, a_t) + r_T(s_T)$ and we can define $C(path) = -R(path)$.

2) The horizon of the problem is when Moses arrives to north-east corner $s_{N,M}$ which will happen in exactly $M + N$ steps, so $T = N + M$.

3) For $M = 2$, there must be exactly one state $s_{i,j}$ where the action $north$ is taken. So each trajectory is uniquely defined by the state where Moses took the action $north$. Since there are N different states where he can take this action there are N different trajectories.

For the general case, notice that the trajectory is uniquely defined by $m - 1$ states where we choose to take the action $north$. We can think of it like choosing from $m - 1$ elements from a set of size N with repetitions. The formula for choosing k elements from n with repetitions is $\binom{n+k-1}{k}$ so for us it means that the number of different trajectories is $\binom{N+M-2}{M-1}$.

For the case where $N = M$ we get $\binom{2N-2}{N-1}$.

4) a) If both mice will go with the optimal policy, then if we assume that such policy is unique they will go through exactly the same trajectory. In this case each one will get half of the optimal reward (assuming they split the cheese). If there is more than a single optimal policy they might choose different trajectories and increase their reward.

b) For the case where the mice split and can move independently the state space will be $S \times S$ where S is the state space defined in section 1. Meaning the number of states in this case is $|S \times S| = (NM)^2$. Notice that not all the states in this set can actually be reached if we assume both mice take each step simultaneously.

Similarly, the action state is $\{north, east\}^2$, or more accurately $A \times A$ meaning an action consists of two legal action, one for each mice. If both $north$ and $east$ are legal for some state there are 4 different actions.

c) Like the section before, each mouse can choose to go $north$ of $east$ (if such actions are legal). So each choice of all mice corresponds to a different action, meaning the action space has size 2^K.

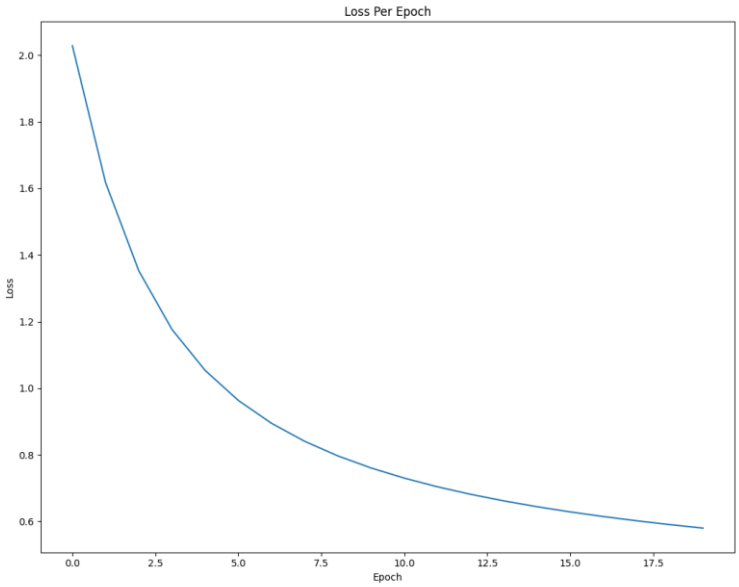
Each mouse has NM states at the room, so together K mice has state space of size $(NM)^K$.

Programing

Question 1: MNIST

1) The following plot shows accuracy per epoch for the parameters given to us:

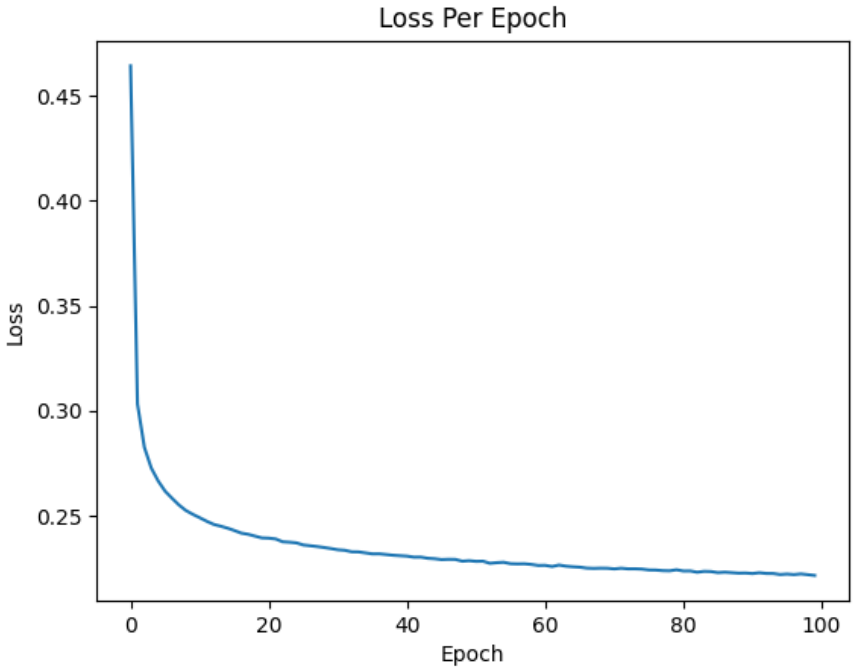
num_epoch = 20
batch_size = 100
learning_rate = 1e-3
optimizer = SGD



Accuracy of the network on the 10000 test images: 87%

2) We found the best parameters to be:

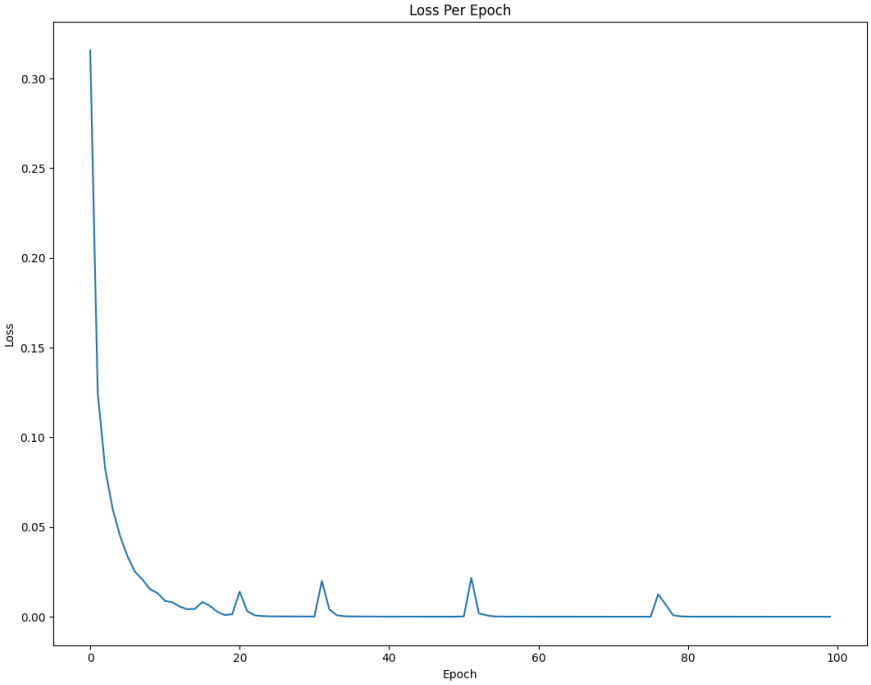
num_epoch = 100
batch_size = 32
learning_rate = 0.001
optimizer = ADAM



Accuracy of the network on the 10000 test images: 92%

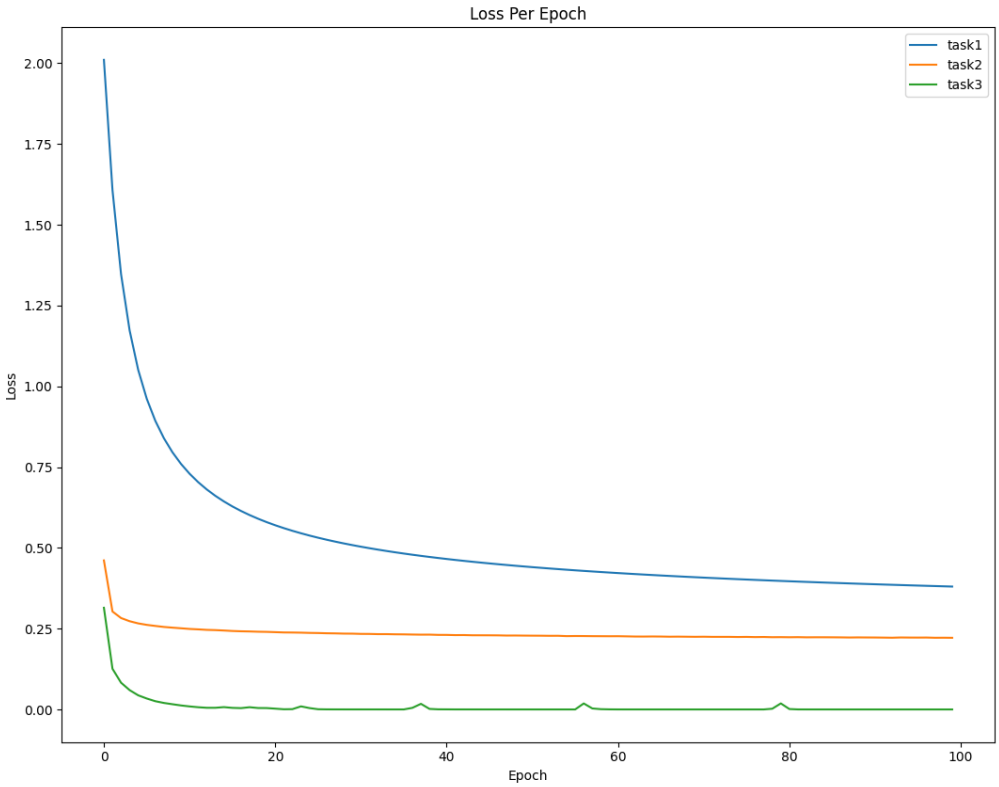
3) We found the best parameters to be:

num_epoch = 100
batch_size = 32
learning_rate = 0.001
optimizer = ADAM



Accuracy of the network on the 10000 test images: 98%

Comparing loss of all parts:



Reinforcement Learning, H.W. 1:

Q.3:

1) $P_1\{\text{"Bob"}\} = \underbrace{(0.25)}_{b\to o} \cdot \underbrace{(0.2)}_{(o\to b)} \cdot \underbrace{(0.325)}_{(b\to -)} =$

0.01625.

(not seen only in B!)

$P_2\{\text{`OK'}\} \neq \underbrace{(0.2)}_{(o\to k)} \cdot \underbrace{(0.2)}_{(k\to -)} = 0.04 \cdot 0 = 0$

$P_1\{\text{`B'}\} = 1 \leftarrow$ (every word must start in B).

$P_1\{\text{"Book"}\} = \underbrace{(0.25)}_{(b\to o)} \cdot \underbrace{(0.2)}_{(o\to o)} \cdot \underbrace{(0.2)}_{(o\to k)} \cdot \underbrace{(0.2)}_{(k\to -)} =$

$= 0.002$.

$P_1\{\text{"Boook"}\} = (0.25) \cdot (0.2) \cdot (0.2) \cdot (0.2) \cdot (0.2) \cdot (0.2) =$

$= (0.25) \cdot (0.2)^5 = 0.00008$.

! (11 RV)

.1

[7] a)

- State-space:

$$\begin{cases} S_0 := \{('B')\} \\ S_{1 \le t \le k-2} := \{('B'), ('k'), ('o')\} \\ S_{k-1} := \{('-')\} \end{cases}$$

$$\Rightarrow \boxed{S = \bigcup_{t=0}^{k-1} S_t}$$

- Action-space:

$$\begin{cases} A_{0 \le t \le k-3} := \{('B'), ('k'), ('o')\} \\ A_{k-2} := \{('-')\} \end{cases}$$

$$\Rightarrow \boxed{A = \bigcup_{t=0}^{k-2} A_t}$$

- Deterministic transition-function:

$$\boxed{\begin{aligned} f_{0 \le t \le k-3}(s_t, a_t) &= a_t \\ f_{k-2}(s_{k-2}, ('-')) &= ('-') \end{aligned}}$$

$$\Downarrow$$

(2)

⇓

• multiplicative cost-function:

$$C_t(S_t, a_t) = 1 - P_r(S_t \to a_t)$$

(זה שקול, למקסום!)

⇒ overall-cost:

$$\prod_{t=0}^{K-1} C_t(S_t, a_t)$$

סוף (a)!

(b)

• נגדיר: $C_t(S)$ להיות העלות המינימלית הכוללת של מסלול שמתחיל ב- s בזמן t עד זמן $K-1$ (עבור $t \le k \le K-1$)

⇐ ונגדיר אלג' DP:

- Init: $C_{K-1}(S) = 1 \quad ; \forall S \in S_{K-1}$

- Backward-Recursion:

for $t = 0, \ldots, K-2$:

$$C_t(S) = \min_{a \in A_t} \{ C_t(S,a) \cdot C_{t+1}(f_t(S,a)) \}$$

$$\pi_t(S) = \arg\min_{a \in A_t} \{ C_t(S,a) \cdot C_{t+1}(f_t(S,a)) \}$$

⇓ 2

• בנוסף כעת, נרצה לסמן את המסלול שמביא למינימום הכולל $L_0(s_0)$, נעשה זאת ע"י שמירת המצבים בשלב הנוכחי ב- $\pi_t(\cdot)$.

- נשים לב, כי בכל שלב, ישנם קודקודי מצבים של גרף בגודל 4 מצבים, וקודקודים נוספים בשכבה הבאה בגודל $4 = C$ ולכן בכל שלב 16 פעולות בכל תת רקורסיה, מאחר ויש $O(k)$ רמות רקורסיה, נקבל סיבוכיות של $O(k)$.

סוף (b)!

(C)

• כמו רקורסיבית, נפעיל ~~אופרטור~~ אופרטור $\log(\cdot)$ על פעולת הכפולות הנדרשות:

$$\log\left(\prod_{t=0}^{k-1} C_t(s_t,a_t)\right) = \sum_{t=0}^{k-1} \log(C_t(s_t,a_t))$$

← מחבר + הפונקציה מונוטונית עולה ⇐ נקבל מינימום עבור הרקורסיה.

סוף (c)!

(d)

- ישנם כמה שיקולים. מצד אחד נרצה מכפלות קטנות כשמספר שגיאות קטנה, מצד שני, בחירה כי מכפלה של ערכים קטנים מאוד מובילה לאיבוד דיוק של מספר שהוא קטן מדי בקשר שבין שגיאות.

סוף (d)!

BK

14

Intro RL – EX1

Theoretical section:

Question 4:

1. The following is the results of calculating $d_i(v)$ for every $i \in \{0, \dots, n=5\}, v \in \{A, B, C, D, E\}$ given initial state $s = A$:

	d_0	d_1	d_2	d_3	d_4	d_5	$max(\frac{d_n(v)-d_k(v)}{n-k})$
A	0	∞	∞	6	10	7	7/5
B	∞	-1	∞	2	-1	5	6
C	∞	∞	2	6	3	2	0
D	∞	∞	1	-2	4	1	3/2
E	∞	∞	1	∞	4	1	0

2. According to Karp's theorem the cost of the minimum mean cost cycle is:

$$u^* = \min_{v \in V} \max_{0 \le k \le n-1} \frac{d_n(v)-d_k(v)}{n-k} = 0$$

   This is obtained for either $(v = C, k = 2)$ or $(v = E, k = 2)$.

3. As was proven in lecture 2, the optimal average cost of a DDP equals exactly to the cycle with the lowest average cost, hence it is 0. The cycle is B>E>D>B.

Programming section:

Question 2:

4. The average number of required random initializations is: 13.491.
   Below is the histogram: