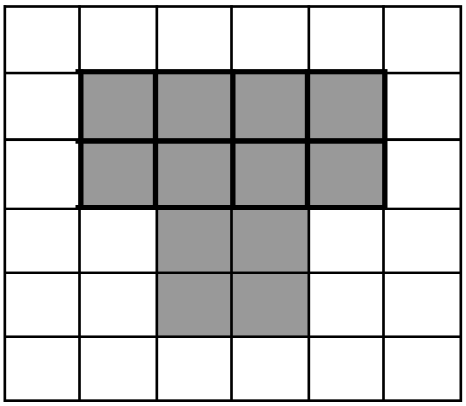
ML Day 2 Solutions

Faculty Name: Dr. Gagan Raj Gupta
Presented by: Arnav Gautam

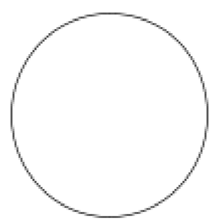
Q5 Given a 6x6 input: (The dark **bordered** cells comprise the **roof** in this problem).



a) Compute Output using the given filter:

-1	1	-1
-1	1	-1
-1	1	-1

1. Stride = 1
2. Padding = 0



ReLU

b) Design a network filter to detect the roof.

Sol 5

a)
In the given 6x6 **input**, we take the dark cells as 1 and light cells as 0 for performing the CNN.

Taking 1 stride at a time (given), we create the following feature map:

0	-2	-2	0
-1	-2	-2	-1
-2	-1	-1	-2

-2	0	0	-2

Methodology in order to obtain the above 4x4 feature map.

Initially the box of 3x3 (which is **kernel**) is crossed with the input network:

-1	1	-1
-1	1	-1
-1	1	-1

which is kept at the top left. We then perform a dot multiplication with the values there in this box with the input network (6x6 matrix). The net addition is the first value of the feature map (4x4). Now that the first cell is computed, we move on to the next cell of the feature map (move 1 cell towards right since stride is 1).
Here is an image for better understanding: LINK

Tip: Feature Map dimensions can be calculated using the following formula:
Output Height = ((N - K + (2 * P)) / S) + 1 → {You can think it analytically in you mind}
Where,
N = Dimensions of Input Matrix
K = Dimensions of Kernel
P = Padding
S = Stride
Here we got 4x4 by: (6 - 3 + (2 * 0) / 1) + 1 = 4.

---

b)

Network Filter:
For creating such a network filter for detecting the roof, we can put a **padding of 1** and then perform the dot product with the kernel, keeping the stride the same. This would detect the roof in the given input matrix (6x6).

Here is the filter that fits the best for finding the roof in the input:

0	0	0	0	0	0
0	**1**	**1**	**1**	**1**	0
0	**1**	**1**	**1**	**1**	0
0	0	0	0	0	0

Q6 Attention

5	6	-2	-5	3
2	3	-1	-2	2
-1	3	5	1	3
-3	-4	4	-3	2

Input:

1. Key = Value
2. All w matrix = $[1]_{identity}$

Compute the context.

-3
-2
4
1

Query

Sol 6

We take the dot product of Input Matrix and Query to get:

-15	-18	6	15	-9
-4	-6	2	4	-4
-4	12	20	4	12
-3	-4	4	-3	2

….. (1)

Now we take the **softmax**:

For each cell, we take $softmax\ of\ each\ cell\ given\ by$: $\frac{e^x}{\sum_{x \in col} e^x}$. We calculate this for each cell.

3.57×10^{-6}	9.35×10^{14}	8.31×10^{-7}	0.989	7.58×10^{-10}
0.21	1.52×10^{-8}	1.52×10^{-8}	1.67×10^{-5}	1.125×10^{-7}
0.21	0.999	0.998	1.67×10^{-5}	0.997
0.58	1.12×10^{-7}	1.12×10^{-7}	1.52×10^{-8}	4.54×10^{-5}

For finally finding the context, we multiply the values of the matrix **(1)**:

-5.3×10^{-5}	-1.6×10^{16}	4.986×10^{-6}	14.835	-6.822×10^{-9}
-0.84	-9.12×10^{-8}	3.04×10^{-8}	6.68×10^{-5}	-4.5×10^{-7}
-0.84	11.988	19.96	6.68×10^{-5}	11.964
1.74	-4.48×10^{-7}	4.48×10^{-7}	-4.56×10^{-8}	9.08×10^{-5}

The above matrix is the context that was required in this problem.

Q7 IPCA Based Auto Encoder

Assume that the data matrix can be written as:

$$U \begin{bmatrix} 5 & & & \\ & 2 & & \\ & & 0.5 & \\ & & & 0 \end{bmatrix} V^T$$

(There can be other values as well (non - diagonal values), but that is not required for finding the answer)

What will be the expected reconstruction error if we use **dim(h) = 2**?

Sol 7

- We know that the energy is always in the diagonal of the matrix.

Given that dim(h) = 2, we just have to take the first 2 principal components of the matrix.

- Taking the 2 principal components: $5^2 + 2^2 = 29$
- Total components = $5^2 + 2^2 + 0.5^2 + 0^2 = 29.25$

We know that the reconstruction error is given by:

Reconstruction error = $(1 - \frac{29}{29.25}) \times 100$

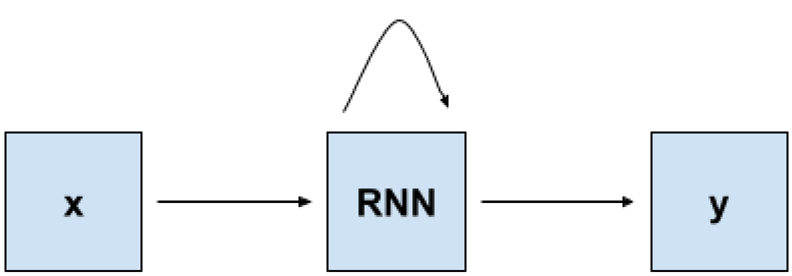
$\Rightarrow 0.8\%$

Q8 RNN

a. Design a RNN that flips the bits in the input:
   i. 1001 → 0110
b. Design a neural network that counts number of times each of the patterns were seen in the input sequence:
   i. 1001
   ii. 0101

Sol 8

<u>General structure of RNN:</u>



$h_t = f(h_{t-1}, x_t)$
$y_t = g(h_t)$

a.

$h_t = f(h_{t-1}, x_t)$

$y_t = g(h_t)$

$h_t = the\ state\ at\ t$
$x_t = the\ input\ at\ t$
$y = output\ based\ on\ the\ state\ h_t$
$f,\ g = functions\ corresponding\ to\ RNN\ architecture$

In the problem, we need to flip the bits: $y_t = \sim x_t$

We can say:

$$y_t = \begin{cases} 1 - x_t, & x_t = 1 \\ x_t & x_t = 0 \end{cases}$$

$\Rightarrow h_t = x_t$ since the current state depends only on the current input (not any previous input).

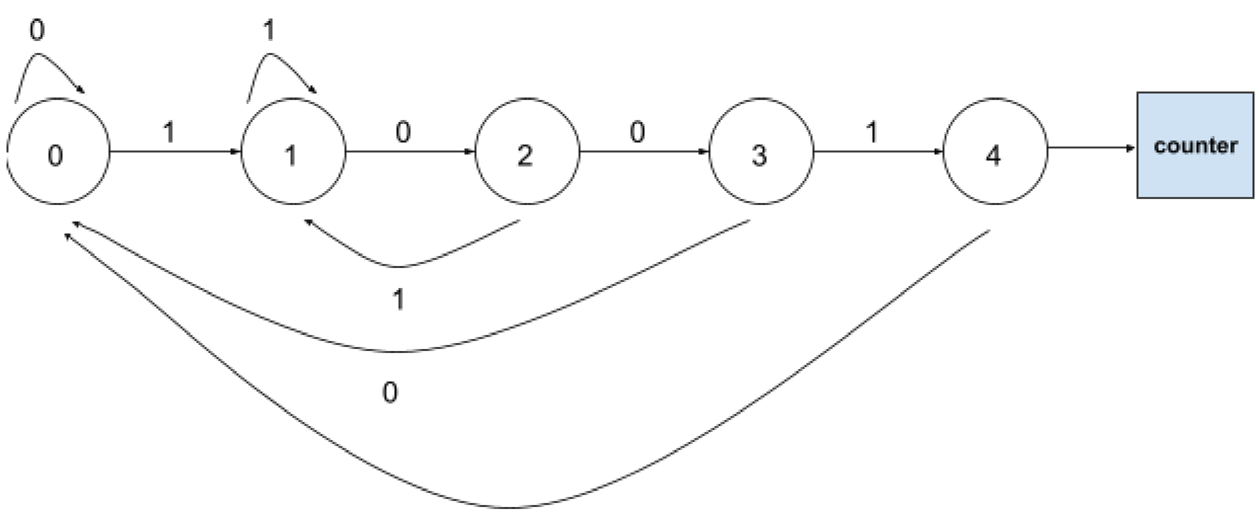
$y_t = g(h_t) = g(x_t)$

$$y = g(x_t) = \begin{cases} 1 - x_t, & x_t = 1 \\ x_t + 1 & x_t = 0 \end{cases}$$

Hence, this is the RNN function that returns the flipped input bits.

---

b.
i.



Note: The difference lies in the use of "**counter**" here in the state machine. Now since we need to count the number of appearances of each pattern in the input sequence (would be given by the user).

Else remains the same as before, this is how we will proceed after:

Converting this state machine to RNN:

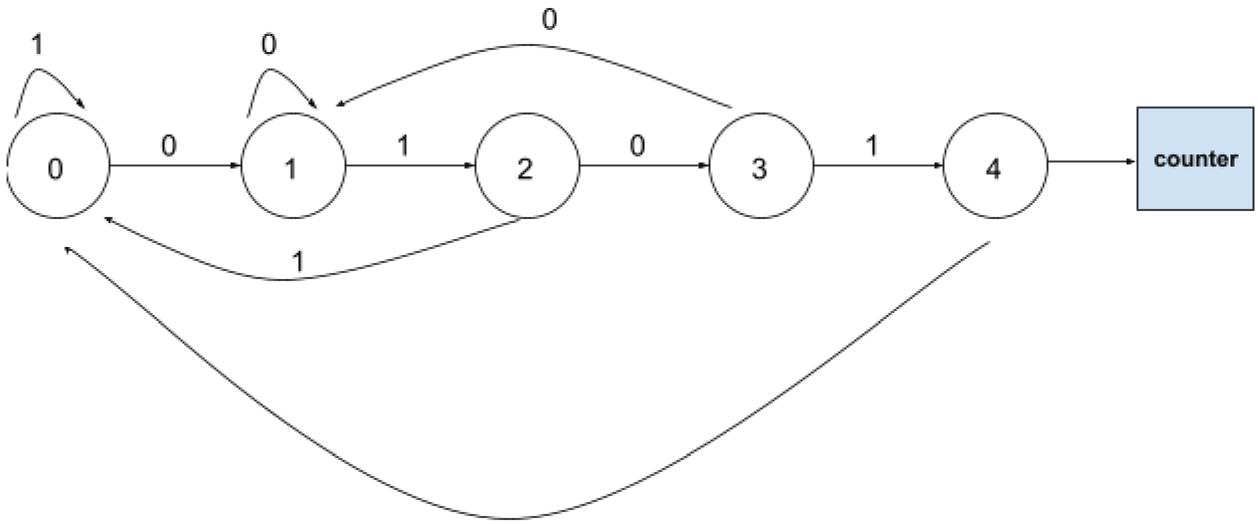
For determining the next state:

h_{t-1}	x_t	$h_t = f(h_{t-1}, x_t)$
0	**0**	**0**
0	**1**	**1**
1	**0**	**2**
1	**1**	**1**
2	**0**	**3**
2	**1**	**1**
3	**0**	**0**
3	**1**	**4**
4	**-**	**0**
4	**-**	**0**

h_t	y_t
0	**0**
1	**0**
2	**0**
3	**0**
4	**1**

---

ii.



Converting this state machine to RNN:

For determining the next state:

h_{t-1}	x_t	$h_t = f(h_{t-1}, x_t)$
0	**0**	**1**
0	**1**	**0**
1	**0**	**1**
1	**1**	**2**
2	**0**	**3**
2	**1**	**0**
3	**0**	**1**
3	**1**	**4**
4	**-**	**0**
4	**-**	**0**

h_t	y_t
0	**0**
1	**0**
2	**0**
3	**0**
4	**1**