(12)

(1)

Kajol Taresh Shah
A20496724
Spring 2022 CS512

Assignment 4

Q1. Neural networks

(a)

Ans. In neural network, each unit defines a template $\theta . x$ which is template matching to see how well the filter fits in the image.

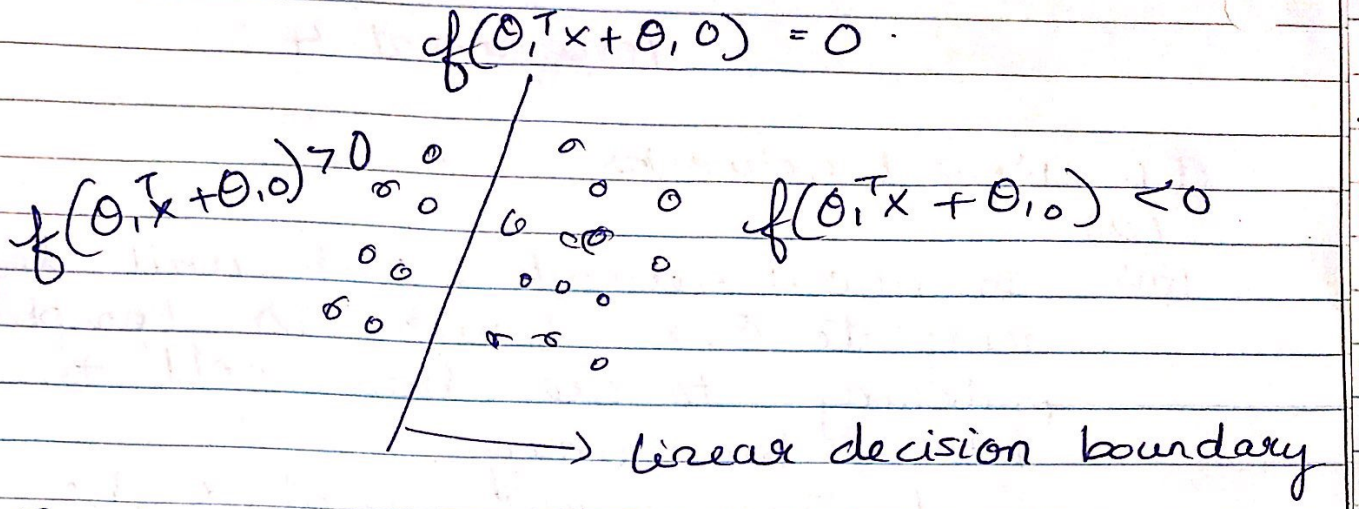
Let the weight matrix be:-

$$\theta^T = \begin{bmatrix} \theta_1^T \\ \theta_2^T \\ \vdots \\ \theta_K^T \end{bmatrix}$$

The rows of the weight matrix θ^T represent templates. With K rows, we have K templates (one template per class). The product $\theta^T x$ measures how well the input vector matches the template (dot product between vectors). High similarity to the template of a particular class indicates membership in this class.

- Decision boundary - interpretation of a linear classifier

Each unit defines a linear decision boundary in linear discrimination. The rows of the weight matrix θ^T represents parameters of K linear discriminant functions. The discriminant function measures distance from a linear linear decision boundary.



If the example belongs to the class, the distance is positive i.e $f(\theta_1^T x + \theta_0) > 0$
If it is negative i.e $f(\theta_1^T x + \theta_0) < 0$, it belongs to other class.
The linear decision boundary separates two classes or one class from other classes.

③

Q1.b)

Ans. To convert a similarity score(s_j) to probability, we need to first compute

$$\hat{y}_j^{(i)} \equiv P(y=j \mid x^{(i)})$$

output prob of j^{th} unit.

Then, in a 2-class classification case, similarity score will be converted to probability in the following way:-

$$\hat{y}_j^{(i)} \equiv P(y=j \mid x^{(i)}) = \text{sigmoid}(s_j^{(i)})$$

where $\text{sigmoid}(x) = \frac{1}{1+e^{-x}}$

It converts value ~~to~~ between 0 & 1.

In a K-class classification case

$$\hat{y}_j^{(i)} = P(y=j \mid x^{(i)}) = \frac{\exp(s_j^{(i)})}{\sum_{c=1}^{K} \exp(s_c^{(i)})}$$

softmax

Softmax is used for K-class classification case

Q1.c)

Ans. L1 Loss : $L_i(\theta) = \sum_{j=1}^{K} |\hat{y}_i^{(i)} - y_j^{(i)}|$

L2 loss : $L_i(\theta) = \sum_{j=1}^{K} (\hat{y}_j^{(i)} - y_j^{(i)})^2$

Huber Loss

$$S_\delta(d) = \begin{cases} \frac{1}{2}d^2 & \text{if } |d| \leq \delta \\ \delta\left(d - \frac{1}{2}\delta\right) & \text{otherwise} \end{cases}$$

quadratic for small d
linear for large d

$$L_i(\theta) = \sum_{j=1}^{k} S_\delta\left(\hat{y}^{(i)} - y^{(i)}\right)$$

Cross entropy loss

- Likelihood (↑ maximize): $L(\theta) = \prod_{i=i}^{m} \prod_{j=1}^{k} \left(\hat{y}_j^{(i)}\right)^{y_j^{(i)}}$

- negative log-likelihood (↑ minimize): $L(\theta) = -\log L(\theta)$

$$= \sum_{i=1}^{m} \sum_{j=1}^{k} y_j^{(i)} \log\left(\hat{y}_j^{(i)}\right)$$

(sample loss)

Simpler explanations are better and a simpler solution is when the weights θ are lower (eg. when $\theta_{ij} = 0$, we remove our coefficient). In order to avoid overfitting, we add the regularization term to the loss function. Also, if the values of the weights are high or large, we add regularization term to peralize it. Smaller coefficients give more stable solutions that will generalize better.

(5)

$$L(\theta) = \frac{1}{m} \sum_{i=1}^{n} L_i(f(x_{ij}w), y_i) + \lambda R(\theta)$$

where $R(\theta) = \sum_i \sum_j \theta_{ij}^2$ — regularization term

where $\lambda \rightarrow$ weight of regularization (hyper parameter)

We use the regularization term so that the coefficients don't take extreme values. ~~for eg.~~ L2 Regularization minimizes weights while spreading them.

Q1. e)

Ans. In gradient descent algorithm, we take a step in the opposite direction of the gradient descent algorithm in order to minimize or reduce the loss function. Therefore, we subtract the gradient instead of adding it in the gradient descent algorithm.

$$\theta^{(i+1)} \leftarrow \theta^{(i)} - \eta \nabla L_j(\theta^{(i)})$$

Q1. f)

Ans. In gradient descent, the parameters are updated after processing all examples.

In stochastic gradient descent, ~~the~~ instead of updating the parameters after processing all examples, the parameters are updated after one example.

In gradient descent, updates are not very frequent but more accurate.

In stochastic, updates are more frequent but less accurate.

Q1 g)
Ans. The learning rate should be selected correctly in the gradient descent algorithm. If the learning rate(η) is too large, we will not get minimum or optimal weights. If η is too small, we will take too much time and iterations to reach optimal solution. So, the learning rate need not be constant. We can try different values for learning rate and choose the one which gives the best result. As said earlier, learning rate need not be fixed and we can make it smaller as iterations progress.

Q1 h) The purpose of momentum in gradient descent is to speed up the learning process and converge towards the minima in a faster pace. Momentum stores the average of previous gradients which decay exponentially.

Q1 i)
Ans. In back propagation algorithm, the network is represented using a computational graph. In the forward pass, we push input to compute all intermediate (na node values. In the backward pass, starting with the end nodes, we push

(7)

the gradients towards the beginning. We multiply the backpropagated gradients (from back) by current gradient and propagate this to the next node.

Q1. j)
Ans. In a fully connected layer, each neuron in the layer is connected to every other neuron in the preceding layer. ~~In a~~ So, all input nodes affect all output nodes in this case as change in one neuron can affect the other neurons as well. Convolutional layers are not densely connected i.e. a neuron in a convolutional layer is not connected to every neuron in the preceding layer, it is connected to only a few nearby neurons in the preceding layer. and same weights are used for each neuron.

Q1. k)
Ans. Large number of parameters tend to overfit. So, to prevent overfitting in the neural network, dropout is introduced. This is to introduce some randomness in the network. At each training stage, dropout units in fully connected layers with probability of (1-p), where p is hyperparameter. The removed nodes are reinstated with original weights in the subsequent stage.

(12)

(8)

Q2. Convolution Neural Networks.

a) 4x4 RGB Image

R ≡

1	1	1	1
1	1	1	1
1	1	1	1
1	1	1	1

G ≡

2	2	2	2
2	2	2	2
2	2	2	2
2	2	2	2

B ≡

1	1	1	1
2	2	2	2
3	3	3	3
4	4	4	4

Filter ≡

1	1	1
1	1	1
1	1	1

R * Filter ≡

1	1	1	1
1	1	1	1
1	1	1	1
1	1	1	1

*

1	1	1
1	1	1
1	1	1

=

9	9
9	9

G * Filter ≡

2	2	2	2
2	2	2	2
2	2	2	2
2	2	2	2

*

1	1	1
1	1	1
1	1	1

=

18	18
18	18

(9)

B * Filter $\equiv$

1	1	1	1
2	2	2	2
3	3	3	3
4	4	4	4

$*$

1	1	1
1	1	1
1	1	1

$\equiv$

18	18
18	18

$= (3 + 6 + 9)$

$\equiv$

18	18
27	27

Final Image $=$

9+18+18	9+18+18
9+18+27	9+18+27

Final Image $=$

45	45
54	54

10

(Q2.b) With zero padding.

R * Filter =

0	0	0	0	0	0
0	1	1	1	1	0
0	1	1	1	1	0
0	1	1	1	1	0
0	1	1	1	1	0
0	0	0	0	0	0

*

1	1	1
1	1	1
1	1	1

=

4	6	6	4
6	9	9	6
6	9	9	6
4	6	6	4

G * Filter =

0	0	0	0	0	0
0	2	2	2	2	0
0	2	2	2	2	0
0	2	2	2	2	0
0	2	2	2	2	0
0	0	0	0	0	0

*

1	1	1
1	1	1
1	1	1

=

8	12	12	8
12	18	18	12
12	18	18	12
8	12	12	8

B * Filter =

0	0	0	0	0	0
0	1	1	1	1	0
0	2	2	2	2	0
0	3	3	3	3	0
0	4	4	4	4	0
0	0	0	0	0	0

*

1	1	1
1	1	1
1	1	1

B * Filter =

6	9	9	6
12	18	18	12
18	27	27	18
14	21	21	14

1
1
1

Final Image =

4+8+6	6+12+9	6+12+9	4+8+6
6+12+12	9+18+18	9+18+18	6+12+12
6+12+18	9+18+27	9+18+27	6+12+18
4+8+14	12+6+21	12+6+21	4+8+14

=

18	27	27	18
30	45	45	30
36	54	54	36
26	39	39	26

1

1
0
1
0
1

(12)

(Q2. c)
Ans. Dilation rate = 2.

R * Filter =

0	0	0	0	0	0	
0	1	1	1	1	0	
0	1	1	1	1	0	
0	1	1	1	1	0	
0	1	1	1	1	0	
0	0	0	0	0	0	

*

1	1	1
1	1	1
1	1	1

=

4	4
4	4

Dilated kernel

1	0	1	0	1
0	0	0	0	0
1	0	1	0	1
0	0	0	0	0
1	0	1	0	1

G * Filter Dilated =

0	0	0	0	0	0	
0	2	2	2	2	0	
0	2	2	2	2	0	
0	2	2	2	2	0	
0	2	2	2	2	0	
0	0	0	0	0	0	

*

1	0	1	0	1
0	0	0	0	0
1	0	1	0	1
0	0	0	0	0
1	0	1	0	1

=

8	8
8	8

B * Filter dilated =

0	0	0	0	0	0	
0	1	1	1	1	0	
0	2	2	2	2	0	
0	3	3	3	3	0	
0	4	4	4	4	0	
0	0	0	0	0	0	

*

1	0	1	0	1
0	0	0	0	0
1	0	1	0	1
0	0	0	0	0
1	0	1	0	1

=

12	12
8	8

(16)

(13)

Q
A

Final Image = $\begin{bmatrix} 4+8+12 & 4+8+12 \\ 4+8+8 & 4+8+8 \end{bmatrix}$

$= \begin{bmatrix} 24 & 24 \\ 20 & 20 \end{bmatrix}$

(16)
(14)

(Q 2. d)

Ans. Template matching interpretation of convolution is the process in which a template or filter is moved over the entire image and then similarity is calculated between the filter and the part of the image that the filter covers. The value of the output pixel is calculated by multiplying the elements of the filter and the covered window on the image. The filter represents the template, also called as kernel. Dot product between the image window & filter is taken (dot product + bias). To move the template/filter over the image, Stride is used. Because of this, the activation map maybe smaller (depends on the value of stride which by default is 1) The template matching of convolution is used to detect features in the image. (for eg. straight lines, eyes, nose, etc)

(16)

(15)

(Q2 e).

Ans. If, for multiple scale analysis, we keep changing the filter size or window size, it might be computationally inefficient. So, instead of scaling the filter or window size, we scale the input image instead. In this way, a pyramid can be formed from a single input image by using various scale factors to reduce it. Then, we can use the fixed window size filter to process the scaled images. If the given size of an image is $I(i,j)$ of $M \times N$ dimensions, the scaled version of the image with scale factor s can be obtained which will have dimensions $\left[\frac{M}{s}\right] \times \left[\frac{N}{s}\right]$. These scaled images can be obtained by either subsampling the image or subaveraging it. In subsampling, a sample from each neighbourhood of size $s \times s$ is selected and used as pixel value in the scaled image. It assumes that the sample appropriately represents its neighbourhood and requires $\left[\frac{M}{s}\right] \times \left[\frac{N}{s}\right]$ computations

In subaveraging, the sample from each neighbourhood will have the average intensity of its neighbourhood.

Q 2. f)

Ans. In order to compensate for spatial resolution decrease, we increase the depth i.e. increase the number of channels. by sampling. For eg. If we start with 3 channels, we sample the input and increase the number of channels to 10 and then 20 and so on. As we go deeper into the network, we will have more and more units. Deeper layers have larger receptive fields and correspond to more specific features. With the increase in depth of the network, we can detect complex spatial shapes from spatial features. So, as we go deeper into the layers, we can capture more patterns i.e. more channels leads to capturing more features.

(11)

Q.2.g)

Ans. Image size :- $128 \times 128 \times 32$

No. of convolution filters = 16

size of filter = $3 \times 3 \times 32$.

without zero padding

Feature map size (O) = Input Image size − kernel size + 1

$$O = 128 - 3 + 1$$

$$O = 126$$

So, size of resulting tensor for 1 filter will be 126×126

For 16 filters, size of the resulting tensor will be ~~$126 \times 126 \times 40$~~

$126 \times 126 \times 16$

Q2.h) $$O = \left[\frac{\text{I/P image size} - \text{kernel size}}{\text{Stride}}\right] + 1$$

$$O = \left[\frac{I - K}{S}\right] + 1$$

$$= \left[\frac{128 - 3}{2}\right] + 1$$

$$= \frac{125}{2} + 1$$

$$= 62 + 1$$

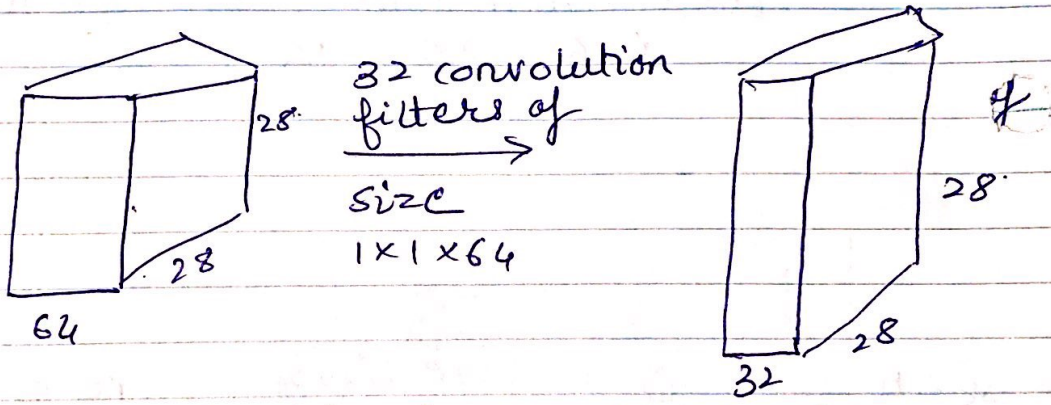
$$O = 63$$

So, size of resulting tensor for 1 filter with Stride will be 63×63. For 16 filters, it will $63 \times 63 \times 16$

Q2.i)

Ans. 1×1 convolution is used to reduce dimensions i.e. Lower the number of channels. The 1×1 convolution filter spans the entire depth of the input. It takes the weighted sum of all the layers and combines them. So, if we have an image of depth M and we want an image with depth N, we will apply N convolution filters of size 1×1 and will get the output image with depth N.

Eg.



depth reduced from 64 to 32.

Q2. i) Ans. Convolutional layers transform the input image in order to extract features from it. Convolutional layers consists of filters which have size smaller than the image. These filters are convolved with the input image and we get the corresponding output. We can convolve image with small filters and share weights of filter between locations (shift invariance). With convolution layers, we can extract image patches. We do this by vectorizing image window and filter (dot product + bias). The convolution filters extend full depth of image and we can have multiple convolution filters per location. Stride is used to move filter and so activation map can be smaller. Early convolution layers learn features like lines, edges, etc. while deeper layers can learn much more complex features like faces. To capture complex features, more channels are needed to detect them. Early convolution layers cannot detect these complex features and can identify only simple features.

Q2.k) Pooling is used to reduce the size of the image, Its function is to reduce dimensions and the number of computations required. It reduces overfitting as the number of parameters are also reduced. It makes the model more tolerant towards variations and distortions. There are two types of pooling:-

Max Pooling - will calculate the maximum number from each window of the feature map.

Average Pooling - will calculate the average of each window in the feature map.

Q2. L) I = 4 X 4 RGB.

R =

1	1	1	1
1	1	1	1
1	1	1	1
1	1	1	1

Filter = 2×2
stride = 2.

After max pooling,

R =

1	1
1	1

G =

2	2	2	2
2	2	2	2
2	2	2	2
2	2	2	2

After max pooling

G =

2	2
2	2

(21)

B =

1	1	1	1
2	2	2	2
3	3	3	3
4	4	4	4

After max pooling

B ≡

2	2
4	4