22nd SCSE – Past Year Paper Solution (2021 – 2022 Semester 1)
CE/CZ 4042 – Neural Networks and Deep Learning

1 **(a)** **(i)** (1, 2, 1, 3)

(ii) Yes, because the activation function is a logistic one which does not give the same output when the inputs have been multiplied by the same constant. They need to be multiplied by varying ones according to the logarithmic graph for the accuracy to remain the same.

(iii) Change the nonlinear function into a linear one by substituting the variables by a variation of x (x1, x2, x3).

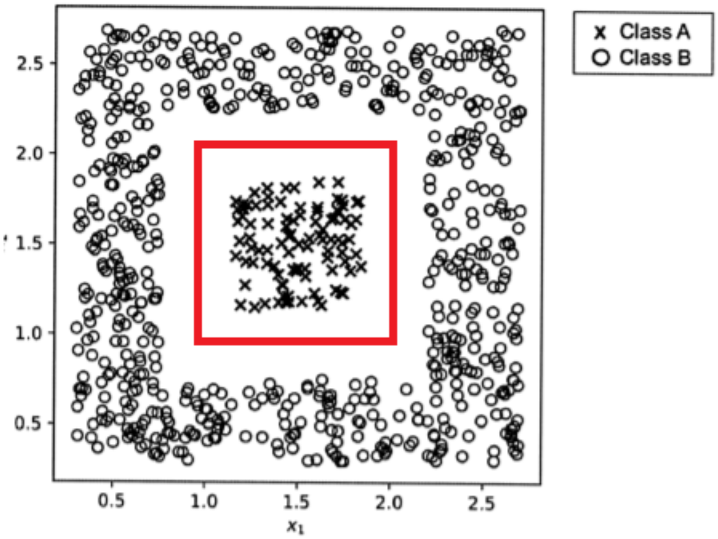
(iv) Easier to fit in memory over batch gradient. It is more computationally efficient than stochastic.

(v) No, it can lead to the exploding gradient problem. Better initialization strategies could be used like random or Xavier initialization.

(vi) (2 x 5) + (5 x 3) + 5 = 30

(vii) 3-way over CV: Decreases training time – exclusive test set – good generalization.
CV over 3-way: Reduces overfitting – better for lesser amount of data.

(b) **(i)**

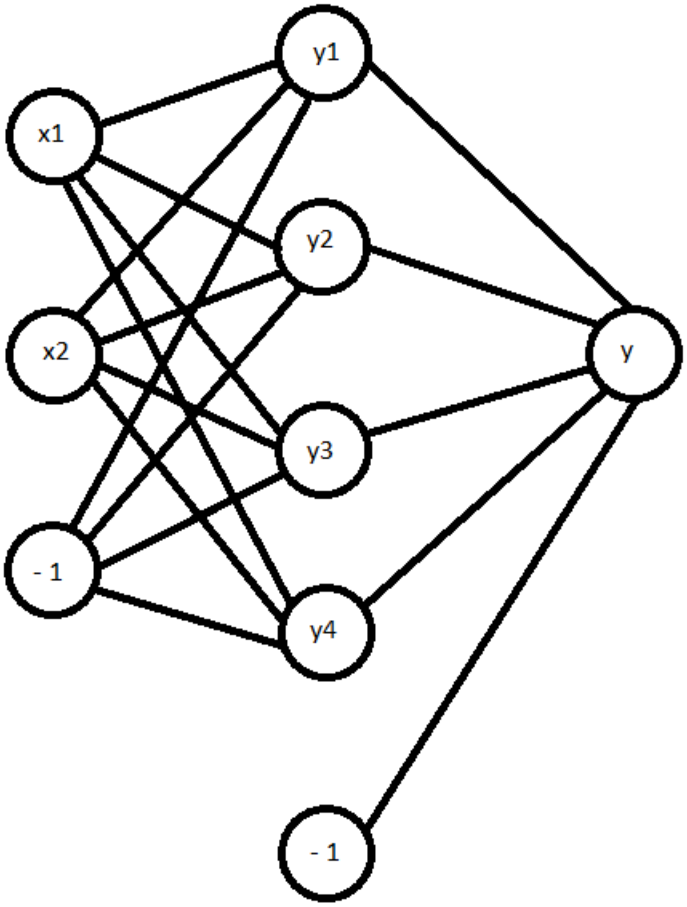


(ii) 4

(iii) Let us initialize all biases to 0.
Using normal distribution initialization,
Weights connecting input layer to hidden layer:

w ~ Uniform Distribution [- $(6^{1/2}/(2+4)^{1/2}$, + $(6^{1/2}/(2+4)^{1/2}$] = [-0.8660 , +0.8660]
w = [-0.8660, -0.6495, -0.433, -0.2165,
0, 0.2165, 0.433, 0.6496]

Weights connecting hidden layer to output layer:
w ~ Uniform Distribution [- $(6^{1/2}/(4+1)^{1/2}$, + $(6^{1/2}/(4+1)^{1/2}$] = [-1.095 , +1.095]
w = [-1.095,
-0.5475,
0,
0.5475]



2 **(a)** W = $\begin{matrix} -0.1 & 0.5 & 0 \\ -0.3 & 0.4 & 0.6 \end{matrix}$

b = $\begin{matrix} 0.2 \\ 0.2 \\ 0 \end{matrix}$

$$V = \begin{matrix} 5.0 & 0 \\ 2.0 & -4.0 \\ 3.0 & 6.0 \end{matrix}$$

$$c = \begin{matrix} 0.2 \\ 0.2 \end{matrix}$$

(b) z = xW + b

$$z = \begin{matrix} 0.5 & 2.0 \end{matrix} \times \begin{matrix} -0.1 & 0.5 & 0 \\ -0.3 & 0.4 & 0.6 \end{matrix} + \begin{matrix} 0.2 & 0.2 & 0 \end{matrix} = \begin{matrix} -0.45 & 1.25 & 1.2 \end{matrix}$$

$$h = g(z) = \frac{1}{1+e^{-z}} = \begin{matrix} 0.39 & 0.78 & 0.77 \end{matrix}$$

u = hV + c

$$u = \begin{matrix} 0.39 & 0.78 & 0.77 \end{matrix} \times \begin{matrix} 5.0 & 0 \\ 2.0 & -4.0 \\ 3.0 & 6.0 \end{matrix} + \begin{matrix} 0.2 & 0.2 \end{matrix} = \begin{matrix} 6.02 & 1.7 \end{matrix}$$

$$y = f(u) = \frac{e^{u}}{\sum_{k=1}^{K} e^{U_k}} = \begin{matrix} 0.99 & 0.01 \end{matrix}$$

(c) 99% probability x belongs to class 1
1% probability x belongs to class 2

(d) Classification error = 1
Cross entropy loss = $-\sum_{p=1}^{P} \log\left(f\left(u_p d_p\right)\right) = -(\log 0.01)$ = 2

(e)

$$\nabla_u J = -\left(k - f(u)\right) = -(\begin{matrix} 0 & 1 \end{matrix} - \begin{matrix} 0.99 & 0.01 \end{matrix}) = \begin{matrix} 0.99 & -0.99 \end{matrix}$$

$$g'(z) = h \times (1-h) = \begin{matrix} 0.39 & 0.78 & 0.77 \end{matrix} \times \begin{matrix} 0.61 & 0.22 & 0.23 \end{matrix} = \begin{matrix} 0.24 & 0.17 & 0.18 \end{matrix}$$

$$\nabla_z J = (\nabla_u J)V^T \times g'(z) = \begin{matrix} 0.99 & -0.99 \end{matrix} \times \begin{matrix} 5 & 2 & 3 \\ 0 & -4 & 6 \end{matrix} \times \begin{matrix} 0.24 & 0.17 & 0.18 \end{matrix}$$
$$= \begin{matrix} 1.19 & 1.01 & -0.53 \end{matrix}$$

(f)

$$\nabla_V J = 0.4 \times \begin{matrix} 0.39 \\ 0.78 \\ 0.77 \end{matrix} \times \begin{matrix} 0.99 & -0.99 \end{matrix} = \begin{matrix} 0.16 & -0.16 \\ 0.31 & -0.31 \\ 0.31 & -0.31 \end{matrix}$$

$$\nabla_c J = 0.4 \times \begin{matrix} 0.99 \\ -0.99 \end{matrix} = \begin{matrix} 0.4 \\ -0.4 \end{matrix}$$

$$\nabla_W J = 0.4 \times \begin{matrix} 0.5 \\ 2.0 \end{matrix} \times \begin{matrix} 1.19 & 1.01 & -0.53 \end{matrix} = \begin{matrix} 0.24 & 0.2 & -0.11 \\ 0.95 & 0.81 & -0.42 \end{matrix}$$

$$\nabla_b J = 0.4 \times 1.19 \quad 1.01 \quad -0.53 \;=\; \begin{matrix} 0.48 \\ 0.4 \\ -0.21 \end{matrix}$$

(g) Updated:

$$V = \begin{matrix} 5.0 & 0 \\ 2.0 & -4.0 \\ 3.0 & 6.0 \end{matrix} \quad - \quad \begin{matrix} 0.16 & -0.16 \\ 0.31 & -0.31 \\ 0.31 & -0.31 \end{matrix} \quad = \quad \begin{matrix} 4.84 & 0.16 \\ 1.69 & -3.69 \\ 2.69 & 6.31 \end{matrix}$$

$$c = \begin{matrix} 0.2 \\ 0.2 \end{matrix} - \begin{matrix} 0.4 \\ -0.4 \end{matrix} = \begin{matrix} -0.2 \\ 0.6 \end{matrix}$$

$$W = \begin{matrix} -0.1 & 0.5 & 0 \\ -0.3 & 0.4 & 0.6 \end{matrix} \quad - \quad \begin{matrix} 0.24 & 0.2 & -0.11 \\ 0.95 & 0.81 & -0.42 \end{matrix} \quad = \quad \begin{matrix} 0.14 & 0.3 & 0.11 \\ -1.25 & -0.41 & 1.02 \end{matrix}$$

$$b = \begin{matrix} 0.2 \\ 0.2 \\ 0 \end{matrix} - \begin{matrix} 0.48 \\ 0.4 \\ -0.21 \end{matrix} = \begin{matrix} -0.28 \\ -0.2 \\ 0.21 \end{matrix}$$

3 **(a)** **(i)** Output = [(N – F + 2P) / S] + 1 = (225 – 3) / 2 + 1 = 112

(ii) 3 x 3 x 3 + 1 = 28
28 x 128 = 3584

(iii) 6 x 128 x 128

(b) **(i)** A 1x1 convolution simply maps an input pixel with all it's channels to an output pixel, not looking at anything around itself. It is often **used to reduce the number of depth channels**, since it is often very slow to multiply volumes with extremely large depths.

(ii) F-1

(iii)

1. Select Source Task - You must select a related predictive modeling problem with an abundance of data where there is some relationship in the input data, output data, and/or concepts learned during the mapping from input to output data.
2. Develop Source Model - Next, you must develop a skillful model for this first task. The model must be better than a naive model to ensure that some feature learning has been performed. Augmentations can be performed on the dataset to artificially increase its size.
3. Reuse Model - The model fit on the source task can then be used as the starting point for a model on the second task of interest. This may involve using all or parts of the model, depending on the modeling technique used.
4. Tune Model - Optionally, the model may need to be adapted or refined on the input-output pair data available for the task of interest.

(c) **(i)** $H = \varphi(XW + B)$

$$= \varphi \ \text{x} \ \begin{matrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{matrix} \ \text{x} \ \begin{matrix} 0.2 & -0.2 \\ 0 & 0.5 \\ 0.5 & 0.5 \end{matrix} \ + \ \begin{matrix} 0.5 \\ -0.5 \end{matrix} \ = \ \begin{matrix} -0.348 \\ 0.876 \end{matrix}$$

(ii) $\text{H}_1 = \varphi(\text{W}^t\text{x}_1 + \text{b}) = \varphi(\begin{matrix} 0.2 & 0 & 0.5 \\ -0.2 & 0.5 & 0.5 \end{matrix} * \begin{matrix} 1 \\ 0 \\ 1 \end{matrix} + \begin{matrix} 0.5 \\ -0.5 \end{matrix}) = \begin{matrix} 0.7685 \\ 0.4501 \end{matrix}$

$\text{Y}_1 = \varphi(\text{Wh} + \text{c})$

$$= \varphi(\begin{matrix} 0.2 & -0.2 \\ 0.0 & 0.5 \\ 0.5 & 0.5 \end{matrix} * \begin{matrix} 0.7685 \\ 0.4501 \end{matrix} + \begin{matrix} -0.5 \\ 0.2 \\ -0.5 \end{matrix}) = \begin{matrix} -0.43632 \\ 0.42505 \\ 0.1093 \end{matrix}$$

Do the same steps as above by replacing x_2 in place of x_1

(iii) Sparse autoencoders are autoencoders with less hidden neurons typically used to learn features for another task such as classification. An autoencoder that has been regularized to be sparse must respond to unique statistical features of the dataset it has been trained on, rather than simply acting as an identity function. In this way, training to perform the copying task with a sparsity penalty can yield a model that has learned useful features as a byproduct.

4 **(a)** h(t) = Tanh $(U^T x(t) + W^T y(t-1) + b)$

y(t) = $\sigma(V^T h(t) + c$

$h^{\rightarrow}$ (1) = $Tanh\ (\begin{matrix} 0.4 & 0.5 \\ 0.1 & 0.4 \end{matrix} \times \begin{matrix} 1 \\ 0.5 \end{matrix} + \begin{matrix} 1 \\ 0.5 \end{matrix} \times 2.0 + \begin{matrix} 0.1 \\ 0.1 \end{matrix}) = \begin{matrix} 1.76 \\ 0.59 \end{matrix}$

$y(1) = \sigma\Big(0.3 \quad -0.3 \times \begin{matrix} 0.56 \\ 0.44 \end{matrix} + \begin{matrix} 0.1 \\ 0.1 \end{matrix}\Big) = \begin{matrix} 0.56 \\ 0.44 \end{matrix}$

Do the same steps as above for the remaining hidden states and outputs but remember to keep substituting $y(t-1)$ with the output you got in the previous step.

(b) **(i)** TRUE

(ii) TRUE

(iii) FALSE

(iv) FALSE

(v) TRUE

(vi) FALSE

(c) These mechanisms allow **a task to focus on a set of elements of an input sequence**, an intermediate sequence or a memory source.

(d) To overcome this method, generate images and classify images in batches. The discriminator network should have extra layers or parameters to compute whether the batch is ‘diverse’ enough or not. Hence, if mode collapse occurs and the images generated aren’t diverse enough, the discriminator will classify the images as fake. This forces the generator to generate diverse images

Solver: Asuri Simhakutty Kamakshi (kamakshi001@e.ntu.edu.sg)