Due Date: October 19th, 2023 at 11:00 pm

Instructions

- *For all questions, show your work!*
- *Use LaTeX and the template we provide when writing your answers. You may reuse most of the notation shorthands, equations and/or tables. See the assignment policy on the course website for more details.*
- *The use of AI tools like Chat-GPT to find answers or parts of answers for any question in this assignment is not allowed. However, you can use these tools to improve the quality of your writing, like fixing grammar or making it more understandable. If you do use these tools, you must clearly explain how you used them and which questions or parts of questions you applied them to. Failing to do so or using these tools to find answers or parts of answers may result in your work being completely rejected, which means you'll receive a score of 0 for the entire theory or practical section.*
- *Submit your answers electronically via Gradescope.*
- *TAs for this assignment are* ***Saba Ahmadi, Sahar Dastani, and Sophie Xhonneux.***

Question 1 (5-2-2-1-3-2)**.** (**Activation Functions and Backpropagation**)

Please note that during marking we will not read more than the question says you should write, e.g. if the question asks for one sentence, only the first sentence will be read.

1. Given the following 3 layer neural network $\boldsymbol{o} = f(\boldsymbol{x})$

$$\begin{aligned} \boldsymbol{h} &= \boldsymbol{W}_1\boldsymbol{x} + \boldsymbol{b}_1 \\ \boldsymbol{a} &= \sigma(\boldsymbol{W}_2\boldsymbol{h} + \boldsymbol{b}_2) \\ \boldsymbol{o} &= \boldsymbol{W}_3\boldsymbol{a} + \boldsymbol{b}_3, \end{aligned} \tag{1}$$

   where $\boldsymbol{x} \in \mathrm{R}^n$, $\boldsymbol{h} \in \mathrm{R}^m$, $\boldsymbol{a} \in \mathrm{R}^r$, $\boldsymbol{o} \in \mathrm{R}^s$, $\sigma(x) = \frac{1}{1+e^{-x}}$, and a loss function $\mathrm{L}(\boldsymbol{o}, \boldsymbol{y})$ apply the back-propagation algorithm computing the gradients with respect to the weights $\boldsymbol{W}_1, \boldsymbol{W}_2, \boldsymbol{W}_3, \boldsymbol{b}_1, \boldsymbol{b}_2, \boldsymbol{b}_3$ to show which are the necessary intermediate vectors that we need to store during the forward pass such as to not need recomputation. Use the notation provided above.
2. Assuming each floating point number takes two bytes to store, give the amount of memory in bytes, needed to store during the forward pass for the backward pass to not need recomputation.
3. Compute the left and right derivative of the `ReLU` function at $x = 0$. Show your work.
4. Give **one** sentence to explain why we cannot use the `ReLU` function if we want to differentiate our neural network twice.
5. The swish activation function is defined as $g(x) = x\sigma(\beta x)$. Simplify the function in the limit as β approaches 0 and infinity. Show your work.
6. Give 1 reason (one sentence) as to why we might want to use the swish function over the `ReLU`. Give 1 reason (one sentence) as to why we might want to use the `ReLU` over the swish (hint: monotonicity).

Answer 1. 1. Let $\boldsymbol{a}'$ be the pre-activation $\boldsymbol{W}_2\boldsymbol{h} + \boldsymbol{b}_2$. The gradients can analytically be written down as:

$$\frac{\partial L(y, \boldsymbol{o})}{\partial (\boldsymbol{W}_3)_{ij}} = \frac{\partial L(y, \boldsymbol{o})}{\partial \boldsymbol{o}} \frac{\partial \boldsymbol{o}}{\partial (\boldsymbol{W}_3)_{ij}} \tag{2}$$

$$\left[\frac{\partial \boldsymbol{o}}{\partial (\boldsymbol{W}_3)_{ij}}\right]_k = \boldsymbol{a}_k \tag{3}$$

$$\tag{4}$$

$$\frac{\partial L(y, \boldsymbol{o})}{\partial (\boldsymbol{b}_3)_{ij}} = \frac{\partial L(y, \boldsymbol{o})}{\partial \boldsymbol{o}} \frac{\partial \boldsymbol{o}}{\partial (\boldsymbol{b}_3)_{ij}} \tag{5}$$

$$= \frac{\partial L(y, \boldsymbol{o})}{\partial \boldsymbol{o}} \tag{6}$$

$$\tag{7}$$

$$\frac{\partial L(y, \boldsymbol{o})}{\partial (\boldsymbol{W}_2)_{ij}} = \frac{\partial L(y, \boldsymbol{o})}{\partial \boldsymbol{o}} \frac{\partial \boldsymbol{o}}{\partial (\boldsymbol{W}_2)_{ij}} \tag{8}$$

$$\left[\frac{\partial \boldsymbol{o}}{\partial (\boldsymbol{W}_2)_{ij}}\right]_k = \sum_l (\boldsymbol{W}_3)_{kl} \frac{\partial \boldsymbol{a}_l}{\partial (\boldsymbol{W}_2)_{ij}} \tag{9}$$

$$= \sum_l (\boldsymbol{W}_3)_{kl} \left(\boldsymbol{a}(1 - \boldsymbol{a})\right)_l \frac{\partial \sum_m (\boldsymbol{W}_2)_{lm} \boldsymbol{h}_m}{\partial (\boldsymbol{W}_2)_{ij}} \tag{10}$$

$$= (\boldsymbol{W}_3)_{ki} \left(\boldsymbol{a}(1 - \boldsymbol{a})\right)_i \boldsymbol{h}_j \tag{11}$$

$$\tag{12}$$

$$\frac{\partial L(y, \boldsymbol{o})}{\partial (\boldsymbol{b}_2)_j} = \frac{\partial L(y, \boldsymbol{o})}{\partial \boldsymbol{o}} \frac{\partial \boldsymbol{o}}{\partial (\boldsymbol{b}_2)_j} \tag{13}$$

$$\left[\frac{\partial \boldsymbol{o}}{\partial (\boldsymbol{b}_2)_j}\right]_k = \sum_l (\boldsymbol{W}_3)_{kl} \frac{\partial \boldsymbol{a}_l}{\partial (\boldsymbol{b}_2)_j} \tag{14}$$

$$= (\boldsymbol{W}_3)_{kj} \left(\boldsymbol{a}(1 - \boldsymbol{a})\right)_j \tag{15}$$

$$\tag{16}$$

$$\frac{\partial L(y,\boldsymbol{o})}{\partial (\boldsymbol{W}_1)_{ij}} = \frac{\partial L(y,\boldsymbol{o})}{\partial \boldsymbol{o}} \frac{\partial \boldsymbol{o}}{\partial (\boldsymbol{W}_1)_{ij}} \tag{17}$$

$$\left[\frac{\partial \boldsymbol{o}}{\partial (\boldsymbol{W}_1)_{ij}}\right]_k = \sum_l (\boldsymbol{W}_3)_{kl} \frac{\partial \boldsymbol{a}_l}{\partial (\boldsymbol{W}_1)_{ij}} \tag{18}$$

$$= \sum_l (\boldsymbol{W}_3)_{kl} \left.\frac{\partial \sigma(z)}{\partial z}\right|_{z=\boldsymbol{a}'_l} \frac{\partial \boldsymbol{a}'_l}{\partial (\boldsymbol{W}_1)_{ij}} \tag{19}$$

$$= \sum_l (\boldsymbol{W}_3)_{kl} \left.\frac{\partial \sigma(z)}{\partial z}\right|_{z=\boldsymbol{a}'_l} \sum_m (\boldsymbol{W}_2)_{lm} \frac{\partial \boldsymbol{h}_m}{\partial (\boldsymbol{W}_1)_{ij}} \tag{20}$$

$$= \sum_l (\boldsymbol{W}_3)_{kl} \left.\frac{\partial \sigma(z)}{\partial z}\right|_{z=\boldsymbol{a}'_l} \sum_m (\boldsymbol{W}_2)_{lm} \sum_n \frac{\partial (\boldsymbol{W}_1)_{mn} \boldsymbol{x}_n}{\partial (\boldsymbol{W}_1)_{ij}} \tag{21}$$

$$= \sum_l (\boldsymbol{W}_3)_{kl} \left(\boldsymbol{a}(1-\boldsymbol{a})\right)_l (\boldsymbol{W}_2)_{li} \boldsymbol{x}_j \tag{22}$$

$$\tag{23}$$

$$\frac{\partial L(y,\boldsymbol{o})}{\partial (\boldsymbol{b}_1)_j} = \frac{\partial L(y,\boldsymbol{o})}{\partial \boldsymbol{o}} \frac{\partial \boldsymbol{o}}{\partial (\boldsymbol{b}_1)_j} \tag{24}$$

$$\left[\frac{\partial \boldsymbol{o}}{\partial (\boldsymbol{b}_1)_j}\right]_k = \sum_l (\boldsymbol{W}_3)_{kl} \frac{\partial \boldsymbol{a}_l}{\partial (\boldsymbol{b}_1)_j} \tag{25}$$

$$= \sum_l (\boldsymbol{W}_3)_{kl} \left.\frac{\partial \sigma(z)}{\partial z}\right|_{z=\boldsymbol{a}'_l} \frac{\partial \boldsymbol{a}'_l}{\partial (\boldsymbol{b}_1)_j} \tag{26}$$

$$= \sum_l (\boldsymbol{W}_3)_{kl} \left.\frac{\partial \sigma(z)}{\partial z}\right|_{z=\boldsymbol{a}'_l} \sum_m (\boldsymbol{W}_2)_{lm} \frac{\partial \boldsymbol{h}_m}{\partial (\boldsymbol{b}_1)_j} \tag{27}$$

$$= \sum_l (\boldsymbol{W}_3)_{kl} \left.\frac{\partial \sigma(z)}{\partial z}\right|_{z=\boldsymbol{a}'_l} \sum_m (\boldsymbol{W}_2)_{lm} \frac{\partial \boldsymbol{b}_m}{\partial (\boldsymbol{b}_1)_j} \tag{28}$$

$$= \sum_l (\boldsymbol{W}_3)_{kl} \left(\boldsymbol{a}(1-\boldsymbol{a})\right)_l (\boldsymbol{W}_2)_{lj} \tag{29}$$

$$\tag{30}$$

Thus, during the **backpropagation algorithm** we only need access to $\boldsymbol{a}$ and $\boldsymbol{o}$. We need those two vectors to compute $\left.\frac{\partial \sigma(z)}{\partial z}\right|_{z=\boldsymbol{a}_j}$ and $\frac{\partial L(y,\boldsymbol{o})}{\partial \boldsymbol{o}}$, respectively.

2. Based on the previous question the answer is $2(r+s)$ bytes. Note that $\boldsymbol{x}$ is the input and is thus not included in the memory calculation.
3. $\lim_{h^-\to 0} \frac{\text{ReLU}(0+h)-\text{ReLU}(0)}{h} = \lim_{h^-\to 0} \frac{0-0}{h} = 0$ $\lim_{h^+\to 0} \frac{\text{ReLU}(0+h)-\text{ReLU}(0)}{h} = \lim_{h^+\to 0} \frac{h-0}{h} = 1$
4. The second derivative of the `ReLU` is 0 everywhere and thus destroys all information during backpropagation.
5. - For $\beta \to 0$: $g(x) = x\sigma(0x) = \frac{x}{2}$
   - For $\beta \to \infty$ and $x \leq 0$: $\lim_{\beta\to\infty} g(x) = \lim_{\beta\to\infty} x\left(\frac{1}{1+e^{-\beta x}}\right) = x\left(\frac{1}{1+e^{-\infty x}}\right) = x\left(\frac{1}{1+\infty}\right) = 0$

- For $\beta \to \infty$ and $x > 0$: $\lim_{\beta\to\infty} g(x) = \lim_{\beta\to\infty} x\left(\frac{1}{1+e^{-\beta x}}\right) = x\left(\frac{1}{1+e^{-\infty x}}\right) = x\left(\frac{1}{1+0}\right) = x$

Thus for $g(x)$ is an interpolation between a linear function and the `ReLU`.

6. Possible answers for why to choose Swish over ReLU:
   - Since the ReLU second derivative destroys all information, if we want to differentiate twice, we should use the Swish
   - The Swish propagates gradients for when pre-activations are negative, thus preventing dead neurons that the ReLU suffers from.

   Possible answers for why to choose ReLU over Swish:
   - The Swish is non-monotonic, this means that for some negative activations they might become more negative when the gradient tells them to move in the opposite direction.
   - The ReLU is much faster to compute, which matters for large neural networks.

Question 2 (4-4-5-3)**.** (**Convolution Basics**)

1. **Short answers:**
   (a) When you apply a filter to feature maps, under what conditions does the size of the feature map decrease ?
   (b) What impact does a larger stride have ?
   (c) Explain what you understand from the term "sparse connections" in terms of connections between two layers in a neural network.
   (d) As we are aware, each *Convolutional* layer contains learnable parameters, including weights and biases. If you decide not to include biases in the *Convolution* layer of the following network, what will be the impact on its performance (compared to the case when the *Convolution* layer includes biases) ?

$$Input \to Convolution \to BatchNorm \to Activation \to Output$$

2. **True/False:**
   (a) In a convolutional layer, the number of weights is contingent upon the depth of the input data volume.
   (b) In a convolutional layer, the number of biases equals the number of filters.
   (c) The total number of parameters is influenced by the stride and padding.
   (d) Maxpooling operates only on the height and width dimensions of an image.
3. **Convolutional Architecture:** Let's analyze the convolutional neural network (CNN) defined by the layers in the left column. For each layer, determine the output volume's shape and the number of parameters. It's worth noting that in our notation, "Conv4-N-Pvalid-S2" represents a convolutional layer with N 4x4 filters, valid padding, and a stride of 2.

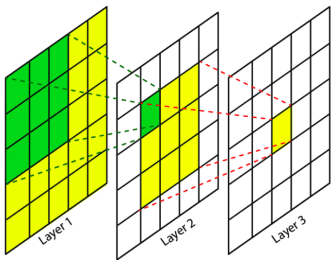

FIGURE 1 – Illustration of Receptive Field.

Layers	Output volume dimensions	Number of parameters
Input	$64 \times 64 \times 1$	
Conv4-5-Pvalid-S2		
Pool3		
Conv3-5-Pvalid-S1		
Pool2		
FC5		

4. **Receptive Fields:** Receptive Field is defined as the size of the region in the input that produces a feature. For example, when using a Conv-3x3 layer, each feature is generated by a 3x3 grid in the input. Having two Conv-3x3 layers in succession leads to a 5x5 receptive field. An example of this is outlined in Figure 1. Note that there is a receptive field associated with each feature (Figure 1 shows the receptive field associated with the yellow-colored feature in Layer 3), where a feature is a cell in the output activation map.

   Consider a network architecture consisting solely of 5 Convolution layers, each with kernel size 5, stride 1, zero-padding of two pixels on all four sides, 8 intermediate channels, and one output channel. What is the receptive field of the final features obtained through this architecture, that is, what is the size of the input that corresponds to a particular pixel position in the output ?

Answer 2. 1. **Short answers:**

(a) With zero padding, Only if the size of the filter is greater than 1.

(b) It will produce a smaller feature map.

(c) Each output value depends on a small number of inputs, instead of taking into account all the inputs.

(d) The performance wouldn't be affected. The positioning of the batch norm, immediately after the conv, ensures that the BN layer can learn any bias previously learned by the conv layer.

2. **True/False:**

(a) T

(b) T

(c) F

(d) T

3. **Convolutional Architecture:**

Layers	Activation volume dimensions	Number of parameters
Input	$64 \times 64 \times 1$	0
Conv4-5-Pvalid-S2	$31 \times 31 \times 5$	$(4 \times 4 \times 1 + 1) \times 5 = 85$
Pool3	$10 \times 10 \times 5$	0
Conv3-5-Pvalid-S1	$8 \times 8 \times 5$	$(3 \times 3 \times 5 + 1) \times 5 = 230$
Pool21	$4 \times 4 \times 5$	0
FC5	5×1	$(2 \times 2 \times 5 + 1) \times 5 = 405$

4. **Receptive Fields:** The receptive field at layer k with filter size of F_j and stride of S_i can be computed with the following formula:

$$R_k = 1 + \Sigma_{j=1}^{k}(F_j - 1)\Pi_{i=0}^{j-1} S_i$$

Therefore we have:

$$R_4 = 1 + (5-1) \times 1 + (5-1) \times 1 + (5-1) \times 1 + (5-1) \times 1 + (5-1) \times 1 = 21$$

As a result the input size that corresponds to a particular pixel position in the output is $= (21, 21)$.

Question 3 (10). (**Mixture Models meet Neural Networks**)

Consider modelling some data $\{(\boldsymbol{x}_n, y_n)\}_{n=1}^{N}$, $\boldsymbol{x}_n \in \mathbb{R}^d$, $y_n \in \{0, 1\}$, using a mixture of logistic regression models, where we model each binary label y_n by first picking one of the K logistic regression models, based on the value of a latent variable $z_n \sim$ Categorical$(\pi_1, ..., \pi_K)$ and then generating y_n *conditioned* on z_n as $y_n \sim$ Bernoulli$[\sigma(\boldsymbol{w}_{z_n}^T \boldsymbol{x}_n)]$, where $\sigma(\cdot)$ is the sigmoid activation function.

Now consider the *marginal* probability of the label $y_n = 1$, given $\boldsymbol{x}_n$, i.e., $p(y_n = 1|\boldsymbol{x}_n)$, and show that this quantity can also be thought of as the output of a neural network. Clearly specify what is the input layer, the hidden layer(s), activations, the output layer, and the connection weights of this neural network.

Answer 3. You can find the answer in the final page.

Question 4 (10). (**Cross Entropy**)

Cross-entropy loss function (a popular loss function) is given by:

$$ce(p, x) = -x \log(p) - (1 - x) \log(1 - p)$$

This derivation assumes that x is binary, i.e. $x \in \{0, 1\}$. However, the same loss function is often also used with real-valued $x \in (0, 1)$.

1. Derive the cross-entropy loss function using the maximum likelihood principle for $x \in \{0, 1\}$.
2. Suggest a probabilistic interpretation of the cross-entropy loss function when $x \in (0, 1)$. (HINT: KL divergence between two distributions)

Answer 4. 1. For a Bernoulli distribution,

$$P(X = x; p) = p^x (1 - p)^{(1-x)}$$

Thus,

$$NLL = -\log P(X = x; p) = -x \log p - (1 - x) \log(1 - p)$$

2. If we interpret the real-valued $x \in (0, 1)$ as parameters of a given Bernoulli distribution over the binary variable b, then cross-entropy objective can be interpreted as the KL divergence between the empirical and model distributions.

$$KL(P\|Q) = \sum_b P(b) \log \frac{P(b)}{Q(b)}$$

$$= P(b = 1) \log \frac{P(b = 1)}{Q(b = 1)} + P(b = 0) \log \frac{P(b = 0)}{Q(0)}$$

$$= x \log \frac{x}{p} + (1 - x) \log \frac{1 - x}{1 - p}$$

$$= -x \log(p) - (1 - x) \log(1 - p) - x \log(x) - (1 - x) \log(1 - x)$$

$$= ce(p, x) + C$$

Question 5 (1-4-2)**.** (**Optimization**)

Suppose we want to minimize $f(x, y) = y + (y - x)^2$.

1. Find the true minimum.
2. For each step size given below, compute:
   (a) The value of (x1, y1) using the full gradient descent (not stochastic) at the starting point (x0, y0) = (1, 1).
   (b) The L2 distance of (x1, y1) from the true minimum.
   Please note that the (x1, y1) is (x, y) coordinate after 1 step of gradient descent.
      i. Step size $s = 0.01$.
      ii. Step size $s = 0.1$.
      iii. Step size $s = 10$.
3. What are the implications of using different step sizes in gradient descent, and what strategies can be employed to effectively address these implications ?

Answer 5. 1- We find partial derivatives and set them to zero to find the critical points:

$$\frac{\partial f}{\partial x} = -2y + 2x \rightarrow -2y + 2x = 0 \rightarrow x = y$$
$$\frac{\partial f}{\partial y} = 4y - 2x \rightarrow 4y = 2x \rightarrow x = 2y$$
$$\rightarrow x = 0, y = 0$$

We use the second derivative to make sure this critical point corresponds to the true minimum:

$$\frac{\partial^2 f}{\partial x^2} = \frac{\partial}{\partial x}(-2(y-x)) = -2 \cdot (-1) = 2$$
$$\frac{\partial^2 f}{\partial y^2} = \frac{\partial}{\partial y}(2y + 2(y-x)) = 2 + 2 = 4$$
$$\frac{\partial^2 f}{\partial x \partial y} = \frac{\partial}{\partial x}(2y + 2(y-x)) = -2$$

We find the Hessian matrix H:

$$H(f) = \begin{pmatrix} \frac{\partial^2 f}{\partial x^2} & \frac{\partial^2 f}{\partial x \partial y} \\ \frac{\partial^2 f}{\partial y \partial x} & \frac{\partial^2 f}{\partial y^2} \end{pmatrix}$$

Determinant of the Hessian matrix at our critical point $(0, 0)$ is:

$$\det(H) = \left(\frac{\partial^2 f}{\partial x^2}\right)\left(\frac{\partial^2 f}{\partial y^2}\right) - \left(\frac{\partial^2 f}{\partial x \partial y}\right)^2 = (2)(4) - (-2)^2 = 8 - 4 = 4$$

Since $D > 0$ and $\frac{\partial^2 f}{\partial x^2} > 0$, the critical point $(0, 0)$ corresponds to a local minimum. Therefore, the true minimum of $f(x, y)$ occurs at $(0, 0)$ with a minimum value of $f(0, 0) = 0$.

Many of you forgot to show why the critical point you found is the true minimum. We recommend students who have problem with this part to read this for a quick refresher.

2- First we compute the gradient:

$$\nabla f(x, y) = [2x - 2y, 4y - 2x] \Longrightarrow \nabla f\,(x_0, y_0) = [0, 2]$$

(a) We first compute the update term:
$[x_1, y_1] = [x_0, y_0] - s\nabla f\,(x_0, y_0) = [1, 1] - s[0, 2] = [1, 1 - 2s]$. So:
i. Step size $s = 0.01 \Longrightarrow [x_1, y_1] = [1, 0.98]$
ii. Step size $s = 0.1 \Longrightarrow [x_1, y_1] = [1, 0.8]$
iii. Step size $s = 10 \Longrightarrow [x_1, y_1] = [1, -19]$
(b) Given that minimum occurs at $(0, 0)$, the $L2$-norm would be $\|[x_1, y_1]\|_2$.
i. Step size $s = 0.01 \Longrightarrow \|[1, 0.98]\|_2 \approx 1.40$

ii. Step size $s = 0.1 \implies \|[1, 0.8]\|_2 \approx 1.28$
iii. Step size $s = 10 \implies \|[1, -19]\|_2 \approx 19.03$

3- A step size that is too small can lead to slow convergence, while a very large step size can result in overshooting. To strike a balance between a good convergence rate and avoiding overshooting in gradient descent, several strategies can be employed: learning rate tuning, learning rate schedules, adaptive methods and momentum.

$$\begin{aligned} p(y_n = 1|\boldsymbol{x}_n) &= \sum_{k=1}^{K} p(y_n = 1|z_n = k, \boldsymbol{x}_n) p(z_n = k) \\ &= \sum_{k=1}^{K} \sigma(\boldsymbol{w}_k^T \boldsymbol{x}_n)\pi_k \\ &= \sum_{k=1}^{K} \frac{\pi_k}{1 + e^{-\boldsymbol{w}_k^T \boldsymbol{x}_n}} \end{aligned}$$

Neural Network Architecture

Consider an input $\boldsymbol{x}$ to the Neural Network. We maintain the hidden layer to have size K. We also maintain all bias terms as 0 in the Neural Network.

Input Layer	$x_1\ x_2 \ldots x_D$
Hidden Layer 1	$\sigma(\boldsymbol{w}_1^T \boldsymbol{x}) \ldots \sigma(\boldsymbol{w}_K^T \boldsymbol{x})$
Output Layer	$\sum_{k=1}^{K} \frac{\pi_k}{\sigma(\boldsymbol{w}_k^T \boldsymbol{x})}$

TABLE 1 – Equivalent Neural Network Architecture

Note that for this equivalence, we define the following parameters of neural network:

- $W^{(1)}$ is the matrix of weights connecting the input layer to the first hidden layer. It has the following structure:

$$W^{(1)}_{D\times K} = \begin{bmatrix} . & . & . & . & . & . & . \\ . & . & . & . & . & . & . \\ \boldsymbol{w}_1 & \boldsymbol{w}_2 & - & - & - & \boldsymbol{w}_{K-1} & \boldsymbol{w}_K \\ . & . & . & . & . & . & . \\ . & . & . & . & . & . & . \end{bmatrix}$$

  This gives us the values at hidden layer $\boldsymbol{a}^{(1)}$ for input $\boldsymbol{x}_n$ without the activation as:

$$\begin{aligned} \boldsymbol{a}_n^{(1)} &= {W^{(1)}}^T \boldsymbol{x}_n \\ &= \begin{bmatrix} \boldsymbol{w}_1^T \boldsymbol{x}_n & \boldsymbol{w}_2^T \boldsymbol{x}_n & . & . & . & \boldsymbol{w}_{K-1}^T \boldsymbol{x}_n & \boldsymbol{w}_K^T \boldsymbol{x}_n \end{bmatrix}^T \end{aligned}$$

- Next, we apply the **sigmoid** non-linearity on the hidden layer. Then, we get the final value $\boldsymbol{z}^{(1)}$ at the first hidden layer as the following:

$$\begin{aligned} \boldsymbol{z}_n^{(1)} &= \sigma(\boldsymbol{a}_n^{(1)}) \\ &= \begin{bmatrix} \sigma(\boldsymbol{w}_1^T \boldsymbol{x}_n) & \sigma(\boldsymbol{w}_2^T \boldsymbol{x}_n) & . & . & . & \sigma(\boldsymbol{w}_{K-1}^T \boldsymbol{x}_n) & \sigma(\boldsymbol{w}_K^T \boldsymbol{x}_n) \end{bmatrix}^T \end{aligned}$$

- $W^{(2)}$ is the matrix of weights connecting the hidden layer to the output node. It has the following structure:

$$W^{(2)} = \begin{bmatrix} \pi_1 & \pi_2 & . & . & . & \pi_{K-1} & \pi_K \end{bmatrix}^T$$

  We do not apply any activation on the output layer.

- This gives us the value at the output layer as follows:

$$\begin{aligned} \text{out} &= {W^{(2)}}^T \boldsymbol{z}_n^{(1)} \\ &= \sum_{k=1}^{K} \pi_k \sigma(\boldsymbol{w}_k^T \boldsymbol{x}_n) \end{aligned}$$

This shows us that the above designed Neural Network gives us the required output exactly. Hence the marginal required can also be thought of as the output of a Neural Network with the above structure.