Deep Learning (IST, 2022-23)

Homework 1

Group 3

Martim Santos 95638, Marina Gomes 95637

This homework was divided equally amongst the members of the group so each member contributed the same. We started by dividing the first paragraph of the first Question with the purpose of both members getting familiar with the given data and code. Martim did the Perceptron and Marina did the Logistic Regression. After that, we decided to solve the rest of the homework together. Each time one of the members had an idea for any question, that idea was discussed and materialized if it made sense. As a result of that, both members have total knowledge about what was done and are able to explain every detail since both participated in every task.

Question 1

Question 1.1

Question 1.1a

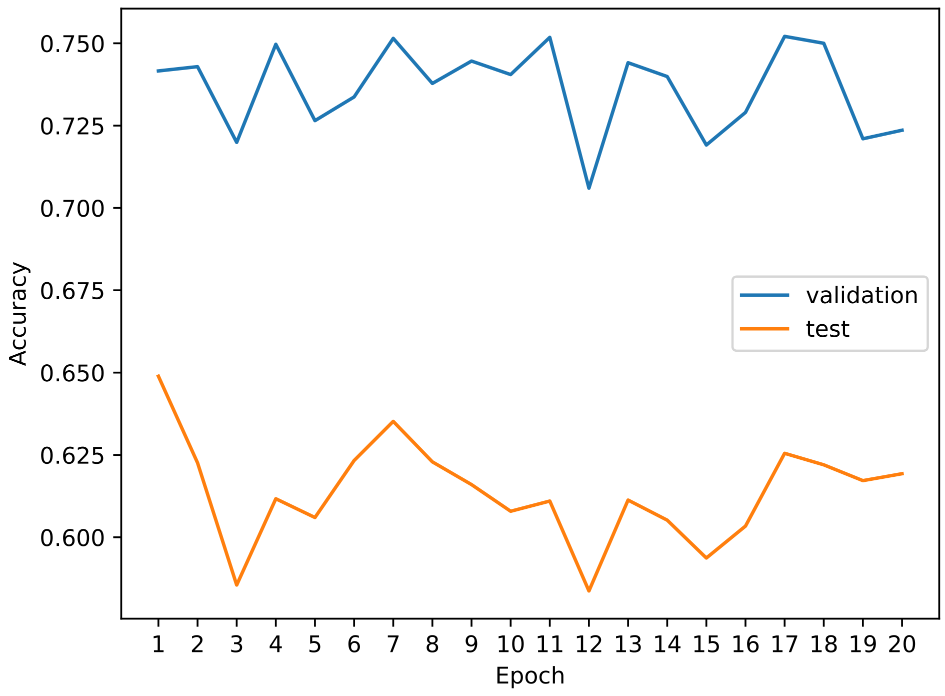

Figure 1: Perceptron − Accuracies as a function of the Epoch number.

Question 1.1b

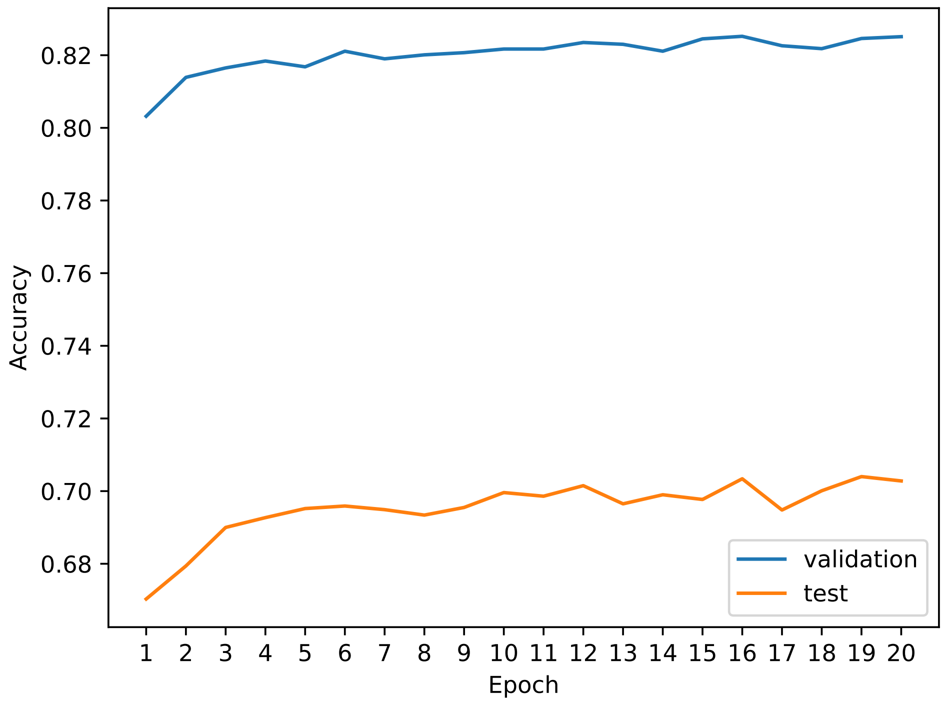

Figure 2: Logistic Regression – Accuracies as a function of the Epoch number.

Question 1.2

Question 1.2a

A multi-layer perceptron (MLP) is a type of neural network that is composed of multiple layers of artificial neurons, or ”perceptrons”. MLPs are able to learn complex relationships in data because they can use multiple layers of perceptrons to build up increasingly complex representations of the input data.

One of the key differences between a perceptron and an MLP is that the MLP can use non-linear activation functions, whereas the simple perceptron can only use a linear activation function. This use of non-linear activation functions in an MLP allows it to overcome some of the limitations of the perceptron, such as its inability to learn non-linear relationships in the data. Amongst others, one known problem that cannot be solved using a perceptron but can be solved using a MLP is the XOR problem. This occurs due to the fact that this problem is not linearly separable. The MLP is therefore more expressive, or more capable of representing a wider range of relationships in the data.

If the activation function of an MLP is linear, it would have the same limitations as a simple perceptron, and would only be able to capture linear relationships in the data. In this case, the MLP would not be more expressive than the simple perceptron.

Question 1.2b

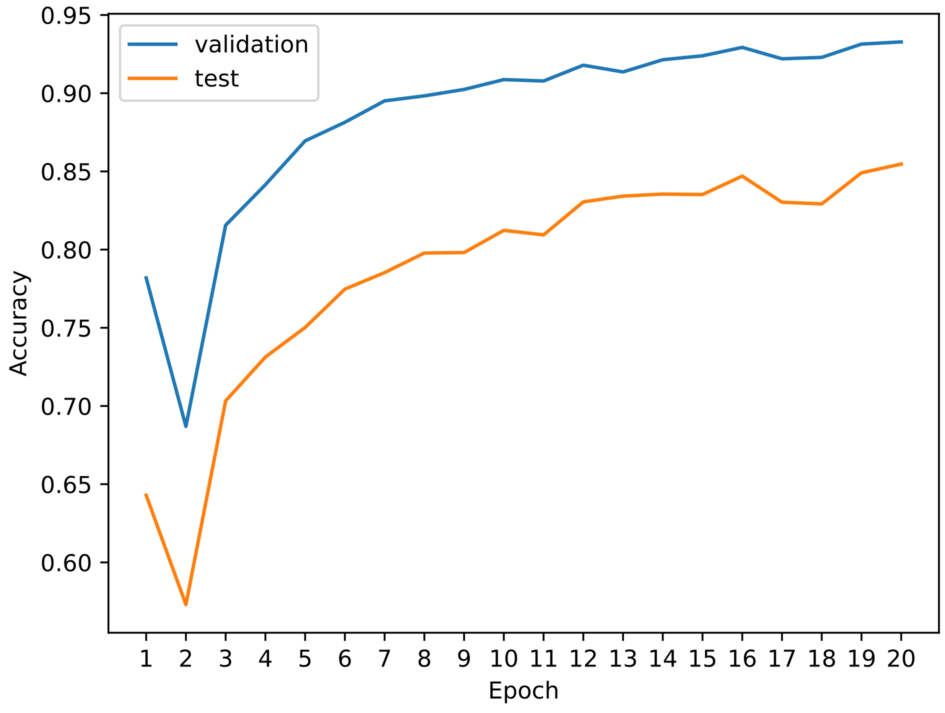

Figure 3: Multi-layer perceptron − Accuracies as a function of the Epoch number.

Question 2

Question 2.1

We obtained the following results:

Learning Rate	**0.001**	0.01	0.1
Training Loss	0.5874	0.6164	2.7043
Validation accuracy	**0.8256**	0.8033	0.7620

Table 1: Obtained results for the different learning rates

The best configuration is when the learning rate is 0.001. After 20 epochs, the linear model's final validation accuracy was 0.8256. The final accuracy on the test set was 0.7019. Below are plots that show the training loss and the validation accuracy, both as a function of the epoch number for this configuration.

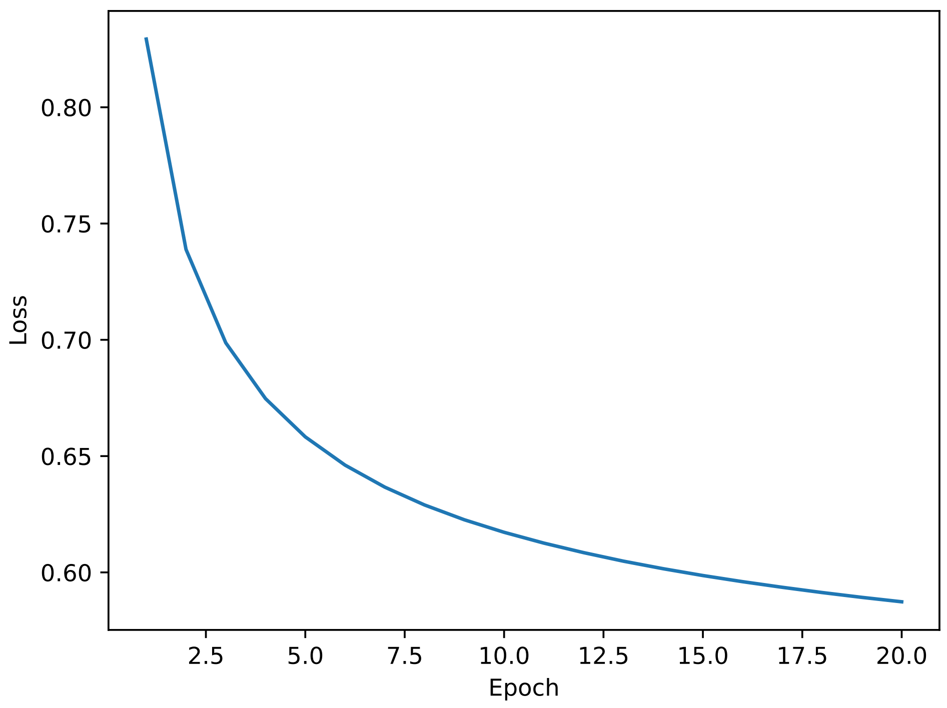

Figure 4: Logistic Regression – Training loss as a function of the Epoch number.

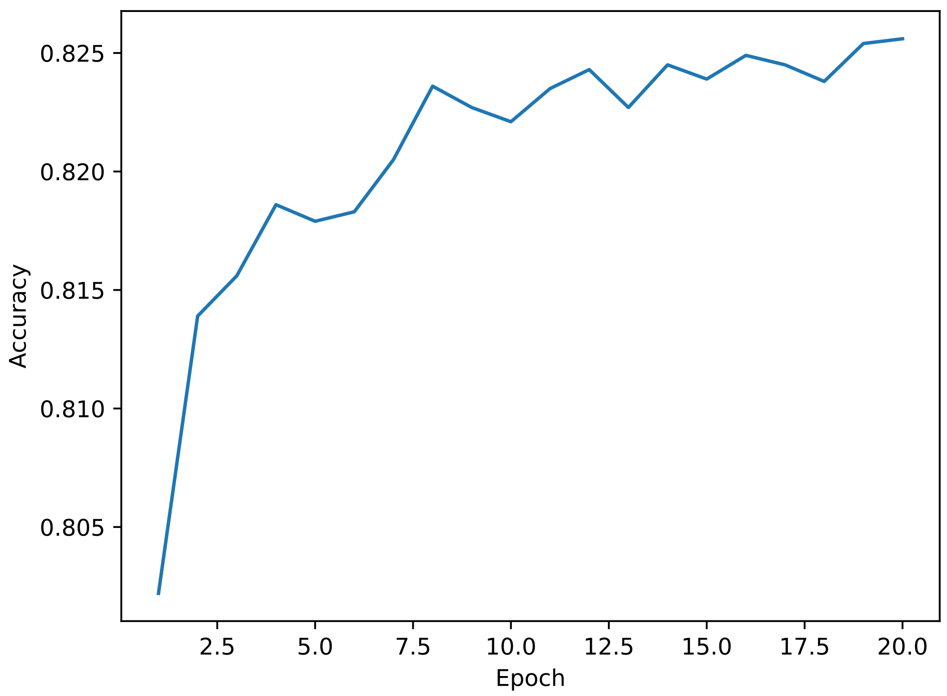

Figure 5: Logistic Regression – Validation accuracy as a function of the Epoch number.

Question 2.2

After implementing the feed-forward neural network with a single hidden layer using dropout regularization, we obtained the following results after training 20 epochs of the model using the different hyperparameters and keeping a batch size of 16 and a SGD optimizer.

Learning Rate	**0.001**	0.01	**0.1**	0.01	0.01	0.01
Hidden size	100	100	100	**200**	100	100
Dropout	0.3	0.3	0.3	0.3	**0.5**	0.3
Activation	ReLU	ReLU	ReLU	ReLU	ReLU	**Tanh**
Training Loss	0.7882	0.3518	0.2153	0.2963	0.4446	0.4525
Validation accuracy	0.8653	0.9397	0.9468	**0.9485**	0.9305	0.9155

Table 2: Obtained results for the different hyperparameters

The best configuration is when the hidden size is 200 while the other hyperparameters remain at their default value. The feed-forward neural network showed a final validation accuracy of 0.9485. The final test accuracy for this configuration was 0.8832. Below are plots that show the training loss and the validation accuracy, both as a function of the epoch number for the best configuration.

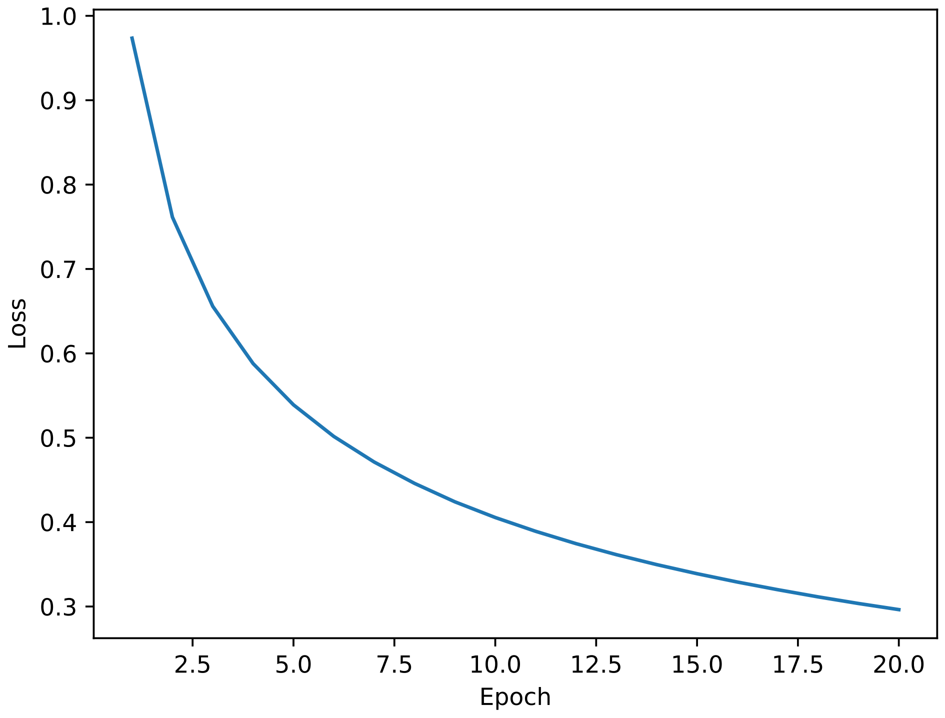

Figure 6: Multi-layer perceptron – Training loss as a function of the Epoch number.

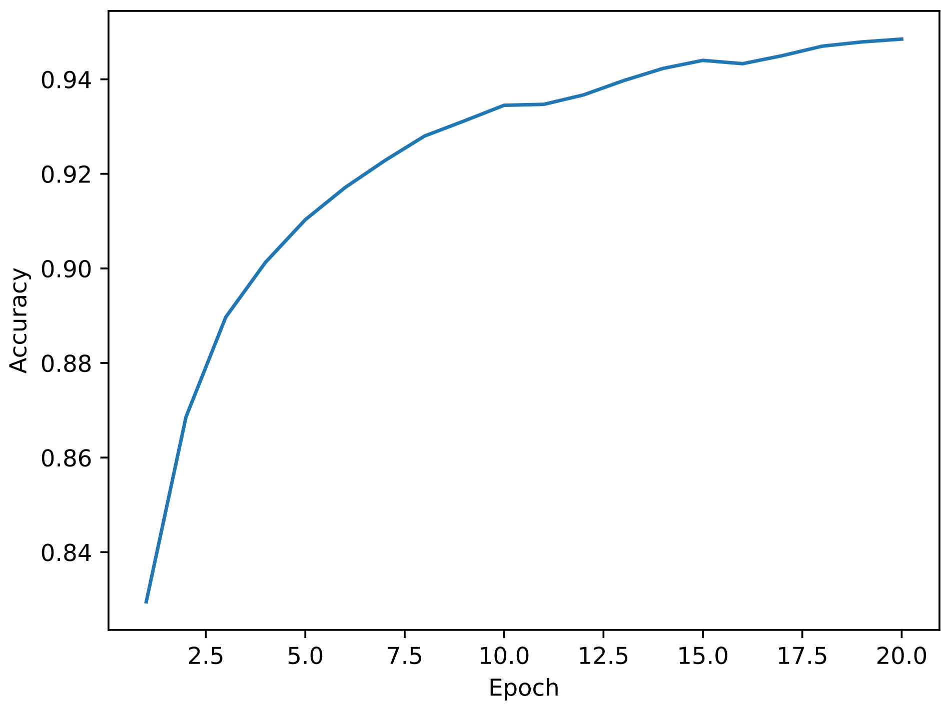

Figure 7: Multi-layer perceptron – Validation accuracy as a function of the Epoch number.

Question 2.3

Using the default hyperparameters, we obtained the following results for the models with 2 and 3 layers:

Hidden Layers	**2**	3
Training Loss	0.3671	0.4113
Validation accuracy	**0.9425**	0.9413

Table 3: Obtained results for the 2 and 3 layers with default hyperparameters

The best configuration of these two is when the model has 2 hidden layers. For this configuration, the model showed a final validation accuracy of 0.9425. The final test accuracy was 0.8717 (for 3 layers, it was 0.8599). Below are plots that show the training loss and the validation accuracy, both as a function of the epoch number for the best configuration.

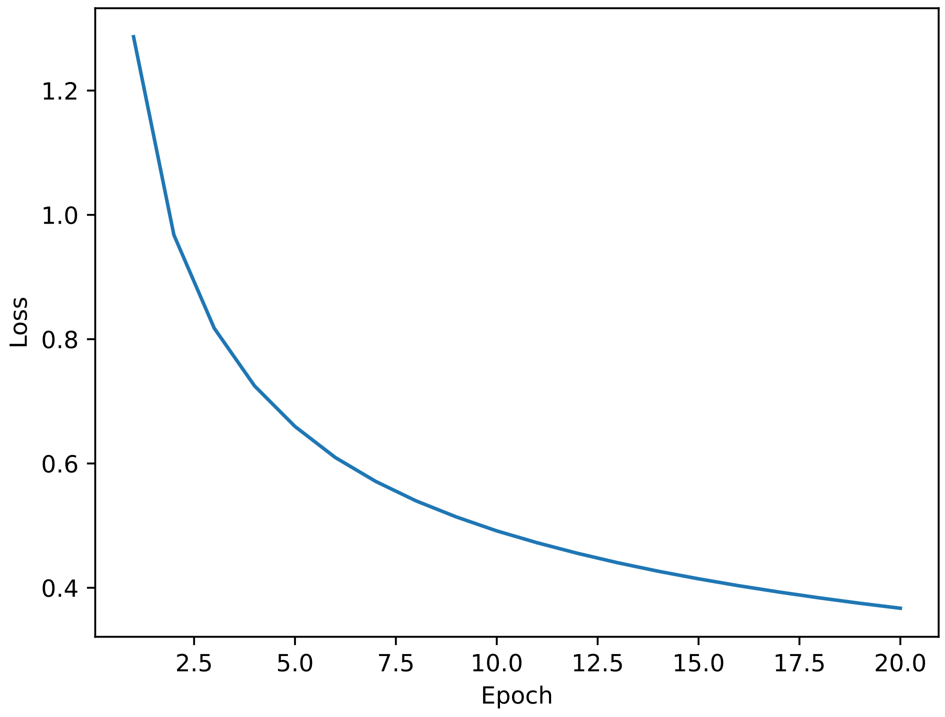

Figure 8: Multi-layer perceptron – Training loss as a function of the Epoch number.

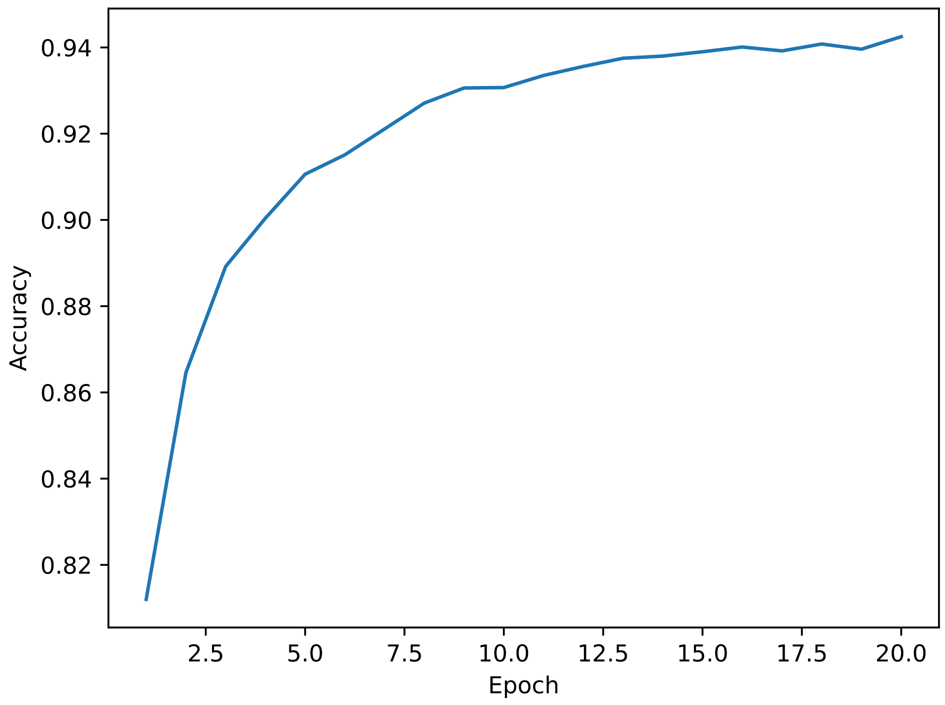

Figure 9: Multi-layer perceptron – Validation accuracy as a function of the Epoch number.

Question 3

Question 3.1

Let w_i be a vector with the entries of the ith row of W. We then have

$$\begin{aligned}
h_i = g\left(w_i^\intercal \cdot x\right) = \left(w_i^\intercal \cdot x\right)^2 &= \left(\sum_{j=1}^{D} w_{ij} \cdot x_j\right)^2 \\
&= \left(\sum_{j=1}^{D} w_{ij} \cdot x_j\right)\left(\sum_{k=1}^{D} w_{ik} \cdot x_k\right) \\
&= \sum_{j=1}^{D} w_{ij} \cdot x_j \left(w_{ij} \cdot x_j + \sum_{k \neq j} w_{ik} \cdot x_k\right) \\
&= \sum_{j=1}^{D} (w_{ij} \cdot x_j)^2 + 2\sum_{j=1}^{D}\sum_{k<j} \left(w_{ik} \cdot x_k\right)\left(w_{ij} \cdot x_j\right) \\
&= \sum_{j=1}^{D} \left(w_{ij}^2 \cdot x_j^2\right) + 2\sum_{j=1}^{D}\sum_{k<j} \left(w_{ik} \cdot w_{ij}\right)\left(x_k \cdot x_j\right)
\end{aligned}$$

Given this expression, we can observe that the left-hand side is a summation of D elements squared. The right-hand side has $\frac{D^2-D}{2} = \frac{D(D-1)}{2} = \binom{D}{2}$ elements. The total numbers of terms is then $\frac{D(D+1)}{2}$. Let

$$a_i = \begin{bmatrix} w_{i1}^2 \\ \vdots \\ w_{iD}^2 \\ w_{i1}w_{i2} \\ \vdots \\ w_{i(D-1)}w_{iD} \end{bmatrix}, \qquad A_\Theta = \begin{bmatrix} a_1^\intercal \\ \vdots \\ a_K^\intercal \end{bmatrix}, \qquad \phi(x) = \begin{bmatrix} x_1^2 \\ \vdots \\ x_D^2 \\ 2x_1x_2 \\ \vdots \\ 2x_{D-1}x_D \end{bmatrix}$$

We then have $h_i = a_i^\intercal \phi(x)$ and $\boldsymbol{h} = \boldsymbol{A_\Theta \phi(x)}$ with $A_\Theta \in \mathbb{R}^{K \times \frac{D(D+1)}{2}}$ and $\phi : \mathbb{R}^D \to \mathbb{R}^{\frac{D(D+1)}{2}}$.

Question 3.2

We have

$$\hat{y} = v^\intercal h = v^\intercal \left(A_\Theta \phi(x)\right) = \left(v^\intercal A_\Theta\right)\phi(x) = c_\Theta^\intercal \phi(x)$$
$$\text{with } c_\Theta = \left(v^\intercal A_\Theta\right)^\intercal = A_\Theta^\intercal v$$

The resulting model will **not be linear** in terms of Θ because c_Θ is **not linear** in terms of the original parameters Θ. Despite c_Θ being a linear combination of A_Θ's rows, A_Θ entries are **quadratic** – i.e. not linear – in terms of W.

Question 3.3

We have

$$\begin{aligned}
\hat{y} = v^\intercal h = \sum_{i=1}^{K} v_i h_i &= \sum_{i=1}^{K} v_i \left(w_i^\intercal \cdot x\right)^2 \\
&= \sum_{i=1}^{K} v_i \cdot w_i^\intercal \cdot x \cdot x^\intercal \cdot w_i \\
&= \sum_{i=1}^{K} v_i \sum_{j} w_{ij}^\intercal \cdot x_j \cdot x_j^\intercal \cdot w_{ij} \\
&= \sum_{i=1}^{K} v_i \sum_{j} (w_{ij}^\intercal \cdot w_{ij})(x_j \cdot x_j^\intercal) \\
&= \sum_{i=1}^{K} v_i \cdot \mathrm{Tr}(w_i^\intercal \cdot w_i, x \cdot x^\intercal) \quad \text{and since } \langle A, B\rangle_F = \mathrm{Tr}(A^\intercal B) \\
&= \sum_{i=1}^{K} v_i \cdot \langle w_i \cdot w_i^\intercal, x \cdot x^\intercal\rangle_F \\
&= \langle \sum_{i=1}^{K} v_i \cdot w_i \cdot w_i^\intercal, x \cdot x^\intercal\rangle_F \quad \text{since } v_i \text{ is a scalar} \\
&= \langle \sum_{i=1}^{K} w_i \cdot v_i \cdot w_i^\intercal, x \cdot x^\intercal\rangle_F \\
&= \langle W^\intercal \cdot V \cdot W, x \cdot x^\intercal\rangle_F \quad \text{where}
\end{aligned}$$

$$W = \begin{bmatrix} w_{11} & w_{12} & \dots & w_{1D} \\ w_{21} & \ddots & & \vdots \\ \vdots & & \ddots & \vdots \\ w_{K1} & \dots & \dots & w_{KD} \end{bmatrix} = \begin{bmatrix} w_1^\intercal \\ \vdots \\ w_K^\intercal \end{bmatrix} \in \mathbb{R}^{K\times D}, \qquad V = \begin{bmatrix} v_1 & & \\ & \ddots & \\ & & v_k \end{bmatrix} = \mathrm{diag}(v) \in \mathbb{R}^K$$

Let $C = W^\intercal \cdot V \cdot W$. Note that $W^\intercal \cdot W$ is symmetric and V is a diagonal matrix, therefore C is also a symmetric matrix and $C \in \mathbb{R}^{D\times D}$.

Since C is symmetrical, the lower triangular side of the matrix is equal to the upper triangular side of the matrix and consequently we only need the values from one side of the matrix and it's diagonal. Therefore we will have $\frac{D(D+1)}{2}$ parameters.

Since $\mathrm{rank}(C) \leq K$, when $K \geq D$ which we can collect in any $\boldsymbol{c_\Theta} \in \mathbb{R}^{\boldsymbol{\frac{D(D+1)}{2}}}$ without losing any expressive power. There are at least D linearly independent vectors that fully characterize $\mathbb{R}^{D\times D}$.

On the other hand, with $K < D$, we would not be able to collect every needed information in a $c_\Theta \in \mathbb{R}^{\frac{D(D+1)}{2}}$ without losing some information due to its dimensions.

Question 3.4

Let $X \in \mathbb{R}^{N\times \frac{D(D+1)}{2}}$ having $\phi(x_n)$ as rows and $y \in \mathbb{R}^N$ as follows:

$$X = \begin{bmatrix} \phi(x_1) \\ \vdots \\ \phi(x_n) \end{bmatrix}, \qquad y = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix}$$

To minimize the squared loss $\frac{1}{2}\sum_{n=1}^{N} \hat{y}(x_n; c_\Theta) - y_n)^2$ with respect to $\hat{c}_\Theta$, we can treat this as a linear least squares problem whose closed form solution is $\boldsymbol{\hat{c}_\Theta = (X^\intercal X)^{-1} X^\intercal y}$.

Demonstration:

$$E(c_\Theta) = \frac{1}{2}\sum_{n=1}^{N} (\hat{y}(x_n; c_\Theta) - y_n)^2 = \frac{1}{2}||Xc_\Theta - y||^2$$

Equate the gradient to zero will give us the global minimum:

$$\begin{aligned}\nabla_{c_\Theta} E = 0 &\Leftrightarrow \frac{1}{2} \cdot \nabla_{c_\Theta} ||Xc_\Theta - y||^2 = 0 \\ &\Leftrightarrow \frac{1}{2} \cdot (2X^\intercal X c_\Theta - 2X^\intercal y) = 0 \\ &\Leftrightarrow X^\intercal X c_\Theta = X^\intercal y\end{aligned}$$

Therefore we have $\boldsymbol{\hat{c}_\Theta = (X^\intercal X)^{-1} X^\intercal y}$ since $X^\intercal X$ is non-singular.

In Feed forward neural networks, the problem of learning the parameters is usually non-convex so the optimization algorithms can get stuck in local-minimums. To avoid getting stuck in a local minimum, it is often useful to use techniques such as weight initialization, early stopping and regularization.

In this particular feed forward neural network, we are dealing with a linear least squares problem which is a convex optimization problem and that there is a closed-form solution for the optimal parameters. This means that there is only one global minimum.