Deep Learning (IST, 2024-25)

Homework 1

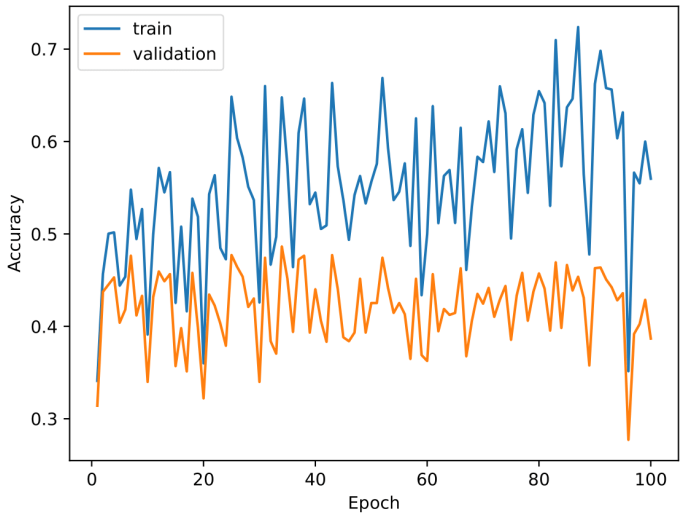
Question 1

1.

a) After training the perceptron for 100 epochs, we obtained the following results:

- **train accuracy**: 0.5598
- **validation accuracy**: 0.3868
- **test accuracy**: 0.3743

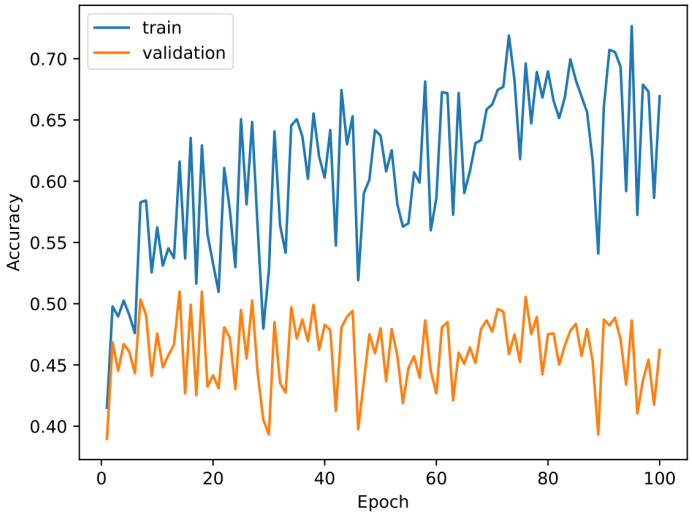
Perceptron's train and validation accuracies as a function of the epoch number



2.

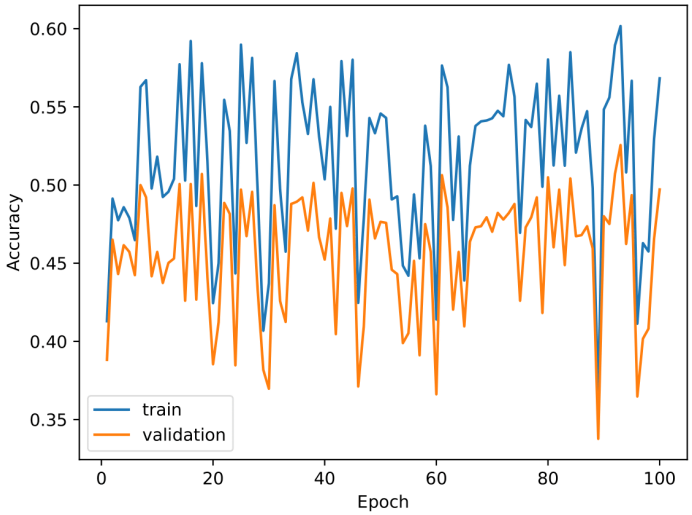
a) After training the logistic regression classifier (without ℓ_2 regularization) for 100 epochs, we obtained a final **test accuracy** of 0.4597.

Non-regularized logistic regression's train and validation accuracies over the epochs



b) After training the logistic regression classifier (with ℓ_2 regularization) for 100 epochs, we obtained a final **test accuracy** of 0.5053.

Regularized logistic regression's train and validation accuracies over the epochs



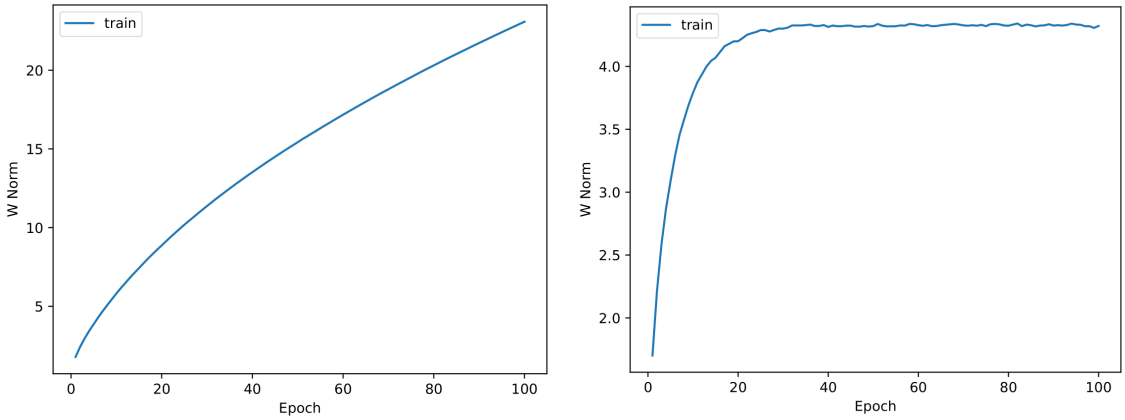
Analysis:

Comparing the training and validation accuracies across the stochastic gradient descent epochs, we can see that without regularization the training accuracies tend to be better than with it. Therefore, the differences between training and validation accuracies are greater and tend to increase with the number of epochs, which can make us infer a tendency for overfitting of the model.

So, our analysis enforces that regularization prevents overfitting as expected in theory, reducing and stabilizing the gap between training and testing accuracies. Furthermore, the final test accuracy is better including the regularization.

c)

ℓ_2-norm of the weights of non-regularized (left) and regularized (right) versions of logistic regression as a function of the number of epochs



Analysis:

According to the obtained values, it is visible that without regularization the weights tendency is to increase over the epochs and with regularization the weights become more stable and also smaller than non-regularized ones.

If the weight becomes too large, the regularization term adds a larger penalty, making the loss function increase and leading to the smaller weights seen in the graphics. Therefore, preventing overfitting and reducing the impact of noise in the training data.

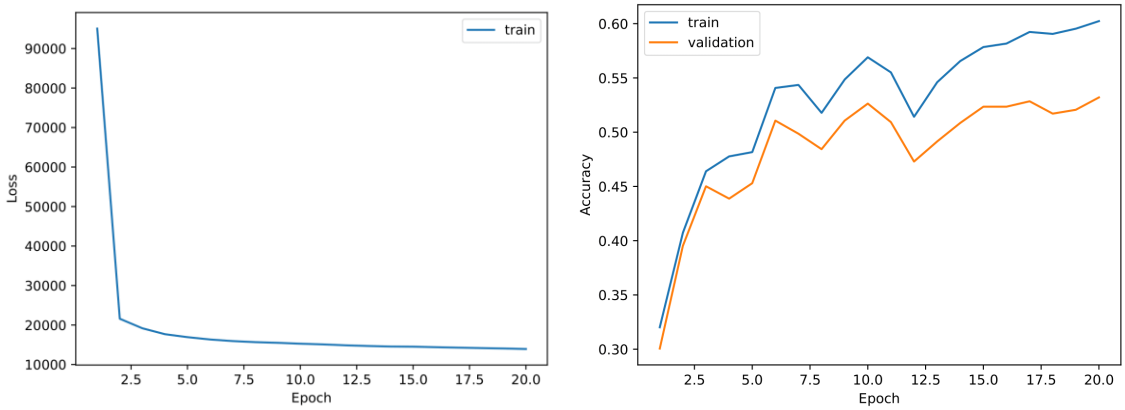
d) As ℓ_2 regularization, using ℓ_1 regularization the weights are smaller than non-regularized weights, thus preventing overfitting. The main difference between these regularizations is that ℓ_1 encourages sparsity. Features with small weights have high probability to be set to zero during training.

As of ℓ_1 norm is defined as the summation of absolute values, the gradient changes abruptly and favors a solution where many weights "spike" down to zero (at sparse points) reducing the number of non-zero weights, which leads some features to be completely ignored and to have fewer but more active features. That does not happen in ℓ_2 regularization that spreads the penalty evenly across all weights, encouraging them to be small but not zero.

3.

a) After training the model (multi-layer perceptron with the parameters given in homework statement) for 20 epochs, we obtained a final **test accuracy** of 0.5417.

Multi-layer perceptron's train loss (left) and train and validation accuracies (right) over the epochs

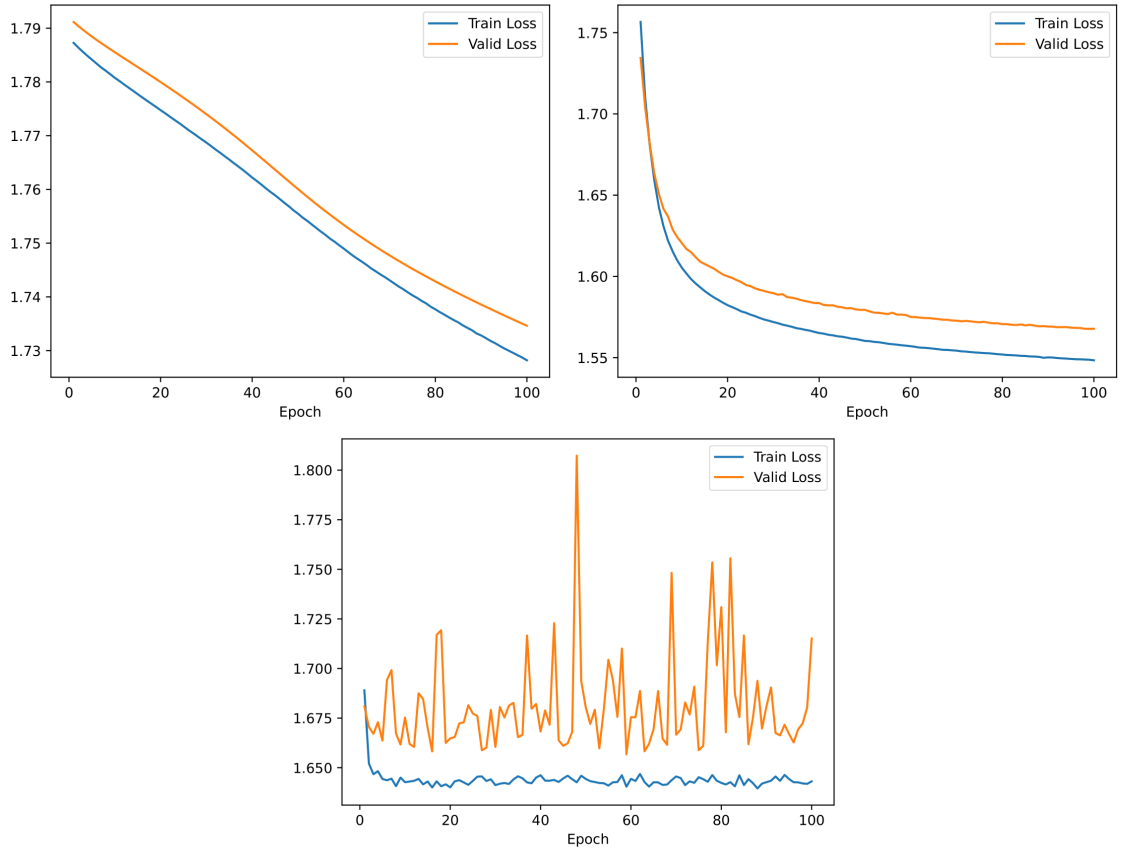


Question 2

1. After training the linear model implemented for 100 epochs with 3 different values for learning rate (0.00001, 0.001, 0.1), we obtained the following results:

- **learning rate 0.00001**
  - **validation accuracy**: 0.2892
  - **test accuracy**: 0.3087
- **learning rate 0.001**
  - **validation accuracy**: 0.4751
  - **test accuracy**: 0.4827
- **learning rate 0.00001**
  - **validation accuracy**: 0.3091
  - **test accuracy**: 0.3093

Training and validation losses for each configuration – 0.00001 (left); 0.001 (right); 0.1 (bottom)



Analysis:

Best validation accuracy was obtained for learning rate 0.001 (final validation accuracy = 0.4751). As it is possible to observe by the validation accuracy results and by the validation loss curves, from learning rates tested the closest to be optimal is 0.001.

It is perceivable that with very low (like 0.00001) learning rates the curve is almost straight, and the weights are being updated steadily but very slowly, which does not make possible to verify any approximation to a convergence point in which the loss curve reaches a minimum.

That is not the case when looking at the curve associated with the learning rate that led to the best result. We can notice the loss decreases smoothly, almost flattening along the epochs which indicates convergence.

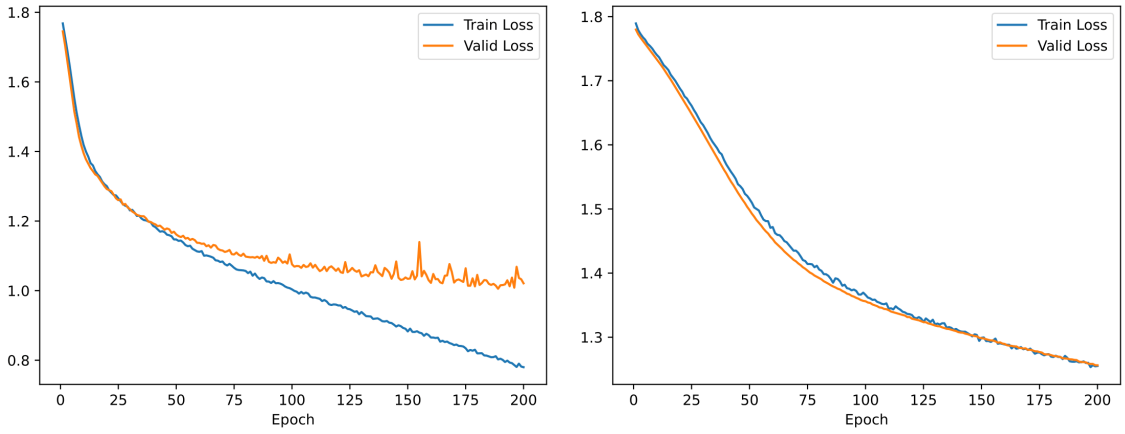
In the other hand, a very high learning rate (like 0.1) cause the loss to fluctuate and possible increase, which shows instability, overshooting the minimum and bouncing around instead of settling.

2.

a) After training the model for 200 epochs with batch size default (64) and batch size of 512, keeping the other parameters at their default value, we obtained:

- **batch size 64 (default)**
  - **validation accuracy**: 0.6061
  - **test accuracy**: 0.6073
- **batch size 512**
  - **validation accuracy**: 0.5028
  - **test accuracy**: 0.5190

Train and validation losses for each model – 64 (left); 512 (right)



Analysis:

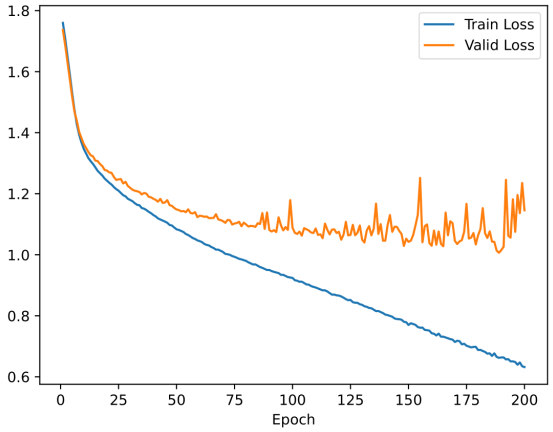
The best performance is observed with the default batch size (64). However, the execution time is shorter with the batch size set to 512. This happens because the weights need to be updated less frequently — specifically, dataset_size / batch_size times. Having a lower batch size implies a more computationally demanding classification task.

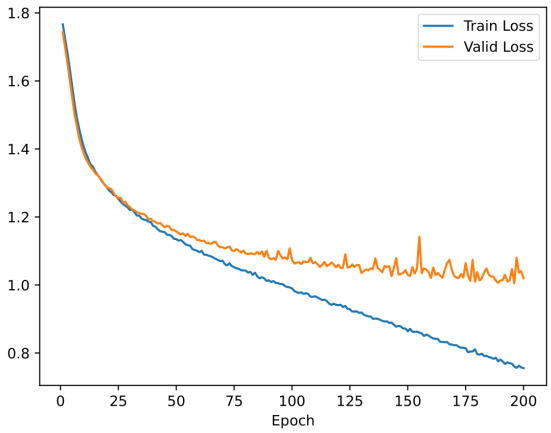
Despite this, a lower batch size allows the model to learn more effectively and capture specific patterns in the data, leading to a better accuracy. Even though, it tends to overfitting (what can be seen in the loss plots, in which the validation loss flattens regardless of the dropping related with the training loss).

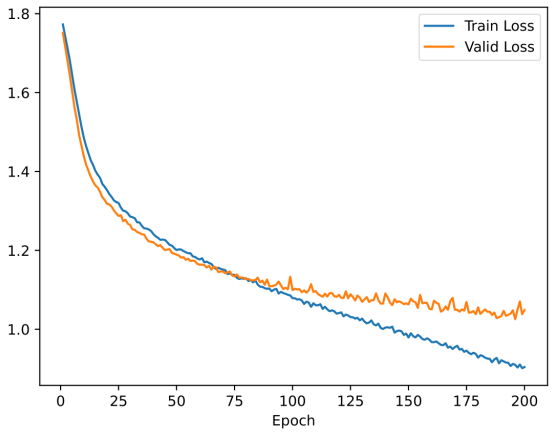
b) After training the model for 200 epochs with 3 different dropout values (0.01, 0.25, 0.5), keeping the other parameters at their default value, we obtained:

- **dropout 0.01**
  - **validation accuracy**: 0.5762
  - **test accuracy**: 0.5803
- **dropout 0.25**
  - **validation accuracy**: 0.6083
  - **test accuracy**: 0.6057
- **dropout 0.5**
  - **validation accuracy**: 0.5990
  - **test accuracy**: 0.5960

Training and validation losses for each dropout value – 0.01 (top); 0.25 (left); 0.5 (right)







Analysis:

Using a low dropout value (e.g. 0.01) only a small fraction of neurons is inactive during training, which translates in a minimal impact in terms of preventing overfitting (this is evident in the loss graph, where for this dropout rate it shows the most pronounced signs of overfitting).

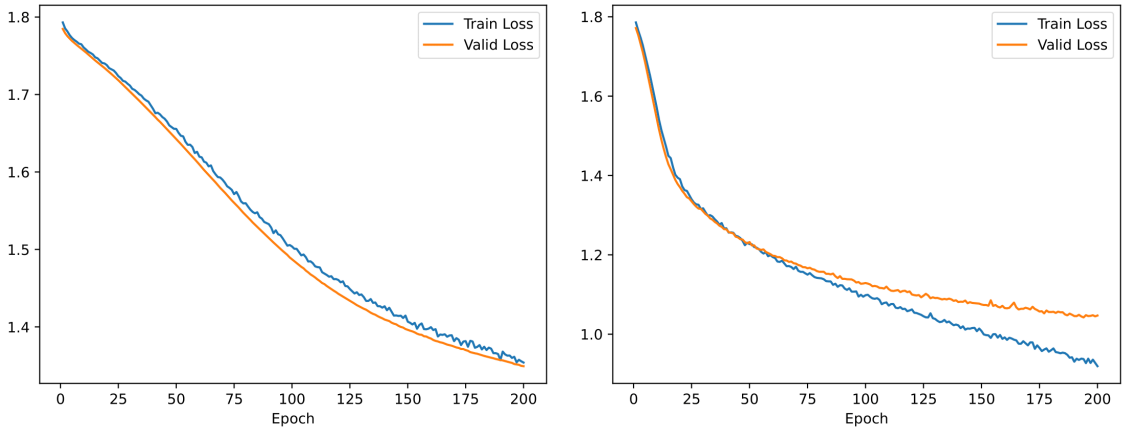
When dropout value is 0.5, half neurons are inactive during each training epoch, besides improving in generalization, it may start to excessively limit the network's capacity to learn patterns, particularly in some cases where the model struggles to converge effectively within each epoch, what is translated in a decreasing of performance. The drop in accuracy from 0.25 enforces this fact.

The best results are from training with dropout 0.25, which may indicate a better trade-off between overfitting and underfitting risks.

c) After training the model for 200 epochs with batch size of 1024 and setting momentum to 0.0 and 0.9, while keeping the other parameters at their default value, we obtained:

- **momentum 0.0**
  - **validation accuracy**: 0.4701
  - **test accuracy**: 0.4883
- **momentum 0.9**
  - **validation accuracy**: 0.5990
  - **test accuracy**: 0.6010

Train and validation losses for each momentum – 0.0 (left); 0.9 (right)



Analysis:

It is possible to verify that with a higher momentum it is achieved a better performance.

From the plots it can be seen that the convergence is faster when a momentum is applied. Hence, less epochs are needed for assisting to smaller loss values.

Moreover, the account for the weight of past gradients in the updates helps the algorithm to avoid being stuck in local minimum or saddle points and to get closer to the global minimum, attaining higher accuracy values.

Question 3

1.

From Question 3 we have:

$$g(z) = z(1-z)$$
$$= z - z^2$$

$$h = g(Wx+b) = (Wx+b) - (Wx+b)^2$$

Let W_k be the k-th column of W and b_k the k-th value of the bias vector:

$$h_k = (W_k^T x + b_k) - (W_k^T x + b_k)^2$$

expanding both terms:

$$(W_k^T x + b_k) = \sum_{i=1}^{D} W_{ki} x_i + b_k$$

$$(W_k^T x + b_k)^2 = (W_k^T x)^2 + 2b_k(W_k^T x) + b_k^2$$
$$= \sum_{i=1}^{D} W_{ki}^2 x_i^2 + 2\sum_{i<j} W_{ki} W_{kj} x_i x_j + 2b_k \sum_{i=1}^{D} W_{ki} x_i + b_k^2$$

So we have:

$$h_k = (1-2b_k)\sum_{i=1}^{D} W_{ki} x_i - \sum_{i=1}^{D} W_{ki}^2 x_i^2 - 2\sum_{i<j} W_{ki} W_{kj} x_i x_j + b_k - b_k^2$$

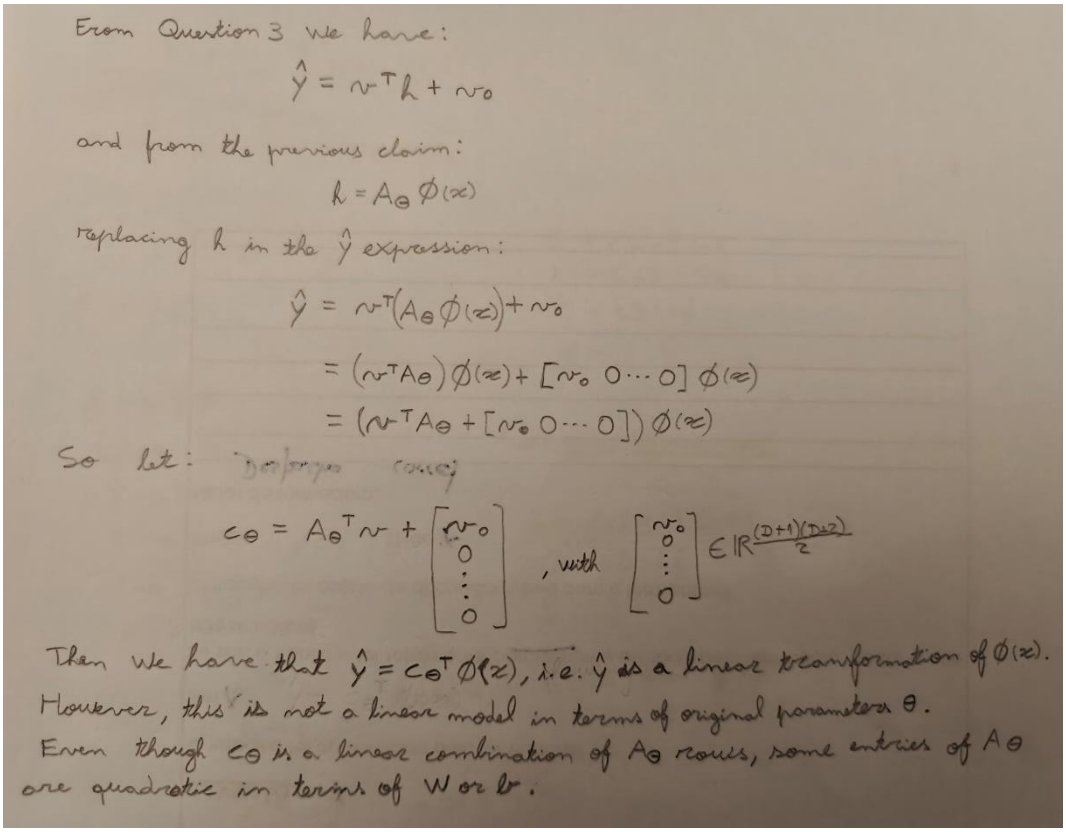
Now let: $\phi(x) = \begin{bmatrix} 1 \\ x_1 \\ \vdots \\ x_D \\ x_1^2 \\ \vdots \\ x_D^2 \\ x_1 x_2 \\ \vdots \\ x_{(D-1)} x_D \end{bmatrix} \in \mathbb{R}^{\frac{(D+1)(D+2)}{2}}$ and $a_k = \begin{bmatrix} b_k - b_k^2 \\ (1-2b_k) W_{k1} \\ \vdots \\ (1-2b_k) W_{kD} \\ -W_{k1}^2 \\ \vdots \\ -W_{kD}^2 \\ -2W_{k1} W_{k2} \\ \vdots \\ -2W_{k(D-1)} W_{kD} \end{bmatrix} \in \mathbb{R}^{\frac{(D+1)(D+2)}{2}}$

Then we have $h_k = a_k^T \phi(x)$, let:

$$A_\Theta = \begin{bmatrix} a_1^T \\ \vdots \\ a_K^T \end{bmatrix} \in \mathbb{R}^{K \times \frac{(D+1)(D+2)}{2}}$$

So $h = A_\Theta \phi(x)$, showing that h is a linear transformation of $\phi(x)$.

2.



From Question 3 we have:

$$\hat{y} = w^T h + w_0$$

and from the previous claim:

$$h = A_\theta \phi(x)$$

replacing h in the $\hat{y}$ expression:

$$\hat{y} = w^T(A_\theta \phi(x)) + w_0$$

$$= (w^T A_\theta)\phi(x) + [w_0\ 0 \cdots 0]\,\phi(x)$$

$$= (w^T A_\theta + [w_0\ 0 \cdots 0])\,\phi(x)$$

So let:

$$c_\theta = A_\theta^T w + \begin{bmatrix} w_0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \text{ with } \begin{bmatrix} w_0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \in \mathbb{R}^{\frac{(D+1)(D+2)}{2}}$$

Then we have that $\hat{y} = c_\theta^T \phi(x)$, i.e. $\hat{y}$ is a linear transformation of $\phi(x)$.
However, this is not a linear model in terms of original parameters θ.
Even though c_θ is a linear combination of A_θ rows, some entries of A_θ are quadratic in terms of W or b.

3.

4.

TODO

IMPORTANT: Please write 1 paragraph indicating clearly what was the contribution of each member of the group in this project. A penalization of 10 points will be applied if this information is missing.