AssignmentReport-Group101

February 9, 2024

1 Assignment 1 Report

Marijan Soric & Zan Stanonik

This is an outline for your report to ease the amount of work required to create your report. Jupyter notebook supports markdown, and I recommend you to check out this cheat sheet. If you are not familiar with markdown.

Before delivery, **remember to convert this file to PDF**. You can do it in two ways: 1. Print the webpage (ctrl+P or cmd+P) 2. Export with latex. This is somewhat more difficult, but you'll get somehwat of a "prettier" PDF. Go to File -> Download as -> PDF via LaTeX. You might have to install nbconvert and pandoc through conda; `conda install nbconvert pandoc`.

2 Task 1

2.1 task 1a)

The gradient for Logistic Regression is:

$$
\begin{aligned}
\frac{\partial C^n}{\partial w_i} &= \frac{\partial}{\partial w_i}\left(-\sum_{k=1}^{K} y_k^n \log(\hat{y}_k^n)\right) \\
&= -\sum_{k=1}^{K} y_k^n \frac{\partial}{\partial w_i}\left(\log(\hat{y}_k^n)\right) \\
&= -\sum_{k=1}^{K} y_k^n \frac{\partial \log(\hat{y}_k^n)}{\partial \hat{y}_k^n}\frac{\partial \hat{y}_k^n}{\partial z_k}\frac{\partial z_k}{\partial w_i} \quad \text{Chain rule} \\
&= -\sum_{k=1}^{K} y_k^n \frac{1}{\hat{y}_k^n}\cdot \hat{y}_k^n(\delta_{i,k}-\hat{y}_i^n)\cdot x_i^n \quad \text{Result from} \star \\
&= -\sum_{k=1}^{K} y_k^n(\delta_{i,k}-\hat{y}_i^n)\cdot x_i^n \\
&= -y_i^n(1-\hat{y}_i^n)x_i^n + \sum_{k\neq i} y_k^n \hat{y}_i^n x_i^n \\
&= \hat{y}_i^n x_i^n \underbrace{\left(y_i^n + \sum_{k\neq i} y_k^n\right)}_{=1} - y_i^n x_i^n \\
&= -(y_i^n - \hat{y}_i^n)x_i^n
\end{aligned}
$$

Results $\star$:

Since $\hat{y}_k^n = f(z_k)$, we have:

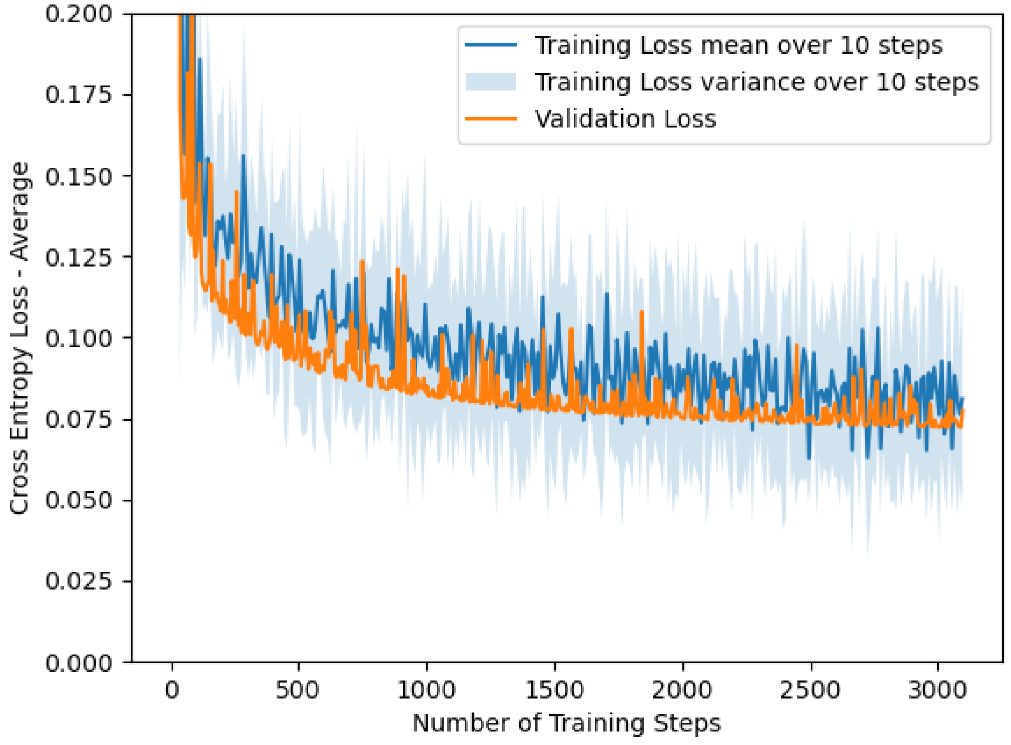
$$
\frac{\partial \hat{y}_k^n}{\partial z_i} = \frac{\partial}{\partial z_i}\left(\frac{e^{z_k}}{\sum_{k'} e^{z_{k'}}}\right) = \frac{\delta_{i,k}e^{z_k}\sum_{k'} e^{z_{k'}} - e^{z_k}e^{z_i}}{(\sum_{k'} e^{z_{k'}})^2} = \frac{e^{z_k}}{\sum_{k'} e^{z_{k'}}}\cdot\frac{\delta_{i,k}\sum_{k'} e^{z_{k'}} - e^{z_i}}{\sum_{k'} e^{z_{k'}}} = \hat{y}_k^n(\delta_{i,k}-\hat{y}_i^n)
$$

And

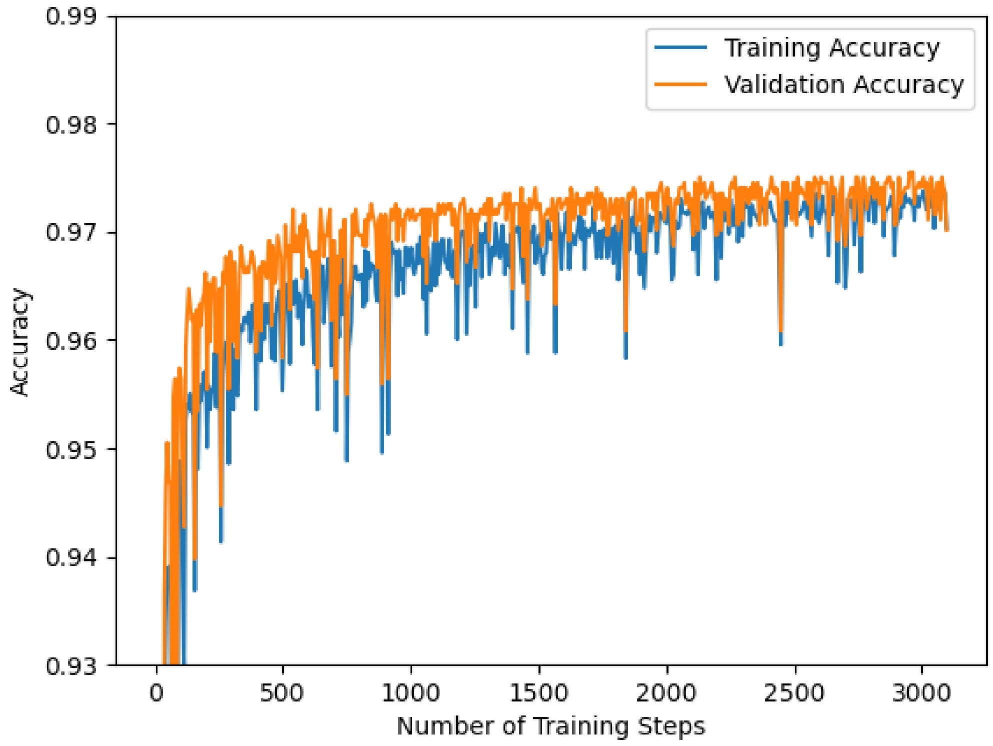
$$
\frac{\partial z_k}{\partial w_i} = \frac{\partial}{\partial w_i}\left(w_k^T \cdot x^n\right) = x_i^n
$$

3 Task 2

3.1 Task 2b)



3.2 Task 2c)



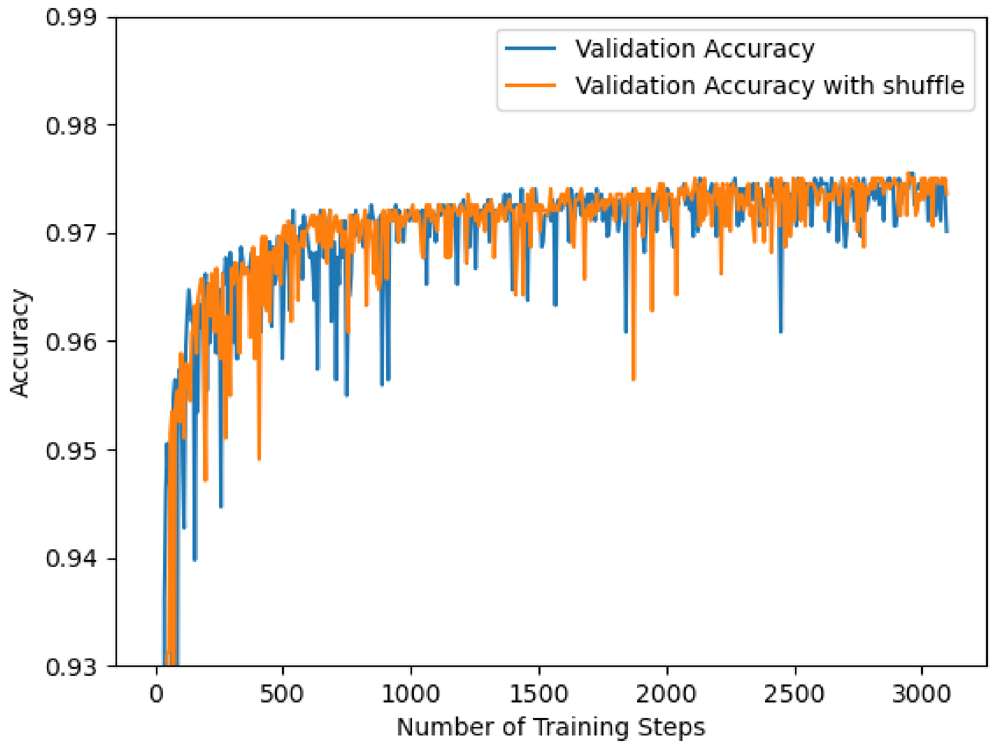
3.3 Task 2d)

After how many epochs does early stopping kick in?

Early stopping kicks in at epoch 14 already, which just goes to show how fast the model can start to overfit. It is interesting to note that with random shuffling, it stops at 74, which just goes to show that this approaches are really valid to prevent overfitting.

3.4 Task 2e)

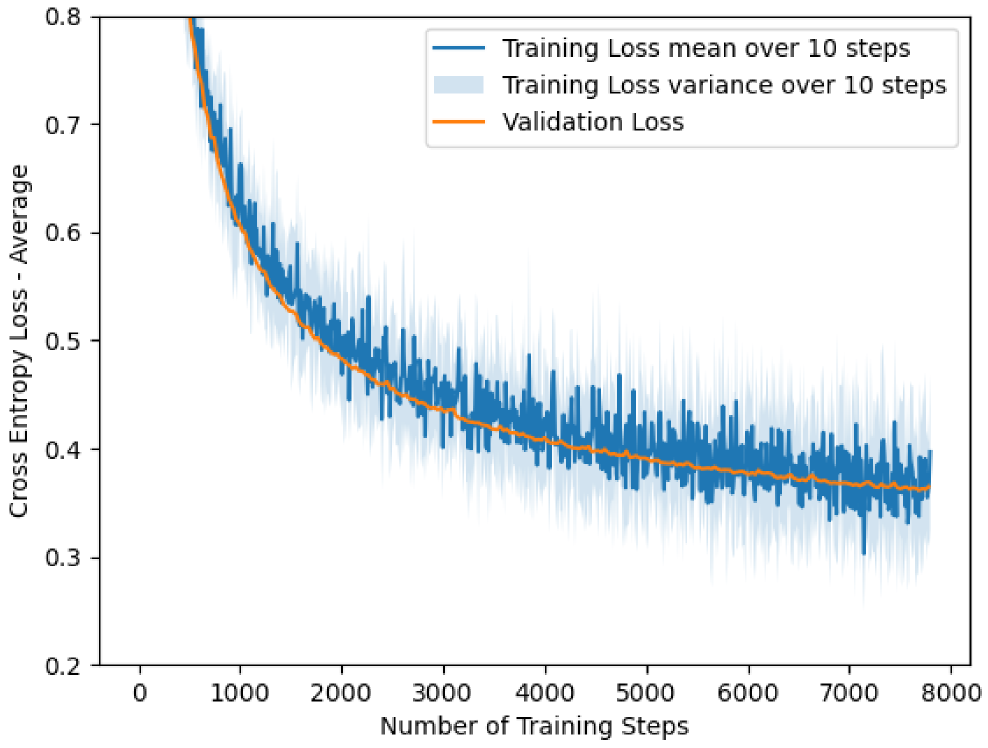
Include a plot in your report of the validation accuracy with and without shuffle. You should notice that the validation accuracy has fewer ”spikes”. Why does this happen?
This happens because the data is drawn randomly, and the model tries to learn more general features with less overfitting. Which eventually leads to less “over shoots” in the loss.



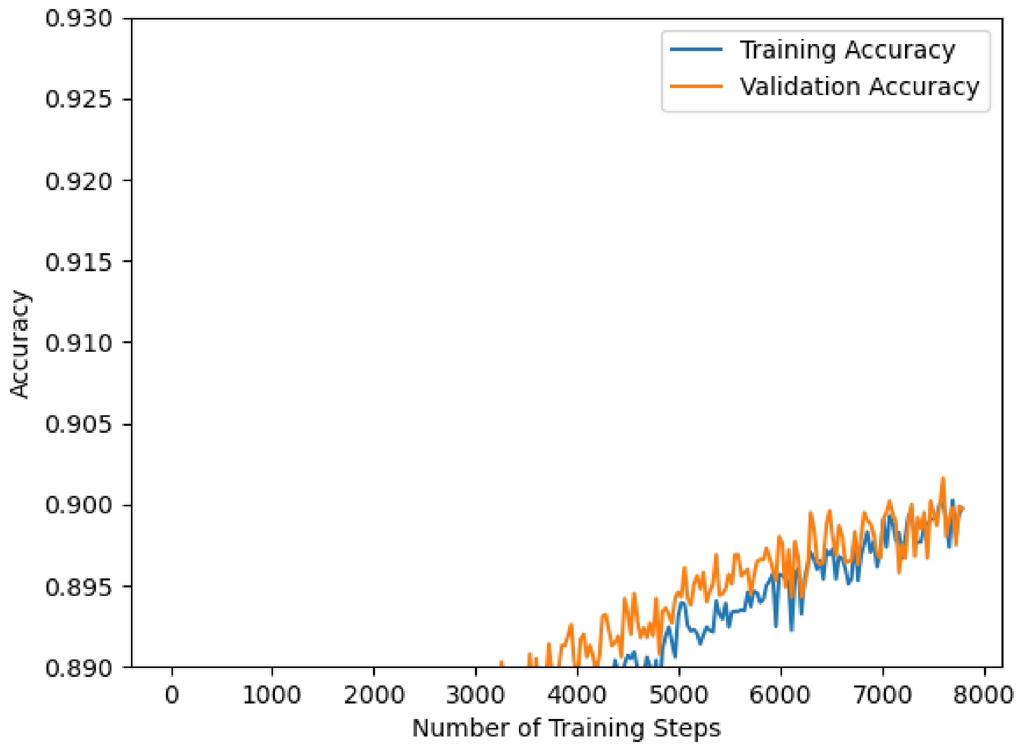
Validation Accuracy
Validation Accuracy with shuffle
Accuracy
Number of Training Steps
0.99
0.98
0.97
0.96
0.95
0.94
0.93
0
500
1000
1500
2000
2500
3000

4 Task 3

4.1 Task 3b)



4.2 Task 3c)



4.3 Task 3d)

For your model trained in task 3c, do you notice any signs of overfitting? Explain your reasoning.

We can notice that the accuracy from the training set and validation set are similar (same order). And even the validation accuracy is higher than the training which is surprising. The overfitting case happens when $accuracy_{training} >> accuracy_{validation}$.

5 Task 4

5.1 Task 4a)

The gradient for Logistic Regression is:

$$\frac{\partial J}{\partial w} = \frac{\partial C}{\partial w} + \lambda \frac{\partial R}{\partial w}$$

$$\begin{aligned}\frac{\partial R}{\partial w_{k,l}} &= \frac{\partial}{\partial w_{k,l}} \left(\sum_{i,j} w_{i,j}^2 \right) \\ &= \sum_{i,j} \frac{\partial w_{i,j}^2}{\partial w_{k,l}} \\ &= \sum_{i,j} \delta_{k,i} \delta_{l,j} 2 w_{i,j} \\ &= 2 w_{k,l}\end{aligned}$$

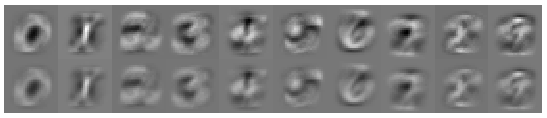
Thus,

$$\frac{\partial J}{\partial w_{k,l}} = -\frac{1}{N} \sum_{n=1}^{N} (x_l^n (y_k^n - \hat{y}_k^n)) + 2\lambda w_{k,l}$$

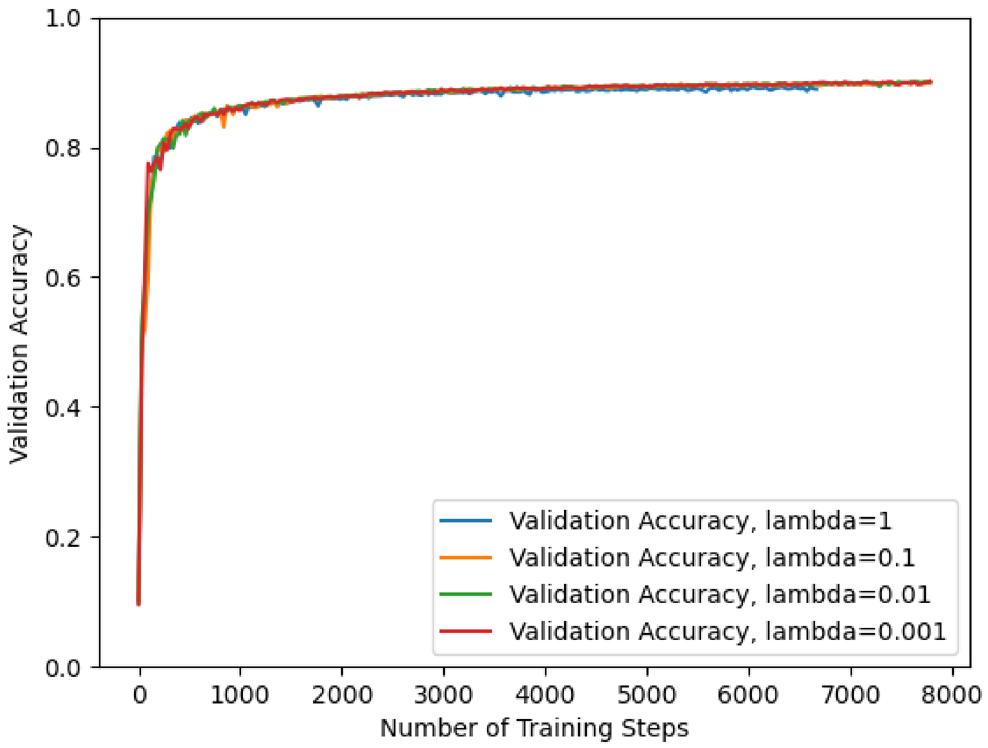
5.2 Task 4b)

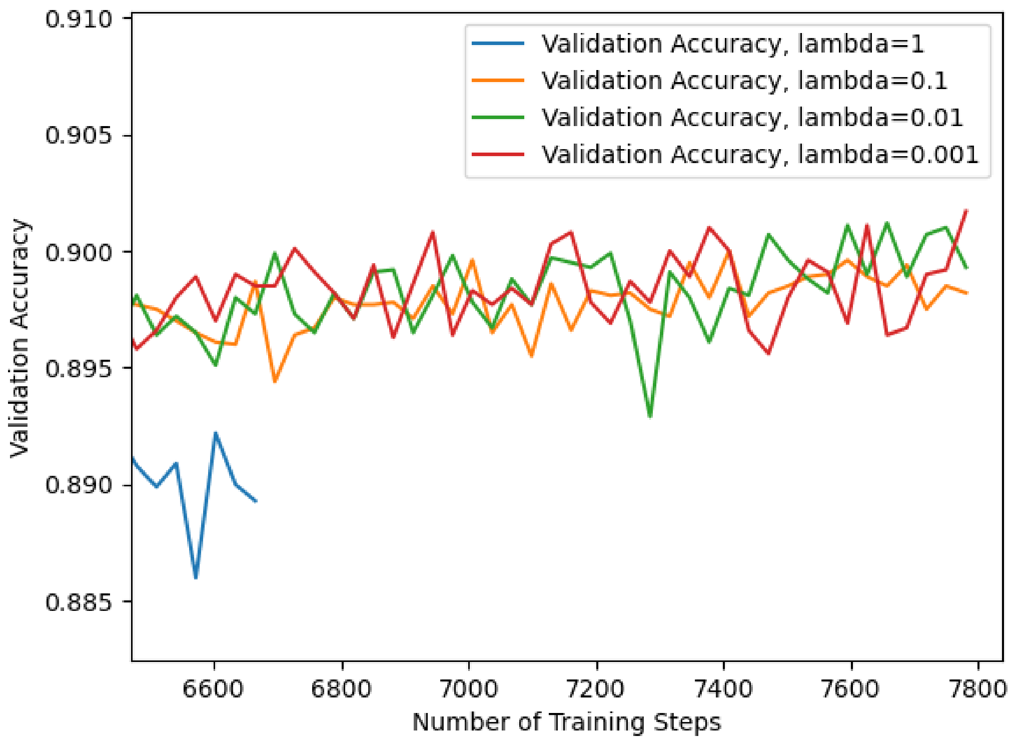
Visualize the weight for each digit for the two models. Why are the weights for the model with $\lambda = 1.0$ less noisy?

It is not clear on the picture... But, in theory, regularization helps to prevent overfitting. The weights can't have value that are too high because the regularization penalizes them (they are included in the gradient). The variance between weights is lower with regularization which means less noisy weights.



5.3 Task 4c)


1.0
0.8
0.6
0.4
0.2
0.0
Validation Accuracy
0
1000
2000
3000
4000
5000
6000
7000
8000
Number of Training Steps
Validation Accuracy, lambda=1
Validation Accuracy, lambda=0.1
Validation Accuracy, lambda=0.01
Validation Accuracy, lambda=0.001



5.4 Task 4d)

You will notice that the validation accuracy degrades when applying any amount of regularization. What do you think is the reason for this?

When λ is too high, the weights become very small and similar, then it degrades the model (low variance but high biais). This is the trade-off biais-variance.

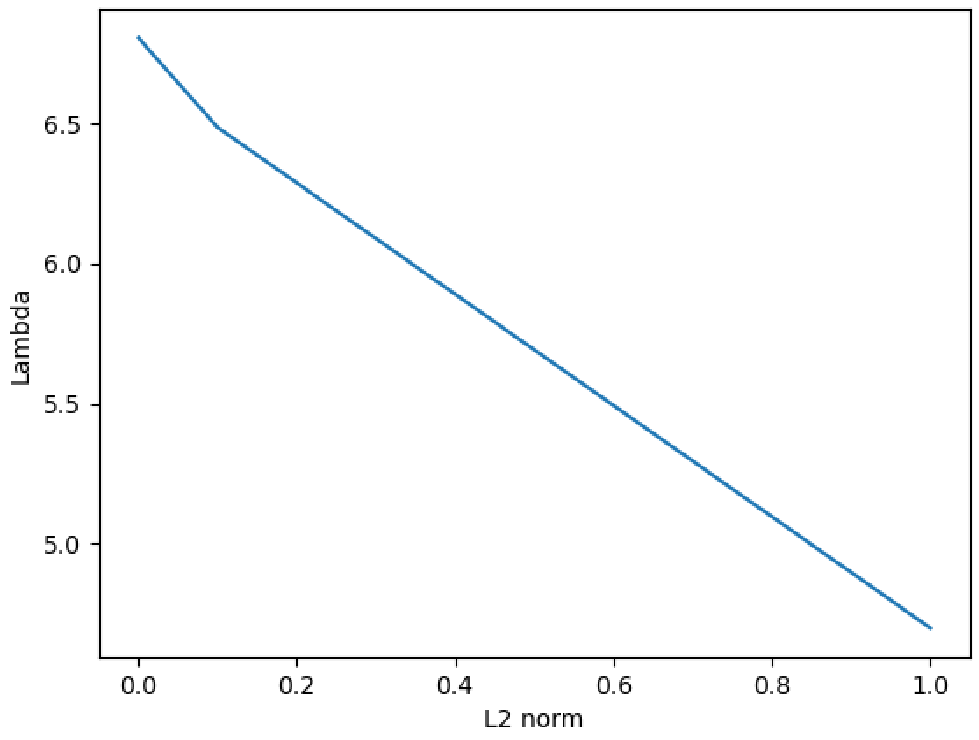
5.5 Task 4e)

NB : There is a mistake here: we should have on x-axis λ and on y-axis L_2.

As expected, $\frac{\partial \|w\|^2}{\partial \lambda} < 0$. As calculated before, in the gradient descent, we have:

$$\frac{\partial J}{\partial w_{k,l}} = -\frac{1}{N} \sum_{n=1}^{N} (x_l^n (y_k^n - \hat{y}_k^n)) + 2\lambda w_{k,l}$$

So the gradient penalizes high value of weights, then the weights are updated such that they are "not too high". That is why we have lower $\|w\|^2$ when we increase the value of the hyperparameter λ.



Lambda
6.5
6.0
5.5
5.0
0.0
0.2
0.4
0.6
0.8
1.0
L2 norm