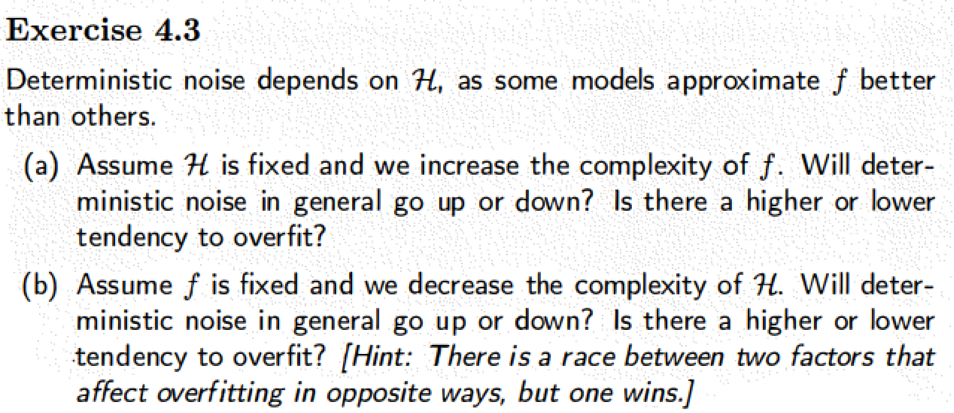
CS436/536: Introduction to Machine Learning
Homework 3

1) LFD Exercise 4.3

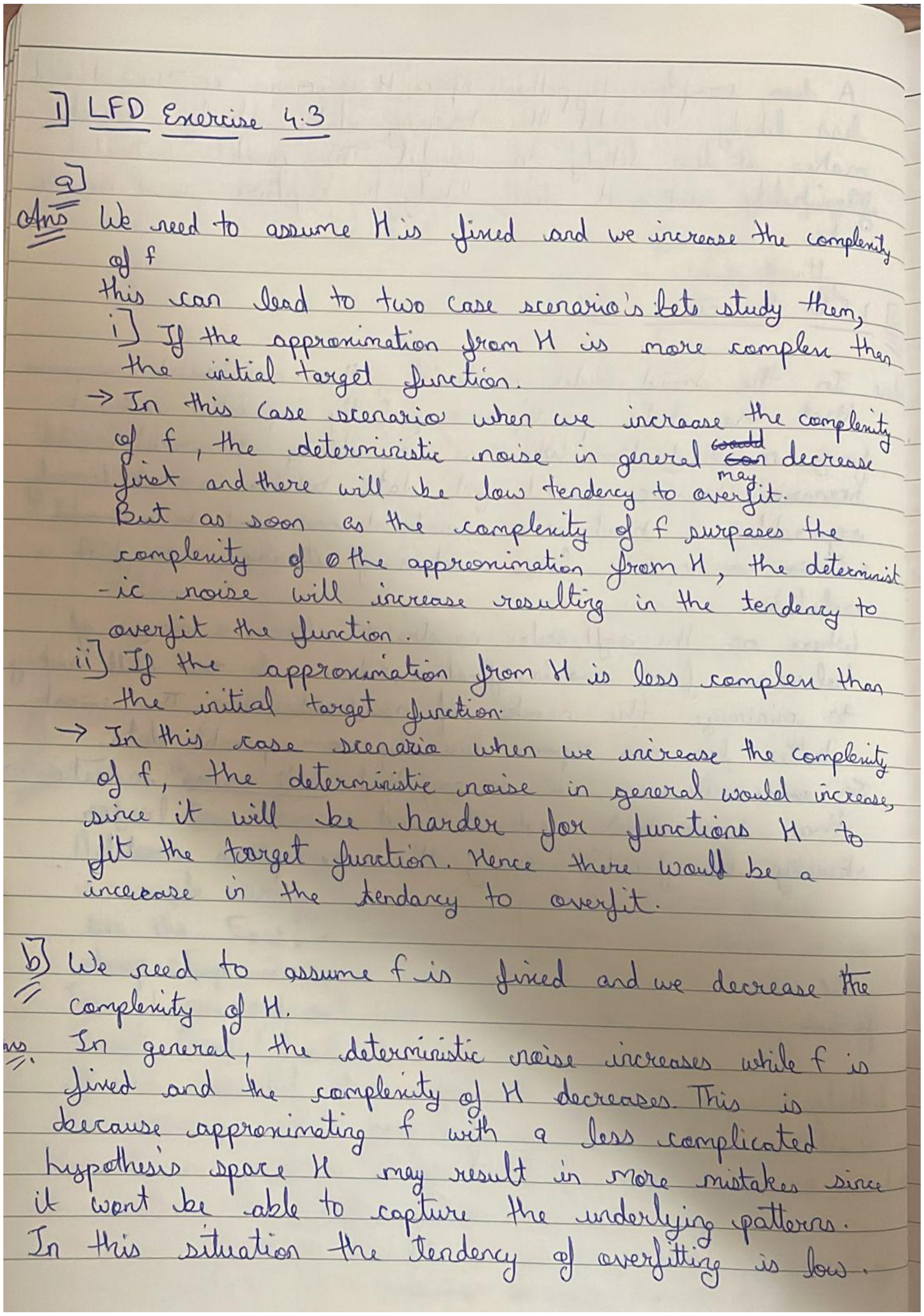


Exercise 4.3

Deterministic noise depends on $\mathcal{H}$, as some models approximate f better than others.

(a) Assume $\mathcal{H}$ is fixed and we increase the complexity of f. Will deterministic noise in general go up or down? Is there a higher or lower tendency to overfit?

(b) Assume f is fixed and we decrease the complexity of $\mathcal{H}$. Will deterministic noise in general go up or down? Is there a higher or lower tendency to overfit? *[Hint: There is a race between two factors that affect overfitting in opposite ways, but one wins.]*



I] LFD Exercise 4.3

a]

Ans We need to assume H is fixed and we increase the complexity of f

this can lead to two case scenario's lets study them,

i] If the approximation from H is more complex than the initial target function.

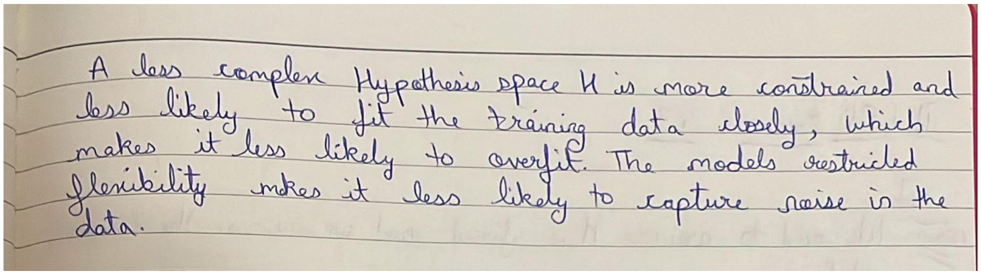
→ In this case scenario when we increase the complexity of f, the deterministic noise in general ~~could~~ ~~can~~ may decrease first and there will be low tendency to overfit.
But as soon as the complexity of f surpases the complexity of the approximation from H, the deterministic noise will increase resulting in the tendency to overfit the function.

ii] If the approximation from H is less complex than the initial target function.

→ In this case scenario when we increase the complexity of f, the deterministic noise in general would increase, since it will be harder for functions H to fit the target function. Hence there would be a increase in the tendency to overfit.

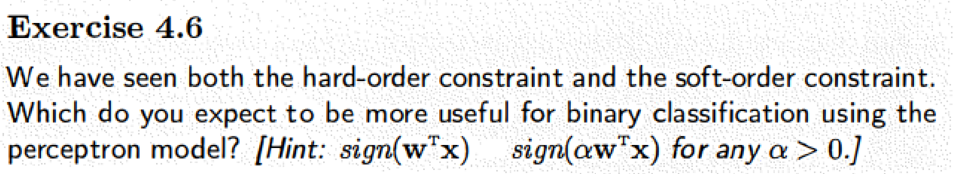
b] We need to assume f is fixed and we decrease the complexity of H.

ns. In general, the deterministic noise increases while f is fixed and the complexity of H decreases. This is because approximating f with a less complicated hypothesis space H may result in more mistakes since it wont be able to capture the underlying patterns.
In this situation the tendency of overfitting is low.



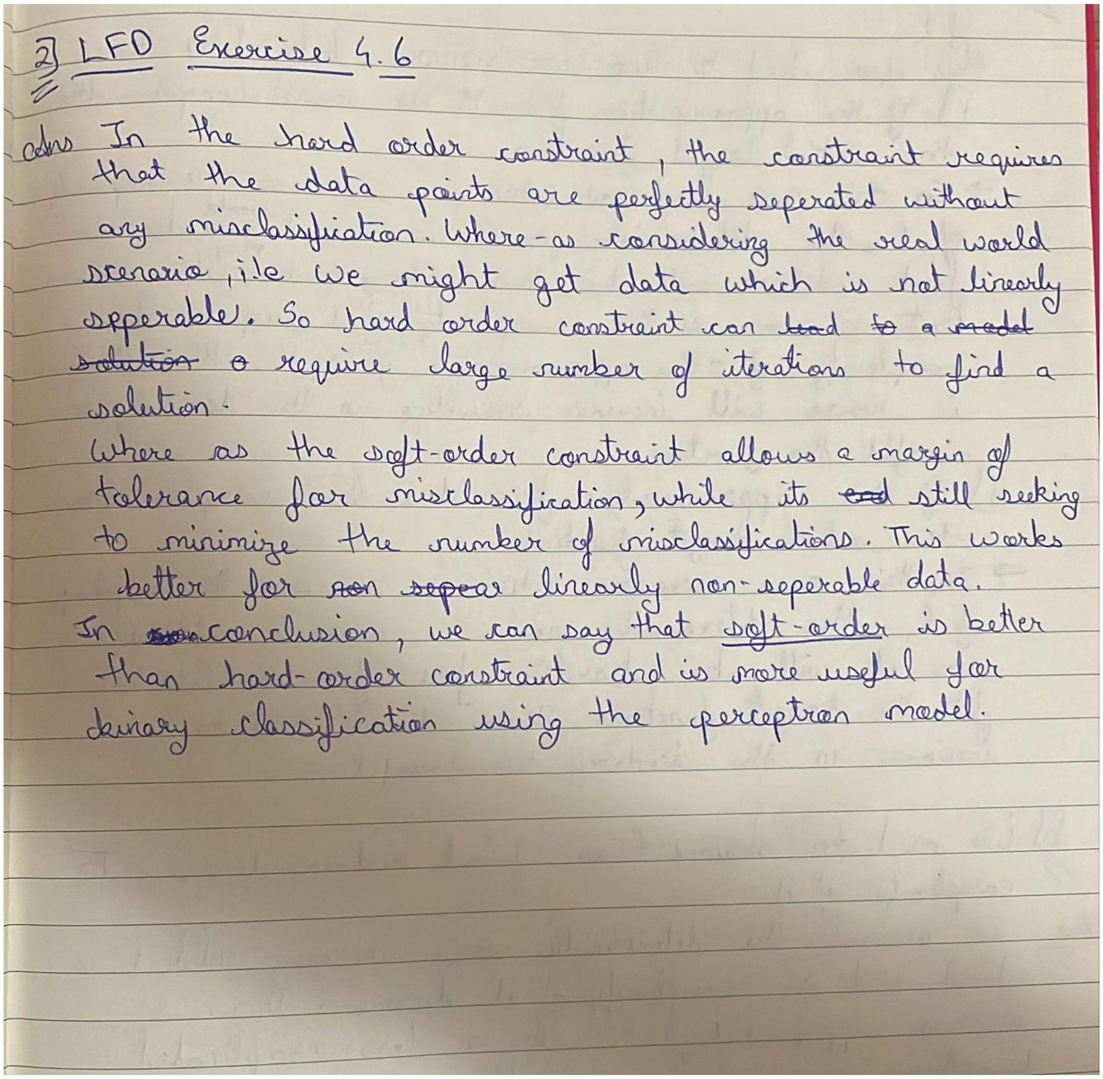
A less complex Hypothesis space H is more constrained and less likely to fit the training data closely, which makes it less likely to overfit. The models restricted flexibility makes it less likely to capture noise in the data.

2) LFD Exercise 4.6



Exercise 4.6

We have seen both the hard-order constraint and the soft-order constraint. Which do you expect to be more useful for binary classification using the perceptron model? *[Hint: sign*$(\mathbf{w}^{\mathsf{T}}\mathbf{x})$ $\quad$ *sign*$(\alpha\mathbf{w}^{\mathsf{T}}\mathbf{x})$ *for any* $\alpha > 0$*.]*



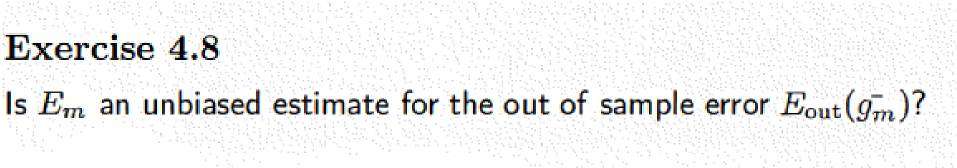
2] LFD Exercise 4.6

Ans In the hard order constraint, the constraint requires that the data points are perfectly seperated without any misclassification. Where-as considering the real world scenario, i.e we might get data which is not linearly sepperable. So hard order constraint can ~~lead~~ ~~to~~ a ~~model~~ ~~solution~~ ~~o~~ require large number of iterations to find a solution.

Where as the soft-order constraint allows a margin of tolerance for misclassification, while its ~~end~~ still seeking to minimize the number of misclassifications. This works better for ~~non~~ ~~sepear~~ linearly non-seperable data.

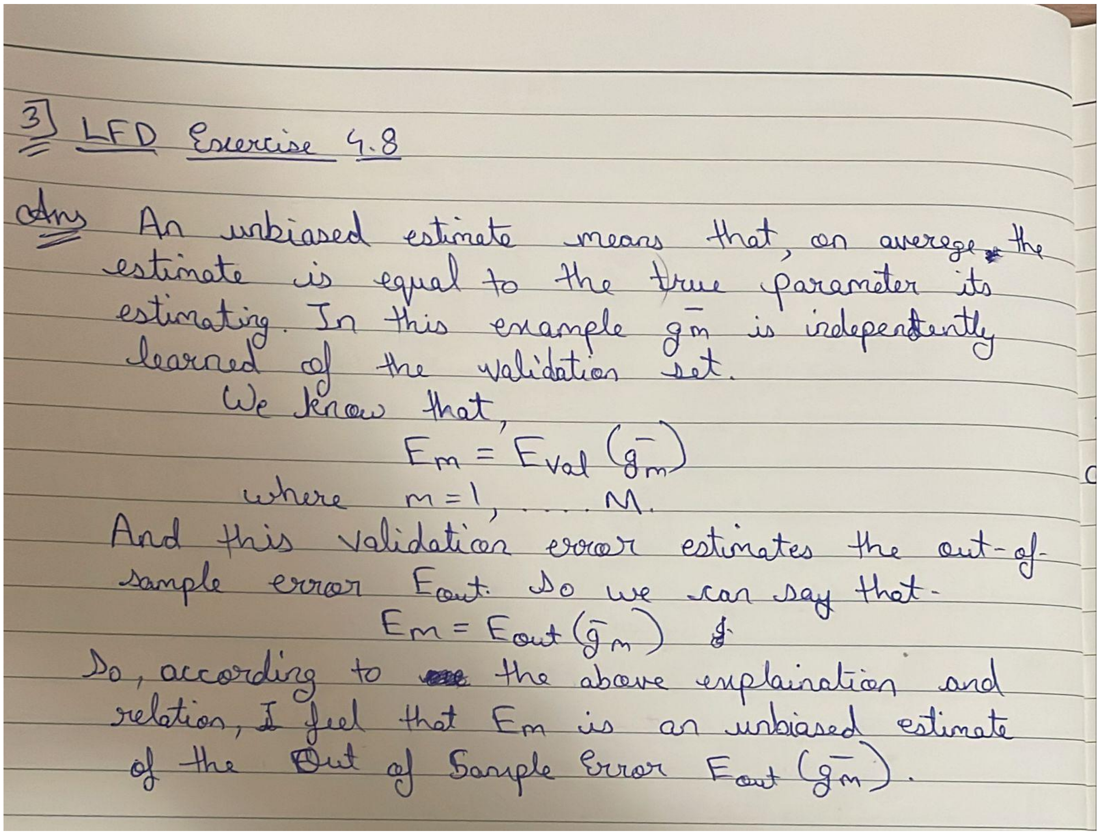
In conclusion, we can say that soft-order is better than hard-order constraint and is more useful for binary classification using the perceptron model.

3) LFD Exercise 4.8



Exercise 4.8

Is E_m an unbiased estimate for the out of sample error $E_{\text{out}}(g_m^-)$?



3] LFD Exercise 4.8

Ans An unbiased estimate means that, on averege the estimate is equal to the true parameter its estimating. In this example $\bar{g_m}$ is independently learned of the validation set.

We know that,

$$E_m = E_{val}(\bar{g_m})$$

where $m = 1, \ldots M$.

And this validation error estimates the out-of-sample error E_{out}. So we can say that -

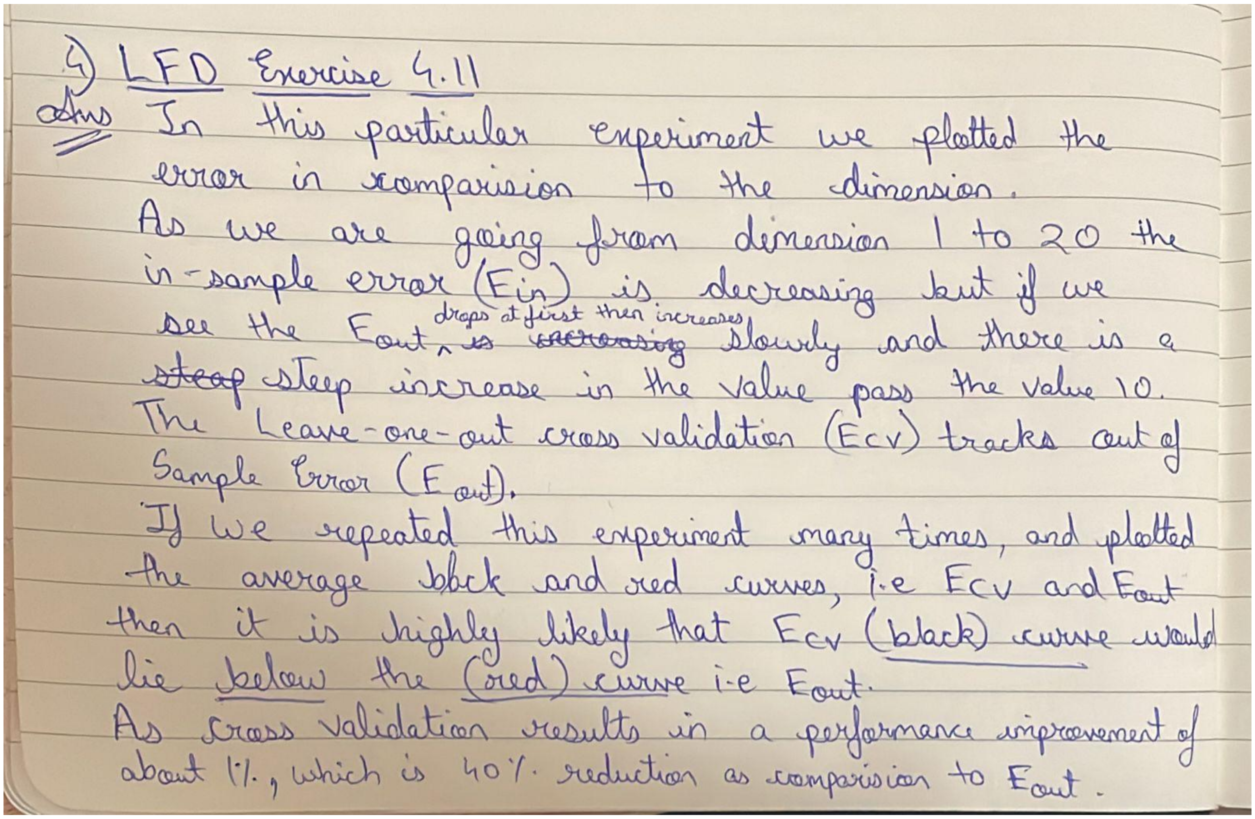
$$E_m = E_{out}(\bar{g_m})$$

So, according to the above explaination and relation, I feel that E_m is an unbiased estimate of the Out of Sample Error $E_{out}(\bar{g_m})$.

4) LFD Exercise 4.11

Exercise 4.11

In this particular experiment, the black curve (E_{cv}) is sometimes below and sometimes above the the red curve (E_{out}). If we repeated this experiment many times, and plotted the average black and red curves, would you expect the black curve to lie above or below the red curve?



4) LFD Exercise 4.11

Ans In this particular experiment we plotted the error in comparision to the dimension.

As we are going from dimension 1 to 20 the in-sample error (E_{in}) is decreasing but if we see the E_{out} drops at first then increases ~~is increasing~~ slowly and there is a ~~steap~~ steep increase in the value pass the value 10.

The Leave-one-out cross validation (E_{cv}) tracks out of Sample Error (E_{out}).

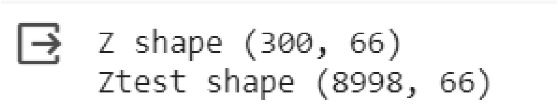
If we repeated this experiment many times, and plotted the average black and red curves, i.e E_{cv} and E_{out} then it is highly likely that E_{cv} (black) curve would lie below the (red) curve i.e E_{out}.

As cross validation results in a performance improvement of about 1%, which is 40% reduction as comparision to E_{out}.

5) An End-to-End Learning System with Regularization and Validation: Predicting 1s vs. Not 1s. We revisit the MNIST Handwritten Digits Dataset we worked with in the last homework to solve the problem of predicting whether a given image of a handwritten digit represents either the digit 1 or not the digit 1, i.e., if the n-th example is labeled as being the digit 1, then yn = +1, and otherwise, yn = −1.

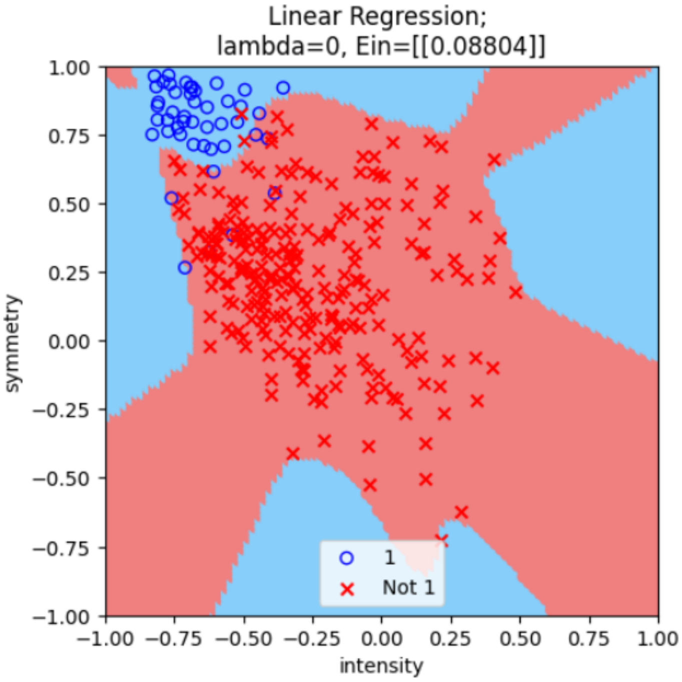
Task:

1. **10-th order Polynomial Transform**. Use the 10-th order Legendre polynomial feature transform to compute Z. Report the dimensions of Z.

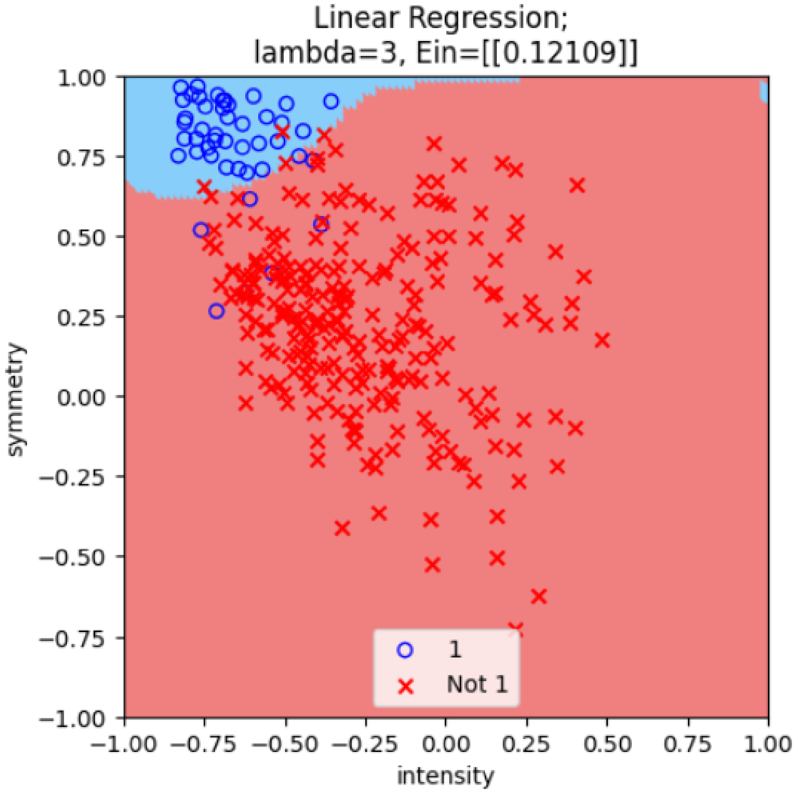


```
Z shape (300, 66)
Ztest shape (8998, 66)
```

2. **Overfitting**. Plot the decision boundary of the output of the regularized linear regression algorithm without any regularization (λ = 0). What do you observe, overfitting or underfitting?
For λ = 0, we observe overfitting as it tries to map the data points way too perfectly, as shown in the graph plotted below.



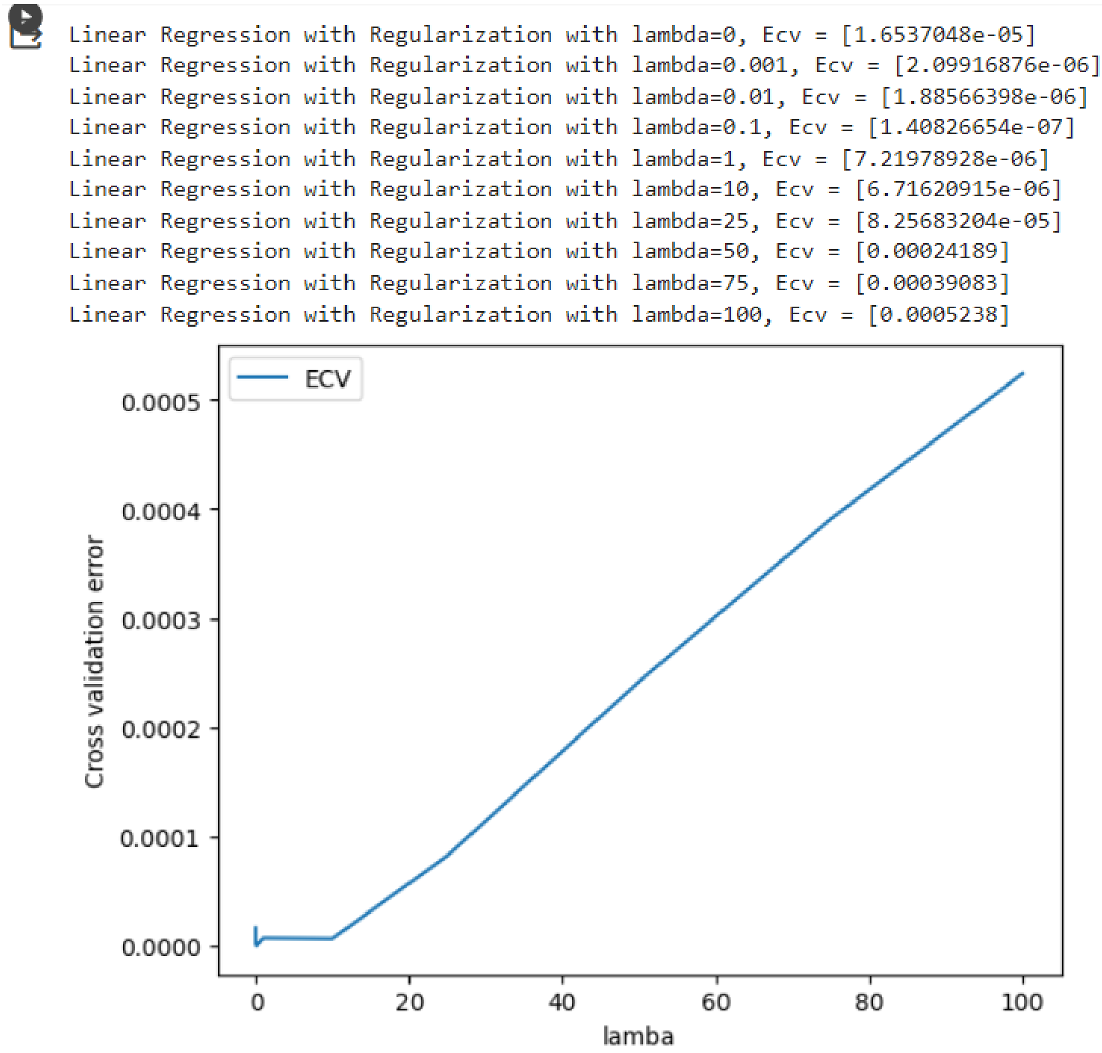
3. **Regularization**. Plot the decision boundary of the output of the regularized linear regression algorithm with λ = 3. Do you observe overfitting or underfitting?
For λ = 3, we observe a good fit, as the data points are classified appropriately with the separator, We should observe smaller out of sample error(Eout) for this particular separator.



4. **Cross Validation**. Use leave-one-out cross validation to estimate ECV(λ) for λ ∈ {0, 0.01, 0.1, 1, 5, 10, 25, 50, 75, 100}. Plot ECV versus λ and Etest(wlin(λ)) versus λ on the same plot. Comment on the behavior of ECV and Etest versus λ. Here, ECV and Etest are the regression, sum of squared errors.

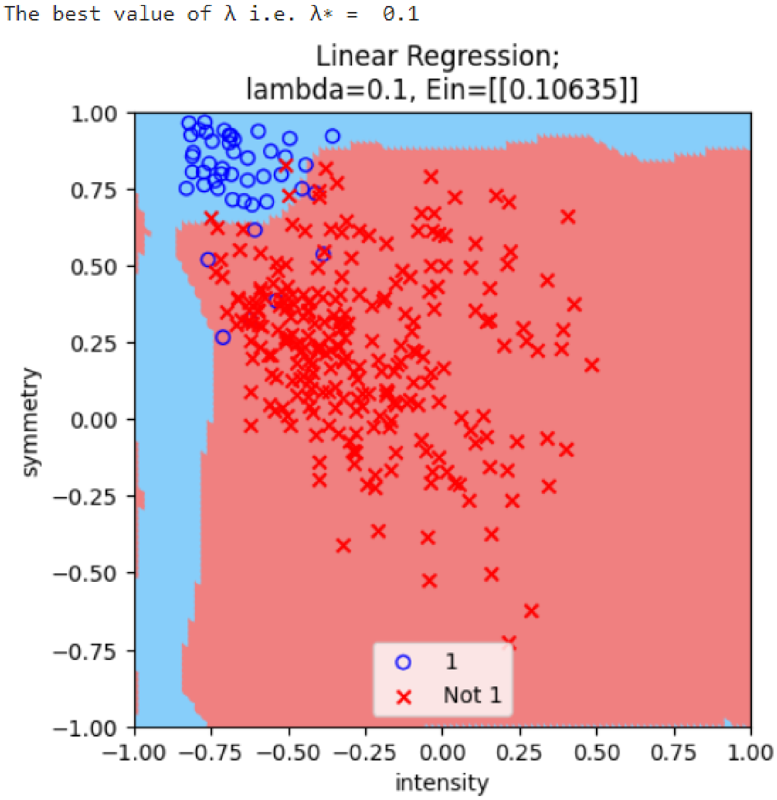
Answer:

The ECV lies below the Etest for most of the values. We can achieve better performance using ECV as compared to Etest as we use data split, averaging and randomization. This allows us to get better performance while also preventing data leakage.



```
Linear Regression with Regularization with lambda=0, Ecv = [1.6537048e-05]
Linear Regression with Regularization with lambda=0.001, Ecv = [2.09916876e-06]
Linear Regression with Regularization with lambda=0.01, Ecv = [1.88566398e-06]
Linear Regression with Regularization with lambda=0.1, Ecv = [1.40826654e-07]
Linear Regression with Regularization with lambda=1, Ecv = [7.21978928e-06]
Linear Regression with Regularization with lambda=10, Ecv = [6.71620915e-06]
Linear Regression with Regularization with lambda=25, Ecv = [8.25683204e-05]
Linear Regression with Regularization with lambda=50, Ecv = [0.00024189]
Linear Regression with Regularization with lambda=75, Ecv = [0.00039083]
Linear Regression with Regularization with lambda=100, Ecv = [0.0005238]
```

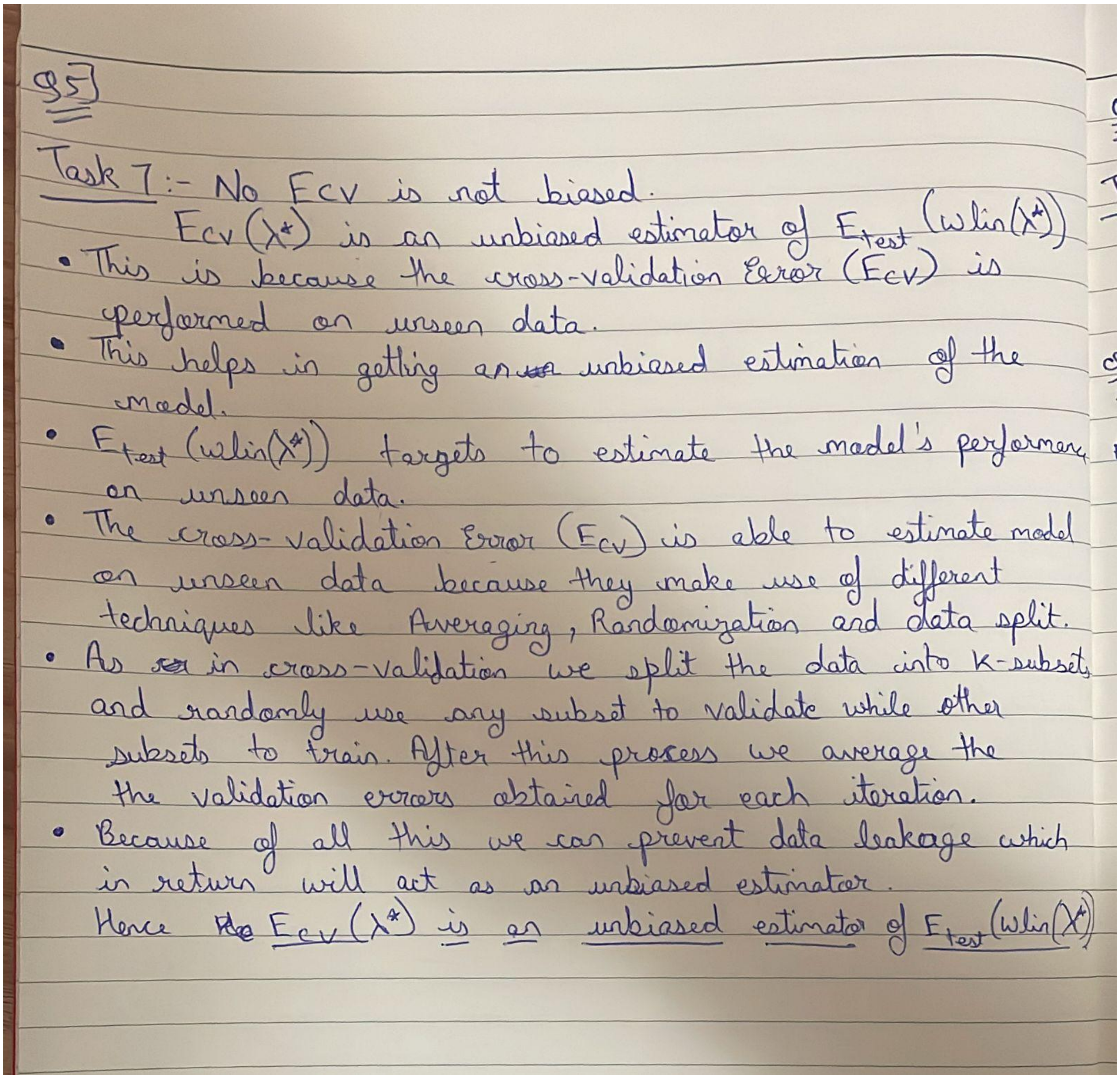
5. **Pick** λ. Use the cross validation errors from the previous step to pick the best value of λ, and call it λ ∗ . Plot the decision boundary corresponding to the weights wlin(λ∗) .



```
The best value of λ i.e. λ* =  0.1
```

6. **Estimate Classification Error**. Use wlin(λ∗) for classification and estimate the classification out-of-sample error Eout(wlin(λ∗)) for your final hypothesis g. Estimate Eout(g) to distinguish between digits that are 1s and not 1s (give the 99% error bar).

7. **Is ECV biased**? Comment on whether ECV(λ ∗) is an unbiased estimator of Etest(wlin(λ ∗))(treated as regression error). Why or why not?



Q5]

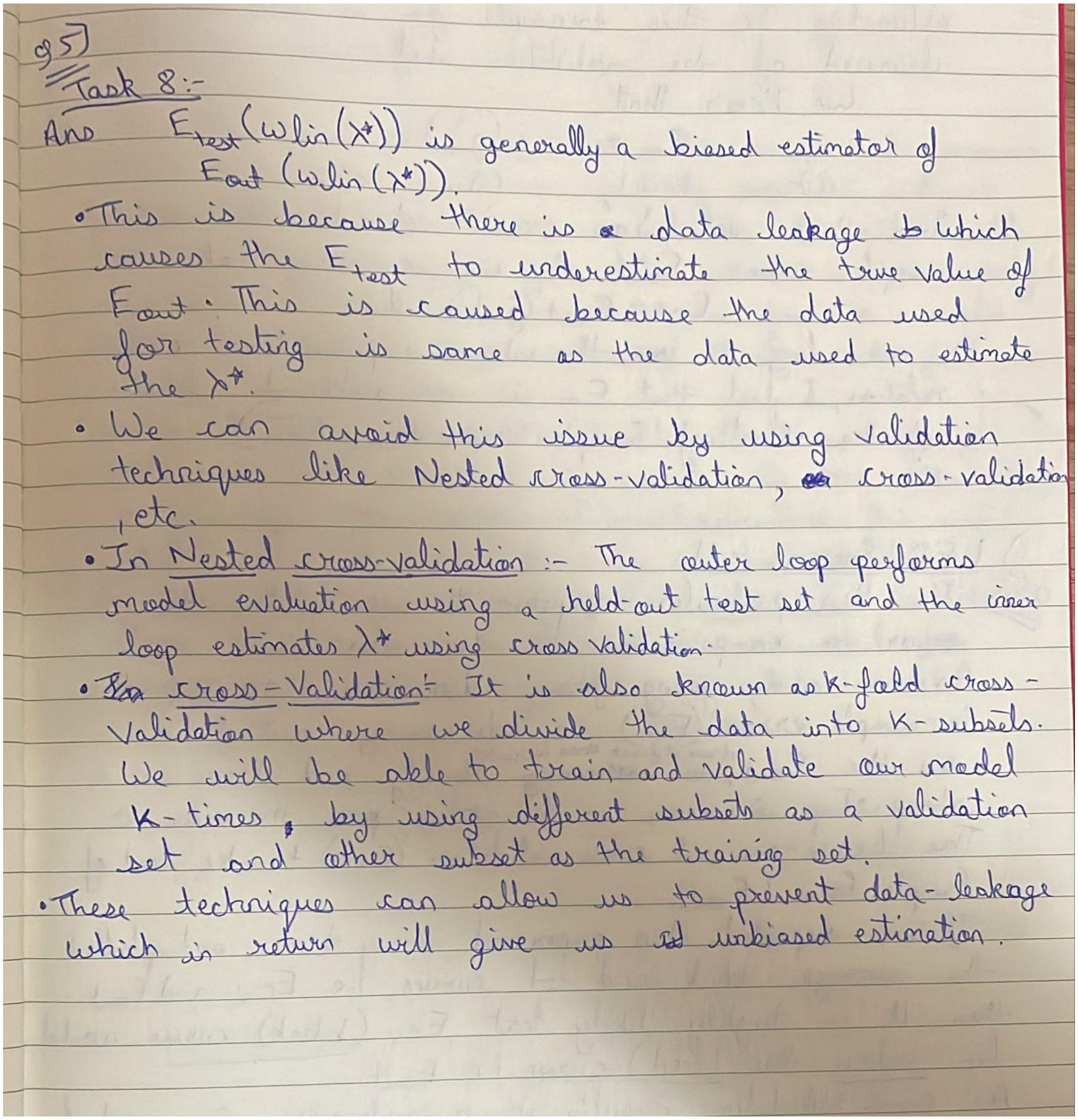
Task 7 :- No Ecv is not biased.

Ecv (λ^*) is an unbiased estimator of E_{test} (wlin(λ^*))

- This is because the cross-validation Error (E_{cv}) is performed on unseen data.
- This helps in getting an unbiased estimation of the model.
- E_{test} (wlin(λ^*)) targets to estimate the model's performance on unseen data.
- The cross-validation Error (E_{cv}) is able to estimate model on unseen data because they make use of different techniques like Averaging, Randomization and data split.
- As in cross-validation we split the data into K-subsets and randomly use any subset to validate while other subsets to train. After this process we average the the validation errors obtained for each iteration.
- Because of all this we can prevent data leakage which in return will act as an unbiased estimator.

Hence $E_{cv}(\lambda^*)$ is an unbiased estimator of E_{test}(wlin(λ^*))

8. **Data snooping**. Etest(wlin(λ ∗)) an unbiased estimator of Eout(wlin(λ ∗)) (treat them as classification errors)? Why or why not? If not, what could we do differently to fix things, so that it is? Explain.



q5]

Task 8:-

Ans $E_{test}(wlin(\lambda^*))$ is generally a biased estimator of $E_{out}(wlin(\lambda^*))$.

- This is because there is data leakage which causes the E_{test} to underestimate the true value of E_{out}. This is caused because the data used for testing is same as the data used to estimate the λ^*.
- We can avoid this issue by using Validation techniques like Nested cross-validation, cross-validation, etc.
- In Nested cross-validation :- The outer loop performs model evaluation using a held-out test set and the inner loop estimates λ^* using cross Validation.
- cross-Validation:- It is also known as K-fold cross-validation where we divide the data into K-subsets. We will be able to train and validate our model K-times, by using different subsets as a validation set and other subset as the training set.
- These techniques can allow us to prevent data-leakage which in return will give us unbiased estimation.

Access Link

HarsimranSinghDhillon_HW_3.ipynb