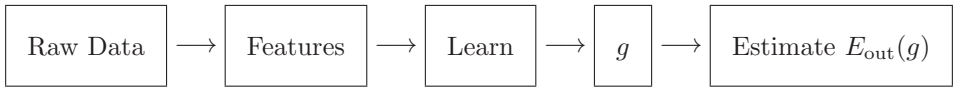
ASSIGNMENT 9

LFD is the class textbook

End to End Learning with Regularization and Validation

In this assignment, you will address the learning problem for predicting 'Digit 1' from 'not Digit 1'.



You will use all the digits available on the class web site and your two features that you constructed in an earlier assignment. $y_n = +1$ if the data point is 'Digit 1' and $y_n = -1$ otherwise.

1. Combine the training and test data into a single data set.
2. Use your feature construction algorithm to create two features for each data point.
3. **Normalize your features** as follows. For each data point, shift the first feature by a shift and then rescale the first feature by a scale. You must determine the shift and scale so that the minimum feature value is -1 and the maximum feature value is $+1$. Repeat this for your second feature. So now you have a new pair of features where each feature (for every data point) is in the range $[-1, 1]$. This process is an example of *input normalization*.
4. *Randomly* select 300 data points for your data set $\mathcal{D}$. Put the remaining data into a test set $\mathcal{D}_{\text{test}}$.
5. Do not look at $\mathcal{D}_{\text{test}}$ until you have created your final hypothesis g and are ready to estimate $E_{\text{out}}(g)$.

We will treat this problem as a regression problem with real valued targets ± 1 until it is time to output our final classification hypothesis. At that point our final classification function will be $\text{sign}(g)$.
The polynomial feature transform is

$$(x_1, x_2) \to (1, x_1, x_2, x_1^2, x_1x_2, x_2^2, x_1^3, x_1^2x_2, x_1x_2^2, x_2^3, x_1^4, x_1^3x_2, \ldots).$$

As discussed in class, this would create 'non-orthogonal' features, which can cause a problem for algorithms like the pseudo-inverse regression algorithm if the columns in the data matrix become too dependent. An 'orthogonal' polynomial transform is given by

$$(x_1, x_2) \to (1, L_1(x_1), L_1(x_2), L_2(x_1), L_1(x_1)L_1(x_2), L_2(x_2), L_3(x_1), L_2(x_1)L_1(x_2), \ldots),$$

where in each feature we replace x_i^k with $L_k(x_i)$ and $L_k(\cdot)$ is the kth order Legendre transform. See Problem 4.3 in LFD for a recursive expression that defines the Legendre polynomials from which you can develop a quick algorithm to compute the $L_k(x)$ for any input x.
We will use the pseudo-inverse linear regression algorithm with weight decay regularization for learning, which corresponds to minimizing $E_{\text{aug}}(\mathbf{w}) = E_{\text{in}}(\mathbf{w}) + \lambda \mathbf{w}^{\mathrm{T}}\mathbf{w}$, and $E_{\text{in}}(\mathbf{w})$ is the sum of squared errors. Recall that the learned linear regression weights are

$$\mathbf{w}_{\text{reg}}(\lambda) = (\mathrm{Z}^{\mathrm{T}}\mathrm{Z} + \lambda \mathrm{I})^{-1}\mathrm{Z}^{\mathrm{T}}\mathbf{y},$$

where $\mathrm{Z} \in \mathbb{R}^{N \times \tilde{d}}$ is the matrix of transformed features and $\mathbf{w}_{\text{reg}} \in \mathbb{R}^{\tilde{d}}$ is the regularized weight vector.

Do the following problems to end up with your final hypothesis g.

1. **(100) 8th order Feature Transform.**
Use the 8th order Legendre polynomial feature transform to compute Z. What are the dimensions of Z?

2. **(100) Overfitting.**
Give a plot of the decision boundary for the resulting weights without any regularization ($\lambda = 0$). Do you think there is overfitting or underfitting?

3. **(100) Regularization.**
Give a plot of the decision boundary for the resulting weights with $\lambda = 2$. Do you think there is overfitting or underfitting?

4. **(200) Cross Validation.**
Use leave-one-out cross validation to estimate $E_{\text{cv}}(\lambda)$ for

$$\lambda \in \{0, 0.01, 0.02, \ldots, 2\}.$$

The upper limit 2 is arbitrary – use your judgement to pick one that is reasonable (you may need to increase λ_{max}, depending on your features). $E_{\text{cv}}(\lambda)$ estimates E_{out} for regularization parameter λ, when learning with $N-1$ data points. You can use the formula in Equation (4.13) of LFD to efficiently compute $E_{\text{cv}}(\lambda)$.

Plot $E_{\text{cv}}(\lambda)$ versus λ, and on the same plot, $E_{\text{test}}(\mathbf{w}_{\text{reg}}(\lambda))$ (the regression error on the test set).

Comment on the behavior of E_{cv} in relation to E_{test}.

5. **(100) Pick λ.**
Use the cross validation error to pick the best value of λ, call it λ^*. Give a plot of the decision boundary for the weights $\mathbf{w}_{\text{reg}}(\lambda^*)$.

6. **(100) Estimate E_{out}.**
Use the classification error $E_{\text{test}}(\mathbf{w}_{\text{reg}}(\lambda^*))$ to estimate E_{out} for your final hypothesis g. What is your estimate of E_{out} for separating 'Digit 1' from 'Not Digit 1'?

7. **(100) Is E_{cv} biased?**
Is $E_{\text{cv}}(\lambda^*)$ and unbiased estimate of $E_{\text{test}}(\mathbf{w}_{\text{reg}}(\lambda^*))$? Explain.

8. **(200) Data snooping.**
Is $E_{\text{test}}(\mathbf{w}_{\text{reg}}(\lambda^*))$ an unbiased estimate of $E_{\text{out}}(\mathbf{w}_{\text{reg}}(\lambda^*))$? Explain.
If it is not an unbiased estimate, how can what you did be fixed so that it is?

*[***Hint:** *Where has there been data snooping?. Data snooping occurs if the data used to estimate the performance of g in* **any way** *affected the selection of g as the final hypothesis. When you have figured this one out, you will have understood one of the most subtle forms of data snooping.]*