CSE 446 HW1

Rohan Mukherjee

April 17, 2024

1. (a) Bias of a model arises from picking too specific / the wrong model for predictions. For example, the true model might be quadratic, but we tried to approximate it with a linear function instead. Variance arises from having too much complexity in the model, where we will overfit each data set, so when we take an expected value, we get huge deviation from the mean since each data set is different and will yield a function that is far from the mean.

   (b) As discussed above and in class, with more complexity we can get closer to the true model, so bias would go down, but we would start overfitting each dataset, so the variance goes up.

   (c) No. As we collect more data, our learned function will get closer to the theoretically optimal function (which is the expected value), so the variance should go down.

   (d) The validation set should be used for hyperparameter tuning. The training set should be used for trianing the model, of course, and the test set should be used to determine the error for the model. Thus, the validation set is best used for hyperparameter tuning. A good procedure for hyperparemeter tuning is cross-validation (which is where the name validation set comes from), where you chunk the training data into k parts, and take 1 of those k parts as the validation data, and use these as the new training data and the validation data as the test data to determine good values of the hyperparameters.

   (e) Since can have bias, the model will tend to overfit the training data, and hence the true error of the training function will beh higher.

2. (a) Given the x_i, the probability of seeing the x_i in that order is given by the following likelihood function:

$$\mathcal{L}(\lambda) = \prod_{i=1}^{n} e^{-\lambda} \frac{\lambda^{x_i}}{x_i!}$$

   Taking the log gives,

$$\log \mathcal{L}(\lambda) = \sum_{i=1}^{n} -\lambda + x_i \log \lambda - \log x_i!$$

The derivative of this function is just

$$\frac{\mathrm{d}}{\mathrm{d}\lambda} \log \mathcal{L}(\lambda) = \sum_{i=1}^{n} -1 + \frac{x_i}{\lambda}$$

Setting this to 0 yields $\lambda = \frac{1}{n} \sum_{i=1}^{n} x_i$.

(b) From this, we can see that the numerical estimate for λ for the first 5 games is just

$$\lambda = \frac{1}{5}(3 + 7 + 5 + 0 + 2) = 3.4$$

Thus, the probability that the Reign score exactly 4 goals in the next game is just

$$\mathbb{P}(4 \mid \lambda) = e^{-3.4} \frac{3.4^4}{4!} \approx 0.186$$

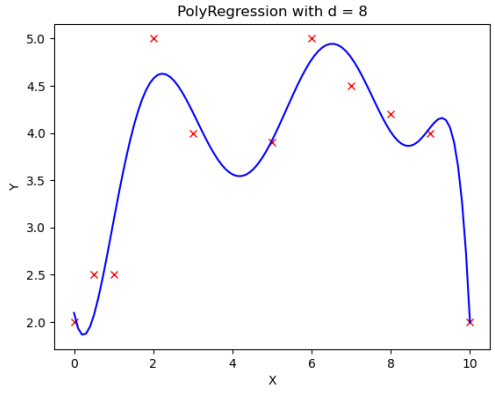
(c) Given that Reign actually scored 8 goals in their last game, the updated λ is just

$$\lambda = \frac{1}{6}(3 + 7 + 5 + 0 + 2 + 8) = \frac{25}{6}$$

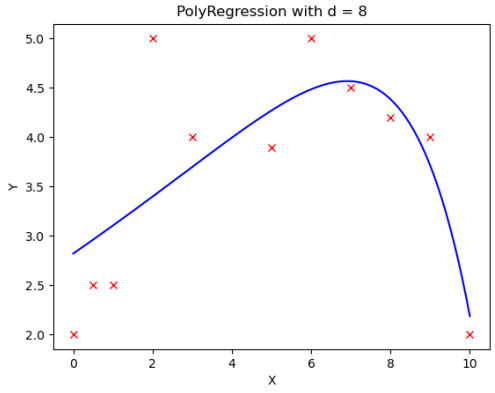
Now, the probability that Reign scores 5 goals in their next game is just

$$\mathbb{P}(5 \mid \lambda) = e^{-4.1667} \frac{4.1667^5}{5!} \approx 0.162$$

3. (b) Here are the plots:



(a) Plot with no regularization



(b) Plot with regularization $\lambda = 1$

Figure 1: Comparison of plots

As you can see, when you increase the regularization, the model tends to look less complex / overfit. For example, on the left the model is trying extremely hard to overfit the 12 data points, but on the right it looks like it took a more averaging approach. Realistically, small changes in the input should not lead to large changes in the output, like in figure (a).

4. Here are the learning curves:

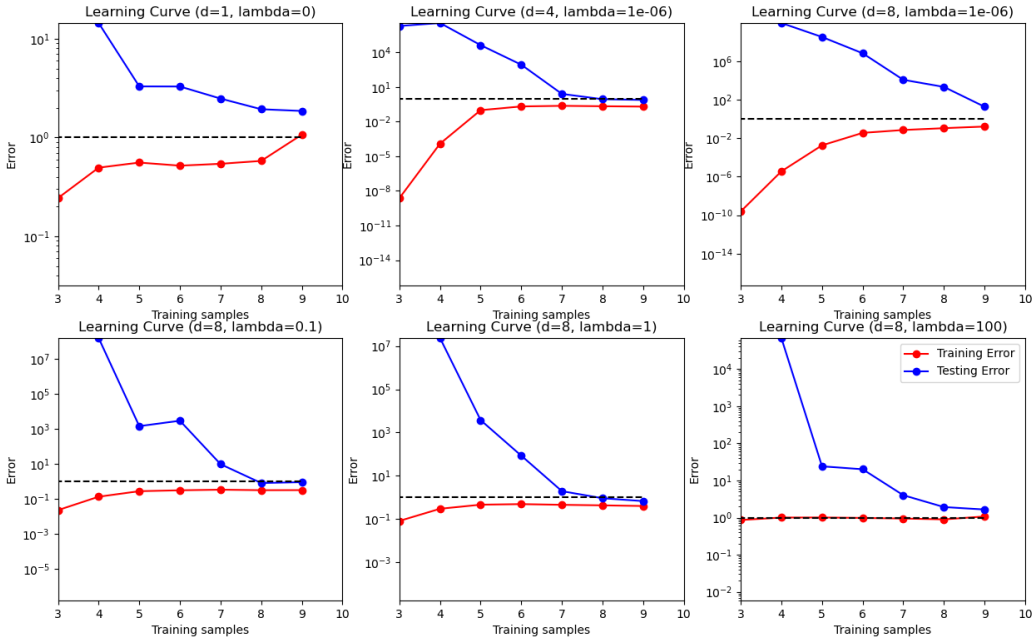


Figure 2: Learning curve for the model

5. (a) By definition, $\|W\|_F^2 = \sum_{i,j} w_{ij}^2$. We see then that $\frac{\partial}{\partial w_{ij}}\|W\|_F^2 = 2w_{ij}$, which shows that $\nabla_W \|W\|_F^2 = 2W$. Per the hint, we know that

$$\sum_{i=1}^{n} \|W^T x_i - y_i\|_2^2 = \sum_{j=1}^{k} \|Xw_j - Ye_j\|_2^2$$

We see now that, if $f(W) = \sum_{i=1}^{n} \|W^T x_i - y_i\|_2^2$,

$$\begin{aligned}
\nabla_{w_j} \sum_{j=1}^{k} \|Xw_j - Ye_j\|_2^2 &= \nabla_{w_j} \|Xw_j - Ye_j\|_2^2 \\
&= \nabla_{w_j} (Xw_j - Ye_j)^T (Xw_j - Ye_j) \\
&= \nabla_{w_j} w_j^T X^T X w_j - 2e_j^T Y^T X w_j \\
&= 2X^T X w_j - 2X^T Y e_j
\end{aligned}$$

That is, the jth column of $\nabla_W f(W)$ is $2X^T Xw_j - 2X^T Ye_j$, so $\nabla_W f(W) = 2X^T XW - 2X^T Y$. Thus the gradient of $g(W) = F(W) + \lambda\|W\|_F^2$ is just $2X^T XW - 2X^T Y + 2\lambda W$. Setting this to zero shows that $(X^T X + \lambda I)W = X^T Y$, which shows that $\widehat{W} = (X^T X + \lambda I)^{-1} X^T Y$.

(b) The training error I got was 14.805% and the test error was 14.66%.

(c) Here is the plot of 10 samples:

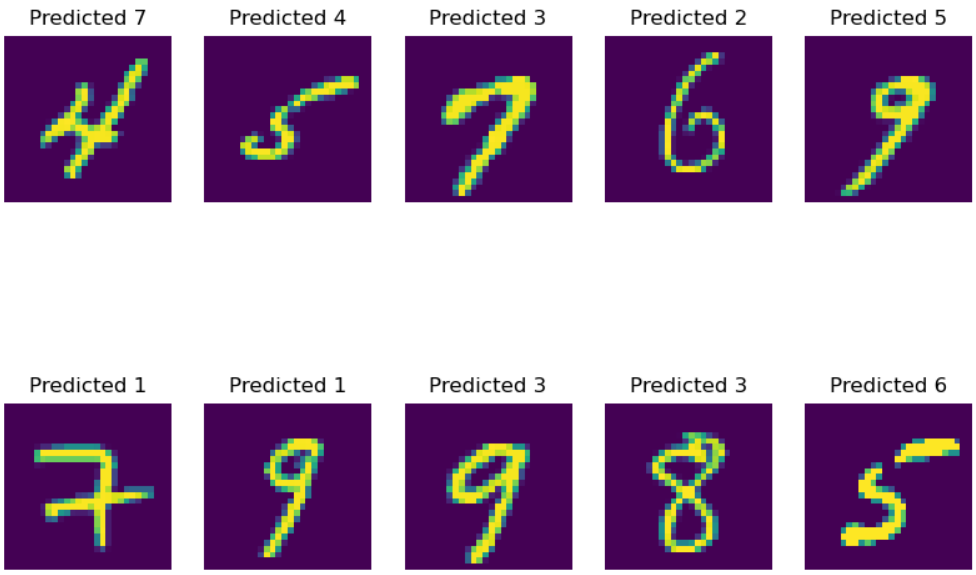


Figure 3: 10 samples of wrong numbers

The big pattern I notice is that most of the numbers that it wrongly predicted seemed to be tilted left or right or disconnected. After looking at the numbers the model got right, most of these features did not show up in those samples.

6. This assignment took me around 10 hours.