Assignment 4

Mushfika Rahman

04/21/2022

Written Problems

1.1 Show the mathematics to minimize the given Lagrangian formula for ridge regression. You can show the gradient descent updates, or you can show how to compute it in one shot.

→ Ridge:

$$\min_w \frac{1}{n}\sum_{i=1}^{n}(h(x_i)-y_i)^2+\frac{\lambda}{n}\sum_{i=1}^{d}(w_i)^2$$

So,

$$l(w)=\frac{1}{n}\sum_{i=1}^{n}(h(x_i)-y_i)^2+\frac{\lambda}{n}\sum_{i=1}^{d}(w_i)^2$$

$$=\frac{1}{n}\sum_{i=1}^{n}(w^Tx_i-y_i)^2+\frac{\lambda}{n}\sum_{i=1}^{d}(w_i)^2$$

$$=\frac{1}{n}\sum_{i=1}^{n}\|(w^Tx_i-y_i)\|_2^2+\frac{\lambda}{n}\sum_{i=1}^{d}\|(w^Tx_i-y_i)\|_2^2$$

Now,

$$\|(w^Tx_i-y_i)\|_2^2=\begin{pmatrix}w^Tx_1-y_1, w^Tx_2-y_2, ..., w^Tx_m-y_m\end{pmatrix}\times\begin{pmatrix}w^Tx_1-y_1\\ w^Tx_2-y_2\\ \vdots\\ w^Tx_m-y_m\end{pmatrix}$$

so we can write

$$l(w)=\frac{1}{n}(w^TX^T-Y^T)(Xw-Y)+\frac{\lambda}{n}w^Tw$$

$$=\frac{1}{n}(w^TX^TXw-Y^TXw-w^TX^TY+Y^TY+\lambda w^Tw)$$

$$=\frac{1}{n}(w^TX^TXw-2w^TX^TY+Y^TY+\lambda w^Tw)$$

To minimize the $l(w)$ function we take the partial derivative,

$$\frac{\partial l(w)}{\partial w} = \frac{2}{n}(X^T X w - X^T Y + \lambda I w)$$

we set, $\frac{\partial l(w)}{\partial w} = 0$ Then,

$$\frac{2}{n}(X^T X w - X^T Y + \lambda I w) = 0$$

$$X^T X w - X^T Y + \lambda I w = 0$$

$$w = (X^T X + \lambda I)^{-1} X^T Y$$

1.2 Use ridge regression to fit (regularized) polynomials of order 10

→ Pseudo-inverse was used to train Ridge regression weights. The number of data points were 100. Train and test set was split into 80-20 ratio. I have also generated a different test set from the same distribution. In here, the features was big enough and data points are bit less in number ,taking small dataset since regularization is most effective with small data points.

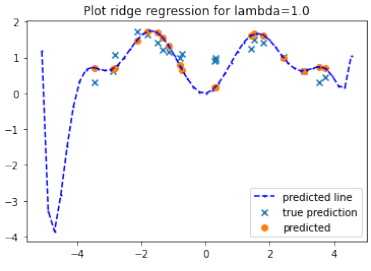

Figure 1: ridge regression fit in terms of big test data

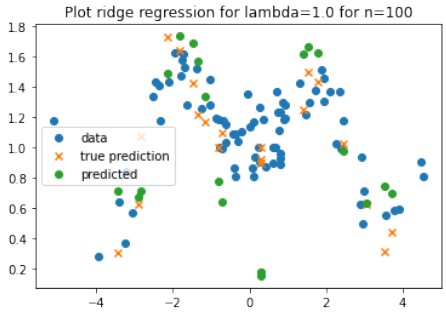

Figure 2: ridge regression fit

1.3 Plot the in-sample and out-of-sample error as a function of . Compare the 'sparseness' of the weights obtained (a 'sparse' solution means many weights are zero, or perhaps close to zero).

→ I experimented with different lambda values to observe the effect of in-sample and out-of-sample errors over lambda. Both in-sample and out-of-sample errors were calculated by taking the mean squared error for

the train and test dataset. Though in-sample error increased with the increasing lambda,out-of-sample error decreased. Since regularization significantly reduces the variance of the model without a substantial increase in its bias. However, while experimenting with a huge lambda value, the out-of-sample also increases. This is because the model starts giving rise to bias and resulting in underfitting.

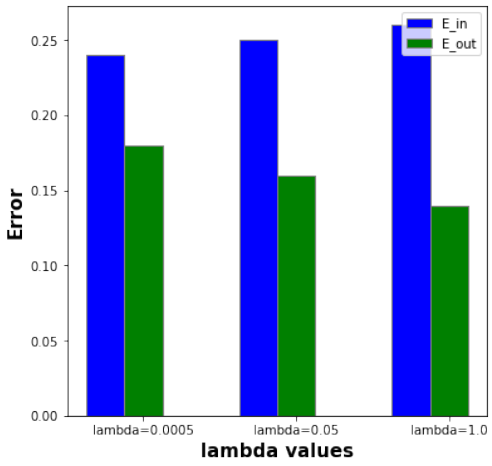

Figure 3: Ein, Etest by varying lambda

With different λ , we can variation in the model parameters, regularization is working well by reducing the magnitudes of the some model parameters. However, when the λ is increased to a very high value the model shrinks all the coefficients thus model looses the ability to predict well on the unseen data.

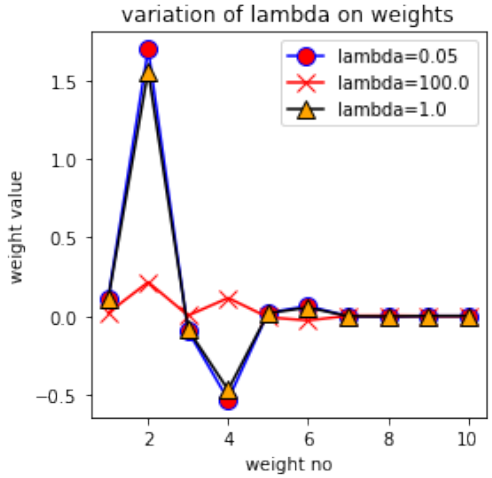

Figure 4: weight sparseness

2.Implement both non-parametric and parametric RBFs.In the parametric case, vary k, and place the centers at regular intervals in the input space. For both parametric and non-parametric models, vary r. Show how both k and r affect the shape of the fit and Etest. How do the book's suggestions (rules of thumb) for k and r work for you?

→ To Observe the effect of r in the non-parametric radial basis function; experimentation was done with different values for r in the non-parametric radial basis function. The dataset was split into 80-20 ratios.

For measuring the out-of-sample error, the mean squared error was calculated. In the non-parametric rbf the E_{test} increases with the increasing r. The model was making more error in predicting in unseen data with the increasing r, and with decreasing value of r(less than reported, the error was very large and cannot be so in the plot) the model was making significant errors on test set prediction. When r is too small the model was rigid thus test error was high, r is too big the model is too flexible thus test error was increasing.

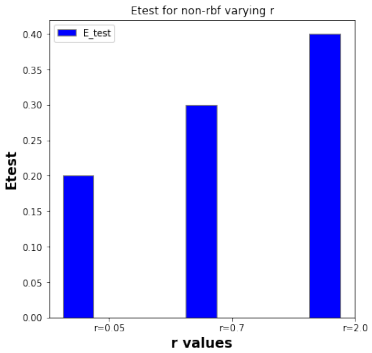

Figure 5: Etest in non-parametric r

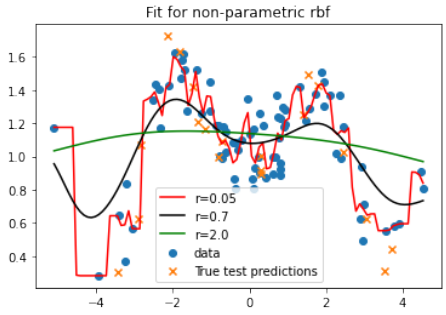

Figure 6: non-parametric fit for different r

I experimented with different values for r and k in the parametric rbf for 100 data points. I split the dataset into an 80-20 ratio. To calculate the out-of-sample error, I used the mean squared error. When I increased the r and k, the E_{test} also decreased for some point. The E_{test} starts to increase when the k and r value is increased too much. The rule of thumb gave the best result in both the cases of r and k in terms of small E_{test}. According to the rule of thumb for $r = \frac{R=\max_{i,j}\|(x_i - x_j)\|}{k^{1/d}}$. In my experiment the rule of thumb would result $r \approx 5$,$r = 5.0$ gave the lowest error in test set.

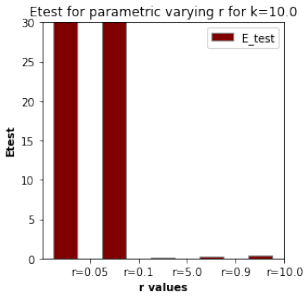

Figure 7: paramteric rbf Etest with varying r

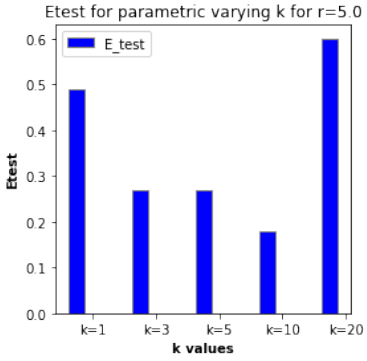

Figure 8: paramteric rbf Etest with varying k

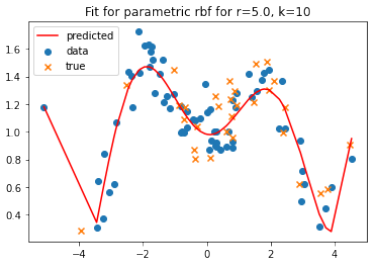

Figure 9: paramteric rbf fit

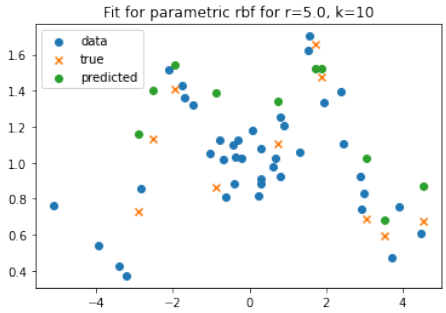

Figure 10: fit for parametric rbf

3.1 Run the condensed nearest neighbor (CNN) algorithm for 3-NN on the digits data. Use all the data for classifying "1" versus "not 1".Set N = 500 and randomly split your data into a training set of size N and use the remaining data as a test/validation set.

→ The experiment was conducted by testing the feature set data file. I converted the text file to a data frame and performed prepossessing on the data, such as converting to float and integer. For the target values, I converted the values to -1, which is not 1.

3.2 Use all your training data and evaluate performance of E_{in} and E_{test} for 3-NN

→ In this experiment, E_{in} was calculated by taking the mean squared error for the training set. E_{test} was calculated similarly for test set. The E_{test} was more than E_{in} but did not deviate largely from E_{in} even though the test set was nearly two times the training set.

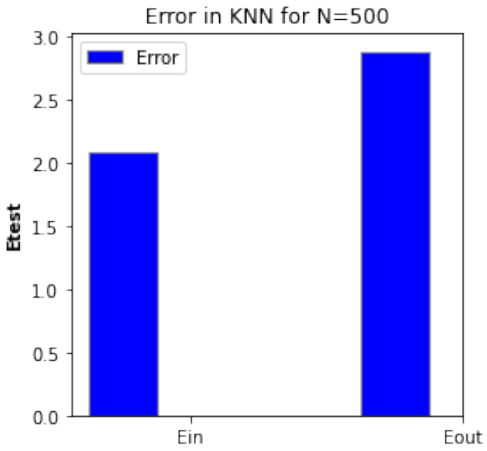

Figure 11: Error for knn

3.3 Apply the CNN algorithm to condense data. Evaluate performance of E_{in} and E_{test} for 3-NN.

→ In my experiment, to condense data, I used CNN, and my choice of the number of nearest neighbors was three. After applying 3-NN, I measured E_{in} and E_{out} by calculating the mean squared error for training and testing data. The E_{in} remained the same for the 3-NN's E_{in} , because the experiment aimed for training set consistency. However, the E_{out} varied in the CNN algorithm; it was greater than 3-NN's E_{out}. Since the training was on fewer examples implementing cnn, the number of misclassification increased.

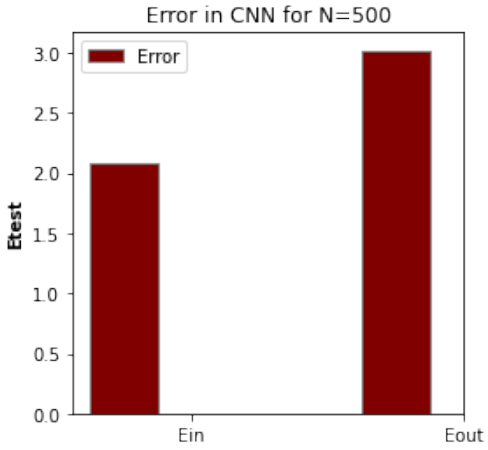

Figure 12: Error for cnn

3.4 Repeat (2) & (3) using 1000 random training-testing split. Report average E_{in} and E_{test}.

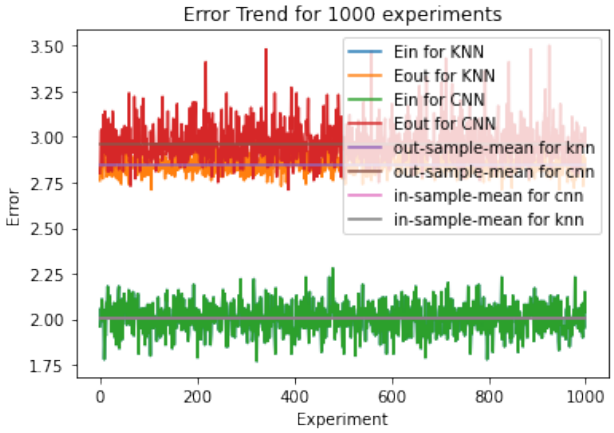

Figure 13: Comparison of knn and cnn