CIS 520, Machine Learning, Fall 2020
Homework 3
Due: Monday, October 5th, 11:59pm
Submit to Gradescope

Sheil Sarda, Yuezhong Chen

1 Regularization Penalties

Assume bias term in the regression is zero. Assume $\lambda = 1$.

1. Assume that the response variable is distributed according to $y_i \sim N\left(w \cdot x_i, \sigma^2\right)$ (no regularization penalty is needed). MLE estimate $\widehat{w}_{MLE}$ of w: $(X^T X)^{-1} X^T Y$
   `[0.88914862, -0.82601591, 4.19023612]`
2. $\widehat{w}$ for $p = 2$. $w = (X^T X + I)^{-1} X^T Y$
   `[0.86455156, -0.82104237, 4.12186079]`
3. $\widehat{w}$ for $p = 1$. Using `sklearn`'s Lasso regression tool:
   `[0.875, -0.818, 4.182]`
4. Considering all cases where different components of w are set to 0.

X_1	X_2	X_3	$\widehat{w}$
o	o	o	3108
o	o		4136
o		o	3138
o			4225
	o	o	3131
	o		4160
		o	3171
			12800

5. Part 1 of this question (OLS) provides a baseline $\widehat{w}$, which is improved upon by L_1 and L_2 regression since it adds a penalty proportional to the magnitude of the weights. L_1 was the most effective in shrinking the errors.
6. Exploring the tradeoff between minimizing the sum of squared errors and the magnitude of $\widehat{w}$.
   (a) $\|\widehat{w}_{MLE}\|_2^2 \,/\, \|y - X\widehat{w}_{MLE}\|_2^2 = 0.006$
   (b) i. Impact on $\|y - X\widehat{w}_{MLE}\|_2^2$ when going from N to $2N$ samples will be proportional, since we expect training error to increase. As N grows, it eventually hits an asymptote where the linear model is fitting the underlying distribution as well as possible. At this point, training error no longer depends on the number of samples N.
   ii. Impact on $\|\widehat{w}_{MLE}\|_2^2$ when going from N to $2N$ samples cannot be mathematically modeled accurately, since it will grow with N within some range, then start shrinking towards an optimal $\widehat{w}$ as the linear model reaches its optimal performance. At this asymptote, $\widehat{w}_{MLE}$ too will no longer depend on the number of samples N.

(c) To find λ for which the estimate $\widehat{w}$ satisfies: $0.8 < \|\widehat{w}\|_2^2 / \|\widehat{w}_{MLE}\|_2^2 < 0.9$, we use trial and error. $\lambda = 5$ satisfies this equation.

2 Feature Selection

OLS regression with L_0 regularization is expressed as: $\text{argmin}_w \|y - Xw\|_2^2 + \lambda\|w\|_0$

	x_1	x_2	x_3	y
Sample 1	1	2	1.3	1
Sample 2	2	1	7.3	9
Sample 3	1	1	2.8	4

1. Streamwise regression: for each round, w is computed based on all the features in the current round model under OLS only without regularization, and then Err is computed based on that w.

   (a) Add each feature in the order of x_1, x_2, x_3. Assume $\lambda = 0.2$ and apply L_0 regularization. Error of models:

   - $Err_0 = 1^2 + 9^2 + 4^2 = 98$
   - Try to add x_1 into the model. $Err_1 = 10.03$.
     Coefficients: `[3.83, 0, 0]`
   - Try to add x_2 into the model. $Err_2 = 0.76$.
     Coefficients: `[5.73, -2.27, 0]`
   - Try to add x_3 into the model. $Err_3 = 0.6$.
     Coefficients: `[50, -18, -10]`

   (b) Add each feature in the order of x_3, x_2, x_1. Assume $\lambda = 0.2$ and apply L_0 regularization. Final selected feature(s): x_3.
   $Err_3 = 0.86$
   Coefficients: `[0, 0, 1.25]`

   (c) From the above parts, we learn that x_3 alone provides performance comparable to x_1 and x_2 combined (0.76 vs 0.86), but the overall error of the second model is higher because Streamwise regression rejects x_1 and x_2 to prevent overfitting at the cost of in-sample performance.

2. Stepwise regression: for each round, w is computed based on all the features in the current round model under OLS only without regularization, and then Err is computed based on that w.

   (a) Initial $Err_{old} = 1^2 + 9^2 + 4^2 = 98$.

   (b) Try adding each feature respectively and compute their Err:

   - $Err_1 = 10.03$.
     Coefficients: `[3.83, 0, 0]`
   - $Err_2 = 60.70$.
     Coefficients: `[0, 2.50, 0]`
   - $Err_3 = 0.86$.
     Coefficients: `[0, 0, 1.25]`

   (c) The feature to add to the model is x_3. The associated error is 0.86.

   (d) Repeat adding each feature respectively with computed Err. Pick features to add to the model and report when to halt.

   - Adding features x_1 and x_3: $Err = 0.88$.
     Coefficients: `[-0.64, 0, 1.44]`

- Adding features x_2 and x_3: $Err = 0.86$.
  Coefficients: `[0, -0.24, 1.30]`

Since both Err values in the above cases is greater than Err_{old}, we halt the stepwise regression here.

(e) Final selected feature(s) is x_3.

(f) Pros of streamwise regression:

- When space of potential features is large, overfitting can be controlled by dynamically adjusting the threshold for adding features.
- All features do not have to be known in advance, enabling dynamic feature generation.
- Less computationally intense when dealing with large datasets with many features.

Cons of streamwise regression:

- Because the method adds variables in a certain order, the combination of predictors in the final model is determined by that order.

3 Kernel Regression

1. Build the model. (auto-graded only)
2. Analysis of the model.

   Figures:

Figure 1: $\sigma = 0.01$

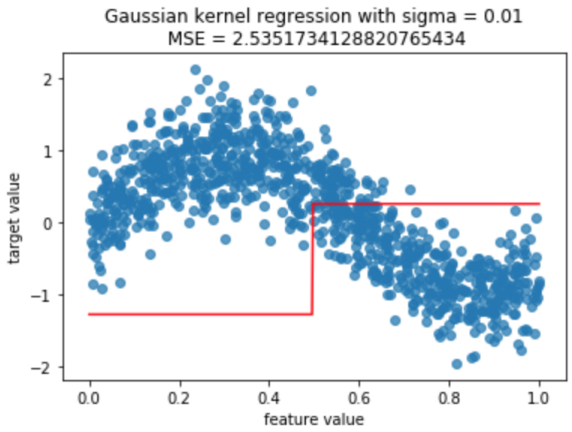


Figure 2: $\sigma = 0.05$

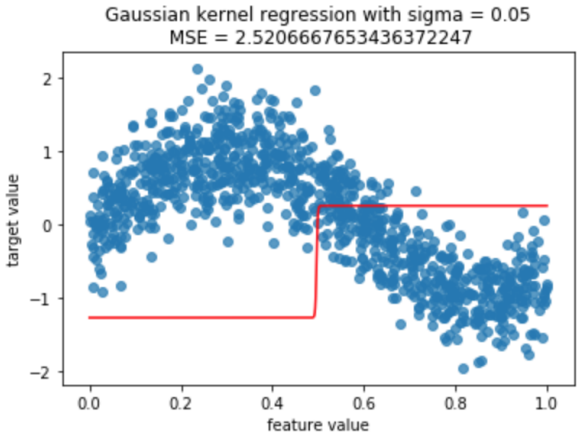


Figure 3: $\sigma = 0.1$

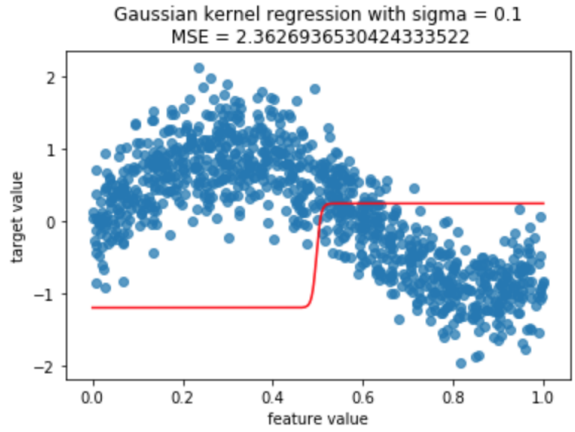


Figure 4: $\sigma = 0.15$

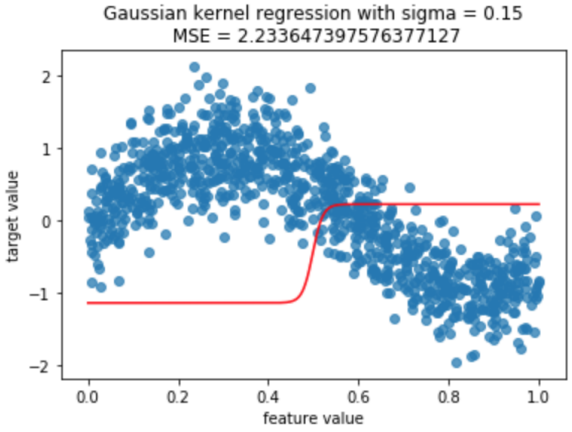


Figure 5: $\sigma = 0.2$

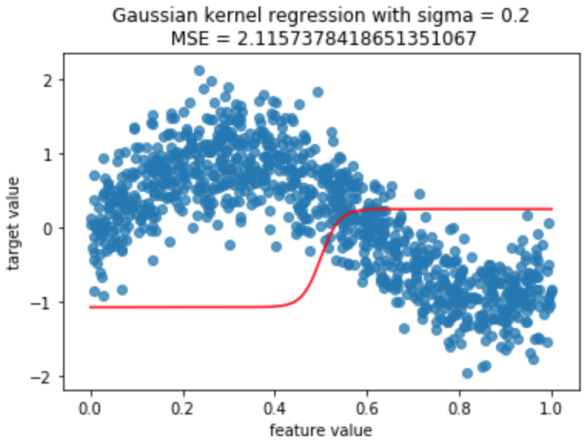


Figure 6: $\sigma = 0.5$

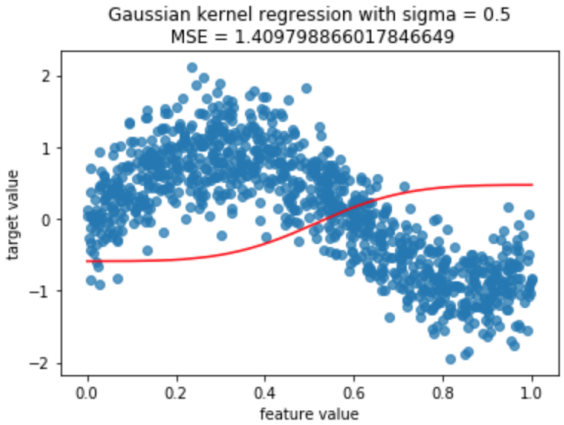
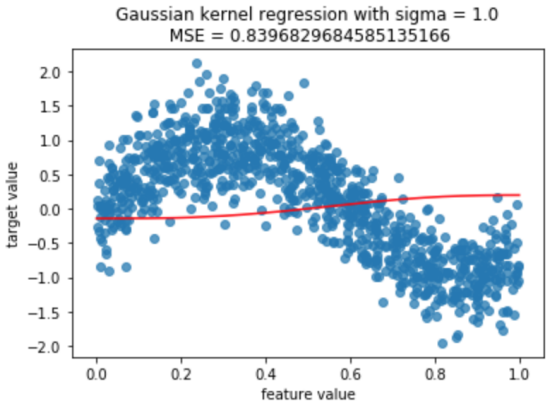


Figure 7: $\sigma = 1.0$



Smallest MSE: 0.83968

Smallest MSE Sigma Value: 1.0

As sigma value is increasing the MSE is decreasing. As from the plots it is observed that training and predicted values are non-linear in nature, the trend is well captured as the sigma increases as it adds more non-linearity.

Learning curve of the model is improving with the increase of sigma value and error is decreasing.

4 Gradient Descent on Logistic Regression

1. Accuracy on the test set: 0.77

   Logistic regression coefficient(Scikit): [1.30299939e-04 1.76132834e-06 -1.30923960e-06 -3.08981964e-04 -3.31896546e-04 5.17574298e-04 6.98099943e-07 -3.10110604e-07 -3.23366122e-07 -3.70267292e-04 -1.84436862e-07]

2. Log Likelihood:

$$-LL = -\frac{1}{N}\sum_{n=1}^{N} y_n(log(h_w(x_n))) + (1-y_n)(log(1-h_w(x_n)))$$

   Derivative:

$$dw = \frac{1}{N}\sum_{n=1}^{N}(h_w(x_n) - y_n)x_n$$

   Update formula:

$$w = w - \frac{\eta}{N}\sum_{n=1}^{N}(h_w(x_n) - y_n)x_n$$

   η: learning rate; N: the size of training data.

3. Accuracy on the training set: 0.76375

   Accuracy on the test set: 0.7675

   Logistic regression coefficient (SGD): [5.67051726e-03 5.21208621e-02 -1.10211910e-02 -2.57983118e-01 -9.05115041e-01 3.51256199e+00 1.55306123e-02 -1.54856541e-02 -6.68514118e-03 -1.74995599e+00 -8.72165559e-04]

4. Accuracy on the training set: 0.82375

   Accuracy on the test set: 0.815

   Logistic regression coefficient (AdaGrad): [8.56662029e-05 7.32614095e-03 -4.06503117e-03 -1.84270727e-04 -1.05933198e-03 4.09573976e-03 7.04325651e-03 -7.02782768e-03 -6.96842728e-03 -2.22061369e-03 -2.21480516e-03]

5. The AdaGrad algorithm has better accuracy on the training set. Because of the feature which is frequently learned, the learning rate of this feature is small. Because the sum of the gradient at each step is very large and the learning rate will be comparably large. Thus, the AdaGrad algorithm learn slowly but pay more attention to rare and informative feature.

Figure 8: Accuracy vs. iteration for SGD and AdaGrad (2 points)