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

Chun Chang

1 Regularization Penalties

1.

$$
\begin{aligned}
LL(P(D \mid wx_i, \sigma^2)) &= \log(\prod_{i=1}^{N} \frac{1}{\sqrt{2\pi^p}\Sigma} exp\frac{-(y_i - wx_i)^T\Sigma^{-1}(y_i - wx_i)}{2}) \\
&= -\frac{N}{2}log(2\pi) + \frac{N}{2}log(\Sigma^{-1}) - \frac{1}{2}\sum_{i=1}^{N}(y_i - wx_i)^T\Sigma^{-1}(y_i - wx_i) \\
\frac{\partial LL(P(D \mid wx_i, \sigma^2))}{\partial w} &= \sum_{i=1}^{N}(y_i - wx_i) * x_i \\
0 &= \sum_{i=1}^{N}(y_i - wx_i) * x_i \\
&= Y^TX - wX^TX \\
w^T &= (X^TX)^{-1}X^TY \\
w &= [0.88914862, -0.82601591, 4.19023612]
\end{aligned}
$$

2.

$$
\begin{aligned}
L_2(X) &= (Y - Xw)^T(Y - Xw) + 1 * w^Tw \\
\frac{\partial L_2(X)}{\partial w} &= -(Y - wX)^TX + w^T \\
0 &= -Y^TX - w^TX^TX + w^T \\
w &= (X^TX + I)^{-1}X^TY \\
w &= [0.86455156, -0.82104237, 4.12186079]
\end{aligned}
$$

3.

$$\begin{aligned} L_1(X) &= (Y - Xw)^T(Y - Xw) + 1 * |w|_1 \\ \frac{\partial L_2(X)}{\partial w} &= -(Y - wX)^T X + \frac{\partial |w|}{\partial w} \\ X^T(Y - Xw) &= \frac{\partial |w|}{\partial w} = s \\ s_{i \in \{1, \ldots p\}} &= \begin{cases} s_i = 1 & \text{for } w_i > 0 \\ s_i = -1 & \text{for } w_i < 0 \\ s_i = [1, -1] & \text{for } w_i = 0 \end{cases} \\ w &= [0, -0, 3.45271493] \end{aligned}$$

4.

$$w = [0.88914862, -0.82601591, 4.19023612]$$

5. The MLE model could be used when we know data set is closed to linear and each feature is uncorrelated then we want a unbiased estimator.

   L_0 is good at feature selections but has nothing to do to increase the model's accuracy

   If we know the dataset has so many independent features, we could use L_1 to eliminate the unrelated features. can both tuning the weights also choosing the covariate features If we know the dataset has some correlated features, we'd better use L_2 to find out how important each of the feature is. L_2 is able to shrink the weight but not able to chose features. And L_2 doesn't have the interpretability because it is not able to shrink the weights to zero.

6. (a)

$$\frac{w_{ML}^2}{(Y - Xw_{ML})^2} = 0.006129550450611514$$

   (b) If we have fairly large amount of the data $N >> P$, and the data is closed to linear, the features are uncorrelated, we might have really good estimator of the data.

   So if we double the data, the twice amount of the data are drawn from the same distribution of the previous data set, and we should have new error closed twice of the previous error.

   (c) If the previous assumptions hold true: we have 1.large amount of data set $N >> P$ 2. The features are uncorrelated 3. data is closed to linear 4. Double the training data drawn from the very same distribution, we should have $W_{2*D,MLE} \cong W_{D,MLE}$

   (d)

$$\begin{aligned} \lambda &= 5 \\ \frac{\hat{w}^2}{w_{ML}^2} &= 0.85227984497172 \end{aligned}$$

2 Feature Selection

1. Streamwise regression.

(a)

$$
\begin{aligned}
Err_0 &= 98 \\
w_{err_0} &= [0, 0, 0] \\
\\
Err_1 &= 9.833333333333336 \\
w_{err_1} &= [3.833, 0, 0] \\
\\
Err_2 &= 0.5636363636363637 \\
w_{err_2} &= [5.72, -2.272, 0] \\
\\
Err_3 &= 0.4 \leftarrow \text{Selected} \\
w_{err_3} &= [50, -18, -10]
\end{aligned}
$$

(b)

$$
\begin{aligned}
Err &= 0.6545686087233369 \\
w_{err_0} &= [0, 0, 1.244846]
\end{aligned}
$$

(c) The features that were chosen by different order of streamwise are different. The reason is the order of the streamwise matter, so in (b) we reached local minimum and cannot get it out. For (a) it looks like the three features were somewhat correlated and the order of (a) included all features into the weights

2. Stepwise regression.

(a)

$$Err_{old} = 98$$

(b)

$$
\begin{aligned}
&\text{first round} \\
&\qquad Err_{x_1} = 9.833333333333336 \\
&\qquad Err_{x_2} = 60.5
\end{aligned}
$$

(c)

$$\text{selected} \rightarrow Err_{x_3} = 0.6545686087233369$$

(d)

$$
\begin{aligned}
&\text{second round} \\
&\qquad Err_{x_3,x_1} = 0.6759456481821517 \\
&\qquad Err_{x_3,x_2} = 0.6637573621481243 \\
\text{nothing updated} \rightarrow \text{halt}
\end{aligned}
$$

(e)

$$w = [0, 0, 1.244846]$$

(f) Findings: The stepwise regression is only able to pick one out of multiple correlated features. So the result of stepwise regression was same the the result of reverse order of streamwise because they both met local minimum and were trapped there.

3 Kernel Regression

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

   Figures:

Figure 1: $\sigma = 0.01$

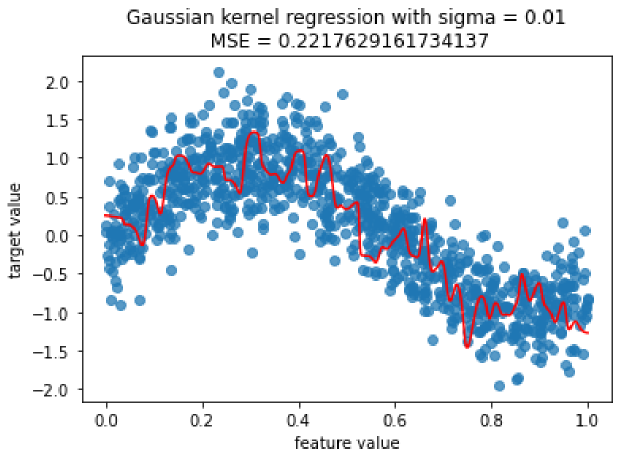


Figure 2: $\sigma = 0.05$

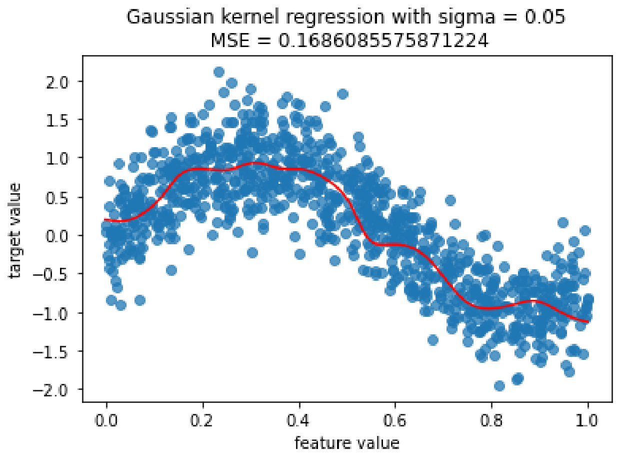


Figure 3: $\sigma = 0.1$

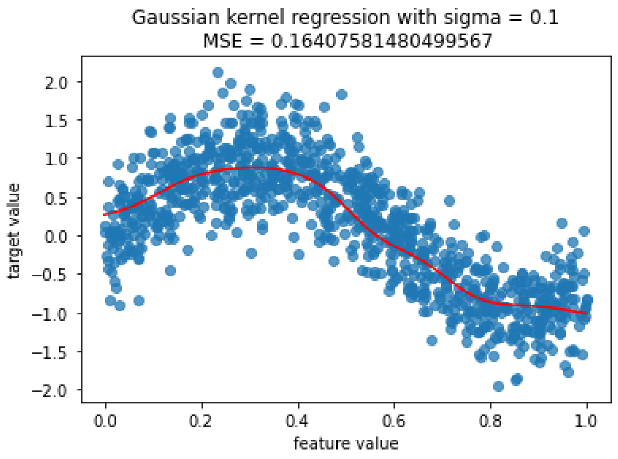


Figure 4: $\sigma = 0.15$

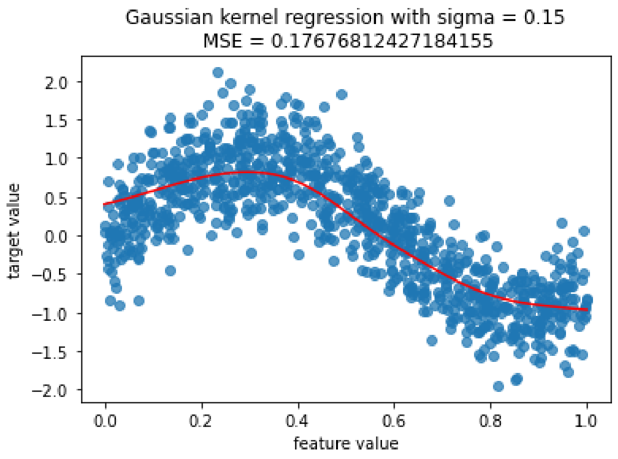


Figure 5: $\sigma = 0.2$

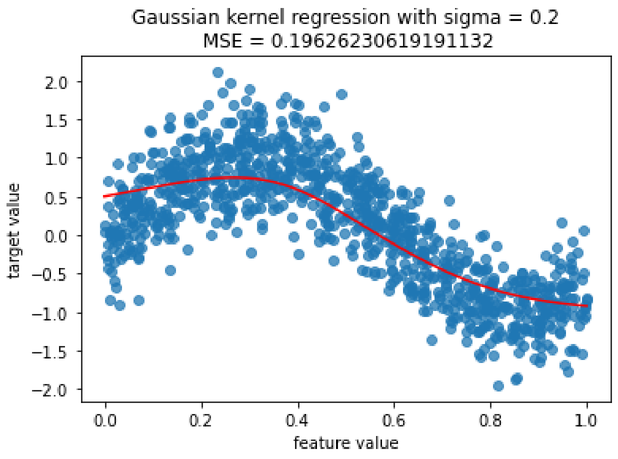


Figure 6: $\sigma = 0.5$

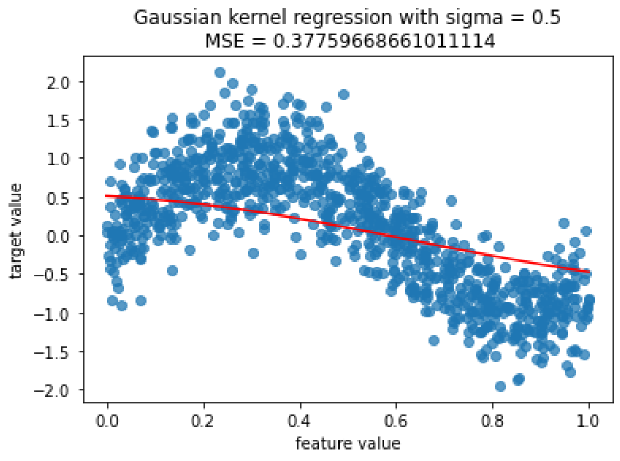
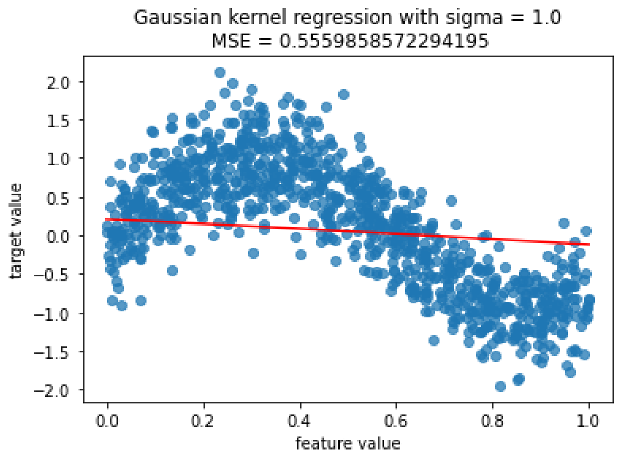


Figure 7: $\sigma = 1.0$



3. Lowest MSE is reached when $\sigma = 0.1$ The σ determines how far is the region we are caring about. The larger the σ is, the bigger the region we consider it is the closed zone to the point and the algorithm averages out lots of points, and lose the Interpretability.

4 Gradient Descent on Logistic Regression

1. Accuracy on the test set:

   Logistic regression coefficient(Scikit): Accuracy : 0.81

Weight:

$$\begin{aligned} w = [&8.77263669e-05, \\ &3.21079507e-05, \\ -&8.33197847e-06, \\ -&6.45709793e-04, \\ -&1.03204937e-03, \\ &3.75013274e-03, \\ &9.91186154e-06, \\ -&7.39026526e-06, \\ -&3.97847369e-06, \\ -&1.54726674e-03, \\ -&7.71020985e-07] \end{aligned}$$

2.

$$\begin{aligned} LL(P(D_Y \mid D_X, w)) &= \log(\prod_{i=1}^{N} h(x_i)^{y_i}(1-h(x_i))^{1-y_i}) \\ &= \sum_{i=1}^{N} y_i log(h(x_i)) + (1-y_i) log(1-h(x_i)) \\ &= \sum_{i=1}^{N} y_i log(\frac{h(x_i)}{1-h(x_i)}) - \sum_{i=1}^{N} log(1-h(x_i)) \\ &= \sum_{i=1}^{N} y_i w^T x_i - \sum_{i=1}^{N} log(1+w^T x_i) \\ \frac{\partial LL(P(D_Y \mid D_X, w))}{\partial w_j} &= \sum_{i=1}^{N} (y_i - h(x_i)) x_{ij} \\ w^{t+1} &= w^t + \eta \nabla \frac{\partial LL(P(D_Y \mid D_X, w))}{\partial w} \end{aligned}$$

3. Accuracy on the training set: 0.77375

   Accuracy on the test set: 0.7625

   Logistic regression coefficient (SGD):

$$\begin{aligned} w = [&5.04998003e-02 \\ &5.19790535e-02 \\ -&1.11979777e-02 \\ -&3.44668763e-01 \\ -&9.92560164e-01 \\ &3.49572109e+00 \\ &1.54843138e-02 \\ -&1.55799008e-02 \\ -&6.73462408e-03 \\ -&1.83915741e+00 \\ -&9.16563871e-04] \end{aligned}$$

4. Accuracy on the training set: 0.81375

   Accuracy on the test set: 0.8075

   Logistic regression coefficient (AdaGrad):

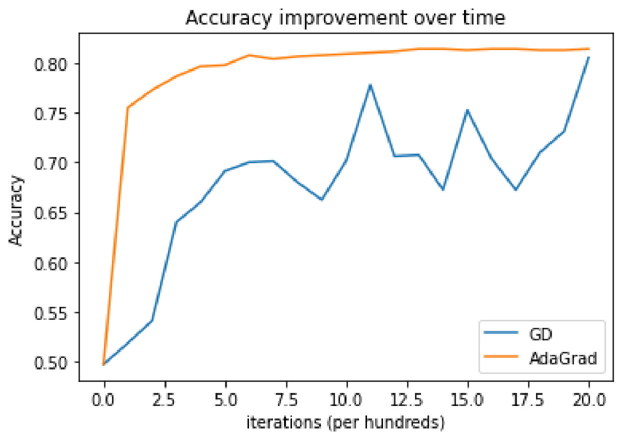
$$\begin{aligned} w = [&9.13157898e-05 \\ &2.45961356e-05 \\ -&6.89098261e-06 \\ -&6.68138807e-04 \\ -&9.56966097e-04 \\ &3.39283484e-03 \\ &7.80913962e-06 \\ -&5.06746258e-06 \\ -&2.98585903e-06 \\ -&1.35440879e-03 \\ -&6.74886189e-07] \end{aligned}$$

5. AdaGrad has better accuracy in the end, since the data in high dimension might not have the same scale, and the AdaGrad provides different learning rate to reach the global minimum faster.

   The cons of AgaGrad method might be the small learning rate caused by the accumulation of the gradient, and it might cause the learning step not directly going into the global minimum. Then the error will be bouncing.

   The SGD has more stable path since the learning rate remains the same during the process and it guarantees good results if we have unlimited training iterations.

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


Accuracy improvement over time
Accuracy
iterations (per hundreds)
GD
AdaGrad