Andrew ElHabr
GT ID: 903888600
ISyE 6740
11/17/17
Are you required to do the extra problem? YES

Homework 4

1. (a) The implementation of KNN is in the HW4_P1_aelhabr6.m file.

   (b) The following 9 figures show the classes of the training data set and the classification results for k values of 1, 2, 5, and 20 using both the ℓ_1 and ℓ_2 distances for the KNN algorithm.

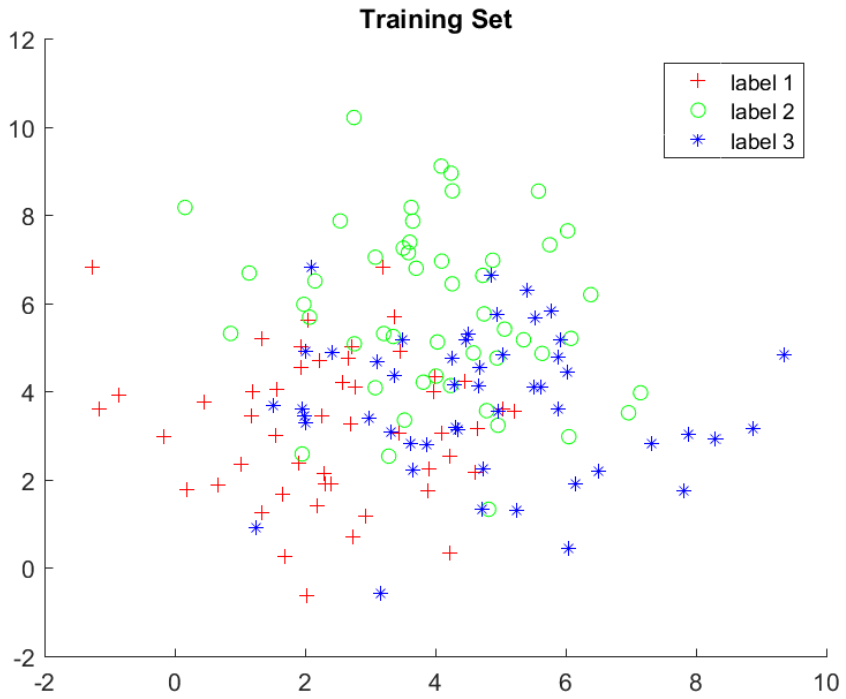

Figure 1: Classes of the training set.

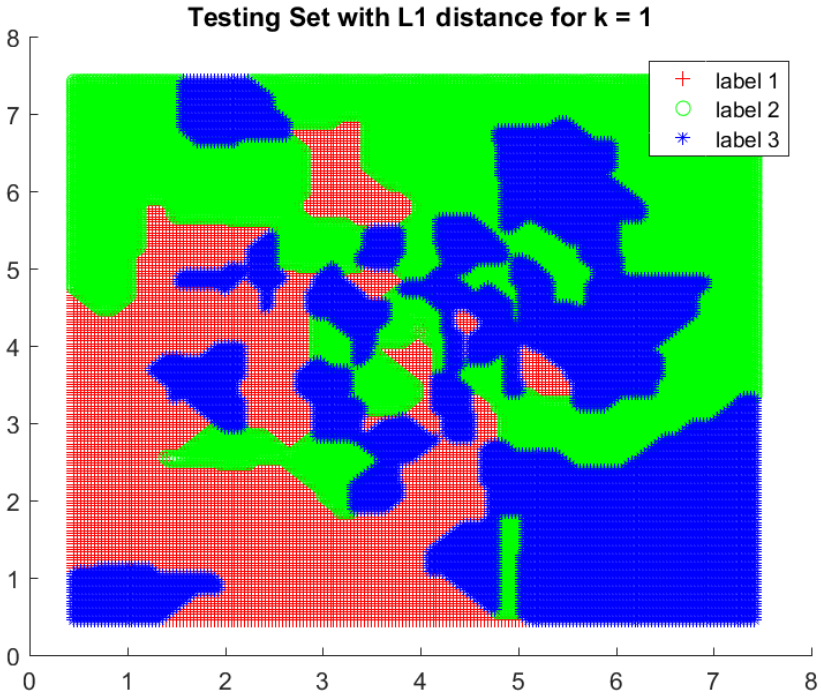

Figure 2: Classification results for $k = 1$ using ℓ_1 distance.

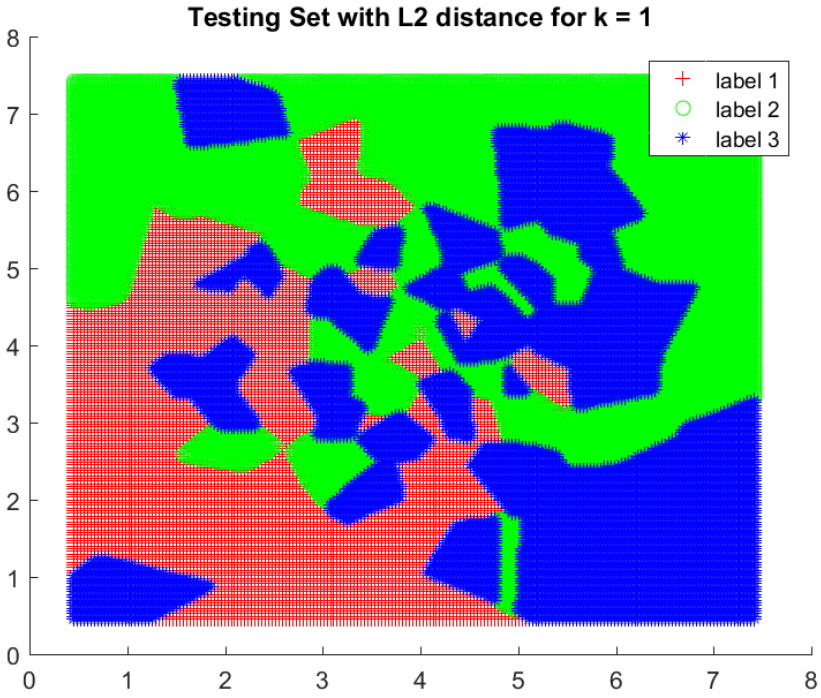

Figure 3: Classification results for $k = 1$ using ℓ_2 distance.

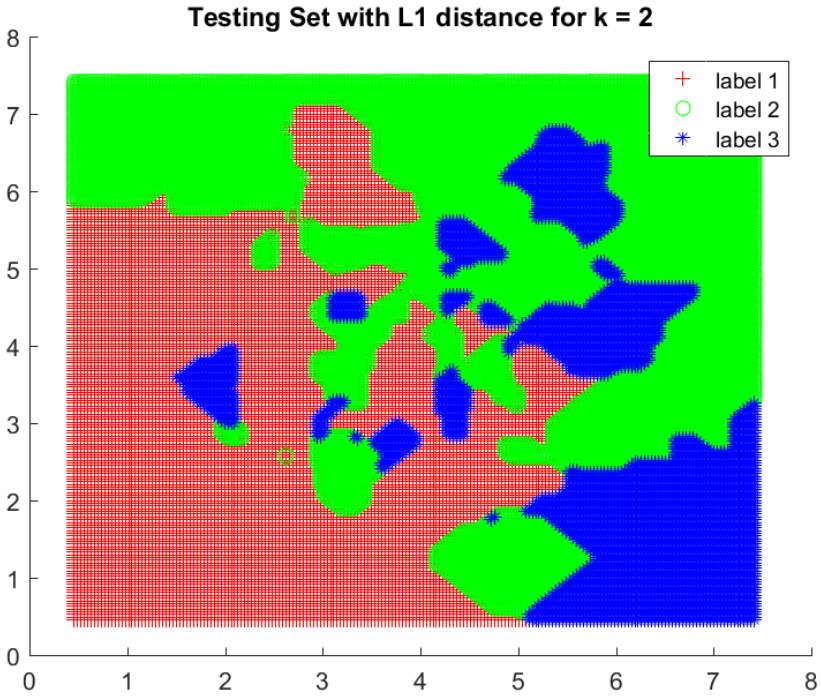

Figure 4: Classification results for $k = 2$ using ℓ_1 distance.

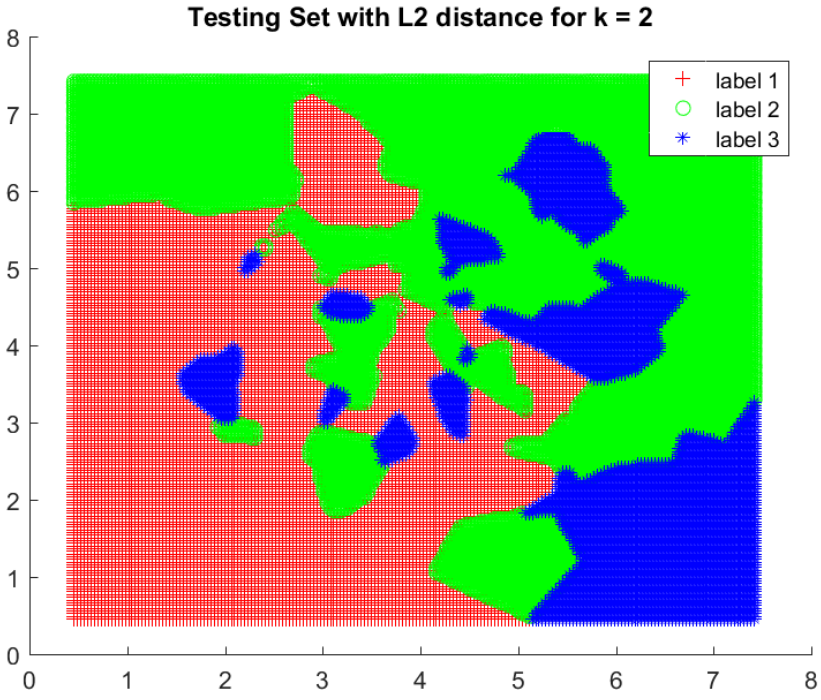

Figure 5: Classification results for $k = 2$ using ℓ_2 distance.

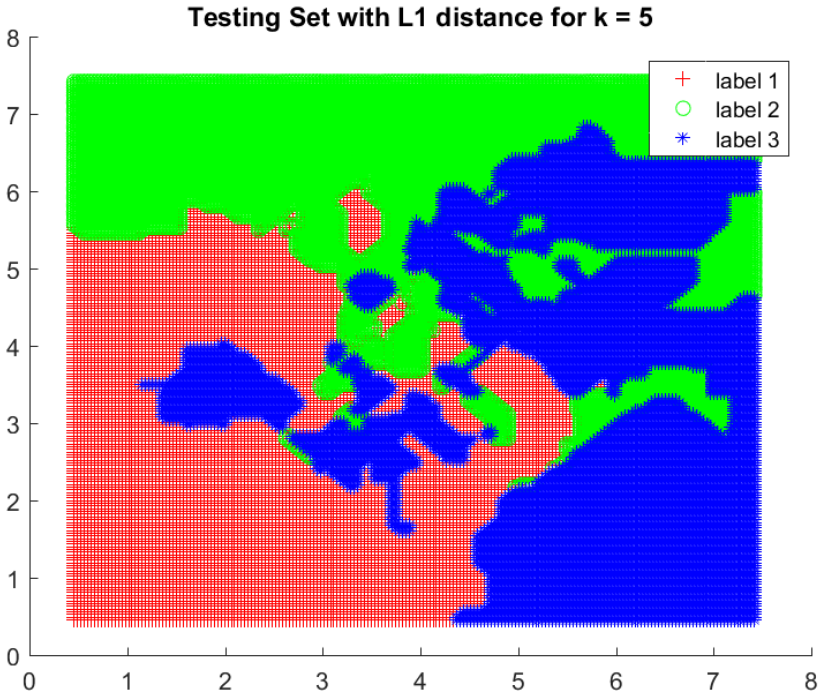

Figure 6: Classification results for $k = 5$ using ℓ_1 distance.

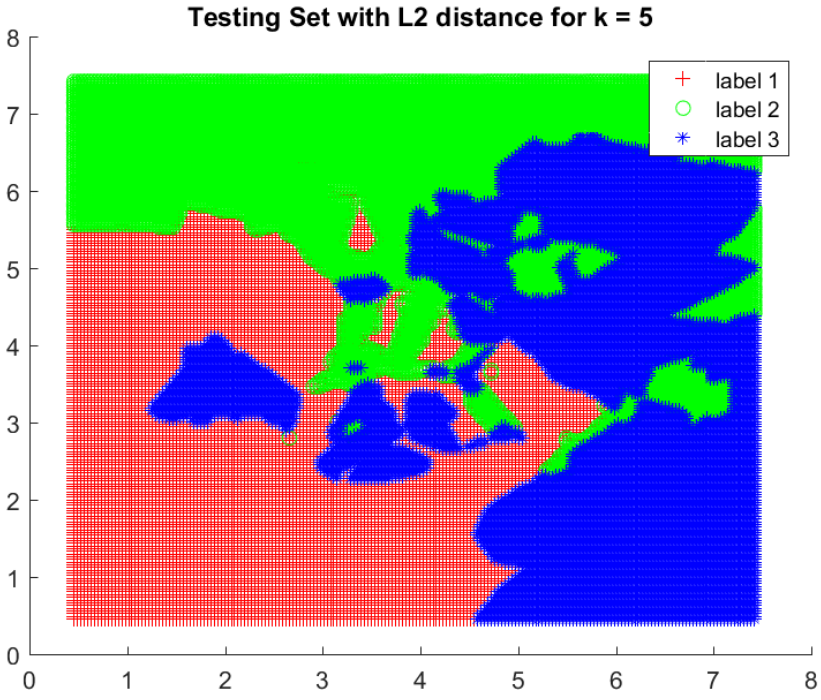

Figure 7: Classification results for $k = 5$ using ℓ_2 distance.

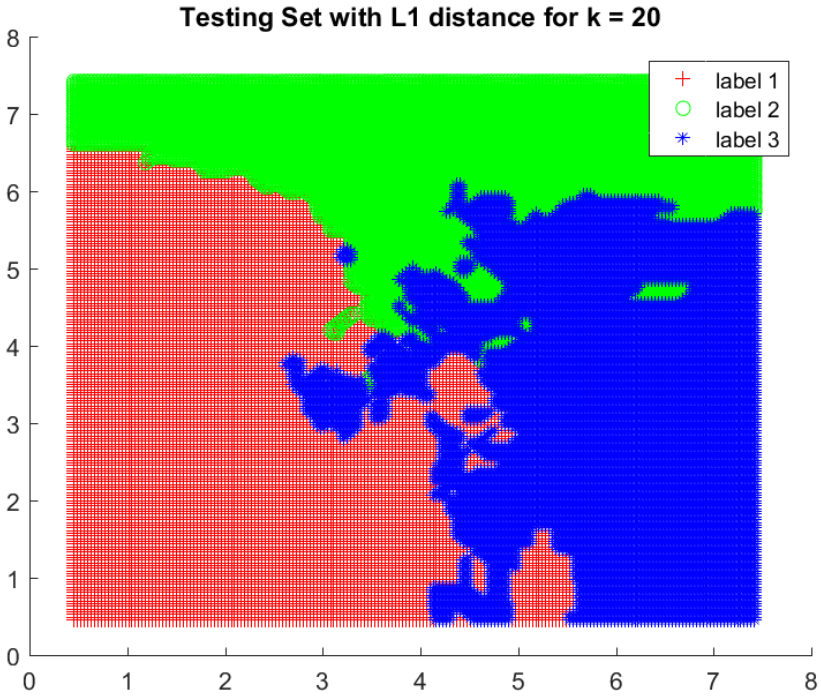

Figure 8: Classification results for $k = 20$ using ℓ_1 distance.

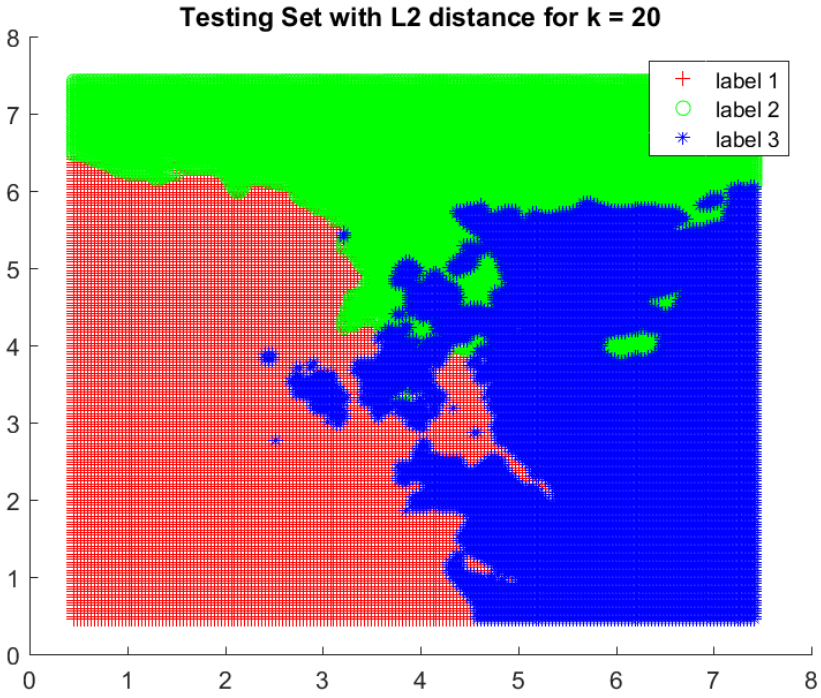

Figure 9: Classification results for $k = 20$ using ℓ_2 distance.

I notice that for any given k, the ℓ_1 and ℓ_2 distances give pretty similar classification results. In general, as k gets larger, the boundaries become more smooth and less granular, i.e. there are

much fewer locations where there are small spots of one label within another label as k gets larger. I notice that for any given k, the ℓ_1 and ℓ_2 distances give pretty similar classification results. If I had to make a statement about the differences between the ℓ_1 and ℓ_2 distances for a given k, I would say that the ℓ_2 distance gives slightly less granular results for this data.

2. (a) Let us define the function $W(v)$ in the following way, where u is a d-dimensional vector:

$$\begin{aligned} W(v) =& \frac{1}{2}\|u-v\|_2^2 + \lambda\|v\|_1 \\ =& \frac{1}{2}\sum_{i=1}^{d}(u_i - v_i)^2 + \lambda\sum_{i=1}^{d}|v_i| \\ =& \frac{1}{2}\sum_{i=1}^{d}(u_i^2 - 2u_iv_i + v_i^2) + \lambda\sum_{i=1}^{d}|v_i|. \end{aligned}$$

Note that $W(v)$ is differentiable everywhere when $v_i = 0$ for any $i = 1, ..., d$. Thus,

$$\frac{\partial W(v)}{\partial v_i} = v_i - u_i + \lambda\partial|v_i|,$$

where $\partial|v_i|$ is notation for the subdifferential of $|v_i|$. In the case where $W(v)$ is differentiable, i.e. $v \neq 0$, we can easily set this derivative to 0 and solve.

$$\begin{aligned} 0 =& v_i - u_i + \lambda\partial|v_i| \\ =& v_i - u_i + \lambda\text{sign}(v_i). \end{aligned}$$

Now, for the case where $u_i > \lambda$ $(v_i > 0)$, $v_i^* = u_i - \lambda$. Otherwise, if $u_i < -\lambda$ $(v_i < 0)$, $v_i^* = u_i + \lambda$. In the case where $|u_i| \leq \lambda$, we have that $\partial|v_i| = [-1, 1]$, and we can appropriately set $\partial|v_i|$ such that $u_i = \lambda\partial|v_i|$. Therefore, the optimal solution to the optimization problem is

$$v_i^* = \begin{cases} u_i - \lambda & \text{if } u_i > \lambda \\ u_i + \lambda & \text{if } u_i < -\lambda \\ 0 & \text{if } |u_i| \leq \lambda \end{cases}$$

for all $i = 1, ..., d$

(b) Similarly, let us define the function $Z(v)$ in the following way, where u is a d-dimensional vector:

$$\begin{aligned} Z(v) =& \frac{1}{2}\|u-v\|_2^2 + \lambda\|v\|_2 \\ =& \frac{1}{2}\sum_{i=1}^{d}(u_i - v_i)^2 + \lambda\sqrt{\sum_{i=1}^{d}v_i^2} \\ =& \frac{1}{2}\sum_{i=1}^{d}(u_i^2 - 2u_iv_i + v_i^2) + \lambda\sqrt{\sum_{i=1}^{d}v_i^2}. \end{aligned}$$

Note that $Z(v)$ is differentiable everywhere except at the point $v = 0$. Thus,

$$\nabla_v Z(v) = v - u + \lambda\partial||v||_2,$$

where $\partial||v||_2$ is notation for the subdifferential of $||v||_2$. By setting this gradient to 0, we see that we must solve the following new problem:

$$\begin{aligned} 0 \in& v - u + \lambda\partial||v||_2 \\ \implies \frac{u-v}{\lambda} \in& \partial||v||_2. \end{aligned}$$

Note that

$$\partial||v||_2 = \begin{cases} \frac{v}{||v||_2} & \text{if } v \neq 0 \\ \{g : ||g||_2 \leq 1\} & \text{if } v = 0. \end{cases}$$

If $v = 0$, we have that

$$\frac{||u||_2}{\lambda} \leq 1 \implies ||u||_2 \leq \lambda.$$

So if $v \neq 0$,

$$\begin{aligned} 0 =& v - u + \lambda \frac{v}{||v||_2} \\ \implies ||v||_2(v-u) =& \lambda v \\ \implies v =& u\Big(\frac{||v||_2}{||v||_2 - \lambda}\Big) \\ \implies u =& v\Big(\frac{||v||_2 - \lambda}{||v||_2}\Big). \end{aligned} \tag{1}$$

Now, by finding the ℓ_2 distance of both sides of this equation, we have that

$$\begin{aligned} ||u||_2 =& ||v||_2\Big(\frac{||v||_2 - \lambda}{||v||_2}\Big) \\ =& ||v||_2 - \lambda \\ \implies ||v||_2 =& ||u||_2 + \lambda, \end{aligned} \tag{2}$$

in which it must be that $||u||_2 > \lambda$ since it must be that $||v||_2 > 0$ for this case. Now, combining Equations 1 and 2, we find that for $||u||_2 > \lambda$,

$$\begin{aligned} v =& u\Big(\frac{||u||_2 + \lambda}{||u||_2 + \lambda - \lambda}\Big) \\ =& u\Big(\frac{||u||_2 + \lambda}{||u||_2}\Big). \end{aligned}$$

Therefore, the optimal solution to the optimization problem is

$$v^* = \begin{cases} 0 & \text{if } ||u||_2 \leq \lambda \\ u\Big(\frac{||u||_2 + \lambda}{||u||_2}\Big) & \text{if } ||u||_2 > \lambda. \end{cases}$$

3. (a) The greedy algorithm is implemented in the HW4_P3_aelhabr6.m file. After arbitrarily setting $K = 10$, I find that the validation error is minimized after 3 iterations, so the number of optimized features is 3. It turns out that the optimal estimator according to this algorithm is $\beta_i = 0$ for $i = 1, ..., 997$ and $\beta_{998} = 3.0947, \beta_{999} = 1.8476$, and $\beta_{1000} = 1.5569$. Clearly, the optimal estimator according to this algorithm is very sparse. The following three figures show how the validation, estimation, and prediction error change with each iteration of the greedy algorithm. They happen to all be minimized after the third iteration.

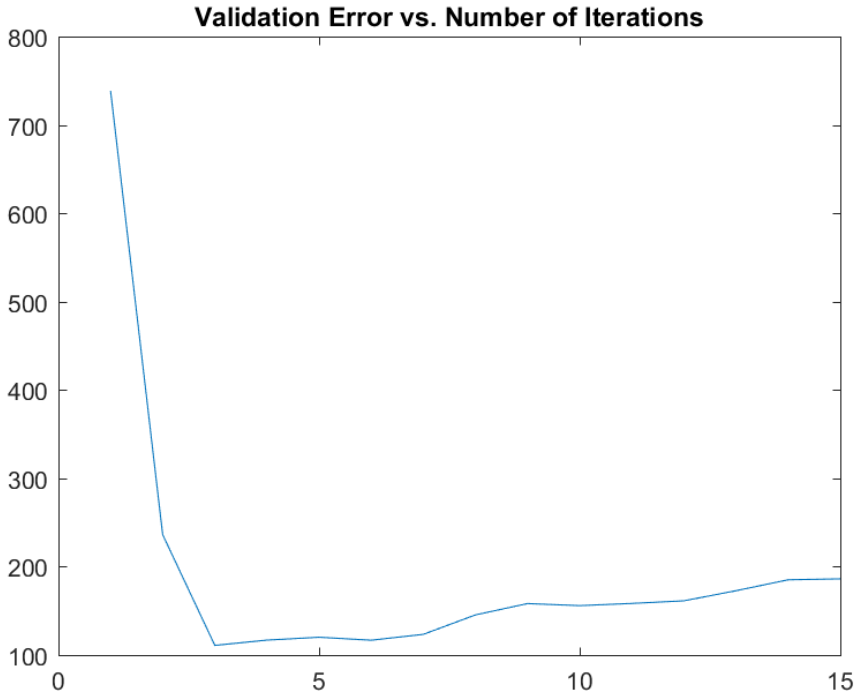

Figure 10: Validation error as a function of the number of iterations in the greedy algorithm.

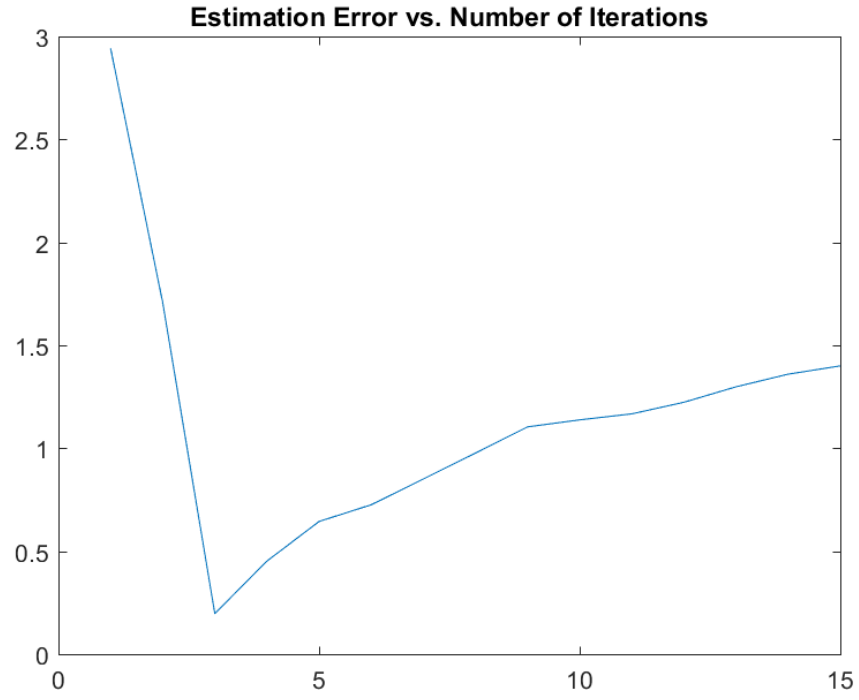

Figure 11: Estimation error as a function of the number of iterations in the greedy algorithm.

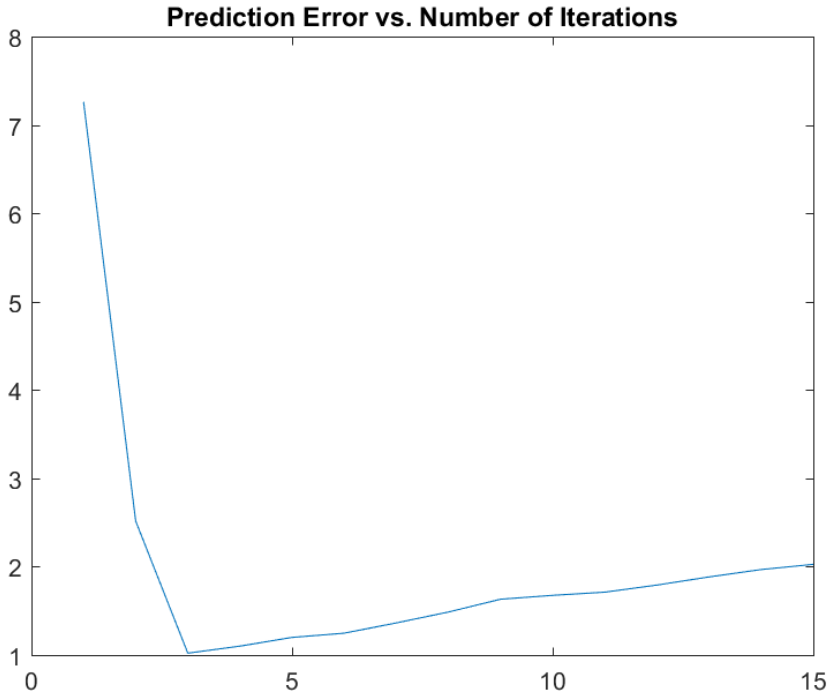

Figure 12: Prediction error as a function of the number of iterations in the greedy algorithm.

(b) The ridge regression estimator for this objective function is

$$\widehat{\beta}^{\text{Ridge}} = (X^\top X + 2n\lambda I)^{-1} X^\top y.$$

(Note that for the objective function found in most literature, $\widehat{\beta}^{\text{Ridge}} = (X^\top X + \lambda I)^{-1} X^\top y$.) The validation error is minimized for $\lambda = 0.0125$. Therefore, we say that the optimal estimator using ridge regression is the one corresponding to $\lambda = 0.0125$, which has all 1000 features and is not sparse at all. The following three figures show how the validation, estimation, and prediction error change with each regulation parameter that was examined. It turns out each error type is minimized for different regulation parameters.

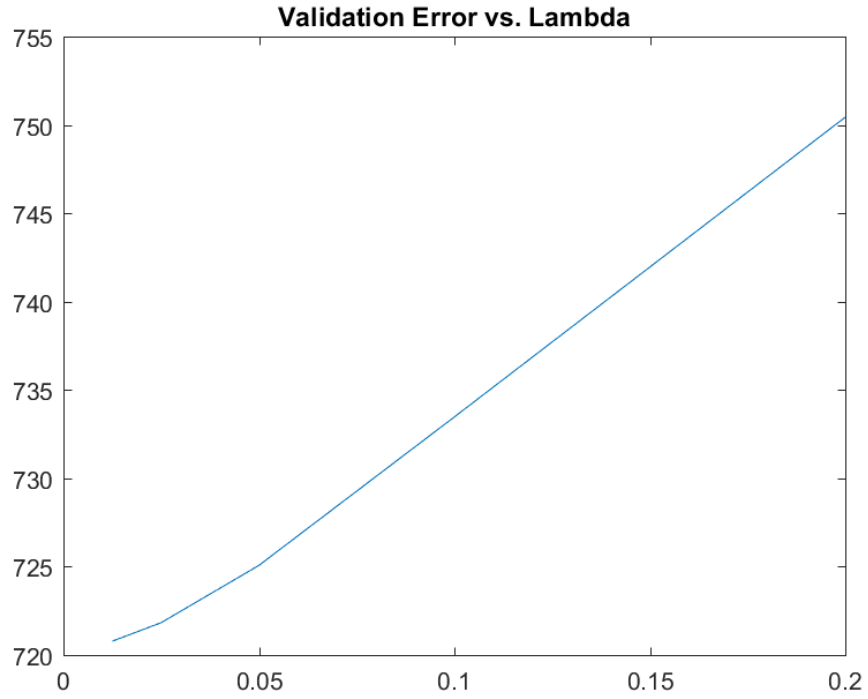

Figure 13: Validation error as a function of the regulation parameter for ridge regression.

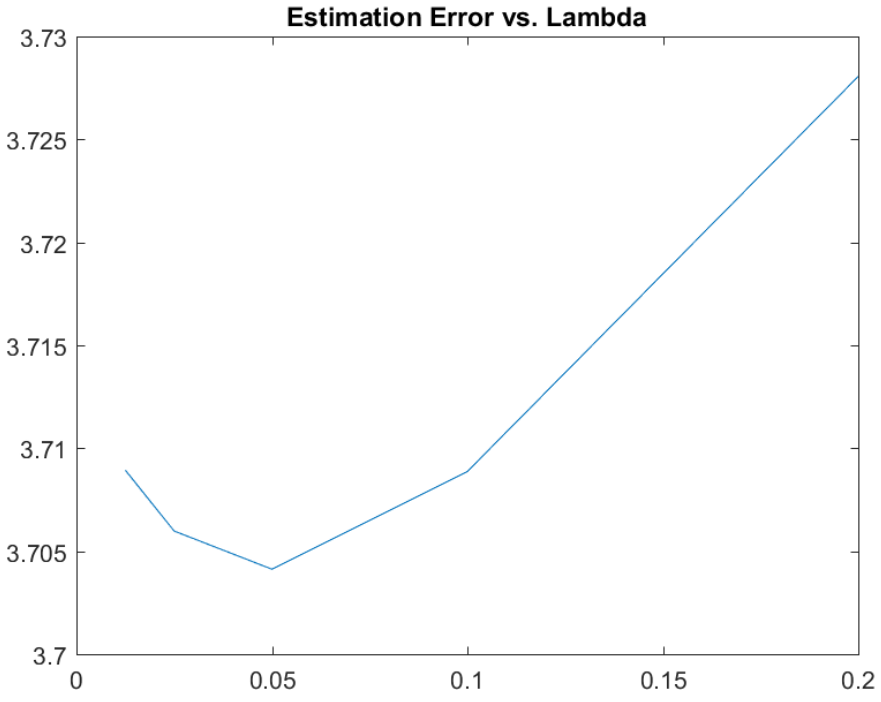

Figure 14: Estimation error as a function of the regulation parameter for ridge regression.

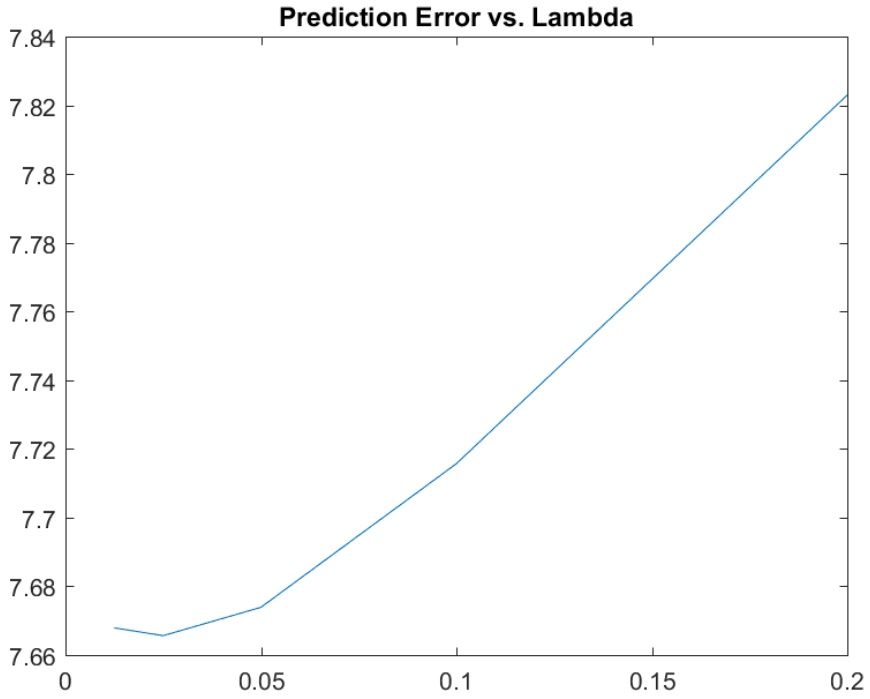

Figure 15: Prediction error as a function of the regulation parameter for ridge regression.

(c) The validation error for LASSO is minimized for $\lambda = 0.1321$. Therefore, we say that the optimal estimator using ridge regression is the one corresponding to $\lambda = 0.1321$, which has 27 features and is relatively sparse. The following three figures show how the validation, estimation, and prediction error change with each MATLAB default regulation parameter that was examined. It turns out each error type is minimized for different regulation parameters.

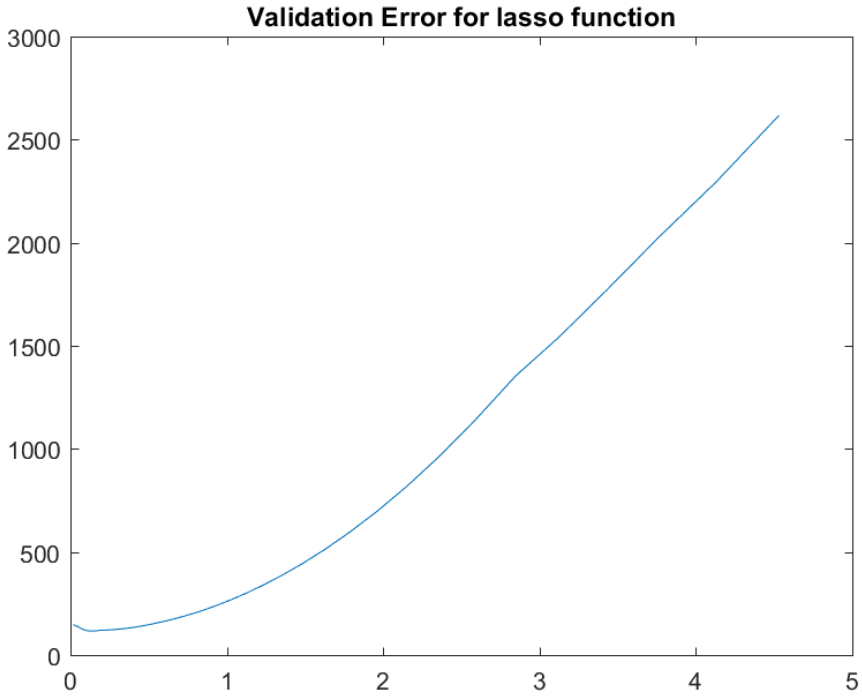

Figure 16: Validation error as a function of the regulation parameter for LASSO.

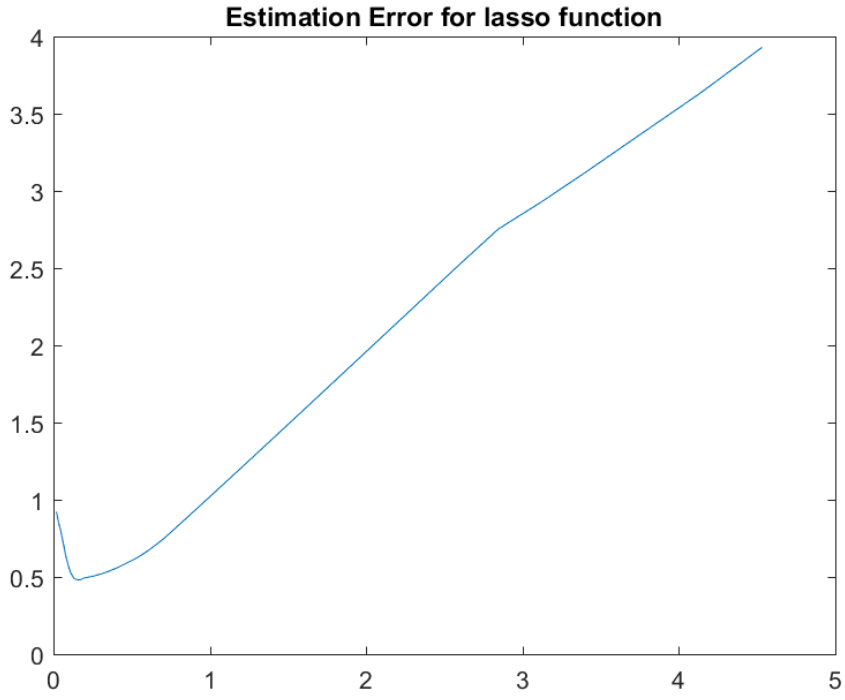

Figure 17: Estimation error as a function of the regulation parameter for LASSO.

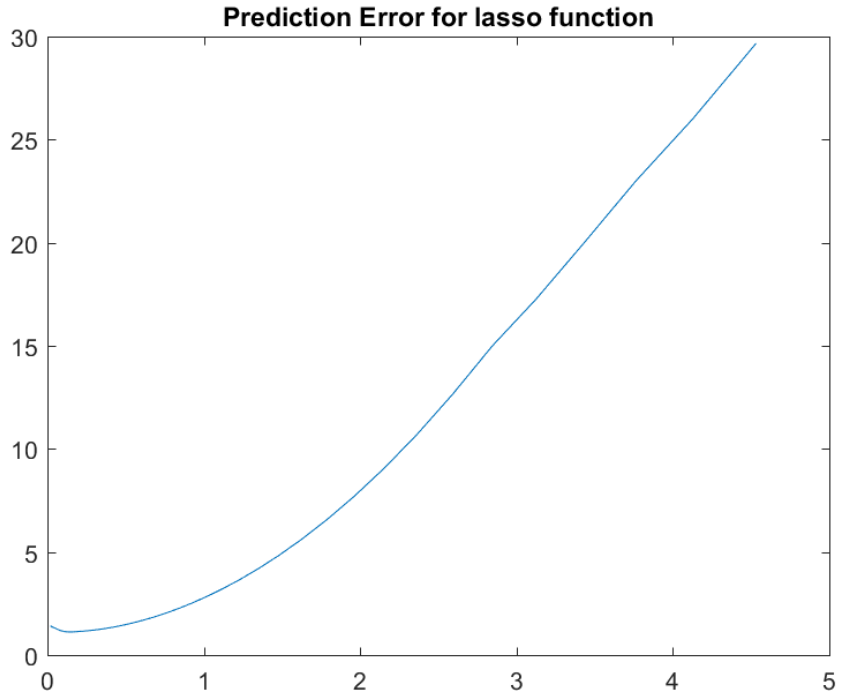

Figure 18: Prediction error as a function of the regulation parameter for LASSO.

(d) The refit estimation error is 1.9678, and the lasso estimation error is 0.4860. So the refit estimation error is actually not smaller than the lasso estimation error. After refitting the estimator, it only loses 2 features. So the refit estimator has no significant difference in sparsity when compared to the lasso estimator. Thus, there is likely not going to be much of a difference between the two errors. In general, I would say the difference between the two is random and has more to do with the fact that there are so many more covariates than observations, trying to refit the lasso estimator does not help much, and in this case, actually made the estimation error worse. In fact, we find that the estimator from the greedy algorithm had the lowest estimation error since it was the most sparse estimator.