CS480/680: Introduction to Machine Learning

Assignment 1 Solution

Exercise 1: Perceptron Implementation (5 pts)

Convention: All algebraic operations, when applied to a vector or matrix, are understood to be element-wise (unless otherwise stated).

Algorithm 1: The perceptron algorithm.

Input: $X \in \mathbb{R}^{n\times d}$, $\mathbf{y} \in \{-1,1\}^n$, $\mathbf{w} = \mathbf{0}_d$, $b = 0$, $\mathsf{max\_pass} \in \mathbb{N}$
Output: $\mathbf{w}, b, mistake$

for $t = 1, 2, \ldots, \mathsf{max\_pass}$ **do**
  $mistake(t) \leftarrow 0$
  for $i = 1, 2, \ldots, n$ **do**
    if $y_i(\langle \mathbf{x}_i, \mathbf{w}\rangle + b) \leq 0$ **then**
      $\mathbf{w} \leftarrow \mathbf{w} + y_i\mathbf{x}_i$ `// x_i is the i-th row of X`
      $b \leftarrow b + y_i$
      $mistake(t) \leftarrow mistake(t) + 1$

Implement the perceptron in Algorithm 1. Your implementation should take input as $X = [\mathbf{x}_1^\top, \ldots, \mathbf{x}_n^\top]^\top \in \mathbb{R}^{n\times d}$, $\mathbf{y} \in \{-1,1\}^n$, an initialization of the hyperplane parameters $\mathbf{w} \in \mathbb{R}^d$ and $b \in \mathbb{R}$, and the maximum number of passes of the training set [suggested $\mathsf{max\_pass} = 500$]. Run your perceptron algorithm on the spambase dataset (use the version on the course website), and plot the number of mistakes (y-axis) w.r.t. the number of passes (x-axis).

Ans:

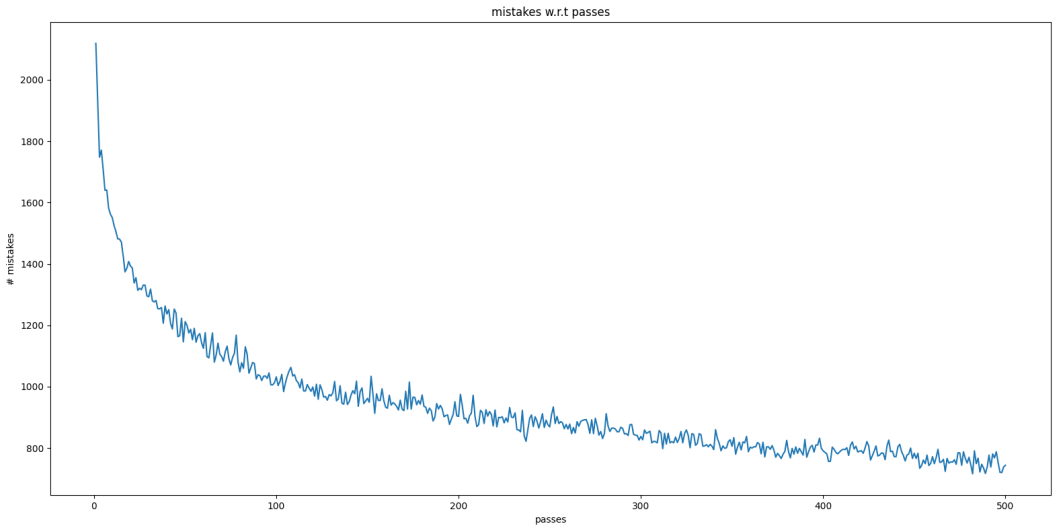

Figure 1: #mistakes per pass)

Exercise 2: Regression Implementation (8 pts)

Recall that ridge regression refers to

$$\min_{\mathbf{w}\in\mathbb{R}^d, b\in\mathbb{R}} \overbrace{\underbrace{\frac{1}{2n}\|X\mathbf{w}+b\mathbf{1}-\mathbf{y}\|_2^2}_{\text{error}}+\lambda\|\mathbf{w}\|_2^2}^{\text{loss}}, \tag{1}$$

where $X \in \mathbb{R}^{n\times d}$ and $\mathbf{y} \in \mathbb{R}^n$ are the given dataset and $\lambda \geq 0$ is the regularization hyperparameter. If $\lambda = 0$, then this is the standard linear regression problem. Observe the distinction between the error (which does not include the regularization term) and the loss (which does).

1. (1 pt) Show that ridge regression can be rewritten as a non-regularized linear regression problem. That is, prove 1 is equivalent to

$$\min_{\mathbf{w}\in\mathbb{R}^d, b\in\mathbb{R}} \frac{1}{2n}\left\|\begin{bmatrix} X & \mathbf{1}_n \\ \sqrt{2\lambda n}I_d & \mathbf{0}_d\end{bmatrix}\begin{bmatrix}\mathbf{w}\\ b\end{bmatrix}-\begin{bmatrix}\mathbf{y}\\ \mathbf{0}_d\end{bmatrix}\right\|_2^2, \tag{2}$$

   where I_d is the d-dimensional identity matrix, and $\mathbf{0}_k$ and $\mathbf{1}_k$ are zero and one column vectors in k dimensions.

   Ans:

$$\begin{aligned}
&\min_{\mathbf{w}\in\mathbb{R}^d, b\in\mathbb{R}} \frac{1}{2n}\left\|\begin{bmatrix} X & \mathbf{1}_n \\ \sqrt{2\lambda n}I_d & \mathbf{0}_d\end{bmatrix}\begin{bmatrix}\mathbf{w}\\ b\end{bmatrix}-\begin{bmatrix}\mathbf{y}\\ \mathbf{0}_d\end{bmatrix}\right\|_2^2,\\
&= \min_{\mathbf{w}\in\mathbb{R}^d, b\in\mathbb{R}} \frac{1}{2n}\left\|\begin{bmatrix} X\mathbf{w}+b\mathbf{1}_n \\ \sqrt{2\lambda n}\mathbf{w}I_d\end{bmatrix}-\begin{bmatrix}\mathbf{y}\\ \mathbf{0}_d\end{bmatrix}\right\|_2^2,\\
&= \min_{\mathbf{w}\in\mathbb{R}^d, b\in\mathbb{R}} \frac{1}{2n}\left\|\begin{bmatrix} X\mathbf{w}+b\mathbf{1}_n-\mathbf{y} \\ \sqrt{2\lambda n}\mathbf{w}I_d\end{bmatrix}\right\|_2^2,\\
&= \min_{\mathbf{w}\in\mathbb{R}^d, b\in\mathbb{R}} \frac{1}{2n}\begin{bmatrix}(X\mathbf{w}+b\mathbf{1}_n-\mathbf{y})^T & (\sqrt{2\lambda n}\mathbf{w}I_d)^T\end{bmatrix}\begin{bmatrix} X\mathbf{w}+b\mathbf{1}_n-\mathbf{y} \\ \sqrt{2\lambda n}\mathbf{w}I_d\end{bmatrix}\\
&= \min_{\mathbf{w}\in\mathbb{R}^d, b\in\mathbb{R}} \frac{1}{2n}\left[\|X\mathbf{w}+b\mathbf{1}_n-\mathbf{y}\|_2^2+2\lambda n\|\mathbf{w}\|_2^2\right],\\
&= \min_{\mathbf{w}\in\mathbb{R}^d, b\in\mathbb{R}} \frac{1}{2n}\|X\mathbf{w}+b\mathbf{1}_n-\mathbf{y}\|_2^2+\lambda\|\mathbf{w}\|_2^2,
\end{aligned}$$

2. (1 pt) Show that the derivatives of 1 are

$$\frac{\partial}{\partial \mathbf{w}} = \frac{1}{n}X^\top(X\mathbf{w}+b\mathbf{1}-\mathbf{y})+2\lambda\mathbf{w} \tag{3}$$

$$\frac{\partial}{\partial b} = \frac{1}{n}\mathbf{1}^\top(X\mathbf{w}+b\mathbf{1}-\mathbf{y}). \tag{4}$$

   Ans:

$$\begin{aligned}
&\frac{\partial}{\partial \mathbf{w}}\left[\frac{1}{2n}\|X\mathbf{w}+b\mathbf{1}_n-\mathbf{y}\|_2^2+\lambda\|\mathbf{w}\|_2^2\right]\\
&= \frac{\partial}{\partial \mathbf{w}}\left\{\frac{1}{2n}\left[X\mathbf{w}+b\mathbf{1}_n-\mathbf{y}\right]^T\left[X\mathbf{w}+b\mathbf{1}_n-\mathbf{y}\right]+\lambda\mathbf{w}^T\mathbf{w}\right\}\\
&= \frac{\partial}{\partial \mathbf{w}}\left\{\frac{1}{2n}\left[\mathbf{w}^TX^T+(b\mathbf{1}_n-\mathbf{y})^T\right]\left[X\mathbf{w}+b\mathbf{1}_n-\mathbf{y}\right]+\lambda\mathbf{w}^T\mathbf{w}\right\}\\
&= \frac{\partial}{\partial \mathbf{w}}\left\{\frac{1}{2n}\left[\mathbf{w}^TX^TX\mathbf{w}+2\mathbf{w}^TX^T(b\mathbf{1}_n-\mathbf{y})+(b\mathbf{1}_n-\mathbf{y})^T(b\mathbf{1}_n-\mathbf{y})\right]+\lambda\mathbf{w}^T\mathbf{w}\right\}
\end{aligned}$$

$$= \tfrac{1}{2n}\left\{\left[(X^TX) + (X^TX)^T\right]\mathbf{w} + 2X^T(b\mathbf{1}_n - \mathbf{y})\right\} + 2\lambda\mathbf{w}$$
$$= \tfrac{1}{2n}\left\{\left[(X^TX) + (X^TX)\right]\mathbf{w} + 2X^T(b\mathbf{1}_n - \mathbf{y})\right\} + 2\lambda\mathbf{w}$$
$$= \tfrac{1}{2n}\left\{\left[2(X^TX)\right]\mathbf{w} + 2X^T(b\mathbf{1}_n - \mathbf{y})\right\} + 2\lambda\mathbf{w}$$
$$= \tfrac{1}{n}\left\{X^TX\mathbf{w} + X^T(b\mathbf{1}_n - \mathbf{y})\right\} + 2\lambda\mathbf{w}$$
$$\frac{\partial}{\partial \mathbf{w}} = \tfrac{1}{n}X^\top(X\mathbf{w} + b\mathbf{1}_n - \mathbf{y}) + 2\lambda\mathbf{w}$$

$$\frac{\partial}{\partial b}\left[\tfrac{1}{2n}\|X\mathbf{w} + b\mathbf{1}_n - \mathbf{y}\|_2^2 + \lambda\|\mathbf{w}\|_2^2\right]$$
$$= \frac{\partial}{\partial b}\left\{\tfrac{1}{2n}\left[\mathbf{w}^TX^TX\mathbf{w} + 2(b\mathbf{1}_n - \mathbf{y})^TX\mathbf{w} + (b^2\mathbf{1}_n^T\mathbf{1}_n - 2b\mathbf{1}_n^T\mathbf{y} + \mathbf{y}^T\mathbf{y})\right] + \lambda\mathbf{w}^T\mathbf{w}\right\}$$
$$= \tfrac{1}{n}\left[\mathbf{1}_n^TX\mathbf{w} + b\mathbf{1}_n^T\mathbf{1}_n - \mathbf{1}_n^T\mathbf{y}\right]$$
$$= \tfrac{1}{n}\mathbf{1}_n^T\left[X\mathbf{w} + b\mathbf{1}_n - \mathbf{y}\right]$$
$$\frac{\partial}{\partial b} = \tfrac{1}{n}\mathbf{1}_n^\top(X\mathbf{w} + b\mathbf{1}_n - \mathbf{y})$$

3. (2 pts) Implement ridge regression using the closed form solution for linear regression as derived in lecture. You may find the function `numpy.linalg.solve` useful.

   Ans:

4. (2 pts) Implement the gradient descent algorithm for solving ridge regression. The following incomplete pseudo-code may of help. Your training loss should monotonically decrease during iteration; if not try to tune your step size η, e.g. make it smaller.

Algorithm 2: Gradient descent for ridge regression.

Input: $X \in \mathbb{R}^{n\times d}$, $\mathbf{y} \in \mathbb{R}^n$, $\mathbf{w}_0 = \mathbf{0}_d$, $b_0 = 0$, max_pass $\in \mathbb{N}$, $\eta > 0$, tol > 0
Output: $\mathbf{w}, b$

```
for t = 1, 2, ..., max_pass do
    w_t ←
    b_t ←
    if ||w_t − w_{t−1}|| ≤ tol then                    // can use other stopping criteria
        break
w ← w_t,  b ← b_t
```

   Ans:

5. (2 pts) Test your two implementations on the Boston housing dataset (to predict the median house price, i.e., y). Use the train and test splits provided on course website. Try $\lambda \in \{0, 10\}$ and report your training error, training loss and test error for each. For the gradient descent algorithm, plot the training loss over iterations. Compare the running time of the two approaches, which is faster? Overall, which approach do you think is better? Explain why.

   Ans: Check table 2 and figure 6.
   Training: error = 4.847, loss = 4.847, test error is 185.111 for closed-form with lambda=0.
   The training error is 18.259, training loss is 22.413, test error is 25.409 for closed-form with lambda=10.
   The training error is 39.437, training loss is 39.437, test error is 72.516 for gradient descent apprpach with lambda=0
   The training error is 39.441, training loss is 39.472, test error is 72.531 for gradient descent apprpach with lambda=10.

Closed-form seems faster than the gradient descent here, as the number of variables was not large. In general, closed-form has higher time/computation complexity than the GD ($O(n^3)$). Also, You cannot find closed-form solutions for all non-linear regression setups, and for the linear regression for some problems the closed form would not work! (as per high time complexity) to fit a model.

6. (Extra credit: 1 pt) You might notice that one of your two optimizers for solving linear regression performs much worse than the other. See if you can improve this method to make it perform comparably to the other (better) approach. You may use ideas beyond the scope of this course. Try to stay within the "spirit" of the problem, since the extra credit will be discretionary. For example, "fixing" the poorly performing method by simply replacing it with the other approach is not allowed. Things that you might explore could involve data preprocessing, alternate approaches to parameter setting, etc.

Ans:

Newton's method, $\lambda = 0$
weights:
[1.2253215 0.01236248 0.01996493 0.67897257 − 8.10628857 9.14471067
−0.04698815 − 0.99589523 0.11477167 − 0.01468719 − 0.61916116 0.0168099
−0.1158987]
bias: −13.02614455283862

Closed form, $\lambda = 0$
weights:
[1.22532732 0.01236254 0.01996503 0.6789758 − 8.10632711 9.14475415
−0.04698837 − 0.99589996 0.11477222 − 0.01468726 − 0.6191641 0.01680998
−0.11589925]
bias: −13.026206486865533

Sklearn, $\lambda = 0$
weights:
[1.22532732 0.01236254 0.01996503 0.6789758 − 8.10632711 9.14475415
−0.04698837 − 0.99589996 0.11477222 − 0.01468726 − 0.6191641 0.01680998
−0.11589925]
bias: −13.02620648686728

Newton's method, $\lambda = 10$
weights:
[0.65896588 0.01864522 − 0.00874125 0.56710527 − 0.48866653 7.92248993
−0.04086783 − 0.91822485 0.1519581 − 0.01648653 − 0.60021546 0.01779576
−0.23013932]
bias: −8.48562934069874

Closed form, $\lambda = 10$
weights:
[0.65896901 0.01864531 − 0.00874129 0.56710796 − 0.48866886 7.92252759
−0.04086802 − 0.91822922 0.15195882 − 0.01648661 − 0.6002183 0.01779585
−0.23014041]
bias: −8.485669686416827

Sklearn, $\lambda = 10$
weights:
[0.65896901 0.01864531 − 0.00874129 0.56710796 − 0.48866886 7.92252759
−0.04086802 − 0.91822922 0.15195882 − 0.01648661 − 0.60021831 0.01779585
−0.23014041]

bias: -8.485669686419946

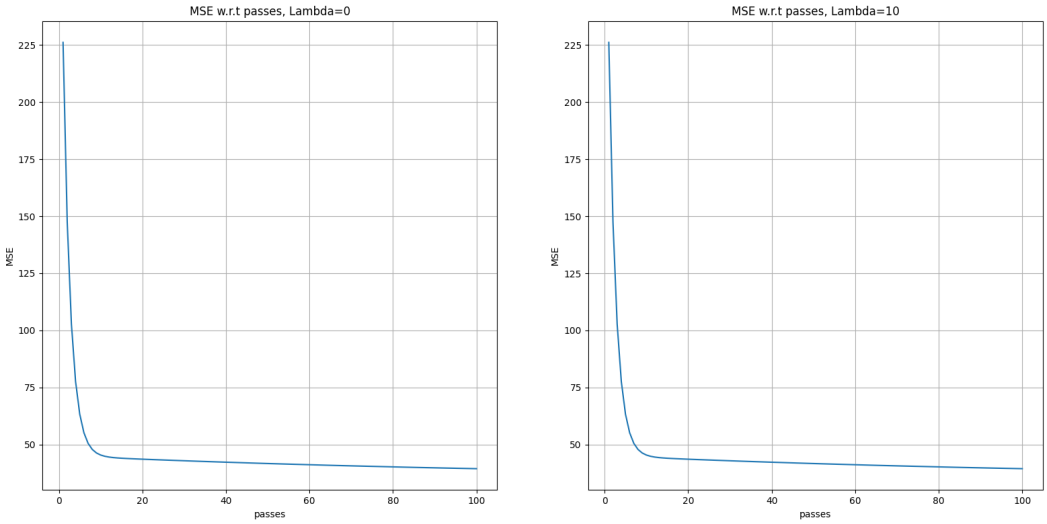

Figure 2: train loss per iterations

Table 1: running time, train error, train loss, and test error of models

model	time	train err	train loss	test err
closed form, $\lambda = 0$	24.6048e-05	4.84715	4.84715	185.111
closed_form, $\lambda = 10$	7.55787e-05	18.2591	22.4135	25.4096
gradient descent, $\lambda = 0$	710.654e-05	39.4373	39.4373	72.5162
gradient descent, $\lambda = 10$	729.299e-05	39.4415	39.4730	72.5319

Exercise 3: Playing with Regression (3 pts)

You may use the Python package scikit-learn for this exercise (and only for this exercise).

Train (unregularized) linear regression, ridge regression, and lasso on the mystery datasets A, B, and C on the course website (using X_train and Y_train for each dataset). For ridge regression and lasso, use regularization parameters 1 and 10 (note that you will have to divide these by n for lasso in scikit-learn – explain why). Report the average mean squared error on the test set for each method. Which approach performs best in each case? Plot the five parameter vectors obtained for each dataset on the same histogram, so they're all visible at once (change the opacity setting for the bars if necessary): specifically, for each parameter vector, plot a histogram of its value in each coordinate.

Ans: The reason why we need to divide the regularization parameters by n for lasso in scikit-learn is that the loss fuction for lasso in scikit-learn is $\frac{1}{n}\|y - Xw\|_2^2 + \lambda\|w\|_1$ if we want to set the regularization parameter to λ, we should normalize the regularization parameter by $\frac{1}{n}\lambda$, or the actual regularization parameter add to weight vector would be effected by the size of the dataset(n).

All errors are illustrated in Table 2

For dataset A

mean square error of Linear Regression is 3.247

mean square error of Ridge Regression with regularization parameter 1 is 3.139

mean square error of Ridge Regression with regularization parameter 10 is 2.778

mean square error of Lasso Regression with regularization parameter 1 is 3.020

mean square error of Lasso Regression with regularization parameter 10 is 3.597

Ridge regression with regularization parameter = 10 performs best.

For dataset B

mean square error of Linear Regression is 2.742

mean square error of Ridge Regression with regularization parameter 1 is 2.616

mean square error of Ridge Regression with regularization parameter 10 is 2.059

mean square error of Lasso Regression with regularization parameter 1 is 2.259

mean square error of Lasso Regression with regularization parameter 10 is 1.808

Lasso regression with regularization parameter = 10 performs best.

For dataset C

mean square error of Linear Regression is 505.000

mean square error of Ridge Regression with regularization parameter 1 is 505.277

mean square error of Ridge Regression with regularization parameter 10 is 515.891

mean square error of Lasso Regression with regularization parameter 1 is 2.205

mean square error of Lasso Regression with regularization parameter 10 is 1.356

Lasso regression with regularization parameter = 10 performs best.

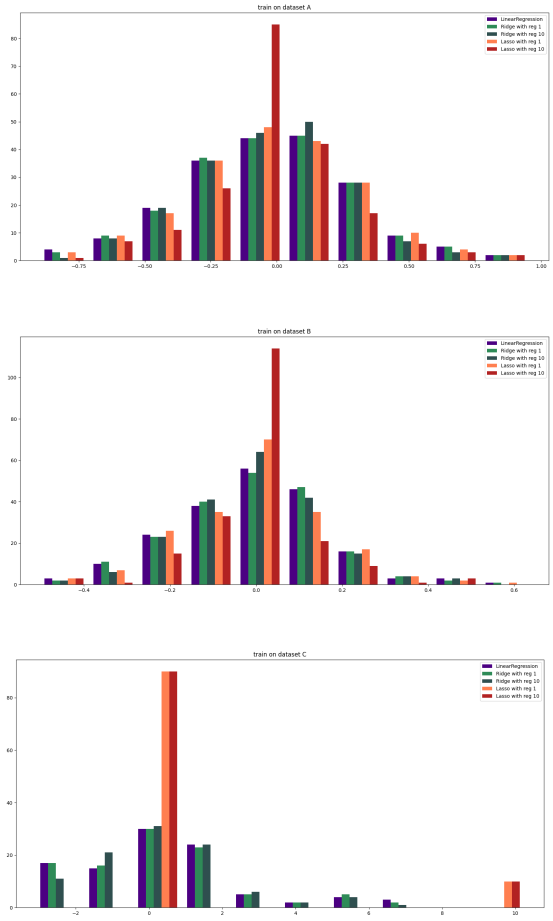

Figure 3: parameter vectors (Exe3)

Table 2: Mean square errors

model	dataset A	dataset B	dataset C
Linear Regression	3.24740	2.74268	505.000
Ridge Regression with $\lambda = 1$	3.13939	2.61620	505.277
Ridge Regression with $\lambda = 10$	2.77803	2.05971	515.891
Lasso Regression with $\lambda = 1$	3.02040	2.26517	1.87747
Lasso Regression with $\lambda = 10$	3.60263	1.80967	1.35256

Exercise 4: Nearest Neighbour Regression (7 pts)

1. (3 pts) Implement k-nearest neighbour regression, for a dataset of n X, y pairs where $X \in \mathbb{R}^d$ and $y \in \mathbb{R}$. This is similar to k-nearest neighbour classification, but instead of aggregating labels by taking the majority, we average the y values of the k nearest neighbours. Use ℓ_2 distance as the distance metric, that is, the distance between points X_1 and X_2 is $\|X_1 - X_2\|_2$. Ensure that your implementation is $O(nd)$ time for all values of k, and argue that this is the case.

   Ans: An algorithm would calculate the distance between every point in the training set, for a training set of size n, and each point has d-dimensional features, it would take a time of $O(nd)$ and store the distances in an array. After that using quick-select algorithm helps returning the k elements with the shortest distance in a time of $O(n)$. In the end, calculating the mean of the k value would take a time of $O(k)$. So the time complexity of the proposed algorithm in total would be of $O(nd)$.

2. (2 pts) For the training sets of the $d = 1$ mystery datasets D and E on the course website, compute a) the (unregularized) least squares linear regression solution and b) the k-nearest neighbour regression solution with each integer k from 1 to 9. For each dataset, on one plot with the x and y axes corresponding to the x and y values, display the least squares solution, the 1-nearest neighbour solution, and the 9-nearest neighbour solution. Be sure to plot these solutions for all points between the minimum and maximum x values, not just for the points in the dataset. On another plot, with k on the x axis and test mean-squared error on the y axis, display the error of the k-NN solution for each value of k. On the same plot, include the test error of the least squares solution as a horizontal line. When does each approach perform better, and why?

   Ans: The data features are from a linear distribution, so linear regression works better than k-NN, as the MSE of linear regression is lower than k-NN. As linear regression avoids the interference of noise in the training set, which could prevent the model from over-fitting, and ensure the decent performance of the model on the test set. However, when the data deviates from the linear distribution, like dataset E, k-NN performs better.

3. (2 pts) For the training set of the $d = 20$ mystery dataset F on the course website, compute a) the (unregularized) least squares linear regression solution and b) the k-nearest neighbour regression solution with each integer k from 1 to 9. Plot, with k on the x axis and test mean-squared error on the y axis, display the error of the k-NN solution for each value of k. On the same plot, include the test error of the least squares solution as a horizontal line. Which approach performs better, and why? Hint: consider inspecting distances between the test points and their k nearest neighbours in the training set.

   Ans: Linear regression performs better than k-NN for dataset F as the average distances from the test point to the nearest neighbours are far compared to datasets D and E.

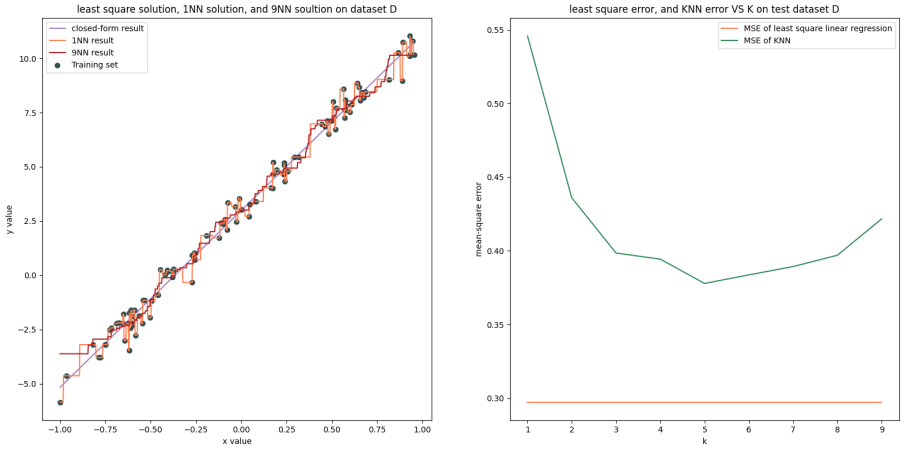

Figure 4: dataset D

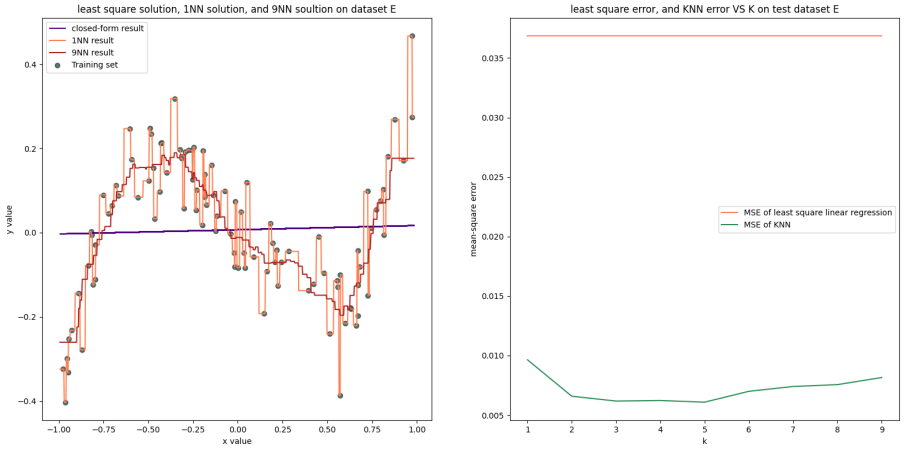

Figure 5: dataset E

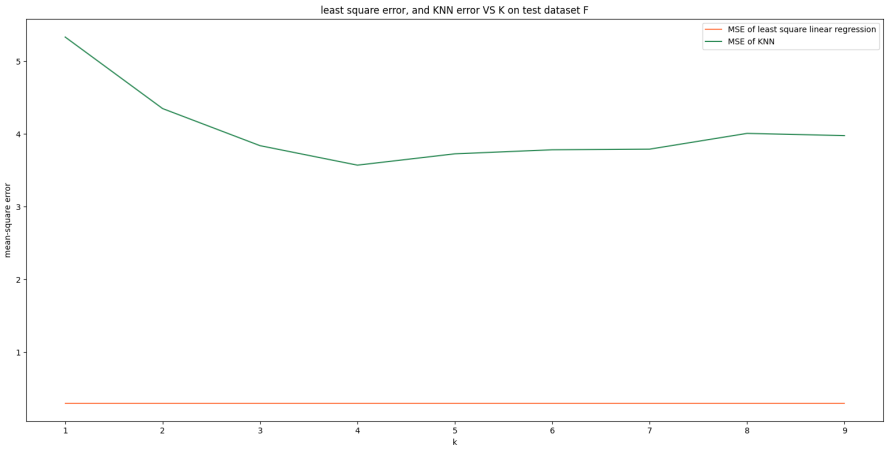

Figure 6: dataset F