Problem 2 (Kernels and kNN, 10pts)

Now, let us compare the kernel-based approach to an approach based on nearest-neighbors. Recall that kNN uses a predictor of the form

$$f(x^*) = \frac{1}{k}\sum_n y_n \mathbb{I}(x_n \texttt{ is one of k-closest to } x^*)$$

where $\mathbb{I}$ is an indicator variable. For this problem, you will use the same kernels as Problem 1, and dataset `data/p2.csv`.

For this problem, you can use our staff **script to compare your code to a set of staff-written test cases.** This requires, however, that you use the structure of the starter code provided in `T1_P2.py`. More specific instructions can be found at the top of the file `T1_P2_Testcases.py`. You may run the test cases in the command-line using `python T1_P2_TestCases.py`. **Note that our set of test cases is not comprehensive: just because you pass does not mean your solution is correct! We strongly encourage you to write your own test cases and read more about ours in the comments of the Python script.**

1. We will be making 6 plots comparing kernel-based and nearest neighbor-based predictors, all using the Mahalanobis distance corresponding to W_1 from Problem 1. In each plot, you will plot the predicted value of y, given x_1 (horizontal axis) and x_2 (vertical axis), as the color of each point (grayscale between 0 and 1). Include the x_1 and x_2 axes, with tick marks spaced every 0.1 units for $x_1 = 0$ to $x_1 = 1$ and $x_2 = 0$ to $x_2 = 1$.

   For the first three plots, use the kernel-based predictor varying $\alpha = \{0.1, 3, 10\}$. For the next three plots, use the kNN predictor with $\alpha = 1$, $k = \{1, 5, N-1\}$, where N is the size of the data set.

   Print the total least squares loss on the training set for each of the 6 plots.

   You may choose to use some starter Python code to create your plots provided in `T1_P2.py`. Please **write your own implementation of kNN** for full credit. Do not use external libraries to find nearest neighbors.

2. Do any of the kernel-based regression plots look like the 1NN? The $(N-1)$NN? Why or why not?

3. Suppose we are given some W for a Mahalanobis distance or kernel function. Then, in general, there exist values of k for which kernel-based regression and kNN disagree (i.e., make different predictions) on at least one input - for all choices of α. Explain why by means of an example (i.e., show that for some value k of your choosing, no value of α will produce two classifiers that are the same).

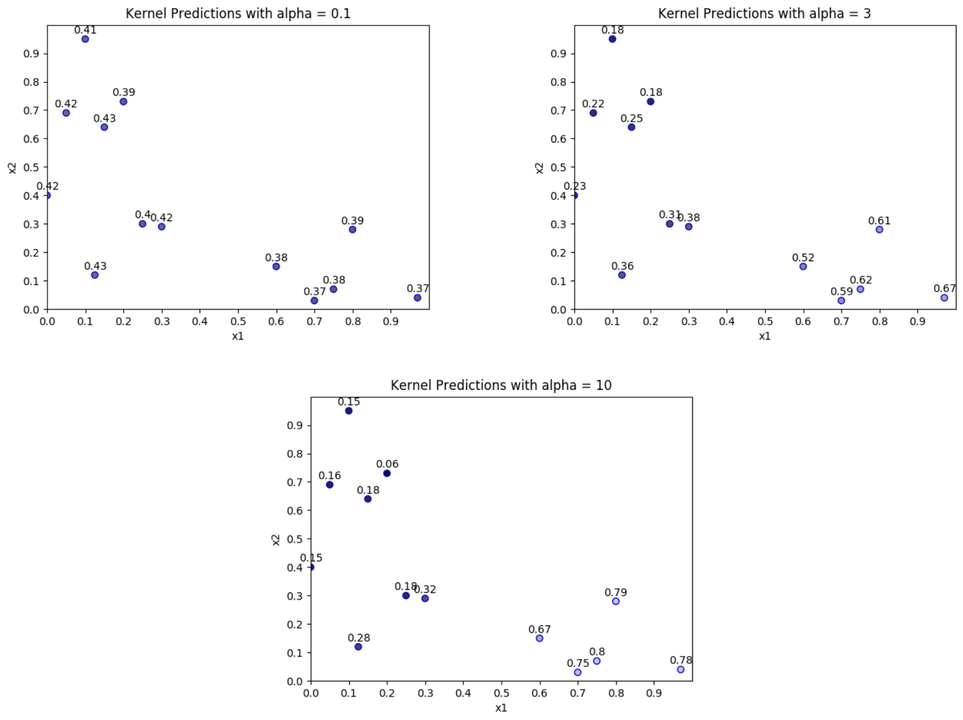
4. Why did we not vary α for the kNN approach?

Solution

1. The total least squares loss on the training set for the kernel-based regressions are as follows in the table (organized based on each regression's α value):

α	$\mathcal{L}(W)$
0.1	1.8399712540879825
3	0.6200161545448001
10	0.39001293585550434

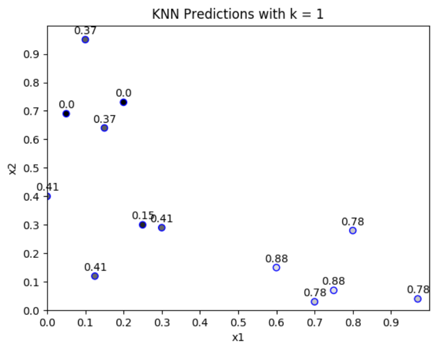
   Our plots for the kernel-based regressions are:

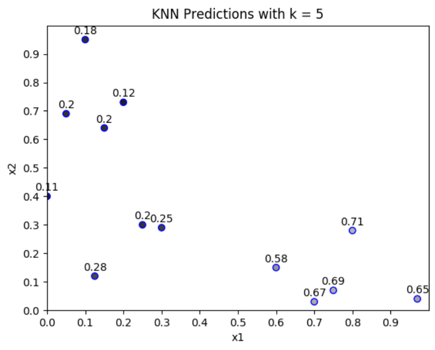


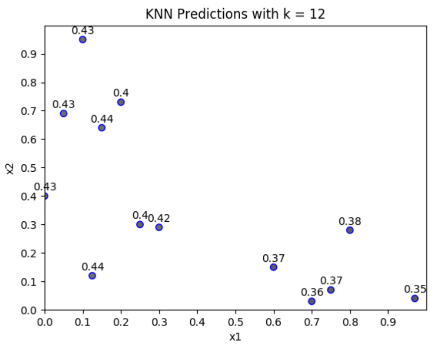
   The total least squares loss on the training set for the kNN regressions are as follows in the table (organized based on each regression's k value):

k	$\mathcal{L}(W)$
1	0.8383999999999999
5	0.4692999999999994
12	1.922573611111111

   Our plots for the kNN regressions are:







2. The kernel-based regression plot where $\alpha = 10$ looks similar to the 1NN regression plot. The kernel-based regression plot where $\alpha = 0.1$ looks similar to the $(N-1)NN$ regression plot. This makes intuitive sense that kernel-based regression plots with bigger α will look more similar to kNN regression plots with smaller k and vice versa. The rationale is that kernel-based regressions with larger α will give greater weights to nearby points than kernel-based regressions with smaller α, all else being equal. This causes kernel-based regressions with larger α to look more similar to kNN regression plots with smaller k since both are focused on just the nearby points. In contrast, kNN regression plots with larger k will equally factor in all the k nearest points, regardless of if the points are near or far away. When k is large, this is akin to a kernel-based regression with a smaller α, since both will give greater weights to points farther away than their counterpart regressions with larger α and smaller k. A kernel-based regression with a smaller α will give more weight to further away points and less weight (relatively) to near points than the same regression with a larger a.

3. Let's choose $k = 1$ such that the kNN regression makes predictions by choosing the closest point (as measured by Mahalanobis distance). The reason why the kernel-based regression and kNN will disagree on at least one input, for all choice of α, is because of their differences on how their handle equidistant points.
   For the sake of our example, suppose we are estimating a point such that there are two or more points equidistant to that point (as measured by Mahalanobis distance). How the kernel-based regression and kNN regression (where $k = 1$) chooses to estimate that point differs, irrespective of whatever α we are using. The kernel-based regression will take the average of all the equidistant points to make a prediction, whereas the kNN regression will randomly choose one of the equidistant points to make its predictions. As long as we ensure that the equidistant points are unique (that is they do not have the exact same x values), then we can guarantee that the kNN regression and the kernel-based regression will not have the same prediction, regardless of the value of α.
   In fact, the choice of α is irrelevant for consideration, because α is just a scalar, and the actual relative distances are decided by the weight matrix W. So if multiple points are equidistant, then the choice of α will not change that the points are still equidistant.

4. The choice of α is meaningless for the kNN approach, since all that α does as a scalar is to scale our distances. However, kNN does not care about the absolute distances of our points, and instead it cares about the relative distances of our points. All α does is rescale our distances, which changes the absolute distances (and thus affects our kernel-based regression), but this doesn't affect our kNN regression because the relative distances are unchanged when we scale by different α, and therefore the predictions that the kNN regression chooses will be the same regardless of the choice of α.