Homework Three

Haley Massa

September 2019

Although this is a programming homework, you only need to hand in a pdf answer file.There is no need to submit the latex source or any code.You can choose any programming language, as long as you implement the algorithm from scratch. Use this latex file as a template to develop your homework.Submit your homework on time as a single pdf file to Canvas.Please check Piazza for updates about the homework.

1 A Simplified 1NN Classifier

You are to implement a 1-nearest-neighbor learner for classification.To simplify your work, your program can assume that

- each item has d continuous features $\mathbf{x} \in^d$
- binary classification and the class label is encoded as $y \in \{0, 1\}$
- data files are in plaintext with one labeled item per line, separated by whitespace:

$$\begin{matrix} x_{11} & \dots & x_{1d} & y_1 \\ & \dots & & \\ x_{n1} & \dots & x_{nd} & y_n \end{matrix}$$

Your program should implement a 1NN classifier:

- Use Mahalanobis distance d_A parametrized by a positive semidefinite (PSD) diagonal matrix A. For $\mathbf{x}, \mathbf{x}' \in^d$,

$$d_A(\mathbf{x}, \mathbf{x}') = \|\mathbf{x} - \mathbf{x}'\|_A = \sqrt{(\mathbf{x} - x')^\top A(\mathbf{x} - x')}.$$

  We will specify A in the questions below. (Hint: d is dimension while d_A with a subscript is distance)

- If multiple training points are the equidistant nearest neighbors of a test point, you may use any one of those training points to predict the label.
- You do not have to implement kd-tree.

2 Questions

1. (5 pts) What is the mathematical condition on the diagonal elements for a diagonal matrix A to be PSD?

   The matrix has to have positive diagonal values and positive eigenvalues.

2. (5 pts) Given a training data set D, how do we preprocess it to make each feature dimension mean 0 and variance 1? (Hint: give the formula for $\hat{\mu}_j, \hat{\sigma}_j$ for each dimension j, and explain how to use them to normalize the data. You may use either the $\frac{1}{n}$ or $\frac{1}{n-1}$ version of sample variance. You may assume the sample variances are non-zero.)

   To pre process the data you have to find the mean and standard deviation of the training data set.

$$\mu_i = \frac{1}{|D|} * \sum_{d=1}^{D} x_i^{(d)}$$

$$\sigma_i = \sqrt{\frac{1}{|D|} * \sum_{d=1}^{D} (x_i^d - \mu_i)^2}$$

   With these values calculated on the training set. You then have to standardize each value of the training set data:

$$\hat{x}_i^d = \frac{x_i - \mu_i}{\sigma_i}$$

   This will normalize all values of the training set. Then when you want to run this on the test set, you will do the exact same procedure. However, you have to make sure you use the SAME values of μ and σ that you did with the training set.

3. (5 pts) Let $\tilde{\mathbf{x}}$ be the preprocessed data. Give the formula for the Euclidean distance between $\tilde{\mathbf{x}}, \tilde{\mathbf{x}}'$.

   The equation for the Euclidean distance between the pre-processed data is :

$$d_{euclidean} = \sqrt{(\frac{x_i - \mu_i}{\sigma_i} - \frac{x_i' - \mu_i}{\sigma_i})^2}$$

which then means the euclidean distance is :

$$d_{euclidean} = \sqrt{(\hat{x} - \hat{x}')^2}$$

4. (5 pts) Give the equivalent Mahalanobis distance on the original data $\mathbf{x},\ x'$ by specifying A. (Hint: you may need $\hat{\mu}_j, \hat{\sigma}_j$)

To find the Mahalanobis distance I first looked at what I need to multiply for the Euclidean.

$(\frac{x_i - \mu_i}{\sigma_i} - \frac{x_i^{'} - \mu_i}{\sigma_i})$

With this we can break it down to

$\frac{x_i}{\sigma_i} - \frac{\mu_i}{\sigma_i} - \frac{x_i^{'}}{\sigma_i} + \frac{\mu_i}{\sigma_i}$

from this we can see the mu fractions cancel out so we need to find an A that can break it down to

$\frac{x_i - x_i^{'}}{\sigma_i}$

with the Mahalanobis distance A to figure this out. Traditionally, the equation calls for a S^{-1} matrix as the A. I am not sure if we want to show the inverse or just technical A. So as a matrix before inverse we would need:

matrix = $\begin{bmatrix} \sigma_1 & \\ & \sigma_2 \end{bmatrix}$

Then when we take the inverse we get:

matrix^{-1} = $\frac{1}{\sigma_1 * \sigma_2} * \begin{bmatrix} \sigma_2^2 & \\ & \sigma_1^2 \end{bmatrix}$

This becomes: $\begin{bmatrix} \frac{1}{\sigma_1^2} & \\ & \frac{1}{\sigma_2^2} \end{bmatrix}$

So then A (when it doesn't represent a matrix that needs to be inverted) becomes:

$\begin{bmatrix} \frac{1}{\sigma_1^2} & & \\ & \frac{1}{\sigma_2^2} & \\ & \cdots & \frac{1}{\sigma_d^2} \end{bmatrix}$

5. (5 pts) Let the diagonal elements of A be $a_{11}, \ldots, a_{dd}$.Define a diagonal matrix L with diagonal $\sqrt{a_{11}}, \ldots, \sqrt{a_{dd}}$.Define $\tilde{\mathbf{x}} = L\mathbf{x}$.Prove that $d_I(\tilde{\mathbf{x}}, \tilde{\mathbf{x}}') = d_A(\mathbf{x}, \mathbf{x}')$ where I is the identity matrix.

$$= \begin{bmatrix} \sqrt{a_{11}} * (\hat{x_1} - \hat{x_1'}) & 0 & ... & 0 \\ 0 & \sqrt{a_{22}} * (\hat{x_2} - \hat{x_2'}) & ... & 0 \\ 0 & ... & 0 & \sqrt{a_{dd}} * (\hat{x_1} - \hat{x_1'}) \end{bmatrix}$$

d = $\sqrt{(\hat{x} - \hat{x'})^T * I * (\hat{x} - \hat{x'})}$

$$= \sqrt{\begin{bmatrix} \sqrt{a_{11}} * (\hat{x_1} - \hat{x_1'}) & 0 & ... & 0 \\ 0 & \sqrt{a_{22}} * (\hat{x_2} - \hat{x_2'}) & ... & 0 \\ 0 & ... & 0 & \sqrt{a_{dd}} * (\hat{x_d} - \hat{x_d'}) \end{bmatrix}^T * \begin{bmatrix} 1 & 0 & ... & 0 \\ 0 & 1 & ... & 0 \\ 0 & ... & 0 & 1 \end{bmatrix} * \begin{bmatrix} \sqrt{a_{11}} * (\hat{x_1} - \hat{x_1'}) & 0 & ... & 0 \\ 0 & \sqrt{a_{22}} * (\hat{x_2} - \hat{x_2'}) & ... & 0 \\ 0 & ... & 0 & \sqrt{a_{dd}} * (\hat{x_d} - \hat{x_d'}) \end{bmatrix}}$$

$$= \sqrt{\begin{bmatrix} a_{11} * (x_1 - x_1')^2 & 0 & ... & 0 \\ 0 & a_{22} * (x_2 - x_2')^2 & ... & 0 \\ 0 & ... & 0 & a_{dd} * (x_d - x_d')^2 \end{bmatrix}}$$

which is equivalent to:

$$\mathrm{d}_A = \sqrt{\begin{bmatrix} (x_1 - x_1') & 0 & ... & 0 \\ 0 & ()x_2 - x_2') & ... & 0 \\ 0 & ... & 0 & ()x_d - x_d') \end{bmatrix}^T * \begin{bmatrix} a_{11} & 0 & ... & 0 \\ 0 & a_{22} & ... & 0 \\ 0 & ... & 0 & a_{dd} \end{bmatrix} * \begin{bmatrix} (x_1 - x_1') & 0 & ... & 0 \\ 0 & ()x_2 - x_2') & ... & 0 \\ 0 & ... & 0 & ()x_d - x_d') \end{bmatrix}}$$

$$= \sqrt{\begin{bmatrix} a_{11} * (x_1 - x_1') & 0 & ... & 0 \\ 0 & a_{22} * (x_2 - x_2') & ... & 0 \\ 0 & ... & 0 & a_{dd} * (x_d - x_d') \end{bmatrix} * \begin{bmatrix} (x_1 - x_1') & 0 & ... & 0 \\ 0 & ()x_2 - x_2') & ... & 0 \\ 0 & ... & 0 & ()x_d - x_d') \end{bmatrix}}$$

$$= \sqrt{\begin{bmatrix} a_{11} * (x_1 - x_1')^2 & 0 & ... & 0 \\ 0 & a_{22} * (x_2 - x_2')^2 & ... & 0 \\ 0 & ... & 0 & a_{dd} * (x_d - x_d')^2 \end{bmatrix}}$$

This proves that both equations come out to the same expression.

6. (5 pts) Geometrically, what does $L\mathbf{x}$ do to the point $\mathbf{x}$? Explain in simple English.

Geometrically $L\mathbf{x}$ points to a change of basis to L. So by multiplying by L, we are projecting the points into the L basis. This basically helps to re-map the data from the coordinates x to coordinates within the basis of L.

7. (10 pts) Let U be any orthogonal matrix.Define $\tilde{\mathbf{x}} = UL\mathbf{x}$.(i) Prove that $d_I(\tilde{\mathbf{x}}, \tilde{\mathbf{x}}') = d_A(\mathbf{x}, \mathbf{x}')$ again. (ii) Geometrically, what does $UL\mathbf{x}$ do to the point $\mathbf{x}$? Explain in simple English.

Calculating : $d_I(\tilde{\mathbf{x}}, \tilde{\mathbf{x}}')$ =

The first part of the calculation looks as follows (having multiplied for Lx in the previous problem):

$$\left(\begin{bmatrix} & U & \end{bmatrix} * \begin{bmatrix} \sqrt{a_{11}} * (\hat{x_1} - \hat{x_1'}) & 0 & ... & 0 \\ 0 & \sqrt{a_{22}} * (\hat{x_2} - \hat{x_2'}) & ... & 0 \\ 0 & ... & 0 & \sqrt{a_{dd}} * (\hat{x_1} - \hat{x_1'}) \end{bmatrix} \right)^T$$

Through transpose rules, we can state that

$$(A * B)^T = B^T * A^T$$

So that first equation becomes:

$$\begin{bmatrix} \sqrt{a_{11}} * (\hat{x_1} - \hat{x_1'}) & 0 & ... & 0 \\ 0 & \sqrt{a_{22}} * (\hat{x_2} - \hat{x_2'}) & ... & 0 \\ 0 & ... & 0 & \sqrt{a_{dd}} * (\hat{x_1} - \hat{x_1'}) \end{bmatrix}^T * \begin{bmatrix} & U & \end{bmatrix}^T$$

This expression multiplied by the identity matrix results in the same expression.

So then it becomes

$$\sqrt{\begin{bmatrix} \sqrt{a_{11}} * (\hat{x_1} - \hat{x_1^{'}}) & 0 & ... & 0 \\ 0 & \sqrt{a_{22}} * (\hat{x_2} - \hat{x_2^{'}}) & ... & 0 \\ 0 & ... & 0 & \sqrt{a_{dd}} * (\hat{x_1} - \hat{x_1^{'}}) \end{bmatrix}^T * \begin{bmatrix} \quad U \quad \end{bmatrix}^T * \begin{bmatrix} \quad U \quad \end{bmatrix} * \begin{bmatrix} \sqrt{a_{11}} * (\hat{x_1} - \hat{x_1^{'}}) & 0 & ... & 0 \\ 0 & \sqrt{a_{22}} * (\hat{x_2} - \hat{x_2^{'}}) & ... & 0 \\ 0 & ... & 0 & \sqrt{a_{dd}} * (\hat{x_1} - \hat{x_1^{'}}) \end{bmatrix}}$$

We know from the properties of an orthogonal matrix, that

$$U^T * U = I$$

where I is the identity matrix.

So this becomes the same calculation as before of

=

$$\sqrt{\begin{bmatrix} \sqrt{a_{11}} * (\hat{x_1} - \hat{x_1^{'}}) & 0 & ... & 0 \\ 0 & \sqrt{a_{22}} * (\hat{x_2} - \hat{x_2^{'}}) & ... & 0 \\ 0 & ... & 0 & \sqrt{a_{dd}} * (\hat{x_d} - \hat{x_d^{'}}) \end{bmatrix}^T * \begin{bmatrix} 1 & 0 & ... & 0 \\ 0 & 1 & ... & 0 \\ 0 & ... & 0 & 1 \end{bmatrix} * \begin{bmatrix} \sqrt{a_{11}} * (\hat{x_1} - \hat{x_1^{'}}) & 0 & ... & 0 \\ 0 & \sqrt{a_{22}} * (\hat{x_2} - \hat{x_2^{'}}) & ... & 0 \\ 0 & ... & 0 & \sqrt{a_{dd}} * (\hat{x_d} - \hat{x_d^{'}}) \end{bmatrix}}$$

From the previous question we know that this is equivalent to :

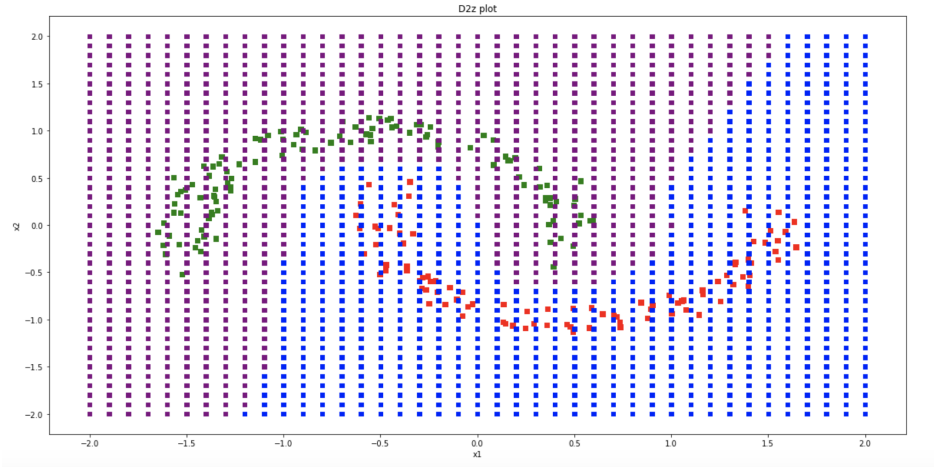
$$= \sqrt{\begin{bmatrix} a_{11} * (x_1 - x_1^{'})^2 & 0 & ... & 0 \\ 0 & a_{22} * (x_2 - x_2^{'})^2 & ... & 0 \\ 0 & ... & 0 & a_{dd} * (x_d - x_d^{'})^2 \end{bmatrix}}$$

It also is equivalent to our DA(X,X$^{'}$)

Geometrically by multiplying by an orthogonal matrix it helps scale instead of remapping. Since it's an orthogonal matrix, the length of the basis are all unit vectors, so it will just rotate the original.

8. (20 pts) Use the whole D2z.txt as training set. Use Euclidean distance (i.e.$A = I$).Visualize the predictions of 1NN on a 2D grid $[-2 : 0.1 : 2]^2$.That is, you should produce test points whose first feature goes over $-2, -1.9, -1.8, \ldots, 1.9, 2$, so does the second feature independent of the first feature.You should overlay the training set in the plot, just make sure we can tell which points are training, which are grid.

[1]



[1]

The training data can be seen in the red and green data points with green representing 1 and red representing 0. The testing data is represented with the colors purple and blue. There the purple represents 1 and the blue represents 0.

9. (To normalize, or not to normalize?)Start from D2a.txt. Perform 5-fold cross validation.

   (a) (5 pts) Do not normalize the data. Report 1NN cross validation error rate for each fold, then the average (that's 6 numbers).

Cross Validation Fold Number	Accuracy in Percentage	Error Rate
1	100	0
2	100	0
3	100	0
4	100	0
5	100	0

This clearly has an average accuracy of 100 percent.

(b) (5 pts) Normalize the data. Report 1NN cross validation error rate (again 6 numbers). (Hints: Do not normalize the labels! The relevant quantities should be estimated from the training portion, but applied to both training and validation portions. This should happen 5 times. Also, you would either change $\mathbf{x}$ into $\tilde{\mathbf{x}} = L\mathbf{x}$ but then use Euclidean distance on $\tilde{\mathbf{x}}$, or do not change $\mathbf{x}$ but use an appropriate A; don't mix the two.)

Cross Validation Fold Number	Accuracy in Percentage	Error Rate
1	92.5	7.5
2	100	0
3	97.5	2.5
4	95	5
5	92.5	7.5

This had an average accuracy of 95.5 and an average error rate of 4.5

(c) (5 pts) Look at D2a.txt, explain the effect of normalization on CV error. Hint: the first 4 features are different than the next 2 features.

The normalization increased the CV error. To explain this we can see that the first 4 features are very different than the last two features. Ideally, it seems the model would be created just on the last two features. However, when we normalize the data we magnify the importance of the first four features.

10. (Again. 10 pts)Repeat the above question, starting from D2b.txt.

Table 1: D2B Table

Cross Validation Number	D2b Original Accuracy	D2b Original Error Rate	D2b Standardized Accuracy	D2b Standardized Error Rate
1	80	20	100	0
2	100	0	100	0
3	82.5	17.5	100	0
4	72.5	27.5	100	0
5	87.5	12.5	100	0
Average	84.5	15.5	100	0

11. (5 pts) What do you learn from Q9 and Q10?

    I learned from these questions that there are different times when you should and shouldn't standardize the data. In Q10, standardizing the data helped because it allowed the data to form two more distinct clusters between points. It helped explain the data better. However, the values of Q9 did not get placed in more distinct areas by standardizing the data. Instead, it made the first four values pollute the data when in an ideal world they would not be used in the formula.

12. (Weka, 10 pts)Repeat Q9 and Q10 with Weka.Convert appropriate data files into ARFF format.Choose classifiers / lazy / IBk.Set $K = 1$.Choose 5-fold cross validation.Let us know what else you needed to set. Compare Weka's results to your Q9 and Q10.

    I needed to go into the settings for the distance function and set DoNotNormalize to True. This means that the distance function does not normalize within itself. This way the only information that is normalized or standardized in any way is the set of standardized training and testing sets for each cross validation.

    As you can see, the Weka has the same patterns seen above in my algorithm. So it helps prove that I was on the right track. In the Weka calculations output, I was able to see the idea that on the ideal models for D2A it used the information from the last two features but none of the first four. This helped my understanding of how standardization polluted the data.

Table 2: Weka for D2A

Cross Validation Number	D2a Original Accuracy	D2a Error Rate	D2A Standardized Accuracy	D2A Standardized Error Rate
1	100	0	97.5	2.5
2	100	0	87.5	12.4
3	100	0	100	0
4	100	0	97.5	2.5
5	100	0	80	20
Average	100	0	92.5	7.5

Table 3: Weka for D2B

Cross Validation Number	D2b Original Accuracy	D2b Original Error Rate	D2b Standardized Accuracy	D2b Standardized Error Rate
1	85	15	97.5	2.5
2	79.4872	20.513	100	0
3	84.6154	15.3846	100	0
4	70	30	97.5	2.5
5	78.5714	21.4286	100	0
Average	79.53	20.47	99	1