HOMEWORK 3

>>Sean(Xiaoyu) Sun<<
>>9078202463<<

Instructions: 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 $\mathbf{x} \in \mathbb{R}^d$
- 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:

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

Your program should implement a 1NN classifier:

- Use Mahalanobis distance d_A parametrized by a positive semidefinite (PSD) diagonal matrix A. For $\mathbf{x}, \mathbf{x}' \in \mathbb{R}^d$,
$$d_A(\mathbf{x}, \mathbf{x}') = \|\mathbf{x} - \mathbf{x}'\|_A = \sqrt{(\mathbf{x} - x')^\top A(\mathbf{x} - x')}.$$
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?
For every real vector x, when $x^T Ax = \sum_{i=1}^n x_i^2 A_i >= 0$, A is PSD. So every element a_{ii} in A need to be non-negative to make A a PSD matrix.
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 make each feature dimension mean 0 and variance 1, we need to use $x'_{di} = \frac{x_{di} - \hat{\mu}_d}{\hat{\sigma}_d}$ for every feature dimension d.
3. (5 pts) Let $\tilde{\mathbf{x}}$ be the preprocessed data. Give the formula for the Euclidean distance between $\tilde{\mathbf{x}}, \tilde{\mathbf{x}}'$.
$D_{Euclidean}(\tilde{\mathbf{x}}, \tilde{\mathbf{x}}') = \sqrt{\sum_{i=1}^n (\tilde{\mathbf{x}}_i - \tilde{\mathbf{x}}'_i)^2}$ where i is the i^{th} dimension of x

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

   $D_{Mahalanobis}(\mathbf{x}, \mathbf{x}') = \sqrt{(\mathbf{x}-\mathbf{x}')^T A(\mathbf{x}-\mathbf{x}')} = \sqrt{\sum_{i=1}^n (\mathbf{x}_i - \mathbf{x}'_i)^2 a_{ii}}$
   $D_{Euclidean}(\tilde{\mathbf{x}}, \tilde{\mathbf{x}}') = \sqrt{\sum_{i=1}^n (\tilde{\mathbf{x}}_i - \tilde{\mathbf{x}}'_i)^2} = \sqrt{\sum_{i=1}^n (\frac{\mathbf{x}_i - \hat{\mu}_i}{\hat{\sigma}_i} - \frac{\mathbf{x}'_i - \hat{\mu}_i}{\hat{\sigma}_i})^2} = \sqrt{\sum_{i=1}^n (\frac{\mathbf{x}_i - \mathbf{x}'_i}{\hat{\sigma}_i})^2}$

   To make $D_{Euclidean}(\tilde{\mathbf{x}}, \tilde{\mathbf{x}}') = D_{Mahalanobis}(\mathbf{x}, \mathbf{x}')$, every a_{ii} in A need to be equal to $\frac{1}{\hat{\sigma}_i}^2$

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.

   $D_I(\tilde{\mathbf{x}}, \tilde{\mathbf{x}}') = \sqrt{\sum_{i=1}^n (\tilde{\mathbf{x}}_i - \tilde{\mathbf{x}}'_i)^2} = \sqrt{\sum_{i=1}^n (\sqrt{a_{ii}}\mathbf{x}_i - \sqrt{a_{ii}}\mathbf{x}'_i)^2} = \sqrt{\sum_{i=1}^n a_{ii}(\mathbf{x}_i - \mathbf{x}'_i)^2}$
   $D_A(\mathbf{x}, \mathbf{x}') = \sqrt{(\mathbf{x}-\mathbf{x}')^T A(\mathbf{x}-\mathbf{x}')} = \sqrt{\sum_{i=1}^n (\mathbf{x}_i - \mathbf{x}'_i)^2 a_{ii}}$

   So, $D_I(\tilde{\mathbf{x}}, \tilde{\mathbf{x}}') = D_A(\mathbf{x}, \mathbf{x}')$

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

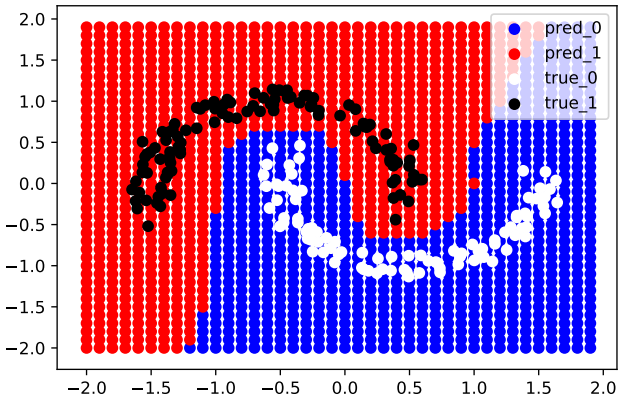
   $L\mathbf{x}$ changes every feature dimension's magnitude. In other words, by applying $L\mathbf{x}$, we can enlarge or condense the value space for every feature by using different a_{ii}. To my uderstanding, $L\mathbf{x}$ can be considered as weight vector.

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.

   (i).
   $D_I(\tilde{\mathbf{x}}, \tilde{\mathbf{x}}') = \sqrt{(\tilde{\mathbf{x}}-\tilde{\mathbf{x}}')^T I(\tilde{\mathbf{x}}-\tilde{\mathbf{x}}')} = \sqrt{(\mathbf{x}-\mathbf{x}')^T L^T U^T U T(\mathbf{x}-\mathbf{x}')}$
   Since U is orthognal matrix, $U^T U = I$ and $L^T I L = A$.
   So $D_I(\tilde{\mathbf{x}}, \tilde{\mathbf{x}}') = \sqrt{(\mathbf{x}-\mathbf{x}')^T A(\mathbf{x}-\mathbf{x}')} = D_A(\mathbf{x}, \mathbf{x}')$

   (ii).
   Still,$L\mathbf{x}$ changes feature's magnitude. By applying orthognal matrix on the left side, we can rotate feature values to a new coordinates without changing the relative Euclidean distance between every corrdinates.

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.



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).

validaiton errors are all zero. The average error is also zero.

(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.)

validaiton errors are [0.075, 0.075, 0.1, 0.05, 0.05]. The average error is 0.07.

(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.

Values for the first 4 features are much smaller than values of the last two features. And the last two features are more important than the first 4 features for making prediction. So without normalization, my 1NN model gives perfect predictions. But after applying normalization, the weights of every features are balaned, which means the model focuses less on the last two important features and focuses more on the first 4 trivial features. That could be the reason why applying normalization leads to worse result.

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

For original dataset: validation errors are [0.275, 0.175, 0.15, 0.175, 0.2], the average error is 0.195.
For normalized dataset: validaiton errors are all zero. The average error is also zero.

Similar to D2a, D2b also has unbalanced feature values. The second feature has much larger value than the first one. However, normalization gives better result this time. My hypohthesis here is that, though having different magnitudes, this two features are equally important for making prediction. So normalization balances the weights of these two features and thus leads to a better result.

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

We should be careful about normalization. Some datasets should be normalized to get better result but some should not be normalized to keep the natural feature weights. In other words, we should be careful about features' weights (or importance) when doing pre-processing.

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.

D2a: 0.085
D2a_norm(Weka normalized version): 0.055
D2a_norm(my own normalized version): 0.055
D2b: 0.05
D2b_norm(Weka normalized version): 0.00
D2b_norm(My own normalized version): 0.00

"Weka normalized version" means applying the "Normalize" fliter in preprcoessing module inside Weka on the orginal D2a and D2b datasets. I notice that passing normalized version and unnormalized dataset give same resutls for both datasets D2a and D2b.
Weka's D2a results are different from what I got by using my 1NN model on unnormalizaed data but it's close to my normalized D2a result, and its D2b results is same as my normalized D2b result. So based on this, my guess about this result is that Weka applies auto-normalization to datasets.