~~tc — dec. stat. surf.~~

Emma
LaPorte

~~start @ #4~~

· $6.7

Pattern Classification and Recognition
ECE 681

Spring 2019
Homework #4: Cross-Validation, Bayes Classifiers, and the DLRT

Due: 5:00 PM, Thursday, February 28, 2019
Grace Period Concludes: 11:30 PM, Tuesday, March 5, 2019

This homework assignment is worth **300 points**.
Each problem is worth some multiple of 10 points, and will be scored on the below letter scale.
The letter grades B through D may be modified by + (+3%) and A through D may be modified by a - (-3%).

- A+ = 100%: Exceeds expectations, and no issues identified
- A = 95%: Meets expectations, and (perhaps) minor/subtle issues
- B = 85%: Issues that need to be addressed
- C = 75%: Significant issues that must be addressed
- D = 65%: Major issues, but with noticeable perceived effort
- F = 50%: Major issues, and insufficient perceived effort
- Z = 30%: Minimal perceived effort
- N = 0%: Missing, or no (or virtually no) perceived effort

Your homework is not considered submitted until both components (**one self-contained pdf file** and your code) have been submitted. Please do not include a print-out of your code in the pdf file.

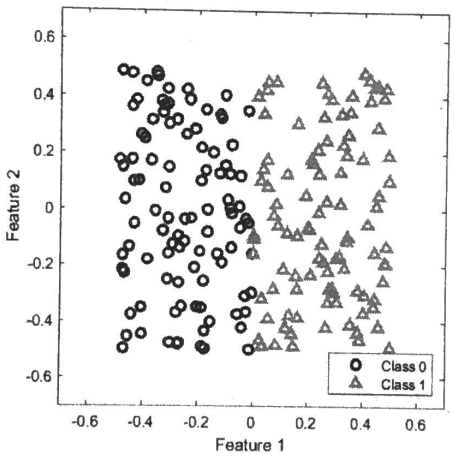
You should strive to submit this assignment by the due date/time. The grace period is intended to afford an opportunity to, for example, work through technical glitches, update your submission if you realize after the due date that you submitted the wrong file or would prefer to answer a question differently, and provide some flexibility to manage your workload as it ebbs and flows during the semester.

Exploring Cross-Validation

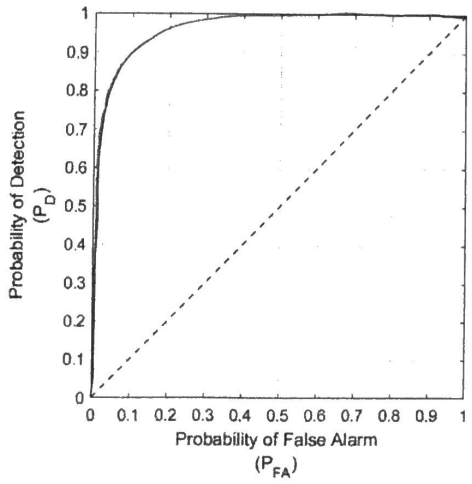
Make sure you are able to apply cross-validation to (more fairly) estimate classifier performance using the available training data.

Regardless of whether you choose to write your own functions or leverage functions that may be available through Matlab or Python packages or libraries, you are responsible for understanding how the function(s) you are using work so you can effectively apply them to suit your needs and correctly interpret the results they provide.

The following questions concern a data set provided as a `csv` file that contains both data and suggested folds for cross-validation, `dataSetCrossValWithKeys.csv`. This csv file is organized such that each row contains the fold assignment (either 1 or 2), followed by the true class (either 0 or 1), followed by the associated (2-dimensional) feature vector. When you visualize the data set, you should see this:

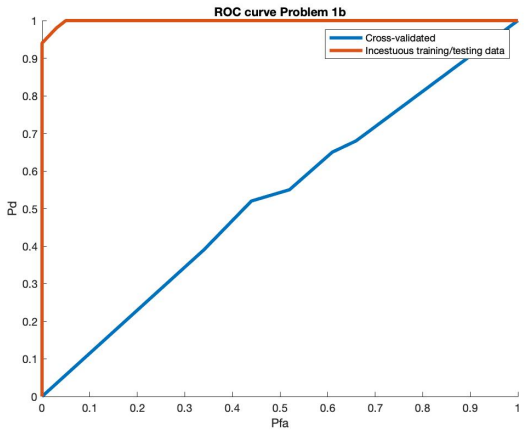


(60) 1. (a) From visual inspection of this dataset (figure above), qualitatively sketch the ROC you would expect to represent performance on this dataset by a KNN classifier with $k = 5$.



I would expect very good performance on this dataset due to the inherent lack of overlap.

(b) Find the cross-validated ROC when a KNN classifier with $k = 5$ is applied to this data, using the folds specified as part of the dataset, and compare that ROC to the ROC found via incestuous training and testing.

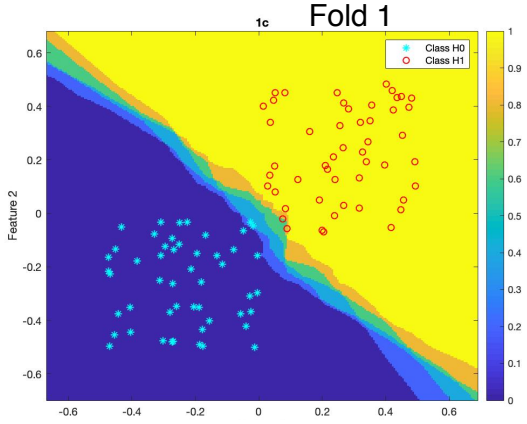


(c) Explain why the ROC found via cross-validation differs from the ROC you (qualitatively) expected. (Decision statistic surfaces for the two folds individually will likely be part of your explanation.)

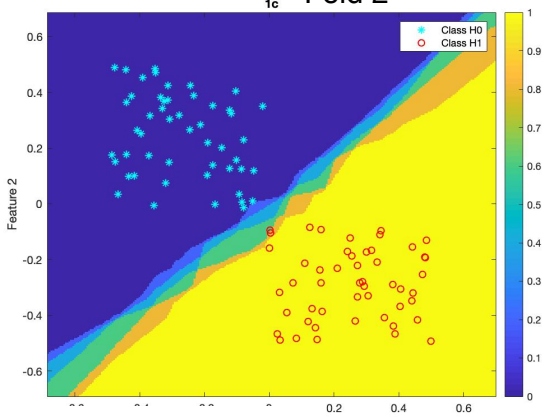
split up by key.

Based on how the folds are assigned, the algorithm performance will vary greatly. It is important that the folds are randomly assigned to ensure each set adequately represents the data.

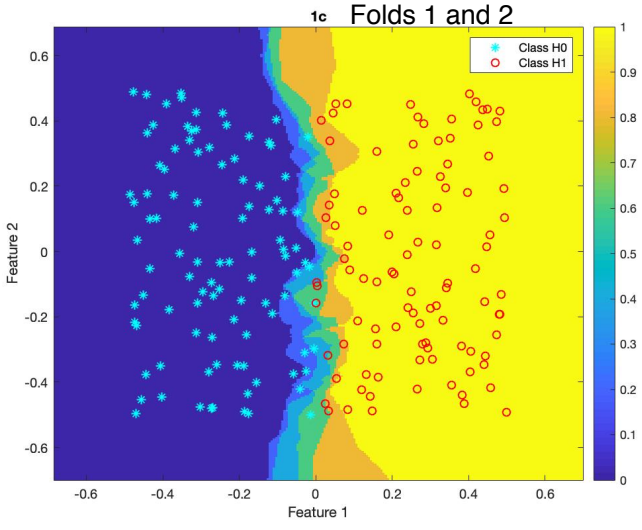
Fold 1



Fold 2



Folds 1 and 2



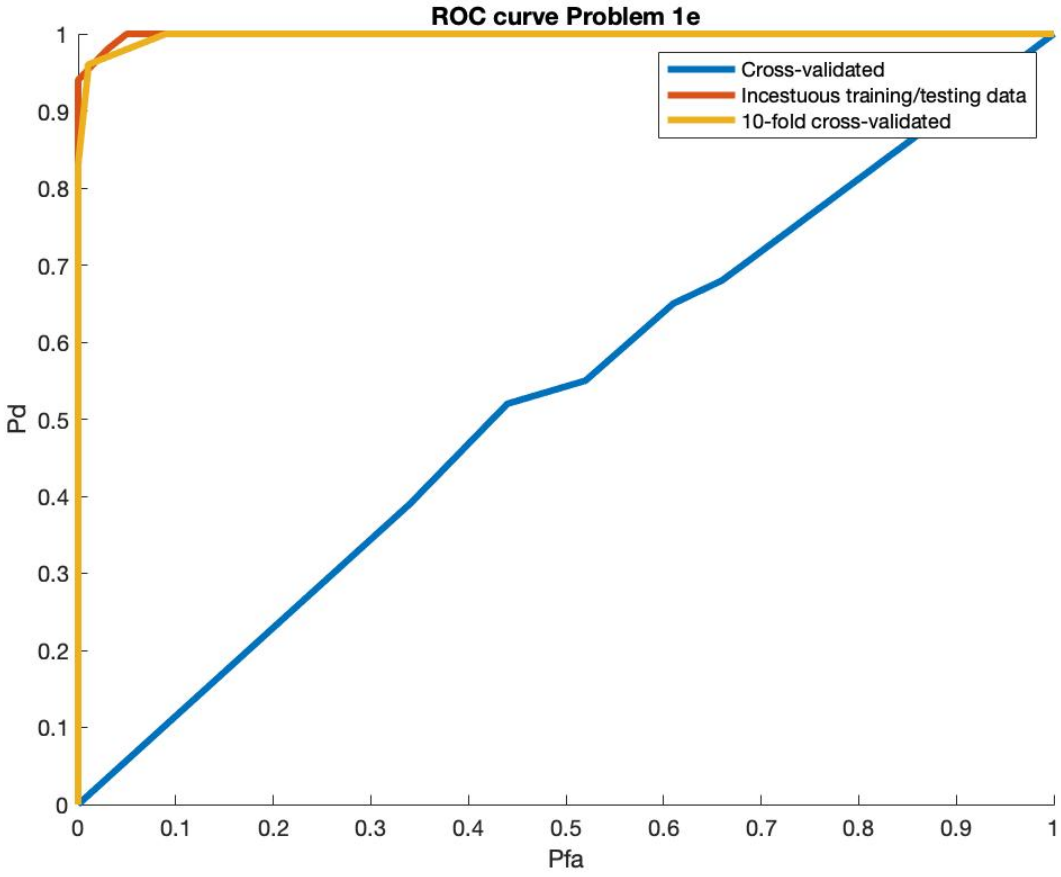
As is seen, neither fold1 nor fold 2 accurately represnts the overall dataset seen in both folds 1 and 2.

(d) How would you modify the cross-validation on this dataset to achieve ROCs that provide more reasonable representative estimate of classifier performance for a KNN classifier with $k = 5$?

I would increase the # of folds, as is typical, to 5 or 10-fold cross-validation. and ensure the folds are representative of the two classes.

By splitting up the data randomly into more folds, each fold will better represent the overall distribution of the data, allowing for an overall more accurate average over all folds.

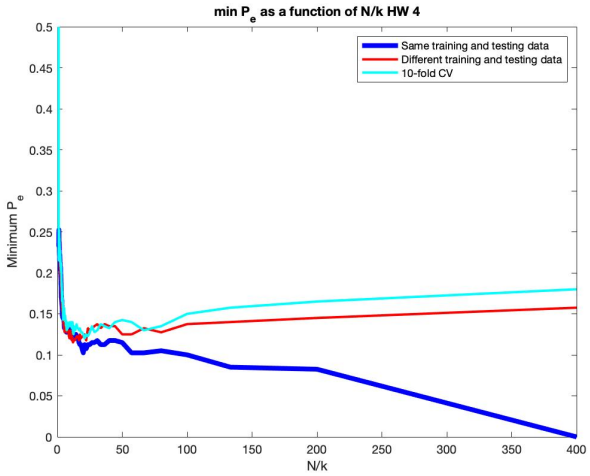
(e) Implement your proposed modification to the cross-validation, and compare your cross-validated ROC when a KNN classifier with $k = 5$ is applied to this data to the ROC found via incestuous training and testing.



Applying Cross-Validation to Further Explore Bias-Variance Tradeoff

(40) 2. This question concerns the `dataSetHorseshoes.csv` data set that was previously explored in Homework #2 (KNN and Bias-Variance Tradeoff). This csv file is organized such that each row contains the true class (either 0 or 1) followed by the associated (2-dimensional) feature vector.

(a) Replicate the plot of $\min P_e (= 1 - \max P_{cd})$ as a function of N/k for a sampling of values of k[1] between 1 and 399 for both testing on the training data and testing on separate testing data that you created for Homework #2, and add to this figure $\min P_e$ for 10-folds cross-validated performance on the training data.[2]



(b) Explain how the graph of cross-validated performance should be interpreted.

The graph of C-V performance speaks to bias-variance tradeoff. At very small values of $\frac{N}{K} \rightarrow 1$, min P_e spikes, due to oversimplification of the model. As N/K increases, the min P_e decreases reaching optimal low pt, then P_e starts to rise due to overfitting and high variance of the model.

(c) Assume you do not have separate test data available to you, how would you select a value of k to recommend to maximize P_{cd}?

Cross-validate the training dataset and iterate thru different values of K and find corresponding P_{CD}. Choose the K that maximizes P_{CD}.

[1] You choose the values of k where you want to sample this curve.

[2] The flexibility of a KNN classifier is proportional to N/k; $k = 1$ provides the greatest flexibility and $k = N$ provides the least flexibility.

Exploring Bayes Classifiers

Make sure you are able to apply a Bayes classifier and that you have flexibility to specify the assumptions you wish to make regarding the covariance structure of the data, *i.e.*, unique covariance matrices for each class, a common covariance matrix for both classes, independence among features (a diagonal covariance matrix), independence and common scaling among features ($\Sigma = \sigma^2 \mathbf{I}$).

Regardless of whether you choose to write your own functions or leverage functions that may be available through standard Matlab or Python packages, it is critical that you understand how the function(s) you are using work so you can effectively apply them to suit your needs and correctly interpret the results they provide.

mvnrnd(mu, sigma)

(10) 3. Generate a 2-dimensional dataset for two distinct classes (class 0 and class 1) such that the dataset satisfies the following properties:

- the covariance matrix[3] for class 0 has a positive correlation, and $\sigma_0^2 \neq \sigma_1^2$
- the covariance matrix for class 1 has a negative correlation, and $\sigma_0^2 \neq \sigma_1^2$
- $\mu_0 \neq \mu_1$, but the distributions overlap noticeably

Provide the parameters for your dataset (μ_0, Σ_0, μ_1, Σ_1), and visualize it as a scatter plot of the data points, clearly indicating the associated class for each data point.

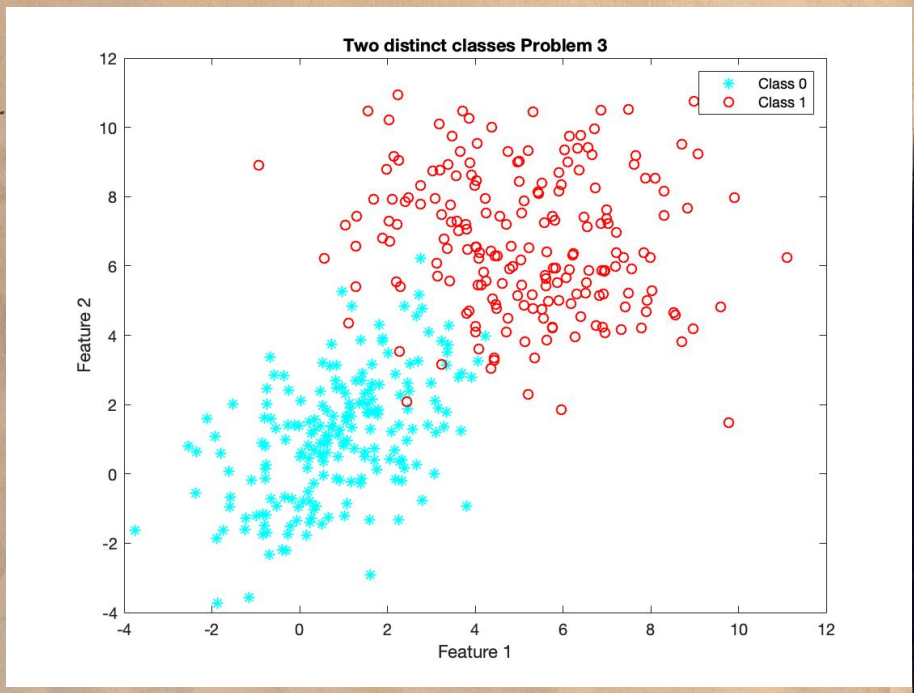
$$\mu_0 = [1 \quad 1]$$

$$\Sigma_0 = \begin{bmatrix} 2 & 1 \\ 1 & 3 \end{bmatrix} \Rightarrow \text{positive correl.}$$

$$\mu_1 = [5 \quad 7]$$

$$\Sigma_1 = \begin{bmatrix} 5 & -1 \\ -1 & 4 \end{bmatrix} \Rightarrow \text{negative correl.}$$

$\sigma_0^2 \neq \sigma_1^2$

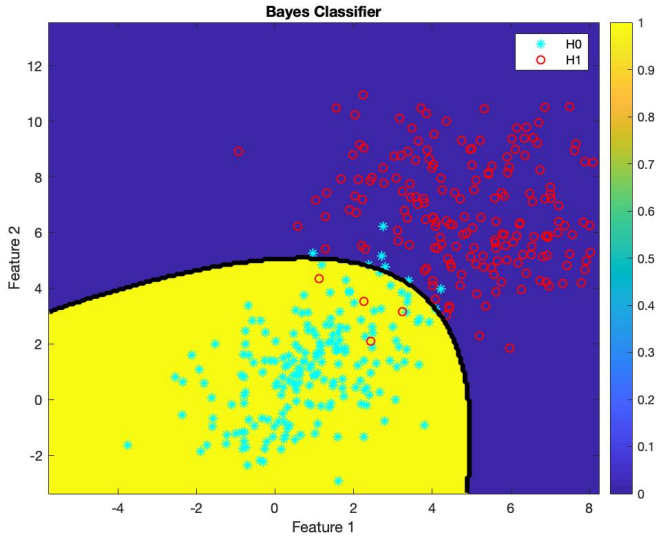


[3] Recall that covariance matrices can be decomposed into components that characterize the "directions" in which the data lie and the strengths of those "directions", so you can define your covariance matrices in terms of this decomposition, and then synthesize the covariance matrices.

(80) 4. Assume you are training a Bayes classifier (*i.e.*, likelihood ratio) under various assumptions regarding the covariance structure of the data.

features dependent
covar. unique

(a) Apply a Bayes Classifier to your dataset, making no simplifying assumptions regarding the covariance structure (*i.e.*, the features may be dependent, and the covariance matrices are unique), and plot the decision statistic surface[4] with both the training data and the decision boundary under the assumptions of equal class priors and symmetric costs superimposed on top.



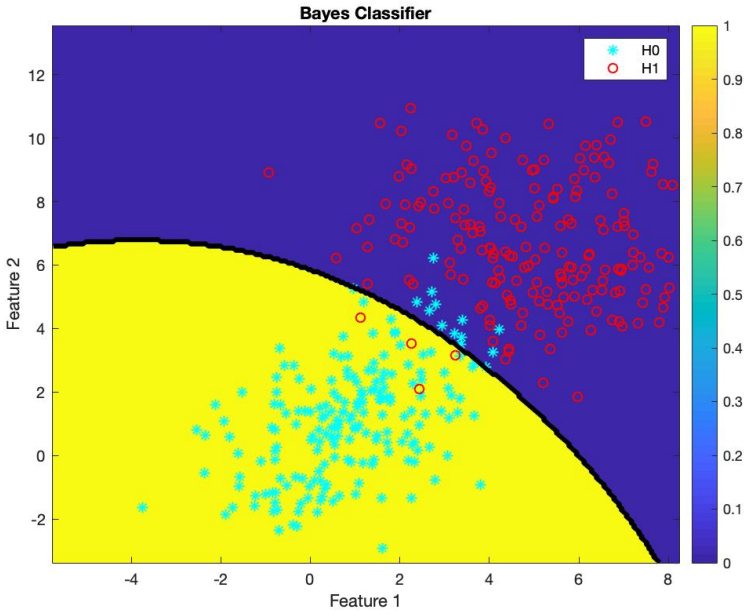
(b) How do the assumptions made regarding the covariance structure of the data while training the classifier affect the resulting decision boundary?

No assumptions were made of the dataset. Therefore, in the training, the covariances and means were calculated directly from each class, with no alterations. Thus, the Bayes classifier represented the dataset better, producing a more specific and flexible classifier.

[4] Visualizing the ln-likelihood ratio is usually preferred as it is easier to interpret visually than the likelihood ratio.

feat - ind.
covar - unique

(c) Apply a Bayes Classifier to your dataset, under the simplifying assumption that the features are independent but the covariance matrices under each class are unique, and plot the decision statistic surface with both the training data and the decision boundary under the assumptions of equal class priors and symmetric costs superimposed on top.



(d) How do the assumptions made regarding the covariance structure of the data while training the classifier affect the resulting decision boundary?

The features were assumed independent by setting the the covariance values in the covariance matrices to 0.

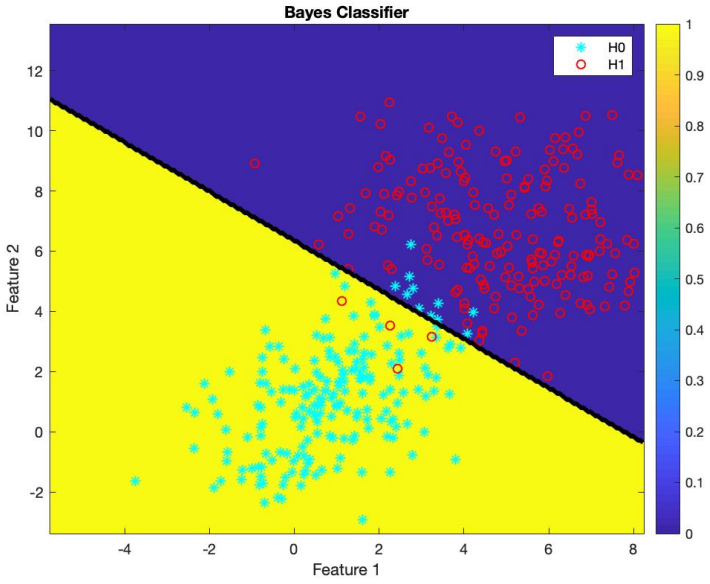
This assumption did not reflect the dataset, and therefore caused a simplification of the classifier.

The decision boundary therefore becomes increasingly linear.

cov(F1+F2)
combine them to get one.

(e) Apply a Bayes Classifier to your dataset, under the simplifying assumption that the covariance matrices under both classes are the same but the features may be dependent, and plot the decision statistic surface with both the training data and the decision boundary under the assumptions of equal class priors and symmetric costs superimposed on top.

- feat - dep.
- covar - same
- equal class priors, Symm costs



(f) How do the assumptions made regarding the covariance structure of the data while training the classifier affect the resulting decision boundary?

(after de-meaning the data)

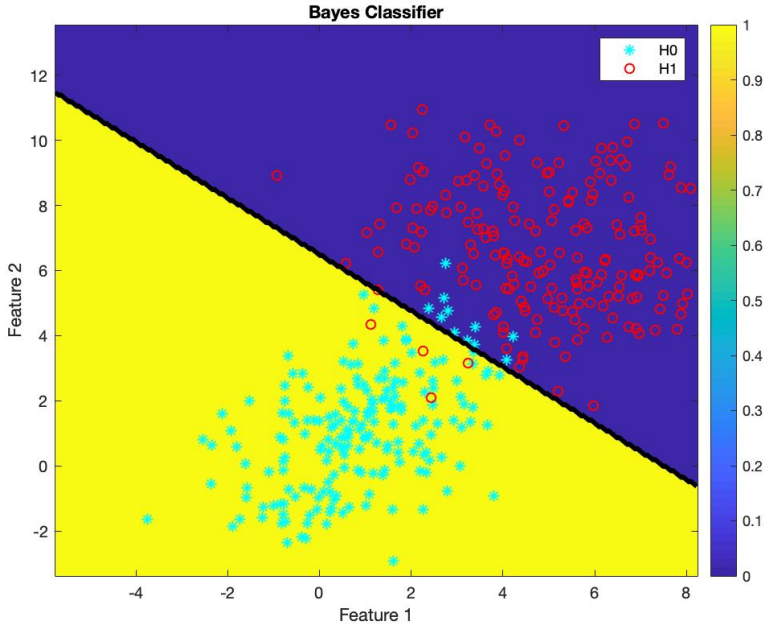
The class 0 and class 1 datasets were combined to calculate an overall covariance matrix. This causes the classifier to lose some of the distinctions between the individual classes, creating a more simplified classifier. Thus the decision boundary becomes more linear.

add datasets →

$\begin{bmatrix} \text{var}(F_1) & 0 \\ 0 & \text{var}(F_2) \end{bmatrix}$

- cov - same → diagonal covariance matrice
- feat - ind.
- equal class priors, symm. costs.

(g) Apply a Bayes Classifier to your dataset, under the simplifying assumptions that the covariance matrices under both classes are the same and the features are independent, and plot the decision statistic surface with both the training data and the decision boundary under the assumptions of equal class priors and symmetric costs superimposed on top.



(h) How do the assumptions made regarding the covariance structure of the data while training the classifier affect the resulting decision boundary?

The covariance matrices were both set to the covariance matrix calculated from the sum of the two datasets. Then to assume feature independence, the covariance values were set to 0. This, again, doses many of the nuances of the data and the distinctions between the two classes. This results in a more simplified classifier and thus an increasingly linear decision boundary.

(10) 5. Describe the factors you would consider when deciding how to model the data (what assumptions to make regarding the covariance structure) for a Bayes classifier, and how you expect your classifier design choices (modeling assumptions) to affect the resulting classifier.

In all of the previous examples, making simplifying assumptions (that is not true of the underlying data produces a more simplified classifier, with higher bias.

The assumptions that we altered in this problem (features dependent or independent, covariance matrix same or different) were assumptions that did not represent the data. Therefore, it resulting classifier became simplified, losing some of the specific nuances of the dataset and the distinctions between the classes.

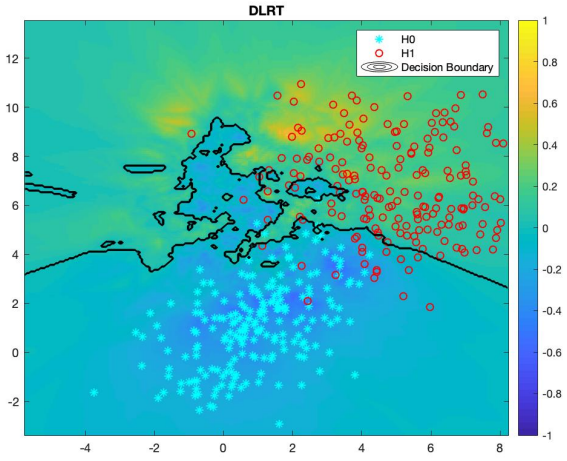
Exploring the DLRT

Make sure you are able to apply the Distance Likelihood Ratio Test (DLRT).

Regardless of whether you choose to write your own functions or leverage functions that may be available through standard Matlab or Python packages, it is critical that you understand how the function(s) you are using work so you can effectively apply them to suit your needs and correctly interpret the results they provide.

(40) 6. The following questions concern the data set you created to explore Bayes classifiers.

(a) Apply the DLRT to your dataset with $k = 5$, and plot the decision statistic surface with both the training data and the decision boundary under the assumptions of equal class priors and symmetric costs superimposed in top.

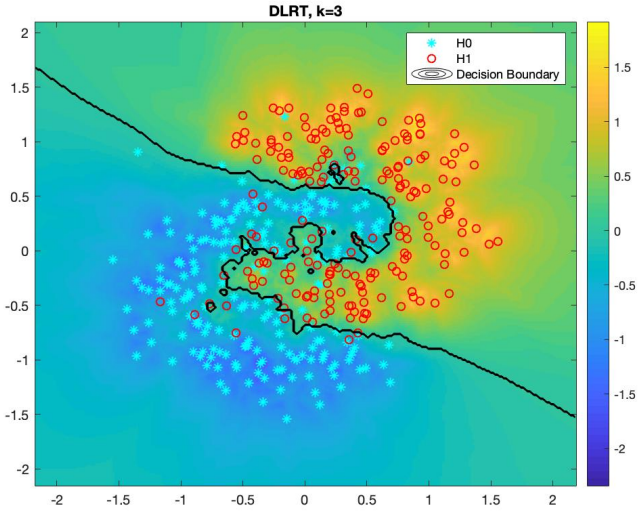


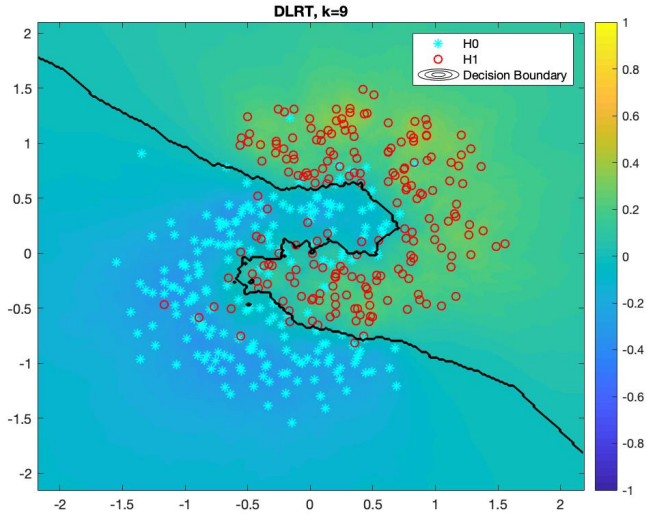
(b) Compare the DLRT decision statistic surface to the Bayes decision statistic surfaces. What do you perceive as relative advantages and disadvantages of the DLRT as compared to Bayes?

The DLRT decision statistic surface and decision boundary showed much greater classifier flexibility than the Bayes. The DLRT was more ~~fitted~~ to the training data, potentially at risk for high variance. It is important to note that w/ changing values of K, the results of the DLRT will alter.

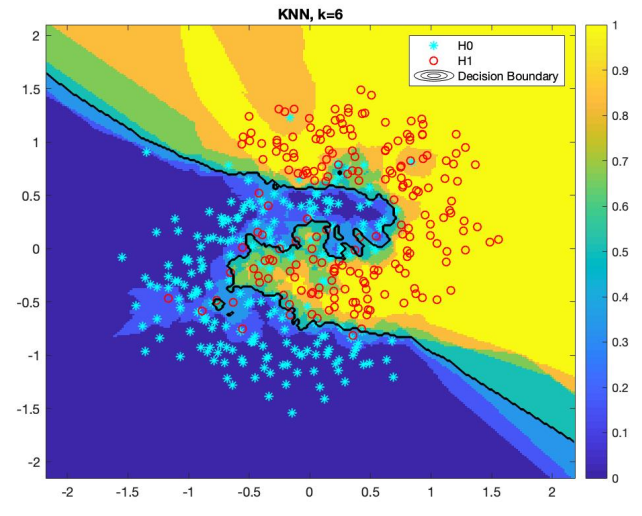
(60) 7. The following questions concern the `dataSetHorseshoes.csv` data set that was previously explored in Homework #2 (KNN and Bias-Variance Tradeoff). This csv file is organized such that each row contains the true class (either 0 or 1) followed by the associated (2-dimensional) feature vector.

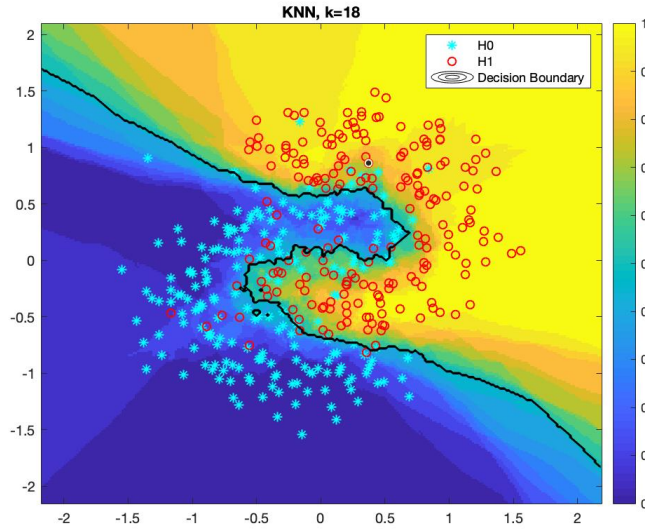
(a) Visualize the decision statistic surface with both the training data and decision boundary under the assumptions of equal class priors and symmetric costs superimposed on top of the decision statistic surface for the DLRT with $k = 3$ and $k = 9$.





(b) Visualize the decision statistic surface with both the training data and majority vote decision boundary (assume the decision is class 1 if the decision statistic is greater than or equal to 0.5) superimposed on top of the decision statistic surface for the KNN classifier using the L_2 distance metric with $k = 6$ and $k = 18$.[5]





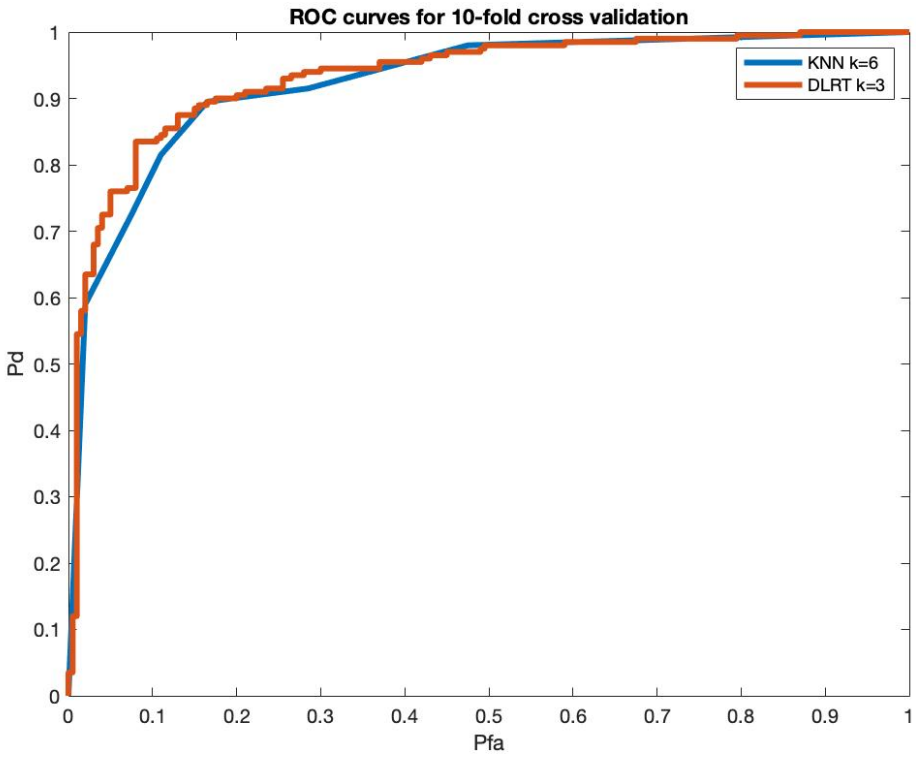
[5]The DLRT takes into account the distance to the k^{th} neighbor in both class 0 and in class 1, and so is making use of $2k$ neighbors when it is parameterized by k. For a fair comparison we want to use the same number of neighbors for both the DLRT and KNN, and so we use $2k$ neighbors for KNN.

(c) What are your impressions of the DLRT decision statistic surfaces as compared to the KNN decision statistic surfaces?

For both DLRT and KNN, the decision boundary gets smoother with increasing k, as expected. With increasing k the model gains simplicity as well as more and more bias and less variance.

The ultimate decision boundaries appear to be very similar when comparing KNN and DLRT, but there are minute differences. At the larger values of k, the KNN and DLRT seem to be more similar. However, at smaller values of k, the differences between KNN and DLRT are more obvious.

(d) Find cross-validated ROCs for the DLRT with $k = 3$, and the KNN with $k = 6$.



(e) What are your impressions of DLRT performance as compared to KNN performance? Under what conditions might you be inclined to prefer one classifier over the other?[6]

As displayed by the ROC curves above, the DLRT performs slightly better than the KNN in this situation. The difference between KNN and DLRT is that DLRT takes into account the distance to the kth neighbor. Therefore, in the extreme case where the next closest neighbor is very far away, it will be counted as "less" than a next closest neighbor that is very close. Therefore, with very spread out data with lots of outliers, DLRT might better account for this than KNN.

Additionally, because KNN does not take into account distance, if the k encapsulates enough values it might falsely classify the test point as something that it isn't, even though visually it should clearly be a specific class. This is because KNN does not account for the distance to the nearest neighbor. DLRT does account for this, so will perform better in this case. This concept is exemplified in the Bayes Classifier lecture slides page 15.

[6]Feel free to further explore and compare the DLRT and KNN in additional scenarios to inform your answer to this question.