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.

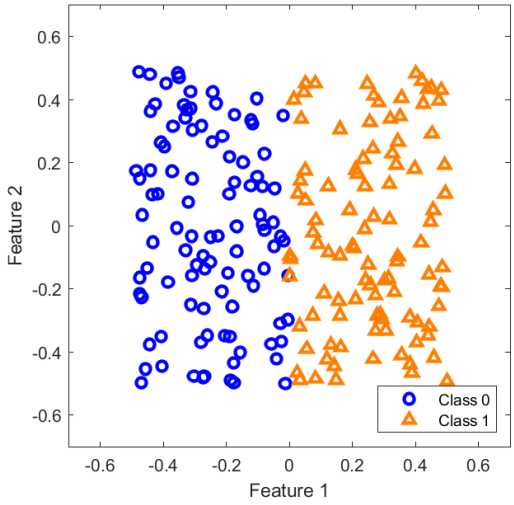
You should strive to submit this assignment by the due date/time. The grace period is intended to afford an opportunity to, for example, work through technical glitches, update your submission if you realize after the due date that you submitted the wrong file or would prefer to answer a question differently,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

Exploring Cross-Validation

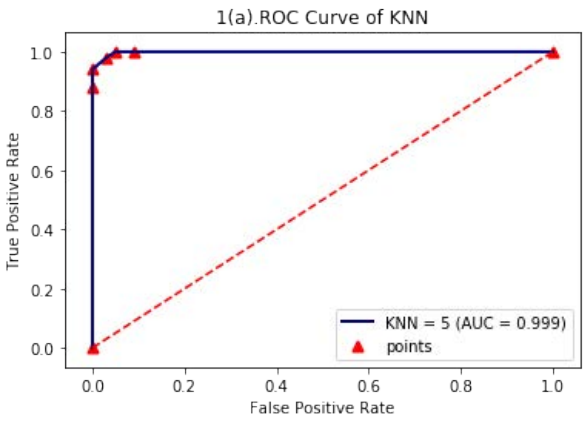
Make sure you are able to apply cross-validation to (more fairly) estimate classifier performance using the available training data.

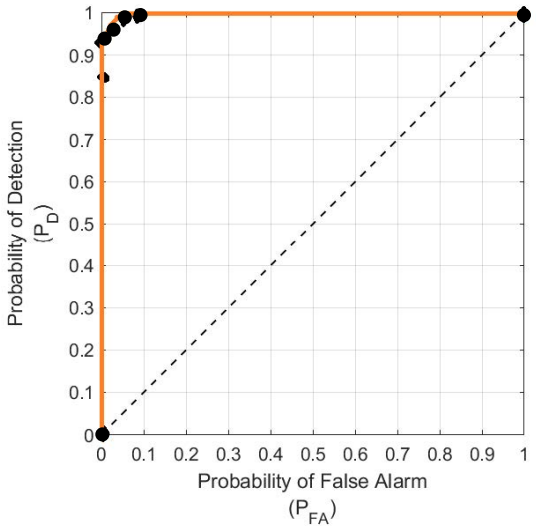
Regardless of whether you choose to write your own functions or leverage functions that may be available through Matlab or Python packages or libraries, you are responsible for understanding how the function(s) you are using work so you can effectively apply them to suit your needs and correctly interpret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

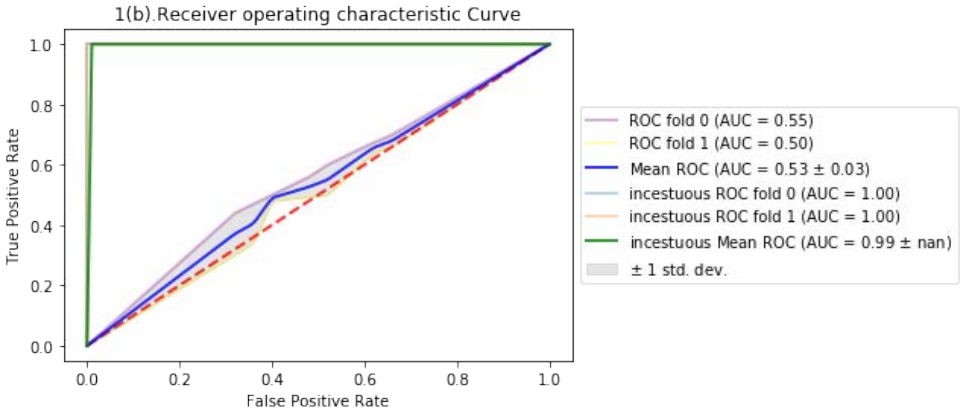


(60) 1. (a) From visual inspection of this dataset (figure above), qualitatively sketch the ROC you would expect to represent performance on this dataset by a KNN classifier with $k = 5$.

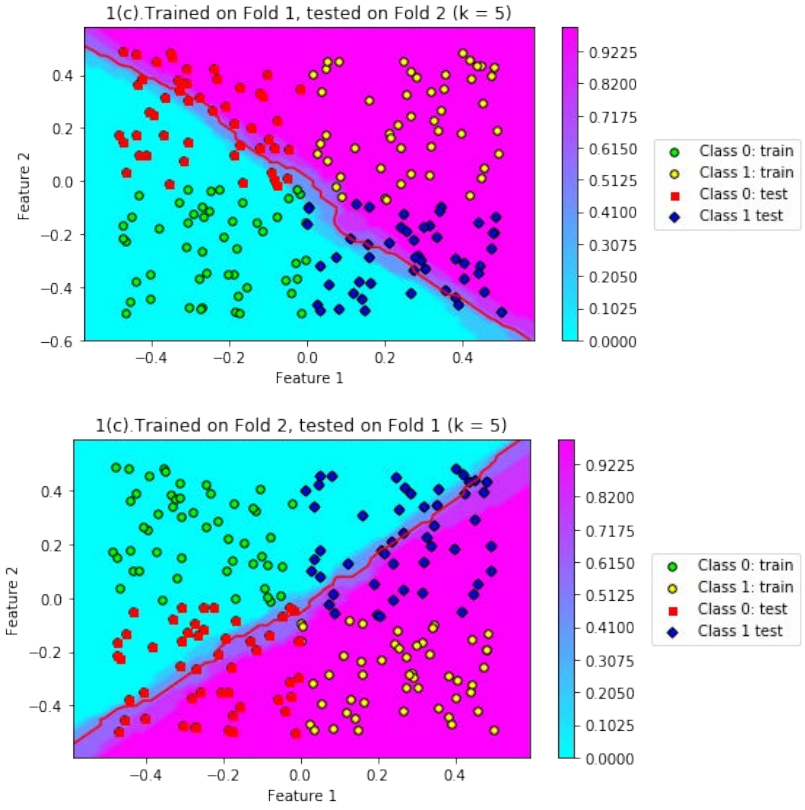




(b) Find the cross-validated ROC when a KNN classifier with $k = 5$ is applied to this data, using the folds specified as part of the dataset, and compare that ROC to the ROC found via incestuous training and testing.



(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.)



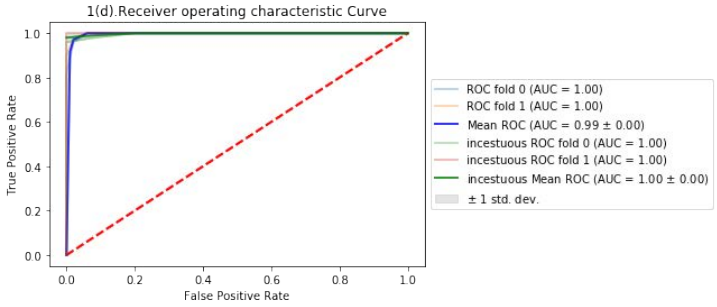
(c) **Answer:**

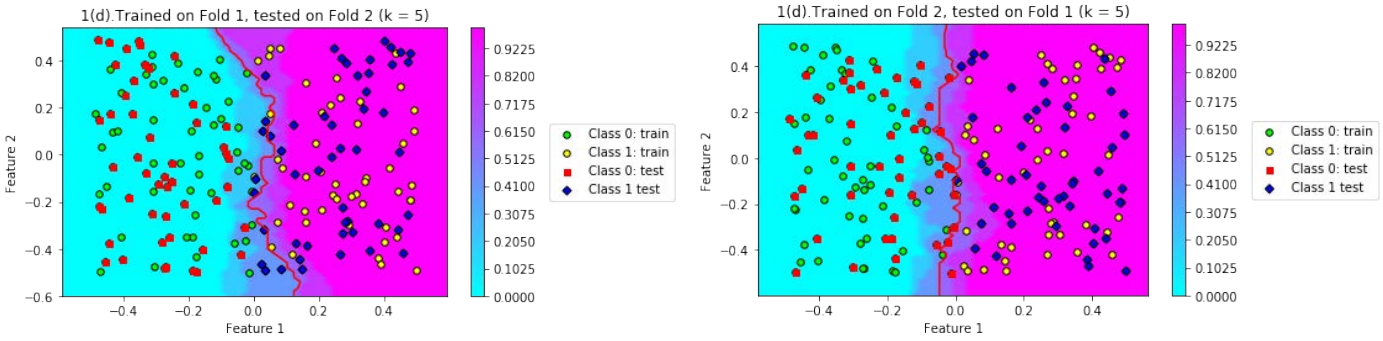
Here I plotted the decision statistic surfaces for both specified fold assignments. The major reason is that the assigned fold label is **not randomly selected**. We could see that a clear characteristic for both fold assignment data is that these data are distributed diagonally. That is for each fold assignment, the H0 and H1 data are **diagonally distributed**. So when we use KNN to train our train data (restricted in a specific fold assignment), we get similar (±)45 degree majority decision boundary, which split the test data **half of H1 as H0**, and **half of H0 as H1**.

Therefore, the fpr and tpr would around 50%, and AUC would result in around 0.5 as we only get 50% correctly prediction.

While the ROC we qualitatively expected, we get a big picture of the whole dataset, and we have no bias to separate the data. So we clearly could use Feature 1 to classify our data. When the threshold move from left to right along the x-axis (Feature 1), we could use Feture 1 as λ to determine tpr and fpr, finally we would get the ROC curve as 1(a).

(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 $k = 5$?





(d) **Answer:**

StratifiedKFold(n_splits = n_splits, shuffle=True, random_state=0)

To modify the cross-validation model, we won't use the specified fold assignments as given. Instead, we need to split our train and test dataset randomly balanced. We could use sklearn.model_selection.**StratifiedKFold** to split the dataset.

StratifiedKFold is useful as a direct k-fold cross-validation operator: as in it will set up n_folds training/testing sets such that classes are equally balanced in both.

With shuffle = True, the data is shuffled by random_state. Otherwise, the data is shuffled by np.random (as default).

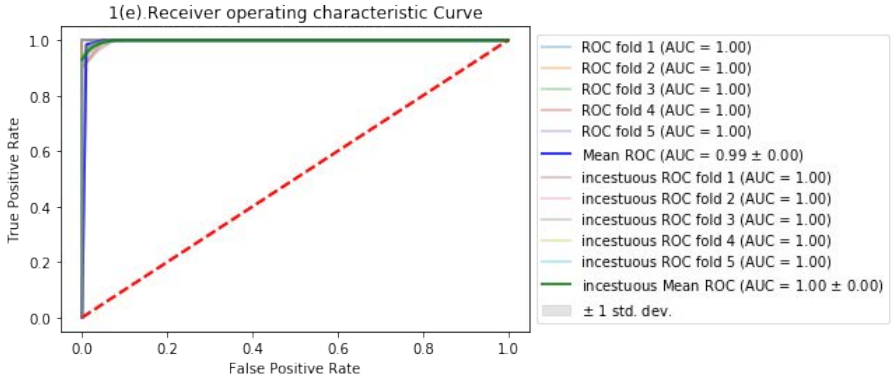
Here I plotted the 2-fold using StratifiedKFold, we could see that each fold iteration, the training and testing dataset are equally balanced in both. Therefore, we received a high AUC for its ROC curve.

To get better performance, we could do 5 folds using StratifiedKFold, and train and test by KNN for the dataset.

(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.

(e) **Answer:** Below I plotted the ROC curve for 5-fold using StratifiedKFold and KNN. We could see that the mean_ROC of 5-folds cross validation and ROC via incestuous training and testing is mostly overlapped.

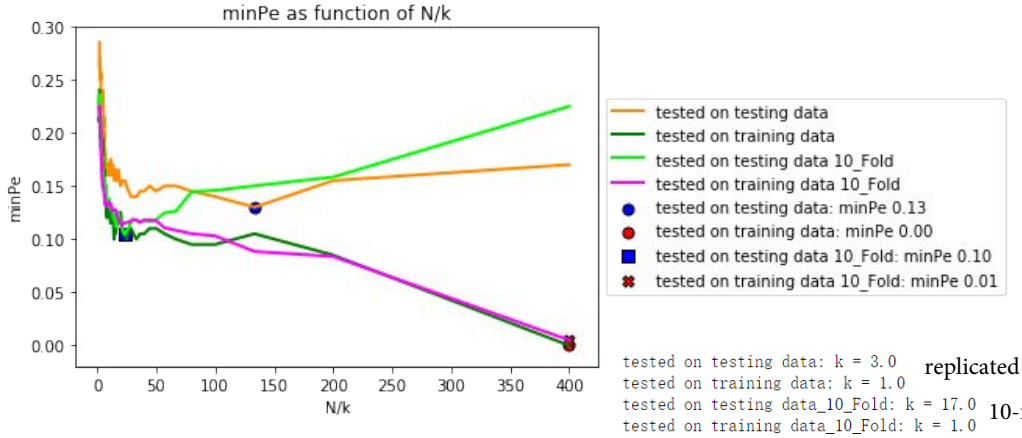
They both get high (almost 1) AUC, which means our modification gets pretty good performance.



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 $\text{minP}_e\,(=1-\max \text{P}_{cd})$ 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 minP_e for 10-folds cross-validated performance on the training data.[2]



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

Answer:
As we could see from the graph as above, the line " tested on testing data 10_fold" is the minPe we are looking for. The 10 Fold works like: each time we separate the dataset into 10 folds, 9 folds will be taken as training data, the other 1 fold will be taken as testing data. And we ran this process 10 times, and get the mean of minPe for N/k. The green line here has similar trend as previous work in HW2, but when k is small, 10 fold has higher minPe than previous work(no fold cross-validation), when k increased to around 5, 10 fold minPe dropped to a lower place than previous work, and it has lower minPe as 0.10. That is because when k is smaller than 5, 10 fold iteratively train and test on the whole dataset, which is more likely to cause overfitting problem when k is small. However, when k is bigger than 5, it started to have lower minPe, and reached to the lowest point when k is 17. It is because the training model taking into account of neighbors which reduce overfitting problems. With high value of k, minPe increased dramatically, because classifier is becoming more complex and no flexibility.

(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}?

Answer:
As we could see from the above graph, even when we don't have test data separately, we could still use KFold cross-validation to train our model, because we would test on one fold each time. And it results in similar minPe (maximize Pcd) trends of N/k vs available test data, the maximum difference between 10 Fold and no-cross-validation in above case, is when k is 1, difference is around 0.1 in minPe.
In these case I would recommend k = 17±3 to maximize Pcd.

[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

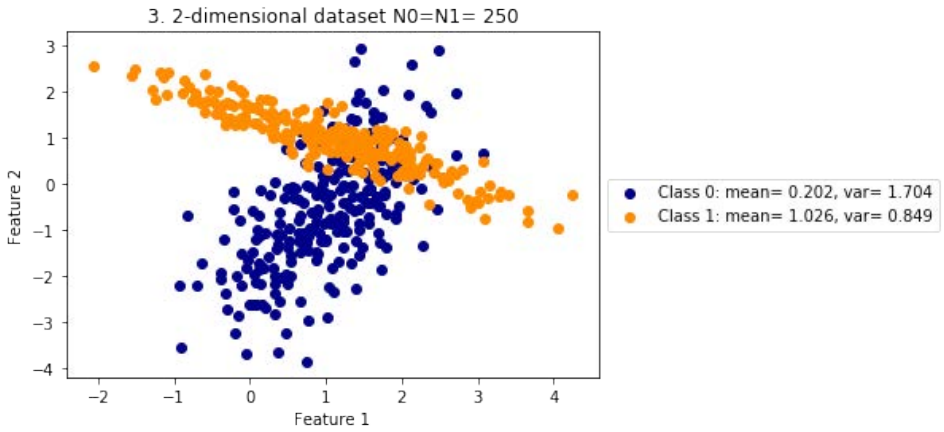
Make sure you are able to apply a Bayes classifier and that you have flexibility to specify the assumptions you wish to make regarding the covariance structure of the data, *i.e.*, unique covariance matrices for each class, a common covariance matrix for both classes, independence among features (a diagonal 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.

(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.

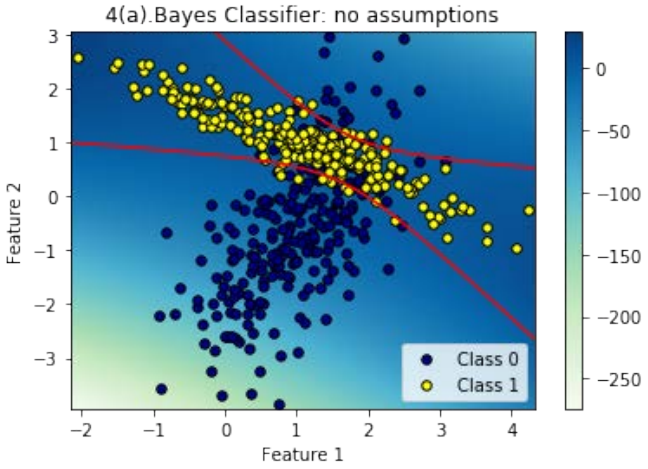


Answer:
Class 0:
N0 = 250
mean0, cov0 = [1, -0.5], np.array([[0.5, 0.5], [0.5, 1.5]])
Class 1:
N1 = 250
mean1,cov1 = [1,1], np.array([[1.5, -0.8], [-0.8, 0.5]])

[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.

(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?

Answer:

As there are no simplifying assumptions, the discriminant functions are listed as below: these are quadratic functions, since only the red term can be removed. When we make no assumptions, both class are using their own mean vector and their own covariance matrices, which make the most correct prediction of data distribution. The decision Boundaries are hyperquadrics: can be hyperplanes, hyperplane pairs, hyperspheres, hyperellipsoids, hyperparabaloids, hyperhyperparabaloids.

$$g_i(\mathbf{x}) = -\frac{1}{2}(\mathbf{x}-\mu_\mathbf{i})^t\mathbf{\Sigma_i}^{-1}(\mathbf{x}-\mu_\mathbf{i}) - \frac{d}{2}\ln 2\pi - \frac{1}{2}\ln|\mathbf{\Sigma_i}| + \ln P(\omega_i)$$

Discriminant Functions: (quadratic)

$$g_i(x) = x^t W_i x + w_i^t x + \omega_{i0}$$

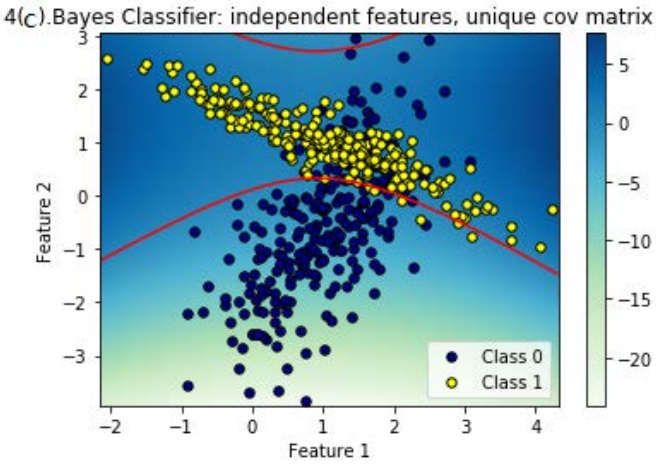
$$g_i(x) = x^t\left(-\frac{1}{2}\Sigma_i^{-1}\right)x + (\Sigma_i^{-1}\mu_i)^t x - \frac{1}{2}\mu_i^t\Sigma_i^{-1}\mu_i - \frac{1}{2}\ln|\Sigma_i| + \ln P(\omega_i)$$

Decision Statistic: $g_1(x) - g_0(x)$

$$= \left[-\frac{1}{2}(x-\mu_1)^T\Sigma^{-1}(x-\mu_1) - \frac{1}{2}\ln|\Sigma_1| + \ln P(\omega_1)\right] - \left[-\frac{1}{2}(x-\mu_0)^T\Sigma^{-1}(x-\mu_0) - \frac{1}{2}\ln|\Sigma_0| + \ln P(\omega_0)\right]$$

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

(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?

Answer:

When features are independent and covariance matrices under each class are unique, we have the following relationship of covariance matrices and discriminant functions which is linear in X. Decision boundary goes through x0 along line between means, orthogonal to this line. Thus we see the quadratic decision boundary is in a horizontal position.

$$\Sigma_i = \sigma^2 I$$

Discriminant Functions:

$$g_i(x) = \mathbf{w_i^t x} + \omega_{i0}$$

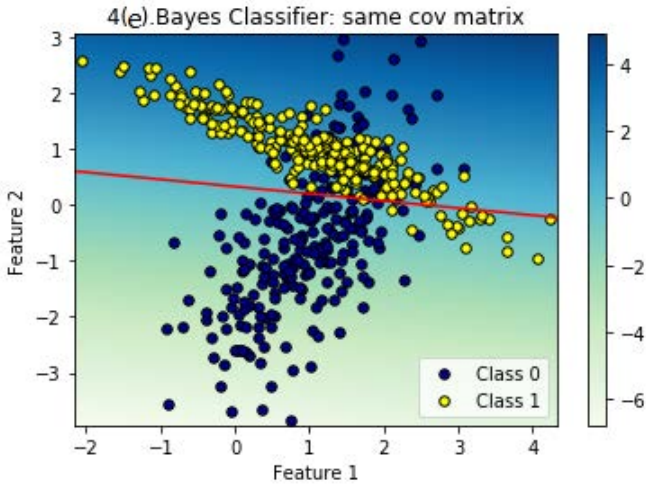
$$g_i(x) = \frac{1}{\sigma^2}\mu_i{}^t \mathbf{x} - \frac{1}{2\sigma^2}\mu_i^t\mu_i + \ln P(\omega_i)$$

Decision Boundary:

$$\mathbf{w}^t(\mathbf{x} - \mathbf{x_0}) = 0$$

$$(\mu_i - \mu_j)^t\left(x - \left(\frac{1}{2}(\mu_i + \mu_j) - \frac{\sigma^2}{(\mu_i - \mu_j)^t(\mu_i - \mu_j)} \ln\frac{P(\omega_i)}{P(\omega_j)}(\mu_i - \mu_j)\right)\right)$$

(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.



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

Answer:

When covariance matrices under each class are the same and features may be dependent, we have the following relationship of covariance matrices and discriminant functions which is linear in X. Decision boundary is a linear function, the hyperplane defined by boundary generally not orthogonal to the line between the two means.

Discriminant Functions:

$$g_i(x) = \mathbf{w_i^t}\mathbf{x} + \omega_{i0}$$

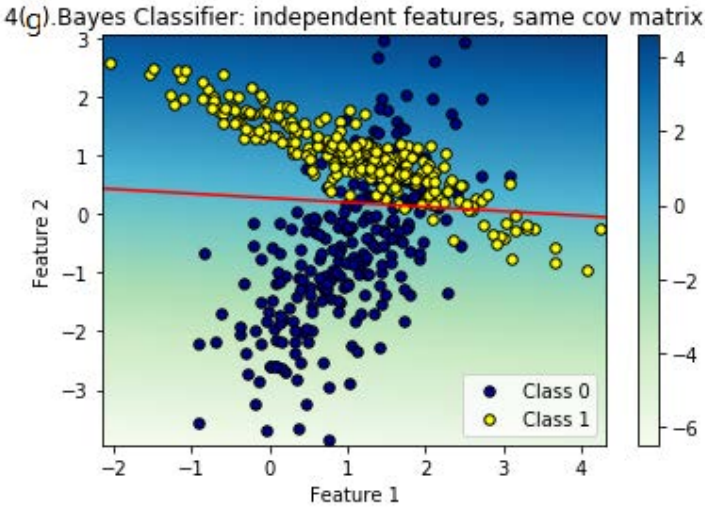
$$g_i(\mathbf{x}) = (\Sigma^{-1}\mu_i)^t\mathbf{x} \; -\frac{1}{2}\mu_i^t\Sigma^{-1}\mu_i + \ln P(\omega_i)$$

Decision Boundary:

$$\mathbf{w}^t(\mathbf{x} - \mathbf{x_0}) = 0$$

$$(\Sigma^{-1}(\mu_\mathrm{i}-\mu_\mathrm{j}))^\mathrm{t}(\mathbf{x}-\left(\frac{1}{2}(\mu_\mathrm{i}+\mu_\mathrm{j}) - \frac{\ln[\mathbf{P}(\omega_\mathrm{i})/\mathbf{P}\omega_\mathrm{j})}{(\mu_\mathrm{i}-\mu_\mathrm{j})\mathbf{\Sigma}^{-1}(\mu_\mathrm{i}-\mu_\mathrm{j})}(\mu_\mathrm{i}-\mu_\mathrm{j})\right)) = 0$$

(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?

Answer:

When covariance matrices under both classes are the same and the features are independent, we have the relationship of covariance matrices and discriminant functions which is similar to question 4(d), the difference is that the Decision boundary is determined by hyperplanes, not quadratic. In this case, the decision boundary passes through point x0 lying on the line between the two class means. If priors are identical, x0 is in the middle.

$$\mathbf{\Sigma_i} = \mathbf{\Sigma} = \sigma^2 I$$

(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

Answer:

Since one of the most important disadvantage of Bayes model is that it has strong feature independence assumptions, therefore we first must know whether the features of the dataset are relatively dependent. If they are closely related with each other, we should consider trying other models rather than Bayes classifier.

On the other hand, if the Bayes conditional independence assumption actually holds for our dataset, then a Bayes classifier will converge quicker than discriminative models like logistic regression, so we need less training data and Bayes model will perform well.

We could make four assumptions as in question 4 for our Bayes classifier.

Case 1: No assumptions, features may be dependent, covariance matrices are unique. We will get hyperquadric decision boundary.

Case 2: Features are independent, covariance matrices are unique. Hyperquadric decision boundary goes through x0 along line between means, orthogonal to this line.

Case 3: Same covariance matrices, features may be dependent. Decision boundary is a linear function, the hyperplane defined by boundary generally not orthogonal to the line between the two means.

Case 4: Same covariance matrices, features are independent. The decision boundary is determined by hyperplanes, not quadratic. In this case, the decision boundary passes through point x0 lying on the line between the two class means. If priors are identical, x0 is in the middle.

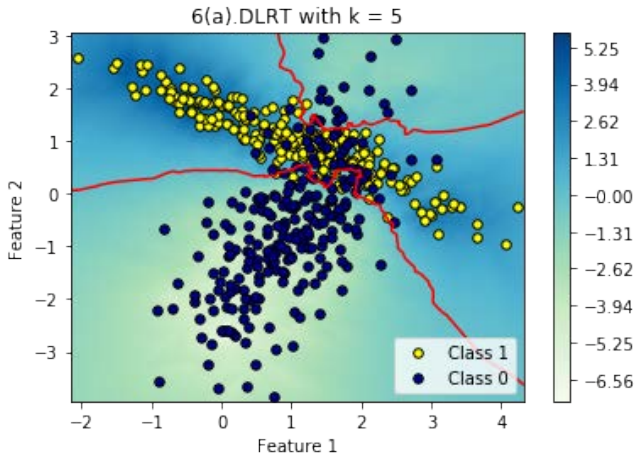
Exploring the DLRT

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?

Answer:

Advantages:

DLRT is simple and robust of the KNN classifier and have the freedom of not assuming some parametric form for the feature distributions.

DLRT decision boundary can take on any form. This is because DLRT is non-parametric, i.e. it makes no assumption about the data distribution. Contrast this to Bayes, which assumes that attributes are conditionally independent to each other given the class, and are normally distributed. Therefore Bayes can only have linear, elliptic, or parabolic decision boundaries, which makes the flexibility of DLRT decision boundary a huge advantage.

DLRT provides decision outputs that are reasonable accurate estimates of the log likelihood ratio test. An advantage of the DLRT method is the continuum of decision metrics that provide access to operating points with higher probability of detection (PD) and lower probability of false alarm
(PFA).

Disadvantages:

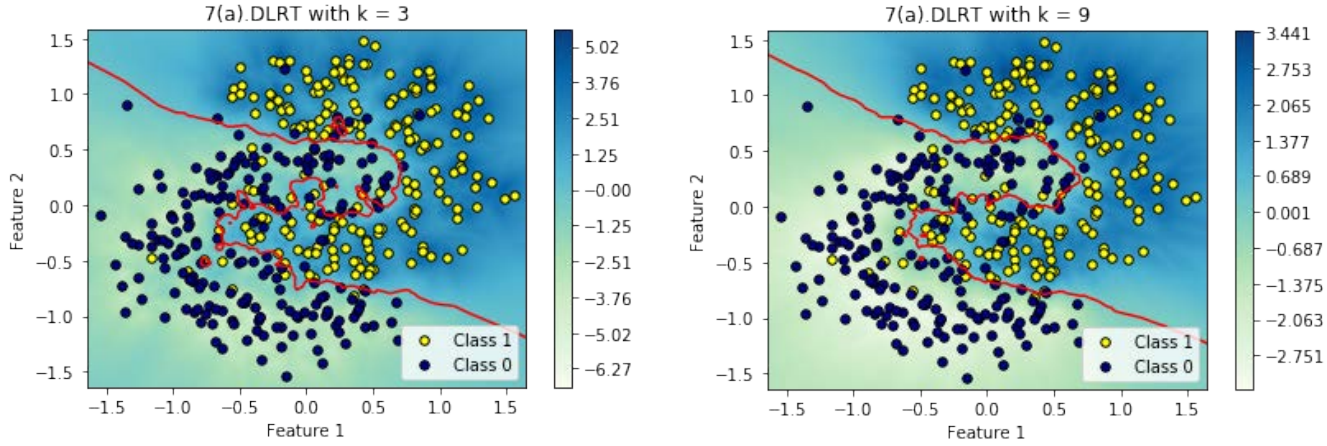
Its accuracy is driven by the density of the data and the appropriateness of the KNN probability density estimation.

When the training data are imbalanced, the decision statistics likely will have a tendency to be skewed toward the more prevalent class and consequently, the subsequent decisions likely will have a tendency to be skewed toward the more prevalent class.

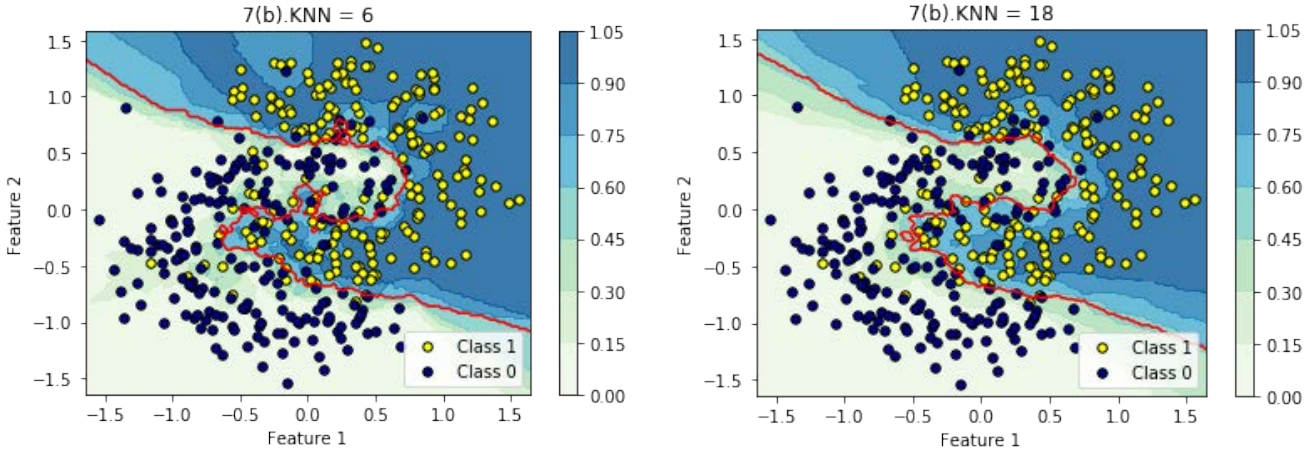
When the Bayes conditional independence assumption actually holds for our dataset, then a Bayes classifier will converge quicker and we need less training data. Under this condition, Bayes has more advantages than DLRT.

(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]

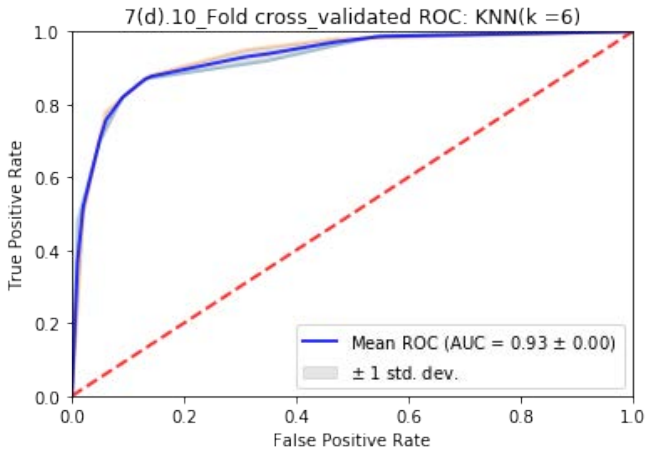


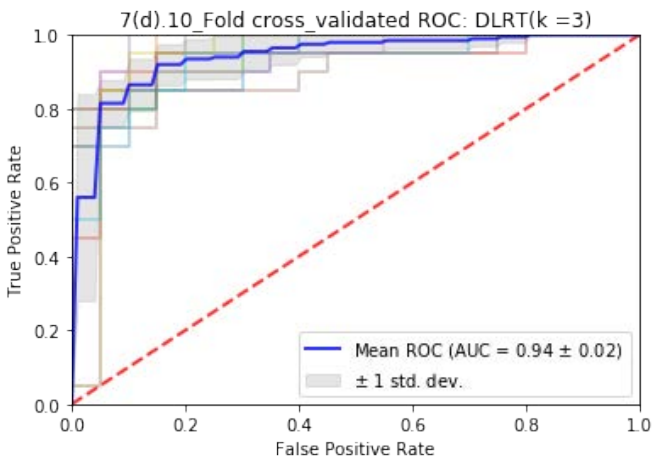
[5]The DLRT takes into account the distance to the k^{th} neighbor in both class 0 and in class 1, and so is making use of $2k$ neighbors when it is parameterized by k. For a fair comparison we want to use the same number of neighbors for both the DLRT and KNN, and so we use $2k$ neighbors for KNN.

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

Answer:
The decision statistic surfaces of DLRT with parameter k are comparable with that of KNN with the parameter of 2k. Since the threshold of DLRT can be chosen continuously, it is easier to achieve a better Pd/Pfa tradeoff points with DLRT. As the threshold is continuous in DLRT, the decision statistic surface are continuous colormap, decision boundary is determined by 0, while 2KNN is 2k classifier colors, majority decision boundary is 0.5.

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

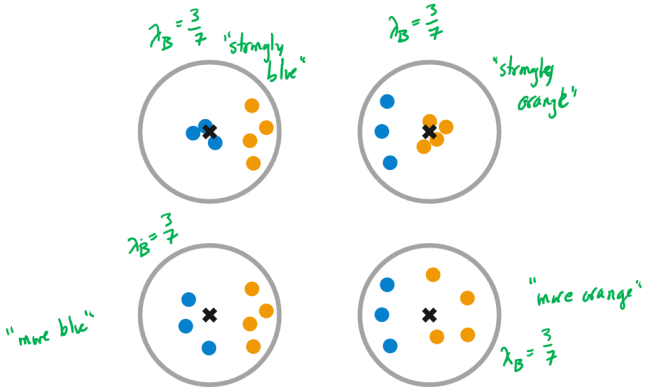




(e) What are your impressions of DLRT performance as compared to KNN performance? Under what conditions might you be inclined to prefer one classifier over the other?[6]

Answer:
Impressions:
The DLRT method results in a functional form similar to the 2KNN method.
The DLRT method provides decision outputs that are reasonably accurate estimates of the log likelihood ratio test; the accuracy is driven by the density of the data and the appropriateness of the KNN probability density estimation.
The DLRT method is the continuum of decision metrics that provide access to operating points with higher probability of detection (PD) and lower probability of false alarm (PFA).
The DLRT performs favorably in comparisons of the classification performance using the simulated data and provides an alternative non-parametric classification method for consideration when designing a distance-weighted KNN classification rule.

Conditions:
When we desired the simplicity and robustness of the k-nearest neighbor (KNN) classifier and the freedom of not assuming some parametric form for the feature distributions we could use DLRT. As the graph shown below, KNN won't consider the relative inclination of classification within the distance unit, while DLRT take the density of data into account. So if the data classes distribution is imbalanced and clustered, we could use DLRT. When data is equally balanced or distributed, and there is no preference importance of features, KNN would work well.



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