Pattern Classification and Recognition
ECE 681

Spring 2016
Homework #1: ROC Curves and Nearest Neighbors Classification
Solution Guide

This solution guide may provide analysis and explanations in greater depth than that requested in the homework assignment. As such, this solution guide *does not* necessarily represent the expectations for homework submissions.

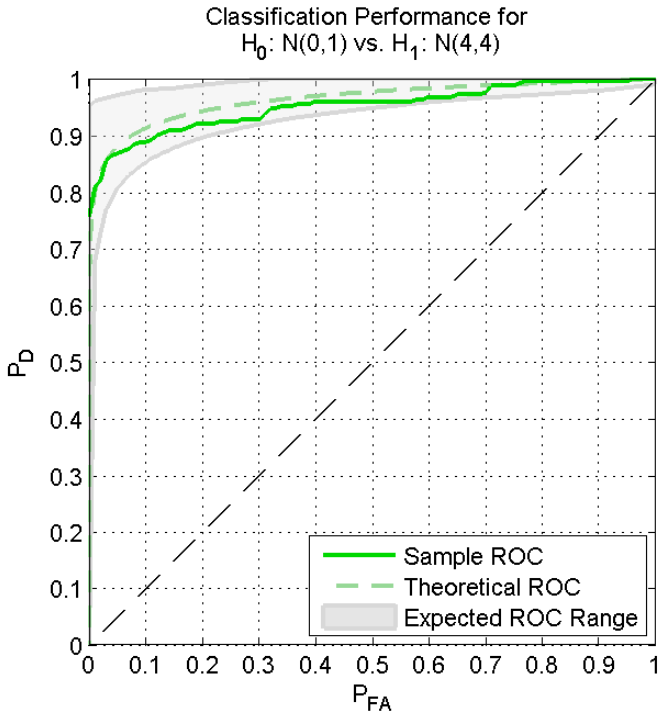
ROC Curves

Implement your own ROC curve generation function.

(5) 1. Generate an ROC curve for the following distributions of decision statistics:

H_0 decision statistics are 250 samples drawn from a normal (Gaussian) distribution with mean $\mu = 0$ and variance $\sigma^2 = 1$.
H_1 decision statistics are 250 samples drawn from a normal (Gaussian) distribution with mean $\mu = 4$ and variance $\sigma^2 = 4$.

One realization of a sample ROC for H_0:$\mathcal{N}(0,1)$ vs. H_1:$\mathcal{N}(4,4)$ is shown below (solid line), as well as the theoretical ROC derived directly from the given distributions (dashed line). The range within which we would expect a sample ROC to fall is shown by the grey region. A sample ROC, which is determined numerically from random samples drawn from the H_0 and H_1 distributions, is not necessarily expected to align with the theoretical ROC, though with high probability it is expected to fall within the grey region. The sample ROC shown here meets those expectations.

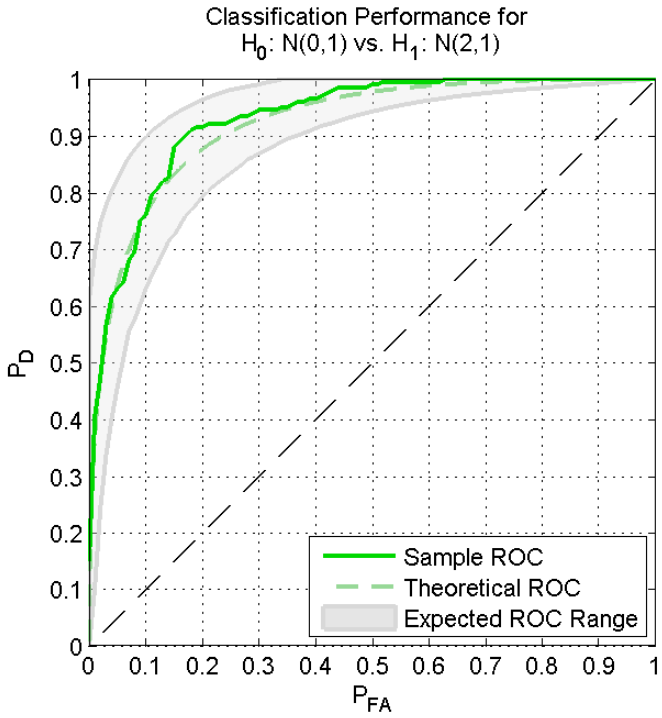


(5) 2. Generate an ROC curve for the following distributions of decision statistics:

H_0 decision statistics are 300 samples drawn from a normal (Gaussian) distribution with mean $\mu = 0$ and variance $\sigma^2 = 1$.

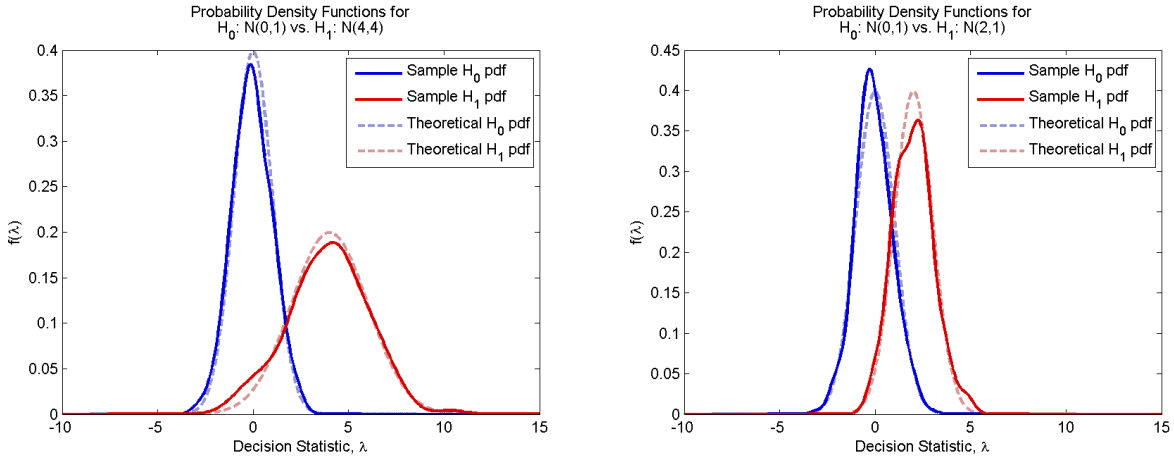
H_1 decision statistics are 200 samples drawn from a normal (Gaussian) distribution with mean $\mu = 2$ and variance $\sigma^2 = 1$.

One realization of a sample ROC for H_0:$\mathcal{N}(0,1)$ vs. H_1:$\mathcal{N}(2,1)$ is shown below (solid line), as well as the theoretical ROC derived directly from the given distributions (dashed line). The range within which we would expect a sample ROC to fall is shown by the grey region. A sample ROC, which is determined numerically from random samples drawn from the H_0 and H_1 distributions, is not necessarily expected to align with the theoretical ROC, though it is expected to fall within the grey region. The sample ROC shown here meets those expectations.



(10) 3. Comment on whether or not the ROCs you obtained in Questions 1 and 2 make sense, considering the distributions of the decision statistics in both cases.
HINT: Visualize the distributions of the decision statistics. The Matlab function `ksdensity` is a nice function for obtaining estimates of probability density functions given samples drawn from the pdfs.

The sample ROCs obtained in Questions 1 and 2 make sense, and are explained by the sample pdfs shown below. The sample pdfs were generated by applying the `ksdensity` function to the samples drawn from the H_0 and H_1 pdfs and the resulting pdf estimates are shown below (solid lines, H_0 in blue, H_1 in red), with the pdfs for Question 1 on the left and the pdfs for Question 2 on the right. For reference, the theoretical pdfs are also shown (dashed lines, H_0 in blue, H_1 in red). The sample pdfs are close approximations to the theoretical pdfs.



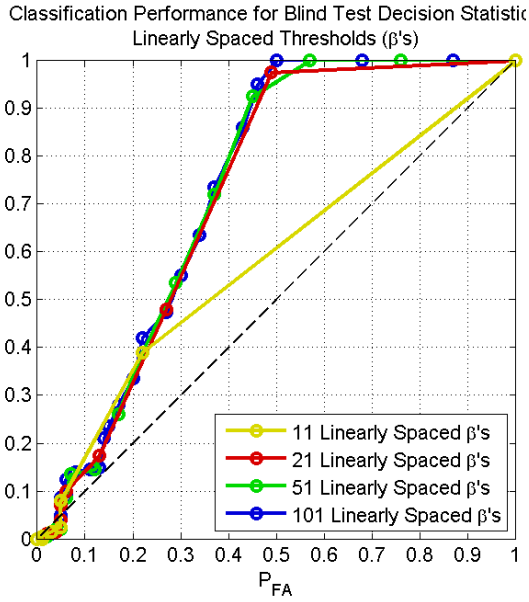
The H_0 and H_1 pdfs for Question 1 have less overlap than the H_0 and H_1 pdfs for Question 2, so we expect the ROC for Question 1 to be higher (closer to the top left corner) than the ROC for Question 2. We do, indeed, observe that the sample ROC for Question 1 is higher than the sample ROC for question 2. (There is some overlap between the range in which we expect a sample ROC for Question 1 to fall and the range in which we expect a sample ROC for Question 2 to fall, so it is possible that your sample ROCs may be about the same, or even have the sample ROC for Question 1 slightly below that for Question 2.)

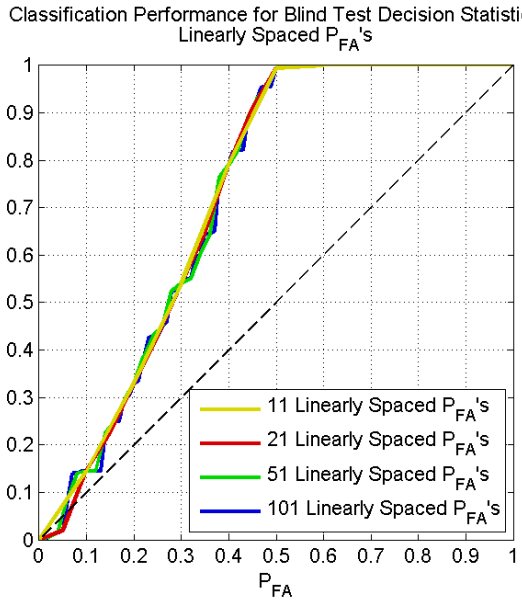
In addition, the pdfs for Question 1 do not have the same variance; the variance for H_1 is larger than the variance for H_0. Thus, we expect that the ROC curve for Question 1 will not be symmetric about the diagonal line from top-left to bottom-right $[(P_{FA}, P_D) = (0,1)$ to $(P_{FA}, P_D) = (1,0)]$, or $P_D = 1 - P_{FA}$. We observe that the ROC for Question 1 is not symmetric about $P_D = 1 - P_{FA}$, and is skewed to the left since the H_1 pdf has larger variance. (If the H_0 pdf had larger variance, we would see the ROC skewed toward to the top.) In contrast, the pdfs for Question 2 have the same variance, so we expect the ROC curves for Question 2 to be symmetric about the diagonal line from top-left to bottom-right, $P_D = 1 - P_{FA}$, which we do observe.

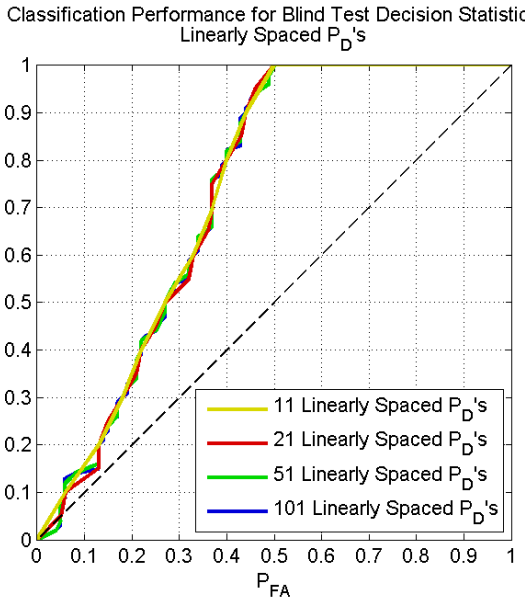
(10) 4. Generate an ROC curve for the decision statistics and targets provided in the Matlab `.mat` file: `HW01rocBlindTestDecisionStatistics.mat`

ROCs for the blind test decision statistics are shown below for three different methods of calculating the ROC curve: linearly spaced thresholds (β's) (top), linearly spaced P_{FA}'s (bottom left), and linearly spaced P_D's (bottom right). In each case, the number of thresholds utilized varies from 11 (a sparsely sampled ROC) to 101 (a densely sampled ROC).

Regardless of the method chosen to generate the ROC, it is expected to look similar to one of these ROCS (except the ROC generated with 11 linearly spaced thresholds).





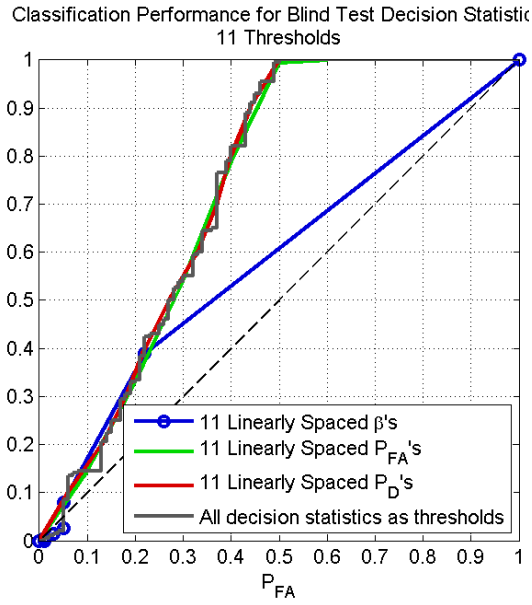


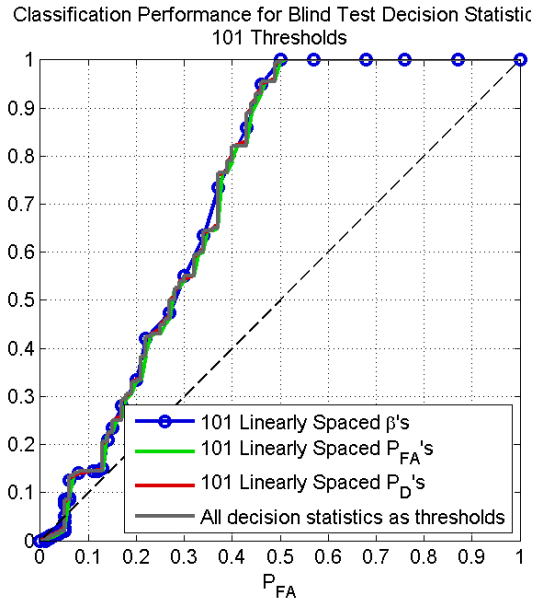
The ROCs calculated using linearly spaced thresholds demonstrate a potential pitfall of this approach – the ROC may not be fully sampled, as illustrated by the ROC obtained with 11 linearly spaced thresholds

in which there is only one sample of the ROC from (P_{FA}, P_D) of about (0.05,0.1) to (P_{FA}, P_D) of (1,1). With as few as 21 samples, however, the ROC is closer to being adequately sampled, though the majority of the samples are concentrated in the lower left corner, resulting in grossly oversampling the ROC in this region and an undersampling the ROC from (P_{FA}, P_D) of about (0.15,0.15) to (P_{FA}, P_D) of (1,1).

The ROCs calculated using linearly spaced P_{FA}'s and linearly spaced P_D's, on the other hand, fully sample the ROC even with as few as 11 samples, demonstrating that both of these approaches are more robust methods for generating ROCs.

Direct comparisons of these three approaches to generating an ROC are shown below for ROCs generated using 11 thresholds (left) and 101 thresholds (right). The ROC obtained by using every decision statistic as a threshold is also shown, as this ROC illustrates the finest possible resolution that can be obtained. The ROCs generated using 11 thresholds again illustrate a potential pitfall of generating ROCs using linearly spaced thresholds – the ROC determined from linearly spaced thresholds is very inaccurate for $P'_{FA}s$ of about 0.4 to 1, while the ROCs determined from linearly spaced P_{FA}'s and linearly spaced P_Ds closely approximate the full ROC obtained using every decision statistic as a threshold. The ROCs generated using 101 thresholds illustrate that increasing the number of samples in the ROC can overcome the limitation of using linearly spaced thresholds, though the ROC may still be undersampled locally. This example illustrates that using 101 linearly spaced thresholds results in a reasonable approximation to the full ROC, though 101 linearly spaced P_{FA}'s and 101 linearly spaced P_Ds both provide ROCs that are closer approximations to the full ROC obtained using every decision statistic as a threshold.





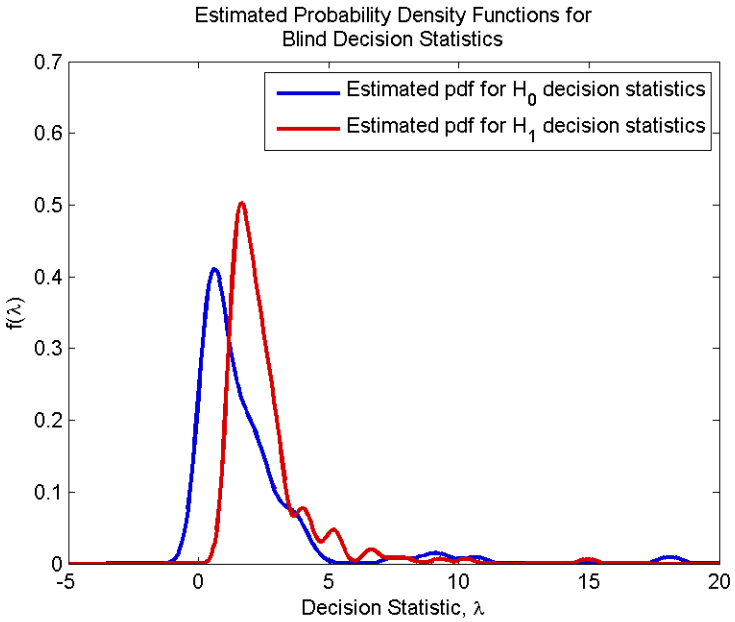
(10) 5. Comment on whether or not the ROC for the blind decision statistics obtained in Question 4 makes sense.

HINT: You can "sanity check" this ROC curve (for which you do not know the underlying distributions for the decision statistics) by visualizing the decision statistics. In addition to (or instead of) looking at estimates of the pdfs, you can also look at a plot of the sorted decision statistics coded by class. Assume `ds` contains the sorted decision statistics, and `t` contains the associated target (truth or class) for each decision statistic and is either `0` or `1`. The following lines of code (which make use of *logical indexing*) will produce this plot:

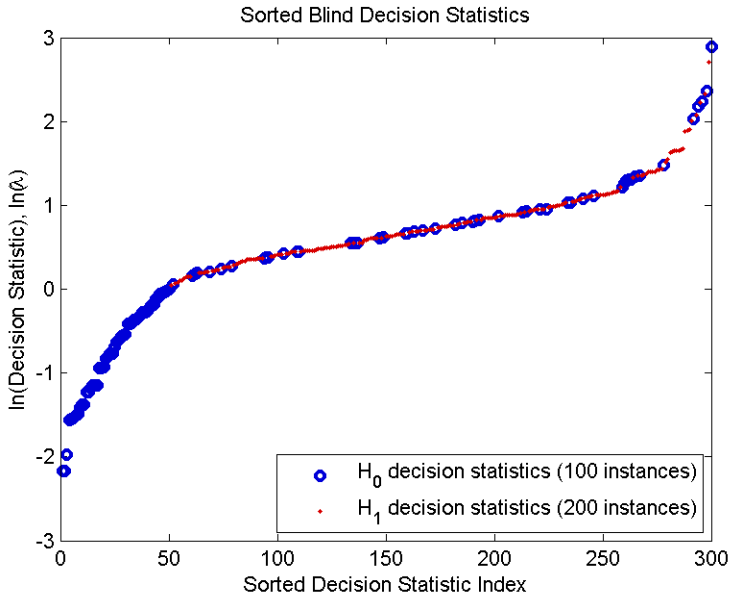
```
figure
plot(find(t==0),ds(t==0),'b.','MarkerSize',8)
hold on
plot(find(t==1),ds(t==1),'ro','MarkerSize',5)
```

The ROCs for the blind decision statistics are somewhat atypical, in that they are approximately linear from P_{FA} of 0 to P_{FA} of about 0.5, and then P_D=1 for P_{FA} greater than 0.5. This suggests the distributions of the decisions statistics are likely not Gaussian.

Estimates of the pdfs for the decision statistics under H_0 and H_1 are shown below. These pdf estimates illustrate that the distributions of the decision statistics are not Gaussian, and are skewed with very heavy tails to the right. These skewed non-Gaussian distributions, with heavy tails, explain the atypical shape of the ROC. For thresholds greater than about 0, the number of false alarms and detections that are lost as the threshold increases are proportional, which explains the approximately linear portion of the ROC form P_{FA} of 0 to P_{FA} of about 0.5. For thresholds less than about 0, primarily false alarms are lost as the threshold increases, as there are few, if any, H_1 decision statistics less than approximately 0, which explains why P_D=1 for P_{FA} greater than 0.5.



The behavior of the ROC can be more fully understood by examining the ordering of the decision statistics themselves. A plot of the sorted decision statistics, color-coded by class, is shown below. The natural logarithm of the decision statistics is plotted to improve the visualization. Since the natural logarithm is a monotonic function, plotting the natural logarithm of the decision statistics does not alter their relative ordering. (Any monotonic transformation would affect neither the ordering of the decision statistics nor the ROC if the ROC were generated from a monotonic transformation of the original decision statistics). This plot demonstrates that the smallest 50 decision statistics are from H_0, thus any (transformed) thresholds less than 0 will result in a P_D of 1, and as the (transformed) threshold is increased from its minimum value to 0, P_{FA} will decrease from 1 to approximately 0.5 (approximately 50 of the 100 H_0 decision statistics are less than 0). Once the (transformed) threshold is greater than 0, H_0 and H_1 decision statistics appear with approximately equal frequency, leading to the linear behavior in the ROC from P_{FA} of approximately 0.5 to P_{FA} of 0.



(30) 6. Submit the Matlab code that produced the above results for **ROC Curves**, including your implementation of a ROC curve generation function, as an Attachment to the Assignment in Sakai. (We should be able to run this code to replicate your results.)

Intentionally omitted.

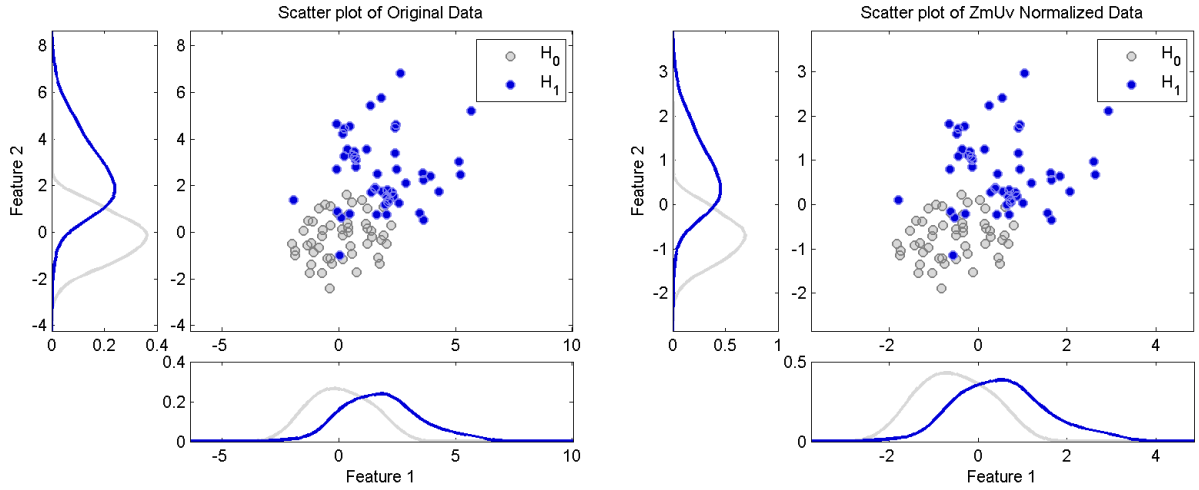
KNN Classification

Implement your own KNN classifier, with the L_2-norm as the distance metric.

Train your KNN classifier, with the L_2-norm as the distance metric, using the following training data:

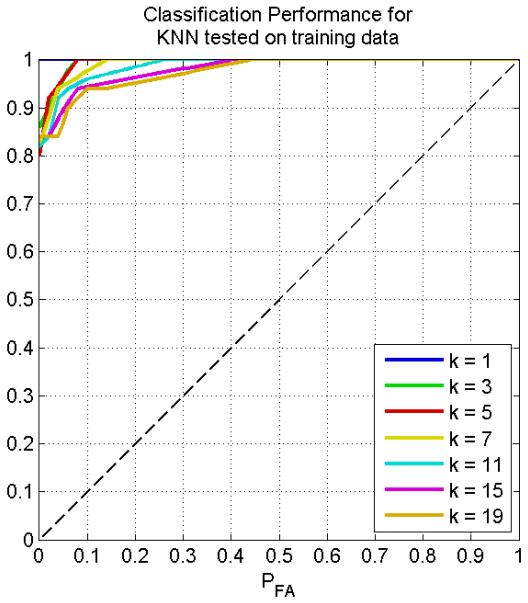
H_0 features are 50 samples drawn from a 2-dimensional normal (Gaussian) distribution with mean $\mu = [0\ 0]'$ and covariance matrix $C = \left[\begin{smallmatrix} 1 & 0 \\ 0 & 1 \end{smallmatrix}\right]$ (the identity matrix).
H_1 features are 50 samples drawn from a 2-dimensional normal (Gaussian) distribution with mean $\mu = [2\ 3]'$ and covariance matrix $C = \left[\begin{smallmatrix} 2 & 0 \\ 0 & 2 \end{smallmatrix}\right]$.

Sample scatter plots of the original training data and the training data normalized to be zero-mean and unit-variance are shown below (original data on left, normalized data on right). In both cases, the axes are scaled so that 1 unit in the x-direction is equal to 1 unit in the y-direction. These scatter plots illustrate the minimal impact of normalization for this data. Quantitatively, the J_0 scatter metric $\left(J_0 = \frac{\text{trace}(S_b)}{\text{trace}(S_w)}\right)$ between classes for the original data is 0.925 while the J_0 scatter metric between classes for the normalized data is 1.0, which further confirms the minimal impact of normalization for this data. Since the effect of zero-mean unit-variance normalization on this data is negligible (as will be shown by comparing the AUC for both cases), ROC curves will be presented only for unnormalized data. Performance for zero-mean, unit-variance normalized data is expected to be similar, and finding performance for either the unnormalized or normalized data is acceptable.

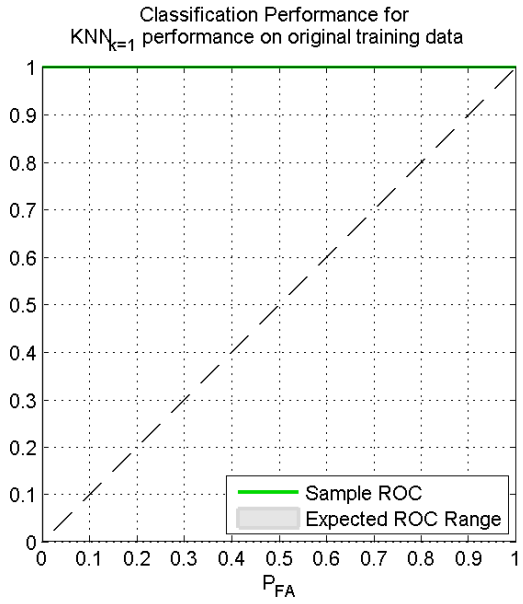


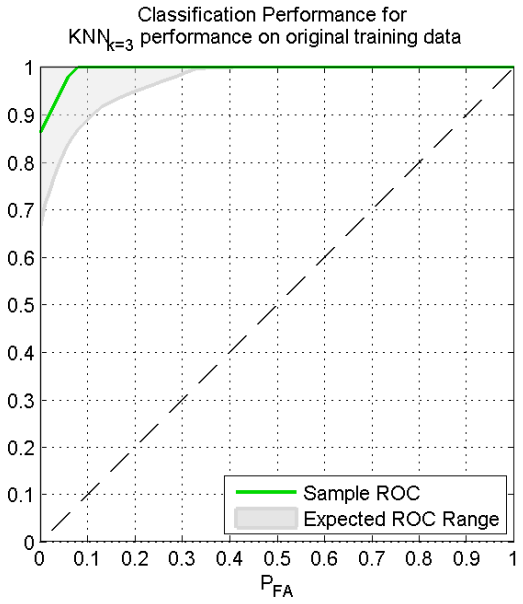
(10) 1. Generate ROC curves for the KNN classifier when applied to the above training data for each of the following choices of k: 1, 3, 5, 7, 11, 15, and 19. (You may plot all the ROCs on a single set of axes, provided that you use different line types and include a legend.)

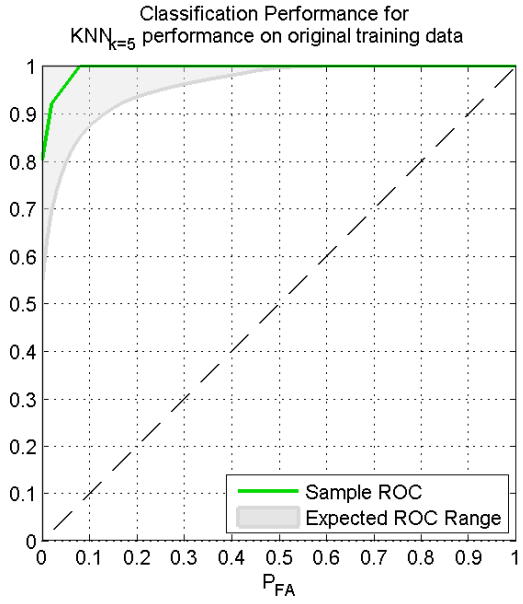
Sample ROCs for each value of k when the KNN classifier is tested on the training data are shown below. When $k = 1$, performance is perfect ($P_D = 1$ for all values of P_{FA}). Performance is very similar, and less than perfect, for the other values of k.

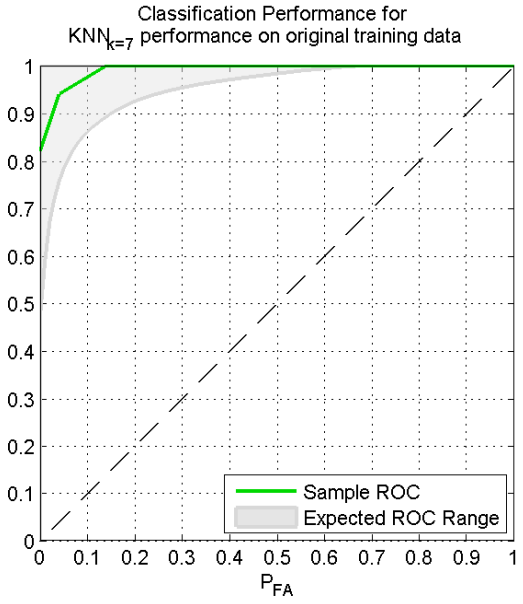


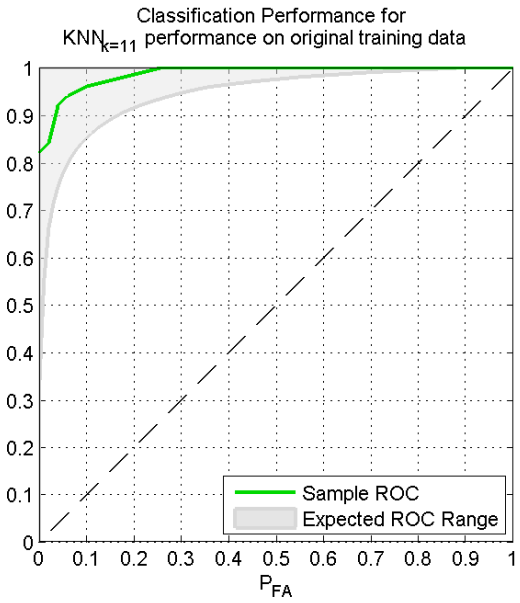
With the exception of $k = 1$, the ROC is highly dependent on the random sample of data obtained for training the KNN classifier. The range within which the ROC is expected to fall when the KNN classifier is tested on the training data is shown below for each value of k. Your sample ROCs may be quite different from the sample ROCs shown here, but they are expected to fall within the grey region with high probability.

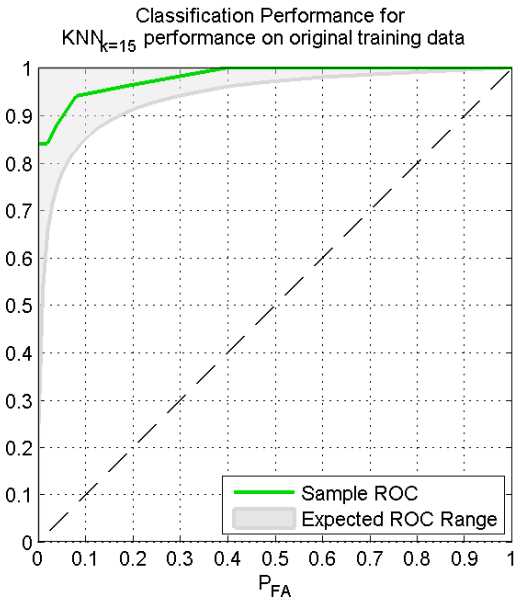


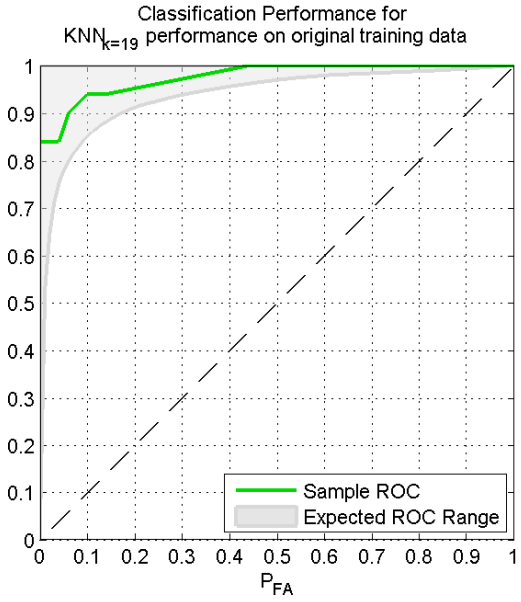






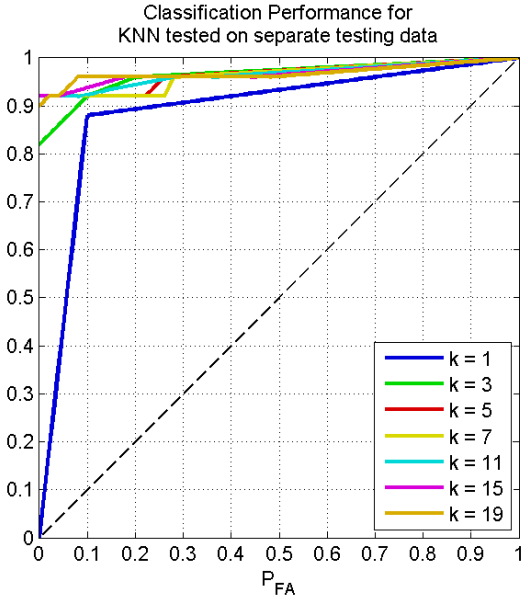

Classification Performance for
KNN$_{k=11}$ performance on original training data
Sample ROC
Expected ROC Range
P_{FA}


Classification Performance for
KNN$_{k=15}$ performance on original training data
Sample ROC
Expected ROC Range
P_{FA}

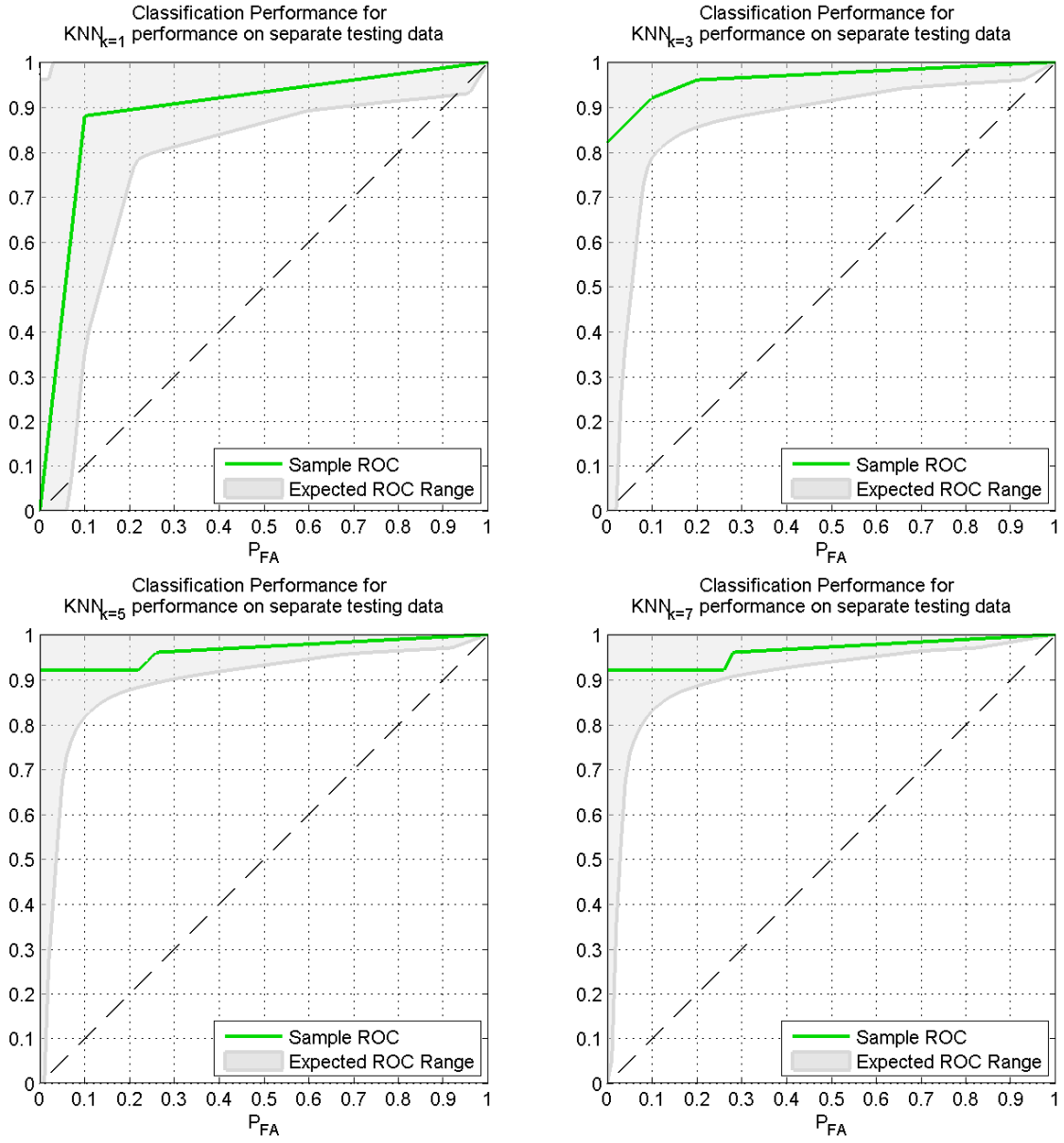

Classification Performance for
KNN$_{k=19}$ performance on original training data
Sample ROC
Expected ROC Range
P_{FA}

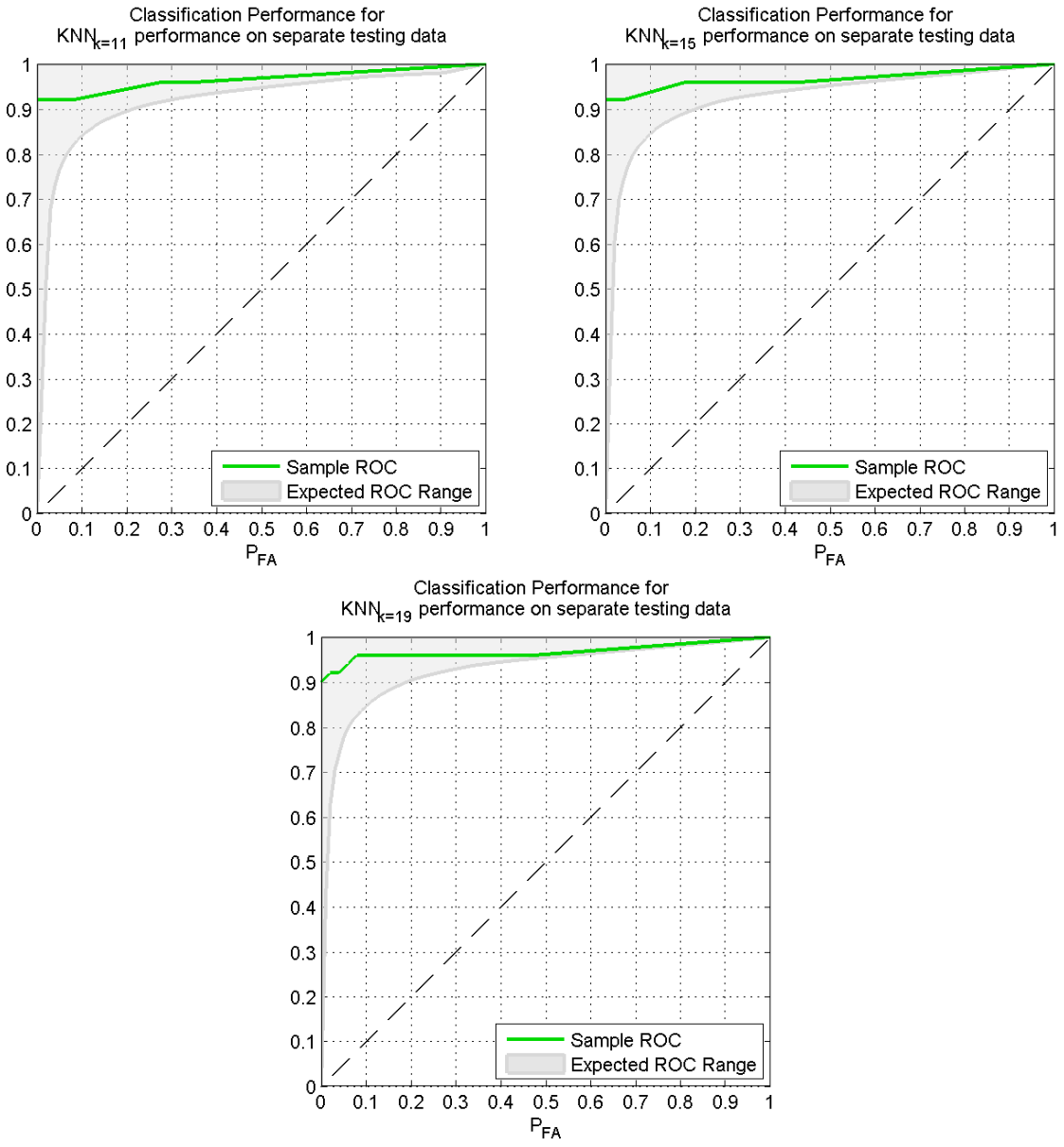
(10) 2. Generate a separate set of testing data that follows the same distributions as the training data. Generate ROC curves for the KNN classifier trained using the original training data then applied to the new (previously unseen) testing data for each of the following choices of k: 1, 3, 5, 7, 11, 15, and 19. (You may plot all the ROCs on a single set of axes, provided that you use different line types and include a legend.)

Sample ROCs for each value of k when the KNN classifier is tested on separate testing data are shown below. When $k = 1$, performance is far from perfect and while performance is generally stronger for the other values of k, performance is generally lower than when the KNN is tested on the training data.



As for the KNN classifier tested on the training data, the ROC is highly dependent on both the random sample of data obtained for training the KNN classifier as well as the random sample for testing the KNN classifier. The range within which the ROC is expected to fall when the KNN classifier is tested on separate testing data is shown below for each value of k. Your sample ROCs may be quite different from the sample ROCs shown here, but they are expected to fall within the grey region with high probability.



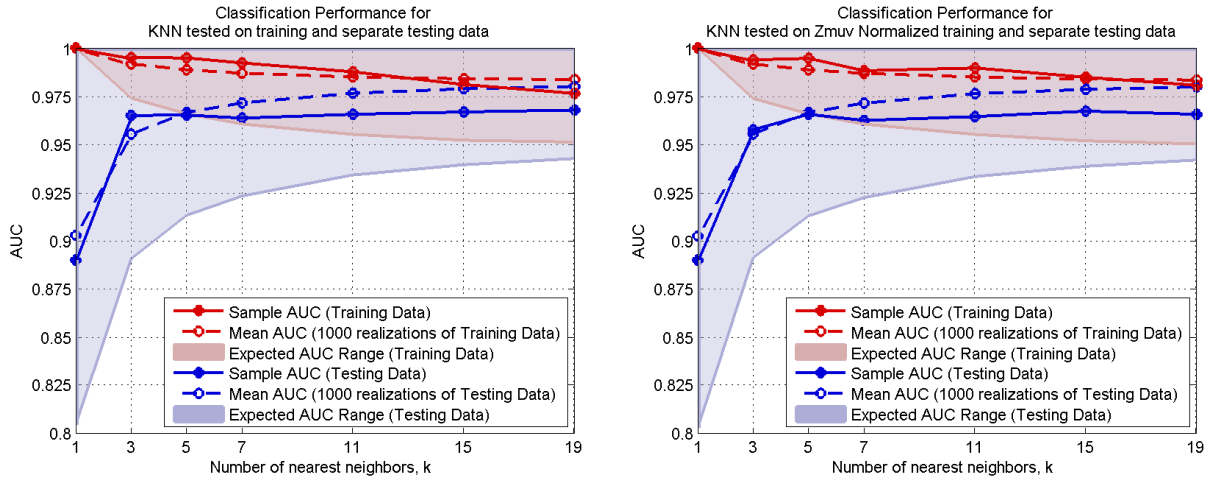


(20) 3. Comment on whether or not the ROCs you obtained in Questions 1 and 2 make sense, considering both the values of k and whether the KNN is applied to the original training data, as in Question 1, or separate testing data, as in Question 2.

The ROCs obtained in Questions 1 and 2 do make sense. When the KNN classifier with $k = 1$ is tested on the training data, performance is perfect, as expected. When it is tested on separate testing data, however, performance is much lower. In addition, performance on the original training data is generally stronger than performance on the separate testing data. This, however, may be an artifact of the sample ROCs shown here, as there is significant overlap between the range in which the ROCs are expected to fall when the KNN is tested on the training data and the range in which the ROCs are expected to fall when the KNN is tested on separate testing data.

(20) 4. Evaluate performance as a function of the number of nearest neighbors, k, by plotting the area under the ROC curve (AUC) as a function of k for *both* the KNN classifier applied to the training data [ROCs generated in Question 1] *and* the KNN classifier applied to the testing data [ROCs generated in Question 2]. (You may plot AUC vs. k for both cases on a single set of axes, provided that you use different line types and include a legend.)

AUC is shown below as a function of k for the KNN classifier tested on the original training data (red solid line) and the KNN classifier tested on separate testing data (blue solid line). Results for the unnormalized data are shown on the left, and results for the zero-mean unit-variance normalized data are shown on the right. Performance trends, and performance levels, are highly similar in both cases, as is expected given the minimal effect of normalization on the separability of the classes as measured by the J_0 scatter metric. Performance on the training data generally decreases with increasing k, while performance on the separate testing data generally increases with increasing k. For the sample ROCs shown previously, the KNN has stronger performance on the training data than the testing data.



The expected ranges for these performance curves are also shown (testing on the original training data in the pink region, and testing on separate testing data in the blue region), along with mean performance (testing on original training data with red dashed line, and testing on separate testing data with blue dashed line). As the number of nearest neighbors, k, increases, the mean AUC for the KNN tested on the training data decreases while the mean AUC for the KNN tested on separate testing data increases. As k increases, performance when tested on training data and when tested on separate testing data converge, and by $k = 19$ performance is nearly equal.

These ranges in which these performance cures are expected to fall overlap significantly, demonstrating that performance on separate testing data may be stronger than performance on the original training data, depending on the specific realizations of random training and testing data.

(20) 5. Comment on whether or not the performance trends you obtained in Question 4 make sense, considering both the values of k and whether the KNN is applied to the original training data, as in Question 1, or separate testing data, as in Question 2.

The performance trends do make sense. As k increases, performance on the training data decreases. There is a tendency for KNN classifiers to be over-trained (highly tuned to the training data), especially at low values of k, so performance on the training data decreasing as k increases is consistent with the tendency toward over-training decreasing as k increases. As k increases, performance on the separate testing data increases. This is also consistent with the tendency toward over-training decreasing as k increases. With the tendency toward over-training decreased as k increases, the KNN classifier generalizes better, meaning it is more suitable to data sets other than the original training data.

(30) 6. Submit the Matlab code that produced the above results for **KNN Classification**, including your implementation of the KNN classifier, as an Attachment to the Assignment in Sakai. (We should be able to run this code to replicate your results.)

Intentionally omitted.

Blind Test

Apply a KNN classifier developed (trained) as in **KNN Classification** to generate decision statistics for blind test data.

(20) 1. Select a value for the parameter k based on the performance evaluation completed for Question 4 in **KNN Classification**. (Feel free to consider additional values for k beyond those specified in Question 4 in **KNN Classification**, if you think it would be helpful or beneficial.)

Comment on the trade-offs you considered when choosing k, and justify your choice of k.

Looking at the sample AUC vs. k curves, I would choose $k = 5$ for the KNN classifier. At $k = 5$, performance on the separate testing data has plateaued. Choosing $k = 5$ rather than $k = 3$ (for which performance on the training data is nearly identical) allows for some margin for error in the choice of k, as there is a sharp increase in performance from $k = 1$ to $k = 3$, which suggests that the performance predicted at $k = 3$ may be less reliable than then performance predicted for $k = 5$. Predicted performance for choices of $k > 5$ may be even more stable, though choosing a higher value of k increases the computational complexity of the KNN. Thus, $k = 5$ is expected to have better predicted performance than $k = 3$ with lower computational complexity than $k \geq 7$.

Looking at the mean AUC vs. k curves, I would choose $k = 11$ for the KNN classifier, for the same reasoning given above; $k = 11$ is a choice where predicted performance has plateaued and predicted performance is expected to be more stable than choices of $k < 11$. Thus, $k = 11$ is expected to have better predicted performance than $k \leq 7$ with lower computational complexity than $k \geq 15$.

(20) 2. Generate decision statistics for the features provided in the Matlab `.mat` file:
`HW01knnBlindTestFeatures.mat`

Save the decision statistics to a Matlab `.mat` file, with the decision statistics stored in the vector `decStat` and saved in the same order as the blind test features. Submit the Matlab `.mat` file containing your decision statistics as an Attachment to the Assignment in Sakai.

(We know the corresponding targets for the blind test data, and will score your decision statistics to generate an ROC curve to evaluate your decision statistics.)

Intentionally omitted.