Pattern Classification and Recognition
ECE 681

Spring 2019
Homework #2: K-Nearest Neighbors and Bias-Variance Tradeoff

Due: 5:00 PM, Thursday, February 7, 2019
Grace Period Concludes: 11:30 PM, Tuesday, February 12, 2019

This homework assignment is worth **220 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.

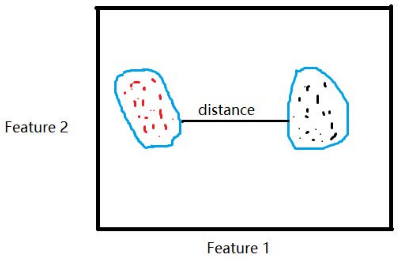
Exploring K-Nearest Neighbors

It is helpful to understand not only *how* a classifier works, but when it might behave unexpectedly[1] and *why* it might behave unexpectedly.

(40) 1. Suppose you are using a K-Nearest Neighbors classifier with the L_2 distance metric and majority vote decision rule, and have a total of N training points for two classes H_0 and H_1. The number of H_0 training points is given by N_0 and the number of H_1 training points is given by N_1, so the total number of training points is $N = N_0 + N_1$.

(a) Assume the classes in the training data are balanced, so $N_0 = N_1$. Sketch an example of 2-dimensional data for which performance is perfect for any value of $k \leq \frac{N}{2}$ when the classifier is tested on the training data. (It may be easier to represent the data as point clouds rather than discrete points.)

Solution:
The distance of H0 and H1 should be bigger than the internal size of H0 and H1 clusters, so we won't pick a different class to verify λ.



(b) Suppose the training data for each class is distributed as you've drawn above, but the classes are not balanced, so $N_0 \neq N_1$ (but $N = N_0 + N_1$ is unchanged). Explain how to determine the maximum value of k for which performance will be perfect when the classifier is tested on the training data.

Kmax = min(N0,N1)
We should determine Kmax as the min of N0 and N1. So we won't pick a different class when we determine λ. If we pick a K which is bigger than the minsize, then we will pick the other class point for the dataset point which has minmal size.

[1] Some might refer to this as the classifier failing, but often the classifier is doing exactly what it is supposed to do on data that is inconsistent with the classifier's underlying assumptions.

(c) What are potential implications for KNN classification decisions when the classes in the training data are imbalanced, $N_0 \neq N_1$? (In other words, how might KNN classification decisions influenced by the relative proportions of H_0 and H_1 training data?)

Solution:
It depends on the distribution of the training data.
- If the training data is distributed as (a) question, distance between H0 and H1 are bigger than their size, then as long as KNN classification decision k ≤ min(N0, N1), there won't be implications on our training model.
- If the training data is distributed disorderedly, and N0<<N1, then when we do KNN classification decisions, as k increases, we will pick the other class to verify our lamda λ value. Under this mechanism, the class with great proportion will take advantage. That is the KNN classification decision will take more weight on class which has larger proportion.
For example, if we have N0=1, N1=99, when k=5, if data is distributed randomly, then it might be possible that we will pick 4 H1 point for each H0 point.

(d) Suppose you believe the environment in which your classifier will be deployed has balanced classes $\left(P(H_0) = P(H_1) = \frac{1}{2}\right)$, but the classes in your training data are imbalanced. How could you modify KNN to mitigate potential issues associated with imbalanced classes in the training data?[2]

Solution:
We could place weight ration when we calculate λ for our dataset.
That is:
λ (x) = (Σ I for k nearest points) * (N0/N1)/k
So for the relative small portion class, we will multiply the corresponding ration to calculate its λ . Then we could adjust our dataset to be applied with balanced classes trained model.

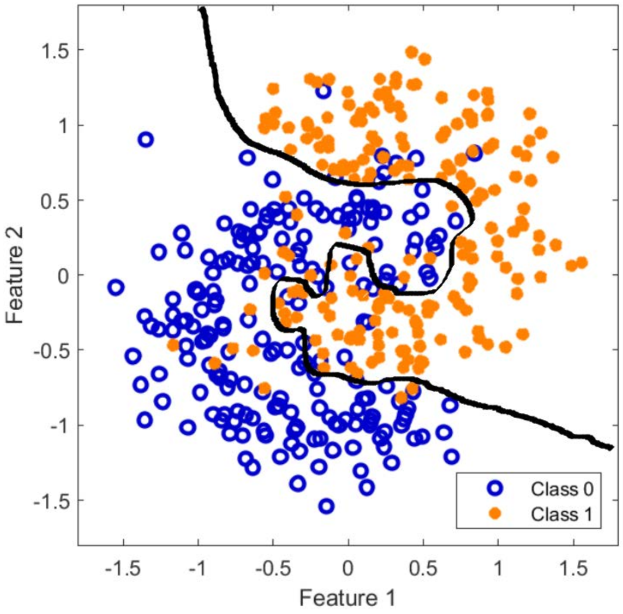
[2] This question is not asking for a formal derivation of an optimal strategy, but a qualitative description of what you would think about and how you might adapt KNN.

Applying K-Nearest Neighbors to Explore Bias-Variance Tradeoff

Make sure you are able to apply K-Nearest Neighbors and that you have flexibility to specify how you want to measure the distance between points.

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.

The following questions concern a data set that is provided as a `csv` file, `dataSetHorseshoes.csv`. 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. When you visualize the data set, you should see this:



(10) 2. Suppose you are using a K-Nearest Neighbors classifier with the L_2 distance metric.

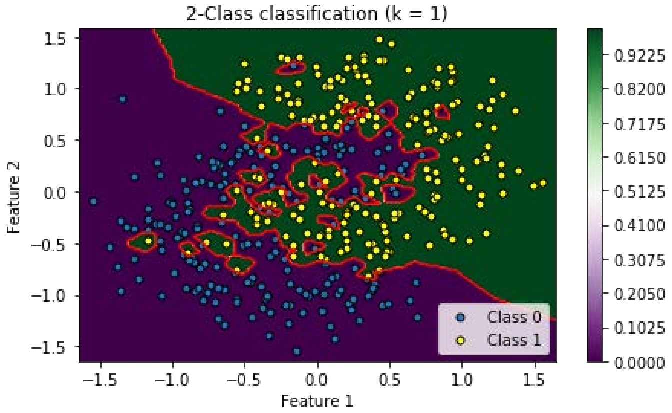
(a) On the visualization of the `Horseshoe` data above, sketch what you would consider to be an ideal boundary between the two classes.

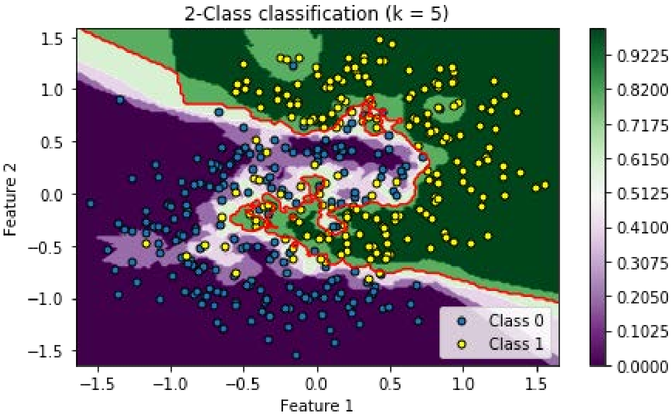
(b) How do you expect the decision boundary to deviate from the ideal boundary you sketched above as k becomes very, very large?[3]
Solution:
As k becomes larger and larger, we will find that we take a cluster as a unit, that is some points which are isolated will be ignored, and we take H0 and H1 as two big group. The H0 points among H1 group will be classified as H1 class, the H1 points among H0 group will be classified as H0 group. When k is large enough, there won't be classifier among H0 and H1 class, the dataset will be recognized as a big group.

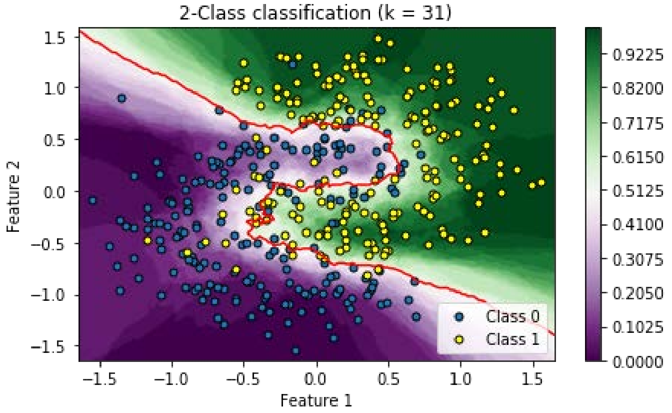
[3] You may also sketch this decision boundary on the above visualization of the `Horseshoe` data if you think that would be helpful... if you do this make sure to clearly label the two decision boundaries you sketch!

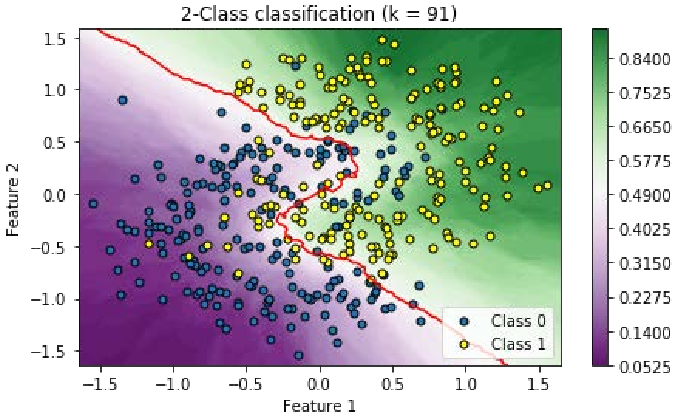
(60) 3. Suppose you are using a K-Nearest Neighbors classifier with the L_2 distance metric.

(a) Visualize the decision statistic surface with both the training data and the majority vote decision boundary[4] superimposed on top of the decision surface for KNN with $k = 1$, $k = 5$, $k = 31$, and $k = 91$.







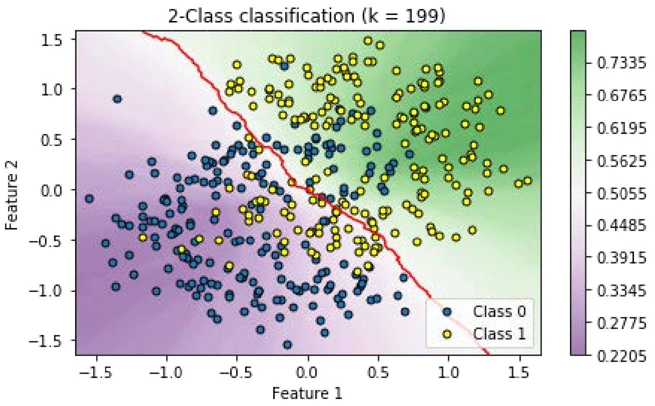


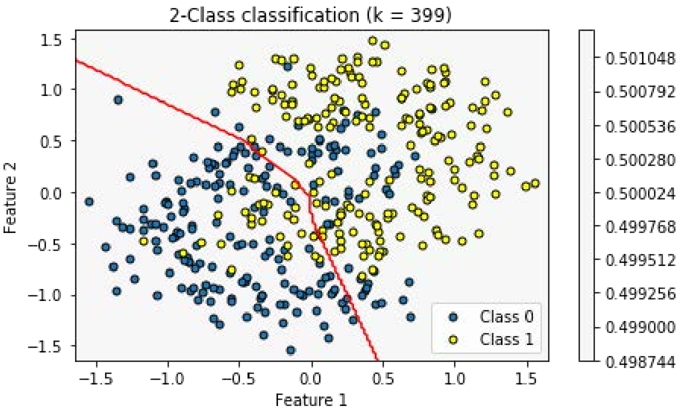
(b) Why does the KNN majority vote decision boundary deviate from your ideal decision boundary as k increases?

Solution:
When k is small, the model become overfitting. Because we consider almost every datapoint for model training. As k increases, when we determined our λ for each data point, we will considered k-1 nearest points around the data we selected, then we have more possibilities and results in more classifiers. When k keeps growing, the dataset will be classified as two classes with smooth boundary. Because isolated points will be ignored. When k outnumbers the dataset, the dataset will be looked as a big group, thus when k is closed to data numbers, the contour field looks like a field.

[4] In Matlab, the `contour` function may be helpful for finding a desired decision boundary.

(c) Visualize the decision statistic surface with both the training data and the majority vote decision boundary superimposed on top of the decision surface for KNN with $k = \frac{N}{2} - 1$ and $k = N - 1$.[5] (N is the total number of training data points.)





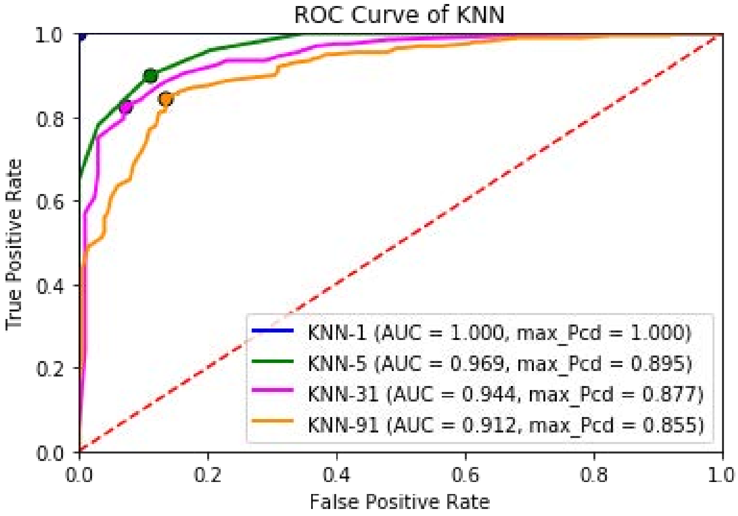
(d) Explain the systematic error (bias) in these classifiers with very large k.

Solution:
We could see that with very large k, especially when k is close to the total number of dataset, H0 and H1 is classified as two group. That is when k becomes very larger , it will consider k-1 nearest neighbors around our selected data point, if H0 is clustered together, most of H0 points on H0 side will be considered as H0, while H0 points on H1 side will be ignored, vice vise. Dataset will be viewed as two clusters. When k becomes really really big, the dataset will be seen as a class, and that make no sense for data training. Because it makes the model underfitting. Thus we should avoid too big k or too small k.

[5] It would be highly unusual to choose k so high in practice, but looking at the full range of values for k, and both boundary conditions on k ($k = 1$ and $k = N$) in particular, can help develop insight. We are looking at $k = N - 1$ instead of $k = N$ so there will be a majority of neighbors in one of the classes.

(80) 4. Suppose you are using a K-Nearest Neighbors classifier with the L_2 distance metric.

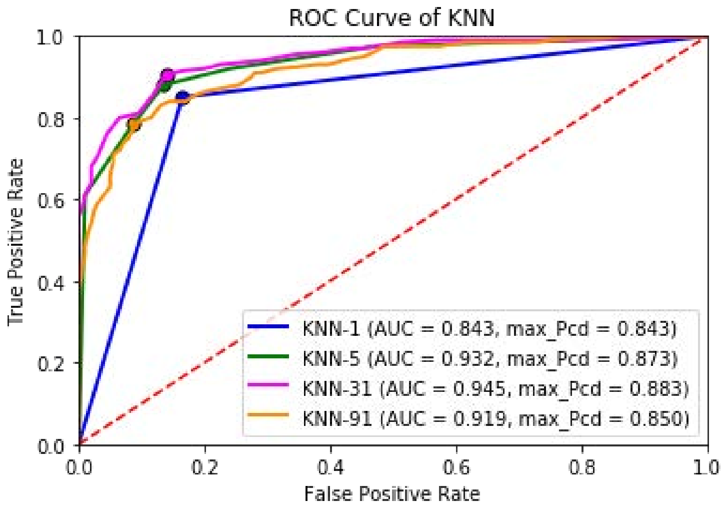
(a) Plot the ROCs for $k = 1$, $k = 5$, $k = 31$, and $k = 91$ when the KNN classifier is trained and tested on the training data, and find the max $\mathrm{P_{cd}}$ for each case.



(b) Based only on these ROCs, what value of k do you recommend to maximize $\mathrm{P_{cd}}$?

Solution:
Based on the ROCs , we could see that we will get maximum Pcd when k is 1.

(c) Plot the ROCs for $k = 1$, $k = 5$, $k = 31$, and $k = 91$ when the KNN classifier is trained on the training data and tested on separate testing data provided in the `csv` file, `dataSetHorseshoesTest`[6], and find the max $\mathrm{P_{cd}}$ for each case.



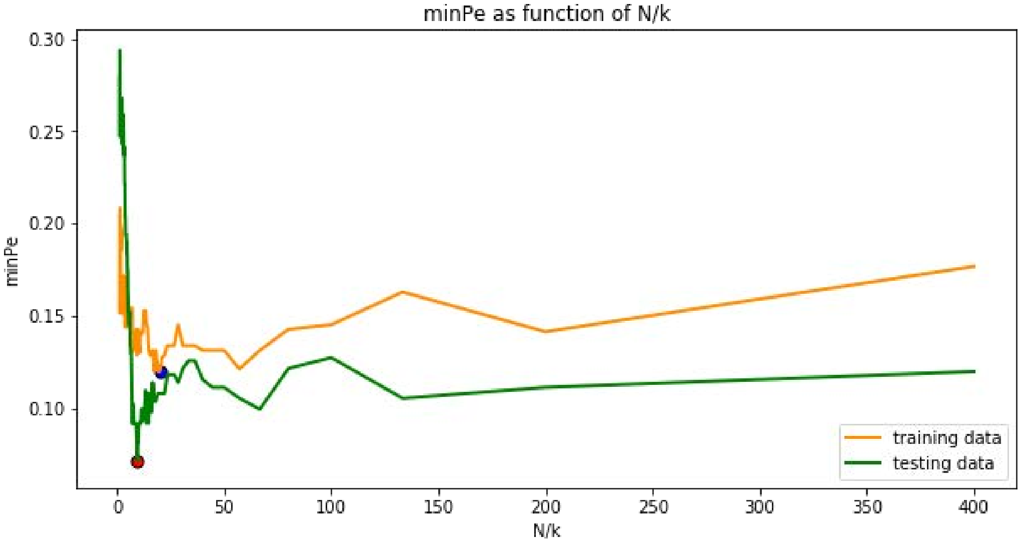
(d) Based only on these ROCs, what value of k do you recommend to maximize $\mathrm{P_{cd}}$?

Solution:
We could see from the graph that we will get maximum Pcd when k is around 31.

[6] 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.

(e) Plot $\min \mathrm{P_e} (= 1 - \max \mathrm{P_{cd}})$ as a function of N/k for a sampling of values of k[7] between 1 and 399 for both testing on the training data and testing on separate testing data.[8]

(Save the data for this graph! You will replicate this graph and add a third curve to it in a future homework on cross-validation.)



training data: k = 20.0
testing data: k = 42.0

(f) Explain how this graph illustrates the principle of bias-variance tradeoff.

Solution:
From the above graph, we know that when k increases, N/k decreases. So k=1, N/k =400. From right to left x axis, minPe changes from big to minimal, then increases to high value again. That is when we start from k=1, we iterated over every data point, the model will become overfitting, thus the error is large. While k increases, we considered k-1 nearest neighbors around our selected point to verify its λ, the model will become smooth, because we will exclude isolated or erroneous points, thus the minPe decreases, and reached to minimum value on the points scattered on line. However, when k keeps growing, the model will become underfitting, because we are taking too many points for a individual point. The model errors for both training and testing will increase. After we reached minimum minPe, when k increases, the error increases sharply. Because we trained the model as two clusters, it became underfitting, thus errors increased sharply.

(g) Based on this graph, what value of k do you recommend to maximize $\mathrm{P_{cd}}$?

Solution:
I would recommend we choose k in range of 20~42, because between this range, the errors for both training data and testing data are very small. Other k will either increase both errors, or makes one error too big, even the other is small. We should choose a k which works for both training data and testing data.

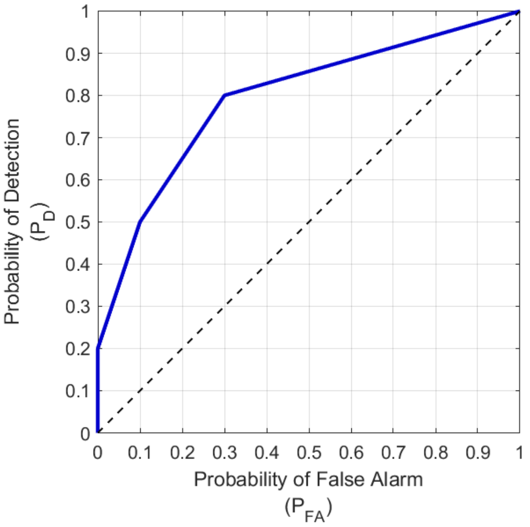
[7] You choose the values of k where you want to sample this curve.

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

Probabilistic Decision Rules

Although ROCs for KNN classifiers can provide (at most) only $k+2$ samples of the ROC because KNN results in one of $k+1$ possible decision statistics $\lambda(x) \in \left\{0, \frac{1}{k}, \frac{2}{k}, \ldots, \frac{k-1}{k}, 1\right\}$ (so $\beta = \left\{0, \frac{1}{k}, \ldots, 1, +\infty\right\}$ to ensure the ROC spans from $(\mathrm{P}_{FA}, \mathrm{P}_D) = (1,1)$ to $(\mathrm{P}_{FA}, \mathrm{P}_D) = (0,0)$), we are not restricted to operating at one of the $k+2$ samples of the ROC found by sweeping through possible thresholds. It is possible to choose an operating point on the straight line connecting two samples of the ROC by using a probabilistic decision rule.

The `csv` file `knn3DecisionStatistics.csv` contains a set of decision statistics produced by a KNN classifier with $k = 3$. This csv file is organized such that each row contains the true class (either 0 or 1) followed by the associated decision statistic. When you plot the ROC curve, you should see this:



(30) 5. This ROC has operating points with $\mathrm{P_D} = 0.8$ for $\beta = 1/3$ and $\mathrm{P_D} = 1$ for $\beta = 0$. You want to operate at $\mathrm{P_D} = 0.95$.

(a) What probabilistic decision rule will allow you to operate at $\mathrm{P_D} = 0.95$ (on average)?

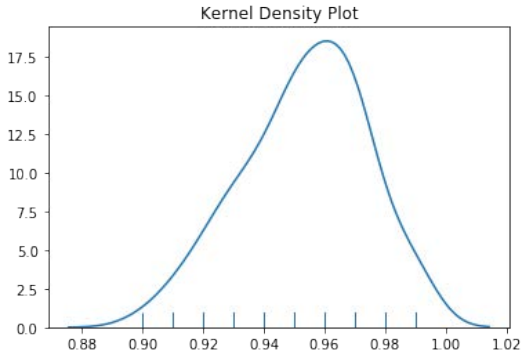
We could apply 0.75 probabilistic decision on (λ<1/3| H1) to label the points as H1.

That is for points which truth are H1, while their λ<1/3, we give them 0.75 possibility to be labeled as H1.

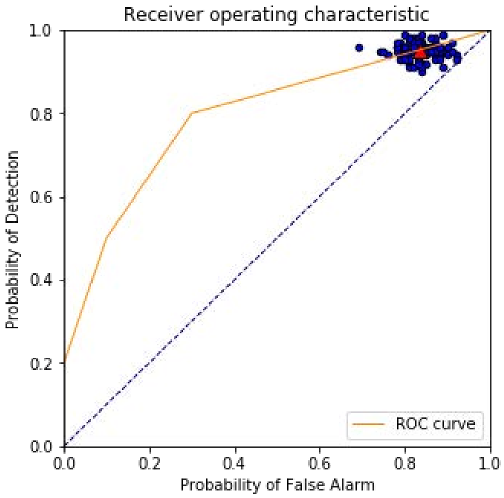
(b) Simulate your probabilistic decision rule, and show that although you may not achieve $P_D = 0.95$ in any given simulation, the expected value of P_D over many simulations converges to 0.95.

Each individual simulation provides an estimate of the probability of detection $\hat{P}_D$, as well as an estimate of the probability of false alarm $\hat{P}_{FA}$.

Run the simulation a large number of times to estimate the pdf of $\hat{P}_D$ via kernel density estimation[9] and find the expected value of $\hat{P}_D$, $\mathbb{E}\left\{\hat{P}_D\right\} = \mu_{\hat{P}_D}$.



(c) Plot the ROC with the expected operating point $\left(\mu_{\hat{P}_{FA}}, \mu_{\hat{P}_D}\right)$ from your probabilistic decision rule indicated on the ROC. Does your operating point fall on (or very near to) the line connecting the $\beta = 1/3$ and $\beta = 0$ operating points?



expectedPd is: 0.9504
expectedPfa is: 0.8351

Yes, you could see from the graph that our operating point fall on the line connecting the β = 1/3 and β = 0 operating points.

[9] In Matlab, `ksdensity` generates kernel density estimates.