Pattern Classification and Recognition:

Classifier Performance Evaluation

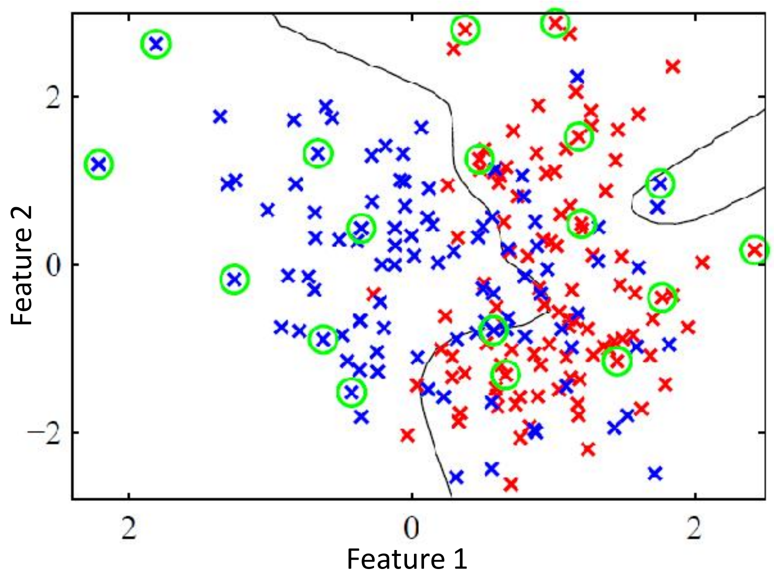
ECE 681

Spring 2016

Stacy Tantum, Ph.D.

Goal of Classifiers

Correctly classify a previously unseen data instance
(with high probability)



Define Your Problem!

Weight [g]	Wingspan [cm]	Webbed Feet?	Back Color	Species
1000.1	125.0	No	Brown	Buteo jamaicensis
3000.7	200.0	No	Gray	Sagittarius serpentarius
4100.0	136.0	Yes	Black	Gavia immer
3.0	11.0	No	Green	Calothorax lucifer
570.0	75.0	No	Black	Campephilius principalis

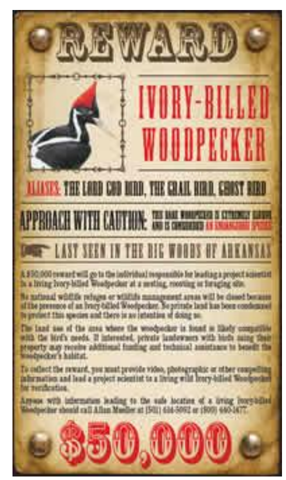
Black birds vs. non-black birds?

Swimming birds vs. non-swimming birds?

Small birds vs. big birds?

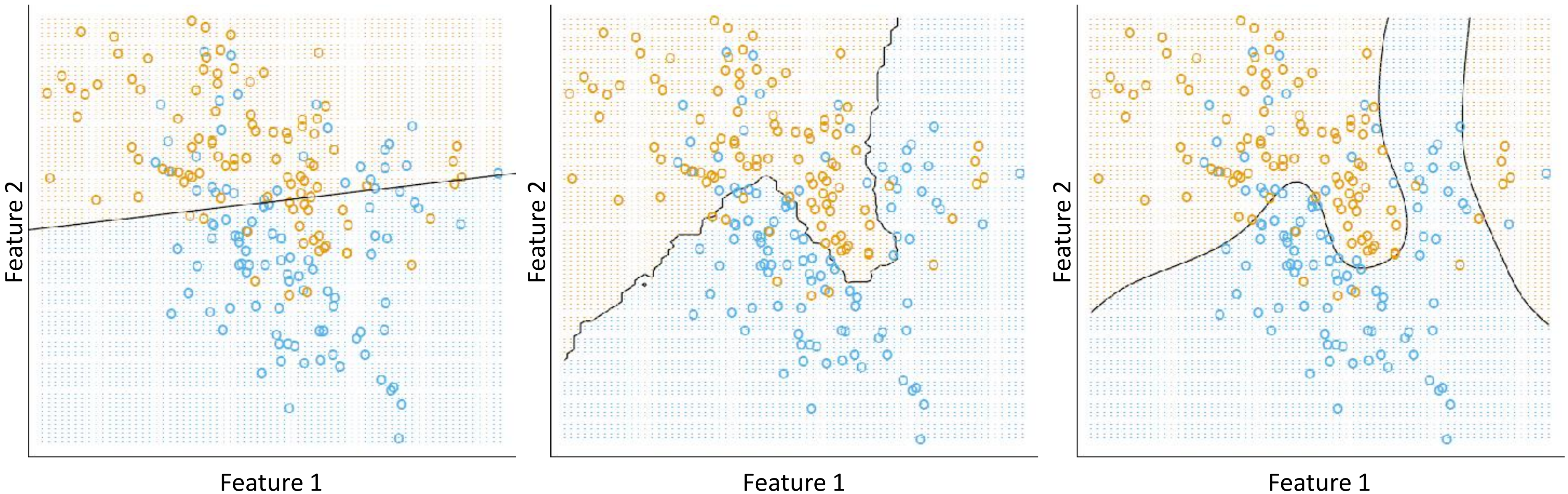
Colorful birds vs. drab birds?

Ivory-billed woodpecker vs. all others?
($50,000 reward!)



Why Evaluate Classifier Performance?

Choose among candidate classifiers

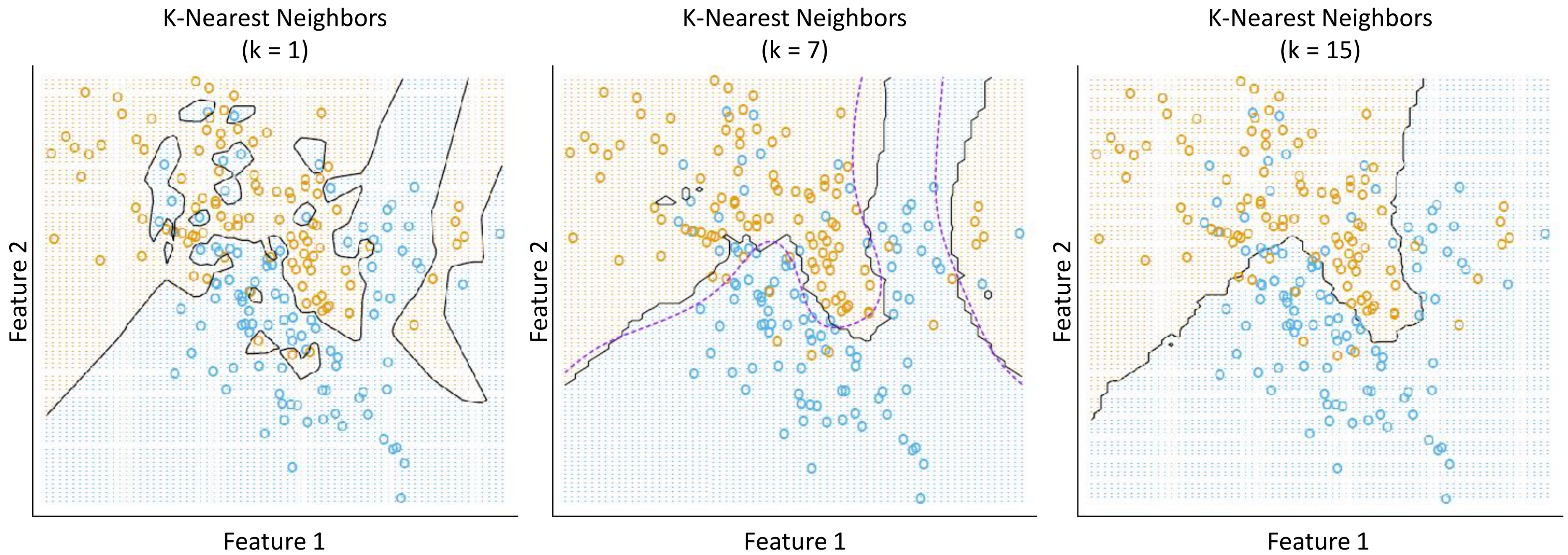


“No Free Lunch” Theorem

No classifier is inherently superior (or inferior) to all others

Why Evaluate Classifier Performance?

Compare/choose classifier parameter(s)

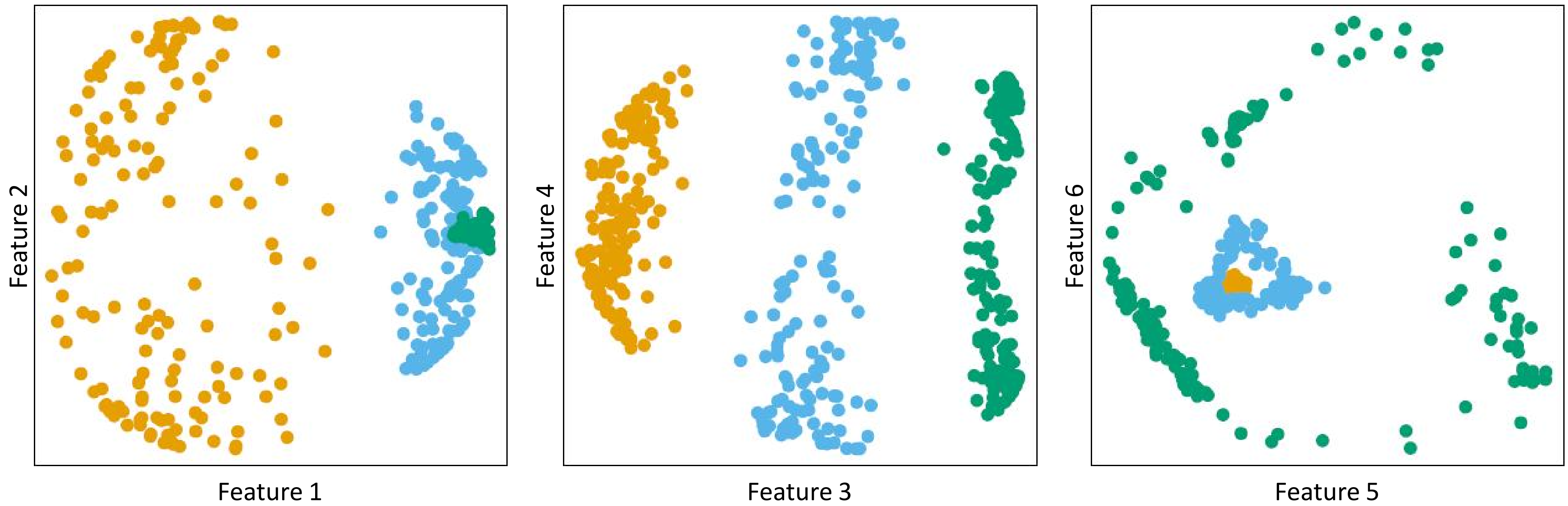


Occam's Razor
(avoiding overfitting/overtraining)

Classifiers should be no more complicated than necessary

Why Evaluate Classifier Performance?

Compare/choose feature subsets

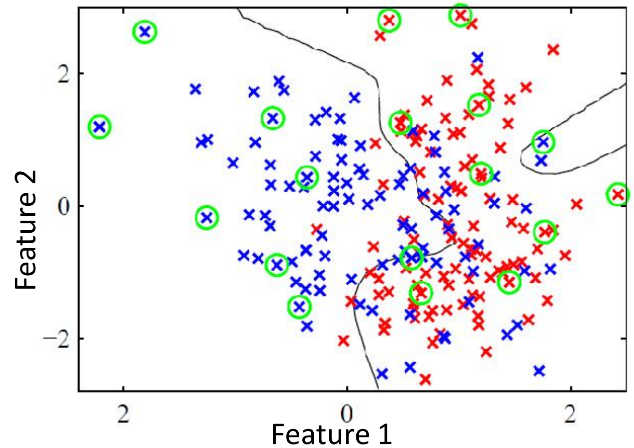


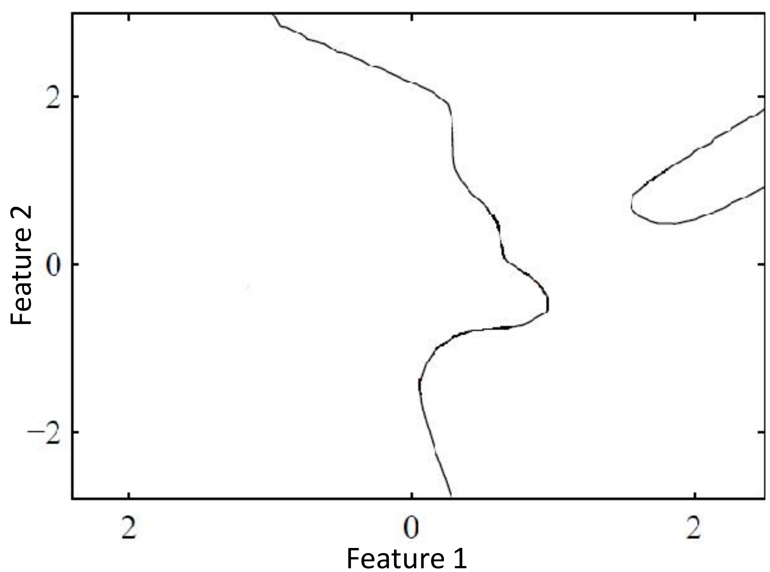
"Ugly Duckling" Theorem

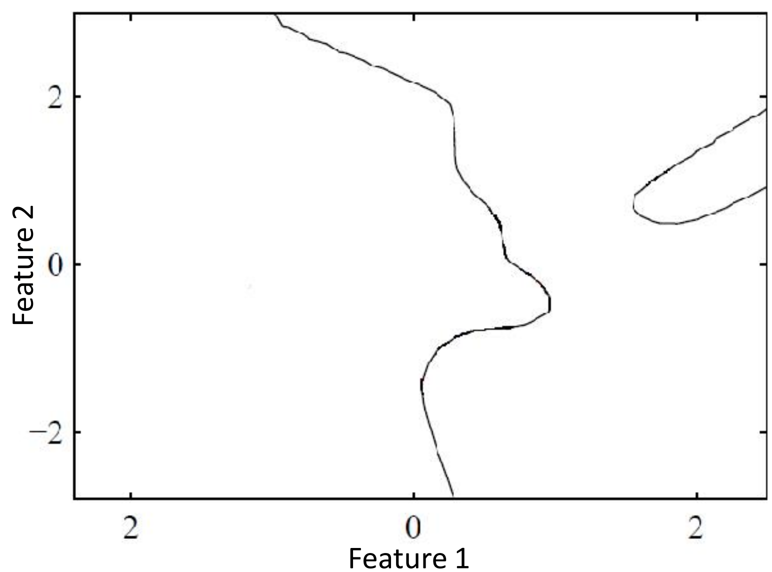
No feature representation is inherently superior (or inferior) to all others

Why Evaluate Classifier Performance?

Predict likely performance when deployed



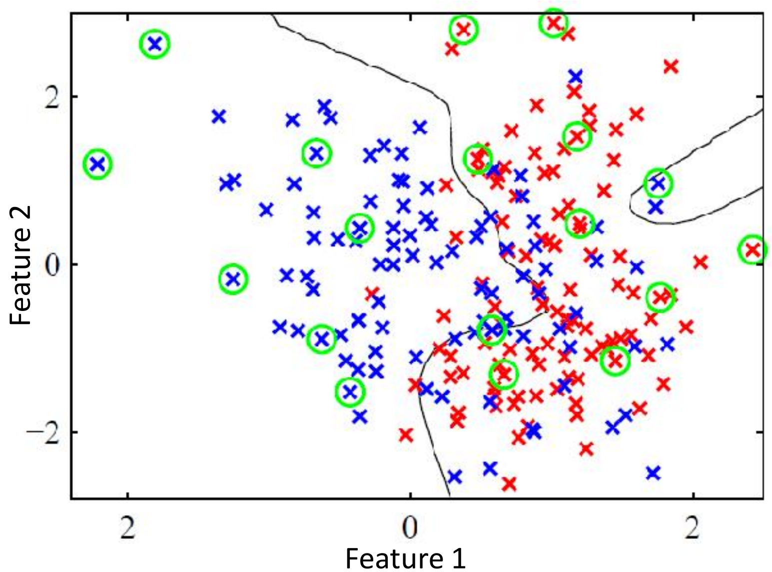


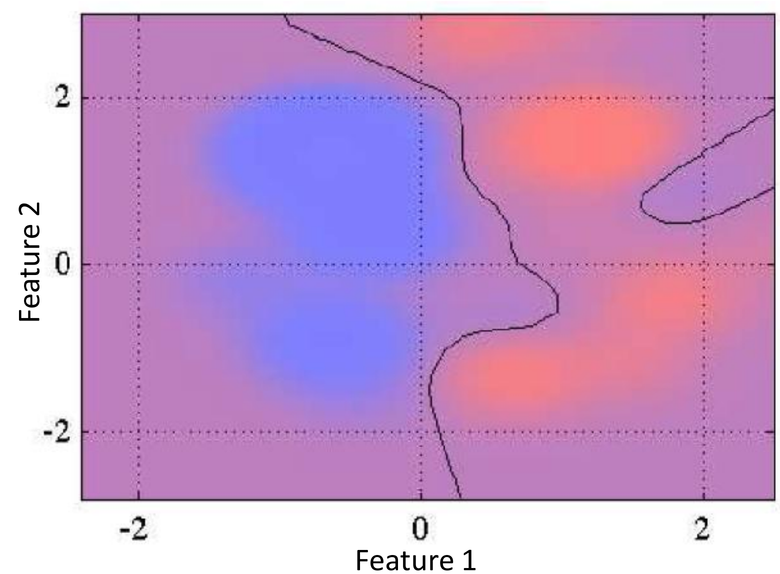


(Binary) Decision Statistics

Classifiers transform the set of features for a data instance to a single number that forms the basis for making a decision

$$\lambda = f(x_1, x_2, \ldots x_N)$$

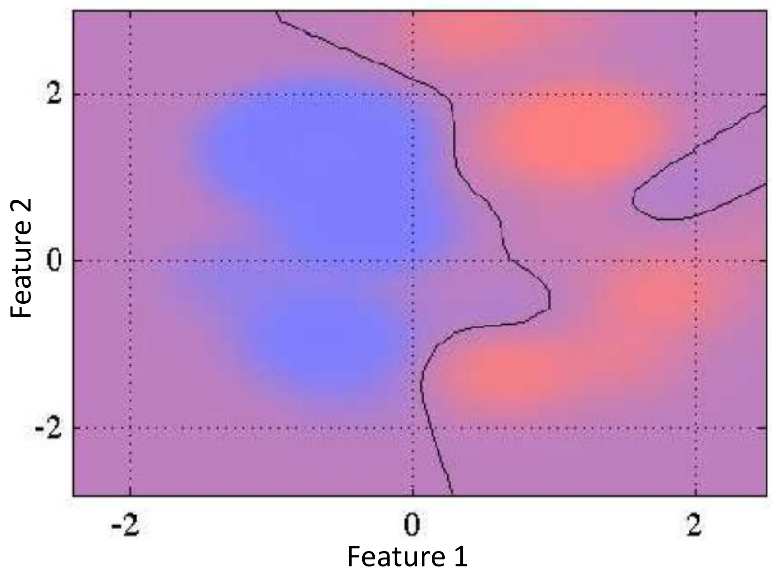




Binary Decision Outcomes

Binary Hypotheses

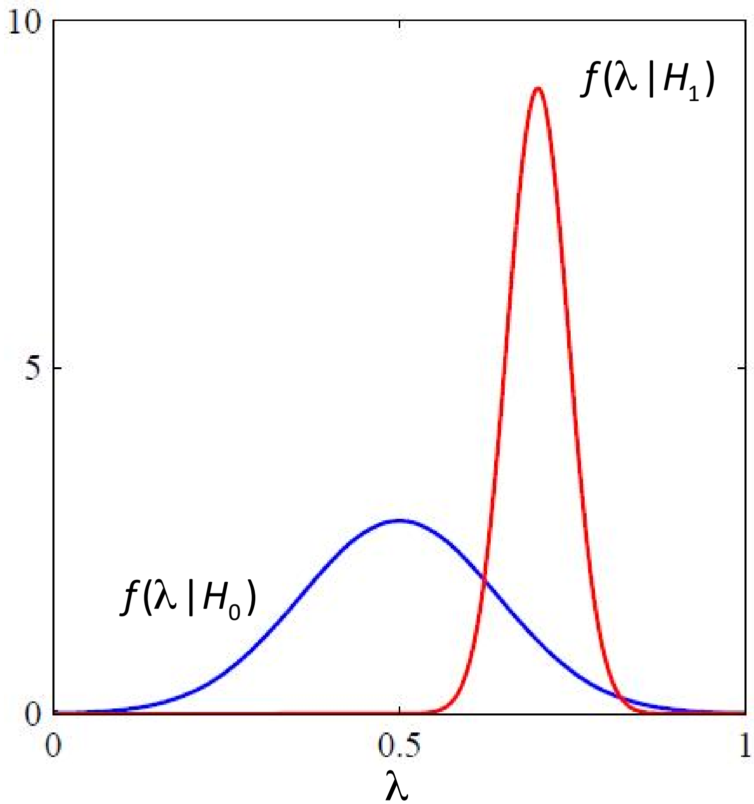
H_0: Data ***does not*** come from the class-of-interest
H_1: Data comes from the class-of-interest

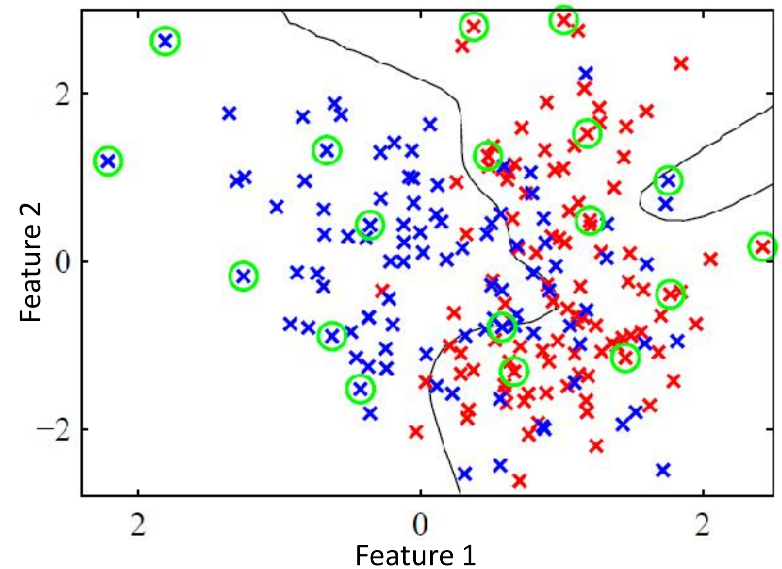


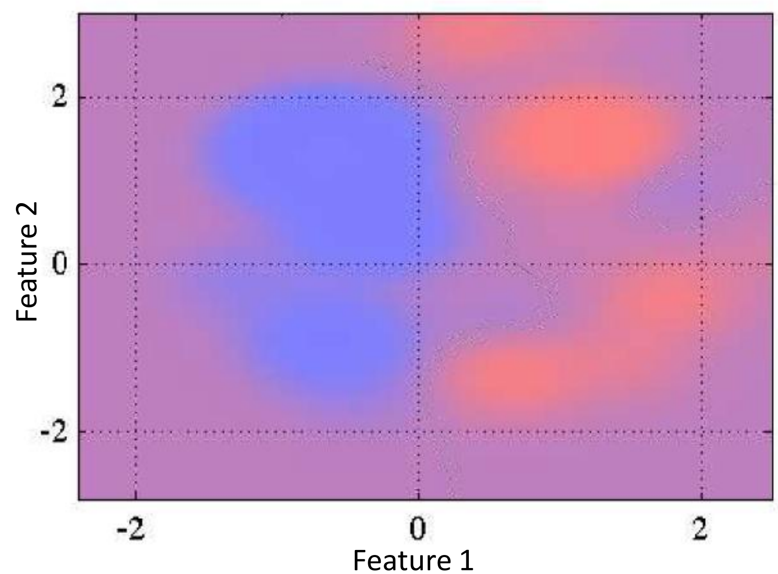
Distributions of Decision Statistics

pdfs of decision statistics for:

- All H_0 data, $f(\lambda \mid H_0)$
- All H_1 data, $f(\lambda \mid H_1)$







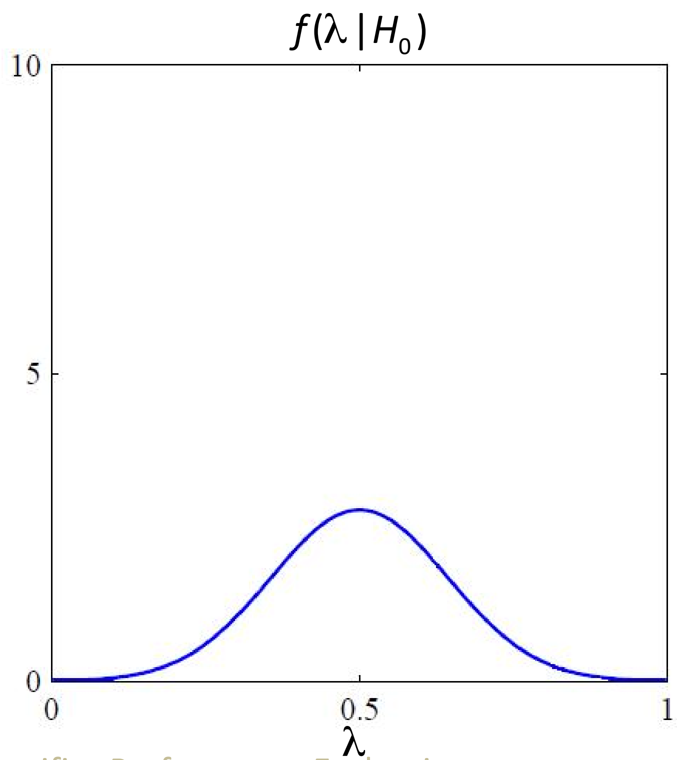
Binary Decision Performance Evaluation

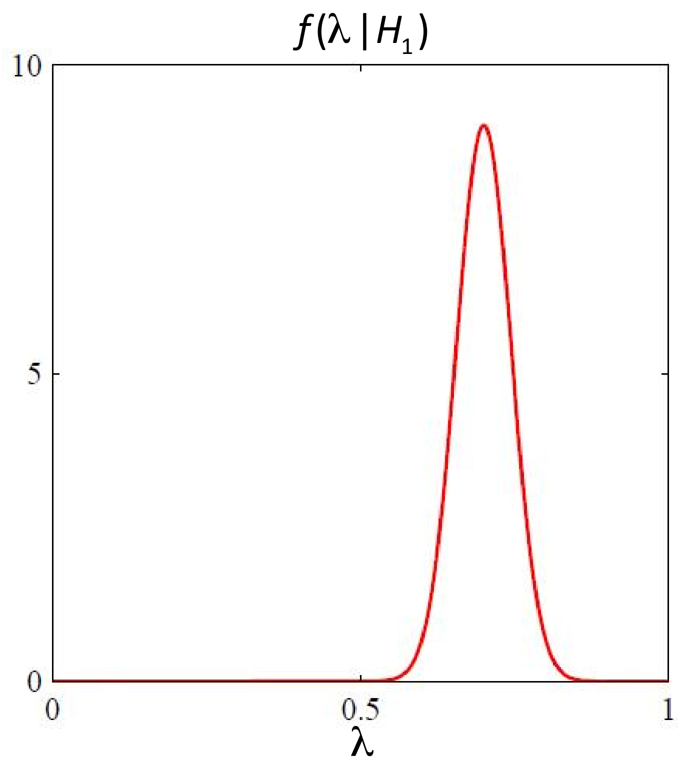
$P_{CR}(\beta) =$

$P_M(\beta) =$

$P_{FA}(\beta) =$

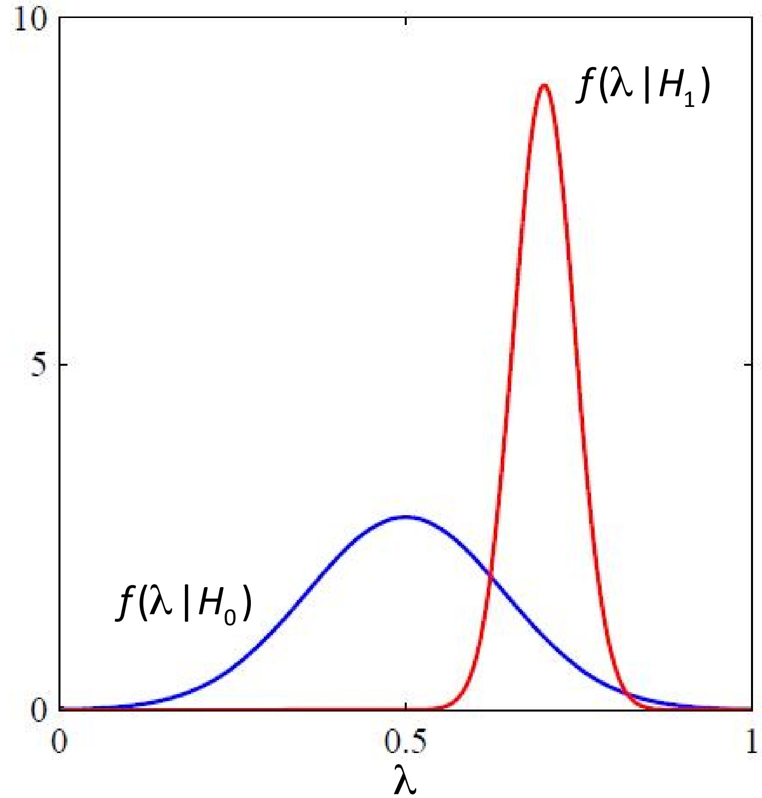
$P_D(\beta) =$

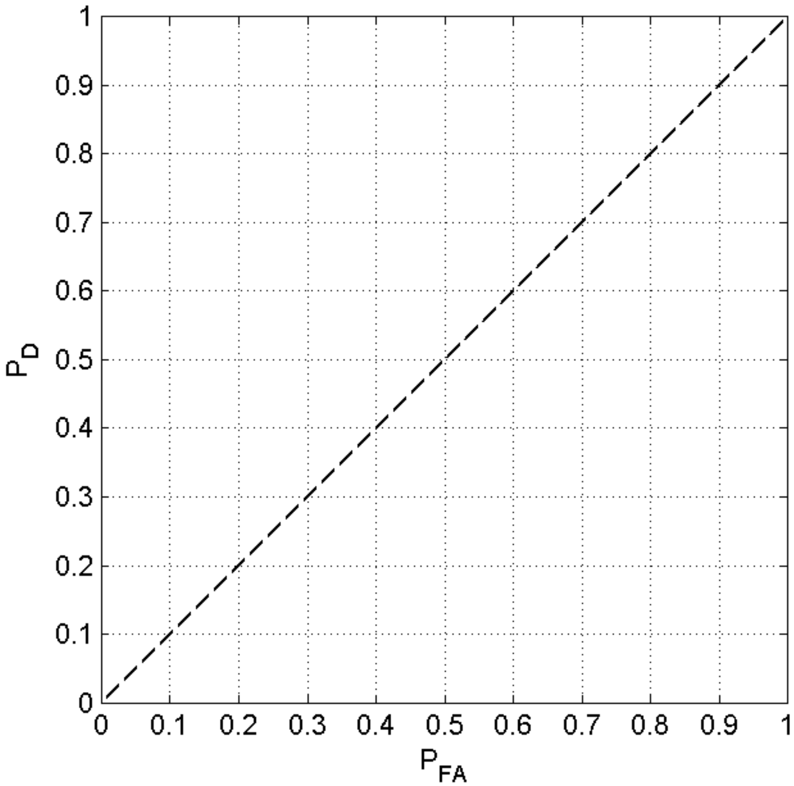




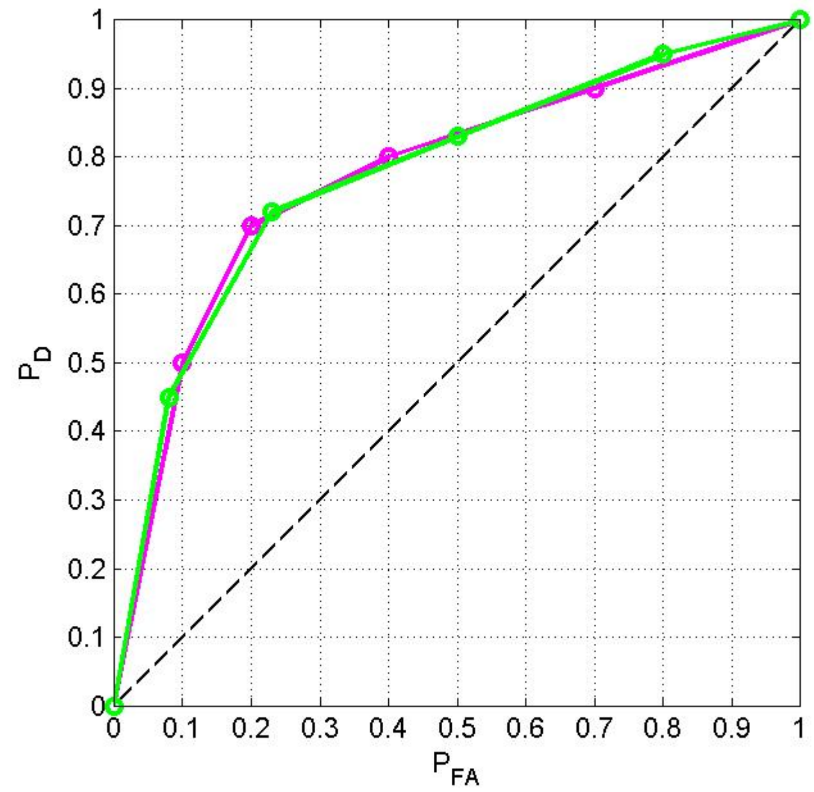
Generating an ROC (Receiver Operating Characteristic)

Sweep the threshold to generate P_{FA}/P_D pairs

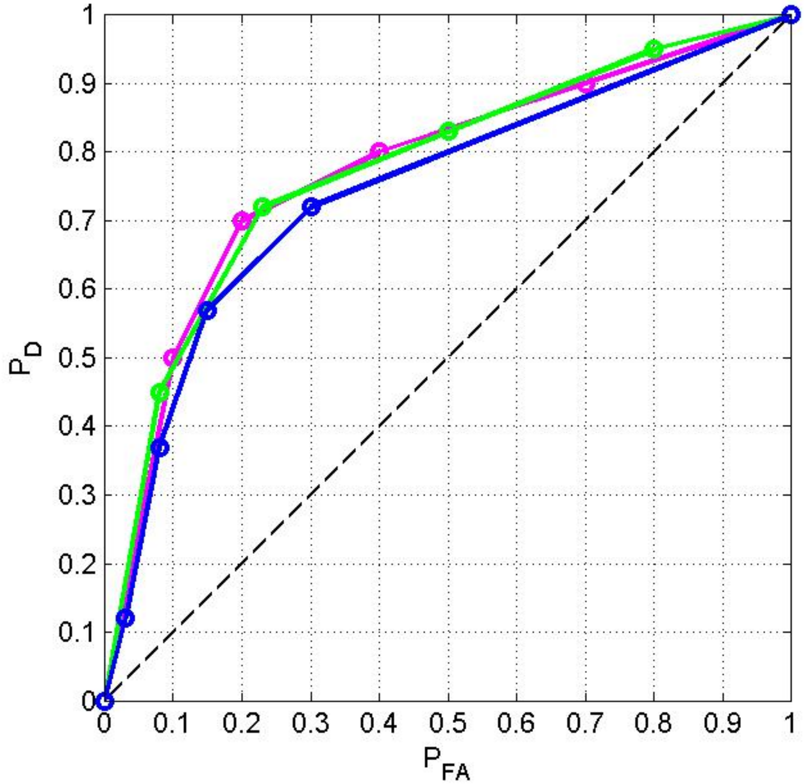




Averaging ROCs

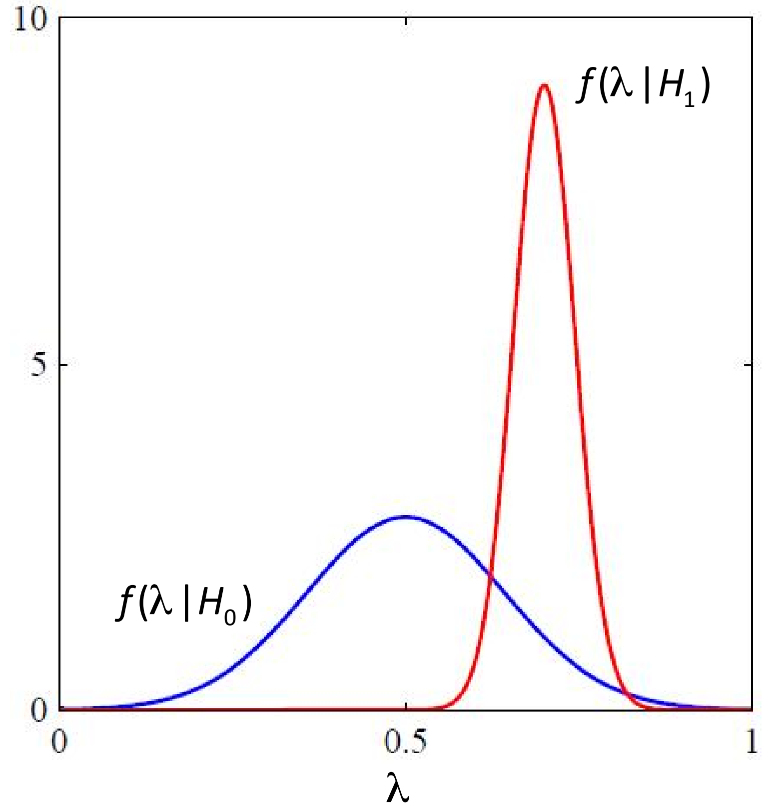


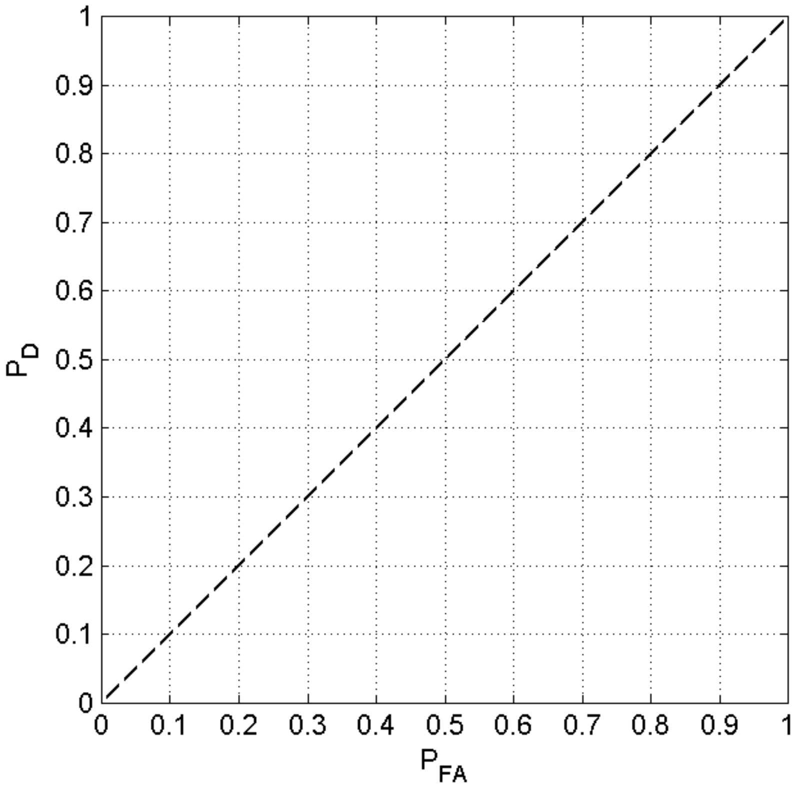
Averaging ROCs



Generating an ROC Another Way

Choose P_{FA}, determine the corresponding threshold, find P_D

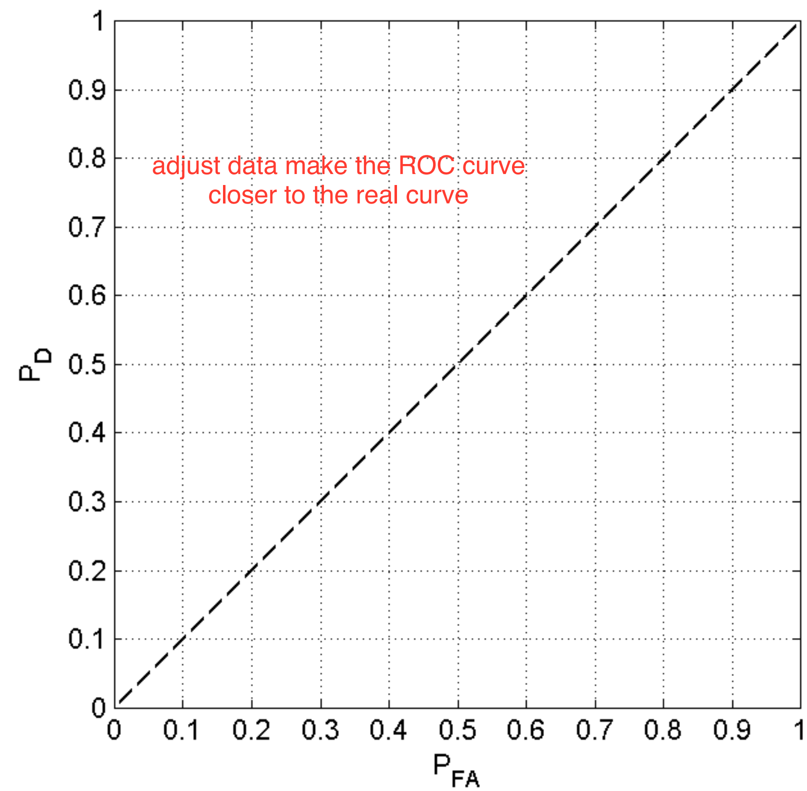




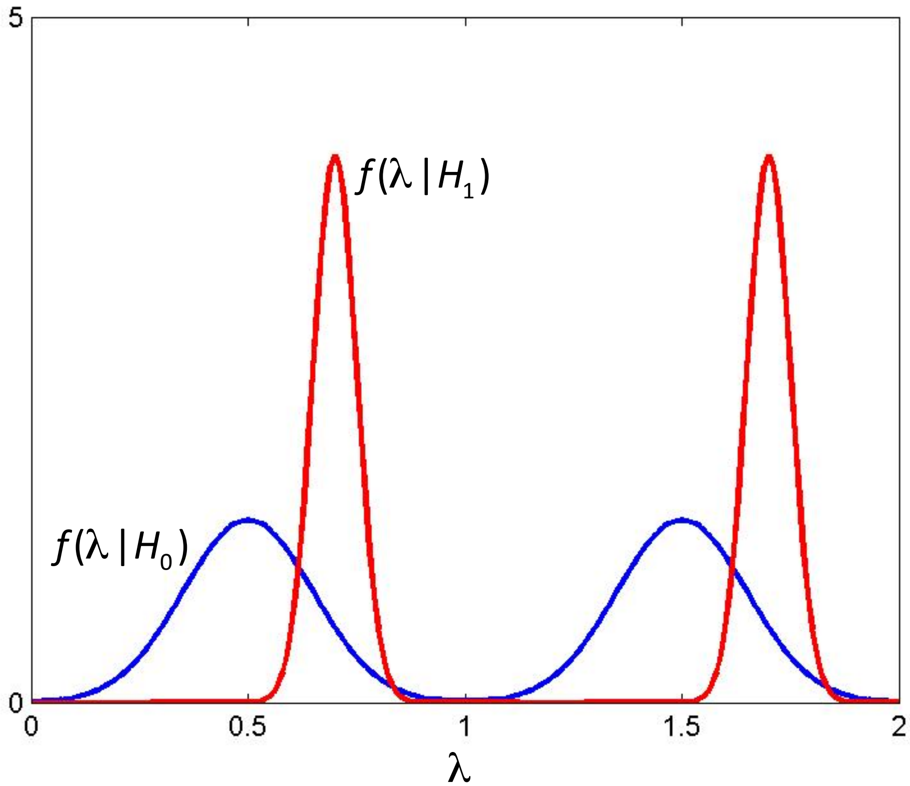
Generating an ROC Yet Another Way

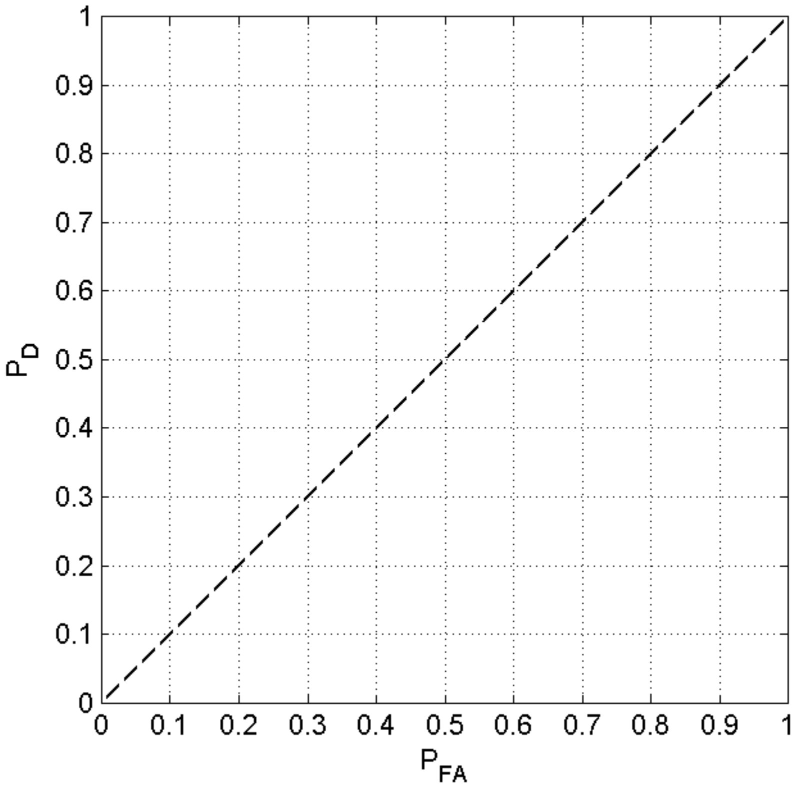
Sort data instances by decision statistic, and choose every N^{th} decision statistic as a threshold (N=1 for finest resolution ROC)

□	target	□	P_{FA}/P_D	□	P_{FA}/P_D
original data set		threshold		-Inf	
0	0	0		0.01	
0.11	0	0.11		0.12	
0.18	1	0.18		0.19	
0.21	0	0.21		0.22	
0.35	0	0.35		0.36	
0.42	1	0.42		0.43	
0.56	0	0.56		0.57	
0.82	1	0.82		0.83	
0.88	1	0.88		0.89	
0.92	1	0.92		0.93	
		Inf		adjusted threshold	

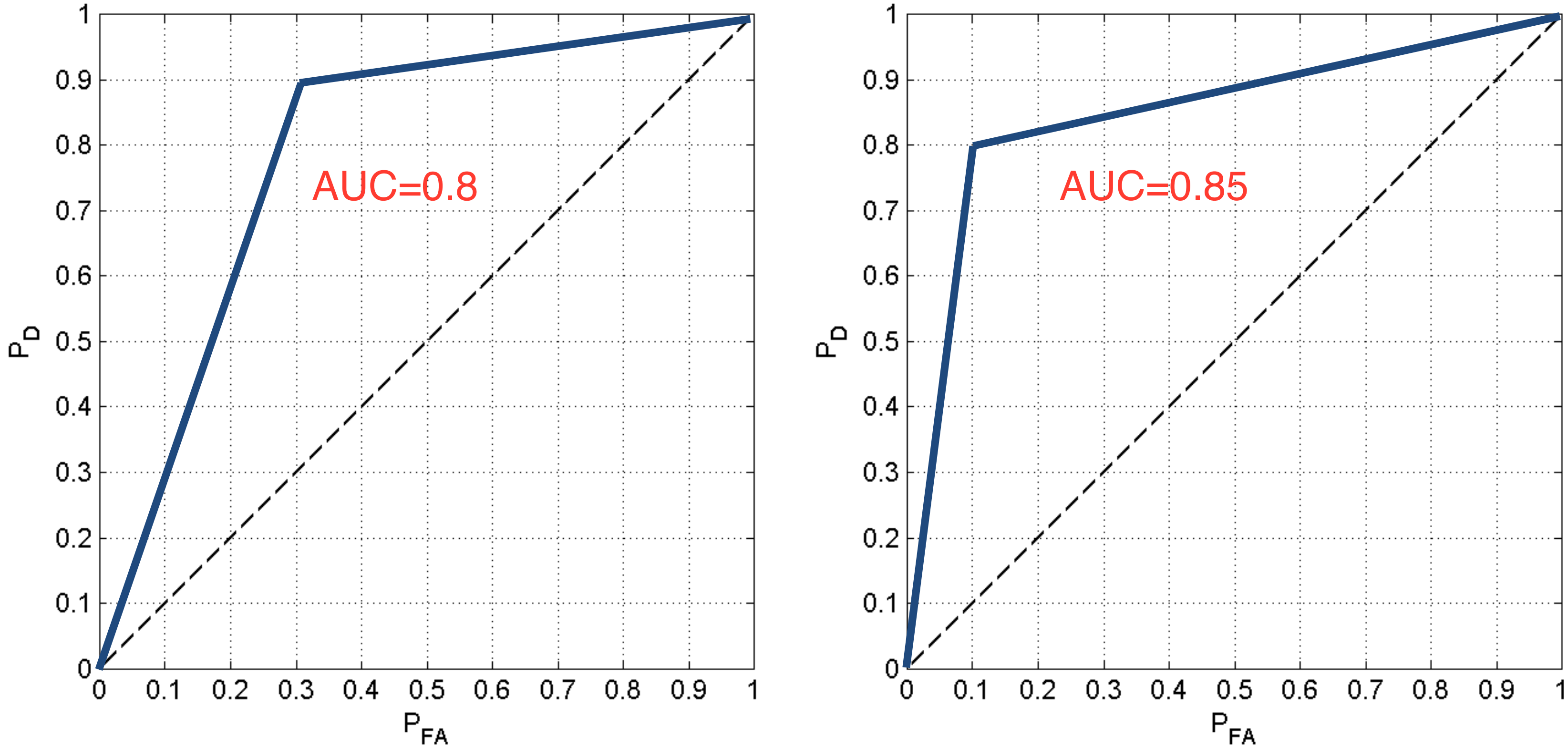


Decision Statistic pdfs & ROCs





Comparing Classifier Performance



AUC: Area under curve to estimate the performance

Pfa@Pd: lower =>ROC is more toward left @ fixed Pd

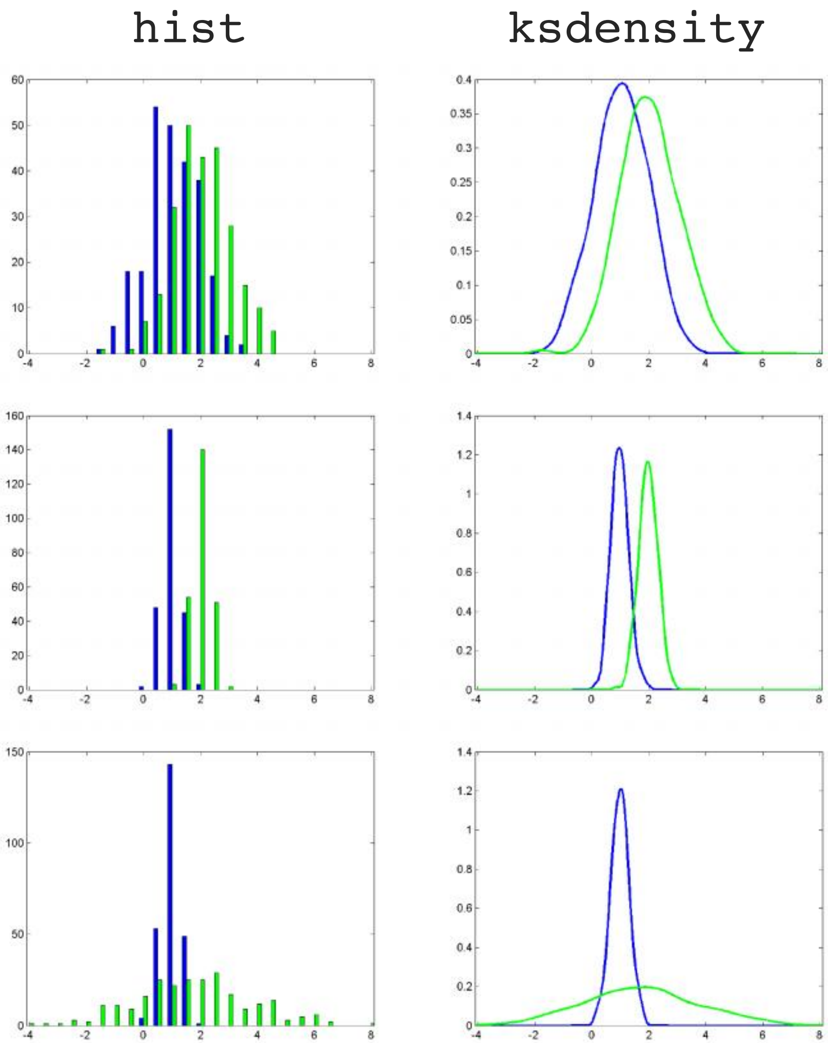
Pfa@Pd: lower =>ROC is more toward left

ROC Coding Tips

[pF,pD] = generateROC(decisionStatistic,target,rocOptions)

- option to generate ROC for fixed thresholds
  - *easiest to implement, a good start for a first ROC*
  - *nice to have as an option, to compare performance at different fixed thresholds, especially when a fixed threshold needs to be chosen to deploy the system*
  - *generally not recommended for performance evaluation during classifier development – linear spacing, logarithmic spacing, other?*
- option to generate ROC at fixed P_{FA}'s (or fixed P_D's)
  - *good for being able to average ROCs (and aesthetics)*
- option to generate ROC with threshold for every *N* data instance (sorted by decision statistics)
  - *N = 1 for "stair-step" ROC (most common use)*
  - *careful with N too large!*
  - *may or may not be able to average ROCs easily*

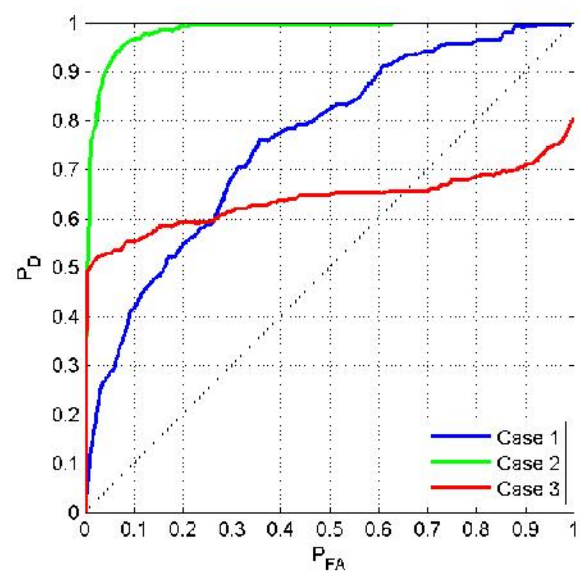
Visualizing Decision Statistics: pdfs



Visualize distributions
for each class separately

Smaller overlap between decision statistic pdfs:

- Fewer opportunities for mis-classification in overlap region (better performance)
- Higher ROC (closer to top-left corner of axes)

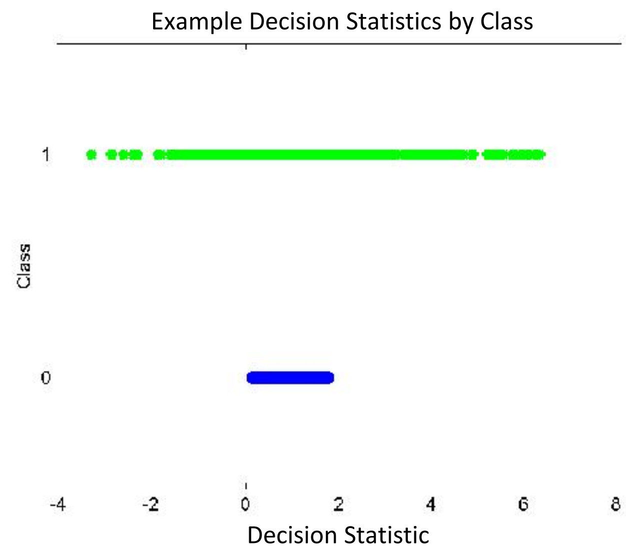


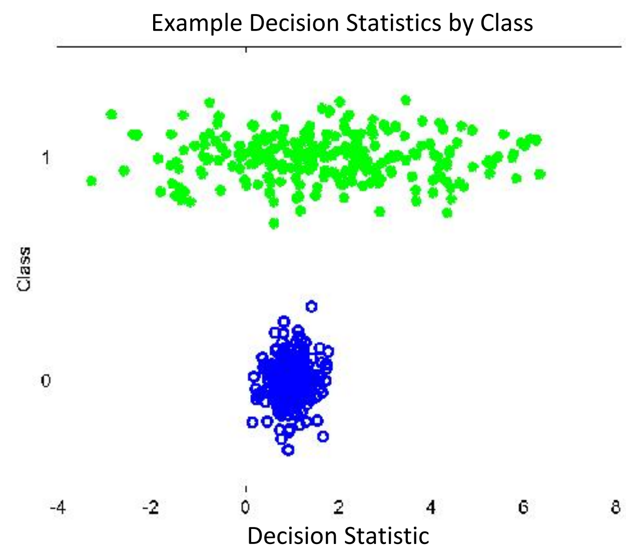
Visualizing Decision Statistics: Scatter Plots

`<your own function>`

Plot the decision statistic on the x-axis and class (target) on the y-axis

- Similar to plotting pdfs, but it may be easier to tell where individual decision statistics are falling in regions where there aren't many of them
- A few high H_0 decision statistics or low H_1 decisions statistics can make the ROC look weird – this can help you understand that weirdness
- Can add a small amount of noise to the class variable to separate similar decision statistics



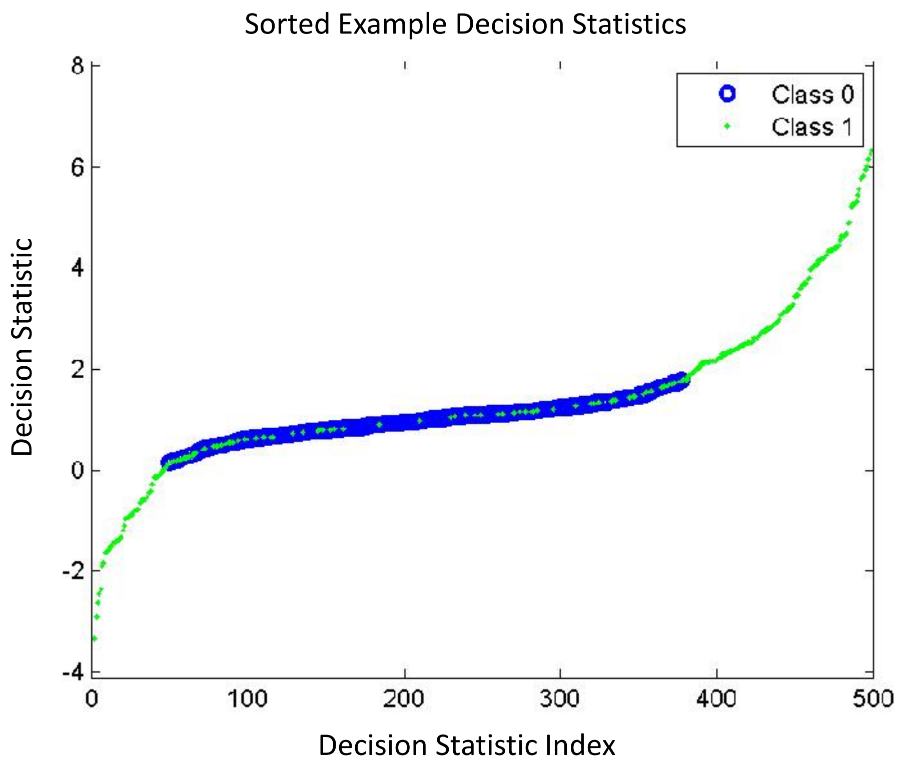


Visualizing Decision Statistics: Sorted Plots

```
<your own function>
```

Sort the decision statistics from smallest to largest, and plot each one in a symbol that corresponds to its class

- Similar to plotting cdfs, but it may be easier to tell where individual decision statistics are falling in regions where there aren't many of them
- Again, a few very high H_0 decision statistics or very low H_1 decisions statistics can make the ROC look weird – this can help you understand that weirdness



Logical Indexing

Allows for selection of a subset of elements without explicitly finding the element indices

Useful for selecting elements by target class

- See example provided in HW #1

```
% Demonstration of logical indexing


% Vector of 100 random numbers
randNum = randn(100,1);

% Mean of all random numbers
meanAll = mean(randNum)

% Mean of random numbers > 0
idxGTzero = find(randNum>0);
meanGTzero = mean(randNum(idxGTzero));

% Mean of random numbers > 0
% with logical indexing
meanGTzero_LI = ...
     mean(randNum(randNum>0));
```

Generating ROCs: Fixed Thresholds

```
<your own function>
```

Choose thresholds that cover the range of the decision statistics

- Linearly-spaced min to max
- Log-spaced min to max
- All decision statistics sorted min to max
- Every n^{th} decision statistic after sorting min to max

For each *unique* threshold β in the list

- Calculate P_{FA}
  - *(# H_0 DecStat ≥ □) / (# H_0 DecStat)*
- Calculate P_D
  - *(# H_1 DecStat ≥ □) / (# H_1 DecStat)*

Generating ROCs: Fixed P_{FA}s (or fixed P_Ds)

```
<your own function>
```

For each P_{FA} in the list

- Determine the associated threshold
  - *P_{FA}*(#H_0 DS) = #H_0 DS ≥* $\square$
  - *[P_{FA}*(#H_0 DS)] will not be an integer…*
    I leave it to you to determine if `floor()`, `ceil()`, *or* `round()` *is the proper conversion to an integer number*
  - *This integer is the index into the vector containing the H_0 decision statistics*

The result is a list of fixed thresholds that will produce P_{FA}s close to the desired P_{FA}s

- May want to calculate actual P_{FA} for each threshold (or not, if averaging ROCs)
- This may produce duplicate thresholds
- Important good practice (*<u>if</u>* producing a single ROC):

```
beta = unique(beta);
```

  - *Do not eliminate duplicate thresholds if you will be averaging ROCs*

Plotting ROCs

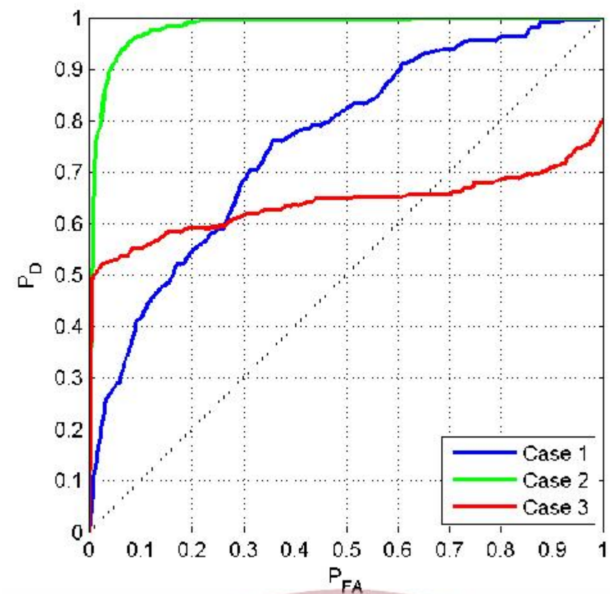
Strongly suggest you create
`<your own function>`

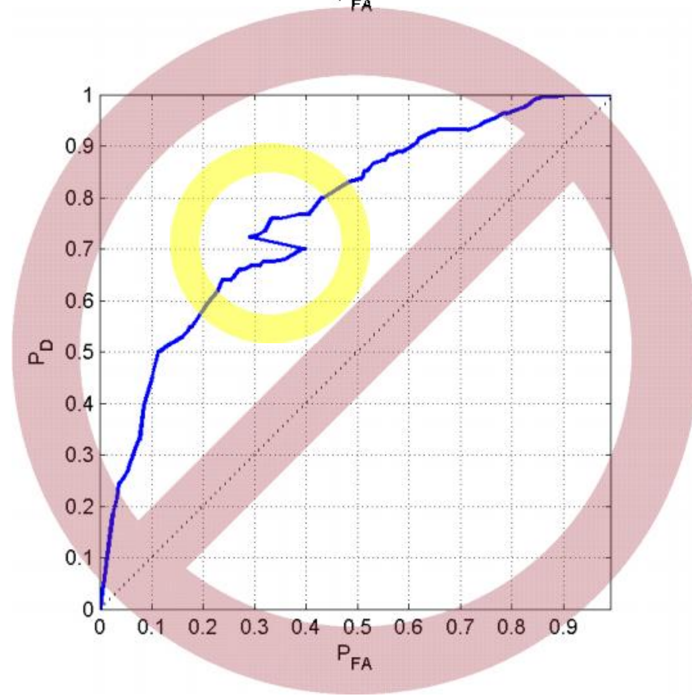
Plot the lines between the pF/pD data points

Symbols alone can be difficult to interpret if there are only a few P_{FA}/P_D points on the ROC

As threshold increases

- P_{FA} cannot increase
- P_D cannot increase
- If you see either P_{FA} or P_D increase when you increase the threshold, something is very wrong





Calculating AUC

Trapezoidal integration (`trapz`) is a robust numerical method

- Provides an exact calculation for piece-wise linear curves – which is exactly what our ROC curve is
- Appropriate when variable of integration (P_{FA} or P_D) is not precisely evenly spaced
- Appropriate when the variable of integration is repeated, as for a “stair-step” ROC