Introduction to Machine Learning:

Nearest Neighbor Classification

ECE 580
Spring 2022

Stacy Tantum, Ph.D.

~x63 $w_0 + \underset{\sim}{A'}\,\underset{\sim}{\gamma'}$ ~x64 $\underline{A}\,\underline{\gamma}$

Re-casting Corrupt Image Reconstruction to "standard" LASSO

scalar + vector = vector

CORRUPT IMAGE MODEL

$A\gamma$ = weighted sum of basis functions

$$\min_{\gamma} \|\overbrace{A\gamma} - B\|_2^2 + \lambda \underbrace{\|\gamma\|_1}$$

→ Do not include γ for a "constant" basis function in regularization

$\underline{A} = \underline{I}_{x,y}$ = $\underline{I}$ sampled at rows corresponding to (x,y) locations of <u>sensed</u> pixels

- Columns of A are basis functions

$\underline{\gamma}$ = DCT coefficients (i.e., weights for basis functions) $= \begin{bmatrix} \gamma_{1,1} \\ \hline \gamma' \end{bmatrix}$

$$\underline{A} = \left[\begin{array}{c|c} | & \overbrace{\begin{array}{ccc} | & & | \\ \underline{I}_{xy}(u=1,v=2) & \cdots & \underline{I}_{xy}(u=K,v=K) \\ | & & | \end{array}}^{\underline{A'}} \\ \underline{c} & \\ | & \end{array} \right] = [\,\underline{c} \mid \underline{A'}\,]$$

1st column is a vector of constants c $[\underline{I}_{xy}(u=1,v=1)]$

$$\min_{\gamma} \|(\underline{c}\,\gamma_{1,1} + \underline{A}'\underline{\gamma}') - \underline{B}\|$$

Make this look like this

$\underline{c}\,\gamma_{1,1} = \boxed{w_0}$

$\underline{c}\,\frac{w_0}{\underline{c}} = w_0$

$\underline{1} \cdot w_0 = w_0$

"STANDARD" LASSO IMPLEMENTATION

$$\text{Basis Fns} = \begin{bmatrix} | & & | \\ b_1 & \cdots & b_N \\ | & & | \end{bmatrix} = \underline{F}$$

→ not "constant"

$$\text{Weights} = \begin{bmatrix} w_1 \\ \vdots \\ w_N \end{bmatrix} = \underline{w}$$

$$\min_{w} \|\underline{F}\underline{w} - \underline{t}\|_2^2 + \lambda \|w\|_1$$

$\underline{t}$ = target variables

Include intercept? (offset, bias, constant, DC term)

"col 1" cols 2→63

$$\min_{w} \|(\cancel{w_0} + \underline{F}\underline{w}) - t\|_2^2 + \lambda \underbrace{\|w\|_1}$$

← implicitly $w_0 \cdot 1$ (not $w_0 \cdot c$)

→ w_0 not included in regularization

Discussion Assignments late submission / re-scoring update

I think I'm getting close to adding all the deadline extensions (11:59pm Sunday 2/27)

Ethical Frameworks discussions

- Discussion preparation
  - Due 11:59pm the night before the discussion
  - Accepted until class starts at 10:15am with 3.2-point late penalty

If discussion preparation not submitted or received less than 6.3 (full credit with late submission penalty), then you may submit (or re-submit) with the late submission penalty

- Discussion reflection
  - Due 11:59pm the date of the discussion
  - Accepted 1 day late (11:59pm the following day) with no late penalty
  - Accepted 2 days late (11:59pm 2 days later) with 3.2-point late penalty

If discussion reflection not submitted or received less than 6.3 (full credit with late submission penalty), then you may submit (or re-submit) with the late submission penalty

MP Peer Feedback

- Feedback preparation
  - Due at 10:00am the date of the feedback session
  - Accepted until class starts at 10:15am with no late penalty
- Feedback reflection
  - Due 11:59pm the date of the feedback session
  - Accepted 1 day late (11:59pm the following day) with no late penalty
  - Accepted 2 days late (11:59pm 2 days later) with 3.2-point late penalty

If discussion reflection not submitted or received less than 6.3 (full credit with late submission penalty), then you may submit (or re-submit) with the late submission penalty

Nearest Neighbor Classification (k=1)

Classify a previously unseen data instance as
the class of the training data point it most closely resembles

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
4.3	14.8	No	Green	???
600.0	80.0	No	Black	???
785.0	100.0	No	Dark Brown	???

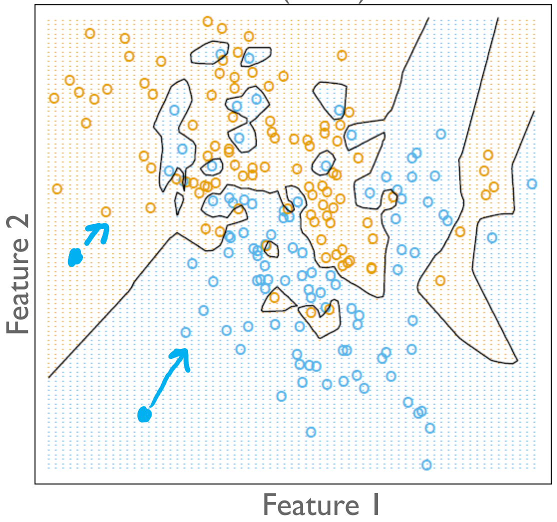
"most similar"

"nearest"

k-Nearest Neighbors (KNN) Classification

What does it do?

K-Nearest Neighbors
(k = 1)

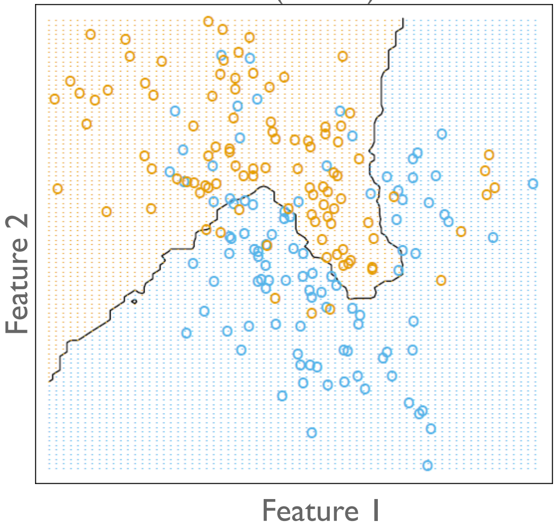


Compares a new (previously unseen) test observation to (previously seen) training observation(s)

What do we need?

- Training observations
- Distance metric
- Decision rule
- Decision statistic
- k (= # of nearest neighbors to consider)

K-Nearest Neighbors
(k = 15)



Assigns a class (according to a decision rule) consistent with the observation(s) closest to (most similar to) the new observation

KNN Decision Statistic

Assign "class-of-interest" (H_1 class)
e.g., "red class"

Assign indicator variables
e.g., $I_{red} = 1$ and $I_{black} = 0$

Identify nearest k neighbors

indicator variable (0 or 1)

$$\lambda(x) = \frac{1}{k}\sum \textcircled{I} \text{ for k nearest points}$$

$$= \frac{\text{\# red data points in k nearest}}{k}$$

$$\lambda(x) \in \left\{0, \frac{1}{k}, \frac{2}{k}, \ldots, \frac{k}{k} = 1\right\}$$

$\Rightarrow$ k+1 possible decision statistics

k = 3

$$\lambda(x_1) = \frac{\text{\# red}}{k} = \boxed{\frac{1}{3}} \Rightarrow \text{"black"}$$

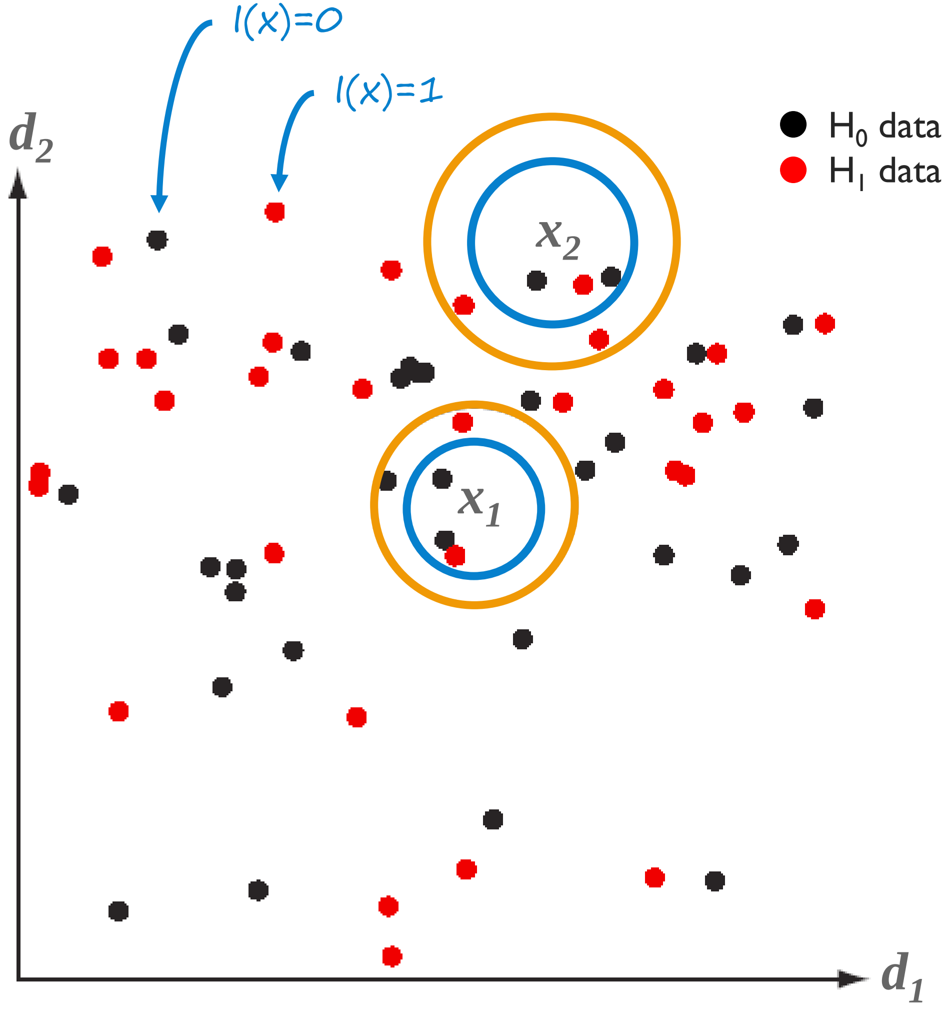
$$\lambda(x_2) = \frac{\text{\# red}}{k} = \boxed{\frac{1}{3}} \Rightarrow \text{"black"}$$

k = 5

$$\lambda(x_1) = \frac{\text{\# red}}{k} = \boxed{\frac{2}{5}} \Rightarrow \text{"black"}$$

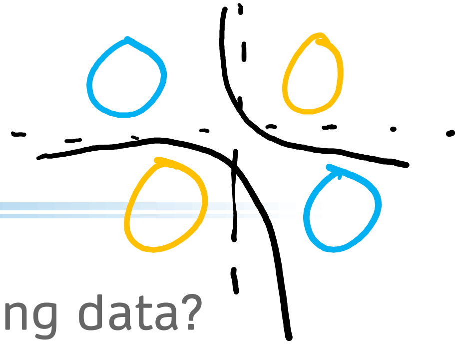
$$\lambda(x_2) = \frac{\text{\# red}}{k} = \boxed{\frac{3}{5}} \Rightarrow \underline{\underline{\text{"red"}}}$$

$\beta = \frac{1}{2}$

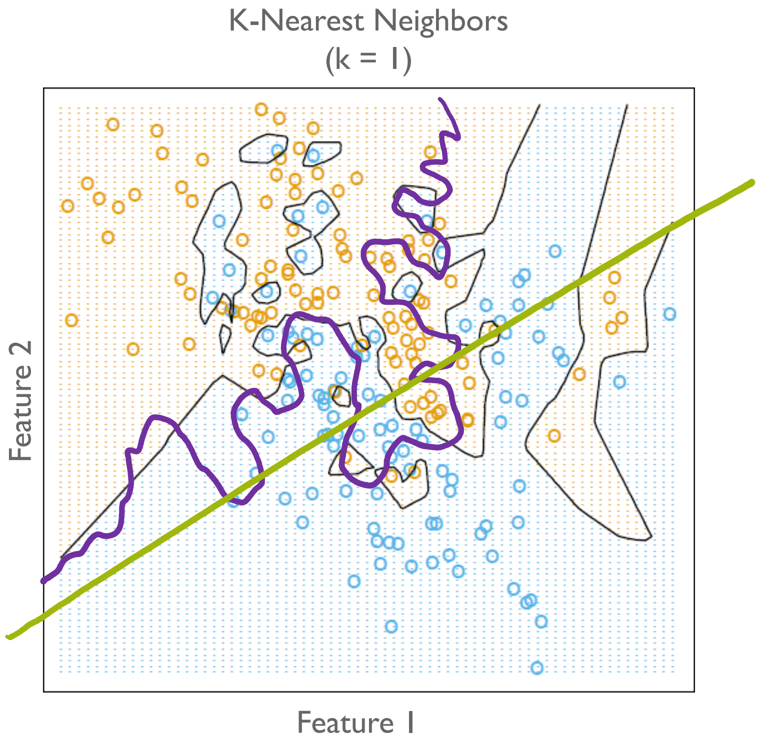


Decision? Compare decision statistic λ to threshold β: $\lambda(x) \geq \beta \rightarrow$ decide H_1 or $\lambda(x) < \beta \rightarrow$ decide H_0
"majority vote" (β=½) is very common, but not the only option

Classifier Flexibility



To what extent does the classifier have the ability to conform to the training data?



What might a decision boundary for new data look like?

What might a decision boundary for an inflexible classifier look like?

Model Complexity (Flexibility)
→ great for training (modeling the data we have)

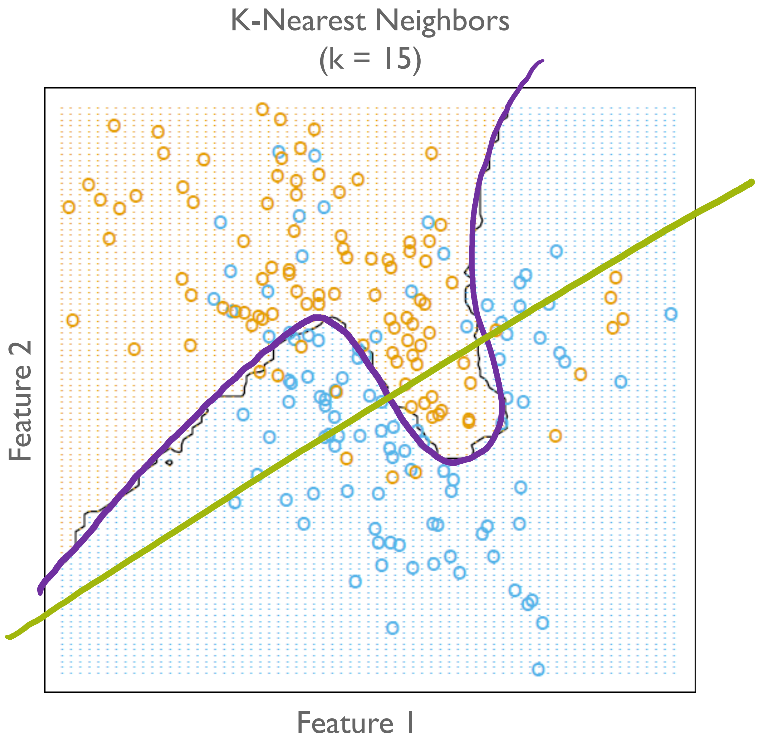
What about new data?
Maybe not so good… overfitting may be a problem

High flexibility/complexity → High variance in model

Classifier Stability

To what extent does the classifier depend on individual training observations?

(or, more generally, the specific training data we have)

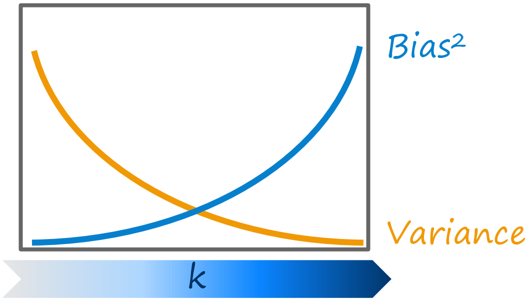


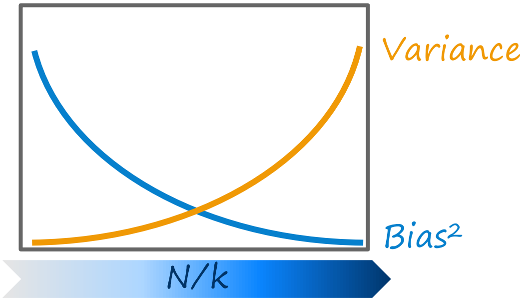
What might a decision boundary for new data look like?

What might a decision boundary for the simplest classifier look like?

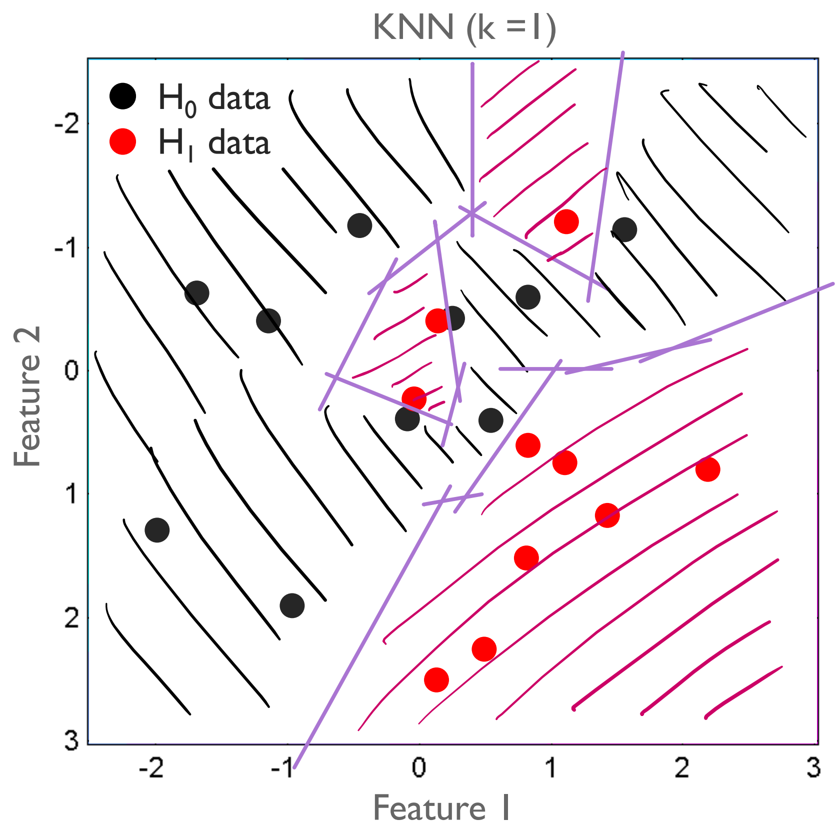
Detail in data not captured by the model
→ Systematic error because model is too simple

High simplicity → High bias (systematic error) in model





KNN Visualization & Performance Evaluation

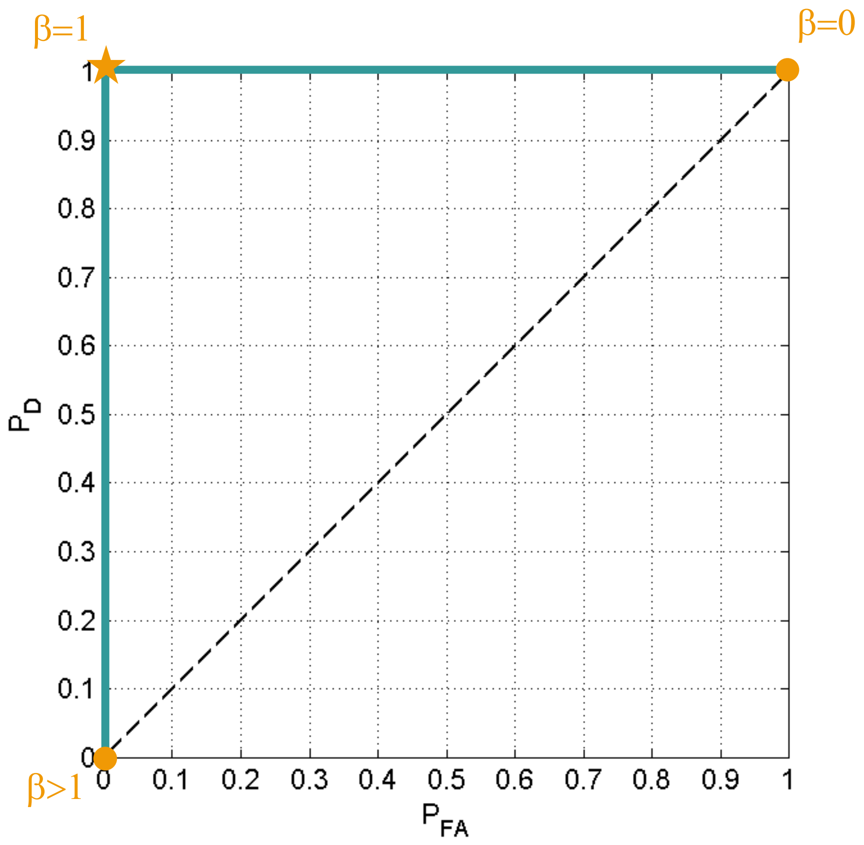


(Assume this (hand-drawn) representation **is** the set of boundaries between "red" and "black" from the Voronoi tessellation)

With no cross-validation, the testing observations are the training observations

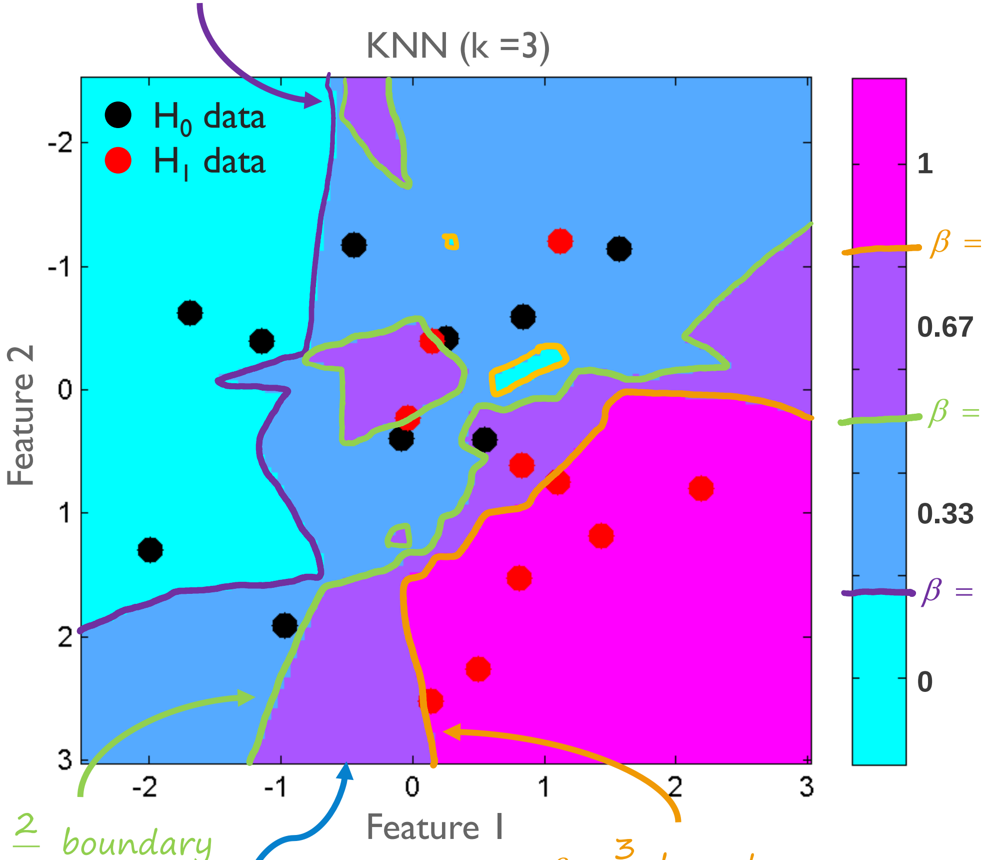
→ every observation's single nearest neighbor is itself !!!

$(P_D, P_{FA}) = (0,1)$ → perfect !



KNN Visualization & Performance Evaluation

$\beta = \frac{1}{3}$ boundary



$\beta = \frac{3}{3}$

$\beta = \frac{2}{3}$ majority vote decision boundary

$\beta = \frac{1}{3}$

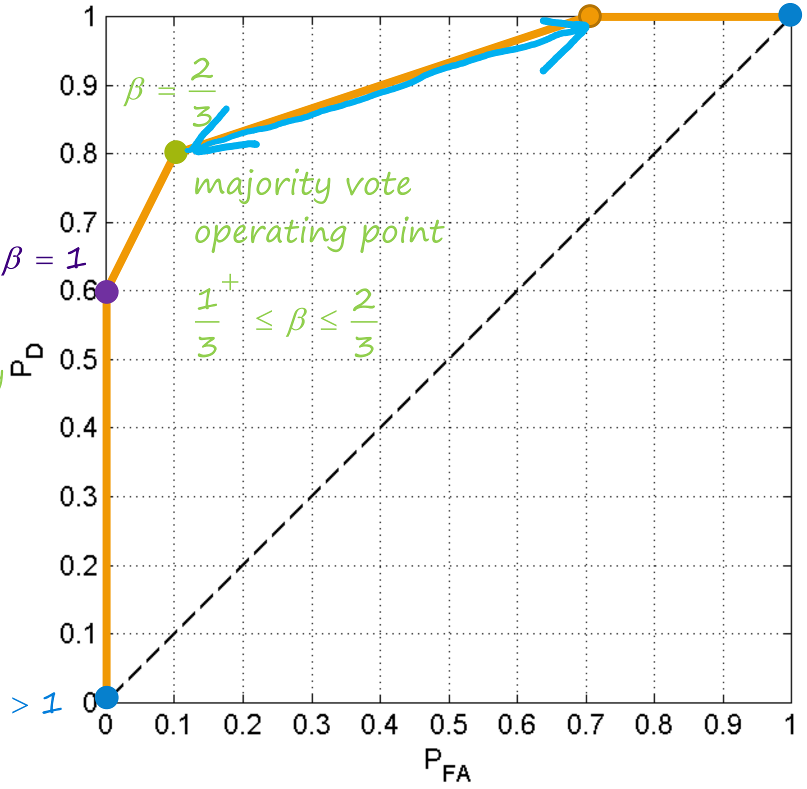
$\beta = \frac{2}{3}$ boundary

$\beta = \frac{3}{3}$ boundary

color indicates $\dfrac{\#\ \text{red observations in 3 nearest}}{3}$

$\beta = \frac{1}{3}$

$\beta = 0$



$\beta = \frac{2}{3}$

majority vote operating point

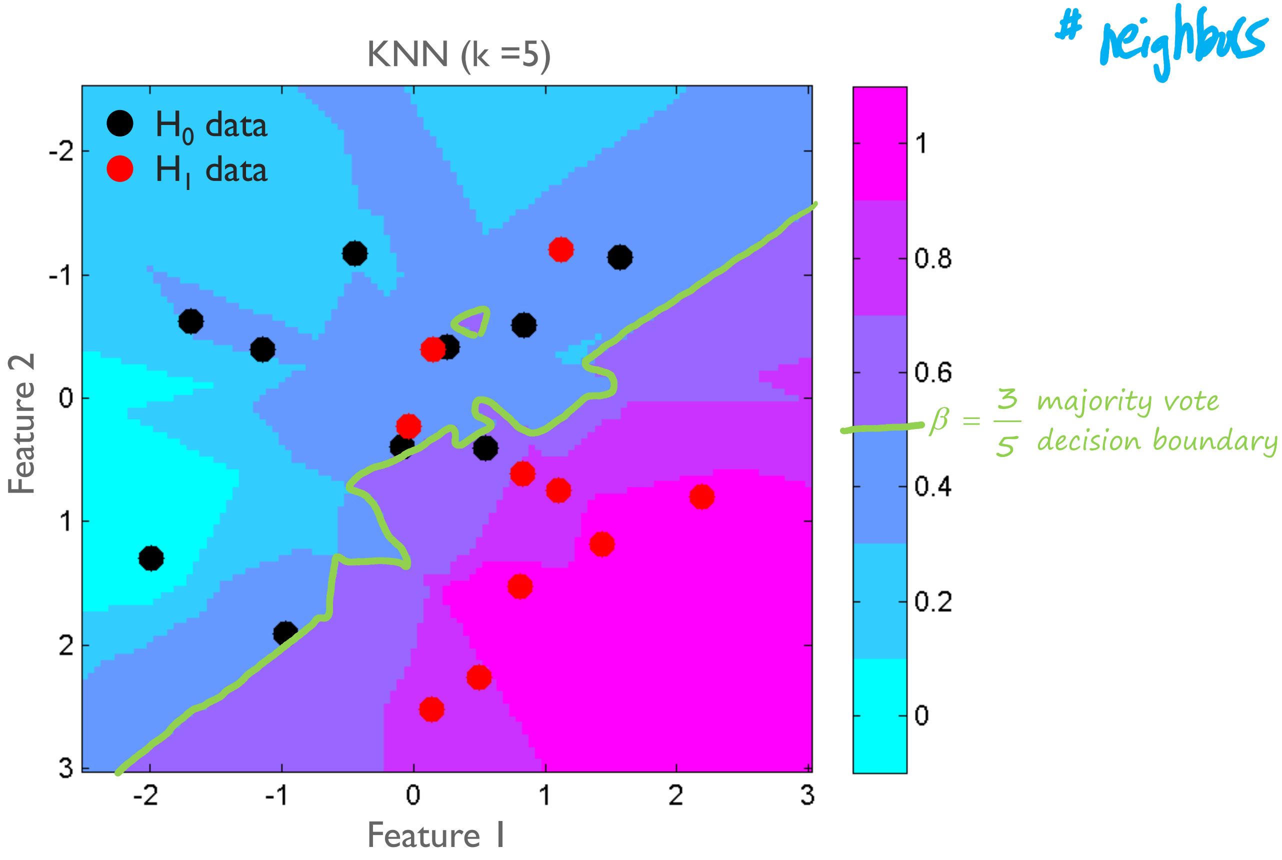
$\frac{1}{3}^{+} \leq \beta \leq \frac{2}{3}$

$\beta = 1$

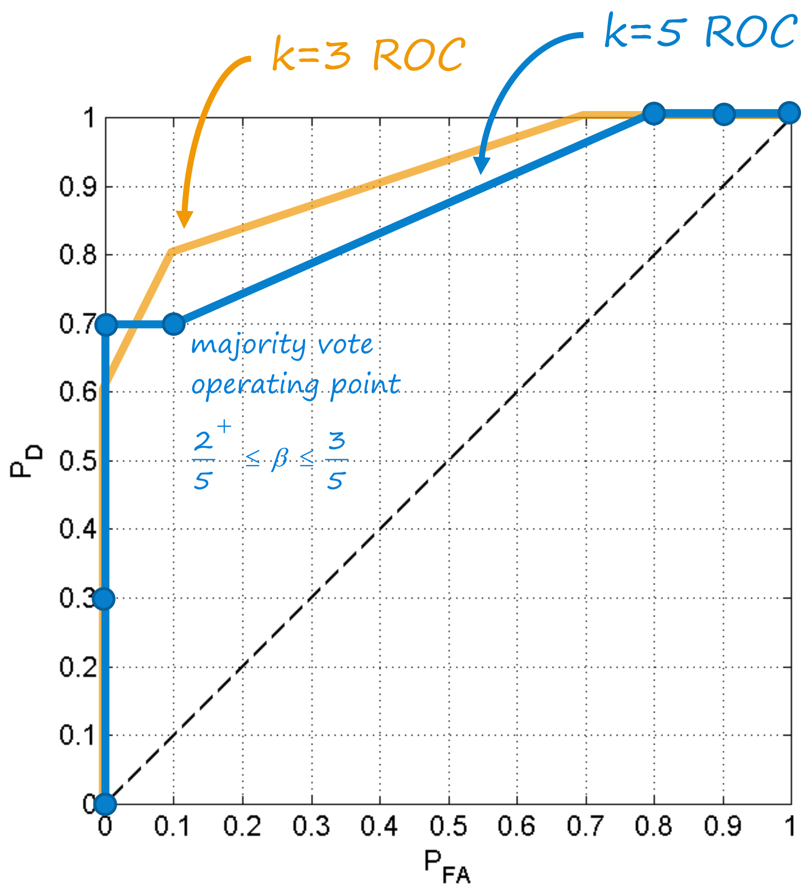
$\beta > 1$

KNN Visualization & Performance Evaluation

$$\frac{\#\ H_1 \text{ neighbors}}{\#\ \text{neighbors}}$$



$\beta = \frac{3}{5}$ majority vote decision boundary



majority vote operating point

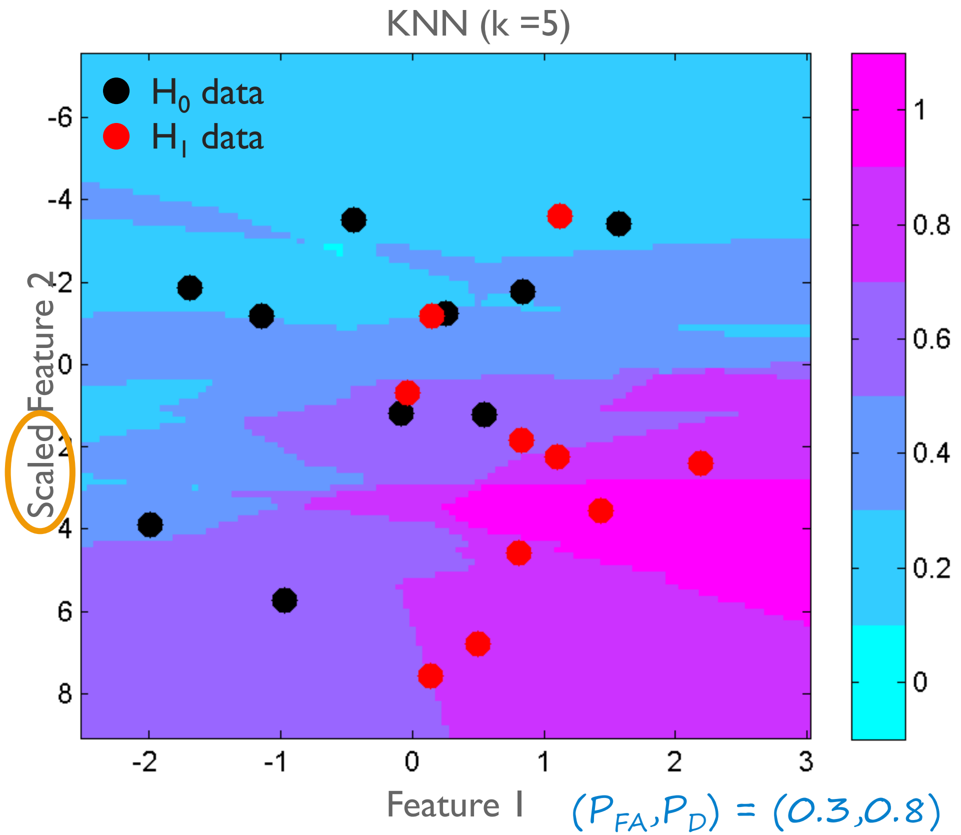
$\frac{2^+}{5} \leq \beta \leq \frac{3}{5}$

KNN decision statistic is a coarse local approximation to p(x is from class 1)

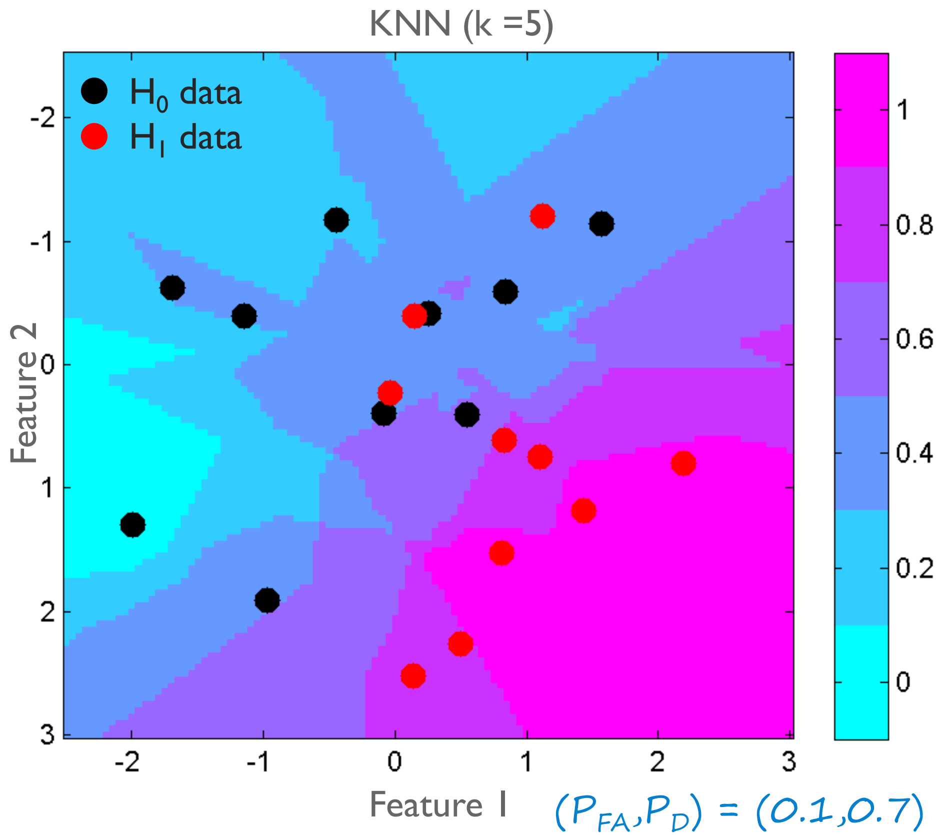
KNN Visualization & Performance Evaluation

Assuming majority vote decision rule (β=½)
Which is feature representation is better? It depends !!
→ Need performance criteria! (and full ROCs)

Scaling
Distance metric } Can matter a lot!



$(P_{FA},P_D) = (0.3,0.8)$
$P_{CD} = 0.75$



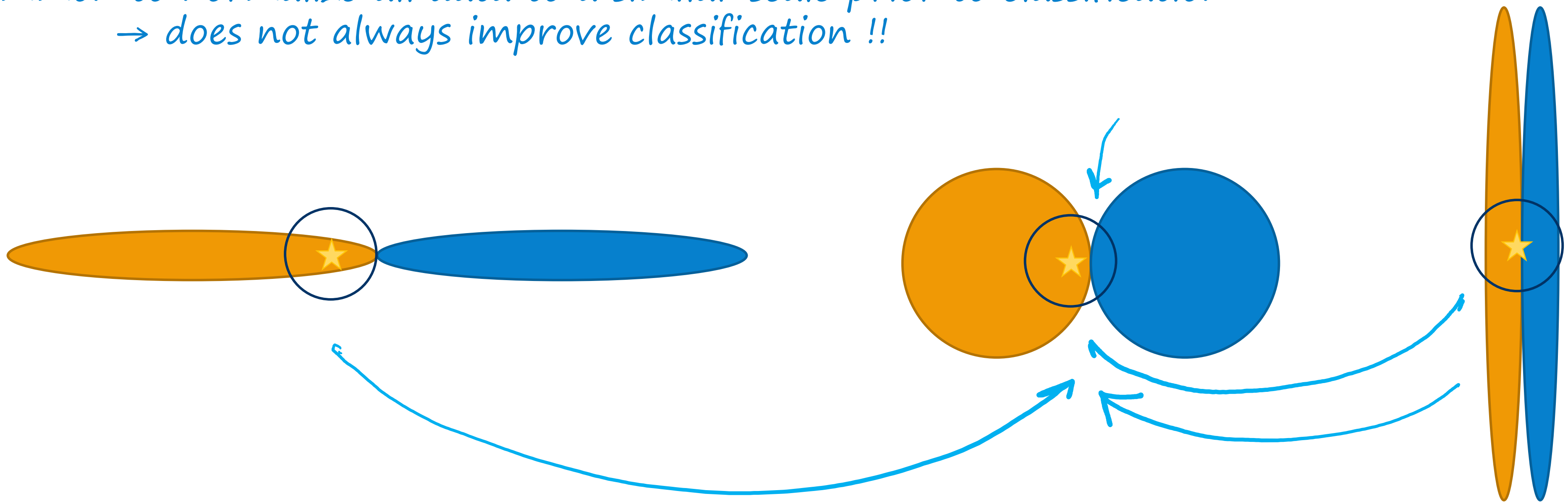
$(P_{FA},P_D) = (0.1,0.7)$
$P_{CD} = 0.8$

Training Data

Scaling (often) matters! → *Can alter decision statistic, and final decision*

Recommend leveraging domain knowledge to influence scaling choices
Should a feature contribute more, to be more heavily weighted?

Common to normalize all data to a similar scale prior to classification
→ does not always improve classification !!



★ *Note how training data are normalized, and apply the same process to the test data*

Data Scaling (Normalization)

- ★ Autoscaling (z-Scoring)

  a.k.a ZmUv (zero-mean, unit-variance)

$$x'_d = \frac{x_d - \mu_d}{\sigma_d} : -\infty \le x'_d \le \infty$$

x_d = original d^{th} feature
μ_d = mean of original d^{th} feature
σ_d = standard deviation of original d^{th} feature
x'_d = scaled d^{th} feature

- Maximum scaling ⚠

$$x'_d = \frac{x_d}{U_d} : \frac{L_d}{U_d} \le x'_d \le 1$$

U_d = max(all x_d's)
L_d = min(all x_d's)

[implicitly assumes $x_d \geq 0$]

What if $U_d < 0$?

$$x'_d = \frac{x_d}{M_d} : -1 \le x'_d \le 1$$

$M_d = \max\left(\left|\text{all } x_d\text{'s}\right|\right)$

- Range scaling

$$x'_d = \frac{x_d - L_d}{U_d - L_d} : 0 \le x'_d \le 1$$

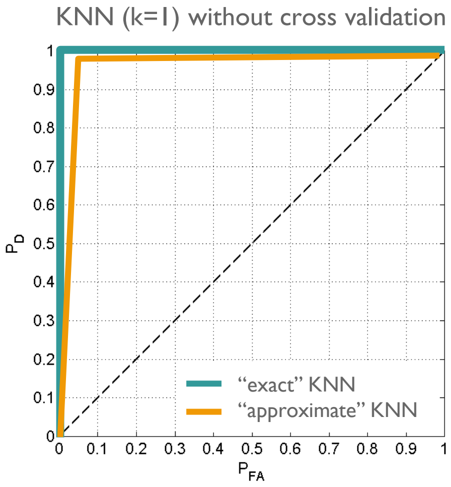
L_d = min(all x_d's)
U_d = max(all x_d's)

Computational Considerations

kNN training: *No learning of model parameters*
"Training" consists of storing the training observations
Hyperparameter k selected through cross-validation

kNN testing: *Calculate distance between x_{test} and all training observations to find k nearest neighbors*
→ Computational load is in testing, not training

("exact" KNN)


KNN (k=1) without cross validation

How to ease computation ("approximate" KNN)?

Generally, 2 strategies:

1) *Store only training observations that are near/define decision boundary*
   → requires choosing the decision rule ahead of time
2) *Strategically search over training observations, perhaps accepting the "almost nearest" neighbors*

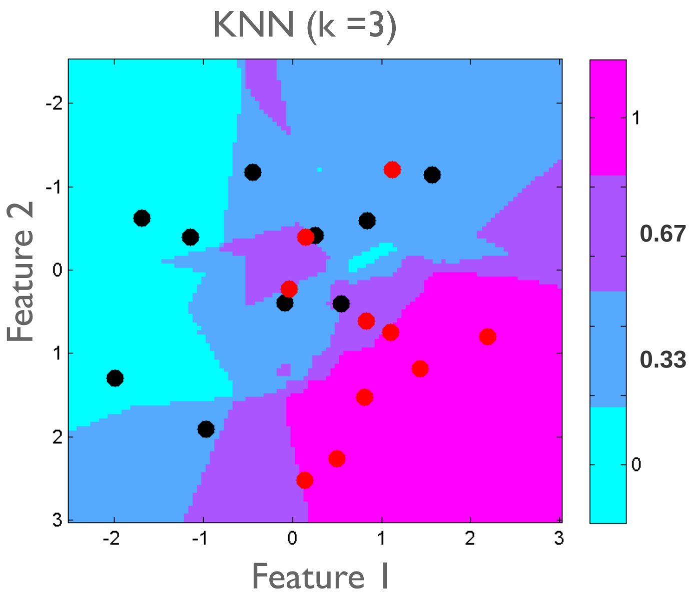
Classifiers: Visualizing Decision Statistic Surfaces

Hypothesize a grid of test data

Calculate the decision statistic at each grid point

- by running the classifier with the list of hypothesized test data as the test data for the classifier

Image (`imagesc`) the grid of decision statistics (the decision statistic surface)



Using only information stored in the classifier structure:

```
x1Range = max(xTrain(:,1))- min(xTrain(:,1));
x2Range = max(xTrain(:,2))- min(xTrain(:,2));
x1 = linspace(min(xTrain(:,1))-0.2*x1Range,
        max(xTrain(:,1))+0.2*x1Range,251);
x2 = linspace(min(xTrain(:,2))-0.2*x2Range,
        max(xTrain(:,2))+0.2*x2Range,251);

% Create the grid of test data points
[xTest1,xTest2] = meshgrid(x1,x2);

% Each column is a feature, each row an observation
xTest = [xTest1(:) xTest2(:)];

% Run the classifier with these test data
dsTest = runClassifier(classifierStructure,xTest);

% dsTest is a vector, reshape it to a matrix
dsTest = reshape(dsTest,length(x2),length(x1));

% Image the decision statistic surface
imagesc(x1([1 end],x2([1 end]),dsTest)

% Add the training data points to the surface
hold on
% H0
plot(xTrain(truth==0,1),xTrain(truth==0,2),’ko’)
% H1
plot(xTrain(truth==1,1),xTrain(truth==1,2),’ro’)
```