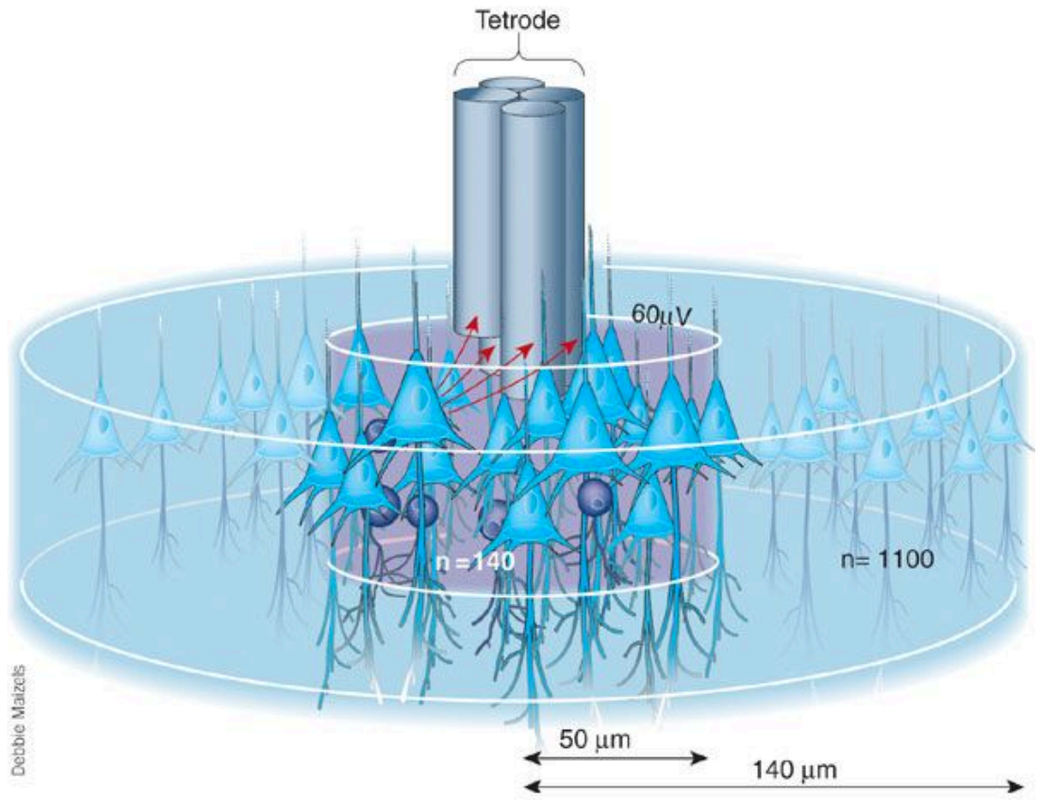
CLASSIFICATION

11.20.2020

SPIKE SORTING

* suppose that you've recorded thousands of spikes from neurons using a tetrode

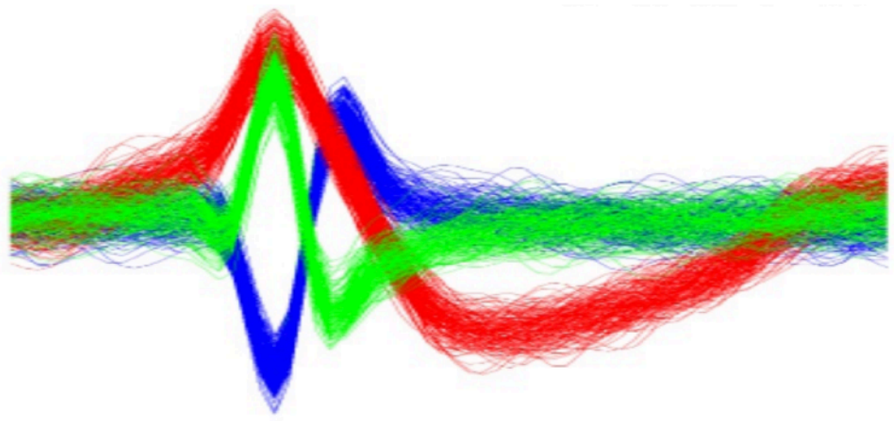
* now you need to figure out which neuron each spike came from based on the recordings from the 4 electrodes



SPIKE SORTING

* you can start by hand-tagging some of the spikes as Neuron A or Neuron B

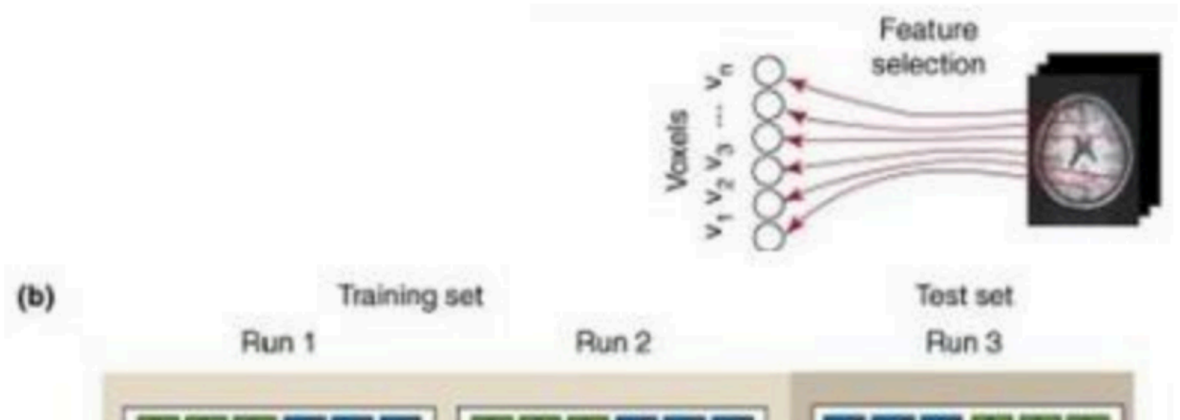
* then you want to automatically *classify* the rest into one of your existing categories



FMRI DECODING

* suppose that you've done an fMRI experiment where people viewed images of SHOES and BOTTLES

* now you want to use the pattern of fMRI responses to guess whether the stimulus was SHOES or BOTTLES

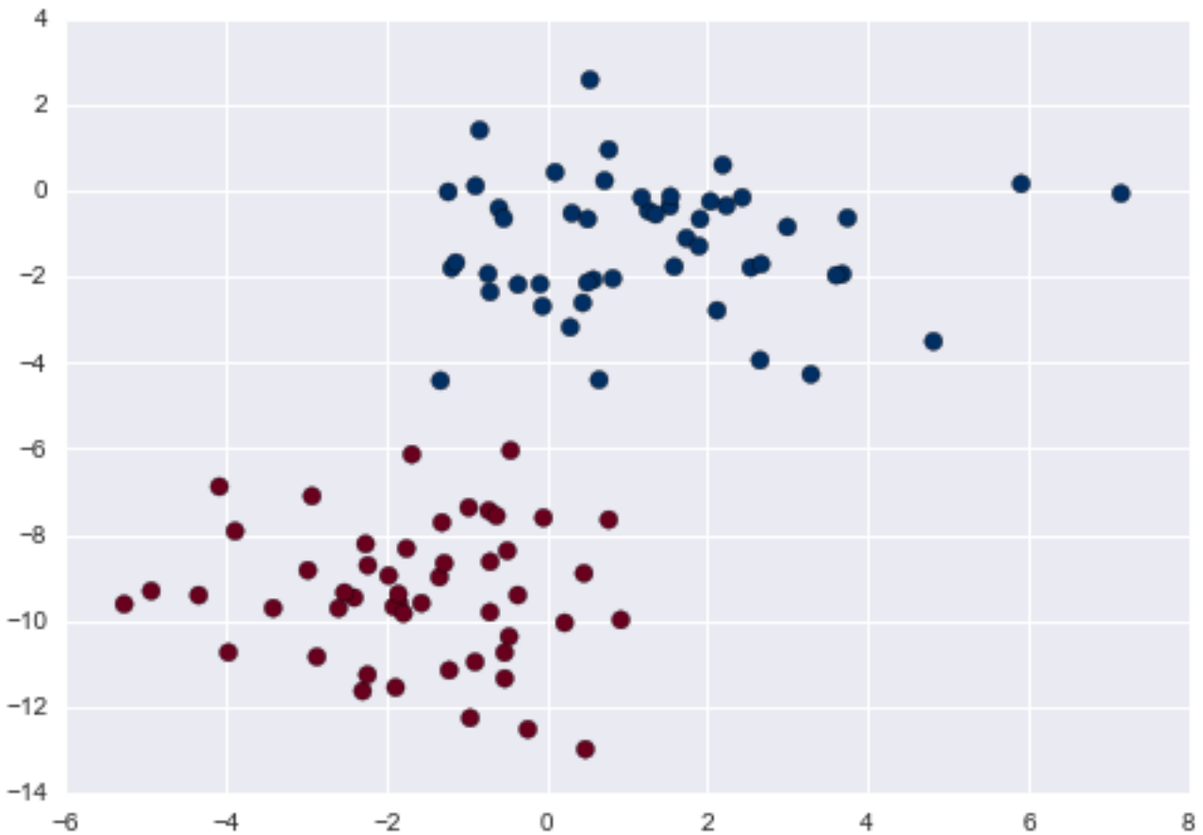


FMRI DECODING

* or! your subjects watch videos, and then you guess which objects or actions were present in each video clip based on the responses

* you want to *classify* each fMRI response into, e.g., “yes there was a face” or “no there was not a face”

CLASSIFIERS



* a classifier is a function that guesses the “class” of a datapoint based on its features

* here “class” is **red** or **blue** and each point has 2 features (x and y axes)

CLASSIFIERS

* there are *MANY* different types of classifiers

* but they can broadly be divided into two families:

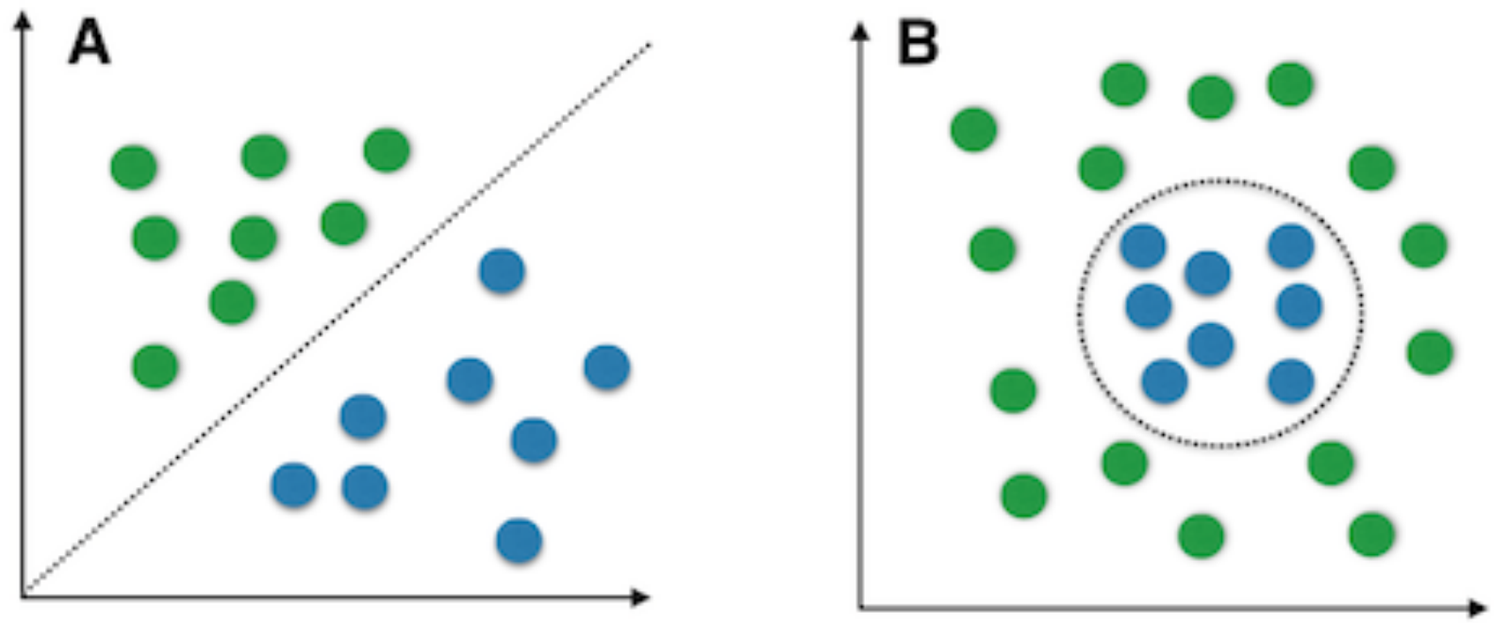
  * linear classifiers

  * non-linear classifiers

CLASSIFIERS

* **linear classifiers** (**A**) separate the classes by using a straight line (or plane, or hyperplane) as the “decision boundary”

* **nonlinear classifiers** (**B**) can separate the classes using an arbitrarily-shaped decision boundary



CLASSIFIERS

* linear classifiers (e.g. logistic regression, linear SVM) are more appropriate when:
  * you don’t have a lot of data
  * your data are high-dimensional (many features)
  * your data are very noisy

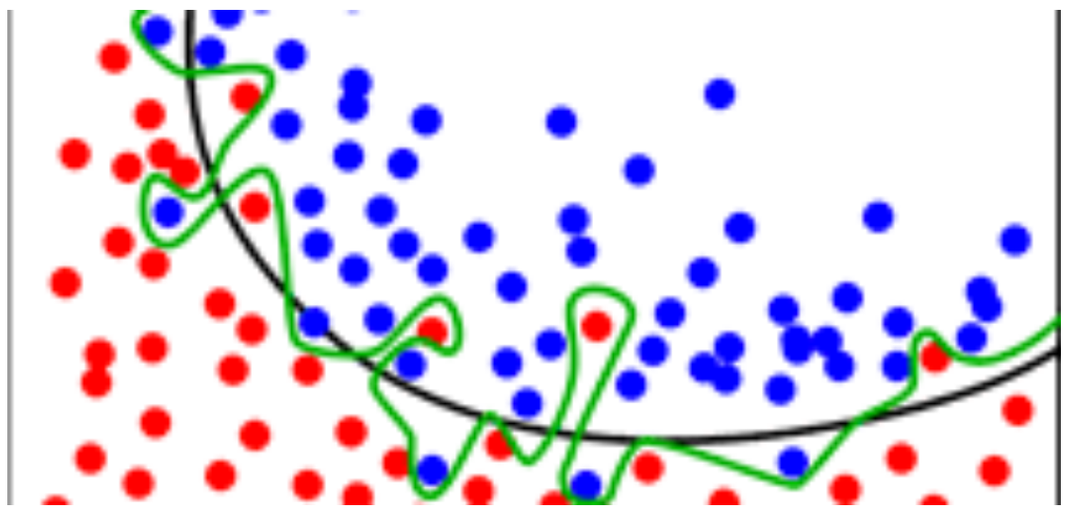
CLASSIFIERS

* nonlinear classifiers (e.g. RBF SVM) are more appropriate when:
  * your data are low-dimensional (few features)
  * your data are not very noisy
  * linear classifiers work poorly :)

TRAINING CLASSIFIERS

* just like regression models, classifiers should be trained on one dataset, and then tested on an *entirely separate* dataset

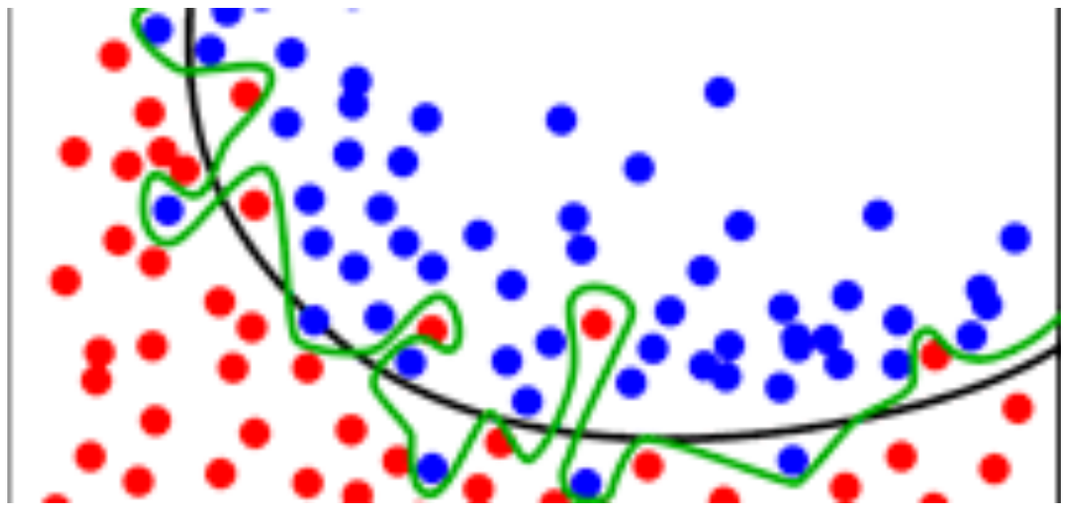
* this controls for *overfitting*



TRAINING CLASSIFIERS

* like regression models, many classifiers can also be *regularized* (see black curve, below)

* this can directly reduce overfitting



EVALUATING CLASSIFIERS

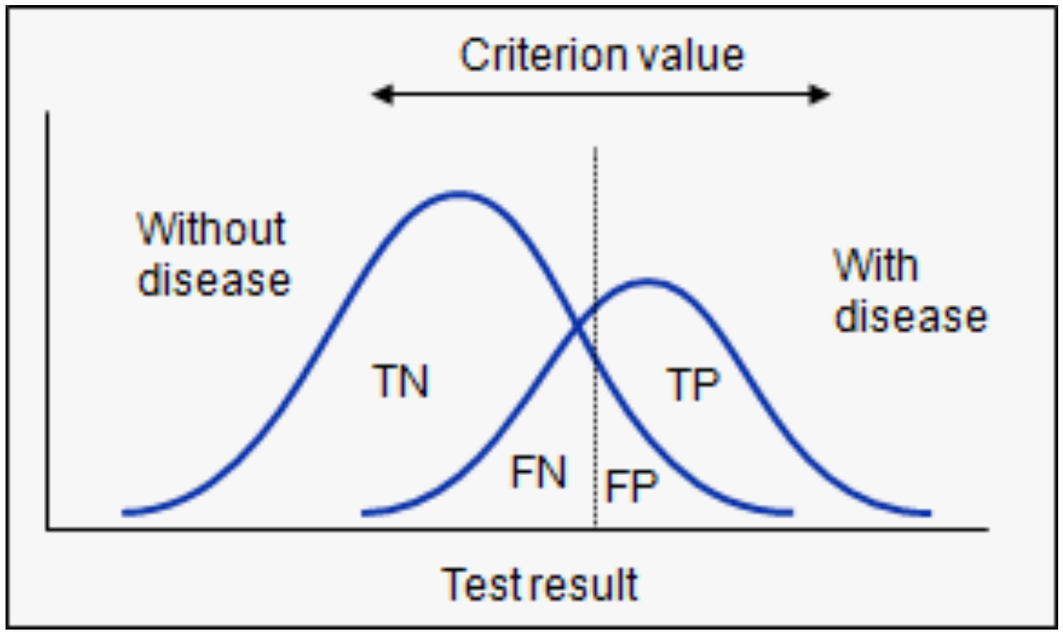
* classifiers can be evaluated by a few metrics

* one is just the **percent correct**, but that doesn’t always tell the whole story

* a more sensitive way to test classifier performance is using **receiver-operating characteristic (ROC) analysis**

ROC ANALYSIS

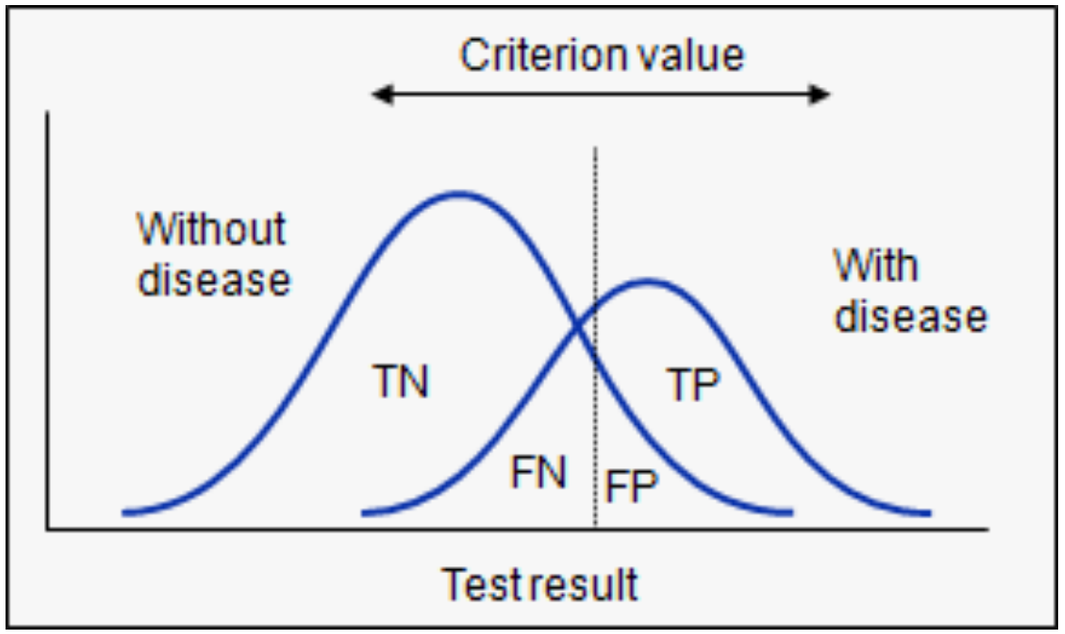
* suppose that your classifier outputs a real number for each datapoint



* (hopefully) your two classes tend to have different values

ROC ANALYSIS

* you turn this real value into classification by setting a “criterion value”

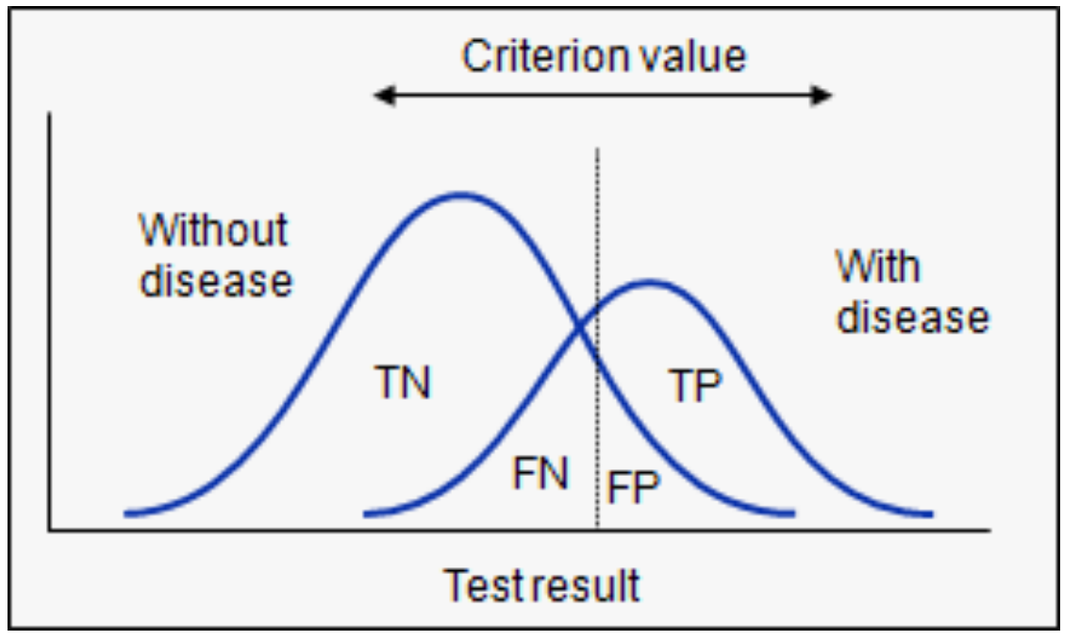


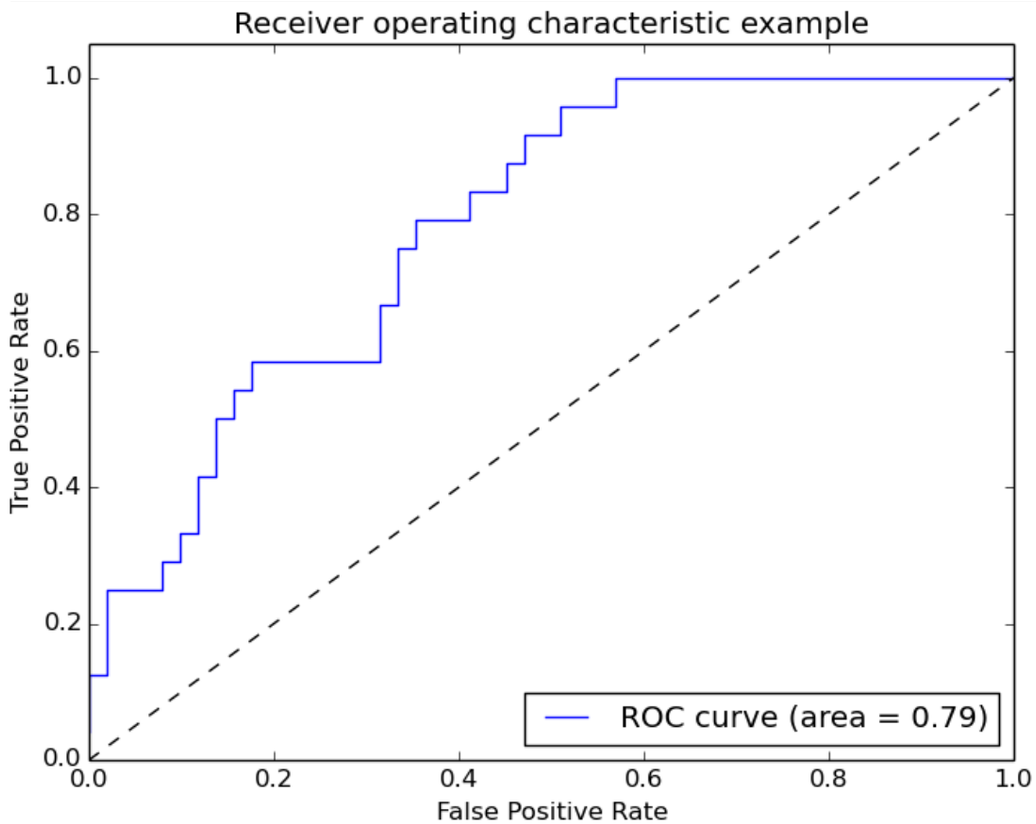
* everything above “criterion” is class A, everything below is class B

ROC ANALYSIS

* in ROC analysis, you check each possible criterion value

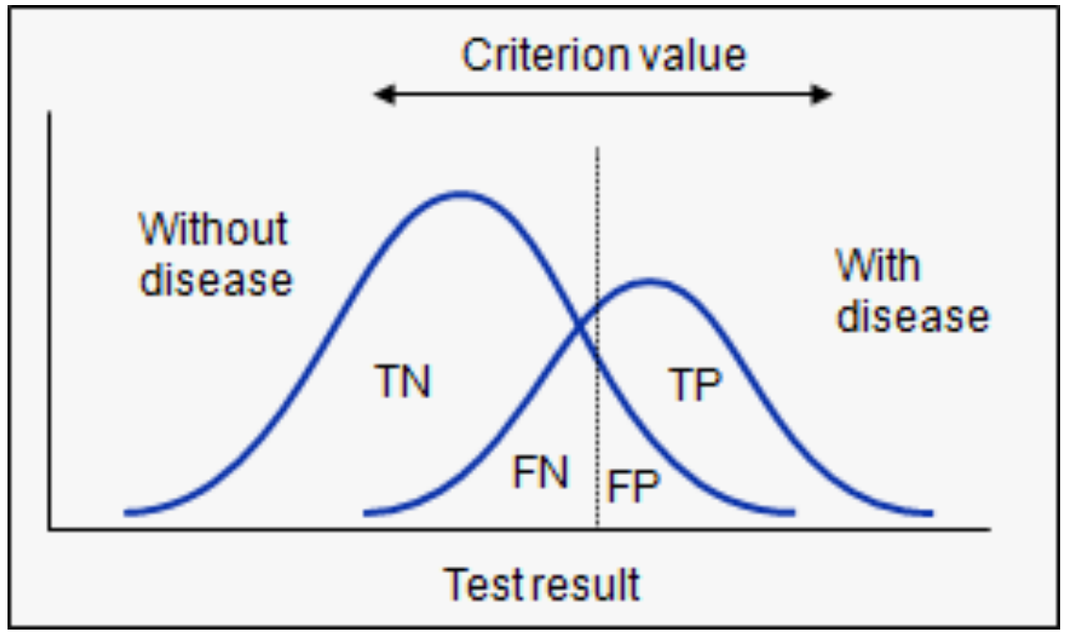
* and compute the true positive (TP) and false positive (FP) rate for each

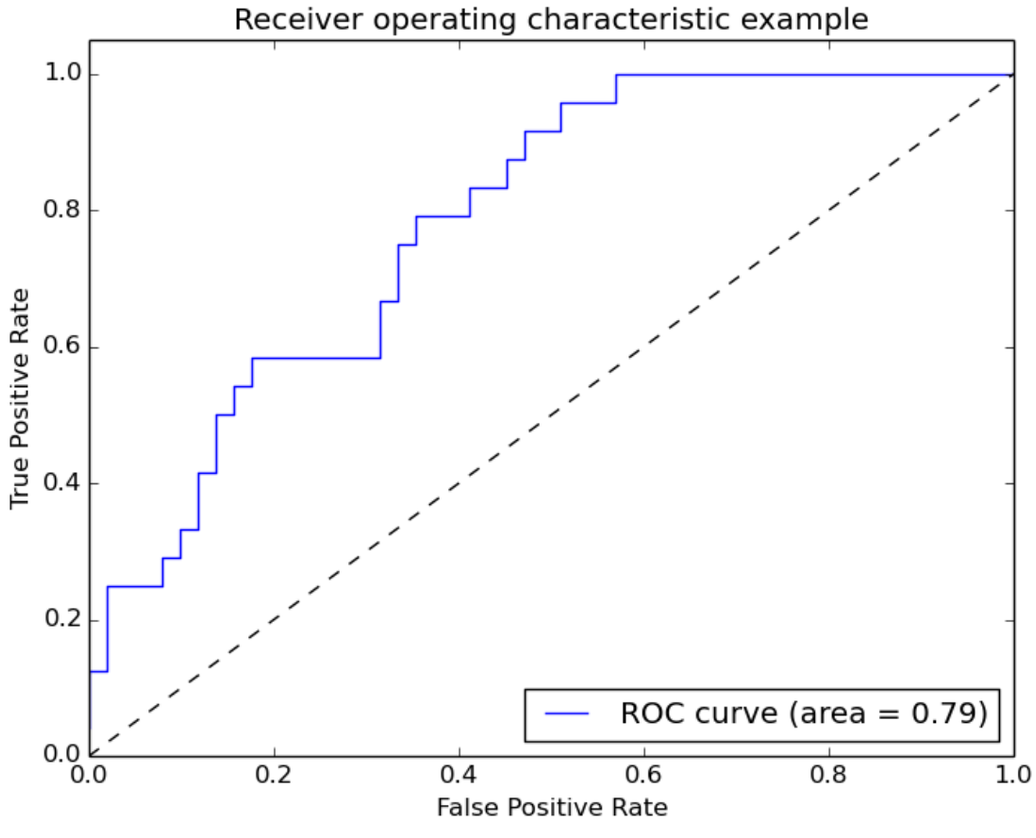




ROC ANALYSIS

* you then plot the TP vs. FP rate to get an ROC curve

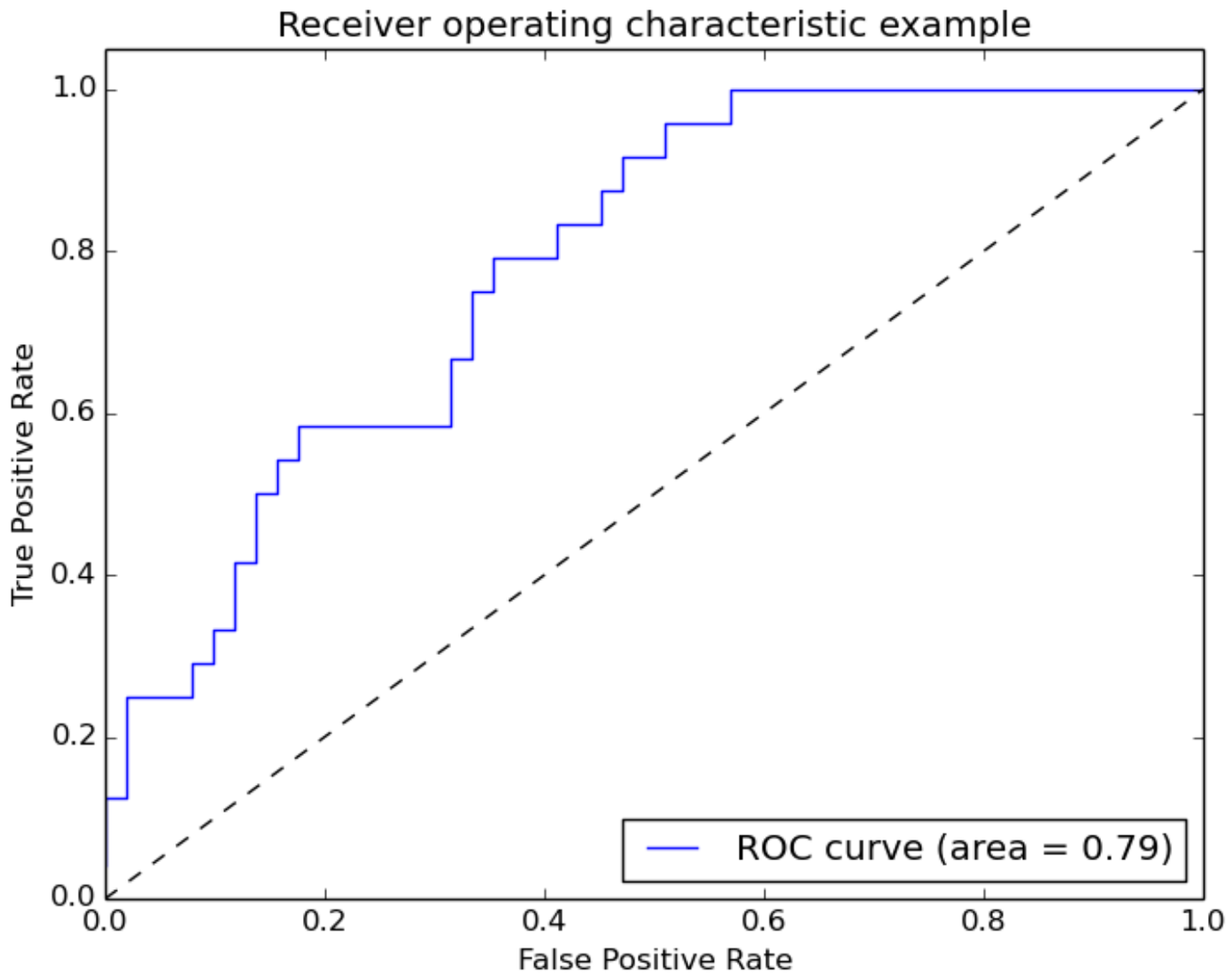




ROC ANALYSIS

* if the classifier is really bad, the ROC should look like a diagonal line

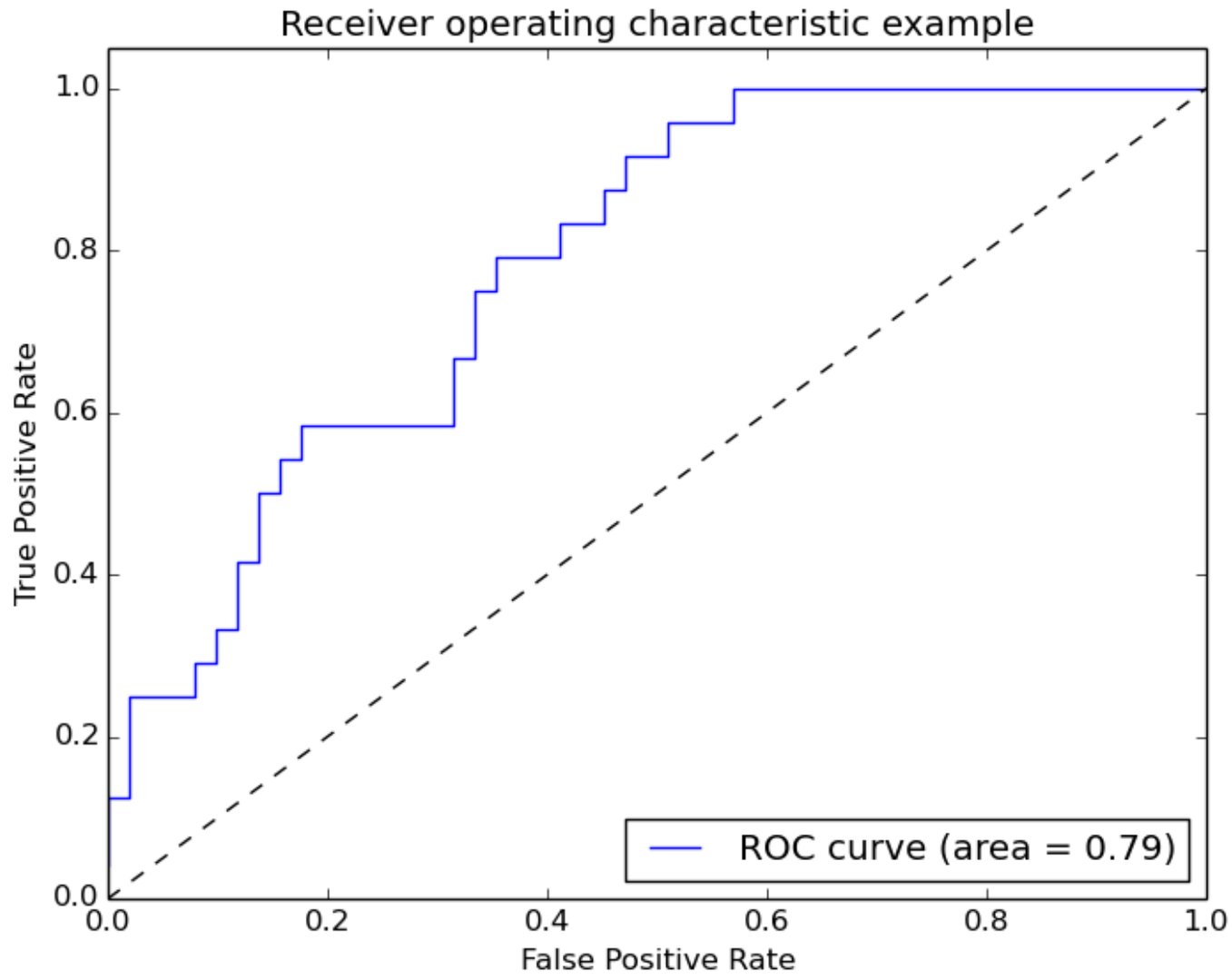
* if it’s really good, the ROC should look like a square



ROC ANALYSIS

* the area under the ROC curve (AUC) is a common metric

* AUC tells you how likely the classifier is to correctly order two random points



NEXT TIME

* on Monday we'll go through some examples of using classification on fMRI data!

THE END