CS[45]783: Machine Learning
Assignment 1: Basic classifiers

Prof. Christopher Crick

1 kD Tree

Design and code your own kD tree data structure as a Python object. Your tree can be less general than a standard implementation; you need only account for two data dimensions, and you need only support 1-nearest-neighbor search. Your class should be structured like the following:

```
class KDTree:

  def __init__(self, matrix):
    # matrix should be a numpy array where each row is a data element
    # and each column is a feature.  This operation should recursively
    # split the data along the median of each feature vector in turn.

  def find_nearest(self, vector):
    # vector is a numpy array with a single row representing a single
    # data element.  This method should traverse the tree to a leaf,
    # then search enough of the tree to guarantee finding the nearest
    # neighbor.  This neighbor vector should be returned.
```

This data structure must work correctly, and you may use it in this assignment. In the future (and on the rest of this assignment, if you wish), you may want to use scipy.spatial.cKDTree instead; it is likely more flexible and efficient.

2 Constructing a simple data set

Draw 5000 data points from each of two different 2D multivariate Gaussian distributions, with means and covariances of your own choosing. The two distributions should be distinct but overlapping, as illustrated in Figure 1. You may use np.random.randn to generate your x_1 and x_2 coordinates independently, but this will limit your distributions to those with diagonal covariance matrices. Alternately, you may use np.random.multivariate_normal to sample from an arbitrarily-covarying 2D distribution. You should end up with a matrix X, which will be 10000×2 in size, and a column vector y of 10000 elements. The ith element of y should be 0 if the ith row of X came from your first distribution, and 1 if it came from the second.

Randomly partition X and y into training and test sets. The easiest way to do this is to construct a 10000-element column vector of booleans and use it as a mask.

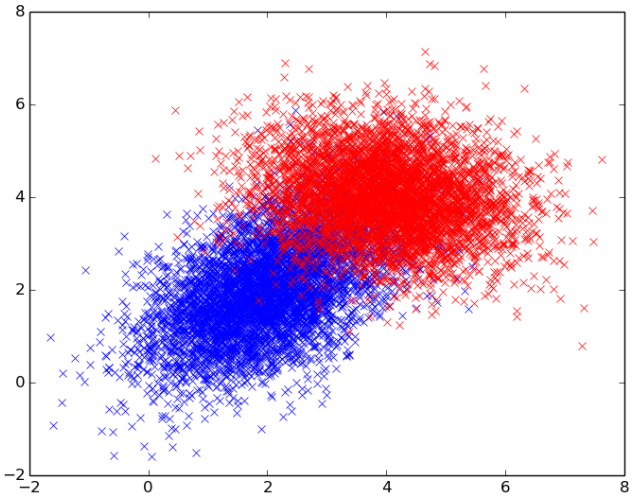

Figure 1: 10000 data points from two Gaussian distributions

3 Linear classifier

Using your training set (which the equation below calls X), construct the maximum-likelihood linear least squares function β to fit your data.

$$\beta = (X^T X)^{-1} X^T y \tag{1}$$

Recall that Numpy multiplication is elementwise by default; use the dot() method for matrix multiplication. Inverses can be computed using Numpy's linalg.inv() method.

You can now use β to classify the elements of your test set. β is itself a regression, of course, but you can use a threshold to turn it into a classifier: $\hat{y} = 0$ if $x^T \beta < 0.5$, 1 otherwise.[1]

Calculate and print the classification accuracy of your algorithm by comparing the true class y and the estimated class $\hat{y}$ for each element of your test set.

Produce a plot that displays all of the following in distinct colors or shapes:

- Training set elements from class 0
- Training set elements from class 1
- Correctly classified test set elements from class 0
- Correctly classified test set elements from class 1
- Incorrectly classified test set elements from class 0
- Incorrectly classified test set elements from class 1

[1] These equations come straight from page 12 of the Hastie text. But if, like me, you think it makes more sense to represent data elements as row vectors rather than column vectors, then you'll want to multiply x and β rather than x^T.

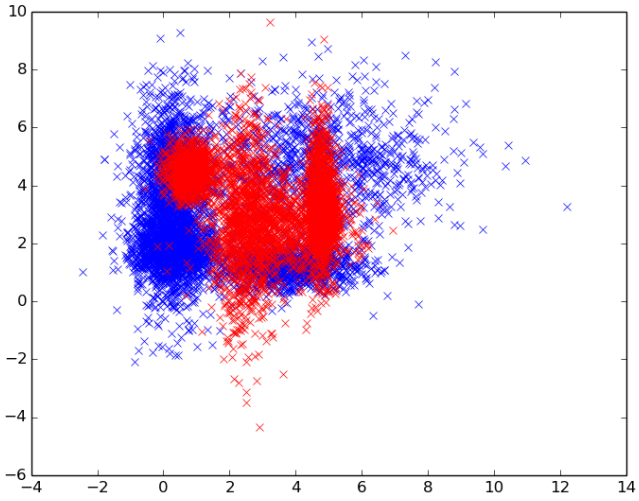

Figure 2: 10000 data points in two classes drawn from ten different Gaussians

4 Nearest neighbors classification

Using the same training and test sets as before, construct a kD tree from your training set. Classify each element of your test set by looking up its nearest neighbor from the training set and assigning $\hat{y}$ to be whatever value y belongs to the nearest training example.

As before, calculate and show the classification accuracy and produce a similar plot.

5 Increasing complexity

Create a new set of training and test data. This time, each classification will be produced by multiple distributions, rather than just one. Draw 1000 samples each from ten different overlapping Gaussian distributions. Five of them should be labeled as class 0 and the others class 1. An example is shown in Figure 2. Perform the same linear and nearest neighbor classification processes, calculate the classification accuracy and plot the results.