Covariance and Sample Covariance

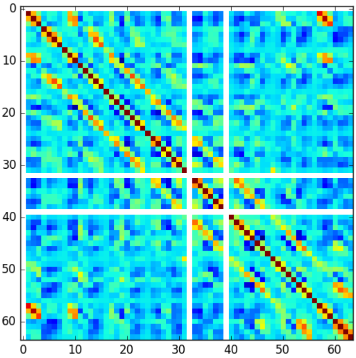
- Many simple data analysis problems involve finding the eigenvectors of the sample covariance matrix given the data
- Test for relationships between variables/features
- PCA: Gives a natural basis in which to express the data

$$\mathbf{R} = E[\mathbf{x}^T \mathbf{x}] \tag{1}$$

$$R = \frac{1}{n-1} \sum_{i=1}^{N} (x_i - \mu)^T (x_i - \mu) \tag{2}$$

$$\frac{1}{n-1} \mathbf{X}^T \mathbf{X} \tag{3}$$

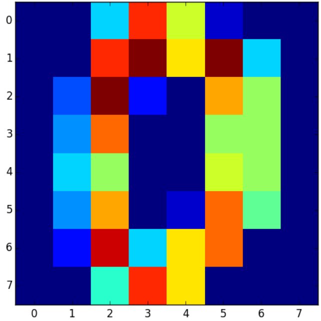
Visualizing the Correlation or Covariance



- Rendering the Correlation as an image immediately reveals which variables are related

Working with the zipcode “digits” dataset

- Dataset of 8×8 images of hand-written characters



Linear Discriminant Analysis

- Linear Discriminant Analysis (LDA) is a supervised learning technique that attempts to provide a one-dimensional projection of the data such that discrimination between classes is maximized
- As in previous examples $\{x_1, x_2, \ldots, x_n\}$ we have a set of observations where each observation belongs to one of a finite number of classes

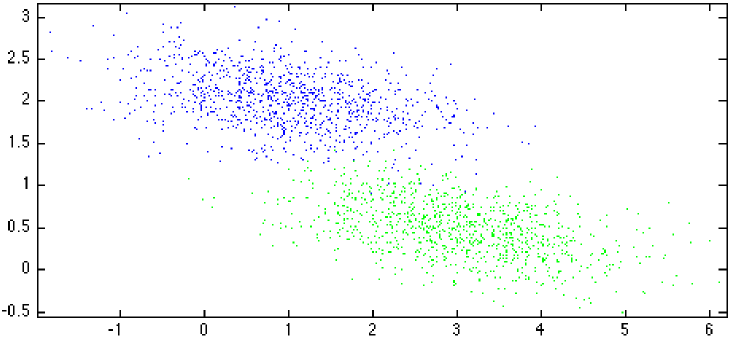
LDA

- Attempt to find some *linear* function of the values of feature values such that we can use the output to classify new data instances
- Which coefficients w maximize separability?

$$\phi = w^T x \tag{4}$$

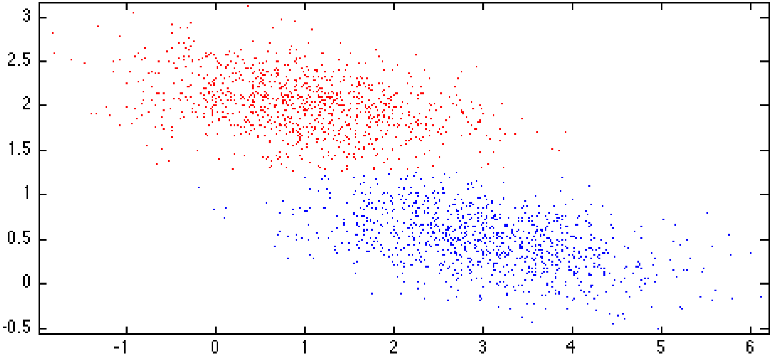
Example

- In this case we can do a decent job of separating the two classes with a single hyperplane



Example

- Our labeled data might clearly fall into some distinct groupings



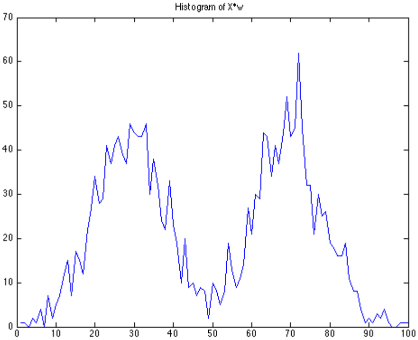
Performance Metrics

- In any binary classification task two types of error can occur, false positives and false negatives
- In general we need to keep track of both of these errors to understand how well our classifier is performing
- The *Receiver Operating Characteristic* (ROC) keeps track of both of these error rates as we vary the threshold

$$\text{threshold} > w^T x \tag{5}$$

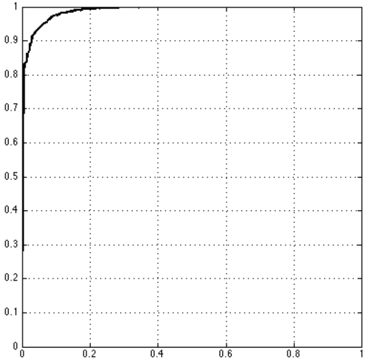
Receiver Operating Characteristic

- Imagine we look at a histogram of our test statistic
- Our threshold will fall somewhere along the x-axis and will determine the two error rates



Example

- Each point along this curve corresponds to a single value of the threshold parameter



Measure of Separation

- Again we are looking to maximize some objective function over the space of all possible models.
- Define the mean vectors in each class

$$\mu_i = \frac{1}{N_i} \sum_{x \in C_i} x \tag{6}$$

- One choice is to find the projection that maximizes the distance between the class means

$$J(w) = |w^T(\mu_1 - \mu_2)| \tag{7}$$

Fisher LDA

- Normalize the difference between the means by a measure of within-class scatter

$$S_i = \sum_{x \in C_i} (x - \mu_i)(x - \mu_i)^T \tag{8}$$

$$S_W = S_1 + S_2 \tag{9}$$

- Fisher Criterion

$$J(w) = \frac{|w^T(\mu_1 - \mu_2)|}{w^T S_W w} \tag{10}$$

Fisher LDA (cont.)

- The Fisher criterion projects the data in such a way that instances from the same class are close to one another and and the projected means are as far apart as possible
- We can also reexpress the difference between projected means in terms of w

$$w^T(\mu_1 - \mu_2)(\mu_1 - \mu_2)^T w = w^T S_B w \tag{11}$$

$$J(w) = \frac{w^T S_B w}{w^T S_W w} \tag{12}$$

LDA recipe

- The LDA solution can be found explicitly in terms of the data by differentiating the objective with respect to w and setting the result equal to 0

$$\frac{d}{dw} J(w) = 0 \tag{13}$$

- When we work this out (homework) the solution we arrive at is equal to the following

$$\hat{w} = S_W^{-1}(\mu_1 - \mu_2) \tag{14}$$

Defining a Symmetry Feature

```
def symlr(t):
    s = np.fliplr(t)
    y = t - s
    val = np.sum(y[:, 0:3])
    return val


def symud(t):
    s = np.flipud(t)
    y = t - s
    val = np.sum(y[0:3, :])
    return val
```

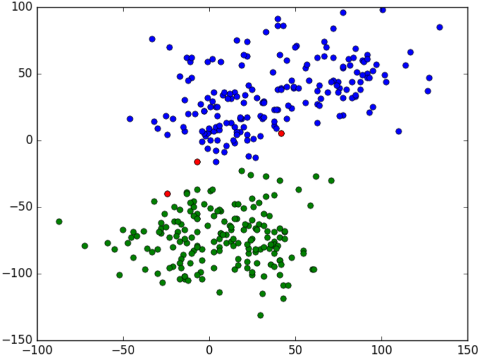
- The above functions calculate a symmetry measure on our 8 by 8 images

Working with the Digits Dataset

```
clf = LinearDiscriminantAnalysis()

clf.fit(X, y)

err = clf.predict(X) != y

for i in [ii, jj]:
    II = digits.target == i
    plt.plot(B[II, 0], B[II, 1], 'o', label=str(i))

plt.plot(X[err, 0], X[err, 1], 'ro')

f = 1
h = scipy.stats.kruskal(B[digits.target == ii, f],
                        B[digits.target == jj, f])
print(h)

plt.figure()
plt.boxplot([B[digits.target == ii, f],
             B[digits.target == jj, f]])
```

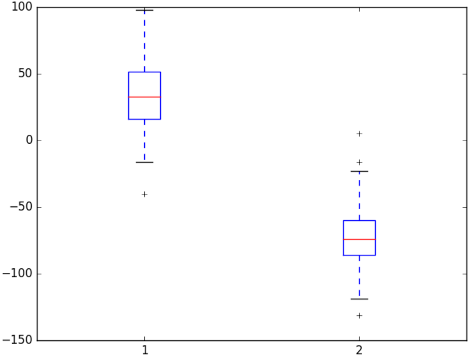
- Defining a classification object and fitting it to our data

Symmetry features for digits 5 and 6



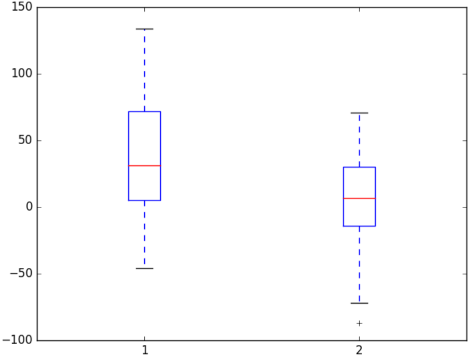
- Are both features equally discriminative?

Discriminative Feature



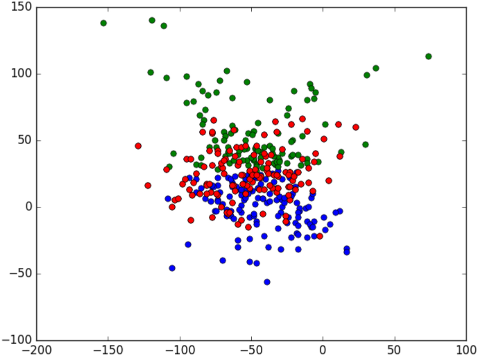
- Box-Plot Shows clear difference between the two classes

Non-discriminative Feature



- Still significant difference, but classes are overlapping

Same result for digits 3 and 9



- Less discriminant power in these features for a different digits