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?

$$\phi = w^T x \tag{1}$$

LDA recipe

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

$$\hat{w} = S_W^{-1}(\mu_1 - \mu_2) \tag{2}$$

- In Python we can perform LDA quickly by defining a new `LinearDiscriminantAnalysis()` object

Loading Dataset with Pandas

```
tt = pd.read_csv("data_lda_circular.txt", header=None)

X = tt.values[:, 0:2]
y = tt.values[:, 2]

indx = (y == 1) | (y == 2)

X = X[indx, :]
y = y[indx]
```

- Additional code does not effect the particular dataset we are working with

Leveraging Polymorphism

```
clf = neighbors.KNeighborsClassifier(1, weights='uniform')
... <or>
clf = LinearDiscriminantAnalysis()
```

- Many of the scikit implementations of machine learning algorithms can be swapped out quickly using polymorphism
- Can construct the `clf` object here in multiple ways

Performing CV Training Test Split

```
X_train, X_test, y_train, y_test = <...>
  cross_validation.train_test_split(X,
  y,
  test_size=0.3,
  random_state=0)

clf.fit(X_train, y_train)

err = clf.predict(X_test) != y_test
```

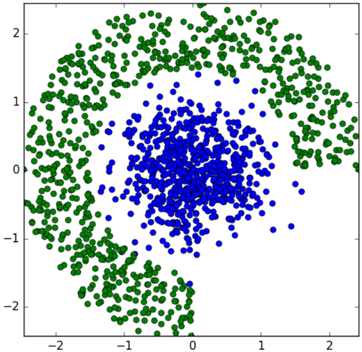
- Many of the scikit implementations of machine learning algorithms can be swapped out quickly using polymorphism
- Can construct the `clf` object here in multiple ways

Plotting Results

```
plt.figure(1)
for i in [1, 2]:
    II = y_test == i
    plt.plot(X_test[II, 0], X_test[II, 1], 'o', label=str(i))

plt.plot(X_test[err, 0], X_test[err, 1], 'ro')
plt.axis("image")

plt.figure(2)
for i in [1, 2]:
    II = y_train == i
    plt.plot(X_train[II, 0], X_train[II, 1], 'o', label=str(i))

plt.axis("image")

plt.show()
```

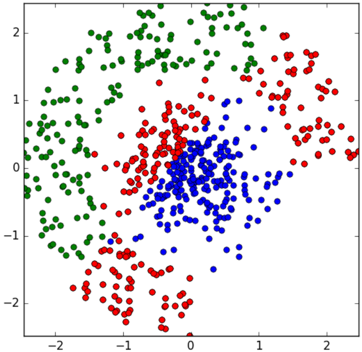
- Code above shows how we can build up a presentation of the classification results

Example Test Dataset



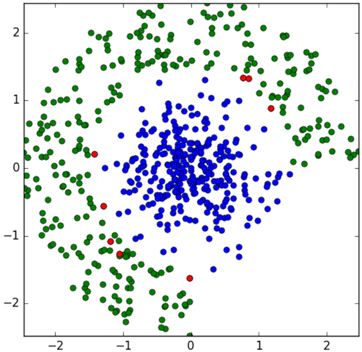
- One particular test set obtained with cross validation

Linear Discriminant Analysis Solution



- High error rate since decision boundary is only a hyperplane

Nearest Neighbors Solution



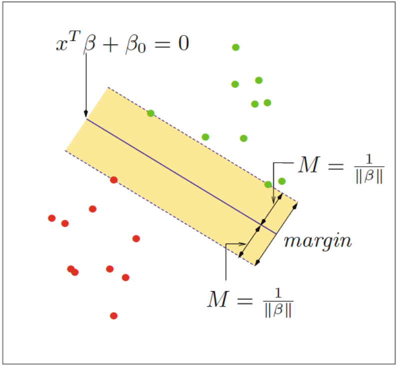
- Achieve better error rate with the nearest neighbor classifier

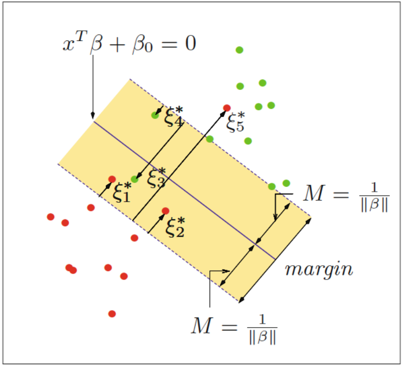
Support Vector Machines

- Support Vector Machines (SVMs) are a flexible class of machine learning algorithms
- Through selection of different *kernel* functions the separating hyperplane can be either linear or non-linear
- Allows us to fit complicated learning surfaces

Training SVMs (Linear Case)

- Attempt to find the *maximum margin* classifier
- From Tibshirani *et al*





Optimization Problem

- Maximum Margin Criterion results in following optimization problem

$$\max_{\beta,\beta_0,||\beta||=1} M \tag{3}$$

$$y_i(x_i^T\beta + \beta_0) \geq M \tag{4}$$

Optimization Problem

- Maximum Margin Criterion results in following optimization problem

$$\frac{1}{||\beta||} y_i(x_i^T \beta + \beta_0) \geq M \tag{5}$$

$$y_i(x_i^T \beta + \beta_0) \geq M||\beta|| \tag{6}$$

Optimization Problem

- Scale invariance in this problem, arbitrarily set $||\beta|| = 1/M$
- Solve by Lagrange Multipliers

$$\min_{\beta,\beta_0} \frac{1}{2}||\beta||^2 \tag{7}$$

$$y_i(x_i^T\beta + \beta_0) \geq 1 \tag{8}$$

Solution

- Solve constrained optimization with lagrange multipliers

$$L_P = (1/2)||\beta||^2 - \sum \alpha_i [y_i (x_i^T \beta + \beta_0) - 1] \tag{9}$$

$$\beta = \sum \alpha_i y_i x_i \tag{10}$$

$$0 = \sum \alpha_i y_i \tag{11}$$

$$f(x) = \sum_{i=1}^{N} \alpha_i y_i (x_i, x) + \beta_0 \tag{12}$$

The Classification Rule

- Usage of the kernel trick results in the following classification rule
- The kernel function is selected in advance the algorithm finds the weights as well as the support vectors that maximize the margin

$$\sum_{s \in \text{support vectors}} K(x_{test}, x_s) + \text{bias term} > 0 \tag{13}$$

Common Kernels

- The polynomial, radial basis function, linear and neural net are the most common kernels

$$K(x, y) = (xy^T + 1)^p \tag{14}$$

$$K(x, y) = e^{-||x-y||^2/2\sigma^2} \tag{15}$$

$$K(x, y) = \tanh(kxy^T - \delta) \tag{16}$$