Discriminant Analysis

Joshua Horowitz

Projections

Project data into subspace where separation of classes is maximized

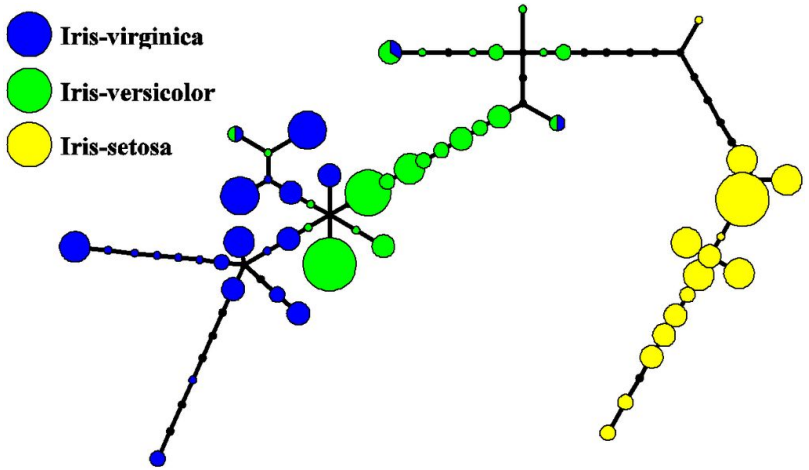
This both:

- Reduces dimensionality of the problem
- Simplifies classification

Classic Motivating Case: Irises

Have length and width of sepals and petals of iris flowers

Basic data clustering methods only find 2 clusters of data - need to train a classifier on pre-identified irises to solve



Supervised Classifiers

Have some objects sorted into categories/classes

Want to sort more objects into those classes

Many options in Ivezic:

TABLE 9.1.
Summary of the practical properties of different classifiers.

Method	Accuracy	Interpretability	Simplicity	Speed
Naive Bayes classifier	L	H	H	H
Mixture Bayes classifier	M	H	H	M
Kernel discriminant analysis	H	H	H	M
Neural networks	H	L	L	M
Logistic regression	L	M	H	M
Support vector machines: linear	L	M	M	M
Support vector machines: kernelized	H	L	L	L
K-nearest-neighbor	H	H	H	M
Decision trees	M	H	H	M
Random forests	H	M	M	M
Boosting	H	L	L	L

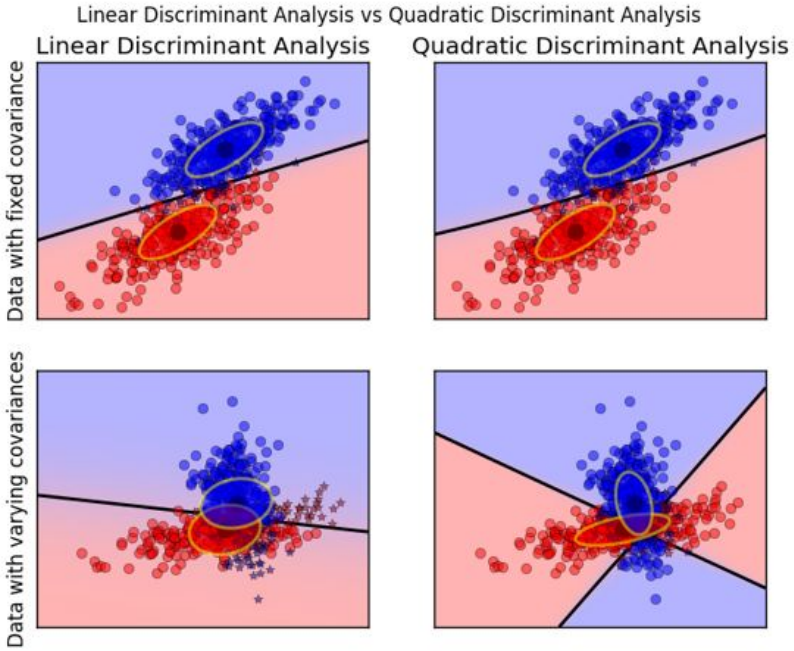
Using Supervised Classifiers

Must divide available data into training and test sets. Ideally, the training set should have balance. If imbalanced, the classifier needs to correct using cost functions or priors.

Discriminant Analysis

Multiclass classifiers with closed-form solutions

Creates decision surfaces



Sample Math for Linear Discriminant Analysis

For data X, classes y, and class means mu (average data for objects in class):

$$p(X|y=k) = \frac{1}{(2\pi)^n|\Sigma_k|^{1/2}} \exp\left(-\frac{1}{2}(X-\mu_k)^t\Sigma_k^{-1}(X-\mu_k)\right)$$

$$\log\left(\frac{P(y=k|X)}{P(y=l|X)}\right) = 0 \Leftrightarrow (\mu_k-\mu_l)\Sigma^{-1}X = \frac{1}{2}(\mu_k^t\Sigma^{-1}\mu_k - \mu_l^t\Sigma^{-1}\mu_l)$$

Implementing Discriminant Analysis

Scikit-learn has Python implementations

For training data Xtrain, training classes Ytrain, and testing data Xtest

```
def LDA(Xtrain,Ytrain,Xtest):
    LDA = LinearDiscriminantAnalysis(solver='eigen')
    LDA.fit(Xtrain, Ytrain)
    LDA_labels = LDA.predict(Xtest)
    return LDA_labels

def QDA(Xtrain,Ytrain,Xtest,lmb):
    QDA = QuadraticDiscriminantAnalysis(reg_param=lmb)
    QDA.fit(Xtrain, Ytrain)
    QDA_labels = QDA.predict(Xtest)
    return QDA_labels
```

Further Functionality

The classifiers can also take in priors giving information about relative frequencies of the classes.

The posterior probability that each object belongs to each of the classes can also be output, in addition to the most likely classification.

Kernel Discriminant Analysis

Generalizes discriminant analysis so do not need to know decision surface

Uses a kernelizing, weighting function to transform problem to feature space (kernelized Principal Component Analysis)

Then applies LDA to new problem, with decision surface of original problem now optimized for a linear solution (discriminant vectors in original space are likely nonlinear)

Implementing KDA

A binary (2 class) version exists in mlpy, with heavy cython usage

A fully Pythonic, multiclass version exists in the AGN classification group’s Github directory

Testing Discriminant Analysis

How do you know how well the classification is doing?

See https://github.com/galastrostats/general/blob/master/CHood_Classifiers.pdf

Relies on knowing the “correct answers,” so may need to split the solved objects into a training set and a verification set (or use k-fold cross-validation)

Sources and Further Reading

Ivezic, et al. Chapter 9

Scikit-learn http://scikit-learn.org/stable/modules/lda_qda.html#lda-qda

Mlpy http://mlpy.sourceforge.net/docs/3.5/nonlin_class.html#kernel-fisher-discriminant-classifier

Alexander Gray and Ryan Riegel. Large-Scale Kernel Discriminant Analysis

with Application to Quasar Discovery.

Jian Yang, et al. Essence of kernel Fisher discriminant: KPCA plus LDA.

Wikipedia. Iris flower data set.