Fisher's Linear Discriminant Analysis

Fisher's Linear Discriminant Analysis(LDA) is a dimension reduction technique commonly used for supervised classification problems. The goal of LDA is to project the features in higher dimension space onto a lower dimensional space, such that samples from different distribution can be 'separated'. Through this project, we are going to learn the basic about Fisher's LDA, and to explore practical issues for a couple of datasets from applications.

Contents

Fisher's LDA via a toy problem

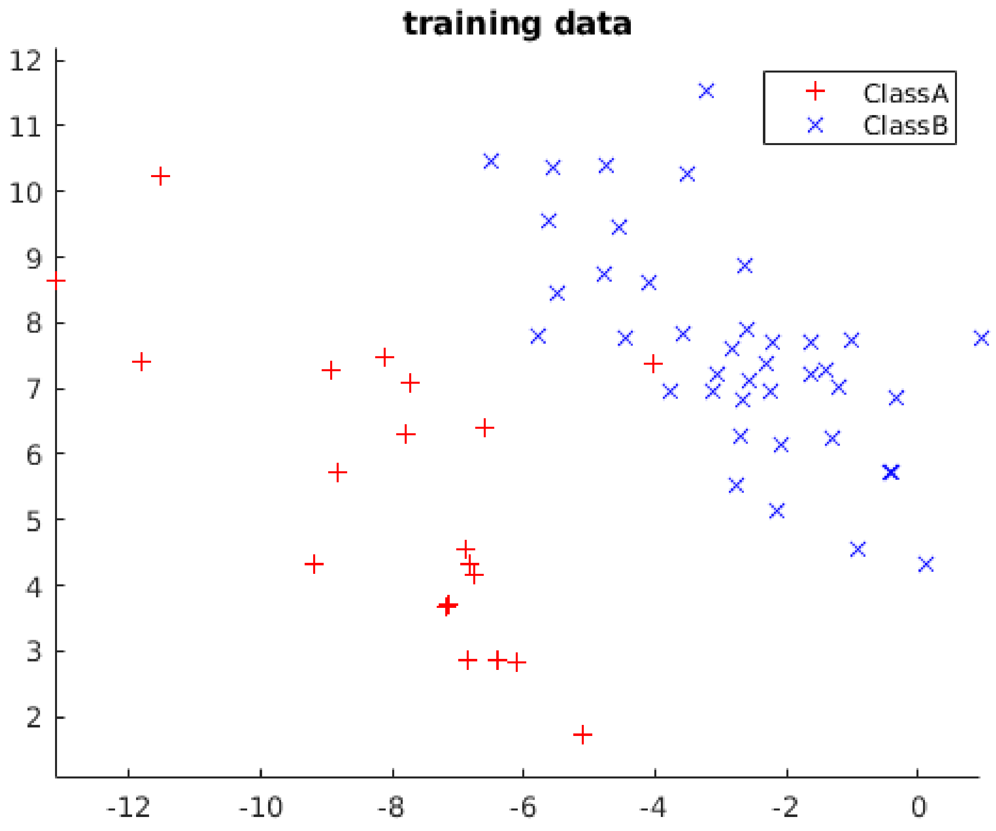
Let's start with a toy problem to get an intuition about Fisher's LDA.

Dataset generation: The following code produces two classes of data points, labelled A and B, each drawn from a normal distribution (with different mean and covariance). Each data sample x is a vector of lenth 2, and they are collected in matrices with each column corresponding to a data point. Here we make two datasets: training and testing. The training data will be used to train the `classifier', while the testing data used to assess the accuray of classification.

```
rng(0);
% generate data points
muA = [-8;5]; muB = [-3;8];
theta = pi/4;
T = [cos(theta),-sin(theta); sin(theta), cos(theta)];
S = T*diag([1,2])*T';
DataA = S*randn(2, 40) + muA * ones(1,40);
DataB = S*randn(2, 60) + muB * ones(1,60);

% training and testing data: each column is a sample
TrainA = DataA(:,1:end-20);
TestA = DataA(:,end-20+1:end);
TrainB = DataB(:,1:end-20);
TestB = DataB(:,end-20+1:end);

% show data points
figure(1)
hold on;
plot(TrainA(1,:),TrainA(2,:), '+r', 'DisplayName','ClassA');
plot(TrainB(1,:),TrainB(2,:), 'xb', 'DisplayName','ClassB');
axis equal; legend show; title('training data')
```



Fisher's LDA attempts to find a_ separation_ *vector* v onto which the projection of different classes are `best separated' by solving the optimization problem

$$\max_{\|v\|\neq 0} \frac{(v^T m_A - v^T m_B))^2}{v^T(\Sigma_A + \Sigma_B)v},$$

where m_C, Σ_C are sampled mean and covariance matrices for $C \in \{A, B\}$.

Task 1: Show that an optimal solution is given by

$$v = (\Sigma_A + \Sigma_B)^{-1}(m_A - m_B).$$

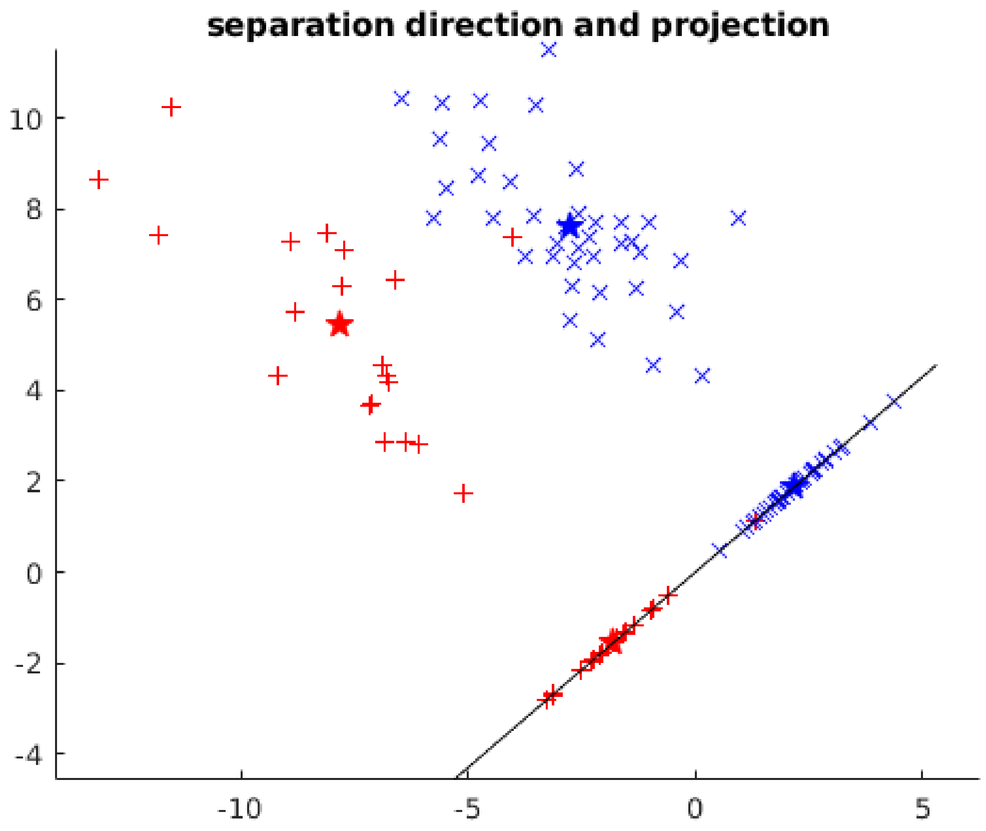
The separation vector computed from training datasets is illustrated below. As we can see from the figure, the projection of the two classes onto the direction v (the solid line) are well separated. The five-pointed stars represent mean values.

```
% sample mean & covariance
mA = mean(TrainA')';
mB = mean(TrainB')';
sA = cov(TrainA');
sB = cov(TrainB');

% separation vector
v = (sA+sB)\(mA-mB);
v = v/norm(v);

% show data
projA = v*v'*TrainA; projB = v*v'*TrainB;
projmA = v*v'*mA; projmB = v*v'*mB;
figure(2)
```

```
hold on;
plot([TrainA(1,:),projA(1,:)], [TrainA(2,:),projA(2,:)], '+r');
plot([TrainB(1,:),projB(1,:)], [TrainB(2,:),projB(2,:)], 'xb');
plot([mA(1),projmA(1)],[mA(2),projmA(2)],'pr','MarkerFaceColor','red','Markersize',10);
plot([mB(1),projmB(1)],[mB(2),projmB(2)],'pb','MarkerFaceColor','blue','Markersize',10);
plot(7*[-v(1),v(1)], 7*[-v(2),v(2)], '-k');
axis equal; title('separation direction and projection')
```



Classification: Using the computed separation vector v from above, we can classify a sample vector x based on its projection on v. Specifically, if $v^T x > c$ then we can guess x belongs to class A, otherwise, to class B (*can you explain why?*). A natural (default) choice of the thresholding constant is $c = v^T(m_A + m_B)/2$, in which case the class of x is assigned to the one with projected sample mean closer to $v^T x$.

Task 2: Use the separation vector v to build a classifer. Test and report the success rate of your classifier on the testing dataset. Note thati the success rate = 1 - missrate, where missrate = (missA + missB)/(sizeofTestA + sizeofTestB)

UCI Benchmark Problems

To show the effectiveness of LDA in real-life data, let's consider application to the sonar and ionosphere datasets from UCI Machine Learning Repository (https://archive.ics.uci.edu). The data matrices can be directly loaded as below. For each case, you will get two matrices: a data matrix containing sample data (each row is a feature vector); a label matrix with 0-1 element representing the class of each row in the data matrix.

The 'sonar' example:

```
load sonar.mat
whos('-file','sonar.mat')
```

```
  Name                Size             Bytes  Class      Attributes

  sonar_data        208x60             99840  double
  sonar_label       208x1                208  logical
```

The 'ionosphere' example:

```
load ionosphere.mat
whos('-file','ionosphere.mat');
```

```
  Name                     Size             Bytes  Class      Attributes

  ionosphere_data        351x34             95472  double
  ionosphere_label       351x1                351  logical
```

Task 3: Use the first 70% of data points as training data, and the rest as testing data. Test and report the success rate for classification by LDA, using the default threshold c.

Task 4: Try different threshold c, with value between $v^T m_A$ and $v^T m_B$. Observe how sensitive is the classification rate to the choice of c.

Task 5: For the ionosphere, the matrix $S_A + S_B$ become singular. Can you figure out why? How to address this issue? In general, when we encounter near singular matrices, what can we do?
