# 实验报告五

学号：

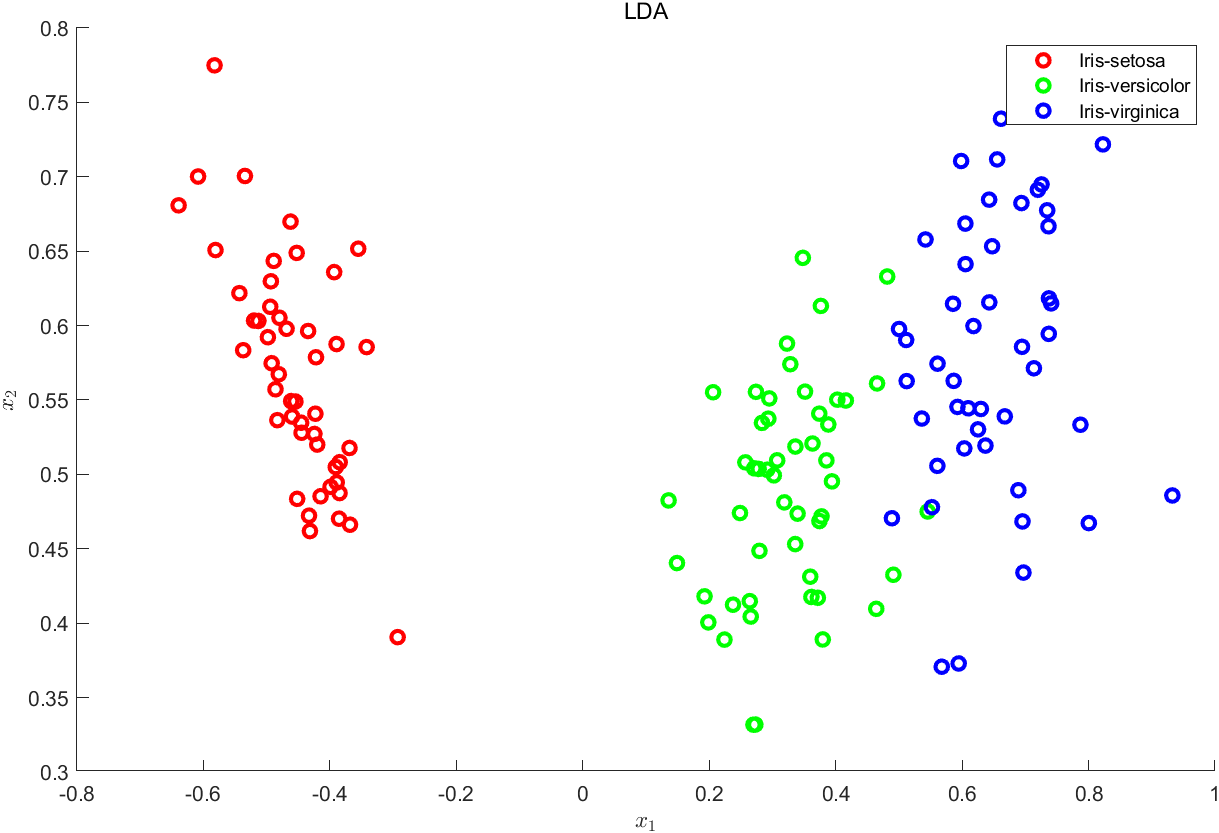
姓名：***Steven***

实验名称：线性判别分析

实验内容：使用MATLAB编程环境，用线性判别分析(LDA)算法对Iris数据集进行降维处理。

实验要求及结果：

采用LDA算法，将Iris数据集从原始的四维空间投影到二维空间，请画出Iris数据集在投影后二维空间中的分布情况（要求用不同的颜色/线型来画不同类别的投影后数据）。



问题回答：

LDA算法中的类内离散度矩阵和类间离散度矩阵分别描述的是什么？它们分别需要最大化还是最小化？这样做有什么好处？

类内离散度矩阵是每个类别内部数据点与该类别均值之间距离的协方差矩阵之和。通过最小化，可以使得同一类别内部数据点更加紧密地聚集在一起，从而提高分类器的准确性。

类间离散度矩阵是所有类别均值之间距离的协方差矩阵之和。通过最大化，可以使得不同类别之间的距离更大，从而提高分类器的泛化能力。

附实验代码：

|  |
| --- |
| %读取数据  fileID = fopen("Iris.data"); % 打开文件  data = textscan(fileID, '%f%f%f%f%s', 'Delimiter', ',');  attrib = [data{1, 1}, data{1, 2}, data{1, 3}, data{1, 4}];  class = data{1, 5};  X = attrib(1:150, :); % 读取共150条数据  label\_set = char('Iris-setosa', 'Iris-versicolor', 'Iris-virginica');  label = zeros(100, 1);  label(strcmp(class, 'Iris-setosa')) = 1;  label(strcmp(class, 'Iris-versicolor')) = 2;  label(strcmp(class, 'Iris-virginica')) = 3;  [vec, val] = LDA(X', label); % 计算  % 降维  X = X \* vec(:, 1:2);  % 画散点图  figure;  scatter(X(1:50, 1), X(1:50, 2), 'r', 'LineWidth', 2);  hold on;  scatter(X(51:100, 1), X(51:100, 2), 'g', 'LineWidth', 2);  hold on;  scatter(X(101:150, 1), X(101:150, 2), 'b', 'LineWidth', 2);  hold on;  legend('Iris-setosa', 'Iris-versicolor', 'Iris-virginica');  title('LDA');  xlabel('$x\_1$', 'Interpreter', 'latex');  ylabel('$x\_2$', 'Interpreter', 'latex');  function [vec, val] = LDA(xtr, ytr)  %%%%%%%%%%%%%%%%%%%%%%%%%%%%  % Input:  % xtr: data matrix (Each column is a data point)  % ytr: class label (class 1, ..., k)  % Output:  % vec: sorted discriminative components  % val: corresponding eigenvalues  %%%%%%%%%%%%%%%%%%%%%%%%%%%%  [D, ntr] = size(xtr);  classnum = length(unique(ytr));  miu = mean(xtr, 2);  sigmaB = sparse(D, D);  miu\_class = zeros(size(xtr, 1), classnum);  for i = 1:classnum  miu\_class(:, i) = mean(xtr(:, ytr == i), 2);  sigmaB = sigmaB + length(find(ytr == i)) \* (miu\_class(:, i) - miu) \* (miu\_class(:, i) - miu)';  end  sigmaB = (sigmaB + sigmaB') / 2;  sigmaT = (ntr - 1) \* cov(xtr');  sigmaT = (sigmaT + sigmaT') / 2;  sigmaW = sigmaT - sigmaB;  sigmaW = (sigmaW + sigmaW') / 2;  [eigvector, eigvalue] = eig(sigmaB, sigmaW);  [val, id] = sort(-diag(eigvalue));  vec = eigvector(:, id);  val = -val;  end |