Lecture 14: Dimensionality Reduction – LDA

Textbook: Chapter 5

LDA: Linear Discriminant Analysis

- LDA is also called Fisher’s LDA because Ronald A. Fisher initially formulated Fisher’s Linear Discriminant for two-class classification problem in 1936.
- Fisher’s LDA was later generalized for multi-class problems by C. Radhakrishna Rao under the assumption of equal class covariances and normally distributed classes in 1948.

LDA and PCA - similarities

- Both techniques are **linear transformation techniques** used to reduce the number of dimensions in a dataset. They project the data points into axes that are *independent* (i.e. covariance is zero among the pair of axes).
- Improve the **computational efficiency** of the learning algorithm.
- Improve the **predictive performance** by reducing the *curse of dimensionality*. This is particularly useful when we work with non-regularized models.

Differences

- PCA is an unsupervised approach and **LDA is a supervised approach**.
- PCA finds the axes with **maximum variance** for the whole data set.
- LDA finds the axes for **best class separability**. Or, the axes are selected in such a way that the interclass distance is maximized whereas the intraclass distance is minimized.
- Mathematically speaking, LDA is to find the **feature subspace that optimizes class separability**.

Conduct LDA step by step

- Standardize the *d*-dimensional dataset
- Construct the between-class scatter matrix S_B and the within-class scatter matrix S_W.
  - For each class, compute the *d*-dimensional mean vector, which is used in the construction of the two matrices in the next step.
  - Calculate $d \times d$ S_W
  - Calculate $d \times d$ S_B
- Computer the eigenvectors and corresponding eigenvalues of the matrix $S_W^{-1} S_B$
- Sort the eigenvalues by decreasing order to rank the corresponding eigenvectors.
- Choose the *k* eigen vectors that correspond to the *k* largest eigen values to construct a *d*×*k*-dimensional transformation matrix **W**; the eigenvectors are the columns of this matrix.
- Project the samples onto the new feature subspace using the transformation matrix **W**.

Step 2: Matrix $\boldsymbol{S}_B$ and $\boldsymbol{S}_W$

- **Step 2.1: calculate class-wise mean**
- For each class C_i, assume that the instances with this class label are in a set CL_i. The mean vector m_i is a *d*-dimensional vector where $m_i[j]$ is the mean value μ_j for feature *j*.
- $m_i = [\mu_1, \mu_2, \ldots, \mu_d]$, where $\mu_j = \frac{1}{|CL_i|}\sum_{\mathbf{x} \in CL_i} \mathbf{x}_j$
- Using vectorized operation, m_i can be calculated as $m_i = \frac{1}{|C_i|}\sum_{\mathbf{x} \in C_i} \mathbf{x}$
- Consider a dataset $\begin{pmatrix} 1 & 1 & Y \\ 3 & 3 & N \\ 1 & 2 & Y \\ 3 & 4 & N \\ 3 & 1 & Y \end{pmatrix}$, $m_1 = [1.7, 1.3]$ for class Y; $m_2 = [3, 3.5]$ for class N.

Step 2.2: calculate scaled $\mathbf{S}_w$

- **Step 2.2: Compute the within-class scatter matrix $\boldsymbol{S_W}$**
- For a class label C_i, the scatter matrix of instances in this class is calculated using

$$\mathbf{S}_i = \sum_{\mathbf{x} \in CL_i} (\mathbf{x} - m_i)(\mathbf{x} - m_i)^T$$

- where **x** and m_i are represented as *d* × 1 vector (i.e., column vector).
- Thus, $\mathbf{S}_i$ is a *d* × *d* matrix.

Step 2.2: calculate scaled $\mathbf{S}_w$

- Note that the instances belong to different classes may not be uniformly distributed. To make sure that each class's contribution to the scatter matrix is equal. The above scatter matrix is normalized.

$$\Sigma_i = \frac{1}{|CL_i|}\,\mathbf{S}_i = \frac{1}{|CL_i|}\sum_{\mathbf{x}\in CL_i}(\mathbf{x}-m)(\mathbf{x}-m_i)^T$$

- $\boldsymbol{\Sigma_i}$ **is the covariance matrix** for instances with class label C_i.

- Assume that the dataset has C classes in total. Then, the within-class scatter matrix is calculated as

$$\mathbf{S}_w = \sum_{i=1}^{C}\Sigma_i$$

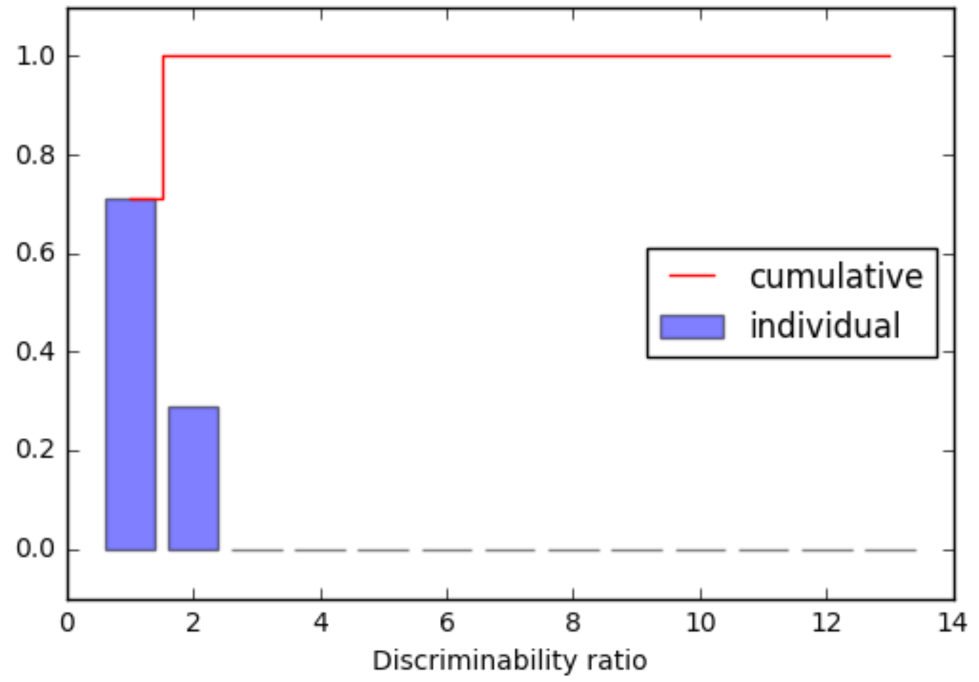
- The output $\boldsymbol{S}_w$ is a *d* × *d* matrix.

Step 2.3: calculate between-class matrix

- $\mathbf{S}_B = \sum_{i=1}^{C} |CL_i|(m_i - m)(m_i - m)^T$
- Where *C* is the total number of classes, $|CL_i|$ is the number of instances with class label C_i.
- m_i is the mean vector for instances with class label C_i and *m* is the mean vector for all the instances in the dataset.
- The output is a *d* × *d* matrix.

More details

- Linear discriminants are the eigenvectors from $\boldsymbol{S}_W^{-1}\ \boldsymbol{S}_B$.
- We can plot a similar figure as the *explained variance plot* to show how much the class-discriminatory information is captured.
- In LDA, the number of linear discriminants is at most *C* − 1 where *C* is the number of class labels, because the in-between scatter matrix $\boldsymbol{S}_B$ is the sum of C matrices with rank C-1 or less.



Using scikit-learn library to conduct LDA

```
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis as LDA
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score

lda = LDA(n_components=2)
X_train_lda = lda.fit_transform(X_train_std, y_train)

tree_model = DecisionTreeClassifier(criterion='gini', max_depth=4,
                                    random_state=1)
tree_model.fit(X_train_lda, y_train)

X_test_lda = lda.transform(X_test_std)
y_pred = tree_model.predict(X_test_lda)
acc = accuracy_score(y_pred, y_test)
print("DT+LDA acc=", acc)
```

```
DT+LDA acc= 0.9629629629629629
```

- The **n_components** denotes the number of LDs.
- Wine dataset (3 class labels)
  - Accuracy (DT) is 0.89
  - Accuracy (DT+PCA) is 0.92
  - Accuracy (DT+LDA) is 0.96

Discussions

- Assume the data is normally distributed.
- LDA can work as a classifier itself.
  - Help page with fit() and predict() method.