CENTRO UNIVERSITÁRIO FEI
Leonardo Contador Neves – 118315-1

Tópicos Especiais em Aprendizagem:

Linear Discriminant Analysis (LDA)
Análise da Discriminante Linear

São Bernardo do Campo, SP
2018

SUMÁRIO

1 INTRODUÇÃO

Este relatório busca a implementação do algoritimo LDA, um método de classificação e redução de dimensão para aplicação em diversos problemas onde podemos separá-los linearmente.

2 REVISÃO BIBLIOGRÁFICA

A análise discriminante é uma técnica da estatística multivariada utilizada para discriminar e classificar objetos. É uma técnica da estatística multivariada que estuda a separação de objetos de uma população em duas ou mais classes. A discriminação ou separação é a primeira etapa, sendo a parte exploratória da análise e consiste em se procurar características capazes de serem utilizadas para alocar objetos em diferentes grupos previamente definidos. A classificação ou alocação pode ser definida com um conjunto de regras que serão usadas para alocar novos objetos.

Contudo, a função que separa objetos pode também servir para alocar, e, o inverso, regras que alocam objetos podem ser usadas para separar.
Normalmente, discriminação e classificação se sobrepõem na análise, e a distinção entre separação e alocação é confusa. O problema da discriminação entre dois ou mais grupos, visando posterior classificação, foi inicialmente abordado por Fisher (1936).
Consiste em obter funções matemáticas capazes de classificar um indivíduo X (uma observação X) em uma de várias populações (pi)i, (i=1, 2, ..., g), com base em medidas de um número pode características, buscando minimizar a probabilidade de má classificação, isto é, minimizar a probabilidade de classificar erroneamente um indivíduo em uma população i, quando realmente pertence a população (pi)j, (i≠j) i,j=1, 2, ..., g.

3 METODOLOGIA

Para implementar o LDA, vamos descobrir algumas variáves que vão ser necessárias em cada etapa do cálculo. Primeiramente, vamos calculas a média da amostra, a covariancia da amostra e o Grand Mean (que seria uma média do grupo de amostras). Para estes cálculos, temos as equações:

$$\bar{x}_i = \frac{1}{N_i} \sum_{j=1}^{N_i} x_{i,j}$$

Equação 1: Média da amostra

$$S_i = \frac{1}{(N_i - 1)} \sum_{j=1}^{N_i} (x_{i,j} - \bar{x}_i)(x_{i,j} - \bar{x}_i)^T$$

Equação 2: Covariância da amostra

$$\bar{x} = \frac{1}{N} \sum_{j=1}^{N_i} N_i x_i = \frac{1}{N} \sum_{i=1}^{g} \sum_{j=1}^{N_i} x_{i,j}$$

Equação 3: Grand Mean Vector

Onde N é o número de elementos da amostra e x é cada elemento da amostra e x barrado é a média da amostra. Para continuarmos com os calculos precisamos agora chegar em duas matrizes essenciais para o algoritmo LDA, a matriz de variância entre as classes (between-classes) e a matriz de variância intra-classe (within-classes). Cada uma delas pode ser calculada como mostra as equações à seguir:

$$S_b = \sum_{i=1}^{g} N_i (\bar{x}_i - \bar{x})(\bar{x}_i - \bar{x})^T$$

Equação 4: Between-class scatter matrix (matriz entre classes)

$$S_w = \sum_{i=1}^{g} (N_i - 1) S_i = \sum_{i=1}^{g} \sum_{j=1}^{N_i} (x_{i,j} - \bar{x}_i)(x_{i,j} - \bar{x}_i)^T$$

Equação 5: Within-class scatter matrix (matriz intra classe)

O critério Fisher se trata da determinação dos auto-vetores de Sb/Sw (matrizes apresentadas à cima), tendo por finalidade extrair coeficientes que minimizam o espalhamento de padrões de mesma classe entre classes e maximizam o espalhamento de padrões de classes diferentes. Para encontrarmos a relação entre essas duas matrizes, vamos achar a matriz de projeção Plda, que tem por objetivo maximizar a relação da determinante da matriz Sb com a determinante da matriz Sw (critério de Fisher), sendo econtrada como mostra a equação a seguir:

$$P_{lda} = \arg\max_{P} \frac{\left|P^T S_b P\right|}{\left|P^T S_w P\right|}$$

Equação 6: Matriz de projeção da relação de Fisher

Uma vez com a matriz de projeção calculada vemos que ela é a solução do problema do autosistema mostrado na equação a baixo:

$$S_b P - S_w P \Lambda = 0$$

Onde multiplicando ambos os lados pela inversa de Sw, temos que:

$$S_w^{-1} S_b P - S_w^{-1} S_w P \Lambda = 0$$
$$\left(S_w^{-1} S_b\right) P = P \Lambda$$

Assim, se o Sw não é uma matriz singular, o critério de Fisher é maximizado quando a matriz de projeção é composta por autovetores de:

$$S_w^{-1} S_b$$

Equação 7: Critério de Fisher

4 DESENVOLVIMENTO

Para implementar os algoritmos a cima descritos, foi feita uma Classe em *Python* que tem uma funções principal, a *fit()* que basicamente implementa o algoritmo de LDA. A função fit recebe 2 argumentos principais, as informações e o numero de componentes para decomposição. Os outros cáculos envolvidos no algoritmo, foram feitos através da biblioteca do *Python para operações matemáticas, a Numpy*. Para a manipulação das bases de dados, foi usado a biblioteca *Pandas.*

Com a biblioteca *Numpy* fazemos uso da função que calcula os autovetores e autovalores e de operações de multiplicação de matrizes simples e transposição.

4.1 CLASSE PARA CÁLCULO DO LDA

4.1.1 Método de *fit* (Cálculo do LDA)

Esse é o método principal do algoritmo, nele são implementados todos os cálculos matriciais e todos os outros que envolvem o algoritmo. Para começar, vamos calcular um vetor de média n dimensional. Esse vetor é responsável por conter as médias das n dimensões de cada classe.

```
#Computing the d-dimensional mean vectors
ld = int(data['target'].max()) + 1  #getting the number of the classes
dim = len(data.keys())-1 #getting the length of features
mean_vectors = []
for i in range(ld):
    mean_vectors.append(np.mean(data[data['target']==i].iloc[:,:dim]).values)
print(mean_vectors)
```

De posse do vetor de n dimensões da média, vamos agora calcular a matriz de variação intra classes (Sw)

```
#Computing the Scatter Matrices
scatter_matrices = np.zeros((dim,dim))
for i,j in zip(range(dim-1), mean_vectors):
    class_sc_mat = np.zeros((dim,dim))
    for row in data[data['target']==i].iloc[:,:dim].values:
        row, j = row.reshape(dim,1), j.reshape(dim,1)
        class_sc_mat += (row-j).dot((row-j).T)
    scatter_matrices += class_sc_mat
print(scatter_matrices)
```

Agora calculamos a matriz de variação entre classes (Sb), como é mostrado a seguir.

```
# Between-class scatter matrix
overall_mean = np.mean(data.iloc[:,:dim].values, axis=0)
S_B = np.zeros((dim,dim))
for i,mean_vec in enumerate(mean_vectors):
    n = data[data['target']==i].values.shape[0]
    mean_vec = mean_vec.reshape(dim,1)
    overall_mean = overall_mean.reshape(dim,1)
    S_B += n * (mean_vec - overall_mean).dot((mean_vec - overall_mean).T)
print(S_B)
```

De posse das matrizes Sw e Sb, vamos agora com a ajuda das funções da biblioteca *Numpy*, calcular os autovalores e autovetores do produto da multiplicação entre a matriz inversa de Sw vezes a matriz Sb e em seguida, vamos formar pares de autovalores e autovetores numa lista e organizá-la de modo que o primeiro elemento da lista contenha o maior valor de autovalor, atrelado ao seu respectivo autovetor e o ultimo elemento da lista seja com o menor valor de autovalor. A imagem a seguir ilustra esses calculos.

```
#Getting the eigvectors and eigvalues
eig_vals, eig_vecs = np.linalg.eig(np.linalg.inv(scatter_matrices).dot(S_B))

# Make a list of eigenvalue and eigenvector
eig_pairs = [(np.abs(eig_vals[i]), eig_vecs[:,i]) for i in range(len(eig_vals))]

# Sort the (eigenvalue, eigenvector) from high to low
eig_pairs = sorted(eig_pairs, key=lambda k: k[0], reverse=True)
```

Agora, de posse dos autovetores, já podemos selecioná-los e transformar nosso conjunto de informação com base nesses autovetores. Para isso, primeiramente é feito um laço onde são selecionados os autovetores de acordo com o número de componentes adquiridos pela função *fit()*. Esses valores são jogados na variável W (autovetores principais) e que será usada para compor a variável X_lda, que é a variável do conjunto de informações transformados no novo espaço vetorial. Para fazer a transformação das informações, apenas foi pego o conjunto de informações original e multiplicado pela matriz de autovetores W, formando um novo conjunto de dados transformados. A figura a seguir ilustra esse raciocínio.

```
W = []
for x in range(n_components):
    W.append(eig_pairs[x][1].reshape(dim,1))

W = np.array(W).T[0].real

# Getting data back
X_lda = data.iloc[:,:dim].values.dot(W.real)

self.lda_data = X_lda
self.eig_vecs = W
```

5 RESULTADOS

O algoritmo implementado foi testado na base de dados iris, que representa um conjunto de informações de 4 dimensões, onde podemos classificar os dados de espécies de flores em 3 tipos. As dimensões do conjunto de dados são basicamente dos tamanhos das pétalas e sépalas e que apresentam uma distribuição significativa no espaço.

Podemos plotar duas à duas variáveis, onde as imagens a seguir, mostram o comprimento pela largura das Pétalas e Sépalas dos três tipos de espécies.

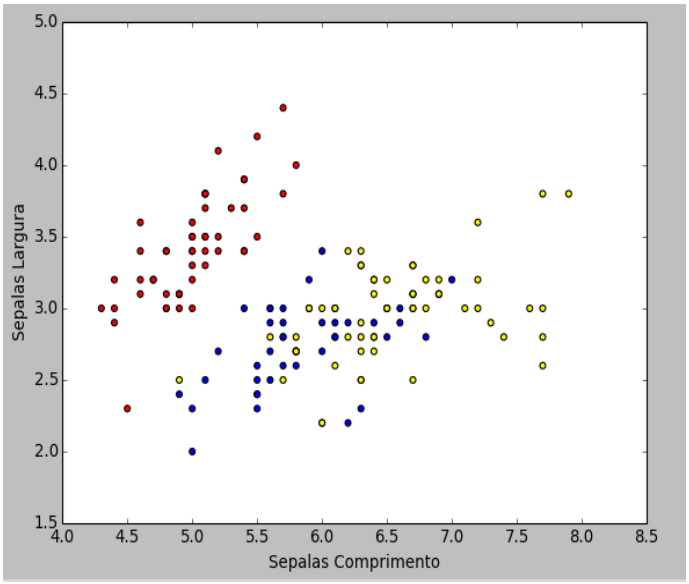

Gráfico 1: Plot de Sépalas por comprimento versus largura

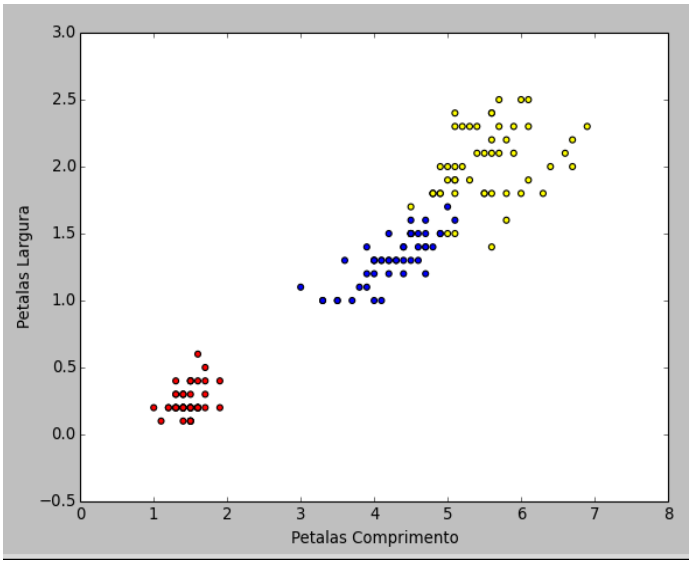

Gráfico 2: Plot de Pétalas por comprimento versus largura

Nas ilustrações a cima, podemos ver que cada cor representa uma das três classes e observamos que, dada a distribuição de comprimento e largura de cada classe, podemos separar linearmente as classes apenas aplicando o método visual. Porém, ainda há questionamentos, pois as classes apresentam uma proximidade muito grande umas das outras, principalemnte visível no primeiro gráfico.

Aplicando o método proposto no trabalho, usando duas componentes, podemos maximizar a variação entre as classes (proposta do modelo de Fisher), assim temos o gráfico da figura a seguir, implementado com as quatro dimensões e resultando apenas em dois componentes lineares de máxima variação entre as classes.

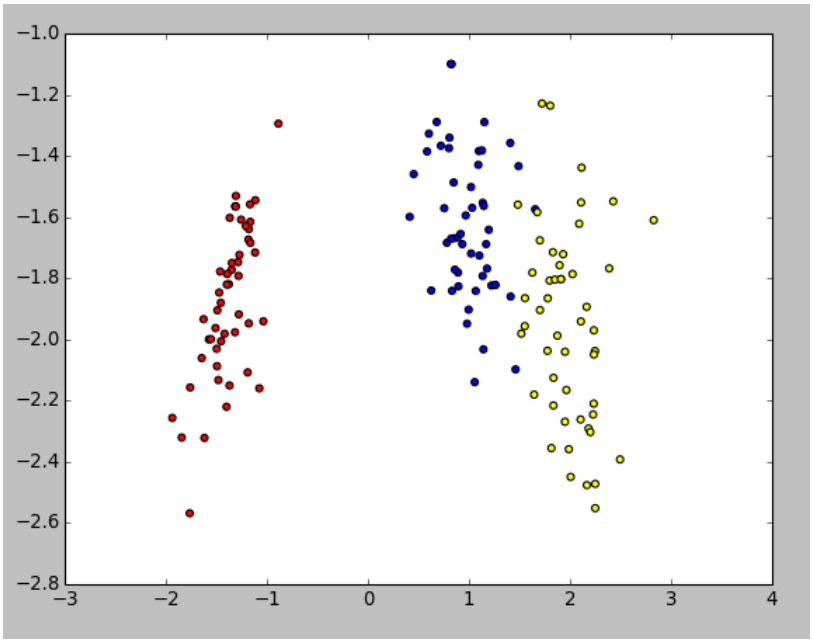

Gráfico 3: Aplicação do LDA no conjunto de dados para duas componentes.

Podemos agora notar que com a aplicação do algoritmo, as classes ficam muito mais visíveis pela classificação visual, melhorando assim, possivelmente, um algoritmo que irá classificar esses grupos. Para uma segunda análise, fiz o retorno do algoritmo para apenas a primeira componente principal da matriz de projeção e obtive o seguinte resultado.

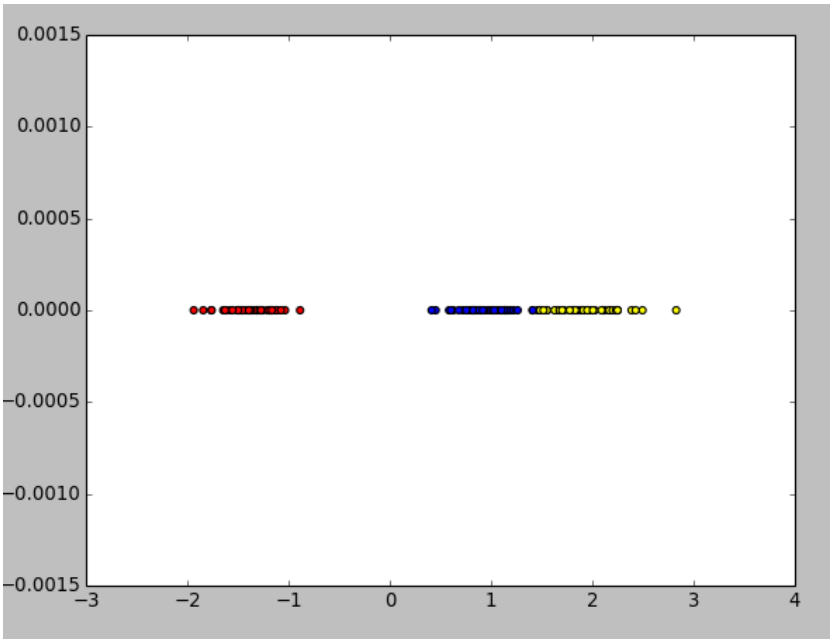

Gráfico 4: Aplicação do LDA no conjunto de dados para uma componente.

Podemos notar agora uma também boa classificação visual para os dados transformados, sendo assim também plausível de entrada para um algoritmo de classificação, visto que as classes estão bem separadas.

Contudo, o LDA tem um problema de performace quando as classes no conjunto de dados tem um número maior que o próprio conjunto de dados. Quando isso acontece a matriz precisamos aplicar um método de redução de dimensionalidade, sendo assim, junto ao LDA, podemos aplicar a análise da componente principal como primeira etapa do nosso processo. Para o problema com a base irirs, temos que o número de classes versus o número de amostras, não impede o funcionamento do LDA, mas para uma melhor experiência coma redução de dimensionalidade como etapa anterior ao LDA, temos na figura à baixo a implementação do PCA para redução de dimensão emseguida o LDA.

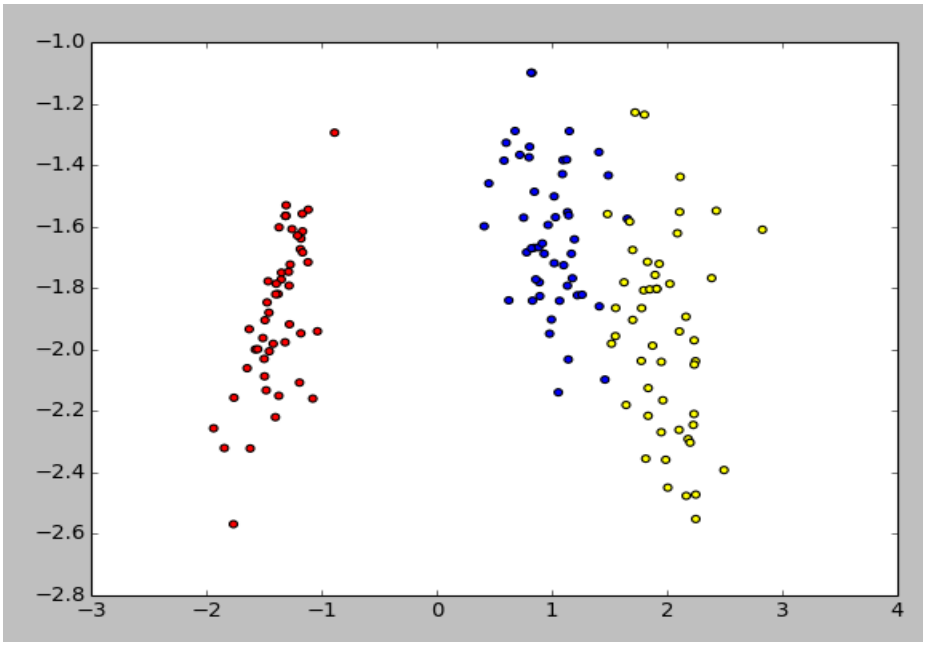

Gráfico 5: Implementação do PCA + LDA

5 CONCLUSÃO

Como pudemos analisar nos resultados, o LDA é um exelente algoritmo para manipular as informações amostrais, de tal modo que tenha uma axcelencia em separação linear das classes contidas no conjunto de informações. Dadas certas condições de funcionamento, como, apenas modelos de separação linear podem ser usados com o algoritmo ou sua restrição perante à relação de número de característocas e número de amostras, o algoritmo se mostrou rápido e entregou o desejado.

Em relação à comparação com o método PCA, ambos apresentam características diferentes na abordagem do problema. Enquanto o PCA tem sua componente principal na direção de mais variação dos dados amostrados, não se importando com classes no conjunto de informação, o LDA tem suas componentes principais na direção de maior separação das classes assumidas, assim, a transformação do conjunto de amostras original, para esse novo sistema de vetores, possibilita a máxima separação entre classes.

REFERÊNCIAS

Wikipedia contributors. (2018, August 25). Linear discriminant analysis. In *Wikipedia*, *The Free Encyclopedia*. Retrieved 04:23, October 31, 2018, from https://en.wikipedia.org/w/index.php?title=Linear_discriminant_analysis&oldid=856541284

ANÁLISE DE COMPONENTES PRINCIPAIS. In: WIKIPÉDIA, a enciclopédia livre. Flórida: Wikimedia Foundation, 2018. Disponível em: <https://pt.wikipedia.org/w/index.php?title=An%C3%A1lise_de_componentes_principais&oldid=53098518>. Acesso em: 28 set. 2018.

NumPy Reference. Disponível em: <https://docs.scipy.org/doc/numpy-1.15.1/reference/index.html>. Acesso em: 28 set. 2018.

Discriminantes Lineares. Disponível em: <http://www.vision.ime.usp.br/~teo/publications/qualificacao/node20.html>. Acesso em: 28 set. 2018.