Atividade 3: Linear Discriminant Analysis e Most Discriminant Features

Linear Discriminant Analysis e Most Discriminant Feature são técnicas utilizadas para agrupamento de dados por meio de combinações lineares . Este trabalho apresenta um implementação do método em linguagem Java 1.8, para a disciplina de tópicos especiais em aprendizagem.

Introdução

Este artigo mostra como a aplicação que implementa lda e o MDF foram codificados. Para isto, uma breve explicação é feita sobre cada método desenvolvido, que compõe o exercício. O objetivo do trabalho é implementar um modelo simples que atenda os requisitos da disciplina.

Métodos

Linear Discriminant Analysis (LDA) é um método que procura a combinação linear de características que conseguem separar as amostras em grupos distintos. Para que isso aconteça, a técnica tenta maximizar a distância entre a classes e minimizar a distância intra-classes.

Para calcular a LDA de um conjunto de dados, utilizamos a equação:

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

P são os autovetores;

• Λ são os autovalores;

• (S w −1 S b) é a multiplicação da matriz de dispersão intra-classes invertida pela matriz de dispersão entre classes.

• Entre classes

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

• Intra-classes

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

Onde:

• x̄ é a média geral

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

• x̄ é a média da classe

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

• x i,j é uma amostra da classe π i

• N i é o número de amostras da classe π i

• g é o total de classes.

Most Discriminant Features é a aplicação da técnica LDA no subconjunto de dados que mais discriminam o conjunto de dados obtido através da técnica PCA. Assim, chegamos a

equação:

$$(P_{pca}^T S_w P_{pca})^{-1}(P_{pca}^T S_b P_{pca})P = P\Lambda$$

Desenvolvimento

Os métodos foram implementados na linguagem de programação C++. Diferente métodos foram criados que em conjunto implementam os métodos MDF e LDA.

Matriz de Dispersão

```
for (int c=0; c<ClassVector.size(); c++){

      int N = ClassVector[c].size();

      Sb = this->somaMatrix( Sb,

this->multiplicaMatrix( N,

this->multiplicaMatrix( this->transpose(this->subMa

trix(meanSample, meanClass[c])),

this->transpose(this->transpose(this->subMatrix(me

anSample, meanClass[c])))

            )

         )

      );

   }
```

Matriz de Dispersão Intra-Classes

```
for (int c=0; c<ClassVector.size(); c++){

      for (int i=0; i<ClassVector[0].size(); i++){

         Sw = this->matrices_sum(

            Sw,

            this->multiplicaMatrix(

               this->transpose(this->subMatrix(ClassVector[c][i],

meanClass[c])),

this->transpose(this->transpose(this->subMatrix(ClassVector[c][i]

, meanClass[c])))

            )

         );

      }

   }
```

Calculo da Matriz Inversa

```
public static double[][] calculaInversa(double[][] matriz, int

tamanho) {

      double[][] inversa = new double[tamanho][tamanho];

      double[][] adjunta = calculaAdjunta(matriz, tamanho);

      double determinante = calculaDeterminante(matriz,

tamanho);
```

```
    for (int i = 0; i < tamanho; i++){

        for (int j = 0; j < tamanho; j++){

            inversa[i][j] = (1/determinante) * adjunta[i][j];

        }

    }

    return inversa;

}
```

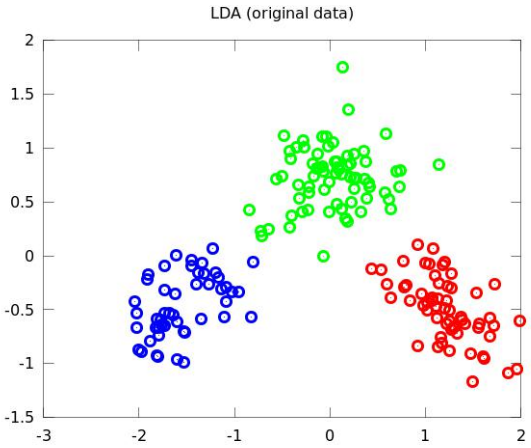
Calculo da Matriz Transposta

```
public static double[][] calculaTransposta(double[][] matriz, int

tamanho) {

    double[][] transp = new double[tamanho][tamanho];




    for (int i = 0; i < tamanho; i++) {

        for (int j = 0; j < tamanho; j++) {

            transp[j][i] = matriz[i][j];

        }

    }
```

```
    return transp;

}
```

Resultados

Abaixo estão representados de forma matricial os resultados obtidos a partir dos dados de exemplo de entrada.



Iris DataSet

- Médias

$$\mu = \begin{bmatrix} 3.054 \\ 3.75867 \\ 1.198867 \end{bmatrix}$$

- Medias das classes

$$\begin{bmatrix} 3.418 & 1.464 & 0.244 \\ 2.77 & 4.26 & 1.326 \\ 2.974 & 5.552 & 2.026 \end{bmatrix}$$

- Matriz de covariância PCA

$$C = \begin{bmatrix} 0.188004 & -0.321713 & -0.117981 \\ -0.321713 & 3.11318 & 1.29639 \\ -0.117981 & 1.29639 & 0.582414 \end{bmatrix}$$

- Autovalores PCA

$$x = \begin{bmatrix} 3.6928 \\ 0.0340654 \\ 0.15673 \end{bmatrix}$$

- Autovetores PCA

$$\lambda = \begin{bmatrix} -0.251794 & -0.145175 & 6.18389 \\ 2.37636 & -0.436194 & 0.234419 \\ 1 & 1 & 1 \end{bmatrix}$$

- Matriz de dispersão intra-classes Sw invertida

$$x = \begin{bmatrix} 0.0790879 & -0.0119072 & -0.050863 \\ -0.0119072 & 0.0497007 & -0.040855 \\ -0.0508633 & -0.0408552 & 0.243874 \end{bmatrix}$$

- Matriz de dispersão entre classes Sb

$$x = \begin{bmatrix} 10.9776 & -56.0552 & -22.4924 \\ -56.0552 & 436.644 & 186.908 \\ -22.4924 & 186.908 & 80.6041 \end{bmatrix}$$

- Autovalores LDA

$$x = \begin{bmatrix} 30.29999 \\ -4.44089e-014 \\ 0.277724 \end{bmatrix}$$

- Autovetores LDA

$$\lambda = \begin{bmatrix} -0.628115 & -0.397312 & 0.780876 \\ 0.483007 & -0.479062 & -0.325435 \\ 1 & 1 & 1 \end{bmatrix}$$

$$(P_{pca}^T S_w P_{pca})^{-1} (P_{pca}^T S_b P_{pca}).$$

$$C = \begin{bmatrix} 29.7474 & 0.105047 & -1.51019 \\ 63.0833 & 0.272534 & -1.8994 \\ -6.07643 & -0.0127263 & 0.537109 \end{bmatrix}$$

Conclusão

A partir do algoritmo implementado foi possível executar os exemplos indicados com sucesso, absorvendo os conceitos teóricos e as aplicações.

Referências

1. Linear discriminant analysis - Wikipedia Disponível em (https://goo.gl/9RoQ85). Acesso em: 27 de out. de 2017

2. Linear discriminant analysis - University of Toronto, Faculty of Applied Science & Engineering Disponível em (https://goo.gl/UZUE4P). Acesso em: 27 de out. de 2017