Atividade 2 **Yan A. S. Duarte**
PEL 208 Tópicos Especiais em Aprendizagem
Prof. Reinaldo A. C. Bianchi Entrega: 18/10/2017

Introdução

Esse relatório tem como objetivo detalhar a teoria, a implementação, os resultados e a conclusão da segunda atividade do curso. A propósta do exercício é implementar o método Principal Component Analysis (PCA).

Teoria

Principal Component Analysis

O método de Principal Component Analysis (PCA) tem como objetivo transformar um conjunto de dados para um novo sistema de coordenadas levando em consideração a variância das características desses dados. Com isso, podemos selecionar as características que possuem mais variância, pois representam informações distintas dos dados, e descartar as que possuem menos variância, que podem ser ruídos. Vale resaltar que, após utilização do PCA, o conjunto inicial de dados acaba diminuindo. O funcionamento do método consiste em seis passos que serão apresentados a seguir.

Adquirir um banco de dados

Nessa etapa são adquiridas amostras de algum dado que se queira observar, como por exemplo, o banco de dados gabminder que possui diversos dados (como incidência de câncer, média de fumantes, consumo de cigarros, entre outros) de diversos países em diversos anos.

Subtrair a média de cada coluna

Após escolher quais os dados serão analisados, deve-se fazer a média de cada característa e subtrair esse valor de cada amostra. Isso faz uma translação nos dados movendo-os de forma que a média passe a ficar no eixo 0.

Calcular a matriz de covariância

A matriz de covariância é uma matriz quadrada onde o tamanho de sua dimensão é igual ao número de características do conjunto de dados. Cada elemento da matriz de covariância representa o relacionamento entre duas dimensões e pode ser representada por:

$$C = \begin{bmatrix} cov(x,x) & cov(x,y) & cov(x,z) \\ cov(y,x) & cov(y,y) & cov(y,z) \\ cov(z,x) & cov(z,y) & cov(z,z) \end{bmatrix}$$

A covariância entre as dimenssões pode ser obtida da seguinte maneira:

$$cov(x,y) = \frac{\sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y})}{(n-1)}$$

Calcular os autovetores e autovalores da matriz de covariância

Uma matriz possui autovetores e autovalores quando a mesma é quadrada. Os autovalores de uma matriz representa as propriedades essenciais da mesma. Para calcular autovalores de uma matriz, devemos subtrair um escalar λ de cada entrada diagonal e, após isso, calcular o determinante da seguinte forma:

$$det\left\{\begin{bmatrix} a & b \\ c & d \end{bmatrix} - \begin{bmatrix} \lambda & 0 \\ 0 & \lambda \end{bmatrix}\right\}$$

O valor de λ só é um autovalor apenas se, e somente se, o resultado do calculo da determinante for igual a 0.

Após calcular o autovalor, conseguimos calcular o autovetor da seguinte maneira:

$$\begin{bmatrix} a-\lambda & b \\ c & d-\lambda \end{bmatrix} \begin{bmatrix} v1 \\ v2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

Escolher os componentes que serão utilizados e criar um vetor de características

Após a execução do método, o maior autovalor encontrado representa a componente principal do conjunto de dados, enquanto o menor representa características que não distinguem muito o seu conjunto de dados. O vetor com as característas escolhidas é formado pelo autovetor correspondente em ordem decrescente.

Derivar o novo conjunto de dados

Após escolher o número de características que serão utilizadas, devemos fazer a multiplicação do vetor de características escolhidas transposto pelo dataset original transposto.

$$dado\_final = caracteristicas\_escolhidas^T * dado\_original^T$$

Implementação

Para a elaboração do exercício foi utilizada a linguagem de programação C++. Foram criadas funções separadas para calcular cada etapa do método que serão descritas a seguir.

check_matrix_size

Função para exibir retorno de erro caso a matriz de entrada tenha dimensão diferente de 2x2 ou 3x3.

```
int Matrix::check_matrix_size(vector< vector<double> > mat){
    int lin = mat.size(),
        col = mat[0].size();

    if (lin != col){
        cout<<"\n\nMatrix nao e quadrada.\n";
        system("pause");
        return 0;
    }
    return 1;
};
```

data_mean

Função para encontrar a média de cada característica tanto do vetor X quanto do vetor y.

```
vector <double> Matrix::data_mean(vector< vector<double> > X,
                                 vector< vector<double> > y)
{
    int lin_X = X.size(),
        col_X = X[0].size(),
        lin_Y = y.size(),
        col_Y = y[0].size();

    double sum;
    vector < double > mean;
    for (int j=0; j<col_X; j++){
        sum = 0;
        for (int i=0; i<lin_X; i++)
            sum += X[i][j];
        mean.push_back(sum/lin_X);
    }

    for (int j=0; j<col_Y; j++){
        sum = 0;
        for (int i=0; i<lin_Y; i++)
            sum += y[i][j];
        mean.push_back(sum/lin_Y);
    }

    return mean;
}
```

covariance

Função para encontrar a matriz de covariância do conjunto de dados de entrada.

```
vector< vector<double> > Matrix::covariance(vector< vector<double> > x_y){
    int lin = x_y.size(),
        col = x_y[0].size();

    vector< vector<double> > result;
    for (int i = 0; i < col; i++){
        vector<double> temp;
        for (int j = 0; j < col; j++){
            double sum = 0;
            for (int k = 0; k<lin; k++){
                sum += (x_y[k][i])*(x_y[k][j]);
            }
            temp.push_back(sum/(lin -1));
        }
        result.push_back(temp);
    }

    return result;
}
```

autovalores

Função para encontrar os autovalores da matriz de covariância.

```
vector <double> Matrix::autovalores(vector< vector<double> > mat){
    this->check_matrix_size(mat);

    int lin = mat.size(),
        col = mat[0].size();

    if (lin == 2 && col == 2){
        double a = 1,
            b = - (mat[0][0] + mat[1][1]),
            c = mat[0][0]*mat[1][1] - (mat[0][1] * mat[1][0]);

        double delta = pow(b,2) - 4*a*c;

        vector <double> result;

        result.push_back((-b - sqrt(delta))/2*a);
        result.push_back((-b + sqrt(delta))/2*a);

        return result;
    }

    if (lin == 3 && col == 3){
        double a = 1,

            b = - (mat[0][0]+mat[1][1]+mat[2][2]),

            c = (mat[0][0]*mat[1][1]) + (mat[0][0]*mat[2][2]) +
                (mat[1][1]*mat[2][2]) - (mat[0][2]*mat[2][0]) -
                (mat[1][2]*mat[2][1]) - (mat[0][1]*mat[1][0]),

            d = - (mat[0][0]*mat[1][1]*mat[2][2]) -
                  (mat[0][1]*mat[1][2]*mat[2][0]) -
                  (mat[0][2]*mat[1][0]*mat[2][1]) +
                  (mat[0][2]*mat[1][1]*mat[2][0]) +
                  (mat[0][0]*mat[1][2]*mat[2][1]) +
                  (mat[0][1]*mat[1][0]*mat[2][2]);

        double r, q, s, discrim, dum1 ,term1, r13, t;
        double x1, x2, x3;
        double x11=0, x22=0, x33=0;

        b /= a;
        c /= a;
        d /= a;

        q = (3*c - (b*b))/9;
        r = -(27*d) + b*(9*c - 2*(b*b));
        r /= 54;

        discrim = q*q*q + r*r;
        x1 = 0;
        term1 = (b/3.0);

        if (discrim > 0){
            cout<<"\n\nO discriminante e maior que zero\nRaizes imaginarias";
        } else if (discrim == 0){
            // Todas as raizes sao reais.
            r13 = (r < 0) ? -cbrt(-r) : cbrt(r);
            x1 = -term1 + 2.0*r13;
            x3 = x2 = -(r13 + term1);
```

```
            vector <double> result;

            result.push_back(x1);
            result.push_back(x2);
            result.push_back(x3);

            return result;

        } else if(discrim < 0) {
            // Raizes sao reais e distintas (q < 0)
            q = -q;
            dum1 = q*q*q;
            dum1 = acos(r/sqrt(dum1));
            r13 = 2.0*sqrt(q);
            x1 = -term1 + r13*cos(dum1/3.0);
            x2 = -term1 + r13*cos((dum1 + 2.0*M_PI)/3.0);
            x3 = -term1 + r13*cos((dum1 + 4.0*M_PI)/3.0);

            vector <double> result;

            result.push_back(x1);
            result.push_back(x2);
            result.push_back(x3);

            return result;
        }
    };

    cout << "\n\nError: Erro no calculo dos autovalores";
}
```

autovetores

Função para encontrar os autovetores da matriz de covariância.

```
vector< vector<double> > Matrix::autovetores(vector< vector<double> > mat,
                                             vector<double> autovalores){
    this->check_matrix_size(mat);

    int lin = mat.size(),
        col = mat[0].size();

    if (lin == 2 && col == 2){
        double a = mat[0][0], b = mat[0][1], c = mat[1][0], d = mat[1][1];
        vector< vector<double> > result (2,vector<double>(2));

        result[0][0] = 1;
        result[1][0] = -c / (d-autovalores[0]);

        result[0][1] = -b / (a-autovalores[1]);
        result[1][1] = 1;

        return result;
    }

    if (lin == 3 && col == 3){
        double a = mat[0][0], b = mat[0][1], c = mat[0][2],
               d = mat[1][0], e = mat[1][1], f = mat[1][2],
               g = mat[2][0], h = mat[2][1], i = mat[2][2];
```

```
        vector< vector<double> > result (3,vector<double>(3));


        for (int i=0; i<autovalores.size(); i++){

            result[0][i] = (b*f - c*(e-autovalores[i]))/((e-autovalores[i]) *
                            (a-autovalores[i]) - b*d);
            result[1][i] = (-d*result[0][i] -f) / (e-autovalores[i]);
            result[2][i] = 1;
        }

        return result;
    };

    cout << "\n\nError: Erro no calculo dos autovetores";
}
```

pca

Função que imprime da tela todos os outputs do método de Principal Component Analysis.

```
void Matrix::pca(vector< vector<double> > X, vector< vector<double> > y,
                bool exclude_first_col){
    if (exclude_first_col==true)
        X = this->exclude_first_col(X);

    vector <double> mean_vector = this->data_mean(X, y);

    vector< vector<double> > x_y = this->concatenate_col(X, y);

    int lin_x_y = x_y.size(),
        col_x_y = x_y[0].size();

    vector< vector<double> > new_x_y= x_y;

    for (int j=0; j<col_x_y; j++)
        for (int i=0; i<lin_x_y; i++)
            new_x_y[i][j] -= mean_vector[j];

    vector< vector<double> > covariance = this->covariance(new_x_y);

    vector <double> autovalores = this->autovalores(covariance);

    vector< vector <double> > autovetores =
        this->autovetores(covariance, autovalores);

    cout<<"\nMedias\n";
    for (int i = 0; i<mean_vector.size();i++){
        cout<<mean_vector[i]<<"\t";
    }
    cout<<"\n\n";

    this->show_matrix(covariance, "Matrix de covariancia");

    cout<<"\n\nAutovalores\n";
    for (int i = 0; i<autovalores.size();i++){
        cout<<autovalores[i]<<"\t";
    }
    cout<<"\n\n";
```

```
    this->show_matrix(autovetores, "Autovetores");
    cout<<"\n\n";

    this->show_matrix(this->multiplicacao_matrizes(covariance,autovetores),
        "Matrix de covariancia * Autovetores");
    cout<<"\n\n";

    vector< vector<double> > temp = autovetores;

    for (int i = 0; i<temp.size(); i++){
        for (int j = 0; j<temp[0].size(); j++){
            temp[i][j] *= autovalores[j];
        }
    }

    this->show_matrix(temp, "Autovalores * Autovetores");
};
```

Testes

Para testar o funcionamento das funções foram utilizados os seguintes conjuntos de dados:

- Height x Shoe size;
- Boiling point at the Alps;
- Books x Grades;
- US Census.

Resultados

Height x Shoe size

Médias

$$\mu = \begin{bmatrix} 68.9 \\ 9.8 \end{bmatrix}$$

Matriz de covariância.

$$C = \begin{bmatrix} 10.3222 & 5.31111 \\ 5.31111 & 4.45556 \end{bmatrix}$$

Autovalores

$$x = \begin{bmatrix} 1.32157 \\ 13.4562 \end{bmatrix}$$

Autovetores

$$\lambda = \begin{bmatrix} 1 & 1.69468 \\ -1.69468 & 1 \end{bmatrix}$$

Conferência

$$\begin{bmatrix} 10.3222 & 5.31111 \\ 5.31111 & 4.45556 \end{bmatrix} \begin{bmatrix} 1 & 1.69468 \\ -1.69468 & 1 \end{bmatrix} = \begin{bmatrix} 1.32157 & 0 \\ 0 & 13.4562 \end{bmatrix} \begin{bmatrix} 1 & 1.69468 \\ -1.69468 & 1 \end{bmatrix}$$

$$\therefore \begin{bmatrix} 1.32157 & 22.804 \\ -2.23964 & 13.4562 \end{bmatrix} = \begin{bmatrix} 1.32157 & 22.804 \\ -2.23964 & 13.4562 \end{bmatrix}$$

Boiling point at the Alps

Médias

$$\mu = \begin{bmatrix} 202.953 \\ 25.0588 \end{bmatrix}$$

Matriz de covariância.

$$C = \begin{bmatrix} 33.1739 & 17.3464 \\ 17.3464 & 9.12111 \end{bmatrix}$$

Autovalores

$$x = \begin{bmatrix} 0.0398992 \\ 42.2551 \end{bmatrix}$$

Autovetores

$$\lambda = \begin{bmatrix} 1 & 1.91014 \\ -1.91014 & 1 \end{bmatrix}$$

Conferência

$$\begin{bmatrix} 33.1739 & 17.3464 \\ 17.3464 & 9.12111 \end{bmatrix} \begin{bmatrix} 1 & 1.91014 \\ -1.91014 & 1 \end{bmatrix} = \begin{bmatrix} 0.0398992 \\ 42.2551 \end{bmatrix} \begin{bmatrix} 1 & 1.91014 \\ -1.91014 & 1 \end{bmatrix}$$

$$\therefore \begin{bmatrix} 0.0398992 & 80.7131 \\ -0.076213 & 42.2551 \end{bmatrix} = \begin{bmatrix} 0.0398992 & 80.7131 \\ -0.076213 & 42.2551 \end{bmatrix}$$

Books x Grades

Médias

$$\mu = \begin{bmatrix} 2 \\ 14.1 \\ 63.55 \end{bmatrix}$$

Matriz de covariância.

$$C = \begin{bmatrix} 2.05128 & 2.71795 & 11.7692 \\ 2.71795 & 18.2974 & 34.4564 \\ 11.7692 & 34.4564 & 279.074 \end{bmatrix}$$

Autovalores

$$x = \begin{bmatrix} 284.063 \\ 1.42686 \\ 13.9335 \end{bmatrix}$$

Autovetores

$$\lambda = \begin{bmatrix} 0.0429869 & -33.3339 & -0.713884 \\ 0.130089 & 3.3279 & -7.45112 \\ 1 & 1 & 1 \end{bmatrix}$$

Conferência

$$\begin{bmatrix} 2.05128 & 2.71795 & 11.7692 \\ 2.71795 & 18.2974 & 34.4564 \\ 11.7692 & 34.4564 & 279.074 \end{bmatrix} \begin{bmatrix} 0.0429869 & -33.3339 & -0.713884 \\ 0.130089 & 3.3279 & -7.45112 \\ 1 & 1 & 1 \end{bmatrix} =$$

$$\begin{bmatrix} 284.063 \\ 1.42686 \\ 13.9335 \end{bmatrix} \begin{bmatrix} 0.0429869 & -33.3339 & -0.713884 \\ 0.130089 & 3.3279 & -7.45112 \\ 1 & 1 & 1 \end{bmatrix}$$

$$\therefore \begin{bmatrix} 12.211 & -47.563 & -9.94692 \\ 36.9535 & 4.74846 & -103.82 \\ 284.063 & 1.42686 & 13.9335 \end{bmatrix} = \begin{bmatrix} 12.211 & -47.563 & -9.94692 \\ 36.9535 & 4.74846 & -103.82 \\ 284.063 & 1.42686 & 13.9335 \end{bmatrix}$$

US Census

Médias

$$\mu = \begin{bmatrix} 1950 \\ 165.395 \end{bmatrix}$$

Matriz de covariância.

$$C = \begin{bmatrix} 1100 & 2227.83 \\ 2227.83 & 4599.6 \end{bmatrix}$$

Autovalores

$$x = \begin{bmatrix} 16.9492 \\ 5682.65 \end{bmatrix}$$

Autovetores

$$\lambda = \begin{bmatrix} 1 & 0.486145 \\ -0.486145 & 1 \end{bmatrix}$$

Conferência

$$\begin{bmatrix} 1100 & 2227.83 \\ 2227.83 & 4599.6 \end{bmatrix} \begin{bmatrix} 1 & 0.486145 \\ -0.486145 & 1 \end{bmatrix} = \begin{bmatrix} 16.9492 \\ 5682.65 \end{bmatrix} \begin{bmatrix} 1 & 0.486145 \\ -0.486145 & 1 \end{bmatrix}$$

$$\therefore \begin{bmatrix} 16.9492 & 2762.59 \\ -8.2398 & 5682.65 \end{bmatrix} = \begin{bmatrix} 16.9492 & 2762.59 \\ -8.2398 & 5682.65 \end{bmatrix}$$

Conclusão

Nesse relatório foi apresentado como se faz a Principal Component Analysis de um conjunto de dados.

Foi implementado, em linguagem c++, um algorítmo para executar esse método e, para testar o algorítimo, foram utilizados 4 conjuntos de dados diferentes.

Referências

[1] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Springer, 2001.*

[2] Lindsay I Smith. Tutorial on Principal Components Analysis. 2002.

[3] *Análise de componentes principais Disponível em (https://goo.gl/MmLrpv). Acesso em: 25 de out. de 2017*

[4] Covariância Disponível em (https://goo.gl/sEooY3). Acesso em: 25 de out. de 2017

[5] *Cálculo de autovalores e autovetores. Disponível em: (https://goo.gl/WxMZD2). Acesso em: 25 de out. de 2017*

[6] Ensino Superior: Álgebra Linear: Autovalores e autovetores. Disponível em: (https://goo.gl/1x6YBu) Acesso em: 25 de out. de 2017

[7] *Mean Vector and Covariance Matrix. Disponível em: (https://goo.gl/1ZkVnH) Acesso em: 25 de out. de 2017*