CENTRO UNIVERSITÁRIO FEI

VICTOR BIAZON

RA: 119.115-4

RELATÓRIO II – TÓPICOS ESPECIAIS DE APRENDIZAGEM
PRINCIPAL COMPONENT ANALYSIS

SÃO BERNARDO DO CAMPO

2019

Sumário:

1. Objetivo

Implementar o algoritmo análise de componentes principais para os datasets propostos e comparar com o resultado obtido previamente com a regressão linear.

2. Teoria

Análise de componente principal

A análise de componentes principais, ou Principal Componente Analysis, PCA, em inglês, tem como objetivo decompor dados em suas relações de covariância e uma pequena quantidade de combinações lineares para descrever os dados. PCA pode também ser utilizado para redução de dimensionalidade para fins de representação dos dados ou apenas para eliminação de variáveis não relevantes para as análises. O método foi inventado em 1901 por Karl Pearson [1].

O método primeiramente retira a média dos valores de cada uma das variáveis, extrai a matriz de covariância destes, calcula os autovalores e autovetores da matriz de covariância e por fim reconstrói os dados utilizando os autovetores mais relevantes para a análise. [2]

Os autovalores representam qual magnitude de maior variância dos dados enquanto os autovetores representam a direção de tal variância.

Os passos a serem seguidos para se executar a análise do PCA são os seguintes:

1. Adquirir os dados.
2. Retirar a média dos dados.
3. Calcular a matriz de covariância
4. Calcular os autovalores e auto vetores da matriz de covariância.
5. Escolher as componentes e formar um vetor de componentes.
6. Derivar os novos dados.

3. Implementação

A implementação partiu do seguinte fluxo:

Para implementação foram criadas as seguintes funções:

```
def Covariance(data): #calcula a matriz de covariancia dos dados
    Cov = np.zeros((len(data[0]),len(data[0])), dtype = float) #inicializa uma matriz para armazenar os dados
    for i in range(0, len(data[0])): #calcula para cada relação de colunas as variâncias de uma para a outra
        for j in range(i, len(data[0])):
            sum = 0
            for k in range(0, len(data)):
                sum +=  data[k,i] * data[k,j]
            Cov[i,j] = sum/(len(data)-1)
            Cov[j,i] = Cov[i,j] #como é uma matriz simétrica calcula apenas para [i,j] e replica para [j,i]
    return Cov


def EigenValues2Dim(CovMat): #calcula os autovalores de uma matrix de covariância 2x2
    #como a matriz de covariância é 2x2 neste caso, e acha os autovalores igualando o determinante  de (A -λI) a 0 e se acha as raízes por Bhaskara
    a = 1
    b = -(CovMat[0,0] + CovMat[1,1])
    c = CovMat[0,0] * CovMat[1,1] - CovMat[0,1] * CovMat[1,0]
    x1 = (-b + math.sqrt(b**2 - 4*a*c))/(2*a)
    x2 = (-b - math.sqrt(b**2 - 4*a*c))/(2*a)
    return x1, x2
```

```
        def EigenVector2Dim(CovMat, EigVal): #calcula os autovetores
de uma matriz de covariancia 2x2
        #    [a b][x] = λ * [x]  --->>>   ax + by = λx      --->>>   (a - λ)x + by = 0
        #    [c d][y]       [y]          cx + dy = λy                cx + (d - λ)y = 0
            EigVect = np.zeros((2,2), float)
            x = 1
            for i in range(0,2): #calcula os autovetores achando a
relação entre X e Y pelo sistema linear
                y = (CovMat[0,0] - EigVal[i])/(- CovMat[0,1]) * x
                EigVect[0,i] = x
                EigVect[1,i] = y
            return EigVect
```

Para executar os cálculos do PCA foi utilizada a seguinte função:

```
def PCA(data):

    Mean = np.mean(data) #calcula medias dos valores das colunas do
dataset
    data = np.asarray(data)
    M_data = np.copy(data)
    M_data[:,0] = data[:,0] - Mean.BPt #retira media da primeira
coluna
    M_data[:,1] = data[:,1] - Mean.Pressure #retira media da segunda
coluna

    Cov = Covariance(M_data) #calcula covariancia do dataset
    EigenValue = EigenValues2Dim(Cov) #calcula autovalores
    EigVect = EigenVector2Dim(Cov, EigenValue) #calcula autovetores
    MatEigen = OrganizeEigen(EigenValue, EigVect) #organiza
autovalores e vetores em ordem crescente de relevancia para analise
    FeatureVector = MatEigen[1:,:]
    FinalData = np.matmul(np.transpose(FeatureVector),
np.transpose(M_data))
```

```
    M_data_pred_x = np.linspace(-10,10,20) * EigVect[0,0] #gera
linha da PC1
    M_data_pred_y = M_data_pred_x * EigVect[1,0]
    M_data_pred_x2 = np.linspace(-7,7,20) * EigVect[0,1] #gera linha
da PC2
    M_data_pred_y2 = M_data_pred_x2 * EigVect[1,1]
```

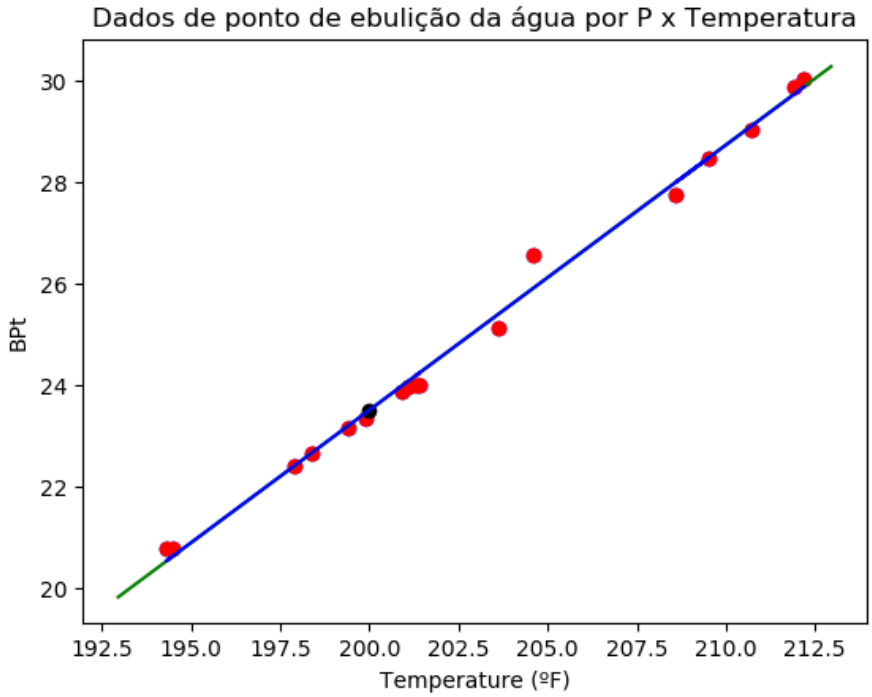
4. Resultados

Para testar os algoritmos foram implementados em três datasets:

A comparação entre a reta gerada pela PC1 do PCA e a regressão linear seguem abaixo para os três datasets.

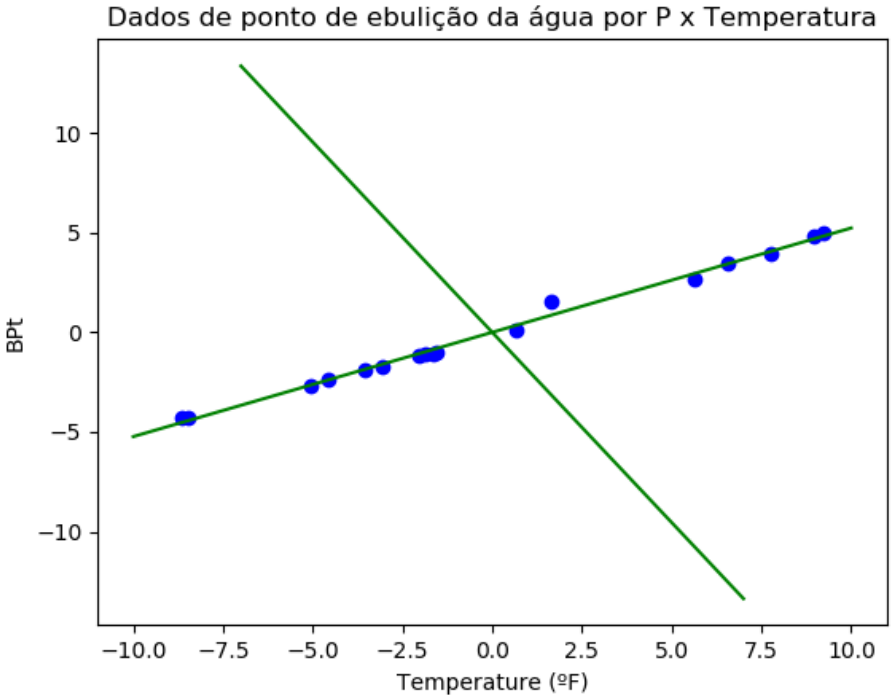
A comparação entre a reta gerada pela PC1 do PCA e a regressão linear seguem abaixo para os três datasets.

AlpsWater:

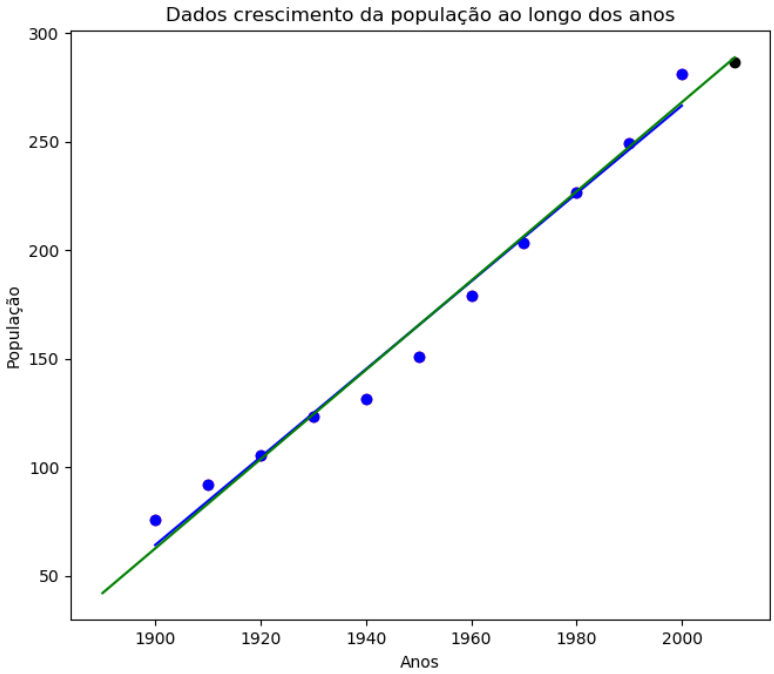


Como pode se notar, as duas linhas ficam exatamente uma em cima da outra, mostrando que para este dataset a regressão linear e a primeira componente são extremamente semelhantes.

Segue as duas componentes do Alpswater plotadas:

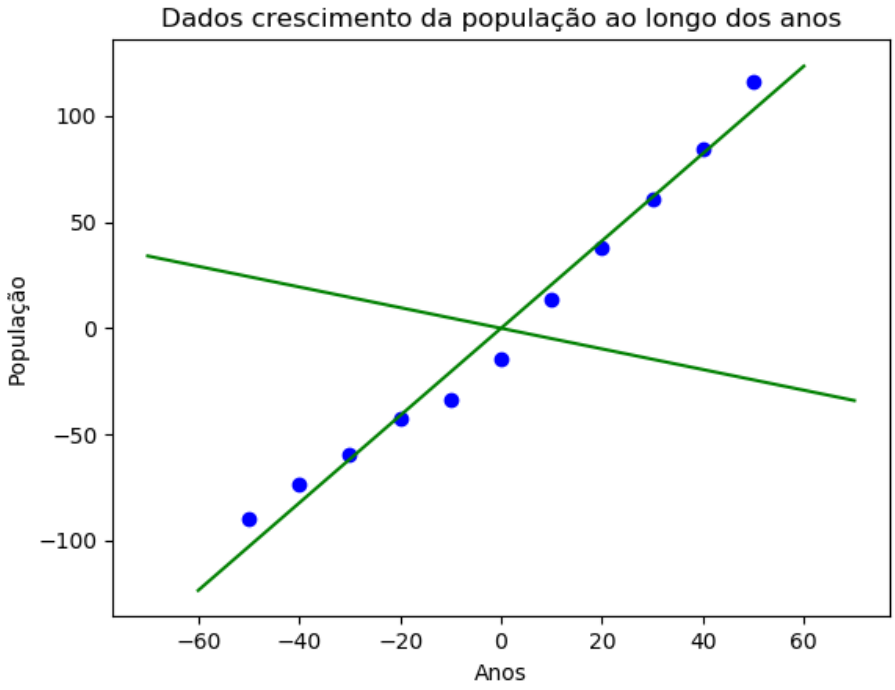


US Census:



Para o dataset do US Census foi possível notar que apesar de muito semelhantes as retas têm uma inclinação levemente diferente.

Abaixo o gráfico da representação das duas PCs do US censos.



Books – Attend vs Grades:

Para o dataset em questão não foi possível plotar de forma inteligível os dois gráficos no mesmo plano. Segue abaixo para fins de comparação os dois resultados:

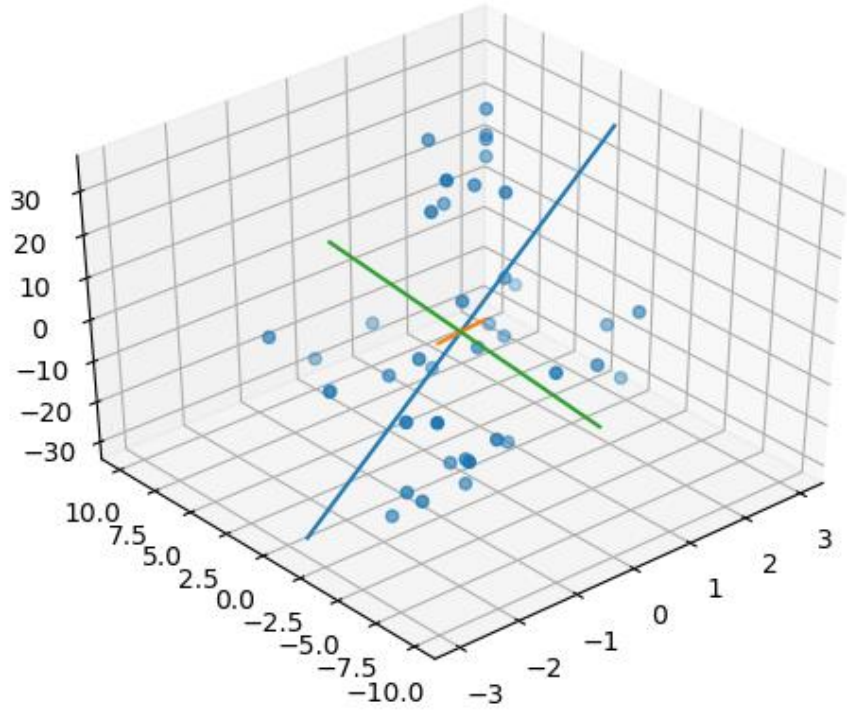
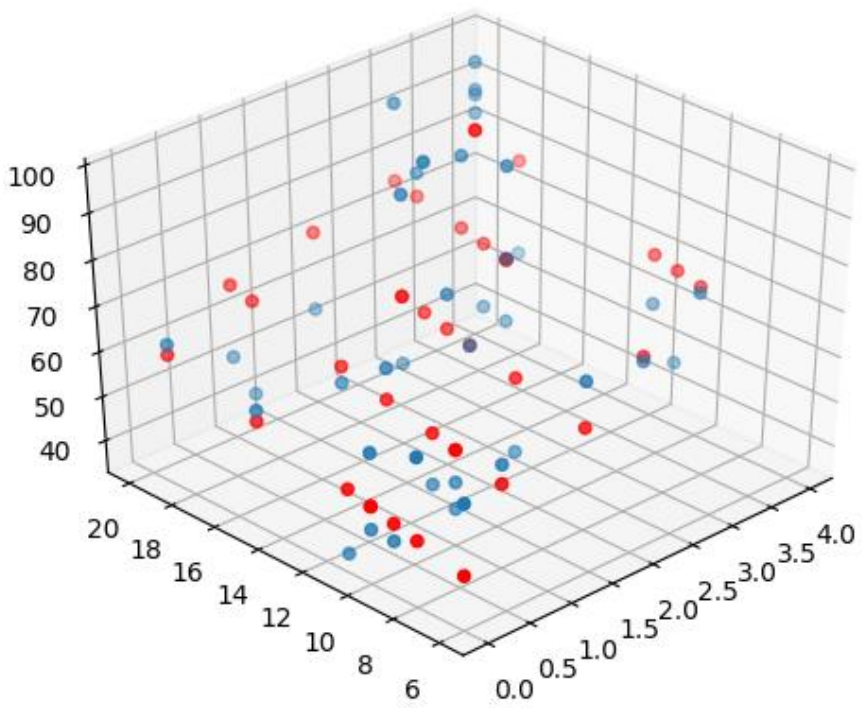


Gráfico dos pontos no espaço com as 3 componentes do PCA ilustrando a variância e os sentidos das mesmas.

Abaixo Plano da regressão linear do mesmo dataset.



5. Conclusão

Com estes experimentos pode-se verificar que as análises das componentes principais dos dados podem ser benéficas em eu entendimento e viabilizar a redução de dimensionalidades dos mesmos. Também foi possível traçar uma comparação entre a regressão linear e o PCA sendo verificado que a componente principal mais relevante da análise é sempre parecida com o resultado da regressão linear.

6. Referências bibliográficas

[1] Jolliffe I.T. **Principal Component Analysis**, Series: Springer Series in Statistics, 2nd ed., Springer

[2] Lindsay I Smith, **A tutorial on Principal Components Analysis**, 2002

Disponivel em: <ro.umontreal.ca/~pift6080/H09/documents/papers/pca_tutorial.pdf> Acesso: 06/11/2019