Redução de Dimensionalidade

1. Introdução

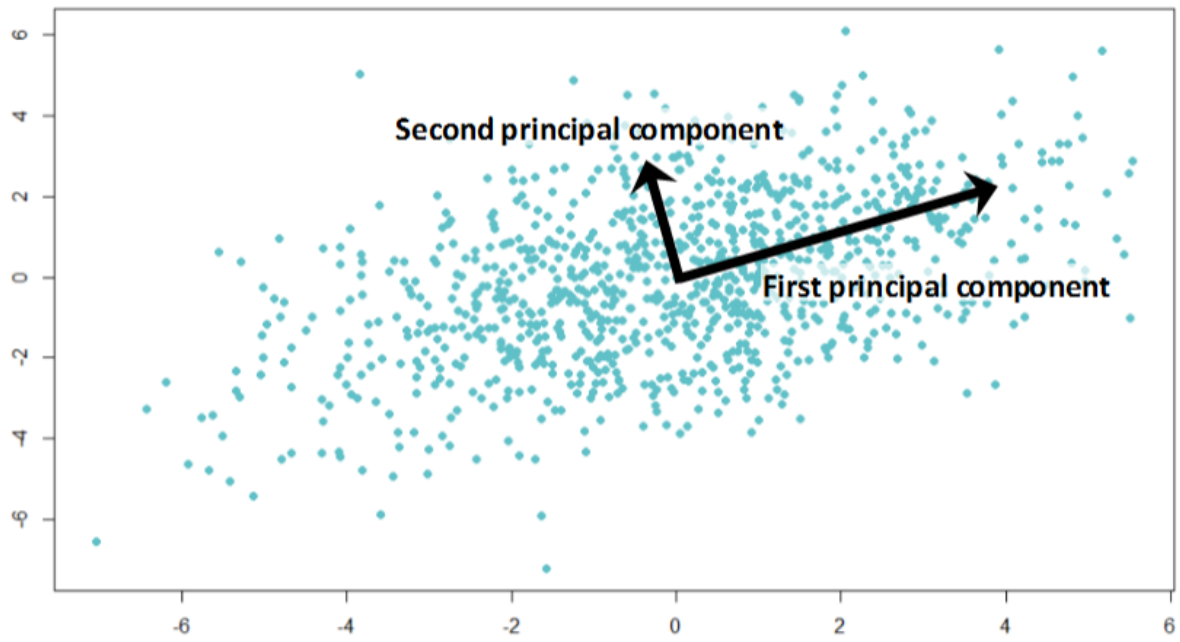
Com o avanço tecnológico e com aumento gradativo da geração e armazenamento de dados, os conjuntos de dados a serem trabalhados por modelos de *Machine Learning* e *Data Science* estão cada vez maiores e mais complexos, implicando também em um aumento gradativo do poder e tempo de processamento de máquinas, sejam elas locais ou em processamento em nuvem.

Mas com o objetivo de minimizar estes impactos, existem algumas técnicas complementares visando auxiliar e diminuir esta carga de processamento. Uma dessas técnicas é a **redução de dimensionalidade**, uma técnica bastante usada em conjunto de dados, normalmente grandes, com o objetivo de aumentar a interpretabilidade dos dados ao diminuir sua dimensionalidade (número de *features*), minimizando a quantidade de informação perdida no processo.

Uma das principais técnicas de redução de dimensionalidade é a **Análise de Componente Principal (PCA)**, técnica esta que será detalhada a seguir.

2. Análise de Componente Principal (PCA)

A Análise de Componente Principal (*Principal Component Analysis* em inglês) é a técnica para reduzir a dimensionalidade de um conjunto de dados, aumentando a interpretabilidade concomitante à minimização da perda de informação. Isso é feito criando novas variáveis não correlacionadas, preservando o máximo de variabilidade possível. O processo matemático por trás disso consiste em uma transformação linear buscando calcular os **autovetores** e indicando as direções principais de variabilidade do conjunto de dados.



Fonte: Analytic Vidhya

O PCA gera um novo conjunto de dados ainda com n variáveis (*features*), mas agora não correlacionadas. Estas variáveis não correlacionadas são denominadas *componentes principais*. A redução da dimensionalidade do conjunto de dados deve-se a justamente escolher a quantidade de componentes principais a serem utilizadas. Tipicamente, escolhemos uma quantidade m ($m < n$) de componentes principais que representam a variabilidade dos dados.

Algumas das vantagens desse processo é o ganho de uma **interpretação gráfica dos dados** minimizando a perda de informação e também um processo interessante para utilizar em testes de modelo onde seria necessário utilizar um **conjunto de dados muito grande**, pois testa-se o modelo com poucas variáveis mas sem perder o valor e a variabilidade dos dados originais.

2.1 Cálculo Matemático para o PCA

O processo matemático por trás do PCA utiliza de conceitos fundamentais da Álgebra Linear como a determinação dos **autovalores** e **autovetores**. Ou seja, dado uma matriz R de dimensão n representativa do conjunto de dados, para determinar os **autovalores** a partir da relação a seguir:

$$det[R - \lambda I] = 0$$

Onde I é a matriz identidade de dimensão n e o vetor λ representa as raízes desta equação característica, no caso os **autovalores**. A determinação da matriz R representativa do conjunto de dados, passa por duas transformações, onde a partir do conjunto de dados

originais X com dimensão n, calcula-se a matriz da covariância de X, nomeada como S. Portanto dada a matriz X:

$$X = \begin{bmatrix} x_{11} & x_{12} & x_{13} & \cdots & x_{1n} & x_{21} & x_{22} & x_{23} & \cdots & x_{2n} \\ \vdots & \vdots & \vdots & \cdots & \vdots & x_{N1} & x_{N2} & x_{N3} & \cdots \end{bmatrix}$$

Determina-se a matriz das covariâncias S:

$$S = \begin{bmatrix} S_{x_1}^2 & S_{x_1,x_2} & S_{x_1,x_3} & \cdots & S_{x_1,x_n} & S_{x_2,x_1} & S_{x_2}^2 & S_{x_2,x_3} & \cdots & S_{x_2,x_n} \\ \vdots & \vdots & \vdots & \cdots & \vdots & S_{x_n,x_1} \end{bmatrix}$$

Onde os cálculos de **variância** e **covariância** visto na estatística, são definidos a partir das equações abaixo:

$$ \begin{array}{ccc} S^2_{x_j} = \frac{\sum_{i=1}^{N}(x_{ij}-\bar{x}_{j})}{N-1}, & S_{x_{j1}, x_{j2}} = \frac{\sum_{i=1}^N(x_{ij1}-\bar{x_{j1}})(x_{ij2}-\bar{x_{j2}})}{N-1} \end{array} $$

Segundo passo seria definir a matriz R também conhecida como matriz das correlações, onde será aplicada a fórmula da correlação definida a seguir:

$$r(x_{j1}, x_{j2}) = \frac{S_{x_{j1},x_{j2}}}{S_{xj1}^2 S_{xj2}^2}$$

Onde a matriz resultante desta transformação será a matriz R:

$$R = \begin{bmatrix} 1 & r(x_1 x_2) & r(x_1 x_3) & \cdots & r(x_1 x_n) & r(x_2 x_1) & 1 & r(x_2 x_3) & \cdots & r(x_2 x_n) \\ \vdots & \vdots & \vdots & \cdots & \vdots & r \end{bmatrix}$$

Definido a matriz R e respectivamente a partir da equação dos autovalores, é possível agora determinar os **autovetores** a partir da equação a seguir:

$$R\tilde{a}_i = \lambda_i \tilde{a}_i$$

Sendo o autovetor $\tilde{a}_i$ relacionado para cada autovalor λ_i da forma:

$$ \tilde{a}_i = \left[\begin{array}{c} a_{i1} \\ a_{i2} \\ \vdots \\ a_{ip} \end{array} \right] $$

No caso dos autovetores, é importante que eles sejam normalizados, dessa forma garante-se que a norma L2 (também conhecida como **Norma Euclidiana**) seja igual a 1:

$$a_i = \frac{\tilde{a}_i}{|\tilde{a}_i|}$$

Determinados os autovetores, pode agora determinar todas as componentes principais, sendo estas as direções de maior variância do conjunto de dados após a transformação. Qualquer componente principal é definida como a **combinação linear** dos autovetores, lembrando que uma importante propriedade dos autovetores é que eles sejam **ortogonais** entre si:

$$PC_i = a_{i1}x_1 + a_{i2}x_2 + a_{i3}x_3 + \cdots + a_{in}x_n$$

2.2 Implementação em *Python*

A implementação em *Python* para o PCA é dada pelo bloco de código abaixo:

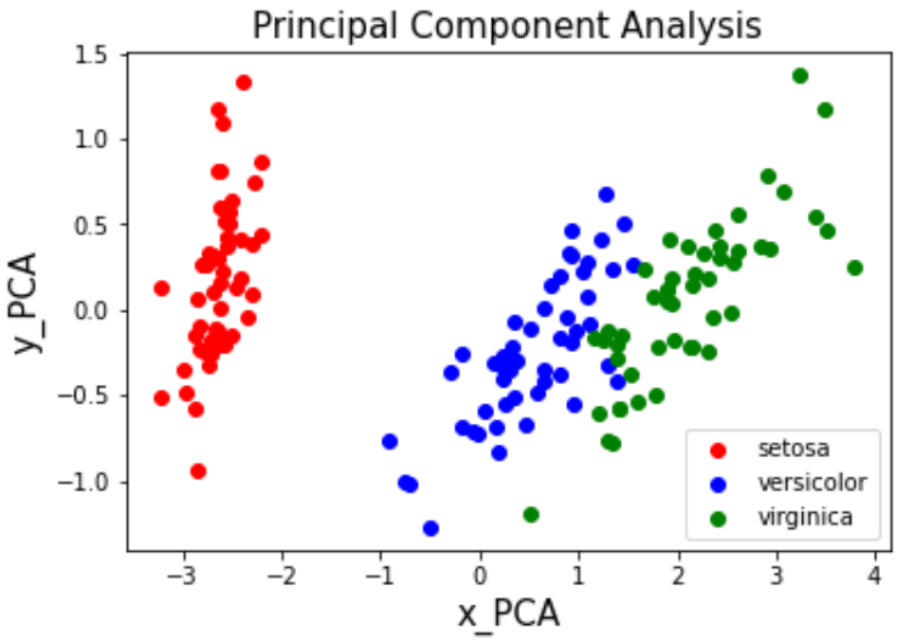
```
# Carregando o PCA do Scikit-Learn
from sklearn.decomposition import PCA

# Instanciando o PCA
pca = PCA(n_components = 2,  # Quantidade de componentes que serão
utilizadas
          random_state = 42) # Semente Aleatório

# Transforma os dados e cria o número de componentes necessários
X_pca = pca.fit_transform(X)
```

Exemplo Prático

No exemplo a seguir, utilizando o dataset `iris`, converteremos os 4 atributos disponíveis em duas componentes principais, e geramos um gráfico para ilustrar:

```
# Carregando as principais bibliotecas
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

# Carregando a base de dados da iris
iris = sns.load_dataset('iris')

# Separa o conjunto de dados entre X e y
X = iris.drop(['species'], axis = 1).values
y = iris['species'].values

# Carrega a função do PCA
from sklearn.decomposition import PCA

# Instancia o PCA
pca = PCA(n_components = 2,   # Quantidade de componentes principais
          random_state = 42)  # Semente aleatória

# Transforma os dados com o PCA
X_pca = pca.fit_transform(X)

# Transforma os dados resultantes em um dataset
iris_PCA = pd.DataFrame(X_pca, columns = ['x_PCA', 'y_PCA'])

# Plot das componentes do PCA e a marcação das espécies
# Define a quantidade de classes
class_labels = np.unique(y)

# Cores para o gráfico
colors = ['red', 'blue', 'green']

# Inicializa variável auxiliar
aux = 0

# Laço para gerar todos os scatterplots
for c in class_labels:
    # Identifica os elementos
    ind = np.where(y == c)
    # Gera o plot do gráfico
    plt.scatter(iris_PCA['x_PCA'].iloc[ind],
iris_PCA['y_PCA'].iloc[ind], color = colors[aux], label = c)
    # Aumenta a variável auxiliar
    aux = aux + 1
# Define nomes para os eixos e título
```

```
plt.xlabel('x_PCA', fontsize=15)
plt.ylabel('y_PCA', fontsize=15)
plt.title('Principal Component Analysis', fontsize=15)
# Define uma legenda
plt.legend()
# Mostra o gráfico
plt.show()
```



Fonte: Autoria própria

A partir da visualização de apenas duas componentes, nota-se que existe uma separação clara entre as espécies da `iris` e estão bem identificadas as direções de mais variância em cada uma delas.

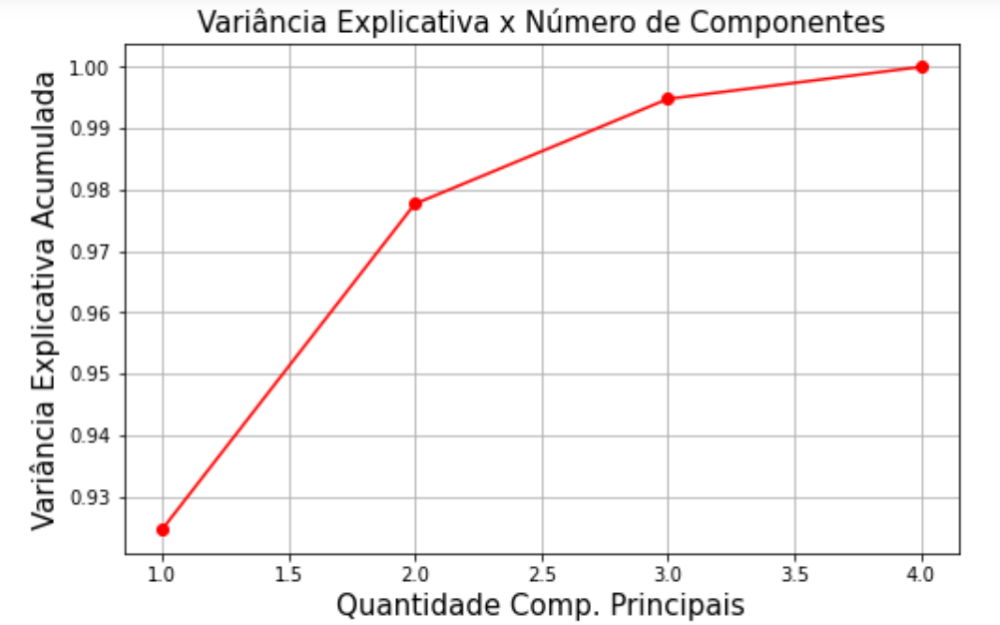
3. Definição da Quantidade de Componentes

Ao avaliarmos uma redução de dimensionalidade é importante que utilizemos a menor quantidade de componentes principais que representam a maior parte da variabilidade dos dados, para que não percamos muita informação durante o processo. Para avaliar isto, deve-se montar um gráfico da **variância explicativa acumulada** de acordo com a quantidade de componentes principais utilizadas pelo modelo, conforme o código abaixo:

```
# Carregando as principais bibliotecas
import pandas as pd
import numpy as np
```

```
import matplotlib.pyplot as plt
import seaborn as sns

# Carregando a base de dados da iris
iris = sns.load_dataset('iris')

# Separa o conjunto de dados entre X e y
X = iris.drop(['species'], axis = 1).values
y = iris['species'].values

# Carrega a função do PCA
from sklearn.decomposition import PCA

# Instancia e faz o treinamento do PCA
pca = PCA(random_state = 42).fit(X)

#Define uma figura para quantidade de componentes necessárias
plt.figure(figsize=(8,5))

# Calcula o máximo de componentes
n_components = np.arange(1, np.shape(X)[1] + 1)

# Define o gráfico com a variância explicativa acumulada
plt.plot(n_components, np.cumsum(pca.explained_variance_ratio_), 'ro-')

# Nome para os eixos e título para o gráfico
plt.xlabel('Quantidade Comp. Principais', fontsize = 15)
plt.ylabel('Variância Explicativa Acumulada', fontsize = 15)
plt.title('Variância Explicativa x Número de Componentes', fontsize =
15)

# Cria uma grade para o gráfico
plt.grid(True)
# Mostra o gráfico
plt.show()
```

O resultado para este código é que, com apenas duas componentes, já representa 98% da variância explicativa dos dados.



Fonte: Autoria própria

Materiais Complementares

Canal *StatQuest*, vídeo sobre *PCA Step-by-Step*;

Documentação no Scikit-Learn sobre o PCA;

Artigo publicado por Chaitanyanarava no *Analytics Vidhya* - *A Complete Guide On Dimensionality Reduction*.

Referências

James, Gareth, et al. An Introduction to Statistical Learning: With Applications in R. Alemanha, Springer New York, 2013;

Bruce A., Bruce P. Estatística Prática para Cientistas de Dados. Segunda Edição, Alta books, 2019;

Tópico anterior

Próximo Tópico