Pedro Blöss Braga

July 2020

Aplicações e implementação da Decomposição de Valores Singulares (SVD), e relações com Análise de Componentes Principais (PCA)

Contents

1 Introdução

Neste texto pretendo abordar a teoria o método de decomposição de valores (Singular Value Decomposition), conhecido como SVD; sua relação forte com o método de decomposição em compontentes principais (Principal Component Analysis), PCA; expor algumas aplicações interessantes destes métodos; Fazer testes de comparação entre métodos.

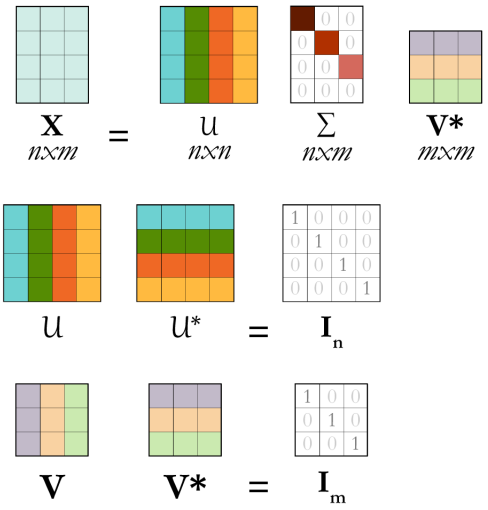
2 Decomposição em Valores Singulares (SVD)

Este método pode ser utilizado para decompor matrizes complexas, separando a matriz original em duas matrizes unitárias e uma retangular diagonal, de valores singulares:

$$X = \mathcal{U}\Sigma V^T \tag{1}$$

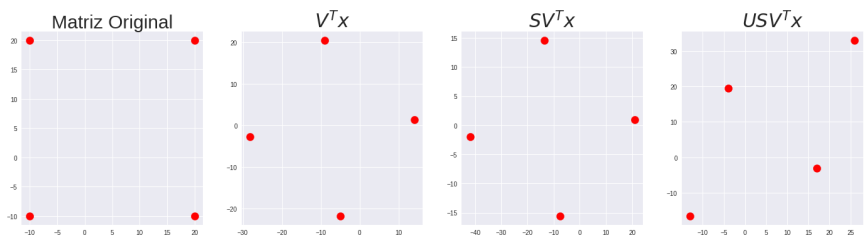
Caso trabalhando com uma matriz X real, as matrizes $\mathcal{U}$ e Σ também serão reais, e as conjugadas transpostas serão então simplesmente as transpostas.

$$\forall x_{ij} \in \mathbb{R} \Rightarrow u_{ij}, v_{ij} \in \mathbb{R}, \forall u_{ij} \in \mathcal{U}, v_{ij} \in V \Rightarrow \bar{X}^T = X^T \tag{2}$$



Exemplo, com a matriz:

$$\begin{pmatrix} -10 & -10 \\ -10 & 20 \\ 20 & 20 \\ 20 & -10 \end{pmatrix} \tag{3}$$



3 Interpretação Geométrica do SVD

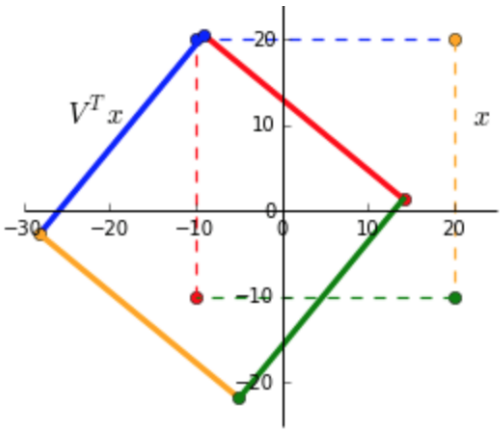
Ainda com a matriz bidimensional

$$X = \begin{pmatrix} -10 & -10 \\ -10 & 20 \\ 20 & 20 \\ 20 & -10 \end{pmatrix} \tag{4}$$

visualizaremos as transformações, geometricamente.

$\mathbf{V^T x}$

Podemos ver que multiplicar por V^T aplica uma transformação e uma rotação à matriz de input X

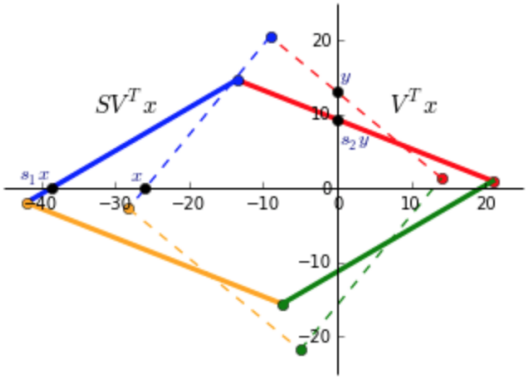


Vx_plot

$\mathbf{SV^T x}$

Agora, com S, a matriz passa a levar em conta valores na diagonal, escalando a matriz (multiplicando por algum valor).

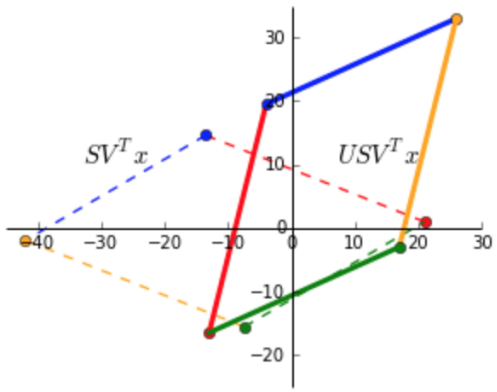
V rotaciona a matriz para uma posição em que os valores singulares representam o fator de escalamento sobre a base V. (Na foto, $V^T X$ está tracejado e $SV^T X$ está em linhas sólidas).



SVx_plot

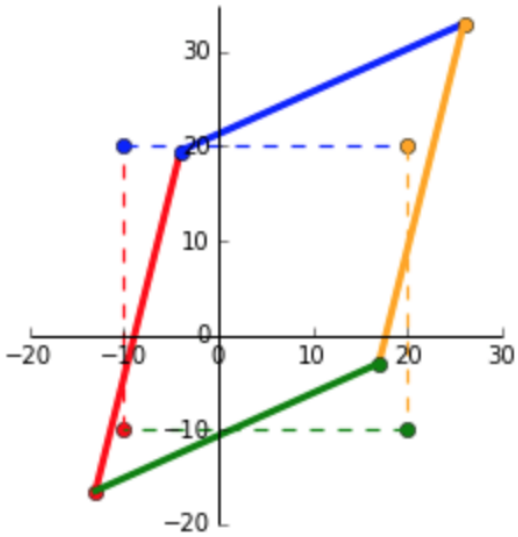
$\mathbf{USV^Tx}$

Por fim, U rotaciona e reflete a matriz de volta à base inicial.



USVx_plot

Demonstrando todos os passos em uma única imagem, obtemos:



transformed_plot

Podemos ver que os eixos no novo espaço são ortogonais. Então, os atributos originais são expressos em termos de novos atributos independentes entre si.

(Isso pode ter relação com a matriz de covariância, que expõe a independência entre componentes).

4 Análise de Componentes Principais (PCA)

Este método expõe eixos ortogonais não-correlacionados, denominados componentes principais (PC)s, no espaço n dimensional.

Os primeiros componentes capturam os dados de maior variância.

Já que os PCs são ortogonais entre si, i.e. $\langle PC_i, PC_j \rangle = 0, i \neq j$, um par de linhas perpendiculares no $\mathbb{R}^2$ podem forma dois PCs.

O primeiro PC, que captura a maior variância, é que melhor se ajusta aos dados, linearmente. Podemos pensar que é uma reta ajustada linear, $Y = \beta_0 + \beta_1 X + \epsilon$, que minimiza $\sum_i \epsilon^2$.

Os componentes principais podem ser obtidos pela decomposição da matriz de covariância:

$$C = W \Lambda W^{-1} \tag{5}$$

Geometricamente, esta decomposição encontra um novo sistema de coordenadas de autovetores de C por meio de rotações.

W é a matriz nxn de autovetores, e Λ é a matriz diagonal de n autovetores.

Os autovetores, que são os vetores coluna de W, são os PCs que procuramos.

Para obter os dados no espaço dos PCs, fazemos:

$$X_k = XW_k \tag{6}$$

5 Relação do SVD com o PCA

Ambos são métodos de decomposição de matrizes retangulares.

Podemos ver, pela decomposição SVD da matriz de covariância:

$$C = \frac{\langle X, X \rangle}{n-1} = \frac{V\Sigma\mathcal{U}^T U\Sigma^T V^T}{n-1} = V\frac{\Sigma^2}{n-1}V^T = V\frac{\Sigma^2}{n-1}V^{-1} \tag{7}$$

Então,

$$\Lambda = \frac{\Sigma^2}{n-1} \tag{8}$$

Pela decomposição de autovetores de C, podemos verificar que os autovalores serão a matriz Λ.

6 Redução de Dimensionalidade

Uma matriz de dados pode descrever o comportamento de algum fator com múltiplos valores, como por exemplo um vetor de preços de algum ativo da bolsa de valores, associado a um vetor de texto podem ser input para algum modelo preditivo.

Contudo, por vezes, gostaríamos de ter um número reduzido e compactado de informações, a fim de "explicar" algum comportamento de alguma maneira mais "simples". Nesse caso podemos tratar de redução de dimensionalidade.

Por exemplo, podemos iniciar de um conjunto de dados com n_1 variáveis reais: $\{x_1, ..., x_{n_1}\} = \{x_j\}_{j=1}^{n_1}$, e convertê-lo para um $\{z_i\}_{i=1}^{n_2}$, de n_2 variáveis reais, tal que $n_2 < n_1$.

Desta maneira, aplicaremos uma função ψ:

$$\psi : \mathbb{R}^{n_1} \to \mathbb{R}^{n_2} \{x_j\}_{j=1}^{n_1} \sim \psi(\{z_i\}_{i=1}^{n_2}) \tag{9}$$

Em PCA, inicialmente, procuramos o vetor que reconstrói os dados da matriz X mais aproximadamente, ou seja, queremos minimizar o erro de reconstrução, promovendo a sequência com melhores aproximações lineares aos dados, com posto $q \leq p$, dadas as observações $\{x_j\}_{j=1}^{N}$ em $\mathbb{R}^p$, o modelo

$$f(\lambda) = \mu + V_q \lambda \tag{10}$$

onde μ é o vetor de localização em $\mathbb{R}^p$, V_q é a matriz p x q com q vetores unitários ortogonais como colunas, e λ é o vetor de q parâmetros.

A figura abaixo expõe o caso em que $q = 2$:

O ajuste do modelo aos dados pelos mínimos quadrados corresponde a minimizar o erro de reconstrução:

$$\underset{\mu, \{\lambda_i\}, V_q}{min} \sum_{i=1}^{N} ||x_i - \mu - V_q \lambda_i||^2 \tag{11}$$

Então, podemos otimizar para μ e λ_i, para obter: (solução 1)

$$\bar{\mu} = \bar{x}, \hat{\lambda}_i = V_q^T (x_i - \bar{x}) \tag{12}$$

Então,

$$\underset{V_q}{min} \sum_{i=1}^{N} ||(x_i - \bar{x}) - V_q V_q^T (x_i - \bar{x})||^2 \tag{13}$$

Por conveniência, podemos assumir $\bar{x} = 0$. A matriz $V_q V_q^T = H_q$ p x p é a matriz de projeção, que mapeia cada ponto x_i para a reconstrução de posto q $H_q x_i$, a projeção ortogonal de x_i sobre o subespaço espandido pelas colunas de V_q.

A solução pode ser exprimida como segue.

Alocando as observações em uma matriz N x p X, podemos construir a **Decomposição de Valores Singulares**:

$$X = \mathcal{U} \Sigma V^T \tag{14}$$

Solução (1)

Encontraremos μ e V_q para minimizar

$$\underset{\mu,\{\lambda_i\},V_q}{min} \sum_{i=1}^{N} ||x_i - \mu - V_q\lambda_i||^2 \tag{15}$$

Derivamos com relação a μ:

$$\frac{\partial}{\partial\mu}(\sum_{i=1}^{N}\langle x_i - \mu - V_q\lambda_i, x_i - \mu - V_q\lambda_i\rangle) = -2\sum_{i=1}^{N}(x_i - \mu - V_q\lambda_i) = 0 \tag{16}$$

Então,

$$\frac{1}{N}\sum_{i=1}^{N}\mu = \frac{1}{N}\sum_{i=1}^{N}x_i - V_q\frac{1}{N}\sum_{i=1}^{N}\lambda_i\mu = \bar{x} - V_q(\frac{1}{N}\sum_{i=1}^{N}\lambda_i) \tag{17}$$

Derivando com relação a λ_i:

$$\frac{\partial}{\partial\lambda_i}(\sum_{i=1}^{N}\langle x_i - \mu - V_q\lambda_i, x_i - \mu - V_q\lambda_i\rangle) = -2V_q\sum_{i=1}^{N}(x_i - \mu - V_q\lambda_i) \tag{18}$$

$$= -2((x_i - \mu)^T V_q)^T + (V_q^T V_q + V_q V_q^T)\lambda_i = 0 \tag{19}$$

$$V_q^T(x_i - \mu) = V_q^T V_q\lambda_i = \lambda_i \tag{20}$$

(Pois $V_q^T V_q = I$, V_q é unitária).

Então, unindo à $\mu = \bar{x} - V_q(\frac{1}{N}\sum_{i=1}^{N}\lambda_i)$:

$$\mu = \bar{x} - V_q V_q^T(\bar{x} - \mu) \tag{21}$$

$$\therefore (I - V_q V_q^T)(\bar{x} - \mu) = 0 \tag{22}$$

$(I - V_q V_q^T)$ é a projeção ortogonal no subespaço estendido pelas colunas de V_q, então $\mu = \bar{x} + h$, $h \in \mathbb{R}^p$ (h está no subespaço estendido por V_q).

Como o posto de V_q é q, o vetor h é obtido de um espaço $p - q$. Tomando $h = 0$, temos que $\mu = \bar{x}$, e por fim:

$$\lambda_i = V_q^T(x_i - \bar{x}) \quad \square \tag{23}$$

7 Algoritmo para obtenção do SVD - Power Method

Para decompor uma matriz $A \in \mathbb{R}^{mxn}$, inicia-se computando o primeiro valor singular σ_1 e os vetores (da esquerda e da direita, respectivamente) u_1 e v_1 de A, para os quais $\underset{i>j}{min} log(\sigma_i/\sigma_j) \geq \lambda$:

1. Gerar x_0 tal que $x_0(i) \sim \mathcal{N}(0,1)$
2. $s \leftarrow log(4log(2n/\sigma)/\epsilon\sigma)/2\lambda$
3. Para $i \in \{1, ..., s\}$:
4. $x_i \leftarrow A^T A x_{i-1}$
5. $v_1 \leftarrow x_i/||x_i||$
6. $\sigma_1 \leftarrow ||Av_1||$
7. $u_1 \leftarrow Av_1/\sigma$
8. OUTPUT (σ_1, u_1, v_1)

Algoritmo, da referência 6:

2.2 Algorithm

Algorithm 2.2: Power Method

1. **Input :** A square matrix $A \in \mathbf{R}^{n \times n}$ and a vector $u^{(0)} \in \mathbf{R}^n$,
2. **Output :** The largest eigenvalue λ_1 and the associated eigenvector
3. for $k = 1, 2, \cdots$ (repeat until convergence)

$$w^{(k)} = Au^{(k-1)},\ u^{(k)} = \frac{w^{(k)}}{\|w^{(k)}\|},\ \lambda^{(k)} = u^{(k)T}\left(Au^{(k)}\right)$$

8 Implementação e testes - SVD

Implementando um método para obter de decomposição SVD, comparei com o método do numpy np.linalg.svd().

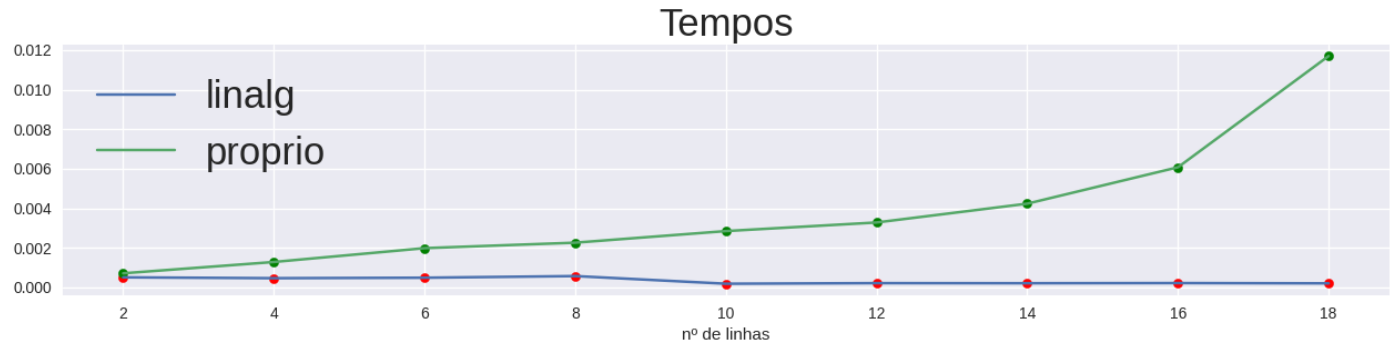
Como medida de comparação, foi gravado o tempo de processamento, e também uma norma erro escolhida.

Essa norma erro foi, dada a matriz original A decomposta, encontrou-se a matriz A_i^*, resultante da decomposição, no teste i:

$$A_i^* = \mathcal{U}_i \Sigma_i V_i^T \tag{24}$$

obtendo como métrica de desvio $\epsilon_i = ||det(A_i) - det(A_i^*)||_2$, e mais genericamente:

$$\bar{\epsilon} = \frac{1}{K} \sum_{i=1}^{K} (||det(A_i) - det(A_i^*)||_2) \tag{25}$$



	Média tempos	Média ($\|\|det(A) - det(A*)\|\|_2$)
linalg	0.000341	2.608744e-21
próprio	0.003820	3.655771e-21

O loop de testes para cada matriz foi implementado utilizando pool de threads como método de paralelismo. Desta maneira, os múltiplos processos de um loop podem ser executados simultaneamente, divididos de acordo com o número de threads e CPUs disponíveis.

As matrizes utilizadas foram geradas de uma sequência de matrizes quadradas, de tamanho múltiplo de 2, e de Hilbert - todo elemento da linha n e coluna m $a_{nm} = \frac{1}{n+m-1}$, no grupo:

$$\mathcal{M} = \{A/(a_{nm} = \frac{1}{n+m-1}, \forall a_{nm} \in A) \wedge (n, m \in \{j\}_{j=1}^{K}, K = |A|, K+1 \equiv 1 mod(2))\} \tag{26}$$

9 Compressão de Imagens - Aplicação do SVD

Por vezes, a compressão de imagens pode ser interessante para reduzir o módulo da quantidade de armazenamento computacional, e está é uma possível aplicação do SVD.

Retornando a matriz resultante da decomposição $\mathcal{U}\Sigma V^T$, mas selecionando uma quantia de componentes:

10 "Auto-faces" (Eigenfaces) - SVD

Primeiramente, vamos recordar que autovetores de uma matriz tem a interpretação de indicarem as direções de máxima variância dos dados. Isto é, a localização da informação mais "relevante".

$$X\lambda_i = v_i\lambda_i \Rightarrow \{\lambda_i\}_{i=1}^{N}, \{v_i\}_{i=1}^{N} \tag{27}$$

são, respectivamente, as sequências de autovalores e autovalores da matriz X.

Coletados os dados, podem ser encontrados os autovetores associados a cada imagem, exprimindo as direções de máxima variância, e autovalores.

Então, escolhem-se os K maiores autovelores

$$\{||\{\lambda_i\}_{i=1}^{N-K}||_\infty, ..., ||\{\lambda_i\}_{i=1}^{N}||_\infty\} = \{||\{\lambda_i\}_{i=1}^{N-j}||_\infty\}_{j=0}^{K} \tag{28}$$

Então diremos que este conjunto escolhido formará um espaço S

$$||\{\lambda_i\}_{i=1}^{N-j}||_\infty \in S, \forall ||\{\lambda_i\}_{i=1}^{N-j}||_\infty \in \{||\{\lambda_i\}_{i=1}^{N-j}||_\infty\}_{j=0}^{K} \tag{29}$$

Usaremos este espaço para projetar os dados, encontrando as "faces" mais "próximas", e podemos classificá-las, dicotômicamente, como faces conhecidas ou não.

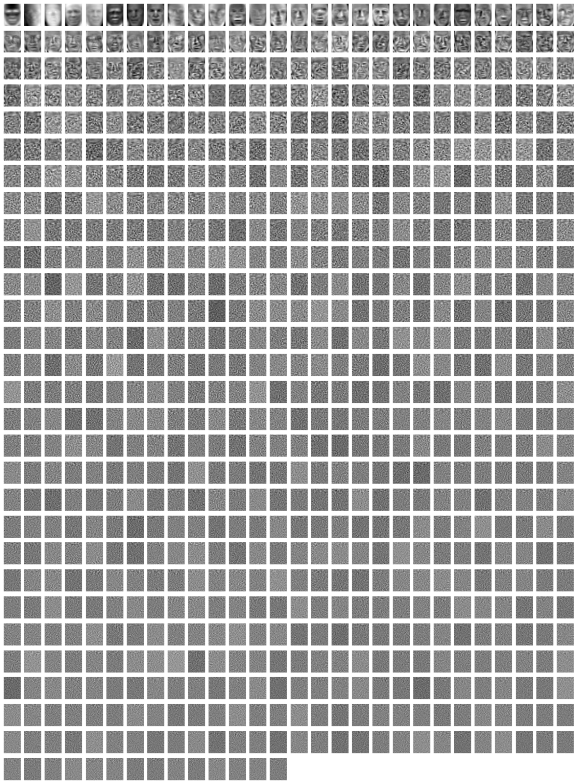
Utilizando o conjunto de dados **h lfw people** da biblioteca **sklearn.datasets** do python:

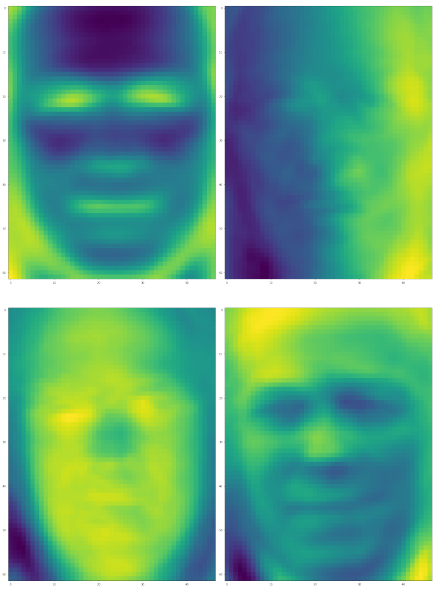
Após a decomposição SVD

$$X = \mathcal{U}\Sigma V^T \tag{30}$$

Podemos plotar os valores singulares

E então podemos plotar as "eigenfaces" resultantes:





Podemos ver que as imagens expõem características marcantes das faces, em concordância com a hipótese de grande variância dos autovetores.

Em seguida, vamos verificar as imagens no novo espaço:

E por fim, podemos aplicar um modelo de classificação para verificar a previsão dos nomes, associados a cada face, no novo espaço.

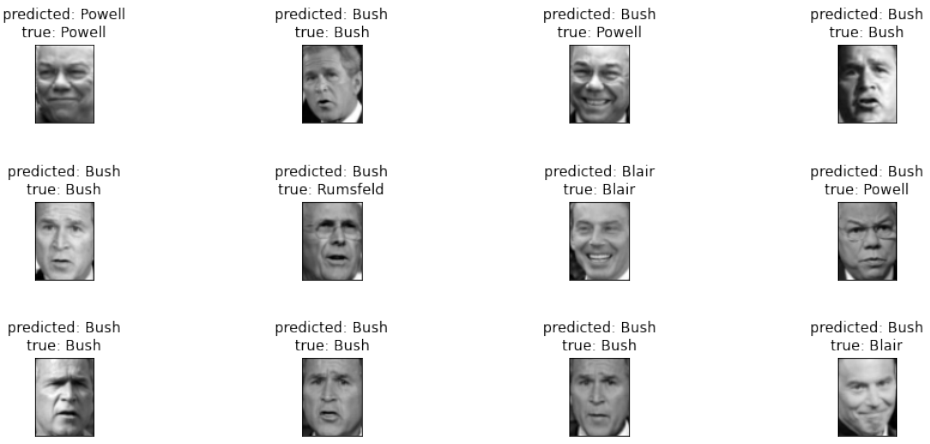
No caso, apliquei dois modelos: Support Vector Classifier e Random Forest Classifier, para escolher aquele de melhor performance.

Então obtive os resultados:

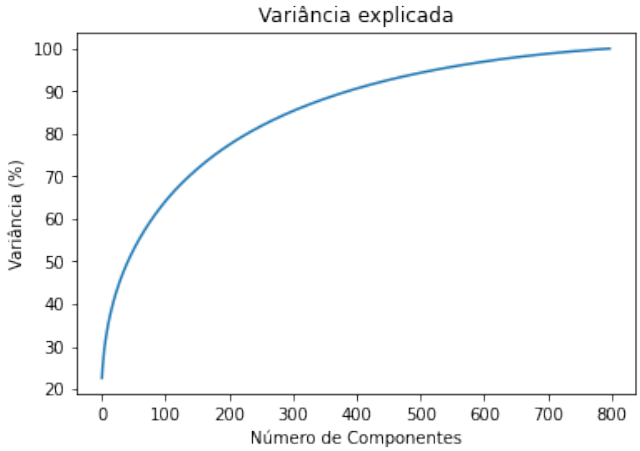
```
=======================================
Support Vector Classifier (MAPE = 25.646%)
=======================================
                   precision    recall  f1-score   support

     Colin Powell       0.95      0.54      0.68        71
  Donald Rumsfeld       0.00      0.00      0.00        36
    George W Bush       0.52      0.99      0.69       159
Gerhard Schroeder       0.00      0.00      0.00        33
       Tony Blair       0.00      0.00      0.00        43

         accuracy                           0.57       342
        macro avg       0.29      0.31      0.27       342
     weighted avg       0.44      0.57      0.46       342
```

```
=======================================
Random Forest Classifier (MAPE = 23.555%)
=======================================
                   precision    recall  f1-score   support

     Colin Powell       0.92      0.49      0.64        71
  Donald Rumsfeld       0.80      0.11      0.20        36
    George W Bush       0.55      0.98      0.70       159
Gerhard Schroeder       1.00      0.12      0.22        33
       Tony Blair       0.67      0.14      0.23        43

         accuracy                           0.60       342
        macro avg       0.79      0.37      0.40       342
     weighted avg       0.71      0.60      0.53       342
```

Comparando os nomes previstos e os reais, para o Random Forest (modelo de menor erro médio percentual:



Pelo gráfico de variância explicada, vemos a soma cumulativa dos vetores singulares, de acordo com o número de componentes.



11 Clusterização - Aplicação do PCA

Utilizando o famoso conjunto de dados Iris, com informações de sépalas e pétalas de 3 diferentes espécies de plantas, podemos aplicar a análise de componentes principais a fim de segmentar os dados de acordo com as espécies.

12 Kernel PCA

Neste exemplo, usa-se o Kernel PCA para encontrar uma projeção dos dados, tornando-os linearmente separáveis.

Kernels são amplamente utilizados em algoritmos de classificação e regressão.

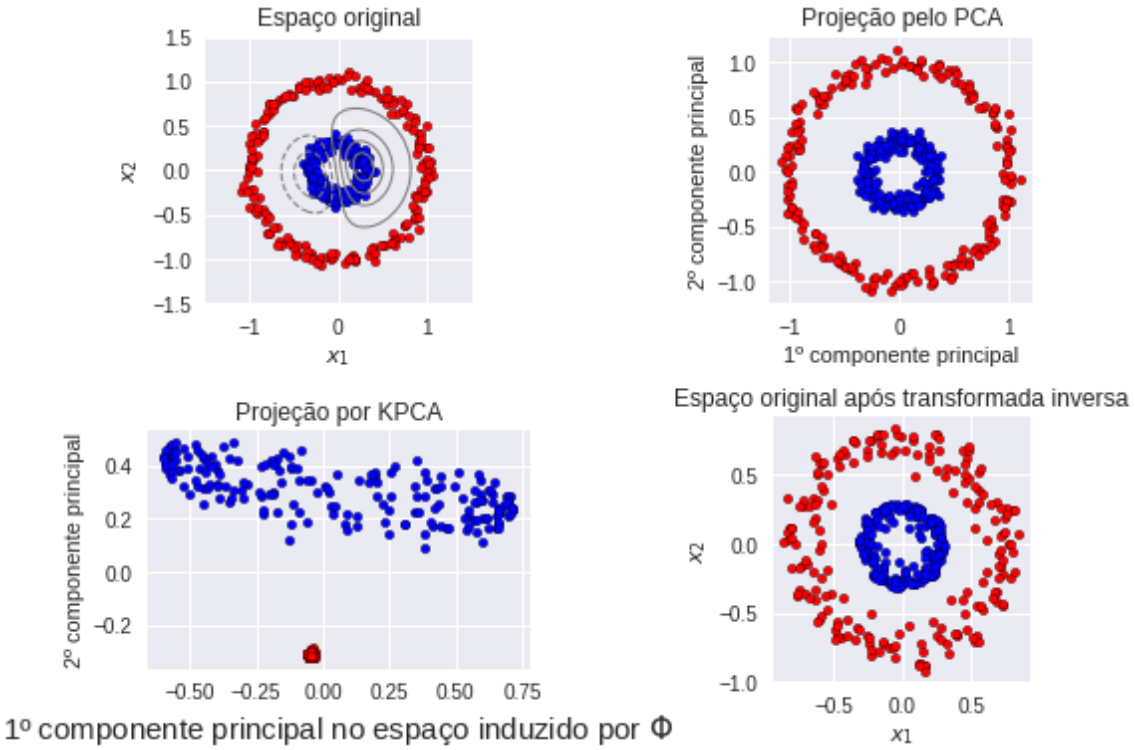
Utilizando o exemplo de Support Vector Machines (SVMs), em que, para classificação, os dados são separados linearmente, com bandas (uma faixa acima da reta e outra abaixo, equidistantes da reta, por um tamanho ϵ). Já para regressão, os dados devem ser ajustados entre duas retas, de maneira que maximize o agrupamento dos dados entre as faixas.

Muitas vezes, os dados estão em um espaço em que não são linearmente separáveis. Para atacar este problema, aplica-se uma transformação de espaço, para outro espaço em que um hiperplano gere as bandas.

A partir dos dados $\{x_j\}_{j=1}^N$ e um espaço S_1 em que não são linearmente separáveis, faz-se a transformação Φ, passando a sequência para o espaço S_2:

$$\Phi : S_1 \to S_2 \tag{31}$$

$$\Phi(\{x_j\}_{j=1}^N) = \{\hat{x}_j\}_{j=1}^N \tag{32}$$



13 Conceitos importantes

Conjugada transposta

É a transposta da matriz conjugada, obtida pelos conjugados complexos de todos elementos da matriz:

$$A_H := \bar{A}^T = \bar{A^T} \tag{33}$$

Matriz unitária

$$AA^{-1} = I \text{ , se } A \text{ é uma matriz real} \tag{34}$$

Matriz de Covariância Esta quantifica a variabilidade conjunta entre duas variáveis aleatórias X, Y :

$$cov(X, Y) = \mathbb{E}[(X - \mathbb{E}[X])(Y - \mathbb{E}[Y])] \tag{35}$$

Vale comentar que caso os valores esperados das duas variáveis aleatórias sejam nulos, a covariância será apenas o produto interno:

$$\mathbb{E}[X] = \mathbb{E}[Y] = 0 \Rightarrow Cov(X, Y) = \langle X, Y \rangle \tag{36}$$

Então, a matriz de covariância de uma dada matriz X será:

$$C = \frac{\langle X, X \rangle}{n-1} = \frac{X^T X}{n-1} \tag{37}$$

Matriz de Rotação

$$M_\theta = \begin{pmatrix} cos\theta & \sin\theta \\ -sin\theta & cos\theta \end{pmatrix} \tag{38}$$

Ao multiplicar uma matriz A pela M_{theta} rotaciono as coordenadas de A no ângulo θ.

Matriz de Hilbert

Uma matriz A é de Hilbert, se

$$a_{ij} = \frac{1}{i+j-1}, \forall a_{ij} \in A \tag{39}$$

14 Referências Principais

1. Richard L. Burden e J.Douglas Faires. "Numerical Analysis".

2. Strang, Gilbert. "Linear Algebra and its applications".

3. TREVOR, Hastie, TIBSHIRANI, Robert, FRIEDMAN, Jerome, "Elements of Statistical Learning: Data Mining, Inference, and Prediction", 2th edition.

4. sítio https://www.analyticsvidhya.com/blog/2019/08/5-applications-singular-value-decomposition-svd-data-science/

5. sítio https://www.cs.yale.edu/homes/el327/datamining2013aFiles/07$_s ingular_v alue_d ecomposition.pdf$

6. A. H. Bentbib and A.Kanber, "Block Power Method for SVD Decomposition", http://www.kurims.kyoto-u.ac.jp/EMIS/journals/ASUO/mathematics$/anale2015vol2/Bentbib_A.H._{K anber_A..pdf}$