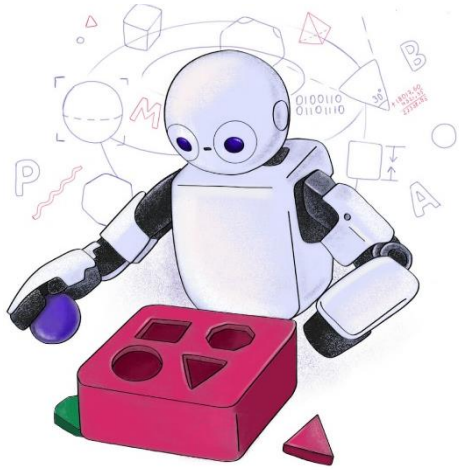
TP555 - Inteligência Artificial e Machine Learning:

Redução de Dimensionalidade



Inatel

Felipe Augusto Pereira de Figueiredo
felipe.figueiredo@inatel.br

Redução de Dimensionalidade

- Muitos problemas de aprendizado de máquina envolvem centenas, milhares ou até milhões de atributos para cada exemplo de treinamento.
- Isso não apenas torna o treinamento extremamente lento, mas também pode tornar muito mais difícil encontrar uma boa solução.
- Este problema é frequentemente referido como a maldição da dimensionalidade.
- Felizmente, em problemas do mundo real, muitas vezes é possível reduzir consideravelmente o número de atributos, transformando um problema intratável em um tratável.

Redução de Dimensionalidade

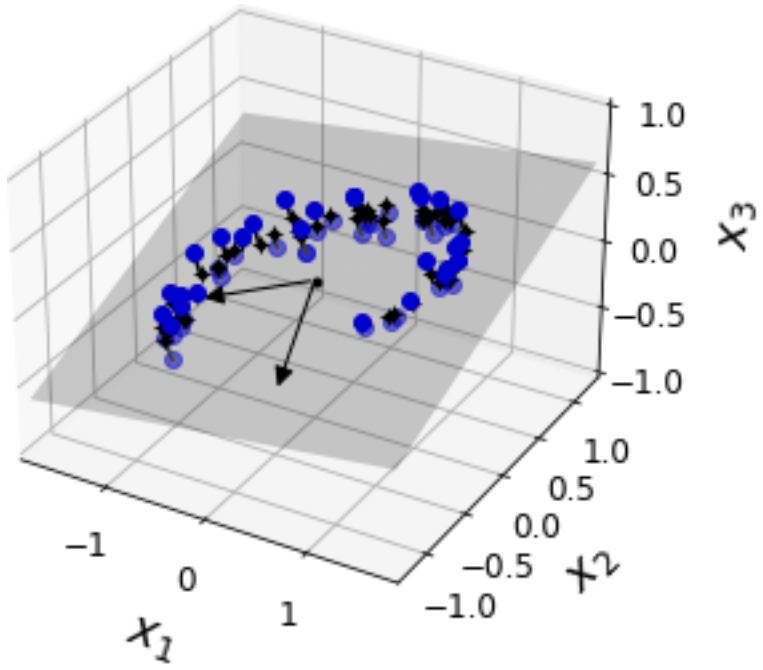
- A redução de dimensionalidade transforma dados de um espaço de alta dimensão em um espaço de baixa dimensão de forma que a representação de baixa dimensão retenha algumas propriedades significativas dos dados originais, idealmente próximas de sua dimensão original.
- As abordagens para redução de dimensionalidade podem ser divididas em ***seleção*** e ***extração de atributos***.
  - **Extração**: os dados originais sofrem algum tipo de transformação (e.g., projeção linear ou não-linear), dando origem a novos atributos em um espaço de dimensão possivelmente diferente.
  - **Seleção**: do conjunto de atributos originais do dado, somente um subconjunto deles é selecionado.

Redução de Dimensionalidade

- A redução de dimensionalidade pode ser usada para (i) redução de ruído, (ii) visualização de dados, análise de *clusters* ou (iii) como uma etapa intermediária para facilitar outras análises (e.g., reduzir o custo computacional envolvido no processamento dos dados através da redução da dimensionalidade.).
- A redução da dimensionalidade faz com que algumas informações sejam perdidas.
  - Exemplo: compactar uma imagem em JPEG pode degradar sua qualidade.
- Desta forma, embora acelere o treinamento, também pode fazer com que o desempenho do modelo seja um pouco pior.
- Entretanto, em alguns casos, a redução de dimensionalidade dos dados de treinamento pode filtrar ruído e detalhes desnecessários e, assim, resultar em melhor desempenho, mas em geral, ela apenas irá acelerar o treinamento.

Extração de Atributos

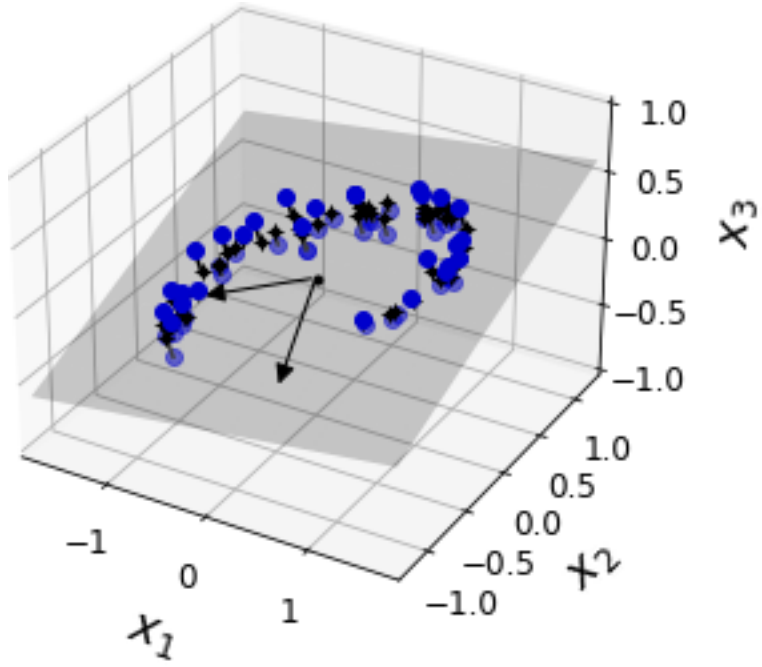
- Também conhecida como ***projeção de atributos***.
- Ela transforma os dados do espaço de alta dimensão em um espaço de menor dimensões.
- Na maioria dos problemas do mundo real, os exemplos de treinamento não são distribuídos uniformemente em todas as dimensões.
- Muitos atributos são quase constantes, enquanto outros são altamente correlacionados.
- Como resultado, todos os exemplos de treinamento na verdade estão dentro (ou perto de) um ***subespaço*** de dimensão muito menor do espaço de alta dimensão.

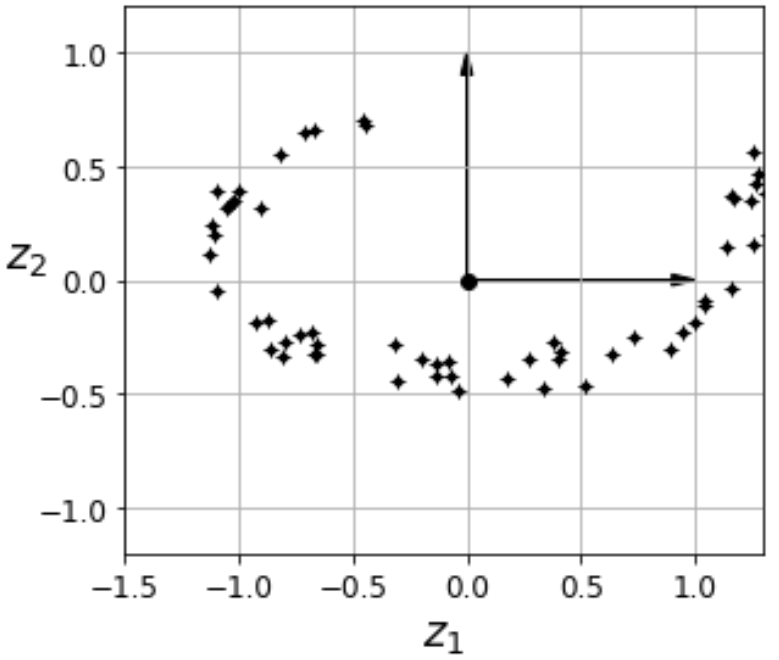


Conjunto de dados em 3D próximo a um subespaço de 2D.

Extração de Atributos

- Observe que todos os exemplos de treinamento ficam próximos a um plano:
  - Este é um subespaço de dimensão menor (2D) do espaço de dimensão alta (3D).
- Agora, se projetarmos cada exemplo de treinamento perpendicularmente neste subespaço (conforme representado pelas linhas curtas conectando os exemplos ao plano), obteremos o novo conjunto de dados 2D mostrado na figura ao lado.
- Observe que os eixos correspondem aos novos atributos z_1 e z_2 (as coordenadas das projeções no plano).



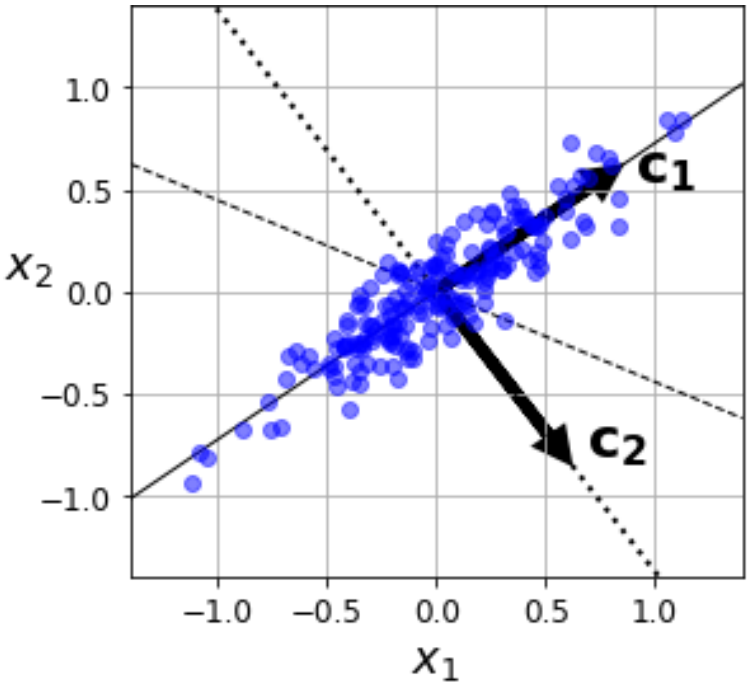


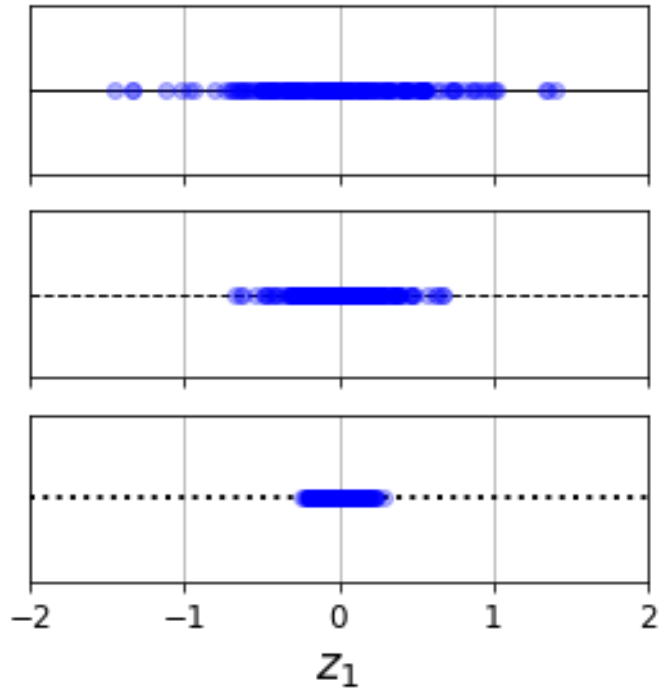
Análise de Componentes Principais (PCA)

- A Análise de Componentes Principais (em inglês, *Principal Component Analysis* - PCA) é o algoritmo de redução de dimensionalidade linear mais popular.
- Primeiro, ele identifica o ***hiperplano*** mais próximo dos dados e, em seguida, projeta os dados nele, como na figura anterior.
  - Ou seja, ele realiza um mapeamento linear dos dados para um espaço de dimensão menor de forma que a variância dos dados na representação de dimensão menor seja maximizada.
- Na prática, a matriz de covariância dos dados é construída e os autovetores e autovalores da matriz são calculados.
- Os autovetores que correspondem aos maiores autovalores (ou seja, os componentes principais) agora podem ser usados para reconstruir uma grande fração da variância dos dados originais.

Preservando a variância

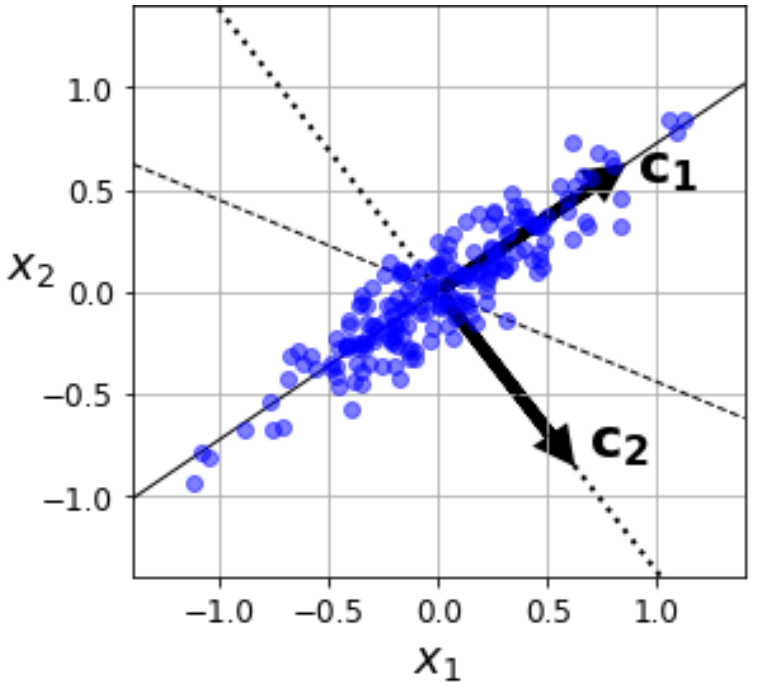
- Antes de projetarmos o conjunto de treinamento em um hiperplano de dimensão menor, primeiro precisamos escolher o hiperplano correto.
- Por exemplo, um conjunto de dados em 2D é representado na figura acima, junto com três eixos diferentes (ou seja, hiperplanos 1D).
- Na figura abaixo, está o resultado da projeção do conjunto de dados em cada um desses eixos.
  - Como podemos ver, a projeção na linha contínua preserva a variância máxima, enquanto a projeção na linha pontilhada preserva muito pouco da variância, e a projeção na linha tracejada preserva uma quantidade intermediária.

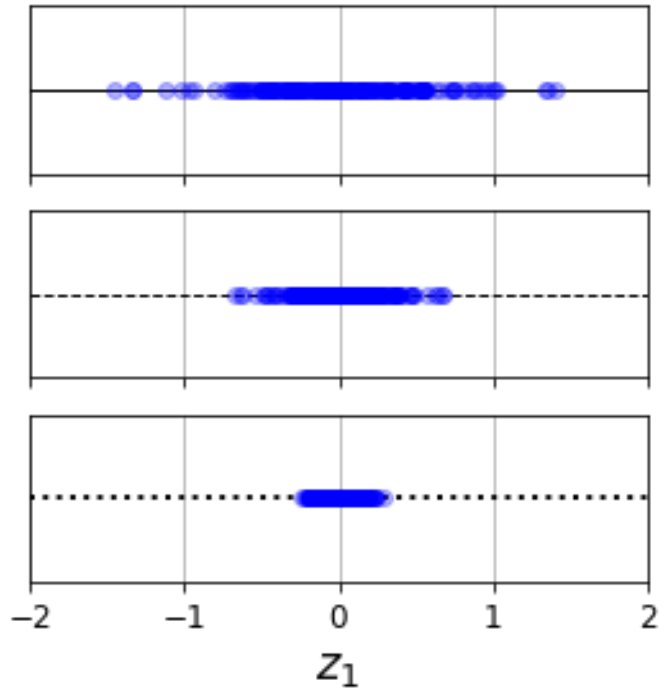




Preservando a variância

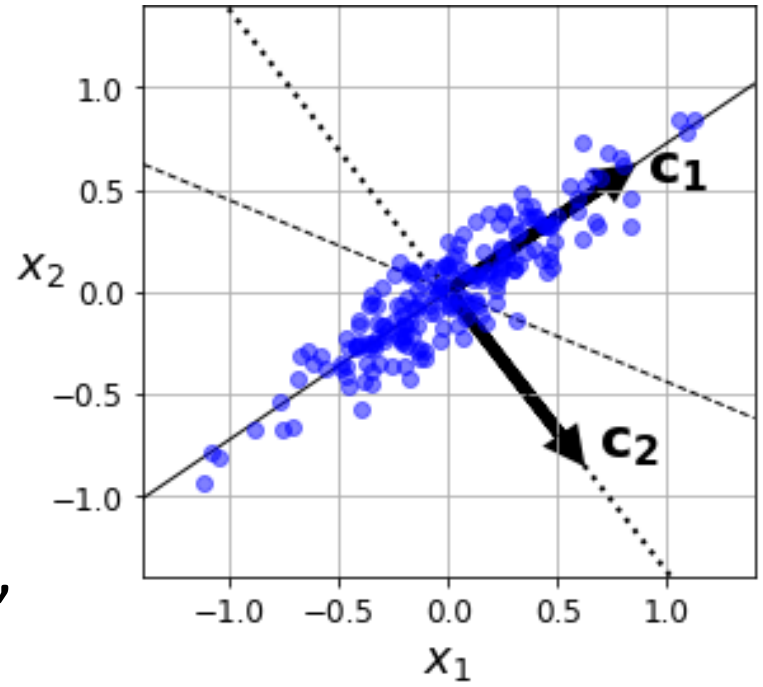
- Parece razoável selecionar o eixo (ou seja, hiperplano 1D) que preserva a quantidade máxima de variância, pois provavelmente será o eixo que perderá menos informações do que as outras projeções.
- Outra forma de justificar essa escolha é que este é o eixo que minimiza a distância quadrática média entre o conjunto de dados original e sua projeção nesse eixo.
- Esta é a ideia simples por trás do PCA: *"encontrar uma sequência de eixos que maximizam a variância e consigam explicar os dados"*.





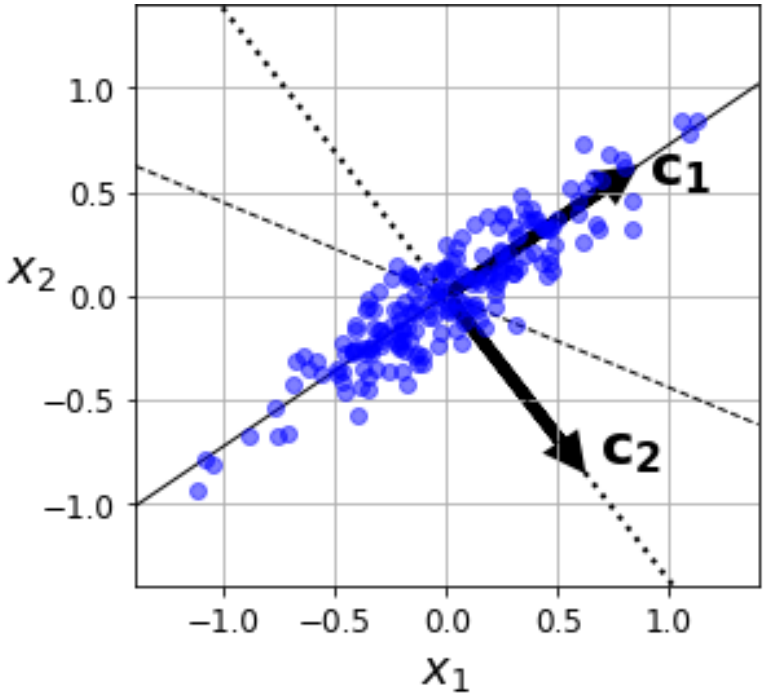
Componentes principais

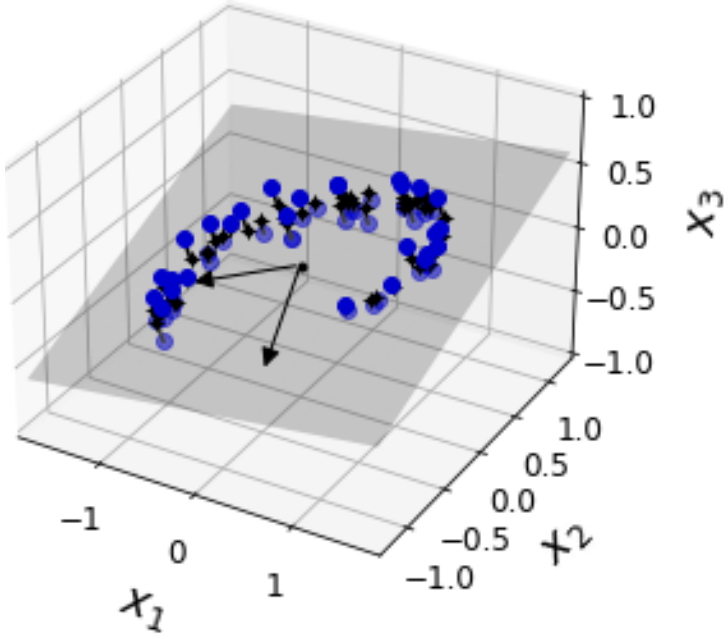
- PCA identifica o eixo responsável pela maior quantidade de variância no conjunto de treinamento. Na figura, é a linha sólida.
- Ele também encontra um segundo eixo, ***ortogonal*** ao primeiro, que responde pela maior quantidade de variância restante. Na figura, é a linha pontilhada.
- Se fosse um conjunto de dados de dimensão maior, o PCA encontraria um terceiro eixo, ***ortogonal*** a ambos os eixos anteriores, e um quarto, um quinto e assim por diante - tantos eixos quanto o número de dimensões do conjunto de dados.



Componentes principais

- O vetor unitário que define o *i*-ésimo eixo é chamado de *i*-ésimo ***componente principal*** (CP).
  - Na figura acima, o primeiro CP é C_1 e o segundo CP é C_2.
  - Na figura abaixo, os primeiros dois CPs são representados pelas setas ***ortogonais*** no plano, e o terceiro CP seria ***ortogonal*** ao plano (apontando para cima ou para baixo).
- Então, como podemos encontrar os CPs de um conjunto de treinamento?
  - Usamos a técnica de fatoração de matrizes chamada decomposição em valores singulares (em inglês, *Singular Value Decomposition* - SVD).





Componentes principais

- SDV decompõe a matriz de treinamento $\mathbf{X}$, com dimensões $n \times p$, a partir de sua matriz de covariância, $\mathbf{C_X}$, na multiplicação de três matrizes $\mathbf{U\Sigma V^T}$, onde
  - $\mathbf{U}$ é uma matriz com dimensões $n \times n$ cujas colunas são vetores unitários ortogonais de comprimento n chamados de vetores singulares à esquerda de $\mathbf{X}$,
  - $\mathbf{\Sigma}$ é uma matriz retangular diagonal com dimensões $n \times p$ com os ***valores singulares*** de $\mathbf{X}$ e
  - $\mathbf{V}$ é uma matriz com dimensões $p \times p$ cujas colunas são vetores unitários ortogonais de comprimento p chamados de vetores singulares à direita de $\mathbf{X}$.
- Como isso se relaciona com PCA?

Componentes principais

- Considere a matriz $\mathbf{X}$, com dimensões $n \times p$, para a qual nós temos a seguinte decomposição em valores singulares

$$\mathbf{X} = \mathbf{U\Sigma V^T}$$

- Existe um teorema em álgebra linear que diz que os valores singulares não-nulos de $\mathbf{X}$ são as raízes quadradas dos autovalores não-nulos de $\mathbf{XX^T}$ ou $\mathbf{X^TX}$.
- A afirmação anterior para o caso $\mathbf{X^TX}$ é comprovada da seguinte forma:

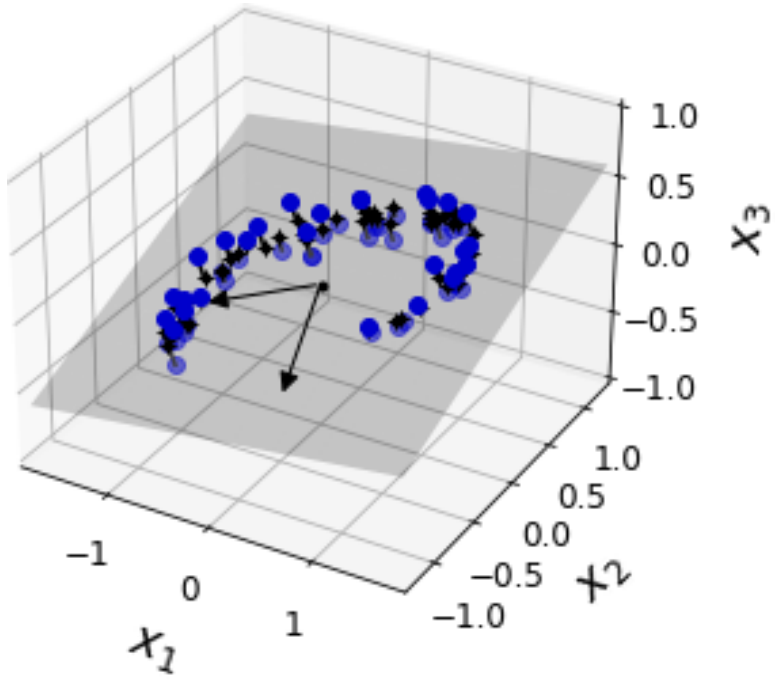
$$\begin{aligned} \mathbf{X^TX} &= \left(\mathbf{U\Sigma V^T}\right)^{\mathbf{T}}\left(\mathbf{U\Sigma V^T}\right) \\ &= \left(\mathbf{V\Sigma^T} \boldsymbol{U}^{\mathbf{T}}\right)\left(\mathbf{U\Sigma V^T}\right) \\ &= \mathbf{V}(\mathbf{\Sigma^T\Sigma})\mathbf{V^T} \end{aligned}$$

Componentes principais

- Observamos que $\mathbf{X^T X}$ é *semelhante* a $\mathbf{\Sigma^T \Sigma}$ e, portanto, possui os mesmos autovalores.
- Como $\mathbf{\Sigma^T \Sigma}$ é uma matriz diagonal quadrada $p \times p$, os autovalores são de fato os elementos diagonais, que são os quadrados dos valores singulares.
- Também deve ser notado que realizamos efetivamente uma decomposição de autovalores para a matriz, $\mathbf{X^T X}$.
- De fato, como a matriz $\mathbf{X^T X}$ é simétrica, esta é uma diagonalização ortogonal e, portanto, os autovetores de $\mathbf{X^T X}$ são as colunas de $\mathbf{V}$.
- Portanto, se realizarmos uma decomposição em valores singulares da matriz $\mathbf{X^T X}$, os componentes principais serão as colunas da matriz ortogonal, $\mathbf{V}$.

Projetando em d dimensões

- Depois de identificar todos os componentes principais, podemos reduzir a dimensionalidade do conjunto de dados para d dimensões projetando-o no hiperplano definido pelos primeiros d componentes principais.
- Selecionar este hiperplano garante que a projeção preservará a maior variância possível, ou seja, a maior quantidade de informação possível.
- Por exemplo, na figura ao lado o conjunto de dados 3D é projetado no plano 2D definido pelos dois primeiros componentes principais, preservando uma grande parte da variância do conjunto de dados.
- Como resultado, a projeção 2D se parece muito com o conjunto de dados 3D original.



Projetando em d dimensões

- Para projetar o conjunto de treinamento no hiperplano de menor dimensão, podemos simplesmente calcular a multiplicação da matriz do conjunto de treinamento $\mathbf{X}$ pela matriz $\mathbf{W}_d$.
- A matriz $\mathbf{W}_d$ é definida como a matriz contendo os primeiros d componentes principais (ou seja, a matriz composta pelas primeiras d colunas de $\mathbf{V}$), conforme mostrado na equação abaixo.

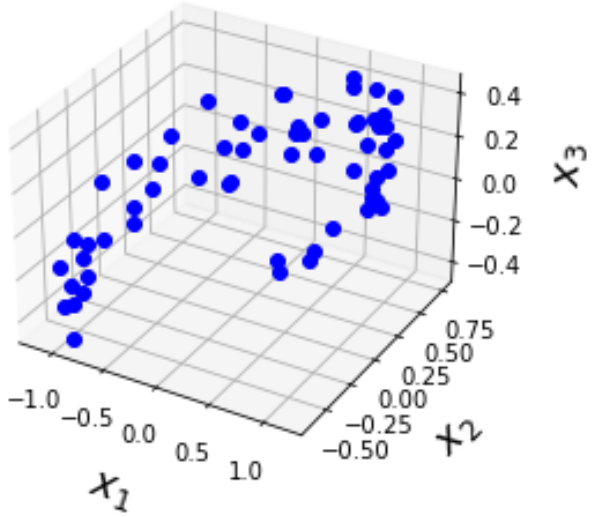
$$\mathbf{X}_{d-\text{proj}} = \mathbf{X}\mathbf{W}_d.$$

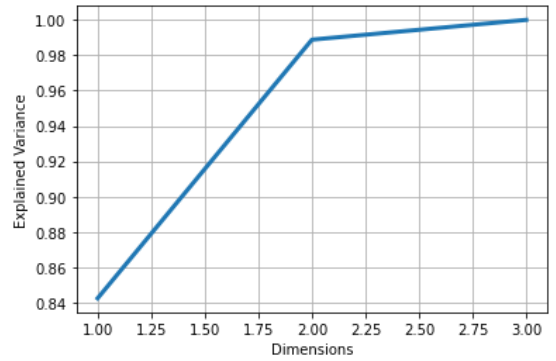
Exemplo: aplicando_pca_em_conjunto_3D.ipynb

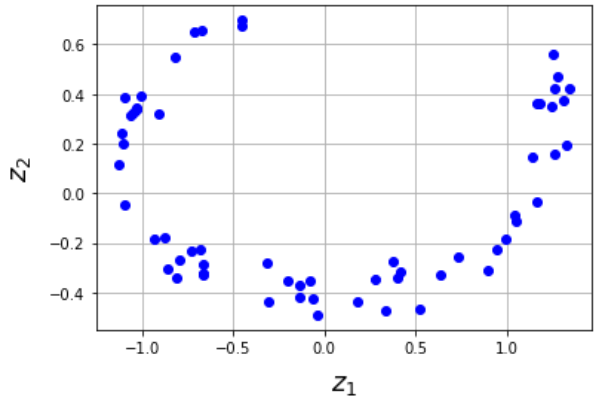
PCA com a biblioteca Scikit-Learn

```
import sklearn
import numpy as np

m = 60
w1, w2 = 0.1, 0.3
noise = 0.1

# Creating a 3D dataset.
angles = np.random.rand(m) * 3 * np.pi / 2 - 0.5
X = np.empty((m, 3))
X[:, 0] = np.cos(angles) + np.sin(angles)/2 + noise *
np.random.randn(m) / 2
X[:, 1] = np.sin(angles) * 0.7 + noise *
np.random.randn(m) / 2
X[:, 2] = X[:, 0] * w1 + X[:, 1] * w2 + noise *
np.random.randn(m)

# Applying PCA to reduce the dataset.
pca = PCA(n_components=0.95)
X_reduced = pca.fit_transform(X)
```

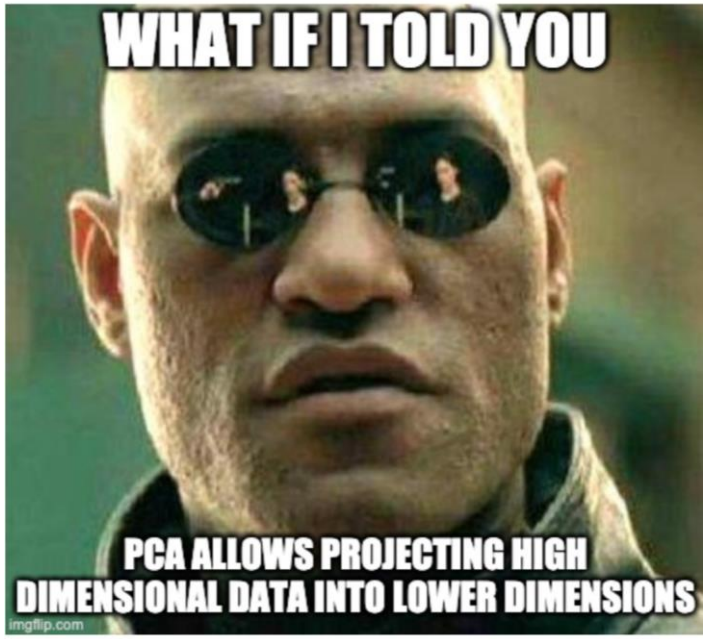
Escolhe número de componentes que explicam no mínimo 95% da variância do conjunto.

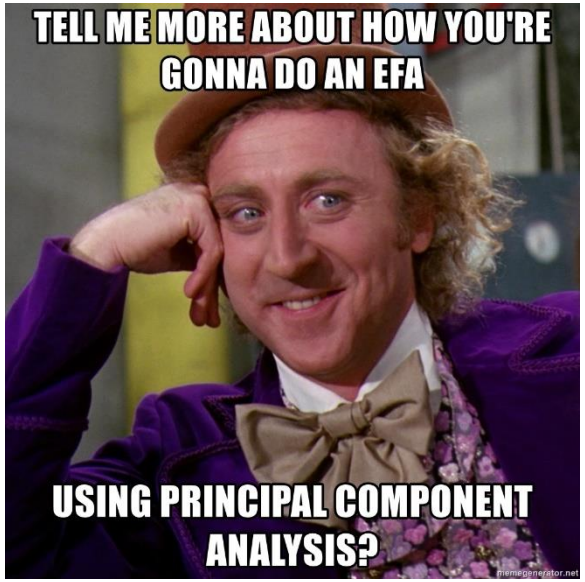


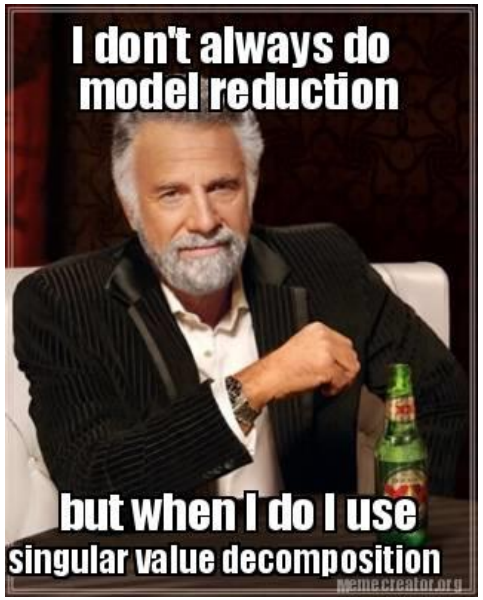


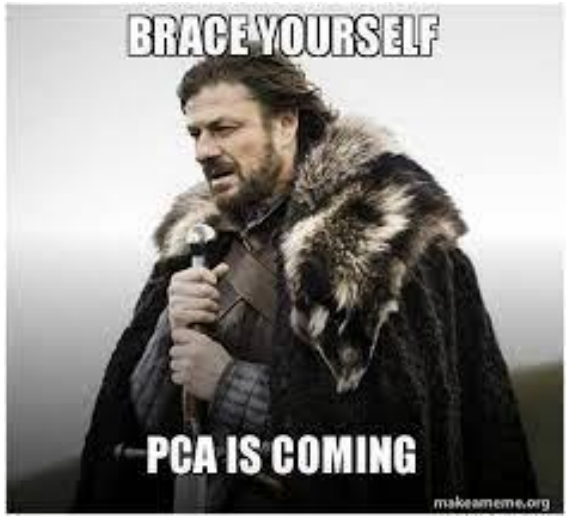


Obrigado!

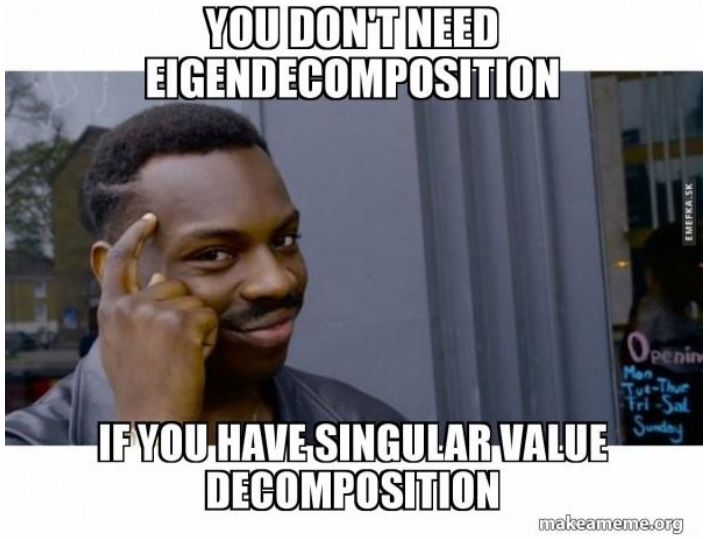








What's your favourite dimensionality reduction algorithm?



Figuras