Mineração de Dados

Aula 11: Aprendizado Não Supervisionado: Redução de Dimensionalidade

Rafael Izbicki

Aprendizado não supervisionado

Aula de Hoje: Redução de Dimensionalidade

"Encontrar transformações dos dados que capturem boa parte das informações presentes neles, de modo a tirar redundâncias nas covariáveis"

Porquê? Para visualização, achar outliers, como primeiro passo para clustering, para classificação, regressão etc

Aula de Hoje: Redução de Dimensionalidade

"Encontrar transformações dos dados que capturem boa parte das informações presentes neles, de modo a tirar redundâncias nas covariáveis"

Porquê? Para visualização, achar outliers, como primeiro passo para clustering, para classificação, regressão etc

Aula de Hoje: Redução de Dimensionalidade

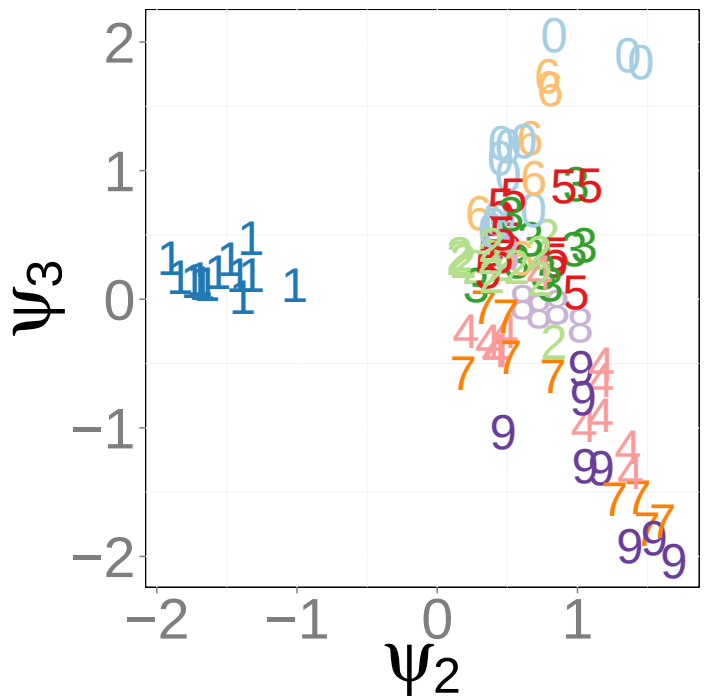
"Encontrar transformações dos dados que capturem boa parte das informações presentes neles, de modo a tirar redundâncias nas covariáveis"

Porquê? Para visualização, achar outliers, como primeiro passo para clustering, para classificação, regressão etc

Aula de Hoje: Redução de Dimensionalidade

"Encontrar transformações dos dados que capturem boa parte das informações presentes neles, de modo a tirar redundâncias nas covariáveis"

Porquê? Para visualização, achar outliers, como primeiro passo para clustering, para classificação, regressão etc



Componentes Principais (PCA)

"Encontrar transformações dos dados que capturem boa parte das informações presentes neles, de modo a tirar redundâncias nas covariáveis"

Queremos criar um número "pequeno"de variáveis $Z_1, Z_2, \ldots$ a partir de $X_1, X_2, \ldots$ que resumam bem a informação presente nelas.

Informação de Z_i é medida através de suas variabilidade

Redundância entre Z_1 e Z_2 é medida através de sua correlação

Nos restringimos combinações lineares das covariáveis originais

Componentes Principais (PCA)

"Encontrar transformações dos dados que capturem boa parte das informações presentes neles, de modo a tirar redundâncias nas covariáveis"

Queremos criar um número "pequeno"de variáveis $Z_1, Z_2, \ldots$ a partir de $X_1, X_2, \ldots$ que resumam bem a informação presente nelas.

Informação de Z_i é medida através de suas variabilidade

Redundância entre Z_1 e Z_2 é medida através de sua correlação

Nos restringimos combinações lineares das covariáveis originais

Componentes Principais (PCA)

"Encontrar transformações dos dados que capturem boa parte das informações presentes neles, de modo a tirar redundâncias nas covariáveis"

Queremos criar um número "pequeno"de variáveis $Z_1, Z_2, \ldots$ a partir de $X_1, X_2, \ldots$ que resumam bem a informação presente nelas.

Informação de Z_i é medida através de suas variabilidade

Redundância entre Z_1 e Z_2 é medida através de sua correlação

Nos restringimos combinações lineares das covariáveis originais

Componentes Principais (PCA)

"Encontrar transformações dos dados que capturem boa parte das informações presentes neles, de modo a tirar redundâncias nas covariáveis"

Queremos criar um número "pequeno"de variáveis $Z_1, Z_2, \ldots$ a partir de $X_1, X_2, \ldots$ que resumam bem a informação presente nelas.

Informação de Z_i é medida através de suas variabilidade

Redundância entre Z_1 e Z_2 é medida através de sua correlação

Nos restringimos combinações lineares das covariáveis originais

Componentes Principais (PCA)

"Encontrar transformações dos dados que capturem boa parte das informações presentes neles, de modo a tirar redundâncias nas covariáveis"

Queremos criar um número "pequeno"de variáveis $Z_1, Z_2, \ldots$ a partir de $X_1, X_2, \ldots$ que resumam bem a informação presente nelas.

Informação de Z_i é medida através de suas variabilidade

Redundância entre Z_1 e Z_2 é medida através de sua correlação

Nos restringimos combinações lineares das covariáveis originais

Formalmente, o primeiro componente principal de $\mathbf{X}$ é a variável Z_1 que

- (i) é combinação linear das variáveis $\mathbf{X}$:

$$Z_1 = \phi_{11} X_1 + \ldots + \phi_{d1} X_d$$

- (ii) tem maior variância.

O segundo componente principal de $\mathbf{X}$ é a variável Z_2 que

- (i) é combinação linear das variáveis $\mathbf{X}$:

$$Z_2 = \phi_{12} X_1 + \ldots + \phi_{d2} X_d$$

- (ii) tem maior variância.
- (iii) tem correlação 0 com Z_1

Formalmente, o primeiro componente principal de $\mathbf{X}$ é a variável Z_1 que

- (i) é combinação linear das variáveis $\mathbf{X}$:

$$Z_1 = \phi_{11} X_1 + \ldots + \phi_{d1} X_d$$

- (ii) tem maior variância.

O segundo componente principal de $\mathbf{X}$ é a variável Z_2 que

- (i) é combinação linear das variáveis $\mathbf{X}$:

$$Z_2 = \phi_{12} X_1 + \ldots + \phi_{d2} X_d$$

- (ii) tem maior variância.
- (iii) tem correlação 0 com Z_1

Solução: encontrar autovetores de $C = X^t X$ (matriz de covariâncias; estamos assumindo que os dados estão normalizados para ter média 0)

Seja U a matrix $d \times d$ em que a i-ésima coluna contém o i-ésimo autovetor de C.

U é justamente a matriz de *cargas*, i.e., o seu elemento i, j é dado pelo coeficiente ótimo ϕ_{ij}

Assim, $Z = U^t X^t$ é a matriz $d \times n$ com as novas d novas covariáveis para cada uma das n observações.

Solução: encontrar autovetores de $C = X^t X$ (matriz de covariâncias; estamos assumindo que os dados estão normalizados para ter média 0)

Seja U a matrix $d \times d$ em que a i-ésima coluna contém o i-ésimo autovetor de C.

U é justamente a matriz de *cargas*, i.e., o seu elemento i, j é dado pelo coeficiente ótimo ϕ_{ij}

Assim, $Z = U^t X^t$ é a matriz $d \times n$ com as novas d novas covariáveis para cada uma das n observações.

Solução: encontrar autovetores de $C = X^t X$ (matriz de covariâncias; estamos assumindo que os dados estão normalizados para ter média 0)

Seja U a matrix $d \times d$ em que a i-ésima coluna contém o i-ésimo autovetor de C.

U é justamente a matriz de *cargas*, i.e., o seu elemento i, j é dado pelo coeficiente ótimo ϕ_{ij}

Assim, $Z = U^t X^t$ é a matriz $d \times n$ com as novas d novas covariáveis para cada uma das n observações.

Solução: encontrar autovetores de $C = X^t X$ (matriz de covariâncias; estamos assumindo que os dados estão normalizados para ter média 0)

Seja U a matrix $d \times d$ em que a i-ésima coluna contém o i-ésimo autovetor de C.

U é justamente a matriz de *cargas*, i.e., o seu elemento i, j é dado pelo coeficiente ótimo ϕ_{ij}

Assim, $Z = U^t X^t$ é a matriz $d \times n$ com as novas d novas covariáveis para cada uma das n observações.

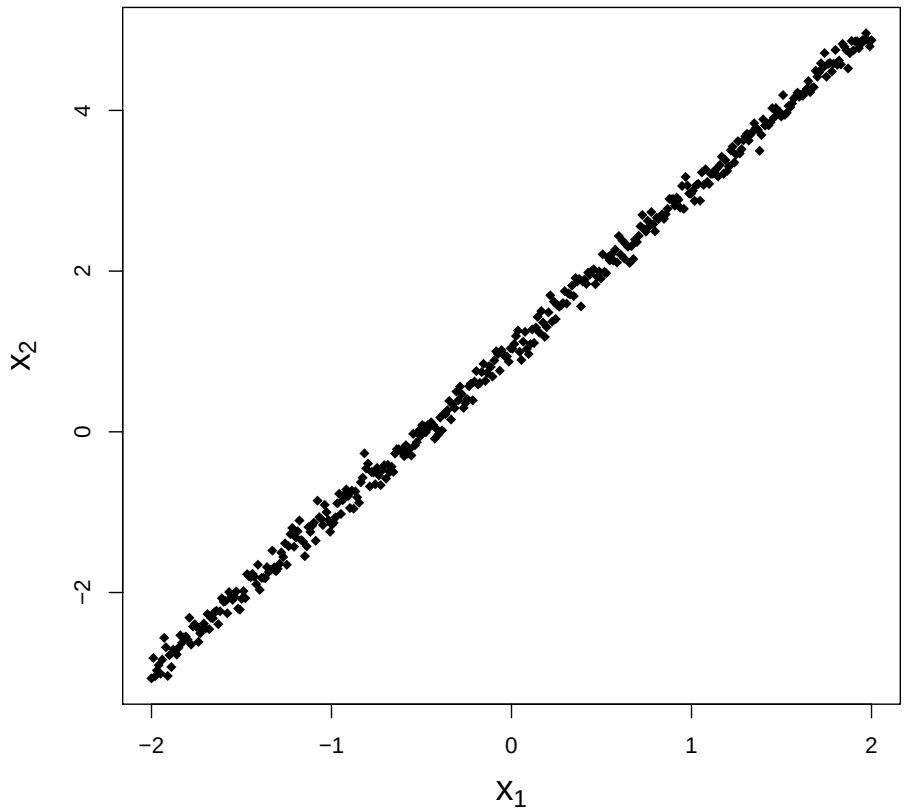
Solução: encontrar autovetores de $C = X^t X$ (matriz de covariâncias; estamos assumindo que os dados estão normalizados para ter média 0)

Seja U a matrix $d \times d$ em que a i-ésima coluna contém o i-ésimo autovetor de C.

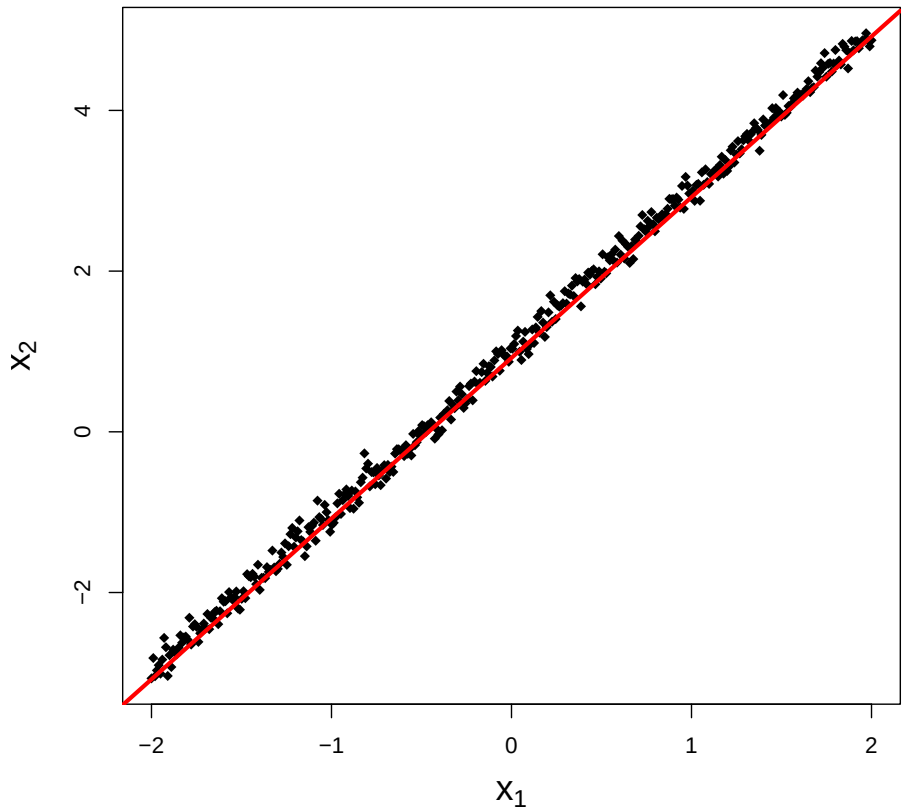
U é justamente a matriz de *cargas*, i.e., o seu elemento i, j é dado pelo coeficiente ótimo ϕ_{ij}

Assim, $Z = U^t X^t$ é a matriz $d \times n$ com as novas d novas covariáveis para cada uma das n observações.

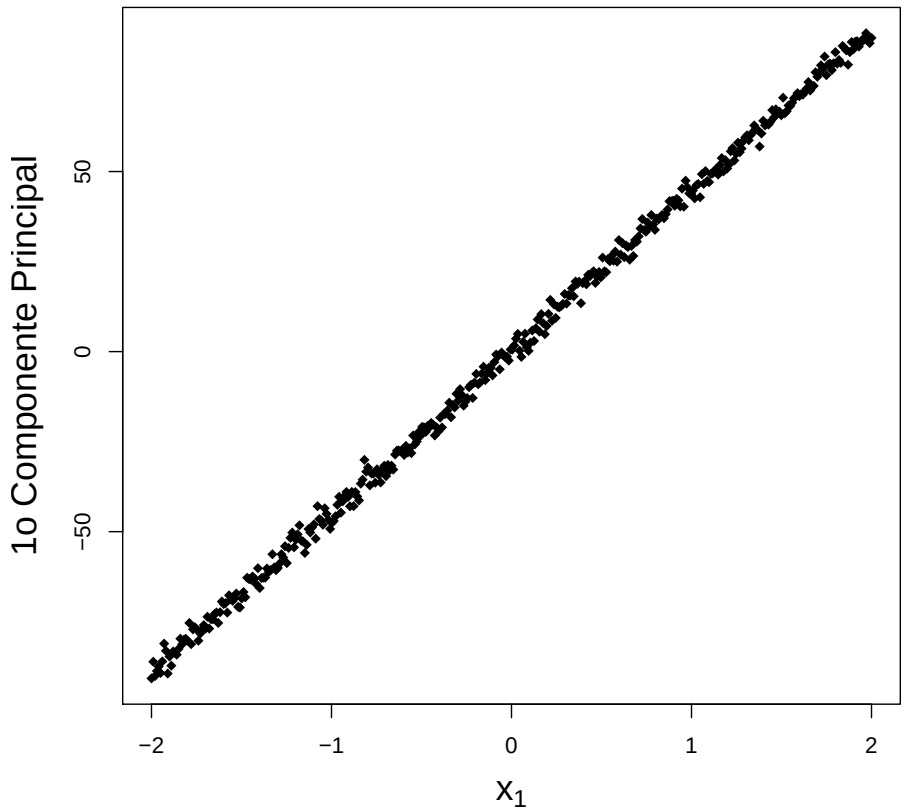
O que PCA está fazendo?



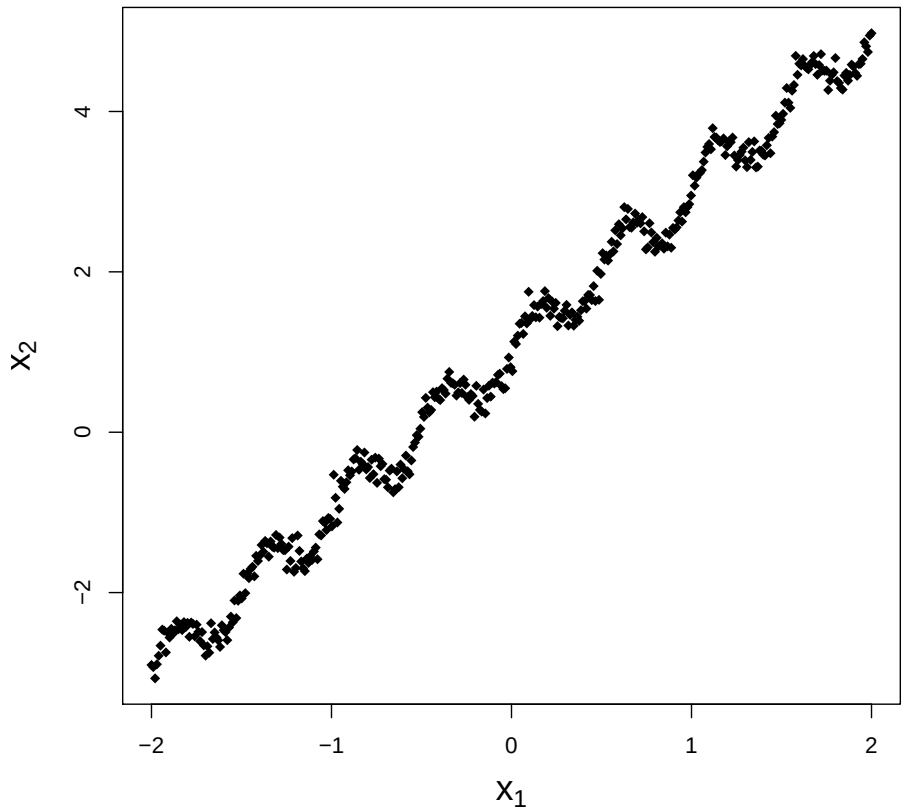
O que PCA está fazendo?



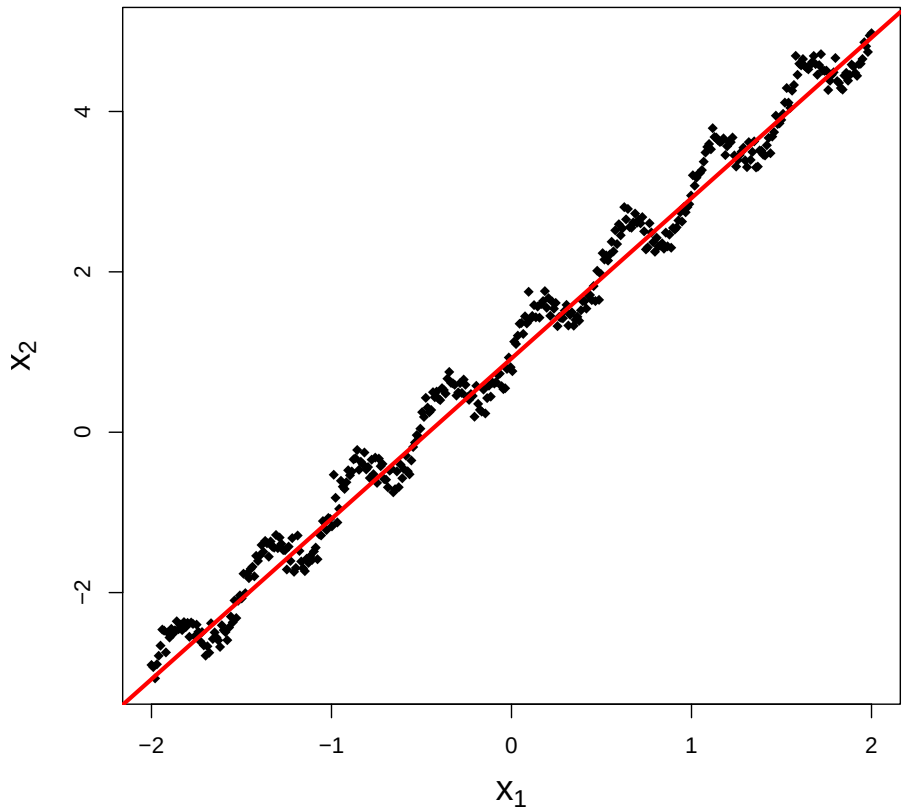
O que PCA está fazendo?



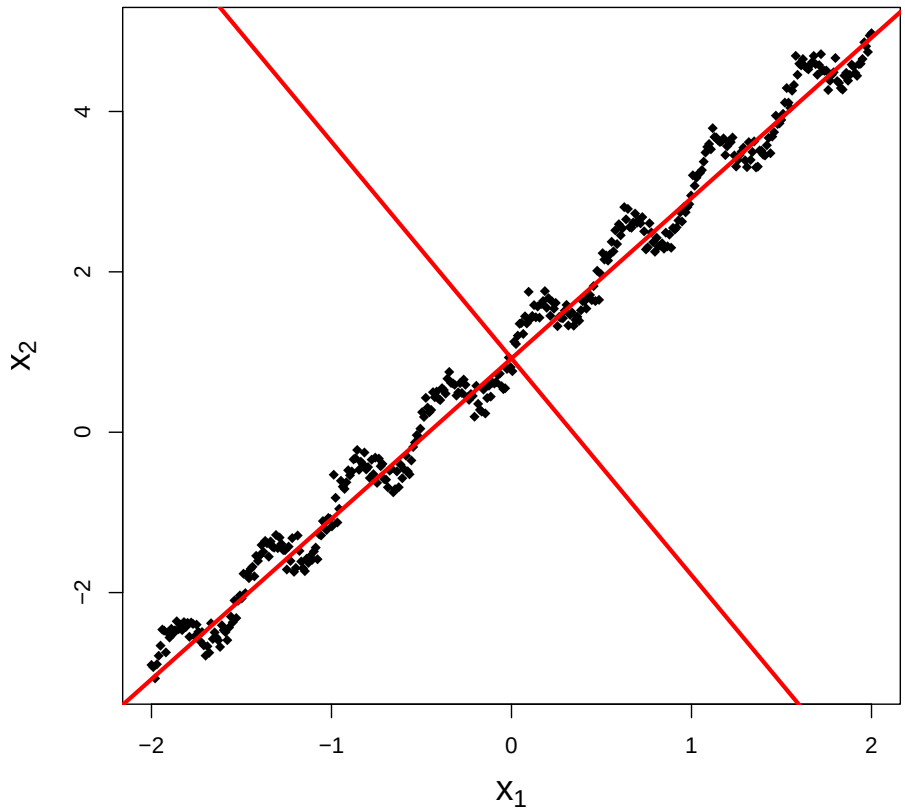
O que PCA está fazendo?



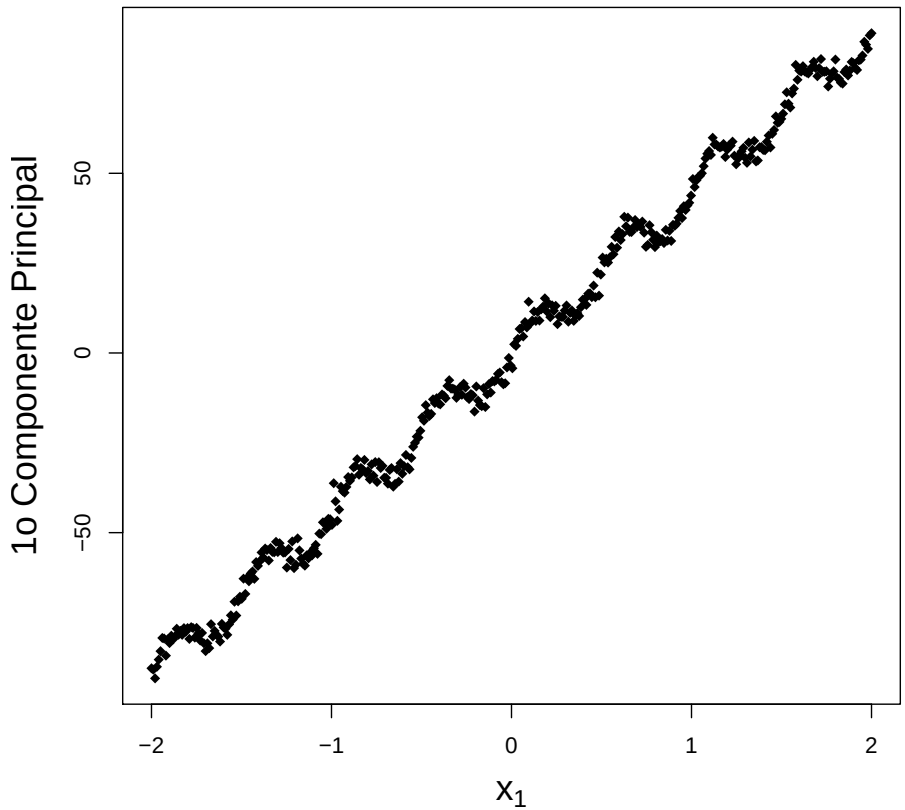
O que PCA está fazendo?



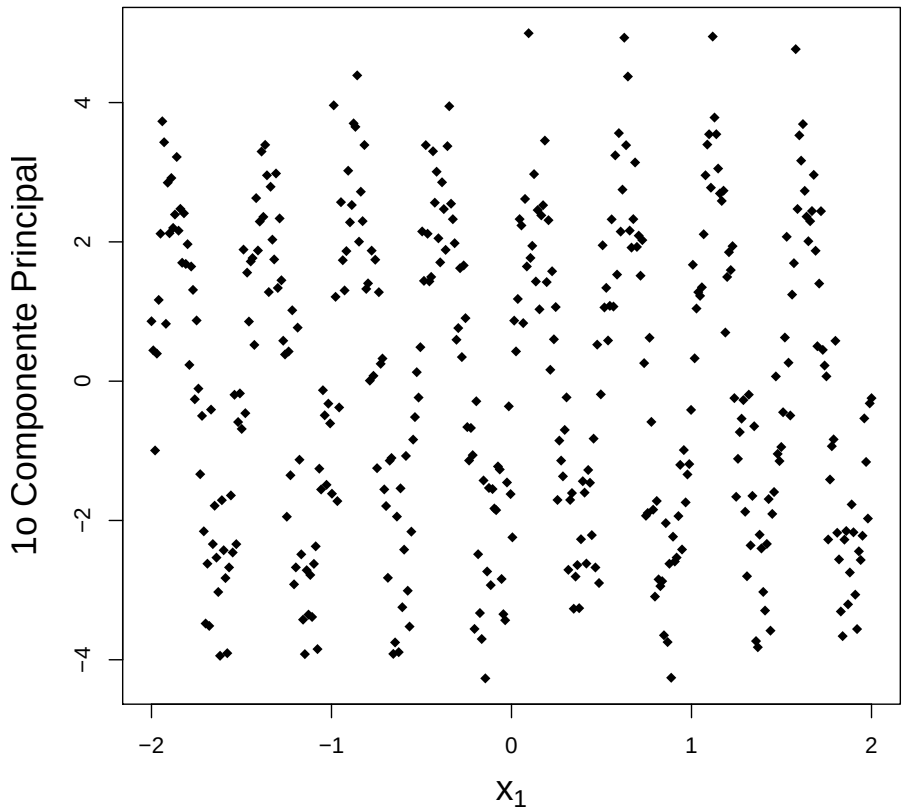
O que PCA está fazendo?



O que PCA está fazendo?



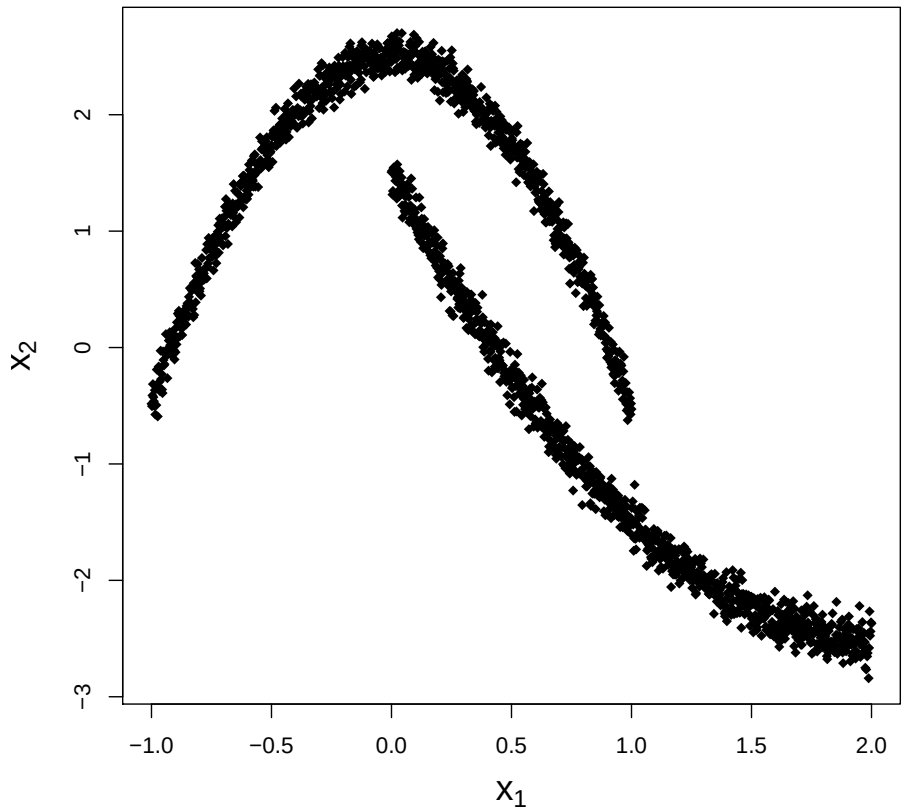
O que PCA está fazendo?



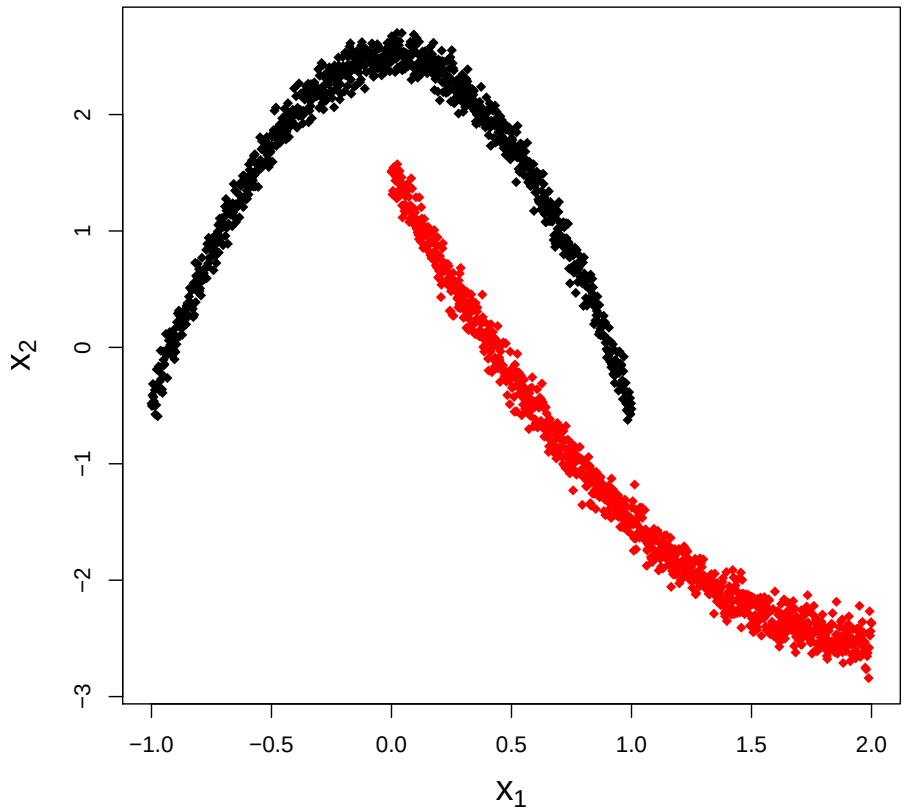
Interpretação alternativa: escalonamento multidimensional.

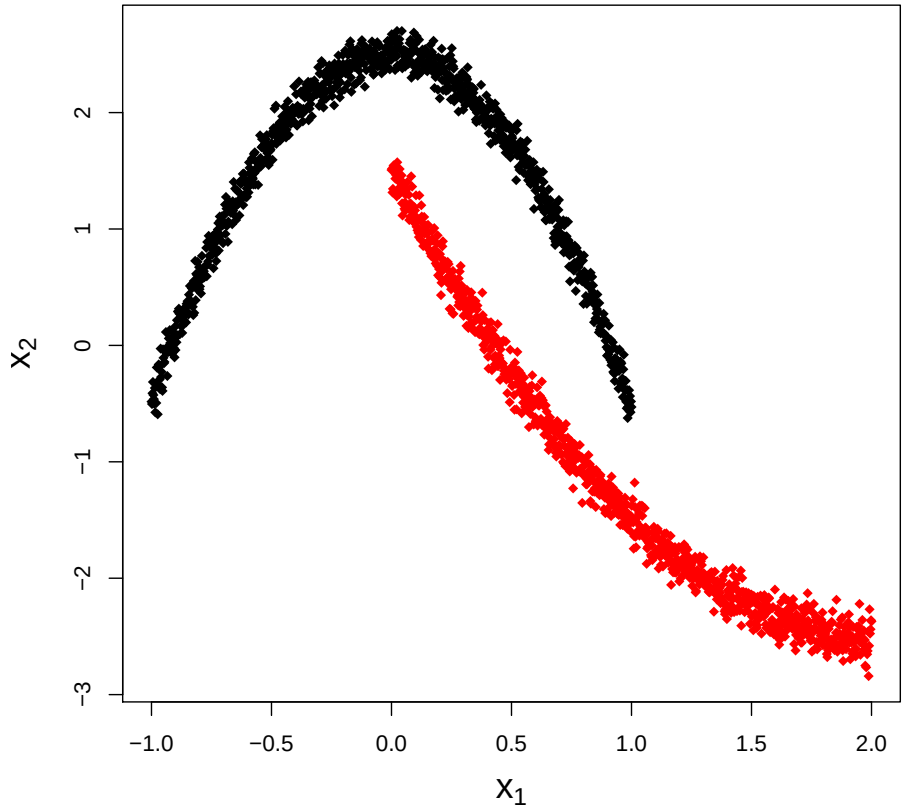
$$\sum_{i,j}(||\mathbf{x}_i - \mathbf{x}_j||^2 - ||\mathbf{z}_i - \mathbf{z}_j||^2)$$

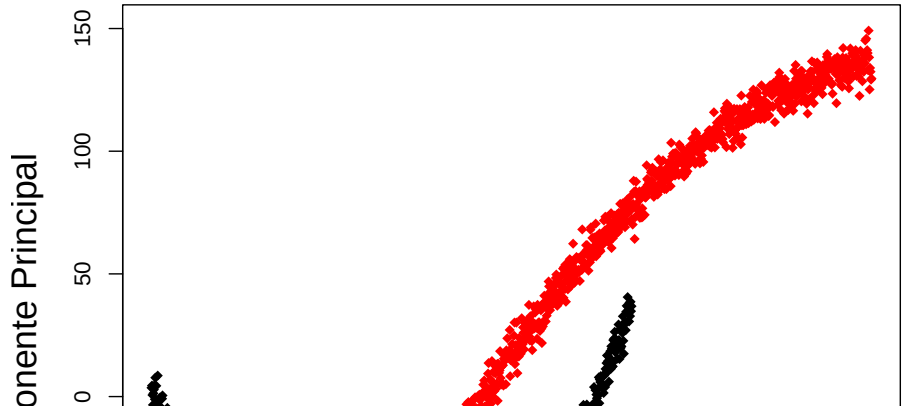
E se...

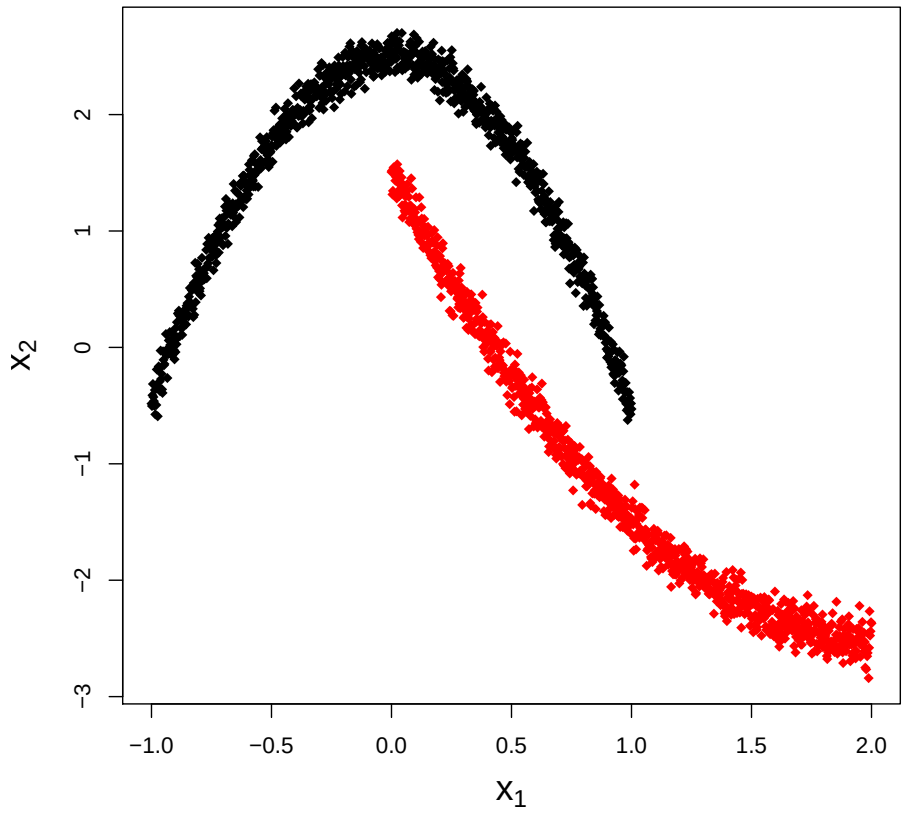


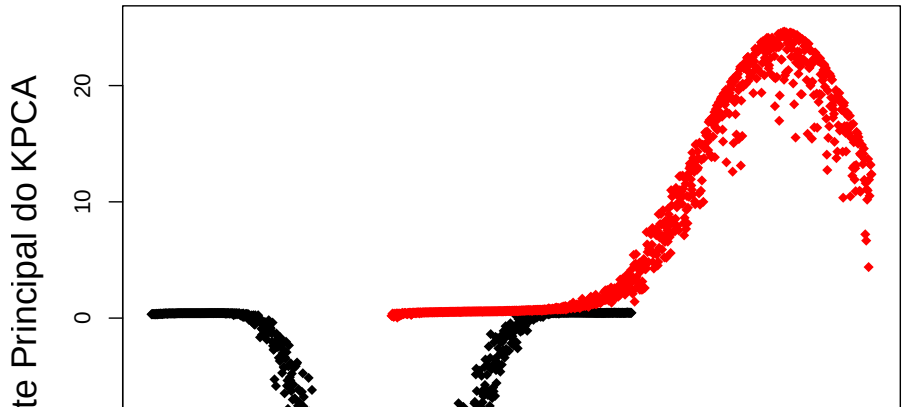
?


X2
2
1
0
−1
−2
−3
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0
X1


x2
2
1
0
−1
−2
−3
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0
x1


onente Principal
150
100
50
0


x2
2
1
0
−1
−2
−3
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0
x1


te Principal do KPCA
20
10
0

Solução do PCA: encontrar autovetores de $C = X^t X$ (matriz de covariâncias)

O fato fundamental para usar o truque do kernel é que Z também pode ser calculada via o cálculo dos autovetores de $K = XX^t$, a matriz com os produtos internos entre cada par de observações.

Mais especificamente: $Z = U_2^t$, em que U_2 é a matriz de autovetores de K.

Solução do PCA: encontrar autovetores de $C = X^t X$ (matriz de covariâncias)

O fato fundamental para usar o truque do kernel é que Z também pode ser calculada via o cálculo dos autovetores de $K = XX^t$, a matriz com os produtos internos entre cada par de observações.

Mais especificamente: $Z = U_2^t$, em que U_2 é a matriz de autovetores de K.

Solução do PCA: encontrar autovetores de $C = X^t X$ (matriz de covariâncias)

O fato fundamental para usar o truque do kernel é que Z também pode ser calculada via o cálculo dos autovetores de $K = XX^t$, a matriz com os produtos internos entre cada par de observações.

Mais especificamente: $Z = U_2^t$, em que U_2 é a matriz de autovetores de K.

Logo, para calcular as componentes principais, basta saber os produtos internos entre as observações.

PCA: extrair covariáveis que são transformações lineares dos dados

KPCA (kernel PCA): extrair covariáveis que são transformações não lineares dos dados

Truque do kernel em PCA: ao invés de calcular a autodecomposição de $K = XX^t$, usamos outros kernels

$$K = \begin{bmatrix} K(\mathbf{x}_1, \mathbf{x}_1) & K(\mathbf{x}_1, \mathbf{x}_2) & \cdots & K(\mathbf{x}_1, \mathbf{x}_n) \\ K(\mathbf{x}_2, \mathbf{x}_1) & K(\mathbf{x}_2, \mathbf{x}_2) & \cdots & K(\mathbf{x}_2, \mathbf{x}_n) \\ \vdots & \vdots & \ddots & \vdots \\ K(\mathbf{x}_n, \mathbf{x}_1) & K(\mathbf{x}_n, \mathbf{x}_2) & \cdots & K(\mathbf{x}_n, \mathbf{x}_n) \end{bmatrix}$$

$$K(\mathbf{x}_i, \mathbf{x}_l) = (1 + \langle \mathbf{x}_i, \mathbf{x}_l \rangle)^d : \text{ kernel polinomial}$$

$$K(\mathbf{x}_i, \mathbf{x}_l) = e^{-\frac{d^2(\mathbf{x}_i, \mathbf{x}_l)}{2h^2}} : \text{ kernel gaussiano}$$

Logo, para calcular as componentes principais, basta saber os produtos internos entre as observações.

PCA: extrair covariáveis que são transformações lineares dos dados

KPCA (kernel PCA): extrair covariáveis que são transformações não lineares dos dados

Truque do kernel em PCA: ao invés de calcular a autodecomposição de $K = XX^t$, usamos outros kernels

$$K = \begin{bmatrix} K(\mathbf{x}_1, \mathbf{x}_1) & K(\mathbf{x}_1, \mathbf{x}_2) & \cdots & K(\mathbf{x}_1, \mathbf{x}_n) \\ K(\mathbf{x}_2, \mathbf{x}_1) & K(\mathbf{x}_2, \mathbf{x}_2) & \cdots & K(\mathbf{x}_2, \mathbf{x}_n) \\ \vdots & \vdots & \ddots & \vdots \\ K(\mathbf{x}_n, \mathbf{x}_1) & K(\mathbf{x}_n, \mathbf{x}_2) & \cdots & K(\mathbf{x}_n, \mathbf{x}_n) \end{bmatrix}$$

$$K(\mathbf{x}_i, \mathbf{x}_l) = (1 + \langle \mathbf{x}_i, \mathbf{x}_l \rangle)^d : \text{ kernel polinomial}$$

$$K(\mathbf{x}_i, \mathbf{x}_l) = e^{-\frac{d^2(\mathbf{x}_i, \mathbf{x}_l)}{2h^2}} : \text{ kernel gaussiano}$$

Logo, para calcular as componentes principais, basta saber os produtos internos entre as observações.

PCA: extrair covariáveis que são transformações lineares dos dados

KPCA (kernel PCA): extrair covariáveis que são transformações não lineares dos dados

Truque do kernel em PCA: ao invés de calcular a autodecomposição de $K = XX^t$, usamos outros kernels

$$K = \begin{bmatrix} K(\mathbf{x}_1, \mathbf{x}_1) & K(\mathbf{x}_1, \mathbf{x}_2) & \cdots & K(\mathbf{x}_1, \mathbf{x}_n) \\ K(\mathbf{x}_2, \mathbf{x}_1) & K(\mathbf{x}_2, \mathbf{x}_2) & \cdots & K(\mathbf{x}_2, \mathbf{x}_n) \\ \vdots & \vdots & \ddots & \vdots \\ K(\mathbf{x}_n, \mathbf{x}_1) & K(\mathbf{x}_n, \mathbf{x}_2) & \cdots & K(\mathbf{x}_n, \mathbf{x}_n) \end{bmatrix}$$

$$K(\mathbf{x}_i, \mathbf{x}_l) = (1 + \langle \mathbf{x}_i, \mathbf{x}_l \rangle)^d : \text{ kernel polinomial}$$

$$K(\mathbf{x}_i, \mathbf{x}_l) = e^{-\frac{d^2(\mathbf{x}_i, \mathbf{x}_l)}{2h^2}} : \text{ kernel gaussiano}$$

Logo, para calcular as componentes principais, basta saber os produtos internos entre as observações.

PCA: extrair covariáveis que são transformações lineares dos dados

KPCA (kernel PCA): extrair covariáveis que são transformações não lineares dos dados

Truque do kernel em PCA: ao invés de calcular a autodecomposição de $K = XX^t$, usamos outros kernels

$$K = \begin{bmatrix} K(\mathbf{x}_1, \mathbf{x}_1) & K(\mathbf{x}_1, \mathbf{x}_2) & \cdots & K(\mathbf{x}_1, \mathbf{x}_n) \\ K(\mathbf{x}_2, \mathbf{x}_1) & K(\mathbf{x}_2, \mathbf{x}_2) & \cdots & K(\mathbf{x}_2, \mathbf{x}_n) \\ \vdots & \vdots & \ddots & \vdots \\ K(\mathbf{x}_n, \mathbf{x}_1) & K(\mathbf{x}_n, \mathbf{x}_2) & \cdots & K(\mathbf{x}_n, \mathbf{x}_n) \end{bmatrix}$$

$$K(\mathbf{x}_i, \mathbf{x}_l) = (1 + \langle \mathbf{x}_i, \mathbf{x}_l \rangle)^d : \text{ kernel polinomial}$$

$$K(\mathbf{x}_i, \mathbf{x}_l) = e^{-\frac{d^2(\mathbf{x}_i, \mathbf{x}_l)}{2h^2}} : \text{ kernel gaussiano}$$

Logo, para calcular as componentes principais, basta saber os produtos internos entre as observações.

PCA: extrair covariáveis que são transformações lineares dos dados

KPCA (kernel PCA): extrair covariáveis que são transformações não lineares dos dados

Truque do kernel em PCA: ao invés de calcular a autodecomposição de $K = XX^t$, usamos outros kernels

$$K = \begin{bmatrix} K(\mathbf{x}_1, \mathbf{x}_1) & K(\mathbf{x}_1, \mathbf{x}_2) & \cdots & K(\mathbf{x}_1, \mathbf{x}_n) \\ K(\mathbf{x}_2, \mathbf{x}_1) & K(\mathbf{x}_2, \mathbf{x}_2) & \cdots & K(\mathbf{x}_2, \mathbf{x}_n) \\ \vdots & \vdots & \ddots & \vdots \\ K(\mathbf{x}_n, \mathbf{x}_1) & K(\mathbf{x}_n, \mathbf{x}_2) & \cdots & K(\mathbf{x}_n, \mathbf{x}_n) \end{bmatrix}$$

$$K(\mathbf{x}_i, \mathbf{x}_l) = (1 + \langle \mathbf{x}_i, \mathbf{x}_l \rangle)^d : \text{ kernel polinomial}$$

$$K(\mathbf{x}_i, \mathbf{x}_l) = e^{-\frac{d^2(\mathbf{x}_i, \mathbf{x}_l)}{2h^2}} : \text{ kernel gaussiano}$$

No R:

```
 library(kernlab)
 # ou kernel polinomial

kpc=kpca(dados,kernel="polydot",
  kpar=list(degree=2),features=2)
  # ou kernel gaussiano: (sigma \'e um sobre a variancia)
kpc=kpca(dados,kernel="rbfdot",
  kpar=list(sigma=1),features=2)

variaveisEmNovosDados <- predict(kpc,novosDados)
```

Como escolher os "tuning parameters"?

Como escolher qual kernel usar?

Depende da aplicação!

Ex: se estamos usando esses componentes como primeiro passo para classificação, podemos escolher por validação cruzada

Se queremos visualizar os dados, podemos testar várias configurações. Elas podem trazer insights diferentes para os dados.

Uma heurística quando usamos o kernel Gaussiano:

$$h \approx \text{mediana}_{i,j} d(\mathbf{x}_i, \mathbf{x}_j)$$

Como escolher os "tuning parameters"?

Como escolher qual kernel usar?

Depende da aplicação!

Ex: se estamos usando esses componentes como primeiro passo para classificação, podemos escolher por validação cruzada

Se queremos visualizar os dados, podemos testar várias configurações. Elas podem trazer insights diferentes para os dados.

Uma heurística quando usamos o kernel Gaussiano:

$$h \approx \text{mediana}_{i,j} d(\mathbf{x}_i, \mathbf{x}_j)$$

Como escolher os "tuning parameters"?

Como escolher qual kernel usar?

Depende da aplicação!

Ex: se estamos usando esses componentes como primeiro passo para classificação, podemos escolher por validação cruzada

Se queremos visualizar os dados, podemos testar várias configurações. Elas podem trazer insights diferentes para os dados.

Uma heurística quando usamos o kernel Gaussiano:

$$h \approx \text{mediana}_{i,j} d(\mathbf{x}_i, \mathbf{x}_j)$$

Como escolher os "tuning parameters"?

Como escolher qual kernel usar?

Depende da aplicação!

Ex: se estamos usando esses componentes como primeiro passo para classificação, podemos escolher por validação cruzada

Se queremos visualizar os dados, podemos testar várias configurações. Elas podem trazer insights diferentes para os dados.

Uma heurística quando usamos o kernel Gaussiano:

$$h \approx \text{mediana}_{i,j} d(\mathbf{x}_i, \mathbf{x}_j)$$

Como escolher os "tuning parameters"?

Como escolher qual kernel usar?

Depende da aplicação!

Ex: se estamos usando esses componentes como primeiro passo para classificação, podemos escolher por validação cruzada

Se queremos visualizar os dados, podemos testar várias configurações. Elas podem trazer insights diferentes para os dados.

Uma heurística quando usamos o kernel Gaussiano:

$$h \approx \text{mediana}_{i,j} d(\mathbf{x}_i, \mathbf{x}_j)$$

Como escolher os "tuning parameters"?

Como escolher qual kernel usar?

Depende da aplicação!

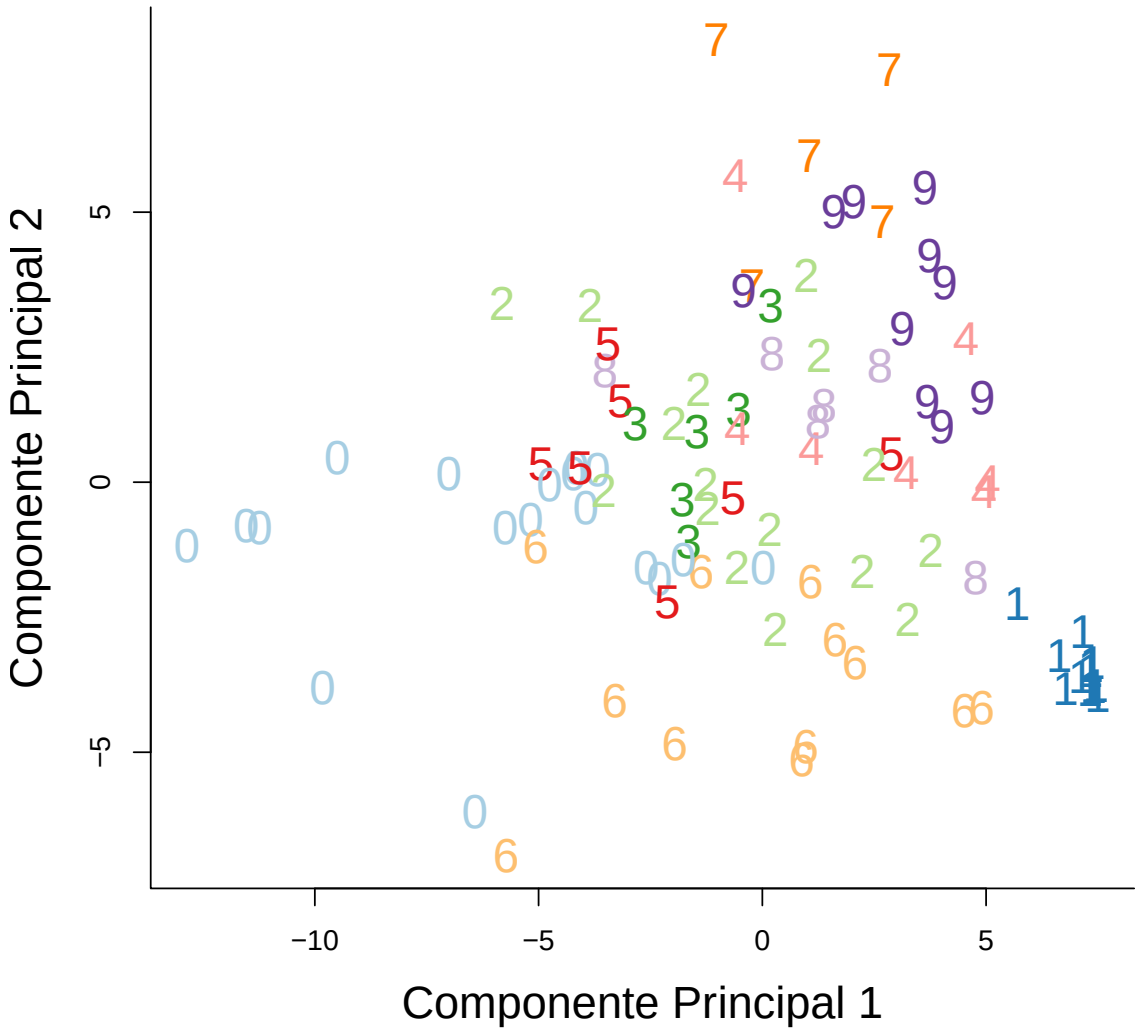
Ex: se estamos usando esses componentes como primeiro passo para classificação, podemos escolher por validação cruzada

Se queremos visualizar os dados, podemos testar várias configurações. Elas podem trazer insights diferentes para os dados.

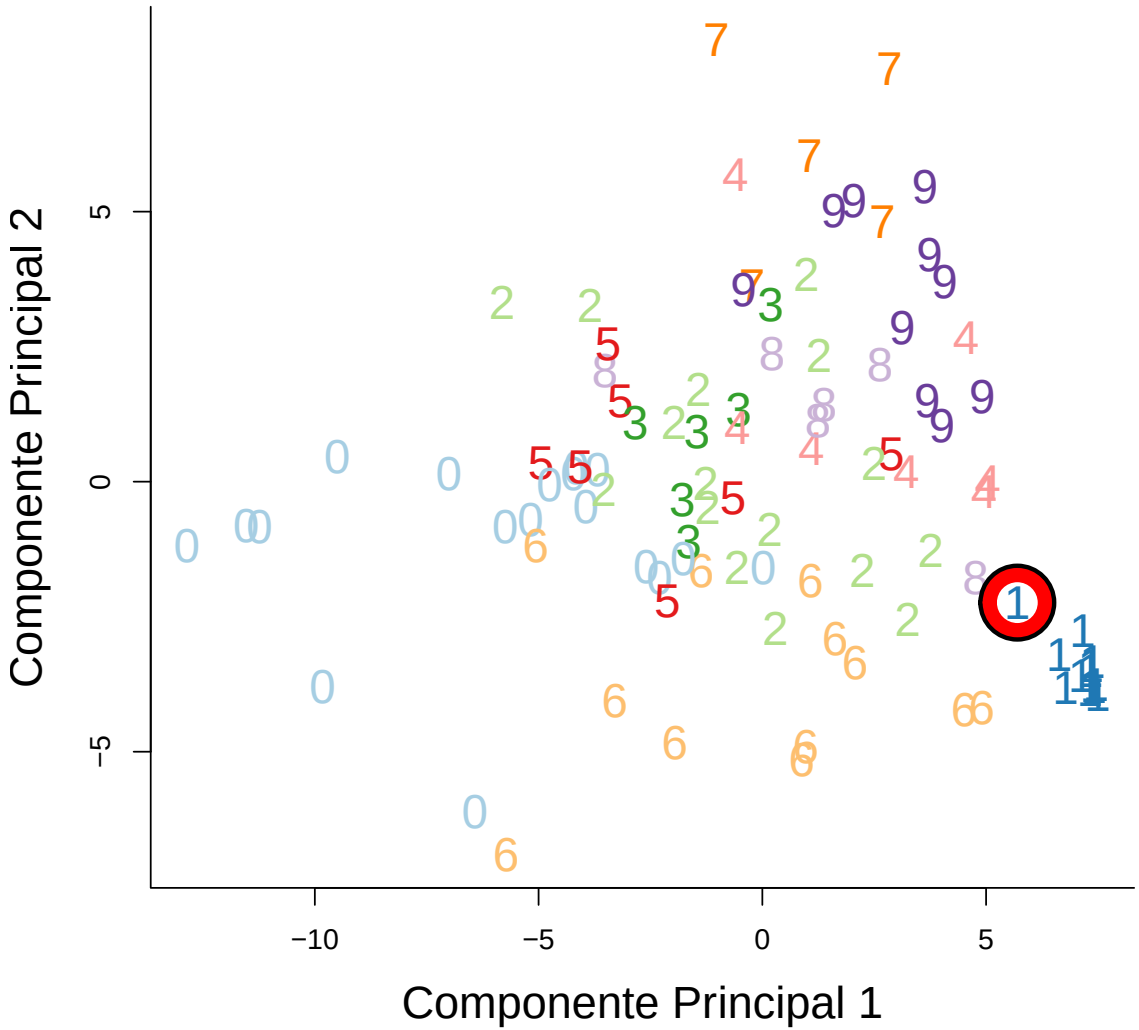
Uma heurística quando usamos o kernel Gaussiano:

$$h \approx \text{mediana}_{i,j} d(\mathbf{x}_i, \mathbf{x}_j)$$

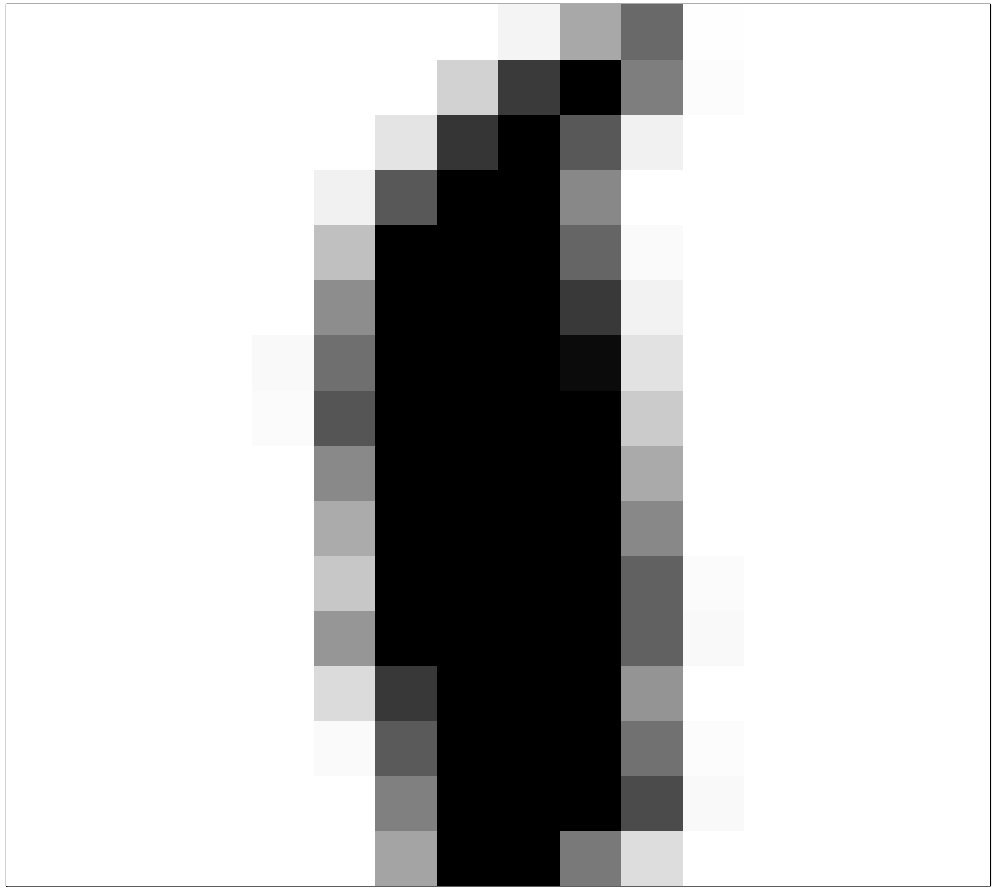
Exemplo de aplicação



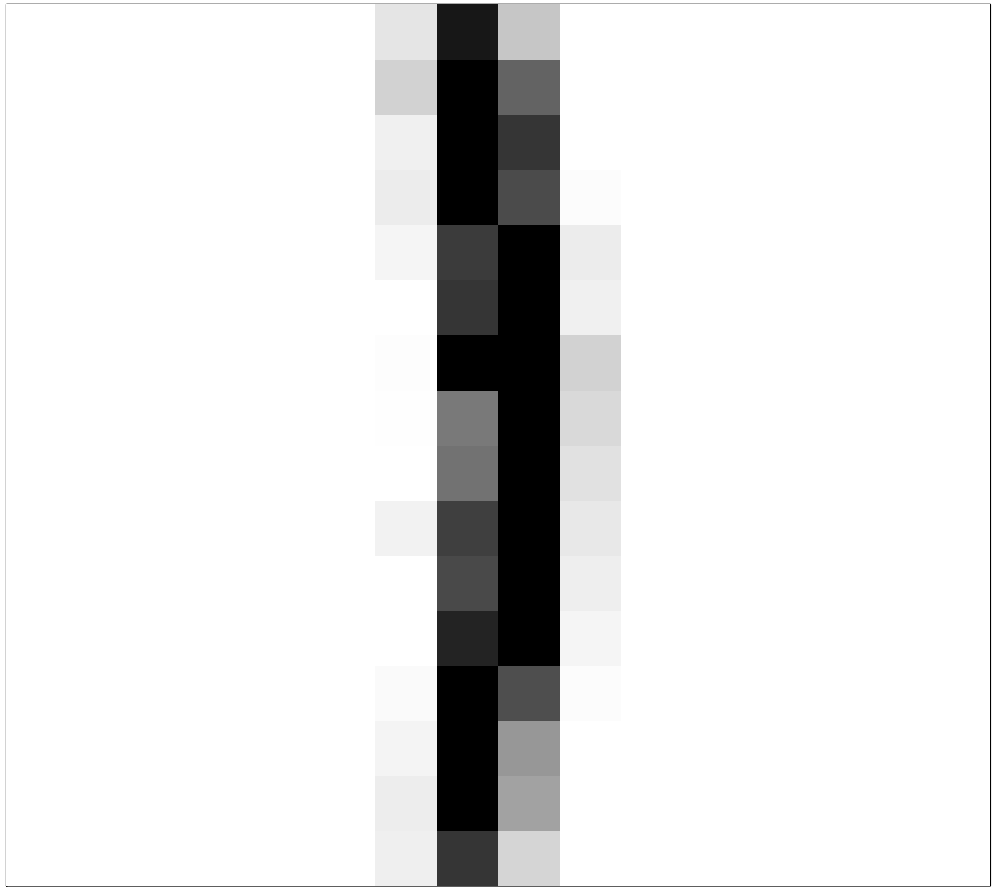
Cada ponto é uma observação, que representamos usando o dígito correspondente.


Componente Principal 2
5
0
−5
−10
−5
0
5
Componente Principal 1

Dígito outlier



Dígito inlier



Projeções aleatória

z_i consiste em uma projeção linear das covariáveis originais com coeficientes escolhidos aleatoriamente

S uma matrix com m linhas e d colunas cujas entradas são amostras i.i.d. de normais com média zero e variância um. Seja $s_{i,k}$ a entrada (i, j) desta matriz.

$$z_i = \sum_{k=1}^{d} \frac{s_{i,k}}{\sqrt{m}} x_k .$$

Projeções aleatória

z_i consiste em uma projeção linear das covariáveis originais com coeficientes escolhidos aleatoriamente

S uma matrix com m linhas e d colunas cujas entradas são amostras i.i.d. de normais com média zero e variância um. Seja $s_{i,k}$ a entrada (i, j) desta matriz.

$$z_i = \sum_{k=1}^{d} \frac{s_{i,k}}{\sqrt{m}} x_k .$$

Projeções aleatória

z_i consiste em uma projeção linear das covariáveis originais com coeficientes escolhidos aleatoriamente

S uma matrix com m linhas e d colunas cujas entradas são amostras i.i.d. de normais com média zero e variância um. Seja $s_{i,k}$ a entrada (i, j) desta matriz.

$$z_i = \sum_{k=1}^{d} \frac{s_{i,k}}{\sqrt{m}} x_k .$$

Projeções aleatória

z_i consiste em uma projeção linear das covariáveis originais com coeficientes escolhidos aleatoriamente

S uma matrix com m linhas e d colunas cujas entradas são amostras i.i.d. de normais com média zero e variância um. Seja $s_{i,k}$ a entrada (i, j) desta matriz.

$$z_i = \sum_{k=1}^{d} \frac{s_{i,k}}{\sqrt{m}} x_k .$$

Seja $\mathbf{z}_i = (z_{i,1}, \ldots, z_{i,m})$ o vetor composto pelas novas variáveis.

Theorem (Johnson-Lindenstrauss)

Fixe $\epsilon > 0$ e seja $m \geq 32 \log n/\epsilon^2$. Então, com probabilidade ao menos $1 - e^{-m\epsilon^2/16}$, vale que

$$(1-\epsilon)||\mathbf{x}_i - \mathbf{x}_j||^2 \leq ||\mathbf{z}_i - \mathbf{z}_j||^2 \leq (1+\epsilon)||\mathbf{x}_i - \mathbf{x}_j||^2$$

para todos $i, j = 1, \ldots, n$.

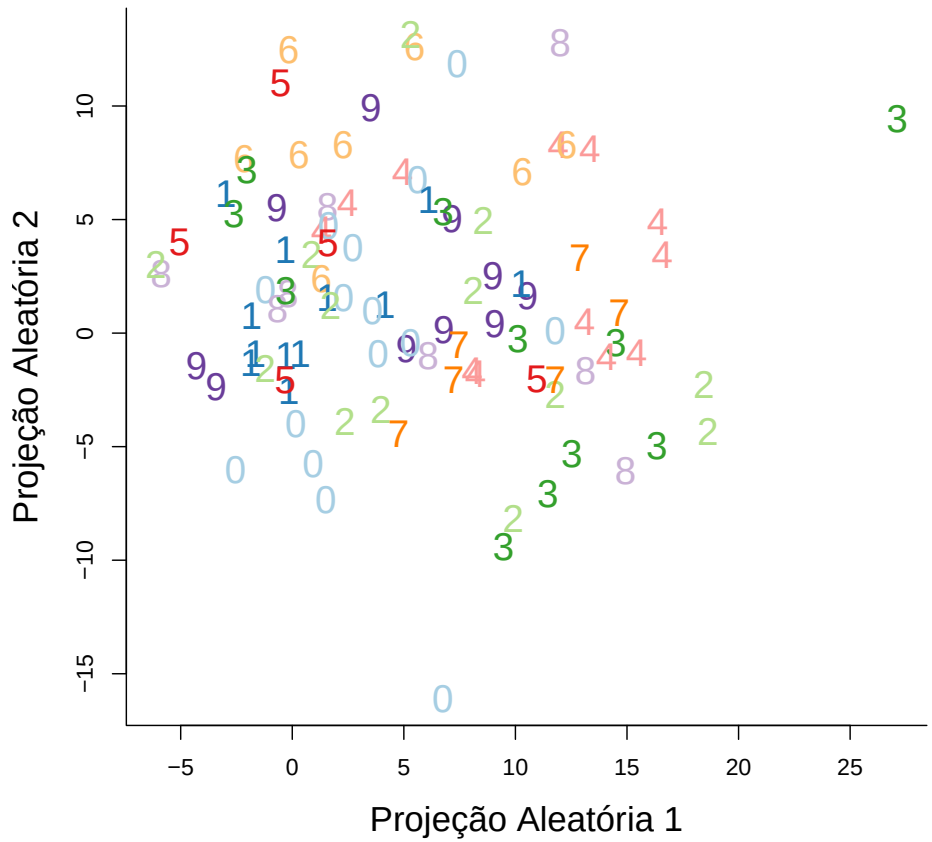
Seja $\mathbf{z}_i = (z_{i,1}, \ldots, z_{i,m})$ o vetor composto pelas novas variáveis.

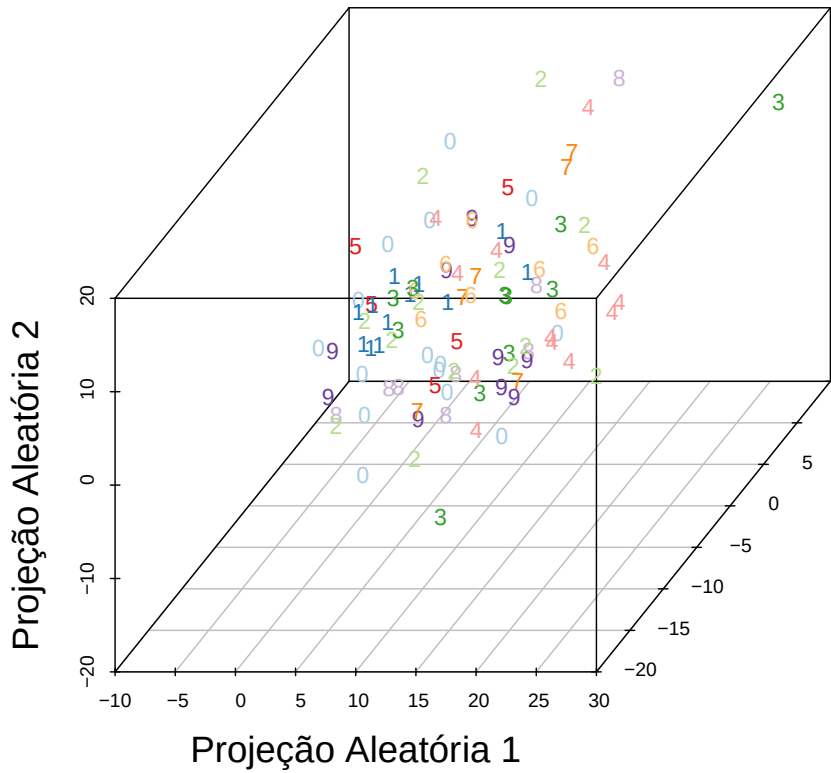
Theorem (Johnson-Lindenstrauss)

Fixe $\epsilon > 0$ e seja $m \geq 32 \log n/\epsilon^2$. Então, com probabilidade ao menos $1 - e^{-m\epsilon^2/16}$, vale que

$$(1-\epsilon)||\mathbf{x}_i - \mathbf{x}_j||^2 \leq ||\mathbf{z}_i - \mathbf{z}_j||^2 \leq (1+\epsilon)||\mathbf{x}_i - \mathbf{x}_j||^2$$

para todos $i, j = 1, \ldots, n$.





Duas (esquerda) e três (direita) primeiras projeções aleatórias calculadas no conjunto de dados de dígitos escritos à mão.

$\Rightarrow$ Forma de deixar diversos algoritmos mas rápidos.

$\Rightarrow$ Métodos regressão e classificação que exigem apenas o cálculo do produto interno para serem implementados podem ser aproximados sem grandes perdas de performance estatística.

$\Rightarrow$ Forma de deixar diversos algoritmos mas rápidos.

$\Rightarrow$ Métodos regressão e classificação que exigem apenas o cálculo do produto interno para serem implementados podem ser aproximados sem grandes perdas de performance estatística.

Resumindo

PCA: encontrar um número pequeno de transformações lineares dos dados que contenham quase toda informação presente nas covariáveis originais

KPCA: Idem, mas usamos o truque do kernel para achar transformações não lineares

Resumindo

PCA: encontrar um número pequeno de transformações lineares dos dados que contenham quase toda informação presente nas covariáveis originais

KPCA: Idem, mas usamos o truque do kernel para achar transformações não lineares