Exemplo de Aplicação de Transformações Lineares: Análise das Componentes Principais

Guilherme de Alencar Barreto

gbarreto@ufc.br

Departamento de Engenharia de Teleinformática (DETI)
Engenharias de Computação, Telecomunicações e Teleinformática
Universidade Federal do Ceará – UFC
www.researchgate.net/profile/Guilherme_Barreto2/

Conteúdo dos Slides

Transformadas Matriciais

Para cada $\mathbf{x} \in \mathbb{R}^{p \times 1}$, uma transformada matricial é definida por

$$\mathbf{z} = \mathbf{W}\mathbf{x} \quad (\text{ou} \quad \mathbf{W}\mathbf{x} = \mathbf{z}), \tag{1}$$

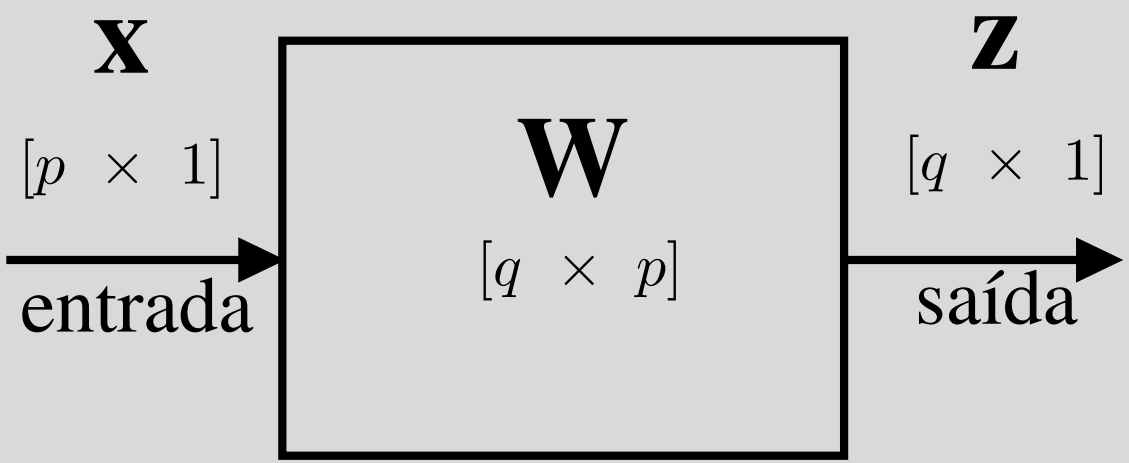
em que $\mathbf{W}$ é uma matriz $q \times p$.

- Para simplificar, muitas vezes denotamos essa transformação matricial por

$$\boxed{\mathbf{x} \mapsto \mathbf{W}\mathbf{x}} \tag{2}$$

Diagrama de Blocos

Ajuda muito no entendimento de uma transformação linear se representarmos a relações $\mathbf{z} = \mathbf{W}\mathbf{x}$ na forma de um diagrama de blocos do tipo entrada-saída.



Formalização Matemática do Problema

- Considere um conjunto de dados formado por N vetores de atributos $\mathbf{x}_k$, $k = 1, \ldots, N$, que estão organizados ao longo das colunas da matriz $\mathbf{X} \in \mathbb{R}^{p \times N}$:

$$\mathbf{X} = \begin{bmatrix} | & | & & | \\ \mathbf{x}_1 & \mathbf{x}_2 & \cdots & \mathbf{x}_N \\ | & | & & | \end{bmatrix}, \tag{3}$$

com cada vetor de atributo $\mathbf{x}_k \in \mathbb{R}^{p \times 1}$ dado por

$$\mathbf{x}_k = \begin{bmatrix} x_{1,k} \\ x_{2,k} \\ \vdots \\ x_{p,k} \end{bmatrix}, \quad \text{tal que } p \to \infty. \tag{4}$$

Formalização Matemática do Problema

- Deseja-se transformar cada vetor $\mathbf{x}_k \in \mathbb{R}^{p\times 1}$ no conjunto de dados em um outro vetor $\mathbf{z}_k \in \mathbb{R}^{q\times 1}$ de dimensão q, ou seja

$$\mathbf{z}_k = \begin{bmatrix} z_{1,k} \\ z_{2,k} \\ \vdots \\ z_{q,k} \end{bmatrix}, \quad \text{tal que } q \leq p. \tag{5}$$

- Isto deve ser feito por meio de uma transformação linear:

$$\mathbf{z}_k = \mathbf{Q}\mathbf{x}_k, \quad \forall k = 1, \ldots, N. \tag{6}$$

- Além disso, esta transformação deve preservar a informação **relevante** constante no conjunto $\mathbf{X}$.

Perspectiva Histórica e Vários Nomes para Mesma Técnica

- Desenvolvido por Harold Hotelling em 1933, que cunhou também o termo PCA, sendo também conhecida como *Transformada de Hotelling*.
  - *H. Hotelling (1933). "Analysis of a complex of statistical variables into principal components", Journal of Educational Psychology, 24(7):498-520.*
- Também conhecida como *Transformada de Karhunen–Loève* em processamento de sinais.
  - *K. Karhunen (1947). "Über lineare Methoden in der Wahrscheinlichkeitsrechnung", Ann. Acad. Sci. Fennicae. Ser. A. I. Math.-Phys., No. 37, 179.*
  - *M. Loève (1955). "Probability Theory". Princeton, New Jersey, USA: D Van Nostrand.*
- É um caso particular da *Decomposição em Valores Singulares* (SVD, sigla do inglês), quando a matriz a ser decomposta é a matriz de covariância dos dados (quadrada, simétrica e definida positiva).
  - *G. W. Stewart (1993). "On the early history of the singular value decomposition". SIAM Review. 35 (4): 551566.*

Algoritmo PCA: Passo-a-Passo

- **Passo 1** - Determinar o vetor-médio dos dados em $\mathbf{X}$:

$$\bar{\mathbf{x}} = \frac{1}{N}\sum_{k=1}^{N}\mathbf{x}_k. \tag{7}$$

- **Passo 2** - Centralizar os dados: $\mathbf{x}_k = \mathbf{x}_k - \bar{\mathbf{x}}$.
- **Passo 3** - Estimar a matriz de covariância dos dados em $\mathbf{X}$:

$$\mathbf{C_x} = E[\mathbf{x}\mathbf{x}^T], \tag{8}$$

$$\approx \frac{1}{N}\sum_{k=1}^{N}\mathbf{x}_k\mathbf{x}_k^T, \tag{9}$$

em que $E[\cdot]$ é o operador valor esperado.

Algoritmo PCA: Passo-a-Passo

- **Passo 4** - Determinar os p autovalores da matriz $\mathbf{C_x} \in \mathbb{R}^{p \times p}$ e os p autovetores correspondentes. Em outras palavras, resolver o seguinte sistema de equações:

$$\mathbf{C_x v} = \lambda \mathbf{v}, \tag{10}$$

em que $\lambda > 0$ e $\mathbf{v} \in \mathbb{R}^{p \times 1}$ denotam, respectivamente, o autovalor e o autovetor associado.

- Os autovalores são as raízes do polinômio em λ, de ordem p, obtido a partir da seguinte expressão:

$$\det(\mathbf{C_x} - \lambda \mathbf{I}_p) = 0. \tag{11}$$

Algoritmo PCA: Pausa para Explicação

- A Eq. (10) trata de um problema clássico em sistemas lineares: o problema do autovalor.
- Este é um caso particular do problema mais geral de transformações lineares do tipo $\mathbf{Ax} = \mathbf{b}$.
- Lembre-se que a operação matricial $\mathbf{Ax}$ produz um vetor $\mathbf{b}$.
- Para gerar o vetor $\mathbf{b}$, o vetor $\mathbf{x}$ tem sua norma e/ou orientação modificados pela multiplicação pela matriz $\mathbf{A}$.
- O problema do autovalor é um caso particular de transformação linear em que $\mathbf{b} = \lambda\mathbf{x}$.
- Ou seja, a matriz $\mathbf{A}$ altera apenas a norma de $\mathbf{x}$, pois gera um vetor múltiplo de $\mathbf{x}$.

Algoritmo PCA: Pausa para Explicação

- Por ser uma matriz simétrica e definida positiva, os p autovalores são sempre positivos.
- Para resolver a Eq. (10), a escrevemos como um sistema homogêneo:

$$\mathbf{C_x v} = \lambda \mathbf{v} = \lambda \mathbf{I}_p \mathbf{v} \quad \Rightarrow \quad (\mathbf{C_x} - \lambda \mathbf{I}_p)\mathbf{v} = \mathbf{0}_p, \qquad (12)$$

  em que $\mathbf{0}_p$ é um vetor de zeros de dimensão $p \times 1$.
- Este sistema tem uma única solução $\mathbf{v} = \mathbf{0}_p$, chamada de trivial, se o determinante de $\mathbf{C_x} - \lambda \mathbf{I}_p$ for diferente de zero.
- A solução trivial não nos interessa. Logo, buscamos as soluções não-triviais, ou seja, aquelas para as quais o determinante de $\mathbf{C_x} - \lambda \mathbf{I}_p$ é nulo. Vide Eq. (11).

Algoritmo PCA: Pausa para Explicação

- A Eq. (11) resulta em uma equação polinomial em λ de ordem p. Os autovalores são as raízes deste polinômio.
- Suponha que a matriz $\mathbf{C_x}$ é dada por

$$\mathbf{C_x} = \begin{bmatrix} 1 & 0,8 \\ 0,8 & 4 \end{bmatrix}$$

- Neste caso, a matriz $\mathbf{C_x} - \lambda\mathbf{I_2}$ é dada por

$$\begin{bmatrix} 1 & 0,8 \\ 0,8 & 4 \end{bmatrix} - \begin{bmatrix} \lambda & 0 \\ 0 & \lambda \end{bmatrix} = \begin{bmatrix} 1-\lambda & 0,8 \\ 0,8 & 4-\lambda \end{bmatrix}$$

Algoritmo PCA: Pausa para Explicação

- Igualando seu determinante a zero, obtemos a seguinte equação polinomial:

$$p(\lambda) = (1-\lambda)(4-\lambda)-(0,8)(0,8) = \lambda^2-5\lambda+3,36 = 0. \quad (13)$$

- Cujas raízes são $\lambda_1 = 0,8$ e $\lambda_2 = 4,2$.
- Em Octave/Matlab, as raízes de $p(\lambda)$ podem ser encontradas por meio do seguinte comando:

```
» roots([1 -5 3.36])
```

Algoritmo PCA: Passo-a-Passo

- Como a matriz $\mathbf{C_x} \in \mathbb{R}^{p \times p}$ é positiva definida, todos os seus p autovalores são positivos e distintos.
- Os autovalores devem ser ordenados em ordem decrescente de suas magnitudes:

$$\lambda_1 > \lambda_2 > \lambda_3 > \cdots > \lambda_p \tag{14}$$

- Os autovetores são determinados resolvendo-se o sistema na Eq. (10) p vezes, uma para cada autovalor:

$$\mathbf{C_x}\mathbf{v}_i = \lambda_i \mathbf{v}_i, \quad i = 1, \ldots, p. \tag{15}$$

em que λ_i e $\mathbf{v}_i \in \mathbb{R}^{p \times 1}$ são, respectivamente, o i-ésimo autovalor e o autovetor associado.

Algoritmo PCA: Passo-a-Passo

- Note que se $\mathbf{v}_i$ é uma solução da Eq. (15), então o seu vetor oposto $-\mathbf{v}_i$ também é, como é facilmente verificável abaixo.

$$\begin{aligned} \mathbf{C_x}(-\mathbf{v}_i) &= \lambda_i(-\mathbf{v}_i), & (16)\\ -\mathbf{C_x}\mathbf{v}_i &= -\lambda_i\mathbf{v}_i, & (17)\\ \mathbf{C_x}\mathbf{v}_i &= \lambda_i\mathbf{v}_i. & (18) \end{aligned}$$

- Esta propriedade faz com que, a depender do algoritmo numérico usado para resolver a Eq. (15), o conjunto de autovetores encontrados difira entre si apenas pelos sinais.
- Por exemplo, compare o conjunto de autovetores retornados as funções `eig` e `pcacov` do Octave/Matlab.

Algoritmo PCA: Passo-a-Passo

- Os autovetores $\mathbf{v}_i \in \mathbb{R}^{p\times 1}$, $i = 1, \ldots, p$, formam um conjunto de vetores **ortonormais**:

$$\mathbf{v}_i^T \mathbf{v}_j = \|\mathbf{v}_i\| \cdot \|\mathbf{v}_j\| \cdot \cos(\theta) = \begin{cases} 1, & \text{se } i = j \ (\theta = 0) \\ 0, & \text{se } i \neq j \ (\theta = \pi/2) \end{cases} \tag{19}$$

- Dispor os autovetores ao longo das colunas da matriz $\mathbf{V}$:

$$\mathbf{V} = \begin{bmatrix} | & | & & | \\ \mathbf{v}_1 & \mathbf{v}_2 & \cdots & \mathbf{v}_p \\ | & | & & | \end{bmatrix}_{p\times p} \tag{20}$$

- A matriz $\mathbf{V}$ é quadrada e de dimensões $p \times p$.

Algoritmo PCA: Passo-a-Passo

- A ortonormalidade de $\mathbf{V}$ pode ser verificada matricialmente por meio da seguinte operação: $\mathbf{V}\mathbf{V}^T = \mathbf{I}_p$, em que $\mathbf{I}_p$ é a matriz identidade de dimensões $p \times p$.
- Ao analisarmos a expressão anterior, nota-se que a transposta da matriz $\mathbf{V}$ é a sua inversa:

$$\mathbf{V}^{-1} = \mathbf{V}^T. \tag{21}$$

- Essa propriedade é resultado da ortogonalidade dos vetores que formam suas colunas e será útil na reconstrução dos dados originais a partir dos dados transformados, ou seja, através do mapeamento inverso de $\mathbf{z}_k$ para $\mathbf{x}_k$.

Algoritmo PCA: Passo-a-Passo

- **Passo 5** - Definir a matriz de transformação $\mathbf{Q}$ como

$$\mathbf{Q} = \mathbf{V}^T. \tag{22}$$

- **Passo 6** - Aplicar a matriz $\mathbf{Q}$ sobre os vetores de atributos originais. Isto pode ser feito vetor a vetor:

$$\mathbf{z}_k = \mathbf{Q}\mathbf{x}_k, \tag{23}$$

para $k = 1, \ldots, N$. Ou, de forma matricial, em uma única operação dada por

$$\mathbf{Z} = \mathbf{Q}\mathbf{X}. \tag{24}$$

- O mapeamento inverso é então dado por $\mathbf{x}_k = \mathbf{Q}^T\mathbf{z}_k$ ou $\mathbf{X} = \mathbf{Q}^T\mathbf{Z}$.

- Note que a aplicação da matriz $\mathbf{Q}$ sobre o vetor de atributos original $\mathbf{x}_k$ produz um novo vetor de atributos $\mathbf{z}_k$.
- Sem perda de generalidade, o índice k é retirado para deixar a notação mais clara.
- Pode-se escrever a transformação linear $\mathbf{z} = \mathbf{V}^T\mathbf{x}$ em sua forma escalar como um sistema de equações lineares:

$$\begin{bmatrix} z_1 \\ z_2 \\ \vdots \\ z_i \\ \vdots \\ z_p \end{bmatrix} = \begin{bmatrix} v_{11} & v_{21} & \cdots & v_{i1} & \cdots & v_{p1} \\ v_{12} & v_{22} & \cdots & v_{i2} & \cdots & v_{p2} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ v_{1i} & v_{2i} & \cdots & v_{ii} & \cdots & v_{pi} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ v_{1p} & v_{2p} & \cdots & v_{ip} & \cdots & v_{pp} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_i \\ \vdots \\ x_p \end{bmatrix} = \begin{bmatrix} \mathbf{v}_1^T\mathbf{x} \\ \mathbf{v}_2^T\mathbf{x} \\ \vdots \\ \mathbf{v}_i^T\mathbf{x} \\ \vdots \\ \mathbf{v}_p^T\mathbf{x} \end{bmatrix} \quad (25)$$

- Tomemos a i-ésima componente de $\mathbf{z}$ na Eq. (25):

$$z_i = \mathbf{v}_i^T \mathbf{x} = \begin{bmatrix} v_{1i} & v_{2i} & \cdots & v_{ii} & \cdots & v_{pi} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_i \\ \vdots \\ x_p \end{bmatrix}, \tag{26}$$

$$= v_{1i}x_1 + v_{2i}x_2 + \cdots + v_{ii}x_i + \cdots + v_{pi}x_p \tag{27}$$

- Assim, a i-ésima componente de $\mathbf{z}$ é uma combinação linear dos atributos de $\mathbf{x}$, com as componentes do i-ésimo autovetor $\mathbf{v}_i$ sendo os coeficientes de ponderação.
- Resumindo, z_i é o produto escalar de $\mathbf{v}_i$ e $\mathbf{x}$.

Diagonalização da Matriz $\mathbf{C_x}$

- Ao aplicar a matriz $\mathbf{Q}$ sobre os vetores de atributos originais, da forma como está definida na Eq. (23), percebemos que as dimensões dos vetores $\mathbf{x}_k$ e $\mathbf{z}_k$ são iguais (i.e. $p = q$). Portanto, não temos aqui uma redução de dimensionalidade.
- O que, então, acontece com os vetores de dados?
- Que tipo de transformação eles sofreram?
- Para responder estas questões, vamos precisar calcular a matriz de covariância dos dados transformados, ou seja, precisamos determinar

$$\mathbf{C_z} = E[\mathbf{z}\mathbf{z}^T]. \tag{28}$$

Diagonalização da Matriz $\mathbf{C_x}$

- A partir da Eq. (28) e da Eq. (24), obtemos:

$$\begin{aligned}
\mathbf{C_z} &= E[\mathbf{z}\mathbf{z}^T] && (29)\\
&= E[(\mathbf{V}^T\mathbf{x})\,(\mathbf{V}^T\mathbf{x})^T] && (30)\\
&= E[(\mathbf{V}^T\mathbf{x})\,(\mathbf{x}^T\mathbf{V})] && (31)\\
&= E[\mathbf{V}^T\,(\mathbf{x}\mathbf{x}^T)\,\mathbf{V}] && (32)\\
&= \mathbf{V}^T E[\mathbf{x}\mathbf{x}^T]\mathbf{V} && (33)\\
&= \mathbf{V}^T\mathbf{C_x}\mathbf{V} && (34)
\end{aligned}$$

- Este resultado mostra que dada a matriz de covariância dos dados originais $\mathbf{C_x}$ e a matriz de autovetores $\mathbf{V}$, facilmente obtemos a matriz de covariância dos dados transformados $\mathbf{C_z}$.

Diagonalização da Matriz $\mathbf{C_x}$

- Contudo, não diz muita coisa sobre a forma da matriz de covariância dos dados transformados. Para isso, vamos expandir os produtos matriz-vetor da Eq. (34), começando pelo produto $\mathbf{C_x V}$:

$$\mathbf{C_x V} = \mathbf{C_x} \begin{bmatrix} | & | & & | \\ \mathbf{v}_1 & \mathbf{v}_2 & \cdots & \mathbf{v}_p \\ | & | & & | \end{bmatrix} \tag{35}$$

$$= \begin{bmatrix} | & | & & | \\ \mathbf{C_x v}_1 & \mathbf{C_x v}_2 & \cdots & \mathbf{C_x v}_p \\ | & | & & | \end{bmatrix}_{p \times p} \tag{36}$$

$$= \begin{bmatrix} | & | & & | \\ \lambda_1 \mathbf{v}_1 & \lambda_2 \mathbf{v}_2 & \cdots & \lambda_p \mathbf{v}_p \\ | & | & & | \end{bmatrix}_{p \times p} \tag{37}$$

usando o fato que $\mathbf{C_x v}_i = \lambda_i \mathbf{v}_i$.

Diagonalização da Matriz $\mathbf{C_x}$

- Agora, lembrando que $\mathbf{V}^T$ também é uma matriz $p \times p$, podemos realizar a segunda parte do produto:

$$\mathbf{V}^T\mathbf{C_x}\mathbf{V} = \underbrace{\begin{bmatrix} — & \mathbf{v}_1^T & — \\ — & \mathbf{v}_2^T & — \\ & \vdots & \\ — & \mathbf{v}_p^T & — \end{bmatrix}}_{p\times p} \underbrace{\begin{bmatrix} | & | & & | \\ \lambda_1\mathbf{v}_1 & \lambda_2\mathbf{v}_2 & \cdots & \lambda_p\mathbf{v}_p \\ | & | & & | \end{bmatrix}}_{p\times p} \tag{38}$$

Diagonalização da Matriz $\mathbf{C_x}$

- Chegamos à seguinte expressão da matriz de covariância $\mathbf{C_z}$:

$$\mathbf{C_z} = \mathbf{V}^T\mathbf{C_x}\mathbf{V} \tag{39}$$

$$= \begin{bmatrix} \lambda_1\mathbf{v}_1^T\mathbf{v}_1 & \lambda_2\mathbf{v}_1^T\mathbf{v}_2 & \cdots & \lambda_p\mathbf{v}_1^T\mathbf{v}_p \\ \lambda_1\mathbf{v}_2^T\mathbf{v}_1 & \lambda_2\mathbf{v}_2^T\mathbf{v}_2 & \cdots & \lambda_p\mathbf{v}_2^T\mathbf{v}_p \\ \vdots & \vdots & \ddots & \vdots \\ \lambda_1\mathbf{v}_p^T\mathbf{v}_1 & \lambda_2\mathbf{v}_p^T\mathbf{v}_2 & \cdots & \lambda_p\mathbf{v}_p^T\mathbf{v}_p \end{bmatrix}_{p\times p} \tag{40}$$

$$= \begin{bmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0\cdots & 0 & \lambda_p \end{bmatrix}_{p\times p} \tag{41}$$

Diagonalização da Matriz $\mathbf{C_x}$

- O produto $\mathbf{V}^T\mathbf{C_x}\mathbf{V}$ pode também ser desenvolvido em função dos elementos de $\mathbf{C_x}$.

$$\mathbf{C_z} = \mathbf{V}^T\mathbf{C_x}\mathbf{V} \tag{42}$$

$$= \begin{bmatrix} \sigma_1^2\mathbf{v}_1^T\mathbf{v}_1 & \sigma_{12}\mathbf{v}_1^T\mathbf{v}_2 & \cdots & \sigma_{1n}\mathbf{v}_1^T\mathbf{v}_p \\ \sigma_{21}\mathbf{v}_2^T\mathbf{v}_1 & \sigma_2^2\mathbf{v}_2^T\mathbf{v}_2 & \cdots & \sigma_{2n}\mathbf{v}_2^T\mathbf{v}_p \\ \vdots & \vdots & \ddots & \vdots \\ \sigma_{n1}\mathbf{v}_p^T\mathbf{v}_1 & \sigma_{n2}\mathbf{v}_p^T\mathbf{v}_2 & \cdots & \sigma_p^2\mathbf{v}_p^T\mathbf{v}_p \end{bmatrix}_{p\times p} \tag{43}$$

$$= \begin{bmatrix} \sigma_1^2 & 0 & \cdots & 0 \\ 0 & \sigma_2^2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0\cdots & 0 & \sigma_p^2 \end{bmatrix}_{p\times p} \tag{44}$$

Diagonalização da Matriz $\mathbf{C_x}$

- Resumindo, temos que a matriz de covariância dos dados transformados $\mathbf{C_z} = \mathbf{V}^T\mathbf{C_x}\mathbf{V}$ tem a seguinte forma:

$$\mathbf{C_z} = \begin{bmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0\cdots & 0 & \lambda_p \end{bmatrix} = \begin{bmatrix} \sigma_1^2 & 0 & \cdots & 0 \\ 0 & \sigma_2^2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0\cdots & 0 & \sigma_p^2 \end{bmatrix} \quad (45)$$

Diagonalização da Matriz $\mathbf{C_x}$

- Do exposto na Eq. (45), tiramos as seguintes conclusões:
  1. A matriz de covariância dos dados transformados é diagonal, ou seja, não há correlação entre as componentes do vetor $\mathbf{z}$.
  2. Ou seja, PCA atua sobre os dados em $\mathbf{X}$ para gerar um novo conjunto de dados $\mathbf{Z}$, cuja matriz de covariância é diagonal.
  3. As variâncias das variáveis transformadas são iguais aos autovalores de $\mathbf{C_x}$.
  4. Os autovalores, por sua vez, são iguais às variâncias das variáveis originais.
- Uma consequência imediata desses resultados é que não é necessário calcular os autovalores da matriz $\mathbf{C_z}$, já que eles são iguais às variâncias das variáveis originais!!

Interpretação Geométrica

- A transformação linear do PCA pode ser entendida como uma mudança de base. Uma base no espaço $\mathbb{R}^n$ é qualquer conjunto de n vetores linearmente independentes (LI) usado para representar os vetores daquele espaço.
- Por exemplo, a base canônica do $\mathbb{R}^2$ é formada pelos vetores

$$\mathbf{e}_1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \quad \text{e} \quad \mathbf{e}_2 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}. \tag{46}$$

- Assim, qualquer vetor $[a \;\; b]^T$ no $\mathbb{R}^2$ pode ser escrito como

$$\begin{bmatrix} a \\ b \end{bmatrix} = a\mathbf{e}_1 + b\mathbf{e}_2 = a \begin{bmatrix} 1 \\ 0 \end{bmatrix} + b \begin{bmatrix} 0 \\ 1 \end{bmatrix}. \tag{47}$$

Interpretação Geométrica

- Uma outra possível base do $\mathbb{R}^2$ é formada pelos vetores

$$\mathbf{v}_1 = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \quad \text{e} \quad \mathbf{v}_2 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}. \tag{48}$$

- Neste caso, o vetor $[a \;\; b]^T$, originalmente escrito com relação à base $\mathcal{B}_1 = \{\mathbf{e}_1, \mathbf{e}_2\}$, com relação à base $\mathcal{B}_2 = \{\mathbf{v}_1, \mathbf{v}_2\}$ passa ser representado como

$$\begin{bmatrix} a \\ b \end{bmatrix} = a\mathbf{v}_1 + (b-a)\mathbf{v}_2 = a \begin{bmatrix} 0 \\ 1 \end{bmatrix} + (b-a) \begin{bmatrix} 1 \\ 1 \end{bmatrix} \tag{49}$$

$$\begin{bmatrix} a \\ b \end{bmatrix}_{\mathcal{B}_1} \equiv \begin{bmatrix} a \\ (b-a) \end{bmatrix}_{\mathcal{B}_2}. \tag{50}$$

Interpretação Geométrica

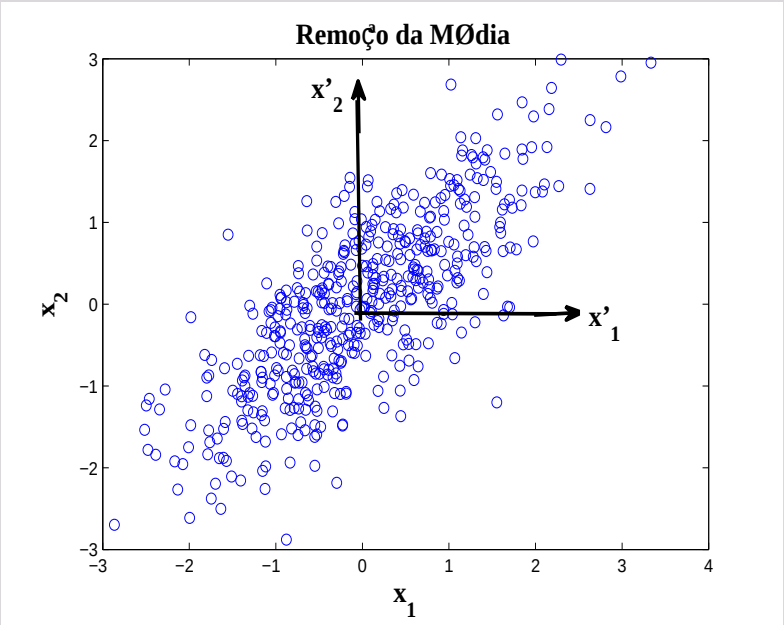
- Existem muitas outras possibilidades de se formar uma base no $\mathbb{R}^2$, bastando para isso que os vetores sejam LI.
- A base $\mathcal{B}_1$ é chamada de base ortonormal $\{\mathbf{e}_1, \mathbf{e}_2\}$, porque os vetores $\mathbf{e}_1$ e $\mathbf{e}_2$ são ortogonais (ou perpendiculares) entre si, pois seu produto interno é nulo.
- Além disso, ambos tem norma unitária (i.e. $\|\mathbf{e}_1\| = \|\mathbf{e}_2\| = 1$).
- Com relação à PCA, os vetores de dados originais $\mathbf{x}_k$ são escritos em relação à base canônica do $\mathbb{R}^p$.
- Enquanto os vetores transformados $\mathbf{z}_k$ são escritos em relação à base formada pelos autovetores da matriz de covariância $\mathbf{C}_\mathbf{x}$.

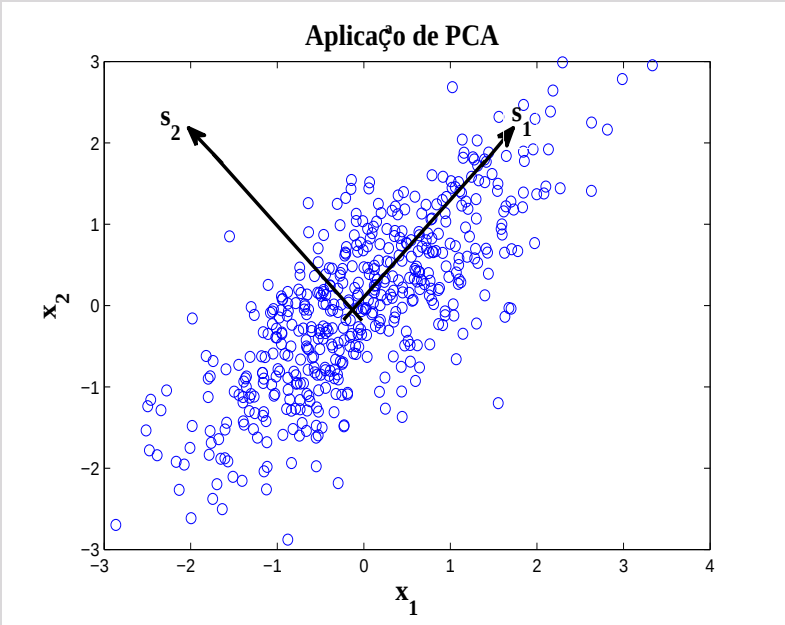
Interpretação Geométrica

- Lembre-se que no espaço original as componentes do vetor $\mathbf{x}_k$ estão correlacionadas, enquanto no espaço transformado as componentes do vetor $\mathbf{z}_k$ não.
- Dito de outra forma, no sistema de coordenadas perpendiculares associado à nova base formada pelos autovetores de $\mathbf{C_x}$, as componentes de $\mathbf{z}_k$ são descorrelacionadas.
- Do ponto de vista geométrico, o processo de descorrelação levado a cabo via PCA corresponde a uma rotação do sistema de coordenadas no qual os dados são representados.

Interpretação Geométrica

- Graficamente, o processo de diagonalização da matriz de covariância de um conjunto de dados, ou equivalente, de descorrelação dos atributos de um conjunto de dados, está mostrado na figura abaixo.





PCA para Seleção de Atributos

- Pode-se usar a primeira componente z_1 para verificar a importância de cada atributo do vetor $\mathbf{x}$ e selecionar os mais relevantes.
- Para isso, basta entender que a operação que gera o atributo $z_1 = \mathbf{v}_1^T \mathbf{x}$ depende do autovetor que define a direção (ou eixo) de maior variância dos dados. Podendo ser interpretada como uma projeção de $\mathbf{x}$ na direção de $\mathbf{v}_1$.
- Percebe-se os coeficientes v_{j1}, $j = 1, \ldots, p$, ponderam os atributos originais $\{x_1, x_2, \ldots, x_p\}$; ou seja

$$z_1 = v_{11}x_1 + v_{21}x_2 + v_{31}x_3 + \cdots + v_{p1}x_p \tag{51}$$

- Assim, quanto maior for o coeficiente v_{j1}, mais importante é a variável x_j, $j = 1, \ldots, p$.

PCA para Redução de Dimensão

- Usando apenas as q primeiras colunas de $\mathbf{V}$, obtemos

$$\mathbf{V}_q = \underbrace{\begin{bmatrix} | & | & & | \\ \mathbf{v}_1 & \mathbf{v}_2 & \cdots & \mathbf{v}_q \\ | & | & & | \end{bmatrix}}_{p \times q} \quad \Rightarrow \quad \mathbf{Q}_q = \mathbf{V}_q^T \tag{52}$$

- A matriz $\mathbf{Q}_q$ tem dimensão $q \times p$. O vetor $\mathbf{z}_k = \mathbf{Q}_q \mathbf{x}_k = \mathbf{V}_q^T \mathbf{x}_k$ terá dimensão $q \times 1$. E a matriz de covariância $\mathbf{C}_{\mathbf{z}}^{(q)}$ é dada por

$$\mathbf{C}_{\mathbf{z}}^{(q)} = \mathbf{V}_q^T \mathbf{C}_{\mathbf{x}} \mathbf{V}_q = \underbrace{\begin{bmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 \cdots & 0 & \lambda_q \end{bmatrix}}_{q \times q} \tag{53}$$

Uma Medida da Informação Contida em $\mathbf{X}$

- Lembrando que nosso objetivo inicial era encontrar uma transformação linear que preservasse a informação relevante contida nos dados originais. Mas, como quantificar a informação relevante em um conjunto de dados?

- Podemos definir a Variância Total (VT) como uma medida da quantidade de informação contida nos dados originais:

$$\begin{aligned} VT &= \sigma_1^2 + \sigma_2^2 + \cdots + \sigma_p^2 \quad (54) \\ &= \lambda_1 + \lambda_2 + \cdots + \lambda_p \quad (55) \end{aligned}$$

- Como $\lambda_i = \sigma_i^2$, podemos criar uma medida de quanto da informação (i.e. variância) do conjunto original está sendo representada no autovalor λ_i.

Uma Medida da Informação Contida em $\mathbf{X}$

- Chamaremos esta medida de variância explicada pelo i-ésimo autovalor (VE_i):

$$VE_i = 100 \times \frac{\lambda_i}{VT} = 100 \times \left(\frac{\lambda_i}{\lambda_1 + \lambda_2 + \cdots + \lambda_p} \right) \tag{56}$$

- Consequentemente, a porcentagem da variância total dos dados explicada pelos primeiros q autovalores é dada por:

$$VE(q) = 100 \times \frac{\sum_{i=1}^{q} \lambda_i}{VT} \tag{57}$$

$$= 100 \times \left(\frac{\lambda_1 + \lambda_2 + \cdots + \lambda_q}{\lambda_1 + \lambda_2 + \cdots + \lambda_p} \right) \tag{58}$$

- O gráfico $VE(q) \times q$ é usualmente chamado de *scree plot*.

Uma Medida da Informação Contida em $\mathbf{X}$

- A grandeza VT pode ser entendida como uma medida linear da informação total contida nos dados.
- Lembre-se que as únicas estatística relevante que sobrou com a transformação imposta aos dados pela equação $\mathbf{z} = \mathbf{V}^T\mathbf{x}$ foram as variâncias, na forma de autovalores na diagonal da matriz $\mathbf{C_z}$.
- As médias dos atributos originais foram eliminadas no Passo 2 do algoritmo do PCA. E as correlações que existiam entre os atributos x_i e x_j não existem mais entre os atributos z_i e z_j.
- Consequentemente, a grandeza $VE(q)$ é uma medida (linear) da informação dos dados devida aos q primeiros autovalores.
- Se $q = p$, não há redução de dimensionalidade; logo, não haverá compressão (perda) de informação e uma eventual reconstrução da entrada será perfeita.
- Se $q < p$, há redução de dimensionalidade; logo, haverá perda de informação e uma reconstrução da entrada não será perfeita.

Implementação em Matlab/Octave (cont.-1)

- Geração de dados sintéticos com a seguinte matriz de covariância:

$$\mathbf{C_x} = \begin{bmatrix} 1 & 1.8 & -0.9 \\ 1.8 & 4 & 0.6 \\ -0.9 & 0.6 & 9 \end{bmatrix} \tag{59}$$

```
» m1=5; sig1 = 1;    % Media/Desvio-padrao atributo 1
» m2=-5; sig2 = 1;    % Media/Desvio-padrao atributo 2
» m3=0; sig3 = 1;    % Media/Desvio-padrao atributo 3
» N = 5000;    % no.  amostras a gerar de cada atributo
» X1=normrnd(m1,sig1,1,N);    % gerar x1 ~ N(m1,1)
» X2=normrnd(m2,sig2,1,N);    % gerar x2 ~ N(m2,1)
» X3=normrnd(m3,sig3,1,N);    % gerar x3 ~ N(m3,1)
» Xu=[X1; X2; X3];    % Agrupa dados em uma matriz 3xN
» Cd=[1 1.8 -0.9;1.8 4 0.6;-0.9 0.6 9];    % Matriz desejada
```

Implementação em Matlab/Octave (cont.-2)

- Geração dos dados correlacionados via Decomposição de Cholesky.

```
» R=chol(Cd);    % Aplica Cholesky na matriz Cd
» Xc=R'*Xu;    % Gera dados correlacionados
```

- Assim, o passo seguinte consiste na estimação da matriz de covariância dos dados originais $\mathbf{C_x}$.

```
» Cx=cov(X');
```

- **Atenção**: A matriz $\mathbf{X}$ entra transposta no comando `COV` porque, por convenção, o Matlab considera que os dados estão dispostos ao longo das linhas de $\mathbf{X}$, e não ao longo das colunas.

Implementação em Matlab/Octave (cont.-3)

- Em seguida, estimam-se os autovalores/autovetores de $\mathbf{C_x}$.

```
» [V L]=eig(Cx);    % matrizes de autovetores/valores
» L=diag(L);    % vetor de autovalores nao-ordenados
» [L I]=sort(L,'descend');    % autovalores ordenados
» V=V(:,I);    % ordena autovetores associados
```

- **Atenção**: A função `EIG` retorna uma matriz "L" cujo os autovalores estão na diagonal principal. Daí a necessidade de se usar em seguida o comando `DIAG`, para extrair os autovalores e colocá-los em um vetor.

Implementação em Matlab/Octave (cont.-4)

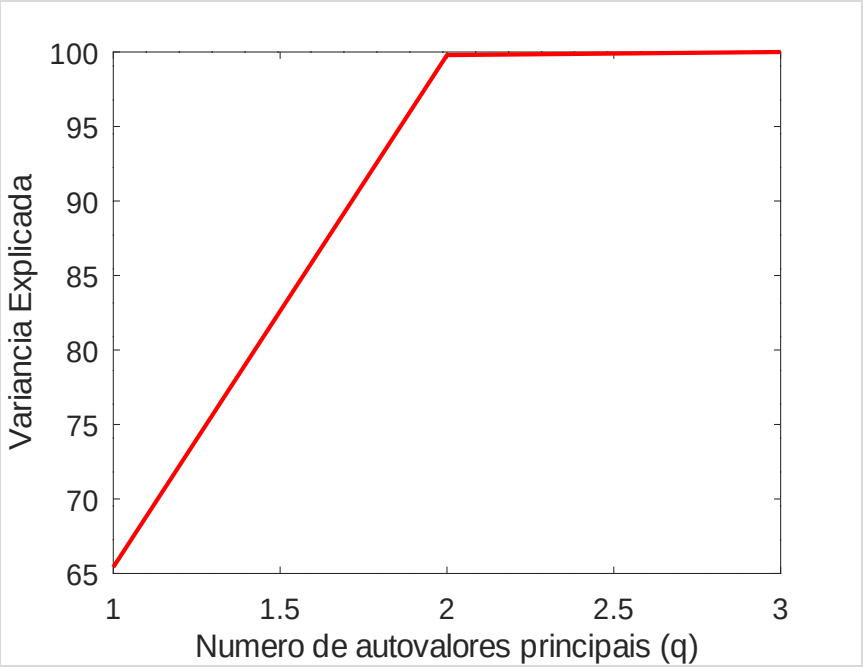
- O passo final consiste em determinar os q maiores autovalores responsáveis por explicar, pelo menos, `tol`% da informação contida nos dados originais.

```
» tol=0.95;    % Informacao a ser preservada
» VE=cumsum(L)/sum(L);    % variancia explicada
» q=length(find(VE<=tol));    % Num. compon. principais
» Vq=V(:,1:q);    % matriz com q primeiros autovetores
» Z=Vq'*X;    % dados transformados
» Xr=Vq*Z;    % reconstroi dados de entrada
» figure; plot(VEq,'r-','linewidth',2);    % Grafico scree
plot
```

- **Atenção**: O número de componentes principais vai variar em função do valor de `tol`. Quanto maior (menor) o valor de `tol`, maior (menor) será o valor de q.

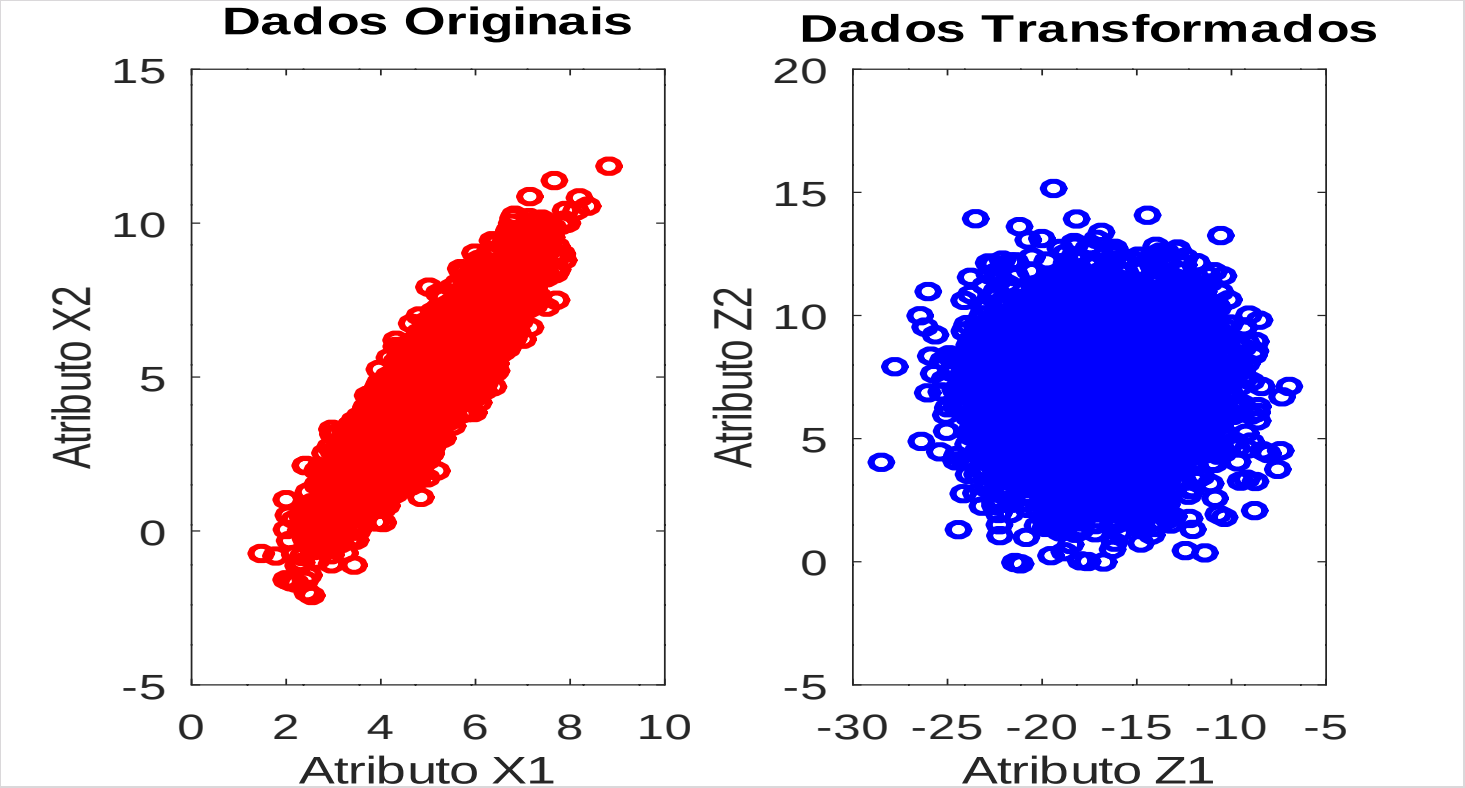
Resultados Simulação

- O gráfico resultante do *scree plot* é mostrado abaixo.



Resultados Simulação

- Gráficos de dispersão dos atributos x_1 e x_2 do conjunto original (esquerda) e z_1 e z_2 do conjunto transformado (direita).



Implementação em Matlab/Octave (cont.-3)

- O método descrito anteriormente não é um método eficiente, computacionalmente falando.
- Ou seja, é um método de *livro-texto*, que tem valor apenas didático, e que não escala bem para dados de alta dimensão.
- Para aplicações práticas, recomenda-se o uso da decomposição em valores singulares (SVD, sigla em Inglês).

  ```
  » [U L V]=svd(Cx);
  ```

  em `L` é uma matriz diagonal com os autovalores ordenados e `V` é a matriz de autovetores correspondentes.
- No Octave/Matlab, este método é usado nas funções `princomp` e `pcacov`.

Aplicação de PCA em Compressão e Recuperação de Imagens

- Considere a seguinte imagem, conhecida como Lena.
- Tamanho: 512×512. Resolução: 8 bits (256 níveis de cinza)

Aplicação de PCA em Compressão e Recuperação de Imagens

- Deseja-se comprimir esta imagem e restaurá-la eventualmente a partir da versão comprimida.
- Inicialmente, pode-se pensar que é preciso ter um conjunto de imagens para se construir um modelo que realize a compressão para qualquer imagem.
- Contudo, pode-se realizar a compressão de apenas uma imagem, que é o caso explorado aqui.
- O truque é associar as dimensões da matriz de dados definida na Eq. (3) com as dimensões da imagem digital (que nada mais é do que uma matriz de intensidades de pixel).
- O caso mais simples consiste em associar o número de linhas da imagem com o número de atributos (p) da matriz $\mathbf{X}$ e o numero de colunas da imagem com a quantidade (N) de vetores de atributos de $\mathbf{X}$. Ou seja, a imagem terá dimensões $p \times N$.

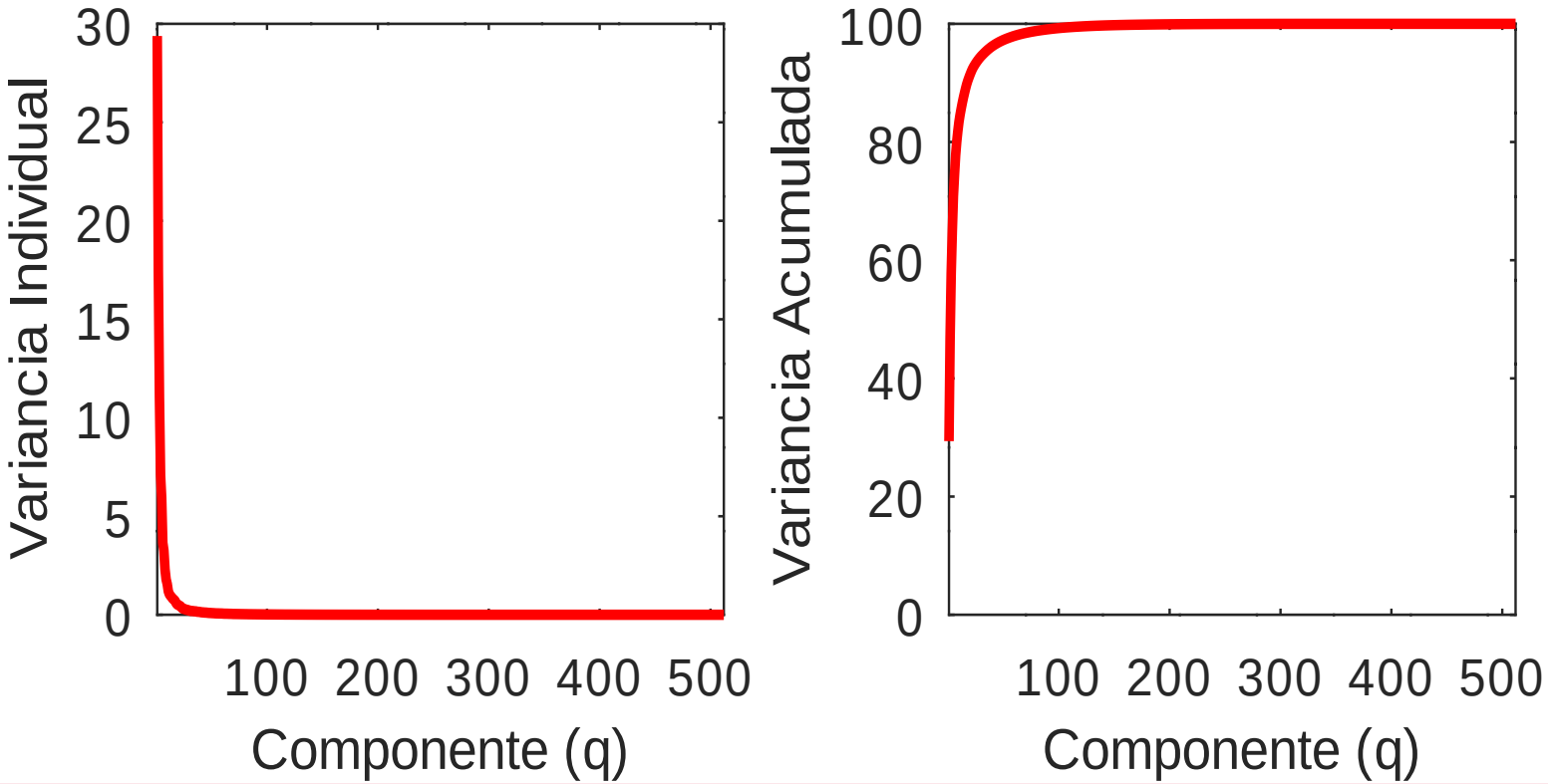
Aplicação de PCA em Compressão e Recuperação de Imagens

- A imagem do exemplo possui dimensões 512×512. Logo, tem-se $p = 512$ e $N = 512$.
- Coincidentemente, a imagem tem as dimensões iguais, resultando em $p = N$. Contudo, isto não é uma restrição, podendo ser aplicada para imagens de dimensões diferentes.
- O objetivo da aplicação de PCA aqui é então reduzir o valor de $p = 512$ para um valor $q \ll p$.
- Em outras palavrs, cada coluna da imagem original tem dimensão 512×1 vai ser transformada para um vetor de dimensão $q \times 1$.
- O código a seguir ilustra como isso pode ser implementado para fins de compressão da informação e também para recuperação da informação.

Implementação em Matlab/Octave

```
» X=imread('lena512.jpg');    % Carrega imagem
» X=im2double(X);    % Converte para ponto flutuante
» Cx=cov(X');    % Estima matriz de covariancia
» [V L VEi]=pcacov(Cx);    % Aplica PCA
» VE=cumsum(VEi);    % Calcula variancia explicada
» figure;
» subplot(2,1,1); plot(VEi,'r-','linewidth',2) % Screeplot
» subplot(2,1,2); plot(VE,'r-','linewidth',2) % Var.  acumulada
```

Gráficos PCA: Screeplot (esq.) + Variância Acumulada (dir.)



Implementação em Matlab/Octave

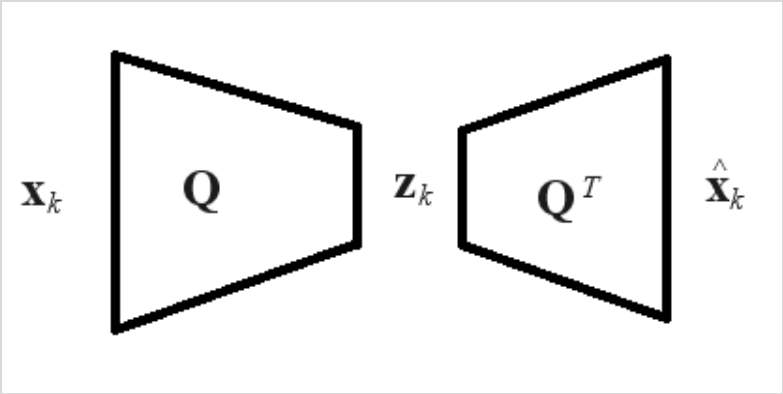
```
» tol=95; q=length(find(VE<=tol));    % Num.  de Componentes Principais
ans =
          32
» Vq=V(:,1:q); Qq = Vq';    % Determina matriz de transformacao
» Zq=Qq*X;    % Reduz dimensionalidade com PCA
» size(Zq)    % Confere nova dimensao dos dados
ans =
          32    512
» Xrec = Qq'*Zq;    % Imagem reconstruida
» size(Xrec)    % Confere nova dimensao dos dados
ans =
          512    512
» figure;
» subplot(1,2,1); imshow(X); % plota imagem original
» subplot(1,2,2); imshow(Xrec); % junto com a imagem reconstruida
```

Imagens: Original (esq.) + Reconstruída ($q = 32$, 90% VE) (dir.)

Imagens: Original (esq.) + Reconstruída ($q = 60$, 98% VE) (dir.)

PCA como Autoencoder Linear

- A técnica PCA pode ser entendida como um *autoencoder* linear.



- A parte do codificador (compressor) é implementada pela operação $\mathbf{z}_k = \mathbf{Q}_q \mathbf{x}_k$.
- A parte do decodificador (reconstrutor) é implementada pela operação $\hat{\mathbf{x}}_k = \mathbf{Q}_q^T \mathbf{z}_k$.
- Dependendo da área de aplicação, o vetor $\mathbf{z}_k$ pode ser chamado de *vetor-código* (telecomunicações) ou de vetor de *variáveis latentes* (aprendizado de máquinas).

PCA como Autoencoder Linear

- A norma quadrática do erro de reconstrução da entrada $\mathbf{x}_k$ do autoencoder é dado por

$$e_k^2 = \|\mathbf{x}_k - \hat{\mathbf{x}}_k\|^2, \quad k = 1, \ldots, N \tag{60}$$

  em que $\|\cdot\|$ denota a norma euclidiana.
- Um autoencoder pode ser usado em problemas de *detecção de anomalias*, que é um tipo de classificação binária.
- Porém, devido à escassez de exemplos de uma das classes, detecção de anomalias é normalmente tratado como um problema de classe única (*one-class problem*).
- Assim, treina-se o autoencoder com os dados da classe com mais exemplos disponíveis, usualmente a classe de exemplos negativos (i.e., contém dados *normais* ou *não-anômalos*).

PCA como Autoencoder Linear

- Suponha que medimos várias grandezas físicas, agrupando-as em um vetor $\mathbf{x} \in \mathbb{R}^p$, e que juntas representam o estado de um processo industrial, de um equipamento, ou até de um paciente.
- Então podemos usar a seguinte regra de decisão para detectar anomalias no processo/equipamento/paciente:

$$\begin{array}{ll} \text{SE} & \|\mathbf{x} - \hat{\mathbf{x}}\|^2 > L, \\ \text{ENTÃO} & \mathbf{x} \text{ é um estado anômalo.} \end{array} \tag{61}$$

  em que L denota o limiar de anomalias.
- Em palavras, se o erro quadrático de reconstrução para um dado vetor $\mathbf{x}$ é maior que o limiar L, então $\mathbf{x}$ provavelmente denota um estado anômalo.
- O limiar de decisão L pode ser calculado de duas maneiras, conforme será descrito a seguir.

Estimação do Limiar de Decisão

- **Método 1** - O limiar de decisão L é determinado pelo valor crítico da distribuição chi-quadrado $\mathcal{X}^2_{(p,1-\alpha)}$ com p graus de liberdade e nível de significância α. A escolha deste método assume que os erros quadráticos $\{e^2(\mathbf{x}, \hat{\mathbf{x}})\}$ seguem tal distribuição.

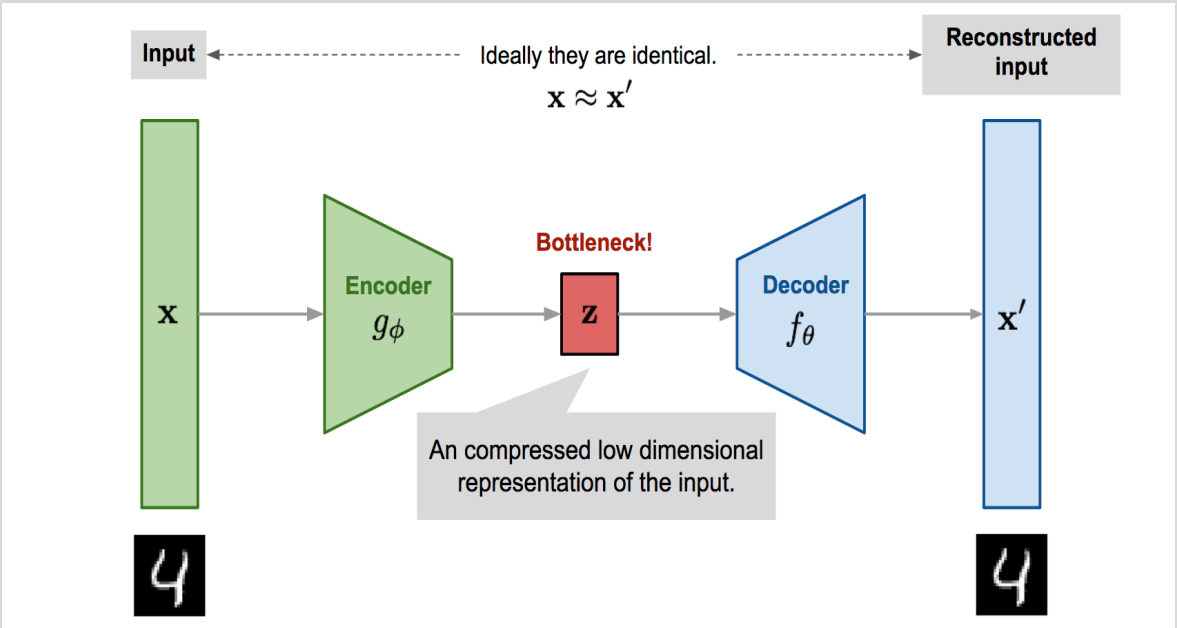
  No Octave/Matlab: `» L=chi2inv(1-`α`,`p`)`.

- **Método 2** - O limiar de decisão L é determinado calculando-se o percentil 100*(1-α) para o conjunto $\{e^2(\mathbf{x}_k, \hat{\mathbf{x}}_k)\}_{k=1}^{N_{trn}}$, que é o conjunto dos erros quadráticos de reconstrução dos N_{trn} vetores de treinamento.

  No Octave/Matlab: `k=prctile(D2trn,100*(1-`α`))`.

Modelo Geral de um Autoencoder

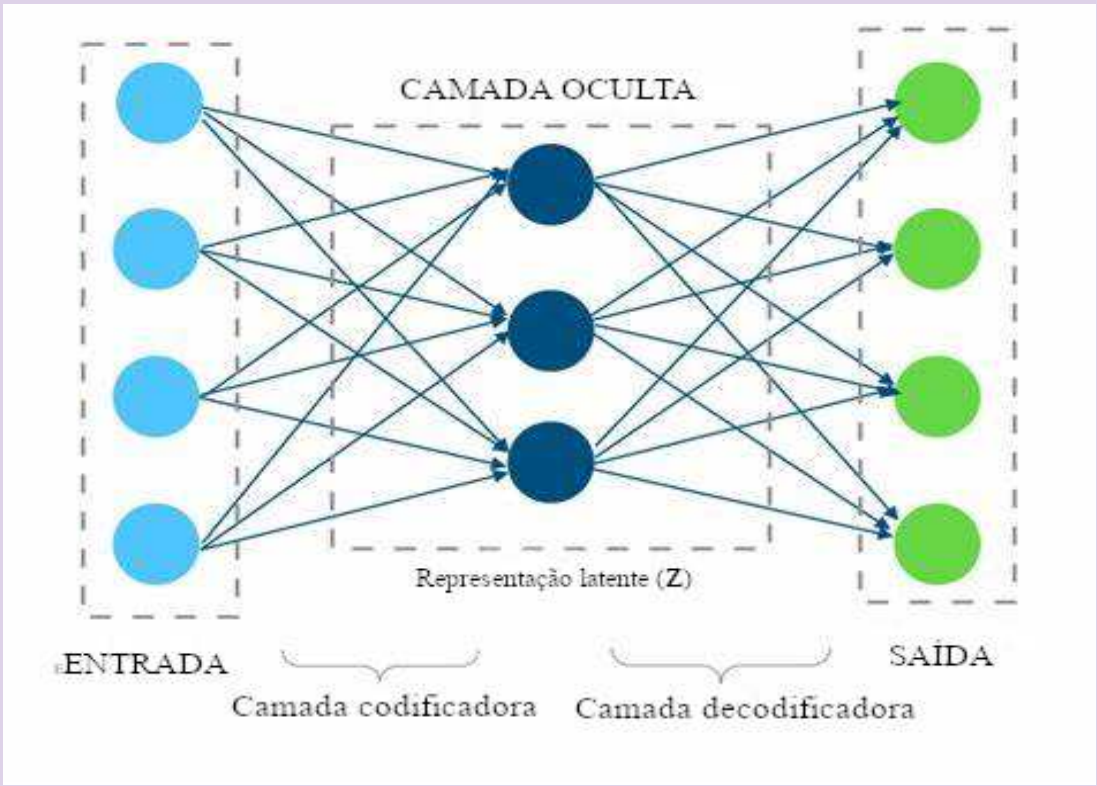
- Vimos que a técnica PCA pode ser entendida como um *Autoencoder* linear. Porém, o conceito de autoencoder é mais amplo e pode ser generalizado pelo modelo mostrado na figura abaixo.



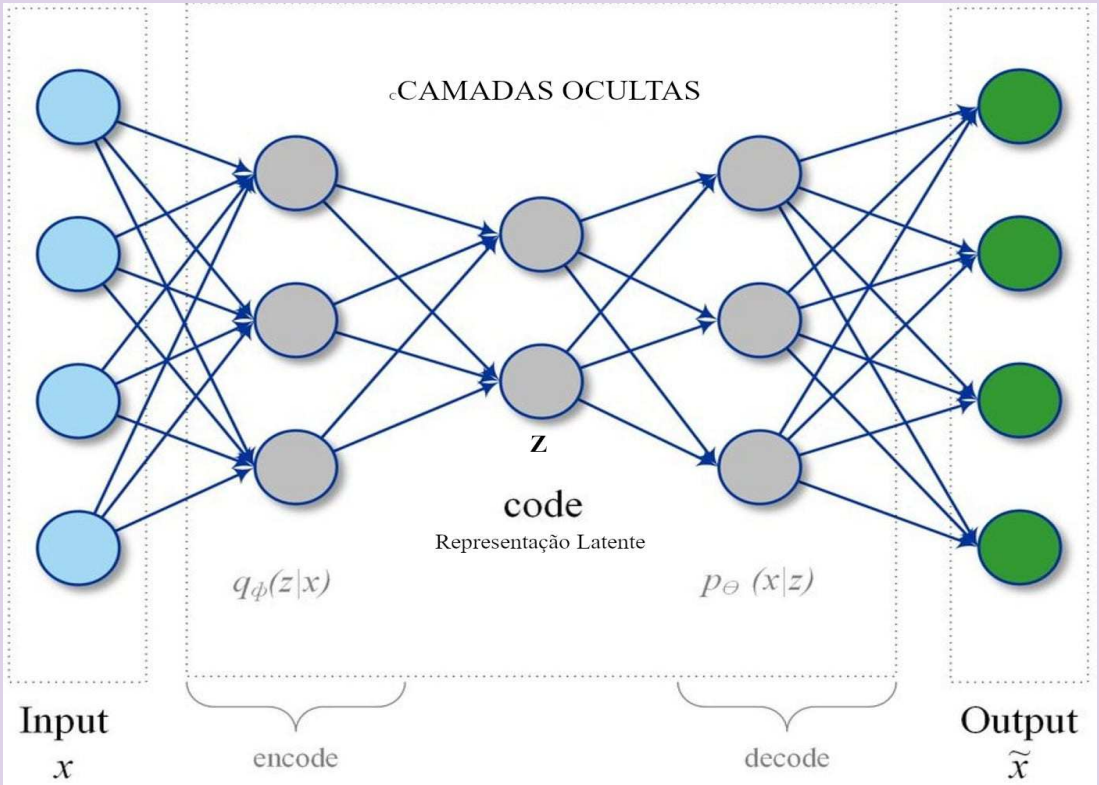
Autoencoder Usando Redes Neurais

- Assim, podemos usar a mesma lógica para projeto de um detetor de anomalias baseado no erro de reconstrução para projetar um *autoencoder* (AE) baseado em redes neurais do tipo MLP (*perceptron multicamadas*).
- Neste caso, as saídas-alvo da rede MLP são as próprias variáveis de entrada. Portanto, o número de neurônios de saída é igual ao à dimensão do vetor de entrada entrada.
- Para uma rede MLP com uma camada oculta, a dimensão do espaço latente (q) é dada pelo número de neurônios ocultos (ver próximo slide).
- O vetor-código $\mathbf{z}_k$, ou seja, a versão codificada do vetor de entrada $\mathbf{x}_k$ é dado pelo vetor formado pelas saídas dos neurônios ocultos.

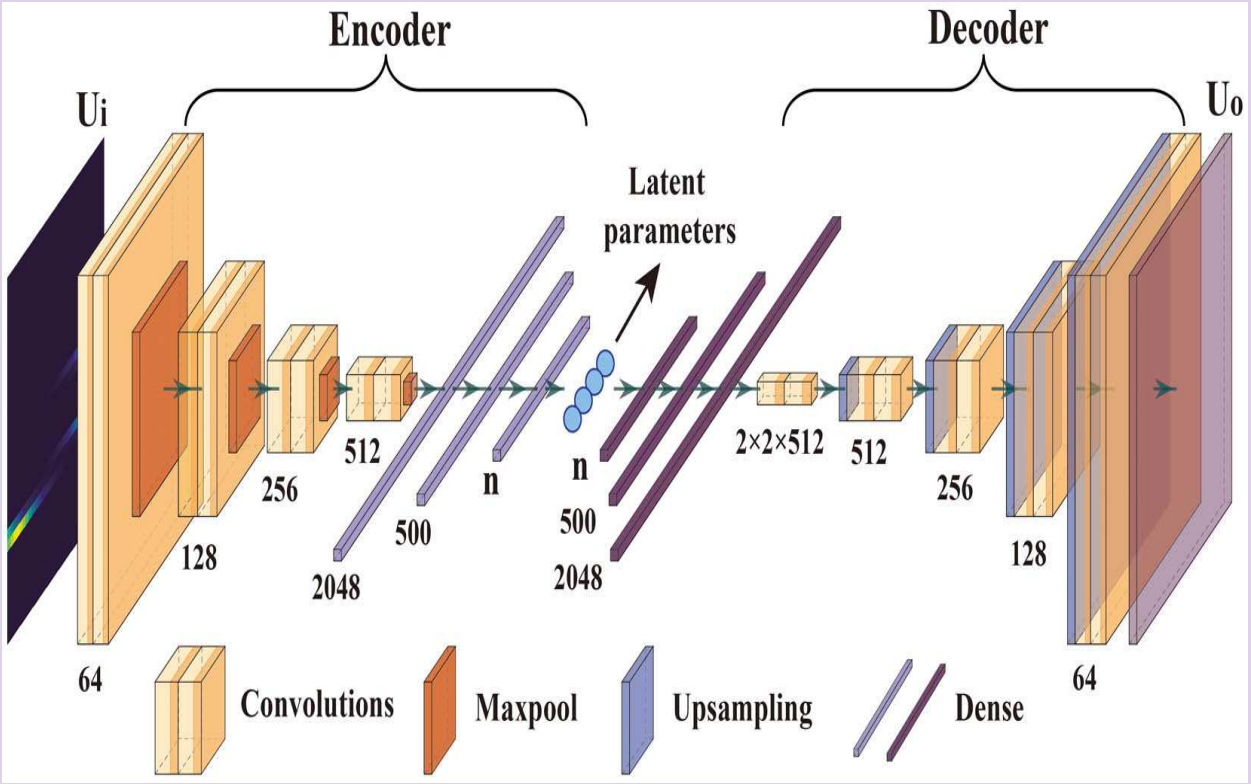
Arquitetura de um AE-MLP de 1 camada oculta. A representação latente do vetor de entrada $\mathbf{x} \in \mathbb{R}^p$ é dada pelo vetor $\mathbf{z} \in \mathbb{R}^q$, $q < p$, contendo as saídas dos neurônios da única camada oculta.



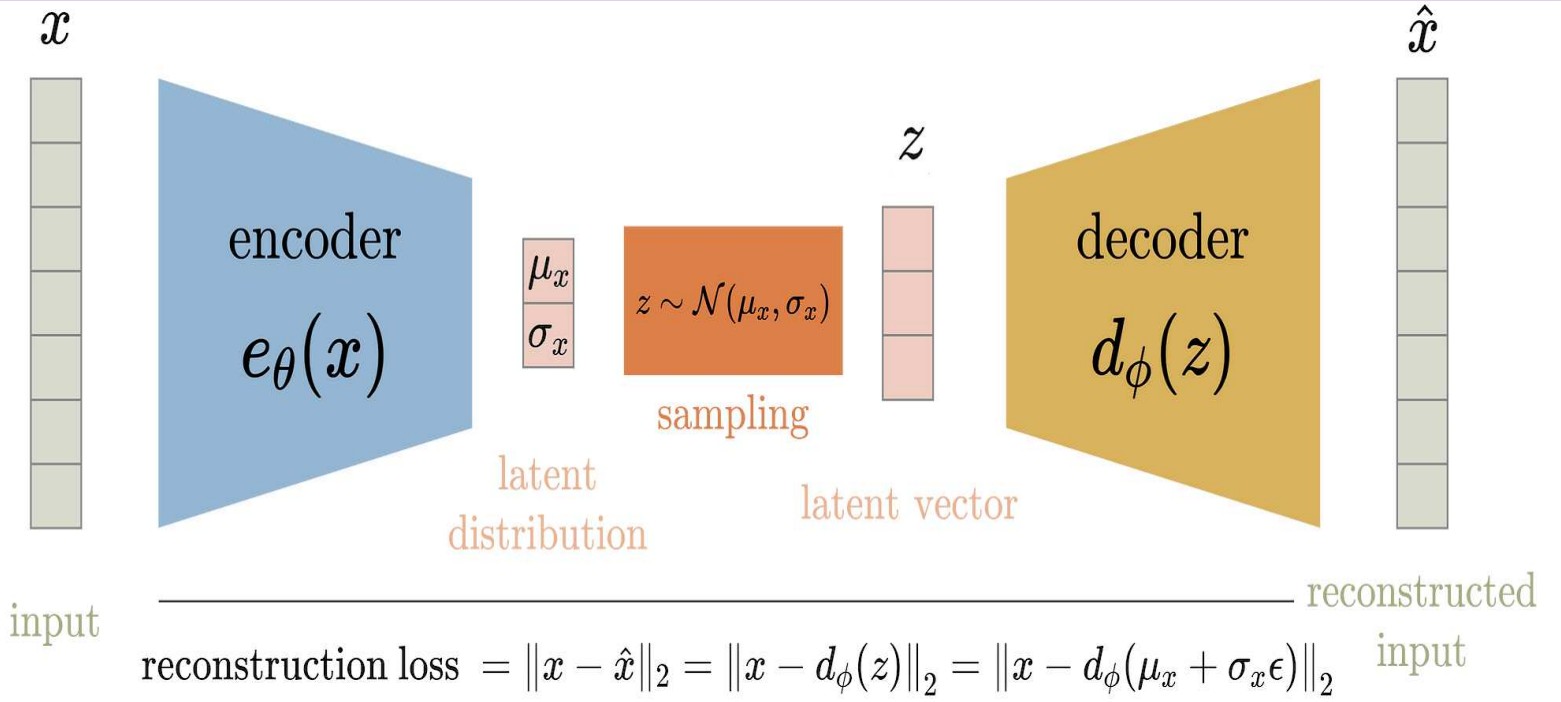
Arquitetura de um AE-MLP de 3 camadas ocultas. A representação latente do vetor de entrada $\mathbf{x} \in \mathbb{R}^p$ é dada pelo vetor $\mathbf{z} \in \mathbb{R}^q$, $q < p$, contendo as saídas dos neurônios da camada oculta central; ou seja, da 2a. camada oculta.



Arquitetura de um AE convolucional.



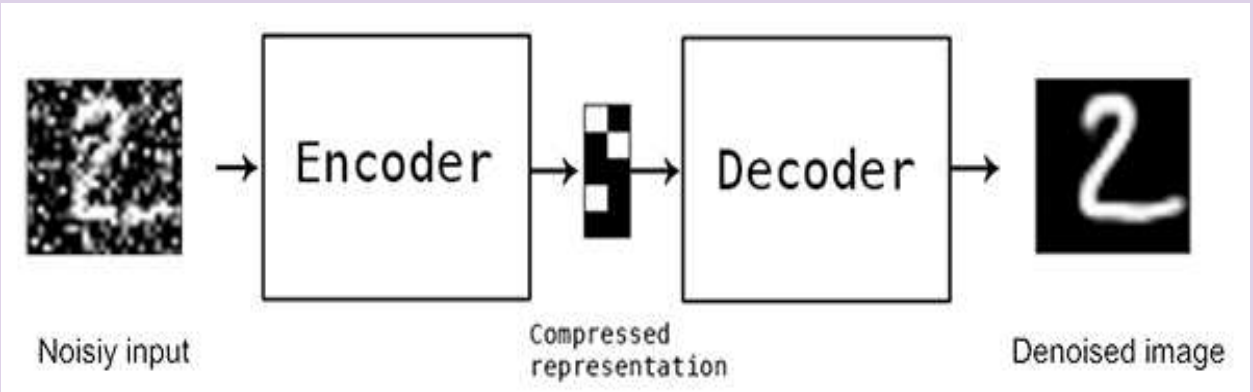
Arquitetura de um AE variacional.



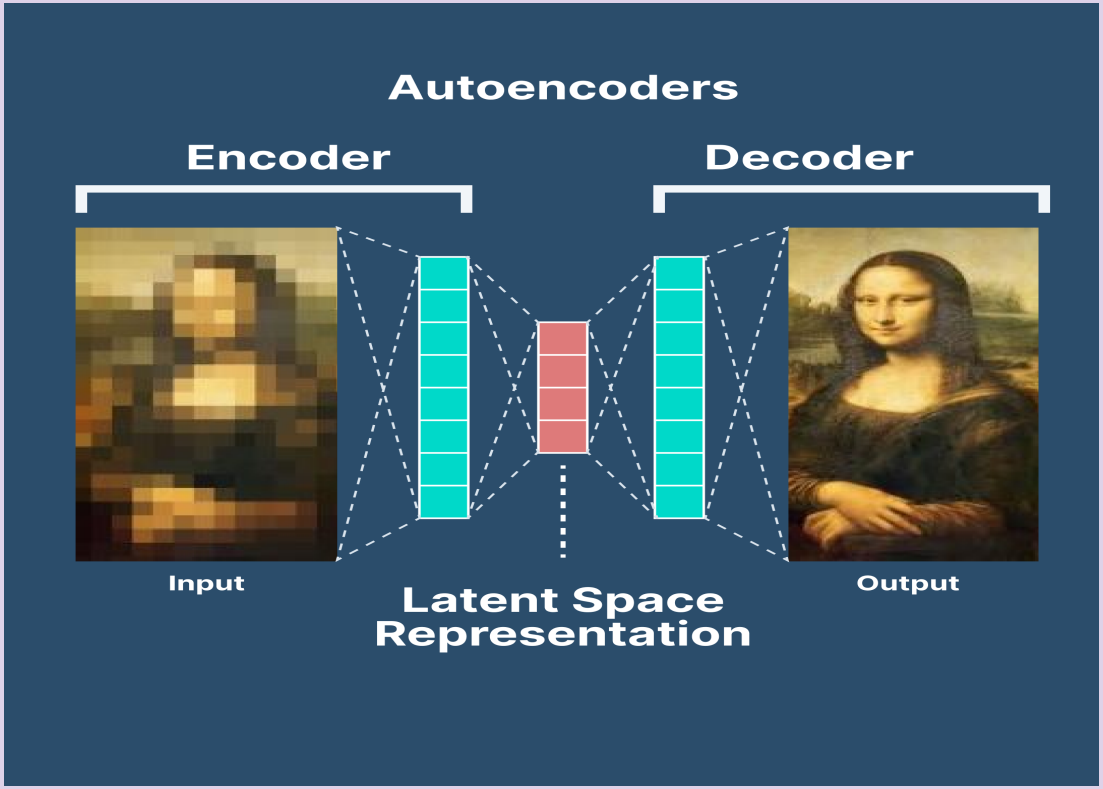
Aplicações de Autoencoders

- O tópico *autoencoders* foi abordado inicialmente através da aplicação em detecção de anomalias.
- Contudo, há diversas outras aplicações possíveis para AEs, mas todas resultantes das propriedades de comprssão da informação de autoencoders, tais como
  - *Compressão de dados*: Usado para comprimir dados de natureza complexa, porém com perdas (*lossy compression*).
  - *Image Denoising*: Usado para filtragem de ruído em imagens.
  - *Redução de Dimensionalidade*: Usado para classificação ou visualização de dados.
  - *Recuperação de Imagens*: Usado para recuperar trechos de imagens ou imagens inteiras.
  - *Produção de Imagens*: Usado como modelo generativo (AE variacional) para produção de novas imagens.

Autoencoder para Filtragem de Ruído. No treinamento adiciona-se um pouco de ruído ao vetor/imagem de entrada: $\tilde{\mathbf{x}}_k = \mathbf{x}_k + \mathbf{n}_k$, $\mathbf{n}_k \sim N(0, \sigma_n^2)$. Porém, deve-se usar como saídas-alvo o vetor/imagem sem ruído $\mathbf{x}_k$.



Autoencoder para Recuperação de imagens: treinamento com imagem na qualidade desejada, teste com imagens de qualidade inferior.



PCA e AEs em Classificação

- A aplicação de PCA e AEs à matriz de dados originais $\mathbf{X}$ $(p \times N)$ gerará uma nova matriz $\mathbf{Z}$ $(q \times N)$.
- As colunas de $\mathbf{Z}$ são formadas por $\mathbf{z}_k = \mathbf{Q}_q \mathbf{x}_k$, para $k = 1, \ldots, N$.
- Assim, em um problemas de classificação, projeta-se o classificador com os pares entrada-saída $\{(\mathbf{z}_k, \mathbf{y}_k)\}_{k=1}^{N}$ extraídos das seguintes matrizes:

$$\mathbf{Z} = [\mathbf{z}_1 \,|\, \mathbf{z}_2 \,|\, \cdots \,|\, \mathbf{z}_k \,|\, \cdots \,|\, \mathbf{z}_N] \tag{62}$$

$$\mathbf{Y} = [\mathbf{y}_1 \,|\, \mathbf{y}_2 \,|\, \cdots \,|\, \mathbf{y}_k \,|\, \cdots \,|\, \mathbf{y}_N], \tag{63}$$

em que $\mathbf{Y}$ é a matriz de rótulos inicialmente associada à matriz $\mathbf{X}$.