UNIVERSIDADE DO ALGARVE

FACULDADE DE CIÊNCIAS E TECNOLOGIA

A DECOMPOSIÇÃO EM VALORES SINGULARES E SUAS APLICAÇÕES

Dissertação para a obtenção do grau de mestre em Matemática,

especialização em Matemática para o Ensino

SANDRA GUERREIRO GONÇALVES NOBRE

FARO

2007

NOME: Sandra Guerreiro Gonçalves Nobre

DEPARTAMENTO: Matemática

ORIENTADOR: Professor Doutor Rafael Brigham Neves Ferreira Santos

DATA: 9 de Julho de 2007

TÍTULO DA DISSERTAÇÃO: A Decomposição em Valores Singulares e suas Aplicações

JÚRI: Professor Doutor Rafael Brigham Neves Ferreira Santos

Professora Doutora Maria Joana Feijão Ehrhardt Soares

Professora Doutora Maria da Graça Nunes da Silva Rendeiro Marques

Agradecimentos

Em primeiro lugar quero agradecer ao meu orientador, professor Rafael Santos, pela sua disponibilidade, apoio e incentivo dado. Pela sua paciência e inestimável ajuda durante todo o desenvolvimento desta tese.

Ao professor Luís Faísca, pela divulgação e esclarecimento acerca da aplicação da decomposição em valores singulares na recuperação de informação.

Ao professor João Rodrigues, pelos documentos, sobre processamento de imagem e visão por computador, disponibilizados e por algumas dicas úteis para o progresso deste trabalho.

À minha família que sempre me apoiou e incentivou, para que eu pudesse concretizar este estudo.

Aos meus amigos e a todas as pessoas que, directa ou indirectamente, contribuíram para a realização desta tese de mestrado.

Resumo

Nesta dissertação analisamos a Decomposição em Valores Singulares e algumas das suas aplicações.

Começamos por evidenciar o trabalho efectuado pelos matemáticos que mais contribuíram para evolução deste tipo de decomposição até à actualidade. De seguida, é feita a exposição, demonstração de existência e interpretação geométrica da decomposição em valores singulares, assim como também são evidenciadas algumas das propriedades que fazem desta decomposição uma das mais importantes e com um maior número de utilizações em diversos campos.

Posteriormente são expostas algumas das imensas aplicações da decomposição em valores singulares, nomeadamente na compressão de imagens, na reconstrução e reconhecimento facial, na recuperação da estrutura tridimensional, na recuperação de informação, na criptografia e na reconstrução da matriz de expressão dos genes.

Palavras-chave: Decomposição em Valores Singulares, reconhecimento facial, recuperação de informação, criptogramas, expressão dos genes, compressão de imagens.

Singular Value Decomposition and its applications

Abstract

In this thesis we study the main properties of the singular value decomposition and some of its applications.

We begin with a short account of the history of the singular value decomposition. After giving the existence theorem and the geometric interpretation, we present its main properties.

The applications we show cover image compression, facial reconstruction and recognition, 3D structure reconstruction, information retrieval, cryptography and missing data retrieval in gene expression.

Key-words: Singular Value Decomposition, facial recognition, information retrieval, cryptograms, gene expression, image compression.

Índice

Capítulo 1

Introdução

O curso de mestrado, no qual esta dissertação se insere, tem como principal objectivo o desenvolvimento profissional dos docentes de matemática dos ensinos básico e secundário, através do aprofundamento da sua formação científica.

Assim, pretendemos com esta tese, actualizar, aperfeiçoar e complementar os conhecimentos já adquiridos, através da análise de um tipo de decomposição matricial não estudado na licenciatura. A decomposição matricial que vamos estudar é a Decomposição em Valores Singulares.

A Decomposição em Valores Singulares, também designada por SVD[1], tem mais de cem anos de existência. Este tipo de factorização para matrizes quadradas foi descoberta por Beltrami em 1873 e Jordan em 1874. A técnica foi posteriormente estendida para matrizes rectangulares por Eckart e Young na década de 30 do séc. XX e o seu uso ao nível computacional deu-se a partir de 1960. Gene H. Golub e Charles F. Van Loan também se têm debruçado sobre este tipo de decomposição e demonstrado a sua aplicabilidade com bastante sucesso em diversas áreas.

O estudo da Decomposição em Valores Singulares é importante pois esta possui propriedades muito interessantes no que diz respeito ao cálculo de normas, à determinação da característica de uma matriz, assim como na resolução de sistemas lineares no sentido dos mínimos quadrados. Devido às suas potencialidades, este tipo de

[1] Do inglês, Singular Value Decomposition.

decomposição é muito utilizado na álgebra linear e tem diversas aplicações, como já foi referido atrás.

A elaboração desta dissertação teve como base o artigo "Singular Value Decomposition, Eigenfaces, and 3D Reconstructions", [1]. Este foca as principais propriedades da Decomposição em Valores Singulares e algumas aplicações como a reconstrução e o reconhecimento facial e a recuperação da estrutura tridimensional a partir de imagens num filme.

Nesta dissertação será abordada a contribuição da Decomposição em Valores Singulares na compressão de imagens, na reconstrução e reconhecimento facial usando faces próprias, na recuperação da estrutura tridimensional a partir de imagens de um filme, na recuperação de informação, na criptografia e na reconstrução da matriz de expressão dos genes.

O carácter de interdisciplinaridade destes conhecimentos é importante quer para a formação profissional, assim como para futura divulgação aos alunos, no sentido de motivá-los para o estudo e interesse pela Matemática evidenciando aplicações práticas e actuais.

De um modo geral, o auxílio da Decomposição em Valores Singulares em todas as aplicações atrás mencionadas, reside no facto de através desta decomposição matricial se poder reduzir significativamente a dimensão do espaço inicial, podendo reter apenas os dados que são mais importantes, desprezando os outros.

Neste trabalho começamos por, no capítulo 2, evidenciar o trabalho efectuado pelos matemáticos que mais contribuíram para evolução deste tipo de decomposição até à actualidade.

Seguidamente é apresentada a decomposição e a respectiva demonstração de existência, sendo depois feita a sua interpretação geométrica e estudadas as suas propriedades, no que diz respeito ao cálculo de normas e característica de uma matriz, condicionamento e resolução de sistemas lineares no sentido dos mínimos quadrados.

Nos capítulos seguintes são estudadas algumas aplicações deste tipo de decomposição matricial, sendo referidos os conteúdos considerados pertinentes para expor a importância da Decomposição em Valores Singulares na resolução desses problemas.

A primeira aplicação, apresentada no capítulo 3, é a compressão de imagens. A Decomposição em Valores Singulares é utilizada como suporte para aumentar a rapidez

de descarga de imagens e poupança de lugares na memória de um computador. Para isso utilizamos matrizes aproximadas, com característica inferior.

No capítulo 4 estuda-se a reconstrução e o reconhecimento facial. Nesta situação, a utilização da decomposição dá um grande auxílio na construção de uma base ortogonal para o subespaço que contém todas as imagens faciais.

A recuperação da estrutura tridimensional a partir de imagens de filmes, é apresentada no capítulo 5. A ajuda da Decomposição em Valores Singulares é notável para o cálculo da característica efectiva da matriz de observação.

No capítulo 6, estudamos a Recuperação de Informação. Neste caso recorre-se à decomposição para organizar os documentos pela sua relevância de acordo com uma pesquisa efectuada.

A descodificação de mensagens cifradas, um tema da criptografia, é estudada no capítulo 7. A decomposição é utilizada no cálculo de matrizes aproximadas com características um e dois para aproximar a frequência de cada letra no criptograma e para fazer a separação das letras, respectivamente.

Por fim no capítulo 8, estudamos a determinação da expressão dos genes. Utilizamos a decomposição para calcular entradas em falta na matriz de expressão dos genes. Escrevemos os vectores que representam os genes com dados em falta como combinação linear dos genes próprios significativos.

Neste trabalho, para além das referências citadas no texto, inclui-se também uma bibliografia adicional, com outros materiais por nós consultados.

Para complementar a exposição das aplicações, a maioria dos capítulos vem acompanhada de exemplos concretos da aplicação. Para isso foram elaborados programas em MATLAB 6.5.

Capítulo 2

A Decomposição em Valores Singulares

2.1 – Introdução

Desde o século XVIII, a teoria das matrizes tem sido tema de estudo por parte de diversos matemáticos, [2]. A introdução do conceito de matriz teve origem no desenvolvimento dos determinantes, para determinar os coeficientes de sistemas de equações lineares.

Uma das descobertas mais proveitosas foi a decomposição matricial, para além da SVD, a decomposição em valores próprios, a decomposição LU e a decomposição QR, são outras decomposições notáveis. A apresentação de matrizes nesta forma tem sido muito utilizada para resolver diversos problemas.

Eugénio Beltrami (1835-1899), em 1873 deu um contributo notável para o aparecimento da decomposição em valores singulares. A sua contribuição mais importante e a primeira nesta área surgiu no jornal " Journal of Mathematics for the use of the students of the Italian Universities" e o seu principal objectivo era incentivar os alunos para o estudo das formas bilineares. É de notar, a falta de generalização e as poucas propriedades enunciadas, para esta apresentação da decomposição em valores singulares. Beltrami, neste trabalho considerou-a apenas para matrizes quadradas não singulares e tendo valores singulares distintos.

Um ano mais tarde, em 1874, Jordan (1838-1922), independentemente de E. Beltrami, publicou o seu trabalho: 'Mémoire sur les formes bilinéaires' onde aparecem os valores singulares para formas bilineares. Jordan reduziu uma forma bilinear para uma forma diagonal por substituições ortogonais.

James Joseph Sylvester (1814-1897) também demonstrou a decomposição em valores singulares, em 1889, para matrizes reais quadradas, tendo chamado multiplicadores canónicos aos valores singulares.

Em 1907, Erhard. Schmidt (1876-1959), definiu os valores singulares no domínio dos operadores integrais.

H. Weyl (1885-1955) em 1912, deu também um contributo importante para o estudo dos valores singulares de uma matriz $\tilde{A}$, aproximada de A.

O desenvolvimento para qualquer tipo de matrizes foi realizado por Eckart e Young em 1939.

Em 1965, Golub e Kahan introduziram a decomposição em valores singulares na análise numérica e em 1970 Golub propôs um algoritmo para a sua computação que ainda continua a ser muito utilizado actualmente. Surgiu também, em meados da década de 70, [3] e nos princípios dos anos oitenta [4] a Decomposição em Valores Singulares Generalizada que tem bastantes aplicações, nomeadamente na comparação de sequências de ADN de dois organismos diferentes [5].

2.2 – Existência da Decomposição em Valores Singulares

Seja $\mathbb{R}^{m\times n}$ o espaço das matrizes reais de ordem $m\times n$.

Definição 2.1: Se $A \in \mathbb{R}^{m\times n}$. Definimos a norma euclideana de A (também chamada norma-dois) através de $\|A\|_2 = \max_{\|x\|_2=1} \|Ax\|_2$.

A decomposição em valores singulares de uma matriz $A\in \mathbb{R}^{m\times n}$, é dada pelo seguinte teorema:

Teorema 2.1: Seja $A \in \mathbb{R}^{m\times n}$ uma matriz. Então existem matrizes $U \in \mathbb{R}^{m\times m}$ e $V \in \mathbb{R}^{n\times n}$, ortogonais, tais que $A = U\Sigma V^T$, onde $\Sigma = \operatorname{diag}(\sigma_1, \sigma_2, \cdots, \sigma_p) \in \mathbb{R}^{m\times n}$, $p = \min\{m,n\}$ e $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_p \geq 0$.

Demonstração: (ver, por exemplo, [6])

Seja $\|A\|_2 = \sigma_1$ e sejam $u_1^{(1)} \in \mathbb{R}^m$, $v_1^{(1)} \in \mathbb{R}^n$, tais que $\|u_1^{(1)}\|_2 = \|v_1^{(1)}\|_2 = 1$ e $\sigma_1 u_1^{(1)} = A v_1^{(1)}$.

Escolhamos agora vectores $u_2^{(1)},\ldots,u_m^{(1)} \in \mathbb{R}^m$ e $v_2^{(1)},\ldots,v_n^{(1)} \in \mathbb{R}^n$ tais que $U^{(1)} = [\, u_1^{(1)} \quad u_2^{(1)} \ldots u_m^{(1)} \,]$ e $V^{(1)} = [\, v_1^{(1)} \quad v_2^{(1)} \ldots v_n^{(1)} \,]$ sejam ortogonais.

Então temos:

$$U^{(1)T} A V^{(1)} = U^{(1)T} \left[A v_1^{(1)} \quad A v_2^{(1)} \ldots A v_n^{(1)} \right]$$

$$= \begin{bmatrix} u_1^{(1)T} \\ u_2^{(1)T} \\ \vdots \\ u_m^{(1)T} \end{bmatrix} \left[\sigma_1 u_1^{(1)} \quad A v_2^{(1)} \ldots A v_n^{(1)} \right] = \begin{bmatrix} \sigma_1 & w_1^T \\ 0 & B_1 \end{bmatrix} \equiv A_1$$

Como $U^{(1)}$ e $V^{(1)}$ são ortogonais, $\sigma_1 = \|A\|_2 = \|A_1\|_2$ e portanto

$$\sigma_1 = \|A_1\|_2 = \max_{\|x\|_2=1} \|A_1 x\|_2 \geq \left\| A_1 \begin{bmatrix} \sigma_1 \\ w_1 \end{bmatrix} \frac{1}{\sqrt{\sigma_1^2 + w_1^T w_1}} \right\|_2 \geq \sqrt{\sigma_1^2 + w_1^T w_1} \geq \sigma_1 \ .$$

A desigualdade anterior só pode ser satisfeita se $w_1 = 0$. A conclusão é que

$$U^{(1)T} A V^{(1)} = \begin{bmatrix} \sigma_1 & 0 \\ 0 & B_1 \end{bmatrix}.$$

Repetindo agora o procedimento para $B_1 \in \mathbb{R}^{(m-1) \times (n-1)}$, sejam $U_1^{(2)} \in \mathbb{R}^{(m-1) \times (m-1)}$ e $V_1^{(2)} \in \mathbb{R}^{(n-1)\times(n-1)}$ ortogonais, tais que

$$U_1^{(2)T} B_1 V_1^{(2)} = \begin{bmatrix} \sigma_2 & 0 \\ 0 & B_2 \end{bmatrix}.$$

Definindo então

$U^{(2)} = \begin{bmatrix} 1 & 0 \\ 0 & U_1^{(2)} \end{bmatrix}$ e $V^{(2)} = \begin{bmatrix} 1 & 0 \\ 0 & V_1^{(2)} \end{bmatrix}$, as matrizes $U^{(i)}$, $\mathrm{V}^{(i)}$ e A satisfazem

$$U^{(2)^T} U^{(1)^T} A V^{(1)} V^{(2)} = \begin{bmatrix} \sigma_1 & 0 & 0 \\ 0 & \sigma_2 & 0 \\ 0 & 0 & B_2 \end{bmatrix}.$$

Continuando da mesma forma para B_2 e assim sucessivamente, chegamos à conclusão que $U^T A V = \Sigma$, o que demonstra o teorema.

Os valores σ_i $(i = 1, \cdots, p)$ são os ***valores singulares*** da matriz A. As colunas das matrizes U e V são os ***vectores singulares esquerdos*** e os ***vectores singulares direitos*** de A, respectivamente.

A SVD pode ser estendida ao conjunto das matrizes complexas, mas tal não será abordado nesta dissertação.

Se $m \neq n$, a matriz Σ apresenta uma das seguintes formas.

Para $m > n$:

$$\Sigma = \begin{bmatrix} \sigma_1 & 0 & \cdots & 0 \\ 0 & \sigma_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \sigma_n \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{bmatrix}.$$

Para $m < n$:

$$\Sigma = \begin{bmatrix} \sigma_1 & 0 & \cdots & 0 & 0 & \cdots & 0 \\ 0 & \sigma_2 & \cdots & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & \sigma_m & 0 & \cdots & 0 \end{bmatrix}.$$

2.3 – Determinação da Decomposição em Valores Singulares

Como é fácil de verificar,

$$AA^T U = U\Sigma V^T V \Sigma^T U^T U = U\Sigma\Sigma^T \tag{2.1}$$

e

$$A^T A\ V = V\Sigma^T U^T U\Sigma V^T V = V\Sigma^T\Sigma\,, \tag{2.2}$$

em que

$$\Sigma\Sigma^T = \text{diag}\,(\sigma_1^2, \sigma_2^2, \ldots, \sigma_p^2, 0, \ldots, 0) \in \mathbb{R}^{m\times m} \tag{2.3}$$

e

$$\Sigma^T\Sigma = \text{diag}\,(\sigma_1^2, \sigma_2^2, \ldots, \sigma_p^2, 0, \ldots, 0) \in \mathbb{R}^{n\times n}, \tag{2.4}$$

sendo $p = \min\{m,n\}$.

Um aspecto importante a ter em conta é que dada uma matriz $A_{m\times n}$, $A^T A$ é uma matriz simétrica e portanto é diagonalizável, sendo os seus valores próprios reais e não negativos e os vectores próprios ortogonais.

As colunas de U são os vectores próprios de AA^T e as colunas de V são os vectores próprios de $A^T A$. Além disso, verifica-se que os quadrados dos valores singulares de A são os valores próprios de AA^T e de $A^T A$.

Como se viu atrás, $A^T A = V\Sigma^T\Sigma V^T$, o que implica que $\Sigma^T\Sigma$ contém os valores próprios de $A^T A$ e V contém os vectores próprios correspondentes.

Desta forma, obtemos a decomposição em valores singulares de uma matriz, resolvendo um problema de valores próprios.

Suponhamos que, de (2.2), já conhecemos os valores singulares e a matriz V. Falta determinar a matriz U.

Sabemos que $A = U\Sigma V^T$. Então

$$AV = U\Sigma\,, \tag{2.5}$$

ou seja,

$$A\begin{bmatrix} v_1 & v_2 \dots v_n \end{bmatrix} = \begin{bmatrix} u_1 & u_2 \dots u_m \end{bmatrix} \begin{bmatrix} \sigma_1 & 0 & \cdots & 0 \\ 0 & \sigma_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \sigma_n \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{bmatrix}, \tag{2.6}$$

(no caso de $m > n$).

Se efectuarmos a multiplicação das matrizes obtemos

$$\begin{bmatrix} Av_1 & Av_2 & \cdots & Av_n \end{bmatrix} = \begin{bmatrix} \sigma_1 u_1 & \sigma_2 u_2 & \dots & \sigma_n u_n \end{bmatrix}. \tag{2.7}$$

Portanto,

$$Av_j = \sigma_j u_j \Leftrightarrow u_j = \frac{Av_j}{\sigma_j}, j=1, \dots, k, \tag{2.8}$$

onde *k* é a característica de *A*.

No caso geral, se $k < m$, é necessário acrescentar $m - k$ vectores de forma a obtermos uma base ortogonal em $\mathbb{R}^m$.

2.4 – Interpretação geométrica da Decomposição em Valores Singulares

Vejamos primeiramente o caso bidimensional, $m = n = 2$.

Consideremos *S*, uma circunferência, à qual é aplicada uma transformação do tipo $A = U\Sigma V^T$.

Seja $z = \begin{bmatrix} x \\ y \end{bmatrix}$, um ponto na circunferência *S*.

$$Az = U\Sigma V^T z \ . \tag{2.9}$$

Seja

$$w = V^T z \ . \tag{2.10}$$

Como V é uma matriz ortogonal, V^T também é ortogonal, isto é, uma matriz de rotação. Temos assim que, sem perda de generalidade,

$$w = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}. \tag{2.11}$$

Esta transformação é uma rotação (no sentido contrário ao dos ponteiros do relógio) com uma amplitude θ. Assim, quando aplicamos V^T à circunferência S ela roda de um ângulo θ, no sentido contrário ao dos ponteiros do relógio.

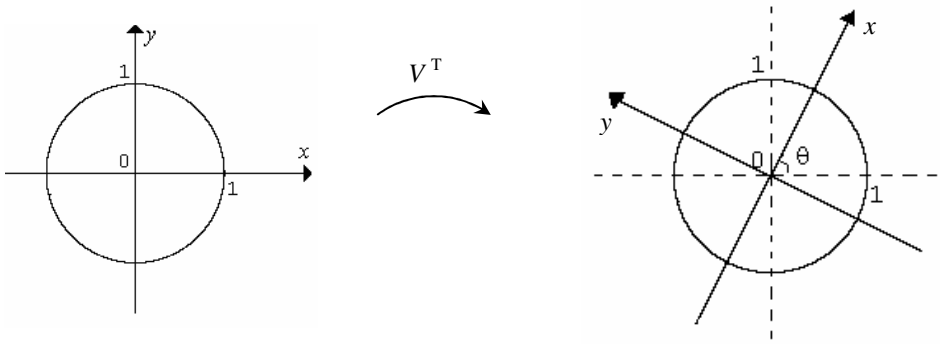


Fig. 2.1 Rotação no sentido contrário aos ponteiros do relógio

Continuamos agora, transformando S, depois de rodada, através de Σ. Como $v \equiv \Sigma w = \begin{bmatrix} \sigma_1 w_1 \\ \sigma_2 w_2 \end{bmatrix}$, vemos que a circunferência é transformada numa elipse cuja excentricidade é $\sqrt{1 - \frac{\sigma_2^{\,2}}{\sigma_1^{\,2}}}$.

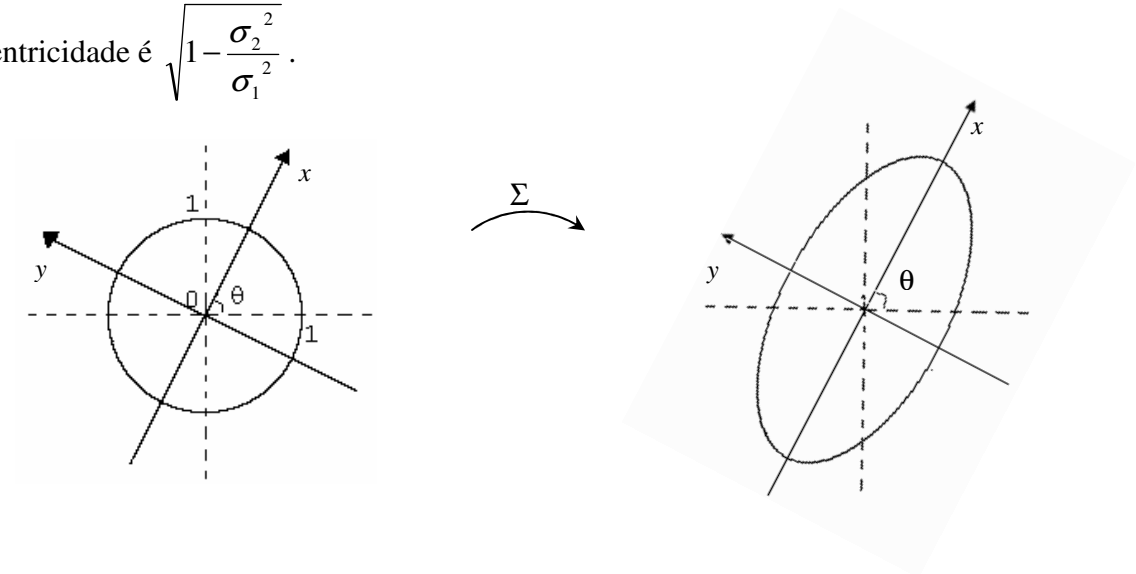


Fig. 2.2 Transformação da circunferência numa elipse

Finalmente a aplicação de *U* à elipse fá-la rodar, em sentido contrário, de um ângulo θ.

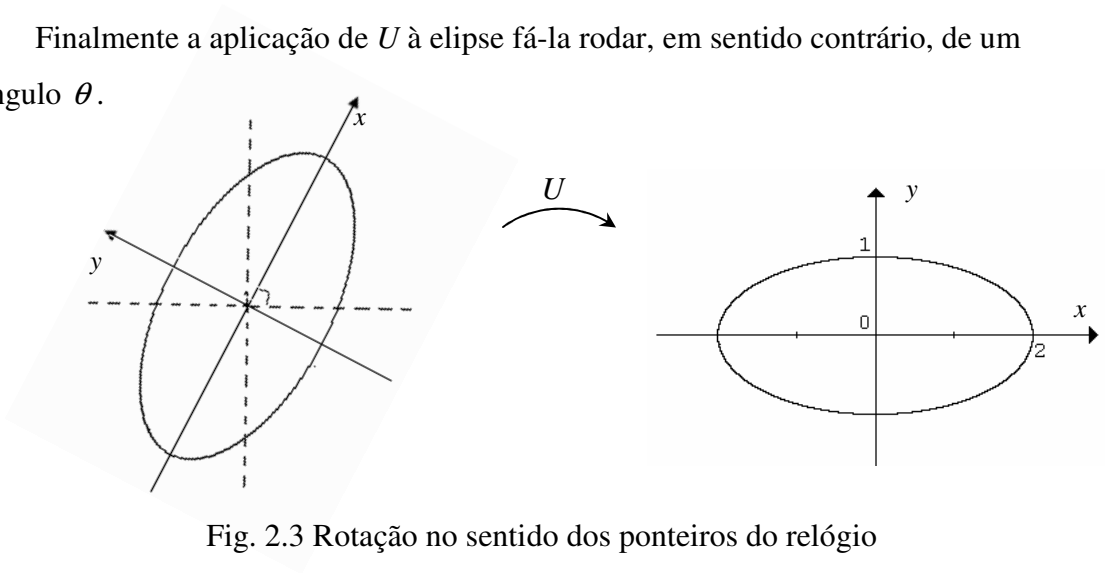


Fig. 2.3 Rotação no sentido dos ponteiros do relógio

A transformação é ilustrada nas figuras 2.1, 2.2 e 2.3 para uma circunferência unitária e $\sigma_1 = 2$ e $\sigma_2 = 1$.

Esta interpretação geométrica é naturalmente estendida ao caso em que *S* é uma esfera unitária em $\mathbb{R}^n$. Neste caso, definimos uma hiperelipse (generalização de uma elipse) em $\mathbb{R}^m$, como a superfície obtida por deformação da esfera unitária ao longo das direcções ortogonais $u_1,\ u_2,\ldots,u_m$ de $\mathbb{R}^m$, de acordo com os valores $\sigma_1,\sigma_2,\cdots,\sigma_m$. Eventualmente, algumas destas quantidades podem ser nulas e assim pode obter-se uma hiperelipse de dimensão *n* num espaço de dimensão *m*, onde *m* > n. Analogamente, os vectores $\sigma_i u_i$ são os semi-eixos principais da hiperelipse e os valores singulares $\sigma_1,\sigma_2,\cdots,\sigma_m$ correspondem aos respectivos comprimentos.

2.5 – As normas euclideana e de Frobenius

Através dos valores singulares de uma matriz podemos facilmente calcular a sua norma euclideana. Com efeito, facilmente se verifica que

$$\|A\|_2 = \sqrt{\max_{\lambda\in\sigma(A^TA)} |\lambda|}\,, \tag{2.12}$$

onde $\sigma(A^TA)$ é o conjunto dos valores próprios da matriz A^TA.

Como vimos, $\sigma_1^2, \ldots, \sigma_n^2$ são os valores próprios de $A^T A$ e estão ordenados por ordem não crescente; logo de (2.12),

$$\|A\|_2 = \sigma_1 . \tag{2.13}$$

Se agora atendermos ao facto que

$$\|A\|_2 = \sup_{\|x\|_2=1} \|Ax\|_2 , \tag{2.14}$$

concluímos que o comprimento do maior semi-eixo da hiperelipse, referido no final do subcapítulo 2.4, é igual a $\|A\|_2$.

Se *A* é de ordem *n* e invertível então

$$\left\|A^{-1}\right\|_2 = \frac{1}{\sigma_n}, \tag{2.15}$$

onde σ_n é o menor valor singular de *A*. Com efeito,

$$A^{-1} = V \Sigma^{-1} U^T, \text{ com } \Sigma^{-1} = \text{diag}\left(\frac{1}{\sigma_1}, \cdots, \frac{1}{\sigma_n}\right). \tag{2.16}$$

Outra norma que também pode ser calculada é a norma de Frobenius. Tal como para norma euclideana, o cálculo da norma de Frobenius está directamente relacionado com os valores singulares, como veremos mais à frente.

Dada uma matriz *A*, a norma de Frobenius é, por definição,

$$\|A\|_F = \left(\sum_{i=1}^{m} \sum_{j=1}^{n} \left| a_{ij} \right|^2 \right)^{\frac{1}{2}}. \tag{2.17}$$

Note-se que $\|A\|_F = \sqrt{tr(A^T A)}$, onde *tr* é o traço da matriz.

A norma de Frobenius é invariante através de transformações ortogonais, pois se *Q* é uma matriz ortogonal, $\|QA\|_F = \sqrt{tr\left[(QA)^T (QA)\right]} = \sqrt{tr(A^T Q^T QA)} = \sqrt{tr(A^T A)} = \|A\|_F$.

Assim sendo, temos

$$\|A\|_F = \|U\Sigma V^T\|_F = \|\Sigma\|_F = (\sigma_1{}^2 + \ldots + \sigma_p{}^2)^{\frac{1}{2}}, \tag{2.18}$$

ou seja,

$$\|A\|_F^2 = \sigma_1{}^2 + \ldots + \sigma_p{}^2; \quad p = \min\{m,n\}. \tag{2.19}$$

2.6 – A característica de uma matriz

No que segue, designamos por *car* (*A*) a característica de uma matriz *A*. Muitos resultados da Álgebra Linear são apenas válidos para matrizes com característica completa; no entanto, problemas que podem ocorrer, sobretudo a nível computacional, como erros devido a arredondamentos, fazem com que seja difícil determinar com precisão a característica de uma matriz.

De seguida, vamos mostrar que a utilização da decomposição em valores singulares é uma solução para esse problema, por caracterizar eficientemente uma aproximação de matrizes de característica definida. A base dessa solução é o facto de o número de valores singulares não nulos de uma matriz *A* ser igual à sua característica.

De facto, se considerarmos os valores singulares de uma matriz *A*,

$$\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r > \sigma_{r+1} = \cdots = \sigma_p = 0, \tag{2.20}$$

podemos reescrever a decomposição em valores singulares de *A* na seguinte forma reduzida:

$$A = \begin{bmatrix} U_r & U_{(m-r)} \end{bmatrix} \begin{bmatrix} \sigma_1 & & 0 \\ & \ddots & \\ & & \sigma_r \\ 0 & & 0 \end{bmatrix} \begin{bmatrix} V_r{}^T \\ V_{(n-r)}{}^T \end{bmatrix} = U_r \, \Sigma_r \, V_r{}^T, \tag{2.21}$$

onde $U_r \in \mathbb{R}^{m \times r}$ contém as primeiras *r* colunas da matriz *U*, $\Sigma_r \in \mathbb{R}^{r \times r}$ contém as primeiras *r* colunas da matriz Σ e $V_r \in \mathbb{R}^{n \times r}$ contém as primeiras *r* colunas da matriz *V*.

Por outro lado, como U e V são ortogonais, car (A) = car ($U\Sigma V^T$) = car(Σ). Mas visto que matriz Σ é uma matriz diagonal, a sua característica é igual ao número de elementos não nulos da diagonal e portanto a característica de A é igual a r .

Repare-se que (2.21) pode ser escrita como a soma de matrizes com característica um, na forma

$$A = \sum_{j=1}^{r} \sigma_j \, u_j \, v_j^{\,T} \, . \tag{2.22}$$

Um aspecto que há a salientar é que por vezes em determinadas aplicações existem matrizes cujos valores singulares mais pequenos deveriam ser nulos, mas não o são por vários motivos, nomeadamente devido ao ruído em experiências. É comum substituir esses valores por zero, desprezando as suas contribuições, e considerar uma matriz aproximada. Pode assim pensar-se em A_k , ($k < r$) como uma boa aproximação a A, sendo

$$A_k = \sum_{j=1}^{k} \sigma_j \, u_j \, v_j^{\,T} \, . \tag{2.23}$$

De (2.22) e (2.23) verifica-se que $A - A_k$ é uma matriz com os valores singulares $\sigma_{k+1}, \ldots, \sigma_r$, logo, de (2.12)

$$\left\| A - A_k \right\|_2 = \sigma_{k+1} . \tag{2.24}$$

(Claro que $\sigma_{k+1} = 0$, se $k > r$).

Este facto é apresentado pelo seguinte teorema.

Teorema 2.2: Seja $A \in \mathbb{R}^{m \times n}$ com SVD como no teorema 2.1. Se $k < r = car(A)$ e $A_k = \sum_{j=1}^{k} \sigma_j \, u_j \, v_j^{\,T}$, então $\min_{car(B)=k} \left\| A - B \right\|_2 = \left\| A - A_k \right\|_2 = \sigma_{k+1}$.

Demonstração: (ver, por exemplo, [6])

Observe-se que, como $U^T A_k V = \text{diag}(\sigma_1,\ldots,\sigma_k,0,\ldots,0)$, temos $car(A_k) = k$ e $U^T(A - A_k)V = \text{diag}(0,\ldots,0,\sigma_{k+1},\ldots,\sigma_p)$. Logo $\|A - A_k\|_2 = \sigma_{k+1}$.

Suponhamos agora que $car(B) = k$, para alguma matriz $B \in \mathbb{R}^{m\times n}$. Então podemos encontrar vectores $x_1, \ldots, x_{n-k}$ ortonormais, tais que o núcleo de B é o espaço gerado por esses vectores.

O teorema das dimensões mostra que $\langle x_1,\ldots,x_{n-k}\rangle \cap \langle v_1,\ldots,v_{k+1}\rangle \neq \{0\}$.

Seja z um vector unitário (norma dois) nesta intersecção. Como $Bz = 0$ e $Az = \sum_{i=1}^{k+1} \sigma_i (v_i^T z)\, u_i$ temos

$$\|A - B\|_2^2 \geq \|(A - B)z\|_2^2 = \|Az\|_2^2 = \sum_{i=1}^{k+1} \sigma_i^2 (v_i^T z)^2 \geq \sigma_{k+1}^2 .$$

A matriz A_k é a melhor aproximação de A, no sentido da norma dois, de todas as matrizes de característica inferior a k. Portanto, se σ_{k+1} for suficientemente pequeno, é seguro considerar apenas k valores singulares. Neste caso diz-se que k é a ***característica efectiva*** da matriz A.

2.7 – Os quatro subespaços fundamentais

Sabemos que o núcleo da matriz A, $N(A)$, é o conjunto de vectores x, tais que $Ax=0$, e o espaço gerado pelas colunas de A, $R(A)$, é o conjunto de vectores b para os quais $Ax = b$, tem solução x.

Considerando as partições das matrizes U e V como se viu em (2.21) podemos afirmar que as colunas de U_r formam uma base ortonormal para o espaço gerado pelas colunas de A, ou seja uma base para $R(A)$.

As restantes colunas de U, $U_{(m-r)}$ formam uma base ortonormal para o núcleo esquerdo de A; ou seja para o complemento ortogonal do espaço gerado pelas colunas de A, $R(A)^{\perp} = N(A^T)$.

No que diz respeito às colunas de V_r, estas formam uma base ortonormal para o espaço gerado pelas linhas de *A*, ou seja para o complemento ortogonal do núcleo de *A*, $N(A)^{\perp} = R(A^T)$.

As colunas de $V_{(n-r)}$ formam uma base para o núcleo de *A*, $N(A) = R(A^T)^{\perp}$.

O seguinte esquema mostra os quatro subespaços fundamentais:

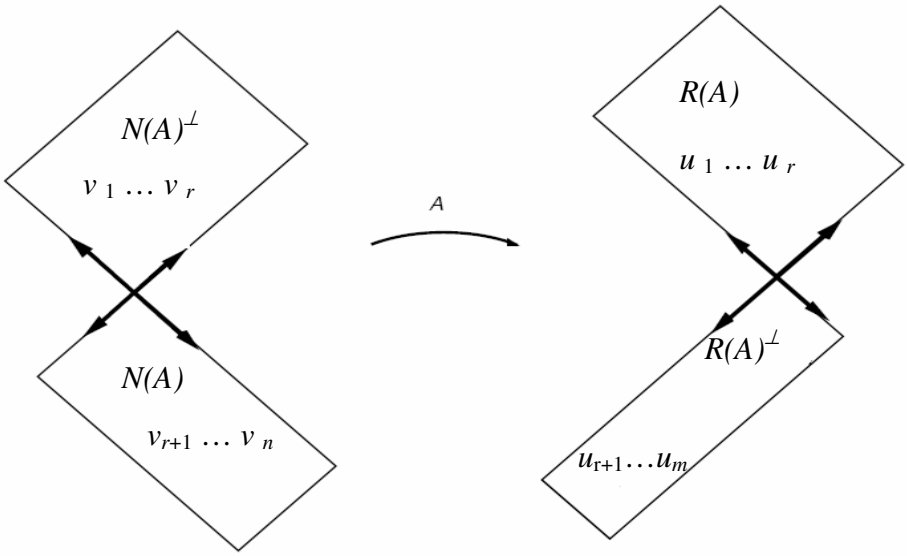


Fig. 2.4 - Os quatro subespaços fundamentais

Desta forma, tem-se que o núcleo de *A* corresponde ao complemento ortogonal do espaço gerado pelas linhas de *A* em $\mathbb{R}^n$ e o núcleo esquerdo de *A* corresponde ao complemento ortogonal do espaço gerado pelas colunas de *A* em $\mathbb{R}^m$.

Se a matriz *A* tem característica *r*, então o espaço gerado pelas colunas de *A* é igual ao espaço gerado pelas primeiras *r* colunas da matriz *U*, ou seja, $R(A) = R(U_r)$. Observe-se que $U_r = AV_r\Sigma_r^{-1}$; então $R(U_r) \subset R(A)$ e $A = U_r\Sigma_r V_r^T$ implica $R(A) \subset R(U_r)$; logo $R(A) = R(U_r)$.

Verifica-se também que o espaço gerado pelas linhas da matriz *A* é igual ao espaço gerado pelas primeiras *r* colunas da matriz *V*, ou seja $R(A^T) = R(V_r)$.

De forma análoga ao caso anterior, $V_r = A^T U_r \Sigma_r^{-1}$ implica $R(V_r) \subset R(A^T)$ e $A^T = V_r\Sigma_r U_r^T$ implica que $R(A^T) \subset R(V_r)$; logo $R(A^T) = R(V_r)$.

Os restantes vectores singulares formam as bases ortogonais para o núcleo de *A* e para o núcleo de A^T.

Tendo em conta (2.21), verifica-se que o núcleo de A é igual ao espaço gerado pelas últimas n - r colunas de V, pois esta partição da matriz A implica que $AV_{n-r} = 0$; logo $R(V_{n-r}) \subset N(A)$. Como V é uma matriz ortogonal, $R(V_{n-r})$ é o complemento ortogonal, em $\mathbb{R}^n$, de $R(V_r) = R(A^T)$; logo $N(A) = R(V_{n-r})$.

Verifica-se também que o núcleo de A^T é igual ao espaço gerado pelas últimas m - r colunas de U.

De forma análoga, $AU_{m-r} = 0$; logo $R(U_{m-r}) \subset N(A^T)$. Como U é uma matriz ortogonal, $R(U_{m-r})$ é o complemento ortogonal, em $\mathbb{R}^m$, de $R(U_r) = R(A)$; logo $N(A^T) = R(U_{m-r})$.

Se usarmos a decomposição em valores singulares de uma matriz A, podemos facilmente obter as projecções ortogonais sobre os quatro espaços associados a A. Assim, se A é uma matriz $m \times n$, com característica r, tem-se:

$U_r U_r^T$ é a projecção sobre $R(A)$; $U_{m-r} U_{m-r}^T$ é a projecção sobre $N(A^T)$

$V_r V_r^T$ é a projecção sobre $R(A^T)$; $V_{n-r} V_{n-r}^T$ é a projecção sobre $N(A)$.

De facto, por exemplo, para a projecção sobre $R(A)$ verifica-se que $R(U_r U_r^T) = R(A)$, $(U_r U_r^T)^2 = (U_r U_r^T)(U_r U_r^T) = U_r U_r^T$ e $(U_r U_r^T)^T = U_r U_r^T$.

2.8 – Condicionamento da Decomposição em Valores Singulares

O número de condição, $K(A)$, de uma matriz $A \in \mathbb{R}^{m \times n}$, relativamente à norma euclideana, pode ser definido, em termos dos valores singulares, como:

$$\begin{cases} K(A) = \dfrac{\sigma_{\max}(A)}{\sigma_{\min}(A)}, & \text{se } \sigma_{\min}(A) \neq 0, \\ K(A) = +\infty, & \text{se } \sigma_{\min}(A) = 0 \end{cases} \tag{2.25}$$

Para o caso particular $m = n$, o número de condição mede o grau de singularidade de uma matriz. Quanto maior é $K(A)$, mais A se aproxima de uma matriz singular.

Se considerarmos o problema da resolução de sistemas de equações lineares,

$$\mathrm{A}\ x = \mathrm{b}, \tag{2.26}$$

o número de condição, atrás definido, mede a sensibilidade da solução x a alterações em b.

Exemplo:

$$A\ x = b \Leftrightarrow \begin{bmatrix} 1.2969 & 0.8648 \\ 0.2161 & 0.1441 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 0.8642 \\ 0.1440 \end{bmatrix}.$$

A solução exacta deste sistema é $x = \begin{bmatrix} 2 \\ -2 \end{bmatrix}$.

Se considerarmos $\tilde{b} = \begin{bmatrix} 0.86419999 \\ 0.14400001 \end{bmatrix}$, a solução obtida é $\tilde{x} = \begin{bmatrix} 0.9911 \\ -0.4870 \end{bmatrix}$. (Cálculos efectuados com o programa MATLAB).

Verificamos que uma pequena alteração no vector b, provoca um grande erro na solução.

Para este exemplo, $K(A)$= $2.497292665336376\times10^{8}$, e o sistema é mal condicionado.

Na realidade demonstra-se que se $Ax = b$, A invertível e $b \neq 0$, $(A + \Delta A)y = b + \Delta b$, $\dfrac{\|\Delta A\|}{\|A\|} < \dfrac{1}{K(A)}$, então

$$\frac{\|x - y\|}{\|x\|} \leq \frac{K(A)}{1 - K(A)\dfrac{\|\Delta A\|}{\|A\|}} \left(\frac{\|\Delta A\|}{\|A\|} + \frac{\|\Delta b\|}{\|b\|} \right). \tag{2.28}$$

No entanto, a determinação da decomposição em valores singulares, para qualquer matriz, é um problema bem condicionado, ou seja, pequenos erros relativos nos dados produzem pequenos erros relativos no resultado.

Sejam

$$A = U\Sigma V^T$$

e

$$\hat{U} = W + \Delta U \text{ , onde } W^T W = I \text{ e } \|\Delta U\|_2 \le \varepsilon ; \tag{2.29}$$

$$\hat{V} = Z + \Delta V \text{ , onde } Z^T Z = I \text{ e } \|\Delta V\|_2 \le \varepsilon ; \tag{2.30}$$

$$\hat{\Sigma} = W^T (A + \Delta A) Z \text{ onde } \|\Delta A\|_2 \le \varepsilon \|A\|_2 = \varepsilon \sigma_1 . \tag{2.31}$$

As matrizes $\hat{U}$, $\hat{V}$ e $\hat{\Sigma}$ são as aproximações de *U*, *V* e Σ respectivamente. O valor ε é um pequeno múltiplo da precisão utilizada.

Tem-se, do teorema 2.2, que

$$\begin{aligned}
\sigma_k &= \min_{car(B_1)=k-1} \|A - B_1\|_2 \\
&= \min_{car(B_1)=k-1} \| W^T (A - B_1) Z\|_2 \\
&= \min_{car(B_1)=k-1} \| W^T AZ - W^T B_1 Z\|_2 \\
&= \min_{car(B)=k-1} \| W^T AZ - B\|_2 \\
&= \min_{car(B)=k-1} \| (\hat{\Sigma} - B) - W^T (\Delta A) Z\|_2 ,
\end{aligned}$$

uma vez que $W^T AZ = \hat{\Sigma} - W^T (\Delta A) Z$ de (2.31).

Como $\|W^T (\Delta A) Z\|_2 \le \varepsilon \|A\|_2 = \varepsilon \sigma_1$ e $\min_{car(B)=k-1} \|\hat{\Sigma}_k - B\|_2 = \hat{\sigma}_k$ verifica-se que $|\sigma_k - \hat{\sigma}_k| < \varepsilon \sigma_1 = \varepsilon \|A\|_2$, para *k* = 1, …, *p*. Sendo $\sigma_1, \cdots, \sigma_p$ os valores singulares de *A* e $\hat{\sigma}_1, \cdots, \hat{\sigma}_p$ os valores calculados numericamente.

Concluímos assim que os valores singulares calculados $\hat{\sigma}_1, \cdots, \hat{\sigma}_p$ estão próximos dos valores singulares exactos. No entanto, a precisão relativa no cálculo dos valores singulares só é realmente boa para valores singulares não muito inferiores a σ_1. Isto tem uma consequência prática: se for encontrado um valor singular com ordem de grandeza

próxima da precisão da máquina não será possível dizer, com certeza, se ele é diferente de zero ou não. Este facto pode gerar uma indeterminação acerca da característica da matriz *A*.

Exemplo:

Seja

$$A = \begin{bmatrix} -1{,}4 & -0.8 & 1.1 & 0.4 \\ 0.3 & 0.6 & -0.45 & 0.3 \\ 0.8 & 1.1 & -0.95 & 0.55 \\ -0.4 & 0.2 & 0.1 & 0.1 \\ 0.5 & 0.5 & -0.5 & 0.25 \end{bmatrix}$$

$$= U \begin{bmatrix} 2.8835 & 0 & 0 & 0 \\ 0 & 0.69684 & 0 & 0 \\ 0 & 0 & 2.0386\times10^{-16} & 0 \\ 0 & 0 & 0 & 4.0255\times10^{-17} \\ 0 & 0 & 0 & 0 \end{bmatrix} V^T$$

Os cálculos foram efectuados com o programa MATLAB e têm precisão $\varepsilon = 2.2204\times10^{-16}$; portanto, em termos práticos, não se pode distinguir, de zero, os dois menores valores singulares de zero. Assim, numericamente, a característica de *A* é dois, embora os valores singulares $\hat{\sigma}_3$ e $\hat{\sigma}_4$ não sejam nulos.

Em geral, os valores singulares de uma matriz são a melhor forma de determinar numericamente a sua característica, mas, como já se viu, está sujeita a erros.

Quanto maior for ε e a imprecisão na obtenção da matriz *A* (por vezes é obtida a partir de resultados de experiências), mais incerta será a determinação da característica.

2.9 – Resolução de sistemas lineares no sentido dos mínimos quadrados

Consideremos o sistema de equações $Ax = b$, onde $A = \begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{mn} \end{bmatrix} \in \mathbb{R}^{m\times n}$, $x = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} \in \mathbb{R}^n$ e $b = \begin{bmatrix} b_1 \\ \vdots \\ b_m \end{bmatrix} \in \mathbb{R}^m$. Para $m \neq n$, a matriz A é rectangular e como tal não tem inversa. No entanto, com o auxílio da decomposição em valores singulares podemos generalizar àquelas matrizes a noção de matriz inversa.

Seja

$$A \in \mathbb{R}^{m\times n} \text{ tal que } A = U\Sigma V^T \text{ e } car(A) = r .$$

Podemos definir

$$A^\dagger = V\Sigma^\dagger U^T, \text{onde} \Sigma^\dagger = \left(\frac{1}{\sigma_1},\ldots,\frac{1}{\sigma_r},0,\ldots,0\right) \in \mathbb{R}^{m\times n} \qquad (2.32)$$

A $A^\dagger$ dá-se o nome de ***pseudo – inversa*** *de A*. É a única matriz que verifica as quatro condições de Moore-Penrose:

i) $AA^\dagger A = A$

ii) $A^\dagger AA^\dagger = A^\dagger$

iii) $(AA^\dagger)^T = AA^\dagger$

iv) $(A^\dagger A)^T = A^\dagger A$.

No caso de *A* ser invertível, $A^\dagger = A^{-1}$.

Calculemos agora a expressão para $A^\dagger$, sendo $m > n$ e $car\,(A) = n$. Nesta situação, $(A^T A)$ é uma matriz invertível.

Como $(A^T A)^{-1} = (V\Sigma^T U^T U\Sigma V^T)^{-1} = (V\Sigma^T\Sigma V^T)^{-1} = V(\Sigma^T\Sigma)^{-1}V^T$, verifica-se que

$$A^\dagger = V\Sigma^\dagger U^T = V\Sigma^\dagger(\Sigma^T)^\dagger V^T V\Sigma^T U^T = V(\Sigma^T\Sigma)^{-1}V^T V\Sigma^T U^T = (A^T A)^{-1}A^T . \qquad (2.33)$$

Por outro lado, se $m < n$ e $car\,(A) = m$, a matriz (AA^T) é invertível e constata-se que $(AA^T)^{-1} = (U\Sigma V^T V\Sigma^T U^T)^{-1} = (U\Sigma\Sigma^T U^T)^{-1} = U(\Sigma\Sigma^T)^{-1}U^T$.

Daqui resulta que

$$A^{\dagger} = V\,\Sigma^{\dagger}U^{T} = V\Sigma^{T}U^{T}\,U(\Sigma^{T})^{\dagger}\Sigma^{\dagger}U^{T} = V\Sigma^{T}U^{T}\,U(\Sigma\Sigma^{T})^{-1}U^{T} = A^{T}(AA^{T})^{-1}. \qquad (2.34)$$

a) A matriz A tem característica completa

Começamos por considerar o caso em que $m > n$ e $car\ (A) = n$, ou seja, A tem característica completa. Nesta situação o sistema possui um maior número de equações do que incógnitas, é por isso sobredeterminado.

Em geral $b \notin R(A)$ por isso o sistema não tem solução. É necessário encontrar uma solução aproximada. Como $Ax - b \neq 0$, consideramos

$$r = Ax - b\,, \qquad (2.35)$$

o resíduo. Pretendemos encontrar $\hat{x} = x_{MQ}$ que minimiza o comprimento desse vector.

A $\hat{x} = x_{MQ}$ chama-se ***solução aproximada no sentido dos mínimos quadrados*** de $Ax=b$.

Calculemos

$$\|r\|_2^2 = \|Ax-b\|_2^2 = (Ax-b)^T(Ax-b) = x^T A^T Ax - 2b^T Ax + b^T b\,. \qquad (2.36)$$

Determinemos agora o mínimo de (2.36). Para isso, temos de resolver

$$\frac{\partial\left[(Ax-b)^T(Ax-b)\right]}{\partial x_i} = 0\,,\ i=1,\ \ldots,\ n. \qquad (2.37)$$

Se juntarmos as n derivadas parciais numa única equação obtemos

$$2x^T A^T A - 2b^T A = 0\,, \qquad (2.38)$$

ou seja

$$A^T Ax = A^T b\,. \qquad (2.39)$$

Como $A_{m\times n}$ $(m > n)$ tem característica completa, $A^T A$ é invertível e temos

$$(A^T A)^{-1} A^T Ax = (A^T A)^{-1} A^T b\,, \qquad (2.40)$$

$$x_{MQ} = (A^T A)^{-1} A^T b\,, \qquad (2.41)$$

ou seja

$$x_{MQ} = A^{\dagger} b\,. \qquad (2.42)$$

Analisemos agora o caso em que $m < n$ e $car\ (A) = m$, tendo portanto A característica completa.

Nestas circunstâncias o sistema $Ax = b$ tem mais variáveis do que equações, ou seja, é indeterminado. Existe um número infinito de soluções, como tal procura-se a solução de norma mínima. Pretende-se encontrar uma solução x_{NM} que minimize $\|x\|_2$.

A solução de norma mínima é dada por

$$x = A^{\dagger} b = A^T (AA^T)^{-1} b, \tag{2.43}$$

b) A matriz A tem característica incompleta

No caso da característica ser incompleta, isto é, $car(A) < \min(m, n)$, o sistema é indeterminado. Nesta situação há um número infinito de soluções, é necessário procurar a solução de norma mínima. Essa solução é

$$x_{NM} = \sum_{i=1}^{r} \frac{u_i^T b}{\sigma_i} v_i, \tag{2.44}$$

sendo r é a característica de A.

A expressão (2.44) minimiza $\|Ax - b\|_2$ e tem a menor norma dois e $\|Ax_{NM} - b\|_2^2 = \sum_{i=r+1}^{m} (u_i^T b)^2$, [6].

De facto, para qualquer $x \in \mathbb{R}^n$ e $\alpha = V^T x$ temos:

$$\|Ax - b\|_2^2 = \|(U^T AV)(V^T x) - U^T b\|_2^2 = \|\Sigma\alpha - U^T b\|_2^2$$

$$= \sum_{i=1}^{r} (\sigma_i \alpha_i - u_i^T b)^2 + \sum_{i=r+1}^{m} (u_i^T b)^2$$

Se o vector x resolve o problema dos mínimos quadrados, então $\alpha_i = \left(\frac{u_i^T b}{\sigma_i} \right)$, para $i = 1, \ldots, r$. Se considerarmos $\alpha = 0$ para $i = r + 1, \ldots, n$, então verifica-se que x tem norma dois mínima.

Capítulo 3

Compressão de imagens

3.1 – Introdução

Desde há alguns anos são bastantes as áreas onde existe a necessidade de transmitir imagens de uma forma adequada. Existem várias técnicas para o fazer. Uma muito utilizada é a que faz uso da decomposição em valores singulares.

A transmissão de imagens, dependendo da sua resolução, pode tornar os sistemas de transmissão pouco eficientes. Para solucionar este problema, são utilizados recursos de compressão que visam reduzir o tempo de transmissão, diminuindo o espaço necessário para armazenamento das imagens estáticas ou vídeo. Neste âmbito, a SVD pode ser utilizada para transmitir uma imagem de forma adequada.

No entanto, deve ter-se sempre em consideração a possibilidade de perda de qualidade da imagem. Esta perda de qualidade por vezes é imperceptível a olho nu e dependendo do fim a que se destinam as imagens o sistema de transmissão pode ou não, ser eficiente.

3.2 – Representação de imagens

Computacionalmente as imagens são geralmente representadas através de matrizes. A posição de cada *pixel* na matriz corresponde à sua posição no plano da imagem e a

intensidade de cada *pixel* é representada pelo valor de cada entrada na matriz. Assim, dada uma imagem, o computador faz a sua leitura armazenando os dados numa matriz. Se a imagem estiver a preto e branco, cada *pixel* é substituído pelo número 0 (zero) ou pelo número 1 (um); caso a imagem esteja em 'grayscale'- escala de cinzentos, cada *pixel* é substituído por um número real no intervalo [0, 1] ou por um número inteiro no intervalo [0, 255], onde 0 corresponde ao preto e 255 ao branco puro.

No que diz respeito a imagens coloridas, a informação transmitida é multidimensional. O espaço de cor usado pela maioria dos computadores é o espaço tridimensional RGB[2]. Neste caso, cada *pixel* pode ser representado por um vector tridimensional onde cada componente representa a intensidade de cada uma das cores primárias: vermelho, verde e azul.

Neste trabalho as imagens serão convertidas para uma escala de cinzentos, usando o comando *rgb2gray,* no programa MATLAB.

Nas situações em que se trabalha com muitas imagens é conveniente que, após a leitura da imagem, a matriz seja transformada num vector. Esta transformação tem como objectivo reduzir a complexidade computacional.

Cada imagem, com resolução $m \times n$, é considerada inicialmente como uma matriz de dimensão $m \times n$. Depois de efectuada a vectorização, a imagem corresponde a um vector num espaço de dimensão $m\,n$.

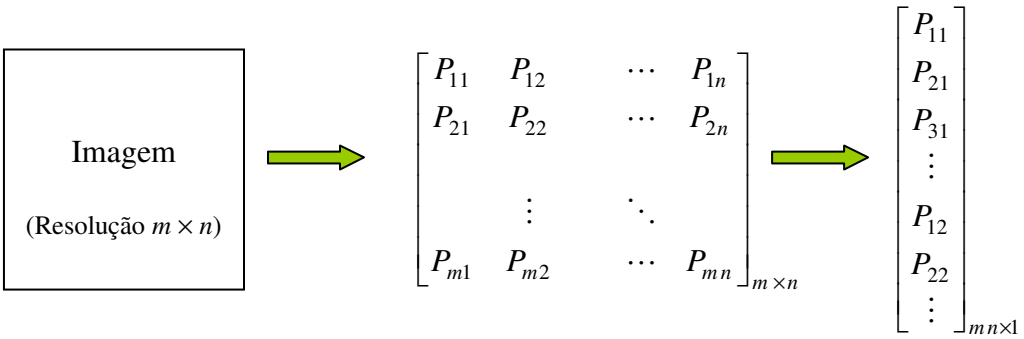


Fig. 3.1 – Leitura e vectorização de uma imagem

O processo de vectorização pode ser feito de diversas formas. Os *pixeis* podem ser organizados de maneira que uma coluna da matriz original fique disposta após outra, como se pode observar na figura 3.1, ou uma linha após outra ou ainda uma ordem definida por uma função específica.

No caso do reconhecimento e reconstrução facial apresentados no capítulo 4, algumas das imagens têm uma resolução de 256×256. Quando é efectuado o processo

[2] Red Green Blue

de vectorização como exemplificado no esquema da figura 3.1, cada imagem é referida como um vector num espaço de dimensão 65536. Assim, este espaço vectorial pode descrever todas as imagens que têm a mesma resolução.

3.3 – A utilização da SVD na compressão de imagens

A SVD pode ser útil na transmissão de imagens no sentido em que a matriz inicial pode ser aproximada de uma matriz com característica inferior, tal como foi visto no capítulo anterior.

De seguida apresentamos um exemplo com uma fotografia com resolução 1536×2048.

Fig. 3.2 - Biblioteca Central da Universidade do Algarve

Consideremos A como sendo a matriz que resulta da leitura da imagem. Façamos depois uma aproximação do tipo: $A \approx A_k = \sum_{j=1}^{k} \sigma_j \, u_j \, v_j^T$.

A matriz que corresponde à imagem é uma matriz $A \in \mathbb{R}^{1536 \times 2048}$ e tem característica completa [3].

Elaborámos um programa em MATLAB que após a leitura da imagem da figura 3.2 converte-a numa imagem em escala de cinzentos. Posteriormente, calcula a SVD da matriz que representa a imagem e faz aproximações do tipo acima referido.

[3] Cálculo efectuado no programa MATLAB

Vejamos algumas dessas aproximações através de matrizes com característica inferior e igual a 1536.

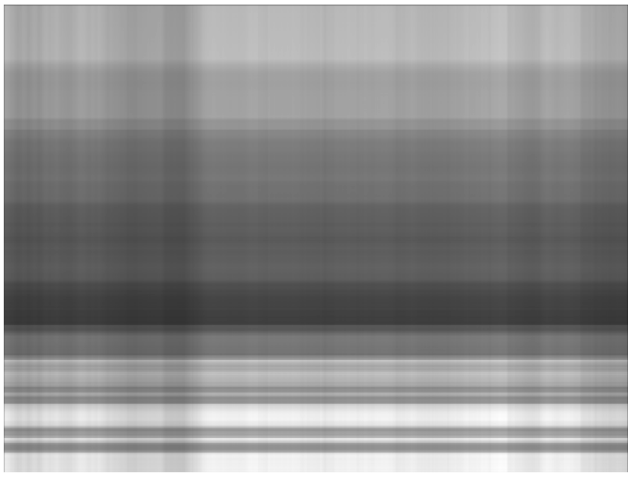


Fig. 3.3 – Imagem obtida para $k = 10$

Fig. 3.4 – Imagem obtida para $k = 30$

Fig. 3.5 – Imagem obtida para $k = 40$

Fig. 3.6 – Imagem obtida para $k = 100$

Fig. 3.7 – Imagem obtida para $k = 1536$

Verifica-se que a nitidez vai aumentando e que, para valores de k próximos de 40, já se obtém uma boa aproximação da imagem original. Isto é devido ao facto de os primeiros valores singulares conterem a informação essencial.

No gráfico 3.1 estão representados os primeiros cem valores singulares de A.

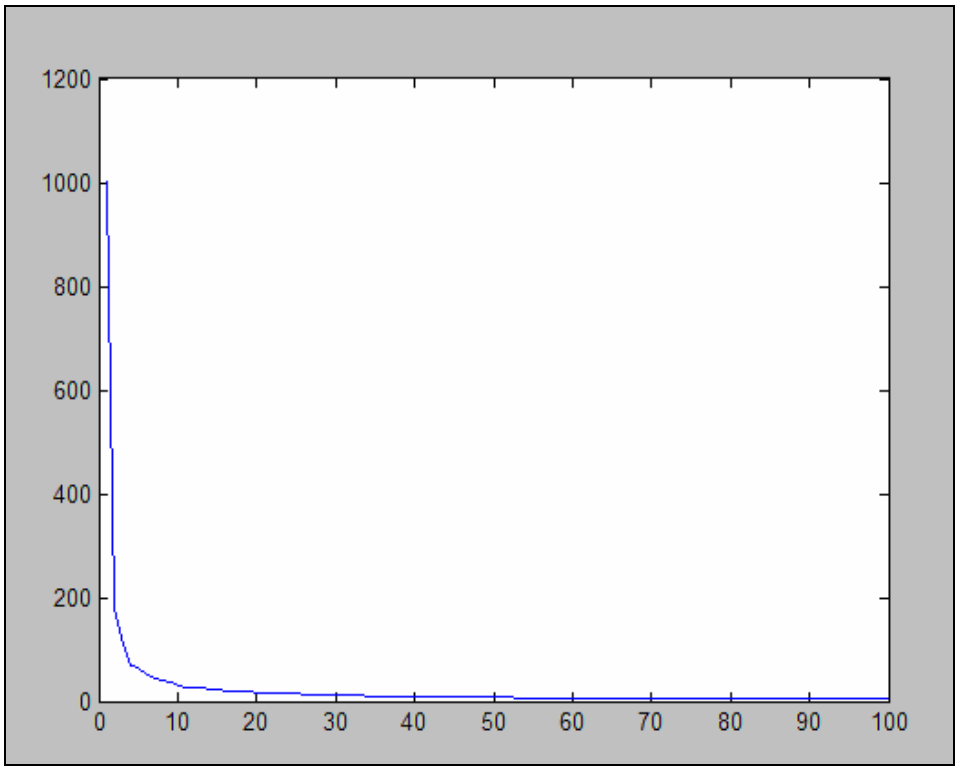


Gráfico 3.1 - Os primeiros cem valores singulares de A

Constata-se que os valores singulares decrescem rapidamente e que o 60º valor já está próximo de zero, enquanto que o 1º é cerca de 1000. Daí resulta que basta considerar os primeiros valores singulares, por exemplo apenas 40, para obter uma boa aproximação da imagem inicial.

A qualidade da aproximação A_k, com respeito a A, pode ser quantificada [7], através do erro relativo,

$$\varepsilon = \frac{\sigma_{k+1}}{\sigma_1} \tag{3.1}$$

e do raio de compressão,

$$r = \frac{(m+n)k}{m\,n}\,. \tag{3.2}$$

O erro relativo, ε, caracteriza os valores singulares dominantes, os que não podem ser ignorados, caso contrário a imagem perde por completo a qualidade.

Verifica-se que a imagem começa a perder qualidade para valores superiores a 10^{-3} e para valores superiores a 10^{-2} a qualidade da imagem é realmente pobre.

O raio de compressão relaciona a quantidade de armazenamento requerida para A_k com respeito à necessária para A. Como se pode observar o raio é menor que a unidade só se $k < \frac{mn}{m+n}$.

Na tabela 3.1 estão listados, para alguns valores de k, os valores singulares, o erro relativo e o raio de compressão.

k	σ_k	ε	r
1	1002.9706	1.7629×10^{-1}	1.1393×10^{-3}
5	64.3929	5.0022×10^{-2}	5.6966×10^{-3}
10	30.5557	2.7162×10^{-2}	1.1393×10^{-2}
15	21.8884	2.0579×10^{-2}	1.7090×10^{-2}
30	11.6811	1.1486×10^{-2}	3.4180×10^{-2}
40	9.0650	8.8138×10^{-3}	4.5573×10^{-2}
100	4.3309	4.2574×10^{-3}	1.1393×10^{-1}
500	0.6471	6.4389×10^{-4}	5.6966×10^{-1}
1000	0.1171	1.1566×10^{-4}	1.1393
1500	0.0202	1.9841×10^{-5}	1.709

Tabela 3.1 – Valores singulares, erro relativo e raio de compressão para a figura 3.2

Verifica-se um grande decréscimo logo nos primeiros valores singulares, o que indica que, de facto, são estes os mais importantes. No entanto observe-se a primeira aproximação que foi feita à imagem, considerando apenas os primeiros dez valores singulares. Não se consegue verificar de que imagem se trata, mas de facto a informação principal esta lá. Ao considerarmos por exemplo mais vinte valores singulares já nos conseguimos aperceber de que imagem se trata, depois à medida que se acrescentam mais valores singulares, outros detalhes são acrescentados à imagem.

Este procedimento é bastante utilizado na Internet para descarregar fotos, diminuindo o tempo de espera por parte de utilizador e ao mesmo tempo poupando lugares na memória do computador. Por vezes também é utilizado para transmitir imagem de satélites para a Terra.

Este método tem ainda outras aplicações, por exemplo pode-se também modificar apenas uma parte da imagem representada pela matriz A, para isso basta partí-la

adequadamente e aproximar a submatriz correspondente por uma matriz de característica inferior. Para a imagem do exemplo anterior, por exemplo, pode-se querer ver com alguma definição a entrada da biblioteca e o resto da imagem pode ter menos nitidez.

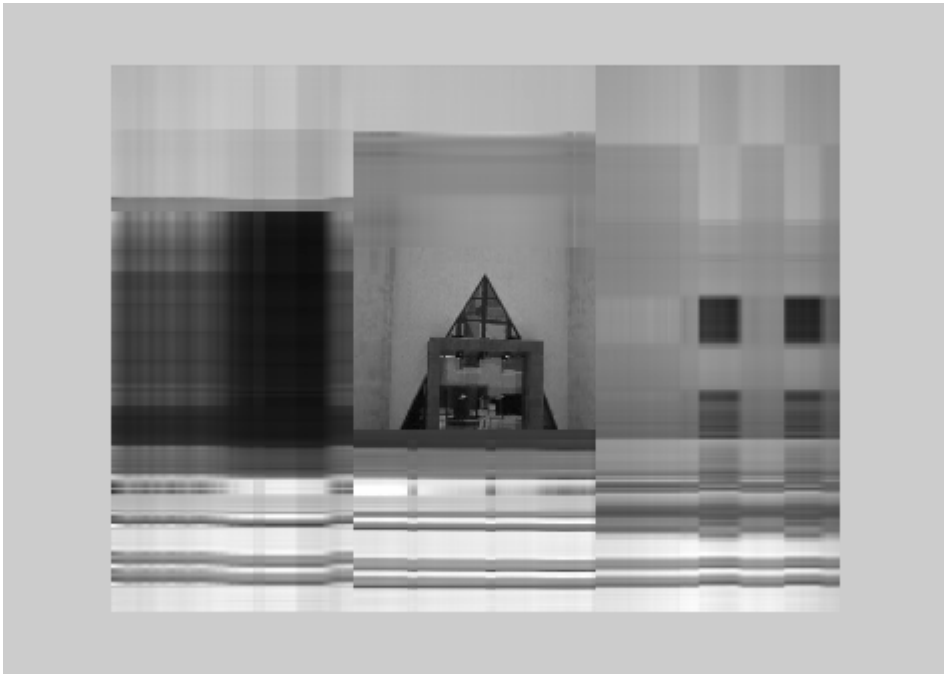


Fig. 3.8 – Biblioteca (entrada com mais nitidez do que o resto da imagem)

Dada uma matriz A correspondente a uma imagem que se pretende transmitir, a ideia é dividir a matriz em blocos, como por exemplo se apresenta a seguir:

$$A = \left[\begin{array}{c|c|c} & A_2 & \\ \hline A_1 & A_3 & A_5 \\ \hline & A_4 & \end{array}\right]$$

Posteriormente aplica-se a SVD a cada um dos blocos. Nesta fase dever-se-á ter em conta que podemos escolher a característica que se considere necessária para cada um dos blocos. Isto de forma a transmitir a imagem com a qualidade que se pretende ocupando o menor espaço possível na memória do computador.

Isto significa que os valores singulares nulos podem ser ignorados porque não transmitem qualquer informação acerca de A.

Foi elaborado um programa em MATLAB para comprimir imagens por blocos, como se pode observar na figura 3.8. Utilizámos cinco blocos como mostra o esquema anterior e considerámos característica 2 para todos os blocos, à excepção do bloco A_3, para o qual considerámos característica 40. Este último, por ter maior característica corresponde à parte da imagem com maior nitidez.

O programa pode ser consultado no Anexo B.

Capítulo 4

Reconstrução e reconhecimento facial

4.1 – Introdução

Um dos sistemas de identificação mais usados, e aquele a que, por enquanto, estamos mais habituados, é a utilização de senhas / códigos (password/ PIN). Todos os dias o utilizamos quando, por exemplo, efectuamos operações bancárias em máquinas automáticas. Mas este, é um sistema que é facilmente corrompido. Todavia, este tipo de identificação de pessoas associado a um eficiente sistema de detecção facial é uma opção pouco evasiva uma vez que não requer uma acção específica do utilizador.

O reconhecimento facial, que pertence à área da biometria, é uma das soluções possíveis, tendo o seu desenvolvimento automático começado em meados dos anos setenta, do século XX.

Para além do reconhecimento facial há outros sistemas biométricos para identificação de pessoas, como por exemplo, o reconhecimento da impressão digital, da retina, da íris, da voz, ou da dinâmica da escrita manual.

A face humana apresenta um problema interessante a nível computacional pela sua imprecisão e complexidade. As faces têm a sua estrutura geral, à qual se acrescenta a originalidade dos detalhes e dos traços, a cor da pele, a adição de acessórios, como

óculos escuros, os quais propõem situações amplamente variadas que normalmente não são vistas em outras aplicações de processamento de imagem e de reconhecimento. Daí o interesse em fazer o reconhecimento de pessoas através da sua face.

Os sistemas de reconhecimento facial, que permitem comparar uma imagem facial com um conjunto de imagens armazenadas, transitaram do laboratório para a realidade e são bastante utilizados, para fazer face às constantes exigências da sociedade. Por exemplo, actualmente, uma delas é o combate ao terrorismo. A pressão da sociedade implicou um rápido desenvolvimento de algoritmos e da tecnologia.

O reconhecimento facial foi escolhido como um problema de visão computacional no início dos anos setenta, tendo progredido bastante devido aos contributos de Kirby e Sirovich [8] e mais tarde Turk e Pentland [9].

Naquela altura, o interesse em identificar faces estava nos padrões e determinados traços faciais; com o passar do tempo, desenvolveu-se para criar um conjunto de faces que combinadas podem formar qualquer face, prestando atenção assim às variações matemáticas entre as faces das pessoas e não às suas características.

Actualmente, o desafio tecnológico não está em lembrar-se de muitas faces mas sim tentar diferenciar as pessoas numa grande base de imagens (porque quanto mais indivíduos forem, menos diferenciados eles são).

Deste modo, na identificação facial compara-se uma imagem de entrada (imagem de teste) com um conjunto de imagens (conjunto de treino) e verifica-se qual destas é a melhor aproximação da outra.

4.2 – Conjunto de treino

O conjunto de treino é um conjunto de imagens faciais escolhido para ser representativo de todas as faces que um sistema pode encontrar.

A construção deste conjunto é um passo muito importante em todo o processo de reconhecimento e reconstrução facial, pois caso este conjunto não seja bem construído os resultados obtidos não serão os melhores [10].

Para a obtenção do conjunto de treino é necessário que as faces sejam normalizadas no que diz respeito à posição, tamanho, orientação, e luminosidade, isto é, todas as faces devem estar na mesma posição, ter o mesmo tamanho, etc. Existem processos para fazer a normalização das faces mas que não serão aqui explorados; parte-se do princípio que todas as faces no conjunto de treino estão normalizadas.

Naturalmente, nem todas as faces estão incluídas no conjunto de treino, podendo até haver alguns indivíduos que são tão únicos que o sistema não consegue abranger.

Há dois tipos de sistemas de reconhecimento facial: o ***sistema em tempo real*** (pode ser visto um exemplo em [11]), em que são armazenadas várias imagens de uma pessoa numa base de dados, e o ***sistema em tempo não real***, onde é armazenada apenas uma imagem por pessoa.

Neste trabalho efectuaram-se testes de reconhecimento com os dois tipos de conjunto de treino atrás referidos.

Para o sistema em tempo não real, construímos o '***Conjunto de Treino 1'***, com uma imagem facial por pessoa. Este conjunto é constituído por 150 imagens faciais, cada uma com resolução 256×256. As condições de luminosidade são idênticas e foram retirados o fundo e o cabelo das fotos. Na figura 4.1 podem ver-se exemplos de faces neste conjunto de treino. Estas imagens faciais podem ser encontradas em http://www.stat.ucla.edu/~sczhu/Courses/UCLA/Stat_231/Face_modeling.html.

Fig.4.1 - Exemplos de imagens faciais no '***Conjunto de Treino 1'***

Para o sistema em tempo real temos o '***Conjunto de treino 2'***, com 10 imagens faciais por pessoa (mais expressões faciais e diferentes ângulos de observação). As condições de luminosidade são diferentes, as expressões faciais variam e por vezes há a utilização de adereços (óculos). Foram escolhidas imagens de 9 pessoas, cada uma delas tem resolução 112×92. Temos na figura 4.2 exemplos de imagens de 3 faces diferentes neste conjunto de treino, com 5 fotografias cada. Estas imagens podem ser encontradas em http://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html.

Fig.4.2 - Exemplos de imagens faciais no '***Conjunto de Treino 2'***

4.3 – Faces próprias

Actualmente existem várias técnicas para se fazer o reconhecimento facial de uma pessoa. De entre as diferentes técnicas, destaca-se a das faces próprias, também conhecida por *eigenfaces*. Este método surgiu devido aos contributos de Sirovich e Kirby [8] em 1987, tendo sido desenvolvido mais tarde, no MIT Media Lab[4], por Turk e

[4] Podem ser vistos alguns trabalhos em http://www.media.mit.edu/.

Pentland [9] em 1991. Presentemente é uma das práticas com maior sucesso no problema do reconhecimento facial.

A técnica das faces próprias consiste no seguinte: de um conjunto inicial de imagens faciais num certo espaço (de imagens), encontrar um subespaço que seja representativo de todas as faces, ou seja, que preserve a informação mais significativa Este procedimento consiste na extracção das características relevantes contidas numa imagem facial, de modo a representá-la o melhor possível.

Como se viu no capítulo 3, cada imagem bidimensional, com resolução $m \times n$, depois de passar pelo processo de vectorização, pode ser considerada como um vector (ou um ponto) num espaço de dimensão mn. Ao utilizarmos a técnica das faces próprias, aplica-se este procedimento a todas as imagens do conjunto de treino. Obtemos assim p vectores de dimensão mn, sendo p o número de imagens no conjunto de treino. O espaço gerado por estes vectores é designado por espaço de faces. As faces próprias são vectores ortogonais que formam uma base para o espaço de faces.

Noutros sistemas, é necessário caracterizar as faces e, para tal, tem de se verificar a presença de características faciais tais como olhos, nariz, boca, sobrancelhas. Perante a utilização de faces próprias, sem analisar as características faciais separadamente, analisa-se a variação dos valores assumidos pelos *pixeis* num conjunto de imagens faciais.

As faces próprias podem ser obtidas através de uma técnica conhecida por Análise de Componentes Principais [12].

A Analise em Componentes Principais tem como objectivo reduzir a dimensão do espaço original, através da eliminação das variáveis de menor variância.

Neste caso em concreto, da reconstrução e reconhecimento de faces, os vectores que representam as faces, são reescritos noutro sistema de eixos mais conveniente para a sua análise. Estes novos vectores são as componentes principais e as suas coordenadas são uma combinação linear das coordenadas dos vectores originais. Para além disso, são ortogonais entre si e ordenados em termos de quantidade de variância dos dados. Isto significa que se os dados estão muito afastados uns dos outros a variância é grande e a estes é associado a primeira componente principal; caso estejam próximos, a sua variância é menor e está-lhes associado outro vector. Assim sendo, o primeiro vector encontra-se na direcção de maior variância. Portanto, a primeira componente principal contém mais informação acerca das faces do que a segunda componente principal, que não abrange informações contempladas anteriormente e assim sucessivamente.

Tendo em conta que os eixos são ortogonais verifica-se que a correlação entre as componentes principais é nula.

O número de total componentes principais, ou seja, de faces próprias é igual ao número total de vectores originais e apresenta a mesma informação que estes vectores. Porém, este método permite a redução do número total de variáveis, uma vez que as primeiras componentes principais contêm normalmente mais de 90% da informação estatística dos dados originais.

Verifica-se que esta técnica reduz o número total de variáveis quando existe redundância nos dados, ou seja, perante dados correlacionados.

Para averiguar a existência ou não de redundância deve analisar-se a matriz de covariância.

4.3.1 – A matriz de covariância

As imagens faciais, por vezes, são difíceis de serem reconhecidas, pois todas possuem basicamente a mesma estrutura. É devido a esta semelhança entre as faces humanas que elas possuem uma grande correlação. Portanto cada *pixel* é correlacionado com os outros *pixeis*. Tal como foi referido anteriormente, esta correlação pode ser verificada através do estudo da matriz de covariância.

Sejam f_j, onde $j = 1, \ldots, n$, os vectores que, depois do processo de vectorização, referido no capítulo 3, representam *n* faces. Temos assim uma distribuição de vectores, cuja matriz de covariância é dada por:

$$C = \frac{1}{n}\sum_{j=1}^{n}\left(f_j - a\right)\left(f_j - a\right)^T , \tag{4.1}$$

onde $a = \frac{1}{n}\sum_{j=1}^{n} f_j$ é a média das *n* faces.

Considerando $x_j = f_j - a$, normalmente designado por ***caricatura*** da face *j* – a versão do desvio não normalizado, podemos reescrever *C* na forma:

$$C = XX^T , \tag{4.2}$$

onde

$$X = \frac{1}{\sqrt{n}}\left[x_1\ x_2\ \ldots\ x_n\right]. \tag{4.3}$$

Quando calculamos a covariância das faces interessa-nos investigar a correlação entre esses vectores.

Geometricamente procuramos a direcção que melhor se aproxima dessa distribuição de vectores, ou seja procuramos a direcção *u* tal que:

$$\mu = \max_{\|u\|_2=1} \frac{1}{n} \sum_{j=1}^{n} (u^T x_j)^2 . \tag{4.4}$$

Tendo em conta a matriz de covariância escrita na forma (4.2) é possível reescrever (4.4) como:

$$\mu = \max_{\|u\|_2=1} \left(u^T \; C \; u \right). \tag{4.5}$$

Uma vez que *C* é uma matriz real e simétrica, é diagonizável com uma matriz ortonormal U_C:

$$C = U_C \Lambda_C U_C{}^T , \tag{4.6}$$

onde $\Lambda_C = \text{diag}(\lambda_1, \ldots, \lambda_m)$, com λ_j ordenados por ordem decrescente. Os λ_j são todos positivos ou nulos.

Assim,

$$\mu = \max_{\|u\|_2=1} \left(u^T \; U_C \Lambda_C U_C{}^T \; u \right) = \max_{\|y\|_2=1} \left(y^T \Lambda_C \; y \right). \tag{4.7}$$

Como U_C é ortonormal e $y = U_C{}^T u$, sabemos que $\|y\|_2 = \|u\|_2 = 1$, temos de calcular

$$\mu = \max_{\|y\|_2=1} \sum_{j=1}^{m} \lambda_i |\, y_i \,|^2 , \tag{4.8}$$

tendo em conta que

$$\sum_{j=1}^{m} |y_i|^2 = 1. \tag{4.9}$$

Logo $\mu = \lambda_1$ e $y = e_1 = [1 \quad 0 \quad \ldots \quad 0]^T$.

Como $u = U_C y = U_C \; e_1$, a direcção de máxima variação *u* é exactamente o vector próprio u_1 da matriz de covariância *C*, que diz respeito ao maior valor próprio λ_1. Desta forma λ_1 mede a variação na direcção de u_1.

Procurando a direcção de máxima variação ortogonal a u_1, temos de calcular (4.5), mas desta vez sujeito a (4.9) e a $e_1^T y = 0$. Isto implica que a primeira componente de *y* tem de ser nula. Então segue-se que (4.8) é calculado sujeito a $\sum_{j=2}^{m} |y_i|^2 = 1$. Isto leva-nos a $y = e_2 = [0 \quad 1 \quad 0 \quad \cdots \quad 0]^T$, ou seja, $u = U_C \, e_2 = u_2$, o vector próprio de *C* correspondente ao segundo maior valor próprio.

Continuando neste sentido concluímos que o primeiro vector próprio da matriz de covariância *C* aponta na direcção da variação máxima, e o valor próprio correspondente mede a variação nessa direcção, isto é, a variância nessa direcção. Os restantes vectores próprios apontam em direcções de máxima variação ortogonal às anteriores direcções e os valores próprios continuam a medir as variações.

Todavia, neste trabalho, utilizamos a função *svd.m* do MATLAB para calcular numericamente a decomposição em valores singulares.

4.3.2 – Determinação das faces próprias

Para o cálculo das faces próprias é necessário encontrar uma base ortonormal para o espaço gerado pelos x_j, ou seja, uma base para R(*X*). Esses vectores são depois utilizados para a reconstrução de qualquer face, quer a face pertença ou não ao conjunto de treino. A base é constituída pelas colunas da matriz *U*. No entanto, lembremos que esses vectores correspondem aos vectores próprios de XX^T; assim sendo basta calcular a decomposição em valores próprios da matriz de covariância *C*.

Encontrados os valores próprios, sabemos que os valores singulares se obtêm da seguinte forma:

$$\sigma_i = \sqrt{\lambda}_i, \;\; i = 1, \cdots, p$$

onde σ_i são os valores singulares e λ_i os valores próprios da matriz de covariância.

Desta forma tomamos os primeiros *k* vectores próprios associados a valores próprios não nulos e assim obtemos a base, cujos vectores constituintes são as faces próprias.

Um aspecto importante a focar no reconhecimento facial, nomeadamente no cálculo das faces próprias, é a média das faces que é subtraída a todas as faces.

Figura 4.3 - Média das faces do 'Conjunto de Treino 1'

A média pode não ser subtraída, o que pode trazer vantagens nalgumas situações e desvantagens noutras. A vantagem é que certos aspectos da análise das imagens poderão ser facilitados, uma vez que a média não tem de ser considerada à parte. No entanto, qualquer imagem é representada através de um vector constituído apenas por elementos não negativos (intensidade dos *pixeis*) e, de um modo geral, situa-se longe da origem. Subtraindo a média a todas as imagens significa retirar os elementos que são comuns a todas as faces e assim estamos a tratá-las de forma uniforme e com mais vantagens pois se a média representar um grande desvio da origem nós, ao subtrairmo-la, temos maior facilidade no tratamento dos dados. Mais ainda, subtraindo a média asseguramos que os desvios normalizados da média da face dados por:

$$x_j = \frac{f_j - a}{d}, \tag{4.10}$$

onde

$$d^2 = \frac{1}{n}\sum_{j=1}^{n}\left\| f_j - a \right\|_2^2 \tag{4.11}$$

são próximos de zero, que é uma propriedade muito útil para alguns algoritmos baseados em faces próprias,[13]. Outra vantagem da subtracção da média é que se compararmos as caricaturas das faces podemos observar a variação entre as faces e não das imagens gerais [14].

Neste trabalho para efectuarmos a reconstrução e o reconhecimento facial, utilizaremos a versão dos desvios normalizados.

Vejamos uma face do conjunto de treino e a mesma sem a média:

Fig. 4.4 – Imagem no 'Conjunto de Treino 1'

$\|Face\|_2 = 113.0951$

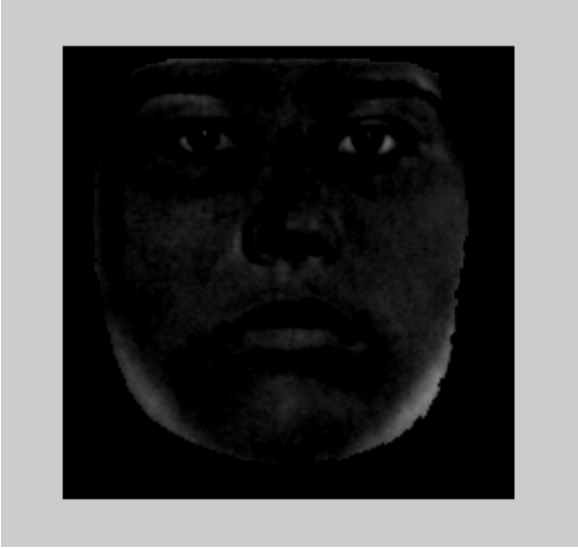


Fig. 4.5 – Imagem 4.4 sem a média

$\|Face-Média\|_2 = 23.9064$

Verificamos que quando se subtrai a média a uma face esta fica mais próxima da origem, como foi referido atrás.

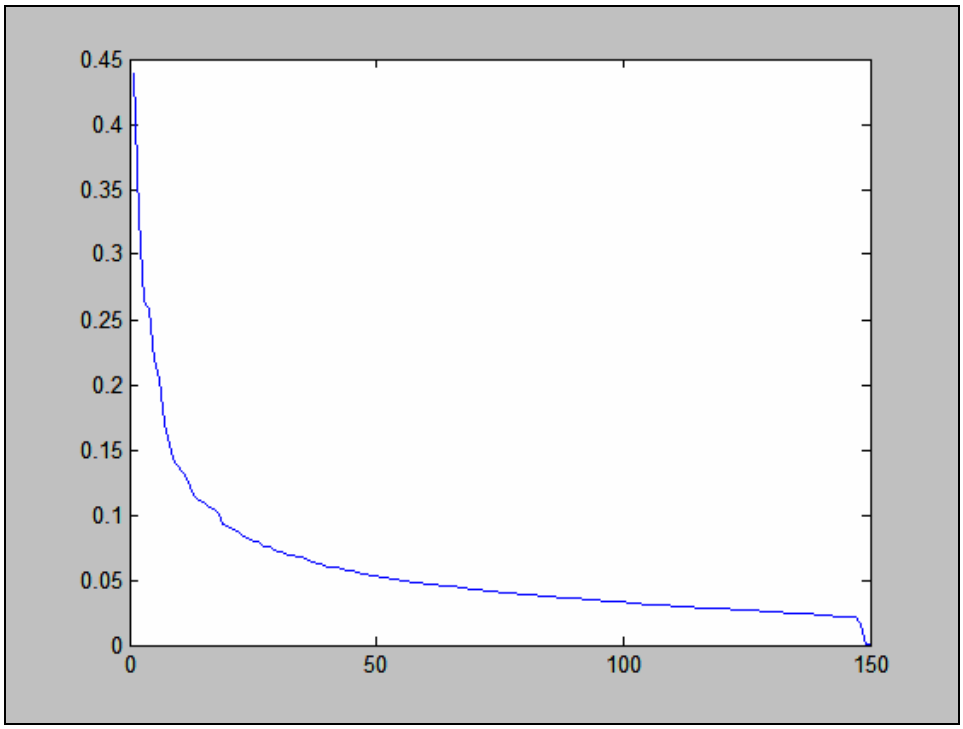


Gráfico 4.1 – Os valores singulares da matriz de faces do 'Conjunto de Treino 1', subtraindo a média

Nos gráficos 4.1 e 4.2 estão representados os valores singulares calculados para a matriz de faces do 'Conjunto de treino 1' subtraindo a média e sem a subtrair, respectivamente.

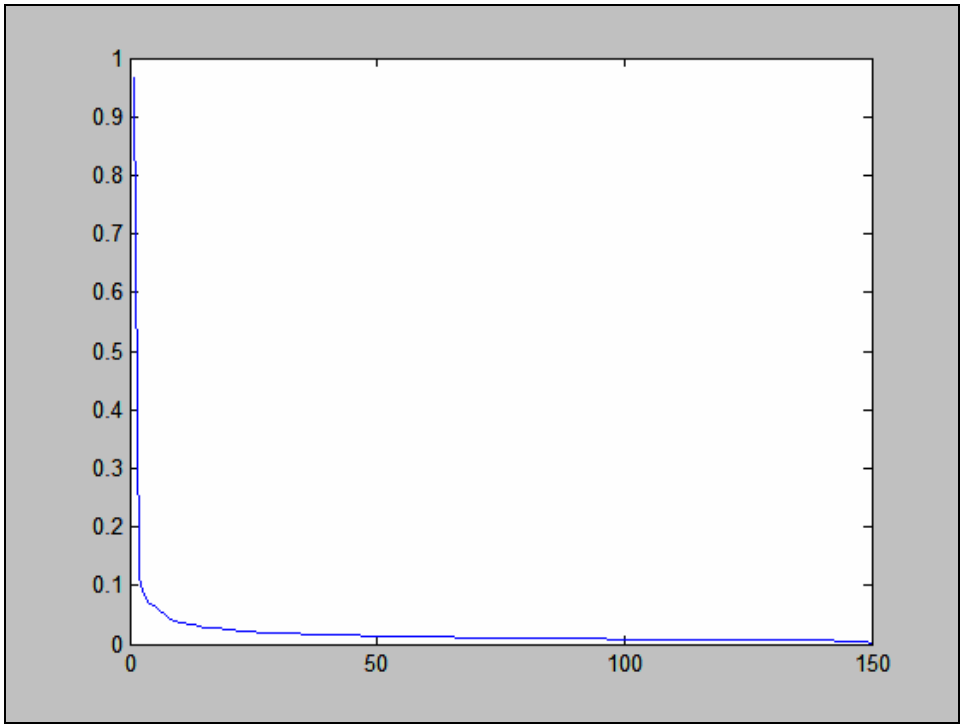


Gráfico 4.2 – Os valores singulares da matriz de faces 'Conjunto de Treino 1', sem subtrair a média

Para complementar esta análise, calculámos também o decréscimo de cada valor singular relativamente ao anterior, $\left(\frac{\sigma_{i+1}-\sigma_i}{\sigma_i}\right)$, e ainda relativamente ao primeiro valor singular, $\left(\frac{\sigma_1-\sigma_i}{\sigma_1}\right)$.

Verificamos que quando não se subtrai a média, o segundo valor singular, relativamente ao primeiro, decresce mais rapidamente. No entanto os valores singulares seguintes decrescem mais lentamente.

Nas figuras 4.6 a 4.11 podemos visualizar as faces próprias associadas aos valores singulares σ_1, σ_2, σ_{40}, σ_{100}, σ_{149} e σ_{150}.

Fig. 4.6 - 1ª face própria

Fig. 4.7 - 2ª face própria

Fig. 4.8 - 40ª face própria

Fig. 4.9 - 100ª face própria

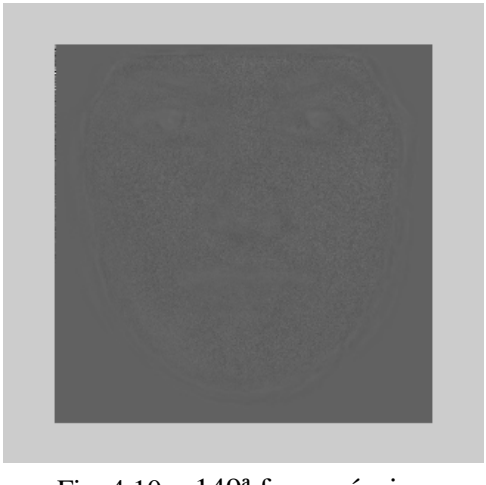


Fig. 4.10 - 149ª face própria

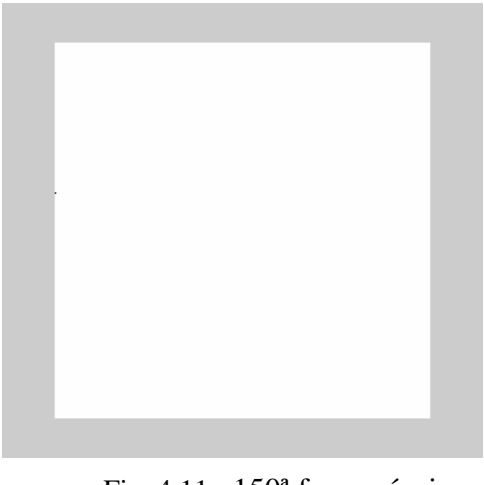


Fig. 4.11 - 150ª face própria

4.3.3 – Reconstrução facial

Uma aplicação possível das faces próprias é a reconstrução facial. Uma vez que um vector que representa uma imagem facial é projectado num espaço de dimensão inferior, então esse vector pode ser escrito como combinação linear dos vectores da nova base. Desta forma podemos reconstruir faces.

Assim, qualquer face pode ser representada exactamente em termos de uma combinação linear de faces próprias ou pode também ser aproximada usando apenas as "melhores" faces próprias, que são aquelas que correspondem aos maiores valores singulares e que são as que apresentam maior variação dentro do conjunto das imagens faciais.

De um modo geral, chama-se ***representação através de faces próprias*** à melhor aproximação de uma face em termos de faces próprias.

Dadas as faces próprias $u_1, u_2, \ldots, u_\nu$ e os coeficientes $y_1, y_2, \ldots, y_\nu$ pode obter-se a representação de uma face através da soma da média com uma combinação linear das faces próprias:

$$\tilde{f} = y_1 u_1 + y_2 u_2 + \ldots + y_\nu u_\nu + a \tag{4.12}$$

Vejamos um exemplo da reconstrução de uma face do conjunto de treino. Para efectuar essa reconstrução utilizaram-se quarenta faces próprias, isto é, $\nu = 40$:

$$f \approx \tilde{f} = \sum_{i=1}^{40} y_i u_i + a .$$

O resultado pode ser visualizado na página 46.

Dado o vector $y = [y_1 \ldots y_\nu]$ e a matriz $U_\nu = [u_1 \ldots u_\nu]$, então a relação (4.12) pode ser escrita na forma

$$\tilde{f} = U_\nu y + a . \tag{4.13}$$

Isto significa que precisamos apenas de utilizar ν faces próprias e os respectivos coeficientes para representar $\tilde{f}$.

Face Original

≈

Face Reconstruída

=

= - 0.3332

1ª face própria

+ 0.3201

2ª face própria

+ … - 0.0647

40ª face própria

+

Média das faces

No exemplo anterior $\nu = 40$; utilizámos menos de um terço de todas as faces próprias. Portanto esta representação é mais eficiente do que se tivéssemos utilizado todas as faces próprias, ou seja, se considerássemos a imagem original.
Segue-se então de (4.13) que

$$\tilde{f} - a = U_\nu y, \tag{4.14}$$

ou seja,

$$y = U_\nu{}^T \left(\tilde{f} - a\right). \tag{4.15}$$

Mas como Δf , o ***erro da representação***, é ortogonal a todos $u_1, \ldots, u_\nu$ e $\tilde{f} = f - \Delta f$

$$y = U_\nu{}^T \left(f - \Delta f - a\right)$$

$$= U_\nu{}^T \left(f - a\right) \tag{4.16}$$

Esta representação captura todas as características associadas a uma face *f*, e em vez de comparar as faces directamente, comparamos as características de *y*.

Uma vez que (4.16) é uma projecção ortogonal, perdemos informação, no entanto *f* é uma face do conjunto de treino, portanto $\|\Delta f\|_2$ é pequeno.

Verifica-se assim que aproximação (4.13) da representação de uma face minimiza

$$\varepsilon(f) = \left\| f - \tilde{f} \right\|_2 = \|\Delta f\|_2 . \tag{4.17}$$

O valor $\varepsilon(f)$ é conhecido como distância ao espaço de faces e utilizado no reconhecimento facial para classificação de imagens.

Foi elaborado um programa em MATLAB, para reconstrução de imagens, com base no código disponível em ftp://dip.sun.ac.za/, que vem mencionado em [1]. Pode ser consultado no Anexo B.

De seguida apresentam-se alguns dos testes de reconstrução facial efectuados com faces que pertencem ao conjunto de treino e outras que não pertencem. Realizámos também testes com partes de faces ocultas.

Para cada teste apresenta-se a imagem de teste e as imagens das reconstruções. Apresentamos também tabelas onde são apresentados a representação das faces de teste assim como as normas dois e de Frobenius do erro de cada reconstrução.

Teste 1: Reconstrução de uma face que pertence ao conjunto de treino

Utilizámos, 10, 20, 100 e 150 faces próprias para fazer as reconstruções. Observemos as diferentes imagens obtidas na figura 4.12.

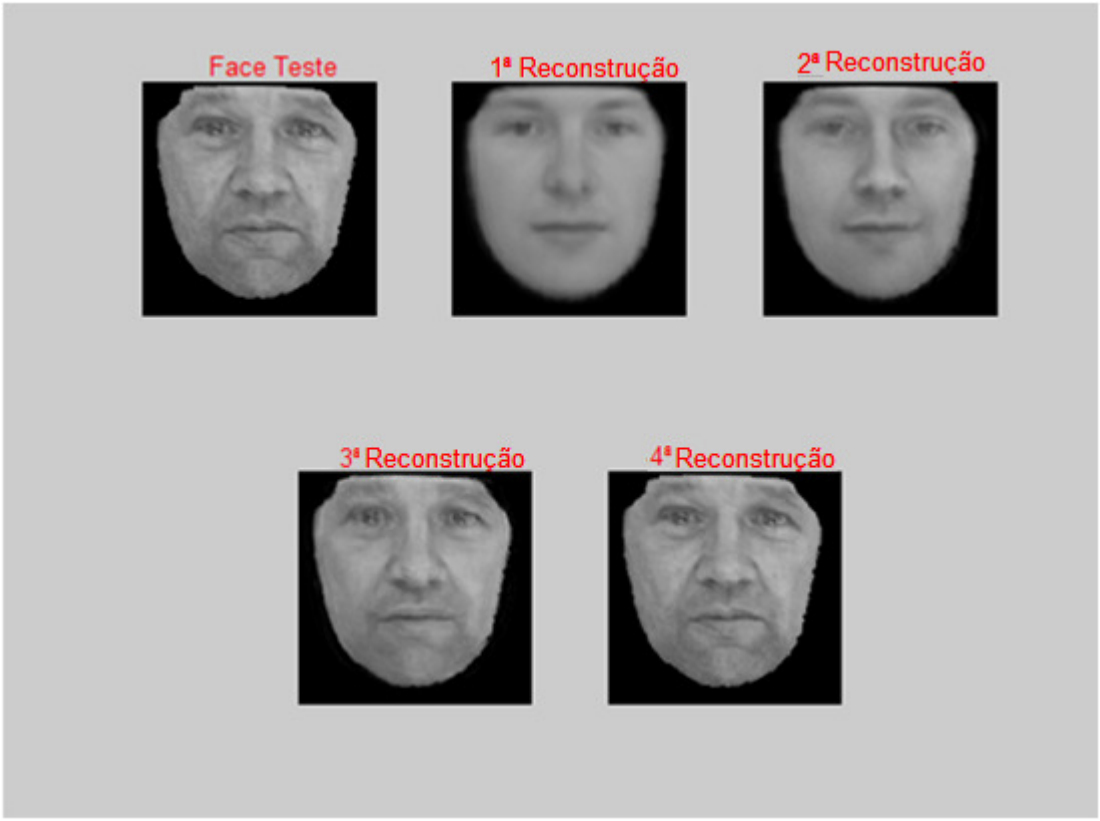


Fig. 4.12 – Reconstruções de uma face que pertence ao conjunto de treino

Nestas reconstruções observámos que à medida que vão sendo acrescentadas faces próprias o erro vai diminuindo, como se pode confirmar na tabela 4.1, e como esta face pertence ao conjunto de treino ela pode ser reconstruída sem erro.

	Representação	$\| \Delta f \|_2$	$\| \Delta f \|_{Frob}$
1ª Reconstrução	$f = \sum_{i=1}^{10} y_i u_i + a + \underbrace{\sum_{i=11}^{150} y_i u_i}_{\Delta f}$	7.6008	16.8678
2ª Reconstrução	$f = \sum_{i=1}^{20} y_i u_i + a + \underbrace{\sum_{i=21}^{150} y_i u_i}_{\Delta f}$	5.1330	13.6618
3ª Reconstrução	$f = \sum_{i=1}^{100} y_i u_i + a + \underbrace{\sum_{i=101}^{150} y_i u_i}_{\Delta f}$	1.8829	6.6085
4ª Reconstrução	$f = \sum_{i=1}^{150} y_i u_i + a$	0	0

Tabela 4.1 – Resultados das reconstruções da face que pertence ao conjunto de treino

Teste 2. Reconstrução de uma face que não pertence ao conjunto de treino e está normalizada

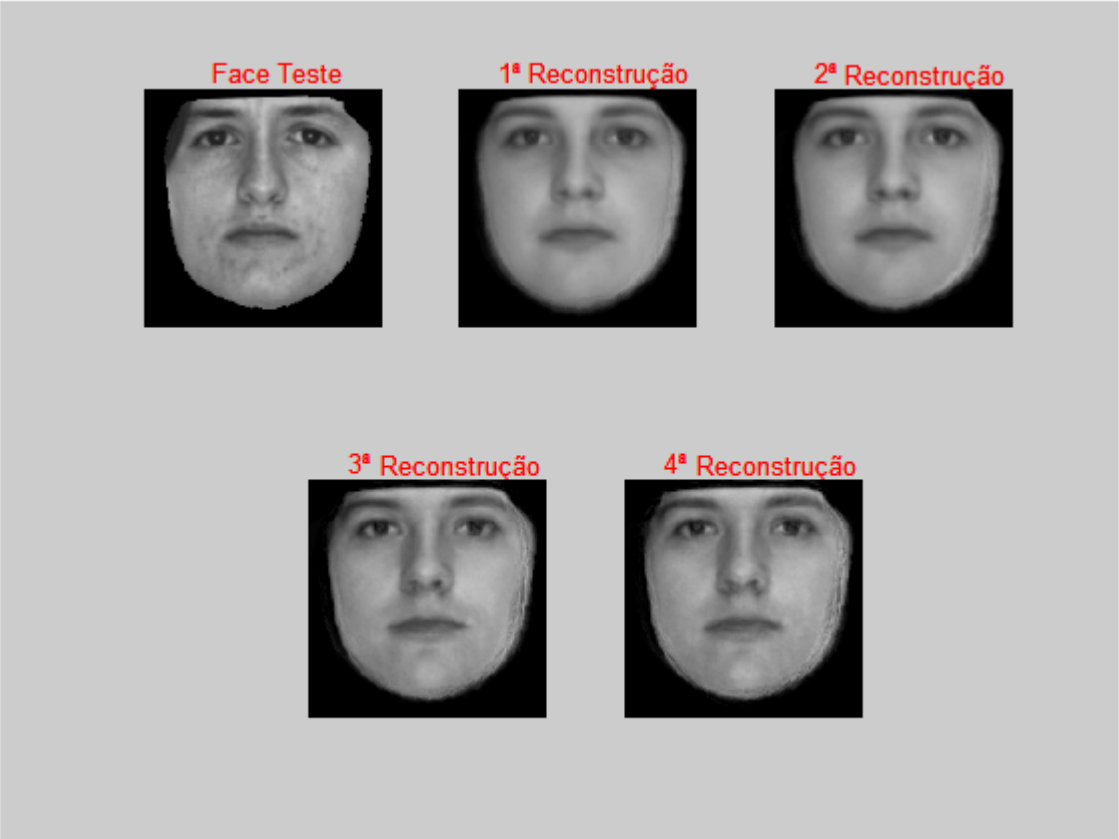


Fig. 4.13 – Reconstruções de uma face que não pertence ao conjunto de treino e está normalizada

Ao efectuarmos estas reconstruções verificámos que o erro vai diminuindo, no entanto nunca é próximo de zero, mesmo utilizando um grande número de faces próprias, como se pode confirmar na tabela 4.2.

	Representação	$\| \Delta f \|_2$	$\| \Delta f \|_{Frob}$
1ª Reconstrução	$f = \sum_{i=1}^{10} y_i u_i + a + \Delta_f$	6.8413	16.0899
2ª Reconstrução	$f = \sum_{i=1}^{20} y_i u_i + a + \Delta_f$	6.4736	15.0977
3ª Reconstrução	$f = \sum_{i=1}^{100} y_i u_i + a + \Delta_f$	4.8511	12.6293
4ª Reconstrução	$f = \sum_{i=1}^{150} y_i u_i + a + \Delta_f$	4.5367	12.1184

Tabela 4.2 – Resultados das reconstruções da face que não pertence ao conjunto de treino

O facto anteriormente mencionado deve-se ao facto desta imagem não pertencer ao conjunto de treino e portanto não estar descrita na sua totalidade pelas faces próprias.

Vejamos ainda outras reconstruções da mesma face. Numa primeira reconstrução omitimos a primeira face própria, considerámos as faces próprias da 2ª até à 20ª e na segunda reconstrução omitimos as duas primeiras faces próprias, ou seja considerámos as faces próprias da 3ª até à 20ª.

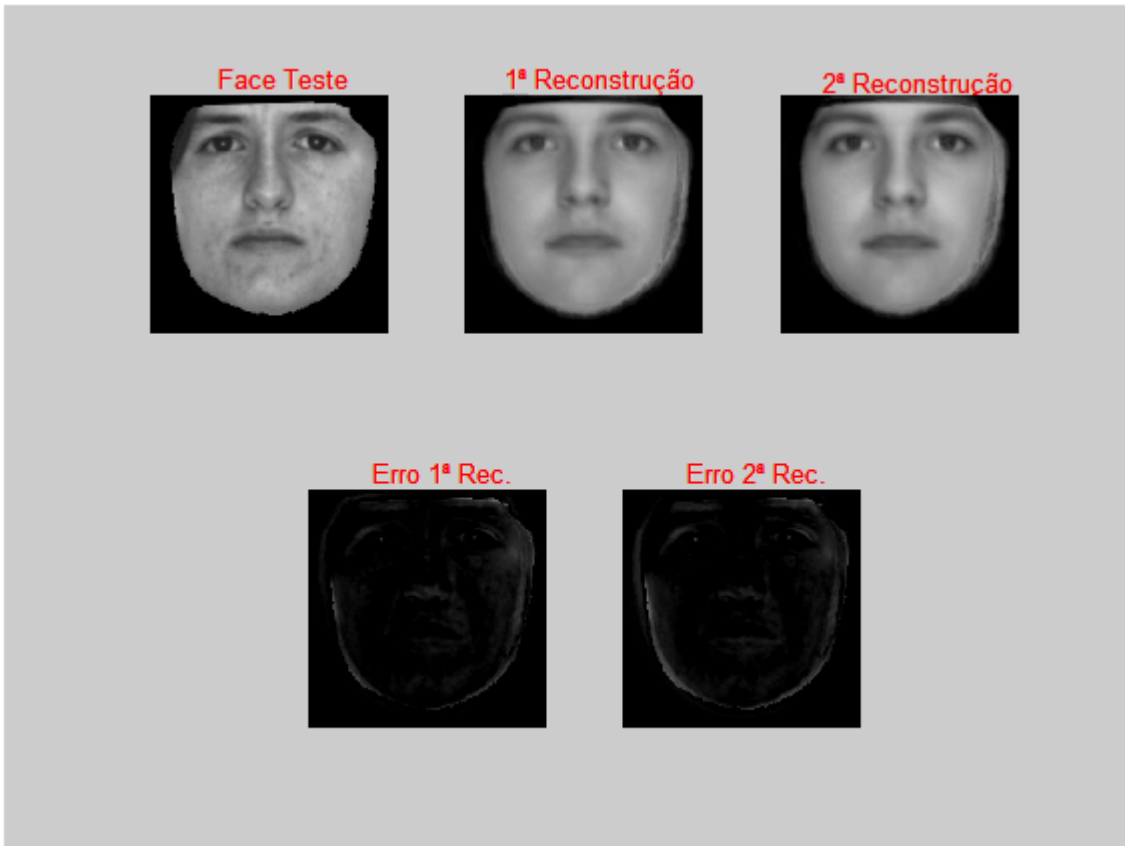


Fig. 4.14 – Reconstruções de uma face que não pertence ao conjunto de treino considerando apenas algumas faces próprias até à 20ª e respectivos erros

	Representação	$\| \Delta f \|_2$	$\| \Delta f \|_{Frob}$
1ª Reconstrução	$f = \sum_{i=2}^{20} y_i u_i + a + \Delta_f$	8.8309	17.3468
2ª Reconstrução	$f = \sum_{i=3}^{20} y_i u_i + a + \Delta_f$	10.0867	19.2319

Tabela 4.3 – Resultados das reconstruções da face que não pertence ao conjunto de treino utilizando apenas algumas faces próprias

Tendo em conta a figura 4.14 e os resultados da tabela 4.3, verificamos que em ambas as reconstruções que o erro aumenta relativamente à 2ª reconstrução efectuada anteriormente, em que foram utilizadas 20 faces próprias. Podemos assim verificar a importância das primeiras faces próprias na reconstrução de uma face.

Teste 3. Reconstrução de uma face que não pertence ao conjunto de treino e não está normalizada

Fig. 4.15 – Reconstruções de uma face que não pertence ao conjunto de treino e não está normalizada (utilizando 10 e 20 faces próprias) e respectivos erros

As reconstruções efectuadas têm uma fraca qualidade. No entanto se observarmos nas figuras 4.15 e 4.16 a imagem do erro de cada uma das reconstruções, nomeadamente a região da boca, verificamos que esta pode ser decisiva na identificação da face em causa.

Podemos concluir também que não se obtém uma boa reconstrução desta face, mesmo utilizando todas as faces próprias, por esta não estar normalizada.

Fig. 4.16 – Reconstruções de uma face que não pertence ao conjunto de treino e não está normalizada (utilizando 100 e 150 faces próprias) e respectivos erros

	Representação	$\| \Delta f \|_2$	$\| \Delta f \|_{Frob}$
1ª Reconstrução	$f = \sum_{i=1}^{10} y_i u_i + a + \Delta_f$	15.0859	28.0344
2ª Reconstrução	$f = \sum_{i=1}^{20} y_i u_i + a + \Delta_f$	13.2204	26.0820
3ª Reconstrução	$f = \sum_{i=1}^{100} y_i u_i + a + \Delta_f$	8.2352	20.8149
4ª Reconstrução	$f = \sum_{i=1}^{150} y_i u_i + a + \Delta_f$	7.5626	19.9709

Tabela 4.4 – Resultados das reconstruções da face que não pertence ao conjunto de treino utilizando apenas algumas faces próprias

Decidimos substituir duas faces no conjunto de treino por duas faces que não estão normalizadas. Nestas novas faces vêem-se os dentes como na face utilizada para realizar os testes.

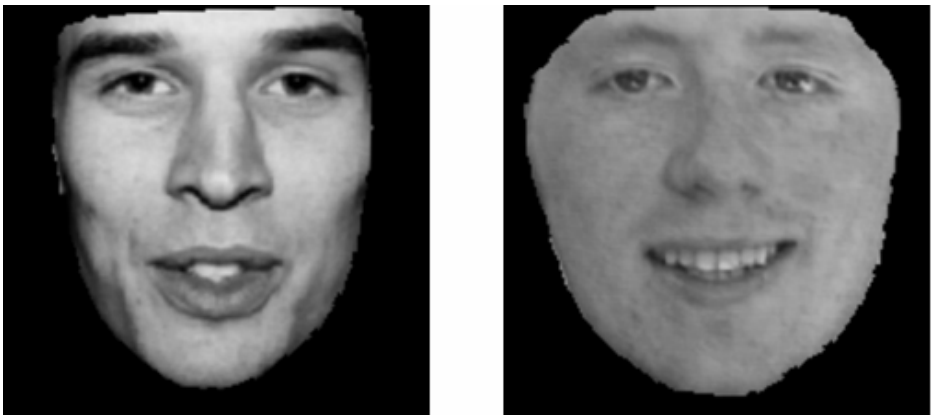


Fig. 4.17 - Imagens de faces não normalizadas colocadas no conjunto de treino

Efectuaram-se testes utilizando o mesmo número de faces próprias que se no caso anterior.

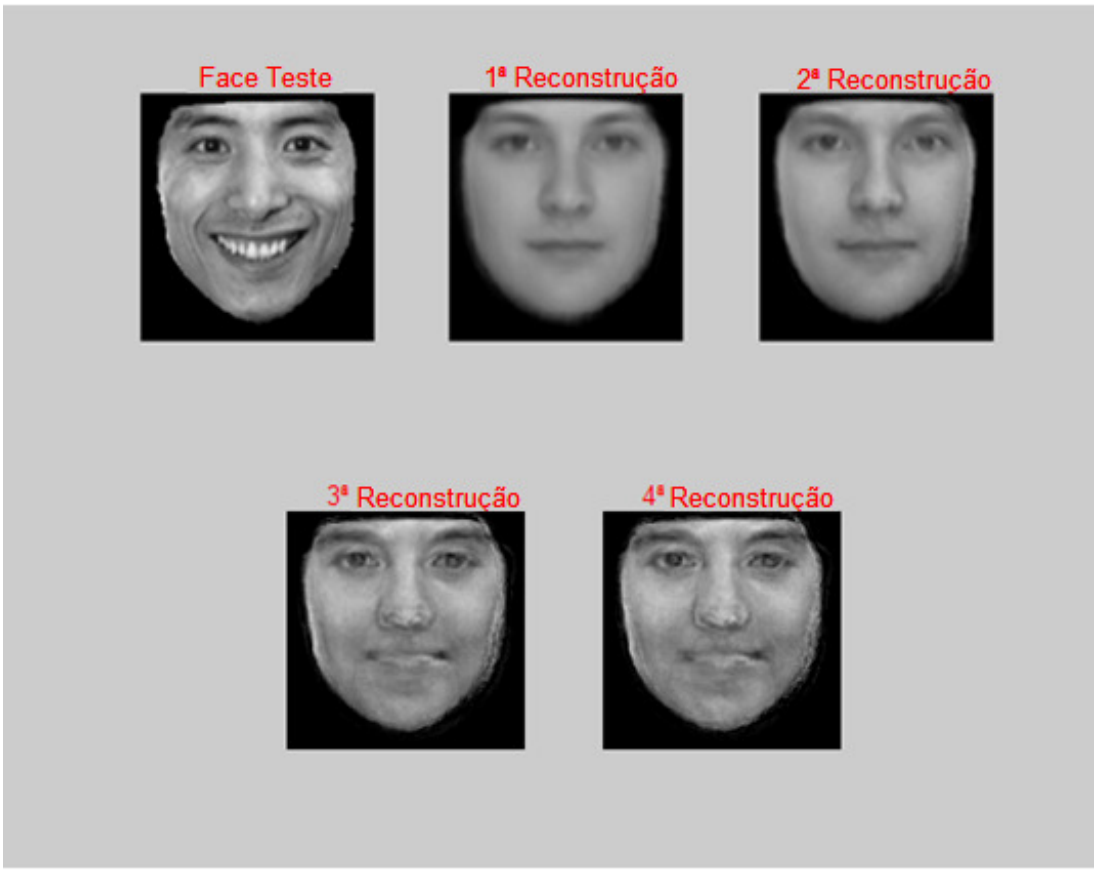


Fig. 4.18 – Reconstruções de uma face que não pertence ao conjunto de treino utilizando algumas das primeiras 20 faces próprias

	Representação	$\| \Delta f \|_2$	$\| \Delta f \|_{Frob}$
1ª Reconstrução	$f = \sum_{i=1}^{10} y_i u_i + a + \Delta_f$	15.1391	28.0628
2ª Reconstrução	$f = \sum_{i=1}^{20} y_i u_i + a + \Delta_f$	13.1839	26.0182
3ª Reconstrução	$f = \sum_{i=1}^{100} y_i u_i + a + \Delta_f$	8.0124	20.6712
4ª Reconstrução	$f = \sum_{i=1}^{150} y_i u_i + a + \Delta_f$	7.4539	19.8900

Tabela 4.5 – Resultados das reconstruções da face que não pertence ao conjunto de treino utilizando outro conjunto de treino

Em todas as reconstruções efectuadas verifica-se pela tabela 4.5 e pela imagem 4.18 que o erro praticamente não se altera, ou seja, a introdução de duas faces não normalizadas não foi o suficiente para que esta face fosse melhor descrita pelas faces próprias.

Constata-se assim que as faces próprias não sofreram alterações significativas de forma a reconstruir esta face com menor erro. Seria necessário fazer mais substituições de faces conjunto de treino por faces não normalizadas para obter melhores resultados.

Sejam Δ_{f1}, o erro da reconstrução apresentada sem utilizar as novas faces no conjunto de treino e Δ_{f2}, o erro da reconstrução apresentada utilizando as novas faces no conjunto de treino, $\tilde{f}_1 = \sum_{i=1}^{150} y_{1i} u_{1_i} + a_1$ a reconstrução efectuada utilizando o conjunto de treino habitual e $\tilde{f}_2 = \sum_{i=1}^{150} y_{2i} u_{2i} + a_2$ a reconstrução efectuada utilizando o novo conjunto de treino (isto para a utilização de todas as faces próprias). Podemos verificar pelas tabelas 4.4 e 4.5 que $\Delta_{f1} \approx \Delta_{f2}$. De certo que $a_1 \approx a_2$, o que implica que $\sum_{i=1}^{150} y_{1i} u_{1_i} \approx \sum_{i=1}^{150} y_{2i} u_{2i}$.

No âmbito da reconstrução facial, podemos pensar na reconstrução de faces com oclusão. Em diversas situações, as pessoas utilizam adereços como óculos escuros, lenços ou cachecóis, o que pode causar oclusões parciais do rosto.

De seguida apresentam-se alguns testes que foram efectuados.

Teste 4. Reconstrução de uma face que pertence ao conjunto de treino, com oclusão

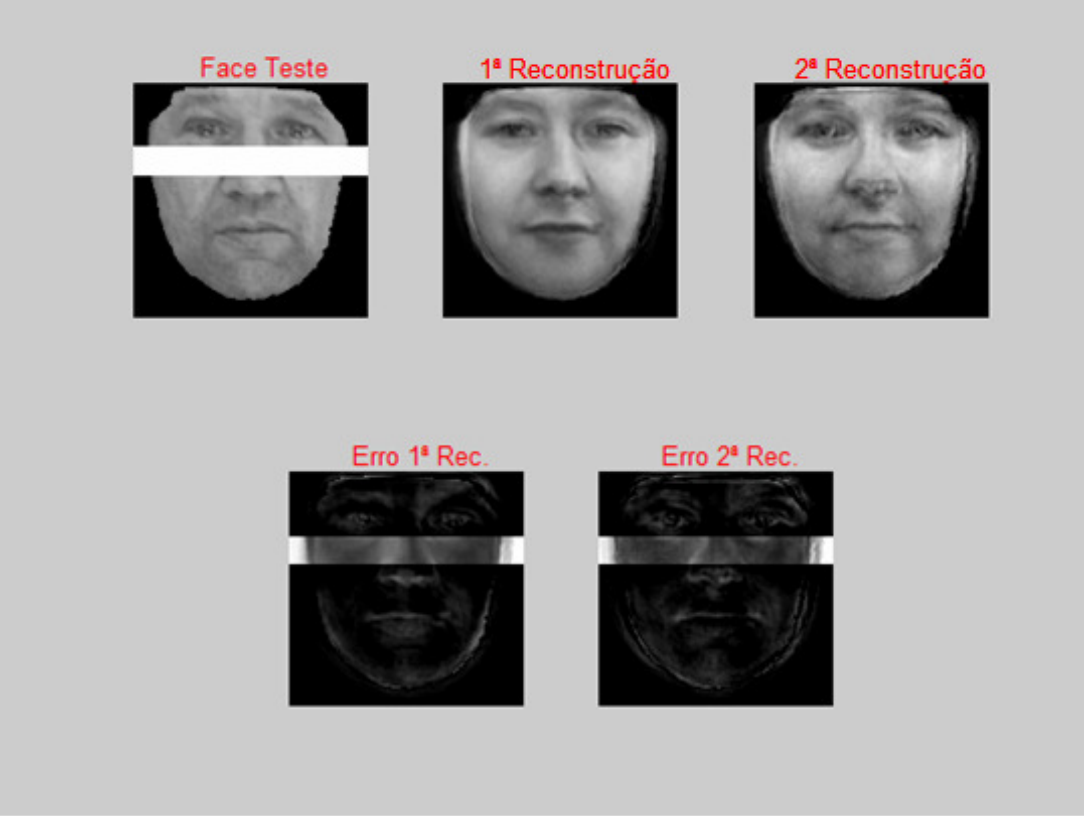


Fig. 4.19 – Imagens de reconstruções de uma face que pertence ao conjunto de treino, com oclusão (a branco) e respectivo erro

	Representação	$\| \Delta f \|_2$	$\| \Delta f \|_{Frob}$
1ª Reconstrução	$f = \sum_{i=1}^{20} y_i u_i + a + \Delta_f$	42.1637	47.2734
2ª Reconstrução	$f = \sum_{i=1}^{50} y_i u_i + a + \Delta_f$	38.3813	44.0895

Tabela 4.6 – Resultados das reconstruções da face que pertence ao conjunto de treino com oclusão (a branco)

Estas reconstruções podem ser comparadas com as reconstruções efectuadas no Teste 1, uma vez que se trata da mesma face. Verifica-se que o facto da face ter uma região oculta faz com que o erro aumente bastante.

Quando é feita a leitura destas imagens, com o programa MATLAB, cada *pixel* é substituído por um número real no intervalo [0, 1], sendo que zero corresponde à cor preta e um ao branco puro. Como tal o aumento do erro deve-se ao facto de um grande número de *pixeis* na imagem ser substituído pelo número um.

Comparando os resultados obtidos nas quatro reconstruções efectuadas, verificamos que para um maior número de faces próprias o erro diminui e o resultado obtido já é mais satisfatório.

Decidimos efectuar também testes com a mesma imagem mas com a oclusão pintada de preto.

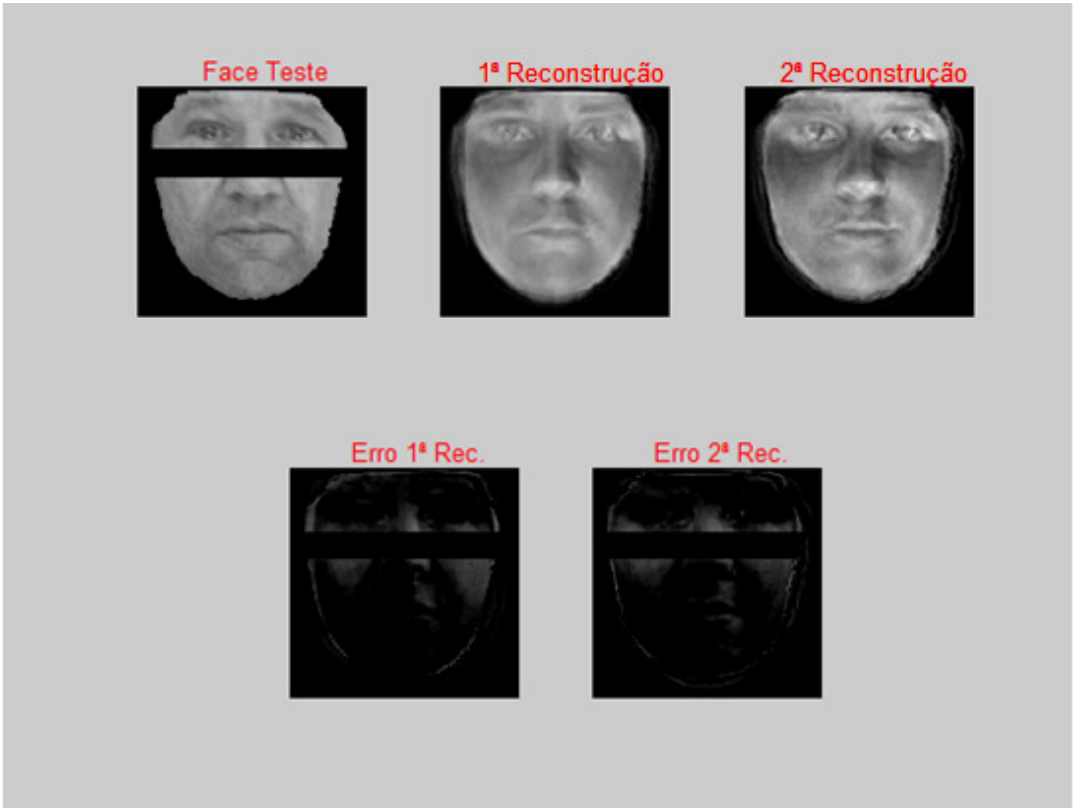


Fig. 4.20 – Imagens de reconstruções de uma face que pertence ao conjunto de treino, com oclusão (a preto) e respectivos erros

	Representação	$\| \Delta f \|_2$	$\| \Delta f \|_{Frob}$
1ª Reconstrução	$f = \sum_{i=1}^{20} y_i u_i + a + \Delta_f$	36.7578	42.9801
2ª Reconstrução	$f = \sum_{i=1}^{50} y_i u_i + a + \Delta_f$	33.4025	39.7346

Tabela 4.7 – Resultados das reconstruções da face que pertence ao conjunto de treino com oclusão (a preto)

Verificamos através da tabela 4.7 e figura 4.20 que há um maior decréscimo do erro (considerando qualquer um dos tipos de norma) para um grande número de faces próprias utilizadas.

Utilizando a oclusão pintada a preto o erro é inferior provavelmente devido à 'máscara'que também é preta utilizada em cada uma das imagens faciais e porque a cor preta é substituída pelo número zero quando é feita a leitura da imagem.

De um modo geral, podemos considerar que quando temos uma face com uma parte oculta podemos também fazer a sua reconstrução através das faces próprias.

4.3.4 - Reconhecimento facial

Tal como foi referido no início deste capítulo, para ser feito o reconhecimento facial de uma pessoa, é comparada uma imagem de entrada, aqui designada por imagem de teste, com um conjunto de imagens, o conjunto de treino. Podem ser vistos alguns trabalhos sobre o reconhecimento facial, por exemplo em [11, 13, 14, 15].

Para o sistema de reconhecimento facial não é requerida uma boa reconstrução. Apenas é necessário que as características se distingam entre diferentes indivíduos.

No reconhecimento facial é necessário fazer a localização da face numa imagem, o que não é uma tarefa simples pois muitos objectos têm a forma semelhante à de uma face e além disso as faces são objectos que não possuem formas bem definidas.

Na indústria por vezes é utilizado o reconhecimento de objectos, como a inspecção automática de uma peça numa linha de produção, mas as peças possuem uma posição e dimensões específicas e luminosidade controlada. Contrariamente, as faces podem apresentar-se com oclusão parcial, variação de posição, diferenças de iluminação (que até podem tornar certas características invisíveis ou deformadas pelo efeito de sombras), além de por vezes, as pessoas usarem óculos, chapéus, lenços, etc.

Devido à sua complexidade; são feitas algumas restrições e por exemplo podemos considerar a detecção de faces apenas quando estas aparecem na posição frontal.

Por vezes, uma razão para que sejam feitas estas restrições tem a ver com a aplicação a que o sistema se destina.

Há vários métodos para fazer a detecção e localização de faces numa imagem, entre eles podem destacar-se:

- **Métodos baseados em conhecimento humano e morfologia**

Estes métodos são desenvolvidos a partir do conhecimento humano sobre a caracterização de faces. Podemos por exemplo caracterizar uma face pelos dois olhos dispostos com simetria vertical, nariz e boca interiores a uma região aproximadamente elíptica. Pelo tamanho e disposição destas características, também é possível estimar o tamanho e posição relativa das outras características.

Alguns métodos foram desenvolvidos de forma a procurar características invariantes nas faces, tais como sobrancelhas, olhos, nariz e boca são facilmente destacados através da detecção de bordas. Depois de destacadas as características, são utilizados métodos estatísticos para verificar a existência de uma face. Os problemas que mais afectam esses métodos são variações na iluminação o que pode causar sombras e oclusão de características.

Pode também haver o problema das imagens das faces serem relativamente pequenas e nesse caso as características ficam representadas por um pequeno número de *píxeis* e normalmente não são reconhecidas pelo sistema.

- **Métodos baseados em comparação de padrões**

Neste método é construído um modelo de face padrão através de representações matemáticas apropriadas ou manualmente. Dada uma imagem de entrada são feitas comparações com a imagem padrão. Por exemplo, o contorno de uma face pode ser parametrizado pela equação de uma elipse ou por diversos segmentos de recta. Dada uma imagem de entrada, o valor de correlação com a imagem padrão é calculado para o contorno da face, olhos, boca e nariz, independentemente. A existência da face é baseada nos valores de correlação. Este método, no entanto, tem-se mostrado inadequado para a detecção facial principalmente quando há alterações de escala, posição e forma das faces.

- **Métodos baseados em aparência**

Neste tipo de método, os sistemas de detecção "aprendem" quais são as características das imagens de interesse através de um conjunto de imagens exemplo. Estes métodos normalmente utilizam métodos estatísticos e de aprendizagem (*machine learning*) para encontrar características de imagem face ou imagem não face. As

características aprendidas são armazenadas na forma de modelos de distribuição ou de funções. Deste modo, muitos métodos podem ser abordados de forma probabilística.

Outra abordagem é a utilização de uma função específica, isto é, que separa as classes de faces e não-faces. Por terem determinadas semelhanças, as faces encontram-se agrupadas numa região específica do espaço imagem. Este subespaço pode ser delimitado pela função referida anteriormente, normalmente baseada numa distância, construída para realizar a classificação em faces ou não-faces.

Após ter sido feita a localização da face são extraídas as suas características e passa-se à fase do reconhecimento.

As imagens faciais de diferentes indivíduos ocupam posições bastante diferentes no espaço de faces.

Verifica-se também que quando mais de uma imagem da face de um mesmo indivíduo é projectada no espaço de faces, as suas projecções agrupam-se numa pequena região desse espaço. Cada um desses agrupamentos pode ser visto como uma classe. A distância euclidiana entre um ponto qualquer, pertencente ao espaço de faces, e o centróide de uma classe é definida por distância no espaço de faces. Designaremos essa distância por $I(f)$.

Vejamos a representação simplificada do espaço de faces, para ilustrar os quatro resultados possíveis da projecção de uma imagem no espaço de faces.

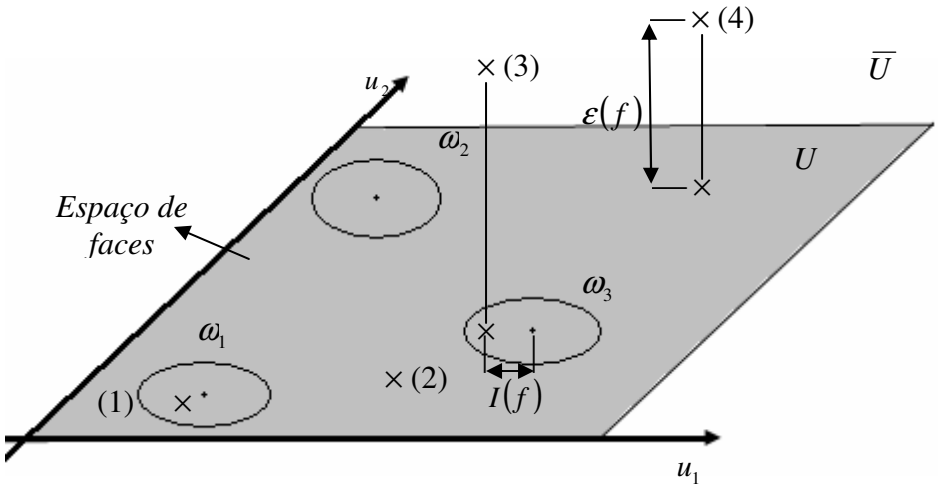


Fig.4.23 – Representação simplificada do espaço de faces

De um modo geral, existem quatro possibilidades para a posição de uma imagem facial qualquer em relação a um espaço de faces (*U*) — neste caso existem duas faces próprias principais (u_1 e u_2) e três classes de faces conhecidas (ω_1, ω_2 e ω_3). Como mostra a figura 4.21.

(1) Perto de espaço de faces e perto de uma classe

A face é detectada e reconhecida

(2) Perto do espaço de faces e distante de qualquer classe

A face é detectada mas a sua identidade não é reconhecida

(3) Fora (distante) do espaço de faces, com projecção no espaço de faces perto de uma das classes

A imagem não representa uma face

(4) Distante do espaço de faces, com projecção no espaço de faces distante de qualquer classe de faces conhecidas

A imagem não representa uma face

A distância entre uma imagem facial e o espaço de faces, $\varepsilon(f)$, deve ser pequena (menor que um determinado limiar). Além disso, uma imagem facial conhecida deve ter a sua projecção compreendida no espaço de faces e próxima da classe correspondente. Isto é, sua distância da respectiva classe no espaço de faces, $I(f)$, deve ser pequena (menor que outro limiar estabelecido). Ambos os limiares são calculados através de experiências.

Apresentam-se de seguida as principais fases no processo do reconhecimento facial:

1. Adquirir um conjunto de treino de imagens faciais e calcular as faces próprias que definem o espaço de faces;
2. Adquirir um conjunto de teste com algumas imagens faciais e outras, para testar o sistema. Projectar uma destas imagens no espaço de faces;
3. Decidir a classificação da imagem, de acordo com a distância ao espaço de faces $\varepsilon(f)$ e distância no espaço de faces $I(f)$.

Será importante analisar a distância das faces no espaço original de faces ao espaço de faces.

É de notar que as faces do 'Conjunto de treino 1' estão todas aproximadamente à mesma distância do espaço de faces por todas terem as mesmas características, ou seja por estarem normalizadas.

Fig. 4.22 – Face não normalizada

Número de faces próprias consideradas/ dimensão do espaço de faces	Distância (euclidiana) média das faces ao espaço de faces	Número de faces próprias consideradas/ dimensão do espaço de faces	Distância (euclidiana) média das faces ao espaço de faces
1	25.5506	50	10.0011
2	23.7209	60	8.9106
3	22.5455	70	7.9230
4	21.3686	80	6.9920
5	20.4404	90	6.0938
10	17.6224	100	5.2273
15	15.9006	110	4.3789
20	14.6046	120	3.5302
30	12.7139	130	2.5961
40	11.2264	140	1.4968

Tabela 4.8 – Distância média das faces do 'Conjunto de Treino1' ao espaço de faces

A face da figura 4.22 não pertence ao conjunto de treino nem está normalizada e está a uma distância de 22.8601, num espaço de faces de dimensão 50. Está portanto a cerca do dobro da distância das outras faces que estão normalizadas (ver tabela 4.8).

Verificamos que à medida que consideramos um espaço de faces de dimensão superior a distância ao espaço de faces diminui. Portanto o espaço de faces é o que mais se aproxima, ou seja, o que está mais próximo, no sentido da norma dois, de todas as faces no conjunto de treino.

Foi elaborado um programa em MATLAB para efectuar reconhecimento de faces. Neste programa para o reconhecimento facial considerámos o espaço de faces com dimensão 50. Os procedimentos são idênticos aos do programa para reconstrução facial, sendo depois necessária fazer a classificação da imagem testada.

Uma das dificuldades que surge frequentemente nos programas de reconhecimento, para a classificação de imagens, é o cálculo de limiares adequados para $\varepsilon(f)$ e para $I(f)$. No nosso sistema, para determinarmos o limiar para $\varepsilon(f)$, calculámos a distância euclideana entre cada uma das faces no conjunto de treino e a respectiva reconstrução. Considerámos como limiar para $\varepsilon(f)$, o maior desses valores. Para o limiar $I(f)$, calculámos a distância euclideana entre a projecção de cada face e todas as outras projecções das faces do conjunto de treino e considerámos o seu valor mínimo.

Para o Conjunto de Treino 1, obtivemos como limiar $\varepsilon(f)$, o valor 0.5333 e como limiar $I(f)$, o valor 0.7791.

Verificámos que para certas faces normalizadas que não pertencem ao conjunto de treino, o sistema reconhece a face. Isto deve-se ao facto desta ser parecida com as faces que pertencem ao conjunto de treino. Em todas as outras situações as imagens foram correctamente classificadas.

Na tabela 4.9 apresentam-se resultados de testes efectuados para uma face do conjunto de treino; para duas faces que não pertencem ao conjunto de treino, uma não normalizada e outra normalizada e também para uma imagem que não é uma face.

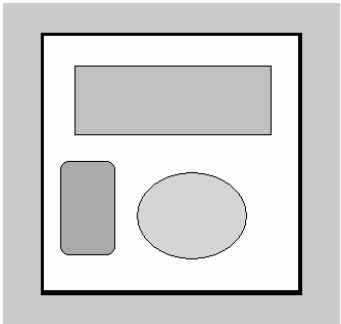
	Imagem de teste	Imagem reconstruída	$\varepsilon(f)$	$I(f)$	Resposta
Face que pertence ao conjunto de treino			0.0957	0	Face conhecida (correcto)
Face que não pertence ao conjunto de treino (normalizada)			0.1218	0.6791	Face conhecida (incorrecto)
Face que não pertence ao conjunto de treino (não normalizada)			0.2033	0.9239	Face desconhecida (correcto)
Imagem que não é uma face			0.6526	0.8495	Não é uma face (correcto)

Tabela 4.9 – Resultados de testes de reconhecimento para o 'Conjunto de Treino1'

Para o Conjunto de Treino 2, obtivemos como limiar $\varepsilon(f)$, o valor 0.3696 e como limiar $I(f)$, o valor 0.2492. Na tabela 4.10 apresentamos alguns resultados obtidos.

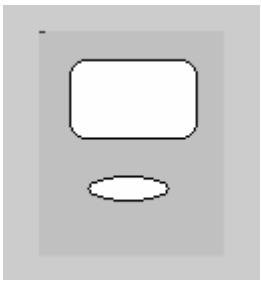
	Imagem de teste	**Imagem reconstruída**	$\varepsilon(f)$	$I(f)$	**Resposta**
Face que pertence ao conjunto de treino			0.0560	0	Face conhecida (correcto)
Face que não pertence ao conjunto de treino			0.2802	0.9277	Face desconhecida (correcto)
Imagem que não é uma face			0.2372	0.7753	Face desconhecida (incorrecto)

Tabela 4.10 – Resultados de testes de reconhecimento para o 'Conjunto de Treino 2'

Verificámos que as imagens do conjunto de treino, assim como as que não pertencem, são correctamente classificadas. No entanto, experiências com imagens que não são faces não foram correctamente classificadas. Isto pode dever-se ao facto de no conjunto de treino estarem imagens de faces em diferentes posições e com adereços, o que pode dar uma certa 'liberdade' no cálculo do limiar $\varepsilon(f)$.

Capítulo 5

Recuperação da estrutura tridimensional

5.1 – Introdução

Na área da análise de movimento, a determinação da estrutura tridimensional de objectos em movimento, por exemplo a partir de imagens de vídeo, tem sido muito estudada. Este problema é usualmente denominado por ***structure from motion*** (*SFM*).

O *SFM* tem sido objecto de maior estudo nas últimas duas décadas. A sua análise iniciou-se com Ullman [16] em 1979, mas só em 1992 foi introduzido por Tomasi e Kanade [17] o método da factorização, que resolve o problema de uma forma simples, fiável e robusta. Este método tem como base um tipo de projecção, designada por projecção ortográfica, sendo depois alargado a outros métodos mais gerais em [18]. Em [19], aplica-se o *SFM* para a segmentação a partir do movimento, permitindo a recuperação da estrutura 3D de vários objectos.

Pretende-se neste capítulo estudar a recuperação da estrutura tridimensional de um corpo rígido a partir de vídeo. Para isso, estudamos as várias fases do seu processamento. Começamos por estimar o movimento bidimensional da projecção do objecto no plano da imagem, fazendo a selecção e seguimento dos pontos característicos.

Posteriormente, estimamos a estrutura tridimensional através da factorização da matriz constituída pelas trajectórias dos pontos seguidos.

5.2 – Projecção ortográfica

Nesta secção analisamos como é que os objectos filmados são projectados no plano da imagem.

Para fazer este estudo consideraremos que é utilizada a projecção ortográfica. Neste tipo de projecção, os objectos são projectados no filme, paralelos ao eixo óptico, eixo Oz. Qualquer projecção é perpendicular ao plano da imagem.

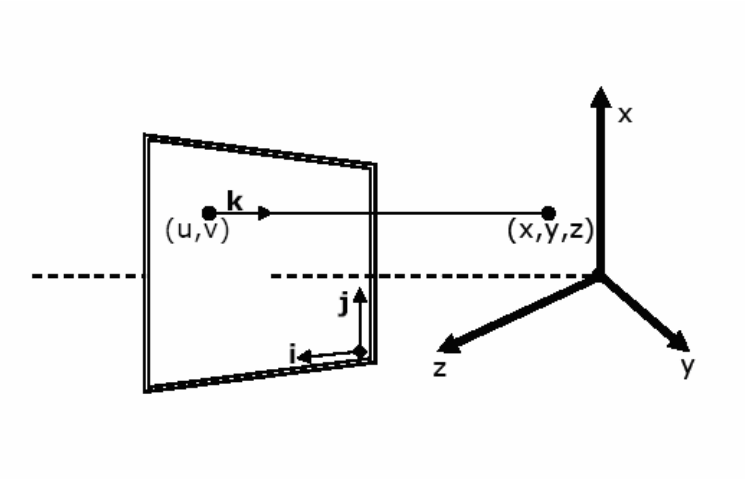


Fig. 5.1 Modelo da projecção ortográfica

Este tipo de projecção é utilizado em desenhos de engenharia e arquitectura para representar um objecto mantendo as proporções relativas do objecto. A aparência deste objecto pode ser reconstruída a partir de observações de diferentes ângulos.

Na projecção ortográfica, os pontos de uma imagem são projectados no plano da imagem ao longo de linhas paralelas. As linhas paralelas da imagem tridimensional são projectadas como linhas paralelas na imagem, não existindo deformação dos objectos. Seleccionando posições diferentes para se observar uma imagem, pode obter-se diferentes vistas bidimensionais dos objectos contidos nesta imagem.

5.3 – Selecção e seguimento dos pontos característicos

Cada imagem do filme é composta por pontos, os chamados *pixeis*. Alguns destes pontos são considerados ***característicos*** quando são passíveis de serem seguidos na sequência das imagens.

Quando pretendemos recuperar a estrutura tridimensional de um objecto num filme, quanto mais imagens do objecto observarmos e de diferentes ângulos, maior será a informação recolhida e mais facilmente se obtém uma aproximação da estrutura pretendida.

A escolha dos pontos característicos deve ser feita de uma forma correcta, pois são importantes para a descrição do movimento devendo ser possíveis de identificá-los nas imagens posteriores. (Normalmente escolhem-se pontos da imagem como cantos, sinais ou marcas). Depois de ser feita a selecção dos pontos característicos do objecto é feito o seu seguimento nas sucessivas imagens. Assim, os pontos característicos são seleccionados em todas as imagens, sendo de seguida feita a correspondência entre os novos e os antigos.

5.4 – Recuperação da estrutura 3D

Nesta secção abordar-se-á a recuperação da estrutura tridimensional de um objecto utilizando factorização matricial.

No trabalho original de Tomasi Kanade [17] a formulação do problema considera apenas o movimento de rotação, não tendo em conta o movimento de translação. Nesse trabalho a translação foi eliminada porque quando é utilizada projecção ortográfica ela não dá informação adicional acerca da forma do objecto em estudo. No entanto se o objectivo for recuperar a estrutura tridimensional de múltiplos objectos que se movem independentemente, esta já é indispensável.

Neste trabalho, também será considerado o movimento de translação.

Para fazer este estudo, assumimos que uma câmara fixa filma um único objecto em movimento. Na representação desta situação são necessários dois sistemas de coordenadas, um no objecto e outro na câmara, como mostra a seguinte figura:

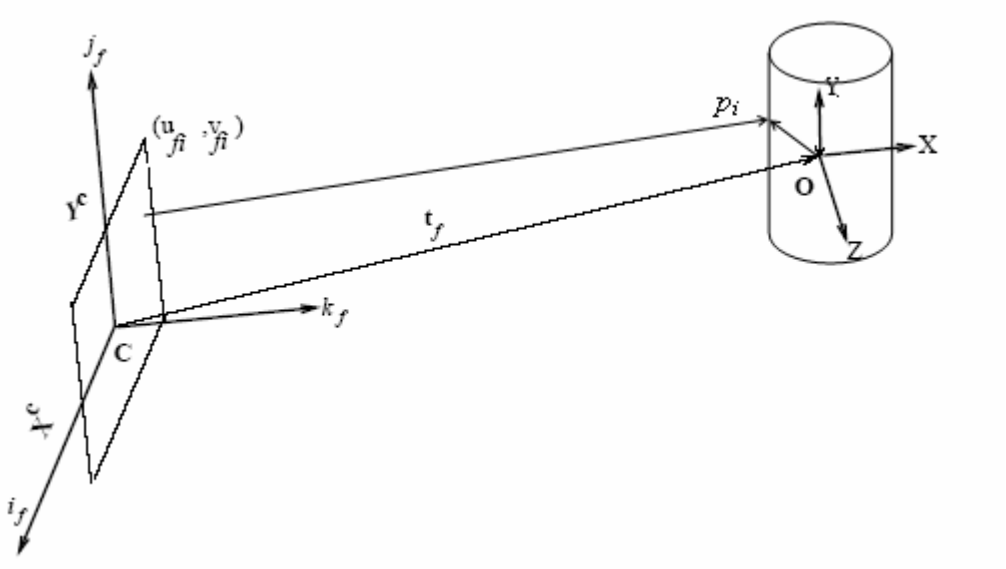


Fig. 5.2 – A câmara e o objecto com os seus sistemas de coordenadas. (Os vectores *i* e *j* definem o plano da imagem)

Escolhemos no objecto um sistema de coordenadas onde se define a origem de um referencial tridimensional.

Cada ponto escreve-se como

$$p_i = \begin{bmatrix} x_i \\ y_i \\ z_i \end{bmatrix}, \quad i = 1,\ldots,n. \tag{5.1}$$

Considerando *n* pontos do objecto obtém-se uma matriz, a chamada ***matriz de forma***, que reúne as coordenadas dos pontos característicos:

$$S = [p_1 \quad p_2 \ldots p_n]. \tag{5.2}$$

Evidentemente que os pontos característicos de um corpo rígido não mudam relativamente ao sistema de coordenadas fixo ao objecto. Mas se o objecto roda com respeito ao sistema de coordenadas fixo à câmara, cada ponto característico é projectado, no plano do filme, nas diferentes posições, em consecutivas imagens da sequência do vídeo.

Consideremos um dos pontos característicos, p_i, do objecto, representado no sistema de coordenadas fixo ao objecto, como é dado em (5.1). Quando o objecto se movimenta, a rotação relativamente ao sistema de coordenadas da câmara, no tempo *t*, na imagem do vídeo, é dada pela matriz de rotação:

$$R_t = \begin{bmatrix} i_t^T \\ j_t^T \\ k_t^T \end{bmatrix}, \text{ onde } \begin{bmatrix} i_t^T \\ j_t^T \\ k_t^T \end{bmatrix} = \begin{bmatrix} i_{xt} & i_{yt} & i_{zt} \\ j_{xt} & j_{yt} & j_{zt} \\ k_{xt} & k_{yt} & k_{zt} \end{bmatrix} \tag{5.3}$$

e a translação dada pelo vector:

$$T_t = \begin{bmatrix} T_{xt} \\ T_{yt} \\ T_{zt} \end{bmatrix}. \tag{5.4}$$

Então p_i é dado no sistema de coordenadas da câmara por

$$p_{ti}{}^c = R_t\, p_i + T_t\,. \tag{5.5}$$

Esta representação pode ser simplificada se forem usadas coordenadas homogéneas[5].

$$s = \begin{bmatrix} p \\ 1 \end{bmatrix} = \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}. \tag{5.6}$$

A equação (5.5) pode ser escrita da seguinte forma:

$$s_{ti}^c = \begin{bmatrix} p_{ti}^c \\ 1 \end{bmatrix} = \begin{bmatrix} R_t & T_t \\ 0_{1\times3} & 1 \end{bmatrix} \begin{bmatrix} p_i \\ 1 \end{bmatrix} = \begin{bmatrix} R_t & T_t \\ 0_{1\times3} & 1 \end{bmatrix} s_i\,, \tag{5.7}$$

---

[5] As coordenadas homogéneas (x_1, x_2, x_3, x_4) de um ponto (x, y, z) no espaço tridimensional, são quaisquer números que satisfaçam: $\frac{x_1}{x_4} = x$, $\frac{x_2}{x_4} = y$ e $\frac{x_3}{x_4} = z$. Por uma questão de comodidade consideramos $x_4 = 1$.

para cada ponto característico do objecto.

Verifica-se que um ponto

$$P_{ti}^{c} = \left[X_{ti}, Y_{ti}, Z_{ti}\right]^{T} \tag{5.8}$$

na câmara, é projectado num ponto cujas coordenadas correspondem aos dois primeiros elementos de P_{ti}^{c}, ou seja,

$$P_{ti} = \left[X_{ti}, Y_{ti}\right]^{T}, \tag{5.9}$$

devido ao facto de se utilizar a projecção ortográfica.

Pode escrever-se:

$$P_{ti} = \begin{bmatrix} X_{ti} \\ Y_{ti} \end{bmatrix} = \left[\begin{array}{ccc|c} i_{xt} & i_{yt} & i_{zt} & t_{xt} \\ j_{xt} & j_{yt} & j_{zt} & t_{yt} \end{array}\right], \tag{5.10}$$

ou seja,

$$P_{ti} = O_Z P_{ti}^{c} = O_Z \left[R_t \middle| T_t\right] p_i \tag{5.11}$$

onde O_Z é a matriz de projecção ortográfica;

$$O_Z = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}. \tag{5.12}$$

Procedendo desta forma para todos os *n* pontos, obtém-se uma matriz:

$$W_t = \begin{bmatrix} X_{t1} & X_{t2} & \cdots & X_{tn} \\ Y_{t1} & Y_{t2} & \cdots & Y_{tn} \end{bmatrix}. \tag{5.13}$$

Por sua vez, considerando uma sequência de f imagens, obtém-se:

$$W = \begin{bmatrix} X_{11} & X_{12} & \cdots & X_{1n} \\ & \vdots & \vdots & \\ X_{f1} & X_{f2} & \cdots & X_{fn} \\ Y_{11} & Y_{12} & \cdots & Y_{1n} \\ & \vdots & \vdots & \\ Y_{f1} & Y_{f2} & \cdots & Y_{fn} \end{bmatrix}, \tag{5.14}$$

a chamada ***matriz de observação***.

Cada linha tem as coordenadas da abcissa ou ordenada de cada ponto característico em cada imagem e cada coluna representa a trajectória de um ponto em toda a sequência de imagens.

Verificamos que a matriz de observação pode ser obtida pelo produto das matrizes de movimento e de forma.

$$W = \begin{bmatrix} X_{11} & X_{12} & \cdots & X_{1n} \\ & \vdots & \vdots & \\ X_{f1} & X_{f2} & \cdots & X_{fn} \\ Y_{11} & Y_{12} & \cdots & Y_{1n} \\ & \vdots & \vdots & \\ Y_{f1} & Y_{f2} & \cdots & Y_{fn} \end{bmatrix} = \left[\begin{array}{ccc|c} i_{x1} & i_{y1} & i_{z1} & t_{x1} \\ & \vdots & & \vdots \\ i_{xf} & i_{yf} & i_{zf} & t_{xf} \\ j_{x1} & j_{y1} & j_{z1} & t_{y1} \\ & \vdots & & \vdots \\ j_{xf} & j_{yf} & j_{zf} & t_{yf} \end{array}\right] \begin{bmatrix} s_{x1} & & s_{xn} \\ s_{y1} & \cdots & s_{yn} \\ s_{z1} & & s_{zn} \\ 1 & & 1 \end{bmatrix}. \qquad (5.15)$$

Considerando

$$M = \left[\begin{array}{ccc|c} i_{x1} & i_{y1} & i_{z1} & t_{x1} \\ & \vdots & & \vdots \\ i_{xf} & i_{yf} & i_{zf} & t_{xf} \\ j_{x1} & j_{y1} & j_{z1} & t_{y1} \\ & \vdots & & \vdots \\ j_{xf} & j_{yf} & j_{zf} & t_{yf} \end{array}\right] \quad \text{e} \quad S = \begin{bmatrix} s_{x1} & & s_{xn} \\ s_{y1} & \cdots & s_{yn} \\ s_{z1} & & s_{zn} \\ 1 & & 1 \end{bmatrix} \qquad (5.16)$$

como a ***matriz de movimento*** e a ***matriz de forma*** respectivamente, tem-se:

$$W = M\,S. \qquad (5.17)$$

Pode assim constatar-se que, dada a matriz W, o problema é encontrar M e S, ou seja, reduz-se à factorização da matriz de observação na matriz de movimento e na matriz de forma.

A característica máxima dessas matrizes é 4, basta observar (5.15), logo W tem também, no máximo, característica 4.

Nesta etapa da reconstrução do objecto 3D utiliza-se a decomposição em valores singulares para calcular a característica efectiva de W.

Viu-se atrás que é necessário factorizar a matriz de observação *W* nas matrizes de movimento $M_{2f\times 4}$ e de forma $S_{4\times n}$. Para isso calcula-se a decomposição em valores singulares da matriz de observação:

$$W = U\Sigma V^T \tag{5.18}$$

Uma vez que $W \in \mathbb{R}^{2f\times n}$, a sua característica é normalmente superior a 4, pois naturalmente, $2f \gg 4$ e $n \gg 4$. Assim, escolhem-se os quatro maiores valores singulares e forma-se uma matriz $\tilde{W}$, aproximação de *W*, com característica 4,

$$\tilde{W} = U_+\Sigma_+ V_+^T, \tag{5.19}$$

onde

$$\Sigma_+ = \text{diag}\ (\sigma_1, \sigma_2, \sigma_3, \sigma_4)\,. \tag{5.20}$$

As matrizes $U_+ \in \mathbb{R}^{2f\times 4}$ e $V_+ \in \mathbb{R}^{n\times 4}$ correspondem às primeiras quatro colunas de *U* e *V* respectivamente.

A expressão (5.19) é a melhor aproximação da matriz *W*, por uma matriz com característica quatro, no sentido da norma dois. Desta forma, pode escrever-se:

$$\tilde{W} = \tilde{M}\ \tilde{S}\,, \tag{5.21}$$

onde

$$\tilde{M} = U_+\Sigma_+^{\frac{1}{2}} \quad \text{e} \quad \tilde{S} = \Sigma_+^{\frac{1}{2}} V_+^T\,. \tag{5.22}$$

Verifica-se no entanto, que esta factorização não é única. Seja $A_{4\times 4}$ uma matriz invertível. Então:

$$\tilde{W} = \left(\tilde{M}A\right)\left(A^{-1}\tilde{S}\right) \tag{5.23}$$

é outra factorização possível. Isto deve-se ao facto das matrizes de movimento e de forma serem encontradas a menos de uma transformação afim. Temos de encontrar uma matriz *A* de modo que

$$M = \tilde{M}\ A\,, \tag{5.24}$$

possua todas as propriedades da matriz de movimento.

Podemos considerar

$$A = \left[A_R \middle| a_t \right], \tag{5.25}$$

onde A_R é uma submatriz de dimensão 4×3 e corresponde à parte rotacional e a_t é um vector 4×1 que diz respeito à translação. Tem-se assim:

$$M = \tilde{M}\ A = \left[\tilde{M}A_R \middle| \tilde{M}a_t \right] . \tag{5.26}$$

As equações que se escrevem a seguir surgem do facto de $\tilde{M}A_R$ ser uma matriz de rotação e como tal as suas linhas e colunas são ortogonais e normalizadas.
Para se obter A_R, que é a parte rotacional do movimento é necessário que:

$$\tilde{m}_i^T\ A_R A_R^T\, \tilde{m}_i = 1 \tag{5.27}$$

$$\tilde{m}_j^T\ A_R A_R^T\, \tilde{m}_j = 1 \tag{5.28}$$

$$\tilde{m}_i^T\ A_R A_R^T\, \tilde{m}_j = 0 \tag{5.29}$$

para $i = 1, \ldots, f$ e $j = f+1, \ldots, 2f$, onde $\tilde{m}_i^T$ e $\tilde{m}_j^T$ são as linhas *i* e *j* da matriz $\tilde{M}$.

Podemos verificar que o sistema (5.27) – (5.29) é dado por (5.30). Para mais detalhes pode consultar-se o Anexo A.

$$B_{3f\times 10}\ a_{10\times 1} = C_{3f\times 1} \tag{5.30}$$

e que é um sistema sobredeterminado, por possuir um maior número de equações do que incógnitas. Como tal procura-se uma solução aproximada, no sentido dos mínimos quadrados,

$$a = B^{\dagger} C . \tag{5.31}$$

Desta forma encontramos as entradas da matriz $A_R A_R^T$. Seguidamente factorizamos esta matriz na forma: $A_R A_R^T = U \Sigma U^T$, por ser uma matriz simétrica.

Calculamos assim $A_R = U\, \Sigma^{\frac{1}{2}}$.

Vejamos de seguida como encontrar a_t, ou seja, o movimento de translação.

Quando se utiliza projecção ortográfica, o centróide dos pontos característicos do objecto 3D coincide com o centróide da projecção dos pontos característicos. Este ponto pode ser escrito na forma

$$\overline{w} = \begin{bmatrix} \frac{1}{n}\sum_{s=1}^{n} X_{s1} \\ \frac{1}{n}\sum_{s=1}^{n} Y_{s1} \\ \vdots \\ \frac{1}{n}\sum_{s=1}^{n} X_{sf} \\ \frac{1}{n}\sum_{s=1}^{n} Y_{sf} \end{bmatrix} = \left[\tilde{M}A_r \middle| \tilde{M}a_t\right] \begin{bmatrix} \overline{p} \\ 1 \end{bmatrix}, \tag{5.32}$$

onde

$$\overline{p} = \frac{1}{n}\sum_{s=1}^{n} p_s \ , \tag{5.33}$$

é o centróide do objecto.

Como a origem do sistema de coordenadas no objecto é arbitrária, pode escolher-se $\overline{p} = 0$. Consequentemente, de (5.32) obtém-se:

$$\overline{w}_{2f\times 1} = \tilde{M}_{2f\times 4}\ a_{t\,4\times 1}\,. \tag{5.34}$$

Este sistema de equações também é sobredeteminado, então resolve-se no sentido dos mínimos quadrados.

$$a_t = \left(\tilde{M}\right)^{\dagger} \overline{w} \tag{5.35}$$

$$= \left(\tilde{M}^T \tilde{M}\right)^{-1} \tilde{M}^T\ \overline{w} \tag{5.36}$$

$$= \left(\Sigma^{\frac{1}{2}} U^T U \Sigma^{\frac{1}{2}}\right)^{-1} \Sigma^{\frac{1}{2}} U^T\ \overline{w} \tag{5.37}$$

$$= \Sigma^{-\frac{1}{2}} U^T\ \overline{w} \tag{5.38}$$

Encontramos assim todas as entradas da matriz A e consequentemente $M = U_+ \Sigma_+^{\frac{1}{2}} A$ e $S = A^{-1} \Sigma_+^{\frac{1}{2}} V_+^T$.

Capítulo 6

Recuperação de Informação

6.1 - Introdução

A evolução das bibliotecas digitais e o crescimento exponencial da quantidade de documentos disponíveis na Internet transformou profundamente o processamento, o armazenamento e a procura de informação. Neste contexto tecnológico, sistemas digitais armazenam dados relativos a pesquisas, factos ou acontecimentos relatados diariamente, que são disponibilizados para buscas e utilizados por um grande número de pessoas.

Uma maneira comum de disponibilizar esses dados na Internet, é sob a forma de hiperdocumentos, que podem ser vistos como uma rede de informações estruturadas com ligações entre si que são facilmente acessíveis aos utilizadores. Um dos problemas relacionados com esta forma de se estruturar e disponibilizar a informação é a dificuldade em se realizar pesquisas sobre conteúdos, na Internet. Essa procura de hiperdocumentos expõe os utilizadores a uma pesada sobrecarga cognitiva, uma vez que são eles próprios os responsáveis por analisar a informação e relacioná-la do ponto de vista semântico.

Desta forma, tornaram-se necessários métodos eficazes para o armazenamento, o processamento e a recuperação de informações. É o que acontece com a generalidade dos motores de busca existentes actualmente.

Os utilizadores definem perguntas, através de palavras-chave e o sistema fornece conjuntos de documentos relacionados com elas através de um processamento de dados. Essas palavras-chave são comparadas com palavras presentes nos documentos da base de dados. Uma desvantagem desta abordagem é que a recuperação baseada em palavras-chave geralmente introduz uma distância semântica entre a necessidade do utilizador e o conjunto de documentos que são devolvidos, ou seja, nem sempre os documentos devolvidos são os do interesse do utilizador. Essa distância pode aumentar perante a dificuldade em se trabalhar com texto em linguagem natural, pois estes textos podem não ser bem estruturados ou serem ambíguos. Como resultado, quando efectuamos uma pesquisa, a presença de documentos não relevantes entre os documentos devolvidos é bastante comum.

A área da ciência da computação denominada Recuperação de Informação, *RI*, é uma área que trata da representação, do armazenamento, da organização e do acesso a informações que tanto podem estar na forma de texto como de imagens multimédia. Apesar destas diferenças de formato, um sistema de recuperação de informação trabalha com essas informações como se fossem apenas textos.

Muitas das vezes são associados índices aos documentos para se fazer a pesquisa de informação. Para manipular colecções de documentos e para comparar consultas aos documentos, os sistemas de Recuperação de Informação utilizam uma série de mecanismos tais como: análise textual léxica; filtragem de informação por meio de extracção de "*stopwords*[6]"; técnicas de redução de palavras aos seus radicais; técnicas de indexação como, por exemplo, arquivo invertido, que corresponde a um índice textual composto por um vocabulário e uma lista de ocorrências; modelos matemáticos e estatísticos para a representação de documentos, de consultas e a definição de coeficientes de similaridades; ou estruturas de categorização e de expansão de consultas por meio da utilização de, por exemplo, dicionário de sinónimos e análise de relevância por parte do utilizador.

Assim, o principal objectivo de um sistema de Recuperação de Informação é recuperar o maior número possível de documentos relevantes e o menor número possível de documentos não relevantes.

Uma forma trivial de gerar um conjunto resposta para a consulta do utilizador é determinar quais são os documentos de uma colecção que contêm exactamente as

[6] Conjunções, proposições e artigos.

palavras-chave presentes na consulta. Esta técnica é muito utilizada, mas tal forma de recuperação implica que documentos relevantes para o utilizador possam não ser recuperados. E além disso, muitas palavras possuem múltiplos significados; como resultado, palavras numa consulta podem aparecer em documentos não relevantes.

Numa tentativa de melhor satisfazer a necessidade de informação do utilizador, tenta-se que os documentos devolvidos sejam ordenados de acordo com o grau de relevância em relação a essa consulta.

É evidente que o sucesso e a eficiência de um sistema de Recuperação de Informação são medidos de maneira subjectiva. Por vezes, as informações desejadas são todos os dados que o sistema possui relacionados com a pesquisa do utilizador. Noutras, o utilizador deseja apenas algumas informações que lhe sejam suficientes, e a devolução de todos os dados relevantes poderia dificultar-lhe o trabalho.

Por outro lado, na visão do sistema, algumas informações não relevantes para o utilizador são consideradas como relevantes. Em resumo, pode acontecer uma das seguintes situações:

- Documentos relevantes serem devolvidos;
- Documentos relevantes não serem devolvidos;
- Documentos não relevantes serem devolvidos;
- Documentos não relevantes não serem devolvidos.

Encontramos aqui uma situação análoga à que acontece no reconhecimento facial, onde uma face que é detectada pode ser reconhecida ou não; ou outra imagem que não é uma face pode ser erradamente reconhecida ou não representar uma face para o sistema.

6.2 – O modelo vectorial

Actualmente este é o modelo mais utilizado. A recuperação de documentos relevantes é feita através da comparação de termos entre os documentos da colecção e a consulta.

Neste modelo os dados são representados através de uma matriz:

$$A = \begin{bmatrix} a_{11} & \cdots & a_{1j} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ a_{i1} & \cdots & a_{ij} & \cdots & a_{in} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{mj} & \cdots & a_{mn} \end{bmatrix} \begin{matrix} t_i \\ \\ \\ \\ \\ \end{matrix}, \qquad (6.1)$$

$$d_j$$

onde t_i são os termos e d_j os documentos.

Na matriz *A*, designada por ***matriz de termos × documentos***, cada a_{ij} é o peso de cada termo t_i associado ao documento d_j ,$1 \le i \le m, 1 \le j \le n$, sendo *m* e *n* o número de termos e documentos respectivamente, contidos na base de dados do sistema. As colunas da matriz *A* podem ser ou não normalizadas.

Normalmente cada a_{ij} é decomposto em l_{ij} e g_i, tal que $a_{ij=}\ l_{ij}\ g_i$. Neste caso g_i é a influência global que um termo t_i tem sobre todos os documentos, por sua vez, l_{ij} é o peso local que um termo t_i possui sobre um documento d_j.

Suponhamos, por exemplo, temos um conjunto de documentos com conteúdo matemático. O termo 'matemática', nesse caso teria uma influência global, sobre todos os documentos. Como a inclusão deste termo na descrição dos documentos não é muito importante, é suficiente termos um g_i pequeno. A palavra 'geometria', por sua vez, é mais específica, nem todos os documentos iriam tratar disso, por isso é apropriado termos pesos l_{ij} diferentes para cada documento d_j.

Os pesos l_{ij} podem ser representados de diversas formas: como uma variável boleana, frequência do termo no documento, funções envolvendo logaritmos, entre outras. Caso se utilizem pesos binários, como faremos nesta apresentação, $l_{ij} = 0$ se o termo não estiver relacionado com documento e $l_{ij} = 1$, caso contrário.

Os pesos g_i podem também ser representados de diversas formas. Podem ser feitas normalizações, entre outro tipo de transformações. Neste trabalho consideraremos $g_i = 1$, ou seja, $a_{ij} = l_{ij}$.

No modelo vectorial, cada pesquisa é definida como um vector $p = (p_1, \ldots, p_m)^T$ e a medida de similaridade entre uma pesquisa *p* e um documento $d_j = (a_{1j}, \ldots, a_{mj})^T$, é dada pelo coseno do ângulo θ_j formado por d_j e *p*:

$$\cos(\theta_j) = \frac{d_j^T p}{\|d_j\|_2 \|p\|_2} \tag{6.2}$$

Se não existir nenhuma relação entre a pesquisa *p* e o documento d_j, diremos que os dois vectores são ortogonais sendo então $\cos(\theta_j) = 0$. Quanto maior for o coseno, mais d_j e *p* estão relacionados.

O conjunto de documentos relevantes para a pesquisa *p* é designado por

$$R_p = \left\{ d_j \middle| \cos(\theta_j) > L \right\}, \tag{6.3}$$

sendo *L* um limiar previamente definido. (Se *cos* (θ_j) > *L*, então d_j é um documento relevante para a pesquisa *p*)

O valor para *L* é normalmente definido através de experiências e depende do tipo de colecção com que se está a trabalhar. De um modo geral o limiar para a similaridade acima mencionada é 0.9 [20].

Um valor elevado para o limiar pode ser justificado pelo facto de que quando se trabalha com bases de dados com milhões de documentos, se o limiar não for elevado quando se faz uma pesquisa, são devolvidos milhares de documentos, o que pode dificultar o trabalho à pessoa que efectuou a pesquisa. Assim são devolvidos apenas os documentos com maior relevância.

Exemplo:

Consideremos os *m* = 7 termos

T_1: geometri(a)(camente) T_4: seminário(s)

T_2: exponencial T_5: professores

T_3: universidade T_6: matemática

T_7: estudo

Observemos que os termos são indicados pelo radical da palavra. Por essa razão 'geometria' e 'geometricamente' são semanticamente equivalentes e identificados de maneira única. Do mesmo modo, o plural de uma palavra é identificado juntamente com o seu singular correspondente.

Consideremos agora os $n = 6$ documentos existentes no sistema:

D_1= **Seminários** para **professores** na **Universidade** do Algarve: 'Um **estudo** acerca da **geometria** hiperbólica';

D_2= O crescimento **exponencial** do preço dos combustíveis;

D_3= Horários dos **professores** de **Matemática;**

D_4= A diminuição do abandono escolar;

D_5= **Seminário**: 'G**eometricamente** falando';

D_6= Milhares de **professores** no desemprego.

A matriz A, de termos × documentos, é dada por:

$$A = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix},$$

onde 1 significa que o termo está no documento e 0 o caso contrário.

Normalizando (norma - dois) as colunas de A, obtém-se:

$$A = \begin{bmatrix} 0.4472 & 0 & 0 & 0 & 0.7071 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0.4472 & 0 & 0 & 0 & 0 & 0 \\ 0.4472 & 0 & 0 & 0 & 0.7071 & 0 \\ 0.4472 & 0 & 0.7071 & 0 & 0 & 1 \\ 0 & 0 & 0.7071 & 0 & 0 & 0 \\ 0.4472 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Seja agora p_1 o vector de pesquisa que representa a procura pela palavra 'seminário':

$$p_1 = [0 \quad 0 \quad 0 \quad 1 \quad 0 \quad 0 \quad 0]^T$$

Aplicando (6.2), obtemos o vector de cosenos:

$$\cos(\theta) = [0.4472 \quad 0 \quad 0 \quad 0 \quad 0.7071 \quad 0]^T .$$

Se estabelecermos um limiar, por exemplo, de 0.4, uma vez que se está perante uma colecção de documentos e termos escassos, verificamos que todos os documentos relevantes são devolvidos, tendo o D_5 uma maior relevância.

Consideremos agora a pesquisa pelas palavras: 'seminário' e ' professor':

$$p_2 = [0 \quad 0 \quad 0 \quad 1 \quad 1 \quad 0 \quad 0]^T$$

O vector de cosenos, que mede a similaridade entre a pesquisa p_2 e os seis documentos, é o seguinte:

$$\cos(\theta) = [0.6324 \quad 0 \quad 0.5 \quad 0 \quad 0.5 \quad 0{,}7071]^T$$

Considerando o mesmo limiar, continuamos a verificar que todos os documentos relevantes são devolvidos. Verificamos que o último é o que tem maior peso; no entanto a palavra seminário não está contemplada neste documento. Portanto este documento não deverá ser o mais relevante para esta pesquisa.

Diversos tipos de técnicas são utilizados para evitar este e outro tipo de erros, comuns neste modelo [20, 21], que também não considera as relações entre os termos e o contexto em que estão inseridos. É sabido que muitas palavras têm múltiplos significados e os termos de uma consulta poderão estar presentes em documentos irrelevantes. Além disso, podem haver muitas maneiras de expressar um conceito e termos de documentos relevantes que não estão indexados por algum termo da consulta não serão recuperados.

6.3 – Indexação Semântica Latente

O método de Indexação Semântica Latente ou Análise Semântica Latente é uma variante do modelo vectorial, em que é feita uma aproximação da matriz de documentos × termos por uma matriz de característica inferior [22]. Este método tem como suporte a

decomposição em valores singulares e faz uso do modelo vectorial para representar as descrições de documentos, de termos e de consultas.

Neste caso consideramos $A = U\Sigma V^T$, onde as colunas da matriz U são designadas por ***vectores dos termos*** e colunas de V por ***vectores dos documentos***.

O modelo de Indexação Semântica Latente realiza uma análise estatística do uso das palavras entre todos os documentos da base de dados, permitindo que documentos relacionados semanticamente com uma consulta sejam recuperados, mesmo sem compartilhar os mesmos termos. Foi criado para minimizar um problema fundamental que dificulta as técnicas de recuperação existentes. Um dos principais problemas é que os utilizadores querem recuperar dados na base do significado dos conceitos e as palavras soltas (fora do seu contexto) não são evidentes acerca do seu significado. Como há muitas maneiras de exprimir um conceito, os termos isolados podem não coincidir com o seu significado num documento relevante. Há a acrescentar que muitas palavras têm vários significados, então os mesmos termos em diferentes documentos podem não ser do interesse do utilizador. Tem-se assim que algumas dificuldades na Recuperação de Informação automática são: os diferentes idiomas; vários tipos de informação: texto, figura, áudio, vídeo; sinónimos (várias palavras com o mesmo significado); polissemia (palavras que se escrevem da mesma maneira com significados diferentes), enorme quantidade de documentos e um recurso limitado de processamento. O método de indexação semântica latente tenta solucionar estes problemas por meio da organização automática de texto numa estrutura semântica mais apropriada para a recuperação de informação.

Neste modelo utiliza-se a matriz de termos × documentos com característica reduzida, pois o espaço gerado pelas colunas da matriz A original não é, necessariamente, a melhor representação da base de dados. A redução da característica de A permite remover algumas informações menos pertinentes.

Seja A_k uma matriz aproximada da matriz A, matriz de termos × documentos. Normalmente k é definido através de experiências. Na escolha desse valor k tem de se ter em conta que ele deve ser suficientemente grande para descrever a estrutura dos dados e suficientemente pequeno para desprezar erros ou detalhes sem importância. Esta aproximação de A diminui a probabilidade de consultas e documentos referenciarem o mesmo conceito utilizando termos diferentes. A partir disso, a matriz aproximada é

considerada para organizar a estrutura semântica e a aproximação das informações a serem recuperadas.

Outro aspecto que também há a salientar é que considerando a existência de muitas palavras-chave manipuladas por um sistema de Recuperação de Informação, é natural que o número de termos relacionados com um documento seja pequeno em relação ao número total de termos de uma colecção. A matriz *A* é, portanto, esparsa, bem como o vector *p*. É de notar que como *p* e *A* são esparsos, os cálculos são consideravelmente pequenos.

Utilizando a decomposição em valores singulares no exemplo atrás mencionado,

$$U = \begin{bmatrix} -0.3292 & 0.5824 & -0.0000 & 0.1727 & 0.1503 & -0.1570 & -0.6895 \\ 0.0000 & 0.0000 & 1.0000 & 0.0000 & -0.0000 & -0.0000 & 0.0000 \\ -0.1727 & 0.1503 & 0.0000 & -0.3292 & -0.5824 & 0.6895 & -0.1570 \\ -0.3292 & 0.5824 & -0.0000 & 0.1727 & 0.1503 & 0.1570 & 0.6895 \\ -0.8121 & -0.4528 & 0.0000 & -0.2533 & 0.2671 & 0.0000 & 0.0000 \\ -0.2533 & -0.2671 & -0.0000 & 0.8121 & -0.4528 & -0.0000 & -0.0000 \\ -0.1727 & 0.1503 & 0.0000 & -0.3292 & -0.5824 & -0.6895 & 0.1570 \end{bmatrix}$$

$$\Sigma = \begin{bmatrix} 1.4502 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1.1609 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1.0000 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.5866 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.4529 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.0000 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$V = \begin{bmatrix} -0.5600 & 0.3901 & 0.0000 & -0.4318 & -0.5898 & 0.0000 \\ 0.0000 & 0.0000 & 1.0000 & 0.0000 & 0.0000 & 0.0000 \\ -0.5194 & -0.4385 & 0.0000 & 0.6736 & -0.2900 & 0.0000 \\ 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & -1.0000 \\ -0.3210 & 0.7095 & 0.0000 & 0.4163 & 0.4693 & 0.0000 \\ -0.5600 & -0.3901 & 0.0000 & -0.4318 & 0.5898 & 0.0000 \end{bmatrix}.$$

Como se viu no capítulo 2, uma aproximação de *A* com característica inferior é dada por $A_k = U_k \Sigma_k V_k^T = \sum_{j=1}^{k} \sigma_j \, u_j \, v_j^T$, podendo portanto desprezar-se os valores singulares mais pequenos, encontrando desta forma uma matriz aproximada da matriz inicial.

Vamos agora mostrar como utilizar a decomposição em valores singulares para calcular a similaridade entre termos e documentos numa pesquisa.

Seja A_k a matriz de *termos* × *documentos* aproximada, com característica *k, p* o vector de pesquisa e $A_k e_j$ a coluna *j* da matriz A_k. Então os cosenos dos ângulos entre o vector de pesquisa e os vectores dos documentos aproximados são dados por:

$$\cos(\theta_j) = \frac{(A_k e_j)^T p}{\|A_k e_j\|_2 \|p\|_2} = \frac{(U_k \Sigma_k V_k^T e_j)^T p}{\|U_k \Sigma_k V_k^T e_j\|_2 \|p\|_2} = \frac{e_j^T V_k \Sigma_k (U_k^T p)}{\|\Sigma_k V_k^T e_j\|_2 \|p\|_2}, \qquad (6.4)$$

para *j*= 1, …., *n*.

Considerando

$$s_j = \Sigma_k V_k^T e_j \ , \qquad (6.5)$$

para *j*=1, …., *n* , tem-se:

$$\cos(\theta_j) = \frac{s_j^T (U_k^T p)}{\|s_j\|_2 \|p\|_2}. \qquad (6.6)$$

É de notar que (6.6) pode ser calculado sem formar a matriz A_k explicitamente e que $\|s_j\|_2$ é calculada apenas uma vez para cada matriz de *termos* × *documentos* e independentemente da pesquisa.

Para o exemplo, consideremos $A \approx A_4 = \sum_{j=1}^{4} \sigma_j \, u_j \, v_j^T$. Seria no entanto necessário efectuarmos experiências para verificar se esta aproximação era suficientemente boa ou não para recuperar sempre a informação que é pertinente.

Consideremos a pesquisa pelas palavras 'seminário' e ' professor'. Esta é representada pelo vector $p_2 = [0 \quad 0 \quad 0 \quad 1 \quad 1 \quad 0 \quad 0]^T$.

Ao aplicarmos (6.6) obtemos:

$\cos(\theta) = [0.7381 \quad 4.4068\times10^{-17} \quad 0.5435 \quad 2.1290\times10^{-16} \quad 0.4475 \quad 0.6520]^T$. No caso de utilizarmos o mesmo limiar, *L*=0.4, seriam devolvidos quatro documentos, sendo D_1

o documento de maior relevância, seguido por D_6, D_3 e D_5. Este resultado é mais satisfatório do que o anterior obtido através de (6.2), uma vez que todos os documentos pertinentes são devolvidos e ordenados pelo seu grau de relevância. Repare-se que o primeiro documento contém as palavras 'seminário' e ' professor', sendo o documento de maior relevância.

Verifica-se assim com um exemplo que a decomposição em valores singulares é muito útil na Recuperação de Informação, nomeadamente na Indexação Semântica Latente.

A matriz A_k captura a estrutura mais importante de associação entre termos e documentos, ignorando o ruído, ou seja, as estruturas com um menor grau de associação, [21].

No contexto da Recuperação de Informação, para além da decomposição em valores singulares, podemos também utilizar a factorização QR.

Consideremos a decomposição em valores singulares da matriz A, $A = U\Sigma V^T$, e a factorização QR, $A=QR$. É sabido que em qualquer uma das situações podemos encontrar uma matriz A_k aproximada de A, com característica k, sendo $A_k = U_k \Sigma_k V_k^T$ (como já foi definido nas secções anteriores) e $A_k = Q_k R_k$, onde as matrizes Q_k e R_k são constituídas pelas primeiras k colunas da matriz Q e da matriz R, respectivamente. Uma grande diferença entre estas duas factorizações é que a SVD fornece duas bases, uma para o espaço gerado pelas colunas de A e outra para o espaço gerado pelas linhas de A, enquanto que a QR fornece apenas uma base para o espaço gerado pelas colunas da matriz A. O facto de encontrarmos uma base para o espaço gerado pelas linhas da matriz A quando utilizamos a SVD é uma vantagem, pois assim podem ser feitas comparações entre termos [21], como veremos de seguida.

6.3.1 – Comparação entre termos

No contexto do modelo vectorial e, particularmente da Indexação Semântica Latente, podemos também fazer comparações entre termos. Este facto ajuda bastante os utilizadores quando efectuam pesquisas.

A comparação é feita através do valor do coseno do ângulo formado entre os vectores de termos.

Considerando um par de termos *i* e *j*, a similaridade será, dada por:

$$sim(\text{termo}\, i, \text{termo}\, j) = \cos(w_{ij}) = \frac{(e_i^T A)(A^T e_j)}{\left\|A^T e_i\right\|_2 \left\|A^T e_j\right\|_2}, \tag{6.7}$$

para *i, j* =1, …., *m.*

Utilizando agora uma aproximação de *A* com característica *k* e a sua decomposição em valores singulares, $A_k = U_k \Sigma_k V_k^T$, obtemos:

$$\cos(w_{ij}) = \frac{(e_i^T U_k \Sigma_k V_k^T)(V_k \Sigma_k U_k^T e_j)}{\left\|V_k \Sigma_k U_k^T e_i\right\|_2 \left\|V_k \Sigma_k U_k^T e_j\right\|_2} = \frac{(e_i^T U_k \Sigma_k)(\Sigma_k U_k^T e_j)}{\left\|\Sigma_k U_k^T e_i\right\|_2 \left\|\Sigma_k U_k^T e_j\right\|_2} \tag{6.8}$$

para *i, j* =1, …., *m.*

Considerando $s_j = \Sigma_k U_k^T e_j$, obtemos:

$$\cos(w_{ij}) = \frac{s_i^T s_j}{\left\|s_i\right\|_2 \left\|s_j\right\|_2} \tag{6.9}$$

para *i, j* =1, …., *m.*

Com os resultados da similaridade entre os diferentes documentos constrói-se uma matriz *W*, tal que $W_{ij} = \cos(w_{ij})$. Esta matriz é simétrica, pois $sim(\text{termo}\, i, \text{termo}\, j)$ é igual a $sim(\text{termo}\, j, \text{termo}\, i)$.

As entradas W_{ij} mostram a associação entre os termos *i* e *j*. Se a entrada é próxima de 1, os termos são similares e têm funções semelhantes na descrição semântica do documento. Se a entrada é próxima de zero, os vectores são quase ortogonais e correspondem a termos que não estão relacionados.

A separação dos vectores de termos pela sua similaridade leva à construção de grupos de termos semanticamente independentes. A este processo dá-se o nome de *clustering*. Esta é uma das formas de ultrapassar o problema da polissemia. Geometricamente, o cálculo de similaridades acima mostra que dois vectores de termos são relacionados se eles estão suficientemente próximos no espaço gerado pelas linhas da matriz de termos × documentos. Além disso, eles não têm relação alguma se forem ortogonais entre si.

No processo de agrupamento, os termos são associados àqueles que normalmente surgem com eles nos diversos documentos de uma base de dados. Quando se efectua uma expansão da pesquisa, essa expansão é feita com base neste facto.

Capítulo 7

Descodificação de mensagens cifradas

7.1 – Introdução

Desde há bastante tempo que a descodificação de mensagens cifradas, também conhecidas por criptogramas, tem sido uma actividade muito popular. Existem muitas maneiras para efectuar essa descodificação.

A criptanálise ou criptoanálise é um método para analisar mensagens cifradas com o objectivo de decifrá-las.

Neste capítulo pretende-se descrever como a decomposição em valores singulares pode ajudar a descodificação de mensagens, nomeadamente na descoberta de vogais e consoantes em mensagens codificadas. Será aqui apresentado o trabalho exposto por Cleve Moler e Donald Morrison [23]. Estes autores, sendo de língua inglesa, fazem o estudo incidir sobre o inglês, utilizando as componentes dos primeiros vectores singulares para aproximar a frequência de cada letra no criptograma e os segundos para separar as vogais das consoantes.

7.2 – O problema dos criptanalistas

Os criptanalistas são aqueles que fazem estudos de técnicas matemáticas, na tentativa de quebrar mensagens cifradas.

A técnica que a seguir se expõe resulta para mensagens codificadas através da troca de letras numa mensagem, por uma permuta entre si. Quando o criptanalista analisa uma mensagem, começa por contar as ocorrências das letras, individualmente, na mensagem e compara a frequência de ocorrência com a do texto normal. No entanto, é necessário que a mensagem seja suficientemente longa para que esta técnica resulte. Um procedimento que é indispensável para descodificar é fazer uma partição do criptograma em dois conjuntos sendo um o das consoantes e outro o das vogais.

Os textos escritos em diversas linguagens, incluindo o inglês, têm a propriedade de que as vogais são frequentemente seguidas por consoantes e vice-versa. No entanto há algumas excepções; por exemplo, nos textos escritos em inglês a letra '*h'* muitas vezes é precedida por outras consoantes para formar os sons: *th*, *gh*, *ph*, *sh* e *ch*.

Dizemos que um texto é "*vogal depois de consoante"* ou mais simplesmente "*vdc*" se a proporção de vogais que seguem vogais é menor que a proporção de vogais que seguem consoantes, ou seja:

$$\frac{\text{nº de pares (vogal, vogal)}}{\text{nº de vogais}} < \frac{\text{nº de pares (consoante, vogal)}}{\text{nº de consoantes}}. \qquad (7.1)$$

Ao dividirmos o texto em dois conjuntos verificamos se a regra "*vdc*" é cumprida ou não. Esta é uma condição essencial para que a técnica resulte; qualquer experiência que tente todas as partições possíveis antes de satisfazer a regra "*vdc*" fará com que o procedimento não resulte.

Seja *n*, o número de letras do alfabeto. O diagrama de frequência é uma matriz $A_{n\times n}$ sendo o elemento a_{ij} o número de ocorrências que a letra *j* segue a letra *i*. Os espaços e pontuação são ignorados e a primeira letra do texto é assumida como seguindo a última.

Em geral, $A_{n\times n}$ não é simétrica e a letra *i* ocorre um número de vezes igual a

$$f_i \equiv \sum_j a_{ij} = \sum_j a_{ji} . \qquad (7.2)$$

Verifica-se assim que a soma da linha *j* e da coluna *j* é igual ao número de vezes que a letra *i* aparece no texto.

Para a partição, ou seja, a divisão das letras do texto em dois conjuntos disjuntos, podem definir-se os vectores v e c da seguinte forma:

$$v_i = \begin{cases} 1, & \text{se é vogal,} \\ 0, & \text{se não é vogal.} \end{cases} \quad c_i = \begin{cases} 1, & \text{se é consoante,} \\ 0, & \text{se não é consoante.} \end{cases}, \tag{7.3}$$

onde $i = 1, \ldots, 26$.

Verifica-se que os vectores *v* e *c* são ortogonais e que

$$v^T A v = \text{número de pares (vogal, vogal) no texto,} \tag{7.4}$$

$$c^T A c = \text{ número de pares (consoante, consoante) no texto} \tag{7.5}$$

Logo a regra "*vdc*" pode escrever-se

$$\frac{v^t A v}{v^T A(v+c)} < \frac{c^t A v}{c^T A(v+c)}; \tag{7.6}$$

ou seja,

$$D = \left(v^t A v\right)\left(c^t A c\right) - \left(v^t A c\right)\left(c^t A v\right) < 0. \tag{7.7}$$

Então o problema do criptanalista é: dado um diagrama de frequências *A*, encontrar uma partição *v* e *c* que verifique (7.7).

Utilizando a SVD podem escrever-se matrizes aproximadas de *A*:

- **Aproximação de *A* com característica 1**

$$A \approx A_1 = \sigma_1 x_1 y_1^T \tag{7.8}$$

Sendo $e = (1, \ldots, 1)^T$ e *f* o vector em que cada entrada, f_i, é o número de ocorrências da *i*-ésima letra no texto, então

$$Ae = A^T e = f \tag{7.9}$$

e logo

$$\sigma_1 \left(y_1^T e \right) x_1 \approx \sigma_1 \left(x_1^T e \right) y_1 \approx f \ . \tag{7.10}$$

Os primeiros vectores singulares esquerdos e direitos tendem a ser aproximadamente iguais e reflectem a frequência das letras no texto. Utilizando esta informação e a aproximação com característica um de *A*, o criptanalista deve ser capaz de estimar, com algum grau de confiança, as entradas do alfabeto permutado.

- **Aproximação de *A* com característica 2**

$$A \approx A_2 = \sigma_1 \, x_1 \, y_1^T + \sigma_2 \, x_2 \, y_2^T \tag{7.11}$$

É a mais simples aproximação que tem em conta a correlação entre os pares de letras no texto. Os segundos vectores singulares (esquerdo e direito) contêm a chave para a solução do problema do criptanalista.

Os sinais dos componentes de x_2 e de y_2 são usados para dividir as letras da mensagem da seguinte forma:

$$v_i = \begin{cases} 1, & \text{se } x_{i2} > 0 \text{ e } \ y_{i2} < 0 \\ 0, & \text{caso contrário.} \end{cases} \tag{7.12}$$

$$c_i = \begin{cases} 1, & \text{se } x_{i2} < 0 \text{ e } \ y_{i2} > 0 \\ 0, & \text{caso contrário.} \end{cases} \tag{7.13}$$

$$n_i = \begin{cases} 1, & \text{se } \ \text{sign} \left(x_{i2} \right) = \text{sign} \left(y_{i2} \right) \\ 0, & \text{caso contrário.} \end{cases} \tag{7.14}$$

A terceira categoria de letras, letras neutras, são os que não podem ser classificados nem como vogais nem como consoantes. Na prática poucas letras ficam nesta categoria. No texto escrito em inglês, por exemplo, o '*h*' normalmente é uma letra neutra, pois é seguida por uma vogal e precedida por uma consoante.

Antes de se analisar como se encontra a solução para o problema dos criptanalistas apresenta-se uma definição, o teorema de Perron-Frobenius e um lema.

Definição 7.1: Uma matriz $A_{n\times n}$ diz-se irredutível se não existe uma matriz de permutação P, que transforme A numa matriz triangular por blocos, sendo os blocos da diagonal matrizes quadradas.

Teorema 7.1 (Perron-Frobenius): Seja $A_{n\times n}$ uma matriz irredutível, com entradas a_{ij} não negativas. Então:

1. A tem um valor próprio real e positivo r tal que qualquer outro valor próprio λ satisfaz: $|\lambda| < r$;
2. O valor próprio r é simples: r é uma raiz simples do polinómio característico em A. O espaço próprio esquerdo ou direito associado tem dimensão 1;
3. Há um vector próprio esquerdo e um vector próprio direito associados a r apenas com entradas positivas. Isto significa que existe um vector linha $v = (v_1, \ldots, v_n)$ e um vector coluna $w = (w_1, \ldots, w_n)^T$ com entradas positivas $v_i > 0$, $w_i > 0$, tais que $vA = rv$ e $Aw = rw$;

Lema 7.1: x_1 e y_1 têm todas as componentes não negativas.

Demonstração: (ver, por exemplo, [24])

O teorema de *Perron-Frobenius* implica que se A é não negativa e irredutível então o vector próprio correspondente ao máximo valor próprio tem componentes positivas.
Sendo

$$A = U \Sigma V^T$$

Verifica-se que

$$A^T A x_1 = \lambda_1 x_1$$

e

$$AA^T y_1 = \kappa_1 y_1,$$

sendo λ_1 e κ_1 valores próprios de $A^T A$ e de AA^T respectivamente.

Como A é não negativa e irredutível implica que $A^T A$ e AA^T também o são.

Como σ_1 é o maior valor singular de A e $\sigma_i = \sqrt{\lambda_i} = \sqrt{\kappa_i}$, λ_1 e κ_1 são os maiores valores próprios. Logo pelo teorema *Perron-Frobenius* x_1 e y_1 têm componentes positivas.

A solução para o problema do criptanalista usando as partições v_i e c_i é encontrada verificando o teorema que a seguir se apresenta.

Teorema 7.2: Seja $A \approx A_2$ uma matriz não negativa de característica dois. Sejam *v* e *c* definidas por (7.12) e (7.13) respectivamente, então a regra "*vdc*",
$D = \left(v^t Av\right)\left(c^t Ac\right) - \left(v^t Ac\right)\left(c^t Av\right) < 0$, é satisfeita.

Demonstração:

Substituindo A_2 em *D*, obtém-se:

$$D = \left(v^t\left(\sigma_1 x_1 y_1^T + \sigma_2 x_2 y_2^T\right)v\right)\left(c^t\left(\sigma_1 x_1 y_1^T + \sigma_2 x_2 y_2^T\right)c\right)$$

$$-\left(v^t\left(\sigma_1 x_1 y_1^T + \sigma_2 x_2 y_2^T\right)c\right)\left(c^t\left(\sigma_1 x_1 y_1^T + \sigma_2 x_2 y_2^T\right)v\right)$$

$$= \sigma_1\sigma_2\left(\left(\underbrace{v^t x_1 y_1^T v}_{1}\right)\left(\underbrace{c^t x_2 y_2^T c}_{2}\right) + \left(\underbrace{v^t x_2 y_2^T v}_{3}\right)\left(\underbrace{c^t x_1 y_1^T c}_{4}\right)\right.$$

$$\left. -\left(\underbrace{v^t x_1 y_1^T c}_{5}\right)\left(\underbrace{c^t x_2 y_2^T v}_{6}\right) - \left(\underbrace{v^t x_2 y_2^T c}_{7}\right)\left(\underbrace{c^t x_1 y_1^T v}_{8}\right)\right).$$

Os produtos 1, 4, 5 e 8 são não negativos, pois pelo lema 7.1, x_1 e y_1 têm todas as componentes não negativas, assim como os vectores *c* e *v*.

Considerem-se as definições (7.12) e (7.13) para v_i e c_i. De todos os produtos restantes desta expressão, apenas $y_2^T v$ e $c^t x_2$ são negativos. Consequentemente todos os termos dentro de parênteses são negativos logo a expressão *D* é negativa, como é pretendido.

Suponha-se que se pretende descodificar a seguinte mensagem (apresentada por Bruce Schatz [24]):

xvqiq hiq h miqhx chso xvzsma ey zccqsaq zsxqiqax hpekx lvzjv ajzqsjq, hx niqaqsx, bsela tzxxtq. zx za xvq uep ey nvzteaenvo xe bqqn anqjkthxzes hpekx xvqaq htzdq

Para separar no criptograma as vogais das consoantes e das letras neutras utilizámos a função digraph.m, disponibilizada pela colecção NCM, que pode ser consultada em http://www.mathworks.com/moler/ncmfilelist.html.

Nesta função constrói-se a matriz $A_{26\times 26}$, tendo em conta que cada a_{ij} é o número de ocorrências que a letra j segue a letra i. Seguidamente calcula-se a sua SVD e constrói-se o gráfico da distribuição das letras de acordo com o sinal das coordenadas dos segundos vectores singulares esquerdo e direito, conforme se apresenta na tabela a seguir.

Letras	**A**	**B**	**C**	**D**	**E**	**F**	**G**	**H**	**I**	**J**	**K**	**L**	**M**
$\mathbf{U_2}$	-0.34	-0.06	-0.04	-0.11	0.13	0	0	0.13	-0.55	-0.13	0.01	0.03	0.08
$\mathbf{V_2}$	0.28	0.0002	-0.02	0.005	-0.15	0	0	0.45	0.21	0.07	0.019	0.0002	0.04
Letras	**N**	**O**	**P**	**Q**	**R**	**S**	**T**	**U**	**V**	**W**	**X**	**Y**	**Z**
$\mathbf{U_2}$	-0.07	0.02	-0.07	0.61	0	0.12	-0.07	-0.02	-0.30	0	-0.12	0.0013	0.03
$\mathbf{V_2}$	0.0416	-0.03	0.06	-0.71	0	0.32	-0.04	0.09	-0.11	0	0.07	0.04	-0.03

Tabela 7.1: Segundos vectores singulares esquerdo e direito da matriz *A*

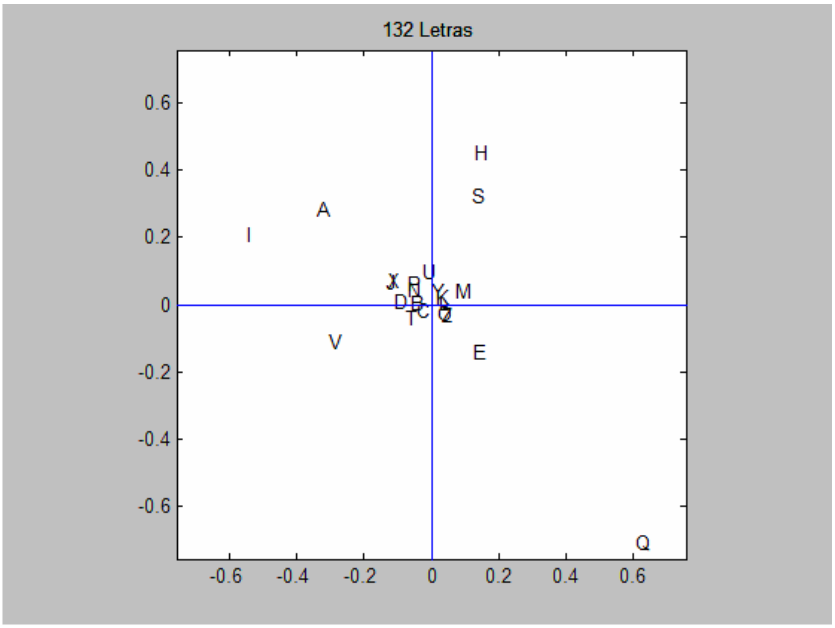


Gráfico 7.1 – Distribuição das vogais, consoantes e letras neutras

Tendo em conta as definições (7.12), (7.13) e (7.14) e o gráfico obtido verifica-se que E e Q são vogais; I e A são consoantes e V, M, S e H são letras neutras.

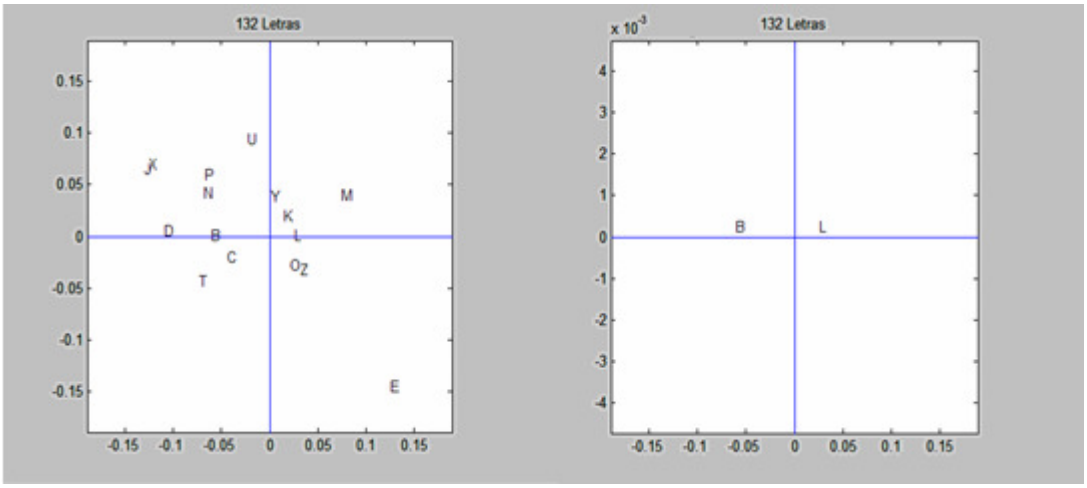


Gráfico 7.2 – Separação das vogais, consoantes e letras neutras

Verifica-se que B é uma consoante e que L é uma letra neutra.

As letras no criptograma classificam-se da seguinte forma: as vogais são E O Q Z, as consoantes ABDIJNPUX e as letras neutras: CHKLMRSTVY.

Para se descodificar a mensagem começa-se por contar a frequência de cada letra no criptograma:

Letras	Frequência	Letras	Frequência
Q	20	I, N, T	5
X	16	J	4
Z	11	C, K, P	3
A, E	10	B, L, M, O, Y	2
H	9	D, U	1
S, V	8	F, G, R, W	0

Tabela 7.2 – Frequência das letras no criptograma

Depois é feita a comparação com a frequência das letras no texto normal, escrito em inglês. Essa frequência é apresentada na tabela a seguir:

Letra	%	Letra	%	Letra	%	Letra	%
A	8.06	H	5.99	O	8.00	V	0.97
B	1.49	I	6.85	P	1.97	W	2.06
C	2.75	J	0.17	Q	0.09	X	0.21
D	4.09	K	0.68	R	6.20	Y	1.70
E	12.51	L	3.68	S	6.11	Z	0.07
F	2.79	M	2.37	T	9.50		
G	1.80	N	7.12	U	2.77		

Tabela 7.3 - Frequência das letras no inglês

Seguidamente faz-se a correspondência entre as letras do criptograma e as do texto normal, ou seja, decifra-se a mensagem, obtendo-se:

xvqiq hiq h miqhx chso xvzsma ey zccqsaq zsxqiqax hpekx lvzjv ajzqsjq, hx niqaqsx,
there are a great many things of immense interest about which science at present
bsela tzxxtq zx za xvq uep ey nvzteaenvo xe bqqn anqjkthxzes hpekx xvqaq htzdq
knows little it is the job of philosophy to keep speculation about these alive

Esta mensagem cifrada serviu apenas de exemplo ilustrativo, pois é demasiado curta para que se possa ter a certeza de que a técnica resulta na íntegra.

O código utilizado anteriormente pode ser também utilizado para fazer experiências em textos não cifrados escritos noutras línguas, para fazer a separação de vogais consoantes e letras neutras.

Capítulo 8

A expressão dos genes

8.1 – Introdução

Em 1869, o médico suíço Friedrich Miescher (1844-1895) fez uma descoberta curiosa: o núcleo de cada célula contém uma substância complexa mas desconhecida, o seu composto químico ficou conhecido como ácido nucleico. Posteriormente descobriu-se que há dois ácidos nucleicos: um deles passou a ser conhecido por *ácido ribonucleico* ou ARN e o outro por *ácido desoxirribonucleico* ou ADN.

O ADN é uma molécula que reproduz o código genético, é responsável pela transmissão das características hereditárias de cada espécie, quer seja nas plantas, nos animais ou nos microrganismos. A molécula do ADN é formada por fosfato e açúcar e por sequências de quatro bases nitrogenadas: adenina (A), timina (T), citosina (C) e guanina (G), ligadas por pontes de hidrogénio, formando uma dupla hélice. A estrutura química das bases é tal que cada uma só pode emparelhar com o mesmo parceiro. A adenina forma sempre um par com a timina e a citosina com a guanina. Estes quatro pares: *AT*, *TA*, *CG* e *GC*, formam as 'letras' do código do ADN. Pode ver-se na figura 8.1 a representação esquemática do ADN.

Cada gene é um segmento de ADN que contém uma fórmula química da composição de uma proteína particular ou de ARN. O genoma humano contém de 25000 a 30000 genes.

Cerca de 98% do ADN humano contém regiões não codificadas.

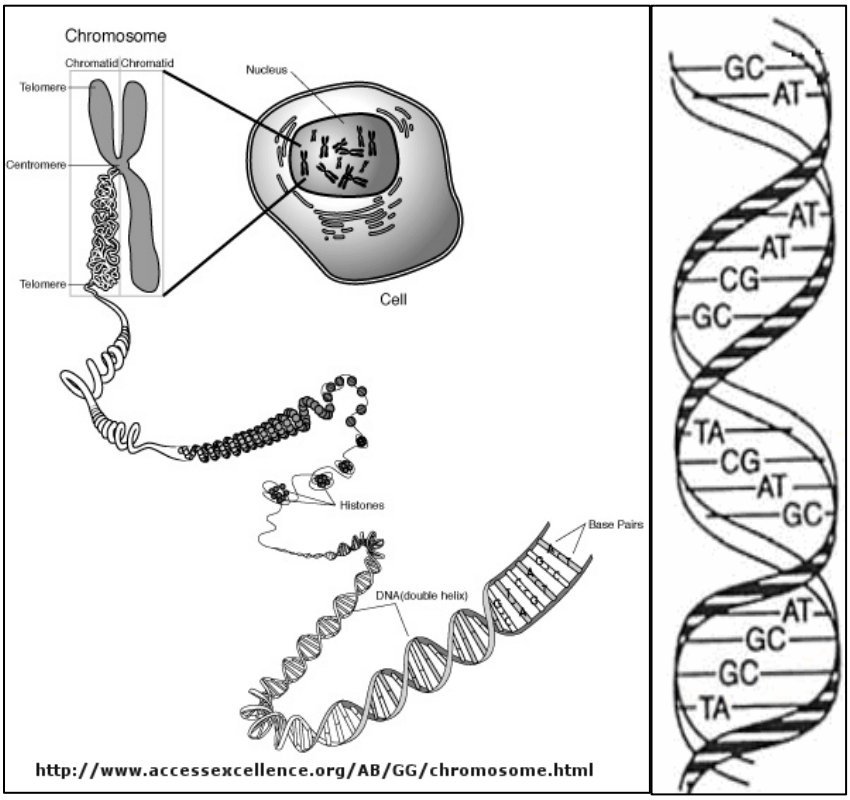


Fig. 8.1 - Representação esquemática do *ADN*

O ARN tem uma composição muito semelhante à do ADN, contudo apresenta algumas diferenças. Tem uma estrutura geralmente em cadeia simples e é formado por moléculas de dimensões muito inferiores às do ADN. Localiza-se no núcleo das células e no citoplasma[7], participando na síntese de ADN quando as células se multiplicam. A quantidade de ARN é variável de célula para célula de acordo com a actividade celular.

Existem diferentes tipos de ARN, o que nos interessa para este estudo é o ARN mensageiro (ARNm). Este forma-se no núcleo da célula e transporta as informações do código genético do ADN para o citoplasma, determinando as sequências dos aminoácidos na construção das proteínas.

Nos últimos anos, a genética tem sido bastante estudada. Esta área pode ter diversas aplicações, como a selecção de espécies, a criação de medicamentos mais eficazes, a melhoria de diagnóstico e tratamento de várias doenças.

Tem-se tentado descobrir, para além da sequência dos genes, como é que estes interagem no controlo do metabolismo. Estuda-se para isso, nos genes, a variação no

[7] É tudo o que compreende a célula menos o núcleo

tempo da produção de ARNm para codificar uma proteína, ou seja, a expressão dos genes. Quando um gene recebe a ordem para se expressar, o segmento correspondente (da dupla hélice) abre e é feita uma cópia dessa informação numa única cadeia, *ARN*m. O ARNm deixa o núcleo da célula e no citoplasma é feita a tradução. A cada três bases é associado um aminoácido. O conjunto desses aminoácidos é a proteína sintetizada, ou seja, é a expressão desse gene.

Para se fazer a leitura da expressão dos genes são utilizadas micropistas ("microarrays") de ADN que são instrumentos que medem simultaneamente a concentração de milhares de moléculas de ARNm.

Cada cadeia simples de ADN, copiada a partir de um segmento de gene particular é ligada à micropista. O conjunto de ARNm produzido por um tipo celular é marcado e é designado por "sonda". Estas sondas vão ligar-se com muita afinidade ao ADN correspondente que se encontra na micropista. Esta ligação torna-se fluorescente quando é iluminada a laser.

A intensidade da luz libertada por cada molécula de ARNm é medida. Estes valores correspondem a entradas numa matriz, com m linhas e n colunas. Cada linha representa a expressão de um gene em n experiências.

A matriz mostra a expressão dos genes (a produção de ARNm) num determinado momento de vida da célula (em etapas consecutivas de crescimento das células, por exemplo).

Pode acontecer que nalguns sítios da sequência do ADN faltem dados, por exemplo devido ao erro de leitura da máquina. Por vezes, também é dispendioso ou torna-se demorado repetir a experiência, então os cientistas têm vindo a tentar descobrir métodos para obter os dados em falta.

Os métodos mais comuns para fazer a reconstrução são [25]: método da substituição por zero, média da soma das linhas, métodos de análise de *clusters* e SVD. Nestes métodos a reconstrução dos dados que faltam é feita independentemente, isto é, a estimação de cada entrada não influencia a estimação de outras entradas.

Em [26] é sugerido um novo método em que a estimação dos dados em falta é feita em simultâneo.

Neste trabalho descreve-se um método que faz uso da SVD.

8.2 – A matriz de expressão dos genes

Seja $E \in \mathbb{R}^{m \times n}$, com $m >> n$, a matriz de expressão dos genes.

$$E = \begin{bmatrix} g_{11} & g_{12} & \cdots & g_{1n} \\ g_{21} & g_{22} & \cdots & g_{2n} \\ \vdots & \vdots & \ddots & \\ g_{m1} & g_{m2} & \cdots & g_{mn} \end{bmatrix} \tag{8.1}$$

O vector $g_i^T = [g_{i1} \quad g_{i2} \quad \ldots \quad g_{in}]$, $(i = 1, \ldots, m)$, corresponde aos níveis de expressão para o *i*-ésimo gene nas *n* experiências.

Consideremos a decomposição em valores singulares da matriz *E*.

$$E_{m \times n} = U_{m \times m} \, \Sigma_{m \times n} \, V_{n \times n}^T \tag{8.2}$$

Às colunas da matriz *V*, $v_1, \cdots, v_n$, dá-se o nome de ***genes próprios*** sendo os valores singulares de *E*, $\sigma_1, \cdots, \sigma_m$, as respectivas ***expressões próprias***.

Através de várias experiências, feitas em bases de dados, verificou-se que apenas alguns genes próprios são necessários para obter toda a expressão dos genes. Esse número de genes significativos nunca excede $\frac{n}{2}$, portanto para obter toda a expressão dos genes é suficiente considerar menos de metade do número de experiências [25].

Existem vários métodos para estimar o número de genes próprios significativos. De seguida descreve-se um deles, conhecido por critério da fracção.

8.3 – O critério da fracção

Considere-se

$$p_q = \frac{\sigma_q^{\,2}}{\sum_{t=1}^{n} \sigma_t^{\,2}}, \tag{8.3}$$

onde $q = 1, \ldots, n$.

O valor p_q representa a contribuição da expressão do *q*-ésimo gene próprio. Escolhem-se *l* genes próprios que contribuam mais de 90% de toda a expressão.

8.4 – Reconstrução da matriz de expressão de genes através da SVD

De seguida descreve-se um método para a reconstrução da matriz *E* que faz uso da decomposição em valores singulares.

Começa-se por considerar que m' é o número de linhas de *E* em que são conhecidas todas as entradas. Seguidamente reorganizam-se as linhas na matriz escrevendo primeiramente aquelas em que não faltam entradas.

$$E = \left[\begin{array}{cccccc} g_{11} & \cdots & g_{1n'} & g_{1(n'+1)} & \cdots & g_{1n} \\ g_{21} & \cdots & g_{2n'} & g_{2(n'+1)} & \cdots & g_{2n} \\ & \ddots & & & \ddots & \\ \vdots & \vdots & \vdots & \vdots & & \vdots \\ g_{m'1} & \cdots & g_{m'n'} & g_{m'(n'+1)} & \cdots & g_{m'n} \\ \hline g_{(m'+1)1} & \cdots & g_{(m'+1)n'} & g_{(m'+1)(n'+1)} & \cdots & g_{(m'+1)n} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ g_{m1} & \cdots & g_{mn'} & g_{m(n'+1)} & \cdots & g_{mn} \end{array}\right] \begin{array}{l} \\ \\ \\ \\ \\ m' \\ \\ \\ \\ m-m' \end{array} . \qquad (8.4)$$

Considera-se agora a submatriz $F \in \mathbb{R}^{m' \times n}$ que se obtém a partir de *E* eliminando as linhas em que faltam entradas.

Neste método, se falta uma entrada no gene g_i^T, esta escreve-se como combinação linear das entradas correspondentes dos *l* genes próprios significativos de F.

Primeiramente considera-se a decomposição em valores singulares da matriz F, na forma

$$F_{m' \times n} = U^1_{m' \times n} \, \Sigma^1_{n \times n} \, V^1_{n \times n} . \qquad (8.5)$$

Encontram-se depois os *l* genes próprios significativos através do critério da fracção.

Seja o gene g_i^T com algumas entradas desconhecidas,

$$g_i^T = \begin{bmatrix} g_{i1} & g_{i2} & \cdots & g_{in'} & g_{i(n'+1)} & \cdots & g_{in} \end{bmatrix} . \tag{8.6}$$

Vamos escrever este gene como combinação linear dos genes próprios significativos de F, ou seja, das primeiras *l* colunas de V^1,

$$g_i^T = \sum_{j=1}^{l} a_i v_j^1 , \tag{8.7}$$

ou mais simplesmente,

$$g^T = V^1 a . \tag{8.8}$$

Considere-se agora o vector $\hat{g}_i^T$ que corresponde ao gene g_i^T tendo em conta apenas as entradas conhecidas. Então $\hat{g}_i^T$ é a projecção de g_i^T num espaço de dimensão igual ao número de entradas conhecidas de g_i^T. É com o auxílio destes vectores, $\hat{g}_i^T$, que se calculam os coeficientes, a_i, da combinação linear (8.7).

Cada um dos $\hat{g}_i^T$ pode escrever-se como combinação linear dos *l* genes próprios $\hat{v}_j$. Os vectores $\hat{v}_j$ correspondem aos genes próprios calculados em (8.5), suprimindo as entradas cuja posição é a mesma das entradas desconhecidas no gene g_i^T. Tem-se assim que

$$\hat{g}_i^T = \sum_{j=1}^{l} a_j \hat{v}_j , \tag{8.9}$$

$$\hat{g}_i^T = a_1 \hat{v}_1 + a_2 \hat{v}_2 + \cdots + a_l \hat{v}_l , \tag{8.10}$$

ou seja,

$$\hat{g}_i^T = \begin{bmatrix} \hat{v}_1 & \hat{v}_2 & \ldots & \hat{v}_l \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_l \end{bmatrix} . \tag{8.11}$$

Para cada gene com entradas em falta, temos de encontrar a solução para o sistema de equações,

$$\hat{g}^T = \hat{V}_{k\times l} a_{l\times 1}, \tag{8.12}$$

onde k é o número de entradas conhecidas do gene g_i^T.

Analisemos a classificação do sistema (8.12), para o caso de $\hat{V}$ ter característica completa.

Se $k = l$, o sistema é determinado e $a = \hat{V}^{-1}\hat{g}^T$.

Se $k > l$, o sistema é sobredeterminado, $\hat{g}^T \notin R(\hat{V})$ e este sistema não tem solução. Nesta situação é necessário encontrar uma solução aproximada no sentido dos mínimos quadrados, $\hat{a}$ tal que

$$\left\| \hat{g}^T - \hat{V}\,\hat{a} \right\|_2 = \min_a \left\| \hat{g}^T - \hat{V}a \right\|_2 . \tag{8.13}$$

Para se determinar os coeficientes é agora necessário multiplicar ambos os membros de (8.13) pela pseudo inversa de $\hat{V}$,

$$\hat{V}^{\dagger}_{l\times k}\hat{g}^T = \hat{V}^{\dagger}_{l\times k}\hat{V}_{k\times l} a_{l\times 1}. \tag{8.14}$$

Encontram-se assim os coeficientes da combinação linear

$$a = \hat{V}^{\dagger}\hat{g}^T. \tag{8.15}$$

No caso de $k < l$, o sistema é indeterminado, tendo por isso infinitas soluções. Temos de procurar a solução de norma mínima.

Verifica-se que se $\hat{V}$ tem característica incompleta, isto é, se $car(\hat{V}) < \min(k,l)$, o sistema pode ser indeterminado e por isso ter infinitas soluções. Neste caso, procura-se também a solução de norma mínima.

Pode agora substituir-se o resultado acima mencionado em (8.9)

$$g^T = V^1\hat{V}^{\dagger}\hat{g}^T . \tag{8.16}$$

Desta forma, para determinar a entrada *p*, em falta no gene g_i^T , utiliza-se a seguinte fórmula

$$g_{ip}^T = \sum_{j=1}^{l} v_{jp} v_j^{\dagger} g_i^T \quad . \tag{8.17}$$

Depois de calculadas as entradas que faltam nos genes são substituídas nas respectivas posições da matriz *E*.

Elaborámos um programa em MATLAB, utilizando dados disponíveis em http://genome-www.stanford.edu/SVD/ e que fazem parte de [27]. Este programa estima as entradas em falta em cada gene. Pode ser consultado no Anexo B.

Os dados atrás referidos contêm a expressão de 6113 genes em 24 experiências. Calculámos as entradas em falta na matriz de expressão dos genes aplicando os procedimentos anteriormente descritos. Verificámos é suficiente considerarmos apenas 5 genes próprios, pois estes contribuem mais de 95% para toda a expressão.

Na impossibilidade prática de apresentar a totalidade da matriz (6113 linhas por 24 colunas) apresentamos apenas um bloco, com falta de dados, já com a troca de linhas efectuada e a respectiva reconstrução, utilizando a decomposição em valores singulares.

1.07	0.70	1.79	0.47	1.50	0.80	1.72	1.29	2.50	0.47	0.91	0.69
0.31	0.44	0.69	0.93	1.38	1.68	1.70	1.11	1.95	0.83	0.56	0.49
1.00	1.14	1.40	0.54	1.20	1.13	1.78	1.27	1.73	0.43	0.84	0.80
1.43	0.54	1.60	0.49	1.57	1.33	1.59	1.57	2.22	0.44	0.80	0.51
1.81	1.26	0.53	1.13	1.75	0.93	1.42	0.65	1.24	1.09	0.97	0.94
1.09	0.98	1.20	0.84	1.45	1.02	1.21	1.18	1.16	0.91	0.82	0.96
1.08	0.56	0.68	1.25	3.01	2.72	5.09	3.40	4.85	2.04	1.62	1.06
0.75	0.80	0.46	0.85	0.88	0.83	1.05	1.07	0.78	0.72	1.00	0.86
1.46	1.13	1.07	0.83	1.14	0.94	1.36	2.57	2.04	2.13	2.26	2.11
1.43	0.74	1.25		1.29	0.69	1.21	1.45	2.00	0.84	0.95	0.93
1.20	1.18	1.69		1.53	0.65	1.97	1.02		0.62	0.87	0.50
0.74	1.02	1.37		1.34	0.75	0.99	1.37	1.02	0.76	0.73	0.69
1.17	1.20	1.80		1.37	0.69	1.71	1.06		0.49	0.87	0.50
1.07	1.06	1.79		1.40	0.62	1.89	1.10	2.88	0.74	1.15	0.48
1.20	0.98	1.13		1.47	0.82	1.58	2.80		0.58	0.87	0.50
1.43	1.07	1.96		1.11	0.85	1.41	0.89	3.00	0.73	1.04	0.47
1.05	1.10	1.85		1.20	0.68	1.23	1.61		0.13	0.87	0.50
1.13	0.97	1.17		1.48	0.88	1.57	0.86	1.30	0.92	0.87	0.50
1.23	0.97	0.88		1.16	1.23	1.11	2.29	1.33	1.82	1.14	1.22
1.44	1.07	2.24		1.20	1.06	1.46	1.60		0.53	0.87	0.50

⇨

1.07	0.70	1.79	0.47	1.50	0.80	1.72	1.29	2.50	0.47	0.91	0.69
0.31	0.44	0.69	0.93	1.38	1.68	1.70	1.11	1.95	0.83	0.56	0.49
1.00	1.14	1.40	0.54	1.20	1.13	1.78	1.27	1.73	0.43	0.84	0.80
1.43	0.54	1.60	0.49	1.57	1.33	1.59	1.57	2.22	0.44	0.80	0.51
1.81	1.26	0.53	1.13	1.75	0.93	1.42	0.65	1.24	1.09	0.97	0.94
1.09	0.98	1.20	0.84	1.45	1.02	1.21	1.18	1.16	0.91	0.82	0.96
1.08	0.56	0.68	1.25	3.01	2.72	5.09	3.40	4.85	2.04	1.62	1.06
0.75	0.80	0.46	0.85	0.88	0.83	1.05	1.07	0.78	0.72	1.00	0.86
1.46	1.13	1.07	0.83	1.14	0.94	1.36	2.57	2.04	2.13	2.26	2.11
1.43	0.74	1.25	0.74	1.29	0.69	1.21	1.45	2.00	0.84	0.95	0.93
1.20	1.18	1.69	0.72	1.53	0.65	1.97	1.02	1.78	0.62	0.87	0.50
0.74	1.02	1.37	0.62	1.34	0.75	0.99	1.37	1.02	0.76	0.73	0.69
1.17	1.20	1.80	0.70	1.37	0.69	1.71	1.06	1.61	0.49	0.87	0.50
1.07	1.06	1.79	0.77	1.40	0.62	1.89	1.10	2.88	0.74	1.15	0.48
1.20	0.98	1.13	0.78	1.47	0.82	1.58	2.80	2.04	0.58	0.87	0.50
1.43	1.07	1.96	0.78	1.11	0.85	1.41	0.89	3.00	0.73	1.04	0.47
1.05	1.10	1.85	0.67	1.20	0.68	1.23	1.61	1.43	0.13	0.87	0.50
1.13	0.97	1.17	0.64	1.48	0.88	1.57	0.86	1.30	0.92	0.87	0.50
1.23	0.97	0.88	0.84	1.16	1.23	1.11	2.29	1.33	1.82	1.14	1.22
1.44	1.07	2.24	0.77	1.20	1.06	1.46	1.60	1.68	0.53	0.87	0.50

Concluímos que os valores calculados pelo método atrás descrito parecem ser fiáveis, já que são da mesma ordem de grandeza dos restantes, obtidos nas experiências.

Referências

[1]- N. Muller, L. Magaia e B. M. Herbst, *Singular Value Decomposition, Eigenfaces, and 3D Reconstructions*, Siam Review, vol. 46, nº 3 (2004), pp. 518-545.

[2]- G. W. Stewart, *On the Early History of the Singular Value Decomposition*, Siam Review, vol. 35, nº 4 (1993), pp. 551-566.

[3]- C. F. Van Loan, *Generalizing the singular value decomposition*, Siam Review, vol. 13, nº1 (1976), pp. 76-81.

[4]- G.C. Pauge e M.A. Saunders, *Towards a generalized singular value decomposition*, vol. 18, nº 3 (1981), pp. 398-405.

[5]- O.Alter, P.O. Brown e D. Botstein, *Generalized singular value decomposition for comparative analysis of genome-scale expression data sets of two different organisms*, PNAS, vol.100, nº 6 (2003).

[6]- G. H. Golub e C. F. Van Loan, *Matrix Computations* – 2ª Edição, Academic Press, Boston, 1990.

[7]- F. Buffo e A. Verdiell, *La decomposición en valores singulares: un enfoque geométrico y aplicaciones*, Departamento de Matemática, Universidad Nacional del Sur, Argentina.

[8]- Kirby, M. e Sirovich L., *Application of the karhunen-loève procedure for the characterization of human faces*. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol.12, nº1 (1990), pp. 103-108.

[9]- M. Turk and A. Pentland, *Face recognition using eigenfaces*, Proc. IEEE Conference on Computer Vision and Pattern Recognition, Maui, Hawaii, 1991.

[10]- N. Muller e B. Herbst, *Bulding a representative training set based on eigenimages*, Department of Applied Mathematics University of Stellenbosch, South Africa.

[11]- C. Leung, *Real Time Face Recognition*, University of Queensland, 2001.

[12]- I. T. Jolliffe, *Principal Component Analysis* – 2ª Edição, Springer Series in Statistics, New York, 2002.

[13]- N. Muller, *Facial Recognition, Eigenfaces and Synthetic Discriminant Functions*, Thesis for the degree of doctor of Philosophy, University of Stellenbosch, 2000.

[14]- N. Muller, *Image Recognition Using Eigenpicture Technique* – M.Sc.Thesis, University of Cape Town, 1997.

[15]- M. Alan Turk, *Interactive-Time Vision: Face Recognition as a Visual Behavoir*, Thesis for the degree of doctor of Philosophy,Massachussetts Institute of Technology, 1991.

[16]- S. Ullman, "The interpretation of visual motion", MIT Press, Cambridge MA, EUA, 1979.

[17]- C. Tomasi e T. Kanade, "Shape and motion from image streams under orthography: a factorization method", *International Journal of Computer Vision*, vol. 9, nº 2 (1992), pp.137-154,.

[18]- C. J. Poelman e T. Kanade, "A paraperspective factorization method for shape and motion recovery", IEEE *Trans. on Pattern Analysis and Machine Intelligence*, vol. 19, nº 3 (1997).

[19]- J. Costeira e T. Kanade, "A Multibody Factorization Method for Motion Analysis",*International Journal of Computer Vision*, vol. 29, nº 3 (1998).

[20]- M.W. Berry, S.T. Dumais, e G.W.O'Brien, "*Using linear algebra for intelligent information retrieval*", SIAM Review, vol. 37, nº 4 (1995), pp. 573-595.

[21]- M.W. Berry, Z. Drmac, e E.R. Jessup, "*Matrices, Vector Spaces, and Information Retrieval*", SIAM Review, vol. 41, nº 2 (1999), pp. 335-362.

[22]- S. Deerwester, S. Dumais, G. Furnas, T. Landauer, e R. Harshman, *Indexing by latent semantic analysis*, J. American Society for Information Science, 41 (1990), pp. 391-407,.

[23]- C. Moler, D. Morrison, *Singular value analysis of cryptograms*. American Mathematical Monthly ,nº 90 (1983), pp. 78-87.

[24]- B. R.Schatz, *Automated analysis of cryptograms*, Cryptologia nº 1 (1977) pp.116-142.

[25]- O. Troyanskaya, M. Cantor, G. Sherlock, P. Brown, T. Hastie, R. Tibshirani, D. Botstein e R. B. Altman, *Missing value estimation for DNA microarrays*, Bioinformatics, vol.17 (2001), pp. 520-525.

[26]- S. Friedland, A. Niknejad e L. Chihara, *A simultaneous Reconstrution of Missing Data in DNA Microarrays*, Institute for Mathematics and its Applications, 2005.

[27]- O. Alter, P. O. Brown e D. Botstein, *Processing and modelling genome-wide expression data using Singular Value Decomposition*, SPIE, vol.4266 (2001), pp. 171-186.

Bibliografia adicional

G. Gregorcic, *The singular value decomposition and the pseudoinverse*, University College Cork, Ireland, 2001.

D. Kalman, *A singularly valuable decomposition: the SVD of a matrix*, The American University, Washinghton, 2002.

S. Friedland, *A new approach to generalized singular value decomposition*, IMA preprint series, Minnesota, 2004.

R. A. Horn e C R. Johnson, *Matrix analysis*. Cambridge: Cambridge University Press, 1990.

F. R. Gantmacher, *The Theory of Matrices*, vol. 2, Chelsea, New York, 1989.

W. Zhao e R. Chellapa, *Face Processing: Advanced Modeling and Methods*, Academic Press, 2005.

E. Trucco e A. Verri, *Introductory techniques for 3-D Computer Vision*, Prentice Hall, Inc., 1998.

C. Tomasi e T. Kanade, *Detection and Tracking of Point Features*, Technical Report, 1991.

T. Morita e T. Kanade, *A sequential Method for Recovering Shape and Motion from Images Streams*, School of Computer Science, Pittsburgh, 1994.

C. Tomasi e T. Kanade, *Point Features in 3D Motion*, School of Computer Science, Pittsburgh, 1991.

N. Sarris, D. Makris e M. Strintzis, *Three dimensional model based rigid tracking of a human head*, Information Processing laboratory, Aristotle University of Thessaloniki.

J. G. Martin, *Cryptanalysis using singular value decomposition*, University of Georgia, 2005.

O. Alter, P. O. Brown e D. Botstein, *Singular value decomposition for genome-wide expression data processing and modelling*, PNAS, vol.97, nº 18, 2000.

A. Niknejad, *Application of Singular Value Decomposition to DNA Microarray*, Thesis, University of Illinois, Chicago, 2005.

ANEXO A

Neste anexo, constroi-se, em detalhe, o sistema (5.30).

Seja $A_R A_R^T = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{12} & a_{22} & a_{23} & a_{24} \\ a_{13} & a_{23} & a_{33} & a_{34} \\ a_{14} & a_{24} & a_{34} & a_{44} \end{bmatrix}$.

Na equação (5.27) temos $\begin{bmatrix} i_x & i_y & i_z & t_x \end{bmatrix} \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{12} & a_{22} & a_{23} & a_{24} \\ a_{13} & a_{23} & a_{33} & a_{34} \\ a_{14} & a_{24} & a_{34} & a_{44} \end{bmatrix} \begin{bmatrix} i_x \\ i_y \\ i_z \\ t_x \end{bmatrix} = 1$, ou seja,

$$\begin{bmatrix} i_x i_x & i_y i_x + i_x i_y & i_z i_x + i_x i_z & t_x i_x + i_x t_x & i_y i_y & i_z i_y + i_y i_z & t_x i_y + i_y t_x & i_z i_z & t_x i_z + i_z t_x & t_x t_x \end{bmatrix} \begin{bmatrix} a_{11} \\ a_{12} \\ a_{13} \\ a_{14} \\ a_{22} \\ a_{23} \\ a_{24} \\ a_{33} \\ a_{34} \\ a_{44} \end{bmatrix} = 1 \cdot$$

A equação (5.28) é análoga à anterior.

Na equação (5.29) temos $\begin{bmatrix} i_x & i_y & i_z & t_x \end{bmatrix} \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{12} & a_{22} & a_{23} & a_{24} \\ a_{13} & a_{23} & a_{33} & a_{34} \\ a_{14} & a_{24} & a_{34} & a_{44} \end{bmatrix} \begin{bmatrix} j_x \\ j_y \\ j_z \\ t_y \end{bmatrix} = 0$, ou seja,

$$\begin{bmatrix} i_x j_x & i_y j_x + i_x j_y & i_z j_x + i_x j_z & t_x j_x + i_x t_y & i_y j_y & i_z j_y + i_y j_z & t_x j_y + i_y t_y & i_z j_z & t_x j_z + i_z t_y & t_x t_y \end{bmatrix} \begin{bmatrix} a_{11} \\ a_{12} \\ a_{13} \\ a_{14} \\ a_{22} \\ a_{23} \\ a_{24} \\ a_{33} \\ a_{34} \\ a_{44} \end{bmatrix} = 0 \cdot$$

Obtemos assim o sistema $B_{3f\times 10}\, a_{10\times 1} = C_{3f\times 1}$, definido em (5.30) pelas equações (5.27 a 5.29), em que B é a matriz

$$\begin{bmatrix}
(i_x\, i_x)_1 & (i_y i_x + i_x i_y)_1 & (i_z i_x + i_x i_z)_1 & (t_x i_x + i_x t_x)_1 & (i_y\, i_y)_1 & (i_z i_y + i_y i_z)_1 & (t_x i_y + i_y t_x)_1 & (i_z\, i_z)_1 & (t_x i_z + i_z t_x)_1 & (t_x t_x)_1 \\
(i_x\, i_x)_f & (i_y i_x + i_x i_y)_f & (i_z i_x + i_x i_z)_f & (t_x i_x + i_x i_x)_f & (i_y\, i_y)_f & (i_z i_y + i_y i_z)_f & (t_x i_y + i_y t_x)_f & (i_z\, i_z)_f & (t_x i_z + i_z t_x)_f & (t_x t_x)_f \\
(j_x\, j_x)_1 & (j_y j_x + j_x j_y)_1 & (j_z j_x + j_x j_z)_1 & (t_x j_x + j_x t_x)_1 & (j_y\, j_y)_1 & (j_z j_y + j_y j_z)_1 & (t_x j_y + j_y t_x)_1 & (j_z\, j_z)_1 & (t_x j_z + j_z t_x)_1 & (t_x t_x)_1 \\
(j_x\, j_x)_f & (j_y j_x + j_x j_y)_f & (j_z j_x + j_x j_z)_f & (t_x j_x + j_x t_x)_f & (j_y\, j_y)_f & (j_z j_y + j_y j_z)_f & (t_x j_y + j_y t_x)_f & (j_z\, j_z)_f & (t_x j_z + j_z t_x)_f & (t_x t_x)_f \\
(i_x\, j_x)_1 & (i_y j_x + i_x j_y)_1 & (i_z j_x + i_x j_z)_1 & (t_x j_x + i_x t_y)_1 & (i_y\, j_y)_1 & (i_z j_y + i_y j_z)_1 & (t_x j_y + i_y t_y)_1 & (i_z\, j_z)_1 & (t_x j_z + i_z t_y)_1 & (t_x t_y)_1 \\
(i_x\, j_x)_f & (i_y j_x + i_x j_y)_f & (i_z j_x + i_x j_z)_f & (t_x j_x + i_x t_y)_f & (i_y\, j_y)_f & (i_z j_y + i_y j_z)_f & (t_x j_y + i_y t_y)_f & (i_z\, j_z)_f & (t_x j_z + i_z t_y)_f & (t_x t_y)_f
\end{bmatrix}$$

$$a = \begin{bmatrix} a_{11} \\ a_{12} \\ a_{13} \\ a_{14} \\ a_{22} \\ a_{23} \\ a_{24} \\ a_{33} \\ a_{34} \\ a_{44} \end{bmatrix}$$

e

$$C = \begin{bmatrix} 1 \\ \vdots \\ 1 \\ \\ 1 \\ \vdots \\ 1 \\ \\ 0 \\ \vdots \\ 0 \end{bmatrix} \quad .$$

ANEXO B

Programa para comprimir imagens por blocos

```
%Este programa divide uma imagem em 5 blocos e comprime cada um desses blocos de acordo com a
%característica pretendida.

    II=zeros([1536 2048]);

% Lê imagem

    I=imread('biblioteca.jpg');
    I=rgb2gray(I);

%Mostra imagem

    imshow(I)
    pause
    close

    I=im2double(I);

    for i=1:1536

%Construçao Bloco A1
        for j=1:682
            A1(i,j)=I(i,j);
        end

%Construçao Bloco A5
        for j=1:684
            A5(i,j)=I(i,j+1364);
        end
    end

    for j=1:682

% Construçao bloco A2
        for i=1:512
            A2(i,j)=I(i,j+682);
       End

% Construçao bloco A3
        for i=1:513
            A3(i,j)=I(i+512,j+682);
       end

% Construçao bloco A4
        for i=1:511
            A4(i,j)=I(i+1025,j+682);
       end
   end

% Escolha da característica para os blocos
    k1=2;
    k2=40;

   [u1,s1,v1]=svd(A1,0);
```

```
% bloco A1
    A1=u1(:,1:k1)*s1(1:k1,1:k1)*v1(:,1:k1)';

    [u2,s2,v2]=svd(A2,0);

% bloco A2
    A2=u2(:,1:k1)*s2(1:k1,1:k1)*v2(:,1:k1)';

    [u3,s3,v3]=svd(A3,0);

% bloco A3
    A3=u3(:,1:k2)*s3(1:k2,1:k2)*v3(:,1:k2)';

    [u4,s4,v4]=svd(A4,0);

% bloco A4
    A4=u4(:,1:k1)*s4(1:k1,1:k1)*v4(:,1:k1)';

    [u5,s5,v5]=svd(A5,0);

% bloco A5
    A5=u5(:,1:k1)*s5(1:k1,1:k1)*v5(:,1:k1)';

%Forma imagem
    for i=1:1536
        for j=1:682
            II(i,j)=A1(i,j);
       end
       for j=1365:2048
            II(i,j)=A5(i,j-1364);
       end
    end
%
    for j=683:1364
        for i=1:512
            II(i,j)=A2(i,j-682);
        end
        for k=513:1025
            II(k,j)=A3(k-512,j-682);
        end
        for l=1026:1536
            II(l,j)=A4(l-1025,j-682);
        end
    end

% Mostra a imagem
   imshow(II)
   pause
   close
```

Programa para reconstrução de imagens

```
%Este programa reconstrói uma face do conjunto de teste considerando o número de faces próprias
%escolhidas, ou seja a dimensão do espaço de faces

% Lê as faces %
Matriz_Faces=zeros([256*256 150]);
```

```
for i=1:150

  nome=sprintf('face_treino/face%d.bmp',i);
  [Face]=im2double(imread(nome));%Lê cada uma das faces colocando-as numa matriz com entradas
%entre 0 e 1%
  Matriz_Faces(:,i)=Face(:);% Processo de vectorização: cada face passa a vector coluna%

end

% Calcula a média%
A=zeros(size(Matriz_Faces(:,1)));

for I=1:150
  A=A+Matriz_Faces(:,I);
end
A = A./150;

D2 = 0;
for I=1:150
  Matriz_Faces(:,I)=Matriz_Faces(:,I)-A; % Subtrai a média a cada face%
  D2 = D2 + norm(Matriz_Faces(:,I))^2;
end

% Calcula as faces próprias%
D2 = D2/150;
D = sqrt(D2);

Matriz_Faces = Matriz_Faces/(sqrt(150)*D);
%Matriz_Faces = Matriz_Faces/(sqrt(150));
[U,S,V] = svd(Matriz_Faces,0);

 k=inputdlg('Introduza o numero', 'Qual e a 1ª face própria a considerar?');
 k=str2num(k{1});

 k1=inputdlg('Introduza o numero', 'Qual e a última face própria a considerar?');
 k1=str2num(k1{1});

Ei = reshape(U(:,i),256,256);

%Le faces teste%
Matriz_FaceT=zeros([256*256 28]);

for I=151:178

  nomes=sprintf('face_teste/face%d.bmp',I);
  [FaceT]=im2double(imread(nomes));
  Matriz_FaceT(:,I)=FaceT(:);

end

i=150+menu('Escolha a face do conjunto de teste', 'Face 151',… ');
F=Matriz_FaceT(:,i);%F e a face a ser projectada%
X = (F-A)/D;
U1=U(:,k:k1);
coef = U1'*X;%coef- coeficientes de projecção
r = length(coef);

Un = U1(:,1:r);

F_recon = Un*coef*D + A;
```

```
Fr = reshape(F_recon,256,256);
nomes=sprintf('face_teste/face%d.bmp',i);
[FaceT]=im2double(imread(nomes));

subplot(2,2,1);axis off; imshow([FaceT])
z=text(75,-15,'Face Teste');
set(z,'FontSize',10,'Color','r')

subplot(2,2,2);axis off; imshow([Fr])
t=text(45,-15,'Face Reconstruida');
set(t,'FontSize',10,'Color','r')

subplot(2,2,3.5);axis off; imshow([FaceT] - [Fr])
t=text(95,-15,'Erro');
set(t,'FontSize',10,'Color','r')

pause(10);
Perg='Quer ver os coeficientes da projecção ?';
titulo= 'Pergunta';
b1='Sim';
b2='Nao';
default='Sim';
t=questdlg(Perg,titulo,b1,b2,default);
  if t=='Sim'; coef
  end

close (figure(1));
Perg='Quer fazer outra reconstrução da mesma face?';
titulo= 'Pergunta';
b1='Sim';
b2='Nao';
default='Sim';
t=questdlg(Perg,titulo,b1,b2,default);
  if t=='Sim'
    k2=inputdlg('Introduza o número', 'Qual e a 1ª face própria considerar?');
    k2=str2num(k2{1});
    k3=inputdlg('Introduza o número', 'Qual e a ultima face própria a considerar?');
    k3=str2num(k3{1});

    Perg='Quer ver as faces próprias?';
    titulo= 'Pergunta';
    b1='Sim';
    b2='Nao';
    default='Sim';
    r=questdlg(Perg,titulo,b1,b2,default);
    if r=='Sim'; figure(1);clf;
            for j = k2:k3
            Ej = reshape(U(:,j),256,256);
            imshow(Ej,[]);pause% pause - para mostrar uma a uma carregando em qualquer tecla%
            close (figure(1));
            end

    else
            for j = k2:k3
            Ej = reshape(U(:,j),256,256);
            end
    end


    X = (F-A)/D;
```

```
    U11=U(:,k2:k3);
    coef1 = U11'*X;%coef- coeficientes de projecção

    r = length(coef1);

    Un = U11(:,1:r);

    F_recon = Un*coef1*D + A;

    Fr1 = reshape(F_recon,256,256);
    nomes=sprintf('face_teste/face%d.bmp',i);
    [FaceT]=im2double(imread(nomes));

    subplot(2,3,1);axis off; imshow([FaceT])
    z=text(75,-15,'Face Teste');
    set(z,'FontSize',10,'Color','r')

    subplot(2,3,2);axis off; imshow([Fr])
    t=text(45,-15,'1ª Reconstrução ');
    set(t,'FontSize',10,'Color','r')

    subplot(2,3,3);axis off; imshow([Fr1])
    t=text(45,-15,'2ª Reconstrução');
    set(t,'FontSize',10,'Color','r')

    subplot(2,3,4.5);axis off; imshow([FaceT] - [Fr])
    t=text(70,-15,'Erro 1ª Rec.');
    set(t,'FontSize',10,'Color','r')

    subplot(2,3,5.5);axis off; imshow([FaceT] - [Fr1])
    t=text(70,-15,'Erro 2ª Rec.');
    set(t,'FontSize',10,'Color','r')

      Norma_2_erro_1reconst= norm([FaceT] - [Fr])
      Norma_Frob=norm([FaceT] - [Fr],'fro')
      coef1
      Norma_2_erro_2reconst= norm([FaceT] - [Fr1])

      Norma_Frob=norm([FaceT] - [Fr1],'fro')
    end

 end
```

Programa para a recuperação de dados em falta na matriz de expressão dos genes

%Este programa, dada a expressão de m genes em n experiências, faz a recuperação dos dados em falta através da SVD.

```
% Introduzir a matriz dos genes
A=[matriz dos genes];
[m,n]=size(A)

% troca linhas na matriz A, sendo as primeiras são as que tem os genes completos

for i=1:m
   ip(i)=1;
end
```

```
for i=1:m
   for j=1:n
      if A(i,j)==0
      ip(i)=0;
      end
      if ip(i)==0
      break
      end
   end
end
ip;
for i=1:m
   if ip(i)==0
      if i+1 <= m
         for jj=i+1:m
            if ip(jj)==1
               for j=1:n
               c=A(jj,j);
               A(jj,j)=A(i,j);
               A(i,j)=c;
               end
            id=ip(jj);
            ip(jj)=ip(i);
            ip(i)=id;
            break
            end
         end
      end
   end
end

ip;

% Construção da matriz A1 - linhas são genes em que não faltam entradas

k=sum(ip); % k é o número de linhas em que não faltam entradas
A1=A(1:k,:);
[m1,n1]=size(A1)

%Calculo da SVD de A1
[U,S,V]=svd(A1,0);

%critério da fracção

b=trace(S*S');

for i= 1:min(m1,n1)

   q(i)=(S(i,i)*S(i,i))/b

end
q % Vector em que cada entrada é a contribuição de cada gene próprio

C=cumsum(q);
for x=1:min(m1,n1)
   if C(x)>=0.95
     g=x;
     break
   end
end
```

```
g; % número de genes próprios a considerar

% Matriz Aproximada, com característica g
U1=U(:,1:g);
S1=S(1:g,1:g);
V1=V(:,1:g);

%linhas em que faltam entradas
B=A(k+1:m,:);
[b1,b2]=size(B);

for kk=k+1:m

     B1=A(kk,:);
     [i,j,B1]=find(A(kk,:));%B1  é  primeiro vector da matriz A(:,k+1) com as entradas nulas suprimidas

% em cada gene próprio tem de se suprimir as coordenadas suprimidas no gene
     b=find(A(kk,:)==0); % este é o vector que indica as posições que são suprimidas em B1
     c=size(b);
    [c1,c2]=size(c);
    V2=V1;
        for e=1:c(2)
            V2(b(e),:)=0;% Anula as coordenadas que estão nas mesmas posições das entradas que faltam
%nos genes
             [i,j,V3]=find(V2);% V3 é um vector com todas as entradas( não nulas) da matriz V1( uma
coluna após a outra)
        end

     V4=reshape(V3,[],g);% cada coluna desta matriz é um gene próprio sem as entradas nas posições que
%faltam no gene

     a=pinv(V4)*B1';

     D=B;
       for f=1:c(2)
           h=b(f);
          D(1,h)=V1(h,:)*a;
          A(kk,h)=D(1,h);% entradas que faltam no gene B1

       end

end
```