ANÁLISE COMPARATIVA ENTRE OS MÉTODOS DECOMPOSIÇÃO EM VALORES SINGULARES E ANÁLISE DE COMPONENTES PRINCIPAIS ENVOLVENDO MATRIZES ESPARSAS DE GRANDE PORTE

CLAUDIANE FONSECA RODRIGUES

ANÁLISE COMPARATIVA ENTRE OS MÉTODOS DECOMPOSIÇÃO EM VALORES SINGULARES E ANÁLISE DE COMPONENTES PRINCIPAIS ENVOLVENDO MATRIZES ESPARSAS DE GRANDE PORTE

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Minas Gerais. como requisito parcial para a obtenção do grau de Mestre em Ciência da Computação.

Orientador: Frederico Ferreira Campos, filho

Coorientador: Dorgival Olavo Guedes Neto

Belo Horizonte, Minas Gerais

Julho de 2011

© 2011, Claudiane Fonseca Rodrigues.
Todos os direitos reservados.

Rodrigues, Claudiane Fonseca
R696a Análise comparativa entre os métodos Decomposição em Valores Singulares e Análise de Componentes Principais envolvendo matrizes esparsas de grande porte / Claudiane Fonseca Rodrigues. — Belo Horizonte, Minas Gerais, 2011
xvi, 171 f. : il. ; 29cm

Dissertação (mestrado) — Universidade Federal de Minas Gerais.

Orientador: Frederico Ferreira Campos, filho
Coorientador: Dorgival Olavo Guedes Neto

1. Decomposição em Valores Singulares (SVD).
2. Análise de Componentes Principais (PCA).
3. Matrizes esparsas. 4. Classificação de dados.
I.Orientador. II. Coorientador. III. Título.

CDU 519.6*72 (043)

UNIVERSIDADE FEDERAL DE MINAS GERAIS
INSTITUTO DE CIÊNCIAS EXATAS
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

FOLHA DE APROVAÇÃO

Análise comparativa entre os métodos Decomposição em Valores Singulares e Análise de Componentes Principais envolvendo matrizes esparsas de grande porte

CLAUDIANE FONSECA RODRIGUES

Dissertação defendida e aprovada pela banca examinadora constituída pelos Senhores:

PROF. FREDERICO FERREIRA CAMPOS, FILHO - Orientador
Departamento de Ciência da Computação - UFMG

PROF. DORGIVAL OLAVO GUEDES NETO - Co-orientador
Departamento de Ciência da Computação - UFMG

PROF. BRAULIO ROBERTO GONÇALVES MARINHO COUTO
Centro Universitário de Belo Horizonte

PROF. LUIZ HENRIQUE DUCZMAL
Departamento de Estatística - UFMG

Belo Horizonte, 29 de julho de 2011.

À Deus,
Aos meus queridos pais, Zilda e Antônio.

Agradecimentos

À Deus pela vida,

Aos meus pais e familiares pela força e amor incondicional,

Aos meus queridos amigos pela compreensão da minha ausência,

Ao meu orientador Frederico Ferreira Campos, filho pela disposição,

Ao meu co-orientador Dorgival Olavo Guedes Neto pela trajetória,

À Fernando Mourão pelo apoio e colaboração,

À instituição de fomento CAPES/REUNI,

muito obrigada!

""Mestre não é aquele que ensina, mas aquele que, de repente, aprende...""

(Guimarães Rosa)

Abstract

The Singular Value Decomposition and Principal Component Analysis techniques are from different areas and have different purposes. Nevertheless, they are often confused. Moreover, apart from more theoretical works, few studies know which technique to use. Questions such as: can a less elaborate choice between the techniques degrade the quality of a task? and when to use each one? are neglected in the literature. In addition, the efficient manipulation and analysis of large volumes of data has become a computational challenge due to the high dimensionality and sparsity of data, which makes it important to use techniques that benefit both the performance and the quality of analysis. However, current studies do not compare the use of those techniques, especially in sparse matrices of high order. So our goal is to compare and find differences between the two techniques on data classification task.

Resumo

As técnicas Decomposição em Valores Singulares e Análise de Componentes Principais são oriundas de áreas diferentes e possuem objetivos distintos. Apesar disso, são comumente confundidas. Mais ainda, salvo em trabalhos mais teóricos, poucos estudos sabem quando preterir o uso de uma técnica frente a outra. Perguntas tais como: a escolha não elaborada entre as técnicas pode degradar a qualidade de uma tarefa? e quando utilizar cada uma delas? são negligenciadas na literatura. Além disso, a manipulação e análise eficientes de grandes volumes de dados é hoje um desafio computacional devido à alta dimensionalidade e esparsidade dos dados, o que torna relevante utilizar técnicas que beneficiem tanto o desempenho quanto a qualidade das análises. No entanto, os trabalhos atuais não comparam o uso das técnicas, principalmente, em matrizes esparsas de alta ordem. Assim, nosso trabalho consiste em comparar e verificar diferenças entre as duas técnicas sobre a tarefa de classificação de dados.

Lista de Figuras

Lista de Tabelas

Sumário

Capítulo 1

Introdução

É notória a crescente utilização de métodos da Álgebra Linear (AL) e da Estatística (ES) em variadas aplicações computacionais. Áreas como Mineração de Dados [10], Bioinformática [18] e [33], Redes Sociais [22], Processamento Digital de Imagens [30], Aprendizado de Máquina, Recuperação de Informação [4], dentre outras, vêm encontrando nesses métodos uma maneira eficaz de abordar alguns de seus principais problemas. Por exemplo, alguns dos mais eficazes recomendadores [21] e classificadores [35] existentes são derivados de métodos bem estabelecidos na AL. A importante ascendência da AL e ES em Computação, pode ser explicada, pela robustez com a qual seus métodos abordam problemas computacionais comuns a diversos domínios, bem como a elegante formalização matemática provida aos mesmos, por meio de seus conceitos.

Dentre os vários métodos existentes em AL e ES que manipulam dados em formato matricial, destaca-se a Decomposição em Valores Singulares (*SVD - Singular Value Decomposition*) e a Análise de Componentes Principais (*PCA - Principal Component Analysis*), respectivamente, dado a popularidade e generalidade de aplicação de ambos. Todavia, apesar da grande popularidade de tais métodos para realizar a redução de dimensionalidade de bases em Mineração de Dados (MD), há pouco entendimento sobre o impacto de tal processo sobre as premissas assumidas pelos algoritmos tradicionais dessa área. Dessa forma, esse trabalho busca, principalmente, diferenciar e comparar esses dois métodos, SVD e PCA, no contexto de classificação de dados. Além disso, existe uma grande dificuldade por parte dos principais programas bem estabelecidos para aplicação desses métodos, ao se depararem com grandes bases esparsas de dados. Portanto, o objetivo desse trabalho pode ser sumarizado da seguinte forma:

Objetivo: Construir uma metodologia de auxílio à escolha entre as técnicas Decomposição em Valores Singulares (SVD) e Análise de Componentes Principais (PCA) envolvendo matrizes esparsas de grande porte sobre a tarefa de classificação de dados.

Para tanto, devido ao custo computacional e à complexidade matemática envolvida, é essencial para aplicação eficiente das técnicas o entendimento teórico e o estudo comparativo entre as principais e diferentes estruturas de armazenamento, juntamente com os respectivos algoritmos de operação sobre as mesmas. Assim, é possível viabilizar a comparação das duas técnicas em cenários que apresentam um grande volume de dados. Atualmente, os softwares mais utilizados para aplicação dessas técnicas como, o MATLAB, SCILAB e o R, por utilizarem linguagem interpretada, não respondem em tempo hábil quando submetidos a testes que envolvam grandes bases de dados.

Em síntese, na prática, esse trabalho foca na tarefa de classificação de dados por representar um cenário de comum utilização das técnicas, bem como de constante observação de desafios tais como o "Mal da Dimensionalidade" [31]. Análises sobre o algoritmo de classificação KNN (*K-nearest-neighbor*) em três bases reais bem estabelecidas permitiram o desenvolvimento de uma metodologia de comparação entre as técnicas sobre a tarefa de classificação, sendo capaz de levantar e verificar algumas hipóteses. Foram avaliadas questões interessantes a respeito da escolha do posto para aproximação de posto incompleto e de uma possível boa escolha do parâmetro k do algoritmo KNN ao se aplicar SVD ou PCA para redução da dimensionalidade. Aliás, o ganho pode ser observado tanto pela taxa de acerto quanto pelo desempenho obtido pelo algoritmo.

1.1 Referencial teórico

Dentre as mais importantes ou emergentes áreas interessadas nas técnicas SVD e PCA se encontram, por exemplo, Mineração de Dados, Recuperação de Informação, Redes Sociais e Bioinformática, que vêm buscando nessas técnicas, formas de aprimorar seus resultados. Por exemplo, em Mineração de Dados, a maioria dos trabalhos aplica SVD e PCA somente com o intuito de reduzir a dimensionalidade dos dados. Steinbach [31] limita a aplicação apenas para a tarefa de agrupamento. Segundo o autor, o motivo da limitação se encontra no efeito que o aumento da dimensionalidade pode causar sobre os cálculos de distância e similaridade, vindo a produzir informação não significativa. Isso se deve ao fato de que em espaços de dimensão elevada as distâncias entre os pon-

tos tornam-se relativamente uniformes e, nesse caso, a noção de vizinho mais próximo passa a não fazer sentido. Por outro lado, Lars Elden [10] apresenta de forma mais clara como várias técnicas da AL, inclusive a SVD, tem sido utilizada nesse contexto auxiliando a extração de informações úteis, principalmente, por se apresentar eficaz não somente na redução da dimensionalidade, mas por apresentar outros benefícios, como por exemplo, a remoção de ruídos (dados redundantes). Já no cenário de Recuperação de Informação, a Indexação Semântica Latente (LSI) [4] é considerada uma das aplicações mais importantes da SVD, a qual é utilizada pela empresa *Google*.

Em Redes Sociais, uma das áreas emergentes de maior impacto no atual estado da arte, mais especificamente no contexto de Sistemas de Recomendação, a SVD tem apresentado grande importância e desempenho frente a outras alternativas. Koren [21], vencedor do prêmio *NetFlix Prize*, competição pelo melhor algoritmo de filtragem colaborativa realizada pela empresa *NetFlix*, utilizou a SVD em seu trabalho. A crescente utilização e consolidação do método nesse cenário pode ser melhor observada em [22].

Já no contexto de Bioinformática, Alter [1] deixa claro que aplica a SVD com objetivo de realizar uma comparação significativa da expressão de diferentes genes em diferentes experimentos, além de reduzir e transformar o espaço de representação dos dados. Por outro lado, tanto Yeung [34] quanto Alkes [28], em trabalho publicado na revista *Nature*, optaram pela aplicação da PCA. Mais especificamente, Yeung [34] cita que seu objetivo é estudar a eficácia da PCA na captura da estrutura de grupos. No entanto, nenhum desses trabalhos se preocupa em entender o real motivo pelo qual essas técnicas atendem ou não aos objetivos propostos, qual a correlação entre as técnicas e seus respectivos domínios de análise e quais as vantagens obtidas pela aplicação de cada uma delas.

Outro cenário importante de aplicação das técnicas que envolvem AL e ES em Ciência da Computação é o Processamento Digital de Imagens. Por exemplo, Stegmann [30] busca realizar uma análise e segmentação de imagens faciais por meio da aplicação da Regressão de Componentes Principais (*PCR - Principal Component Regression*). Por outro lado, Muller [25] utiliza a SVD para fornecer uma representação eficiente de imagens faciais na busca de bons métodos para reconhecimento de faces. Além disso, o autor também aprensenta nesse mesmo trabalho, como a SVD pode ser utilizada para reconstruir objetos tridimensionais a partir de um fluxo de vídeo bidimensional.

1.2 Contribuições

As principais contribuições desse trabalho podem ser resumidas em:

- Apresentação teórica sobre as técnicas SVD e PCA e suas principais aplicações.
- Discussão sobre as estruturas de armazenamento para matrizes esparsas.
- Disponibilização das rotinas de multiplicação matriz-vetor e (matriz transposta)-vetor para cálculo da PCA mantendo a esparsidade da matriz.
- Proposta de critério para auxílio ao corte no número de valores singulares (SVD) e autovalores (PCA).
- Proposta de uma metodologia de suporte à escolha entre as técnicas SVD e PCA sobre a tarefa de classificação de dados.
- Identificação de tendência de estabilidade das métricas de qualidade eficácia e F1 do algoritmo KNN em relação à redução de dimensionalidade.

1.3 Organização da dissertação

Essa dissertação está organizada da seguinte forma: os Capítulos 2, 3 e 4 descrevem as decomposições Espectral, SVD e PCA de forma didática, acompanhadas por exemplos numéricos, com intuito de suavizar a complexidade matemática envolvida. O Capítulo 5 descreve as estruturas e algoritmos de multiplicação matriz-vetor para matrizes esparsas e o Capítulo 6 descreve, principalmente, as subrotinas implementadas para realização dos experimentos, bem como as otimizações algébricas realizadas para calcular a PCA, mantendo a esparsidade da matriz original. Por fim, o Capítulo 7 apresenta uma análise comparativa entre as técnicas SVD e PCA sobre a tarefa de classificação de dados, bem como a metodologia proposta e os resultados obtidos e o Capítulo 8 apresenta as conclusões, limitações e sugestões de trabalhos futuros.

Capítulo 2

Decomposição espectral

Várias são as aplicações computacionais atuais que envolvem o uso de matrizes. Por isso, o processo conhecido por decomposição ou fatoração de uma matriz, que consiste em representá-la por meio do produto de matrizes mais simples, com intuito de facilitar a resolução de determinado problema [7], é uma técnica muito explorada dentro da Álgebra Linear Numérica. Em particular, a Decomposição espectral, que é um tipo de fatoração de matrizes quadradas, consiste basicamente em representá-las por meio do conjunto de seus autovalores, designado por espectro, e de seus respectivos autovetores. Essa possibilidade de simplificação da representação, como por exemplo, ao se reduzir uma matriz à forma diagonal ou triangular, preservando características algébricas e geométricas, tem sido muito contemplada na modelagem de diversos problemas. Formalmente, a decomposição espectral de uma matriz $A_{n\times n}$ é representada por:

$$A = V\Lambda V^{-1} \rightarrow AV = V\Lambda, \tag{2.1}$$

$$\begin{bmatrix} & & \\ & A & \\ & & \end{bmatrix} \left[\begin{array}{c|c|c|c} & & & \\ v_1 & v_2 & \ldots & v_n \\ & & & \end{array}\right] = \left[\begin{array}{c|c|c|c} & & & \\ v_1 & v_2 & \ldots & v_n \\ & & & \end{array}\right] \begin{bmatrix} \lambda_1 & & & \\ & \lambda_2 & & \\ & & \ddots & \\ & & & \lambda_n \end{bmatrix},$$

onde $V_{n\times n}$ é uma matriz não singular (invertível) composta pelos autovetores de A e $\Lambda_{n\times n}$ é uma matriz diagonal composta pelos respectivos autovalores associados. Aliás, um escalar λ_i é dito autovalor de A, se existir um vetor não nulo $v_i \in \mathbb{R}^n$, dito autovetor de A, tal que Av_i é um múltiplo escalar de v_i. Portanto, $Av_i = \lambda_i v_i$ $(i = 1, 2, ..., n)$.

Além disso, os autovetores associados a cada autovalor λ_i da matriz A são soluções não nulas do sistema homogêneo $(A - I\lambda)v = 0$, onde I é uma matriz identidade. Ou seja, os autovalores são soluções da equação $\det(A - \lambda I) = 0$, designada por equação característica de A. Em outras palavras, os autovalores são os zeros do polinômio característico dado por (2.2). Segundo Trefethen [32], a principal consequência disso é que mesmo se a matriz A for real, alguns de seus autovalores podem ser complexos.

$$p(\lambda) = \det(A - \lambda I). \tag{2.2}$$

Por outro lado, o critério que define qual tipo de matriz pode ser representada pela forma (2.1) está diretamente associado aos conceitos de multiplicidade algébrica e geométrica de autovalores. Enquanto a multiplicidade algébrica $m_a(\lambda_i)$ representa o número de vezes em que o autovalor λ_i aparece como zero do polinômio característico (2.2), a geométrica $m_g(\lambda_i)$ representa a dimensão do subespaço vetorial de $\mathbb{R}^n$ associado a ele, designado por autoespaço, gerado juntamente com o vetor nulo, pelos respectivos autovetores linearmente independentes associados. Além disso, a principal relação entre essas duas multiplicidades é descrita pelo Teorema 2.1 [32, Teorema 24.4].

Teorema 2.1 *A multiplicidade geométrica de um autovalor λ é menor ou igual a sua multiplicidade algébrica.*

Entretanto, apesar da relação $1 \leq m_g(\lambda_i) \leq m_a(\lambda_i)$, onde o número de autovetores linearmente independentes associados a determinado autovalor é no mínimo um e no máximo a quantidade de vezes que em ele aparece como zero do polinômio característico (2.2), um autovalor λ_i cuja multiplicidade algébrica excede a multiplicidade geométrica é considerado um autovalor degenerado (*defective eigenvalue*). Consequentemente, uma matriz que possui um ou mais autovalores degenerados é considerada degenerada (*defective matrix*). Por sua vez, considerando que matrizes diagonais são não degeneradas (*nondefective matrix*) [32], a multiplicidade geométrica de todos os seus autovalores é necessariamente igual a multiplicidade algébrica dos mesmos.

Portanto, como declarado pelo Teorema 2.2 [32, Teorema 24.5], a classe de matrizes não degeneradas é a classe de matrizes diagonalizáveis, isto é, que possuem uma decomposição espectral do tipo (2.1). Nesse caso, com o Teorema 2.3 pode-se concluir que uma matriz quadrada de ordem n, diagonalizável, possui n autovetores linearmente independentes, ou seja, número suficiente de vetores para formar uma base de $\mathbb{R}^n$.

Teorema 2.2 *Uma matriz A é não degenerada se, e somente se, possuir decomposição espectral definida por 2.1.*

Teorema 2.3 *Se A for uma matriz de ordem n, então possuirá n autovalores contados pela multiplicidade algébrica. Em particular, se os zeros do polinômio característico forem simples, então a matriz possuirá n autovalores distintos.*

Todavia, como nem toda matriz quadrada é não degenerada, nem toda matriz quadrada pode ser decomposta na forma (2.1). Portanto, a alternativa é utilizar a fatoração de Schur, designada por Triangularização ortogonal, cujo principal objetivo é simplificar a matriz original, reduzindo-a a uma matriz triangular. Basicamente, a fatoração de Schur de uma matriz quadrada qualquer é representada por $A = QTQ^T$, onde $Q_{n\times n}$ é uma matriz ortogonal e $T_{n\times n}$ uma matriz triangular superior, dita similar à matriz $A_{n\times n}$. Além disso, sua existência é garantida pelo Teorema 2.4 [32, Teorema 24.9].

Teorema 2.4 *Toda matriz quadrada possui uma fatoração de Schur.*

Essa decomposição, assim como a diagonalização (2.1), que reduz a matriz original a uma matriz diagonal similar, é obtida por meio de sucessivas transformações lineares conhecidas por transformações de similaridade, as quais são responsáveis pela preservação de propriedades algébricas e geométricas. Segundo Trefethen [32], se $X_{n\times n}$ for uma matriz não singular, o mapeamento descrito em (2.3) representará a transformação de similaridade de A. Portanto, A e Λ e A e T são matrizes similares, se existir uma matriz não singular $X_{n\times n}$, tal que $\Lambda = X^{-1}AX$ e $T = X^{-1}AX$, respectivamente. A principal vantagem desse mapeamento se encontra nas propriedades apresentadas pelo Teorema 2.5 [32, Teorema 24.3], as quais são compartilhadas por matrizes similares.

$$A \to X^{-1}AX. \tag{2.3}$$

Teorema 2.5 *Se X for uma matriz não singular, então A e $X^{-1}AX$ possuirão o mesmo polinômio característico, os mesmos autovalores e as mesmas multiplicidades algébricas e geométricas.*

Trefethen [32] aponta a fatoração de Schur como sendo uma das técnicas mais utilizadas em Análise Numérica porque todo tipo de matriz, inclusive, as degeneradas, podem ser fatoradas desta forma. Além disso, o autor menciona que no caso da fatoração de matrizes simétricas, essa técnica requer menos da metade do trabalho necessário comparado à fatoração de matrizes genéricas. Basicamente, essa fatoração consiste em reduzir qualquer matriz quadrada para a forma compacta triangular superior (caso seja simétrica) ou para forma Hessenberg (caso seja não simétrica) e, consequentemente, reduzi-la para forma compacta diagonal (caso seja simétrica) ou para forma compacta triangular superior ou quase triangular superior (caso seja não simétrica). No entanto, em ambos os casos, os autovalores da matriz original aparecem, necessariamente, na diagonal da forma compacta resultante, diagonal similar, triangular (autovalores reais) ou quase triangular similar (autovalores complexos). Note que a aplicação da fatoração de Schur sobre uma matriz simétrica qualquer (não degenerada) resulta na diagonalização da mesma, como pode ser observado pelo fluxograma da Figura 2.10.

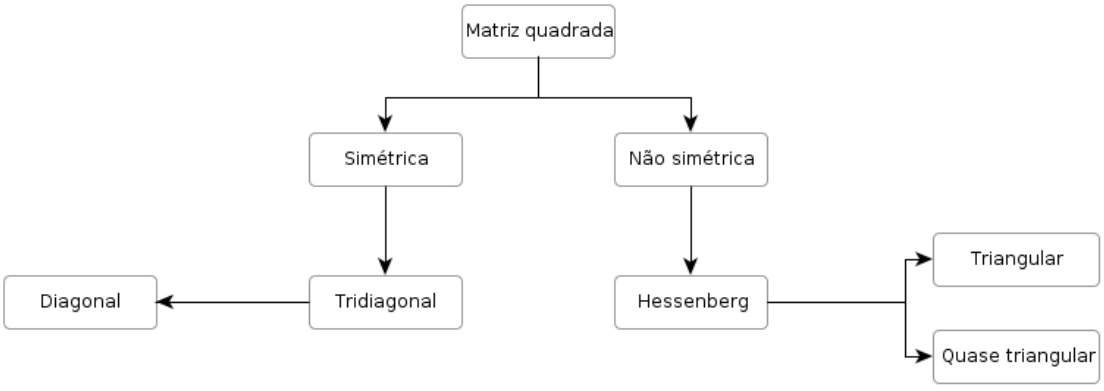


Figura 2.1: Fluxograma da decomposição dependendo do tipo de matriz.

Para facilitar o entendimento do processo de decomposição a Seção 2.1 introduz os conceitos de transformações lineares, matrizes ortogonais e transformações utilizadas no contexto de autovalores e autovetores. A Seção 2.2 apresenta as principais propriedades associadas à decomposição espectral e os tipos e características de matrizes que possuem tal decomposição. A Seção 2.3 apresenta a fatoração QR. A Seção 2.4 detalha o processo da fatoração de Schur e, por fim, a Seção 3.7 apresenta os métodos diretos (matrizes densas) e iterativos (matrizes esparsas) para cálculo de autovalores e autovetores.

2.1 Transformações lineares

Por definição, sejam V e W espaços vetoriais, uma transformação linear T de V em W, denotada por $T : V \to W$, é uma aplicação que associa a um vetor $x \in V$ outro vetor $Tx \in W$, de forma a preservar as operações de adição e multiplicação por escalar:

$$T(x + y) = Tx + Ty \qquad \forall\, x, y \in V, \tag{2.4}$$

$$T(\alpha x) = \alpha Tx \qquad \forall\, \alpha \in \mathbb{R}. \tag{2.5}$$

Dessa forma, se $V = W = \mathbb{R}^n$, a transformação linear envolvida corresponde a matrizes $T_{n\times n}$. Por outro lado, se $V = \mathbb{R}^p$ e $W = \mathbb{R}^n$, a transformação $T : \mathbb{R}^p \to \mathbb{R}^n$ corresponde a matrizes do tipo $T_{n\times p}$ e caso o contrário ($T : \mathbb{R}^n \to \mathbb{R}^p$) a matrizes do tipo $T_{p\times n}$. Por exemplo, seja x um vetor coluna $\in \mathbb{R}^n$ e $T : \mathbb{R}^n \to \mathbb{R}^m$, $y \in \mathbb{R}^m$ é escrito como produto matriz-vetor $y = Tx$, ou seja, pela combinação linear definida por:

$$y_i = \sum_{j=1}^{n} t_{ij} x_j \qquad \text{onde} \qquad i = 1, 2, ..., m. \tag{2.6}$$

Vale ressaltar, que toda transformação linear $T : V \to W$ mapeia o elemento nulo de V no elemento nulo de W e apresenta dois subconjuntos importantes denotados por:

- **Núcleo**: conjunto de elementos de V cuja imagem pela aplicação de T é o elemento nulo de W (vetores $x \in V$ tais que $Tx = 0 \in W$).
- **Imagem**: conjunto de elementos de W que são imagens da transformação T (vetores $y \in W$ que se pode escrever na forma $y = Tx$, para algum $x \in V$).

Além disso, enquanto a imagem $Im(T)$ é um subespaço de W, cuja dimensão é designada por posto e definida pelo número de linhas ou colunas linearmente independentes, o núcleo $N(T)$ é um subespaço de V, cuja dimensão é designada por nulidade e definida pelo número total de colunas de T menos o posto de T. Essa relação pode ser verificada pelo Teorema 2.6 [[2] Teorema 5.6.3] e a transformação T visualizada pela Figura 2.2.

Teorema 2.6 *Se T for uma matriz com n colunas, então posto(T)+nulidade(T)= n.*

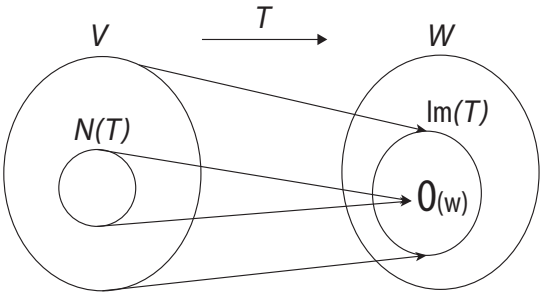


Figura 2.2: Transformação linear $T : V \to W$.

Além do mais, é com base nesses dois subconjuntos que uma transformação linear pode ser classificada como injetora, sobrejetora ou bijetora. Enquanto na transformação linear injetora para quaisquer $u, v \in V$, se $u \neq v$ então $Tu \neq Tv$ (Figura 2.3), na sobrejetora o conjunto imagem de T é o próprio conjunto W ($Im(T) = W$) (Figura 2.4).

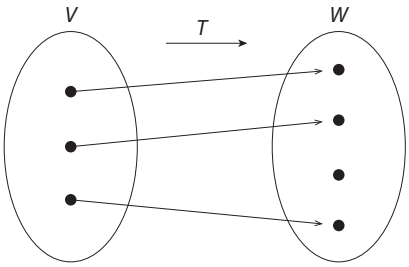


Figura 2.3: Injetora

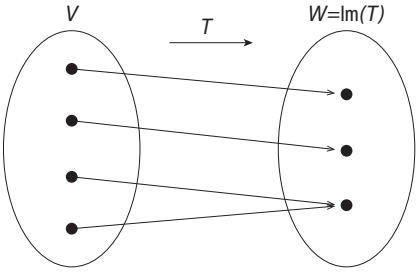


Figura 2.4: Sobrejetora

Por outro lado, a transformação bijetora é definida como sendo injetora e sobrejetora ao mesmo tempo. Isso implica que uma trasnformação bijetora é invertível, ou seja, existe uma transformação do tipo $T^{-1} : W \to V$ tal que $TT^{-1} = I$. Dentre as matrizes invertíveis mais interessantes se encontram as matrizes ortogonais definidas a seguir.

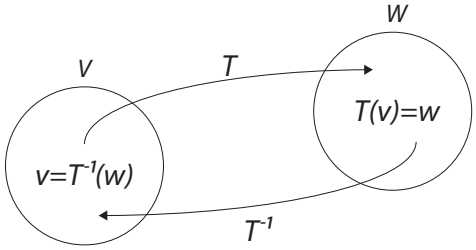


Figura 2.5: Transformação linear bijetora T.

2.1.1 Matrizes ortogonais

Grande parte dos algoritmos de Álgebra Linear Numérica baseiam-se na ortogonalidade devido à estabilidade numérica obtida por meio de matrizes ortogonais, também designadas por matrizes unitárias. Por definição, dois vetores x e y são ortogonais se $x^T y = 0$, ou seja, se o cosseno do ângulo ($90\,^\circ$) formado entre eles for zero. Dessa forma, um conjunto composto por m vetores ortogonais entre si são considerados linearmente independentes e, por isso, constituem uma base de $\mathbb{R}^m$. Por outro lado, um vetor q é dito normal ou unitário se possuir norma um ($||q||_2 = 1$). Portanto, um conjunto de m vetores ortonormais constituem uma base ortonormal de $\mathbb{R}^m$. Consequentemente, matrizes ortogonais (2.7) são matrizes quadradas cujas colunas são vetores ortonormais.

$$Q = (q_1, q_2, ..., q_m) \in \mathbb{R}^m \text{ onde } q_i^T q_i = 1 \; e \; q_i^T q_j = 0, \;\; \forall \; i \neq j \text{ onde } i \; e \; j = 1, 2, ..., m. \tag{2.7}$$

Essas matrizes, por sua vez, satisfazem as seguintes propriedades:

Proposição 2.1 (Proposição 4.3, [32]) *Uma matriz ortogonal Q satisfaz $Q^T Q = I$.*

Proposição 2.2 (Proposição 4.4, [32]) *Uma matriz ortogonal $Q_{m\times m}$ possui posto m e, dado que $Q^T Q = I$, sua inversa é igual a sua transposta ($Q^{-1} = Q^T$).*

Proposição 2.3 (Proposição 4.5, [32]) *As linhas de uma matriz ortogonal são ortogonais, isto é, $Q^T Q = I$.*

Proposição 2.4 (Proposição 4.6, [32]) *O produto de duas matrizes ortogonais é uma matriz ortogonal.*

Proposição 2.5 (Proposição 4.7, [32]) *Dada uma matriz $Q1_{m\times k}$ com colunas ortonormais, então existe uma matriz $Q2_{m\times(m-k)}$ onde $Q = (Q1Q2)$ é uma matriz ortogonal.*

2.1.2 Transformações ortogonais

Dado que uma matriz ortogonal $Q_{m\times m}$ é uma matriz invertível, uma transformação ortogonal $Q : X \to Y$ é uma transformação linear invertível do tipo $Q : \mathbb{R}^m \to \mathbb{R}^m$:

$$y = Qx \text{ e } x = Q^T y \quad \forall\, x \in \mathbb{R}^m \text{ e } \forall\, y \in \mathbb{R}^m. \tag{2.8}$$

Além disso, Trefethen [32] menciona que a multiplicação por uma matriz ortogonal preserva a estrutura geométrica do espaço vetorial envolvido. Isso implica que o produto interno entre vetores é invariante (preservado) sob uma transformação ortogonal 2.9.

$$(Qx)^T(Qy) = x^T Q^T Q y = x^T y. \tag{2.9}$$

A consequência direta dessa propriedade é que, em problemas que envolvam transformações ortogonais, as normas de vetores e matrizes, os ângulos entre vetores e, consequentemente, as distâncias entre vetores, são preservados. Por exemplo, seja x e $y \in \mathbb{R}^m$, $P_{m\times m}$ e $Q_{n\times n}$ matrizes ortogonais e $A_{m\times n}$ uma matriz qualquer, tem-se que:

$$||Px||_2 = ||x||_2, \tag{2.10}$$

$$||PA||_2 = ||A||_2 \text{ ou } ||PAQ||_2 = ||A||_2. \tag{2.11}$$

$$||PA||_F = ||A||_F \text{ ou } ||PAQ||_F = ||A||_F. \tag{2.12}$$

$$cos(\theta) = \frac{(Px)^T Py}{||Px||||Py||} = \frac{x^T y}{||x||||y||}. \tag{2.13}$$

$$||Px - Py|| = ||x - y|| \tag{2.14}$$

Finalmente, uma das propriedades mais interessantes apresentadas por Trefethen [32] é que o determinante de uma matriz ortogonal $Q_{m\times m}$ é igual a ± 1. Devido a isso, a multiplicação por uma matriz desse tipo corresponde a realizar uma rotação no espaço vetorial se o determinante for 1, ou uma reflexão se for -1. Assim, os vetores são refletivos ou rotacionados, mas as relações entre eles, bem como suas propriedades geométricas são preservadas. Dentre as transformações ortogonais mais conhecidas e utilizadas, principalmente, no contexto de fatoração de matrizes, encontram-se a reflexão de Householder e a rotação de Givens. Ambas, possuem a capacidade de anular determinadas componentes de um vetor muito contribuindo, por meio de transformações de similaridade, com o processo de redução de matrizes para formas compactas.

2.1.2.1 Reflexão de Householder

A transformação de Householder é uma reflexão obtida a partir de uma matriz ortogonal $P_{m\times m}$ descrita por (2.15) onde $u \in \mathbb{R}^m$ é um vetor unitário e $I_{m\times m}$ é a matriz identidade.

$$P = I - 2uu^T. \tag{2.15}$$

Trefethen [32] mostra que é possível construir uma matriz P (2.15) capaz de anular determinadas componentes de um vetor. Por exemplo, para que $Px = y$ anule as $(n-k)+1$ últimas componentes de $x = (x_1, ..., x_k..., x_n)$, resultando em y (2.17), é necessário que u seja descrito por (2.16). Além disso, mostra que se as componentes x_k e y_k tiverem sinais opostos, o erro cometido pela transformação será minimizado.

$$u = \frac{u'}{||u'||_2} \quad \text{onde} \quad u' = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ x_k + \text{sinal}(x_k)r \\ x_{k+1} \\ \vdots \\ x_n \end{bmatrix} \quad \text{e} \quad r = \sqrt{\sum_{i=k}^{n} x_i^2}. \tag{2.16}$$

$$y = \begin{bmatrix} x_1, \cdots, x_{k-1}, -\text{sinal}(x_k)r, 0, \cdots, 0 \end{bmatrix}^T. \tag{2.17}$$

Geometricamente, multiplicar um vetor $x \in \mathbb{R}^n$ por P significa refleti-lo no hiperplano (subespaço de dimensão $n-1$) perpendicular a u.

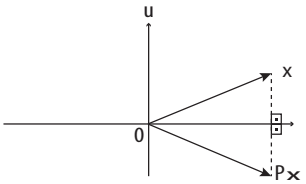


Figura 2.6: Reflexão de Householder de $x \in \mathbb{R}^3$ no $\mathbb{R}^2$

O Exemplo 2.1, a seguir, ilustra sua propriedade em anular componentes vetoriais.

Exemplo 2.1 [Anular a terceira componente do vetor $x = [7 \quad 4 \quad 3]^T$]

Como $k = 2$ e $r = \sqrt{4^2 + 3^2} = 5$. Logo,

$$u = \frac{1}{\sqrt{90}} \begin{bmatrix} 0 \\ 9 \\ 3 \end{bmatrix} \quad e \quad P = I - 2uu^T = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -4/5 & -3/5 \\ 0 & -3/5 & 4/5 \end{bmatrix}.$$

Portanto, como $Px = y$, $y_1 = x_1 = 7$, $y_2 = -\text{sinal}(x_k)r = -5$ e $y_3 = 0$.

$$y = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -4/5 & -3/5 \\ 0 & -3/5 & 4/5 \end{bmatrix} \begin{bmatrix} 7 \\ 4 \\ 3 \end{bmatrix} = \begin{bmatrix} 7 \\ -5 \\ 0 \end{bmatrix}.$$

Note que as normas dos vetores envolvidos são preservadas após a transformação:

$$||x|| = \sqrt{7^2 + 4^2 + 3^2} = \sqrt{74} \qquad e \qquad ||y|| = \sqrt{7^2 + (-5)^2} = \sqrt{74}.$$

Além disso, devido à propriedade de invertibilidade $x = P^T y$:

$$x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -4/5 & -3/5 \\ 0 & -3/5 & 4/5 \end{bmatrix} \begin{bmatrix} 7 \\ -5 \\ 0 \end{bmatrix} = \begin{bmatrix} 7 \\ 4 \\ 3 \end{bmatrix}.$$

■

Além do mais, é possível reduzir uma matriz para determinada forma compacta ao anular determinadas componentes, a partir de uma sequência de transformações de similaridade realizadas por meio da matriz ortogonal (2.15). O esquema a seguir mostra, por exemplo, o processo de redução da matriz A para forma compacta de Hessenberg:

$$\underbrace{\begin{bmatrix} \times & \times & \times & \times \\ \times & \times & \times & \times \\ \times & \times & \times & \times \\ \times & \times & \times & \times \end{bmatrix}}_{A} \rightarrow^{P_1^T} \underbrace{\begin{bmatrix} \times & \times & \times & \times \\ \mathbf{x} & \mathbf{x} & \mathbf{x} & \mathbf{x} \\ \mathbf{0} & \mathbf{x} & \mathbf{x} & \mathbf{x} \\ \mathbf{0} & \mathbf{x} & \mathbf{x} & \mathbf{x} \end{bmatrix}}_{P_1^T A} \rightarrow^{P_1} \underbrace{\begin{bmatrix} \times & \mathbf{x} & \mathbf{x} & \mathbf{x} \\ \times & \mathbf{x} & \mathbf{x} & \mathbf{x} \\ & \mathbf{x} & \mathbf{x} & \mathbf{x} \\ & \mathbf{x} & \mathbf{x} & \mathbf{x} \end{bmatrix}}_{A_1=P_1^T A P_1}$$

$$\rightarrow^{P_2^T} \underbrace{\begin{bmatrix} \times & \times & \times & \times \\ \times & \times & \times & \times \\ & \mathbf{x} & \mathbf{x} & \mathbf{x} \\ & \mathbf{0} & \mathbf{x} & \mathbf{x} \end{bmatrix}}_{P_2^T A_1=P_2^T P_1^T A P_1} \rightarrow^{P_2} \underbrace{\begin{bmatrix} \times & \times & \mathbf{x} & \mathbf{x} \\ \times & \times & \mathbf{x} & \mathbf{x} \\ & \times & \mathbf{x} & \mathbf{x} \\ & & \mathbf{x} & \mathbf{x} \end{bmatrix}}_{A_2=P_2^T P_1^T A P_1 P_2} \rightarrow \underbrace{\begin{bmatrix} \times & \times & \times & \times \\ \times & \times & \times & \times \\ & \times & \times & \times \\ & & \times & \times \end{bmatrix}}_{A_2=H}.$$

Sendo A uma matriz composta pelas colunas $(x_1, x_2, ..., x_m)$, primeiramente, aplica-se a reflexão de Householder P_1^T em A para anular as linhas $3...m$ da primeira coluna (x_1) de modo a não alterar a primeira linha da matriz. Em seguida, aplica-se à direita do resultado de $P_1^T A$ a reflexão de Householder correspondente P_1, com intuito de manter inalterada a primeira coluna (x_1) que já possui algumas componentes anuladas. Note que $P_1^T A P_1$ nada mais é que uma transformação de similaridade do tipo (2.3), onde a matriz ortogonal P equivale à matriz invertível X. Dessa forma, esse processo é repetido até anular as componentes das demais colunas e obter a matriz no formato compacto desejado. Nesse caso, da redução de uma matriz para forma compacta de Hessenberg em especial, são necessárias $m-2$ transformações de similaridade para reduzir a matriz $A_{m\times m}$, como pode ser observado por (2.18) e pelo Exemplo 2.2.

$$\underbrace{P_{m-2}^T ... P_2^T P_1^T}_{P^T} \mathrm{A} \underbrace{P_1 P_2 ... P_{m-2}}_{P} = \mathrm{H} \underbrace{\begin{bmatrix} \times & \times & \times & \times \\ \times & \times & \times & \times \\ & \times & \times & \times \\ & & \times & \times \end{bmatrix}}_{} \tag{2.18}$$

Exemplo 2.2 [Redução de uma matriz $A_{4\times4}$ para forma de Hessenberg]

```
-->A = [450 75 -525 150; 75 253 380 -79; 150 5 325 -215; 150 -604 160 322]
 A  =
    450.    75.   - 525.    150.
    75.     253.    380.  - 79.
    150.    5.      325.  - 215.
    150.  - 604.    160.    322.

-->x1 = A(:,1);                                 // Alvo: primeira coluna de A
-->r1 = sqrt(x1(2)^2 + x1(3)^2 + x1(4)^2);      // k1 = 2. Logo, para r1, i = 2,3,4.
-->ul1 = [0;x1(2)+r1;150;150];                  // Cálculo de u1'
-->u1 = ul1/norm(ul1);                          // Cálculo de u1
-->I = eye(4,4)                                 // Matriz identidade de ordem 4
-->Q1 = I - 2*u1*u1'                            // Cálculo de Q1
 Q1  =
    1.    0.           0.           0.
    0.  - 0.3333333  - 0.6666667  - 0.6666667
    0.  - 0.6666667    0.6666667  - 0.3333333
    0.  - 0.6666667  - 0.3333333    0.6666667
-->Q1'*A                                        // Reflexão à esquerda (linha 1 inalterada)
 ans  =
    450.          75.   - 525.    150.
  - 225.          315.  - 450.  - 45.
  - 5.684D-14     36.   - 90.   - 198.
  - 5.684D-14   - 573.  - 255.    339.

-->A1 = Q1'*A*Q1                                // Reflexão à direita (coluna 1 inalterada)
 A1  =
    450.          225.  - 450.    225.
  - 225.          225.  - 495.  - 90.
  - 5.684D-14     180.  - 18.   - 126.
  - 5.684D-14     135.    99.     693.

-->x2 = A1(:,2);                               // Alvo: segunda coluna de A1
-->r2 = sqrt(x2(3)^2 + x2(4)^2);               // k2 = 3. Logo, para r2, i = 3,4.
-->ul2 = [0;0;x(3)+r2;135];                    // Cálculo de u2'
-->u2 = ul2/norm(ul2);                         // Cálculo de u2
-->Q2 = I - 2*u2*u2'                           // Cálculo de Q2
 Q2  =
    1.    0.    0.     0.
    0.    1.    0.     0.
    0.    0.  - 0.8  - 0.6
    0.    0.  - 0.6    0.8
```

```
-->Q2'*A1                                          // Linhas 1 e 2 inalteradas
 ans  =
    450.          225.        - 450.          225.
  - 225.          225.        - 495.        - 90.
    8.016D-14   - 224.51146   - 50.829916   - 355.80941
  - 6.103D-15   - 14.819059     86.840772     607.8854

-->A2 = Q2'*A1*Q2                                  // Colunas 1 e 2 inalteradas
 A2  =
    450.          225.          225.     450.
  - 225.          225.          450.     225.
    7.958D-14   - 225.          225.   - 225.
  - 1.137D-14     1.847D-13   - 450.     450.

-->spec(A), spec(A2)                               // Autovalores de A e de A2

 ans  =
    104.15572 + 357.94433i
    104.15572 - 357.94433i
    570.84428 + 78.022167i
    570.84428 - 78.022167i
 ans  =
    104.15572 + 357.94433i
    104.15572 - 357.94433i
    570.84428 + 78.022167i
    570.84428 - 78.022167i

-->norm(A),norm(A2),norm(A,'f'),norm(A2,'f')       // Normas 2 e de Frobenius

 ans  =
    812.800322
 ans  =
    812.800322
 ans  =
    1190.5881
 ans  =
    1190.5881
```

■

2.1.2.2 Rotação de Givens

A transformação de Givens é uma rotação obtida a partir de uma matriz ortogonal $G_{m\times m}$ descrita por (2.19) que concide com a matriz identidade $I_{n\times n}$, exceto pelas quatro entradas descritas em (2.20).

$$G(i,j,\theta) \equiv \begin{bmatrix} 1 & \cdots & 0 & \cdots & 0 & \cdots & 0 & \\ \vdots & \ddots & \vdots & & \vdots & & \vdots & \\ 0 & \cdots & c & \cdots & s & \cdots & 0 & i \\ \vdots & & \vdots & \ddots & \vdots & & \vdots & \\ 0 & \cdots & -s & \cdots & c & \cdots & 0 & j \\ \vdots & & \vdots & & \vdots & \ddots & \vdots & \\ 0 & \cdots & 0 & \cdots & 0 & \cdots & 1 & \\ & & i & & j & & & \end{bmatrix}. \tag{2.19}$$

$$\begin{bmatrix} c & s \\ -s & c \end{bmatrix} = \begin{bmatrix} \cos(\theta) & \text{sen}(\theta) \\ -\text{sen}(\theta) & \cos(\theta) \end{bmatrix}. \tag{2.20}$$

Em outras palavras, cada rotação de Givens $G(i,j,\theta)$ está associada a um plano de coordenadas (i,j) e para aplicá-la é necessário utilizar:

$$(i,i) = \cos(\theta),\ \ (i,j) = \text{sen}(\theta),\ \ (j,i) = -\text{sen}(\theta)\ \ e\ \ (j,j) = \cos(\theta). \tag{2.21}$$

Assim como a reflexão de Householder, essa transformação pode ser utilizada para anular determinada componente de um vetor. Por exemplo, dado x, i e j, é possível anular a componente x_j pela escolha dos seguintes valores de $\cos(\theta)$ e $\text{sen}(\theta)$:

$$\begin{bmatrix} \cos(\theta) & \text{sen}(\theta) \\ -\text{sen}(\theta) & \cos(\theta) \end{bmatrix} \begin{bmatrix} x_i \\ x_j \end{bmatrix} = \begin{bmatrix} \sqrt{x_i^2 + x_j^2} \\ 0 \end{bmatrix},$$

$$\cos(\theta) = \frac{x_i}{\sqrt{x_i^2 + x_j^2}}, \qquad \text{sen}(\theta) = \frac{x_j}{\sqrt{x_i^2 + x_j^2}}. \tag{2.22}$$

Geometricamente, como a transformação obtida a partir dessa matriz produz uma rotação θ nas coordenadas i e j, multiplicar um vetor $x \in \mathbb{R}^n$ por uma matriz Givens significa rotacioná-lo um ângulo θ.

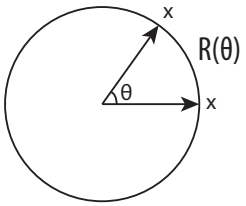


Figura 2.7: Rotação de Givens em $\mathbb{R}^2$

Por outro lado, vale ressaltar que a associação dos valores descritos em (2.21) com a matriz identidade desconsidera os demais eixos da matriz diferentes de i e j. Por exemplo, a rotação de um vetor associada ao plano de coordenadas (2,4) altera apenas os valores da segunda e da quarta componente do vetor, como por ser visto por meio do Exemplo 2.3. Do mesmo modo, a rotação associada ao plano de coordenadas (3,4) altera apenas a terceira e a quarta componente do vetor, como mostra o Exemplo 2.4.

Exemplo 2.3 [Rotação de $x = [1, 2, 3, 4]$ associada ao plano $(i = 2,\ j = 4)$]

```
-->x = [1,2,3,4];
-->sq=sqrt(x(2)^2 + x(4)^2);
-->c=x(2)/sq; s=x(4)/sq;
-->G = [1 0 0 0; 0 c 0 s;0 0 1 0; 0 -s 0 c];
-->y = (G*x')'
 y  =
    1.  4.472136  3.  0.
```

■

Exemplo 2.4 [Rotação de $x = [1, 2, 3, 4]$ associada ao plano $(i = 3,\ j = 4)$]

```
-->x = [1,2,3,4];
-->sq = sqrt(x(3)^2 + x(4)^2);
-->c = x(3)/sq; s = x(4)/sq;
-->G = [1 0 0 0; 0 1 0 0; 0 0 c s; 0 0 -s c];
-->y = (G*x')'
 y  =
    1.    2.    5.  - 4.441D-16
```

■

Dessa forma, para anular $n - 1$ componentes de um vetor será necessário aplicar $n - 1$ rotações, como pode ser observado pelo Exemplo 2.5.

Exemplo 2.5 [Anular a terceira e a segunda componente do vetor $x = [7\ 4\ 3]^T$]

Anulando componente 3 de x: $x_{(2)} = 4, x_{(3)} = 3$. Logo, $\cos(\theta) = \frac{4}{5}$, $\text{sen}(\theta) = \frac{3}{5}$,

$$G_1(2,3,\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 4/5 & 3/5 \\ 0 & -3/5 & 4/5 \end{bmatrix} \rightarrow x_1 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 4/5 & 3/5 \\ 0 & -3/5 & 4/5 \end{bmatrix} \begin{bmatrix} 7 \\ 4 \\ 3 \end{bmatrix} = \begin{bmatrix} 7 \\ 5 \\ 0 \end{bmatrix}.$$

Anulando componente 2 de x_1: $x1_{(1)} = 7, x1_{(2)} = 5$. Logo, $\cos(\theta) = \frac{7}{\sqrt{74}}$, $\text{sen}(\theta) = \frac{5}{\sqrt{74}}$,

$$G_2(1,2,\theta) = \begin{bmatrix} \frac{7}{\sqrt{74}} & \frac{5}{\sqrt{74}} & 0 \\ -\frac{5}{\sqrt{74}} & \frac{7}{\sqrt{74}} & 0 \\ 0 & 0 & 1 \end{bmatrix} \rightarrow x_2 = \begin{bmatrix} \frac{7}{\sqrt{74}} & \frac{5}{\sqrt{74}} & 0 \\ -\frac{5}{\sqrt{74}} & \frac{7}{\sqrt{74}} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 7 \\ 5 \\ 0 \end{bmatrix} = \begin{bmatrix} \sqrt{74} \\ 0 \\ 0 \end{bmatrix}.$$

É possível verificar a seguir que $Q = G_2 G_1$ implica em $Qx = x_2$:

$$Q = \underbrace{\begin{bmatrix} \frac{7}{\sqrt{74}} & \frac{5}{\sqrt{74}} & 0 \\ -\frac{5}{\sqrt{74}} & \frac{7}{\sqrt{74}} & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{G_2} \underbrace{\begin{bmatrix} 1 & 0 & 0 \\ 0 & 4/5 & 3/5 \\ 0 & -3/5 & 4/5 \end{bmatrix}}_{G_1} = \begin{bmatrix} \frac{7}{\sqrt{74}} & \frac{4}{\sqrt{74}} & \frac{3}{\sqrt{74}} \\ -\frac{5}{\sqrt{74}} & \frac{28}{5\sqrt{74}} & \frac{21}{5\sqrt{74}} \\ 0 & -3/5 & 4/5 \end{bmatrix}$$

$$\underbrace{\begin{bmatrix} \frac{7}{\sqrt{74}} & \frac{4}{\sqrt{74}} & \frac{3}{\sqrt{74}} \\ -\frac{5}{\sqrt{74}} & \frac{28}{5\sqrt{74}} & \frac{21}{5\sqrt{74}} \\ 0 & -3/5 & 4/5 \end{bmatrix}}_{Q} \begin{bmatrix} 7 \\ 4 \\ 3 \end{bmatrix} = \begin{bmatrix} \sqrt{74} \\ 0 \\ 0 \end{bmatrix}.$$

Além disso, devido à propriedade de invertibilidade, $x = Q^T y$:

$$x = \begin{bmatrix} \frac{7}{\sqrt{74}} & -\frac{5}{\sqrt{74}} & 0 \\ \frac{4}{\sqrt{74}} & \frac{28}{5\sqrt{74}} & -3/5 \\ \frac{3}{\sqrt{74}} & \frac{21}{5\sqrt{74}} & 4/5 \end{bmatrix} \begin{bmatrix} \sqrt{74} \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 7 \\ 4 \\ 3 \end{bmatrix}.$$

■

Analogamente à transformação de Householder, é possível reduzir uma matriz para determinada forma compacta ao anular determinadas componentes a partir de uma sequência de transformações de similaridade realizadas por meio da matriz ortogonal de Givens definida em (2.19). O esquema a seguir apresenta, em destaque, as linhas que são alteradas a cada transformação realizada. Note que após a aplicação de G_1 as linhas 3 (i) e 4 (j) da matriz A_0 foram alteradas, enquanto as linhas 1 e 2 permaneceram inalteradas. Do mesmo modo, após a aplicação de G_1^T as colunas 3 (i) e 4 (j) foram alteradas, enquanto as colunas 1 e 2 mantiveram-se inalteradas. Esse processo é realizado n vezes ($A_0 ... A_{n-1}$) até que todas as n componentes sejam anuladas.

$$\underbrace{\begin{bmatrix} \times & \times & \times & \times \\ \times & \times & \times & \times \\ \times & \times & \times & \times \\ \times & \times & \times & \times \end{bmatrix}}_{A_0} \rightarrow^{G_1} \underbrace{\begin{bmatrix} \times & \times & \times & \times \\ \times & \times & \times & \times \\ * & * & * & * \\ \mathbf{0} & * & * & * \end{bmatrix}}_{G_1 A} \rightarrow^{G_1^T} \underbrace{\begin{bmatrix} \times & \times & * & * \\ \times & \times & * & * \\ \times & \times & * & * \\ 0 & \times & * & * \end{bmatrix}}_{A_1 = G_1 A G_1^T} \rightarrow^{G_2} \underbrace{\begin{bmatrix} \times & \times & \times & \times \\ * & * & * & * \\ \mathbf{0} & * & * & * \\ 0 & \times & \times & \times \end{bmatrix}}_{G_2 A_1}$$

$$\rightarrow^{G_2^T} \underbrace{\begin{bmatrix} \times & * & * & \times \\ \times & * & * & \times \\ 0 & * & * & \times \\ 0 & * & * & \times \end{bmatrix}}_{A_2 = G_2 A_1 G_2^T} \rightarrow^{G_3} \underbrace{\begin{bmatrix} \times & \times & \times & \times \\ \times & \times & \times & \times \\ 0 & * & * & * \\ 0 & \mathbf{0} & * & * \end{bmatrix}}_{G_3 A_2} \rightarrow^{G_3^T} \underbrace{\begin{bmatrix} \times & \times & * & * \\ \times & \times & * & * \\ 0 & \times & * & * \\ 0 & 0 & * & * \end{bmatrix}}_{A_3 = G_3 A_2 G_3^T} \rightarrow \underbrace{\begin{bmatrix} \times & \times & \times & \times \\ \times & \times & \times & \times \\ & \times & \times & \times \\ & & \times & \times \end{bmatrix}}_{A_3 = H}$$

Em especial, para reduzir uma matriz A de ordem m para forma compacta de Hessenberg deve-se aplicar exatamente $[\frac{m^2-m}{2} - (m-1)]$ rotações. Isso é o mesmo que dizer que é necessário aplicar uma rotação para cada componente que se deseja anular. A partir do Exemplo 2.6 é possível observar as propridades dessa transformação.

$$\underbrace{G_{m-1} ... G_2 G_1}_{G} \mathrm{A} \underbrace{G_1^T G_2^T ... G_{m-1}^T}_{G^T} = \underbrace{\begin{bmatrix} \times & \times & \times & \times \\ \times & \times & \times & \times \\ & \times & \times & \times \\ & & \times & \times \end{bmatrix}}_{\mathrm{H}} \tag{2.23}$$

Exemplo 2.6 [Redução de uma matriz $A_{4\times4}$ para forma de Hessenberg]

```
->A = [450 75 -525 150; 75 253 380 -79; 150 5 325 -215; 150 -604 160 322]
 A  =
    450.    75.   - 525.    150.
    75.     253.    380.  - 79.
    150.    5.      325.  - 215.
    150.  - 604.    160.    322.

-->x1  = A(:,1);                                    // x1 = primeira coluna de A
-->sq1 = sqrt(x1(3)^2 + x1(4)^2);                   // Anular j (componente 4 de x1)
-->c1  = x1(3)/sq1; s1 = x1(4)/sq1;                 // Calculando valores de seno e cosseno
-->G1  = [1 0 0 0; 0 1 0 0; 0 0 c1 s1; 0 0 -s1 c1]; // Matriz de rotação do plano (3,4)
-->G1*A                                             // Rotação à esquerda
 ans  =
    450.         75.        - 525.         150.
    75.          253.         380.       - 79.
    212.13203  - 423.55696    342.94679    75.660426
    0.         - 430.62803  - 116.67262    379.71634
-->A1 = G1*A*G1'                                    // Rotação à direita
 A1  =
    450.         75.        - 265.16504    477.29708
    75.          253.         212.83914  - 324.56201
    212.13203  - 423.55696    296.       - 189.
    0.         - 430.62803    186.         351.

-->x2 = A1(:,1);                                    // x2 = primeira coluna de A1
-->sq2 = sqrt(x2(2)^2 + x2(3)^2);                   // Anular j (componente 3 de x2)
-->c2 = x2(2)/sq2; s2 = x2(3)/sq2;                  // Calculando valores de seno e cosseno
-->G2 = [1 0 0 0; 0 c2 s2 0; 0 -s2 c2 0; 0 0 0 1];  // Matriz de rotação do plano (2,3)
-->G2*A1                                            // Rotação à esquerda
 ans  =
    450.    75.        - 265.16504    477.29708
    225.  - 315.         350.01786   -286.37825
    0.    - 379.71634  - 102.         243.
    0.    - 430.62803    186.         351.

-->A2 = G2*A1*G2'                                   // Rotação à direita
 A2  =
    450.  - 225.       - 159.09903    477.29708
    225.    225.         413.65747  - 286.37825
    0.    - 222.73864    324.         243.
    0.      31.819805    468.         351.
```

```
-->x3 = A2(:,2);                                    // x3 = segunda coluna de A2
-->sq3 = sqrt(x3(3)^2 + x3(4)^2);                   // Anular j (componente 4 de x3)
-->c3 = x3(3)/sq3; s3 = x3(4)/sq3;                  // Calculando valores de seno e cosseno
-->G3 = [1 0 0 0; 0 1 0 0; 0 0 c3 s3; 0 0 -s3 c3]; // Matriz de rotação do plano (3,4)
-->G3*A2                                            // Rotação à esquerda
 ans  =
   450.  - 225.       - 159.09903    477.29708
   225.    225.         413.65747  - 286.37825
   0.      225.       - 254.55844  - 190.91883
   0.      0.         - 509.11688  - 381.83766
-->A3 = G3*A2*G3'                                   // Rotação à direita
A3  =
   450.  - 225.         225.       - 450.
   225.    225.       - 450.         225.
   0.      225.         225.         225.
   0.      0.           450.         450.

-->spec(A), spec(A3)                                // Autovalores de A e de A3

ans  =
   104.15572 + 357.94433i
   104.15572 - 357.94433i
   570.84428 + 78.022167i
   570.84428 - 78.022167i
ans  =
   104.15572 + 357.94433i
   104.15572 - 357.94433i
   570.84428 + 78.022167i
   570.84428 - 78.022167i

-->norm(A),norm(A3),norm(A,'f'),norm(A3,'f')        // Normas 2 e de Frobenius

ans  =
   812.800322
ans  =
   812.800322
ans  =
   1190.5881
ans  =
   1190.5881
```

■

2.2 Diagonalização ortogonal

Em particular, se $A_{n\times n}$ for uma matriz normal, ou seja, apresentar a propriedade 2.24

$$A^T A = AA^T, \tag{2.24}$$

sua decomposição espectral resultará em uma diagonalização ortogonal, garantida pelo Teorema 2.7 [[32] Teorema 24.8]. Isso significa que a matriz V composta pelos autovetores de $A_{n\times n}$, além de ser não singular (invertível) é, inclusive, uma matriz ortogonal. Dessa forma, a decomposição espectral de $A_{n\times n}$ pode ser descrita por (2.25).

Teorema 2.7 *Uma matriz $A_{n\times n}$ é diagonalizável ortogonalmente se, e somente se, for normal.*

$$A = V\Lambda V^T. \tag{2.25}$$

Por sua vez, toda matriz simétrica composta por elementos reais, também designada por matriz Hermitiana, é um caso especial de matriz normal e, portanto, pode ser diagonalizada ortogonalmente, como afirma o Teorema 2.8 [32, Teorema 24.7]. Além disso, por meio de (2.26) é possível ver que se $V^{-1} = V^T$, a matriz $A_{n\times n}$ é realmente simétrica. Aliás, o fato de uma matriz simétrica de ordem n ser não degenerada implica que a mesma possui n autovetores ortogonais entre si, ou seja, número suficiente de vetores para formar uma base ortonormal de $\mathbb{R}^n$ (Teorema 2.9 [16, Teorema 10.4]).

$$A^T = (V\Lambda V^T)^T = (V^T)^T\Lambda^T V^T = V\Lambda V^T = A. \tag{2.26}$$

Teorema 2.8 *Uma matriz Hermitiana é diagonalizável ortogonalmente e possui autovalores reais.*

Teorema 2.9 *Seja $A_{n\times n}$ uma matriz simétrica (operador simétrico $A : \mathbb{R}^n \to \mathbb{R}^n$), existe uma base ortonormal de $\mathbb{R}^n$ formada por seus autovetores.*

A principal consequência do Teorema 2.9 é que se $A_{n\times n}$ for uma matriz simétrica pode ser escrita como (2.27) e a partir desse produto pode ser decomposta em (2.28), onde cada termo da decomposição representa uma matriz $n\times n$ de posto 1. Isso acontece pois cada coluna de $\lambda_i v_i v_i^T$ é um múltiplo de v_i. O Exemplo 2.7 ilustra essa decomposição.

$$A = V\Lambda V^T = \begin{bmatrix} v_1 & \cdots & v_n \end{bmatrix} \begin{bmatrix} \lambda_1 & & 0 \\ & \ddots & \\ 0 & & \lambda_n \end{bmatrix} \begin{bmatrix} v_1 \\ \vdots \\ v_n \end{bmatrix} = \begin{bmatrix} \lambda_1 v_1 & \cdots & \lambda_n v_n \end{bmatrix} \begin{bmatrix} v_1 \\ \vdots \\ v_n \end{bmatrix}. \tag{2.27}$$

$$A = \sum_{i=1}^{n} \lambda_i v_i v_i^T = \lambda_1 v_1 v_1^T + \lambda_2 v_2 v_2^T + ... + \lambda_n v_n v_n^T. \tag{2.28}$$

Exemplo 2.7 $[A = \lambda_1 v_1 v_1^T + \lambda_2 v_2 v_2^T + ... + \lambda_n v_n v_n^T.]$

$$A = \begin{bmatrix} 7 & 2 \\ 2 & 4 \end{bmatrix} = \begin{bmatrix} \frac{2}{\sqrt{5}} & \frac{-1}{\sqrt{5}} \\ \frac{1}{\sqrt{5}} & \frac{2}{\sqrt{5}} \end{bmatrix} \begin{bmatrix} 8 & 0 \\ 0 & 3 \end{bmatrix} \begin{bmatrix} \frac{2}{\sqrt{5}} & \frac{1}{\sqrt{5}} \\ \frac{-1}{\sqrt{5}} & \frac{2}{\sqrt{5}} \end{bmatrix}$$

$$A = 8v_1 v_1^T + 3v_2 v_2^T$$

$$8v_1 v_1^T = \begin{bmatrix} \frac{2}{\sqrt{5}} \\ \frac{1}{\sqrt{5}} \end{bmatrix} \begin{bmatrix} \frac{2}{\sqrt{5}} & \frac{1}{\sqrt{5}} \end{bmatrix} = 8 \begin{bmatrix} \frac{4}{5} & \frac{2}{5} \\ \frac{2}{5} & \frac{1}{5} \end{bmatrix} = \begin{bmatrix} \frac{32}{5} & \frac{16}{5} \\ \frac{16}{5} & \frac{8}{5} \end{bmatrix}$$

$$3v_1 v_1^T = \begin{bmatrix} \frac{-1}{\sqrt{5}} \\ \frac{2}{\sqrt{5}} \end{bmatrix} \begin{bmatrix} \frac{-1}{\sqrt{5}} & \frac{2}{\sqrt{5}} \end{bmatrix} = 3 \begin{bmatrix} \frac{1}{5} & \frac{-2}{5} \\ \frac{-2}{5} & \frac{4}{5} \end{bmatrix} = \begin{bmatrix} \frac{3}{5} & \frac{-6}{5} \\ \frac{-6}{5} & \frac{12}{5} \end{bmatrix}$$

$$\begin{bmatrix} \frac{32}{5} & \frac{16}{5} \\ \frac{16}{5} & \frac{8}{5} \end{bmatrix} + \begin{bmatrix} \frac{3}{5} & \frac{-6}{5} \\ \frac{-6}{5} & \frac{12}{5} \end{bmatrix} = \begin{bmatrix} 7 & 2 \\ 2 & 4 \end{bmatrix}$$

■

Além disso, cada matriz $v_i v_i^T$ é uma matriz de projeção, onde para cada $x \in \mathbb{R}^n$, o vetor $v_i v_i^T x$ é a projeção ortogonal de x sobre o subespaço gerado por v_i. Algumas propriedades importantes podem ser verificadas por meio dos Exemplos 2.8 e 2.9.

Exemplo 2.8 [Cálculo de potências de uma matriz $A^k = \sum_{i=1}^{n} \lambda_i^k v_i v_i^T$]

$$A^2 = \lambda_1^2 v_1 v_1^T + \lambda_2^2 v_2 v_2^T,$$

$$\begin{bmatrix} 7 & 2 \\ 2 & 4 \end{bmatrix} \begin{bmatrix} 7 & 2 \\ 2 & 4 \end{bmatrix} = 64 \begin{bmatrix} \frac{4}{5} & \frac{2}{5} \\ \frac{2}{5} & \frac{1}{5} \end{bmatrix} + 9 \begin{bmatrix} \frac{1}{5} & \frac{-2}{5} \\ \frac{-2}{5} & \frac{4}{5} \end{bmatrix} = \begin{bmatrix} 53 & 22 \\ 22 & 20 \end{bmatrix}.$$

■

Exemplo 2.9 [Decomposição espectral de matriz simétrica e ortogonalidade]

```
-->A = [12 4 2 3; 4 5 6 8; 2 6 5 6; 3 8 6 9]
 A  =
    12.    4.    2.    3.
    4.     5.    6.    8.
    2.     6.    5.    6.
    3.     8.    6.    9.

-->[V,D] = spec(A); // Decomposição espectral de A (autovalores reais)
-->D
 D  =
  - 1.8203383    0.           0.           0.
    0.           0.6849934    0.           0.
    0.           0.           9.5531903    0.
    0.           0.           0.           22.582155

-->V
 V  =
    0.1056673  - 0.0066395  - 0.8895031    0.4444937
  - 0.8392231  - 0.0547946    0.1591115    0.5170935
    0.3655358  - 0.7827235    0.2637215    0.4291602
    0.3884876    0.6199173    0.3375185    0.5923354

-->round((V*V')*1e4)*1e-4  // Verificando propriedade de ortogonalidade de V
 ans  =
    1.    0.    0.    0.
    0.    1.    0.    0.
    0.    0.    1.    0.
    0.    0.    0.    1.
```

■

2.3 Fatoração QR

Dada uma matriz A, a fatoração QR dessa matriz consiste em representá-la por meio do produto das matrizes Q e R, como é definido pelo Teorema 2.10 [[11] Teorema 5.1].

Teorema 2.10 (Decomposição QR) *Qualquer matriz $A_{m\times n}$, $m \geq n$, pode ser transformada para forma triangular superior por uma matriz ortogonal. A transformação é equivalente à decomposição:*

$$A = Q \begin{pmatrix} R \\ 0 \end{pmatrix},$$

onde $Q_{m\times m}$ é uma matriz ortogonal e $R_{n\times n}$ é uma matriz triangular superior. Além do mais, se as colunas de A forem linearmente independentes, então R será uma matriz não singular (invertível).

Além disso, Trefethen [32] menciona que as sucessivas colunas $(a_1, ..., a_n)$ de uma matriz $A_{m\times n}$ $(m \geq n)$ de posto completo (n) são capazes de gerar sucessivos espaços:

$$< a_1 > \subseteq < a_1, a_2 > \subseteq \cdots \subseteq < a_1, a_2, ..., a_n > . \tag{2.29}$$

Nesse caso, uma vez que a imagem de $A_{m\times n}$ é o espaço gerado por suas n colunas, a idéia da fatoração QR consiste na construção de uma sequência de vetores ortonormais $q_1, q_2, ..., q_n$ capazes de gerar essa mesma sequência de espaços. Isto é, consiste em encontrar uma base ortonormal para a imagem de A $(< q_1, q_2, ..., q_k > = < a_1, a_2, ..., a_k >)$. Todavia, Lars Eldén [11] mostra que é possível obter uma decomposição QR completa obtida por meio da matriz ortogonal $Q_{m\times m} = (Q1Q2)$, como mostra (2.30). Mais especificamente, Golub define por meio do Teorema 2.11 [[15] Teorema 5.2.1] que $Q1_{m\times n}$ é composta por n vetores capazes de gerar a $Im(A)$ e $Q2$ é uma matriz composta por $m - n$ vetores capazes de gerar a $Im(A^T)$.

$$A = Q \begin{pmatrix} R \\ 0 \end{pmatrix} = Q1Q2 \begin{pmatrix} R \\ 0 \end{pmatrix} = Q1R. \tag{2.30}$$

Teorema 2.11 *Dada uma matriz $A_{m\times n}$ $(m \geq n)$ de posto completo (n), onde $A = QR$ e $A = [a_1, ..., a_n]$ e $Q = [q_1, ..., q_n]$ são partições de suas colunas, então:*

$$espaço\ [a_1, ..., a_k] = espaço\ [q_1, ..., q_k] \quad para \quad k = 1 : n.$$

Em particular, se $Q1 = Q(1 : m, 1 : n)$ e $Q2 = Q(1 : m, n + 1 : m)$ então:

$$Im(A) = Im(Q1),$$

$$Im(A^T) = Im(Q2),$$

e $A = Q_1R_1$ com $R_1 = R(1 : n, 1 : n)$.

Por outro lado, é possível notar pelas Figuras 2.8 e 2.9 que no caso da decomposição completa a matriz R possui $m-n$ linhas nulas e no caso da decomposição reduzida, que envolve a matriz $Q1$, composta por n colunas ortonormais ($Im(A) = Im(Q)$), a matriz R possui apenas n linhas, como aponta o Teorema 2.10. Na prática, a fatoração QR completa via reflexões de Householder pode ser entendida por meio do Exemplo 2.10.

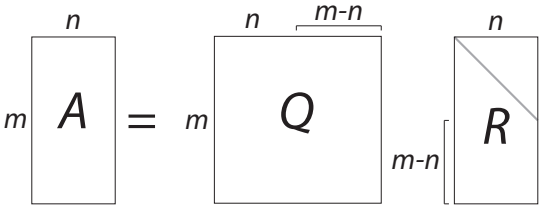


Figura 2.8: QR completa

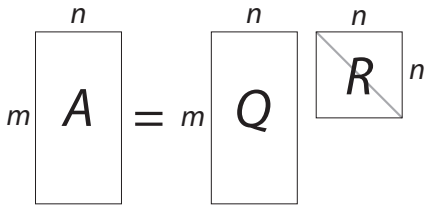


Figura 2.9: QR reduzida

Exemplo 2.10 [Fatoração QR via reflexões de Householder [[15] Algoritmo 5.2.1]]

```
--> // Vetor Householder
--> function [v] = house(x)
-->     n = length(x);
-->     mu = norm(x);
-->     v = x;
-->     if mu != 0
-->         bt = x(1) + sign(x(1))*mu;
-->         v(2:n) = v(2:n)/bt;
-->     end;
-->     v(1) = 1;
--> endfunction
```

```
--> // Pré-multiplicação Householder
--> function A = rowhouse(A,v)
-->     bt = -2/(v'*v);
-->     w = bt*A'*v;
-->     A = A + v*w';
--> endfunction

--> A = [1 3 2; 3 5 2; 11 13 19; 4 3 1]
    A  =
    1.     3.     2.
    3.     5.     2.
    11.    13.    19.
    4.     3.     1.

--> [m,n] = size(A); R=A;
--> for j=1:n
-->     v(j:m) = house(R(j:m,j));
-->     R(j:m,j:n) = rowhouse(R(j:m,j:n),v(j:m));
--> end;

--> R = round(R*1e4)*1e-4
    R  =
  - 12.1244  - 14.2688  - 18.2278
    0.       - 2.8985   - 2.0395
    0.         0.       - 5.7956
    0.         0.         0.
--> // Resultado do Lapack
--> [Q,R] = qr(A);
--> R = round(R*1e4)*1e-4
    R  =
    12.1244    14.2688    18.2278
    0.         2.8985     2.0395
    0.         0.         5.7956
    0.         0.         0.
```

■

Vale ressaltar que a fatoração QR por meio de transformações ortogonais, como por exemplo, as reflexões de Householder e as rotações de Givens, consiste em aplicar sucessivas transformações com intuito de construir a matriz ortogonal $Q_{m\times m}$ composta pelos vetores ortonormais geradores da imagem de A e, ao mesmo tempo, reduzir a matriz A para forma triangular superior. No entanto, para obtenção da matriz $Q_{m\times m}$, o cálculo deve ser realizado explicitamente conforme mostra a relação dada em (2.31).

$$
\begin{aligned}
&Q^T A = \begin{pmatrix} R \\ 0 \end{pmatrix} && \text{onde} \quad Q^T = Q_{m-1} * ... * Q_2 * Q_1 \quad \text{e} \\
&Q_1 = I - 2u_1 u_1^T && \text{onde} \quad u_1 \in \mathbb{R}^m, \\
&Q_2 = \begin{pmatrix} 1 & 0 \\ 0 & P_2 \end{pmatrix} && \text{onde} \quad P_2 = I - 2u_2 u_2^T \quad \text{e} \quad u_2 \in \mathbb{R}^{m-1}, \\
&Q_3 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & P_3 \end{pmatrix} && \text{onde} \quad P_3 = I - 2u_3 u_3^T \quad \text{e} \quad u_3 \in \mathbb{R}^{m-2}, \\
&\vdots \\
&Q_m = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & P_m \end{pmatrix} && \text{onde} \quad P_m = I - 2u_m u_m^T \quad \text{e} \quad u_m \in \mathbb{R}^1.
\end{aligned}
\tag{2.31}
$$

O Exemplo 2.11 ilustra, na prática, como obter a matriz Q de forma explícita.

É importante ressaltar que se a matriz não possuir posto completo a fatoração QR não fornece, necessariamente, uma base ortonormal para sua imagem. Dessa forma, a alternativa para realizar a decomposição corretamente consiste, basicamente, em realizar a fatoração QR com pivotamento de colunas, a qual é representada por $AP = QR$, onde P é uma matriz de permutação das colunas da matriz identidade e $R = Q^T AP$. Essa permutação é fundamental para encontrar o conjunto de vetores capazes de gerar a imagem da matriz envolvida, como pode ser facilmente verificado em Golub [15].

Exemplo 2.11 [Fatoração QR via reflexões de Householder - Cálculo explícito de Q]

```
--> function [v] = get_house(x)
-->     I = eye(length(x),length(x));
-->     if norm(x) != 0
-->         v = sign(x(1))*norm(x)*I(:,1) + x;
-->         v = v/norm(v);
-->     else
-->         v = x; v(1) = 1;
-->     end;
--> endfunction
--> A = [1 3 2; 3 5 2; 11 13 19; 4 3 1]; R = A; [m,n] = size(R);
--> Qi = eye(m,m); Qx = eye(m,m);
--> for j=1:n
-->     vk(j:m) = get_house(R(j:m,j));
-->     // Calcula matrizes Qs
-->     Ic = eye(m-j+1,m-j+1);
-->     Qc = Ic - 2*vk(j:m)*vk(j:m)';
-->     if m-j+1 < m
-->         Q = Qi;
-->         Q(1:j-1,1:j-1) = Qi(1:j-1,1:j-1);
-->         Q(j:m,j:m) = Qc(1:m-j+1,1:m-j+1);
-->     else
-->         Q = Qc;
-->     end;
-->     // Multiplica Qs [Qm-1...Q3 Q2 Q1]
-->     Qf = Qx*Q; Qx = Qf;
-->     // Gera matriz triangular
-->     R = Q'*R;
--> end;
--> round(R*1e4)*1e-4
    ans  =  - 12.1244  - 14.2688  - 18.2278
              0.       - 2.8985   - 2.0395
              0.         0.       - 5.7956
              0.         0.         0.
--> Rf = Qf'*A;
--> round(Rf*1e4)*1e-4
    ans  = - 12.1244  - 14.2688  - 18.2278
             0.       - 2.8985   - 2.0395
             0.         0.       - 5.7956
             0.         0.         0.
```

■

2.4 Fatoração de Schur

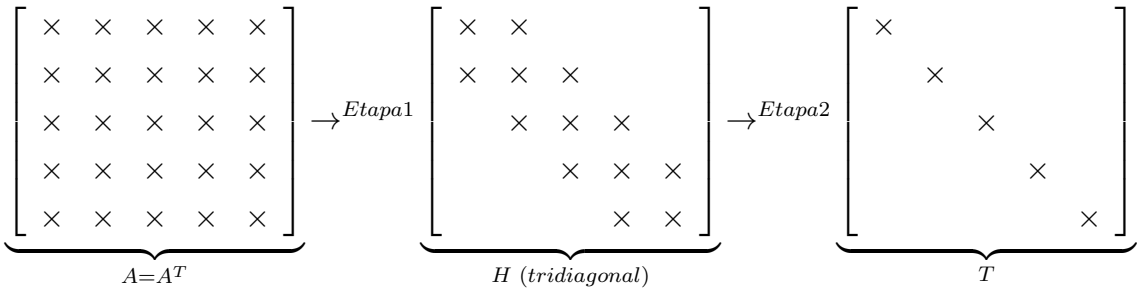
Considerando que nem toda matriz quadrada pode ser reduzida à forma diagonal, a fatoração de Schur se apresenta como uma alternativa onde qualquer matriz quadrada pode ser representada pelo produto QTQ^T, no qual Q é uma matriz ortogonal e T é uma matriz triangular superior similar à matriz original, cujos autovalores aparecem necessariamente na diagonal de T. Entretanto, uma matriz real ($A \neq A^T$) pode conter autovalores complexos, uma vez que os zeros do polinômio característico podem ser reais ou complexos. Portanto, nesse caso, a matriz T é considerada matriz bloco triangular ou quase triangular, como define o Teorema de Schur 2.12 [[7] Teorema 4.3].

Teorema 2.12 *Se A for real, existirá uma matriz ortogonal real V onde $V^TAV = T$ será uma matriz quase triangular superior. Isso significa que T será bloco triangular superior com blocos 1-por-1 ou 2-por-2 em sua diagonal e seus autovalores serão autovalores dos blocos diagonais. Além disso, blocos 1-por-1 corresponderão a autovalores reais e blocos 2-por-2 corresponderão a pares conjugados de autovalores complexos.*

Trefethen [32] mostra que a forma de computar a fatoração $A = QTQ^T$ consiste basicamente em utilizar uma sequência de transformações ortogonais similares $X \to Q_j^T X Q_j$ para que o produto (2.32) convirja para uma matriz triangular superior T.

$$\underbrace{Q_j^T \cdots Q_2^T Q_1^T}_{Q^T} A \underbrace{Q_1 Q_2 \cdots Q_j}_{Q}. \tag{2.32}$$

Em particular, se A for simétrica (não degenerada), a fatoração de Schur apresentará a vantagem de conseguir reduzi-la para a forma diagonal, por meio de duas etapas:



Enquanto a primeira etapa consiste em aplicar transformações de similaridade para produzir uma matriz tridiagonal T (Hessenberg simétrica) a partir de A, a segunda consiste em aplicar uma iteração em T, pela qual gera-se uma sequência de matrizes tridiagonais que convergem para a forma diagonal, composta pelos autovalores de A. Vale ressaltar que a forma de Schur não é única, pois os autovalores podem aparecer na diagonal de T em qualquer ordem. O resultado dessa fatoração pode ser observado pelo Exemplo 2.12.

Exemplo 2.12 [Fatoração de Schur - Matriz simétrica]

```
--> A = [1 4 3; 4 2 7; 3 7 8]
    A  =
    1.    4.    3.
    4.    2.    7.
    3.    7.    8.

--> T = schur(A);   // Matriz diagonal
--> round(T*1e8)*1e-8
    ans  =
    14.302323    0.           0.
    0.           0.2184481    0.
    0.           0.         - 3.5207713

--> spec(A)'
    ans  =
  - 3.5207713    0.2184481    14.302323
```

■

Por sua vez, se a matriz A for não simétrica, a primeira etapa consistirá em aplicar transformações de similaridade para transformá-la em uma matriz superior de Hessenberg H e a segunda, consistirá em aplicar iterações com intuito de obter uma triangularização superior de H por blocos diagonais, onde os autovalores desses blocos serão os autovalores de A. Analogamente, isso pode ser observado pelos Exemplos 2.13 e 2.14.

$$\underbrace{\begin{bmatrix} \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \end{bmatrix}}_{A \neq A^T} \rightarrow^{Etapa1} \underbrace{\begin{bmatrix} \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ & \times & \times & \times & \times \\ & & \times & \times & \times \\ & & & \times & \times \end{bmatrix}}_{H} \rightarrow^{Etapa2} \underbrace{\begin{bmatrix} \times & \times & \times & \times & \times \\ & \times & \times & \times & \times \\ & & \times & \times & \times \\ & & & \times & \times \\ & & & & \times \end{bmatrix}}_{T}$$

Exemplo 2.13 [Fatoração de Schur - Matriz não simétrica]

```
--> A = [1 24 3; 3 2 7; 13 9 -15]
    A  =
    1.     24.    3.
    3.     2.     7.
    13.    9.   - 15.

--> T = schur(A)       // Quase triangular superior
    T  =
  - 13.71018     11.577099  - 15.338701
  - 1.5429392  - 13.71018   - 11.729828
    0.           0.           15.420359
--> bloco = T(1:2,1:2) // Quase triagular - Bloco 2-por-2 (autovalores complexos)
    bloco  =
  - 13.71018     11.577099
  - 1.5429392  - 13.71018

--> spec(bloco)'       // Autovalores do bloco 2-por2
    ans  =
  - 13.71018 + 4.2264358i  - 13.71018 - 4.2264358i

--> spec(A)'           // Autovalores da matriz A
    ans  =
    15.420359  - 13.71018 - 4.2264358i  - 13.71018 + 4.2264358i
```

■

Exemplo 2.14 [Fatoração de Schur - Matriz não simétrica]

```
--> A = [1 32 2; 32 1 1; 36 9 3]
    A  =
    1.     32.    2.
    32.    1.     1.
    36.    9.     3.

--> T = schur(A) // Matriz triangular - Blocos 1-por-1 (autovalores reais)
    T  =
    35.115254    23.456599    22.496722
    0.         - 31.404717    12.793963
    0.           0.           1.2894631

--> spec(A)'
    ans  =
    35.115254  - 31.404717    1.2894631
```

■

2.5 Cálculo de autovalores e autovetores

Aparentemente, achar os zeros do polinômio característico (2.2) seria a forma mais fácil de se calcular os autovalores de qualquer matriz. No entanto, esse método depende do cálculo do determinante, o que para matrizes grandes é impraticável. Além do mais, não existem fórmulas para se calcular soluções analíticas de equações polinomiais de grau maior que quatro, como indica o Teorema 2.13 [Teorema 25.1 [32]]. Portanto, a saída é utilizar métodos que calculam valores aproximados, isto é, métodos que produzam bons números que convirjam rapidamente na direção dos autovalores e autovetores. Para tanto, existem duas classes desses métodos: os diretos, geralmente utilizados em caso de matrizes densas e os iterativos, geralmente usados em caso de matrizes esparsas.

Teorema 2.13 *Para qualquer $m \geq 5$, existe um polinômio $p(z)$ de grau m com coeficientes racionais que possui um zero real ($p(r) = 0$) com a propriedade no qual r não pode ser escrito usando qualquer expressão envolvendo número racionais, operações de adição, subtração, multiplicação, divisão e k-ésimos zeros.*

2.5.1 Métodos diretos

O que caracteriza os métodos diretos é a realização do processo completo de decomposição espectral, calculando todos os autovalores e, opcionalmente, todos os autovetores de uma matriz. Vários são os algoritmos diretos para cálculo de ambos. Segundo o estudo comparativo de Demmel [7], dentre os métodos mais rápidos que realizam o cálculo completo se encontram a Iteração QR e o método de Divisão e conquista. Por sua vez, o primeiro é muito utilizado tanto para matrizes simétricas quanto para matrizes não simétricas. No entanto, para matrizes simétricas, ele é ideal quando se deseja computar apenas os autovalores independente da ordem da matriz e/ou os autovetores, considerando uma dimensão aproximadamente menor que 25. Por outro lado, o método de Divisão e conquista é o mais utilizado quando se deseja computar tanto os autovalores quanto os autovetores, considerando um dimensão aproximadamente maior que 25. Vale ressaltar que o valor exato desse *threshold* depende do computador a ser utilizado. Além do mais, dado a eficiência, esses métodos são utilizados pela função *eig* do MATLAB e pela *spec* do SCILAB, realizando a decomposição espectral de uma matriz densa por meio das duas etapas da fatoração de Schur (Seção 2.4):

1) Etapa Redução de A para forma tridiagonal (se $A = A^T$) e Hessenberg (se $A \neq A^T$) por meio de uma sequência finita de transformações ortogonais de Householder.

$$A_{n\times n} \rightarrow T_{n\times n} = Q^T A Q, \quad Q = Q_1 * Q_2 ... Q_{n-2}. \tag{2.33}$$

2) Etapa Redução iterativa para forma diagonal (se $A = A^T$), triangular ou quase triangular (se $A \neq A^T$), por meio do algoritmo de Iteração QR ou Divisão e conquista.

$$T_{n\times n} \text{ ou } \Lambda_{n\times n} = Q^T T Q, \quad Q = Q_1 * Q_2 ... Q_k. \tag{2.34}$$

A principal diferença entres essas duas etapas é que a primeira é finita, enquanto a segunda é iterativa e depende de algum critério de parada. Contudo, segundo Lars Eldén [11], esse tipo de iteração, como por exemplo, a Iteração QR, converge muito rapidamente, de modo que em aritmética de ponto flutuante é considerada finita. Para Demmel [7], a justificativa pela qual é considerada um método direto se deve ao fato de que, na prática, funciona tão bem que, em média, necessita apenas de duas iterações por autovalor para convergir para quase todas as matrizes. Alem disso, apesar da sua taxa de convergência ser garantida e acelerada pela incorporação de deslocamentos (*shifts*), o autor esclarece que encontrar uma estratégia que garanta a convergência rápida para todas as matrizes não é trivial. Agora, o fluxograma da Figura 2.10 apresentado no início do capítulo pode ser claramente dividido em duas etapas distintas: 1) redução da matriz original para forma tridiagonal ou Hessenberg via sucessivas transformações de Householder e 2) redução da matriz tridiagonal ou Hessenberg resultante para forma diagonal ou triangular ou quase-triangular respectivamente, via iteração QR.

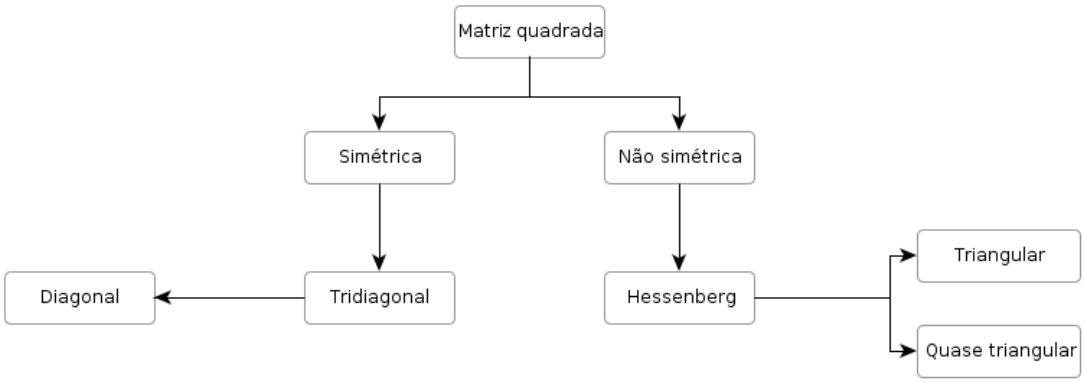


Figura 2.10: Fluxograma da decomposição utilizando a iteração QR na etapa 2.

2.5.1.1 Iteração QR

Dada a matriz $A_{n\times n}$, a Iteração QR retorna no corpo da própria matriz uma matriz diagonal (se $A = A^T$), triangular superior ou quase triangular (se $A \neq A^T$) com os elementos da diagonal convergindo para os autovalores de A. Dado que sua taxa de convergência é garantida e acelerada pela incorporação de deslocamentos (*shifts*), o método passa a ser designado por Iteração QR com deslocamento, como pode ser observado pelo Algoritmo 1 [Algoritmo 7.5.2, [15]]. Note que, inicialmente, a matriz é reduzida para forma de Hessenberg. Demmel [7] mostra que essa redução, juntamente com a incorporação implícita dos *shifts*, reduz o custo de computar apenas autovalores de $O(n^3)$ para $O(n^2)$ e o custo total, incluindo os autovetores, de $O(n^4)$ para $O(n^3)$.

Algoritmo 1: Iteração QR com deslocamento

Entrada: A e *maxiter* (matriz quadrada A e número máximo de iterações)
Saída: H (matriz cuja diagonal é composta por blocos 1-por-1 (autovalores reais) e/ou 2-por-2 (cujos autovalores são reais ou complexos em pares conjugados))

```
início
    H^(0) = Q^T A Q
    para cada k ← 1 até maxiter faça
        Escolha do shift μ^(k)
        [Q^(k), R^(k)] = qr(H^(k-1) − μ^(k) I)
        H^(k) = R^(k) Q^(k) + μ^(k) I
fim
```

2.5.1.2 Matrizes não simétricas

Durante o processo de Iteração QR, a matriz não reduzida de Hessenberg (H) tende a sofrer um esvaziamendo (*deflation*), convergindo para forma triangular ou quase triangular. No entanto, esse esvaziamento acontece quando sua subdiagonal torna-se devidamente pequena. Por sua vez, Golub [15] mostra pelo Teorema 2.14 [Teorema 7.5.1, [15]] que se o *shift* escolhido for um autovalor exato de H, então, em aritmética exata, o esvaziamento relativo ocorre em um passo, como mostra o Exemplo 2.15.

Teorema 2.14 *Seja μ autovalor de uma matriz não reduzida de Hessenberg $H_{n\times n}$. Se $H = RQ + \mu I$, onde $H - \mu I = QR$ é a fatoração QR de $H - \mu I$, então $h_{(n,n-1)} = 0$ e $h_{(n,n)} = \mu$.*

Exemplo 2.15 [Esvaziamento (deflation) em uma iteração - Exemplo 7.5.1 [Golub]]

```
--> H = [9 -1 -2; 2 6 -2; 0 1 5]
    H  =
    9.  - 1.  - 2.
    2.    6.  - 2.
    0.    1.    5.
--> spec(H)'
    ans  =
    7.0000001    6.9999999    6.
--> shift = 6;                    // Autovalor de H
--> [Q,R] = qr(H-shift*eye(3,3)); // Esvaziamento relativo ao shift em uma iteração!
--> H = R*Q + shift*eye(3,3)      // H(n,n-1) = H(3,2) = 0 e H(n,n) = H(3,3) = shift.
    H  =
    8.5384615  - 3.7313173    1.0090092
    0.6343239    5.4615385  - 1.3867505
    0.           0.           6.
--> shift = 6.9999999;            // Autovalor de H
--> [Q,R] = qr(H-shift*eye(3,3)); // Esvaziamento relativo ao shift em uma iteração!
--> H = R*Q + shift*eye(3,3)      // H(n-1,n-2) = H(2,1) = 0, H(n-1,n-1) = shift.
    H  =
    7.0000001  - 4.3656413  - 0.4042261
    0.           6.9999999    1.6666667
    0.           0.           6.
```

■

Dessa forma, considerando o *shift* (μ) variando a cada iteração, Golub [15] mostra que uma boa heurística é considerar $\mu = H(n,n)$ como a melhor aproximação de um autovalor ao longo da diagonal. Essa estratégia pode ser observada pelo Exemplo 2.16.

Exemplo 2.16 [Iteração com deslocamento simples - Exemplo 7.5.2 [Golub]]

```
--> H = [1 2 3; 4 5 6; 0 0.001 7];
--> spec(H)'
    ans  =
  - 0.4640394    6.4522979    7.0117415
--> shift = H(3,3);               // shift = 7
--> [Q,R] = qr(H-shift*eye(3,3)); // Apenas uma iteração!
--> H = R*Q + shift*eye(3,3)      // H(n,n-1) próximo de 0 e H(n,n) próximo do autovalor.
    H  =
  - 0.5384615  - 1.6908049    0.8350998
    0.3076928    6.5264616  - 6.6555487
    0.           0.0000216    7.012
```

■

Essa estratégia é designada por Iteração QR com deslocamento simples (*single-shift*), ideal para o caso em que os blocos da diagonal de H são do tipo 1-por-1. No entanto, apesar dessa estratégia também atender o caso em que os autovalores são complexos, como ilustra parte do Exemplo 2.18, o ideal é considerar tanto deslocamentos simples quanto deslocamentos duplos (*double-shift*), uma vez que a matriz T é quase triangular e apresenta, em sua diagonal, blocos 2-por-2. Essa nova estratégia consiste basicamente em aplicar duas Iterações QR consecutivas, na qual a primeira aborda o deslocamento μ e a segunda seu complexo conjugado μ^*. Informações e aplicações mais detalhadas dessa medida podem ser obtidas em Demmel [7] e Golub [15]. O Exemplo 2.17 ilustra a aplicação da Iteração QR com deslocamento simples sobre a matriz de Hessenberg.

Exemplo 2.17 [Iteração QR com deslocamento simples]

```
--> H = [1. -4.8413866 1.2493901; -6.4031242 10. -8.; 0. -2. 2.];
--> spec(H)'
    ans  =
    13.863454  - 2.22503    1.3615763
--> [n,n] = size(H);
--> m = 6; // Aproximadamente 2 iterações por autovalor!
--> for i=1:m
-->     shift = H(n,n)
-->     [Q,R] = qr(H-shift*eye(n,n));
-->     H = R*Q + shift*eye(n,n);
--> end
--> round(H*1e7)*1e-7
    ans  =
    13.867967    3.5242395  - 5.1789674
  - 0.0206112  - 2.2295437    0.7825149
    0.           0.           1.3615763
--> bloco = H(1:2,1:2); spec(bloco)'    // Autovalores do bloco
    ans  =
    13.863454  - 2.22503
```

■

Como mencionado no início dessa seção, a versão mais eficiente da Iteração QR, tanto no caso de matrizes simétricas quanto não simétricas é baseada em deslocamentos implícitos, por possuir convergência quadrática. Basicamente, essa versão é obtida a partir de transformações de similaridade realizadas por meio da rotação de Givens, como pode ser melhor observada em [Algoritmos 8.2.2 e 8.2.3, [15]]. No entanto, para fins didáticos e práticos, os exemplos apresentados ilustram a versão explícita da Iteração QR, de forma a facilitar o entendimento do processo de atualização dos deslocamentos.

Exemplo 2.18 [Decomposição espectral - Matrizes não simétricas]

```
--> // Claudiane Fonseca Rodrigues 15-02-2011
--> // Ilustra as duas etapas da decomposição espectral de uma matriz não simétrica
--> //     1) Redução de Hessenberg
--> //     2) Iteração QR com deslocamento Hi(n,n)
--> // Entrada:
--> //     A = matriz não simétrica
--> //     m = número de iterações
-->
--> A = [1 3 4; 5 2 8; 4 2 1];
--> // 1) Etapa - Redução de Hessenberg
--> H = hess(A);
--> [n,n] = size(H);
--> // 2) Etapa - Iteração QR
--> m = 8;
--> for i=1:m+1
-->     // Computa o deslocamento
-->     shift = H(n,n);
-->     [Q,R] = qr(H-shift*eye(n,n));
-->     H = R*Q + shift*eye(n,n);
--> end

--> round(H*1e8)*1e-8
--> ans  =
    9.3499287    5.9621873    0.5639659
    0.         - 2.633687   - 1.4238853
    0.           0.6094039  - 2.7162417

--> bloco = H(2:3,2:3)
    bloco  =
  - 2.633687   - 1.4238852
    0.6094039  - 2.7162417

--> spec(bloco = H(2:3,2:3))'
    ans  =
  - 2.6749643 - 0.9306005i  - 2.6749643 + 0.9306005i

--> spec(A)'
    ans  =
    9.3499287  - 2.6749643 - 0.9306005i  - 2.6749643 + 0.9306005i
```

■

2.5.1.3 Matrizes simétricas

No caso de matrizes simétricas, na qual sua forma reduzida de Hessenberg é tridiagonal (2.35), o deslocamento $\mu^{(k)}$ mais eficiente é o de Wilkinson, definido por (2.36). A convergência desse deslocamento é garantida pela Proposição 2.6 [Proposição 15.14, [11]].

$$T^{(k)} = \begin{bmatrix} \alpha_1 & \beta_1 & & \\ \beta_1 & \ddots & \ddots & \\ & \ddots & \ddots & \beta_{n-1} \\ & & \beta_{n-1} & \alpha_n \end{bmatrix} \tag{2.35}$$

$$\mu^{(k)} = \text{autovalor de } \begin{bmatrix} \alpha_{n-1} & \beta_{n-1} \\ \beta_{n-1} & \alpha_n \end{bmatrix} \text{ mais próximo de } \alpha_n. \tag{2.36}$$

Proposição 2.6 (Wilkinson) *O algoritmo simétrico de iteração QR com deslocamento de Wilkinson converge cubicamente em relação à decomposição espectral.*

Exemplo 2.19 [Decomposição espectral - Matrizes simétricas]

```
--> // Claudiane F. Rodrigues 10-02-2011
--> // Baseado em http://www.cs.berkeley.edu/~demmel/ma221_Fall09/Matlab/tridiQR.m
--> // Ilustra as duas etapas da decomposição espectral de uma matriz simétrica
--> //     1) Redução de Hessenberg (Tridiagonal)
--> //     2) Iteração QR com deslocamento (explícito) de Wilkinson
--> // Entrada:
--> //     A = matriz simétrica
--> //     m = número de iterações
--> // 1 Etapa
--> A = [1 2 9 14; 2 3 5 2; 9 5 7 5; 14 2 5 8];
--> t = hess(A);
--> round(t*1e8)*1e-8
    ans  =
    1.          - 16.763055    0.           0.
  - 16.763055     13.163701  - 3.7342646    0.
    0.          - 3.7342646    4.892971   - 1.5625513
    0.            0.         - 1.5625513  - 0.0566720
-->
--> [n,n] = size(t);
--> m = 20;
```

```
--> // Salva as entradas das últimas subdiagonal e diagonal
--> NN = [t(n,n-1), t(n,n)];
--> // 2 Etapa - Iteração QR
--> for i=1:m
-->     // Computa o deslocamento de Wilkinson
-->     lc=t(n-1:n,n-1:n);
-->     elc = spec(lc);
-->     if(abs(t(n,n)-elc(1)) < abs(t(n,n)-elc(2))),
-->         shift = elc(1);
-->     else
-->         shift = elc(2);
-->     end;
-->     // Executa QR; Força simetria explícita
-->     [q,r] = qr(t-shift*eye(n,n));
-->     t= r*q + shift*eye(n,n);
-->     t = tril(triu(t,-1),1);
-->     t = (t+t')/2;
-->     // Atualiza NN
-->     NN = [NN;[t(n,n-1),t(n,n)]];
--> end;
--> NN = [NN,NN(:,2)-NN(m + 1,2)];
--> NN
--> NN  =
    T(n,n-1)     T(n,n)       T(n,n)-eigenvalue
  - 1.5625513  - 0.0566720    0.4458704
  - 0.0017985  - 0.5025419    0.0000006
  - 3.303D-12  - 0.5025424    0
  - 4.039D-28  - 0.5025424    0
    Converg.     Converg.     Converg.

--> disp("Matriz tridiagonal final")
--> round(t*1e4)*1e-4
--> Matriz tridiagonal final
    ans  =
    25.375624    0            0          0
    0          - 11.047602    0          0
    0            0            5.17452    0
    0            0            0        - 0.5025424

--> spec(A)'
  ans  =
  - 11.047602  - 0.5025424  5.17452    25.375624
```

■

2.5.2 Métodos iterativos

Métodos iterativos são obtidos a partir de uma sequência de iterações que tendem a convergir para boas aproximações dos autovalores e autovetores da matriz. O que caracteriza-os é a capacidade que possuem de calcular boas aproximações apenas dos k primeiros maiores ou menores autovalores e, opcionalmente, de seus respectivos autovetores. Para tanto, são baseados na projeção de um problema m-dimensional em um subespaço de Krylov de menor dimensão. Basicamente, essa projeção consiste em construir essas boas aproximações, também designadas por valores e vetores de Ritz, por meio de uma sequência iterativa desses subespaços. Dessa forma, a projeção em k subespaços de Krylov consiste em reduzir o problema de uma matriz original de ordem m em uma sequência de problemas de matrizes de dimensões $1, 2, ..., k$. Dentre os principais métodos iterativos referentes aos subespaços de Krylov, o algoritmo de Cornelius Lanczos (1950) [23] é o mais indicado para trabalhar com matrizes simétricas, enquanto o de W. E. Arnoldi (1951) [3] é o mais indicado para trabalhar com matrizes não simétricas. Vale ressaltar que esses dois algoritmos constroem iterativamente as matrizes Q_k (não singular, invertível), T_k tridiagonal (se $A = A^T$) ou H_k Hessenberg (se $A \neq A^T$) respectivamente, a partir de k iterações tal que:

$$T_k \text{ ou } H_k = Q_k^T A Q_k. \tag{2.37}$$

2.5.2.1 Método de Rayleigh-Ritz e subspaços de Krylov

Por definição [Definição 5.1, [7]], o quociente de Rayleigh-Ritz de uma matriz simétrica $A_{n\times n}$ e de um vetor $x \neq 0$ é dado por

$$r(x, A) = \frac{x^T A x}{x^T x}. \tag{2.38}$$

Demmel [7] apresenta duas importantes propriedades desse coeficiente. A primeira é que para qualquer escalar $c \neq 0$, $r(cx, A) = r(x, A)$. A segunda é que se $Aq_i = \lambda q_i$, então $r(q_i, A) = \lambda_i$. Assim, o quociente de Rayleigh-Ritz descreve a decomposição espectral $Q^T A Q = \Lambda = \text{diag}(\lambda_i)$ da matriz A onde $Q = [q_i, ..., q_n]$. Por outro lado, expandindo x na base dos autovetores (q_i) tem-se que: $x = Q(Q^T x) = Q\xi = \sum_i q_i \xi_i$.

Portanto,

$$r(x,A) = \frac{\xi^T Q^T A Q \xi}{\xi^T Q^T Q \xi} = \frac{\xi^T \Lambda \xi}{\xi^T \xi} = \frac{\sum_i \lambda_i \xi_i^2}{\sum_i \xi_i^2}.$$

Em outras palavras, $r(x,A)$ é uma média ponderada dos autovalores de A. Seu maior valor, $max_{x\neq 0}(x,A)$, ocorre em $x = q_1(\xi = e_1)$ igual a $r(q_1,A) = \lambda_1$ e seu menor valor, $min_{x\neq 0}(x,A)$, ocorre em $x = q_n(\xi = e_n)$ igual a $r(q_n,A) = \lambda_n$. Conjuntamente, esses fatos implicam em (2.39), sendo confirmados pelo Teorema 2.15 [[7] Teorema 5.2].

$$max_{x\neq 0}|r(x,A)| = max(|\lambda_1|,|\lambda_n|) = ||A||_2. \tag{2.39}$$

Teorema 2.15 (Teorema Minimax de Courant-Fischer) *Sendo $\lambda_1 \geq ... \geq \lambda_n$ os autovalores de uma matriz simétrica A e $q_1,...,q_n$ seus correspondentes autovetores normalizados,*

$$\max_{\mathbb{R}^j} \min_{0\neq x \,\in\, \mathbb{R}^j} r(x,A) = \lambda_j = \min_{\mathbb{S}^{n-j+1}} \max_{0\neq s \,\in\, \mathbb{S}^{n-j+1}} r(s,A). \tag{2.40}$$

O máximo da primeira expressão para λ_j ocorre sobre todos os j subespaços $\mathbb{R}^j$ de $\mathbb{R}^n$ e o subsequente mínimo ocorre sobre todos vetores $x \neq 0$ desse subespaço. O máximo é atingido em $\mathbb{R}^j$, espaço gerado por $[q_1,...,q_j]$ e uma minimização ocorre quando $x = q_j$.

O mínimo da segunda expressão para λ_j ocorre sobre todos $(n-j+1)$ subespaços dimensionais $\mathbb{S}^{n-j+1}$ de $\mathbb{R}^n$ e o subsequente máximo ocorre sobre todos os vetores $s \neq 0$ desse subespaço. O mínimo é atingido em $\mathbb{S}^{n-j+1}$, espaço gerado por $[q_j,...,q_n]$ e uma máximização ocorre quando $s = q_j$.

Logo, $\lambda_1(A) \geq ...\lambda_j(A) \geq ...\lambda_n(A)$. Segundo Trefethen [32], o quociente (2.38) é uma função contínua na esfera unitária $|x| = 1 \in \mathbb{R}^n$, no qual os autovetores são pontos estacionários da função $r(x,A)$ e os autovalores são os valores de $r(x,A)$ desses pontos. Demmel [7] ilustra esse fato, geometricamente, a partir da Figura 2.11, a qual representa a função contínua do Quociente de Rayleigh-Ritz da seguinte matriz:

$$A = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0.25 & 0 \\ 0 & 0 & 0 \end{bmatrix} = \underbrace{\begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0.25 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix}}_{Q^T \Lambda Q}.$$

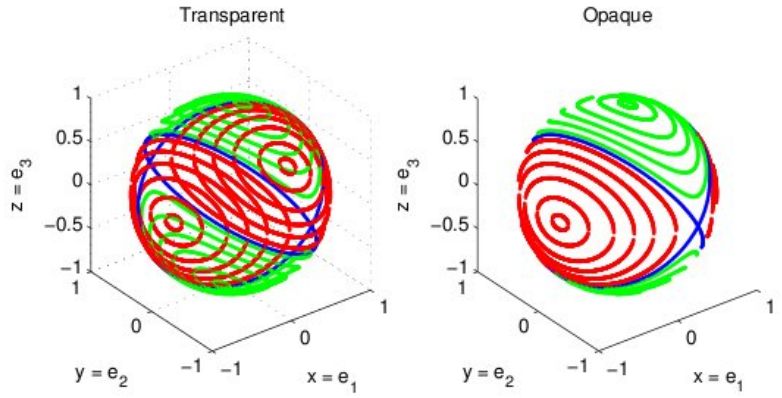


Figura 2.11: Gráfico de contorno do Quociente de Rayleigh na esfera unitária.

A Figura 2.11 apresenta o contorno da função (2.38) em uma esfera unitária para $A = \text{diag}(1, 0.25, 1)$, onde Q é uma simples matriz, no qual $q_i = e_i$ (colunas da matriz identidade). Essa figura é simétrica em relação a origem pois $r(x, A) = r(-x, A)$. Além disso, percebe-se que enquanto os pequenos círculos vermelhos próximos à $\pm\, q_1$ cercam o máximo global $r(\pm\, q_1, A) = \lambda_1 = 1$, os pequenos círculos verdes próximos à $\pm\, q_3$ cercam o mínimo global $r(\pm\, q_3, A) = \lambda_3 = 0$. Por fim, os dois grandes círculos azuis contornam o segundo autovalor, $r(\pm\, q_2, A) = \lambda_2 = 0.25$. Além do mais, dentro dos limites dos círculos azuis se encontram estreitas fatias verdes onde $r(x, A) \leq 0.25$ e por fora do limite, onde $r(x, A) \geq 0.25$, se encontram largas fatias vermelhas. Dessa forma, completa-se o contorno da esfera unitária. O código utilizado para esboçar o contorno da matriz pode ser encontrado em [1] e adaptado para matrizes arbitrárias.

Trefethen [32] discute que (2.38) tende a responder a seguinte pergunta: dado $x \in \mathbb{R}^n$, que escalar α mais se aproxima do autovalor de x no sentido de minimizar $||Ax - \alpha x||_2$? Em resposta, o autor mostra que se o gradiente de $r(x)$ dado por (2.41)

$$\nabla r(x, A) = \frac{2}{x^T x}(Ax - r(x)x) \tag{2.41}$$

[1] `http://www.cs.berkeley.edu/~demmel/ma221_Fall09/Matlab/RayleighContour.m`

for igual a zero, onde $x \neq 0$, então x é um autovetor de A e $r(x, A)$ é seu autovalor correspondente. Por sua vez, Golub [15] apresenta o seguinte raciocínio, supondo uma sequência de vetores ortonormais $q_i \in \mathbb{R}^n$, pode-se definir escalares M_j e m_j por:

$$M_j = \lambda_1(Q_j^T A Q_j) = \max_{y \neq 0} \frac{y^T (Q_j^T A Q_j) y}{y^T y} = \max_{||y||=1} r(Q_j y) \leq \lambda_1(A) \ e \tag{2.42}$$

$$m_j = \lambda_j(Q_j^T A Q_j) = \min_{y \neq 0} \frac{y^T (Q_j^T A Q_j) y}{y^T y} = \min_{||y||=1} r(Q_j y) \geq \lambda_n(A), \tag{2.43}$$

onde $Q_j = [q_1, ..., q_j]$. Segundo o autor, o algoritmo de Lanczos é baseado em como gerar q_j de forma que M_j e m_j sejam boas estimativas de $\lambda_1(A)$ e $\lambda_n(A)$ respectivamente. Para tanto, supondo $u_j \in$ espaço $[q_1, ..., q_j]$ tal que $M_j = r(u_j, A)$, dado que $r(x, A)$ cresce mais rapidamente na direção do seu gradiente (2.41), pode-se assegurar que $M_{j+1} \geq M_j$ se q_{j+1} for determinado por $\nabla r(u_j) \in$ espaço $[q_1, ..., q_{j+1}]$. Da mesma forma, se $v_j \in [q_1, ..., q_j]$ satisfaz $r(v_j) = m_j$, então $\nabla r(v_j) \in$ espaço $[q_1, ..., q_{j+1}]$, uma vez que $r(x)$ decresce mais rapidamente na direção de $-\nabla r(x)$. Encontrar um único q_j que satisfaça esses requisitos parece uma tarefa impossível. No entanto, dado que $\nabla r(x) \in$ espaço $[x, Ax]$, Golub [15] mostra que (2.42) e (2.43) são simultaneamente satisfeitas por meio dos sucessivos subespaços, designados por subespaços de Krylov, gerados pelas colunas da matriz definida em (2.44). Note que esses subespaços são gerados a partir da própria matriz $A_{n \times n}$ e de um vetor unitário $q \in \mathbb{R}^n$ ($||q||_2 = 1$).

$$K_n(A_{n \times n}, q_1, n) = \left[\begin{array}{c|c|c|c|c} q_1 & Aq_1 & A^2 q_1 & ... & A^{n-1} q_1 \end{array} \right]. \tag{2.44}$$

Isto é,

$$\text{espaço}[q_1, ..., q_j] = \text{espaço}[q_1, Aq_1, ..., A^{j-1} q_1] = K(A, q_1, j).$$

Além do mais, pode-se escolher q_{j+1} para que $K(A, q_1, j+1) =$ espaço $[q_1, ..., q_{j+1}]$. Assim, o problema de gerar os vetores $[q_1, ..., q_j]$ adequados, passa a ser o problema de computar bases ortonormais dos subespaços de Krylov $K(A, q_1, j)$.

Em síntese, o procedimento de Rayleigh-Ritz consiste em se aproximar dos autovalores de A pelos autovalores da matriz similar $T_k = Q_k^T A Q_k$ ou $H_k = V_k^T A V_k$. Por sua vez, dada a decomposição de $T_k = V^T \Lambda V$ ou $H_k = V^T T V$, as melhores aproximações dos autovalores, designadas por valores de Ritz, são os valores da diagonal de Λ ou T. Consequentemente, as melhores aproximações dos autovetores correspondentes, designadas por vetores de Ritz, são obtidas pelas colunas de $Q_k V$. Finalmente, o meio pelo qual se torna possível calcular apenas parte dos autovalores e autovetores é computar iterativamente bases ortonormais dos subespaços de Krylov de dimensão K.

2.5.2.2 Extraindo informação de A a partir da multiplicação matriz-vetor

Dado um vetor unitário $x = q$ e uma subrotina para computar o produto Ax, é possível computar uma sequência de produtos matriz-vetor tal que

$$q = q_1,\ q_2 = Aq_1,\ q_3 = Aq_2 = A^2 q_1, ...,\ q_n = Aq_{n-1} = A^{n-1} q_1. \tag{2.45}$$

Deste modo, é possível computar iterativamente até n subespaços de Krylov de uma matriz $A_{n\times n}$ e de um vetor $q \in \mathbb{R}^n$ simultaneamente ao cômputo de T ou H tal que

$K(q_1, A, n) = [q_1, Aq_1, ..., A^{n-1}] = Q = [q_1, ..., q_n]$ onde $T = Q^T A Q$ ou $H = Q^T A Q$.

O algoritmo de Lanczos consiste em construir iterativamente a matriz tridiagonal T_k (similar à matriz A) a partir da base $q_1, ..., q_k$ dos sucessivos subespaços de Krylov, construídos iterativamente via multiplicação matriz-vetor. Analogamente, o algoritmo de Arnoldi consiste em construir a matriz H_k (similar à matriz A). Assim, considerando $Q = [Q_k, Q_u]$ onde $Q_k = [q_1, ..., q_k]$ e $Q_u = [q_{k+1}, ..., q_n]$ tem-se que

$$T = Q^T A Q = [Q_k, Q_u]^T A [Q_k, Q_u] = \begin{bmatrix} Q_k^T A Q_k & Q_k^T A Q_u \\ Q_u^T A Q_k & Q_u^T A Q_u \end{bmatrix} = \begin{bmatrix} T_{(k\times k)} & T_{(k\times n-k)} \\ T_{(n-k\times k)} & T_{(n-k\times n-k)} \end{bmatrix} \text{ e}$$

$$H = Q^T A Q = [Q_k, Q_u]^T A [Q_k, Q_u] = \begin{bmatrix} Q_k^T A Q_k & Q_k^T A Q_u \\ Q_u^T A Q_k & Q_u^T A Q_u \end{bmatrix} = \begin{bmatrix} H_{(k\times k)} & H_{(k\times n-k)} \\ H_{(n-k\times k)} & H_{(n-k\times n-k)} \end{bmatrix}.$$

Portanto, para computar boas aproximações dos k maiores autovalores da matriz T, basta construir iterativamente a matriz $T_{k\times k}$ por meio da aplicação do algoritmo de Lanczos 2.5.2.3 e, posteriormente, realizar a decomposição espectral $T_k = V_k^T \Lambda_k V_k$. Analogamente, para computar boas aproximações dos k maiores autovalores da matriz H, basta construir iterativamente a matriz $H_{k\times k}$ por meio da aplicação do algoritmo de Arnoldi 2.5.2.4 e, posteriormente, realizar a decomposição espectral $H_k = V_k^T T_k V_k$. Em ambos os casos, os vetores $q_1, ..., q_k$, geradores dos iterativos subespaços de Krylov, são utilizados para efetuar o cálculo dos vetores de Ritz definidos pelo produto $Q_k V_k$.

2.5.2.3 Método de Lanczos

Dada a matriz tridiagonal descrita em (2.46), a idéia principal do algoritmo de Lanczos é computar diretamente os elementos da tridiagonal $T_k = Q_k^T A Q_k$ contida em T. Para tanto, dado $Q = [q_1, ..., q_k]$, as colunas de $AQ_k = Q_k T_k$ são equacionadas por

$$Aq_j = \beta_{j-1} q_{j-1} + \alpha_j q_j + \beta_j q_{j+1} \text{ onde } \beta_0 q_0 = 0 \text{ para } j = 1 : (n-1).$$

$$T = \left[\begin{array}{c|c} T_{(k\times k)} & T_{(k\times n-k)} \\ \hline T_{(n-k\times k)} & T_{(n-k\times n-k)} \end{array}\right] = \left[\begin{array}{ccccc|ccccc} \alpha_1 & \beta_1 & & & & & & & & \\ \beta_1 & \ddots & \ddots & & & & & & & \\ & \ddots & \ddots & \beta_{k-1} & & & & & & \\ & & \beta_{k-1} & \alpha_k & \beta_k & & & & & \\ \hline & & & \beta_k & \alpha_{k+1} & \beta_{k+1} & & & & \\ & & & & \beta_{k+1} & \ddots & \ddots & & & \\ & & & & & \ddots & \ddots & \beta_{n-1} & & \\ & & & & & & \beta_{n-1} & \alpha_n & & \end{array}\right] \tag{2.46}$$

Vale ressaltar que a ortonormalidade de q_j implica em $q_j^T A q_j = \alpha_j$. No entanto, erros de arredondamento podem comprometer o comportamento do algoritmo, causando a perda de ortogonalidade dos vetores de Lanczos $(q_1, ..., q_k)$, fenômeno que complica a proximidade entre os autovalores das matrizes A e T. Demmel [7] mostra que uma saída é realizar a reortogonalização completa por meio do processo de Gram-Schimidt. Além disso, outros tipos de reortogonalização podem ser encontrados em [7] e [15].

Algoritmo 2: O algoritmo de Lanczos - Algoritmo 7.1, [7].

Entrada: Matriz simétrica $A_{m\times m}$, vetor $b \in \mathbb{R}^m$ e k.
Saída: Matriz tridiagonal e vetores $q_1, ..., q_k$ do subespaço de Krylov.

início
  $q_1 = b/||b||_2$, $\beta_0 = 0$, $q_0 = 0$
  para cada $j \leftarrow 1$ *até* k **faça**
    $z = Aq_j$
    $\alpha_j = q_j^T z$
    $z = z - \alpha_j q_j - \beta_{j-1} q_{j-1}$
    $\beta_j = ||z||_2$
    se $\beta_j == 0$ **então**
      pare
    $q_{j+1} = z/\beta_j$
fim

Considerando que cada passo consiste em uma multiplicação matriz-vetor, um produto interno e um par de operações vetoriais, o algoritmo é computado com complexidade $O(n^2)$. Pelo Exemplo 2.20 é possível verificar que o algoritmo gera iterativamente a matriz similar T por meio da construção dos subespaços de Krylov via multiplicação matriz-vetor. Além disso, apesar de não realilzar a reortogonalização, é possível perceber com uma precisão razoável (10^{-8}) a ortogonalidade entre os vetores de Lanczos.

Exemplo 2.20 [Lanczos sem reortogonalização]

```
--> //---------------------------------------------------------------------------
--> // Entradas: A = matriz simétrica, q = vetor inicial e k = número de iterações
--> // Saídas:   T = matriz tridiagonal de ordem k e Qs vetores do subspaço de Krylov
--> //---------------------------------------------------------------------------
--> A = [1 0 3 0 7 0; 0 3 2 0 0 1; 3 2 5 1 0 0; 0 0 1 6 3 0; 7 0 0 3 4 8; 0 1 0 0 8 2];
--> q = A(:,1); q = q/norm(q);
--> Qs = q;
--> k = 6;
--> for i=1:k
-->     z = A*q;
-->     Alpha(i) = q'*z;
-->     z = z - Alpha(i)*q;
-->     if (i>1)
-->         z = z - Beta(i-1)*Qs(:,i-1);
-->     end
-->     Beta(i) = norm(z);
-->     q = z/Beta(i);
-->     Qs = [Qs,q];
--> end
```

```
--> T = diag(Alpha(1:k)) + diag(Beta(1:k-1),1) + diag(Beta(1:k-1),-1)
    T  =
    6.0677966    10.581186    0.           0.           0.           0.
    10.581186    0.0214454    3.0679247    0.           0.           0.
    0.           3.0679247    4.5144034    1.7055322    0.           0.
    0.           0.           1.7055322    4.4965996    1.3371224    0.
    0.           0.           0.           1.3371224    1.9754825    2.8406764
    0.           0.           0.           0.           2.8406764    3.9242726
--> spec(T)'
    ans  = - 8.4371534  - 0.3353623    3.0091693    5.6193029    6.72917    14.414874
--> spec(A)'
    ans  = - 8.4371534  - 0.3353623    3.0091693    5.6193029    6.72917    14.414874
--> round((Qs(1:k,1:k)'*Qs(1:k,1:k))*1e8)*1e-8
    ans  =
    1.           0.           0.           0.           0.           0.
    0.           1.           0.           0.           0.           0.
    0.           0.           1.           0.           0.           0.
    0.           0.           0.           1.           0.           0.
    0.           0.           0.           0.           1.           0.
    0.           0.           0.           0.           0.           1.
```

■

Por outro lado, Demmel [7] mostra que os autovalores extremos (maiores e menores) convergem primeiro, enquanto os interiores convergem por último. Essa convergência é monotônica, ou seja, os autovalores de T crescem ou decrescem estritamente em relação aos autovalores de A. O Exemplo 2.21 apresenta a convergência dos autovalores dos extremos e reflete a perda de ortogonalidade ocorrida no cálculo do autovalor 4.6286246.

Exemplo 2.21 [Resultado do Exemplo 2.20 para $k = 3$.]

```
--> T = diag(Alpha(1:k)) + diag(Beta(1:k-1),1) + diag(Beta(1:k-1),-1)
    T  =
    6.0677966    10.581186    0.
    10.581186    0.0214454    3.0679247
    0.           3.0679247    4.5144034
--> //Autovalores do extremo e a perda de ortogonalidade!
--> spec(T)'
    ans  =
  - 8.4289511    4.6286246    14.403972
--> spec(A)'
    ans  =
  - 8.4371534  - 0.3353623    3.0091693    5.6193029    6.72917    14.414874
```

■

2.5.2.4 Método de Arnoldi

Dada a matriz (2.47), a idéia principal do algoritmo de Arnoldi é computar diretamente os elementos da mesma. Para tanto, dado que $H = Q^T AQ$ e $Q = [q_1, ..., q_n]$, as colunas de $AQ = QH$ são inicialmente equacionadas por (2.48). Visto que q é vetor ortonormal, ao multiplicar ambos os lados de (2.48) por q_m^T chega-se a (2.49). Da mesma forma, para evitar perda de ortogonalidade deve-se aplicar algum tipo de reortogonalização.

$$H_{(k\times k)} = \begin{bmatrix} h_{1,1} & h_{2,1} & \cdots & h_{1,k} \\ h_{2,1} & \ddots & \ddots & \vdots \\ & \ddots & \ddots & h_{k-1,k} \\ & & h_{k,k-1} & h_{k,k} \end{bmatrix} \tag{2.47}$$

$$Aq_j = \sum_{i=1}^{j+1} h_{i,j} q_i \tag{2.48}$$

$$q_m^T Aq_j = \sum_{i=1}^{j+1} h_{(i,j)} q_m^T q_i = h_{m,j} \ (1 \leq m \leq j) \ \rightarrow \ h_{(j+1,j)} q_{(j+1)} = Aq_j - \sum_{i=1}^{j} h_{(i,j)} q_i. \tag{2.49}$$

Algoritmo 3: O algoritmo de Arnoldi - Algoritmo 6.9, [7].

Entrada: Matriz não simétrica $A_{m\times m}$, vetor $b \in \mathbb{R}^m$ e k.
Saída: Matriz Hessenberg e vetores $q_1, ..., q_k$ do subespaço de Krylov.

```
início
    q_1 = b/||b||_2
    para cada j ← 1 até k faça
        z = A q_(j)
        para cada i ← 1 até j faça
            h_(i,j) = q_(i)^T z
            z = z − h_(i,j) q_(i)
        h_(j+1,j) = ||z||_2
        se h_(j+1,j) == 0 então
            pare
        q_(j+1) = z/h_(j+1,j)
fim
```

Considerando que cada passo consiste em uma multiplicação matriz-vetor, um produto interno, um par de operações vetoriais e um laço interno, se a matriz for esparsa, o produto matriz-vetor também pode ser computado rapidamente. No entanto, a complexidade desse algoritmo é da ordem de $O(n^2)$, como mostra o Exemplo 2.22.

Exemplo 2.22 [Arnoldi sem reortogonalização]

```
--> //-------------------------------------------------------------------------------
--> // Entradas: A = matriz não simétrica, q = vetor inicial e k = número de iterações
--> // Saídas:   H = matriz hessenberg de ordem k e vetores do subspaço de Krylov (Qs)
--> //-------------------------------------------------------------------------------
--> A = [1 0 3 0 7 0; 0 5 8 0 0 1; 3 2 9 2 0 0; 0 2 1 6 0 0; 7 0 0 3 4 0; 0 1 0 0 9 2];
--> q = A(:,1); q = q/norm(q);
--> Qs = q;
--> k = 6;
--> for j=1:k
-->     z = A*Qs(:,j);
-->     for i=1:j
-->         H(i,j) = Qs(:,i)'*z;
-->         z = z - H(i,j)*Qs(:,i);
-->     end
-->     H(j+1,j) = norm(z);
-->     if (H(j+1,j) == 0)
-->         break;
-->     end
-->     q = z/H(j+1,j);
-->     Qs = [Qs,q];
--> end
--> H = H(1:k,1:k)
    H  =
    6.6779661    4.8011209  - 0.3562360    6.1831165  - 0.4510534  - 1.3325821
    11.296378    1.1242261    2.5713782  - 1.8664603    0.2173053  - 0.0992242
    0.           3.4929125    8.681938     3.5385904    3.1679445    4.2640128
    0.           0.           2.9203215    3.3495899    2.6394056    1.0264124
    0.           0.           0.           0.9821639    1.3515603  - 2.8190331
    0.           0.           0.           0.           0.5992564    5.8147195
--> spec(H)'
    ans  = - 5.0835157    13.440031    1.1628329    8.0236983    4.5354312    4.9215221
--> spec(A)'
    ans  = - 5.0835157    13.440031    1.1628329    8.0236983    4.5354312    4.9215221
```

■

Capítulo 3

Decomposição em Valores Singulares

O método de Decomposição em Valores Singulares (*Singular Value Decomposition* - SVD) foi enunciado pela primeira vez para matrizes quadradas reais, de forma independente, pelos matemáticos Eugenio Beltrani em 1873 e Camille Jordan em 1874. Posteriormente, outros matemáticos, como por exemplo, James Joseph Sylvester (1889) e Autonne (1915), redescobriram o método. No entanto, somente em 1936 Carl Eckart e Gale Young generalizaram e provaram sua existência para matrizes retangulares e complexas [9]. Em relação aos principais métodos de cálculo, o primeiro deles, baseado nas rotações de Givens, surgiu com o matemático Ervand G. Kogbetliantz em 1958 [17]. Em 1965 foi publicado por Gene Howard Golub e William Morton Kahan, uma forma alternativa baseada nas reflexões de Householder [13]. Finalmente, em 1970 foi publicado por Gene Howard Golub e Christian Reinsch uma variante do algoritmo de Golub/Kahan que é , inclusive, o método mais utilizado no atual estado da arte [14].

O presente capítulo está organizado da seguinte forma: Seção 3.1 Definição da SVD (completa e reduzida), Seção 3.2 Interpretação geométrica, Seção 3.3 Subespaços fundamentais, Seção 3.4 Aproximação de matrizes, Seção 3.5 SVD *versus* Decomposição espectral, Seção 3.6 Propriedades e aplicações e Seção 3.7 Cálculo (direto e iterativo).

3.1 Definição

Por definição, a SVD é uma técnica de fatoração de matrizes que consiste em representar qualquer matriz $A_{m\times n}$ na forma (3.1), onde $U_{m\times m}$ e $V^T_{n\times n}$ são matrizes ortogonais, ou seja, $UU^T = I_m$ e $VV^T = I_n$ e $\Sigma_{n\times n}$ é uma matriz diagonal que contém os valores singulares σ_j em ordem não crescente $\sigma_1 \geq \sigma_2 ... \geq \sigma_j \geq 0$ para $1 \leq j \leq \min(m, n)$,

$$A_{m\times n} = U_{m\times m} \begin{bmatrix} \Sigma_{n\times n} \\ 0_{(m-n)\times n} \end{bmatrix} V^T_{n\times n}. \tag{3.1}$$

O Teorema 3.1 [[32] Teorema 4.1] garante a existência e unicidade da SVD para qualquer matriz $A_{m\times n}$. Analogamente à fatoração QR, a SVD pode ser completa ou reduzida.

Teorema 3.1 *Qualquer matriz $A_{m\times n}$ possui uma decomposição em valores singulares (SVD) da forma $A = U\Sigma V^T$. Além disso, tem-se que os valores singulares da mesma são sempre únicos. Porém, os vetores singulares à direita e os vetores singulares à esquerda somente serão únicos se a matriz A for quadrada e possuir autovalores distintos.*

3.1.1 SVD completa

A forma completa da SVD de uma matriz $A_{m\times n}$ $(m \geq n)$ é composta pelas matrizes ortogonais $U_{m\times m}$ e $V^T_{n\times n}$ e pela matriz $\Sigma_{m\times n}$. Se a matriz $A_{m\times n}$ tiver posto completo $(r = n)$ existirá um conjunto de n vetores ortonormais de $U \in \mathbb{R}^m$ e um conjunto de n vetores ortonormais de $V^T \in \mathbb{R}^n$. No entanto, se $n < m$, o conjunto de n vetores $\in \mathbb{R}^m$ não será suficiente para formar uma base de $\mathbb{R}^m$ e $U_{m\times n}$ não será uma matriz quadrada ortogonal. Portanto, para obter a forma completa da SVD, será necessário a adição de $m-n$ colunas ortonormais em U para transformá-la em uma matriz ortogonal $(U_{m\times m})$ e $m - n$ linhas nulas em Σ para possibilitar a multiplicação $U_{m\times m}\Sigma_{m\times n}$. Por outro lado, se a matriz $A_{m\times n}$ possuir posto incompleto $(r < n)$, será necessário adicionar $m - r$ colunas ortonormais em U e $n - r$ linhas ortonormais em V^T para completar as bases dos espaços $\mathbb{R}^m$ e $\mathbb{R}^n$ respectivamente e transformar U e V em matrizes quadradas ortogonais. Além disso, será necessário adicionar $m-r$ linhas nulas e $n-r$ colunas nulas em Σ para possibilitar a representação $A_{m\times n} = U_{m\times m}\Sigma_{m\times n}V^T_{n\times n}$. Em outras palavras, qualquer matriz pode ser diagonalizada (Σ), desde que escolhamos um sistema de coordenadas ortogonal apropriado para R^m (colunas de U) e R^n (linhas de V).

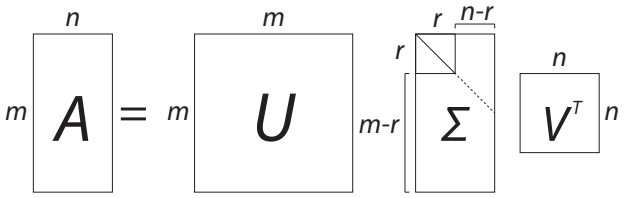


Figura 3.1: SVD completa de uma matriz $A_{m\times n}(m \geq n)$.

Exemplo 3.1 [SVD completa.]

```
--> A = [1 1; 1 2; 1 3; 1 4]; [U,S,V] = svd(A)
    V  =
    0.3220062  - 0.9467376
    0.9467376    0.3220062
    S  =
    5.7793788    0.
    0.           0.7738091
    0.           0.
    0.           0.
    U  =
    0.2195294  - 0.8073455    0.2236068    0.5
    0.3833425  - 0.3912142  - 0.6708204  - 0.5
    0.5471555    0.0249171    0.6708204  - 0.5
    0.7109685    0.4410485  - 0.2236068    0.5
```

■

3.1.2 SVD Reduzida

A forma reduzida da SVD de uma matriz $A_{m\times n}$ $(m \geq n)$ de posto r consiste na remoção das colunas e linhas extras colocadas com intuito de transformar $U_{m\times r}$ e $V_{r\times n}$ em matrizes ortogonais $U_{m\times m}$ e $V_{n\times n}$. Essa remoção tem por objetivo fornecer apenas a informação fornecida pelo conjunto de vetores definidos pelo posto da matriz.

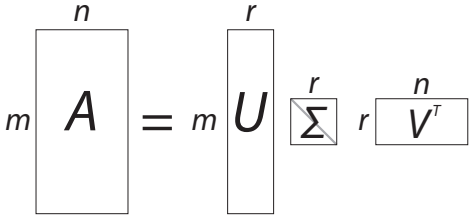


Figura 3.2: SVD reduzida de uma matriz $A_{m\times n}(m \geq n)$.

Exemplo 3.2 [SVD reduzida.]

```
--> [U,S,V] = svd(A,0)
    V  =
    0.3220062  - 0.9467376
    0.9467376    0.3220062
    S  =
    5.7793788    0.
    0.           0.7738091
    U  =
    0.2195294  - 0.8073455
    0.3833425  - 0.3912142
    0.5471555    0.0249171
    0.7109685    0.4410485
```

■

3.2 Interpretação geométrica

Segundo Trefethen [32], a SVD é motivada pelo fato geométrico de que a imagem de uma esfera unitária $S \in \mathbb{R}^n$ sob uma matriz $A_{m\times n}$ resulta numa hiperelipse $AS \in \mathbb{R}^m$. Por um lado, a esfera unitária é definida por vetores ortonormais $v_1, v_2, ..., v_n \in \mathbb{R}^n$. Por outro, a hiperelipse, considerada uma generalização de uma elipse, é obtida por meio do alongamento/achatamento da esfera unitária no espaço $\mathbb{R}^m$, onde sua representação passa a ser definida por vetores ortonormais $u_1, u_2, ..., u_m \in \mathbb{R}^m$ que apontam em direções ortogonais e escalares $\sigma_1, \sigma_2, ..., \sigma_m$, designados por valores singulares, comumente apresentados em ordem decrescente ($\sigma_1 \geq \sigma_2 ... \geq \sigma_m \geq 0$), tal que $u_i\sigma_i$ representam os semi-eixos de comprimento σ_i dessa hiperelipse. Segundo o autor, supondo que $A_{m\times n}$ possui posto completo (n) e que $m \geq n$, então as equações que envolvem a transformação da SVD dessa matriz podem ser descritas como $Av_j = u_j\sigma_j$ para $1 \leq j \leq n$:

$$\begin{bmatrix} & & \\ & A & \\ & & \end{bmatrix} \begin{bmatrix} & & \\ v_1 & \dots & v_n \\ & & \end{bmatrix} = \begin{bmatrix} & & \\ u_1 & \dots & u_n \\ & & \end{bmatrix} \begin{bmatrix} \sigma_1 & & \\ & \ddots & \\ & & \sigma_n \end{bmatrix}$$

Com isso, a equação $AV = U\Sigma$ implica na fatoração da matriz dada por $A = U\Sigma V^{-1}$. Além disso, dado que V é composta por uma base ortonormal de vetores, $V^{-1} = V^T$ e $A = U\Sigma V^T$. Dessa forma, os vetores unitários $u_1, u_2, ..., u_n$ orientados nas direções dos semi-eixos principais de AS são definidos como vetores singulares à esquerda e os vetores unitários $v_1, v_2, ..., v_n \in S$ são definidos como vetores singulares à direita. As Figuras 3.3 e 3.4 ilustram a SVD de matrizes $A_{2\times2}$ e $A_{2\times3}$, respectivamente.

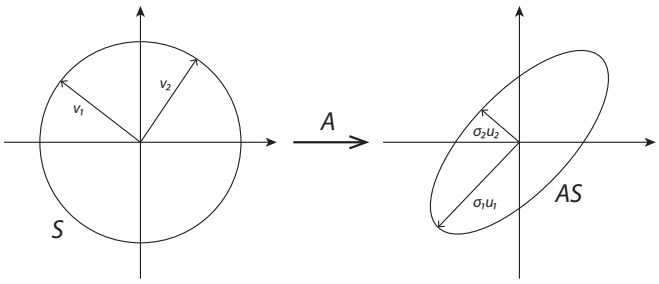


Figura 3.3: SVD de $A_{2\times2}$.

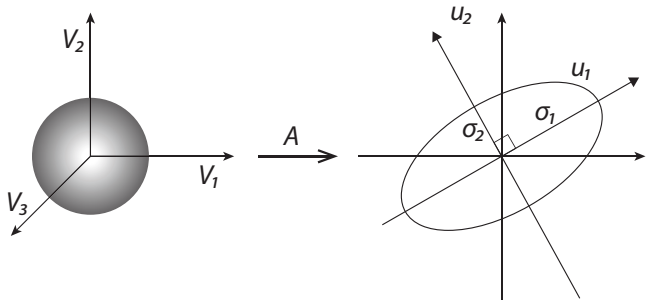


Figura 3.4: SVD de $A_{3\times2}$.

3.3 Subespaços fundamentais

Os principais subespaços associados a uma matriz $A_{m\times n}$ e a sua respectiva transposta $A^T_{n\times m}$ são geralmente definidos por: espaço-coluna de $A_{m\times n}$, espaço-coluna de $A^T_{n\times m}$, espaço-linha de $A_{m\times n}$, espaço-linha de $A^T_{n\times m}$, espaço-nulo de $A_{m\times n}$ e, por fim, o espaço-nulo de $A^T_{n\times m}$. No entanto, como o espaço-coluna de $A_{m\times n}$ equivale ao espaço-linha de $A^T_{n\times m}$ e o espaço-linha de $A_{m\times n}$ equivale ao espaço-coluna de $A^T_{n\times m}$, esses seis possíveis subespaços se reduzem aos quatro subespaços fundamentais abaixo:

- **Espaço-coluna de $A_{m\times n}$ ou espaço-linha de $A^T_{n\times m}$**: Subespaço $\in \mathbb{R}^m$ gerado pelos n vetores-coluna de $A_{m\times n}$ ou pelos n vetores-linha de $A^T_{n\times m}$.

- **Espaço-linha de $A_{m\times n}$ ou espaço-coluna $A^T_{n\times m}$**: Subespaço $\in \mathbb{R}^n$ gerado pelos m vetores-linha de $A_{m\times n}$ ou pelos m vetores-coluna de $A^T_{n\times m}$.
- **Espaço-nulo de $A_{m\times n}$:** Subespaço $\in \mathbb{R}^n$ dado pelo espaço-solução de $Ax = 0$.
- **Espaço-nulo de $A^T_{n\times m}$:** Subespaço $\in \mathbb{R}^m$ dado pelo espaço-solução de $A^Tx = 0$.

Esses quatro subespaços estão intimamente relacionados aos subconjuntos designados por núcleo e imagem de uma transformação linear matricial definidos na Seção 2.1. Em outras palavras, o espaço-nulo de A equivale ao núcleo de A e o espaço-nulo de A^T equivale ao núcleo de A^T. Além disso, dado que a imagem de uma matriz é o espaço gerado por suas próprias colunas [Teorema 1.1, [32]], o espaço-coluna de A é equivalente à imagem de A e o espaço-linha de A é equivalente à imagem de A^T. Por outro lado, considerando que o posto de uma matriz é a dimensão da sua imagem e a nulidade de uma matriz é a dimensão do seu núcleo, a partir dos Teoremas 3.2 [Teorema 5.6.2, [2]] e 3.3 [Teorema 5.6.3, [2]] é possível definir a dimensão de cada um dos subespaços fundamentais de uma matriz baseado em seu posto, como mostra a Tabela (3.1).

Teorema 3.2 *Sendo A uma matriz qualquer, então posto(A) = posto(A^T).*

Teorema 3.3 *Se $A_{m\times n}$ possuir n colunas, então posto(A) + nulidade(A) = n.*

Tabela 3.1: Dimensões dos subespaços fundamentais de uma matriz $A_{m\times n}$ de posto r

Subespaço fundamental	Dimensão
imagem de A	r
núcleo de A	$n-r$
imagem de A^T	r
núcleo de A^T	$m-r$

Lars Eldén [11] aponta a SVD como a técnica que apresenta claramente bases ortogonais referentes a esses subespaços fundamentais. Para tanto, os vetores resultantes da SVD de uma matriz $A_{m\times n}$ de posto r relacionados às bases dos subespaços envolvidos $\mathbb{R}^m$ e $\mathbb{R}^n$, representados por vetores ortonormais das matrizes $U_{m\times m}$ e $V^T_{n\times n}$ respectivamente, foram apresentados pelo próprio autor por meio do Teorema 3.4 [11, Teorema 6.4].

Teorema 3.4 (Subespaços fundamentais) *1. Os vetores singulares $u_1, u_2, ..., u_r$ constituem uma base ortonormal da $I(A)$ e posto(A) = dim(I(A)) = r.*

2. Os vetores singulares $v_{r+1}, v_{r+2}, ..., v_n$ constituem uma base ortonormal do $N(A)$ e dim(N(A)) = n − r.

3. Os vetores singulares $v_1, v_2, ..., v_r$ formam uma base ortonormal da $I(A^T)$ e posto(A^T) = dim($I(A^T)$) = r.

4. Os vetores singulares $u_{r+1}, u_{r+2}, ..., u_m$ constituem uma base ortonormal do $N(A^T)$ e dim($N(A^T)$) = m − r.

O Exemplo 3.3 ilustra a SVD de uma matriz de posto incompleto cuja terceira coluna é combinação linear da primeira e da segunda. Portanto, tem-se que a terceira coluna de V^T faz parte do núcleo de A e a terceira coluna de U faz parte do núcleo de A^T.

Exemplo 3.3 [SVD de matriz com posto incompleto (Núcleo de A e A^T).]

```
--> A = [1. 1. 1.5; 1. 2. 2.; 1. 3. 2.5; 1. 4. 3.];
--> [U,S,V] = svd(A,0);
--> round(U*1e8)*1e-8
    ans  =  0.2611886    0.7948462  - 0.5393039
            0.4031549    0.3707642    0.6477691
            0.5451213  - 0.0533178    0.3223734
            0.6870876  - 0.4773998  - 0.4308386
--> round(S*1e8)*1e-8
    ans  =  7.3943887    0.           0.
            0.           0.9072020    0.
            0.           0.           0.
--> round(V*1e8)*1e-8
    ans  =  0.2564854    0.6998363  - 0.6666667
            0.7372094  - 0.5877169  - 0.3333333
            0.6250901    0.4059778    0.6666667
--> // Terceira coluna de V faz parte do núcleo de A
--> (round((A*V(:,3))*1e8)*1e-8)'
    ans  =  0.  0.  0.  0.
--> // Terceira coluna de U faz parte do núcleo de A'
--> (round((A'*U(:,3))*1e8)*1e-8) '
    ans  =  0.  0.  0.
```

■

3.4 Aproximação de matriz

Apesar da maioria dos problemas matemáticos lidar com a suposição de que matrizes do tipo $A_{m\times n}$ possuem posto completo, ou seja, $posto = \min(m,n)$, por definição, tem-se que posto $\leq \min(m,n)$. No entanto, vale ressaltar que o mapeamento desse problema para computação numérica acarreta erros de arredondamento que dificulta a determinação exata do posto de uma matriz. Entretanto, apesar da dificuldade, devido a algumas propriedades da SVD é possível realizar essa aproximação de forma eficiente e, dessa forma, conseguir definir o posto de matrizes com determinado nível de precisão.

Assim como em (2.28) da decomposição espectral, uma matriz pode ser representada como uma soma de matrizes de posto 1 por meio da sua forma decomposta em valores singulares, como pode ser observado a seguir pelo Teorema 3.5 (Teorema 5.7, [32]).

Teorema 3.5 *Uma matriz $A_{m\times n}$ é uma soma de r (posto de A) matrizes de posto* 1*:*

$$A = \sum_{j=1}^{r} \sigma_j u_j v_j^T. \tag{3.2}$$

Supondo que uma matriz A pode ser representada pela soma de uma matriz de posto incompleto e de uma matriz de ruído: $A = A_0 + N$, onde esse ruído N é pequeno comparado com a matriz A_0, Lars Eldén [11] menciona que, tipicamente, os valores singulares de A irão apresentar um comportamento similar ao da Figura 3.5. O autor discute que se o posto exato for conhecido ou estimado pela inspeção dos valores singulares, pode-se remover o ruído e, com isso, realizar uma boa aproximação de A.

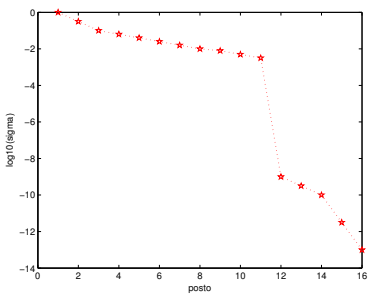


Figura 3.5: Valores singulares de uma matriz $A_{50\times 16}$ de posto 11.

Portanto, ao assumir ou estimar que o posto numérico de uma matriz é igual a r, sua aproximação é usualmente representada por (3.3), obtida pelo truncamento de (3.2):

$$A = \sum_{i=1}^{n} \sigma_i u_i v_i^T \approx \sum_{i=1}^{r} \sigma_i u_i v_i^T = A_k. \tag{3.3}$$

Além disso, essa aproximação, usualmente representada e validada a partir das normas matriciais 2 e de Frobenius, pode ser entendida a partir dos Teoremas 3.6 e 3.7 [Teoremas 6.6 e 6.7 [11]] e da Proposição 3.1 [Proposição 4.9 [11]], que indicam que essas normas são invariantes (preservadas) quando submetidas a transformações ortogonais.

Proposição 3.1 *Sejam $U_{m\times m}$ e $V_{n\times n}$ matrizes ortogonais. Então para qualquer $A_{m\times n}$:*

$$||UAV||_2 = ||A||_2 \quad e \quad ||UAV||_F = ||A||_F.$$

Exemplo 3.4 [Preservação das normas 2 e Frobenius sob transformações ortogonais.]

```
--> A = [23 1; 34 2; 2 1]
    A
    23. 1.
    34. 2.
    2. 1.
--> [U,S,V] = svd(A,0);
--> norm(A,2),norm(U'*A*V)
    ans  =  41.159731
    ans  =  41.159731
--> norm(A,'fro'),norm(U'*A*V, 'fro')
    ans  =  41.170378
    ans  =  41.170378
```

■

Teorema 3.6 *Considerando uma matriz $A_{m\times n}$ de posto $r > k$. A solução do problema de aproximar a matriz pela norma de Frobenius $\min_{posto(Z)=k} ||A - Z||_F$ é descrita como $Z = A_k = U_k \Sigma_k V_k^T$ onde o mínimo atingido é dado por $||A - A_k||_F = \sqrt{\sigma_{k+1}^2 + \sigma_{k+2}^2 + ... + \sigma_{min(m,n)}^2}$.*

Exemplo 3.5 [SVD de $A_{4\times4}$ (posto 2) - Norma de Frobenius para σ_{k+1}.]

```
--> A = [1 2 3 5; 1 3 4 7; 1 4 5 9; 1 5 6 11];
--> (round((svd(A,0))*1e8)*1e-8)'
    ans  =  20.483432    0.6549883    0.    0.
--> k = 3; Ak = U(:,1:k)*S(1:k,1:k)*V(:,1:k)';
--> round((norm(A - Ak,'fromc')*1e8)*1e-8)
    ans  =  0.
```

■

Teorema 3.7 *Considerando uma matriz $A_{m\times n}$ de posto $r > k$. A solução do problema de aproximar a matriz pela norma 2 $\min_{posto(Z)=k} ||A - Z||_2$ é descrita como $Z = A_k = U_k\Sigma_k V_k^T$ onde o mínimo atingido é dado por $||A - A_k||_2 = \sigma_{k+1}$.*

O Exemplo 3.6 mostra a norma calculada para os cortes de $k = 1, 2, 3$ e 4. Note que o valor de k igual ao posto da matriz aproxima bem dos valores da matriz original.

Exemplo 3.6 [SVD de $A_{4\times4}$ (posto 2) - Normas 2 para diferentes cortes (k).]

```
--> A = [1 2 3 5; 1 3 4 7; 1 4 5 9; 1 5 6 11];
--> [U,S,V] = svd(A);
--> diag(round(S*1e8)*1e-8)'
    ans  =  20.483432    0.6549883    0.    0.
--> k=1; Ak = U(:,1:k)*S(1:k,1:k)*V(:,1:k)', norm(A - Ak,2)
    Ak  =
    0.5858898    2.2310864    2.8169762    5.0480626
    0.8149291    3.1032753    3.9182044    7.0214798
    1.0439684    3.9754643    5.0194327    8.9948969
    1.2730078    4.8476532    6.1206609    10.968314
    ans  =  0.6549883
--> k=2; Ak = U(:,1:k)*S(1:k,1:k)*V(:,1:k)', norm(A - Ak,2)
    Ak  =
    1.    2.    3.    5.
    1.    3.    4.    7.
    1.    4.    5.    9.
    1.    5.    6.    11.
    ans  =  5.575D-15
--> k=3; Ak = U(:,1:k)*S(1:k,1:k)*V(:,1:k)'; norm(A - Ak,2)
    ans  =  5.565D-15
--> k=4; Ak = U(:,1:k)*S(1:k,1:k)*V(:,1:k)'; norm(A - Ak,2)
    ans  =  5.564D-15
```

■

3.5 SVD × Decomposição espectral

Existem pelo menos três diferenças fundamentais entre a decomposição espectral e a SVD. A primeira delas é que a decomposição espectral baseia-se em apenas uma base de autovetores, enquanto que a SVD baseia-se em duas diferentes bases, uma de vetores singulares à direita $\in \mathbb{R}^n$ (colunas de V) e outra à esquerda $\in \mathbb{R}^m$ (colunas de U). Por exemplo, no primeiro caso, uma matriz $A_{m\times m}$ não degenerada pode ser expressa por meio de uma matriz diagonal de autovalores D, se tanto a imagem quanto o domínio da transformação linear envolvida forem representados pela mesma base de autovetores. Ou seja, dada a decomposição $A = VDV^{-1}$, os vetores x e $b \in \mathbb{R}^m$ que satisfazem

$$b = Ax, \tag{3.4}$$

quando representados pela mesma base de autovetores onde $b' = V^{-1}b$ e $x' = V^{-1}x$, permitem que a relação (3.4) seja expressa em termos de b' e x'. Portanto, essa transformação linear se reduz à $b' = Dx'$ ao se multiplicar ambos os lados pela mesma base de vetores (V^{-1}) e substituir a matriz A pela decomposição VDV^{-1} como mostra (3.5).

$$b = Ax \quad \rightarrow \quad V^{-1}b = V^{-1}(VDV^{-1})x \quad \rightarrow \quad (V^{-1}b) = D(V^{-1}x) \quad \rightarrow \quad b' = Dx'. \tag{3.5}$$

Exemplo 3.7 [Mudança de base via decomposição espectral ($b = Ax \rightarrow b' = Dx'$).]

```
--> A = [1 2 3; 1 3 4; 1 4 5];
--> [V,D] = spec(A);
--> x = [1; 4; 5];
--> b = (A*x)'
    b  =
    24.  33.  42.
--> bl = inv(V)*b; xl = inv(V)*x;
--> bl'
    ans  =  58.721406  - 0.7228259  - 7.105D-14
--> (D*xl)'
    ans  =  58.721406  - 0.7228259  - 6.911D-16
```

■

Por outro lado, a mudança de bases realizada pela SVD consiste no fato de que qualquer vetor $b \in \mathbb{R}^m$ pode ser expresso pela base de vetores singulares à esquerda de A (colunas de U) e qualquer vetor $x \in \mathbb{R}^n$ pode ser expresso pela base de vetores singulares à direita de A (colunas de V). Isto é, $b' = U^T b$ e $x' = V^T x$. Dessa forma, pela decomposição $A = U\Sigma V^T$, a relação (3.4) pode ser expressa em termos de b' e x' como mostra (3.6).

$$b = Ax \quad \rightarrow \quad U^T b = U^T U \Sigma V^T x \quad \rightarrow \quad U^T b = \Sigma V^T x \quad \rightarrow \quad b' = \Sigma x'. \tag{3.6}$$

Portanto, a transformação linear (3.4) se reduz à transformação $b' = \Sigma x'$ quando a imagem ($b \in \mathbb{R}^m$) é expressa por meio da base formada pelas colunas de $U \in \mathbb{R}^{m \times m}$ e o domínio ($x \in \mathbb{R}^n$) é expresso por meio da base formada pelas colunas de $V \in \mathbb{R}^{n \times n}$.

Exemplo 3.8 [Mudança de base da SVD ($b = Ax \rightarrow b' = Sx'$).]

```
--> A = [1 2; 1 3; 1 4]
--> A  =
    1.    2.
    1.    3.
    1.    4.
--> [U,S,V] = svd(A,0);
--> x = [1; 4];
--> b = (A*x)'
    b  =
    9.    13.    17.
--> bl = U'*b';
--> xl = V'*x;
--> bl'
    ans  =  23.216143    0.1034302
--> (S*xl)'
    ans  =  23.216143    0.1034302
```

■

A segunda diferença entre as duas decomposições é que a SVD sempre utiliza bases vetoriais ortonormais, enquanto que na decomposição espectral, nem sempre a base de vetores gerada é sequer ortogonal. Por fim, a terceira e última principal diferença é que somente matrizes quadradas possuem decomposição espectral, ao contrário da SVD, que pode ser aplicada à qualquer matriz, inclusive, às retangulares.

3.6 Propriedades e aplicações da SVD

São várias as propriedades apresentadas pela SVD de uma matriz $A_{m\times n}$, bem como as aplicações da mesma dentro da própria Matemática. Algumas delas como por exemplo, a determinação de posto, a aproximação de posto incompleto e algumas normas matriciais, já foram apresentadas nas Seções 3.3 e 3.4. No entanto, existem outras propriedades e aplicações importantes, as quais se encontram a seguir listadas e ilustradas.

3.6.1 Propriedades algébricas

Teorema 3.8 (Teorema 5.1, [32] - Exemplo 3.9) *O posto r de uma matriz é igual ao número de valores singulares diferentes de zero.*

Exemplo 3.9 [Posto da matriz $A_{4\times 4}$.]

```
--> A = [1 2 3 5; 1 3 4 7; 1 4 5 9; 1 5 6 11];
--> [U,S,V] = svd(A);
--> diag(round(S*1e8)*1e-8)'
    ans  =  20.483432    0.6549883    0.    0.
```

■

Teorema 3.9 (Teorema 5.5, [32] - Exemplo 3.10) *Se $A = A^T$, então os valores singulares de A são os autovalores de A, em módulo.*

Exemplo 3.10 [Valores singulares = |autovalores| para $A = A^T$.]

```
--> A = [1 2 3; 2 1 4; 3 4 5]
    A  =
    1.    2.    3.
    2.    1.    4.
    3.    4.    5.
--> spec(A)'
    ans  =  - 1.4869798  - 0.5925456    9.0795254
--> [U,S,V] = svd(A,0);
--> diag(S)'
    ans  =    9.0795254    1.4869798    0.5925456
```

■

Teorema 3.10 (Teorema 3.3 (2) [7] - Exemplo 3.11) *Os autovalores da matriz simétrica A^TA são σ_i^2 de A. Os vetores singulares à direita v_i são os correspondentes autovetores ortonormais de A^TA.*

Exemplo 3.11 [Teorema 3.10.]

```
--> A = [1 2; 2 8; 1 5];
--> [Autovetor, Autovalor] = spec(A'*A) // autovalores e autovetores de A'*A
    Autovalor  =    0.2938012    0.
                    0.           98.706199
    Autovetor  =  - 0.9705758    0.2407956
                    0.2407956    0.9705758
--> [U, S, V] = svd(A,0);              // valores e vetores singulares de A
--> S.^2 // quadrado dos valores singulares
    ans  =  98.706199    0.
            0.           0.2938012
--> V    // vetores singulares à direita
    V  =
    - 0.2407956    0.9705758
    - 0.9705758  - 0.2407956
```

■

Teorema 3.11 (Teorema 3.3 (3) [7] - Exemplo 3.12) *Os autovalores da matriz simétrica AA^T são σ_i^2 de A mais $m-n$ zeros. Os vetores singulares à esquera u_i são os correspondentes autovetores ortonormais de AA^T correspondentes aos seus respectivos autovalores σ_i^2.*

Exemplo 3.12 [Teorema 3.11.]

```
--> A = [1 2; 2 8; 1 5];
--> [Autovetor, Autovalor] = spec(A*A') // autovalores e autovetores de A*A'
    Autovalor  =
    6.546D-15    0.           0.
    0.           0.2938012    0.
    0.           0.           98.706199
    Autovetor  =
  - 0.3713907    0.9021286    0.2196201
    0.5570860    0.0272804    0.8300066
  - 0.7427814  - 0.4306040    0.5126949
--> [U, S, V] = svd(A,0);              // valores e vetores singulares de A
```

```
--> S.^2 // quadrado dos valores singulares
    ans  =
    98.706199    0.
    0.           0.2938012
--> U    // vetores singulares à esquerda
    U  =
  - 0.2196201    0.9021286
  - 0.8300066    0.0272804
  - 0.5126949  - 0.4306040
```

■

Teorema 3.12 (Teorema 3.3 (4) [7] - Exemplo 3.13) *Seja* $H = \begin{bmatrix} 0 & A^T \\ A & 0 \end{bmatrix}$ *onde a SVD de* $A_{n\times n}$ *é dada por* $A = U\Sigma V^T$. *Então os* $2n$ *autovalores de* H *são os* $\pm\sigma_i$ *de* A, *com correspondentes autovetores unitários* $\frac{1}{\sqrt{2}} \begin{bmatrix} v_i & v_i \\ u_i & -u_i \end{bmatrix}$.

Exemplo 3.13 [Teorema 3.12.]

```
--> A = [1 3; 2 4]; H(3:4,1:2) = A; H(1:2,3:4) = A'
    H  =
    0.    0.    1.    2.
    0.    0.    3.    4.
    1.    3.    0.    0.
    2.    4.    0.    0.
--> [V,D] = spec(H); diag(D)'
    ans   =  - 5.4649857  - 0.3659662    0.3659662    5.4649857
--> V
    ans   =    0.2860626    0.6466593    0.6466593  - 0.2860626
               0.6466593  - 0.2860626  - 0.2860626  - 0.6466593
             - 0.4073278    0.5780001  - 0.5780001  - 0.4073278
             - 0.5780001  - 0.4073278    0.4073278  - 0.5780001
--> [U,S,V] = svd(A,0); diag(S)'
    ans     = 5.4649857    0.3659662
--> [V V; U -U]/sqrt(2)
    ans     = - 0.2860626    0.6466593  - 0.2860626    0.6466593
              - 0.6466593  - 0.2860626  - 0.6466593  - 0.2860626
              - 0.4073278  - 0.5780001    0.4073278    0.5780001
              - 0.5780001    0.4073278    0.5780001  - 0.4073278
```

■

Teorema 3.13 (Teorema 5.6, [32] - Exemplo 3.14) $|det(A_{m\times m})| = \prod_{i=1}^{m} \sigma_i$.

Exemplo 3.14 [Teorema 3.13.]

```
--> A = [1 2 3; 3 5 4; 5 6 7]; det(A)
    ans  =  - 12.
--> [U,S,V] = svd(A,0); prod(diag(S))
    ans  =    12.
```

■

Teorema 3.14 (Teorema 5.3, [32]) $||A||_2 = \sigma_1$ *e* $||A||_F = \sqrt{\sigma_1^2 + \sigma_2^2 + ...\sigma_r^2}$.

Exemplo 3.15 [Teorema 3.14.]

```
--> A = [1 2; 2 5; 3 5]; norm(A,2), norm(A,'fro')
    ans  =  8.2219581  // Norm 2
    ans  =  8.2462113  // Norm frobenius
--> [U,S,V] = svd(A,0);
--> diag(S)'
    ans  =  8.2219581    0.6319848
--> sqrt(diag(S)'*diag(S))
    ans  =  8.2462113
```

■

Teorema 3.15 (Teorema 6.10, [11] - Ver Exemplo 3.16) *Dada uma matriz* $A_{m\times n}$ *de posto completo e sua SVD* $A = U_1\Sigma V^T$ *onde* $U_1 = U_{m\times n}$. *Então o problema de quadrados mínimos* $min_x||Ax - b||$ *tem solução única:*

$$x = V\Sigma^{-1}U_1^T b = \sum_{i=1}^{n} \frac{u_i^T b}{\sigma_i} v_i.$$

Exemplo 3.16 [Teorema 3.15.]

```
--> A = [1 0.3; 1 2.7; 1 4.5; 1 5.9; 1 7.8]; b = [1.8 1.9 3.1 3.9 3.3]';
--> [U,S,V] = svd(A,0);
--> diag(S)'
    ans  =  11.267874   1.1467381
--> x = (V*inv(S)*U'*b)'
    x    =  1.6559415   0.2698251
```

■

3.6.2 Fatores latentes

Uma das principais aplicações relacionadas ao uso da SVD, principalmente, no contexto de Mineração de dados e Recuperação de informação está associada à possibilidade de se capturar fatores latentes, isto é, correlações não aparentes e não manifestadas exteriormente entre as entidades e os atributos. Essa propriedade, inclusive, explica o sucesso da SVD em vários outros cenários onde, por exemplo, as entidades podem ser não apenas documentos, mas sim usuários de uma rede social ou proteínas, enquanto os atributos podem ser além de termos, músicas e resíduos protéicos respectivamente. Lars Eldén [11] ilustra o significado numérico dessa captura por meio do Exemplo 3.17 onde correlaciona o conjunto dos cinco documentos com os respectivos termos marcados em negrito. A frequência de ocorrência dos termos é apresentada pela Tabela 3.2.

- *Documento 1*: The **Google matrix** P is a model of the **Internet**.
- *Documento 2*: P_{ij} is nonzero if there is a **link** from **Web page** j to i.
- *Documento 3*: The **Google matrix** is used to **rank** all **Web pages**.
- *Documento 4*: The **ranking** is done by solving a **matrix eigenvalue** problem.
- *Documento 5*: **England** dropped out of the top 10 in the **FIFA ranking**.

Tabela 3.2: Frequência que os termos marcados aparecem nos documentos.

Termo	Doc 1	Doc 2	Doc 3	Doc 4	Doc 5
eigenvalue	0	0	0	1	0
England	0	0	0	0	1
FIFA	0	0	0	0	1
Google	1	0	1	0	0
Internet	1	0	0	0	0
link	0	1	0	0	0
matrix	1	0	1	1	0
page	0	1	1	0	0
rank	0	0	1	1	1
Web	0	1	1	0	0

A modelagem consiste basicamente em considerar cada documento como um vetor ou um ponto no espaço $\mathbb{R}^{10}$, organizando-os em uma matriz de termos por documentos.

Para realizar uma consulta com intuito de retornar documentos relaciondos com os termos **ranking**, **Web** e **pages**, utiliza-se a representação do vetor de consulta q ($query$) dado por (3.7) análogo à matriz de termos por documentos dada pela Tabela 3.2.

$$q = [0\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 1\ 1]^T \tag{3.7}$$

O Exemplo 3.17 ilustra como recuperar ou capturar a informação referente à frequência da ocorrência desses termos na matriz de dados. Por exemplo, pelo vetor $ocorrencia = [0\ 2\ 3\ 1\ 1]$ resultante da multiplicação D^Tq, percebe-se que não existe uma correlação direta entre os termos de interesse **ranking**, **Web** e **pages** e o Documento 1.

Exemplo 3.17 [Ocorrência de termos específicos nos documentos $(d_1, d_2, d_3, d_4, d_5)$.]

```
--> // Frequência dos termos nos documentos.
--> D = [0 0 0 1 0; 0 0 0 0 1; 0 0 0 0 1; 1 0 1 0 0; 1 0 0 0 0;
         0 1 0 0 0; 1 0 1 1 0; 0 1 1 0 0; 0 0 1 1 1; 0 1 1 0 0]
    D  =
    0.    0.    0.    1.    0.
    0.    0.    0.    0.    1.
    0.    0.    0.    0.    1.
    1.    0.    1.    0.    0.
    1.    0.    0.    0.    0.
    0.    1.    0.    0.    0.
    1.    0.    1.    1.    0.
    0.    1.    1.    0.    0.
    0.    0.    1.    1.    1.
    0.    1.    1.    0.    0.
--> // Consulta para verificar ocorrência dos termos: ranking, Web e pages
--> q = [0; 0; 0; 0; 0; 0; 0; 1; 1; 1];
--> ocorrencia = (D'*q)'
    ocorrencia = 0.  2.  3.  1.  1.
```

■

Segundo Lars Eldén [11], outra forma de mensurar a correlação direta entre esses termos e os documentos é aplicar a similaridade de cosseno descrita em (3.8) entre o vetor consulta e os dados originais, ou seja, a similaridade de cosseno entre o vetor consulta q e cada documento d_i para $i = 1, ..., 5$, a qual resulta em [0 0.6667 0.7746 0.3333 0.3333].

$$cos(\theta) = \frac{q^T d_i}{||q||_2 ||d_i||_2} \tag{3.8}$$

Observe que analogamente ao Exemplo 3.17 não existe correlação direta entre os termos **ranking**, **Web** e **pages** e o Documento 1 e, como nenhum desses termos ocorre no Documento 1, o vetor que representa esse documento é ortogonal ao vetor consulta. No entanto, apesar disso, é possível perceber que os termos **Google** e **matrix** que ocorrem no Documento 1 também ocorrem no Documento 3, inclusive, onde ocorrem simultaneamente os três termos de interesse. Além disso, o termo **matrix** também ocorre no Documento 4, no qual ocorre o termo de interesse **rank**. Dessa forma, seria possível capturar esse grau de associação oculta entre esses termos e o Documento 1?

Lars Eldén [11] menciona que essa associação é corrompida tornando-se latente (oculta) devido à grande variedade de termos utilizados. No entanto, mostra como pode ser descoberta pela técnica designada por Indexação de Semântica Latente (LSI - Latent Semantic Indexing) ou Análise de Semântica Latente (LSA - Latent Semantic Analysis). Para tanto, mostra como capturá-la de forma robusta pela aplicação da SVD, onde a estrutura semântica pode ser descoberta e reforçada ou destacada pela projeção dos dados em um baixo espaço dimensional obtido pela aproximação da matriz D por (3.9).

$$D = U_k(\Sigma_k V_k^T) \approx U_k H_k \tag{3.9}$$

Enquanto as colunas de U_k formam uma base ortogonal usada para aproximar todos os documentos D, $H_k = (h_1, h_2, ..., h_k)$ é uma projeção de D em um subespaço gerado por U_k, ou seja, $H_k = D^T U_k$. Portanto, a coluna j de H_k possui as coordenadas do documento j em termos da base ortogonal. Logo, de $D \approx U_k H_k$ tem-se que $d_j = U_k h_j$.

Exemplo 3.18 [Aproximação da SVD de D para posto 2.]

```
>> [Uk,Sk,Vk] = svds(D,2)
    Uk =                       Sk =                Vk =
   -0.1425   -0.2430           2.8546    0         -0.3702   -0.1393
   -0.0787   -0.2607           0         1.8823    -0.2912    0.7030
   -0.0787   -0.2607                               -0.7498    0.1909
   -0.3924    0.0274                               -0.4068   -0.4573
   -0.1297   -0.0740                               -0.2246   -0.4908
   -0.1020    0.3735
   -0.5348   -0.2156
   -0.3647    0.4749
   -0.4838   -0.4023
   -0.3647    0.4749
```

■

Após feita a aproximação da matriz M por meio da SVD com posto igual a 2 como mostra o Exemplo 3.18, é possível observar o resultado da projeção no subespaço e a relação entre $D \approx U_k H_k$ e $d_j = U_k h_j$ por meio dos dois Exemplos 3.19 e 3.20.

Exemplo 3.19 [Projeção de D no subespaço gerado por U_k onde $k = 2$.]

```
>> [Uk,Sk,Vk] = svds(D,2); // k=2
>> Hk = Sk*Vk'
   Hk =
   -1.0569   -0.8314   -2.1404   -1.1612   -0.6412
    0.2622   -1.3232   -0.3592    0.8608    0.9238
>> (D'*Uk)'
    ans =
   -1.0569   -0.8314   -2.1404   -1.1612   -0.6412
    0.2622   -1.3232   -0.3592    0.8608    0.9238
```

■

Exemplo 3.20 [Relação entre $D \approx U_k H_k$ e $d_j = U_k h_j$]

```
>> // Decomposição completa D = U*H e dj = U*hj para j=1...5
>> [U,S,V] = svd(D);
>> H = S*V';
>> D = U*H;
>> d1 = (U*H(:,1))'
>> d1 = 0.0000    0.0000    0.0000    1.0000    1.0000
        0.0000    1.0000   -0.0000   -0.0000   -0.0000

>> // Norma de Frobenius
>> norm(D, 'fro')
   ans = 4.1231

>> // Aproximação: remove ruídos e contribui com o destaque dos fatores latentes!
>> [Uk,Sk,Vk] = svds(M,2);
>> Hk = Sk*Vk';
>> Dk = Uk*Hk;
>> d1 = (Uk*Hk(:,1))'
   d1 = 0.2143    0.1515    0.1515    0.4075    0.1565
        0.0099    0.6218    0.2609    0.6168    0.2609

>> // Aproximação da matriz D por Dk.
>> norm(Dk, 'fro')/norm(D, 'fro')
   ans = 0.8293
```

■

Com isso, a intuição por trás dessa aplicação está associada à possibilidade de expressar numericamente o grau da correlação existente entre os termos e os documentos de forma indireta, obtida por meio da nova representação dos vetores de documentos em função da base ortogonal U_k que melhor representa e se aproxima do espaço n-dimensional de $D_{m\times n}$ (m termos e n documentos). Portanto, como a aproximação da matriz de termos por documentos é representada por $D_k = U_k H_k$ e a consulta é computada por $q^T D_k = q_k^T U_k H_k = (U_k^T q)^T H_k$, as coordenadas da consulta em termos da nova base de documentos e o cálculo da similaridade de cosseno se encontram em (3.10).

$$cos(\theta_j) = \frac{q_k^T h_j}{||q_k||_2 ||h_j||_2} \qquad q_k = U_k^T q. \tag{3.10}$$

Finalmente, o cálculo da similaridade entre o vetor consulta q_k e todos os documentos $d_j = U_k h_j$ ($j = 1, 2, ...5$) foi capaz de capturar a correlação latente existente entre os termos **ranking**, **Web** e **pages** e todos os documentos. Observe pelo Exemplo 3.21 que com base nessa nova representação foi possível obter numericamente o grau de correlação oculta e indireta entre os termos de interesse e os documentos, inclusive, capaz de descobrir a alta correlação existente entre esses termos e o Documento 1.

Exemplo 3.21 [Correlação entre **ranking**, **Web** e **pages** e todos os documentos.]

```
>> qk  = Uk'*q; qk'
   ans = -1.2132   -0.5474
>> for i=1:5
>>     (qk'*Hk(:,i))/(norm(qk)*norm(Hk(:,i)))
>> end
ans = 0.7857
ans = 0.8332
ans = 0.9670
ans = 0.4873
ans = 0.1819
```

■

A principal diferença entre as duas consultas pode ser melhor entendida pela ordem de retorno dos documentos, os quais foram correlacionados com os termos de interesse. Por exemplo, a ordem de retorno da consulta simples é Doc 3, Doc 2, Doc 4, Doc 5 e, por fim, Doc 1. Por outro lado, a consulta após o uso da SVD retorna, em ordem, Doc 3, Doc 2, Doc 1, Doc 4 e Doc 5. Dessa forma, a correlação oculta existente entre os termos **ranking**, **Web** e **pages** e o Doc 1 somente é revelada após o uso da SVD.

3.7 Cálculo da SVD

Existem relações importantes entre a decomposição em valores singulares e a decomposição de Schur das respectivas matrizes simétricas descritas em (3.11). Golub [15] mostra que se $U^TAV = \text{diag}(\sigma_1, ..., \sigma_n)$ é a SVD de $A_{m\times n}(m \geq n)$ então (3.12) acontece.

$$A^TA, \quad AA^T \quad \text{e} \quad H = \begin{bmatrix} 0 & A^T \\ A & 0 \end{bmatrix}. \tag{3.11}$$

$$V^T(A^TA)V = \text{diag}(\sigma_1^2, ..., \sigma_n^2) \in \mathbb{R}^{n\times n} \quad e \quad U^T(AA^T)U = \text{diag}(\sigma_1^2, ..., \sigma_n^2, \underbrace{0, ..., 0}_{m-n}) \in \mathbb{R}^{m\times m}. \tag{3.12}$$

Exemplo 3.22 [SVD de A e Decomposição espectral de A^TA e AA^T - Parte 1]

```
--> A = [2 4 2; 3 23 3; 32 2 2; 21 1 2] // Matriz A (m = 4 > n = 3)
--> [U,S,V] = svd(A);
--> diag(S)', diag(S'*S)'
    ans  =
    38.948726   23.017822    1.4753364
    ans  =
    1517.0033   529.82012    2.1766174
--> round((V'*(A'*A)*V)*1e8)*1e-8
    ans  =
    1517.0033    0.           0.
    0.           529.82012    0.
    0.           0.           2.1766175
--> spec(A'*A)'
    ans  =  2.1766174  529.82012  1517.0033
--> round((U'*(A*A')*U)*1e8)*1e-8
    ans  =
    1517.0033    0.           0.           0.
    0.           529.82012    0.           0.
    0.           0.           2.1766175    0.
    0.           0.           0.           0.
--> spec(A*A')'
    ans  =  2.1766174  529.82012  1517.0033
```

■

Além disso, o autor menciona que dado $U = [U1_{m\times n}\, U2_{m\times(m-n)}]$ e a matriz ortogonal

$$Q = 1/\sqrt{2}\begin{bmatrix} V & V & 0 \\ U1 & -U1 & \sqrt{2}U2 \end{bmatrix}. \tag{3.13}$$

Então,

$$Q^T\begin{bmatrix} 0 & A^T \\ A & 0 \end{bmatrix} Q = \text{diag}(\sigma_1, ..., \sigma_n, -\sigma_1, ..., -\sigma_n, \underbrace{0, ..., 0}_{m-n}). \tag{3.14}$$

Exemplo 3.23 [SVD de A e decomposição espectral de H - Parte 2]

```
--> A = [2 4 2; 3 23 3; 32 2 2; 21 1 2]
    A  =
    2.     4.     2.
    3.     23.    3.
    32.    2.     2.
    21.    1.     2.
--> [U,S,V] = svd(A);
--> diag(S)'
    ans =   38.948726   23.017822   1.4753364
--> H(4:7,1:3) = A;
--> H(1:3,4:7) = A';
--> H
    H  =
    0.     0.     0.    2.    3.     32.    21.
    0.     0.     0.    4.    23.    2.     1.
    0.     0.     0.    2.    3.     2.     2.
    2.     4.     2.    0.    0.     0.     0.
    3.     23.    3.    0.    0.     0.     0.
    32.    2.     2.    0.    0.     0.     0.
    21.    1.     2.    0.    0.     0.     0.
--> U1 = U(:,1:3); U2 = U(:,4);
--> Q(1:3,1:3) = V; Q(1:3,4:6) = V; Q(1:3,7) = 0;
--> Q(4:7,1:3) = U1; Q(4:7,4:6) = -U1; Q(4:7,7) = sqrt(2)*U2;
--> // Autovetores da decomposição espectral!
--> Q = 1/sqrt(2)*Q;
--> round(diag(Q'*H*Q)'*1e8)*1e-8
    ans =   38.948726   23.017822   1.4753364   -38.948726   -23.017822   -1.4753364   0.
```

■

Dessa forma, dado (3.12), as etapas da SVD de $A_{m\times n}$ onde $m \geq n$ resumem-se em:

- Calcular $C = A^T A$;
- Computar a decomposição espectral $V^T CV = \text{diag}(\sigma_i^2)$ via Iteração QR (simétrica);
- Resolver $U\Sigma = AV$ para matriz U via fatoração QR com pivotação de colunas.

Exemplo 3.24 [Etapas tradicionais - Algoritmo SVD - [15] Algoritmo 8.3.2]

```
--> A = [2 4 2; 3 23 3; 32 2 2; 21 1 2]; [ma,na] = size(A);
--> C = A'*A; [nc,nc] = size(C);    // Cálculo de A'*A
--> V = eye(nc,nc);
--> k = 20;
--> for i=1:k
-->     [Q,R] = qr(C);
-->     C = R*Q;
-->     V = V*Q;                    // Cálculo de V via fatoração QR
--> end
--> S = eye(ma,na);
--> ss = sqrt(diag((V'*(A'*A)*V))); // Valores singulares de sqrt(V'*C*V)
--> for i=1:length(ss)
-->     S(i,i) = ss(i);
--> end
--> S
    S  =
    38.948726    0.           0.
    0.           23.017822    0.
    0.           0.           1.4753364
    0.           0.           0.
--> U = eye(ma,ma); A = A*V;
--> for i=1:k
-->     [Q,A,E] = qr(A);
-->     U = U*Q; // Cálculo de U via fatoração QR de AV com pivotação de colunas
--> end
--> U*S*V'       // Verificando SVD de A = U*S*V'
    ans  =
    2.     4.     2.
    3.     23.    3.
    32.    2.     2.
    21.    1.     2.
```

■

No entanto, apesar da simplicidade do algoritmo apresentado a partir do Exemplo 3.24, devido aos erros de arrendondamento e à perda de informação causada pelos cálculos mencionados, principalmente, pelo cálculo do produto A^TA, a principal forma de se computar a SVD apresentada por Golub e Karan em 1965 [13], apontada por vários autores como sendo a mais estável das formas existentes, inclusive, implementada pelo LAPACK e ARPACK, baseia-se na matriz simétrica H descrita em (3.11). A idéia consiste no fato de que as igualdades $A = U\Sigma V^T \rightarrow AV = U\Sigma$ e $A^TU = V\Sigma^T = V\Sigma$ podem ser arranjadas pela forma (3.15), a qual corresponde à decomposição espectral.

$$\begin{bmatrix} 0 & A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} V & V \\ U & -U \end{bmatrix} = \begin{bmatrix} V & V \\ U & -U \end{bmatrix} \begin{bmatrix} \Sigma & 0 \\ 0 & \Sigma \end{bmatrix}. \tag{3.15}$$

Pelo Teorema 3.12 e pelo Exemplo 3.23 verifica-se que os valores singulares de A são os valores absolutos dos autovalores de H e os vetores singulares de A são obtidos facilmente a partir dos autovetores de H. Entretanto, o algoritmo de Golub e Karan não forma H explicitamente. Basicamente, a técnica utiliza-se de duas fases. Enquanto a primeira consiste em reduzir A para forma bidiagonal com intuito de reduzir a complexidade do algoritmo final, a segunda consiste em diagonalizar a matriz bidiagonal resultante da primeira etapa, na qual aparecem os valores singulares e, simultaneamente, computar os respectivos autovetores, como representado pelo esquema (3.16).

$$\underbrace{\begin{bmatrix} \times & \times & \times & \times \\ \times & \times & \times & \times \\ \times & \times & \times & \times \\ \times & \times & \times & \times \\ \times & \times & \times & \times \end{bmatrix}}_{A_{m\times n}} \overset{Etapa\ 1}{\rightarrow} \underbrace{\begin{bmatrix} \times & \times & & \\ & \times & \times & \\ & & \times & \times \\ & & & \times \\ & & & \end{bmatrix}}_{Bidiagonal} \overset{Etapa\ 2}{\rightarrow} \underbrace{\begin{bmatrix} \times & & & \\ & \times & & \\ & & \times & \\ & & & \times \\ & & & \end{bmatrix}}_{Diagonal} \tag{3.16}$$

Entretanto, os métodos dependem do tipo da matriz, densa (método direto) ou esparsa (método iterativo). Demmel [7] apresenta vários métodos diretos, como por exemplo, Iteração QR, Divisão e conquista, Jacobi, entre outros. Entre os principais se encontram QR e Divisão e conquista que, assim, como no cálculo de autovalores, é mais adequado para o cálculo em matrizes de dimensão aproximadamente maior que 25.

3.7.1 Método direto

O método direto de cálculo da SVD de $A_{m\times n}(n \geq n)$ via algoritmo de Golub e Karan é realizado por meio das duas etapas apresentadas em (3.16).

Etapa 1) Reduzir a matriz A para forma bidiagonal por uma série de transformações de similaridade (Householder) pela direita e pela esquerda, onde U e V são calculadas por $U = U_1U_2...U_n$ e $V = V_1V_2...V_{n-2}$ de acordo com o esquema apresentado a seguir. (O Exemplo (3.25) ilustra o cômputo implícito de U e V na etapa de bidiagonalização).

$$\underbrace{\begin{bmatrix} \times & \times & \times & \times \\ \times & \times & \times & \times \\ \times & \times & \times & \times \\ \times & \times & \times & \times \\ \times & \times & \times & \times \end{bmatrix}}_{A} \xrightarrow{U_1^T} \underbrace{\begin{bmatrix} \times & \times & \times & \times \\ 0 & \times & \times & \times \\ 0 & \times & \times & \times \\ 0 & \times & \times & \times \\ 0 & \times & \times & \times \end{bmatrix}}_{U_1^TA} \xrightarrow{V_1} \underbrace{\begin{bmatrix} \times & \times & 0 & 0 \\ & \times & \times & \times \\ & \times & \times & \times \\ & \times & \times & \times \\ & \times & \times & \times \end{bmatrix}}_{U_1^TAV_1} \xrightarrow{U_2^T}$$

$$\underbrace{\begin{bmatrix} \times & \times & & \\ & \times & \times & \times \\ & 0 & \times & \times \\ & 0 & \times & \times \\ & 0 & \times & \times \end{bmatrix}}_{U_2^TU_1^TAV_1} \xrightarrow{V_2} \underbrace{\begin{bmatrix} \times & \times & & \\ & \times & \times & 0 \\ & & \times & \times \\ & & \times & \times \\ & & \times & \times \end{bmatrix}}_{U_2^TU_1^TAV_1V_2} \xrightarrow{U_3^T} \underbrace{\begin{bmatrix} \times & \times & & \\ & \times & \times & \\ & & \times & \times \\ & & 0 & \times \\ & & 0 & \times \end{bmatrix}}_{U_3^TU_2^TU_1^TAV_1V_2} \xrightarrow{U_4^T} \underbrace{\begin{bmatrix} \times & \times & & \\ & \times & \times & \\ & & \times & \times \\ & & & \times \\ & & & \end{bmatrix}}_{B=U_4^TU_3^TU_2^TU_1^TAV_1V_2}$$

Etapa 2) Diagonalizar iterativamente a matriz tridiagonal resultante de B^TB aplicando a Iteração QR com pivotação de colunas sobre a mesma, sem formá-la explicitamente.

$$\underbrace{\begin{bmatrix} \times & \times & & \\ & \times & \times & \\ & & \times & \times \\ & & & \times \\ & & & \end{bmatrix}}_{Bidiagonal} \xrightarrow{Etapa\ 2} \underbrace{\begin{bmatrix} \times & & & \\ & \times & & \\ & & \times & \\ & & & \times \\ & & & \end{bmatrix}}_{Diagonal}$$

Os valores singulares aparecem na diagonal e os vetores singulares à direita e à esquerda são computados por meio da Iteração QR de A e A^T, como mostra o Exemplo 3.26.

Exemplo 3.25 [Etapa 1) - Bidiagonalização de Householder - Algoritmo 5.4.2, [15]]

```
--> function [v] = house(x) // Vetor Householder
-->     n = length(x);
-->     mu = norm(x);
-->     v = x;
-->     if mu != 0
-->         bt = x(1) + sign(x(1))*mu;
-->         v(2:n) = v(2:n)/bt;
-->     end;
-->     v(1) = 1;
--> endfunction
--> function A = rowhouse(A,v) // Pré-multiplicação Householder
-->     bt = -2/(v'*v);
-->     w = bt*A'*v;
-->     A = A + v*w';
--> endfunction
--> function A = colhouse(A,v) // Pós-multiplicação Householder
-->     bt = -2/(v'*v);
-->     w = bt*A*v;
-->     A = A + w*v';
--> endfunction
--> A = [2 4 2; 3 23 3; 32 2 2; 21 1 2];
--> svd(A)'
    ans  =  38.948726    23.017822    1.4753364
--> [m,n] = size(A);
--> for j=1:n  // As matrizes U e V são utilizadas de forma implícita (menor custo)!
-->     v(j:m) = house(A(j:m,j));
-->     A(j:m,j:n) = rowhouse(A(j:m,j:n), v(j:m));
-->     if(j <= (n-2))
-->         v(j+1:n) = house(A(j,j+1:n)');
-->         A(j:m,j+1:n) = colhouse(A(j:m,j+1:n), v(j+1:n));
-->     end
--> end
--> round(A*1e8)*1e-8
    ans  =
  - 38.444766    5.2285388    0.
    0.           20.407404    11.151596
    0.           0.           1.685868
    0.           0.           0.
--> svd(A)' // Verificando que a SVD de A é similar à SVD da forma reduzida bidiagonal!
    ans  =  38.948726    23.017822    1.4753364
```

■

Exemplo 3.26 [Etapa 2) - Iteração QR com pivotação de colunas (versão explícita)]

```
--> // Continuação do Exemplo (3.3) - Matriz A agora é bidiagonal!
--> U = eye(m,m); V = eye(n,n);
--> S = A';
--> k=25;
--> for i=1:k
-->    [Q,S,E]=qr(S'); // Cálculo dos vetores singulares à esquerda (qr(A))
-->    U=U*Q;
-->    [Q,S,E]=qr(S'); // Cálculo dos vetores singulares à direita  (qr(A'))
-->    V=V*Q;
--> end
--> ss=diag(S);
--> S=zeros(m,n);
--> for n=1:length(ss)
-->     S(n,n) = abs(ss(n));
--> end
--> round((U'*U)*1e8)*1E-8 // Ortogonalidade da matriz U
    ans  =
    1.    0.    0.    0.
    0.    1.    0.    0.
    0.    0.    1.    0.
    0.    0.    0.    1.
--> round((V'*V)*1e8)*1E-8 // Ortogonalidade da matriz V
    ans  =
    1.    0.    0.
    0.    1.    0.
    0.    0.    1.
--> S // Valores singulares de A
    S  =
    38.948727    0.           0.
    0.           23.017822    0.
    0.           0.           1.4753364
    0.           0.           0.
--> round((U*S*V')*1e8)*1e-8 // A = USV'
    ans  =
  - 38.444766    5.2285388    0.
    0.           20.407404    11.151596
    0.           0.           1.685868
    0.           0.           0.
--> svd(U*S*V')'
    ans  =  38.948727    23.017822    1.4753364
```

■

3.7.2 Método iterativo

Segundo Lars Eldén [11], foi provado em [6] que o procedimento de bidiagonalização de A é equivalente ao procedimento de tridiagonalização de Lanczos sobre H. Assim, os valores e vetores singulares de A podem ser computados via Lanczos seguido de QR. Inclusive, esse processo iterativo da SVD é implementado pela função *svds* do MATLAB e *sva* do SCILAB, onde aplica-se Lanczos sobre H e, em seguida, QR sobre T.

Exemplo 3.27 [Etapa 1) - Tridiagonalização de H via Lanczos]

```
--> A = [2 4 2; 3 23 3; 32 2 2; 21 1 2]; H(4:7,1:3) = A; H(1:3,4:7) = A'; // Matriz H
--> svd(H)'
    ans  =
    38.948726  38.948726  23.017822  23.017822  1.4753364  1.4753364  2.122D-15
--> q = H(:,1); q = q/norm(q); Qs = q;
--> k = 7;
--> for i=1:k
-->     z = H*q;
-->     Alpha(i) = q'*z;
-->     z = z - Alpha(i)*q;
-->     if (i>1)
-->         z = z - Beta(i-1)*Qs(:,i-1);
-->     end
-->     Beta(i) = norm(z);
-->     if (Beta(i) == 0)
-->         break;
-->     end
-->     q = z/Beta(i);
-->     Qs = [Qs,q];
--> end
--> A = diag(Alpha(1:k)) + diag(Beta(1:k-1),1) + diag(Beta(1:k-1),-1)
    A  =
    0.          38.798681   0.          0.          0.          0.          0.
    38.798681   0.          2.7501168   0.          0.          0.          0.
    0.          2.7501168   0.          23.09236    0.          0.          0.
    0.          0.          23.09236    0.          0.8141272   0.          0.
    0.          0.          0.          0.8141272   0.          1.4762614   0.
    0.          0.          0.          0.          1.4762614   0.          2.867D-10
    0.          0.          0.          0.          0.          2.867D-10   0.
--> svd(A)' // Verificando similaridade com H
    ans  =
    38.948726  38.948726  23.017822  23.017822  1.4753364  1.4753364  1.547D-26
```

■

Exemplo 3.28 [Etapa 2) - Iteração QR sobre T]

```
--> // Obs: agora a matriz A é tridiagonal!
--> // O Código mostra uma forma possível de se utilizar a iteração QR
--> // para cálculo dos valores singulares
--> S = A'
    S  =
    0.         38.798681  0.         0.         0.         0.         0.
    38.798681  0.         2.7501168  0.         0.         0.         0.
    0.         2.7501168  0.         23.09236   0.         0.         0.
    0.         0.         23.09236   0.         0.8141272  0.         0.
    0.         0.         0.         0.8141272  0.         1.4762614  0.
    0.         0.         0.         0.         1.4762614  0.         2.867D-10
    0.         0.         0.         0.         0.         2.867D-10  0.
--> k = 25;
--> [nc,nc] = size(S);
--> V = eye(nc,nc);
--> for i=1:k
-->     [Q,S,E] = qr(S');
-->     V = V*Q;
--> end
--> ss = diag(S); S = zeros(nc,nc);
--> for n=1:length(ss)
-->     S(n,n)=ss(n);
--> end
--> S
    S  =
    38.948726  0.         0.         0.         0.         0.         0.
    0.         38.948726  0.         0.         0.         0.         0.
    0.         0.         23.017822  0.         0.         0.         0.
    0.         0.         0.         23.017822  0.         0.         0.
    0.         0.         0.         0.         1.4753364  0.         0.
    0.         0.         0.         0.         0.         1.4753364  0.
    0.         0.         0.         0.         0.         0.         2.009D-35
--> round((V*V')*1e8)*1e-8
    ans  =
    1.    0.    0.    0.    0.    0.    0.
    0.    1.    0.    0.    0.    0.    0.
    0.    0.    1.    0.    0.    0.    0.
    0.    0.    0.    1.    0.    0.    0.
    0.    0.    0.    0.    1.    0.    0.
    0.    0.    0.    0.    0.    1.    0.
    0.    0.    0.    0.    0.    0.    1.
```

■

Capítulo 4

Análise de Componentes Principais

A Análise de Componentes Principais (*Principal component analysis* - PCA), um dos métodos mais conhecidos de extração de características e análise de dados multivariados, foi desenvolvido por Karl Pearson em 1901 [26]. No entanto, foi consolidada por Harold Hotelling apenas em 1933 [19] e, por isso, também é conhecida por Transformada de Hotelling. Basicamente, o principal objetivo da PCA se resume em reduzir a dimensionalidade de dados multivariados de forma a preservar o máximo da informação contida nos mesmos. Inicialmente, esse método consiste em aplicar uma transformação linear com intuito de projetar os dados originais para um novo sistema de coordenadas, obtidas pelas combinações lineares das variáveis originais, as quais são designadas por componentes principais. Posteriormente, a redução é aplicada pela escolha do número de variáveis (componentes) a serem utilizadas. Em outras palavras, Sueli Mingoti [24] menciona que a PCA visa obter a redução do número de variáveis a serem avaliadas, bem como a interpretação das combinações lineares construídas, em que a informação contida nas variáveis originais é de fato substituída pela informação contida em um número reduzido de componentes principais. Por outro lado, a autora destaca que a qualidade dessa aproximação depende do número de componentes utilizadas e pode ser mensurada, por exemplo, por meio da avaliação da proporção da variância total explicada pelas mesmas, uma vez que a variância dos dados é o principal critério a ser maximizado na PCA. As duas formas de se calcular a PCA - por meio da matriz de covariância dos dados e por meio da matriz de correlação dos dados - serão apresentadas a seguir, bem como as propriedades, os exemplos ilustrativos, os testes de hipóteses, os principais critérios de determinação do número de componentes e as formas de cálculo.

4.1 PCA via matriz de covariância

A medida de covariância entre duas variáveis fornece o grau de relacionamento linear entre as mesmas. Assim, uma medida de covariância positiva significa que, linearmente, quando uma das variáveis cresce (decresce) a outra também cresce (decresce). Por outro lado, uma medida de covariância negativa significa que, linearmente, quando uma variável cresce (decresce) a outra decresce (cresce). Logo, a partir dessa definição, a medida de covariância entre duas variáveis é normalmente representada da seguinte forma:

Considerando $x = (x_1, x_2, ..., x_p)$ um vetor aleatório (A.1) seguido de seu respectivo vetor de médias (esperanças) $\mu = E(x) = (\mu_1, \mu_2, ..., \mu_p)$ (A.2), tem-se que a covariância entre os valores da i-ésima e j-ésima variáveis do vetor x é definida por:

$$Cov(x_i, x_j) = \sigma_{ij} = E[(x_i - \mu_i)(x_j - \mu_j)]. \tag{4.1}$$

Dessa forma, dado que a covariância entre um elemento x_i e ele mesmo é igual a sua variância (A.4), tem-se que a matriz Σ_x ($\Sigma_{p\times p}$) que sumariza as covariâncias entre as p variáveis do vetor original x é definida por:

$$\Sigma_x = \begin{bmatrix} E[(x_1-\mu_1)(x_1-\mu_1)] & E[(x_1-\mu_1)(x_2-\mu_2)] & \cdots & E[(x_1-\mu_1)(x_p-\mu_p)] \\ E[(x_2-\mu_2)(x_1-\mu_1)] & E[(x_2-\mu_2)(x_2-\mu_2)] & \cdots & E[(x_2-\mu_2)(x_p-\mu_p)] \\ \vdots & \vdots & \ddots & \vdots \\ E[(x_p-\mu_p)(x_1-\mu_1)] & E[(x_p-\mu_p)(x_2-\mu_2)] & \cdots & E[(x_p-\mu_p)(x_p-\mu_p)] \end{bmatrix}. \tag{4.2}$$

Além disso, é a partir da decomposição espectral de Σ_x (4.2) descrita por (4.3),

$$\Sigma_x = V_x \Lambda_x V_x^T = \sum_{i=1}^{p} \lambda_i v_i v_i^T, \tag{4.3}$$

onde a matriz diagonal Λ_x é composta pelos autovalores da matriz Σ_x ($\lambda_1 \geq \lambda_2 \geq ... \geq \lambda_p$) e a matriz V_x é composta por seus respectivos autovetores normalizados

$(v_1, v_2, ..., v_p)$, que definem-se as p componentes principais $(pc_1, pc_2, ..., pc_p)$ por meio de (4.4) e suas respectivas variâncias associadas por meio de (4.5). Assim, a matriz de covariâncias (4.2) fornece informação sobre as variâncias (autovalores) em torno dos eixos (autovetores) onde os dados estão distribuídos. Isso, é o mesmo que dizer que os autovetores de Σ_x apontam nas direções de maior variabilidade de x (dados originais).

$$pc_j = v_j^T x = v_{j1}x_1 + v_{j2}x_2 + ... + v_{jp}x_p \qquad j = 1, 2, ..., p, \tag{4.4}$$

$$\text{var}[pc_j] = \lambda_j \qquad j = 1, 2, ..., p. \tag{4.5}$$

Note que as componentes principais são compostas por combinações lineares das p variáveis aleatórias de x, apresentando a vantagem de serem variáveis não correlacionadas, o que facilita, inclusive, a interpretação das mesmas. Essa descorrelação está associada ao fato de que os autovetores de uma matriz simétrica (como é o caso de $\Sigma_{p\times p}$) serem ortogonais entre si. Portanto, o vetor x é descorrelacionado a partir da transformação linear (4.4). Além disso, vale ressaltar que as componentes compartilham da propriedade na qual a maior variância λ_1 está associada à primeira componente principal pc_1, a segunda maior variância λ_2 está associada com a segunda componente principal pc_2 e assim por diante (4.5).

Por sua vez, dado que os vetores x (variáveis originais) e pc (componentes principais) possuem a mesma variância total (A.6) e a mesma variância generalizada (A.7), como pode ser observado por (4.6) onde a matriz que fornece informação das variâncias em torno das componentes (Σ_{pc}) é equivalente à matriz que representa a variância dos dados originais (Λ_x), é possível reduzir o espaço das p variáveis originais pela aproximação do seu sistema de variabilidade, utilizando k componentes principais $(k < p)$.

$$\Sigma_{pc} = V_x^T \Sigma_x V_x = V_x^T \underbrace{V_x \Lambda_x V_x^T}_{\Sigma_x} V_x = \Lambda_x. \tag{4.6}$$

Entretanto, apesar da definição teórica da matriz de covariância apresentada em (4.1), na prática, essa matriz precisa ser estimada a partir dos elementos amostrais coletados, assim como o vetor de médias μ precisa ser estimado a partir do vetor de médias amostrais $\overline{x}$ (A.3). Para tanto, supondo uma matriz de dados $X_{n\times p}$ composta por

n vetores aleatórios constituídos de p variáveis aleatórias de interesse, sua respectiva matriz de covariância amostral $S_{p\times p}$ é definida em (4.7) pelas respectivas covariâncias amostrais descritas em (4.8).

$$X_{n\times p} = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1p} \\ x_{21} & x_{22} & \cdots & x_{2p} \\ \vdots & \vdots & \vdots & \vdots \\ x_{n1} & x_{n2} & \cdots & x_{np} \end{bmatrix} \quad \rightarrow \quad S_{p\times p} = \begin{bmatrix} s_{11} & s_{12} & \cdots & s_{1p} \\ s_{21} & s_{22} & \cdots & s_{2p} \\ \vdots & \vdots & \vdots & \vdots \\ s_{p1} & s_{p2} & \cdots & s_{pp} \end{bmatrix}. \tag{4.7}$$

$$s_{ii} = \frac{\sum_{l=1}^{n}(x_{il} - \overline{x}_i)^2}{n-1} \quad e \quad s_{ij} = s_{ji} = \frac{\sum_{l=1}^{n}(x_{il} - \overline{x}_i)(x_{jl} - \overline{x}_j)}{n-1} \quad para \quad i \neq j. \tag{4.8}$$

Dessa forma, sejam $\widehat{\lambda}_1, \widehat{\lambda}_2, ..., \widehat{\lambda}_p$ os autovalores da matriz $S_{p\times p}$ e $\widehat{v}_1, \widehat{v}_2, ..., \widehat{v}_p$ seus correspondentes autovetores normalizados, as componentes principais amostrais estimadas $(\widehat{pc_1}, \widehat{pc_2}, ..., \widehat{pc_p})$ são descritas por (4.9), enquanto suas respectivas variâncias estimadas associadas são descritas por (4.10).

$$\widehat{pc}_j = \widehat{v}_j^T x = \widehat{v}_{j1}x_1 + \widehat{v}_{j2}x_2 + ... + \widehat{v}_{jp}x_p \qquad j = 1, 2, ..., p. \tag{4.9}$$

$$\text{var}[\widehat{pc}_j] = \widehat{\lambda}_j \qquad j = 1, 2, ..., p. \tag{4.10}$$

Entre as informações geradas pela PCA se encontram os valores numéricos das componentes, denomindados escores, que viabilizam uma análise de desempenho global dos elementos, as correlações entre variáveis e componentes, que viabilizam uma interpretação individualizada das variáveis e a qualidade da aproximação do sistema original pelas k componentes. Apesar disso, segundo Sueli Mingoti [24], para que a PCA tenha algum sentido, uma vez que busca descorrelacionar dados para facilitar a interpretação dos mesmos, é necessário que exista correlação entre eles. Caso contrário, sua aplicação irá retornar as próprias variáveis originais. Portanto, é aconselhável verificar a existência de correlação a partir da análise de distribuição das probabilidades das variáveis e/ou testes de hipóteses [24]. A seguir são apresentadas algumas propriedades da PCA.

4.1.1 Propriedades

Propriedade 4.1 (Propriedade 2, [24]) *A covariância entre duas componentes quaisquer $\widehat{pc}_j$ e $\widehat{pc}_k$ é zero para todo $j \neq k$ ($\widehat{pc}_j$ e $\widehat{pc}_k$ são não correlacionadas).*

Propriedade 4.2 (Propriedade 3, [24]) *A variância total explicada pela j-ésima componente principal amostral é dada por:*

$$\frac{var[\widehat{pc}_j]}{Variância\ total\ estimada\ de\ X} = \frac{\widehat{\lambda}_j}{traço\ (S_{p\times p})} = \frac{\widehat{\lambda}_j}{\sum_{i=1}^{p} \widehat{\lambda}_i}. \tag{4.11}$$

Consequentemente, a variância total explicada pelas k primeiras componentes principais é definida por:

$$\frac{\sum_{j=1}^{k} var[\widehat{pc}_j]}{Variância\ total\ estimada\ de\ X} = \frac{\sum_{j=1}^{k} \widehat{\lambda}_j}{traço\ (S_{p\times p})} = \frac{\sum_{j=1}^{k} \widehat{\lambda}_j}{\sum_{i=1}^{p} \widehat{\lambda}_i}. \tag{4.12}$$

Propriedade 4.3 (Propriedade 4, [24]) *A correlação estimada entre a j-ésima componente principal amostral (pc_j) e a variável aleatória x_i ($i = 1, 2, .., p$) é dada por (4.13) onde s_{ii} é a variância amostral da variável aleatória x_i.*

$$r_{\widehat{pc}_j, x_i} = \frac{\widehat{v}_{ji}\sqrt{\widehat{\lambda}_j}}{\sqrt{s_{ii}}}. \tag{4.13}$$

Propriedade 4.4 (Propriedade 5, [24]) *Pelo teorema da decomposição espectral, a matriz de covariâncias $S_{p\times p}$ pode ser expressa como (4.14) ou aproximada por (4.15), caso sejam usadas apenas as k primeiras componentes principais amostrais.*

$$S_{p\times p} = \sum_{j=i}^{p} \widehat{\lambda}_j \widehat{v}_j \widehat{v}_j^T. \tag{4.14}$$

$$S_{p\times p} \approx \sum_{j=1}^{k} \widehat{\lambda}_j \widehat{v}_j \widehat{v}_j^T. \tag{4.15}$$

4.1.2 Interpretação geométrica

Geometricamente, a PCA pode ser entendida pela sequência - dados originais (Figura 4.1), dados ajustados pela média (Figura 4.2) e dados transformados (componentes principais, Figura 4.3). Ou seja, ajusta-se os dados e identifica-se as direções de maiores variabilidades via (4.3) e, por fim, transforma-se os dados ajustados por meio de (4.4).

X	Y
2.5	2.4
0.5	0.7
2.2	2.9
1.9	2.2
3.1	3.3
2.3	2.7
2.0	1.6
1.0	1.1
1.5	1.6
1.1	1.9

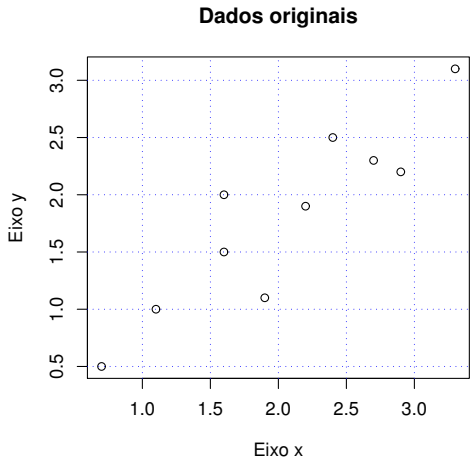


Figura 4.1: Dados originais.

X	Y
0.69	0.49
-1.31	-1.21
0.39	0.99
0.09	0.29
1.29	1.09
0.49	0.79
0.19	-0.31
-0.81	-0.81
-0.31	-0.31
-0.71	-1.01

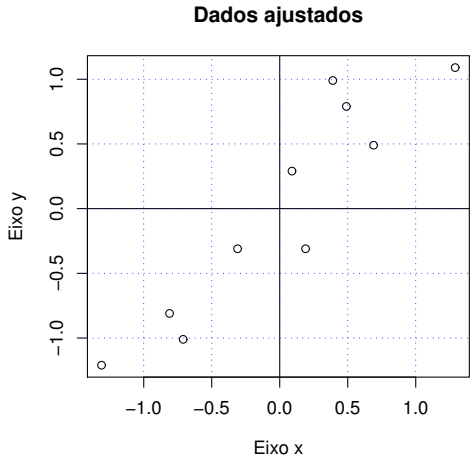


Figura 4.2: Dados ajustados pelas médias e autovetores da matriz $\Sigma_{p\times p}$ sobrepostos.

PC_1	PC_2
-0.8280	-0.1751
1.7776	0.1429
-0.9922	0.3844
-0.2742	0.1304
-1.6758	-0.2095
-0.9129	0.1753
0.0991	-0.3498
1.1446	0.0464
0.4380	0.0178
1.2238	-0.1627

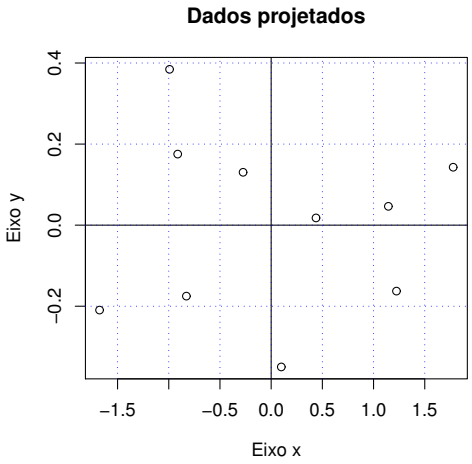


Figura 4.3: Dados transformados.

Pela Figura 4.3 é possível notar que a PCA consiste em rotacionar os dados por meio do conjunto de autovetores ortogonais de $\Sigma_{p\times p}$, de forma a gerar um novo sistema de coordenadas definido pelas componentes principais. Portanto, como apresentado pela Figura (4.4), cada ponto no sistema de coordenadas (x,y) é projetado ortogonalmente no novo sistema (pc_1, pc_2). Vale ressaltar que isso é possível pois, como os autovetores são ortogonais entre si, a transformação é ortogonal. Além disso, a grande vantagem dessa transformação, além da possibilidade de se mensurar a quantidade de informação perdida ao reduzir o número de variáveis (componentes) do sistema, é que as relações de distância entre os dados projetados no novo sistema de coordenadas são preservadas.

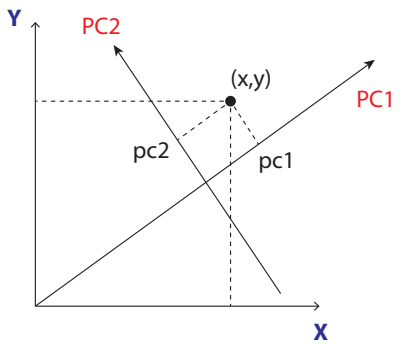


Figura 4.4: Interpretação geométrica da PCA

4.1.3 Exemplo de aplicação

Exemplo 4.1 [PCA via matriz de covariância amostral (Exemplo 3.1, [24])]

Considerando os dados referentes a 3 variáveis de 12 empresas apresentados pela Tabela 4.1, para aplicação da PCA é necessário, inicialmente, construir a matriz de covariâncias amostrais definida em (4.7) e realizar sua decomposição espectral. Para tanto, segue um código em SCILAB contendo os cálculos iniciais.

Tabela 4.1: Dados relativos as 12 empresas

Empresa	Ganho bruto(R$)	Ganho líquido(R$)	Patrimônio(R$)
E1	9893	564	17689
E2	8776	389	17359
E3	13572	1103	18597
E4	6455	743	8745
E5	5129	203	14397
E6	5432	215	3467
E7	3807	385	4679
E8	3423	187	6754
E9	3708	127	2275
E10	3294	297	6754
E11	5433	432	5589
E12	6287	451	8972

```
-->X = [9893  564  17689;
        8776  389  17359;
        13572 1103 18597;
        6455  743  8745;
        5129  203  14397;
        5432  215  3467;
        3807  385  4679;
        3423  187  6754;
        3708  127  2275;
        3294  297  6754;
        5433  432  5589;
        6287  451  8972];

-->[n,p] = size(X);          // n = empresas, p = variáveis
-->media = mean(X,'r');      // Vetor de médias amostrais
-->Xm = X - ones(n,1)*media; // Subtração da média amostral de cada coluna de X
-->S = Xm' * Xm / (n-1);     // Cálculo da matriz de covariâncias amostrais
```

```
--> round(S*1e0)*1e-0         // Arredondamento de S para 0 casas decimais
    ans  =
    9550609.     706121.    14978233.
    706121.      762670.    933915.
    14978233.    933915.    34408113.

--> [V,D] = spec(S);          // Decomposição espectral de S
--> round(D*1e0)*1e-0         // Arredondamento dos autovalores para 0 casas decimais
    ans  =
    21093.       0.          0.
    0.           2539507.    0.
    0.           0.          41474391.

--> round(V*1e4)*1e-4         // Arredondamento dos autovetores para 4 casas decimais
    ans  =
    0.0991    0.8997    0.4251
  - 0.9949    0.0965    0.0277
  - 0.0161  - 0.4257    0.9047
```

■

Logo, a partir dos autovetores normalizados é possível construir as componentes principais via (4.9). Ou seja, como os autovalores em ordem de importância são $\widehat{\lambda}_1 = 41474391$, $\widehat{\lambda}_2 = 2539507$ e $\widehat{\lambda}_3 = 21093$ e seus correspondentes autovetores normalizados são descritos por $\widehat{v}_1$, $\widehat{v}_2$ e $\widehat{v}_3$, as componentes principais são descritas por $\widehat{pc}_1$,$\widehat{pc}_2$ e $\widehat{pc}_3$ como apresentado a seguir.

$$\widehat{v}_1 = \begin{bmatrix} 0,4251 \\ 0,0277 \\ 0,9047 \end{bmatrix} \qquad \widehat{v}_2 = \begin{bmatrix} 0,8997 \\ 0,0965 \\ -0,4257 \end{bmatrix} \quad e \quad \widehat{v}_3 = \begin{bmatrix} 0,0991 \\ -0,9949 \\ -0,0161 \end{bmatrix},$$

$\widehat{pc}_1 = 0,4251$ (ganho bruto) $+ 0,0277$ (ganho líquido) $+ 0,9047$ (patrimônio),

$\widehat{pc}_2 = 0,8997$ (ganho bruto) $+ 0,0965$ (ganho líquido) $- 0,4257$ (patrimônio),

$\widehat{pc}_3 = 0,0991$ (ganho bruto) $- 0,9949$ (ganho líquido) $- 0,0161$ (patrimônio).

Além disso, por meio de (4.11) é possível definir a variância total explicada por cada componente (4.16, 4.17 e 4.18). Note que as informações mais relevantes dos dados amostrais originais estão contidas nas duas primeiras componentes, por descreverem

aproximadamente 100 % da variância total. Todavia, o fato da $\widehat{pc}_1$ representar 94, 18 % da variância total dos dados a torna, basicamente, um índice de desempenho global. Com isso, a partir de seus escores calculados individualmente, como por exemplo, em (4.19) para $E1$, é possível indicar o ranking das empresas baseado nos melhores desempenhos. Pela Tabela 4.2 percebe-se que $E3$, $E1$ e $E2$ estão no topo.

$$\frac{\widehat{\lambda}_1}{\sum_{j=1}^{3}\widehat{\lambda}_j} \times (100) = \frac{41474391}{44034991}(100) = 94,18\ \% \tag{4.16}$$

$$\frac{\widehat{\lambda}_2}{\sum_{j=1}^{3}\widehat{\lambda}_j} \times (100) = \frac{2539507}{44034991}(100) = 5,77\ \% \tag{4.17}$$

$$\frac{\widehat{\lambda}_3}{\sum_{j=1}^{3}\widehat{\lambda}_j} \times (100) = \frac{21093}{44034991}(100) = 0,05\ \% \tag{4.18}$$

$$\widehat{pc}_1 = 0,4251(9893) + 0,0277(564) + 0,9047(17689) \approx 20224,4. \tag{4.19}$$

Tabela 4.2: Escores referentes a $\widehat{pc}_1$

Empresa	Escores
E1	20224,4
E2	19446,1
E3	22624,7
E4	10676,2
E5	15210,9
E6	5451,7
E7	5862,1
E8	7570,6
E9	3638,0
E10	7518,9
E11	7377,9
E12	10802,1

Em se tratando do significado da correlação existente entre variáveis e componentes, é por meio de (4.13) que torna-se possível interpretar o impacto global gerado por cada variável individualmente. Por exemplo, como mostra a Tabela 4.3, a primeira com-

ponente ($\widehat{pc}_1$) está correlacionada com as três variáveis, em especial, com a variável Patrimônio, que como pode ser observado pela Tabela 4.4, possui o maior valor de variância. Por outro lado, existe uma maior correlação entre a terceira componente ($\widehat{pc}_3$) e a váriavel Ganho líquido que, por sua vez, possui a menor variância. Além disso, essa componente apresenta quase nenhuma correlação com as outras duas variáveis. Dessa forma, considerando as informações anteriores e observando os valores dos coeficientes, pode-se concluir que a segunda componente ($\widehat{pc}_2$) possui maior correlação com a variável Ganho bruto, no qual apresenta a segunda maior variância. Em síntese, como a variável Patrimônio domina a $\widehat{pc}_1$ por apresentar maior variância, pode vir a ser considerada a variável mais importante ou mais discriminativa, enquanto a variável Ganho Líquido por dominar a $\widehat{pc}_3$, pode vir a ser considerada a menos importante ou menos discriminativa, simplesmente por apresentar menor variância. Essa característica, na qual cada componente é dominada por uma variável em particular devido a uma grande discrepância entre as variâncias, gera algumas críticas em relação a interpretação da PCA via matriz de covariância.

Tabela 4.3: Correlação entre as componentes e as variáveis originais via (4.13).

Variável/Componente	$\widehat{pc}_1$	$\widehat{pc}_2$	$\widehat{pc}_3$
Ganho bruto	0,8859	0,4639	0,0047
Ganho líquido	0,6450	0,5569	-0,5232
Patrimônio	0,9933	-0.1156	-0,0004

Tabela 4.4: Estatísticas descritivas.

Variável	Média	Desvio Padrão	Variância
Ganho bruto	6267,4	3090,4	9550608,6
Ganho líquido	424,7	276,2	76269,5
Patrimônio	9606,4	5865,8	34408113,0

4.2 PCA via matriz de correlação

Apesar de a medida de covariância entre duas variáveis fornecer o grau de relacionamento linear entre elas, não existe um valor mínimo ou máximo determinado para comparação, ou seja, a covariância não é considerada uma medida padronizada. Consequentemente, como pode ser observado no Exemplo 4.1, existe uma forte relação entre o coeficiente de maior valor absoluto na primeira componente principal e a variável original de maior variância, entre o coeficiente de maior valor absoluto na segunda componente principal e a segunda variável de maior variância e assim, sucessivamente.

Com isso, as componentes principais são facilmente influenciadas pelas variáveis que possuem maior variância. Dessa forma, essa medida não se apresenta muito útil em casos onde existe uma grande discrepância entre as variâncias. Isso acontece, principalmente, quando os valores das variáveis envolvidas não se encontram numa mesma escala de medida. Logo, a medida de covariância normalizada pelos desvios padrões (σ) das variáveis (4.20), também designada por medida de correlação, torna-se uma boa alternativa para se comparar variáveis, por apresentar valores padronizados entre -1 e 1. Nesse contexto, um valor de correlação mais próximo de 1 indica que existe um relacionamento linear crescente (positivo) entre as variáveis, um valor mais próximo de -1 indica que existe um relacionamento linear decrescente (negativo) e, por fim, um valor mais próximo de 0 indica que não existe um relacionamento linear entre elas. Além disso, a matriz $P_{p\times p}$ que sumariza a correlação entre as p variáveis de um vetor aleatório x é definida por (4.21). Note que quando $i = j$ o coeficiente de correlação (4.20) possui valor igual a 1.

$$\rho_{ij} = \frac{\sigma_{ij}}{\sqrt{\sigma_{ii}\sigma_{ij}}} = \frac{\sigma_{ij}}{\sigma_i\sigma_j} \tag{4.20}$$

$$P_{p\times p} = \begin{bmatrix} 1 & \rho_{12} & \cdots & \rho_{1p} \\ \rho_{21} & 1 & \cdots & \rho_{2p} \\ \vdots & \vdots & \vdots & \vdots \\ \rho_{p1} & \rho_{p2} & \cdots & 1 \end{bmatrix} \tag{4.21}$$

Pode ser verificado [24] que a matriz $P_{p\times p}$ é a matriz de covariâncias das variáveis (4.22) onde $E(x) = \mu_i$ e $var(x_i) = \sigma^2$ para $i = 1, 2, ..., p$.

$$z_i = \frac{(x_i - \mu_i)}{\sigma_i}. \tag{4.22}$$

Ao aplicar PCA sobre a matriz $P_{p\times p}$, tem-se que as componentes principais são constituídas pelas combinações lineares das variáveis x_i padronizadas (4.22). No entanto, assim como no caso da medida de covariância, a medida de correlação deve ser estimada a partir dos elementos amostrais coletados como visto em (4.7). Portanto, o coeficiente estimado é designado por correlação amostral ou por correlação de Pearson (4.23) [24]. Dessa forma, a matriz $R_{p\times p}$ que sumariza a correlação amostral é definida por (4.24).

$$R_{ij} = \frac{s_{ij}}{\sqrt{s_{ii}s_{jj}}} \tag{4.23}$$

$$R_{p\times p} = \begin{bmatrix} r_{11} & r_{12} & \cdots & r_{1p} \\ r_{21} & r_{22} & \cdots & r_{2p} \\ \vdots & \vdots & \vdots & \vdots \\ r_{p1} & r_{p2} & \cdots & r_{pp} \end{bmatrix} \tag{4.24}$$

Dessa forma, sejam $\widehat{\lambda}_1, \widehat{\lambda}_2, ..., \widehat{\lambda}_p$ os autovalores da matriz $R_{p\times p}$ e $\widehat{v}_1, \widehat{v}_2, ..., \widehat{v}_p$ os correspondentes autovetores normalizados, as componentes principais amostrais estimadas $(\widehat{pc_1}, \widehat{pc_2}, ..., \widehat{pc_p})$ são descritas por (4.25), enquanto suas respectivas variâncias estimadas associadas são descritas por (4.26).

$$\widehat{pc}_j = \widehat{v}_j^T z = \widehat{v}_{j1}z_1 + \widehat{v}_{j2}z_2 + ... + \widehat{v}_{jp}z_p \qquad j = 1, 2, ..., p. \tag{4.25}$$

$$\text{var}[\widehat{pc}_j] = \widehat{\lambda}_j \qquad j = 1, 2, ..., p. \tag{4.26}$$

Analogamente à aplicação da PCA via matriz de covariâncias, entre as informações geradas pela PCA via matriz de correlação se encontram os valores numéricos das componentes, que viabilizam uma análise de desempenho global dos elementos, as correlações entre variáveis e componentes (Propriedade 4.5), que viabilizam uma interpretação individualizada das variáveis e a qualidade da aproximação do sistema original pelas k componentes (Propriedades 4.6 e 4.7). Contudo, apesar das componentes serem gera-

das a partir da matriz de correlação, a Propriedade 4.1 continua valendo nesse caso e, do mesmo modo, para que a PCA tenha algum sentido, é interessante, sempre quando possível, verificar a existência de correlação entre os dados, como por exemplo, a partir da análise de distribuição das probabilidades das variáveis e testes de hipóteses [24].

Propriedades

Propriedade 4.5 (Propriedade b), [24]) *A correlação estimada entre a j-ésima componente principal amostral e a variável padronizada* z_i $(i = 1, 2, .., p)$ *é dada por:*

$$r_{\widehat{pc}_j, z_i} = r_{\widehat{pc}_j, x_i} = \widehat{v}_{ji}\sqrt{\widehat{\lambda}_j}. \tag{4.27}$$

Consequentemente, as variáveis z_i*s com os maiores coeficientes na* $\widehat{pc}_j$ *são as mais correlacionadas com a componente.*

Propriedade 4.6 (Propriedade c), [24]) *A variância total do vetor aleatório* z $(z_1, z_2, ..., z_p)^T$ *é igual ao traço da matriz* $P_{p\times p}$*, que por sua vez é igual ao número p de variáveis medidas em cada elemento amostral. Portanto, a proporção da variância total explicada pela j-ésima componente principal é igual a:*

$$\frac{var[\widehat{pc}_j]}{Variância\ total\ estimada\ de\ z} = \frac{\widehat{\lambda}_j}{p} \quad para \quad j = 1, 2, ..., p. \tag{4.28}$$

Consequentemente, a variância total explicada pelas k primeiras componentes principais é definida por:

$$\frac{\sum_{i=1}^{k} var[\widehat{pc}_j]}{Variância\ total\ estimada\ de\ z} = \frac{\sum_{i=1}^{k} \widehat{\lambda}_i}{p}. \tag{4.29}$$

Propriedade 4.7 *Analogamente à Propriedade 4.4, a matriz de covariâncias* $R_{p\times p}$ *pode ser expressa como (4.30) ou aproximada por (4.31), caso sejam usadas apenas as k primeiras componentes principais amostrais.*

$$R_{p\times p} = \sum_{j=i}^{p} \widehat{\lambda}_j \widehat{v}_j \widehat{v}_j^T \tag{4.30}$$

$$R_{p\times p} \approx \sum_{j=1}^{k} \widehat{\lambda}_j \widehat{v}_j \widehat{v}_j^T . \tag{4.31}$$

Exemplo 4.2 [Exemplo (4.1) via matriz de correlação amostral (Exemplo 3.5, [24])]

Afim de comparar as medidas covariância e correlação, será utilizado o mesmo conjunto de dados do Exemplo 4.1. Portanto, segue um código em SCILAB contendo a construção da matriz de correlação amostral e a decomposição espectral da mesma.

```
X = [ 9893  564  17689;
      8776  389  17359;
      13572 1103 18597;
      6455  743  8745;
      5129  203  14397;
      5432  215  3467;
      3807  385  4679;
      3423  187  6754;
      3708  127  2275;
      3294  297  6754;
      5433  432  5589;
      6287  451  8972];

--> [n,p] = size(X);              // n = empresas, p = variáveis
--> media = mean(X,'r');          // Vetor de médias amostrais
--> Xm = X - ones(n,1)*media;     // Subtrai media de cada coluna de X
--> S = Xm' * Xm / (n-1);         // Matriz de covariâncias amostrais
--> diagS = diag(S);              // Diagonal de S
--> // Matriz de correlação amostral
--> R = S ./ sqrt(diagS*diagS')
    R  =
    1.           0.8273479    0.8262560
    0.8273479    1.           0.5765028
    0.8262560    0.5765028    1.
--> // Decomposição espectral de R
--> [V,D] = spec(R);
```

```
--> round(D*1e4)*1e-4          // Arredondando os autovalores para 4 casas decimais
    ans  =
    0.084    0.        0.
    0.       0.4235    0.
    0.       0.        2.4925

--> round(V*1e4)*1e-4          // Arredondando os autovetores para 4 casas decimais
    ans  =
  - 0.7872  - 0.0013    0.6167
    0.4373  - 0.7062    0.5568
    0.4348    0.708     0.5565
```

■

Portanto, como os autovalores em ordem de importância são $\widehat{\lambda}_1 = 2,4925$, $\widehat{\lambda}_2 = 0,4235$ e $\widehat{\lambda}_3 = 0,0840$ e seus correspondentes autovetores normalizados são descritos por $\widehat{v}_1$, $\widehat{v}_2$ e $\widehat{v}_3$, as componentes principais ($\widehat{pc}_1$,$\widehat{pc}_2$ e $\widehat{pc}_3$) das variáveis padronizadas (4.32) são obtidas por meio de (4.25).

$$\widehat{v}_1 = \begin{bmatrix} 0,6167 \\ 0,5568 \\ 0,5565 \end{bmatrix} \qquad \widehat{v}_2 = \begin{bmatrix} -0,0013 \\ -0,7062 \\ 0,7080 \end{bmatrix} \quad e \quad \widehat{v}_3 = \begin{bmatrix} -0,7872 \\ 0,4373 \\ 0,4348 \end{bmatrix},$$

$\widehat{pc}_1 = 0,6167\ (z_1) + 0,5568\ (z_2) + 0,5565\ (z_3)$,

$\widehat{pc}_2 = -\ 0,0013\ (z_1) - 0,7062\ (z_2) + 0,7080\ (z_3)$,

$\widehat{pc}_3 = -\ 0,7872\ (z_1) + 0,4373\ (z_2) + 0,4348\ (z_3)$,.

$$z_1 = \frac{x_1 - 6267,4}{3090,4}, \quad z_2 = \frac{x_2 - 424,7}{276,2} \quad e \quad z_3 = \frac{x_3 - 9606,4}{5865,8}. \tag{4.32}$$

A variância total explicada por cada componente principal é dada por (4.33), (4.34) e (4.35). Note que existe uma melhor distribuição da variância para as outras duas componentes e uma menor concentração na primeira componente. Consequentemente, ocorre uma maior uniformização das correlações e, com isso, não necessariamente a variável de maior variância domina a primeira componente, a segunda variável de maior variância domina a segunda componente e assim por diante (Tabela 4.5).

$$\frac{\widehat{\lambda}_1}{p} \times (100) = \frac{2,4925}{3}(100) = 83,08\ \% \tag{4.33}$$

$$\frac{\widehat{\lambda}_2}{p} \times (100) = \frac{0,4235}{3}(100) = 14,12\ \% \tag{4.34}$$

$$\frac{\widehat{\lambda}_3}{p} \times (100) = \frac{0,0840}{3}(100) = 2,80\ \% \tag{4.35}$$

Tabela 4.5: Correlação entre as componentes e as variáveis via (4.27).

Variável/Componente	$\widehat{pc}_1$	$\widehat{pc}_2$	$\widehat{pc}_3$
Ganho bruto	0,9736	-0,0008	-0,2281
Ganho líquido	0,8791	-0,4596	0,1267
Patrimônio	0,8785	0,4608	0,1260

Outro ponto a ser confrontado é a diferença da ordem de importância dos desempenhos globais das empresas, resultante dos valores dos escores das componentes. A Tabela 4.7 mostra a diferença na ordem (*ranking*) dos desempenhos globais baseados na $\widehat{pc}_1$ ao se utilizar as medidas covariância (Tabela 4.2) e correlação (Tabela 4.6). Essa diferença está relacionada com o fato de que as componentes foram obtidas pela decomposição espectral de matrizes diferentes . No entanto, pode-se verificar que apesar disso, existe consistência na classificação das melhores ($E3$, $E1$ e $E2$) e da pior empresa ($E9$). Além do mais, para facilitar a visualização das diferenças ocorridas na ordenação em relação aos dois tipos de aplicação da PCA, as empresas que apresentam posicionamentos distintos se encontram destacadas em negrito na Tabela 4.7.

Em síntese, as principais diferenças ocorridas entre aplicar a PCA via medida de covariância ou via medida de correlação foram: diferença da distribuição da explicação da variância total pelas componentes, diferença da ordem de importância ou desempenho global das empresas e não correlação entre a variável original de maior variância e a variável que domina (possui o maior coeficiente numérico) as componentes. Isso, aparentemente, pode provocar grande impacto, como por exemplo, na resolução do problema de seleção de características (*features*) por meio da interpretação das medidas de correlação entre variáveis e componentes.

Tabela 4.6: Escores referentes a $\widehat{pc}_1$

Empresa	Escores
E1	1.7712
E2	1.1642
E3	3.6781
E4	0.5974
E5	-0.2195
E6	-1.1718
E7	-1.0384
E8	-1.3173
E9	-1.8064
E10	-1.1213
E11	-0.5329
E12	-0.0032

Tabela 4.7: Ranking das empresas baseado nos escores da $\widehat{pc}_1$

Posição	Empresa (covariância)	Empresa (correlação)
1	E3	E3
2	E1	E1
3	E2	E2
4	E5	**E4**
5	E12	E12
6	E4	**E5**
7	E8	**E11**
8	E10	**E7**
9	E11	**E10**
10	E7	**E6**
11	E6	**E8**
12	E9	E9

4.3 Determinação do número de componentes

Atualmente, não existe um critério de determinação ideal do número de componentes principais que seja independente do conjunto de dados. No entanto, apesar dessa questão se encontrar em aberto na literatura, existem alguns critérios que têm sido muito utilizados. Sueli Mingoti [24] descreve três procedimentos usuais que considera serem os mais apropriados para essa tarefa. O primeiro deles é baseado na representatividade

da variância total e depende da medida utilizada, covariância ou correlação. No caso da PCA via matriz de covariância, o número k de componentes é determinado via (4.36) onde P representa a porcentagem da variância total que deseja-se reter.

$$\frac{\sum_{j=1}^{k} \widehat{\lambda}_j}{\sum_{i=1}^{p} \widehat{\lambda}_i} = P. \tag{4.36}$$

Por outro lado, no caso da PCA via matriz de correlação, onde a variância total é igual ao número de variáveis originais [Propriedade 4.6], o valor de k pode ser determinado pelo critério de Kaiser [20], o qual consiste em considerar apenas as componentes que conseguem explicar pelo menos a variância referente a uma variável padronizada, ou seja, as componentes associadas aos autovalores $\lambda_i \geq 1$. Da mesma forma, esse critério pode ser aplicado à matriz de covariância. No entanto, a idéia consiste em considerar apenas as componentes associadas aos autovalores que são maiores ou iguais à variância média $(\widehat{\lambda}_m)$ das variáveis originais X_i $(i = 1, 2, ..., p)$ que é descrita por:

$$\widehat{\lambda}_m = \frac{\sum_{j=1}^{p} \widehat{\lambda}_j}{p}. \tag{4.37}$$

Em se tratando das duas medidas covariância e correlação, geralmente para uma mesma base de dados, o valor de k tende a ser maior perante a utilização da medida de correlação devido à padronização e, consequentemente, à melhor distribuição da variância.

O segundo procedimento a ser descrito baseia-se na teoria de aproximação de matrizes explicada pelo Teorema 3.5. Nesse caso, as aproximações envolvem as matrizes de covariância ($S_{p\times p}$, 4.15) e de correlação ($R_{p\times p}$, 4.31). Além do mais, a idéia principal da aproximação é maximizar o corte de componentes e ao mesmo tempo minimizar a perda de informação. Entre as principais formas de se mensurar a qualidade desse critério se encontram o uso das normas matricias 2 (Teorema 3.7) e de Frobenius (Teorema 3.6) e da técnica conhecida por scree-plot [5]. Essa técnica consiste, basicamente, em construir o gráfico de ordenação dos autovalores que explicam a variância total dos dados e verificar a existência de algum ponto no qual esses valores se estabilizam. Ou seja, dado que as matrizes de covariância e correlação são simétricas, os autovalores são reais, positivos e apresentam uma queda semelhante à apresenta pela Figura 4.5.

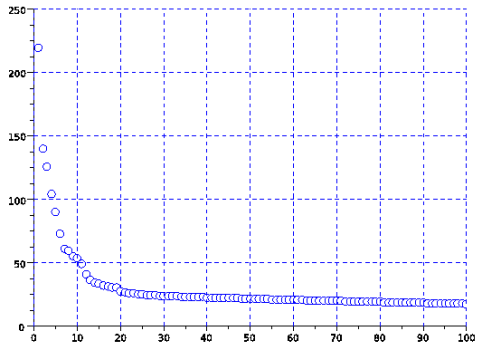


Figura 4.5: Exemplo de distribuição de autovalores.

Alguns autores consideram a escolha do k pela análise de um gráfico scree-plot um processo experimental devido à diversidade dos tipos de queda e da dificuldade de identificar o melhor ponto. No entanto, até então, não existe processo diferente do experimental que auxilie na escolha entre os pontos próximos à zona de estabilização. Entretanto, na maioria dos casos, o ganho resultante da maximização do corte ou da minimização da perda de informação não se apresenta significante entre esses pontos.

Finalmente, o terceiro procedimento está embasado na análise prática no número de componentes. Esse critério é o mais simples de todos e consiste em escolher, experimentalmente, o número de componentes que apresente maior utilidade para o usuário. Uma vez que, quanto maior o número de componentes utilizadas maior a dificuldade de interpretação das mesmas, o melhor caso nesse contexto, apesar de ser muito raro, é o que apresenta um menor número de componentes de maior interesse e ao mesmo tempo explique a maior parte da variação.

4.4 Cálculo da PCA

O cálculo da PCA resume-se na decomposição espectral das matrizes obtidas por meio das medidas de covariância ou correlação dos dados. Existem duas principais formas de se calcular a PCA. A primeira delas é pelo método direto e tradicional, onde manipula-se toda a matriz, extraindo-se inicialmente todos os autovalores/autovetores. O outro método é o iterativo, no qual é possível calcular, inclusive, apenas uma quantidade dos maiores ou menores autovalores/autovetores de interesse.

4.4.1 Método direto

Tradicionalmente, a PCA é obtida a partir das seguintes etapas:

- Cálculo da matriz de covariância ou correlação,
- Decomposição espectral da matriz escolhida no passo anterior,
- Ordenação decrescente dos autovalores,
- Transformação dos dados ajustados por meio dos autovetores correspondentes aos k maiores autovalores.

Essas etapas podem ser realizadas diretamente com o comando *pca* do SCILAB ou separadamente por meio da SVD, como mostra o Exemplo 4.3.

Exemplo 4.3 [PCA por meio de métodos diretos via SCILAB]

```
--> // Dados originais
--> X = [2.5 2.4; 0.5 0.7; 2.2 2.9; 1.9 2.2; 3.1 3.3;
        2.3 2.7; 2.0 1.6; 1.0 1.1; 1.5 1.6; 1.1 1.9];
--> // Uso do comando pca
--> [autos, comps, projecao] = pca(X);
--> autos(:,1)', comps, projecao
    autos    =  1.90367      0.0963300

    comps    =  0.7071068  - 0.7071068
                0.7071068    0.7071068

    projecao =  0.9330320  - 0.3097014
              - 2.3397836    0.0196088
                1.095741     0.3933264
                0.2195657    0.0574701
                2.2525141  - 0.0708571
                1.0126468    0.1301259
              - 0.2098232  - 0.5520252
              - 1.5432232  - 0.0843622
              - 0.6600890  - 0.1017595
              - 0.7605805    0.5181742

--> // Uso da SVD para realizar a PCA
--> [n,p] = size(X);
--> media = mean(X,'r'); Xm = X - ones(n,1)*media;
```

```
--> // Matriz de covariância
--> S = Xm'*Xm / (n-1); S
    S =  0.6165556    0.5795556
         0.5795556    0.6671111
--> desvio = st_deviation(X,'r'), Xmd = Xm ./ (ones(n,1)*desvio);
--> // Matriz de correlação
--> R = Xmd'*Xmd / (n-1)
    R =  1            0.9036700
         0.9036700    1
--> // Obtendo componentes para PCA (correlação)
--> [U,autos,comps] = svd(R);
--> diag(autos)', comps, projecao = Xmd*comps

    autos    =    1.90367      0.0963300

    comps    =  - 0.7071068  - 0.7071068
                - 0.7071068    0.7071068

    projecao =    0.9330320  - 0.3097014
                - 2.3397836    0.0196088
                  1.095741     0.3933264
                  0.2195657    0.0574701
                  2.2525141  - 0.0708571
                  1.0126468    0.1301259
                - 0.2098232  - 0.5520252
                - 1.5432232  - 0.0843622
                - 0.6600890  - 0.1017595
                - 0.7605805    0.5181742
```

■

4.4.2 Método iterativo

A grande vantagem de calcular a PCA via método iterativo (Lanczos) é a possibilidade de trabalhar com grandes matrizes e de extrair apenas k maiores autovalores e respectivos autovetores. O Exemplo 4.4 mostra a PCA realizada por meio do comando sva (svd iterativo) para o mesmo conjunto de dados utilizados no Exemplo 4.3 do cálculo direto e tradicional. É possível notar pelo código que a decomposição iterativa retorna a matriz V contendo apenas as k componentes principais de maior importância, baseado nos k maiores autovalores que representam o grau de variabilidade dos dados que cada componente representa respectivamente.

Exemplo 4.4 [PCA por meio de método iterativo via SCILAB]

```
--> k = 1; // Utilizar apenas a primeira componente
--> media = mean(X,'r'); Xm = X - ones(n,1)*media;
--> // Matriz de covariância
--> S = Xm'*Xm / (n-1);
--> [Uk,autosk,compsk] = sva(S,k); diag(autosk)', compsk, projecao = Xm*compsk
    autosk    =    1.2219399

    compsk    =  - 0.6915293
                 - 0.7223484

    projecao  =  - 0.7372007
                   1.8738503
                 - 0.8909160
                 - 0.1778134
                 - 1.8022318
                 - 0.8155993
                   0.1864427
                   1.2391462
                   0.5322074
                   0.5921146
--> desvio = st_deviation(X,'r'), Xmd = Xm ./ (ones(n,1)*desvio);
--> // Matriz de correlação
--> R = Xmd'*Xmd / (n-1);
--> [Uk,autosk,compsk] = sva(R,k); diag(autosk)', compsk, projecao = Xmd*compsk
    autosk    =    1.90367

    compsk    =  - 0.7071068
                 - 0.7071068

    projecao  =  - 0.9330320
                   2.3397836
                 - 1.095741
                 - 0.2195657
                 - 2.2525141
                 - 1.0126468
                   0.2098232
                   1.5432232
                   0.6600890
                   0.7605805
```

■

Capítulo 5

Algoritmos e estruturas de dados para matrizes esparsas

Em muitas aplicações, principalmente, naquelas que envolvem matrizes de alta ordem, a proporção de elementos não nulos tende a ser muito pequena em relação à ordem da matriz. Por definição, matrizes que apresentam essa propriedade são conhecidas por matrizes esparsas, ou seja, apresentam poucos elementos não nulos. Além do mais, dado que a complexidade de memória de um algoritmo está relacionada com a quantidade de armazenamento e, a complexidade de tempo com o número de operações realizadas, as estruturas de dados e os algoritmos envolvidos nesse contexto visam explorar e manter a esparsidade da matriz envolvida, armazenando somente os valores não nulos e operando somente sobre os mesmos. Além disso, Lars Eldén [11] menciona que devido à possibilidade dos métodos diretos destruirem a esparsidade da matriz em questão, os métodos iterativos passam a ser utilizados para encontrar a solução numérica de um problema, como por exemplo, o problema de autovalor. Aliás, manter e tirar proveito da esparsidade é um dos motivos pelo qual os métodos iterativos são de grande importância na resolução de problemas que envolvam esse tipo de matriz.

Dentre as principais estruturas de dados mais utilizadas para o armazenamento de matrizes esparsas se encontram o Esquema de coordenadas ou tripleto, a Coluna esparsa compactada (*Compressed Sparse Column* - CSC) e a Linha esparsa compactada (*Compressed Sparse Row* - CSR). Vale ressaltar ainda, que a estrutura CSC é comumente representada pelo formato esparso designado por Harwell-Boeing [8].

É fácil perceber a ineficiência de uma matriz esparsa mal utilizada, no qual os valores nulos são armazenados e manipulados. Por exemplo, se todos os elementos de uma matriz esparsa de ordem 50.000 forem armazenados em variáveis de 8 *bytes*, a quantidade total de memória utilizada será de aproximadamente 19 *gigabytes* (19GB), ou seja, para uma matriz desse porte, o custo computacional referente à memória será muito elevado. Além disso, da mesma forma, a complexidade de tempo do algoritmo envolvido será elevada devido ao número de operações aritméticas desnecessárias que podem vir a serem realizadas sobre esses elementos nulos armazenados.

Por questões de simplicidade, as matrizes esparsas simétrica e não simétrica das respectivas Figuras 5.1 e 5.2, serão utilizadas para exemplificar as três estruturas.

i/j	1	2	3	4	5	6	7	8	9	10
1	5.0		4.1		4.0				9.0	
2		2.4								
3	4.1		8.5				17.0			
4				1.9						
5	4.0				5.7					−2.0
6						10.3				
7			17.0				15.0			
8								5.8		
9	9.0								14.2	
10					−2.0					6.1

Figura 5.1: Exemplo de matriz simétrica.

i/j	1	2	3	4	5	6	7	8	9	10
1	5.0						−4.0			
2		2.4		2.1				1.5		
3			8.5		−3.0					3.5
4	1.1			1.9			6.0			
5			−2.0		5.7			4.0		
6	−3.0		6.0			10.3				
7		1.4				2.0	15.0		4.8	
8								5.8		7.7
9	3.6			2.2		3.3			14.2	
10					−3.2		3.8			6.1

Figura 5.2: Exemplo de matriz não simétrica.

5.1 Esquema de coordenadas ou tripleto

O esquema de coordenadas baseia-se em três vetores de comprimento (m) igual ao número de elementos não nulos armazenados. Por exemplo, os esquemas das Figuras 5.3 e 5.4 são compostos por três vetores de tamanho m, um que especifica a linha ($IndLin$), um que especifica a coluna ($IndCol$) e, finalmente, outro que especifica o valor do elemento (A), referentes às matrizes das Figuras 5.1 e 5.2 respectivamente. No entanto, é possível observar que o esquema simétrico armazena apenas os valores da diagonal e os valores abaixo ou acima da mesma. Dessa forma, ou esse esquema armazena a matriz triangular superior ou armazena a matriz triangular inferior, como foi, inclusive, o caso apresentado na Figura 5.3. Entretanto, apesar da simplicidade de representação do esquema de coordenadas, a maior dificuldade dessa estrutura se encontra na sua forma de acesso, uma vez que os vetores linha e coluna não precisam estar necessariamente ordenados.

i	IndLin	IndCol	A
1	1	1	5.0
2	2	2	2.4
3	3	1	4.1
4	3	3	8.5
5	4	4	1.9

i	IndLin	IndCol	A
6	5	1	4.0
7	5	5	5.7
8	6	6	10.3
9	7	3	17.0
10	7	7	15.0

i	IndLin	IndCol	A
11	8	8	5.8
12	9	1	9.0
13	9	9	14.2
14	10	5	−2.0
15	10	10	6.1

Figura 5.3: Esquema de coordenadas para matriz simétrica.

i	IndLin	IndCol	A
1	1	1	5.0
2	1	7	−4.0
3	2	2	2.4
4	2	4	2.1
5	2	8	1.5
6	3	3	8.5
7	3	5	−3.0
8	3	10	3.5
9	4	1	1.1
10	4	4	1.9

i	IndLin	IndCol	A
11	4	7	6.0
12	5	3	−2.0
13	5	5	5.7
14	5	8	4.0
15	6	1	−3.0
16	6	3	6.0
17	6	6	10.3
18	7	2	1.4
19	7	6	2.0
20	7	7	15.0

i	IndLin	IndCol	A
21	7	9	4.8
22	8	8	5.8
23	8	10	7.7
24	9	1	3.6
25	9	4	2.2
26	9	6	3.3
27	9	9	14.2
28	10	5	−3.2
29	10	7	3.8
30	10	10	6.1

Figura 5.4: Esquema de coordenadas para matriz não simétrica.

5.2 Coluna esparsa compactada (CSC)

A coluna esparsa compactada é uma estrutura muito mais eficiente que o tripleto pois evita a repetição desnecessária dos índices das colunas do vetor $IndCol$ que ocorre nos esquemas das Figuras 5.3 e 5.4. Para tanto, o vetor $IndCol$ é substituído pelo vetor $PCol$ composto por índices que representam posições do vetor A, nas quais aparecem o primeiro elemento não nulo de cada coluna, como ilustrado pelas Figuras 5.5 e 5.6. Logo, $PCol(2) = 5$ significa que a posição 5 do vetor A contém o primeiro valor não nulo (2.4) da coluna 2, além disso, a diferença entre as posições $PCol(i+1)$ e $PCol(i)$ possibilita determinar o número de elementos não nulos na i-ésima coluna da matriz.

i	Pcol	IndLin	A
1	1	1	5.0
2	5	3	4.1
3	6	5	4.0
4	8	9	9.0
5	9	2	2.4

i	Pcol	IndLin	A
6	11	3	8.5
7	12	7	17.0
8	13	4	1.9
9	14	5	5.7
10	15	10	−2.0

i	Pcol	IndLin	A
11	16	6	10.3
12		7	15.0
13		8	5.8
14		9	14.2
15		10	6.1

Figura 5.5: Esquema coluna esparsa compactada para matriz simétrica.

i	Pcol	IndLin	A
1	1	1	5.0
2	5	4	1.1
3	7	6	−3.0
4	10	9	3.6
5	13	2	2.4
6	16	7	1.4
7	19	3	8.5
8	23	5	−2.0
9	26	6	6.0
10	28	2	2.1

i	Pcol	IndLin	A
11	31	4	1.9
12		9	2.2
13		3	−3.0
14		5	5.7
15		10	−3.2
16		6	10.3
17		7	2.0
18		9	3.3
19		1	−4.0
20		4	6.0

i	IndLin	A
21	7	15.0
22	10	3.8
23	2	1.5
24	5	4.0
25	8	5.8
26	7	4.8
27	9	14.2
28	3	3.5
29	8	7.7
30	10	6.1

Figura 5.6: Esquema coluna esparsa compactada para matriz não simétrica.

Vale ressaltar que ao utilizar variáveis inteiras com i bytes e reais com r bytes, o número total de bytes gastos para armazenar esse tipo de matriz é $B = (m + n + 1)i + mr$.

5.3 Linha esparsa compactada (CSR)

A estrutura linha esparsa compactada é esquematicamente similar à estrutura coluna esparsa compactada apresentada na Seção 5.2. A única diferença é que agora o vetor $IndLin$ é substituído pelo vetor $PLin$, evitando a repetição desnecessária dos índices das linhas da matriz, como mostram as Figuras 6.2 e 5.8. Portanto, enquanto CSC compacta a informação referente às colunas, CSR compacta a informação referente às linhas. De modo similar, o vetor $PLin$ é composto por índices que representam as posições do vetor A, nas quais aparecem o primeiro valor não nulo de cada linha da matriz. Assim, $PLin(2) = 2$ significa que $A(2)$ contém o primeiro valor não nulo (2.4) da linha 2. Da mesma forma, a diferença entre as posições $PLin(i+1)$ e $PLin(i)$ possibilita determinar o número de elementos não nulos na i-ésima linha da matriz.

i	PLin	IndCol	A
1	1	1	5.0
2	2	2	2.4
3	3	1	4.1
4	5	3	8.5
5	6	4	1.9

i	PLin	IndCol	A
6	8	1	4.0
7	9	5	5.7
8	11	6	10.3
9	12	3	17.0
10	14	7	15.0

i	PLin	IndCol	A
11	16	8	5.8
12		1	9.0
13		9	14.2
14		5	−2.0
15		10	6.1

Figura 5.7: Esquema linha esparsa compactada para matriz simétrica.

i	PLin	IndCol	A
1	1	1	5.0
2	3	7	−4.0
3	6	2	2.4
4	9	4	2.1
5	12	8	1.5
6	15	3	8.5
7	18	5	−3.0
8	22	10	3.5
9	24	1	1.1
10	28	4	1.9

i	PLin	IndCol	A
11	31	7	6.0
12		3	−2.0
13		5	5.7
14		8	4.0
15		1	−3.0
16		3	6.0
17		6	10.3
18		2	1.4
19		6	2.0
20		7	15.0

i	IndCol	A
21	9	4.8
22	8	5.8
23	10	7.7
24	1	3.6
25	4	2.2
26	6	3.3
27	9	14.2
28	5	−3.2
29	7	3.8
30	10	6.1

Figura 5.8: Esquema linha esparsa compactada para matriz não simétrica.

Analogamente à capacidade de armazenamento de CSC, o número total de bytes gastos para armazenar uma matriz por meio do formato CSR é $B = (m + n + 1)i + mr$.

5.4 Algoritmos de multiplicação matriz-vetor

Assim como no caso das matrizes representadas por estruturas bidimensionais, várias são as operações que podem ser realizadas via matrizes esparsas. No entanto, a operação mais utilizada pelos métodos iterativos para a resolução do problema de autovalor que requer maior custo computacional é a multiplicação matriz-vetor. Com isso, considerando a eficiência das estruturas CSC e CSR, serão apresentados os algoritmos de multiplicação matriz-vetor referentes às mesmas, acompanhados de respectivos exemplos numéricos. Contudo, devido à diferença de armazenamento entre matrizes não simétricas e simétricas tanto para a estrutura CSC quanto para a CSR, os algoritmos referentes à operação de multiplicação matriz-vetor serão apresentados separadamente.

5.4.1 Matriz não simétrica

Algoritmo 4: Multiplicação matriz-vetor CSC

Entrada: n, IndLin, PCol, Val, y.
Saída: vetor x resultante do produto matriz-vetor y

```
1 início
2     para cada i ← 1 até n faça
3         x[i] → 0
4     para cada i ← 1 até n faça
5         para cada k ← PCol(i) até PCol(i + 1) − 1 faça
6             x(IndLin(k)) = x(IndLin(k)) + Val(k) ∗ y(i)
7 fim
```

Algoritmo 5: Multiplicação matriz-vetor CSR

Entrada: n, IndCol, PLin, Val, y.
Saída: vetor x resultante do produto matriz-vetor y

```
1 início
2     para cada i ← 1 até n faça
3         x[i] → 0
4         para cada k ← PLin(i) até PLin(i + 1) − 1 faça
5             x(i) = x(i) + Val(k) ∗ y(IndCol(k))
6 fim
```

Exemplo 5.1 [Multiplicação matriz-vetor x = Ay.]

```
--> A = [0.67 0 0 0.29; 0 0.71 0.41 0.29; 0.34 0 0.41 0.29; 0.67 0 0 0]
    A  =
    0.67    0.      0.      0.29
    0.      0.71    0.41    0.29
    0.34    0.      0.41    0.29
    0.67    0.      0.      0.
--> y = [1; 2; 3; 4];
--> x = (A*y)'
    x  =  1.83    3.81    2.73    0.67
--> // Vetores referentes ao formato CSC [Compressed sparse column]
--> val    = [0.67 0.34 0.67 0.71 0.41 0.41 0.29 0.29 0.29];
--> indlin = [1 3 4 2 2 3 1 2 3];
--> pcol   = [1 4 5 7 10];
--> n = length(pcol)-1;
--> // Multiplicação matriz-vetor
--> for i=1:n
-->     xcsc(i) = 0;
--> end
--> for i=1:n
-->     for k=pcol(i):pcol(i+1)-1
-->         xcsc(indlin(k)) = xcsc(indlin(k)) + val(k)*y(i);
-->     end
--> end
--> xcsc'
    xcsc  =  1.83    3.81    2.73    0.67
--> // Vetores referentes ao formato CSR [Compressed sparse row]
--> val    = [0.67 0.29 0.71 0.41 0.29 0.34 0.41 0.29 0.67];
--> indcol = [1 4 2 3 4 1 3 4 1];
--> plin   = [1 3 6 9 10];
--> n = length(plin)-1;
--> // Multiplicação matriz-vetor
--> for i=1:n
-->     xcsr(i) = 0;
-->     for k=plin(i):plin(i+1)-1
-->         xcsr(i) = xcsr(i) + val(k)*y(indcol(k));
-->     end
--> end
--> xcsr'
    xcsr  =  1.83    3.81    2.73    0.67
```

■

5.4.2 Matriz simétrica

A estratégia de armazenamento dos elementos de uma matriz simétrica por meio das estruturas CSC e CSR visa armazenar apenas uma das duas partes da matriz envolvida, parte triangular superior ou parte triangular inferior. Mais ainda, envolve armazenar necessariamente todos os valores da diagonal, inclusive, os valores nulos, juntamente com os valores não nulos contidos acima ou abaixo da diagonal respectivamente. Além disso, vale ressaltar que os valores nulos da diagonal são armazenados para viabilizar a indexação correta das linhas em CSC e das colunas em CSR. Dessa forma, consegue-se explorar a eficiência que a simetria da matriz propicia, pois é possível representá-la apenas por parte dos elementos e facilmente operar sobre os mesmos. O Exemplo 5.2 apresenta a representação e a operação de multiplicação matriz-vetor das estruturas CSC e CSR perante uma matriz simétrica que contém alguns valores nulos na diagonal.

Algoritmo 6: Multiplicação matriz-vetor CSC

Entrada: n, IndLin, PCol, Val, y.
Saída: vetor x resultante do produto matriz-vetor y

```
1 início
2     para cada j ← 1 até n faça
3         x[j] → 0
4     para cada j ← 1 até n faça
5         para cada k ← PCol(j) até PCol(j + 1) − 1 faça
6             x(IndLin[k]) = x(IndLin[k]) + Val[k] ∗ y(j)
7             se IndLin[k] > j então
8                 x[j] = x[j] + Val[k] ∗ y[IndLin[k]]
9 fim
```

Algoritmo 7: Multiplicação matriz-vetor CSR

Entrada: n, IndCol, PLin, Val, y.
Saída: vetor x resultante do produto matriz-vetor y

```
1 início
2     para cada j ← 1 até n faça
3         x[j] → 0
4         para cada k ← PLin(j) até PLin(j + 1) − 1 faça
5             x[j] = x[j] + Val[k] ∗ y[IndCol[k]]
6             se IndCol[k] < j então
7                 x[IndCol[k]] = x[IndCol(k)] + Val[k] ∗ y[j]
8 fim
```

Exemplo 5.2 [Multiplicação matriz-vetor x = Ay.]

```
--> A = [1 5 0 -4; 5 0 0 0; 0 0 -2 3; -4 0 3 0]
    A  =
    1.    5.    0.  - 4.
    5.    0.    0.    0.
    0.    0.  - 2.    3.
  - 4.    0.    3.    0.
--> y = [1; 0; 4; -1]; (A*y)'
    ans  =  5.    5.  - 11.    8.
--> // Vetores referentes ao formato CSC [Compressed sparse column]
--> val    = [1 5 -4 0 -2 3 0];
--> indlin = [1 2 4 2 3 4 4];
--> pcol   = [1 4 5 7 8]; n = length(pcol)-1;
--> for j=1:n
-->     xcsc(j)=0;
--> end
--> for j=1:n
-->     for k=pcol(j):pcol(j+1)-1
-->         xcsc(indlin(k)) = xcsc(indlin(k)) + val(k)*y(j);
-->         if indlin(k) > j
-->             xcsc(j) = xcsc(j) + val(k)*y(indlin(k));
-->         end
-->     end
--> end
--> xcsc'
    ans  =  5.    5.  - 11.    8.
// Vetores referentes ao formato CSR [Compressed sparse row]
--> val    = [1 5 0 -2 -4 3 0];
--> indcol = [1 1 2 3 1 3 4];
--> plin   = [1 2 4 5 8]; n = length(plin)-1;
--> for j=1:n
-->    xcsr(j)=0;
-->    for k=plin(j):plin(j+1)-1
-->        xcsr(j) = xcsr(j) + val(k)*y(indcol(k));
-->        if indcol(k) < j
-->            xcsr(indcol(k)) = xcsr(indcol(k)) + val(k)*y(j);
-->        end
-->    end
--> end
--> xcsr'
    ans  =  5.    5.  - 11.    8.
```

■

5.5 Algoritmos de multiplicação (matriz transposta)-vetor

A operação de multiplicação matriz-vetor também pode ser realizada considerando a transposta da matriz. Logo, pode ser realizada de forma que o resultado da operação seja $x = A^T y$. Esse processo é tão importante quanto o processo da multiplicação matriz-vetor representada por $x = Ay$ que foi apresentada na Seção 5.4. Juntas, essas duas operações contribuem, por exemplo, para o cálculo eficiente do produto $A^T A$ ou AA^T, uma vez que, ao serem aplicadas separadamente, preservam a esparsidade da matriz envolvida. Além do mais, se a matriz A for simétrica, tem-se que $A = A^T$ e, portanto, os algoritmos de multiplicação, por exemplo, referentes às estruturas CSC e CSR são os mesmos apresentados na Seção 5.4. No entanto, se não for simétrica, os algoritmos de multiplicação (matriz transposta)-vetor referentes às estruturas CSC e CSR são diferentes e se encontram descritos pelos Algoritmos 8 e 9 respectivamente. Além disso, esses algoritmos se encontram numericamente ilustrados pelo Exemplo 5.3.

Algoritmo 8: Multiplicação (matriz transposta)-vetor CSC

Entrada: n, IndLin, PCol, Val, y.
Saída: vetor x resultante do produto (matriz transposta)-vetor y

```
1 início
2     para cada i ← 1 até n faça
3         x[i] → 0
4         para cada k ← PCol(i) até PCol(i + 1) − 1 faça
5             x[i] = x[i] + Val[k] ∗ y[IndLin[k]]
6 fim
```

Algoritmo 9: Multiplicação (matriz transposta)-vetor CSR

Entrada: n, IndCol, PLin, Val, y.
Saída: vetor x resultante do produto (matriz transposta)-vetor y

```
1 início
2     para cada i ← 1 até n faça
3         x[i] → 0
4     para cada i ← 1 até n faça
5         para cada k ← PLin(i) até PLin(i + 1) − 1 faça
6             x[IndCol[k]] = x[IndCol[k]] + Val[k] ∗ y[i]
7 fim
```

Exemplo 5.3 [Multiplicação (matriz transposta)-vetor $x = A^T y$.]

```
--> A = [1 0 0 2; 0 2 0 3; 1 0 4 0; 5 0 0 1]
    A  =
    1.    0.    0.    2.
    0.    2.    0.    3.
    1.    0.    4.    0.
    5.    0.    0.    1.
--> y = [2;1;3;2];
--> (A'*y)'
    ans  =   15.  2.  12.  9.
--> // Vetores referentes ao formato CSC [Compressed sparse column]
--> val    = [1 1 5 2 4 2 3 1];
--> indlin = [1 3 4 2 3 1 2 4];
--> pcol   = [1 4 5 6 9];
--> n = length(pcol)-1;
--> // Multiplicação (matriz transposta)-vetor
--> for i=1:n
-->    xcsc(i) = 0
-->    for k=pcol(i):pcol(i+1)-1
-->        xcsc(i) = xcsc(i) + val(k)*y(indlin(k));
-->    end
--> end
--> xcsc'
    xcsc  =   15.  2.  12.  9.
--> // Vetores referentes ao formato CSR [Compressed sparse row]
--> val    = [1 2 2 3 1 4 5 1];
--> indcol = [1 4 2 4 1 3 1 4];
--> plin   = [1 3 5 7 9];
--> n = length(plin)-1;
--> // Multiplicação (matriz transposta)-vetor
--> for i=1:n
-->     xcsr(i)=0;
--> end
--> for i=1:n
-->     for k=plin(i):plin(i+1)-1
-->         xcsr(indcol(k)) = xcsr(indcol(k)) + val(k)*y(i);
-->     end
--> end
--> xcsr'
    xcsr  =  15.  2.  12.  9.
```

■

5.6 Algoritmo de conversão de estruturas esparsas

Devido a sua simplicidade, o esquema de coordenadas é a estrutura mais comumente apresentada ao usuário, como por exemplo, pelas plataformas SCILAB e MATLAB. Contudo, a biblioteca numérica ARPACK utilizada tanto pelo SCILAB quanto pelo MATLAB usa o formato designado por Harwell-Boeing (HB) que, por sua vez, consiste em representar a matriz pela estrutura CSC, juntamente com alguns padrões orientados para linguagem de programação FORTRAN. Logo, para apresentar o HB, formato utilizado nos experimentos desse trabalho, serão apresentados alguns detalhes do mesmo e o algoritmo de conversão da estrutura tripleto para a HB, cuja principal característica é um cabeçalho composto pelas quatro ou cinco linhas da Tabela 5.1.

Tabela 5.1: Cabeçalho do formato Harwell-Boeing (tamanho fixo de 80 colunas) [8].

Linha	Campo	Tipo	Significado	Caracteres
1	TITLE	Str.	Título.	72
	KEY	Str.	Identificador pelo qual a matriz é referenciada.	08
2	TOTCRD	Int.	N° total de linhas excluindo o cabeçalho.	14
	PTRCRD	Int.	N° de linhas para ponteiros.	14
	INDCRD	Int.	N° de linhas para índices de linhas.	14
	VALCRD	Int.	N° de linhas para valores numéricos da matriz.	14
	RHSCRD	Int.	N° de linhas para termos independentes.	14
3	MXTYPE	Str.	Tipo da matriz (ver Tabela 5.2).	03
			Espaço em branco.	11
	NROW	Int.	N° de linhas.	14
	NCOL	Int.	N° de colunas.	14
	NNZERO	Int.	N° de entradas não nulas de matrizes montadas.	14
	NELTVL	Int.	N° de entradas de matrizes elementares.	14
4	PTRFMT	Str.	Formato para ponteiros.	16
	INDFMT	Str.	Formato para índices de linhas.	16
	VALFMT	Str.	Formato para valores de entrada da matriz.	20
	RHSFMT	Str.	Formato para termos independentes.	20
5	RHSTYP	Str.e Int.	Informações referentes aos termos independentes.	03
Ocorre se		—	Espaço em branco.	11
RHSCRD > 0	NRHS	Int.	N° de termos independentes.	14
	NRHS	Int.	N° de índices de linhas.	14

O formato esparso Harwell-Boeing consiste basicamente na representação da estrutura CSC da matriz envolvida, composta pelo cabeçalho descrito pelas Tabelas 5.1 e 5.2.

Tabela 5.2: Variável MXTYPE (3 caracteres): indicam, em ordem, o tipo da matriz.

Ordem	Caractere	Significado
1	R, C ou P	Real, complexa ou somente o padrão de valores não nulos.
2	U, S, H, Z ou R	Não simétrica, simétrica, Hermitiana, quase simétrica ou retangular.
3	A E	Matriz montada (caso típico). Matriz elementar (soma de matrizes menores).

No entanto, na prática, o formato pode ser melhor entendido por meio do Exemplo 5.4 que ilustra a representação HB da matriz real, não simétrica e montada, dada por (5.1).

$$A = \begin{bmatrix} 11.0 & 0.0 & 0.0 & 14.0 & 0.0 \\ 0.0 & 22.0 & 0.0 & 0.0 & 0.0 \\ 31.0 & 32.0 & 33.0 & 34.0 & 35.0 \\ 0.0 & 0.0 & 0.0 & 44.0 & 45.0 \\ 51.0 & 52.0 & 0.0 & 0.0 & 55.0 \end{bmatrix} \tag{5.1}$$

Representação da CSC que acompanha o cabeçalho constituinte do formato HB:

PCol = [1 4 7 8 11 14],
IndLin = [1 3 5 2 3 5 3 1 3 4 3 4 5],
Val = [11.0 31.0 51.0 22.0 32.0 52.0 33.0 14.0 34.0 44.0 35.0 45.0 55.0].

Exemplo 5.4 [Matriz A (5.1) (real, não simétrica e montada) representada em HB.]

```
Title                                                                   Key
             5             1             1             3             0
RUA                        5             5            13             0
(6I3)           (13I3)          (5E15.8)
  1  4  7  8 11 14
  1  3  5  2  3  5  3  1  3  4  3  4  5
11.0           31.0           51.0           22.0           32.0
52.0           33.0           14.0           34.0           44.0
35.0           45.0           55.0
```

■

A principal diferença dos cabeçalhos do formato HB em relação à representação de matrizes simétricas e não simétricas aparece na variável MXTYPE pois a forma de armazenar os valores é dado propriamente pelo formato CSC, como apresentado na Seção 5.2. Na prática, outras diferenças podem ser observadas a partir do Exemplo 5.5, o qual ilustra a representação do formato HB da matriz simétrica B descrita em (5.2).

$$B = \begin{bmatrix} 1.0 & 0.0 & 2.0 & 0.0 & 3.0 \\ 0.0 & 0.0 & 4.0 & 0.0 & 5.0 \\ 2.0 & 4.0 & 3.0 & 25.0 & 0.0 \\ 0.0 & 0.0 & 25.0 & 0.0 & 0.0 \\ 3.0 & 5.0 & 0.0 & 0.0 & 17.0 \end{bmatrix} \tag{5.2}$$

Representação da CSC que acompanha o cabeçalho constituinte do formato HB:

PCol = [1 4 7 9 10 11],
IndLin = [1 3 5 2 3 5 3 4 4 5],
Val = [1.0 2.0 3.0 0.0 4.0 5.0 3.0 25.0 0.0 17.0].

Exemplo 5.5 [Matriz B (5.2) (real, simétrica e montada) representada em HB.]

```
Title                                                                   Key
             4             1             1             2             0
RSA                        5             5            10             0
(6I3)           (13I3)          (5E15.8)
  1  4  7  9 10 11
  1  3  5  2  3  5  3  4  4  5
1.0           2.0           3.0           0.0           4.0
5.0           3.0           25.0          0.0           17.0
```

■

Diferenças dos cabeçalhos HB das matrizes A (5.1) e B (5.2):

Linha 2: TOTCRD (5 e 4) e VALCRD (3 e 2) respectivamente.
Linha 3: MXTYPE (RUA e RSA) e NNZERO (13 e 10) respectivamente.

Maiores informações sobre o formato Harwell-Boeing podem ser encontrados em [8].

Conversão de tripleto para Harwell-Boeing

Dado que todas as bases de dados utilizadas nesse trabalho resumem-se em matrizes retangulares, montadas (*assembled*), compostas por elementos reais e que, em geral, os dados reais mais comuns são representados tanto por meio de matrizes quadradas não simétricas como simétricas, serão abordadas, especificamente, no processo de conversão, matrizes não simétricas, simétricas e retangulares. Isto é, casos em que a variável MXTYPE associada for RUA, RSA ou RRA. Além disso, dado que o formato HB utiliza-se da estrutura CSC, para simplificar, o esquema tripleto da matriz envolvida deve ser organizado necessariamente por ordem de colunas, como ilustra a Figura 5.9. Além do mais, as variáveis RHSCRD e NELTVL do cabeçalho HB serão sempre nulas pois os experimentos de interesse envolvem apenas matrizes montadas (*assembled*) e não carecem de termos independentes, como por ser observado por meio da Figura 5.10.

$$A = \begin{bmatrix} 11.0 & 0.0 & 0.0 & 14.0 & 0.0 & 16.0 \\ 0.0 & 22.0 & 0.0 & 0.0 & 25.0 & 26.0 \\ 0.0 & 0.0 & 33.0 & 34.0 & 0.0 & 36.0 \\ 41.0 & 0.0 & 43.0 & 44.0 & 0.0 & 46.0 \end{bmatrix}$$

IndLin	IndCol	Valor
1	1	11.0
4	1	41.0
2	2	22.0
3	3	33.0
4	3	43.0
1	4	14.0
3	4	34.0
4	4	44.0
2	5	25.0
1	6	16.0
2	6	26.0
3	6	36.0
4	6	46.0

Figura 5.9: Entrada: esquema tripleto da respectiva matriz ordenado por colunas.

```
tripleto.txt                                                          Key
             4             1             1             2             0
RUA                        4             6            13             0
(16I5)          (16I5)          (10F7.1)
    1    3    4    6    9   10   14
    1    4    2    3    4    1    3    4    2    1    2    3    4
   11.0   41.0   22.0   33.0   43.0   14.0   34.0   44.0   25.0   16.0
   26.0   36.0   46.0
```

Figura 5.10: Saída: respectivo esquema HB (variáveis nulas: RHSCRD e NELTVL).

Algoritmo 10: Tripleto para Harwell-Boeing.

Entrada: Vetores IndLin, IndCol, Valor e MXTYPE (RUA ou RRA).
Saída: Arquivo contendo o formato Harwell-Boeing da respectiva matriz.

```
1  início
2      nEntradas = tamanho(Valor); nLin = 0; l = 1; nCol = 0; c = 1
3      para cada i ← 1 aténEntradas faça
4          se IndLin[i] == l então
5              nLin = nLin + 1
6              l = l + 1
7          se IndCol[i] == c então
8              nCol = nCol + 1
9              c = c + 1
10     // Cria vetor de ponteiros de colunas (pCol)
11     pCol[1] = 1, j = 2
12     para cada k ← 1 aténEntradas faça
13         se IndCol[k] == j então
14             pCol[j] = k
15             j = j + 1
16     Pcol[nCol] = nEntradas + 1
17     //Define formatos, calcula e atribui variáveis (TOTCRD, INDCRD, entre outras)
18     //Escreve o cabeçalho HB seguido do formato CSC (pCol, IndLin, Valor).
19 fim
```

Exemplo 5.6 [Cria vetor de ponteiros pCol a partir do IndCol da Figura 5.9.]

```
--> IndLin = [1 4 2 3 4 1 3 4 2 1 2 3 4];
--> IndCol = [1 1 2 3 3 4 4 4 5 6 6 6 6];
--> Val    = [11.0 41.0 22.0 33.0 43.0 14.0 34.0 44.0 25.0 16.0 26.0 36.0 46.0];
--> nnZero = length(Val); nCol = 0; c = 1;
--> for i=1:nnZero
-->     if (IndCol(i) == c)
-->         nCol = nCol + 1; c = c + 1;
-->     end
--> end
--> pCol(1) = 1; j=2;
--> for k=1:nnZero
-->     if (IndCol(k) == j)
-->         pCol(j) = k; j = j + 1;
-->     end
--> end
--> pCol(nCol+1) = nnZero+1; pCol'
   pCol  =   1.   3.   4.   6.   9.   10.   14.
```

■

Caso a matriz seja simétrica, os valores nulos da diagonal devem ser armazenados para a correta indexação das linhas no esquema CSC. Portanto, se o tripleto apresentado se refere a uma matriz montada de elementos reais do tipo RSA, o tripleto fornecido deve ser pré-processado, ou seja, deve-se acrescentar zeros nas posições diagonais do mesmo que se encontram ausentes, como por ser observado por meio do Algoritmo 11 e via Exemplo 5.7. Assim, a conversão pode ser normalmente realizada via Algoritmo 10.

Algoritmo 11: Pré-processamento do tripleto no qual MXTYPE é igual a RSA.

Entrada: Vetores IndLin, IndCol, Valor e MXTYPE = RSA.
Saída: Vetores auxLin, auxCol e auxVal.

```
1  início
2      // Acrescentar zeros nas posições diagonais ausentes no tripleto.
3      se MXTYPE == RSA então
4          // Número de entradas armazenadas e índice de coluna.
5          nEntradas = tamanho(Valor); ind = 1
6          // Quantidade de zeros já inseridos.
7          qts = 0
8          // Vetores auxiliares.
9          auxLin = IndLin; auxCol = IndCol; auxVal = Valor
10         para cada i ← 1 até nEntradas faça
11             // Verifica se existe elemento na diagonal e atualiza índice.
12             se IndCol(i) == IndLin(i) então
13                 ind = ind + 1
14             senão
15                 // Acrescenta ind e 0 na posição k onde (pos − 1) < k < (pos).
16                 se IndCol(i) == ind então
17                     pos = i + qts
18                     auxLin[k] = ind; auxCol[k] = ind; auxVal[k] = 0
19                     qts = qts + 1
20                     ind = ind + 1
21                 senão
22                     // Acrescenta ind e 0 na posição k onde (pos) < k < (pos + 1).
23                     se IndLin(i) == ind então
24                         pos = i + qts
25                         auxLin[k] = ind; auxCol[k] = ind; auxVal[k] = 0
26                         qts = qts + 1
27                         ind = ind + 1
28 fim
```

Exemplo 5.7 [Pré-processamento do tripleto da matriz RSA descrita em (5.2)]

```
--> // Tripleto da matriz de tipo MXTYPE = RSA (real, simétrica e montada).
--> IndLin = [1 3 5 3 5 3 4 5];
--> IndCol = [1 1 1 2 2 3 3 5];
--> Val    = [1. 2. 3. 4. 5. 3. 25. 17.];
--> ind = 1; // Índice de coluna.
--> qts = 0; // Quantidade de zeros inseridos.
--> // Vetores auxiliares.
--> auxLin = IndLin; auxCol = IndCol; auxVal = Val;
--> // Acrescenta zeros nas posições diagonais ausentes.
--> for i=1:length(Val)
-->     if IndLin(i) == IndCol(i)
-->         ind = ind + 1;
-->     else
-->         if IndCol(i) == ind
-->             pos = i+qts;
-->             auxLin = [auxLin(1:pos-1) ind auxLin(pos:length(auxLin))];
-->             auxCol = [auxCol(1:pos-1) ind auxCol(pos:length(auxCol))];
-->             auxVal = [auxVal(1:pos-1) 0.0 auxVal(pos:length(auxVal))];
-->             ind = ind + 1;
-->             qts = qts + 1;
-->         else
-->             if IndLin(i) == ind
-->                 pos = i+qts;
-->                 auxLin = [auxLin(1:pos) ind auxLin(pos+1:length(auxLin))];
-->                 auxCol = [auxCol(1:pos) ind auxCol(pos+1:length(auxCol))];
-->                 auxVal = [auxVal(1:pos) 0.0 auxVal(pos+1:length(auxVal))];
-->                 ind = ind + 1;
-->                 qts = qts + 1;
-->             end
-->         end
-->    end
--> end
--> auxLin
    auxLin  =    1.   3.   5.   2.   3.   5.   3.   4.    4.   5.
--> auxCol
    auxCol  =    1.   1.   1.   2.   2.   2.   3.   3.    4.   5.
--> auxVal
    auxVal  =    1.   2.   3.   0.   4.   5.   3.   25.   0.   17.
```

■

Capítulo 6

Programas para SVD e PCA

Dentre os principais softwares numéricos utilizados para cômputo da SVD e da PCA se encontram SCILAB, MATLAB e R. No entanto, todos esses softwares são plataformas constituídas, em geral, por um ambiente e sua respectiva linguagem de programação. No entanto, apesar de utilizarem bibliotecas numéricas eficientes, como por exemplo, o SCILAB que utiliza o LAPACK para operar sobre matrizes densas e o ARPACK para operar sobre matrizes esparsas, apresentam limitações ao se depararem frente a grandes bases de dados, por manipularem linguagens interpretadas. Dessa forma, para usufruir da eficiência dessas bibliotecas é preciso utilizá-las diretamente de forma compilada, como será feito com o ARPACK, utilizado para operar com matrizes esparsas de alta ordem. Para verificar seu desempenho frente a função de executar ambas as técnicas, SVD e PCA, alguns resultados serão validados via SCILAB e via R respectivamente.

6.1 SCILAB

O SCILAB [1] é uma plataforma gratuita apropriada para computação de cálculos numéricos, desenvolvida e distribuída por uma equipe de pesquisa do *Institut National de Recherche en Informatique et en Automatique* (INRIA) [2]. Além disso, fornece uma grande variedade de técnicas matemáticas e, devido a sua confiabilidade, será utilizada para validar os resultados da SVD, implementada via código do ARPACK.

[1] `http://www.scilab.org/`
[2] `http://www.inria.fr/`

6.2 Projeto R

Assim como o SCILAB e o MATLAB são linguagens e ambientes apropriados para computação matemática, o Projeto R (*The R Project for Statistical Computing* [3]) é uma linguagem e ambiente apropriado para computação estatística. Dessa forma, fornece uma ampla variedade de técnicas estatísticas, como por exemplo, modelagem linear e não linear, análise de séries temporais, classificação e agrupamento de dados, entre outras. Frente a sua popularidade e desempenho, será o meio de validação dos resultados referentes ao cômputo da PCA via código compilado do ARPACK.

6.3 ARPACK

O ARPACK (*ARnoldi PACKage*) é uma biblioteca numérica composta por subrotinas implementadas na linguagem de programação FORTRAN 77, projetada com objetivo de resolver problemas de autovalor para matrizes de alta ordem. Portanto, seu uso é geralmente mais apropriado para casos onde a matriz envolvida é esparsa, uma vez que, nesse contexto, a operação que requer maior custo computacional é a multiplicação matriz-vetor. Além disso, se a matriz envolvida for não simétrica o ARPACK utiliza-se do processo de Arnoldi 2.5.2.4. Caso contrário, se for simétrica, utiliza-se do processo de Lanczos 2.5.2.3. Ambos, são baseados na fatoração QR com deslocamentos implícitos e foram implementados com reortogonalização. Por isso, são nomeados por IRAM (*Implicitly Restarted Arnoldi Method*) e IRLM (*Implicitly Restarted Lanczos Method*) respectivamente. Maiores informações sobre esses processos, bem como detalhes do tipo de reortogonalização utilizada, podem ser encontradas por meio da url [4].

A principal característica relacionada a essa biblioteca é a interface de comunicação reversa, na qual o usuário é livre para escolher o formato esparso mais adequado para seu contexto. Além do mais, outras características, como por exemplo, precisão simples, dupla e rotinas para realizar Decomposição em valores singulares, são disponibilizadas e possuem grande importância dentro do projeto. No entanto, devido a flexibilidade da interface, a única rotina que o usuário deve fornecer é a que descreve a operação multiplicação matriz-vetor, já que o formato de armazenamento é decidido pelo próprio usuário. Entretanto, todo o resto do processo de fatoração é realizado pelo ARPACK.

[3] `http://www.r-project.org/`
[4] `http://www.caam.rice.edu/software/ARPACK/`

6.3.1 Estrutura

O diretório raiz (ARPACK) possui três arquivos básicos, o READ-ME, composto por instruções básicas apresentadas a seguir, o Makefile e o ARmake.in, principal arquivo de configuração. Além do mais, possui os sete diretórios listados pelo Fluxograma da Figura 6.1. Desses sete, quatro são compostos pelas subrotinas implementadas em FORTRAN (BLAS, LAPACK, SRC e UTIL), um é composto por exemplos práticos de utilização (EXAMPLES) e dois são compostos por arquivos referentes à configuração do sistema, no qual, o primeiro deles (ARMAKES), possui informações referentes à plataforma que está sendo utilizada para executar o ARPACK, como por exemplo, CRAY, SP2, SUN4, LINUX, entre outros. Por outro lado, o segundo (DOCUMENTS) possui cinco arquivos no total, três referentes ao tipo de matriz (não simétrica, simétrica ou complexa), um referente às estatísticas de tempo e outro referente às opções para impressão de informações do sistema. Os diretórios ARMAKES e DOCUMENTS encontram-se descritos na Tabela 6.1.

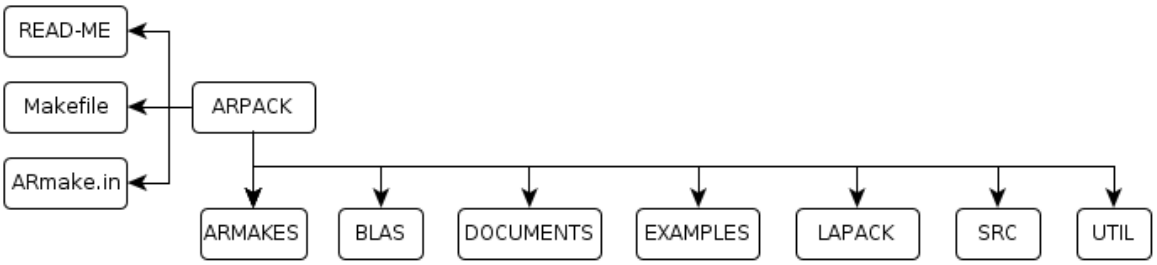


Figura 6.1: Estrutura de arquivos e diretórios do ARPACK.

Tabela 6.1: Diretórios contendo arquivos de configuração e informações sobre o sistema.

Diretório	Arquivos	Descrição
ARMAKES	ARmake.CRAY ARmake.SP2 ARmake.SUN4	Arquivos contendo informações específicas da plataforma escolhida. (Usados para compor o principal arquivo de configuração ARmake.in).
DOCUMENTS	debug.doc	Explica as diferentes opções referentes à impressão do ARPACK.
	stat.doc	Fornece estatísticas referentes ao tempo do ARPACK.
	ex-nonsym.doc	Exemplos de uso para matrizes não simétricas.
	ex-sym.doc	Exemplos de uso para matrizes simétricas.
	ex-complex.doc	Exemplos de uso para matrizes complexas.

6.3.2 Instalação, configurações e flags

As instruções a seguir são básicas e suficientes para instalar e compilar o ARPACK:

1. Obter o código do ARPACK cuja instalação consiste basicamente em extrair (descompactar) o diretório principal que, inclusive, pode ser encontrado em [5].

2. Editar o principal arquivo de configuração Armake.in. Alguns exemplos para configuração desse arquivo se encontram no diretório ARMAKES. Basicamente, sua configuração consiste em mudar o conteúdo de quatro variáveis, HOME para raiz do diretório ARPACK, PLAT para plataforma utilizada e, FC e FFLAGS de acordo com o compilador utilizado. Por exemplo, home = /home/speed/fonseca, PLAT = LINUX, FC = gfortran e FFLAGS = -02. É possível visualizar um exemplo completo da configuração em *ARmake.in para plataforma LINUX* 6.3.3.

3. Alterar o arquivo **second.f** que se encontra no diretório UTIL, de forma a tornar o código mais apropriado para o sincronismo do sistema. No caso específico da plataforma LINUX, o procedimento mais indicado seria comentar a linha EXTERNAL ETIME por meio do símbolo * (asterisco).

4. Executar no terminal o comando *make lib* que viabiliza a construção de uma biblioteca independente que inclui rotinas necessárias do BLAS 1/2/3 e LAPACK. Nesse momento, o sistema constrói e liga a biblioteca padrão nomeada por libarpack_$(PLAT).a, onde PLAT, no nosso caso, estará atribuída como LINUX. Além disso, compila via Make os códigos em FORTRAN que se encontram nos diretórios BLAS, LAPACK, SRC e UTIL, gerando executáveis .o a partir dos .f.

```
> make lib
> Making lib in /home/speed/fonseca/ARPACK/BLAS
> (...compila códigos em BLAS, LAPACK, SRC, UTIL...)
> ranlib /home/speed/fonseca/ARPACK/libarpack_LINUX.a
> ls
ARMAKES     BLAS       EXAMPLES  Makefile  SRC   libarpack_LINUX.a
ARmake.inc  DOCUMENTS  LAPACK    README    UTIL
```

Maiores informações de como utilizar o ARPACK por meio do LAPACK ou BLAS já instalados no sistema ou obtidos de outras fontes se encontram no arquivo READ-ME.

[5] http://www.caam.rice.edu/software/ARPACK/

6.3.3 ARmake.in para plataforma LINUX

```
############################################################################
#  Module:          ARmake.inc
############################################################################
#-------------------------------#
# SECTION 1: PATHS AND LIBRARIES |
#-------------------------------#
home       = /home/speed/fonseca/ARPACK
PLAT       = LINUX
BLASdir    = $(home)/BLAS
LAPACKdir  = $(home)/LAPACK
UTILdir    = $(home)/UTIL
SRCdir     = $(home)/SRC
DIRS       = $(BLASdir) $(LAPACKdir) $(UTILdir) $(SRCdir)
ARPACKLIB  = $(home)/libarpack_$(PLAT).a
ALIBS      = $(ARPACKLIB)
#-------------------------------#
# SECTION 2: COMPILERS          |
#-------------------------------#
.SUFFIXES:
.SUFFIXES: .f .o
.DEFAULT:
@$(ECHO) "Unknown target $@, try:  make help"
# Command to build .o files from .f files.
.f.o:
@$(ECHO) Making $@ from $<
@$(FC) -c $(FFLAGS) $<
FC      = gfortran
FFLAGS = -O2
LDFLAGS =
CD      = cd
ECHO    = echo
LN      = ln
LNFLAGS = -s
MAKE    = /usr/bin/make
RM      = rm
RMFLAGS = -f
SHELL   = /bin/sh
AR      = ar
ARFLAGS = rv
RANLIB  = ranlib
# This is the general help target.
help:
@$(ECHO) "usage: make ?"
```

Figura 6.2: Arquivo de configuração ARmake.in para LINUX.

6.3.4 Subrotinas

As principais subrotinas do ARPACK e o local no qual encontram-se exemplos básicos de como utilizá-las para calcular autovalores e autovetores de matrizes não simétricas, simétricas e complexas, podem ser visualizadas na Tabela 6.2. Vale ressaltar que todas as subrotinas possuem prefixos relacionados ao tipo de precisão utilizada, por exemplo, 'd' ocorre em subrotinas que utilizam precisão dupla (*double*), enquanto 's' ocorre nas subrotinas que utilizam precisão simples (*single*). No caso de matrizes compostas por elementos complexos, o prefixo 'c' define a precisão simples e 'z' define a precisão dupla.

Tabela 6.2: Principais subrotinas do ARPACK.

Tipo de matriz	ARPACK/EXAMPLES/	Subrotinas
Não simétrica	NONSYM	[d,s]naup.f e [d,s]neup.f
Simétrica	SYM	[d,s]saup.f e [d,s]seup.f
Complexa	COMPLEX	[c,z]naup.f e [c,z]neup.f

Além do cálculo de autovalores e autovetores, o ARPACK utiliza essas subrotinas para o cálculo específico de valores e vetores singulares, como pode ser observado a seguir.

6.3.5 Subrotinas para SVD

Do mesmo modo, exemplos de como utilizar tais subrotinas encontram-se no diretório SVD que, por sua vez, se encontra no diretório ARPACK/EXAMPLES. Além disso, o nome de cada programa possui a forma específica 'Xsvd.f' onde X é igual a 'd' ou 's'. Por exemplo, os parâmetros e as variáveis a serem definidas pelo usuário para modificar o exemplo /ARPACK/EXAMPLES/SVD/dsvd.f, se encontram nas Tabelas 6.3 e 6.4.

Tabela 6.3: Parâmetros a serem definidos pelo usuário em dsvd.f.

Parâmetro	Valor (padrão)	Descrição
MAXM	500	Número máximo de linhas alocadas para matriz A.
MAXN	250	Número máximo de colunas alocadas para matriz A.
MAXNEV	10	Número máximo de valores/vetores singulares alocados.
MAXNCV	25	Número máximo vetores usados para o processo de reinício implícito.

Tabela 6.4: Variáveis de dimensão e das subrotinas dsaup.f (autovalor) e dseup.f (autovetor).

Variável	Valor (padrão)	Descrição
m	500	Número de linhas alocadas para matriz A.
n	100	Número de colunas alocadas para matriz A.
nev	4	Número de valores/vetores singulares desejados.
nvc	10	Número de vetores usados para o processo de reinício implícito.
bmat	'I'	Problema padrão de autovalor $(A^TA)v = \sigma v$.
which	'LM'	Indica que os maiores autovalores em magnitude serão calculados.
tol	0	Tolerância utilizada. Se zero, usa a da máquina.
ido	0	Parâmetro da comunicação reversa. Inicialmente é sempre zero.
info	0	Indica qual vetor iniciará a iteração de ARNOLDI (Se zero, vetor aleatório).
iparam(1) = ishfts	1	Deslocamentos (Se zero, providos via interface. Se um, via matriz tridiagonal)
iparam(3) = maxitr	n	Número máximo de iterações de ARNOLDI alocadas.
iparam(7) = mode1	1	Tipo do problema a ser resolvido (1 para problema padrão).

Mais especificamente, a variável *which* viabiliza outras configurações em relação a que parte do espectro deseja-se calcular, como mostra as opções apresentadas na Tabela 6.5.

Tabela 6.5: Especifica os valores do espectro a serem calculados.

Valor	Descrição
'LA'	calcula maiores autovalores algébricos
'SA'	calcula menores autovalores algébricos
'LM'	calcula maiores autovalores em magnitude
'SA'	calcula menores autovalores em magnitude
'BE'	calcula metades de cada entre maiores e menores

As variáveis relacionadas à saída (impressão de resultados) do ARPACK dependem do tipo de matriz envolvida (não simétrica, simétrica ou matriz composta por elementos complexos). Além disso, das três variáveis que são comuns aos três casos, duas, $logfil$ (número da unidade onde o arquivo de log (debug) será escrito) e $ndigit$ (número de dígitos usados na saída do log, se $ndigit > 0$ usa 132 colunas e se $ndigit < 0$ usa 72 colunas) são utilizadas no código exemplo dsvd.f, onde $logfil = 6$ e $ndigit = -3$. Além dessas, as variáveis específicas, como por exemplo, as do caso simétrico, msgets, msaitr, msapps, msaupd, msaup2, mseigt e mseupd podem receber diversos valores conforme a descrição de cada uma. Para tanto, todas as possibilidades de configuração relacionadas à saída podem ser encontradas em ARPACK/DOCUMENTS/debug.doc.

Todas as variáveis dos exemplos disponibilizados junto ao código fonte do ARPACK encontram-se atribuidas com valores padrões. Além disso, o cálculo dos vetores singulares é opcional e pode ser realizado por meio da variável rvec ($rvec = .true.$). Para exibi-los, basta atribuir à variável msaitr, um valor maior que três (> 3), como indicado pelo arquivo de configurações ARPACK/DOCUMENTS/debug.doc. Maiores informações sobre os exemplos relacionados ao procedimento da SVD e aos formatos de impressão, se encontram no arquivo ARPACK/EXAMPLES/SVD/README.

Finalmente, o Exemplo 6.1 mostra o uso da SVD de uma matriz gerada pelo próprio programa. Todavia, a grande vantagem do ARPACK é permitir ao usuário escolher o formato esparso desejado, desde que forneça as rotinas de multiplicação matriz-vetor envolvidas ($w \leftarrow Ax$ e $y \leftarrow A^T w$). Aliás, essa flexibilidade é obtida por meio de uma interface que evita ter de se expressar o produto matriz-vetor por meio de uma subrotina com chamada fixa, como mostra a Seção Interface de comunicação reversa 6.3.6

Exemplo 6.1 [Executando dsvd.f com M = 500, N = 100, NEV = 4 e NVC = 10.]

```
fonseca@cisteina:~/ARPACK/EXAMPLES/SVD$ make dsvd
Making dsvd.o from dsvd.f
gfortran -O2 dsvd.o /home/speed/fonseca/ARPACK/libarpack_LINUX.a  -o dsvd
fonseca@cisteina:~/ARPACK/EXAMPLES/SVD$ ./dsvd
 Singular values and direct residuals
 -------------------------------------
               Col   1       Col   2
  Row   1:    4.10123D-02   2.89559D-17
  Row   2:    6.04881D-02   1.39054D-17
  Row   3:    1.17844D-01   1.35936D-16
  Row   4:    5.57234D-01   4.22583D-16

  _SVD
  ====
  Size of the matrix is                                    100
  The number of Ritz values requested is                   4
  The number of Arnoldi vectors generated (NCV) is         10
  What portion of the spectrum:                            LM
  The number of converged Ritz values is                   4
  The number of Implicit Arnoldi update iterations taken is 4
  The number of OP*x is                                    22
  The convergence criterion is                             1.11022302462515654E-016
```

■

6.3.6 Interface de comunicação reversa

A interface de comunicação reversa nada mais é que a possibilidade de utilizar as subrotinas av ($w \leftarrow Ax$) e atv ($y \leftarrow A^T w$) implementadas pelo usuário, de acordo com o formato esparso escolhido pelo mesmo para representar a matriz de dados desejada. Observe que o problema de calcular $y = A^T Ax$ é quebrado em dois problemas, calcular $w \leftarrow Ax$ e em seguida $y \leftarrow A^T w$. Supondo que a matriz esteja representada no formato CSC (Val, IndLIn e pCol), as subrotinas av( ) e atv( ) devem ser implementadas pelo usuário realizando as respectivas operações de multiplicação matriz-vetor vistas nas Seções 5.4 e 5.5. A interface de comunicação pode ser observada pelo Exemplo 6.2.

Exemplo 6.2 [Trecho do código que realiza a comunicação reversa.]

```
c     %---------------------------------------------%
c     | M A I N   L O O P (Reverse communication loop) |
c     %---------------------------------------------%
 10   continue
c        %-------------------------------------------%
c        | Repeatedly call the routine DSAUPD and take |
c        | actions indicated by parameter IDO until    |
c        | either convergence is indicated or maxitr   |
c        | has been exceeded.                          |
c        %-------------------------------------------%
         call dsaupd ( ido, bmat, n, which, nev, tol, resid,
     &                 ncv, v, ldv, iparam, ipntr, workd, workl,
     &                 lworkl, info )
         if (ido .eq. newprod) then
c           %--------------------------------------%
c           | Perform matrix vector multiplications |
c           |              w <--- A*x       (av())  |
c           |              y <--- A'*w      (atv()) |
c           | The user should supply his/her own    |
c           | matrix vector multiplication routines |
c           | here that takes workd(ipntr(1)) as    |
c           | the input, and returns the result in  |
c           | workd(ipntr(2)).                      |
c           %--------------------------------------%
            call av  (PCol,IndLin,Val,nCol,nnZero,m,n,workd(ipntr(1)),ax)
            call atv (PCol,IndLin,Val,nCol,nnZero,m,n,ax,workd(ipntr(2)))
            go to 10
         end if
```

■

6.4 Subrotinas av( ) e atv( ) para SVD

As subrotinas de multiplicação matriz-vetor e (matriz transposta)-vetor para o formato HB (CSC) utilizadas nos experimentos, baseiam-se em matrizes do tipo RUA.

Exemplo 6.3 [Subrotina av($w \leftarrow Ax$) para CSC (MXTYPE = RUA).]

```
subroutine av (PCol, IndLin, Val, nCol, nnZero, m, n, x, w)
        integer          m, n, i
        Double precision x(n), w(m)
        integer          (kind = 4) PCol(nCol+1)
        integer          (kind = 4) IndLin(nnZero)
        real             (kind = 8) Val(nnZero)
        parameter        (zero = 0.0D+0)
        do 30 i = 1, m
                w(i) = zero
  30    continue
                do 31 i = 1, n
                        do 32 k = PCol(i), PCol(i+1)-1
                                w(IndLin(k)) = w(IndLin(k)) + Val(k)*x(i)
  32    continue
  31    continue
        return
end
```

■

Exemplo 6.4 [Subrotina atv($y \leftarrow A^T w$) para CSC (MXTYPE = RUA).]

```
subroutine atv (PCol, IndLin, Val, nCol, nnZero, m, n, w, y)
        integer          m, n, j
        Double precision w(m), y(n)
        integer          (kind = 4) PCol(nCol+1)
        integer          (kind = 4) IndLin(nnZero)
        real             (kind = 8) Val(nnZero)
        parameter        (zero = 0.0D+0)
        do 33 j = 1, n
                y(j) = zero
                do 34 k = PCol(j), PCol(j+1)-1
                        y(j) = y(j) + Val(k)*w(IndLin(k))
  34    continue
  33    continue
        return
end
```

■

6.5 Validação da SVD pelo ARPACK via SCILAB

A fim de avaliar o resultado da SVD executada via ARPACK, foram realizados vários testes que, inclusive, foram comparados com os respectivos resultados fornecidos pelo SCILAB. Para tanto, foram utilizadas algumas matrizes conhecidas e armazenadas no formato Harwell-Boeing que podem ser obtidas em [6]. Dentre essas, a matriz utilizada pelo Exemplo 5.4 foi escolhida para ser apresentada por meio dos Exemplos 6.5 e 6.6.

Exemplo 6.5 [SVD (k = 3) da matriz do Exemplo 5.4 via SCILAB.]

```
--> // Lê matriz no formato Harwell-Boeing
--> A = ReadHBSparse("rua_5x5.dat");
--> // Exibe matriz completa
--> full(A)
    ans  =
    11.    0.     0.     14.    0.
    0.     22.    0.     0.     0.
    31.    32.    33.    34.    35.
    0.     0.     0.     44.    45.
    51.    52.    0.     0.     55.
--> // Método iterativo do SVD via SCILAB = sva(matriz,k)
--> [U,S,V] = sva(A,3)
    V  =
  - 0.4701299    0.3624744    0.1362720
  - 0.4906255    0.4642429    0.1903835
  - 0.1624496  - 0.1228914    0.7811656
  - 0.3113783  - 0.7691642    0.2057721
  - 0.6441414  - 0.2153483  - 0.5409457
    S  =
    118.19164    0.           0.
    0.           55.935955    0.
    0.           0.           31.946168
    U  =
  - 0.0806379  - 0.1212294    0.1370994
  - 0.0913242    0.1825900    0.1311092
  - 0.5818237  - 0.2083045    0.7562196
  - 0.3611678  - 0.7782812  - 0.4785734
  - 0.7184682    0.5503199  - 0.4038732
```

■

[6] http://people.sc.fsu.edu/~jburkardt/datasets/hbsmc/hbsmc.html

Exemplo 6.6 [SVD (k=3) da matriz do Exemplo 5.4 via ARPACK]

```
Entre com o nome do arquivo de origem:
rua_5x5.dat
Entre com a quantidade desejada de valores/vetores singulares:
3
Title                                                                   Key
             5             1             1             3             0
RUA                        5             5            13             0
(6I3)           (13I3)          (5E15.8)
  1  4  7  8 11 14
  1  3  5  2  3  5  3  1  3  4  3  4  5
 0.11000000E+02 0.31000000E+02 0.51000000E+02 0.22000000E+02 0.32000000E+02
 0.52000000E+02 0.33000000E+02 0.14000000E+02 0.34000000E+02 0.44000000E+02
 0.35000000E+02 0.45000000E+02 0.55000000E+02
-------------------------------------
 ARPACK
 -------------------------------------
 Valores singulares

  1.18192D+02
  5.59360D+01
  3.19462D+01

 Vetores singulares

 Matrix V
 ------------------------------------------------------------------------
  0.47012992458864378       0.36247443107054189      -0.13627198215968905
  0.49062552099981782       0.46424292277129819      -0.19038349641803287
  0.16244956824966172      -0.12289140454336242      -0.78116557835185141
  0.31137833929471537      -0.76916421876808605      -0.20577207819676557
  0.64414138179711578      -0.21534832883746491       0.54094571095694655
 Matrix U
 ------------------------------------------------------------------------
  8.06379028824096428E-002 -0.12122936533948164      -0.13709941412067558
  9.13242386528972799E-002  0.18258996929586516      -0.13110921278925250
  0.58182365919022860      -0.20830448700145748      -0.75621959700166030
  0.36116775612014773      -0.77828116897018385       0.47857338051165554
  0.71846815767911798       0.55031991315359918       0.40387320478813976
```

■

6.6 Manipulação álgebrica para cálculo da PCA

A vantagem de utilizar o ARPACK para realizar a PCA é a possibilidade de manter a esparsidade da matriz original durante todo o processo de obtenção das componentes principais. Para tanto, as etapas de subtrair a média referente aos elementos de cada coluna e/ou dividir pelo desvio padrão da respectiva coluna, de todos os valores da matriz, inclusive, dos elementos nulos, foram realizadas de forma implícita no cálculo dos vetores w definido em (6.1) e y definido em (6.2), referentes às subrotinas av( ) e atv( ), utilizadas para a realização das PCAs via matriz de covariância e correlação.

Dado que as matrizes de covariância e correlação são semi-definidas positiva, isto é, $\lambda_1 > \lambda_2 > ... > \lambda_n > 0$, o procedimento de encontrar as componentes principais consiste em realizar a decomposição espectral dessas matrizes, onde os autovalores são obtidos a partir dos valores singulares resultantes da aplicação da SVD. Para tanto, foram realizadas manipulações algébricas tanto para otimizar o cálculo da PCA de matrizes esparsas de alta ordem, quanto para obter os respectivos autovalores envolvidos.

6.6.1 PCA via matriz de covariância

Para descrever o cálculo da PCA via matriz de covariância matendo a esparsidade da matriz original é apresentada a quebra do processo $y = A^T Ax$ em dois outros $w = Ax$ e $y = A^T w$, juntamente com o passo de subtrair de cada elemento da matriz, a média de sua coluna correspondente. É possível observar pela equação (6.1) que realizar a mutliplicação do resultado da matriz A menos as médias pelo vetor x é o mesmo que multiplicar A por x e, em seguida, subtrair do vetor w o resultado da média vezes o vetor x. Do mesmo modo, pela equação (6.2) é possível ver que realizar a multiplicação do resultado da transposta da matriz A menos as médias pelo vetor w é o mesmo que multiplicar A por w e, em seguida, subtrair do vetor y a multiplicação do transposto do vetor de médias pelo vetor w.

$$w = (A - \text{média})x = (A)x - (\text{média})x \tag{6.1}$$

$$y = (A - \text{média})^T w = (A^T)w - (\text{média}^T)w \tag{6.2}$$

Por outro lado, a relação existente entre os autovalores da matriz de covariância e correlação e os valores singulares (s) resultantes da SVD é a mesma. Em ambos os casos, os autovalores (a) são obtidos por meio da expressão (6.3).

$$a = s^2/(n-1) \tag{6.3}$$

Os Exemplos 6.7 e 6.8 ilustram as subrotinas implementadas na linguagem de programação FORTRAN, utilizadas durante os experimentos com bases sintéticas e reais.

6.6.2 Subrotinas av( ) e atv( ) para PCA (covariância)

Exemplo 6.7 [Subrotina av($w \leftarrow Ax$).]

```
subroutine av (PCol, IndLin, Val, nCol, nnZero, m, n, mds, x, w)

        integer          m, n, i, j, l, d
        Double precision x(n), w(m)
        integer          (kind = 4) PCol(nCol+1)
        integer          (kind = 4) IndLin(nnZero)
        integer          (kind = 4) Val(nnZero)
        real             (kind = 8) mds(n)
        real             valor_inicial

        valor_inicial = 0
        do 29 d = 1, n
                valor_inicial = valor_inicial + mds(d) * x(d)
  29    continue
        do 30 j = 1, m
                w(j) = - valor_inicial
  30    continue
        do 31 i = 1, n
                do 32 k = PCol(i), PCol(i+1)-1
                        w(IndLin(k)) = w(IndLin(k)) + Val(k)*x(i)
  32    continue
  31    continue
        return
end
```

■

Exemplo 6.8 [Subrotina atv($y \leftarrow A^T w$).]

```
subroutine atv (PCol, IndLin, Val, nCol, nnZero, m, n, mds, w, y)

        integer          m, n, i, j, d
        Double precision w(m), y(n)
        integer         (kind = 4) PCol(nCol+1)
        integer         (kind = 4) IndLin(nnZero)
        integer         (kind = 4) Val(nnZero)
        real            (kind = 8) mds(n)
        real            valor_inicial

        do 35 i = 1, n
                valor_inicial = 0
        do 30 j = 1, m
                valor_inicial=valor_inicial+mds(i)*w(j)
  30    continue
                y(i) = - valor_inicial
  35    continue

        do 33 j = 1, n
                do 34 k = PCol(j), PCol(j+1)-1
                        y(j) = y(j) + Val(k)*w(IndLin(k))
  34    continue
  33    continue
        return
end
```

■

6.6.3 PCA via matriz de correlação

Analogamente ao cálculo da PCA via matriz de covariância, para calcular a PCA via matriz de correlação basta descrever a quebra do processo $y = A^T Ax$ em dois outros, $w = Ax$ e $y = A^T w$, juntamente com o passo de subtrair e dividir, respectivamente, cada elemento da matriz original, pela média e pelo desvio padrão de sua correspondente coluna. A manipulação algébrica para manter a esparsidade da matriz pode ser observada pelas equações (6.4) e (6.5). Os autovalores da matriz de correlação são obtidos pelo mesmo processo de obtenção dos autovalores da matriz de covariância.

$$w = (\frac{A - \text{média}}{\text{desvio}})x = (\frac{A}{\text{desvio}})x - (\frac{\text{média}}{\text{desvio}})x \tag{6.4}$$

$$y = (\frac{A - \text{média}}{\text{desvio}})^T w = (\frac{A^T}{\text{desvio}})w - (\frac{\text{média}^T}{\text{desvio}})w \tag{6.5}$$

Os Exemplos 6.9 e 6.10 ilustram as subrotinas implementadas na linguagem de programação FORTRAN, utilizadas durante os experimentos com bases sintéticas e reais.

6.6.4 Subrotinas av( ) e atv( ) para PCA (correlação)

Exemplo 6.9 [Subrotina av($w \leftarrow Ax$).]

```
subroutine av (PCol,IndLin,Val,nCol,nnZero,m,n,mds,stds,x,w)
        integer          m, n, i, j, l, d
        Double precision x(n), w(m)
        integer          (kind = 4) PCol(nCol+1)
        integer          (kind = 4) IndLin(nnZero)
        integer          (kind = 4) Val(nnZero)
        real             (kind = 8) v(n)
        real             (kind = 8) mds(n)
        real             (kind = 8) stds(n)
        real             valor_inicial
        do 28 i = 1, n
                v(i) = -(mds(i) / stds(i))
  28    continue
        valor_inicial = 0
        do 29 d = 1, n
                valor_inicial = valor_inicial + v(d) * x(d)
  29    continue
        do 30 j = 1, m
                w(j) = valor_inicial
  30    continue
        do 31 i = 1, n
               do 32 k = PCol(i), PCol(i+1)-1
                       w(IndLin(k)) = w(IndLin(k))+Val(k)/stds(i)*x(i)
  32    continue
  31    continue
        return
end
```

■

Exemplo 6.10 [Subrotina atv($y \leftarrow A^T w$).]

```
subroutine atv (PCol,IndLin,Val,nCol,nnZero,m,n,mds,stds,w,y)
        integer          m, n, i, j, d
        Double precision w(m), y(n)
        integer         (kind = 4) PCol(nCol+1)
        integer         (kind = 4) IndLin(nnZero)
        integer         (kind = 4) Val(nnZero)
        real            (kind = 8) mds(n)
        real            (kind = 8) stds(n)
        real            valor_inicial
        do 35 i = 1, n
                valor_inicial = 0
                do 30 j = 1, m
                        valor_inicial=valor_inicial+mds(i)/stds(i)*w(j)
  30    continue
                y(i) = valor_inicial
  35    continue
        do 33 j = 1, n
                do 34 k = PCol(j), PCol(j+1)-1
                        y(j) = y(j) + (Val(k) / stds(j))*w(IndLin(k))
  34    continue
  33    continue
        return
end
```

■

6.7 Validação da PCA pelo ARPACK via R

A fim de avaliar o resultado da PCA executada via ARPACK, foram realizados vários testes que, inclusive, foram comparados com os respectivos resultados obtidos pelo R. Para tanto, também foram utilizadas algumas matrizes conhecidas e armazenadas no formato Harwell-Boeing. Dentre essas, a matriz utilizada pelo Exemplo 5.4 foi escolhida para ser apresentada e comparada por meio dos Exemplos 6.11 e 6.12 e 6.13.

Exemplo 6.11 [PCA via R para matriz do Exemplo 5.4.]

```
> A = read.table("/home/speed/fonseca/Desktop/A")
> A
  V1 V2 V3 V4 V5
1 11  0  0 14  0
2  0 22  0  0  0
3 31 32 33 34 35
4  0  0  0 44 45
5 51 52  0  0 55

> // PCA via matriz de covariância!
> pcadat = prcomp(A)
> pcadat

Standard deviations:
[1] 3.562861e+01 2.583065e+01 1.586325e+01 8.083129e+00 2.211360e-15

Rotation:
           PC1        PC2        PC3         PC4        PC5
V1  0.58182667  0.1400448 -0.1599356 -0.77316792  0.1360042
V2  0.55748458  0.3164504 -0.1663989  0.59203721  0.4592095
V3  0.12857078 -0.1977561 -0.8204620  0.14254323 -0.5008848
V4 -0.09066097 -0.7509898 -0.2173839 -0.05114619  0.6147546
V5  0.57091052 -0.5264536  0.4757292  0.16961369 -0.3765909

> // PCA via matriz de correlação!
> pcadat = prcomp(A, scale=TRUE)
> pcadat

Standard deviations:
[1] 1.573004e+00 1.265622e+00 8.890513e-01 3.653044e-01 1.046233e-16

Rotation:
          PC1         PC2          PC3         PC4        PC5
V1  0.6069734 -0.06925162  0.005145356 -0.77768675  0.1482037
V2  0.5920823 -0.21557483  0.131156920  0.57782015  0.5018842
V3  0.2506817  0.51305949  0.730593440  0.08533226 -0.3645275
V4 -0.1373195  0.76105016 -0.178171793 -0.06075379  0.6054005
V5  0.4464570  0.32604342 -0.645956210  0.22439824 -0.4761910
```

■

Exemplo 6.12 [PCA (covariância) via ARPACK para matriz do Exemplo 5.4]

```
Entre com o nome do arquivo de origem:
rua_5x5.dat
Entre com a quantidade desejada de componentes principais:
3
Title                                                                    Key
             5             1             1             3             0
RUA                        5             5            13             0
(6I3)           (13I3)          (5E15.8)
  1  4  7  8 11 14
  1  3  5  2  3  5  3  1  3  4  3  4  5
 0.11000000E+02 0.31000000E+02 0.51000000E+02 0.22000000E+02 0.32000000E+02
 0.52000000E+02 0.33000000E+02 0.14000000E+02 0.34000000E+02 0.44000000E+02
 0.35000000E+02 0.45000000E+02 0.55000000E+02
 ---------------------------------------------------------------------------
 medias :    18.600000381469727    21.200000762939453    6.5999999046325684
             18.399999618530273    27.000000000000000
---------------------------------------------------------------------------
 ARPACK
 ---------------------------------------------------------------------------
 Autovalores

   1269.3979874925587
   667.22221578037147
   251.64281039250781

 Principal components
 ---------------------------------------------------------------------------
 -0.58182667153384615      -0.14004483883853652       0.15993567189897823
 -0.55748457325104961      -0.31645044589639693       0.16639885518481864
 -0.12857078252372481       0.19775614973930394       0.82046200519172474
  9.0660981134019658E-002  0.75098981901310746       0.21738392537021900
 -0.57091051423142991       0.52645356432035562      -0.47572926005745519
```

■

Exemplo 6.13 [PCA (correlação) via ARPACK para matriz do Exemplo 5.4]

```
Entre com o nome do arquivo de origem:
rua_5x5.dat
Entre com a quantidade desejada de componentes principais:
3
Title                                                                   Key
             5             1             1             3             0
RUA                        5             5            13             0
(6I3)           (13I3)          (5E15.8)
  1  4  7  8 11 14
  1  3  5  2  3  5  3  1  3  4  3  4  5
 0.11000000E+02 0.31000000E+02 0.51000000E+02 0.22000000E+02 0.32000000E+02
 0.52000000E+02 0.33000000E+02 0.14000000E+02 0.34000000E+02 0.44000000E+02
 0.35000000E+02 0.45000000E+02 0.55000000E+02
 ------------------------------------------------------------------------
 medias :    18.600000381469727    21.200000762939453    6.5999999046325684
             18.399999618530273    27.000000000000000
 desvios:    22.097511291503906    22.163032531738281    14.758049011230469
             19.969976425170898    25.641763687133789
 ------------------------------------------------------------------------
 ARPACK
 ------------------------------------------------------------------------
 Autovalores

   2.4743411620394782
   1.6017992635301119
   0.79041208657628159

 Principal components
 ------------------------------------------------------------------------
 -0.60697345713129824      -6.92515214698584E-002     5.14525744655752E-003
 -0.59208225687374760      -0.21557481491683328       0.13115703391066000
 -0.25068167286072940       0.51305947097844085       0.73059345383376606
  0.13731965268417029       0.76105015805496279      -0.17817170370802088
 -0.44645698035960257       0.32604349726918602      -0.64595619665314485
```

■

Capítulo 7

Análise comparativa sobre a tarefa de classificação de dados

A crescente capacidade de armazenamento, bem como a dinamicidade propiciada pela *Web*, permitiram o surgimento e a consolidação de grandes volumes de dados, constituindo um grande desafio computacional. Mesmo técnicas, oriundas da Mineração de Dados (MD), destinadas às análises de grandes massas de dados, estão sendo constantemente redefinidas de forma a tratar tais dados de forma eficiente. Steinbach [31], inclusive, menciona que o crescimento dos dados a grandezas suficientemente altas, por impor novos desafios, são capazes de representar não somente uma mera mudança quantitativa de análise, mas também qualitativa. Isto é, por um lado, o aumento do número de objetos ou instâncias analisados impõe um desafio de eficiência e, por outro, o aumento do número de características ou dimensões de representação acerca de cada objeto provoca desafios de eficácia, por extrapolar premissas usualmente aceitas por diversas técnicas. Aliás, o mais notório exemplo de desafio é o denominado "Mal da Dimensionalidade", o qual estabelece que um número fixo de objetos tornam-se crescentemente "esparsos" à medida que cresce o número de dimensões, dificultando mensurações de distância ou similaridade entre os objetos [31].

Nesse contexto, apesar da grande popularidade das técnicas SVD e PCA para redução de dimensionalidade e aproximação de bases de dados para técnicas de MD, há pouco entendimento sobre o impacto de tal processo sobre as premissas assumidas pelos algoritmos tradicionais dessa área. Salvo em estudos mais teóricos, os conceitos de

ambas as técnicas são, em geral, apresentados de forma confusa. Existem perguntas que, apesar de pertinentes, são negligenciadas na literatura. Por exemplo: 1) Como premissas básicas de um dado algoritmo podem ser afetadas por transformações sobre os dados realizadas por essas técnicas? 2) Quando utilizar a SVD ou a PCA? 3) Há alguma característica dos dados, ou mesmo do algoritmo de Mineração, que beneficia o uso de uma ou outra técnica?. Dessa forma, o objetivo do capítulo é apresentar uma metodologia de análise comparativa entre as técnicas SVD e PCA sobre a tarefa de classificação de dados por meio do algoritmo KNN (*k-nearest neighbor*), baseada em três diferentes estudos de caso, os quais são apresentados na Seção 7.1.

Em se tratando da classificação em si, foram utilizadas duas métricas para avaliar a qualidade do algoritmo: eficácia, que é a taxa de acerto do algoritmo e F1, que é média harmônica entre outras duas métricas, precisão (*precision*) e revocação (*recall*). Para apresentá-las no contexto de classificação, os seguintes termos: verdadeiros positivos (tp = *true positives*), verdadeiros negativos (tn = *true negatives*), falsos positivos (fp = *false positives*) e falsos negativos (fn = *false negatives*) são utilizados para comparar a classe prevista de um item (classe atribuída pelo classificador) com a classe real do próprio item. Dessa forma, como descrito em (7.1), a precisão representa a proporção do número de casos positivos previstos que estavam corretos e a revocação representa a proporção do número de casos positivos que foram corretamente identificados.

$$\text{precisão} = \frac{tp}{tp + fp} \qquad \text{e} \qquad \text{revocação} = \frac{tp}{tp + fn}. \tag{7.1}$$

Logo, a métrica F1 pode ser descrita por (7.2) e a eficácia (*accuracy*) por (7.3).

$$\text{F1} = 2 \times \frac{\text{precisão} \times \text{revocação}}{\text{precisão} + \text{revocação}}, \tag{7.2}$$

$$\text{eficácia} = \frac{tp + tn}{tp + tn + fp + fn}. \tag{7.3}$$

Vale ressaltar que o Algoritmo KNN é baseado em um número k de vizinhos para definição das classes, sendo esse, o único parâmetro do algoritmo. Aliás, até então, não há um critério bem estabelecido para sua escolha e, por isso, os valores usualmente utilizados na literatura no contexto de MD se encontram no intervalo entre 30 e 50.

Portanto, um dos principais objetivos de aplicar especificamente esse algoritmo (KNN) é averiguar a possibilidade de que exista uma relação entre o valor do parâmetro k (número de vizinhos) a ser utilizado e a redução de dimensionalidade. Além disso, é um dos algoritmos de menor custo e um dos mais bem estabelecidos na área de MD. Dessa forma, para fins de comparação, esse trabalho apresenta uma metodologia, na qual as tarefas realizadas foram divididas em quatro etapas principais: 1) caracterização dos dados esparsos oriundos de bases reais, 2) definição do critério de escolha do posto para aproximação de posto incompleto, 3) análise geométrica dos dados submetidos à redução por meio das duas técnicas e, 4) análise da distribuição de frequência das distâncias entre as entidaddes antes e depois da redução de dimensionalidade.

7.1 Estudos de casos

De forma a melhor descrever os experimentos realizados, são descritas as três coleções utilizadas, constituídas, por sua vez, de características distintas. A primeira coleção (*Reuters*) contém 8184 documentos de notícias organizados em 8 classes distintas e compostos por 24985 termos distintos [12]. A segunda coleção (*Nature*) é composta por periódicos da revista *Nature*, semanalmente publicados e organizados em cinco classes distintas. Nesta coleção, há 7964 artigos de periódicos com texto completo publicados entre 01/01/2005 e 21/12/2006, compostos por 104288 termos distintos [29]. Por fim, a terceira coleção utilizada (*Sequence Triples*), é constituída por uma base de 1003 proteínas descritas por 7525 sequências de tripletos, organizadas em 8 classes distintas [27]. Todos os objetos de cada coleção foram pré-processados a fim de remover *stopwords*. Além disso, é importante ressaltar que cada objeto das coleções utilizadas no estudo de caso é atribuído à apenas uma classe.

7.1.1 Caracterização dos dados

A caracterização tem como principal objetivo avaliar o balanceamento das entidades nas respectivas classes de cada base de dados e a distribuição da média e do desvio padrão de cada instância ou dimensão, uma vez que constituem aspectos que podem influenciar tanto o algoritmo de classificação quanto as técnicas. Os três gráficos apresentados na Figura 7.1 mostram, respectivamente, o desbalanceamento das classes da base *Reuters* e muitos baixos valores de médias e desvios padrões de seus atributos.

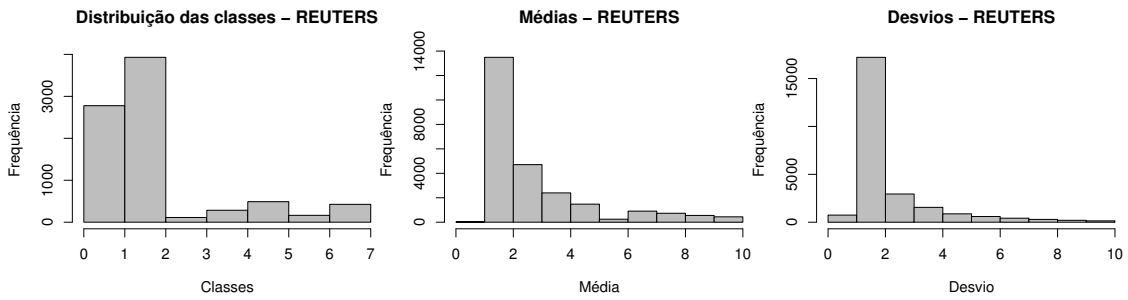


Figura 7.1: *Reuters*: distribuição das classes, médias e desvios dos atributos.

A coleção da *Nature* também apresenta alto desbalanceamento das classes e, em grande parte, médias e desvios padrões próximos de zero.

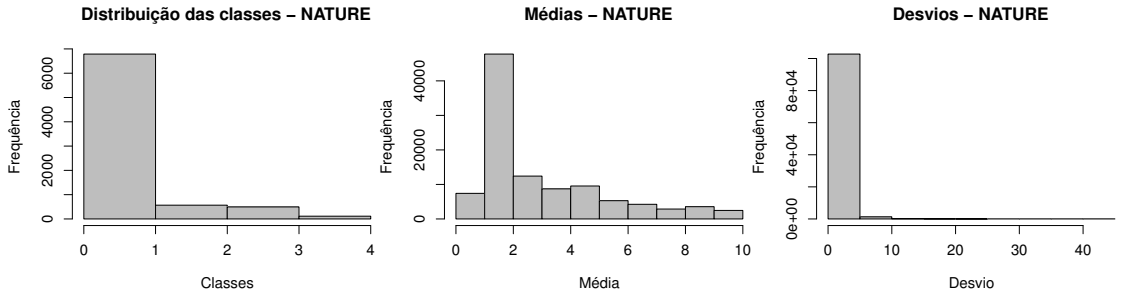


Figura 7.2: *Nature*: distribuição das classes, médias e desvios dos atributos.

Por outro lado, a coleção *Sequence* é a única que apresenta melhor balaceamento das entidades por classe. No entanto, assim como as outras bases de dados, apresenta valores baixos de médias e desvios.

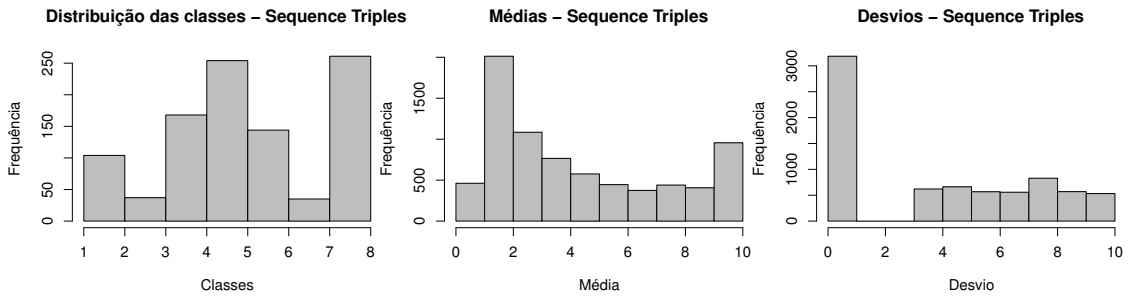


Figura 7.3: *Sequence*: distribuição das classes, médias e desvios dos atributos.

7.1.2 Escolha de posto para aproximação de posto incompleto

Essa etapa consiste em justificar a escolha do valor do posto para aproximação de posto incompleto que, por sinal, não é uma tarefa simples, uma vez que a distribuição dos valores singulares e autovalores pode apresentar um comportamento bem suave no seu decaimento. Além do mais, é notável que as distribuições dos valores singulares e autovalores observadas comumente na literatura e durante os experimentos, podem ser divididas em pelo menos duas principais regiões distintas. A primeira está associada, em geral, com uma grande variação decrescente de informação, enquanto que a segunda região está associada a um aparente comportamento de regime linear, capaz de manter a taxa de variação praticamente constante. Dessa forma, a questão crucial para o corte a ser realizado é encontrar em qual ponto uma região termina e a outra começa.

Para tanto, a modelagem matemática utilizada nesse trabalho que sustenta a escolha desse ponto de transição está fundamentada na aproximação da derivada segunda da região linear. Logo, o ponto no qual a derivada segunda dessa região linear se anula, parece ser o local mais provável de transição entre essas as regiões. A princípio, a atitude mais prudente parece ser escolher um intervalo potencial que represente boas estimativas de cortes para os valores singulares e autovalores da matriz de dados. Portanto, essa etapa da metodologia consiste em escolher três pontos:

1) **Prematuro**, ponto que pertence à região de maior variação.

2) **Transição**, suposto ponto que melhor separa as duas regiões.

3) **Estável**, ponto já pertencente ao regime linear.

A hipótese que sustenta a necessidade de identificar esses três pontos é que não há ganho relevante de informação após o ponto de transição. Para verificar isso, dado que a derivada segunda de um regime linear se anula, auxiliando a escolha desse suposto ponto de transição, a intenção é reforçar a importância da identificação desse ponto, mostrando que a escolha de um ponto prematuro implica na variação dos resultados e a escolha de um ponto estável apresenta estabilidade dos resultados, retornando um ganho insignificativo em relação ao ganho já obtido por meio do ponto de transição.

A seguir, se encontram os três pontos escolhidos (prematuro, transição e estável) referentes às bases reais *Reuters*, *Nature* e *Sequence* para SVD e PCA.

7.1.2.1 Base: Reuters

Distribuição dos valores singulares e autovalores e suas respectivas derivadas.

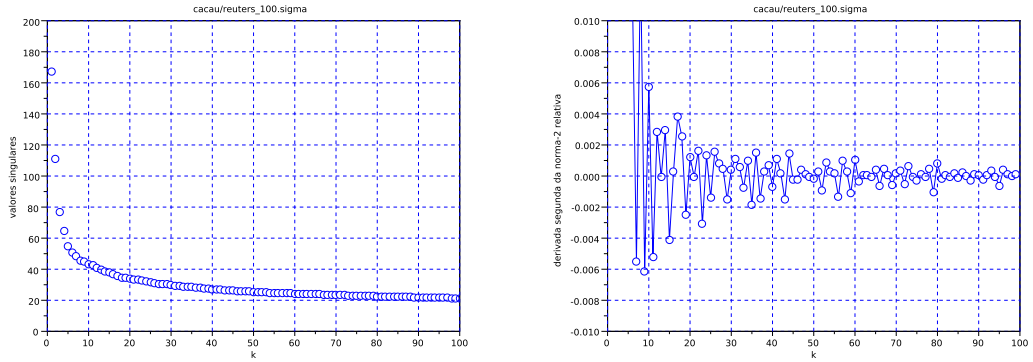


Figura 7.4: SVD - cortes escolhidos 10, 25 e 40.

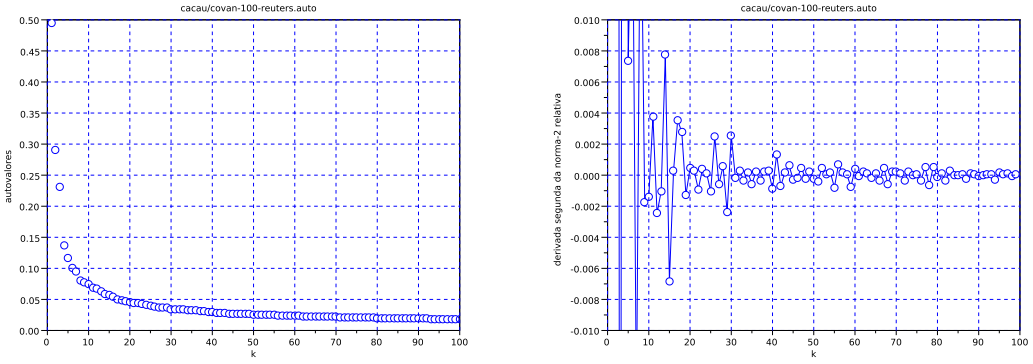


Figura 7.5: PCA via matriz de covariância - cortes escolhidos 10, 30 e 50.

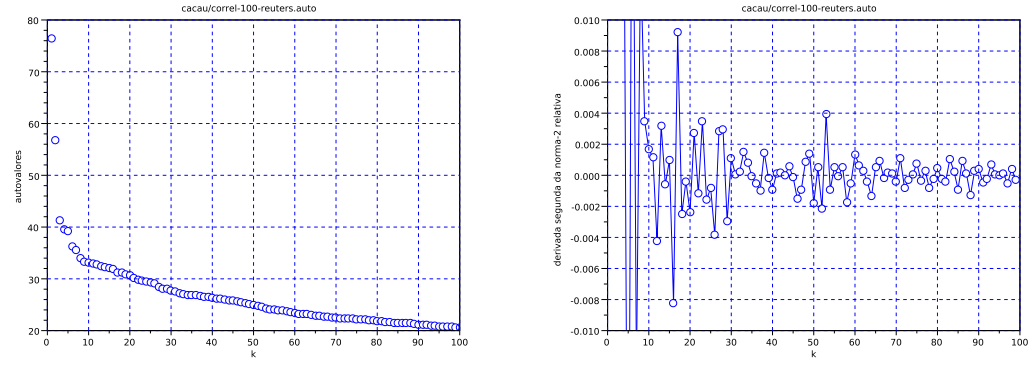


Figura 7.6: PCA via matriz de correlação - cortes escolhidos 10, 30 e 50.

7.1.2.2 Base: Nature

Distribuição dos valores singulares e autovalores e suas respectivas derivadas.

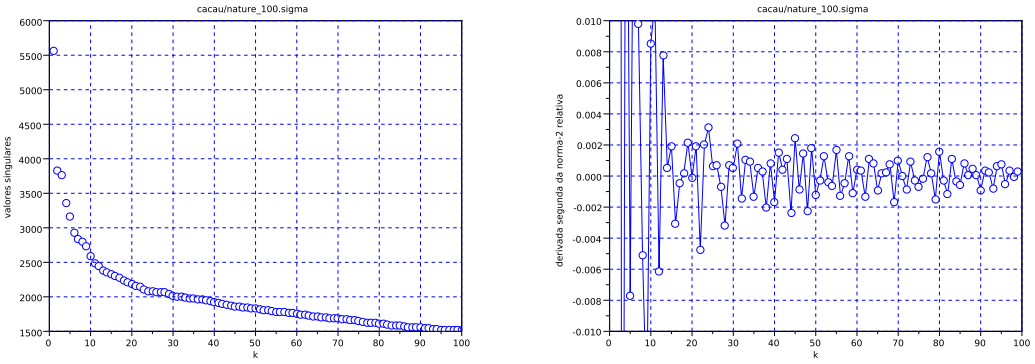


Figura 7.7: SVD - cortes escolhidos 15, 25 e 35.

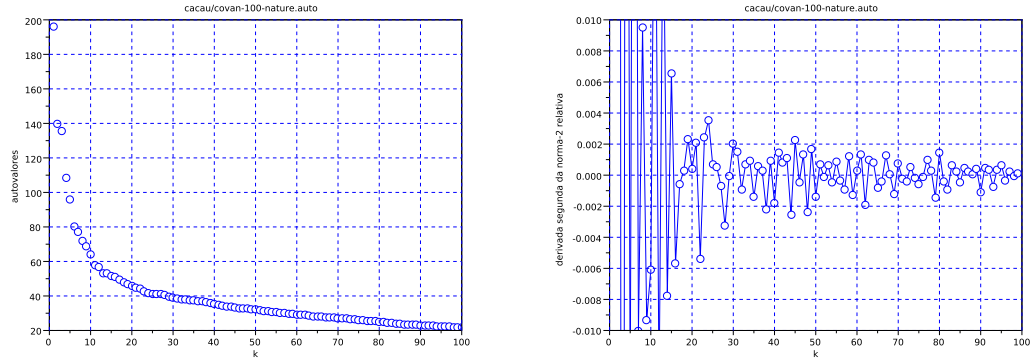


Figura 7.8: PCA via matriz de covariância - cortes escolhidos 15, 25 e 35.

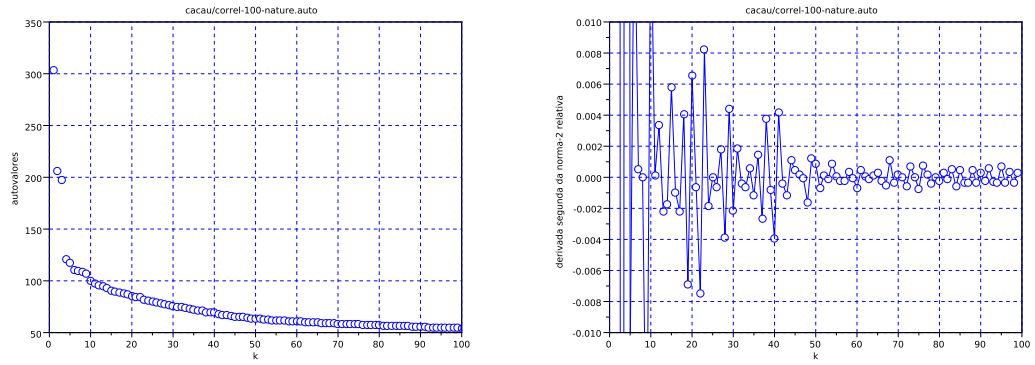


Figura 7.9: PCA via matriz de correlação - cortes escolhidos 20, 40 e 60.

7.1.2.3 Base: Sequence Triples

Distribuição dos valores singulares e autovalores e suas respectivas derivadas.

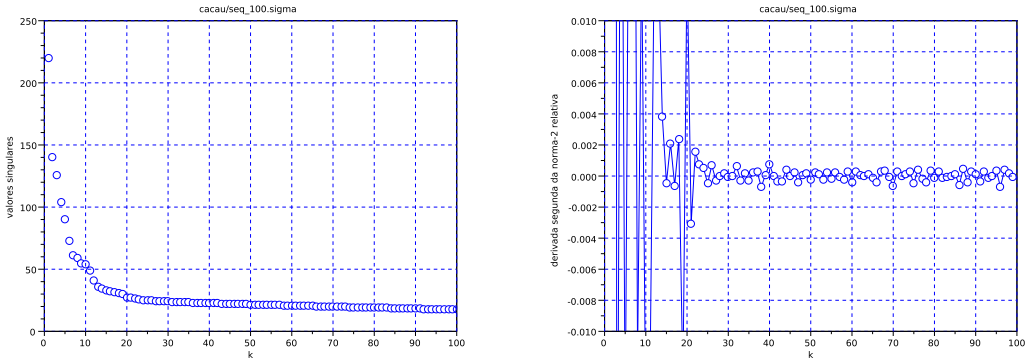


Figura 7.10: SVD - cortes escolhidos 10, 20 e 30.

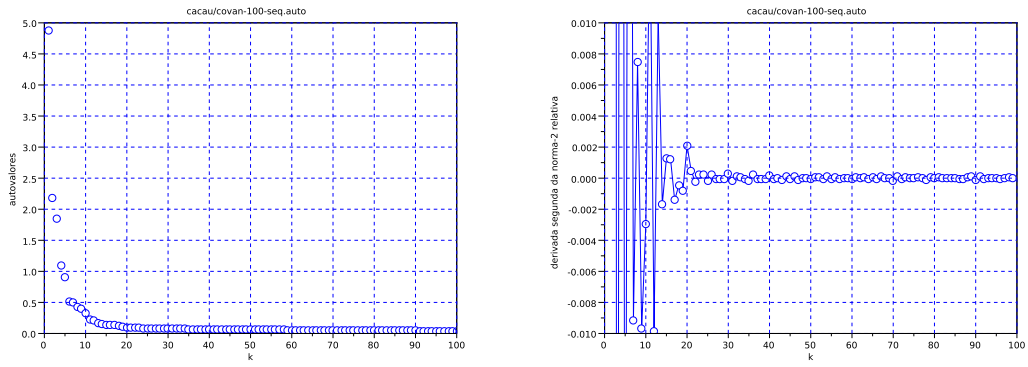


Figura 7.11: PCA via matriz de covariância - cortes escolhidos 5, 15 e 25.

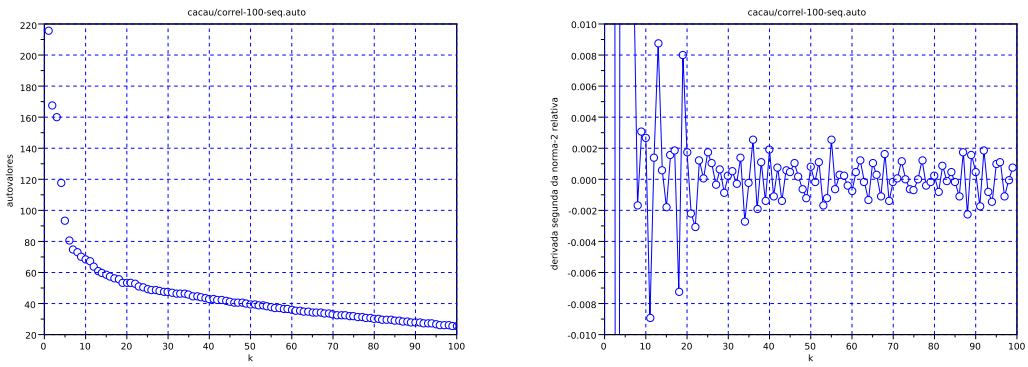


Figura 7.12: PCA via matriz de correlação - cortes escolhidos 10, 20 e 30.

7.1.3 Análise geométrica das técnicas

A fim de verificar o impacto que cada técnica gera na distribuição dos dados devido à redução da dimensionalidade, foram gerados gráficos para as três bases reais, contendo três dimensões tanto para a técnica SVD quanto para a PCA utilizando a matriz de covariância e a de correlação. É possível observar um comportamento mais similar entre a SVD e a PCA (covariância) para as três bases de dados. Isso, por sua vez, pode ser explicado pelos baixos valores das médias apresentados nas distribuições de frequência de média dos atributos para cada base de dados. Por outro lado, devido às transformações lineares ortogonais envolvidas, essa diferença apresenta um aspecto de rotacionado no espaço tridimensional, enquanto o comportamento da PCA (correlação) parace, além disso, possuir um aspecto mais disperso devido aos desvios padrões. As primeiras visões geométricas apresentadas são referentes à base *Sequence* composta por 8 classes distintas, que podem ser observadas por meio das Figuras 7.13 e 7.14.

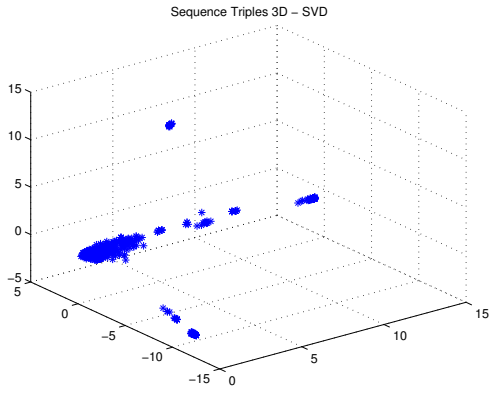


Figura 7.13: Sequence 3D - SVD

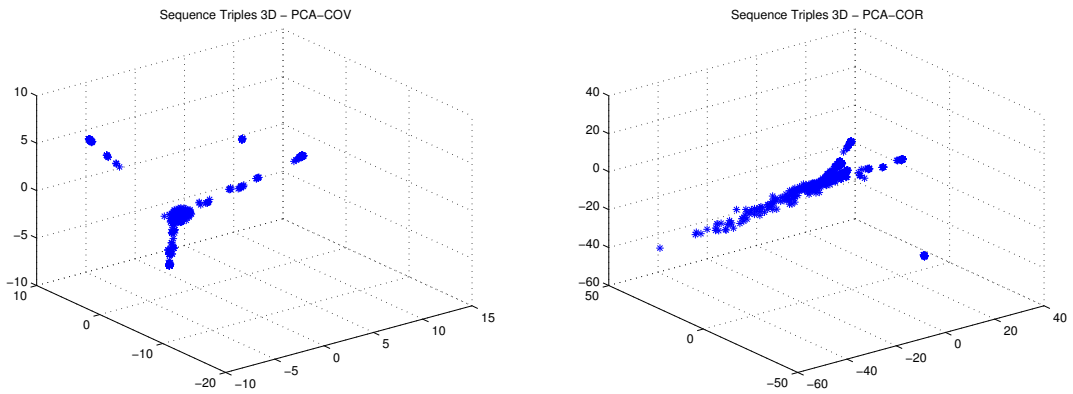


Figura 7.14: Sequence 3D - PCA-COV e PCA-COR

A coleção *Nature*, composta por cinco classes distintas, apresentou um aspecto mais compacto em todas as visualizações. Essa compactação dos dados, inclusive, pode ser explicada devido à alta dimensionalidade (104288) que essa base apresenta, dificultando uma análise geométrica em relação à quantidade de classes existentes.

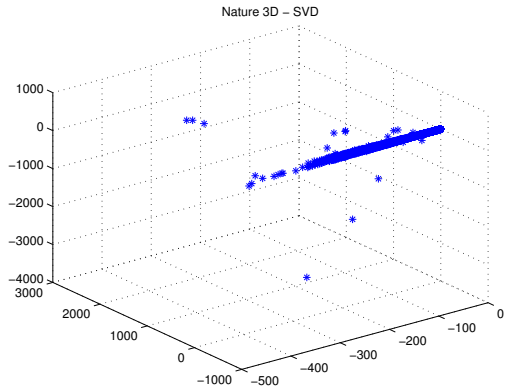


Figura 7.15: Nature 3D - SVD

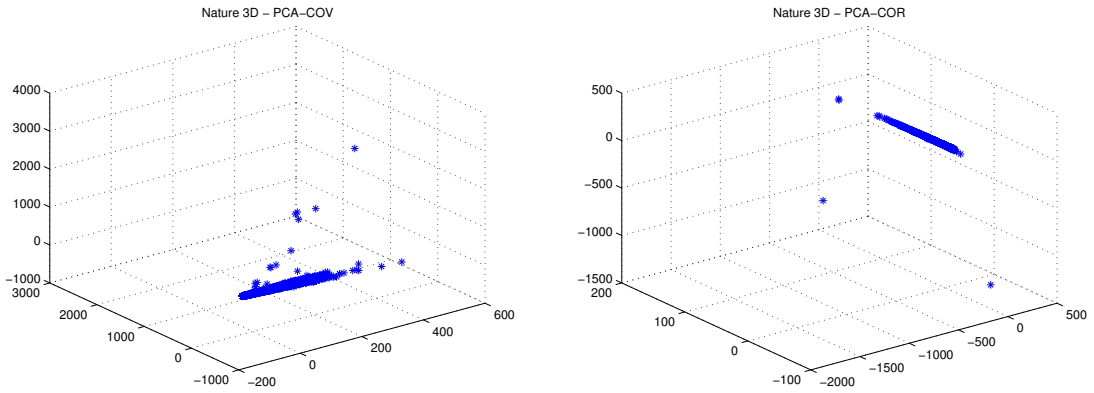


Figura 7.16: Nature 3D - PCA-COV e PCA-COR

Por outro lado, a coleção *Reuters*, composta por oito classes distintas, apresentou um aspecto mais disperso em todas as visualizações, também devido à alta dimensionalidade (24985) que apresenta. No entanto, assim como nas duas outras bases, os baixos valores apontados na distribuição das médias dos atributos, justificam, principalmente, a similaridade do aspecto entre as técnicas SVD e da PCA via matriz de covariância, a qual pode ser observada pelas Figuras 7.17 e 7.18.

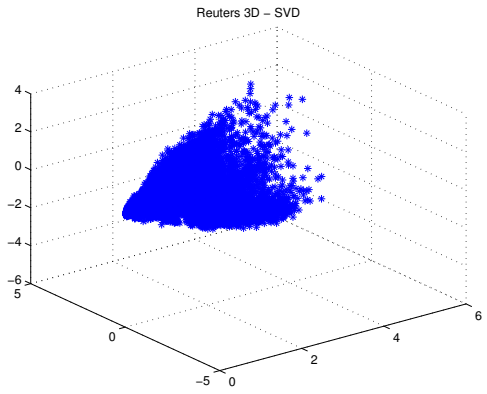


Figura 7.17: Reuters 3D - SVD

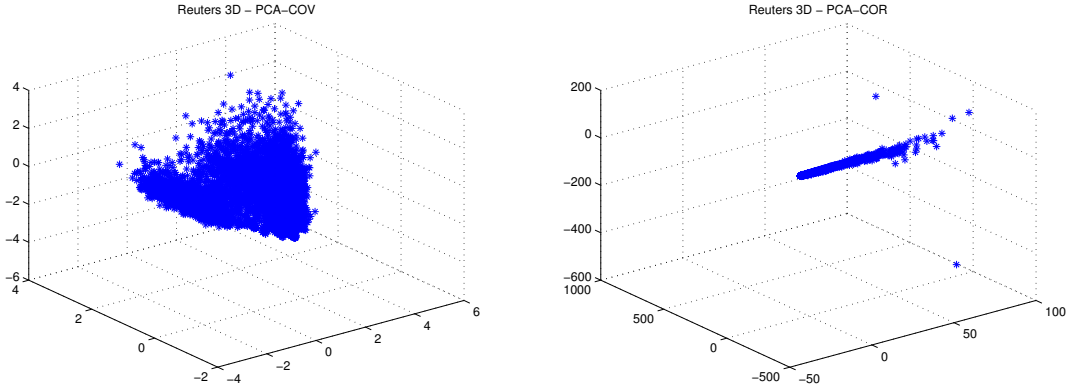


Figura 7.18: Reuters 3D - PCA-COV e PCA-COR

7.1.4 Análise de distribuição das distâncias entre as entidades

A fim de verificar o impacto que cada técnica gera na distribuição das distâncias de similaridade devido à redução da dimensionalidade, foram gerados histogramas para os cortes referentes ao meio do intervalo escolhido. Por meio das distribuições resultantes da base *Sequence* apresentadas pelas Figura 7.19, percebe-se que a redução de dimensionalidade, preserva, de forma geral, o aspecto do comportamento da distribuição das distâncias de cosseno entre entidades de cada base de dados. Do mesmo modo, o comportamento da distribuição de distâncias de cosseno para as bases *Nature* e *Reuters* são preservadas após a redução de dimensionalidade, principalmente, para a redução realizada pela SVD (Figuras 7.20 e 7.21). No entanto, é possível verificar pequenas distorções nas distribuições da PCA em relação à distribuição original, o que talvez possa influenciar na classificação dos dados pelo KNN. Outra hipótese a se pensar é: distribuições mais similares à original proporcionam melhor eficácia da classificação?

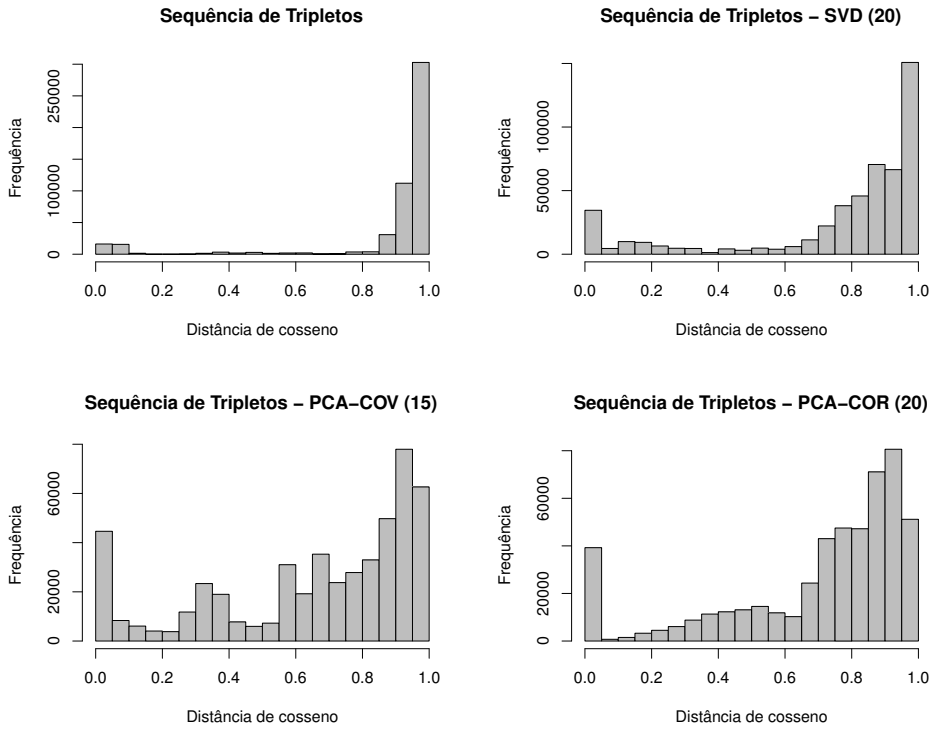


Figura 7.19: Histogramas: Base original, SVD (20), PCA-COV(15) e PCA-COR(20)

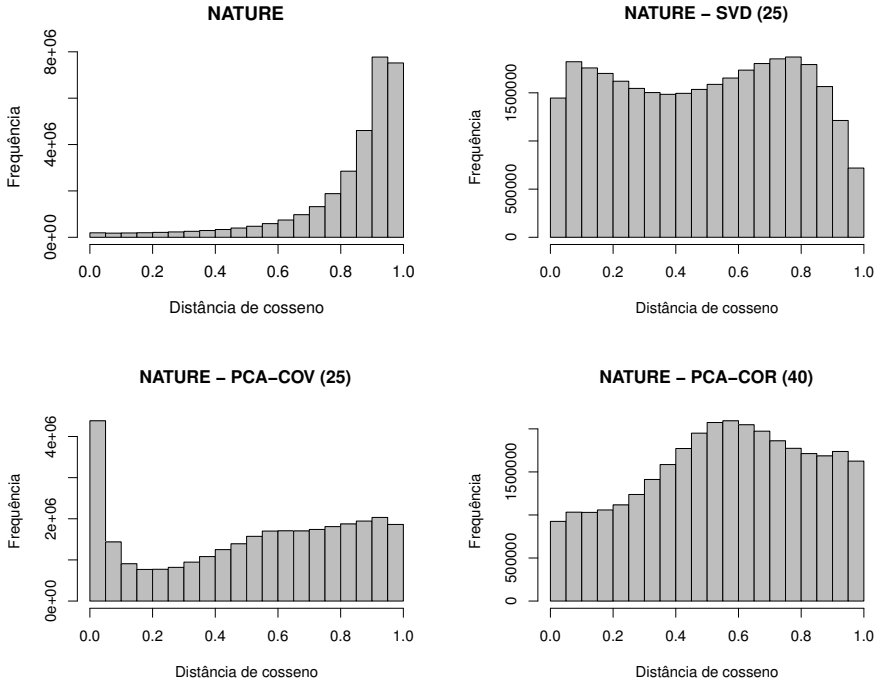


Figura 7.20: Histogramas: Base original, SVD (25), PCA-COV(25) e PCA-COR(40)

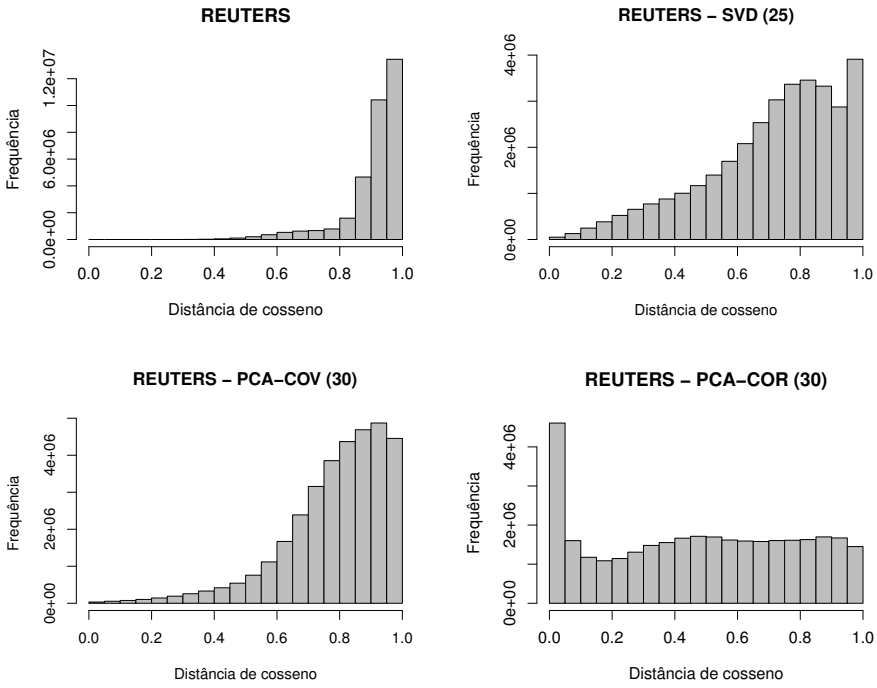


Figura 7.21: Histogramas: Base original, SVD (25), PCA-COV(30) e PCA-COR(30)

7.2 Tempo de execução e eficácia da classificação dos dados originais via aplicação da SVD

A primeira etapa dos experimentos, realizados de forma exclusiva numa máquina de 8GB de RAM com processador Intel(R) Core(TM) i5 CPU 2.80GHz, busca avaliar tanto a eficácia quanto o desempenho da classificação dos dados originais. Para tanto, foi utilizado 30 como principal valor do parâmetro k do algoritmo KNN, como é usualmente indicado na literatura e 1, para fins de comparação, na busca de desempenho.

É sabido que a escolha do parâmetro k pode influenciar tanto no ganho da eficácia quanto no ganho do desempenho da classificação. No entanto, é um desafio encontrar o melhor valor a ser escolhido, pois o mesmo pode estar associado a fatores complexos e imprecisos sobre a base de dados em questão. Entretanto, trabalhos bem estabelecidos da área de Mineração de Dados (MD) apontam como potenciais valores os que se encontram no intervalo de 30 a 50. Por isso, o valor 30 será utilizado como base de comparação visando verificar a hipótese de que exista uma relação entre o parâmetro k e a redução de dimensionalidade realizada sobre os dados originais. A Tabela 7.1 apresenta a eficácia, a F1 e o desempenho do uso do KNN sobre os dados originais.

Tabela 7.1: Resultados da classificação dos dados originais.

Coleção	k	Eficácia (%)	F1 (%)	Tempo (hor : min : seg)
Reuters	1	88.28	88.00	00 : 15 : 19
	30	88.28	88.00	00 : 18 : 28
Nature	1	72.65	68.50	04 : 35 : 59
	30	77.12	67.90	05 : 00 : 09
Sequence	1	92.12	91.90	00 : 00 : 41
	30	97.51	92.50	00 : 01 : 07

É possível notar a partir dos valores da Tabela 7.1 que a escolha do parâmetro k pode favorecer, como por exemplo, no caso das bases *Nature* e *Sequence*, as quais tiveram 5% de aumento da eficácia em função da escolha do parâmetro k ou, tornar-se indiferente, como no caso da base *Reuters*. Por sua vez, a Tabela 7.2 apresenta os resultados da classificação sobre os dados transformados por meio da SVD. Vale ressaltar que os tempos fornecidos são tempos médios de 5 execuções, uma vez que, como a máquina foi utilizada de forma exclusiva, quase não houve variação na medição. Além disso, com intuito de verificar relações entre o ponto de transição e o parâmetro k, foram realizados alguns experimentos que apontam uma tendência de estabilidade da eficácia e da F1 para valores de k iguais ou maiores que o corte de transição. Essa tendência reforça a hipótese de que existe uma relação entre o valor de corte para redução da dimensionalidade e o valor do número de vizinhos a serem considerados pelo KNN.

Tabela 7.2: Resultados da classificação via SVD (tempo dado em min : seg).

(Tempo de corte: tempo de cálculo dos 100 maiores valores singulares.)

(Corte: prematuro, transição (em negrito) e estável.)

Coleção	Tempo de corte	Corte	Tempo da SVD	Tempo do KNN		Eficácia KNN (%)	
				k = Corte	$k = 30$	k = Corte	$k = 30$
Reuters	02 : 05	10	00 : 01	00 : 57	01 : 06	90.91	90.90
		25	00 : 02	02 : 08	02 : 12	**92.00**	**92.00**
		40	00 : 05	03 : 38	03 : 27	92.14	92.14
Nature	11 : 50	15	00 : 21	00 : 42	00 : 48	86.99	86.97
		25	01 : 27	00 : 55	00 : 57	**87.36**	**87.36**
		35	01 : 48	01 : 12	01 : 09	87.39	87.39
Sequence	00 : 29	10	00 : 01	00 : 01	00 : 02	97.11	97.21
		20	00 : 01	00 : 02	00 : 02	**97.61**	**97.61**
		30	00 : 01	00 : 03	00 : 03	97.21	97.21

Aparentemente, o ponto de transição, ponto de início de um suposto regime linear, se mostra como ponto mínimo de estabilidade da eficácia obtida, ao considerar como valor do parâmetro k, o mesmo valor ou valor maior ao escolhido para posto na aproximação de posto incompleto. Essa tendência pode ser melhor observada pela Tabela 7.3, que apresenta os resultados da eficácia e da F1 para diferentes valores de k considerando uma redução fixa no ponto de transição. É possível observar duas relações interessantes.

A primeira delas é que o número mínimo do k (KNN) considerando o corte de transição realizado, que garante, inclusive, uma suposta estabilidade da eficácia, inicia-se, em geral, no próprio valor do corte. A outra observação interessante é que após realizar a redução de dimensionalidade, o valor do parâmetro k igual a 1 ou 5, resulta em valores de eficácia e F1 muito próximos dos valores obtidos por meio do corte de transição. Com isso, esses valores, 1 e 5, mostram-se melhores opções em casos onde o desempenho é primordial, como por exemplo, em cenários de classificação em tempo real.

Tabela 7.3: Desempenho e eficácia da classificação variando o parâmetro k.

Coleção	Corte de transição	k do (KNN)	Tempo do KNN	Eficácia (%)	F1 (%)
Reuters	25	1	01 : 18	91.14	91.20
		5	01 : 40	92.02	91.90
		40	02 : 20	**92.00**	**91.90**
		45	02 : 23	**92.00**	**91.90**
		50	02 : 27	**92.00**	**91.90**
Nature	25	1	00 : 35	84.93	84.90
		5	00 : 42	87.02	86.60
		40	01 : 02	**87.36**	**86.80**
		45	01 : 05	**87.36**	**86.80**
		50	01 : 07	**87.36**	**86.80**
Sequence	20	1	00 : 02	98.00	98.00
		5	00 : 02	97.61	97.70
		40	00 : 03	**97.61**	**97.60**
		45	00 : 03	**97.61**	**97.60**
		50	00 : 04	**97.61**	**97.60**

Do mesmo modo, como foi realizado para a técnica SVD, foram medidos os tempos de execução, de corte, de processamento da técnica e das métricas de qualidade eficácia e F1 do KNN frente à redução de dimensionalidade por meio da PCA via matriz de covariância (Seção 7.3) e correlação (Seção 7.4), nas quais podem ser observadas as mesmas características tendenciosas entre o parâmetro k e o corte de transição.

7.3 Tempo de execução e eficácia da classificação via PCA utilizando matriz de covariância

Tabela 7.4: Resultado da classificação via PCA (tempo dado em min : seg).

(Tempo de corte: tempo de cálculo dos 100 maiores autovalores da matriz de covariância.)

(Corte: prematuro, transição (em negrito) e estável.)

Coleção	Tempo de corte	Corte	Tempo da PCA	Tempo do KNN		Eficácia KNN (%)	
				k = Corte	$k = 30$	k = Corte	$k = 30$
Reuters	11 : 35	10	01 : 26	00 : 56	01 : 05	91.21	91.29
		30	02 : 43	02 : 43	02 : 43	**92.07**	**92.07**
		50	04 : 27	04 : 27	03 : 59	92.52	92.52
Nature	59 : 08	15	11 : 25	00 : 48	00 : 55	86.56	86.41
		25	31 : 47	01 : 05	01 : 07	**86.80**	**86.80**
		35	32 : 58	01 : 17	01 : 20	87.42	87.42
Sequence	00 : 40	5	00 : 01	00 : 01	00 : 02	95.21	94.92
		15	00 : 03	00 : 02	00 : 02	**98.01**	**98.01**
		25	00 : 06	00 : 03	00 : 03	97.71	97.71

Tabela 7.5: Desempenho e eficácia da classificação variando o parâmetro k

Coleção	Corte de transição	k do (KNN)	Tempo do KNN	Eficácia (%)	F1 (%)
Reuters	30	1	01 : 33	91.54	91.60
		5	02 : 03	91.89	91.80
		35	02 : 47	**92.07**	**92.00**
		40	02 : 50	**92.07**	**92.00**
		45	02 : 53	**92.07**	**92.00**
Nature	25	1	00 : 40	84.37	84.40
		5	00 : 49	87.03	86.60
		40	01 : 14	**86.80**	**86.30**
		45	01 : 16	**86.80**	**86.30**
		50	01 : 19	**86.80**	**86.30**
Sequence	15	1	00 : 01	98.01	98.00
		5	00 : 02	98.01	98.00
		20	00 : 03	**98.01**	**98.00**
		35	00 : 03	**98.01**	**98.00**
		50	00 : 03	**98.01**	**98.00**

7.4 Tempo de execução e eficácia da classificação via PCA utilizando matriz de correlação

Tabela 7.6: Resultado da classificação via PCA (tempo dado em hor: min : seg).

(Tempo de corte: tempo de cálculo dos 100 maiores autovalores da matriz de correlação.)

(Corte: prematuro, transição (em negrito) e estável.)

Coleção	Tempo de corte	Corte	Tempo da PCA	KNN (min : seg)		Eficácia KNN (%)	
				K = Corte	K = 30	K = Corte	K = 30
Reuters	00 : 24 : 48	10	00 : 09 : 06	00 : 33	00 : 38	78.48	78.73
		30	00 : 09 : 38	01 : 04	01 : 04	**83.98**	**83.98**
		50	00 : 10 : 12	01 : 57	01 : 42	85.08	85.08
Nature	02 : 25 : 03	20	00 : 22 : 11	00 : 51	00 : 55	87.37	87.37
		40	00 : 30 : 48	01 : 23	01 : 16	**89.23**	**89.23**
		60	01 : 11 : 06	02 : 21	01 : 55	90.12	90.12
Sequence	00 : 00 : 47	10	00 : 00 : 03	00 : 01	00 : 02	97.61	97.61
		20	00 : 00 : 09	00 : 02	00 : 02	**97.91**	**97.91**
		30	00 : 00 : 12	00 : 03	00 : 03	98.21	98.21

Tabela 7.7: Desempenho e eficácia da classificação variando o parâmetro k.

Coleção	Corte de transição	k do (KNN)	Tempo do KNN	Eficácia (%)	F1 (%)
Reuters	30	1	00 : 42	82.57	82.60
		5	00 : 49	83.85	83.80
		35	01 : 07	**83.98**	**83.90**
		40	01 : 09	**83.98**	**83.90**
		45	01 : 11	**83.98**	**83.90**
Nature	40	1	00 : 51	86.25	86.20
		5	00 : 59	89.20	88.70
		35	01 : 21	**89.23**	**88.60**
		45	01 : 27	**89.23**	**88.60**
		50	01 : 30	**89.23**	**88.60**
Sequence	20	1	00 : 02	98.21	98.20
		5	00 : 02	97,91	97.90
		25	00 : 03	**97,91**	**97.90**
		35	00 : 03	**97,91**	**97.90**
		45	00 : 03	**97,91**	**97.90**

7.5 Análise de desempenho sobre as bases reais

Para facilitar a análise somente do desempenho entre as técnicas, os tempos de corte das bases reais para o valor 100 foram sumarizados na Tabela 7.8 e os tempos de pré-processamento das bases de dados por meio das técnicas, para o corte simultâneo igual a 10, realizado apenas para fins de comparação, podem ser observados na Tabela 7.9.

Tabela 7.8: Desempenho do cálculo dos 100 maiores valores singulares ou autovalores.

(Ta: tempo absoluto de processamento de cada técnica (hor : min : seg).)
(Tr: tempo relativo em relação à técnica SVD.)

	SVD		PCA-COR		PCA-COR	
Coleção	Ta	Tr	Ta	Tr	Ta	Tr
Reuters	**00 : 02 : 05**	1	00 : 11 : 35	5.56	00 : 24 : 48	11.90
Nature	**00 : 11 : 50**	1	00 : 59 : 08	5.00	02 : 25 : 03	12.26
Sequence	**00 : 00 : 29**	1	00 : 00 : 40	1.38	00 : 00 : 47	1.62

Tabela 7.9: Desempenho do processamento das técnicas para corte igual a dez (10).

(Ta: tempo absoluto de processamento de cada técnica (hor : min : seg).)
(Tr: tempo relativo em relação à técnica SVD.)

	SVD		PCA-COR		PCA-COR	
Coleção	Ta	Tr	Ta	Tr	Ta	Tr
Reuters	**00 : 01**	1	01 : 26	86	09 : 06	546
Nature	**00 : 09**	1	05 : 29	36.56	14 : 45	98.33
Sequence	**00 : 01**	1	00 : 02	2	00 : 03	3

Pelos resultados obtidos, percebe-se que quanto maior for o número de atributos das entidades (colunas), menor o desempenho das técnicas PCA-COV e PCA-COR frente à técnica SVD. Dessa forma, salvo em cenários onde a eficácia é o elemento crucial, aplicar a PCA para grandes matrizes, mesmo em matrizes esparsas com otimização da manipulação dos dados de forma, inclusive, a manter sempre a esparsidade das mesmas, não parece ser um procedimento interessante. Além do mais, as médias dos atributos parecem possuir uma tendência a decrescer com o aumento do número de atributos, resultando no final, em resultados muito próximos aos fornecidos pela SVD.

7.6 Análise de eficácia sobre as bases reais

Para facilitar a análise somente da eficácia entre as técnicas, os valores das métricas de qualidade eficácia e F1 sobre a tarefa de classificação dos dados considerando o corte de transição do intervalo escolhido para cada base, foram sumarizadas na Tabela 7.10. É possível observar que os melhores resultados foram obtidos pela aplicação da PCA (covariância ou correlação) para as três bases utilizadas. No entanto, os valores para a SVD ficaram bem próximos dos melhores valores da PCA. Dessa forma, a escolha entre as técnicas vai depender do quanto é possível abrir mão de certa precisão em função do desempenho obtido. O cenário em que é mais notório a escolha baseada no desempenho é o de classificação em tempo real, onde é aceitável aplicar técnicas um pouco menos precisas em função do ganho do desempenho. Por outro lado, em cenários em que o tempo não é tão importante e sim a precisão do resultado, a PCA-COV ou PCA-COR pode ser mais interresante, como mostram os principais resultados obtidos.

Tabela 7.10: Comparação da classificação entre as técnicas para o corte de transição.

Coleção	Técnica	Corte de transição	Eficácia (%)	F1 (%)
Reuters	SVD	25	92.00	91.90
	PCA-COV	30	**92.07**	**92.00**
	PCA-COR	30	83.89	83.90
Nature	SVD	25	87.36	86.80
	PCA-COV	25	86.80	86.30
	PCA-COR	40	**89.23**	**88.60**
Sequence	SVD	20	97.61	97.60
	PCA-COV	15	**98.01**	**98.00**
	PCA-COR	20	97.91	97.90

Capítulo 8

Conclusões e trabalhos futuros

Nesse trabalho foram descritas diferenças teóricas importantes sobre as técnicas Decomposição em Valores Singulares (SVD) e Análise de Componentes Principais (PCA) e obtidos resultados interessantes referentes à comparação da aplicação dessas técnicas sobre a tarefa de classificação de dados. Para tanto, após o primeiro capítulo de introdução do assunto, o segundo capítulo apresenta um profundo estudo teórico da técnica Decomposição espectral e da sua forma de cálculo, tanto de forma direta, usualmente utilizada para matrizes densas, quanto de forma iterativa, usualmente utilizada para matrizes esparsas de alta ordem. Por sua vez, o terceiro e o quarto capítulos apresentam um estudo teórico dos dois tipos de decomposição espectral (SVD e PCA) e das ferramentas matemáticas e estatísticas envolvidas no contexto de cada uma delas, bem como das principais aplicações às quais são submetidas. Esse estudo foi necessário para entender quais são as premissas impostas por essas duas diferentes decomposições.

Além disso, dado que o foco do trabalho esteve relacionado com dados representados por matrizes esparsas de alta ordem, o quinto capítulo apresenta o estudo de diferentes tipos de formatos para armazenamento e manipulação dessas matrizes, o qual levou à escolha da estrutura esparsa designada por Harwell-Boeing (HB) para realização dos experimentos. Nesse momento foram desenvolvidos vários algoritmos de multiplicação matriz-vetor envolvendo esse formato com intuito de avaliar o ganho da complexidade em termos de processamento e memória. Tudo isso foi de extrema importância, pois além de ter contruibuído para a escolha da estrutura a ser utilizada, também possibilitou vislumbrar mecanismos eficientes para a manipulação algébrica das operações impostas pela técnica PCA, frente à esparsidade contida nas matrizes envolvidas.

Consequentemente, o sexto capítulo do trabalho identifica a existência dos principais programas disponíveis para o cálculo da SVD e da PCA e destaca a implementação do código fonte na linguagem de programação FORTRAN que, por sua vez, utilizou a biblioteca numérica ARPACK. Essa biblioteca foi escolhida por apresentar melhor desempenho frente ao SCILAB, MATLAB e R e por viabilizar o cálculo da PCA para matrizes grandes. Além do mais, as comparações feitas entre os resultados fornecidos pelo código fonte desenvolvido e os resultados fornecidos por esses programas bem estabelecidos, permitiram validar os resultados alcançados.

Por fim, os conteúdos desses capítulos foram necessários para a consolidação do sétimo capítulo do trabalho, que apresenta a metodologia desenvolvida de suporte à escolha entre SVD e PCA sobre a tarefa de classificação de dados em matrizes esparsas de alta ordem. Essa metodologia, dividida em quatro etapas distintas e desenvolvida por meio da realização de três estudos de caso, levou a algumas observações interessantes. Por exemplo, embora às técnicas SVD e PCA já sejam usadas no contexto de Mineração de Dados, foram verificadas algumas hipóteses pertinentes levantadas quanto a esse uso.

A primeira hipótese verificada é que existe uma relação entre o parâmetro k do algoritmo KNN e o posto escolhido para a aproximação de posto incompleto durante a realização da redução de dimensionalidade. Não obstante, apesar de não ter sido devidamente validada, muitos experimentos foram realizados com intuito de esclarecer a tendência de estabilidade da eficácia e da F1 retornadas pelo algoritmo frente aos cortes de transição dos intervalos sugeridos em uma das etapas da metodologia.

Outra hipótese verificada é que a redução de dimensionalidade realizada por meio da SVD e PCA, por preservar de modo geral o aspecto das distribuições de distâncias entre as entidades, retornou resultados no mínimo semelhantes à classificação sobre os dados originais com enorme ganho do desempenho. Isso, provavelmente, se deve ao fato do algoritmo KNN trabalhar com distâncias de similaridade entre os k vizinhos mais próximos durante a definição das classes. Dentre os resultados, o que mais se destacou foi a classificação da coleção *Nature*, que teve sua eficácia melhorada de 77% para 87%, além do enorme ganho em seu desempenho. Por fim, é possível concluir também que em se tratando de matrizes esparsas de alta ordem, em cenários, como por exemplo, classificação em tempo real, a SVD é muito superior. Por outro lado, em se tratando apenas de eficácia, a PCA se mostrou ligeiramente melhor em todas as coleções. Entretanto, por retornar eficácia competitiva com melhor desempenho, a SVD se mostrou a mais indicada para processar bases esparsas de alta ordem .

8.1 Limitações

Apesar de duas outras bases de dados, a *Medline* e *Agnews*, terem sido processadas pelo ARPACK, sendo bastante reduzidas pelas técnicas SVD e PCA, não foi possível verificar o ganho do desempenho e da eficácia da classificação desses dados frente à redução de dimensionalidade, devido às limitações do *software* WEKA. Essa impossibilidade se ateve ao fato dessas bases apresentarem, aproximadamente, magnitudes da ordem de 800.000 por 200.000 elementos (estado original) e 800.000 por 10, 20 ou 30 (estado reduzido) e devido à falta de recurso (alocação de máquina apropriada) para processá-las por vários dias consecutivos. Além disso, essas bases em estado original demoram em média cerca de 52 dias para serem classificadas por *softwares* mais robustos.

8.2 Trabalhos Futuros

Durante a realização do trabalho foram visualizadas novas possibilidades para aplicação das técnicas juntamente com a metodologia desenvolvida. Por exemplo, uma das possibilidades observadas consiste em aplicar essas técnicas de redução de dimensionalidade e ruído dos dados frente a outros cenários. Um passo importante é o de associar o impacto que as técnicas possuem frente ao desempenho do Naive Bayes, do SVM, entre outros importantes algoritmos da área de Mineração de Dados, com intuito não só de fortalecer o procedimento, mas também de generalizar a metodologia.

Outra importante vertente que esse trabalho introduziu é a necessidade da validação da hipótese parcialmente verificada por meio dos experimentos. Essa hipótese sugere que independente do cenário, o valor do parâmetro k (número de vizinhos) do tradicional algoritmo de classificação KNN, possui uma íntima relação com a escolha de, no mínimo, dois dos pontos sugeridos pela metodologia como sendo o posto "ideal" para aproximação de posto incompleto. Dentre esses pontos se encontram o ponto de transição, que representa a mudança entre as duas regiões distintas apontadas pela distribuição dos valores singulares e autovalores e, o ponto dito mais estável, ponto pertencente ao suposto regime linear comumente contido nesse tipo de distribuição.

Apêndice A

Fundamentos de Estatística

A.1 Conceitos fundamentais

Definição A.1 (Vetor aleatório [24] - pag 27) *Seja X um vetor contendo p componentes, onde cada componente é uma variável aleatória, isto é, X_i é uma variável aleatória, $\forall i = 1, 2, .., p$. Então, X é chamado de vetor aleatório e é denotado por:*

$$X = \begin{bmatrix} X_1 \\ X_2 \\ \vdots \\ X_p \end{bmatrix}.$$

Definição A.2 (Vetor de médias [24] - pag 28) *Seja X um vetor aleatório. O vetor $\mu = E(X)$ é chamado de vetor de médias do vetor $X = (X_1, X_2, ..., X_p)$ sendo:*

$$\mu = E(X) = \begin{bmatrix} \mu_1 \\ \mu_2 \\ \vdots \\ \mu_p \end{bmatrix}.$$

onde $\mu_i = E(X_i)$ denota a média ou esperança, da variável aleatória X_i, $i = 1, 2, ..., p$. A média é uma das medidas mais utilizadas para sintetizar a informação de tendência central da distribuição de valores da variável X_i.

Definição A.3 (Vetor de médias amostrais [24] - pag 38) *O vetor de médias amostrais $\overline{X}$ é definido por:*

$$\overline{X} = \frac{1}{n}[X_1 + X_2 + ...X_n] = \begin{bmatrix} \overline{X}_1 \\ \overline{X}_2 \\ \vdots \\ \overline{X}_p \end{bmatrix},$$

onde $\overline{X}_i$ é a média amostral da i-ésima variável, $i = 1, 2, ..., p$.

Definição A.4 (Variância [24] - pag 28) *A variância do i-ésimo componente do vetor X é denotada por $Var(X_i) = \sigma_i^2 = \sigma_{ii}$. O desvio padrão é denotado por σ_i ou $\sqrt{\sigma_{ii}}$ fornece a informação sobre a disposição dos valores da variável X_i em relação a μ_i, isto é, indica se os valores de X_i estão próximos ou distantes da média μ_i. Dessa forma, valores grandes de σ_i indicam uma maior dispersão de valores em relação à média da distribuição.*

Definição A.5 (Covariância [24] - pag 28) *A covariância entre os valores da i-ésima e j-ésima variáveis do vetor X é definida por:*

$$Cov(X_i, X_j) = \sigma_{ij} = E[(X_i - \mu_i)(X_j - \mu_j)].$$

Quando $i = j$ a expressão torna-se a variância da variável.

Definição A.6 (Variância total [24] - pag 32) *A variância total do vetor aleatório X é definida como: traço($\Sigma_{p\times p}$) $= \sigma_{11} + \sigma_{22} + ... + \sigma_{pp}$. O traço é uma forma de sintetização da variância global da distribuição multivariada, uma vez que esta é a soma das variâncias de todas as variáveis envolvidas no vetor X. Altos valores de variâncias totais indicam uma maior dispersão global das variáveis X_i, $i = 1, 2, ..., p$.*

Definição A.7 (Variância generalizada [24] - pag 32) *A variância generalizada do vetor aleatório X é definida como o determinante da matriz $\Sigma_{p\times p}$, isto é, $|\Sigma_{p\times p}|$. O desvio padrão generalizado de X é definido como $\sqrt{|\Sigma_{p\times p}|}$. A variância generalizada fornece uma noção de dispersão multivariada no espaço, sendo que distribuições com maiores variabilidades globais apresentam maiores valores de variâncias generalizadas. Ao contrário da variância total, a variância generalizada é influenciada pelas covariâncias (ou correlações) entre as variáveis.*

Referências Bibliográficas

[1] Alter, O.; Brown, P. O. & Botstein, D. (2001). Processing and modeling genome-wide expression data using singular value decomposition (http://www.ncbi.nlm.nih.gov/pubmed/10963673).

[2] Anton, H. & Rorres (2001). *Álgebra Linear com aplicações*. Porto Alegre: Bookman, 8 edição.

[3] Arnoldi, W. E. (1951). The principle of minimized iterations in the solution of the matrix eigenvalue problem. *Quarterly of Applied Mathematics*, 9:17–29.

[4] Berry, M. W.; Dumais, S. T. & O'Brien, G. W. (1995). Using linear algebra for intelligent information retrieval. *SIAM Rev.*, 37(4):573--595.

[5] Cattell, R. B. (1966). The Scree Test For The Number Of Factors. *Multivariate Behavioral Research*, 1(2):245--276.

[6] Demmel, J.; Dongarra, J.; Ruhe, A. & van der Vorst, H. (2000). *Templates for the solution of algebraic eigenvalue problems: a practical guide*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA.

[7] Demmel, J. W. (1997). *Applied Numerical Linear Algebra*. SIAM, 1 edição.

[8] Duff, I. S.; Grimes, R. G. & Lewis, J. G. (1992). Users' guide for the harwell-boeing sparse matrix collection (release i).

[9] Eckart, C. & Young, G. (1936). The approximation of one matrix by another of lower rank. *Psychometrika*, 1(3):211--218.

[10] Elden, L. (2006). Numerical linear algebra in data mining. *Acta Numerica*, 15, 327-384.

[11] Eldén, L. (2007). *Matrix Methods in Data Mining and Pattern Recognition*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA.

[12] Figueiredo, F.; Rocha, L.; Couto, T.; Salles, T.; Gonçalves, M. A. & Jr., W. M. (2011). Word co-occurrence features for text classification. *Information Systems*, 36(5):843 – 858.

[13] Golub, G. & Kahan, W. (1965). Calculating the Singular Values and Pseudo-Inverse of a Matrix. *Journal of the Society for Industrial and Applied Mathematics: Series B, Numerical Analysis*, 2(2):205–224.

[14] Golub, G. & Reinsch, C. (1970). Singular value decomposition and least squares solutions. *Numerische Mathematik*, 14(5):403–420.

[15] Golub, G. H. & Van Loan, C. F. (1996). *Matrix computations (3rd ed.)*. Johns Hopkins University Press, Baltimore, MD, USA.

[16] Hamilton, P. B. (2006). *Álgebra Linear Um segundo curso*. Rio de Janeiro: SBM, 1 edição.

[17] Hestenes, M. R. (1958). Inversion of Matrices by Biorthogonalization and Related Results. *Journal of the Society for Industrial and Applied Mathematics*, 1(6):51–90.

[18] Horn, D. & Axel, I. (2003). Novel clustering algorithm for microarray expression data in a truncated svd. *Bioinformatics*, pp. 1110--1115.

[19] Hotelling, H. (1933). Analysis of a Complex of Statistical Variables Into Principal Components. *Journal of Educational Psychology*, 24:417–441 and 498–520.

[20] Kaiser, H. F. (1958). The varimax criterion for analytic rotation in factor analysis. *Psychometrika*, 23(3):187–200.

[21] Koren, Y. (2008). Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *KDD '08: Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 426--434, New York, NY, USA. ACM.

[22] Koren, Y.; Bell, R. & Volinsky, C. (2009). Matrix factorization techniques for recommender systems. *Computer*, 42(8):30--37.

[23] Lanczos, C. (1950). An Iteration Method for the Solution of the Eigenvalue Problem of Linear Differential and Integral Operators. *Journal of research of the National Bureau of Standards*, pp. 255–282.

[24] Mingoti, S. A. (2007). *Análise de dados através de métodos de Estatística Multivariada - uma abordagem aplicada*. Editora UFMG.

[25] Muller, N.; Magaia, L. & Herbst, B. M. (2004). Singular Value Decomposition, Eigenfaces, and 3D Reconstructions. *SIAM Review*, 46(3):518--545.

[26] Pearson, K. (1901). On Lines and Planes of Closest Fit to Systems of Points in Space. *Philosophical Magazine*, 2(6):559–572.

[27] Pires, D. (2011). personal communication.

[28] Price, A. L.; Patterson, N. J.; Plenge, R. M.; Weinblatt, M. E.; Shadick, N. A. & Reich, D. (2006). Principal components analysis corrects for stratification in genome-wide association studies. *NATURE GENETICS*, 38(8):904–909.

[29] Rocha, L.; Mourão, F.; Pereira, A.; Gonçalves, M. A. & Meira, Jr., W. (2008). Exploiting temporal contexts in text classification. In *Proceeding of the 17th ACM conference on Information and knowledge management*, CIKM '08, pp. 243--252, New York, NY, USA. ACM.

[30] Stegmann, M. B. (2002). Analysis and segmentation of face images using point annotations and linear subspace techniques. Technical report, Informatics and Mathematical Modelling, Technical University of Denmark, DTU, Richard Petersens Plads, Building 321, DK-2800 Kgs. Lyngby.

[31] Steinbach, M.; Ertöz, L. & Kumar, V. (2003). The challenges of clustering high-dimensional data. In *In New Vistas in Statistical Physics: Applications in Econophysics, Bioinformatics, and Pattern Recognition*. Springer-Verlag.

[32] Trefethen, L. N. & Bau, D. (1997). *Numerical Linear Algebra*. SIAM: Society for Industrial and Applied Mathematics.

[33] Wall, M. E.; Rechtsteiner, A. & Rocha, L. M. (2003). *Singular Value Decomposition and Principal Component Analysis*, chapter 5, pp. 91--109. Kluwel.

[34] Yeung, K. Y. & Ruzzo, W. L. (2001). An empirical study on principal component analysis for clustering gene expression data. *Bioinformatics*, 17:763--774.

[35] Yu, H. & Kim, S. (2010). *SVM Tutorial – Classification, Regression and Ranking*. Springer.