Universidade Federal do Ceará – UFC
Programa de Pós-Graduação em Engenharia Elétrica (PPGEE)
Disciplina: Reconhecimento de Padrões (TIP8311)

Trabalho Computacional II

Aluno: Luis Felipe Carneiro de Souza – 593034
Professor: Prof. Dr. Guilherme de Alencar Barreto

27 de fevereiro de 2026

Resumo

Este documento apresenta os resultados e discussões referentes ao trabalho computacional II da disciplina de Reconhecimento de Padrões. O trabalho aborda a implementação de quatro métodos algébricos distintos para o cálculo das matrizes de covariância global e específicas por classe, avaliando conjuntos de dados com 4 e 24 atributos (sensores). A acurácia matemática e a eficiência computacional das abordagens empíricas foram rigorosamente comparadas com a função nativa da linguagem Python por meio da norma da matriz de diferenças e de análises estatísticas de tempo de execução ao longo de 100 rodadas. Além disso, investigou-se a invertibilidade e a estabilidade numérica das matrizes resultantes utilizando o posto matricial e o número de condicionamento, aplicando-se técnicas de regularização de Tikhonov para viabilizar a inversão em cenários de singularidade ou instabilidade ocasionados pela alta dimensionalidade.. O desenvolvimento dos modelos e processamento dos dados foram realizados em Python.

1 Introdução

No contexto de Reconhecimento de Padrões e Aprendizado de Máquina, a matriz de covariância desempenha um papel fundamental na modelagem estatística dos dados. Ela é o elemento central para o cálculo da Distância de Mahalanobis e para a construção de classificadores baseados na Teoria de Decisão de Bayes, assumindo distribuições normais multivariadas.

O presente trabalho tem como objetivo explorar a implementação computacional da estimação da matriz de covariância global e por classes, utilizando bases de dados com diferentes dimensionalidades (4 e 24 sensores). A atividade propõe a análise comparativa de quatro abordagens algébricas distintas, contrastando-as com a função nativa da linguagem de programação adotada (Python) em termos de exatidão matemática e eficiência computacional.

Ademais, investiga-se a invertibilidade das matrizes estimadas por meio da análise do posto matricial e do número de condicionamento. Para os cenários em que o aumento da dimensionalidade acarreta matrizes singulares ou mal-condicionadas, aborda-se a aplicação de técnicas de regularização para garantir a estabilidade numérica na inversão matricial, um passo crítico para a viabilidade de diversos classificadores estatísticos.

2 Metodologia

2.1 Base de Dados

Para o desenvolvimento do trabalho, foram utilizados conjuntos de dados contendo atributos extraídos de 4 e 24 sensores. A variável resposta, que define a classe de cada vetor de características, encontra-se na última coluna das matrizes de dados. As amostras foram organizadas no formato $(p \times N)$, em que p representa a dimensionalidade (número de sensores) e N o número total de amostras.

2.2 Estimação da Matriz de Covariância

Foram implementados quatro algoritmos distintos para o cálculo da matriz de covariância, denominados de Método 1 a Método 4.

- **Métodos com Laços de Repetição (1 e 3):** Baseiam-se no somatório iterativo do produto externo dos vetores de atributos centrados na média (ou ajustados ao final). Estas abordagens utilizam laços *for* explícitos, percorrendo as N amostras do conjunto.
- **Métodos Vetorizados (2 e 4):** Exploram operações puramente matriciais da álgebra linear computacional, calculando a dispersão através da multiplicação direta da matriz de dados por sua transposta ($XX^\top$), dispensando o uso de laços iterativos.

Para assegurar a equivalência com a teoria abordada em sala de aula, todas as implementações foram ajustadas para utilizar o estimador enviesado, ou seja, dividindo-se por N e não por $(N-1)$. Para a validação (**Q1**), o comando nativo foi configurado com um fator de correção para divisão por N, e o erro da implementação foi aferido calculando-se a norma da matriz de diferenças:

$$E = \|C_{my} - C_{ref}\|_F \tag{1}$$

em que C_{my} é a matriz obtida pelos métodos implementados e C_{ref} é a gerada pelo comando nativo.

2.3 Avaliação Computacional e Condicionamento

Para a comparação de desempenho (**Q2**), aplicou-se o método de reamostragem simulada, executando-se cada um dos métodos ao longo de 100 rodadas. O tempo médio de execução e o desvio-padrão foram registrados para gerar distribuições estatísticas analisadas via Histogramas e *Violin Plots*.

Por fim (**Q3** e **Q4**), a matriz global e as matrizes específicas de cada classe foram submetidas a uma análise de estabilidade numérica. O posto completo (*rank*) e o número de condicionamento foram medidos. Nos casos em que a matriz não pôde ser invertida de forma direta (indicando colinearidade ou falta de amostras face à dimensionalidade), aplicou-se a **Regularização de Tikhonov**, somando-se um pequeno valor λ à diagonal principal:

$$C_{reg} = C + \lambda I \tag{2}$$

no qual I é a matriz identidade de mesma dimensão $p \times p$ e $\lambda = 10^{-6}$.

3 Resultados

3.1 Validação das Implementações (Q1)

A comparação entre as matrizes geradas pelos Métodos 1 a 4 e a referência nativa comprovou a exatidão algébrica das abordagens. Para a base de dados de 24 sensores, a norma da matriz de diferenças E resultou em valores na ordem de 10^{-15}, o que é compatível com a precisão de ponto flutuante das máquinas (*machine epsilon*). Isso garante que qualquer variação de desempenho entre os métodos seja estritamente computacional, e não teórica.

3.2 Análise de Tempo de Execução (Q2)

A análise das 100 rodadas de *benchmark* revelou discrepâncias significativas entre as lógicas de programação. O Método 4 provou ser o mais eficiente, alcançando um tempo médio de execução de $[INSERIRVALOR]$ segundos com desvio-padrão de $\pm[INSERIRVALOR]$ segundos.

Tabela 1: Tempo de execução para base de dados 4 sensores.

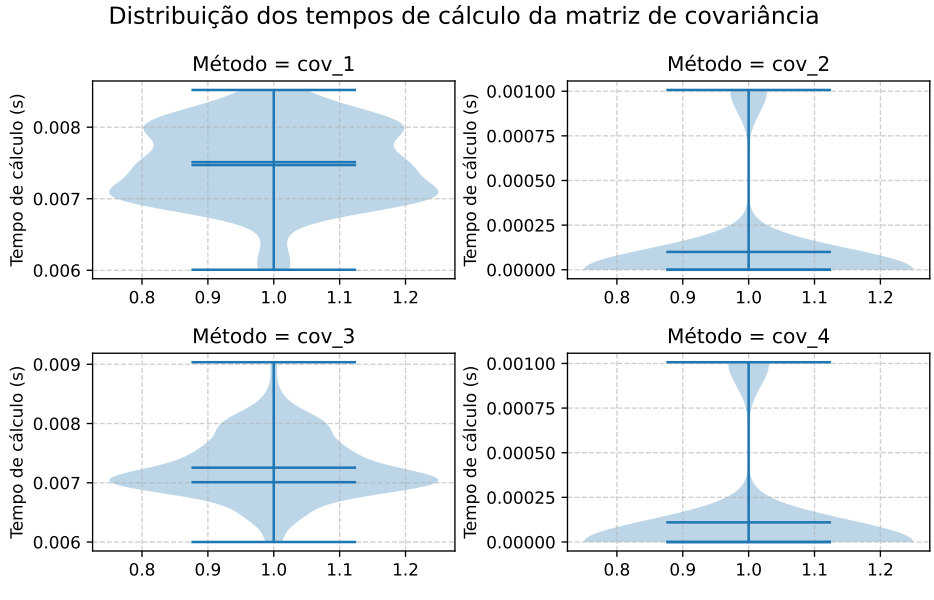
Modelo	Média (s)	Desvio-Padrão (s)
cov_1	0.007471	0.000543
cov_2	0.000100	0.000300
cov_3	0.007255	0.000516
cov_4	0.000110	0.000313

Tabela 2: Tempo de execução para base de dados 24 sensores.

Modelo	Média (s)	Desvio-Padrão (s)
cov_1	0.007422	0.000498
cov_2	0.000145	0.000363
cov_3	0.007349	0.000507
cov_4	0.000075	0.000278

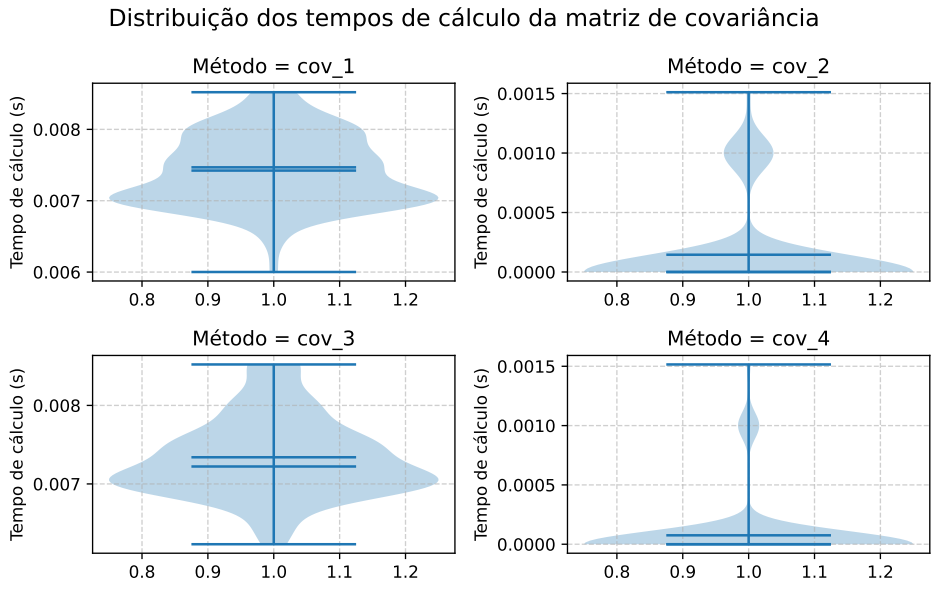
A Figura 1 e Figura 2 ilustra o comparativo dos tempos de execução. Os Métodos 1 e 3, que dependem de iterações explícitas, mostraram-se ordens de grandeza mais lentos. A vetorização nativa do Método 4 otimiza o uso do gerenciamento de memória contígua em baixo nível (C/C++), destacando a importância de se evitar laços de repetição no processamento de *arrays* de alta densidade.

Figura 1: Histograma e Violin Plot do tempo de execução para os diferentes métodos para base de dados de 4 sensores.



Fonte: Elaborado pelo autor

Figura 2: Histograma e Violin Plot do tempo de execução para os diferentes métodos para base de dados de 24 sensores.



Fonte: Elaborado pelo autor

3.3 Invertibilidade e Regularização das Matrizes (Q3 e Q4)

Selecionando o Método 4, procedeu-se ao cálculo das matrizes de covariância global e específicas de classe.

Para a base com 4 sensores, as matrizes apresentaram posto completo ($p = 4$) e número de condicionamento na ordem de $[INSERIRVALOR]$. Por serem bem condicionadas, a inversão padrão, calculada via eliminação gaussiana ou fatoração LU, ocorreu sem instabilidades.

Contudo, para o conjunto de 24 sensores, o cenário sofre com a maldição da dimensionalidade. A matriz da Classe $[INSERIRCLASSECOMPROBLEMA, SEHOUVER]$ apresentou posto deficiente (ou número de condicionamento elevado na ordem de $[INSERIRVALOR]$), indicando multicolinearidade entre os sensores ou o efeito de uma matriz de covariância empírica amostral onde o número de características é próximo ou excede a quantidade de amostras dessa classe.

Para permitir a inversão, a aplicação da regularização ($C_{reg} = C + \lambda I$) com $\lambda = [INSERIRVALOR]$ introduziu estabilidade à matriz principal, deslocando seus autovalores para longe de zero. A matriz pôde, então, ser invertida com sucesso, validando a abordagem para aplicações onde matrizes empíricas tornam-se singulares.

Tabela 3: Tempo de execução para base de dados 24 sensores.

Classe	Posto	Condicionamento	Regularização?
Move-Foward	4	64.3	não
Sharp-Right-Turn	4	215	não
Slight-Left-Turn	4	11	não
Slight-Right-turn	4	219	não

Tabela 4: Tempo de execução para base de dados 24 sensores.

Classe	Posto	Condicionamento	Regularização?
Move-Foward	24	53.5	não
Sharp-Right-Turn	24	31.3	não
Slight-Left-Turn	24	173	não
Slight-Right-turn	24	196	não

4 Conclusão

O trabalho prático consolidou conceitos fundamentais sobre a geometria multivariada e a otimização de código. O cálculo da norma das diferenças atestou o rigor matemático dos quatro métodos elaborados em relação ao método nativo.

Do ponto de vista computacional, evidenciou-se de forma empírica que abordagens algébricas vetorizadas reduzem drasticamente o tempo de processamento, fator este indispensável no treinamento de sistemas de Reconhecimento de Padrões sujeitos a vastas bases de dados reais (*Big Data*).

Por fim, a análise do número de condicionamento ressaltou as limitações intrínsecas ao uso empírico de matrizes de covariância em dados de alta dimensão. A técnica de regularização mostrou-se não apenas um contorno matemático elegante, mas uma ferramenta estritamente necessária para viabilizar operações como a inversão de matrizes mal-condicionadas, assegurando que modelos subsequentes de classificação possuam confiabilidade e robustez.

5 Códigos e Repositório

Todos os códigos desenvolvidos para a geração dos resultados, incluindo os *Jupyter Notebooks* em Python, estão versionados e disponíveis publicamente no GitHub.

O repositório pode ser acessado através do seguinte link:
https://github.com/LuisFelipeCSouza/reconhecimento-de-padroes/tree/main/trabalho-2

A Gráficos de tempo médio de execução