
# INSTITUTO FEDERAL DE MINAS GERAIS

## Bacharelado em Ciência da Computação

**Disciplina: Matemática Computacional**

Trabalho Prático

Professor: Diego Mello da Silva

Formiga-MG
19 de maio de 2017

# Sumário

**1 Nota** **1**

**2 Entrega, Prazo e Pontuação** **2**

**3 Um modelo simples de *Pattern Recognition*** **2**
3.1 Conceitos e Modelo . . . . . 2
3.2 Exemplo de Treinamento e Cálculo . . . . . 6

**4 *Datasets*** **8**

**5 Especificação** **11**
5.1 Requisito 01 - *Datasets* de Classificação . . . . . 11
5.2 Requisito 02 - Entrada de Dados . . . . . 13
5.3 Requisito 03 - Representação de Matrizes . . . . . 13
5.4 Requisito 04 - Cálculo dos Vetores Média . . . . . 13
5.5 Requisito 05 - Cálculo da Matriz de Covariância . . . . . 14
5.6 Requisito 06 - Operação de Transposta de Matriz . . . . . 14
5.7 Requisito 07 - Operação de Produto de Real por Matriz . . . . . 15
5.8 Requisito 08 - Operação de Soma/Diferença de Matrizes . . . . . 15
5.9 Requisito 09 - Operação de Produto Matricial . . . . . 15
5.10 Requisito 10 - Operação de Produto Escalar . . . . . 16
5.11 Requisito 11 - Solução de Sistemas Lineares por LU . . . . . 16
5.12 Requisito 12 - Operação de Inversa de Matrizes . . . . . 18
5.13 Requisito 13 - Treinamento do Classificador . . . . . 19
5.14 Requisito 14 - Cálculo *Maximum Likelihood Ratio Test* . . . . . 19
5.15 Requisito 15 - Classificação de *Dataset* . . . . . 20
5.16 Requisito 16 - Saída de Dados . . . . . 20
5.17 Requisito 17 - Modularização, Doc. e Corretude . . . . . 21

**6 Barema** **22**

# 1 Nota

O conteúdo descrito neste documento é inspirado nas Notas de Aula `1010n4a.pdf`[1] da disciplina ***ECE 1010 - Problem Solving in Engineering*** ministrada pelo Dr. Mark Wickert em 1997 na *University of Colorado - College of Engineering & Applied Sciences*. Segundo o site da disciplina[2], o curso consiste em *"an introductory course which combines elementary applied mathematics, basic numerical methods, computer programming, and problem solving methodology to introduce the student to tools and techniques which will be useful throughout his/her engineering career "*. Neste trabalho será abordado um modelo de reconhecimento de padrões descrito nas referidas notas de aula, página 20, que aplica alguns conceitos de álgebra linear, em especial as operações de produto vetorial, produto escalar, transposta de matriz e inversa de matriz.

# 2 Entrega, Prazo e Pontuação

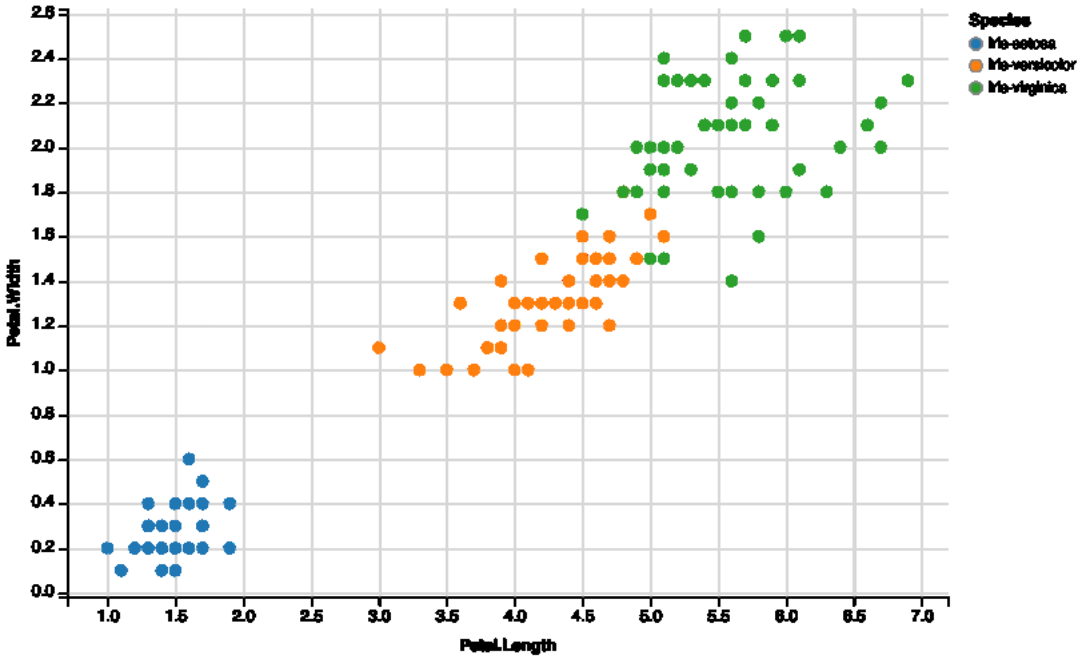
Este trabalho especifica a implementação de um aplicativo capaz de classificar dados de entradas com base em classes já conhecidas por meio de uma técnica de aprendizado de máquina muito simples, inspirada em estatística e operações de álgebra linear. O trabalho deverá ser codificado em linguagem C ou Pascal, e poderá ser desenvolvido em grupo de até 03 (três) alunos. A entrega deverá ser feita por meio de um único arquivo compactado em formato `.zip` contendo os arquivos de *dataset* e código fonte (`pattern.c` para implementações em linguagem C, ou `pattern.pas` para implementações em Pascal). Os códigos fontes serão corrigidos em ambiente Linux Ubuntu, com compiladores `gcc` e `fpc`, respectivamente. O trabalho deverá ser entregue **impreterivelmente** até o uma semana e meia antes do término das aulas. **Trabalhos plagiados ou com alta similaridade serão desconsiderados e valerão zero**. O trabalho tem valor total de 30 (trinta) pontos, distribuídos segundo o barema descrito na seção 6.

# 3 Um modelo simples de *Pattern Recognition*

## 3.1 Conceitos e Modelo

Este documento descreve a especificação do Trabalho Prático da disciplina de Matemática Computacional que trata do tema de construção de um reconhecedor de padrões simples que usa conceitos básicos de Algebra Linear, mais especificamente inversa de matrizes. O termo *pattern recognition* ou reconhecimento de padrões é usado para designar um ramo da área denominada aprendizado de máquina que tem por objetivo reconhecer 'padrões' ou 'regularidades' em dados. Portanto o reconhecimento de padrões tem por objetivo atribuir um 'rótulo' para um dado valor de entrada informado.

Um exemplo clássico de reconhecimento de padrões trata da **classificação**, onde o modelo de reconhecedor tenta atribuir cada valor de entrada para uma classe dentre um dado conjunto de classes. Para exemplificar, seja a figura abaixo que representa um plot de duas características de flores do tipo Iris, mais especificamente comprimento e largura da pétala, divididas em três espécies diferentes de Iris: virgínica (verde), versicolor (laranja) e setosa (azul). No plot realizado é evidente que existem três classes distintas de espécies de iris, que podem ser identificadas apenas com informações sobre o comprimento e largura das pétalas. Um bom algoritmo reconhecedor de padrões deve, com base nestas características de entrada, classificar adequadamente uma flor Iris em uma das três classes apenas considerando um vetor de entrada com duas características: comprimento da pétala e largura da pétala.



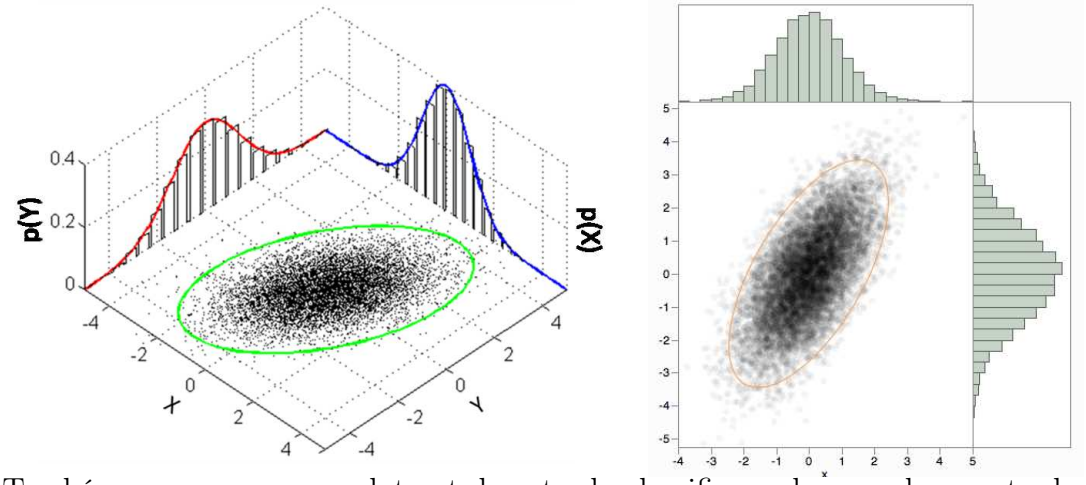

A figura acima [3] ilustra dados reais de um famoso *dataset* obtido pelo biólogo britânico Ronald Fisher em 1963 e documentado no artigo *The use of multiple measurements in taxonomic problems*. É um dos *datasets* mais famosos usados para classificação. Mais adiante será apresentado um repositório de *datasets* para diferentes problemas de aprendizado de máquina. Por ora considere apenas que um *dataset* consiste em um arquivo contendo coleções de dados em formato tabular, que pode ser manipulado com facilidade por um algoritmo computacional para testar seu desempenho ou qualidade de resultados.

Quando tratamos do tema de reconhecimento de padrões tipicamente consideramos que o modelo de reconhecimento recebe como entrada um **conjunto de medidas** na forma vetorial $\begin{bmatrix} x_1 & x_2 & \dots & x_n \end{bmatrix}$ que guardam valores obtidos, por exemplo, de uma coleção de sensores ou características e determina à qual classe ou padrão estes valores pertencem. Para tal o modelo reconhecedor é 'treinado' por meio de um conjunto de entrada de amostras representativas de cada classe que deve reconhecer. Após seu treinamento, o modelo reconhecedor poderá receber novas medidas (entradas) e decidir à qual classe conhecida elas pertencem. Atualmente as técnicas de reconhecimento de padrões tem aplicações amplas que variam de diagnóstico médico, detecção em sonares e radares, processamento de imagens, controle de processos, interpretação de fotos aéreas, previsão de tempo, reconhecimento de caracteres, reconhecimento de fala, dentre outros.

No modelo de reconhecimento apresentado neste trabalho, algumas considerações são feitas. A primeira delas é a consideração de que os vetores de medidas de cada classe são normalmente distribuídos com dimensão $n$, isto é, se tomarmos as medidas de uma determinada classe teremos que os valores das medidas se distribuem segundo uma distribuição de probabilidades gaussiana com uma grande concentração de valores aproximadamente simétricos ao redor da média. Para fins de ilustração, sejam as figuras abaixo que mostram vetores no espaço $\mathbb{R}^2$ representando medidas tomadas com duas dimensões. Se analisarmos cada dimensão em

[3] `https://www.datacamp.com/community/tutorials/machine-learning-in-r#gs.DFauUZQ`

separado, a distribuição dos valores de cada dimensão segue uma distribuição normal, com grande concentração de observações tomadas ao redor da média de cada dimensão (neste caso, média zero).




Também assume-se que o dataset de entrada classifica cada uma de suas tuplas previamente dentre uma das $M$ classes conhecidas. Isto posto, é assumido neste modelo que os vetores medida de cada classe são normalmente distribuídos com um **vetor média** particular $m_k$ de $n$ dimensões, e **matriz de covariância** $C$ de dimensões $n \times n$ e $k \in \{1, 2, \ldots, M\}$. No contexto de classificação de padrões a matriz de covariância mostra-nos como cada um dos componentes no vetor de medidas estão estatisticamente relacionados aos outros vetores de uma dada classe.

Formalmente, um vetor média consiste nas médias de cada variável, e a matriz de covariância consiste na variância das variáveis ao longo da diagonal principal e a covariância entre cada par de variáveis nas demais posições da matriz. De maneira geral, para a $k$-ésima classe podemos obter a $j$-ésima componente do vetor média aplicando a Equação (1)

$$\boxed{m_k^{(j)} = \frac{1}{n_k}\left(\sum_{i=1}^{n_k} x_{ij}\right)} \tag{1}$$

onde:

$m_k^{(j)}$: $j$-ésima componente do vetor média da classe $k$ (isto é, índice da coluna cuja média será calculada);

$n_k$: número de observações da $k$-ésima classe (isto é, quantidade de linhas do *dataset* que possui tuplas relacionadas com a classe $k$);

$x_{ij}$: observação contida na linha $i$, coluna $j$ do dataset de treinamento (isto é, valor contido na $i$-ésima linha da coluna cuja média está sendo calculada).

Após calcular a média de cada variável $j$, com $j = 1, \ldots, n$, constrói-se o vetor média para a $k$-ésima classe organizando as componentes como segue:

$$m_k = \begin{bmatrix} m_k^{(1)} & m_k^{(2)} & \ldots & m_k^{(n)} \end{bmatrix}^T$$

Seja um *dataset* com $n$ linhas de dados, e sejam $X_1$ e $X_2$ duas colunas quaisquer deste *dataset*. A fórmula que computa a covariância entre as duas variáveis $X_1$ e $X_2$ com média amostral $\overline{x}_1$ e $\overline{x}_2$, respectivamente, é dada pela Equação (2)

$$\boxed{cov(X_1, X_2) = \frac{1}{(n-1)} \sum_{i=1}^{n} (x_{1i} - \overline{x}_1)(x_{2i} - \overline{x}_2)} \tag{2}$$

onde

$n$: número de linhas totais do *dataset*;

$x_{1i}$: observação contida na $i$-ésima linha da variável $X_1$;

$x_{2i}$: observação contida na $i$-ésima linha da variável $X_2$;

$\overline{x}_1$: média de todos os valores observados da variável $X_1$ (isto é, média dos valores da coluna representada por $X_1$);

$\overline{x}_2$: média de todos os valores observados da variável $X_2$ (isto é, média dos valores da coluna representada por $X_2$).

Fazendo-se o cálculo da covariância de cada variável do *dataset* (isto é, cada coluna de entrada) duas a duas obtem-se todas as combinações possíveis de resultados. A matriz de covariância $C$ é dada, portanto, pelo cômputo:

$$C = \begin{bmatrix} cov(X_1, X_1) & cov(X_1, X_2) & \dots & cov(X_1, X_n) \\ cov(X_2, X_1) & cov(X_2, X_2) & \dots & cov(X_2, X_n) \\ \vdots & \vdots & \ddots & \vdots \\ cov(X_n, X_1) & cov(X_n, X_2) & \dots & cov(X_n, X_n) \end{bmatrix}$$

De posse dos vetores de médias $m_k$ e da matriz de covariância $C$ calcula-se a pertinência de uma nova observação $x$ de dimensão $n$ (isto é, um vetor de $n$ dimensões que não tenha sido usada para calcular a média nem a covariância entre as variáveis) utilizando o resultado do chamado ***maximum likelihood ratio test*** entre duas classes $i$ e $j$ conhecidas no treinamento por meio da Equação (3)

$$\boxed{r_{ij}(x) = x^T C^{-1}(m_i - m_j) - \frac{1}{2}(m_i + m_j)^T C^{-1}(m_i - m_j)} \tag{3}$$

onde:

$r_{ij}(x)$: escalar que resulta do teste para entrada $x$ considerando classes $i$ e $j$.

$x^T$: vetor transposto da entrada informada, de dimensões $(1 \times n)$

$C^{-1}$: matriz inversa da matriz de covariância, de dimensões $(n \times n)$

$m_i$: vetor média da classe $i$, de dimensões $(n \times 1)$

$m_j$: vetor média da classe $j$, de dimensões $(n \times 1)$

Se tomarmos o resultado deste teste através de todos os pares $ij$, com $i = 1, \ldots, M$, $j = 1, \ldots, M$, e $i \neq j$, então selecionamos o maior e dizemos que se uma observação particular $x$ possui como maior ***maximum likelihood ratio test*** o valor $r_{kj}$, então $x$ pertence à classe $k$, onde $k \in \{1, 2, \ldots, M\}$. Em outras palavras, testam-se todas as combinações de classes $ij$, exceto quando $i = j$, e devolve a classe $k$ que possui maior valor de teste $r_{kj}$ para o vetor $x$ de entrada.

## 3.2 Exemplo de Treinamento e Cálculo

Seja um *dataset* hipotético que contém uma coleção de dados distribuídos em $m$ linhas e $n$ colunas. Cada linha representa uma entrada do *dataset*, enquanto que cada coluna representa uma característica, leitura ou variável do *dataset*. O vetor média pode ser computado considerando a média dos valores de cada coluna agrupados em tuplas que pertencem à mesma classe, isto é, se o vetor de medidas obtido do *dataset* para a classe $k$ possui $n_k$ linhas, então o vetor média terá $n$ colunas e conterá em cada um dos $j$ componentes a média dos valores observados nas linhas da $j$-ésima coluna correspondente. Para exemplificar, seja o seguinte *dataset*, contido em um arquivo ASCII com primeira linha contendo o nome das 04 (quatro) variáveis de entrada e 01 (uma) coluna de classificação (i.e., nome das colunas do *dataset*) e as demais linhas contendo os valores observados e classe previamente conhecida.

```
V1   V2   V3   V4   C
5.1  3.5  1.4  0.2  1
4.9  3.0  1.4  0.2  1
4.7  3.2  1.3  0.2  1

4.6  3.1  1.5  0.2  2
5.0  3.6  1.4  0.2  2

5.4  3.9  1.7  0.4  3
5.2  3.7  1.7  0.3  3
5.7  3.2  1.5  0.2  3
```

Teremos para cada uma das três classes do *dataset* os seguinte vetor média correspondentes com aproximação de 02 (duas) casas decimais:

$$m_1 = \begin{bmatrix} 4.90 \\ 7.56 \\ 2.23 \\ 0.20 \end{bmatrix}, m_2 = \begin{bmatrix} 4.80 \\ 3.35 \\ 1.45 \\ 0.20 \end{bmatrix}, m_3 = \begin{bmatrix} 5.43 \\ 3.60 \\ 1.63 \\ 0.30 \end{bmatrix}$$

A matriz de covariância $C$ para os dados do exemplo é dada a seguir, com 04 (quatro) casas decimais de precisão.

$$C = \begin{bmatrix} 0.1307 & 0.0500 & 0.02678 & 0.0110 \\ 0.0500 & 0.1028 & 0.03000 & 0.0185 \\ 0.0267 & 0.0300 & 0.02125 & 0.0091 \\ 0.0110 & 0.0185 & 0.00910 & 0.0055 \end{bmatrix}$$

Com base nela calcula-se a sua inversa $C^{-1}$, que é dada a seguir com 04 (quatro) casas de precisão.

$$C^{-1} = \begin{bmatrix} 10.8463 & -3.7063 & -15.3898 & 16.2372 \\ -3.7042 & 25.9493 & 7.6833 & -92.5880 \\ -15.2495 & 7.6383 & 183.3322 & -298.5252 \\ 15.9980 & -92.5094 & -298.3958 & 954.6997 \end{bmatrix}$$

Seja agora o cálculo de máxima verossimilhança de um dado vetor de entrada $x = \begin{bmatrix} 5.0 & 3.2 & 1.7 & 1.7 \end{bmatrix}$ considerando as classes 2 e 3, isto é, $r_{23}$. Teríamos, portanto, o seguinte cálculo derivado da Equação (3) adaptado para os dados das classes envolvidas. Dividiremos o cálculo em duas parcelas, destacadas abaixo.

$$r_{23}(x) = \underbrace{x^T C^{-1}(m_2 - m_3)}_{(I)} - \underbrace{1/2(m_2 + m_3)^T C^{-1}(m_2 - m_3)}_{(II)}$$

Calculando a parte $(I) = x^T C^{-1}(m_2 - m_3)$, temos

$$(I) = \begin{bmatrix} 5.0 & 3.2 & 1.7 & 1.7 \end{bmatrix} \begin{bmatrix} 10.8463 & -3.7063 & -15.3898 & 16.2372 \\ -3.7042 & 25.9493 & 7.6833 & -92.5880 \\ -15.2495 & 7.6383 & 183.3322 & -298.5252 \\ 15.9980 & -92.5094 & -298.3958 & 954.6997 \end{bmatrix} \left( \begin{bmatrix} 4.80 \\ 3.35 \\ 1.45 \\ 0.20 \end{bmatrix} - \begin{bmatrix} 5.43 \\ 3.60 \\ 1.63 \\ 0.30 \end{bmatrix} \right)$$

$$(I) = -52.96157$$

Calculando a parte $(II) = 1/2(m_2 + m_3)^T C^{-1}(m_2 - m_3)$, temos

$$(II) = \frac{1}{2} \left( \begin{bmatrix} 4.80 \\ 3.35 \\ 1.45 \\ 0.20 \end{bmatrix} + \begin{bmatrix} 5.43 \\ 3.60 \\ 1.63 \\ 0.30 \end{bmatrix} \right)^T \begin{bmatrix} 10.8463 & -3.7063 & -15.3898 & 16.2372 \\ -3.7042 & 25.9493 & 7.6833 & -92.5880 \\ -15.2495 & 7.6383 & 183.3322 & -298.5252 \\ 15.9980 & -92.5094 & -298.3958 & 954.6997 \end{bmatrix} \left( \begin{bmatrix} 4.80 \\ 3.35 \\ 1.45 \\ 0.20 \end{bmatrix} - \begin{bmatrix} 5.43 \\ 3.60 \\ 1.63 \\ 0.30 \end{bmatrix} \right)$$

$$(II) = -11.58379$$

Realizando a diferença entre a parte (I) e parte (II), temos

$$r_{23} = (I) - (II) = -52.96157 - (-11.58379) = -41.37778$$

Logo, o valor de $r_{23}$ calculado é $-41.37778$. Procedendo de maneira semelhante para os cálculos dos outros pares $r_{12}$, $r_{13}$, $r_{21}$, $r_{31}$ e $r_{32}$ teremos todos os valores do teste de máxima verossimilhança calculados. O maior $r_{kj}$ deles indicará, portanto, a classe recomendada $k$ pelo algoritmo de reconhecimento de padrões dados os vetores média e matriz de covariância obtidas pelo treinamento. Para fins de aprendizado e melhor compreensão da técnica consultar um arquivo de *script*[4] na linguagem R disponível no site da disciplina que poderá ser usado para conferir os resultados da implementação do grupo.

# 4 *Datasets*

*Datasets* são coleções de dados sobre determinada classe de problemas ou temas. Muitos são arquivos em formato ASCII tabulados, uma observação por linha, que contêm atributos descritos em suas colunas. *Datasets* possuem papel fundamental em Ciência da Computação por prover um conjunto conhecido de observações para problemas de diversas áreas tais como Pesquisa Operacional (*benchmarks* para otimização combinatória, por exemplo), Inteligência Artificial (*datasets* com dados para teste e implementação de métodos de aprendizado de máquina), Mineração de Dados (aprendizado de máquina, descoberta de conhecimento, etc), Estatística (análise de dados, inferência, séries temporais, etc), Bioinformática (bases de dados com genoma, etc) dentre outros.

Um dos mais famosos repositórios de *datasets* é o ***UC Irvine Machine Learning Repository***[5] do *Center of Machine Learning and Intelligent Systems*. Na data de publicação deste trabalho o repositório mantinha 317 conjuntos de dados à serviço da comunidade de Aprendizado de Máquina, oriundos de trabalhos realizados sobre Ciências da Vida, Ciências Físicas, Ciência da Computação, Engenharias, Ciências Sociais, Administração e Negócios, Jogos e outros. Os *datasets* são rotulados como sendo adequados para problemas de classificação (271 conjuntos), regressão (69 conjuntos), agrupamento (57 conjuntos) e outros tipos de problema em aprendizado de máquina (52 conjuntos). Quanto aos atributos do *dataset*, podem ser categóricos e/ou numéricos. O link de cada *dataset* no repositório dá acesso aos dados propriamente ditos, e à informações sobre a estrutura do *dataset*.

Seguem alguns exemplos de *datasets* clássicos do UCI famosos por seu uso em problemas de classificação: Iris (`http://archive.ics.uci.edu/ml/datasets/Iris`), que contêm informações sobre tamanho e largura de pétalas e sépalas de flores do tipo Iris, Wine (`https://archive.ics.uci.edu/ml/datasets/Wine`) com resultados de análises químicas de vinhedos que cresceram na mesma região da Itália mas derivaram de três diferentes cultivares; Seeds (`http://archive.ics.uci.edu/ml/`

[4] `https://sites.google.com/a/ifmg.edu.br/diegosilva/disciplinas/2017/matematica-computacional/pattern-model.r?attredirects=0&d=1`

[5] `http://archive.ics.uci.edu/ml/`

datasets/seeds#), que possui dados sobre medidas de propriedades geométricas de grãos pertencentes à três diferentes variedades de trigo.

Outros *datasets* mais específicos da área de Ciência da Computação/Engenharia distribuídos entre as categorias de classificação, agrupamento, e regressão são:

- Anonymous Microsoft Web Data

  http://archive.ics.uci.edu/ml/datasets/Anonymous+Microsoft+Web+Data ,

- Artificial Characters

  http://archive.ics.uci.edu/ml/datasets/Artificial+Characters,

- Computer Hardware

  http://archive.ics.uci.edu/ml/datasets/Computer+Hardware,

- Image Segmentation

  http://archive.ics.uci.edu/ml/datasets/Image+Segmentation,

- Internet Advertisements

  http://archive.ics.uci.edu/ml/datasets/Internet+Advertisements,

- LED Display Domain

  http://archive.ics.uci.edu/ml/datasets/LED+Display+Domain,

- Letter Recognition

  http://archive.ics.uci.edu/ml/datasets/Letter+Recognition,

- Mobile Robots

  http://archive.ics.uci.edu/ml/datasets/Mobile+Robots,

- Page Blocks Classification

  http://archive.ics.uci.edu/ml/datasets/Page+Blocks+Classification,

- Servo Dataset

  http://archive.ics.uci.edu/ml/datasets/Servo,

- Shuttle Landing Control

  http://archive.ics.uci.edu/ml/datasets/Shuttle+Landing+Control,

- Waveform Database Generator

  http://archive.ics.uci.edu/ml/datasets/Waveform+Database+Generator+\%28Version+1\%29,

- Internet Usage

  http://archive.ics.uci.edu/ml/datasets/Internet+Usage+Data,

- Pionner-1 Mobile Robot Data

  http://archive.ics.uci.edu/ml/datasets/Pioneer-1+Mobile+Robot+Data,

- Robot Execution Failures

  http://archive.ics.uci.edu/ml/datasets/Robot+Execution+Failures,

- UNIX User Data

  http://archive.ics.uci.edu/ml/datasets/UNIX+User+Data,

- Character Trajectories

  http://archive.ics.uci.edu/ml/datasets/Character+Trajectories,

- URL Reputation

  http://archive.ics.uci.edu/ml/datasets/URL+Reputation,

- Record Linkage Comparison

  http://archive.ics.uci.edu/ml/datasets/Record+Linkage+Comparison+Patterns,

- Amazon Commerce Reviews

  http://archive.ics.uci.edu/ml/datasets/Amazon+Commerce+reviews+set,

- Amazon Access Samples

  http://archive.ics.uci.edu/ml/datasets/Amazon+Access+Samples,

- DBWorld e-mails

  http://archive.ics.uci.edu/ml/datasets/DBWorld+e-mails,

- Youtube Comedy Slam

  http://archive.ics.uci.edu/ml/datasets/YouTube+Comedy+Slam+Preference+Data,

- SMS Spam Collection

  http://archive.ics.uci.edu/ml/datasets/SMS+Spam+Collection,

- Wearable Computing

  http://archive.ics.uci.edu/ml/datasets/Wearable+Computing\%3A+Classification+of+Body+Postures+and+Movements+\%28PUC-Rio\%29,

- BLOGGER

  http://archive.ics.uci.edu/ml/datasets/BLOGGER,

- Youtube Multiview Video Games

  http://archive.ics.uci.edu/ml/datasets/YouTube+Multiview+Video+Games+Dataset,

- USPTO Algorithm Challenge

  http://archive.ics.uci.edu/ml/datasets/USPTO+Algorithm+Challenge\%2C+run+by+NASA-Harvard+Tournament+Lab+and+TopCoder++++Problem\%3A+Pat,

- GPS Trajectories

  http://archive.ics.uci.edu/ml/datasets/GPS+Trajectories,

- Detect Malacious Executable

  `http://archive.ics.uci.edu/ml/datasets/Detect+Malacious+Executable\%28AntiVirus\%29`,

- Facebook Comment Volume

  `http://archive.ics.uci.edu/ml/datasets/Facebook+Comment+Volume+Dataset`,

- Gas Sensors For Home Activity Monitoring

  `http://archive.ics.uci.edu/ml/datasets/Gas+sensors+for+home+activity+monitoring`,

- Facebook Metrics

  `http://archive.ics.uci.edu/ml/datasets/Facebook+metrics`,

- Website Phishing

  `http://archive.ics.uci.edu/ml/datasets/Website+Phishing`, e

- Youtube SPAM Collection

  `http://archive.ics.uci.edu/ml/datasets/YouTube+Spam+Collection`.

Cada *dataset* possui uma estrutura única de registrar as informações. Para compreender sua estrutura, atributos e significado de cada atributo acesse o *dataset*, seguido pelo link *Data Set Description* correspondente. Muitos *datasets* podem ter campos em branco, sendo necessário aplicar tratamento aos dados antes de utilizá-los.

# 5 Especificação

Esta seção descreve os requisitos do trabalho que, se implementados na totalidade, resultarão no classificador objeto deste trabalho.

## 5.1 Requisito 01 - *Datasets* de Classificação

Neste requisito o grupo deverá providenciar 02 (dois) *dataset* diferentes para teste de sua implementação. Eles devem ser adaptados de qualquer *dataset* de classificação da ***UCI Machine Learning Repository***, e devem ser adaptados para o formato dado a seguir que já descreve junto aos dados quais são as colunas de entrada e a classe de saída. Cada *dataset* deverá ser dividido em duas partes: uma para treinamento, e outra para classificação. São elegíveis quaisquer *datasets* de classificação do repositório, exceto o *dataset* Iris já apresentado em sala de aula. Cada *dataset* deverá ser construído em arquivos texto plano (ASCII), sem formatação, utilizando editores como `Notepad++`, `Notepad`, `gedit` ou similar.

O *dataset* original deverá ser adaptado para o seguinte formato, dado abaixo. Nela, cada linha do arquivo inicia com um caracter de controle. Linhas iniciadas com `'#'` indicam linhas de comentário e devem ser ignoradas até encontrar um caracter fim de linha; a linha iniciada com `'N'` indica a quantidade de tuplas que estão contidas no arquivo, uma por linha; a linha iniciada com caracter `'V'` indica a quantidade de variáveis do *dataset*, isto é, quantas colunas possui sendo necessariamente $(V-1)$

de entrada e a última de classificação; a linha iniciada com o caracter `'K'` indica a quantidade de diferentes classes contidas no *dataset* (iniciadas, obrigatoriamente, em 0); as linhas iniciadas com o caracter `'t'` indicam a listagem dos $V$ valores que compõem uma tupla do *dataset*; por fim o caracter `'e'` indica término do arquivo de entrada.

```
################################################################################
#                                                                              #
# IRIS-TREINAMENTO.TXT                                                         #
#                                                                              #
# Dataset inspirado no dataset Iris de Fisher                                  #
# Link: http://archive.ics.uci.edu/ml/datasets/Iris                            #
#                                                                              #
################################################################################

# Numero de linhas totais do dataset
N 15

# Numero de variaveis (colunas) do dataset. Destes, (V-1) sao entrada, e a
# ultima e' a classe
V 5

# Numero de classes do dataset
K 3

# Listagem das tuplas do dataset, incluido as (V-1) entradas e 1 classe. Neste
# caso, as colunas correspondem as caracteristicas de flores Iris, a saber:
# Coluna 1 (entrada) : sepal.length; Coluna 2 (entrada) : sepal.width,
# Coluna 3 (entrada) : petal.length; Coluna 4 (entrada) : petal.width,
# Coluna 5 (saida)   : class

# Tuplas da classe 0: 'Iris Setosa'
t 5.1  3.5  1.4  0.2  0
t 4.9  3.0  1.4  0.2  0
t 4.7  3.2  1.3  0.2  0
t 4.6  3.1  1.5  0.2  0
t 5.0  3.6  1.4  0.2  0

# Tuplas da classe 1: 'Iris Versicolor'
t 7.0  3.2  4.7  1.4  1
t 6.4  3.2  4.5  1.5  1
t 6.9  3.1  4.9  1.5  1
t 5.5  2.3  4.0  1.3  1
t 6.5  2.8  4.6  1.5  1

# Tuplas da classe 2: 'Iris Virginica'
t 6.9  3.2  5.7  2.3  2
t 5.6  2.8  4.9  2.0  2
t 7.7  2.8  6.7  2.0  2
t 6.3  2.7  4.9  1.8  2
t 6.7  3.3  5.7  2.1  2

# Fim de arquivo
e
```

## 5.2 Requisito 02 - Entrada de Dados

Neste requisito o grupo deverá implementar a entrada de dados da aplicação. Esta entrada será feita por linha de comando, que receberá dois arquivos de entrada. O primeiro arquivo corresponde ao *dataset* de treinamento, que conterá as tuplas de onde serão calculados os vetores média e a matriz de covariância. O segundo arquivo corresponderá no arquivo de verificação e será usado para determinar o percentual de sucesso do classificador com base no treinamento realizado. Em ambos os casos, o formato dos arquivos deverá respeitar as regras determinadas no Requisito 5.1.

A sintaxe esperada para esta aplicação é a que segue:

```
usar@machine$ ./pattern <dataset-treinamento> <dataset-verificacao>
```

Ao abrir o arquivo deve-se armazenar em tabelas em memória todas as tuplas do *dataset*, com o atributo classe correspondente, sendo uma tabela para os dados de treinamento, e outra para os dados de classificação. Tais dados serão usados adiante para o cálculo dos vetores média $m_k$, para $k \in \{0, 1, \ldots, (K-1)\}$ (ver Requisito 5.4) e para o cálculo da matriz de covariância $C$ (ver Requisito 5.5) que são os parâmetros obtidos após treinamento do classificador. Este requisito dispara uma série de processos na aplicação, a saber: recebe-se os dados de entrada, treina-se o classificador com os dados do arquivo de treinamento, computa-se a classe para cada tupla do arquivo de verificação, e exibe-se a saída de dados da aplicação segundo modelo que pode ser consultado diretamente no Requisito 5.16.

## 5.3 Requisito 03 - Representação de Matrizes

Neste requisito o grupo deverá implementar um Tipo Abstrato de Dados (TAD) matriz, que deve armazenar pelo menos as seguintes informações para uma dada matriz $A$ hipotética:

- Número de Linhas
- Número de Colunas
- *Buffer* de tamanho variável para armazenar cada elemento $[a_{ij}]$ da matriz $A$

Ela deverá ser a estrutura de dados básica dos cômputos do classificador. Ela será usada tanto para informar os dados da matriz nos métodos que implementam operações de Álgebra Linear, quanto para obter o resultado destes métodos. Deverá ser alocada dinamicamente pois as dimensões de uma matriz poderá variar de acordo com a operação aplicada, sendo necessário observar a alocação e desalocação de memória.

## 5.4 Requisito 04 - Cálculo dos Vetores Média

Neste requisito o grupo deverá implementar um método computacional que opera sobre os valores de todas as colunas de entrada do *dataset* informado, calculando um

vetor para cada classe em questão. O vetor conterá a média dos valores observados, por classe, de cada uma das $(V-1)$ colunas de entrada do dataset de treinamento por meio da fórmula dada na Equação (1). Após construir o método que gera um vetor para a classe $k$ deve-se aplicá-lo para todas as $K$ classes descritas no *dataset*, armazenando-os sob a forma da TAD matriz. Dados $K$ classes e $V$ variáveis, espera-se portanto obter os seguintes vetores de dimensão $(V-1) \times 1$:

$$m_0 = \begin{bmatrix} m_0^{(1)} \\ m_0^{(2)} \\ \vdots \\ m_0^{(V-1)} \end{bmatrix}, m_1 = \begin{bmatrix} m_1^{(1)} \\ m_1^{(2)} \\ \vdots \\ m_1^{(V-1)} \end{bmatrix}, \ldots m_k = \begin{bmatrix} m_k^{(1)} \\ m_k^{(2)} \\ \vdots \\ m_k^{(V-1)} \end{bmatrix}$$

| | |
|---|---|
| **Entrada** | Tabela de valores observados da $k$-ésima classe |
| **Saída** | Vetor de dimensão $n_k \times 1$ contendo as médias das $V$ variáveis para classe $k$ |

## 5.5 Requisito 05 - Cálculo da Matriz de Covariância

Neste requisito o grupo deverá calcular, para cada par de variáveis $i, j \in \{1, 2, \ldots, V\}$ a covariância existente entre a variável $i$ e a variável $j$. Tal cálculo deverá seguir a fórmula descrita na Equação 2, e os resultados deverão ser armazenados como elementos de uma matriz $C$ de dimensões $V \times V$. A inversa desta matriz será usada a posteriori no cálculo de máxima verossimilhança de um dado vetor de entrada $x$, descrito no Requisito 5.14. O resultado deverá ser armazenado em memória usando uma TAD matriz.

| | |
|---|---|
| **Entrada** | Tabela de valores observados de todas as classes |
| **Saída** | Matriz de dimensão $V \times V$ contendo os valores de $cov(i, j)$ para os pares $i, j \in \{1, 2, \ldots, V\}$ |

## 5.6 Requisito 06 - Operação de Transposta de Matriz

Neste requisito o grupo deverá implementar a operação de matriz transposta. Recebe-se uma matriz $A$ de dimensões $m \times n$, e devolve-se uma matriz $A^T$ de dimensões $n \times m$ onde os elementos das linhas de $A$ tornam-se os elementos da coluna de $A^T$, como ilustra o exemplo abaixo. O resultado deverá ser armazenado em uma TAD matriz.

$$A = \begin{bmatrix} a_{11} & a_{12} & \ldots & a_{1n} \\ a_{21} & a_{22} & \ldots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \ldots & a_{mn} \end{bmatrix}, A^T = \begin{bmatrix} a_{11} & a_{21} & \ldots & a_{m1} \\ a_{12} & a_{22} & \ldots & a_{m2} \\ \vdots & \vdots & \ddots & \vdots \\ a_{1n} & a_{2n} & \ldots & a_{mn} \end{bmatrix}$$

| | |
|---|---|
| **Entrada** | Matriz $A$ de dimensão $m \times n$ |
| **Saída** | Matriz $A^T$ de dimensão $n \times m$, onde linhas de $A$ são colunas de $A^T$ |

## 5.7 Requisito 07 - Operação de Produto de Real por Matriz

Neste requisito o grupo deverá implementar o produto de um escalar real $\alpha \in \mathbb{R}$ por uma matriz. Esta operação será usada para calcular o resultado do fator $\boxed{1/2(m_i + m_j)}$ da parte (II) da Equação (3). Por definição, o resultado é

$$\alpha\, A = \alpha \left( \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{bmatrix} \right) = \begin{bmatrix} \alpha\, a_{11} & \alpha\, a_{12} & \dots & \alpha\, a_{1n} \\ \alpha\, a_{21} & \alpha\, a_{22} & \dots & \alpha\, a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ \alpha\, a_{m1} & \alpha\, a_{m2} & \dots & \alpha\, a_{mn} \end{bmatrix}$$

| | |
|---|---|
| **Entrada** | Matriz de dimensão $m \times n$ ; fator de escala $\alpha \in \mathbb{R}$ |
| **Saída** | Matriz de dimensão $n \times m$, com cada componente escalado em $\alpha$ unidades |

## 5.8 Requisito 08 - Operação de Soma/Diferença de Matrizes

Neste requisito o grupo deverá implementar um método que, dado duas matrizes de dimensões compatíveis e um operador de soma ou subtração, resulta em uma matriz com a soma/subtração componente a componente de seus elementos. O resultado deverá ser armazenar em uma TAD matriz com as mesmas dimensões de cada matriz parcela. Para exemplificar, seja a soma

$$\begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{bmatrix} - \begin{bmatrix} b_{11} & b_{12} & \dots & b_{1n} \\ b_{21} & b_{22} & \dots & b_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ b_{m1} & b_{m2} & \dots & b_{mn} \end{bmatrix} = \begin{bmatrix} (a_{11} - b_{11}) & (a_{12} - b_{12}) & \dots & (a_{1n} - b_{1n}) \\ (a_{21} - b_{21}) & (a_{22} - b_{22}) & \dots & (a_{2n} - b_{2n}) \\ \vdots & \vdots & \ddots & \vdots \\ (a_{m1} - b_{m1}) & (a_{m2} - b_{m1}) & \dots & (a_{mn} - b_{mn}) \end{bmatrix}$$

| | |
|---|---|
| **Entrada** | Matrizes $A$ e $B$ de dimensões compatíveis $m \times n$ ; operação de '+' ou '-' |
| **Saída** | Matriz de dimensão $m \times n$, com cada componente dado pela soma $(a_{ij} + b_{ij})$, ou pela diferença $(a_{ij} - b_{ij})$ |

## 5.9 Requisito 09 - Operação de Produto Matricial

Neste requisito o grupo deverá implementar a operação de produto matricial. Dadas duas matrizes $A$ e $B$, determinar o produto $A \times B = C$ por meio de uma implementação que segue o pseudo-código dado pelo Algoritmo 1. Este código será amplamente utilizado no cálculo do teste de máxima verossimilhança, dado no Requisito 5.14.

**Algorithm 1** PRODUTO-MATRICIAL($A$: matriz $m \times l$, $B$: matriz $l \times n$)

```
1: Verifica se colunas de A = linhas de B
2: Cria matriz C vazia de dimensões m × n
3: for i ← 1 to m do
4:     for j ← 1 to n do
5:         c_ij ← 0
6:         for k ← 1 to l do
7:             c_ij ← c_ij + a_ik · b_kj
8:         end for
9:     end for
10: end for
11: return C
```

| | |
|---|---|
| **Entrada** | Matrizes $A$ de dimensões $(m \times l)$ e $B$ de dimensões $(l \times n)$ compatíveis |
| **Saída** | Matriz $C$ de dimensões $(m \times n)$ contendo o produto matricial $A \times B$ |

## 5.10 Requisito 10 - Operação de Produto Escalar

O cálculo do teste de máxima verossimilhança (ver Requisito 5.14) utiliza, nos últimos cômputos, um produto escalar de dois vetores resultantes: um vetor linha e um vetor coluna. Como resultado deste cômputo obtem-se um escalar real que é um valor proporcional à semelhança que um dado vetor $x$ de entrada do *dataset* de verificação possui com as duas classes envolvidas no teste. Neste requisito pede-se que o grupo implemente a operação de produto escalar. Dados duas matrizes $A$ de dimensão $(1 \times n)$ e $B$ de dimensão $(n \times 1)$, obter-se o produto escalar $A \cdot B$. Sugere-se adaptar o código do Algoritmo 1 para operar sobre matrizes linha ou coluna.

| | |
|---|---|
| **Entrada** | Matrizes $A$ de dimensões $(m \times 1)$ e $B$ de dimensões $(1 \times n)$ compatíveis |
| **Saída** | Matriz $C$ de dimensões $(1 \times 1)$ ou escalar real contendo o produto matricial $A \cdot B$ |

## 5.11 Requisito 11 - Solução de Sistemas Lineares por LU

O método de resolução de sistemas lineares conhecido por Decomposição LU (ou Fatoração LU) consiste em um método que decompõe o sistema $\boxed{Ax = b}$ original em um sistema $\boxed{(LU)x = b}$, que pode ser resolvido em duas etapas: na primeira a matriz $A$ é decomposta no produto matricial $LU$, onde $L$ é uma matriz triangular inferior e contêm os multiplicadores usados para pivotear a matriz $A$ e dela extrair a matriz $U$, que é triangular superior e pode ser obtida pelo método da Eliminação de Gauss. De posse de ambas as matrizes, resolve-se o sistema linear $\boxed{Ax = b}$ por meio de duas substituições, uma sucessiva no sistema $\boxed{Ly = b}$ para encontrar o vetor $y$, e outra retroativa no sistema $\boxed{Ux = y}$ para determinar a solução final do sistema. Em termos de esforço computacional, a decomposição LU é útil quando existe a necessidade de se resolver vários sistemas diferentes, cada qual com mesma matriz de coeficientes $A$, mas diferentes termos independentes $b$. Ao decompor a matriz $A$ no produto $LU$ apenas uma vez para todos os sistemas envolvidos economiza-se tempo de computação precioso.

Uma variante mais robusta do método de Decomposição LU é consiste em re-

alizar o pivoteamento da matriz $A$ utilizando trocas de linha que garantam que a diagonal da matriz sob pivoteamento jamais tenham valores nulos (se a matriz for não singular, isto é, se $det(A) \neq 0$). Trata-se da Decomposição LU com pivoteamento parcial. O pseudo-código dado no Algoritmo 2 apresenta em alto nível como realizar o pivoteamento da matriz $A$ no produto $(LU)$ considerando trocas de linha; o Algoritmo 3 realiza substituição sucessiva considerando o pivoteamento parcial de $A = LU$, onde aplica-se na resolução do sistema derivado $\boxed{Ly = Pb}$ onde $P$ é a matriz de pivoteamento que troca as linhas do vetor $b$ de acordo com as trocas ocorridas em $A$; por fim o Algoritmo 4 apresenta pseudo-código para, de posse do vetor $y$ obtido na resolução $\boxed{Ly = Pb}$, obter a solução final $x$ por meio da resolução do sistema triangular $\boxed{Ux = y}$. Os códigos são dados como referência.

**Algorithm 2** AlgoritmoDecomposicaoLU($n$, $A$)

```
1: for (i ← 1 to n) do
2:     Pivot[i] ← i;                                   {Inicialização ordenada de Pivot[]}
3: end for
4: for (j ← 1 to (n − 1)) do
5:     p ← índice linha k do maior elemento |A[k, j]|, j + 1 ≤ k ≤ n        {Escolha do pivô}
6:     if (p ≠ j) then
7:         for (k ← 1 to n) do
8:             t ← A[j, k]; A[j, k] ← A[p, k]; A[p, k] ← t                  {Troca das Linhas p e j}
9:         end for
10:        m ← Pivot[j]; Pivot[j] ← Pivot[p]; Pivot[p] ← m                  {Armazena permutas de b}
11:    end if
12:    if (abs(A[j, j] ≠ 0) then
13:        for (i ← j + 1 to n) do
14:            multiplicador ← A[i, j]/A[j, j]          {Pivoteamento por Eliminação de Gauss}
15:            A[i, j] ← A[i, j] ← multiplicador                {Multiplicadores m_ij de L}
16:            for (k ← j + 1 to n) do
17:                A[i, k] ← A[i, k] − multiplicador ∗ A[j, k]          {Coeficientes u_ij de U}
18:            end for
19:        end for
20:    end if
21: end for
22: return (A, Pivot)                                  {Retorna matriz A = L/U e vetor Pivot[]}
```

**Algorithm 3** AlgoritmoSubstituicoesSucessivasPivotal($n$, $L$, $b$, $Pivot$)

```
1: k ← Pivot[1];                                       {Solução do Sist. Triangular Inferior}
2: y[1] ← b[k]
3: for (i ← 2 to n) do
4:     soma ← 0
5:     for (j ← 1 to (i − 1)) do
6:         soma ← soma + L[i, j] ∗ y[j]
7:     end for
8:     k ← Pivot[i]                                    {Obtém índice de b para o i-ésimo pivô}
9:     y[i] ← b[k] − soma
10: end for
11: return y
```

**Algorithm 4** AlgoritmoSubstituicoesRetroativas($n$, $U$, $d$)

```
1: x[n] ← d[n]/U[n, n]
2: for (i ← (n − 1) to 1) do
3:     soma ← 0
4:     for (j ← (i + 1) to (n)) do
5:         soma ← soma + U[i, j] ∗ x[j]
6:     end for
7:     x[i] ← (d[i] − soma)/U[i, i]
8: end for
9: return x
```

A resolução de sistemas lineares usando Decomposição LU com pivoteamento parcial será de grande utilidade na determinação da matriz $C^{-1}$ que é a matriz inversa da matriz de covariância $C$. Mais detalhes sobre como aplicar os conceitos do método LU para gerar uma matriz inversa é dado no Requisito 5.12.

## 5.12 Requisito 12 - Operação de Inversa de Matrizes

Seja $A$ uma matriz quadrada não singular. A inversa de $A$ (denotada por $A^{-1}$) satisfaz a propriedade $\boxed{A \cdot A^{-1} = I}$, onde $I$ é a matriz identidade. Em notação matricial, a relação acima pode ser dada por

$$\underbrace{\begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{bmatrix}}_{A} \cdot \underbrace{\begin{bmatrix} v_{11} & v_{12} & \dots & v_{1n} \\ v_{21} & v_{22} & \dots & v_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ v_{n1} & v_{n2} & \dots & v_{nn} \end{bmatrix}}_{A^{-1}} = \underbrace{\begin{bmatrix} 1 & 0 & \dots & 0 \\ 0 & 1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 1 \end{bmatrix}}_{I}$$

Seja $v_i$ o $i$-ésimo vetor coluna de $A^{-1}$, e $e_i$ o $i$-ésimo vetor coluna de $I$. Observe que a formação das colunas de $A^{-1}$ pode, portanto, ser feita a partir da resolução de $n$ sistemas lineares diferentes do tipo $\boxed{Av_i = e_i}$:

Sistema 1

$$\underbrace{\begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{bmatrix}}_{A} \cdot \underbrace{\begin{bmatrix} v_{11} \\ v_{21} \\ \vdots \\ v_{n1} \end{bmatrix}}_{v_1} = \underbrace{\begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}}_{e_1}$$

Sistema 2

$$\underbrace{\begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{bmatrix}}_{A} \cdot \underbrace{\begin{bmatrix} v_{12} \\ v_{22} \\ \vdots \\ v_{n2} \end{bmatrix}}_{v_2} = \underbrace{\begin{bmatrix} 0 \\ 1 \\ \vdots \\ 0 \end{bmatrix}}_{e_2}$$

$$\vdots$$

$$\text{Sistema } n \qquad \underbrace{\begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{bmatrix}}_{A} \cdot \underbrace{\begin{bmatrix} v_{1n} \\ v_{2n} \\ \vdots \\ v_{nn} \end{bmatrix}}_{v_1} = \underbrace{\begin{bmatrix} 0 \\ 0 \\ \vdots \\ 1 \end{bmatrix}}_{e_n}$$

Em outras palavras, o $i$-ésimo vetor coluna $v_i$ da matriz $A^{-1}$ pode ser calculado por meio da resolução dos sistemas $Av_i = e_i$, com $A$ e $e_i$ dados. A solução de cada um dos $n$ sistemas lineares apresentados 'gera' uma coluna $v_i$, $i = 1, 2, \dots, n$ da matriz inversa $A^{-1}$. Após as $n$ resoluções, todas as colunas da matriz $A^{-1}$ foram obtidas e a matriz inversa é obtida como resultado. Esta técnica será empregada para, dada a matriz de covariância $C$, obter-se a inversa desta matriz $C^{-1}$. Usando o mesmo conceito acima obtem-se, coluna a coluna, os valores da inversa de $C$. O cálculo desta inversa tem papel fundamental no cômputo do valor do teste de máxima verossimilhança, no Requisito 5.14.

## 5.13 Requisito 13 - Treinamento do Classificador

Neste requisito o grupo deverá implementar o processo de treinamento do classificador, isto é, o processo de gerar os $k$ vetores média, a matriz de covariância $C$, e a inversa da matriz $C^{-1}$. Cada processo já foi definido e explicado, respectivamente, no Requisito 5.4, Requisito 5.5 e Requisito 5.12.

## 5.14 Requisito 14 - Cálculo *Maximum Likelihood Ratio Test*

Neste requisito o grupo deverá calcular o valor do *Maximum Likelihood Ratio Test* dados um vetor de entrad $x$ e duas classes $i, j \in \{0, 1, \dots, (K-1)\}$ segundo fórmulada dada na Equação (3), dada por

$$r_{ij}(x) = \underbrace{x^T C^{-1}(m_i - m_j)}_{(I)} - \underbrace{{}^1\!/_2 (m_i + m_j)^T C^{-1}(m_i - m_j)}_{(II)}$$

Para cada trecho da fórmula deve-se utilizar a implementação feita por vários requisitos de software já descritos. O Requisito 5.6 será usado para transpor o vetor de entrada $x$ que contêm alguma das tuplas do *dataset* de verificação cuja classe deverá ser determinada, e também para transpor o resultado da soma $(m_i + m_j)$; o Requisito 5.8 será usado nos trechos da fórmula que requerem a soma dos vetores média $(m_i + mj)$, ou a diferença dos vetores média $(m_i - m_j)$; o Requisito 5.7 será usado para multiplicar o resultado da transposição da soma dos vetores média pelo escalar ${}^1\!/_2$; o Requisito 5.12 devolverá a inversa de matriz $C$ construída a partir das covariâncias entre as $(V-1)$ variáveis de entrada do *dataset* de treinamento; o Requisito 5.9 será usado para realizar o produto matricial entre a transposta do

vetor de entrada $x^T$ pela inversa da covariância $C^{-1}$ na parte (I) do cálculo, e para realizar o produto matricial entre o resultado $1/2(m_i + m_j)^T$ com a inversa da matriz de covariância $C^{-1}$, na parte (II) do cálculo. Por fim, o Requisito 5.10 será usado para realizar o produto escalar entre o resultado de $x^T C^{-1}$ e $(m_i - m_j)$ na parte (I) do cálculo, e para determinar o produto escalar entre o resultado de $1/2(m_i + m_j)^T C^{-1}$ e $(m_i - m_j)$ na parte (II) do cálculo. Este último cálculo devolve dois valores escalares reais, de onde deve-se fazer a diferença entre ambos para obter o valor de $r_{ij}$ pretendido. A compatibilidade dos cálculos deve ser observada em termos das dimensões de cada matriz/vetor de entrada, e dos resultados intermediários.

## 5.15 Requisito 15 - Classificação de *Dataset*

Neste requisito deve-se calcular, para cada tupla $x$ do *dataset* de verificação, qual é a classe em que o classificador implementado designa a respectiva tupla de entrada. Para tal é preciso gerar todos os resultados possíveis, dois a dois, da máxima verossimilhança $r_{ij}$ para $i \neq j$, com $i, j \in \{0, 1, \ldots, (K-1)\}$.

De posse dos resultados de cada teste atribui-se a tupla à classe $k$ cujo cálculo $r_{kj}$ é o de maior magnitude. Este processo deverá ser repetido para cada uma das tuplas do *dataset* de verificação, contabilizando a quantidade de acertos ou erros ao comparar-se a classificação dada pelo algoritmo com a coluna $V$ do *dataset*, que contêm a classe. Esta estatística será usada adiante na exibição dos resultados finais.

## 5.16 Requisito 16 - Saída de Dados

Neste requisito o grupo deverá implementar a saída da aplicação em console. Após receber os *datasets* informados, gerar os vetores média e matriz de covariância, calcular o resultado do *Maximum Likelihood Ratio Test* e determinar a melhor classificação para cada tupla de entrada do *dataset* de verificação a aplicação deverá gerar alguns resultados em tela, no seguinte formato:

```
Pattern Recognition Application
======= =========== ===========

Input Dataset     : iris-treinamento.dat (15 inputs in 5 columns)
Output Dataset    : iris-verificacao.dat (37 inputs in 5 columns)

Number os Classes : 3

Mean Vectors
==== =======

m_0 = [ 4.86  3.28  1.40  0.20 ]

m_1 = [ 6.46  2.92  4.54  1.44 ]

m_2 = [ 6.64  2.96  5.58  2.04 ]

Covariance Matrix
========== ======
```

```
|  0.96980952 -0.05566667  1.7148571  0.6596667 |
| -0.05566667  0.10980952 -0.2465714 -0.1022381 |
|  1.71485714 -0.24657143  3.5740000  1.4731429 |
|  0.65966667 -0.10223810  1.4731429  0.6406667 |

Summary
=======

# of Hits      : 30
# of Misses    : 7
Percent        : 81.08 %

End of processing
```

São apresentados em console o nome de cada um dos *datasets* informados, a quantidade de tuplas de cada arquivo, a quantidade de variáveis (colunas) de cada arquivo, a quantidade de classes do *dataset* de treinamento, os vetores média de cada classe e a matriz de covariância obtidas durante o treinamento, o número de acertos obtidos na verificação, o número de erros cometidos e, por fim, o percentual de acertos do algoritmo.

## 5.17 Requisito 17 - Modularização, Doc. e Corretude

Neste requisito o grupo deverá garantir que o código fonte gerado na linguagem escolhida foi bem modularizado e documentado. Por **documentação** entende-se, no contexto deste trabalho prático, como a correta inserção de comentários de código no cabeçalho do arquivo, nos cabeçalhos das funções e procedimentos, no corpo das funções e procedimentos e afins de forma a deixar mais fácil o entendimento do código por terceiros. A **modularização** será avaliada considerando como o problema foi decomposto em subproblemas menores, cada qual lidando com apenas uma parte atômica da solução sendo construídos de modo que a junção de cada procedimento ou função implementado faça sentido para completar o objetivo final do trabalho. As funções e procedimentos deverão tratar um único tema, sendo independentes e projetados de acordo com a semântica da especificação. Por fim, a **corretude** é o requisito mais importante do trabalho, e será considerado como uma medida do quão correto são os resultados gerados pela aplicação. A aplicação só irá gerar resultados corretos se cada função ou procedimento implementado também o estiver.

# 6 Barema

O trabalho tem valor de 30 pontos, distribuídos da seguinte maneira:

| **Requisito** | **Pontos** |
|---|---|
| Requisito 01 - Datasets de Classificação | 1.0 |
| Requisito 02 - Entrada de Dados | 2.0 |
| Requisito 03 - Representação de Matrizes | 1.0 |
| Requisito 04 - Cálculo dos Vetores Média | 1.0 |
| Requisito 05 - Cálculo da Matriz de Covariância | 2.0 |
| Requisito 06 - Operação de Transposta de Matriz | 1.0 |
| Requisito 07 - Operação de Produto de Real por Matriz | 0.5 |
| Requisito 08 - Operação de Soma/Diferença de Matrizes | 0.5 |
| Requisito 09 - Operação de Produto Matricial | 1.0 |
| Requisito 10 - Operação de Produto Escalar | 1.0 |
| Requisito 11 - Solução de Sistemas Lineares por LU | 4.0 |
| Requisito 12 - Operação de Inversa de Matrizes | 2.0 |
| Requisito 13 - Treinamento do Classificador | 1.0 |
| Requisito 14 - Cálculo do *Maximum Likelihood Ratio Test* | 3.0 |
| Requisito 15 - Classificação de Dataset | 2.0 |
| Requisito 16 - Saída de Dados | 2.0 |
| Requisito 17 - Modularização, Documentação e Corretude | 5.0 |
| **TOTAL** | **30.0** |