Revista do PICME na UNICAMP

1ª Edição, 2025

IMECC

Conteúdo

Prefácio

É com grande satisfação que apresentamos a primeira edição da Revista do PICME na UNICAMP.

O Programa de Iniciação Científica e Mestrado (PICME) oferece a estudantes universitários(as) que se destacaram nas Olimpíadas de Matemática —medalhistas da OBMEP ou da OBM— a oportunidade de realizar estudos avançados em Matemática simultaneamente com a graduação. Esta publicação reúne os melhores relatórios de iniciação científica de 2024 dentro do programa em nosso instituto, refletindo o talento, a dedicação e o rigor científico de nossos jovens pesquisadores.

Embora sejam reconhecidamente talentosos, a transição para a pesquisa em Matemática pode não ser imediata. Assim, esperamos que esta revista, ao divulgar trabalhos acadêmicos de excelência, sirva de inspiração para futuros estudantes do programa e bolsistas de outras agências.

Consideramos essa revista como um marco importante na valorização da produção de nossos estudantes e da orientação zelosa de nossos(as) docentes. Cada trabalho aqui apresentado representa não apenas um avanço no conhecimento em suas respectivas áreas, mas também o compromisso do instituto com a formação de novas gerações de cientistas.

A seleção dos trabalhos foi desafiadora, dado o alto nível dos relatórios enviados, mas esperamos que esta seja apenas a primeira de muitas edições, fortalecendo a iniciação científica dentro do IMECC. Desejamos a todos(as) uma excelente leitura!

Os Editores

Campinas, fevereiro de 2025

Editores

Daniel Aguilar Gomes

Giuliano Angelo Zugliani, coordenador do PICME-UNICAMP

Neemias Silva Martins

Paulo Lupatini

Apoio: IMECC-UNICAMP

SVMs com Gradiente Estocástico: Teoria e Aplicações em Machine Learning

Resumo

SVMs com Gradiente Estocástico:
Teoria e Aplicações em Machine Learning

Marcos Ferreira Semolini

Orientador: Prof. Christian Horacio Olivera

Este relatório abrange o terceiro semestre do projeto. O foco do estudo foi a aplicação de técnicas de Machine Learning, com ênfase na VC Dimension e em Support Vector Machines (SVMs), utilizando o método de Gradiente Estocástico para otimização da função de perda.

Começamos com uma introdução aos conceitos fundamentais de Machine Learning, seguido pela definição e comprovação da VC Dimension, o que possibilitou a formulação e validação da SVMs.

O método de Gradiente Estocástico foi explorado em conjunto com o Gradiente Descendente, podendo assim compararmos as diferenças dos dois métodos.

Como aplicação prática, desenvolvemos um modelo em Python de classificação para prever a presença de diabetes em pacientes, utilizando um conjunto de dados contendo informações de diversos indivíduos. Através dessa aplicação, foi possível demonstrar a eficácia das técnicas estudadas na solução de problemas reais de classificação, destacando a importância e a aplicabilidade dos métodos de Machine Learning.

Para a realização do projeto, foram utilizadas as bibliografias [1], [2] e [3] para o aprendizado dos tópicos de VC dimension e SVMs. Para a implementação do método de Gradiente Estocástico, utilizamos a bibliografia [4]. Por fim, para obter os dados para a aplicação prática, utilizamos como referência [5].

Como perspectivas futuras do projeto, sabemos que nosso modelo pode ser melhorado a partir da utilização de outras técnicas de Machine Learning, como por exemplo Kernels, que permitiria que nosso modelo se adaptasse a dados não linearmente separáveis.

1. Introdução a Machine Learning

1.1 Definição

Pode-se definir Machine Learning como métodos computacionais que, através de experiências com dados, buscam fazer previsões precisas ou melhorar sua eficiência.

Existe uma confusão comum sobre a diferença entre Inteligência Artificial, Machine Learning e Deep Learning. Inteligência Artificial é um campo no qual sistemas computacionais se tornam capazes de realizar tarefas que normalmente exigiriam inteligência humana, como aprender e raciocinar. Machine Learning, por sua vez, é um subcampo da Inteligência Artificial que permite que computadores aprendam a realizar tarefas específicas sem serem explicitamente programados para isso. E finalmente, Deep Learning é uma técnica de Machine Learning que utiliza redes neurais profundas para processar dados mais complexos e extensos.

Dessa forma, o objetivo de Machine Learning é construir algoritmos de previsão eficientes e precisos. O sucesso desse processo está intrinsecamente ligado aos dados utilizados, fazendo com que Machine Learning esteja relacionado a técnicas de estatística, probabilidade e otimização.

1.2 Método Geral

A estratégia geralmente utilizada para solucionar problemas com Machine Learning é dividir os dados em três partes distintas: amostra de treino, amostra de validação e amostra de teste. Inicialmente, começamos treinando nosso programa com a amostra de treino, na qual já conhecemos os resultados esperados. Durante o treinamento, nosso objetivo é encontrar um algoritmo capaz de generalizar padrões nos dados, de modo que possa fazer previsões precisas em novos conjuntos de dados. Em seguida, validamos o desempenho desse algoritmo utilizando a amostra de validação, a fim de avaliar sua capacidade de generalização e detectar possíveis problemas. Caso identifiquemos um algoritmo com desempenho satisfatório, finalmente testamos sua eficácia utilizando a amostra de teste, a qual nunca foi vista pelo algoritmo durante o treinamento. Essa etapa final nos fornece uma avaliação objetiva do desempenho do modelo em condições do mundo real.

1.3 Cenários Possíveis

Existe uma enorme variedade de problemas que podem ser solucionados utilizando Machine Learning. Entre os mais comuns estão os de classificação, regressão e ranking.

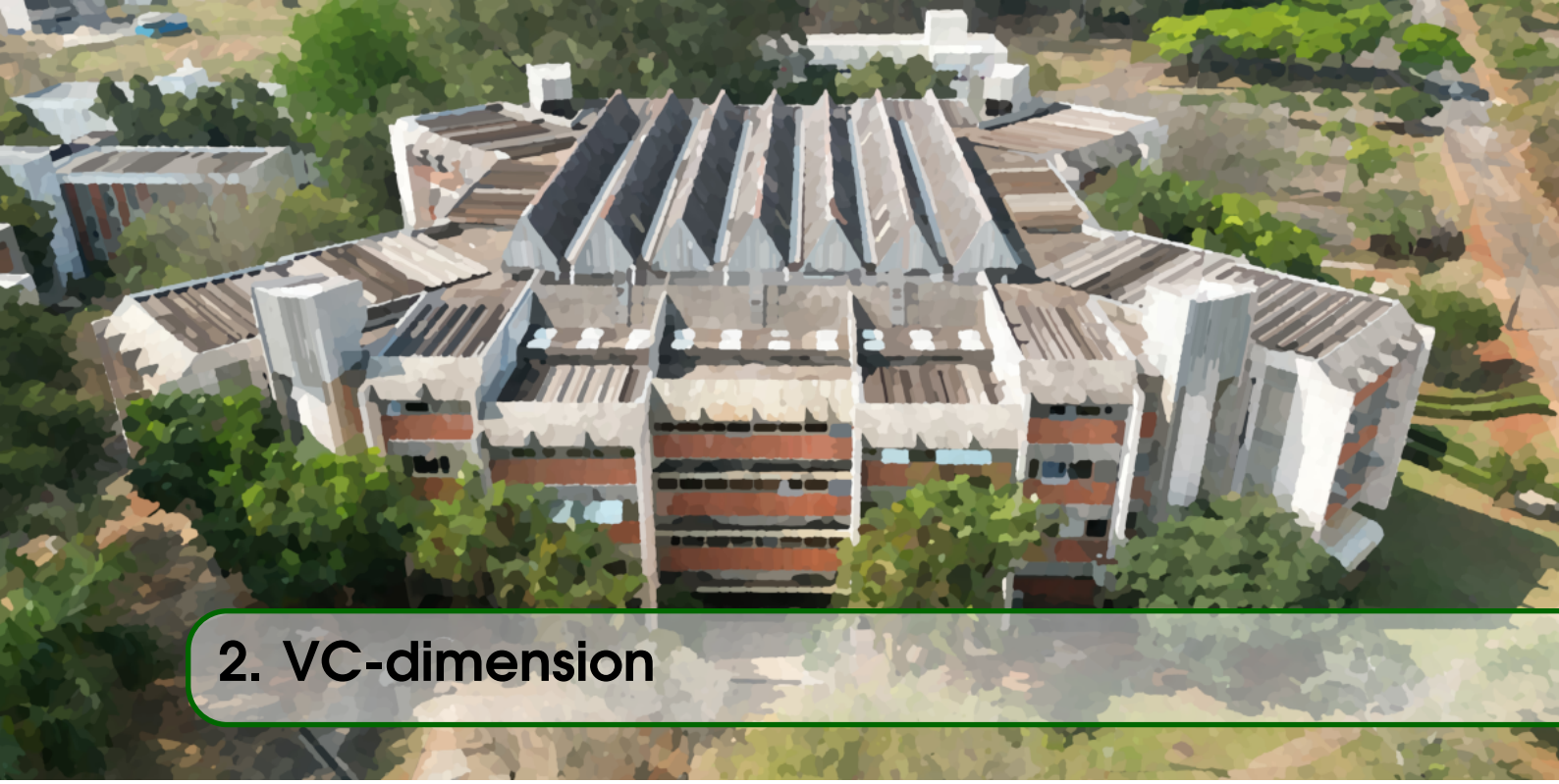
Dentre os problemas que queremos resolver, existem diversos cenários possíveis. Os mais comuns incluem a Aprendizagem Supervisionada, na qual possuímos todos os rótulos de nossos dados; a Aprendizagem Não Supervisionada, na qual não temos rótulos para nenhum dado; e a Aprendizagem Semi-Supervisionada, na qual temos rótulos apenas para alguns dados.

Neste projeto, para testar nosso algoritmo de Machine Learning, utilizaremos apenas a Aprendizagem Supervisionada em uma situação de classificação binária.

1.4 Generalização

Um dos princípios mais importantes de Machine Learning é a generalização. É necessário que nosso algoritmo seja suficientemente genérico para conseguir prever dados que não foram previamente observados. No entanto, ao mesmo tempo, desejamos evitar que ele seja excessivamente genérico a ponto de comprometer a precisão dos resultados. O desafio reside em encontrar um equilíbrio entre a complexidade do aprendizado que nos permita atender a essas duas necessidades simultaneamente.

Caso nosso algoritmo fique muito genérico, dizemos que ocorreu um underfitting, e caso ele fique específico demais, dizemos que ocorreu um overfitting.



2. VC-dimension

2.1 Introdução

Neste capítulo, exploraremos a dimensão VC (Vapnik-Chervonenkis), um conceito fundamental em teoria da aprendizagem estatística. A dimensão VC é uma medida da capacidade de um modelo de aprendizado em termos de sua complexidade e poder de generalização. Em essência, a dimensão VC nos ajuda a entender até que ponto um conjunto de funções pode classificar corretamente qualquer conjunto de dados, independentemente de sua distribuição.

Para chegarmos à definição formal da dimensão VC, é necessário abordar conceitos importantes como a complexidade de Rademacher e a função de crescimento (growth function). Com a definição da dimensão VC, poderemos discutir seu papel na obtenção de limites de erro generalizados. Esses limites são cruciais para garantir que um modelo de aprendizado não apenas se ajuste bem aos dados de treinamento, mas também tenha um bom desempenho em dados não vistos.

Daremos um foco especial aos hiperplanos, uma vez que são centrais para o funcionamento das SVMs (Capítulo 3). Discutiremos como a dimensão VC se aplica aos hiperplanos e como isso influencia o desempenho das SVMs.

Alguns conceitos básicos são importantes serem definidos desde o início, como o erro generalizado e o erro empírico. Dada uma hipótese $h \in H$, um conceito $c \in C$, uma distribuição D e uma amostra $S = (x_1, \ldots, x_m)$, o erro ou risco generalizado é definido como:

$$R(h) = \Pr_{(x \sim D)}[h(x) \neq c(x)] = \mathbb{E}_{(x \sim D)}[1_{h(x) \neq c(x)}] \tag{2.1}$$

e o erro empírico, por sua vez, é definido como:

$$\hat{R}_S(h) = \frac{1}{m} \sum_{i=1}^{m} 1_{h(x_i) \neq c(x_i)}. \tag{2.2}$$

2.2 Complexidade de Rademacher

Usaremos H para denotar um conjunto de hipóteses, L para uma função de perda arbitrária que mapeia $Y \times Y \to \mathbb{R}$ e G como a família de funções de perda associadas a H que mapeiam de $Z = X \times Y$ para $\mathbb{R}$.

Definição da Complexidade empírica de Rademacher

Seja $S = (z_1, \ldots, z_m)$ uma amostra fixa de tamanho m. Definiremos a complexidade empírica de Rademacher de G com relação à amostra S como

$$\hat{\mathfrak{R}}_S(G) = \mathbb{E}_\sigma \left[\sup_{g \in G} \frac{1}{m} \sum_{i=1}^{m} \sigma_i g(z_i) \right], \tag{2.3}$$

onde σ é uma variável de Rademacher tomando valores em $\{-1, +1\}$.

A complexidade empírica de Rademacher mede, em média, quão bem a classe de funções G se correlaciona com ruído aleatório em S.

Complexidade de Rademacher

Seja D uma distribuição. Para qualquer $m \geq 1$, a complexidade de Rademacher de G é a expectativa da complexidade empírica de Rademacher:

$$\mathfrak{R}_m(G) = \mathbb{E}_{S \sim D^m} \left[\hat{\mathfrak{R}}_S(G) \right]. \tag{2.4}$$

Teorema

Seja G uma família de funções que mapeiam em $[0, 1]$. Então, para qualquer $\delta > 0$ com probabilidade pelo menos $1 - \delta$ em uma amostra i.i.d S de tamanho m temos:

$$\mathbb{E}[g(z)] \leq \frac{1}{m} \sum_{i=1}^{m} g(z_i) + 2\mathfrak{R}_m(G) + \sqrt{\frac{\log \frac{1}{\delta}}{2m}} \tag{2.5}$$

$$\mathbb{E}[g(z)] \leq \frac{1}{m} \sum_{i=1}^{m} g(z_i) + 2\hat{\mathfrak{R}}_S(G) + 3\sqrt{\frac{\log \frac{2}{\delta}}{2m}}. \tag{2.6}$$

Demonstração:

A ideia consiste em aplicar a Inequação de McDiarmid para a função Φ definida para qualquer amostra S por:

$$\Phi(S) = \sup_{g \in G} \left(\mathbb{E}[g] - \hat{\mathbb{E}}_S[g] \right), \tag{2.7}$$

$$\text{onde } \Phi(S') - \Phi(S) \leq \sup_{g \in G} \left(\hat{\mathbb{E}}_S[g] - \hat{\mathbb{E}}_{S'}[g] \right) = \sup_{g \in G} \frac{g(z_m) - g(z_m')}{m} \leq \frac{1}{m}. \tag{2.8}$$

Utilizando a inequação de Mcdiarmid

$$\Phi(S) \leq \mathbb{E}_S[\Phi(S)] + \sqrt{\frac{\log \frac{2}{\delta}}{2m}}, \tag{2.9}$$

$$\text{que é equivalente a } \Phi(S) \leq 2\mathfrak{R}_m(G) + \sqrt{\frac{\log \frac{2}{\delta}}{2m}}. \tag{2.10}$$

Lema

Seja H uma família de funções com valores em $\{-1,+1\}$ e seja G a família de funções de perda associadas a H para a perda de zero-um. Para qualquer amostra $S = ((x_1,y_1),\ldots,(x_m,y_m))$, seja S_x a sua projeção sobre X. Então vale:

$$\hat{\mathfrak{R}}_s(G) = \frac{1}{2}\hat{\mathfrak{R}}_{S_x}(H). \tag{2.11}$$

Usando o Teorema e o Lema apresentados, podemos obter limitações da diferença do erro generalizado e o erro empírico usando a complexidade de Rademacher.

Limites para classificação binária com a complexidade de Rademacher

Seja H uma família de funções com valores em $\{-1,+1\}$ distribuídas de acordo com D sobre o espaço X. Então, para qualquer $\delta \geq 0$ com probabilidade pelo menos $1-\delta$ sobre uma amostra S de tamanho m, temos:

$$R(h) \leq \hat{R}_s(h) + \mathfrak{R}_m(H) + \sqrt{\frac{\log\frac{1}{\delta}}{2m}}, \tag{2.12}$$

$$R(h) \leq \hat{R}_s(h) + \hat{\mathfrak{R}}_s(H) + 3\sqrt{\frac{\log\frac{2}{\delta}}{2m}}. \tag{2.13}$$

Note que calcular $\hat{\mathfrak{R}}_s(H)$ pode ser computacionalmente difícil, por conta disso, veremos a seguir métodos mais fáceis de se calcular.

2.3 Função de Crescimento (Growth Function)

Agora mostraremos que a complexidade de Rademacher pode ser limitada em termos da função de crescimento.

Definição da Função de Crescimento

Definimos a função de crescimento Π_H como:

$$\Pi_H(m) = \max_{\{(x_1,\ldots,x_m)\}\in X} |\{(h(x_1),\ldots,h(x_m)) : h \in H\}|. \tag{2.14}$$

Utilizando o Lema de Massart, que diz que

$$\mathbb{E}_\sigma\left[\frac{1}{m}\sup\sum_{i=1}^{m}\sigma_i x_i\right] \leq \frac{r\sqrt{2\log A}}{m} \tag{2.15}$$

podemos limitar a complexidade de Rademacher em termos da função de crescimento.

Corolário

Seja G uma família de funções com valores em $\{-1,+1\}$, então temos que

$$\mathfrak{R}_m(G) \leq \sqrt{\frac{2\log\Pi_g(m)}{m}}. \tag{2.16}$$

Utilizando as Equações 2.12 e 2.16, podemos então obter um limite generalizado em termos da função de crescimento.

Limite de Generalização com Função de Crescimento

Seja H uma família de funções com valores em $\{-1,+1\}$. Então, para qualquer $\delta \geq 0$ com probabilidade pelo menos $1-\delta$,

$$R(h) \leq \hat{R}_s(h) + \sqrt{\frac{2\log \Pi_H(m)}{m}} + \sqrt{\frac{\log \frac{1}{\delta}}{2m}}. \tag{2.17}$$

A desvantagem do cálculo da função de crescimento é que, por definição, requer o cálculo de $\Pi_H(m)$ para todo $m \geq 1$. Por isso, introduziremos uma medida de complexidade mais interessante.

2.4 Dimensão-VC

Introduziremos agora a noção de Dimensão-VC. Frequentemente ela é mais fácil de ser calculada, em comparação com a função de crescimento ou a complexidade de Rademacher, apesar de também possuir uma noção puramente combinatorial.

Devemos lembrar que uma dicotomia de um conjunto S é uma das possíveis maneiras de rotular os seus pontos usando uma hipótese em H. Um conjunto S de $m \geq 1$ pontos é dito ser fragmentado por um conjunto de hipóteses H quando H realiza todas as possíveis dicotomias de S, isso ocorre quando $\Pi_H(m) = 2^m$.

Definição de Dimensão-VC

A dimensão VC de um conjunto de hipóteses H é o tamanho do maior conjunto que pode ser fragmentado por H:

$$\text{VCdim}(H) = \max\{m : \Pi_H(m) = 2^m\}. \tag{2.18}$$

Exemplo no hiperplano

Considere hiperplanos no $\mathbb{R}^2$. Digamos que temos 3 pontos. O número de dicotomias possíveis será $2^3 = 8$. Pela imagem 2.1, retirada da bibliografia [2], podemos observar que todas as dicotomias puderam ser separadas.

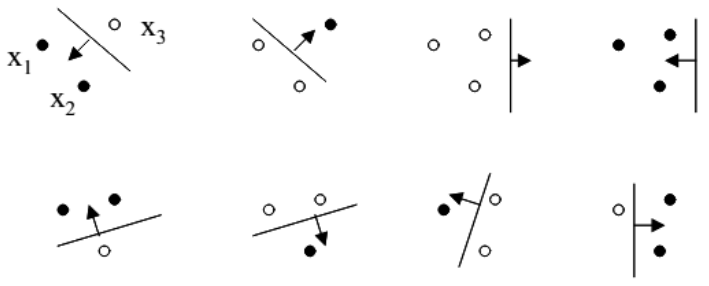


Figura 2.1: Três pontos no $\mathbb{R}^2$ separados por hiperplanos.

Já se tentarmos fazer o mesmo com 4 pontos, pela imagem 2.2, não conseguiremos separar todas as dicotomias. Concluímos, assim, que a dimensão VC dos hiperplanos no $\mathbb{R}^2$ é igual a 3, ou seja, VCdim(hiperplanos no $\mathbb{R}^2$) $= 3$.

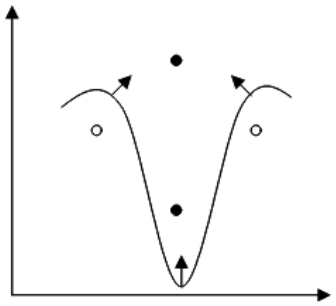


Figura 2.2: Quatro pontos no $\mathbb{R}^2$ que não podem ser separados por hiperplanos.

Generalizando para o $\mathbb{R}^N$, encontraremos que a dimensão VC para o hiperplano é dada por:

$$VCdim(\text{hiperplanos no } \mathbb{R}^N) = N + 1. \tag{2.19}$$

Utilizaremos agora o lema de Sauer para termos uma conexão entre a noção de função de crescimento e Dimensão-VC.

Lema de Sauer

Seja H um conjunto de hipóteses com $\text{VCdim}(H) = d$. Então, para todo $m \in \mathbb{N}$, a seguinte desigualdade vale:

$$\Pi_H(m) \leq \sum_{i=0}^{d} \binom{m}{i}. \tag{2.20}$$

Segue desse lema o seguinte corolário:
Seja H um conjunto de hipóteses com $\text{VCdim}(H) = d$. Então, para todo $m \geq d$,

$$\Pi_H(m) \leq (\frac{em}{d})^d. \tag{2.21}$$

Com isso conseguimos limites de generalização baseados na dimensão-VC.

Limites de Generalização com Dimensão-VC

Seja H uma família de funções que tomam valores em {-1, +1} com dimensão-VC $= d$. Então, para qualquer $\delta > 0$, com probabilidade pelo menos $1 - \delta$, o seguinte vale para todo $h \in H$:

$$R(h) \leq \hat{R}_s(h) + \sqrt{\frac{2d \log\left(\frac{em}{d}\right)}{m}} + \sqrt{\frac{\log\left(\frac{1}{\delta}\right)}{2m}}. \tag{2.22}$$

3. Support Vector Machines

3.1 Introdução

Support Vector Machines (SVM) é uma técnica de aprendizado de máquina introduzida por Vladimir Vapnik em 1992. Tem como fundamento o princípio da minimização do risco estrutural, e como objetivo principal encontrar o hiperplano que melhor separa as classes de dados, maximizando a margem entre elas. Essa abordagem garante uma melhor generalização do modelo a novos dados.

O SVM está baseada no fato de que o erro de generalização pode ser limitado pelo erro de treinamento (erro empírico) mais um termo que depende da dimensão VC (capítulo 2). Isso permite que o SVM lide eficientemente com dados de alta dimensão, tornando-se uma boa ferramenta para problemas de classificação e regressão.

Nesse capítulo vamos analisar os dois casos que encontraremos ao utilizar o SVM: os casos de classificação linearmente separáveis e os não linearmente separáveis. Daremos o foco principal no segundo caso, visto que é o mais realista. Depois deduziremos um limite para o erro de generalização do algoritmo.

3.2 Conceitos importantes

Consideraremos um espaço de entrada $X \subseteq \mathbb{R}^N$ com $N \geq 1$ e um espaço de saída $Y = \{-1, +1\}$, classificação binária. A função alvo f é desconhecida e uma amostra de treinamento S de tamanho m é extraída de X de forma i.i.d. segundo uma distribuição desconhecida D. O objetivo é encontrar uma hipótese $h \in H$ que minimize o erro de generalização:

$$R_D(h) = P_{x \sim D}[h(x) \neq f(x)].$$

Classificação Linear

Um classificador linear, ou hiperplano, é definido como:

$$H = \{\mathbf{x} \mapsto \text{sign}(\mathbf{w} \cdot \mathbf{x} + b) : \ \mathbf{w} \in \mathbb{R}^N, b \in \mathbb{R}\}.$$

Neste caso, o problema de aprendizagem é chamado de problema de classificação linear, onde a hipótese $\mathbf{x} \mapsto \text{sign}(\mathbf{w} \cdot \mathbf{x} + b)$ rotula positivamente os pontos de um lado do hiperplano $\mathbf{w} \cdot \mathbf{x} + b = 0$ e negativamente os do outro lado.

Margem geométrica

A margem geométrica $\rho_h(x)$ de um classificador linear $h : \mathbf{x} \mapsto \mathbf{w} \cdot \mathbf{x} + b$ em um ponto é a sua distância Euclidiana até o hiperplano:

$$\rho_h(x) = \frac{|\mathbf{w} \cdot \mathbf{x} + b|}{\|\mathbf{w}\|_2}.$$

A margem geométrica ρ_h de um classificador linear h é a mínima margem geométrica sobre os pontos da amostra, ou seja, é a distância do hiperplano que define h aos pontos mais próximos da amostra.

A solução do SVM é o hiperplano separador que maximiza a margem geométrica. A figura a seguir retirada do livro [1] ilustra esse hiperplano.

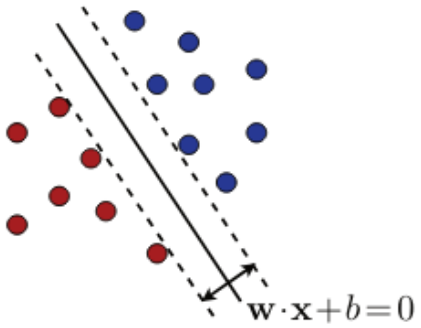


Figura 3.1: Hiperplano separador com máxima margem geométrica.

3.3 Caso linearmente separável

Assumimos que a amostra de treinamento S pode ser linearmente separada, ou seja, existe um hiperplano que separa perfeitamente a amostra de treinamento em pontos rotulados positivamente e negativamente. Isso é equivalente à existência de $(\mathbf{w}, b)$ tal que:

$$\forall i \in [m], y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 0.$$

Como pode ser analisado, existem infinitos hiperplanos separadores dessa forma. A definição da solução do SVM é o hiperplano separador que maximiza a margem geométrica.

Dessa forma, a máxima margem geométrica ρ de um hiperplano separador é dado por:

$$\rho = \max_{\mathbf{w},b:y_i(\mathbf{w}\cdot\mathbf{x}_i+b)\geq 0} \min_{i\in[m]} \frac{|\mathbf{w} \cdot \mathbf{x}_i + b|}{\|\mathbf{w}\|} = \max_{\mathbf{w},b} \min_{i\in[m]} y_i \frac{\mathbf{w} \cdot \mathbf{x}_i + b}{\|\mathbf{w}\|}.$$

A segunda igualdade vale pelo fato de que para o máximo par (w, b), $y_i(w \cdot x_i + b)$ será não negativo para qualquer $i \in [m]$.

Podemos restringir o par $(\mathbf{w}, b)$ de forma que $\min_{i\in[m]} y_i(\mathbf{w} \cdot \mathbf{x}_i + b) = 1$, obtendo:

$$\rho = \max_{\mathbf{w},b:\min_{i\in[m]} y_i(\mathbf{w}\cdot\mathbf{x}_i+b)=1} \frac{1}{\|\mathbf{w}\|} = \max_{\mathbf{w},b:\forall i\in[m], y_i(\mathbf{w}\cdot\mathbf{x}_i+b)\geq 1} \frac{1}{\|\mathbf{w}\|}.$$

Como maximizar $\frac{1}{\|\mathbf{w}\|}$ é equivalente a minimizar $\frac{1}{2}\|\mathbf{w}\|^2$, o par $(\mathbf{w}, b)$ retornado pelo SVM será a solução do problema de otimização:

$$\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2 \tag{3.1}$$

$$\text{sujeito a: } y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1, \forall i \in [m].$$

Note que, pela função $F : \mathbf{w} \mapsto \frac{1}{2}\|\mathbf{w}\|^2$ ser estritamente convexa, uma vez que $\nabla^2 F(\mathbf{w}) > 0$, e as restrições serem funções afins, o problema de otimização admite uma solução única.

Esse problema de otimização pode ser resolvido utilizando o método dos multiplicadores de Lagrange, uma abordagem que será empregada no próximo caso por ser mais eficaz na prática.

3.4 Caso não linearmente separável

Em muitos casos na prática os dados de treinamento não são linearmente separáveis ou, mesmo que sejam, desejamos evitar o overfitting em nosso algoritmo, conforme discutido no capítulo 1. Por conta disso, implementaremos uma versão relaxada da restrição introduzindo uma nova variável $\xi_i \geq 0$ para cada x_i tal que:

$$y_i[\mathbf{w} \cdot \mathbf{x}_i + b] \geq 1 - \xi_i.$$

Essa variável ξ_i mede a distância pela qual um ponto x_i viola a desigualdade $y_i[\mathbf{w} \cdot \mathbf{x}_i + b] \geq 1$. Se $\xi_i \geq 1$, o ponto foi classificado incorretamente; se $0 < \xi_i < 1$, o ponto foi classificado corretamente, mas violou a margem; e, por fim, se $\xi_i \leq 1$, o ponto foi classificado corretamente e está obedecendo a margem. Essa interpretação pode ser visualizada na figura a seguir retirada do livro [1].

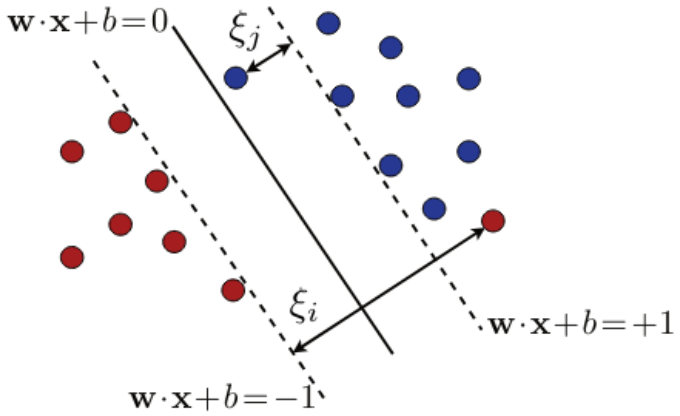


Figura 3.2: Ilustração do caso não linearmente separável com a variável ξ_i.

Desse modo, o par $(\mathbf{w}, b)$ retornado pelo SVM será a solução do problema de otimização:

$$\min_{\mathbf{w},b,\xi} \left(\frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{m} \xi_i \right), \tag{3.2}$$

$$\text{sujeito a: } y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1 - \xi_i \text{ e } \xi_i \geq 0, i \in [m].$$

Onde o parâmetro $C \geq 0$ determina o trade-off entre a maximização da margem e a minimização dos erros ξ_i. Quanto maior for o valor do parâmetro C, mais importância será dada aos erros, ou seja, menos generalizada será a nossa solução.

Chamamos esse problema de otimização de forma Primal. Note que ele adimite solução única, uma vez que a função $F : \mathbf{w}, \xi_i \mapsto \left(\frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{m} \xi_i\right)$ é convexa.

Podemos resolver esse problema de otimização utilizando o método dos multiplicadores de Lagrange

$$L(\mathbf{w}, b, \xi, \alpha, \beta) = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{m} \xi_i - \sum_{i=1}^{m} \alpha_i [y_i(\mathbf{w} \cdot \mathbf{x}_i + b) - 1 + \xi_i] - \sum_{i=1}^{m} \beta_i \xi_i. \tag{3.3}$$

Onde α_i e β_i são as variáveis de Lagrange, a primeira sendo associada à restrição $y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1 - \xi_i$ e a segunda associada à restrição $\xi_i \geq 0$. Calculando os gradientes, obtemos:

$$\nabla_w L = w - \sum_{i=1}^{m} \alpha_i y_i x_i = 0 \implies w = \sum_{i=1}^{m} \alpha_i y_i x_i \tag{3.4}$$

$$\nabla_b L = -\sum_{i=1}^{m} \alpha_i y_i = 0 \implies \sum_{i=1}^{m} \alpha_i y_i = 0 \tag{3.5}$$

$$\nabla_{\xi_i} L = C - \alpha_i - \beta_i = 0 \implies \alpha_i + \beta_i = C \tag{3.6}$$

Podemos derivar a forma Dual do problema de otimização 3.2 substituindo 3.4 e 3.5 em 3.3, assim obtemos:

$$\begin{aligned} D(\xi,\alpha,\beta) = \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m} y_i y_j \alpha_i \alpha_j \langle x_i, x_j\rangle + C\sum_{i=1}^{m}\xi_i - \sum_{i=1}^{m}\alpha_i y_i \left\langle \sum_{j=1}^{m} y_j \alpha_j x_j x_i \right\rangle \\ \underbrace{-b\sum_{i=1}^{m} y_i \alpha_i}_{0} + \sum_{i=1}^{m}\alpha_i - \sum_{i=1}^{m}\alpha_i \xi_i - \sum_{i=1}^{m}\beta_i \xi_i \end{aligned} \tag{3.7}$$

$$\begin{aligned} D(\xi,\alpha,\beta) = \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m} y_i y_j \alpha_i \alpha_j \langle x_i, x_j\rangle + C\sum_{i=1}^{m}\xi_i - \sum_{i=1}^{m}\alpha_i y_i \left\langle \sum_{j=1}^{m} y_j \alpha_j x_j x_i \right\rangle \\ + \sum_{i=1}^{m}\alpha_i - \sum_{i=1}^{m}\alpha_i \xi_i - \sum_{i=1}^{m}\beta_i \xi_i. \end{aligned} \tag{3.8}$$

Agrupando o termos e usando 3.6 temos:

$$D(\xi,\alpha,\beta) = -\frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m} y_i y_j \alpha_i \alpha_j \langle x_i, x_j\rangle + \underbrace{\sum_{i=1}^{m}\xi_i(C - \alpha_i - \beta_i)}_{0} + \sum_{i=1}^{m}\alpha_i \tag{3.9}$$

$$D(\xi,\alpha,\beta) = -\frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m} y_i y_j \alpha_i \alpha_j \langle x_i, x_j\rangle + \sum_{i=1}^{m}\alpha_i \tag{3.10}$$

Além de $\alpha_i \geq 0$, devemos restringir $\beta_i \geq 0$. Assim, por 3.6, devemos ter $\alpha_i \leq C$. Portanto, chegamos ao problema de otimização dual para SVMs no caso separável:

$$\max_{\alpha} \quad \sum_{i=1}^{m}\alpha_i - \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m}\alpha_i \alpha_j y_i y_j \langle x_i, x_j\rangle \tag{3.11}$$

$$\text{sujeito a:} \quad 0 \leq \alpha_i \leq C, \quad \forall i \in [m]; \quad \sum_{i=1}^{m}\alpha_i y_i = 0$$

Se tivéssemos aplicado esse mesmo método no caso linearmente separável, teríamos chegado à mesma equação, porém com uma restrição menos rígida de $\alpha_i \geq 0$.

Pelo mesmo raciocínio dos problemas anteriores, como esse problema de maximização é convexo, ele admite uma solução única.

Preferir o problema Dual em vez do Primal se deve ao fato de ser computacionalmente mais fácil, além do fato de que ele depende de $\langle x_i, x_j\rangle$, o que é particularmente interessante quando discutimos sobre Kernels.

A solução para $\boldsymbol{\alpha}$ do problema 3.11 pode ser utilizada para encontrar o par $(\mathbf{w}^*, b^*)$ do hiperplano separador usando a equação 3.4, uma vez que os problemas primal e dual são equivalentes.

$$\mathbf{w}^* = \sum_{i=1}^{m}\alpha_i y_i x_i$$

$$b^* = y_i - \mathbf{w}^* \cdot x_i$$

Com isso, podemos concluir que a decisão feita pelo SVM será:

$$h(\mathbf{x}) = \operatorname{sgn}\left(\sum_{i=1}^{m}\alpha_i y_i (\mathbf{x}_i \cdot \mathbf{x}) + b^*\right).$$

3.5 Teoria das Margens

Visto o método de SVM acima, é interessante mostrarmos que seu erro de generalização pode ser limitado, oferecendo assim uma forte justificativa para o algoritmo.

Pela equação 2.19, que fornece a dimensão VC do hiperplano no $\mathbb{R}^N$, e pela equação 2.22, que fornece o limite com a dimensão VC, podemos gerar um limite para o erro generalizado do SVM, uma vez que este consiste em um problema de hiperplanos. Dessa forma, para qualquer $\delta > 0$, com uma probabilidade de pelo menos $1 - \delta$ e para qualquer hipótese h, temos:

$$R(h) \leq \hat{R}_S(h) + \sqrt{\frac{2(N+1)\log\frac{em}{N+1}}{m}} + \sqrt{\frac{\log\frac{1}{\delta}}{2m}}. \tag{3.12}$$

Essa inequação já poderia ser considerada um limite para o erro do método. No entanto, para valores de N muito grandes, ela acaba sendo um pouco desinformativa. Por conta disso, vamos gerar outro limite. Para isso, introduziremos algumas definições.

Definiremos uma função de perda de margem ρ que penaliza h com o custo de 1 quando ele classifica incorretamente o ponto x ($yh(x) \leq 0$), mas também penaliza h quando ele classifica corretamente x com confiança menor ou igual a ρ ($yh(x) \leq \rho$), ou seja:

Função de erro da margem

$$\Phi_\rho(x) = \min\left(1, \max\left(0, 1 - \frac{x}{\rho}\right)\right) = \begin{cases} 1 & \text{se } x \leq 0, \\ 1 - \frac{x}{\rho} & \text{se } 0 \leq x \leq \rho, \\ 0 & \text{se } x \geq \rho. \end{cases} \tag{3.13}$$

O erro empirico da margem será então:

$$\hat{R}_{S,\rho}(h) = \frac{1}{m}\sum_{i=1}^{m} \Phi_\rho(y_i h(x_i)) \leq \frac{1}{m}\sum_{i=1}^{m} 1_{y_i h(x_i) \leq \rho}.$$

Lema de Talagrand

Seja Φ funções l-Lipschitz, então a inequação é válida:

$$\hat{\mathfrak{R}}_S(\Phi \circ H) \leq l\,\hat{\mathfrak{R}}_S(H).$$

Limite para a margem em classificação binária

Utilizando o teorema 2.16 e o Lema de Talagrand, e considerando que Φ_ρ é uma função 1/p-Lipschitz, podemos realizar algumas manipulações matemáticas para obter o seguinte limite para a margem:

$$R(h) \leq \hat{R}_{S,\rho}(h) + \frac{2}{\rho}\mathfrak{R}_m(H) + \sqrt{\frac{\log\frac{1}{\delta}}{2m}}, \tag{3.14}$$

$$R(h) \leq \hat{R}_{S,\rho}(h) + \frac{2}{\rho}\hat{\mathfrak{R}}_S(H) + 3\sqrt{\frac{\log\frac{2}{\delta}}{2m}}. \tag{3.15}$$

Deve-se notar que chegamos a um trade-off: quanto maior for a margem ρ, menor será o segundo termo da inequação, mas, em contrapartida, o erro empírico será maior. O limite pode ser generalizado para considerar uniformemente todas as margens $\rho \in (0, r]$ ao adicionar alguns termos, como mostrado nas equações a seguir.

$$R(h) \leq \hat{R}_{S,\rho}(h) + \frac{4}{\rho}\mathfrak{R}_m(H) + \sqrt{\frac{\log\log_2\frac{2r}{\rho}}{m}} + \sqrt{\frac{\log\frac{2}{\delta}}{2m}} \tag{3.16}$$

$$R(h) \leq \hat{R}_{S,\rho}(h) + \frac{4}{\rho}\hat{\mathfrak{R}}_S(H) + \sqrt{\frac{\log\log_2 \frac{2r}{\rho}}{m}} + 3\sqrt{\frac{\log \frac{4}{\delta}}{2m}} \tag{3.17}$$

Deve-se notar que se definindo $\|\mathbf{x}\| \leq r$ e $||\mathbf{w}|| \leq \Lambda$, o erro empírico da complexidade de Rademacher pode ser limitada como segue:

$$\hat{\mathfrak{R}}_S(H) \leq \sqrt{\frac{r^2\Lambda^2}{m}}. \tag{3.18}$$

Combinando isso com a equação 3.16 e escolhendo $\Lambda = 1$ chegamos em:

$$R(h) \leq \hat{R}_{S,\rho}(h) + 4\sqrt{\frac{r^2}{\rho^2 m}} + \sqrt{\frac{\log\log_2 \frac{2r}{\rho}}{m}} + \sqrt{\frac{\log \frac{2}{\delta}}{2m}}. \tag{3.19}$$

Portanto, chegamos que, com uma probabilidade de pelo menos $1-\delta$, é valido para qualquer $h \in \{\mathbf{x} \mapsto \mathbf{w}\cdot\mathbf{x} \mid \|\mathbf{w}\| \leq 1\}$ e qualquer $\rho > 0$:

$$R(h) \leq \frac{1}{m}\sum_{i=1}^{m} \max\left(0, 1 - y_i(\mathbf{w}\cdot\mathbf{x}_i)\right) + 4\sqrt{\frac{r^2}{\rho^2 m}} + \sqrt{\frac{\log\log_2 \frac{2r}{\rho}}{m}} + \sqrt{\frac{\log \frac{2}{\delta}}{2m}}. \tag{3.20}$$

4. Método do Gradiente Estocástico

4.1 Introdução

Para realizar a tarefa de encontrar os parâmetros do modelo que minimizam a função de perca são utilizados algoritmos de otimização, sendo o Gradiente Descendente um dos mais fundamentais.

O Gradiente Descendente é um método iterativo que ajusta os parâmetros do modelo na direção oposta ao gradiente da função de custo, com o objetivo de encontrar um mínimo local ou global. No entanto, para conjuntos de dados muito grandes, a computação do gradiente pode se tornar proibitivamente cara, pois envolve a avaliação da função de custo para todo o conjunto de dados em cada iteração.

Para contornar essa limitação, surge o Método do Gradiente Estocástico, uma variação do Gradiente Descendente que introduz estocasticidade no processo de otimização. Em vez de calcular o gradiente em relação a todo o conjunto de dados, ele faz isso utilizando apenas um pequeno subconjunto de exemplos de treinamento em cada iteração. Isso resulta em uma redução significativa no custo computacional, permitindo atualizações mais frequentes dos parâmetros e uma convergência mais rápida, embora mais ruidosa.

Esse método é muito utilizado quando temos grandes conjuntos de dados, mais frequente quando trabalhamos com Redes Neurais e Deep Learning.

4.2 Gradiente Descendente

O Gradiente Descendente é um algoritmo que mede o gradiente local da função de perda em relação a um parâmetro θ e ajusta θ na direção oposta ao gradiente. Quando o gradiente é zero, isso indica que atingimos um mínimo. O processo começa com a escolha de um θ inicial aleatório e, iteração após iteração, o algoritmo ajusta θ de forma a minimizar a função de perda. Esse processo é ilustrado na Figura 4.1, retirada da bibliografia [4], abaixo.

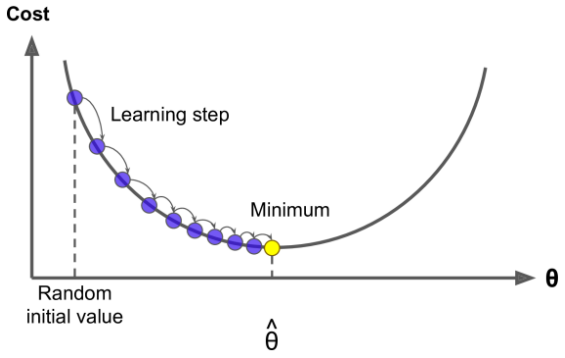


Figura 4.1: Gradiente descendente.

O tamanho de cada passo nas iterações é definido por um parâmetro chamado taxa de aprendizado. A taxa de aprendizado determina a magnitude das atualizações de θ em cada iteração, influenciando a velocidade e a estabilidade da convergência do algoritmo.

A função de perda pode ser definida como

$$J(\theta) = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2, \tag{4.1}$$

onde $h_\theta(x^{(i)})$ é o valor previsto pelo nosso modelo e $y^{(i)}$ seu verdadeiro valor.

Devemos então aplicar o gradiente parcial na função J, com relação a todos os parâmetros de interesse $(\theta_0, \theta_1, ..., \theta_n)$, encontrando assim as equações:

$$\frac{\partial J(\theta)}{\partial \theta_j} = \frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})j = 0, 1, ..., n. \tag{4.2}$$

Assim, os parâmetros serão ajustados de acordo com a equação a seguir, onde η é o parâmetro chamado de taxa de aprendizado já discutido

$$\theta_{k+1} = \theta_k - \eta\nabla_\theta J(\theta). \tag{4.3}$$

Podemos então fazer quantas interações forem necessárias para chegar a um resultado considerado satisfatório.

4.3 Gradiente Estocástico

O Gradiente Estocástico segue a mesma lógica do Gradiente Descendente, mas, em vez de usarmos todos os dados para calcular o gradiente, utilizamos apenas um conjunto aleatório de dados da amostra de treinamento em cada iteração.

Apesar de ser mais irregular, é uma boa opção quando estamos trabalhando com muitos dados.

4.4 Método do Gradiente em SVMs

Em SVMs nossa função de custo, que queremos minimizar, foi definida em 3.2 como

$$J(w,b) = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{n=1}^{N}\max\left\{0, 1 - y_i(\langle \mathbf{w}, \mathbf{x}_i\rangle + b)\right\}. \tag{4.4}$$

Dessa forma, podemos dividir nosso ploblema em dois casos:

- Se $y^{(n)}(\langle \mathbf{w}, \mathbf{x}^{(n)}\rangle + b \geq 1$:

$$\frac{\partial J(\mathbf{w},b)}{\partial \mathbf{w}} = \mathbf{w}\frac{\partial J(\mathbf{w},b)}{\partial b} = 0. \tag{4.5}$$

- Se $y^{(n)}(\langle \mathbf{w}, \mathbf{x}^{(n)}. \rangle + b < 1$:

$$\frac{\partial J(\mathbf{w},b)}{\partial \mathbf{w}} = \mathbf{w} - C\sum_{n=1}^{N} y_i \mathbf{x}_i \frac{\partial J(\mathbf{w},b)}{\partial b} = -C\sum_{n=1}^{N} y_i \tag{4.6}$$

Logo, cada interação do método executará:

$$\mathbf{w}_{k+1} = \mathbf{w}_k - \eta \frac{\partial J(\mathbf{w},b)}{\partial \mathbf{w}} \tag{4.7}$$

$$b_{k+1} = b_k - \eta \frac{\partial J(\mathbf{w},b)}{\partial b}. \tag{4.8}$$

Portanto, após encontrar um par $(\mathbf{w}^*, b^*)$ que nos fatisfaça, podemos ter a decisão feita pelo SVM:

$$h(\mathbf{x}) = \operatorname{sgn}(\mathbf{w}^* \cdot \mathbf{x} + b^*). \tag{4.9}$$

5. Aplicação

Neste capítulo, vamos explorar uma aplicação prática do método de gradiente estocástico aplicado ao algoritmo de SVM.

Utilizaremos o Google Colab, um ambiente interativo que permite executar código Python na nuvem, para implementar o algoritmo. Os dados que utilizaremos foram retirados do Kaggle [5], uma plataforma conhecida por hospedar conjuntos de dados e competições de ciência de dados.

5.1 Ploblema proposto

Analisaremos um conjunto de dados contendo informações de várias pessoas que realizaram diversos testes médicos. Nosso objetivo é utilizar esses dados para desenvolver um modelo capaz de prever se um indivíduo é diabético ou não, com base nos resultados dos testes.

Primeiro vamos importar algumas bibliotecas e analisar nossos dados:

```
import numpy as np
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
```

```
diabetes_data.head()
```

Pregnancies	Glucose	Pressure	Skin	Insulin	BMI	DiabetesFunction	Age	Outcome
6	148	72	35	0	33.6	0.627	50	1
1	85	66	29	0	26.6	0.351	31	0
8	183	64	0	0	23.3	0.672	32	1
1	89	66	23	94	28.1	0.167	21	0
0	137	40	35	168	43.1	2.288	33	1

```
diabetes_data.shape
```

```
(768, 9)
```

Podemos observar que os dados abrangem 768 pacientes, cada um com 9 informações. As 8 primeiras colunas descrevem características individuais dos pacientes, enquanto a última coluna indica a presença ou ausência de diabetes (0 = sem diabete, 1 = com diabete).

Vamos separar as características do resultado.

```
caract = diabetes_data.drop(columns='Outcome', axis=1)
result = diabetes_data['Outcome']
```

Observe que os dados não estão padronizados, isso pode dificultar o modelo a achar uma solução, por conta disso vamos padronizar as características.

```
scaler = StandardScaler()
scaler.fit(caract)
caract = scaler.transform(caract)
```

Podemos, então, dividir nossos dados em amostra de treino e amostra de teste. Como descrito no Capítulo 1, utilizaremos a amostra de treino para treinar nosso modelo e avaliaremos seu desempenho na amostra de teste. Vamos dividir aleatoriamente em 80% dos dados para treino e 20% para teste.

```
x_train, x_test, y_train, y_test = train_test_split(caract, result, test_size=0.2,
    ↪ random_state = 1)
```

5.2 Método do Gradiente Descendente em SVMs

Com os dados já preparados, podemos agora focar no modelo em si. Este modelo envolve a definição de parâmetros importantes, como a taxa de aprendizado, o número de interações e o parâmetro C. Em seguida, ajustamos os dados para o algoritmo e iniciamos o processo de iteração. Durante cada iteração, o modelo executa as equações 4.5, 4.6, 4.7 e 4.8. Finalmente, o modelo retorna a previsão feita pelo algoritmo, utilizando a equação 4.9.

```
class SVM_method():
# Definicao dos parametros necessarios
  def __init__(self, learning_rate, num_iterations, c_parameter):
    self.learning_rate = learning_rate # tamanho do passo da interacao
    self.num_iterations = num_iterations # numero de interacoes
    self.c_parameter = c_parameter # peso dado aos erros

# Ajustando os dados ao algoritmo do SVM
  def fit(self, X, Y):
    self.m, self.n = X.shape
    self.w = np.zeros(self.n) # iniciando com um w
    self.b = 0 # iniciando com um b
    self.X = X
    self.Y = Y
    for i in range(self.num_iterations): # chamando as interacoes
      self.update()

# Uso do Gradiente Descendente para as interacoes
  def update(self):
    y_label = np.where(self.Y <= 0, -1, 1) # ajuste dos labels
    for index, x_i in enumerate(self.X):
      condition = y_label[index] * (np.dot(x_i, self.w) + self.b) >= 1
      if (condition == True):
        dw = self.w
        db = 0
```

```
        else:
          dw = self.w - self.c_parameter * np.dot(x_i, y_label[index])
          db = -self.c_parameter * y_label[index]
        self.w = self.w - self.learning_rate * dw
        self.b = self.b - self.learning_rate * db

    # Previsao do resultado dado um input
      def predict(self, X):
        h = np.sign(np.dot(X, self.w) + self.b)
        result = np.where(h <= -1, 0, 1)
        return result
```

Verificamos, então, como o modelo se comportará ao analisar nossos dados. Optamos por uma taxa de aprendizado de 0.0001 (para garantir passos pequenos e maior precisão), um número de interações de 100.000 e um parâmetro C de 100 (para atribuir um peso maior aos erros).

```
classifier = SVM_method(learning_rate=0.0001, num_iterations=100000, c_parameter
    ↪ =100)
classifier.fit(x_train, y_train)
```

Por fim, usamos a biblioteca accuracy score para verificarmos quanto nosso modelo acertou tanto no treino como no teste.

```
x_train_prediction = classifier.predict(x_train)
training_data_accuracy = accuracy_score(y_train, x_train_prediction)
print(training_data_accuracy)
```

```
    0.7833876221498371
```

```
x_test_prediction = classifier.predict(x_test)
test_data_accuracy = accuracy_score(y_test, x_test_prediction)
print(test_data_accuracy)
```

```
    0.7792207792207793
```

Como era esperado, nosso algoritmo conseguiu criar um modelo genérico o suficiente para que, na amostra de teste, ele desempenhasse tão bem quanto no treino, acertando 77,9% dos casos de diabetes não vistos. No entanto, esse algoritmo levou 12 minutos para encontrar o modelo, um tempo relativamente longo, especialmente considerando que nossa base de dados não era tão grande (com menos de 1.000 dados). Por isso, vamos testá-lo agora com o gradiente estocástico para compararmos os resultados.

5.3 Metodo do Gradiente Estocástico em SVMs

Basicamente, o processo será semelhante ao do gradiente descendente, mas com uma diferença: durante o processo de atualização nas interações, algumas linhas de código serão adicionadas para selecionar aleatoriamente apenas uma fração dos dados (no nosso caso, escolhemos 25%) para cada interação, ou seja, um novo parâmetro será adicionado (sample fraction).

```
# Uso do Gradiente Estocastico para as interacoes
    def update(self):
        # Seleciona aleatoriamente uma fracao dos dados
        indices = np.random.choice(self.m, size=int(self.sample_fraction * self.m),
            ↪ replace=False)
        X_sample = self.X[indices]
        Y_sample = self.Y[indices]

        # Atualiza o modelo para cada amostra da fracao selecionada
```

```
            for index in range(len(X_sample)):
                x_i = X_sample[index]
                y_i = Y_sample[index]
                y_label = np.where(self.Y <= 0, -1, 1)
                condition = y_label * (np.dot(x_i, self.w) + self.b) >= 1
                if condition:
                    dw = self.w
                    db = 0
                else:
                    dw = self.w - self.c_parameter * np.dot(x_i, y_label)
                    db = -self.c_parameter * y_label
                self.w = self.w - self.learning_rate * dw
                self.b = self.b - self.learning_rate * db
```

Aplicando esse algoritmo para a os mesmos dados e condições do gradiente descendente chegamos aos seguintes resultados:

```
x_train_prediction = classifier.predict(x_train)
training_data_accuracy = accuracy_score(y_train, x_train_prediction)
print(training_data_accuracy)
```

```
    0.7785016286644951
```

```
x_test_prediction = classifier.predict(x_test)
test_data_accuracy = accuracy_score(y_test, x_test_prediction)
print(test_data_accuracy)
```

```
     0.7662337662337663
```

Observa-se que, novamente, obtivemos resultados satisfatórios e alinhados com a teoria. Embora nosso algoritmo tenha acertado 76,6% dos casos, aproximadamente 1% a menos do que o anterior, ele completou a tarefa em apenas 1 minuto, representando uma melhoria de 12 vezes em termos de velocidade quando comparado ao gradiente descendente, justamente o que queriamos mostrar.

Gradiente	Precisão no Treino	Precisão no Teste	Tempo Necessário
Descendente	78,33%	77,92%	12 min
Estocástico	77,85%	76,62%	1 min

Tabela 5.1: Resultados da utilização dos dois métodos de otimização.

6. Conclusões

Este relatório forneceu uma visão sobre a aplicação de técnicas de Machine Learning, com foco na Dimensão VC, SVM e o método de Gradiente Estocástico. A partir da revisão teórica e prática desses conceitos, foi possível compreender a importância da Dimensão VC na análise da capacidade de generalização dos modelos e como o SVM pode ser eficaz na construção de modelos de previsão.

A aplicação do método de Gradiente Estocástico demonstrou ser uma ferramenta valiosa para a otimização do treinamento de modelos, especialmente em grandes conjuntos de dados. O exemplo prático de classificação de diabetes ilustrou a capacidade dessas técnicas de resolver problemas reais, evidenciando a eficácia do SVM combinado com Gradiente Estocástico na criação de modelos de previsão precisos e eficientes.

Obtivemos uma clara noção da importância do método de Gradiente Estocástico através dos resultados da aplicação prática. Utilizando o gradiente descendente, alcançamos uma precisão de 77,9 % na previsão dos casos de diabetes em 12 minutos. Em contrapartida, com o gradiente estocástico, obtivemos uma precisão similar de 76,6 % em apenas 1 minuto. Isso demonstra que, em análises envolvendo grandes volumes de dados, o método de Gradiente Estocástico pode ser essencial devido à sua eficiência em tempo de processamento.

Por fim, acreditamos que nosso modelo ainda pode ser significativamente melhorado com a introdução de algumas técnicas de machine learning como a utilização de kernels. Alcançar apenas cerca de 77 % de precisão é um indicativo de que a distribuição dos dados provavelmente não é linear. Com o uso de kernels, poderíamos permitir que SVMs lidassem com dados não linearmente separáveis, ampliando ainda mais suas capacidades. Esse será nosso objetivo em próximos projetos.

Bibliografia

[1] Mohri, Mehryar, Rostamizadeh, Afshin, and Talwalker, Ameet. *Foundations of Machine Learning*, second edition. The MIT Press, 2018. Cambridge, MA.

[2] Robinson Semolini. *Support Vector Machines, Transductive Inference, and the Classification Problem*. Master's Thesis, Universidade Estadual de Campinas, 2002. Campinas, SP.

[3] James, Gareth, Witten, Daniela, Hastie, Trevor, Tibshirani, Robert, and Taylor, Jonathan. *An Introduction to Statistical Learning with Applications in Python*. 2023.

[4] Géron, Aurelien. *Hands–On Machine Learning with Scikit–Learn and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*, 2nd edition. O'Reilly, 2019. Sebastopol, California, USA. ISBN 1492032646.

[5] Dilekci, Melike. *Diabetes Dataset for Beginners*. 2023. Available online: `https://www.kaggle.com/code/melikedilekci/diabetes-dataset-for-beginners`. Accessed: 2025-01-29.

Tudo tem seu centro

Resumo

Tudo tem seu centro

Gabriel Costa de Oliveira

Orientador: Prof. Lúcio Tunes dos Santos

O trabalho "Tudo tem seu centro", proposto pelo Professor Lúcio, teve como objetivo principal o estudo do cálculo da área, perímetro e centroide de um polígono genérico utilizando conceitos de Cálculo I e II. Utilizei como base artigos sugeridos pelo próprio professor, que trouxeram uma introdução ao assunto. Em seguida, apliquei os novos conhecimentos em alguns exemplos, focando no mapa do Brasil. Por meio dos cálculos obtidos na parte teórica, consegui encontrar a cidade do centroide do país, bem como de cada um dos 26 estados e do Distrito Federal. Nos capítulos finais, estudei o conceito de centroides com pesos e algumas aplicações práticas. Por meio de um novo artigo, encontrei maneiras de interpolar pontos, algo que pode ser útil para o propósito visto. Os passos, e dificuldades, de todo o desenvolvimento estão devidamente descritos neste relatório.

Como foi a iniciação científica

No começo do semestre, marquei uma reunião com o Professor Lúcio para discutir sobre temas que poderiam ser estudados. Ele, então, me apresentou o projeto "Tudo tem seu centro", que já havia realizado com outros alunos anteriormente. Acabei me interessante pelos temas propostos, então decidi por estudar este trabalho.

Como já realizei outros dois projetos com o professor, já entendia bem como funcionava toda a metodologia do PICME. Para que eu relatasse o progresso feito na pesquisa, realizávamos reuniões quinzenais, onde eu sempre busquei enviar relatórios parciais com os acréscimos que havia feito (esse relatório foi construído ao longo do semestre inteiro). Isso acabou por ser uma ótima estratégia, já que consegui relatar exatamente o que fiz em cada etapa, além de evitar que todo o trabalho se acumulasse para o final do prazo de entrega.

7. Área, perímetro e centroide

Durante as aulas de matemática no período escolar, somos apresentados a diversas fórmulas e conceitos sobre como calcular a área de certos polígonos. Todos sabemos como calcular a área de um triângulo ($\frac{b \cdot h}{2}$), onde b representa o comprimento da base e h o da altura, de um retângulo ($b \cdot h$) ou de qualquer outro polígono regular de n lados ($\frac{n \cdot l \cdot a}{2}$), onde a equivale à apótema da figura. Do mesmo modo, o perímetro também é algo ao qual já estamos familiarizados, sendo bem simples o caminho para se obter tal medida. O centroide deve ser aquele que é o mais desconhecido dos três, já que, devido a sua natureza matemática que envolve conceitos um pouco mais avançados, não é apresentado durante o período escolar.

Mesmo sabendo essas informações, é interessante analisar e estudar como o cálculo desses itens é realizado. Conhecer as fórmulas utilizadas em cada caso nem sempre é o melhor jeito de entender como realmente o processo é feito. Deduzir, nós mesmos, esses caminhos certamente é o meio mais eficaz para a real compreensão de algo.

Pode parecer uma tarefa difícil, mas existe um método, não muito complicado, que nos permite calcular a área de qualquer polígono, independentemente do formato ou do número de lados. Com um certo conhecimento sobre cálculo, principalmente sobre integrais de linhas, podemos iniciar nossos estudos sobre os polígonos.

7.1 Área: Teorema de Green

Vamos considerar uma certa região D que representa a área interna do polígono que queremos calcular. Conhecendo a aplicação de integrais, temos que a área da figura é obtida pela integral

$$A = \iint_D dA.$$

Olhando dessa maneira, talvez não fique muito claro. Porém, analisando o Teorema de Green, que nos traz informações acerca do cálculo de integrais em regiões fechadas, temos seu enunciado:

$$\oint_C P dx + Q dy = \iint_D \left(\frac{\partial Q}{\partial x} - \frac{\partial P}{\partial y} \right) dA.$$

Visto isso, precisamos apenas fazer com que o integrando da integral dupla seja uma constante para, enfim, relacionarmos as integrais com a área da região D. Podemos fazer isso de diversas maneira, porém,

com uma pequena dica do Professor Lúcio, vou considerar Q=x e P=-y (o que vai facilitar os próximos cálculos). Dessa maneira,

$$\iint_D \left(\frac{\partial Q}{\partial x} - \frac{\partial P}{\partial y}\right) dA = 2\iint_D dA \implies \iint_D dA = \frac{1}{2}\oint_C -ydx + xdy.$$

Como encontrar uma parametrização para a região D nem sempre é possível (ou é desnecessariamente complicado), podemos utilizar essa relação buscando algo mais acessível. Contudo, esbarramos novamente na dificuldade de parametrização, desta vez em relação a curva C. Para contornar esse problema, temos uma importante propriedade das integrais de linha. Sejam γ e δ duas curvas consecutivas, é verdade que

$$\oint_{\gamma\cup\delta} Pdx + Qdy = \oint_{\gamma} Pdx + Qdy + \oint_{\delta} Pdx + Qdy.$$

Esse resultado nos permite quebrar a curva C em n curvas, desde que estas sejam consecutivas e que $C_1 \cup C_2 \cup ... \cup C_n = C$.

$$\frac{1}{2}\oint_C -ydx + xdy = \frac{1}{2}\left(\oint_{C_1} -ydx + xdy + \oint_{C_2} -ydx + xdy + ... + \oint_{C_n} -ydx + xdy\right).$$

Finalmente, podemos calcular a integral de linha referente a cada lado do polígono e obter sua área (como se tratam de segmentos de reta, a parametrização é mais simples).

7.1.1 Parametrizando segmentos

A resolução de uma integral de linha é feita da seguinte maneira:

$$\oint_C Pdx + Qdy = \int_a^b (P(x(t),y(t)))x'(t) + (Q(x(t),y(t)))y'(t)dt,$$

onde $(x(t),y(t))$ é a parametrização da curva e $t \in [a,b]$.

Anteriormente, quando nós quebramos a integral de linha em curvas menores, nós decidimos que cada curva representaria um segmento do polígono. Para generalizar, podemos definir que o segmento de número k terá como extremidades os pontos (x_{k-1}, y_{k-1}) e (x_k, y_k).

Visto isso, precisamos apenas encontrar uma parametrização para os segmentos de reta e, finalmente, calcular a expressão. Como temos dois pontos ((x_{k-1}, y_{k-1}) e (x_k, y_k)), podemos encontrar a função da reta do seguinte modo:

$$r : A + (B-A)t \implies r : (x_{k-1}, y_{k-1}) + (x_k - x_{k-1}, y_k - y_{k-1})t \implies r : (x_{k-1} + x_k t - x_{k-1}t, y_{k-1} + y_k t - y_{k-1}t).$$

Dessa maneira, temos todos os dados de que precisamos para o cálculo da integral de linha.

$$x(t) = (x_{k-1} + x_k t - x_{k-1}t) \implies x'(t) = (x_k - x_{k-1})$$
$$y(t) = (y_{k-1} + y_k t - y_{k-1}t) \implies y'(t) = (y_k - y_{k-1})$$
$$t \in [0,1]\ (t = 0 \implies \text{Ponto A e } t = 1 \implies \text{Ponto B})$$

Substituindo tais resultados na expressão, segue que

$$\oint_{C_k} -ydx + xdy = \int_0^1 -(y_{k-1} + y_k t - y_{k-1}t)(x_k - x_{k-1}) + (x_{k-1} + x_k t - x_{k-1}t)(y_k - y_{k-1})dt.$$

Olhando apenas para o integrando, podemos simplificar a expressão (vou considerar j=k-1 para simplificar a notação):

$$(-y_j - y_k t + y_j t)(x_k - x_j) + (x_j + x_k t - x_j t)(y_k - y_j) =$$
$$-y_j x_k + y_j x_j - y_k x_k t + y_k x_j t + y_j x_k t - y_j x_j t + x_j y_k - x_j y_j + x_k y_k t - x_k y_j t - x_j y_k t + x_j y_j t =$$

$$x_j y_k - y_j x_k$$

Voltando para as integrais:

$$\oint_{C_k} -ydx + xdy = \int_0^1 x_{k-1}y_k - x_k y_{k-1} dt = x_{k-1}y_k - x_k y_{k-1},$$

sendo esta a integral referente a um lado da figura. Como temos n lados no polígono, concluímos, por fim, que sua área é obtida pela seguinte expressão:

Fórmula da Área

$$A = \iint_D dA = \frac{1}{2}\oint_C -ydx + xdy = \frac{1}{2}\sum_{k=1}^{n}(x_{k-1}y_k - x_k y_{k-1}),$$

onde $(x_n, y_n) = (x_0, y_0)$.

7.1.2 Outras notações

Analisando mais afundo o resultado obtido, é possível notar que existem meios diferentes de apresentá-lo. Uma dessas maneiras, por exemplo, é utilizando o conceito de determinante. Podemos utilizar o seguinte:

$$\begin{vmatrix} x_{k-1} & x_k \\ y_{k-1} & y_k \end{vmatrix} = (x_{k-1}y_k - x_k y_{k-1}).$$

Como pode-se observar, a determinante apresentada equivale exatamente ao interior do somatório que fornece a área. Desse modo, temos que a fórmula da área pode ser reescrita como

$$\frac{1}{2}\sum_{k=1}^{n}(x_{k-1}y_k - x_k y_{k-1}) = \frac{1}{2}\sum_{k=1}^{n}\begin{vmatrix} x_{k-1} & x_k \\ y_{k-1} & y_k \end{vmatrix}.$$

Além desta, existe outra notação que, por vezes, é utilizada para a fórmula da área. Pegando os dois primeiros termos do somatório obtido anteriormente, temos:

$$(x_0y_1 - x_1y_0) + (x_1y_2 - x_2y_1) = x_0y_1 + x_1(y_2 - y_0) - x_2y_1.$$

A partir dessa reorganização de termos, temos que o valor x_1 pôde ser colocado em evidência. Realizando o somatório total, e lembrando que $(x_n, y_n) = (x_0, y_0)$, temos que todos os valores x_i dos vértices seguem esse caminho. Assim, podemos reescrever o somatório como

$$\frac{1}{2}\sum_{k=1}^{n}(x_{k-1}y_k - x_k y_{k-1}) = \frac{1}{2}\sum_{k=1}^{n} x_i(y_{i+1} - y_{i-1}).$$

Indo além, podemos, ao invés de colocar x_i em evidência, escolher y_i como fator comum. Com isso, a seguinte fórmula também é equivalente:

$$(x_0y_1 - x_1y_0) + (x_1y_2 - x_2y_1) = -x_1y_0 + y_1(x_0 - x_2) + x_1y_2$$

$$\implies \frac{1}{2}\sum_{k=1}^{n} y_i(x_{i-1} - x_{i+1}).$$

7.1.3 Uma outra estratégia

Pelo desenvolvimento feito há pouco, temos que a fórmula obtida é relativamente simples, e pode ser facilmente aplicada em qualquer polígono. Porém, existe um outro caminho para o cálculo de uma área que é interessante ser mencionado.

O artigo [2] traz o processo do autor David Richeson no cálculo da área dos Estados Unidos. David, porém, não faz uso de nada do que vimos. Na verdade, ele usa apenas conceitos de Cálculo I, e é por isso que esse método pode ser interessante de se analisar.

Primeiramente, David define duas funções: $N(x)$ como sendo a função que passa pelos pontos superiores do mapa e $S(x)$ a função que passa pelos pontos inferiores. Dado essas duas funções, temos que a integral da primeira ao longo dos pontos extremos do mapa fornece a área completa da região, mas com um pequeno acréscimo abaixo da figura. A integral da segunda, por sua vez, fornece exatamente esse pequeno acréscimo, fazendo com que a diferença das duas seja exatamente o valor buscado:

$$A = \int_W^E (N(x) - S(x))dx,$$

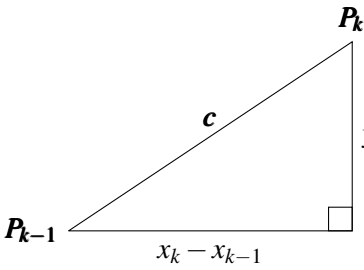
onde W representa o ponto mais a oeste do mapa e E o ponto mais a leste (ou seja, os extremos da figura).

Porém, caso os pontos do mapa não se comportem como uma função (x_i não necessariamente menor do que x_{i+1}), não podemos encontrar as funções $N(x)$ e $S(x)$. Para contornar esse problema, o autor do artigo realiza uma pequena rotação no mapa dos Estados Unidos, garantindo que os valores de x sejam estritamente crescentes nos pontos do contorno de cada parte (norte e sul).

O último ponto a ser verificado nesse tópico é a obtenção das funções necessárias na integração. Para muito pontos, temos que a função pode se tornar complicada, sendo necessário o auxílio de um computador para os cálculos. Mais adiante, veremos métodos de interpolação de pontos que podem ser úteis para essa tarefa.

7.2 Perímetro

O cálculo do perímetro de uma figura é relativamente mais simples do que o processo para obtenção de sua área. Apenas conhecendo o teorema de Pitágoras, podemos realizar triangulações sobre cada segmento e calcular seu comprimento. É evidente que o valor procurado será o somatório de todos os lados do polígono.



$$\implies c = \sqrt{(x_k - x_{k-1})^2 + (y_k - y_{k-1})^2}$$

Fórmula do Perímetro

$$\implies \sum_{k=1}^{n} \sqrt{(x_k - x_{k-1})^2 + (y_k - y_{k-1})^2},$$

onde **n** representa o número de lados do polígono e, novamente, $(x_n, y_n) = (x_0, y_0)$.

7.2.1 Um pequeno problema

Mesmo a fórmula do perímetro sendo completamente precisa, temos que existe um pequeno problema relacionado ao perímetro de uma figura, e que será observado mais adiante na pesquisa.

Quando buscamos contornar uma região, é intuitivo que, com mais pontos, a precisão do contorno será maior, então os valores obtidos devem ser mais fiéis à realidade. Isso realmente é observado para a

área da figura, que tende a se aproximar do valor esperado. Porém, para o perímetro, temos que este é uma quantidade que nunca converge para um certo valor. Quanto mais pontos, maior é o perímetro obtido, fazendo com que ele seja algo muito incerto, e que, nos nossos próximos testes, sempre apresentou um grande erro para o valor real.

7.3 Centroide

O centroide acaba por ser um conceito um pouco mais complexo. Podemos utilizar diversos parâmetros diferentes que alteram o resultado final. Porém, inicialmente, vamos apenas considerar tal definição como sendo o centro geométrico da figura, sem que sejam concedidos pesos para regiões ou algo do tipo.

As coordenadas G do centroide são dadas pela equação

$$G = \frac{1}{A}\left(\iint_D x\,dx\,dy, \iint_D y\,dx\,dy\right),$$

onde D representa a região de interesse e A sua área. Como nós já desenvolvemos a parte da pesquisa relacionada a esse último valor, não teremos dificuldades em determiná-lo.

7.3.1 Aplicando o teorema de Green para os centroides

Visto toda a parte do desenvolvimento em relação à área, não precisamos passar por todas as etapas novamente. É evidente que podemos aplicar os mesmos passos feitos anteriormente agora na equação do centroide.

Analisando primeiro a integral referente à coordenada x, temos que, pelo Teorema de Green,

$$\oint_C P dx + Q dy = \iint_D x\,dx\,dy \implies \frac{\partial Q}{\partial x} - \frac{\partial P}{\partial y} = x.$$

Novamente, temos várias escolhas diferentes para os valores de Q e P. Vou tomar $Q = x^2$ e $P = 0$ (apenas um chute que pode facilitar os cálculos, mas é claro que, sendo tudo feito corretamente, o resultado final será o mesmo). Desse modo, temos que a expressão vista é satisfeita.

$$\iint_D \left(\frac{\partial Q}{\partial x} - \frac{\partial P}{\partial y}\right) dx\,dy = 2\iint_D x\,dx\,dy \implies \iint_D x\,dx\,dy = \frac{1}{2}\oint_C x^2 dy.$$

Aplicando a mesma ideia de quebrar a integral de linha e calcular seu valor em cada segmento, podemos simplesmente calcular a integral de linha do lado de ordem k e então somar todos os lados. Como nós já realizamos toda a parametrização, podemos pular essa etapa.

$$\oint_{C_k} x^2\,dy = \int_0^1 (x_{k-1} + x_k t - x_{k-1}t)^2 (y_k - y_{k-1}) dt = \frac{1}{3}(x_{k-1}^2 + x_k x_{k-1} + x_k^2)(y_k - y_{k-1}).$$

Repetindo o somatório anteriormente visto para os n lados do polígonos, chegamos a equação final para o cálculo da coordenada x:

$$G_{(x)} = \frac{1}{A}\iint_D x\,dx\,dy = \frac{1}{2A}\oint_C x^2\,dy = \frac{1}{6A}\sum_{k=1}^{n}(x_{k-1}^2 + x_k x_{k-1} + x_k^2)(y_k - y_{k-1}),$$

com $(x_n, y_n) = (x_0, y_0)$.

De maneira análoga, podemos calcular a integral referente à coordenada y do centroide. Nesse caso, vamos considerar $Q = 0$ e $P = y^2$. Para não repetir todos os passos novamente, vou passar para a integral de linha final:

$$-\oint_{C_k} y^2\,dx = -\int_0^1 (y_{k-1} + y_k t - y_{k-1}t)^2 (x_k - x_{k-1}) dt = \frac{1}{3}(y_{k-1}^2 + y_k y_{k-1} + y_k^2)(x_{k-1} - x_k).$$

Dessa maneira, obtemos a formulação final para a coordenada y e, consequentemente, para o ponto do centroide:

Fórmula do Centroide

$$G = \frac{1}{6A}\left(\sum_{k=1}^{n}(x_{k-1}^2 + x_k x_{k-1} + x_k^2)(y_k - y_{k-1}), \sum_{k=1}^{n}(y_{k-1}^2 + y_k y_{k-1} + y_k^2)(x_{k-1} - x_k)\right),$$

sempre considerando $(x_n, y_n) = (x_0, y_0)$ (já que lidamos com curvas fechadas em todos os casos).

7.3.2 Diferentes caminhos, mesmos resultados

Observando a fórmula para o centroide utilizada em alguns artigos matemáticos, temos que ela se apresenta de maneira diferente daquela que obtemos por meio das integrações:

$$G = \frac{1}{6A}\left(\sum_{k=1}^{n}(x_{k-1} + x_k)(x_{k-1}y_k - x_k y_{k-1}), \sum_{k=1}^{n}(y_{k-1} + y_k)(x_{k-1}y_k - x_k y_{k-1})\right).$$

Num primeiro momento, existe a dúvida se realmente realizamos todo o processo corretamente. O que será que justifica tal diferença nos resultados? Isso acaba por alterar os pontos calculados em cada fórmula?

Precisamos realizar uma análise mais correta do problema para entender o que ocorreu. Como as equações de cada coordenada apresentaram um desenvolvimento semelhante, podemos olhar apenas uma delas, já que a outra terá uma conclusão análoga.

Visto que a parte fracionária que está em evidência e os índices dos somatórios são iguais, podemos verificar apenas o interior de cada soma. Rearranjando os termos da equação que obtivemos, temos o seguinte:

$$(x_{k-1}^2 + x_k x_{k-1} + x_k^2)(y_k - y_{k-1}) = (x_{k-1} + x_k)(x_{k-1}y_k - x_k y_{k-1}) - x_{k-1}^2 y_{k-1} + x_k^2 y_k.$$

Desse modo, notamos que a expressão resultante de nosso desenvolvimento tem um pequeno excedente $(-x_{k-1}^2 y_{k-1} + x_k^2 y_k)$. Para que os resultados sejam equivalentes, o somatória referente a essa pequena parcela precisa ser nulo.

$$\sum_{k=1}^{n}(-x_{k-1}^2 y_{k-1} + x_k^2 y_k) = (-x_0^2 y_0 + x_1^2 y_1) + (-x_1^2 y_1 + x_2^2 y_2) + \ldots + (-x_{n-1}^2 y_{n-1} + x_n^2 y_n) = 0.$$

Na verdade, é bem simples perceber que isso é verdade. Por meio da somatória completa, temos que o último termo de cada parcela se cancela com o primeiro da próxima parcela. No final, sobrariam apenas $(-x_0^2 y_0 + x_n^2 y_n)$, mas, como esses pontos são iguais devido à condição da curva fechada, eles também se anulam, fazendo com que esse excedente da fórmula não altere o resultado das operações.

De um ponto de vista computacional, talvez seja até mais interessante utilizar esta fórmula ao invés daquela que obtemos por meio dos cálculos. É evidente que essa somatória de termos que resultam em zero não causa nenhuma diferença no resultado final, fazendo com que isso seja apenas um custo extra.

Tendo confirmado isso, ainda podemos buscar entender o que gerou essa diferença entre as equações. Isso também é algo bem simples de perceber, e deriva de uma escolha que fizemos no começo de cada processo: a escolha dos fatores Q e P. Como existem diferentes valores possíveis que satisfazem a condição necessária, obtemos que cada um leva para uma solução diferente.

No caso da formulação vista nos artigos, a escolha que levou a tal resultado foi $Q = x^2$ e $P = -xy$ (para a coordenada x). Desenvolvendo, é fácil verificar que de fato se chega à equação analisada. Os valores $Q = 0$ e $P = -xy$, por exemplo, também fornecem um resultado diferente, mas que acaba levando ao mesmo destino.

$$\oint_{C_k} -xy\,dx + x^2\,dy =$$

$$\int_0^1 -(x_{k-1}+x_k t - x_{k-1}t)(y_{k-1}+y_k t - y_{k-1}t)(x_k - x_{k-1}) + (x_{k-1}+x_k t - x_{k-1}t)^2(y_k - y_{k-1})dt$$

$$= \frac{1}{2}(x_{k-1}+x_k)(x_{k-1}y_k - x_k y_{k-1}).$$

7.3.3 Por que o centroide funciona?

Ao longo desta sessão, nós desenvolvemos cálculos que simplificam a integral dupla inicial que tínhamos para o centroide. Como cada passo foi feito minuciosamente, essa parte ficou bem clara. Porém, ainda é interessante fazer a seguinte pergunta: por que exatamente as integrais das componentes x e y fornecem as coordenas do centroide da figura?

Para a área, é bem fácil ver que as integrais duplas, por definição, devem resultar nessa quantidade. Contudo, não é tão trivial, pelo menos para mim, entender a diferença que a mudança no integrando de constante para funções (x e y) causa no resultado.

Agora, em relação aos centroides, temos que ele está diretamente relacionado ao conceito de centro de massa, muito utilizado na física e engenharia. No caso específico que estamos utilizando, consideramos o centroide como o centro geométrico de uma figura, porém veremos mais adiante (seção 3.1.1) que este é, na verdade, um caso específico de um centro de massa. Neste momento, então, finalmente entenderemos a verdadeira natureza do centro geométrico de uma figura.

8. Testando na prática

Feito toda essa parte do desenvolvimento matemático, podemos passar para a parte de testes e, de fato, verificar a acurácia das fórmulas. Faremos isso calculando os valores de área, perímetro e centroide do Brasil. Porém, antes, precisamos passar por algumas etapas de preparação:

8.1 Algoritmo em Python

Para figuras mais simples, com um número pequeno de vértices, aplicar as fórmulas obtidas é algo rápido. Contudo, quando, mais para frente, lidarmos com mapas de regiões, o mapeamento envolverá centenas de pontos. Por isso, calcular cada valor à mão se torna uma tarefa muito extensa.

Visando simplificar esse passo, criei um código em Python que, dado o número de pontos e seus respectivos valores, todos os cálculos são feitos (para as três quantidades estudadas). Eis sua forma:

```
n = int(input("Numero de pontos:"))
P = 0; A = 0; Cx = 0; Cy = 0
x = []
y = []
for i in range(n):
    X, Y = [float(v) for v in input().split()]
    x.append(X)
    y.append(Y)
x.append(x[0])
y.append(y[0])
for k in range(1, n+1):
    P += ((x[k] - x[k-1])**2 + (y[k] - y[k-1])**2)**0.5
    A += (x[k-1]*y[k] - x[k]*y[k-1])
    Cx += ((x[k-1]**2 + x[k-1]*x[k] + x[k]**2) * (y[k] - y[k-1]))
    Cy += ((y[k-1]**2 + y[k-1]*y[k] + y[k]**2) * (x[k-1] - x[k]))
A = abs(A * 0.5)
Cx = Cx / (6 * A)
Cy = Cy / (6 * A)
print("Area =", A)
print("Perimetro =", P)
print("Coordenadas do centroide =", (Cx, Cy))
```

Utilizando o site *GeoGebra* ([5]), é fácil criar um polígono de teste. Como o próprio site já fornece as coordenadas de cada ponto, fica simples aplicá-los no programa e computar os valores de interesse. Veja o exemplo a seguir:

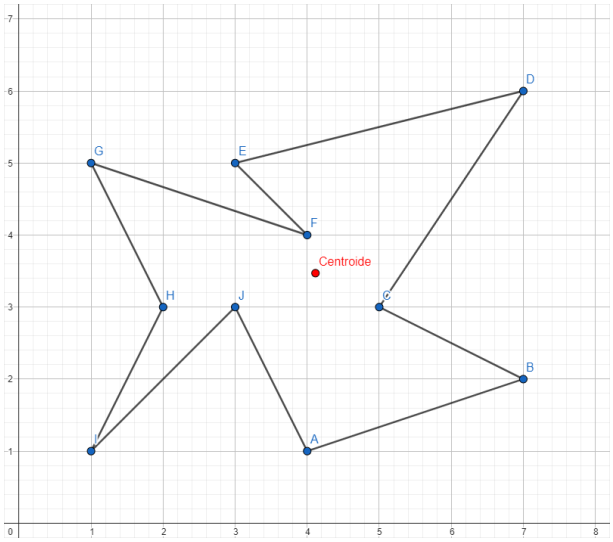


Figura 2: Exemplo feito no GeoGebra.

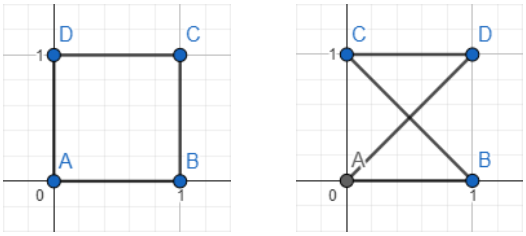
Através do algoritmo, obtemos que esse polígono tem área igual a $14,5$ e perímetro de, aproximadamente, $27,2401$ (sem unidades de medida, já que não definimos nenhuma). As coordenadas do centroide são $(4,1149425; 3,4712644)$, já estando demarcadas no exemplo.

8.1.1 Alguns cuidados

Durante o uso das fórmulas, existem alguns cuidados que devemos ter para evitar obter resultados errados. Na aplicação dos pontos na fórmula, devemos nos atentar que eles estejam seguidamente dispostos e, de preferência, no sentido anti-horário da figura. Caso esse último item não seja seguido, todos os valores estariam corretos (área, perímetro e centroide), porém com a única ressalva de que as coordenadas do centroide estariam com o sinal trocado.

Para a primeira restrição, quando a ordem dos pontos é alterada, temos que a figura não é descrita fielmente. Imaginemos um quadrado com vértices (0,0), (1,0), (1,1) e (0,1). Seguindo essa ordem, temos que o quadrado é definido corretamente (e também no sentido anti-horário). Contudo, denotando os pontos como (0,0), (1,0), (0,1) e (1,1), nos chegamos a um outro polígono.

Plotando esse valores novamente no *GeoGebra*, temos as seguintes imagens:



(a) 1º ordenação (b) 2º ordenação

Figura 3: Exemplo de ordenações diferentes.

Como visto, a ordenação gerou duas figuras diferentes. Através das fórmulas, obtemos $A = 1$, $P = 4$ e $C = (0,5; 0,5)$ para a primeira forma e $A = 0$ e $P = 4,8284271247$ para a segunda (como a área é nula, não podemos calcular o centroide). Claramente, obtemos valores muito equivocados. Na verdade, o perímetro é a única medida que a fórmula sempre vai nos fornecer a quantidade correta (para o polígono descrito).

Mesmo que seja evidente que as figuras são diferentes, ainda fica o questionamento sobre o porquê de o valor obtido para a área na segunda figura não corresponder com a sua respectiva área. De maneira resumida, temos que o algoritmo não pode ser aplicado para polígonos que apresentem pontos de interseção

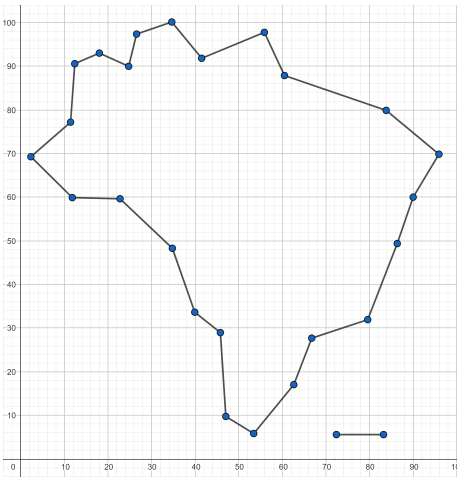
entre seus lados. Uma solução para esse caso seria definir o ponto de interseção como um novo ponto, fazendo com que a figura b seja definida pelos pontos $(0,0),(1,0),(0,5;0,5),(0,1),(1,1)$ e $(0,5;0,5)$ (o novo ponto aparece duas vezes). Através desse método, obtemos $A = 0,5$, o perímetro continua igual e $C = (0,5;0,5)$, que são as medidas corretas.

8.2 Mapeamento de regiões

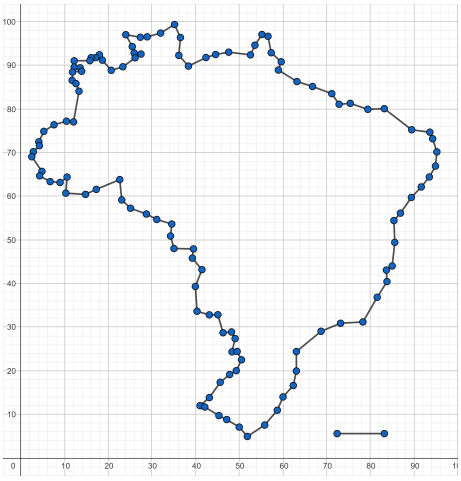
Com todas as ferramentas de cálculo prontas para uso, podemos prosseguir para o objetivo principal desse trabalho: encontrar medidas geográficas para regiões reais. Para isso, precisamos definir uma estratégia para a coleta das coordenadas do mapa de interesse.

Como uma solução simples, decidi usar o *Google Maps* ([6]) em conjunto com o *GeoGebra*, que já usei anteriormente. Para isso, capturei a imagem do território do Brasil pelo *Google Maps*, aproveitando também que ele já fornece a escala do mapa, que será importante mais para frente. Feito isso, copiei a imagem para o site do *GeoGebra* e simplesmente contornei a área de interesse com as ferramentas disponíveis.

(a) Mapa do Brasil no *Google Maps*



(b) 1º contorno no *GeoGebra*



(c) 2º contorno no *GeoGebra*

Figura 4: Mapeamento do território brasileiro.

Para comparação, realizei dois contornos do território brasileiro: um com 24 pontos, menos preciso, e outro com 114 pontos, onde fui mais minucioso nos detalhes. Do mesmo modo que feito nos exemplos, copiei as coordenadas de cada ponto para um bloco de notas e, então, apliquei tais valores no algoritmo. Os resultados para os dois contornos estão dispostos na tabela a seguir.

Contorno do Brasil			
Pontos	**Área**	**Perímetro**	**Centroide**
24	4.338,2804668	304,2925364	(51,6498891; 60,7816659)
114	4.135,1894591	358,5873526	(51,9857553; 60,4132757)

Primeiramente, é importante perceber que esses não representam as quantidades reais do território. Devemos considerar a escala utilizada no mapa para, assim, tirar conclusões. É evidente que as coordenadas do centroide já podem ser aplicadas no mapa, mas devemos, posteriormente, explorar um meio de localizar a região, e até a cidade, localizada em tal ponto.

Observando a imagem do mapa do Brasil no *Google Maps*, é possível notar que há uma marcação referente à escala do mapa. Utilizando essa informação, eu demarquei um segmento com exatamente a medida da escala, fazendo com que seja possível realizar a conversão. Através do próprio *GeoGebra*, foi determinado que esse segmento tem um comprimento de 10,815367 unidades. Como a escala é de 500 km, realizando a divisão, temos que cada unidade do gráfico representa cerca de 46,2305163 km.

Com isso, podemos chamar essa medida de α, e ela será o nosso conversor de unidades. Para o perímetro do mapa, podemos simplesmente multiplicar os valores obtidos por α, chegando em

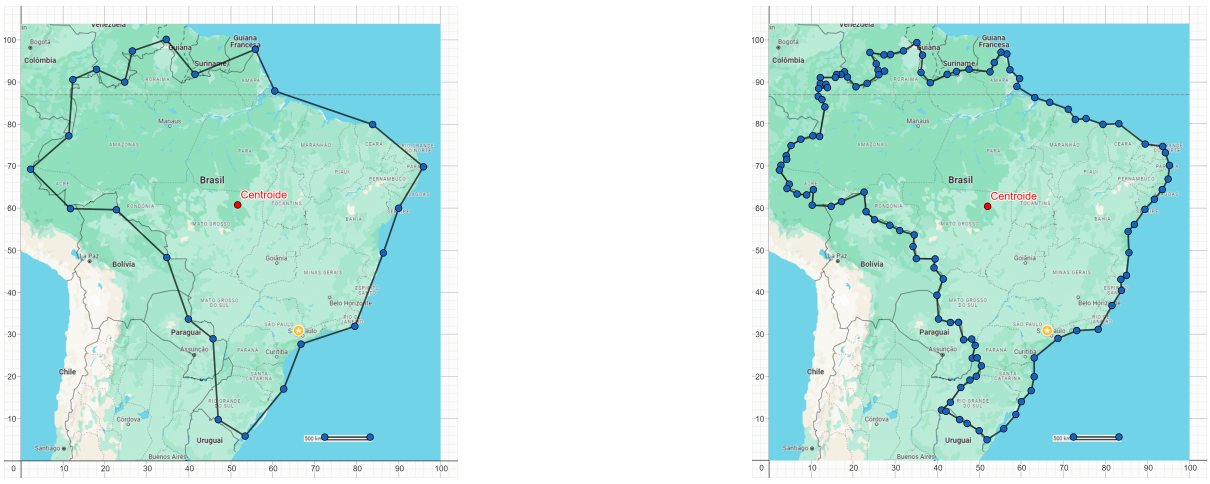
Pontos	**Perímetro (km)**
24	14.067,601
114	16.577,678

Por meio de uma pesquisa, descobri que o valor que se tem hoje para o comprimento do território brasileiro é de cerca de 23.086 km, representando uma diferença considerável dos resultados alcançados. Como foi dito anteriormente, o perímetro é uma quantidade que aumenta com o número de pontos, então ele se torna um valor difícil de definir.

Para a área, precisamos multiplicar o valor alcançado por α^2. Realizando as contas, conclui-se que

Pontos	**Área (km^2)**
24	9.272.036,076
114	8.837.977,659

o que representam quantidades relativamente distantes do esperado. Segundo o IBGE, a área do Brasil é de 8.510.417,771 km^2. Porém, é notório que o aumento nos pontos do contorno aproximou o resultado do valor esperado. A grande disparidade observada se deve, principalmente, pela distorção do mapa (plano), algo que irei debater mais profundamente nas próximas seções. O centroide, por sua vez, ficou bem



(a) Centroide do contorno de 24 pontos

(b) Centroide do contorno de 114 pontos

Figura 5: Mapas do Brasil com o centroide.

próximo nos dois contornos, tendo apenas um pequeno desvio. Ambos estão localizados numa porção nordeste do estado do Mato Grosso.

Nota: fica bem visível na Figura 5 que o contorno de 24 pontos não é nada preciso, tendo segmentos passando dentro do território. Isso é algo proposital, e que será importante na seção 4.4.1.

8.2.1 Aumentando a precisão

Mesmo o método utilizado a pouco tendo fornecido resultados razoáveis, é evidente que é possível melhorar, e muito, a precisão. Pelo GeoGebra, a seleção de pontos foi realizada corretamente, porém o grande problema está na escala do mapa. Uma unidade de diferença na escala já causa dispersão nos resultado, então, como esse meio de medição através da imagem acaba envolvendo muitas incertezas, ele não é o mais adequado para a tarefa.

Pesquisando outras maneiras de realizar o mapeamento da região, encontrei um programa exatamente com essa finalidade: o *QGIS* ([14]). Ele é um software gratuito disponível na internet, então baixá-lo foi relativamente fácil. Estudando um pouco seu funcionamento, vi que ele tinha todas as ferramentas de que eu precisava, como criação de polígonos (para contornar o mapa) e sistema de coordenadas para os vértices da figura. O único ponto que faltava para completar o quebra-cabeça era o molde do mapa.

Por sorte, isso acabou sendo algo fácil de resolver: pelo site do *IBGE* ([4]), são disponibilizados arquivos com o mapeamento do Brasil em diversos formatos, sendo possível encontrar um com todos os municípios brasileiros (algo que vai facilitar a identificação do centroide no mapa). Tais arquivos são de formato compatível com o *QGIS*, então podemos iniciar nossa tarefa.

Realizando um contorno razoavelmente preciso, chegamos no seguinte:

(a) Mapa original (b) Contorno

Figura 6: Aproximação do mapa por um contorno.

Através do programa, temos uma ferramenta que mapeia automaticamente todos os vértices de um polígono, algo que vai facilitar e muito nosso trabalho. Com um total de 192 pontos, podemos aplicar as coordenadas no nosso algoritmo e analisar os resultados.

Antes disso, acabei esbarrando em um conceito fundamental para os próximos passos: o SRC (Sistema de Referência de Coordenadas). Basicamente, ele representa a maneira como as coordenadas serão dispostas, como em latitude e longitude, por exemplo. Para o nosso propósito, o ideal seria um sistema com unidades em metros, já que isso nos pouparia de realizar conversões.

Pesquisando um pouco, encontrei o sistema denominado *SIRGAS 2000 / UTM zone 22S*, que representa uma região central do Brasil e nos fornece coordenadas em metros, que era o que procurávamos. Note

que tudo isso são conceitos geográficos e cartográficos, não sendo o foco principal da pesquisa. Estou apenas passando por esses pontos pois eles serão importantes mais para frente.

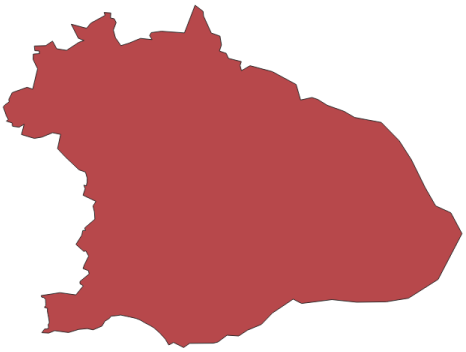
Finalmente, aplicando os pontos em nosso programa, chegamos em: área = 8.799.694,254 km^2, perímetro=17.432,013 km e centroide= (240.132,235; 8.839.420,240).

Surpreendentemente, a área obtida para esse método é semelhante àquela que obtemos utilizando o *GeoGebra* (apenas um pouco melhor), tendo um erro de, aproximadamente, 3,4% em relação ao valor real. Por outro lado, o perímetro seguiu o comportamento esperado, já que, com mais pontos, seu valor aumentou. Para o centroide, aplicaremos, daqui a pouco, seu ponto no mapa.

Pensando um pouco sobre isso, encontrei o fator crucial da disparidade dos resultados: por ser um país muito grande, o Brasil acaba sofrendo, nos mapas, de uma distorção de seu território. Como o mundo é um globo, sua representação através de mapas planos acaba causando um aumento no tamanho de algumas regiões, que é exatamente o que aconteceu.

O problema da distorção vai ainda além: ele depende do sistema de coordenadas que escolhemos para o mapa. O modelo que utilizei, que apresenta as coordenadas em metros, mapeia com precisão a região central do Brasil, fazendo com que as bordas fiquem fora desse espaço. Selecionando, por exemplo, o sistema *SIRGAS 2000 / UTM zone 11N*, que mapeia uma pequena região dos Estados Unidos, obtemos o seguinte mapa:

(c) Região do SRC utilizado (primeiro contorno)



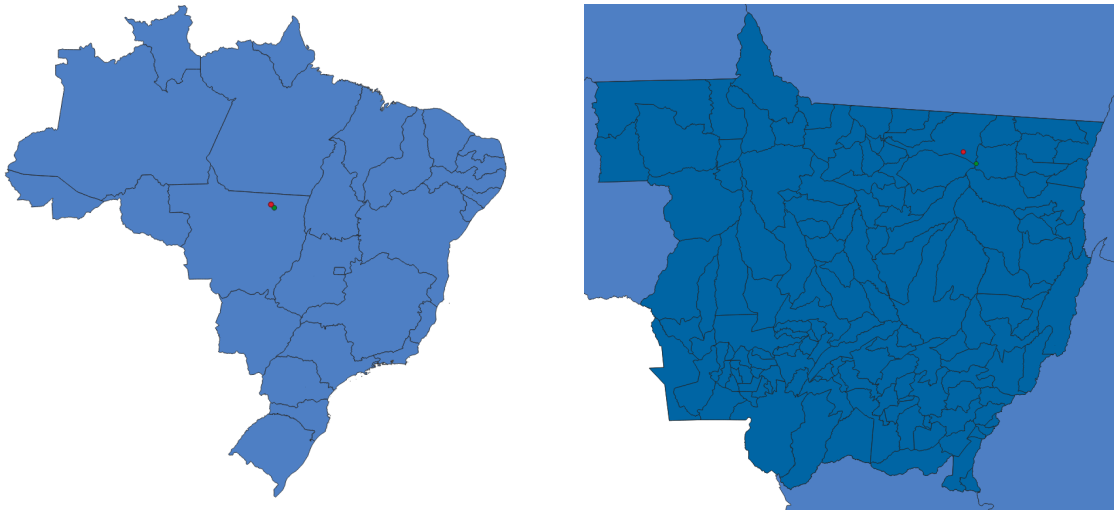
(d) Contorno distorcido

Figura 7: Variações no mapeamento.

Explorando um pouco as funcionalidades do *QGIS*, descobri que ele possui a sua própria ferramenta de cálculo da área. Utilizando tal função, o programa informou que o território do mapa (contorno) tem cerca de 8.592.470,512 km^2, o que representa apenas 0,96% de diferença sobre o valor esperado. A minha teoria para esse resultado diferente para o mesmo contorno é que o próprio *QGIS* realiza correções sobre a escala do programa, fazendo com que o cálculo da área seja feito sobre o globo, e não para o mapa planificado. Uma evidência disso é que o mapa completamente distorcido mostrado acima fornece exatamente essa mesma área, mostrando que o programa realiza os cálculos independentemente do sistema de coordenadas selecionado.

Assim como a área, o centroide também é afetado pelo sistema de coordenadas utilizado. Por meio de nosso teste, encontramos uma certa coordenada para o centroide. Porém, para comparar com outro resultado, decidi utilizar o SRC original do mapa (*WGS 84*, que é baseado no globo) para encontrar esse ponto. Os resultados estão na figura 8 (página seguinte).

Para o centroide do sistema em metros (ponto vermelho), que é levemente distorcido, temos que ele localiza a cidade de Peixoto de Azevedo, no Mato Grosso. O centroide do sistema baseado no globo (ponto verde), por sua vez, teve uma pequena diferença, marcando a cidade de São José do Xingu, também no Mato Grosso. Por todo o nosso desenvolvimento acerca das distorções do mapa, o segundo ponto apresenta um resultado mais confiável, mas precisamos verificar isso.



(a) Centroides no mapa (b) Região ampliada

Figura 8: Centroide do Brasil.

8.2.2 Precisão nível *IBGE*

Aproveitando que nos temos a nossa disposição o mapeamento mais exato do território brasileiro, podemos utilizar ele em nossos testes e, assim, comparar com os resultados obtidos para o contorno do mapa.

Através das ferramentas do *QGIS*, foram contabilizados incríveis 494.632 vértices para o mapa. Infelizmente, não consegui aplicar todos esses pontos no algoritmo (precisei reiniciar meu computador após simplesmente tentar copiar os pontos para um arquivo). Por causa disso, utilizei as próprias funções do programa para calcular os valores de interesse. Foram obtidos os seguintes resultados: área = 8.510.419,034 km^2, perímetro = 28.591,430 km e centroide = $(-53,0717046; -10,7792674)$.

Surpreendentemente (ou não), a área obtida apresenta um erro de apenas 0,01%, provavelmente por algum fator de precisão do programa. Isso representa um erro desprezível, mostrando que este realmente é o método utilizado para esta tarefa. O perímetro aumentou consideravelmente em relação ao obtido anteriormente, mas acabou sendo superior ao valor esperado (23.086 km), mas isso se dá por realmente ser complicado definir uma maneira de medição para essa quantidade.

Por fim, o centroide para este mapa apresentou apenas uma diferença mínima sobre aquele obtido anteriormente, também estando na cidade de São José do Xingu. Porém, pesquisando um pouco sobre qual é de fato a cidade oficial demarcada como centroide do Brasil ([15]), encontrei que, em 1958, uma expedição foi realizada para marcar tal local, que foi localizado na cidade de Peixoto de Azevedo, exatamente como aquele encontrado no primeiro teste, que eu descartei para selecionar um resultado "melhor".

Acredito que, pela expedição ter sido feita há muito tempo, as ferramentas da época apresentavam uma precisão reduzida se comparada à tecnologia atual, causando o pequeno desvio observado na cidade definida como centroide. Com base em todo o desenvolvimento desse capítulo, penso que, ainda assim, a cidade de São José do Xingu seja o verdadeiro centroide do país.

8.3 Mapeando os estados brasileiros

Feita toda essa análise do território brasileiro, temos agora a possibilidade de expandi-la para cada estado do Brasil. Como estes apresentam formatos bem peculiares, será interessante analisar cada um e encontrar seus respectivos centroides. Além disso, será uma maneira de testar se a minha hipótese sobre regiões grandes terem uma maior distorção (como é o caso do Brasil) realmente está certa.

Como já passamos por toda a procura de ferramentas, podemos pular essas etapas. Utilizarei, novamente, o programa *QGIS*, aproveitando também a disponibilidade de mapas individuais de cada

estado pelo site do IBGE. Para não me prolongar muito, farei apenas o processo completo para o estado de São Paulo. Nos demais estados, utilizarei o mapa original (não farei contornos).

Iniciando, realizei um rápido contorno no território de São Paulo, não me atentando aos mínimos detalhes. Foram contabilizados 150 pontos, um pouco menos que para o contorno do mapa do Brasil. Para esse mapeamento, obtive os seguintes dados: área = 251.578,337 km^2, perímetro = 3.012,463 km e centroide = $(733835,579; 7535996,578)$.

A área real do estado de São Paulo é cerca de 248.219 km^2, então a aproximação realizada apresentou um erro de 1,4%, um resultado muito melhor do que o obtido para o território brasileiro. Utilizando a ferramenta de cálculo de área do programa (que, na teoria, corrige as distorções) o valor obtido é de 251.242,648 km^2 (0,1% de diferença para a quantidade que obtemos), o que confirma a suposição de que territórios maiores apresentam maior distorção.

Em relação ao perímetro, o dado correto é 3.671 km (erro de 1,8%), então nosso resultado está dentro dos conformes. Aumentando o número de pontos, não poderíamos garantir que esse valor tenderia a se aproximar do esperado, devido ao fato de essa ser uma medida com muitas incertezas.

Finalmente para o centroide, o município encontrado foi Itapuí, como mostra a figura 9b.

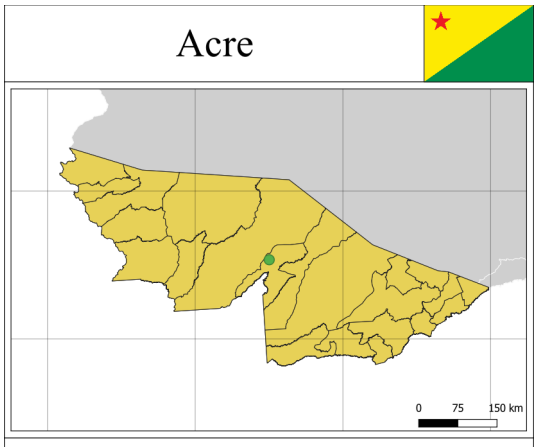
(a) Contorno do mapa

(b) Centroide no mapa

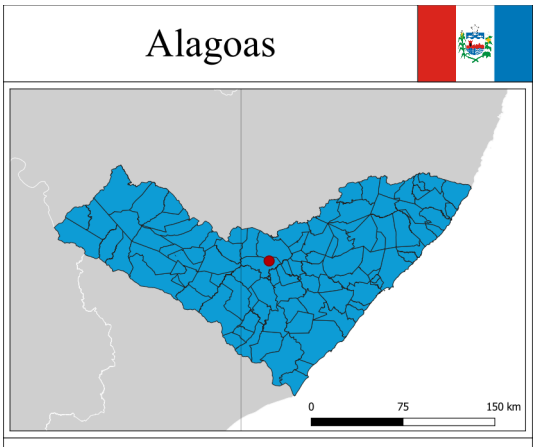
Figura 9: Mapas de São Paulo.

8.3.1 Coletânea dos estados

A seguir, deixo uma coletânea contendo área, perímetro, centroide e população ([3]) (será importante para os próximos passos da pesquisa) de todos os 26 estados brasileiros e do Distrito Federal.

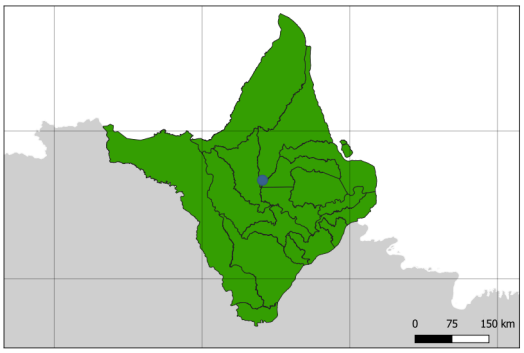


Área: 164.173,637 km²
Perímetro: 2.928,914 km
Centroide (município): Santa Rosa do Purus
População: 830.018



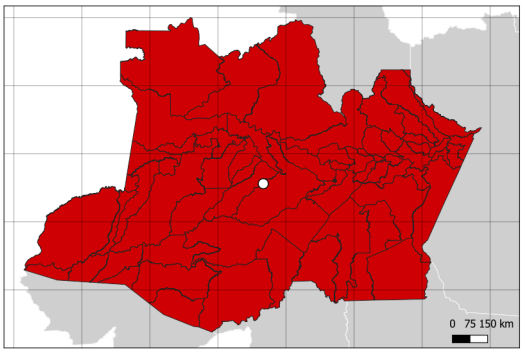
Área: 27.830,661 km²
Perímetro: 1.095,048 km
Centroide (município): Igaci
População: 3.127.683

Amapá



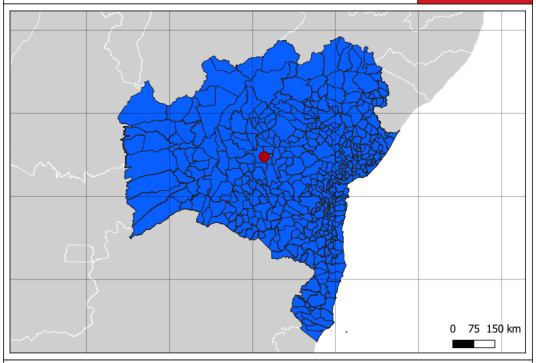
Área: 142.470,946 km²
Perímetro: 2.557,394 km
Centroide (município): Calçoene
População: 733.759

Amazonas



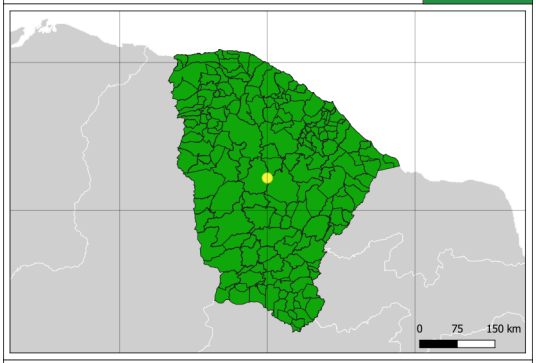
Área: 1.559.257,100 km²
Perímetro: 8.664,175 km
Centroide (município): Coari
População: 3.941.613

Bahia



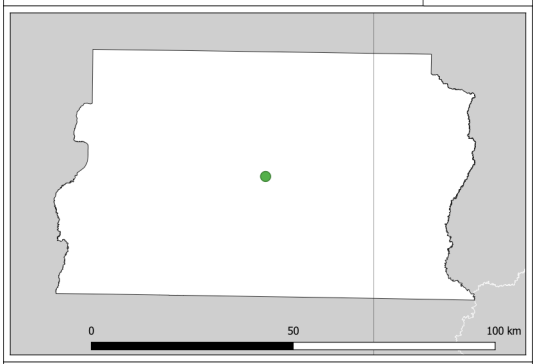
Área: 564.760,447 km²
Perímetro: 6.174,866 km
Centroide (município): Seabra
População: 14.141.626

Ceará



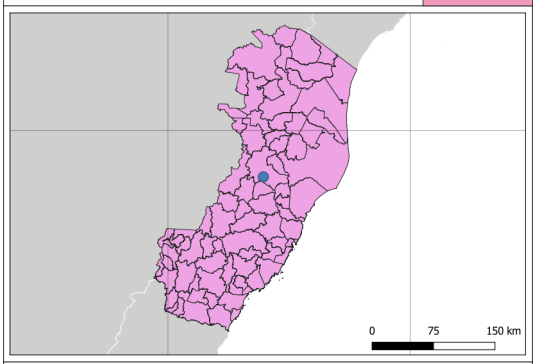
Área: 148.894,444 km²
Perímetro: 2.161,823 km
Centroide (município): Boa Viagem
População: 8.794.957

Distrito Federal



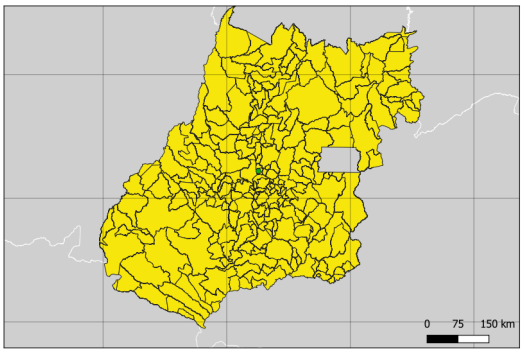
Área: 5.760,782 km²
Perímetro: 419,049 km
Centroide (município): Brasília
População: 2.817.381

Espírito Santo



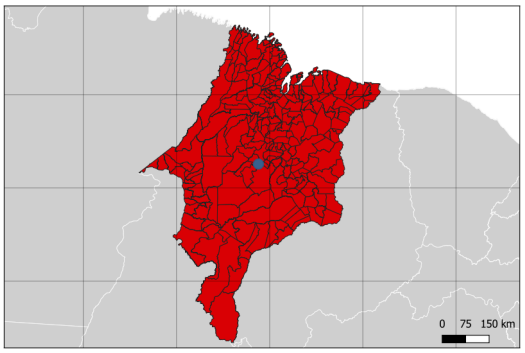
Área: 46.074,448 km²
Perímetro: 1.596,985 km
Centroide (município): Colatina
População: 3.833.712

Goiás



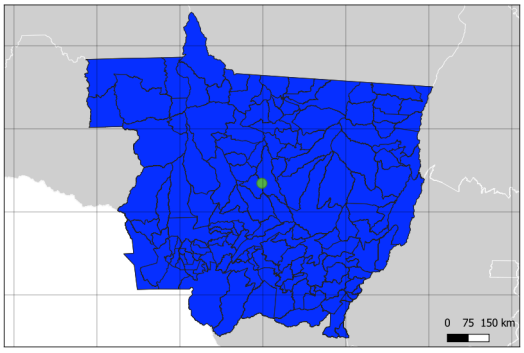
Área: 340.242,855 km²
Perímetro: 5.144,912 km
Centroide (município): Taquaral de Goiás
População: 7.056.495

Maranhão



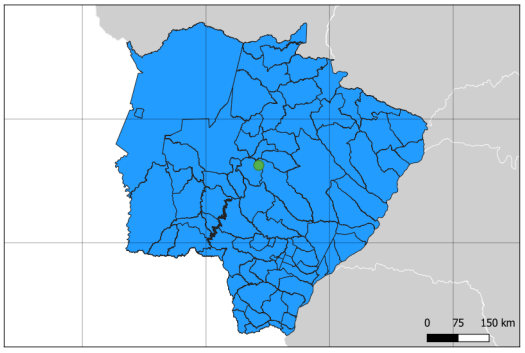
Área: 329.651,536 km²
Perímetro: 5.783,154 km
Centroide (município): São Raimundo do Doca Bezerra
População: 6.775.805

Mato Grosso



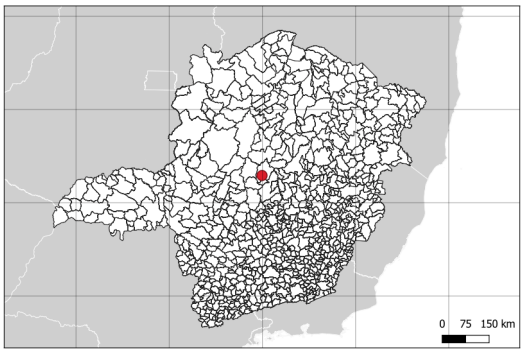
Área: 903.208,791 km²
Perímetro: 6.061,657 km
Centroide (município): Sorriso
População: 3.658.649

Mato Grosso do Sul



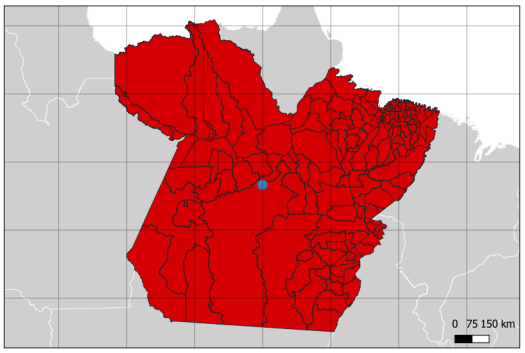
Área: 357.141,530 km²
Perímetro: 3.873,289 km
Centroide (município): Terenos
População: 2.757.013

Minas Gerais



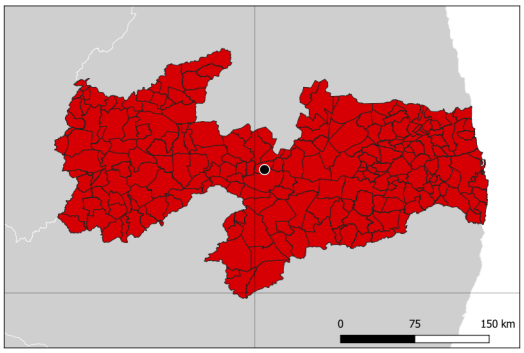
Área: 586.513,966 km²
Perímetro: 6.227,205 km
Centroide (município): Corinto
População: 20.538.718

Pará



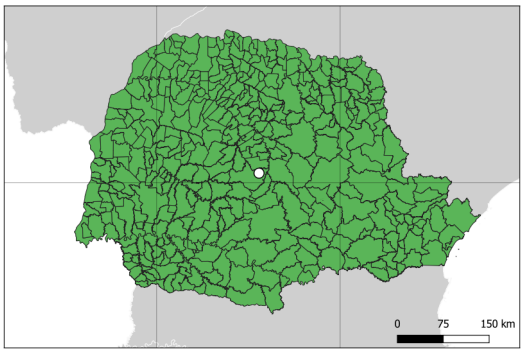
Área: 1.245.870,919 km²
Perímetro: 7.767,583 km
Centroide (município): Altamira
População: 8.121.025

Paraíba



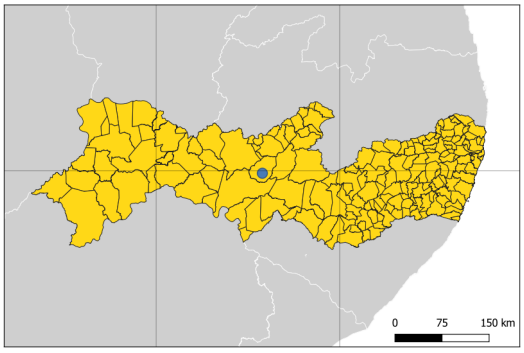
Área: 56.467,244 km²
Perímetro: 2.113,396 km
Centroide (município): Salgadinho
População: 3.974.687

Paraná



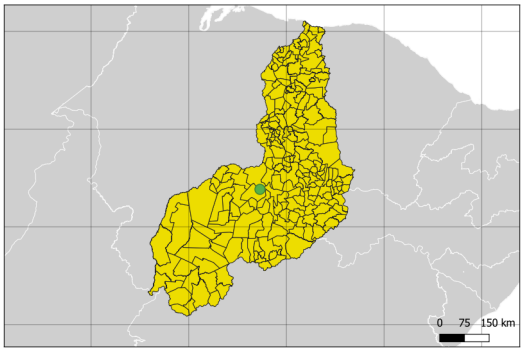
Área: 199.298,983 km²
Perímetro: 2.933,019 km
Centroide (município): Pitanga
População: 11.444.380

Pernambuco



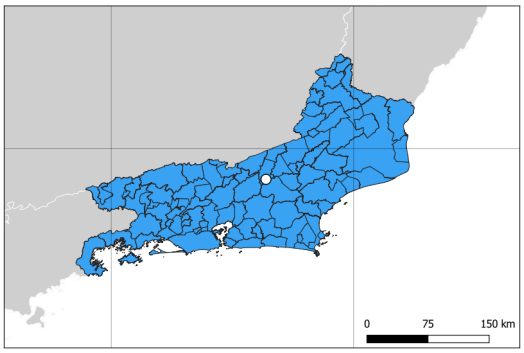
Área: 98.067,876 km²
Perímetro: 2.881,308 km
Centroide (município): Betânia
População: 9.058.931

Piauí



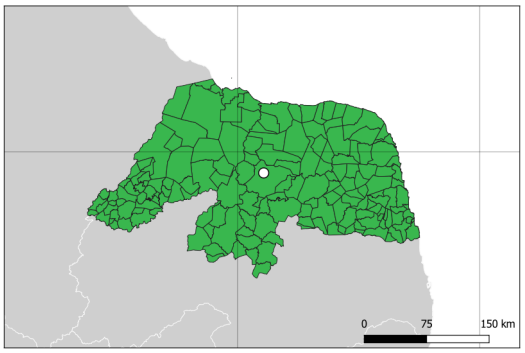
Área: 251.755,481 km²
Perímetro: 3.416,401 km
Centroide (município): Itaueira
População: 3.271.199

Rio de Janeiro



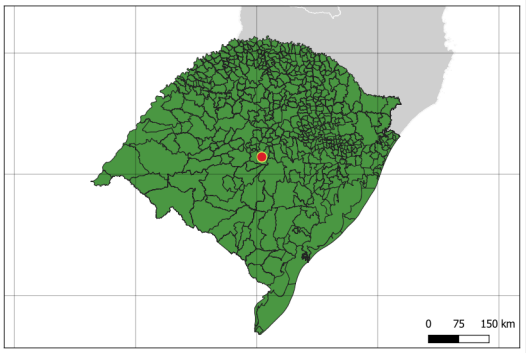
Área: 43.750,425 km²
Perímetro: 3.274,076 km
Centroide (município): Sumidouro
População: 16.054.524

Rio Grande do Norte



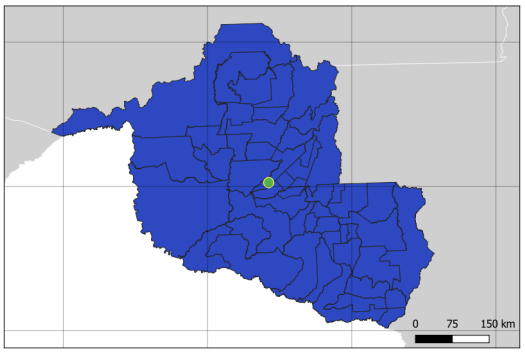
Área: 52.809,601 km²
Perímetro: 1.538,959 km
Centroide (município): Santana do Matos
População: 3.302.729

Rio Grande do Sul



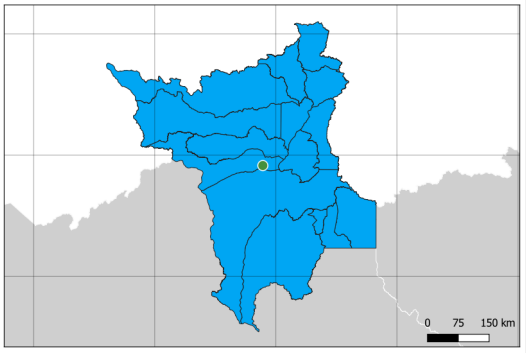
Área: 281.707,065 km²
Perímetro: 3.742,727 km
Centroide (município): Agudo
População: 10.882.965

Rondônia



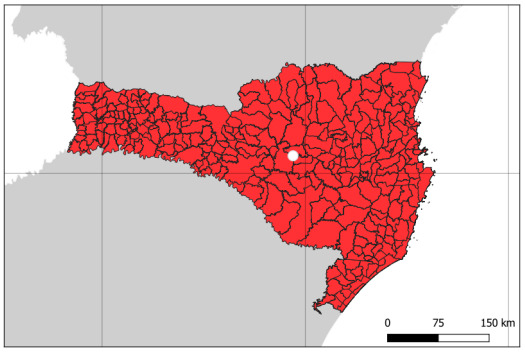
Área: 237.754,195 km²
Perímetro: 3.531,494 km
Centroide (município): Jaru
População: 1.581.196

Roraima



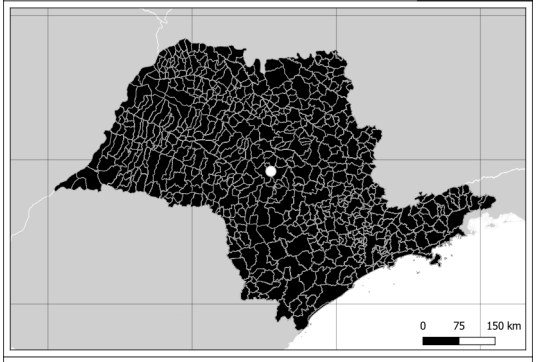
Área: 223.644,109 km²
Perímetro: 3.830,666 km
Centroide (município): Iracema
População: 636.707

Santa Catarina



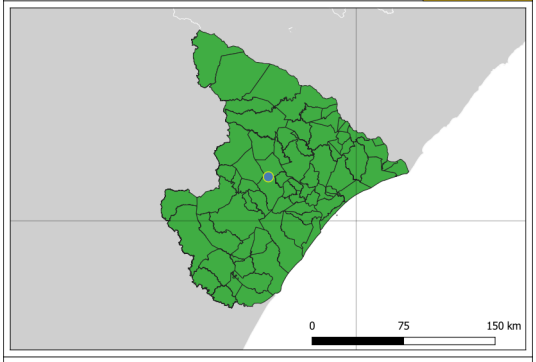
Área: 95.730,682 km²
Perímetro: 3.328,877 km
Centroide (município): São Cristóvão do Sul
População: 7.610.361

São Paulo

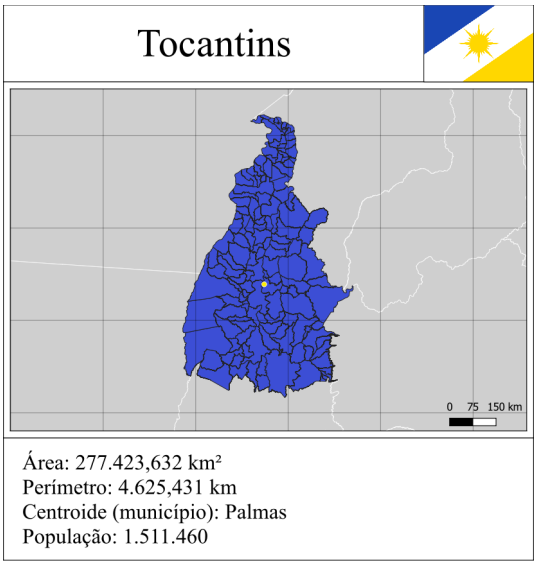


Área: 248.219,482 km²
Perímetro: 4.361,948 km
Centroide (município): Itapuí
População: 44.411.238

Sergipe



Área: 21.938,188 km²
Perímetro: 994,610 km
Centroide (município): Ribeirópolis
População: 2.209.558


Tocantins
0 75 150 km
Área: 277.423,632 km²
Perímetro: 4.625,431 km
Centroide (município): Palmas
População: 1.511.460

9. Ampliando o conceito de centroide

Durante todo o desenvolvimento realizado até agora, consideramos o centroide como sendo apenas o centro geométrico de uma região. Todos os cálculos feitos nos dois primeiros capítulos foram derivados com base nesse conceito.

Porém, é interessante pensar que, para certas situações, talvez existam pontos centrais que possam ser mais vantajosos para uma certa necessidade. Por exemplo, caso seja necessário construir um terminal de ônibus em uma região estratégica, visando facilitar o acesso de todos, encontrar o centro populacional do local, e não o geométrico, acabaria sendo o ideal. É exatamente isso que veremos nessa etapa do projeto.

9.1 Centroides com pesos

Para resolver a situação do exemplo anterior, podemos definir pontos de aglomeração da população ao longo do mapa e fornecer pesos, relacionados com o número de pessoas na região, para cada um. Finalmente, buscaríamos uma maneira de encontrar um possível centro populacional baseado nos dados coletados.

Por meio da leitura do artigo [2], que relata a maneira como o autor calcula centroides do território dos Estados Unidos, fui apresentado à fórmula utilizada para calcular o centro de uma região aplicando pesos nos pontos. É ela a seguinte:

Fórmula do Centroide com peso

$$G' = \frac{1}{M}\left(\sum_{i=1}^{n} m_i x_i, \sum_{i=1}^{n} m_i y_i\right).$$

Em um primeiro momento, ela parece ser relativamente mais simples que as demais equações vistas anteriormente, e isso realmente é verdade. O índice n dos somatórios representa a quantidade de pontos que escolhemos, m_i o peso de cada ponto (x_i, y_i) e M a soma de todos os pesos.

Novamente no exemplo utilizado, temos que cada m_i da fórmula representa o número de pessoas em cada região que definimos pelos pontos de aglomeração, e M a população total. Inicialmente, nossos cálculos não dependem da geometria do mapa total (já que as fronteiras não tem peso, a menos que definamos tal). Note também que essas contas são indiferentes em relação à ordenação dos pontos (comutatividade da soma), diferentemente do que foi visto e mostrado para o centroide geográfico.

Saindo um pouco de exemplos teóricos, podemos testar nosso novo conhecimento em uma situação real.

9.1.1 Entendendo a fórmula

Antes de começarmos nossos testes, é interessante buscar o real entendimento do que foi apresentado.

Como foi dito anteriormente, os centroides estão relacionados com o centro de massa de uma figura. Assim, o centroide com peso é, na verdade, o centro de massa da figura. A fórmula apresentada é semelhante aquela utilizada na física, vista em problemas de relações gravitacionais. A principal diferença é que, nesses problemas, a figura de interesse é dividida em infinitos pontos, fazendo-se necessária a substituição do somatório pela integral. Para o nosso caso, considerar apenas alguns pontos finitos já é o suficiente.

Entender a fórmula apresentada é relativamente simples. Para esse caso, estamos apenas considerando a média aritmética ponderada dos pontos que definimos. Com isso, o centroide com peso (ou centro de massa), da maneira que nós utilizamos, é bem mais fácil de compreender que o centro geométrico aplicado em todos os testes anteriores. Porém, ele é a base para entendermos o primeiro centroide visto, que é apenas uma ramificação desse novo conceito.

9.1.2 A verdadeira natureza do centro geométrico

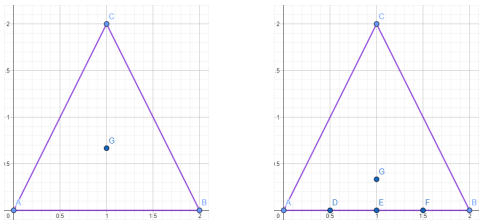
Tendo compreendido o conceito de centro de massa, podemos finalmente encontrar a verdade sobre o centro geométrico aplicado em todo o projeto.

Iniciando da fórmula vista há pouco, podemos tomar todos os pesos m_i como sendo iguais, já que não queremos distinguir os pontos da figura. Assim, $m_i = m$ é um valor fixo. Porém, apenas realizando essa alteração, não chegamos às integrais utilizadas na dedução da fórmula do centroide. Na verdade, alcançamos apenas uma equação que fornece a média aritmética dos pontos, que pode sim ser utilizada para o cálculo do centroide, mas que apresenta algumas limitações.

$$G \approx \frac{m}{M}\left(\sum_{i=1}^{n} x_i, \sum_{i=1}^{n} y_i\right).$$

Como $M = n \cdot m$, podemos simplificar a fração de fora para $\dfrac{1}{n}$, resultando exatamente na média aritmética.

Um dos principais problemas desse método é que, para um desequilíbrio na quantidade de pontos em um lado do polígono em relação aos outros, temos que o centroide tende a se aproximar deste lado, mesmo que a figura descrita seja exatamente a mesma.



(a) 3 pontos (b) 6 pontos

Figura 10: Diferentes centroides para a mesma figura.

Para contornar esse problema, podemos dividir a figura em pontos igualmente distribuídos, obtendo, assim, uma fórmula exata que sempre fornece o mesmo centroide para uma mesma figura. Dessa maneira, repartimos a figura (tanto a fronteira quanto o interior) em infinitos pontos de comprimento dx e altura dy, o que implica na alteração do somatório para uma integral. Como esses pontos estão distribuídos por toda a área do polígono, utilizamos integrais duplas, garantindo que todos contribuam para o cálculo realizado.

$$\sum_{i=1}^{n} x_i \implies \iint_D x\,dx\,dy \quad \text{e} \quad \sum_{i=1}^{n} y_i \implies \iint_D y\,dx\,dy,$$

onde D representa a região de interesse.

Por último, temos que a somatória do número de pontos, n, agora deve ser feita como a soma de todos os pontos do polígono, ou seja,

$$n = \iint_D dx\,dy = A,$$

resultando exatamente na área da figura, assim como tínhamos visto na fórmula inicial.

Finalmente, isso nos leva à conclusão de que o centro geométrico de um polígono é, na verdade, apenas a média aritmética de seus pontos vista de uma maneira contínua.

9.2 Centro populacional do Brasil

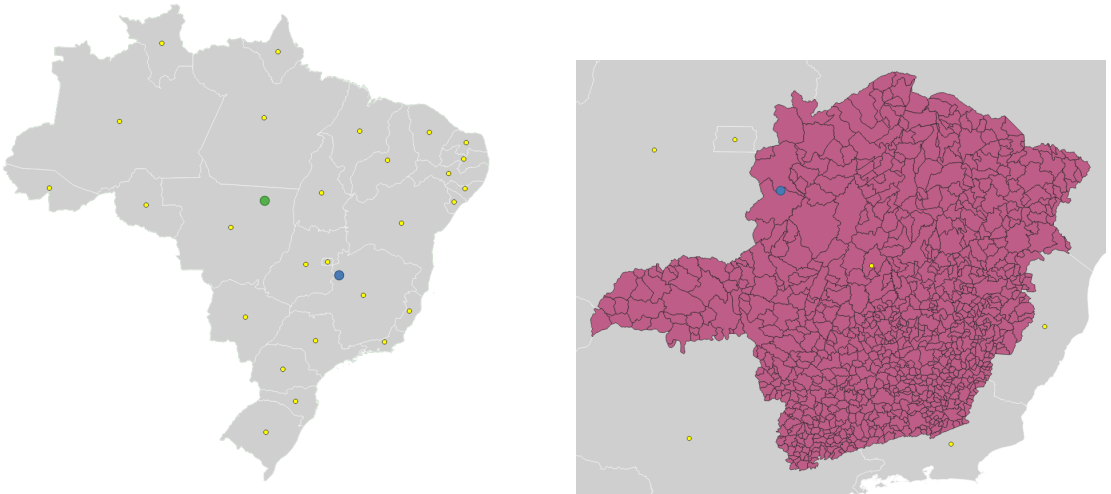
Seguindo tudo que já fizemos no capítulo anterior, vamos aplicar as fórmulas no território brasileiro. Para tal, definiremos pontos de aglomeração no centroide de cada estado, e relacionaremos o peso de cada um como a população do estado em questão (por isso adicionei essa informação na coletânea feita anteriormente).

Assim como feito para os cálculos anteriores, criei um código em Python que, dados as coordenadas dos pontos e seus respectivos pesos, calcula o centroide com peso de interesse. Sua forma é a seguinte:

```
n = int(input("Numero de pontos:"))
Cx = 0; Cy = 0; M = 0
for i in range(n):
    x, y, m = [float(v) for v in input().split()]
    Cx += m*x
    Cy += m*y
    M += m
Cx = Cx / M
Cy = Cy / M
print("Coordenadas do centroide =", (Cx, Cy))
```

Aproveitando que já temos todos os dados necessários para os cálculos, podemos simplesmente aplicá-los no nosso programa e adicionar o centroide populacional no mapa. Temos o seguinte resultado:

$$\text{Centroide populacional} = (952.778,893725652;\ 8.130.299,985955003).$$



(c) Centroides no mapa

(d) Região ampliada

Figura 12: Centro Populacional do Brasil.

Para o primeiro mapa, temos os centroides de cada estado em amarelo, o centroide geográfico em verde e o centroide populacional em azul, localizado em Minas Gerais. No segundo mapa, temos esse

estado ampliado, o que permitiu que a cidade de Paracatu fosse identificada como centro populacional do Brasil. Como esperado, os pesos aproximaram o centroide da região Sudeste, que é de longe a mais populosa do país.

9.3 Outros centroides

As possibilidades que esse novo conceito de centroide permite são muitas. Utilizando os mesmos 27 pontos utilizados no exemplo anterior, podemos simplesmente alterar os pesos de cada um e obter outros centroides. Realizei o centro populacional pois os dados sobre a população de cada estado são de fácil acesso, mas qualquer assunto pode ser utilizado.

Voltando um pouco para o centro geométrico, encontrei, durante as pesquisas para este trabalho, um fato interessante: a cidade de Palmas, em Tocantins, possui um marco histórico do centro geométrico brasileiro na Praça dos Girassóis, onde fica o Monumento à Geodésia. Porém, como vimos no capítulo anterior, o centroide verídico do Brasil fica na cidade de São José do Xingu, no Mato Grosso. Não sei exatamente o porquê de Palmas possuir esse monumento, mas o curioso é que esta é a única capital que é o centro geométrico de seu estado (mas o marco não se refere a isso).

Figura 13: Foto do Monumento à Geodésia.

10. Interpolação de pontos

Quando lidamos com figuras contendo diversos pontos, como é o caso dos mapas utilizados nos capítulos anteriores, temos que conectar tais pontos com segmentos de retas se apresenta como uma boa alternativa para essa aproximação, visto que esses segmentos são quase imperceptíveis devido ao seu tamanho. Na verdade, essa é a maneira utilizada por computadores para criar gráficos de funções: para um grande número de pontos calculados, aproximar a função entre os pontos por segmentos acaba sendo aceitável, e não afeta o estudo da função.

Porém, quando não temos tantos pontos à disposição, essa aproximação acaba ficando menos adequada, já que, visivelmente, não teremos um gráfico suave. Buscando resolver esse impasse, foram desenvolvidas algumas técnicas, que serão estudadas nesse capítulo do desenvolvimento.

10.1 Conectando pontos através de uma função polinomial

Inicialmente, podemos buscar encontrar, dado pontos em um gráfico (pontos para os quais $x_i < x_{i+1}$), uma função que passe por esses pontos, cumprindo o nosso requisito. Para isso, podemos utilizar os dados fornecidos para montar um simples sistema que encontra os coeficientes de um polinômio. Escolhendo o grau certo desse polinômio, temos que os valores obtidos são únicos.

Por exemplo, para os pontos $(0,1)$, $(1,2)$ e $(2,1)$, podemos fazer o seguinte:

$$p(x) = ax^2 + bx + c,$$

$$\begin{cases} p(0) = c = 1 \\ p(1) = a + b + c = 2 \\ p(2) = 4a + 2b + c = 1 \end{cases}$$

Escolhendo um polinômio de grau $(n-1)$, onde n é o número de pontos fornecidos, obtemos um sistema quadrado, ou seja, de solução única. Note que o grau do polinômio pode ser maior do que $(n-1)$, mas nunca menor (na realidade, pode sim ser menor, porém isso envolveria um problema de quadrados mínimos, que não é nosso foco).

Resolvendo o sistema, chegamos nos valores $a = -1, b = 2$ e $c = 1$. Plotando os pontos e a função no *GeoGebra* (já utilizado anteriormente), obtemos:

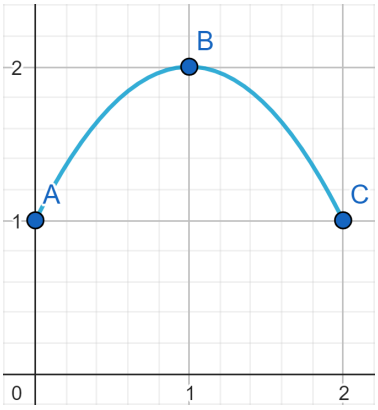


Figura 14: Interpolação com polinômio de grau 2.

Contudo, esse método apresenta um ponto negativo que precisa ser considerado. Caso busquemos aproximar o gráfico de uma função por um polinômio, temos que, com mais pontos disponíveis, a qualidade da aproximação não necessariamente se torna melhor. Na verdade, o polinômio só e recomendado até o grau 4, pois, além desse valor, ele começa a oscilar muito, afetando a suavidade desejada (a primeira derivada de polinômios de grau elevado apresenta muitos zeros da função, causando essa oscilação).

Para analisar isso, podemos tomar, para os mesmos três pontos, um polinômio de grau 3. Temos:

$$p(x) = ax^3 + bx^2 + cx + d$$

$$\begin{cases} p(0) = d = 1 \\ p(1) = a+b+c+d = 2 \\ p(2) = 8a+4b+2c+d = 1 \end{cases} \implies (a,b,c,d) = \left(\frac{c-2}{2}, \frac{4-3c}{2}, c, 1 \right)$$

Como temos mais incógnitas do que equações, o sistema apresenta infinitas soluções. Escolhendo $c = 1$, chegamos no seguinte:

$$p(x) = -\frac{x^3}{2} + \frac{x^2}{2} + x + 1$$

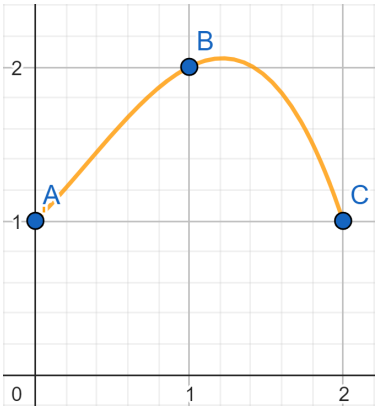


Figura 15: Interpolação com polinômio de grau 3.

Observe que, como o grau desse polinômio é maior do que o do obtido anteriormente, ele apresenta uma oscilação levemente maior, e isso é algo que piora ainda mais com graus maiores. Por causa disso, é interessante buscar métodos de interpolação que minimizem esse defeito. E assim faremos:

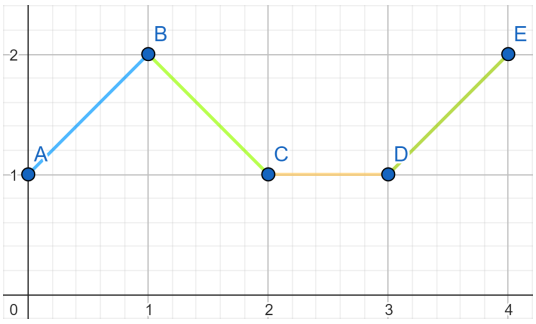
10.2 Melhorando a qualidade da interpolação

Como muitos pontos levam a um polinômio de grau elevado, e isso não seria o ideal, podemos dividir os pontos em pequenos intervalos e, então, criar polinômios interpoladores para esses agrupamentos de

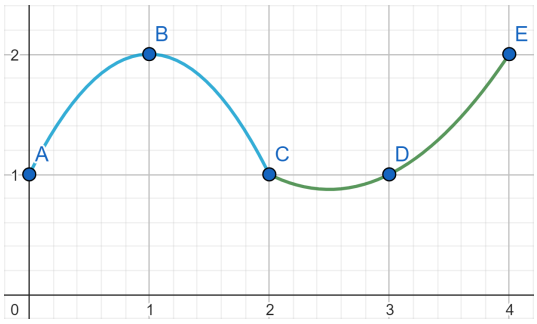
pontos. Isso é o que fizemos para os mapas, por exemplo. Pontos consecutivos eram ligados com um polinômio de grau 1 (reta).

A quantidade de pontos presente em cada intervalo acaba sendo arbitrária. Podemos escolher intervalos que melhor realizam a tarefa de nosso interesse. Além disso, o grau dos polinômios também é de nossa escolha, desde que este grau seja maior ou igual a quantidade de pontos no intervalo menos um $(n-1)$.

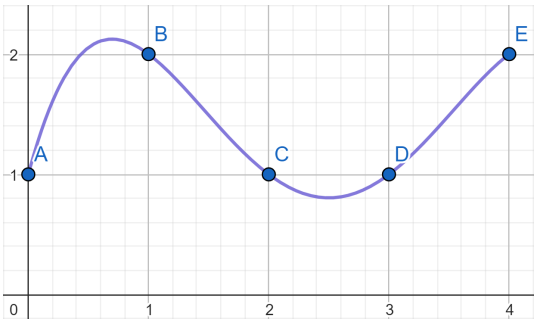
Adicionando os pontos $(3,1)$ e $(4,2)$ ao nosso teste anterior, podemos escolher intervalos com dois pontos (interpolação por retas), intervalos com 3 pontos (polinômio de grau 2) ou até mesmo mesclar os dois (mas isso não traria uma imagem tão suave para o gráfico). Após isso, criamos sistemas para cada intervalo e, então, encontramos o polinômio interpolador referente aos pontos considerados. Podemos, então plotar os resultados e analisar o que obtivemos (figura 16).



(a) 4 intervalos



(b) 2 intervalos



(c) 1 intervalo

Figura 16: Diferentes intervalos para os mesmos pontos.

Mesmo com uma redução das curvas muito acentuadas no gráfico, chegamos em um novo problema que novamente traz a necessidade de um aprimoramento dos métodos: a conexão entre dois polinômios interpoladores não acontece de forma suave. Isso acontece pois, mesmo que estas funções apresentem os mesmos valores no ponto de encontro ($p_i(x_j) = p_{i+1}(x_j) = y_j$), suas derivadas diferem para esse mesmo ponto ($p_i'(x_j) \neq p_{i+1}'(x_j)$), o que implica no encontro súbito dos polinômios no ponto. Isso novamente acaba sendo um problema que reduz a suavidade do gráfico, fazendo-se necessário aprimorar a técnica.

10.2.1 À procura da suavidade

Para solucionar nossos problemas, existe um método muito utilizado que consegue encontrar polinômios interpoladores que seguem a nossa última exigência para a suavidade: as derivadas das funções são iguais nos pontos de encontro. Tal método, denominado **spline cúbico**, utiliza, como o próprio nome sugere, polinômios de grau 3 na interpolação, porém, dessa vez, relacionando as derivadas das funções no sistema formado.

Para garantir a suavidade do gráfico, esse método impõe que os polinômios interpoladores, suas derivadas de primeira ordem e também suas derivadas de segunda ordem (algo que vai além do que buscávamos no início) sejam contínuas no intervalo total. Isso resolve o problema de conexão abrupta visto nos exemplos anteriores.

Como já utilizamos um polinômio de grau 3 no exemplo da figura 15, sabemos que ele apresenta 4 graus de liberdade (variáveis). Novamente considerando os cinco pontos $(0,1),(1,2),(2,1),(3,1)$ e $(4,2)$, formaremos uma interpolação que utiliza 4polinômios cúbicos no total (p_0, p_1, p_2 e p_3), totalizando 16 incógnitas. Para encontrar os valores desses coeficientes, utilizamos os valores dos pontos dados e de suas derivadas (primeira e segunda ordem).

Porém, como as derivadas dependem dos polinômio de intervalos seguidos, e não apenas de um único intervalo, precisamos criar um único sistema que englobe todos os polinômios do intervalo total, e então solucioná-lo. Dessa maneira, precisamos de 16 equações para garantir que o sistema tenha solução única.

Inicialmente, temos as seguintes informações:

- 8 equações pelos pontos dados e pela continuidade dos polinômios:

$$\begin{cases} p_0(0) = 1 \\ p_0(1) = p_1(1) = 2 \\ p_1(2) = p_2(2) = 1 \\ p_2(3) = p_3(3) = 1 \\ p_3(4) = 2; \end{cases}$$

- 3 equações pela continuidade das derivadas de primeira ordem:

$$\begin{cases} p_0'(1) = p_1'(1) \\ p_1'(2) = p_2'(2) \\ p_2'(3) = p_3'(3); \end{cases}$$

- 3 equações pela continuidade das derivadas de segunda ordem:

$$\begin{cases} p_0''(1) = p_1''(1) \\ p_1''(2) = p_2''(2) \\ p_2''(3) = p_3''(3). \end{cases}$$

Ao todo, temos 14 equações, sendo 16 necessárias para garantir a unicidade de solução do sistema. Como utilizamos todas as informações referentes aos pontos dados, as últimas duas informações são obtidas por uma imposição realizada sobre a interpolação. Para este exemplo, utilizarei o **spline cúbico natural**, que consiste em definir as segundas derivadas dos pontos inicial e final como nula ($p_0''(0) = p_3''(4) = 0$).

Como, para muitos pontos, o sistema pode ficar muito grande, é interessante utilizar métodos computacionais para solucioná-lo (o sistema é de tamanho $4(n-1)$ x $4(n-1)$, onde n representa o número de pontos).

Solucionando o sistema obtido e plotando os polinômios, chegamos em:

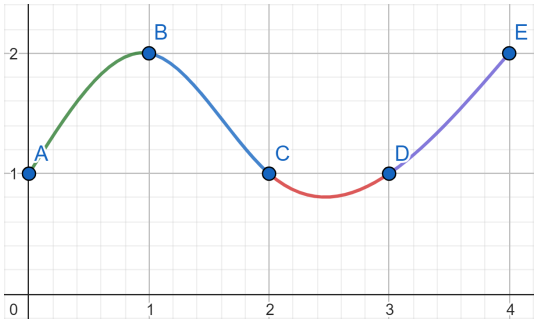


Figura 17: Exemplo utilizando o spline cúbico.

Nota: o método do spline cúbico possui uma representação muito mais formal do que aquela apresentada aqui. Busquei apenas trazer o entendimento fundamental da técnica, já que o foco do trabalho não é estudar esse spline.

10.3 Um novo método

No último exemplo, vimos que o spline cúbico está em um outro nível em relação aos outros métodos de interpolação. A curva obtida por essa técnica cumpre bem o requisito de suavidade, apresentando um resultado satisfatório.

Porém, mesmo com todas as qualidades, esse caminho ainda possui duas principais falhas: primeiro, temos que não é possível adicionar um novo ponto a uma curva já feita sem que seja necessário alterar todos os polinômios do intervalo total, e segundo, não conseguimos contornar figuras onde a coordenada y não é uma função de x, ou seja, não podemos aplicar essas técnicas no contorno de mapas.

Visando solucionar esses defeitos, temos um novo método, apresentado a mim pelo artigo [11], que traz uma visão diferente para o problema. Desta vez, não precisamos lidar com as derivadas dos polinômios, mas ainda assim temos que a primeira derivada seja contínua no intervalo (o que já é suficiente para apresentar um aspecto suave).

Essa técnica produz, ao invés de polinômios de grau 3, curvas parametrizadas, sendo uma para cada intervalo de dois pontos, assim como no spline cúbico. Essa diferença é o que expande a interpolação para o contorno de figuras. Além dessa, outra grande sacada dessa técnica é considerar, além dos dois pontos do intervalo em questão $[x_k, x_{k+1}]$, os pontos x_{k-1} (antecessor) e x_{k+2} (sucessor).

Basicamente, devemos construir duas curvas parametrizadas (uma passando pelo 3 primeiros pontos e a outra pelos 3 últimos) e então produzir uma curva geral como combinação destas. Fazemos isso para todos os pares de pontos do gráfico. Note que, como os pontos x_0 (primeiro) e x_n (último) não possuem, respectivamente, antecessor e sucessor, podemos apenas considerar o ponto sucessor (para x_0) e antecessor (para x_n) e produzir apenas uma curva, sem a necessidade de combinar duas (apenas para curvas abertas).

Exemplificando, sejam os pontos A, B, C e D quatro pontos de interesse que queremos interpolar. Considerando o intervalo $[B,C]$, primeiramente devemos parametrizar duas curvas, uma passando pelos pontos A, B e C e outra por B, C e D. Para isso, seja $(x_{AC}(t), y_{AC}(t))$ as equações da primeira curva e $(x_{BD}(t), y_{BD}(t))$ as equações da última. Como temos 4 pontos ao todo, podemos escolher quatro valores de t $(-1, 0, 1 \text{e } 2)$, um correspondendo a cada ponto, e, então, montar dois simples sistemas. Lembremos também que o grau dos polinômios do sistema equivale a $n-1$.

$$(x_{AC}(t), y_{AC}(t)) = \begin{cases} (x_{AC}(-1), y_{AC}(-1)) = A \\ (x_{AC}(0), y_{AC}(0)) = B \\ (x_{AC}(1), y_{AC}(1)) = C \end{cases} \qquad (x_{BD}(t), y_{BD}(t)) = \begin{cases} (x_{BD}(0), y_{BD}(0)) = B \\ (x_{BD}(1), y_{BD}(1)) = C \\ (x_{BD}(2), y_{BD}(2)) = D \end{cases}$$

Feito isso, podemos, enfim, encontrar a parametrização para a interpolação dos pontos B e C. Tomemos $x(t) = (1-t)x_{AC}(t) + t x_{BD}(t)$ e $y(t) = (1-t)y_{AC} + t y_{BD}(t)$. Para $t \in [0,1]$, temos que está curva conecta os pontos B e C, exatamente como buscávamos $((x(0), y(0)) = (x_{AC}(0), y_{AC}(0)) = B$ e $(x(1), y(1)) = (x_{BD}(1), y_{BD}(1)) = C)$. Esse processo deve ser repetido para cada par de pontos no intervalo de interesse. Como, nesse exemplo, os pontos A e D não possuem antecessor e sucessor (curva aberta), respectivamente, podemos considerar, para o intervalo $[A,B]$, $(x_{AC}(t), y_{AC}(t))$ com $t \in [-1,0]$ e, para o intervalo $[C,D]$, $(x_{BD}(t), y_{BD}(t))$ com $t \in [1,2]$.

Aplicando essa nova estratégia no nosso exemplo dos cinco pontos, obtemos o seguinte gráfico:

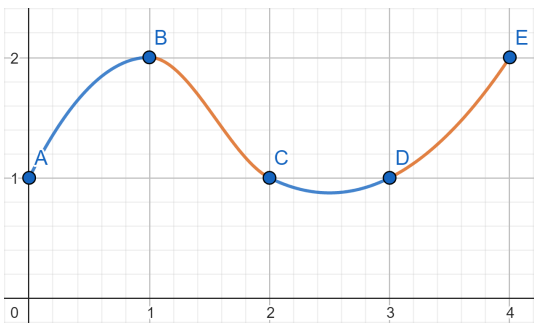


Figura 18: Exemplo utilizando o método paramétrico.

10.3.1 Comparando as técnicas

Para os exemplos feitos, talvez não fique tão claro a diferença entre todos os métodos apresentados. Porém, o artigo utilizado na última seção traz um bom exemplo, e que vale a pena ser comentado aqui.

Basicamente, o autor traz pontos que, conectados, formam o perfil da parte superior de um carro. Ao todo, são utilizados 11 pontos $(-10,1),(-9.6,1.1),(-9.1,2.2),(-6.7,2.3),(-3.8,3.9),(-1,4),(1.7,3.8),(4.5,2.3),(8,$ e $(10,1)$, o que representa uma quantidade que torna as diferenças mais visíveis. Ele realiza três interpolações: retas, spline cúbico e o método paramétrico.

Para verificar os métodos, refiz esses mesmos testes, agora com a adição de uma quarta interpolação que utiliza um único polinômio (neste caso, um polinômio de grau 10). Os resultados obtidos estão na figura 19.

Por meio das imagens, podemos finalmente ver com clareza como a suavidade da interpolação evolui através de cada método. Além disso, é importante ressaltar o trabalho necessário para aplicá-los: a interpolação por retas é a mais simples de todas; a interpolação com polinômio único trouxe um sistema de $11x11$; o spline cúbico apresentou um sistema de $40x40$ (!!!), que eu claramente não resolvi na mão; por fim, o método paramétrico apresenta dois sistemas $3x3$ por intervalo que não dependem das curvas de outros intervalos, o que representa uma ótima característica desta técnica.

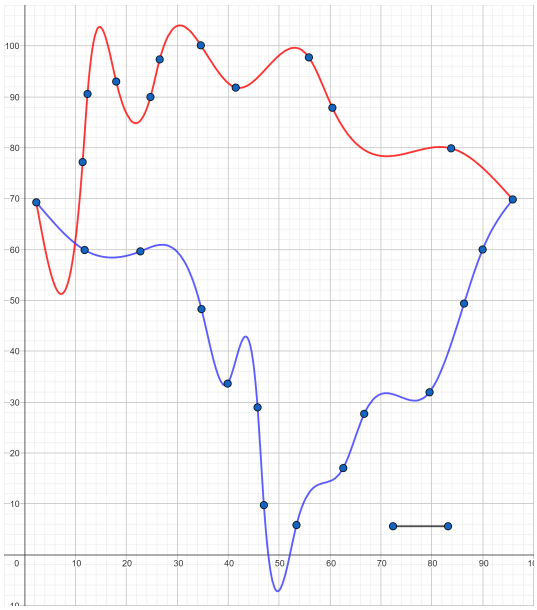
10.4 Interpolação na integração

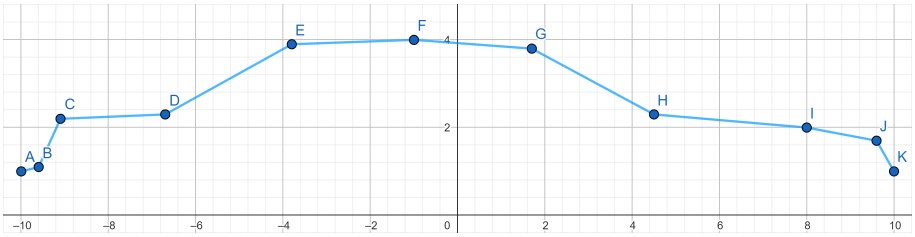
Voltando para o cálculo da área de um mapa, temos que, para poucos pontos, o valor obtido pode não ser tão preciso. Com as novas técnicas em mão, podemos testar se, no lugar de simples retas, a interpolação por outros métodos pode compensar essa baixa quantidade de pontos e trazer um melhor resultado. Para isso, utilizaremos o contorno de 24 pontos do território brasileiro da seção 2.2.

10.4.1 Aproximação pelo spline cúbico

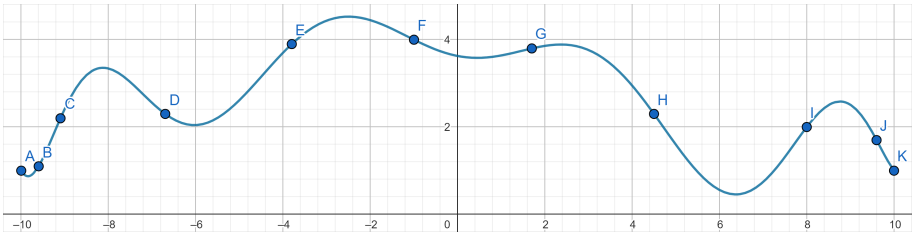
Para conectar pontos pelo método do spline cúbico, precisamos garantir que estes pontos estejam dispostos de tal maneira que $x_i < x_{i+1}$. Sabendo disso, o contorno que realizei para os 24 pontos foi de tal maneira que isso foi obedecido para os pontos superiores e inferiores do mapa (por isso comentei anteriormente que a baixa precisão foi proposital).

Caso o parágrafo anterior não tenha ficado muito clara, quis dizer que, dividindo o mapa entre pontos do norte e pontos do sul, temos que os pontos de cada agrupamento seguem a regra necessária. Dessa maneira, determinei o spline cúbico de cada grupo, chegando ao seguinte resultado:

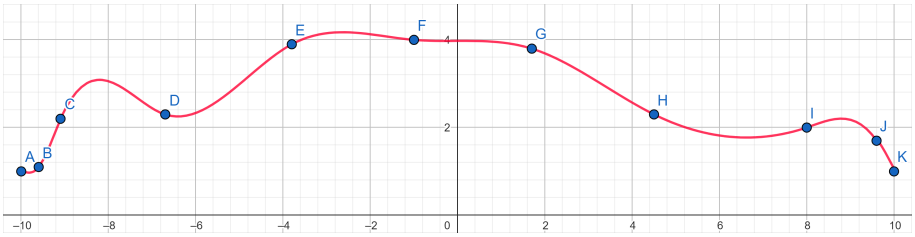




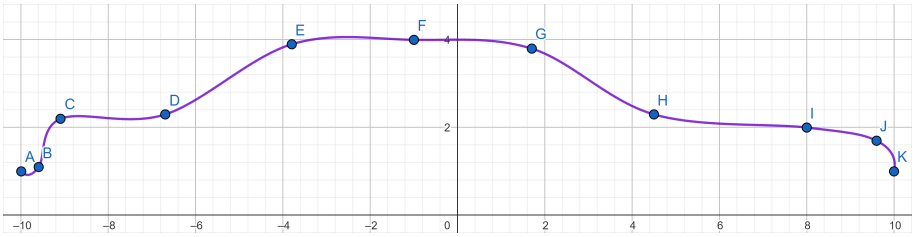
(a) Interpolação por retas



(b) Interpolação com polinômio único



(c) Spline cúbico



(d) Método paramétrico

Figura 19: Diferenças entre os métodos.

Figura 20: Contorno pelo spline cúbico.

Como é possível observar, esse método não é nada efetivo para o contorno de figuras, já que, além da grande variação entre pontos, as duas funções se interceptam no primeiro intervalo. Mesmo assim, podemos verificar a área da região (ou algo próximo disso).

No gráfico, o spline de cada agrupamento está separado pela cor (vermelho e azul). Como temos os polinômios de cada subintervalo, podemos calcular, assim como David Richeson fez no artigo [2] (e como comentei na seção 1.1.3), a integral referente a cada um desses intervalos. Então, teremos a área sob os pontos do grupo norte e sob os pontos do sul. Finalmente, teremos que a área do mapa é a diferença desses dois valores.

Spline Cúbico	
Grupo	**Área**
Norte	8.741,8719467467
Sul	4.073,2611691
Diferença	4.668,6107776467

Assim como antes, esses não são os valores reais do mapa. Utilizando α^2 para a conversão, chegamos em

$$\text{Área} = 9.978.038,046 \text{ km}^2,$$

que representa um valor muito pior do que o obtido anteriormente (9.272.036,076 km^2), mas isso já era esperado pela visualização do gráfico. Para esse propósito, o método do spline cúbico não é nenhum pouco eficiente.

10.4.2 Aproximação pelo método paramétrico

Utilizando agora o método paramétrico, que, para o propósito de contorno de figuras, é mais eficiente, já que não requer que $x_i < x_{i+1}$, temos uma pequena mudança na estratégia de integração.

Antes de passar para essa parte, apliquei o método para o mesmo contorno de 24 pontos, obtendo o seguinte gráfico:

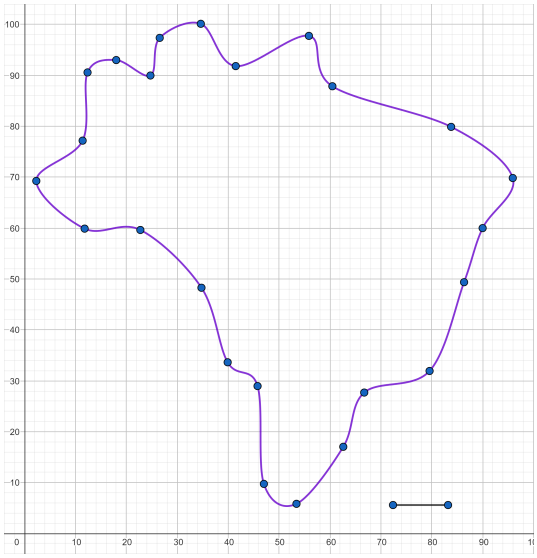


Figura 21: Contorno pelo método paramétrico.

Agora, como estamos lidando com curvas ao invés de polinômios, não podemos simplesmente integrar nos subintervalos. Porém, retornando ao que foi feito na seção 1.1.1, onde estávamos desenvolvendo a fórmula para a área de um polígono, temos que, após aplicar o Teorema de Green na integral dupla, chegamos em uma integral de linha, onde realizamos a partição da curva total em subintervalos (intervalos iguais ao do processo de interpolação).

Como, naquele momento, estávamos utilizando retas para conectar os pontos, nossa parametrização realizada para resolver as integrais de linha foi igual para todos os subintervalos, e era referente exatamente às retas empregadas. Dessa vez, como já temos a parametrização de cada intervalo, podemos aplicá-las nas integrais de linhas, porém teremos que fazer isso separadamente para cada subintervalo, já que cada um possui sua própria curva parametrizada.

Após um longo trabalho de resolução das integrais de linha para cada curva, finalmente chegamos ao valor para a área do mapa aproximado pelo método paramétrico:

$$\text{Área} = 4.381,90655 \cdot \alpha^2 \implies 9.365.276,386 \text{ km}^2,$$

o que também representa um resultado pior do que aquele alcançado com as retas (pouca diferença), mas muito melhor do que o obtido pela interpolação com spline cúbico.

Concluído todo esse processo, é importante ressaltar que ele é realmente trabalhoso. Isso, somado com os resultados melhores obtidos anteriormente, mostram que a simples utilização de retas para o cálculo da área, bem como para o cálculo do perímetro e do centroide, é uma opção muito melhor do que os métodos de interpolação apresentados. Contudo, temos que a interpolação, em específico o método paramétrico, é útil exatamente para fornecer uma melhor representação de uma figura, já que compensa a baixa quantidade de pontos no gráfico.

11. Considerações finais

Gostei muito de desenvolver o projeto "Tudo tem seu centro". Dos três projetos PICME que fiz nos últimos semestres, esse foi o que mais consegui me aprofundar no estudo. Achei muito interessante todas as conexões de diferentes conhecimentos para o objetivo do trabalho. Cada capítulo feito representou um novo mundo de conceitos, que ainda assim contribuíram para o projeto.

Antes deste projeto, nunca tinha sido apresentado às fórmulas para o cálculo da área de um polígono genérico, tão pouco ao conceito de centroide. A possibilidade de eu mesmo deduzir estes itens foi algo muito importante para o real entendimento do conteúdo.

Além disso, parte do conteúdo encaixou exatamente com aulas que eu tive durante o semestre, como foi com a parte de interpolação, onde estudei em Cálculo Numérico, que cursei durante esses meses. Também foi uma ótima oportunidade de rever alguns conceitos de cálculo II, e conseguir entender um pouco mais suas aplicações.

A minha experiência com os projetos PICME nesses três semestres que participei foi algo extremamente positivo. Os temas propostos pelo Professor Lúcio são sempre muito interessantes, sobre coisas que mostram aplicações dos conhecimentos vistos em aulas. A maneira que suas orientações são feitas também permitem grande liberdade ao aluno para buscar seus próprios caminhos no desenvolvimento. Para a parte de mapeamento, por exemplo, utilizar o *QGIS* e o próprio *GeoGebra* foram soluções que eu tive que buscar, e acredito que outros alunos que fizeram este projeto antes de mim também encontraram outros meios para o desenvolvimento da pesquisa.

Por fim, estou muito empolgado com os próximos passos no PICME. Penso que, a cada trabalho feito, consigo melhorar pouco a pouco minhas habilidades. Olhando meus últimos relatórios, é nítido que muita coisa mudou. Para esse semestre, como me comprometi em escrever pequenas partes da pesquisa a cada reunião, consegui explorar muito mais cada etapa, além de que as descrições ficaram muito mais precisas sobre o que foi feito. Isso certamente seria algo que eu recomendaria a outros alunos, e para mim mesmo no passado.

Bibliografia

[1] Albert Einstein. *Como vejo o mundo*, 2017. Nova Fronteira.

[2] David Richeson. *Centers of the United States*, The College Mathematics Journal, vol. 36, no. 5, pp. 366–373, 2005. Taylor & Francis.

[3] Wikipédia. *Lista de unidades federativas do Brasil por população*. Disponível em: https://pt.wikipedia.org/wiki/Lista_de_unidades_federativas_do_Brasil_por_populaÃğÃčo. Acesso em: 5 de junho, 2024.

[4] IBGE. *Malha Municipal*. Disponível em: https://www.ibge.gov.br/geociencias/organizacao-do-territorio/malhas-territoriais/15774-malhas.html. Acesso em: 5 de maio, 2024.

[5] GeoGebra. *Calculadora Gráfica*. Disponível em: https://www.geogebra.org/calculator. Acesso em: 1 de maio, 2024.

[6] Google Maps. *Brasil*. Disponível em: https://www.google.com/maps/@-14.5151244,-52.5165345,4.67z?entry=ttu. Acesso em: 1 de maio, 2024.

[7] Wikipédia. *Lista de municípios do Acre*. Disponível em: https://pt.wikipedia.org/wiki/Lista_de_munic%C3%ADpios_do_Acre. Acesso em: 5 de junho, 2024.

[8] Wikipédia. *Lista de municípios de Alagoas*. Disponível em: https://pt.wikipedia.org/wiki/Lista_de_munic%C3%ADpios_de_Alagoas. Acesso em: 5 de junho, 2024.

[9] Wikipédia. *Lista de municípios do Amapá*. Disponível em: https://pt.wikipedia.org/wiki/Lista_de_munic%C3%ADpios_do_Amap%C3%A1. Acesso em: 5 de junho, 2024.

[10] Wikipédia. *Lista de municípios do Amazonas*. Disponível em: https://pt.wikipedia.org/wiki/Lista_de_munic%C3%ADpios_do_Amazonas. Acesso em: 5 de junho, 2024.

[11] Wilbur J. Hildebrand. *Connecting the dots parametrically: an alternative to cubic splines*, The College Mathematics Journal, vol. 21, no. 3, pp. 208–215, 1990. Taylor & Francis.

[12] Ricardo Biloti. *O problema de interpolação*. Disponível em: https://www.ime.unicamp.br/~biloti/an/211/interp-01.html. Acesso em: 20 de junho, 2024.

[13] USP. *Splines Cúbicos*. Disponível em: https://edisciplinas.usp.br/pluginfile.php/6619417/mod_resource/content/2/ep2_2021.pdf. Acesso em: 8 de julho, 2024.

[14] QGIS. *Bem-vindo ao projeto QGIS*. Disponível em: https://www.qgis.org. Acesso em: 6 de maio, 2024.

[15] Wikipédia. *Centro geográfico*. Disponível em: https://pt.wikipedia.org/wiki/Centro_geogr%C3%A1fico#:~:text=06âĂš%2005âĂş%20O-,Brasil,fica%20o%20Monumento%20%C3%A0%20Geod%C3%A9sia.. Acesso em: 15 de junho, 2024.

[16] Wikipédia. *Lista de municípios do Piauí*. Disponível em: https://pt.wikipedia.org/wiki/Lista_de_munic%C3%ADpios_do_Piau%C3%AD. Acesso em: 29 de junho, 2024.

[17] Wikipédia. *Lista de municípios do Rio Grande do Sul*. Disponível em: https://pt.wikipedia.org/wiki/Lista_de_munic%C3%ADpios_do_Rio_Grande_do_Sul. Acesso em: 2 de julho, 2024.

II Uma Introdução a Simulações Estocásticas por meio do Modelo Binomial

Resumo

SVMs com Gradiente Estocástico:
Uma Introdução a Simulações Estocásticas por meio do Modelo Binomial

Clayton Alves Luiz

Orientador: Prof. Pedro José Catuogno

Este artigo é um estudo do modelo binomial para uso em derivativos, inicialmente há uma introdução sobre a teoria utilizada,posteriormente temos a aplicação destes conceitos através da linguagem de programação Python, há também uso de métodos estatístico e do método de Monte Carlo. É um trabalho introdutório sobre processos estocásticos voltados ao mercado financeiro, mais especificamente a derivativos.

12. Introdução e desenvolvimento

12.1 Introdução

O modelo binomial associado a Random Walks podem descrever muitas aplicações no mercado financeiro, em especial os derivativos. Uma vez que é possível, através do uso de tempos discretos, simular o comportamento de uma opção, por meio de poucos parâmetros, porém é necessário um embasamento teórico para tais simulações. Neste artigo escrevo um pouco da teoria e da prática destes processos.

12.2 Desenvolvimento das atividades

Inicialmente houve uma reunião com o professor Pedro José Catuogno, na qual optamos por encontros semanais, ás quartas feiras no período da tarde. O professor me passou alguns livros como base teórica para que eu estudasse ao longo da semana e levasse as dúvidas para ele na nossa reunião semanal. Os temas do material teórico foram probabilidade, estatística e finanças matemática, pois queríamos observar os fenômenos estocásticos no mercado de derivativos.

Iniciamos o estudo teórico por meio do modelo binomial Cox-Ross & Rubinstein (CRR), pois conseguimos descrever o comportamento dos derivativos de maneira simplificada, além de ser facilmente convertido em um projeto computacional.

13. Um pouco da teoria

13.1 Modelo de um período

Neste modelo o valor de uma ação no tempo 0, $S_0 > 0$, é conhecido e no tempo 1 , S_1 ,possui apenas duas possibilidades ou sobe, $S_1 > S_0$, ou desce $S_1 < S_0$, o qual é determinado por um evento aleatório. Esse evento aleatório pode ser comparado com o lançamento de uma moeda, dado que são apenas duas possibilidades, então adotaremos a subida como (H, cara) e a descida como (T, coroa) com probabilidade $p \neq 1$ e $p \neq 0$,caso contrario teríamos um evento determinístico e não aleatório. Tomamos p como probabilidade de subida (H), logo $1 - p$ é a probabilidade de queda (T) do derivativo.

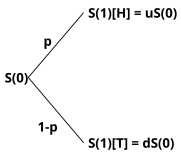


Figura 13.1: Modelo binomial de um período

Na figura acima, $up = u$ e $down = d$, com $d < u$. O valor de S_0 é conhecido e o valor de S_1 depende do lançamento de uma moeda.

Um conceito muito importante no mercado financeiro é o de arbitragem. É um estado no qual, a partir de uma carteira de ativos sem custo, não existe a possibilidade de prejuízo e há possibilidade de lucro. Um dos princípios do mercado é a ausência de arbitragem permanente, pois ela pode causar grandes danos em uma economia. Logo sua existência se restringe a curtos períodos de tempo.

Hipóteses necessárias ao modelo

Para que o modelo binomial de opções faça sentido é necessário algumas hipóteses:

- As ações podem ser subdivididas quantas vezes forem necessárias.
- Juros recebidos em aplicações em renda fixa são iguais aos juros aplicados em empréstimos
- Preço de compra e venda de uma ação é igual
- A ação assume apenas dois valores em terminado tempo

- A taxa de juro (r) é sempre maior que -1
- A ação tem valor maior que zero o que implica $u > 0$ e $d > 0$

Ausência de arbitragem no modelo

Suponha uma carteira de ativos $X_0 = 0$, uma taxa de juros r, para renda fixa, u e d como os valores de up e down para uma terminada ação. Nesse cenário existem algumas possibilidades.

- $1 + r < d < u$
  Nesse caso poderíamos pegar dinheiro emprestado em alguma instituição financeira e aplicar tudo em ações, assim no tempo 1 teríamos um lucro garantido, o que caracteriza uma arbitragem
- $d < u < 1 + r$
  Nesse caso compramos ações a descoberto, isto é ficamos com uma quantidade negativa de ações na carteira,e imediatamente aplicamos esse valor em um renda fixa,assim no tempo 1 teríamos um lucro garantido, o que novamente caracterizaria uma arbitragem.
- $d < 1 + r < u$
  A única configuração que possibilita a ausência de arbitragem.

Com isso chegamos a condição de ausência de arbitragem, a qual será utilizada em todo o artigo.

$$d < 1 + r < u. \tag{13.1}$$

Probabilidade Neutra ao Risco

A probabilidade neutra ao risco são valores relacionados ao preço justo de um ativo, isto é, garantem a ausência de arbitragem. Elas respeitam as seguintes características:

$$\tilde{p}, \tilde{q} \geq 0, \qquad \tilde{p} + \tilde{q} = 1, \qquad \tilde{p}u + \tilde{q}d = 1 + r. \tag{13.2}$$

Com isso chegamos a definição de $\tilde{p}$ e $\tilde{q}$:

$$\tilde{p} = \frac{1 + r - d}{u - d}, \qquad \tilde{q} = \frac{u - (1 + r)}{u - d}. \tag{13.3}$$

O valor de uma opção

Considere uma carteira de ativos de valor $X_0 = V_0$ no tempo 0, a partir de aplicações em renda fixa e compra de ações queremos chegar ao valor $X_1 = V_1$ no tempo 1. No tempo 1 o valor V_1 possui duas possibilidades $V_1(H)$ e $V_1(T)$. Além disso, o valor da ação no tempo 0 é S_0. Agora é possível replicar o valor da opção:

1. Suponha que no tempo 0 eu compre Δ_0 ações que valem S_0 e o restante $(X_0 - \Delta_0 S_0)$ seja aplicado em renda fixa
2. Agora no tempo 1,as ações possuem o valor $\Delta_0 S_1$, o qual depende do lançamento de uma moeda, e o valor da aplicação em renda fixa é $(X_0 - \Delta_0 S_0)(1 + r)$

Logo nossa carteira no tempo 1 valerá:

$$X_1 = \Delta_0 S_1 + (X_0 - \Delta_0 S_0)(1 + r). \tag{13.4}$$

Porém, sabemos que S_1 possui dois possíveis valores, $S_1 = uS_0$ e $S_1 = dS_0$, então podemos resolver um sistema linear:

$$\begin{cases} X_1(H) = V_1(H) = \Delta_0 u S_0 + (X_0 - \Delta_0 S_0)(1 + r), \\ X_1(T) = V_1(T) = \Delta_0 d S_0 + (X_0 - \Delta_0 S_0)(1 + r). \end{cases}$$

Resolvendo o sistema e considerando uma solução única,isto é o determinante da matriz gerada é diferente de zero, chegamos a conclusão que $r > -1$ e $d \neq u$, que são nossas hipóteses para o modelo. Além disso:

$$\Delta_0 = \frac{V_1 - V_0}{S_1 - S_0}, \qquad V_0 = \frac{1}{1 + r}(\tilde{p} V_1(H) + \tilde{q} V_1(T)). \tag{13.5}$$

13.2 Modelo de vários períodos

O problema da explosão combinatória

Quando tentamos adaptar o modelo de 1 período para o modelo de n períodos surge um problema computacional, pois se tentarmos valores de u e d diferentes, a árvore binomial não vai se recombinar,isto é, os nós da árvore binomial não se encontrarão em um dado momento com o mesmo número de ups e downs, por exemplo em $S_2(HT)$ e $S_2(TH)$ temos o mesmo valor para o derivativo. Isto é importante, pois com a recombinação temos um crescimento linear e com isso um custo computacional menor. Caso contrário, teríamos um gasto de memória exponencial o que inviabilizaria o modelo. Além disso, como estamos observando derivativos, podemos assumir r constante ao longo do período analisado.

Opção no modelo de vários períodos

Semelhante ao modelo de um período, podemos descobrir o valor de uma opção voltando através dos galhos da árvore binomial e descontando a taxa de juros correspondente. Mas, é importante observar que a posição de V_N na árvore depende dos ω primeiros lançamentos de moeda(H ou T). Assim, adaptando a fórmula 13.5, e considerando $0 \leq n \leq N-1$, temos:

$$V_n(\omega) = \frac{1}{1+r}\Big(V_{n+1}(\omega H)\tilde{p} + V_{n+1}(\omega T)\tilde{q}\Big). \tag{13.6}$$

Além disso, também é possível calcular o número de ações para replicar essa carteira, uma vez que esta também depende dos n lançamentos e da sequência ω de resultados, assim:

$$\Delta_n(\omega) = \frac{\Delta V_{n+1}(\omega)}{\Delta S_{n+1}(\omega)}. \tag{13.7}$$

O valor de $V_n(\omega)$ é a média ponderada dos valores de $V_{n+1}(\omega)$ descontado uma taxa de juros, o equivalente a voltar um intervalo de tempo. Também podemos escrever a fórmula 13.7 da seguinte forma:

$$V_n(\omega) = \frac{1}{1+r}\sum V_{n+1}(\omega)P(X). \tag{13.8}$$

Onde X é uma variável aleatória que assume os valores $\tilde{p}(\omega)$ e $\tilde{q}(\omega)$. Fazendo mais algumas manipulações,temos:

$$V_n(\omega) = \sum \frac{V_{n+1}(\omega)P(X)}{1+r} = \tilde{E_n}\left[\frac{V_{n+1}}{1+r}\right]. \tag{13.9}$$

Onde $\tilde{E_n}$ é a esperança com as probabilidades neutras ao risco. Em particular podemos calcular o valor de V_0 a partir da equação acima por um processo de recursão. Assim para o uma opção com vencimento em N, temos:

$$V_0 = \tilde{E}\left[\frac{V_N}{(1+r)^N}\right]. \tag{13.10}$$

Temos $(1+r)^N$, pois utilizaremos a fórmula 13.9 a cada valor $n \in 0 \leq n \leq N$, o que seriam N vezes.

Embora estamos analisando o valor de uma opção com base em u,d e r constantes,também é possível variá-los de acordo com os lançamentos de uma moeda, dessa forma teríamos um $r(\omega)$, $u(\omega)$ e um $d(\omega)$, e consequentemente, é necessário adaptar a fórmula da probabilidade condicional.

$$\tilde{p}(\omega) = \frac{1+r(\omega)-d(\omega)}{u(\omega)-d(\omega)}, \qquad \tilde{q}(\omega) = 1-\tilde{p}(\omega). \tag{13.11}$$

13.3 Processo estocástico e processo estocástico adaptado

Um processo estocástico é uma sequência de variáveis aleatórias X_n, de modo que a cada $n \in \mathbb{N}$ exista um valor para X. Note que estamos observando um processo aleatório discreto, pois $n \in \mathbb{N}$, porém ele pode ser contínuo, com $n \in \mathbb{R}$.

Um processo estocástico adaptado depende somente dos n primeiros termos, em um espaço amostral,com $n \in \mathbb{N}$, isto é no tempo n, o valor de X_n é totalmente conhecido, além disso com a proximidade de n, o valor de X_n vai ser tornando cada vez mais preciso.

Martingale

É um tipo especial de processo estocástico adaptado. Olhando para finanças, são processos imprevisíveis, isto é, não é possível determinar com exatidão se em um $t_n > t_{n+1}$ o derivativo irá subir ou descer. A vantagem desse processo é que não existe arbitragem,uma vez que é impossível se beneficiar disso. Além disso, o valor esperado é constante,isto é o valor presente é a esperança do valor futuro.

$$M_n = E[M_{n+1}] \tag{13.12}$$

Um importante exemplo de martingale é descrito pela fórmula 13.9.

Random Walk

Considere um ponto que pode se mover para a direita na reta dos números inteiros, isto é ter sua posição acrescida em 1 unidade, ou pode se mover para a esquerda na reta dos números inteiros, ter sua posição decrescida uma unidade, com probabilidades iguais. Supondo que ele saia da posição zero e considerando sua natureza probabilística, podemos dizer que é um processo estocástico e mais especificamente um martingale, uma vez que não sabemos para onde o ponto vai no tempo t_{n+1}, mas temos certeza da sua posição no tempo t_n. O valor da sua esperança é $E_n = 0$, mas sua variância tem valor $W_n = n$. Esse tipo de movimento é chamado de Random Walk e é muito empregado em finanças. Abaixo um exemplo de código e de gráfico desse movimento.

```
1  import numpy as np
2  import matplotlib.pyplot as plt
3  walk = []
4  n = 500
5  k = 0
6  for i in range(n):
7      x = np.random.rand(1)
8      if x < 0.5:
9          k = k - 1
10         walk.append(k)
11     if x > 0.5:
12         k = k + 1
13         walk.append(k)
14 eixo_x = [i for i in range(501) if i
       ↪ >= 1]
15 plt.plot(eixo_x, walk)
16 plt.show()
```

Código 13.1: Código em Python

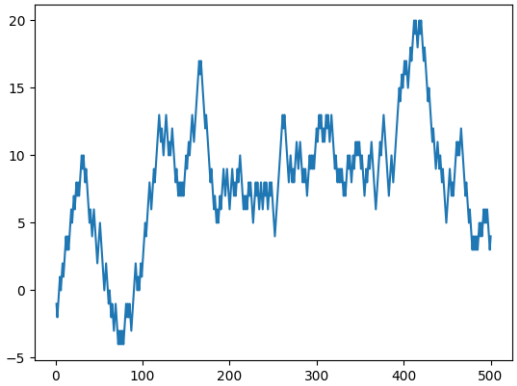


Figura 13.2: Exemplo de Random Walk

14. Um pouco da parte Prática

14.1 Tratamento dos Dados

Para aplicar o conhecimento adquirido da parte teórica queríamos usar dados reais do mercado de derivativos. É possível adquirir esses dados pelo site da B3. Os dados vem em um documento txt padronizado. Porém, não conhecia uma maneira de tratar aqueles dados e selecionar os necessários para a análise. Por conta disso, comecei a estudar um pouco da biblioteca pandas e tratamento de dados com a linguagem Python. Consegui tratar os dados e separei os valores das opções de três meses (setembro,outubro e novembro). Escolhi opções de compra da Petrobras com vencimento em dezembro, seus códigos são 'PETRL373' e 'PETRL500'. Além disso, era necessário saber o valor da taxa de juros no período analisado, usamos como referência a taxa Selic. No período analisado, ela não se manteve constante, então considerei a média pondera pelos dias da taxa Selic. Logo: $r = 12,695\%$.

Salto e Retorno

No modelo teórico utilizamos o conceito de up e down para precificar as ações, porém partindo de dados reais, não temos como saber o quanto um ativo vai subir e o quanto vai descer, por conta disso, temos a ideia de Salto e Retorno. Isto é:

$$J_n = \frac{S_{n+1}}{S_n}$$
$$H_n = \frac{\Delta S_n}{S_n} = \frac{S_{n+1} - S_n}{Sn} = J_n - 1$$

É importante notar que cada valor J_n corresponde a um up ou um down. E uma boa estimativa para a média diária desses saltos é a esperança desses valores. Assim como o desvio padrão é uma boa estimativa para a volatilidade no modelo. Além disso, esse são valores com base na probabilidade real.

$$m_{dia} \approx E[J_n] \qquad \sigma_{dia} \approx StDev[J_n] \tag{14.1}$$

Também é preciso converter o valor da taxa de juros anual para diário, pois os saltos são diários.

$$r_{dia} = \sqrt[360]{(1 + r_{ano})} - 1 \tag{14.2}$$

O modelo utilizará períodos de tempo discretos, isto é, teremos um intervalo Δt, em que os tempos para o derivativo serão $n\Delta t$, $n \in \mathbb{N}$.

14.2 Cálculos necessários para o algoritmo de mudança de escala temporal

Inicialmente temos que mostrar que o valor da esperança dos saltos é constante. Considere $P(J_n)$ as probabilidades que J_n tem de subir (p) e descer $(1-p)$, assim:

$$E[J_n] = \sum J_n P(J_n) = up + (1-p)d. \tag{14.3}$$

O que implica que o valor da esperança é constante. Além disso, podemos olhar para o $E[J_n^2]$, logo:

$$E[J_n^2] = \sum J_n^2 P(J_n) = u^2 p + (1-p)d^2. \tag{14.4}$$

Com esse valores podemos calcular a variância dos saltos.

$$\mathrm{Var}[J_n] = E[J_n^2] - (E[J_n])^2, \tag{14.5}$$
$$\mathrm{Var}[J_n] = (u^2 p + (1-p)d^2) - (up + (1-p))^2, \tag{14.6}$$
$$\mathrm{Var}[J_n] = p(1-p)(u-d)^2. \tag{14.7}$$

Logo, podemos calcular o valor do desvio padrão:

$$\sigma = \sqrt{\mathrm{Var}[J_n]},$$
$$\sigma = |u-d|\sqrt{p(1-p)}. \tag{14.8}$$

Note que temos um sistema com duas equações 14.3 e 14.8 e três incógnitas, ou seja, possui infinitas soluções,mas isso não é interessante para nós, então vamos considerar $p = \frac{1}{2}$, assim:

$$m_{dia} = \frac{u+d}{2}, \qquad \sigma_{dia} = \frac{u-d}{2}. \tag{14.9}$$

Com esses parâmetros estamos seguindo o Modelo de Jarrow e Rudd. Dessa forma conseguimos um valor para u e d, baseados no m_{dia} e σ_{dia}:

$$u_{dia} = m_{dia} + \sigma_{dia}, \qquad d_{dia} = m_{dia} - \sigma_{dia}. \tag{14.10}$$

Esse modelo faz sentido quando $u_{\Delta t}$ e $d_{\Delta t}$ são analisados em 1 dia por vez, mas podemos ter um Δt menores,porém com $\Delta t > 0$, o que deixaria o modelo mais preciso. Vamos analisar o modelo com um salto de dois em dois dias, isto é $H_n = \frac{H_{n+2}}{H_n}$,assim temos:

$$H_n = \frac{H_{n+2}}{H_n} = \frac{H_{n+2}}{H_n}\frac{H_{n+1}}{H_{n+1}} = \frac{H_{n+2}}{H_{n+1}}\frac{H_{n+1}}{H_n} = J_{n+1}J_n. \tag{14.11}$$

Montando novamente o sistema linear e assumindo $p = \frac{1}{2}$, chegamos em:

$$u_2 = m_{dia}^2 + \sigma_{dia}\sqrt{2m_{dia}^2 + \sigma_{dia}^2}, \qquad d_2 = m_{dia}^2 - \sigma_{dia}\sqrt{2m_{dia}^2 + \sigma_{dia}^2}. \tag{14.12}$$

Isso implica que conforme aumentamos ou diminuímos nosso Δt podemos reescrever os valores $H_{n+\Delta t}$ como um produtório dos valores J_n. Assim para um Δt qualquer, temos:

$$u_{\Delta t} = m_{dia}^{\Delta t} + \sqrt{(m_{dia}^2 + \sigma_{dia}^2)^{\Delta t} - (m_{dia}^2)^{\Delta t}}, \qquad d_{\Delta t} = m_{dia}^{\Delta t} - \sqrt{(m_{dia}^2 + \sigma_{dia}^2)^{\Delta t} - (m_{dia}^2)^{\Delta t}}. \tag{14.13}$$

Além disso é preciso converter a taxa de juros diário para um período Δt qualquer, assim:

$$1 + r_{\Delta t} = (1 + r_{dia})^{\Delta t}. \tag{14.14}$$

Com isso temos alguns parâmetro que possibilitam a criação de um algoritmo para converter as escalas de tempo, mas antes de ser utilizado é necessário criar algum mecanismo para garantir a ausência de arbitragem. Ou seja, é preciso satisfazer a condição abaixo:

$$0 < d_{\Delta t} < 1 + r_{\Delta t} < u_{\Delta t}. \tag{14.15}$$

Considerando $\Delta t = 1$ dia e utilizando a fórmula para o $d_{\Delta t}$, é simples ver que

$$d_{dia} > 0 \Leftrightarrow m_{dia} > \sigma_{dia}. \tag{14.16}$$

Pelo modelo utilizado,temos:

$$|m_{\Delta t} - (1 + r_{\Delta t})| < \sigma_{\Delta t}. \tag{14.17}$$

Manipulando está expressão e elevando ambos os lados ao quadrado, chegamos a condição de ausência de arbitragem do modelo Jarrow Rudd(JR):

$$\left(1 - (\frac{1 + r_{dia}}{m_{dia}})^{\Delta t}\right)^2 < \left(1 + (\frac{\sigma_{dia}}{m_{dia}})^2\right)^{\Delta t} - 1. \tag{14.18}$$

14.3 Algoritmo de Mudança de escala temporal

Este algoritmo é fundamental, pois a partir dele poderemos descobrir os valores de u,d e r em um dado Δt. Com isso será possível simular alguns caminhos de ações e garantir a ausência de arbitragem. Como entrada para o algoritmo teremos um valor Δt_{dado}, m_{dia}, σ_{dia} e r_{dia}. A partir desses valores deduziremos um valor para $u_{\Delta t}$, $d_{\Delta t}$,$r_{\Delta t}$ e Δt ($\Delta t \leq \Delta t_{dado}$) Para o modelo utilizado existem duas restrições para garantir a ausência de arbitragem (14.16 e 14.18). Enquanto a condição de ausência de arbitragem não for satisfeita, reduziremos o Δt_{dado} pela metade, dessa forma teremos o maior intervalo Δt possível. O código abaixo foi escrito em Python.

```python
def escala_tempo(dt0, r_dia, m_dia, sigma_dia):
    if m_dia <= sigma_dia:
        print("Parametros inconsistentes")
        return -1

    M0 = (1 + r_dia) / m_dia
    M1 = 1 + (sigma_dia / m_dia) ** 2
    dt = dt0

    while (1 - M0 ** dt) ** 2 >= M1 ** dt - 1:
        dt = dt / 2

    # Calculando up, down e r no intervalo dt
    m_dt = m_dia ** dt
    sigma_dt = ((m_dia ** 2 + sigma_dia ** 2) ** dt - m_dia ** (2 * dt)) ** 0.5
    u_dt = m_dt + sigma_dt
    d_dt = m_dt - sigma_dt
    r_dt = (1 + r_dia) ** dt - 1
    p = 0.5

    return dt, u_dt, d_dt, r_dt, p
```

Código 14.1: Algoritmo de mudança de escala

14.4 Simulando um caminho de ação

Agora é possível simular um caminho para uma ação dado um valor inicial S_0. Para isso, teremos como entrada um valor T que indica o tempo final em dias da simulação, o Δt calculado pelo último algoritmo, o $u_{\Delta t}$, o $d_{\Delta t}$, o $r_{\Delta t}$ e a probabilidade $p = \frac{1}{2}$. Além disso, é necessário o sorteio de um número aleatório, se esse número for menor que p teremos um up, caso contrário, um down.

```python
def run_de_acao(S0, T, dt, u_dt, d_dt, p):
    valores = []
    S = S0
```

```
    t = 0
    while t < T:
        t = t + dt
        if np.random.rand(1) < p:
            S = u_dt * S
            valores.append(S)
        else:
            S = d_dt * S
            valores.append(S)
    return valores
```

Código 14.2: Algoritmo do caminho de uma ação

A partir desse código, fiz uma simulação com o valor real de uma opção de compra da Petrobras com um T = 90 dias e $\Delta t = 0,01$.

```
x = escala_tempo(0.01, r_dia, m_dia,
    ↪ sigma_dia)
resultados = run_de_acao(array[0], 90, x
    ↪ [0], x[1], x[2], x[4])
x0 = [0.01 * i for i in range(len(
    ↪ resultados) + 1) if i >= 1]
plt.xlabel('Periodos')
plt.ylabel('Valores')
plt.plot(x0, resultados, color='blue')
plt.show()
```

Código 14.3: Run de ação 'PETRL373'

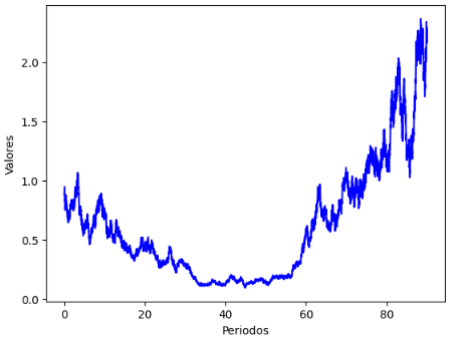


Figura 14.1: Gráfico da simulação

14.5 Monte Carlo

O método de Monte Carlo se baseia em uma repetição massiva de elementos aleatórios para estimações numéricas. Pode ser usado para calcular áreas, por exemplo. No código abaixo temos uma estimativa de π, calculado através do método de Monte Carlo.

```
n, valores_circulo = 100000, 0
for i in range(n):
    x = np.random.rand()
    y = np.random.rand()
    if x**2 + y**2 <= 1:
        valores_circulo = valores_circulo + 1
pi = 4 * valores_circulo / n
print(pi) ##retorna 3.14256
```

Código 14.4: Calculando π por Monte Carlo

Para utilizar esse método para o calculo de área, é preciso uma função objetivo, um sorteador de números aleatórios com distribuição uniforme, isto é não existe nenhuma tendência para o sorteio, e uma área de referência. Por exemplo, no código acima, o sorteador aleatório é um método da biblioteca numpy (np.random.rand()) a função objetivo é o circulo de raio unitário e a área de referência é o quadrado 1×1 no primeiro quadrante. De modo que:

$$\frac{\pi R^2}{4}\frac{1}{R^2} \approx \frac{Q_{in}}{Q_{total}}, \qquad \pi \approx \frac{4Q_{in}}{Q_{total}}.$$

Onde Q_{in} são os pontos que estão abaixo da função $x^2 + y^2 \leq 1$ e Q_{total} é o total de pontos sorteados. Este é um método muito interessante, pois é indiferente a dimensão do problema, já que necessita apenas do sorteio de um número aleatório. E chegamos a boas estimativas.

Também podemos usá-lo para calcular integrais de forma numérica, por exemplo a função $y = \frac{1}{1+x^5}$ possui primitiva muito complexa, utilizando o método de Monte Carlo se torna simples calcular o valor da área.

```
nt = 1000000
nin = 0
for i in range(nt):
    x = np.random.rand(1)
    y = np.random.rand(1)
    if y + y * x**5 <= 1:
        nin = nin + 1
area = nin / nt ##retorna 0.888354
```

Código 14.5: Calculando área por Monte Carlo

Observe que está é uma ótima estimativa, pois usando outros softwares cheguei ao valor de 0,8883.

Por conta disso, aplicar o método de Monte Carlo para opções Europeias é uma ótima ideia, pois teremos uma ótima estimativa e um custo computacional menor.

Algoritmo de Monte Carlo

Para calcular o valor presente de uma opção é necessário alguns pré requisitos, o primeiro seria diferentes runs de opções em dado período de tempo, pois quanto mais simulações houverem, mais próximo do valor real estarei. Além disso, é preciso de uma função de payoff para esta opção. Por fim, podemos utilizar a fórmula 4 para trazer essa opção a valor presente.

Porém isso só faz sentido se for feito com a probabilidade neutra ao risco. Acima, temos os códigos dos runs de caminhos e da função payoff.

```
def gena_caminho(N, S0, u, d, p, r):
    if d >= 1 + r or u <= 1 + r:
        return "Ha arbitragem"
    p = (1 + r - d) / (u - d)
    s = [0] * (N + 1)
    s[0] = S0
    m = np.random.rand(N)
    D = 1 / (1 + r) ** N
    for i in range(1, N + 1):
        if m[i - 1] < p:
            s[i] = u * s[i - 1]
        elif m[i - 1] > p:
            s[i] = d * s[i - 1]
    return s, D
```

Código 14.6: Runs para opção

```
def payoff_caminho_call(N, s):
    strike = 3.73
    V = [0] * (N + 1)
    for i in range(N + 1):
        if s[i] - strike < 0:
            V[i] = 0
        elif s[i] - strike > 0:
            V[i] = s[i] - strike
    return V[N]
```

Código 14.7: Payoff de uma opção de compra

Na figura 10 temos como entrada o número de partições desejadas N, o valor S_0 da opção,os valores de up, down e a taxa de juros em um período Δt, calculados pelo algoritmo 1. E como saída tem um vetor s com o run da ação e D que é o desconto da taxa de juros no período analisado.

Já na figura 11 temos um Payoff de uma opção de compra com strike de R$ 3.73, está é a Payoff da opção PETRL373. Ela retorna o valor V[N] da opção.

Enfim é possível escrever o algoritmo utilizando Monte Carlo.

```
def monte_carlo(S0, N, MAX, u, d, r, gera_caminho, payoff_caminho_call):
    soma = 0
    for i in range(MAX):
        s, D = gera_caminho(N, S0, u, d, r)
        soma = soma + payoff_caminho_call(N, s) * D
    return soma / MAX
```

Código 14.8: Calculando área por Monte Carlo

Esse algoritmo retorna V[0] descontado a taxa de juros e são feitas no total MAX repetições. Como o resultado desse algoritmo depende de escolhas aleatória, é interessante repeti-lo algumas vezes e usar a média dessas repetições. Utilizei esse algoritmo nas duas opções da Petrobras analisadas.

```
temp0 = escala_tempo(1, r_dia, m_dia, sigma_dia) # [dt, u_dt, d_dt, r_dt, p]
valores_V0 = []
for i in range(1000):
    T = 90 # dias
    N = int(T / temp0[0]) # Como N e o numero de particoes, faz sentido que seja
        ↪ inteiro
    a = monte_carlo(array[0], N, 15000, temp0[1], temp0[2], temp0[3], gera_caminho,
        ↪ payoff_caminho_call)
    valores_V0.append(a)

v = np.array(valores_V0)
m_V0 = np.mean(v)
m_V0 ##retorna 0.36299163618489094
```

Código 14.9: Valor médio da Opção 'PETRL373'

```
temp0 = escala_tempo(1, r_dia, m_dia, sigma_dia) # [dt, u_dt, d_dt, r_dt, p]
valores_V0 = []
for i in range(1000):
    T = 90 # dias
    N = int(T / temp0[0]) # Como N e o numero de particoes, faz sentido que seja
        ↪ inteiro
    a = monte_carlo(array[0], N, 15000, temp0[1], temp0[2], temp0[3], gera_caminho,
        ↪ payoff_caminho_call)
    valores_V0.append(a)

v = np.array(valores_V0)
m_V0 = np.mean(v)
m_V0 ##retorna 2.2260822841568007
```

Código 14.10: Valor médio da Opção 'PETRL500'

Usando com exemplo a opção de compra 'PETRL500', seu valor S_0 é R$ 5,80, então um valor V[0] de R$ 2,20 é bem aceitável, o que demonstra que o código está funcionando bem. E nesse caso seria interessante exercer a opção de compra, pois o strike é menor que o valor da opção no tempo zero, haveria uma diferença de R$ 2,20 entre o valor de compra e o valor de mercado de cada opção comprada.

Além disso, fiz um gráfico com todos os 1000 valores utilizados para calcular a média de V[0].

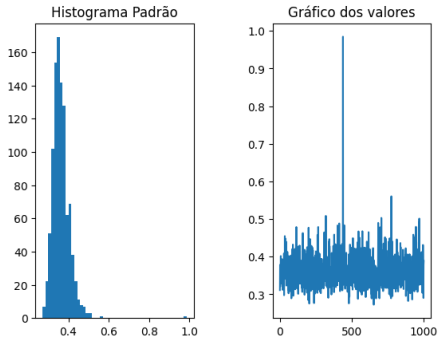


Figura 14.2: Valores V[0] da Opção 'PETRL373'

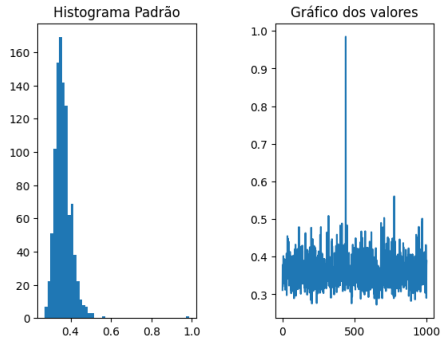


Figura 14.3: Valores V[0] da Opção 'PETRL500'

Por meio do histograma fica evidente a necessidade de fazer mais de uma repetição do código, pois existem valores extremos bem distante da média de V[0].

15. Considerações Finais

A partir desse projeto, foram utilizados muitos conceitos do mundo financeiro e da estatística, além de utilizar métodos numéricos com muitas aplicações, em especial voltado as finanças. Também foi abordado um pouco de tratamento de dados e alguns códigos em Python. Também foi visto que o método de Monte Carlo funciona muito bem com opções vanila e ao utilizar dados reais para entender essa aplicação foi possível observar que no caso da opção de compra PETRL500, por meio do nosso modelo, faria sentido fazer essa aplicação.

Podemos encontrar ótimas noções sobre o mercado financeiro e suas aplicações em [2] e em [1]. Podemos encontrar ótimas noções sobre métodos numéricos para processos estocásticos em [3].

Bibliografia

[1] Cabral, Marco Aurélio Palumbo, *Finanças Matemática: Teoria e Prática*, Instituto de Matemática, Departamento de Matemática Aplicada, (2020)

[2] Avallaneda, Marco and Laurence, Peter, *Quantitative Modeling of Derivative Securities: From Theory to Practice*. CHAPMAN & HALL/CRC, (1999).

[3] Higham, Desmond J. and Kloeden, Peter E, *An Introduction to the Numerical Simulation of Stochastic Differential Equations*. SIAM, (2021);