UNIVERSIDADE FEDERAL DE SERGIPE
PRÓ-REITORIA DE PÓS-GRADUAÇÃO E PESQUISA
CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO EM MATEMÁTICA

Lays Vanessa Santana Rodrigues

Estimativa do tensor de Ricci sobre dados discretos

Dissertação de mestrado

São Cristóvão
18 de setembro de 2025

Lays Vanessa Santana Rodrigues

Estimativa do tensor de Ricci sobre dados discretos

Dissertação de mestrado requerida pelo Programa de Pós-graduação em Matemática da Universidade Federal de Sergipe como requisito final para a obtenção do título de mestre em Matemática.

Orientador: Profº Dr. Gastão Florêncio Miranda Junior

São Cristóvão
18 de setembro de 2025

FICHA CATALOGRÁFICA ELABORADA PELO SIBIUFS
UNIVERSIDADE FEDERAL DE SERGIPE

R696e

Rodrigues, Lays Vanessa Santana
Estimativa do tensor de Ricci sobre dados discretos / Lays Vanessa Santana Rodrigues ; orientador Gastão Florêncio Miranda Junior. – São Cristóvão, SE, 2025.
73 f. : il.

Dissertação (Mestrado em Matemática) - Universidade Federal de Sergipe, 2025.

1. Matemática. 2. Geometria diferencial. 3. Geometria riemaniana. 4. Tensor de Ricci. 5. Redes neurais (Computação). 6. Aprendizado de máquina. I. Miranda Junior, Gastão Florêncio, orient. II. Título.

CDU 514.76/.77

Dissertação submetida à aprovação pelo Programa de Pós-Graduação em Matemática da Universidade Federal de Sergipe, como parte dos requisitos para obtenção do grau de Mestre em Matemática.

Estimativas do tensor de Ricci sobre dados discretos

por

Lays Vanessa Santana Rodrigues

Aprovada pela banca examinadora:

Documento assinado digitalmente
gov.br GASTAO FLORENCIO MIRANDA JUNIOR
Data: 01/09/2025 14:25:06-0300
Verifique em https://validar.iti.gov.br

Prof. Dr. Gastão Florêncio Miranda Junior - UFS
Orientador

Prof. Dr. Jônison Lucas dos Santos Carvalho - UFS
Primeiro Examinador

Documento assinado digitalmente
gov.br CAYO RODRIGO FELIZARDO DORIA
Data: 01/09/2025 21:20:27-0300
Verifique em https://validar.iti.gov.br

Prof. Dr. Cayo Rodrigo Felizardo Doria - UFS
Segundo Examinador

Documento assinado digitalmente
gov.br GILSON ANTONIO GIRALDI
Data: 01/09/2025 05:24:07-0300
Verifique em https://validar.iti.gov.br

Prof. Dr. Gilson Antonio Giraldi - UFRJ
Segundo Examinador

São Cristóvão, 28 de Agosto de 2025

Cidade Universitária "Prof. José Aloísio de Campos" – Av. Marcelo Déda Chagas, s/n - Rosa Elze – Campus de São Cristóvão. Tel. (00 55 79) 3194-6887 CEP: 49107-230 - São Cristóvão – Sergipe - Brasil – E-mail: promat@academico.ufs.br

Aos meus pais,
pelo apoio e incentivo à educação
e por acreditarem em mim.

Agradecimentos

Como diria meu querido professor Paulo: "Recordar é viver!" Por isso, vou recordar todos que me ajudaram a "viver" nessa caminhada.

Agradeço, em primeiro lugar, a Deus, por me ouvir e me dar forças para continuar. À minha família, especialmente: Ao meu pai, pelo apoio emocional e financeiro, à minha mãe (que hoje está em outro plano), pelo incentivo à educação e ao meu irmão, Neto.

Ao meu namorado, Franciel, por estar sempre comigo, cuidando e apoiando.

Ao meu querido orientador, Gastão, por me entender, confiar em mim e me apoiar. Aos examinadores da banca, pela disponibilidade em avaliar e contribuir para o aprimoramento do meu trabalho.

Aos meus mestres, em especial: Paulo Rabelo e Ivanete Batista, minhas maiores inspirações como professores.

À minha "gangue" de sempre: Duda, Elaine, Ediva, Vanessa, Angelina e Adionelle, que, apesar de seguirem trajetórias diferentes, sempre se mantiveram presentes na minha vida.

Aos meus amigos do mestrado, em especial: A turma da sala 10 (Jamisson, Angelina e Edivangel); E da sala 2 (Junior, Adriana, Samorane e Jardel), com quem criei vínculos especiais. Muito obrigada por tornarem minha trajetória mais leve.

A Ort e Junior, pela ajuda, pela disponibilidade em ler e dar sugestões para o meu trabalho.

Aos meus amigos e ex-companheiros de casa: Emisson e Gil, por toda a força, compreensão e apoio.

À minha psicóloga, Luciana, por me acompanhar e me ajudar durante todo o processo do mestrado.

Às novas pessoas que conheci (ou conheci melhor) nessa jornada: Hellen Kássia, Teixeirinha, Antônio, entre outros.

À Marilene (tia da limpeza), por todo o carinho e cuidado, e ao Neto (secretário), pelo trabalho excepcional.

A Naruto e Luffy, por me inspirarem e me motivarem.

A todos que contribuíram, direta ou indiretamente, para a minha trajetória acadêmica.

Ah! Quase ia me esquecer: A Jake e Frajola, por serem meu cachorro e meu gato (e por toda a alegria que me deram).

MUITO OBRIGADA!

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Código de Financiamento 001.

Resumo

O estudo da geometria em espaços de alta dimensão, especialmente em nuvens de pontos, apresenta desafios no cálculo e na interpretação de propriedades geométricas, como o tensor métrico e a curvatura de Ricci. Este trabalho propõe uma abordagem que integra conceitos da geometria riemanniana com ferramentas de aprendizado de máquina para estimar esses tensores em dados discretos, baseando-se em um método construído com diferenças finitas e estendendo-o por meio de uma adaptação que emprega redes neurais. A metodologia foi validada experimentalmente em superfícies bidimensionais, produzindo resultados compatíveis com os do trabalho de referência, isto é, estimativas precisas em superfícies com curvatura positiva e constante, mas apresentando limitações em superfícies com curvatura negativa. Por sua vez, os experimentos com a adaptação baseada em redes neurais mostraram resultados inferiores quando comparado ao método original, mas comparáveis, em certos aspectos, a outras abordagens já existentes, como as curvaturas de Ollivier-Ricci e Forman-Ricci. Esses resultados demonstram a viabilidade da abordagem proposta e evidenciam desafios importantes a serem enfrentados em trabalhos futuros.
Palavras-chave: Geometria Riemanniana; Aprendizado de Máquina; Redes neurais; Tensor de Curvatura de Ricci;

Abstract

The study of geometry in high-dimensional spaces, particularly in point clouds, presents challenges in the calculation and interpretation of geometric properties, such as the metric tensor and Ricci curvature. This work proposes an approach that integrates concepts from Riemannian geometry with machine learning tools to estimate these tensors in discrete data, building upon a finite difference-based method and extending it through an adaptation employing neural networks. The methodology was experimentally validated on two-dimensional surfaces, yielding results consistent with those of the reference work—that is, accurate estimates on surfaces with positive and constant curvature, but with limitations on surfaces with negative curvature. Meanwhile, experiments with the neural network-based adaptation showed inferior results when compared to the original method, though they were comparable, in certain aspects, to other existing approaches, such as Ollivier-Ricci and Forman-Ricci curvatures. These results demonstrate the feasibility of the proposed approach and highlight important challenges to be addressed in future work.

Keywords: Riemannian Geometry; Machine Learning; Neural Networks; Ricci Tensor;

Sumário

Lista de Figuras

Lista de Tabelas

Introdução

A Geometria de Riemann (ou geometria riemanniana) foi formulada pelo matemático alemão Bernhard Riemann no século XIX. Ela generaliza a Geometria Diferencial clássica, que estuda superfícies no $\mathbb{R}^3$, e um dos seus principais conceitos é o de variedade riemanniana, que é uma variedade diferenciável equipada com uma métrica específica, chamada métrica riemanniana. No estudo da geometria de variedades, a Geometria Riemanniana fornece ferramentas teóricas para descrever propriedades intrínsecas de um espaço n-dimensional, tais como: distâncias, ângulos, curvaturas e geodésicas. Essa teoria foi fundamental para a física, pois possibilitou a formulação da Teoria da Relatividade Geral de Albert Einstein e, atualmente, vem sendo aplicada em diversas outras áreas, como a de ciência de dados.

Com a disponibilidade de dados em alta dimensão o interesse em aplicar conceitos da geometria riemanniana na análise desses conjuntos de dados tem crescido significativamente. Isso se deve ao fato de que tais dados podem ser modelados como nuvens de pontos amostradas de uma variedade imersa em um espaço de maior dimensão. A motivação teórica central reside no fato de que a geometria intrínseca de uma variedade codifica toda a informação sobre sua forma e estrutura. O desafio, portanto, está em inferir essas propriedades geométricas fundamentais a partir de uma amostra discreta e finita de pontos. Conceitos como a curvatura de Ricci ou a métrica riemanniana, antes restritos ao domínio contínuo, tornam-se acessíveis numericamente. Nessa perspectiva, compreender a geometria desses dados pode impulsionar avanços em tarefas de aprendizado de máquina, como reconhecimento de padrões e classificação, transformando o que antes era visto apenas como "dados brutos" em espaços métricos ricos em informação.

Apesar de sua importância, o cálculo dessas propriedades a partir de dados discretos ainda é desafiador. Diferentemente de superfícies contínuas, onde o tensor métrico, curvaturas e geodésicas podem ser obtidos analiticamente, nuvens de pontos requerem estimativas numéricas sujeitas a ruídos, irregularidades e limitações computacionais. Nesse cenário, a interseção entre Geometria Riemanniana e Aprendizado de Máquina surge como uma ideia promissora.

As redes neurais introduzidas por McCulloch e Pitts em 1943 são inspiradas no funcionamento biológico dos neurônios. Elas passaram por um período inicial de entusiasmo em torno das décadas de 1950 e 1960, seguido de um declínio na pesquisa devido às limitações teóricas e computacionais. Esse período de crescimento atrofiado durou até 1981, sendo conhecido como *AI Winter* (inverno da inteligência artificial). A retomada dos estudos ocorreu a partir dos anos seguintes, com o avanço do algoritmo de retropropagação e, mais recentemente, foi impulsionada pela disponibilidade de grandes volumes de dados e poder computacional. Essa evolução sugere que as redes neurais podem ser aplicadas não apenas em tarefas tradicionais de predição e classificação, mas também na estimativa de propriedades geométricas de dados discretos.

Para compreender a importância do problema abordado neste trabalho, é essencial destacar dois conceitos centrais da geometria riemanniana: o tensor de curvatura de Riemann e o tensor de Ricci. O tensor de Riemann é uma aplicação que define a curvatura de uma variedade riemanniana, capturando como o espaço se desvia localmente da geometria euclidiana plana. Já o tensor de Ricci é obtido pela contração do tensor de Riemann, fornece uma descrição resumida da curvatura e é importante para diversas aplicações, incluindo a relatividade geral. No contexto de dados discretos, estimar esses tensores representa um desafio significativo, devido à ausência de uma estrutura contínua ou de uma parametrização que os definam, mesmo localmente.

Neste trabalho, buscamos implementar e adaptar o método RF-ML de (LI; LU, 2019) para estimar o tensor de curvatura de Ricci em dados discretos. Embora o artigo original tenha como objetivo principal o uso do fluxo de Ricci para redução de dimensionalidade, suas

ferramentas de estimativas servem como base para a abordagem aqui proposta. Nossa proposta de adaptação combina a ideia das estimativas do método original com técnicas de aprendizado de máquina, em específico, redes neurais. Essa integração visa, principalmente, investigar se a capacidade de aprendizado das redes pode lidar melhor com casos desafiadores, como os reportados pelos autores do RF-ML, que são as variedades com regiões de curvatura negativa.

Este trabalho está organizado da seguinte forma: no capítulo 1, são apresentados os conceitos fundamentais sobre variedades diferenciáveis, que servem de base para o estudo da geometria riemanniana, desenvolvido no capítulo 2. O capítulo 3 aborda técnicas de aprendizado de máquina relevantes para a pesquisa, incluindo métodos clássicos, como KNN e PCA, e redes neurais. Os trabalhos relacionados são apresentados no capítulo 4, e no capítulo 5, descrevemos a metodologia proposta, incluindo o método original utilizado como base e sua adaptação com redes neurais para a estimativa do tensor de Ricci em dados discretos. O capítulo 6 apresenta os experimentos computacionais realizados, com análise em diferentes superfícies clássicas, como paraboloide, esfera e sela de macaco. Por fim, o trabalho é concluído no capítulo 7, onde são discutidos os resultados obtidos e apontadas possíveis direções para trabalhos futuros.

1 Preliminares

Nesta seção, serão introduzidos conceitos essenciais para a compreensão da proposta deste trabalho. Para isso, é necessária uma breve introdução à geometria riemanniana, que será apresentada na seção 2, e servirá para modelar e entender o comportamento intrínseco de dados amostrados a partir de uma variedade suave. Neste momento, serão discutidos conceitos de variedades diferenciáveis, campos de vetores e colchetes de Lie, estabelecendo uma base teórica sólida para o desenvolvimento subsequente. As principais referências utilizadas são (CARMO, 2015) e (LEE, John M, 2006).

Definição 1 (Variedade Diferenciável)**.** *Uma variedade diferenciável de dimensão n é um conjunto $\mathcal{M}$ munido de uma família de aplicações biunívocas $\mathbf{x}_\alpha : U_\alpha \subset \mathbb{R}^n \to \mathcal{M}$, onde U_α são subconjuntos abertos do $\mathbb{R}^n$, tais que:*

1. $\bigcup_{\alpha \in L} \mathbf{x}_\alpha(U_\alpha) = \mathcal{M}$*, onde L é um conjunto de índices;*
2. *Para todos os índices $\alpha, \beta \in L$, com $W = \mathbf{x}_\alpha(U_\alpha) \cap \mathbf{x}_\beta(U_\beta) \neq \emptyset$, os conjuntos $\mathbf{x}_\alpha^{-1}(W)$ e $\mathbf{x}_\beta^{-1}(W)$ são abertos em $\mathbb{R}^n$ e as aplicações $\mathbf{x}_\beta^{-1} \circ \mathbf{x}_\alpha : \mathbf{x}_\alpha^{-1}(W) \to \mathbf{x}_\beta^{-1}(W)$ e $\mathbf{x}_\alpha^{-1} \circ \mathbf{x}_\beta : \mathbf{x}_\beta^{-1}(W) \to \mathbf{x}_\alpha^{-1}(W)$ são diferenciáveis.*

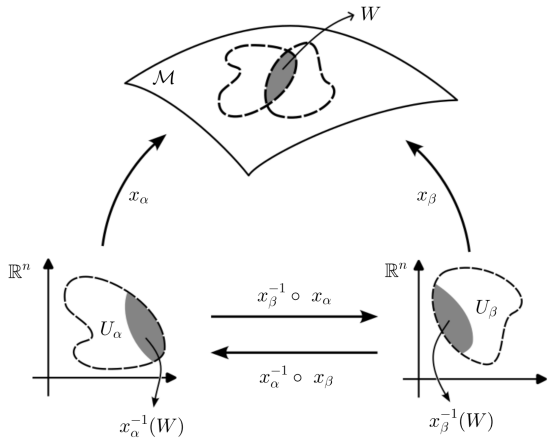


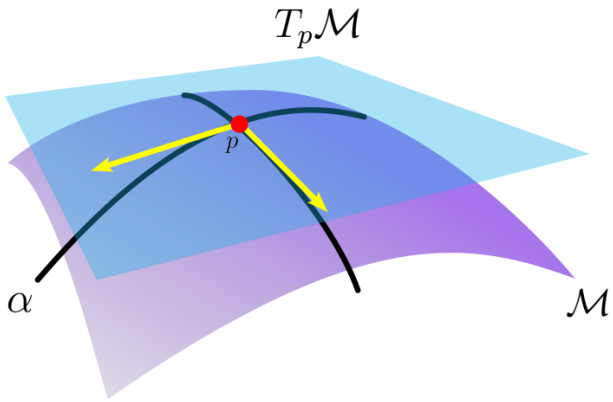
Figura 1: Variedade Diferenciável

A aplicação $\mathbf{x}_\alpha$ é chamada de *parametrização* (ou *sistema de coordenadas*) de $\mathcal{M}$ em $p \in \mathcal{M}$, e $\mathbf{x}_\alpha(U_\alpha)$ é chamada de *vizinhança coordenada* de p. Além disso, se $\mathbf{x}_\alpha(p) = (x_1(p), \ldots, x_n(p))$, cada x_i é um funcional de $U_\alpha \subset \mathbb{R}^n$ em $\mathbb{R}$ para $i = 1, \ldots, n$, e são chamadas de *coordenadas locais* de p no sistema de coordenadas $\mathbf{x}_\alpha$.

Neste trabalho, exceto quando indicado o contrário, consideraremos que todas as variedades são de Hausdorff e possuem base enumerável. Isto é, elas satisfazem os seguintes axiomas:

i. *Axioma de Hausdorff*: Dados dois pontos distintos de $\mathcal{M}$, existem vizinhanças desses pontos que não se intersectam.

Figura 2: Representação de um Espaço Tangente a uma variedade $\mathcal{M}$.



Elaborada pela autora.

ii. *Axioma da base enumerável*: $\mathcal{M}$ pode ser coberta por uma quantidade enumerável de vizinhanças coordenadas, ou seja, $\mathcal{M}$ tem base enumerável.

Definição 2. *Seja $\mathcal{M}$ uma variedade diferenciável. Uma aplicação diferenciável $\alpha : (-\epsilon, \epsilon) \subset \mathbb{R} \to \mathcal{M}$ é chamada de curva diferenciável em $\mathcal{M}$.*

Definição 3 (Espaço Tangente)**.** *Sejam $\mathcal{M}$ uma variedade diferenciável e $\alpha : (-\epsilon, \epsilon) \subset \mathbb{R} \to \mathcal{M}$ uma curva diferenciável em $\mathcal{M}$. Seja $\mathcal{D}$ o conjunto das funções diferenciáveis de $\mathcal{M}$ em p. Suponha que $\alpha(t_0) = p \in \mathcal{M}$. O vetor tangente à curva α em $t = t_0$ é a função $\alpha'(t_0) : \mathcal{D} \to \mathbb{R}$ dada por*

$$\alpha'(t_0)(f) = \frac{d(f \circ \alpha)}{dt}(t_0) \quad f \in \mathcal{D}.$$

Um vetor tangente à variedade $\mathcal{M}$ em p é qualquer vetor tangente à uma curva diferenciável α passando por p. Além disso, o conjunto de todos os vetores tangentes a $\mathcal{M}$ em p será indicado por $T_p\mathcal{M}$ e nomeado de espaço tangente de $\mathcal{M}$ em p.

A escolha de um sistema de coordenadas $\mathbf{x} : U \subset \mathbb{R}^n \to \mathcal{M}$, com $\dfrac{\partial}{\partial x_1}(q) = dx_q(e_i)$ e $p = x(q)$, determina uma base associada $\left\{ \dfrac{\partial}{\partial x_1}, ..., \dfrac{\partial}{\partial x_n} \right\}$ em $T_p\mathcal{M}$.

Definição 4. *Seja $\mathcal{M}$ uma variedade diferenciável. O fibrado tangente de $\mathcal{M}$, denotado por $T\mathcal{M}$, é a união disjunta de todos os espaços tangente em todos os pontos de $\mathcal{M}$. Elementos de $T\mathcal{M}$ podem ser representados por (p, v), onde $p \in \mathcal{M}$ e $v \in T_p\mathcal{M}$.*

Definição 5. *Um campo de vetores X em uma variedade diferenciável $\mathcal{M}$ é uma correspondência que a cada ponto $p \in \mathcal{M}$ associa um vetor $X(p) \in T_p\mathcal{M}$. Em termos de aplicações, X é uma aplicação de $\mathcal{M}$ no fibrado tangente $T\mathcal{M}$. O campo é diferenciável se a aplicação $X : \mathcal{M} \to T\mathcal{M}$ é diferenciável. Denotaremos o conjunto desses campos diferenciáveis em $\mathcal{M}$ por $\mathfrak{X}(\mathcal{M})$.*

Considerando uma parametrização $\boldsymbol{x} : U \subseteq \mathbb{R}^n \to \mathcal{M}$, é possível escrever:

$$X(p) = \sum_{i=1}^{n} a_i(p) \frac{\partial}{\partial x_i},$$

onde cada $a_i : U \subset \mathbb{R}^n \to \mathbb{R}$ *é uma função em* U *e* $\left\{\frac{\partial}{\partial x_i}\right\}$ *é a base associada a* $\boldsymbol{x}$*,* $i = 1, \ldots, n$*. Note que* X *é diferenciável se e só se as funções* a_i *são diferenciáveis para alguma (e, portanto, para qualquer) parametrização.*

Definição 6 (Colchete de Lie)**.** *Sejam* $X, Y \in \mathfrak{X}(\mathcal{M})$ *e* $f \in \mathcal{D}$*. Defina uma aplicação* $[X, Y] : \mathfrak{X}(\mathcal{M}) \times \mathfrak{X}(\mathcal{M}) \to \mathfrak{X}(\mathcal{M})$ *como*

$$[X, Y](f) = XY(f) - YX(f).$$

Essa aplicação define um campo de vetores diferenciável e é chamada de Colchete de Lie.

Proposição 1 (Propriedades do Colchete de Lie)**.** *Seja* $\mathcal{M}$ *uma variedade diferenciável. Se* X, Y *e* Z *são campos diferenciáveis em* $\mathcal{M}$*, então valem as seguintes propriedades:*

1. $[X, Y] = -[Y, X]$ *(anticomutatividade);*
2. $[aX + bY, Z] = a[X, Z] + b[Y, Z]$ *(linearidade);*
3. $[[X, Y], Z] + [[Y, Z], X] + [[Z, X], Y] = 0$ *(identidade de Jacobi);*
4. $[fX, gY] = fg[X, Y] + (fXg)Y - (gYf)X,$

onde $a, b \in \mathbb{R}$ *e* $f, g \in \mathcal{D}$*.*

Demonstração. Sejam $X, Y, Z \in \mathfrak{X}(\mathcal{M})$.

1. $[X, Y] = XY - YX = -(YX - XY) = -[Y, X]$.

2. $$\begin{aligned}[aX + bY, Z] &= (aX + bY)Z - Z(aX + bY) \\ &= aXZ + bYZ - aZX - bZY \\ &= a(XZ - ZX) + b(YZ - ZY) \\ &= a[X, Z] + b[Y, Z].\end{aligned}$$

3. Note que

   - $[[X, Y], Z] = [X, Y]Z - Z[X, Y] = XYZ - YXZ - ZXY + ZYX,$
   - $[[Y, Z], X] = [Y, Z]X - X[Y, Z] = YZX - ZYX - XYZ + XZY,$
   - $[[Z, X], Y] = [Z, X]Y - Y[Z, X] = ZXY - XZY - YZX + YXZ.$

   Somando todas as expressões, obtêm-se o resultado.

4. $$\begin{aligned}[fX, gY] &= fX(gY) - gY(fX) \\ &= f((Xg)Y + gXY) - g((Yf)X + fYX) \\ &= f(Xg)Y + fg(XY) - g(Yf)X + gfYX \\ &= fg(XY - YX) + f(Xg)Y - g(Yf)X \\ &= fg[X, Y] + f(Xg)Y - g(Yf)X.\end{aligned}$$

□

2 Geometria Riemanniana

Nesta seção, serão apresentados alguns conceitos fundamentais de geometria riemanniana, que constituem a base teórica principal deste trabalho. Entre os tópicos abordados, destacam-se a métrica riemanniana, a conexão riemanniana, os símbolos de Christoffel e a curvatura riemanniana. As principais referências para esta seção são (CARMO, 2015) e (LEE, John M, 2006).

Definição 7 (Métrica Riemanniana)**.** *Seja $\mathcal{M}$ uma variedade diferenciável. Uma métrica riemanniana em $\mathcal{M}$ é uma correspondência que associa a cada ponto p de $\mathcal{M}$ um produto interno $g_p = \langle , \rangle_p : T_p\mathcal{M} \times T_p\mathcal{M} \longrightarrow \mathbb{R}$ no espaço tangente $T_p\mathcal{M}$, que varia diferencialmente no seguinte sentido: se $\boldsymbol{x} : U \subset \mathbb{R}^n \longrightarrow \mathcal{M}$ é um sistema de coordenadas locais em torno de p, com $p = \boldsymbol{x}(x_1, x_2, ..., x_n) \in \boldsymbol{x}(U)$ e $\dfrac{\partial}{\partial x_i}\boldsymbol{x}(q) = d\boldsymbol{x}_q(0, ..., 1, ..., 0)$, então, para cada $i, j = 1, 2, ..., n$, a função $g_{ij}(x_1, ..., x_n) = \left\langle \dfrac{\partial}{\partial x_i}\boldsymbol{x}(q), \dfrac{\partial}{\partial x_j}\boldsymbol{x}(q) \right\rangle_q$ é diferenciável em U.*

As funções g_{ij} são chamadas de **componentes da métrica riemanniana** no sistema de coordenadas **x**. Além disso, uma variedade diferenciável munida de uma métrica riemanniana g, chama-se uma **variedade riemanniana**.

Proposição 2. *Uma variedade diferenciável $\mathcal{M}$ de Hausdorff e com base enumerável possui uma métrica riemanniana.*

Demonstração. Como $\mathcal{M}$ é uma variedade diferenciável com base enumerável e é de Hausdorff, é possível construir uma partição da unidade (ver Apêndice 8).

Seja $\{f_\alpha\}$ uma partição diferenciável da unidade de $\mathcal{M}$ subordinada a uma cobertura V_α de $\mathcal{M}$ por vizinhanças coordenadas.

Isto é, V_α é uma cobertura localmente finita (cada ponto de $\mathcal{M}$ possui uma vizinhança U tal que $U \cap V_\alpha \neq \emptyset$ apenas para um número finito de índices) e f_α é um conjunto de funções diferenciáveis em $\mathcal{M}$, tais que:

i) $f_\alpha \geq 0$, $f_\alpha = 0$ no complementar do fecho $\overline{V}_\alpha$.

ii) $\sum_\alpha f_\alpha(p) = 1, \forall p \in \mathcal{M}$.

Para cada α, pode-se definir uma métrica riemanniana $\langle , \rangle_\alpha$ induzida pelo sistema coordenadas em cada V_α. Tomemos então

$$\langle u, v\rangle_p = \sum_\alpha f_\alpha(p)\langle u, v\rangle_p^\alpha$$

para todo $p \in \mathcal{M}, u$ e $v \in T\mathcal{M}$

A função é de fato uma métrica riemanniana pois é uma combinação convexa da métrica riemanniana. □

2.1 Conexão Afim

Definição 8 (Conexão afim)**.** *Uma conexão afim ∇ em uma variedade diferenciável $\mathcal{M}$ é uma aplicação*

$$\nabla : \mathfrak{X}(\mathcal{M}) \times \mathfrak{X}(\mathcal{M}) \to \mathfrak{X}(\mathcal{M})$$

indicada por $(X, Y) \xrightarrow{\nabla} \nabla_X Y$ e que satisfaz as seguintes propriedades:

i) $\nabla_{fX+gY}Z = f\nabla_X Z + g\nabla_Y Z$,

ii) $\nabla_X(Y+Z) = \nabla_X Y + \nabla_X Z$,

iii) $\nabla_X(fY) = f\nabla_X Y + X(f)Y$,

onde $X, Y, Z \in \mathfrak{X}(\mathcal{M})$ *e* $f, g \in \mathcal{D}(\mathcal{M})$.

Escolha um sistema de coordenadas $(x_1, ..., x_n)$ em torno de p e escreva

$$X = \sum_i x_i \partial_i \; e \; Y = \sum_j y_j \partial_j$$

onde $\partial_i = \dfrac{\partial}{\partial x_i}$ e $\partial_j = \dfrac{\partial}{\partial x_j}$. Deste modo, temos

$$\begin{aligned}
\nabla_X Y =& \nabla_{\sum_i x_i \partial_i} Y \\
=& \sum_i x_i \nabla_{\partial_i} Y \\
=& \sum_i x_i \nabla_{\partial_i} \left(\sum_j y_j \partial_j \right) \\
=& \sum_{ij} x_i y_j \nabla_{\partial_i} \partial_j + \sum_{ij} x_i \partial_i(y_j) \partial_j.
\end{aligned}$$

Escreva, $\nabla_{\partial_i}\partial_j = \sum_k \Gamma_{ij}^k \partial_k$. Como a conexão afim é definida de maneira diferenciável, concluímos que Γ_{ij}^k devem ser funções diferenciáveis e, por um cálculo direto, obtemos

$$\begin{aligned}
\nabla_X Y =& \sum_{ij} x_i y_j \sum_k \Gamma_{ij}^k \partial_k + \sum_{ij} x_i \partial_i(y_j) \partial_j \\
=& \sum_{ij} x_i \left(y_j \sum_k \Gamma_{ij}^k \partial_k + \partial_i(y_k) \partial_k \right) \\
=& \sum_{ij} x_i y_j \left(\sum_k \Gamma_{ij}^k \partial_k \right) + \sum_{ik} x_i \partial_i(y_k) \partial_k \\
=& \sum_k \left(\sum_{ij} x_i y_i \Gamma_{ij}^k + \sum_i x_i \partial_i(y_k) \right) \partial_k \\
=& \sum_k \left(\sum_{ij} x_i y_i \Gamma_{ij}^k + X(y_k) \right) \partial_k.
\end{aligned}$$

Logo, temos que $\nabla_X Y(p)$ depende de $x_i(p)$ e $y_k(p)$ e das derivadas $X(y_k)(p)$. O símbolo $\nabla_X Y$ também pode ser interpretado como a derivada covariante do campo Y na direção de X.

Proposição 3. *Seja $\mathcal{M}$ uma variedade diferenciável com uma conexão afim ∇. Então existe uma única correspondência que associa a um campo vetorial V ao longo da curva diferenciável $c : I \to M$ um outro campo vetorial $\dfrac{DV}{dt}$ ao longo de c, denominado derivada covariante de V ao longo de c, tal que:*

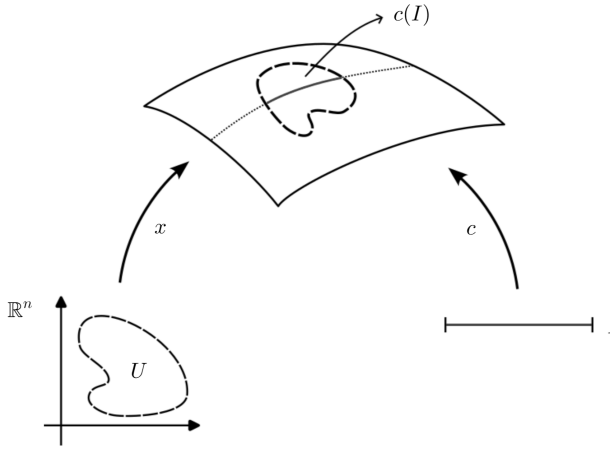
(a) $\dfrac{D}{dt}(V+W) = \dfrac{DV}{dt} + \dfrac{DW}{dt}$.

(b) $\dfrac{D}{dt}(fV) = \dfrac{df}{dt}V + f\dfrac{DV}{dt}$, *onde* V *é um campo de vetores ao longo de* c *e* f *é uma função diferenciável em* I.

(c) Se V *é induzido por um campo de vetores* $Y \in \mathfrak{X}(\mathcal{M})$*, isto é,* $V(t) = Y(c(t))$*, então* $\dfrac{DV}{dt} = \nabla_{\frac{dc}{dt}} Y$.

Demonstração. Suponha que existe uma correspondência que satisfaz a), b) e c). Seja $x : U \subset \mathbb{R}^n \to \mathcal{M}$ um sistema de coordenadas com $c(I) \cap x(U) \neq \emptyset$ e seja $(x_1(t), ..., x_n(t))$ a expressão local de $c(t), t \in I$.

Figura 3: Representação de $c(I)$.



Fonte: Elaborado pela autora.

Seja $\partial_i = \dfrac{\partial}{\partial x_i}$. Então podemos expressar o campo V localmente como $V = \sum_j v^j \partial_j, j = 1, ..., n$ onde $v^j = v^t(t)$ e $\partial_j = \partial_j(c(t))$. Usando as condições (a) e (b), tem-se

$$\begin{aligned}\frac{DV}{dt} &= \sum_j \frac{D}{dt}(v^j \partial_j) \\ &= \sum_j \left(\frac{dv^j}{dt}\partial_j + v^j \frac{D\partial_j}{dt} \right) \\ &= \sum_j \frac{dv^j}{dt}\partial_j + \sum_j v^j \frac{D\partial_j}{dt}.\end{aligned}$$

Por (c) e (i) da definição de conexão afim:

$$\frac{D\partial_j}{dt} = \nabla_{dc/dt}\partial_j = \nabla_{\left(\sum_i \frac{dx_i}{dt}\partial_i\right)}\partial_j$$

$$= \sum_i \frac{dx_i}{dt}\nabla_{\partial_i}\partial_j, \quad i, j = 1, \ldots, n.$$

Portanto,

$$\frac{DV}{dt} = \sum_j \frac{dv^j}{dt}\partial_j + \sum_i \frac{dx_i}{dt} v^j \nabla_{\partial_i}\partial_j. \tag{1}$$

A expressão (1) mostra que se existe uma correspondência satisfazendo as condições (a), (b) e (c), então ela é única.

Para a existência, basta definir $\frac{DV}{dt}$ em $\mathbf{x}(U)$ por (1). Note que, (1) possui as propriedades desejadas. Agora, se $\mathbf{y}(W)$ é uma outra vizinhança coordenada, com $\mathbf{y}(W) \cap \mathbf{x}(U) \neq \emptyset$ e definirmos $\frac{DV}{dt}$ em $\mathbf{y}(W)$ por (1), as definições "concordam"em $\mathbf{y}(W) \cap \mathbf{x}(U)$, pela unicidade de $\frac{DV}{dt}$ em $\mathbf{x}(U)$. A demonstração pode ser concluída estendendo a definição em toda variedade $\mathcal{M}$. □

Definição 9. *Seja $\mathcal{M}$ uma variedade diferenciável com uma conexão afim ∇. Um campo vetorial V ao longo de uma curva $c : I \to \mathcal{M}$ é chamado paralelo quando $\frac{DV}{dt} = 0$, para todo $t \in I$.*

Proposição 4. *Seja $\mathcal{M}$ uma variedade diferenciável com uma conexão afim ∇. Seja $c : I \to \mathcal{M}$ uma curva diferenciável em $\mathcal{M}$ e V_0 um vetor tangente a $\mathcal{M}$ em $c(t_0), t_0 \in I$ (isto é $V_0 \in T_{c(t_0)}\mathcal{M}$). Então existe um único campo de vetores paralelo V ao longo de c, tal que $V(t_0) = V_0$, ($V(t)$ é chamado o transporte paralelo de $V(t_0)$ ao longo de c).*

Demonstração. Ver (CARMO, 2015), página 58. □

Definição 10. *Seja $\mathcal{M}$ uma variedade diferenciável com uma conexão afim ∇ e uma métrica riemanniana $\langle , \rangle$. A conexão é dita compatível com a métrica $\langle , \rangle$, quando para toda curva diferenciável c e quaisquer pares de campos de vetores paralelos P e P' ao longo de c, tivermos $\langle P, P' \rangle = cte$.*

Essa definição é justificada pela proposição seguinte que mostra que se ∇ é compatível com $\langle , \rangle$, então podemos diferenciar o produto interno pela regra do produto usual.

Proposição 5. *Seja $\mathcal{M}$ uma variedade riemanniana. Uma conexão ∇ em $\mathcal{M}$ é compatível com a métrica se e só se para todo par V e W de campos de vetores ao longo da curva diferenciável $c : I \to \mathcal{M}$ tem-se*

$$\frac{d}{dt}\langle V, W \rangle = \left\langle \frac{DV}{dt}, W \right\rangle + \left\langle V, \frac{DW}{dt} \right\rangle, t \in I. \tag{2}$$

Demonstração. De fato a equação (2) implica na Definição (10), visto que se V e W são campos de vetores paralelos, então pela definição (9), temos $\frac{DV}{dt} = \mathbf{0}$ e $\frac{DW}{dt} = \mathbf{0}$. Logo, $\langle V, W \rangle = 0$.

Para mostrar a recíproca, escolha uma base ortonormal $\{P_1(t_0), ..., P_2(t_0)\}$ de $T_{c(t_0)}(\mathcal{M})$, $t_0 \in I$.

Utilizando a proposição 4, estenda paralelamente cada um dos vetores $P_i(t_0), i = 1, ..., n$ ao longo de c. Como ∇ é compatível com a métrica, $\{P_1(t), ..., P_2(t)\}$ é uma base ortonormal de $T_{c(t)}(\mathcal{M})$, para todo $t \in I$.

Portanto, podemos escrever

$$V = \sum_i v^i P_i, \quad W = \sum_i w^i P_i, \quad i = 1, \ldots, n$$

onde v^i e w^i são funções diferenciáveis em I. Daí, segue-se que

$$\frac{DV}{dt} = \sum_i \frac{dv^i}{dt} P_i, \quad \frac{DW}{dt} = \sum_i \frac{dw^i}{dt} P_i.$$

Portanto,

$$\begin{aligned}\left\langle \frac{DV}{dt}, W \right\rangle + \left\langle V, \frac{DW}{dt} \right\rangle &= \left\langle \sum_i \frac{dv^i}{dt} P_i, \sum_i w^i P_i \right\rangle + \left\langle \sum_i v^i P_i, \sum_i \frac{dw^i}{dt} P_i \right\rangle \\ &= \sum_i \left(\frac{dv^i}{dt} w^i + v^i \frac{dw^i}{dt} \right) \\ &= \frac{d}{dt} \left(\sum_i v^i w^i \right) \\ &= \frac{d}{dt} (\langle V, W \rangle).\end{aligned}$$

□

Corolário 1. *Uma conexão ∇ em $\mathcal{M}$ é dita compatível com a métrica se, e somente se,*

$$X\langle Y, Z\rangle = \langle \nabla_X Y, Z\rangle + \langle Y, \nabla_X Z\rangle,$$

para todo $X, Y, Z \in \mathfrak{X}(\mathcal{M})$.

Demonstração. Suponha que ∇ é compatível com a métrica. Seja $p \in \mathcal{M}$ e $c : I \to \mathcal{M}$, uma curva diferenciável com $c(t_0) = p, t_0 \in I$ e com $\left.\frac{dc}{dt}\right|_{t=t_0} = X(p)$. Então,

$$\begin{aligned}X(p)\langle Y, Z\rangle &= \left.\frac{d}{dt}\langle Y, Z\rangle\right|_{t=t_0} \\ &= \langle \nabla_{X(p)} Y, Z\rangle_p + \langle Y, \nabla_{X(p)} Z\rangle_p.\end{aligned}$$

Como p é arbitrário, segue o resultado. A recíproca é direta. □

Definição 11. *Uma conexão ∇ em $\mathcal{M}$ é dita simétrica quando*

$$\nabla_X Y - \nabla_Y X = [X, Y]$$

para todo $X, Y \in \mathcal{M}$.

Note que, em um sistema de coordenadas $\mathbf{x} : U \subset \mathbb{R}^n \to \mathcal{M}$, o fato de ∇ ser simétrica implica que, para todo $i, j = 1, ..., n$,

$$\nabla_{\partial_i} \partial_j - \nabla_{\partial_j} \partial_i = [\partial_i, \partial_j] = 0.$$

2.2 Conexão Riemanniana

Teorema 1 (Levi-Civita)**.** *Seja $\mathcal{M}$ uma variedade riemanniana. Então, existe uma única conexão afim ∇, denominada **conexão riemanniana**, em $\mathcal{M}$ satisfazendo as seguintes condições:*

i. ∇ *é simétrica;*

ii. ∇ *é compatível com a métrica riemanniana.*

Demonstração. Suponha que existe uma conexão afim ∇ que satisfaz as condições citadas.
Pela compatibilidade com a métrica riemanniana,

$$X\langle Y,Z\rangle = \langle \nabla_X Y, Z\rangle + \langle Y, \nabla_X Z\rangle, \tag{3}$$
$$Y\langle Z,X\rangle = \langle \nabla_Y Z, X\rangle + \langle Z, \nabla_Y X\rangle, \tag{4}$$
$$Z\langle X,Y\rangle = \langle \nabla_Z X, Y\rangle + \langle X, \nabla_Z Y\rangle. \tag{5}$$

Somando as duas primeiras expressões e subtraindo a terceira, teremos, usando a simetria de ∇,

$$\begin{aligned}
&X\langle Y,Z\rangle + Y\langle Z,X\rangle - Z\langle X,Y\rangle \\
&= \langle \nabla_X Y, Z\rangle + \langle Y, \nabla_X Z\rangle + \langle \nabla_Y Z, X\rangle + \langle Z, \nabla_Y X\rangle - \langle \nabla_Z X, Y\rangle - \langle X, \nabla_Z Y\rangle \\
&= \langle [X,Z], Y\rangle + \langle [Y,Z], X\rangle + \langle \nabla_X Y, Z\rangle + \langle \nabla_Y X, Z\rangle - \langle \nabla_Y X, Z\rangle + \langle \nabla_Y X, Z\rangle \\
&= \langle [X,Z], Y\rangle + \langle [Y,Z], X\rangle + \langle [X,Y], Z\rangle + 2\langle \nabla_X Y, Z\rangle.
\end{aligned}$$

Portanto,

$$\langle \nabla_Y X, Z\rangle = \frac{1}{2}\Big\{ X\langle Y,Z\rangle + Y\langle Z,X\rangle - Z\langle X,Y\rangle - \langle [X,Z],Y\rangle - \langle [Y,Z],X\rangle - \langle [X,Y],Z\rangle \Big\}. \tag{6}$$

Para mostrar a existência, basta tomar ∇ como sendo a expressão 6. A unicidade é determinada pela métrica. Suponha que ∇ e ∇' são duas conexões simétricas e compatíveis com a métrica g. Como o lado direito de 6 não depende da conexão, segue que

$$\langle \nabla_Y X - \nabla'_Y X, Z\rangle = 0, \forall X, Y, Z \in \mathfrak{X}(\mathcal{M}).$$

Logo, $\nabla_Y X = \nabla'_Y X$ e, portanto, $\nabla = \nabla'$. □

2.3 Símbolos de Christoffel

Em um sistema de coordenadas $\mathbf{x} : U \subset \mathbb{R}^n \to \mathcal{M}$, as funções Γ_{ij}^k definidas em U como $\nabla_{\partial_i}\partial_j = \sum_k \Gamma_{ij}^k \partial_k$ são os coeficientes da conexão ∇ em U, e são chamadas de **símbolos de Christoffel** da conexão. Utilizando os dois lemas seguintes, é possível obter uma expressão para os símbolos de Christoffel.

Lema 1. *Seja $\mathcal{M}$ uma variedade riemanniana com uma conexão riemanniana ∇. Então*

$$\langle \nabla_{\partial_i}\partial_j, \partial_k\rangle = \sum_{m=1}^{n} \Gamma_{ij}^m g_{mk}$$

onde $g_{ij} = \langle \partial_i, \partial_j\rangle$.

Demonstração. Vimos que

$$\nabla_{\partial_i}\partial_j = \sum_{m=1}^{n} \Gamma_{ij}^m \partial_m,$$

logo, segue que

$$\begin{aligned}\langle \nabla_{\partial_i}\partial_j, \partial_k\rangle =& \langle \sum_{m=1}^{n} \Gamma_{ij}^m \partial_m, \partial_k\rangle \\ =& \sum_{m=1}^{n} \Gamma_{ij}^m \langle \partial_m, \partial_k\rangle \\ =& \sum_{m=1}^{n} \Gamma_{ij}^m g_{mk}.\end{aligned}$$

□

Lema 2. *Seja $\mathcal{M}$ uma variedade riemanniana com uma conexão riemanniana ∇. Então*

$$\langle \nabla_{\partial_i}\partial_j, \partial_k\rangle = \frac{1}{2}\left(\partial_i g_{jk} + \partial_j g_{ik} - \partial_k g_{ij}\right).$$

Demonstração. Da equação 6, temos

$$\begin{aligned}\langle \nabla_{\partial_i}\partial_j, \partial_k\rangle =& \frac{1}{2}\{\partial_i\langle \partial_j, \partial_k\rangle + \partial_j\langle \partial_k, \partial_i\rangle - \partial_k\langle \partial_i, \partial_j\rangle \\ & -\langle \partial_i, [\partial_j, \partial_k]\rangle - \langle \partial_j, [\partial_i, \partial_k]\rangle - \langle \partial_k, [\partial_i, \partial_j]\rangle\} \\ =& \frac{1}{2}(\partial_j g_{ik} + \partial_i g_{jk} - \partial_k g_{ij}),\end{aligned}$$

pois, $[\partial_m, \partial_l] = 0$. □

Para conseguir explicitar a fórmula dos símbolos de Christoffel, note que

$$\sum_{l=1}^{n} \Gamma_{ij}^l g_{lm} = \frac{1}{2}\left(\partial_i g_{jm} + \partial_j g_{im} - \partial_m g_{ij}\right)$$

Daí, utilizando as componentes inversas do tensor métrico,

$$\sum_{m=1}^{n} g^{mk} \sum_{l=1}^{n} \Gamma_{ij}^l g_{lm} = \frac{1}{2}\sum_{m=1}^{n}\left(\partial_i g_{jm} + \partial_j g_{im} - \partial_m g_{ij}\right) g^{mk}.$$

Do lado esquerdo, temos

$$\sum_{l=1}^{n}\sum_{m=1}^{n} g^{km} g_{ml} \Gamma_{ij}^l = \sum_{l=1}^{n} \delta_{kl}\Gamma_{ij}^l = \Gamma_{ij}^k.$$

Portanto,

$$\Gamma_{ij}^k = \frac{1}{2}\sum_{m=1}^{n}\left(\partial_i g_{jm} + \partial_j g_{im} - \partial_m g_{ij}\right) g^{mk}.$$

2.4 Curvatura Riemanniana

Definição 12 (Curvatura)**.** *Seja $\mathcal{M}^n$ uma variedade riemanniana e ∇ a conexão riemanniana de $\mathcal{M}$. A curvatura R de $\mathcal{M}$ é uma aplicação*

$$R : \mathfrak{X}(\mathcal{M}) \times \mathfrak{X}(\mathcal{M}) \times \mathfrak{X}(\mathcal{M}) \to \mathfrak{X}(\mathcal{M})$$

definida por

$$R(X,Y)Z = \nabla_Y \nabla_X Z - \nabla_X \nabla_Y Z + \nabla_{[X,Y]} Z,$$

para quaisquer $X, Y, Z \in \mathfrak{X}(\mathcal{M})$.

Proposição 6. *A curvatura R de uma variedade riemanniana possui as seguintes propriedades:*

i) R é bilinear em $\mathfrak{X}(\mathcal{M}) \times \mathfrak{X}(\mathcal{M})$, isto é,

$$R(fX_1 + gX_2, Y_1)Z = fR(X_1, Y_1)Z + gR(X_2, Y_1)Z,$$
$$R(X_1, fY_1 + gY_2)Z = fR(X_1, Y_1)Z + gR(X_1, Y_2)Z,$$

$f, g \in \mathcal{D}(\mathcal{M})$, $X_1, X_2, Y_1, Y_2, Z \in \mathfrak{X}(\mathcal{M})$.

ii) Para todo par $X, Y \in \mathfrak{X}(\mathcal{M})$, o operador curvatura $R(X,Y) : \mathfrak{X}(\mathcal{M}) \to \mathfrak{X}(\mathcal{M})$ é linear, isto é,

$$R(X,Y)(Z+W) = R(X,Y)Z + R(X,Y)W,$$
$$R(X,Y)(fZ) = fR(X,Y)Z,$$

$f \in \mathcal{D}(M)$, $Z, W \in \mathfrak{X}(\mathcal{M})$.

Demonstração.

i) Verificaremos apenas um dos casos, pois o outro é análogo. Usando a definição de curvatura:

$$R(fX_1 + gX_2)Z = \nabla_{Y_1} \nabla_{fX_1+gX_2} Z - \nabla_{fX_1+gX_2} \nabla_{Y_1} Z + \nabla_{[fX_1+gX_2, Y_1]} Z$$

Analisando cada parcela separadamente, temos

(1) $\nabla_{Y_1} \nabla_{fX_1+gX_2} Z = f\nabla_{Y_1} \nabla_{X_1} Z + Y_1 f \nabla_{X_1} Z + g\nabla_{Y_1} \nabla_{X_2} Z + Y_1 g \nabla_{X_2} Z$

(2) $-\nabla_{fX_1+gX_2} \nabla_{Y_1} Z = -f\nabla_{X_1} \nabla_{Y_1} Z - g\nabla_{X_2} \nabla_{Y_1} Z$

(3) $\nabla_{[fX_1+gX_2,Y_1]} Z = f\nabla_{[X_1,Y_1]} Z - Y_1 f \nabla_{X_1} Z + g\nabla_{[fX_1+gX_2,Y_1]} Z - Y_1 g \nabla_{X_2} Z.$

Somando (1), (2) e (3), obtemos o resultado.

ii) Pela definição de curvatura,

$$R(X,Y)(Z+W) = \nabla_Y \nabla_X (Z+W) - \nabla_X \nabla_Y (Z+W) + \nabla_{[X,Y]}(Z+W)$$
$$= \nabla_Y \nabla_X Z - \nabla_X \nabla_Y Z + \nabla_{[X,Y]} Z + \nabla_Y \nabla_X W - \nabla_X \nabla_Y W + \nabla_{[X,Y]} W$$
$$= R(X,Y)Z + R(X,Y)W.$$

Agora, para a segunda pate, temos

$$R(X,Y)fZ = \nabla_Y\nabla_X(fZ) - \nabla_X\nabla_Y(fZ) + \nabla_{[Y,X]}(fZ). \tag{7}$$

Note que,

$$\nabla_Y\nabla_X(fZ) = \nabla_Y(f\nabla_X Z + (Xf)Z)$$
$$= f\nabla_Y\nabla_X Z + (Yf)(\nabla_X Z) + (Xf)(\nabla_Y Z) + Y(Xf)Z$$

e, do mesmo modo,

$$\nabla_X\nabla_Y(fZ) = \nabla_X(f\nabla_Y Z + (Yf)Z)$$
$$= f\nabla_X\nabla_Y Z + (Xf)(\nabla_Y Z) + (Yf)(\nabla_X Z) + (X(Yf))Z.$$

Portanto,

$$\nabla_Y\nabla_X(fZ) - \nabla_X\nabla_Y(fZ) = f(\nabla_Y\nabla_X - \nabla_X\nabla_Y)Z + ((YX - XY)f)Z$$
$$= f(\nabla_Y\nabla_X - \nabla_X\nabla_Y)Z + ([X,Y]f)Z.$$

Por fim, reescrevendo 7:

$$R(X,Y)fZ = f\nabla_Y\nabla_X Z - f\nabla_X\nabla_Y Z + ([Y,X]f)Z + f\nabla_{[X,Y]}Z + ([X,Y]f)Z$$
$$= fR(X,Y)Z.$$

□

Proposição 7 (Primeira Identidade de Bianchi)**.** *Seja $\mathcal{M}$ uma variedade riemanniana. A curvatura*

$$R : \mathfrak{X}(\mathcal{M}) \times \mathfrak{X}(\mathcal{M}) \times \mathfrak{X}(\mathcal{M}) \to \mathfrak{X}(\mathcal{M})$$

de $\mathcal{M}$ satisfaz a seguinte propriedade:

$$R(X,Y)Z + R(Y,Z)X + R(Z,X)Y = 0.$$

Demonstração. Para todo $X, Y, Z \in \mathfrak{X}(\mathcal{M})$, pela simetria da conexão riemanniana, temos que

$$\begin{aligned}
R(X,Y)Z + R(Y,Z)X + R(Z,X)Y &= \nabla_Y\nabla_X Z - \nabla_X\nabla_Y Z + \nabla_{[X,Y]}Z \\
&\quad + \nabla_Z\nabla_Y X - \nabla_Y\nabla_Z X + \nabla_{[Y,Z]}X \\
&\quad + \nabla_X\nabla_Z Y - \nabla_Z\nabla_X Y + \nabla_{[Z,X]}Y \\
&= \nabla_Y(\nabla_X Z - \nabla_Z X) - \nabla_{[X,Z]}Y + \nabla_Z(\nabla_Y X - \nabla_X Y) \\
&\quad - \nabla_{[Y,X]}Z + \nabla_X(\nabla_Z Y - \nabla_Y Z) - \nabla_{[Z,Y]}X \\
&= \nabla_Y[X,Z] - \nabla_{[X,Z]}Y + \nabla_Z[Y,X] \\
&\quad - \nabla_{[Y,X]}Z + \nabla_X[Z,Y] - \nabla_{[Z,Y]}X \\
&= [Y,[X,Z]] + [Z,[Y,X]] + [X,[Z,Y]] = 0,
\end{aligned}$$

onde foi utilizado a identidade de Jacobi. □

Podemos associar a curvatura com a métrica da seguinte foma

Proposição 8. *Seja $\mathcal{M}$ uma variedade riemanniana com uma métrica riemanniana. A curvatura*

$$R : \mathfrak{X}(\mathcal{M}) \times \mathfrak{X}(\mathcal{M}) \times \mathfrak{X}(\mathcal{M}) \to \mathfrak{X}(\mathcal{M})$$

de $\mathcal{M}$ satisfaz as seguintes propriedades:

i. $\langle R(X,Y)Z, W\rangle + \langle R(Y,Z)X, W\rangle + \langle R(Z,X)Y, W\rangle = 0$;

ii. $\langle R(X,Y)Z, W\rangle = -\langle R(Y,X)Z, W\rangle$;

iii. $\langle R(X,Y)Z, W\rangle = -\langle R(X,Y)W, Z\rangle$;

iv. $\langle R(X,Y)Z, W\rangle = \langle R(Z,W)X, Y\rangle$.

Demonstração. Sejam $X, Y, Z, W \in \mathfrak{X}(\mathcal{M})$.
i) Segue direto da proposição 7.
ii) Basta notar que $R(X,Y)Z = -R(Y,X)Z$.
iii) Considere a seguinte identidade

$$\begin{aligned}\langle R(X,Y)(Z+W), Z+W\rangle =& \langle R(X,Y)Z, Z\rangle + \langle R(X,Y)W, W\rangle \\ &+ \langle R(X,Y)Z, W\rangle + \langle R(X,Y)W, Z\rangle.\end{aligned}$$

Note que para todo $T \in \mathfrak{X}(\mathcal{M})$, tem-se $\langle R(X,Y)T, T\rangle = \langle \nabla_Y \nabla_X T - \nabla_X \nabla_Y T + \nabla_{[X,Y]}T, T\rangle$. Como

$$\langle \nabla_Y \nabla_X T, T\rangle = Y\langle \nabla_X T, T\rangle - \langle \nabla_X T, \nabla_Y T\rangle,$$
$$\langle \nabla_X \nabla_Y T, T\rangle = X\langle \nabla_Y T, T\rangle - \langle \nabla_Y T, \nabla_X T\rangle$$

e

$$\langle \nabla_{[X,Y]}T, T\rangle = \frac{1}{2}[X,Y]\langle T, T\rangle,$$

então

$$\begin{aligned}\langle R(X,Y)T, T\rangle &= Y\langle \nabla_X T, T\rangle - X\langle \nabla_Y T, T\rangle + \frac{1}{2}[X,Y]\langle T, T\rangle \\ &= \frac{1}{2}Y(X\langle T, T\rangle) - \frac{1}{2}X(Y\langle T, T\rangle) + \frac{1}{2}[X,Y]\langle T, T\rangle \\ &= -\frac{1}{2}[X,Y]\langle T, T\rangle + \frac{1}{2}[X,Y]\langle T, T\rangle = 0,\end{aligned}$$

para todo $T \in \mathfrak{X}(\mathcal{M})$. Portanto, a identidade fornece $\langle R(X,Y)Z, W\rangle = -\langle R(X,Y)W, Z\rangle$.
iv) Usando i), tem-se

$$\begin{aligned}\langle R(X,Y)Z, W\rangle + \langle R(Y,Z)X, W\rangle + \langle R(Z,X)Y, W\rangle = 0,\\ \langle R(Y,Z)W, X\rangle + \langle R(Z,W)Y, X\rangle + \langle R(W,Y)Z, X\rangle = 0,\\ \langle R(Z,W)X, Y\rangle + \langle R(W,X)Z, Y\rangle + \langle R(X,Z)W, Y\rangle = 0,\\ \langle R(W,X)Y, Z\rangle + \langle R(X,Y)W, Z\rangle + \langle R(Y,W)X, Z\rangle = 0.\end{aligned}$$

Somando as equações acima e utilizando as propriedades ii) e iii), obtemos

$$2\langle R(Z,X)Y, W\rangle + 2\langle R(W,Y)Z, X\rangle = 0.$$

Portanto, $\langle R(Z,X)Y, W\rangle = -\langle R(W,Y)Z, X\rangle = \langle R(Y,W)Z, X\rangle$. □

Dado um sistema de coordenadas $\mathbf{x} : U \subset \mathbb{R}^n \to \mathcal{M}$ em torno de um ponto $p \in \mathcal{M}$. Seja

$$R(\partial_i, \partial_j)\,\partial_k = \sum_{l=1}^{n} R^l_{ijk}\partial_l,$$

onde R^l_{ijk} são as chamadas **componentes da curvatura** R em x. Sejam $X, Y, Z \in \mathfrak{X}(\mathcal{M})$, de forma que

$$X = \sum_{i=1}^{n} u^i\partial_i, \quad Y = \sum_{j=1}^{n} v^j\partial_j, \quad Z = \sum_{k=1}^{n} w^k\partial_k,$$

segue da linearidade de R que,

$$R(X,Y)Z = \sum_{i,j,k,l=1}^{n} R^l_{ijk}u^i v^j w^k \partial_l.$$

Agora, precisamos explicitar R^l_{ijk} em termos dos símbolos de Christoffel Γ^k_{ij} da conexão riemanniana. Pela definição 12 e pelo fato de que $\left[\dfrac{\partial}{\partial x_i}, \dfrac{\partial}{\partial x_i}\right] = 0$, temos

$$\begin{aligned}
R(\partial_i, \partial_j)\,\partial_k &= \nabla_{\partial_j}\nabla_{\partial_i}\partial_k - \nabla_{\partial_i}\nabla_{\partial_j}\partial_k \\
&= \nabla_{\partial_j}\left(\sum_l \Gamma^l_{ik}\partial_l\right) - \nabla_{\partial_i}\left(\sum_l \Gamma^l_{jk}\partial_l\right) \\
&= \sum_l \partial_j\Gamma^l_{ik}\partial_l + \sum_l \Gamma^l_{ik}\nabla_{\partial_j}\partial_l - \sum_l \partial_i\Gamma^l_{jk}\partial_j - \sum_l \Gamma^l_{jk}\nabla_{\partial_i}\partial_l \\
&= \sum_l \partial_j\Gamma^l_{ik}\partial_l + \sum_l \Gamma^l_{ik}\sum_m \Gamma^m_{jl}\partial_m - \sum_l \partial_i\Gamma^l_{jk}\partial_l - \sum_l \Gamma^l_{jk}\sum_m \Gamma^m_{il}\partial_m,
\end{aligned}$$

por consequência,

$$R^s_{ijk} = \sum_l \Gamma^l_{ik}\Gamma^s_{jl} - \sum_l \Gamma^l_{ik}\Gamma^s_{jl} + \partial_j\Gamma^s_{ik} - \partial_i\Gamma^s_{jk}.$$

Fazendo,

$$\langle R(\partial_i, \partial_j)\,\partial_k, \partial_s\rangle = \sum_l R^l_{ijk}g_{ls} = R_{ijks},$$

podemos escrever as identidades da proposição 8 como:

$$\begin{aligned}
R_{ijks} + R_{jkis} + R_{kijs} &= 0 \\
R_{ijks} &= -R_{jiks} \\
R_{ijks} &= -R_{ijsk} \\
R_{ijks} &= R_{ksij}.
\end{aligned}$$

Definição 13. *Seja $R : \mathfrak{X}(\mathcal{M}) \times \mathfrak{X}(\mathcal{M}) \times \mathfrak{X}(\mathcal{M}) \to \mathfrak{X}(\mathcal{M})$. Definimos:*

(a) A curvatura de Ricci ou Tensor de Ricci, denotado por Ric é definido como o traço da aplicação R. As componentes de Ric, denotadas como Ric_{ij}, são dadas por

$$R_{ij} = \sum_{k,l} g^{kl}R_{kijl}.$$

(b) A curvatura escalar é a função R_g definida como o traço do tensor de Ricci,

$$R_g = \sum_{i,j} Ric_{ij}.$$

Teorema 2 (Teorema Egregium de Gauss)**.** *Seja $\mathcal{M} \subset \mathbb{R}^3$ uma subvariedade bidimensional e g a métrica induzida em $\mathcal{M}$. Para todo $p \in \mathcal{M}$ e qualquer base X, Y de $T_p\mathcal{M}$, a curvatura Gaussiana de M em p é dada por*

$$K = \frac{\langle R(X,Y)Y, X\rangle}{|X|^2|Y|^2 - \langle X, Y\rangle^2}.$$

Portanto, a curvatura Gaussiana é invariante por isometrias.

Demonstração. Ver (LEE, John M, 2006), página 143. □

Lema 3. *A curvatura Gaussiana de uma variedade riemanniana bidimensional está relacionada com o tensor curvatura, o tensor de Ricci e a curvatura escalar pelas fórmulas*

$$\begin{aligned} R(X,Y,Z,W) &= K(\langle X, W\rangle\langle Y, Z\rangle - \langle X, Z\rangle\langle Y, W\rangle) \\ Ric(X,Y) &= K\langle X, Y\rangle \\ R_g &= 2K. \end{aligned}$$

Demonstração. Ver (LEE, John M, 2006), página 144. □

Sejam $(\widetilde{\mathcal{M}}, \widetilde{g})$ uma variedade riemanniana de dimensão n, $\mathcal{M}$ uma variedade de dimensão m, e $f : \mathcal{M} \to \widetilde{\mathcal{M}}$ uma imersão (ver Apêndice 8 Definição 19). Se em $\mathcal{M}$ consideramos a métrica riemanniana induzida $g = f^*\widetilde{g}$ (Ver Apêndice 8, Definição 20), dizemos que f é uma imersão isométrica (ou um mergulho isométrico se f for um mergulho de acordo com Apêndice 8 Definição 19). Se além disso f é injetiva, tal que $\mathcal{M}$ é uma subvariedade (imersa ou mergulhada) de $\widetilde{\mathcal{M}}$, dizemos que $\mathcal{M}$ é uma subvariedade riemanniana de $\widetilde{\mathcal{M}}$. Em qualquer caso, $\widetilde{\mathcal{M}}$ é dita ser o espaço ambiente.

Como toda imersão é localmente um mergulho (ver Teorema 5), então vamos supor que todas as subvariedades são mergulhadas, embora todos os resultados locais se apliquem à imersões isométricas. Os objetos relativos $(\widetilde{\mathcal{M}}, \widetilde{g})$ serão denotados com o símbolo $\sim$. Além disso, frequentemente iremos identificar $\mathcal{M}$ com sua imagem em $\widetilde{\mathcal{M}}$. Qualquer campo vetorial suave $X \in \mathfrak{X}(\widetilde{\mathcal{M}})$ claramente restringe-se a um campo vetorial suave em $\mathfrak{X}(\mathcal{M})$. Reciprocamente, qualquer campo vetorial suave em $\mathfrak{X}(\mathcal{M})$ pode ser estendido a um campo vetorial suave em $\mathfrak{X}(\widetilde{\mathcal{M}})$.

Em cada ponto $p \in \widetilde{\mathcal{M}}$, o produto interno em $T_p\widetilde{\mathcal{M}}$ induz uma decomposição ortogonal em soma direta

$$T_p\widetilde{\mathcal{M}} = T_p\mathcal{M} \oplus (T_p\mathcal{M})^{\perp}.$$

O espaço $(T_p\mathcal{M})^{\perp}$ é o espaço ortogonal ao espaço $T_p\mathcal{M}$ com respeito ao produto interno $\widetilde{g}$, é usual denotá-lo como $N_p\mathcal{M}$, e é chamado de espaço normal a $\mathcal{M}$ em p. O conjunto

$$N\mathcal{M} := \bigsqcup_{p\in\mathcal{M}} N_p\mathcal{M}$$

é chamado de fibrado normal de $\mathcal{M}$. Assim, dado um campo vetorial $X \in \mathcal{X}(\widetilde{\mathcal{M}})$, em cada ponto $p \in \widetilde{\mathcal{M}}$ podemos decompô-lo como

$$X_p = X_p^\top + X_p^\perp,$$

onde $X_p^\top \in T_p\mathcal{M}$ e $X_p^\perp \in N_p\mathcal{M}$.

Na Geometria Diferencial, a métrica riemanniana é expressa pela primeira forma fundamental. Com relação à subvariedade riemanniana, o tensor de curvatura riemanniana é capturado pela segunda forma fundamental $II(X, Y)$, que é uma forma bilinear e simétrica definida em campos vetoriais tangentes X e Y.

2.5 Segunda Forma Fundamental

Sejam $X, Y \in X(\mathcal{M})$ campos de vetores suaves em $\mathcal{M}$, e $\widetilde{X}, \widetilde{Y} \in X(\widetilde{\mathcal{M}})$ extensões suaves dos campos $X \in Y$, respectivamente, à variedade $\widetilde{\mathcal{M}}$. Aplique a derivada covariante do ambiente $\widetilde{Y}$ e então decomponha em pontos de $\mathcal{M}$ para obter

$$\widetilde{\nabla}_{\widetilde{X}}\widetilde{Y} = (\widetilde{\nabla}_{\widetilde{X}}\widetilde{Y})^\top + (\widetilde{\nabla}_{\widetilde{X}}\widetilde{Y})^\perp. \tag{8}$$

Definição 14 (Segunda Forma Fundamental)**.** *A segunda forma fundamental de $\mathcal{M}$ é a aplicação $II : \mathfrak{X}(\mathcal{M}) \times \mathfrak{X}(\mathcal{M}) \to N\mathcal{M}$ dada por*

$$II(X, Y) := (\widetilde{\nabla}_X Y)^\perp.$$

Figura 4: Segunda Forma Fundamental

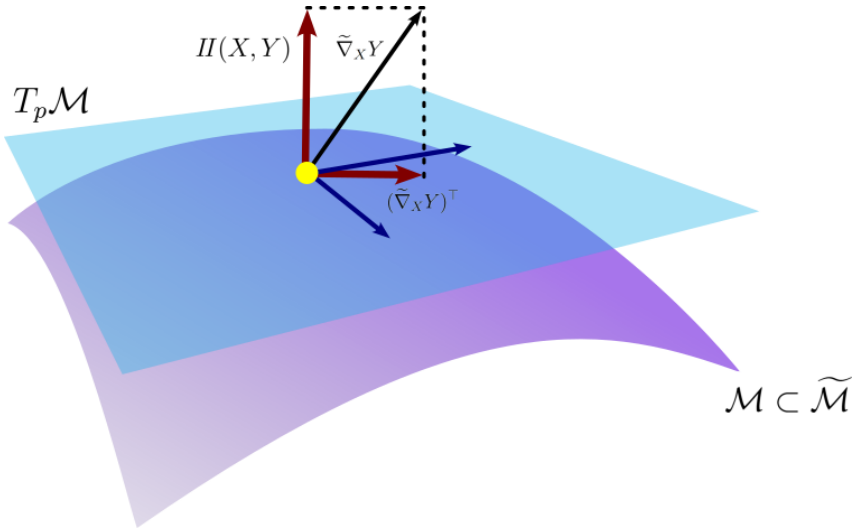


Figura 5: Elaborado pela autora.

Lema 4. *A segunda forma fundamental é*

a) independente das extensões de X e Y;

b) bilinear sobre $C^\infty(\mathcal{M})$[1]*;*

[1] $C^\infty(\mathcal{M})$ denota o conjunto das funções suaves (ou infinitamente diferenciáveis) definidas na variedade $\mathcal{M}$.

c) simétrica em X e Y.

Demonstração. Inicialmente, vamos mostrar que a simetria de II segue da simetria da conexão $\widetilde{\nabla}$. Sejam X e Y extensões arbitrárias de M, então

$$II(X,Y) - II(Y,X) = (\widetilde{\nabla}_{\widetilde{X}}\widetilde{Y} - \widetilde{\nabla}_{\widetilde{Y}}\widetilde{X})^{\perp} = [\widetilde{X},\widetilde{Y}]^{\perp}.$$

Como $X, Y \in \mathfrak{X}(\mathcal{M})$, em todos os pontos de $\mathcal{M}$ eles são tangentes a $\mathcal{M}$, logo, o colchete de Lie também é. Portanto, $[\widetilde{X},\widetilde{Y}]^{\perp} = 0$. Isto implica que $(\widetilde{\nabla}_{\widetilde{X}}\widetilde{Y})^{\perp} = (\widetilde{\nabla}_{\widetilde{Y}}\widetilde{X})^{\perp}$. Ou seja, II é simétrica.

Como $\widetilde{\nabla}_{\widetilde{X}}\widetilde{Y}|_P$ depende somente de X_P, para todo $P \in \mathcal{M}$, segue que $(\widetilde{\nabla}_{\widetilde{X}}\widetilde{Y})^{\perp}$ é independente da extensão escolhida para X, e que ele também é $C^{\infty}(\mathcal{M})$-linear em X. Da simetria, segue o mesmo para Y.

□

Do lema, segue que não há ambiguidade em continuarmos denotando as extensões ainda como X, Y.

Teorema 3 (Fórmula de Gauss)**.** *Se $X, Y \in \mathfrak{X}(\mathcal{M})$ são estendidas arbitrariamente a campos vetoriais de $\widetilde{\mathcal{M}}$, ao longo de $\mathcal{M}$ tem-se que*

$$\widetilde{\nabla}_X Y = \nabla_X Y + II(X,Y).$$

Demonstração. Da decomposição (8) e da definição da segunda forma fundamental, é suficiente mostrar que $(\widetilde{\nabla}_X Y)^T = \nabla_X Y$ em todos os pontos de $\mathcal{M}$. Defina $\nabla^T \colon \mathcal{X}(\mathcal{M}) \times \mathcal{X}(\mathcal{M}) \to \mathcal{X}(\mathcal{M})$ como

$$\nabla^{\top}_X Y = (\widetilde{\nabla}_X Y)^{\top},$$

onde X, Y são extensões arbitrárias a $\widetilde{\mathcal{M}}$. Não é difícil ver que $\nabla^{\top}$ define uma conexão em $\mathcal{M}$. Portanto, basta mostrar que ela é simétrica e compatível com a métrica g. Da unicidade da conexão riemanniana em $\mathcal{M}$, isto mostra que $\nabla^{\top} = \nabla$.
Para ver que $\nabla^{\top}$ é simétrica, note que

$$\nabla^{\top}_X Y - \nabla^{\top}_Y X = (\widetilde{\nabla}_X Y)^{\top} - (\widetilde{\nabla}_Y X)^{\top} = (\widetilde{\nabla}_X Y - \widetilde{\nabla}_Y X)^{\top} = [X,Y]^{\top} = [X,Y].$$

Agora, queremos mostrar a compatibilidade de $\nabla^{\top}$ com g. Para isso, consideremos $X, Y, Z \in \mathcal{X}(\mathcal{M})$ e suas respectivas extensões arbitrárias sobre $\widetilde{\mathcal{M}}$. Usando a compatibilidade de $\widetilde{\nabla}$ com $\widetilde{g}$, e calculando em pontos de $\mathcal{M}$, tem-se

$$X\langle Y,Z\rangle = \langle \widetilde{\nabla}_X Y, Z\rangle + \langle Y, \widetilde{\nabla}_X Z\rangle = \langle (\widetilde{\nabla}_X Y)^{\top}, Z\rangle + \langle Y, (\widetilde{\nabla}_X Z)^{\top}\rangle = \langle \nabla^{\top}_X Y, Z\rangle + \langle Y, \nabla^{\top}_X Z\rangle.$$

Portanto, $\nabla^{\top}$ é compatível com g, donde $\nabla^{\top} = \nabla$. □

Lema 5. *(Equação de Weingarten) Sejam $X, Y \in \mathfrak{X}(\mathcal{M})$ e $N \in N\mathcal{M}$. Quando $X, Y \in N$ são estendidas a $\widetilde{\mathcal{M}}$, então em $\mathcal{M}$ temos*

$$\langle \widetilde{\nabla}_X N, Y\rangle = -\langle N, II(X,Y)\rangle.$$

Demonstração. Como $\langle N, Y\rangle \equiv 0$ em $\mathcal{M}$ e X é tangente a $\mathcal{M}$, então ao longo de $\mathcal{M}$ temos

$$0 = X\langle N,X\rangle = \langle \widetilde{\nabla}_X N, Y\rangle + \langle N, \widetilde{\nabla}_X Y\rangle = \langle \widetilde{\nabla}_X N, Y\rangle + \langle N, \nabla_X Y + II(X,Y)\rangle$$

$$= \langle \widetilde{\nabla}_X N, Y\rangle + \langle N, II(X,Y)\rangle.$$

□

O Teorema 3 nos fornece uma relação entre as conexões em $\mathcal{M}$ e $\widetilde{\mathcal{M}}$. Do mesmo modo, podemos obter uma relação entre os tensores de Riemann de $\mathcal{M}$ e $\widetilde{\mathcal{M}}$. Vejamos o teorema a seguir.

Teorema 4 (Equação de Gauss)**.** *Para quaisquer $X, Y, Z, W \in T_p\mathcal{M}$, tem-se*

$$\widetilde{Rm}(X,Y,Z,W) = Rm(X,Y,Z,W) - \langle II(X,W), II(Y,Z)\rangle + \langle II(X,Z), II(Y,W)\rangle.$$

Demonstração. Sejam X, Y, Z e W extensões a campos vetoriais em $\mathcal{M}$, e consequentemente a campos vetoriais em $\widetilde{\mathcal{M}}$ que são tangentes a $\mathcal{M}$. Ao longo de $\mathcal{M}$, da fórmula de Gauss obtemos

$$\begin{aligned}\widetilde{Rm}(X,Y,Z,W) &= \langle \widetilde{\nabla}_X\widetilde{\nabla}_Y Z - \widetilde{\nabla}_Y\widetilde{\nabla}_X Z - \widetilde{\nabla}_{[X,Y]}Z, W\rangle \\ &= \langle \widetilde{\nabla}_X(\nabla_Y Z + II(Y,Z)) - \widetilde{\nabla}_Y(\nabla_X Z + II(X,Z)) \\ &\quad - \nabla_{[X,Y]}Z - II([X,Y],Z), W\rangle.\end{aligned}$$

Como a segunda forma fundamental assume valores ortogonais a $\mathcal{M}$ e W é tangente a $\mathcal{M}$, o último termo II é nulo. Aplique a equação de Weingarten aos outros dois termos que envolvem II e obtenha

$$\begin{aligned}\widetilde{Rm}(X,Y,Z,W) &= \langle \widetilde{\nabla}_X\nabla_Y Z, W\rangle - \langle II(Y,Z), II(X,W)\rangle \\ &\quad - \langle \widetilde{\nabla}_Y\nabla_X Z, W\rangle + \langle II(X,Z), II(Y,W)\rangle \\ &\quad - \langle \nabla_{[X,Y]}Z, W\rangle.\end{aligned}$$

Decomponha cada termo envolvendo $\widetilde{\nabla}$ em suas componentes tangentes e normais. A componente normal desaparece, já que W é tangente. Pela equação de Gauss, obtemos

$$\begin{aligned}\widetilde{Rm}(X,Y,Z,W) &= \langle \nabla_X\nabla_Y Z, W\rangle - \langle \nabla_Y\nabla_X Z, W\rangle \\ &\quad - \langle \nabla_{[X,Y]}Z, W\rangle - \langle II(Y,Z), II(X,W)\rangle \\ &\quad + \langle II(X,Z), II(Y,W)\rangle \\ &= \langle R(X,Y)Z, W\rangle - \langle II(X,W), II(Y,Z)\rangle \\ &\quad + \langle II(X,Z), II(Y,W)\rangle.\end{aligned}$$

□

3 Formas alternativas de estimar o tensor de Ricci

Na seção anterior, tratamos de definir o tensor de curvatura de Ricci a partir de informações bem conhecidas da variedade, em particular assumindo que a família de parametrizações que a descreve é conhecida. No entanto, quando lidamos com dados discretos, em geral dispomos apenas das posições amostradas, sem acesso à parametrização subjacente. Nesse contexto, destacam-se na literatura dois métodos que podem auxiliar na estimativa da curvatura de Ricci: o método de Ollivier–Ricci (NI et al., 2019) e o método de Forman–Ricci (FORMAN, 2003).

3.1 Ollivier-Ricci

O método de Ollivier–Ricci (NI et al., 2019) constitui uma formulação discreta da curvatura de Ricci baseada na teoria do transporte ótimo em espaços métricos. Seja $\{p_i\}_{i=1}^{N}$ um conjunto finito de pontos pertencentes a uma superfície regular desconhecida, cuja métrica intrínseca não é explicitamente parametrizada. Inicialmente, constrói-se um grafo $G = (V, E)$ em que cada vértice $v_i \in V$ corresponde a um ponto p_i, e as arestas $e_{ij} \in E$ conectam pares de pontos vizinhos, determinados por um critério de k-vizinhos mais próximos ou por um raio geodésico aproximado. A cada aresta associa-se um peso d_{ij} que representa a distância métrica estimada entre p_i e p_j, usualmente aproximada pela distância euclidiana ou pelo comprimento mínimo no grafo.

Para cada vértice v_i, define-se uma medida de probabilidade local μ_{v_i}, tipicamente uniforme sobre os vértices adjacentes, podendo incluir o próprio v_i com um peso $\alpha \in [0, 1]$ para regularização. A curvatura de Ollivier-Ricci ao longo de uma aresta (v_i, v_j) é então obtida como

$$\kappa(v_i, v_j) = 1 - \frac{W_1(\mu_{v_i}, \mu_{v_j})}{d_{ij}},$$

onde W_1 denota a *distância de Wasserstein de ordem 1* entre as distribuições μ_{v_i} e μ_{v_j}, definida no caso discreto da seguinte forma: dado um espaço métrico finito (X, d) com pontos $\{x_1, \ldots, x_m\}$ e duas distribuições de probabilidade discretas

$$\mu = \sum_{i=1}^{m} \mu_i \, \delta_{x_i}, \quad \nu = \sum_{j=1}^{m} \nu_j \, \delta_{x_j},$$

a distância W_1 é

$$W_1(\mu, \nu) = \min_{\pi \in \mathbb{R}_+^{m \times m}} \sum_{i=1}^{m} \sum_{j=1}^{m} d(x_i, x_j) \, \pi_{ij},$$

sujeita a

$$\sum_{j=1}^{m} \pi_{ij} = \mu_i, \quad \sum_{i=1}^{m} \pi_{ij} = \nu_j.$$

Aqui, π_{ij} representa a quantidade de massa transportada de x_i para x_j, e o problema consiste em minimizar o custo total de transporte. Valores positivos de κ indicam concentração das vizinhanças (curvatura positiva), enquanto valores negativos indicam dispersão relativa (curvatura negativa). A curvatura associada a um vértice pode ser obtida pela média das curvaturas das arestas incidentes, fornecendo assim uma estimativa escalar intrínseca para cada ponto amostrado.

3.2 Forman-Ricci

O método de Forman-Ricci (FORMAN, 2003), usa a abordagem combinatória para estimar a curvatura de Ricci em grafos. Dada uma amostra finita de pontos de uma superfície, constrói-se inicialmente um grafo de vizinhança, em que cada vértice representa um ponto da amostra e as arestas conectam pares de pontos considerados vizinhos, tipicamente definidos por critérios de k-vizinhos mais próximos ou raio fixo. Diferentemente do método de Ollivier-Ricci, a abordagem de Forman-Ricci não depende do cálculo do transporte ótimo, sendo baseada diretamente nas propriedades locais de conectividade do grafo. Cada aresta recebe um peso, que pode ser definido a partir de distâncias entre os pontos ou uniformemente, e a curvatura de

Forman ao longo da aresta é computada a partir dos graus dos vértices incidentes e das arestas adjacentes.

Especificamente, para uma aresta $e = (u, v)$ conectando os vértices u e v, a curvatura de Forman é dada por

$$\mathrm{F}(e) = w_e \left(\frac{w_u}{w_e} + \frac{w_v}{w_e} - \sum_{e_u \sim e,\, e_v \sim e} \left[\frac{w_u}{\sqrt{w_e w_{e_u}}} + \frac{w_v}{\sqrt{w_e w_{e_v}}} \right] \right),$$

onde w_u e w_v são os pesos dos vértices, w_e é o peso da aresta, e e_u e e_v são arestas incidentes a u e v, respectivamente, diferentes de e. No caso não ponderado, em que todos os pesos são iguais a 1, a fórmula se reduz a $\mathrm{F}(e) = 4 - (\deg(u) + \deg(v))$. A curvatura associada a um vértice pode ser obtida como a média das curvaturas das arestas incidentes, fornecendo assim uma estimativa escalar discreta da curvatura de Ricci local para cada ponto da amostra.

4 Aprendizado de Máquina

Nesta seção, serão introduzidos alguns conceitos fundamentais de aprendizado de máquina, que posteriormente, desempenharão um papel crucial no desenvolvimento deste trabalho. Quando combinados com a geometria riemanniana, esses conceitos se tornam ferramentas poderosas para a formulação de soluções eficientes aplicáveis a diversos tipos de problemas. Serão abordados em particular, tópicos como: Método dos K-Vizinhos Mais Próximos (KNN), Análise de Componentes Principais (PCA), Método dos Mínimos Quadrados e Redes Neurais.

Cada um desses métodos desepenhará uma tarefa essencial na nossa proposta de abordagem, sendo os três primeiros parte do método original proposto em (LI; LU, 2019), enquanto as redes neurais serão introduzidas para adaptar essa abordagem.

O KNN será utilizado para identificar a vizinhança local de cada ponto da nuvem, construindo uma estrutura de vizinhança para essa amostra de pontos da variedade, o PCA será empregado para determinar uma aproximação do espaço tangente em cada vizinhança e com o objetivo de fundamentar a proposta de (LI; LU, 2019), vamos aplicar o Método dos Mínimos Quadrados para obter uma aproximação suave da variedade dentro de cada vizinhança, dada por $f(x_1, x_2, ..., x_d) = (x_1, ..., x_d, f_{d+1}(x_1, ..., x_d), ..., f_D(x_1, ..., x_d))$ onde cada f_α com $\alpha = d + 1, ..., D$ é obtido através de um polinômio. Para exemplificar o caso de uma superfície em (ou seja, uma variedade bidimensional) em $\mathbb{R}^3$, esse método permite ajustar uma função polinomial local $f_3(u, v)$ que aproxima a parametrização da superfície por $f(u, v) = (u, v, f_3(u, v))$. A proposta deste trabalho é substituir essa etapa de aproximação por redes neurais, explorando a capacidade de capturar padrões complexos e, potencialmente, melhorar a precisão da estimativa da curvatura. Nos capítulos seguintes, serão apresentados detalhes sobre o funcionamento de cada tópico, e no capítulo de metodologia veremos mais detalhes de como eles serão utilizados no algoritmo.

4.1 Método dos K-vizinhos Mais Próximos (KNN)

O Método dos K-vizinhos Mais Próximos, conhecido em inglês como *K-Nearest Neighbours* (KNN), é um algoritmo amplamente utilizado na área de aprendizado de máquina, principalmente devido à sua simplicidade e eficácia. Sua popularidade se deve tanto à facilidade de compreensão quanto à sua utilidade em diversas aplicações.

A ideia central do algoritmo pode ser comparada à expressão "pássaros da mesma espécie voam juntos"— adaptada para o contexto: pontos semelhantes tendem a estar próximos uns

dos outros. O KNN é empregado em problemas de regressão[2] e classificação[3] e, como o nome sugere, baseia-se na identificação dos k "vizinhos mais próximos"de um novo ponto inserido, utilizando uma métrica de distância para determinar essa proximidade. Formalmente, dado um conjunto $X = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_m\} \subset \mathbb{R}^n$ de m pontos, $k \in \mathbb{Z}^+$ e $\mathbf{x}_i \in X$, defina uma função de distância $d_{ij} = d(\mathbf{x}_i, \mathbf{x}_j)$. Agora, seja D a matriz de distâncias e l_i na linha i, a lista das distâncias de $\mathbf{x}_i$ a $\mathbf{x}_j$ $\forall_j = 1, ..., m$.

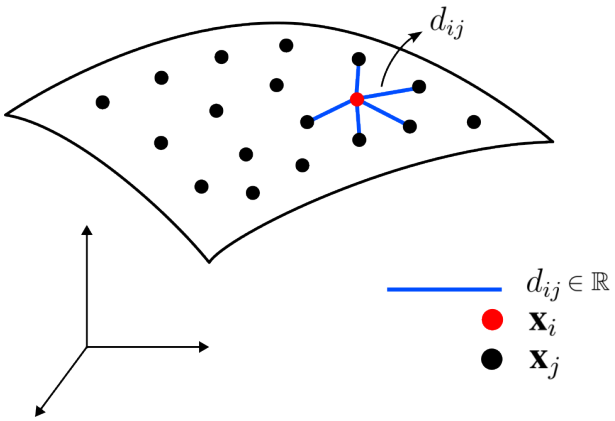
$$D = \begin{bmatrix} d_{11} & d_{12} & ... & d_{1m} \\ \vdots & \vdots & \vdots & \vdots \\ d_{i1} & d_{i2} & ... & d_{im} \\ \vdots & \vdots & \vdots & \vdots \\ d_{m1} & d_{m2} & ... & d_{mm} \end{bmatrix}.$$

Ordenando l_i de forma crescente obtemos uma nova lista $l_i' = \{d_{in_1}, d_{in_2}, ..., d_{in_m}\}$, onde $d_{in_1} \leq d_{in_2} \leq ... \leq d_{in_m}$ e $n_1, n_2, ..., n_m$ é outra ordenação para os índices. Selecionando os k primeiros elementos de l_i' obtemos os k vizinhos mais próximos a $\mathbf{x}_i$.

Existem diversas funções de distância que podem ser utilizadas nesse método. Entre as mais comuns estão: Euclidiana, Hamming, Manhattan, Mahalanobis e Minkowski. Neste trabalho, será utilizada a distância euclidiana, que, dado dois pontos $\mathbf{x} = (x_1, ..., x_n)$, $\mathbf{y} = (y_1, ..., y_n) \in \mathbb{R}^n$, é definida por:

$$d(\mathbf{x}, \mathbf{y}) = \sqrt{(x_1 - y_1)^2 + ... + (x_n - y_n)^2}.$$

Figura 6: Os k-vizinhos mais próximos de $\mathbf{x}_i$, com $k = 5$.



Fonte: Elaborado pela autora.

4.2 Análise de Componentes Principais

O PCA (*Principal Components Analysis*), em português Análise de Componentes Principais, é uma técnica fundamental de redução de dimensionalidade que preserva a variabilidade dos dados, introduzida por Hotelling (1933). Supondo que os dados $\mathcal{D}$ estão contidos em um

[2]Técnica de aprendizado supervisionado em que o objetivo é prever valores numéricos a partir de variáveis de entrada.

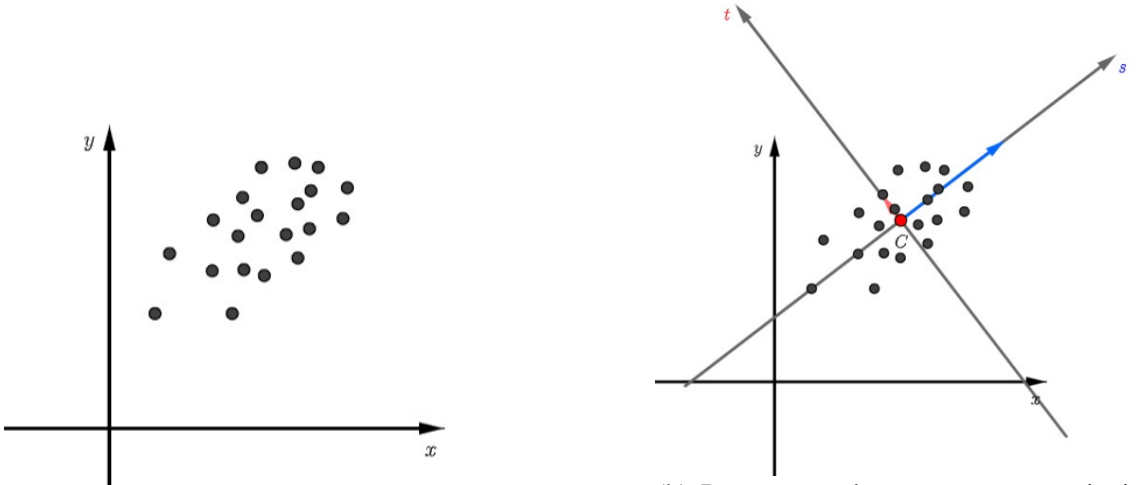
[3]Técnica de aprendizado supervisionado em que o objetivo é atribuir rótulos ou categorias a cada entrada, de acordo com suas características.

espaço Euclidiano de dimensão D, o método tem como objetivo procurar por k vetores ortonormais em $\mathbb{R}^D$ que melhor representem os dados, sendo $k \leq D$.

A Análise dos Componentes Principais (PCA) é, de longe, o algoritmo de redução de dimensionalidade mais popular. O PCA identifica o eixo que representa a maior quantidade de variância no conjunto de treinamento. A primeira componente principal é a direção que retém a maior variância dos dados. A segunda componente principal é ortogonal à primeira e representa a segunda maior variância, e assim sucessivamente. O vetor da unidade que define o i-ésimo eixo é chamado de i-ésimo componente principal. (GERON, 2021).

O algoritmo para a obtenção das componentes principais inicia-se com a centralização dos dados. Seja $C = \frac{1}{N}\sum_{i=1}^{N} p_i$ o centróide do conjunto dos dados. Caso tenhamos $C \neq \vec{0}$, realiza-se uma translação do sistema de coordenadas, tal que $u_i = p_i - C$, obtendo uma nova origem que coincida com o contróide, como ilustrado na Figura 7.

Figura 7: Dados originais em $\mathbb{R}^2$ e componentes principais no sistema de coordenadas st



(a) Dados originais

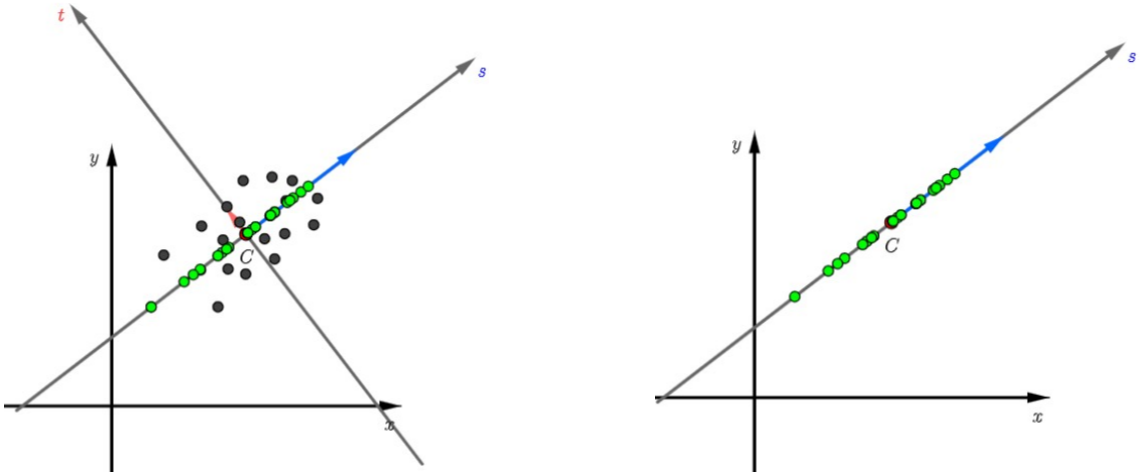
(b) Ponto central e componentes principais

Fonte: Elaborado pela autora.

Para determinar as direções dos dados que apresentam maior variância, calcula-se a matriz de covariância onde os componentes são dados por $R_i = \sum_{i=1}^{N} u_i {u_i}^T$. Os autovetores de R indicam as direções das componentes principais, enquanto os autovalores correspondentes revelam quão importante é essa componente, no sentido de que, quanto maior for o autovalor, maior será a relevância do autovetor associado na dispersão dos dados. Do mesmo modo, um autovalor pequeno indica uma direção menos relevante, ou seja, os dados espalham-se menos nesta direção. Esses autovalores são organizados em ordem decrescente e uma matriz de autovalores ortonormais também é montada obedecendo a mesma ordem.

Quando o objetivo é reduzir a dimensionalidade, selecionamos apenas as k primeiras componentes principais (ou seja, as mais significativas) e as agrupamos em uma matriz A. A transformação final dos dados é realizada projetando cada ponto u_i no novo espaço através da operação $v_i = A^T u_i$, como ilustra a Figura 8. Essa projeção mantém o máximo possível da variação original dos dados.

Figura 8: Projeção dos dados no espaço mais significativo



Fonte: Elaborado pela autora.

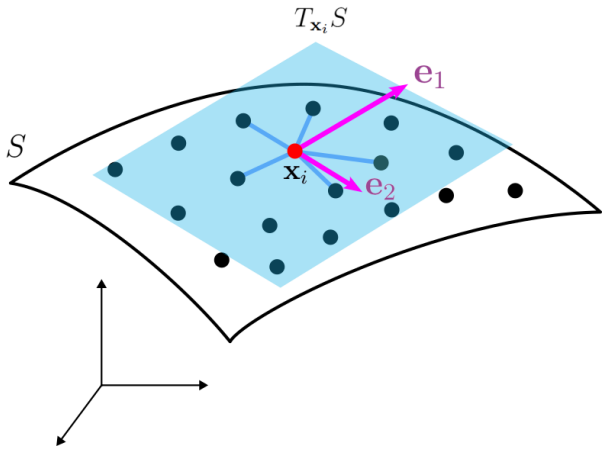
O método do PCA é baseado na técnica de Decomposição em Valores Singulares (SVD, do inglês *Singular Value Decomposition*). Em essência, seja uma matriz de dados X, de ordem $m \times n$, cujo centro é a origem do sistema de coordenadas, então X pode ser expressa como

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

em que, U e V são matrizes ortogonais e Σ é uma matriz $m \times n$ diagonal cujas entradas diagonais são os valores singulares de X, que são as raízes quadradas dos autovalores de $X^T X$, e denotamos por σ_i (ANTON; RORRES, 2012). Além disso, V contém os autovetores da matriz de covariância de X, dessa forma, os vetores coluna de V são justamente as componentes principais procuradas pelo PCA.

O PCA também pode ser utilizado para fazer uma aproximação do espaço tangente de uma nuvem de pontos desde que seja escolhida uma vizinhança adequada. Dada uma vizinhança local $U_i,\ i = 1, 2, ..., N$, estimada pelo método dos K-vizinhos mais próximos, para cada ponto $\mathbf{x}_i \in \mathbb{R}^D$. Em cada vizinhança U_i de $\mathbf{x}_i$, é estimado uma matriz de covariância $R_i = \sum_{\mathbf{x}_k \in U_i} (\mathbf{x}_k - \overline{\mathbf{x}_i})(\mathbf{x}_k - \overline{\mathbf{x}_i})^T$, onde $\overline{\mathbf{x}_i}$ é o vetor médio dos KNN. Daí, assumindo que a dimensão intrínseca é $d << D$, temos que, o espaço tangente $T_{\mathbf{x}_i}\mathcal{M}$ em um ponto $\mathbf{x}_i$ é aproximado pelo subespaço gerado pelos d primeiros autovetores (ou seja, as componentes principais) $(\mathbf{e}_1, \mathbf{e}_2, ..., \mathbf{e}_d)$. Os outros $D - d$ autovetores formam um sistema de coordenadas para o espaço normal (LI; LU, 2019).

Figura 9: $T_{\mathbf{x}_i}S$ é o subespaço gerado pelo vetores $\mathbf{e}_1$ e $\mathbf{e}_2$ fornecidos pelo PCA.



Fonte: Elaborado pela autora.

4.3 Métodos dos Mínimos Quadrados

O método dos mínimos quadrados consiste em encontrar a melhor aproximação de um conjunto de dados por meio de uma função específica, minimizando a soma dos quadrados dos erros entre os valores observados e os valores previstos pela função.

Para exemplificar, dado um conjunto de n pontos $\{(x_i, y_i, z_i)_{i=1}^n\} \in \mathbb{R}^3$ e uma função $f : \mathbb{R}^2 \to \mathbb{R}$, o problema consiste em ajustar essa função f aos dados de z_i, minimizando a função do erro quadrático definida por

$$E = \sum_{i=1}^{n} (z_i - f(x_i, y_i))^2.$$

Neste caso, suponha que é desejado aproximar uma superfície por uma função polinomial de grau 2. Seja $f(x, y) = a_0 + a_1x + a_2y + a_3x^2 + a_4xy + a_5y^2$, de modo que

$$E(w) = \sum_{i=1}^{n} (z_i - (a_0 + a_1x_i + a_2y_i + a_3x_i^2 + a_4x_iy_i + a_5y_i^2))^2,$$

onde $w = (a_0, a_1, a_2, a_3, a_4, a_5)$. Para encontrar os coeficientes que minimizam E, deriva-se em relação a cada a_j e iguala-se a zero,

$$\frac{\partial E}{\partial a_i} = 0, \forall i = 0, ..., 5.$$

Isso resulta em um sistema de equações lineares nos coeficientes a_j, cuja solução fornece os parâmetros ótimos necessários.

Pode-se também observar o problema em notação matricial. Para construir a matriz X, considere que cada linha é dada por

$$X_i = \begin{bmatrix} 1 & x_i & y_i & x_i^2 & x_iy_i & y_i^2 \end{bmatrix}_{i=1}^{n}.$$

Sejam $W = \begin{bmatrix} a_0 & a_1 & a_2 & a_3 & a_4 & a_5 \end{bmatrix}^T$ e $Z = [z_i]_{i=1}^n$. Dessa forma, a função do erro quadrático pode ser reescrita como

$$E(W) = ||Z - XW||^2.$$

Para determinar a condição minimizante, impõe-se $\nabla E(W) = \vec{\mathbf{0}}$. Como $E(W) = \langle Z - XW, Z - XW \rangle$, segue que

$$\begin{aligned}\nabla E(W) =& -2\langle X, Z - XW \rangle \\ =& X^T \cdot (Z - XW) \\ =& X^T Z - X^T XW \\ =& \mathbf{0} \in \mathbb{R}^n.\end{aligned}$$

Isso implica que, $X^T XW = X^T Z$, e por consequência, $W = (X^T X)^{-1} X^T Z$. Pois, $(X^T X)^{-1}$ é inversível, visto que suas colunas são linearmente independentes.

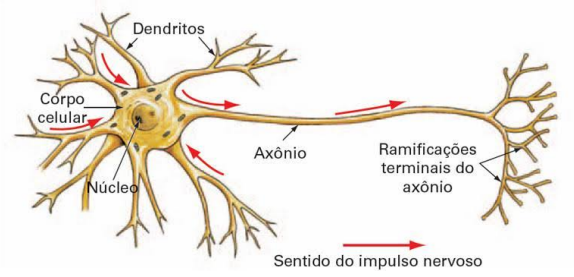
Dessa forma, a solução do problema de mínimos quadrados nos fornece uma superfície S que melhor aproximam os dados, e a expressão $W = (X^T X)^{-1} X^T Z$ nos dá uma solução explícita dos parâmetros que definem S, desde que $(X^T X)^{-1}$ seja inversível.

4.4 Redes Neurais

4.4.1 Perceptron

O neurônio biológico é uma célula de aparência incomum encontrada principalmente no córtex cerebral animal (por exemplo, o cérebro), composto de um corpo celular contendo o núcleo e a maioria dos componentes complexos da célula e muitas extensões de ramificação chamadas dendritos, além de uma extensão alongada chamada de axônio. Perto de sua extremidade, o axônio divide-se em muitos ramos chamados de telodendros, e na ponta desses ramos estão estruturas minúsculas chamadas terminais sinápticos (ou simplesmente sinapses) que estão conectadas aos dendritos de outros neurônios. Os neurônios biológicos recebem curtos impulsos elétricos de outros neurônios através dessas sinapses chamadas sinais. Quando um neurônio recebe um número suficiente de sinais de outros neurônios em alguns milissegundos, ele dispara seus próprios sinais, passando informação adiante (GERON, 2021). Na figura 10, é ilustrado o neurônio biológico.

Figura 10: Neurônio Biológico.



Fonte: Deep learning book, 2023.

O perceptron é um modelo de um neurônio artificial que simula o comportamento de um neurônio biológico. Esse modelo foi idealizado por Frank Resenblatt no final dos anos 50, em (ROSENBLATT, 1958), inspirado nos trabalhos de (MCCULLOCH; PITTS, 1943). Assim como o neurônio biológico, o perceptron é uma unidade básica de processamento, porém em redes neurais artificiais. Inspirado no funcionamento do neurônio, o perceptron replica de forma simplificada seu comportamento: recebe múltiplos sinais de entrada (análogos aos impulsos

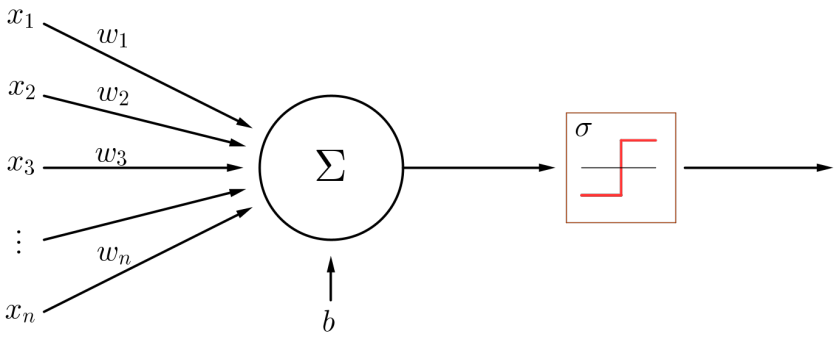
recebidos pelos dendritos), pondera cada uma dessas entradas (simulando a força das sinapses), soma esses valores e produz uma saída (similar ao axônio).

Trazendo para um contexto matemático, dado um vetor $\mathbf{x} = (x_1, x_2, ..., x_n) \in \mathbb{R}^n$ como sendo o dado de entrada e $\mathbf{w} = (w_1, w_2, ..., w_n) \in \mathbb{R}^n$ denotado como o vetor de pesos, definimos o funcional

$$\begin{aligned} f : \mathbb{R}^n &\longrightarrow \mathbb{R} \\ \mathbf{x} &\longmapsto f(x) = \mathbf{w}^T\mathbf{x} + b,\ b \in \mathbb{R}. \end{aligned}$$

Em outras palavras, f faz o papel da soma ponderada entre as entradas e os pesos, e o termo b adicionado é denominado de *bias* ou viés, ele serve para que o modelo não seja limitado a oferecer somente uma saída linear. Ao resultado dessa soma ainda pode ser aplicado uma função de ativação (que será explicado mais adiante). Na figura abaixo segue uma noção intuitiva do processo do modelo perceptron simples.

Figura 11: Neurônio Artificial do Modelo Perceptron



Fonte: Elaborado pela autora.

Em um perceptron na forma $f(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b$ onde $\mathbf{x}, \mathbf{w} \in \mathbb{R}^n$ e $b \in \mathbb{R}$, é possível representar a mesma função em $\mathbb{R}^{n+1}$ da seguinte forma:

$$\tilde{f}(\tilde{\mathbf{x}}) = \tilde{\mathbf{w}}^T\tilde{\mathbf{x}},$$

sendo $\tilde{\mathbf{w}} = \begin{bmatrix} \mathbf{w} \\ b \end{bmatrix}, \tilde{\mathbf{x}} = \begin{bmatrix} \mathbf{x} \\ 1 \end{bmatrix} \in \mathbb{R}^{n+1}$.

Note que a função f não é linear, porém $\tilde{f}$ é. Essa representação é resultado da imersão de $\mathbb{R}^n$ em $\mathbb{R}^{n+1}$.

4.4.2 Funções de Ativação

Uma função de ativação é responsável por transmitir a informação adiante na rede neural, transformando a soma ponderada das entradas e pesos em uma saída que pode ser não linear. No caso de um Perceptron simples, a função de ativação é aplicada à soma ponderada das entradas e dos pesos, podendo ser usada desde a função identidade (que simplesmente retorna o valor do somatório) até várias outras funções que podem ser escolhidas de acordo com a necessidade e complexidade do modelo.

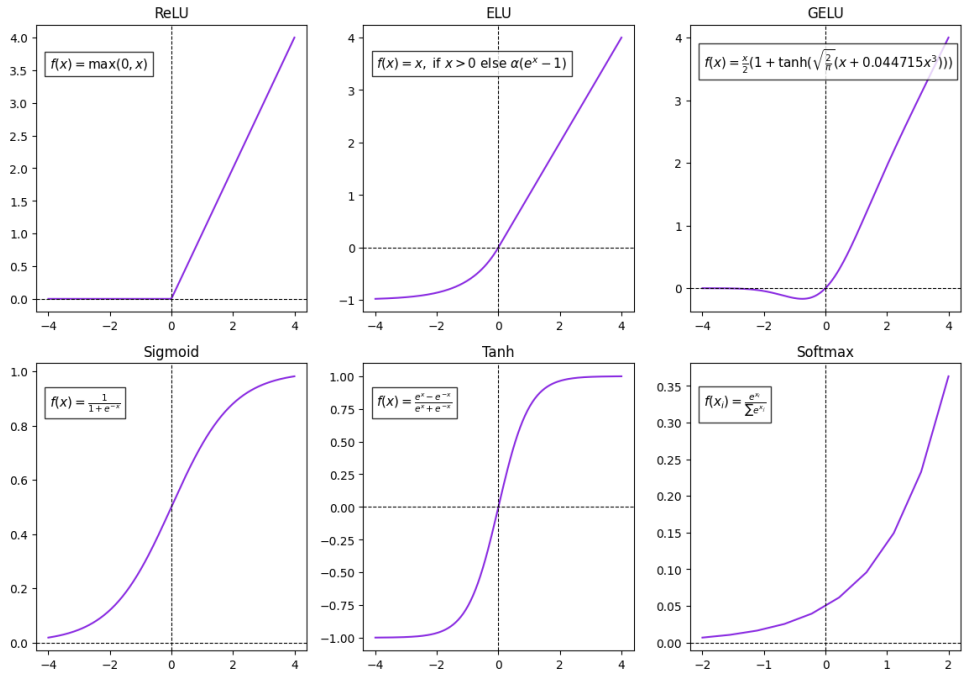
Formalmente, o neurônio pode ser representado como uma composição de funções $g(x) = \sigma(f(x))$, onde $f(x)$ é o resultado do somatório das entradas ponderadas adicionado ao viés, e σ é a função de ativação aplicada a esse resultado.

Dentre os diversos tipos de funções de ativação existentes, destacamos algumas das mais utilizadas, como por exemplo a RELU e suas variações (ELU, SELU, GELU, entre outros), Sigmóide, Tangente Hiperbólica e Softmax. A tabela 1 e a Figura 12, mostram alguns exemplos de funções de ativação.

Tabela 1: Exemplos de Funções de ativação

Nome	**Lei de formação**	**Descrição**
ReLU (Rectified Linear Unit)	$\sigma(x) = \max\{0, x\}$	Devolve o máximo entre 0 e a entrada.
ELU (Exponential Linear Unit)	$\sigma(x) = \begin{cases} x & \text{se } x \geq 0 \\ \alpha(e^x - 1) & \text{se } x < 0 \end{cases}$	Versão suavizada da ReLU que evita neurônios "mortos"ao permitir valores negativos.
GELU (Gaussian Error Linear Unit)	$\sigma(x) = \frac{1}{2}\left[1 + \tanh\left(\sqrt{\frac{2}{\pi}}(x + Cx^3)\right)\right]$, onde $C = 0.044715$.	A GELU suaviza a ReLU, ponderando a entrada x pela probabilidade de que ela seja válida segundo uma distribuição normal.
Sigmóide	$\sigma(x) = \dfrac{1}{1 + \exp^{-x}}$	Retorna a entrada em um valor entre 0 e 1. Frequentemente usada em camadas de saída para classificação binária.
Tangente Hiperbólica (tanh)	$\sigma(x) = \tanh(x)$	Transforma a entrada em um valor entre -1 e 1. Frequentemente usada em camadas ocultas.
Softmax	$\sigma(x_i) = \dfrac{e^{x_i}}{\sum_{j=1}^{n} e^{x_j}}$	Normaliza as saídas em uma distribuição de probabilidade. Usada principalmente em problemas de classificação multiclasse.

Figura 12: Gráficos das funções de ativação



Fonte: Elaborado pela autora.

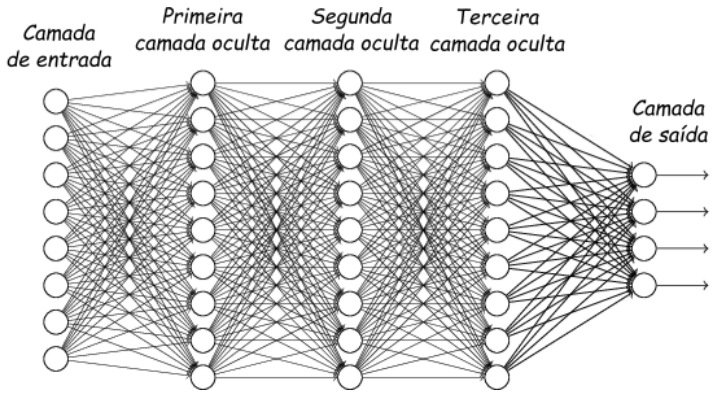
Cada uma dessas funções de ativação tem suas próprias vantagens e desvantagens, e a escolha da função correta depende da arquitetura da rede e do problema específico que se está tentando resolver. Segundo (GERON, 2021), quando as classes são mutuamente exclusivas, a função de ativação softmax geralmente é uma boa opção para as tarefas de classificação na camada de saída. Nas tarefas de regressão, você pode simplesmente não utilizar nenhuma função de ativação para a camada de saída.

4.4.3 Perceptron Multicamadas

O Perceptron Multicamadas, do inglês *Multilayer Perceptron* (MLP), é um tipo de rede neural que se assemelha ao Perceptron simples, mas possui uma arquitetura mais complexa composta por múltiplas camadas de neurônios. Diferente do Perceptron simples, o MLP inclui uma ou mais camadas entre a camada de entrada e de saída, que são nomeadas de camadas ocultas (ou camadas intermediárias) e possibilitam que a rede aprenda padrões mais complexos nos dados.

A profundidade de uma rede neural é determinada pelo número de camadas ocultas que ela possui. Se uma rede neural tem apenas uma camada oculta, ela é chamada de "rasa". Em contrapartida, se a rede possui várias camadas ocultas, ela é denominada de "profunda". A MLP é um exemplo de uma rede totalmente conectada (ou densa), isso se dá ao fato de que toda camada, exceto a camada de saída, inclui um neurônio de viés e está totalmente conectada à próxima camada (GERON, 2021), como ilustrado na figura 13.

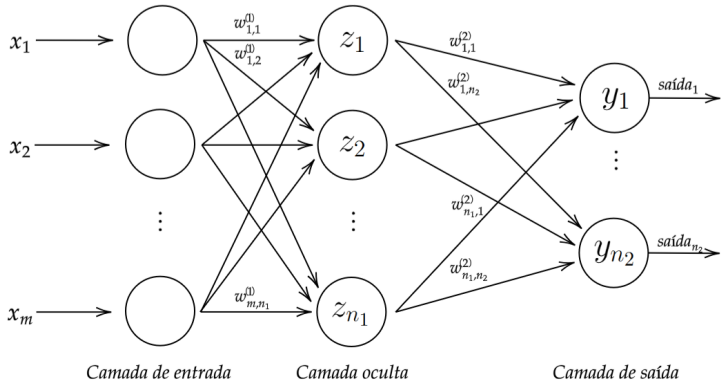
Figura 13: Ilustração de uma MLP profunda e totalmente conectada.



Fonte: Adaptado de (Deep Learning Book, 2022). [4]

Em constraste com o Perceptron simples, a camada de saída do MLP pode conter múltiplos neurônios, o que permite ao modelo realizar tarefas diversas, como por exemplo: a classificação multiclasse ou a predição de múltiplas saídas simultaneamente.

Figura 14: Exemplo de MLP



Fonte: Elaborado pela autora.

No exemplo de arquitetura ilustrada na Figura 14, a rede representada tem como vetor de entrada $(x_1, x_2, ..., x_m) \in \mathbb{R}^m$, $w^{(k)}_{i,j}$ são os pesos associados à conexão entre o neurônio j da camada $k-1$ e o neurônio i da camada k, e $b^{(k)}_j$ é o viés associado ao neurônio j da camada k. A camada de entrada possui m neurônios, a camada oculta possui n_1 neurônios e saída n_2 neurônios.

A rede pode ser descrita como uma série de transformações funcionais (BISHOP, 2006). Primeiramente, constrói-se $M = n_1$ combinações das variáveis de entradas $x_1, ..., x_m$ da seguinte forma:

$$a^{(1)}_i = \sum_{j=1}^{m} w^{(1)}_{ji} x_j + b^{(1)}_i, \ para\ i = 1, ..., n_1,$$

na qual o índice sobrescrito (1) indica a primeira camada oculta. Segundo (BISHOP, 2006), as quantidades a_i são conhecidas como ativações, e cada uma delas é transformada por uma

função de ativação σ, isto é, $z_i = \sigma(a_i^{(1)})$. Os z_i representam as saídas dos neurônios da camada oculta. Em seguida, esses valores são combinados novamente para fornecer

$$a_k^{(2)} = \sum_{i=1}^{n_1} w_{ni}^{(2)} z_i + b_N^{(2)} \; para \; k = 1, ..., n_2$$

e n_2 é o número total de saídas na representação dada pela Figura 14. Essa transformação é correspondente a segunda camada.

Finalmente, as ativações dos neurônios de saída são transformadas usando uma função de ativação, resultando em um conjunto de saída y_k. A escolha dessa função depende da natureza dos dados, neste caso, a função adotada será a a identidade, ou seja, $y_k = a_k^{(2)}, k = 1, ..., n_2$.

Seguindo essa linha de raciocínio, obtém-se a função geral da rede,

$$y_k(x, w) = \sum_{i=1}^{n_1} w_{ni}^{(2)} \sigma \left(\sum_{j=1}^{m} w_{ji}^{(1)} x_j + b_i^{(1)} \right) + b_k^{(2)} \; para \; k = 1, ..., n_2.$$

Assim, segundo (BISHOP, 2006) o modelo da rede neural é simplesmente uma função não linear que mapeia um conjunto de entradas x_i para uma saída y_k, sendo essa transformação definida por um conjunto de parâmetros w ajustáveis.

Também é possível representar a rede usando a notação matricial. Seja $W^{(K)}$ a matriz de pesos da camada k e $b^{(k)} = (b_1^{(k)}, b_2^{(k)}, ..., b_{n_k}^{(k)})$, onde n_k é a quantidade de neurônios da camada $k = 1, 2$. Desta forma, para cada camada k da rede, define-se $f_k(X, W) = (W^{(k)})^T X + b^{(k)}$. Exemplificando, para $k = 1$, tem-se

$$f_1(X, W) = \begin{bmatrix} W_{1,1}^{(1)} & W_{2,1}^{(1)} & ... & W_{m,1}^{(1)} \\ W_{1,2}^{(1)} & W_{2,2}^{(1)} & ... & W_{m,2}^{(1)} \\ \vdots & \vdots & & \vdots \\ W_{1,n_1}^{(1)} & W_{2,n_1}^{(1)} & ... & W_{m,n_1}^{(1)} \end{bmatrix}_{n_1 \times m} \cdot \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{bmatrix}_{m \times 1} + \begin{bmatrix} b_1^{(1)} \\ b_2^{(1)} \\ \vdots \\ b_{n_1}^{(1)} \end{bmatrix}_{n_1 \times 1}.$$

Esta notação matricial facilita a compreensão e implementação dos cálculos realizados pela rede, bem como a análise dos parâmetros envolvidos.

Para treinar a rede, os dados fornecidos são separados em dois conjuntos, um de treinamento e um de teste. Os dados de treino são usados para estimar os parâmetros ótimos da função de custo, ou seja, o treinamento consiste em minimizar uma função de custo L, que definiremos pela função do erro quadrático médio (MSE)

$$L(\mathbf{w}) = \frac{1}{N} \sum_{i=1}^{N} (y_i - f(\mathbf{x}_i, \mathbf{w}))^2$$

onde, N é o número de amostras, y_i são as saídas reais e $f(\mathbf{x}_i, \mathbf{w})$ é a predição da rede para as entradas $\mathbf{x}_i$ parametrizada por $\mathbf{w}$ (composta por pesos e viéses estimados a cada iteração). Os pesos $\mathbf{w}$ são atualizados utilizando o método do gradiente descendente (GD), que em essência, dá um pequeno passo na direção do gradiente negativo, minimizando a função de custo, segundo (BISHOP, 2006), seguindo a regra

$$\mathbf{w}^{n+1} = \mathbf{w}^n - \eta \nabla L(\mathbf{w}^n)$$

onde o parâmetro $\eta > 0$ é conhecido como taxa de aprendizado (*learning rate*) e L é calculado sobre todo o conjunto de treinamento. Para dados com muitas amostras, o gradiente descendente

estocástico (SGD) aproxima o gradiente usando subconjuntos aleatórios chamado de mini-lotes (*mini-batchs*), ele atualiza os pesos usando o gradiente de erros individuais, amostrado ponto a ponto ou em mini-lotes, ajudando a escapar de mínimos locais e acelerando a convergência quando comparado ao GD tradicional.

4.4.4 Retropropagação (*Backpropagation)*

Segundo a formalização descrita por (BISHOP, 2006), uma técnica eficiente para calcular o gradiente de uma função de erro $L(\mathbf{w})$ de uma rede neural *feedforward*[5] é conhecido como retropropagação do erro, ou em inglês, *Backpropagation*. Nessa técnica, a informação é enviada alternadamente para frente e para trás na rede.

Desenvolvido com base na regra da cadeia do cálculo diferencial, esse método propaga sistematicamente os erros da camada de saída para as camadas ocultas, ajustando os pesos sinápticos de maneira a minimizar a função de custo L. Desse modo, o algoritmo de backpropagation é fundamental para o treinamento de redes neurais artificiais, pois permite o cálculo eficiente de gradientes necessários para a otimização via gradiente descendente (GD).

Primeiro considere um modelo simples como o Perceptron 4.4.1, em que as saídas y_k são dadas da forma $y_k = \sum_i w_{ki} x_i$ junto com uma função de erro que, para um padrão de entrada particular com n valores, assume a forma $L_n = \frac{1}{2}\sum_k (y_{nk} - t_{nk})^2$, onde $y_{nk} = y_k(\mathbf{x}_n, \mathbf{w})$. Daí, o gradiente da função de custo com respeito aos pesos w_{ji} são dados por

$$\frac{\partial L_n}{\partial w_{ji}} = (y_{nj} - t_{nj}) x_{ni}.$$

Agora vejamos como este resultado simples se estende ao ambiente mais complexo das redes neurais de múltiplas camadas. Primeiro, notamos que L_n depende do peso w_{ji} apenas através da entrada somada a_j para a unidade j. Portanto, podemos aplicar a regra da cadeia, obtendo

$$\frac{\partial L_n}{\partial w_{ji}} = \frac{\partial L_n}{\partial a_j} \frac{\partial a_j}{\partial w_{ji}}. \tag{9}$$

Como vimos na seção anterior, cada unidade (seja ela oculta ou a camada de saída) calcula $a_i = \sum_j w_{ji} z_j$, onde z_i é a ativação da unidade anterior (ou a entrada x_j se i for da primeira camada oculta), w_{ji} é o peso da conexão entre j e i. Novamente, z_i pode ser obtido aplicando uma função de ativação, $z_i = \sigma(a_i)$.

Iremos introduzir a notação

$$\delta_j \equiv \frac{\partial L_n}{\partial a_j},$$

onde os $\delta' s$ refere-se aos erros. Veja que pela definição de a_j, podemos escrever

$$\frac{\partial a_j}{\partial w_{ji}} = z_i.$$

Daí, da expressão 9, temos que

$$\frac{\partial L_n}{\partial w_{ji}} = \delta_j \cdot z_i.$$

Veja que o erro δ_k na unidades de saída k é dado por

$$\delta_k = y_k - t_k,$$

[5] As redes neurais *feedforward processam dados em uma direção, do nó de entrada para o nó de saída*

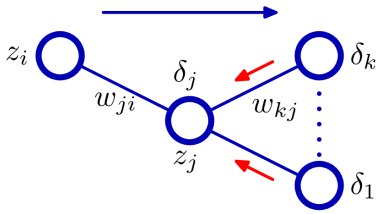
onde y_k é a saída e t_k é o alvo. Para as unidades ocultas j, o erro δ_j é calculado recursivamente é calculado propagando os erros δ_k das unidades k nas camadas posteriores

$$\delta_j \equiv \frac{\partial L_n}{\partial a_j} = \sum_k \frac{\partial L_n}{\partial a_k} \cdot \frac{\partial a_k}{\partial a_j}.$$

onde a soma é feita sobre todas as unidades k para as quais a unidade j envia conexões. Agora, veja que, $\frac{\partial a_k}{\partial a_j} = \frac{\partial}{\partial a_j}(\sum_l w_{kl} z_l) = w_{kl} \cdot \sigma'(a_j)$. Assim,

$$\delta_j = \sigma'(a_j) \sum_k w_{kj} \delta_k.$$

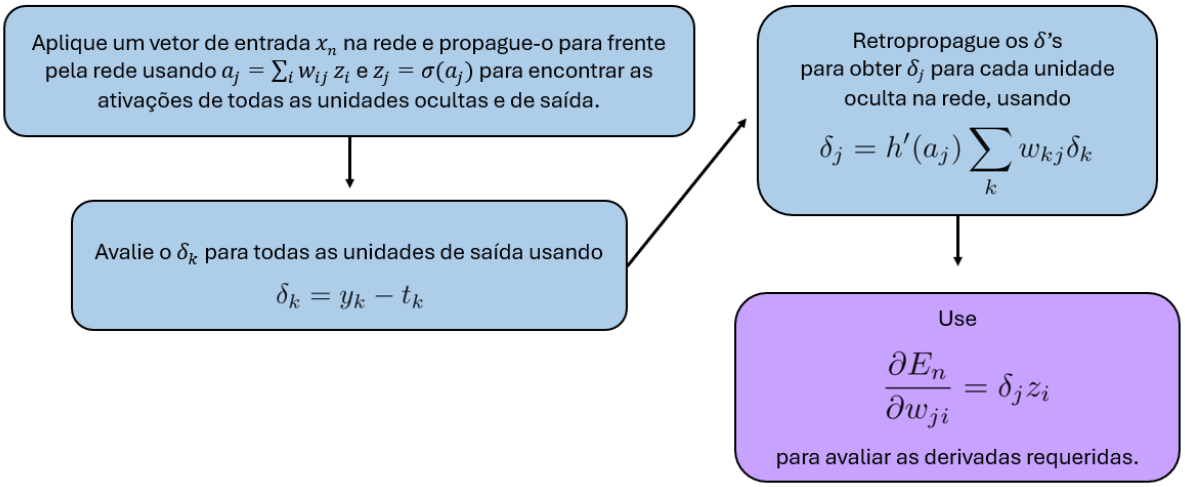
Figura 15: Ilustração do cálculo de δ_j para a unidade oculta j pela retropropagação dos $\delta's$ daquelas unidades k para as quais a unidade j envia conexões. A seta azul denota a direção do fluxo de informação durante a propagação direta, e as setas vermelhas indicam a retropropagação da informação de erro.



Fonte: (BISHOP, 2006)

O método do *backpropagation* pode ser resumido seguindo o diagrama dado na Figura 16.

Figura 16: Processo do Backpropagation

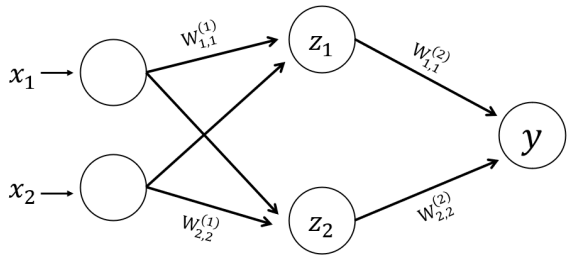


Fonte: Elaborado pela autora.

Para métodos em lote (*batch*), a derivada do erro total L pode ser obtida repetindo os passos acima para cada padrão n no conjunto de treinamento e então somando sobre todos os padrões, ficando com

$$\frac{\partial L}{\partial w_{ji}} = \sum_n \frac{\partial L_n}{\partial w_{ji}}.$$

Exemplo 1. *Vamos considerar um exemplo simples de uma rede neural com uma camada oculta (com função de ativação signmóide) e sem função de ativação na camada de saída.*



Fonte: Elaborado pela autora.

Resolução. *Os pesos da camada oculta (conectando $x$para z) e da camada de saída (conectando z para y), respectivamente, são dados por*

$$W^{(1)} = \begin{bmatrix} w_{11}^{(1)} & w_{12}^{(1)} \\ w_{21}^{(1)} & w_{22}^{(1)} \end{bmatrix}$$
$$W^{(2)} = \begin{bmatrix} w_{11}^{(2)} & w_{21}^{(2)} \end{bmatrix}.$$

Fazendo a propagação para frente, obtemos

$$\begin{aligned} a_1 &= w_{11}^{(1)} x_1 + w_{12}^{(1)} x_2 \\ a_2 &= w_{21}^{(1)} x_1 + w_{22}^{(1)} x_2 \\ z_1 &= \sigma(a_1) = \frac{1}{1+e^{-a_1}} \\ z_2 &= \sigma(a_2) = \frac{1}{1+e^{-a_2}} \\ b &= w_{11}^{(2)} z_1 + w_{21}^{(2)} z_2 \\ y &= b. \end{aligned}$$

A função de perda é dada por $L = \frac{1}{2}(y-t)^2$. Agora iremos fazer o processo de retropopagação. Vamos encontrar o erro na camada de saída, denotado por $\delta^{(y)}$. Como não há função de ativação na saída, então $\dfrac{\partial y}{\partial b} = 1$. Daí,

$$\delta^{(y)} = \frac{\partial L}{\partial b} = \frac{\partial L}{\partial y}\frac{\partial y}{\partial b} = (y-t)\cdot 1 = y - t.$$

Agora, calculemos os gradientes para os pesos da camada de saída $(W^{(2)})$, ficando com

$$\frac{\partial L}{\partial w_{11}^{(2)}} = \delta^{(y)} \cdot z_1 = (y-t)\cdot z_1$$
$$\frac{\partial L}{\partial w_{21}^{(2)}} = \delta^{(y)} \cdot z_2 = (y-t)\cdot z_2.$$

De forma semelhante, encontramos os erros $\delta^{(z)}$ na camada oculta.Precisamos calcular $\delta_1^{(z)}$ e $\delta_2^{(z)}$, daí

$$\delta_1^{(z)} = \frac{\partial L}{\partial a_1} = \frac{\partial L}{\partial b} \cdot \frac{\partial b}{\partial z_1} \cdot \frac{\partial z_1}{\partial a_1} = \delta^{(y)} \cdot w_{11}^{(2)} \cdot \sigma'(a_1)$$
$$\delta_2^{(z)} = \frac{\partial L}{\partial a_2} = \delta^{(y)} \cdot w_{21}^{(2)} \cdot \sigma'(a_2).$$

Finalmente,

$$\frac{\partial L}{\partial w_{11}^{(1)}} = \delta_1^{(z)} \cdot x_1 = \left[\delta^{(y)} \cdot w_{11}^{(2)} \cdot \sigma'(a_1)\right] \cdot x_1$$
$$\frac{\partial L}{\partial w_{12}^{(1)}} = \delta_1^{(z)} \cdot x_2 = \left[\delta^{(y)} \cdot w_{11}^{(2)} \cdot \sigma'(a_1)\right] \cdot x_2$$
$$\frac{\partial L}{\partial w_{21}^{(1)}} = \delta_2^{(z)} \cdot x_1 = \left[\delta^{(y)} \cdot w_{21}^{(2)} \cdot \sigma'(a_2)\right] \cdot x_1$$
$$\frac{\partial L}{\partial w_{22}^{(1)}} = \delta_2^{(z)} \cdot x_2 = \left[\delta^{(y)} \cdot w_{21}^{(2)} \cdot \sigma'(a_2)\right] \cdot x_2.$$

4.4.5 A matriz Jacobiana

Na seção 4.4.4, vimos como as derivadas de uma função de custo em relação aos pesos podem ser obtidas pela propagação de erros para trás através da rede. A técnica de retropropagação também pode ser aplicada ao cálculo de outras derivadas, como a matriz Jacobiana, cujos elementos são dados pelas derivadas das saídas da rede em relação às entradas

$$J_{ki} \equiv \frac{\partial y_k}{\partial x_i}$$

onde cada uma dessas derivadas é avaliada com todas as outras entradas mantidas fixas.

Essa matriz Jacobiana pode ser avaliada usando um modo semelhante ao de retropropagação. Escrevendo o elemento J_{ki} na forma

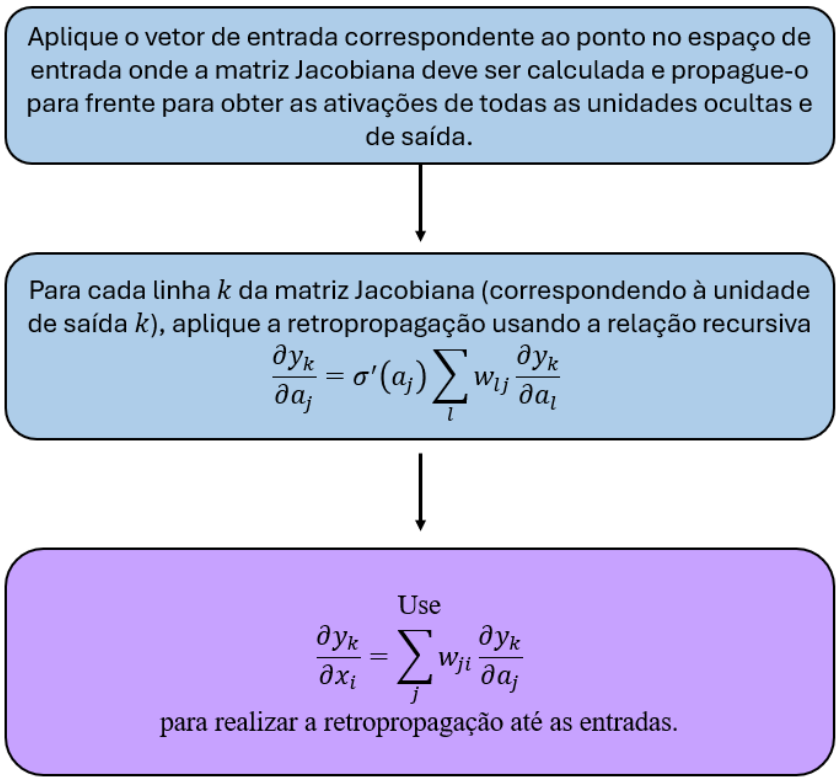
$$\begin{aligned} J_{ki} &= \frac{\partial y_k}{\partial x_i} \\ &= \sum_j \frac{\partial y_k}{\partial a_j}\frac{\partial a_j}{\partial x_i} \\ &= \sum_j w_{ji}\frac{\partial y_k}{\partial x_i} \end{aligned}$$

Agora, basta escrever uma fórmula de retropopagação recursiva para determinar $\dfrac{\partial y_k}{\partial a_j}$, assim

$$\begin{aligned} \frac{\partial y_k}{\partial a_j} &= \sum_l \frac{\partial y_k}{\partial a_l}\frac{\partial a_l}{\partial a_j} \\ &= \sigma'(a_j)\sum_l w_{lj}\frac{\partial y_k}{\partial a_l}. \end{aligned}$$

Essa retropropagação começa nas unidades de saída, para as quais as derivadas necessárias podem ser obtidas diretamente da forma funcional da função de ativação da unidade de saída. O diagrama representado na Figura 17 descreve o processo de cálculo da matriz Jacobiana em uma rede neural, que envolve duas etapas principais: a propagação direta e a retropropagação.

Figura 17: Ilustração do algoritmo para encontrar a jacobiana.



Fonte: Elaborado pela autora.

A diferenciação automática (AD) é um método computacional que calcula derivadas exatas de funções, decompondo-as em operações elementares e aplicando a regra da cadeia do cálculo. Segundo (BAYDIN et al., 2018), a AD é uma família de técnicas semelhantes, mas mais gerais do que a retropropagação. No contexto de redes neurais, o *backpropagation* é uma implementação especializada do modo reverso da AD, onde gradientes são propagados a partir da saída em direção às entradas para eficientemente calcular derivadas parciais em relação aos parâmetros do modelo. Do mesmo modo, o cálculo da matriz Jacobiana também se beneficia da AD reversa, pois cada linha da Jacobiana é obtida retropropagando derivadas das saídas através da rede. Assim, tanto o *backpropagation* quanto a Jacobiana são aplicações diretas da AD.

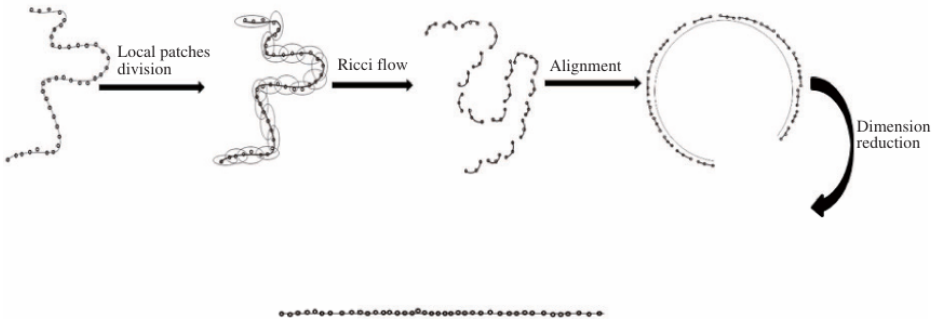
5 Trabalhos Relacionados

O estudo do aprendizado de máquina aliado à geometria tem crescido significamente devido a sua grande relevância em diversos campos de pesquisa. Enquanto a geometria fornece as ferramentas necessárias para compreender e analisar a complexidade de dados, o aprendizado de máquina oferece técnicas para modelar esses dados. Dessa forma, a combinação desses dois tópicos fornece instrumentos poderosos que permitem a resolução de diversos tipos de problemas, visto que essa integração possibilita o desenvolvimento de algoritmos mais eficientes.

No trabalho de (LI; LU, 2019), a principal problemática é a dificuldade em preservar adequadamente as estruturas geométricas de vizinhança ao realizar o processo da redução de dimensionalidade em conjuntos de dados distribuídos em variedades de alta dimensão. Para resolver esse problema, o artigo propõe o uso do fluxo de Ricci para transformar cada vizinhança local da variedade de alta dimensão em uma região de curvatura constante, convertendo a variedade em um subconjunto de uma esfera antes de aplicar as técnicas de redução, proporcionando assim um resultado mais preciso.

A técnica utilizada no artigo é um algoritmo de aprendizado de variedades *(Manifold Learning)* nomeado de *Ricci Flow Manifold Learning* (RF-ML). O método proposto adapta o cálculo do fluxo de Ricci do caso contínuo para o contexto de nuvens de pontos, discretizando as equações que normalmente são utilizadas em variedades contínuas. O algoritmo é dividido em seis passos. Nos três primeiros, é construído uma estrutura de vizinhança para cada ponto usando o KNN e, em seguida, a dimensão intrínseca de cada vizinhança é estimada através de um PCA local. Nas etapas seguintes, o fluxo de Ricci é discretizado junto com o cálculo do tensor métrico e da curvatura de Ricci, permitindo que cada vizinhança seja transformada em uma vizinhança esférica de curvatura constante. Por fim, o algoritmo realiza a redução de dimensionalidade. A representação intuitiva do algoritmo é ilustrada na imagem abaixo.

Figura 18: O processo intuitivo do algoritmo RF-ML. As subimagens da esquerda para a direita são pontos de dados de entrada, vizinhanças sobrepostos, vizinhanças sobrepostos fluindo em patches esféricos discretos usando fluxo de Ricci e alinhamento de patches esféricos para um subconjunto de uma esfera, respectivamente. Na parte inferior está a representação de pontos de dados em um espaço euclidiano de baixa dimensão.



Fonte: (LI; LU, 2019).

Vale destacar que apesar dos bons resultados quando comparado a outros métodos de redução de dimensionalide tradicionais, os autores mencionam que o RF-ML funciona apenas em variedades com curvatura não negativa, o que limita sua aplicação em casos de curvatura negativa, e que abre margem para trabalhos futuros nessa linha.

O artigo de (KAUL; LALL, 2019) tem como objetivo analisar redes neurais profundas

utilizando conceitos da geometria riemanniana, em particular, os tensores de curvatura de Riemann e Ricci, e a curvatura escalar. A ideia busca entender como essas métricas geométricas podem prever o comportamento de redes neurais treinadas relacionando os valores da saída com transformações de entrada a partir das equações de curvatura. A abordagem central do artigo é desenvolver um método sistemático para calcular e visualizar as propriedades de curvatura local das redes neurais profundas. Isso permite compreender como diferentes arquiteturas de redes se comportam diante de transformações específicas.

Para isso, os autores propoem a seguinte abordagem: primeiro é definido a curvatura em redes neurais utilizando como base teorias de geometria diferencial e riemanniana para desenvolver as métricas nas redes neurais. Em seguida, os tensores de métrica, símbolos de Christoffel e tensores de curvatura de Riemann e Ricci são calculados a partir das saídas softmax das redes. Após isso, a curvatura é medida para diferentes redes treinadas usando o método do gradiente descendente estocástico (SGD).

Os modelos de rede utilizadas incluem os perceptrons multicamadas (MLPs) e as redes convolucionais (CNNs) aplicadas a conjuntos de dados como MNIST e CIFAR-10. Além disso, os dados de entrada são modificados incrementalmente por meio de transformações como rotação e translação. As saídas das redes são analisadas para calcular métricas diferenciais e visualizar como a curvatura evolui com tais mudanças.

Por fim, foi visto que os resultados alcançados com a proposta foram positivos e indicaram principalmente que as regiões de transição entre classes possuem curvaturas significativamente altas, o que sugere que a rede neural pode utilizar essa característica para classificação. Além disso, foi visto que o aumento na profundidade da rede aumenta a curvatura e também, para classes linearmente separáveis, a curvatura é baixa, indicando que a rede não precisa de transformações não lineares. Por fim, nos experimentos com o conjunto de dados CIFAR-10, foi visto que a curvatura é alta nas fronteiras entre classes, reforçando a ideia de que a curvatura está relacionada à arquitetura da rede.

O trabalho de (BAPTISTA et al., 2024) tem como objetivo estabelecer uma conexão entre o processo de transformação geométrica dos dados em redes neurais profundas (DNNs) e o fluxo de Ricci. A intenção é propor uma nova forma de entender e avaliar as mudanças geométricas que ocorrem nos dados ao passar por uma rede neural, usando conceitos de geometria diferencial. Para ilustrar esta ideia, os autores apresentam uma estrutura computacional para quantificar as mudanças geométricas que ocorrem à medida que os dados passam por camadas sucessivas de uma DNN. Eles utilizam esta estrutura para motivar uma noção de 'fluxo de rede de Ricci global' que pode ser usada para avaliar a capacidade de uma DNN de desatar as geometrias de dados complexos para resolver problemas de classificação. Para validar, os autores treinaram mais de 1500 classificadores DNN de diferentes larguras e profundidades em dados sintéticos e do mundo real, e mostraram com isso, que a força do comportamento semelhante ao fluxo de rede de Ricci global correlaciona-se com a precisão para DNNs bem treinadas, independentemente de profundidade, largura e conjunto de dados.

O estudo reconhece que a adaptação do fluxo de Ricci para ambientes discretos (como redes neurais) apresenta desafios na precisão e interpretação das medições. Além disso, a análise é mais robusta para redes bem treinadas e conjuntos de dados de estrutura relativamente compreensível, já em contextos com dados complexos ou não estruturados, a métrica pode não captar toda a complexidade das transformações geométricas. Ainda, o artigo aponta que a compreensão do comportamento de redes profundas com ativação não suave, como ReLU, ainda é um campo em aberto, o que abre possibilidades para trabalhos futuros nessa linha.

Ainda sobre relacionar geometria com aprendizado de máquina, dois trabalhos recentes abordam o problema a partir de perspectivas semelhantes. O estudo de (LEE et al., 2022) in-

troduz uma estrutura geométrica riemanniana baseada na métrica de Fisher para tratar nuvens de pontos como distribuições de probabilidade, permitindo medir distâncias, ângulos e volumes de forma geometricamente consistente e resultando em interpolações de forma mais suaves e representações latentes mais significativas. Em linha similar, porém com uma abordagem baseada em aprendizado profundo, (FASINA et al., 2023) traz o *Neural FIM*, um método que utiliza redes neurais para inferir a métrica riemanniana diretamente dos dados, permitindo calcular geodésicas, volumes locais e analisar a estrutura da variedade subjacente.

Todos os trabalhos apresentados demonstram a relevância da integração entre geometria e aprendizado de máquina para a análise e otimização de algoritmos, utilizando ferramentas geométricas — como curvatura riemanniana, métrica de Fisher ou fluxo de Ricci — para capturar a estrutura intrínseca dos dados. Enquanto (LI; LU, 2019) aborda a preservação de estruturas geométricas em redução de dimensionalidade via fluxo de Ricci discreto, (KAUL; LALL, 2019) explora a curvatura riemanniana para interpretar o comportamento de redes neurais profundas, e (BAPTISTA et al., 2024) avança nessa linha, propondo um fluxo de Ricci global para avaliar transformações geométricas em redes neurais, relacionando com desempenho em tarefas de classificação. Um aspecto comum entre esses estudos é a modelagem geométrica para melhorar interpolações, calcular geodésicas ou compreender o comportamento de modelos, reforçando a importância de aliar geometria e aprendizado de máquina. No entanto, também são apontadas limitações e desafios, como a restrição a curvaturas negativas ou a complexidade inerente a dados não estruturados, abrindo caminho para investigações futuras.

Diante desse contexto, optamos por basear nossa abordagem no método proposto por (LI; LU, 2019). Apesar dos autores não disponibilizarem o código-fonte, o artigo fornece uma descrição do algoritmo, junto com um pseudo-código, o que facilita significativamente a reprodução e a adaptação da proposta. Além disso, os próprios autores reconhecem as limitações de seu método em variedades de curvatura negativa, o que motivou nossa investigação sobre o uso de redes neurais especificamente para esse contexto. Acreditamos que a flexibilidade representacional desses modelos possui potencial para superar as restrições mencionadas, promovendo assim uma melhor capacidade de generalização em diversos cenários.

6 Metodologia Proposta

No estudo da geometria em dados provenientes de nuvens de pontos, enfrenta-se o desafio de calcular e interpretar certas propriedades geométricas. Este problema tem implicações diretas em diversas áreas, como aprendizado de máquina, motivando a necessidade de métodos eficientes e precisos para a estimativa de propriedades geométricas diversas como: a métrica, as curvas geodésicas, o tensor de curvatura de Ricci, entre outros. O objetivo do nosso trabalho é calcular em específico o tensor de curvatura de Ricci de uma nuvem de pontos dada. Para isso, será utilizado e adaptado um método apresentado na literatura, no artigo de (LI; LU, 2019), que embora seu foco principal seja o uso do fluxo de Ricci para melhorar técnicas de redução de dimensionalidade, o artigo apresenta ferramentas que estimam os tensores requeridos, as quais serão fundamentais para este trabalho.

6.1 Método Original

No método original (LI; LU, 2019), tem-se que uma representação suave para uma vizinhança $U_i \subset \mathbb{R}^d$ no sistema de coordenadas local é descrita por

$$f(x_1, x_2, ..., x_d) = (x_1, ..., x_d, f_{d+1}(x_1, x_2, ..., x_d), ..., f_D(x_1, x_2, ..., x_d)), \tag{10}$$

onde $(x_1, x_2, ..., x_d) \in U_i$ e as funções coordenadas f_α com $\alpha = d+1, ..., D$ são definidas como $f_\alpha(\mathbf{x}) = W_\alpha \phi^T(\mathbf{x})$, sendo $W_\alpha = [a_0^\alpha, a_1^\alpha, ..., a_d^\alpha, a_{11}^\alpha, a_{12}^\alpha, ..., a_{dd}^\alpha]$ é o vetor dos coeficientes determinados pelos mínimos quadrados e $\phi = [1, x_1, ..., x_d, x_1x_1, x_1x_2, ..., x_dx_d]$ é composto por $1, x_1, ..., x_d$ e monômios de grau 2 sobre as variáveis $x_1, ..., x_d$.

Sob a representação suave de f de U_i, a base vetorial tangente correspondente em $\mathbf{x}_i$ é dada por $\left\{ \frac{\partial f}{\partial \mathbf{x}_1}(\mathbf{x}_i), ..., \frac{\partial f}{\partial x_d}(\mathbf{x}_i) \right\}$, onde

$$\frac{\partial f}{\partial \mathbf{x}_j} = \left(0, ..., 1_j, ..., 0, W_{d+1} \frac{\partial \phi^T}{\partial \mathbf{x}_j}(\mathbf{x}_i), ..., W_D \frac{\partial \phi^T}{\partial \mathbf{x}_j}(\mathbf{x}_i) \right). \tag{11}$$

O tensor métrico local é calculado a partir dos vetores acima e pela expressão dada na Definição (7), ou seja, $G_{jk} = [g_{ik}]$, onde $g_{jk} = \langle \frac{\partial f}{\partial x_j}, \frac{\partial f}{\partial x_k} \rangle$. No caso contínuo, o fluxo de Ricci é dado por

$$\frac{\partial}{\partial t} g_{ij} = -2Ric_{ij}.$$

Em uma subvariedade imersa (Ver Apêndice 8), Ric_{ij} pode ser expresso em termos da segunda forma fundamental. Veja que, pela equação de Gauss (Teorema 4), temos

$$\widetilde{Rm}(X, Y, Z, W) = Rm(X, Y, Z, W) - \langle II(X, W), II(Y, Z) \rangle + \langle II(X, Z), II(Y, W) \rangle.$$

Se o espaço ambiente $\widetilde{\mathcal{M}} = \mathbb{R}^D$, então $\widetilde{Rm}(X, Y, Z, W) = \mathbf{0}$. A curvatura riemanniana de $\mathcal{M}$ pode ser calculada somente pela segunda forma fundamental, desse modo

$$Rm(X, Y, Z, W) = \langle II(X, W), II(Y, Z) \rangle - \langle II(X, Z), II(Y, W) \rangle.$$

Em um sistema de coordenadas locais, a segunda forma fundamental pode ser representada como

$$II(X, Y) = \sum_{\alpha=d+1}^{D} h^\alpha(X, Y) \epsilon_\alpha,$$

onde ϵ_α é um elemento da base de campo normal de $\mathcal{M}$ e $h^\alpha(X,Y)$ é dado pela segunda derivada da função mergulhada f.

Daí, tomando os vetores da base canônica $X = e_l, Y = e_i, W = e_l$ e $Z = e_j$ onde $l = 1, ..., d$ e $i, j = \{1, ..., d, ..., D\}$. Temos

$$\begin{aligned} Rm(X,Y,W,Z) =& Rm(e_l, e_i, e_l, e_j) \\ =& \langle II(e_l, e_l), II(e_i, e_j)\rangle - \langle II(e_l, e_j), II(e_i, e_l)\rangle \\ =& \left\langle \sum_{\alpha=d+1}^{D} h^\alpha_{ll}\epsilon_\alpha, \sum_{\alpha=d+1}^{D} h^\alpha_{ij}\epsilon_\alpha \right\rangle - \left\langle \sum_{\alpha=d+1}^{D} h^\alpha_{lj}\epsilon_\alpha, \sum_{\alpha=d+1}^{D} h^\alpha_{il}\epsilon_\alpha \right\rangle \\ =& \sum_{\alpha=d+1}^{D} (h^\alpha_{ll}h^\alpha_{ij} - h^\alpha_{lj}h^\alpha_{il}). \end{aligned}$$

Extraindo o traço, encontramos a expressão para o tensor de curvatura de Ricci. Logo,

$$Ric_{ij} = \sum_{\alpha=d+1}^{D}\sum_{l=1}^{d}(h^\alpha_{ll}h^\alpha_{ij} - h^\alpha_{lj}h^\alpha_{il}).$$

Agora, precisamos discretizar os operadores diferenciais usando diferenças finitas. Suponha $V_j = \left(\frac{\partial f_{d+1}}{\partial x_j}, \frac{\partial f_{d+2}}{\partial x_j}, ..., \frac{\partial f_D}{\partial x_j}\right)$, a equação do fluxo de Ricci pode ser representada em função de V_j, ou seja, $\frac{\partial}{\partial t}V_j = F(V_j, \nabla V_j)$, quando

$$F(V_j, \nabla V_j) = -\sum_{\alpha=d+1}^{D}\sum_{l=1}^{d}\left(\frac{\partial^2 f_\alpha}{\partial x_l \partial x_l}\cdot\frac{\partial^2 f_\alpha}{\partial x_j \partial x_j} - \frac{\partial^2 f_\alpha}{\partial x_l \partial x_j}\cdot\frac{\partial^2 f_\alpha}{\partial x_l \partial x_j}\right)\cdot\left(\frac{\partial f}{\partial x_j}\right)^\perp$$

, onde $\perp$ denota a componente normal. Desse modo, o tensor métrico local, e o tensor de curvatura de Ricci correspondente podem ser iterados sob o fluxo de Ricci.

Semelhante a equação 11, podemos reescrever $\dfrac{\partial f}{\partial x_j} = (e_j, V_j),\ j = 1, ..., d$ onde $e_j \in \mathbb{R}^d$ e $V_j \in \mathbb{R}^{D-d}$. Sabemos que a métrica é calculada como $g_{jk} = \langle\frac{\partial f}{\partial x_j}, \frac{\partial f}{\partial x_k}\rangle$. Considerando a métrica euclidiana, teremos

$$g_{jk} = \delta_{jk} + V_j \cdot V_k^T,$$

onde δ_{jk} é o chamado delta de Kronecker[6]. Agora, ao invés de trabalhar com $\frac{\partial^2 f}{\partial x_i \partial x_j}$, substituímos por $\nabla_l V_j^{n+1}$, que será a aproximação da segunda derivada. Usando o fato de que $\frac{\partial}{\partial t}V_j = F(V_j, \nabla V_j)$ aliado ao método de diferenças finitas, temos

$$\begin{aligned} &\frac{V_j^{n+1} - V_j^n}{\Delta t} = F(V_j^n, \nabla V_j^n) \\ &\Rightarrow V_j^{n+1} = V_j^n + \Delta t F(V_j^n, \nabla V_j^n),\ j = 1, ..., d. \end{aligned}$$

Logo,

$$g_{jk}^{n+1} = \delta_{jk} + V_j^{n+1} \cdot V_k^{n+1^T},$$

[6] $\delta_{jk} = 1$ se $j = k$, e $\delta_{jk} = 0$ se $j \neq k$.

$$Ric_{jk}^{n+1} = \sum_{l} \left(\nabla_l V_l^{n+1} \cdot \nabla_k V_j^{n+1^\top} - \nabla_l V_j^{n+1} \cdot \nabla_l V_k^{n+1^T} \right).$$

A noção do algoritmo do trabalho de (LI; LU, 2019) pode ser explicada no Algoritmo 1. No nosso caso, como queremos apenas calcular o tensor de Ricci, iremos até o passo de atualizar as equações, ilustrado no pseudo-código.

Algoritmo 1: Aplicação do Fluxo de Ricci ao Aprendizado de Variedades (RF-ML)

Entrada: Pontos de treinamento $\{x_1, x_2, \ldots, x_N\} \in \mathbb{R}^D$, parâmetro de vizinhança K

início

  para $i = 1$ *até* N **faça**

    Encontrar os K-vizinhos mais próximos de $\mathbf{x}_i$

    Calcular o espaço tangente $T_{\mathbf{x}_i}\mathcal{M}$

    Atualizar as equações do fluxo de Ricci dadas por V_j^{n+1}, g_{jk}^{n+1} e Ric_{jk}^{n+1} **até** *convergência*

    Alinhar os pontos deformados $\{Y_1, Y_2, \ldots, Y_N\}$ a um subconjunto completo P da esfera

    Aplicar algoritmos tradicionais de aprendizado de variedades para reduzir a dimensão dos pontos esféricos

  fim

fim

Saída: Representações de baixa dimensão $\{z_1, z_2, \ldots z_N\} \in \mathbb{R}^d$

Para entender melhor, consideremos o caso de uma superfície (ou seja, uma variedade bidimensional). Desse modo, seguindo o método, a representação local da função pode ser simplificada para um gráfico da forma $f(u, v) = (u, v, f_3(u, v))$, (tomando $x_1 = u$ e $x_2 = v$) em que $f_3(u, v)$ é modelada por um polinômio de segunda ordem, expresso por

$$f_3(u, v) = a_0 + a_1 u + a_2 v + a_3 u^2 + a_4 uv + a_5 v^2, \tag{12}$$

com os coeficientes $a_0, a_1, \ldots, a_5$ determinados pelo método dos mínimos quadrados. Para esse caso, temos que o campo é somente $V_j = \dfrac{\partial f_3}{\partial x_j}$ e

$$\begin{bmatrix} F(V_1, \nabla V_1) \\ F(V_2, \nabla V_2) \end{bmatrix} = \begin{bmatrix} -\left(\frac{\partial^2 f_3}{\partial v^2} \cdot \frac{\partial^2 f_3}{\partial u^2} - \frac{\partial^2 f_3}{\partial v \partial u} \cdot \frac{\partial^2 f_3}{\partial v \partial u} \right) \cdot \left(\frac{\partial f}{\partial u} \right)^{\perp} \\ -\left(\frac{\partial^2 f_3}{\partial u^2} \cdot \frac{\partial^2 f_3}{\partial v^2} - \frac{\partial^2 f_3}{\partial u \partial v} \cdot \frac{\partial^2 f_3}{\partial u \partial v} \right) \cdot \left(\frac{\partial f}{\partial v} \right)^{\perp} \end{bmatrix}.$$

Além disso,

$$V^{n+1} = \begin{bmatrix} V_1^n + \Delta t F(V_1^n, \nabla V_1^n) \\ V_2^n + \Delta t F(V_2^n, \nabla V_2^n) \end{bmatrix},$$

$$G = [g_{jk}^{n+1}] = \begin{bmatrix} 1 + (V_1^{n+1})^2 & V_1^{n+1} \cdot V_2^{n+1} \\ V_2^{n+1} \cdot V_1^{n+1} & 1 + (V_2^{n+1})^2 \end{bmatrix},$$

Deste modo,

$$R^{n+1} = [Ric_{jk}^{n+1}]$$
$$= \begin{bmatrix} \nabla_2 V_2^{n+1} \cdot (\nabla_1 V_1^{n+1})^T - \nabla_2 V_1^{n+1} \cdot (\nabla_2 V_1^{n+1})^T & \nabla_1 V_1^{n+1} \cdot (\nabla_2 V_1^{n+1})^T - \nabla_1 V_1^{n+1} \cdot (\nabla_1 V_2^{n+1})^T \\ \nabla_2 V_2^{n+1} \cdot (\nabla_1 V_2^{n+1})^T - \nabla_1 V_1^{n+1} \cdot (\nabla_1 V_2^{n+1})^T & \nabla_1 V_1^{n+1} \cdot (\nabla_2 V_2^{n+1})^T - \nabla_1 V_2^{n+1} \cdot (\nabla_1 V_2^{n+1})^T \end{bmatrix}.$$

Os experimentos ilustrados na seção 7 seguiram essa ideia. Além disso, foi construído um comparativo com o caso contínuo em algumas superfícies regulares.

6.2 Adaptação com Redes Neurais

A adaptação proposta neste trabalho é substituir a aproximação polinomial de (LI; LU, 2019) por uma rede neural do tipo MLP, adaptando assim o método de mínimos quadrados tradicionalmente utilizado para aproximar funções para uma abordagem baseada em redes neurais. Isto é, na expressão dada na equação 10, agora as funções f_α serão dadas pela rede neural. Esperamos com esta adaptação construir um modelo mais geral que ajude a observar casos que a versão original não aborda, como superfícies com curvatura negativa, além de explorar o potencial de generalização das redes neurais e avaliar se elas produzem resultados mais precisos em comparação com o método anterior.

Para analisar os resultados, os dois métodos – o tradicional e o adaptado – serão implementados em superfícies (variedades bidimensionais) conhecidas, como a esfera, o parabolóide e a sela. Após a implementação, os resultados obtidos serão analisados e comparados, avaliando se a nova proposta trouxe resultados coerentes. Além disso, iremos comparar os resultados com métodos já existentes na literatura que calculam versões da curvatura de Ricci, como Forman-Ricci e Ollivier-Ricci.

Pela equação 10, temos que

$$f(x_1, x_2, ..., x_d) = (x_1, ..., x_d, f_{d+1}(x_1, x_2, ..., x_d), ..., f_D(x_1, x_2, ..., x_d)).$$

Queremos estimar as funções coordenadas de classe $C^2(\Omega)$, com $\Omega \subset \mathbb{R}^d$, $f_\alpha : \Omega \subset \mathbb{R}^d \longrightarrow \mathbb{R}$ para cada $\alpha = d+1, ..., D$. A parametrização f será aproximada por uma regressão através de uma rede neural do tipo MLP. Após aproximar f_α, obtemos suas derivadas parciais e portanto, seus respectivos gradientes dados por ∇f_α por meio da diferenciação automática. Como a precisão da segunda derivada não é suficiente para resolver o problema de estimar o tensor de curvatura, definiremos mais uma rede neural, $H_\alpha = \nabla f_\alpha$. Desse modo, teremos algo como $\nabla f_\alpha = (\frac{\partial f_\alpha}{\partial x_1}, ..., \frac{\partial f_\alpha}{\partial x_d})$, ou ainda,

$$H_\alpha = \begin{bmatrix} h_{1\alpha} \\ \vdots \\ h_{d\alpha} \end{bmatrix}, \text{onde } h_{j\alpha} = \frac{\partial f_\alpha}{\partial x_j}.$$

A partir disso, podemos obter as segundas derivadas pela diferenciação automática de H_α. Ou seja, usando as derivadas parciais $\dfrac{\partial f_\alpha}{\partial x_i}$, faremos uma nova regressão, afim de obter a função H_α. Em seguida, aplicamos o mesmo procedimento para obter as derivadas parciais de H_α, e assim, contruímos

$$JH_\alpha = \begin{bmatrix} \nabla h_{1\alpha} \\ \vdots \\ \nabla h_{d\alpha} \end{bmatrix}.$$

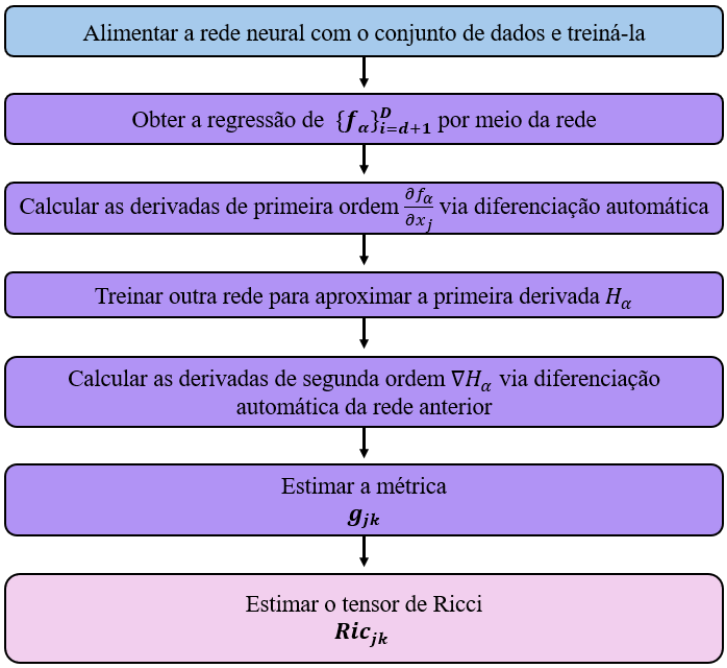
Após estimar as primeiras e segundas derivadas parciais de f_α, podemos finalmente utilizar as equações para calcular o tensor métrico g_{jk} e Ric_{jk}, desse modo, de forma semelhante ao método original, ficamos com

$$g_{ij} = \delta_{jk} + \sum_{\alpha=d+1}^{D} \frac{\partial f_\alpha}{\partial x_j} \frac{\partial f_\alpha}{\partial x_k} \tag{13}$$

$$Ric_{jk} = \sum_{l=1}^{d} \left(\frac{\partial^2 f_\alpha}{\partial x_l \partial x_l} \cdot \frac{\partial^2 f_\alpha}{\partial x_j \partial x_k} - \frac{\partial^2 f_\alpha}{\partial x_l \partial x_j} \cdot \frac{\partial^2 f_\alpha}{\partial x_k \partial x_l} \right). \tag{14}$$

O algoritmo da adaptação funcionará intuitivamente de acordo com a Figura 19. Os experimentos seguindo esse método estão ilustrados na seção 7.

Figura 19: Diagrama ilustrando o algoritmo adaptado para redes neurais



Elaborado pela autora.

6.3 Dificuldades e Ajustes Metodológicos

Durante o desenvolvimento da metodologia proposta, algumas dificuldades práticas surgiram no processo de estimação das derivadas de segunda ordem. Inicialmente, testamos a utilização de derivadas simbólicas, mas o custo computacional mostrou-se extremamente elevado, mesmo em exemplos simples, além de gerar resultados muito próximos àqueles obtidos pela diferenciação automática. Por esse motivo, descartamos essa abordagem.

Figura 20: Regressão do Semi-círculo através de rede neural

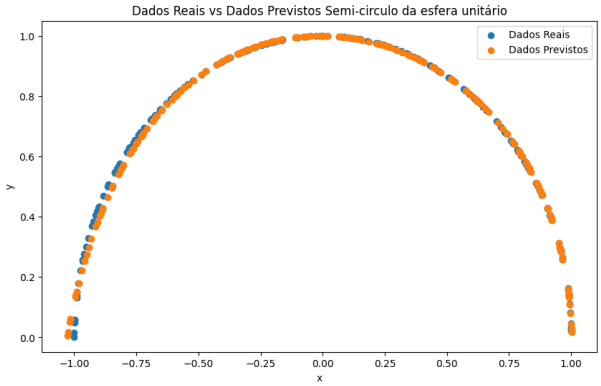


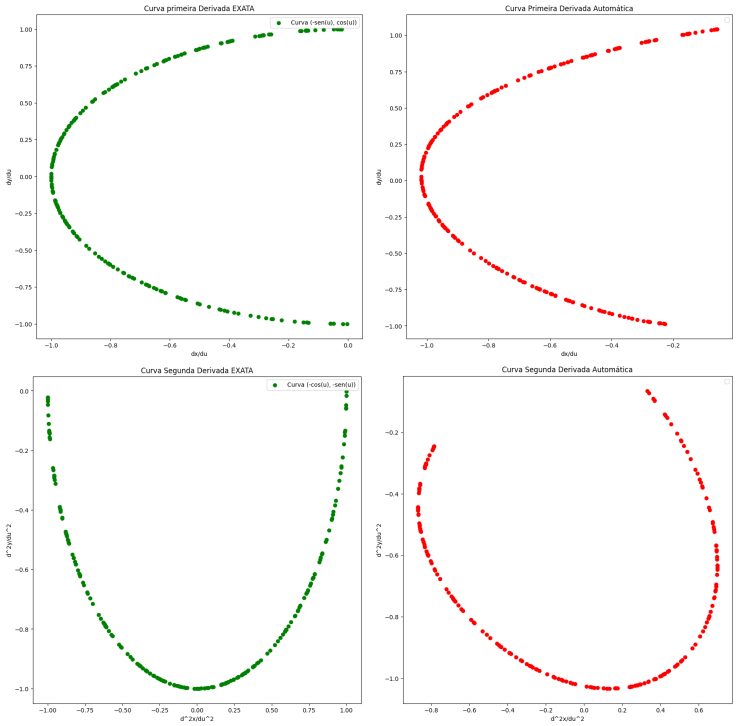
Figura 21: Elaborado pela autora.

A utilização direta da diferenciação automática, embora mais viável, também apresentou limitações, sobretudo na estabilidade numérica e na precisão das segundas derivadas necessárias ao cálculo da métrica e de quantidades geométricas associadas. Como alternativa, optamos pela construção de uma rede neural adicional dedicada especificamente à estimação dessas segundas derivadas. Vejamos um exemplo para um semi-círculo de parametrização $f(u) = (cos(u), sen(u))$ com $0 \leq u \leq \pi$.

Inicialmente, foi treinada uma rede neural para realizar a regressão do semi-círculo, utilizando uma arquitetura composta por três camadas ocultas com 32 neurônios em cada camada e a função de ativação GELU. O resultado obtido é apresentado em laranja na Figura 21.

Em seguida, empregamos a diferenciação automática diretamente para o cálculo das primeiras e segundas derivadas. Note que, quando comparada com a derivada exata, a segunda derivada apresenta discrepâncias significativas.

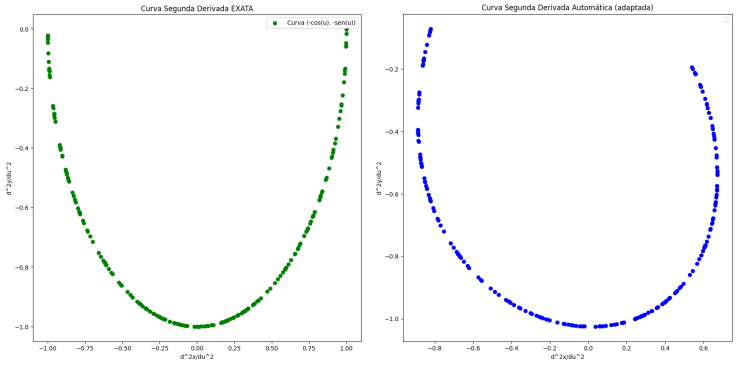
Figura 22: Comparação das curvas derivadas exatas com as obtidas por diferenciação automática.



Elaborado pela autora.

Com o objetivo de reduzir essa diferença, adotamos uma estratégia alternativa: a definição de uma rede neural auxiliar, com a mesma arquitetura da anterior, treinada especificamente para aproximar as primeiras derivadas automáticas. A partir dessa rede auxiliar, foi então calculada a derivada automática, que corresponde à estimativa da segunda derivada. Esse procedimento resultou em aproximações mais consistentes, conforme ilustrado na Figura 23.

Figura 23: Comparação da curva segunda derivada exata com a obtida pela diferenciação automática da segunda rede.

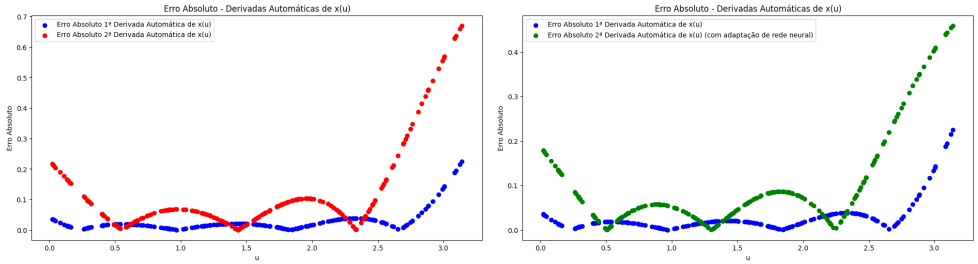


Elaborado pela autora.

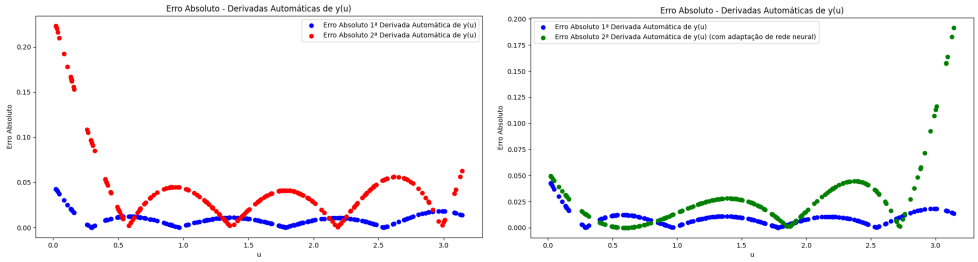
Para validar a melhoria alcançada, realizamos a comparação do erro absoluto entre as derivadas exatas e aquelas obtidas pelas duas abordagens (diferenciação direta e método adaptado). Para visualizar o erro, plotamos separadamente para cada uma das componentes $x(u)$ e

$y(u)$, como ilustrado na Figura 24 e nas tabelas 2 e 3.

Figura 24: Erros absolutos da segunda derivada



Elaborado pela autora.



Elaborado pela autora.

Tabela 2: Erro Absoluto das Segundas Derivadas de $x(u)$.

	2º Derivada Automática de $x(u)$	**2º Derivada Automática Adaptada de** $x(u)$
Erro absoluto mínimo	4.71999×10^{-4}	5.08529×10^{-4}
Erro absoluto máximo	6.70299×10^{-1}	4.59171×10^{-1}
Mediana do erros absolutos	6.60069×10^{-2}	6.02835×10^{-2}
Média dos erros absolutos	1.13131×10^{-1}	9.36602×10^{-2}

Tabela 3: Erro Absoluto das Segundas Derivadas de $y(u)$.

	2º Derivada Automática de $y(u)$	**2º Derivada Automática Adaptada de** $y(u)$
Erro absoluto mínimo	2.46218×10^{-4}	1.88232×10^{-5}
Erro absoluto máximo	2.23274×10^{-1}	1.916834×10^{-1}
Mediana do erros absolutos	3.70069×10^{-2}	2.24410×10^{-2}
Média dos erros absolutos	4.43220×10^{-2}	2.59680×10^{-2}

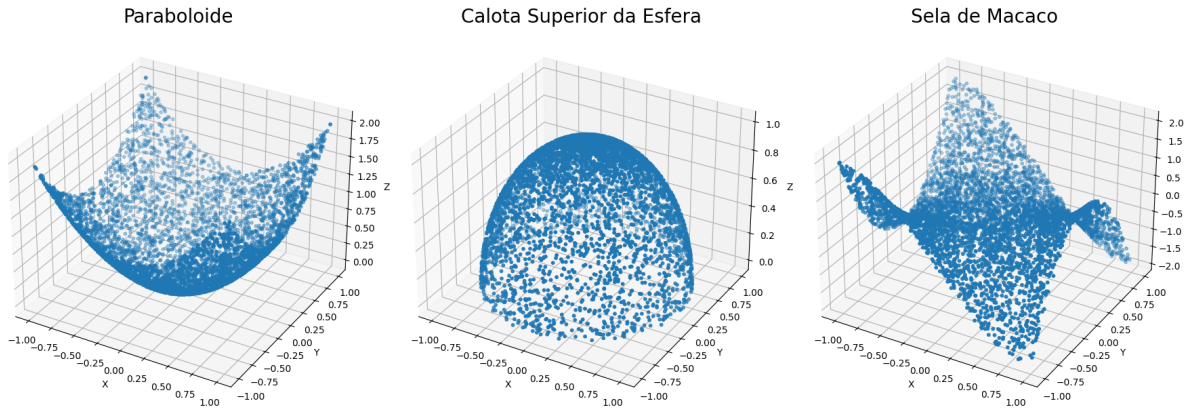
Desse modo, seguimos com essa estratégia para as estimativas das derivadas de segunda ordem, como demonstrado na seção anterior.

7 Experimentos Computacionais

Esta seção descreve os experimentos computacionais realizados para validar os métodos propostos na seção 6. Utilizamos o ambiente do *Google Colab*, com configuração de 12.7 GB de RAM e um processador *Intel(R) Xeon(R) CPU @ 2.20GHz model* 79. A implementação foi realizada em *Python 3.11.12*. Os experimentos foram realizados em três superfícies: o parabolóide (curvatura gaussiana positiva), a esfera (curvatura constante) e a sela de macaco (curvatura negativa). A escolha dessas superfícies permite uma boa avaliação da eficácia do método, visto que elas representam casos distintos de curvatura gaussiana.

Os dados foram gerados sinteticamente a partir das parametrizações de cada superfície. A figura 25 ilustra uma amostra de 5000 pontos nas superfícies simuladas, destacando sua geometria e distribuição.

Figura 25: Amostra de 5000 pontos nas superfícies: parabolóide, esfera e sela de macado.



Fonte: Elaborado pela autora.

As bibliotecas *numpy*[7], *random* e *matplotlib*[8] foram utilizadas para gerar, manipular e visualizar os dados de todas as superfícies analisadas. No método orriginal, o *scikit-learn*[9] auxiliou no cálculo de vizinhanças e na aplicação do PCA. Para cada superfície, os parâmetros do método proposto por (LI; LU, 2019) foram: número máximo de iterações igual a 100, passo de tempo adaptativo com Δt inicial de $0,001$, atualizado a cada iteração pela regra $\Delta t = 10^{-3}/(1 + it \cdot 0{,}1)$, em que it representa o índice da iteração, e regularização $L2$ com parâmetro $\alpha = 10^{-5}$ aplicada no ajuste dos gradientes para aumentar a estabilidade numérica. Além disso, foi incorporado um critério de parada que interrompe o processo iterativo quando a variação relativa entre tensores métricos consecutivos é menor que 10^{-6} ou quando a variação relativa da norma dos traços do tensor de Ricci entre duas iterações sucessivas é inferior ao mesmo limiar, indicando que o método atingiu a convergência. O número de vizinhos foi ajustado individualmente, adotando-se $k = 25$ para o parabolóide, $k = 15$ para a esfera e $k = 50$ para a sela de macaco.

Na adaptação com redes neurais, foi utilizado o *TensorFlow*[10], biblioteca amplamente popular para o desenvolvimento e treinamento de modelos de aprendizado de máquina. Como visto na seção 6.2, a primeira rede foi feita para capturar a parametrização de f por meio de

[7] https://numpy.org/
[8] https://matplotlib.org/
[9] https://scikit-learn.org/stable/
[10] https://www.tensorflow.org/?hl=pt-br

uma regressão. A arquitetura adotada consistiu em uma rede do tipo *Multilayer Perceptron* (4.4.3), totalmente conectada com 4 camadas ocultas. Como função de ativação, utilizou-se a GELU e ELU, escolhidas por serem diferenciáveis. Para evitar sobreajuste, foram incorporadas estratégias de regularização, incluindo *dropout* (técnica de regularização que descarta aleatoriamente neurônios durante o treinamento, introduzida por (SRIVASTAVA et al., 2014) para reduzir *overfitting*[11] em redes neurais) com taxa de 10%.

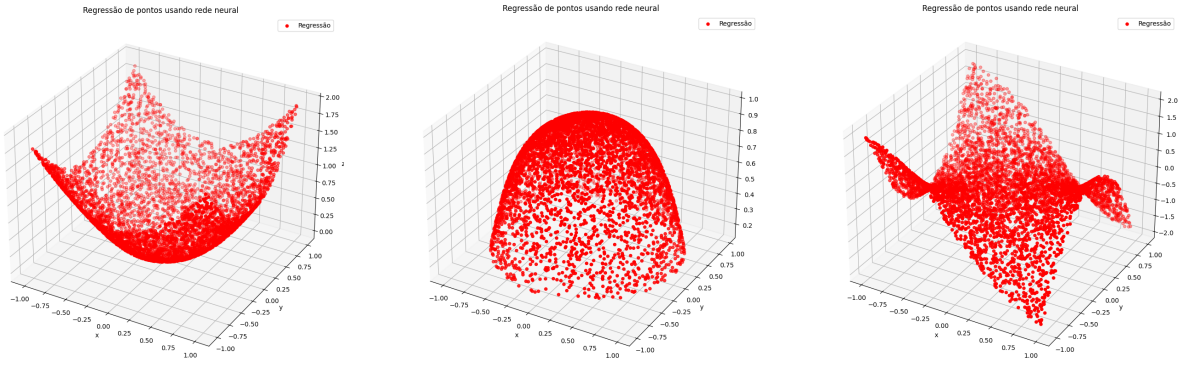
Essa arquitetura foi obtida usando o *keras tuner*[12] que utiliza inferência bayesiana para estimar a melhor arquitetura dentre os conjuntos de parâmetros fornecidos.

Tabela 4: Arquitetura da primeira rede neural

Camada	**Nº de neurônios**	**Função de ativação**	**Dropout**
Entrada	2	linear	0.0
Primeira camada	32	elu	0.0
Segunda camada	64	gelu	0.0
Terceira camada	32	elu	0.1
Quarta camada	16	elu	0.0
Saída	3	linear	0.0

O treinamento foi conduzido utilizando o otimizador ADAM com taxa de aprendizado inicial de 0.001. A função de perda adotada foi o erro quadrático médio, e o modelo foi treinado por 500 épocas, com tamanho de *batch*[13] igual a 64, empregando o critério de parada *early stopping*[14] (com o parâmetro de paciência de 50) para evitar degradação da performance. A validação cruzada foi realizada sobre 30% dos dados, permitindo avaliar a generalização do modelo para novas amostras. O resultado da regressão das superfícies pode ser visuallizado na Figura 26 e as suas respectivas funções de custo (*loss*) na Figura 27.

Figura 26: Regressão obtida através da rede neural.



(a) Parabolóide (b) Esfera (c) Sela de Macaco

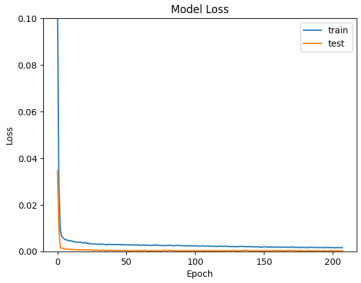
[11]Overfitting (ou sobreajuste) ocorre quando um modelo de aprendizado de máquina se ajusta muito bem aos dados de treinamento, mas falha em generalizar para dados novos.

[12]`https://keras.io/keras_tuner/`

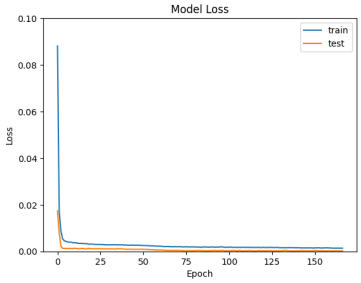
[13]Refere-se ao número de amostras processadas em uma iteração durante o treinamento de modelos de *machine learning*.

[14]É um critério de parada antecipada no treinamento de modelos de que interrompe o processo quando o desempenho em um conjunto de validação para de melhorar, evitando *overfitting* e otimizando o tempo computacional.

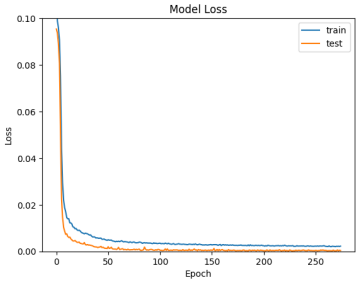
Figura 27: Gráfico da *loss* rede neural.



(a) Parabolóide



(b) Esfera



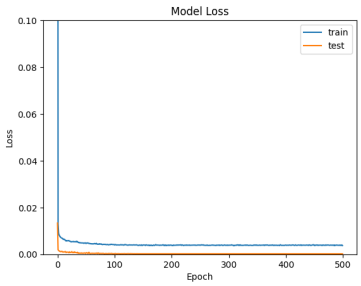
(c) Sela de Macaco

Para a segunda rede, responsável por realizar a regressão das derivadas e, assim, permitir as derivadas automáticas de segunda ordem conforme descrito na subseção 6.2, o treinamento foi conduzido utilizando o otimizador ADAM, com taxa de aprendizado inicial de 0.001. Essa taxa foi ajustada dinamicamente por meio do método *ReduceLROnPlateau*, disponível no *TensorFlow/Keras*, que monitora uma métrica de interesse — neste caso, a perda de validação — e, caso não haja melhora após 20 épocas consecutivas, reduz a taxa de aprendizado multiplicando-a por um fator predefinido (neste caso, 0.5). Esse mecanismo busca melhorar a convergência permitindo que o otimizador realize ajustes mais finos à medida que se aproxima de um mínimo. A função de perda adotada foi o erro quadrático médio, e o modelo foi treinado por 500 épocas, com tamanho de *batch* igual a 64, sem critério de parada antecipada. A validação cruzada foi realizada sobre 30% dos dados. A arquitetura dessa rede e a curva de custo serão apresentadas na Tabela 5 e Figura 28, respectivamente.

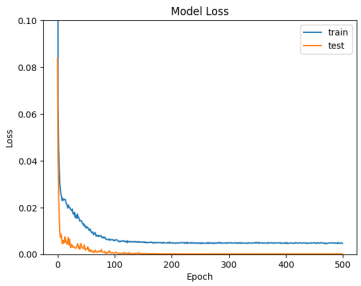
Tabela 5: Arquitetura da segunda rede neural

Camada	**Nº de neurônios**	**Função de ativação**	**Dropout**
Entrada	2	linear	0.0
Primeira camada	128	gelu	0.2
Segunda camada	128	gelu	0.1
Terceira camada	64	gelu	0.1
Quarta camada	32	gelu	0.0
Quinta camada	16	gelu	0.0
Saída	3	linear	0.0

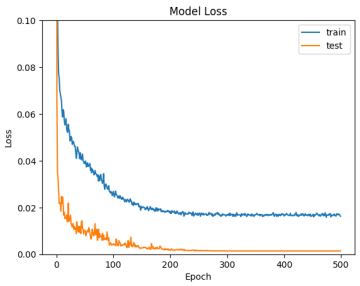
Figura 28: Gráfico da *loss* da segunda rede neural.



(a) Parabolóide



(b) Esfera



(c) Sela de Macaco

Note que os gráficos das funções de custo do parabolóide e da esfera apresentam um comportamento estável. Já para a sela de macaco, é mostrado um comportamento com um custo mais elevado, porém estável.

Para a comparação com as curvaturas discretas de Forman e Ollivier, utilizaram-se as bibliotecas *ot*, *networkx*[15] e *scipy*[16], respectivamente para o cálculo de distâncias de transporte ótimo, modelagem e análise de grafos. O tensor de curvatura de Ricci foi estimado conforme os procedimentos apresentados na Seção 6. Após estimá-los, a curvatura escalar pode ser obtida a partir desse tensor com base na relação estabelecida no Lema 3, determinou-se a curvatura gaussiana discreta, utilizada para a comparação com os valores analíticos exatos. Os resultados para cada superfície são apresentados nas subseções seguintes.

Para ilustrar a curvatura gaussiana exata, utilizamos as expressões analíticas para cada uma das superfícies, isto é para o paraboloide, calota superior da esfera e sela de macaco de parametrizações respectivas: $P(u,v) = (u, v, u^2 + v^2)$, $E(u,v) = (u, v, \sqrt{1-u^2-v^2})$ e $S(u,v) = (u, v, u^3 - 3uv^2)$, tem-se a curvatura gaussiana dada por

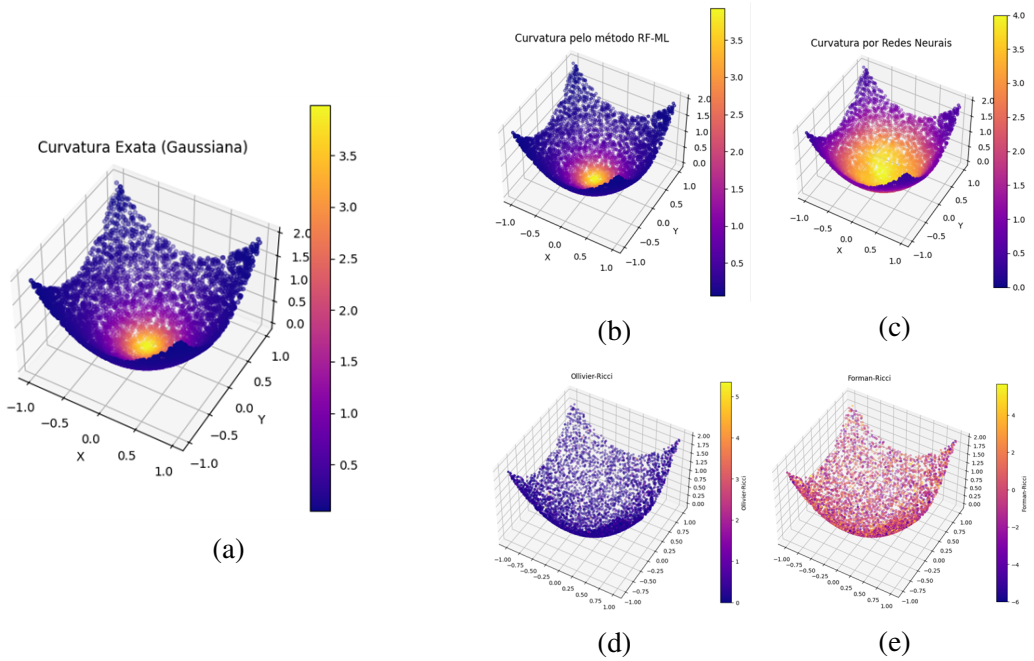
$$K_P(u,v) = \frac{4}{(1+4u^2+4v^2)^2} \tag{15}$$

$$K_E(u,v) = 1 \tag{16}$$

$$K_S(u,v) = \frac{-36\,(x^2+y^2)}{\left(1+9(x^2+y^2)^2\right)^2}. \tag{17}$$

7.1 Parabolóide

Figura 29: Estimativas de curvatura de Ricci para o parabolóide.



(a) (b) (c) (d) (e)

Fonte: Elaborado pela autora.

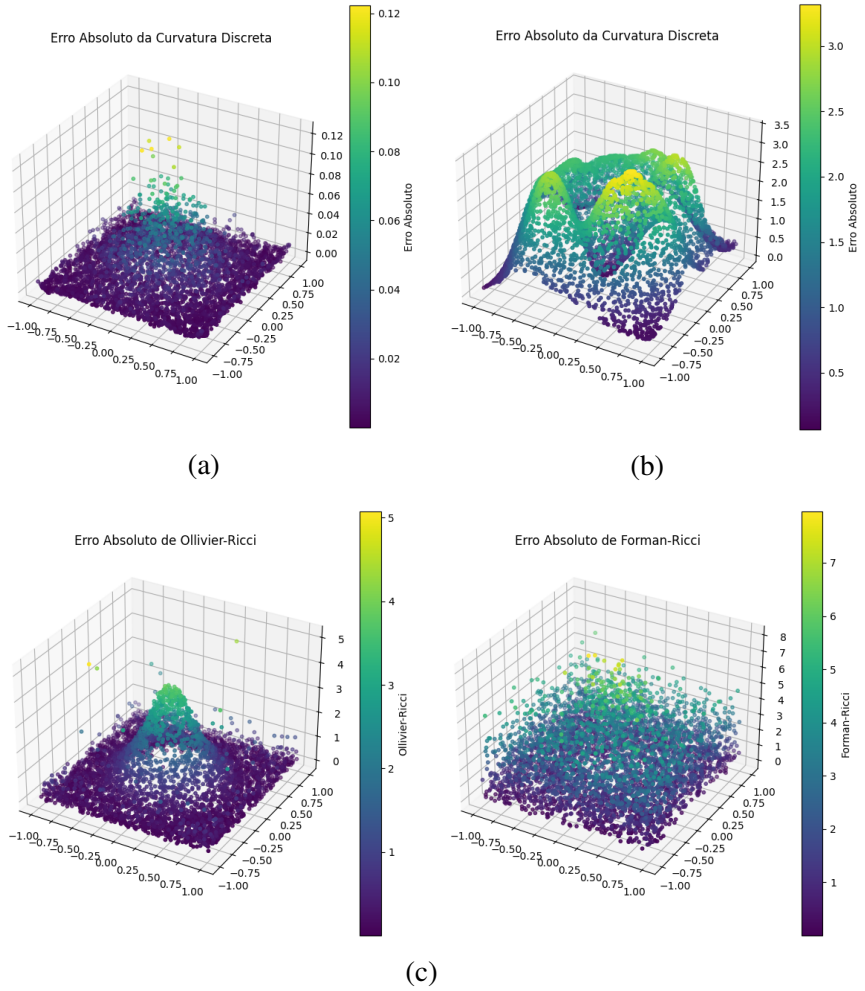
[15] https://networkx.org/

[16] https://docs.scipy.org/doc/scipy/index.html

Os resultados apresentados na Figura 29 possibilitam comparar a estimativa da curvatura gaussiana (através do tensor de Ricci) para o paraboloide, através dos diferentes métodos. Veja que, o método RF-ML (33b) entre todos, demonstra uma melhor aproximação em relação a curvatura exata (33a), enquanto as redes neurais (29c) apresenta uma variação mais abrupta, o que pode sugerir algum problema no modelo. Entretanto, o método de Ollivier-Ricci (33d) mostra uma aproximação grosseira com relação ao resultado exato, e o de Formann-Ricci (33e) demonstra ser o menos preciso.

Vejamos agora, as superfícies que representam o erro absoluto de cada um em comparação com a curvatura exata na Figura 30.

Figura 30: Superfícies de erros absolutos de curvatura de Ricci para o paraboloide.



(a) (b)

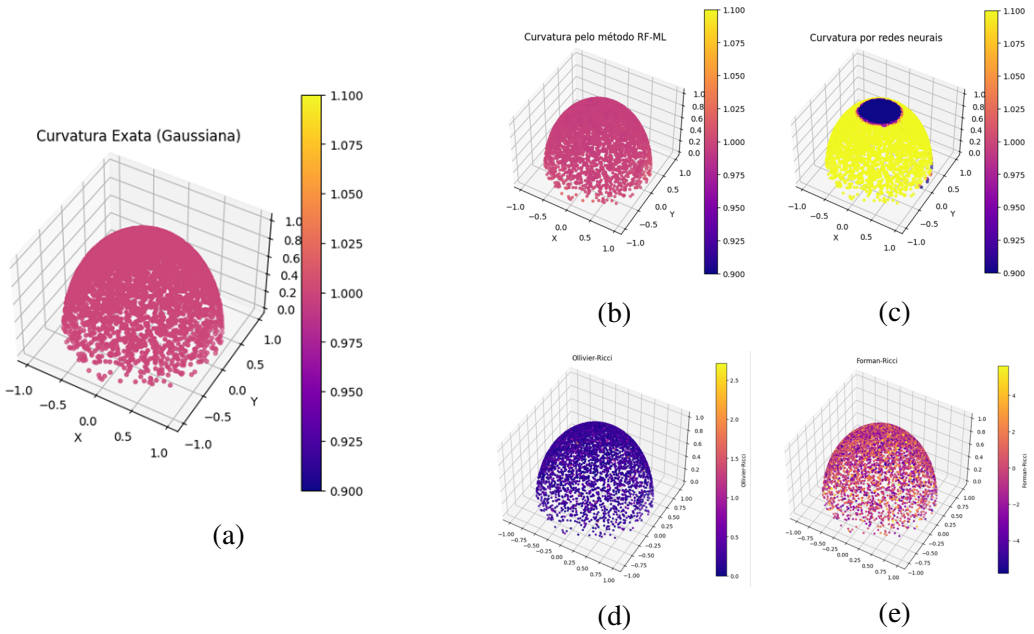
(c)

Fonte: Elaborado pela autora.

Na Figura 30 é ilustrado o erro absoluto entre a curvatura gaussiana exata e, respectivamente, as curvaturas estimadas pelo método RF-ML, por redes neurais e por Ollivier-Ricci e Forman-Ricci. Note que, o erro absoluto em (a) varia até 0.12, e tem pontos concentrados bem próximos a 0, com poucos picos de erro. Já em (b), temos erros mais distribuídos e chegando até, aproximadamente 3.5, o que é relativamente alto, mas que quando é comparado com (c), contínua sendo um pouco melhor. O que vai de acordo com a análise que fizemos da Figura 30.

7.2 Calota superior da esfera

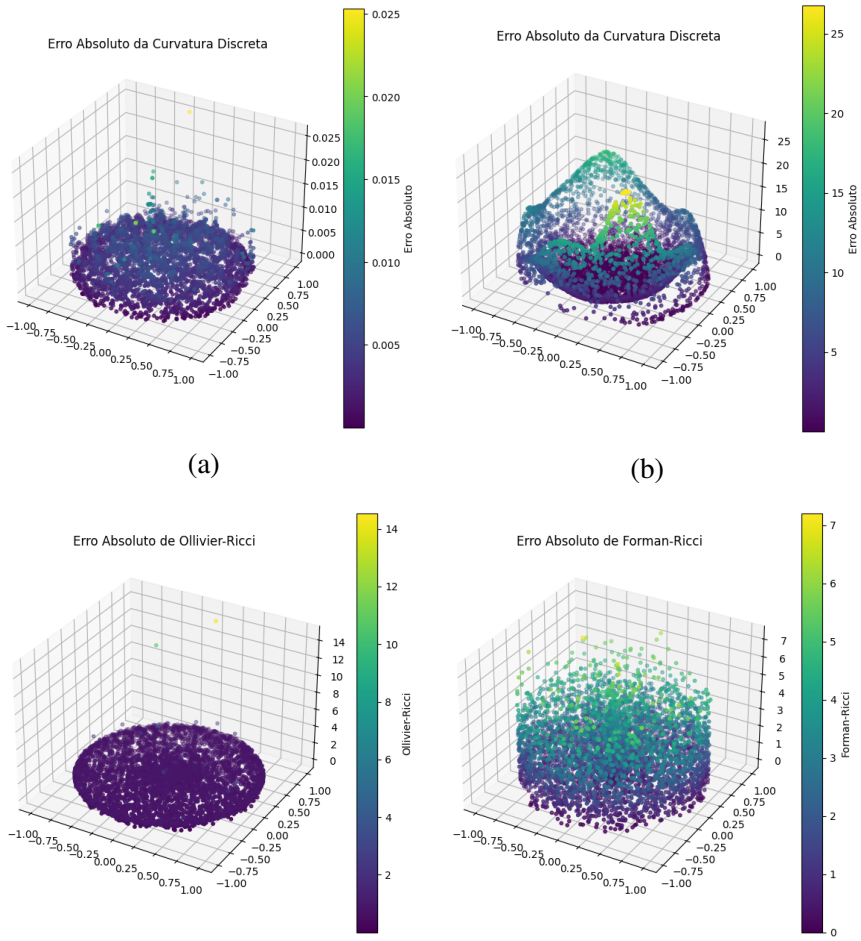
Figura 31: Estimativas de curvatura de Ricci para a calota superior da esfera.



Fonte: Elaborado pela autora.

A Figura 31 ilustra o comparativo entre a estimativa da curvatura gaussiana exata (a) (através do tensor de Ricci) para a calota superior da esfera, através dos métodos, respectivamente: (b) RF-ML, (c) Redes Neurais, (d) Ollivier-Ricci e (e) Formann-Ricci. Note que, o método (b) tem uma diferença visual mínima em relação a (a), o que indica uma boa precisão. Em (c), vemos uma grande concentração de valores no limite superior, próximo a 1.1, o que sugere uma perca de detalhes quando comparado aos outros, apresentando visualmente, o pior desempenho aqui. No caso de (d), vemos que não captura a variação suave da curvatura original e a coloração está concentrada entre a faixa de valores de 0 a 1, o que também não sugere uma boa aproxima;ção. Por fim, em (e), assim como no caso do parabolóide, há bastante ruído e dispersão e é o que menos captura, visualmente a curvatura.

Figura 32: Erros absolutos de curvatura de Ricci para a calota superior da esfera.



(a) (b)

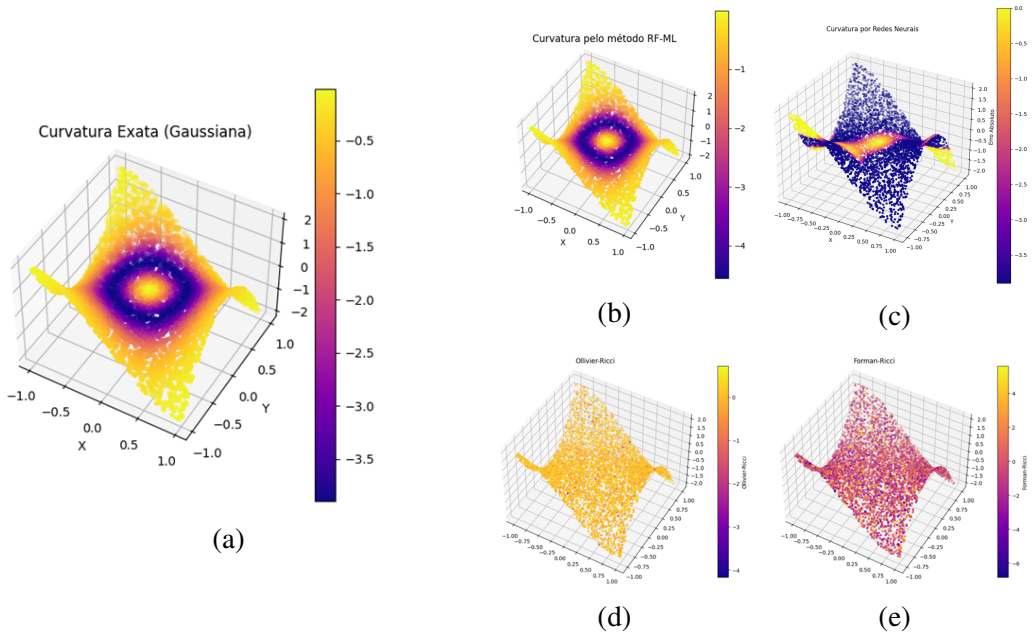
(c)

Fonte: Elaborado pela autora.

Agora, na Figura 32 é apresentado erro absoluto entre a curvatura gaussiana exata e, respectivamente, as curvaturas estimadas pelo método RF-ML, por redes neurais e por Ollivier-Ricci e Forman-Ricci. Em (a) vemos que o erro é muito baixo, indo no máximo até 0.025, o que remete a uma alta precisão. Na figura (b), temos um erro absoluto máximo muito alto, chegando a aproximadamente 25, além disso, é possível notar uma distribuição mais espalhada e com altos picos. Em (c), na primeira figura temos um erro máximo de aproximadamente 14, porém, a distribuição está bem concentrada entre 0 e 2, o que sugere um desempenho mediano. Já na segunda figura, temos um erro máximo de aproximadamente 7, porém novamente com uma distribuição muito espalhada. Nesse caso, o método RF-ML configura o melhor desempenho e o baseado em redes neurais é que possui menor eficiência.

7.3 Sela de Macaco

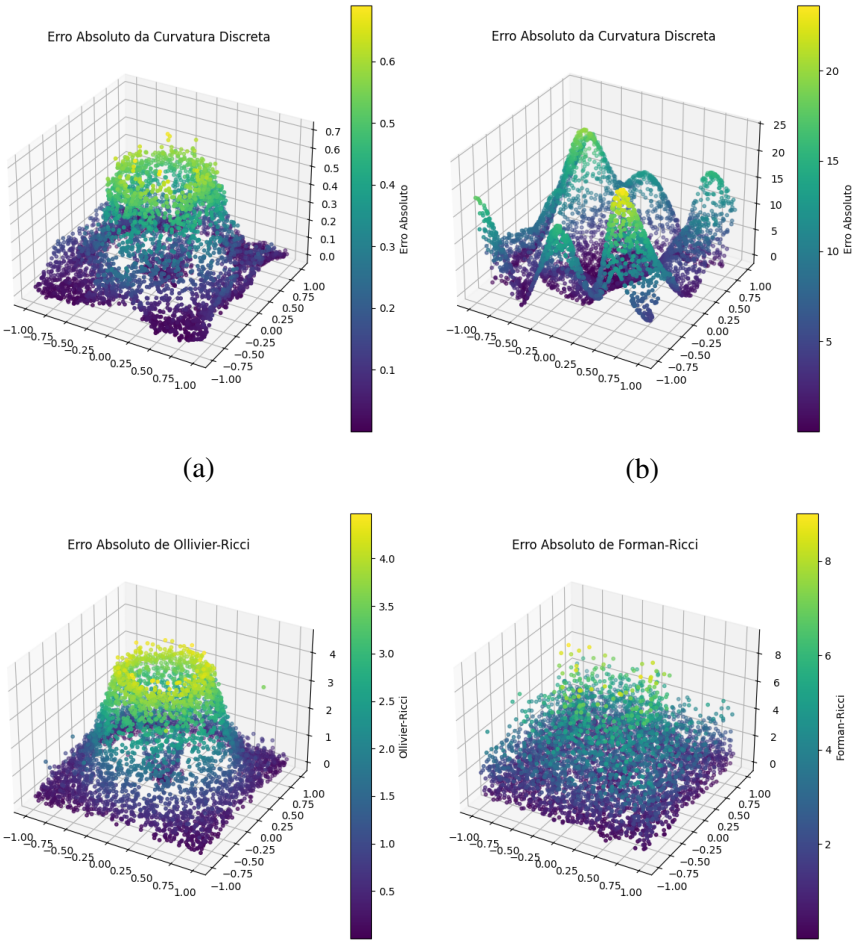
Figura 33: Estimativas de curvatura de Ricci para a sela do macaco.



Fonte: Elaborado pela autora.

Na Figura 33 é apresentado um comparativo entre a estimativa exata da curvatura gaussiana (a), obtida a partir do tensor de Ricci para a superfície sela de macaco, e as estimativas fornecidas, respectivamente, pelos métodos: (b) RF-ML, (c) Redes Neurais, (d) Ollivier–Ricci e (e) Forman–Ricci. Observa-se que, no caso do método RF-ML (b), a simetria da distribuição é preservada e as escalas, embora apresentem alguns desvios, permanecem próximas às do caso original. Já em (c), o ajuste é bastante grosseiro, não preservando nem a forma visual nem a escala. Para os métodos Ollivier–Ricci (d) e Forman–Ricci (e), nota-se que o ajuste também se afasta do formato exato; entretanto, o método (d) apresenta o segundo melhor desempenho entre os avaliados.

Figura 34: Erros absolutos de curvatura de Ricci para sela do macaco.



(a) (b)

(c)

Fonte: Elaborado pela autora.

Na Figura 34 é apresentada a superfície de erro absoluto entre a curvatura gaussiana exata e, respectivamente, as curvaturas estimadas pelos métodos RF-ML, Redes Neurais, Ollivier–Ricci e Forman–Ricci. Ao analisar e comparar cada caso, observa-se que (a) apresenta a menor escala de erro em relação aos demais métodos, embora ainda seja relativamente alta se comparada aos erros absolutos obtidos para as superfícies anteriores. Nota-se que (b) exibe o maior pico de erro, enquanto a primeira figura de (c) apresenta o segundo menor pico, ficando atrás apenas de (a). Já a segunda figura de (c), assim como nos casos anteriores, mostra uma distribuição de erros mais ruidosa.

7.4 Erros Relativos

Os erros relativos para Ollivier-Ricci e Formann-Ricci estão ilustrados na tabela 8 e 9, respectivamente. O número de vizinhos utilizados para o primeiro, foram $k = 3$ para o paraboloide), $k = 4$ para a calota superior da esfera e $k = 3$ na sela de macaco. Já para Formann-Ricci foi utilizado $k = 5$ em todas as superfícies.

Como métrica de avaliação, adotamos o erro relativo entre os valores de curvatura estimados pelos métodos e os valores analíticos exatos.

Tabela 6: Erro Relativo entre a Curvatura Estimada pelo método RF-ML de (LI; LU, 2019) e a Curvatura Gaussiana Exata

	Parabolóide	**Esfera**	**Sela**
Erro relativo mínimo	1.58031×10^{-6}	3.09967×10^{-7}	1.56769×10^{-5}
Erro relativo máximo	2.77484×10^{-1}	2.5281×10^{-2}	1.00453×10^{0}
Mediana do erros relativos	1.29441×10^{-2}	3.77734×10^{-3}	1.16065×10^{-1}
Média dos erros relativos	1.74507×10^{-2}	3.70748×10^{-3}	1.43163×10^{-1}

Tabela 7: Erro Relativo entre a Curvatura Estimada usando Redes Neurais e a Curvatura Gaussiana Exata

	Parabolóide	**Esfera**	**Sela**
Erro relativo mínimo	3.27956×10^{-4}	1.21445×10^{-4}	3.27241×10^{-5}
Erro relativo máximo	9.95745×10^{0}	2.67790×10^{1}	3.12524×10^{1}
Mediana do erros relativos	5.03236×10^{0}	2.36302×10^{0}	9.17189×10^{-1}
Média dos erros relativos	4.50773×10^{0}	4.35647×10^{0}	2.37428×10^{0}

Tabela 8: Erro Relativo entre a Curvatura Estimada usando Ollivier-Ricci e a Curvatura Gaussiana Exata

	Parabolóide	**Esfera**	**Sela**
Erro relativo mínimo	7.81232×10^{-4}	1.62036×10^{-3}	2.20817×10^{-3}
Erro relativo máximo	2.90098×10^{0}	1.71140×10^{1}	1.78510×10^{1}
Mediana do erros relativos	7.69012×10^{-1}	7.95890×10^{-1}	1.0334×10^{0}
Média dos erros relativos	7.28060×10^{-1}	7.60145×10^{-1}	1.17288×10^{0}

Tabela 9: Erro Relativo entre a Curvatura Estimada usando Formann-Ricci e a Curvatura Gaussiana Exata

	Parabolóide	**Esfera**	**Sela**
Erro relativo mínimo	0.00000×10^{0}	0.00000×10^{0}	2.11054×10^{-4}
Erro relativo máximo	7.20000×10^{0}	6.80000×10^{0}	1.55846×10^{2}
Mediana do erros relativos	1.71429×10^{0}	1.66666×10^{0}	1.52045×10^{0}
Média dos erros relativos	1.92213×10^{0}	1.89369×10^{0}	3.43200×10^{0}

8 Conclusão e Trabalhos Futuros

Os resultados demonstram que o método RF-ML de (LI; LU, 2019) apresenta desempenho satisfatório para superfícies de curvatura positiva (parabolóide) e constante (esfera), com erros relativamente baixos, tanto em termos de média quanto mediana dos erros relativos (Tabela 6). Entretanto, no caso da superfície com curvatura negativa (sela), observamos uma pequena discrepância nos valores calculados, com erros mais elevados em relação as outras superfícies. Este comportamento está alinhado com as limitações reportadas pelos autores, particularmente no que diz respeito a regiões com curvatura negativa.

Ao comparar com as demais abordagens, nota-se que:

- **Redes neurais** (Tabela 7) apresentaram desempenho inferior nas três superfícies, com erros médios acima de 1,0 em todos os casos. Isso sugere que a rede, mesmo treinada para regressão das derivadas, apresentou dificuldades possivelmente devido ao acúmulo de erros no cálculo das derivadas de segunda ordem;

- **Ollivier-Ricci** (Tabela 8) apresentou resultados mais consistentes, com erros médios inferiores a 1,0 para o parabolóide e a esfera, mas com degradação no caso da sela. Embora o método tenha se mostrado positivo em regiões de curvatura positiva, ainda sofre com instabilidades em curvaturas negativas;

- **Formann-Ricci** (Tabela 9) foi o que mais apresentou resultados irregulares. Apesar de em alguns casos atingir o valor da curvatura muito próxima da gaussiana, em sua maioria, o método apresenta erro elevado.

Em síntese, o método RF-ML permanece como a solução mais estável para as superfícies testadas, embora apresente algumas limitações em superfícies de curvatura negativa. A abordagem baseada em redes neurais necessita de ajustes significativos para alcançar um bom desempenho, enquanto as métricas de curvatura de grafos (Ollivier-Ricci e Formann-Ricci) ainda apresentam lacunas para uso quantitativo preciso. Levando em consideração tudo que foi exposto, como perspectivas de continuidade em possíveis trabalhos futuros, destacam-se:

- Explorar arquiteturas mais profundas, capazes de capturar melhor padrões locais de curvatura;

- Explorar outros tipos de redes neurais, como as convolucionais, PINN's e entre outros;

- Aplicar sobre dados de imagens para analisar o uso do Tensor de Ricci como um classificador;

- Buscar métodos mais precisos e menos custosos para estimativa de derivadas de redes neurais.

Com essas melhorias, espera-se não apenas reduzir os erros observados, mas também aumentar a eficácia das estimativas de curvatura em diferentes tipos de dados, e incluindo casos de regiões de variedades com curvatura negativa, os quais se mostraram mais desafiadores neste estudo.

Referências Bibliográficas

ABADI, Martín et al. {TensorFlow}: a system for {Large-Scale} machine learning. In: 12TH USENIX symposium on operating systems design and implementation (OSDI 16). [S.l.: s.n.], 2016. P. 265–283.

ANTON, Howard; RORRES, Chris. **Álgebra Linear com Aplicações-10**. [S.l.]: Bookman Editora, 2012.

BAPTISTA, Anthony et al. Deep learning as Ricci flow. **Scientific Reports**, Nature Publishing Group UK London, v. 14, n. 1, p. 23383, 2024.

BAYDIN, Atilim Gunes et al. Automatic differentiation in machine learning: a survey. **Journal of machine learning research**, v. 18, n. 153, p. 1–43, 2018.

BISHOP, Christopher M. **Pattern recognition and machine learning**. [S.l.]: Springer, 2006. v. 4.

CARMO, Manfredo P. do. **Geometria Riemanniana**. 5º ed. Rio de Janeiro: Instituto de Matemática Pura e Aplicada (IMPA), 2015.

FASINA, Oluwadamilola et al. Neural FIM for learning Fisher information metrics from point cloud data. In: PMLR. INTERNATIONAL Conference on Machine Learning. [S.l.: s.n.], 2023. P. 9814–9826.

FORMAN, R. Bochner's Method for Cell Complexes and Combinatorial Ricci Curvature. **Discrete & Compututational Geometry**, v. 29, n. 1, p. 323–374, 2003.

GERON, Aurelien. **Mãos à obra: aprendizado de maquina com Scikit-Learn, Keras & Tensorflow**. [S.l.]: Alta Books, 2021.

HARRIS, Charles R et al. Array programming with NumPy. **nature**, Nature Publishing Group UK London, v. 585, n. 7825, p. 357–362, 2020.

HUNTER, John D. Matplotlib: A 2D graphics environment. **Computing in science & engineering**, IEEE Computer Society, v. 9, n. 03, p. 90–95, 2007.

KAUL, Piyush; LALL, Brejesh. Riemannian curvature of deep neural networks. **IEEE transactions on neural networks and learning systems**, IEEE, v. 31, n. 4, p. 1410–1416, 2019.

LEE, John M. **Riemannian manifolds: an introduction to curvature**. [S.l.]: Springer Science & Business Media, 2006. v. 176.

_____. **Introduction to Smooth Manifolds**. 2º ed. Seattle: Springer International Publishing, 2018.

_____. **Smooth manifolds**. [S.l.]: Springer, 2003.

LEE, Yonghyeon et al. A statistical manifold framework for point cloud data. In: PMLR. INTERNATIONAL Conference on Machine Learning. [S.l.: s.n.], 2022. P. 12378–12402.

LI, Yangyang; LU, Ruqian. Applying Ricci flow to high dimensional manifold learning. **Science China Information Sciences**, Springer, v. 62, n. 9, p. 192101, 2019.

MCCULLOCH, Warren S; PITTS, Walter. A logical calculus of the ideas immanent in nervous activity. **The bulletin of mathematical biophysics**, Springer, v. 5, n. 4, p. 115–133, 1943.

NI, Chien-Chun et al. Community detection on networks with Ricci flow. **Scientific reports**, Nature Publishing Group UK London, v. 9, n. 1, p. 9984, 2019.

O'MALLEY, Tom et al. **KerasTuner**. [S.l.], 2019. Acesso em: 2023.

PEDREGOSA, Fabian et al. Scikit-learn: Machine learning in Python. **the Journal of machine Learning research**, JMLR. org, v. 12, p. 2825–2830, 2011.

ROSENBLATT, Frank. The perceptron: a probabilistic model for information storage and organization in the brain. **Psychological review**, American Psychological Association, v. 65, n. 6, p. 386, 1958.

SRIVASTAVA, Nitish et al. Dropout: a simple way to prevent neural networks from overfitting. **The journal of machine learning research**, JMLR. org, v. 15, n. 1, p. 1929–1958, 2014.

Apêndice A: Conceitos de Variedades Diferenciáveis

Definição 15. *Seja X um conjunto. Uma **topologia** em X é uma coleção τ de subconjuntos de X (chamados **abertos**) que satisfaz os seguintes axiomas:*

i) O conjunto vazio e o próprio X são abertos:

$$\emptyset \in \tau \quad e \quad X \in \tau.$$

ii) União arbitrária de abertos é aberta: Se $\{U_i\}_{i\in I} \subseteq \tau$, então

$$\bigcup_{i\in I} U_i \in \tau.$$

iii) Interseção finita de abertos é aberta: Se $U_1, \ldots, U_n \in \tau$, então

$$\bigcap_{k=1}^{n} U_k \in \tau.$$

*O par (X, τ) é chamado de **espaço topológico**.*

Definição 16. *Uma **variedade topológica** M de dimensão n é um espaço topológico que satisfaz as seguintes propriedades:*

*i) **Hausdorff**: Para quaisquer dois pontos distintos $p, q \in M$, existem abertos $p \in U$ e $q \in V$ disjuntos ($U \cap V = \emptyset$).*

*ii) **Segundo-contável**: Existe uma base contável para a topologia.*

*iii) **Localmente Euclidiana**: Para todo $p \in M$, existe uma vizinhança aberta $U \subseteq M$ contendo p e um homeomorfismo $\phi : U \to \widetilde{U}$, onde $\widetilde{U}$ é um aberto de $\mathbb{R}^n$.*

Definição 17. *Seja $\mathcal{M}$ uma variedade diferenciável. Uma família de abertos $V_\alpha \subset \mathcal{M}$ com*

$$\bigcup_{\alpha} V_\alpha = \mathcal{M}$$

é localmente finita se todo ponto $p \in \mathcal{M}$ possui uma vizinhança U tal que $U \cap V_\alpha \neq \phi$ apenas para um número finito de índices. O suporte de uma função $f : \mathcal{M} \to \mathbb{R}$ é o fecho do conjunto dos pontos onde f é diferente de zero.

Definição 18 (Partição da unidade)**.** *Dizemos que uma família $\{f_\alpha\}$ de funções diferenciáveis $f_\alpha : \mathcal{M} \to \mathbb{R}$ é uma **partição diferenciável da unidade** se:*

i) Para todo α, $f_\alpha \geq 0$ e o suporte de f_α está contido em uma vizinhança coordenada $V_\alpha = x_\alpha(U_\alpha)$ de uma estrutura diferenciável $\{(U_\beta, x_\beta)\}$ de $\mathcal{M}$.

ii) A família $\{V_\alpha\}$ é localmente finita.

iii) $\sum_\alpha f_\alpha(p) = 1$, para todo $p \in \mathcal{M}$ (esta condição faz sentido, pois em cada p, $f_\alpha(p) \neq 0$ para apenas um número finito de índices).

Costuma-se dizer que a partição $\{f_\alpha\}$ da unidade está subordinada à cobertura $\{V_\alpha\}$.

Definição 19. *Sejam $F : \mathcal{M} \to N$ uma aplicação suave entre variedades diferenciáveis e $dF_p : T_p\mathcal{M} \to T_{F(p)}N$ a derivada em $p \in \mathcal{M}$. Dizemos que F é:*

*(i) uma **imersão** se dF_p é injetiva para todo $p \in \mathcal{M}$.*

*(ii) um **mergulho suave** se F é uma imersão e um homeomorfismo sobre a imagem $F(\mathcal{M}) \subset N$ com a topologia induzida.*

Seja $\mathcal{M}$ uma variedade suave com ou sem fronteira. Uma **subvariedade imersa** de $\mathcal{M}$ é um subconjunto $S \subset \mathcal{M}$ dotado com uma topologia com respeito a qual ela é uma variedade topológica (sem fronteira), e uma estrutura suave com respeito a qual a aplicação inclusão $i : S \to \mathcal{M}$ é uma imersão suave.

Pela definição toda variedade mergulhada é uma subvariedade imersa. Desta forma, por convenção o termo **subvariedade suave** significa uma subvariedade imersa.

Proposição 9 (Imagens de Imersões como Subvariedades)**.** *Sejam $\mathcal{M}$ uma variedade suave com ou sem fronteira, N uma variedade suave, $F : N \to \mathcal{M}$ uma imersão suave injetiva e $S = F(N)$. Então S possui uma única topologia e estrutura suave tal que ela é uma subvariedade suave de $\mathcal{M}$ e tal que $F : N \to S$ é um difeomorfismo sobre sua imagem.*

Demonstração. Ver (LEE, John M., 2003), página 109. □

Teorema 5 (Subvariedades Imersas são Localmente Mergulhadas)**.** *Se $\mathcal{M}$ é uma variedade suave com ou sem fronteira, e $S \subset M$ é uma subvariedade imersa, então para cada $p \in S$ existe uma vizinhança U de p em S que é uma subvariedade mergulhada de $\mathcal{M}$.*

Demonstração. Ver (LEE, John M., 2003), página 87 e 109. □

Definição 20. *Seja $f : \mathcal{M} \to N$ uma imersão. Se N tem uma estrutura Riemanniana, f induz uma estrutura Riemanniana em $\mathcal{M}$ por*

$$\langle u, v\rangle_p = \langle df_p(u), df_p(v)\rangle_{f(p)}, \quad u, v \in T_p\mathcal{M}.$$

A métrica de $\mathcal{M}$ é chamada de métrica induzida (ou métrica pullback por f, e f é uma imersão isométrica.