Iniciação Científica

Relatório Final

Aprendizado profundo e geometria hiperbólica

Universidade de São Paulo
Instituto de Ciências Matemáticas e Computação

Aluno: Lucas Giraldi Almeida Coimbra
Orientador: Carlos Henrique Grossi Ferreira

Junho de 2023

São Carlos

Conteúdo

1 Introdução **2**

2 Geometria riemanniana **2**
2.1 Variedades e métricas riemannianas . . . 2
2.2 Conexões e derivada covariante . . . 3
2.3 Jacobiana e teorema da função inversa . . . 4
2.4 Geodésicas e transporte paralelo . . . 5
2.5 Mapa exponencial e mapa logarítmico . . . 6
2.6 Curvatura por via de tensores . . . 6
2.7 Conceitos métricos . . . 8

3 Modelos para a geometria hiperbólica **8**
3.1 O semi-espaço de Poincaré . . . 8
3.2 O hiperboloide de Lorentz . . . 9
3.3 O n-disco de Poincaré . . . 9
3.4 O n-disco de Beltrami-Klein . . . 9
3.5 O modelo do hemisfério . . . 10
3.6 Isometrias entre os modelos . . . 10
3.7 Generalizando operações euclidianas . . . 10

4 Redes neurais **12**
4.1 `PyTorch` e tensores . . . 12
4.2 Trabalhando com tensores . . . 15

1 Introdução

2 Geometria riemanniana

2.1 Variedades e métricas riemannianas

Uma *variedade topológica de dimensão n* é um espaço topológico M Hausdorff com base enumerável que é *localmente euclidiano de dimensão n*, isso é, para cada $p \in M$ existe um aberto U e um homeomorfismo $\phi\colon U \to V \subset\circ \mathbb{R}^n$. O par (U, ϕ) será comumente chamado de *carta sobre p*. Se (V, ψ) é uma outra carta em M tal que $U \cap V \neq \varnothing$, chamamos de *mapas de transição* as funções

$$\phi \circ \psi^{-1}\colon \psi(U \cap V) \to \mathbb{R}^n \quad \text{e} \quad \psi \circ \phi^{-1}\colon \phi(U \cap V) \to \mathbb{R}^n. \tag{1}$$

Se os mapas de transição forem suaves, diremos que (U, ϕ) e (V, ψ) são *compatíveis*. Uma *estrutura diferenciável* em M é uma cobertura de M por cartas que são duas a duas compatíveis. Dizemos que M é *suave* ou *diferenciável* se possuir uma estrutura diferenciável.

A partir de agora, toda carta estará em uma estrutura diferenciável previamente fixada, e portanto toda variedade será suave. Se $p \in M$ dizemos que $F\colon M \to N$ é *suave em p* se existirem (U, ϕ) carta sobre p e (V, ψ) carta sobre $F(p)$ tais que $\psi \circ F \circ \phi^{-1}$ é suave. A função F é *suave em $U \subset\circ M$* se for suave em todo ponto de U, e é apenas *suave* se for suave em todo ponto de M.

Uma *curva* em M é um mapa suave $c\colon I \to M$ onde I é um intervalo de $\mathbb{R}$. Se $p \in M$, definimos por C^∞_p como o conjunto dos mapas $f\colon U \subset\circ M \to \mathbb{R}$ suaves, onde U é uma vizinhança qualquer de p. Esse espaço é uma álgebra com as três operações:

- se $f\colon U \to \mathbb{R}$ e $g\colon V \to \mathbb{R}$, definimos $f + g\colon U \cap V \to \mathbb{R}$ por $(f + g)(p) = f(p) + g(p)$;
- se $f\colon U \to \mathbb{R}$ e $\lambda \in \mathbb{R}$, definimos $\lambda f\colon U \to \mathbb{R}$ por $(\lambda f)(p) = \lambda f(p)$;
- se $f\colon U \to \mathbb{R}$ e $g\colon V \to \mathbb{R}$, definimos $fg\colon U \cap V \to \mathbb{R}$ por $(fg)(p) = f(p)g(p)$.

Dada uma curva $c\colon]-\varepsilon, \varepsilon[\to M$, definimos $c'(0)$ como sendo um mapa $c'(0)\colon C^\infty_p \to \mathbb{R}$ dado por

$$c'(0)f = \left.\frac{d}{dt}\right|_{t=0} (f \circ c)(t). \tag{2}$$

Esse mapa é linear e satisfaz a *regra de Leibniz*, isso é,

$$c'(0)(fg) = f(c(0)) \cdot c'(0)g + c'(0)f \cdot g(c(0)). \tag{3}$$

Se $p \in M$, o *espaço tangente a M em $p \in M$* como o conjunto

$$T_pM = \{c'(0) \mid c\colon]-\varepsilon, \varepsilon[\to \mathbb{R} \text{ e } c(0) = p\}. \tag{4}$$

Se M tem dimensão n, então T_pM é um espaço vetorial de dimensão n. Seus elementos são chamados de *vetores tangentes*. Uma *métrica riemanniana* em M é a associação de um produto interno $\mathfrak{g}_p(-,-)$ em T_pM para cada $p \in M$. Mais do que isso, pedimos que essa associação seja suave. Entenderemos o que isso significa a seguir.

Um *campo vetorial* em M é uma associação X de um vetor $X_p \in T_pM$ para cada $p \in M$. Se $\phi = (x^1, \ldots, x^n)$ é uma carta sobre $p \in M$ e $r = (r^1, \ldots, r^n)$ são as coordenadas em $\mathbb{R}^n$, definimos as derivadas parciais de $f \in C^\infty_p$ por

$$\left.\frac{\partial f}{\partial x^i}\right|_p = \left.\frac{\partial}{\partial r^i}\right|_{\phi(p)} (f \circ \phi^{-1})(r). \tag{5}$$

Cada derivada parcial em p pode ser vista como um elemento de T_pM, afinal, se $e^1, \ldots, e^n$ é a base canônica de $\mathbb{R}^n$, então dadas as curvas $c^i(t) = te^i$ temos

$$\left.\frac{\partial}{\partial x^i}\right|_p = (\phi^{-1} \circ c^i)'(0). \tag{6}$$

Esses vetores tangentes formam uma base para T_pM.

Se (U, ϕ) é uma em M e X é um campo vetorial em M, então para cada $p \in M$ podemos escrever, de maneira única,

$$X_p = \sum_{k=1}^{n} a^i(p) \left.\frac{\partial}{\partial x^i}\right|_p . \tag{7}$$

Dizemos que o campo vetorial X é *suave* se existir uma cobertura de M por cartas tais que os mapas a^i são sempre suaves. Ao dizermos que a métrica riemanniana tem que ser suave, queremos dizer que, para quaisquer X, Y campos suaves em M, o mapa $p \mapsto \mathfrak{g}_p(X_p, Y_p)$ tem que ser suave. Uma *variedade riemanniana* é uma variedade suave equipada com uma métrica riemanniana.

2.2 Conexões e derivada covariante

Denotamos o conjunto de todos os campos suaves em M por $\mathfrak{X}(M)$. Se $M = \mathbb{R}^n$, vamos entender quem é a derivada direcional. Se $X = (v^1, \ldots, v^n) \in \mathbb{R}^n$ e X_p é o vetor tangente a p na direção X, então dada $f\colon \mathbb{R}^n \to \mathbb{R}$ definimos a *derivada direcional de f na direção X_p*

$$D_{X_p} f = \lim_{t\to 0} \frac{f(p + tX) - f(p)}{t} = \sum_{k=1}^{n} v^k \left.\frac{\partial f}{\partial x^i}\right|_p = X_p f. \tag{8}$$

Podemos então trocar f por um campo vetorial suave $Y = \sum b^i \partial/\partial x^i$ e obtermos a *derivada direcional de Y na direção X_p*

$$D_{X_p} Y = \sum_{k=1}^{n} D_{X_p} b^i \left.\frac{\partial}{\partial x^i}\right|_p . \tag{9}$$

Note que a derivada $D_{X_p}Y$ é um vetor tangente em p. Dessa forma, se X é um campo vetorial em $\mathbb{R}^n$ podemos definir $D_X Y$ como o campo vetorial que, em p, vale $D_{X_p}Y$. Esse mapa é a *derivada direcional de Y na direção X*.

Agora vamos generalizar a derivada direcional em $\mathbb{R}^n$ para uma variedade riemanniana qualquer. Uma *conexão afim* em M é um mapa

$$\begin{aligned} \nabla\colon \mathfrak{X}(M) \times \mathfrak{X}(M) &\to \mathfrak{X}(M) \\ (X, Y) &\mapsto \nabla_X Y \end{aligned}$$

que satisfaz as seguintes propriedades:

- se $C^\infty(M)$ é o conjunto dos mapas suaves $M \to \mathbb{R}$, então ∇ é $C^\infty(M)$-linear na primeira coordenada;
- ∇ satisfaz a regra de Leibniz na segunda coordenada, isso é, se $f \in C^\infty(M)$, então

$$\nabla_X(fY) = (Xf)Y + f\nabla_X Y, \tag{10}$$

  onde Xf é o mapa suave dado por $(Xf)(p) = X_p f$.

Conexões e métricas riemannianas não estão sempre conectadas. Porém, se M é uma variedade riemanniana e ∇ uma conexão afim em M, então podemos falar sobre alguns aspectos geométricos de ∇. Definimos o *tensor torção* de ∇ como sendo o mapa $T(X, Y) = \nabla_X Y - \nabla_Y X - [X, Y]$, onde $[X, Y]_p f = X_p(Yf) - Y_p(Xf)$ é o *bracket de Lie*. Do mesmo modo, definimos o *tensor curvatura* de ∇ como sendo o mapa $R(X, Y) = [\nabla_X, \nabla_Y] - \nabla_{[X,Y]}$, isso é, para um campo vetorial suave Z, temos

$$R(X, Y)Z = \nabla_X \nabla_Y Z - \nabla_Y \nabla_X Z - \nabla_{[X,Y]} Z. \tag{11}$$

Dizemos que uma conexão ∇ em uma variedade riemanniana M é *compatível com a métrica* se $Z\mathfrak{g}(X, Y) = \mathfrak{g}(\nabla_Z X, Y) + \mathfrak{g}(X, \nabla_Z Y)$. Uma *conexão de Levi-Civita* é uma conexão compatível com a métrica e que satisfaz $T(X, Y) = 0$ para todos X, Y campos suaves em M.

Proposição 2.1. *Toda variedade riemanniana possui uma, e apenas uma, conexão de Levi-Civita.*

Um campo vetorial *ao longo* de uma curva $c\colon I \to M$ é a associação V de um vetor $V(t) \in T_{c(t)}M$ para cada $t \in I$. Dizemos que V é *suave* se, para cada $f\colon M \to \mathbb{R}$, $(Vf)(t) = V(t)f$ é suave.

Se $c\colon I \to \mathbb{R}^n$ é uma curva e V é um campo ao longo de c, temos

$$V(t) = \sum_{k=1}^{n} v^i(t) \left.\frac{\partial}{\partial x^i}\right|_{c(t)}, \tag{12}$$

portanto podemos definir a *derivada de V com respeito a t* como sendo o campo

$$\frac{dV}{dt} = \sum_{k=1}^{n} \frac{dv^i}{dt}\frac{\partial}{\partial x^i}. \tag{13}$$

Essa derivada satisfaz algumas propriedades importantes:

- ela é linear com respeito a V, isso é, se $\lambda \in \mathbb{R}$ e U é outro campo ao longo de c, então

$$\frac{d(\lambda V + U)}{dt} = \lambda\frac{dV}{dt} + \frac{dU}{dt}; \tag{14}$$

- ela satisfaz a regra de Leibniz, isso é, se $f\colon I \to \mathbb{R}$ (lembrando aqui que I é o domínio de c) é suave, então

$$\frac{d(fV)}{dt} = \frac{df}{dt}V + f\frac{dV}{dt}; \tag{15}$$

- ela é compatível com a derivada direcional em $\mathbb{R}^n$, isso é, se V se estende para um campo $\tilde{V}$ em $\mathbb{R}^n$, então

$$\frac{dV}{dt} = D_{c'(t)}\tilde{V}. \tag{16}$$

Vamos agora generalizar o conceito da derivada de V para uma variedade M qualquer, utilizando de conexões afins. Se ∇ é uma conexão afim em M e $c\colon I \to \mathbb{R}$ é uma curva, então definimos uma *derivada covariante* como um operador D/dt que, para cada campo V ao londo de c associa um outro campo DV/dt ao longo de c. Pedimos que essa associação satisfaça as três propriedades que a derivada definida acima satisfaz:

- D/dt é linear, isso é, se V e U são campos ao longo de c e $\lambda \in \mathbb{R}$ então

$$\frac{D(\lambda V + U)}{dt} = \lambda\frac{DV}{dt} + \frac{DU}{dt}; \tag{17}$$

- D/dt satisfaz a regra de Leibniz, isso é, se $f\colon I \to \mathbb{R}$ é suave, então

$$\frac{D(fV)}{dt} = \frac{df}{dt}V + f\frac{DV}{dt}; \tag{18}$$

- D/dt é compatível com a conexão afim, isso é: se $\tilde{V}$ é um campo em M que estende V, então

$$\frac{DV}{dt} = \nabla_{c'(t)}V. \tag{19}$$

Definimos acima o que seria **uma** derivada covariante, mas acontece que, fixadas uma conexão e uma curva, sempre existe uma e apenas uma derivada covariante, portanto podemos falar **da** derivada covariante.

2.3 Jacobiana e teorema da função inversa

Se $F\colon M \to N$ é suave, então para todas as carta (U, ϕ) sobre p e (V, ψ) sobre $F(p)$ o mapa $\psi \circ F \circ \phi^{-1}$ é suave. Sabemos da teoria de variedades que as derivadas parciais $\partial/\partial\phi^i$ e $\partial/\partial\psi^j$ formam base para T_pM e $T_{F(p)}N$, respectivamente. Considere agora a transformação linear D_pF dada por

$$D_pF(v)f = v(f \circ F) \tag{20}$$

que manda vetores tangentes a p para vetores tangentes a $F(p)$. Na expressão acima, estamos apenas descrevendo como o vetor $D_pF(v)$ age em uma função $f\colon N \to \mathbb{R}$ suave. O mapa D_pF é chamado de *derivada de F em p.*

Podemos então considerar a matriz de D_pF conforme as bases $\partial/\partial\phi^i$ e $\partial/\partial\psi^j$. Se denotarmos por F^i o mapa $\psi^i \circ F$, então temos que

$$D_pF = \left[\left.\frac{\partial F^i}{\partial \phi^j}\right|_p \right]. \tag{21}$$

Note que ela coincide com a matriz jacobiana que conhecemos do cálculo. De fato, essa coincidência motiva uma nova versão do teorema da função inversa.

Teorema 2.2. *Se $F\colon M \to N$ é suave e $\dim M = \dim N$, então F é um difeomorfismo local em $p \in M$ se, e somente se, $\det D_pF \neq 0$.*

2.4 Geodésicas e transporte paralelo

Se $c\colon I \to M$ é uma curva, então dizemos que c é uma *geodésica* se a derivada covariante DT/dt do seu campo velocidade $T(t) = c'(t)$ é nula. Note que a existência de uma conexão, e portanto de uma derivada covariante, não depende da existência de uma métrica riemanniana. Porém, caso a variedade M possua uma métrica, vamos sempre assumir que a conexão considerada é a conexão de Levi-Civita em M.

Proposição 2.3. *Geodésicas em variedades riemannianas possuem velocidade constante, isso é, se $c\colon I \to M$ é uma geodésica, então $||c'(t)||$ é constante para cada $t \in I$.*

Seja M uma variedade suave com uma conexão ∇. Se $(U, x^1, \ldots, x^n)$ é uma carta em M, então temos os campos vetoriais $\partial_i = \partial/\partial x^i$. Sabemos que todo campo vetorial em U se escreve como combinação linear destes, e portanto temos

$$\nabla_{\partial_i}\partial_j = \sum_{k=1}^{n} \Gamma_{ij}^k \partial_k. \tag{22}$$

Os coeficientes Γ_{ij}^k são chamados de *símbolos de Christoffel de ∇ em $(U, x^1, \ldots, x^n)$*.

Sejam M uma variedade com uma conexão ∇, $(U, \phi) = (U, x^1, \ldots, x^n)$ uma carta em M e Γ_{ij}^k os seus símbolos de Christoffel. Note que, se $c\colon I \to M$ é uma curva e $y = \phi \circ c$, então temos

$$T = c'(t) = \sum_{k=1}^{n} \frac{dy^k}{dt}\partial_k. \tag{23}$$

Dessa maneira, segue que

$$\frac{DT}{dt} = \sum_{j=1}^{n} \frac{d^2y^j}{dt^2}\partial_j + \sum_{j=1}^{n} \frac{dy^j}{dt}\frac{D\partial_j}{dt} = \sum_{j=1}^{n} \frac{d^2y^j}{dt^2}\partial_j + \sum_{j=1}^{n} \frac{dy^j}{dt}\nabla_{c'(t)}\partial_j \tag{24}$$

$$= \sum_{j=1}^{n} \frac{d^2y^j}{dt^2}\partial_j + \sum_{i,j=1}^{n} \frac{dy^j}{dt}\nabla_{\frac{dy^i}{dt}\partial_i}\partial_j = \sum_{j=1}^{n} \frac{d^2y^j}{dt^2}\partial_j + \sum_{i,j=1}^{n} \frac{dy^j}{dt}\frac{dy^i}{dt}\nabla_{\partial_i}\partial_j \tag{25}$$

$$= \sum_{k=1}^{n} \frac{d^2y^k}{dt^2}\partial_k + \sum_{i,j,k=1}^{n} \frac{dy^j}{dt}\frac{dy^i}{dt}\Gamma_{ij}^k\partial_k = \sum_{k=1}^{n} \left(\frac{d^2y^k}{dt^2} + \sum_{i,j=1}^{n} \frac{dy^i}{dt}\frac{dy^j}{dt}\Gamma_{ij}^k \right) \partial_k. \tag{26}$$

Portanto, temos o seguinte resultado.

Teorema 2.4. *Se M é uma variedade suave com uma conexão ∇ e $c\colon I \to M$ é uma curva, então c é uma geodésica se, com respeito a qualquer carta $(U, \phi) = (U, x^1, \ldots, x^n)$, as componentes de $y = \phi \circ c$ satisfazem o sistema de EDOs*

$$\frac{d^2y^k}{dt^2} + \sum_{i,j=1}^{n} \frac{dy^i}{dt}\frac{dy^j}{dt}\Gamma_{ij}^k = 0 \tag{27}$$

As equações do sistema acima são chamadas de *equações geodésicas*. Pelo teorema de existência e unicidade de solução para EDOs temos a existência e unicidade de geodésicas.

Teorema 2.5. *Seja M uma variedade suave com uma conexão ∇. Dado $p \in M$ e $X_p \in T_pM$, existe uma geodésica $c\colon I \to M$ tal que $c(0) = p$ e $c'(0) = X_p$. Mais do que isso, essa geodésica é única no sentido de que qualquer outra geodésica satisfazendo essas propriedades deve coincidir com c na intersecção de seus domínios.*

Um *difeomorfismo* entre variedades suaves M e N é um mapa $F\colon M \to N$ suave, bijetor e com inversa suave. Se M e N forem riemannianas, dizemos que F é uma *isometria* se, para todos $p \in M$ e $X_p, Y_p \in T_pM$, temos

$$\mathfrak{g}_p(X_p, Y_p) = \mathfrak{g}_{F(p)}(D_pF(X_p), D_pF(Y_p)). \tag{28}$$

Proposição 2.6. *Isometrias preservam conexões de Levi-Civita. Mais ainda, mapas que preservam conexões, preservam geodésicas. Como corolário, isometrias preservam geodésicas.*

Se $c\colon I \to M$ é uma curva e V é um campo ao longo de c, então dizemos que V é *paralelo* se $DV/Dt = 0$. Dessa forma, uma geodésica é uma curva cujo campo velocidade é paralelo. Fixado $X_p \in T_{c(t_0)}M$, existe um único campo V ao longo de c, paralelo, tal que $V(t_0) = X_p$. Se $c\colon [a,b] \to M$ é uma curva e V é um campo paralelo ao longo de c, dizemos que $V(b)$ é obtido a partir de $V(a)$ por *translação paralela*. Dizemos que $V(b)$ é o *transporte paralelo* de $V(a)$ ao longo de c.

Proposição 2.7. *Se V e W são paralelos ao longo de c em uma variedade riemanniana M, então $||V||$ e $\mathfrak{g}(V,W)$ são constantes.*

Um problema importante com questão ao transporte paralelo é a existência. Ela está garantida pelo resultado abaixo.

Teorema 2.8. *Se M é uma variedade suave com uma conexão ∇ e $c\colon [a,b] \to M$ uma curva. Dado $v \in T_{c(a)}M$, existe um campo vetorial paralelo V_t ao longo de c tal que $V_a = v$.*

2.5 Mapa exponencial e mapa logarítmico

Uma geodésica $c\colon I \to M$ é *maximal* se não podemos estender c para um intervalo maior do que I sem que a curva deixe de ser uma geodésica. Do Teorema 2.5 temos que, dado $p \in M$ e $X_p \in T_pM$ existe uma única geodésica maximal c com $c(0) = p$ e $c'(0) = X_p$. Vamos denotar essa geodésica por γ_{X_p}.

O *mapa exponencial* em um ponto $p \in M$ é a função dada por $\operatorname{Exp}_p(X_p) = \gamma_{X_p}(1)$. Esse mapa não está necessariamente definido para todo $X_p \in T_pM$, visto que nem sempre γ_{X_p} possui 1 no seu domínio. Uma variedade com uma conexão é dita *completa* se toda geodésica puder ter seu domínio estendido para todo $\mathbb{R}$. No caso de variedades riemannianas consideradas com a conexão de Levi-Civita, temos dois resultados que nos ajudam no sentido de definir Exp_p para um conjunto satisfatório de vetores.

Proposição 2.9. *Para qualquer $p \in M$, com M variedade riemanniana, existem uma vizinhança U de p e dois números $\epsilon, \delta > 0$ tais que para todos $q \in U$ e $v \in T_qM$ com $||v|| < \delta$, existe uma única geodésica $\gamma\colon]-\varepsilon, \varepsilon[\to M$ com $\gamma(0) = q$ e $\gamma'(0) = v$.*

Corolário 2.10. *Para qualquer $p \in M$, com M variedade riemanniana, existem uma vizinhança U de p e um número $\delta > 0$ tais que para todos $q \in U$ e $v \in T_qM$ com $||v|| < \delta$ existe uma única geodésica $\gamma\colon]-2, 2[\to M$ com $\gamma(0) = q$ e $\gamma'(0) = v$.*

O Corolário 2.10 nos diz que o mapa exponencial está sempre definido em todas as direções, porém essa existência só está garantida para velocidades pequenas. Se você for muito rápido, pode ficar cansado muito rápido e não dar tempo do seu conjunto de parâmetros englobar o 1.

Proposição 2.11. *A derivada $D_0 \operatorname{Exp}_p$ é a identidade em T_pN para qualquer $p \in M$.*

A Proposição acima garante, em particular, que sempre existe um $\varepsilon > 0$ tal que Exp_p mapeia $B(0, \varepsilon)$ difeomorficamente em M. Por causa disso, existe uma inversa para o mapa exponencial, que chamaremos de *mapa logarítmico*.

2.6 Curvatura por via de tensores

Fixada uma conexão ∇ em M, já conhecemos o tensor de torção, que é dado por

$$T(X,Y) = \nabla_X Y - \nabla_Y X - [X,Y]. \tag{29}$$

Ao tomarmos $X = \partial_i$ e $Y = \partial_j$ em uma carta (U, ϕ), temos $[X,Y] = \partial_i\partial_j - \partial_j\partial_i = 0$, portanto se a conexão ∇ é a de Levi-Citiva, temos pelo anulamento da torção que $\nabla_{\partial_i}\partial_j = \nabla_{\partial_j}\partial_i$. Dessa maneira, as derivadas desses campos comutam.

Vamos tentar entender o que acontece para o tensor de curvatura. Comecemos lembrando que ele é dado por

$$R(X,Y)Z = \nabla_X \nabla_Y Z - \nabla_Y \nabla_X Z - \nabla_{[X,Y]} Z. \tag{30}$$

Ao tomarmos $X = \partial_i$, $Y = \partial_j$ e Z qualquer, temos novamente que $[X,Y] = 0$, portanto $\nabla_{[X,Y]} Z = 0$. Dessa forma, segue que

$$R(\partial_i, \partial_j)Z = \nabla_{\partial_i} \nabla_{\partial_j} Z - \nabla_{\partial_j} \nabla_{\partial_i} Z. \tag{31}$$

Porém, mesmo que ∇ seja a conexão de Levi-Civita, não temos garantia de que $R(X,Y)Z = 0$, dessa forma nem sempre derivar um campo em direções diferentes independe da ordem dessas derivadas. O que vai medir a diferença entre essas operações é a curvatura da sua variedade.

Para entendermos a curvatura geometricamente, precisamos falar de holonomia. Dado $p \in M$ e $\gamma\colon [a,b] \to M$ fechada em p e contrátil, para cada $v \in T_pM$ podemos considerar o campo V_t ao longo de γ que seja paralelo e satisfaça $V_a = v$. O vetor $v' = V_b$, que é o transporte paralelo de v ao longo de γ, é chamado de *holonomia de v ao longo de γ*.

A menos que sua variedade possua curvatura 0, isso é, se $R(X,Y)Z = 0$ para todos X, Y, Z, a sempre existirá $v \in T_pM$ tal que $v' \neq v$. Ou seja, o ângulo entre esses dois vetores é também medido pela curvatura.

Agora vamos entender quem são as possíveis curvaturas de uma variedade riemanniana. Por enquanto, vamos dar enfoque em três principais tipos: a seccional (ou gaussiana), a de Ricci e a escalar.

A primeira coisa a notar é que, dados $x, y, z \in T_pM$, podemos construir campos suaves X, Y, Z ao redor de p de maneira que $X_p = x$, $Y_p = y$ e $Z_p = z$. Definimos então $R(x,y)z$ como sendo o campo $R(X,Y)Z$ no ponto p.

Teorema 2.12. *Seja $p \in M$ com $\dim M \geq 2$ e $W \leq T_pM$ um subespaço de dimensão 2 (também conhecido como plano). Considere x, y uma base para W e defina o número*

$$K(W) = \frac{\mathfrak{g}_p(R(x,y)x, y)}{|x \wedge y|^2}, \tag{32}$$

onde $|x \wedge y|$ é a área do paralelogramo formado por x e y, que pode ser explicitamente calculada por

$$|x \wedge y| = \sqrt{||x||^2 ||y||^2 - \mathfrak{g}_p(x,y)}, \tag{33}$$

onde $||\cdot||$ é a norma induzida pela métrica $\mathfrak{g}$. Temos então que $K(W)$ não depende da base escolhida para W.

A quantidade $K(W)$ definida no teorema acima é a *curvatura seccional* ou *curvatura gaussiana* de W em p. Agora podemos usar essa curvatura para falarmos de curvatura de Ricci. Dado $p \in M$ e $v \in T_pM$ unitário, considere $w \in v^\perp$. Se $P_w = \mathbb{R}v + \mathbb{R}w$, então temos a curvatura seccional $K(P_w)$. A *curvatura de Ricci em p na direção v* é a média de todas essas curvaturas seccionais, ou seja,

$$\mathrm{Ricci}_p(v) = \lambda \int_S K(P_w) dV, \tag{34}$$

onde S é a esfera unitária em T_pM, dV é a forma de volume em S e λ é uma constante positiva que é, honestamente, irrelevante. De fato, ela pode ser calculada explicitamente se utilizarmos a seguinte proposição.

Proposição 2.13. *Se $w_1, \ldots, w_{n-1}$ é uma base ortonormal de $v^\perp$, então*

$$\mathrm{Ricci}_p(v) = \frac{1}{n-1} \sum_{i=1}^{n-1} \mathfrak{g}_p(R(x, z_i)x, z_i). \tag{35}$$

Probabilisticamente, podemos pensar que curvatura de Ricci é a curvatura seccional média de planos aleatórios da forma P_w. Isso nos dá alguma ideia do porquê não conseguimos recuperar as curvaturas seccionais a partir da curvatura de Ricci: há perda de informação, pois a partir da média de um conjunto de dados raramente conseguimos recuperar quem são esses dados.

Por fim, a *curvatura escalar* nada mais é do que uma média das curvaturas de Ricci, ou seja, se $p \in M$, podemos considerar uma base ortonormal $z_1, \ldots, z_n$ de T_pM. A curvatura escalar é o número definido por

$$K_s(p) = \frac{1}{n} \sum_{i=1}^{n} \text{Ricci}_p(z_i). \tag{36}$$

Por mais que a curvatura escalar não dependa de uma direção, e apenas do ponto, não é a ela que nos referimos ao dizer que M tem curvatura constante κ em $p \in M$. Essa expressão diz que todas as curvaturas seccionais em p valem κ.

2.7 Conceitos métricos

Antes de brincarmos com a geometria hiperbólica, vamos falar de duas definições que aparecem na teoria de espaços métricos e que podem ser úteis mais para frente. Dado um espaço métrico (X, d), o *produto de Gromov* é uma operação que, dados três pontos $x, y, z \in X$, retorna o número

$$(y, z)_x = \frac{1}{2}(d(x, y) + d(x, z) - d(y, z)). \tag{37}$$

Dizemos que X é *δ-hiperbólico*, com $\delta > 0$, se para todos $x, y, z, w \in X$ temos

$$(x, z)_w \geq \min\{(x, y)_w, (y, z)_w\} - \delta. \tag{38}$$

Uma definição equivalente envolve triângulos geodésicos. Uma *geodésica* em (X, d) é a imagem isométrica de um intervalo $[a, b]$. Se γ é essa isometria, $x = \gamma(a)$ e $y = \gamma(b)$, denotamos a imagem de γ por $[x, y]$. Um *triângulo geodésico com vértices* $x, y, z \in X$ é a união das geodésicas $[x, y]$, $[y, z]$ e $[z, x]$. Se para cada $m \in [x, y]$ existe um ponto em $n \in [y, z] \cup [z, w]$ tal que $d(m, n) < \delta$, dizemos que o triângulo geodésico $\Delta(x, y, z)$ é *δ-fino*. Dizemos que (X, d) é *δ-hiperbólico* se todo triângulo geodésico é δ-fino.

Por fim, precisamos falar de distorção, que é uma medida de fidelidade para certos mergulhos de dados em aprendizado de máquina. Se X e Y são espaços métricos e $f\colon X \to Y$ é um mergulho de dados em X para pontos de Y, a *distorção de f em $x, y \in X$* é dada por

$$\mathbb{D}_f(a, b) = \frac{|d(a, b) - d(f(a), f(b))|}{d(a, b)}. \tag{39}$$

3 Modelos para a geometria hiperbólica

O espaço hiperbólico real é uma variedade riemanniana de curvatura constante igual a -1. Existem diversos modelos isométricos para ele, e agora vamos falar de alguns. Aqui, é importante notar que todos esses modelos estão mergulhados

3.1 O semi-espaço de Poincaré

O primeiro modelo do qual vamos falar é o semi-espaço de Poincaré. Ele é dado, como variedade suave, pelo semi-espaço

$$\mathbb{H}^n = \{x \in \mathbb{R}^n \mid x_n > 0\}. \tag{40}$$

Sua métrica, por sua vez, é dada pela expressão

$$ds^2 = \frac{dx_1^2 + \cdots + dx_n^2}{x_n^2}. \tag{41}$$

Essa expressão significa que, para cada $p \in \mathbb{H}^n$, temos

$$\mathfrak{g}_p(u, v) = \frac{u_1 v_1 + \cdots u_n v_n}{p_n^2}. \tag{42}$$

As formas diferenciais dx_i recebem os vetores u, v e retornam as respectivas coordenadas. Por sua vez, sempre que x_i aparecer em uma fórmula, ele será substituído pela i-ésima coordenada do ponto onde a métrica está sendo construída.

Esse espaço, assim como toda variedade riemanniana, possui uma estrutura de espaço métrico induzida pela métrica riemanniana. Essa métrica, em $\mathbb{H}^n$, é dada por

$$d(x,y) = \operatorname{arccosh}\left(1 + \frac{||x-y||^2}{2x_n y_n}\right). \tag{43}$$

3.2 O hiperboloide de Lorentz

O modelo do hiperboloide depende do que chamamos de *métrica de Lorentz* em $\mathbb{R}^{n+1}$. Ela é definida por

$$\langle x, y\rangle_{\mathbb{L}} = -x_0 y_0 + x_1 y_1 + \cdots + x_n y_n. \tag{44}$$

Como variedade suave, o hiperboloide é definido por

$$\mathbb{L}^n = \{x \in \mathbb{R}^{n+1} \mid \langle x, x\rangle_{\mathbb{L}} = -1, x_0 > 0\}. \tag{45}$$

A métrica riemanniana em $\mathbb{L}^n$ é induzida também da métrica de Lorentz, e é dada por

$$ds^2 = -dx_0^2 + dx_1^2 + \cdots + dx_n^2. \tag{46}$$

Note que essa métrica não faz muito sentido, visto que o espaço tangente de $\mathbb{L}^n$ deveria ter dimensão n, mas aqui utilizamos $n+1$ coordenadas. De fato, essa métrica se aplica apenas ao considerarmos o espaço tangente $T_p\mathbb{H}^n$ como o conjunto $p^\perp = \{x \in \mathbb{R}^n \mid \langle x, p\rangle_{\mathbb{L}} = 0\}$. A distância no hiperboloide, por sua vez, é dada por

$$d(x,y) = \operatorname{arccosh}(-\langle x, y\rangle_{\mathbb{L}}). \tag{47}$$

É importante notarmos aqui que a métrica riemanniana que definimos não parece ser um produto interno, justamente pelo fator negativo $-dx_0^2$. De fato, a métrica de Lorentz não é um produto interno em $\mathbb{R}^n$, mas ao restringirmos ela ao espaço $p^\perp$ para qualquer $p \in \mathbb{L}^n$, essa restrição é sempre um produto interno.

3.3 O n-disco de Poincaré

O n-disco de Poincaré é, como variedade suave, dado por

$$\mathbb{B}^n = \{x \in \mathbb{R}^n \mid ||x|| < 1\}. \tag{48}$$

Sua métrica é definida como

$$ds^2 = 4\frac{dx_1^2 + \cdots + dx_n^2}{(1 - x_1^2 + \cdots + x_n^2)^2} \tag{49}$$

e sua distância por

$$d(x,y) = \operatorname{arccosh}\left(1 + 2\frac{||x-y||^2}{(1-||x||^2)(1-||y||^2)}\right). \tag{50}$$

3.4 O n-disco de Beltrami-Klein

O n-disco de Beltrami-Klein é, como variedade suave, dado por

$$\mathbb{K}^n = \{x \in \mathbb{R}^n \mid ||x|| < 1\}. \tag{51}$$

É a mesma variedade suave que da origem ao disco de Poincaré, mas a denotamos por uma letra diferente apenas para diferenciar os dois modelos. A diferença é na métrica riemanniana, que em $\mathbb{K}^n$ é dada por

$$ds^2 = \frac{dx_1^2 + \cdots dx_n^2}{1 - x_1^2 - \cdots - x_n^2} + \frac{(x_1 dx_1 + \cdots + x_n dx_n)^2}{(1 - x_1^2 - \cdots - x_n^2)^2}. \tag{52}$$

Como consequência, a distância nesse espaço é diferente da distância no disco de Poincaré, e nesse caso é dada por

$$d(x,y) = \frac{1}{2}\ln\left(\frac{||a-x|| \cdot ||b-y||}{||a-y|| \cdot ||b-x||}\right). \tag{53}$$

Aqui, a e b são pontos construídos a partir de x e y pelo seguinte método: considere a reta r que passa por x e y e defina por a e b os pontos em que r intersecta $\mathbb{S}^{n-1}$. O ponto a será escolhido como o que estiver mais próximo a x, e o ponto b por consequência será escolhido como o que estiver mais próximo a y.

3.5 O modelo do hemisfério

O último modelo que iremos visitar é o do hemisfério. Como variedade suave, ele é dado por

$$\mathbb{J}^n = \{x \in \mathbb{S}^n \mid x_{n+1} > 0\}, \tag{54}$$

e possui métrica dada por

$$ds^2 = \frac{dx_1^2 + \cdots + dx_{n+1}^2}{x_{n+1}^2}. \tag{55}$$

A distância em $\mathbb{J}^n$, por sua vez, é dada por

$$d(x, y) = \operatorname{arccosh}\left(\langle \phi(x), \phi(y) \rangle_{\mathbb{L}}\right), \tag{56}$$

onde ϕ é o mapa dado por

$$(x_1, \ldots, x_{n+1}) \mapsto \left(\frac{x_1}{x_{n+1}}, \ldots, \frac{x_n}{x_{n+1}}, \frac{1}{x_{n+1}}\right). \tag{57}$$

3.6 Isometrias entre os modelos

Devemos agora explicitar isometrias entre os modelos definidos acima. Para isso, definiremos apenas quatro desses mapas e, como composição de isometrias é uma isometria, definir apenas essas quatro funções nos dará isometrias entre quaisquer dois modelos por meio de tomar inversas e compor.

- O isomorfismo entre $\mathbb{L}^n$ e $\mathbb{B}^n$ é dado por

$$(x_1, \ldots, x_{n+1}) \in \mathbb{L}^n \mapsto \left(\frac{x_2}{1 + x_1}, \ldots, \frac{x_{n+1}}{1 + x_1}\right) \in \mathbb{B}^n; \tag{58}$$

- O isomorfismo entre $\mathbb{B}^n$ e $\mathbb{H}^n$ é dado por

$$(x_1, \ldots, x_n) \in \mathbb{B}^n \mapsto \frac{1}{1 + 2x_1 + ||x||^2}(1 - ||x||^2, 2x_2, \ldots, 2x_n) \in \mathbb{H}^n; \tag{59}$$

- O isomorfismo entre $\mathbb{L}^n$ e $\mathbb{K}^n$ é dado por

$$(x_1, \ldots, x_{n+1}) \in \mathbb{L}^n \mapsto \left(\frac{x_2}{x_1}, \ldots, \frac{x_{n+1}}{x_1}\right) \in \mathbb{K}^n; \tag{60}$$

- O isomorfismo entre $\mathbb{L}^n$ e $\mathbb{J}^n$ é dado por

$$(x_1, \ldots, x_{n+1}) \in \mathbb{L}^n \mapsto \left(\frac{x_1}{x_{n+1}}, \ldots, \frac{x_n}{x_{n+1}}, \frac{1}{x_{n+1}}\right) \in \mathbb{J}^n. \tag{61}$$

3.7 Generalizando operações euclidianas

O próximo passo agora é generalizar algumas operações de espaços euclidianos para os espaços hiperbólicos. Faremos isso pois, para construir redes neurais em espaços euclidianos, precisamos de álgebra linear, e portanto é uma boa ideia aprender como fazer álgebra linear em um espaço que não é vetorial. Um *girogrupo* é um conjunto G munido de uma operação binária $\oplus$ satisfazendo as seguintes propriedades:

- existe ao menos um elemento $0 \in G$ tal que $0 \oplus a = a$ para todo $a \in G$. Todo elemento satisfazendo essa condição é chamado de *identidade à esquerda*.
- existe alguma identidade à esquerda $0 \in G$ de maneira que, para todo $a \in G$, existe $\ominus a \in G$ de maneira que $\ominus a \oplus a = 0$;
- para todos $a, b, c \in G$, existe um elemento $\operatorname{gyr}[a, b]c \in G$ tal que vale a igualdade $a \oplus (b \oplus c) = (a \oplus b) \oplus \operatorname{gyr}[a, b]c$;

- O mapa $\mathrm{gyr}[a,b]\colon c \mapsto \mathrm{gyr}[a,b]c$ é um automorfismo de G, isso é, é uma bijeção que satisfaz $\mathrm{gyr}[a,b](c \oplus d) = \mathrm{gyr}[a,b]c \oplus \mathrm{gyr}[a,b]d$;
- vale a *propriedade de redução à direita*, isso é, $\mathrm{gyr}[a,b] = \mathrm{gyr}[a \oplus b, b]$ para todos $a, b \in G$.

Um girogrupo G é *girocomutativo* se para todos $a, b \in G$ vale $a \oplus b = \mathrm{gyr}[a,b](b \oplus a)$.

O principal exemplo de girogrupo para o estudo de aprendizado profundo é o *girogrupo de Möbius*. Considere o n-disco de Poincaré $\mathbb{B}^n$ e defina nele a *soma de Möbius* dada por

$$x \oplus y = \frac{(1 + 2\langle x, y\rangle + ||y||^2)x + (1 - ||x||^2)y}{1 + 2\langle x, y\rangle + ||x||^2||y||^2}. \tag{62}$$

O par $(\mathbb{B}^n, \oplus)$ é um girogrupo girocomutativo, que é chamado de girogrupo de Möbius de raio 1. Em um contexto mais geral, poderíamos tomar o interior de qualquer esfera centrada em 0 em qualquer espaço vetorial real com produto interno, mas para nossos estudos isso não será necessário.

No girogrupo de Möbius podemos definir algumas operações extremamente importantes para a construção de redes neurais:

- o *produto por escalar de Möbius* é definido por

$$\lambda \otimes x = \begin{cases} \tanh(\lambda \operatorname{arctanh} ||x||)\frac{x}{||x||}, & \text{se } x \neq 0, \\ 0, & \text{se } x = 0; \end{cases} \tag{63}$$

- podemos aplicar uma matriz $M \in M_n(\mathbb{R})$ em $x \in \mathbb{B}^n$ pela operação

$$M^{\otimes}(x) = \tanh\left(\frac{||Mx||}{||x||} \operatorname{arctanh} ||x||\right)\frac{Mx}{||Mx||}; \tag{64}$$

Utilizando essas operações e algum conhecimento sobre geodésicas em $\mathbb{B}^n$ podemos derivar expressões explicitas para os mapas exponencial e logarítmico:

$$\mathrm{Exp}_p(v) = p \oplus \left(\tanh\left(\frac{\lambda_p||v||}{2}\right)\frac{v}{||v||}\right) \quad \text{e} \quad \mathrm{Log}_p(q) = \frac{2}{\lambda_p}\operatorname{arctanh}(||-x \oplus y||)\frac{-x \oplus y}{||-x \oplus y||}. \tag{65}$$

onde $p, q \in \mathbb{B}^n$, $v \in T_p\mathbb{B}^n$ e λ_x é um fator de conformalidade entre a métrica euclidiana e a métrica do disco de Poincaré, isso é, λ_p é dado por

$$\lambda_p = \frac{2}{1 - ||p||^2} \tag{66}$$

e claramente satisfaz

$$4\frac{dx_1^2 + \cdots + dx_n^2}{(1 - x_1^2 - \cdots - x_n^2)^2} = \lambda_x^2(dx_1^2 + \cdots + dx_n^2), \tag{67}$$

ou seja, se $\mathfrak{g}$ é a métrica em $\mathbb{R}^n$ e $\mathfrak{g}_B$ é a métrica no n-disco, então $\mathfrak{g}_B = \lambda_x^2\mathfrak{g}$.

Se estamos estudando um conjunto de dados em $\mathbb{B}^n$, é interessante sabermos computar a média desses dados. Existem três maneiras de fazer isso:

- se os dados fazem parte de um grafo, então podemos computar a média de todos os vizinhos x_j de um ponto $x_i \in \mathbb{B}^n$ pela fórmula

$$\mu = \mathrm{Exp}_{x_i}\left(\sum_{j \in \mathcal{N}(i)} w_{ij} \mathrm{Log}_{x_i}(x_j)\right), \tag{68}$$

  onde cada $w_{ij} \in \mathbb{R}$ é um peso associado a aresta que liga x_i com x_j;

- se $x_1, \ldots, x_n \in \mathbb{K}^n$ (agora estamos no disco e Beltrami-Klein), podemos computar o *ponto médio de Einstein* por

$$\mu = \frac{\sum_{i=1}^n \frac{x_i}{||x_i||^2}}{\sum_{i=1}^n \frac{1}{||x_i||^2}}; \tag{69}$$

- por último, podemos utilizar as operações já definidas para construir o ponto médio entre $x_1, \ldots, x_n \in \mathbb{B}^n$, que é chamado de *ponto giromédio* dado por

$$m(x_1, \ldots, x_n, \alpha) = \frac{1}{2} \otimes \left(\sum_{i=1}^n \frac{\frac{2\alpha_i}{||x_i||^2}}{\sum_{j=1}^n \alpha_j\left(\frac{2}{||x_i||^2} - 1\right)} x_i\right), \tag{70}$$

  onde $\alpha = (\alpha_1, \ldots, \alpha_n)$ é uma lista de pesos para cada x_i.

4 Redes neurais

4.1 PyTorch e tensores

Para nossos estudos, definiremos um tensor de maneira indutiva. Um tensor de dimensão 0 é um número. Um tensor de dimensão 1 é uma lista de números, e seu formato é o número de elementos, denotado por (n). Um tensor de dimensão 2 é uma lista de tensores de dimensão 1, que devem ter todos o mesmo formato, ou seja, o mesmo número de elementos. O formato de um tensor bidimensional é um par ordenado (m, n), onde m é o número de tensores de dimensão 1 que o compõe, e n o número de elementos de cada um desses tensores unidimensionais.

De maneira indutiva, um tensor de dimensão n é uma lista de tensores de dimensão $n-1$, que devem todos ter o mesmo formato. O formato de um tensor n-dimensional é uma lista de n números $(x_1, \ldots, x_n)$, onde x_1 é o número de tensores $(n-1)$-dimensionais que o compõe, e $(x_2, \ldots, x_n)$ é o formato de cada um deles.

Vamos agora começar a brincar com programação. Utilizaremos `Python` com a biblioteca `PyTorch` para mexer com aprendizado profundo. Para importar essa biblioteca, utilizamos `import torch`. O próximo passo é entender como funcionam os tensores. Para criar um tensor, utilizamos o comando `torch.tensor()`. Dentro dos parênteses, podemos colocar 4 parâmetros (existem mais deles, mas esses são os importantes):

- o primeiro parâmetro é um tensor, ou seja, uma lista de listas de listas e etc. assim como definimos nos parágrafos acima;
- `dtype` recebe um tipo de dado, que o tensor irá armazenar em cada uma de suas entradas;
- `device` recebe o dispositivo no qual o tensor irá ser armazenado, como uma *CPU*, uma *GPU* ou uma *TPU*;
- `requires_grad` recebe `True` ou `False` e, em caso de `True`, computa o gradiente do tensor (veremos o que é isso mais para frente) e o armazena na memória.

Podemos retornar o formato, o tipo dos dados ou a dimensão de um tensor pelos atributos `shape`, `dtype` e `ndim`. Por exemplo, considere o tensor:

```
import torch # A biblioteca foi importada pois eh a primeira vez que escrevemos codigo em
# bloco, mas isso sera evitado nas proximas instancias.

TENSOR = torch.tensor([[1, 2],
                       [3, 4]])
```

Podemos utilizar os atributos comentados acima para retornar certas informações importantes:

- `TENSOR.shape` vai retornar `torch.Size([2, 2])`, visto que `TENSOR` tem formato $(2, 2)$;
- `TENSOR.dtype` vai retornar `torch.int64`, visto que as entradas da matriz são números inteiros;
- `TENSOR.ndim` vai retornar `2`, visto que o tensor em questão tem dimensão 2, que é a quantidade de números do seu formato.

Podemos indexar tensores. Para retirar um número de um tensor, precisamos especificar as coordenadas desse número em cada dimensão do tensor. Por exemplo, considere o código abaixo.

```
TENSOR = torch.tensor([[[1, 2, 3],
                        [4, 5, 6],
                        [7, 8, 9]],

                       [[10, 11, 12],
                        [13, 14, 15],
                        [16, 17, 18]],

                       [[19, 20, 21],
                        [22, 23, 24],
                        [25, 26, 27]]])

print(TENSOR.shape)
```

O output desse código será, como vimos acima, `torch.Size([3, 3, 3])`. É importante notar que o `Python` sempre começa a contar do 0, e não é diferente quanto trabalhamos com `PyTorch`.

Para descobrirmos a posição em que o número 15 está, basta fazermos o processo de trás pra frente. Primeiro, percebemos que ele está na segunda matriz, então ele faz parte de `TENSOR[1]`. Nessa matriz, ele está segunda linha, portanto ele é um elemento de `TENSOR[1,1]`. Por fim, ele é o terceiro elemento dessa linha, dessa maneira temos `TENSOR[1,1,2] = 15`. De fato, se após o código acima escrevermos `print(TENSOR[1,1,2])` o resultado será precisamente `tensor(15)`. O retorno não é precisamente `15` pois elementos de tensores também são tensores, mas isso é apenas uma tecnicalidade.

Existem diversas maneiras de construir tensores em `Python`. Além de `torch.tensor`, podemos utilizar

- `torch.ones`, que recebe diversos argumentos, um para cada dimensão que você deseja que o tensor tenha. Esses argumentos precisam ser números maiores do que 1 e representam quantas entradas o tensor terá nas respectivas dimensões. Por exemplo, `torch.ones(2,2,2)` resulta no tensor abaixo;

```
tensor([[[1., 1.],
         [1., 1.]],

        [[1., 1.],
         [1., 1.]]])
```

- `torch.zeros` é idêntico ao `torch.ones`, mas o tensor terá 0 em todas as entradas;
- `torch.rand` recebe argumentos da mesma maneira que os comandos acima, mas as entradas do tensor serão números aleatórios no intervalo $[0, 1[$ gerados a partir de uma distribuição uniforme;
- `torch.zeros_like` e `torch.ones_like` ambos recebem um tensor como entrada e retornam um outro tensor cheio de zeros, no primeiro caso, ou uns, no segundo, que tenha o mesmo formato do tensor recebido;
- `torch.full_like` recebe um tensor e um valor numérico, e cria um tensor com o mesmo formato do tensor recebido, mas em que todas as entradas são idênticas ao valor recebido;
- `torch.arange` recebe três parâmetros numéricos `start`, `end` e `step`. Ele retorna um tensor unidimensional que possui valores em $[start, end[$, começando em `start` e com um espaçamento de tamanho `step` entre cada valor.

Tensores só aceita valores numéricos, mas mesmo dentre esses existem vários tipos que podem ser utilizados, e muitas vezes esses tipos geram conflitos quando vamos operar com tensores. É importante portanto aprender a converter tensores de um tipo para outro.

É importante notar que conversão entre tipos não vai, na maioria das vezes, alterar os valores armazenados no tensor. A diferença, por exemplo, é que o tensor `[1]` possui um valor do tipo inteiro, enquanto o tensor `[1.]` possui um valor do tipo ponto flutuante. O problema acontece quando convertemos pontos flutuantes para inteiros, visto que acontece um processo de arredondamento.

```
TENSOR = torch.tensor([1.44]) # TENSOR.dtype = torch.float32
print(TENSOR.type(torch.int64))
```

O código acima converte um tensor do tipo `torch.float32` para um tensor do tipo `torch.int64` e exibe o resultado. Ao executarmos esse código em uma IDE, percebemos que o retorno é `tensor([1])`, visto que o tensor resultante deve possuir apenas inteiros. É importante que, ao contrário do que pensamos que seria o correto, o arredondamento é sempre feito para baixo, e não seguindo a regra do 5 como nos acostumamos. É possível achar uma lista com todos os possíveis tipos de valores na documentação do `PyTorch`.

Vamos agora aprender a operar com tensores. Existem cinco operações importantes:

- a adição de tensores se dá elemento a elemento, ou seja, se `T` e `S` são k-tensores (tensores com k dimensões), então dado `K = T + S` temos `K[x1, ..., xk] = T[x1, ..., xk] + S[x1, ..., xk]`. Por exemplo,

```
T = torch.tensor([[1, 2],
                  [3, 4]])
S = torch.tensor([[5, 6]
                  [7, 8]])
K = T + S
print(K[0,1])
```

retorna `tensor(8)`, já que `T[0,1] + S[0,1] = 2 + 6 = 8`.

- a subtração, a multiplicação e a divisão funcionam exatamente da mesma maneira, e são calculadas por `T - S`, `T * S` e `T / S` respectivamente. É importante notar que, para a divisão, o tensor `S` não pode possuir valores valendo `0`;

A última operação que vamos analisar é o produto matricial, mas esse é assustadoramente mais complicado então vamos ter que fazer tudo com calma. Ele é definido a partir e vários casos, da seguinte maneira: sejam `T` e `S` dois tensores. Se

- `T` e `S` tiverem formato (n), então `K = torch.matmul(T, S)` tem formato $()$ e é dado por

$$K = \sum_{j=0}^{n-1} T[j] \cdot S[j]; \tag{71}$$

- `T` tem formato (n) e `S` tem formato (n, m), então `K = torch.matmul(T, S)` tem formato (m) e é dado por

$$K[i] = \sum_{j=0}^{n-1} T[j] \cdot S[j, i]; \tag{72}$$

- `T` tem formato (n, m) e `S` tem formato (m), então `K = torch.matmul(T, S)` tem formato (n) e é dado por

$$K[i] = \sum_{j=0}^{m-1} T[i, j] \cdot S[j]; \tag{73}$$

- `T` tem formato (n, m) e `S` tem formato (m, p), então `K = torch.matmul(T, S)` tem formato (n, p) e é dado por

$$K[i, j] = \sum_{k=0}^{m-1} T[i, k] \cdot S[k, j]; \tag{74}$$

- `T` tem formato (x_{k-1}) e `S` tem formato $(x_1, \ldots, x_k)$ com $k > 2$, então `K = torch.matmul(T, S)` tem formato $(x_1, \ldots, x_{k-2}, x_k)$ e é dado por

$$K[i_1, \ldots, i_{k-1}] = \sum_{j=0}^{x_{k-1}-1} T[j] \cdot S[i_1, \ldots, i_{k-2}, j, i_{k-1}] \tag{75}$$

- `T` tem formato $(x_1, \ldots, x_k)$ com $k > 2$ e `S` tem formato (x_k), então `K = torch.matmul(T, S)` tem formato $(x_1, \ldots, x_{k-1})$ e é dado por

$$K[i_1, \ldots, i_{k-1}] = \sum_{j=0}^{x_k-1} T[i_1, \ldots, i_{k-1}, j] \cdot S[j]; \tag{76}$$

- `T` tem formato (n, x_{k-1}) e `S` tem formato $(x_1, \ldots, x_k)$ com $k > 2$, então `K = torch.matmul(T, S)` tem formato $(x_1, \ldots, x_{k-2}, n, x_k)$ e é dado por

$$K[i_1, \ldots, i_k] = \sum_{j=0}^{x_{k-1}-1} T[i_{k-1}, j] \cdot S[i_1, \ldots, i_{k-2}, j, i_k]; \tag{77}$$

- `T` tem formato $(x_1, \ldots, x_k)$ com $k > 2$ e `S` tem formato (x_k, m), então `K = torch.matmul(T, S)` tem formato $(x_1, \ldots, x_{k-1}, m)$ e é dado por

$$K[i_1, \ldots, i_k] = \sum_{j=0}^{x_k-1} T[i_1, \ldots, i_{k-1}, j] \cdot S[j, i_k]; \tag{78}$$

Os últimos dois casos são os mais complicados. Primeiro, definimos que dois formatos $(x_1, \ldots, x_k)$ e $(y_1, \ldots, y_s)$ são *transmissíveis* se $k, s \geq 1$, ou seja, se ambos não são escalares, e se, para todo i com $1 \leq i \leq \max\{k, s\}$ temos que

- y_i ou x_i não existem, ou
- $y_i = 1$ ou $x_i = 1$ ou,
- $x_i = y_i$.

Se $k > 2$, `T` tem formato $(x_1, \ldots, x_k)$ e `S` tem formato $(y_1, \ldots, y_k)$, então se $(x_1, \ldots, x_{k-2})$ e $(y_1, \ldots, y_{k-2})$ são transmissíveis e $x_k = y_{k-1}$, podemos calcular `K = torch.matmul(T, S)`, que tem formato

$$(\max\{x_1, y_1\}, \ldots, \max\{x_{k-2}, y_{k-2}\}, x_{k-1}, y_k) \tag{79}$$

e é dado por

$$K[i_1, \ldots, i_k] = \sum_{j=0}^{x_k - 1} T[\alpha_1, \ldots, \alpha_{k-2}, i_{k-1}, j] \cdot S[\beta_1, \ldots, \beta_{k-2}, j, i_k], \tag{80}$$

onde $\alpha_n = \min(i_n, x_n - 1)$ e $\beta_m = \min(i_m, y_m - 1)$.

O último caso é o que acontece se os tensores não tem a mesma dimensão. Se `T` tem formato $(x_1, \ldots, x_k)$ e `S` tem formato $(y_1, \ldots, y_s)$ com $k \neq s$ e $k, s > 2$, então temos dois casos: $k > s$ ou $k < s$. Se $k > s$, vamos construir um novo tensor `S1` com dimensão k e formato dado por $(1, \ldots, 1, y_1, \ldots, y_s)$ onde as $k - s$ primeiras entradas valem 1. Esse tensor, claro, é dado por

$$S_1[0, ..., 0, i_1, \ldots, i_s] = S[i_1, \ldots, i_s]. \tag{81}$$

Ao assumirmos que `T` e `S` são transmissíveis (ou seja, os seus formatos menos as duas últimas dimensões são transmissíveis), então `T` e `S1` também serão, e portanto pelo caso anterior podemos calcular `K = torch.matmul(T,S1)`, que será precisamente o tensor resultante ao calcularmos `K = torch.matmul(T, S)`. O caso $k < s$ é análogo, porém o tensor que será modificado é o `T`.

Para terminarmos a seção, podemos construir um análogo à transposição de matrizes, mas para tensores. Se `T` é um tensor de formato $(x_1, \ldots, x_k)$, o seu *transposto* é de formato $(x_k, \ldots, x_1)$ e é dado por

$$T^{\perp}[i_1, \ldots, i_k] = T[i_k, \ldots, i_1]. \tag{82}$$

Para calcular o transposto pelo `PyTorch`, basta utilizar

```
torch.permute(T,list(torch.arange(T.ndim - 1, -1, -1))).
```

A função `torch.permute` recebe um tensor de dimensão k e uma lista com os números de 0 até $k - 1$ em qualquer ordem. Ela retorna então um novo tensor k-dimensional, contendo as mesmas entradas do tensor original, mas com o formato trocado de maneira a respeitar a lista de números. Para ser um pouco mais construtivo, imagine que `T` tem formato $(x_1, \ldots, x_k)$. Se `l = [n1, ..., nk]` é uma lista com os inteiros de 0 até $k - 1$, então o tensor `T1 = torch.permute(T, l)` é dado por

$$T_1[i_1, \ldots, i_k] = T[i_{n_1+1}, \ldots, i_{n_k+1}]. \tag{83}$$

Note que, para transpor um tensor k-dimensional, basta que a lista passada como argumento de `torch.permute` seja `[k-1, ..., 0]`, que é justamente o resultado de

```
list(torch.arange(T.ndim - 1, -1, -1))
```

4.2 Trabalhando com tensores

Referências

[1] Hugo Cattarucci Botós. «Geometrias Clássicas». 2020. URL: https://github.com/HugoCBotos/geometria-classica/blob/master/Geometrias%20Cl%C3%A1ssicas%20-%20Hugo%20C.%20Bot%C3%B3s.pdf.

[2] M. Ferreira e G. Ren. «Möbius gyrogroups: A Clifford algebra approach». Em: *Journal of Algebra* 328.1 (2011).

[3] Anna Wienhard e Gye-Seon Lee. «Curvature of Riemannian Manifolds». 2015. URL: https://www.mathi.uni-heidelberg.de/~lee/Soeren05.pdf.

[4] *PyTorch documentation - PyTorch 1.13 documentation*. Acessado: 01-03-2023. URL: https://pytorch.org/docs/stable/index.html.

[5] Loring W. Tu. *Differential Geometry: Connections, Curvature and Characteristic Classes*. Springer-Verlag New York Inc, 2017. ISBN: 978-3-319-55082-4.

[6] Wei Peng e Tuomas Varanka e Abdelrahman Mostafa e Henglin Shi e Guoying Zhao. «Hyperbolic Deep Neural Networks: A Survey». Em: *JOURNAL OF LaTeX CLASS FILES* 14.8 (2015).

[7] Abraham Ungar. «Beyond Pseudo-rotations in Pseudo-Euclidean Spaces - An Introduction to the Theory of Bi-gyrogroups and Bi-gyrovector Spaces». Em: ed. por Themistocles M. Rassias. Elsevier, 2018. Cap. 2 e 3, 9 até 97.

[8] James W. Cannon e William J. Floyd e Richard Kenyon e Walter R. Parry. «Flavours of Geometry». Em: ed. por Silvio Levy. MSRI Publications, 1997. Cap. 2, 59 até 115.