Luiz Guilherme Morais da Costa Faria

APRENDIZADO DE MÁQUINA

Brasília, DF

21 de setembro de 2025

Luiz Guilherme Morais da Costa Faria

APRENDIZADO DE MÁQUINA

Universidade de Brasília

Orientador: Nome do Orientador/Revisor (se aplicável)

Brasília, DF
21 de setembro de 2025

Sumário

Sumário 3

I HISTÓRIA DA IA E DO COMPUTADOR 9

1 UMA BREVE HISTÓRIA DO COMPUTADOR 11

2 UMA BREVE HISTÓRIA DA INTELIGÊNCIA ARTIFICIAL . . . 13

II CONCEITOS MATEMÁTICOS 15

3 CÁLCULO PARA APRENDIZADO DE MÁQUINA 17

3.1 Funções: A Base do Cálculo 17

3.2 Derivadas Ordinárias 17

3.3 Integrais Simples 17

3.4 Derivadas Parciais 17

4 ÁLGEBRA LINEAR PARA APRENDIZADO DE MÁQUINA . . . 19

4.1 A Unidade Fundamental: Vetores e Espaços Vetoriais 19

4.2 Organizando Dados: Matrizes e Suas Operações 19

4.3 Tensores: A Estrutura de Dados do Deep Learning 19

4.4 Resolvendo Sistemas e Encontrando Propriedades: Autovalores e Autovetores 19

4.5 Decomposição de Matrizes (SVD e PCA) 19

5 PROBABILIDADE E ESTATÍSTICA PARA APRENDIZADO DE MÁQUINA 21

5.1 Medindo a Incerteza: Probabilidade Básica e Condicional 21

5.2 O Teorema de Bayes: Aprendendo com Evidências 21

5.3 Descrevendo os Dados: Estatística Descritiva: Média, mediana, variância, desvio padrão 21

5.4 Variáveis Aleatórias e Distribuições de Probabilidade 21

5.5 A Função de Máxima Verossimilhança (Maximum Likelihood Estimation - MLE) 21

III PILARES DAS REDES NEURAIS 23

6 O ALGORITMO DA REPROPROPAGAÇÃO E OS OTIMIZADORES BASEADOS EM GRADIENTE 25
6.1 O Método do Gradiente Descendente 25
6.1.1 Exemplo Ilustrativo 25
6.1.2 O Método em Si 26
6.1.3 Implementação em Python 28
6.2 A Retropropagação: Aprendendo com os Erros 28
6.3 Otimizadores Baseados em Gradiente 29
6.3.1 Método do Gradiente Estocástico 29
6.3.2 Método do Gradiente com Momentum 29
6.3.3 Nesterov 29
6.3.4 AdaGrad 29
6.3.5 RMSProp 29
6.3.6 Adam 29
6.3.7 Nadam 29
6.4 O Método de Newton: Indo Além do Gradiente 29

7 FUNÇÕES DE ATIVAÇÃO SIGMOIDAIS 31
7.1 Teoremas da Aproximação Universal 31
7.2 Exemplos Ilustrativo 31
7.3 A Sigmoide Logística 31
7.4 Tangente Hiperbólica 31
7.5 Softsign: Uma Sigmoidal Mais Barata 31
7.6 Hard Sigmoid e Hard Tanh: O Sacrifício da Suavidade em Prol do Desempenho 31
7.7 O Desaparecimento de Gradientes 31
7.8 Comparativo de Desempenho das Sigmoidais 31

8 FUNÇÕES DE ATIVAÇÃO RETIFICADORAS 33
8.1 Exemplo Ilustrativo 33
8.2 Rectified Linear Unit e Revolução Retificadora 33
8.3 Dying ReLUs Problem 33
8.4 Corrigindo o Dying ReLUs Problem: As Variantes com Vazamento 33
8.4.1 Leaky ReLU 33
8.4.2 Parametric ReLU 33
8.4.3 Randomized Leaky ReLU 33
8.5 Em Busca da Suavidade 33
8.5.1 Exponential Linear Unit 33

8.5.2 Scaled Exponential Linear Unit 33
8.5.3 Noisy ReLU 33
8.6 O Problema dos Gradientes Explosivos 33
8.7 Comparativo de Desempenho das Funções Retificadoras 33

9 FUNÇÕES DE ATIVAÇÃO MODERNAS E OUTRAS FUNÇÕES DE ATIVAÇÃO 35

10 FUNÇÕES DE PERDA PARA CLASSIFICAÇÃO BINÁRIA 37
10.1 A Intuição da Perda: Medindo o Erro do Modelo 37
10.2 Entropia Cruzada Binária (Binary Cross-Entropy): A função de perda padrão 37
10.3 Perda Hinge (Hinge Loss) 37
10.4 Comparativo Visual e Prático 37

11 FUNÇÕES DE PERDA PARA CLASSIFICAÇÃO MULTILABEL . 39
11.1 Softmax e a Distribuição de Probabilidades 39
11.2 Entropia Cruzada Categórica (Categorical Cross-Entropy) 39
11.3 Entropia Cruzada Categórica Esparsa (Sparse Categorical Cross-Entropy) 39

12 METAHEURÍSTICAS: OTIMIZANDO REDES NEURAIS SEM O GRADIENTE 41
12.1 Algoritmos Evolutivos 41
12.2 Inteligência de Enxame 41

IV APRENDIZADO DE MÁQUINA CLÁSSICO 43

13 TÉCNICAS DE REGRESSÃO 45
13.1 Exemplo Ilustrativo 45
13.2 Regressão Linear 45
13.2.1 Função de Custo MSE 45
13.2.2 Equação Normal 45
13.2.3 Implementação em Python 45
13.3 Regressão Polininomial 45
13.3.1 Impletanção em Python 45
13.4 Regressão de Ridge 45
13.4.1 Implementação em Python 45
13.5 Regressão de Lasso 45
13.5.1 Implementação em Python 45

13.6 Elastic Net **45**
13.6.1 Implementação em Python 45
13.7 Regressão Logística **45**
13.7.1 Implementação em Python 45
13.8 Regressão Softmax **45**
13.8.1 Implementação em Python 45
13.9 Outras Técnicas de Regressão **45**

14 ÁRVORES DE DECISÃO E FLORESTAS ALEATÓRIAS 47
14.1 Exemplo Ilustrativo **47**
14.2 Entendendo o Conceito de Árvores **47**
14.2.1 Árvores Binárias 47
14.3 Árvores de Decisão **47**
14.3.1 Implementação em Python 47
14.4 Florestas Aleatórias **47**
14.4.1 Implementação em Python 47

15 MÁQUINAS DE VETORES DE SUPORTE 49
15.1 Exemplo Ilustrativo **49**

16 ENSAMBLE 51
16.1 Exemplo Ilustrativo **51**

17 DIMENSIONALIDADE 53
17.1 Exemplo Ilustrativo **53**
17.2 A Maldição da Dimensionalidade **53**
17.3 Seleção de Características (Feature Selection) **53**
17.4 Extração de Características (Feature Extraction) **53**
17.4.1 Análise de Componentes Principais (PCA) 53
17.4.2 t-SNE (t-Distributed Stochastic Neighbor Embedding) e UMAP 53

18 CLUSTERIZAÇÃO 55
18.1 Exemplo Ilustrativo **55**
18.2 Aprendizado Não Supervisionado: Encontrando Grupos nos Dados **55**
18.3 Clusterização Particional: K-Means **55**
18.4 Clusterização Hierárquica **55**
18.5 Clusterização Baseada em Densidade: DBSCAN **55**

V REDES NEURAIS PROFUNDAS (DNNS) 57

19 PERCEPTRONS MLP - REDES NEURAIS ARTIFICIAIS 59

20 REDES FEEDFORWARD (FFNS) 61

21 REDES DE CRENÇA PROFUNDA (DBNS) E MÁQUINAS DE BOLTZMANN RESTRITAS 63

22 REDES NEURAIS CONVOLUCIONAIS (CNN) 66

22.1 Exemplo Ilustrativo 66

22.2 Camadas Convolucionais: O Bloco Fundamental para as CNNs . . 66

22.2.1 Implementação em Python 66

22.3 Camadas de Poooling: Reduzindo a Dimensionalidade 66

22.3.1 Max Pooling 66

22.3.2 Avg Pooling 66

22.3.3 Global Abg Pooling 66

22.3.4 Implementação em Python 66

22.4 Camada Flatten: Achatando os Dados 66

22.4.1 Implementação em Python 66

22.5 Criando uma CNN 66

22.6 Detecção de Objetos 66

22.7 Redes Totalmente Convolucionais (FCNs) 66

22.8 You Only Look Once (YOLO) 66

22.9 Algumas Arquiteturas de CNNs 66

22.9.1 LeNet-5 66

22.9.2 AlexNet 66

22.9.3 GoogLeNet 66

22.9.4 VGGNet 66

22.9.5 ResNet 66

22.9.6 Xception 66

22.9.7 SENet 66

23 REDES RESIDUAIS (RESNETS) 67

24 REDES NEURAIS RECORRENTES (RNN) 69

24.1 Exemplo Ilustrativo 69

24.2 Neurônios e Células Recorrentes 69

24.2.1 Implementação em Python 69

24.3 Células de Memória 69

24.3.1 Implementação em Python 69

24.4 Criando uma RNN 69

24.5 O Problema da Memória de Curto Prazo 69

24.5.1 Células LSTM 69

24.5.2 Conexões Peephole 69
24.5.3 Células GRU 69

25 TÉCNICAS PARA MELHORAR O DESEMPENHO DE REDES NEURAIS 71
25.1 Técnicas de Inicialização 71
25.2 Reguralização L1 e L2 71
25.3 Normalização 71
25.3.1 Normalização de Camadas 71
25.3.2 Normalização de Batch 71
25.4 Cliping do Gradiente 71
25.5 Dropout: Menos Neurônios Mais Aprendizado 71
25.6 Data Augmentation 71

26 TRANSFORMERS 73
26.1 As Limitações das RNNs: O Gargalo Sequencial 73
26.2 A Ideia Central: Self-Attention (Query, Key, Value) 73
26.3 Escalando a Atenção: Multi-Head Attention 73
26.4 A Arquitetura Completa: O Bloco Transformer 73
26.5 Entendendo a Posição: Codificação Posicional 73
26.6 As Três Grandes Arquiteturas 73
26.6.1 Encoder-Only (Ex: BERT): Para tarefas de entendimento 73
26.6.2 Decoder-Only (Ex: GPT): Para tarefas de geração 73
26.6.3 Encoder-Decoder (Ex: T5): Para tarefas de tradução/sumarização 73
26.7 Além do Texto: Vision Transformers (ViT) 73

27 REDES ADVERSÁRIAS GENERATIVAS (GANS) 75

28 MIXTURE OF EXPERTS (MOE) 77

29 MODELOS DE DIFUSÃO 79

30 REDES NEURAIS DE GRAFOS (GNNS) 81

VI APÊNDICES 83

Referências 85

Parte I

História da IA e do Computador

1 Uma Breve História do Computador

O texto do seu capítulo começa aqui...

2 Uma Breve História da Inteligência Artificial

O texto do seu capítulo começa aqui...

Parte II

Conceitos Matemáticos

3 Cálculo para Aprendizado de Máquina

3.1 Funções: A Base do Cálculo

3.2 Derivadas Ordinárias

3.3 Integrais Simples

3.4 Derivadas Parciais

4 Álgebra Linear para Aprendizado de Máquina

4.1 A Unidade Fundamental: Vetores e Espaços Vetoriais

4.2 Organizando Dados: Matrizes e Suas Operações

4.3 Tensores: A Estrutura de Dados do Deep Learning

4.4 Resolvendo Sistemas e Encontrando Propriedades: Autovalores e Autovetores

4.5 Decomposição de Matrizes (SVD e PCA)

5 Probabilidade e Estatística para Aprendizado de Máquina

5.1 Medindo a Incerteza: Probabilidade Básica e Condicional

5.2 O Teorema de Bayes: Aprendendo com Evidências

5.3 Descrevendo os Dados: Estatística Descritiva: Média, mediana, variância, desvio padrão

5.4 Variáveis Aleatórias e Distribuições de Probabilidade

5.5 A Função de Máxima Verossimilhança (Maximum Likelihood Estimation - MLE)

Parte III

Pilares das Redes Neurais

6 O Algoritmo da Repropropagação e Os Otimizadores Baseados em Gradiente

6.1 O Método do Gradiente Descendente

O Método do Gradiente faz parte de uma série de métodos numéricos que possuem como função otimizar diferentes funções. Métodos dessa forma veem sendo estudados a séculos, um exemplo disso é o trabalho Méthode générale pour la résolution des systèmes d'équations simultanées (1847) no matemático francês do século XVIII Cauchy (1847), em que o autor apresenta um método pode ser considera um precursor para o método do gradiente atual.

Nesse texto, o autor apresenta uma fórma de minimizar uma função de múltiplas variáveis ($u = f(x, y, z)$) que não assume valores negativos, para fazer isso, ele faz uso de cálculo de derivadas parciais dessa função de cada um dos seus componentes ($D_x u, D_y u, D_z u$), em seguida, ele realiza um passo de atualização, de forma que os os valores de cada uma das variáveis sejam ligeiramente incrementados por valores (α, β, γ) (Cauchy, 1847). Um ponto importante destacado por Cauchy (1847) é de que esses incrementos devem ser proporcionais ao negativo das suas respectivas derivadas parciais, ele descreve que esse processo de calcular as derivadas e fazer pequenos incrementos deve ser feito de forma iterativa, assim, calculá-se as derivas, faz os incrementos, e o passo é repetido até convergir para o valor mínimo de u.

Esse trabalho explica bem como aplicar o método do gradiente para se calcular mínimos de funções, mas para facilitar o entendimento do leitor, em seguida está um exemplo ilutrativo explicando o funcionamento dessa ferramenta.

6.1.1 Exemplo Ilustrativo

Imagine que você adora aventuras, e por isso, decidiu fazer uma trilha em uma floresta que fica em uma cadeia de montanhas que podem ser escaladas. Então, você teve a incrivel ideia de ir para o menor ponto dessa cadeia de montanhas, pois, no guia que você estava seguindo, falava que havia um lago com uma água cristalina, perfeito para tirar fotos.

Para chegar até esse lago, você conta com uma bússula um tanto quanto diferente, ao invés dela apontar para o norte como uma bússola comum, ela aponta para a direção do lugar com menor altitude de uma região. Isso é perfeito para o que você precisa, pois ela irá apontar justamente para o lago de você quer ir.

Com isso em mente, você criou um plano de como irá chegar a esse lago, ele é método iteativo que conta com os seguintes passos considerando que você está em uma posição inicial qualquer:

1. Olha na bússola qual a direção ela está apontando;
2. Anda um metro em na direção apontada pela bússola.

Esse é um método iterativo, você sabe que se ficar repetindo-o por um certo tempo, você provavelmente irá chegar ao seu destino.

Na matemática, existe um método semelhante a este, que busca com base em uma bússola (chamada de vetor gradiente), encontrar um ponto de mínimo de um determinado lugar (neste caso, uma função composta por múltiplas variáveis). Esse é o método do gradiente, ele é ponto central desse capítulo, pois, ele (e suas variações) junto com o algortimo da retroprogação são uma as principais ferramentas que colaboram para que os modelos de aprendizado de máquina possam aprender com os seus erros e com isso se tornarem melhores a cada iteração.

6.1.2 O Método em Si

A vantagem do método do gradiente é que ele é uma ferramenta matemática, e por isso pode ser representado utilizando notações mais formais e de forma enxuta. As notações utilizadas por Cauchy são diferentes das que são utilizadas hoje em dia, mas o seu significado não muda. Em Deep Learning (2016), Goodfellow, Bengio e Courville (2016) explicam essa ferramenta através da equação 6.1 que deve ser repetida por múltiplos passos até o modelo convergir, ou seja, encontrar o ponto de mínimo da função estudada.

Método do Gradiente Descendente

$$x' = x - \epsilon \nabla f(x) \tag{6.1}$$

Em que:

- x': as coordenadas do próximo ponto;
- x: as coordernadas do ponto atual;
- ϵ: representa o tamanho do passo, também chamado de taxa de aprendizado;
- $\nabla f(x)$: representa o vetor gradiente calculado na posição do ponto atual (x) para função que se deseja otimizar.

Um ponto a ser destacado nesse metodo é na hora de escolher uma taxa de aprendizado para ser utilizada no método. Uma taxa de aprendizado muito pequena signifca que o passo que o modelo irá dar de um ponto para outro será menor, e com isso implica que ele levará mais passos para encontrar um ponto de mínimo. É como se você fosse comparar a quantidade de passos que você gasta para andar do seu quarto até a sua cozinha com a quantidade de passos dados por uma formiga até lá, ambos vão chegar no local, mas a formiga certamente irá demorar bem mais. Considerando isso, surge então a hipótese de que quanto maior for o passo, mais rápido será a convergência, mas isso também não funciona muito bem, pois um passo muito largo pode ultrapassar o ponto de mínimo indo parar em outro canto da função, e ficará tentando chegar até o mínimo mas não irá conseguir pois caminha uma distância muito grande de uma só vez. Essas duas situações são ilustradas na figura 1.

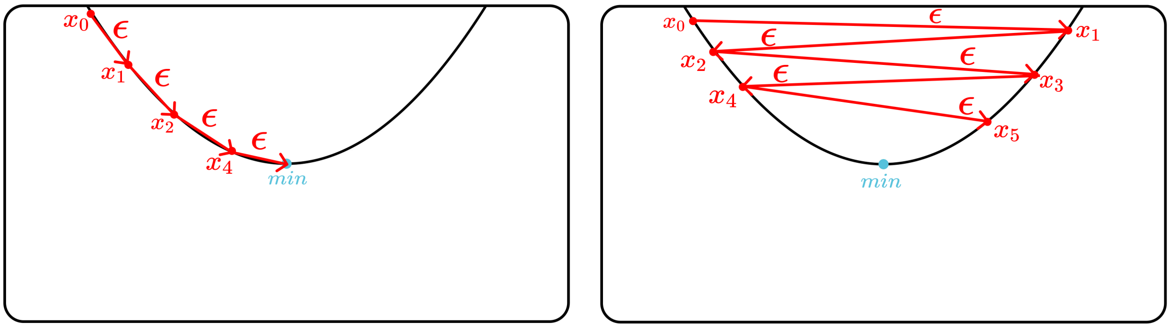

Figura 1 – Comparativo do tamanho de passos em uma função polinomial.

Na prática, escolher o valor da taxa de aprendizado é uma tarefa que irá depender de modelo em modelo, também irá variar com os diferentes métodos de otimização além da topologia da rede neural que está sendo construída. É sempre recomendado então experimentar diferentes tamanhos de passo, de forma que seja encontrado um que melhor se ajusta ao cenário que está sendo trabalhado.

Outro ponto que deve-se atentar é com relação as funções que estarão sendo analisadas ao utilizar o método do gradiente, uma função convexa faz garante uma melhor convergência quando comparada com uma não convexa pois sitações que em que o modelo ficará preso em pontos de mínimos locais ou em pontos de sela não irão acontecer. Assim, também deve-se atentar com esse fator, pois ele poderá ser responsável por indicar ao treinar um modelo haverão problemas em sua convergência para encontrar um ponto de mínimo.

Outro ponto que deve-se atentar é com relação as funções que estão sendo analisas ao utilizar o método do gradiente mas também qualquer otimizador que seja baseado nele. Se tivermos uma função convexa, que seu formato lembra um funil, será bem mais fácil para o modelo encontrar o ponto de mínimo global daquela função. Mas se tivermos uma função não convexa, cheia de ondas e com muitos pontos de mínimos locais e pontos de sela, a convergência do modelo será pior, pois existe a chance de que ele fique preso em um ponto de mínimo local ou em um ponto de sela. Isso afeta diretamente o desempenho da rede neural que estará sendo criada, fazendo com que ela tenha métricas piores. O problema é que muitas das vezes a função $f(x)$ que estaremos interessados para calcular o desemepenho do modelo será não convexa, dificultando o seu aprendizado.

Na figura 2 é possível ver o gráfico de duas funções diferentes, a primeira sendo uma função convexa e a segunda uma função não convexa.

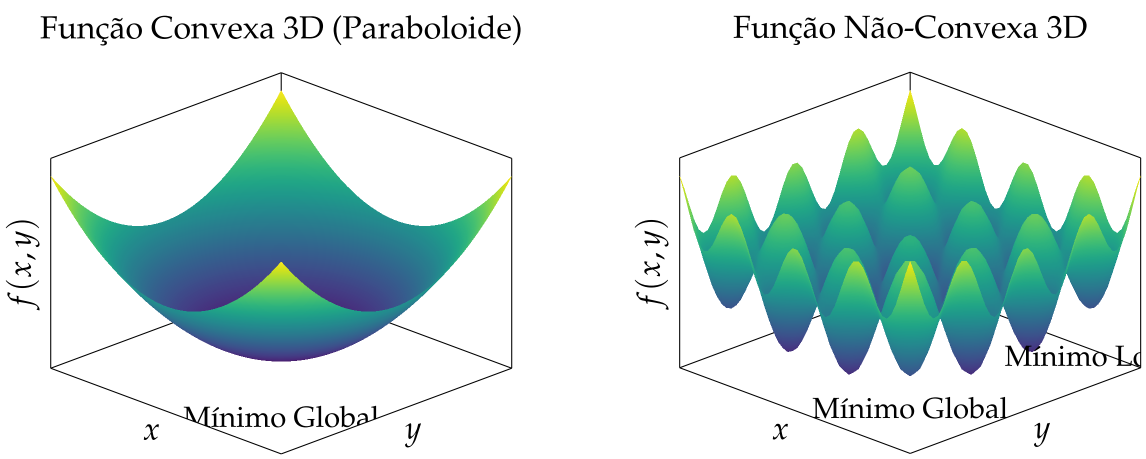

Figura 2 – Comparação entre funções 3D convexas e não-convexas.

6.1.3 Implementação em Python

6.2 A Retropropagação: Aprendendo com os Erros

Ainda no contexto de utilizar com o vetor gradiente para otimizar um modelo de rede neural, existe uma ferramenta que trabalha justamente com essse processo, ela é a retropropagação ou *backpropagation* em inglês.

> **Definição** A **retropropagação** é uma ferramenta que veio para permitir que redes que fazem o uso de unidades de neurônios possam aprender, para isso, o procedimento ajusta repetidamente os pesos das conexões da rede para minimizar a diferença entre o valor atual da saída do vetor da rede neural com o valor real desejado (Rumelhart; Hinton; Williams, 1986).

Essa ferramenta foi introduzida para a comunidade científica pelos pesquisadores Rumelhart, Hinton e Williams (1986) no texto *Learning Representations by Back-Propagating Errors (1986)*,

6.3 Otimizadores Baseados em Gradiente

6.3.1 Método do Gradiente Estocástico

6.3.2 Método do Gradiente com Momentum

6.3.3 Nesterov

6.3.4 AdaGrad

6.3.5 RMSProp

6.3.6 Adam

6.3.7 Nadam

6.4 O Método de Newton: Indo Além do Gradiente

7 Funções de Ativação Sigmoidais

7.1 Teoremas da Aproximação Universal

7.2 Exemplos Ilustrativo

7.3 A Sigmoide Logística

7.4 Tangente Hiperbólica

7.5 Softsign: Uma Sigmoidal Mais Barata

7.6 Hard Sigmoid e Hard Tanh: O Sacrifício da Suavidade em Prol do Desempenho

7.7 O Desaparecimento de Gradientes

7.8 Comparativo de Desempenho das Sigmoidais

8 Funções de Ativação Retificadoras

8.1 Exemplo Ilustrativo

8.2 Rectified Linear Unit e Revolução Retificadora

8.3 Dying ReLUs Problem

8.4 Corrigindo o Dying ReLUs Problem: As Variantes com Vazamento

8.4.1 Leaky ReLU

8.4.2 Parametric ReLU

8.4.3 Randomized Leaky ReLU

8.5 Em Busca da Suavidade

8.5.1 Exponential Linear Unit

8.5.2 Scaled Exponential Linear Unit

8.5.3 Noisy ReLU

8.6 O Problema dos Gradientes Explosivos

8.7 Comparativo de Desempenho das Funções Retificadoras

9 Funções de Ativação Modernas e Outras Funções de Ativação

O texto do seu capítulo começa aqui...

10 Funções de Perda para Classificação Binária

10.1 A Intuição da Perda: Medindo o Erro do Modelo

10.2 Entropia Cruzada Binária (Binary Cross-Entropy): A função de perda padrão

10.3 Perda Hinge (Hinge Loss)

10.4 Comparativo Visual e Prático

11 Funções de Perda para Classificação Multilabel

11.1 Softmax e a Distribuição de Probabilidades

11.2 Entropia Cruzada Categórica (Categorical Cross-Entropy)

11.3 Entropia Cruzada Categórica Esparsa (Sparse Categorical Cross-Entropy)

12 Metaheurísticas: Otimizando Redes Neurais Sem o Gradiente

O texto do seu capítulo começa aqui...

12.1 Algoritmos Evolutivos

12.2 Inteligência de Enxame

Parte IV

Aprendizado de Máquina Clássico

13 Técnicas de Regressão

13.1 Exemplo Ilustrativo

13.2 Regressão Linear

13.2.1 Função de Custo MSE

13.2.2 Equação Normal

13.2.3 Implementação em Python

13.3 Regressão Polininomial

13.3.1 Impletanção em Python

13.4 Regressão de Ridge

13.4.1 Implementação em Python

13.5 Regressão de Lasso

13.5.1 Implementação em Python

13.6 Elastic Net

13.6.1 Implementação em Python

13.7 Regressão Logística

13.7.1 Implementação em Python

13.8 Regressão Softmax

13.8.1 Implementação em Python

13.9 Outras Técnicas de Regressão

14 Árvores de Decisão e Florestas Aleatórias

14.1 Exemplo Ilustrativo

14.2 Entendendo o Conceito de Árvores

14.2.1 Árvores Binárias

14.3 Árvores de Decisão

14.3.1 Implementação em Python

14.4 Florestas Aleatórias

14.4.1 Implementação em Python

15 Máquinas de Vetores de Suporte

15.1 Exemplo Ilustrativo

16 Ensamble

16.1 Exemplo Ilustrativo

17 Dimensionalidade

17.1 Exemplo Ilustrativo

17.2 A Maldição da Dimensionalidade

17.3 Seleção de Características (Feature Selection)

17.4 Extração de Características (Feature Extraction)

17.4.1 Análise de Componentes Principais (PCA)

17.4.2 t-SNE (t-Distributed Stochastic Neighbor Embedding) e UMAP

18 Clusterização

18.1 Exemplo Ilustrativo

18.2 Aprendizado Não Supervisionado: Encontrando Grupos nos Dados

18.3 Clusterização Particional: K-Means

18.4 Clusterização Hierárquica

18.5 Clusterização Baseada em Densidade: DBSCAN

Parte V

Redes Neurais Profundas (DNNs)

19 Perceptrons MLP - Redes Neurais Artificiais

O texto do seu capítulo começa aqui...

20 Redes FeedForward (FFNs)

O texto do seu capítulo começa aqui...

21 Redes de Crença Profunda (DBNs) e Máquinas de Boltzmann Restritas

O texto do seu capítulo começa aqui...

22 Redes Neurais Convolucionais (CNN)

22.1 Exemplo Ilustrativo

22.2 Camadas Convolucionais: O Bloco Fundamental para as CNNs

22.2.1 Implementação em Python

22.3 Camadas de Pooolling: Reduzindo a Dimensionalidade

22.3.1 Max Pooling

22.3.2 Avg Pooling

22.3.3 Global Abg Pooling

22.3.4 Implementação em Python

22.4 Camada Flatten: Achatando os Dados

22.4.1 Implementação em Python

22.5 Criando uma CNN

22.6 Detecção de Objetos

22.7 Redes Totalmente Convolucionais (FCNs)

22.8 You Only Look Once (YOLO)

22.9 Algumas Arquiteturas de CNNs

22.9.1 LeNet-5

22.9.2 AlexNet

22.9.3 GoogLeNet

22.9.4 VGGNet

22.9.5 ResNet

22.9.6 Xception

23 Redes Residuais (ResNets)

O texto do seu capítulo começa aqui...

24 Redes Neurais Recorrentes (RNN)

O texto do seu capítulo começa aqui...

24.1 Exemplo Ilustrativo

24.2 Neurônios e Células Recorrentes

24.2.1 Implementação em Python

24.3 Células de Memória

24.3.1 Implementação em Python

24.4 Criando uma RNN

24.5 O Problema da Memória de Curto Prazo

24.5.1 Células LSTM

24.5.2 Conexões Peephole

24.5.3 Células GRU

25 Técnicas para Melhorar o Desempenho de Redes Neurais

25.1 Técnicas de Inicialização

25.2 Reguralização L1 e L2

25.3 Normalização

25.3.1 Normalização de Camadas

25.3.2 Normalização de Batch

25.4 Cliping do Gradiente

25.5 Dropout: Menos Neurônios Mais Aprendizado

25.6 Data Augmentation

26 Transformers

26.1 As Limitações das RNNs: O Gargalo Sequencial

26.2 A Ideia Central: Self-Attention (Query, Key, Value)

26.3 Escalando a Atenção: Multi-Head Attention

26.4 A Arquitetura Completa: O Bloco Transformer

26.5 Entendendo a Posição: Codificação Posicional

26.6 As Três Grandes Arquiteturas

26.6.1 Encoder-Only (Ex: BERT): Para tarefas de entendimento

26.6.2 Decoder-Only (Ex: GPT): Para tarefas de geração

26.6.3 Encoder-Decoder (Ex: T5): Para tarefas de tradução/sumarização

26.7 Além do Texto: Vision Transformers (ViT)

27 Redes Adversárias Generativas (GANs)

O texto do seu capítulo começa aqui...

28 Mixture of Experts (MoE)

O texto do seu capítulo começa aqui...

29 Modelos de Difusão

O texto do seu capítulo começa aqui...

30 Redes Neurais de Grafos (GNNs)

O texto do seu capítulo começa aqui...

Parte VI

Apêndices

Referências

CAUCHY, Augustin-Louis. Méthode générale pour la résolution des systèmes d'équations simultanées. *Comptes Rendus Hebdomadaires des Séances de l'Académie des Sciences*, v. 25, p. 536–538, 1847. Citado na p. 25.

GOODFELLOW, Ian; BENGIO, Yoshua; COURVILLE, Aaron. *Deep Learning*. [*S. l.*]: MIT Press, 2016. Citado na p. 26.

RUMELHART, David E.; HINTON, Geoffrey E.; WILLIAMS, Ronald J. Learning Representations by Back-Propagating Errors. *Nature*, v. 323, p. 533–536, 1986. Citado na p. 28.