

Aprendizado de Máquina e Redes Neurais com Pytorch

Thiago Esterci¹, Yan Werneck¹

¹ Programa de Pós-graduação em Modelagem Computacional
Universidade federal de Juiz de fora

November 28, 2024



1 Apresentação do curso

2 Introdução

③ Redes Neurais

1 Apresentação do curso

Tutores

Pré requisitos

Exemplo prático

2 Introdução

3 Redes Neurais

1 Apresentação do curso

Tutores

Pré requisitos

Exemplo prático

2 Introdução

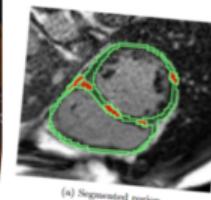
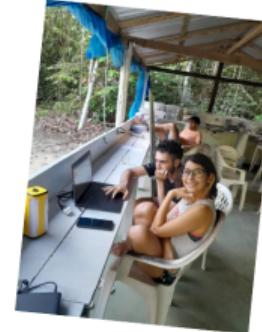
3 Redes Neurais

Thiago Esterci



Sou Thiago Esterci Fernandes, engenheiro mecânico formado e atualmente estou trabalhando para conquistar o título de mestre em Modelagem Computacional também na Universidade Federal de Juiz de Fora (UFJF). Atuo na área de Machine Learning desde 2018 em projetos de diversas áreas, com destaque para minha participação em projetos como o ATLAS junto ao CERN e para projetos de eletrônica com modelos embarcados junto ao CNPq.

Yan Werneck



(a) Segmented regions



(b) Computational m

Sou o Yan, doutorando do PGMC, trabalho com redes neurais aplicadas a eletrofisiologia e a ecologia.

1 Apresentação do curso

Tutores

Pré requisitos

Exemplo prático

2 Introdução

3 Redes Neurais

Pontapé inicial

- ① Navegador atualizado
 - ② Visual Studio Code
 - ③ Anaconda: <https://docs.anaconda.com/anaconda/install>
 - ④ Git: <https://git-scm.com/book/en/v2/Getting-Started-Installing-Git>
 - ⑤ Clonar repositório: https://github.com/Esterci/workshop_ml_pytorch.git
 - ⑥ Seguir README do repositório;

1 Apresentação do curso

Tutores

Pré requisitos

Exemplo prático

2 Introdução

3 Redes Neurais

Desgaste de ferramentas de torno

- Tinha o objectivo de classificar nível de desgaste das ferramentas de corte;
 - Trabalhos publicados:
 - An Autonomous Model to Classify Lathe's Cutting Tools Based on TSFRESH, Self-Organised Direction Aware Data Partitioning Algorithm and Machine Learning Techniques
 - Classification of Lathes Cutting Tool Wear Based on an Autonomous Machine Learning Model
 - Multivariate Time Series and Machine Learning Techniques for the multi-class Classification of Lathes Cutting Tools Wear Condition

① Navegador atualizado
② Visual Studio Code
③ Anaconda: <https://docs.anaconda.com/anaconda/install>
④ Git: <https://git-scm.com/book/en/v2/Getting-Started-Installing-Git>
⑤ Clonar repositório: https://github.com/Esterci/workshop_ml_pytorch.git
⑥ Seguir README do repositório;

1 Apresentação do curso

② Introdução

Modelos baseado em Dados vs Mecanicistas

Aplicação de Aprendizado de Máquinas

O que é Aprendizado de Máquina?

Dilema viés-variância

3 Redes Neurais

1 Apresentação do curso

② Introdução

Modelos baseado em Dados vs Mecanicistas

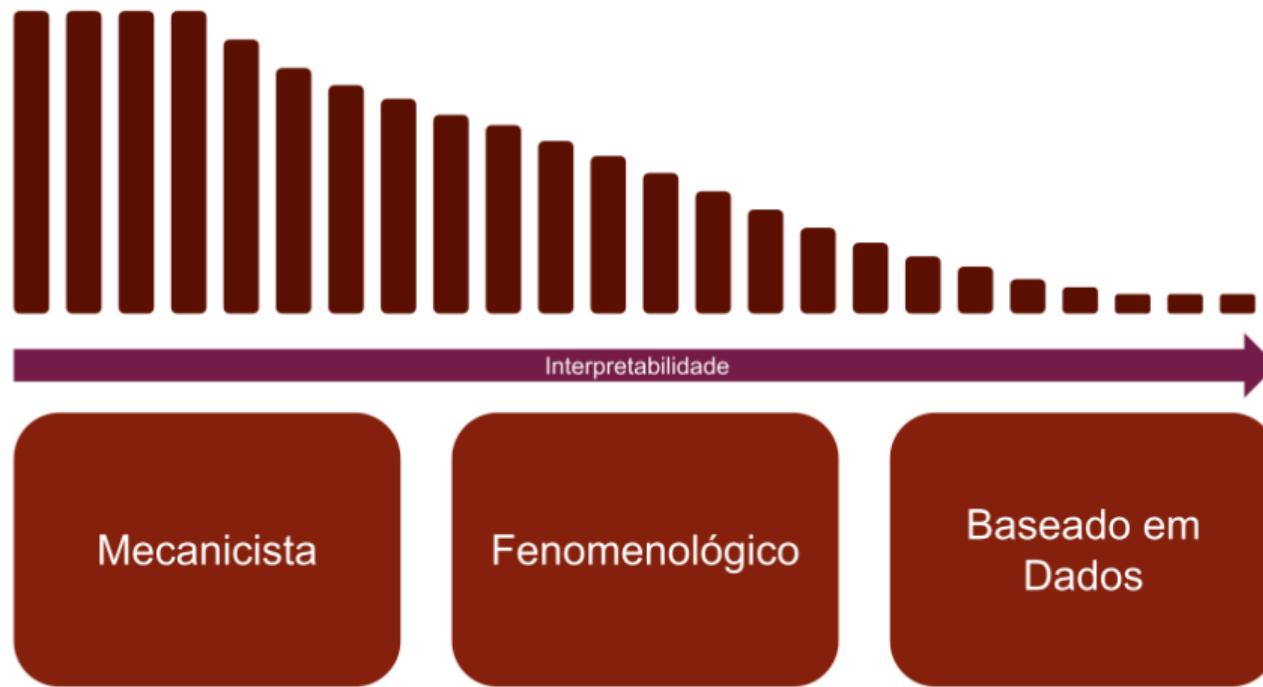
Aplicação de Aprendizado de Máquinas

O que é Aprendizado de Máquina?

Dilema viés-variância

3 Redes Neurais

Cabeça Mecanicista, Coração Fenomenológico e Alma de Dados



1 Apresentação do curso

② Introdução

Modelos baseado em Dados vs Mecanicistas

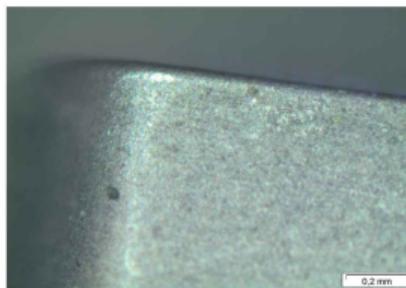
Aplicação de Aprendizado de Máquinas

O que é Aprendizado de Máquina?

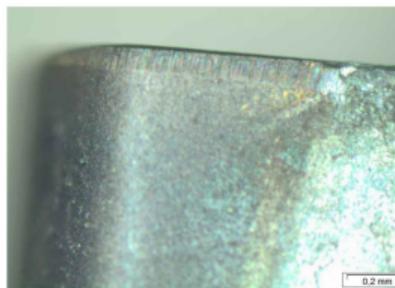
Dilema viés-variância

3 Redes Neurais

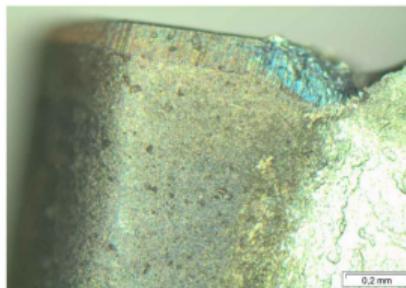
Classificação de ferramentas de torno



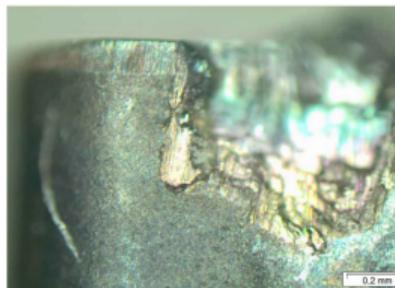
(a) Adequate Condition



(b) Adequate Condition



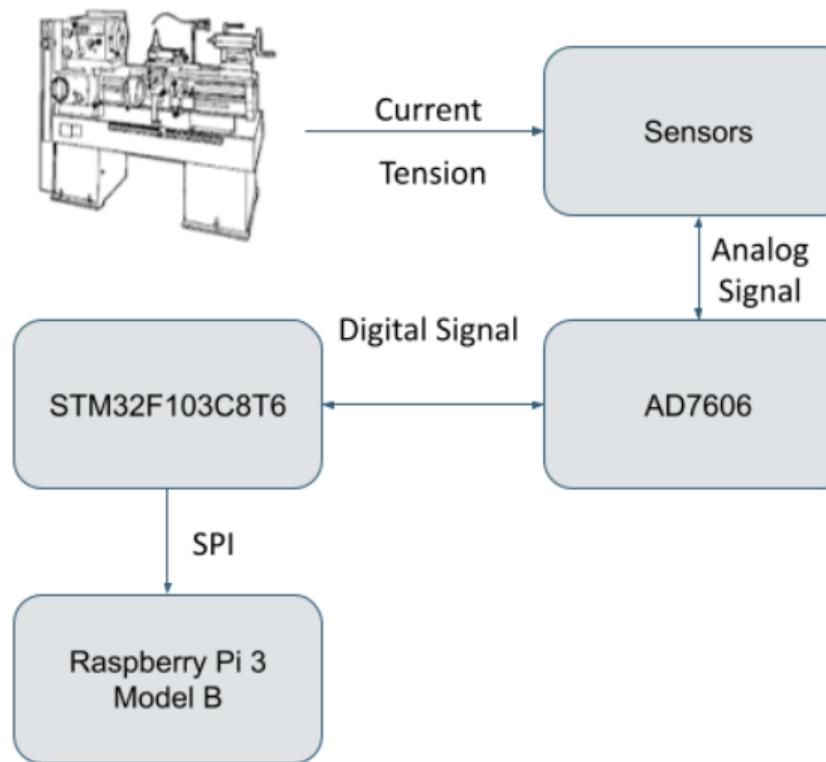
(c) Intermediate Condition



(d) Inadequate Condition

- O trabalho tinha como objetivo classificar o estado de desgaste da ferramenta de corte;
 - Trabalhos publicados:
 - An Autonomous Model to Classify Lathe's Cutting Tools Based on TSFRESH, Self-Organised Direction Aware Data Partitioning Algorithm and Machine Learning Techniques
 - Classification of Lathes Cutting Tool Wear Based on an Autonomous Machine Learning Model
 - Multivariate Time Series and Machine Learning Techniques for the multi-class Classification of Lathes Cutting Tools Wear Condition

Classificação de ferramentas de torno



Modelos mecanicistas envolvidos:

- Equações de Maxwell:

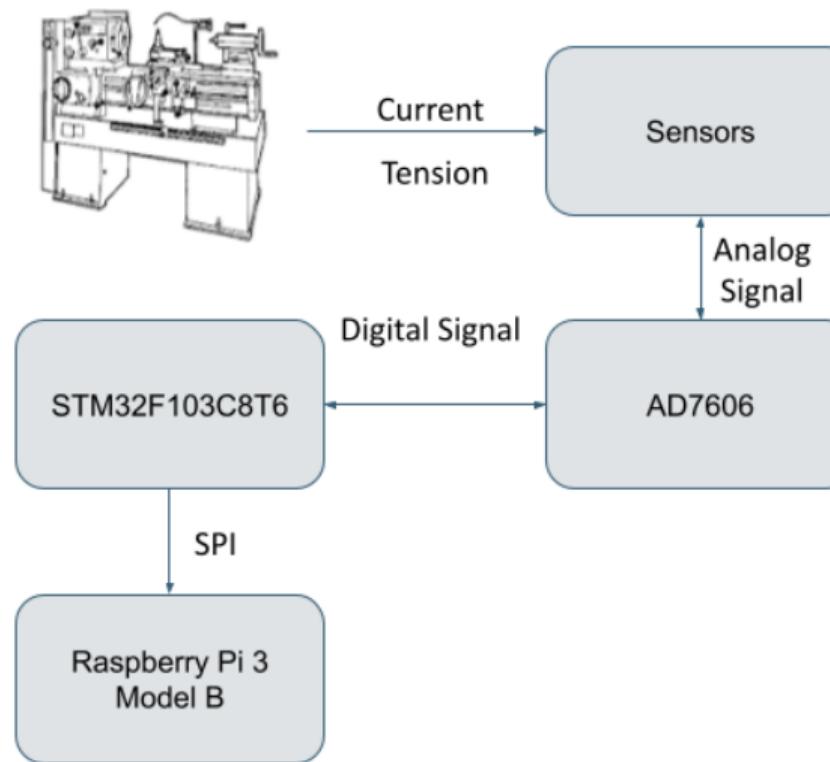
$$\nabla \cdot \mathbf{E} = \frac{\rho}{\epsilon_0}$$

$$\nabla \cdot \mathbf{B} = 0$$

$$\nabla \times \mathbf{E} = - \frac{\partial \mathbf{B}}{\partial t}$$

$$\nabla \times \mathbf{B} = \mu_0 \left(\mathbf{J} + \epsilon_0 \frac{\mathbf{E}}{t} \right)$$

Classificação de ferramentas de torno



Modelos mecanicistas envolvidos:

- Equação do atrito

$$f = \mu N$$

- Filtro de Kalman

$$\mathbf{x}_k = \mathbf{F}_k \mathbf{x}_{k-1} + \mathbf{B}_k \mathbf{u}_k + \mathbf{w}_k$$

No tempo k , uma observação (ou medição) \mathbf{z}_k do estado real \mathbf{x}_k é feita de acordo com

$$\mathbf{z}_k = \mathbf{H}_k \mathbf{x}_k + \mathbf{v}_k$$

1 Apresentação do curso

2 Introdução

Modelos baseado em Dados vs Mecanicistas

Aplicação de Aprendizado de Máquinas

O que é Aprendizado de Máquina?

Treinamento de modelos baseados em dados

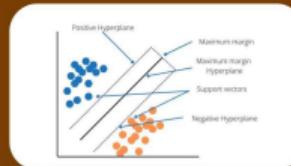
Dilema viés-variância

3 Redes Neurais

O que é Aprendizado de Máquina?

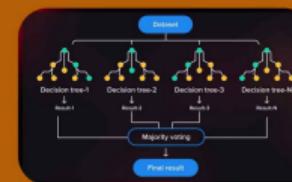
Inteligência Artificial

Qualquer técnica que permita aos computadores imitar a inteligência humana, utilizando lógica, regras do tipo 'if-else', árvores de decisão e aprendizado de máquina.



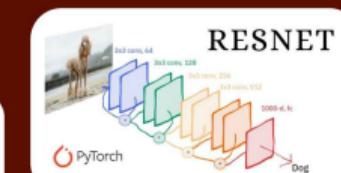
Aprendizado de Máquina

Um subconjunto da IA que inclui técnicas estatísticas complexas que permitem às máquinas melhorar em tarefas com a experiência. Essa categoria inclui o aprendizado profundo.

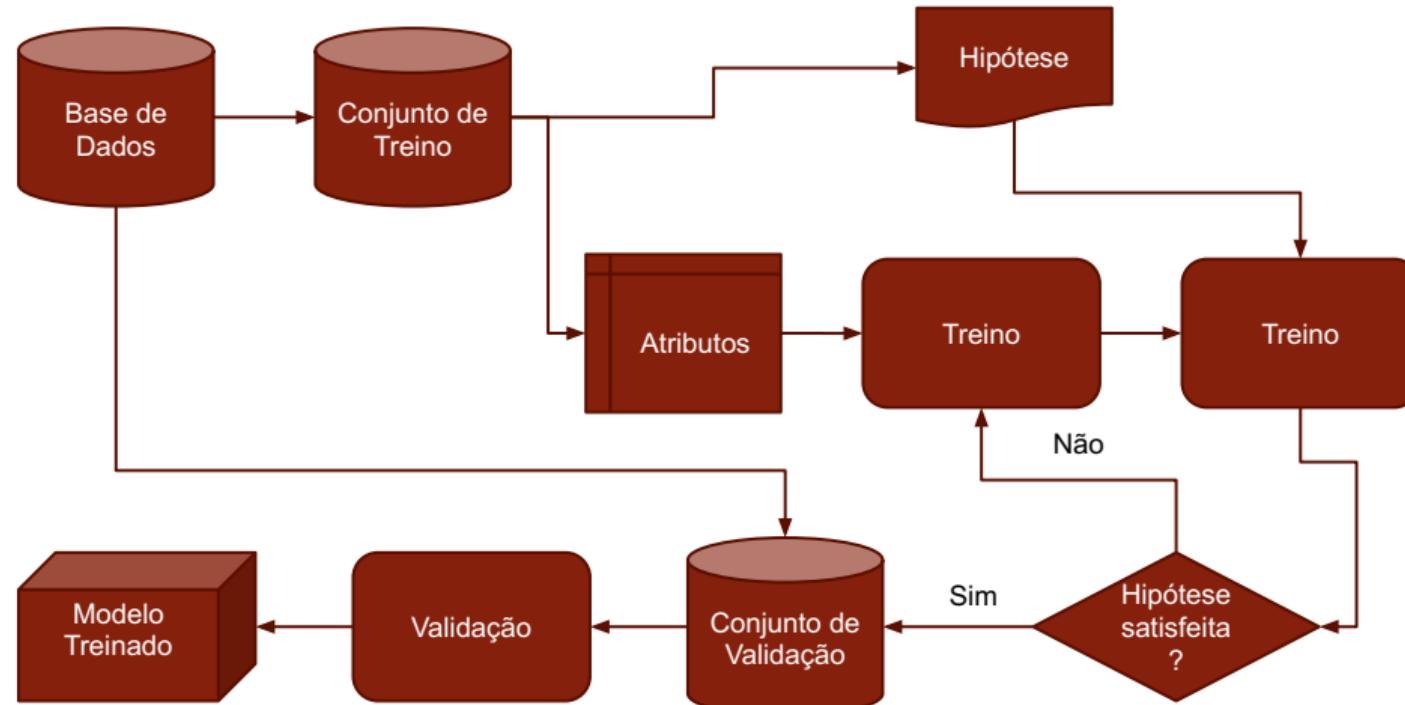


Aprendizado Profundo

O subconjunto do aprendizado de máquina composto por algoritmos que permitem ao software treinar a si mesmo para realizar tarefas, como reconhecimento de fala e imagens, expondo redes neurais multicamadas a vastas quantidades de dados.



Treinamento de modelos baseados em dados



1 Apresentação do curso

② Introdução

Modelos baseado em Dados vs Mecanicistas

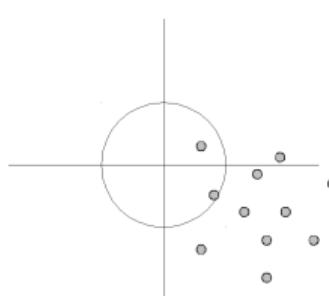
Aplicação de Aprendizado de Máquinas

O que é Aprendizado de Máquina?

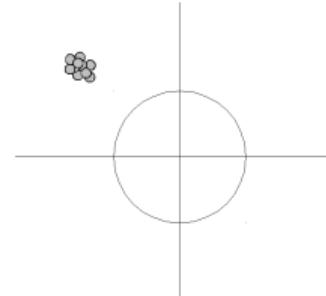
Dilema viés-variância

3 Redes Neurais

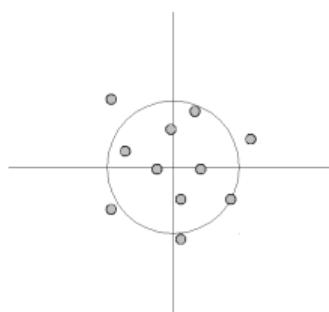
Dilema viés-variância



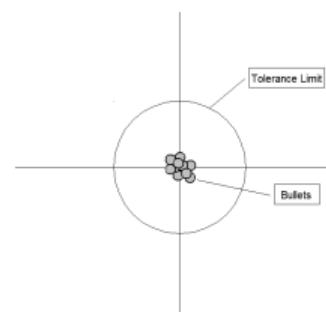
(a) Alto viés, alta variância.



(b) Alto viés, baixa variância



(c) Baixo viés, alta variância.



(d) Baixo viés, baixa variância

O dilema de viés-variância ou problema de viés-variância é o conflito em tentar minimizar simultaneamente essas duas fontes de erro que impedem os algoritmos de aprendizado supervisionado de generalizarem além do seu conjunto de treinamento:

- Erro de viés: é um erro decorrente de suposições equivocadas no algoritmo de aprendizado (underfitting).
 - Variância: é um erro causado pela sensibilidade a pequenas flutuações no conjunto de treinamento (overfitting).

1 Apresentação do curso

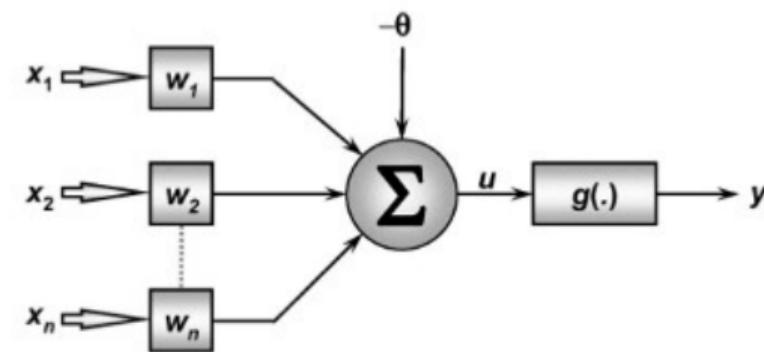
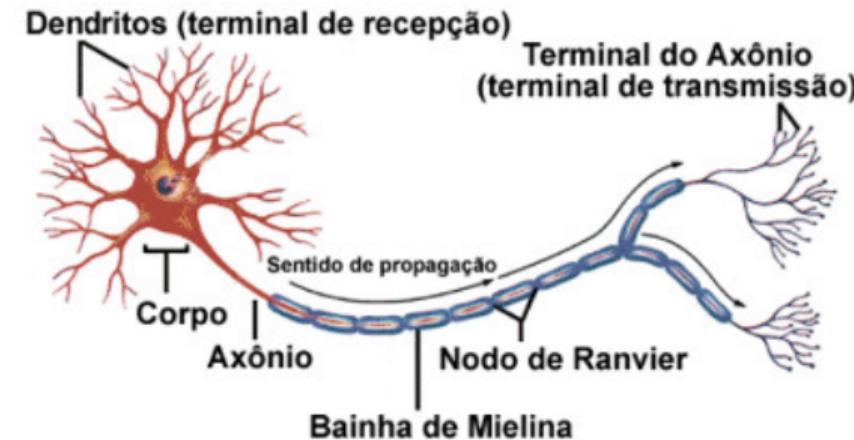
2 Introdução

③ Redes Neurais

Exemplo mais complexo: classificação multiclasse com MNIST

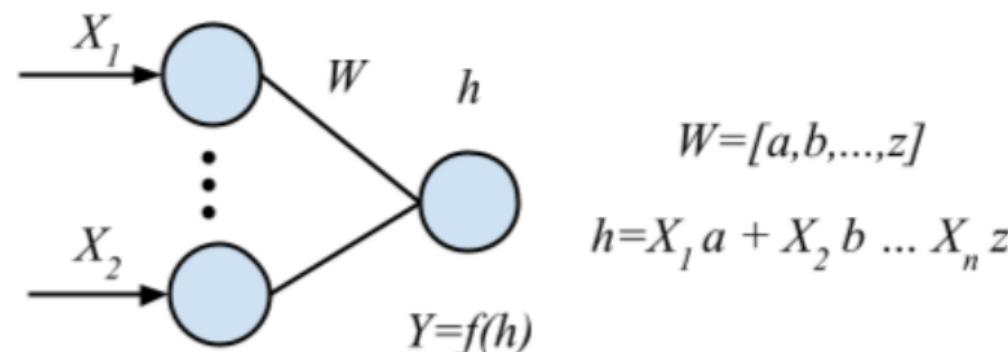
Busca por Regressores: Uma inspiração biológica

- **Funcionamento dos neurônios:** uma inspiração para uma **função base** capaz de expressar relações complexas.
 - Estudo com os neurônios gigantes das lulas revelaram seu funcionamento.
 - Simples, porém quando conectados em redes têm muito poder expressivo!



O Perceptron

- Equivalente a **um neurônio**.
 - Tem **vários** inputs e um output.
 - É simplesmente composto por uma **combinação linear** e uma **função não linear**.



Um Exemplo Simples

Entrada

dados 32x32 -> 1024 inteiros



Saída

The diagram illustrates a linear model. Inputs X_1 through X_{1024} enter nodes. The output h is calculated as a weighted sum of these inputs, where weights $W = [a, b, \dots, z]$. The output $Y = h$ is produced by node h .

$$Y =$$

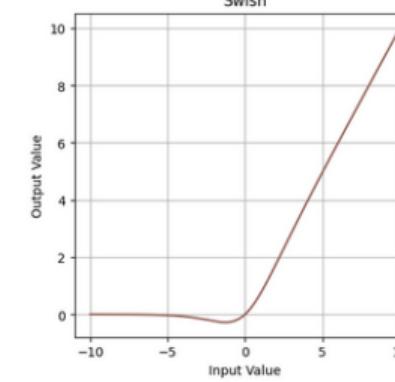
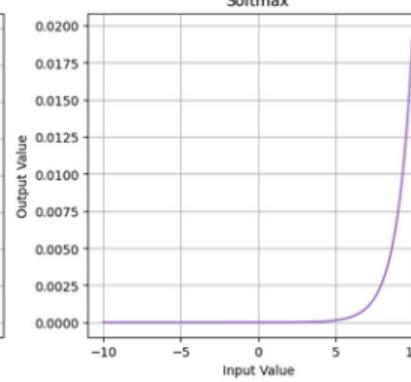
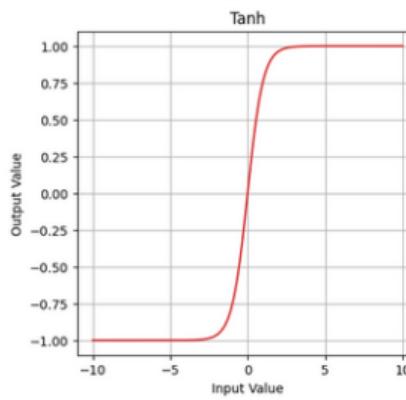
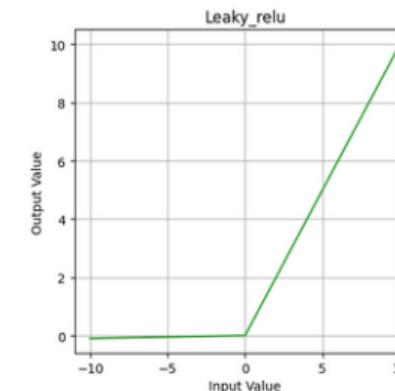
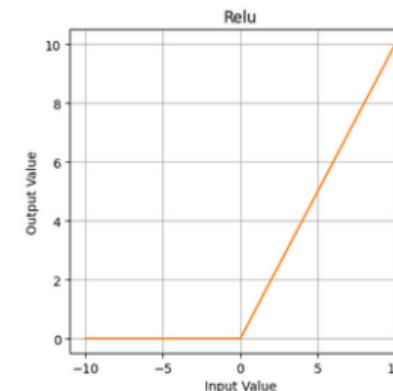
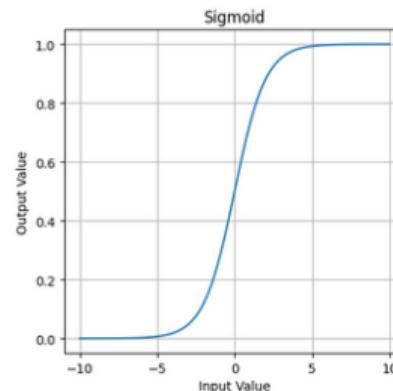
$$W = [a, b, \dots, z]$$

$h = X_1 a + X_2 b + \dots + X_{m+1} z$

Y=

1 se cachorro
-1 se gato

Tipos de função de ativação



Um Exemplo Simples

Devemos encontrar os coeficientes da combinação linear

$$\text{Classificação} = \sum_i^{1024} X_i \cdot W_i \quad (1)$$

Para que:

- quando os X_i pixeis tiverem uma foto de **gato**, classificação = 1
 - quando os X_i pixeis tiverem uma foto de **cachorro**, classificação = -1

Para isso precisamos de dados de treinamento

- Cada entrada, um valor real para cada um dos pixels e 1 ou -1

$$X_0, X_1, X_2, \dots, X_n \equiv 1 \text{ ou } -1 \quad (2)$$

Um Exemplo Simples

Isso se trata de um problema de otimização:

$$\mathbf{AX} = \mathbf{Y} \implies [a_1 \ a_2 \ \cdots \ a_n] \begin{bmatrix} | & | & \cdots & | \\ \mathbf{x}_1 & \mathbf{x}_2 & \cdots & \mathbf{x}_p \\ | & | & \cdots & | \end{bmatrix} = [+1 \ +1 \ \cdots \ -1 \ -1]$$

- Encontre A tal que:

$$\text{Min } AX - Y \tag{3}$$

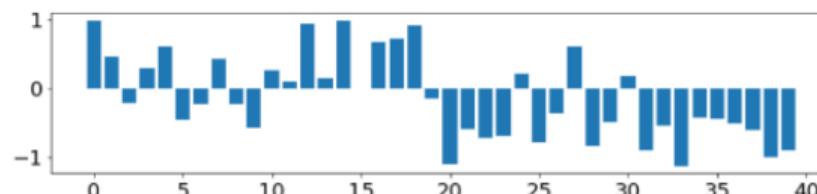
Nesse caso específico A pode ser aproximado com uma pseudo-inversa:

$$A = YX^\dagger \tag{4}$$

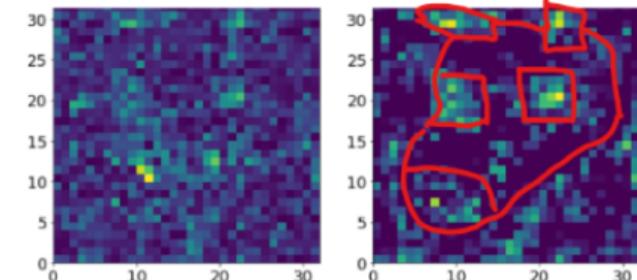
Uma vez encontrado A pode ser usado para prever classificações para fotos novas!

Um Exemplo Simples

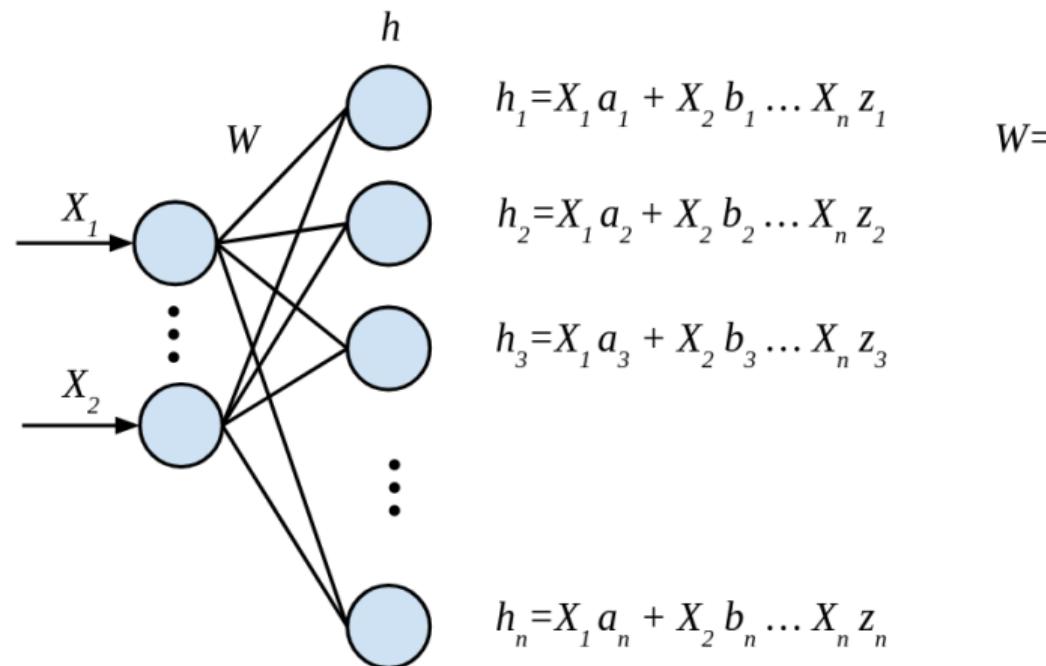
Previsões



Pesos



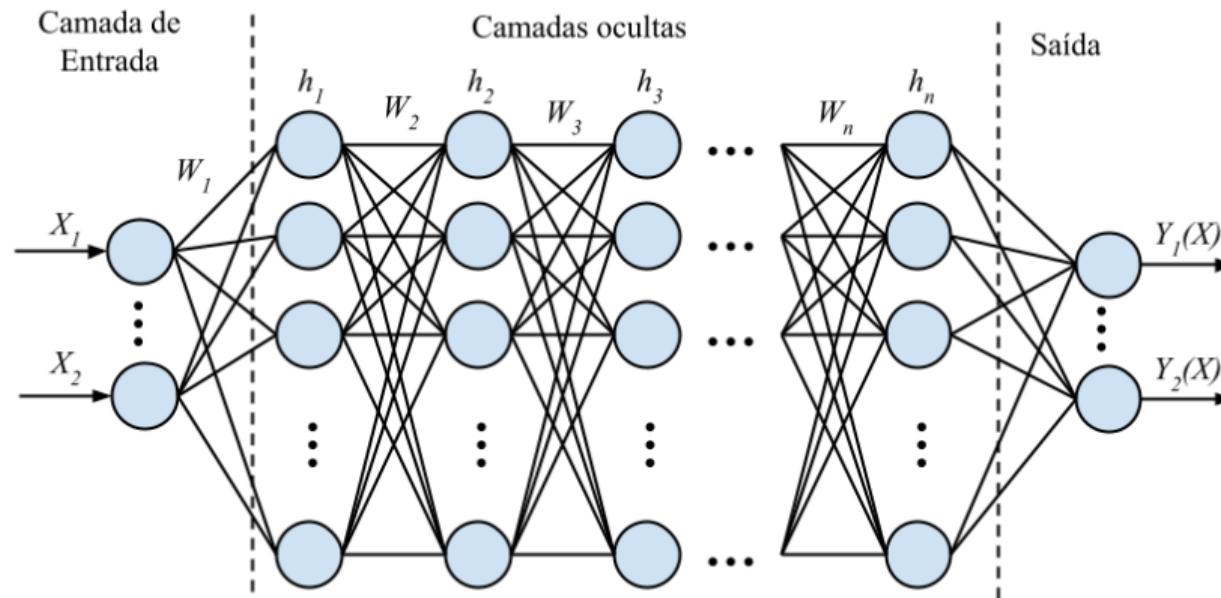
Empilhando Perceptrons



a_1, b_1, \dots, z_1
 a_2, b_2, \dots, z_2
 a_3, b_3, \dots, z_3
⋮
 a_n, b_n, \dots, z_n

$$Y = f(h_1), f(h_2) \dots f(h_n)$$

Múltiplas camadas de Perceptrons Empilhados: Redes MLP



Um caso Especial de Regressor

Redes neurais são um tipo especial de **modelo regressor**, onde a função base é uma **função encadeada, com transformações lineares e ativação não lineares**.

O output da rede neural é obtido passando a entrada por essa cadeia:

$$\mathbf{y} = f(\mathbf{W}^{(L)} \cdot f(\mathbf{W}^{(L-1)} \cdot f(\dots f(\mathbf{W}^{(1)} \cdot \mathbf{x} + \mathbf{b}^{(1)}) + \mathbf{b}^{(2)}) \dots) + \mathbf{b}^{(L-1)}) + \mathbf{b}^{(L)}) \quad (5)$$

Cada elo da cadeia (ou camada da rede) contém uma transformação linear e uma função não linear :

$$f^i(\mathbf{W}^i \cdot \mathbf{x} + b) \quad (6)$$

Treinando Redes Neurais: Um problema de otimização

Queremos encontrar os coeficientes W para cada uma das camadas, que melhor se encaixe nos dados:

$$\text{Para cada entrada } i: y_i = f(\mathbf{W}^{(L)} \cdot f(\mathbf{W}^{(L-1)} \cdot f(\dots f(\mathbf{W}^{(1)} \cdot \mathbf{x} + \mathbf{b}^{(1)}) + \mathbf{b}^{(2)}) \dots) + \mathbf{b}^{(L-1)}) + \mathbf{b}^{(L)}) \quad (7)$$

Uma função perda agrupa as diferenças em cada entrada:

$$\text{MSE} = \left(\frac{1}{n} \sum_{i=1}^n y_i - \hat{y}_i \right)^2 \quad (8)$$

Esse processo de otimização no entanto não pode ser resolvida com pseudo-inversa ou outros métodos **lineares**.

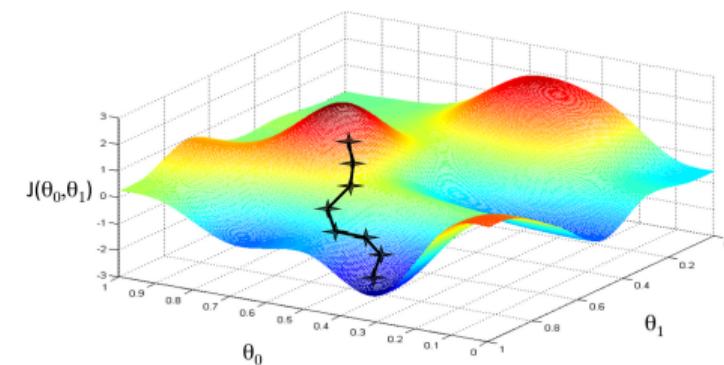
Normalmente, a complexidade da função composta não linear, requer o uso de métodos de otimização especializados como a descida de gradiente!

Metodologia: Otimizadores e Conceitos Básicos de Treinamento

- **Gradiente Descendente (GD):** Minimizar o erro através de atualizações sucessivas nos parâmetros na direção dos gradiente negativo:

$$\theta = \theta - \eta \cdot \nabla_{\theta} \text{Loss}$$

- **Gradiente Descendente Estocástico (SGD):** Versão mais rápida do GD que usa um pequeno lote de dados aleatórios em cada passo.
 - **Otimização Adam:** Combina o GD e o SGD com técnicas de estabilização e aceleração.
 - **Retropropagação:** Método para calcular o gradiente do erro de parâmetro.



Recapitulando: Pipeline de uma Rede Neural

1. Inicialização da Rede Neural e Preparação dos Dados

- **Arquitetura da Rede Neural:** Definir o número de camadas e neurônios.
- **Dados de Treinamento e Validação:**
 - Dividir dados em treino e validação.
 - Escalonar ou normalizar os dados, se necessário.
- **Função de Perda e Otimizador:**
 - Exemplo: Função de Perda MSE ou Cross-Entropy.
 - Otimizadores: SGD, Adam, etc.

2. Loop de Treinamento

- **Forward Pass:** Passar dados de entrada pela rede neural para obter previsões.
- **Cálculo da Perda (Loss):** Comparar previsões com rótulos reais.
- **Backward Pass (Backpropagation):** Calcular os gradientes e propagar o erro para ajustar os pesos.
- **Atualização dos Pesos:** Usar o otimizador para atualizar os parâmetros (pesos) da rede.

3. Inferência e Validação

- **Inferência:** Realizar previsões no conjunto de teste (dados não vistos).
- **Cálculo da Perda de Validação:** Avaliar a performance no conjunto de validação.
- **Visualização dos Resultados:** Gráficos de erros ou curvas de aprendizado.
- **Uso do Modelo:** Para quantificação de incertezas, análise de sensibilidade, etc...

Pytorch

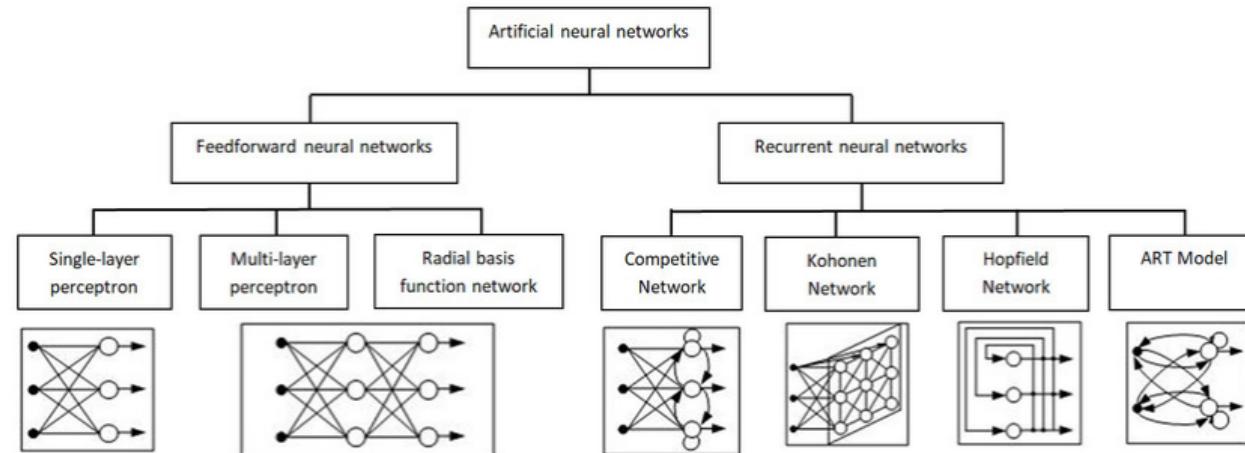
- PyTorch é uma biblioteca poderosa e flexível para redes neurais e aprendizado de máquina.
- Suporta fácil definição de redes neurais, cálculo de gradientes e treinamento de modelos.
- Utiliza tensores, que são arrays multidimensionais, e operações rápidas em GPU.
- Como outras bibliotecas, PyTorch permite a construção, treinamento e validação de modelos de aprendizado profundo.

Exemplo Prático: Mão a massa

- A qualidade de um vinho é descrita com valores inteiros de 0 a 10.
 - Existem inúmeras propriedades químicas que afetam essa qualidade.
 - Com uma tabela com cerca de mil vinhos, suas classificações, e algumas dessas propriedades, vamos usar redes neurais para aprender essa relação!
 - Os modelos treinados serão capazes de qualificar novos vinhos!



Outros Tipos de Arquitetura



Inferências e observações sobre otimização

- Após o treino, o processo de usar a rede em novos dados é denominado inferência.
 - A velocidade de inferência pode ser um métrica importante ou não!
 - A velocidade diminui com o tamanho do modelo.
 - Pytorch permite fazer o processo de treinamento e inferência tanto em gpu quanto em cpus!
 - Gpus são muito mais rápidas!
 - Os modelos treinados podem ser exportados para ferramentas externas que acelaram a inferência.
 - Engines Otimizadoras, Prummers, TensorRT.

1 Apresentação do curso

2 Introdução

③ Redes Neurais

Exemplo mais complexo: classificação multiclasse com MNIST

Base de dados MNIST

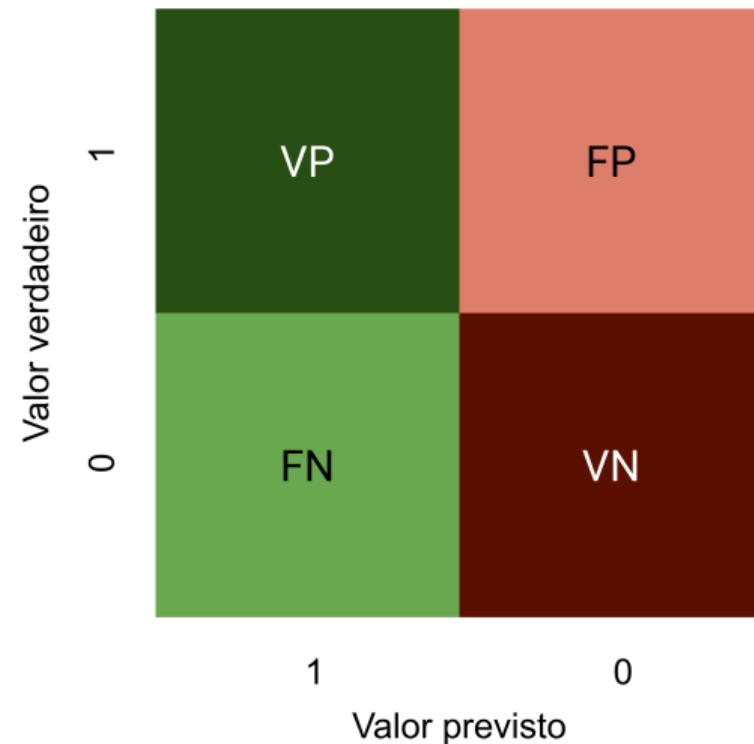
- O banco de dados MNIST (Modified National Institute of Standards and Technology database) é um grande conjunto de dados de dígitos manuscritos amplamente utilizado para o treinamento de diversos sistemas de processamento de imagens.
 - O conjunto de treinamento do é composto por dados de funcionários do Censo dos Estados Unidos.
 - Além disso, as imagens em preto e branco do MNIST foram normalizadas para caber em uma caixa delimitadora de 28x28 pixels e suavizadas com antisserrilhamento, o que introduziu níveis de tons de cinza.



Avaliando modelos de classificação

Uma matriz de confusão é uma ferramenta usada para avaliar o desempenho de um modelo de classificação. Ela organiza os resultados das previsões em uma tabela, comparando os valores previstos com os valores reais. A matriz é composta por quatro principais métricas:

- Verdadeiros Positivos (VP): Casos corretamente classificados como positivos.
 - Falsos Positivos (FP): Casos incorretamente classificados como positivos (falsos alarmes).
 - Verdadeiros Negativos (VN): Casos corretamente classificados como negativos.
 - Falsos Negativos (FN): Casos incorretamente classificados como negativos (erros de omissão).



Avaliando modelos de classificação

$$Preciso = \frac{VP}{VP + FP}$$

$$Recall = \frac{VP}{VP + FN}$$

$$Acurcia = \frac{VP + VN}{VP + VN + FP + FN}$$

$$F1 = 2 \cdot \frac{precision \cdot recall}{precision + recall}$$

