



Machine Learning

Professor: Elton Sarmanho¹

E-mail: eltonss@ufpa.br



¹Faculdade de Sistemas de Informação - UFPA/CUNTINS

31 de janeiro de 2025

Roteiro

Planejamento

Fundamentação

Conceitos Fundamentais

Supervised Learning

Conceitos Fundamentais



Roteiro

Orientações para um Projeto de ML

Refleta sobre o problema e observe o quadro geral

Obtenha os dados

Explore os dados para obter insights

Prepare os dados

Explore muitos modelos e selecione os melhores

Ajuste seus modelos e combine-os

Apresente sua solução

Coloque seu projeto em produção

Model Selection e Optimization

Matrix de Confusão



Roteiro

Fundamentos de Regressão

- Tipos de Modelos de Regressão
- Métricas de Avaliação
- Suposições da Regressão Linear

Redes Neurais

- Funções de Ativação
- Propriedades das Redes Neurais Artificiais
- Arquitetura da Rede Neural
- Perceptron
- RNA
- Multi Layer Perceptron



Roteiro

Unsupervised learning

Conceitos Fundamentais

K-Means

PCA

Referências Bibliográficas



Licença

Este trabalho está licenciado sob a licença Creative Commons:



Nesta aula:

- ▶ Vamos explorar os conceitos fundamentais de aprendizagem de máquina.
- ▶ Ter panorama sobre conceito.
- ▶ Códigos do livro estão: <https://github.com/aimacode>
- ▶ Códigos do Professor estão no github



Conceitos Fundamentais

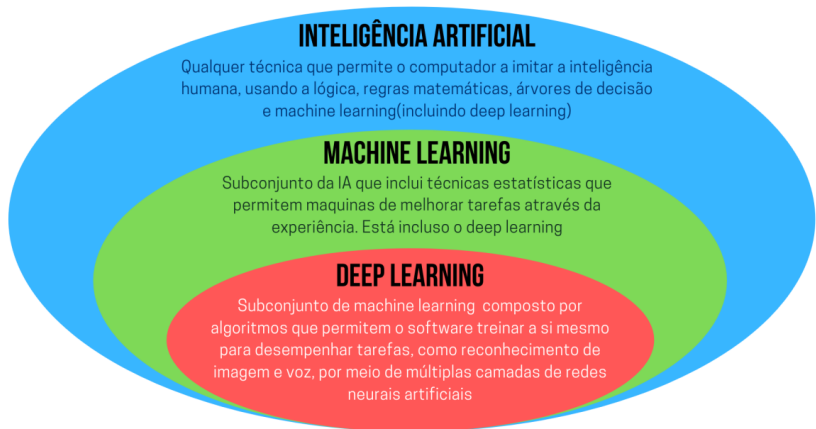


O que é Machine Learning?

- ▶ Machine Learning (ML) é um campo da Inteligência Artificial que se concentra no desenvolvimento de algoritmos que permitem aos computadores aprenderem e fazerem previsões ou decisões baseadas em dados.
- ▶ O aprendizado ocorre a partir de padrões e inferências extraídas de conjuntos de dados, sem serem explicitamente programados.
- ▶ É amplamente utilizado em aplicações como reconhecimento de fala, visão computacional, diagnósticos médicos e sistemas de recomendação.



Machine Learning e IA



História e Motivação

- ▶ O conceito de Machine Learning tem raízes nos campos de estatística, ciência da computação e neurociência.
- ▶ Nos últimos anos, avanços em poder computacional e volume de dados aceleraram o desenvolvimento de técnicas de ML.
- ▶ Motivação principal: permitir que máquinas processem grandes volumes de dados e aprendam automaticamente padrões úteis.



Principais Aplicações

- ▶ Detecção de fraudes financeiras.
- ▶ Reconhecimento de imagem e fala.
- ▶ Sistemas de recomendação (ex.: Netflix, Amazon).
- ▶ Diagnóstico e previsão em saúde.
- ▶ Carros autônomos.

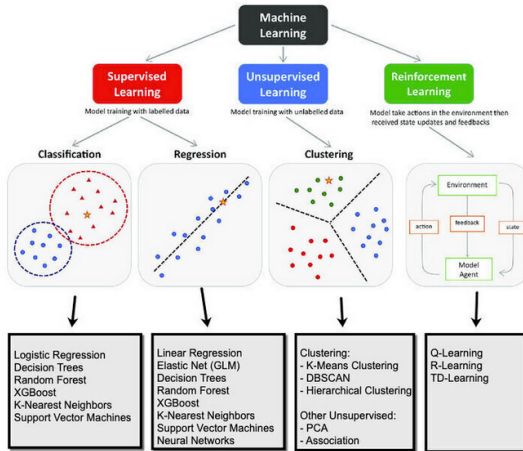


Formas de Aprendizado

- ▶ **Aprendizado Supervisionado:**
 - ▶ O algoritmo é treinado com dados rotulados.
 - ▶ Exemplo: Classificação de imagens em categorias como "gato" ou "cachorro".
- ▶ **Aprendizado Não Supervisionado:**
 - ▶ Não há rótulos; o algoritmo identifica padrões nos dados.
 - ▶ Exemplo: Agrupamento de clientes em segmentos de mercado.
- ▶ **Aprendizado por Reforço:**
 - ▶ O agente aprende interagindo com o ambiente e recebendo recompensas ou penalidades.
 - ▶ Exemplo: Treinar um robô para jogar xadrez.



Formas de Aprendizado



Detalhes sobre Formas de Aprendizado

- ▶ Supervisionado:
 - ▶ Tarefas comuns: regressão e classificação.
 - ▶ Requer um conjunto grande de dados rotulados.
- ▶ Não Supervisionado:
 - ▶ Focado em identificar estrutura ou padrões ocultos.
 - ▶ Métodos: clustering, redução de dimensionalidade.
- ▶ Reforço:
 - ▶ Baseado na ideia de explorar e explorar (explore vs. exploit).
 - ▶ Utilizado em controle robótico e jogos.



Supervised Learning



Aprendizado Supervisionado

- ▶ **Definição:** Um tipo de aprendizado de máquina onde o modelo é treinado utilizando dados rotulados.
- ▶ **Objetivo:** Aprender um mapeamento de entradas (características) para saídas (rótulos).
- ▶ **Aplicações:**
 - ▶ Classificação de imagens
 - ▶ Detecção de spam
 - ▶ Análises preditivas



Componentes Chave

▶ Conjunto de Dados:

- ▶ Características de entrada (X)
- ▶ Rótulos ou valores-alvo (y)

▶ Modelo:

- ▶ Função que mapeia X para y

▶ Treinamento:

- ▶ Processo de ajuste dos parâmetros do modelo para minimizar o erro.

▶ Avaliação:

- ▶ Avaliar o desempenho do modelo usando métricas como acurácia, precisão, etc.



Fluxo de Trabalho

1. Coletar e pré-processar dados rotulados.
2. Dividir os dados em conjuntos de treinamento e teste.
3. Treinar o modelo no conjunto de treinamento.
4. Avaliar o modelo no conjunto de teste.
5. Ajustar o modelo conforme necessário.





Tipos de Aprendizado Supervisionado

► Classificação:

- As saídas são rótulos discretos (ex.: spam ou não spam).

► Regressão:

- As saídas são valores contínuos (ex.: preços de imóveis).

Regression



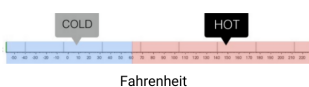
What will be the temperature tomorrow?



Classification



Will it be hot or cold tomorrow?



Underfitting e Overfitting

▶ Underfitting:

- ▶ O modelo é muito simples para capturar a complexidade dos dados.
- ▶ Características principais:
 - ▶ Alta taxa de erro tanto nos dados de treinamento quanto nos de teste.
 - ▶ Modelo insuficiente para aprender os padrões subjacentes.
- ▶ Exemplo: Ajustar uma linha reta em um conjunto de dados não linear.



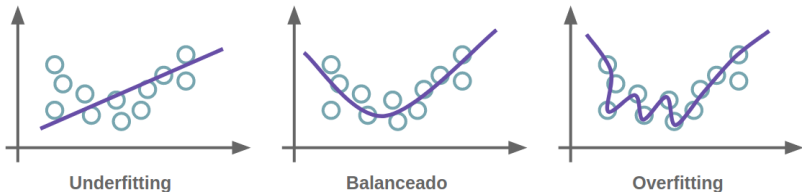
Underfitting e Overfitting

► **Overfitting:**

- O modelo é excessivamente complexo e captura ruído dos dados de treinamento.
- Características principais:
 - Baixa taxa de erro nos dados de treinamento, mas alta nos dados de teste.
 - Modelo foca em detalhes irrelevantes, reduzindo a generalização.
- Exemplo: Ajustar um polinômio de alta ordem em um conjunto pequeno de dados.



Underfitting e Overfitting



Importância de Evitar

- ▶ O objetivo do aprendizado supervisionado é construir um modelo que generalize bem para novos dados.
- ▶ Estratégias para evitar underfitting e overfitting:
 - ▶ Escolher a complexidade adequada do modelo.
 - ▶ Utilizar validação cruzada para avaliar o desempenho do modelo.
 - ▶ Aplicar regularização para reduzir a complexidade do modelo.
 - ▶ Obter mais dados de treinamento de qualidade.



└ Orientações para um Projeto de ML

└ Reflita sobre o problema e observe o quadro geral

1. Reflita sobre o problema e observe o quadro geral

- ▶ Defina o objetivo do ponto de vista do negócio.
- ▶ Como sua solução será usada?
- ▶ Quais são as soluções alternativas atuais?
- ▶ Como enquadrar o problema (supervisionado/não supervisionado, online/offline)?
- ▶ Como o desempenho deve ser medido?
- ▶ A medida de desempenho está alinhada com o objetivo do negócio?
- ▶ Qual seria o desempenho mínimo necessário?
- ▶ O que são problemas comparáveis? Reutilize experiências ou ferramentas.
- ▶ A experiência humana está disponível?
- ▶ Como você resolveria o problema manualmente?
- ▶ Liste e verifique as suposições feitas.



2. Obtenha os dados

- ▶ Liste os dados necessários e a quantidade.
- ▶ Encontre e documente onde obter os dados.
- ▶ Verifique o espaço necessário para armazenamento.
- ▶ Verifique as obrigações legais e obtenha autorizações.
- ▶ Crie um espaço de trabalho com armazenamento suficiente.
- ▶ Certifique-se de que informações confidenciais sejam protegidas.
- ▶ Verifique o tamanho e o tipo dos dados (série temporal, geográfico, etc.).
- ▶ Separe um conjunto de teste e não o utilize durante a exploração.



3. Explore os dados para obter insights

- ▶ Crie uma cópia dos dados para exploração.
- ▶ Use um Notebook Jupyter para documentar a exploração.
- ▶ Estude cada atributo:
 - ▶ Nome, tipo, % de valores ausentes, ruído, utilidade, distribuição.
- ▶ Para tarefas supervisionadas, identifique o(s) atributo(s) alvo.
- ▶ Visualize os dados.
- ▶ Estude correlações entre atributos.
- ▶ Identifique transformações promissoras.
- ▶ Documente o que foi aprendido.



4. Prepare os dados

- ▶ Trabalhe em cópias dos dados (mantenha o original intacto).
- ▶ Grave funções para todas as transformações de dados.
- ▶ Limpeza de dados:
 - ▶ Corrija ou remova outliers.
 - ▶ Preencha valores ausentes (zero, média, mediana) ou descarte linhas/colunas.
- ▶ Seleção de atributos:
 - ▶ Descarte atributos que não fornecem informações úteis.
- ▶ Engenharia de atributos:
 - ▶ Decomponha atributos (categóricos, data/hora).
 - ▶ Adicione transformações promissoras (log, sqrt, etc.).
 - ▶ Agregue atributos em novos atributos promissores.
- ▶ Escalonamento de atributos:
 - ▶ Padronize ou normalize atributos.



5. Explore muitos modelos e selecione os melhores

- ▶ Treine modelos de diferentes categorias (linear, SVM, Random Forest, Redes Neurais, etc.).
- ▶ Avalie e compare o desempenho usando validação cruzada.
- ▶ Analise as variáveis mais significativas para cada algoritmo.
- ▶ Analise os tipos de erros cometidos pelos modelos.
- ▶ Execute uma rodada rápida de seleção e engenharia de atributos.
- ▶ Liste os três a cinco modelos mais promissores.



6. Ajuste seus modelos e combine-os

- ▶ Ajuste os hiperparâmetros usando validação cruzada.
- ▶ Trate as transformações de dados como hiperparâmetros.
- ▶ Prefira busca aleatória em vez de busca em grade.
- ▶ Experimente métodos de ensemble (combinação de modelos).
- ▶ **Atenção:** Não ajuste o modelo após medir o erro de generalização.



7. Apresente sua solução

- ▶ Documente o que foi feito.
- ▶ Crie uma boa apresentação, destacando o quadro geral.
- ▶ Explique por que a solução atinge o objetivo de negócios.
- ▶ Apresente pontos interessantes observados durante o projeto.
- ▶ Liste suposições e limitações do sistema.
- ▶ Use visualizações para comunicar descobertas importantes.



└ Orientações para um Projeto de ML

└ Coloque seu projeto em produção

8. Coloque seu projeto em produção

- ▶ Prepare a solução para produção (conecte-a às entradas de dados, escreva testes).
- ▶ Escreva código de monitoramento para verificar o desempenho ao vivo.
- ▶ Monitore a qualidade das entradas (sensores defeituosos, dados obsoletos).
- ▶ Treine modelos regularmente com dados atualizados (automatize o máximo possível).



└ Orientações para um Projeto de ML

└ Coloque seu projeto em produção

Como funciona na prática

- ▶ github.com/eltonsarmanho/InteligenciaArtificial
 - ▶ Dataset
 - ▶ k-Nearest Neighbors
 - ▶ Decision Tree
 - ▶ Random Forest
 - ▶ NAIVE BAYES
 - ▶ ENSEMBLE LEARNER
 - ▶ SVM
- ▶ Grid Search
- ▶ Cross-Validation
- ▶ Curva de Aprendizado



Model Selection Optimization



Seleção de Modelos (Model Selection)

Objetivo: Escolher o melhor modelo ou espaço de hipóteses para um problema de aprendizado de máquina.

- ▶ **Model Selection:** Escolha de um espaço de hipóteses (e.g., árvores de decisão, redes neurais, etc.).
- ▶ **Otimização:** Encontrar a melhor hipótese dentro do espaço escolhido.

Desafio: Equilibrar underfitting e overfitting.



Model Selection

Objetivo da seleção de modelos:

- ▶ Identificar o modelo que melhor generaliza para dados não vistos.

Etapas principais:

1. Divisão dos dados:

- ▶ *Training set*: para ajustar os parâmetros do modelo.
- ▶ *Validation set*: para ajustar hiperparâmetros.
- ▶ *Test set*: para avaliar a performance final.

2. Validação cruzada (Cross-validation):

- ▶ Técnica para maximizar o uso dos dados disponíveis e reduzir viés.

3. Escolha de métricas:

- ▶ Exemplo: Acurácia, precisão, recall, F1-score para classificação.



Cross-validation

Definição:

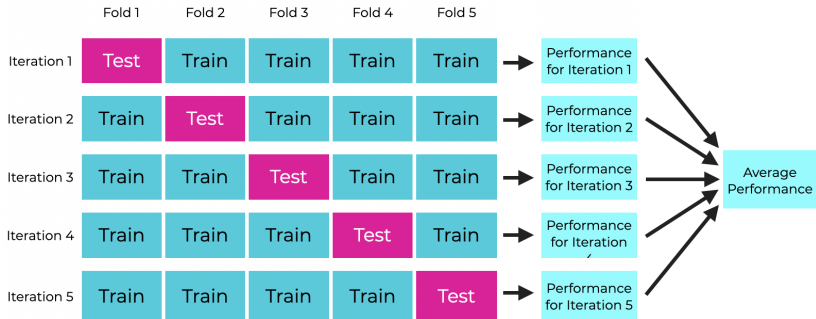
- ▶ Divide os dados em k subconjuntos (*folds*).
- ▶ Treina e avalia o modelo k vezes, cada vez utilizando um fold diferente como validação.

Vantagens:

- ▶ Usa os dados de forma eficiente.
- ▶ Reduz a variação associada à divisão específica de dados.



Cross-validation

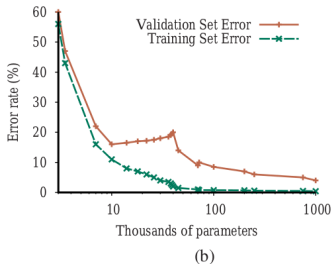


Curva de Aprendizado

- ▶ **Definição:** Representação gráfica que mostra como o desempenho de um modelo muda com o aumento de dados de treinamento.
- ▶ **Componentes:**
 - ▶ **Erro de treinamento:** Diminui à medida que o modelo aprende melhor os padrões dos dados.
 - ▶ **Erro de validação:** Inicialmente diminui, mas pode aumentar devido ao *overfitting*.
- ▶ **Objetivo:** Utilizar a curva para identificar problemas como *underfitting* (alta taxa de erro) ou *overfitting* (grande diferença entre erros de treinamento e validação).



Exemplo de Curva de Aprendizado



Curva de Aprendizado: Mostra o erro no conjunto de treinamento e validação em função do tamanho do conjunto de treinamento.

- ▶ **Underfitting:** Erro alto em ambos os conjuntos.
- ▶ **Overfitting:** Erro baixo no treinamento, mas alto na validação.



Hyperparameter Tuning

Definição:

- ▶ Parâmetros que não são aprendidos diretamente pelo modelo.
- ▶ Exemplo: número de árvores em uma floresta aleatória.

Técnicas comuns:

- ▶ *Grid search*: Busca exaustiva por combinações de hiperparâmetros.
- ▶ *Random search*: Seleção aleatória de combinações.
- ▶ *Bayesian optimization*: Abordagem probabilística para encontrar a melhor configuração.



Matrix de Confusão



O que é uma Matriz de Confusão?

Matriz de Confusão:

- ▶ Ferramenta para avaliar o desempenho de modelos de classificação.
- ▶ Compara as previsões do modelo com os valores reais (rótulos verdadeiros).
- ▶ Útil para problemas de classificação binária ou multiclasse.

Estrutura Básica:

	Previsto Positivo	Previsto Negativo
Real Positivo	Verdadeiro Positivo (VP)	Falso Negativo (FN)
Real Negativo	Falso Positivo (FP)	Verdadeiro Negativo (VN)

Exemplo de Matriz de Confusão

Cenário: Classificação de e-mails como **spam** ou **não spam**.

	Previsto Spam	Previsto Não Spam
Real Spam	50 (VP)	10 (FN)
Real Não Spam	5 (FP)	100 (VN)

Legenda:

- ▶ **VP (Verdadeiro Positivo):** E-mails corretamente classificados como spam.
- ▶ **FN (Falso Negativo):** E-mails spam classificados como não spam.
- ▶ **FP (Falso Positivo):** E-mails não spam classificados como spam.
- ▶ **VN (Verdadeiro Negativo):** E-mails corretamente classificados como não spam.



Métricas Derivadas da Matriz de Confusão

1. Acurácia (Accuracy):

$$\text{Acurácia} = \frac{VP + VN}{VP + FP + FN + VN}$$

Exemplo: $\frac{50+100}{50+5+10+100} = \frac{150}{165} \approx 90.9\%$.

2. Precisão (Precision):

$$\text{Precisão} = \frac{VP}{VP + FP}$$

Exemplo: $\frac{50}{50+5} = \frac{50}{55} \approx 90.9\%$.

3. Recall (Sensibilidade):

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

Exemplo: $\frac{50}{50+10} = \frac{50}{60} \approx 83.3\%$.

4. F1-Score:

$$\text{F1-Score} = 2 \cdot \frac{\text{Precisão} \cdot \text{Recall}}{\text{Precisão} + \text{Recall}}$$

Exemplo: $2 \cdot \frac{0.909 \cdot 0.833}{0.909 + 0.833} \approx 0.869$.



Interpretação da Matriz de Confusão

1. Underfitting (Subajuste):

- ▶ Alto erro tanto no treinamento quanto na validação.
- ▶ Modelo muito simples para capturar os padrões dos dados.

2. Overfitting (Sobreajuste):

- ▶ Baixo erro no treinamento, mas alto erro na validação.
- ▶ Modelo se ajusta demais aos dados de treinamento e não generaliza bem.

3. Bom Ajuste:

- ▶ Baixo erro tanto no treinamento quanto na validação.
- ▶ Modelo generaliza bem para novos dados.



Matriz de Confusão para Classificação Multiclasse

Exemplo: Problema com três classes (**A**, **B**, **C**).

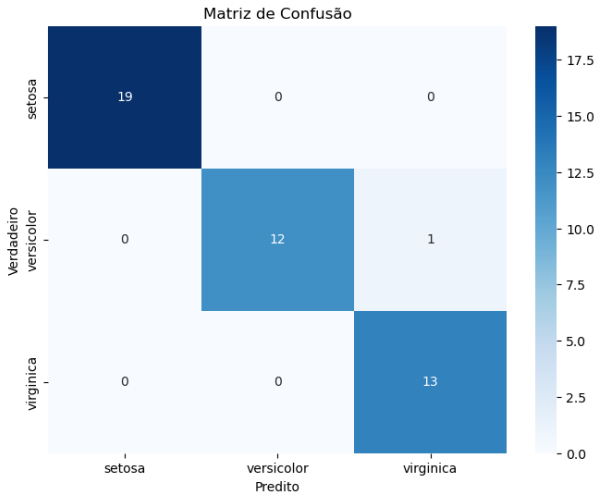
	Previsto A	Previsto B	Previsto C
Real A	VP_A	FN_A→B	FN_A→C
Real B	FN_B→A	VP_B	FN_B→C
Real C	FN_C→A	FN_C→B	VP_C

Legenda:

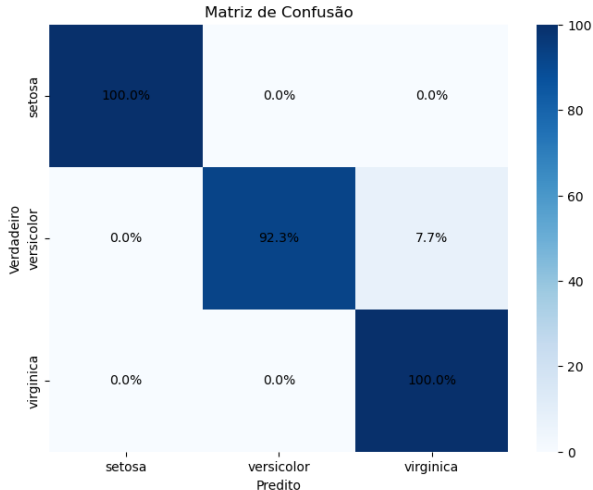
- ▶ **VP_X:** Verdadeiros positivos para a classe X.
- ▶ **FN_X→Y:** Falsos negativos onde a classe real era X, mas foi prevista como Y.



Matriz de Confusão para Classificação Multiclasse



Matriz de Confusão para Classificação Multiclasse



Quando Usar a Matriz de Confusão?

Aplicações:

- ▶ Avaliação de modelos de classificação.
- ▶ Identificação de desbalanceamento de classes.
- ▶ Ajuste de limiares de decisão para equilibrar precisão e recall.



Regressão



Definição

Modelo estatístico para prever uma variável contínua y a partir de uma ou mais variáveis de entrada X .

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_n x_n + \varepsilon$$

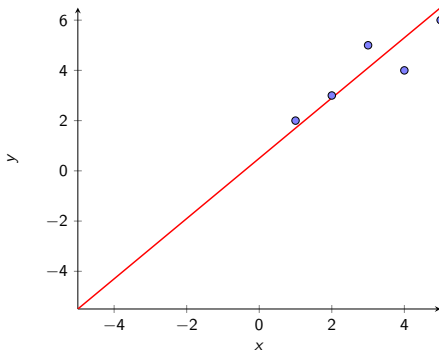


Figura: Exemplo de regressão linear.



Regressão vs. Classificação

Característica	Regressão	Classificação
Saída	Contínua (\mathbb{R})	Discreta (rótulos)
Exemplo	Preço de imóveis	Diagnóstico médico (doente/saudável)
Métricas	MSE, R^2	Acurácia, F1-Score



Variáveis Independentes vs. Dependentes

Formulação Matemática

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon}$$

Onde:

- ▶ $\mathbf{X} \in \mathbb{R}^{n \times p}$: Matriz de features
- ▶ $\boldsymbol{\beta} \in \mathbb{R}^p$: Vetor de coeficientes
- ▶ $\boldsymbol{\varepsilon} \in \mathbb{R}^n$: Erro aleatório

Variáveis Independentes (X)

$\mathbf{X}\boldsymbol{\beta}$

Modelo (f)

\hat{y}

Variável Dependente (y)



Regressão Linear Simples

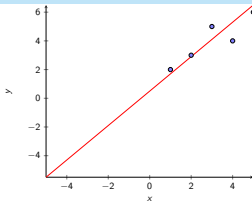
Definição

Modela a relação linear entre uma variável independente x e uma variável dependente y :

$$y = \beta_0 + \beta_1 x + \varepsilon$$

Onde:

- ▶ β_0 : Intercepto (valor de y quando $x = 0$).
- ▶ β_1 : Coeficiente angular (inclinação da reta).
- ▶ ε : Erro aleatório.



Regressão Linear Múltipla

Definição

Estende a regressão linear para múltiplas variáveis independentes:

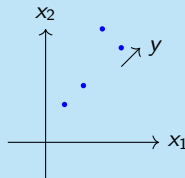
$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_n x_n + \varepsilon$$

Em forma matricial:

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon}$$

Onde:

- ▶ \mathbf{X} : Matriz de features ($n \times p$).
- ▶ $\boldsymbol{\beta}$: Vetor de coeficientes ($p \times 1$).



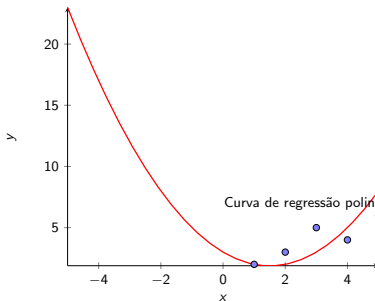
Regressão Polinomial

Definição

Modela relações não lineares usando polinômios:

$$y = \beta_0 + \beta_1 x + \beta_2 x^2 + \cdots + \beta_n x^n + \varepsilon$$

Exemplo: $y = \beta_0 + \beta_1 x + \beta_2 x^2$.



Regressão Regularizada

Objetivo

Penalizar coeficientes grandes para evitar overfitting:

- ▶ Ridge (L2):

$$\text{Minimizar } \|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|^2 + \lambda \|\boldsymbol{\beta}\|^2$$

- ▶ Lasso (L1):

$$\text{Minimizar } \|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|^2 + \lambda \|\boldsymbol{\beta}\|_1$$

- ▶ Elastic Net:

$$\text{Minimizar } \|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|^2 + \lambda_1 \|\boldsymbol{\beta}\|_1 + \lambda_2 \|\boldsymbol{\beta}\|^2$$



Regressão Logística (Contexto de Regressão)

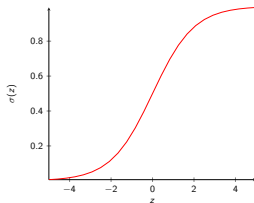
Definição

Usada para problemas de classificação binária, mas fundamentada em regressão:

$$P(y = 1|\mathbf{x}) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x_1 + \dots + \beta_n x_n)}}$$

Onde:

- ▶ $P(y = 1|\mathbf{x})$: Probabilidade de $y = 1$.
- ▶ Função logística: $\sigma(z) = \frac{1}{1+e^{-z}}$.



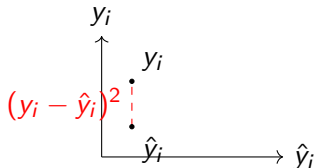
Erro Quadrático Médio (MSE)

Definição: O Erro Quadrático Médio é uma métrica que mede a média dos erros ao quadrado entre os valores reais (y_i) e os valores preditos (\hat{y}_i):

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

Interpretação: Quanto menor o MSE, melhor a performance do modelo.

Visualização:



Raiz do Erro Quadrático Médio (RMSE)

Definição: A Raiz do Erro Quadrático Médio é simplesmente a raiz quadrada do MSE:

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

Interpretação: É uma métrica na mesma unidade dos dados originais, tornando sua interpretação mais intuitiva.



Erro Absoluto Médio (MAE)

Definição: O Erro Absoluto Médio mede a média das diferenças absolutas entre os valores reais e preditos:

$$\text{MAE} = \frac{1}{n} \sum_{i=1}^n |y_i - \hat{y}_i| \quad (3)$$

Interpretação: Valores menores indicam que o modelo está mais próximo dos valores reais em média.



Coeficiente de Determinação (R^2) e R^2 Ajustado

Coeficiente de Determinação (R^2):

- ▶ Mede a proporção da variância explicada pelo modelo.
- ▶ Fórmula:

$$R^2 = 1 - \frac{\sum_{i=1}^n (y_i - \hat{y}_i)^2}{\sum_{i=1}^n (y_i - \bar{y})^2} \quad (4)$$

- ▶ Intervalo: $0 \leq R^2 \leq 1$ (quanto mais próximo de 1, melhor).

R^2 Ajustado:

- ▶ Penaliza a adição de variáveis irrelevantes ao modelo.
- ▶ Fórmula:

$$R^2_{\text{ajustado}} = 1 - (1 - R^2) \frac{n - 1}{n - p - 1} \quad (5)$$

- ▶ Onde p é o número de variáveis explicativas.



Linearidade

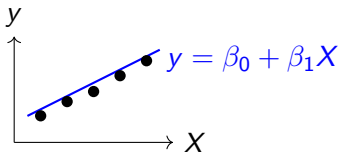
Definição: A relação entre a variável dependente (y) e a variável independente (X) deve ser linear.

Representação Matemática:

$$y = \beta_0 + \beta_1 X + \epsilon \quad (6)$$

Onde:

- ▶ β_0 é o intercepto;
- ▶ β_1 é o coeficiente angular;
- ▶ ϵ representa o termo de erro.



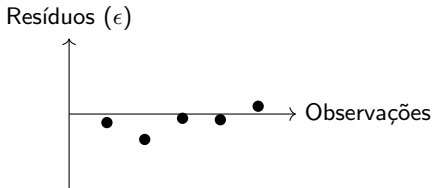
Independência dos Resíduos

Definição: Os resíduos (ϵ_i) não devem estar correlacionados entre si.

Teste Comum: Teste de Durbin-Watson

$$DW = \frac{\sum_{i=2}^n (\epsilon_i - \epsilon_{i-1})^2}{\sum_{i=1}^n \epsilon_i^2} \quad (7)$$

Valores próximos de 2 indicam independência.

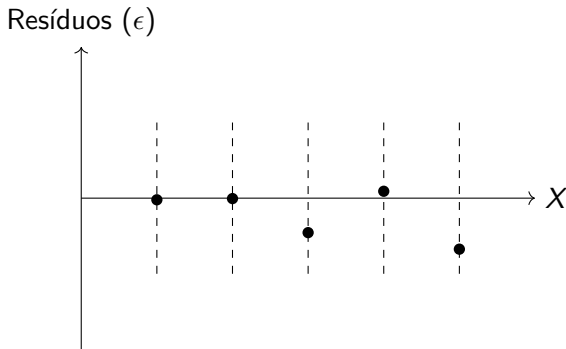


Visualização: Pontos aleatórios indicam independência



Homocedasticidade

Definição: A variância dos resíduos deve ser constante para todos os valores de X .

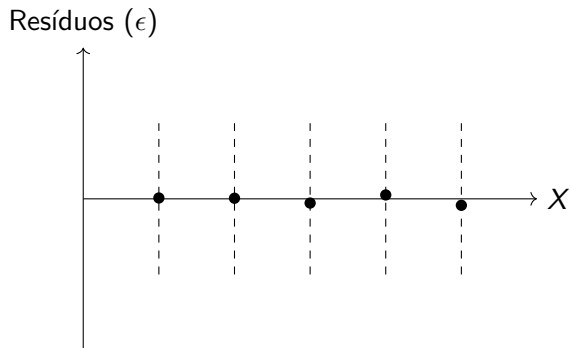


Visualização:



Homocedasticidade

Interpretacao: A ausência de padrões nos resíduos (em torno do eixo x) indica que o modelo atende à suposição de homocedasticidade.



Visualização:



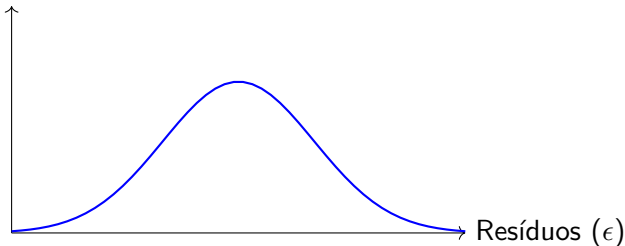
Normalidade dos Resíduos

Definição: Os resíduos devem seguir uma distribuição aproximadamente normal.

Teste Comum: Teste de Shapiro-Wilk

Visualização:

Frequência



Ausência de Multicolinearidade

Definição: As variáveis independentes não devem ser altamente correlacionadas entre si.

Teste Comum: Fator de Inflacionamento da Variância (VIF)

$$VIF_j = \frac{1}{1 - R_j^2} \quad (8)$$

Onde R_j^2 é o coeficiente de determinação da regressão da variável j contra as demais.

Interpretação: $VIF > 10$ indica alta multicolinearidade.



Redes Neurais



Objetivos da Apresentação

- ▶ Explorar as características fundamentais das Redes Neurais Artificiais (RNAs).
- ▶ Compreender o funcionamento básico de uma RNA.
- ▶ Analisar as principais arquiteturas de RNAs disponíveis.
- ▶ Ilustrar aplicações de RNAs em Reconhecimento de Padrões e Controle.



Motivação Biológica

Ideia Central

- ▶ Utilizar neurônios biológicos como modelos para neurônios artificiais.
- ▶ Neurônios biológicos são o elemento fundamental do sistema nervoso.
- ▶ Existem diversos tipos de neurônios biológicos.



Funcionamento Simplificado de um Neurônio Biológico

- ▶ Neurônios podem estar em dois estados:
 - ▶ **Ativo ou excitado:** Envia sinais para outros neurônios por meio do axônio e sinapses.
 - ▶ **Inativo ou inibido:** Não envia sinais.
- ▶ Sinapses podem ser de dois tipos:
 - ▶ **Excitatorias:** Excitam o neurônio receptor.
 - ▶ **Inibitórias:** Inibem o neurônio receptor.



Ativação do Neurônio

- ▶ Quando o efeito cumulativo das várias sinapses que chegam a um neurônio excede um valor limite, o neurônio dispara.
- ▶ O neurônio fica ativo por um período e envia um sinal para outros neurônios.



Introdução às Redes Neurais Artificiais

- ▶ Redes neurais artificiais são modelos computacionais inspirados no funcionamento do cérebro humano.
- ▶ Esses modelos são amplamente utilizados em aprendizado de máquina e inteligência artificial.
- ▶ As redes neurais foram desenvolvidas a partir de contribuições de diversas áreas, como matemática aplicada, estatística e ciência da computação.



Origens das Redes Neurais Artificiais

- ▶ Os primeiros estudos formais sobre redes neurais artificiais foram realizados por McCulloch e Pitts em 1943.
- ▶ Eles propuseram um modelo matemático para um neurônio artificial.
- ▶ Esse modelo era capaz de implementar operações lógicas básicas, como "E" e "OU".



Modelo Matemático do Neurônio Artificial

- ▶ Um neurônio artificial pode ser representado por uma função $\eta : \mathbb{R}^n \rightarrow \{0, 1\}$.
- ▶ A função é definida pela equação:

$$\eta(x) = \chi_{\geq 0} \left[w^T x - b \right],$$

onde:

- ▶ w é o vetor de pesos,
- ▶ x é o vetor de entradas,
- ▶ b é o limiar de ativação,
- ▶ $\chi_{\geq 0}$ é a função indicadora que retorna 1 se o argumento for maior ou igual a zero, e 0 caso contrário.



Aplicações Iniciais

- ▶ O modelo de McCulloch e Pitts demonstrou que redes neurais podem resolver problemas de lógica clássica.
- ▶ Essa descoberta foi um marco inicial para o desenvolvimento de sistemas de inteligência artificial.
- ▶ Hoje, as redes neurais são aplicadas em diversas áreas, como reconhecimento de padrões, processamento de linguagem natural e visão computacional.



Neurônio Artificial

- ▶ Um neurônio artificial é uma função $\eta : \mathbb{R}^n \rightarrow \mathbb{R}$.
- ▶ A função é definida por:

$$\eta(x) = \varphi(w^T x - b),$$

onde:

- ▶ φ é a função de ativação,
- ▶ w é o vetor de pesos sinápticos,
- ▶ b é o viés.



- A função de ativação φ determina a saída do neurônio.

Função Limiar

$$\chi_{\geq 0}(t) = \begin{cases} 1, & t \geq 0, \\ 0, & t < 0. \end{cases}$$

Função Logística

$$\sigma(t) = \frac{1}{(1 + e^{-t})}.$$

Função ReLU

$$\text{relu}(t) = \max\{0, t\}.$$



Aplicações das Funções de Ativação

- ▶ A função limiar é usada em modelos binários.
- ▶ A função logística é comum em problemas de classificação.
- ▶ A função ReLU é amplamente utilizada em redes neurais profundas.



Propriedades das Redes Neurais Artificiais

- ▶ As redes neurais artificiais (RNAs) possuem características que as tornam poderosas para diversas aplicações.
- ▶ Entre as principais propriedades estão: aprendizado, generalização e abstração.



Aprendizado

- ▶ As RNAs são capazes de modificar seu comportamento com base nos dados de entrada.
- ▶ Isso permite que elas produzam saídas consistentes e adaptadas ao domínio do problema.
- ▶ O aprendizado ocorre através da ajuste dos pesos sinápticos durante o treinamento.



Generalização

- ▶ Após o treinamento, uma RNA pode lidar com pequenas variações nas entradas, como ruídos ou distorções.
- ▶ Essa capacidade de generalização é essencial para aplicações em cenários do mundo real.
- ▶ A generalização permite que a rede mantenha um bom desempenho mesmo com dados não vistos durante o treinamento.



Arquitetura de Redes Neurais

- ▶ Os neurônios artificiais são as unidades básicas de processamento em uma rede neural.
- ▶ Uma rede neural pode ser representada como um grafo direcionado.
- ▶ Nesse grafo, os vértices correspondem aos neurônios e as arestas indicam conexões entre eles.



Topologia da Rede Neural

- ▶ A estrutura do grafo é chamada de arquitetura ou topologia da rede neural.
- ▶ A topologia define como os neurônios estão organizados e conectados.
- ▶ Existem diferentes tipos de topologias, como redes neurais recorrentes e progressivas.

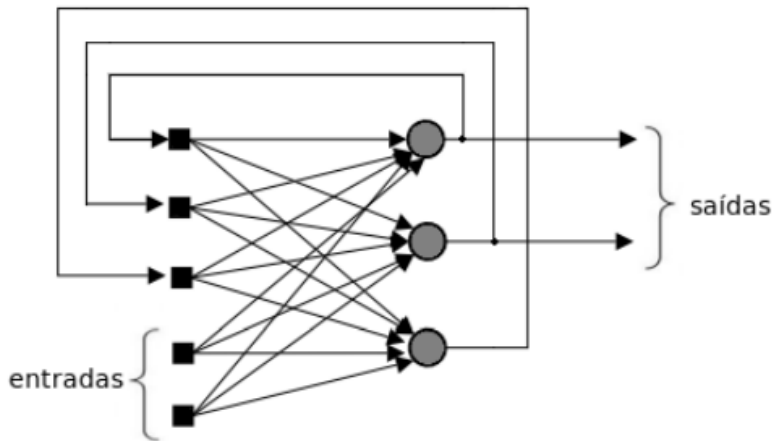


Redes Neurais Recorrentes ou *Feed-backward networks*

- ▶ Uma rede neural é dita recorrente quando o grafo associado contém ciclos.
- ▶ Isso permite que a informação flua em loops, o que é útil para modelar sequências temporais.
- ▶ Aplicações comuns incluem processamento de linguagem natural e previsão de séries temporais.



Redes Neurais Recorrentes ou *Feed-backward networks*

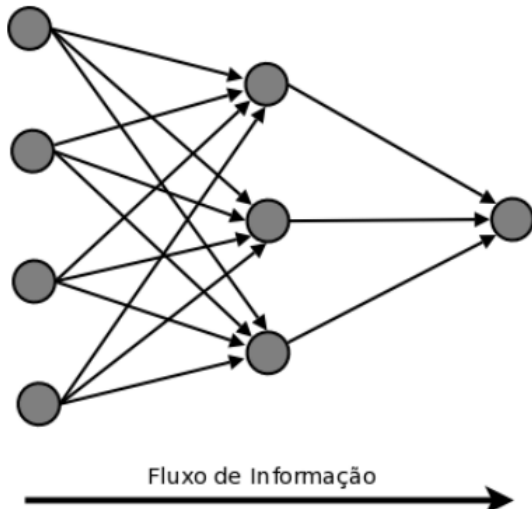


Redes Neurais Progressivas ou *Feed-forward networks*

- ▶ Uma rede neural é progressiva quando o fluxo de informação segue em uma única direção, sem ciclos.
- ▶ Também conhecidas como redes feedforward, são amplamente utilizadas em tarefas de classificação e regressão.
- ▶ Exemplos incluem redes neurais convolucionais (CNNs) e perceptrons multicamadas.



Redes Neurais Progressivas ou *Feed-forward networks*



Modelo Matemático do Perceptron

- O Perceptron é uma função $f : \mathbb{R}^n \rightarrow \{0, 1\}$ definida por:

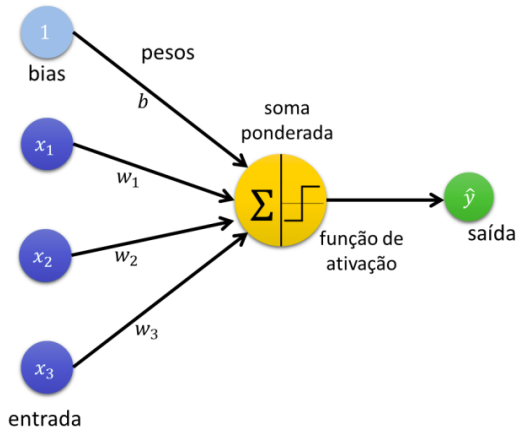
$$f(x) = \begin{cases} 1, & \text{se } w^T x + b \geq 0, \\ 0, & \text{caso contrário,} \end{cases}$$

onde:

- $x \in \mathbb{R}^n$ é o vetor de entrada,
- $w \in \mathbb{R}^n$ é o vetor de pesos,
- $b \in \mathbb{R}$ é o viés.



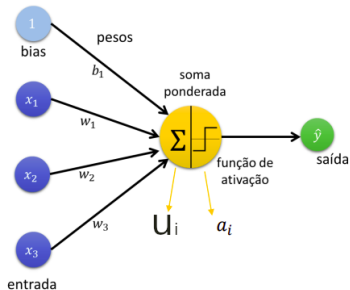
Estrutura do Perceptron



- Neurônio de entrada
- Neurônio intermediário
- Neurônio de saída



Estrutura do Perceptron



Vetor de entrada, de dimensão $(1, n_x)$

$$\mathbf{x}^{(i)} = [x_1, x_2, \dots, x_{n_x}]$$

Vetor de pesos, de dimensão $(1, n_x)$

$$\mathbf{w} = [w_1, w_2, \dots, w_{n_x}]$$

Estado do neurônio, escalar

$$U_i = w_1 x_1 + \dots + w_{n_x} x_{n_x} + b = \sum_{j=0}^{n_x} x_j^{(i)} w_j + b = \mathbf{w}^T \mathbf{x}^{(i)} + b$$

Ativação, escalar

$$a_i = g(U_i)$$

Previsão, escalar

$$\hat{y}_i = a_i$$



Neurônio que Implementa a Porta OR

- ▶ Considere um neurônio com duas entradas x_1 e x_2 .
- ▶ Os pesos sinápticos são $w_1 = 1$ e $w_2 = 1$.
- ▶ O limiar de ativação é $\theta = 0,5$.
- ▶ A saída y é dada por:

$$y = \begin{cases} 1, & \text{se } u \geq 0, \\ 0, & \text{se } u < 0, \end{cases}$$

onde $u = w_1x_1 + w_2x_2 - \theta$.



Cálculo da Saída para a Porta OR

- ▶ A porta OR é definida pela seguinte tabela verdade:

x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	1

- ▶ Vamos calcular a saída y para cada combinação de entradas.



Exemplo de Cálculo

- ▶ Para $x_1 = 0$ e $x_2 = 0$:

$$u = (1 \cdot 0) + (1 \cdot 0) - 0,5 = -0,5 \Rightarrow y = 0.$$

- ▶ Para $x_1 = 0$ e $x_2 = 1$:

$$u = (1 \cdot 0) + (1 \cdot 1) - 0,5 = 0,5 \Rightarrow y = 1.$$

- ▶ Para $x_1 = 1$ e $x_2 = 0$:

$$u = (1 \cdot 1) + (1 \cdot 0) - 0,5 = 0,5 \Rightarrow y = 1.$$

- ▶ Para $x_1 = 1$ e $x_2 = 1$:

$$u = (1 \cdot 1) + (1 \cdot 1) - 0,5 = 1,5 \Rightarrow y = 1.$$



Regra de Treinamento do Perceptron

- ▶ A regra de treinamento do Perceptron ajusta os pesos w e o viés b para classificar corretamente os dados de treinamento.
- ▶ Para um conjunto de dados linearmente separável, o algoritmo converge em um número finito de passos.
- ▶ A atualização dos pesos é dada por:

$$w \leftarrow w + \eta(y_i - \hat{y}_i)x_i,$$

onde:

- ▶ η é a taxa de aprendizado,
- ▶ y_i é a saída desejada,
- ▶ \hat{y}_i é a saída prevista.



Neurônio que Implementa a Porta AND

- ▶ Considere um neurônio com duas entradas x_1 e x_2 .
- ▶ Inicialmente, os pesos sinápticos são $w_1 = 0,2$ e $w_2 = 0,2$.
- ▶ O limiar de ativação é $\theta = 0,5$.
- ▶ A saída y é dada por:

$$y = \begin{cases} 1, & \text{se } u \geq 0, \\ 0, & \text{se } u < 0, \end{cases}$$

onde $u = w_1x_1 + w_2x_2 - \theta$.



Regra de Atualização dos Pesos

- ▶ A regra de atualização dos pesos é dada por:

$$w_i \leftarrow w_i + \eta(y_{\text{esperado}} - y_{\text{calculado}})x_i,$$

onde:

- ▶ η é a taxa de aprendizado (vamos usar $\eta = 0,1$),
- ▶ y_{esperado} é a saída desejada,
- ▶ $y_{\text{calculado}}$ é a saída calculada pelo neurônio.



Passo 1: $x_1 = 1$, $x_2 = 1$

- ▶ Saída esperada: $y_{\text{esperado}} = 1$.
- ▶ Cálculo de u :

$$u = (0,2 \cdot 1) + (0,2 \cdot 1) - 0,5 = -0,1 \Rightarrow y = 0.$$

- ▶ Erro: $y_{\text{esperado}} - y_{\text{calculado}} = 1 - 0 = 1$.
- ▶ Atualização dos pesos:

$$w_1 \leftarrow 0,2 + 0,1 \cdot (1 - 0) \cdot 1 = 0,3,$$

$$w_2 \leftarrow 0,2 + 0,1 \cdot (1 - 0) \cdot 1 = 0,3.$$

- ▶ Novos pesos: $w_1 = 0,3$, $w_2 = 0,3$.



Passo 1.1: $x_1 = 1, x_2 = 1$

- ▶ Saída esperada: $y_{\text{esperado}} = 1$.
- ▶ Cálculo de u :

$$u = (0,3 \cdot 1) + (0,3 \cdot 1) - 0,5 = 0,1 \quad \Rightarrow \quad y = 1.$$

- ▶ Erro: $y_{\text{esperado}} - y_{\text{calculado}} = 1 - 1 = 0$.
- ▶ Não há atualização dos pesos.



Passo 2: $x_1 = 1$, $x_2 = 0$

- ▶ Saída esperada: $y_{\text{esperado}} = 0$.
- ▶ Cálculo de u :

$$u = (0,3 \cdot 1) + (0,3 \cdot 0) - 0,5 = -0,2 \Rightarrow y = 0.$$

- ▶ Erro: $y_{\text{esperado}} - y_{\text{calculado}} = 0 - 0 = 0$.
- ▶ Não há atualização dos pesos.



Passo 3: $x_1 = 0$, $x_2 = 1$

- ▶ Saída esperada: $y_{\text{esperado}} = 0$.
- ▶ Cálculo de u :

$$u = (0, 3 \cdot 0) + (0, 3 \cdot 1) - 0,5 = -0,2 \Rightarrow y = 0.$$

- ▶ Erro: $y_{\text{esperado}} - y_{\text{calculado}} = 0 - 0 = 0$.
- ▶ Não há atualização dos pesos.



Passo 4: $x_1 = 0$, $x_2 = 0$

- ▶ Saída esperada: $y_{\text{esperado}} = 0$.
- ▶ Cálculo de u :

$$u = (0, 3 \cdot 0) + (0, 3 \cdot 0) - 0,5 = -0,5 \Rightarrow y = 0.$$

- ▶ Erro: $y_{\text{esperado}} - y_{\text{calculado}} = 0 - 0 = 0$.
- ▶ Não há atualização dos pesos.



Função de Ativação

- ▶ A função de ativação $\varphi : \mathbb{R} \rightarrow \mathbb{R}$ é aplicada à saída de um neurônio artificial.
- ▶ Ela introduz não linearidade no modelo, permitindo que a rede neural aprenda padrões complexos.
- ▶ Sem a não linearidade, uma rede neural com múltiplas camadas seria equivalente a uma única transformação linear.



Importância da Não Linearidade

- ▶ Em redes neurais com várias camadas ocultas, a ausência de não linearidade resultaria em uma composição de transformações afins.
- ▶ Matematicamente, isso pode ser expresso como:

$$f(x) = W_n(W_{n-1}(\dots W_1(x) + b_1) + b_{n-1}) + b_n,$$

onde W_i são matrizes de pesos e b_i são vetores de viés.

- ▶ Sem funções de ativação não lineares, $f(x)$ seria equivalente a uma única transformação afim:

$$f(x) = W'x + b'.$$



Exemplos de Funções de Ativação

► **Função Limiar (Step):**

$$\varphi(t) = \begin{cases} 1, & t \geq 0, \\ 0, & t < 0. \end{cases}$$

► **Função Logística (Sigmoide):**

$$\sigma(t) = \frac{1}{1 + e^{-t}}.$$

► **Função ReLU (Rectified Linear Unit):**

$$\text{ReLU}(t) = \max(0, t).$$



Impacto na Aprendizagem

- ▶ A não linearidade introduzida pela função de ativação permite que a rede neural modele relações complexas entre entradas e saídas.
- ▶ Sem funções de ativação não lineares, a rede não seria capaz de aprender funções não lineares, limitando sua aplicabilidade.
- ▶ A escolha da função de ativação afeta a eficiência do treinamento e a capacidade de generalização do modelo.



Limitações do Perceptron

- ▶ O Perceptron só pode classificar dados linearmente separáveis.
- ▶ Ele não consegue resolver problemas não lineares, como o problema do XOR (ou-exclusivo).
- ▶ Essa limitação foi destacada por Minsky e Papert em 1969, o que levou a um declínio temporário no interesse por redes neurais.



Definição Matemática de uma RNA

- ▶ Uma Rede Neural Artificial (RNA) pode ser vista como uma função f que mapeia um conjunto de dados de entrada X para um conjunto de saídas Y .
- ▶ Formalmente, a RNA é uma função:

$$f : \mathbb{R}^{n_x} \rightarrow \mathbb{R}^{n_y},$$

onde:

- ▶ n_x é o número de features (entradas),
- ▶ n_y é o número de saídas.



Representação dos Dados

- ▶ Os dados de entrada são representados por uma matriz $X \in \mathbb{R}^{n_x \times m}$, onde:
 - ▶ n_x é o número de features por exemplo,
 - ▶ m é o número total de exemplos de treinamento.
- ▶ As saídas desejadas são representadas por uma matriz $Y \in \mathbb{R}^{n_y \times m}$, onde:
 - ▶ n_y é o número de saídas por exemplo.



Processo de Aprendizagem

- ▶ A principal característica de uma rede neural é sua capacidade de aprender e melhorar o desempenho com base em estímulos externos.
- ▶ A aprendizagem envolve a atualização da representação interna da rede, ajustando sua arquitetura e os pesos das conexões entre neurônios.
- ▶ Esse processo permite que a rede desempenhe tarefas específicas de forma mais eficiente ao longo do tempo.



Atualização da Representação Interna

- ▶ A aprendizagem ocorre por meio da modificação da arquitetura da rede e do ajuste dos pesos sinápticos.
- ▶ As regras de aprendizagem determinam como os pesos são atualizados em resposta aos erros ou estímulos externos.
- ▶ Essas regras são essenciais para garantir que a rede se adapte e melhore seu desempenho.



Tipos de Regras de Aprendizagem

- ▶ **Aprendizagem por Correção de Erro:**
 - ▶ Utilizada em treinamento supervisionado.
 - ▶ Os pesos são ajustados com base no erro, que é a diferença entre a saída da rede e o valor esperado.
 - ▶ O erro é minimizado gradualmente ao longo dos ciclos de treinamento.
- ▶ Outras regras de aprendizagem incluem:
 - ▶ Aprendizagem Hebbiana,
 - ▶ Aprendizagem Competitiva,
 - ▶ Aprendizagem por Reforço.



Aprendizagem por Correção de Erro

- ▶ A regra de correção de erro é uma das mais comuns em redes neurais supervisionadas.
- ▶ O erro é calculado como:

$$\text{Erro} = y_{\text{esperado}} - y_{\text{calculado}}.$$

- ▶ Os pesos são atualizados usando a fórmula:

$$w_i \leftarrow w_i + \eta \cdot \text{Erro} \cdot x_i,$$

onde η é a taxa de aprendizado e x_i é a entrada.



Treinamento da RNA

- ▶ Para que a RNA aprenda a mapear X para Y , ela precisa ser treinada.
- ▶ O treinamento é um processo de otimização que minimiza uma função de perda \mathcal{L} , que mede a diferença entre as saídas previstas \hat{Y} e as saídas desejadas Y :

$$\mathcal{L}(Y, \hat{Y}) = \frac{1}{m} \sum_{i=1}^m \|y_i - \hat{y}_i\|^2.$$

- ▶ Durante o treinamento, os parâmetros da RNA (pesos e vieses) são ajustados para minimizar \mathcal{L} .



Aprendizado Supervisionado

- ▶ O treinamento de uma RNA é tipicamente supervisionado, o que requer um conjunto de dados rotulados $\{(x_i, y_i)\}_{i=1}^m$.
- ▶ A cada iteração, a RNA gera uma saída \hat{y}_i para a entrada x_i .
- ▶ A diferença entre y_i e \hat{y}_i é usada para atualizar os parâmetros da rede via retropropagação.



Introdução ao Multi-Layer Perceptron (MLP)

- ▶ As limitações do Perceptron simples levaram ao desenvolvimento de redes neurais com múltiplas camadas, como o Adaline e o MLP.
- ▶ O MLP (Multi-Layer Perceptron) foi proposto por Rumelhart e McClelland em 1986, com base em trabalhos anteriores de Widrow e Hoff (1960).
- ▶ O MLP é composto por várias camadas de neurônios artificiais, permitindo a modelagem de funções não lineares complexas.



Introdução ao MLP

- ▶ O Multi-Layer Perceptron (MLP) é uma rede neural artificial com múltiplas camadas de neurônios.
- ▶ Ele é composto por:
 - ▶ Uma camada de entrada,
 - ▶ Uma ou mais camadas ocultas,
 - ▶ Uma camada de saída.
- ▶ A topologia do MLP permite a modelagem de funções não lineares complexas.



Topologia do MLP

- ▶ A topologia de um MLP é definida pela organização das camadas e pelas conexões entre os neurônios.
- ▶ Cada camada é composta por um conjunto de neurônios que processam informações e passam os resultados para a camada seguinte.
- ▶ As conexões entre os neurônios são representadas por pesos sinápticos.



Definição Matemática de uma Camada

- ▶ Uma camada de m neurônios pode ser representada por uma função $L : \mathbb{R}^n \rightarrow \mathbb{R}^m$.
- ▶ A função de uma camada é dada por:

$$L(x) = \varphi(Wx - b),$$

onde:

- ▶ $W \in \mathbb{R}^{m \times n}$ é a matriz de pesos sinápticos,
- ▶ $b \in \mathbb{R}^m$ é o vetor de vieses,
- ▶ φ é a função de ativação aplicada componente a componente.



Camada Densa ou Totalmente Conectada

- ▶ Uma camada é chamada de **densa** ou **totalmente conectada** se a matriz W é uma matriz cheia, ou seja, todos os elementos da matriz podem ser diferentes de zero.
- ▶ Isso significa que cada neurônio na camada recebe entradas de todos os neurônios da camada anterior.
- ▶ Matematicamente, a densidade da camada é expressa pela estrutura completa da matriz W .



Exemplo de MLP com Duas Camadas

- ▶ Considere um MLP com duas camadas:
 - ▶ A primeira camada $L_1 : \mathbb{R}^n \rightarrow \mathbb{R}^k$ com função de ativação φ_1 .
 - ▶ A segunda camada $L_2 : \mathbb{R}^k \rightarrow \mathbb{R}^m$ com função de ativação φ_2 .
- ▶ A função total do MLP é dada por:

$$f(x) = L_2(L_1(x)) = \varphi_2(W_2\varphi_1(W_1x - b_1) - b_2).$$

- ▶ Essa composição de camadas permite a modelagem de funções não lineares complexas.



Papel das Camadas

- ▶ As primeiras camadas (L_1, L_2, \dots) atuam como **extratores de características**, transformando a entrada em representações intermediárias.
- ▶ As últimas camadas (L_{K-1}, L_K) realizam a tarefa de aprendizado de máquina, como classificação ou regressão.
- ▶ A camada L_0 pode ser incluída como a identidade, representando a entrada original.



Redes Neurais Profundas

- ▶ Uma rede neural é considerada profunda (**deep**) quando possui três ou mais camadas ($K \geq 3$).
- ▶ Redes profundas são capazes de modelar funções altamente não lineares e capturar relações complexas nos dados.
- ▶ A profundidade da rede permite a extração de características hierárquicas, onde camadas iniciais detectam padrões simples e camadas posteriores combinam esses padrões em representações mais complexas.



Exemplo de Rede Profunda

- ▶ Considere uma rede com $K = 4$ camadas:

- ▶ $L_1 : \mathbb{R}^n \rightarrow \mathbb{R}^{h_1},$

- ▶ $L_2 : \mathbb{R}^{h_1} \rightarrow \mathbb{R}^{h_2},$

- ▶ $L_3 : \mathbb{R}^{h_2} \rightarrow \mathbb{R}^{h_3},$

- ▶ $L_4 : \mathbb{R}^{h_3} \rightarrow \mathbb{R}^m.$

- ▶ A função total da rede é:

$$N(x) = L_4(L_3(L_2(L_1(x)))).$$

- ▶ Cada camada aplica uma transformação não linear, permitindo que a rede aprenda representações complexas.



Treinamento de Redes Neurais

- ▶ O treinamento de uma rede neural (superficial ou profunda) envolve a minimização de uma função de perda.
- ▶ Em problemas de regressão, a função de perda comum é o **Erro Quadrático Médio (MSE)**:

$$\text{MSE} = \frac{1}{m} \sum_{i=1}^m (y_i - \hat{y}_i)^2,$$

onde y_i é o valor esperado e \hat{y}_i é a saída da rede.

- ▶ Em problemas de classificação, a função de perda comum é a **Entropia Cruzada**:

$$\text{Entropia Cruzada} = - \sum_{i=1}^m y_i \log(\hat{y}_i).$$



Formulação do Problema de Otimização

- ▶ O treinamento é formulado como um problema de otimização irrestrito nos parâmetros da rede (pesos W e vieses b).
- ▶ O objetivo é encontrar os parâmetros que minimizam a função de perda:

$$\min_{W,b} \mathcal{L}(W, b),$$

onde \mathcal{L} é a função de perda.

- ▶ Devido ao grande número de parâmetros, métodos baseados no gradiente são utilizados.



Métodos Baseados no Gradiente

- ▶ O gradiente da função de perda em relação aos parâmetros é calculado usando a **regra da cadeia**.
- ▶ O gradiente é dado por:

$$\nabla_{W,b}\mathcal{L} = \left(\frac{\partial \mathcal{L}}{\partial W}, \frac{\partial \mathcal{L}}{\partial b} \right).$$

- ▶ O cálculo do gradiente é realizado de forma eficiente usando o algoritmo de retropropagação (*backpropagation*).



Algoritmo de Retropropagação

- ▶ O algoritmo de retropropagação consiste em duas fases:
 - ▶ **Fase Forward:** Calcula a saída da rede para uma dada entrada.
 - ▶ **Fase Backward:** Propaga o erro da saída para as camadas anteriores, calculando os gradientes.
- ▶ O gradiente é usado para atualizar os pesos e vieses:

$$W \leftarrow W - \eta \frac{\partial \mathcal{L}}{\partial W}, \quad b \leftarrow b - \eta \frac{\partial \mathcal{L}}{\partial b},$$

onde η é a taxa de aprendizado.

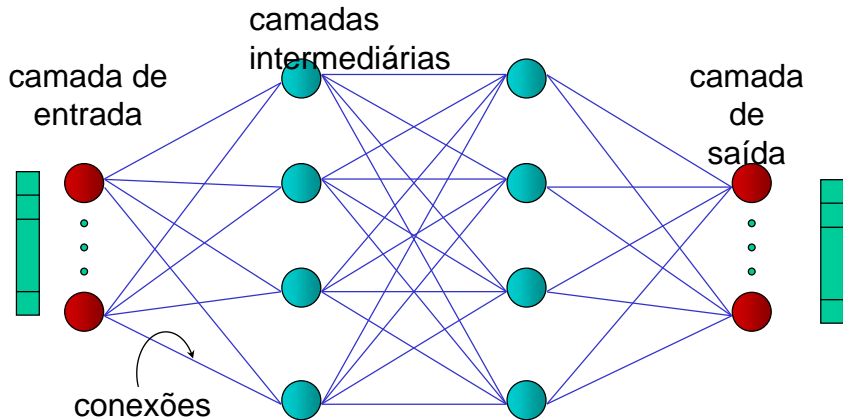


Ilustração do Algoritmo de Retropropagação

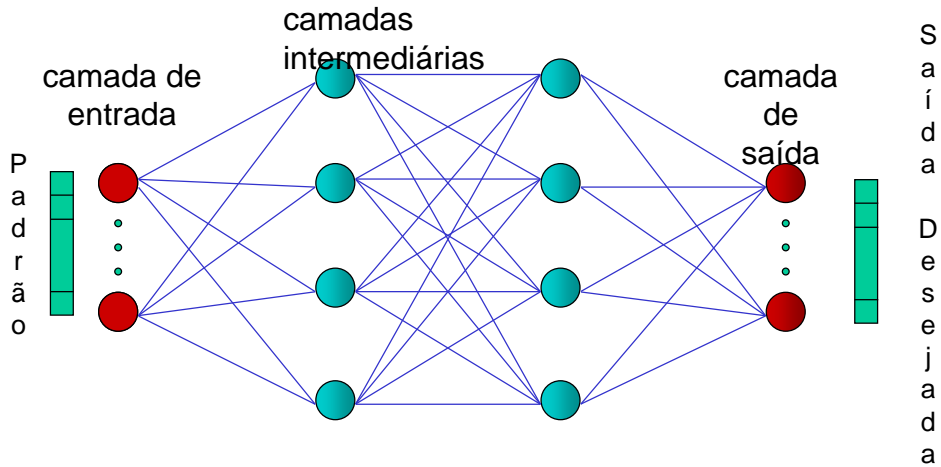
Veja o GIF animado:



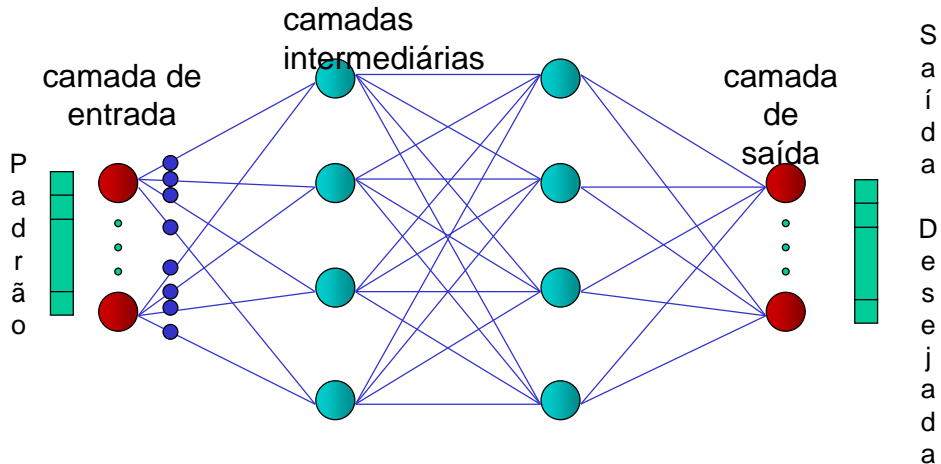
Rede MLP



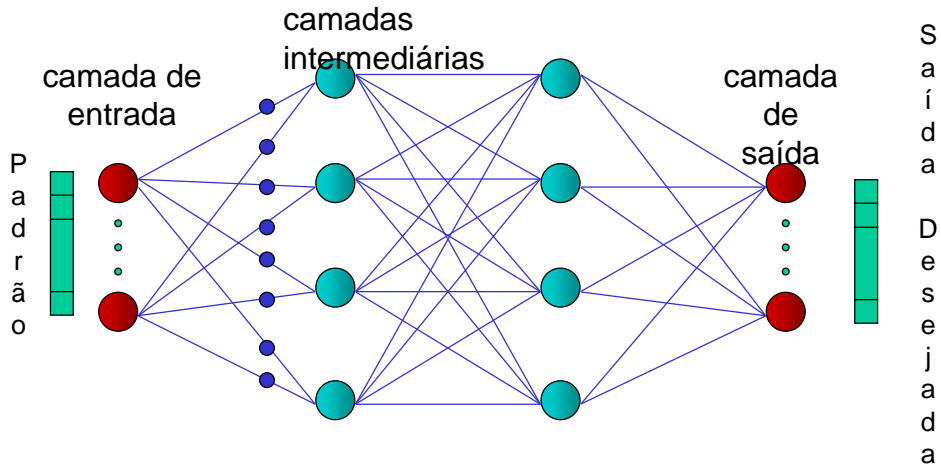
Aprendizado



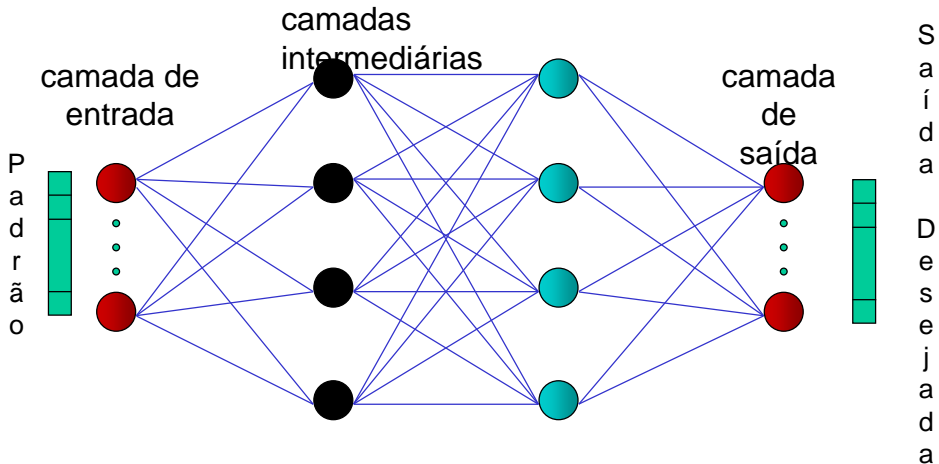
RNA - Aprendizado



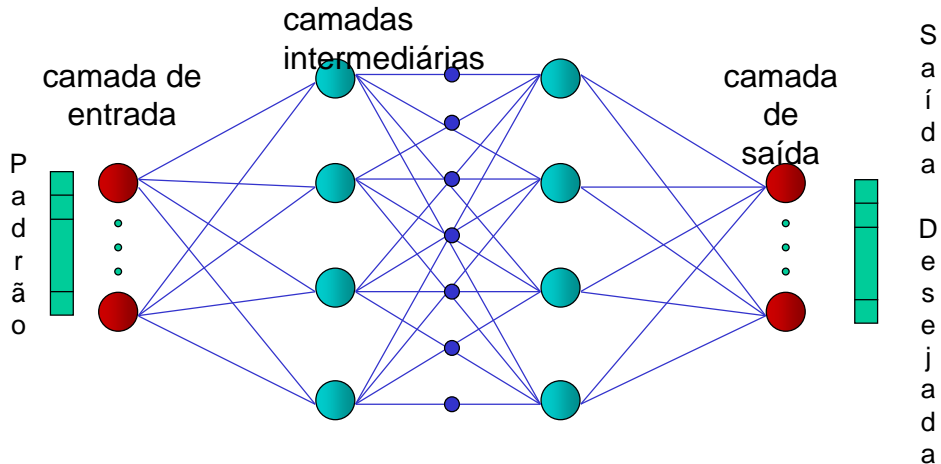
RNA - Aprendizado



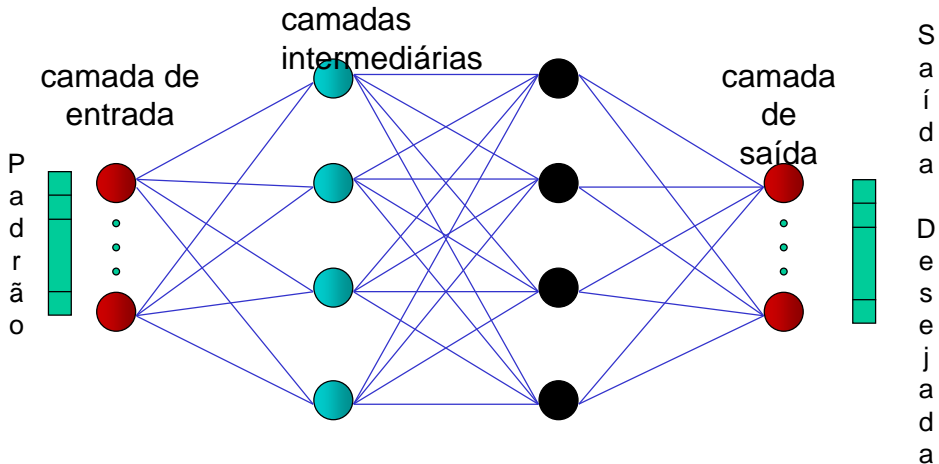
RNA - Aprendizado



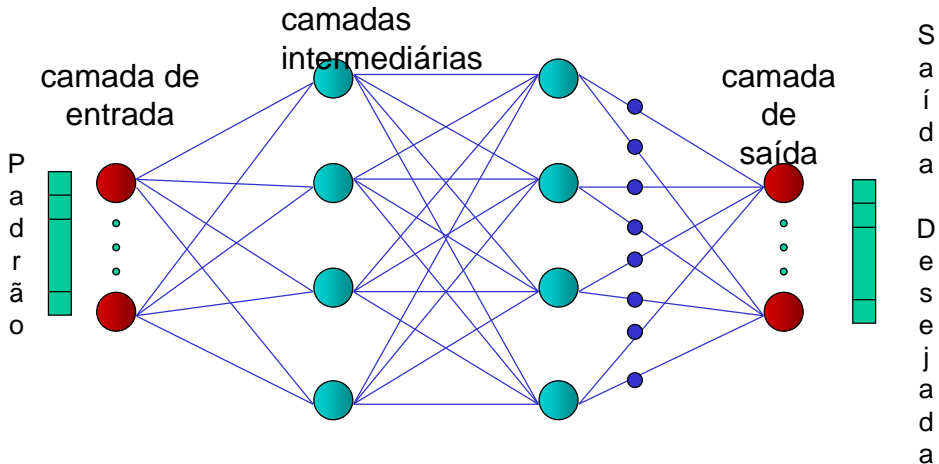
RNA - Aprendizado



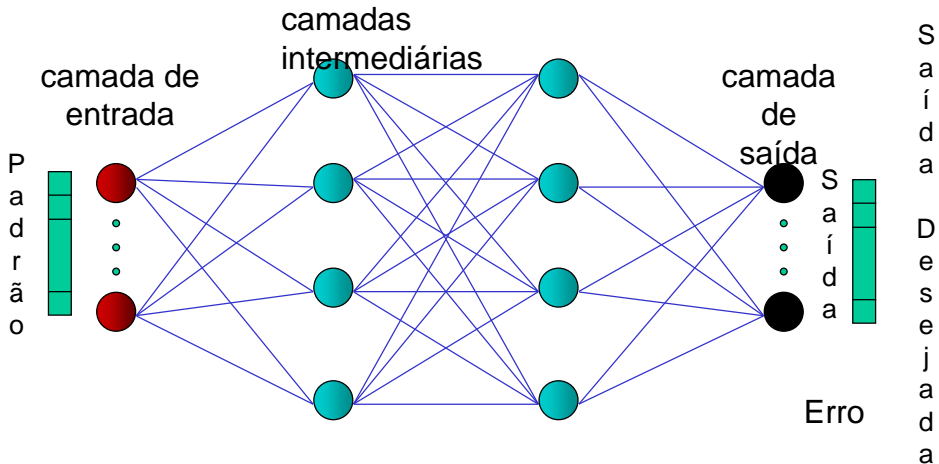
RNA - Aprendizado



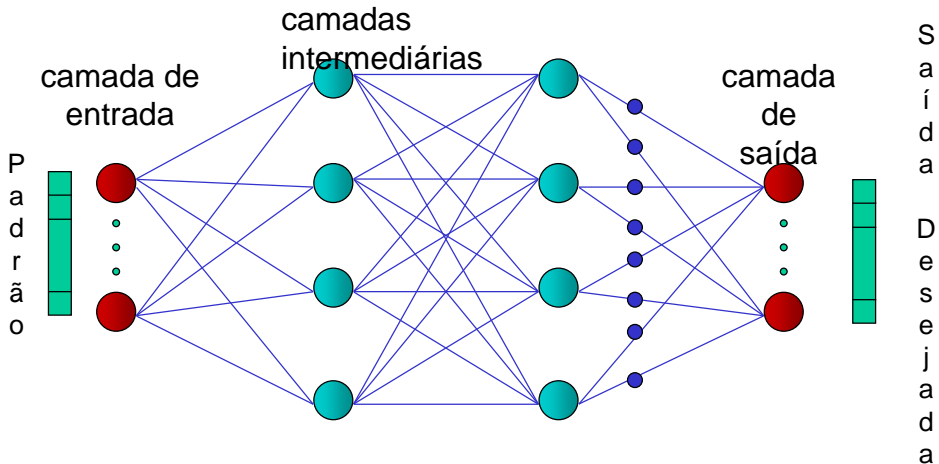
RNA - Aprendizado



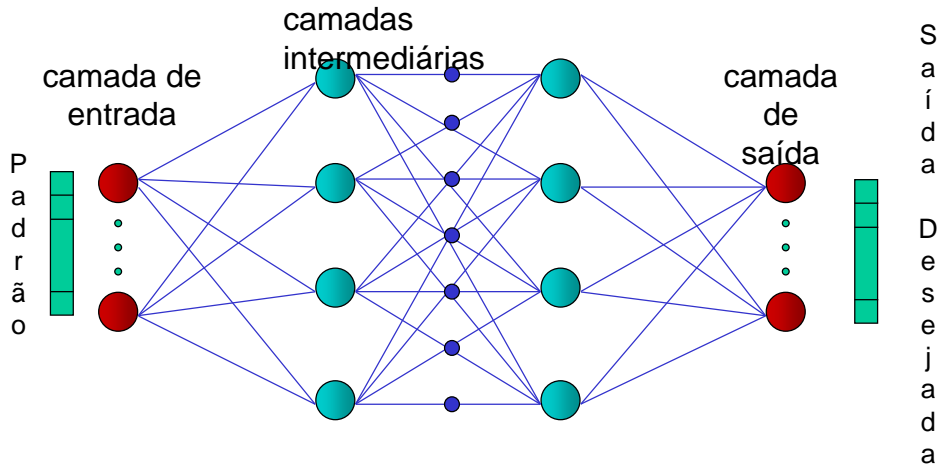
RNA - Aprendizado



RNA - Aprendizado



RNA - Aprendizado



RNA - Aprendizado

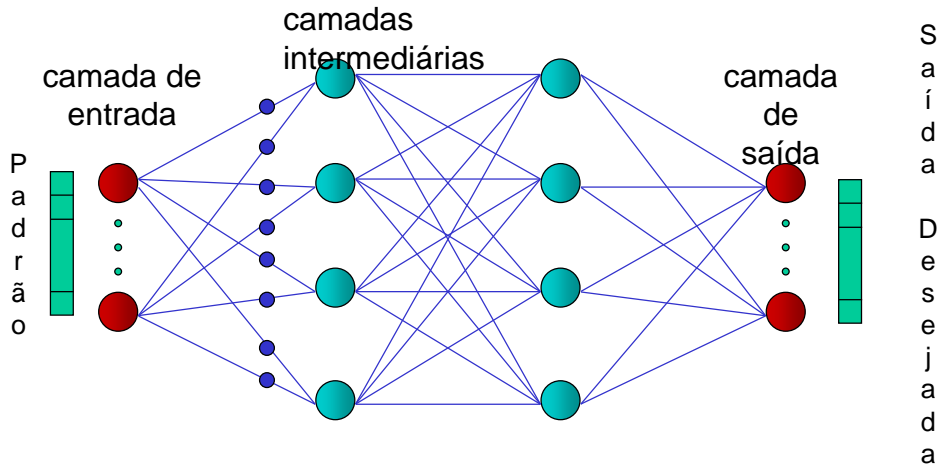
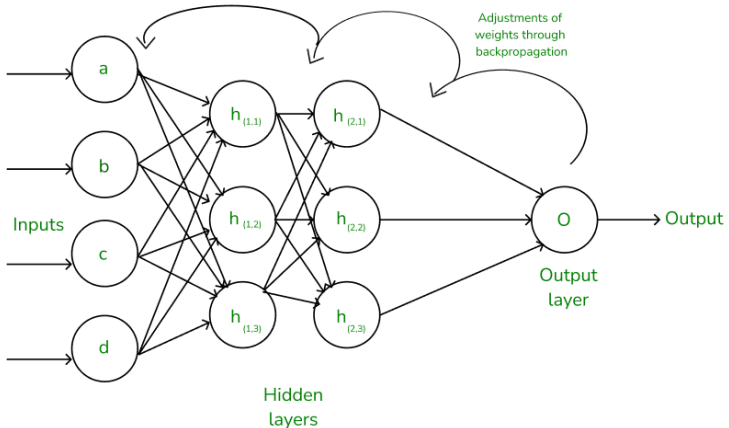


Ilustração do Algoritmo de Retropropagação

- A figura abaixo ilustra o fluxo de informações no algoritmo de retropropagação:



Conclusão

- ▶ O treinamento de redes neurais é um problema de otimização que busca minimizar uma função de perda.
- ▶ Métodos baseados no gradiente, como o algoritmo de retropropagação, são essenciais para o treinamento eficiente de redes neurais.
- ▶ A escolha da função de perda e da taxa de aprendizado é crucial para o desempenho da rede.



└ Redes Neurais

└ Multi Layer Perceptron

RNA na prática

- ▶ sklearn
- ▶ keras



Unsupervised Learning



O Que é Aprendizado Não Supervisionado?

- ▶ O aprendizado não supervisionado é um tipo de aprendizado de máquina onde o modelo é treinado com dados **não rotulados**.
- ▶ O objetivo é encontrar padrões, estruturas ou relações nos dados sem a necessidade de rótulos pré-definidos.
- ▶ É amplamente utilizado em tarefas como:
 - ▶ Agrupamento (clustering),
 - ▶ Redução de dimensionalidade,
 - ▶ Detecção de anomalias.



Formulação Matemática

- ▶ Dado um conjunto de dados $X = \{x_1, x_2, \dots, x_n\}$, onde $x_i \in \mathbb{R}^d$, o objetivo é aprender uma representação ou estrutura subjacente.
- ▶ Em problemas de agrupamento, por exemplo, busca-se particionar X em k grupos C_1, C_2, \dots, C_k , onde:

$$\bigcup_{i=1}^k C_i = X \quad \text{e} \quad C_i \cap C_j = \emptyset \quad \text{para} \quad i \neq j.$$

- ▶ A qualidade do agrupamento é medida por uma função de custo, como a soma dos quadrados intra-cluster:

$$\mathcal{J} = \sum_{i=1}^k \sum_{x \in C_i} \|x - \mu_i\|^2,$$

onde μ_i é o centróide do cluster C_i .



Exemplo: Algoritmo K-Means

- ▶ O K-Means é um algoritmo clássico de agrupamento não supervisionado.
- ▶ Dado um número k de clusters, o algoritmo minimiza a função de custo:

$$\mathcal{J} = \sum_{i=1}^k \sum_{x \in C_i} \|x - \mu_i\|^2.$$

- ▶ Os passos do algoritmo são:
 1. Inicializar os centróides $\mu_1, \mu_2, \dots, \mu_k$.
 2. Atribuir cada ponto x ao cluster mais próximo.
 3. Atualizar os centróides como a média dos pontos no cluster.
 4. Repetir até convergência.



Ilustração do K-Means

- ▶ A figura abaixo ilustra o funcionamento do algoritmo K-Means:

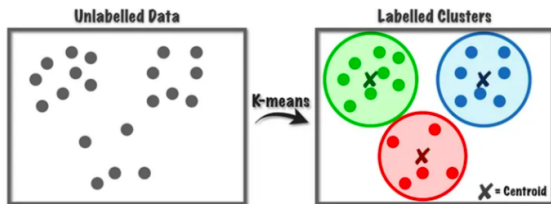


Figura: Passos do algoritmo K-Means: (1) Inicialização dos centróides, (2) Atribuição de pontos aos clusters, (3) Atualização dos centróides.

- ▶ O algoritmo alterna entre atribuir pontos aos clusters e atualizar os centróides até convergir.



Redução de Dimensionalidade

- ▶ Outra tarefa comum no aprendizado não supervisionado é a redução de dimensionalidade.
- ▶ O objetivo é mapear os dados de um espaço de alta dimensão \mathbb{R}^d para um espaço de menor dimensão \mathbb{R}^k , onde $k < d$.
- ▶ Um método popular é a **Análise de Componentes Principais** (PCA), que busca encontrar as direções de máxima variância nos dados.
- ▶ Matematicamente, o PCA resolve o problema de autovalores:

$$\Sigma v = \lambda v,$$

onde Σ é a matriz de covariância dos dados e v são os autovetores.



Ilustração do PCA

- ▶ A figura abaixo ilustra a projeção dos dados em componentes principais:

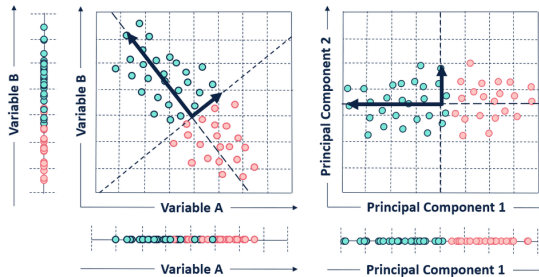
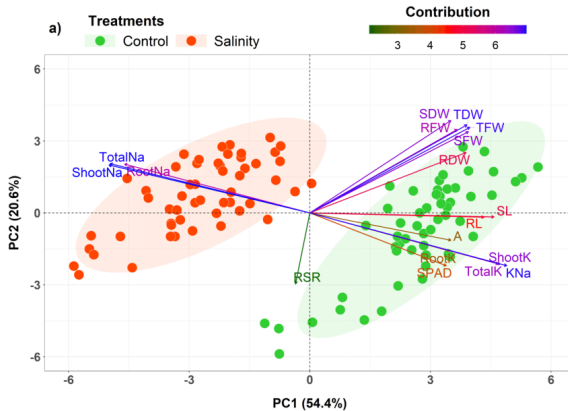


Figura: os dados são projetados em duas componentes principais.

- ▶ As componentes principais são as direções que maximizam a variância dos dados



Ilustração do PCA







Conclusão

- ▶ O aprendizado não supervisionado é essencial para explorar dados não rotulados e descobrir padrões ocultos.
- ▶ Técnicas como agrupamento e redução de dimensionalidade são fundamentais em muitas aplicações.
- ▶ A formulação matemática desses métodos permite a otimização e a interpretação dos resultados.



Referências I

-  Chollet, F. (2021). Deep Learning with Python, Second Edition. Shelter Island, NY: Manning Publications.
-  Géron, A. (2019). Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow. Sebastopol, CA: O'Reilly Media..
-  Russell, S., & Norvig, P. (2021). Artificial Intelligence: A Modern Approach (4^a ed.). Hoboken, NJ: Pearson.
-  Moroney, L. (2020). AI and Machine Learning for Coders. Sebastopol, CA: O'Reilly Media.





Machine Learning

Professor: Elton Sarmanho¹

E-mail: eltonss@ufpa.br



¹Faculdade de Sistemas de Informação - UFPA/CUNTINS

31 de janeiro de 2025