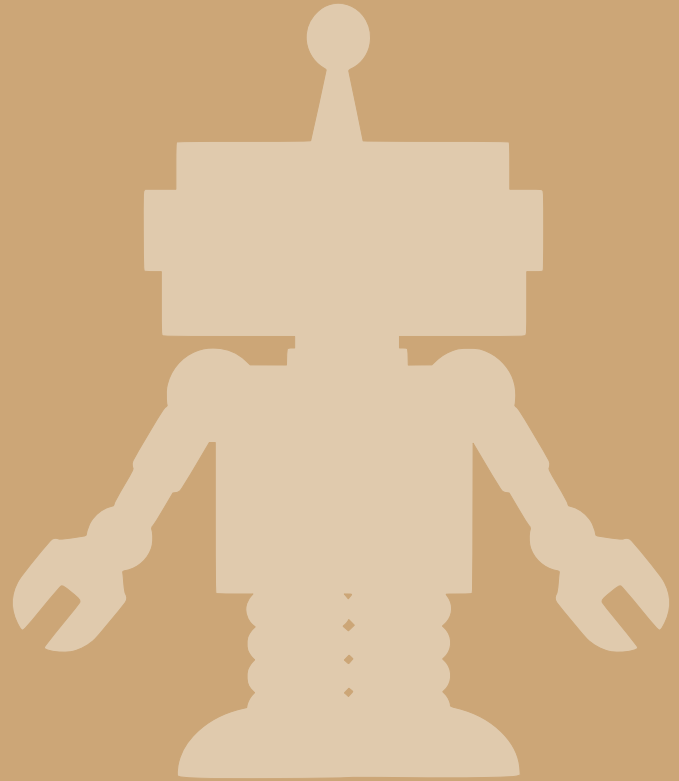


# Aprendizado de Máquina 2

Aula 10

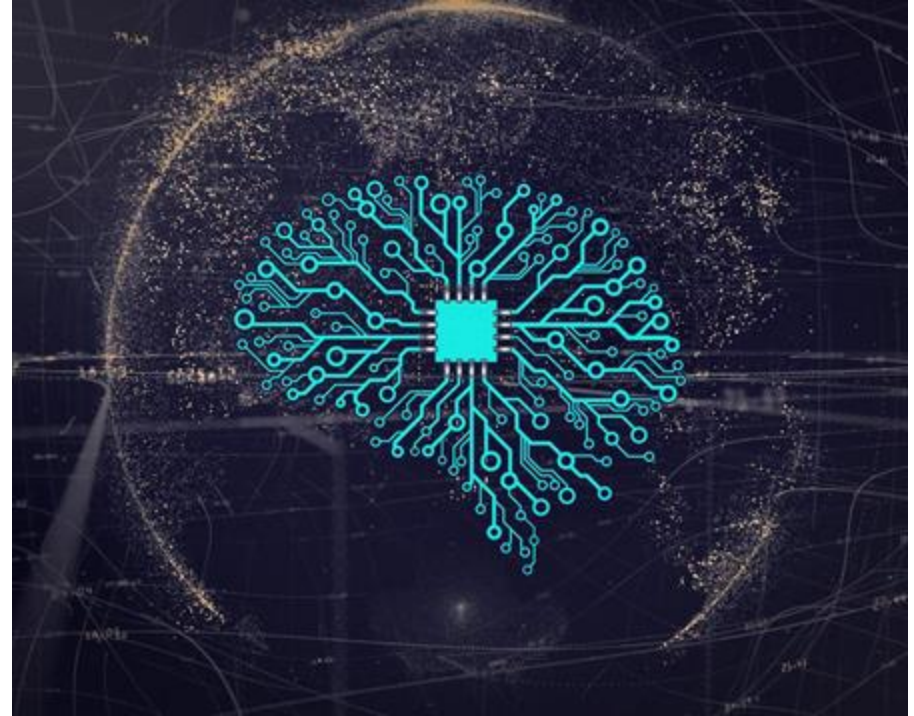
Professora: Patrícia Pampanelli

[patricia.pampanelli@usp.br](mailto:patricia.pampanelli@usp.br)



# Aula de Hoje

- **Algoritmos Genéticos**



# Algoritmos genéticos

# Origem dos Algoritmos Genéticos

Desenvolvimento de simulações computacionais de sistemas genéticos nos anos 1950 e 1960

Pioneiro John Holland iniciou pesquisas na Universidade de Michigan

Holland formalizou princípios que guiam os algoritmos genéticos em seu trabalho '*Adaptation in Natural and Artificial Systems*'



# Introdução

Definição:

- Algoritmos genéticos (AGs) são métodos de busca e otimização baseados nos princípios da seleção natural e genética
- Eles são usados para encontrar soluções aproximadas para problemas complexos



# Introdução

Simulação do Processo de Seleção Natural:

- Os AGs imitam o processo de evolução biológica, onde populações de soluções potenciais evoluem ao longo do tempo
- Através de iterações sucessivas, as soluções "mais aptas" são selecionadas para gerar novas soluções



# Introdução

Inspiração na Teoria da Evolução de Charles Darwin

As espécies evoluem através da seleção natural

Indivíduos mais adaptados têm maior probabilidade de sobreviver e reproduzir

AGs utilizam processos de seleção, cruzamento e mutação para evoluir soluções

Uma solução é avaliada por uma função de desempenho, semelhante à sobrevivência do mais apto na natureza



# Componentes Principais

- **População:**

- Conjunto de soluções potenciais para o problema, representadas como indivíduos ou cromossomos

- **Cromossomos e Genes:**

- Cada solução é representada por um cromossomo, que é composto por genes. Os genes representam as variáveis ou características da solução





# Componentes Principais

- **Operadores Genéticos:**
  - Seleção: Processo de escolher os melhores indivíduos para reprodução, baseado na sua aptidão
- **Cruzamento (Crossover):**
  - Combinação de dois ou mais cromossomos para criar descendentes que herdam características dos pais
- **Mutação:**
  - Alteração aleatória em um ou mais genes para introduzir diversidade genética e evitar convergência prematura.



# Princípios dos Algoritmos Genéticos

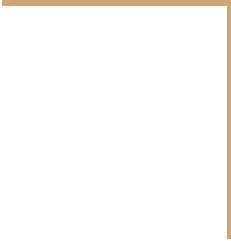
- **Princípio da variabilidade**
  - As características dos indivíduos em uma população pode variar. Como resultado, as amostras de uma população variam entre si
- **Princípio da hereditariedade**
  - Algumas características são passadas para as gerações seguintes. Desta forma, os descendentes se assemelham mais com seus pais do que com qualquer outro indivíduo da população
- **Princípio da seleção natural**
  - As populações em geral competem pelos recursos disponíveis no ambiente. Os indivíduos que estão mais bem adaptados terão mais sucesso ao sobreviver e contribuir com mais descendentes para as próximas gerações



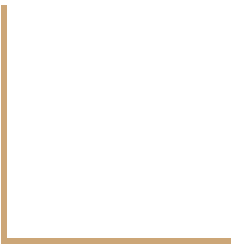
# Analogia dos algoritmos genéticos

- Nos algoritmos genéticos é mantido um conjunto de **indivíduos** que representam **soluções candidatas** a resolver um determinado **problema**
- Estes indivíduos são avaliados iterativamente e usados para criar novas gerações de soluções
- Aqueles que são melhores em resolver determinado problema têm chances maiores de serem selecionados e passar suas características para as gerações futuras
- Como resultado, a cada geração as possíveis soluções contidas na população ficam melhores em resolver o problema em questão





# Componentes de um algoritmo genético



# Componentes de um algoritmo genético

Estrutura e Funcionamento dos Elementos-Chave

- **Genótipo**
  - **População**
  - **Função de desempenho**
  - **Métodos de Seleção**
  - **Cruzamento**
  - **Mutação**
-

# Componentes de um algoritmo genético

## Genótipo:

- No caso dos algoritmos genéticos, este conceito representa uma coleção de características de um indivíduo. Estas características podem ser expressas por uma string com representação binária:

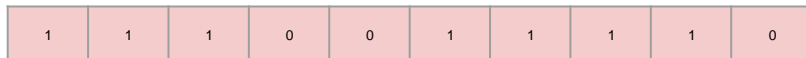
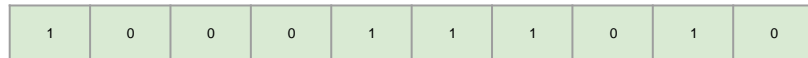
1	0	0	0	1	1	1	0	1	0
---	---	---	---	---	---	---	---	---	---



# Componentes de um algoritmo genético

## População:

- Em qualquer momento no processo de otimização, o algoritmo genético mantém um conjunto de soluções possíveis (população):



# Componentes de um algoritmo genético

## **Função de desempenho:**

- A função de desempenho é usada para avaliar o quão bem um determinado indivíduo resolve o problema em questão
- Também conhecida como função objetivo
- Esta medida é usada para verificar quando o processo de evolução pode ser interrompido visto que uma solução adequada foi encontrada





# Componentes de um algoritmo genético

## Métodos de Seleção:

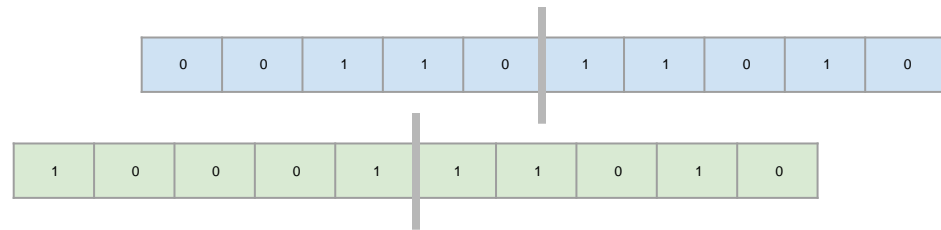
- Roleta (ou Seleção Proporcional): Probabilidade de seleção proporcional à aptidão
- Torneio: Um grupo aleatório de indivíduos é escolhido, e o mais apto é selecionado
- Ranking: Indivíduos são ordenados por aptidão, e a seleção é baseada na posição no ranking



# Componentes de um algoritmo genético

## Cruzamento:

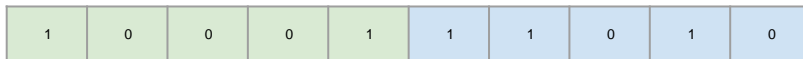
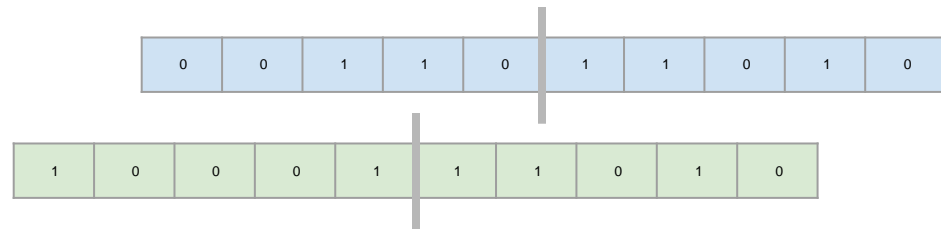
- Para criar um novo indivíduo, um par é escolhido dentro da população atual. Sua representação é definida a partir de parte da representação dos pais:



# Componentes de um algoritmo genético

## Cruzamento:

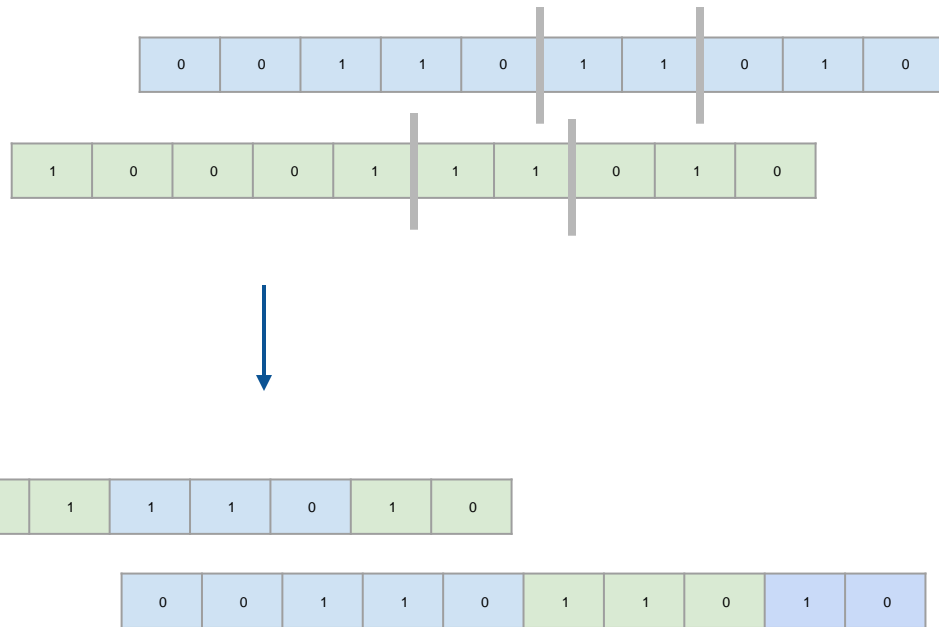
- Para criar um novo indivíduo, um par é escolhido dentro da população atual. Sua representação é definida a partir de parte da representação dos pais:
- Este é um cruzamento com um único ponto de contato



# Componentes de um algoritmo genético

## Cruzamento:

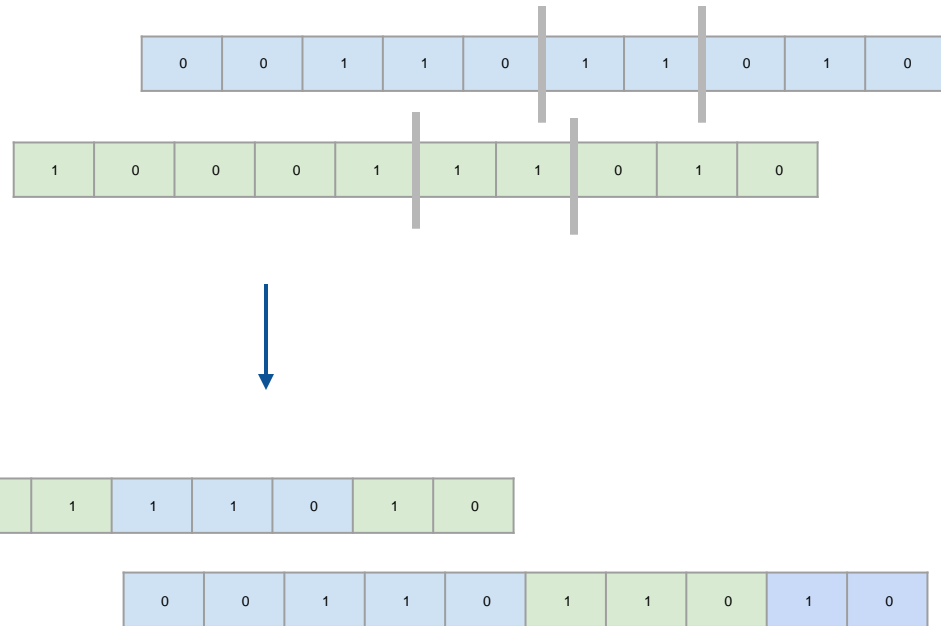
- O cruzamento pode ter 2 ou mais pontos de contato



# Componentes de um algoritmo genético

## Cruzamento:

- Outras estratégias de cruzamento também podem ser implementadas. Por exemplo, onde o número de genes herdados de um dos pais é diferente do outro.



# Componentes de um algoritmo genético

## **Cruzamento:**

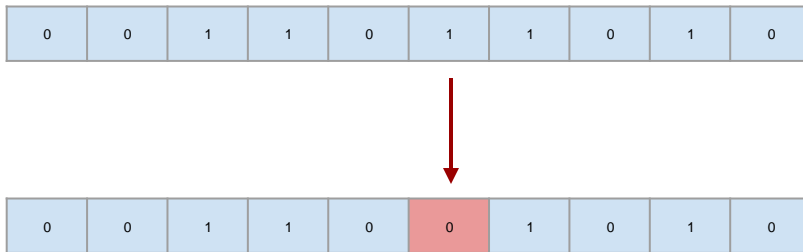
- Esta operação deve ser implementada respeitando a definição do problema. Por exemplo, ao resolver o problema do caixeiro viajante, uma representação para este problema é o número das cidades na ordem que elas serão visitadas. Neste caso, não podemos simplesmente combinar os vetores como fizemos anteriormente.



# Componentes de um algoritmo genético

## Mutação:

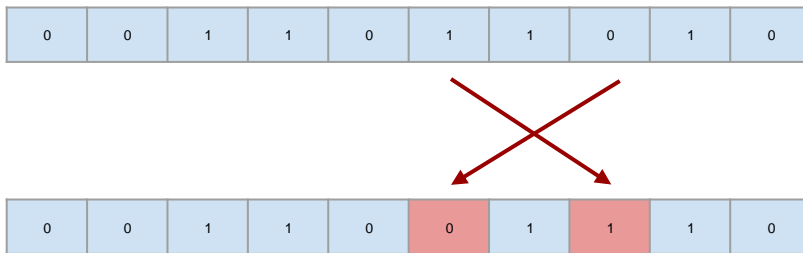
- O objetivo da mutação é trazer periodicamente e de forma randômica novas características para o conjunto de soluções possíveis presentes na população



# Componentes de um algoritmo genético

## Mutação:

- As mutações, assim como os cruzamentos, devem respeitar as restrições do problema que está sendo resolvido. No mesmo caso do caixeiro viajante, o flip de um valor pode não ser adequado.
- Neste caso, uma mutação trocando dois genes de lugar pode ser bem mais interessante.

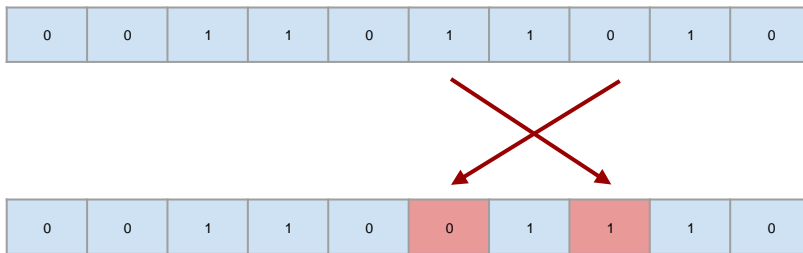




# Componentes de um algoritmo genético

## Mutação:

- As mutações, assim como os cruzamentos, devem respeitar as restrições do problema que está sendo resolvido. No mesmo caso do caixeiro viajante, o flip de um valor pode não ser adequado.
- Neste caso, uma mutação trocando dois genes de lugar pode ser bem mais interessante.

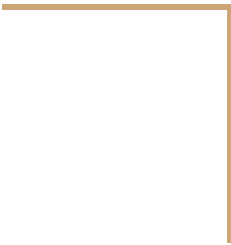


# Componentes de um algoritmo genético

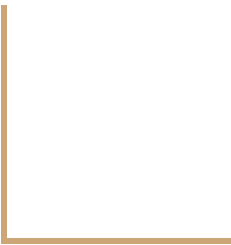
## **Mutação:**

- A mutação pode não envolver a modificação dos genes
- Os genes podem ser simplesmente reordenados em um processo de mutação





# Diferenças entre algoritmos genéticos e os algoritmos tradicionais



# Conjunto de Soluções

A maioria dos algoritmos que utilizamos até este momento, como o gradiente descendente, mantém uma única solução viável em um determinado momento no processo de otimização

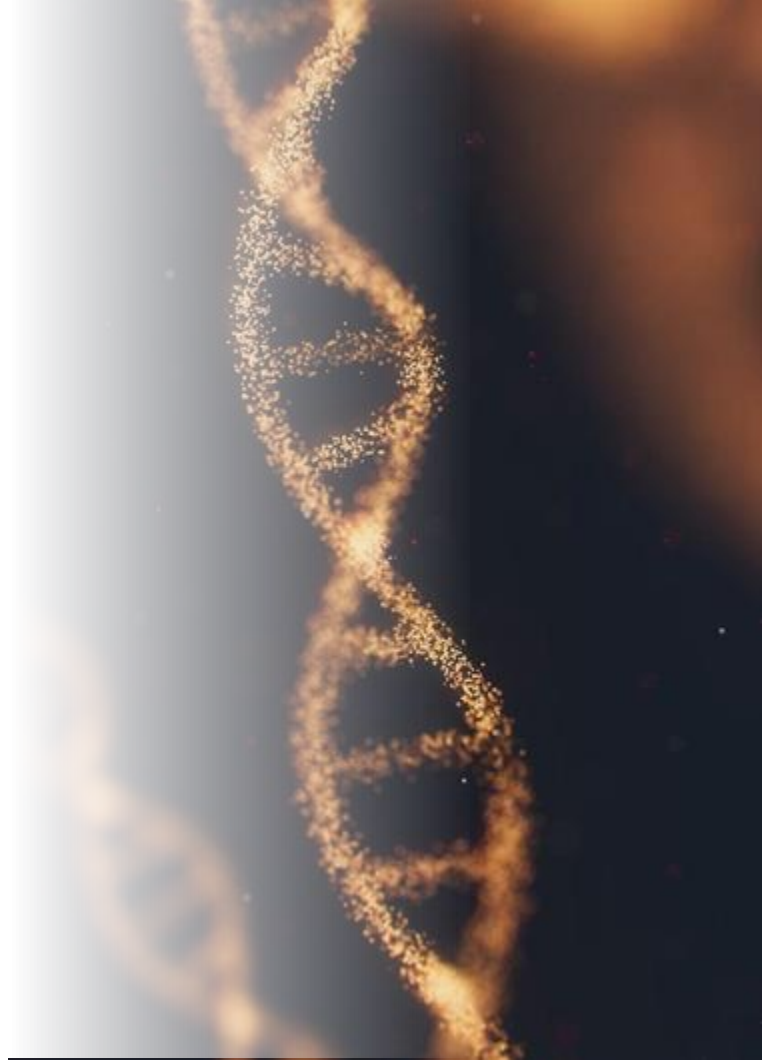
Isso pode ser interessante, visto que toda a população representam soluções viáveis (melhores ou piores) para o problema em questão



# Representação Genética

Os algoritmos genéticos trabalham somente com a representação (ou codificação) de um determinado problema. Um exemplo de representação é a codificação binária que vimos nos slides anteriores

Dependendo do problema, podemos utilizar representações mais complexas





# Integração de Algoritmos Genéticos com Redes Neurais



# Otimização de Pesos e Biases



## Representação dos Indivíduos

Cada indivíduo na população representa uma rede neural com um conjunto específico de pesos e biases.



## Hiperparâmetros

Esses parâmetros são tratados como genes dentro do cromossomo de cada indivíduo.



## Processo de Evolução

A população inicial é composta por várias redes neurais, cada uma com pesos e biases inicializados aleatoriamente.



## Desempenho

O desempenho de cada rede é avaliado usando uma função que geralmente é baseada no erro de previsão da rede.

# Otimização de Pesos e Biases



## **Crossover**

Redes com melhor desempenho são selecionadas para crossover, onde seus parâmetros são combinados para criar novas redes.



## **Mutação**

A mutação é aplicada para introduzir pequenas alterações nos parâmetros, promovendo diversidade genética.



# Otimização de Hiperparâmetros



## Hiperparâmetros como Genes

Hiperparâmetros, como a taxa de aprendizado ou o número de camadas ocultas, são considerados genes no cromossomo.

A população inicial é composta por várias configurações diferentes de hiperparâmetros.



## Avaliação e Seleção

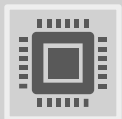
Cada configuração é avaliada em termos de desempenho da rede treinada com esses hiperparâmetros.

As melhores configurações são selecionadas para crossover e mutação, permitindo a exploração eficiente do espaço de hiperparâmetros.

# Vantagens da Integração

- **Superação de Mínimos Locais:**
  - Os algoritmos genéticos podem ajudar a evitar que as redes neurais fiquem presas em mínimos locais, um problema comum em métodos baseados em gradiente.
- **Exploração Eficiente:**
  - Permitem uma exploração mais ampla do espaço de soluções possíveis, potencialmente encontrando configurações que métodos tradicionais poderiam ignorar.

# Desafios e Considerações



**Complexidade Computacional:** O uso combinado pode ser computacionalmente intensivo devido à necessidade de avaliar muitas redes diferentes.



**Calibração dos Parâmetros dos AGs:** É importante ajustar corretamente parâmetros como taxas de crossover e mutação para garantir uma evolução eficaz sem convergência prematura.



Vantagens



# Vantagens

- Capacidade de encontrar soluções satisfatórias em espaços de busca vastos e mal definidos.
- Capazes de lidar com problemas para os quais não conseguimos obter uma representação matemática
- Não requerem suposições sobre continuidade ou derivadas do espaço de busca.



# Vantagens

- Resilientes em cenários ruidosos
- São altamente paralelizáveis e distribuíveis
- Adaptabilidade a mudanças nas condições do problema ou do ambiente





Desvantagens



# Desvantagens

- Necessidade de uma definição especial para codificação, para função de desempenho, para as operações de cruzamento e mutação
- Risco de convergência prematura
- Não tem garantia de solução

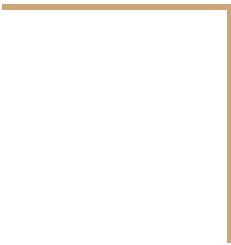




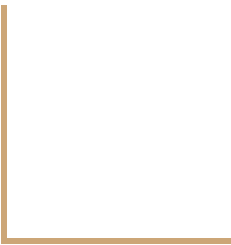
# Desvantagens

- Ajuste complexo dos parâmetros, como taxas de mutação e cruzamento, que afetam o desempenho:
  - Tamanho da população
  - Taxa de cruzamento
  - Taxa de mutação
  - Número máximo de gerações
  - etc

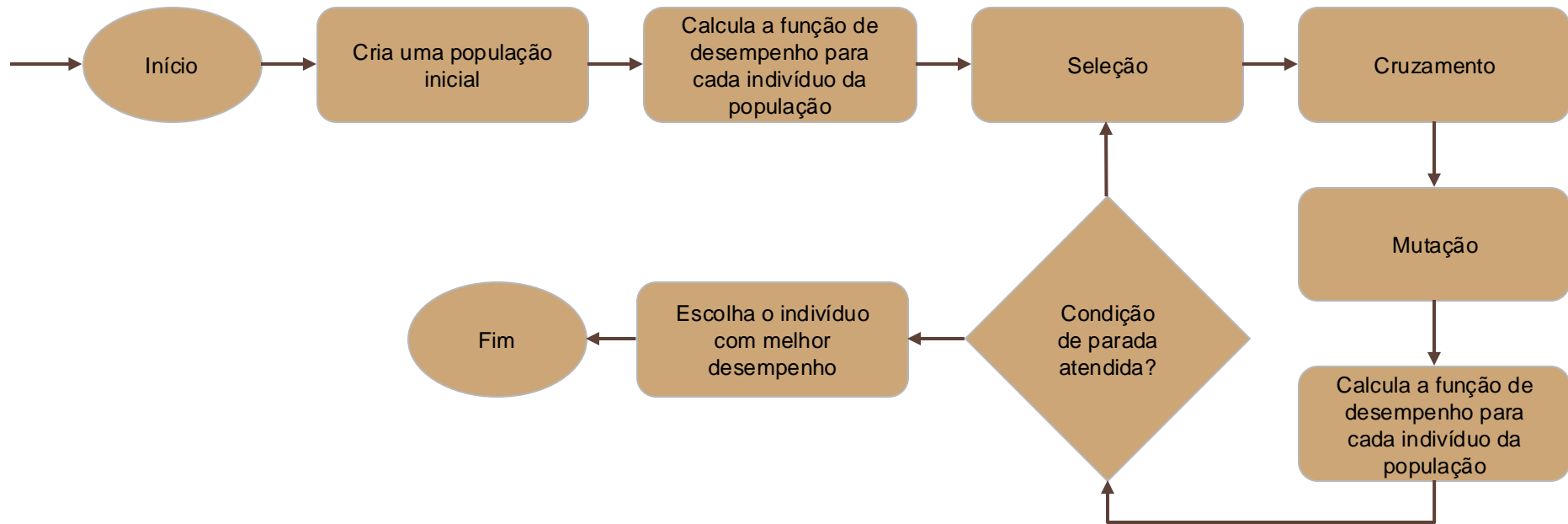




# Fluxo básico de um algoritmo genético



# Fluxo básico de um algoritmo genético



# Aplicações

# Aplicações iniciais

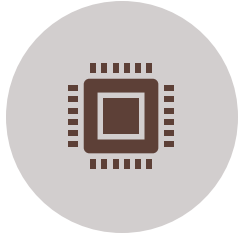


OTIMIZAÇÃO DE FUNÇÕES  
MATEMÁTICAS COMPLEXAS

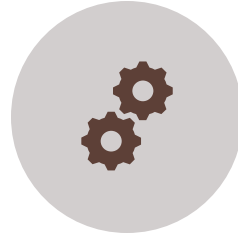


PROBLEMAS CLÁSSICOS  
COMO O CAIXEIRO VIAJANTE  
E ALOCAÇÃO DE RECURSOS

# Evolução das Aplicações




Com o aumento do poder computacional, AGs passaram a ser usados em simulações complexas, como **modelagem molecular e previsão do tempo**



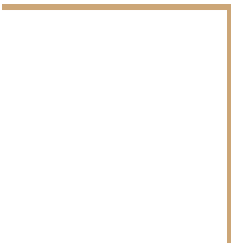
Na medicina, são usados para **segmentação de imagens médicas e otimização de tratamentos**

# Discussão sobre Evolução

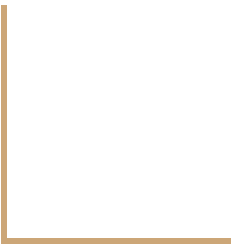
As aplicações evoluíram para incluir problemas multidisciplinares que exigem soluções adaptativas e robustas



A capacidade dos AGs de lidar com problemas não-lineares e mal-definidos os torna ferramentas valiosas em contextos onde métodos tradicionais falham



# Framework para Algoritmos Genéticos





# DEAP (Distributed Evolutionary Algorithms in Python)

- Oferece uma ampla gama de ferramentas para implementar algoritmos evolucionários, incluindo operadores genéticos e suporte para execução paralela



# DEAP



DISTRIBUTED  
EVOLUTIONARY  
ALGORITHMS IN  
PYTHON

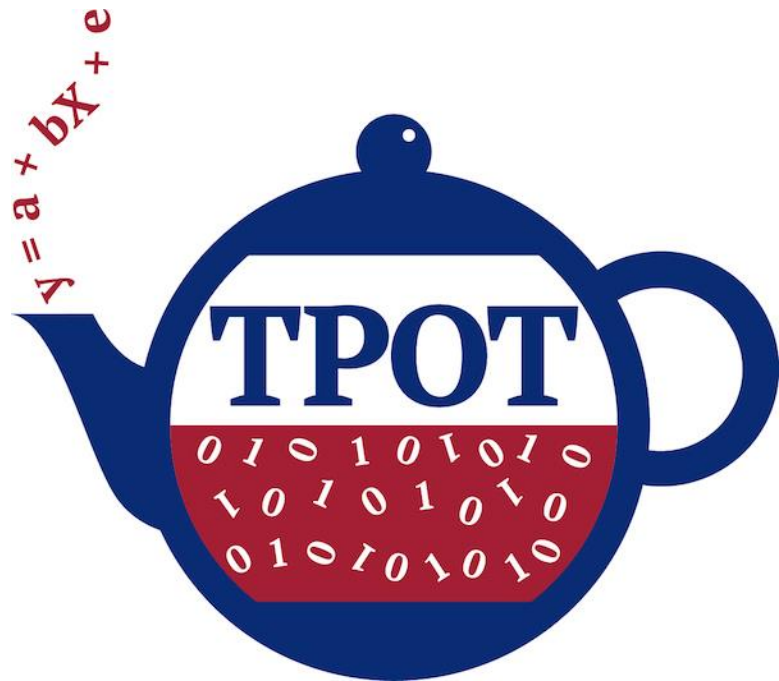
# PyGAD

- [PyGAD: Genetic Algorithm in Python](#)
- Uma biblioteca fácil de usar para construir algoritmos genéticos. Suporta otimização de problemas de múltiplos objetivos e integração com bibliotecas como Keras e PyTorch



# TPOT (Tree-based Pipeline Optimization Tool)

- [TPOT](#) é uma ferramenta de AutoML (Automated Machine Learning) em Python que utiliza programação genética para otimizar pipelines de aprendizado de máquina. Ele automatiza a seleção do melhor modelo e dos hiperparâmetros correspondentes, economizando tempo e melhorando os resultados.



# Como TPOT Funciona

## **Estrutura Baseada em Árvores**

Utiliza árvores binárias para representar modelos de pipeline, incluindo preparação de dados, modelagem algorítmica, configuração de hiperparâmetros e seleção de modelos.

## **Programação Genética**

Explora milhares de possíveis pipelines para encontrar o mais adequado ao conjunto de dados específico.

## **Integração com Scikit-learn**

Baseado no scikit-learn, facilitando a familiaridade com o código gerado.

# Vantagens do TPOT

## **Automatização Completa**

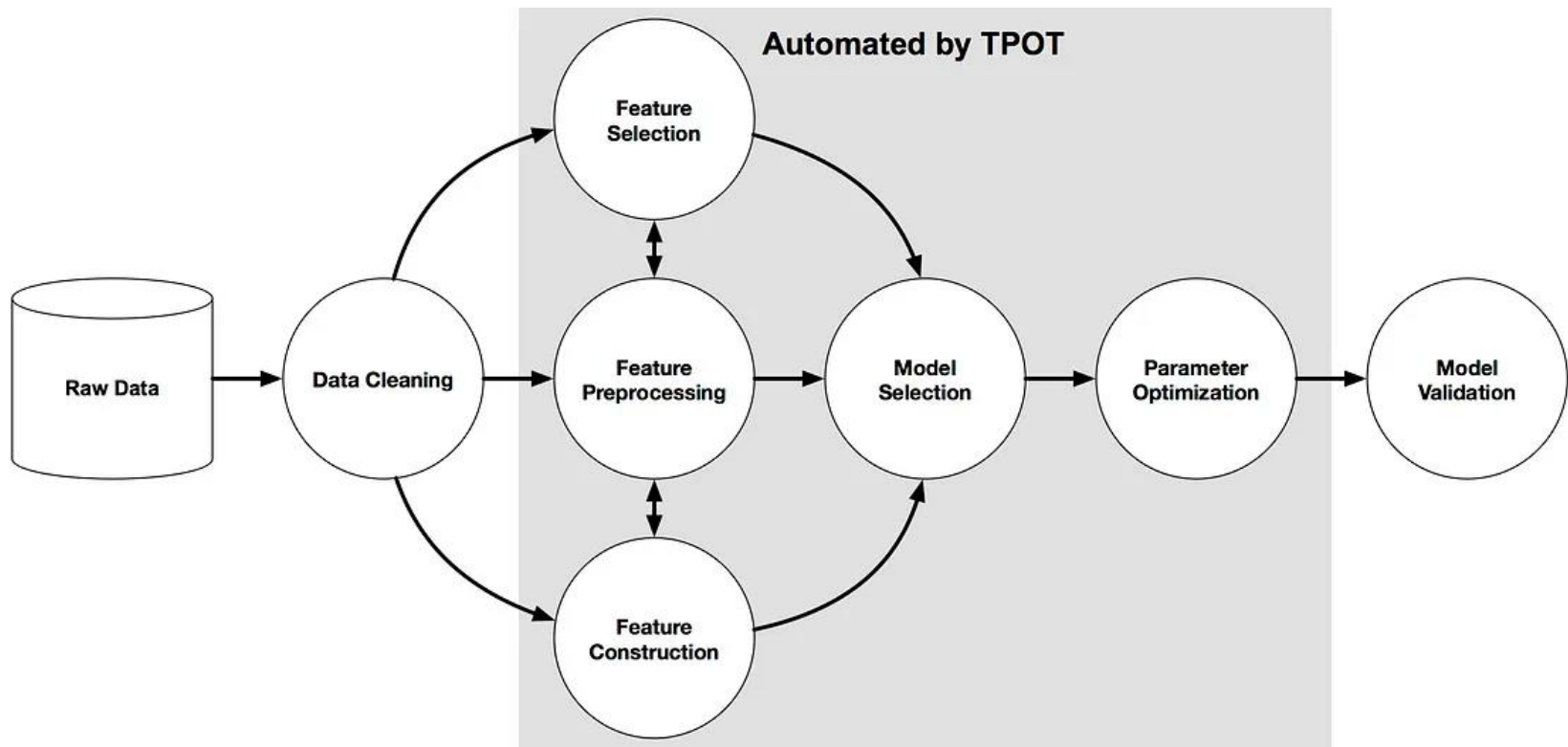
Automatiza as partes mais tediosas do aprendizado de máquina, como seleção de modelos e ajuste de hiperparâmetros.

## **Geração de Código**

Fornece o código Python para o melhor pipeline encontrado, permitindo ajustes posteriores.

## **Eficiência**

Acelera o desenvolvimento de modelos de aprendizado de máquina e ajuda a alcançar melhor desempenho nas tarefas de análise de dados.



# Revisão



# Árvores de Decisão

**Os modelos baseados em árvore de decisão são amplamente utilizados em machine learning. São capazes de resolver desde problemas bastante simples até problemas com alta complexidade.**

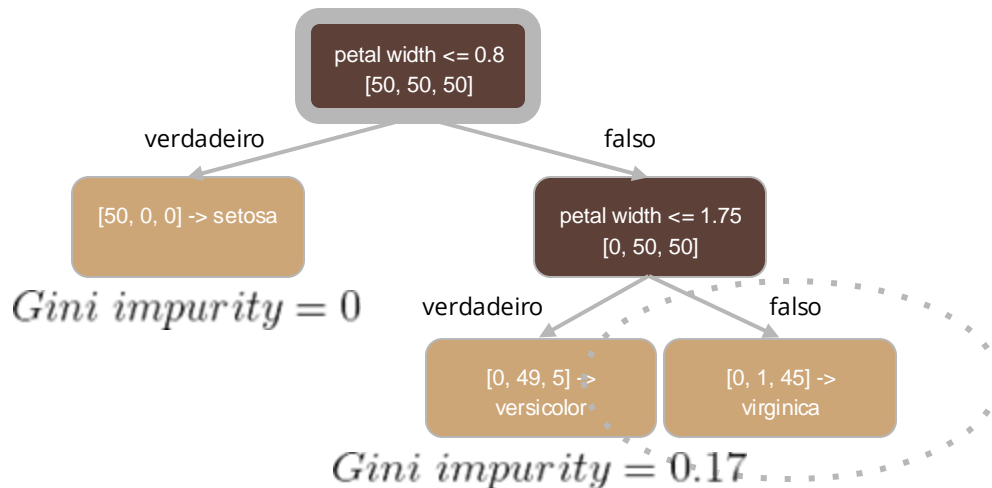


# Árvores de Decisão

## Gini Index

$$I_G = 1 - \sum_{j=1}^c p_j^2$$

$p_j$ : proportion of the samples that belongs to class  $c$  for a particular node



$$Gini\ impurity = 1 - (prob.\ de\ setosa)^2 - (prob.\ de\ versicolor)^2 - (prob.\ de\ virginica)^2$$

# Ensembles - XGBoost

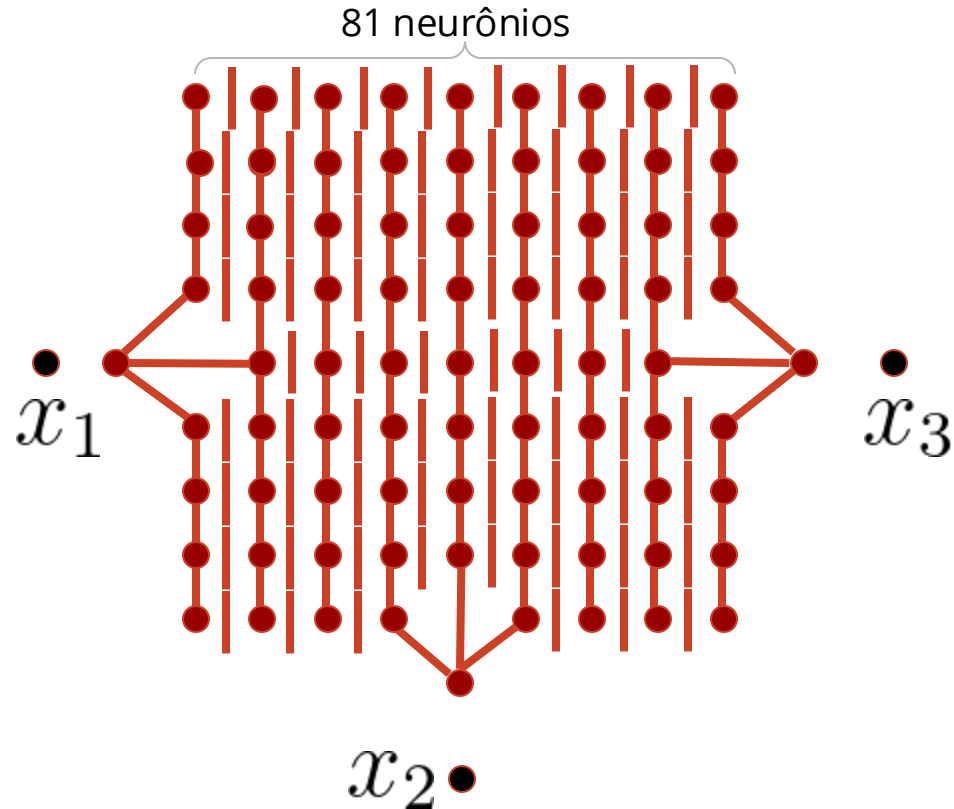
- O modelo XGBoost foi desenhado para trabalhar com datasets de alta complexidade
- Assim como o Gradient Boosting, ele é um modelo sequencial
- Normalmente, as árvores são limitadas a 6 níveis



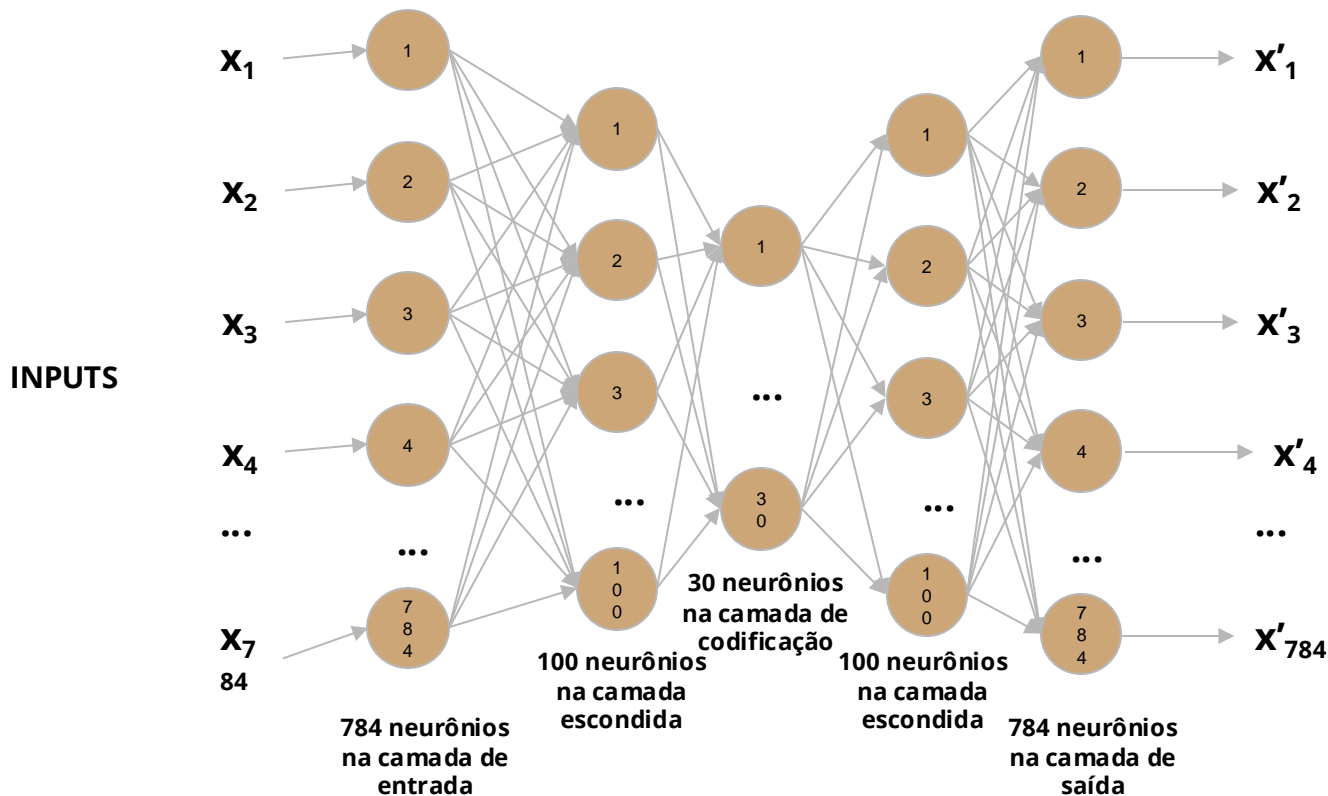
# Mapas autoorganizáveis

Passos do algoritmo:

1. Inicialização dos pesos da rede
2. Passos do treinamento repetidos até que não haja modificação significativa nos ajustes dos pesos ou que o número máximo de iterações seja atingidos:
  - a. Escolha da amostra de entrada da rede.  
Escolha uma das amostras pertencentes ao dataset de entrada
  - b. Determinar quem é o neurônio vencedor.  
Normalmente é utilizada a distância euclidiana entre a amostra selecionada e o peso dos neurônios
  - c. Atualização dos pesos do neurônios vencedor e dos neurônios vizinhos



# Autoencoders



Dúvidas?



Obrigada!