Algoritmos Clássicos

Prof. Edson Pedro Ferlin

Algoritmos Clássicos de IA

Busca

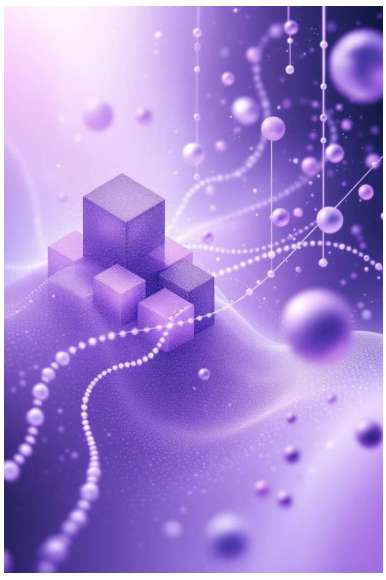
Algoritmos de busca em grafos e árvores

Aprendizado de Máquina

Algoritmos de classificação e regressão

Algoritmos Evolutivos

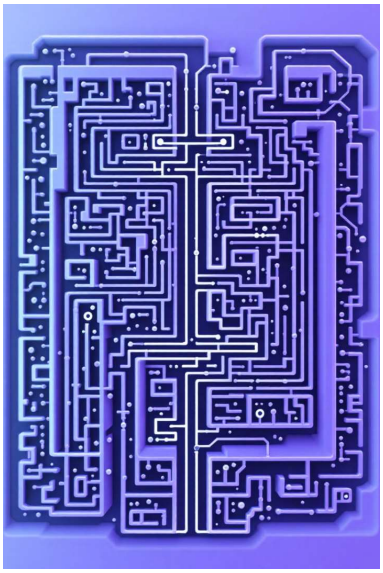
Algoritmos genéticos para otimização



Busca em Profundidade (DFS)

Objetivo: Explorar uma árvore ou grafo visitando os nós o mais profundamente possível antes de retroceder.
Aplicação: Resolver quebra-cabeças, como o jogo de labirinto.

- Comece do nó raiz
  - Ponto inicial da busca
- Explore filhos em sequência
  - Vá o mais profundo possível
- Retroceda quando necessário
  - Explore nós irmãos após chegar ao fim



Exemplo em Python do Algoritmo de Busca em Profundidade (DFS)

```
# Algoritmo de Busca em Profundidade

def dfs(graph, start, visited=None):
    if visited is None:
        visited = set()
    visited.add(start)
    for neighbor in graph[start]:
        if neighbor not in visited:
            dfs(graph, neighbor, visited)
    return visited

# Grafo Exemplo
graph = {
    'A': ['B', 'C'],
    'B': ['A', 'D', 'E'],
    'C': ['A', 'F'],
    'D': ['B'],
    'E': ['B', 'F'],
    'F': ['C', 'E']
}

# Testando DFS
print(dfs(graph, 'A'))
```

Busca em Largura (BFS)

Objetivo: Explorar uma árvore ou grafo nível por nível.
Aplicação: Encontrar o caminho mais curto entre dois pontos em um grafo.

1. Inicie no nó raiz
   Ponto de partida da busca
2. Explore nível por nível
   Visite todos os vizinhos
3. Use uma fila
   Armazene nós visitados

Exemplo em Python do Algoritmo de Busca em Largura (BFS)

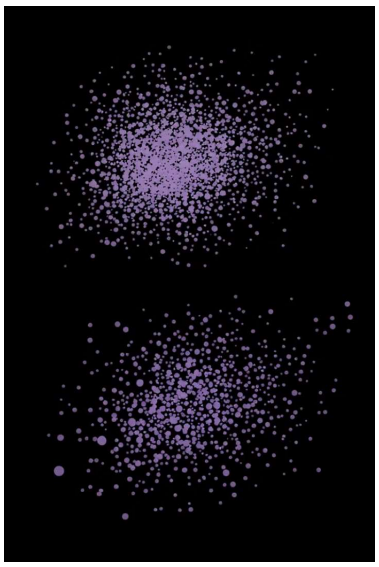
```
# Algoritmo de Busca em Largura

from collections import deque

def bfs(graph, start):
    visited = set()
    queue = deque([start])
    visited.add(start)

    while queue:
        node = queue.popleft()
        print(node, end=" ")
        for neighbor in graph[node]:
            if neighbor not in visited:
                visited.add(neighbor)
                queue.append(neighbor)
```

```
# Grafo Exemplo
graph = {
    'A': ['B', 'C'],
    'B': ['A', 'D', 'E'],
    'C': ['A', 'F'],
    'D': ['B'],
    'E': ['B', 'F'],
    'F': ['C', 'E']
}

# Testando BFS
bfs(graph, 'A')
```

k-Nearest Neighbors (k-NN)

Objetivo: Classificar um ponto com base nos k vizinhos mais próximos.
Aplicação: Classificação de imagens, reconhecimento de padrões.

1. Calcular distâncias
   Entre ponto de teste e todos os pontos de treinamento
2. Escolher k vizinhos
   Selecionar os k pontos mais próximos
3. Atribuir classe
   Usar a classe mais comum entre os vizinhos

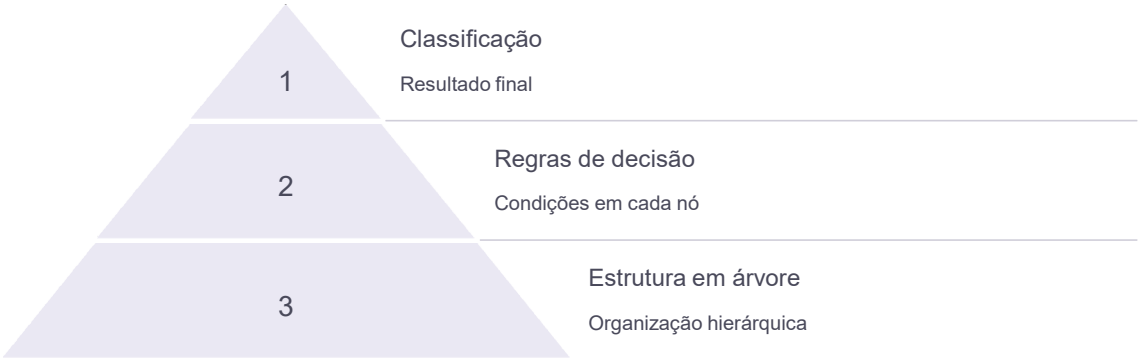


Exemplo em Python do Algoritmo k-Nearest Neighbors (k-NN)

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split

# Carregar dados de exemplo
data = load_iris()
X = data.data
y = data.target

# Dividir em conjunto de treino e teste
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
random_state=42)

# Criar e treinar o modelo k-NN
knn = KNeighborsClassifier(n_neighbors=3)
knn.fit(X_train, y_train)

# Fazer previsões
predictions = knn.predict(X_test)
print(predictions)
```

Árvore de Decisão

Objetivo: Classificar dados baseados em um conjunto de regras de decisão em formato de árvore.
Aplicação: Diagnóstico médico, análise de crédito.

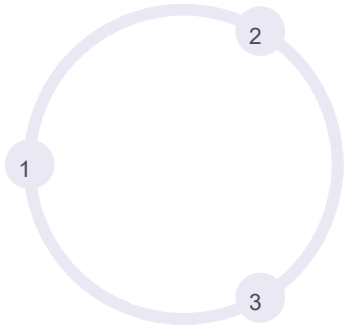


Exemplo em Python do Algoritmo da Árvore de Decisão

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split

# Carregar dados de exemplo
data = load_iris()
X = data.data
y = data.target

# Dividir em conjunto de treino e teste
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
random_state=42)

# Criar e treinar a Árvore de Decisão
dtree = DecisionTreeClassifier()
dtree.fit(X_train, y_train)

# Fazer previsões
predictions = dtree.predict(X_test)
print(predictions)
```

K-means Clustering

Objetivo: Agrupar dados em k clusters, onde os pontos dentro de um cluster são mais semelhantes entre si.
Aplicação: Segmentação de mercado, compressão de imagem.



Agrupamento

Pontos similares em clusters

Segmentação

Divisão de mercado ou imagens

Otimização

Minimizar distâncias dentro dos clusters

Exemplo em Python do Algoritmos K-Means Clustering

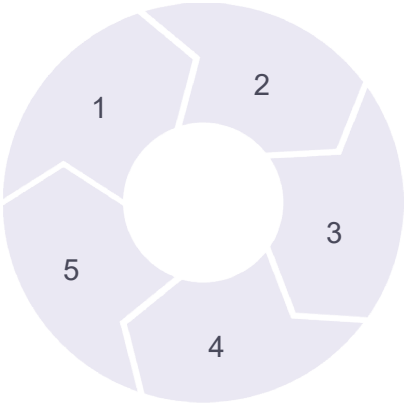
```
from sklearn.cluster import KMeans
import numpy as np

# Dados de exemplo
X = np.array([[1, 2], [2, 3], [3, 3], [8, 7], [8, 8], [25, 80]])

# Aplicando K-means
kmeans = KMeans(n_clusters=2)
kmeans.fit(X)

# Previsão dos clusters
print(kmeans.labels_)
```

Algoritmos Genéticos

Objetivo: Utilizar operadores evolutivos para otimizar uma função, baseado no processo de seleção natural.
Aplicação: Problemas de otimização complexos, como agendamento de tarefas e design de sistemas.

Inicialização

População de soluções aleatórias



Seleção

Escolha das melhores soluções

Cruzamento

Combinação de soluções

Mutação

Alterações aleatórias

Substituição

Atualização da população

Exemplo em Python do Algoritmo Genético (pseudo-código)

```
# Pseudo-código simplificado para um Algoritmo Genético

def fitness_function(solution):
    # Avaliar a qualidade da solução
    return score

def crossover(parent1, parent2):
    # Combinar dois pais para gerar um filho
    return offspring

def mutate(offspring):
    # Mutar uma solução
    return mutated_offspring
```

```
def genetic_algorithm():
    # Inicializar população
    population = initialize_population()

    for generation in range(num_generations):
        # Avaliar fitness
        population = sorted(population, key=fitness_function)

        # Seleção dos melhores pais
        parents = select_parents(population)

        # Cruzamento e mutação
        offspring = crossover(parents)
        offspring = mutate(offspring)

        # Substituição da população
        population = replace_population(population, offspring)

    # Retornar a melhor solução
    return best_solution
```

Desafios e Melhorias

Desafios

- Overfitting em modelos
- Escalabilidade de algoritmos
- Interpretação dos resultados

Melhorias

- Regularização contra overfitting
- Algoritmos paralelos para escala
- Modelos explicáveis

Contato

eferlin@live.com

(BLOG) professorferlin.blogspot.com

(SITE) professorferlin.webnode.com.br

(YOUTUBE) ProfEdsonPedroFerlin