Técnicas de Seleção de Subconjunto

Grupo 6

Matheus Araujo
Gabriel Teixeira
Marcelo Mota

Definição

O processo é mais custoso que a ordenação dos atributos.

Tornam-se intratáveis quando o número de atributos é muito grande.

Objetivo

Obter um conjunto de atributos que melhore o desempenho do modelo.

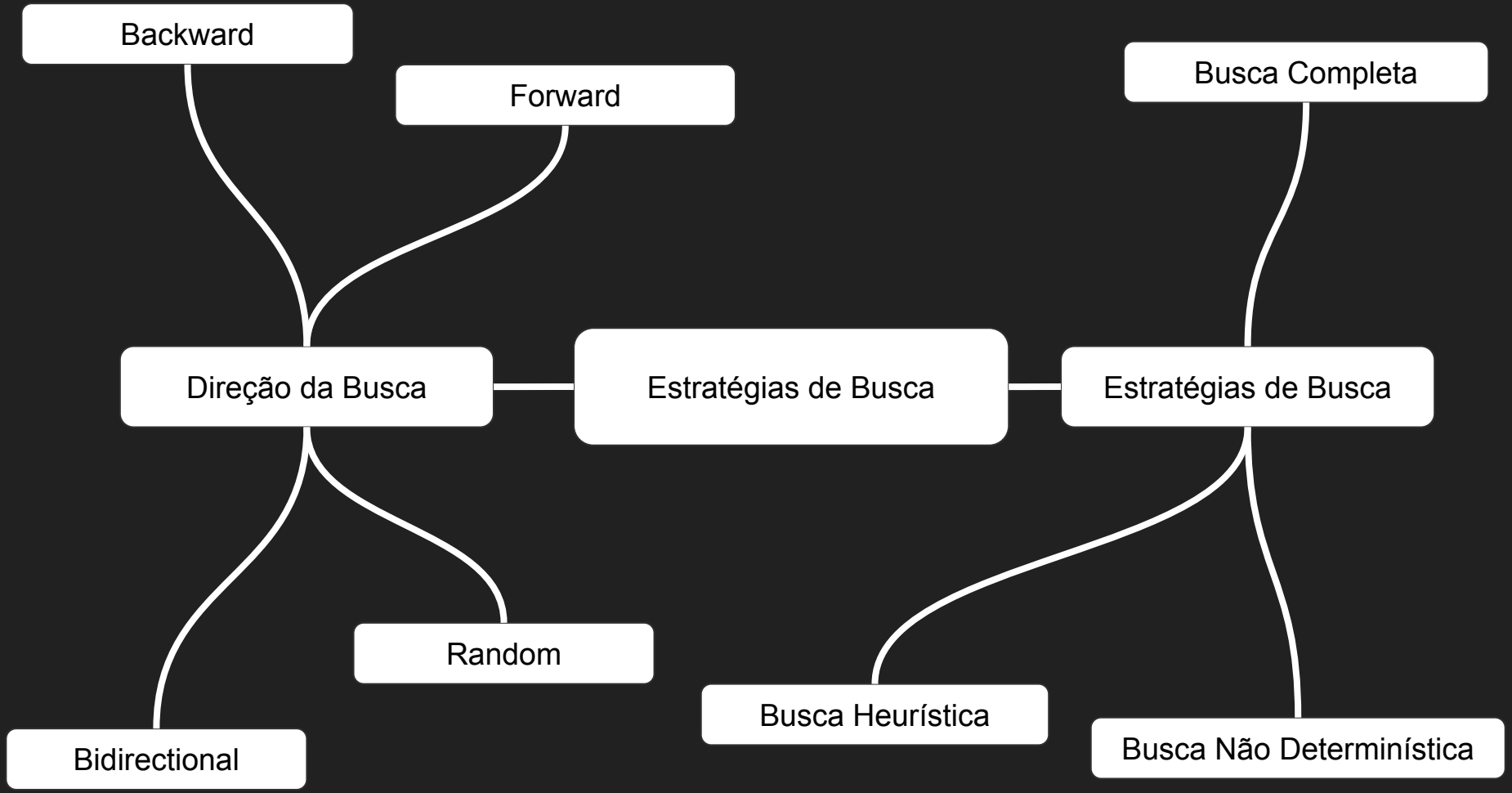
Passos fundamentais (Blum & Langley, 1997)

1° - Definir os pontos de partida e a direção da busca;

2° - Escolher a estratégia de busca;

3° - Selecionar critérios de avaliação;

4° - Determinar o critério de parada.


Backward
Forward
Busca Completa
Direção da Busca
Estratégias de Busca
Estratégias de Busca
Random
Bidirectional
Busca Heurística
Busca Não Determinística

Critérios de Parada

Encerrar a busca quando:

- Opção 1 - Todos os subconjuntos forem testados (EXAUSTIVA);
- Opção 2 - Um critério de desempenho for atingido.

Algumas considerações

As técnicas de seleção de subconjunto podem ser aplicada em tarefas supervisionadas e não supervisionada.

A qualidade do agrupamento de dados impacta os resultados.

Exemplo

Implementar uma estratégia de busca por geração aleatória usando uma técnica de Seleção de Subconjunto, onde geramos aleatoriamente um subconjunto de possíveis soluções e avaliamos qual delas é a melhor.

Vamos definir uma função que avalia uma solução (neste caso, a soma dos quadrados dos elementos de um vetor).

```python
# Função objetivo: soma dos quadrados dos elementos
def objetivo(solucao):
    return sum(x**2 for x in solucao)
```

Vamos gerar aleatoriamente soluções possíveis (subconjuntos).

```python
# Função para gerar uma solução aleatória (subconjunto de números entre -10 e 10)
def gerar_solucao(tamanho=5, limite=-10, max_value=10):
    return [random.randint(limite, max_value) for _ in range(tamanho)]
```

Para cada solução gerada, avaliamos a qualidade usando a função objetivo.

```
# Parâmetros da busca
num_solucoes = 100  # Número de soluções geradas
tamanho_subconjunto = 5  # Tamanho do subconjunto (tamanho do vetor)
melhor_solucao = None
melhor_valor = float("inf")  # Inicializando com um valor muito alto
```

Escolhemos a melhor solução gerada.

```python
# Estratégia de busca por geração aleatória
for _ in range(num_solucoes):
    solucao_aleatoria = gerar_solucao(tamanho_subconjunto)
    valor = objetivo(solucao_aleatoria)

    if valor < melhor_valor:  # Se encontramos uma solução melhor, atualizamos
        melhor_valor = valor
        melhor_solucao = solucao_aleatoria
```

Resultado

```python
# Resultado final
print("Melhor solução encontrada:", melhor_solucao)
print("Valor da função objetivo:", melhor_valor)
```

```
PS C:\Users\mathe\Desktop\Aprendizagem de Máquina> python -u "c:\Users\mathe\Desktop\Aprendizagem de Máquina\unidade_01\busca_aleatoria.py"
Melhor solução encontrada: [1, 0, -1, -1, -5]
Valor da função objetivo: 28
PS C:\Users\mathe\Desktop\Aprendizagem de Máquina> python -u "c:\Users\mathe\Desktop\Aprendizagem de Máquina\unidade_01\busca_aleatoria.py"
Melhor solução encontrada: [-3, 2, 1, -1, 1]
Valor da função objetivo: 16
PS C:\Users\mathe\Desktop\Aprendizagem de Máquina> python -u "c:\Users\mathe\Desktop\Aprendizagem de Máquina\unidade_01\busca_aleatoria.py"
Melhor solução encontrada: [-2, 0, -1, 3, 1]
Valor da função objetivo: 15
PS C:\Users\mathe\Desktop\Aprendizagem de Máquina>
```

Referências

FACELI, Katti et al. Inteligência artificial: uma abordagem de aprendizado de máquina. . Rio de Janeiro: LTC. . Acesso em: 13 nov. 2023. , 2011

Aurélien Géron. Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems. "O'Reilly Media, Inc.", 2019.