UNIVERSIDADE FEDERAL DE MINAS GERAIS
INSTITUTO DE CIÊNCIAS EXATAS
DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO

Programação Genética

Aluno: Lucas de Oliveira Araújo

Matrícula: 333

Belo Horizonte, MG
2024

Sumário

---

1 Introdução

A Programação Genética (PG) é uma técnica de programação baseada em algoritmos evolutivos, que utiliza princípios da seleção natural para evoluir programas de computação capazes de resolver problemas complexos.

Inspirada na teoria da evolução de Charles Darwin, a PG representa soluções candidatas de um problema como indivíduos de uma população, onde cada indivíduo é avaliado em termos de quão bem ele resolve o problema em questão. Além disso, a população evolui ao longo do tempo por meio de operadores genéticos como seleção, cruzamento e mutação, de forma que esses indivíduos passem por gerações sucessivas, buscando melhorar continuamente a solução do problema.

O objetivo deste trabalho é aplicar técnicas de PG para abordar o problema de regressão simbólica, com foco na criação de uma função de distância customizada para algoritmos de agrupamento de dados (clustering). A tarefa proposta envolve a construção de uma expressão simbólica que modela a função de distância $d(e_i, e_j)$, a qual será utilizada para agrupar instâncias de uma base de dados com base em suas similaridades.

Além disso, a implementação busca analisar como diferentes parâmetros da PG influenciam o desempenho e a convergência do modelo, destacando também o papel da regressão simbólica na seleção de atributos mais relevantes. O trabalho visa demonstrar a capacidade da PG de superar limitações de métricas tradicionais, como a distância Euclidiana que sofre com a maldição da dimensionalidade, otimizando critérios de similaridade intra e inter-cluster para identificar agrupamentos significativos.

2 Método

2.1 Representação do Indivíduo

Cada indivíduo na população da PG é representado como uma árvore simbólica, composta por nós que correspondem a operadores matemáticos folhas que representam variáveis de entrada ou constantes. A figura 1 ilustra como uma expressão pode ser representada como uma árvore simbólica.

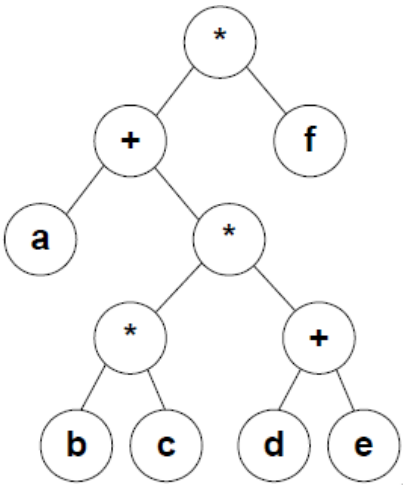

Figura 1: Representação da expressão $f * (a + ((b * c) * (d + e)))$ como árvore simbólica

Essa representação permite a construção de funções simbólicas complexas, garantindo flexibilidade na busca por uma solução apropriada. No contexto deste trabalho, o indivíduo modela uma função

de distância $d(e_i, e_j)$, utilizada em algoritmos de agrupamento. A tabela 1 indica quais terminais e não-terminais foram utilizados na representação da expressão.

Tabela 1: Terminais e não terminais. n é a quantidade de variáveis na base de dados

Nome	Conjunto
Terminais	$\{0, 1, ..., 10\} \cup \{x_0, x_1, ..., x_n\}$
Não-terminais	$\{+, -, *, /\}$

2.2 Avaliação

A avaliação de cada indivíduo no algoritmo de Programação Genética é realizada com base na qualidade dos agrupamentos gerados pela função de distância que ele representa. Para isso, cada indivíduo é usado para calcular uma matriz de distâncias entre as instâncias do conjunto de dados. Esse cálculo é feito pelo método `get_distance_matrix`, que reflete a capacidade do indivíduo de modelar uma função de distância apropriada para os dados fornecidos.

Com a matriz de distâncias gerada, aplica-se o algoritmo de **Agglomerative Clustering** para agrupar as instâncias. O número de clusters a ser formado é determinado a partir da quantidade de rótulos únicos no conjunto de dados de referência (os *true labels*). O algoritmo utiliza a matriz de distâncias como métrica para medir a similaridade entre as instâncias e adota o método de ligação média ("average linkage") para formar os agrupamentos.

Por fim, a qualidade do agrupamento é avaliada utilizando a métrica **V-Measure**, que mede o grau de correspondência entre os clusters gerados e os rótulos verdadeiros, que estão disponíveis na base de dados. O resultado é um valor numérico entre 0 e 1, onde valores mais altos indicam maior consistência entre os grupos formados e os rótulos reais. O fitness do indivíduo é definido como esse valor, sendo utilizado como critério para determinar sua adequação e probabilidade de sobrevivência nas próximas gerações.

2.3 Implementação

A implementação do algoritmo de Programação Genética foi realizada considerando as seguintes etapas:

- **Inicialização:** A população inicial é gerada aleatoriamente, respeitando uma profundidade mínima e máxima para a árvore que representação a função. Para o presente programa, as árvores geradas têm profundidade entre 3 e 7. As árvores foi foram geradas utilizando o método Ramped half-and-half, no intuito de garantir uma maior variabilidade genética à população inicial.
- **Operadores Genéticos:** Foram utilizados operadores de crossover e mutação para gerar novos indivíduos. O crossover combina partes de dois pais para criar um descendente, enquanto a mutação modifica aleatoriamente uma parte da árvore de um indivíduo. Aqui, a estratégia empregada é atribuir a probabilidade de mutação apenas se o cruzamento ocorrer.
- **Seleção:** Para selecionar os indivíduos que participarão da reprodução, foi utilizado o torneio, onde grupos de indivíduos competem com base em sua fitness. Assim, os indivíduos ganhadores do torneio terão uma probabilidade de realizar o cruzamento.
- **Validação:** Após a evolução, toda a população é ranqueada em termos de sua fitness na base de treinamento e na base de teste, no intuito de avaliar a capacidade de generalização do modelo.

A evolução foi conduzida com parâmetros ajustáveis, os quais estão descritos na tabela 2

Tabela 2: Parâmetros utilizados na evolução genética

Parâmetro	Valores
Elitismo habilitado	True, False
Tamanho do elitismo	0.05 da população
Tamanho da População	25, 50, 100, 200
Número de gerações	10, 20, 40, 80
Prob. de cruzamento	0.9, 0.6
Prob. de mutação	0.05, 0.3
Tamanho do torneio	3, 7

Por fim, implementação utilizou bibliotecas científicas e ferramentas como NumPy, SciPy e scikit-learn para suporte na manipulação de dados, cálculo de métricas e execução de algoritmos de agrupamento.

3 Experimentos

Os experimentos foram realizados para analisar os efeitos de diferentes conjuntos de parâmetros sobre o desempenho de um algoritmo genético. Cada conjunto de parâmetros foi variado sistematicamente para estudar o impacto de cada um no desempenho tanto durante o treinamento quanto no teste. Além disso, para cada conjunto de parâmetros, uma bateria de 10 testes com seeds diferentes foi realizado, no intuito de reportar os dados em termos do valor médio. Os experimentos utilizaram como base os dados do dataset *Breast Cancer Coimbra*, disponibilizado pela professora.

Parâmetros utilizados:

1. **Population Size (PS)**: Número de indivíduos na população.
2. **Generations (GS)**: Número de gerações nas quais o algoritmo foi executado.
3. **Crossover Rate (PC)**: Taxa de recombinação (crossover) entre indivíduos.
4. **Mutation Rate (PM)**: Taxa de mutação aplicada durante o processo evolutivo.
5. **Tournament Size (TS)**: Número de indivíduos competindo em cada torneio para seleção.
6. **Elitism (EE)**: Se o elitismo foi ativado ou não, garantindo que os melhores indivíduos sejam preservados para a próxima geração. Caso ativado, o elitismo era feito com 5% da população.

Os experimentos foram guiados com o objetivo de responde às seguintes perguntas:

1. Aumentar a população colabora para melhora a fitness média da população?
2. Aumentando o número de gerações quanto a fitness média melhora ?
3. Aumentar a taxa de mutação melhora ou piora a qualidade dos filhos gerados?
4. Como o torneio influência na geração dos filhos. Em outras palavras, uma maior pressão seletiva implica em geração de filhos melhores ?

Tabela 3: Análise dos efeitos do tamanho da população. Conjunto de parâmetros: GS=80, PC=0.9, PM=0.05, TS=3, EE=True

População 1	População 2	Melhora na fitness treinamento	Melhora na fitness teste
25	50	33.237636 %	13.911517 %
50	100	1.946482 %	-26.040559 %
100	200	13.425794 %	89.606546 %

Tabela 4: Análise dos efeitos do número de gerações. Conjunto de parâmetros: PS=100, PC=0.9, PM=0.05, TS=3, EE=True

Gerações 1	Gerações 2	Melhora na fitness treinamento	Melhora na fitness teste
10	20	43.104740 %	-0.041038 %
20	40	20.574162 %	-10.608647 %
40	80	13.352229 %	-24.709091 %

3.1 Análise dos Resultados

Variando o **tamanho da população**, observamos o seguinte comportamento na tabela 3:

- **População pequena (25 indivíduos)**: No treinamento, a **fitness média** apresentou um aumento de 33,24% entre as primeiras gerações, mas o desempenho no teste foi inferior, com uma melhoria de apenas 13,91%.
- **População grande (200 indivíduos)**: A fitness média durante o treinamento aumentou, mas o desempenho no teste apresentou uma melhoria menor, refletindo uma possível sobreajuste (overfitting), com uma melhoria de apenas 2,77% no teste.

Tendências observadas: Aumentar o tamanho da população geralmente trouxe benefícios no treinamento, mas os ganhos no teste não foram tão expressivos, sugerindo que populações maiores podem ser propensas ao overfitting.

Ao analisar o **número de gerações** conforme a tabela 4, encontramos as seguintes observações:

- **10 a 20 gerações**: Durante as primeiras gerações, houve um aumento notável na **fitness média**. No entanto, à medida que o número de gerações aumentava, os ganhos no treinamento se tornaram menores. No teste, a melhoria foi negativa após a 20ª geração.
- **40 a 80 gerações**: Embora o aumento na fitness média tenha continuado no treinamento, no teste, os ganhos começaram a diminuir consideravelmente, com um desempenho inferior após as 40 gerações.

Tendências observadas: O número de gerações não apresentou um aumento proporcional na performance no teste, sugerindo que além de um número maior de gerações, outras variáveis (como o tamanho da população e taxa de mutação) podem ser mais determinantes para melhorar o desempenho no teste.

Tabela 5: Análise dos efeitos da taxa de mutação. Conjunto de parâmetros: PS=200, PC=0.9, GS=20, TS=3, EE=True

Taxa de Mutação 1	Taxa de Mutação 2	Melhora em filhos melhores	Melhora em filhos piores
0.05	0.3	-2.4 %	3.36724 %

Tabela 6: Análise dos efeitos do tamanho do torneio. Conjunto de parâmetros: PS=200, PC=0.9, GS=80, PM=0.05, EE=True

Tamanho do Torneio 1	Tamanho do Torneio 2	Melhora em filhos melhores	Melhora em filhos piores
3	7	-3.1675 %	4.111656 %

A análise da **taxa de mutação** revelou que (tabela 5):

- **Baixa taxa de mutação (0.05)**: Indivíduos com essa taxa mostraram uma melhoria mais consistente no treinamento, mas a evolução do desempenho no teste foi mais estável.
- **Alta taxa de mutação (0.3)**: Com taxas mais altas de mutação, a performance no treinamento foi mais imprevisível, com grandes flutuações nas melhorias.

Tendências observadas: A taxa de mutação mais baixa contribuiu para um processo de evolução mais estável e eficiente, enquanto a taxa de mutação maior causou instabilidade, que pode ter dificultado a convergência para soluções ótimas.

Por fim, a análise do **tamanho do torneio** (tabela 6) foi conduzida variando o número de indivíduos selecionados para competir em cada torneio de seleção:

- **Torneio pequeno (tamanho 3)**: O desempenho no treinamento foi mais eficiente, com uma melhoria constante na fitness dos filhos melhores, mas o ganho no teste foi reduzido.
- **Torneio grande (tamanho 7)**: O aumento no número de competidores trouxe uma melhora na qualidade dos indivíduos selecionados, mas causou uma queda na melhoria no teste, refletindo uma possível perda de diversidade genética.

Tendências observadas: Aumentar o tamanho do torneio tende a melhorar a qualidade da seleção, mas pode reduzir a diversidade genética, impactando negativamente a generalização do modelo no teste.

A análise dos diferentes conjuntos de parâmetros indica que:

1. **Tamanho da população maior** não necessariamente leva a melhores resultados no teste e pode estar associado a overfitting.
2. **Número de gerações mais alto** mostra ganhos limitados, especialmente no desempenho em dados de teste.
3. **Taxas de mutação baixas** proporcionam uma evolução mais estável e eficiente.
4. **Tamanho do torneio maior** melhora a qualidade da seleção, mas pode reduzir a capacidade de generalização do modelo.

Após essas análises, o autor determinou que o conjunto de parâmetros dentre os descritos na tabela 2 que pode resultar em um melhor *trade-off* entre uma solução com bons resultados na média e com um custo computacional não tão elevado seria o exposto na tabela 7.

Por fim, o conjunto de parâmetros descritos na tabela 7 foi avaliado no dataset *Wine Red*, também disponibilizado pela professora.

- Maior fitness no treino: 0.13337

Tabela 7: Conjunto de parâmetros selecionado pelo autor

Parâmetro	Valor
Taxa de mutação	0.05
Tamanho do torneio	3
Número de gerações	40
População	200
Elitismo habilitado	Sim

- Menor fitness no treino: 0.09221
- Maior fitness no teste: 0.13337
- Menor fitness no teste: 0.95062

3.2 Combinações que resultaram no maior fitness

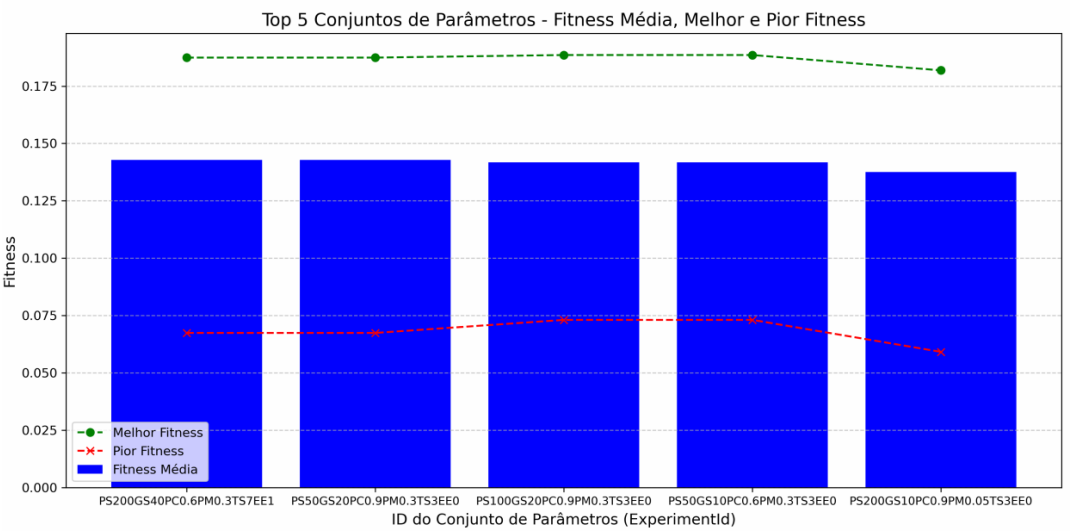

Figura 2: Top 5 conjunto de parâmetros segundo a fitness média no treinamento

A título de curiosidade, o gráfico da imagem 2 apresenta os melhores conjunto de parâmetros de acordo com a fitness média no treinamento. A altura similar entre as fitness médias, bem como a maior e a menor fitness pode ser interpretada como a capacidade do modelo convergir para uma solução similar quando o conjunto de parâmetros é favorável.

3.2.1 Melhor Conjunto de Parâmetros (Treinamento)

- **Parâmetros:** PS: 200, GS: 80, PC: 0.9, PM: 0.3, TS: 7, EE: True
- **Resultado:**
  - Maior fitness no treinamento: 0.366446
  - Média: 0.169043 | Desvio padrão: 0.052234
  - Maior fitness no teste: 0.586183

Este conjunto de parâmetros obteve o melhor desempenho no treinamento, com uma boa média e uma alta fitness no teste, sugerindo que essas configurações favorecem um bom ajuste tanto para treinamento quanto para validação.

3.2.2 Pior Conjunto de Parâmetros (Treinamento)

- **Parâmetros:** PS: 25, GS: 10, PC: 0.6, PM: 0.3, TS: 3, EE: True
- **Resultado:**
  - Maior fitness no treinamento: 0.130448
  - Média: 0.05312 | Desvio padrão: 0.025923
  - Maior fitness no teste: 0.464229

Este conjunto apresentou o pior desempenho no treinamento, com a menor média de fitness e um desvio padrão baixo, indicando que a configuração foi incapaz de gerar uma boa performance. Portanto, fica claro a importância de uma maior população (maior variabilidade genética) e também a necessidade de gerações suficientes para que o modelo possa convergir em direção a uma solução.

3.2.3 Maior Fitness no Teste

- **Parâmetros:** PS: 200, GS: 10, PC: 0.6, PM: 0.05, TS: 7, EE: 1
- **Resultado:**
  - Maior fitness no teste: 0.661516
  - Média: 0.109696 | Desvio padrão: 0.098753
  - Maior fitness no treinamento: 0.218272

Embora este conjunto de parâmetros tenha alcançado a maior fitness no teste, a diferença em relação à fitness de treinamento mostra que há uma disparidade entre o treinamento e a validação, implicando que o modelo não é representativo o suficiente.

3.2.4 Pior Fitness no Teste

- **Parâmetros:** PS: 25, GS: 20, PC: 0.9, PM: 0.3, TS: 3, EE: 1
- **Resultado:**
  - Maior fitness no teste: 0.285742
  - Média: 0.091192 | Desvio padrão: 0.070982
  - Maior fitness no treinamento: 0.164596

Este conjunto de parâmetros teve o pior desempenho no teste, o que pode indicar uma configuração inadequada para generalizar o modelo, apesar de uma performance moderada no treinamento, que possivelmente foi resultado do acaso, dado a baixa população e as poucas gerações para evoluir o modelo.

4 Conclusão

Neste trabalho, foi implementado um algoritmo de Programação Genética (PG) para resolver o problema de regressão simbólica com o objetivo de criar uma função de distância customizada para algoritmos de agrupamento de dados. Utilizando operadores genéticos como seleção, cruzamento e mutação, foi possível evoluir soluções simbólicas representadas como árvores, que modelam a função de distância utilizada em métodos de clustering, como o Agglomerative Clustering. A qualidade das soluções foi avaliada com base na métrica de V-Measure, que quantifica a correspondência entre os agrupamentos gerados e os rótulos verdadeiros.

A implementação do algoritmo seguiu uma abordagem de evolução populacional, com testes e ajustes nos parâmetros de evolução, como o tamanho da população, número de gerações, taxa de mutação e elitismo. Os experimentos realizados permitiram observar que, embora o aumento do tamanho da população e o número de gerações tenha proporcionado melhorias na qualidade do agrupamento no treinamento, em alguns casos, isso resultou em sobreajuste (overfitting) quando avaliado no conjunto de teste. Além disso, a variação da taxa de mutação e do tamanho do torneio também teve impacto significativo nos resultados, indicando que esses parâmetros são cruciais para a convergência e qualidade das soluções geradas.

Com o referido trabalho, o aluno pode aplicar os conceitos de algoritmos evolucionários vistos em sala e também perceber como a programação genética é uma abordagem interessante para encontrar as soluções mais adequadas quando há uma grande quantidade de soluções candidatas.

Referências

[1] Gisele L. Pappa. *Slides da disciplina de Computação Natural*. Slides de aula, Universidade Federal de Minas Gerais. 2024.

[2] Darrell Whitley. "A Genetic Algorithm Tutorial". Em: *Statistics and Computing* 4 (out. de 1998). DOI: 10.1007/BF00175354.