Universidade Federal de Minas Gerais

Computação Natural

Programação Genética: Regressão Simbólica

Professor(a): Gisele Lobo Pappa
Autor: Pedro Nascimento Costa

1 Introdução

Um dos problemas mais associados com programação genética, é o de regressão simbólica, dito isso, esse tipo de problema pode ser descrito como; dado uma representação de quantidades de amostras, provenientes de uma função desconhecida, representada através de uma tupla **(X,Y)**, na qual **X** representa **N** variáveis e **Y** representa o resultado de uma função, deseja-se aproximar ao máximo e possivelmente descobrir a função.

Com a técnica da programação genética, existe uma tradução desse problema para uma representação em genótipo, de tal forma que possa ser interpretado por um algoritmo, o outro quesito necessário é a apresentação de um fenótipo, no qual o genótipo se traduz. Assim, o primeiro passo é a interpretação do problema em uma árvore de expressões(genótipo) com o resultado da expressão sendo o fenótipo, como por exemplo:

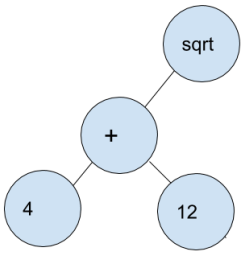

Figura 1: Representação em árvore, exemplo.

Nesse indivíduo, tem se o resultado de 4, essencialmente, primeiro se faz a soma de 4 e 12, depois a raiz quadrada do resultado da soma. Dessa forma, somos então capazes de analisar diferentes expressões e compará-las.

2 Desenvolvimento

2.1 Considerações

O trabalho foi desenvolvido em python, fazendo uso de algumas bibliotecas disponíveis. Em particular:

1. sys: Uso do *argv*, para passar entradas no algoritmo pela linha de comando ao rodá-lo.
2. pandas: Leitura dos arquivos .csv
3. math: Uso de operações, raiz quadrada(*sqrt()*), logaritmo natural(*log()*).
4. copy, a função de cópia profunda(*deepcopy()*).
5. functools: função *reduce()*.
6. random: tratamento de sorteios de números 'aleatórios'.
7. matplotlib: plotagem dos gráficos aqui presentes.

Demais partes, presentes no algoritmo, foram implementadas, como a representação da árvore e os algoritmos descritos mais abaixo, por exemplo.

Para executar o código presente, deve-se rodar:

python3 main.py [1] [2] [3] [4] [5] [6] [7] [8] [9] [10] [11]
No qual:

1. Tamanho da sua população
2. Tamanho máximo da árvore(começa em 0 para tamanho 1, então para tamanho 7, esse parâmetro deve ser 6)
3. Chance de Crossover (entre 0 e 1)
4. Chance de Mutação (entre 0 e 1)
5. Tamanho do torneio
6. Número de gerações
7. Uso de elitismo (yes/no)
8. Caminho do arquivo de treinamento
9. Caminho do arquivo de teste
10. Mudança feita(pop-50/gen-100, etc), usado para separar os plots.
11. Caso ('20' para rodar parte preliminar)

2.2 Metodologia

O diagrama disponível abaixo, foi o seguido para realização do fluxo do trabalho:

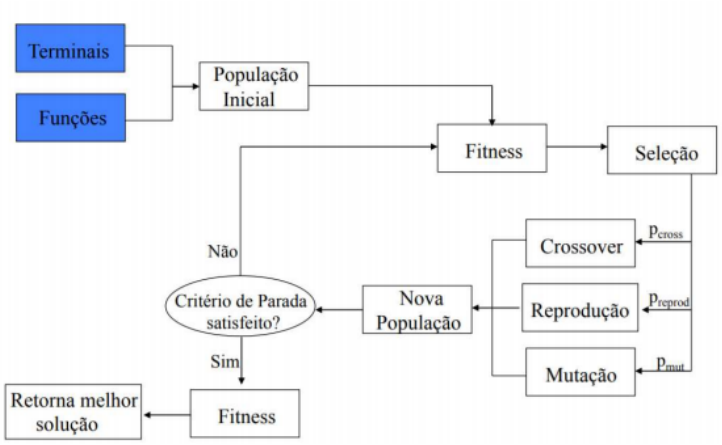

Figura 2: Diagrama, fluxo e estados de uma Programação Genética.

2.2.1 Terminais e Funções

Dentre terminais e funções, foram escolhidos alguns em particular para poderem compor a estruturação da árvore.

Terminais:

Um número aleatório entre (0,1), de até 4 casas decimais. Escolhido dessa forma, para que os terminais não variáveis, não tenham um impacto significativamente grande nos resultados.

Uma variável, no caso, podemos ter **N** variáveis, número determinadado no caso, pelos datasets de entrada, por exemplo, no dataset concrete disponibilizado, existiriam 8 possíveis variáveis.

Operadores:

Podem ser **Binários**: (+, -, /, *), soma, subtração, divisão ou multiplicação, ou **Unários**: (sqrt, log, **2), raíz quadrada, logaritmo natural, ou elevado ao quadrado.

2.2.2 População Inicial

A população inicial é gerada através do algoritmo *Ramped Half and Half(RHH)*, fazendo uso de duas formas de se gerar uma árvore, as *Grow Trees(GT)* e as *Full Trees(FT)*. De maneira sucinta, dado um tamanho máximo de T que uma dada árvore pode ter de profundidade e uma população de **P** indivíduos, no *RHH*, cria-se **P**-1 subgrupos, cada um correspondente à árvores de no máximo tamanhos 2 a **T**. Então para cada um desses subgrupos, metade dos indivíduos são criados usando o *GT* e a outra metade usando o *FT*.

Essa técnica foi utilizada devido sua capacidade de introduzir uma grande diversidade na população incial.

1. *Grow Tree(GT*: cada ramificação da árvore pode estar no máximo na profundidade **T**, em particular, podem ser menores
2. *Full Trees(FT)*: A árvore é sempre de tamanho **T**, em toda ramificação, inclusive.

2.2.3 Fitness

A fitness é calculada para avaliar cada indivíduo. Foi utilizada a equação presente no enunciado do trabalho, a "RMSE".

2.2.4 Seleção

A seleção é feita através de um torneio de tamanho **K**. Dentre os indivíduos de uma população, aleatoriamente se escolhe **K** e o de maior fitness é o vencedor, dado um vencedor, sobre ele poderão se realizar as operações genéticas.

2.2.5 Operações Genéticas

São elas duas, crossover e mutação. Ocorrem de maneira exclusiva.

2.2.6 Nova População

Ao se realizr as operações genéticas, existem duas possibilidades, uma escolha elitista, que compara o pai com o filho, o de melhor fitness segue para a geração seguinte no caso da mutação e os dois melhores, dentre filhos e pais, no caso do crossover.

2.2.7 Critério de Parada

O critério de parada adotado foi o de número de gerações. Desse modo temos um diagrama completo de tudo que foi acontecendo ao longo do tempo.

2.3 Experimentos

Nos experimentos, foram observadas algumas estatísticas, em particular as presentes na especificação do trabalho. Sendo elas:

1. Média do melhor indivíduo.
2. Média do pior indivíduo.
3. Média do número de filhos com fitness melhor do que a média das fitness dos pais.
4. Média do número de indivíduos repetidos, no caso, comparamos fitness(avaliação do fenótipo) iguais.

Dessas estatísticas, nas tomadas de decisões para quais fatores mais afetam o desempenho, foi colocado um peso maior sobre as de melhores indivíduos e piores indivíduos.

O teste padrão utilizado foi o de:

Tamanho da População: 50

Tamanho das ávores: 7

Gerações: 50

Chance de Crossover/Mutação: 0,6/0,3

Elitismo: Sim

Tamanho do torneio: 2

Os demais testes, são uma variação em cima disso, especificando as mudanças na legenda.

2.3.1 Determinado os parâmetros ideias

Foram feitos 20 testes com *seed* diferentes para cada alteração, alterando todos os possíveis parâmetros, menos o tamanho das árvores, com o objetivo de ver qual em média teria os melhores indivíduos com consistência.

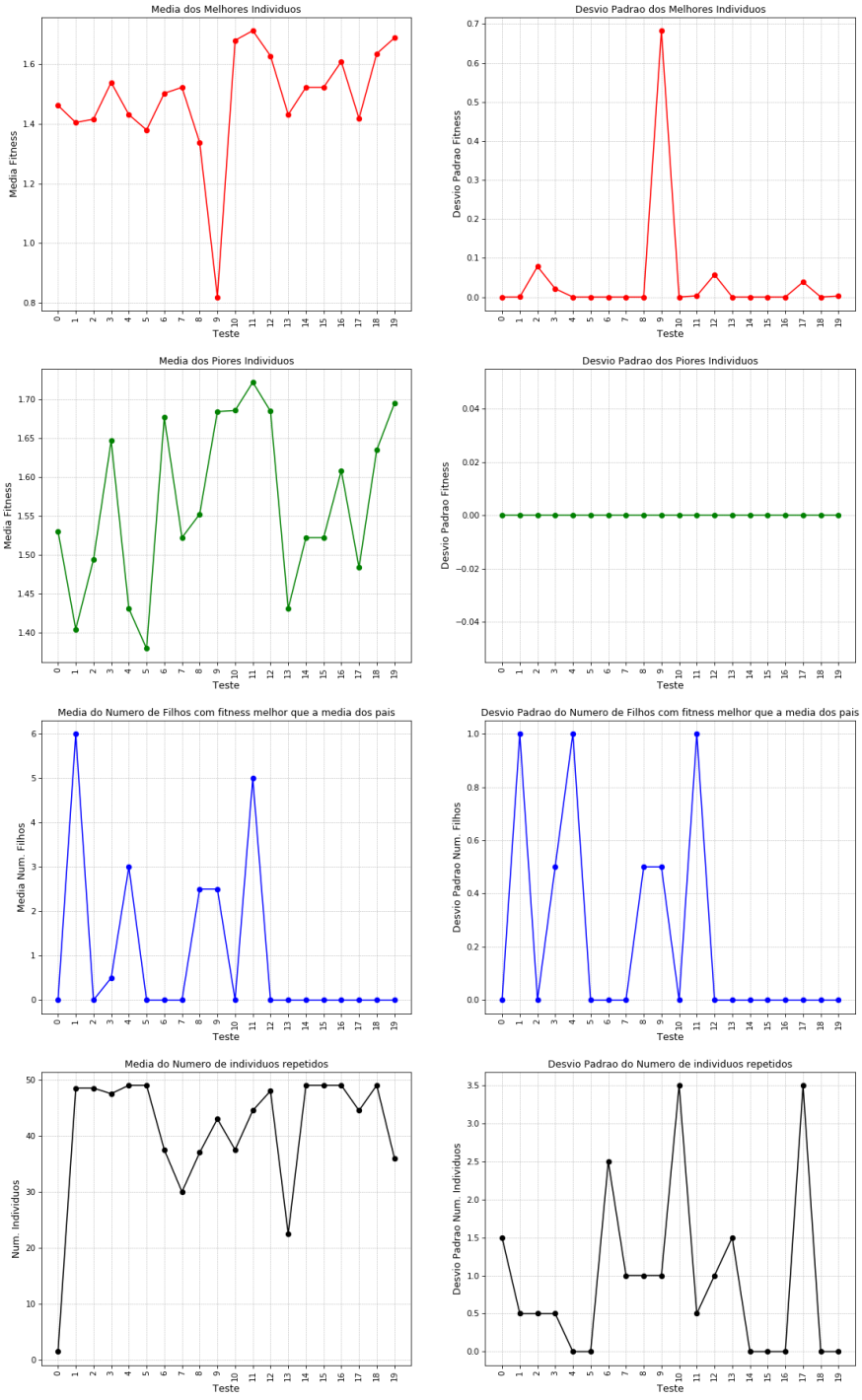

Figura 3: Usando o padrão

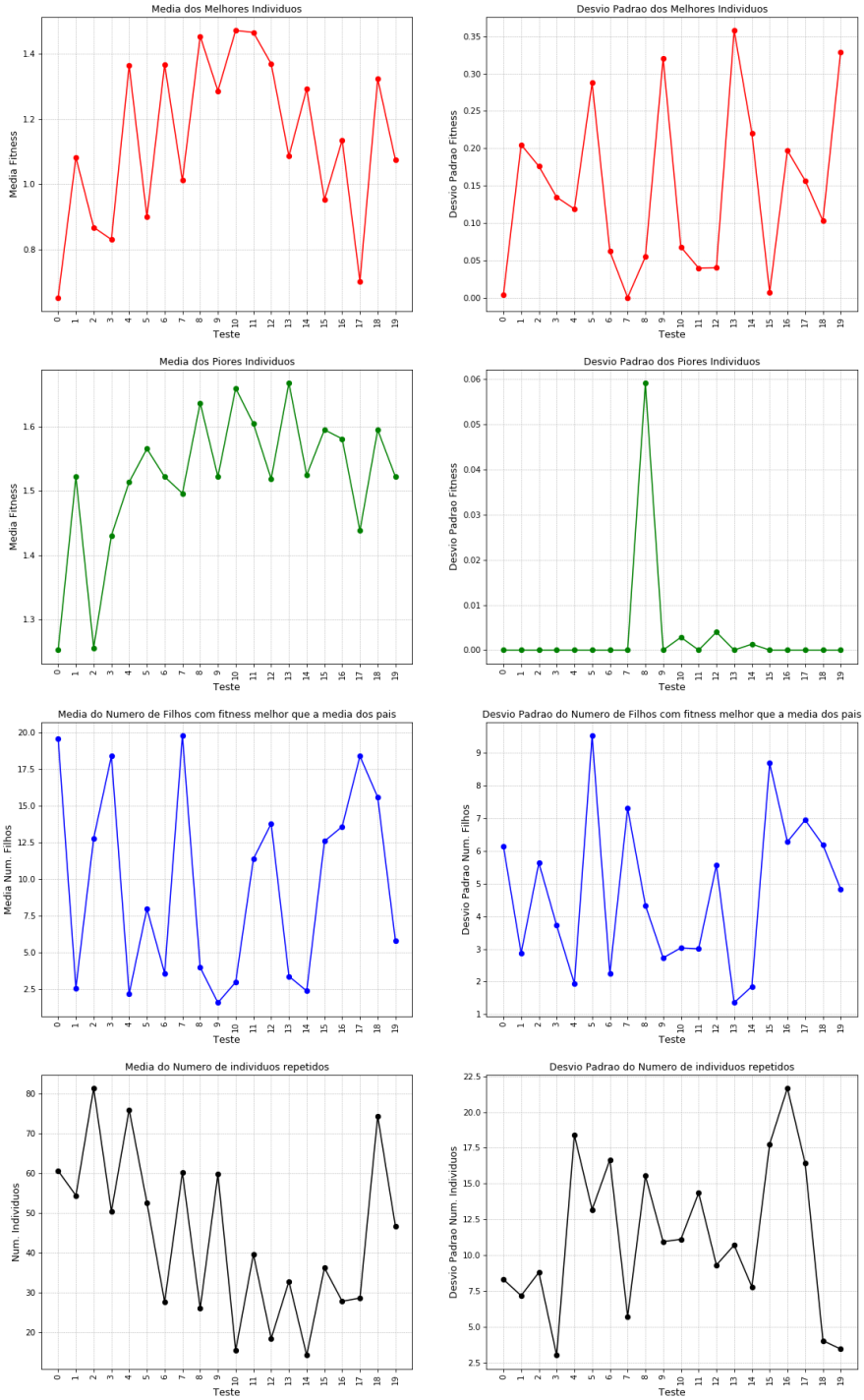

Figura 4: Variando a chance de Crossover/mutação para 0.9/0,1

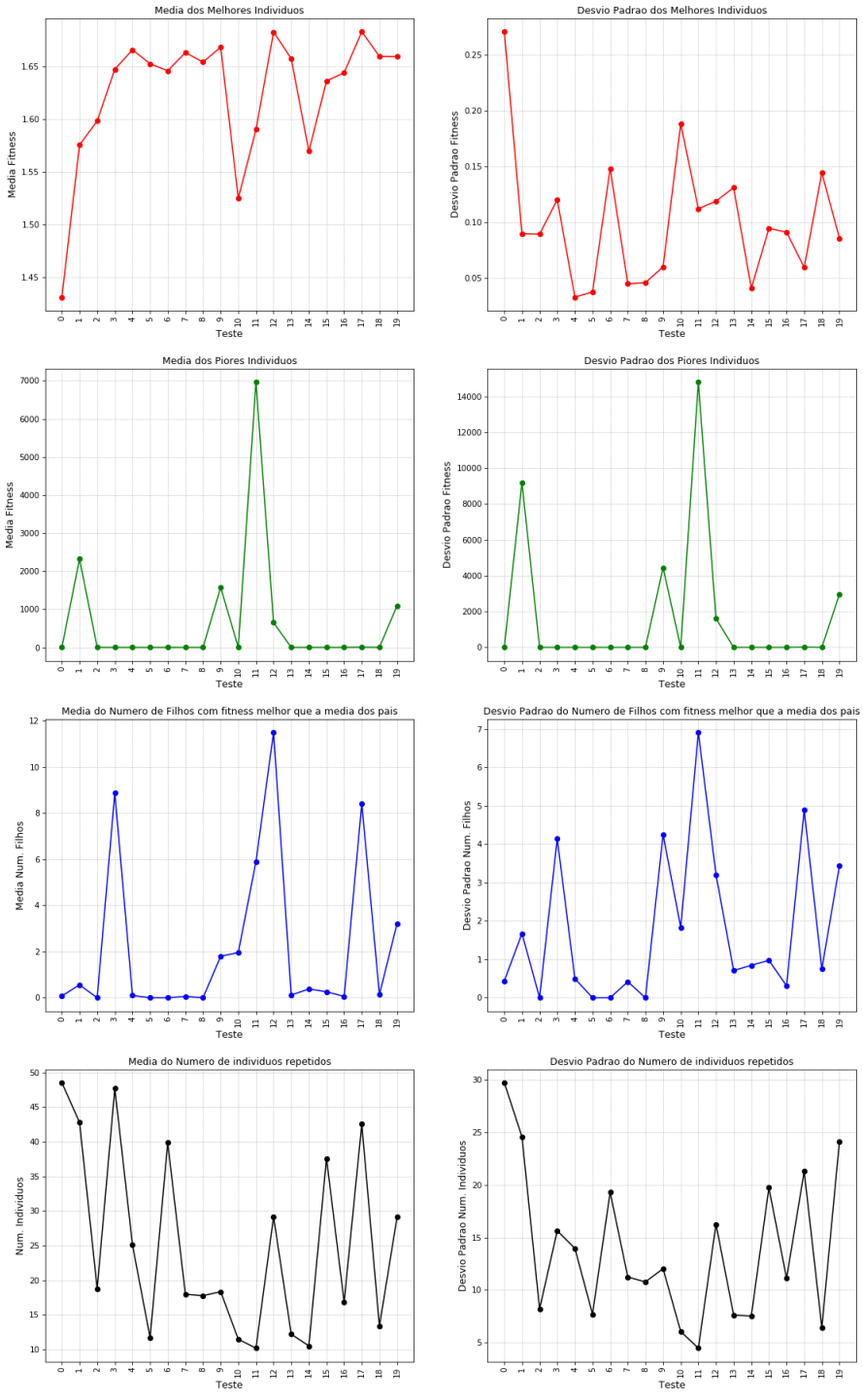

Figura 5: Variando a população para 100

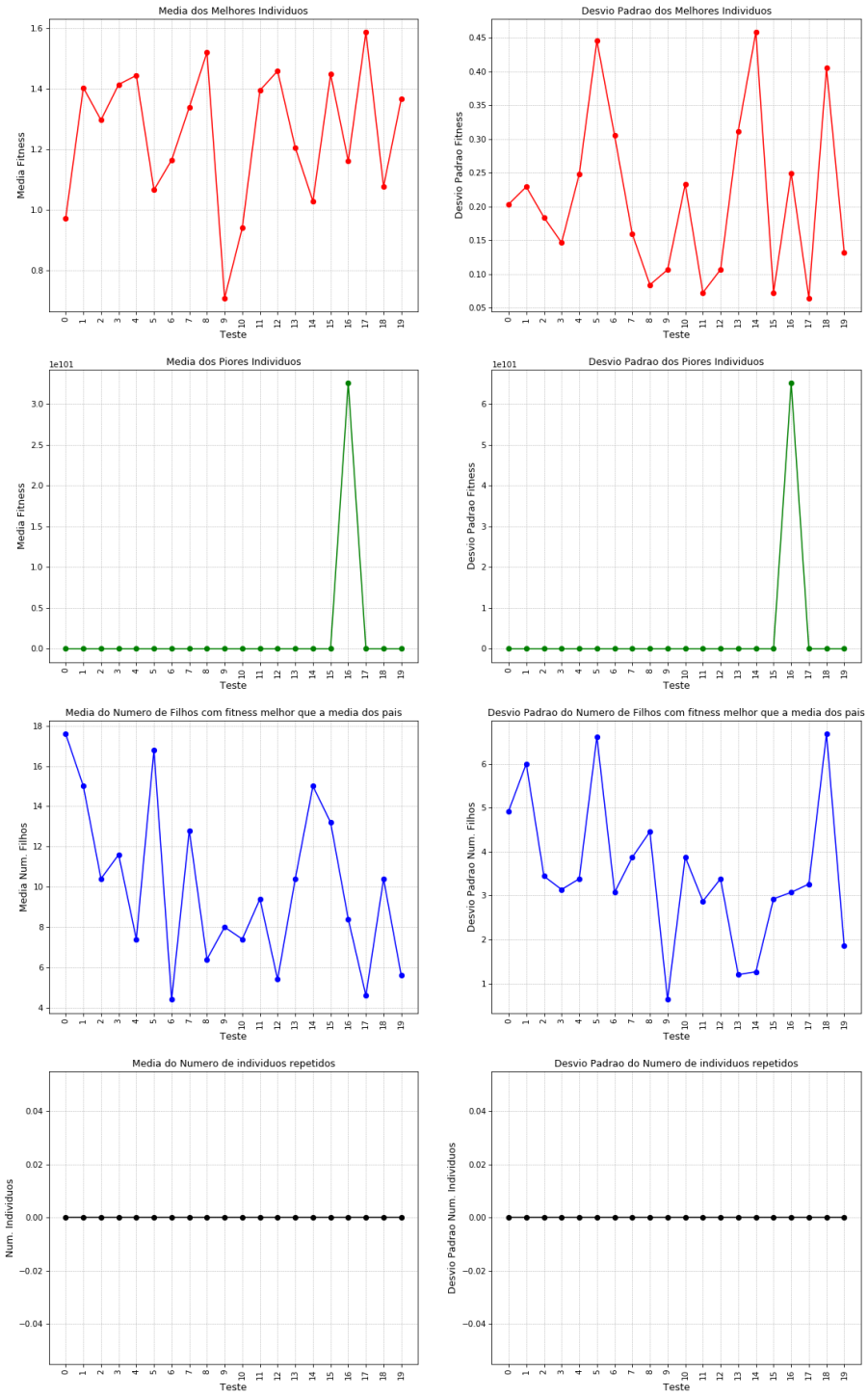

Figura 6: Variando para não ter elitismo

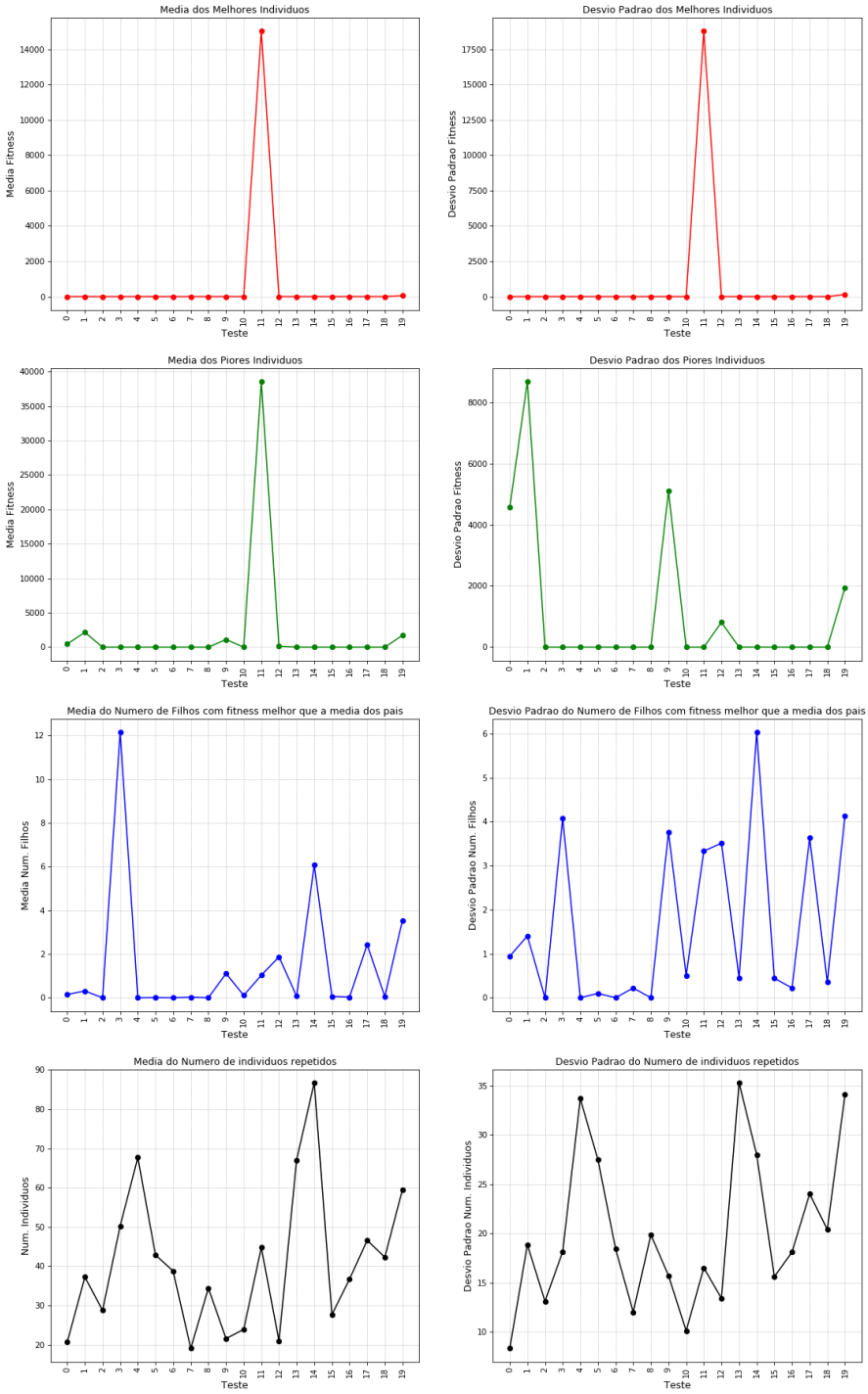

Figura 7: Variando as gerações para 100

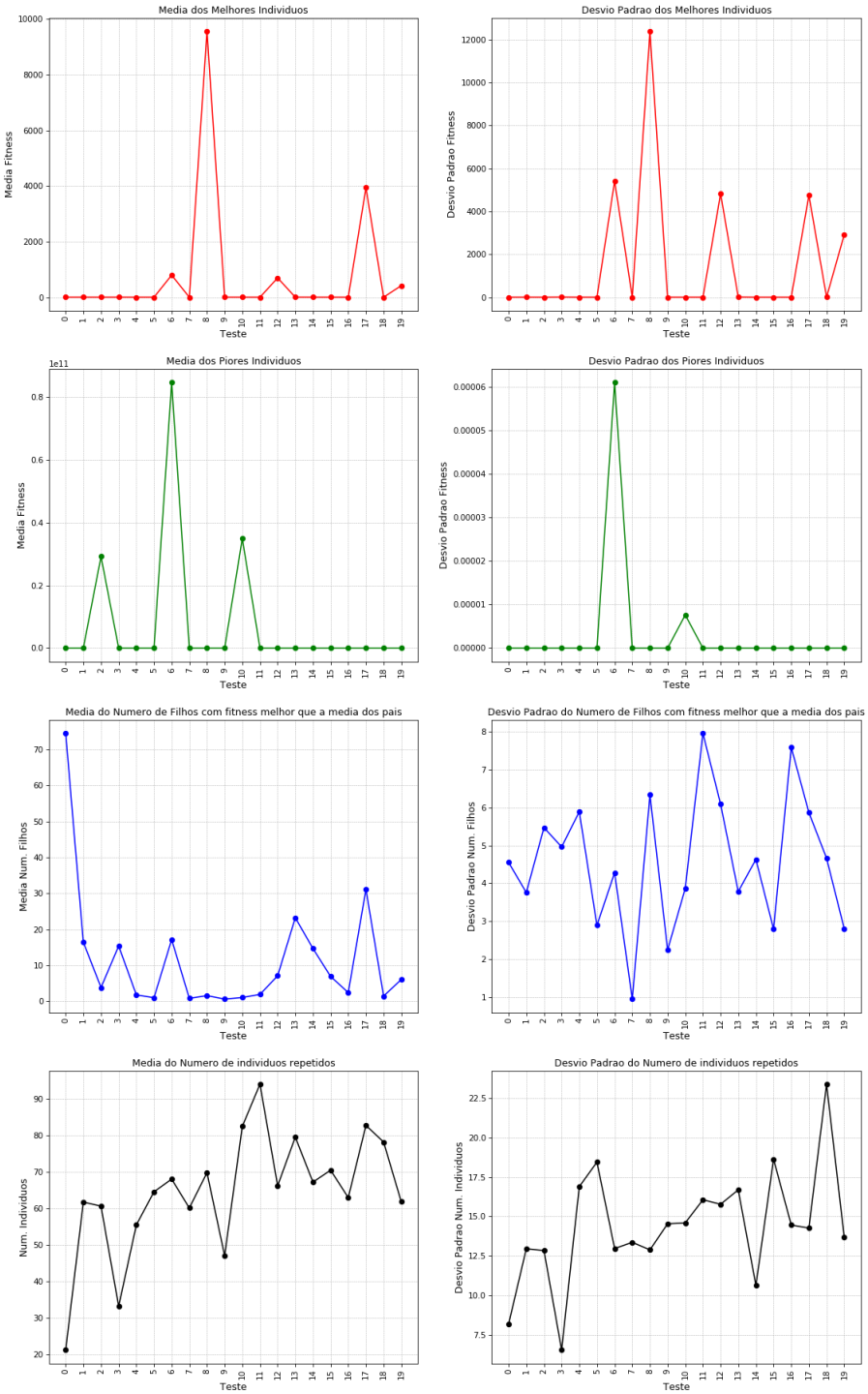

Figura 8: Variando o tamanho do torneio para 5

Dado essas informações, olhamos para principalmente os melhores e piores indivíduos, podemos observar que dado isso, os parâmetros que mostraram melhorias foram os de Crossover/mutação e o de número de indivíduos em dada população.

Nos casos do aumento do torneio, ocorre uma seleção maior dos mesmos indivíduos, acar-

retando numa diminuição da diversidade, que pode ocasionar em individuos piores através das gerações.

Quando se remove elitismo, é mais comum se perder os melhores indivíduos pelas gerações, tanto que se observa que na média, o melhor de cada geração se torna pior e o pior de cada geração, na média, também.

No caso da maior chance de crossover, temos uma convergência, na média, para indivíduos melhores, enquanto aumentar a mutação leva a indivíduos mais diversificados, não necessariamente melhores, mas aumenta a área de busca. Assim vemos uma confirmação da teoria, na qual crossover faz *exploitation* e mutação *exploration*.

Aumentar a população inicial, nos leva a uma diversidade maior, que acarreta numa *exploration* inicial melhor.

Aumentar o número de gerações, acarreta numa maior convergência para o mesmo resultado, entretanto, a partir de certo ponto, as fitness médias continuam próximas(mais observável nas próximas análises abaixo). Feita essa análise preliminar, parte-se para a análise em cima da base de dados concreta, dessa vez, olhando o desenvolvimento através das gerações.

2.3.2 Experimentos Concretos

Os experimentos realizados, comparam os três melhores testes(seeds) dentre um conjunto testado. O fator "melhor"deles vem de possuírem uma média melhor do melhor indivíduo.

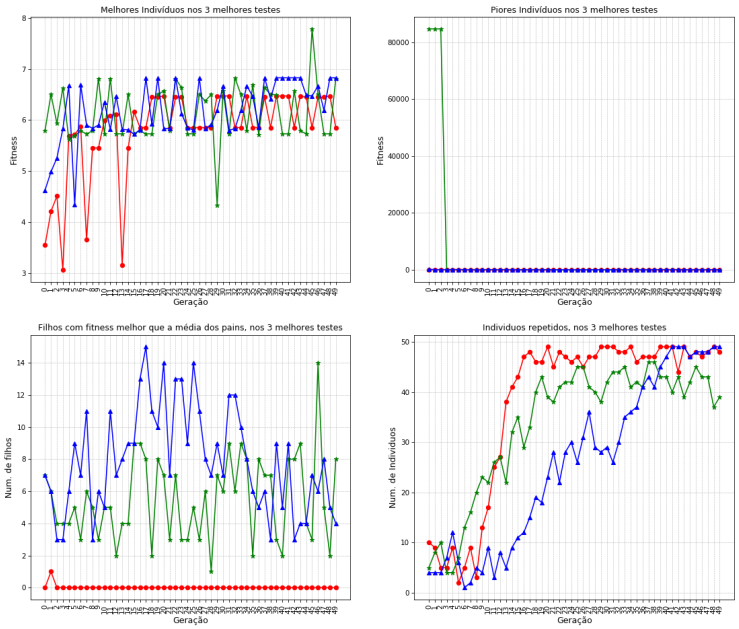

Figura 9: Usando os dados default

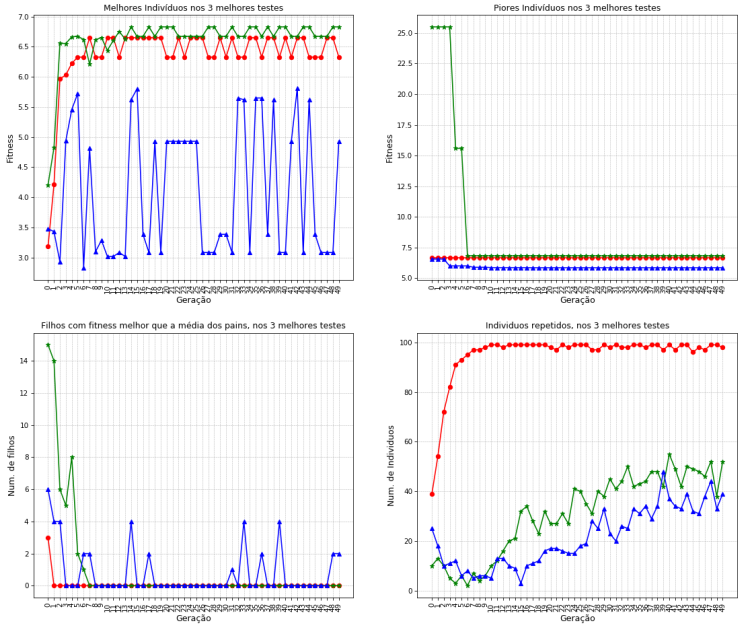

Figura 10: Variando a população para 100

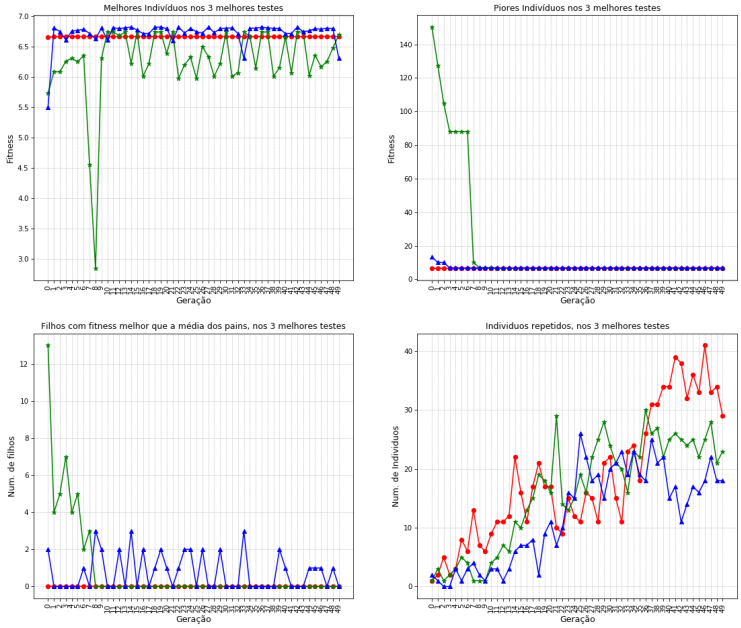

Figura 11: Variando o crossover/mutação para 0,9/0,1

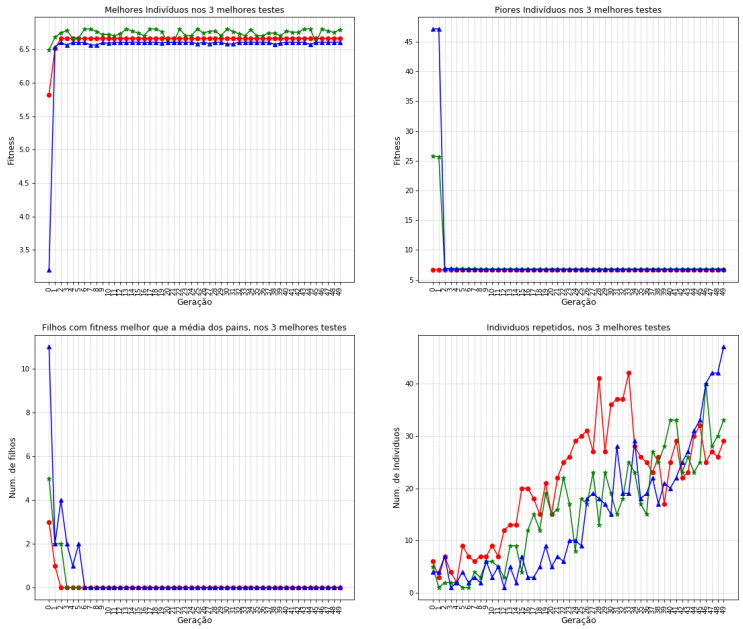

Figura 12: Variando o crossover/mutação para 0,9/0,1 e a população para 100

Alguns fatores interessantes, são o aumento na população, acarreta no aumento da exploração, porém, isso também inlfuencia numa maior chance de, através dos torneios, se perder os melhores indivíduos a cada geração. Já uma população pequena, pode simplesmente se manter em um espaço de busca muito limitado.

Outro fator de interesse, é a rápida convergência ao colocar o crossover em 0,9, vemos que tanto o melhor como o pior indivíduo ficam muito próximos, muito rápido, isso acarreta uma pressão seletiva alta. Observamos que isso também gera impacto no número de repetidos crescer e o número de filhos melhores que a média dos pais cair.

É notável a diferença da média do melhor indivíduo ao se comparar essa base testada, com a anterior. Ness base, ocorre um *overfiting*, visto que existem bastante operadores para as funções. Provavelmente a função representada por essa base não se parece com as anteriores e ou é mais simples(usa menos operadores) ou usa operadores não presentes na implementação, por exemplo, seno e coseno. Outro fator que afeta aqui, é o tamanho da árvore, que limita até em qual estágio de uma operação mais complexa, se pode quebrar em operações mais simples, por exemplo, fazer um exponencial usando várias operações de multiplicação é algo limitado pelo tamanho da árvore.

Podemos observar o comportamento de convergência mais rápido nos casos em que as chances de crossover são maiores, confirmando novamente que crossover faz *exploitation* e mutação faz *exploration*. Assim como podemos observar o aumento nos indivíduos repetidos.

3 Conclusão

Executando os testes, foi-se concluindo a teoria vista nas aulas, o modelo implementado apresenta diversos fatores que refletem o esperado, operadores elitistas levam a uma convergência em cima dos melhores indivíduos, sua ausência leva a uma perda eventualmente de alguns dos melhores resultados durante dado momento, por exemplo.

A escolha feita de terminais e operações também acarreta diretamente no escopo de possíveis funções. Aumentar demais o número de operadores, pode levar a *overfiting*, diminuir demais, pode levar a uma cobertura muito pequena de possíveis funções. Já no caso de terminais, permitir números muito grandes, ou pequenos, nas constantes, pode desviar e influenciar mais do que as próprias variáveis o resultado.

A representação em árvores de expressões mostrou seu valor na eficácia da programação genética para resolver o problema de regressão simbólica, pois possibilitou uma exploração de possibilidades e auxiliou na avaliação da fitness.

No mais, a importância de se testar de maneira progressiva, fazendo pequenos ajustes foi importante, foram necessárias várias execuções para se constatar e observar os devidos comportamentos dos operadores. Assim como as diferentes *seeds* se mostraram com um valor impactante, principalmente nas primeiras gerações, antes da convergência mais geral.

Referências

- Slides aulas de Computação Natural disponíveis no moodle.
- Artigo que diz respeito à geração da população inicial:
  https://www.win.tue.nl/ipa/archive/falldays2007/HandoutEggermont.pdf