Trabalho sobre Meta Heurísticas

Rhuan Garcia de Assis Teixeira
Universidade Federal do Espírito Santo, Goiabeiras

Abstract

Este trabalho apresenta a implementação de uma rede neural de arquitetura 7x4x1 com pesos otimizados por meio da meta-heurística do Algoritmo Genético. Foram realizadas diversas experimentações para ajustar os hiperparâmetros e diferentes funções porém os resultados finais não alcançaram a performance desejada.

1. Introdução

Neste documento iremos descrever a aplicação de um classificador de rede neural, com valores de peso otimizados pela meta heurística de Algoritmo Genético, destrinchando detalhes da implementação feita, analisando os resultados e possíveis fatores que influenciaram esse resultado.

2. Descrição do Classificador

Foi implementada uma rede neural básica com a arquitetura especificada anteriormente pelo professor: 7 entradas, 4 neurônios em uma camada intermediária e 1 neurônio de saída, sem diferenciação de bias, apenas com os pesos das arestas.

Para a implementação, optei por criar uma classe Neurônio, onde são armazenados os pesos das arestas de entrada, o bias (não diferenciado nesta aplicação) e a função de ativação (sigmoid ou ReLU). A partir desta classe, criei uma classe Camada de Neurônios e, subsequentemente, a classe Rede Neural, organizadas como vetores umas das outras, mas com funções de construção

e forward (KeySelector na classe de rede neural) já definidas para facilitar o entendimento. A rede é um vetor de camadas, e a camada é um vetor de neurônios.

Essa implementação foi escolhida em detrimento de outras, como a construção de um grande vetor armazenando apenas os pesos, para facilitar o entendimento do código, o debugging, e proporcionar maior flexibilidade para testar outras arquiteturas ou utilizar novas funções, caso fosse necessário.

3. Descrição da Meta Heurística

A meta-heurística requisitada foi o algoritmo genético, e a implementação final utiliza seleção por torneio com 5 candidatos iniciais por seleção, possui elitismo, mutação (implementada substituindo um gene aleatório por um valor também aleatório) e utiliza um crossover multipoint em que todos os genes pares são trocados, resultando sempre em genes de origem alternada. Além disso, devido aos resultados obtidos, estou também fazendo uso da inserção de 1 indivíduo completamente aleatório em cada geração, a fim de obter maior variabilidade e eventualmente gerar bons genes dessa forma.

Quanto aos valores de hiperparâmetros escolhidos, em minha implementação estou experimentando com algo um pouco fora do comum, a fim de obter resultados mais interessantes. Os valores de crossover, mutação e elitismo mudam durante a execução do algoritmo. Inicialmente, a taxa de crossover é de 80%, a de mutação é de 10% e a de elitismo é de 1% (1 indivíduo). Esses valores são alterados após a execução de 1/4 da quantidade de gerações esperada, para 70%, 5% e 2%, respectivamente. Isso é feito com o objetivo de intensificar levemente algo similar a uma fase de diversificação, com maior mutação e crossover para aumentar a diversidade na população..

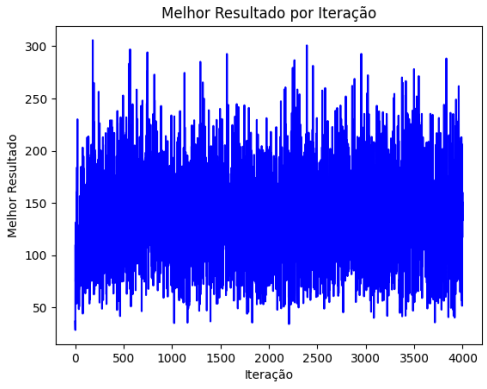
Vale notar também que foram experimentadas outras formas de seleção (rank, roleta e seleção estocástica), formas de gerar a população inicial, diversos valores para os hiperparâmetros e diversas faixas de valor para os genes. Os presentes na implementação final foram escolhidos entre todos..

4. Resultados

Após o treinamento utilizando 4000 iterações, o que levou aproximadamente 7 horas, obtive o resultado abaixo do melhor indivíduo da população resultante da última iteração:

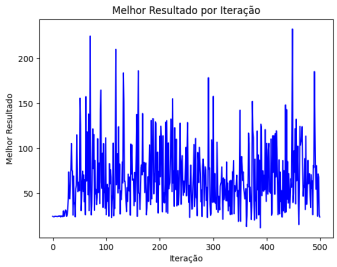
Melhor Indivíduo: 24, Media: 118.375, Desvio Padrão: 62.23837742904293 [114.25 212.75 226.5 154.25 82.25 140. 69.75 137.25 102.25 81.75 111.25 182.75 125.25 223.25 51.25 55. 155.5 129. 53.5 56.25 89.75 38. 50.75 265. 64.25 154. 197.75 137. 49.5 41.25]

Essa pontuação foi obtida com o indivíduo referente a seguinte combinação de pesos: [-0.9500608442340153, 0.4931683405457319, -0.75777005883232, -0.35921541636814625, 0.09433699628850703, 0.4624090418869009, 0.0911949061605899, -0.05401158441778647, -0.5071338915964199, -0.32123385007872196, 0.05541717671772628, -0.080328556963283, 0.03024246313963963, 0.8618540969774773, 0.18266624878015914, 0.4746317498826014, -0.8429306315137941, 0.7990614686830482, 0.03738362454311067, 0.12420632402390641, 0.884492092952186, -0.4195781488376713, -0.8281885255104691, -0.9893336285939414, -0.8662317191319553, 0.0855888872310327, 0.3631501331014946, 0.9803560834956875, 0.4569463267213576, -0.629927353701208, -0.8182260834083475, 0.2778529086927348]

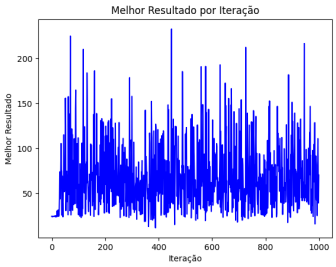


Após várias execuções de treinamento por 12 horas sem limite de iterações, constatei que o programa converge em aproximadamente 300 iterações. O gráfico

abaixo, gerado a partir de 4000 iterações, plota a cada iteração o maior score (média - desvio padrão obtidos após o mesmo indivíduo jogar 5 vezes).



(a) Evolução com 500 iterações



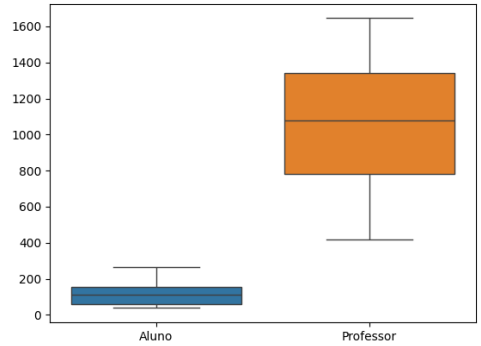
(b) Evolução com 1000 iterações

O gráfico resultante torna-se poluído e pouco útil para análise. Portanto, decidi treinar novamente com um número menor de iterações, visando uma análise mais clara. Os gráficos de 500 e 1000 iterações, que permitem uma melhor visualização da evolução da população, estão apresentados abaixo.

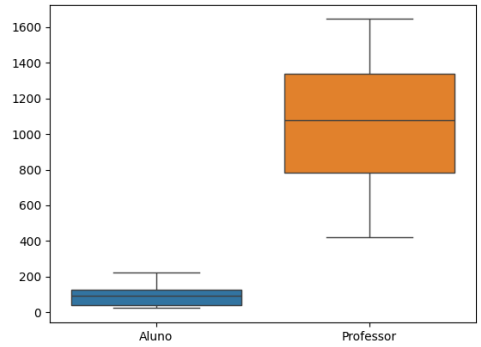
Além desses gráficos que tomei liberdade de fazer, para ver de fato alguma evolução quando comparado a um aleatório, também foi feita a comparação pedida, com o resultado profresult, presente na especificação do trabalho. Abaixo temos a tabela em que estão presentes os (p-values) do teste t pareado com amostras independentes no triângulo superior e do teste não paramétrico de wilcoxon no triângulo inferior.

Aluno	0.0000
0.0000	Professor

Também podemos ver a distribuição dos resultados encontrados por cada um no boxplot abaixo, em qual podemos ver que, os resultados diferem bastante, os meus se posicionam na faixa de 50 a 226 enquanto os do professor tem maior valor, variando entre 420 e 1650. Com isso podemos dizer que meus valores são próximos entre si, mas bem desapontadores quando vimos a implementação simples do professor obter valores tão significativamente mais altos.



Além desses testes, também experimentei brevemente com uma rede 7x17x4x1, imaginando que com essa quantidade significativamente maior de neurônios obteria um resultado diferente, mas surpreendentemente não foi o que aconteceu, me levando a mais perguntas do que respostas. Podemos ver o resultado similar no boxplot abaixo:.



Mas, como foi um breve experimento, achei válido citar aqui mas não estou levando muito em consideração, já que imagino que isso tenha ocorrido por algum erro meu enquanto alterava a rede neural.

5. Conclusões

5.1. Análise geral dos resultados

Os resultados foram desapontadores, mas podemos levantar algumas hipóteses sobre o porquê. Uma hipótese é a diversidade insuficiente da população, apesar de esforços como a inserção de indivíduos aleatórios e maior mutação inicial, ainda assim os indivíduos apresentaram grande similaridade. Isso pode ter levado a uma convergência prematura em algum mínimo local..

Outra possível causa seria a métrica utilizada, mas foram testadas várias métricas, como a média dos resultados, o valor máximo e diferentes quantidades de jogadas por rodada. No entanto, não houve diferença significativa, exceto no aumento do tempo de execução com mais rodadas, tornando essa hipótese improvável..

5.2. Contribuições do Trabalho

O trabalho foi de grande utilidade para maior entendimento principalmente da meta heurística que meu trabalho se baseia, já que, apesar dos resultados obtidos, muitos experimentos foram realizados tentando ao máximo obter um resultado diferente.

5.3. Melhorias e trabalhos futuros

Sugiro fazer mais experimentos com outras arquiteturas para a rede neural, apesar do resultado preliminar, tem potêncial para obter resultados mais interessantes e competitivos. Além disso, no ambito da disciplina, trabalhar com um colega ajudaria a encontrar soluções criativas e a detectar erros mais rapidamente. .

References

[1] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, E. Duchesnay, Scikit-learn: Machine learning in Python, Journal of Machine Learning Research 12 (2011) 2825–2830.