Relatório do Laboratório 7: Redes Neurais

Isabelle Ferreira de Oliveira
CT-213 - Engenharia da Computação 2020
Instituto Tecnológico de Aeronáutica (ITA)
São José dos Campos, Brasil
isabelle.ferreira3000@gmail.com

***Resumo*—Esse relatório documenta a implementação de uma rede neural de duas camadas para realizar a segmentação de cores para o futebol de robôs. Para isso, foi necessário configurar essa rede neural para realizar classificação multi-classe, implementando os algoritmos de Forward Propagation (inferência) e Back Propagation (treinamento) para essa rede.**

Index Terms*—Redes neurais, segmentação de cores, *Forward Propagation*, *Back Propagation

I. Introdução

Otimização consiste em encontrar o mínimo (ou máximo) de uma função, ou seja, encontrar o conjunto de parâmetros que levem essa função ao seu mínimo (ou máximo). Também pode ser visto como encontrar a melhor solução dentre todas as soluções viáveis. Nesses problemas de otimização também é possível haver restrições acerca dos parâmetros que serão analisados.

Dentre os mais diversos tipos de algoritmos de otimização, existem os métodos baseado em estratégias evolutivas, métodos esses inspirados na evolução natural das espécies. Esses métodos seguem a ideia de: dada uma população de possíveis soluções, aplica-se uma função para medir a qualidade dessas soluções candidatas. Essa qualidade quantitativa é chamade de *fitness*. Com base no *fitness*, é possível escolher as melhores soluções e, a partir delas, gerar mutações para se criar uma nova população. Esse processo é repetido até que a convergência chegue a um resultado satisfatório de otimização.

O pseudo-código geral de algortimos utilizando estratégias evolutivas pode ser visto a seguir. Em seguida, será apresentado como esse algoritmo foi implementado no contexto do laboratório.

```
def evolution_strategy(J, m0, C0,
    hyperparams):
  mu, lambda = unwrap_hyperparams(hyperparams)
  m, C = m0, C0
  while not check_stopping_condition():
    population = multivariate_normal(m, C,
        lambda)
    population = sort_ascending(population, J)
    parents = population[0:mu]
    m = mean(parents)
  return population[0]
```

No pseudocódigo acima, *J* é a função para medir a qualidade das soluções candidatas; *m0* e *C0* são a média e a matriz de covariância iniciais da população que será gerada, respectivamente; *m* e *C* são, de forma análoga, a média e a matriz de covariância atuais da população que será gerada, respectivamente; e *mu* é o tamanho da população considerada como "melhores soluções até então".

II. Implementação do algoritmo

Na parte relativa a implementação do algoritmo utilizando uma simples estratégia evolutiva (SES, do inglês *Simple Evolution Strategy*), era necessário preencher a função *tell()* da classe *SimpleEvolutionStrategy*. Recebendo os valores de *fitness* encontrados na população anterior, essa função era a responsável por atualizar a média e a matriz de covariância utilizando os melhores avaliados na antiga população e também evoluir a própria população a cada iteração. Essa função a se completar estava no código base fornecido [1].

Além disso, era necessário comparar os desempenhos desse algoritmo SES com o algoritmo CMA-ES, já fornecido no código base, aplicando-os na otimização de quatro funções, a se saber: esfera transladada, e as fuções de Ackley, Schaffer 2 e Rastrigin 2D.

A análise de vários pontos do algoritmo descrito acima terá uma breve descrição em alto nível da sua implementação a seguir.

Primeiramente, foi necessário ordenar a população atual tendo em vista os valores de *fitness* associados a ela recebido por parâmetro. Isso foi feito utilizando a função *argsort()* da biblioteca *Numpy*, conforme sugerido pelo roteiro do labotatório.

Após isso, as *mu* melhores amostras dessa população foram separadas para ajudar no cálculo de sua matriz de covariância e, posteriormente, realizar o cálculo da nova média das melhores amostras dessa geração em questão. Essa matriz de covariância foi feita a partir da multiplicação de uma determinada matriz auxiliar pela sua transposta, sendo essa matriz auxiliar a diferença entre essas melhores amostras e a média encontrada na população anterior.

Dessa forma, tendo em posse a nova matriz de covariância e a nova média das melhores amostras da população anterior, é possível gerar uma nova população, evoluindo para a próxima geração de possíveis candidatas a solução.

Para testar o funcionamento dessa implementação, foram alterados os valores das variáveis *algorithm*

(entre "ses"e "cmaes") e *function* (entre as funções *translated_sphere*, *ackley*, *schaffer2d*, *rastrigin*) no arquivo *test_evolution_strategy.py* do código base, gerando imagens dos resultados da otimização de cada uma dessas funções usando a estratégia evolutiva escolhida.

Por fim, a fim de realizar o *benchmark* através de simulações de Monte Carlo para comparar os desempenhos dessa estratégia evolutiva simples e do CMA-ES, foram alterados novamente os valores das variáveis *algorithm* e *function*, dessa vez no arquivo *benchmark_evolution_strategy.py* do código base, gerando dessa vez os gráficos comparativos de rendimento para diferentes situações de SES e CMA-ES. Esses gráficos foram apresentados nas Figuras 9 a 16.

III. Resultados e Conclusões

A. Teste das Estratégias Evolutivas

A otimização para testar a implementação foi executada para as quatro funções evolutivas já citadas na seção anterior, cuja equação matemática se encontra no roteiro do laboratório [1]. Os resultados dessas execuções foram satisfatórios e saíram conforme o esperado, comprovando o correto funcionamento da implementação e a validade da utilização das estratégias evolutivos na otimização de funções. Esses resultados foram apresentados nas Figuras de 1 a 8.

Vale reparar que os resultados de ambas implementações foram bastante equivalentes, com exceção do caso da função Rastrigin, no qual cada algoritmo convergiu em míminos locais diferentes.

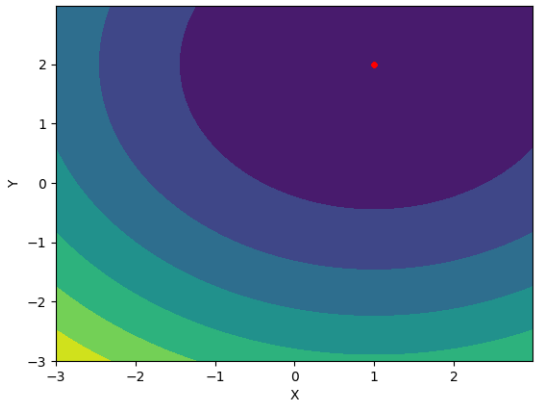

Figura 1. Otimização da função de Esfera Transladada usando a estratégia evolutiva SES. O resultado encontrado é o ponto vermelho.

B. Benchmark das Estratégias Evolutivas

A otimização para realizar o *benchmark* do desempenho de cada um dos métodos evolutivos estudados (SES e CMA-ES) para diferentes funções e parâmetros foi executada para as quatro funções evolutivas já citadas. Os resultados dessas execuções demonstraram comportamentos coerentes e foram apresentados nas Figuras de 1 a 8.

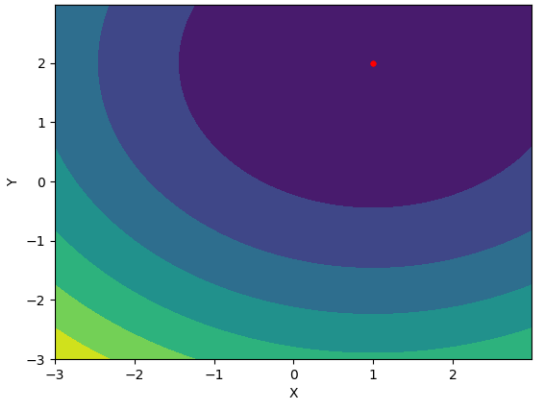

Figura 2. Otimização da função de Esfera Transladada usando a estratégia evolutiva CMA-ES. O resultado encontrado é o ponto vermelho.

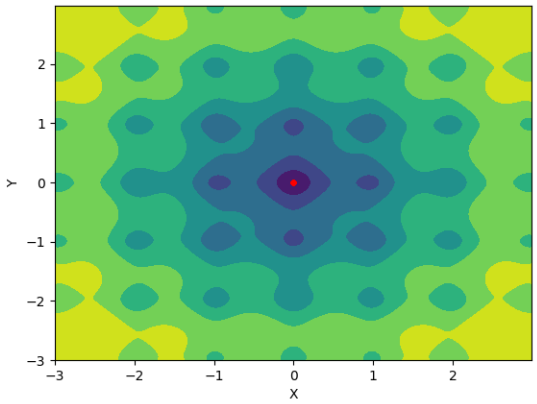

Figura 3. Otimização da função de Ackley usando a estratégia evolutiva SES. O resultado encontrado é o ponto vermelho.

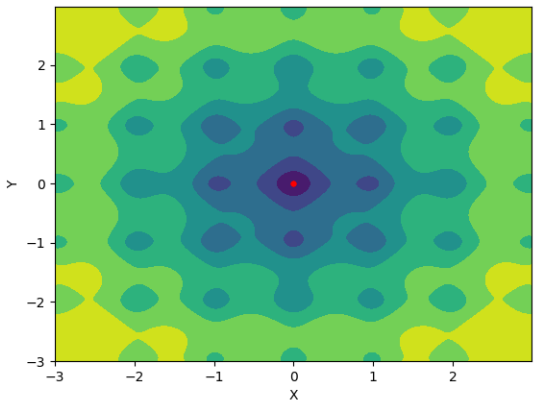

Figura 4. Otimização da função de Ackley usando a estratégia evolutiva CMA-ES. O resultado encontrado é o ponto vermelho.

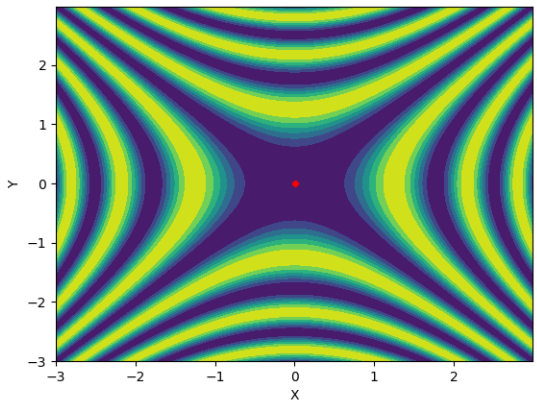

Figura 5. Otimização da função de Schaffer Nº 2 usando a estratégia evolutiva SES. O resultado encontrado é o ponto vermelho.

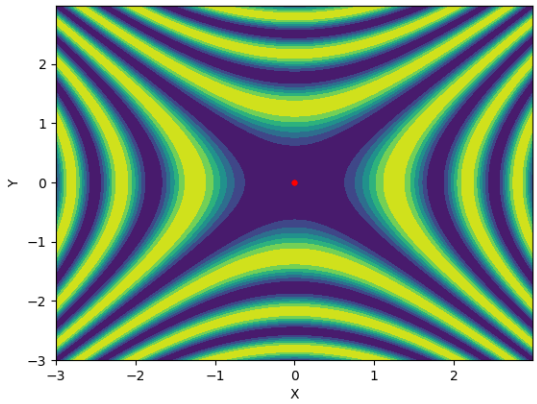

Figura 6. Otimização da função de Schaffer Nº 2 usando a estratégia evolutiva CMA-ES. O resultado encontrado é o ponto vermelho.

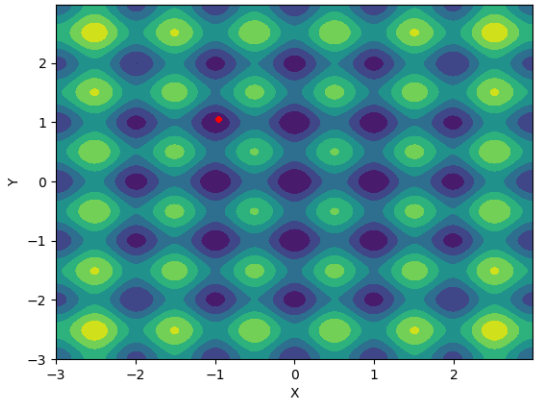

Figura 7. Otimização da função Rastrigin (2D) usando a estratégia evolutiva SES. O resultado encontrado é o ponto vermelho.

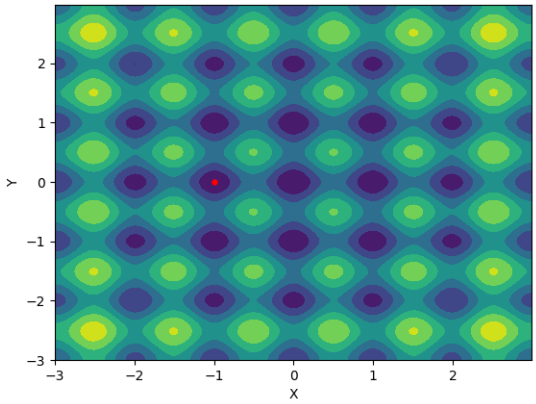

Figura 8. Otimização da função Rastrigin (2D) usando a estratégia evolutiva CMA-ES. O resultado encontrado é o ponto vermelho.

É possível salientar que, embora sempre tenha havido convergências após um número significativo de iterações, nem todas as convergências chegaram a mínimos locais, como por exemplo para a função da Esfera Transaladada, que só possui um mínimo local (que também é global), mas cada situação testada chegou a valores diferentes de *fitness*.

Além disso, notou-se que, para estratégias evolutivas mais simples, é necessário um número cada vez maior de elementos na população para que os resultados se tornem cada vez mais otimizados. Já para o algoritmo CMA-ES precisou de cerca de 1/4 de população para alcançar resultados similares aos do SES. Isso aconteceu para todos os casos com exceção à função Schaffer Nº 2, que estratégias mais simples e com menores populações se saíram melhor em desempenho do que CMA-ES.

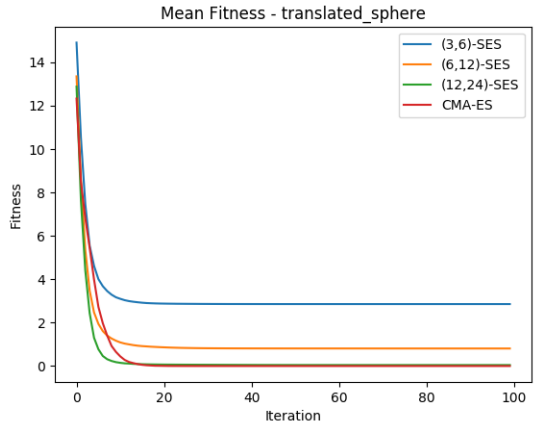

Figura 9. Evolução (e convergência) dos valores médios de fitness das populações em cada iteração para diferentes métodos evolutivos e parâmetros para a função Esfera Transladada.

Tendo em vista o que foi apresentado, pode-se notar, por fim, que esses algoritmos realmente se demonstraram eficazes

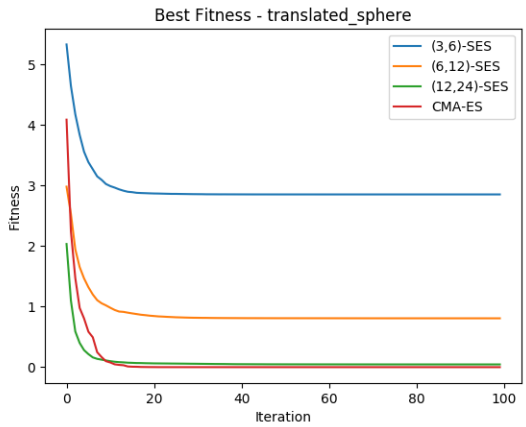

Figura 10. Evolução (e convergência) dos melhores valores de fitness das populações em cada iteração para diferentes métodos evolutivos e parâmetros para a função Esfera Transladada.

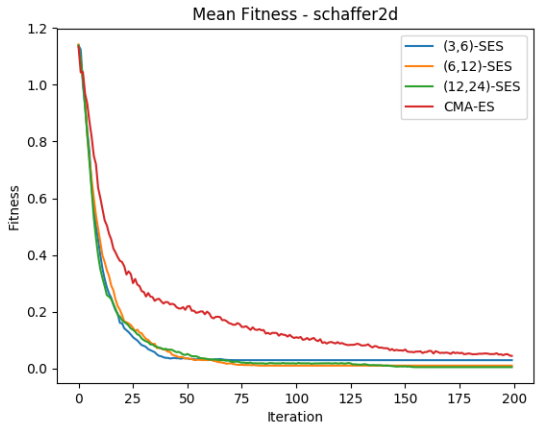

Figura 13. Evolução (e convergência) dos valores médios de fitness das populações em cada iteração para diferentes métodos evolutivos e parâmetros para a função Schaffer Nº 2.

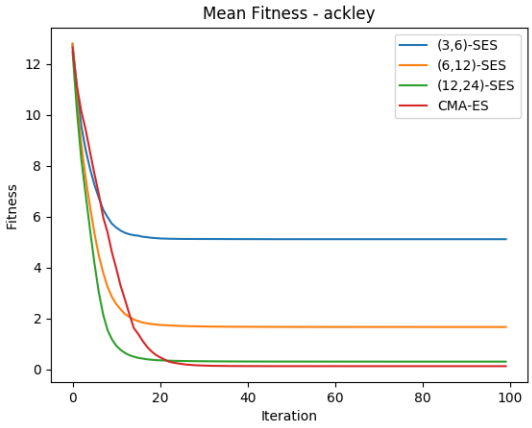

Figura 11. Evolução (e convergência) dos valores médios de fitness das populações em cada iteração para diferentes métodos evolutivos e parâmetros para a função Ackley.

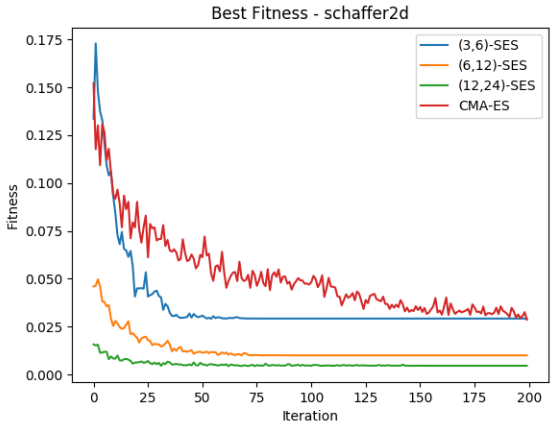

Figura 14. Evolução (e convergência) dos melhores valores de fitness das populações em cada iteração para diferentes métodos evolutivos e parâmetros para a função Schaffer Nº 2.

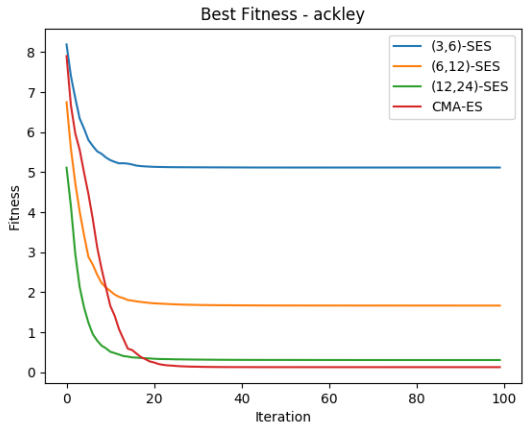

Figura 12. Evolução (e convergência) dos melhores valores de fitness das populações em cada iteração para diferentes métodos evolutivos e parâmetros para a função Ackley.

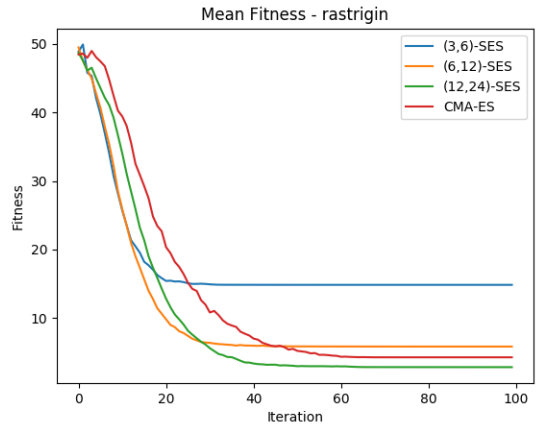

Figura 15. Evolução (e convergência) dos valores médios de fitness das populações em cada iteração para diferentes métodos evolutivos e parâmetros para a função Rastrigin (2D).

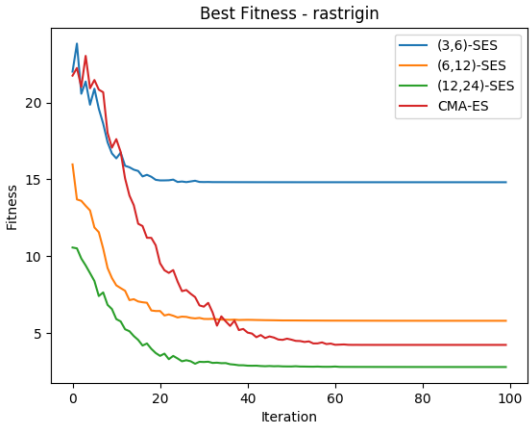

Figura 16. Evolução (e convergência) dos melhores valores de fitness das populações em cada iteração para diferentes métodos evolutivos e parâmetros para a função Rastrigin (2D).

em encontrar parâmetros otimizados para uma determinada função.

REFERÊNCIAS

[1] M. Maximo, “Roteiro: Laboratório 5 - Estratégias Evolutivas”. Instituto Tecnológico de Aeronáutica, Departamento de Computação. CT-213, 2019.