Instituto Superior de Engenharia de Coimbra

Inteligencia Computacional

2024/2025

Trabalho realizado por:

Pedro Paiva - 2021134365

Rodrigo Lourenço– 2021155662

2. Em que consiste a Computação Swarm

Computação swarm é um paradigma de inteligência artificial inspirado no comportamento coletivo de sistemas naturais, como colônias de formigas, enxames de abelhas, ou bandos de pássaros. O princípio fundamental da computação swarm é a colaboração distribuída entre agentes ou partículas que, apesar de individuais, compartilham informações para resolver problemas complexos de forma eficiente e descentralizada.

O paradigma da computação swarm apresenta algumas características fundamentais:

- **Descentralização**: As decisões são tomadas localmente por cada agente, sem a necessidade de um controle centralizado.
- **Auto-organização**: O comportamento global do sistema emerge da interação entre os agentes e com o ambiente.
- **Simplicidade**: Cada agente segue regras simples, mas o comportamento coletivo resulta em soluções eficazes para problemas complexos.

No contexto de redes neurais, a computação swarm é amplamente utilizada para otimizar a configuração de hiperparâmetros, como o número de camadas, o número de neurônios por camada, ou o coeficiente de aprendizagem. Algoritmos de otimização swarm exploram o espaço de hiperparâmetros para encontrar configurações que maximizem o desempenho do modelo sem a necessidade de uma pesquisa exaustiva.

Exemplos de Aplicações:

- Otimização de hiperparâmetros em redes neurais para melhorar a acurácia e a eficiência do modelo.
- Ajuste de parâmetros em algoritmos de machine learning em tarefas de classificação ou regressão.

3. Como funciona o GWO

O algoritmo Gray Wolf Optimization (GWO) é inspirado no comportamento social e na hierarquia de caça dos lobos cinzentos. O GWO emula a estrutura de uma matilha de lobos, onde os indivíduos são classificados hierarquicamente em alfa, beta, delta e ômega. Este algoritmo foca na simulação das técnicas de caça dos lobos.

Estrutura do Algoritmo:

1. **Hierarquia dos Lobos**:
    - **Alfa (α)**: O lobo alfa é o líder da matilha e é responsável pela toma de decisões, como a direção da caça.
    - **Beta (β)**: Lobos betas são os sub-líderes que ajudam o alfa na toma de decisões e na organização da matilha.
    - **Delta (δ)**: Lobos delta obedecem aos alfas e betas, mas lideram os lobos ômega.
    - **Ômega (ω)**: São os lobos subordinados e os últimos na hierarquia, desempenhando um papel de seguidores.
2. **Estratégias de Caça**:
    - **Cercar a presa**: Durante a otimização, os lobos ajustam suas posições em relação à melhor solução conhecida, imitando o comportamento de cercar a presa.
    - **Ataque e encurralamento**: As posições dos lobos são ajustadas para se aproximarem da presa, que corresponde ao ótimo global ou a uma solução promissora.
    - **Exploração :** O algoritmo mantém um equilíbrio entre a exploração (pesquisar em novas regiões) e a refinação de soluções conhecidas.

Funcionamento Detalhado:

- Inicializa-se a população de lobos e define-se a hierarquia com base nos melhores indivíduos (soluções).
- Durante cada iteração, os lobos alfa, beta e delta guiam a caça, ajustando suas posições com base em fórmulas que simulam o movimento em direção à presa.
- A convergência é alcançada quando os lobos se aproximam suficientemente da presa, representando uma solução ótima ou quase ótima.

Como funciona o Particle Swarm Optimization:

O PSO (Particle Swarm Optimization) é um algoritmo de otimização que simula o comportamento de enxames, como bandos de pássaros. Cada partícula representa uma solução candidata no espaço de busca move-se com base na sua própria experiência (melhor posição pessoal) e na experiência coletiva (melhor posição global).

Como Funciona:

1. Inicialização: Um grupo de partículas é distribuído aleatoriamente no espaço de busca.
2. Movimento: As partículas ajustam suas velocidades e posições com base:

   Na inércia (mantendo o movimento anterior),

   No cognitivo (atração para a sua melhor posição encontrada),

   No social (atração para a sua melhor posição global).
3. Atualização: As partículas movem-se, procuram posições melhores e atualizam informações até que um critério de paragem seja alcançado (como número máximo de iterações ou melhoria mínima).

3.1. Comparação

O PSO utiliza partículas que ajustam as suas posições no espaço de busca com base nas suas experiências passadas e nas melhores soluções da população.

O GWO imita um comportamento hierárquico mais estruturado, que pode ser mais eficaz em evitar “armadilhas” de mínimos locais.

Vantagens do GWO:

Melhor equilíbrio entre exploração e refinamento das soluções conhecidas.

Estrutura hierárquica que ajuda na diversidade da população.

Desvantagens do GWO:

Pode convergir prematuramente em problemas altamente complexos.

Dependente da inicialização da população e de parâmetros específicos para garantir um bom desempenho.

4. Função Ackley

A função Ackley é uma função matemática usada como benchmark em problemas de otimização. É uma função não linear e multimodal, caracterizada por ter muitos mínimos locais. O seu formato cria um desafio significativo para algoritmos de otimização, tornando-a ideal para avaliar o desempenho de técnicas de busca global.

4.1. Testes

Algoritmo	Teste	Parâmetros	Posição Ótima (R)	Valor Ótimo
PSO	Teste 1	População: 10, Iterações: 50, Inércia: 0.5, Cognitivo: 1.5, Social: 1.5	[1.1671e-06, -5.6238e-06]	1.6246e-05
PSO	Teste 2	População: 20, Iterações: 100, Inércia: 0.3, Cognitivo: 2.0, Social: 2.0	[-1.8590e-14, -7.6719e-15]	5.7288e-14
PSO	Teste 3	População: 30, Iterações: 150, Inércia: 0.8, Cognitivo: 2.5, Social: 2.0	[1.6875e-04, 8.9199e-05]	0.0005408
PSO	Teste 4	População: 40, Iterações: 200, Inércia: 0.4, Cognitivo: 2.0, Social: 1.5	[-9.8726e-17, -1.8540e-16]	4.4409e-16
PSO	Teste 5	População: 50, Iterações: 250, Inércia: 0.6, Cognitivo: 1.8, Social: 2.2	[-1.3121e-15, -1.0141e-15]	3.9968e-15
GWO	Teste 1	Lobos: 5, Iterações: 50	[-5.6562e-04, -4.7555e-04]	0.0021047
GWO	Teste 2	Lobos: 10, Iterações: 100	[7.1900e-14, 9.5126e-14]	3.3795e-13
GWO	Teste 3	Lobos: 15, Iterações: 150	[-1.9461e-15, -2.3076e-16]	3.9968e-15
GWO	Teste 4	Lobos: 20, Iterações: 200	[-3.9160e-16, 1.5852e-17]	4.4409e-16
GWO	Teste 5	Lobos: 30, Iterações: 250	[-3.9247e-16, -5.8041e-18]	4.4409e-16

PSO Resultados: O algoritmo PSO (Particle Swarm Optimization) mostra uma melhoria significativa no valor ótimo ao aumentar a população e as iterações. A configuração com uma inércia mais alta (Teste 3) produziu um valor maior, mas ainda assim com uma precisão razoável. Os melhores resultados foram alcançados com configurações de inércia média (Testes 2, 4 e 5) e um maior número de partículas, mostrando que PSO é eficiente em encontrar soluções precisas.

GWO Resultados: O algoritmo GWO (Grey Wolf Optimization) também apresentou uma melhoria no valor ótimo com o aumento do número de lobos e iterações. Os resultados mais precisos foram observados nos Testes 3, 4 e 5, com o valor se aproximando de zero, indicando uma solução ótima. No entanto, o valor inicial (Teste 1) mostrou um desempenho pior, o que destaca a importância de ter uma população maior e mais iterações.

Comparação Geral: Ambos os algoritmos demonstraram que um aumento na população e iterações leva a melhores resultados. No entanto, o PSO apresentou uma convergência ligeiramente mais eficiente, alcançando valores próximos de zero mais rapidamente do que o GWO. Isso sugere que, em termos de precisão e velocidade de convergência, o PSO pode ser mais vantajoso para otimizações com a função Ackley em dimensões baixas.

5. Otimização de Hiper-parâmetros

5.1 Introdução

A otimização de hiperparâmetros é um processo completamente crucial para as redes neuronais, ajustando os parâmetros externos de modo a melhorar o desempenho da rede. Contrariamente aos parâmetros internos (pesos e bias) que são ajustados durante o treino do modelo, os parâmetros externos são definidos antes do treino, modificando a arquitetura da rede. A escolha destes hiperparâmetros é fundamental, pois estes influenciam diretamente a capacidade de o modelo aprender e generalizar dados novos, o que por consequente influencia também a accuracy do modelo e a sua capacidade de responder tanto a overfitting como underfitting.

5.2 Hiper-parâmetros selecionados

Learning Rate

- **Justificação:** A learning rate é um dos hiperparâmetros mais importantes numa rede neuronal, isto porque é a responsável pela velocidade que o modelo ajusta os seus pesos durante o treino. Se for muito alta fazem o modelo divergir, se for muito baixa faz com que o treino seja demasiado lento.
- **Intervalo de Valores:** 1e-4 a 1e-2
- **Restrição:**

```
param_value = np.clip(param_value, 1e-4, 1e-2)
```

Numero de neurónios em Camadas Densas

- **Justificação:** O número de neurónios nas camadas densas impacta diretamente a capacidade de o modelo aprender padrões complexos. Se o número de neurónios for demasiado baixa pode resultar num modelo incapaz de capturar e aprender padrões complexos, e se for demasiado alto leva a overfitting.
- **Intervalo de Valores:** 32, 64, 128, 256, 512
- **Restrição:**

```
allowed_dense_units = [32, 64, 128, 256, 512]

param_value = np.random.choice(allowed_dense_units)
```

Valor de dropout

- **Justificação:** O dropout é importante para reduzir ou evitar o overfitting, desativando aleatoriamente uma percentagem de neurónios durante o treino.
- **Intervalo de Valores:** 0 a 0.5
- **Restrição:**

```
param_value = np.clip(param_value, 0, 0.5)
```

5.3 Configuração de Algoritmo

Configuração de Algoritmo

Algoritmo PSO

- **Tamanho da População:** 3 partículas
- **Número de Iterações:** 4 iterações de otimização
- **Parâmetros:**
  - **Inércia (W):** 0.5
  - **Aceleração Pessoal (C1):** 1.5
  - **Aceleração Global (C2):** 1.5
- **Função objetivo:** Minimizar a perda de validação (val_loss)
- **Condições de Paragem:** No algoritmo de PSO não foram implementadas condições de paragem, sendo que continuará até finalizar as 4 iterações.

Algoritmo GWO

- **Tamanho da população: 6** lobos
- **Número de Iterações: 6** iterações
- **Parâmetros:**
  - **Alfa:** 1.0
  - **Beta:** 1.0
  - **Delta:** 1.0
- **Função objetivo:** Minimizar a perda de validação (val_loss)
- **Condições de Paragem:** Não foram implementadas condições de paragem, sendo que continuará até realizar as **6** iterações.

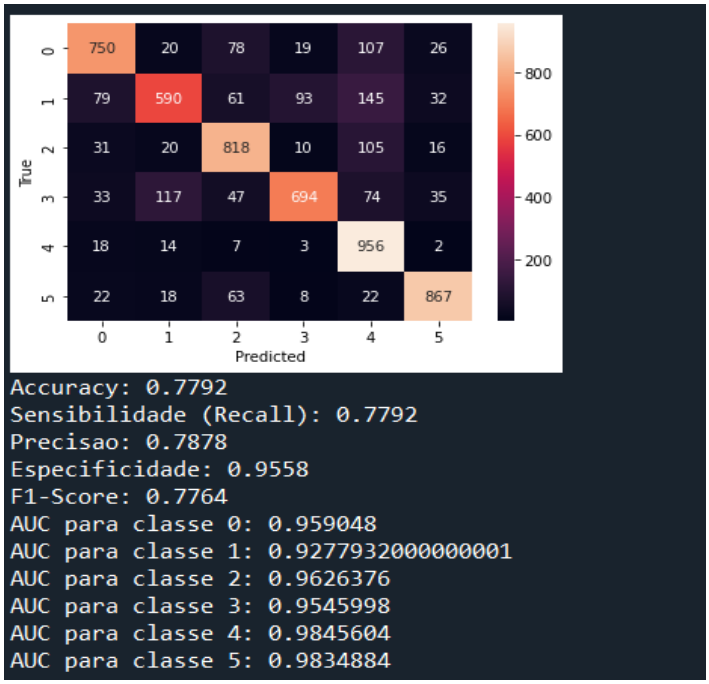
Resultados da Otimização

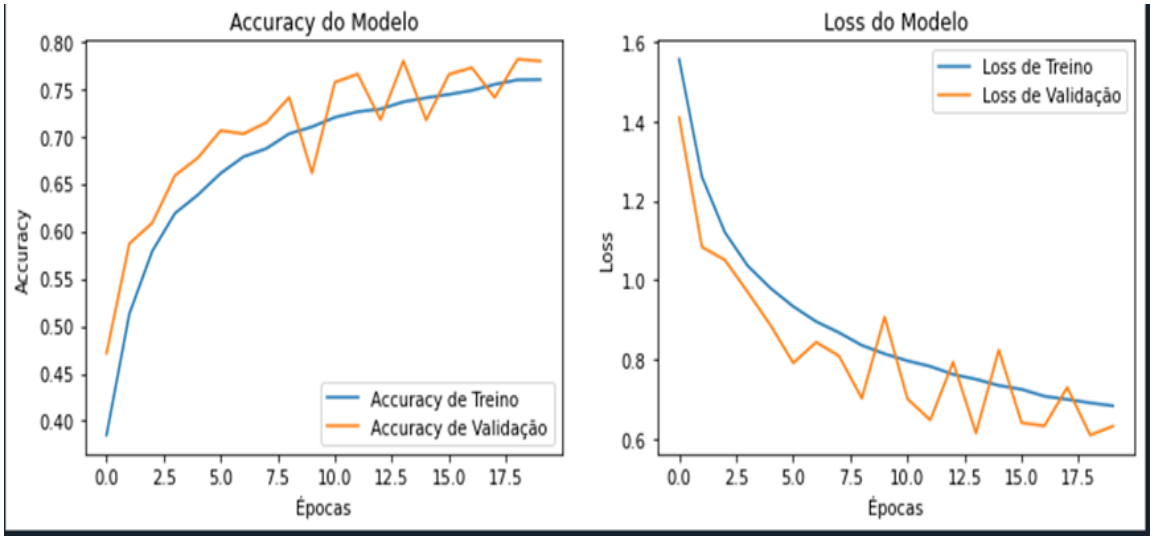
Algoritmo PSO

Obteve como melhores valores para os parâmetros:

- **Learning Rate:** 0.004714241416041165
- **Neurónios na camada densa:** 128
- **Dropout Rate:** 0.07372999925768438

Usando estes parametros para o treino do modelo, durante 20 épocas, obteve-se os seguintes resultados:





Algoritmo GWO

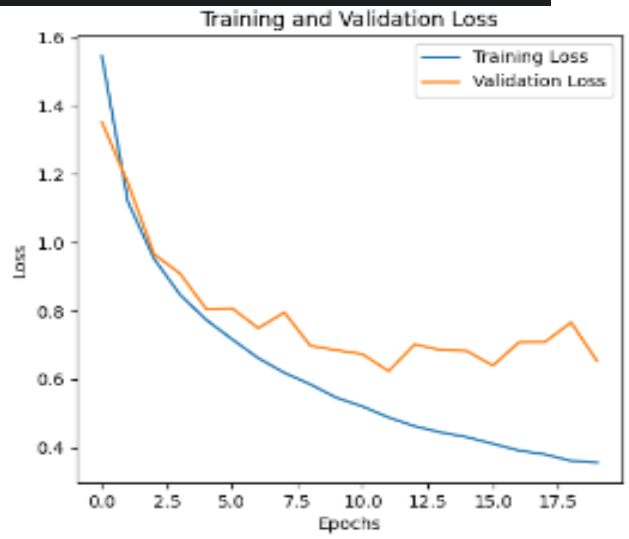
Obteve os melhores valores para os parâmetros:

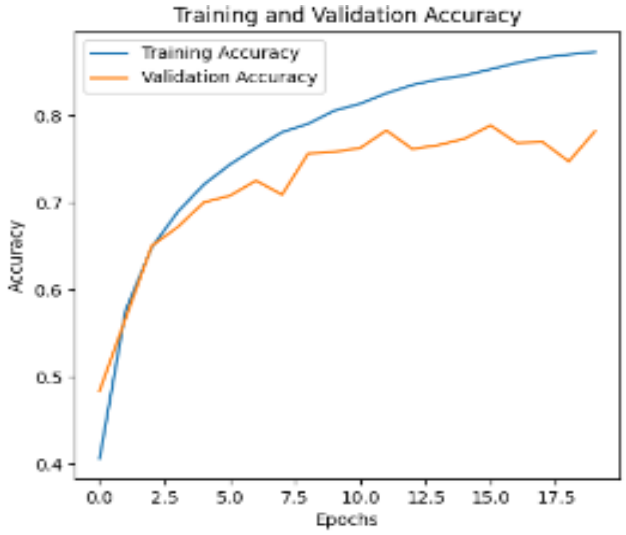
- **Learning Rate:** 0.004021107152617102
- **Neurónios na camada densa:** 512
- **Dropout Rate:** 0.06630556730897098

(Estes valores foram obtidos com os parâmetros referidos anteriormente e com 7 épocas para treino com um batch size de 128)

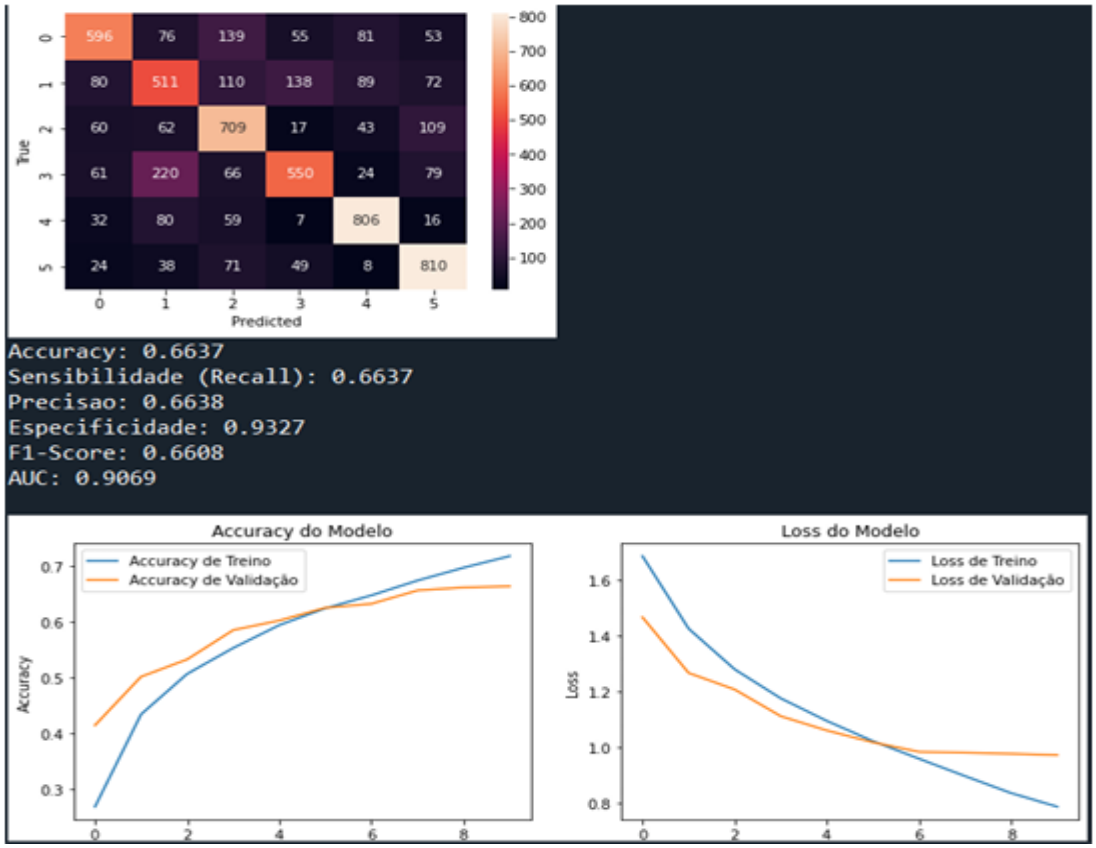
Usando estes parametros para o treino do modelo durante 20 épocas, obteve-se os seguintes resultados:

```
Accuracy: 0.7820
Sensibilidade (Recall): 0.7820
Precisao: 0.7841
Especificidade: 0.9564
F1-Score: 0.7811
AUC para classe 0: 0.9556986
AUC para classe 1: 0.9236732
AUC para classe 2: 0.9674728
AUC para classe 3: 0.9512682
AUC para classe 4: 0.9830557
AUC para classe 5: 0.9852021999999999
Melhores parâmetros encontrados pelo GWO:
```





Conclusões do capitulo



Estes resultados são referentes aos resultados do modelo na Meta 1 em apenas 10 épocas. Como podemos observar, a partir da 5ª ou 6ª época o modelo entrava em overfitting comprometendo o mesmo.

Observou-se que ambos os algoritmos apresentam vantagens específicas: o PSO mostrou convergência rápida e eficiência em cenários com dimensões menores, enquanto o GWO ofereceu um melhor equilíbrio entre exploração e refinamento das soluções. No entanto, ambos os métodos destacaram a importância de ajustar corretamente a população e o número de iterações para alcançar um desempenho ideal. Adicionalmente, a otimização de hiperparâmetros, como a learning rate, número de neurónios e dropout, foi fundamental para melhorar a capacidade do modelo de generalizar sem overfitting, demonstrando a necessidade de técnicas avançadas para a configuração eficiente de redes neurais.