Otimização de pesos de rede neural MLP na tarefa de detecção de pessoas em imagem de praia.

Leonardo A. Monte, Luciano D. S. Pacifico
Departamento de computação - DC
Universidade Federal Rural de Pernambuco - UFRPE
{leopk98@gmail.com, luciano.pacifico@ufrpe.br

Resumo: ***Monitoramento de ambientes aquáticos, por conta de riscos como afogamento e ataque de tubarão é uma tarefa que requer grande atenção e constante trabalho por equipe de resgates. Um software de sistema inteligente pode atuar diretamente no reconhecimento de banhistas em zonas de riscos e alertar uma equipe de resgate, de forma que, acidentes sejam evitados e haja uma redução no quantitativo de mão de obra necessário para a tarefa. Neste artigo é apresentado os resultados da aplicação dos algoritmos evolutivos (ALGORITMO GENÉTICO, ESTRATÉGIA EVOLUTIVA, PROGRAMAÇÃO EVOLUTIVA, EVOLUÇÃO DIFERENCIAL, OTIMIZAÇÃO POR ENXAME DE PARTÍCULAS, OTIMIZADOR DE BUSCA EM GRUPO, ALGORITMO DE BUSCA GRAVITACIONAL e OTIMIZADOR DE LOBO CINZA) na otimização de pesos de uma rede neural MLP usada para a tarefa de reconhecimento de pessoas em imagens tiradas na praia de Boa viagem, Pernambuco. O conjunto de imagem é composto por fotos de pessoas e não pessoas em diferentes tamanhos. Os resultados sugerem que com maior poder computacional e maior quantidade de tentativas os algoritmos convergiriam para melhores resultados.***

Abstract: ***Aquatic environments monitoring because the risks of drownings and shark attacks, is a task that requires big attention and constant job from rescues teams. A intelligent software system can act directly in the recognize of bathers in danger zones and generate an alert to the rescue team, thus that, accidentes can be avoided and can occur a reduction in the necessary amount of manpower to the job. In this work is presented the application results of the evolutionary algorithms (GENETIC ALGORITHM, EVOLUTIONARY STRATEGY, EVOLUTIONARY PROGRAMING, DIFFERENTIAL EVOLUTION, PARTICLE SWARM OPTIMIZATION, GROUP SEARCH OPTIMIZATION, GRAVITY SEARCH ALGORITHM and GREY WOLF OPTIMIZER) in optimization of weights of a neural network MLP used to the task of recognize of persons in images taken in Boa viagem beach, Pernambuco. The set of images is composed by person and non person photos in different sizes. The results suggests that with bigger computational power and larger quantity of tries the algorithm would converge to better results.***

I. Introdução

Com a construção do porto de Suape no estado de Pernambuco, Brasil, e consequemente a destruição dos mangues (local onde os tubarões da espécie cabeça-chata tinham os seus filhotes) os tubarões que ali residiam tiveram que buscar novos lugares para alimentação e reprodução. Outros fatores como abundância de alimentos e grandes navios cargueiros que servem como uma espécie de “guia” levaram a um alto nível de incidentes de ataques de tubarão nas praias de Pernambuco. Um dos locais mais afetado por tais problemas é a cidade do Recife, localizada em Pernambuco, estado do nordeste brasileiro, com uma população média de 4 milhões de habitantes, que tem como um forte ponto econômico o turismo. Recife representa cerca de 5,15% do número total de turistas no Brasil, possui uma costa de cerca de 20 km de águas de temperatura quente, que atrai pessoas de todo o mundo a procura dessas praias. De 1992 a 2006, foram registrados cerca de 47 ataques de tubarão, incluindo 17 fatalidades, que foram altamente noticiadas por canais de mídia brasileiros e internacionais, prejudicando economicamente o mercado de turismo pernambucano[1]. Correntes marítimas são fatores de riscos para banhistas desavisados que não conhecem as mediações da praia e tendem a ser arrastados para zonas de risco de morte por afogamento. Existem nas imediações das praias salva vidas que fazem a vigia e fiscalização da área, de modo que seja evitado o

maior número possível de acidentes. Porém, tal tarefa não é de fácil realização devido ao grande número de banhistas e a grande extensão das áreas restritas de banho. Avisos em tais zonas são constantes e numerosos, porém, nem sempre são relevados pela população gerando assim acidentes e uma constante necessidade de monitoramento. Um algoritmo eficiente de classificação de banhistas e não banhistas, é um primeiro passo para um sistema de monitoramento de câmeras em tempo real e eficiente utilizando inteligência artificial, garantindo uma melhor fiscalização e uma diminuição nos riscos de acidentes e ataques de tubarões na costa pernambucana. Tal problema encontra barreiras na detectação devido ao clima e ambiente da praia, como por exemplo: Variação na luminosidade dependendo do horário do dia, posição do sol referente ao movimento de rotação da terra, banhistas parcialmente submersos na água e a posição da câmera. Pensando nisso, este artigo apresenta a utilização de um conjunto de algoritmos evolutivos (ALGORITMO GENÉTICO, ESTRATÉGIA EVOLUTIVA, PROGRAMAÇÃO EVOLUTIVA e EVOLUÇÃO DIFERENCIAL) na tarefa de otimização de pesos de uma rede neural MLP, que é usada na tarefa de classificação de pessoas em imagens tiradas na praia de Boa viagem, Recife, Pernambuco, de banhistas e não banhistas.

Foram utilizados dois descritores de características na construção deste artigo, o descritor Hu moments e o HOG de forma conjunta. A técnica de PCA (Análise de componentes principais) foi utilizado para uma redução na quantidade de dados e melhoria no desempenho e velocidade de execução dos algoritmos

II. BACKGROUND

Esta seção discute os métodos de extração de características usados e os algoritmos evolutivos.

A. Extração de carácteristicas

1) HOG(Histograma de Gradientes Orientados): A ideia principal por trás do descritor HOG, é que uma imagem pode ser representada por seus gradientes, mais especificamente contando a ocorrência de padrões de gradientes locais.

2) Hu moments: O conjunto Hu de momentos invariantes é um dos mais antigos e bem estabelecidos descritores de imagens. Seus valores permanecem imutáveis mesmo sob mudança de tamanho, rotação ou translação. Os sete momentos são definidos da seguinte forma:

$$\phi_1 = m_{20} + m_{02}$$
$$\phi_2 = (m_{20} - m_{02})^2 + 4m_{11}^2$$
$$\phi_3 = (m_{30} - 3m_{12})^2 + (3m_{21} - m_{03})^2$$
$$\phi_4 = (m_{30} + m_{12})^2 + (m_{21} + m_{03})^2$$
$$\phi_5 = (m_{30} - 3m_{12})(m_{30} + m_{12})\left((m_{30} + m_{12})^2 - 3(m_{12} + m_{03})^2\right) + (3m_{21} - m_{03})(m_{21} + m_{03})\left(3(m_{30} + m_{12})^2 - (m_{21} - m_{03})^2\right)$$
$$\phi_6 = (m_{20} - m_{02})\left((m_{30} + m_{12})^2 - (m_{21} + m_{03})^2\right) + 4m_{11}(m_{30} + m_{12})(m_{21} + m_{03})$$
$$\phi_7 = (3m_{21} - m_{03})(m_{30} + m_{12})\left((m_{30} + m_{12})^2 - 3(m_{12} + m_{03})^2\right) - (m_{30} - 3m_{12})(m_{21} + m_{03})\left(3(m_{30} + m_{12})^2 - (m_{21} + m_{03})^2\right)$$

Fig1. Definição dos sete momentos Hu

.

B. Algoritmos Evolutivos e rede neural MLP

Computação evolutiva é a área da inteligência artificial que engloba um conjunto de métodos computacionais inspirados na teoria da evolução das espécies. Este artigo utiliza quatro algoritmos evolutivos, os quais são apresentados a seguir:

1) Algoritmo genético: É um algoritmo baseado na aplicação dos conceitos da teoria da evolução darwiniana. Foi desenvolvido por John Holland e seus alunos no ano de 1975 e popularizado por David Goldberg em 1989. O algoritmo se inspira em cima das seguintes questões da teoria da evolução: Indivíduos com melhor adequação ao meio ambiente (melhor fitness) reproduzem mais; Ao reproduzirem mais, têm mais chances de passar seus genes para a próxima geração; Devido a utilização dos operadores genéticos, recombinação e mutação, os

cromossomos filhos não são exatamente iguais aos pais. Dessa maneira a população pode evoluir e se adaptar cada vez mais ao "meio ambiente".

```
t = 0;
initialize(P(t=0));
evaluate(P(t=0));
while isNotTerminated() do
    P_p(t) = P(t).selectParents();
    P_c(t) = reproduction(P_p);
    mutate(P_c(t));
    evaluate(P_c(t));
    P(t+1) = buildNextGenerationFrom(P_c(t), P(t));
    t = t +1;
end
```

Fig2. Pseudocódigo do algoritmo Genético

2) Estratégias Evolutivas: É uma classe de algoritmos da computação evolutiva que surgiu para solucionar problemas de otimização de parâmetros contínuos ou discretos. Foi proposto por Ingo Rechenberg e Hans-Paul schwefel, na alemanha em 1964. Baseia-se na teoria evolutiva de Lamarck, a qual defende que as variações no meio ambiente levam o indivíduo a se adaptar buscando a perfeição, seguindo duas leis: a do uso e desuso e a da transmissão das características adquiridas.

```
t:=0;
inicializa P(t);
avalia P(t);
enquanto (critério de parada não satisfeito)
P'(t):= recombinação [P(t)]
P"(t):=mutação [P'(t)]
avalia P"(t)
se (μ, λ)-EE  então
P(t+1):=seleção [P"(t)];
  senão P(t+1):=seleção [P"(t) U P(t)];
t:=t+1
fim enquanto
```

Fig3. Pseudocódigo do algoritmo Estratégia evolutiva

3) Programação Evolutiva: Foi proposto por Lawrence j. Fogel em 1996 como uma técnica de simulação da evolução para desenvolver uma forma alternativa de inteligência artificial. O comportamento inteligente era visto como requerendo as seguintes habilidades: Predizer um determinado ambiente; Responder apropriadamente a este ambiente baseado na predição feita. O ambiente foi considerado de forma genérica como sendo descrito por uma sequência de símbolos tomados a partir de um alfabeto finito. Inicialmente a programação evolutiva foi criada para trabalhar como autômatos finitos, porém em 1992 D. B Fogel estendeu para a utilização de vetores reais de atributos sujeitos a mutações com distribuição normal.

$t := 0;$
initialize $P(0) := \{x_1(0), \ldots, x_\mu(0)\} \in I^\mu$
  where $I = \mathbb{R}^n$;
evaluate $P(0)$: $\mathcal{F}(x_k(0)) = \mathcal{G}(f(x_k(0)), \nu_k)$;
while *termination criterion not fulfilled* **do**
  mutate $x'_k(t) := m(x_k(t)) \quad \forall k \in \{1, \ldots, \mu\}$;
  evaluate $P'(t) := \{x'_1(t), \ldots, x'_\mu(t)\}$;
    $(\{\mathcal{F}(x'_1(t)), \ldots, \mathcal{F}(x'_\mu(t))\})$;
  select $P(t+1) := s(P(t) \cup P'(t))$;
  $t := t + 1;$
od

Fig4. Pseudocódigo do algoritmo Programação Evolutiva

4) Evolução Diferencial: Algoritmo evolutivo primeiramente publicado em 1995. Utiliza representação dos valores como vetor de números reais. Diferentemente dos outros algoritmos evolutivos, esse utiliza três indivíduos nos seus operadores, mutação e recombinação.

```
Function x = DE(NP,CR,F,range,f)
x ⇐ random(range,NP)
fit_x ⇐ f(x)
while critério de parada não for satisfeito do
    for i = 1 até NP do
        v_{i,G+1} ⇐ mutação(x_{i,G},F)
        u_{i,G+1} ⇐ crossover(x_{i,G},v_{i,G+1},CR)
    end for
    fit_u ⇐ f(u)
    for i = 1 até NP do
        if fit_u(i) > fit_x(i) then
            x_{i,G+1} ⇐ u_{i,G+1}
        else
            x_{i,G+1} ⇐ x_{i,G}
        end if
    end for
end while
```

Fig5. Pseudocódigo do algoritmo Evolução Diferencial

5) Multilayer perceptron: É uma classe de redes neurais artificiais feedforward, que funciona na associação de vários perceptrons em mais de uma camada.

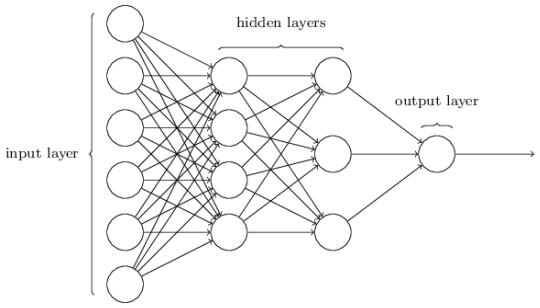

Fig6. Exemplo de uma MLP

6) Estratégia evolutiva com mutação de cauchy: Algoritmo de estratégia evolutiva utilizando uma abordagem de mutação diferente. Distribuição de cauchy é a distribuição de probabilidades dada pela função densidade de probabilidade.[2]

7) Algoritmo genético + evolução diferencial: Junção dos algoritmo genético com o algoritmo evolução diferencial, na forma em que a população é dividida pela metade e cada algoritmo itera em uma das metades e insere seu melhor indivíduo na população do outro.

8) Otimização por enxame de partículas(PSO): Criado por James Kennedy e Russell Eberhart, encontra-se atualmente, atualmente, entre as meta-heurísticas de algoritmos de otimização, baseadas em padrões da natureza. PSO é inspirado pelo comportamento social e cooperativo exibido por várias espécies por forma a realizar as suas necessidades no espaço de pesquisa ("Search-space"). Além disso, PSO é uma meta=heurística, pois realiza pouca ou nenhumas premissas sobre o problema que está a ser otimizado e pode procurar soluções candidatas em espaços de grandes dimensões.
   Em termos gerais, o algoritmo guia-se por experiência pessoal (Pbest), experiência geral (Gbest) e o movimento das partículas atual para decidir as posições seguintes no espaço de pesquisa. O PSO resolve um problema criando uma população de soluções candidatas, também conhecidas como partículas, e movendo estas partículas em torno do espaço de pesquisa, de acordo com fórmulas matemáticas simples sobre a posição e velocidade da partícula.

9) Otimização por busca de grupo(GSO): Algoritmo que encontra-se entre as meta-heurísticas de algoritmos de otimização, que se baseia em padrões da natureza. GSO é inspirado no comportamento de busca de recursos de grupos de animais, onde o grupo pode ser definido por um conjunto de organismos que interagem entre si. GSO é baseado no comportamento Producer-Scrounger (PS), onde o Producer é o que busca a melhor solução, enquanto o Scrounger se junta a melhor solução encontrada. Outra classe representada no problema são os rangers, entidades que andam livremente pelo espaço de busca, atrás de soluções ainda não encontradas.

10) Algoritmo de Busca Gravitacional(GSA): Algoritmo de comportamento de colônia baseado nas leis da gravidade e iteração de massas. O algoritmo é baseado na

gravidade newtoniana onde : “ Toda partícula no universo atrai toda outra partícula com uma força que é diretamente proporcional ao produto das massas e inversamente proporcional ao quadrado da distância entre eles.

11) Otimizador de lobo cinza(GWO): Algoritmo de comportamento de colônia baseado no comportamento de caça dos lobos cinzentos(canis lupus). Algoritmo utiliza a técnica de caça dos lobos como modo de heurística, tendo quatro possíveis tipo de indivíduos na população: Alpha, Beta, delta e ômega onde cada um desses possui um comportamento de caça específico.

12) Otimizador de lobo cinza + Otimizador por busca de grupo (GWO+GSO): Junção dos algoritmo GWO com o algoritmo GSO, na forma em que a população é dividida pela metade e cada algoritmo itera em uma das metades e insere seu melhor indivíduo na população do outro.

13) Algoritmo de Busca Gravitacional + Otimização por enxame de particulas (GSA+PSO): Junção dos algoritmo GSA com o algoritmo PSO, na forma em que a população é dividida pela metade e cada algoritmo itera em uma das metades e insere seu melhor indivíduo na população do outro.

III. Resultados

Nesta seção serão abordados os resultados obtidos pela rede neural MLP e as especificações de cada algoritmo utilizado para construção do artigo.

Para função de fitness, foi utilizado a biblioteca Keras, onde foi construída uma rede neural MLP com 3 camadas, a camada de entrada, a intermediária formada por 6 neurônios e a camada de saída formada por 1 neurônio. Entre a camada de entrada e camada intermediária existia 50x6 pesos, onde 50 é o valor do tamanho da entrada do algoritmo e 6 a quantidade de neurônios da camada intermediária e entre a camada intermediária e a de saída 6x1 pesos. Gerando no total 306 valores para cada indivíduo da população dos algoritmos evolutivos. Foi utilizado como parâmetro de melhor fitness a maximização da acurácia.

No algoritmo genético foi utilizado os valores 0.7 e 0.03 para taxa de crossover e mutação respectivamente, foi também utilizado elitismo, escolhendo os dois melhores indivíduos e torneio para construção da nova geração da população. O algoritmo de estratégia evolutiva implementado foi o (mi,lambda)[2]. O algoritmo de programação evolutiva foi o Meta-EP[3]. No algoritmo evolução diferencial foram utilizados os valores 0.7 e 0.8 para taxa de crossover e mutação respectivamente. Em todos os algoritmos a população consistia em 30 indivíduos, sendo cada indivíduo um vetor de valores reais de tamanho 306.

Abaixo pode ser visto o resultado obtidos nos algoritmos e seus tempos aproximados.

	GA	DE	EE	EEC	PE	GA+ DE	MLP
Média	64,28	66,28	67,78	66,71	65,99	67,64	57,71
STD	0,064	0,060	0,035	0,041	0,042	0,024	0,039
Melhor ac.	74,49	72,85	72,85	74,28	75,71	70,71	63,57
Tempo médio	31,47	31,37	61,03	60,98	31,49	45,04	1,97
Mediana	63,92	67,85	68,57	66,42	65	68,57	58,21

Fig7.. Resultados dos algoritmos1

	PSO	GSO	GWO	GSA	GWO +GSO	GSA+P SO
Média	69,21	69,14	69,42	68,07	68,57	68,14
STD	0,023	0,024	0,037	0,029	0,035	0,041
Melhor ac.	72,85	73,57	75,71	73,57	74,28	74,28
Tempo médio	38,64	42,59	40,64	97,54	92,13	102,65
Mediana	69,64	69,28	68,57	67,14	68,57	69,28

Fig8.. Resultados dos algoritmos2

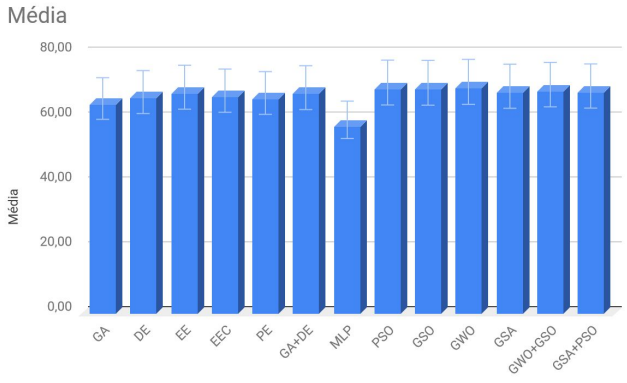

Fig9.. Resultados com o conjunto de teste

A. Análise estatística dos melhores casos:

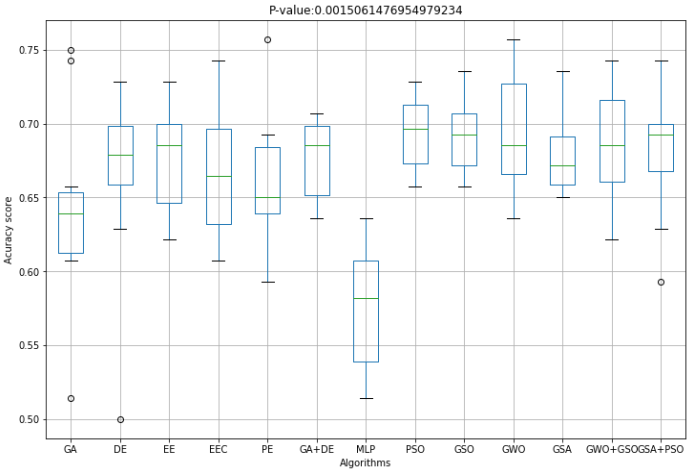

Fig8. P-value e boxplot dos algoritmos

IV. CONCLUSÕES

É possível notar a partir da análise das tabelas, que o resultado se manteve em escalas próximas em todos os algoritmos,porém, com uma vantagem nos dados dos algoritmos genéticos em relação a MLP sozinha.

Devido ao grande esforço computacional, os algoritmos não conseguiram apresentar resultados ainda melhores.

Tendo um maior número de iterações e maior diversidade populacional é esperado que se atinja resultados mais sólidos nesta tarefa.

V. Referências.

[1] F. H. V. Hazin, G. H. Burgess, and F. C. Carvalho, “A shark attack outbreak off Recife, Pernambuco, Brazil: 19922006,” Bulletin of MarineScience, vol. 82, no. 2, 2008.

[2] M. L. Nogueira, O. R. Saavedra, “Estratégias evolutivas aplicadas à otimização multimodal ” Simpósio Brasileiro de automação. 1999

[3] Ahmad Hoofar. “” IEEE TRANSACTIONS ON ANTENNAS AND PROPAGATION, VOL. 55, NO. 3, MARCH 2007.