isec
Politécnico de Coimbra

Inteligência Computacional

Docente
Inês Dominguês
Carlos Pereira

Alunos
Paulo Henrique Figueira Pestana de Gouveia - a2020121705
Nuno Alexandre Almeida Santos - a2019110035

December 14, 2022

Índice

1 Introdução

2 Em que consiste a Computação Evolucionária?

A evolução é um processo de otimização em que o objetivo é melhorar a capacidade de um organismo (ou sistema) sobreviver em ambientes dinâmicos e competitivos. Ao falar sobre evolução, é importante primeiro identificar a área em que a evolução pode ser definida, por exemplo, cósmica, química, estelar e planetária, sistemas de evolução orgânicos ou feitos pelo homem. Para essas diferentes áreas, a evolução pode ser interpretado de forma diferente.

A Computação Evolucionária compreende um conjunto de técnicas de busca e otimização inspiradas na evolução natural das espécies. Desta forma, cria-se uma população de indivíduos que vão reproduzir e competir pela sobrevivência. Os melhores sobrevivem e transferem suas características a novas gerações.

2.1 Descrição do paradigma de computação evolucionária e possíveis aplicações no contexto de treino de uma rede neuronal

Sistemas de CE resolvem problemas por meio de população, tentativa e erro, metaheurística ou otimização estocástica. Um conjunto inicial de candidatos a solução é gerado e atualizado iterativamente: remoção das soluções menos desejadas, inserção de ruído.

Em termos técnicos, as populações de solução evoluem e se adaptam à medida que estão sujeitas à seleção e mutação naturais ou artificiais. Ou seja, melhor ajuste (a função quantifica o quão adaptável/desejável é a solução). A CE é popular na IC porque leva a soluções otimizadas em uma ampla variedade de contextos, e há muitas variantes e extensões para problemas e estruturas de dados específicos.

No contexto de redes neuronais, a computação evolutiva pode ser aplicada a soluções dadas enquanto treina a rede. No exemplo de classificação de forma geométrica usando CE, a solução evolui através de seleção e mutação natural ou artificial à medida que a rede é treinada. Isso nos permite generalizar a forma com mais precisão com o auxílio de descartar soluções menos desejáveis.

3 Inteligência Swarm

Conjunto estruturado de indivíduos (ou agentes) que interagem entre si. Os Indivíduos pertencentes ao swarm (enxame) interagem para atingirem um objectivo comum, de forma mais eficiente do que agindo individualmente.

Formalmente, um enxame pode ser definido como um grupo de agentes (geralmente móveis) que se comunicam entre si (seja direta ou indiretamente), agindo no seu ambiente local. Mais formalmente, inteligência Swarm é a propriedade de um sistema pelo qual os comportamentos coletivos de agentes não sofisticados interagindo localmente com seu ambiente causam o surgimento de padrões funcionais globais coerentes. O objetivo dos modelos computacionais de inteligência Swarm é modelar o simples comportamento dos indivíduos e as interações locais com o ambiente e os indivíduos vizinhos, a fim de obter comportamentos mais complexos que podem ser usados para resolver problemas complexos, principalmente problemas de otimização.

Inteligência Swarm faz uso de algoritmos de convergência baseados em fenômenos emergentes da natureza como: colônias de insetos, estratégias coletivas de peixes e pássaros e ainda comportamento auto-organizativo de partículas atômicas e subatômicas.

Por outro lado, a otimização de colônia de formigas modela o comportamento muito simples de seguir a trilha de feromônio das formigas, onde cada formiga percebe as concentrações de feromônio em sua ambiente local e age selecionando probabilisticamente a direção com maior concentração de feromônio. Daí surge o comportamento de encontrar a melhor alternativa (caminho mais curto) a partir de uma coleção de alternativas. Modelos do comportamento local de formigas que frequentam cemitérios resultam no comportamento complexo de agrupar objetos semelhantes em clusters.

3.1 PSO

O algoritmo de otimização de enxame de partículas (PSO) é um algoritmo de pesquisa baseado em população baseado na simulação do comportamento social de pássaros dentro de um bando, que são denominados por partículas. Esse método se inicializa aleatoriamente, através de um conjunto de partículas com velocidades e posições aleatórias. Após essa inicialização os indivíduos são avaliados através da função de avaliação. Em um algoritmo PSO existe um conjunto de vetores cujas trajetórias oscilam em torno de uma região definida por cada melhor posição individual (PBEST) e a melhor posição dos outros (GBEST).

A posição da partícula, xi, vai sendo atualizada de acordo com a equação:

$$x_i(t+1) = x_i(t) + w.v_i(t) + C1.rnd(PBEST - x_i(t)) + C2.rnd(GBEST - x_i(t)) \quad (1)$$

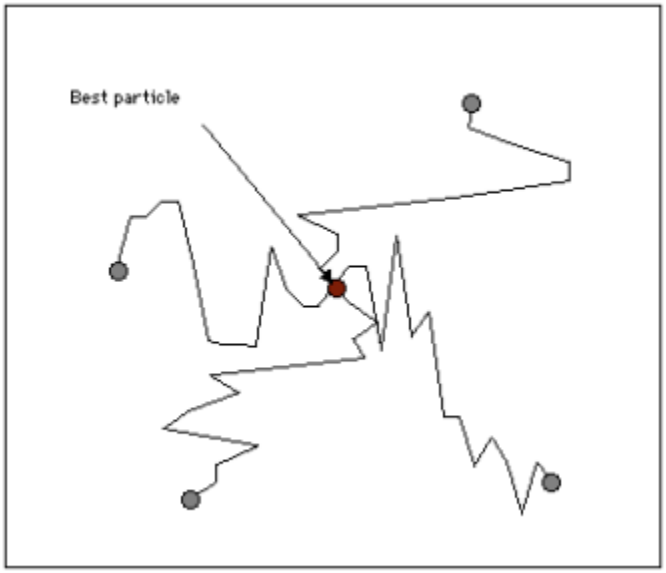
Na equação (1), vi(t) representa o vetor velocidade da partícula i no tempo t, w é o fator de inércia, rnd representa números aleatórios de distribuição uniforme entre 0-1, C1 e C2

representam respectivamente os parâmetros social e cognitivo, PBEST é a melhor posição individual e GBEST é a melhor posição social. Os parâmetros C1 e C2 ajustam o balanço entre a influência social e a aprendizagem da partícula individual.

Algoritmo de otimização de enxame de partículas (PSO) modela dois comportamentos simples: cada indivíduo (1) se move em direção seu melhor vizinho mais próximo e (2) retorna ao estado que o indivíduo experimentou ser o melhor para si mesmo. Como resultado, o comportamento coletivo que emerge é que de todos os indivíduos convergindo para o estado ambiental que é melhor para todos os indivíduos.

Exempo:

Uma forma mais simples de explicação pode ser um bando de gaivotas telepáticas à procura da melhor fonte de comida. Todas começam num mapa distribuídas aleatoriamente, no inicio antes de se moverem têm velocidade zero mas sabem qual a gaivota mais próximo da solução ótima, alteram então a sua velocidade e direcção para irem nesse sentido. À medida que vão andando vão-se lembrando da sua melhor posição, usam a sua velocidade actual, a sua melhor posição e a melhor posição global para alterarem a sua velocidade e logo a sua posição para se dirigirem para a posição óptima. Como iniciaram em posições aleatórias no mapa ao dirigirem-se todas para o mesmo ponto inevitavelmente vão passar pelo ponto óptimo, alterando ao melhor global e fazendo com que todas consigam convergir nesse ponto.



4 Descrição em detalhe do "Firefly algorithm" e análise comparativamente com a versão base, o PSO.

4.1 Firefly algorithm

A maioria das espécies de pirilampos são capazes de brilhar produzindo flashes curtos. Considera-se que a principal função dos flashes é atrair pirilampo do sexo oposto e potenciais presas. Além disso, um flash de sinal pode comunicar a um predador que um pirilampo tem um gosto amargo.

Com base nas propriedades piscantes das espécies de pirilampos, Xin-She Yang desenvolveu o Firefly Algorithm (FA) em 2008. Um sistema não linear que combina o decaimento exponencial da absorção de luz e lei do inverso do quadrado da variação de luz usa a distância. Em FA, a principal equação algorítmica para a posição é xi (como o vetor de solução do problema) é:

$$x_i^{t+1} = x_i^t + \beta_0 e^{-\gamma r_{ij}^2}(x_j^t - x_i^t) + \alpha \epsilon_i^t \tag{2}$$

onde α é um fator de escala que controla o tamanho do passo do passeio aleatório e γé um parâmetro dependente de escala que controla a visibilidade do pirilampo (e o modo de busca). Além disso β_0 é a constante gravitacional quando a distância entre dois pirilampos é zero (ou seja, $r_{ij} = 0$). Este sistema é um sistema não linear que leva a recursos ricos em termos de comportamento algorítmico.

Como o brilho dos pirilampos está relacionado à paisagem objetiva usando sua posição como índice, a atratividade dos pirilampos vistos por outros depende de sua posição relativa e brilho relativo. Portanto, a beleza está nos olhos de quem vê. Portanto, comparações vinculadas são necessárias para comparar todos os pirilampos.

Um exemplo de um problema que pode ser resolvido usando o algoritmo de vaga-lumes é o problema do vendedor viajante (TSP). No TSP, um vendedor deve visitar um número de cidades na rota mais curta possível, visitando cada cidade apenas uma vez e retornando à cidade de origem. O algoritmo de vaga-lumes pode ser usado para encontrar a rota ótima para o vendedor seguir, simulando o comportamento de vaga-lumes, onde o brilho de cada vaga-lume corresponde à qualidade da solução que ele representa. Os vaga-lumes se movem em direção aos vaga-lumes mais brilhantes, e a solução com o maior brilho (menor distância total) é retornada como a solução ótima.

4.2 Análise comparativamente a PSO

O algoritmo Firefly compartilha muitas semelhanças com algoritmos SI, como PSO.

O algoritmo Firefly tem cinco regras baseadas nas propriedades piscantes dos pirilampos reais:

1. Todos os vaga-lumes são capazes de se atrair independentemente do sexo;
2. A atratividade de um vaga-lume para outros indivíduos é proporcional ao seu brilho.
3. Vaga-lumes menos atraentes se movem na direção do mais atraente.
4. À medida que a distância entre dois vaga-lumes aumenta, o brilho visível de um determinado vaga-lume para o outro diminui.
5. Se um vaga-lume não vê nenhum vaga-lume que seja mais brilhante do que ele, ele se move aleatoriamente.

4.2.1 Vantagens

- Parece ser mais eficaz na otimização multiobjetivo e em curto tempo.
- É melhor em problemas não lineares com ruído.
- Fácil de implementar.
- Bom a resolver problemas com poucos Local optimum.

4.2.2 Desvantagens

- Tem uma convergência baixa.
- Fica preso em problemas com muitos de Local optimum.
- Os parametros do algoritmo são fixos e não mudam com o tempo de computação.
- Não memoriza de qualquer história de melhor situação para firefly e isso faz com que eles se movam independentemente de sua melhor situação anterior, podendo acabam perdendo as suas situações.

5 Análsie de desempenho

5.1 Algoritmo função Sphere

$$f(x) = \sum_{i=1}^{\infty}(x^2) \tag{3}$$

5.2 Algoritmo função Ackley

$$f(x) = -aexp(-b\sqrt{\frac{1}{d}\sum_{i=1}^{d}x_i^2}) - exp(\frac{1}{d}\sum_{i=1}^{d}\cos(cx_i)) + a + exp(1) \tag{4}$$

5.3 PSO

5.3.1 Sphere

	1000 Iterações		1000 Iterações	
	Sphere	Sphere	Sphere	Sphere
	Dim = 2	Dim = 2	Dim = 3	Dim = 3
	GlobalBestPSO	GlobalBestPSO	GlobalBestPSO	GlobalBestPSO
Hyperparameters	Best Cost	Best Pos	Best Cost	Best Pos
Cognitive Parameter : C1 = 0,5 Social Parameter : C2 = 0,3 Inertica Parameter : C3 = 0,9	2.3426613822128837e-43	[3.78010095e-22 3.02282164e-22]	2.892168557925198e-43	[3.70725911e-22 1.50180620e-22 3.59478700e-22]
Cognitive Parameter : C1 = 1 Social Parameter : C2 = 0,3 Inertica Parameter : C3 = 0,9	6.82591817230103e-35	[8.26191148e-18 6.51925432e-22]	6.601392309724231e-29	[1.63094340e-15 -7.90336984e-15 -9.43764746e-16]
Cognitive Parameter : C1 = 0,5 Social Parameter : C2 = 0,5 Inertica Parameter : C3 = 0,9	1.655895823107274 7e-42	[-1.13074696e-21 -6.14253311e-22]	7.713450177295974e-41	[-9.75916841e-22 -8.05586987e-21 3.35932267e-21]
Cognitive Parameter : C1 = 0,5 Social Parameter : C2 = 0,3 Inertica Parameter : C3 = 1,5	0.017185260251670408	[-0.11461615 0.06362702]	0.3410916714059792	[0.01915389 0.21052588 -0.54442966]

Em ambos os casos, encontramos os parâmetros iniciais como ótimos. O aumento dos parâmetros cognitivos resultou em custos relativamente mais altos, enquanto o aumento dos parâmetros sociais resultou em 'pequenas' diferenças nos custos. O parâmetro que mais afetou o custo foi a inércia, e aumentá-la resultou em um aumento muito grande no custo. Isso era esperado porque a aceleração da partícula anterior tem um grande impacto em como a posição ideal é alcançada.

5.3.2 Ackley

	1000 Iterações		1000 Iterações	
	Ackley	Ackley	Ackley	Ackley
	Dim = 2	Dim = 2	Dim = 3	Dim = 3
	GlobalBestPSO	GlobalBestPSO	GlobalBestPSO	GlobalBestPSO
Hyperparameters	Best Cost	Best Pos	Best Cost	Best Pos
Cognitive Parameter : C1 = 0,5 Social Parameter : C2 = 0,3 Inertica Parameter : C3 = 0,9	4.440892098500626e-16	[2.43017952e-16 5.75366776e-17]	4.440892098500626e-16	[1.49131119e-17 2.21907232e-17 -1.94572574e-17]
Cognitive Parameter : C1 = 1 Social Parameter : C2 = 0,3 Inertica Parameter : C3 = 0,9	1.1102230246251565e-14	[3.55484677e-15 2.27967387e-15]	4.440892098500626e-16	[1.22054023e-16 -1.35944167e-16 2.68470410e-16]
Cognitive Parameter : C1 = 0,5 Social Parameter : C2 = 0,5 Inertica Parameter : C3 = 0,9	4.440892098500626e-16	[-3.09068469e-16 -1.79678046e-16]	4.440892098500626e-16	[-3.12487365e-16 1.69535904e-17 -1.18603468e-16]
Cognitive Parameter : C1 = 0,5 Social Parameter : C2 = 0,3 Inertica Parameter : C3 = 0,2	1.7055433336693864	[0.08731274 0.22147976]	2.334126531312514	[0.03548606 0.20605561 0.33586263]

O mesmo pode ser visto com Ackley. Em ambos os casos, os parâmetros cognitivos tiveram um pequeno efeito sobre os custos e os parâmetros sociais não tiveram efeito. Novamente, a aceleração das partículas anteriores teve o maior impacto no custo.

5.4 Firefly

5.4.1 Sphere

	1000 Iterações		1000 Iterações	
	Sphere	Sphere	Sphere	Sphere
	Dim = 2	Dim = 2	Dim = 3	Dim = 3
	Firefly	Firefly	Firefly	Firefly
Hyperparameters	Best Cost	Best Pos	Best Cost	Best Pos
Csi = 1 Psi = 1 Alpha0 = 1 Alpha1 = 0,1 Norm0 = 0 Norm1 = 0,1	0.00395186407770451	[0.0222734502443322, 0.05878569121748733]	0.01282675493322168	[-0.07994145925209713, 0.07880615771486285, 0.015023565891196938]
Csi = 2 Psi = 1 Alpha0 = 1 Alpha1 = 0,1 Norm0 = 0 Norm1 = 0,1	0.038296738751059156	[0.07821306007499607, 0.17938633165535275]	0.0014109238185286003	[0.031619468924901806, -0.019721120243566564, 0.004712793203033472]
Csi = 1 Psi = 2 Alpha0 = 1 Alpha1 = 0,1 Norm0 = 0 Norm1 = 0,1	0.0022051175913362737	[0.04605037278821221, -0.009191341436534 62]	0.023354783004942564	[-0.04310586398747744, -0.09645917010338433, -0.110418730285228 44]
Csi = 4 Psi = 1 Alpha0 = 1 Alpha1 = 0,1 Norm0 = 0 Norm1 = 0,1	0.0008633931524454859	[-0.028230171641905735, -0.008151721383488714]	0.04814147408701885	0.020995212914999974, -0.18613696653209552, 0.11425281095842768]

Para FA, já temos mais parâmetros para alterar, o que nos dá uma direção maior para tentar mais variações de parâmetros. Comparação com valores padrão:

Um leve aumento em Csi (atração mútua) aumentou o custo na dimensão 2, mas uma diminuição na dimensão 3 deu-nos os melhores resultados.A modificação do Psi (coeficiente de absorção óptica) resultou em uma ligeira redução de custo na 2ª dimensão e um ligeiro aumento na 3ª dimensão.

Infelizmente, os outros parâmetros eram muito pesados e demorados computacionalmente, então não os alteramos. Portanto, experimentamos com Csi significativamente aumentado. Com isso obtivemos o valor mais alto na dimensão 2. Isso é surpreendente dado os aumentos anteriores, e é o aumento na dimensão 3, que anteriormente tinha o menor custo.

5.4.2 Ackley

	1000 Iterações		1000 Iterações	
	Ackley	Ackley	Ackley	Ackley
	Dim = 2	Dim = 2	Dim = 3	Dim = 3
	Firefly	Firefly	Firefly	Firefly
Hyperparameters	Best Cost	Best Pos	Best Cost	Best Pos
Csi = 1 Psi = 1 Alpha0 = 1 Alpha1 = 0,1 Norm0 = 0 Norm1 = 0,1	0.050444783767076284	[-0.061146838445405444, 0.030807186654672561]	0.3951026051864819	-0.87455501485315, 0.10415995680749096, -5.567606740780509]
Csi = 2 Psi = 1 Alpha0 = 1 Alpha1 = 0,1 Norm0 = 0 Norm1 = 0,1	0.03894687250092410 *rever linha	[0.035853421036423985, 0.03949227647859949]	-1.7196527716305714	[-0.041362575164262214, -0.03920582231180181, 0.009386051371798973]
Csi = 1 Psi = 2 Alpha0 = 1 Alpha1 = 0,1 Norm0 = 0 Norm1 = 0,1	1.7628721343308305	[-0.23759383818889362, 5.059976189960237]	0.4757564693300358	[-0.5189948225858666, 0.05198846105591157, -5.202112601826584]
Csi = 4 Psi = 1 Alpha0 = 1 Alpha1 = 0,1 Norm0 = 0 Norm1 = 0,1	0.3141668269078335	[-0.06545994027392536, -0.3576333396044171]	-1.4815465879388028	[-0.11267833180690037, 0.0003136210919636450 7, 0.27053203948772364]

Ackley deu resultados diferentes. Em ambos os casos, o aumento de Csi tornou o custo negativo. Em ambos os casos, o aumento do psi aumentou o custo. E seguimos a mesma filosofia da esfera, aumentando significativamente o Csi, o que causou o custo na dimensão 2 a aumentar e o custo na 3 dimensão a ficar novamente negativo.

6 Conclusões

Podemos observar pelos gráficos acima que os valores obtidos pelo algoritmo PSO são melhores (custo mais aproximado a 0) para este caso de problema.

Na função Ackley para o algoritmo firefly, é possível um custo negativo porque a função inclui um termo que é o quadrado do negativo do valor de entrada. Isso significa que, para determinados valores de entrada, esse termo pode ser negativo, o que pode resultar em um custo total negativo para a função.

Apesar de nas pequisas que efetuamos já muitos casos se verificam que o FA é um algoritmo que tem bom desempenho em problemas com um grande número de variáveis e uma função objetiva complexa e não linear. Exemplos de tais problemas incluem o problema do vendedor viajante, o problema da mochila e o problema de otimização multiobjetivo. Em geral, o falar é adequado para problemas em que o espaço de solução é contínuo e a função objetiva é difícil de otimizar usando métodos tradicionais, o que não é o caso aqui.

Esperamos que, com este relatório, fique claro o que são o PSO e o Firefly Algorithm, como eles funcionam, para que são úteis e quais as diferenças entre eles.

7 Referências

[1] Firefly vs PSO.

[2] Firefly parametros.