Multilayer Perceptron Perceptron optimization through Simulated Annealing and Fast Simulated Annealing

Pedro H. Cardoso Camelo[1] e Rafael Lima de Carvalho[1]

[1] *Curso de Ciência da Computação, Universidade Federal do Tocantins, Tocantins, Brasil*

Data de recebimento do manuscrito: 30/05/2020
Data de aceitação do manuscrito: 10/06/2020
Data de publicação: 12/06/2020

Abstract— The Multilayer Perceptron (MLP) is a classic and widely used neural network model in machine learning applications. As the majority of classifiers, MLPs need well-defined parameters to produce optimized results. Generally, machine learning engineers use grid search to optimize the hyper-parameters of the models, which requires to re-train the models. In this work, we show a computational experiment using metaheuristics Simulated Annealing and Fast Simulated Annealing for optimization of MLPs in order to optimize the hyper-parameters. In the reported experiment, the model is used to optimize two parameters: the configuration of the neural network layers and its neuron weights. The experiment compares the best MLPs produced by the SA and FastSA using the accuracy and classifier complexity as comparison measures. The MLPs are optimized in order to produce a classifier for the MNIST database. The experiment showed that FastSA has produced a better MLP, using less computational time and less fitness evaluations.

Keywords— Neural Network, Multilayer Perceptron, Simulated Annealing, MNIST Database

I. Introdução

As redes neurais artificiais (RNAs) se popularizaram grandemente pela capacidade de aproximação de funções não-lineares. Alguns problemas de classificação múltipla podem ser aproximados por tais funções, por exemplo. O modelo *Multilayer Perceptron* (MLP) teve seu auge inicial na década de 60 [1], ao ser incorporado o algoritmo *Backpropagation*, para realizar o treinamento de seus pesos. Uma MLP é composta de pelo menos três camadas: a camada de entrada, uma camada oculta e uma camada de saída. A quantidade de neurônios na camada de entrada é dependente diretamente da quantidade de características envolvidas na aplicação. O número de neurônios assim como o número de camadas ocultas compõem parâmetros ajustáveis da rede. A camada de saída, para problemas de classificação, em geral possuem o mesmo número de classes da previsão desejada [2].

Em geral, ao se utilizar uma MLP para classificação, os parâmetros devem ser ajustados de maneira a permitir a otimização do resultado do algoritmo. Existem diversos algoritmos que auxiliam nesta etapa, como o RPROP [3] e até algoritmos evolutivos como o Algoritmos Genéticos [4]. No caso das MLPs, a escolha de seus parâmetros como configuração das camadas, peso dos neurônios e taxa de aprendizado influenciam na sua eficiência. Dentre os algoritmos capazes de realizar a otimização de tais parâmetros, este experimento considerou o algoritmo *Simulated Annealing* (SA) [5].

Simulated Annealing é um algoritmo metaheurístico para otimização, consistindo em uma técnica de busca local probabilística que simula o processo de recozimento utilizado na metalurgia para obtenção de estados de baixa energia em sólidos [6]. O processo metalúrgico envolve duas principais etapas [7]:

1. Aquecer o material a uma temperatura próxima de 1100^{o}C
2. Realizar o resfriamento acompanhado até que o material se solidifique.

Em geral o processo de SA pode ser acelerado, ao ser incorporado outras funções de resfriamento. Uma das variações populares consiste no *Fast Simulated Annealing (FastSA)*. A exemplo de sua melhora da eficiência, em [8], os autores o aplicaram com o objetivo de reduzir o tempo de execução do algoritmo para a resolução do problema de agendamento de horários para exames avaliativos.

Diante do exposto, este artigo apresenta um experimento computacional, mostrando a aplicação do Simulated Annealing e sua variação Fast Simulated Annealing no problema de otimização de parâmetros de uma rede neural MLP para classificação da base de dados de dígitos manuscritos MNIST [9]. O restante do artigo está organizado da seguinte maneira: A Seção II apresenta os algoritmos utilizados, a estruturação da base de dados e como os algoritmos foram configurados para resolução do problema em questão. Na Seção III são apresentados os resultados das simulações implementadas sob

Dados de contato: Pedro H. Cardoso Camelo, pedrohc8407@gmail.com

as medidas adotadas para o desempenho do algoritmo de classificação. Por fim, a Seção IV apresenta as considerações finais do experimento.

II. METODOLOGIA

Nesta seção são descritos os algoritmos SA e FastSA, a base de dados MNIST e a aplicação destes para a resolução do problema.

a. Simulated Annealing

O algoritmo se inicia definindo uma solução inicial aleatória dentro do espaço do problema ($curr \in P$), temperatura inicial(T) e mínima (Tmin), taxa de resfriamento (Tx), número de iterações(k). Em seguida ele inicia o processo de "resfriamento", em que a temperatura decresce a uma taxa definida ($C(T) = Tx * T$) até que atinja o limite mínimo ($T \geq Tmin$), geralmente 0, sendo cada iteração de resfriamento chamada de *época*. A cada época de resfriamento, é feito um laço com o máximo de k iterações. Nele, a cada iteração k, é escolhida uma nova solução pertencente à vizinhança atual, chamada de vizinho($v \in N$), as duas então são comparadas de acordo com uma função de energia (E) e, caso o vizinho apresente uma melhor energia, ele se torna a solução atual, configurando um movimento. Mesmo se o vizinho se mostrar pior, ele ainda pode ser aceito de acordo com o fator de Boltzmann, que é dado por:

$$e^{-\frac{\Delta}{T}}$$

Onde Δ é a diferença das avaliações das duas soluções consideradas. Ao final a solução atual é retornada como a melhor solução. A Figura 1 mostra um pseudocódigo do SA.

Algorithm 1: Algoritmo Simulated Annealing

Input: Taxa de resfriamento
$s \leftarrow s_0$; ▷ *Solução inicial*
$T \leftarrow T_{max}$; ▷ *Temperatura inicial*
repeat
  repeat ▷ *A uma temperatura fixa*
    Gerar um vizinho aleatório $nb \in N(s)$
    Comparar as energias

$$\triangle E = \frac{E(nb) - E(s)}{E(s)}$$

    if $\triangle E \leq 0$ **then**
      Aceite *nb* como a nova melhor solução;
    else if $random(0,1) \leq B$ **then**
      Aceite *nb* como a nova melhor solução;
  until *Máximo de iterações (k)*;
  $T \leftarrow C(T)$; ▷ *Atualização da temperatura*
until *Temperatura mínima* ($T < T_{min}$);
Output: Melhor solução encontrada

b. Fast Simulated Annealing

Desenvolvido por Nuno Leitea, Fernando Melícioa e Agostinho C. Rosac [8] para resolver o problema de alocação de horários para exames avaliativos, o FastSA é um algoritmo derivado do SA, possuindo alterações com o intuito de reduzir o número de avaliações (comparações entre as soluções) realizadas pelo algoritmo. Nele, são armazenados os movimentos feitos em cada época. Quando se seleciona um vizinho, é verificado se houve movimentos para ele na época anterior. Em caso negativo, ele não é avaliado e não é feito nenhum movimento. De acordo com os autores, este algoritmo consegue resultados mais rápidos, ao custo da degradação dos resultados. A Figura **??** apresenta o pseudocódigo do algoritmo utilizado para o problema dos autores.

c. Base de dados MNIST

MNIST é uma base de dados de dígitos manuscritos composta por 70.000 imagens, sendo bastante popular para tarefas de aprendizado e reconhecimento de padrões. É dividida em 60.000 exemplos de treinamento e 10.000 exemplos de teste, como imagens devidamente normalizadas e centralizadas, gerando, assim, uma economia de tempo de pré-processamento. Na Figura 1 é exibida uma amostragem desta base de dados.

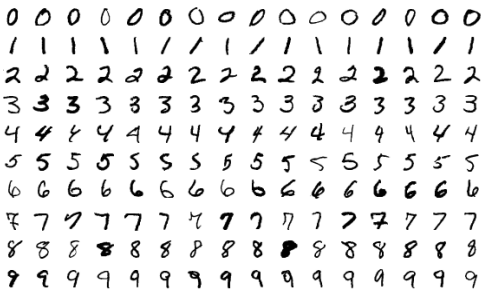

Figura 1: Amostra da base de dados MNIST [10].

d. Aplicação

Nesta seção são descritos os algoritmos aplicados ao problema que este trabalho busca resolver.

1. Pré-Processamento

Antes de se iniciar a aplicação, os dados precisam ser pré-processados, sendo um procedimento padrão na maioria das atividades de aprendizado de máquina. Mesmo com o auxílio da própria base de dados, que já se encontra pré-processada, algumas características ainda precisam ser tratadas. Neste trabalho, as amostras foram transformadas em vetores unidimensionais de 784 elementos, sendo estes elementos os próprios pixels das imagens de tamanho 28x28, codificados em escala de cinza. O vetor de classes das amostras foi transformado em uma matriz bidimensional através da técnica de *One Hot Encoding*, que facilita a classificação pela rede.

2. Soluções

As soluções utilizadas no algoritmo são redes neurais MLP. A solução inicial é uma MLP com os parâmetros:

- Taxa de Aprendizado: 0.1;
- Camada Interna: 3 níveis de tamanhos iguais com valor aleatório entre 1 e 100 neurônios;

Algorithm 2: Algoritmo Fast Simulated Annealing

Input: Taxa de resfriamento

prevBin $\leftarrow$ *nil* ▷ *Lista contendo "n" movimentos aceitos na época anterior, inicialmente vazio*
currBin $\leftarrow$ *createArray*(*numExams*) ▷ *Lista contendo "n" movimentos aceitos na época atual, inicializada com zeros*
$s \leftarrow s_0$; ▷ *Solução inicial*
$T \leftarrow T_{max}$; ▷ *Temperatura inicial*
repeat
  repeat ▷ *A uma temperatura fixa*
    Gerar um vizinho aleatório $nb \in N(s)$; *examToMove* $\leftarrow$ *selectExamFromSolution*(*nb*) ▷ *Seleciona um exame que será movido*
    if *prevBin* = *nil or*(*prevBin* <> *nil and prevBin*[*examToMove*] > 0) **then**
      Calcular a energia ($\triangle E$)
      if $\triangle E \leq 0$ **then**
        Aceite *nb* como a nova solução;
      else if $random(0,1) \leq B$ **then**
        Aceite *nb* como a nova solução;
      if *nb foi aceito* **then**
        currBin[*examToMove*] $\leftarrow$ *currBin*[*examToMove*] + 1
  until *Condição de equilíbrio* ▷ *Exemplo: máximo de iterações alcançado*;
  $T \leftarrow C(T)$; ; ▷ *Atualização de temperatura*
  prevBin $\leftarrow$ updateBin(T,currBin) ; ▷ *Se T>=Tmin, prevBin*$\leftarrow$ *currBin e zere currBin*
until *Critério de parada*;

Output: Melhor solução encontrada

- Máximo de iterações: 200
- Função de ativação: ReLU, Unidade Linear Rectificada;
- Camada de entrada: 784 neurônios, sendo os pixels das imagens de tamanho 28x28.
- Camada de Saída: 10 neurônios, sendo as classificações possíveis, que vão de 0 a 9.

Os parâmetros que foram alterados para seleção da melhor rede são a Taxa de Aprendizado e a configuração da Camada Interna. Assim, a solução vizinha é escolhida a partir da solução atual, alterando os parâmetros da Camada Interna e Taxa de Aprendizado, seguindo as regras:

- Quantidade de níveis da camada interna: Soma a um valor inteira aleatório entre -2 e 2, gerando a possibilidade de aumentar, diminuir ou se manter inalterada;
- Quantidade neurônios por nível: Soma ao o valor -4, 0 ou 4, visto em testes que as mudanças mais significativas na energia se davam ao se utilizar estes valores;
- Taxa de aprendizado: Soma a um valor aleatório obtido de um distribuição uniforme entre -0,05 e 0,05.

3. Energia

A função de energia é dada pela acurácia da solução, que é o resultado de seu aprendizado.

$$E(s) = score(S)$$

4. Resfriamento

O resfriamento é realizado através da função:

$$C(T,k) = T/(1+n).$$

Sendo n a época em que se encontra o algoritmo.

e. Treinamento

1. Busca

Para os dois algoritmos, SA e FastSA, foram definidos os parâmetros k, T e $Tmin$ como 5,0, 2,0 e 0,001 respectivamente. Para aumentar ainda mais a confiabilidade dos resultados, os algoritmos foram executados, cada um, 10 vezes, gerando 10 redes cada. Estas então foram comparadas de acordo com sua acurácia e algoritmo utilizado, resultando em 2 redes, uma resultante do SA e outra do FastSA.

2. Tamanho da Base

Como os algoritmos demandam um tempo de execução razoavelmente longo, o processo pode se tornar custoso pois os dados são utilizados extensivamente durante a execução, principalmente no processo do SA. Para sanar este problema, foi utilizada inicialmente uma parcela de dez por cento (10%) da base, respeitando as proporções entre a divisão de treinamento e teste. Após a finalização da busca, as redes resultantes são treinadas com a totalidade dos dados.

III. RESULTADOS

Para a avaliação dos algoritmos, foram utilizados os resultados de tempo de execução, número de avaliações e acurácia da rede obtida. Para um maior aprofundamento, foram utilizadas outras métricas para avaliar as redes. As Tabelas 1 e 2 mostram as comparações entre os resultados de execução dos algoritmos e suas respectivas redes obtidas.

Como se pode observar, o algoritmo FastSA obteve resultados superiores em relação ao tempo computacional e a quantidade de chamadas à função de avaliação. Além do mais, ainda que a degradação dos resultados é um fator apontado como esperado, o FastSA permitiu gerar uma

TABELA 1: RESULTADO DE DESEMPENHO DOS ALGORITMOS AVALIADOS.

Alg.	Tempo (seg.)	Acc. (Parcial)	Nº de Aval	Acc (Total)
SA	4350,06	0,9070	300	0,9656
FastSA	482,58	0,9090	50	0,9663

TABELA 2: MELHORES CONFIGURAÇÕES DE MLPS OBTIDAS PELOS ALGORITMOS SELECIONADOS.

Alg.	Camada Interna	Taxa de Aprendizado
SA	4 níveis 85 Neur. / nível	0,14
FastSA	3 níveis 79 Neur. / nível	0,06

MLP menos complexa, enquanto que a diferença nas acurácias foi mínima, sobretudo ainda superior na configuração encontrada pelo FastSA.

IV. CONCLUSÃO

Este artigo apresentou a utilização do *Simulated Annealing* e sua variante *Fast Simulated Annealing* para otimização dos parâmetros de uma rede neural MLP, com objetivo de maximizar a acurácia de classificação na base de dados de dígitos manuscritos MNIST. Pelo experimento, observou-se que o algoritmo *Fast Simulated Annealing* obteve resultados superiores em relação ao *Simulated Annealing* convencional nos seguintes índices: 83,33% menos chamadas à função de avaliação, tempo computacional 88,91% menor.

Em relação à Rede Neural gerada, percebeu-se que a acurácia dos melhores classificadores resultantes tanto no SA quanto no FastSA foi basicamente a mesma (ou seja, ambos permitiram ajustes para um classificador otimizado). Sobretudo, vale destacar que a rede neural gerada pelo FastSA é de menor complexidade.

REFERÊNCIAS

[1] S. Haykin, *Neural networks: a comprehensive foundation*. Prentice Hall PTR, 1994.

[2] S. Raschka, *Python machine learning*. Packt Publishing Ltd, 2015.

[3] M. Riedmiller and H. Braun, "A direct adaptive method for faster backpropagation learning: the rprop algorithm," in *IEEE International Conference on Neural Networks*, March 1993, pp. 586–591 vol.1.

[4] W. Kinnebrock, "Accelerating the standard backpropagation method using a genetic approach," *Neurocomputing*, vol. 6, no. 5-6, pp. 583–588, 1994.

[5] P. A. Castillo, J. J. Merelo, J. González, V. Rivas, and G. Romero, "Saprop: Optimization of multilayer perceptron parameters using simulated annealing," in *International Work-Conference on Artificial Neural Networks*. Springer, 1999, pp. 661–670.

[6] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," *science*, vol. 220, no. 4598, pp. 671–680, 1983.

[7] S. Kirkpatrick, "Optimization by simulated annealing: Quantitative studies," *Journal of statistical physics*, vol. 34, no. 5-6, pp. 975–986, 1984.

[8] N. Leite, F. Melício, and A. C. Rosa, "A fast simulated annealing algorithm for the examination timetabling problem," *Expert Systems with Applications*, vol. 122, pp. 137–151, 2019.

[9] L. Deng, "The mnist database of handwritten digit images for machine learning research [best of the web]," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 141–142, 2012.

[10] J. Steppan. (2020) Own work, cc by-sa 4.0. Tomado de https://commons.wikimedia.org/w/index.php?curid=64810040 (30/05/2020).