Implementação e Validação do Multilayer Perceptron com *Backpropagation*

Jesuino Vieira Filho
Centro Tecnológico de Joinville
Universidade Federal de Santa Catarina
Joinville, Brasil
jesuino.vieira@grad.ufsc.com

***Resumo*—Este relatório tem por objetivo descrever as atividades desenvolvidas durante a implementação e validação de uma rede neural artificial do tipo *multilayer perceptron* (MLP). Este tipo de rede pode ser utilizada para a classificação de problemas multi-classe e não linearmente separáveis. Para sua validação, foram realizados experimentos variando a arquitetura da rede, a taxa de aprendizado e o *momentum*. As MLPs foram treinadas e testadas em um subconjunto da base de dados MNIST e permitiram avaliar o impacto das diferentes configurações, assim como identificar possíveis falhas de implementação.**

Index Terms*—rede neural artifical, *multi layer perceptron

I. Introdução

Uma rede neural do tipo *multilayer perceptron* (MLP) consiste em pelo menos três camadas de neurônios: uma camada de entrada, uma camada oculta e uma camada de saída. Cada neurônio utiliza uma função de ativação não linear, exceto os de entrada. Para treinamento, utiliza-se uma técnica de aprendizado supervisionado denominada *backpropagation*. Suas múltiplas camadas e ativação não linear distinguem a MLP de um perceptron, sendo capaz de aprender padrões complexos e distinguir dados que não são linearmente separáveis.

O presente relatório encontra-se organizado da seguinte forma. Na Seção II, aspectos relacionados a implementação da MLP são brevemente descritos, assim como os detalhes acerca dos experimentos realizados. Na Seção III, os resultados obtidos são relatados de forma simples e objetiva. Por fim, na Seção IV são apresentadas as interpretações dos resultados e considerações finais acerca do referido trabalho.

II. Materiais e Métodos

Esta seção é dividida em duas partes. Primeiro, detalhes acerca da implementação são brevemente apresentados. Em sequência, os experimentos realizados para validação são descritos com mais detalhes, em que são descritas as configurações de rede e a base de dados utilizadas.

A. Implementação

A implementação da MLP foi desenvolvida utilizando a linguagem de programação Python, em sua versão 3.9.6. Para tanto, um módulo denominado `mlp` foi implementado. Neste módulo, uma classe denominada `MLP` permite inicializar, configurar, treinar, avaliar e utilizar a rede através de uma interface similar a da biblioteca TensorFlow (popular escolha para o desenvolvimento de aplicações de aprendizado de máquina), como evidenciado no Algoritmo 1.

Algoritmo 1: Exemplo de uso do módulo implementado.

```
m = mlp.MLP()
m.add(Input(units=isize))
m.add(Dense(units=hsize, eta=eta, alpha=
    alpha, activation="tanh"))
m.add(Dense(units=osize, eta=eta, alpha=
    alpha, activation="tanh"))
m.compile(eta=eta, loss="mse", tol=0.0)
m.fit(X_train, y_train, epochs=50)
acc, loss = m.evaluate(X_test, y_test)
```

O usuário é capaz de configurar o modelo para a etapa de treinamento em tempo de execução, como pode ser observado na chamada dos métodos `add`, `compile` e `fit`. Nessa situação, os seguintes parâmetros podem ser controlados: o **número de camadas** (através do método `add`), o **número de neurônios em cada camada** (units), a **taxa de aprendizado** (eta), o ***momentum*** (alpha), a **função de ativação** (activation), a **função de custo** (loss), a **tolerância** (tol) e o número de **épocas de treinamento** (epochs).

Para treinar a rede, o algoritmo denominado *backpropagation* seguindo o método online de aprendizagem supervisionada foi utilizado. Isto significa que os ajustes aos pesos sinápticos são realizados exemplo a exemplo. Toda a implementação da MLP foi baseada nas definições fornecidas por Haykin [1]. Desse modo, vale ressaltar que os pesos são inicializados seguindo uma distribuição uniforme, com valores entre -1 e 1. Além disso, antes de cada época os dados são embaralhados.

B. Experimentos

1) Configurações de Rede: Com o objetivo de validar a implementação, foram realizados alguns experimentos computacionais. Para estes experimentos considera-se as seguintes variações de parâmetros para a rede:

- **Número de camadas escondidas:** foram considerados dois valores, a saber: 1 e 2. Para uma camada escondida, foram considerados quatro redes com 16, 32, 64 e 128

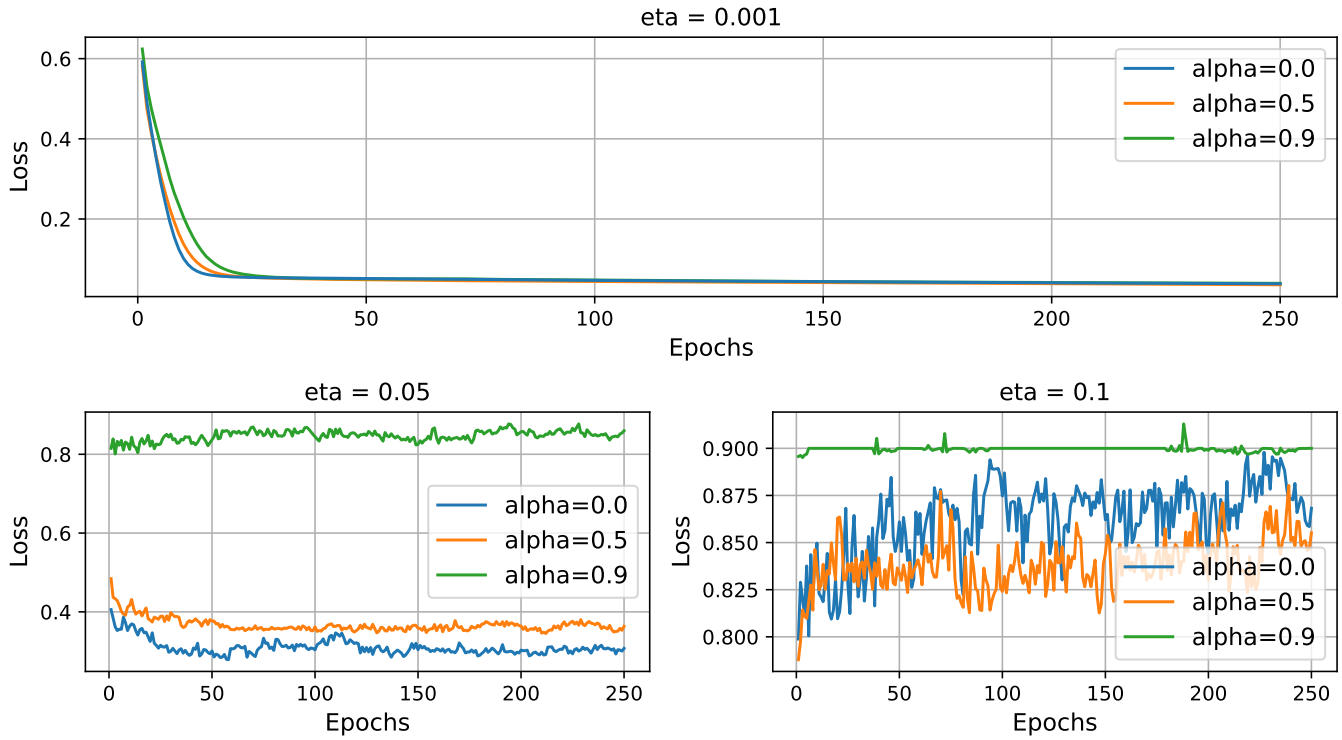

Figura 1: Curva de aprendizado para diferentes valores de taxa de aprendizado (eta) e *momentum* (alpha). Valores referentes à arquitetura com 1 camada escondida com 64 neurônios.

neurônios na camada escondida. Devido ao tempo de treinamento, para duas camadas escondidas foram considerados apenas duas redes: 16-16 e 128-16.

- **Taxa de aprendizado:** foram considerados três valores, a saber: 0.001, 0.050 e 0.100.
- ***Momentum:*** foram considerados três valores, a saber: 0.0, 0.5 e 0.9.

Assim, 54 diferentes configurações de MLP foram analisadas: 6 diferentes arquiteturas de redes x 3 valores de taxa de aprendizado para cada arquitetura x 3 valores de *momentum* para cada arquitetura = 54. Vale a pena ressaltar que cada uma das redes possui 784 neurônios de entrada e 10 neurônios de saída, cada um correspondendo a uma das dez classes.

Por outro lado, alguns parâmetros foram fixados a fim de minimizar o tempo de processamento necessário para realização dos experimentos. Optou-se por utilizar 250 épocas de treinamento, o erro quadrático médio como função de custo, a tangente hiperbólica como função de ativação dos neurônios e um valor 0 como tolerância (ou seja, as 250 épocas de treinamento serão executas caso o modelo não apresente erro nulo em uma época).

Embora este foi o planejamento dos experimentos, o tempo necessário para treinar todas as combinações não foi suficiente até a data de entrega deste relatório. Em vista disso, apenas as combinações que envolvem uma única camada com 16, 32 e 64 neurônios será discutida, totalizando apenas 27 combinações diferentes redes.

2) Base de Dados: Optou-se por utilizar um subconjunto da base de dados MNIST [2]. Este subconjunto é representado por 3000 objetos, em que cada um representa uma imagem de um dígito manuscrito entre 0 e 9. A imagem é quadrada e possui 28x28 *pixels* em escala de cinza. Sendo assim, este conjunto de dados possui 784 atributos e 10 possíveis classes.

Dos 3000 objetos, 2/3 foram utilizados para treino e 1/3 para teste. Tal separação foi realizada de maneira estratificada para manter o balanceamento entre as diferentes classes. Além disso, todas os atributos de entrada (784 pixels) foram normalizados entre 0 e 1.

III. Resultados

A seção atual é dividade em duas partes. Primeiro são apresentados os resultados relacionados a etapa de treinamento, e em seguida, os resultados obtidos no conjunto de teste.

A. Treino

Na Figura 1 são apresentadas as curvas de aprendizado para uma rede de 1 camada escondida com 64 neurônios. Nota-se que, para maiores valores de taxa de aprendizado o processo de treinamento é instável. De fato, a função de custo parece convergir apenas para uma taxa de aprendizado de 0.001.

Quanto ao *momentum*, observa-se um comportamento diferente do esperado, em que a convergência mais rápida do modelo é obtida sem *momentum* (ou seja, alpha = 0.0). Em contrapartida, espera-se que maiores valores de *momentum* levem a uma convergência mais rápida.

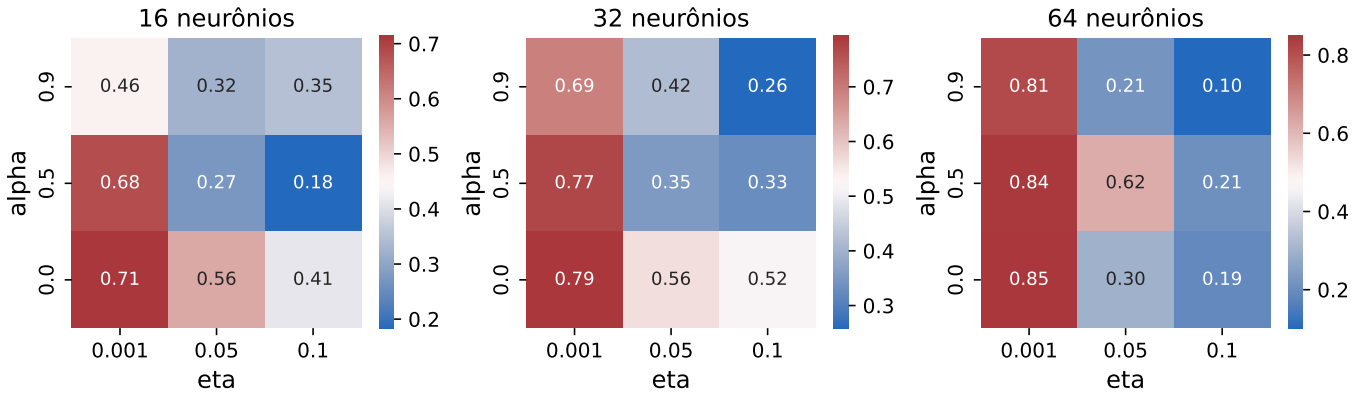

Figura 2: Acurácia obtida para cada uma das arquiteturas de 1 camada escondida em suas diferentes configurações. XL-YN significa o número de camadas ocultas (X) e neurônios (Y). Note que a acurácia foi calculada no conjunto de teste.

Os mesmo resultados foram observados para as outras arquiteturas de rede analisadas. Figuras com tais resultados são omitidos desta seção (uma vez que observaram-se comportamentos semelhantes) e apresentadas no Apêndice A.

B. Teste

A fim de avaliar a capacidade de generalização das diferentes arquiteturas utilizadas, foi calculado a acurácia de cada modelo no conjunto de teste. Os resultados são apresentados na Figura 2. Nota-se que, em geral, quanto menor ambos os parâmetros que foram variados (taxa de aprendizado e *momentum*), melhor é a acurácia do modelo. Todavia, vale ressaltar que essa conclusão é estabelecida da maneira em que a MLP foi implementada, podendo conter erros conforme os resultados incoerentes apresentados na Seção III-A.

Ademais, considerando apenas os valores de taxa de aprendizado em que a MLP convergiu (ou seja, 0.001), percebe-se que aumentar a quantidade de neurônios resulta numa melhor acurácia para a rede. Isso se mostra verdade para todos os valores de *momentum*.

IV. Discussões

A implementação da rede neural do tipo *multilayer perceptron* permitiu um maior entendimento a respeito da lógica matemática que descreve uma MLP. Destaca-se o estudo do algoritmo de treinamento da rede, o *backpropagation*. Os experimentos realizados para a validação também foram de grande valia, uma vez que foi possível analisar o impacto na variação de diferentes parâmetros, assim como identificar possíveis falhas de implementação.

No que diz respeito a taxa de aprendizado, deve-se investigar com maior fundamento se de fato uma maior taxa de aprendizado impede a convergência do modelo. Também há outros fatores que podem influenciar este resultado, como por exemplo, caso o espaço de características (i.e., *feature space*) da base de dados seja complexo.

Por fim, com relação ao *momentum*, dado que os resultados obtidos vão contra a teoria, acredita-se que a implementação possui um defeito. Deve-se-ia explorar o código a fim de entender o que está acontecendo e corrigir os possíveis erros.

Referências

[1] S. S. Haykin, *Neural networks and learning machines*, 3rd ed. Upper Saddle River, NJ: Pearson Education, 2009.

[2] Y. LeCun and C. Cortes, "MNIST handwritten digit database," 2010. [Online]. Available: http://yann.lecun.com/exdb/mnist/

Apêndice A
Resultados Adicionais

Além dos resultados já discutidos anteriormente, são apresentados neste Apêndice resultados adicionais obtidos durante o treinamento das diferentes arquiteturas.

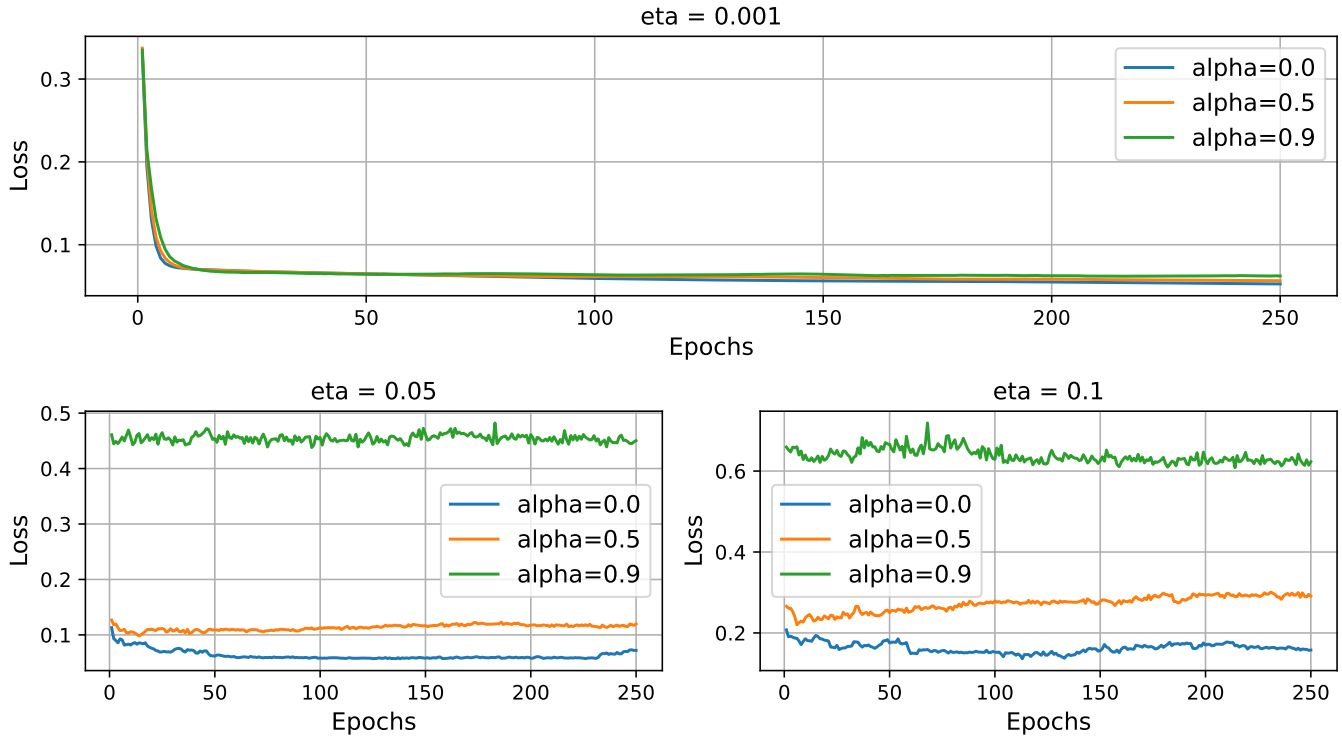

Figura 3: Curva de aprendizado para diferentes valores de taxa de aprendizado (eta) e *momentum* (alpha). Valores referentes à arquitetura com 1 camada escondida com 16 neurônios.

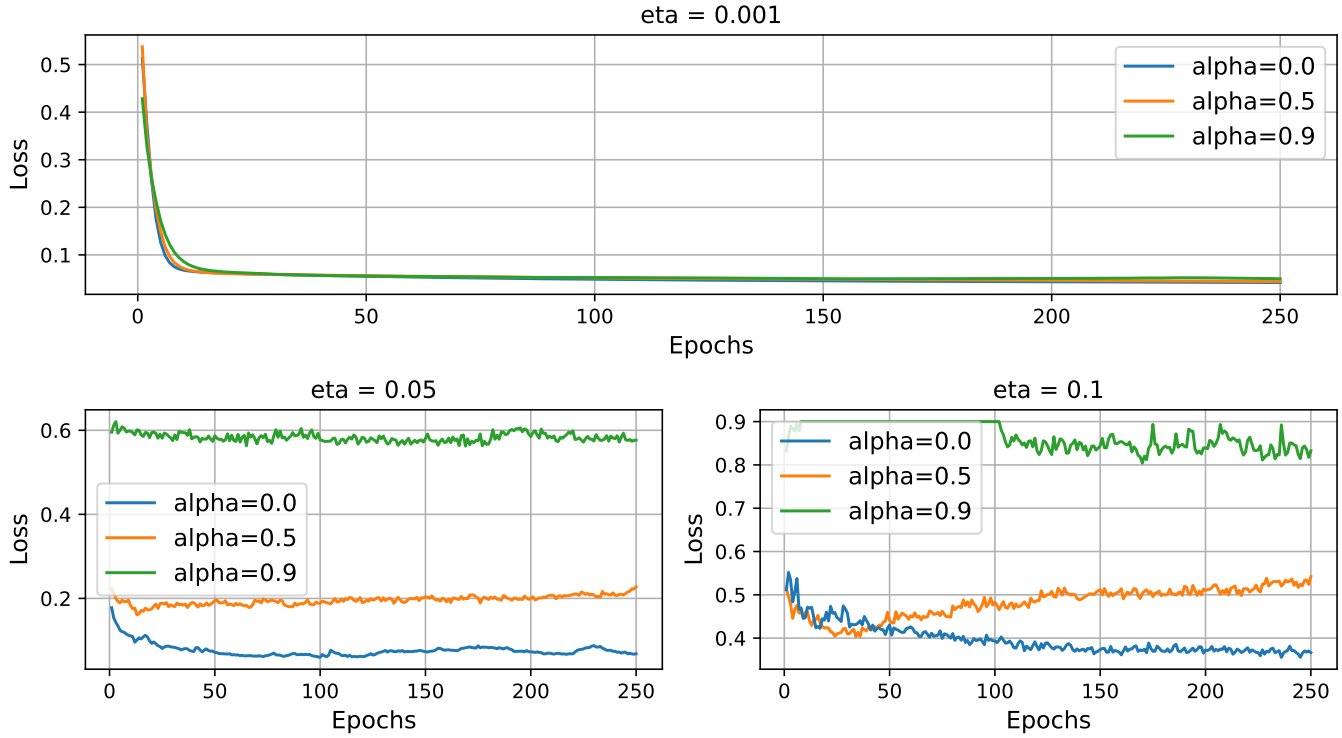

Figura 4: Curva de aprendizado para diferentes valores de taxa de aprendizado (eta) e *momentum* (alpha). Valores referentes à arquitetura com 1 camada escondida com 32 neurônios.