Trabalho 2 - MNIST Database

Disciplina: Tópicos em Engenharia - Sistemas Inteligentes 2019/1

Matheus Schmitz Oliveira 15/0018371
Universidade de Brasília (UnB)
Brasília, Brazil

Pedro Aurélio Coelho de Almeida 14/0158103
Universidade de Brasília (UnB)
Brasília, Brazil

***Abstract*—O projeto tem como objetivo explorar a base de dados MNIST, constituída de imagens de dígitos manuscritos. Os dados foram normalizados e treinados por uma rede neuronal artificial conhecida por *Perceptron multicamada*. Variações dessa rede com relação à sua configuração padrão, bem como o número de épocas e taxa de aprendizado foram testados para avaliar sua influência na qualidade da classificação. Por fim, escolheu-se um dos modelos que manteve sua capacidade de generalização alta e analisou-se a matriz de confusão.**

***Index Terms*—aprendizado de máquina, classificação, aprendizado supervisionado, *Perceptron multicamada*, redes neuronais**

I. Introdução

A classificação de imagens é um dos grandes problemas estudados pela área da inteligência artificial (IA). O emprego das técnicas de aprendizado de máquina (AP) torna possível processar uma grande quantidade de dados e extrair deles padrões não triviais. Assim, a partir do 'aprendizado' obtido pelo computador, é possível classificar dígitos manuscritos ou até mesmo reconhecer pessoas potencialmente perigosas em grandes multidões.

Dada a grande importância e gama de aplicações do reconhecimento de imagens, diversas técnicas de AP foram criadas, entre elas redes neuronais artificiais (RNAs) e máquinas de vetores suporte, com o intuito de se atingir a maior generalização possível. Um sistema possui boa capacidade de generalização quando é capaz de manter altas taxas de acurácia para dados diferentes daqueles usados no conjunto de treinamento.

No presente trabalho, a classificação de dígitos manualmente escritos presentes na base de dados MNIST [1] foi realizada utilizando um modelo de rede neuronal artificial conhecido como *Perceptron multicamada* e algumas variações com relação à sua configuração padrão.

A. Base de Dados MNIST

A base de dados utilizada consiste em um conjunto de setenta mil imagens de dígitos manuscritos. Tais dígitos foram previamente centralizadas e seus tamanhos normalizados. Do conjunto total, sessenta mil exemplos são para o treinamento de algoritmos e dez mil para teste.

Os dígitos estão igualmente distribuídos em dez classes diferentes (0, 1, 2, ..., 9) e foram coletados por aproximadamente duzentos e cinquenta escritores diferentes.

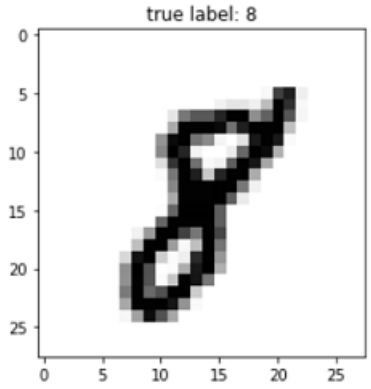

Figure 1. Visualização de um dígito rotulado como 8

As imagens possuem um dimensão de 28x28 pixels, ou seja, um total de 784 pixels por imagem. A Figura 1 é um exemplo de um dígito e sua respectiva classe. Ambas as informações estão contidas na base de dados.

B. Redes Neuronais Artificiais

Uma rede neuronal artificial pode ser vista como um grafo orientado, onde cada aresta é uma conexão ponderada entre 2 nós (neurônios). Os neurônios presentes em RNAs constituem a unidade básica de processamento da rede e são inspirados nos neurônios naturais presentes no cérebro humano.

Pela analogia à fisiologia humana, neurônios artificiais (a partir de agora tratados apenas como neurônios) recebem entradas numéricas e realizam uma soma ponderada s delas com um termo de *bias*. A saída é dada pela aplicação de uma função de ativação (FA) f à s, podendo ser representada como $saida = f(s)$.

A conexão de vários neurônios em uma estrutura de camadas cria uma RNA. A Figura 2 é um exemplo de RNA, com a estrutura de um neurônio em destaque.

A rede chamada Perceptron foi um dos primeiros modelos de RNA. Consistia em uma camada de neurônios que possuíam funções do tipo degrau. Por ser essencialmente uma função linear, não conseguia solucionar a maioria dos problemas reais.

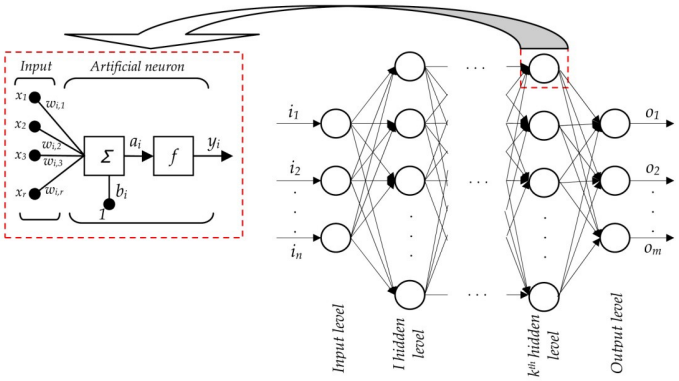

Figure 2. Modelo padrão de uma RNA com destaque para estrutura interna de cada neurônio.

C. Perceptron Multicamada

A rede conhecida por *Perceptron Multicamada* ('*Multilayer Perceptron*' ou 'MLP') é um extensão da arquitetura da rede *Perceptron*, utilizando mais camadas em sua composição e funções de ativação contínuas não lineares. Esse tipo de rede é classificada como '*Feedforward*', em que os dados seguem da camada de entrada até a camada de saída, percorrendo as camadas escondidas. Nesse tipo de rede, não há ciclos entre os neurônios.

A FA padrão do MLP é a função logística (muitas vezes chamada apenas de *sigmoide*). Para o treinamento da rede, o erro quadrático é utilizado com função de custo (FC) em conjunto com o negativo do gradiente desta com relação aos pesos. Assim, sendo $W[n]$ o vetor de pesos associados a um neurônio na iteração atual, $W[n+1]$ o conjunto de pesos da iteração seguinte, ν a taxa de aprendizado e C a FC, pode-se escrever a equação de treinamento padrão de um MLP como:

$$W[n+1] = W[n] - \nu\nabla(C) \tag{1}$$

Modificações do 'MLP' costumam ser feitas com relação à FA de uma ou mais camadas ou com relação à FC. Tangente hiperbólica ou unidade linear retificada ('Relu', em inglês) costumam ser substituições da FA padrão do 'MLP' nas camadas escondidas (todas as camadas que não são de saída). A camada de saída costuma ter como FAs as funções 'Relu' ou 'softmax'. A FC costuma ser substituída pelo erro quadrático médio ou pela entropia cruzada. Esta é utilizada a fim de anular a multiplicação da derivada da função de ativação presente no gradiente da Equação 1, quando FAs do tipo logística são utilizadas. Por fim, o otimizador 'adam' pode ser usado em detrimento do gradiente presente na Equação 1 para obter uma invariância da taxa de atualização dos pesos com relação à escala do gradiente [2].

II. Metodologia

As setenta mil imagens presentes na base MNIST [1] foram divididas em 3 conjuntos: 51 mil para treino ($\pm 72,86\%$) , 9 mil para validação ($\pm 12,86\%$) e 10 mil para teste ($\pm 14,28\%$).

Em seguida, a normalização do conjunto de variáveis de entrada foi realizada. Essa técnica de pré-processamento é de extrema importância para o bom desempenho dos algoritmos de aprendizado de máquina. Nela, as entradas são convertidas em vetores unitários, o que facilita a convergência do gradiente evanescente para um mínimo local/global em uma quantidade menor de épocas de treinamento.

Após a definição das bases de treinamento, validação e teste, além da normalização dos dados, o modelo 'MLP' padrão e 6 variações foram criadas a fim de obter o melhor modelo para a classificação. Cada modelo recebeu um Id para identificação. A Tabela I contém o número de neurônios por camada escondida, bem como a taxa de aprendizado e número de épocas utilizado no treinamento. A Tabela II contém as FAs das camadas escondidas, da camada de saída, bem como o otimizador e FCs utilizadas.

Table I
QUANTIDADE DE NEURÔNIOS POR CAMADA ESCONDIDA, TAXA DE APRENDIZADO E NÚMERO DE ÉPOCAS UTILIZADAS PARA TREINAMENTO.

Id	Camadas	Neurônios	T. Aprendizado	Épocas
1	1	[16]	0.01	20
2	2	[16,8]	0.01	10
3	2	[32,16]	0.001	15
4	2	[64,32]	0.005	20
5	3	[64,32,16]	0.01	10
6	3	[64,32,16]	0.001	15
7	3	[64,32,16]	0.005	20

Table II
FUNÇÕES DE ATIVAÇÃO DAS CAMADAS ESCONDIDAS, FUNÇÃO DE ATIVAÇÃO DA SAÍDA, OTIMIZADOR E FUNÇÃO DE CUSTO

Id	Ativações	Ativação Saída	Otimiz.	Custo
1	*sigmoid*	*sigmoid*	*SGD*	*crossentropy*
2	*sigmoid*	*sigmoid*	*SGD*	*SE*
3	*tanh*	*relu*	*adam*	*MSE*
4	*relu*	*softmax*	*SGD*	*crossentropy*
5	*tanh*	*relu*	*adam*	*MSE*
6	*relu*	*sigmoid*	*SGD*	*SE*
7	*relu*	*softmax*	*adam*	*crossentropy*

Uma vez que não se utilizou grande quantidade de camadas, há um risco menor de que a rede perca capacidade de generalização devido à sua complexidade. Sendo assim, nenhuma regularização (*L1* ou *L2*) nem 'Dropout' foram utilizados.

Os modelos com vinte épocas de treinamento (*Ids 1, 4 e 7*) foram escolhidos para a analisar o impacto da quantidade de épocas na acurácia final da base de validação. O modelo 7 em especial foi também utilizado para verificar a importância da taxa de aprendizado ν com relação ao resultado final. 0, 4, 8, 12, 16 e 20 foram o número de épocas escolhidas. Os valores de taxa de aprendizado ν avaliados foram: 0.00001, 0.0001, 0.005, 0.1, 0.9.

Os neurônios da camada escondida podem ser entendidos como 'ativos' se o somatório ponderado das entradas for positivo e 'inativos' caso contrário. Esse processo pode ser entendido como uma imagem (matriz) de padrão M_p, na

qual pixeis com pesos negativos tem valores 0 e pixeis com pesos positivos tem o valor proporcional ao peso. Além disso, para as redes de 2 camadas, uma imagem será classificada como pertencente à classe c_i caso os neurônios da camada escondida estejam 'ativos' e tiverem pesos positivos. Assim, o somatório de todas as M_p com pesos positivos na camada de saída representam os padrões reconhecidos pelo classificador ao reconhecer uma imagem como pertencente a c_i. O padrão reconhecido para o dígito 3, obtido da forma descrita acima, é mostrado na Figura 3.

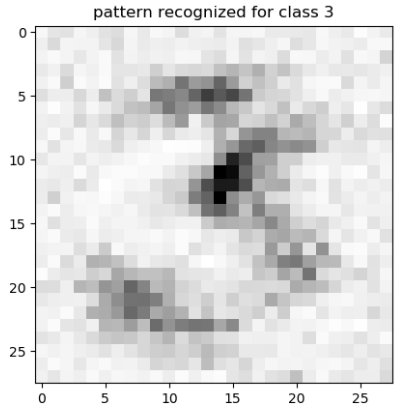

Figure 3. Padrão reconhecido para o número 3 utilizando modelo Perceptron Multicamada como uma camada escondidas e funções logísticas como funções de ativação.

A. *Linguagem de Programação, Bibliotecas e Implementação*

A linguagem de programação Python ™ foi utilizada dada a sua versatilidade e facilidade em se desenvolver programas.

A biblioteca *Sklearn* [6] fornece uma série de funções para o tratamento dos dados, construção de modelos de aprendizado de máquina e avaliação de resultados e, por isso, foi amplamente utilizada.

A biblioteca *Tensorflow* [3] foi utilizada para facilitar a construção de arquiteturas de redes neurais mais flexíveis e expansíveis. Além disso, disponibiliza uma ferramenta de visualização denominada *Tensorboard*, onde é possível analisar visualmente os modelos criados. Em conjunto, a biblioteca *Keras* foi usada para a obtenção de funções de custo, ativação e perda.

Usou-se um repositório no Github [7] para armazenar a implementação do projeto e auxiliar o gerenciamento do controle de versões.

III. Resultados

Os resultados finais exibidos na Tabela III foram obtidos a partir de modelos construídos com bases nos parâmetros presentes nas Tabelas I e II. Apesar da variação dos modelos, a classificação final foi satisfatória em todos eles.

Table III
Resultados obtidos

Id	Treinamento		Teste	
	Acc. Train	Acc. Valid	Acurácia	F1-Score
1	88.18%	90.03%	88.99%	88.83%
2	92.47%	93.74%	92.58%	92.46%
3	88.22%	86.97%	87.11%	83.34%
4	96.36%	96.48%	95.63%	95.59%
5	97.15%	95.30%	95.41%	95.30%
6	98.93%	97.38%	97.13%	97.10%
7	99.45%	97.16%	97.11%	97.08%

Os modelos com vinte épocas de treinamento (*Ids: 1, 4 e 7*), descritos nas Tabelas I e II, foram escolhidos para a análise do impacto do número de épocas utilizadas no treinamento. As Figuras 4 e 5 mostram a evolução das taxas de acurácia e função de custo em função do número de épocas utilizadas para o modelo 7 nas fases de treinamento e validação.

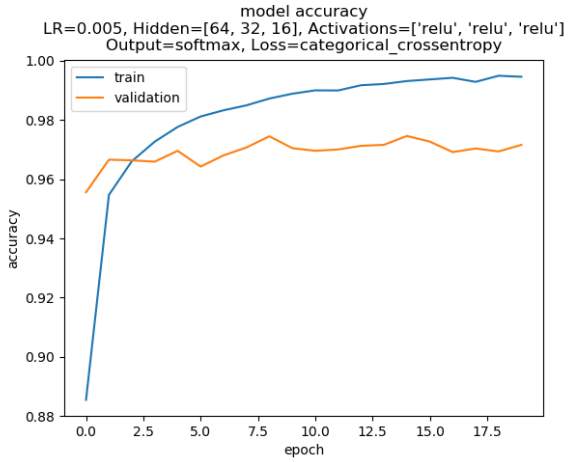

Figure 4. Acurácia x Épocas para o modelo 7

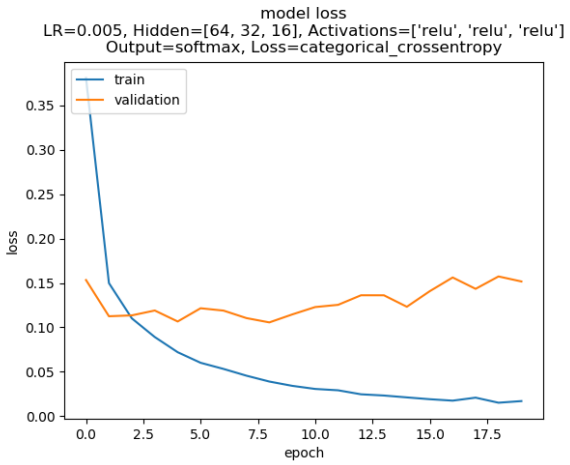

Figure 5. Função de custo x Épocas para o modelo 7

A comparação entre os três modelos é apresentada nas Figuras 6 e 7 as quais apresentam, respectivamente, as acurácias

e funções de custo para os modelos 1, 4 e 7 (em vermelho, laranja e azul respectivamente).

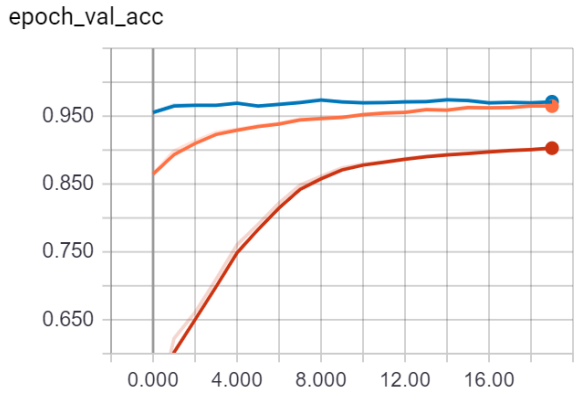

Figure 6. Evolução da Acurácia na etapa de validação. Modelos 1 em vermelho, 4 em laranja e 7 em azul.

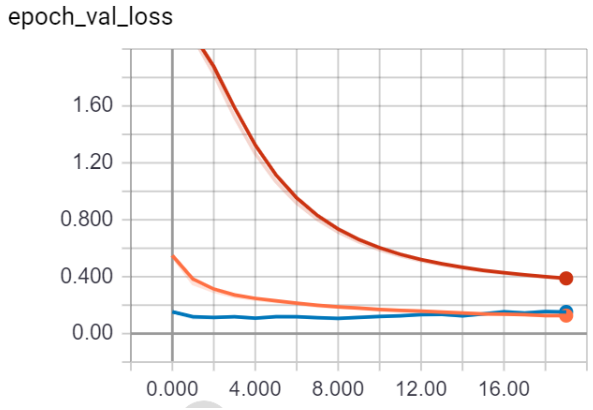

Figure 7. Evolução da Função de custo na etapa de validação. Modelos 1 em vermelho, 4 em laranja e 7 em azul.

O modelo 7 foi escolhido para a comparação de desempenho ao variar a taxa de aprendizado e o número de épocas. A Figura 8 apresenta os resultados de acurácia obtidos.

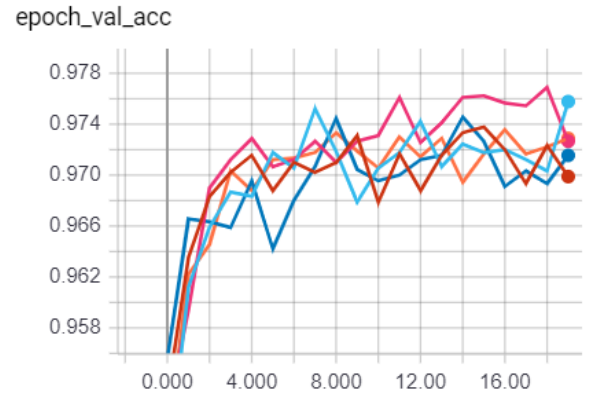

Figure 8. Evolução da acurácia na etapa de validação. Taxas de aprendizado em valores diversos (Rosa: 0.00001, Azul Claro: 0.0001, Azul Escuro: 0.005, Laranja: 0.1, Vermelho: 0.9).

Por fim, Figura 9 apresenta a matriz de confusão obtida a partir do modelo 7. Observa-se uma acurácia elevada em cada uma das classes do problema.

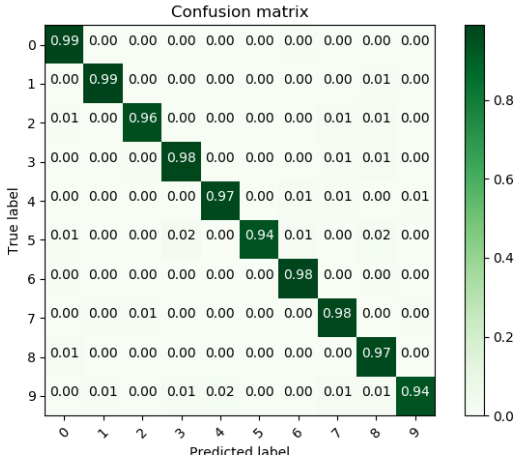

Figure 9. Matriz de Confusão para a configuração de Id 7

IV. Discussão

A partir dos resultados presentes na Tabela III, pode-se considerar que os melhores modelos foram 6 e 7, uma vez que ambos apresentam as maiores acurácias nas fases de validação e teste. Analisando esses resultados, é possível perceber que o modelo 7 pode apresentar uma tendência maior ao sobre-ajuste quando comparado ao modelo 6, uma vez que o primeiro obteve uma acurácia maior no treinamento, porém atingiu resultados levemente menores nas 2 fases seguintes (validação e teste). Considerando que as configurações de ambos são similares, o maior número de épocas pode estar associado à pequena perda da capacidade de generalização.

As Figuras 4 a 7 combinadas com os resultados dos modelos 1 e 2 presentes na Tabela III mostram que, para redes suficientemente complexas (2 ou mais camadas ocultas), a partir de um limiar (aproximadamente 8 épocas para os modelos 1, 4 e 7), aumentar o número de épocas não melhoram significativamente a taxa de acurácia. Esse comportamento pode ser explicado pelo fato de que a rede se torna suficientemente complexa para rapidamente se adaptar ao conjunto de dados inseridos. A partir desse ponto, a quantidade e qualidade dos dados disponíveis no treinamento não é suficiente para permitir melhoras na classificação.

Observando a evolução da acurácia do modelo 7 de acordo com o número de épocas e taxa de aprendizado, pode-se concluir que para $\nu \in [0.00001, 0.9]$ a taxa de aprendizado não causa alteração significativa na acurácia obtida, além de não gerar instabilidade no classificador. A compatibilidade entre complexidade do modelo e variabilidade dos dados pode ser o fator que determina a estabilidade do modelo para a ampla faixa de valores de ν.

Os dígitos 5 e 9 foram os que apresentaram maiores taxas de erro, obtidas pelo modelo 7, de acordo com a matriz de confusão presente na Figura 9. Uma das possibilidades para

esse resultado pode ser devido a rotações sofridas por esses dígitos no momento de sua digitalização, mas maiores análises devem ser realizadas para confirmar essa hipótese.

V. Conclusão

O trabalho de classificar dígitos manualmente escritos utilizando o modelo *Perceptron multicamadas* pode ser considerado bem sucedido, uma vez que, de acordo com a Tabela III, a maioria dos modelos apresentou erros menores que 10% em todas as etapas (treino, teste e validação), tendo o classificador recebido como entrada somente a imagem 'crua', sem nenhum pré-processamento ou extração de características realizado.

Uma análise detalhada da razão pela qual os dígitos 5 e 9 apresentaram erros maiores, como indicado pela Figura 9, é recomendada como possibilidade de pesquisa futura. Além disso, a fim de comparação, máquinas de Boltzmann restritas ('RBM', em inglês) e extração de características adicionais, como Momentos de Zernike, podem ser usadas e possivelmente resultar em um pequeno incremento da acurácia média do melhor classificador. Uma vez que as 2 últimas abordagens podem aumentar a complexidade do classificador gerado, recomenda-se que seja tomado o cuidado de verificar se o modelo não perdeu capacidade de generalização.

Conclui-se que o ajuste da configuração da rede é essencial para obter resultados de classificação satisfatórios. Além disso, um grande número de exemplos também contribui de forma positiva para a identificação dos padrões inerentes às imagens.

References

[1] Y. LeCunn, C. Cortes, C.J.C Burges, "THE MNIST DATABASE of handwritten digits". Link para repositório da base de dados utilizada: http://yann.lecun.com/exdb/mnist/ (Acessado em 17/05/2019)

[2] Diederik P. Kingma, Jimmy Lei Ba, "ADAM: A METHOD FOR STOCHASTIC OPTIMIZATION", 2015

[3] Link para página da biblioteca 'Tensorflow', https://www.tensorflow.org/ (Acessado em 17/05/2019)

[4] Haykin, Simon, 'Neural Networks and Learning Machines', 2009

[5] Link para página da biblioteca 'Keras': https://keras.io/ (Acessado em 17/05/2019)

[6] Link para página da biblioteca "Sklearn": https://scikit-learn.org/stable/index.html (Acessado em 17/05/2019)

[7] Link para repositório do projeto: https://github.com/mso13/trabalho2-mnist (Acessado em 17/05/2019)