Treinamento distribuído de redes MLP para classificação de figuras geométricas

Lucca Gamballi, Daniel Gilio Tiglea, Renato Candido e Magno T. M. Silva

***Resumo*— Redes neurais perceptron multicamada são utilizadas para classificar figuras geométricas com uma abordagem distribuída. Os dados de treinamento são divididos entre redes que se comunicam por meio de uma determinada topologia, sem que haja compartilhamento direto desses dados entre as redes. Resultados de simulação indicam que o desempenho obtido com o treinamento distribuído e uma topologia adequada é semelhante ao observado com o treinamento clássico. Assim, garante-se a privacidade dos dados sem que ocorra perda no desempenho.**

***Palavras-Chave*— Redes neurais artificiais, perceptron multicamada, processamento distribuído, topologia, figuras geométricas.**

***Abstract*— Multilayer perceptron neural networks are used to classify geometric figures using a distributed approach. Training data are divided among neural networks that communicate through a certain topology, in which each network does not have access to the training data of the others. Simulation results indicate that the performance achieved with distributed training and a suitable topology is similar to that observed with classical training. Thus, data privacy is guaranteed without loss of performance.**

***Keywords*— Artificial neural networks, multi-layer perceptron, distributed processing, topology, geometric figures.**

I. Introdução

Na aprendizagem supervisionada, um modelo é treinado com dados de entrada que contêm rótulos conhecidos [1]. Usualmente, o processo de treinamento ocorre em apenas uma máquina. Contudo, essa abordagem centralizada pode não ser adequada em problemas de larga escala ou quando a privacidade é objeto de preocupação devido à presença de informações sensíveis. Nesses casos, o armazenamento e/ou processamento de todos os dados em uma única máquina pode ser problemático. Exemplos incluem aplicações médicas ou as que lidam com hábitos de vida dos usuários. Uma possível solução consiste em considerar uma abordagem distribuída, em que cada máquina treina um modelo local utilizando apenas uma parte dos dados [2]. Em uma segunda etapa, elas trocam informações acerca de seus modelos por meio de uma topologia pré-definida, sem compartilhar os dados de treinamento. Assim, pode-se obter um único modelo global utilizando todos os dados de treinamento sem que uma única máquina tenha acesso a todos eles. Neste trabalho, será considerado o treinamento distribuído de redes perceptron multicamada (*multilayer perceptron* – MLP) com base nos algoritmos *backpropagation* [3] e de consenso de [4]. Simulações mostram que, adotando-se uma topologia adequada, é possível preservar o desempenho da abordagem centralizada, conciliando com as vantagens do treinamento distribuído.

Lucca Gamballi, Daniel G. Tiglea, Renato Candido e Magno T. M. Silva, Depto. de Engenharia de Sistemas Eletrônicos, Escola Politécnica, Universidade de São Paulo, São Paulo, SP, Brasil, e-mails: lucca.gamballi@usp.br, {dtiglea, renatocan}@lps.usp.br e magno.silva@usp.br. Este trabalho foi financiado pelo CNPq (139071/2021-0), CAPES (88887.512247/2020-00 e Código de Financiamento 001) e FAPESP (2021/02063-6).

II. Treinamento distribuído de redes MLP

No treinamento distribuído de redes MLP, os N exemplos de treinamento são divididos entre $\mathcal{V}$ máquinas que se comunicam segundo uma determinada topologia. Cada máquina $i \in \{1, 2, \ldots, \mathcal{V}\}$ treina uma MLP com o conjunto de dados $\mathcal{D}_i$ de cardinalidade N_i, tal que $\sum_{i=1}^{\mathcal{V}} N_i = N$. Considerando um problema de classificação com C classes, é comum adotar a entropia cruzada categórica ponderada como função custo. Assim, a MLP da i-ésima máquina minimiza [5]

$$J_{\mathrm{ECCP}_i} = -\sum_{n_i=1}^{N_i}\sum_{\ell=1}^{C} p_{i,\ell}\; d_{i,\ell}(n_i)\;\ln[y_{i,\ell}(n_i)],$$

em que $y_{i,\ell}(n_i)$ é a ℓ-ésima saída da rede para o n_i-ésimo dado de treinamento com rótulo $d_{i,\ell}(n_i)$, $\ell = 1, 2, \ldots, C$ e $n_i = 1, 2, \ldots, N_i$. Classes minoritárias devem ser ponderadas com pesos $p_{i,\ell}$ maiores, o que é adequado para lidar com conjuntos de dados desbalanceados [5]. Inspirando-se na heurística de [6], considerou-se neste trabalho o peso $p_{i,\ell} = \lceil 10N_i/(CB_{i,\ell})\rceil$, em que $B_{i,\ell}$ é o número de amostras da classe ℓ em $\mathcal{D}_i$.

Utilizando-se o algoritmo de retropropagação do erro (*backpropagation*) para o treinamento das redes MLP [3], [7] e uma única iteração do algoritmo de consenso para a comunicação entre as máquinas a cada iteração m do *backpropagation*, obtém-se a seguinte regra de atualização da matriz de pesos da k-ésima camada da MLP da máquina i (os *biases* estão incluídos na matriz):

$$\mathbf{W}_i^{[k]}(m+1) = \sum_{j=1}^{\mathcal{V}} a_{ij}\left(\mathbf{W}_j^{[k]}(m) - \eta\frac{\partial J_{\mathrm{ECCP}_j}}{\partial \mathbf{W}_j^{[k]}(m)}\right),$$

em que η é o passo de adaptação e $a_{ij} \geq 0$ são pesos de combinação, tal que $a_{ij} = 0$ se as máquinas i e j não estão diretamente conectadas e $\sum_{j=1}^{\mathcal{V}} a_{ij} = 1$ para todo i, j. Existem diversas regras possíveis para a escolha desses pesos. Neste trabalho, adota-se a regra Metropolis [4], [8]. O caso particular em que $a_{ij} = 1$ se $i = j$ e $a_{ij} = 0$ caso contrário corresponde a um cenário de treinamento individual sem comunicação entre as máquinas. O fato da comunicação entre as máquinas ocorrer apenas quando os pesos da rede são atualizados é fundamental para se ter um custo computacional reduzido.

III. Banco de dados

O conjunto de dados utilizado neste artigo é constituído de imagens em preto e branco com dimensões 30×30 criadas aleatoriamente com auxílio da biblioteca PILLOW [9]. Cada imagem apresenta uma de três possíveis figuras geométricas: quadrados (Q), retângulos (R) ou circunferências (C).

A espessura do traço pode ser fina ou grossa, determinada aleatoriamente. Foram considerados dois cenários: um com classes balanceadas e outro com classes desbalanceadas, como mostrado na Tabela I.

TABELA I: Número de amostras para cada figura.

		Quadrado	Retângulo	Circunferência
Balanceado	Treino	5000	5000	5000
	Teste	5000	5000	5000
Desbalanceado	Treino	5940	689	2819
	Teste	6347	1390	2371

IV. Resultados de simulação

Foram considerados três tipos de treinamento: o centralizado, o individual e o distribuído. Nestes dois últimos, foram consideradas as topologias ilustradas na Fig. 1 [4]. Cada nó do grafo representa uma máquina onde é treinada uma MLP. As redes MLP foram implementadas usando a biblioteca Tensorflow [10] e são compostas por duas camadas ocultas com 32 e 16 neurônios cada e função de ativação ReLU. A camada de saída é composta de três neurônios com a função *Softmax*. Por fim, foi considerado o algoritmo de otimização Adam [11] com parâmetros $\beta_1 = 0,9$, $\beta_2 = 0,99$ e $\epsilon = 10^{-7}$. Os pesos foram inicializados de acordo com a inicialização Glorot [12] e foi considerado o *backpropagation* com passo de adaptação $\eta = 0,001$, tamanho de mini-batch $k = 2048$ e 1600 épocas de treinamento.

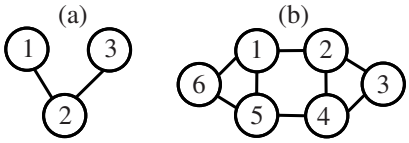

Fig. 1: Topologias com (a) 3 e (b) 6 máquinas.

Nas Tabelas II e III, são mostrados a sensibilidade (Se), precisão (P), F1-*Score* (F1) e coeficiente de correlação de Matthews (MCC) alcançados com cada abordagem para os dados de teste no cenário balanceado e desbalanceado, respectivamente. Em cada linha das tabelas, destaca-se em negrito a maior métrica alcançada, sendo que nos casos distribuído e individual, considerou-se a média das métricas das máquinas.

Analisando-se a Tabela II, nota-se que o treinamento centralizado apresenta as melhores métricas para todas as classes. O treinamento distribuído com a topologia da Fig. 1(a) apresenta métricas comparáveis às do caso centralizado para a classe C e um pouco inferiores para as classes Q e R. Comparando com o caso individual, o treinamento distribuído alcançou resultados superiores para a maior parte das métricas (22 de 24 métricas) considerando a mesma topologia. Por fim, os resultados obtidos com o treinamento distribuído para a topologia da Fig. 1(a) foram superiores aos da Fig. 1(b) para as classes Q e R e próximos para a classe C.

Para o caso desbalanceado (Tabela III) e topologia da Fig. 1(a), o treinamento individual forneceu os melhores resultados, com exceção das métricas da classe C e a precisão da classe R. Nota-se também neste caso que o treinamento distribuído com a topologia da Fig. 1(a) superou as métricas do centralizado, com exceção da sensibilidade para a classe Q e precisão para a classe R. O mesmo não é válido para a topologia da Fig. 1(b). Por fim, observa-se que o treinamento individual para a topologia da Fig. 1(b) obteve resultados piores que os da topologia da Fig. 1(a).

TABELA II: Métricas de desempenho para dados balanceados.

		Central	Individual		Distribuído	
			Topologia (a)	Topologia (b)	Topologia (a)	Topologia (b)
Q	Se(%)	**96,02**	89,40	81,85	88,34	82,45
	P(%)	99,36	89,30	90,81	**99,48**	95,31
	F1(%)	**97,66**	89,12	86,05	93,58	88,37
	MCC	**0,96**	0,84	0,79	0,90	0,83
R	Se(%)	99,34	90,44	89,26	**99,41**	94,56
	P(%)	**95,78**	89,90	79,69	86,82	79,74
	F1(%)	**97,3**	89,97	84,14	92,68	86,35
	MCC	**0,96**	0,85	0,77	0,90	0,81
C	Se(%)	99,90	99,92	99,52	99,94	**99,98**
	P(%)	**100,0**	99,96	99,12	**100,0**	**100,0**
	F1(%)	99,95	99,94	99,31	99,97	**99,99**
	MCC	**1,00**	**1,00**	0,99	**1,00**	**1,00**

TABELA III: Métricas de desempenho para dados desbalanceados.

		Central	Individual		Distribuído	
			Topologia (a)	Topologia (b)	Topologia (a)	Topologia (b)
Q	Se(%)	**93,77**	93,54	89,03	93,27	90,45
	P(%)	91,57	**99,80**	98,69	93,48	74,13
	F1(%)	92,66	**96,53**	93,60	93,36	77,39
	MCC	0,80	**0,90**	0,82	0,82	0,63
R	Se(%)	65,06	**98,02**	75,15	71,25	57,64
	P(%)	**71,65**	67,52	43,93	68,75	60,65
	F1(%)	68,20	**78,64**	57,72	69,63	47,27
	MCC	0,63	**0,78**	0,58	0,65	0,44
C	Se(%)	99,63	99,26	98,83	99,72	**99,95**
	P(%)	**100,0**	99,68	98,84	**100,0**	99,42
	F1(%)	99,81	99,47	98,84	**99,86**	99,69
	MCC	**1,00**	0,99	0,99	**1,00**	**1,00**

Apesar da abordagem distribuída apresentar um custo computacional total maior que o da centralizada devido ao algoritmo de consenso, esse custo é distribuído entre as máquinas. Assim, os requisitos quanto à capacidade computacional dos processadores das máquinas são menores no caso distribuído.

V. Conclusões

O treinamento distribuído de redes neurais é relativamente recente e tem grande potencial, pois visa resolver problemas de privacidade. Neste trabalho, constatou-se que o treinamento distribuído pode alcançar um desempenho comparável ao do caso centralizado desde que se considere uma topologia adequada. Deve-se tomar cuidado com topologias que contêm muitas máquinas, pois isso pode resultar em conjuntos de treinamento pequenos para cada rede neural, o que pode prejudicar o desempenho. O desempenho superior do treinamento individual em relação ao distribuído e ao centralizado, observado em alguns casos, será investigado posteriormente.

Referências

[1] M. I. Jordan and T. M. Mitchell, "Machine learning: Trends, perspectives, and prospects," *Science*, vol. 349, pp. 255–260, 2015.

[2] M. Fahimi and A. Ghasemi, "A distributed learning automata scheme for spectrum management in self-organized cognitive radio network," *IEEE Transactions on Mobile Computing*, vol. 16, pp. 1490–1501, 2016.

[3] S. Haykin, *Neural netw. and learning machines*, Pearson, 3rd ed., 2009.

[4] B. Liu, Z. Ding, and C. Lv, "Distributed training for multi-layer neural networks by consensus," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, pp. 1771–1778, 2019.

[5] Y. Ho and S. Wookey, "The real-world-weight cross-entropy loss function: Modeling the costs of mislabeling," *IEEE Access*, vol. 8, pp. 4806–4813, 2020.

[6] G. King and L. Zeng, "Logistic regression in rare events data," *Political Analysis*, vol. 9, pp. 137–163, 2001.

[7] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*, MIT Press, 2016.

[8] V. Schwarz, G. Hannak, and G. Matz, "On the convergence of average consensus with generalized metropolis-hasting weights," in *Proc. ICASSP*, 2014, pp. 5442–5446.

[9] A. Clark *et al.*, "Pillow (PIL fork) documentation," Disponível em `https://pillow.readthedocs.io/`, 2015.

[10] M. Abadi *et al.*, "Tensorflow: Large-scale machine learning on heterogeneous distributed systems," *arXiv preprint arXiv:1603.04467*, 2016.

[11] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[12] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proc. ICAIS*, PMLR, vol. 9, pp.249-256, 2010.