Análise de requisitos computacionais de técnicas de aprendizado de máquina para implementação em hardwares de baixo custo

José Edson Moreno Júnior
Universidade Federal do ABC
Santo André, SP
Email:edson.moreno@ufabc.edu.br

Murilo Bellezoni Loiola
Universidade Federal do ABC
Santo André, SP
Email:murilo.loiola@ufabc.edu.br

***Resumo*—Com o aumento da popularidade dos hardwares de baixo custo, como os dispositivos embarcados, e o crescente uso dos algoritmos de aprendizagem de máquina, há uma demanda para que estes trabalhem nas mais diversas áreas, pois nem sempre há a possibilidade de uma comunicação dos dispositivos com os hardwares mais robustos onde é executado normalmente o algoritmo. Portanto, este trabalho vem propor uma avaliação dos algoritmos de regressão linear polinomial e rede neural Perceptron multicamada, efetuando uma análise de requisitos computacionais para uso em hardwares de baixo custo.**

***Keywords*—regressão linear, rede neural, hardware de baixo custo**

I. Introdução

O progresso recente no aprendizado de máquina tem sido impulsionado pela alta disponibilidade de dados e pela computação de baixo custo [1], hardwares com microprocessadores e memória no qual o custo final não ultrapassa US$ 50,00 [2], [3]. Juntamente com este avanço, bilhões de microcontroladores têm sido utilizados com diversos propósitos nas mais diversas áreas, principalmente integrando hardwares para a chamada Internet das Coisas (Internet of Things - IoT).

Contudo, o uso mais comum para os dispositivos IoT, devido às restrições de memória, energia e processamento, é como coletores de dados, como [4], [5], [6], [7], sendo que todo o processamento e tomada de decisão ocorre fora dos dispositivos [8], [9]. Porém, nem sempre é possível transmitir os dados coletados para processamento, pois não há conectividade, há limitações de energia que dificultam a transmissão, problemas relacionados à latência e à privacidade dos dados, tudo isto com a necessidade ou urgência deste processamento dos dados coletados. Diante deste cenário há justificativa que estes dispositivos processem os próprios dados.

Há vários trabalhos, como [10], [1], [11], que tratam do assunto IoT e aprendizado de máquinas, apresentando o aumento do uso destes tipos de dispositivos e, com eles, o aumento da massa de dados e a necessidade de analisá-los. Contudo, poucos são os trabalhos relativos à implementação de algoritmos de aprendizagem de máquina nesses dispositivos. Os existentes, como em [12] são simulações, ou são para necessidades específicas, como o descrito por [9].

Assim este trabalho vem propor uma análise das necessidades mínimas de hardware do tipo IoT para o uso dos algoritmos de regressão linear polinomial e rede neural Perceptron multicamada, determinando o que é preciso para execução dos algoritmos em termos de hardware, como memória *heap* e *stack pointer* e processamento.

O algoritmo de regressão linear polinomial é utilizado para prever valores e comportamentos a partir da análise estatística de dados. É uma técnica que envolve normalmente um conjunto de dados ordenados e que gera, ao final, os coeficientes de uma equação polinomial e seus erros estatísticos a partir de cálculos matriciais. Normalmente, o treinamento ou geração dos coeficientes do polinômio é feito em uma máquina com um poder de processamento maior, para a obtenção dos valores de erro e posteriormente, é feita a inserção da função obtida pelo algoritmo no código implementado no hardware de baixo custo [4], [13]. Já quando a solução do algoritmo é calculada diretamente no hardware, a sua utilização é para uma solução de regressão linear simples [14], [15].

Já a rede neural Perceptron multicamada, pode ser utilizada para classificação [16], reconhecimento de imagens [6], [17] ou sons [18], dentre outros. Como na regressão linear, o seu treinamento normalmente ocorre em equipamento de maior processamento, no qual se obtêm os pesos sinápticos da rede neural, o qual posteriormente é inserido junto com código de programação, no hardware de baixo custo.

Dessa forma, na Seção II serão apresentadas as técnicas analisadas neste trabalho, enquanto na seção III, será descrita a metodologia utilizada e apresentados alguns resultados. Por fim, as conclusões do trabalho são mostradas na Seção IV.

II. Algoritmos

A. Regressão Linear Polinomial

A regressão linear polinomial é uma técnica estatística de modelagem preditiva, que tem como objetivo verificar se existe uma relação entre uma ou mais variáveis independentes, chamadas de variáveis preditoras, com uma única variável dependente contínua, a variável resposta [19].

O modelo de regressão linear tenta obter uma equação de como se comportam os valores da variável dependente y em

função das variáveis independentes $x_1, x_2, ..., x_n$ [20], [19], [21].

A regressão linear polinomial pode ser expressa pelo pseudo código 1, o qual β é o vetor solução com os coeficientes do polinômio, SSE é a soma dos quadrados dos erros da solução e var a variância da solução.

pseudo código 1 Regressão Linear Polinomial

Require: X vetor de variáveis independentes, Y vetor de variáveis dependente, np grau do polinômio

1: $\beta \leftarrow \left(X^T X\right)^{-1} X^T Y$
2: $b \leftarrow \beta[0]$É
3: **for** $lvl = 1$ **to** np **do**
4: $\quad b \leftarrow b + \beta[lvl] * (X^{lvl})$
5: **end for**
6: $SSE \leftarrow \sum((Y - b)^2)$
7: $var \leftarrow (SSE/(tamanho_vetor(X) - np - 1)^{0.5}$
8: **return** $[SSE, var, \beta]$

B. Rede Neural Perceptron Multicamada com Backpropagation

O Perceptron é um modelo matemático que representa um neurônio com múltiplas entradas ponderadas e uma saída e que tem como resultado um valor binário oriundo de um mapeamento não-linear. O Perceptron é representado graficamente na Figura 1 [22], [23].

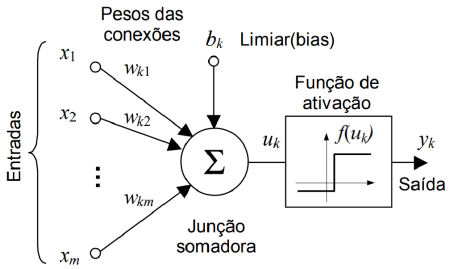

Figura 1. Representação simples de um neurônio Perceptron.

Uma rede neural *Multi Layer Perceptron* (MLP) é formada por Perceptrons interligados, onde a saída de um neurônio é ligada à entrada de outro, formando camadas como pode ser visualizado na Figura 2. Para que a MLP "aprenda"uma determinada tarefa, ela precisa ser treinada. Isto pode ser feito usando técnicas de aprendizado supervisionado, onde um conjunto de padrões entrada / saída é apresentado à rede para adaptação dos pesos dos neurônios [22], [23].

Um dos algoritmos mais conhecidos para o treinamento das redes MLP é o *backpropagation* [24]. No MLP com *backpropagation*, todos os pesos da rede são inicializados com pequenos valores aleatórios. Após a inicialização, são fornecidos os dados de entrada à rede neural, a qual calcula o valor de saída. Com este valor, é feito o cálculo da função de erro, obtido pela diferença entre os valores de saída esperado e o obtido [23]. O erro, então, é retro propagado e usado na

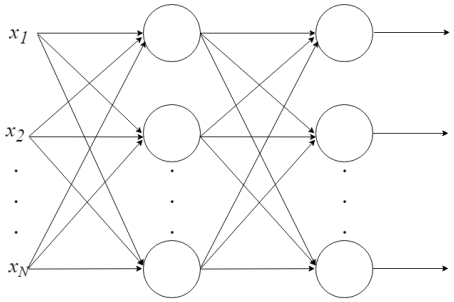

Figura 2. Perceptrons combinados formando uma MLP. Cada círculo representa um neurônio.

atualização dos pesos de cada neurônio. O erro é minimizado de forma que os ajustes ocorram gradualmente, possibilitando que seja encontrado um conjunto adequado de pesos nas camadas da rede MLP, e assim, evitando uma convergência prematura do resultado. Esse processo é descrito no pseudo código 2 [24].

pseudo código 2 Rede Neural MLP

Require: $(\vec{x}, \vec{d}) \in D$, $\vec{x}$ dados de treinamento e $\vec{d}$ valor esperado no conjunto de treinamento D

1: inicializa rede neural
2: **while** Não converge **do**
3: $\quad$ **for** Todo $(\vec{x}, \vec{d}) \in D$ **do**
4: $\quad\quad$ Submeter $\vec{x}$ à entrada da rede neural e calcular sua saída $\vec{y}$
5: $\quad\quad$ Calcular o erro $\xi(\vec{w}) = \frac{1}{2} \sum_{j=1}^{m} (y_j - d_j)^2$
6: $\quad\quad$ **for** cada neurônio, do último ao primeiro na rede **do**
7: $\quad\quad\quad$ Calcular $\frac{\partial \xi(\vec{x})}{\partial w_{ij}}$ para todos os pesos
8: $\quad\quad\quad$ Proceder correção parcial para todos os pesos
9: $\quad\quad$ **end for**
10: $\quad$ **end for**
11: **end while**

III. Método de análise

Toda a análise foi efetuada em código escrito com a linguagem C baseado no pseudo código de cada algoritmo. Optou-se pela análise dos algoritmos nesta linguagem por ser a de uso mais comum nas implementações em hardwares de baixo custo. Primeiramente, foram identificadas as variáveis do código e analisados seus usos durante a execução do programa e com isto, calculado o espaço de memória necessário para que cada algoritmo possa ser executado. A análise baseou-se no uso de memória necessária para a execução dos algoritmos, e não levou em consideração outros possíveis dados carregados conjuntamente.

Para o algoritmo de regressão linear polinomial, a alocação de memória pode ser expressa pela equação 1

$$m = 8+9*s+s*(op+1)^2+2*s(1+n+op+n*(op+1)+(op+1)^2) \quad (1)$$

onde

- m - máximo de memória necessária para execução do algoritmo;
- s - tamanho da variável utilizada no cálculo da regressão (*integer*: 2 ou 4 *bytes*, *float*: 4 *bytes*, *double*: 8 *bytes*) ;
- op - ordem do polinômio usado na regressão linear;
- n - número de dados de treinamento utilizados no cálculo da regressão;

Ainda para a regressão linear polinomial, verificou-se que o uso de memória *heap* é dado por $2 * s * n * (op + 1)$ *bytes* e o máximo de memória *stack pointer* por $8+9*s+s*(op+1)^2+2*s*(1+n+op+(op+1)^2)$ *bytes*. Este valor foi obtido evitando o uso de recursividade nos cálculos de matrizes e determinantes, que faz parte da solução da regressão linear [25]. Para isso foi adotada a solução algébrica baseada na decomposição LU, pois em sua implementação são utilizados basicamente *loops* encadeados, permitindo um controle da quantia de variáveis usadas.

A tabela III verifica-se o valor calculado de memória para o número máximo de amostra que o algoritmo de regressão linear polinomial implementado em um STM32F103 conseguiu executar durante seu treinamento. Com os valores obtidos na execução, de números de amostras, ordem de polinômio e o tipo de variável utilizado, foi calculada a memória necessária, o qual apresentou uma variação entre o valor máximo e mínimo de 5,1%. Já a tabela III apresenta os valores calculados para um STM32F407, com uma variação da memória necessária de 0,7%.

num. amostras	Ordem polinômio	tipo variável	mem. necessária	mem *heap*	mem. *stack pointer*
529	3	float	21428	16928	4500
432	4	float	21120	17280	3840
359	5	float	20628	17232	3396
307	6	float	20336	17192	3144

Tabela I
LIMITES NO QUAL O STM32F103 CONSEGUIU EFETUAR A REGRESSÃO LINEAR, MEMÓRIA (MEM.) EM *bytes*

num. amostras	Ordem polinômio	tipo variável	mem. necessária	mem *heap*	mem. *stack pointer*
3990	3	float	159868	127680	32188
3189	4	float	153456	127560	25896
2643	5	float	148532	126864	21668
2275	6	float	146288	127400	18888

Tabela II
LIMITES NO QUAL O STM32F407 CONSEGUIU EFETUAR A REGRESSÃO LINEAR, MEMÓRIA (MEM.) EM *bytes*

Para o algoritmo de rede neural MLP, a alocação de memória, em *bytes*, é dada pela equação 2

$$m = 527 + 16 * c + n * ns * sp + n * ne * sp + 24 * nn + 2 * nn * ne * sp + \sum_{i=1}^{c} \left(24 * nc_i + 2 * sp * nc_i * nc_{(i-1)}\right) \quad (2)$$

onde

- m - memória necessária para execução do algoritmo;;
- c - número de camadas ocultas;
- ne - número de entradas por neurônio;
- nn - número de neurônios na camada de entrada ($nn <= ne$);
- ns - número de neurônios na camada de saída ($ns > 0$);
- nc_i - número de neurônios na camada oculta i;
- sp - tamanho da variável peso (*integer, float, double*);
- n - número de tupla utilizadas para treinamento;

Têm-se ainda que $24 * nn + 2 * nn * ne * sp + \sum_{i=1}^{c} \left(24 * nc_i + 2 * sp * nc_i * nc_{(i-1)}\right)$ é o espaço máximo de memória *stack pointer*, em *bytes*, e $527 + 16 * c + n * ns * sp + n * ne * sp$ é o espaço alocado pela memória *heap*, em *bytes*.

Na tabela III, têm-se os valores calculados de memória para a rede MLP no limite da execução do programa para os dispositivos STM32F407 e STM32F103 durante o treinamento. Verificou-se que o cálculo de uso de memória necessária, apresenta uma variação de 7,9% para o STM32F407 e 13,9% para o STM32F103, entre os valores mínimos e máximo.

	sp	n	ne	ns	nn	cc	nc_i $i=1$	nc_i $i=2$	nc_i $i=3$	mem. necessária	mem. *heap*	mem. *stack pointer*
STM32F407	float	70	100	10	100	3	12	11	10	114567	31375	83192
	float	79	100	5	100	2	12	5	0	116547	33739	82808
	double	70	63	10	63	3	12	11	10	107263	41455	65808
	double	79	65	5	65	2	12	5	0	114367	44799	69568
STM32F103	float	40	26	10	26	3	12	11	10	13159	6335	6824
	float	40	30	5	30	2	12	5	0	14487	8328	6159
	double	40	14	10	14	3	12	11	10	12471	4216	8255
	double	40	17	5	17	2	12	5	0	13039	5440	7599

Tabela III
LIMITES NOS QUAIS OS STM32F407 E STMF103 CONSEGUIRAM EXECUTAR A REDE MLP, MEMÓRIA (MEM.) EM *bytes*

IV. CONCLUSÃO

Este trabalho teve como objetivo avaliar o potencial de implementação de regressão linear e redes MLP em dispositivos embarcados. Os resultados mostraram que é funcional a equação obtida para determinar a quantia mínima de memória para cada algoritmo. Essas equações não fornecem um valor exato, pois há outras variáveis que são relativas ao funcionamento do hardware de baixo custo, mas permitem ao desenvolvedor uma tomada de decisão antecipada, evitando uma implementação desnecessária.

Na análise do uso de tipos de variáveis possíveis de se utilizar com os algoritmos, verificou-se que há flexibilidade de escolha de tipo e isso permite uma liberdade ao desenvolvedor em sua escolha. Ficou evidente que o tipo de variável (inteiro, *float* ou outro) determina o quanto de memória será usado pelos algoritmos e, no caso dos hardwares analisados, é importante saber o impacto que o uso desta variável terá, mesmo que seja aproximado. Em trabalhos futuros, pretende-se ampliar o estudo para outros hardwares e algoritmos de aprendizado de máquina e assim conseguir saber o real potencial de utilização.

REFERÊNCIAS

[1] M. I. Jordan and T. M. Mitchell, "Machine learning: Trends, perspectives, and prospects," *Science*, vol. 349, no. 6245, pp. 255–260, 2015.

[2] P. Reguera, D. García, M. Domínguez, M. A. Prada, and S. Alonso, "A Low-cost Open Source Hardware in Control Education. Case Study: Arduino-Feedback Ms-150," *IFAC-PapersOnLine*, vol. 48, no. 29, pp. 117–122, 2015.

[3] E. G. da Silva and A. L. F. Perez, "Aplicação De Hardware De Baixo Custo Na Automação Residencial," *Revista Técnico Científica do IFSC*, vol. 1, no. 5, pp. 171–180, 2013. [Online]. Available: http://periodicos.ifsc.edu.br/index.php/rtc/article/view/1177

[4] S. Singh, S. L. Arun, and M. P. Selvan, "Regression based approach for measurement of current in single-phase smart energy meter," *2017 IEEE Region 10 Symposium (TENSYMP)*, no. 1, pp. 1–5, 2017. [Online]. Available: http://ieeexplore.ieee.org/document/8070103/

[5] V. J. Kartsch, S. Benatti, P. D. Schiavone, D. Rossi, and L. Benini, "A sensor fusion approach for drowsiness detection in wearable ultra-low-power systems," *Information Fusion*, vol. 43, no. March 2018, pp. 66–76, 2018. [Online]. Available: https://doi.org/10.1016/j.inffus.2017.11.005

[6] F. Rundo, S. Conoci, S. Petralia, G. L. Banna, and F. Rundo, "Advanced Bio-inspired Point of Care for Skin Cancer Early Detection," *Scientific Literature*, vol. 1, no. 1, pp. 1–6, 2017.

[7] S. Zhang, T. Zhang, Y. Yin, and W. Xiao, "Alumina Concentration Detection Based on the Kernel Extreme Learning Machine," *Sensors*, vol. 17, no. 12, p. 2002, 2017. [Online]. Available: http://www.mdpi.com/1424-8220/17/9/2002

[8] A. Kumar, S. Goyal, and M. Varma, "Resource-efficient Machine Learning in 2 {KB} {RAM} for the Internet of Things," *Proceedings of the 34th International Conference on Machine Learning*, vol. 70, pp. 1935–1944, 2017.

[9] J. Roach, "AI's big leap to tiny devices opens world of possibilities," 2017. [Online]. Available: https://blogs.microsoft.com/ai/ais-big-leap-tiny-devices-opens-world-possibilities/

[10] S. Earley, "Analytics, machine learning, and the internet of things," *IT Professional*, vol. 17, no. 1, pp. 10–13, 2015.

[11] M. Shafique, T. Theocharides, C.-S. Bouganis, M. A. Hanif, F. Khalid, R. Hafiz, and S. Rehman, "An overview of next-generation architectures for machine learning: Roadmap, opportunities and challenges in the IoT era," *2018 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, no. March, pp. 827–832, 2018.

[12] F. Alam, R. Mehmood, I. Katib, and A. Albeshri, "Analysis of Eight Data Mining Algorithms for Smarter Internet of Things (IoT)," *Procedia Computer Science*, vol. 58, no. December, pp. 437–442, 2016. [Online]. Available: http://dx.doi.org/10.1016/j.procs.2016.09.068

[13] A. Agresta, S. D. Vito, F. Formisano, E. Massera, E. Esposito, M. Salvato, G. Fattoruso, and G. D. Francia, "Cooperative Air Quality Sensing with Crowdfunded Mobile Chemical Multisensor Devices," *Proceedings*, vol. 1, no. 4, p. 602, 2017. [Online]. Available: http://www.mdpi.com/2504-3900/1/4/602

[14] A. Fourney and M. Terry, "Picl," *Proceedings of the 25th annual ACM symposium on User interface software and technology - UIST '12*, p. 569, 2012. [Online]. Available: http://dl.acm.org/citation.cfm?id=2380116.2380188

[15] B. Li, L. Fu, W. Zhang, W. Feng, and L. Chen, "Portable paper-based device for quantitative colorimetric assays relying on light reflectance principle," *Electrophoresis*, vol. 35, no. 8, pp. 1152–1159, 2014.

[16] G. Michael, N. Efstathiou, K. Mantis, T. Theocharides, and D. Pau, "Intelligent embedded and real-time ANN-based motor control for multi-rotor unmanned aircraft systems," *IEEE/IFIP International Conference on VLSI and System-on-Chip, VLSI-SoC*, 2017.

[17] F. Rundo, S. Conoci, G. L. Banna, F. Stanco, and S. Battiato, "Bio-Inspired Feed-Forward System for Skin Lesion Analysis, Screening and Follow-up," in *ICIAP 2017: 19th International Conference, Catania, Italy*, no. September, 2017. [Online]. Available: http://link.springer.com/10.1007/978-3-319-68560-1

[18] A. A. Bashit and D. Valles, "A Solar Powered Raspberry Pi Houston Toad Call Detection System using Neural Network Model," *2018 International Conference on Computational Science and Computational Intelligence*, no. April, 2018.

[19] M. A. Matos, "Manual Operacional para a Regressão Linear," *Faculdade de Engenharia da Universidade do Porto*, pp. 1–26, 1995.

[20] P.-N. Tan, M. Steinbach, and V. Kumar, *Introdução ao DATAMINING Mineração de Dados*. Rio de Janeiro: Editora Ciência Moderna Ltda, 2009.

[21] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning*, 2001, vol. 27, no. 2.

[22] S. Haykin, "Redes Neurais - Princípios e Prática," p. 902, 2001.

[23] ——, *Neural Networks and Learning Machines Third Edition*, 2014. [Online]. Available: https://arxiv.org/pdf/1312.6199v4.pdf

[24] M. Nielsen, *Neural Networks and Deep Learning*, 2018.

[25] R. Charnet, C. A. d. L. Freire, E. M. R. Charnet, and H. Bonvino, *Análise de modelos de regressão linear: com aplicações*, 2nd ed. Campinas-SP: Editora Unicamp, 2008.