Utilizando de redes neurais sem peso para predição de cargas de trabalho em clusters

Stanley C. de Sousa[1]

[1]PESC/COPPE – Universidade Federal do Rio de Janeiro (UFRJ)
Rio de Janeiro, RJ – Brasil

{stanley}@cos.ufrj.br

1. Introdução

A computação em nuvem, através da utilização de clusters, possibilita o desenvolvimento de aplicações escaláveis e de alta disponibilidade com expectativa de custos reduzidos de infraestrutura. Um dos elementos fundamentais para materializar essas premissas é a adoção de estratégias de auto escalonamento, no entanto, esta é uma tarefa complexa dada a imprevisibilidade das cargas de trabalhos geradas pelos usuários finais[1]

As estratégias de auto escalonamento podem ser classificadas como reativas ou proativas. As estratégias reativas monitoram os eventos do sistema e decidem alocar ou liberar recursos de acordo com parâmetros pré configurados. Já as estratégias pró-ativas buscam prever os recursos que serão utilizados em um determinado momento e gerenciá-los de acordo[2].

A predição da carga de trabalho é um problema de regressão, pois busca-se estimar variáveis ao longo do tempo. Este tipo de problema é considerado mais leve computacionalmente do que problemas de classificação[1], apesar disso, sua aplicação prática em sistemas de produção ainda é uma tarefa difícil. Um conjunto de modelos de aprendizagem de máquina que têm apresentado resultados de acurácia bastante competitivos com o estado da arte, em diversos domínios, mas com performance superior em até algumas ordens de grandeza[3] são as redes neurais sem peso.

Neste trabalho iremos utilizar redes neurais sem peso, em particular a Regression WiSARD(ReW) e a ClusRegression WiSARD(CReW)[4], para predição de cargas de trabalho no contexto citado anteriormente. Para os testes do modelo de aprendizagem foram utilizados registros de requisições HTTP de um período de dois meses disponibilizado pela NASA[5].

2. Dados utilizados

Os dados utilizados neste trabalho são registros de requisições HTTP do servidor WWW do centro espacial Kennedy, da NASA[5]. Os dados estão disponíveis em:

- ftp://ita.ee.lbl.gov/html/contrib/NASA-HTTP.html

O log, em codificação ASCII, contém uma requisição por linha conforme exemplo a seguir:
kgtyk4.kj.yamagata-u.ac.jp - - [01/Aug/1995:00:00:17 -0400] ”GET / HTTP/1.0”200 7280

Cada linha contém os seguintes dados, em ordem:

- Hostname (endereço do solicitante)
- Timestamp com resolução de segundo
- Requisição (ação e recurso solicitado)
- Código HTTP de estado da resposta
- Tamanho da resposta em bytes

Neste trabalho utilizamos apenas o Timestamp e contamos a quantidade de requisições em diferentes janelas de tempo. A série das requisições pode ser vista na Figura 1.

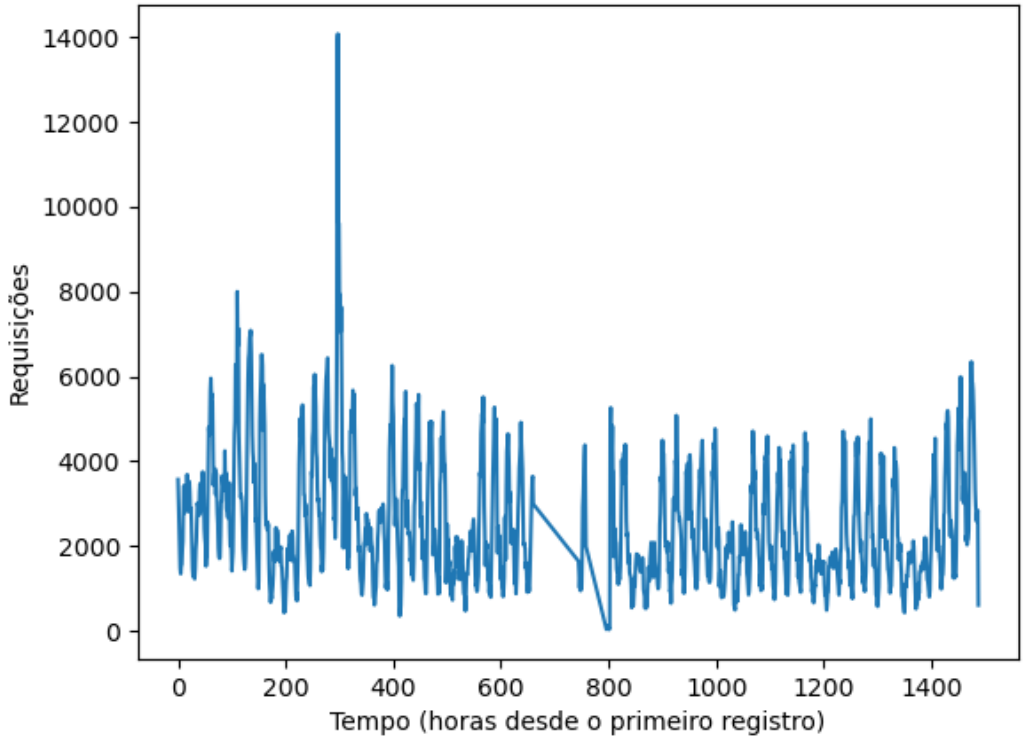

Figura 1. Requisições ao longo do tempo

As janelas de tempo utilizadas foram 1, 5, 15, 30 e 60 minutos.

3. Modelagem dos dados e pré-processamento de entrada

Os logs são disponibilizados em arquivos mensais e seu formato não era adequado para utilização direta por este trabalho. Foi necessário o tratamento dos dados em duas fases, o pré-processamento e a codificação binária.

3.1. Pré-processamento

O pré-processamento foi realizado da seguinte forma:

- Extração do timestamp de cada linha do log
- Agregação dos tempos de acordo com as diferentes janelas, contando as ocorrências
- Conversão do timestamp para o padrão ISO
- Escrita dos registros agregados em um novo arquivo

Foi gerado um arquivo para cada janela de tempo.

3.2. Codificação

Observando a Figura1 notamos que a série tem comportamento periódico. Para tentar capturar essa periodicidade convertemos o timestamp em um número inteiro capaz de representar variações intra-dia e intra-semana descartando a informação referente ao mês, pois os dados não são suficientes para observarmos sazonalidade em intervalos maiores. A conversão foi dada de acordo com os seguintes passos:

- Contagem de segundos desde as 00:00 da data avaliada
- Conversão do dia da semana em um número de 0 a 6
- Concatenação dos dois números anteriores, de forma que o último dígito represente o dia da semana

O número inteiro gerado por esta conversão foi então transformado em um número binário utilizado a codificação termômetro para que pudesse ser utilizado na entrada da rede neural. Como optamos por utilizar um modelo de regressão, não foi necessário transformar a contagem de requisições associadas a cada timestamp em um número binário, pois tanto a ReW quanto a CReW recebem números de ponto flutuante como valor alvo.

4. Métricas, exploração de modelos e parâmetros

Neste trabalho utilizamos a raiz do erro médio quadrático (RMSE) como métrica de acurácia. Antes de fazermos uma análise dos resultados exploramos parâmetros para a ReW e para a CReW, as redes neurais sem peso escolhidas para este trabalho. Para efeitos comparativos, utilizamos também os modelos Linear SVM (Support Vector Machine) e KNN (K Nearest Neighbors) com uma exploração mínima de seus parâmetros.

Para todos os ensaios foi utilizada a janela de 30 minutos, o processamento foi realizado 10 vezes e o resultado médio com o desvio padrão é apresentado. As tabelas a seguir apresentam um resumo dos parâmetros explorados, na tabela1 vemos os dados para a ReW, na tabela2 para a CReW, na tabela3 para o KNN e na tabela4 a exploração para o modelo Linear SVM.

Tabela 1. Exploração da ReW

AddressSize	Mean	RMSE
5	SimpleMean	593.022±0.000
10	SimpleMean	552.525±0.000
15	SimpleMean	554.201±0.000
20	SimpleMean	580.119±0.000
10	Median	571.270±0.000
10	GeometricMean	553.381±0.061
10	HarmonicMean	548.410±0.000
10	PowerMean(2)	559.541±0.000

Tabela 2. Exploração da CReW

AddressSize	Threshold	Limit	RMSE
5	0	0	585.374±0.000
10	0	0	571.218±0.000
15	0	0	569.946±0.000
20	0	0	553.296±0.000
30	0	0	555.884±0.000
20	5	50	559.929±3.303
15	5	75	555.984±2.224
10	5	75	547.641±1.637

Tabela 3. Exploração do KNN

Neighbors	Weights	Power	RMSE
5	Uniform	1	571.644±0.000
5	Distance	1	570.973±0.000
5	Uniform	2	522.033±0.000
5	Distance	1	608.116±0.000
10	Uniform	2	748.078±0.000

Tabela 4. Exploração do LinearSVM

Epsilon	Tolerance	RMSE
0.5	0.01	746.812±0.000
0.5	0.001	747.105±0.000
0.5	0.0001	611.491±0.000
1.0	0.0001	611.467±0.000
2.0	0.0001	611.198±0.000
3.0	0.0001	611.592±0.000

5. Resultados

Escolhemos a ReW(10, SimpleMean) e a CReW(10, 5 75) para testes com as várias janelas de tempo preparadas. Utilizamos os modelos KNN(5, Uniform, 2) e LinearSVM(2, 000.1) para comparação. A tabela5 apresenta os resultados. Vemos que a CReW apresentou melhores resultados em todas as janelas de tempo avaliadas, sendo que para janelas de tempo menores a diferença entre a acurácia dos modelos se torna proporcionalmente menor.

Tabela 5. Acurácia da predição (RMSE)

Janela	ReW	CReW	KNN	LinearSVM
1min	23.714±0.193	23.375±0.070	24.955±0.000	28.954±0.000
5min	103.219±0.204	101.735±0.059	107.864±0.000	130.038±0.000
15min	291.730±1.385	289.226±0.799	330.338±0.000	376.280±0.000
30min	560.319±0.000	548.575±2.920	748.078±0.000	747.082±0.000
60min	1024.333±0.000	980.816±3.777	1479.846±0.000	1491.327±0.000

A imagem2 apresenta um corte de 50 intervalos de 5 minutos da série atual e das predições de cada modelo, enquanto a imagem3 apresenta as mesmas informações para intervalos de 60 minutos.

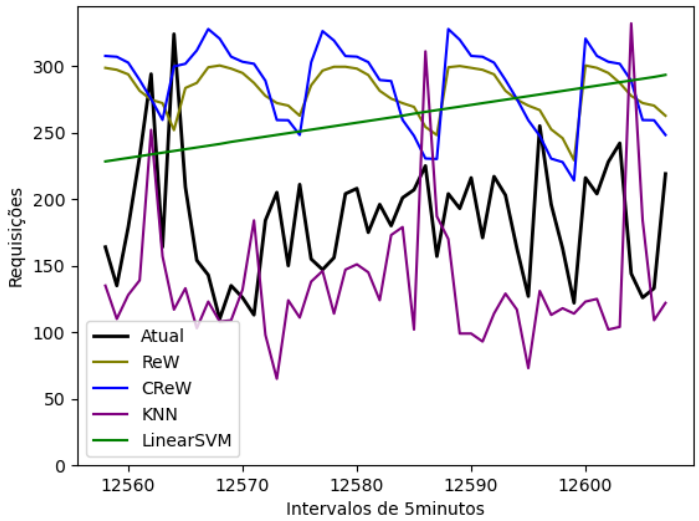

Figura 2. Requisições em intervalos de 5 minutos

A imagem2 parece indicar que a CReW apresenta uma tendência de superestimar o valor de sua predição em períodos de variações bruscas, apresentando inércia apenas para descida da curva. Este comportamento pode estar relacionando ao mecanismo de predição da CReW se basear na média de valores acumulados, que se ajusta melhor a variações mais suaves. Na imagem3 é possível ver que para variações menos bruscas a CReW se ajusta bem à curva base.

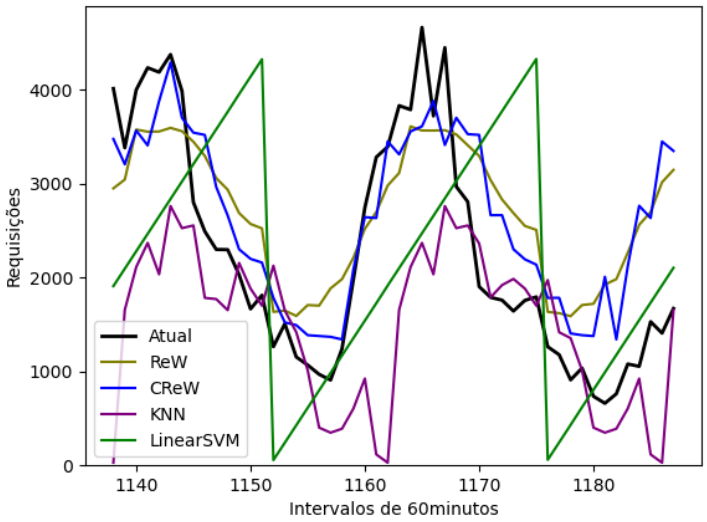

Figura 3. Requisições em intervalos de 60 minutos

Tabela 6. Tempos em segundos (treinamento / predição)

Janela	ReW	CReW	KNN	LinearSVM
1min	0.110 / 0.112	7.263 / 3.788	0.086 / 0.908	0.025 / 0.010
5min	0.023 / 0.024	1.499 / 0.810	0.011 / 0.184	0.006 / 0.002
15min	0.008 / 0.008	0.499 / 0.272	0.004 / 0.064	0.002 / 0.001
30min	0.004 / 0.004	0.242 / 0.006	0.003 / 0.033	0.001 / 0.000
60min	0.002 / 0.002	0.115 / 0.067	0.002 / 0.017	0.001 / 0.000

A tabela6 apresenta os tempos de treinamento e predição de cada modelo.

O código fonte deste trabalho se encontra em: https://github.com/stanleysousa/wisard-workload-prediction

6. Conclusão

Como podemos observar na imagem1, a série temporal abordada neste trabalho apresenta comportamento periódico, mas com amplitude bastante variável. Neste sentido, a média e a variância das requisições por unidade de tempo mudam ao longo da série, sendo assim um processo não estacionário. Neste contexto, a CReW obteve melhor acurácia em comparação com as técnicas simples de regressão também implementadas neste trabalho, no entanto, outros modelos mais sofisticados apresentam resultados significativamente melhores. Na tabela7 apresentamos os resultados deste trabalho em conjunto com os resultados para os modelos Multi Layer Perceptron(MLP), Gated Recurrent Unit(GRU) e Auto Regressive Integrated Moving Average(ARIMA) apresentados por Kirchoff et al. 2019[1].

Tabela 7. RMSE - Comparação com outros modelos

Janela	CReW	KNN	LinSVM	MLP[1]	GRU[1]	ARIMA[1]
1min	23.375	24.955	28.954	16.26	16.51	14.27
5min	101.735	107.864	130.038	54.34	53.04	47.86
15min	289.226	330.338	376.280	136.36	132	135.08
30min	548.575	748.078	747.082	249.55	242.86	237.73
60min	980.816	1479.846	1491.327	544.08	521.66	493.90

No modelo MLP um problema conhecido é a dissipação do gradiente, situação que pode ser minimizada com a utilização de uma função de ativação retificada[6]. Já a GRU, que é uma rede neural recorrente com peso, foi desenvolvida com o mesmo problema de gradiente em mente, visando também minimizá-lo[7]. A ARIMA, por sua vez, é um modelo autoegressivo projetado especificamente para lidar com processos não estacionários. Estas características justificam os bons resultados desses modelos.

Na CReW, a dificuldade em ajustar para baixo a predição, diante de variações de alta frequência e amplitude pouco uniforme, parece ser um importante fator a ser abordado para melhorar sua acurácia em séries não estacionárias, assim como o problema de gradiente em redes neurais com peso. Esta hipótese, no entanto, precisa ser melhor avaliada com mais trabalhos deste tipo.

Referências

[1] Dionatra F. Kirchoff, Miguel Xavier, Juliana Mastella, and Cesar A. F De Rose. A preliminary study of machine learning workload prediction techniques for cloud applications, 2019.

[2] Valter Messias, Julio Estrella, Ricardo Ehlers, Marcos Santana, Regina Santana, and Stephan Reiff-Marganiec. Combining time series prediction models using genetic algorithm to autoscaling web applications hosted in the cloud infrastructure. *Neural Computing & Applications*, 27(8):2383 – 2406, 2016.

[3] Leopoldo AD Lusquino Filho, Luiz FR Oliveira, Aluizio Lima Filho, Gabriel P Guarisa, Lucca M Felix, Priscila MV Lima, and Felipe MG França. Prediction of palm oil production with an enhanced n-tuple regression network. *ESANN 2019 proceedings*, 2019.

[4] Leopoldo AD Lusquino Filho, Luiz FR Oliveira, Aluizio Lima Filho, Gabriel P Guarisa, Lucca M Felix, Priscila MV Lima, and Felipe MG França. Extending the weightless wisard classifier for regression. *Neurocomputing*, 2020.

[5] NASA. *HTTP requests to the NASA Kennedy Space Center WWW server in Florida*, Acessado em 05 de Julho de 2020.

[6] Chi-Feng Wang. *The Vanishing Gradient Problem*, 2019.

[7] Data Science Academy. *Deep Learning Book*, Acessado em 01 de Julho de 2020.