MAC6958 - Tópicos Avançados em Ciência de Dados para Redes de Computadores

Reprodução do Artigo:
Understanding the Modeling of Computer Network Delays using Neural Networks
Mestres, Albert & Alarcón, Eduard & Ji, Yusheng & Cabellos-Aparicio, Albert. (2018). Understanding the Modeling of Computer Network Delays using Neural Networks. 46-52. 10.1145/3229607.3229613.

Caio Lorenzetti Martinelli - nº USP 8539899
Orientador:
- Carlos Hitoshi Morimoto

Professores da Disciplina:
- Daniel Macedo Batista
- Roberto Hirata Junior

Introdução

O uso de *Machine Learning* (ML) no ramo de redes é crescente devido a duas tecnologias possibilitadoras:

1. *Software-Defined Networking* (SDN)
   a. Transforma um sistema distribuído em um sistema centralizado - logicamente.
2. Network Analytics (NA)
   a. Permite a extração de informações úteis para o controle do primeiro.

A combinação dos dois possibilita um novo paradigma: o *Knowledge-Defined Networking* (KDN) que vem tendo uso crescente.

Introdução + *Use-Case*

Modelagem considera a rede uma caixa preta:

- *Input*: Matriz de tráfego entre nós.
- *Output:* Atraso entre nós.

Paper tenta validar a possibilidade e criar *guidelines* para a modelagem de redes.

Outras formas de modelar:

- Analiticamente: Modelos de fila.
- Computacionalmente: Simuladores.

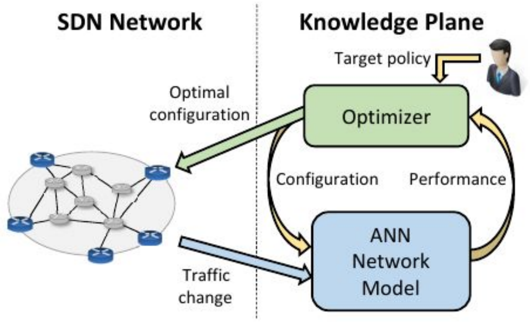
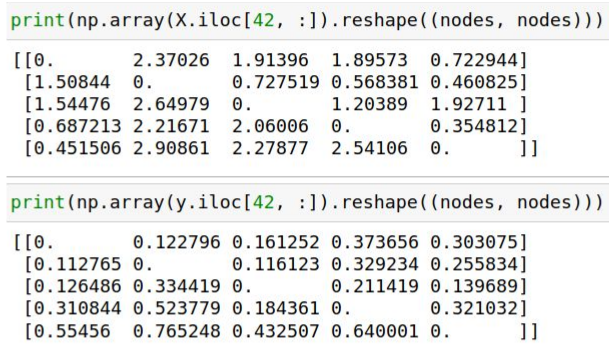


Figure 1: Graphical representation of the optimization use-case.

Problem Statement

$$\mathbf{D} = f(\mathbf{T})$$



```
print(np.array(X.iloc[42, :]).reshape((nodes, nodes)))

[[0.       2.37026  1.91396  1.89573  0.722944]
 [1.50844  0.       0.727519 0.568381 0.460825]
 [1.54476  2.64979  0.       1.20389  1.92711 ]
 [0.687213 2.21671  2.06006  0.       0.354812]
 [0.451506 2.90861  2.27877  2.54106  0.      ]]
```

```
print(np.array(y.iloc[42, :]).reshape((nodes, nodes)))

[[0.       0.122796 0.161252 0.373656 0.303075]
 [0.112765 0.       0.116123 0.329234 0.255834]
 [0.126486 0.334419 0.       0.211419 0.139689]
 [0.310844 0.523779 0.184361 0.       0.321032]
 [0.55456  0.765248 0.432507 0.640001 0.      ]]
```

Configuração da SDN é constante para a ANN.

ANN:

1. Primeira camada: Real, leva em conta topologia, *routing*, tamanho.
2. Camada do meio: Abstração do sistema.
3. Camada de saída: Atrasos.

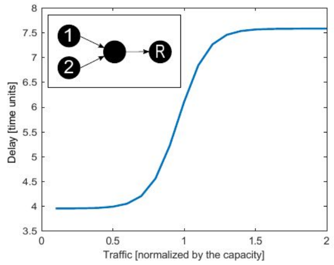
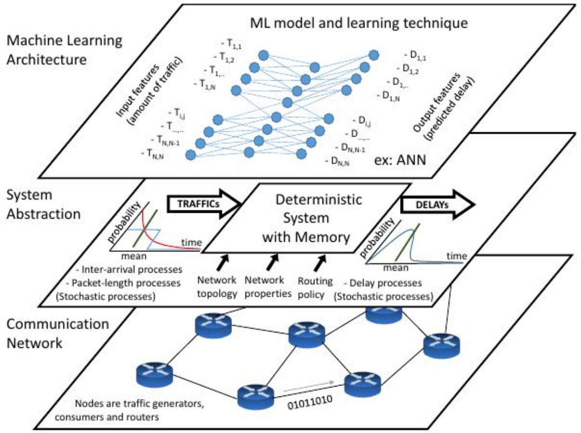


Figure 3: Simple example of the function to learn

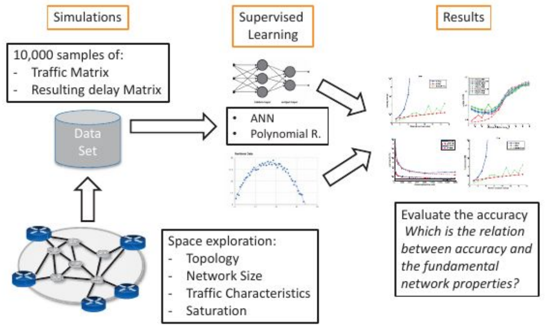


Metodologia

"We split the dataset into three sets, the training set with 60% of the samples, the validation set with 20% of the samples and the test set with the remaining 20% of the samples. The training set is used to optimize the ML model, the validation set is used to evaluate the model during the training phase, and the test set is used to provide an independent evaluation of the performance."

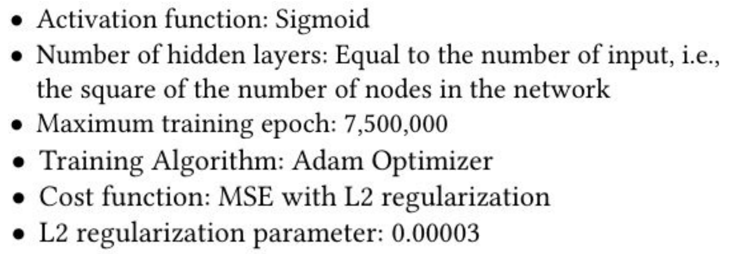
Datasets gerados a partir do simulador *Omnet++* (versão 4.6). São considerados diversos parametros para as simulações:

1. Topologia.
2. Tamanho da rede.
3. Distribuição de probabilidade do tráfego (tamanho dos pacotes).
4. Intensidade máxima por nó.
5. *Routing.*



Metodologia

Parâmetros padrão das redes, obtidos por *tunning* manual:



- Activation function: Sigmoid
- Number of hidden layers: Equal to the number of input, i.e., the square of the number of nodes in the network
- Maximum training epoch: 7,500,000
- Training Algorithm: Adam Optimizer
- Cost function: MSE with L2 regularization
- L2 regularization parameter: 0.00003

$$error = \frac{1}{S}\frac{1}{N^2}\sum_{i=1}^{S}\sum_{i=1}^{N^2}(\hat{d}_i - d_i)^2$$

Utilizei a função de ativação *relu* na última camada.

Uma ou duas camadas internas.

Early Stopping nos dados de validação.

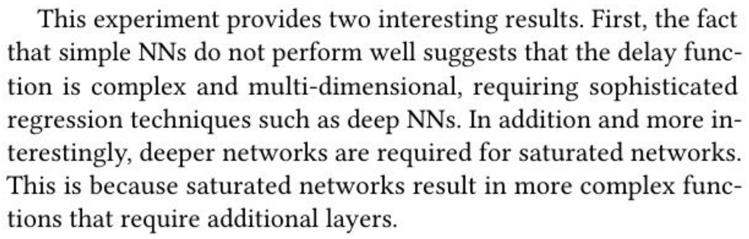
Função de custo customizada. Tive problemas com a função MSE do *TensorFlow* (2.1.0).

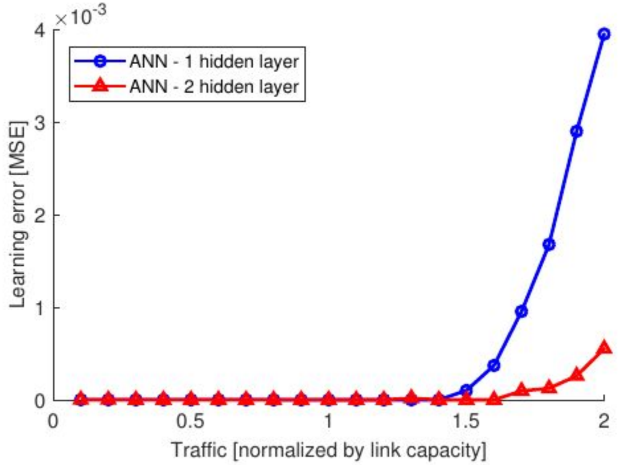
Não é especificada a unidade de medida.

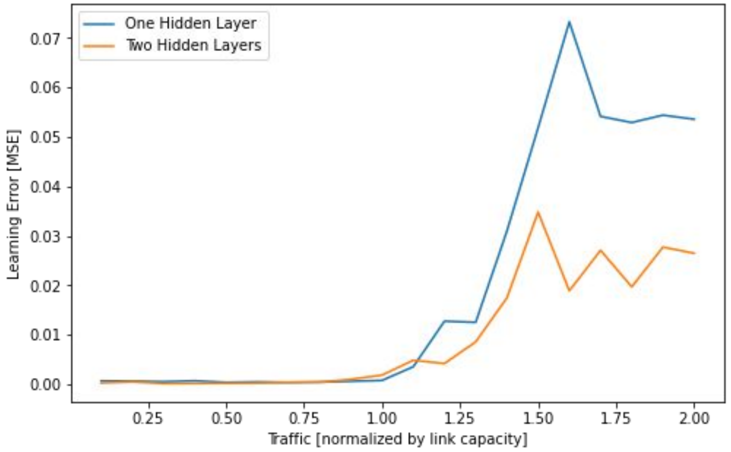
Erro x Intensidade

Tráfego por nó segue distribuição uniforme (0, ρmax] (eixo x).

Topologia *Scale-Free.*


This experiment provides two interesting results. First, the fact that simple NNs do not perform well suggests that the delay function is complex and multi-dimensional, requiring sophisticated regression techniques such as deep NNs. In addition and more interestingly, deeper networks are required for saturated networks. This is because saturated networks result in more complex functions that require additional layers.

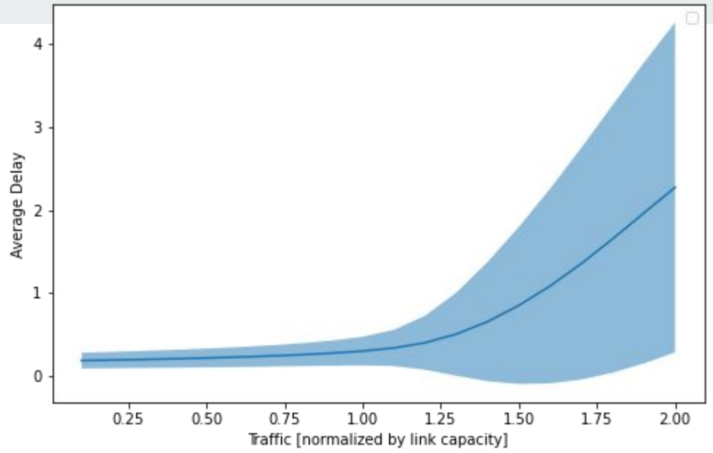


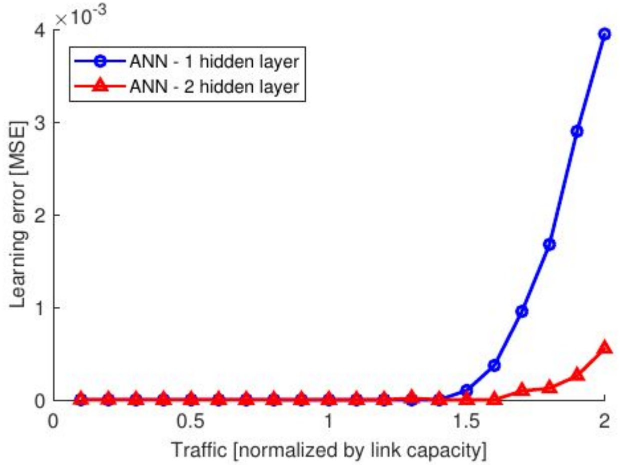


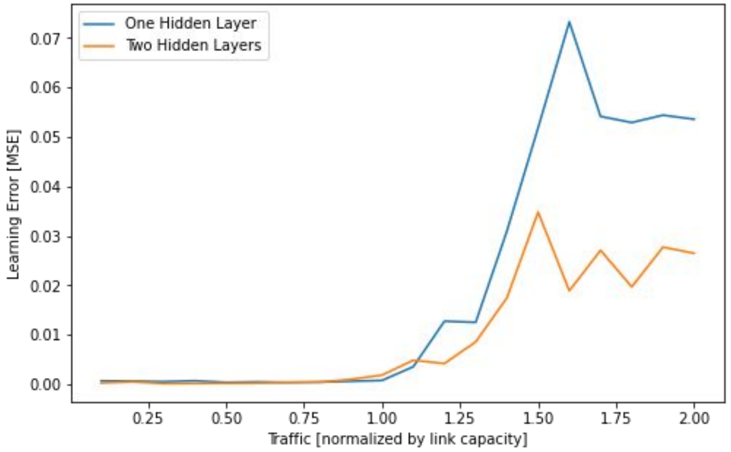
Erro x Intensidade

Atraso aumenta com aumento do tráfego.

Métrica de erro não considera valor da resposta e não é robusta a *outliers*.





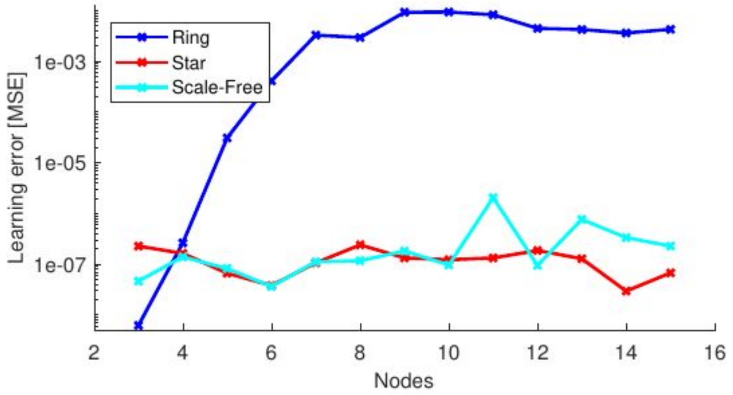


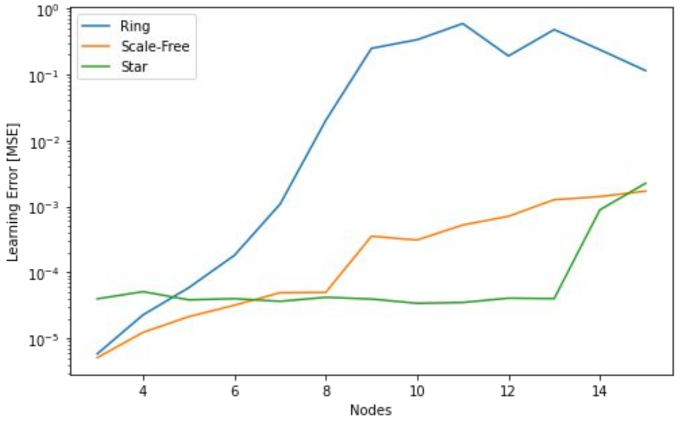
Erro x Topologia

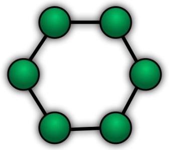
Três topologias consideradas:

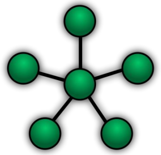
1. *Ring.*
2. *Star.*
3. *Scale-Free.*

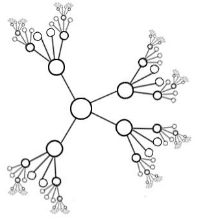
ρmax = 0.6 - embora nome da base indique 0.4.





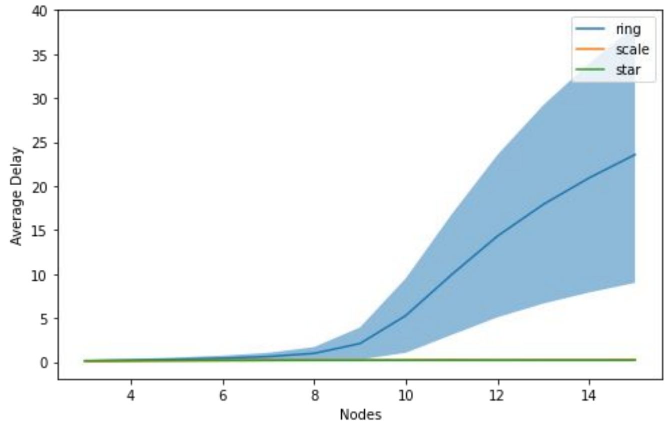

Ring

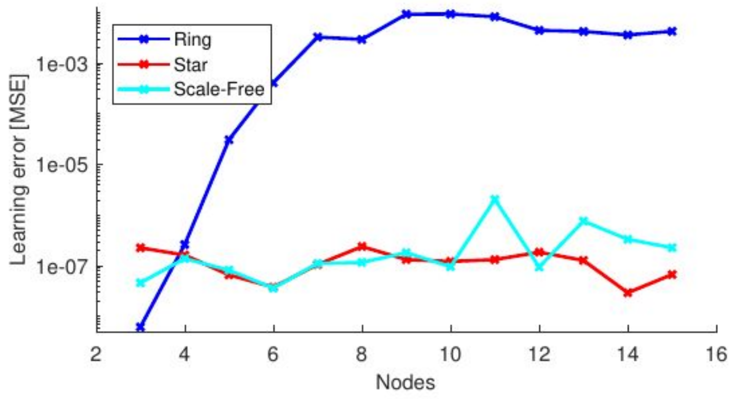

Star

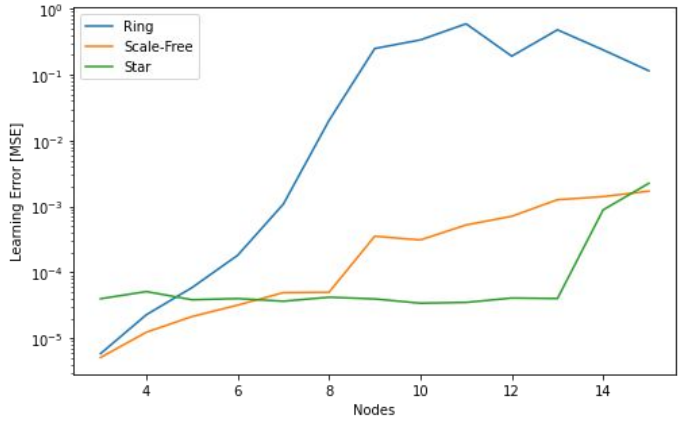


$$P(k) \sim k^{-\gamma}$$

Erro x Topologia







We conclude that, for the considered scenarios, the topology or the size *has no impact* on the accuracy of the NN estimates. The only impact of such fundamental network characteristics is that depending on the specific topology this may increase the saturation, which results in more queuing requiring deeper networks as we have seen in the previous experiment.

Erro x Neurônios

Topologia *Scale-Free.*

ρmax alto - nome da base indica 1.5.

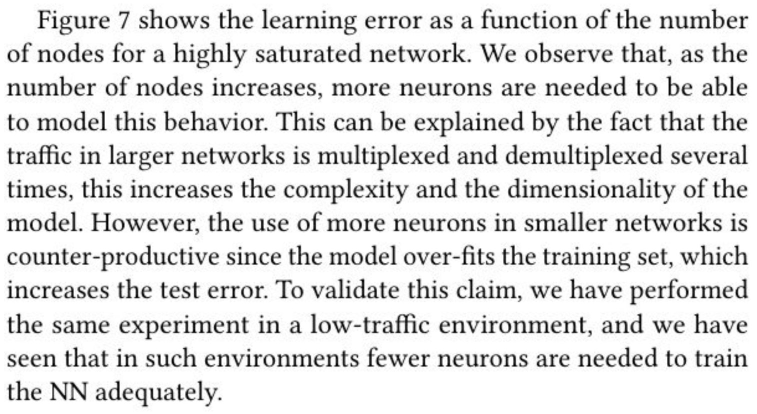
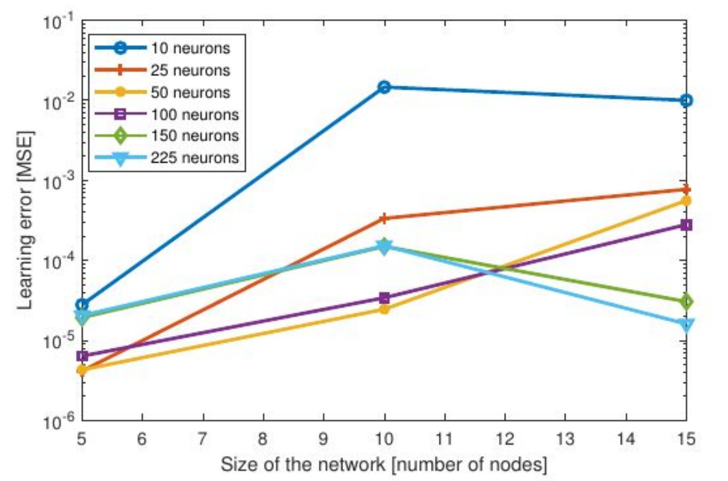
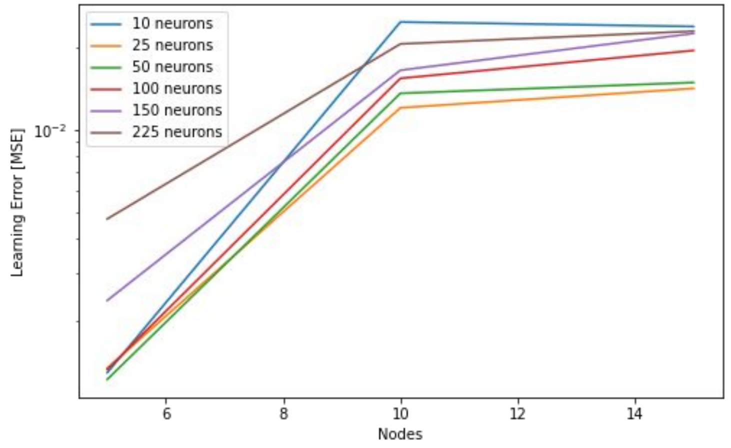


Figure 7 shows the learning error as a function of the number of nodes for a highly saturated network. We observe that, as the number of nodes increases, more neurons are needed to be able to model this behavior. This can be explained by the fact that the traffic in larger networks is multiplexed and demultiplexed several times, this increases the complexity and the dimensionality of the model. However, the use of more neurons in smaller networks is counter-productive since the model over-fits the training set, which increases the test error. To validate this claim, we have performed the same experiment in a low-traffic environment, and we have seen that in such environments fewer neurons are needed to train the NN adequately.





Erro x *Routing*

ρmax 0.5.

Não consegui reproduzir devido ao nome das bases.

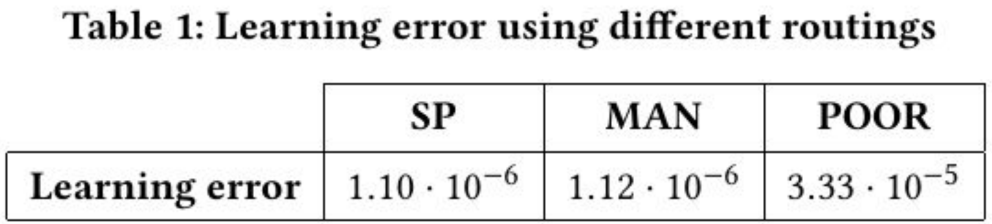
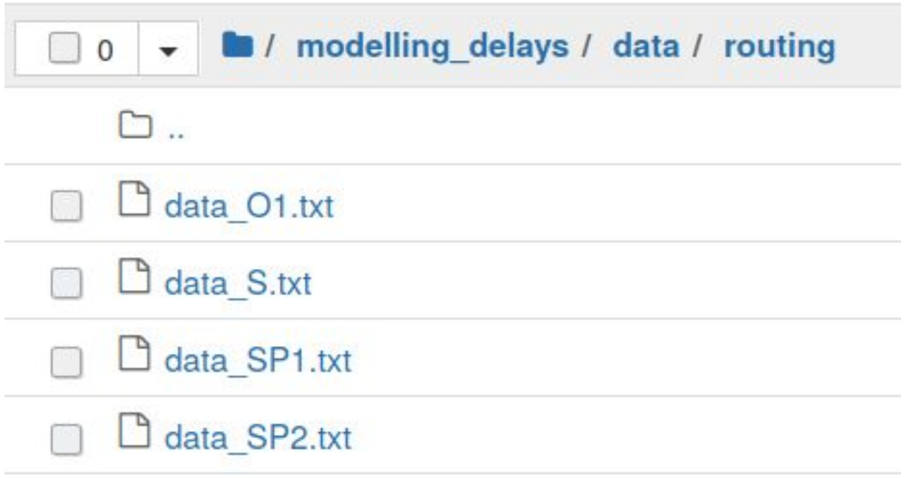


Table 1: Learning error using different routings

	SP	MAN	POOR
Learning error	$1.10 \cdot 10^{-6}$	$1.12 \cdot 10^{-6}$	$3.33 \cdot 10^{-5}$



/ modelling_delays / data / routing

..

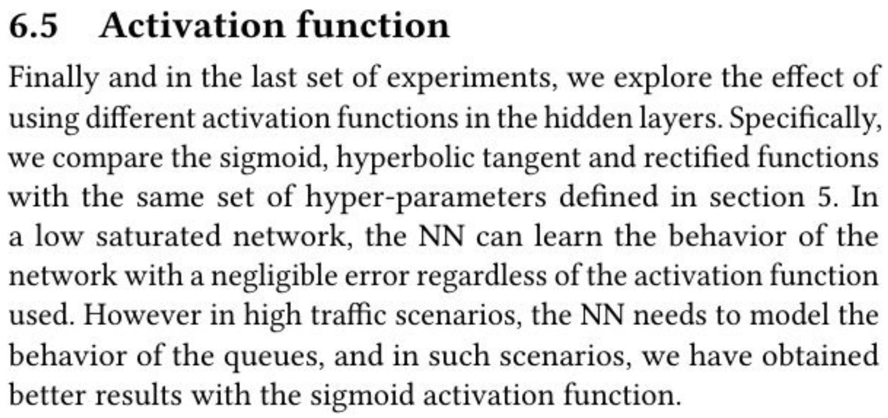
data_O1.txt

data_S.txt

data_SP1.txt

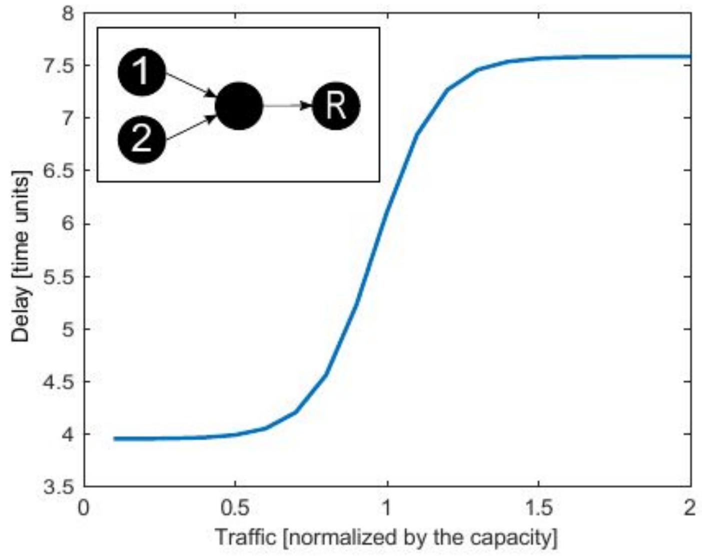
data_SP2.txt

Erro x Função de Ativação

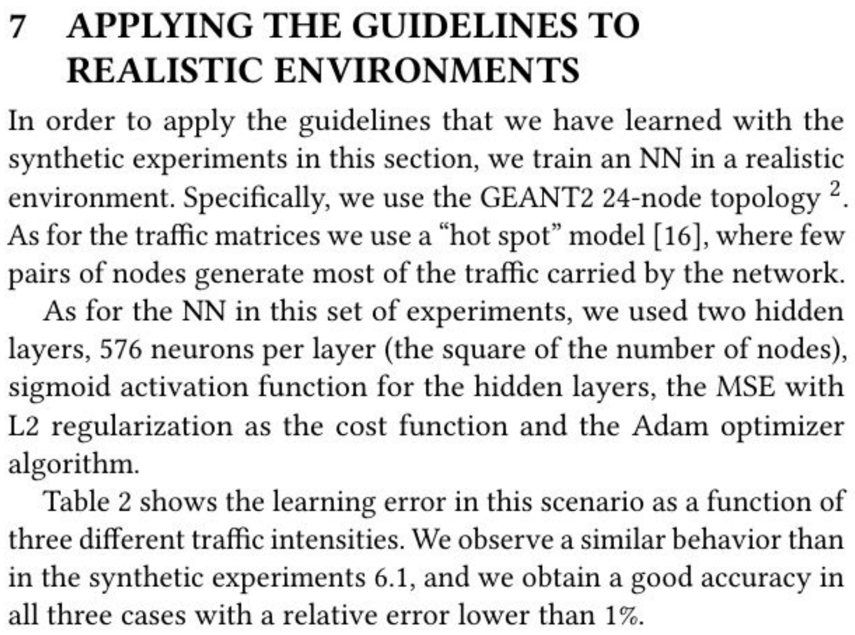


6.5 Activation function

Finally and in the last set of experiments, we explore the effect of using different activation functions in the hidden layers. Specifically, we compare the sigmoid, hyperbolic tangent and rectified functions with the same set of hyper-parameters defined in section 5. In a low saturated network, the NN can learn the behavior of the network with a negligible error regardless of the activation function used. However in high traffic scenarios, the NN needs to model the behavior of the queues, and in such scenarios, we have obtained better results with the sigmoid activation function.



Teste em Ambiente Realista



7 APPLYING THE GUIDELINES TO REALISTIC ENVIRONMENTS

In order to apply the guidelines that we have learned with the synthetic experiments in this section, we train an NN in a realistic environment. Specifically, we use the GEANT2 24-node topology [2]. As for the traffic matrices we use a "hot spot" model [16], where few pairs of nodes generate most of the traffic carried by the network.

As for the NN in this set of experiments, we used two hidden layers, 576 neurons per layer (the square of the number of nodes), sigmoid activation function for the hidden layers, the MSE with L2 regularization as the cost function and the Adam optimizer algorithm.

Table 2 shows the learning error in this scenario as a function of three different traffic intensities. We observe a similar behavior than in the synthetic experiments 6.1, and we obtain a good accuracy in all three cases with a relative error lower than 1%.

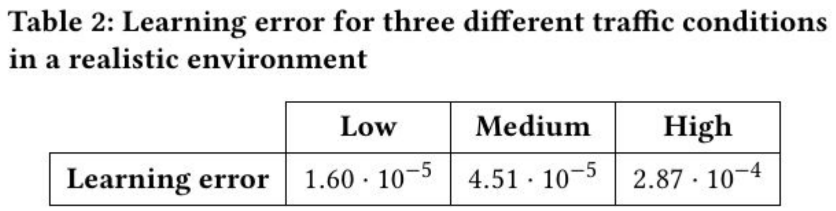


Table 2: Learning error for three different traffic conditions in a realistic environment

	Low	**Medium**	**High**
Learning error	$1.60 \cdot 10^{-5}$	$4.51 \cdot 10^{-5}$	$2.87 \cdot 10^{-4}$

Discussão + Conclusão

Discussão:

- Acurada mesmo em cenários complexos:
  - Vantagem sobre simplificações analíticas.
- Rápida e de baixo custo:
  - Vantagem sobre simuladores - para avaliar o atraso (escoragem).
- *Data-driven*:
  - Desvantagem. É necessária coleta de dados (virtualmente para todas as configurações da rede, para uma predição acurada). É necessário alto processamento para treinar.

Conclusões:

- ANNs modelam bem o atraso a partir da matriz de tráfego.
- Alto tráfego gera uma necessidade por redes mais profundas.

Dúvidas????

Thanks!

Contact us:

caio.martinelli@usp.br
https://github.com/caiolmart/reproducing_articles/tree/master/modelling_delays

Referências

https://github.com/knowledgedefinednetworking/Understanding-the-Modeling-of-Network-Delays-using-NN

https://en.wikipedia.org/wiki/Scale-free_network

https://en.wikipedia.org/wiki/Network_topology