Trabalho 3

Frederico Schardong

1 Enunciado do trabalho

O enunciado do trabalho 3 propõem a alteração de múltiplos parâmetros de um código matlab onde uma rede neural é criada para realizar a regressão das funções seno e cosseno juntas. A entrada da rede possui duas dimensões, que são alimentadas com o resultado das funções seno e cosseno aplicadas a 200 valores aleatórios no intervalo $[0; 2\pi)$ somadas a um ruído no intervalo $[-0,5; 0,05]$. A rede possui uma camada intermediária com 10 neurônios, usa as funções de ativiação `logsig` e `tansig`, e o algoritmo de treinamento é o gradiente descendente. O enunciado define o limite de 20000 épocas de aprendizado ou taxa de erro de $0,001$, usando taxa de aprendizado de $0,05$.

2 Detalhes de implementação

O trabalho foi implementado em Python 3, utilizando as bibliotecas: (i) `numpy` [2] para gerar números aleatórios e calcular o seno e cosseno; (ii) `matplotlib` [1] para gerar os gráficos referente aos resultados; e (iii) `neurolab`[1] que implementa a função `newff`, responsável por criar, treinar e simular a rede neural. Todas as bibliotecas podem ser instaladas através do gerenciador de pacotes padrão do Python, o `pip`, através do comando:

```
$ pip install numpy matplotlib neurolab
```

O código fonte desta tarefa é publico[2] e possui a licensa MIT. Ele foi submetido junto com este relatório. Para gerar as imagens listas neste relatório e presentes na pasta resultados, basta executa o comando abaixo.

```
$ python3 main.py
```

3 Resultados

O primeiro experimento conduzido foi com os parâmetros fornecidos no enunciado do trabalho (no arquivo `senos_cosseno.m`), descritos na Seção 1. Como pode ser visto na Figura 1, as configurações informadas no enunciado produzem uma saída bastante distante dos dados reais.

3.1 Variação de ruído

O segundo experimento conduzido foi o de variar o ruído dos dados de treinamento. O caso base usa ruído de $0,05$, ou seja, o resultado correto de cada valor usado para treinar a rede neural foi adicionado de valores aleatoriamente e uniformemente selecionados no intervalo $[-0,05; 0,05]$. As Figuras 2a e 2b mostram os dois cenarios com variação de ruído distintos testados, respectivamente $0,001$ e $0,1$. Conforme esperado, o erro total dos dados simulados em relação aos dados reais foi menor no cenário com menor ruído.

3.2 Diferentes taxas de aprendizado

Variando a taxa de aprendizado pode ser observado uma correlação positiva entre taxa de aprendizado e erro. A simulação usando $0,001$ produziu um erro de $0,8049$, menor que o erro de $1,8580$ do caso base, cuja taxa de aprendizado é $0,05$. Já os casos com taxa de aprendizado maiores, produziram erros maiores, conforme pode ser observado nas Figuras 3a, 3b, 3c e 3d.

3.3 Dados além dos de treinamento

As configurações sugeridas no enunciado do trabalho (caso base) não produziram um resultado satisfatório em relação a previsão de seno e cosseno para além dos dados de treinamento, conforme pode ser visto na Figura 4. Para este cenário os dados de entrada utilizados foram escolhidos aleatoriamente no intervalo $[-3\pi; 3\pi)$.

[1] `https://pythonhosted.org/neurolab/`

[2] Código fonte disponível em `https://github.com/fredericoschardong/programming-exercise-3-single-hidden-layer`.

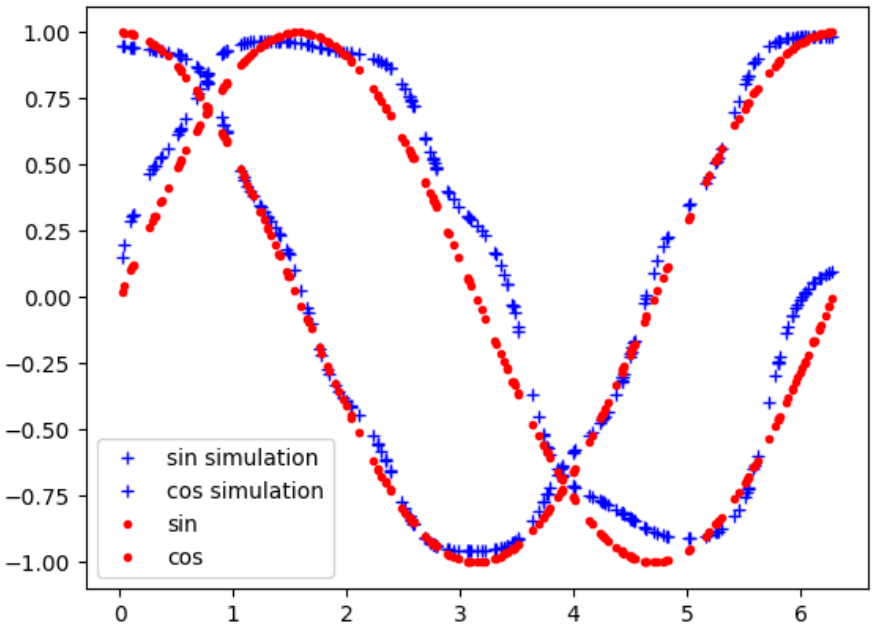
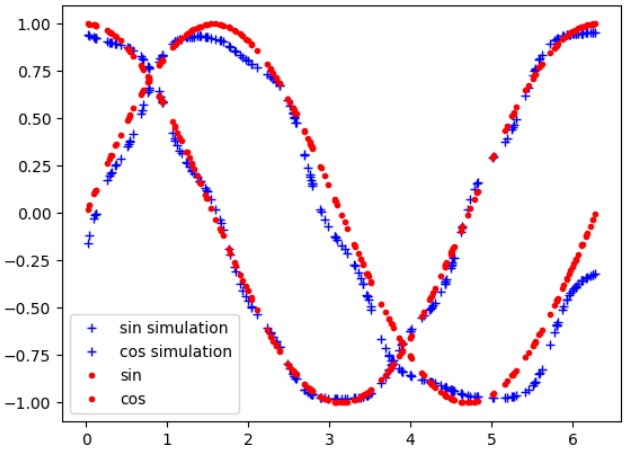
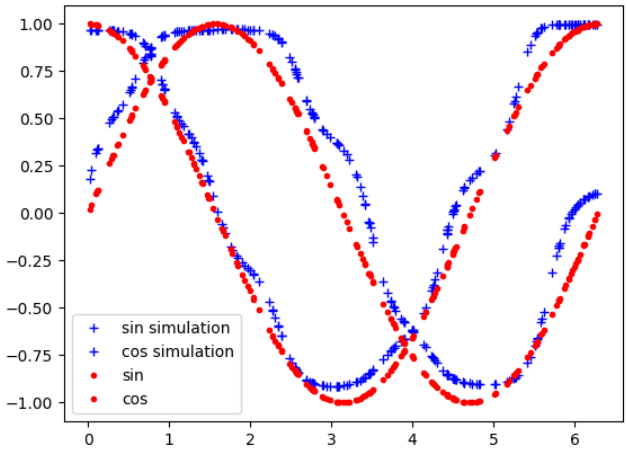


Figure 1: Caso base: em vermelho os dados reais e em azul o resultado da simulação. Erro de $1,8580$.

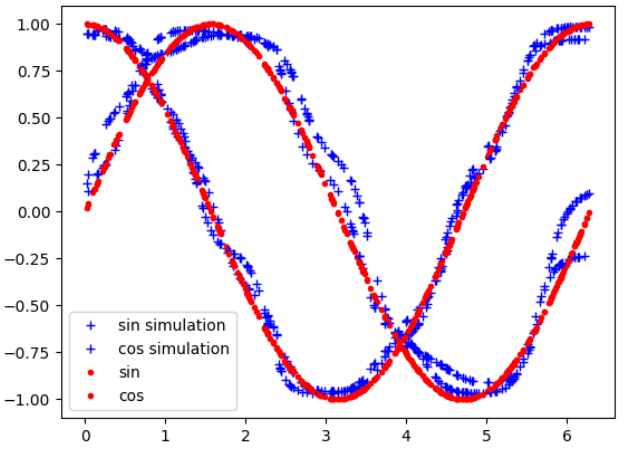


(a) Ruído no intervalo $[-0,001; 0,001]$. Erro total $1,6948$.

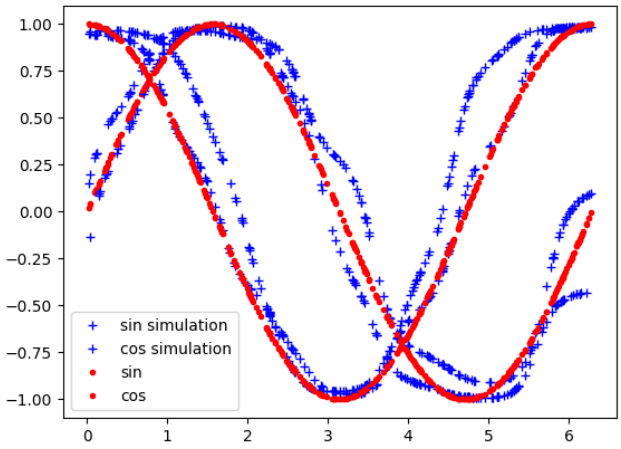


(b) 100 iterações e $[-0,1; 0,1]$. Erro total $2,1685$.

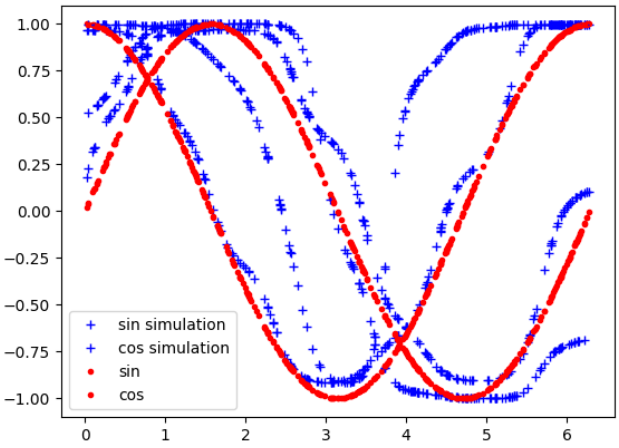
Figure 2: Resultado da regressão para diferentes níveis de ruído durante o treinamento.



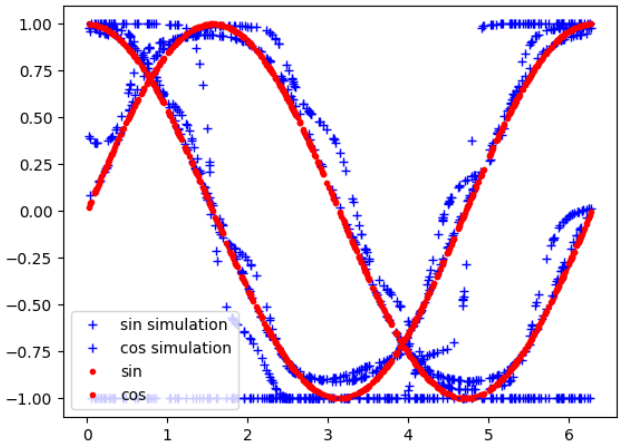
(a) Taxa de aprendizado de $0,001$, erro de $0,8049$.



(b) Taxa de aprendizado de $0,01$, erro de $7,6600$.



(c) Taxa de aprendizado de $0,02$, erro de $47,1092$.



(d) Taxa de aprendizado de $0,05$, erro de $180,6118$.

Figure 3: Resultado da regressão para diferentes taxas de aprendizado.

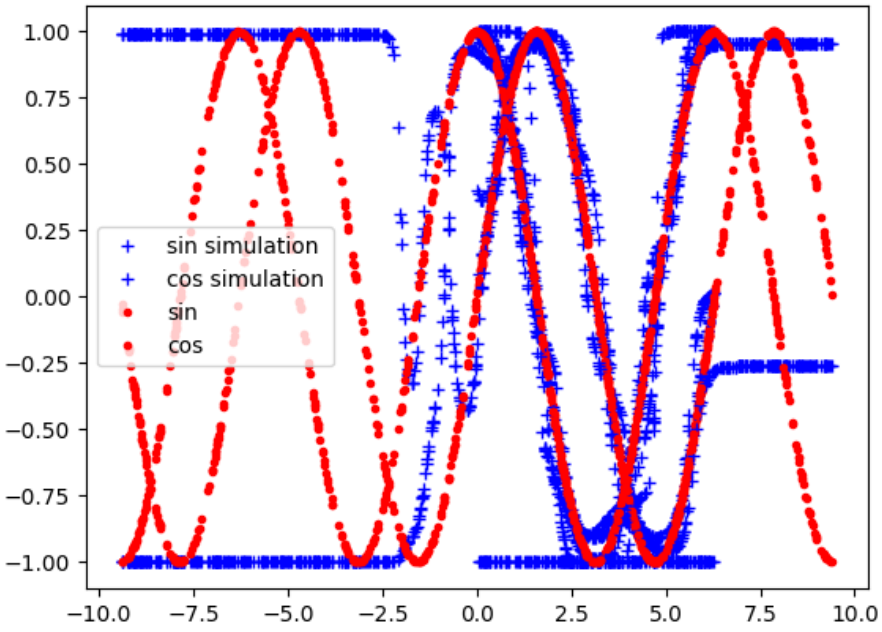


Figure 4: Dados no intervalo $[-3\pi; 3\pi)$.

3.4 Outros experimentos

No enunciado do trabalho também foi solicitado que novos cenários fossem propostos. Duas simulações diferentes foram criadas, conforme descritas abaixo.

3.4.1 Diferentes quantidades de neurônios na camada escondida

Neste experimento foram simuladas redes com diferentes quantidades de neurônios na camada intermediária da rede. A utilização de apenas 5 neurônios apresentou menor erro que todas as outras configurações, inclusive o caso base, que utiliza 10 neurônios. Os resultados podem ser vistos nas Figuras 5a, 5b e 5c.

3.4.2 Diferentes quantidades de dados de treinamento

A variação da quantidade de dados de treinamento também apresentou correlação positiva com o erro. Utilizando apenas 50 instâncias para treinamento produziu erro de $0,2371$, menor que o erro de $1,8580$ do caso base, que utilizou 200 instâncias para treinamento. Os resultados deste experimento são mostrados nas Figuras 6a, 6b e 6c.

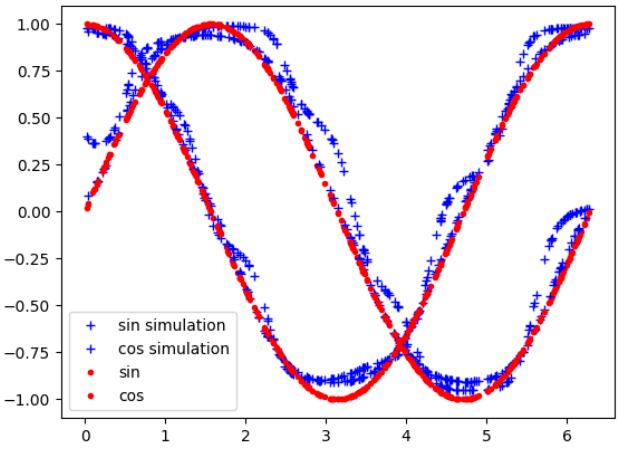
4 Conclusão

Gradiente descendente é um método custoso, que requer muitas épocas de treinamento para produzir resultados satisfatórios. Nenhum dos experimentos executados chegou na taxa de erro desejada de 0.001. Neste trabalho é possível observar como diferentes e isoladas alterações nas configurações de uma rede neural do tipo *feedforward* impactam expressivamente no resultado final.

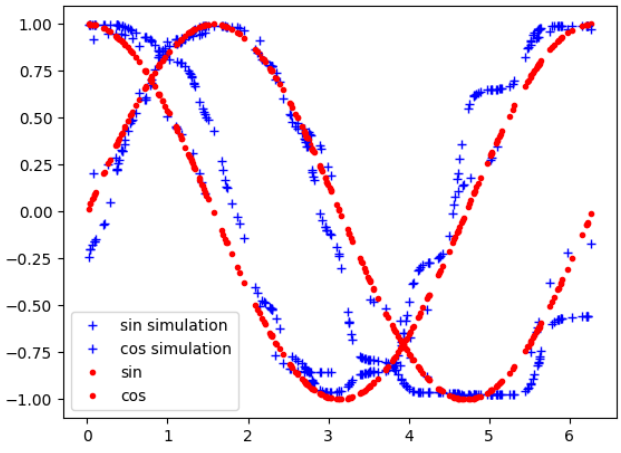
References

[1] John D Hunter. “Matplotlib: A 2D graphics environment”. In: *Computing in science & engineering* 9.3 (2007), pp. 90–95.

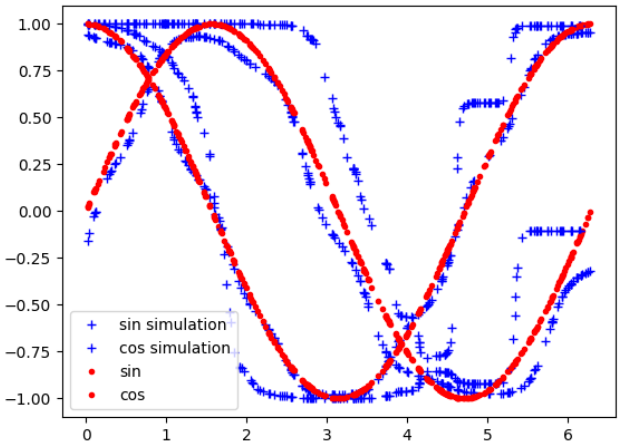
[2] Stéfan van der Walt, S Chris Colbert, and Gael Varoquaux. “The NumPy array: a structure for efficient numerical computation”. In: *Computing in Science & Engineering* 13.2 (2011), pp. 22–30.



(a) 5 neurônios produziu erro de $0,2813$.

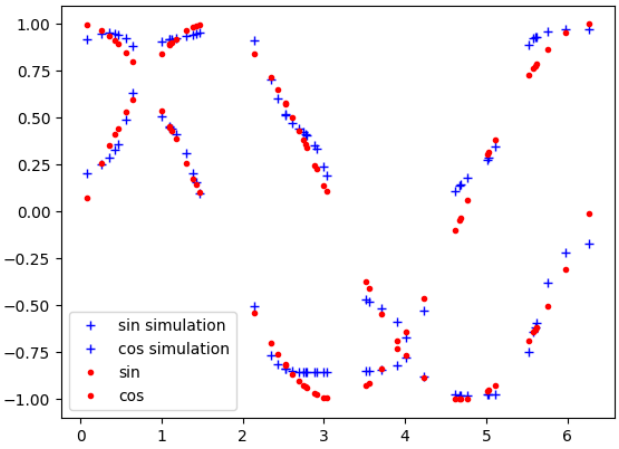


(b) 20 neurônios produziu erro de $8,5810$.

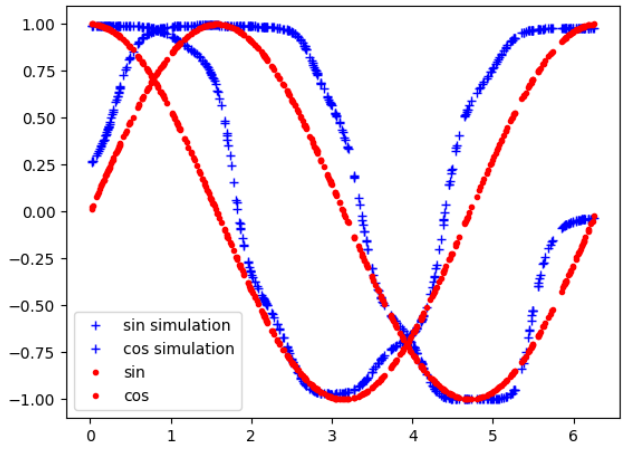


(c) 50 neurônios produziu erro de $19,4609$.

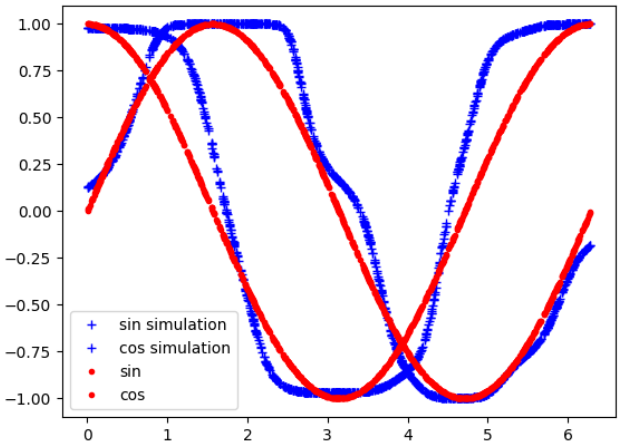
Figure 5: Resultado da regressão para diferentes quantidades de neurônios na camada intermediária.



(a) 50 instâncias para treinamento produziu erro de $0,2371$.



(b) 400 instâncias para treinamento produziu erro de $1,85801$.



(c) 1000 instâncias para treinamento produziu erro de $34,0176$.

Figure 6: Resultado da regressão para diferentes quantidades de dados de treinamento.