ML Under Modern Optimization Lens - Redes Neurais - Exercícios

Giovanni Amorim

Junho, 2023

1 Fully Connected Neural Networks

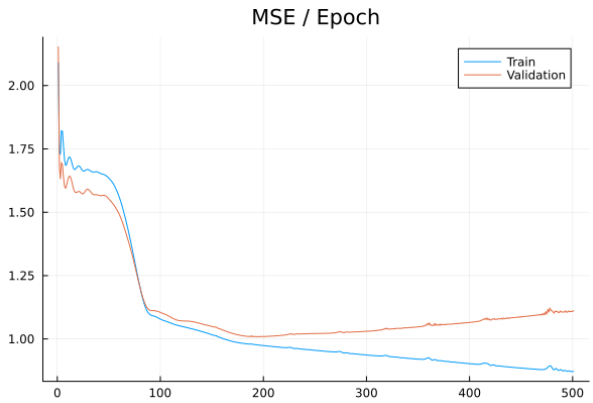
O exercício consiste na aplicação de técnicas de treinamento e avaliação de uma rede neural em um conjunto de dados simulados. Para isso, foram geradas 3000 amostras aleatórias de valores aplicados a uma função que segue a seguinte fórmula:

$$y(x_1, x_2, x_3) = |x_1| + sin(x_2) - cos(x_3) + \epsilon$$

sendo ϵ um ruído branco. O conjunto, depois de criado, foi separado em treino e teste seguindo as proporções de 70%, 30%. A rede neural utilizada segue uma arquitetura totalmente conectada com uma camada escondida, da seguinte forma:

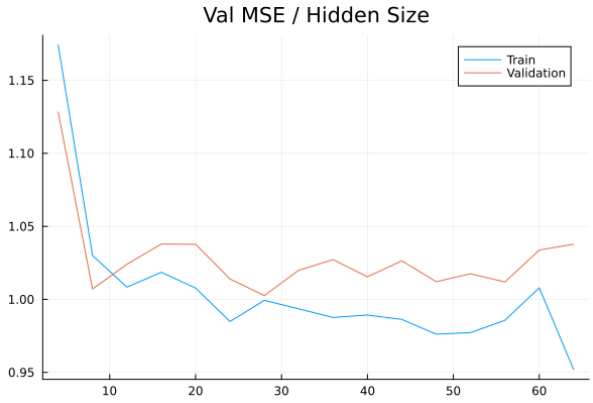
1. Camada de input: 3
2. Camada escondida: $3 => 256$, tanh
3. Camada de output: $256 => 1$

O otimizador utilizado foi o Adam, com taxa de aprendizado igual a 0.01 e a função de custo utilizada foi a de erro médio quadrado. Após 500 épocas de aprendizado, o gráfico de erro (MSE) foi o seguinte:



Como podemos perceber, o erro de validação começa a aumentar e ter uma evolução com comportamento diferente do erro de treino a partir da época 200 aproximadamente. Tendo isso em mente, os próximos testes serão feitos considerando apenas 200 épocas de treino.

O próximo passo buscou avaliar a quantidade de neurônios da camada escondida. Para isso, foi repetido a sequência de passos acima (para 200 épocas) variando o tamanho dessa camada no modelo para o valores:
4, 8, 12, 16, 20, 24, 28, 32, 36, 40, 44, 48, 52, 56, 60, 64
O gráfico abaixo mostra os erros de teste e validação para cada rodada:

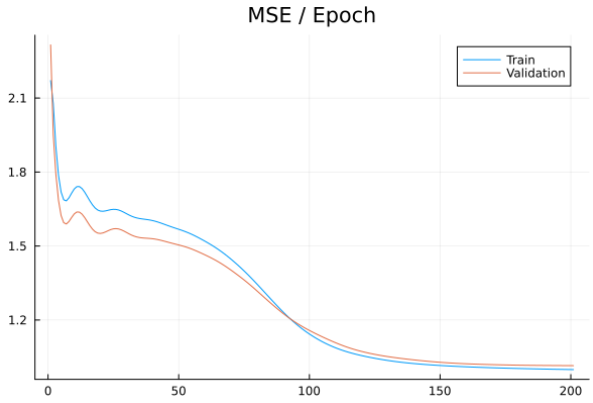


Como podemos perceber, o erro de validação não tem um comportamento de queda (ou aumento) claro pelo tamanho da camada escondida para esse caso, desde que seja acima de um valor mínimo. Porém, seguindo a lógica, foi identificado que o tamanho que apresentou o menor erro de validação foi 28 neurônios.

A seguir, foi feita uma avaliação final com o modelo de 28 neurônios, sendo treinado por 200 épocas, com um termo adicional na função de custo relacionado a regularização. O termo de regularização escolhido foi simplesmente a norma 2 das matrizes que compõem a rede ($W,\ b$ por camada além do input):

$$r = |W_1|_2^2 + |b_1|_2^2 + |W_2|_2^2 + |b_2|_2^2$$

O resultado dessa avaliação mostra um comportamento mais suave e estável de aprendizado da rede, como esperado:



2 Convolutional Neural Networks

Na parte de redes neurais convolucionais, foram utilizados pesos de uma rede pré-treinada para definir um modelo de previsão de dígitos escritos a mão.

Depois da leitura dos pesos e da base e tratamentos necessários, foi feita uma avaliação básica de 10 casos aleatórios:

i	y	$\hat{y}$
1	8	2
2	5	5
3	1	1
4	6	6
5	8	8
6	9	9
7	2	2
8	8	5
9	2	2
10	6	5

Aqui já podemos notar que o modelo tem a capacidade de detectar corretamente o dígito escrito pelo menos para alguns casos, visto que 8 dos 10 casos foram classificados corretamente. Foi feito uma avaliação visual dos casos para entendimento e não foram encontradas características claras na imagens não classificadas corretamente. Após essa avaliação inicial, foi feito o cálculo da acurácia do modelo na amostra de teste definida inicialmente (1000 registros). A acurácia computada foi de $94,4\%$.

Por fim foi aplicada a função de ataque de perturbação em algumas imagens para avaliação. O teste seguiu uma lógica de avaliação de 100 imagens aleatórias pré-definidas após ser aplicada a função de perturbação para diferentes valores de ϵ com o fim de cálculo da acurácia nessa amostra. Os resultados são como a seguir:

ϵ	acc
0	93%
0.01	94%
0.1	96%
1	68%
10	20%

Por fim, percebe-se que o modelo é muito sensível às perturbações aplicadas pela técnica utilizada, principalmente para valores de ϵ altos (≥ 1). Como etapa de melhoria do modelo para evitar essa fraqueza, pode-se adicionar amostras em uma base de treino de refinação com imagens utilizadas no treino inicial após passadas pelas perturbações. Esse passo pode ser repetido múltiplas vezes até perceber-se uma maior robustez à variação (sempre sendo aplicado á versão atualizada do modelo, para computação adequada da matriz Jacobiana).

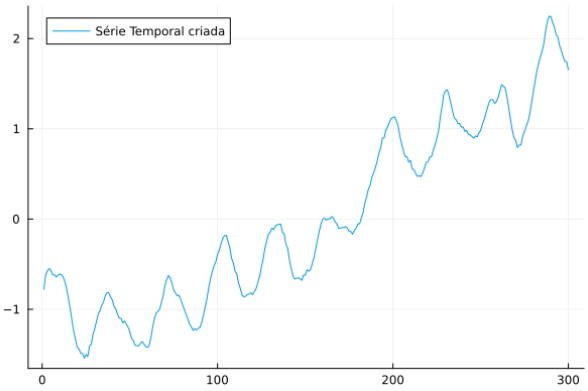
3 Recurrent Neural Networks

Para exercício de redes recorrentes, foi primeiro gerada uma série temporal simulada. A geração seguiu os seguintes passos:

1. Geração de ruído aleatório

2. Aplicação de parâmetros de forma autoregressiva em uma janela de tamanho 5
3. Aplicação da soma acumulada
4. Adição de uma sazonalidade senoidal
5. Normalização

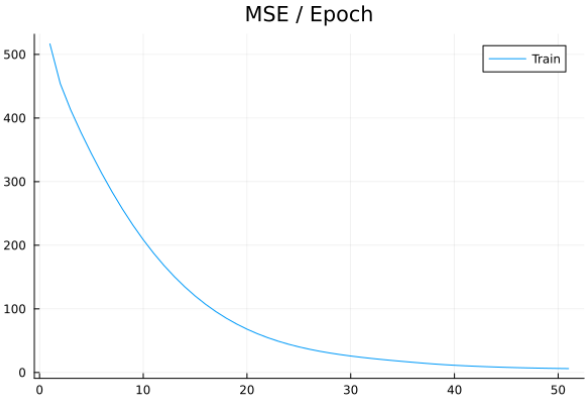
A série resultante (com 300 observações) é a seguinte:



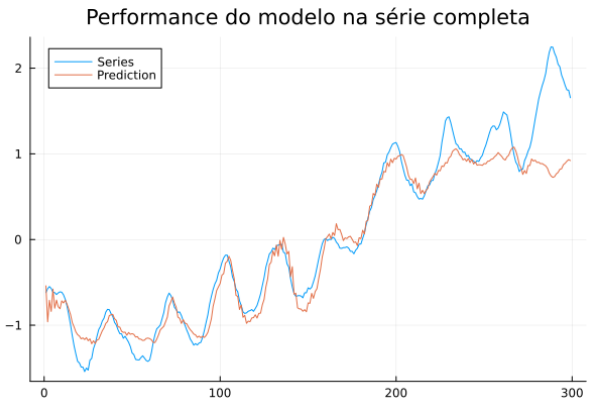
A seguir, foi feita uma separação em conjuntos de treino e teste. O conjunto de treino foi utilizado para treinar uma rede recorrente com a seguinte estrutura:

1. Camada de input: 1
2. Camada recorrente: $1 => 8$, tanh
3. Camada de output (densa): $8 => 1$

Mais uma vez, o otimizador utilizado foi o Adam (taxa de aprendizado = 0.005) e a função de custo utilizada foi o erro total quadrático. O treinamento foi feito por 50 épocas e foi acompanhado o erro de treino:



Após o treinamento, foram feitas previsões para a série completa, considerando tanto o período de treino quanto de teste. O resultado pode ser visto na imagem a seguir:



Como podemos perceber, o modelo foi capaz de aprender alguns comportamentos básicos de série, como sazonalidade, porém não é capaz de captar muito bem os altos e baixos nos dados de treino e desvia do valor real nos dados de teste.