

ENG1456 - Redes Neurais - Trabalho 2

Previsão de Séries Temporais

Aluno: Matheus Carneiro Nogueira - 1810764

Professora: Marlley Velasco

Sumário

1 Compreensão do Problema	2
1.1 Visualize, em forma de gráfico, a dinâmica temporal da série escolhida. A série é adequada para a modelagem usando Redes Neurais? Caso não seja, que técnicas podem ser aplicadas para ajustar o comportamento da série?	2
1.2 No problema escolhido, usaremos uma variável exógena que representa o mês de previsão (i.e. no instante $t+1$). De que forma esta variável pode auxiliar na previsão da série temporal?	4
2 Previsão One-Step	4
2.1 Execute o script para a previsão one-step. Analise o resultado (conjunto de treinamento e teste), usando as métricas RMSE e MAE.	4
2.2 Modifique a técnica de codificação mensal de ‘numérico’ para ‘binário’. Qual a mudança existente na arquitetura da Rede Neural? Analise o resultado (conjunto de treinamento e teste), usando as métricas RMSE e MAE.	6
3 Previsão Multi-Step	6
3.1 Implemente o processo de previsão multi-step	6
3.2 Faça a previsão multi-step para o horizonte de previsão igual a 12 e compare com o resultado da previsão one-step.	7
3.3 Modifique o tamanho da janela de entrada e avalie os resultados	8
3.4 Modifique a topologia da rede para obter um melhor desempenho. Altere seus parâmetros (e.g. número de processadores na camada escondida, tipo de função na camada de saída) e avalie o desempenho em termos das métricas RMSE e MAE.	10
3.5 Implemente a codificação ‘1 of N’ e use-a para modificar a representação da variável ‘mês’. Qual a mudança existente na arquitetura da Rede Neural? Avalie o desempenho em termos das métricas RMSE e MAE.	12

Resumo

Este documento consiste no relatório do trabalho 2 do módulo de Redes Neurais da disciplina ENG1456 da PUC-Rio. Nele será explicada a implementação de

modelos de Redes Neurais MLP para a previsão de uma série temporal do dataset MicroClima2 , disponibilizado pela professora da disciplina. A seções do relatório são definidas de acordo com as perguntas principais que constam no arquivo Guia de Atividades II. Foram consultados os materiais de aula, o livro [1] e outros materiais devidamente referenciados.

1 Compreensão do Problema

- 1.1 Visualize, em forma de gráfico, a dinâmica temporal da série escolhida. A série é adequada para a modelagem usando Redes Neurais? Caso não seja, que técnicas podem ser aplicadas para ajustar o comportamento da série?**

A série utilizada neste trabalho, denominada *microclima2* consiste em 144 temperaturas média mensais, ou seja, temos informação sobre as temperaturas dos últimos 12 anos.

Com o intuito de analisar a dinâmica temporal da série, foi gerado o gráfico da série em si e da decomposição dela, com o intuito de verificar *trending* e *sazonalidade*. As figuras abaixo ilustram esses resultados.

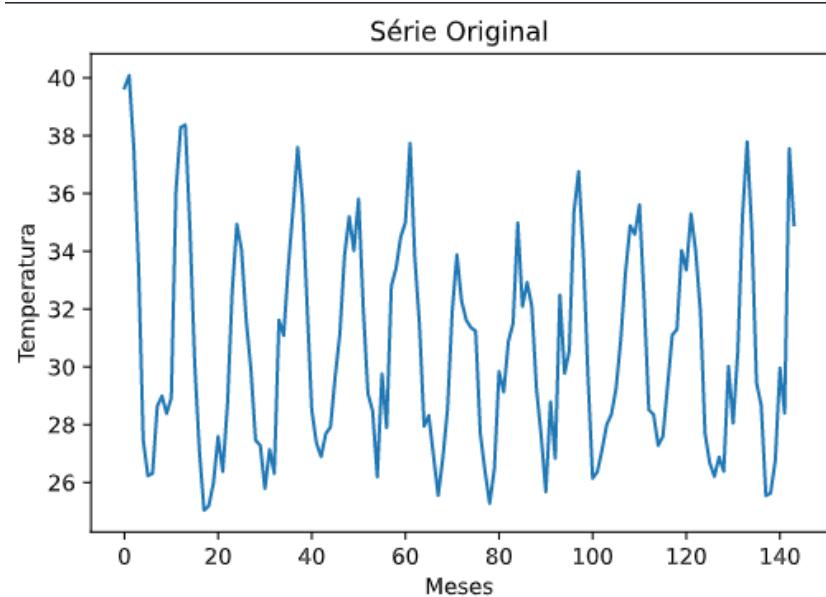


Figura 1: Série Original

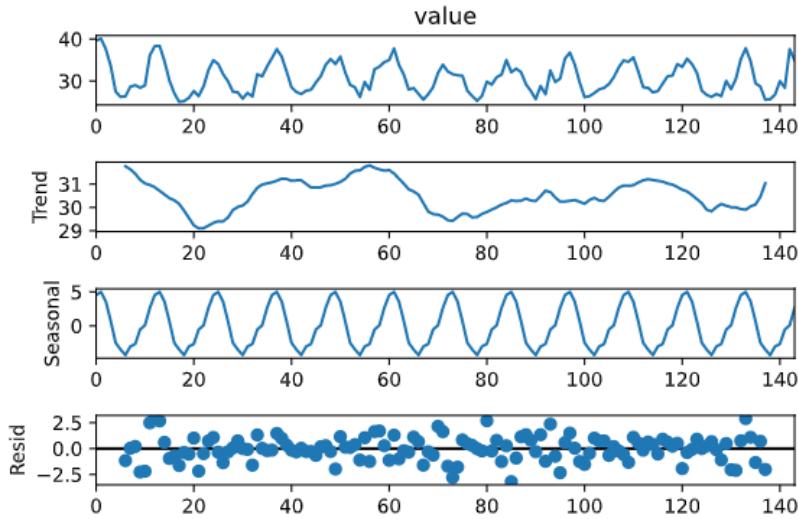


Figura 2: Decomposição da Série Original

Ao analisar a figura 1 notamos que a série possui um perfil estacionário ao longo dos 12 anos. Isso quer dizer que a temperatura média de cada mês do ano a é similar à temperatura média de cada mês no ano $a+1$. Além disso, a figura 2 revela, em seu campo *Trend*, a tendência da série ao longo do tempo. Embora existam oscilações nesse gráfico, não se percebe nenhuma tendência geral da série. Com o intuito de ir além da análise visual, foi executado o *Augmented Dickey-Fuller test* para verificar a probabilidade de existência de uma raiz unitária, que indicaria perfil não estacionário. O resultado é expresso na figura abaixo.

```
ADF Statistic: -3.346082
p-value: 0.013179
Critical Values:
    1%: -3.482
    5%: -2.884
    10%: -2.579
```

Figura 3: ADF Test

Nota-se que o *p-value* é muito pequeno e que o valor de *ADF Statistic* é menor que o valor crítico para 1%. Isso indica que esta série possui probabilidade baixíssima a alto grau de confiança em relação à inexistência de uma raiz unitária. Sendo assim, podemos tratá-la como uma série estacionária. Essa análise é importante pois, caso a série fosse não-estacionária, a rede neural precisaria, além de aprender o andamento temporal da série, aprender também a sua tendência, o que aumentaria a complexidade do modelo. Para corrigir esse problema, poderíamos tornar a série estacionária por meio de uma diferenciação, por exemplo, lembrando, apenas, de voltar aos valores originais ao final.

Como a série em questão possui valores de temperaturas médias mensais ao longo dos anos, é de se esperar que exista uma sazonalidade razoavelmente perceptível. Ambas as

figuras 1 e 2 mostram que essa sazonalidade existe de fato. Com essas imagens em mãos e supondo que o mês 1.0 é Janeiro, podemos inferir que as temperaturas referem-se a um local do hemisfério sul, onde é verão em janeiro, uma vez que as maiores temperaturas encontram-se próximas desse mês. Essa informação (o mês da temperatura) será útil para o treinamento da série, portanto deve constar nos dados de entrada. Como a sazonalidade não oferece problemas para a modelagem com um *MLP*, não precisamos fazer nenhum tipo de correção.

1.2 No problema escolhido, usaremos uma variável exógena que representa o mês de previsão (i.e. no instante $t+1$). De que forma esta variável pode auxiliar na previsão da série temporal?

Como comentado na seção 1.1, a série de temperaturas apresenta sazonalidade anual, isto é, o perfil de evolução da série se repete de ano em ano, o que é de se esperar dada a natureza da série. Desse modo, fornecer o mês do valor de entrada pode auxiliar bastante na qualidade da previsão da série, uma vez que meses como Dezembro a Fevereiro (12 a 3) geralmente apresentam temperaturas médias mais altas, enquanto meses como Maio a Setembro (5 a 9) apresentam temperaturas mais baixas. Ao fornecer essa informação para Rede Neural, ela possuirá mais informações para aprender o perfil de sazonalidade da série, aumentando sua qualidade de previsão.

2 Previsão One-Step

2.1 Execute o script para a previsão one-step. Analise o resultado (conjunto de treinamento e teste), usando as métricas RMSE e MAE.

Para realizar a previsão de um passo à frente, foram testadas diversas configurações de redes neurais, variando a quantidade de neurônios na camada escondida. Os erros *RMSE* e *MAE* de cada uma dessas configurações estão apresentados na tabela a seguir. Vale comentar que, no arquivo *jupyter notebook* enviado junto deste relatório está presente apenas o modelo final escolhido. Além disso, o script definia a métrica *MSE*, então, para obter a métrica desejada, *RMSE* foi calculada a raiz quadrada da *MSE* fornecida.

# neurônios	5	10	15	20	25	30
RMSE	3.007	2.871	3.022	2.584	2.598	2.936
MAE	2.090	2.105	2.434	1.938	1.968	2.343

Tabela 1: Comparação dos Erros para diferentes números de neurônios

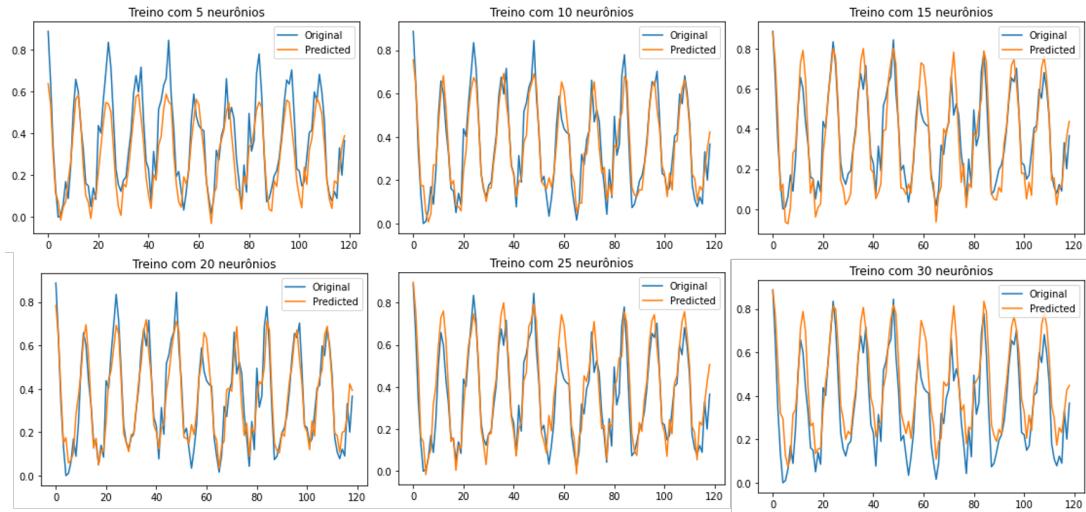


Figura 4: Comparação de Treinamento para diferentes quantidades de neurônios

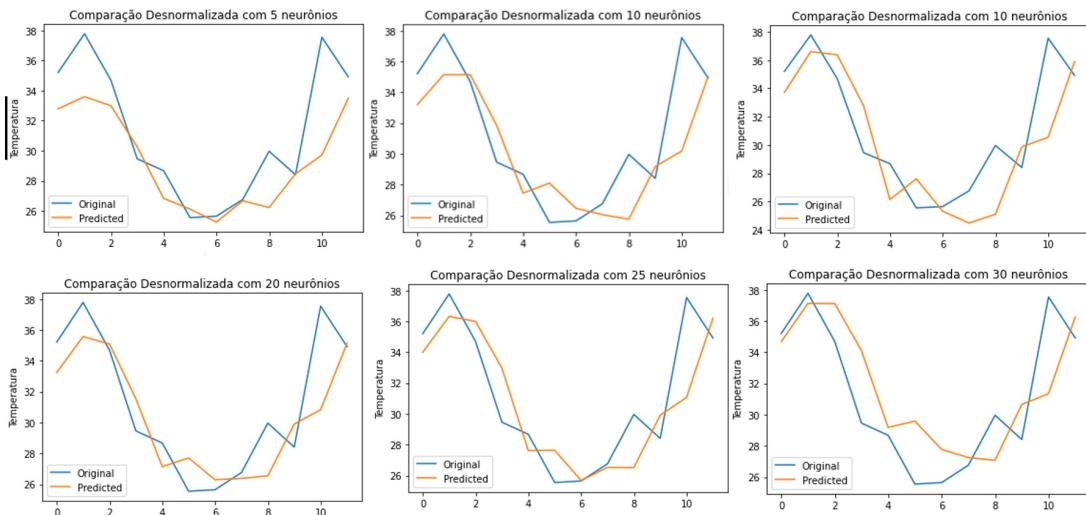


Figura 5: Comparação de testes para diferentes quantidades de neurônios

Com base na análise da tabela 2.1 e das imagens 4 e 5, percebe-se que para valores de 20 e 25 neurônios os erros são muito similares. Sendo assim, foi optado por eleger a rede com 20 neurônios na camada escondida por apresentar os melhores valores para as métricas analisadas e ser um modelo mais simples do que o de 25 neurônios.

Podemos perceber perfeitamente, com as imagens acima, a importância da generalização e da especialização da rede. Note que para 5 neurônios a curva de treino e de teste é muito suave, o que indica que a rede não conseguiu aprender o suficiente por possuir poucos pesos treináveis. Ou seja, não adquiriu especialização. Por outro lado, com 30 neurônio a curva passa a ser desnecessariamente ruidosa devido à quantidade grande de parâmetros treináveis, causando perda de generalização.

A figura abaixo exibe as métricas da rede escolhida.

Erro RMSE = 2.584
Erro MAE = 1.938

Figura 6: Erros da Rede Neural escolhida com 20 neurônios

2.2 Modifique a técnica de codificação mensal de ‘numérico’ para ‘binário’. Qual a mudança existente na arquitetura da Rede Neural? Analise o resultado (conjunto de treinamento e teste), usando as métricas RMSE e MAE.

As redes treinadas até então já estavam com a codificação mensal binária. Sendo assim, foi criado um modelo com exatamente os mesmos parâmetros da rede escolhida na seção 2.1 mas com a codificação numérica para os meses. Ao treinar essa nova rede pela primeira vez, os valores para as métrica *RMSE* e *MAE* foram melhores que aqueles apresentados pela rede com codificação binária. Para ter maior certeza dessa melhora, foram treinadas 6 redes com as mesmas configurações, variando, obviamente, apenas os pesos iniciais ,que são aleatórios. Os erros para as 6 redes encontram-se an tabela abaixo. A primeira coluna é a referência da rede com configuração binária.

#Rede	Binária	1	2	3	4	5	6
RMSE	2.584	2.448	2.276	2.572	2.386	2.587	2.827
MAE	1.938	1.751	1.741	1.910	1.842	1.871	2.104

Tabela 2: Comparação dos erros das redes com codificação binária e numérica

Ao analisar a tabela 2.2, não é possível perceber grande melhora, ou piora, no desempenho da rede neural em relação ao tipo de codificação utilizado. Tentemos encontrar uma explicação para essa consistência.

Como visto na seção 1.1, a série em questão é bem comportada, inclusive em relação ao padrão de sazonalidade. O fato do tipo de codificação da entrada ”mês” não ter influenciado na qualidade da previsão pode ser explicado, justamente, por esse bom comportamento. Independente do tipo de codificação a Rede parece já possuir informações suficientes para fazer a previsão com uma boa qualidade. Além disso, o tipo de codificação também altera a quantidade de entradas da rede neural. No caso numérico há apenas 1 entrada para indicar o mês, enquanto no caso binário são 4 entradas. Essa diferença aumenta o número de pesos a serem treinados e aumenta a complexidade da rede, podendo gerar perda de generalização. No entanto, uma vez que as métricas obtidas não foram muito distintas, isso não ocorreu.

Para as demais seções, foi utilizada a codificação numérica.

3 Previsão Multi-Step

3.1 Implemente o processo de previsão multi-step

A ideia central da previsão multi-step consiste em utilizar, ao invés de valores reais, valores previstos pela rede para ”realimentar” a série. Em detalhes, para prever o valor da série

em $t + 1$, sabendo que utilizamos um *lag* de 12 passos passados, precisamos fornecer os valores de $t - 0$ até $t - 11$ (além de outras possíveis variáveis externas, como o mês). Agora, para prever o valor da série em $t + 2$, precisamos ainda fornecer 12 valores passados, mas sabendo que, desses, apenas 11 valores são reais, enquanto um é previsto. Isso quer dizer que, para prever $t + 2$, precisamos fornecer $t + 1$ até $t - 10$, sendo $t + 1$ fruto de uma previsão. Com o intuito de prever 12 passos à frente, repetimos este processo até $t + 12$.

Para implementar essa previsão, foi desenvolvida a seguinte função:

```
def multi_step(model,x_serie,stepsNum):
    for i in range(stepsNum):
        y_hat=model.predict(x_serie)
        #escorregar todos os valores de x_train[1] a x_train[-1] para a esquerda
        # x_train[4]=x_train[5] ... x_train[15]=x_train[16] e x_train[16]= valor previsto
        for j in range(1,13):
            for k in range(0,x_serie.shape[0]):
                x_serie[k][j]=x_serie[k][j+1]
        x_serie[0][-1]= y_hat[0]
    return y_hat
```

Figura 7: Função para a previsão multi-step

3.2 Faça a previsão multi-step para o horizonte de previsão igual a 12 e compare com o resultado da previsão one-step.

Foi executada a função descrita anteriormente tanto para comparação com o conjunto de treino quanto para o conjunto de teste. As imagens abaixo ilustram essas execuções e a tabela, as métrica *RMSE* e *MAE*:

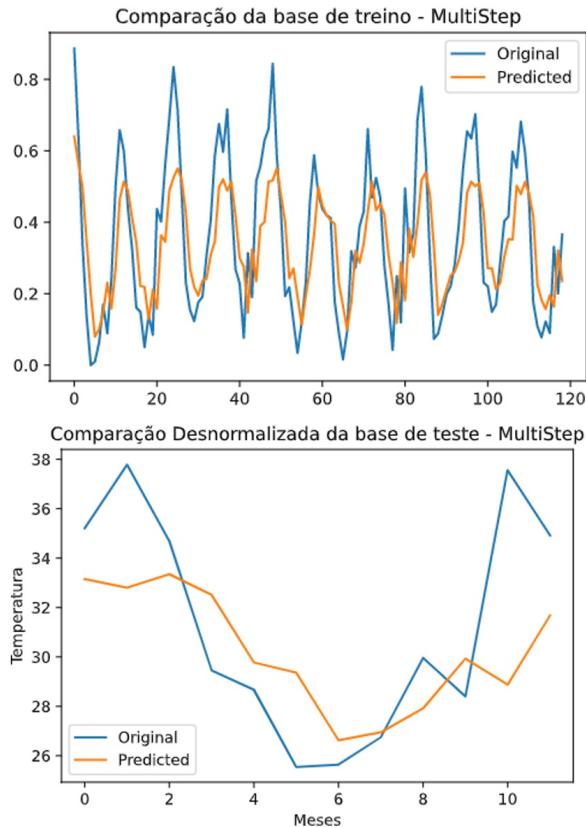


Figura 8: Execução da Previsão Multi-Step

#Rede	Multi-Step	One-Step
RMSE	3.528	2.584
MAE	2.753	1.938

Tabela 3: Erros para a previsão multi step

Pode-se perceber que o resultado da previsão multi-step foi consideravelmente pior que o da one-step. Um motivo que pode explicar essa piora, é o fato da previsão multi-step funcionar com a previsão de valores futuros com base em dados previstos, não apenas reais. Isso faz com que ocorra uma acumulação de erros ao longo da previsão. Sendo assim, não é de se estranhar que, embora entre os meses 2 e 9 os valores previstos e reais sejam similares, como mostra a figura 8, a previsão para os 3 últimos meses seja ruim. Outra característica que vale ser comentada é o perfil mais suave dos valores previstos em relação aos valores reais. Nota-se, também pela figura 8, que os valores previstos para o treinamento nunca alcançam a temperatura máxima nem a mínima.

Dito isso, ainda existe a possibilidade de a perda de qualidade observada ser fruto da má implementação da previsão multi-step. Embora a lógica explicada na seção 3.1 esteja correta, erros de programação podem ter sido cometidos.

Ao longo das próximas seções serão variados parâmetros da rede para tentar melhorar seu desempenho.

3.3 Modifique o tamanho da janela de entrada e avalie os resultados

Primeiramente, é necessário apontar que todas as janelas testadas são sequenciais, uma vez que a função de transformação de dados fornecida gera apenas janelas deste tipo. Ao modificar o tamanho da janela de entrada, o que estamos, de fato, fazendo, é alterando a quantidade de meses passados que a rede terá conhecimento para prever a temperatura do próximo mês. Não necessariamente fornecer mais meses para a rede influenciará positivamente a precisão. Como a série em questão é de temperaturas de um dado local, e apresenta um perfil sazonal forte, eventualmente fornecer apenas os últimos 6 meses, ou 4 meses seria o suficiente para a rede aprender o padrão de temperaturas. Uma possível melhor escolha de janela seria fornecer o último trimestre ou quadrimestre e a temperatura do mesmo mês da previsão, mas de um ano antes. No entanto, esta janela não foi testada por motivos já explicados.

Outra influência do tamanho da janela é o dilema entre especialização e generalização. Como sabemos, maior janela equivale a mais dados de entrada que, por sua vez, equivale a mais pesos treináveis, maior complexidade da rede que pode levar a uma hiper especialização e perda de generalização. O raciocínio contrário também vale, menor janela, menos entradas, menos pesos, rede mais simples e risco de falta de especialização.

Vejamos os resultados obtidos para os valores de janelas abaixo para o **One Step Forecast**:

#Janela	12	10	8	6	4	2	1
RMSE	2.584	2.906	2.873	2.557	2.546	3.359	2.854
MAE	1.938	2.238	2.305	2.007	1.926	2.73	1.975

Tabela 4: Erros para redes de diferentes tamanhos de janela

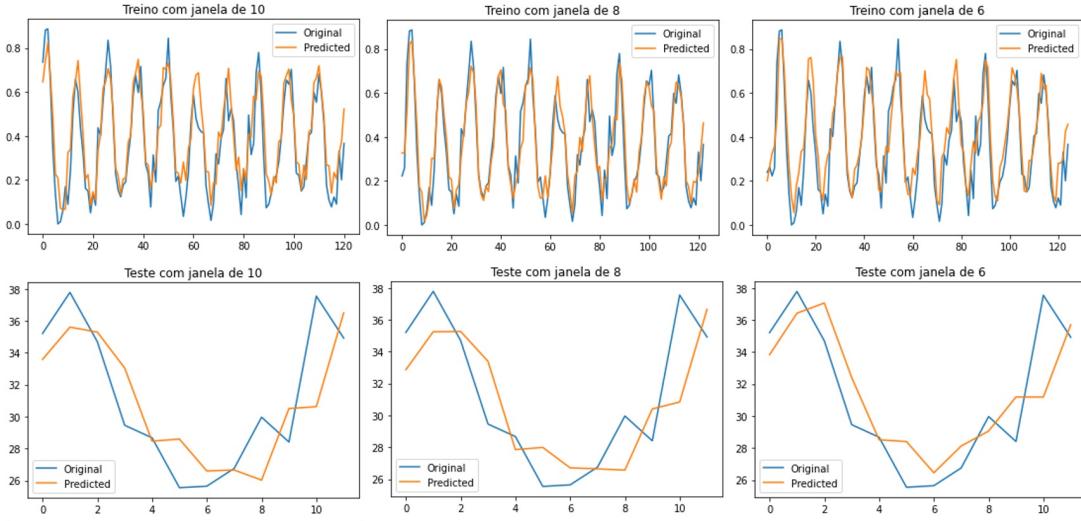


Figura 9: Comparações de treino e teste para janelas 10, 8 e 6

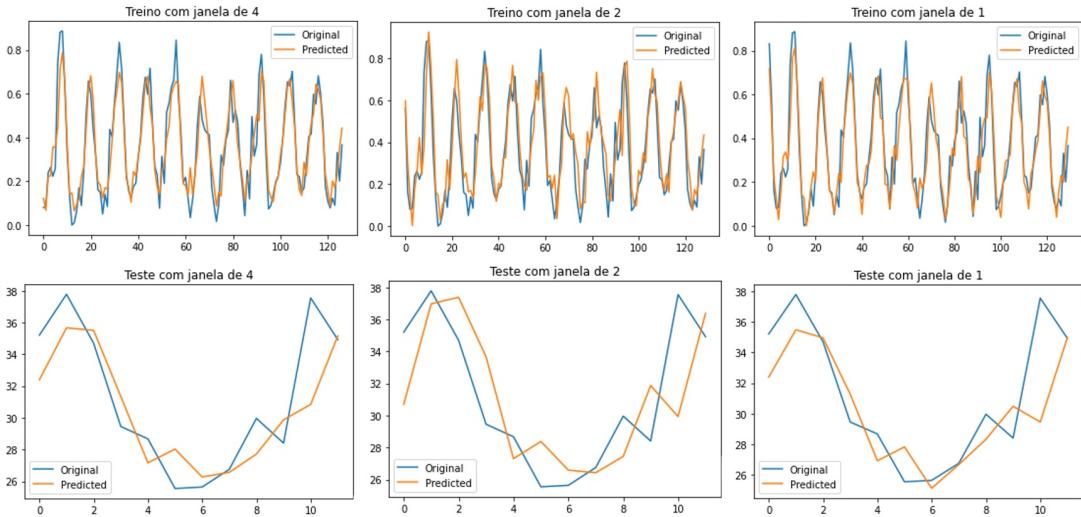


Figura 10: Comparações de treino e teste para janelas 4, 2 e 1

Percebe-se que os melhores resultados em termos dos erros analisados são para 12, 6 e 4 janelas. Assim como havíamos suposto e comentado, conhecer apenas o trimestre ou semestre passado, dada a sazonalidade e bom comportamento da série, mostra-se tão bom quanto conhecer o ano anterior inteiro. Sendo assim, como redes mais simples tendem a ser mais interessantes, a rede com janela de 4 meses seria a melhor escolha dentre as apresentadas.

Vejamos agora os resultados para o **Multi Step Forecast**:

#Janela	12	10	8	6	4	2	1
RMSE	3.528	3.897	4.023	3.767	3.297	3.362	6.736
MAE	2.753	2.971	3.235	3.222	2.516	2.607	4.042

Tabela 5: Erros para redes de diferentes tamanhos de janela

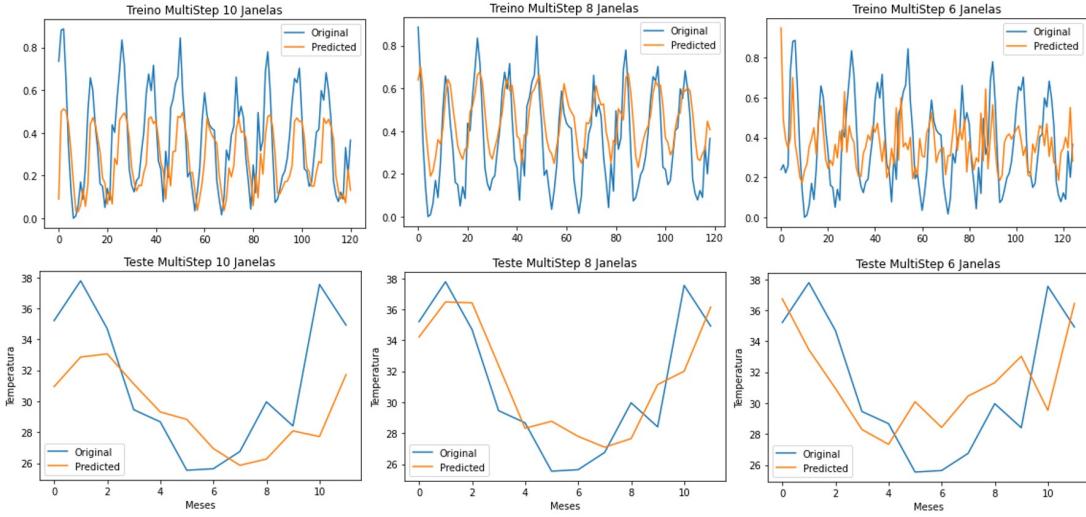


Figura 11: Comparações de treino e teste para janelas 10, 8 e 6

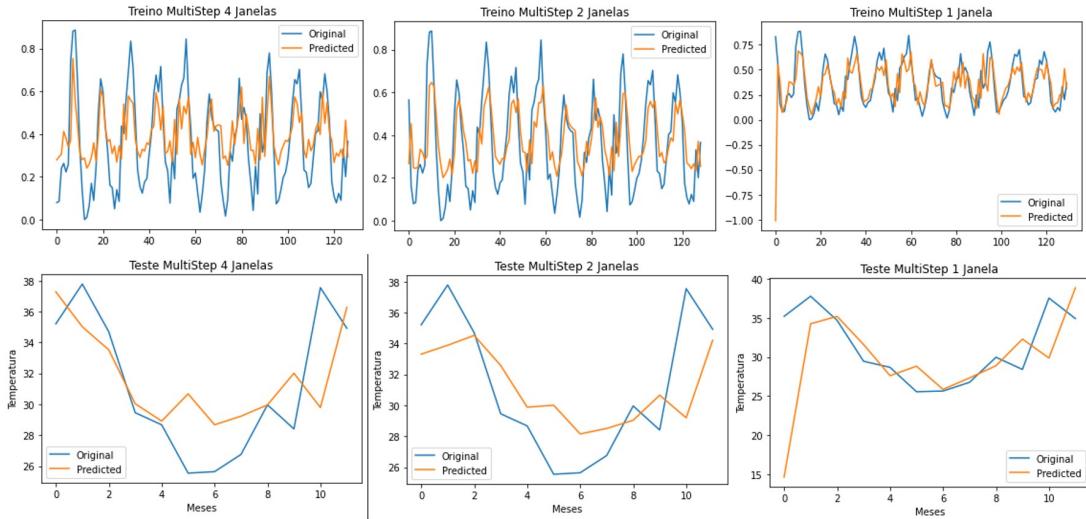


Figura 12: Comparações de treino e teste para janelas 4, 2 e 1

Resultados similares foram obtidos. Novamente a rede com janela de tamanho 4 mostrou-se superior à rede com janela de tamanho 12. Desse modo, as mesmas conclusões comentadas para o caso *one-step* valem para o *multi-step*.

3.4 Modifique a topologia da rede para obter um melhor desempenho. Altere seus parâmetros (e.g. número de processadores na camada escondida, tipo de função na camada de saída) e avalie o desempenho em termos das métricas RMSE e MAE.

Serão alterados 2 parâmetros da topologia da rede de forma sequencial. São eles o número de neurônios na camada escondida e o tipo de função de ativação no neurônio da camada de saída. Por forma sequencial quero dizer que primeiro serão testados diferentes quantidades

#Neurônios	20	40	30	10	5
RMSE	3.528	3.693	3.762	4.511	4.046
MAE	2.753	3.157	3.087	3.776	3.327

Tabela 6: Erros para diferentes quantidades de neurônios

de neurônios e, após selecionar a melhor quantidade n_{otima} serão testadas redes com outras funções de ativação mas com quantidade de neurônios igual a n_{otima} .

Vejamos os resultados para a alteração do número de neurônios:

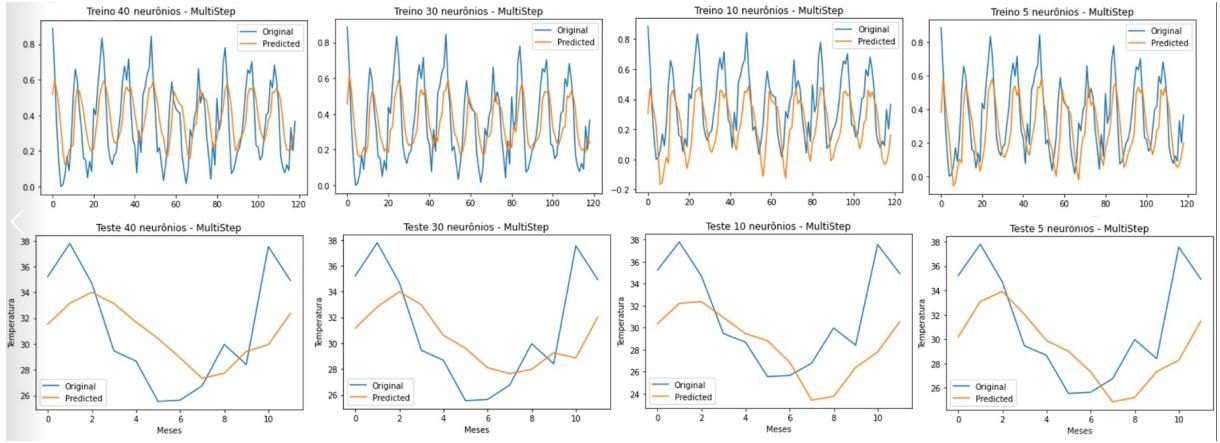


Figura 13: Conjuntos de treino e teste para diferentes quantidades de neurônios

Não é surpresa o fato de a rede com 20 neurônios ter apresentado o melhor resultado com base nas métricas $RMSE$ e MAE , uma vez que, na seção de codificação binária X numérica redes de diferentes quantidades de neurônios na camada escondida e, assim como agora, 20 foi o número com melhor resultado.

Ao analisar a figura 13 não percebemos padrões de grande perda de generalização ou de falta de especialização entre as diferentes quantidades. É provável que, para verificar esses erros, deveriam ser variadas as épocas de treinamento em conjunto com a quantidade de neurônios.

Como estamos analisando apenas as métricas $RMSE$ e MAE , não há nenhum outro comentário a ser feito. Passemos para a variação da função de ativação da camada de saída. Foram testadas as funções *sigmoid*, *relu*, *softplus* e *linear*, sendo esta última a que foi usada até então.

Função	linear	softplus	sigmoid	relu
RMSE	3.528	3.661	3.665	7.451
MAE	2.753	2.774	2.746	5.901

Tabela 7: Erros para diferentes funções de ativação

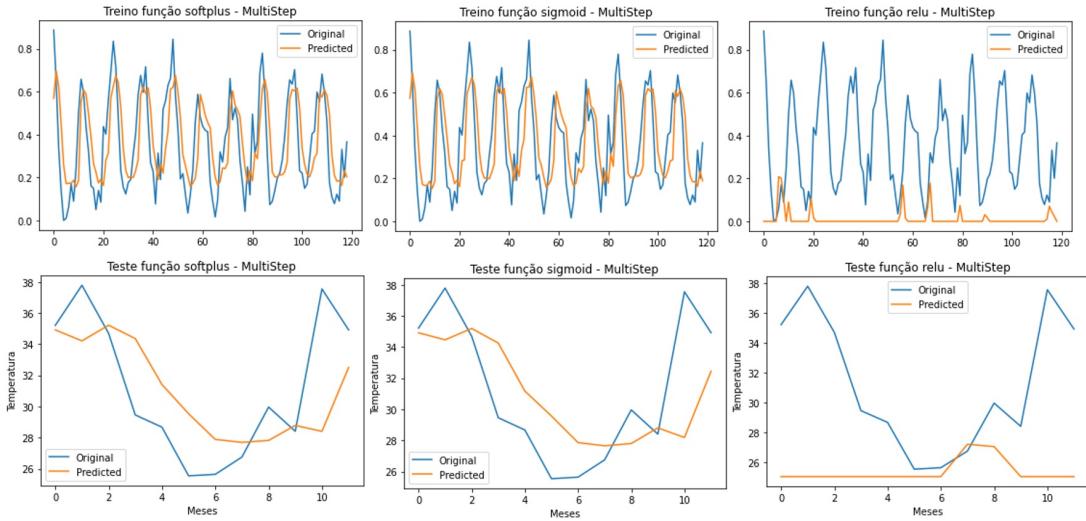


Figura 14: Treinamento e teste para diferentes funções de ativação

Percebe-se resultados similares para as 3 primeiras funções de ativação do neurônio na camada de saída, mas muito distinto, e muito pior, para a função *relu*. O motivo para essa grande diferença para a função *Relu* é desconhecido por mim.

3.5 Implemente a codificação ‘1 of N’ e use-a para modificar a representação da variável ‘mês’. Qual a mudança existente na arquitetura da Rede Neural? Avalie o desempenho em termos das métricas RMSE e MAE.

A codificação *1 of N* se refere a referenciar cada um dos 12 meses de uma forma binária distinta com 12 entradas para a rede. Isso quer dizer que, ao invés de representar os meses de forma numérica [112] ou de forma binária [00001100], representamos os cada mês como uma sequencia de 12 bits sendo 11 deles 0 e apenas 1 valendo 1. O mês 4, por exemplo, pode ser representado como 000000001000.

Embora o ideal fosse implementar essa codificação dentro da função *transform_data()*, não fui capaz de fazê-la. Sendo assim, implementei a codificação de forma braçal e exaustiva, mas o resultado foi o mesmo da solução ideal.

Ao implementar essa codificação, estamos também aumentando o número de entradas fornecidas para a rede neural. As possíveis implicações desse fato já foram comentadas anteriormente. Além disso, como as novas colunas da tabela de entrada fornecida à rede possuem agora diversos valores similares de zeros e uns, talvez o aprendizado seja prejudicado.

Vejamos os resultados dos erros para a codificação *1 of N* para a previsão one step e multi step:

Codificação	Numérica		1 of N	
	OneStep	MultiStep	OneStep	MultiStep
RMSE	2.584	3.528	2.487	3.759
MAE	1.938	2.753	1.927	2.79

Tabela 8: Erros para codificação numérica e 1 of N comparados por tipo de previsão

Nota-se que, para a previsão one-step, o resultado da codificação *1 of N* apresentou erros minimamente menores que a codificação numérica. Para a previsão *multi-step*, por sua vez, o resultado foi um pouco pior para a *1 of N* em relação à numérica. Podemos concluir que, embora o número de entradas seja bem maior, uma vez que uma coluna para mês se tornou 12 colunas, a simplicidade dos dados, um ou zero, pode ter freado o aumento de complexidade. Como os resultados foram muito próximos, não se pode afirmar categoricamente uma vantagem clara em utilizar um dos dois tipos de codificação.

Referências

- [1] Ivan Nunes da Silva, Danilo Hernane Spatti, Rogerio Andrade Flauzino, Luisa Helena Bartocci Liboni, and Silas Franco dos Reis Alves. *Artificial Neural Networks*. Springer, 08 2016.