UNIVERSIDADE​ ​FEDERAL​ ​DO​ ​RIO​ ​GRANDE​ ​DO​ ​SUL (UFRGS)

ESCOLA DE ENGENHARIA

DEPARTAMENTO​ ​DE​ ​ENGENHARIA​ ​ELÉTRICA (DELET)

TÓPICOS ESPECIAIS EM INSTRUMENTAÇÃO (ENG04019)

MATHEUS QUEVEDO SIVELLI

**TRABALHO​ 6 – RADIAL BASIS FUNCTION NEURAL NETWORK**

Porto alegre

2020

SUMÁRIO

[1 INTRODUÇÃO 4](#_Toc43661847)

[2. EXERCÍCIOS 5](#_Toc43661848)

[2.1 INTRODUÇÃO TÉORICA E APOIO PROGRAMÁTICO 5](#_Toc43661849)

[2.2 PROBLEMA 1 - TIPPING PROBLEM 8](#_Toc43661850)

[2.3 PROBLEMA 2 - TIPPING PROBLEM 12](#_Toc43661851)

[2.4 PROBLEMA 3 – TIPPING PROBLEM 16](#_Toc43661852)

[2.5 PROBLEMA 4 – TIPPING PROBLEM 19](#_Toc43661853)

[2.6 COMPARAÇÃO 20](#_Toc43661854)

[2.7 PROBLEMA – AVALIÇÃO DE VENDAS ONLINE 21](#_Toc43661855)

[3. CONCLUSÃO 29](#_Toc43661856)

[4. REFERÊNCIAS 30](#_Toc43661857)

LISTA DE FIGURAS

[**Figura 1 -** Esquema simplificado de um sistema fuzzy. 5](#_Toc43661419)

[**Figura 2 -** Funções de pertinência das variáveis linguísticas 11](#_Toc43661420)

[**Figura 3 -** Resposta do sistema variando os valores de entrada 12](#_Toc43661421)

[**Figura 4 -** Função de pertinência das variáveis linguísticas 15](#_Toc43661422)

[**Figura 5 -** Resultado das diferentes entradas para uma função trapezoidal 16](#_Toc43661423)

[**Figura 6 -** Resultado dos diferentes pontos de entrada 18](#_Toc43661424)

[**Figura 7 -** Resultado das diferentes possibilidades de entrada 19](#_Toc43661425)

[**Figura 8 -** Funções de pertinência 21](#_Toc43661426)

[**Figura 9 -** Resultado com seis regras 22](#_Toc43661427)

[**Figura 10 -** Resultado com todas regras 23](#_Toc43661428)

[**Figura 11 -** Resultado com cinco regras 24](#_Toc43661429)

[**Figura 12 -** Resultado com todas as regras 24](#_Toc43661430)

1 INTRODUÇÃO

Inteligência artificial e o aprendizado de máquina são usualmente definidos como o futuro da humanidade. Dentro desse pilar, o aprendizado profundo – deep learning, em inglês – tem se destacado e criado espaço em diversas aplicações como reconhecimento facial, visão computacional, processamento natural de linguagem e outras.

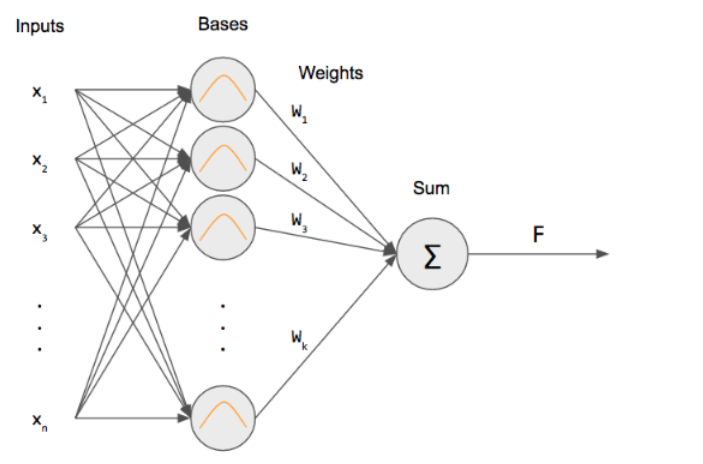
As redes neurais são os principais elementos entre a vertente de deep learning e sua abordagem mais clássica como as famosas MLP são excelente métodos para utilizarmos em problemas de classificação, porém não apresentam o mesmo nível de performance em problemas de regressão, e para solucionar esse problema existem as redes neurais de base radial, que fazem uso da distância dos centros – variáveis definidas por algum método, seja aleatório ou via algoritmos de clusterização – e seus respectivos valores de entrada.

Por fim, as RBF’s podem ser definidas como um caso especial das MLP’s pois possuem, em sua essência, apenas uma camada oculta e com pesos apenas na camada de saída.

2. EXERCÍCIOS

2.1 INTRODUÇÃO TÉORICA

As redes neurais de base radial podem ser classificadas como um caso específico das MLP’s contendo apenas uma camada oculta e a na camada de saída contendo apenas um neurônio de saída. Além disso, as RBF’s não apresentam biases em sua camada escondida e não utilizam o conceito de pesos, por outro lado, observamos o uso de centros, que são medidas arbitrárias que podem ser definidas aleatoriamente ou por algum algoritmo de agrupamento como o K-Means, os centros são utilizados para medir a distância entre o valor de entrada e o centro respectivo do neurônio. Por fim, os pesos são apenas aplicados na camada de saída. A figura 1 representa a arquitetura usual de uma RBF.



Como podemos analisar, as funções de ativação da RBF se baseiam em funções gaussianas, também conhecida como distribuição normal.

2.2 PROBLEMA 1 – INTERPOLAÇÃO

Neste problema são criados 13 pontos aleatórios com seus respectivos valores e é utilizado uma RBFNN para fazer a interpolação desses pontos, que é encontrar a curva que melhor representa o conjunto. Para que o problema seja de interpolação, devemos utilizar os centros exatamente iguais aos pontos de entradas, fazendo com que cada entrada se conecte em cada neurônio da camada oculta e cada neurônio possua um centro que seja equivalente a uma entrada.

Aplicando a técnica de interpolação, encontramos uma matriz quadrada que possuem exatamente a saída da camada oculta. Com a inversa da matriz quadrada e a multiplicação da saída, conseguimos obter exatamente os pesos que representam a aproximação da função, caracterizando assim, a interpolação.

Utilizamos o código apresentado na seção 2.1 e aplicamos a seguinte técnica para provar as constatações acima. O código e resultados são obtidos abaixo.

inputs = np.array([[-2,-2], [-2,-1], [-2,0], [-1,-2], [-1,-1], [-1,0], [0,0], [1,0], [1,1], [1,2], [2,0], [2,1], [2,2]])

outputs = np.array([-5, -3, 2, -1, 2, -4, -6, 2, 10, 7, -8,  4, 2])

num\_outputs = np.size(outputs[0])

num\_inputs = np.size(inputs[0])

num\_neurons = np.array([13, num\_outputs])

num\_layers = len(num\_neurons)

model = RBFNeuralNetwork(num\_inputs, num\_outputs, num\_neurons, ["gaussian", "linear"], epochs = 5, learning\_rate = 0.1, mode = 'sequential')

model.set\_centers(inputs)

model.set\_weights\_interpolation\_matrix(inputs, outputs)

model.calc\_output(inputs[1])

for i in range (0,len(inputs)):

    print("real output: %f \t predicted output: %f" %(outputs[i], model.calc\_output(inputs[i])))

from mpl\_toolkits.mplot3d import Axes3D

from matplotlib import cm

x0\_sampled = np.arange(-2, 2, 0.1)

x1\_sampled = np.arange(-2, 2, 0.1)

x0, x1 = np.meshgrid(x0\_sampled, x1\_sampled)

z = np.zeros\_like(x0)

for i, x0i in enumerate(x0\_sampled):

    for j, x1i in enumerate(x1\_sampled):

        z[i,j] = model.calc\_output([x0i, x1i])

fig = plt.figure()

ax = fig.add\_subplot(111, projection = '3d')

ax.set\_xlabel("x0")

ax.set\_ylabel("x1")

ax.set\_zlabel("output")

surf = ax.plot\_surface(x0, x1, z, rstride = 1, cstride = 1, cmap = 'viridis', linewidth = 0.1)

x0s = inputs[:,0]

x1s = inputs[:,1]

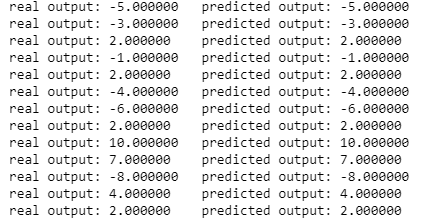
zs = outputs

ax.scatter(x0s, x1s, zs, marker = 'o', c = 'gray', s = 50, depthshade = 0)

plt.show()

A saída numérica da rede é apresentada na figura 1.

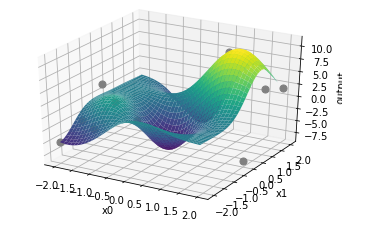
Figura 1 - Saída da RBF



Fonte: Elaboração própria

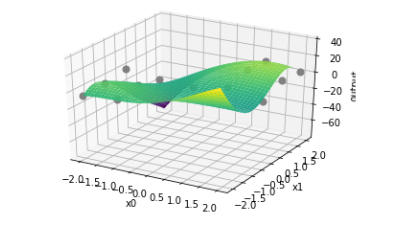
Os resultados gráficos são apresentados abaixo.

Figura 2 - Saída com a largura da função radial = 1



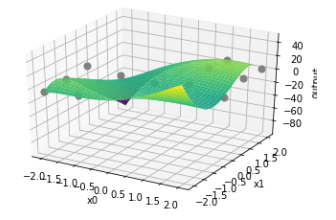
Fonte: Elaboração própria

Figura 3 - Saída com a largura da função radial = 5



Fonte: Elaboração própria

Figura 4 - Saída com a largura da função radial = 10



Fonte: Elaboração própria

Analisando os diferentes valores de sigma, conseguimos perceber que quanto menor a largura aplica na função radial, o comportamento é mais oscilatório e, portanto, a função tende a contemplar muito mais pontos do conjunto que está sendo aproximado a uma função. Já quando aumentamos essa largura, conseguimos perceber que o efeito oscilatório não aparece mais e, consequentemente, tende a não contemplar todos os pontos que estão sendo considerados no problema.

Esse é um clássico caso de otimização de parâmetros, idealmente, a largura da função radial deve ser extraída de um algoritmo não supervisionado de categorização como o ***k-means***.

Por fim, conseguimos extrapolar a ideia até a generalização dessa função, uma largura de base radial específica ao conjunto de dados, pode afetar pequena pode afetar diretamente o desempenho da rede em uma eventual generalização.

2.3 PROBLEMA 2 – APRENDIZADO DE MÁQUINA

Diferente do problema de interpolação, nessa abordagem usaremos técnicas de aprendizado de máquina para determinar os centros e os respectivos pesos entre a saída da camada oculta e a camada de saída. Na seção 2.2, foi preciso utilizar o número de neurônios na camada oculta igual a quantidade de valores na camada de entrada e os centros foram definidos iguais aos valores de entrada. Já nesta abordagem, usaremos um algoritmo não supervisionado, o kmeans, para encontrar os centros dos clusters que melhor representam o nosso conjunto de dados e um número inferior de neurônios na camada oculta.

Como a RBF clássica é uma rede neural feed forward e não possui pesos entre a camada de entrada e a camada oculta, não foi preciso implementar o backpropagation em sua completude. Utilizamos a técnica de backpropagation para atualizar os pesos até a camada anterior a camada de saída, a camada oculta. A implementação do backpropagation é apresentada abaixo.

Código do backpropagation

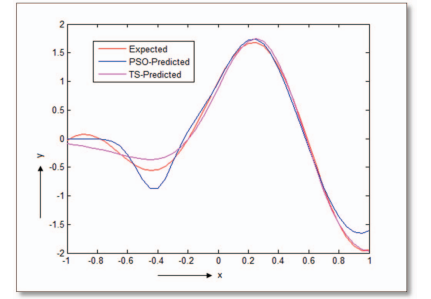
Aplicamos a técnica em um problema clássico de regressão. O código e resultado são apresentados abaixo.

2.4 PROBLEMA 3 – ARTIGO I

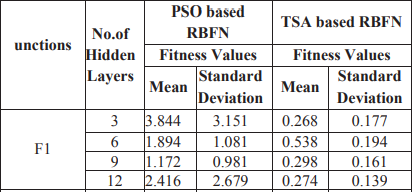
As RBFNN, radial basis functions neural network, não diferem de qualquer outro caso de inteligência artificial e grande partes de seus segredos estão no ato de otimização dos hiperparametros da rede. Portanto, o artigo em questão testa uma abordagem mista, utilizando o algoritmo ***tree seed algorithm,*** algoritmo baseado em 'populações' e idealmente usado para problemas de otimização, para determinar os parâmetros como centros, largura da função gaussiana e os pesos entre a camada de saída e a camada oculta.

O autor inicia o mesmo número de populações do TSA (tree seed algorithm) com a mesma quantidade de camadas ocultas da RBFNN, após isso, é definido o range máximo dos hiperparametros que serão otimizados, como largura da função radial, centros, pesos e inicia todos os valores. Com os hiperparametros inicializados, é determinado o valor da árvore e das folhas, que aplicando o processo de fit, neste resultado é determinado se existe convergência. Esse processo é repetido até encontrar a condição de terminação, depois é replicado para diferentes valores até que se encontre os melhores hiperparametros para o problema.

Para apurar os resultados é utilizado uma outra técnica de otimização de parâmetros, o ***particle swarm optimization*** e o TSA e fazendo a aproximação de uma função. A figura a baixo é a regressão da função com os dois métodos de otimização.



Como podemos perceber, o TSA teve uma performance melhor na regressão da respectiva função. A figura 2 evidencia ainda mais a diferença entre os dois métodos demonstrando os resultados numéricos de cada abordagem.



Por fim, conseguimos analisar tanto graficamente e numericamente que o TSA se mostrou um recurso melhor para otimizar problemas de aproximação de funções do que o outro método comparado.

[https://sci-hub.tw/https://ieeexplore.ieee.org/document/7530267](https://sci-hub.tw/https:/ieeexplore.ieee.org/document/7530267)

2.5 PROBLEMA 4 - ARTIGO II

O problema de encontrar raízes polinomiais e aproximar funções são um ramo importante da matemática e tecnologia, para isto, o autor propõe um uso de RBFNN para abordar esse problema. As RBFNN são redes neurais feed fowards que possuem, em sua maioria, apenas uma camada oculta. Nesta camada oculta a função de ativação é a função gaussiana e não possui pesos associados ao núcleo de processamento do neurônio, este conceito é tomado pelos centros, valor que desempenha um papel semelhante aos pesos nas MLPS. Os centros são utilizados para medir, usualmente, a distância euclideana entre os valores de entrada e os centros. Dentro deste contexto, o autor trás abordagens modificadas para os problemas de otimização e para as funções de ativação.

A metodologia de implementação do autor se resume em uma abordagem mista entre o método clássico gaussiano e o método de Newton-Rapshon, técnica de estimar numericamente raízes de funções, aplicado nas funções de ativação, gradient descent e como na maneira de selecionar os centros.

Basicamente, o autor inicializa os parâmetros da rede sendo maiores que zero e atribui uma tolerância de erro. Aplica-se o grandient decent modificado para encontrar os valores otimizados até que a tolerância seja maior que o erro.

O autor ainda compara a abordagem de utilizar apenas uma RBFNN modificada com o uso delas em paralelo, aplicando exatamente a mesma função que deve ser solucionada com a mesma tolerância a ser permitida. Os resultados apresentam que as redes em paralelo foram 4 vezes mais velozes que apenas uma RBFNN e ainda apresentando um resultado mais assertivo.

Por fim, concluímos que a abordagem das RBFNN modificadas, em paralelo especialmente, podem ser uma alternativa interessante para a aproximação de funções não lineares, tendo registrado o erro muito próximo de zero.

[https://sci-hub.tw/https://ieeexplore.ieee.org/document/7759938](https://sci-hub.tw/https:/ieeexplore.ieee.org/document/7759938)

3. CONCLUSÃO

Como podemos perceber ao decorrer deste trabalho, as redes neurais com funções de base radial continuam sendo um ramo a ser explorado dentro da ciência moderna, possuindo artigos recentes que tratam sobre os principais problemas em algoritmos de machine learning, a otimização.

Além disso, é possível ver o importantíssimo papel que as RBFNN desempenham no ramo matemático em aproximar funções complexas, tendo um desempenho invejável quando otimizado da maneira correta.

4. REFERÊNCIAS

<https://pythonmachinelearning.pro/using-neural-networks-for-regression-radial-basis-function-networks/#:~:text=Radial%20Basis%20Function%20Networks%20(RBF,using%20many%20Gaussians%2Fbell%20curves.>