Student task: Interpolation with Neural Networks

Rajasekar Sankar
28.09.2022

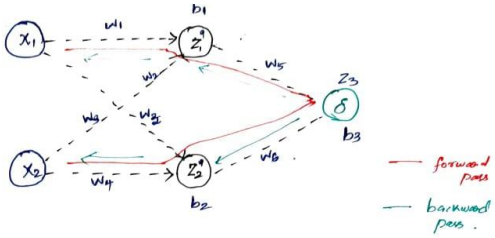
Solutions

1. Theoretical Questions

1. How is the forward pass of a neural network defined?

- In forward pass, values of the output layers are calculated using the input data by traversing through all the neurons from first to the last layer.
- Weighted inputs (z) and respective outputs (a) of the hidden layers will be calculated from input data (x) and weights (w)
- Then, output node (y) will be calculated
- The final step in the forward pass is to compute the loss function (δ), loss function is calculated from the output values.

- After doing forward pass, back propagation will be carried out to update the weights.



x_1, x_2 → input nodes.

$w_1, w_2 \dots w_6$ → weights of the intermediate layers.

$b_1 \dots b_3$ → bias function.

δ → output node value error.

a_1, a_2 → output value of intermediate nodes.

y → output node value.

Example problem,

$x_1 = 0.35$, $x_2 = 0.9$; $w_1 = 0.1$, $w_2 = 0.8$; $w_3 = 0.4$, $w_4 = 0.6$.

$w_5 = 0.3$, $w_6 = 0.9$, $t = 0.5$, $b = 0$., sigmoid function, (σ)

$$\begin{bmatrix} z_1 \\ z_2 \end{bmatrix} = \begin{bmatrix} w_1 & w_2 & b_1 \\ w_3 & w_4 & b_2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ 1 \end{bmatrix}$$

$$= \begin{bmatrix} 0.1 & 0.8 & 0 \\ 0.4 & 0.6 & 0 \end{bmatrix} \begin{bmatrix} 0.35 \\ 0.9 \\ 1 \end{bmatrix}$$

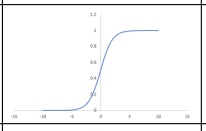
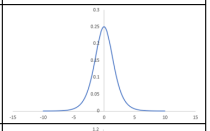
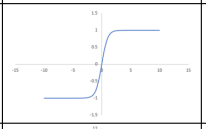
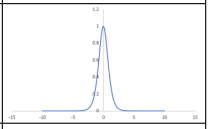
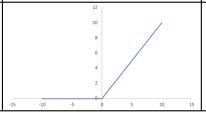
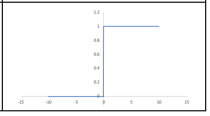
$$\begin{bmatrix} z_1 \\ z_2 \end{bmatrix} = \begin{bmatrix} 0.76 \\ 0.68 \end{bmatrix} \Rightarrow \begin{bmatrix} a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} \sigma(z_1) \\ \sigma(z_2) \end{bmatrix} = \begin{bmatrix} 0.68 \\ 0.66 \end{bmatrix}$$

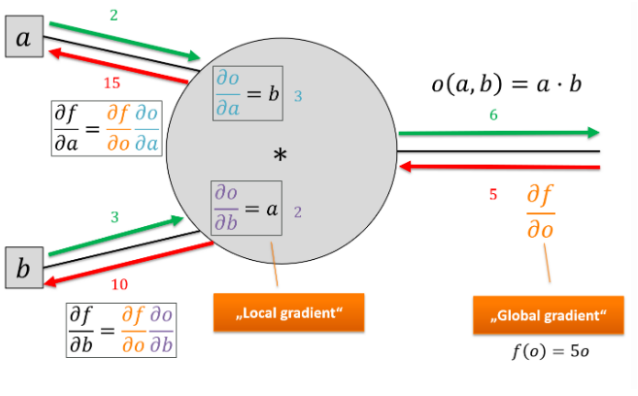
Now, $z_3 = 0.8$, $y = 0.69$

loss function $\delta = \dots \Rightarrow \delta = 0.04$

2. Let u be a fully connected neural network consisting of 4 layers with ReLu activation function approximating the sine function sin(x). Is it possible to compute the second order derivative with respect to x of the network by auto differentiation and obtain the correct derivative, i.e., the approximation of −sin(x). If not, why?

- No, it is not possible to compute the second order derivative for sin(x) function by auto-differentiation using 4 layers with the ReLu activation function.
- Sigmoid function has a similar shape to sin function, so it will work out. But ReLu is a linear function, so it requires a bigger network for approximation of sin function.

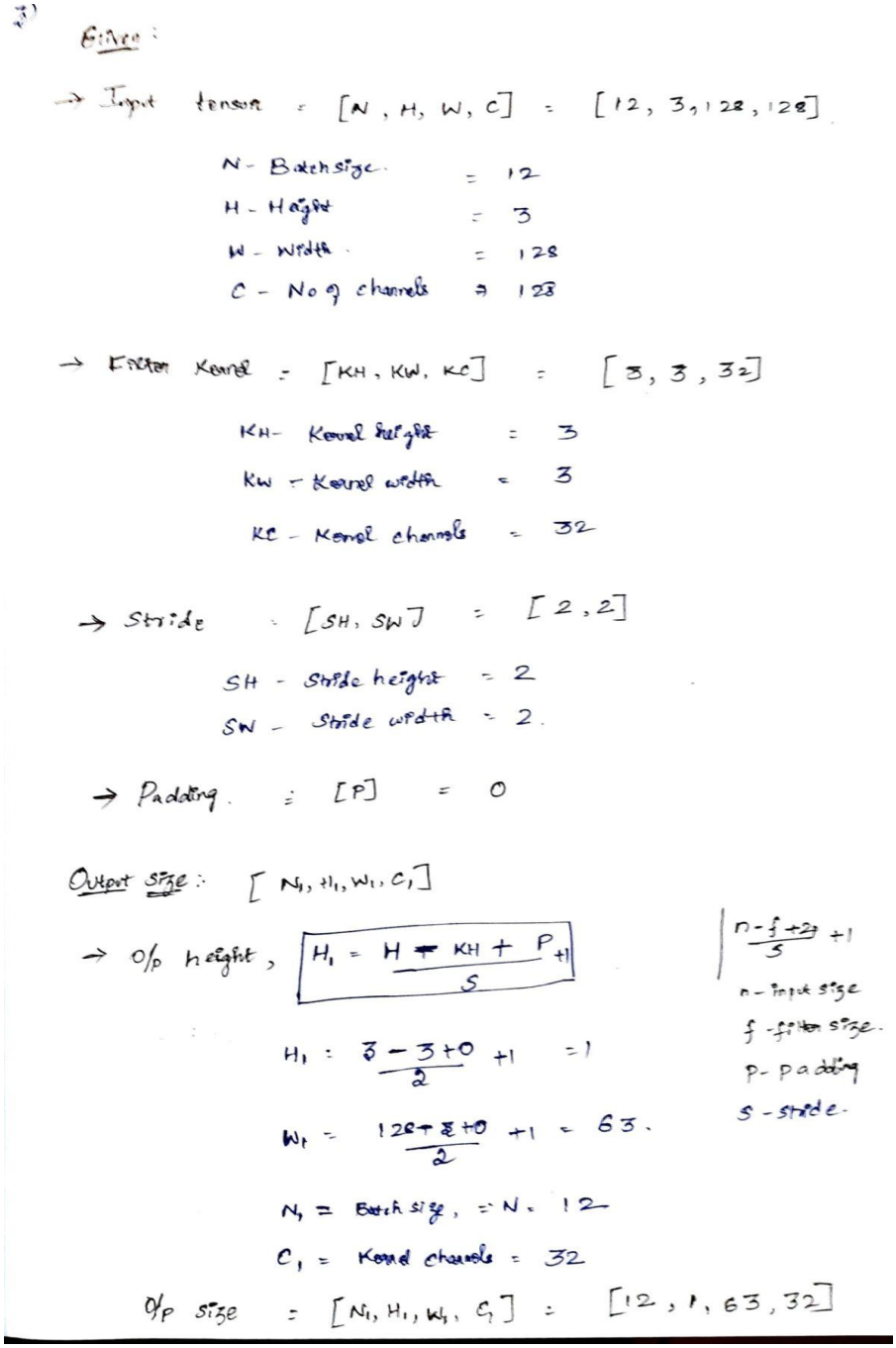
Activation function	f(x)	f(x)	$\frac{df(x)}{dx}$
sigmoid	$\frac{1}{1+e^{-x}}$		
tanh	$tanh(x)$		
RelU	max(0,x)		



- The periodic nature of sinusoidal activation functions can give rise to a 'rippling' cost function with bad local minima, which may make training difficult
- The problem may not be so bad when the data is dominated by low-frequency components.

3. Let x be a tensor with shape (12, 3, 128, 128). What is the output shape of the tensor x if I perform a calculation with 2D convolution layers with 32 3x3 filters, a stride of 2 and zero padding?

Solution:



- The output shape of the tensor is **[12, 1, 63, 32]**
- If we take N=12, C=3, H=128, W=128, then output tensor size = **[12, 32, 63, 63]**

2. Programming Task

Build a neural network that fixed the measured data y at given sampling points X. Use a supervised learning technique in order to train the neural network. Finally, visualize the given data and the trained function by the neural network.

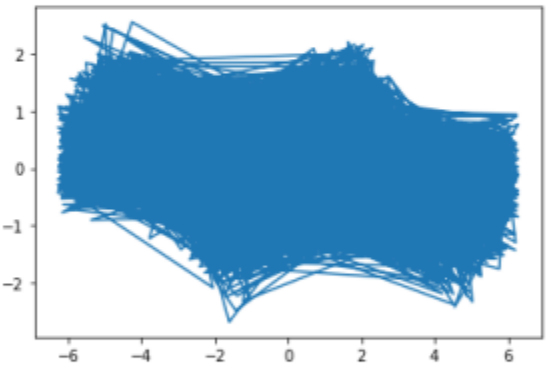
In [16]:
```
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
```

In [17]:
```
data1 = np.load('C:\\Users\\RAJIAS\\Desktop\\HZDR\\Student_Task\\x.npy')
data2 = np.load('C:\\Users\\RAJIAS\\Desktop\\HZDR\\Student_Task\\y.npy')

print(data1[0:5000])
print(data2[0:5000])
print(len(data1))
print(len(data2))

plt.plot(data1, data2)
plt.show()
```

```
[-0.02693753  0.1913795  -2.91954442 ... -4.58140501  5.36439399
 -6.16835198]
[ 0.00358727 -0.22992998 -0.12298599 ...  1.01926216 -0.98550066
 -0.03792922]
5000
5000
```



In [18]:
```
x = pd.DataFrame(data1)
y = pd.DataFrame(data2)
```

In [19]:
```
from sklearn.model_selection import train_test_split
```

In [20]:
```
X_train, X_test. y_train, y_test = train_test_split(x, y, test_size=0.3, random_st
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Input In [20], in <cell line: 1>()
----> 1 X_train, X_test. y_train, y_test = train_test_split(x, y, test_size=0.3, r
andom_state=42)

ValueError: too many values to unpack (expected 3)
```