Even Odds Excercise

Hans C. Suganda

9^{th} August 2021

Even Functions

Let an arbitrary neural network with 1 node for inputs and outputs be denoted as N. If the function N is arbitrary, then,

$$N(x) \neq N(-x)$$

Let the composite function g be defined as the following,

$$g(x) = N(f(x))$$

If $f(x)$ is even, then $g(x)$ must be even since,

$$f(x) = f(-x)$$

$$g(-x) = N(f(-x)) = N(f(x))$$

By definition of $g(x)$m

$$g(-x) = N(f(x)) = g(x)$$

Hence, g satisfying the even function property. The algorithmic implementation is shown below,

```
import numpy as np
import keras
import tensorflow as tf

#Custom Define a layer acting as even function
class EvenLayer(keras.layers.Layer):
    def call(self, inputs):
        return tf.math.cos(inputs)

#Defining Inputs of Neural Network
inputs = tf.keras.Input(shape=(1,))

#Implementing the even function $f(x)$,
layer0 = EvenLayer()(inputs)

#Some Arbitrary Neural Network Architecture
layer1 = tf.keras.layers.Dense(4, activation='sigmoid')(layer0)
layer2 = tf.keras.layers.Dense(4, activation='sigmoid')(layer1)

#Defining Outputs of Neural Network
outputs = tf.keras.layers.Dense(1, activation='sigmoid')(layer2)

#Defining Model Used
model = tf.keras.Model(inputs = inputs, outputs = outputs)
```

```
#Testing the model
test = np.array([-4,-3,-2,-1, 0, 1, 2, 3, 4])
print(model(test))

#Customary End
print('Leaves␣Blow␣in␣the␣Wind...')
```

In this specific case, the even function was chosen to be $\cos x$. Any even function would serve the same purpose.

Odd Functions

Let $E(x)$ be an even function and $O(x)$ be an odd function. Let $g(x)$ be a composite function defined below,

$$g(x) = O(x) \times E(x)$$

The function $g(x)$ must be an odd function. Taking inspiration from this, one can simply substitute $E(x)$ with the case for the neural networks even functions case,

$$g(x) = O(x) \times N(f(x))$$

```
import numpy as np
import keras
import tensorflow as tf

#Custom Define a layer acting as even function
class EvenLayer(keras.layers.Layer):
    def call(self, inputs):
        return tf.math.cos(inputs)

#Custom Define a layer acting as odd function
class OddLayer(keras.layers.Layer):
    def call(self, inputs):
        return tf.math.sin(inputs)

#Custom Define a layer acting as multiplication operation
class MultLayer(keras.layers.Layer):
    def call(self, input1, input2):
        return tf.math.multiply(input1, input2)

#Defining Inputs of Neural Network
inputs = tf.keras.Input(shape=(1,))

#Implementing the even function $f(x)$,
layer0a = EvenLayer()(inputs)

#Some Arbitrary Neural Network Architecture
layer1a = tf.keras.layers.Dense(4, activation='sigmoid')(layer0a)
layer2a = tf.keras.layers.Dense(4, activation='sigmoid')(layer1a)
layer3a = tf.keras.layers.Dense(1, activation='sigmoid')(layer1a)

#Implementing the Odd function $g(x)$
layer0b = OddLayer()(inputs)

#Defining Outputs of Neural Network
outputs = MultLayer()(layer3a,layer0b)

#Defining Model used
model = tf.keras.Model(inputs = inputs, outputs = outputs)

#Testing the model
test = np.array([-4,-3,-2,-1, 0, 1, 2, 3, 4])
print(model(test))

#Customary End
print('Leaves␣Blow␣in␣the␣Wind...')
```

In this specific case, the odd function was chosen to be $\sin x$. However, any odd function would serve the same purpose.