tf_debugging-help

November 1, 2022

1 Fully connected MLP with Tensorflow + Keras

[1]:
```
#import necessary packages (we use tensorflow as tf to present results)

print("TensorFlow version:", tf.__version__)
```

```
/Users/ronja/opt/anaconda3/lib/python3.9/site-packages/scipy/__init__.py:146:
UserWarning: A NumPy version >=1.16.5 and <1.23.0 is required for this version
of SciPy (detected version 1.23.4
  warnings.warn(f"A NumPy version >={np_minversion} and <{np_maxversion}"
```

```
TensorFlow version: 2.9.1
```

[2]:
```
#read and process date (observations X with ground true labels Ytrue)

print('Observations (%d samples x %d variables per sample):' % (X.shape[0], X.
 ↪shape[1]))
print(X)
print('Ground truth labels:', Ytrue.shape)
```

```
Observations (10 samples x 1 variables per sample):
[[0.5503]
 [0.9206]
 [0.5359]
 [0.6081]
 [0.0202]
 [0.8545]
 [0.2357]
 [0.4847]
 [0.3996]
 [0.1957]]
Ground truth labels: (10, 1)
```

1.1 Create model

[5]:
```
#initialize number of layers and weights

#model
    model = ...
    #for debugging purposes, we consider the tanh-function as an additional␣
 ↪tf-layer, which is not intended
    #for the model construction.
    #As an additional layer, it provides information for "before and after␣
 ↪effects" of the activation function.
    #This is not necessary and can be ommitted by that layer given implicitly␣
 ↪in a dense layer.
```

1.2 Perform a forward pass without any training

[6]:
```
#model.build()

#extract features

#print, what we have after initialization

for layer, layer_out in zip(model.layers, features):
    print('What we initially know about', layer.name)
    if len(layer.get_weights()) > 0:
        print('- Initial weight values:')
        print(layer.get_weights()[0]) # weights
        print('- Initial bias values:')
        print(layer.get_weights()[1]) # biases
    print("- Response:")
    print(layer_out)
```

```
What we initially know about total_input_hidden_layer_h
- Initial weight values:
[[-0.31885076 -0.18409753  0.3326131 ]]
- Initial bias values:
[ 0.38397026 -0.00957215 -0.0463587 ]
- Response:
tf.Tensor(
[[ 0.20850669 -0.11088102  0.1366783 ]
 [ 0.09043625 -0.17905234  0.25984493]
 [ 0.21309814 -0.10823001  0.13188866]
 [ 0.19007711 -0.12152185  0.15590332]
 [ 0.37752947 -0.01329092 -0.03963992]
 [ 0.1115123  -0.16688348  0.23785919]
 [ 0.30881715 -0.05296393  0.0320382 ]
 [ 0.2294233  -0.09880422  0.11485887]
```

```
 [ 0.2565575  -0.08313752  0.0865535 ]
 [ 0.32157117 -0.04560003  0.01873368]], shape=(10, 3), dtype=float32)
What we initially know about transfer_function_hidden_layer_f1
- Response:
tf.Tensor(
[[ 0.2055367  -0.11042881  0.1358335 ]
 [ 0.09019049 -0.1771631   0.25415048]
 [ 0.20992999 -0.10780938  0.1311292 ]
 [ 0.18782058 -0.12092716  0.15465234]
 [ 0.36056    -0.01329013 -0.03961917]
 [ 0.11105235 -0.1653513   0.23347262]
 [ 0.2993606  -0.05291446  0.03202724]
 [ 0.22548103 -0.09848395  0.11435642]
 [ 0.25107282 -0.08294649  0.086338  ]
 [ 0.31092688 -0.04556845  0.01873149]], shape=(10, 3), dtype=float32)
What we initially know about my_output_layer_hL
- Initial weight values:
[[-0.21384752]
 [ 0.46822393]
 [-0.32129157]]
- Initial bias values:
[0.16896272]
- Response:
tf.Tensor(
[[ 0.02966163]
 [-0.01493269]
 [ 0.03146008]
 [ 0.02248827]
 [ 0.0983644 ]
 [-0.00721976]
 [ 0.06987929]
 [ 0.03788987]
 [ 0.04869421]
 [ 0.07511728]], shape=(10, 1), dtype=float32)
```

1.3 Create the learning components and train the model for a single epoch

[7]:
```
#choose your optimization method (e.g. gradient descent) and initialize it's␣
 ↪parameters respectivly
#print, what we have after one learning step

# Perform a forward pass to get information after one learning step
features = feature_extractor(X)
for layer, layer_out in zip(model.layers, features):
    print('What we initially know about', layer.name)
    if len(layer.get_weights()) > 0:
        print('- Initial weight values:')
```

```
        print(layer.get_weights()[0]) # weights
        print('- Initial bias values:')
        print(layer.get_weights()[1]) # biases
    print("- Response:")
    print(layer_out)
```

```
1/1 [==============================] - 0s 183ms/step - loss: 0.4012 -
mean_squared_error: 0.4012
What we initially know about total_input_hidden_layer_h
- Initial weight values:
[[-0.29452956 -0.23493257  0.36630684]]
- Initial bias values:
[ 0.3824977   0.00402485 -0.05721266]
- Response:
tf.Tensor(
[[ 0.22041808 -0.12525855  0.144366  ]
 [ 0.11135378 -0.21225408  0.28000942]
 [ 0.22465931 -0.12187551  0.13909116]
 [ 0.20339428 -0.13883765  0.16553852]
 [ 0.3765482  -0.00072079 -0.04981326]
 [ 0.13082218 -0.19672503  0.25579655]
 [ 0.3130771  -0.05134876  0.02912586]
 [ 0.23973922 -0.10984696  0.12033626]
 [ 0.26480368 -0.08985421  0.08916354]
 [ 0.32485825 -0.04195146  0.01447359]], shape=(10, 3), dtype=float32)
What we initially know about transfer_function_hidden_layer_f1
- Response:
tf.Tensor(
[[ 0.21691649 -0.12460753  0.14337133]
 [ 0.1108958  -0.209123    0.27291378]
 [ 0.22095442 -0.12127562  0.13820107]
 [ 0.20063516 -0.13795239  0.16404282]
 [ 0.35970604 -0.00072079 -0.0497721 ]
 [ 0.13008094 -0.19422589  0.2503597 ]
 [ 0.3032338  -0.05130367  0.02911762]
 [ 0.2352494  -0.10940727  0.11975875]
 [ 0.25878304 -0.08961316  0.08892799]
 [ 0.3138931  -0.04192686  0.01447258]], shape=(10, 3), dtype=float32)
What we initially know about my_output_layer_hL
- Initial weight values:
[[-0.1707217 ]
 [ 0.48841658]
 [-0.359099  ]]
- Initial bias values:
[0.19376245]
- Response:
tf.Tensor(
```

```
[[ 0.04438522]
 [-0.02531207]
 [ 0.04717985]
 [ 0.03322384]
 [ 0.14987388]
 [-0.01321226]
 [ 0.10648019]
 [ 0.05715871]
 [ 0.07388006]
 [ 0.11449923]], shape=(10, 1), dtype=float32)
```

1.4 Continue training

[1]:
```
#fit the model and collect the loss
```

[2]:
```
#plot the resuts: true and predicted
```

[ ]:
```

```