Parallel Coordinates

November 23, 2021

[1]:
```
import pandas as pd
import numpy as np
import plotly.graph_objects as go
```

[2]:
```
def parallel_plot(df: pd.DataFrame, color_column, exclude_columns:list=[],␣
↪put_last_columns:list=[], exclude_color_column=True):
    cols = df.columns.tolist()
    #cols =␣
↪list(set(cols)-set(put_last_columns)-set(exclude_columns))+put_last_columns
    cols = [c for c in cols if not (c in exclude_columns or c in␣
↪put_last_columns)] + put_last_columns
    df = df[cols]
    dimensions = [_make_plotly_dict(column_name, data) for column_name, data in␣
↪df.iteritems() if not (exclude_color_column and column_name == color_column)]
    fig = go.Figure(data=
        go.Parcoords(
            line = dict(color = df[color_column],
                       colorscale = 'Electric',
                        #autocolorscale=True,
                       showscale = True,
                       cmin = df[color_column].min(),
                       cmax = df[color_column].max()),
            dimensions = dimensions
        )
    )
    fig.update_traces(labelangle=-90, selector=dict(type='parcoords'))
    fig.show();

def _make_plotly_dict(column_name, data):
    d = dict()
    t = data.dtype
    if t == bool:
        d['range'] = [-0.5,1.5]
        d['tickvals'] = [True, False]
        d['ticktext'] = ['True', 'False']
        d['values'] = data
    elif t == str or t == object:
        da = data.astype('category').cat
        d['tickvals'] = da.codes
        d['ticktext'] = da.categories
        d['values'] = da.codes
    elif t == int:
        print(column_name, 'is int')
        d['range'] = [data.min(), data.max()]
        d['tickformat'] = 'd'
        d['values'] = data
    else:
        d['range'] = [data.min(), data.max()]
        d['values'] = data
    d['label'] = column_name
    return d
```

[ ]:

```
df = pd.read_csv("/home/julian/Desktop/attributesmodelsnosort(curated).csv")
#df = df.set_index("Model Name")
import re

convert = lambda text: int(text) if text.isdigit() else text.lower()
alphanum_key = lambda key: [[convert(c) for c in re.split('([0-9]+)', k)] for k␣
 ↪in key]
df = df.sort_values("Model Name", key=alphanum_key)
df['random'] = (np.random.randn(len(df))+10)*10
```

[12]:

```
df
```

[12]:

```
              Model Name  use_class_weights  Convolutional Layer Number  \
20            BL_alex_v2              False                           5
0                 BL_FCN              False                           3
19                BL_MLP              False                           0
18  BL_rnn_simplest_lstm              False                           0
17          BL_TCN_block              False                           3
14           BL_TCN_down              False                          15
15        BL_TCN_flatten              False                          15
22           BL_TCN_last              False                          14
10                 BL_v0              False                           3
9                BL_v0_1              False                           3
1                BL_v0_2              False                           7
13               BL_v0_3              False                           7
4                  BL_v1              False                           6
2                  BL_v2              False                           5
8                  BL_v3              False                           5
6                  BL_v4              False                           5
5                  BL_v5              False                           5
3                  BL_v6              False                           7
16                 BL_v7              False                           6
```

```
12                 BL_v8              False                           5
11                 BL_v9              False                           5
7                 BL_v14              False                          29
21                BL_v15              False                           5

    Sum of Strides  Sum of Dilation  Sum of Paddings  Sum of Filters  \
20              25                8                7             174
0                3                3                0              16
19               0                0                0               0
18               0                0                0               0
17               4                4                0               7
14              15               31               56              39
15              15               31               56              39
22              14               30               56              38
10               3                5                0              11
9                4                5                0              11
1               14               11                0              27
13              14               11                0              27
4               17                6                0              36
2               20               36                0              30
8                5               16                0              13
6                5               16                0              13
5                5               16                0              13
3                7               64                0              19
16               7               32                0              20
12              14               19                0              22
11              14                8                0              22
7               71               30               22             143
21              14               96                0              26

    uses BatchNorm  uses Max Pool  uses Adaptive Average Pooling  uses Linear  \
20           False           True                           True         True
0             True          False                          False        False
19           False          False                          False         True
18           False          False                          False         True
17           False          False                          False         True
14           False          False                          False         True
15           False          False                          False         True
22           False          False                          False         True
10           False          False                          False         True
9            False          False                          False         True
1            False          False                          False         True
13           False          False                          False         True
4             True          False                           True         True
2             True           True                           True         True
8             True          False                          False         True
6            False          False                          False         True
```

```
5            False          False                          False         True
3            False          False                          False         True
16           False          False                          False         True
12            True           True                          False         True
11            True           True                          False         True
7             True           True                          False         True
21            True           True                          False         True

    uses LSTM  Final Layer      random
20      False            1   99.703232
0       False            3   99.264930
19      False            2  118.884953
18       True            5   99.613251
17      False            1   95.098469
14      False            4   99.658692
15      False            1  110.620083
22      False            2   82.070784
10      False            4  104.610637
9       False            4  100.097587
1       False            4   96.909880
13      False            4   87.537222
4       False            3  105.304706
2       False            3  100.495110
8       False            4   85.070036
6       False            4   99.914496
5       False            4   82.720868
3       False            4   79.965544
16      False            4  112.218193
12      False            4  105.190433
11      False            4   87.231979
7       False            4   86.466601
21      False            4   99.698386
```

[13]:
```
parallel_plot(df.set_index("Model Name"), 'random', put_last_columns=['Final␣
 ↪Layer', 'random'], exclude_color_column=False)
```

```
Convolutional Layer Number is int
Sum of Strides is int
Sum of Dilation is int
Sum of Paddings is int
Sum of Filters is int
Final Layer is int
```

[ ]:
```
df = pd.read_csv("/home/julian/Desktop/ALLMODELSATTRIBUTESTrue.csv")
#df = df.set_index("Model Name")
import re
```

```
convert = lambda text: int(text) if text.isdigit() else text.lower()
alphanum_key = lambda key: [[convert(c) for c in re.split('([0-9]+)', k)] for k␣
 ↪in key]
df = df.sort_values("model", key=alphanum_key)
```

[25]:
```
parallel_plot(df.set_index("model"), 'macro', put_last_columns=['Final Layer',␣
 ↪'micro', 'macro'], exclude_columns=['use_class_weights'],␣
 ↪exclude_color_column=False)
```

```
Convolutional Layer Number is int
Sum of Strides is int
Sum of Dilation is int
Sum of Paddings is int
Sum of Filters is int
Final Layer is int
```

[27]:
```
parallel_plot(df.set_index("model"), 'micro', put_last_columns=['Final Layer',␣
 ↪'macro', 'micro'], exclude_columns=['use_class_weights'],␣
 ↪exclude_color_column=False)
```

```
Convolutional Layer Number is int
Sum of Strides is int
Sum of Dilation is int
Sum of Paddings is int
Sum of Filters is int
Final Layer is int
```

[ ]:
```

```