explore_cost_functions

October 8, 2018

1 Explore cost functions

Here we will explore how different cost functions can help find the most optimal set of parameters for a microburst detector

```
In [53]: import numpy as np
         import matplotlib.pyplot as plt
         %matplotlib inline
```

Load and investigate the wavelet parameters first

```
In [54]: fPath = 'wavelet_params.csv'
         d = np.genfromtxt(fPath, delimiter=',', names=True)
```

d is a structure (very similar to a dictionary, but looks like it consists of tuples which are immutable

```
In [55]: d[:5]
```

```
Out[55]: array([(0.05, 0.3, 1. , 1029., 478., 40., 7.),
                (0.05, 0.3, 0.5, 1029., 492., 41., 7.),
                (0.05, 0.3, 0.3, 1029., 494., 41., 7.),
                (0.05, 0.3, 0.2, 1029., 494., 41., 7.),
                (0.05, 0.3, 0.2, 1029., 494., 41., 7.)],
               dtype=[('thresh', '<f8'), ('maxWidth', '<f8'), ('SIGNIF_LEVEL', '<f8'), ('validI
```

```
In [56]: def dana_cost(alpha, beta, d):
             """
             This function implement's Data's idea of a cost function that gives a
             certain weight (1/alpha) to false positive rates.

             Cost = FP + alpha*TN

             where FP is the # of false positives, TN is the number of true-negatives
             which is given by TN = (N-TP) where N is the total number of valid
             detections, and TP is the number of true positives.

             Intuitively, this is stating that false positives are (1/alpha) worse
```

```
            than true negatives.
            """
            cost = (d['validNum']*d['FPR']/100 + alpha*d['validNum']*(1 - d['TPR']/100) +
                    beta*np.abs(d['detNum'] - d['validNum']))
            return cost
```

```
In [57]: alpha = 1/5; beta = 10
         cost = dana_cost(alpha, beta, d)
```

```
In [58]: np.min(cost), np.argmin(cost)
```

```
Out[58]: (2582.554, 29)
```

Print all parameters which give you the smallest cost

```
In [59]: idx = np.where(cost == np.min(cost))[0]
         print()
         for i in idx:
             print(d[i])
```

```
(0.05, 0.5, 0.1, 1029., 794., 62., 15.)
```

1.0.1 Now find the optimal parameters for the burst parameter

```
In [60]: fPath = 'burst_params.csv'
         d_burst = np.genfromtxt(fPath, delimiter=',', names=True)
         cost_burst = dana_cost(alpha, beta, d_burst)
```

```
In [61]: np.min(cost_burst), np.argmin(cost_burst)
```

```
Out[61]: (553.312, 413)
```

```
In [62]: idx = np.where(cost_burst == np.min(cost_burst))[0]
         print()
         for i in idx:
             print(d[i])
```

```
(0.15, 1., 0.2, 1029., 412., 35., 5.)
```