Signatures

Team Red

Libraries used

- scipy
- pprint (file handling)
- pandas
- Sklearn
- multiprocessing

DTW Implementation

- Own implementation, as discussed before
- Sakoe Chiba band: 20% of width

Features

```
def compute_features(data):
    result = sc.zeros([data.shape[0], 12]);
    r = sc.sqrt(data[:,1]**2 + data[:,2]**2);
    r_std = r.std();

    result[:, 0]   = (data[:,1] - data[:,1].mean()) / r_std;      # x, normalised
    result[:, 1]   = (data[:,2] - data[:,2].mean()) / r_std;      # y, normalised
    result[:, 2]   = (r - r.mean()) / r_std;                      # r, normalised
    result[1:,3]   = (result[1:, 0] - result[:-1, 0]);            # vx
    result[1:,4]   = (result[1:, 1] - result[:-1, 1]);            # vy
    result[1:,5]   = (result[1:, 2] - result[:-1, 2]);            # vr
    result[:,3:6]  = (result[:,3:6] - result[:,5].mean())/result[:,5].std();
    result[1:,6]   = (result[1:, 3] - result[:-1, 3]);            # ax
    result[1:,7]   = (result[1:, 4] - result[:-1, 4]);            # ay
    result[1:,8]   = (result[1:, 5] - result[:-1, 5]);            # ar
    result[:,6:9]  = (result[:,6:9] - result[:,8].mean())/result[:,8].std();
    result[:, 9]   = 3*(data[:,3] - data[:,3].mean()) / data[:,3].std(); # pressure, normalised
    result[:, 10]  =  70*data[:,4];                                      # penup
    result[:, 11]  = (data[:, 5] - data[:,5].mean()) / data[:, 5].std();
    #result[:, 8]   = data[:,0] #timestamp
    return result;
```

Statistics from enrollment signatures

- Mean minimum distance
- Mean maximum distance
- Mean distance
- Root mean squared distance:
  sqrt(sum(distances)^2) / N

Distance computation loop

```
# Compute relevant DTW distances on test set
try:
    import multiprocessing as mp;
    n_threads = mp.cpu_count();
    n_threads = min([n_threads, 16]);
    n_threads = max([n_threads, 2]);

    distance_dict = {};
    with mp.Pool(n_threads) as pool:
        for writerID in test_sigs:
            distance_dict[writerID] = pool.apply_async(compare_test_train_set, (test_sigs,
            train_sigs, writerID));
        pool.close()
        pool.join()
    for writerID in distance_dict:
        distance_dict[writerID] = distance_dict[writerID].get();
except:
    print("Multiprocessing not supported, falling back to single thread.");
    distance_dict = {};
    for writerID in test_sigs:
            distance_dict[writerID] = compare_test_train_set(test_sigs, train_sigs,
            writerID);
```

Prediction model

```
stats["diff_min"]  = stats["min_dist"]  / stats["min_mean_internal"];
stats["diff_max"]  = stats["max_dist"]  / stats["max_mean_internal"];
stats["diff_mean"] = stats["mean_dist"] / stats["mean_internal"];
stats["diff_rms"]  = stats["rms_dist"]  / stats["rms_internal"];
```

- Linear regression model
- Multi level perceptron
- Monte Carlo cross validation
- Search the best parameters
- Turned out to be root mean square difference

Performance

- Linear regression model
- 1 parameter: root mean square distance divided by internal root mean square distance
- Cross validated score: 81.6 %