SyedHassaanTauqeer 19-01-2019 Final Code Notebook Draft

In [1]:

```
import os, glob
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import collections
import datetime
import sklearn.cluster as skc
from sklearn.model_selection import train_test_split
from sklearn import svm
import pickle
```

In [2]:

```
def dataTrainGen(df, var):#This function takes in the main training dataframe
 and converts the data columns to feature groups
    #These groups are then sent together with the specified label vector
    #This function is used twice because two different classifiers are trained
 on the spin and end spin phases respectively
    if var == 's':
        tempLab = df['spins'].values
    elif var == 'e':
        tempLab = df['end_spin'].values

    pair = []#The feature vector container
    for index, rows in df.iterrows():
        pair.append( ( rows['scaledTime'].astype(float), rows['scaledPower'].a
stype(float),
                      rows['scaledPowMin'].astype(float), rows['scaledPowDiff'
] ) )

    return pair, tempLab
```

The Training Phase begins here

The data is converted to a feature matrix and the corresponding label vector, and are then split to train/test portions.
Two classifiers are trained separately for the "spin phase" and "end spin phase" detection.
The classifiers are then run on the test portion to give an accuracy value.
The accuracy is computed by checking the matches of the positively labeled test samples against their predicted values, since the negative labels are not really relevant in checking for accuracy

In [3]:

```
os.chdir('C:\\Users\\Labyrinth\\JUPYTER NOTEBOOKS\\WeWash_Praktikum_TUM3sem\\W
eWash_Analysis_ver2\\Data\\processed')
```

In [4]:
```
_file = 'UniFeatScaledV2_146-150-400_696_128895-212236.csv'
bigDF = pd.read_csv(_file, delimiter=';')
print 'Machines: ', len(bigDF['machine'].unique())
print bigDF.head(2)
```

```
Machines:  696
   end_spin  machine   pow  powMin  scaledPowMin  scaledPower  scaledTime  \
0       0.0   128895  0.04    0.04           0.0          0.0         0.0
1       0.0   128895  0.04    0.04           0.0          0.0         0.0

   spins  time  scaledPowDiff
0    0.0   0.0            0.0
1    0.0   1.0            0.0
```

In [5]:
```
X_spin, Lab_spin = dataTrainGen(bigDF, 's')#calling this function for obtainin
g the training data and spin labels
X_end, Lab_end = dataTrainGen(bigDF, 'e')#calling this function for obtaining
 the same training data and end spin labels
X_spin = np.asarray(X_spin)#converting to numpy array for faster computations
 and wider function support
X_end = np.asarray(X_end)#converting to numpy array for faster computations an
d wider function support
print X_spin.shape, X_end.shape
Lab_spin = np.asarray(Lab_spin)
Lab_end = np.asarray(Lab_end)
print Lab_spin.shape, Lab_end.shape
```

```
(983140L, 4L) (983140L, 4L)
(983140L,) (983140L,)
```

In [6]:
```
X_spin_train, X_spin_test, y_spin_train, y_spin_test = train_test_split(X_spin
, Lab_spin, test_size=0.1)#splitting the data in
#train/test pair with a 10% ratio for testing to maximize on training data
X_end_train, X_end_test, y_end_train, y_end_test = train_test_split(X_end, Lab
_end, test_size=0.1)
print X_spin_train.shape, y_spin_train.shape, '\t\t', X_end_train.shape, y_end
_train.shape
print X_spin_test.shape, y_spin_test.shape, '\t\t', X_end_test.shape, y_end_te
st.shape
```

```
(884826L, 4L) (884826L,)                (884826L, 4L) (884826L,)
(98314L, 4L) (98314L,)          (98314L, 4L) (98314L,)
```

In [ ]:
```
#Spin classifier Model - takes about 4-5 hours to train
clf_spin = svm.SVC()
clf_spin.fit(X_spin_train, y_spin_train)
```

Out[ ]:
```
SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
  decision_function_shape='ovr', degree=3, gamma='auto', kernel='rbf',
  max_iter=-1, probability=False, random_state=None, shrinking=True,
  tol=0.001, verbose=False)
```

In [ ]:

```
#End spin classifier model - takes roughly the same time as the spin model
clf_end = svm.SVC()
clf_end.fit(X_end_train, y_end_train)
```

In [ ]:

```
#saving the models to files to avoid the training computation repeatedly
pickle.dump(clf_spin, open('SVM_spinModel4f_146-150-400.sav', 'wb'))
pickle.dump(clf_end, open('SVM_endModel4f_146-150-400.sav', 'wb'))
pickle.dump(clf_spin, open('SVM_spinModel4f_146-150-400.pkl', 'wb'))
pickle.dump(clf_end, open('SVM_endModel4f_146-150-400.pkl', 'wb'))
```

In [ ]:

```
#testing on 10% of the data
yPred_spin = clf_spin.predict(X_spin_test)
print yPred_spin.shape
yPred_end = clf_end.predict(X_end_test)
print yPred_end.shape
```

In [ ]:

```
#checking for accuracy
corr = 0
total = np.count_nonzero(y_spin_test)
for i in range(len(yPred_spin)):
    if yPred_spin[i]==y_spin_test[i] and y_spin_test[i]==1:
        corr+=1
print 'Spin Test accuracy: ', float(corr)/float(total)

corr = 0
total = np.count_nonzero(y_end_test)
for i in range(len(yPred_end)):
    if yPred_end[i]==y_end_test[i] and y_end_test[i]==1:
        corr+=1
print 'End Test accuracy: ', float(corr)/float(total)
```