# Pines_SVM

July 11, 2018

In [1]:
```
import numpy as np
from sklearn import svm
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score
import pickle
from sklearn.model_selection import train_test_split
import numpy as np

I_gt = np.load('/home/abhi/Desktop/Hyperspectral_dataset/vectorized_data/I_gt.npy')
I_vect = np.load('/home/abhi/Desktop/Hyperspectral_dataset/vectorized_data/I_vect.npy')


#To verify the number of samples present in dataset and the unclassified points
pines_sampleno=np.sort(I_gt)


#To get all the indices where a specific class is present
pines_indices=np.argsort(I_gt,kind='quicksort')


#Array slicing to get each class of Indian Pines
igt=pines_sampleno[10776:]
I_new=pines_indices[10776:]


a=I_new[0:46]
Class_1= I_vect[a]

b=I_new[46:1474]
Class_2= I_vect[b]

c=I_new[1474:2304]
Class_3= I_vect[c]

d=I_new[2304:2541]
Class_4= I_vect[d]

e=I_new[2541:3024]
```

```
Class_5= I_vect[e]

f=I_new[3024:3754]
Class_6= I_vect[f]

g=I_new[3754:3782]
Class_7= I_vect[g]

h=I_new[3782:4260]
Class_8= I_vect[h]

i=I_new[4260:4280]
Class_9= I_vect[i]

j=I_new[4280:5252]
Class_10= I_vect[j]

k=I_new[5252:7707]
Class_11= I_vect[k]

l=I_new[7707:8300]
Class_12= I_vect[l]

m=I_new[8300:8505]
Class_13= I_vect[m]

n=I_new[8505:9770]
Class_14= I_vect[n]

o=I_new[9770:10156]
Class_15= I_vect[o]

p=I_new[10156:10249]
Class_16= I_vect[p]

#To split the ground truth according to the classes
gt_1=I_gt[a]
gt_2=I_gt[b]
gt_3=I_gt[c]
gt_4=I_gt[d]
gt_5=I_gt[e]
gt_6=I_gt[f]
gt_7=I_gt[g]
gt_8=I_gt[h]
gt_9=I_gt[i]
gt_10=I_gt[j]
gt_11=I_gt[k]
gt_12=I_gt[l]
```

```
gt_13=I_gt[m]
gt_14=I_gt[n]
gt_15=I_gt[o]
gt_16=I_gt[p]


#Splitting training and testing data for each class for Indian Pines dataset

ip_1, ip_1_test, y_1, y_1_test = train_test_split(Class_1 ,gt_1, test_size=0.5, shuffle

ip_2, ip_2_test, y_2, y_2_test = train_test_split(Class_2 ,gt_2, test_size=0.5, shuffle


ip_3, ip_3_test, y_3, y_3_test = train_test_split(Class_3 ,gt_3, test_size=0.5, shuffle

ip_4, ip_4_test, y_4, y_4_test = train_test_split(Class_4 ,gt_4, test_size=0.5, shuffle

ip_5, ip_5_test, y_5, y_5_test = train_test_split(Class_5 ,gt_5, test_size=0.5, shuffle

ip_6, ip_6_test, y_6, y_6_test = train_test_split(Class_6 ,gt_6, test_size=0.5, shuffle

ip_7, ip_7_test, y_7, y_7_test = train_test_split(Class_7 ,gt_7, test_size=0.5, shuffle

ip_8, ip_8_test, y_8, y_8_test = train_test_split(Class_8 ,gt_8, test_size=0.5, shuffle

ip_9, ip_9_test, y_9, y_9_test = train_test_split(Class_9 ,gt_9, test_size=0.5, shuffle

ip_10, ip_10_test, y_10, y_10_test = train_test_split(Class_10 ,gt_10, test_size=0.5, s

ip_11, ip_11_test, y_11, y_11_test = train_test_split(Class_11 ,gt_11, test_size=0.5, s

ip_12, ip_12_test, y_12, y_12_test = train_test_split(Class_12 ,gt_12, test_size=0.5, s

ip_13, ip_13_test, y_13, y_13_test = train_test_split(Class_13 ,gt_13, test_size=0.5, s

ip_14, ip_14_test, y_14, y_14_test = train_test_split(Class_14 ,gt_14, test_size=0.5, s

ip_15, ip_15_test, y_15, y_15_test = train_test_split(Class_15 ,gt_15, test_size=0.5, s

ip_16, ip_16_test, y_16, y_16_test = train_test_split(Class_16 ,gt_16, test_size=0.5, s

ix_train=np.concatenate((ip_1,ip_2,ip_3,ip_4,ip_5,ip_6,ip_7,ip_8,ip_9,ip_10,ip_11,ip_1
iy_train=np.concatenate((y_1,y_2,y_3,y_4,y_5,y_6,y_7,y_8,y_9,y_10,y_11,y_12,y_13,y_14,
ix_test=np.concatenate((ip_1_test,ip_2_test,ip_3_test,ip_4_test,ip_5_test,ip_6_test,ip
iy_test=np.concatenate((y_1_test,y_2_test,y_3_test,y_4_test,y_5_test,y_6_test,y_7_test
```

## 0.1 Using default parameters and realising that accuracy is not optimal

### 0.1.1 Linear Kernel SVM

```
In [5]: linear_I = svm.SVC(C=0.5 , kernel='linear',cache_size=5000,gamma=1)
        linear_I.fit(ix_train,iy_train)
        linear_I.score(ix_train,iy_train)
        ipred=linear_I.predict(ix_test)
```

```
In [6]: accuracy_score(iy_test, ipred)
```

```
Out[6]: 0.7576053042121685
```

### 0.1.2 Polynomial Kernel SVM

```
In [9]: poly_p = svm.SVC(C=1 ,kernel='poly' ,cache_size=9000,degree=3, gamma=1)
        poly_p.fit(ix_train,iy_train)
        poly_p.score(ix_train,iy_train)
        ipred1=poly_p.predict(ix_test)
        ipred1
```

```
Out[9]: array([ 1,  1,  1, ..., 16, 16, 16], dtype=uint8)
```

```
In [10]: accuracy_score(iy_test, ipred1)
```

```
Out[10]: 0.8190327613104524
```

```
In [11]: save_classifier = open ("IndianPines_LinearSVM.pickle","wb")
         pickle.dump(linear_I,save_classifier)
         save_classifier.close()
```

```
In [12]: save_classifier = open ("IndianPines_PolySVM.pickle","wb")
         pickle.dump(poly_p,save_classifier)
         save_classifier.close()
```

### 0.1.3 Using Grid Search , varying C from 1 to 10 , and kernel linear and RBF

```
In [14]: from sklearn.model_selection import GridSearchCV
         parameters = {'kernel':('linear', 'rbf'), 'C':[1, 10]}
         svc = svm.SVC()
         clf = GridSearchCV(svc, parameters)
```

```
In [16]:
```

```
Out[16]: SVC(C=1, cache_size=9000, class_weight=None, coef0=0.0,
           decision_function_shape='ovr', degree=3, gamma=1, kernel='poly',
           max_iter=-1, probability=False, random_state=None, shrinking=True,
           tol=0.001, verbose=False)
```

```
In [4]: save_classifier = open ("poly_k.pickle","wb")
        pickle.dump(linear_I,save_classifier)
        save_classifier.close()
```