In [134]:
```
import numpy as np
from svmutil import *
```

In [135]:
```
def readfile(filename):
    X = []
    Y = []
    for lines in open(filename).readlines():
        temp = lines.strip().split()
        Y.append(int(temp[0]))
        x = np.zeros(36)
        for i in range(1, len(temp)):
            index_value = temp[i].split(":")
            x[int(index_value[0])-1] = float(index_value[1])
        X.append(x.tolist())
    return X, Y
def label_OneOfTheClass(Y, Class):
    label = np.zeros(len(Y))
    for i in range(len(Y)):
        if Y[i]==Class:
            label[i] = 1
        else:
            label[i] = -1
    return label
TrainFile = "../../satimage.scale"
TestFile = "../../satimage.scale.t"
X, Y = readfile(TrainFile)
Xt, Yt = readfile(TestFile)
```

In [136]:
```
## 15
label = label_OneOfTheClass(Y, 3)
m = svm_train(label, X, '-s 0 -t 0 -c 10')
SV = m.get_SV()
alpha = m.get_sv_coef()
SVnum = len(SV)
w = np.zeros(36)
for i in range(SVnum):
    for j in range(36):
        if SV[i].get(j+1) != None:
            w[j] += alpha[i][0]*SV[i][j+1]
print("15: ", np.sqrt(np.dot(w, w)))
```

```
15:  8.457084298367683
```

In [137]:
```
## 16, 17
max_acc = -1
max_SVnum = 0
for i in range(1, 6):
    Class = i
    label = label_OneOfTheClass(Y, i)
    m = svm_train(label, X, '-s 0 -t 1 -c 10 -d 2 -g 1 -r 1')
    p_label, p_acc, p_val = svm_predict(label, X, m)
    if p_acc[0] > max_acc:
        max_acc = p_acc[0]
        best = i
    SV = m.get_SV()
    SVnum = len(SV)
    if SVnum > max_SVnum:
        max_SVnum = SVnum
print("16: ", best)
print("17: ", max_SVnum)
```

```
Accuracy = 99.9324% (4432/4435) (classification)
Accuracy = 100% (4435/4435) (classification)
Accuracy = 97.7678% (4336/4435) (classification)
Accuracy = 95.9865% (4257/4435) (classification)
Accuracy = 99.3236% (4405/4435) (classification)
16:  2
17:  712
```

In [138]:
```
## 18
C = [0.01, 0.1, 1, 10, 100]
label = label_OneOfTheClass(Y, 6)
label_t = label_OneOfTheClass(Yt, 6)
min_Eout = 100
for i in range(5):
    param = '-s 0 -t 2 -g 10 -c ' + str(C[i])
    m = svm_train(label, X, param)
    p_label, p_acc, p_val = svm_predict(label_t, Xt, m)
    if (100-p_acc[0]) < min_Eout:
        min_Eout = 100-p_acc[0]
        best = C[i]
print("18: ", best)
```

```
Accuracy = 76.5% (1530/2000) (classification)
Accuracy = 83.65% (1673/2000) (classification)
Accuracy = 89.35% (1787/2000) (classification)
Accuracy = 90.3% (1806/2000) (classification)
Accuracy = 90.3% (1806/2000) (classification)
18:  10
```

In [139]:
```
## 19
gamma = [0.1, 1, 10, 100, 1000]
min_Eout = 100
for i in range(5):
    param = '-s 0 -t 2 -c 0.1 -g ' + str(gamma[i])
    m = svm_train(label, X, param)
    p_label, p_acc, p_val = svm_predict(label_t, Xt, m)
    if (100-p_acc[0]) < min_Eout:
        min_Eout = 100-p_acc[0]
        best = gamma[i]
print("19: ", best)
```

```
Accuracy = 90.15% (1803/2000) (classification)
Accuracy = 93% (1860/2000) (classification)
Accuracy = 83.65% (1673/2000) (classification)
Accuracy = 76.5% (1530/2000) (classification)
Accuracy = 76.5% (1530/2000) (classification)
19:  1
```

In [147]:
```
## 20
gamma = [0.1, 1, 10, 100, 1000]
data_num = len(X)
choose_time = np.zeros([5])
for i in range(1000):
    X_val = []
    Y_val = []
    X_train = []
    Y_train = []
    val = np.random.randint(0, data_num, 200)
    is_val = np.zeros(data_num)
    for j in range(200):
        is_val[val[j]] = 1
    for j in range(data_num):
        if is_val[j] == 1:
            X_val.append(X[j])
            Y_val.append(Y[j])
        else:
            X_train.append(X[j])
            Y_train.append(Y[j])
    label_val = label_OneOfTheClass(Y_val, 6)
    label_train = label_OneOfTheClass(Y_train, 6)
    max_acc = 0
    for g in range(5):
        param = '-s 0 -t 2 -c 0.1 -g ' + str(gamma[g])
        m = svm_train(label_train, X_train, param)
        p_label, p_acc, p_val = svm_predict(label_val, X_val, m)
        if p_acc[0] > max_acc:
            max_acc = p_acc[0]
            best = g
    print(i, ":", gamma[best], end = "\r")
    choose_time[best] += 1
print("20: ", choose_time)
```