In [1]:
```
from svmutil import *
import pandas as pd
import numpy as np
from math import pow
import timeit
import operator
```

In [2]:
```
# LibSVM function:

#svm_type = model.get_svm_type()
#nr_class = model.get_nr_class()
#svr_probability = model.get_svr_probability()
#class_labels = model.get_labels()
#sv_indices = model.get_sv_indices()
#nr_sv = model.get_nr_sv()
#is_prob_model = model.is_probability_model()
#support_vector_coefficients = model.get_sv_coef()
#support_vectors = model.get_SV()
```

In [3]:
```
# Read Data

y, x = svm_read_problem("satimage.scale")
test_y, test_x = svm_read_problem("satimage.scale.t")
```

In [4]:
```
def problem15():
    trainY = [1 if target == 3 else -1 for target in y]
    trainX = x
    model = svm_train(trainY, trainX, "-s 0 -t 0 -c 10")
    SV = pd.DataFrame(model.get_SV())
    SV = SV.fillna(0)
    alphaY = pd.DataFrame(model.get_sv_coef())
    wNorm = np.linalg.norm(SV.T.dot(alphaY))
    print("Problem 15: |w| is ", wNorm)
    return

problem15()
```

```
Problem 15: |w| is  8.457084298367056
```

In [5]:
```
def problem16():
    print("Problem 16: \n")
    for i in range(1, 6):
        trainY = [1 if target == i else -1 for target in y]
        trainX = x
        model = svm_train(trainY, trainX, "-s 0 -t 1 -d 2 -c 10")
        print(f"This model depicts {i} versus not {i}:")
        result = svm_predict(trainY, trainX, model)
        print()
    return

problem16()
```

```
Problem 16:

This model depicts 1 versus not 1:
Accuracy = 98.3766% (4363/4435) (classification)

This model depicts 2 versus not 2:
Accuracy = 99.301% (4404/4435) (classification)

This model depicts 3 versus not 3:
Accuracy = 95.8061% (4249/4435) (classification)

This model depicts 4 versus not 4:
Accuracy = 90.6426% (4020/4435) (classification)

This model depicts 5 versus not 5:
Accuracy = 95.6483% (4242/4435) (classification)
```

In [6]:
```python
def problem17():
    print("Problem 17: \n")
    MostNrSV = 0
    for i in range(1, 6):
        trainY = [1 if target == i else -1 for target in y]
        trainX = x
        model = svm_train(trainY, trainX, "-s 0 -t 1 -d 2 -c 10")
        print(f"This model depicts {i} versus not {i}:")
        result = svm_predict(trainY, trainX, model)
        print(f"And the nr_sv is: {model.get_nr_sv()} \n")
        if MostNrSV < model.get_nr_sv():
            MostNrSV = model.get_nr_sv()

    print(f"The biggest number of SV in these models is {MostNrSV}")
    return

problem17()
```

```
Problem 17:

This model depicts 1 versus not 1:
Accuracy = 98.3766% (4363/4435) (classification)
And the nr_sv is: 520

This model depicts 2 versus not 2:
Accuracy = 99.301% (4404/4435) (classification)
And the nr_sv is: 165

This model depicts 3 versus not 3:
Accuracy = 95.8061% (4249/4435) (classification)
And the nr_sv is: 750

This model depicts 4 versus not 4:
Accuracy = 90.6426% (4020/4435) (classification)
And the nr_sv is: 859

This model depicts 5 versus not 5:
Accuracy = 95.6483% (4242/4435) (classification)
And the nr_sv is: 690

The biggest number of SV in these models is 859
```

In [7]:
```python
def problem18():
    print("problem 18: \n")
    BestACC, BestC = -1, 0
    trainY, trainX = [1 if target == 6 else -1 for target in y], x
    testY, testX = [1 if target == 6 else -1 for target in test_y], test_x

    for i in range(-2, 3, 1):
        cost = pow(10, i)
        print(f"This model consider cost is {cost}:")
        model = svm_train(trainY, trainX, f'-s 0 -t 2 -g 10 -c {cost}')
        result = svm_predict(testY, testX, model)
        print()
        if BestACC < result[1][0]:
            BestC = cost
            BestACC = result[1][0]

    print(f"Among these five models, the best ACC is {BestACC}, best C is {BestC}")
    return
```

```python
problem18()
```

```
problem 18:

This model consider cost is 0.01:
Accuracy = 76.5% (1530/2000) (classification)

This model consider cost is 0.1:
Accuracy = 83.65% (1673/2000) (classification)

This model consider cost is 1.0:
Accuracy = 89.35% (1787/2000) (classification)

This model consider cost is 10.0:
Accuracy = 90.3% (1806/2000) (classification)

This model consider cost is 100.0:
Accuracy = 90.3% (1806/2000) (classification)

Among these five models, the best ACC is 90.3, best C is 10.0
```

In [8]:
```python
def problem19():
    print("Problem 19: \n")
    BestACC, BestGamma = -1, 0
    trainY, trainX = [1 if target == 6 else -1 for target in y], x
    testY, testX = [1 if target == 6 else -1 for target in test_y], test_x

    for i in range(-1, 4, 1):
        gamma = pow(10, i)
        print(f"This model using gamma equal {gamma}: ")
        model = svm_train(trainY, trainX, f'-s 0 -t 2 -g {gamma} -c 0.1')
        result = svm_predict(testY, testX, model)
        print()
        if BestACC < result[1][0]:
            BestGamma = gamma
            BestACC = result[1][0]

    print(f"Among these five models, the best ACC is {BestACC}, best gamma is {BestGamma}")
    return

problem19()
```

```
Problem 19:

This model using gamma equal 0.1:
Accuracy = 90.15% (1803/2000) (classification)

This model using gamma equal 1.0:
Accuracy = 93% (1860/2000) (classification)

This model using gamma equal 10.0:
Accuracy = 83.65% (1673/2000) (classification)

This model using gamma equal 100.0:
Accuracy = 76.5% (1530/2000) (classification)

This model using gamma equal 1000.0:
Accuracy = 76.5% (1530/2000) (classification)

Among these five models, the best ACC is 93.0, best gamma is 1.0
```

In [12]:

```
def problem20():
    start = timeit.default_timer()
    gammaDict = {0.1:0, 1:0, 10:0, 100:0, 1000:0}

    X = pd.DataFrame(x)
    X = X.fillna(0)
    col = X.columns.tolist()
    col.sort()
    X = X[col]
    Y = [1 if target == 6 else -1 for target in y]
    Y = pd.Series(Y)

    for _ in range(1000):
        best_gamma = 0
        best_ACC = -1
        indexes = np.random.choice(X.index, size = 200)
        ValX, ValY = X.loc[indexes], Y.loc[indexes]
        TrainX, TrainY = X.drop(indexes, axis = 0), Y.drop(indexes, axis = 0)
        for gamma in [0.1, 1, 10, 100, 1000]:
            #print(f"This model uses gamma equal {gamma}: ")
            model = svm_train(TrainY.to_numpy(), TrainX.to_numpy(), f'-s 0 -t 2 -g {gamma} -c 0.1')
            result = svm_predict(ValY.to_numpy(), ValX.to_numpy(), model)
            if result[1][0] > best_ACC:
                best_ACC = result[1][0]
                best_gamma = gamma
        gammaDict[best_gamma] += 1
        #print(f"The best gamma is {best_gamma}")

    stop = timeit.default_timer()
    print('Time: ', stop - start)
    return gammaDict

answer20 = problem20()
```

In [13]:
```
print(f"The most frequent gamma is {max(answer20.items(), key=operator.itemgetter(1))[0]}")
```

```
The most frequent gamma is 1
```