In [39]:
```
from liblinearutil import *
import pandas as pd
import numpy as np
```

In [51]:
```
train_data = pd.read_table("hw4_train.dat.txt", header=None, names=["x_{1}",
                                                                    "x_{2}", "x_{3}", "x_{4}", "x_{5}", "x_{6}", "y"], sep=" ")
test_data = pd.read_table("hw4_test.dat.txt", header=None, names=["x_{1}",
                                                                  "x_{2}", "x_{3}", "x_{4}", "x_{5}", "x_{6}", "y"], sep=" ")
```

In [52]:
```
def phi(x):
    phi_array = np.array([1])
    phi_array = np.append(phi_array, x)
    for i in range(len(x)):
        for j in range(i, len(x)):
            phi_array = np.append(phi_array, x[i] * x[j])
    return phi_array
```

In [53]:
```
train_Y = train_data["y"]
train_X = train_data.drop("y", axis=1)

test_Y = test_data["y"]
test_X = test_data.drop("y", axis=1)
```

In [54]:
```
train_X = train_X.apply(phi, axis=1)
train_X = pd.DataFrame(train_X.tolist())

test_X = test_X.apply(phi, axis=1)
test_X = pd.DataFrame(test_X.tolist())
```

Problem 16: lambda and c are reciprocal to each other

```
In [80]: m_1 = train(train_Y.to_numpy(), train_X.to_numpy(), "-s 0 -c 0.0001 -e
          0.000001")
         m_2 = train(train_Y.to_numpy(), train_X.to_numpy(), "-s 0 -c 0.01 -e 0.
         000001")
         m_3 = train(train_Y.to_numpy(), train_X.to_numpy(), "-s 0 -c 1 -e 0.000
         001")
         m_4 = train(train_Y.to_numpy(), train_X.to_numpy(), "-s 0 -c 100 -e 0.0
         00001")
         m_5 = train(train_Y.to_numpy(), train_X.to_numpy(), "-s 0 -c 10000 -e
          0.000001")
```

```
In [81]: p_labels, p_acc, p_vals = predict(test_Y.to_numpy(), test_X.to_numpy(),
          m_1)
         p_labels, p_acc, p_vals = predict(test_Y.to_numpy(), test_X.to_numpy(),
          m_2)
         p_labels, p_acc, p_vals = predict(test_Y.to_numpy(), test_X.to_numpy(),
          m_3)
         p_labels, p_acc, p_vals = predict(test_Y.to_numpy(), test_X.to_numpy(),
          m_4)
         p_labels, p_acc, p_vals = predict(test_Y.to_numpy(), test_X.to_numpy(),
          m_5)
```

```
Accuracy = 51.6667% (155/300) (classification)
Accuracy = 74.3333% (223/300) (classification)
Accuracy = 82% (246/300) (classification)
Accuracy = 87.6667% (263/300) (classification)
Accuracy = 86% (258/300) (classification)
```

Problem 17:

```
In [82]: p_labels, p_acc, p_vals = predict(train_Y.to_numpy(), train_X.to_numpy
         (), m_1)
         p_labels, p_acc, p_vals = predict(train_Y.to_numpy(), train_X.to_numpy
         (), m_2)
         p_labels, p_acc, p_vals = predict(train_Y.to_numpy(), train_X.to_numpy
         (), m_3)
         p_labels, p_acc, p_vals = predict(train_Y.to_numpy(), train_X.to_numpy
         (), m_4)
         p_labels, p_acc, p_vals = predict(train_Y.to_numpy(), train_X.to_numpy
         (), m_5)
```

```
Accuracy = 46.5% (93/200) (classification)
Accuracy = 80.5% (161/200) (classification)
Accuracy = 87.5% (175/200) (classification)
Accuracy = 91% (182/200) (classification)
Accuracy = 91% (182/200) (classification)
```

Problem 18:

In [84]:
```
m_1 = train(train_Y[:120].to_numpy(), train_X[:120].to_numpy(), "-s 0 -
c 0.0001 -e 0.000001")
m_2 = train(train_Y[:120].to_numpy(), train_X[:120].to_numpy(), "-s 0 -
c 0.01 -e 0.000001")
m_3 = train(train_Y[:120].to_numpy(), train_X[:120].to_numpy(), "-s 0 -
c 1 -e 0.000001")
m_4 = train(train_Y[:120].to_numpy(), train_X[:120].to_numpy(), "-s 0 -
c 100 -e 0.000001")
m_5 = train(train_Y[:120].to_numpy(), train_X[:120].to_numpy(), "-s 0 -
c 10000 -e 0.000001")
```

In [85]:
```
p_labels, p_acc, p_vals = predict(train_Y[120:].to_numpy(), train_X[120
:].to_numpy(), m_1)
p_labels, p_acc, p_vals = predict(train_Y[120:].to_numpy(), train_X[120
:].to_numpy(), m_2)
p_labels, p_acc, p_vals = predict(train_Y[120:].to_numpy(), train_X[120
:].to_numpy(), m_3)
p_labels, p_acc, p_vals = predict(train_Y[120:].to_numpy(), train_X[120
:].to_numpy(), m_4)
p_labels, p_acc, p_vals = predict(train_Y[120:].to_numpy(), train_X[120
:].to_numpy(), m_5)
```

```
Accuracy = 42.5% (34/80) (classification)
Accuracy = 75% (60/80) (classification)
Accuracy = 80% (64/80) (classification)
Accuracy = 86.25% (69/80) (classification)
Accuracy = 81.25% (65/80) (classification)
Accuracy = 81.25% (65/80) (classification)
```

In [76]:
```
p_labels, p_acc, p_vals = predict(test_Y.to_numpy(), test_X.to_numpy(),
 m_4)
```

```
Accuracy = 86% (258/300) (classification)
```

Problem 19: Consider the result in Problem 16

Problem 20:

In [109]:
```
ACC_list = []
```

In [110]:
```
Y = train_Y.drop([i for i in range(0, 40)]).to_numpy()
X = train_X.drop([i for i in range(0, 40)]).to_numpy()
m_1 = train(Y, X, "-s 0 -c 0.0001 -e 0.000001")
m_2 = train(Y, X, "-s 0 -c 0.01 -e 0.000001")
m_3 = train(Y, X, "-s 0 -c 1 -e 0.000001")
m_4 = train(Y, X, "-s 0 -c 100 -e 0.000001")
m_5 = train(Y, X, "-s 0 -c 10000 -e 0.000001")
```

In [111]:
```
fold_1 = []
Y = train_Y[:40].to_numpy()
X = train_X[:40].to_numpy()
p_labels, p_acc, p_vals = predict(Y, X, m_1)
fold_1.append(p_acc[0])
p_labels, p_acc, p_vals = predict(Y, X, m_2)
fold_1.append(p_acc[0])
p_labels, p_acc, p_vals = predict(Y, X, m_3)
fold_1.append(p_acc[0])
p_labels, p_acc, p_vals = predict(Y, X, m_4)
fold_1.append(p_acc[0])
p_labels, p_acc, p_vals = predict(Y, X, m_5)
fold_1.append(p_acc[0])
ACC_list.append(fold_1)
```

```
Accuracy = 42.5% (17/40) (classification)
Accuracy = 75% (30/40) (classification)
Accuracy = 82.5% (33/40) (classification)
Accuracy = 85% (34/40) (classification)
Accuracy = 87.5% (35/40) (classification)
```

In [112]:
```
Y = train_Y.drop([i for i in range(40, 80)]).to_numpy()
X = train_X.drop([i for i in range(40, 80)]).to_numpy()
m_1 = train(Y, X, "-s 0 -c 0.0001 -e 0.000001")
m_2 = train(Y, X, "-s 0 -c 0.01 -e 0.000001")
m_3 = train(Y, X, "-s 0 -c 1 -e 0.000001")
m_4 = train(Y, X, "-s 0 -c 100 -e 0.000001")
m_5 = train(Y, X, "-s 0 -c 10000 -e 0.000001")
```

In [113]:
```
fold_1 = []
Y = train_Y[40:80].to_numpy()
X = train_X[40:80].to_numpy()
p_labels, p_acc, p_vals = predict(Y, X, m_1)
fold_1.append(p_acc[0])
p_labels, p_acc, p_vals = predict(Y, X, m_2)
fold_1.append(p_acc[0])
p_labels, p_acc, p_vals = predict(Y, X, m_3)
fold_1.append(p_acc[0])
p_labels, p_acc, p_vals = predict(Y, X, m_4)
fold_1.append(p_acc[0])
p_labels, p_acc, p_vals = predict(Y, X, m_5)
fold_1.append(p_acc[0])
ACC_list.append(fold_1)
```

```
Accuracy = 65% (26/40) (classification)
Accuracy = 92.5% (37/40) (classification)
Accuracy = 90% (36/40) (classification)
Accuracy = 77.5% (31/40) (classification)
Accuracy = 75% (30/40) (classification)
```

In [115]:
```
Y = train_Y.drop([i for i in range(80, 120)]).to_numpy()
X = train_X.drop([i for i in range(80, 120)]).to_numpy()
m_1 = train(Y, X, "-s 0 -c 0.0001 -e 0.000001")
m_2 = train(Y, X, "-s 0 -c 0.01 -e 0.000001")
```

```
m_3 = train(Y, X, "-s 0 -c 1 -e 0.000001")
m_4 = train(Y, X, "-s 0 -c 100 -e 0.000001")
m_5 = train(Y, X, "-s 0 -c 10000 -e 0.000001")
```

In [116]:
```
fold_1 = []
Y = train_Y[80:120].to_numpy()
X = train_X[80:120].to_numpy()
p_labels, p_acc, p_vals = predict(Y, X, m_1)
fold_1.append(p_acc[0])
p_labels, p_acc, p_vals = predict(Y, X, m_2)
fold_1.append(p_acc[0])
p_labels, p_acc, p_vals = predict(Y, X, m_3)
fold_1.append(p_acc[0])
p_labels, p_acc, p_vals = predict(Y, X, m_4)
fold_1.append(p_acc[0])
p_labels, p_acc, p_vals = predict(Y, X, m_5)
fold_1.append(p_acc[0])
ACC_list.append(fold_1)
```

```
Accuracy = 47.5% (19/40) (classification)
Accuracy = 85% (34/40) (classification)
Accuracy = 90% (36/40) (classification)
Accuracy = 95% (38/40) (classification)
Accuracy = 95% (38/40) (classification)
```

In [117]:
```
Y = train_Y.drop([i for i in range(120, 160)]).to_numpy()
X = train_X.drop([i for i in range(120, 160)]).to_numpy()
m_1 = train(Y, X, "-s 0 -c 0.0001 -e 0.000001")
m_2 = train(Y, X, "-s 0 -c 0.01 -e 0.000001")
m_3 = train(Y, X, "-s 0 -c 1 -e 0.000001")
m_4 = train(Y, X, "-s 0 -c 100 -e 0.000001")
m_5 = train(Y, X, "-s 0 -c 10000 -e 0.000001")
```

In [118]:
```
fold_1 = []
Y = train_Y[120:160].to_numpy()
X = train_X[120:160].to_numpy()
p_labels, p_acc, p_vals = predict(Y, X, m_1)
fold_1.append(p_acc[0])
```

```
p_labels, p_acc, p_vals = predict(Y, X, m_2)
fold_1.append(p_acc[0])
p_labels, p_acc, p_vals = predict(Y, X, m_3)
fold_1.append(p_acc[0])
p_labels, p_acc, p_vals = predict(Y, X, m_4)
fold_1.append(p_acc[0])
p_labels, p_acc, p_vals = predict(Y, X, m_5)
fold_1.append(p_acc[0])
ACC_list.append(fold_1)
```

```
Accuracy = 40% (16/40) (classification)
Accuracy = 72.5% (29/40) (classification)
Accuracy = 82.5% (33/40) (classification)
Accuracy = 85% (34/40) (classification)
Accuracy = 77.5% (31/40) (classification)
```

In [119]:

```
Y = train_Y.drop([i for i in range(160, 200)]).to_numpy()
X = train_X.drop([i for i in range(160, 200)]).to_numpy()
m_1 = train(Y, X, "-s 0 -c 0.0001 -e 0.000001")
m_2 = train(Y, X, "-s 0 -c 0.01 -e 0.000001")
m_3 = train(Y, X, "-s 0 -c 1 -e 0.000001")
m_4 = train(Y, X, "-s 0 -c 100 -e 0.000001")
m_5 = train(Y, X, "-s 0 -c 10000 -e 0.000001")
```

In [120]:

```
fold_1 = []
Y = train_Y[160:].to_numpy()
X = train_X[160:].to_numpy()
p_labels, p_acc, p_vals = predict(Y, X, m_1)
fold_1.append(p_acc[0])
p_labels, p_acc, p_vals = predict(Y, X, m_2)
fold_1.append(p_acc[0])
p_labels, p_acc, p_vals = predict(Y, X, m_3)
fold_1.append(p_acc[0])
p_labels, p_acc, p_vals = predict(Y, X, m_4)
fold_1.append(p_acc[0])
p_labels, p_acc, p_vals = predict(Y, X, m_5)
fold_1.append(p_acc[0])
ACC_list.append(fold_1)
```

```
Accuracy = 45% (18/40) (classification)
Accuracy = 77.5% (31/40) (classification)
Accuracy = 85% (34/40) (classification)
Accuracy = 95% (38/40) (classification)
Accuracy = 90% (36/40) (classification)
```

In [122]:
```
ACC_df = pd.DataFrame(ACC_list)
```

In [124]:
```
ACC_df.mean()
```

Out[124]:
```
0    48.0
1    80.5
2    86.0
3    87.5
4    85.0
dtype: float64
```