CS 156a Problem Set 8

Amitesh Pandey

November 2024

Primal versus Dual Problem

Problem 1

Recall that the formulation is

$$\min\left(\frac{1}{2}\mathbf{w}^T\mathbf{w}\right)$$

subject to $y_n(\mathbf{w}^T x_n + b) \geq 1, \forall n$. The components of **w** and the bias b compose of all the variables in this problem. Thus it is a $d+1$ variable quadratic programming problem. Option **[d]** is correct.

Note: Code for all following problems is attached at the end.

Polynomial Kernels

Problem 2

We can see that with $E_{\text{in}} = 0.11$, **[a]**, 0 versus all, has the highest E_{in}.

Problem 3

We can see that with $E_{\text{in}} = 0.021$, **[a]**, 1 versus all, has the highest E_{in}.

Problem 4

We can see that 0 versus all has a vector size of 2179, whereas 1 versus all has a vector size of 386, so 2179 - 386 = 1793, which is closest to option **[c]**.

Problem 5

We can see that the maximum $C = 1$ achieves the lowest $E_{\text{in}} = 0.0032$, so option **[d]** is correct.

Problem 6

At $C = 0.001$, $Q = 2$ has a support vector of size 76 and $Q = 5$ has a vector of size 25, so **[b]** is correct.

Cross Validation

Problem 7

Out of the $100 \times 10 = 1000$ folds, we can see that $C = 0.001$ was selected the most times, by having the lowest error 526 times. So **[b]** is correct.

Problem 8

We get the error for $C = 0.001$ as 0.0047, closest to 0.005 so **[c]** is correct.

RBF Kernel

Problem 9

With $E_{\text{in}} = 0.006$, it was $C = 10^6$ that achieved the lowest E_{in}. So option **[e]** is correct.

Problem 10

With $E_{\text{out}} = 0.018$, it was $C = 100$ that achieved the lowest E_{out}. So option **[c]** is correct.

Problems 2-10

[161]:

```python
import numpy as np
from sklearn import svm, model_selection
from sklearn.model_selection import RepeatedStratifiedKFold
```

[9]:

```python
train_data = np.loadtxt('features.train')
test_data = np.loadtxt('features.test')
```

[152]:

```python
x_train = train_data[:,1:]
y_train = train_data[:,0]

x_test = test_data[:,1:]
y_test = test_data[:,0]
```

[24]:

```python
def binarize(posClass, yvals):
    return np.array([1 if classval == posClass else -1 for classval in yvals])
```

[163]:

```python
C, Q = [0.01, 2]
avg_vec_size = 0
for classval in range(10):
    binary_y_train = binarize(classval, y_train)
    binary_y_test = binarize(classval, y_test)
    model = svm.SVC(kernel='poly', C=C, degree=Q, gamma=1.0, coef0=1.0)
    model.fit(x_train, binary_y_train)
    print('-'*25, end = '\n')
    print('Classifier: ' + str(classval) + ' versus all')
    print('Ein: ' + str(1 - model.score(x_test, binary_y_test)))
    print('Support vector size: ' + str(sum(model.n_support_)))
```

```
-------------------------
Classifier: 0 versus all
Ein: 0.11160936721474835
Support vector size: 2179
-------------------------
Classifier: 1 versus all
Ein: 0.0219232685600399
Support vector size: 386
-------------------------
Classifier: 2 versus all
Ein: 0.09865470852017932
Support vector size: 1970
-------------------------
Classifier: 3 versus all
Ein: 0.08271051320378675
Support vector size: 1964
-------------------------
Classifier: 4 versus all
```

```
Ein: 0.09965122072745392
Support vector size: 1856
-------------------------
Classifier: 5 versus all
Ein: 0.07972097658196309
Support vector size: 1585
-------------------------
Classifier: 6 versus all
Ein: 0.08470353761833582
Support vector size: 1893
-------------------------
Classifier: 7 versus all
Ein: 0.07324364723467858
Support vector size: 1704
-------------------------
Classifier: 8 versus all
Ein: 0.08271051320378675
Support vector size: 1776
-------------------------
Classifier: 9 versus all
Ein: 0.08819133034379667
Support vector size: 1978
```

[30]:

```
print(2179 - 386)
```

```
1793
```

[153]:

```
def retain_classes(xvals, yvals, classOne, classTwo):
    idxs = []
    for i in range(yvals.shape[0]):
        if yvals[i] not in [classOne, classTwo]:
            idxs.append(i)
    new_y_vals = np.delete(yvals, idxs)
    new_x_vals = np.delete(xvals, idxs, 0)
    return (new_x_vals, new_y_vals)
```

[165]:

```
Q = 2
Cvals = [0.001, 0.01, 0.1, 1]

for C in Cvals:
    x_train_1v5, y_train_1v5 = retain_classes(x_train, y_train, 1, 5)
    x_test_1v5, y_test_1v5 = retain_classes(x_test, y_test, 1, 5)
    model = svm.SVC(kernel='poly', C=C, degree=Q, gamma=1.0, coef0=1.0)
    model.fit(x_train_1v5, y_train_1v5)
    print('-'*25)
    print('For C = ' + str(C), end = '\n')
    print('Support Vector Size: ' + str(sum(model.n_support_)))
    print('Ein: ' + str(1 - model.score(x_train_1v5, y_train_1v5)))
    print('Eout: ' + str(1 - model.score(x_test_1v5, y_test_1v5)))
```

```
-------------------------
For C = 0.001
Support Vector Size: 76
Ein: 0.004484304932735439
Eout: 0.01650943396226412
-------------------------
For C = 0.01
Support Vector Size: 34
Ein: 0.004484304932735439
Eout: 0.018867924528301883
-------------------------
For C = 0.1
Support Vector Size: 24
Ein: 0.004484304932735439
Eout: 0.018867924528301883
-------------------------
For C = 1
Support Vector Size: 24
Ein: 0.0032030749519538215
Eout: 0.018867924528301883
```

[166]:

```
Qvals = [2, 5]
Cvals = [0.001, 0.01, 0.1, 1]

for Q in Qvals:
    for C in Cvals:
        x_train_1v5, y_train_1v5 = retain_classes(x_train, y_train, 1, 5)
        x_test_1v5, y_test_1v5 = retain_classes(x_test, y_test, 1, 5)
        model = svm.SVC(kernel='poly', C=C, degree=Q, gamma=1.0, coef0=1.0)
        model.fit(x_train_1v5, y_train_1v5)
        print('-'*25)
        print('For C = ' + str(C) + ' and Q = ' + str(Q), end = '\n')
        print('Support Vector Size: ' + str(sum(model.n_support_)))
        print('Ein: ' + str(1 - model.score(x_test_1v5, y_test_1v5)))
```

```
-------------------------
For C = 0.001 and Q = 2
Support Vector Size: 76
Ein: 0.01650943396226412
-------------------------
For C = 0.01 and Q = 2
Support Vector Size: 34
Ein: 0.018867924528301883
-------------------------
For C = 0.1 and Q = 2
Support Vector Size: 24
```

```
Ein: 0.018867924528301883
-------------------------
For C = 1 and Q = 2
Support Vector Size: 24
Ein: 0.018867924528301883
-------------------------
For C = 0.001 and Q = 5
Support Vector Size: 25
Ein: 0.02122641509433964б
-------------------------
For C = 0.01 and Q = 5
Support Vector Size: 23
Ein: 0.021226415094339646
-------------------------
For C = 0.1 and Q = 5
Support Vector Size: 25
Ein: 0.018867924528301883
-------------------------
For C = 1 and Q = 5
Support Vector Size: 21
Ein: 0.021226415094339646
```

[167]:

```
Q = 2
rskf = RepeatedStratifiedKFold(n_splits=10, n_repeats=100)
Cvalss = [0.0001, 0.001, 0.01, 0.1, 1]
chosenct = []
for idxt, idxv in rskf.split(x_train_1v5, y_train_1v5):
    x_train_iter, x_val_iter = x_train_1v5[idxt], x_train_1v5[idxv]
    y_train_iter, y_val_iter = y_train_1v5[idxt], y_train_1v5[idxv]
    mostaccurate = 0
    chosenOne = None
    for C in Cvals:
        model = svm.SVC(kernel='poly', C=C, degree=Q, gamma=1.0, coef0=1.0)
        model.fit(x_train_iter, y_train_iter)
        acc = model.score(x_val_iter, y_val_iter)
        if acc > mostaccurate:
            mostaccurate = acc
            chosenOne = C
    chosenct.append(chosenOne)
```

[117]:

```
for C in Cvals:
    print('C: ' + str(C) + ' was chosen ' + str(chosenct.count(C)) + ' times')
```

```
C: 0.0001 was chosen 364 times
C: 0.001 was chosen 526 times
C: 0.01 was chosen 45 times
C: 0.1 was chosen 25 times
C: 1 was chosen 40 times
```

[120]:

```
C = 0.001
rskf = RepeatedStratifiedKFold(n_splits=10, n_repeats=100)
model = svm.SVC(kernel='poly', C=C, degree=Q, gamma=1.0, coef0=1.0)
1 - model_selection.cross_val_score(model, x_train_1v5, y_train_1v5, cv=rskf).
 ↪mean()
```

[120]:

```
0.004753143883717059
```

[160]:

```
x_train_1v5, y_train_1v5 = retain_classes(x_train, y_train, 1, 5)
x_test_1v5, y_test_1v5 = retain_classes(x_test, y_test, 1, 5)
Cvals = [0.01, 1, 100, 10000, 1000000]
for C in Cvals:
    model = svm.SVC(kernel='rbf', C=C, degree=Q, gamma=1.0)
    model.fit(x_train_1v5, y_train_1v5)
    print('-'*25, end = '\n')
    print('C: ' + str(C))
    print('Ein: ' +  str(1 - model.score(x_train_1v5, y_train_1v5)))
    print('Eout: ' + str(1 - model.score(x_test_1v5, y_test_1v5)))
```

```
-------------------------
C: 0.01
Ein: 0.0038436899423446302
Eout: 0.02358490566037741
-------------------------
C: 1
Ein: 0.004484304932735439
Eout: 0.021226415094339646
-------------------------
C: 100
Ein: 0.0032030749519538215
Eout: 0.018867924528301883
-------------------------
C: 10000
Ein: 0.002562459961563124
Eout: 0.02358490566037741
-------------------------
C: 1000000
Ein: 0.0006406149903908087
Eout: 0.02358490566037741
```