Machine Learning **(Due: 04/12/20)**

Homework #4: Referred Answers

Instructor: Lin, Hsuan Tien *Name:* Hao-Cheng Lo, *Id:* D08227104

Problem 1:

[c] is correct.

Define the error between target function and the hypothesis in $[0, 2]$ is:

$$\int_0^2 \left(e^x - wx\right)^2 dx = \frac{8}{3}w^2 - 2w(e^2+1) + \frac{e^4-1}{2}$$

To find the minimal value of $\int_0^2 \left(e^x - wx\right)^2 dx$, let $\frac{d}{dw}(\int_0^2 \left(e^x - wx\right)^2 dx) = 0$:

$$\frac{d}{dw}(\frac{8}{3}w^2 - 2w(e^2+1) + \frac{e^4-1}{2}) = 0$$
$$\frac{16}{3}w = 2(e^2+1)$$
$$w = \frac{3e^2+3}{8}$$

Accordingly, the answer is [c].

Problem 2:

[b] is correct.

Given that we choose a hypothesis from a certain algorithm that makes E_{in} reach minimal, it is possible that $E_{in} < E_{out}$ under the same hypothesis and algorithm. Moreover, given hoeffding's inequality, we know that when N increases, the distance between E_{in} and E_{out} is decreasing. Hence, it is possible that $E_{in} = E_{out}$. In a similar vein, we can regard E_{in}'s sample is sampled from E_{out}'s population. Accordingly, the sample's error/deviation is restricted by the error/deviation of population. Therefore, it is impossible that $E_{in} > E_{out}$. Taken together, 1 statement is false.

Problem 3:

[d] is correct.

Denote $x_i = [x_{i1}, x_{i2}, ..., x_{id+1}],\ \forall i \in \{1, ..., d+1\}$.

Let $A = E(X_h^T X_h)$, for any element a_{ij}:

$a_{ij} = E(\sum_{k=1}^{N} x_{ki}x_{kj} + \sum_{k=1}^{N} \tilde{x_{ki}}\tilde{x_{kj}})$

$a_{ij} = \sum_{k=1}^{N} x_{ki}x_{kj} + E(\sum_{k=1}^{N}(x_{ki} + \epsilon_i)(x_{kj} + \epsilon_j))$

$a_{ij} = \sum_{k=1}^{N} x_{ki}x_{kj} + \sum_{k=1}^{N} x_{ki}x_{kj} + \sum_{k=1}^{N} x_{ki}E(\epsilon_j) + \sum_{k=1}^{N} x_{kj}E(\epsilon_i) + \sum_{k=1}^{N} E(\epsilon_j\epsilon_i)$

$a_{ij} = 2\sum_{k=1}^{N} x_{ki}x_{kj} + \sum_{k=1}^{N} x_{ki}0 + \sum_{k=1}^{N} x_{kj}0 + NCov(\epsilon_j, \epsilon_i)$

Hence, $E(X_h^T X_h) = A = 2X^T X + N\sigma^2 I_{d+1}$

Problem 4:

[e] is correct.

Denote $x_i = [x_{i1}, x_{i2}, ..., x_{id+1}],\ \forall i \in \{1, ..., d+1\}$ and $y = [y_1, y_2, ..., y_{d+1}]$.

Let $A = E(X_h^T y_h)$, for any element a_i:

$a_i = E(\sum_{k=1}^{N} x_{ki} y_k + \sum_{k=1}^{N} \tilde{x_{ki}} y_k)$

$a_i = \sum_{k=1}^{N} x_{ki} y_k + \sum_{k=1}^{N} E((x_{ki} + \epsilon_i) y_k)$

$a_i = 2\sum_{k=1}^{N} x_{ki} y_k + \sum_{k=1}^{N} E(\epsilon_i) y_k$

$a_i = 2\sum_{k=1}^{N} x_{ki} y_k + \sum_{k=1}^{N} 0 y_k$

Hence, $E(X_h^T y_h) = A = 2X^T y$

Problem 5:

[d] is correct.

$\mathbf{w}_{\text{lin}} = \mathbf{v} = (Z^T Z)^{-1} Z^T y = (Q^T X^T X Q)^{-1} Z^T y = (Q^T Q \Gamma Q^T Q)^{-1} Z^T y = (\Gamma)^{-1} Z^T y$

$\mathbf{w}_{\text{reg}} = \mathbf{u} = (Z^T Z + \lambda I)^{-1} Z^T y = (\Gamma + \lambda I)^{-1} Z^T y$

Denote $Z^T = \begin{bmatrix} - & z_1^T & - \\ \vdots & \vdots & \vdots \\ - & z_{d+1}^T & - \end{bmatrix}$

Therefore, $u_i / v_i = (\gamma_i + \lambda)^{-1} z_i^T y / (\gamma_i)^{-1} z_i^T y = \dfrac{\gamma_i}{\gamma_i + \lambda}$

Problem 6:

[a] is correct.

I employee the Lagrange function with inequality constraints.

The Lagrange function:

$$L(w,\nu) = \frac{1}{N}\sum(wx_n - y_n)^2 + \nu(w^2 - C)$$

The KKT conditions are: (1) $\nu > 0$; (2) $\dfrac{\partial L(w,\nu)}{\partial w} = 0$; (3) $w^2 \leq C$; (4) $\nu(w^2 - C) = 0$.

$\because$ (1) and (4) $\therefore w = \pm\sqrt{C}$

$(2) = \dfrac{\partial L(w,\nu)}{\partial w}(\dfrac{1}{N}\sum(w^2x_n^2 - 2wx_ny_n + y_n^2) + \nu w^2 - \nu C) = (\dfrac{2}{N}\sum x_n^2)w - (\dfrac{2}{N}\sum x_ny_n) + 2\nu w = 0$

$w = \dfrac{N^-\sum x_ny_n}{N^-\sum x_n^2 + \nu} = \dfrac{\sum x_ny_n}{\sum x_n^2 + \lambda} = \pm\sqrt{C}$

Hence, $C = (\dfrac{\sum x_ny_n}{\sum x_n^2 + \lambda})^2$

Problem 7:

[d] is correct.

Suppose $\Omega(y) = (y + C)^2,\ C \in R$, $\Omega'(y) = 2y + 2C$

$$\frac{\partial f(y)}{\partial y} = \frac{1}{N}\sum 2(y - y_n) + \frac{2K}{N}\Omega'(y) = 2y - \frac{2}{N}\sum y_n + \frac{2K}{N}\Omega'(y) = 0$$

$$\Omega'(y) = 2y + 2C = \frac{\sum y_n - Ny}{K}$$

$$C = \frac{\sum y_n - (N + 2K)y}{2K}$$

Substitute y with the optimal solution.

$$C = \frac{\Sigma - (N + 2K)\dfrac{\Sigma + K}{N + 2K}}{2K} = \frac{\Sigma - (\Sigma + K)}{2K} = -0.5$$

Problem 8:

[b] is correct.

Note that the SVD of $X = UDV^T$, so $X^TX = (UDV^T)^T(UDV^T) = VD^2V^T$. In addition, for the whole data set, denote the transformation $\Phi(X) = X\Gamma^{-1} = XQ$.

In Z-space,

$$\begin{aligned}
\hat{y} &= Z(Z^TZ + \lambda I)^{-1}Z^Ty \\
&= XQ(Q^TX^TXQ + \lambda I)^{-1}Q^TX^Ty \\
&= UDV^TQ(Q^TVD^2V^TQ + \lambda I)^{-1}Q^TVDU^Ty \\
&= UDV^TQ(QVD^2V^TQ + \lambda I)^{-1}QVDU^Ty \\
&= UDV^T(Q^{-1}(QVD^2V^TQ + \lambda I)Q^{-1})^{-1}VDU^Ty \\
&= UDV^T(VD^2V^T + \lambda Q^{-1}Q^{-1})^{-1}VDU^Ty \\
&= X(X^TX + \lambda Q^{-1}Q^{-1})^{-1}X^Ty \quad \text{(In X-space)}
\end{aligned} \tag{0.1}$$

Accordingly, $\lambda\nabla_w w^T\Gamma^2 w = \lambda\nabla_w w^TQ^{-2}w = \lambda Q^{-2}w$. Hence, the answer is $w^T\Gamma^2 w$.

Problem 9:

[b] is correct.

Let $\nabla_w \frac{1}{N+K}(||Xw - Y||^2 + ||\tilde{X}w - \tilde{Y}||^2) = 0$.

$\rightarrow \frac{2}{N+K}(X^T Xw - X^T Y + \tilde{X}^T \tilde{X}w - \tilde{X}^T \tilde{Y}) = 0$

$\rightarrow w = (X^T X + \tilde{X}^T \tilde{X})^{-1}(X^T Y + \tilde{X}^T \tilde{Y})$

Based on the optimization problem of the problem 9 form, the $\mathbf{w}_{\text{reg}} = (X^T X + \lambda B)^{-1} X^T y$. Hence, $\tilde{X} = \sqrt{\lambda B}$ and $\tilde{y} = \mathbf{0}$.

Problem 10:

[e] is correct.

Suppose we randomly choose a positive example (resp. negative example) s as validation data. The remaining training data would be $N-1$ positive example (resp. negative example) and N negative example (resp. positive example). Because A always predicts the majority class, $h(s) = -1$ ($h(s) = +1$), which is different from the value of s. That is, A on one example is always wrong which indicates $e_i = 1$. Taken together, $E_{loocv} = \frac{N}{N} = 1$.

Problem 11:

[c] is correct.

Given N points $= \{(x_1, y_1), (x_2, y_2)..., (x_N, y_N)\}$ which is increasingly ordered with xs' values. WLOG, let $y_{i \leq p} = -1$ and $y_{j>p} = +1$ where $p \in \{2, ..., N-2\}$.

We can find that the hypotheses hs in leave one out process would be (1) $h_a = \frac{x_p + x_{p+1}}{2}$ when the points of (x_i, y_i) where $i \in \{1, ..., p-1, p+2, ..., N\}$ are taken as validation data, which error is 0 due to correct classification; (2) $h_b = \frac{x_{p-1} + x_{p+1}}{2}$ when the point of (x_p, y_p) is taken as validation data, which error may be 1 due to wrong classification (e.g. $x_{p-1} = 2,\ x_p = 5,\ x_{p+1} = 6$); (3) $h_c = \frac{x_p + x_{p+2}}{2}$ when the point of (x_{p+1}, y_{p+1}) is taken as validation data, which error may be 1 due to wrong classification (e.g. $x_p = 5,\ x_{p+1} = 6,\ x_{p+2} = 8$). Taken together, $E_{loocv} = \frac{2}{N}$.

Problem 12:

[e] is correct.

Take p_1 as val data:

The linear hypothesis is $2x - \rho y - 3y + 6 = 0$. The error of p_1 in linear hypothesis is $(y - \hat{y})^2 = (0 - \frac{12}{\rho + 3})^2 = 144/(\rho+3)^2$; The constant hypothesis is $y = 1$. The error of p_1 in constant hypothesis is $(y - \hat{y})^2 = (0-1)^2 = 1$.

Take p_2 as val data:

The linear hypothesis is $y = 0$. The error of p_2 in linear hypothesis is $(y - \hat{y})^2 = (2 - 0)^2 = 4$; The constant hypothesis is $y = 0$. The error of p_2 in constant hypothesis is $(y - \hat{y})^2 = (2 - 0)^2 = 4$.

Take p_3 as val data:

The linear hypothesis is $-2x + \rho y - 3y + 6 = 0$. The error of p_3 in linear hypothesis is $(y - \hat{y})^2 = (0 - \frac{-12}{\rho - 3})^2 = 144/(\rho-3)^2$; The constant hypothesis is $y = 1$. The error of p_3 in constant hypothesis is $(y - \hat{y})^2 = (0-1)^2 = 1$.

Solve $(\frac{12}{\rho + 3})^2 + (\frac{12}{\rho - 3})^2 = 2$, one of solution is $\rho = 3\sqrt{9 + 4\sqrt{6}}$.

Problem 13:

[d]

Given K iid examples, we have:

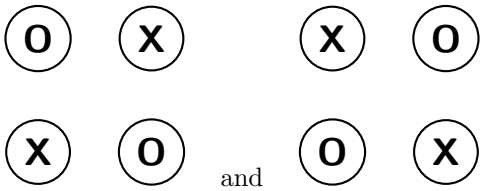
$$\begin{aligned} V[E_{val}(h)] &= V[\frac{1}{K}\sum_{i=1}^{K} err(h(\mathbf{x}_i), y_i)] \\ &= \frac{1}{K^2}\sum_{i=1}^{K} V[err(h(\mathbf{x}_i), y_i)] \\ &= \frac{K}{K^2} V[err(h(\mathbf{x}), y)] \end{aligned} \tag{0.2}$$

Hence, the answer is $\frac{K}{K^2} = \frac{1}{K}$.

Problem 14:

[c]

In 16 combinations, there are 2 combination that is:



and

In these 2 combinations, there are at least 2 vertices wrongly classified; in other 14 combinations, there are at least 0 vertex wrongly classified. Hence, in total of 64 vertices, there are expected at least 2 vertices wrongly classified. So, the answer is $\frac{2}{64}$.

Problem 15:

[a]

$E_{\text{out}}(g) = P(y = +1)\epsilon_+ + P(y = -1)\epsilon_- = p\epsilon_+ + (1-p)\epsilon_- = E_{\text{out}}(g_c) = 1 - p$

$\rightarrow p\epsilon_+ - p\epsilon_- + p + \epsilon_- = 1$

$\rightarrow p = \dfrac{1 - \epsilon_-}{\epsilon_+ - \epsilon_- + 1}$

In [1]:

```
import numpy as np
import scipy
from liblinearutil import *
```

Data Preprocessing

In [2]:

```
# data loading
traindta = np.loadtxt("hw4_train.dat", dtype=np.float, delimiter=' ')
train_x = traindta[:, 0:6]
train_y = traindta[:, 6]
```

In [3]:

```
# data loading
testdta = np.loadtxt("hw4_test.dat", dtype=np.float, delimiter=' ')
test_x = testdta[:, 0:6]
test_y = testdta[:, 6]
```

In [4]:

```
# feature tranformation
from sklearn.preprocessing import PolynomialFeatures
poly = PolynomialFeatures(2)
train_tx = poly.fit_transform(train_x)
test_tx = poly.fit_transform(test_x)
```

In [5]:

```
np.shape(train_tx)
```

Out[5]:

```
(200, 28)
```

In [6]:

```
np.shape(test_tx)
```

Out[6]:

```
(300, 28)
```

Relation of λ and C

Set $\frac{\lambda}{N} = \frac{1}{2}$. Set $C = \frac{1}{N}$.

Therefore, $C = \frac{1}{2\lambda}$.

$\log \lambda \in \{-4, -2, 0, 2, 4\} \rightarrow C \in \{5000, 50, 0.5, 0.005, 0.00005\}$

Q16

In [19]:

```
c = [5000,50,0.5,0.005,0.00005]
e = 0.000001

for i in c:

    prob = problem(train_y, train_tx)
    param = parameter('-s 0 -c {c} -e {e}'.format(c = i, e = e))
    m = train(prob,param)
    p_labs, p_acc, p_vals = predict(test_y, test_tx, m)
    print("When c =", i,"The error in test data is", 100-p_acc[0])
```

```
Accuracy = 86.6667% (260/300) (classification)
When c = 5000 The error in test data is 13.333333333333329
Accuracy = 87% (261/300) (classification)
When c = 50 The error in test data is 13.0
Accuracy = 80.6667% (242/300) (classification)
When c = 0.5 The error in test data is 19.333333333333343
Accuracy = 74.3333% (223/300) (classification)
When c = 0.005 The error in test data is 25.66666666666667
Accuracy = 51.6667% (155/300) (classification)
When c = 5e-05 The error in test data is 48.33333333333333
```

In [80]:

```
# SKLEARN VERSION
from sklearn.linear_model import LogisticRegression

c = [5000,50,0.5,0.005,0.00005]
e = 0.000001

for i in c:

    lr_model = LogisticRegression(C = i, penalty = 'l2', tol = e, solver ='liblinear')
    w = lr_model.fit(train_tx, train_y)
    print(w.score(test_tx, test_y))
```

```
0.8666666666666667
0.8733333333333333
0.82
0.7466666666666667
0.52
```

Thus, when $C = 50$ ($\log \lambda = -2$), the 0/1 accuracy is highest. [b]

Q17

In [23]:

```
c = [5000,50,0.5,0.005,0.00005]
e = 0.000001

for i in c:

    prob = problem(train_y, train_tx)
    param = parameter('-s 0 -c {c} -e {e}'.format(c = i, e = e))
    m = train(prob,param)
    p_labs, p_acc, p_vals = predict(train_y, train_tx, m)
    print("When c =", i,"The error in train data is", 100-p_acc[0])
    ACC, MSE, SCC = evaluations(train_y, p_labs)
    print("Error:",100-ACC)
```

```
Accuracy = 91% (182/200) (classification)
When c = 5000 The error in train data is 9.0
Error: 9.0
Accuracy = 90% (180/200) (classification)
When c = 50 The error in train data is 10.0
Error: 10.0
Accuracy = 87% (174/200) (classification)
When c = 0.5 The error in train data is 13.0
Error: 13.0
Accuracy = 80.5% (161/200) (classification)
When c = 0.005 The error in train data is 19.5
Error: 19.5
Accuracy = 46.5% (93/200) (classification)
When c = 5e-05 The error in train data is 53.5
Error: 53.5
```

In [81]:

```
# SKLEARN VERSION
from sklearn.linear_model import LogisticRegression

c = [5000,50,0.5,0.005,0.00005]
e = 0.000001

for i in c:

    lr_model = LogisticRegression(C = i, penalty = 'l2', tol = e, solver ='liblinear')
    w = lr_model.fit(train_tx, train_y)
    print(w.score(train_tx, train_y))
```

```
0.91
0.905
0.875
0.805
0.465
```

Thus, when $C = 5000$ ($\log \lambda = -4$), the 0/1 accuracy is highest. [a]

Q18

In [16]:

```
subtrain_tx = train_tx[0:120,:]
subval_tx = train_tx[120:,:]
subtrain_y = train_y[0:120]
subval_y = train_y[120:]
```

In [24]:

```
c = [5000,50,0.5,0.005,0.00005]
e = 0.000001

for i in c:

    prob = problem(subtrain_y, subtrain_tx)
    param = parameter('-s 0 -c {c} -e {e}'.format(c = i, e = e))
    m = train(prob,param)
    p_labs, p_acc, p_vals = predict(subval_y, subval_tx, m)
    p_labs2, p_acc2, p_vals2 = predict(test_y, test_tx, m)
    print("When c =", i,"The err in val data is", 100-p_acc[0],"The err in test data is", 1
```

```
Accuracy = 80% (64/80) (classification)
Accuracy = 82.3333% (247/300) (classification)
When c = 5000 The err in val data is 20.0 The err in test data is 17.6666666
66666657
Accuracy = 86.25% (69/80) (classification)
Accuracy = 85.6667% (257/300) (classification)
When c = 50 The err in val data is 13.75 The err in test data is 14.33333333
3333329
Accuracy = 76.25% (61/80) (classification)
Accuracy = 76% (228/300) (classification)
When c = 0.5 The err in val data is 23.75 The err in test data is 24.0
Accuracy = 73.75% (59/80) (classification)
Accuracy = 76.3333% (229/300) (classification)
When c = 0.005 The err in val data is 26.25 The err in test data is 23.66666
666666667
Accuracy = 42.5% (34/80) (classification)
Accuracy = 51.6667% (155/300) (classification)
When c = 5e-05 The err in val data is 57.5 The err in test data is 48.333333
33333333
```

In [82]:

```
# SKLEARN VERSION
from sklearn.linear_model import LogisticRegression

c = [5000,50,0.5,0.005,0.00005]
e = 0.000001

for i in c:

    lr_model = LogisticRegression(C = i, penalty = 'l2', tol = e, solver ='liblinear')
    w = lr_model.fit(subtrain_tx, subtrain_y)
    print(w.score(subval_tx, subval_y), w.score(test_tx, test_y))
```

```
0.8125 0.8266666666666667
0.8625 0.8633333333333333
0.775 0.7766666666666666
0.7625 0.7666666666666667
0.425 0.5166666666666667
```

Cause Min $E_{val} = 13.75$ whose $\log \lambda = -2$, by the w^- the validation data indicates, the corresponding $E_{out}(w^-) = 14.3\%$. [e]

Q19

In [26]:

```
c = 50
e = 0.000001

prob = problem(train_y, train_tx)
param = parameter('-s 0 -c {c} -e {e}'.format(c = c, e = e))
m = train(prob,param)
p_labs, p_acc, p_vals = predict(test_y, test_tx, m)
print("When c =", c,"The error in test data is", 100-p_acc[0])
```

```
Accuracy = 87% (261/300) (classification)
When c = 50 The error in test data is 13.0
```

Using $\log \lambda = -2$, by the w the whole training data indicates, the corresponding $E_{out}(w) = 13.0\%$. [d]

Q20

In [72]:

```
c = [5000,50,0.5,0.005,0.00005]
e = 0.000001


f1x = train_tx[0:40,:]
f1y = train_y[0:40]
f2x = train_tx[40:80,:]
f2y = train_y[40:80]
f3x = train_tx[80:120,:]
f3y = train_y[80:120]
f4x = train_tx[120:160,:]
f4y = train_y[120:160]
f5x = train_tx[160:200,:]
f5y = train_y[160:200]

fx = [f1x,f2x,f3x,f4x,f5x]
fy = [f1y,f2y,f3y,f4y,f5y]
```

In [74]:

```
for i in c:
    ttlerr = 0
    for k in range(5):
        tempx = np.concatenate((fx[(k+1)%5],fx[(k+2)%5],fx[(k+3)%5],fx[(k+4)%5]), axis=0)
        tempy = np.concatenate((fy[(k+1)%5],fy[(k+2)%5],fy[(k+3)%5],fy[(k+4)%5]), axis=0)
        prob = problem(tempy,tempx)
        param = parameter('-s 0 -c {c} -e {e}'.format(c = i, e = e))
        m = train(prob,param)
        p_labs, p_acc, p_vals = predict(fy[k], fx[k], m)
        ttlerr += 100-p_acc[0]
    print("When c =", i,"The error in CV is", ttlerr/5)
```

```
Accuracy = 87.5% (35/40) (classification)
Accuracy = 77.5% (31/40) (classification)
Accuracy = 95% (38/40) (classification)
Accuracy = 77.5% (31/40) (classification)
Accuracy = 90% (36/40) (classification)
When c = 5000 The error in CV is 14.5
Accuracy = 85% (34/40) (classification)
Accuracy = 80% (32/40) (classification)
Accuracy = 95% (38/40) (classification)
Accuracy = 85% (34/40) (classification)
Accuracy = 95% (38/40) (classification)
When c = 50 The error in CV is 12.0
Accuracy = 80% (32/40) (classification)
Accuracy = 90% (36/40) (classification)
Accuracy = 90% (36/40) (classification)
Accuracy = 80% (32/40) (classification)
Accuracy = 82.5% (33/40) (classification)
When c = 0.5 The error in CV is 15.5
Accuracy = 77.5% (31/40) (classification)
Accuracy = 92.5% (37/40) (classification)
Accuracy = 85% (34/40) (classification)
Accuracy = 75% (30/40) (classification)
Accuracy = 80% (32/40) (classification)
When c = 0.005 The error in CV is 18.0
Accuracy = 42.5% (17/40) (classification)
Accuracy = 65% (26/40) (classification)
Accuracy = 47.5% (19/40) (classification)
Accuracy = 40% (16/40) (classification)
Accuracy = 45% (18/40) (classification)
When c = 5e-05 The error in CV is 52.0
```

In [83]:

```
#in-built cross validation

c = [5000,50,0.5,0.005,0.00005]
e = 0.000001

for i in c:

    prob = problem(train_y, train_tx)
    param = parameter('-s 0 -c {c} -e {e} -v 5'.format(c = i, e = e))
    m = train(prob,param)
```

```
Cross Validation Accuracy = 86%
Cross Validation Accuracy = 88.5%
Cross Validation Accuracy = 85.5%
Cross Validation Accuracy = 81.5%
Cross Validation Accuracy = 50.5%
```

Hence, $\text{Min } E_{VC} = 12\%$. [c]

In [ ]:

```

```