HW #5

1. $L(w) = w_1^2 + 2w_2^2 + w_3^2 + 4 - 2w_3w_4 + w_4^2 + 2w_1 - 4w_2 + 4$

a. $dL/dw_1 = dL/dw_2 = 2w_1 + 2$

$dL/dw_2 = 4w_2 - 4$

$dL/dw_3 = 2w_3 - 2w_4$

$dL/dw_4 = -2w_3 + 2w_4$

b. $\nabla L(w) = \begin{bmatrix} 2w_1+2 \\ 4w_2-4 \\ 2w_3-2w_4 \\ -2w_3+2w_4 \end{bmatrix}$ $\begin{bmatrix} dL/dw_1 \\ dL/dw_2 \\ dL/dw_3 \\ dL/dw_4 \end{bmatrix}$ $w = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$

c. $w_{t+1} = w_t - \eta \nabla L(w_t)$

$w_{t+1} = 0 - 0.1 \begin{bmatrix} 2(0)+2 \\ 4(0)-4 \\ 2(0)-2(0) \\ -2(0)+2(0) \end{bmatrix} = \begin{bmatrix} -0.2 \\ 0.4 \\ 0 \\ 0 \end{bmatrix}$

$= \begin{bmatrix} -0.2 \\ 0.4 \\ 0 \\ 0 \end{bmatrix}$

d. $\min(L(w))$

$\nabla L(w) = 0$

$\begin{bmatrix} 2w_1+2 \\ 4w_2-4 \\ 2w_3-2w_4 \\ -2w_3+2w_4 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$

$w_1 = -1 \quad w_2 = 1 \quad w_3 = w_4 \qquad w_3 = w_4 = x$

$L(w) = (-1)^2 + 2(1)^2 + x^2 - 2x^2 + x^2 + 2(-1) - 4(1)^2 + 4$

$= \boxed{1}$

e. No, there are infinitely many solutions with the form $\begin{bmatrix} -1 \\ 1 \\ x \\ x \end{bmatrix}$ $\forall x \in \mathbb{R}$

when $w_3 = w_4$.

2. $x^{(1)}, \ldots, x^{(n)} \in \mathbb{R}^d$, find $w \in \mathbb{R}^d$ that

minimizes $L(w) = \sum_{i=1}^{n} (w \cdot x^{(i)}) + \frac{1}{2C} \|w\|^2$

$C > 0$

a. $dL/dw_j = \sum_{i=1}^{n} x_j^{(i)} + \frac{1}{2C} \frac{dL}{dw_j} \|w\|^2$

$= \sum_{i=1}^{n} x_j^{(i)} + Cw_j$

b. $\nabla L(w) = \begin{bmatrix} \sum_{i=1}^{n} x_1^{(i)} + Cw_1 \\ \sum_{i=1}^{n} x_2^{(i)} + Cw_2 \\ \vdots \\ \sum_{i=1}^{n} x_d^{(i)} + Cw_d \end{bmatrix}$

$= \sum_{i=1}^{n} x^{(i)} + Cw$

c. $\nabla L(w) = 0$

$\sum_{i=1}^{n} x^{(i)} + Cw = 0$

$w = -\sum_{i=1}^{n} x^{(i)} = \boxed{-\frac{1}{C} \sum_{i=1}^{n} x^{(i)}}$

3. a. $L(w) = \sum_{i=1}^{n} (y^{(i)} - w \cdot x^{(i)})^2 + \lambda \|w\|^2$

$\nabla L(w)$ $\frac{dL}{dw_j} = \sum_{i=1}^{n} -2y^{(i)} x^{(i)} + 2(w \cdot x^{(i)}) x^{(i)}$

$= \sum_{i=1}^{n} 2w \cdot x^{(i)2} - y^{(i)} x^{(i)}$

$\nabla L(w) = \begin{bmatrix} 2\sum_{i=1}^{n} w_1 \cdot x_1^{(i)2} - y^{(i)} x^{(i)} \\ \vdots \\ 2\sum_{i=1}^{n} w_d \cdot x_d^{(i)2} - y^{(i)} x^{(i)} \end{bmatrix}$

ignoring intercept form

$\nabla L(w) = \begin{bmatrix} 2(\sum_{i=1}^{n} w_1 \cdot x^{(i)2} - y^{(i)} x^{(i)}) \\ \vdots \\ 2(\sum_{i=1}^{n} w_d \cdot x^{(i)2} - y^{(i)} x^{(i)}) \end{bmatrix}$

$= 2(\sum_{i=1}^{n} w \cdot x^{(i)2} -$

3. a. $L(w) = \sum_{i=1}^{n} (y^{(i)} - w \cdot x^{(i)})^2 + \lambda \|w\|^2$

$\frac{dL}{dw_j} \sum_{i=1}^{n} (-2y^{(i)} + 2 w \cdot x^{(i)}) x_j^{(i)} + 2\lambda w$

$\boxed{\nabla L(w) = 2 \sum_{i=1}^{n} (w \cdot x^{(i)} - y^{(i)}) x^{(i)} + 2\lambda w}$

b. $w_{t+1} = w_t + \eta_t \nabla L(w_t)$

$\boxed{w_{t+1} = w_t + 2\eta \sum_{i=1}^{n} (w_t \cdot x^{(i)} - y^{(i)}) x^{(i)} - 2\eta\lambda w_t}$

c. - set $w_0 = 0$
- for $t = 0, 1, 2, \ldots$ until convergence or fixed iterations
- ~~$w_{t+1} = w_t + 2\eta (\sum_{i=1}^{n} w_t \cdot x^{(i)})$~~
- randomly select a data point $(x^{(i)}, y^{(i)})$ from the dataset
- $w_{t+1} = w_t + 2\eta (w_t \cdot x^{(i)} - y^{(i)}) x^{(i)} - 2\eta\lambda w_t$
  - update w using gradient descent based off the one point
- get final vector w

4. a. $f''(x) = 2 > 0$ convex
b. $f''(x) = -2 < 0$ concave
c. $f''(x) = 2 > 0$ convex
d. $f''(x) = 0$ ~~neither~~ both
e. $f''(x) = 6x$ neither
f. $f''(x) = 12x^2 \geq 0$ convex
g. $f'' = -\frac{1}{x^2} < 0$ concave

Danny_Xia_HW5

February 9, 2025

[69]:
```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
from sklearn.feature_selection import SequentialFeatureSelector
import matplotlib.pyplot as plt
```

[70]:
```
df = pd.read_csv("heart.csv")
display(df)
```

```
     age  sex  cp  trestbps  chol  fbs  restecg  thalach  exang  oldpeak  \
0     63    1   3       145   233    1        0      150      0      2.3
1     37    1   2       130   250    0        1      187      0      3.5
2     41    0   1       130   204    0        0      172      0      1.4
3     56    1   1       120   236    0        1      178      0      0.8
4     57    0   0       120   354    0        1      163      1      0.6
..   ...  ...  ..       ...   ...  ...      ...      ...    ...      ...
298   57    0   0       140   241    0        1      123      1      0.2
299   45    1   3       110   264    0        1      132      0      1.2
300   68    1   0       144   193    1        1      141      0      3.4
301   57    1   0       130   131    0        1      115      1      1.2
302   57    0   1       130   236    0        0      174      0      0.0

     slope  ca  thal  target
0        0   0     1       1
1        0   0     2       1
2        2   0     2       1
3        2   0     2       1
4        2   0     2       1
..     ...  ..   ...     ...
298      1   0     3       0
299      1   0     3       0
300      1   2     3       0
301      1   1     3       0
302      1   1     2       0
```

```
[303 rows x 14 columns]
```

[71]:
```
X=df.drop('target', axis=1)
y=df['target']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=103/303,␣
 ↪random_state=42)
```

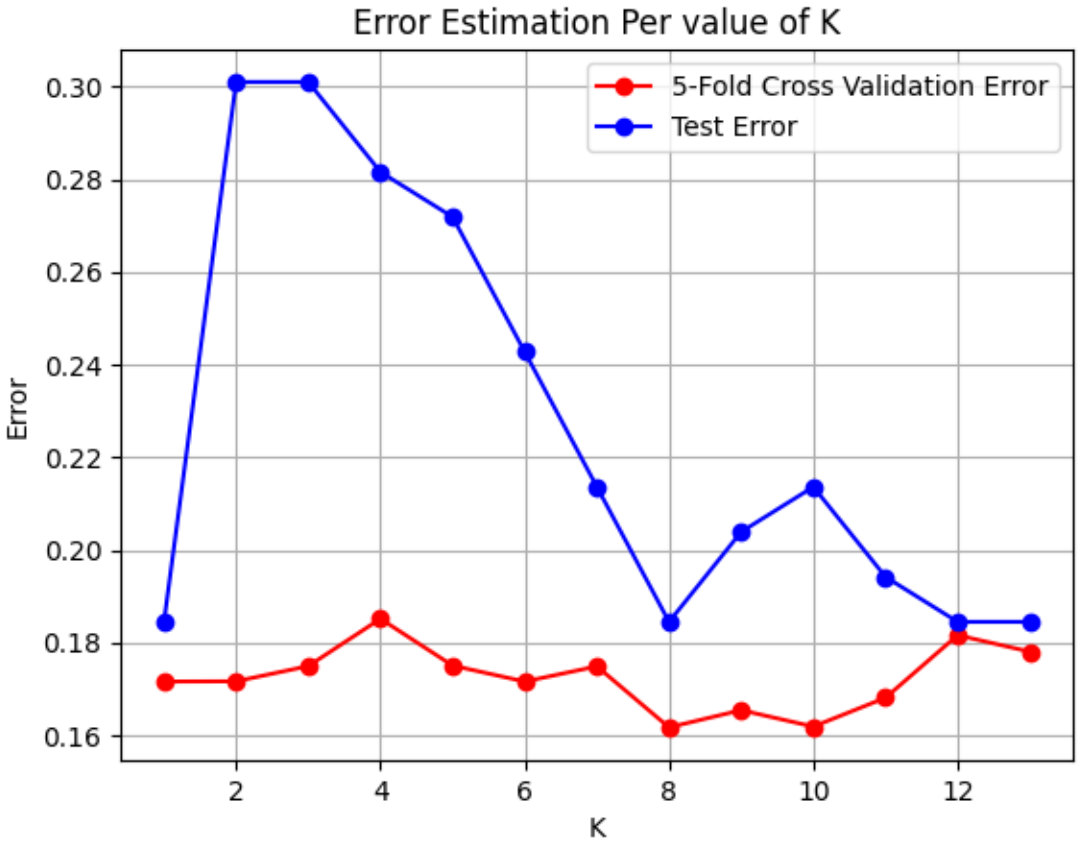
[72]:
```
def ErrorEstimate(X_train_selected,X_test_selected, y_train, y_test):
    model = LogisticRegression(max_iter=3000,penalty = None)
    model.fit(X_train_selected, y_train)

    kf= KFold(n_splits=5, shuffle=True)
    cv= cross_val_score(model, X, y, cv=kf)
    print(f'Error From K-Fold Cross-Validation with {k} features::')
    print(1 - np.mean(cv))
    error = 1 - np.mean(cv)
    accuracy = model.score(X_test_selected, y_test)
    print(f"Accuracy: {accuracy}")
    test_error = 1 - accuracy
    return error, test_error
```

[95]:
```
model = LogisticRegression(max_iter=3000, penalty='l1', solver='liblinear')
model.fit(X_train, y_train)

coefs = model.coef_[0]
abs_coefs = np.abs(coefs)

cv_errors = []
test_errors = []

for k in range(0,13):
    indices = np.argpartition(abs_coefs, -k)[-k:]

    X_train_selected = X_train.iloc[:, indices]
    X_test_selected = X_test.iloc[:, indices]

    if k == 2:
        best_columns = X_train.columns[indices].tolist()

    cross_error, test_error = ErrorEstimate(X_train_selected, X_test_selected,␣
 ↪y_train, y_test)
    cv_errors.append(cross_error)
    test_errors.append(test_error)


features=list(range(1,14))
```

```
plt.figure
plt.plot(features, cv_errors, marker='o',linestyle='-', color='r',␣
 ↪label='5-Fold Cross Validation Error')
plt.plot(features, test_errors ,marker='o', linestyle='-', color='b',␣
 ↪label='Test Error')
plt.xlabel('K')
plt.ylabel('Error')
plt.title('Error Estimation Per value of K')
plt.legend()
plt.grid(True)
plt.show()
```

```
Estimated Error From 5-Fold Cross-Validation with 0 features::
0.17158469945355181
Accuracy with 0 features: 0.8155339805825242
Estimated Error From 5-Fold Cross-Validation with 1 features::
0.171639344262295
Accuracy with 1 features: 0.6990291262135923
Estimated Error From 5-Fold Cross-Validation with 2 features::
0.17491803278688534
Accuracy with 2 features: 0.6990291262135923
Estimated Error From 5-Fold Cross-Validation with 3 features::
0.18513661202185783
Accuracy with 3 features: 0.7184466019417476
Estimated Error From 5-Fold Cross-Validation with 4 features::
0.1750273224043717
Accuracy with 4 features: 0.7281553398058253
Estimated Error From 5-Fold Cross-Validation with 5 features::
0.17158469945355193
Accuracy with 5 features: 0.7572815533980582
Estimated Error From 5-Fold Cross-Validation with 6 features::
0.17486338797814205
Accuracy with 6 features: 0.7864077669902912
Estimated Error From 5-Fold Cross-Validation with 7 features::
0.16169398907103827
Accuracy with 7 features: 0.8155339805825242
Estimated Error From 5-Fold Cross-Validation with 8 features::
0.1654098360655738
Accuracy with 8 features: 0.7961165048543689
Estimated Error From 5-Fold Cross-Validation with 9 features::
0.1618579234972678
Accuracy with 9 features: 0.7864077669902912
Estimated Error From 5-Fold Cross-Validation with 10 features::
0.1680874316939891
Accuracy with 10 features: 0.8058252427184466
Estimated Error From 5-Fold Cross-Validation with 11 features::
```

```
0.18158469945355193
Accuracy with 11 features: 0.8155339805825242
Estimated Error From 5-Fold Cross-Validation with 12 features::
0.17803278688524604
Accuracy with 12 features: 0.8155339805825242
```

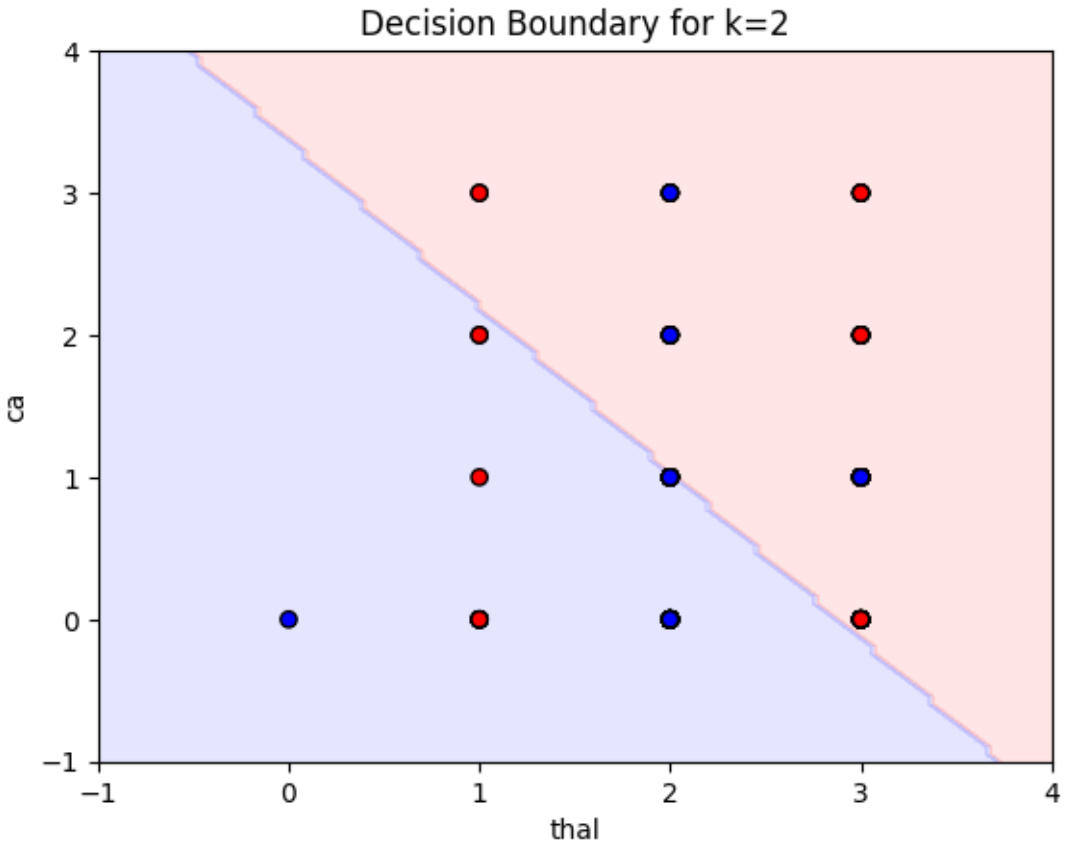


[98]:
```
from matplotlib.colors import ListedColormap

def plot_decision_boundary(X, y, model, selected_features):
    X_selected = X[selected_features]
    x_min, x_max = X_selected.iloc[:, 0].min() - 1, X_selected.iloc[:, 0].max()␣
 ↪+ 1
    y_min, y_max = X_selected.iloc[:, 1].min() - 1, X_selected.iloc[:, 1].max()␣
 ↪+ 1
    xx, yy = np.meshgrid(np.linspace(x_min, x_max, 100), np.linspace(y_min,␣
 ↪y_max, 100))

    Z = model.predict(np.c_[xx.ravel(), yy.ravel()])
    Z = Z.reshape(xx.shape)
```

```
    plt.contourf(xx, yy, Z, alpha=0.3, cmap=ListedColormap(['#FFAAAA',␣
 ↪'#AAAAFF']))
    plt.scatter(X_selected.iloc[:, 0], X_selected.iloc[:, 1], c=y,␣
 ↪edgecolors='k', cmap=ListedColormap(['#FF0000', '#0000FF']))
    plt.xlabel(selected_features[0])
    plt.ylabel(selected_features[1])
    plt.title("Decision Boundary for k=2")
    plt.show()

print(best_columns)
model = LogisticRegression(max_iter=1500, penalty=None)
model.fit(X_train[best_columns], y_train)

plot_decision_boundary(X_train, y_train, model, best_columns)
```

```
['thal', 'ca']
```

```
/Library/Frameworks/Python.framework/Versions/3.12/lib/python3.12/site-
packages/sklearn/utils/validation.py:2739: UserWarning: X does not have valid
feature names, but LogisticRegression was fitted with feature names
  warnings.warn(
```



[ ]: