CSE258 - Homework2

October 28, 2019

1 CSE 258, Fall 2019: Homework 1

Student ID: A53308934 Student Name: Deng Zhang

1.1 Tasks - Diagnostics

[1]:
```
import gzip
import numpy as np
import matplotlib.pyplot as plt
from sklearn import linear_model
import sklearn
from random import shuffle
import random
```

[2]:
```
f = open("./Data/hw2/5year.arff", 'r')
```

[3]:
```
while not '@data' in f.readline():
    pass
```

[4]:
```
dataset = []
for l in f:
    if '?' in l: # Missing entry
        continue
    l = l.split(',')
    values = [1] + [float(x) for x in l]
    values[-1] = values[-1] > 0 # Convert to bool
    dataset.append(values)
```

1.1.1 Question 1:

Code to read the data is available in the stub. Train a logistic regressor (e.g. sklearn.linear model.LogisticRegression) with regularization coefficient C = 1.0. Report the accuracy and Balanced Error Rate (BER) of your classifier.

[5]:
```
# use the last col as y, the reset as x
X = [values[:-1] for values in dataset]
y = [values[-1] for values in dataset]
```

[6]:
```
model = linear_model.LogisticRegression(C=1.0)
```

[7]:

```
model.fit(X, y)
```

```
E:\anaconda\lib\site-packages\sklearn\linear_model\logistic.py:432:
FutureWarning: Default solver will be changed to 'lbfgs' in 0.22. Specify a
solver to silence this warning.
  FutureWarning)
E:\anaconda\lib\site-packages\sklearn\svm\base.py:929: ConvergenceWarning:
Liblinear failed to converge, increase the number of iterations.
  "the number of iterations.", ConvergenceWarning)
```

[7]:

```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='warn', n_jobs=None, penalty='l2',
                   random_state=None, solver='warn', tol=0.0001, verbose=0,
                   warm_start=False)
```

[8]:

```
predictions = model.predict(X)

correct = predictions == y

print("Accuracy = " + str(sum(correct) / len(correct)))
```

```
Accuracy = 0.9663477400197954
```

[9]:

```
TP = sum([(p and l) for (p,l) in zip(predictions, y)])
FP = sum([(p and not l) for (p,l) in zip(predictions, y)])
TN = sum([(not p and not l) for (p,l) in zip(predictions, y)])
FN = sum([(not p and l) for (p,l) in zip(predictions, y)])
```

[10]:

```
TPR = TP / (TP + FN)
TNR = TN / (TN + FP)
```

[11]:

```
BER = 1 - 1/2 * (TPR + TNR)
print("Balanced error rate = " + str(BER))
```

```
Balanced error rate = 0.48580623782459387
```

[12]:

```
# Answer of Question 1:

# Accuracy = 0.9663477400197954
# Balanced error rate = 0.48580623782459387
```

1.1.2 Question 3:

[77]:

```
random.shuffle(dataset)
```

[78]:

```
X = [values[:-1] for values in dataset]
y = [values[-1] for values in dataset]
N = len(X)
X_train = X[:N//2]
X_valid = X[N//2:3*N//4]
X_test = X[3*N//4:]
y_train = y[:N//2]
y_valid = y[N//2:3*N//4]
y_test = y[3*N//4:]

len(X), len(X_train), len(X_test)
```

[78]:

```
(3031, 1515, 758)
```

[79]:

```
model = linear_model.LogisticRegression(class_weight='balanced')
model.fit(X_train, y_train)
```

```
E:\anaconda\lib\site-packages\sklearn\linear_model\logistic.py:432:
FutureWarning: Default solver will be changed to 'lbfgs' in 0.22. Specify a
solver to silence this warning.
  FutureWarning)
E:\anaconda\lib\site-packages\sklearn\svm\base.py:929: ConvergenceWarning:
Liblinear failed to converge, increase the number of iterations.
  "the number of iterations.", ConvergenceWarning)
```

[79]:

```
LogisticRegression(C=1.0, class_weight='balanced', dual=False,
                   fit_intercept=True, intercept_scaling=1, l1_ratio=None,
                   max_iter=100, multi_class='warn', n_jobs=None, penalty='l2',
                   random_state=None, solver='warn', tol=0.0001, verbose=0,
                   warm_start=False)
```

[80]:

```
predictionsTrain = model.predict(X_train)
predictionsValid = model.predict(X_valid)
predictionsTest = model.predict(X_test)

correctPredictionsTrain = predictionsTrain == y_train
correctPredictionsValid = predictionsValid == y_valid
correctPredictionsTest = predictionsTest == y_test
```

[81]:

```
print("Accuracy of Train = " + str(sum(correctPredictionsTrain) /␣
 ↪len(correctPredictionsTrain)))
print("Accuracy of Valid = " + str(sum(correctPredictionsValid) /␣
 ↪len(correctPredictionsValid)))
print("Accuracy of Test = " + str(sum(correctPredictionsTest) /␣
 ↪len(correctPredictionsTest)))
```

```
Accuracy of Train = 0.7795379537953795
Accuracy of Valid = 0.7717678100263852
Accuracy of Test = 0.7928759894459103
```

[82]:

```
def countBer(predictions, Y):
    TP = sum([(p and l) for (p,l) in zip(predictions, Y)])
    FP = sum([(p and not l) for (p,l) in zip(predictions, Y)])
    TN = sum([(not p and not l) for (p,l) in zip(predictions, Y)])
    FN = sum([(not p and l) for (p,l) in zip(predictions, Y)])
    TPR = TP / (TP + FN)
    TNR = TN / (TN + FP)
    precision = TP / (TP + FP)
    recall = TP / (TP + FN)
    #F1 = 2 * (precision*recall) / (precision + recall)
    #print(F1)
    #F10 = 101 * (precision*recall) / (100 * precision + recall)
    #print(F10)
    return 1 - 1/2 * (TPR + TNR)
```

[83]:

```
print("Balanced error rate of Train = " + str(countBer(predictionsTrain,␣
 ↪y_train)))
print("Balanced error rate of Valid = " + str(countBer(predictionsValid,␣
 ↪y_valid)))
print("Balanced error rate of Test = " + str(countBer(predictionsTest, y_test)))
```

```
Balanced error rate of Train = 0.23242263118498818
Balanced error rate of Valid = 0.31118690313778985
Balanced error rate of Test = 0.24801316984559496
```

[84]:

```
# Answer of Question 3:

# Accuracy of Train = 0.7947194719471947
# Balanced error rate of Train = 0.22465886939571145
# Accuracy of Valid = 0.7968337730870713
# Balanced error rate of Valid = 0.16536103542234337
# Accuracy of Test = 0.7770448548812665
# Balanced error rate of Test = 0.27657168701944823
```

1.1.3 Question 4:

[86]:

```
def getBerNAccu(c, X, y):
    model = linear_model.LogisticRegression(C=c, class_weight='balanced')
    model.fit(X, y)
    predictions = model.predict(X)
    correct = predictions == y
    accu = sum(correct) / len(correct)
    TP = sum([(p and l) for (p,l) in zip(predictions, y)])
    FP = sum([(p and not l) for (p,l) in zip(predictions, y)])
    TN = sum([(not p and not l) for (p,l) in zip(predictions, y)])
    FN = sum([(not p and l) for (p,l) in zip(predictions, y)])
    TPR = TP / (TP + FN)
    TNR = TN / (TN + FP)
    ber = 1 - 1/2 * (TPR + TNR)
    return ber, accu
```

[87]:

```
C = [0.0001, 0.001, 0.01, 0.1, 1, 10, 100, 1000, 10000]
berTrain = []
berValid = []
berTest = []
accuTrain = []
accuValid = []
accuTest = []
for c in C:
    ber_train, accu_train = getBerNAccu(c, X_train, y_train)
    berTrain.append(ber_train)
    accuTrain.append(accu_train)
    ber_valid, accu_valid = getBerNAccu(c, X_valid, y_valid)
    berValid.append(ber_valid)
    accuValid.append(accu_valid)
    ber_test, accu_test = getBerNAccu(c, X_test, y_test)
    berTest.append(ber_test)
    accuTest.append(accu_test)
```

```
E:\anaconda\lib\site-packages\sklearn\linear_model\logistic.py:432:
FutureWarning: Default solver will be changed to 'lbfgs' in 0.22. Specify a
solver to silence this warning.
  FutureWarning)
E:\anaconda\lib\site-packages\sklearn\linear_model\logistic.py:432:
FutureWarning: Default solver will be changed to 'lbfgs' in 0.22. Specify a
solver to silence this warning.
  FutureWarning)
E:\anaconda\lib\site-packages\sklearn\linear_model\logistic.py:432:
FutureWarning: Default solver will be changed to 'lbfgs' in 0.22. Specify a
solver to silence this warning.
  FutureWarning)
E:\anaconda\lib\site-packages\sklearn\linear_model\logistic.py:432:
FutureWarning: Default solver will be changed to 'lbfgs' in 0.22. Specify a
solver to silence this warning.
  FutureWarning)
E:\anaconda\lib\site-packages\sklearn\svm\base.py:929: ConvergenceWarning:
Liblinear failed to converge, increase the number of iterations.
  "the number of iterations.", ConvergenceWarning)
E:\anaconda\lib\site-packages\sklearn\linear_model\logistic.py:432:
FutureWarning: Default solver will be changed to 'lbfgs' in 0.22. Specify a
solver to silence this warning.
  FutureWarning)
E:\anaconda\lib\site-packages\sklearn\svm\base.py:929: ConvergenceWarning:
Liblinear failed to converge, increase the number of iterations.
  "the number of iterations.", ConvergenceWarning)
```

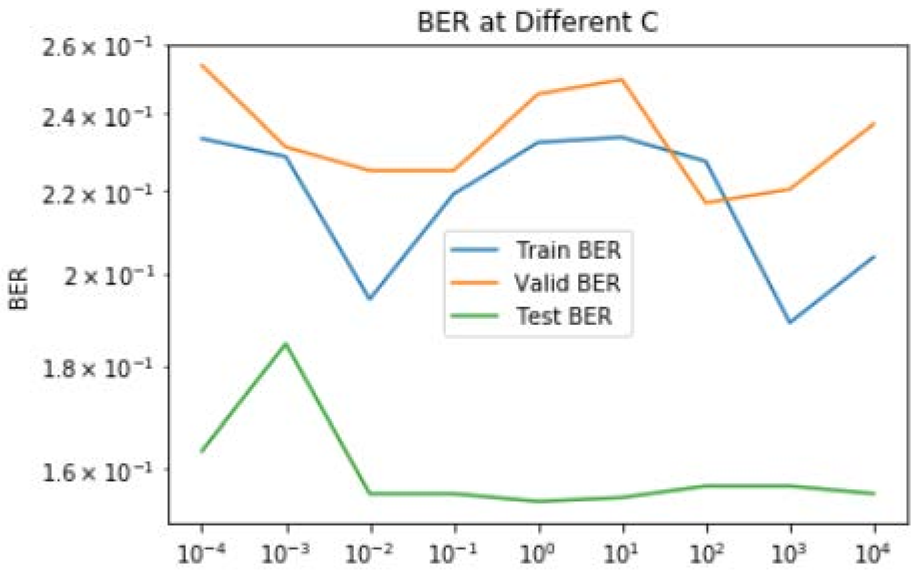
```
E:\anaconda\lib\site-packages\sklearn\linear_model\logistic.py:432:
FutureWarning: Default solver will be changed to 'lbfgs' in 0.22. Specify a
solver to silence this warning.
  FutureWarning)
E:\anaconda\lib\site-packages\sklearn\svm\base.py:929: ConvergenceWarning:
Liblinear failed to converge, increase the number of iterations.
  "the number of iterations.", ConvergenceWarning)
```

[88]:

```python
x = [10**i for i in range(-4, 5, 1)]
plt.loglog(x, berTrain, label = 'Train BER')
plt.loglog(x, berValid, label = 'Valid BER')
plt.loglog(x, berTest, label = 'Test BER')
plt.title("BER at Different C")
plt.xticks(x)
plt.xlabel("C")
plt.ylabel("BER")
plt.legend()
```

[88]:

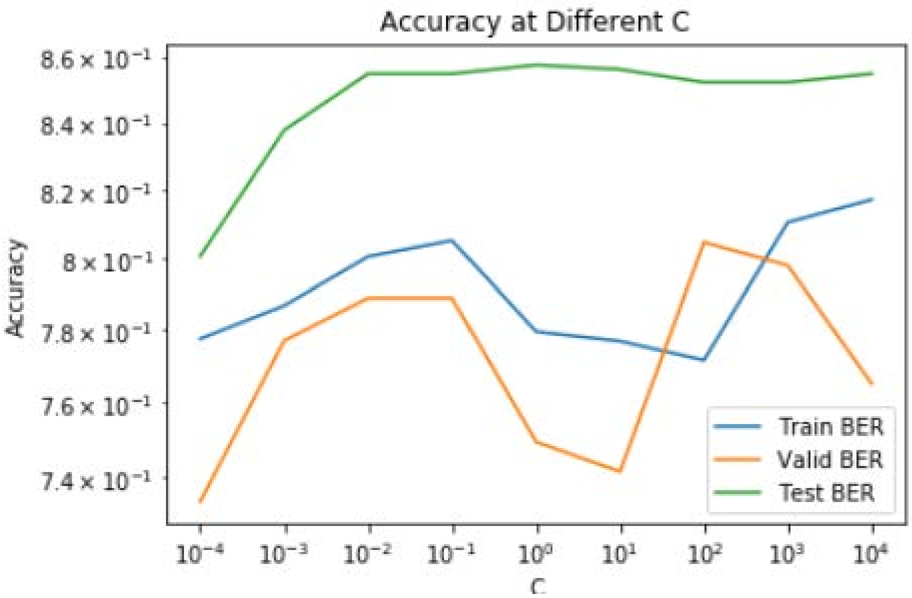
```
<matplotlib.legend.Legend at 0x1a5569c98d0>
```



[89]:

```python
x = [10**i for i in range(-4, 5, 1)]
plt.loglog(x, accuTrain, label = 'Train BER')
plt.loglog(x, accuValid, label = 'Valid BER')
plt.loglog(x, accuTest, label = 'Test BER')
plt.title("Accuracy at Different C")
plt.xticks(x)
plt.xlabel("C")
plt.ylabel("Accuracy")
plt.legend()
```

[89]:
```
<matplotlib.legend.Legend at 0x1a556ad47f0>
```



[90]:
```
# Answer to Question 4:
# BER at Different C is shown in the graph below
# I would choose 0.01 as my classifier, because the accuracy of 0.01
# is high and its BER is low comparably.
```

1.1.4 Question 6:

[91]:
```
weights = [1.0] * len(y_train)
mod = linear_model.LogisticRegression(C=1, solver='lbfgs')
mod.fit(X_train, y_train, sample_weight=weights)
```

```
E:\anaconda\lib\site-packages\sklearn\linear_model\logistic.py:947:
ConvergenceWarning: lbfgs failed to converge. Increase the number of iterations.
  "of iterations.", ConvergenceWarning)
```

[91]:

```
LogisticRegression(C=1, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='warn', n_jobs=None, penalty='l2',
                   random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False)
```

[92]:

```python
def countTF(predictions, Y):
    TP = sum([(p and l) for (p,l) in zip(predictions, Y)])
    FP = sum([(p and not l) for (p,l) in zip(predictions, Y)])
    TN = sum([(not p and not l) for (p,l) in zip(predictions, Y)])
    FN = sum([(not p and l) for (p,l) in zip(predictions, Y)])
    return TP, FP, TN, FN
```

[93]:

```python
predictionsTest = model.predict(X_test)
```

[94]:

```python
TP, FP, TN, FN = countTF(predictionsTest, y_test)
```

[95]:

```python
precision = TP / (TP + FP)
recall = TP / (TP + FN)
precision, recall
```

[95]:

```
(0.10179640718562874, 0.7083333333333334)
```

[96]:

```python
F1 = 2 * (precision*recall) / (precision + recall)
print("Unweighted F1 = " + str(F1))
F10 = 101 * (precision*recall) / (100 * (precision + recall))
print("Unweighted F10 = " + str(F10))
```

```
Unweighted F1 = 0.17801047120418848
Unweighted F10 = 0.08989528795811519
```

[97]:

```python
weightPos = 1 - sum(d == True for d in y_train) / len(y_train)
weightNeg = 1 - weightPos

weights = [weightPos if i == True else weightNeg for i in y_train]
model = linear_model.LogisticRegression(C = 1, solver='lbfgs')
model.fit(X_train, y_train, sample_weight=weights);

predictionsTest = model.predict(X_test)
TP, FP, TN, FN = countTF(predictionsTest, y_test)

precision = TP / (TP + FP)
recall = TP / (TP + FN)

F1 = 2 * (precision*recall) / (precision + recall)
print("Weighted F1 = " + str(F1))
F10 = 101 * (precision*recall) / (100 * (precision + recall))
print("Weighted F10 = " + str(F10))
```

```
Weighted F1 = 0.15625
Weighted F10 = 0.07890625
```

```
E:\anaconda\lib\site-packages\sklearn\linear_model\logistic.py:947:
ConvergenceWarning: lbfgs failed to converge. Increase the number of iterations.
  "of iterations.", ConvergenceWarning)
```

1.2 Tasks - Diagnostics

1.2.1 Question 7

[98]:

```
from sklearn.decomposition import PCA
```

[99]:

```
pca = PCA()
pca.fit(X_train)
print(pca.components_[0])
```

```
[ 1.72047597e-20  1.86056760e-07 -8.30722754e-07  9.59030869e-07
  4.38028415e-06  6.35779403e-04  7.13938829e-07  2.21209550e-07
  4.96949362e-06 -3.36725227e-07  7.71860399e-07  1.83234085e-07
  9.49388385e-07 -3.81787900e-06  2.20951177e-07 -2.98474327e-03
  9.61385635e-07  5.56488421e-06  1.25987227e-07  2.32714260e-07
  2.38926550e-05  9.14450714e-08  1.60469057e-07  1.97574372e-07
  7.00817918e-07  9.28372692e-07  8.11792660e-07 -2.43908703e-05
  3.01158151e-05  2.14868217e-06 -1.19847051e-06  2.12564711e-07
 -4.36950355e-04  4.05797604e-06 -1.37304269e-06  1.62908714e-07
 -5.35818142e-07 -4.15681416e-03  6.80511463e-07  1.77568890e-07
  2.03089404e-06  1.14186465e-06  1.12139990e-07  4.34390932e-05
  1.95464930e-05 -2.50390847e-06  3.30813105e-06 -2.20967729e-04
  1.90814663e-07  1.90788127e-07  3.78915519e-06 -7.09356562e-07
 -1.19191870e-06 -2.90611196e-06  3.00529136e-05  9.99986545e-01
  1.52955233e-07  1.14923540e-06 -1.98702936e-07 -1.25922567e-06
 -1.38350257e-04 -3.08172804e-06 -2.33019801e-04  5.03041552e-06
 -5.23478902e-07]
```

1.2.2 Question 8

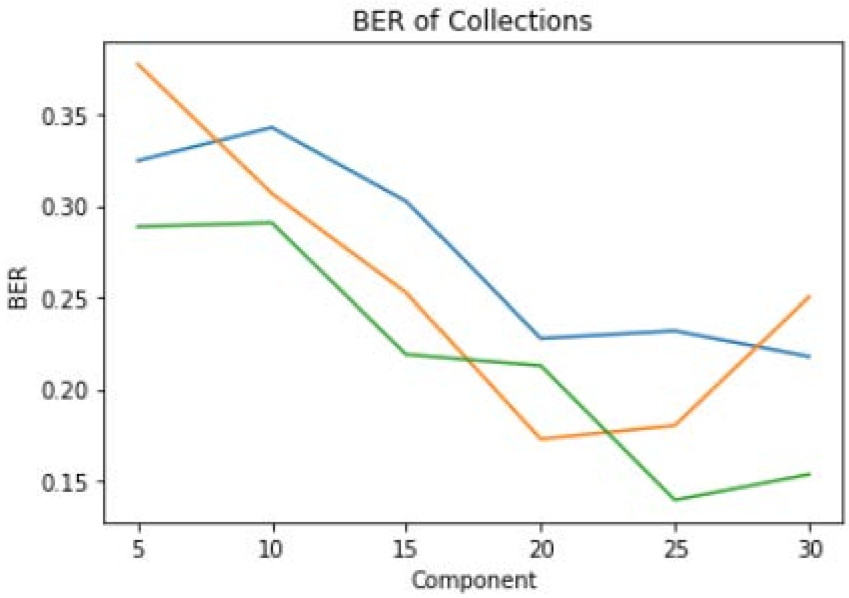
[100]:

```
def countComponent(X, y):
    berList = []
    for component in range(5, 31, 5):
        pca = PCA(n_components=component)
        pca.fit(X)
        Xpca = np.matmul(X, pca.components_.T)
        model = linear_model.LogisticRegression(C=1.0, class_weight='balanced')
        model.fit(Xpca, y)
        predictions = model.predict(Xpca)
        berList.append(countBer(predictions, y))
    return berList
```

[101]:

```
ber_train = countComponent(X_train, y_train)
ber_valid = countComponent(X_valid, y_valid)
ber_test = countComponent(X_test, y_test)
```

```
plt.plot(range(5, 31, 5), ber_train, label='Train BER')
plt.plot(range(5, 31, 5), ber_valid, label='Valid BER')
plt.plot(range(5, 31, 5), ber_test, label='Test BER')
plt.title("BER of Collections")
plt.xlabel("Component")
plt.ylabel("BER")
plt.show()
```


BER of Collections
BER
Component