PA5

March 25, 2021

[1]:
```
import numpy as np
```

[2]:
```
def extract_data(file):
    data = []
    with open(file) as f:
        for lines in f.readlines():
            data.append(np.array([int(i) for i in lines.strip().split(" ")]))

    return data
```

[3]:
```
def extract_dict(file):
    data = []
    with open(file) as f:
        for lines in f.readlines():
            data.append(lines.strip())
    return data
```

[4]:
```
def classifier(value):
    if (value == 1):
        return 1
    else:
        return -1
```

[5]:
```
def negative_classifier(value):
    if (value == 0):
        return 1
    else:
        return -1
```

[6]:
```
def calculate_error(feature, dataset):
    error_output = 0

    for i in range(len(dataset)):
        if (classifier(dataset[i][feature]) != dataset[i][-1]):
            indicator = 1
        else:
            indicator = 0
        error_output += weights[i]*indicator

    return error_output
```

[7]:
```
def calculate_alpha(error):
    return (1/2)*np.log((1-error)/error)
```

[8]:
```
def calculate_label(features, boost_classifiers):
    summation = 0
    for boost in boost_classifiers:
        word = boost[0]
        alpha = boost[2]

        if boost[1] == 1:
            if features[word] == 1:
                h = 1
            else:
                h = -1
        else:
            if features[word] == 0:
                h = 1
            else:
                h = -1

        summation += alpha * h

    return np.sign(summation)
```

[9]:
```
# Extracting the data
train = extract_data("pa5train.txt")
test = extract_data("pa5test.txt")
```

[10]:
```
# Extracting the dictionary
vocab = extract_dict("pa5dictionary.txt")
```

[11]:
```
# Spam Classification w/ 4 rounds of boosting
boost_rounds = 4
boost_classifier = []
weights = np.zeros(len(train)) + 1/len(train)

for t in range(boost_rounds):
    best_error = 100
    best_feature = -1
    label = 0
    for feature in range(len(vocab)):
        error = calculate_error(feature, train)

        if (error < best_error):
            best_feature = feature
            best_error = error
            label = 1
        elif (1 - error < best_error):
            best_feature = feature
            best_error = 1 - error
            label = -1

    alpha = (1/2)*np.log((1-best_error)/best_error)

    for i in range(len(train)):
        y = train[i][-1]
        h = 0

        if (label == 1):
            h = classifier(train[i][best_feature])
        else:
            h = negative_classifier(train[i][best_feature])

        weights[i] = weights[i]*np.exp(-alpha*y*h)

    z = np.sum(weights)
    weights = weights/z #

    boost_classifier.append([best_feature, label, alpha])


incorrect = 0
total = len(train)
for vector in train:
    h = calculate_label(vector[:-1], boost_classifier)

    if (h != vector[-1]):
        incorrect += 1

training_error = incorrect/total
print("Training Error: " + str(training_error))

incorrect = 0
total = len(test)
for vector in test:
    h = calculate_label(vector[:-1], boost_classifier)

    if (h != vector[-1]):
        incorrect += 1

test_error = incorrect/total
print("Test Error: " + str(test_error))
```

```
Training Error: 0.05111111111111114
Test Error: 0.03875968992248062
```

[12]:
```
# Question 1:
boost_rounds = [3,4,7,10,15,20]

for rounds in boost_rounds:
    boost_classifier = []
    weights = np.zeros(len(train)) + 1/len(train)

    for t in range(rounds):
        best_error = 100
        best_feature = -1
        label = 0
        for feature in range(len(vocab)):
            error = calculate_error(feature, train)

            if (error < best_error):
                best_feature = feature
                best_error = error
                label = 1
            elif (1 - error < best_error):
                best_feature = feature
                best_error = 1 - error
                label = -1

        alpha = (1/2)*np.log((1-best_error)/best_error)

        for i in range(len(train)):
            y = train[i][-1]
            h = 0

            if (label == 1):
                h = classifier(train[i][best_feature])
            else:
                h = negative_classifier(train[i][best_feature])

            weights[i] = weights[i]*np.exp(-alpha*y*h)

        z = np.sum(weights)
        weights = weights/z #
        boost_classifier.append([best_feature, label, alpha])


    incorrect = 0
    total = len(train)
    for vector in train:
        h = calculate_label(vector[:-1], boost_classifier)

        if (h != vector[-1]):
            incorrect += 1

    training_error = incorrect/total
    print("Number of boost rounds: " + str(rounds))
    print("Training Error: " + str(training_error))

    incorrect = 0
    total = len(test)
    for vector in test:
        h = calculate_label(vector[:-1], boost_classifier)

        if (h != vector[-1]):
            incorrect += 1

    test_error = incorrect/total
    print("Test Error: " + str(test_error))
    print("")
```

```
Number of boost rounds: 3
Training Error: 0.06444444444444444
Test Error: 0.03875968992248062

Number of boost rounds: 4
Training Error: 0.051111111111111114
Test Error: 0.03875968992248062

Number of boost rounds: 7
Training Error: 0.028888888888888888
Test Error: 0.031007751937984496

Number of boost rounds: 10
Training Error: 0.015555555555555555
Test Error: 0.03875968992248062

Number of boost rounds: 15
Training Error: 0.0
Test Error: 0.023255813953488372
```

```
Number of boost rounds: 20
Training Error: 0.0
Test Error: 0.023255813953488372
```

```
[13]: # Question 2:
boost_rounds = 10
boost_classifier = []
weights = np.zeros(len(train)) + 1/len(train)

for t in range(boost_rounds):
    best_error = 100
    best_feature = -1
    label = 0
    for feature in range(len(vocab)):
        error = calculate_error(feature, train)

        if (error < best_error):
            best_feature = feature
            best_error = error
            label = 1
        elif (1 - error < best_error):
            best_feature = feature
            best_error = 1 - error
            label = -1

    alpha = (1/2)*np.log((1-best_error)/best_error)

    for i in range(len(train)):
        y = train[i][-1]
        h = 0

        if (label == 1):
            h = classifier(train[i][best_feature])
        else:
            h = negative_classifier(train[i][best_feature])

        weights[i] = weights[i]*np.exp(-alpha*y*h)

    z = np.sum(weights)
    weights = weights/z #

    boost_classifier.append([best_feature, label, alpha])


incorrect = 0
total = len(train)
```

```
for vector in train:
    h = calculate_label(vector[:-1], boost_classifier)

    if (h != vector[-1]):
        incorrect += 1

training_error = incorrect/total
print("Training Error: " + str(training_error))

incorrect = 0
total = len(test)
for vector in test:
    h = calculate_label(vector[:-1], boost_classifier)

    if (h != vector[-1]):
        incorrect += 1

test_error = incorrect/total
print("Test Error: " + str(test_error))
print("")
```

```
Training Error: 0.015555555555555555
Test Error: 0.03875968992248062
```

[14]:
```
# Question 2:
[vocab[j] for j in [i[0] for i in boost_classifier]]
```

[14]:
```
['remove',
 'language',
 'free',
 'university',
 'money',
 'linguistic',
 'click',
 'fax',
 'want',
 'de']
```

[ ]:
```

```