CS464 Machine Learning
Homework 1
Batıhan Akça / 21502824

1 The Chess Game

Define: Y = {W_{Steve}, Draw, W_{Willam}} for winning state of a game.

W = {Timid, Bold} for playing style of William.

P ($W_{William}$ | William = Timid) = 0 P ($W_{William}$ | William = Bold) = 0.6

P (W_{Steve} | William = Timid) = 0.35 P (W_{Steve} | William = Bold) = 0.4

P (Draw | William = Timid) = 0.65 P (Draw | William = Bold) = 0

(1.1) Gates's winning possibilities ;

3 Win - > W, W, W $(0.6)^3 = 0.216$

2 Win - > W, W, L $(0.6)^2 * (0.4)\ (3! / 2!*1!) = 0.432$
- > W, L, W
- > L, W, W

Answer: 0.432 + 0.216 = 0.648

(1.2) Gates's winning possibilities ;

If the first three games are played in timid style then there is no chance for Gates for winning any of the first three game. Only possibility for him to win the match is to draw the first three game and win the fourth game with bold strategy.

Draw, Draw, Draw, Win - > $(0.65)^3 * (0.6) = 0.164775$

(1.3)

$P(William = Bold \mid Y = Wwilliam) =$

$$\frac{P(William = Bold) * P(Wsteve \mid William = Bold)}{P(William = Bold) * P(Wsteve \mid William = Bold) + P(William = Timid) * P(Wsteve \mid William = Timid)}$$

$$\frac{(0.5)*(0.4)}{(0.5)*(0.4)+(0.5)*(0.35)} = \left(\frac{0.4}{0.75}\right) = 8/15$$

(1.4)

$P(Y = Wsteve) = ?$

$= P(\rho = True) * P(Wsteve|\rho = True) + P(\rho = False) * P(Wsteve|\rho = False)$
$= (0.75) * (0.85) + (0.25) * P(Wsteve \mid \rho = False)$

$P(Wsteve \mid \rho = False) = ?$

$= P(William = Bold) * P(Wsteve|William = Bold) + P(William = Timid) * P(Wsteve|William = Timid)$
$= (0.5) * (0.4) + (0.5) * (0.35)$
$= 0.375$

$P(Y = Wsteve) = (0.75) * (0.85) + (0.25) * (0.375) = (0.73125)$

2 Medical Diagnosis

(2.1) $P(S = disease) = 0.005$
$P(S = healthy) = 0.995$
$P(T = positive \mid S = disease) = 0.970$
$P(T = negative \mid S = disease) = 0.030$
$P(T = positive \mid S = healthy) = 0.020$
$P(T = negative \mid S = healthy) = 0.980$

(2.2) $P(S \mid T = positive),\ S = \{disease, healthy\}$

Let's compare $(*)P(S = disease \mid T = positive)\ and\ (**)P(S = heatlthy \mid T = positive)$ probabilities by Naïve Bayes to decide S.

$(*) \approx P(T = positive \mid S = disease) * P(S = disease)$
$(*) \approx (0.970) * (0.005) = 0.00485$

$(**) \approx P(T = positive \mid S = healthy) * P(S = healthy)$
$(**) \approx (0.020) * (0.995) = 0.0199$

$(**) >> (*)$

Therefore, the patient should be diagnosed as healthy. The reason the test fails is that it fails 2%, 3% of the time where the disease occurs only 5‰ of the time. It means that the test is not successful.

3 MLE and MAP

Let's call lambda as δ for today because of technical issues and $\gamma(\alpha)$ is the gamma function.

(3.1) $\hat{\delta}_{MLE} = argmax_{\delta}\, P(D \mid \delta)$

$$P(D \mid \delta) = \frac{\delta^{x} * e^{-\delta}}{x!}$$

We should find $\frac{d}{d\delta} \ln P(D \mid \delta)$ and equalize it to zero to find $\hat{\delta}_{MLE}$.

$$0 = \frac{d}{d\delta} \ln \prod_{i=1}^{n} \frac{\delta^{x_i} * e^{-\delta}}{x_i!}$$
$$0 = \frac{d}{d\delta} \ln \frac{\delta^{\sum x_i} * e^{-n\delta}}{\sum x_i!}$$
$$0 = \frac{d}{d\delta} (\sum x_i \ln \delta - n\delta - \ln \sum x_i!)$$
$$0 = \frac{\sum x_i}{\delta} - n - 0$$
$$\delta = \frac{\sum x_i}{n} => \hat{\delta}_{MLE} = \frac{\sum x_i}{n}$$

(3.2) $\hat{\delta}_{MAP} = argmax_{\delta}\, P(\delta \mid D) = argmax_{\delta}\, \frac{P(D \mid \delta) * P(\delta)}{P(D)}$

We can neglect $P(D)$ because we will lost it when we take the derivative of the logarithm. It is not depend on δ.

$$\hat{\delta}_{MAP} = argmax_{\delta}\, (P(D \mid \delta) * P(\delta))$$

$$0 = \frac{d}{d\delta} \ln(P(D \mid \delta) * P(\delta))$$
$$0 = \frac{d}{d\delta} \ln((\prod_{i=1}^{n} \frac{\delta^{x_i} * e^{-\delta}}{x_i!}) * \frac{\beta^{\alpha} * \delta^{\alpha-1} * e^{-\beta\delta}}{\gamma(\alpha)})$$
$$0 = \frac{d}{d\delta} (\sum x_i \ln \delta - \delta n - \ln \prod x_i! + \alpha \ln \beta + (\alpha - 1) \ln \delta - \beta\delta - \ln \gamma(\alpha))$$
$$0 = \frac{\sum x_i}{\delta} - n - 0 + 0 + \frac{\alpha - 1}{\delta} - \beta - 0$$
$$\delta = \frac{\sum x_i + \alpha - 1}{\beta + n} => \hat{\delta}_{MAP} = \frac{\sum x_i + \alpha - 1}{\beta + n}$$

(3.3) $\hat{\delta}_{MLE} = \frac{\sum x_i}{n}$

$\hat{\delta}_{MAP} = argmax_{\delta}\, (P(D \mid \delta) * P(\delta))$

$0 = \frac{d}{d\delta} \ln(P(D \mid \delta) * P(\delta))$

$$0 = \frac{d}{d\delta} \ln((\prod_{i=1}^{n} \frac{\delta^{x_i} * e^{-\delta}}{x_i!}) * \frac{1}{b-a})$$

From the new prior distribution $\delta \sim U(a, b)$ we will get 0 by taking logarithm and then the derivative of it.

$$0 = \frac{\sum x_i}{\delta} - n$$

$$\delta = \frac{\sum x_i}{n} => \hat{\delta}_{MAP} = \frac{\sum x_i}{n}$$

$$\hat{\delta}_{MLE} = \hat{\delta}_{MAP}$$

MLE estimate and MAP estimate are the same for any "a" and "b" because they are not depending on 'a' or 'b'.

4 Sentiment Analysis on Tweets

(4.1) Like it is used in the solution of 2.2, denominator can be ignored because it will be the same for all tweets. Since we are calculating denominator with all k and j's we do not need to calculate it for classification, if we would do this that would mean that we are dividing every numerator to the same denominator and this won't make any difference in terms of comparing the numerators. Therefore, since we do not need to find the exact value but we want to find the highest probability, we do not need to calculate denominator because it won't change anything.

(4.2) Percentage of negative tweets is: **7091/11712 > > 60%**

I don't think that given training data set is imbalanced. Even if we think about the content of our data set, it is like the reviews of people from twitter. In my opinion, majority of negative comments is natural in this case because most of the people tweets about a company when there is a problem about it and people rarely tweets when they are satisfied by a company or a product. If we were able to collect every tweets about our topic, there would be a majority of negative comments too.

Yes, definitely imbalanced training set can affect the model. If in our dataset if negative tweets had %95 percentage, this would cause our model to be weaker when detecting positive or neutral tweets because of the lack of enough examples. We can use under sampling method to prevent this problem in those kind of cases. Basically, we delete instances from the majority class in order to even-up the classes.

(4.3) We will need estimate 3 parameters for $P(Y) \mid Y = \{neutral, positive, negative\}$.
And 5722 x 3 = 17166 parameters for words' occurrence in 3 types of tweets
Total = 17169

(4.4) My model end up with 903 true predictions out of 2928. 903 includes 587 negative, 162 positive and 154 neutral tweets. Accuracy is 903/2928 -> 30.84%.

MLE tried to find predicted proportions close to the labels distribution on training data set and it made it in the true percentages (587/903 => %65, there was %60 negatives in total). However; in MLE we gave the weights to the words as how frequent they occur, this may cause some words to dominate the others and we may almost ignored some specific but rarely occurred words that can have a strong direction to one of the labels also we lost information from very rarely occurring words.

(4.5) In this model with a little touch the accuracy increased more than double. 2050 true predictions and accuracy is %70. α = 1 reduced the lost information from the words that are not occur frequently and we

also made a change in the summation. We were assuming 0 in 0*log0 cases in the previous part but this time we gave every words at least 1 occurrence, this reduced the domination of frequently occurring words and clearly increased the accuracy.

(4.6) This model end up with 1025 true predictions out of 2928. 1025 includes 638 negative, 172 positive and 215 neutral tweets. Accuracy is 1025/2928 -> 35%.

Bernoulli Naïve Bayes did a better job than Multinomial Model but it is 35% still. I think this happens because there are some effective/dominant words in the vocabulary but they are common for both three classes. Bernoulli is better than Multinomial in this case because it decreased the effect of the dominant words with rescaling them every occurring word to 1. As it was seen in the MAP model, how much we can include the rare words to the model we increase our accuracy. Bernoulli could not include them but decreased the dominant ones effect so it became more successful than Multinomial Model.

(4.7) Most Common 20 Words in Positive Tweets

@southwestair
@jetblue
@united
flight
@usairways
great
@virginamerica
service
love
best
guys
customer
time
awesome
help
airline
amazing
today
fly
flying

Most Common 20 Words in Negative Tweets

@united
flight
@usairways
@southwestair
@jetblue
cancelled
service
hours
hold
time
customer
help

delayed
plane
hour
flights
bag
gate
late
flightled

Most Common 20 Words in Neutral Tweets

_____

@jetblue
@united
@southwestair
flight
@usairways
@virginamerica
flights
help
fleek
fleet's
dm
time
tomorrow
flying
cancelled
fly
change
today
travel
check

I see that my guess has very rightful parts because top five of each class contains the same words **@united, flight, @usairways, @southwestair, @jetblue** that have contain nothing characteristic that can make a tweet positive, negative or neutral about a company. Problem with the dataset was it contains non-characteristic common words for both of the labels so when we tried to predict labels with either Bernoulli or Multinomial models under the dominance of those words, model could not predict with a higher accuracy.

CODES

(.py files are also included in the .zip file.)

4.4 Naïve Bayes

```
import csv
import math

prob_positive = 2004/11712
prob_negative = 7091/11712
prob_neutral = 2617/11712

train= list()
labels= list()
test= list()
testlabels= list()

train_pos = list()
train_neg = list()
train_neut = list()

Y = [['positive'], ['negative'], ['neutral']]

with open('question-4-train-features.csv', newline='') as csvfile:
    lines = csv.reader(csvfile)
    i=0
    for row in lines:
        train.append(row)

with open('question-4-train-labels.csv', newline='') as csvfile:
    lines = csv.reader(csvfile)
    i=0
    for row in lines:
        labels.append(row)

i = 0
for row in train:
    if labels[i] == Y[0]:
        train_pos.append(row)
    elif labels[i] == Y[1]:
        train_neg.append(row)
    elif labels[i] == Y[2]:
        train_neut.append(row)
    i = i + 1

total_word_occurance_pos = 0
total_word_occurance_neg = 0
total_word_occurance_neut = 0

for row in train_pos:
```

```
    for i in range(len(row)):
        total_word_occurance_pos = total_word_occurance_pos + int(row[i])

for row in train_neg:
    for i in range(len(row)):
        total_word_occurance_neg = total_word_occurance_neg + int(row[i])

for row in train_neut:
    for i in range(len(row)):
        total_word_occurance_neut = total_word_occurance_neut + int(row[i])

def word_occurance(i, label):
    occurance = 0

    if label == 'pos':
        for row in train_pos:
            occurance = occurance + int(row[i])
    elif label == 'neg':
        for row in train_neg:
            occurance = occurance + int(row[i])
    elif label == 'neut':
        for row in train_neut:
            occurance = occurance + int(row[i])
    return occurance

occurance_pos = list()
occurance_neg = list()
occurance_neut = list()

for i in range(5722):
    occurance_pos.append(word_occurance(i, "pos"))
    occurance_neg.append(word_occurance(i, "neg"))
    occurance_neut.append(word_occurance(i, "neut"))

def prediction(tweet):

    positive = math.log(prob_positive)
    negative = math.log(prob_negative)
    neutral = math.log(prob_neutral)

    for i in range(5722):
        if occurance_pos[i] > 0:
            positive = positive + (int(tweet[i]) * math.log(occurance_pos[i]/total_word_occurance_pos))
    for i in range(5722):
        if occurance_neg[i] > 0:
            negative = negative + (int(tweet[i]) * math.log(occurance_neg[i]/total_word_occurance_neg))
    for i in range(5722):
        if occurance_neut[i] > 0:
            neutral = neutral + (int(tweet[i]) * math.log(occurance_neut[i]/total_word_occurance_neut))

    if positive > negative:
        if positive > neutral:
            result = "positive"
    if negative > positive:
        if negative > neutral:
            result = "negative"
```

```
    if neutral >= positive:
        if neutral >= negative:
            result="neutral"

    return result

with open('question-4-test-features.csv', newline='') as csvfile:
    lines = csv.reader(csvfile)
    i=0
    for row in lines:
        test.append(row)

with open('question-4-test-labels.csv', newline='') as csvfile:
    lines = csv.reader(csvfile)
    i=0
    for row in lines:
        testlabels.append(row)

predicted = list()
true_pos = 0
true_neg = 0
true_neut = 0
false = 0

for i in range(len(test)):
    p = prediction(test[i])

    if p == testlabels[i][0]:
        if p == "positive":
            true_pos = true_pos + 1
        elif p == "negative":
            true_neg = true_neg + 1
        else:
            true_neut = true_neut + 1
    else:
        false = false + 1
    true = true_pos + true_neg + true_neut
    predicted.append(p)

print("True predictions: " + str(true))
print("Wrong predictions: " + str(false))
print("Accurracy: " + str(true*100/(true+false)) + "%")
```

4.5 MAP

```
import csv
import math

prob_positive = 2004/11712
prob_negative = 7091/11712
prob_neutral = 2617/11712
alpha = 1
V = 5722
```

```
train= list()
labels= list()
test= list()
testlabels= list()

train_pos = list()
train_neg = list()
train_neut = list()

Y = [['positive'], ['negative'], ['neutral']]

with open('question-4-train-features.csv', newline='') as csvfile:
    lines = csv.reader(csvfile)
    i=0
    for row in lines:
        train.append(row)

with open('question-4-train-labels.csv', newline='') as csvfile:
    lines = csv.reader(csvfile)
    i=0
    for row in lines:
        labels.append(row)

i = 0
for row in train:
    if labels[i] == Y[0]:
        train_pos.append(row)
    elif labels[i] == Y[1]:
        train_neg.append(row)
    elif labels[i] == Y[2]:
        train_neut.append(row)
    i = i + 1

total_word_occurance_pos = 0
total_word_occurance_neg = 0
total_word_occurance_neut = 0

for row in train_pos:
    for i in range(len(row)):
        total_word_occurance_pos = total_word_occurance_pos + int(row[i])

for row in train_neg:
    for i in range(len(row)):
        total_word_occurance_neg = total_word_occurance_neg + int(row[i])

for row in train_neut:
    for i in range(len(row)):
        total_word_occurance_neut = total_word_occurance_neut + int(row[i])

def word_occurance(i, label):
    occurance = 0

    if label == 'pos':
        for row in train_pos:
            occurance = occurance + int(row[i])
    elif label == 'neg':
```

```
        for row in train_neg:
            occurance = occurance + int(row[i])
    elif label == 'neut':
        for row in train_neut:
            occurance = occurance + int(row[i])
    occurance = occurance + 1

    return occurance

occurance_pos = list()
occurance_neg = list()
occurance_neut = list()

for i in range(5722):
    occurance_pos.append(word_occurance(i, "pos"))
    occurance_neg.append(word_occurance(i, "neg"))
    occurance_neut.append(word_occurance(i, "neut"))

def prediction(tweet):

    positive = math.log(prob_positive)
    negative = math.log(prob_negative)
    neutral = math.log(prob_neutral)

    for i in range(5722):
        if occurance_pos[i] > 0:
            positive = positive + (int(tweet[i]) * math.log(occurance_pos[i]/total_word_occurance_pos))
    for i in range(5722):
        if occurance_neg[i] > 0:
            negative = negative + (int(tweet[i]) * math.log(occurance_neg[i]/total_word_occurance_neg))
    for i in range(5722):
        if occurance_neut[i] > 0:
            neutral = neutral + (int(tweet[i]) * math.log(occurance_neut[i]/total_word_occurance_neut))

    if positive > negative:
        if positive > neutral:
            result = "positive"
    if negative > positive:
        if negative > neutral:
            result = "negative"
    if neutral >= positive:
        if neutral >= negative:
            result="neutral"

    return result

with open('question-4-test-features.csv', newline='') as csvfile:
    lines = csv.reader(csvfile)
    i=0
    for row in lines:
        test.append(row)

with open('question-4-test-labels.csv', newline='') as csvfile:
    lines = csv.reader(csvfile)
    i=0
    for row in lines:
```

```
        testlabels.append(row)

predicted = list()
true_pos = 0
true_neg = 0
true_neut = 0
false = 0

for i in range(len(test)):
    p = prediction(test[i])

    if p == testlabels[i][0]:
        if p == "positive":
            true_pos = true_pos + 1
        elif p == "negative":
            true_neg = true_neg + 1
        else:
            true_neut = true_neut + 1
    else:
        false = false + 1
    true = true_pos + true_neg + true_neut
    predicted.append(p)

print("True predictions: " + str(true))
print("Wrong predictions: " + str(false))
print("Accurracy: " + str(true*100/(true+false)) + "%")
```

4.6 Bernoulli

```
import csv
import math

prob_positive = 2004/11712
prob_negative = 7091/11712
prob_neutral = 2617/11712

train= list()
labels= list()
test= list()
testlabels= list()

train_pos = list()
train_neg = list()
train_neut = list()

Y = [['positive'], ['negative'], ['neutral']]

with open('question-4-train-features.csv', newline='') as csvfile:
    lines = csv.reader(csvfile)
    i=0
    for row in lines:
        train.append(row)

with open('question-4-train-labels.csv', newline='') as csvfile:
```

```
    lines = csv.reader(csvfile)
    i=0
    for row in lines:
        labels.append(row)

i = 0
for row in train:
    if labels[i] == Y[0]:
        train_pos.append(row)
    elif labels[i] == Y[1]:
        train_neg.append(row)
    elif labels[i] == Y[2]:
        train_neut.append(row)
    i = i + 1


def word_occurance(i, label):
    occurance = 0

    if label == 'pos':
        for row in train_pos:
            if int(row[i]) > 0:
                occurance = occurance + 1
    elif label == 'neg':
        for row in train_neg:
            if int(row[i]) > 0:
                occurance = occurance + 1
    elif label == 'neut':
        for row in train_neut:
            if int(row[i]) > 0:
                occurance = occurance + 1

    return occurance

occurance_pos = list()
occurance_neg = list()
occurance_neut = list()

for i in range(5722):
    occurance_pos.append(word_occurance(i, "pos"))
    occurance_neg.append(word_occurance(i, "neg"))
    occurance_neut.append(word_occurance(i, "neut"))

def prediction(tweet):

    positive = math.log(prob_positive)
    negative = math.log(prob_negative)
    neutral = math.log(prob_neutral)

    for i in range(5722):
        if occurance_pos[i] > 0:
            if int(tweet[i]) > 0:
                positive = positive + math.log(occurance_pos[i]/len(train_pos))
            else:
                positive = positive + math.log(1-(occurance_pos[i]/len(train_pos)))
```

```
    for i in range(5722):
        if occurance_neg[i] > 0:
            if int(tweet[i]) > 0:
                negative = negative + math.log(occurance_neg[i]/len(train_neg))
            else:
                negative = negative + math.log(1-(occurance_neg[i]/len(train_neg)))

    for i in range(5722):
        if occurance_neut[i] > 0:
            if int(tweet[i]) > 0:
                neutral = neutral + math.log(occurance_neut[i]/len(train_neut))
            else:
                neutral = neutral + math.log(1-(occurance_neut[i]/len(train_neut)))

    if positive > negative:
        if positive > neutral:
            result = "positive"
    if negative > positive:
        if negative > neutral:
            result = "negative"
    if neutral >= positive:
        if neutral >= negative:
            result="neutral"

    return result

with open('question-4-test-features.csv', newline='') as csvfile:
    lines = csv.reader(csvfile)
    i=0
    for row in lines:
        test.append(row)

with open('question-4-test-labels.csv', newline='') as csvfile:
    lines = csv.reader(csvfile)
    i=0
    for row in lines:
        testlabels.append(row)

predicted = list()
true_pos = 0
true_neg = 0
true_neut = 0
false = 0

for i in range(len(test)):
    p = prediction(test[i])

    if p == testlabels[i][0]:
        if p == "positive":
            true_pos = true_pos + 1
        elif p == "negative":
            true_neg = true_neg + 1
        else:
            true_neut = true_neut + 1
    else:
        false = false + 1
```

```
    true = true_pos + true_neg + true_neut
    predicted.append(p)

print("True predictions: " + str(true))
print("Wrong predictions: " + str(false))
print("Accurracy: " + str(true*100/(true+false)) + "%")
```

4.7 Vocabulary

```
import csv

train= list()
labels= list()

train_pos = list()
train_neg = list()
train_neut = list()

Y = [['positive'], ['negative'], ['neutral']]

fileHandle = open('question-4-vocab.txt', encoding="utf8")
vocab = list()
for line in fileHandle:
    vocab.append(line)
fileHandle.close()

words = list()

for line in vocab:
    pos = line.index("\t")
    num = line[:pos]
    words.append(num)


with open('question-4-train-features.csv', newline='') as csvfile:
    lines = csv.reader(csvfile)
    i=0
    for row in lines:
        train.append(row)

with open('question-4-train-labels.csv', newline='') as csvfile:
    lines = csv.reader(csvfile)
    i=0
    for row in lines:
        labels.append(row)

i = 0
for row in train:
    if labels[i] == Y[0]:
        train_pos.append(row)
    elif labels[i] == Y[1]:
        train_neg.append(row)
    elif labels[i] == Y[2]:
        train_neut.append(row)
```

```
    i = i + 1

def word_occurance(i, label):
    occurance = 0

    if label == 'pos':
        for row in train_pos:
            occurance = occurance + int(row[i])
    elif label == 'neg':
        for row in train_neg:
            occurance = occurance + int(row[i])
    elif label == 'neut':
        for row in train_neut:
            occurance = occurance + int(row[i])
    return occurance

occurance_pos = list()
occurance_neg = list()
occurance_neut = list()

for i in range(5722):
    occurance_pos.append(word_occurance(i, "pos"))
    occurance_neg.append(word_occurance(i, "neg"))
    occurance_neut.append(word_occurance(i, "neut"))

positive_20 = list()
negative_20 = list()
neutral_20 = list()

positive_20_words = list()
negative_20_words = list()
neutral_20_words = list()

print("Most Common 20 Words in Positive Tweets")
print("_____")
for i in range(20):
    index = occurance_pos.index(max(occurance_pos))
    occurance_pos[index] = 0
    positive_20.append(index)
    positive_20_words.append(words[index])
    print(words[index])

print()
print("Most Common 20 Words in Negative Tweets")
print("_____")
for i in range(20):
    index = occurance_neg.index(max(occurance_neg))
    occurance_neg[index] = 0
    negative_20.append(index)
    negative_20_words.append(words[index])
    print(words[index])

print()
print("Most Common 20 Words in Neutral Tweets")
print("_____")
for i in range(20):
```

```
    index = occurance_neut.index(max(occurance_neut))
    occurance_neut[index] = 0
    neutral_20.append(index)
    neutral_20_words.append(words[index])
print(words[index])
```