Machine Learning Fall 2016

HW5 Writeup

Luyao Zhou

Lz6nf

**1.3**

**thetaPos**

love: 0.001003368842756988

wonderful: 0.0002166985340981517

best: 0.0010616407174724575

great: 0.0009796958936538287

superb: 9.469179641263771e-05

still: 0.000728398433943367

beautiful: 0.00026404443230447055

bad: 0.00048438495857233905

worst: 6.19138668851862e-05

stupid: 6.009287080032778e-05

waste: 2.913593735773468e-05

boring: 7.101884730947828e-05

?: 0.002010379677683693

!: 0.0008121642538468542

UNK: 0.9921223709369025

**thetaNeg**

love: 0.0007775400830054571

wonderful: 8.141780973879131e-05

best: 0.0007124058352144239

great: 0.0005231094275717342

superb: 2.6460788165107175e-05

still: 0.0006716969303450283

beautiful: 0.00014044572179941503

bad: 0.0014125989989680293

worst: 0.0003663801438245609

stupid: 0.0002971750055465883

waste: 0.00016894195520799198

boring: 0.00034806113663333284

?: 0.0031060894415348888

!: 0.0015387966040631559

UNK: 0.9898288801183815

\*Note: “love” includes words love, loving, loves, loved

**Explanation of how thetaPos and thetaNeg were computed (use equations if necessary):**

**1.4**

**Multinomial Naive Bayes Classifier (MNBC) test set Accuracy:**

MNBC classification accuracy = 0.6783333333333333

**sklearn.naive\_bayes.MultinomialNB test set Accuracy:**

Sklearn MultinomialNB accuracy = 0.678333333333

**Explanation of how you test using MNBC (use equations if necessary):**

**For each file, calculate the P(c = Pos|xi) and P(c = Neg|xi).**

**P(Pos) = 0.5; P(Neg) = 0.5**

**n\_xi is the frequency of word xi in this file.**

**CNB has 2 classes, Positive and Negative.**

**If positive score is bigger than the negative score, classify this file as positive. If positive score is smaller than the negative score, classify this file as negative.**

**\*\*MNBC Extra Credit test set accuracy and discussion\*\*:**

MNBC classification accuracy = 0.7733333333333333

In the file I turned in, I used snowballstemmer from NLTK package, and use words that appear more than 100 times to build the dictionary. Building a bigger dictionary increase the accuracy. I also tried to build dictionary using words appear more than 3 times. It generates an accuracy of 0.799.

I also tried WordNetLemmatizer to stem the words. It generates an accuracy of 0.81

**1.5**

**Multinomial Naive Bayes Classifier Testing with non-BOW test set Accuracy:**

Directly MNBC tesing accuracy = 0.6783333333333333

**Explanation of how you test using MNBC with non-BOW (use equations if necessary):**

For each new file:

P\_positive\_file = log(0.5); P\_negative\_file = log(0.5)

iterate through all the words in the file, for each word w in file:

P\_positive\_file += log( thetaPos(w) )

P\_negative\_file += log( thetaNeg(w) )

If (P\_positive\_file > P\_negative\_file):

File is classified as positive

Else:

File is classified as negative

**\*\*MNBC non-Bow testing Extra Credit test set accuracy and discussion\*\*:**

Directly MNBC tesing accuracy = 0.7733333333333333

In the file I turned in, I used snowballstemmer from NLTK package, and use words that appear more than 100 times to build the dictionary. Building a bigger dictionary increase the accuracy. I also tried to build dictionary using words appear more than 3 times. It generates an accuracy of 0.799.

I also tried WordNetLemmatizer to stem the words. It generates an accuracy of 0.81

**1.6**

**thetaPosTrue**

love: 0.40883190883190884

wonderful: 0.12393162393162394

best: 0.4928774928774929

great: 0.41452991452991456

superb: 0.06552706552706553

still: 0.37037037037037035

beautiful: 0.1467236467236467

bad: 0.2621082621082621

worst: 0.045584045584045586

stupid: 0.038461538461538464

waste: 0.022792022792022793

boring: 0.05128205128205128

?: 0.5427350427350427

!: 0.2777777777777778

UNK: 0.9985754985754985

**thetaNegTrue**

love: 0.34045584045584043

wonderful: 0.05270655270655271

best: 0.3504273504273504

great: 0.26638176638176636

superb: 0.018518518518518517

still: 0.3247863247863248

beautiful: 0.08974358974358974

bad: 0.4886039886039886

worst: 0.19230769230769232

stupid: 0.15954415954415954

waste: 0.10541310541310542

boring: 0.18518518518518517

?: 0.688034188034188

!: 0.39886039886039887

UNK: 0.9985754985754985

\*Note: “love” includes words love, loving, loves, loved

**Explanation of how thetaPosTrue and thetaNegTrue were computed (use equations if necessary):**

**thetaPosTrue for each word wi**

**= (#files which include wi and are positive + 1)/(#files that are positive + 2 )**

**thetaNegTrue for each word wi**

**= (#files which include wi and are negative + 1)/(#files that are negative + 2 )**

**Multivariate Bernoulli Naive Bayes Classifier (BNBC) test set Accuracy:**

BNBC classification accuracy = 0.6866666666666666

**Explanation of how you test using BNBC (use equations if necessary):**

For each file f:

For each word wi in predefined dict:

If (wi exists in this file):

pos\_score = pos\_score + log(thetaPosTrue[wi])

neg\_score = neg\_score + log(thetaNegTrue[wi])

else:

pos\_score = pos\_score + log(1-thetaPosTrue[wi])

neg\_score = neg\_score + log(1-thetaNegTrue[wi])

if (pos\_score > neg\_score):

file is classified as positive

else:

file is classified as negative

**\*\*BNBC Extra credit test set accuracy and discussion\*\*:**

BNBC classification accuracy = 0.7783333333333333

In the file I turned in, I used snowballstemmer from NLTK package, and use words that appear more than 100 times to build the dictionary. Building a bigger dictionary increase the accuracy. I also tried to build dictionary using words appear more than 3 times. It generates an accuracy of 0.8.

I also tried WordNetLemmatizer to stem the words. It generates an accuracy of 0.81334