Yonghyun_Lab1

February 8, 2020

0.1 Learn the Naive Bayes Model

0.1.1 Class prior $P(\omega_j)$

[3]:
```
### Class prior $P(\omega_j)$
prior = np.bincount(train_label)[1:] / len_train # p(omega_j)
len_prior = len(prior) # 20

print("1. Class priors")
for i in range(len_prior):
    print("P(Omega = %d)= %g" % (i + 1, prior[i]))
```

```
1. Class priors
P(Omega = 1)= 0.0425947
P(Omega = 2)= 0.0515574
P(Omega = 3)= 0.0507587
P(Omega = 4)= 0.0520898
P(Omega = 5)= 0.0510249
P(Omega = 6)= 0.0525335
P(Omega = 7)= 0.0516461
P(Omega = 8)= 0.0525335
P(Omega = 9)= 0.0528885
P(Omega = 10)= 0.052711
P(Omega = 11)= 0.0530659
P(Omega = 12)= 0.052711
P(Omega = 13)= 0.0524448
P(Omega = 14)= 0.052711
P(Omega = 15)= 0.0526222
P(Omega = 16)= 0.0531547
P(Omega = 17)= 0.0483628
P(Omega = 18)= 0.0500488
P(Omega = 19)= 0.0411749
P(Omega = 20)= 0.0333659
```

We can check that the class priors are almost uniformly distributed.

0.1.2 n and n_k

Assign the total number of words in all documents in class n in the list of length 20, and assign the number of times word w_k occures in all documents in class ω_j in the matrix of row 61188 and column 20.

[4]:
```
# n_k: number of times word w_k occurs in all documents in class omega_j
nk = np.zeros(len_voca*len_prior).reshape(len_voca,len_prior)
# n: total number of words in all documents in class omega_j
n = np.zeros(len_prior)
with open("./20newsgroups/train_data.csv", "r") as f:
    rdr = csv.reader(f)
    for line in rdr:
        docIdx = int(line[0]);
        wordIdx = int(line[1]);
        labelIdx = train_label[docIdx - 1]
        nk[wordIdx - 1, labelIdx - 1] += int(line[2])
        n[labelIdx - 1] += int(line[2])
```

0.1.3 Maximum Likelihood Estimator and Bayesian Estimator

[5]:
```
BE = np.true_divide(nk + 1, n + len_voca)
MLE = np.true_divide(nk, n)
print(BE)
print(MLE)
```

```
[[6.66666667e-05 3.55589754e-04 7.89707479e-05 … 3.48108977e-05
  4.03854386e-06 5.54680393e-06]
 [3.04761905e-04 3.49760414e-04 4.60662696e-04 … 4.90517195e-04
  1.61541755e-04 2.55152981e-04]
 [1.31428571e-03 5.82934024e-06 6.58089566e-06 … 3.16462706e-06
  4.03854386e-06 5.54680393e-05]
 …
 [4.76190476e-06 5.82934024e-06 6.58089566e-06 … 3.16462706e-06
  4.03854386e-06 5.54680393e-06]
 [4.76190476e-06 5.82934024e-06 6.58089566e-06 … 3.16462706e-06
  4.03854386e-06 5.54680393e-06]
 [4.76190476e-06 5.82934024e-06 6.58089566e-06 … 3.16462706e-06
  4.03854386e-06 5.54680393e-06]]
[[8.73585464e-05 5.43685098e-04 1.21189419e-04 … 3.92456977e-05
  0.00000000e+00 0.00000000e+00]
 [4.23352955e-04 5.34623679e-04 7.60188174e-04 … 6.04383744e-04
  2.09198288e-04 3.77846443e-04]
 [1.84796925e-03 0.00000000e+00 0.00000000e+00 … 0.00000000e+00
  0.00000000e+00 7.55692886e-05]
 …
```

```
[0.00000000e+00 0.00000000e+00 0.00000000e+00 … 0.00000000e+00
 0.00000000e+00 0.00000000e+00]
[0.00000000e+00 0.00000000e+00 0.00000000e+00 … 0.00000000e+00
 0.00000000e+00 0.00000000e+00]
[0.00000000e+00 0.00000000e+00 0.00000000e+00 … 0.00000000e+00
 0.00000000e+00 0.00000000e+00]]
```

We can observe that all the values of Bayesian Estimators are positive, while some of Maximum Likelihoood Estimators are equal to zero. There is strong dependency between Bayesian Estimator and ML Estimator with positive correlation, but we can expect that there would be a difference in performance between two classifier, because of zero's in Maximum Likelihood Estimator. Becuase the Bayesian Estimator assigns positive value where ML Estimator is zero, and assins lower values than large ML Estimators, Bayesian Estimator has lower variablity than ML Estimator.

Note that the posterior probability $P(w_k|\omega_j)$ terms may be zero for MLE. Hence we have to make an adjustment to the MLE matrix. Consider the following psudo posterior probability.

$$\begin{pmatrix} 0 & 0 & 0.5 & 0.1 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

After taking log, assuming that $\log(0) = -\infty$, the matrix would be

$$\begin{pmatrix} -\infty & -\infty & -*** & -*** \\ -\infty & -\infty & -\infty & -\infty \end{pmatrix}$$

and the value of $-***$'s would be no longer important. That is, whether the posterior probability is zero or not only counts. In particular, we only have to focus on the number of zero's, because all of the non-infinity values vanishes after added to $-\infty$. Hence, we can assign all the non-zero posterior probabilities to the same fixed value(eg. 0) and zero posterior probabilities to the same value(eg. -1).

[6]:
```
MLE2 = np.where(MLE == 0, -1, 0)
```

0.2 The Performance of the Classifier

0.2.1 Training Data; Bayesian Estimator

[7]:
```
print("Training Data on Bayesian estimator")
prediction11 = prediction(open("./20newsgroups/train_data.csv", "r"), np.
↪log(BE))
PrintResult(prediction11, train_label)
```

```
Training Data on Bayesian estimator
Overall Accuracy = 0.941077
Class Accuracy:
Group 1: 0.966667
Group 2: 0.919105
Group 3: 0.879371
Group 4: 0.930153
```

```
Group 5: 0.94087
Group 6: 0.949324
Group 7: 0.774914
Group 8: 0.966216
Group 9: 0.963087
Group 10: 0.97138
Group 11: 0.978261
Group 12: 0.979798
Group 13: 0.923858
Group 14: 0.976431
Group 15: 0.978078
Group 16: 0.983306
Group 17: 0.985321
Group 18: 0.968085
Group 19: 0.969828
Group 20: 0.760638
Confusion Matrix:
464 0 0 0 0 0 0 0 0 0 1 0 0 0 0 11 0 1 1 2
1 534 6 15 1 9 2 0 1 0 0 2 1 1 2 4 0 0 2 0
1 10 503 23 1 20 2 0 0 0 0 7 1 1 0 2 0 0 1 0
0 10 4 546 4 4 6 2 0 0 0 0 3 0 1 2 0 2 2 1
2 5 2 7 541 3 1 0 2 0 0 2 1 2 2 3 0 1 1 0
0 11 8 1 2 562 0 0 1 1 0 2 0 1 1 0 1 0 1 0
2 3 2 34 6 2 451 17 1 3 3 16 15 5 4 5 5 1 7 0
1 0 0 3 1 2 3 572 1 1 0 1 0 0 0 1 1 1 3 1
0 1 0 1 1 0 4 1 574 0 0 0 0 2 0 2 6 1 3 0
0 3 0 1 0 1 1 3 0 577 4 0 0 1 0 1 2 0 0 0
1 0 1 2 0 1 0 2 0 0 585 1 0 0 0 1 0 2 2 0
0 2 0 0 0 0 0 0 0 0 0 582 0 1 0 0 3 1 5 0
1 4 0 15 5 0 3 2 0 0 1 5 546 2 2 1 2 0 2 0
0 1 0 0 0 1 0 1 0 0 0 1 2 580 0 5 2 0 1 0
2 2 0 1 0 1 0 1 0 0 0 1 0 2 580 1 0 0 2 0
0 0 0 2 0 1 0 0 0 0 1 0 0 0 0 589 1 3 2 0
0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 2 537 2 3 0
1 1 0 0 0 0 0 0 0 1 1 1 0 2 0 6 0 546 5 0
2 2 0 0 0 0 0 0 0 1 0 3 0 1 0 1 2 2 450 0
25 0 0 0 0 0 0 0 2 0 0 0 0 0 1 39 15 4 4 286
```

Overall accuracy of the training data on Bayesian estimator is 0.941077, which is desirable. Groups with small accuracy, for instance group 7, are reflected in the confusion matrix, where 7-th row has a lot of elements in has non-diagonal entries. Even though the accuracy varies by groups, the performance of Bayesian Estimator on the training data is good.

0.2.2 Training Data; Maximum Likelihood Estimator

[8]:
```
print("Training Data of Maximum Likelihood estimator")
prediction21 = prediction(open("./20newsgroups/train_data.csv", "r"), MLE2)
PrintResult(prediction21, train_label)
```

```
Training Data of Maximum Likelihood estimator
Overall Accuracy = 0.979413
Class Accuracy:
Group 1: 1
Group 2: 0.946644
Group 3: 0.937063
Group 4: 0.974446
Group 5: 0.963478
Group 6: 0.981419
Group 7: 0.955326
Group 8: 0.974662
Group 9: 0.998322
Group 10: 0.979798
Group 11: 0.993311
Group 12: 1
Group 13: 0.974619
Group 14: 0.993266
Group 15: 0.989882
Group 16: 1
Group 17: 0.994495
Group 18: 0.987589
Group 19: 0.974138
Group 20: 0.965426
Confusion Matrix:
480 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 550 0 3 0 11 1 1 2 0 1 5 1 0 4 2 0 0 0 0
0 9 536 8 1 4 1 0 3 0 0 3 2 1 3 1 0 0 0 0
0 0 0 572 0 2 0 0 1 1 0 0 5 2 2 2 0 0 0 0
0 3 0 4 554 0 1 0 1 0 1 2 4 0 1 4 0 0 0 0
0 0 0 0 0 581 0 0 2 0 1 2 0 4 2 0 0 0 0 0
0 0 0 7 0 2 556 2 3 0 4 1 3 1 1 2 0 0 0 0
0 0 0 0 0 0 0 577 3 0 3 3 0 0 2 4 0 0 0 0
0 0 0 0 0 0 0 0 595 0 1 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 1 582 9 0 0 0 0 2 0 0 0 0
0 0 0 0 0 0 0 0 0 0 594 0 0 0 0 4 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 594 0 0 0 0 0 0 0 0
0 0 0 0 0 2 0 2 1 0 1 3 576 3 0 3 0 0 0 0
0 0 0 0 0 0 0 0 0 1 0 1 0 590 0 2 0 0 0 0
0 0 0 0 0 0 0 0 1 0 1 1 0 1 587 2 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 599 0 0 0 0
```

```
0 0 0 0 0 0 0 1 0 0 0 0 1 0 0 0 542 1 0 0
0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 5 0 557 0 0
1 0 0 0 0 0 0 0 1 0 1 1 0 0 0 3 3 2 452 0
4 0 0 0 0 0 0 0 1 0 0 0 0 0 0 6 1 1 0 363
```

It is somewhat remarkable that Maximum Likelihood Estimator performs better than Bayesian Estimator in case of classifying the training data. It is possible, however, that the training data are over-fitted, because the overall accuracy is too high(0.979413) and some of the group accuracy is 1, which implies perfect fitting.(eg. Group1, Group12, and Group 16). This may not hold true for the test data.

0.2.3 Test Data; Bayesian Estimator

[9]:
```
print("Test Data on Bayesian estimator")
prediction12 = prediction(open("./20newsgroups/test_data.csv", "r"), np.
↪log(BE))
PrintResult(prediction12, test_label)
```

```
Test Data on Bayesian estimator
Overall Accuracy = 0.781079
Class Accuracy:
Group 1: 0.738994
Group 2: 0.760925
Group 3: 0.529412
Group 4: 0.778061
Group 5: 0.712794
Group 6: 0.784615
Group 7: 0.591623
Group 8: 0.901266
Group 9: 0.889169
Group 10: 0.869018
Group 11: 0.954887
Group 12: 0.913924
Group 13: 0.659033
Group 14: 0.824427
Group 15: 0.854592
Group 16: 0.947236
Group 17: 0.892857
Group 18: 0.864362
Group 19: 0.593548
Group 20: 0.354582
Confusion Matrix:
235 0 0 0 0 1 0 0 0 0 1 1 1 2 3 45 3 10 7 9
3 296 6 12 7 22 1 3 2 0 0 17 4 4 7 4 0 0 1 0
3 33 207 58 11 31 0 2 2 2 1 17 1 4 4 5 0 0 9 1
```

```
0 8 15 305 21 2 4 6 0 0 1 6 23 0 1 0 0 0 0 0
0 8 10 37 273 3 4 4 1 1 0 6 17 8 2 0 3 0 6 0
0 42 7 10 2 306 1 0 2 1 0 10 0 0 3 2 1 1 2 0
0 8 4 50 20 1 226 33 5 0 1 3 11 2 3 4 2 3 6 0
1 1 0 2 0 1 5 356 4 2 0 1 4 0 2 1 4 2 9 0
0 1 0 0 0 0 0 26 353 2 0 1 1 1 0 1 4 2 5 0
4 1 0 1 1 2 3 3 1 345 17 2 2 0 0 3 1 2 9 0
2 0 0 0 0 0 1 1 0 4 381 1 0 2 1 2 0 1 3 0
0 4 1 1 2 1 1 0 0 0 0 361 3 2 0 2 8 0 8 1
2 18 0 27 8 3 1 10 2 0 0 46 259 6 3 6 0 2 0 0
10 7 1 3 0 0 0 4 0 1 0 1 3 324 3 17 3 6 10 0
3 7 0 0 0 2 0 0 1 0 1 4 4 4 335 5 1 2 22 1
7 2 1 0 1 2 0 0 0 0 0 0 1 0 1 0 377 2 2 1 1
1 0 0 0 1 0 1 2 1 1 1 3 0 1 2 3 325 2 16 4
12 1 0 0 0 0 0 2 1 1 1 4 0 0 0 8 3 325 18 0
6 1 0 0 0 1 0 1 0 0 0 3 0 3 7 3 95 5 184 1
47 3 0 0 0 0 0 0 1 0 0 1 0 3 5 70 19 5 8 89
```

Overfitting is possible, but it is not a serious issue in this case because the difference of overall accuracy between training data and test data is not so large(For Bayesian Estimator, Training: 0.941077, Test: 0.781079). This implies that Bayesian Estimator performs well on both training data and test data.

0.2.4 Test Data; Maximum Likelihood Estimator

[10]:
```
print("Test Data of Maximum Likelihood estimator")
prediction22 = prediction(open("./20newsgroups/test_data.csv", "r"), MLE2)
PrintResult(prediction22, test_label)
```

```
Test Data of Maximum Likelihood estimator
Overall Accuracy = 0.715123
Class Accuracy:
Group 1: 0.735849
Group 2: 0.575835
Group 3: 0.347826
Group 4: 0.604592
Group 5: 0.509138
Group 6: 0.738462
Group 7: 0.458115
Group 8: 0.779747
Group 9: 0.899244
Group 10: 0.798489
Group 11: 0.954887
Group 12: 0.901266
Group 13: 0.549618
```

```
Group 14: 0.814249
Group 15: 0.839286
Group 16: 0.90201
Group 17: 0.747253
Group 18: 0.890957
Group 19: 0.641935
Group 20: 0.513944
Confusion Matrix:
234 0 0 0 0 0 0 0 1 0 0 3 0 4 7 37 1 7 5 19
0 224 7 6 3 51 2 1 3 2 3 32 10 17 23 4 0 0 1 0
1 19 136 45 12 53 8 1 6 0 3 41 6 31 18 6 0 1 4 0
0 12 27 237 10 11 11 3 3 0 1 18 46 5 6 2 0 0 0 0
1 24 4 28 195 19 6 3 3 1 3 27 26 29 9 3 1 0 1 0
0 37 7 2 0 288 0 1 3 0 1 14 3 16 13 2 1 1 1 0
0 11 5 39 10 8 175 19 15 2 10 15 19 19 19 5 3 4 2 2
0 2 1 0 1 0 5 308 29 0 1 8 7 4 13 6 3 3 4 0
0 0 0 2 0 1 1 23 357 0 0 1 5 2 1 0 1 2 1 0
0 0 0 1 0 0 1 0 4 317 33 3 0 7 6 11 2 2 10 0
0 1 1 0 0 0 3 0 2 3 381 0 0 0 0 3 1 2 1 1
0 4 0 1 1 2 0 0 1 0 1 356 1 2 6 3 9 3 4 1
3 17 1 14 1 7 8 7 13 0 3 47 216 22 26 3 3 1 1 0
2 5 0 1 0 1 0 3 2 0 0 5 4 320 9 22 7 8 2 2
0 5 0 1 0 4 1 1 2 2 2 11 4 14 329 3 4 4 5 0
10 2 0 0 0 2 1 0 0 0 0 0 0 2 2 359 0 5 4 11
1 0 0 0 0 0 1 2 0 0 2 14 0 11 3 7 272 8 28 15
8 0 0 1 0 0 0 0 1 2 0 4 0 0 0 13 2 335 10 0
5 0 0 0 1 1 0 0 1 0 1 11 1 13 8 4 40 16 199 9
33 0 0 0 0 0 0 0 4 1 0 6 0 7 8 38 13 6 6 129
```

While the Maximum Likelihood Estimator is superior to Bayesain Estimator in term of classification of training data, it turns out that Bayesian Estimator performs better in classifying the test data. This clearly shows that there is a possibility of overfitting when the classification method is based on Maximum Likelihood Estimation. Although Maximum Likelihood Estimates show high accuracy for training data, it may not also hold for test data. Therefore, we can conclude that Bayesian Estimators should be recommended for classifying unclassified data, such as test data.

0.2.5 Appendix: Functions used

[2]:
```python
# Functions
def prediction(stream, logE):
    res = []
    with stream as f:
        rdr = csv.reader(f)
        docIdx = 1
        vec = np.zeros(len_prior)
        for line in rdr:
```

```python
            if(docIdx != int(line[0])):
                res.append(np.argmax(np.log(prior) + vec) + 1)
                docIdx += 1
                vec = np.zeros(len_prior)
            vec = vec + int(line[2]) * logE[int(line[1]) - 1, ]
        res.append(np.argmax(np.log(prior) + vec) + 1)
    return res

def OverallAccuracy(prediction, label):
    return np.sum(np.array(prediction) == np.array(label)) / len(label)

def ClassAccuracy(prediction, label):
    return(np.bincount(np.array(label)[np.array(prediction) == np.
↪array(label)], minlength = len_prior + 1)[1:] / np.bincount(label)[1:])

def ConfusionMatrix(prediction, label):
    mat = np.zeros(len_prior*len_prior).reshape(len_prior,len_prior)
    for i in range(len_prior):
        mat[i:] = np.bincount(np.array(prediction)[np.array(label) == i +␣
↪1], minlength = len_prior + 1)[1:]
    return mat

def PrintMatrix(mat):
    print('\n'.join([' '.join([str(cell) for cell in row]) for row in mat.
↪astype(int)]))

def PrintResult(prediction, label):
    print("Overall Accuracy = %g" % OverallAccuracy(prediction, label))
    print("Class Accuracy:")
    for i in range(len_prior):
        print("Group %d: %g" % (i + 1, ClassAccuracy(prediction, label)[i]))
    print("Confusion Matrix:")
    PrintMatrix(ConfusionMatrix(prediction, label))
    print("")
```