Brianna Shade

CS545: Machine Learning

2/16/14

Project 3: Naïve Bayes Classification

* 1. P(Att1=1|+1) = 4/6 P(Att1=0|+1) = 2/6 P(Att1=1|-1) = 3/6 P(Att1=0|-1) = 3/6

P(Att2=1|+1) = 4/6 P(Att2=0|+1) = 2/6 P(Att2=1|-1) = 2/6 P(Att2=0|-1) = 4/6

P(Att3=1|+1) = 3/6 P(Att3=0|+1) = 3/6 P(Att3=1|-1) = 3/6 P(Att3=0|-1) = 3/6

P(Att1=1|+1) = 4/6 P(Att1=0|+1) = 2/6 P(Att1=1|-1) = 1/6 P(Att1=0|-1) = 5/6

* 1. +1: (4/8)(4/6)(4/6)(3/6)(2/6) = .5 \* .67 \* .67 \* .5 \* .33 = 0.037

-1: (4/8)(3/6)(2/6)(3/6)(5/6) = .5 \* .5 \* .33 \* .5 \* .83 = 0.0347

Classification: +1

|  |  |
| --- | --- |
| Rep | .7 |
| Dem | .3 |

|  |  |  |
| --- | --- | --- |
|  | “Right” | “Left” |
| Rep | .1 | .9 |
| Dem | .9 | .1 |

|  |  |  |
| --- | --- | --- |
|  | True | False |
| “Right” | .8 | .2 |
| “Left” | .5 | .5 |

* 1. P(Dem|”Left”) = (P(“Left”|Dem) \* P(Dem))/P(“Left”)

P(“Left”|Dem) = .1

P(Dem) = .3

P(“Left”) = (P(“Left”|Rep) \* P(Rep)) + (P(“Left”|Dem) \* P(Dem)) = (.9 \* .7)+(.1 \* .3) = .63 + .03 = .69

(.1 \* .3)/.69 = .03/.69 = .0435 = 4.35%

* 1. P(Dem|Five-star, “Right”) = (P(Five-star|”Right”) \* P(“Right”))/P(Five-star)

P(Five-star|”Right”) = .8

P(“Right”) = (P(“Right”|Rep) \* P(Rep)) + (P(“Right”|Dem) \* P(Dem)) = (.1 \* .7)+(.9 \* .3) = .07 + .27 = .34

P(Five-star) = (P(Five-star|”Right”) \* P(“Right”)) + (P(Five-star|”Left”) \* P(“Left”)) = (.8 \* .34) + (.5 \* .69) = .272 + .345 = .617 = 61.7%

Naïve Bayes Classifying Numerical Digits

OVERVIEW

The computing portion of this project explored naïve Bayes classification of numerical digits, represented by 64 features indicating a bitmap of black and white pixels from each digit’s image. Further, a binning technique was used to test for improved accuracy.

IMPLEMENTATION

This project was coded in Java 1.7.0\_09, using Eclipse on a MacBook Pro running Mavericks. Training and test data was stored in external files, read in and assimilated into respective arrays. Probabilities were calculated for each digit, based on the number of occurrences within the training data, and for each digit’s feature, smoothed with Laplace smoothing. Calculating all of these probabilities and overlaying these with the actual features of each test instance produced a probability for classification. The digit classification with the highest probability was then selected as the test instance’s classification.

For binning, the 17 values for each feature were reduced to four bins, abstracting the data and simplifying overall calculations. This was implemented by simply reassigning feature values to a bin from 0-3, based on the original feature value. Classification then proceeded as described above.

RESULTS

Naïve Bayes Test Data:

1797 total instances

Total Accuracy: 97.96%

0: 173(TP), 1617(TN), 2(FP), 5(FN)

1: 153(TP), 1586(TN), 29(FP), 29(FN)

2: 152(TP), 1601(TN), 19(FP), 25(FN)

3: 158(TP), 1600(TN), 14(FP), 25(FN)

4: 170(TP), 1595(TN), 21(FP), 11(FN)

5: 166(TP), 1604(TN), 11(FP), 16(FN)

6: 176(TP), 1611(TN), 5(FP), 5(FN)

7: 169(TP), 1601(TN), 17(FP), 10(FN)

8: 141(TP), 1605(TN), 18(FP), 33(FN)

9: 156(TP), 1570(TN), 47(FP), 24(FN)

Binning Test Data:

1797 total instances

Accuracy: 97.97%

0: 173(TP), 1616(TN), 3(FP), 5(FN)

1: 144(TP), 1587(TN), 28(FP), 38(FN)

2: 157(TP), 1602(TN), 18(FP), 20(FN)

3: 150(TP), 1605(TN), 9(FP), 33(FN)

4: 170(TP), 1598(TN), 18(FP), 11(FN)

5: 174(TP), 1603(TN), 12(FP), 8(FN)

6: 175(TP), 1613(TN), 3(FP), 6(FN)

7: 167(TP), 1601(TN), 17(FP), 12(FN)

8: 147(TP), 1596(TN), 27(FP), 27(FN)

9: 158(TP), 1570(TN), 47(FP), 22(FN)

OBSERVATIONS

Basic naïve Bayes performed rather well on this data set, scoring almost 98% accuracy. I do not believe that these features are necessarily completely independent of each other. The probability that a given pixel is filled would increase with the number of adjacent pixels also filled. For example, when observing a 3x3 pixel matrix, if all edge pixels are filled, it would be very highly likely the center pixel is also filled.

However, this implementation performs quite well, despite this lack of independence. This might be due to the fact that we’re looking at somewhat distinct digits; all of the 7s look similar and not much like other digits. Further, this is a bitmap; a pixel is either filled or not. Therefore, there isn’t any “grey area” (literally) to introduce error on fringe pixels.

Binning the data really doesn’t change the results significantly. For some digits, this increased the accuracy, while for others, accuracy decreased. Overall, accuracy improved slightly. This may be attributed to the fact that the pictorial representations of these digits are very “blocky,” without fine, granular details. Therefore, the features can be safely abstracted without sacrifice to accuracy.

CONCLUSION

The two techniques outlined here were almost equally effective. The process of binning added one more level of computation on the data with very limited added benefit. All features still have to be evaluated - regardless of value - to compute the associated probabilities. Therefore, it hardly seems worth the 0.01% accuracy improvement. I would expect this algorithm would not perform as well with images more detailed than simple numerical digits, but this somewhat rough approach is sufficient for simpler images.