Assignment 5 analysis

– Ksheerasagar Akella

I first tried simple linear classifier with 4 class labels:

\*\*\* original data, 4 class labels \*\*\*

class\_labels:

['acc', 'good', 'unacc', 'vgood']

confusion\_matrix:

[[ 283. 0. 101. 0.]

[ 69. 0. 0. 0.]

[ 30. 0. 1180. 0.]

[ 65. 0. 0. 0.]]

So the 134 entries for ‘good’ and ‘vgood’ are being misclassified. This is not nice, because we are actually interested in the ‘good’ and ‘vgood’ entries during our car buying decision. This seems to be a case where small number of positives are being lost in a sea of negatives. It is actually strange, because 134 out of 1728 is a pretty decent number of samples.

I then combined ‘good’ and ‘vgood’ into 1 class called ‘good’, and ‘acc’ and ‘unacc’ into another class called ‘unacc’. The logic being I want to make the distinction between good and average more well-defined. (Some students are grouping the classes differently. While that may be a valid approach, I do not want to buy cars that are just acceptable, but at least should be good if not vgood).

\*\*\* original data, 2 class labels \*\*\*

class\_labels:

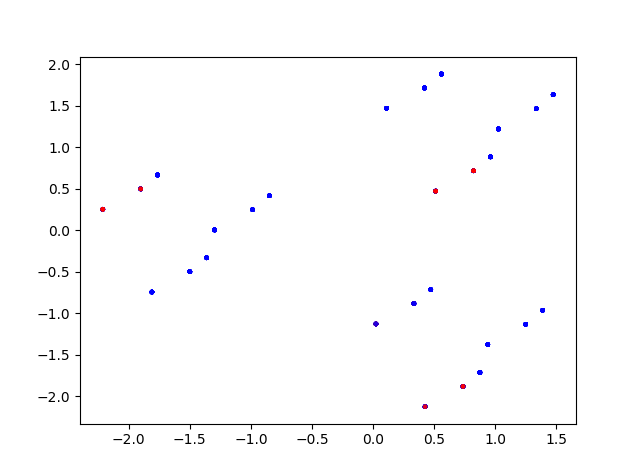
['good', 'unacc']

confusion\_matrix:

[[ 0. 134.]

[ 0. 1594.]]

So that did not help. It is not surprising once I think about it, since the linear classifier did not get any additional help to make its classification job easier. It appears we need to go for quadratic terms, but I do not want to use entire input data of 21 keslerized columns. So I try PCA on those 21 columns. Fortunately, since ALL the input columns have been keslerized, there are no units to deal with, thus making PCA acceptable.

Scatter plot of first 2 PCs just to see the layout of data. My initial suspicion was right, we cannot go with strictly linear classifier, we need to go for quadratic or higher.

I first run linear classifier on all PCA columns:

\*\*\* pca data, 4 class labels \*\*\*

class\_labels:

['acc', 'good', 'unacc', 'vgood']

confusion\_matrix:

[[ 0. 0. 384. 0.]

[ 0. 0. 69. 0.]

[ 0. 0. 1210. 0.]

[ 0. 0. 65. 0.]]

Nothing useful there. Now I start playing around with first few columns of PCA data, along with quadratic terms, but surprisingly first 4-5 PCA columns do not give good classifier. I finally settle on 10 PCA columns.

\*\*\* pca data 10 columns quadratic, 4 class labels \*\*\*

class\_labels:

['acc', 'good', 'unacc', 'vgood']

confusion\_matrix:

[[ 316. 0. 68. 0.]

[ 69. 0. 0. 0.]

[ 108. 0. 1102. 0.]

[ 64. 0. 0. 1.]]

\*\*\* pca data 10 columns quadratic, 2 class labels \*\*\*

class\_labels:

['good', 'unacc']

confusion\_matrix:

[[ 9. 125.]

[ 2. 1592.]]

Having got dismal results with quadratic, I move to cubic. Again I play around with first 4-5 PCA columns, but still do not get very good classifier. Below is output for 10 PCA columns again.

\*\*\* pca data 10 columns cubic, 4 class labels \*\*\*

class\_labels:

['acc', 'good', 'unacc', 'vgood']

confusion\_matrix:

[[ 345. 0. 36. 3.]

[ 45. 20. 1. 3.]

[ 34. 0. 1176. 0.]

[ 27. 0. 0. 38.]]

\*\*\* pca data 10 columns cubic, 2 class labels \*\*\*

class\_labels:

['good', 'unacc']

confusion\_matrix:

[[ 68. 66.]

[ 7. 1587.]]

Finally, after having done all that, I realize that the data set seems to demand at bare minimum quadratic, and ideally cubic terms. Since I have to go with PCA 10 columns to get useful results, I want to see what happens when I run quadratic and cubic with original data set of 21 columns.

\*\*\* original data quadratic, 4 class labels \*\*\*

class\_labels:

['acc', 'good', 'unacc', 'vgood']

confusion\_matrix:

[[ 375. 3. 6. 0.]

[ 48. 16. 0. 5.]

[ 22. 2. 1186. 0.]

[ 20. 0. 0. 45.]]

\*\*\* original data quadratic, 2 class labels \*\*\*

class\_labels:

['good', 'unacc']

confusion\_matrix:

[[ 117. 17.]

[ 2. 1592.]]

\*\*\* original data cubic, 4 class labels \*\*\*

class\_labels:

['acc', 'good', 'unacc', 'vgood']

confusion\_matrix:

[[ 380. 4. 0. 0.]

[ 0. 69. 0. 0.]

[ 0. 0. 1210. 0.]

[ 0. 0. 0. 65.]]

\*\*\* original data cubic, 2 class labels \*\*\*

class\_labels:

['good', 'unacc']

confusion\_matrix:

[[ 134. 0.]

[ 0. 1594.]]

Running cubic on the original data set gives me perfect matches as per my expectations, but of course there is a big risk of over-training here. To be honest, I am not happy with linear classifiers as a good tool to use with this data set. In a real world situation I would definitely try to find a way to split the data set into training and testing sets, while ensuring that each class is adequately represented in each set. I would still not be comfortable with the classifier though…

As a last attempt, I tried playing around with Tau value, i.e. manipulating the Xa[:, 0] column value that is initially set to 1. It is very strange again, that any value other than 1, for example: 0.99999 or 1.00001 will throw the classifier off and give completely different results. Again this leads me to believe that this problem is not a good target for linear classifier. For example:

\*\*\* original data, 4 class labels, with initialized value ‘1’ \*\*\*

class\_labels:

['acc', 'good', 'unacc', 'vgood']

confusion\_matrix:

[[ 283. 0. 101. 0.]

[ 69. 0. 0. 0.]

[ 30. 0. 1180. 0.]

[ 65. 0. 0. 0.]]

\*\*\* original data, 4 class labels, with initialized value ‘0. 99999’ \*\*\*

class\_labels:

['acc', 'good', 'unacc', 'vgood']

confusion\_matrix:

[[ 0. 0. 384. 0.]

[ 0. 0. 69. 0.]

[ 0. 0. 1210. 0.]

[ 0. 0. 65. 0.]]

\*\*\* original data, 4 class labels, with initialized value ‘1.00001’ \*\*\*

class\_labels:

['acc', 'good', 'unacc', 'vgood']

confusion\_matrix:

[[ 384. 0. 0. 0.]

[ 69. 0. 0. 0.]

[ 1210. 0. 0. 0.]

[ 65. 0. 0. 0.]]

\*\*\* original data, 4 class labels, with initialized value ‘-1’ \*\*\*

class\_labels:

['acc', 'good', 'unacc', 'vgood']

confusion\_matrix:

[[ 0. 0. 384. 0.]

[ 0. 0. 69. 0.]

[ 0. 0. 1210. 0.]

[ 0. 0. 65. 0.]]

So any number less than 1 gives all data classified as ‘unacc’, while any number greater than 1 gives all data classified as ‘acc’. With exactly 1, ‘good’ and ‘vgood’ are classified incorrectly. All options seem bleak here.

I would be very interested to receive detailed comments about my analysis, more so than previous assignments since this assignment seems more open-ended and representative of problems we might face in the real world with limited data sets. Also would like to know what would be your recommendation of appropriate algorithm for such data sets, where we have small number of very important positive values that need to be classified correctly. Do we need to use weighted average for initial values somehow??