Workshop *Learning from Data*

Session 4: kNN and Naive Bayes

TO TRY:

The program **kNN.py** uses KFold cross validation to create and test kNN models that attempt to predict the species of each animal. Sample output, with k=5 is given below:

```
(cv) workshop@vitalstatistix:~/working/4$ python3 kNN.py
Loading the images from  /data/WP/WilkesPets/stretched128/
Loaded  2000 images
Performing PCA with n_components = 0.95 reduced dimensions from 49152 to 673
Running the kNN classifier, k =  5
Split  1  Result:
[[202  14   0   0]
 [ 59 123   0   0]
 [  1   0   0   0]
 [  1   0   0   0]] 81.65829145728644

Split  2  Result:
[[228  21   0   0   0]
 [ 45 103   0   0   0]
 [  1   0   0   0   0]
 [  0   1   0   0   0]
 [  1   0   0   0   0]] 83.37531486146096

Split  3  Result:
[[204  13   0]
 [ 52 129   0]
 [  1   1   0]] 83.66834170854271

Split  4  Result:
[[202  20   0]
 [ 58 118   0]
 [  0   2   0]] 80.40201005025126

Split  5  Result:
[[219  16   0   0]
 [ 45 117   0   0]
 [  1   1   0   0]
 [  1   0   0   0]] 84.63476070528966

Aggregate results:
[81.65829146 83.37531486 83.66834171 80.40201005 84.63476071]
Accuracy = 82.74774375656621 +/- 1.5160179404611127
```

Again, this looks like a nice result!

You might edit the program to change the *SourceDirectory* to point to your data. You can also change the PCA parameter (lower it to further reduce dimensionality) and the kNN parameter *kNNK* (which indicates how many near-neighbors to consider).

A second program, **naivebayes.py**, doesn't provide any model-specific parameters and because PCA may generate negative valued data, we only make use of SciKit-Learn's NBGaussian() model[1]. It produces the following output:

```
((cv) workshop@vitalstatistix:~/working/4$ python3 naivebayes.py
Loading the images from  /data/WP/WilkesPets/padded128/
Loaded  2000 images
Performing PCA with n_components = 0.95 reduced dimensions from 49152 to 652
Running the Naive Bayes classifier,
Split  1  Result:
[[128 104   0   0]
 [ 34 131   0   0]
 [  0   2   0   0]
 [  0   1   0   0]] 65.23929471032746

Split  2  Result:
[[123 100   0   0   0]
 [ 34 138   0   0   0]
 [  0   2   0   0   0]
 [  2   0   0   0   0]
 [  0   1   0   0   0]] 66.07594936708861

Split  3  Result:
[[105 116   1   0]
 [ 50 126   0   0]
 [  0   1   0   0]
 [  0   1   0   0]] 58.18639798488665

Split  4  Result:
[[122 105   0]
 [ 41 131   0]
 [  0   1   0]] 63.40852130325815

Split  5  Result:
[[135 100   0]
 [ 61 103   0]
 [  0   1   0]] 59.64912280701754

Aggregate results:
[65.23929471 66.07594937 58.18639798 63.4085213  59.64912281]
Accuracy = 62.51185723451569 +/- 3.0935584846157016
```

The accuracy is only slightly better than guessing every animal is a Cat (which would approach 57% accuracy), even though an examination of the confusion matrices show that the classifier is indeed making real decisions. For this dataset, Naive Bayes is certainly worse than any of the other machine learning methods we've tried.

As with the other classifiers, change the input to try out your version of the dataset.

Our concluding video segment will consider how you might design and analyze experiments.

[1]If the image dataset is represented as histograms (discussed in Session 2) you can also try using Scikit-Learn's MultinomialNB() model.