Workshop *Learning from Data*

Session 3: Support Vector Machines

BACKGROUND:

Our initial research using SVMs led our students to make a surprising discovery. When trying to predict nose color or ear morphology in dogs, they could vary image resolution over a large range with little effect. In fact, a single pixel proved as effective as the original images. This led us to quickly review canine genetics where we learned two things:

1) Coat and nose color is genetically linked in dogs. (S*ee TYRP1 and MC1R genotypes and their effects on coat color in dogs*, Schmutz et al. Mammalian Genome (2002) 13:380.)

2) Coat color and ear morphology appears to be proximally linked. (See *Linked genetic variants on chromosome 10 control ear morphology and body mass among dog breeds*, Webster et al. BMC Genomics (2015) 16:474.)

These linkages should make these specific machine learning problems easier.

TO TRY:

The program **svm.py** uses KFold cross validation to create and test SVM models that attempt to predict the species of each animal. A C value of 0.001 was arbitrarily chosen. Sample output is given below:

```
(cv) workshop@vitalstatistix:~/working/3$ python3 svm.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 SVM classifier, C =  0.001
Split  1  Result:
[[179  36   0]
 [ 45 139   0]
 [  0   0   1]] 79.69924812030075

Split  2  Result:
[[199  31   0   0]
 [ 42 126   0   0]
 [  1   0   0   0]
 [  0   1   0   0]] 81.65829145728644

Split  3  Result:
[[195  47   0   0]
 [ 17 138   0   0]
 [  0   2   0   0]
 [  0   1   0   0]] 83.87909319899244
```

```
Split   4  Result:
[[188   35    0]
 [ 39 137    0]
 [  0    1    0]] 81.45363408521304

Split   5  Result:
[[195   34    0    0    0]
 [ 34 132    0    0    0]
 [  1    2    0    0    0]
 [  1    0    0    0    0]
 [  0    0    0    0    1]] 82.78481012658227

Aggregate results:
[79.69924812 81.65829146 83.8790932  81.45363409 82.78481013]
Accuracy = 81.895015397675 +/- 1.4000392140653095
```

This looks like a better result than those achieved using Decision Trees![1]

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 SVM parameter *SVMC* (raise it to decrease the margin, which increases the computational complexity and slows execution.)

A second program, **dogtag.py**, is provided to illustrate exploring a subset of the images – here just the dogs, where the program attempts to classify them on the presence of one specific tag – say black noses or floppy ears. You can chose the tag by setting the *t* variable near the top of the code to the feature tag of interest. (A complete taglist can be found in the dataset description that is distributed with the images, but here 'k' indicates a black nose, 'f' represents floppy ears.)

The program loops over the C values in the *Clist* array. Change that list as you wish.

Based on literature, we expect coat color to predict both nose color and ear morphology in dogs. Because OpenCV averages pixel colors when reducing resolution, a very low resolution image will still retain the animal's coat color. (In fact, we discovered that literature to explain our own results.) To test this this program also demonstrates a call to resize the image based on the *NewSize* variable (here set to 4.)

Partial sample output is given below:

```
(cv) workshop@vitalstatistix:~/working/3$ python3 dogtag.py
Loading the images from  /data/WP/WilkesPets/stretched128/
Loaded  849  images,  649  with  k  tag ( 76.44 %)
Performing PCA with n_components = 0.95 reduced dimensions from 48 to 16
Running the SVM classifier, C = 0.001
```

[1]Note: as in the Session 2 results, because the dataset is imbalanced (with cats and dogs being much better represented than the other species), our experimental confusion matrices here vary in size when, at the validating stage of the KFold algorithm, some classes were not presented to the model.

```
Split  1  Result:
[[131   0]
 [ 39   0]] 77.05882352941177

Split  2  Result:
[[134   0]
 [ 36   0]] 78.82352941176471

Split  3  Result:
[[132   0]
 [ 38   0]] 77.64705882352942

Split  4  Result:
[[124   0]
 [ 46   0]] 72.94117647058823

Split  5  Result:
[[128   0]
 [ 41   0]] 75.7396449704142

 Aggregate results:
   [77.05882353 78.82352941 77.64705882 72.94117647 75.73964497]
  Accuracy = 76.44204664114167 +/- 2.0126295404019596

Running the SVM classifier, C = 0.01
Split  1  Result:
[[129   0]
 [ 41   0]] 75.88235294117646

Split  2  Result:
[[129   0]
 [ 41   0]] 75.88235294117646

Split  3  Result:
[[129   0]
 [ 41   0]] 75.88235294117646

Split  4  Result:
[[137   0]
 [ 33   0]] 80.58823529411765
```

The accuracy seems good, but take a close look at the confusion matrices. What do you notice?

These classifiers has learned to claim that all dogs have black noses! Our performance, 76.44% matches the population of dogs with black noses. That isn't a very intelligent classifier.

You can change the *SourceDirectory* variable to point to other input data. Can you make data preparation choices to produce better results? A model that actually does something useful?

Our next video segment will introduce two more learning methods.