Tag Me! – Implementation Write-up

Prepared by – Team ML-Addicts

Brief Summary:

We used the given features for each image and we also used some image processing techniques to extract our own features and we used them together as the data for training our classifier. For classification, we used an ensemble approach, Random Forest to be more precise and created the model and then performed prediction on both the validation as well as the test dataset as was asked.

Software used:

Development Environment: Python 2.7.5 |Anaconda 1.8.0 (64-bit)| (default, Jul 1 2013, 12:37:52) [MSC v.1500 64 bit (AMD64)] on win32

Following are the software\IDEs which were used by us.

- Python programming language.
- Spyder IDE for scientific python development.
- Mahotas open source image processing python library module created by MIT.
- NumPy, SciPy, scikit-learn python modules for Machine Learning.

Links to necessary software required to run the code:

- Anaconda 1.8.0 64 bit for 64 bit Windows: http://repo.continuum.io/archive/Anaconda-1.8.0-Windows-x86_64.exe
- Mahotas Python Module for 64 bit Windows: http://www.lfd.uci.edu/~gohlke/pythonlibs/3m3k3j67/mahotas-1.1.0.win-amd64-py2.7.exe

Feature Extraction Process:

Following were the main approaches for feature extraction.

- Read default (given) features from file
- Using sobel's filter for edge formation and extracting a global feature per image
- Using Haralick texture features
- Speeded-Up Robust Features (SURF)
- PCA (abandoned in final code)

Classifier:

We have used the Random Forest Classifier algorithm for classification. This falls under the ensemble classifiers.

References:

1. http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=4309314
2. http://www.packtpub.com/building-machine-learning-systems-with-python/book

Feature Extraction Steps:

- Use the default features available for each image by reading from given file.
- Apply Harlick Texture Feature extraction algorithm for each image.
  - Read in the image in the form of matrix of pixel values.
  - Get the required features in the form of a 4x13 matrix for each direction (up, down, left, right)
  - Use mean to get 13 features for each image based on mean overall directions
- Apply Sobel's Filtering algorithm for each image
  - Read the image in grayscale format in the form of matrix of pixel values
  - Filter\Convolve the image with two filtering matrices
  - Sum up the squared result of the values of edges to get overall edginess for each image
- Apply the SURF algorithm to extract features for each image
  - Read in grayscale image in from of matrix of pixel values
  - Extract value of descriptors computed on points that are at a distance of 16 pixels from each other
  - Use every 32nd vector from this array of descriptors as a feature for that image
  - Append all the required features from the descriptor array
- Combine all the above features for each image to get the feature set.

Training Algorithm and Prediction Algorithm:

- **Input format:** Array of 382 features for each image
- **Tuneable parameters:** Number of estimators, criterion, minimum samples leaf, maximum depth, minimum samples split, random state, maximum features
- **Output format:** File containing image name and class label per row

```
Let N trees be the number of trees to build
for each of N trees iterations
        1. Select a new bootstrap sample from training set
        2. Grow an un-pruned tree on this bootstrap.
        3. At each internal node, randomly select m predictors and
           determine the best split using only these predictors.
        4. Do not perform cost complexity pruning. Save tree as is,
           alongside those built thus far.

Output overall prediction as the majority vote (classification) from all individually trained trees.
( Prediction )
```

Validation and Parameter tuning:

Validation was done on the test and validation data and the following parameters were tuned accordingly in the classifier.

- **n_estimators:** The number of trees in the forest. (630)
- **criterion:** The function to measure the quality of a split. ('entropy')
- **max_features:** The number of features to consider when looking for the best split ('None' == all the features)
- **min_samples_split:** The minimum number of samples required to split an internal node. (1)

Interpretation of results on validation data:

The accuracy obtained was quite decent and it stood at around 76.5% the last time it was run. This accuracy was obtained because we used the ensemble approach which is often better than the approach used by a single classifier which includes,

- A different subset of the training data are selected, with replacement, to train each tree.
- Remaining training data are used to estimate error and variable importance.
- Class assignment is made by the number of votes from all of the trees.

There is definitely scope for improvement because the feature extraction wasn't perfect and there exist lots of better methods to extract proper features to classify the images in a better way with more confidence.

For any doubts related to the code or write-up please contact:

Dipanjan Sarkar
IIIT Bangalore
dipanzan.sarkar@gmail.com
9980960048