

BIOMETRICS

Fall 2012

Assignment 6: Face Verification

Due: ~~November 15~~ **November 20**, 2012 at 5pm by email



Lab Description:

In this lab you will build a *face verification* system. In the face verification problem, you are presented with two face images and must determine if they are of the same person or of different people. You will evaluate the performance of your system using the Labeled Faces in the Wild (LFW) benchmark from the University of Massachusetts.

The Dataset:

First, you should download the dataset from the course web page. There are 13,233 images of 5,749 people. The images we will use are the “LFW-a,” aligned versions of the original LFW images. LFW-a was produced by Lior Wolf, Tal Hassner, and Yaniv Taigman, and a brief description of the alignment procedure is available at <http://www.openu.ac.il/home/hassner/data/lfwa/>.

The Benchmark:

The folds file, `pairs.txt`, defines ten folds for the cross-validation experiment you will use to evaluate your system. This is a tab-delimited file. The first line indicates that there are 10 folds, each containing 300 “same” pairs and 300 “different” pairs.

```
10      300
```

The next 300 lines have three fields each, and define the “same” pairs in the first fold. For example, the first pair is described by the line

```
Abel_Pacheco 1      4
```

This indicates that the first “same” pair of the first fold consists of images 1 and 4 of Abel Pacheco. In the downloaded images directory, the paths to these files will be `Abel_Pacheco/Abel_Pacheco_0001.jpg` and `Abel_Pacheco/Abel_Pacheco_0004.jpg`. After the 300 lines describing “same” pairs, there are 300 lines describing the “different” pairs in the first fold. The first of these is

```
Abdel_Madi_Shabneh 1      Dean_Barker 1
```

This indicates that the first “different” pair of the first fold is the images `Abdel_Madi_Shabneh/Abdel_Madi_Shabneh_0001.jpg` and `Dean_Barker/Dean_Barker_0001.jpg`. After the 600 lines that describe fold 1, there are 600 lines for fold 2, 600 lines for fold 3, etc. up to fold 10. There is a matlab function to parse this file, `read_lfw_folds()`, on the course web page.

Your program should read this file and perform the cross-validation experiment. First train on folds 2-10 and evaluate the pairs in fold 1, then train on folds 1 and 3-10 and evaluate on fold 2, etc. Your program should output a file that is a copy of `pairs.txt`, but with each line prefixed by a decision value and an additional tab character. A positive decision value indicates that your program classifies this as a “same” pair, and a negative value indicates that it is a “different” pair. For example the first two lines of output might be

```
10      300
1.1211      Abel_Pacheco 1      4
-0.2455      Akhmed_Zakayev 1      3
```

if your classifier decides that the first pair is “same” (correct) and the second pair is different (incorrect).

As for the previous labs, you are to consider this lab as a friendly competition between you and the other class members. The winner will get a guaranteed increase in their letter grade by 2/3 – as explained in class (twice the bonus granted for previous labs).

What to Program:

Unlike the previous labs, for this lab you are permitted to use a couple software packages, **vlfeat**, a library that includes functions for extracting SIFT features from images, and **libsvm**, a popular implementation of support vector machines. We recommend you implement your

verification system as an SVM using SIFT features. You must use Matlab. Vlfeat and libsvm include Matlab interfaces.

Recommended Approach:

1. Define some square windows on the face that seem to contain important information. For example, you might have a window around each eye, including the eyebrow, and another window around the mouth. Using vlfeat, extract a SIFT descriptor from each of these windows (with center point at the center of the window, and binsize equal to a quarter of the window side length). Concatenate the SIFT descriptors to get a feature vector describing the face.
2. To obtain a feature vector for a pair of images, take the absolute difference of the feature vectors for the two images.
3. Use an SVM to classify image pairs as “same” or “different.”

Tools:

You may (and are encouraged to) use **vlfeat** for feature extraction and **libsvm** to build your classifier. Both have Matlab interfaces. Vlfeat is available from <http://www.vlfeat.org/>. Libsvm is available from <http://www.csie.ntu.edu.tw/~cjlin/libsvm/>. A small matlab file demonstrating how to use these functions is available on the course web page.

What to Hand In:

Students must hand in a zip file, which should be named “hw6_YourUNI.zip.” For example, if your UNI is xy1234, the filename should be hw5_xy1234.zip. Please send this zip file to tberg@cs.columbia.edu by the due date. Included in this zip should be:

1. A well-written report on the experimentation done, including the logic behind your choice of classifier. This should be a pdf document, named report.pdf. In this lab, unlike in previous labs, you know the true class (“same” or “different”) for each. You should therefore calculate and report on your accuracy. The LFW maintainers ask all researchers reporting results on the benchmark to include at least the mean and standard deviations of the accuracy rates over the ten folds, so please include at least that information. Please also include an ROC curve, as described in class, in your report. A function to generate the points on an ROC curve is available on the course web page.
2. Your well-commented source code. You should have one **function**, verifier.m, which should be callable like this:

```
>> verifier(path-of-folds-file, path-to-root-of-aligned-image-tree, ...  
path-for-output-file)
```
3. The output file produced by running your verifier using the pairs.txt file. The format of this file is described above. Please name this file pairs_out.txt.

Hints:

1. When the location you want to extract SIFT is very close to the image border, the SIFT window is likely to extend outside the image. In that case, `vl_phow()` will return an empty array, and `vl_sift()` will return a descriptor by padding the out-of-image region with 0, which introduces many 0's in the descriptor.
2. When you call `libsvm` to train and test, the feature vectors should be normalized so that the mean of each feature dimension is 0 and the standard deviation is 1.
3. When you call `libsvm` to train and test, the feature vectors should not contain value "inf" or "nan", which is likely to happen during the feature normalization. So when normalizing the features, add a small number "eps" to the standard deviation of each feature dimension.
4. Be sure to experiment with SIFT descriptors at different scales. You may even want to extract descriptors from the same points at different scales.
5. Having too many features makes the job of the classifier difficult. Each SIFT descriptor adds 128 to the overall feature dimension. Your overall feature dimension should probably not be more than a few thousand.
6. Note that chance accuracy on verification is 50%. If you are around or below 50%, something is wrong.
7. You can learn more about the LFW dataset at its home, <http://vis-www.cs.umass.edu/lfw/>.