CSC 411: bake off

Rohan Chandra and Alexander Kondratskiy
996274142 and 996366285

a) Initially we experimented with applying different models of the same classifier to different parts of the face as we thought better results may have been achieved by breaking up the face into relevant chunks, i.e the eyes, nose, mouth, etc. The results from each of the classifiers on their respective face component would then be weighted separately according to how well a classifier could predict the emotion of the entire face from that component in isolation. In isolated testing, the eyes provided the strongest indicator of what the correct emotion was, followed by the nose, left cheek, right cheek and finally the mouth and the weighting was awarded proportionally. Hence, the more components of the face that agreed on a particular emotion, weighted by their validity, the more likely the entire face displayed the given emotion. The classifier used on each portion of the face was multinomial logistic regression with L2-regularization, though other classifiers (neural networks), were experimented with and found to be less effective. This may in part be due to the sparseness of the data as, when training with a set aside validation set, the number of points in the training set is less than the number of dimensions we are considering.

Separately, in order to reduce differences between the images that are being compared, whitening was performed based on the variance obtained from PCA decomposition. It was thought that this would help reduce errors in classification on faces with similar emotions that differ in skin color or image lighting. After processing the images in this way, a number of different classifiers were experimented with to generate a model to classify the data, such as SVMM, both linear and with a polynomial kernel, multinomial linear regression with l2 regularization and neural networks. However, these methods failed to provided better results than the base line classifier.

Subsequently, we experimented with various forms of dimensionality reduction, particularly PCA. It was hoped that by projecting the multidimensional data to a simpler space, faces displaying similar emotions would be projected close to other faces displaying the same emotion. Thus, we could project our labeled examples to the principal components and then classify test data using k nearest neighbors. By itself, we found this not be an effective form of classification, but it appeared as though we could begin to label examples in the unlabelled set using this method. Specifically we could break the unlabelled data set up into chunks, projecting it onto 55 of the principal components (obtained from unlabelled data), and then using k nearest neighbours to try and obtain new training examples. If the 12 nearest neighbors from the training set all had the same class, then we consider it to be a confident indicator that the given unlabeled example was displaying the same emotion, and could be added to the training set as such. Although we may only find a small percentage of these example, we are able to expand our training set and create better models of the data.

b)

In general, our algorithim functions by preprocessing all the faces to normalize the range of colors, and reduce the area of the face we are examining. PCA decomposition is performed on the processed unlabelled faces and the top principal components are selected. The training set is then projected onto this basis, as is the unlabeled set, to allow for the classification of new examples to be added to the training set. Given this expanded set of training data and pre processed set of faces, we then train a multinomial logistic regression classifier with L2-regularization.

We attempted to improve our results through several forms of data processing. We first normalize the image by calculating the mean of each face, subtracting that from each image, and finally dividing by the norm of this image. We then apply an oval shaped mask meant to isolate the relevant parts of the image and cut out the background that may appear under the chin and to the sides of the face. This mask can be seen in detail in the figure below,

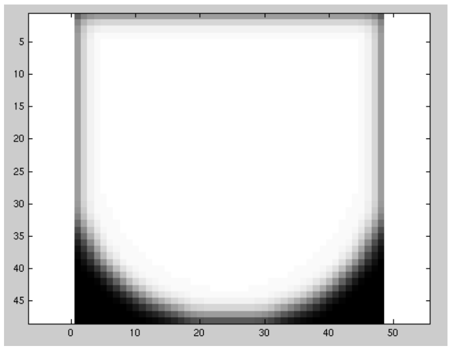
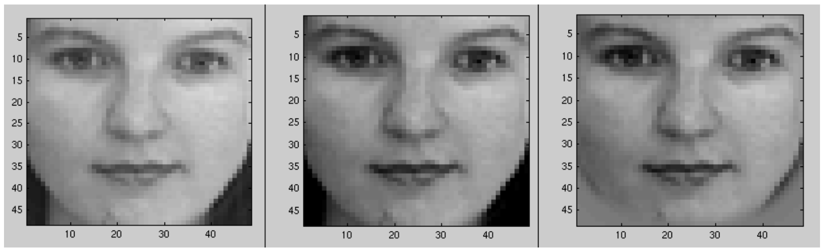


Figure 1: a grey scale image of the facemask applied to each face

The full effects of the preprocessing are demonstrated in the following figure. The left most image is the initial face, the middle face is after normalization, and the right most image is the face with the mask applied at the end of processing. Experiments with gaussian blurring and whitening in conjunction and separately proved to perform worse.



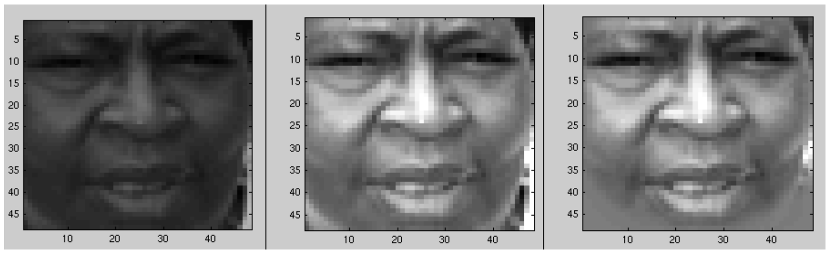


figure 2: The effects of pre-processing on two extremely different faces

After prepossessing the training data, we divide the unlabeled data into several sets of chunks, such that we can process them without exhausting the limitations of a typical cdf machine's resources. For each chunk of the unlabeled data we project the faces onto the precomputed top 55 components of the PCA decomposition of the unlabelled set. As we have also projected the labeled training example onto the same components, we can use k nearest-neighbors to assess what the probable emotion of the face from the unlabeled set. If 12 of the nearest neighbours from the training set all share the same emotion, the unlabeled face is labeled with the same emotion and added to the training set.

When testing locally and attempting this expansion using a training set and a set aside validation set, using 55 components and 12 neighbors gave us an accuracy of 100% in the relabeling, but were only able to obtain around 5% of the validation set as new examples. However, given the volume of images in the unlabelled set, we were able obtain around 1500 extra examples after one pass. In theory, we could keep expanding the training set by running multiple passes on the unlabeled set, but we run into memory problems on the second pass (now that the training set is much larger, k nearest neighbours takes up a lot more memory, even with careful chunk-by-chunk processing).

Finally, given this set of expanded training examples with the aforementioned prepossessing, a multinomial logistic regression classifier with L2-regularization (where lambda is 5*10^-3) is used to generate a model. When training this model, we changed the "tolX" option to 10^-3, to halt training earlier, thus avoiding overfitting.

c)

When separating the training data into random sets of training and validation data, the algorithim performs on average with accuracy of 76%. The breakdown of the effectiveness of our steps on the kaggle test set can be seen in the following:

Method used	**Accuracy on Test set**	**Accuracy on Validation set**
normalizing + masking + PCA assisted training set expansion + MLR classifier with penalization of 5e-3 run with new tolX	71.05623	74.12
normalizing + masking + PCA assisted training set expansion + MLR classifier with penalization of 5e-3	70.57416	76.06

normalizing + masking + PCA assisted training set expansion + MLR classifier with penalization of 1e-2	67.46411	75.54
similar masking/normalizing as below but with linear svmm	66.74641	72.12
blurring + normalize + masking + Softmax classifier with penalization of 1e-2	70.09569	74.02
normalizing + masking + regularized weighting MLR	70.33493	74.43
masking + normalization + MLR	69.85646	74.02
Simple MLR with face normalization	68.18182	73.33

Below we compare our algorithms performance, to that of the given handout classifier and our pca decomposition with k-nearest neighbours classifier.

As the handout code baseline achieves an average of 69.74 accuracy on the set aside validation code and 68.89% on the kaggle test set, it is clear that our algorithm performs better. This improvement is likely a result of regularization, better initial data processing and limiting over fitting by using less iterations than the default tolerance would allow for.

In comparison, our experiments with a PCA based decomposition with k-nearest neighbours as a classifier yielded the following results on the set aside validation set.

	100 components	**75 components**	**50 components**	**20 components**	**10 components**
12 neighbours			50.60%	50.77%	
8 neighbours	51.45%	51.97%	51.97%	**52.48%**	46.15%
5 neighbours	50.26%	51.62%	49.91%	50.26%	

Narrowing the search around 20 components and 8 neighbours

	18 components	**19 components**	**20 components**
10 neighbours		52.14%	
9 neighbours	52.14%	**53.16%**	52.65%
8 neighbours		52.82%	52.48%

We tried comparing the performance of PCA between decomposing just the training set and the unlabelled set. Running this using PCA obtained from just the labeled set, gives 51.28%. Almost a 2% reduction.

After adding normalization to the faces and using the above optimal of 19 components and 9 neighbours, an accuracy of 61.88% was obtained using PCA components of the labeled training data.

We also found that normalizing the data allowed us to use more PCA components without degrading performance- intuitively, we decreased the useless and misleading variation in the data, that might arise from differences in skin color or image lighting. After this normalization, the PCA components that would represent those variations are no longer present:

Neighbours/comps	**40**	**50**	**55**
15		64.79%	
12		65.13%	**65.64%**
9	62.56%	63.76%	

d) Though the algorithm seems to perform well locally, results of the model on the kaggle set seem to average around 71.05%. This could be the result of a bias in either the training or test sets. It may also be a case of over fitting, despite using less iterations in our training and regularization to combat such overfitting. We could further improve our results with increased preprocessing. Though we normalize and try to isolate the area of the face we are looking at, there isn't any guarantee that the facial features of each training face actually overlap. It may be possible to fix this by computing a homography and warping all of the test faces to one common face, such that the major facial features overlap and are of similar scale. Computing this homography would involve blurring each face by a large radius blurring kernel, calculating the gradient on each image and then computing an optimal homography per face. This homography could then be used on the unblurred faces to align their facial features and improve our results.

e) No outside sources were used.

f) The topmost script is classifyScript.m

Since we do processing on the unlabelled set (which takes up a lot of space in memory), we choose to store it in a global variable called **BIGDATA**, to avoid copy on write operations typically done by matlab when passing variables into functions. As such you will see this being declared in classifyScript.m, and we request that the unlabelled data set is not passed to any functions using other methods.

Additionally, a lot of intermediate files are generated when training the model, so these .mat files should be deleted if a new training or test set is used when running the script.