CIS 520 Project Report

Mike Phillips
James Anderson
Chinwendu Enyioha

Group name: Something naïve (We just love the umlaut over the i, that's why)

TASKS

1. Given the first 1000 words in 1700 blogs, develop age (13-17, 23-27, 33-47) and gender (Male, Female) prediction algorithms, with a goal of beating a baseline prediction accuracies of 73.24% (for gender) and 70.23% (for age).
2. Given 488 images, develop age (under 30, 35-50, over 55) and gender (Male, Female) prediction algorithms, with a goal of beating a baseline prediction accuracies of 71.8% (for gender) and 48.6% (for age).

Based on these goals, we explored a number of classifiers and sets of features before concluding on which one(s) to use. This report briefly presents some of the ideas we explored, and discusses the approaches we chose to implement for our final submission. It consists of two main sections – work done for images and for blogs.

IMAGES

The two major classifiers we used for images were the Support Vector Machine, SVM, and Sparse Multinomial Logistic Regression, SMLR. Determining the most viable and reasonable features for the classifier was by far the most tedious component of the project. Some of the features we used included:

1. The Haar Wavelets algorithm,[1] which detects edges of different orientations at many resolutions of the image. We used these features on the cropped area just around the face. The different resolutions were particularly useful since as many pixels get blurred together into one, lower resolution versions of images tend to be more robust to small translations. These worked particularly well with SVM for determining gender and *we used this in our final submission.*
2. Pixel color values of key regions of the image (such as the mouth region, eye region, top hair region, side hair region, and cheek region) which we thought may have been able to distinguish images of different gender well. For instance, men may have facial hair which would show up in the mouth and cheek region and women tend to have longer hair showing up in top head region. Extracting the key regions was done with hard-coded cropping. This approach worked reasonably well. Being given images that were more or less centered and upright helped this process. These features predicted gender and age well, but not as well as wavelets and Principal Component Analysis, PCA, thus *we did not use the key regions in our final submission.*
3. PCA was used to reduce the dimensionality of the images and eliminate features that were not relatively useful. We computed the principal components on the RGB values of an image that was cropped around the face. After looking at the principal components' corresponding eigenvalues, it seemed like using the first 100 was sufficient to capture most of the useful information. This enabled us transform our images into a lower dimensional space by subtracting the mean face from the cropped image, and multiplying by our top 100 principal components. We then used the

[1] Pascal Getreuer, MATLAB Central – File Detail – Wavelet Transforms in MATLAB, http://www.mathworks.com/matlabcentral/fileexchange/11133-wavelet-transforms-in-matlab

resulting 100 values per image as our features. This method worked reasonably well with SMLR on predicting age and was used in our submission. Below is a visualization of the top 25 eigenfaces produced by PCA (we actually used 100 but they would be too small to see).

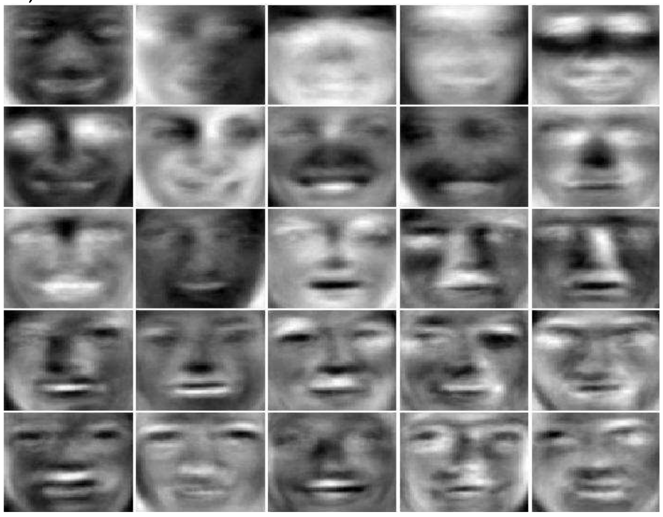
Figure 1: Visualization of the top 25 eigenfaces produced by PCA

4. Experimenting with variants on the original features such as the pixel color values from downscaled images provided in the baseline.
5. Color Histogram, an attempt at making our features somewhat less brittle to translations and rotations. We divided the image into small squares of pixels and computed a color histogram within each. Then these histograms were given to the classifiers as features. These features actually did not work nearly as well as we had thought they would.

At the suggestion of David Weiss, we added an additional virtual feature – flipping the images horizontally (i.e. the left side is now the right side and vice versa). This was done to prevent the classifier from using the head position (pose) as a feature. As will be seen later in the results, flipping the faces enhanced our percentage accuracy further. Another plus was that it enriched the data set we worked with.

We also looked at different image formats[2]; i.e. using *HSV*, *YUV*, and *LAB* instead of *RGB.* Since each of these formats has a component dedicated to brightness, and the remaining two components define color, independent of brightness, our hope was to eliminate shadow and/or lighting issues. This did not, however, increase our accuracy.

Results from Images:

SVM

We used *libsvm* as our implementation for Support Vector Machines[3]. In all of our uses with the SVM we always used the radial basis kernel that the baseline used. We used 6-fold cross validation to choose the optimal penalty term for the SVM. After we knew which constant we were using, we retrained the SVM using all the data.

[2] Pascal Getreuer, MATLAB Central – File Detail – Color Space Converter, http://www.mathworks.com/matlabcentral/fileexchange/7744-color-space-converter
[3] Chih-Chung Chang and Chih-Jen Lin, LIBSVM – A Library for Support Vector Machines, http://www.csie.ntu.edu.tw/~cjlin/libsvm/

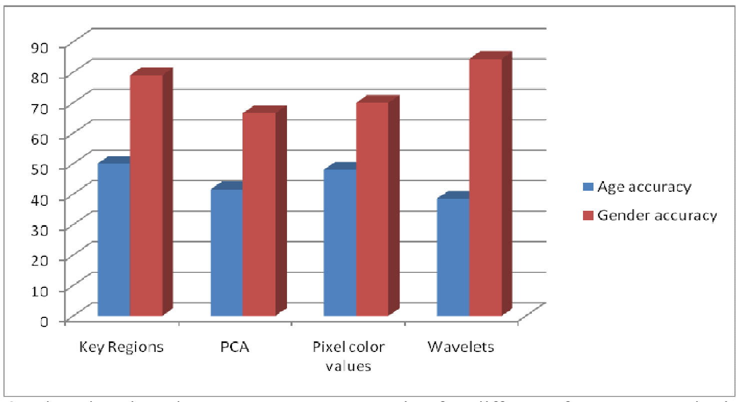

Figure 2: Plot showing the percentage accuracies for different features used with SVM

We found that PCA and SVM did not work well together. We believe that this because the first two principal components have much larger eigenvalues. Since SVM tries to maximize the margin, it tends to get a very large margin by just making use of the first few components and basically ignoring the rest. As a result, our classifier performs poorly because the first two or three principal components are not sufficient to discriminate age or gender. We think a better scaling technique might have been able to remedy this problem.

SMLR

We also tried using Sparse Multinomial Logistic Regression (SMLR), created by David Weiss, as an alternate discriminative method. We were given only the training function for SMLR; thence, had to implement the prediction function ourselves. We did this by multiplying X (an example by attribute matrix) by W (an attribute by output class matrix of learned weights). In the resulting matrix we could have used the values in each row to compute the probability of each example falling into each of the classes, but since we were only interested in the determining which class is the most likely, we did not do any further computation; and just took the maximum value per row. The column this value lies in is the most likely class; thence, the one we predict. SMLR would probably have run faster than the SVM but because we could not get the *C code* to compile, SMLR had to use the Matlab version which took a long time to run.

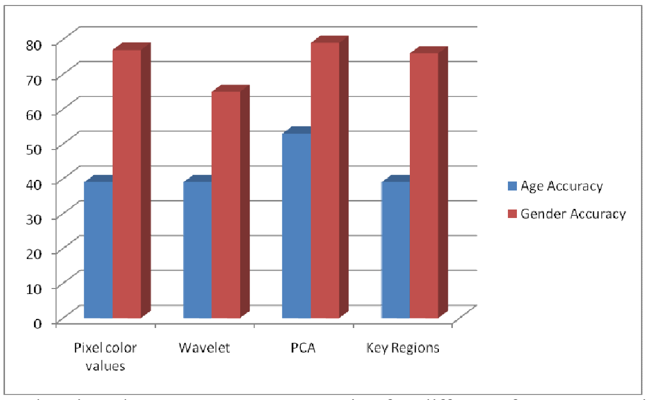

Figure 3: Plot showing the percentage accuracies for different features used with SMLR

Meta-Classifier

We also attempted to improve upon our accuracies by constructing a meta-classifier which took the predictions of several other classifiers as inputs to output an optimal classification. We tried two different kinds of algorithms for the meta-classifier, Naive Bayes and weighted voting. The advantage of Naive Bayes was that it could learn different weights for each output of each classifier. If we have classifiers that make different kinds of mistakes, we could learn to ignore a classifier when it outputs a prediction that tends to be wrong. However, in order to train the meta-classifier well, we had to split our training data into a portion for training the regular classifiers and then development section for training the meta-classifier that the regular classifiers had not seen yet. Since we only had 488 images to start with and we split off 20% for testing and then another 20% off the remaining 80% training set, the data was getting quite thin. The outcome of the Naive Bayes meta-classifier was scattered and we think this is a result of insufficient training data. To overcome this we tried a less complicated model, weighted voting, that we thought could train using less data. This meta-classifier simply looks at the accuracy of the classifiers on the development data and gives each classifier a proportional weight in the vote. This method worked reasonably well as it performed only slightly better than the base classifiers and also took much longer to train. Perhaps with more classifiers it may have done better, as we only ran it with one more classifier than there were output classes (for gender we used 3 classifiers and with age we used 4).

BLOGS

1. *Stemming:* We initially thought that pruning the dictionary down to fairly distinct words would give us a better accuracy. This led us to consider using Porter Stemming[4] on the dictionary to combine words that belonged to a family of words. For example, consider the word *fished*; it belongs to the same family or class of words as *fish, fishing,* and *fishier.* The stemming process would take all words these words and group them together as the root word *fish.* Carrying out the stemming process reduced our dictionary from 83,000 entries to approximately 57,000 entries. However, our assumption that using this new smaller dictionary would lead towards a higher accuracy turned out to be false. We believe that our accuracy was low because as we combined the words with common stems, we also increased the weight attributed to the all the misspellings and random punctuation that was present in some of the blogs thus making it difficult for our classifier to accurately predict.
2. *Information Gain:* Even though our assumption that shrinking the dictionary would increase accuracy was shown not to hold when using stemming, we continued to believe that certain select words from the dictionary were more useful than others. After considering using decision trees with a fixed height, we decided to just use the information gain calculations used to build the trees to pick out a small group of useful words. To decide how many words we should keep, we did cross validation to choose the best threshold. We discovered that on our training/development setup the cross validation always choose between 1,000 and 10,000 words that were the most useful. This system was the core of how we beat the baselines for blogs.
3. *Nearest Neighbor:* For our instance method we wrote up a nearest neighbor algorithm that compares blogs in the test set to those already seen based on the amount of words they have in common. We then predict the test blog as being the same class as the blog in the training set that was most similar to it. This

[4] M.F. Porter, An algorithm for suffix stripping, *Program*, 14(3) 1980 pp 130–137.

approach gave us worst results than just picking a class randomly so we end up not using it.

4. *Histograms:* We tried computing some different features other than word count or word occurrence. We thought that a histogram of word lengths in a blog and a histogram of sentence lengths in a blog might prove useful. The intuition was that older people use longer words and sentences than younger people. We tried using these features with a SVM and it still rarely predicted old people. We think that while young people have shorter words and sentences, the middle aged and older group have similar word and sentence length histograms.
5. *Latent Dirichlet Allocation*[5]*:* *LDA* was suggested as possibly being useful. The output from this algorithm is a set of words that describes a set of blogs. The words we got as output consisted of short words such as *the, a, and,* etc. Intuitively we dismissed the effectiveness of using these words as they did not seem capable of differentiating between either age or gender groups.
6. *String kernel:* We used this kernel with SVM for the blogs. It initially seemed like a viable option; however, after spending some time trying to understand how the authors determined the run time of the algorithm they presented in their paper, and figured out we were not ready to pay the computational cost associated with implementing it.
7. Being a *naïve group* (recall our group name), we had no choice but to attempt the popular *Naïve Bayes (NB*) algorithm. We discovered the learner predicted old people very rarely, if at all. Determining what features to feed the learner to enable it identify blogs in the oldest age class was challenging. After skimming through some of the blogs managed by older people, we picked up upon a trend in most of them. The ones we perused did not have overarching topics or sets of words differentiating them from the blogs managed by the teenagers. For instance, the blogs managed by teenagers had common words like *lol, omg, school, homework,* etc whereas the blogs from the older group had no such group of words. We did, however, find that blogs in the oldest age range tended to have well structured, grammatically correct, well punctuated and fairly longer sentences. These patterns do not lend themselves to being encoded into features that a classifier could use to make its predictions hence trying to find a concrete link between the blogs managed by people in the oldest age range remained a core task which was never adequately completed.
8. A number of papers we perused on text classification usually used the *Reuter Data Set,* which we assume is a fairly clean, well structured text data set, relative to our data set. We think it will be interesting to see how tools and approaches developed for classification of the *Reuter Data Set* translate to unstructured data sets like ours.

SUMMARY

From this project we learned quite a bit about using machine learning in practice by applying theory we learned from class and finding when those ideas potentially break down. Finding good features that discriminate the classes well and are not too noisy turned out to be one of the largest challenges. Additionally, we found using simpler methods tended to work better than the more complicated approaches that we tried.

[5] D. Blei, A. Ng, and M. Jordan. Latent Dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, January 2003.