CIS520 Project Report for A Channeling Mire

Sean O'Hara, Efstathios Kanterakis, Carl Mackey

1 Blogs

For blogs, we used a bag-of-words approach. Each blog was viewed as a sparse vector of the counts of uses of each word. We additionally used the porter stemming algorithm to produce smaller vectors that would (hopefully) better reflect the concepts discussed in the blogs. From the original 89,000 or so entries, the stemmed dictionary had only 55,000. However, there may be words in the test set that share stems with the training set, that we are blind to.

We eventually went with simple SVM on the stemmed word bags for the blog posts, producing as much as 6% improvement in age over the baseline in crossvalidation, but not a significant improvement in gender (about 2%). As seen in the final test results, it actually performed worse on gender.

Our final test results were 72.2% for age and 72.2% for gender.

1.1 LSI + SVM

We experimented with using latent semantic indexing (LSI) for dimensionality reduction (We didn't realize until later that we needed to subtract the empirical means... so maybe that contributed to the poor performance). We also tried various local and global weighting functions (as Wikipedia suggested, log+entropy seemed to do well). The entropy weighting algorithm required a bit of effort to work in Matlab, since it would typically require an intermediate matrix whose dimensions were too large for Matlab, and couldn't be made sparse.

To deal with excessive size of the transformation matrix in SVD (since we must store an m by k matrix), we deleted near-zero values, keeping only the 10,000 highest values (rather than m) for each of the k vectors, and storing it in a sparse matrix. It was noted that higher numbers of non-zero values improved results, though it could easily pass the 50 megabyte limit. With higher k, the SVD took longer to compute, which was inconvenient.

In crossvalidation, the best we could get was 68.5% for age and 65.4% for gender, trying both stemmed, not stemmed, and combined – unfortunately faring even worse on the test set. Furthermore, using the method of making the concept matrix sparse would decrease performance, by about 2% when taking the highest 10,000, and about 4% when taking the highest 1,000. We had high hopes for this method, as intuitively this was the most promising. Perhaps due to forgetting to subtract the means, it did not do as well as desired.

1.2 k-NN and Boosting

We additionally attempted to use k-nearest neighbors and a downloaded decision-stump boosting library for the blogs, but they fared very poorly, rarely getting more than 60% in crossvalidation on age or gender using either algorithm. With k-NN, we found $k = 30$ or so to be best, with $k \leq 9$ often not be more than a few percent better than random in crossvalidation. Boosting required hundreds or thousands of iterations before becoming reasonable even in terms of training accuracy, but would tend to overfit when testing on crossvalidation. The poor performance isn't too surprising, as the decision stumps can only take into account one dimension per iteration, while there are thousands of dimensions, and the k-nearest neighbors could easily overfit.

1.3 Pronouns

In an attempt to improve gender accuracy, we looked in literature for known differences between the writing styles of men and women. One paper, from Argamon et Al, detailed how females use more pronouns than males in formal writing[1]. We identified 68 pronouns in our stemmed dictionary, and created feature vectors by counting the number of pronoun occurrences in each blog post. Using a 10-fold cross validation SVM model we were able to attain an accuracy of 55%. It was thought that certain pronouns could be used in similar counts across all genders, which could lead to a decrease in accuracy. In order to combat this, we ran a PCA to determine the five least significant pronouns: neither, anybody, yourselves, whomever, and whichever. The accuracy of a 10-fold SVM model using the remaining 63 pronouns improved to 61.2% accuracy, which was still short of the mark. While these few pronouns were able to describe a large amount of the variability between sexes, it was not enough to beat the baseline. Attempts to incorporate this into our LSI+SVM model were not successful.

2 Images

In order to build an efficient classifier for images, we first cropped them to remove some of the confounding background noise. We also simplified our computation by converting images to grayscale, which did not affect classification accuracy. Using just the face did not seem as a compromise initially, however, a cropping that included the hair was used later on in an attempt to improve gender classification. Further pre-processing of the images was carried out by using the following steps: an adaptive-histogram matching filter to normalize each image and use the full histogram spectrum, Gaussian blurring, to incorporate neighbourhood information, and a gradient image, to incorporate edge information. This seemed like a good pre-processing strategy that was a good compromise between visual information and processing speed. Training on a combination of the original dataset and a right-left flipped version of the data significatly improved our classification accuracy, most likely by removing orientation-dependent features which make the classifier susceptible to less robust features.

[1]Shlomo Argamon, Moshe Koppel, Jonathan Fine, and Anat Rachel Shimonib. Gender, Genre, and Writing Style in Formal Written Texts. (http://u.cs.biu.ac.il/∼koppel/papers/male-female-text-final.pdf)

A sparse representation of images was also attempted. The 'graycomatrix' and 'graycoprops' functions in Matlab were used to extract texture features such as neighbour contrast, correlation, energy and homogeneity; additionally, the sum of a "canny" binary edge filter was also used as a feature. This approach however was not able to classify our data well possibly because it summarized our features too much and was not able to discriminate fine distinctive characteristics such as wrinkles (for age) or hair (for gender).

Finally, a precomputed kernel was attempted in svmtrain. A tract similarity metric was used according to methods previously described[2]. However, this added much computational overhead and did not improve results much. This method was therefore not carried forward.

[2]Mark Everingham, Josef Sivic, Andrew Zisserman, Taking the bite out of automated naming of characters in TV video, Image and Vision Computing, Volume 27, Issue 5, (http://www.sciencedirect.com/science/article/B6V09-4SF302F-1/2/ee4982d1c127bcc3f073512c388d66e4)

Steps	Accuracy	Gender	Age	Comments:
Baseline	Test Set	79%	40%	
whole image	Train Set	84.33%	87%	
	CV	67.5%	48.17%	
	Slack	16	128	
Face	Train Set	78.83%	73.50%	
	CV	72%	50.67%	
	Slack	16	128	
Face Gray	Train Set	77.33%	54.67%	Grayscale results in little loss
	CV	72.67%	49.67%	
	Slack	16	16	
Face Gray sharpened	Train Set	85.67%	70.83%	Too sparse
	CV	73.5%	45%	
	Slack	16	16	
Face Gray Hist Match	Train Set	91.19%	72.75%	Doesn't do much
	CV	77.65%	46.73%	
	Slack	16	16	
Face gray hist eq	Train Set	91.19%	72.75%	Improves contrast
	CV	77.65%	46.73%	
	Slack	16	16	
Face gray hist eq blur	Train Set	84.22%	77.05%	Averaging over a neighborhood
	CV	78.27%	53.27%	
	Slack	16	128	
+flipped dataset	Train Set	99.9%	92.6%	Drop orientation-specific training
	CV	92.73%	85.15%	
	Slack	1024	8192	
+gradient image	Train Set	99.9%	91.9%	Incorporates edge features
	CV	97.75%	93.34%	
	Slack	8192	8192	

Figure 1: Image Pre-Processing Steps

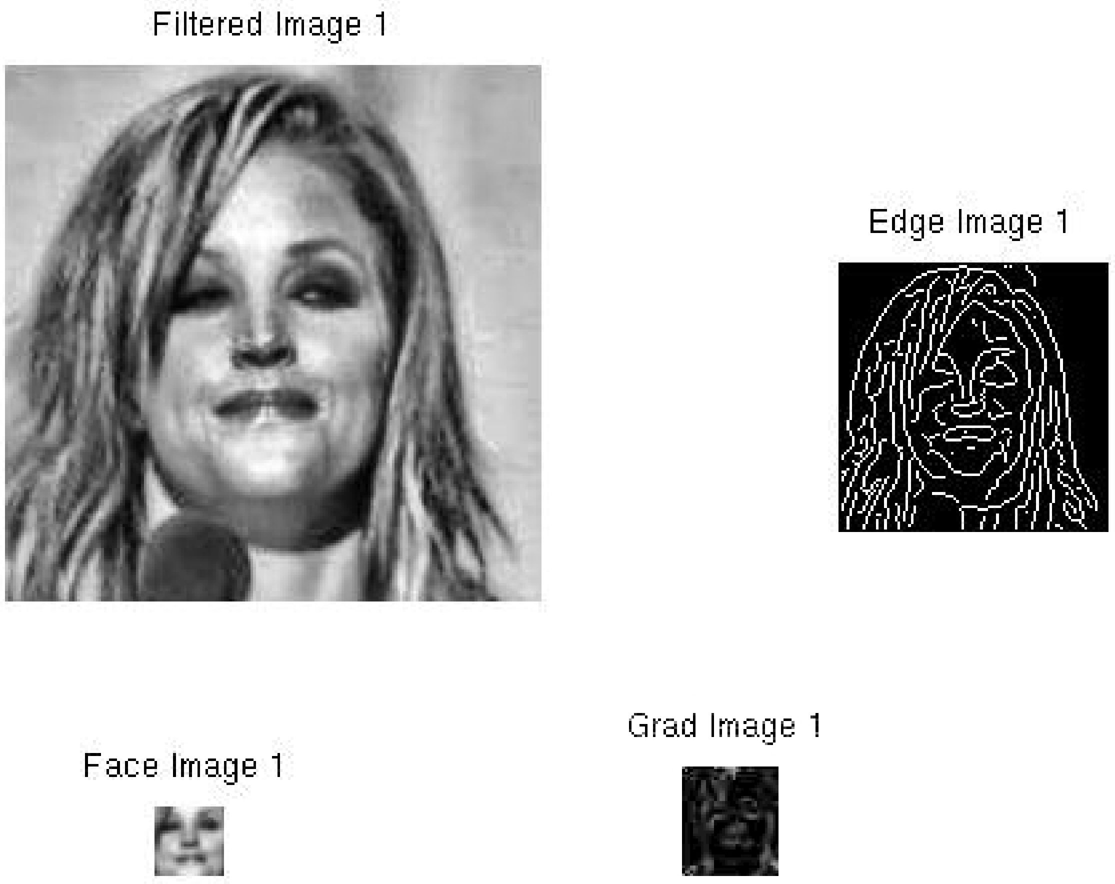

Figure 2: Image Filter Examples

3 Libraries Used

Count Unique:
`http://www.mathworks.com/matlabcentral/fileexchange/23333-determine-and-count-unique-values-of-an-array`
AdaBoost:
`http://www.mathworks.com/matlabcentral/fileexchange/21317-adaboost`