Beating the Baselines: Applying ML Algorithms to Real Data

Richard Garcia, Jason Owens[*] and Oscar Nunez[†]

December 4, 2009

1 Introduction

Learning how to induce a machine to learn is a difficult task. Working with the equations, algorithms, and theorems is necessary to develop a proficiency in the language and mechanics of the process, but there is no substitute for applying this knowledge to real data. Application of the theory in practice is paramount to work effectively with the tools. This report details our experiences while trying to beat baseline performance (in terms of accuracy) on four real-world classification tasks: classifying the gender and age group of web log (blog) excerpts extracted from the web, and similarly extracting the gender and age group of color images of celebrity faces.

In partial fulfillment of the project requirements, this paper also briefly describes our implementation of: an alternative discriminative method (neural network for images), an instance-based method (k-NN for blogs and images), and an unsupervised method (PCA and LDA for blogs and images).

2 Experiments

2.1 Blogs

The blog experiments proceeded through the exploration of papers and topics relevant to text classification. Interestingly, we found that topic classification over standard datasets (e.g. Reuters) was an overarching aspect to most of the papers we reviewed. We note that that kind of classification and content is very different from the personal, widely varying writing quality of blogs we classified in this project; thus, the results from the algorithms we tried did not tend to match those in the papers[1].

Our first efforts centered around modifying the baseline Naive Bayes (NB) and support vector machine (SVM) classifiers to use generated features and dimensionality reduction techniques, described in the next sections.

2.1.1 Features and Dimensionality Reduction

We attempted to use a variety of features in our experiments, some driven by simplicity of implementation (due to time constraints), and others driven by how well they might discriminate between classes. The various facilities we developed for this task are described below.

Spelling Checker: Spell checking over the entire dictionary was implemented by utilizing the standard `/usr/share/dict/words` file found on most Unix-like systems. It provided indicators of when a token may have been a proper noun (looking at capitalization), when it was a number (all digits), and whether it contained a swear word.

Word Stemming: We implemented word stemming using the Matlab porterStemmer() function [Lop06]. The stemmer used the results of the spell-checker and only stemmed properly spelled words; other words were either mapped to placeholders indicating their type (swear words or numbers) or passed through, but made lowercase (possible proper nouns). This process effectively reduced the size of the dictionary from 89182 tokens to around 16000 tokens.

Stop Word Removal: We implemented a small set of stop words and removed them, but generally found a reduction in performance across the board. This technique was generally not used.

Low Document Frequency Term Removal: Several papers ([JNR98, LS07]) suggested removing words with low document frequency (i.e. those words present in less than 3 documents overall). This did not seem to affect performance positively, and was also not used.

Principal Component Analysis: In an effort to reduce dimensionality, we applied PCA to the document matrix, using term frequency vectors.

Document Statistics: A set of per-document statistics were developed to help discriminate between classes: average word length, average sen-

[*]PennKey: owensj

[†]PennKey: oscf

[1]Unfortunately, there may also have been serious bugs in our various implementations, ultimately skewing the results.

tence length, capitalization-after-punctuation frequency, misspellings count, swearword count, number of hypertext links, frequency of alphanumeric repetition within words, frequency of slang words, and frequency of punctuation repetition within words. Various subsets of these statistics were used during development.

2.1.2 Other Algorithms and Techniques

An online video lecture by William Cohen at CMU 2006 Autumn School on Machine Learning over Text and Images [Coh06] discussed Naive Bayes and SVMs for use in text classification. The lecture discussed some techniques we hoped to explore (word and character-based *n*-grams), but ultimately could not find the time to implement. We did, however, explore **clustering** for feature generation using principal component analysis (PCA) and latent Dirichlet allocation (LDA), discussed in [BNJ03]. Using suggestions from the paper, we attempted to integrate the results into an SVM-based classifier. We achieved poor accuracy results with both PCA and LDA (not reported), quite possibly due to minimal class separation with the default features (words), and inadequate parameter search for the number of LDA topics[2].

The Schapire et al. BoosTexter paper [SS00] discussed his efforts using AdaBoost for text classification, specifically using a custom weak learner that classifies according to single instances of bi-grams. While we didn't have time to implement his methods, we did try boosting decision stumps and short decision trees on both our document statistics and raw word features. As an accessible **ensemble method**, we examined and implemented bagging [DHS01], a method to reduce variance by training a set of classifiers on smaller subsets of the data and then merging the individual responses into a single response (in our case, using a majority vote). Yan & Yan's paper [YY06] discussed the use of NB for blog gender classification; although we made attempts to utilize some of their methods, our results were not as favorable.

Exploring other easy-to-implement methods that might give good results, we found a paper[GZG09] describing a simple centroid-based method for classifying text using standard tf-idf[3] vectors; the results, however, were less than stellar, and far different from the results reported in the paper.

[2]The processing for LDA took quite a while using the toolkit at [Ver06].

[3]Term frequency-Inverse Document Frequency vectors [BYRN99], a common method for encoding documents for retrieval and comparison purposes.

Finally, we implemented an **instance-based** k-NN algorithm using an implementation of kd-trees found on MathWorks [Tag08].

2.2 Faces

Literature related to this project describes multiple methods for image classification including SVMs, k-NN (Figure 3a) and neural nets (Figure 2). After experimenting with all three classifiers, it was determined that the best accuracy was obtained from SVMs. We deemed the feature generation problem a larger issue than the classification method, mostly due to the difficulty in generating meaningful features that would allow for separable classes. Thus, our effort included implementing wavelets, eigenfaces (Figure 3b), Gabor images, Canny-edge images, binary mappings, multiple histogram equalization techniques and texture counting. Feature vector sizes during experimentation ranged from several hundred per image to over 600,000 per image.

While neural nets and k-NN can be robust to some amount of noise, the training set must be relatively separable in order to effectively classify the groups; in this problem we were faced with noisy images that included a variety of different lighting and pose conditions; in addition, differences between male and female individuals can be very subtle even in the eigenfaces space. On reflection, generating virtual examples may have been a useful technique here.

We found that not one of the individual feature generation techniques provided adequate and compact information. Therefore it was determined that the best course of action was to create an ensemble of the best candidates and allow them to vote on the classification. This method, which we refer to as the "Combined Voting Algorithms", utilized our top three performing algorithms and classifies an image based on majority rule. Due to the multi-class age problem any tie was simply overruled by the edge detection algorithm as it proved to have the overall highest accuracy. This method ultimately allows the classifier to exploit the strengths of the group to overcome the weakness of any individual algorithm.

2.2.1 Feature Generation Descriptions

Baseline: This is the baseline code and algorithms provided for the project.

Basic Image Algorithms: Three basic image algorithms were tested: #1 Cropped Color Face, #2 Cropped Gray Face, #3 Full Gray. These algorithms used the individual pixels as the feature set and are either the full image (#3) or the cropped face from the image (#1 & #2). The cropping technique used is simply the repaired "Baseline" code.

Contrast Adapting Algorithms: Three types of contrast adapting algorithms were tested: #1 Full Gray w/ Equalization, #2 Full Gray w/ Adaptive Equalization, #3 Full Gray w/ Contrast Spreading: These algorithms adapted/adjusted the image contrast as their name implies. Each algorithm operated on the full gray-scale image and then scaled the image to 50x50 pixels. Each individual pixel in the cropped image was then used as a feature.

Edge Detection Algorithms: Four types of edge detection algorithms were tested: #1 Gray Sectioned Edge Detection Counting, #2 Gray Sectioned Edge Detection (0.1) Counting, #3 Gray Edge Sectioned Detection Counting w/ Feature Normalization, #4 Gray Sectioned Edge Detection Counting (large sections), #5 Gray Sectioned Edge Detection Counting w/ Equalization, #6 Cropped Gray Sectioned Edge Detection Counting w/ Equalization. Algorithms #1-5 use full gray-scale images (#6 used the cropped face); algorithms #5 and #6 are histogram equalized. "Canny" edge detection was then performed to generate a binary edge-only image (threshold of 0.1 used for #2). The image was then sectioned into 25x25 pixel regions (10x10 for algorithms 4,5, and 6). Each pixel of a region is then compared with its surrounding pixels to determine if they create a line. Lines are classified as one of 8 labels: up, down, left, right, up & left, up & right, down & left and down & right. To be considered a line both the center pixel and the corresponding neighbor pixel must be an edge. For each region the types of lines are summed and used as part of the feature vector. Algorithm #3 scaled individual features to be between zero and one inclusive while the rest were scaled by simply dividing the entire set of feature vectors by the overall maximum value.

Local Gabor Binary Mapping w/Class Center Connecting Line: This algorithm, described in [XSL08], gray-scales, histogram equalizes and resizes the image to 52x52 pixels. The face is then cropped from the image using the baseline cropping mechanism. Next, a set of forty Gabor-filtered images are created from the single face images. The Gabor parameters include every combination of five scales and eight equal rotations. The local binary mapping is calculated and the image is then section into 25x25 pixel regions. Texture types are counted for each region and then used as part of the feature vector. Finally, the Center Connecting Line (CCL) method is used for dimensionality reduction. Figure 1 illustrates the framework of this method as it

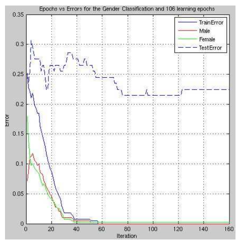


Figure 2: Learning curve for neural net cross-validation; learning terminated after 85 iterations to avoid over-fitting.

was applied in the project. Source code for generating the LBP and Gabor wavelets was downloaded from [HA09] and [Cha08] respectively and modified for use in this project.

Wavelets: First four wavelets were used to create features, yielding sub-bands of 40x40 pixels. While the wavelets gave good local features, those features did not generalize across examples which yielded a poor performance in the nearest neighbor search. Wavelets on SVMs were tried as well but did not improve accuracy significantly.

Gray-scale vs. Lab eigenfaces: In order to reduce the dimensionality of the dataset and to standardize lighting conditions across images the luminescence component of the Lab color-space was used to produce features. When compared to the gray-scale histogram-equalized counterpart the L component diminished accuracy and was therefore discarded. Both color preprocessing routines were applied in conjunction with SVM, neural networks and k-NN; the first outperformed the last two algorithms. There are two possible reasons why the gray-scale histogram-equalized images outperformed the Lab conversion: bad normalization across images and the loss of important high frequency information when omitting the a and b components.

Combined Voting Algorithms: This algorithm combined three separately trained classification routines. Each classification routine used an SVM but was trained utilizing different feature generation algorithms. We included "Local Gabor Binary Mapping w/ Class Center Connecting Line", "Gray Edge Sectioned Detection Counting w/ Feature Normalization" and "Full Gray" in our winning ensemble. Each component algorithm then individually classified the images and the final response was based on majority rule. In the case of age, when a majority was not found the output of "Gray Edge Sectioned Detection Counting w/ Feature Normalization" was utilized as it provided the highest accuracy rate in cross validation.

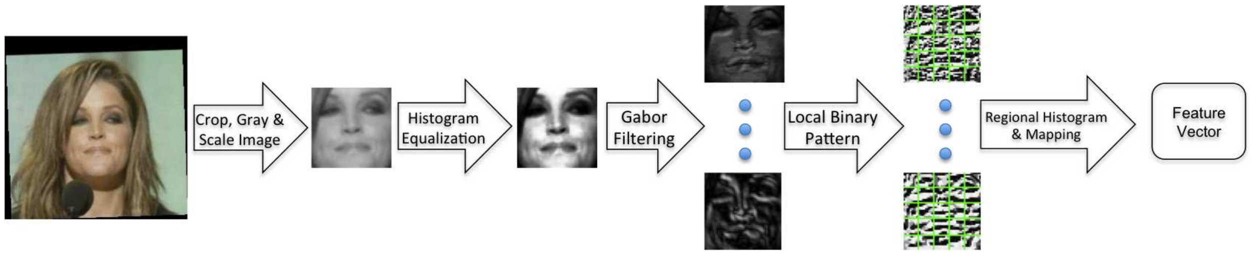
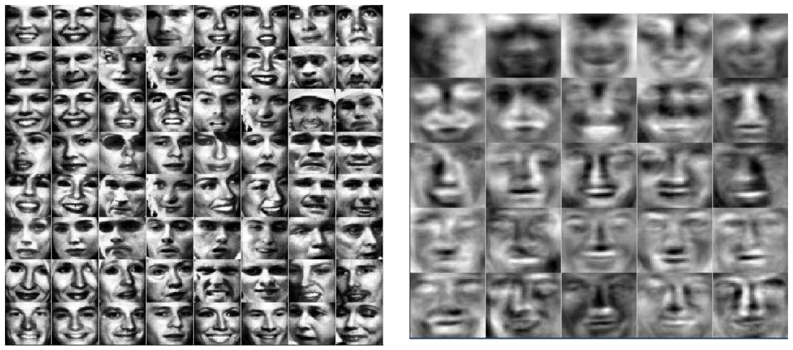


Figure 1: The framework of the utilized LGBMP feature extraction method.



(a) k-NN sample pairs; the nearest neighbor is on the left of each pair (b) First 25 eigenfaces (histogram equalized)

Figure 3: Image feature visualization

3 Results and Analysis

3.1 Blogs

Table 1 describes the representative results across our various blog experiments. Ultimately, the "tweaked" form of the original Naive Bayes formulation (1) proved most reliable in beating the baseline, and included the following modifications: to classify gender, we added the likelihood of above-mean misspellings, above-mean alphanumeric and punctuation character repetition, and above-mean presence of swear words, on top of the likelihood of the individual stemmed terms; when classifying age, we included the above components as well as the likelihood of above-mean sentence length and above-mean sentence capitalization. While the results indicate improvement over the baseline (which was run under the same ten-fold cross validation), they do not indicate the actual hold-out test accuracy, which seemed to change drastically given small modifications in our code (e.g. adding or removing a single feature). This variance led us to explore bootstrap aggregation, i.e. bagging (2), using the tweaked classifier, which showed some improvement for both age and gender, but which did *not* beat the baseline on the hold-out test set.

Next, since [Coh06] and others ([DPHS98, JNR98]) indicated that support vector machines perform quite well on text categorization, even with high feature counts, we applied them in several variations: using the reduced-dimension vectors computed from PCA and LDA, using the stemmed term count vectors (results shown in (4)), and using the non-stemmed complete term count vectors. Most results other than the one shown here were no better than random; examination of the learned SVM models showed very high support vector counts, indicative of over-fitting. We probably spent[4] the most time debugging and researching this issue: we first tried expanding the default parameter search through C values; every test yielded 100% training accuracy and low (i.e. random) test accuracy; still over-fitting. We varied the kernels, trying both the radial basis and linear kernel, as well as adding an extended grid search over both C and γ. As suggested by the LIBSVM authors in [LCH04], we also scaled all vectors to 0-1. Unfortunately, the best performance we eked out is shown in Table 1.

We then investigated boosting (7); again we were stymied: the version of AdaBoost [Vez09] we adapted failed to make progress on one fold after more than 24 hours and was terminated. The most likely cause of slowness was simply too many attributes (stemmed

[4]Wasted? Perhaps the term is too strong, but...

Id	Final Algorithm	Age	Gender
1	Tweaked NB	**69.88**	67.44
2	Bagging (Tweaked NB) 75%, 3 bags	69.71	**67.71**
3	Baseline	68.59	67.59
4	SVM (stemmed term count vectors)	61.76	58.41
5	Class Feature Centroid	58.53	54.18
6	5-NN	47.06	50.59
7	AdaBoost (with decision stumps)	DNF	DNF

Table 1: 10-fold cross validation accuracy across selected algorithms

word terms) for the decision-stump weak learner. Further investigation, given more time, would include alternate learners (e.g. those described in [SS00]) as well as effective application of PCA or LDA.

One of the most surprising results was the poor performance of the nearest neighbor algorithm (6), which, on reflection, probably indicates an implementation bug. Some techniques that may have improved performance include weighting the results from each neighbor according to their distance, learning weights for attributes (so irrelevant attributes don't contribute too much), and probably most effective: a different feature representation.

3.2 Faces

While facial image classification is a widely researched and well-published area, it appears that many of the published techniques are specifically tuned to very narrow tasks or types of input data (e.g. assumption of straight face alignment, or the use of controlled lighting and background color). This became apparent over the course of the project as techniques such as eigenfaces, Gabor filtering and edge detection proved to result in less than 80% accuracy when the published papers report accuracies around 95%. Further evaluation showed that many of these algorithms pruned images that had distinctive facial expressions, non-standard backgrounds, or partially rotated orientations.

The data provided in Table 2 shows the training and testing accuracies given the various algorithms. Note that the data marked with an asterisk (*) was obtained from a single fold cross validation run using 80% of the data for testing and 20% for training. This algorithm took several hours to run and was omitted from the 5-fold cross validation routine. It's worthwhile to observe that the results are all within a very small range of each other and some variation may be attributed to noise. It should also be noted that several of the algorithms generated results with large variance over the folds and were generally considered unstable.

4 Conclusion

This project impressed on us the real life difficulties (and joys) of applying machine learning techniques. Each component: feature selection, information compression/generalization, limitations of hardware (speed/memory), data selection/pruning and testing/validation techniques has an important role and can make the difference in whether an algorithm is effective and informative or simply wastes CPU cycles. Despite the challenges of implementing the classifiers, we did find that machine learning can work (with practice) and produce real, usable results.

References

[BNJ03] David M. Blei, Andrew Y. Ng, and Michael I. Jordan, *Latent dirichlet allocation*, Journal of Machine Learning Research **3** (2003), 993–1022.

[BYRN99] R. Baeza-Yates and B. Ribeiro-Neto, *Modern information retrieval*, vol. 96, Addison Wesley, New York. ACM Multimedia, 1999.

[Cha08] Zhi Chai, *2d gabor wavelets*, July 2008, www.mathworks.com/matlabcentral /fileexchange/authors/31372.

[Coh06] William Cohen, *Text classification: An advanced tutorial*, Video Lecture, Sep 2006, Machine Learning over.

[DHS01] R.O. Duda, P.E. Hart, and D.G. Stork, *Pattern classification, 2nd Edition*, Wiley Interscience, New York, 2001.

[DPHS98] S. Dumais, J. Platt, D. Heckerman, and M. Sahami, *Inductive learning algorithms*

Method	Train (Age)	Test (Age)	Train (Gender)	Test (Gender)
Baseline	0.9964	0.5347	0.9979	0.7224
Cropped Color Face	0.7810	0.4980	0.9595	0.7694
Cropped Gray Face	0.7672	0.4816	0.9144	0.7510
Full Gray	0.9544	0.5041	0.9805	0.7449
Full Gray w/ Equalization	0.9005	0.4918	0.9605	0.7429
Full Gray w/ Adaptive Equalization	0.8226	0.4633	0.9000	0.7469
Gray w/ Contrast Spreading	0.8585	0.5041	0.9344	0.7429
Gray Sectioned Edge Detection Counting w/ Equalization	0.9942	0.5154	0.9981	0.7772
Cropped Gray Sectioned Edge Detection Counting w/ Equalization	0.9750	0.5026	0.9942	0.7564
Gray Sectioned Edge Detection Counting	0.9904	0.5282	0.9923	0.8051
Gray Sectioned Edge Detection (0.1) Counting	0.9878	0.5026	0.9904	0.8026
Gray Edge Sectioned Detection Counting w/ Feature Normalization	0.9705	0.5590	0.9923	0.8154
Gray Sectioned Edge Detection Counting (large sections)	0.9426	0.5462	0.9622	0.7692
Local Gabor Binary Mapping w/ Class Center Connecting Line (LGBM-CCL)	*0.8436	*0.5102	*1	*0.7959
k-Nearest Neighbors on PCA (histogram equalized images)	0.8692	0.3985	0.6735	0.2385
Neural Nets on PCA (histogram equalized images)	0.9808	0.4643	0.9974	0.7284
Combined Voting Algorithms	0.9917	0.5436	0.9949	0.7974

Table 2: 5-fold cross validation accuracy across selected algorithms

and representations for text categorization, Proceedings of the seventh international conference on Information and knowledge management, ACM New York, NY, USA, 1998, pp. 148–155.

[GZG09] H. Guan, J. Zhou, and M. Guo, *A class-feature-centroid classifier for text categorization*, Proceedings of the 18th international conference on World wide web, ACM New York, NY, USA, 2009, pp. 201–210.

[HA09] Marko Heikkilä and Timo Ahonen, *Lbp matlab source code*, June 2009, www.ee.oulu.fi/mvg/page/lbp_matlab.

[JNR98] T. Joachims, C. Nedellec, and C. Rouveirol, *Text categorization with support vector machines: learning with many relevant*, Machine Learning: ECML-98 10th European Conference on Machine Learning, Chemnitz, Germany, Springer, 1998, pp. 137–142.

[LCH04] C.J. Lin, C. Chang, and C. Hsu, *A practical guide to support vector classification*, National Taiwan University (2004).

[Lop06] Juan Carlos Lopez, *Matlab porter stemmer function*, Web link, Sep 2006, Links hosted on http://tartarus.org/ martin/PorterStemmer/.

[LS07] J. Li and M. Sun, *Scalable term selection for text categorization*, Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, 2007, pp. 774–82.

[SS00] R.E. Schapire and Y. Singer, *Boostexter: A boosting-based system for text categorization*, Machine learning **39** (2000), no. 2, 135–168.

[Tag08] Andrea Tagliasacchi, *KD–tree for Matlab*, Matlab Central Website, Dec 2008, BSD License.

[Ver06] Jakob Verbeek, *Latent dirichlet allocation / probabilistic latent semantic analysis*, Website download, Aug 2006.

[Vez09] Alexander Vezhnevets, *Gml adaboost matlab toolbox*, Website, Nov 2009.

[XSL08] B. Xia, H. Sun, and B.L. Lu, *Multi-view Gender Classification based on Local Gabor Binary Mapping Pattern and*

Support Vector Machines, IEEE International Joint Conference on Neural Networks, 2008. IJCNN 2008.(IEEE World Congress on Computational Intelligence), 2008, pp. 3388–3395.

[YY06] Xiang Yan and Ling Yan, *Gender classification of weblog authors*, AAAI Spring Symposium Series on Computation Approaches to Analyzing Weblogs, 2006, pp. 228–230.