CIS520 Project Report, Fall-2009

Geetika Vasudeo (geetikav@seas) Inderpreet Nanda (inanda@seas)
Sibasish Acharya (sibasish@seas)

December 2, 2009

1 Introduction

The goal of this project is to engineer various age and gender prediction algorithms and compare their results. We have two types of data available for training and testing - blog postings(text) and faces (images). In our experiments, we have used some basic machine learning algorithms like Naive Bayes, Tree Augumented Naive Bayes, Logistic Regression, Support Vector Machine, Boosting, Principal Component Analysis, Latent Semantic Indexing/Singular Value Decomposition as well as few tricks like Stemming and Sanitization for blog data and Cropping and Resizing for image data.

2 Part 1 - Blogs

Blog data contains around 1700 blog entries. The dataset we use for training and testing has words from each blog extracted and tokenized. For all of these below experiments we use 80% of the given data for training and rest for testing.

2.1 Methods tried

1. **Boosting** [10]:

   Description- Boosting induces multiple classifiers in sequential trials by adaptively changing the distribution of the training set based on the performance of previously created classifiers. At the end of each trial, instance weights are adjusted to reflect the importance of each training example for the next induction trial. The objective of the adjustment is to increase the weights of mis-classified training examples. Change of instance weights causes the learner to concentrate on different training examples in different trials, thus resulting in different classifiers. Finally, the individual classifiers are combined through voting to form a composite classifiers which would perform better than any of the individual classifiers.

   We applied boosting on top of the baseline Naive Bayes (NB) classifier by adding **weights** to the occurrence count of the words for calculating the parameters of the NB classifier, and then updated the weights after each boosting iteration. We updated the weights for each successive round of boosting based on the training error of the previous round. Equations/Code used for implementing boosting on top of the baseline NB classifier:

   (a) Calculation of NB parameters using the boosting weights (D)

   for *all blogs in trainset* **do**
   Get all features for a blog entry.
   if output == female **then**
   female count = female count + 1 + D(current blog)
   else
   male count = male count + 1 + D(current blog)
   end if
   end for

   (b) Update of weights after boosting iteration was done on the basis of the training error of the previous iteration

   $$D_{t+1}(i) = \frac{D_t(i)\exp(-\alpha_t y_i h_t(x_i))}{Z_t}$$

   where Z_t is the normalization factor for distribution D_{t+1}

   Results- We implemented boosting for the blogs gender classification. Accuracy equal to the baseline was obtained. But the boosted classifier did not improve upon the accuracy of the baseline. (See consolidated statistics below for exact figures).

   Comments/Reasons- The possible explanation is that the naive Bayesian classier is quite stable with respect to small changes to training data, since these changes will not result in big changes to the estimated probabilities. Therefore boosting naive Bayesian classifier may not be able to generate multiple models with sufficient diversity. These models cannot selectively correct each other's errors during classification by voting. In such a situation, boosting cannot reduce the error of naive Bayesian classification [8].

2. **Data preprocessing (Reducing number of features)**:

   Description- The blogs were represented as sets of words indexed by a dictionary that stores all the words that occur in any of the training dataset. The dictionary contained many tokens which were garbage/did not provide any information gain in the prediction of age and gender. e.g- '!@$\tilde{)}$)', '12##aj%%', etc. To eliminate such unwanted entries, we pre-processed the dictionary and then mapped it to a new dictionary containing more relevant information.

   Some rules applied for pre-processing are as follows:

(a) If the word contains at least an alphanumeric character then
   i. For the words that start with a special character- If the number of special characters is less than 4 then prune them else declare the word as an OUTLIER.
   ii. For the words that end in special characters- If the number of special characters is less than 4 then prune them else declare the word as an OUTLIER.
   iii. For the words that contain only numerals- If the number of digits is less than 6 then group the word into the category NUMBER else declare it as an OUTLIER.
   iv. For the words that contain only alphabets and numbers- If the words are of the form (1-5)alphabets(1-5)numerals or (1-5)numerals(1-5)alphabets then categorize them as NUMCHAR else discard the word as an OUTLIER.
   v. For the words that contain substring of the form (alphabet)(special character)(alphabet)- If only one special character is present then test for the presence of an apostrophe and leave the word as it is, else discard it as an OUTLIER.

(b) For the words that contain only special characters.
   i. If it is a single special character then categorize it as- ?, ! and SPECIALCHAR for the rest.
   ii. If there are multiple special characters of the same type then map them as:
   ???? as ?
   !!!!!! as !
   . as DOTTSS
   and all the others are categorized as SPECIALCHAR.

(c) There are more intermediate preprocessing rules like the two above.

(d) Finally, all all-caps words are left as such and the mixed-case words are changed to lower case and then passed to a stemmer [9] that finds the root words and all the words mapping to the same stem are clubbed together.

We thus removed the garbage/noise patterns from the data by classifying them as OUTLIERS and then removing them completely from the dictionary. And we also grouped together similar patterns into categories like NUMCHAR, NUM, SPECIALCHAR, etc. We then created a new dictionary containing only the relevant patterns left and also created an index-mapping from the indices of the words in the base dictionary to the indices of the stemmed words/patterns/category descriptors they mapped to in the new dictionary. As a result of this pre-processing the total number of words/features was greatly reduced.

Result- As a result of pre-processing, we reduced the no. of words in the dictionary from approx. 90000 to approx 45000(50% reduction). After pre-processing the NB classifier gave better results than with the basic data. (See consolidated statistics below for exact figures).

Comments/Reasons - The performance of the classifier improved because a lot of noise was removed from the data. The predictions were based on the more significant words. Also the words having the same meaning/significance are grouped together, thereby improving their impact/weightage for the prediction.

3. **Change NB classifier to another better generative classifier**:

Description- We chose to try to implement Tree Augmented Naive Bayes (TAN) classifier [6]. But for the training of the TAN classifier, a bayes net has to be built, which requires a (no. of features) by (no. of features) loop. So we encountered time and space complexity problem, using even the reduced set of words (after pre-processing). We ran a matlab implementation of TAN with the reduced set of 45000 words as the features, but the process seemed to take absurdly long- it did not return even after about 8 hours! So as we had exhausted the data pre-processing/pruning option, this motivated us to next go for dimensionality reduction, using Singular Value Decomposition (SVD). We also linked this with Latent Semantic Indexing (LSI) [11]. LSI computes the term and document vector spaces by transforming the single term-frequency matrix, A, into three other matrices a term-concept vector matrix, T, a singular values matrix, S, and a concept-document vector matrix, D, which satisfy the following relations:

$$\begin{aligned} A &= TSD^T \\ T^TT &= D^TD \end{aligned}$$

$$
\begin{aligned}
&= I_r \\
TT^T &= I_m \\
DD^T &= I_n
\end{aligned}
$$

where $S_{1,1} \geq S_{2,2} \geq ... \geq S_{r,r} > 0$ *and* $S_{i,j} = 0$ *when* $i \neq j$. We formed the term-document matrix A using the words after pre-processing as the terms and the blogs as the documents. We then obtained the SVD decomposition of the term-document matrix. We then chose the most important words by taking only the top K rows of the concept-term matrix T^T and then choosing only those words which had the N highest variance figures for the top K most important concepts (the top K rows of T^T). We choose parameters like K = 200, N = 2000. After this dimensionality reduction, we reduced the total no. of words to just around 15000 (starting from a total word count of 90000)- a reduction of more than 80%. We then again created a new dictionary containing only the relevant 15000 words left and also created an index-mapping from the indices of the words in dictionary obtained after Pre-Processing, to the indices of the words left in the new dictionary. This mapping would then be used along with the mapping defined in the previous step to get the overall mapping from the base dictionary of 90000 words to the dictionary after LSI/SVD containing 15000 words. Out next goal was to apply TAN using this reduced set of words/features. But our time ran out and we had to stop after using SVD.

Result- After reducing the no. of words to just 15000 words, the NB classifier still gave comparable performance to the original version operating on 90000 words. The classifier after SVD did better than the baseline in the tests, for gender prediction. But it did not beat the age baseline in the tests, though it gave comparable performance to it. That is, despite using only 15000(20%) words of the 90000 words used by the base classifier, the LSI / SVD classifier performed better than the baseline for gender prediction and gave comparable performance for age prediction. This indicated that the LSI / SVD procedure was proceeding on the correct lines; some more tweaking would have most probably improved its performance beyond the baseline for both age and gender. We were unable to complete the implementation of TAN classifier on top the reduced data set.

Comments / Reasons - The reason why the classifier's performance improved in one part(gender) and was maintained in the other(gender) despite a great reduction in the no. of words is that- The SVD operation, with the LSI modification of reducing the rank of the singular matrix, has the effect of preserving the most important semantic information in the text while reducing noise and other undesirable artifacts of the original space of A. Therefore the words that correspond to the most important concepts that separate the data would be preserved after the SVD.

2.2 Consolidated Statistics

Methods tried	**Age**	**Gender**
Baseline (NB)	67%	69%
NB + Boosting	67%	69%
NB + Data preprocessing	**71%**	70.2%
NB + Data preprocessing + (LSI)/(SVD)	67.7%	**71.6%**

Our final classification are shown in bold

3 Part 2 - Images

Image data contains around 600 faces. For all of these below experiments we use 80% of the given data for training and rest for testing.

3.1 Methods tried

1. **Cropping and Resizing**:

   Description- Cropping refers to the removal of the outer parts of an image to improve framing and accentuate subject matter. We define a rectangular region around the face and crop the image for training and prediction. Resizing of images is the change in the pixel resolution of the images. The images are resized to a square of 25X25 for the classification purpose.

   Results- We saw more improvements in age prediction accuracy (5%) than that in gender (1%).

   Comments/Reasons- Cropping of images basically helps in reducing the noise in the data by eliminating not-so-important features of the image which barely contribute describing the image. The images are cropped around the faces to obtain only the facial features and leaving out unessential details like background, hair style etc. Also, resizing is helpful in obtaining equal number of features from every example as the pixels of image define it's features.

2. **HOG Histogram of Oriented Gradients** [4]:

   Description- HOG is a feature extraction method based on evaluating well-normalized local histograms of image gradient orientations in a dense grid. The basic idea is that local object appearance and shape can often be characterized rather well by the distribution of local intensity gradients or edge directions, even without precise knowledge of the corresponding gradient or edge positions. In practice this is implemented by dividing the image window into small cells, for each cell accumulating a local 1-D histogram of gradient directions or edge orientations over the pixels of the cell. The combined histogram entries form the representation. For better invariance to illumination, shadowing, etc., it is also useful to contrast-normalize the local responses before using them. This can be done by accumulating a measure of local histogram energy over somewhat larger spatial regions and using the results to normalize all of the cells in the block. These normalized descriptor blocks are referred to as Histogram of Oriented Gradient (HOG) descriptors. We applied HOG over the cropped and resized image examples to reduce the features.

   Result- After extracting features by HOG technique, we noticed the accuracy of the classifier, instead of improving, showed signs of decline.

   Comments / Reasons - The HOG method did not help in strengthening the classifier as it emphasizes more on pixels that lie on edges where as our dataset did not convey a lot of significant information just via the edges. We noticed a small improvement in accuracy when we did not crop/resize images. The reason for this could be, HOG somewhat reduced noise by normalization (A single pixel as a feature would add much more noise than a histogram based feature).

3. **SIFT Scale Invariant Feature Transform** [7]:

   Description- For any object in an image, there are many features which are interesting points on the object that can be extracted to provide a feature description of the object. It is important that the set of features extracted from the training image is robust to changes in image scale, noise, illumination and local geometric distortion, for performing reliable recognition. Scale Invariant Feature Transform (SIFT) transforms an image into a large collection of feature vectors, each of which is invariant to all of the mentioned changes. Key locations are defined as maxima and minima of the result of Difference of Gaussians (DOG) function applied in scale-space to a series of smoothed and resampled images. Low contrast candidate points and edge response points along an edge are discarded. Dominant orientations are assigned to localized key points. These steps ensure that the key points are more stable for matching and recognition.

   Results- The SIFT feature extraction does not help much in improving the performance of the classifier.

   Comments/Reasons- Methods without SIFT could exploit small transformation invariance from multiple images of the same face (training dataset has almost 4 similar images for the same face with minute transformation) and since SIFT basically does the same thing, it is kind of redundant.

Also, the images have already been cropped and resized, so there is not much noise which could be eliminated with this technique and hence, we could not benefit much by applying it to our classifier.

4. **Virtual example generation**:

   Description- Virtual examples are examples created from the given data by applying transforms like translation, rotation, scaling, etc. assuming that making minor changes to the images does not change its classification. Virtual examples help in increasing the total number of examples and facilitates better training of the model.

   Results- The virtualization of data improves the efficiency of the classifier but is easily prone to overfitting. It does not give a very consistent result over the test data.

   Comments/Reasons- The introduction of more examples improves the training model. Since, the number of features in case of images is pixels and thus is very large, so increasing the number of examples makes the two comparable. This leads to better prediction and hence improved classification. The only drawback is the overfitting that arises due to this method which may be overcome by applying an appropriate proportion of virtual examples in the training dataset.

5. **Logistic Regression + Support Vector Machine** [3]:

   Description- The baseline uses Support Vector Machine for classification. After we virtualize the training data, we could observe a heterogeneity in training data, e.g - number of female examples are more than number of male examples. SVM is known not to work well if the training data is heterogeneous. So, we try to combine SVM and LR [3] for training our classifier. The basic approach is, we divide the female (larger set) of examples to k chunks. Then we add up all male (smaller set) examples to create a new training set. So, in effect, we have k training sets. Applying SVM technique [5] [2], we find k hyperplanes. Now, for each example in the original dataset, we compute distances to the k hyperplanes. These k distances are nothing but our new feature vector for a single example. We run a standard LR [1] algorithm to train our classifier on the whole training data.

   Results- We used this technique on Gender classification and saw significant (6%) improvement in over baseline accuracy.

   Comments/Reasons- The basic reason of improvement could be neutralizing the heterogeneity in dataset. When choosing the number k, we take care that number of female examples in each k chunks are almost comparable to total number of male examples.

3.2 Consolidated Statistics

Methods tried	**Age**	**Gender**
Baseline (SVM)	48%	74%
SVM + Cropping + Resizing	**53.15%**	75.94%
SVM + HOG + Cropping + Resizing	45.37%	73.22%
SVM + SIFT + Cropping + Resizing	47.42%	73.8%
SVM + Virtualization + Cropping + Resizing	52.04%	76.81%
SVM+LR + Virtualization + Cropping + Resizing	50.13%	**79.56%**

Our final classification are shown in bold

References

[1] C. M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[2] C. J. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2:121–167, 1998.

[3] Y. chin Ivan Chang. Boosting svm classifiers with logistic regression, 2003.

[4] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In C. Schmid, S. Soatto, and C. Tomasi, editors, *International Conference on Computer Vision & Pattern Recognition*, volume 2, pages 886–893, INRIA Rhône-Alpes, ZIRST-655, av. de l'Europe, Montbonnot-38334, June 2005.

[5] M. A. Hearst. Support vector machines. *IEEE Intelligent Systems*, 13(4):18–28, 1998.

[6] S. Hong-bo, W. Zhi-Hai, H. Hou-Kuan, and J. Li-Ping. Text classification based on the tan model. In *TENCON '02. Proceedings. 2002 IEEE Region 10 Conference on Computers, Communications, Control and Power Engineering*, volume 1, pages 43–46 vol.1, Oct. 2002.

[7] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60, 2004.

[8] K. Ming, K. M. Ting, and Z. Zheng. Improving the performance of boosting for naive bayesian classification. In *In Proc. 3rd Pacific-Asia Conf. on Knowledge Discovery and Data Mining*, pages 296–305, 1999.

[9] M. F. Porter. An algorithm for suffix stripping. 14, 1980.

[10] R. E. Schapire. A brief introduction to boosting, 1999.

[11] Wikipedia. Latent semantic indexing.