Machine Learning Project Report

Great Wall
Authors: Lu Wang, Yue Hou

1 Project Review

For this project we are provided with both extracted texts from blog postings and face images. The goal is to develop both age and gender prediction algorithms from these two kinds of data.

2 Blogs

2.1 Overview

The blog data we have for the project has already been tokenized, more specifically, the data consist of two parts, training vectors and dictionary. Each training sample is represented as a vector, with approximately 1000 components and each component as a word id mapping to the word in the dictionary. However, the dictionary we have is quite crude. Many words are actually closely related, with rather similar meanings and could be seen as the same word. Also there are some random signs and characters with no specific meanings, such as: $\dotplus$

2.1.1 Words Steming

First of all, we consider steming the dictionary to find the dependencies between different items in the dictionary. We first run a script to stem the words and find that 15985 words to be redundant, and in total 73197 unique words. Thus we create a new dictionary, with an extra column mapping the words to its root. (See mydic2.mat, myWords).

Then we would like to determine the stop words as well as those random words in the dictionary, so that we could test the classification, to see what the difference is before and after removing stop words and random signs from the feature space. Therefore we define a simple filtering rule. We remove those words if its initial character is $'.', ',', '?'$ or $'!'$ or symbols that does not fall into the range 'a-z' or 'A-Z'. Meanwhile, we record the length of each item in the dictionary. This is separately recorded in $'words\_gender.mat'$ and 'mydic2.mat'.

2.1.2 Words Clustering

After filtering the words we reduce the dimension of the feature space. But still it is huge and sparse. Therefore we would like to cluster the words into several groups, with each group representing a same topic or holding similar property, hoping to represent the samples more easily and precisely. Thus we write a script that could cluster the words into k-clusters. The feature we use for clustering is defined as:

$$\frac{P(\omega_t)}{P(\omega_t)+P(\omega_s)}D(P(C|\omega_t)||P(C|\omega_t \vee \omega_s)) + \frac{P(\omega_s)}{P(\omega_t)+P(\omega_s)}D(P(C|\omega_s)||P(C|\omega_t \vee \omega_s))$$

where:

$$P(C|\omega_t \vee \omega_s) = \frac{P(\omega_t)}{P(\omega_t)+P(\omega_s)}P(C|\omega_t) + \frac{P(\omega_s)}{P(\omega_t)+P(\omega_s)}P(C|\omega_s)$$

and $P(\omega_t)$ is the probability of words t, $P(C|\omega_t)$ is the conditional probability of class C given word t and $D(P(C|\omega_s)||P(C|\omega_t \vee \omega_s))$ is defined as the KL divergence between the class distributions introduced by ω_t and ω_s.(For the clustering rule, see reference [1])

We initially select k+1 words randomly and each round we merge the most similar two clusters, leaving always k clusters and adding one more words for the next round. Each round we have not more than k clusters left. In order to keep a better accuracy for the classification next, we intend to group the words into 1000 clusters. The program is not so quick and when I run it on my laptop(IBM T60, 1.66GHz, 1GB Memory, Intel Duo), I get a 'out of memory' error. Also, the function kmeans() in Matlab gives a 'out of memory' error when I try to use the whole words collection as an input.

2.2 implementation

2.2.1 Naive Bayes Revised

To beat the baseline, the intuitive idea is that we run a cross validation training trying to get a better classifiers. Therefore, we run a 8 fold cross validation on the 1700 training samples. The classification results over the training set is quite good. For the gender it reaches around 99.04% and for age 95.12%. But when we run the model we train on the test set, the accuracy drop dramatically to 73.23% for gender and 65.03% for age. This result can be explained by the sparsity of the feature space and the diversity of the samples set. As we run cross validation, the more accurate we get for the training set, the more overfitting we are making our classifiers for a specific set.

Then we turn to doing some tricks on Naive Bayes. First of all, instead of using the whole words set as the feature space for classification, we only take those words that appear in the training sample into concern, multiplying the conditional probability of those words and make the decision. With this method, we also achieve really high accuracy over training set(96.12% for gender and 92.38% for age). But then this method also suffers a great drop in accuracy for the test set(67.2% for gender and 55.67% for age). This result can also be explained by the sparsity of the feature space. We think the test samples contains some words that rarely appears in the training set, therefore if we just simply multiply those conditional probability, the decision gets highly affected.

Then we consider making some changes to the conditional probabilities. The basic idea is that for a interest word, although it does not occur in the blogs of a certain class frequently, its occurrence tend to make the blogs belong to a certain class. Therefore we change the conditional probability $P(X_i|C)$ to be:

$$P(X_i|C_j) \leftarrow P(X_i|C_j)exp(\frac{P(X_i, C_j)}{\sum_j P(X_i, C_j)})$$

Also notice that we does not use those words we find dependent on the root words when we are investigating the data. To achieve this, we either set a flag for these words or set their conditional probabilities to a really small value.

After applying these changes, the classifier does not perform better as we expect. The classification accuracy for training set drops to 85.46% for gender and 78.45% for age. Although this change seems intuitive and reasonable, we believe the reason why it does not work well is that those changes we made break the rules of conditional independence and also overemphasize the role of some rare words in the classification.

2.2.2 Logistic Regression

Since we observe quite an amount of redundant words in this dataset, we then consider carrying out Logistics Regression, hoping that we can eliminate the dependencies by properly updating the weighting parameters in the expression of logistic regression for each feature component. The most intuitive way is that we use the whole word set as the input feature space for each training sample. Separately, for gender classification, we train one classifier, for the age classification, we train a classifier for each class. As for the decision combination, we pick up the classification result with the highest value. One problem for logistics regression is that since we are dealing with huge and sparse feature space, it is really hard for the classifiers to converge to its maximal value and it easily passes across the maximal value and bounces around the maximal point. Thus we set the convergence threshold to a higher value(2 for our case, and it takes about 5 minutes to converge). Also when we runs the classifier on the training set, the classification accuracy is not quite satisfying even on the training set(67.50% for gender and 57.08% for age). Then we try to use less features

as the input for logistic regression classifier. We count and draw the histogram (using 20 bins) of the length of the words set and use the distribution of the length of the words that occur in the blog sample as the feature.

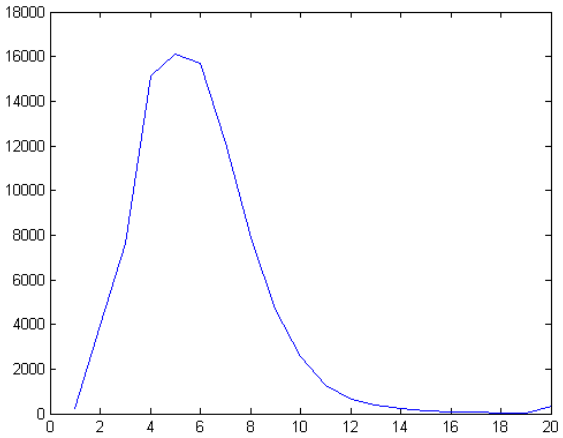

Figure 1: length distribution

In this way, the classifiers converge much faster. However, the classification result gets really bad(Often the decision boundary falls around 0.5. The reason is simple and intuitive. The distribution of the word length is not a good feature for classification in either gender or age. The other reason that since the data we have for this problem is quite crude(we have really long and meaningless terms and phrases), using just 20 bins is not enough. We search for some references and find that the average length of English words is 5.1 letters, which seems align with the histogram we draw.

2.2.3 LSA

We also work on LSA algorithms. The key of the LSA model is the term-document matrix which describes the occurrences of terms in documents. It is a sparse matrix with rows corresponding to terms and columns to documents. Using the LSA transformation, we can transform the occurrence matrix into a relation between the terms and some concepts, and a relation between those concepts and the documents. Therefore the terms and documents are indirected related through these concepts. The key of the the transformation is SVD decomposition. We try to implement the whole process of the LSA transformation. But sill we did not make to create the sparse term-document matrix in Matlab. Here we just hand in the script for the SVD decomposition.(See svdDecompose.m)

2.2.4 tf-idf based SVM

To deal with the sparse feature space, we would like to use SVM algorithm, hoping to improve the classification. Choosing the proper feature is the key for the performance of the SVM classification. We choose tf-idf as the input feature(See reference). tf-idf feature is defined as:

$$tf_{i,j} = \frac{n_{i,j}}{\sum_k nk, j}$$

and

$$idf_i = \log \frac{|D|}{|d : t_i \in d|}$$

so tf-idf:

$$(tf_i df)_{i,j} = tf_{i,j} \times idf_i$$

where

1. $|D|$: total number of documents
2. $|d : t_i \in d|$: number of documents where the term t_i appears.

We use the decision-tree-based svm classification for the multinomial classification case, age prediction. Two classifiers are trained, one for mid-age classification and one for young-old classification. First we use a classifier for the mid-age class separation, and then for the negative samples, we use the young-old classifier to determine its class.

Using svm, we got a accuracy of 74.2% on the test set for gender prediction and 69.2% for the age prediction. We still have to improve the strategy of decision combination in multinomial classification.

2.2.5 Side notes

When working on the words filtering, we found a famous theory: Zipf's Law. It is an empirical law formulated using mathematical statistics. It states that given some corpus of natural language utterances, the frequency of any word is inversely proportional to its rank in the frequency table. Therefore the most frequent words occur twice as often as the second most frequent word. We try to testify this theory. After removing stop words, we find the most frequent words are: 'the', 'and', 'that', 'for','with'. They are of similar high frequency.(0.1, compared with the occurrences frequencies of all the words in each class). Also we find some frequent words that tend to occur in the blogs of a certain gender but not the other as well as those frequent words of a certain age group.

male	female
game	love
movie	feel
music	eyes
mail	he's
news	mom
site	lol

Table 1: frequent words for gender

young	mid-age	old
dont	work	kids
gonna	job	family
haha	looking	weekend
yeah	own	old
thats	enough	debut
lol	There	christ

Table 2: frequent words for age

3 Images

3.1 Overview

In order to realize face classification for gender as well as age, we use SVM classifier and PCA method, and we design a supervised method similar to K-means which doesnt yield classification accuracy as high as the first two methods.

3.2 Implementation

3.2.1 SVM using pixel value

First, we develop SVM classifier with pixel value as feature. The procedure of images pre-process is: i. crop a 170*170 patch from original 250*250 image to remove irrelevant background. ii. use Gaussian pyramid to resize the patch to a smaller 25*25 patch. Thus the patch is small enough to be computationally efficient for vector computation in SVM training. We used both gray scale value and RGB value. RGB value vector is three times larger than gray scale value, and the resulting accuracy is relatively higher than classifier with gray scale value. With gray-scale value classifier, the average accuracy of six fold cross-validation is 79% for gender and 52% for age; while with RGB color value classifier, the average accuracy is 82% for gender and 56% for age. As test result shows, the test accuracy is $4-5\%$ lower than training accuracy.

3.2.2 SVM using histogram of 256 gray-scale pixel value

We also considered using histogram of 256 gray-scale pixel value. This is not a very distinctive feature for gender/age classification with accuracy of 60%/40%

In order to make the SVM classifier irrelevant to rotation of face in the image, we try to add virtual examples that are made with rotated and flipped images as suggested. However, the growth of training set size results in really low training speed. We enlarge the training set four times: for each image, we add rotated images(+/-20 degrees) and flipped image. The six fold cross-validation result is 88% for gender and 60% for age. Considering the recurrence of faces in training set, the accuracy for test set may be more than 5% lower. We didnt made to turn in this classifier so we didnt get test result.

3.2.3 PCA

Then we considered using PCA coefficient as feature. First we resize images to 50*50 gray-scale images as mentioned previously.

Let the training set of images be $\Gamma_1\Gamma_2...\Gamma_m$, the average face is defined by $\Psi = \frac{1}{M}\sum_{i=1}^{M}\Gamma_i$. Each face differs from the average face by $\Phi_i = \Gamma_i - \Psi$. The large set of training vectors is then subject to PCA, which seeks a set of ortho-normal vectors μ_n which best describe the distribution of the data. The k-th vector μ_k is chosen such that μ_k and λ_k is eigenvector and eigenvalue of covariance matrix:

$$C = \frac{1}{M}\sum_{i=1}^{M}\Phi_i\Phi_i^T = AA^T$$

where

$$A = [\Phi_1, \Phi_2, ...\Phi_M]$$

The matrix C is 2500 by 2500 and determining the 2500 eigenvectors and eigenvalues is computationally expensive. We need a computationally feasible method introduced by M.Turk and A.Pentland. Consider eigenvectors v_i ofAA^T such that:

$$A^TAv_i = u_iv_i$$

we have

$$AA^Tv_i = u_iAv_i$$

From this trick we can see that Av_i is the eigenvector of $C = A^TA$ .Thus we can greatly reduce the calculation.

We use PCA coefficients for SVM classifier input after regularization. The prediction accuracy for six fold cross-validation is 78% for gender and 53% for age.

We develop a similar algorithm to unsupervised K-means algorithm. The difference is we use the known label of training set to make the algorithm more supervised. First we divided the data into training set and test set. For training set, we compute each classs center, then with new data from test, we predict the label of new data to be the label of nearest center, then we compute that classs center again with the new point. We use PCAs coefficient as feature vector. However, the result is not so well as SVMs. The accuracy of gender prediction is merely 55%. One reason this didnt work might be that the first 3-6 coefficients is much more greater than the others. However, the most informative coefficients might not be among them.

3.3 Side notes

Below is the picture of first ten eigenfaces weve found:

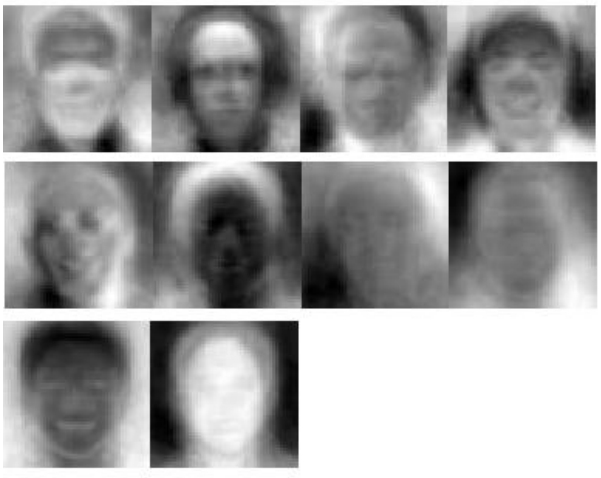

Figure 2: eigenfaces

PCA is a good method for face detection and recognition. Though it doesnt outperform SVM classifier with pixel value in this task. We use SVM classifier with pixel value trained without virtual example for final submission.

Reference

[1] Distributed Clustering of Words for Text Classification L.Douglas Baker Andrew McCallum
[2] Some Effective Techniques for Naive Bayes Text Classification Sang-Bum Kim, Kyoung-Soo Han, etal, IEEE Transactions on Knowledge and Data Engineering, Vol. 18, No. 11, Nov. 2006
[3] Text Categorization with Support Vector Machines. How to Represent Texts in Input Space? Edda Leopold Machine Learning 46,423-444,2002
[4] Eigenface for recognition M.Turk and A. Pentland, Journal of Cognitive Neuroscience 1991
[5] $http://tartarus.org/\ martin/PorterStemmer/$ for words steming scripts
[6] Face Recognition: A literature survey W. Zhao etc, ACM Computing Surveys 2003
[7] www.face-rec.org for interesting papers and code
[8] $http://en.wikipedia.org/wiki/Zipf\%27s_lawZipf'sLaw$
[9] $http://en.wikipedia.org/wiki/TF_IDFtf-idf$
[10] $http://en.wikipedia.org/wiki/Latent_semantic_analysis$