Gender Prediction by App Usage

Josh Gartman
Northeastern University
Boston, MA
gartman.j@husky.neu.edu

Abstract

The ubiquity of mobile phones has created demand for models that can accurately predict a users demographic information based on their app usage. A major challenge faced by such models is their high dimensionality which can slow down training and negatively impact their generalization ability. This work addresses this need by constructing predictive models for gender using Naive Bayes classifiers and support vector machines and considering both the apps a user has installed and the apps they actually use. Dimensionality of the models is reduced in two key ways. Firstly, by considering the general categories to which an app belongs, multiple distinct apps can be grouped together into the same category. Secondly, by considering which apps provide the most information about a users gender, irrelevant or non-informative features can be removed from the model.

1. Introduction

Consumer demographic information is a valuable commodity to businesses. Accurate consumer demographics allow organizations to target their products and advertising effectively. It is estimated that in the United States alone the market for consumer data is a $125 billion dollar a year industry[1]. A common situation is that a business may have access to information about a users buying or usage habits of a product or service but lacks direct knowledge of demographics such as gender. For example, makers of an e-commerce app can track what products a user views and what products they order but may not have actual demographic information about the user themselves. More efficient and improved methods of predicting demographics based on behavior would be extremely valuable to these organizations.

[1] http://www.forbes.com/sites/gilpress/2014/12/11/6-predictions-for-the-125-billion-big-data-analytics-market-in-2015/7092445d2b20

1.1. Related Works

There are several existing papers that examine gender prediction models based on app or Internet usage. Seneviratne et al. [4] used category, item and content based features to train a linear support vector machine classifier which predicted the gender of a user based on their installed apps with accuracy of around 70 %. Each app was grouped into 1 of 30 categories such as games or social media apps and these categories were used as the category based features. Item based features were the individual apps themselves while content based features were mined from the description text of the app on the Google Play Store. This model showed that with a training set size consisting of little more than 100 users and their installed apps, reasonable prediction accuracy can be obtained. To control the dimensionality of the model, only apps installed by more than 10% of users were considered.

Hu et al. [1] used a Naive Bayes framework on a much larger dataset consisting of the Internet browsing habits of almost 200,000 users. Each web page in their dataset was first classified using a hierarchical SVM classifier and then the demographic distribution of the users of those pages were used as features. Similar to Seneviratne et al., Hu et al. also mined the content of the pages themselves for use as content based features. Using this approach the authors were able to achieve accuracy of almost 80% on the gender prediction task.

Finally, Malmi et al. [3] used a "bag of apps" approach where each user is represented by a feature vector that indicates the presence or absence of each app for that user. While Seneviratne et al. used installed apps as features, Malmi et al. only considered apps that had been used at least once for a given user. These feature vectors were then used to train a logistic regression classifier that achieves accuracy of around 80% on the gender prediction task. Malmi et al. also explore dimensionality reduction where only a reduced number of apps are considered for the prediction model but, when the dimensionality reduction methods used by Seneviratne et al. were implemented, significant decrease in prediction accuracy was observed.

1.2. Overall Approach

The gender classification problem has many similarities with the well studied problem of text classification. Text classification methods are often based on constructing feature vectors of the words that appear in a document and then using these feature vectors to classify the document itself. Two commonly used classifiers are Naive Bayes and support vector machines. Analogously, feature vectors of apps can be constructed and these can then be used in classification models. As alluded to in the related works section, one difficulty of such methods is that the resulting feature vectors can have relatively high dimensionality if the number of apps is large. This can lead to models that are slower to train and prone to overfitting. By considering which apps give the most information about a users gender, models with reduced dimensionality but good generalization can be constructed.

2. Technical Details of Approach

2.1. Naive Bayes

A Naive Bayes classifier can be motivated by considering a feature vector $x = (x_1, ..., x_n)$ where the x_i are binary and a class conditional probability density $P(x|C_k)$. Suppose the x's are feature vectors of apps that could possibly be installed by a user and there are 10000 apps in total. In this case $x \in \{0, 1\}^{10000}$ and if x were to be modeled with a multinomial distribution over the 2^{10000} possible outcomes then a $2^{10000} - 1$ parameter vector would be required.

This problem can be greatly simplified by making the Naive Bayes assumption. Under the Naive Bayes assumption the x_i are assumed to be conditionally independent of one another so that $P(x_1, ..., x_n|C_k) = \Pi_{i=1}^{n} P(x_i|C_k)$. The resulting model has a much more reasonable number of parameters. Although the Naive Bayes assumption is rarely true, in practice it leads to classifiers that perform reasonably well on many tasks. In the case of gender classification by app usage, making the Naive Bayes assumption means assuming that usage of a particular app by some user is independent of any other app conditioned on that users gender. The priors are calculated to be the empirical probabilities of each class and Bayes rule can then be used to compute the posterior probabilities $P(C_k|x_i)$.

2.2. Bernoulli Naive Bayes

The simplest approach to constructing a Naive Bayes classifier for gender classification is Bernoulli Naive Bayes. In this approach a user is represented by a feature vector with length equal to the total number of possible apps. The feature vectors are binary so that if a user has used an app there is a 1 at the appropriate index, otherwise a 0. In the Bernoulli Naive Bayes model the class conditional densities $P(x_i|C_k)$ are computed by observing the total number of users from class k who use app i.

2.3. Multinomial Naive Bayes

One crucial drawback of the Bernoulli Naive Bayes approach is that valuable information can be lost when the number of occurrences of a feature i are ignored. In the multinomial Naive Bayes model users are again represented by feature vectors of length equal to the total number of apps but in this case the value at index i indicates the number of times that user has used app i. In this case the class conditional densities $P(x_i|C_k)$ are computed by observing the total number of times all users from class k have used app i and dividing by the total number of apps that were used including multiplicities of the same app.

2.4. Support Vector Machine

Support vector machines or SVM's are classifiers that have been shown to perform well on classification tasks with sparse matrices [2]. An SVM tries to fit a hyperplane to separate classes with as large a margin as possible. One advantage of SVM's is that their decision rule can be formulated as an inner product between feature vectors. This allows for the use of so called kernel methods that enable computation in higher dimensional feature spaces without explicit representation in those feature spaces. Typically, in text classification tasks, which have many similarities with app usage based classification, either a linear or occasionally a polynomial kernel can be used.

2.5. Dimensionality Reduction

One drawback of the previously described methods is that with a large number unique apps the feature vectors can be relatively high dimensional. The issues created by high dimensional features are two-fold. Firstly, models constructed with high dimensional features can be slower to train and to use for prediction. Secondly, higher dimensional features may be prone to overfitting. In Seneviratne et al. the authors attempted to reduce the dimensionality of the feature vectors by considering only apps that were installed by at least 10% of users [4]. This could, however, cause valuable information to be lost. An alternative approach is to calculate the mutual information between the features and the class labels. In the general case, the mutual information score between a feature X_j and a class label Y is defined as $\sum_{x_j} \sum_y p(x_j, y) log(\frac{p(x_j,y)}{p(x_j),p(y)})$. In the case of binary features it can be shown that this relation has the form:

$$I_j = \sum_c [\theta_{jc} \pi_c log(\frac{\theta_{jc}}{\theta_j}) + (1 - \theta_{jc}) \pi_c log(\frac{(1 - \theta_{jc})}{(1 - \theta_j)})] \quad (1)$$

Where π_c is the prior probability for class c, θ_{jc} is $p(x_j = 1|y = c)$ and θ_j is $p(x_j = 1)$. The mutual information score

can be thought of as the reduction in entropy of the class once the feature x_j is observed. For example, if a large percentage of women have an app j installed but few men do then feature x_j gives more information about the users gender than an app that is installed by an approximately equal percentage of men and women. If the apps are divided into categories such as games or social media apps then this definition could easily be extended to the categories as well. By considering apps or categories with high mutual information scores irrelevant or non-informative features could be removed from the model.

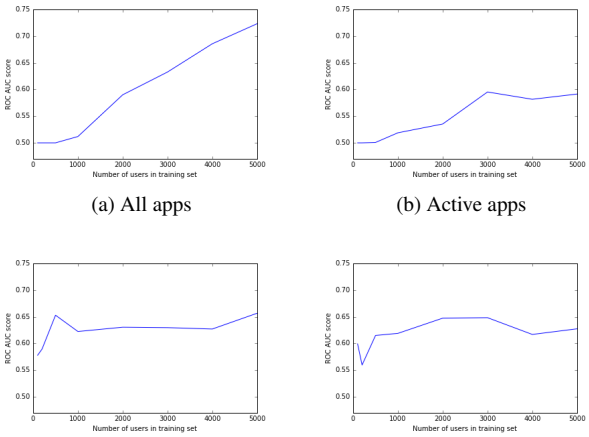
3. Experiments and Results

The studied dataset[2] was released by Talking Data, a Chinese big data platform service. The dataset consists of 23309 unique users and was collected over a week long period in May 2016. The gender breakdown of the users is 8048 women and 15261 men. For each user there is one or more associated events. An event was recorded when an app on the users mobile device attempted to use a service provided by Talking Data. When an event occurred a list of apps installed on that users phone as well as a list of apps that were active during that event were sent to Talking Data. The events may not have been driven directly by user behavior but may instead have been caused by the users device connecting to a new wi-fi network or a running app making a background network request.

There are 13762 unique installed apps of which 6942 were active for at least one user during at least one of that users events. The app names have been removed and replaced with unique id numbers but each app is associated with one or more label. The labels give information about what general category an app belongs to. For example the most popular app in the dataset is app id #8693964245073640147 which was installed by 92.8% of all users. This app is associated with the labels "Relatives" and "IM" indicating it is likely a social media messaging app. In total there are 485 unique app labels and all were associated with an app that was active for at least one event for at least one user.

3.1. Bernouilli Naive Bayes Results

Several Bernouilli Naive Bayes models were considered. Figure 1 summerizes the results of these models as the training set size is varied. Since the dataset is imbalanced between genders, ROC AUC scores are more appropriate for assessing the models effectiveness than simple misclassification rates. The first model, represented by sub-figure (a), constructs binary feature vectors using all the possible apps in the training set and counts any app that was installed on the users device. The second model, represented by sub-figure (b), is similar but considers only apps that were active during at least one app event for each user instead of just installed apps. The third model, represented by sub-figure (c) replaces each app with its corresponding labels so the feature vectors have the dimensionality of the number of unique labels instead of the number of unique apps. The fourth model, represented by sub-figure (d) is similar to the third but considers only those labels associated with an app that was active during at least one event. All four models seem to generally improve their predictive ability when the size of the training set is increased although this increase is less dramatic for the models based on labels. The best performing model by highest ROC AUC score is the first model which takes into consideration all of a users installed apps and not just those that were active. However, observe that the models constructed from labels show much better performance on small training sets than those constructed from apps. This is likely because there are far fewer unique labels than unique apps and distinct but similar apps will have the same labels so it's easier to generalize from small training sets.



(a) All apps (b) Active apps

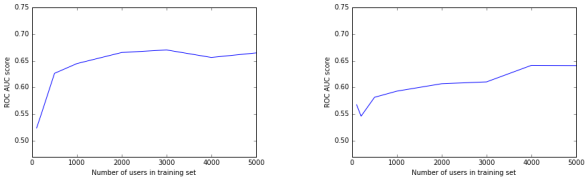
(c) All labels (d) Active labels

Figure 1: ROC AUC scores for Bernoulli Naive Bayes

3.2. Multinomial Naive Bayes Results

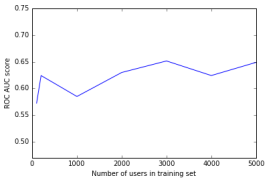
Next, four multinomial Naive Bayes models, were constructed in similar fashion to the Bernoulli models. In this case the constructed features count the number of times a given app or label was involved in an event for a given user. The results are shown in figure 2. Observe that compared with the Bernoulli app based models, the multinomial app based models appear to show better performance on small training sets. It is perhaps surprising that the multinomial models with active apps or labels did not perform better than the multinomial models which considered all apps or

[2]https://www.kaggle.com/c/talkingdata-mobile-user-demographics/data

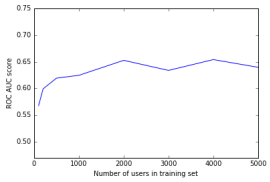


(a) All apps

(b) Active apps



(c) All labels



(d) Active labels

Figure 2: ROC AUC scores for multinomial Naive Bayes

all labels. The multinomial models with active apps or labels would seem to give the best indication of a users habits since they count the number of times a user was actively using a given app. Because any installed app would show up in all events for a given user unless it was installed part-way through the data collection period, the multinomial counts for all apps or labels installed on the users phone should be very similar. However, this is not necessarily true for the active app and label counts. One explanation why performance is not better could be that, as mentioned in the description of the dataset, the events may not be driven directly by user behavior and so the counts of apps or active apps don't actually give much insight into the users app usage habits. Another important observation is that the performance of all of these models seems to level off after the training set reaches a size of about 2000. This is as opposed to the Bernoulli models which generally seem to keep improving on larger training sets.

3.3. Dimensionality Reduction

Many apps or app labels in the data set do not give much information about the users gender. By removing these irrelevant features models can be more compact, easier to train, and, hopefully, give similar or improved generalization performance. As mentioned in section 2.5, calculating the mutual information scores between apps and classes or labels and classes indicate which features give the most information about the class a user belongs to. The results of this procedure for Bernoulli and multinomial models based on apps are summarized in figure 3. Note that all installed apps were considered. Each column is labeled by the how many of the top N apps by mutual information scores were considered so, for example, in the first column the top 100 apps by MI score were used.

Model	100	200	500	1000	2000	3000	4000	5000
Bernoulli	0.668	0.694	0.677	0.713	0.696	0.645	0.693	0.664
Multinomial	0.683	0.710	0.694	0.691	0.698	0.703	0.716	0.689

Figure 3: ROC AUC based on apps with top MI scores (training set size = 5000 users)

Model	10	20	30	50	100	200	400
Bernoulli	0.641	0.642	0.663	0.624	0.605	0.6127	0.665
Multinomial	0.654	0.662	0.646	0.659	0.681	0.672	0.639

Figure 4: ROC AUC based on labels with top MI scores (training set size = 5000 users)

It is apparent that many apps can be removed as features without greatly affecting the overall predictive ability of the model. In both the Bernoulli and multinomial NB models, results similar to those achieved with the full set of apps were achieved by using only the thousand or so apps with the highest mutual information scores. This shows that the app feature vectors can at least be reduced from dimensionality of order 10^4 to order 10^3 without sacrificing performance. The results of the same analysis for app labels is shown in figure 4.

These results show that even with only a hundred labels performance roughly similar to that using all labels or even that using all apps can be achieved. However, the app based models still seem to perform better than the label based models. This may indicate that their is a gender preference for apps within the same general app categories. For example, females may prefer one social media app while males prefer a different one. Figure 5 shows the top ten most informative labels on the entire dataset as well as the percentage of all female or male uses who had an installed app with that label.

3.4. Support Vector Machine Results

Support vector machine models were constructed using binary feature vectors. Since training and testing SVM's is much less efficient than Naive Bayes classifiers, only models based on app labels were used to control the dimensionality of the feature vectors. The variation in ROC AUC score with training sample size is shown in figure 6. Note that the polynomial kernel was of degree two.

The first important observation is the poor performance of the polynomial kernel. It should also be noted that the performance of these classifiers does not seem to be greatly improved by increasing the training set size. Overall the performance of the SVM classifiers with linear kernels seems to be similar to the Naive Bayes classifiers on the same sized training sets. For example, the Bernoulli Naive Bayes classifier with all app labels on a training set of size 1000 users had a ROC AUC score of .630 and a linear kernel SVM on the same sized training set had a ROC AUC

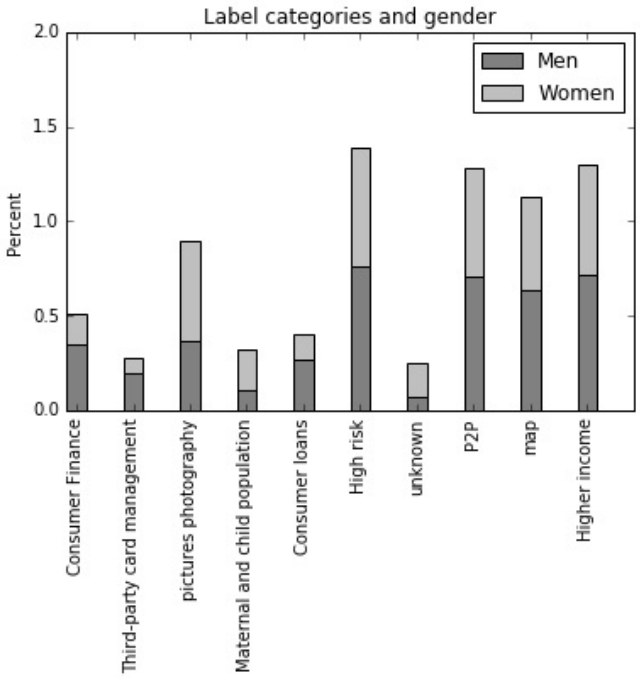
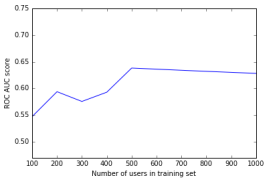
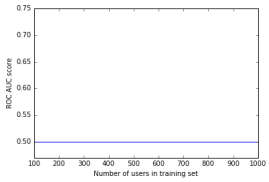


Figure 5: Most informative labels



(a) Linear kernel, all labels



(b) Polynomial kernel, all labels

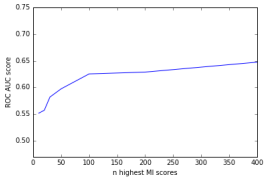
Figure 6: ROC AUC scores SVM's

score of .628.

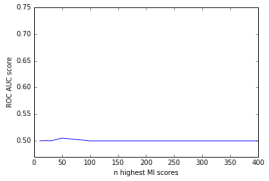
Next, dimensionality reduction using mutual information scores was studied. The results are summarized in figure 7. The poor performance of the polynomial kernel is again apparent. Performance of the linear kernel SVM approximately levels off after the 100 most informative labels are included in the model which mirrors the results found for Naive Bayes classifiers.

3.5. Soft Margin

The SVM's discussed above all used the value of 1.0 for the regularization parameter C. When the value of C is varied the amount which the model penalizes misclassified points changes as well. Figure 8 shows the ROC AUC scores for models with training sets of size 1000 using all app labels while varying the value of the regularization parameter. It is apparent that the model performs best when this parameter is around 1.0 and from there, as C gets larger model performance generally decreases.



(a) Linear kernel



(b) Polynomial kernel

Figure 7: ROC AUC scores for SVM's (training set size = 1000 users)

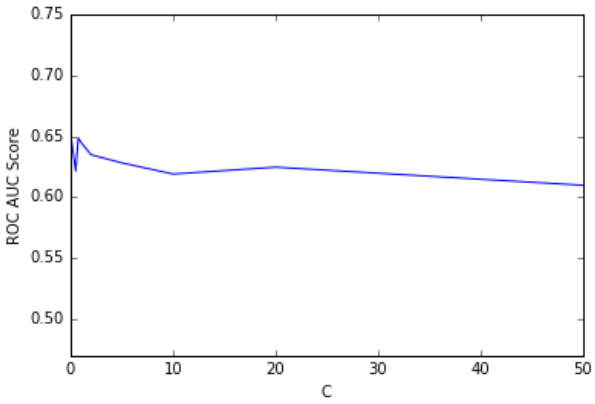


Figure 8: ROC AUC scores and regularization parameter (training set size = 1000 users)

4. Conclusions and Future Work

Models based on two classification techniques, Naive Bayes and support vector machines, were studied and differences in their performance were observed. By predictive ability, the best performing of all Naive Bayes models was the Bernoulli Naive Bayes model that only considered whether a user had installed a given app on their phone and ignored whether that app was in use or how many times it was used. The ROC AUC score of this model was .722 on a training set of size 5000 which was the highest ROC AUC score of any model considered. However, label based models with much lower dimensionality performed well and models that considered only the 100 most informative app labels were able to achieve ROC AUC scores close to .700. SVM models were slower to train but those that used a linear kernel showed performance similar to the Naive Bayes classifiers. The results of dimensionality reduction on SVM based models was similar to that of the Naive Bayes based models.

This work could be extended in many ways. The original dataset contains much information that the models do not take into account. For example, each event is time stamped and has location information associated with it. This information could perhaps be used to engineer new features. Other classification techniques could also be explored as well. For example, neural networks have shown good performance on classification tasks for models with a large number of features.

References

[1] J. Hu, H.-J. Zeng, H. Li, C. Niu, and Z. Chen. Demographic prediction based on user's browsing behavior. In *Proceedings of the 16th international conference on World Wide Web*, pages 151–160. ACM, 2007.

[2] T. Joachims. Text categorization with suport vector machines: Learning with many relevant features. In *Proceedings of the 10th European Conference on Machine Learning*, ECML '98, pages 137–142, London, UK, UK, 1998. Springer-Verlag.

[3] E. Malmi and I. Weber. You are what apps you use: Demographic prediction based on user's apps. *CoRR*, abs/1603.00059, 2016.

[4] S. Seneviratne, A. Seneviratne, P. Mohapatra, and A. Mahanti. Your installed apps reveal your gender and more! In *Proceedings of the ACM MobiCom Workshop on Security and Privacy in Mobile Environments*, SPME '14, pages 1–6, New York, NY, USA, 2014. ACM.