Training Sparse Polynomial Expansion on Graph Features Fast in the R Package Recommendation System Contest

Ming-Hen Tsai SCAN33SCAN33@GMAIL.COM
Alumni of Computer Science Department, National Taiwan University
Taipei 106, Taiwan

Abstract

The document introduces some experiments on the *R Package Recommendation Engine* competition. The learning scheme is divided into two phases : pre-processing and learning. In pre-processing, we have found explicit polynomial expansion with global graph information is useful. In the learning part, L2 regularized logistic regression is not only accurate but also solved very fast by using dual coordinate descent. To boost power of learners, a random space method with a weighted voting scheme is used. Also, a modified version of adaboost is proposed. The methods are fast, accurate and easy-to-implement with existing software available. Thus, we ranks the third in the competition in merely one week.

1. Introduction

In Kaggle's *R Package Recommendation Engine* [1] competition in 2010, competitors were given a package set, $P = \{p_1, p_2, \ldots, p_n\}$, and a user set $U = \{u_1, u_2, \ldots, u_m\}$. Each package p_i was associated with a feature vector, $\boldsymbol{q}_i \in \boldsymbol{R}^{13\times 1}$, but we had no explicit features for users. Competitors were given a training set, a subset of $P \times U$. For each (p, u) in the set, a label $y_{(p,u)}$ is given to indicate if user u installs package p. The objective was to learn a rank function $f : P \times U \to \boldsymbol{R}$, where for each $(p, u) \in P \times U$, $f(p, u)$ means the odds that user u installs package p. The evaluation criteria was AUC (Area Under Curve) of f on a given test set.

Previous R package recommendation system used logit to fit the data and has AUC about 0.94. Likewise, we used L2 regularized logistic regression to learn a model. In addition, we exploited various feature extraction techniques. Also, we used some meta-learner to boost the original model. We finally achieved AUC 0.981242 and ranked in the third place. The competition winner has AUC 0.988157.

All techniques mentioned above have existing tools to realize them. They are LIBGUNDAM[2] for data preprocessing, LIBLINEAR[3] for learning logistic model, LIBLINEAR with instance weight [4] for training weighted logistic model and Hi-Ensemble[5] for boosting the power of existing models.

1. The competition website is at `http://www.kaggle.com/R`
2. Software available at `http://www.csie.ntu.edu.tw/~b95028/software/lib-gundam/index.php`
3. Software available at `www.csie.ntu.edu.tw/~cjlin/liblinear/`
4. Software available at `http://ntu.csie.org/~cjlin/libsvmtools/#weights_for_data_instances`
5. Software available at `http://www.csie.ntu.edu.tw/~b95028/software/hi-ensemble/index.php`

The paper is organized as follows, Section 2 describes data preprocessing techniques to add features that improve AUC significantly. Section 3 introduces L2 logistic regression and variants of random subspace methods and adaboost. Section 4 presents the experiment results by the methods above by cross validation on training data and reports results on testing data. Section 5 concludes the work, and point out some experiments we are going to complete. We hope one day to really build an excellent R package recommendation system.

2. Data Preprocessing

The data is given in text and pretty noisy. We noticed that and thought good data preprocessing would fully utilize the power of classifiers afterwards.

2.1. Data Representation

The training data is given in a data matrix X_{raw} with each row represented as $\{p, u, y_{(p,u)}, \boldsymbol{q}_p^T\}$. We move the third column in X_{raw} to a vector $\boldsymbol{y}$, leaving other fields unmoved, yielding a training set X without labels and a label vector $\boldsymbol{y}$.

2.2. Denoising

After basic profiling, we found the package names and package maintainer's names and e-mails are the main source of noise. In the string representations, lower case letters, upper case letter and special characters are mixed together. Furthermore, many of the package maintainers name and e-mail are missing. Thus, we removed all special characters from the data and make all letter lower case. Also, we decoupled the field "maintainer's name and e-mail address" into two separate dimensions meaning "maintainer's name" and "maintainer's e-mail".

By doing this, the number of unique user names turned from 2487 into 2486. The number of unique maintainer's names and e-mail address turned from 1529 into 1375 in maintainer's name and 1427 in maintainer's e-mail address.

2.3. Numerical Form

To be learned by most learning algorithms, we transformed the data into numerical form. This is simply done by assigning each unique nominal attribute in the data an unique number arbitrarily.

2.4. Generating Features from Individual Feature Vectors

Categorical feature expansion is a method to make the training data meaningful especially for linear classifiers. It converts a categorical variable to several indicator variables. The framework for categorical feature expansion is shown in Algorithm 1.

Furthermore, we do explicit polynomial expansion on the data to utilize information in mutual combination of features. For each row $\boldsymbol{x} = \{x_1, x_2, \ldots, x_n\}$ in the training data X, we expand it as

$$\{x_i x_j | j \in \{i, 1, 2, \ldots, n\}, i \in \{0, 1, 2, \ldots, n\}\}, \text{where } x_0 \text{ is defined as zero.}$$

Algorithm 1 A framework of categorical feature expansion

1. Given training data , $X \in \boldsymbol{R}^{l \times n}$
2. For a unique variable v in a given categorical feature dimension j,
   build a one-to-one function $f : \boldsymbol{N} \times \boldsymbol{R} \to \boldsymbol{N}$, where column $f(j, v)$ contains no value.
3. For each row i and categorical feature dimension j
   a) Set $X_{if(j,X_{ij})} = 1$
   b) Set X_{ij} as zero.
4. Output X.

It is suitable in most case on categorical data. Although the number of non-zeros may not grow fast after expansion, the actual dimension may grow very large, making it unsuitable to be trained by many linear classifiers. Chang et al. (2010) proposed to use hashmap to condense the feature dimension online when doing polynomial expansion.

2.5. Generating Features from Graph Information

As stated above, no statistics about users are present in the data set. Thus, we tried to add features to each user by averaging the statistics of the packages he owns. That is, for user u, we add a feature $\boldsymbol{v}_u$ to the user by setting $\boldsymbol{v}_u$ as

$$\frac{\sum_{p \in P \text{ and [u installed p]}} \boldsymbol{q}_p}{\sum_{p \in P \text{ and [u installed p]}} 1}.$$

According to Huang et al. (2005), collaborative filtering problems like this may be modeled by link prediction problems. Thus, we tried to add some features usually added in link prediction problem to enhance the accuracy. Due to time constraint, the only feature we tried to add is preferential attachment, that is, for each package and user pair, we add (number of users using the package) $\times$ (number of packages installed by the users) to the feature of the pair.

3. Learning Algorithms

In the contest, we used 5 cross-validation to choose the best model and do not consult the leaderboard that presents the preliminary results on 20% of the testing data. We did this because previous experiences told me that that would usually lead to overfitting a small portion of the data set.

3.1. Models

Here, we will introduce the Logistic Regression model and a parameter search method.

3.1.1. Logistic Regression

Logistic regression models the conditional probability as in (1).

$$P_{\boldsymbol{w}}(y = \pm 1 | \boldsymbol{x}) \equiv \frac{1}{1 + e^{-y \boldsymbol{w}^T \boldsymbol{x}}}, \tag{1}$$

Algorithm 2 A framework of random subspace method

1. Given labels $\boldsymbol{y} \in \boldsymbol{R}^l$, training data $, X \in \boldsymbol{R}^{l \times n}$, sample instance ratio r_l, sample subspace ratio r_n, number of iterations t, and learners $\{L_1, L_2, L_3, \ldots, L_T\}$.
2. Set $Output$ = empty array
3. For $k = 1, 2, \ldots t$
   a) $\bar{X}$ = random sample $\lceil l \times r_l \rceil$ instances and $\lceil n \times r_n \rceil$ dimensions from X.
   b) Random select a learner L from $\{L_1, L_2, \ldots, L_T\}$.
   c) Train a model, M, by $\bar{X}$ using learner L.
   d) Add (L, M) to $Output$
4. return $Output$

where $\boldsymbol{x}$ is the data, y is the class label, and $\boldsymbol{w} \in R^n$ is the weight vector. Given training data $\{\boldsymbol{x}_i, y_i\}_{i=1}^l, \boldsymbol{x}_i \in R^n, y_i = \{1, -1\}$, logistic regression minimizes the following regularized negative log-likelihood:

$$L_P(\boldsymbol{w}) = \sum_{i=1}^{l} C_i \log\left(1 + e^{-y_i \boldsymbol{w}^T \boldsymbol{x}_i}\right) + \frac{1}{2}\boldsymbol{w}^T \boldsymbol{w}, \tag{2}$$

where $C_i > 0$ are penalty parameters associate with each instance. Problem (2) is referred to as L2 logistic regression.

According to Yu et al. (2010), the form of L2 logistic regression could be fast solved by dual coordinate descent. On my computer, training the whole expanded data set took less than five second. Such results enabled us to do very exhausted parameter selection. We used the implementation in Fan et al. (2008) LIBLINEAR.

3.1.2. Parameter Search

Let us denote C_+ in (2) for all C_i, where $y_i = 1$, and denote C_- for other C_i's. Parameter search is done by exhausted search on C_+ and C_-. We search for the range $(C_+, C_+/C_-) \in \{2^{-5}, 2^{-4}, \ldots, 2^5\} \times \{2^0, 2^1, \ldots, 2^5\}$ for best parameters.

3.2. Meta Learners

Meta learners used to boost power of general learners are used in many contests and have good results. Thus, we did experiments on some of them here.

3.2.1. Random Subspace Method

The random subspace method we implemented, is the framework in Algorithm 2. The result of random subspace methods are a lot of classifiers, we conducted equal weighted mean in decision value for the final output. In the contest, we mixed linear SVM and L2 logistic regression to train a random subspace model.

Algorithm 3 A framework of AUC adaptive boost

1. Given $\boldsymbol{y} \in \boldsymbol{R}^l, X \in \boldsymbol{R}^{l \times n}$, number of iterations t and a weak learner L.
2. For $i = 1, 2, 3 \dots l$
   a) $W_{1i} = \frac{1}{l}$
3. For $k = 1, 2, 3, \dots t$
   a) Train a classifier C_k by learner L using $W_{k:}$ as instance weight
   b) $\boldsymbol{p} =$ the function value of predicting X with C_k
   c) $\epsilon = 1-$ AUC by predicting X with C_k
   d) $\alpha_k = \frac{1}{2} \log \frac{1-\epsilon}{\epsilon}$
   e) For $i = 1, 2, 3 \dots l$
      i. Let $\boldsymbol{w}_i = W_{ki} \exp(-\alpha_k \boldsymbol{y}_i \boldsymbol{p}_i)$
   f) Let $W_{(k+1):} = \frac{\boldsymbol{w}}{\|\boldsymbol{w}\|_1}$
4. For any given input $\boldsymbol{x}$, the predicted function value is $\sum_{i=1}^{i=t} \alpha_i p_i$, where p_i is the predicted function value by running C_i on $\boldsymbol{x}$.

3.2.2. Continuous Valued Adaptive Boost

Adaboost is a way to boost power of an existing weak learner. However, the initial Adaboost setting proposed by Schapire (1999) does not fit our need to improve AUC directly. Thus, a new method for fitting data to gain AUC is proposed.

Before using the boosting algorithm here, we have to make sure the label vector $\boldsymbol{y}$ has only value 1 and -1. Since the label in the data set are 1 and 0, we set 0 to -1 to form a new label vector. In the contest, we tried to boost the performance of logistic regression model by the adaboost framework in Algorithm 3.

3.3. Human Intelligence

To win the competition, we employed human intelligence by investigating data by our eyes.

3.3.1. Rule Based Learning

There was a field indicating if the package is a core one. we assigned the core packages the highest score. Furthermore, if a user never installed a package or installed all package in the training set, we would assume he keep the same behavior in the whole data. Lastly, there were some overlap between training and testing set, my system hardcoded there information and output the prediction in testing phase directly.

4. Experiment Results

Methods and its AUC on the test set are presented in Table 1. In the table, all classifiers are logistic regression.

Features	AUC	Rank
scaled numerical data	0.969921	10
bi-gram expansion	0.977998	7
bi-gram expansion + adaboost	0.978614	7
added user information + bi-gram expansion	0.981202	3
preferential attachment + previous one	0.980734	3

Table 1: Features and its corresponding AUC on the whole test set along with its final rank

5. Discussion

Here, we would like to summarize what is a good methods and potential improvement we can make on the algorithms.

5.1. Conclusion

By experiment results, using best parameters for L2 logistic regression on expanded feature set with made user information yielded comparable results. Feature expansion helped us to get a more accurate model; while L2 logistic regression is solve fast on the expanded sparse data so that we can do exhausted parameter search on it in a short period of time.

5.2. Future Direction

As we expended data to a very large scale, we did not have time to run many other classifier (e.g. decision tree, naive Bayes) under competition time constraints. We might want to run those learners on my meta-algorithms to see if they could produce better results. How to exploit link prediction features in my approach was still not clear and might need further study.

In addition, we might want to run the algorithm on other data for recommendation system to see if it can produce good results.

6. Acknowledgement

The author thanks Chun-Sung Ferng and Guo-Xun Yuan for their valuable comments. The author is also grateful to the Machine Learning Group in Computer Science Department National Taiwan University for providing a competitive environment and the newest information. The author especially thanks Chia-Hua Ho, who have helped him to write pieces of software some of which are used in the competition.

References

Yin-Wen Chang, Cho-Jui Hsieh, Kai-Wei Chang, Michael Ringgaard, and Chih-Jen Lin. Training and testing low-degree polynomial data mappings via linear SVM. *Journal of Machine Learning Research*, 11:1471–1490, 2010. URL `http://www.csie.ntu.edu.tw/~cjlin/papers/lowpoly_journal.pdf`.

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research*, 9: 1871–1874, 2008. URL `http://www.csie.ntu.edu.tw/~cjlin/papers/liblinear.pdf`.

Zan Huang, Xin Li, and Hsinchun Chen. Link prediction approach to collaborative filtering. In *Joint Conference on Digital Libraries*, 2005. URL `http://www.personal.psu.edu/faculty/h/u/huz2/Zan/papers/link.jcdl05.pdf`.

Robert E. Schapire. A brief introduction to boosting. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*, 1999. URL `http://www.cs.princeton.edu/~schapire/uncompress-papers.cgi/Schapire99c.ps`.

Hsiang-Fu Yu, Fang-Lan Huang, and Chih-Jen Lin. Dual coordinate descent methods for logistic regression and maximum entropy models. *Machine Learning*, 2010. URL `http://www.csie.ntu.edu.tw/~cjlin/papers/maxent_dual.pdf`.