Predicting Magazine Sales Using Machine Learning

Mei Ling (Zan) Chu, Freeman Fan, Yisha Peng

Abstract

In this project, we apply machine learning techniques to a real-world problem of predicting magazine sales, i.e. the number of magazine copies to be placed at newly-opened newsstand locations using past data gathered from existing stores. Given the raw data from Hearst Corporation regarding store sales, store locations, demographics and other related facts, we designed a nonlinear multi-class SVM classifier to predict the amount of magazine sales at newly-opened stores. A *C*-SVC model with radial basis function (RBF) kernel is built for analyzing this highly heterogeneous set of data. Cross-validation via grid-search is applied for parameter tuning. In view of the nature of our problem, Root Mean Square Error (RMSE) is used to measure the prediction accuracy. The theoretical framework and experimental results for this problem are discussed in our article.

1.0 Introduction

1.1 Problem statement

In this project, we attempt to solve an important problem faced by magazine publishers today, which if successful, can dramatically help publishers maximize their profit potential. We are provided with over 10GB of data by Hearst Corporation, containing 10,929,954 store sales observations with a total of 102 features, as well as other related demographics data for predicting the sales in all of the newly-opened stores. We narrow down our scope to stores located in California because we would like to focus on applying machine learning techniques to find an excellent model rather than tackling the computational difficulties arising from using the entire dataset. After this filtering process, we obtain approximately 14,000 sales records from the past 4 years. We would like to predict the sales of individual magazine issues at 810 newly-opened stores. With a high accuracy in our resulting prediction, we can estimate precisely the number of magazine copies to be placed at these stores according to the predicted demand. This information can be used in distributing magazines to stores so that both under stock (lost sales) and over stock (cost of unsold copies) can be minimized.

Sales prediction can be transformed to a multi-class classification task. In the following sections, we make use of a nonlinear multi-class support vector machine to solve our problem.

1.2 Data Description

All data is stored in 5 database tables, while the main table is the SALES table where four variables in each record of SALES are used as foreign keys to locate related records in the other tables, namely ISSUES, WHOLESALERS, STORES and ZIP+4.

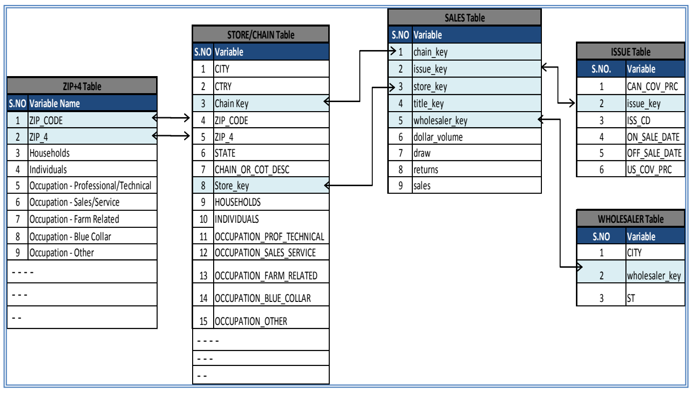

Figure 1 Relational diagram of the dataset

2.0 Methodology

2.1 C-Support Vector Classification (C-SVC)

2.1.1 Two- class *C*-Support Vector Classification

Given training vectors $\mathbf{x}_i \in R^n, i = 1,\dots,l$, in two classes, and a vector $\mathbf{y} \in R^l$ such that $y_i \in \{1,-1\}$, *C*-SVC (Boser et al., 1992; Cortes and Vapnik, 1995) solves the following primal problem:

$$\begin{aligned} \min_{\mathbf{w},b,\xi} \quad & \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{l}\xi_i \\ \text{subject to} \quad & y_i(\mathbf{w}^T\phi(\mathbf{x}_i)+b) \ge 1-\xi_i, \\ & \xi_i \ge 0, i=1,\dots,l. \end{aligned}$$

Its dual is

$$\begin{aligned} \min_{\boldsymbol{\alpha}} \quad & \frac{1}{2}\boldsymbol{\alpha}^T Q\boldsymbol{\alpha} - \mathbf{e}^T\boldsymbol{\alpha} \\ \text{subject to} \quad & \mathbf{y}^T\boldsymbol{\alpha} = 0, \\ & 0 \le \alpha_i \le C, i=1,\dots,l, \end{aligned}$$

where $\mathbf{e}$ is the vector of all ones, $C>0$ is the upper bound, Q is an l by l positive semi-definite matrix, where $Q_{ij} = y_i y_j K(\mathbf{x}_i,\mathbf{x}_j)$, and the kernel is $K(\mathbf{x}_i,\mathbf{x}_j) = \phi(\mathbf{x}_i)^T\phi(\mathbf{x}_j)$. Here training vectors $\mathbf{x}_i$ are mapped into a higher (maybe infinite) dimensional space by the function ϕ.

The decision function is

$$\text{sgn}(\sum_{i=1}^{l} y_i\alpha_i K(\mathbf{x}_i,\mathbf{x}) + b).$$

The four common kernels are:

Linear: $K(\mathbf{x}_i,\mathbf{x}_j) = \mathbf{x}_i{}^T\mathbf{x}_j.$

Polynomial: $K(\mathbf{x}_i,\mathbf{x}_j) = (\gamma\mathbf{x}_i{}^T\mathbf{x}_j + r)^d, \gamma > 0.$

Radial Basis Function (RBF):
$K(\mathbf{x}_i,\mathbf{x}_j) = \exp(-\gamma \|\mathbf{x}_i - \mathbf{x}_j\|^2), \gamma > 0.$

Sigmoid: $K(\mathbf{x}_i,\mathbf{x}_j) = \tanh(\gamma\mathbf{x}_i{}^T\mathbf{x}_j + r).$

2.1.2 Multi-class *C*-Support Vector Classification

We use the "one-against-one" approach (Knerr et al., 1990) in which $k(k-1)/2$ classifiers are constructed and each classifier trains data from two different classes. For training data from the ith and jth classes, we solve the following two-classification problem:

$$\begin{aligned} \min_{\mathbf{w}^{ij},b^{ij},\xi^{ij}} \quad & \frac{1}{2}(\mathbf{w}^{ij})^T\mathbf{w}^{ij} + C(\sum_t(\boldsymbol{\xi}^{ij})_t) \\ \text{subject to} \quad & (\mathbf{w}^{ij})^T\phi(\mathbf{x}_t) + b^{ij} \ge 1-\xi_t^{ij}, \text{ if } \mathbf{x}_t \text{ in the } i\text{th class}, \\ & (\mathbf{w}^{ij})^T\phi(\mathbf{x}_t) + b^{ij} \le -1+\xi_t^{ij}, \text{ if } \mathbf{x}_t \text{ in the } j\text{th class}, \\ & \xi_t^{ij} \ge 0. \end{aligned}$$

In classification, we use a voting strategy: each binary classification is considered to be a voting where votes can be cast for all data points $\mathbf{x}$. In the end, each point is designated to be in a class with maximum number of votes.

In cases where two classes have identical votes, though it may not be a good strategy, we simply select the one with the smallest index.

There are other methods for multi-class classification. Some reasons for why we choose this "one-against-one" approach and detailed comparisons are in Hsu and Lin (2002).

2.2 *Model Selection*

Four common kernels are mentioned in the previous section. Therefore, we must decide which one to try first. The penalty parameter C and kernel parameters are then chosen according to the selected kernel.

2.2.1 RBF Kernel

In general, the RBF kernel is a reasonable first choice. This kernel nonlinearly maps samples into a higher dimensional space so that, unlike the linear kernel, it can handle the case when the relation between class labels and attributes is nonlinear. Furthermore, the linear kernel is a special case of RBF (Keerthi and Lin, 2003) since the linear kernel with a penalty parameter $\tilde{C}$ has the same performance as the RBF kernel with some parameters (C,γ). In addition, the sigmoid kernel behaves like the RBF kernel for certain parameters (Lin and Lin, 2003).

The second reason for choosing the RBF kernel is due to the number of hyperparameters which influences the complexity of model selection. The polynomial kernel has more hyperparameters than the RBF kernel.

Finally, the RBF kernel has fewer numerical difficulties.

2.2.2 Cross-validation and Grid-search

There are two parameters for C-SVC with an RBF kernel: C and γ. It is not known beforehand which C and γ are best for a given problem; consequently some kind of model selection (parameter search) must be done. The goal is to identify good (C,γ) so that the classifier can accurately predict unknown data (i.e. testing data). We select our parameters (C,γ) using cross-validation via parallel grid-search.

For medium-sized problems, cross-validation might be the most reliable way for parameter selection. First, training data is separated into several folds. Sequentially a fold is considered as the validation set and the rest are for training. The average accuracy of predictions on the validation sets is the cross-validation accuracy.

Our implementation is as follows. First, we provide a possible interval of (C,γ) with the grid space (e.g. $C = 2^{-5}, 2^{-3}, ..., 2^{15}$, $\gamma = 2^{-15}, 2^{-13}, ..., 2^{3}$). Then, all grid points of (C, γ) are tried to see which one gives the highest cross-validation accuracy. Finally, we use the best parameters to train the whole training set and generate the final model.

The grid-search is straightforward and seemingly na ïve. In fact, there are several advanced methods which can save computational cost by, for example, approximating the cross-validation rate. However, there are two motivations why we prefer the simple grid-search approach. One is that, psychologically, we may not rest assured using methods which avoid doing an exhaustive parameter search by approximations or heuristics. The other reason is that the computational time required to find good parameters by grid-search is not much more than that required by advanced methods since there are only two parameters. Furthermore, the grid-search can be easily parallelized because each (C,γ) is independent. Many advanced methods are iterative processes, e.g. walking along a path, which can be hard to parallelize.

For easy implementation, we consider each SVM with parameters (C,γ) as an independent problem. As they are different jobs, we can easily solve them in parallel. Note that now under the same (C, γ), the "one-against-one" method is used for training multi-class data. Hence, in the final model, all $k(k-1)/2$ decision functions share the same (C,γ).

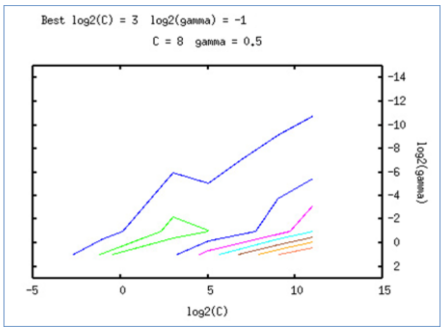

Figure 2 Contour of cross-validation accuracy via grid-search

3.0 Numerical Experiments

This section describes the main procedures and experiments we carried out in implementing our nonlinear multi-class C-SVC model and grid-search method described above using the LIBSVM package. Source codes are available at

https://afs.stanford.edu/download/?path=/afs/ir/users/f/w/fwf/public/cs229_magsales_project.zip.

3.1 Data Preprocessing

3.1.1 Data formatting and feature extraction

The original dataset is given in CSV format. We parse the dataset and import it into MySQL using a Ruby script in order to facilitate data manipulation in later stages. In order to better understand the machine learning techniques in each of the trial models, we choose to confine our scope of training and testing to the stores in California only. Observations and attributes are extracted from the MySQL database. In total, there are 14,058 training data and 810 testing data after filtering and extraction. The extracted dataset is further transformed by our Java program in order to comply with the LIBSVM input format.

3.1.2 Imputation of missing data

We tried two different ways to impute missing data: replace the missing data with average values of existing data or delete the entire features or observations. Among these two methods, deleting the features or observations with missing data gives better testing results; therefore, we set the following two rules in our deletion process:

1. If a feature contains more than 75% zero value entries, the feature is regarded as uninformative and removed.
2. If an observation contains more than 75% zero value attributes, the observation is regarded as incomplete and removed.

3.1.3 Scaling of data

The values of our features range from 10 to 10,000. To avoid domination of features with large numerical range and to reduce computational difficulties, we scale each feature to the range [0, 1] for both training and testing datasets.

3.1.4 Labeling of classes and categorization of features

This sales prediction problem can be considered as a multi-class classification problem. Due to the highly unbalanced training dataset label (e.g. approximately 15% of “0” labels, 1% of “101” label), we classify the labels into 25 classes, namely “0”, “1”, “2”,…, “20”, “25”, “35”, “45” and “101”. This varying bin-width labeling method would yield good prediction result from the originally unbalanced labeled dataset.

Furthermore, in order to use SVM, we have to transform some of the non-numerical attributes into m-category numerical attributes. For example, label “0” is used to represent “San Francisco”, “1” to represent “Palo Alto”, and so on. In addition, some of the attributes require further interpretation and manipulation in order to be effectively used in SVM. For example, the issue code entry “200911”, which means Issue #11, 2009, is split into two features “2009” and “11”.

3.2 Model selection results - Parameter tuning with cross-validation

Using 6,068 training data to carry out multiple experiments for the SVC parameters tuning, we compare the effect of parameters tuning on prediction accuracy on the 810 testing data.

	Without Parameters Tuning	**With Parameters Tuning**
C	1	32768
γ	0.01	0. 0078125
Cross val. accuracy	17.691%	46.9184%
Prediction accuracy	17.683%	36.261%
RMSE^	17.804	11.091*

Table 1 Comparison of performance on parameters tuning

* This result surpasses the first place entry on hearstchallenge.com, a past machine learning competition having the same task and same dataset as ours.
^ In sales prediction task, Root Mean Square Error (RMSE) is a more effective measure than accuracy, since it measures how close the predicted sales are compared to the true sales. This is more important than finding how many sales are predicted exactly correct, which is measured by accuracy.

The results indicate that parameter tuning is essential to making accurate predictions.

We found that parameter tuning must be performed for each training dataset, as well as for each set of features. A set of parameters that works well on one dataset cannot be assumed to work well on another dataset.

3.3 Different sizes of training datasets

We use the optimized parameters (C=32768, Gamma=0.0078125) obtained from our parameters tuning procedures for training on two datasets of different sizes.

Number of training data	6068	14058
Accuracy	36.261%	47.704%
RMSE	11.091	10.857

Table 2 Comparison of performance with different training dataset sizes

A larger training set gives us better prediction accuracy.

3.4 Using different sets of features

Using 6068 training data, 810 testing data, and optimized parameters after tuning, we compare the effect of different sets of features on prediction accuracy. We manually select 28 features which we believe might be more relevant to predicting sales (e.g. income, mean value of housing, etc.).

	28 Selected feature set	***102 Total feature set***
Accuracy	*17.2464%*	*36.261%*
RMSE	*17.124*	*11.091*

Table 3 Comparison of performance using different sets of features

The results indicate that using all the given features increases prediction accuracy.

3.5 Scaling/ Un-scaling of data

Using 6068 training data and 810 testing data, we compare the effect of scaling feature values into the range of [0, 1] on testing accuracy.

	With Scaled features	***With Un-scaled features***
Accuracy	36.261%	17.942%
RMSE	11.091	17.804

Table 4 Comparison of performance on features scaling

The result indicates that data should indeed be scaled.

4.0 Discussions and Conclusions

In some situations the procedure outlined above is not good enough, and other techniques such as feature selection may be needed. Our experience indicates that our procedure works well for data which do not have many features. If there are thousands of features, there may be a need to choose a subset of them before giving the data to the SVM. In our problem, we only have 102 features and our procedure works well. This conclusion is verified in the section of Numerical Experiments, where we compare our prediction accuracy based on 102 features with that based on 28 features.

5.0 Future Work

For this project we select a random subset from the entire data provided, perform parameter tuning on this subset, and train our model on this subset as well. To further improve accuracy, we envision using the same random subset to carry out parameter tuning; possibly better-region-only grid-search on the complete dataset, and then train our model on the entire dataset; or possibly parallelizing the process to reduce computation time.

6.0 Acknowledgements

We would like to acknowledge Professor Andrew Ng and the TA team of CS 229 for their advice on the project. We would also like to thank the developers of LIBSVM for their useful toolkit and hard work.

7.0 References

B. E. Boser, I. Guyon, and V. Vapnik. A training algorithm for optimal margin classifiers. *In Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, pages 144-152. ACM Press, 1992.

Chih-Chung Chang and Chih-Jen Lin, LIBSVM : a library for support vector machines, 2001. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.

C. Cortes and V. Vapnik. Support-vector network. *Machine Learning*, 20:273-297, 1995.

C.-W. Hsu, C.-C. Chang, and C.-J. Lin. A Practical Guide to Support Vector Classification. Department of Computer Science, National Taiwan University, 2010.

C.-W. Hsu and C.-J. Lin. A comparison of methods for multi-class support vector machines. *IEEE Transactions on Neural Networks*, 13(2):415-425, 2002.

S. S. Keerthi and C.-J. Lin. Asymptotic behaviors of support vector machines with Gaussian kernel. *Neural Computation*, 15(7):1667-1689, 2003.

S. Knerr, L. Personnaz, and G. Dreyfus. Single-layer learning revisited: a stepwise procedure for building and training a neural network. In J. Fogelman, editor, *Neurocomputing: Algorithms, Architectures and Applications*. Springer-Verlag, 1990.

H.-T. Lin and C.-J. Lin. A study on sigmoid kernels for SVM and the training of non-PSD kernels by SMO-type methods. Technical report, Department of Computer Science, National Taiwan University, 2003.