Yelp Recommendation System

Jason Ting, Swaroop Indra Ramaswamy
Institute for Computational and Mathematical Engineering

Abstract

We apply principles and techniques of recommendation systems to develop a predictive model of how customers would rate businesses they have not been to. Using Yelp's dataset, we extract collaborative and content based features to identify customer and restaurant profiles. We use generalized regression models, ensemble models, collaborative filtering and factorization machines. We evaluate the performance of the models using the Root Mean Squared Error (RMSE) metric and compare the models' performances. For dealing with cold start problems, we use segmentation ensembles and 3 different imputation methods (mean, random values and predicted values) for filling in missing information.

Introduction

Yelp is a web/mobile application that publishes crowdsourced reviews about local businesses and restaurants. The users of Yelp engage and interact with the application through searching businesses, writing reviews, rating businesses, connecting with other users, and "checking in" at businesses. Through this project we utilized Yelp's data to make personalized business recommendations for Yelp users by predicting the rating that a given user would give to a specific business.

We begin by describing the dataset used to predict ratings. This is followed by a description of the evaluation metric and the features used. We then elaborate on the different models used to predict the rating that a user would assign to a business. Later, we list the refinements we applied on our models to improve the performance. Finally, we conclude with our results, discussion and talk briefly about future direction.

Data

The primary dataset comes from the Yelp recommendation Kaggle competition[1]. This information contains actual business, user, and users' review data from the greater Phoenix, AZ metropolitan area as JSON files. In total there are 11,537 businesses, 43,873 users, and 229,907 reviews in the training data set and 1,205 businesses, 734 check-in sets, 5,105 users, and 36,404 reviews in the test data set. The ratings users give to businesses range from 1-5 as discrete values as a number of review stars.

Missing Data/Cold Start problem

Although all the data is available to download, some of the test data fields are missing certain features. For all the reviews in the test data, there are no written user reviews, so we cannot use text analysis to build a model even though the written user reviews are available in the training data. The data field that is missing is the average rating for the user/business which, unsurprisingly turns out to be best indicator of the predicted rating. Therefore how we handle missing data is a critical part of our project. A breakdown of the test data set which illustrates the different types of missing data can be seen in Figure 1.

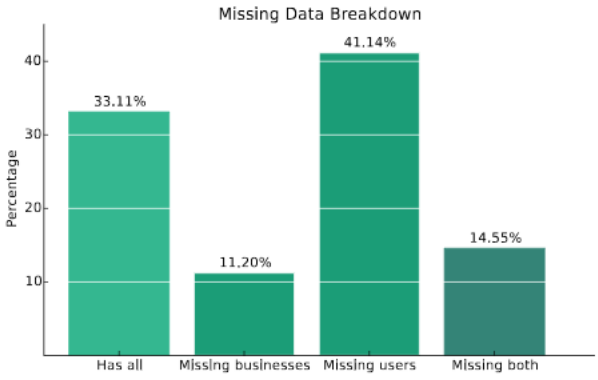

Figure 1: Test data set split

Because a significant portion of the test data is missing a field and a prediction for all the test data must be made to find the error, we use a technique called simple imputation to address the missing data in the model. Simple imputation is the process of replacing missing data with substituted values and is the standard approach for handling missing fields. We use the following three values for using simple imputation:

- The mean of the training set
- Random sample from the training set
- A regression estimate of the missing features using other features

Evaluation Metric

We chose to evaluate our model through the root mean squared error (RMSE) to measure the accuracy. The equation for the root mean squared error is as follows:

$$\text{RMSE} = \sqrt{\frac{\sum_{i=1}^{n}(p_i - a_i)^2}{n}}$$

n is the total number of review ratings to predict
p_i is the predicted rating for review i
a_i is the actual rating for review i

We evaluate the RMSE on the test data set by making submissions to the Kaggle contest. In this case, the training set RMSE is not a good indication of the test set RMSE due to data leakage in the training set. Say we split our training set into a cross validation set and a training set. For example, if a user has rated two businesses (user review count = 2) and the user's average rating is 4.5. Since the ratings can only be 1,2,3,4 or 5, it means that one of the reviews is a 5-star review and the other is a 4-star review. Now, if the 5-star review is present in the training set, and if we come across a review by the user in the cross-validation set, then we know that it is a 4-star review.

Although we have highlighted the issues with using cross-validation, we use leave one out cross validation to estimate some of our model parameters because it is the best estimate we can get given that we don't have access to the test data set.

Features

Raw input features: User ID, Business ID, User Average Rating, Business Average Rating, Category List, # "cool" votes for user, # "funny" votes for user, # "useful" votes for user, Bus Open/Closed, Business Review Count and User Review Count.

User and Business ID were binarized so that the i^{th} user/business is represented by a binary row vector x, with zeros every where except $x[i]$

User Average Rating and Business Average Rating are the strongest indicators of the rating given by a user to a business. # "cool" votes for user, # "funny" votes for user, # "useful" votes for user, Business Review Count and User Review Count are indicative of the reliability of the user average rating. Closed Businesses often have lower ratings (which is probably why they are closed!).

Derived/Computed features: Category average, Franchise Average, Gender, Name Score. Category average is a powerful indicator of the rating given by a user. Franchise averages are used to fill in for missing business averages. Gender is also a very good indicator. Name score is computed for each business based on the average rating of words in the training set. Some words tend to have a positive connotation. For example, Businesses that contain the words 'Diamondbacks' and 'Yelp' have much higher ratings.

Interaction Terms: For example, Gender terms interacting with categories/franchises, ie Women are likely to give lower ratings to sports bars, higher ratings to chiropractors etc.

Models

Linear Regression

Linear regression fits a linear model to minimize the residual sum of squares between the observed responses in the dataset. This model solves the following optimization problem to find the estimated parameters. The optimization problem is given by,

$$\theta = \text{argmin}_\theta ||X\theta - y||_2^2$$

Ridge Regression

Ridge regression uses the same model as linear regression. The ridge regression uses an optimization is similar to the linear regression model to find the features, except it imposes a penalty on the size of coefficients. The ridge regression minimizes the following equation:

$$\theta = \text{argmin}_\theta ||X\theta - y||_2^2 + \lambda||\theta||_2^2$$

λ is the shrinking parameter, and it affects the second term. The shrinkage parameter has the effect of shrinking the coefficients towards 0 by adding a cost to the l_2. norm By including the shrinkage parameter the model decreases the variance and increases the bias. In order to selecting the parameter λ, we performed leave one out cross validation on a series of λ values and selected λ that minimized the cross-validation error to refit on the entire model[4].

Lasso

The lasso is an alternative type of linear regression model to ridge regression, which uses the same model as linear regression. The lasso uses an optimization is similar to the the ridge regression with a different type of penalty. The lasso minimizes the following equation:

$$\theta = \text{argmin}_\theta ||X\theta - y||_2^2 + \lambda||\theta||_1$$

The equation is similar to the ridge regression except that the shrinkage parameter is replaced by $||\theta||_1$ in the lasso penalty, which is taking the l_1 norm. Just like the ridge regression, the lasso shrinks the coefficients towards zero, except for in the lasso some of the coefficients estimates can be exactly equal to 0. Just like selecting the shrinkage parameter in ridge regression, we performed leave one out cross validation on a series of λ values and selected the λ that minimized the cross-validation error to refit on the entire model[4].

Elastic Net

Elastic Net is a linear regression model trained with L_1 and L_2 prior as regularizer. This combination allows for learning a sparse model where few of the weights are non-zero like lasso, while still maintaining the regularization properties of ridge. We control the convex combination of L_1 and L_2 using the $l_1$ratio parameter. The practical advantage of trading-off between Lasso and Ridge is it allows Elastic-Net to inherit some of Ridge's stability under rotation. The objective function to minimize is in this case

$$\theta{=}\text{argmin}_\theta ||X\theta - y||_2^2 + \lambda||\theta||_2^2 + \lambda||\theta||_1$$

The tuning parameters were chosen by cross validation on a series of tuning parameter values values and selected the tuning parameters that minimized the cross-validation error to re-fit on the entire model[7].

Random Forest Regressors

Random forest is a type of ensemble learning method that is based from using decision trees. The goal of decision trees are to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features. Random forests are a way of averaging multiple deep decision trees trained on different parts of the same training set through bootstrapping and using only a subset of the features to create the trees, where an optimal number of trees can be found using cross-validation. This method reduces the variance since each tree uses only a subset of predictors at the expense of an increase in the bias and some loss of interpretability, but generally greatly boosts the performance of the final model.

In general random forest and decision trees are approaches to handle models when there is missing data since it can use an alternative predictors where the data is missing when splitting the node when constructing the tree. [4, 3]

Factorization Machine

Factorization machine as presented by Rendle[5], is a general predictor that is able to estimate reliable parameters under very high sparsity. The factorization machine models all nested variable interactions, but uses a factorized parameterization instead of a dense parametrization like in SVMs. The model equation of factorization machines can be computed in linear time and that it depends only on a linear number of parameters. This allows direct optimization and storage of model parameters without the need of storing any training data (e.g. support vectors) for prediction. In contrast to this, non-linear SVMs are usually optimized in the dual form and computing a prediction (the model equation) depends on parts of the training data.

The model equation for a factorization machine of degree d = 2 is defined as:

$$\hat{y}(x) = w_0 + \sum_{i=1}^{n} w_i x_i + \sum_{i=1}^{n} \sum_{j=i+1}^{n} \langle v_i, v_j \rangle \, x_i x_j$$

where the model parameters that have to be estimated are

$$w_o \in \mathbb{R}, w \in \mathbb{R}^n, V \in \mathbb{R}^{n \times k}$$

and$\langle v_i, v_j \rangle$ is the the inner product of v_i and v_j.

A row v_i within V describes the i^{th} variable with k factors. $k \in \mathbb{N}_0^+$ is a hyperparameter that defines the dimensionality of the factorization.

The key point is that instead of using a separate model parameter $w_{i,j}$ for each interaction, the interaction is modeled by factorizing it. This is very useful in sparse settings.

For example, Alice has rated Walmart and Rainbow and assume we want to estimate the interaction between Alice (A) and Trader Joe's (TJ) for predicting the target y (the rating). Obviously, there is no case x in the training data where both variables x_A and x_{TJ} are non-zero and thus a direct estimate would lead to no interaction ($w_{A,TJ} = 0$). But with the factorized interaction parameters $\langle v_A, v_{TJ} \rangle$ we can estimate the interaction even in this case. Say, Bob and Charlie have similar factor vectors v_B and v_C because both have similar interactions with Rainbow (v_R) for predicting ratings. Say, Alice has rated Walmart and Rainbow differently from Charlie. Alice (v_A) will have a different factor vector from Charlie (v_C) because she has different interactions with the factors of Walmart(v_W) and Rainbow(v_R) for predicting ratings. Say, Bob has given similar ratings to Trader Joe's and Rainbow. The factor vectors of Trader Joe's are likely to be similar to the one of Rainbow because Bob has similar interactions for both grocery stores for predicting y. In total, this means that the inner product (i.e. the interaction) of the factor vectors of Alice(v_A) and Trader Joe's(v_{TJ}) will be similar to the one of Alice(v_A) and Rainbow(v_R). Therefore, factorization machines handle sparse data far more elegantly than SVMs where we would not have been able to predict the interaction because we use a separate term $w_{A,TJ}$ to model the interaction.

Factorization machines can also mimic standard factorization models like Matrix Factorization (MF) and SVD++ with the right set of feature vectors. So just by changing the feature extraction method, we can mimic different factorization models.

Consider the Matrix Factorization model. It factorizes a relationship between two Users(U) and Businesses(B). The standard approach is to binarize the Users and the Businesses as described in the Features section.

$$n = |U \cup B|, \qquad x_j = \delta(j = b \vee j = u)$$

A FM using this feature vector x is identical to the Matrix Factorization model because x_j is non-zero for u

and i, so all other interactions and biases drop. Therefore, the model equation becomes,

$$\hat{y}(x) = w_0 + w_u + w_i + \langle v_u, v_i \rangle$$

The parameters for the factorization machine model were learned using Markov Chain Monte Carlo(MCMC) with the help of the `libFM` library[6]. It is also possible to learn them using Adaptive Stochastic Gradient Descent or Alternating Least Squares.

Refining our models

Collaborative Filtering

Collaborative filtering is a technique that identifies patterns of user preferences towards certain items and makes targeted recommendations. Collaborative filtering uses a sparse matrix holding the rating of users to businesses and calculates a similarity score between the users and a similarity score between the businesses. The algorithm for this type of collaborative filtering goes as follows:

1. Initialize $x^{(1)}, ..., x^{(n_b)}, \theta^{(1)}, ..., \theta^{(n_u)}$ to small random values. $x^{(i)}$ denotes the features of business i and $\theta^{(j)}$ denotes the features of user j.
2. Minimize the objective function J with gradient descent to find the parameters. The objective function is defined as follows.

$$J\left(x^{(1)}, ..., x^{(n_b)}, \theta^{(1)}, ..., \theta^{(n_u)}\right) = \frac{1}{2} \sum_{(i,j):r(i,j)=1} \left(\left(\theta^{(j)}\right)^T x^{(i)} - y^{(i,j)}\right)^2 + \frac{\lambda}{2} \sum_{i=1}^{n_b} \sum_{k=1}^{n} \left(x_k^{(i)}\right)^2 + \frac{\lambda}{2} \sum_{j=1}^{n_u} \sum_{k=1}^{n} \binom{(j)}{k}^2$$

For a user with parameter $\theta^{(j)}$ and a businesses with features $x^{(i)}$, the predicted rating is $\left(\theta^{(j)}\right)^T x^{(i)}$.[2]

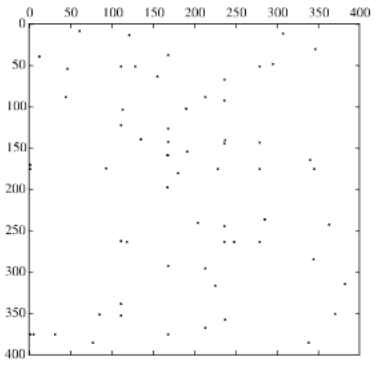

Figure 2: A random 400x400 section of the user-business matrix. The overall sparsity is 99.56%.

In order for collaborative filtering to work efficiently, we require the user-business matrix to be fairly dense. In case a user has barely visited any businesses, then the similarity scores computed for the user are unreliable and therefore do a poor job of estimating the predicted rating. Similarly, for the case when a business has had very few reviewers, the similarity scores are unreliable. Therefore, if the user-business matrix is sparse, then collaborative filtering is likely to do a very poor job.

As we can see from Figure 2, the user-business matrix for the training set is very sparse and therefore collaborative filtering is not likely to work very well. Therefore, we take a much smaller, denser matrix and apply collaborative filtering to that matrix. This matrix is such that all the users in the matrix have rated at least 15 businesses in the matrix and all the businesses in the matrix have been rated by at least 15 users in the matrix. The results are then merged with the results from the other models.

Principal Components Analysis

Principal Components Analysis (PCA) reduces the model's complexity and the random noise. PCA is linear dimensionality reduction using Singular Value Decomposition of the data and keeping only the most significant singular vectors to project the data to a lower dimensional space. Essentially, we find the directions that account for most of the variability in the data. These are the first n eigenvectors of the covariance matrix, $X^T X$. We use PCA to reduce the dimensionality of our feature matrix.

Segmentation Ensemble

Segmentation ensemble is a technique that divides the data into different partitions and uses a distinct model for each partition and combines the results. The primary idea for this method is that each model would have different weights for each feature since the partition for each dataset vary greatly from each other, thus improving the overall results. The partition for the data was based on which portion of the data is missing. For example, for the part of the test data set where the user average rating is missing, we build a model without the user average rating.

Results

The models were trained using the training set of size 229,907 and test set of size 36,404. The RMSE was calculated by using the model's prediction and submitting it via Kaggle. The results are graphed out in Figure 3.

We can see from Figure 3 that the best model is Elastic Net with random value imputation. We use our model refining techniques in conjunction with Elastic Nets to further improve our predictions. PCA and Collaborative filtering were applied on the matrix with random value imputation. We also tried segmentation ensembles to handle the missing data and used Elastic

Nets to predict the rating. These results are presented in Figure 4.

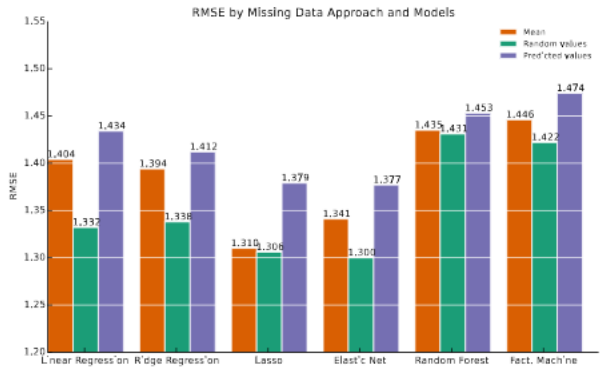

Figure 3: RMSE for the different models with different imputation methods

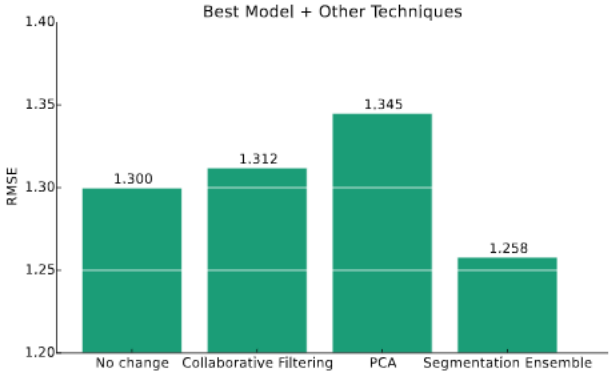

Figure 4: Elastic Net with refinements

Conclusion and Discussion

Using random values for simple imputation and the Elastic Net model performed the best. Combining those 2 approaches with segmentation ensemble yielded the best result.

It is not very surprising that collaborative filtering does not work very well with this dataset. The number of users who have rated businesses for them to have a reliable enough similarity score with other users is very low. It is surprising that filling in with random values instead mean or predicted values did better across different models, since random values don't take into account the inherent bias. Given that the Random Forest handles missing data well, the poor performance is unexpected. The good performance of the Lasso method and the Elastic Net method suggests there are extraneous features in the model that these models were able to detect and reduce. Factorization machines did not work as well as expected, given that they have a track record of doing extremely well in similar settings (KDD cup, Netflix dataset). It is possible that factorization machines did not work as well because of the lack of regularization parameters in the formulation. If we add regularization parameters to the factorization machines model, then we have to solve a non-linear dual problem which is often not feasible for large data sets like this. We were able to score in the top 25% of the Kaggle leaderboard with the best score, so we believe that our overall results are good. Further, most of the submissions in the top 25% include using up to 20 different models to fit different subparts of the dataset. We did not adopt this approach as we believe that a general model is better since it is more likely to be robust and give good results for other test sets.

Future Work

Some improvements that can be made to the performance of the problem are the following.

- Figure out why factorization machines performed so poorly in our setting and improve the model (for instance, by adding regularization terms).
- Consider other features in the model such as using locations derived from Business addresses to see if particular neighborhoods had overall higher ratings and deriving more information about the user from his/her name.
- Explore alternative ways of handling missing data.

References

[1] Recsys challenge 2013: Yelp business rating prediction. https://www.kaggle.com/c/yelp-recsys-2013. Accessed : 2014-12-12.

[2] Montanari A. Recommendation systems. EE378B: Inference, Estimation and Information Processing, 2013.

[3] Leo Breiman. Random forests. *Machine Learning*, 2001.

[4] Daniela Witten Trevor Hastie James, Gareth and Robert Tibshirani. *An Introduction to Statistical Learning: With Applications in R*. Springer.

[5] Steffen Rendle. Factorization machines. In *Proceedings of the 10th IEEE International Conference on Data Mining*. IEEE Computer Society, 2010.

[6] Steffen Rendle. Factorization machines with libFM. *ACM Trans. Intell. Syst. Technol.*, 3(3):57:1–57:22, May 2012.

[7] Hastie Trevor Zou, Hui. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society*, 2005.