Predicting Jobs of Users' Interest

Vincent Yun Lou
Computer Science Department
Stanford University
yunlou@stanford.edu

Jinchao Ye
Computer Science Department
Stanford University
jcye@stanford.edu

Abstract

Nowadays, more and more companies are dedicated to help people to find ideal jobs. Such websites includes LinkedIn, GlassDoor, CareerBuilder and etc. Good job recommendation will benefit not only applicants, but also employers. Therefore, good job recommendation will make such companies more competitive.

Current algorithms usually categorize an applicant and then recommend popular jobs in that category to that applicant. However, many factors will affect which jobs an applicant will likely to apply, including location, salary, employer's reputation, the applicant's working history, the applicant's recent applications and so on.

We plan to extract a feature vector from the job description of each job and use logistic regression and collaborative filtering to classify whether a specific user will likely to apply.

Future Distribution Permission

Data

The project idea is from a competition on Kaggle. We also got the data there. There are 2 files:

user_history.tsv contains information about a user's work history. Each row of this file describes a job that a user held. Each row consists of UserId and JobID.

apps.tsv contains information about applications made by users to jobs. Each row describes an application. The UserID, and JobID columns have the same meanings as above, and the ApplicationDate column indicates the date and time at which UserID applied to JobId.

The whole dataset consists of 353582 applications from 63412 users to 81213 jobs. We use the first 53412 users as the training data set and the last 10000 users as test data set. We pick a particular job to see predict whether the test users will apply to that particular job.

Methods

Feature extraction

1. Using only apps.tsv

For each user i, X_i denotes the feature vector of the user. The element in X_i at index j is defined as:

$$X_i(j) = \begin{cases} 1, & if\ user\ i\ applied\ to\ job\ j \\ 0, & if\ user\ i\ didn't\ apply\ to\ job\ j \end{cases}$$

2. Using both apps.tsv and user_history.tsv

For each user i, X_i' denotes the feature vector of the user.

Learning and inference

1. Logistic regression

Suppose we want to test job s.

Let y_i denote whether user i actually applied to job s (1) or not (0).

Hypothesis for logistic regression:

$$h_\theta(x) = \frac{1}{1+e^{-\theta^T x}}$$

Then

$$p(y|x;\theta) = \big(h_\theta(x)\big)^y \big(1 - h_\theta(x)\big)^{1-y}$$

Likelihood function:

$$L_\theta = \prod_{i=1}^{m} \big(h_\theta(x_i)\big)^{y_i} \big(1 - h_\theta(x_i)\big)^{1-y_i}$$

We can get θ using maximum likelihood.

As for interference,

$$y_i(predict) = \begin{cases} 1, & h_\theta(x_i) < threshold \\ 0, & h_\theta(x_i) \geq threshold \end{cases}$$

Here threshold is a parameter we can control.

Of course, we actually implement logistic regression using LIBLINEAR.

2. Collaborative filtering

Now we have a job application vector in each user; given any two users, we use the following equation to calculate the similarity score, which is denoted as $S_{i,j}$:

$$S_{i,j} = \frac{X_i^T X_j}{\sqrt{||X_i|| \, ||X_j||}}$$

Where $||X_i||$ is L2 norm of X_i.

For each test user, we calculate the similarity score between the user and every user in the training set and keeps the top 200 most similar users to him/her. We denote the set SU.

To predict whether a test user will apply for the j^{th} job or not, we use the following formula to get a score $I_{u,j}$ for user u on job j:

$$I_{u,j} = \frac{\sum_{w \in SU} S_{u,w} * X_w(j)}{\sum_{w \in SU} S_{u,w}}$$

When the score is greater than threshold, which is controlled by us, we predict that the user wants to apply for the jobs.

Evaluation Metric

We cannot use accuracy as the metric here because typical user applies to 10 jobs, while there are 81213 jobs. So the accuracy is almost always larger than 99.9% if we simply predict they don't apply for any job.

Two reasonable evaluation metrics are as following:

1. Precision-Recall Curve
   We can get precision and recall when we fix a threshold. By varying the threshold, we can plot a precision-recall curve.
2. F1 score

$$F1 = \frac{2PR}{P + R}$$