COMP90051 Statistical and Evolutionary Learning Project 1 Report

Yuan Li & Hao Wang

September 4, 2014

Abstract

In this report, we use the k-NN Regression to predict the location of a user based on social graph and limited profile information. We present our proposed k-NN Regression method as well as discuss other approaches.

1 Final Approach

In this section, we describe our k-NN Regression method. First, we present the vector space model that we use to represent users. Then, we describe our k-NN Regression method which utilizes dot product as similarity measure and predicts the location for a user as the weighted average location of users whose similarity are in top-k.

1.1 Vector Space Model

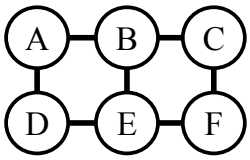
One difficulty of this project is to extract features and represent users in feature vectors if we want to use existing machine learning algorithms. Inspired by the vector space model used in information retrieval, we represent users as vectors of identifiers, where the identifiers are user index and hour index. For example, in Fig. 1, the vector of a user has 10 dimensions, where the identifiers include $\{A, B, C, D, E, F\}$ and $\{0, 1, 2, 3\}$.

Let $\mathcal{T}$ denote the training set, $\mathcal{Q}$ the test set, $\mathcal{F}(u)$ the set of user u's friends, $\mathcal{H}(u)$ the set of user u's frequent hours. Suppose that we know the locations of B, C, D, E, F and would like to predict the location of A, then $\mathcal{T} = \{B, C, D, E, F\}$, $\mathcal{Q} = \{A\}$.

For a user $u \in \mathcal{T}$, the vector of u is calculated as follows: the value of user v in the vector is 1 if $v \in \mathcal{F}(u)$, 0 otherwise; the value of hour h in the vector is 1 if $h \in \mathcal{H}(u)$, 0 otherwise.

For a user $u \in \mathcal{Q}$, the vector of u is calculated as follows: the value of user v in the vector is α if $v = u$, β_v if $v \in \mathcal{F}(u)$, 0 otherwise; the value of hour h in the vector is γ if $h \in \mathcal{H}(u)$, 0 otherwise; α and γ are parameters, β_v is a "idf"-like value calculated for user v, i.e. the more friend user v has, the smaller β_v is.

In our final approach, we set $\alpha = 2$, $\beta_v = \frac{1}{\sqrt[4]{|\mathcal{F}(v) \cap \mathcal{T}| + 1}}$, $\gamma = \frac{1}{2000}$. For example, $\beta_B = \frac{1}{\sqrt[4]{3}}$ for that $|\mathcal{F}(B) \cap \mathcal{T}| = |\{C, E\}| = 2$. Vectors for users in $\mathcal{T}$ are shown in Fig. 2. According to the description, the vector for A is $(2, \frac{1}{\sqrt[4]{3}}, 0, \frac{1}{\sqrt[4]{2}}, 0, 0, 0, \frac{1}{2000}, \frac{1}{2000}, \frac{1}{2000})$.



	Hours		Hours
A	(1,2,3)	D	(1,2,3)
B	(0,1,3)	E	(0,1,2)
C	(0,1,2)	F	(0,2,3)

Figure 1: Example graph and frequent hour table

		User						Hour			
		A	B	C	D	E	F	0	1	2	3
Query	A	α	β_B	0	β_D	0	0	0	γ	γ	γ
Training Instances	B	1	0	1	0	1	0	1	1	0	1
	C	0	1	0	0	0	1	1	1	1	0
	D	1	0	0	0	1	0	0	1	1	1
	E	0	1	0	1	0	1	1	1	1	0
	F	0	0	1	0	1	0	1	0	1	1

Figure 2: Vector table

1.2 k-NN Regression

We first define the similarity between two vectors, then use the reciprocal of similarity as distance. Inspired by the cosine similarity and pivoted document length normalization, we define the similarity between two users $u \in \mathcal{Q}$, $v \in \mathcal{T}$ as

$$\text{simi}(u, v) = \frac{\text{vec}_u \cdot \text{vec}_v}{|\text{vec}_u|(s|\text{vec}_v| + (1-s)L_{\text{avg}})}$$

where vec_u and vec_u are the vectors of u and v, $s \in [0, 1]$ is called slope parameter, L_{avg} is the average length of vectors in training set, i.e. $L_{\text{avg}} = \frac{1}{|\mathcal{T}|}\sum_{v\in\mathcal{T}} |\text{vec}_v|$.

Let l_v denote the location of user v, $\hat{l}_u$ our prediction for user u's location, $\mathcal{R}(u)$ the set of user u's k nearest neighbours in $\mathcal{T}$, then $\hat{l}_u$ is calculated as

$$\hat{l}_u = \frac{\sum_{v\in\mathcal{R}(u)} \text{simi}(u, v) \cdot l_v}{\sum_{v\in\mathcal{R}(u)} \text{simi}(u, v)}$$

We set $k = 40$, $s = 0.5$ in our final approach and achieves a public score of 28.15952 and a private score of 25.80840 in the Kaggle completion.

1.3 Summary

As described above, we begin with the idea that use friends to represent a user like use terms to represent a document. Then we borrow the idea of cosine similarity and pivoted document length normalization from IR. Our prosed method achieves satisfactory result, however, this method seems to be a little ad-hoc and doesn't have a theoretical background.

2 Discussion

2.1 SVM or k-NN?

Both SVM and k-NN can do regression task. Actually, their output function are similar. According to the Representer Theorem, the output function of SVM can be written as

$$\sum_i \alpha_i \cdot \text{simi}(x_i, x)$$

where $\text{simi}(\cdot, \cdot)$ can be dot product or kernel function. Whereas, the output function of k-NN Regression is like

$$\frac{\sum_i \alpha_i \cdot \text{simi}(x_i, x)}{\sum_i \text{simi}(x_i, x)}$$

where $\text{simi}(\cdot, \cdot)$ is as we defined in Section 1.

One reason why we don't use SVM is that it looks more suitable for fitting a mixed model. For example, we would like to fit a curve which is known to be a mixture of several gaussian function, then an SVM using RBF kernel has the exact same form of output function. Another reason is that k-NN Regression is more flexible that we can choose different k for different users. For example, choose smaller k for users have less friends and larger k for users have more friends. Whereas, SVM will use all the support vectors regardless of the input.

2.2 Gaussian Model

An approach that we didn't further implement is called Gaussian Model. The assumption is that the locations of a user's friends follows Gaussian distribution. We can first estimate the mean μ and variance σ for every user based on the location of his/her friends. When μ and σ of all users are known, we can use a maximum likelihood method to predict the location of user u as

$$\hat{l}_u = \arg\max_{l_u} \prod_{v \in \mathcal{F}(u)} \exp\{-\frac{(l_u - \mu_v)^2}{2\sigma^2}\}$$

However, the assumption is not true for some users whose friends live in different countries, e.g. U.S. and Japan. One possible way to address this problem is clustering the locations of friends first and then use a mixed Gaussian model instead of a single distribution, though we didn't further implement this method.