Using a Growing Neural Gas Network to Recommend Restaurants to Yelp Users

Wesley Fishburn and Razi Shaban

May 4, 2015

1 Introduction

For our project we set out to design a restaurant recommendation system for the crowd-sourced online review site Yelp. After considering several methods, we settled on employing a growing neural gas (GNG) network to implement our project. Broadly speaking, a GNG is an incremental network model that when given a set of input vectors can create a network of topological relationships. With data provided by Yelp as part of the 2015 Yelp Dataset Challenge, and after some thorough data pre-processing, we used restaurant reviews in Pittsburgh from 908 users reviewing 1350 local businesses to cluster users on a plot. Having constructed this topological mapping of users and reviews, we are able to recommend similar users to follow and restaurants to visit to a given user based on their review history.

2 Growing Neural Gas network

A GNG network is comprised of a set of units (nodes), in which each unit has a corresponding reference vector [Fritzke et al., 1995]. These reference vectors are used to maintain the position of units in the input space.

The general idea of the growing neural gas algorithm is to systematically add new units to a network by calculating local statistic measures dating back to previous insertions.

Initially, the network has only two units that are randomly placed in the network. When a new unit is to be inserted, s_0, the algorithm finds the two

nearest units, s_1 and s_2, and increments the age of all edges connected to s_1. Next, the error for s_2 is incremented by adding the squared Euclidean distance between the newly inserted unit and s_1. Upon re-calculating error, s_1 and its neighbors (neighbors here is defined as units connected by edges to s_1) are moved towards s_0 by the distance of some constant, k, times s_0 - s_n. In mathematical terms,

$$\Delta s_1 = k(s_0 - s_1)$$
$$\Delta s_n = k(s_0 - s_n)$$

If s_1 and s_2 are not previously joined by an edge, an edge is inserted to connect them; however, if one exists, the age of this edge is reset to zero. Before continuing to the next iteration, every edge in the network is checked to verify its age is larger than the maximum age specified. If such an edge exists, it removed is from the graph to filter out edges that are not reinforced by multiple insertions. If the removal of expired edges creates units that are not connected to any edges, these units are removed, as well.

If the number of units in the graph equals an integer multiple of the parameter λ, then a new unit, r, is inserted between the unit with the highest accumulated error in the network, q, and q's neighbor with the highest error, f. After the insertion, the edge connecting q and f is deleted, and new edges connecting r to both q and f, respectively, are created. Lastly, q and f's error values are decreased by multiplying them by a constant. The error value for r is then set equal to q's new error value.

Next, the error value for all units in the network is decreased by multiplying them all by a constant.

Finally, if the network has not yet reached its stopping criteria (for example, a maximum size or satisfied some performance measure), this series of steps is repeated.

In summary, new units are inserted in between its two most similar pre-existing units. Relationships between units are maintained by slowly decreasing error throughout the network, augmenting the shape of the network to group similar units together so that they are proximate to one another.

3 Details

The first step of the recommendation system is preprocessing the raw data that Yelp provided. Initially, the dataset included 1.6 million reviews from

366,000 users reviewing over 61,000 businesses with 481,000 business attributes (e.g. hours, parking availability, ambiance) in ten cities worldwide. In order to give our project a more narrow focus, we decided to focus on restaurants in one city (Pittsburgh) and one attribute (5 star rating scale). In the file **parse.py**, we searched for restaurants in Pittsburgh that had at least ten reviews, and for users that had reviewed at least ten restaurants. It was important that we limit our data to restaurants and users with at least ten reviews because we needed to ensure there would be sufficient overlap between users so that we could analyze common restaurants reviewed by the user-base. After parsing the data, we ended up with 1350 restaurants and 908 users with at least ten reviews per restaurant and user, respectively. In order to insert the data into a GNG network, we created a dictionary of dictionaries, in which the key is a userID and the value is a subsequent dictionary with businessIDs as keys and star ratings (1-5 stars) as the values. This allowed us to access any user's entire rating history.

We used a GNG network previously implemented by Professor Lisa Meeden. Professor Meeden's implementation contains a Unit class, in which each unit of the GNG network has a reference vector, error measure, and a list of edges to neighboring units. One adjustment we needed to make to the Unit class to suit our project was to adjust the minVal and maxval of each unit to 0 and 5, respectively, to represent the realm of possible star ratings a restaurant could receive from a user. We used zero to indicate that a user had not reviewed a restaurant. Professor Meeden's implementation also includes an Edge class. All edges in the GNG network are undirected but are created as a set of two directed edges going both ways for ease of implementation. Last is the GrowingNeuralGas class, which is always initialized with two randomly placed units and generates the next point from the generateNext() function (an input parameter). The heart of the GrowingNeuralGas class is the step() function, which is called by main() repeatedly to move through the GNG process. Step() begins by receiving a new point from generateNext(), and clusters it into the plot, situating it next to its nearest matches. A given point's nearest neighbors, in terms of Euclidean distance, represent users who are the closest match based on their rating history. Once we determine a user's closest user match(es), we can recommend restaurants that like-minded users have also rated highly and users to follow with similar tastes.

The key function that we needed to implement to adapt Professor Meeden's implementation to our project was generateNext(). Our generateNext()

selected a random userID by calling randrange() on the length of the dictionary, and returning the indexed value of the dictionary using the randomly generated value. Our generateNext is passed in as parameter when called the GrowingNeuralGas().

Another important alteration we made to Professor Meeden's implementation was adding userID as a data member to the Unit class. This was necessary because when calling computeDistances(), which returns a unit's closest and next closest units, we needed to be able to access the corresponding userIDs of the closest and next closest units in order to look up their restaurant history.

In order to minimize error, we adjusted several parameters to attain optimal accuracy. The variable that we ended up determining having the largest effect on average error was self.stepsToInsert, which regulated how frequently the network inserted a new data point. Initially, it was set to 10, which resulted in a final average error of about 66 after 5000 steps. After several trials with different stepsToInsert values, we found that a stepsToInsert value of 2 allowed us to minimize error the most effectively, reducing average error to about 3 after 6000 steps. We found that error started to plateau at about 6000 steps, so we used 6000 as a cut-off point.

4 Results

We were ultimately able to successfully recommend restaurants and fellow users to follow for a given userID. The interface asks for a userID returns both a list of both other users to follow and restaurants the network recommends. A sample run is shown below:

```
Loading user and restaurant data...
Restaurants: 1350
Users: 908
Building GNG network...
[...]
GNG step 6000
Number of units: 3001
Average error: 3.57087865848
```

```
Recommendation system ready!
Input user ID or 'q' to exit: x1qmBwKEOFCaGiYxs6ppRw
We recommend you follow the following users:  AtLRo5Dx0ir00RK_3lhxKQ
We recommend you follow the following users:  _gA5BmEB4PRR2QsZFSOdpw

We recommend you look at the following restaurants:
TRLMm_deLvlPNmGNqRxcow  with a rating of  5
vcZWlI0eHiGwlUfP-5Ov8Q  with a rating of  5
y_jsnI8NUThDecWmgL_soA  with a rating of  5
ZL6DNzyenFm9Kpz6uADlNg  with a rating of  5
H18C9tH4BPHkxan96bkMXQ  with a rating of  5
ZDKi3qsO8lSSogrIdbLJKg  with a rating of  5
x8GTTMIAtUJ5EsIMnsOacQ  with a rating of  5
V7r0F2kr1WlyNOBy-C_HMA  with a rating of  5
OhjPgyBZudumBTF52HKMGg  with a rating of  5
AwaxCI7NMG3_nqCLqC4qWw  with a rating of  5
```

These results convey that the GNG network successfully clustered the data and was able to find matches between users. For better integration into the Yelp data system, we chose not to render the restaurant IDs as their names

5 Conclusions

Our project was ultimately successful, in that we were able to use a growing neural gas network to cluster Yelp data efficiently. This allowed us to make restaurant recommendations and other user recommendations. This was exciting on several levels, the first of which was that we learned a new machine learning technique and used it effectively to create something with a real-world application. Additionally, it is satisfying to envision how by connecting users with other users, there could be the beginnings of something of a social network on Yelp, in the sense that users could begin to interact with each other and gain from each other's experiences.

References

B. Fritzke et al. A growing neural gas network learns topologies. *Advances in neural information processing systems*, 7:625–632, 1995.