Jonathan Smith
Prof. McCauley
CSE 190A, FA
December 1, 2015

Assignment 2: Data Mine Kampf

Background

Wal-Mart uses data mining techniques to categorize the shopping trips their customers are taking so that they may better serve their customer base. While Wal-Mart has an extensive list of categories that they use to describe their customer's shopping habits, the issue is that they have a rather slow way to fit these categories to the purchase data. This is particularly troublesome when you consider the volume of purchases that Wal-Mart expects to see as the zombie hordes converge on the establishment in search of shiny new toys – their poor data miners won't keep up! As such, Wal-Mart is sponsoring a competition on Kaggle to find a better feature set that helps them categorize their customers and their purchases faster.

Predictive Task

Given the data set, train a model to help categorize customers and their purchases into the categories that Wal-mart provides.

Description of the Data

Before we tailor a good model to our data, It is prudent to play with the data to it in order to understand its particulars. In this section we will look at a summary of the descriptions of our dimensions, the form they take and their respective lengths, and then we will examine the number of unique items in each of them.

1. *Visit Number* denotes one customer's Wal-Mart visit at a single time.
2. *Weekday* denotes the day of the visit from Monday to Sunday.
3. *UPC* refers to the barcode of the item purchased.
4. *Scan Count* refers to the number of items of a particular UPC purchased.
5. *Department Description* is a high-level description of the department that the item belongs to.
6. *Fineline Number* refers to a more refined category for the products than department description.

In the training set, we are given one more dimension -- the *TripType*. This is a numerical identifier for the type of trip that the customer is taking, and is the ground truth for what we are asked to predict in the test set. To reiterate, the goal here is to categorize the type of visit using a description of the items purchased, they day they were purchased, and the departments they were purchased in.

On initial examination, our training data set consists of *647,054* purchase records, which falls above the number of samples needed for the purposes of this assignment. Additionally, we have ~5,000 unique fine lines, 69 unique departments, 97,715 unique kinds of product, and most importantly, *38 different kinds of trip types*. We will need to fit a model that can help us predict 38 different kinds of trip types from our data set.

Evaluation Measure

According to the Kaggle evaluation page, submissions in this competition are evaluated using the multi-class logarithmic loss, wherein for each visit, we submit a set of predicted probabilities (one for every trip type) following the formula:

$$-\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M} y_{ij}\log(p_{ij}),$$

where N is the number of visits in the test set, M is the trip type number, log() is the natural log, Y_ij is 1 if observation i belongs to class j and 0 otherwise, and where P_ij is the

prediction that observation i belongs to class j. Another key thing to remember is because the probabilities are rescaled by the row sum, they submitted probabilities are not required to sum to 1. This is evaluation measure helps us select a model that can help us solve this problem. Throughout the paper, however, since we are dealing primarily with the training dataset and its validation portion, we will be using Mean Squared Error.

Analysis of Requirements

In light of these requirements and the nature of the data set, we can immediately make the following observations.

1. We need a model that can help us *categorize* into 30+ discrete sets of *probabilistic outputs* which can extend to a training set of substantially bigger corpus size.
2. We have a relatively low number of training points to work with, so we need to be careful of overfitting if we use regression-based approaches.
3. There are different "types" of data here. Days of the week are included, which is a temporal aspect of the corpus. Fineline categories may be interpreted to be subset of larger departmental categories. Item counts take an appearance.
4. The model must be quick to make.

Immediately coming to mind, following algorithms learned in class, we have three options available to us. First, from a clustering standpoint, we have the K-Means algorithm. Second, from a categorization standpoint we have the K-Nearest Neighbor algorithm. Finally, from a probabilistic standpoint, we have the Bayes algorithm. These three are listed in the top 10 "Most Important" data mining algorithms to date according to a Knowledge and Information Systems paper dated 2006, and they will be expressed in brief here.

Literature

K-Means

K-Means is an agglomerative unsupervised clustering algorithm where given the corpus and the number of groups desired, it will output that number of clusters in the data set. The disadvantage of using this algorithm is that we do not know a priori whether the clusters we see correspond to Wal-Mart's propriertary TripTypes. We are not given a high-level description of trip types, only numerical identifiers, which means that at best we can try to fit the groups found by our algorithm to those groups in the TripTypes list, and at worst, completely miss the mark by assigning the wrong group to the wrong trip type, and fail on the test set and Kaggle's review set. Therefore K-means will not be useful to us.

K-Nearest-Neighbor

K-Nearest-Neighbor is a classification algorithm where given the corpus, a metric for evaluating distances between data points in the corpus, and a set of labels, we can assign a set of labels to our data points. It is one of the more relevant solutions to our problem and solves our task exactly, but it also has the following drawbacks.

First, its outputs are not probabilistic, which means that the saving grace our evaluation measure provides goes by the wayside. Computing probabilities is helpful in the event of a false classification so that you do not eat big errors. For example, if I predict a [0, 0, 1, 0] and the classification is [1, 0, 0, 0], then errors count heavily against me, moreso than if my prediction were [0, 0, .6, 0]. We may be able to remedy this by subtracting off an error measure according to the distance between the data point and the centroid, but that invites the following problem:

Since we have no a priori knowledge of what kind of categories we are observing, the categories themselves may vary in distance.

TripTypes which are close together in the corpus may combine algorithmically, and another TripType which means that other categories may be captured that do not belong in our corpus, and items which are close together in the corpus would be combined under wrong TripTypes. Utilizing K-Nearest-Neighbor assumes that data falls under a certain distribution, and that may lead to classification error if we are using the incorrect distance measure.

Bayes

One of the simplest things to implement would be the Bayes algorithm, where the inputs are the feature sets (counts work effectively here), and the outputs are a prediction for whether it falls under a specific category. The drawback here is that we have little time to construct 38 different models each replete with their own feature set, and the issue of double counting becomes ever a problem with our coarse "department" feature and our finer-tuned "fineline" feature. (Wu, Kumar, and Co.)

The Modeling Procedure

It seems here that our best option would be K-Nearest-Neighbor, since we have libraries which make this algorithm accessible, we have the time to build the model, we potentially have a mechanism to deal with error in calculating distances from centroids, and we simply don't know enough about the nature of our data to make a call as to whether or not TripType proximity will be problematic. So let's begin.

The basic KNN algorithm runs as follows:

1. **Acquire** data: Open CSV file and grab all necessary features.
2. **Similarity**: Calculate the distance between two data instances using Euclidean distance.
3. **Neighbors**: Locate k most similar data instances according to the similarity function.
4. **Response**: Generate a response from a set of data instances using a voting mechanism.
5. **Accuracy**: Summarize the accuracy of predictions by calculating the mean-squared error.

(machinelearningmastery.com)

Trials

Trial 1

First, we will apply a manual version of K-NN to the data set without any strict modifications to the data itself. For now, we will omit the Department feature entirely since its content is not in integer form and its dimensions are covered by the Fineline category, use the Euclidean distance as the distance measure, and see what happens.

Features: {Scan Count, Visit Count, UPC, Fineline Number} Neighbors: 2 Results: N/A

Observations

This particular algorithm took far too long to run (nixed it) because the neighborhood voting step required sorting a constantly growing list of tuples. It also surfaced that the Euclidean distance is heavily dependent on the nature of the other datapoints. Because distances between datapoints varied according to the type of feature (some features have bigger distances between each other, others have smaller ones), Euclidean distances are hard to manage with differing datapoint types.

Trial 2

Next, we employ Scikit Learn K-Nearest Neighbor libraries to acquire the distance measures between test tuples and our data points. The algorithm itself automatically chooses between "ball_tree", "brute force" and "kd_tree" depending on the fit of our

data, which solves the problem mentioned above.

Features: {Scan Count, Visit Count, UPC, Fineline Number}
Equation: Auto (Ball_tree)
Neighbors: 2
Results: 3.8% Correct

Trial 3

Next, we will apply K-NN to the data by adding in the *Scan Count* feature.

Features: {Scan Count, Visit Count, UPC, Fineline Number}
Equation: Auto (Ball_tree)
Neighbors: 2
Results: 86.89% Mean Squared Error

Observations:
Jeez! What a jump. The number of items purchased seems to heavily correlate to the categories that Wal-Mart has selected. The working code can be found in **Appendix 1.** Chances are that also employing the Department feature set will solidify these, but there may be double-counting involved, which will increase the weight that department-related categories hold. Since Fineline categories held little sway in Trial 2, this would not be an effective use of our time. Instead, we will play with the number of nearest-neighbors.

Trial 4

We will apply K-NN with the same features as above, using 5 neighbors.

Features: {Scan Count, Visit Count, UPC, Fineline Number}
Neighbors: 5
Results: 86.80% Mean Squared Error

Trial 6

We will apply K-NN with 10 neighbors.

Features: {Scan Count, Visit Count, UPC, Fineline Number}
Neighbors: 10
Results: 86.80% Mean Squared Error

Trial 7

We will apply K-NN with 50 neighbors.

Features: {Scan Count, Visit Count, UPC, Fineline Number}
Neighbors: 50
Results: 86.80% Mean Squared Error

Observations
We observe in the previous three trials we observe the effect of multiple neighbors. The categorization and voting process converges to some defined number and there is a negligible increase in the utility of including more neighbors.

Trial 8

We will apply K-NN with 1 neighbors.

Features: *{Scan Count, Visit Count, UPC, Fineline Number}*
Neighbors: 1
Results: 95.8% Mean Squared Error

Weird! This should be a seriously underfit model. This is like saying the closest centroid directly corresponds to the data point's category. At this point we will explore what happens when we add one final feature -- the department.

Trial 9

We will apply K-NN with 2 neighbors.

Features: {Scan Count, Visit Count, UPC, Fineline Number, Department Number}
Neighbors: 2
Results: 95.8% Mean Squared Error

Because the Department is actually a textual representation of the part of the store the item was picked up from (whether it be Men's Fashion or Electronics, and so forth), we must generate a mechanism to create an index for it in some space, and have faith that SciKit can accommodate for our chosen sampling distance (which is just a linear increase according to alphabetical order).

Analysis

We have observed the following things from our implementation of K-NN.

1. K-NN is sufficient to model the state of our data and acquire nonnegligible results
2. An increase in the number of neighbors beyond some maximum does not increase the MSE.
3. The equation used to determine the distances between the corpus and a test point matters. Euclidean space-based distances works for distances which are homogenous or normalized according to their demarcation lines (the tick marks of an axis), whereas other tree-based algorithms are employed to accommodate for distances otherwise relevant.
4. The Categories selected by Wal-mart are highly correlated to the number of purchases a customer makes of a given item.

Conclusion

The results of our paper conclude that K-NN should be employed whenever there is a corpus whose data has already been classified and we need to classify further data, as we needed to in this task.

The numerical categories Wal-Mart had us attempt to predict seemed fairly arbitrary at first, but after careful examination we found that they were strongly correlated to the Scan Count feature set, which implies that when people buy things in bulk it is easier to categorize what kind of trip they are making. This makes some logical sense considering that a grocery trip consists of buying foods in bulk, whereas an electronics trip consists of buying electronics items.

Out of an urbane sense of curiosity, this student also learned that there –is- a slight uptick in the number of electronics a user buys on Fridays, which is perhaps an homage to that wonderful time of year where we get to shirk any pretense that our country was founded on any principles other than personal gain and beating the Joneses. Happy Thanksgiving!

Bibliography

"Tutorial To Implement K-Nearest Neighbors in Python From Scratch - Machine Learning Mastery." *Machine Learning Mastery*. N.p., 12 Sept. 2014. Web. 02 Dec. 2015.

"1.6. Nearest Neighbors¶." *1.6. Nearest Neighbors — Scikit-learn 0.17 Documentation*. N.p., n.d. Web. 02 Dec. 2015.

Xindong, Wu J., Vipin Kumar, and Joydeep Ghosh. "SURVEY PAPER Top 10 Algorithms in Data Mining." *Top 10 Algorithms in Data Mining* (n.d.): n. pag.*Www.cs.uvm.edu*. Springer, 4 Dec. 2007. Web. 1 Dec. 2015.