What's for Dinner? Recommendations in Online Grocery Shopping

Alan Flores-Lopez (alanf94), Skip Perry (vperry), Poorvi Bhargava (poorvib)

CS229 Final Project Report - Stanford University

Abstract: We present an approach for online grocery shopping recommendation, framed as a binary classification problem. We describe the methods we used to extract features from a recent data set by Instacart, and we compare the results of our recommendation experiments with those of traditional and enhanced market basket analysis approaches. We use approximate recommendation precision to measure the success of our recommender, and ROC AUC values to evaluate some of our underlying classifiers. Although our recommendations were better than the market basket analysis baseline, our best models learned primarily the purchase frequency of products.

1. Introduction

One of the major predictive questions facing retailers is which products consumers will purchase and when. We explore a rich, recent data set of 3 million Instacart orders made by about 200,000 users[8], supplemented with nutritional data from Open Food Facts[12]. The Instacart data includes information about the date and time orders were placed, in what order items were placed in the user's shopping cart, and even which virtual departments and aisles the purchased products belong to. Instacart has already answered questions on which products tend to be re-purchased and at what time of day, but some areas remain unexplored:

- *Market basket analysis:* Which items are often included in the same cart? It's clear that chips and salsa go together, but are there less obvious connections hidden in the data?
- *Shopper types:* Since we are given no information about users, can we uncover groups that behave similarly? Which features allow us to deduce clear user clusters?

We focus our work on the problem of recommendation:

- ***Recommendation system***: Given that a user has placed products $P_1, \ldots, P_{n-1}$ in their shopping cart, what are the next best k products $P_n, \ldots, P_{n+k-1}$ to advertise? Can shopper types or market basket analysis inform the recommendation?

2. Related Work

Recommendation systems have been widely and fruitfully studied in recent years. The 2009 Netflix Prize collaborative filtering competition is the most famous of many studies examining the best way to recommend products and services to consumers. Grocery recommendations are a tougher nut to crack. Unlike Netflix, which has a limited number of movies and TV shows to credibly recommend, grocery shopping presents a challenge in its high sparsity. A grocery store stocks thousands of items, yet most people only buy a handful of them at a time. Analysis of this question has included methods like basket-sensitive random walks[9] and SVD approximations[13] to recommend items to consumers. Others have delved more into the theory, hoping to lay out a process that incorporates both product and user features into a recommendation process. [14]

A problem specific to our methods, uneven class label size in binary classification, has been studied as well. Work has been done in gauging the benefit of weighing to offset the class label skew[7], a method we utilize via Sci-kit's `class_weight` argument. The results are not always promising or consistent, with at least one paper showing that improving classification scores on the underrepresented class comes at the expense of overall classification. Other work has acknowledged the difficulty in measuring correctness with class skew, recommending data and domain-specific approaches for evaluation[4]. The inconclusiveness of research related to class label skew is reflected in a recent survey[3] and even in our own findings.

3. Dataset

Our data set, made freely available by Instacart[8], contained the following information:

- **Orders:** 3 million orders. Includes user ID, product bought, order of purchase, time of order, and whether each product was previously purchased by that user.
- **Users:** About 200,000. Only user ID's given. User-specific features were derived by us for our models.
- **Products:** About 50,000 products available for purchase from Instacart, along with aisle and department identifiers.[1] We supplemented this data with Open Food Facts' Nutri-Score[12], which rates the healthiness of food products, using a fuzzy match on product name with the Instacart data.

3.1. Generating Training, Validation, and Test Sets for Each Model

3.1.1. (Smart) Baseline: Market Basket Analysis Dataset Generation

A random subset of 500,000 orders was used to generate a "common order" list. This was a computationally intense but conceptually simple process for counting the number of times products were ordered, along with the number of times pairs of products appeared in the same cart.

3.1.2. Classification-Based Recommendation Models Dataset Generation

As previously mentioned, we cast the recommendation process as a binary classification problem. From Instacart's raw information, we constructed a data set of the following form:

$$T = \{(\phi(p,u,o), y^{p,u,o}) : p \in P_{rec}, u \in U, o \in O_u^*\} \tag{1}$$

For a selected amount of unique users in the Instacart data U, and for a set of items from which to recommend P_{rec}[2], we constructed a meaningful set of sub-carts related to that user, O_u^*.[3] The extraction of features $\phi(p,u,o)$ for each recommendation, user, and partial cart tuple is described in the next section. Lastly, to match the structure of our recommendations, the value of $y^{p,u,o}$ was chosen as follows:

$$y^{p,u,o} = \begin{cases} 1, & \exists o' \in O_u^* \text{ s.t. } o \text{ is a prefix of } o' \text{ and } p \text{ is the next item in the cart with } o' \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

We built a dataset of this format using 116,990,000 examples (116,897,206 zero-labeled and 92,794 nonzero-labeled). Splits for the training, validation, and test sets were chosen specific to each model (depending on whether batch implementation was used), and are summarized in the results section.

4. Feature Extraction

We used various attributes for defining $\phi(p,u,o)$ in the generation of our training set. In addition, we conducted user clustering and word2vec-based product and cart embeddings to develop additional features, described in more detail below. Forward search wrapper model was used to select features from the following:

1) Cart-Based: number of products in the cart, last element placed in cart, cart embeddings, one-hot representation of P_{rec} items within cart.
2) Product-Based: nutrition score, aisle number, department number, food item or not, product embeddings.
3) User-Based: which cluster did the user belong to, average nutrition score, most common time of day, most common day of week, likelihood of shopping in different departments.

4.1. User Clustering

We performed user clustering to categorize users based on their shopping behavior. K-means, agglomerative clustering and affinity propagation were applied to various combinations of features. Since no ground truth labels were available and since data points were visually found to belong to one large cluster or lay evenly on the feature space, extraction and interpretation of distinct clusters became difficult. Therefore, we used silhouette scores[4] to evaluate the relative performance of each algorithm. We chose to categorize the users into five clusters, which we directly used as a feature for our model.

[1]Note: The vast majority of products are ordered very rarely - 99.7% of products appear in less than 1% of orders.
[2]In our experiments we choose this to be the top 500 most popular products in the training data.
[3]For simplicity we used all of u's cart prefixes, although we considered other methods like cart n-grams as well.
[4]A method to interpret and validate clustering techniques, in which a higher score is assigned to dense and well separated clusters.

4.2. Product and Cart Embeddings

To capture the semantic meaning of the products within a cart, Google's Pre-Trained News Vectors [10] were used. Product embeddings were created by averaging the word vectors of each word in the product name[5]. Cart embeddings were constructed by averaging the product embeddings for every product in a cart.

5. Methods

5.1. (Smart) Baseline: Market Basket Analysis/Association Rules

Market basket analysis, also known as association rules in machine learning literature, is used widely in market research to uncover interactions between products. We used the following metrics:

- Support of A, $P(A)$, is the probability of a product A appearing in a user's cart. As described above, the vast majority of items for sale by Instacart are purchased very rarely; a marketer could do worse than to recommend the most popular items, especially when lacking user information.
- Confidence $A \rightarrow B$, $P(A \cap B)/P(A)$, measures the probability of a shopper purchasing Item B given that Item A is included in the order.
- Lift $A \rightarrow B$, $P(A \cap B)/(P(A) * P(B))$, is confidence $A \rightarrow B$ scaled by the support of B. [5] This is designed to find interactions including lower-probability items missed by the confidence rating.

We generated support scores for all products, along with confidence and lift scores for all pairs of items appearing in the same cart at least once, partitioned into user clustered and non-clustered data sets (clustering described in more detail below). Parameter tuning included setting a minimum number of orders in order for a product to appear in a recommendation, using different scaling factors (taking $P(B)$ to powers between 0 and 1), and testing whether user clustering improved recommendations.[6]

5.2. Logistic Regression and Linear SVM Classifier

As previously mentioned, another way to think of this problem is as a binary classification task, with models that classify possible recommendations into 'good' (1) or 'bad' (0) recommendations. The top ten candidates are the recommendations with the highest confidence of a 1 label.

The first methods to make use of this process used Sci-Kit learn's `LogisticRegression` and `SGDClassifier` with hinge loss. Both of these algorithms tune parameters θ to minimize a certain loss function defined over our training set T:

$$J(\theta) = \sum_{p,u,o} \text{Loss-}\{\text{hinge, logistic}\}(x^{(p,u,o)}, y^{(p,u,o)}) \tag{3}$$

The parameters are then used to predict y labels for future examples. Because our training data T contained far more 0 labels than 1 labels (a ratio of about 1000 to 1), we used the `class_weight` argument option to weight up positive training examples about 1000 times more than negative ones. To gauge confidence for 1 labels, we use `predict_proba` in logistic regression to get a soft-label value between 0 and 1. For the linear SVM, we use the signed distance to the separating hyperplane as our metric. In order to train on very large training sets (several million examples), we operated on batches of examples small enough to load into memory, making multiple calls to the `partial_fit` function.[7]

5.3. Weighted Linear SVM

We attempted a version of weighted classification by minimizing the loss with respect to query-specific weights governed by cart similarity:

$$\sum_{p,u,o} w^{(o)} \text{Loss-hinge}(x^{(p,u,o)}, y^{(p,u,o)}) \; \Big| \; w^{(o)} = \exp\left(\frac{-(x^{(i)}.\text{cart} - x.\text{cart})^2}{2\tau^2}\right) \tag{4}$$

[5]In our case this could more accurately be described as $f(P[B])$ - due to extreme sparsity in the data, we scaled up $P(B)$ toward 1 to avoid numerically and statistically problematic denominators.

[6]Our results table uses the best parameters from this dev process: 0 floor, 0.1 scaling, clustering for confidence but not for lift.

[7]`partial_fit` is not available for `LogisticRegression`, so our training set sizes in that case were restricted to about 500,000.

This method gives more importance to training examples that have carts similar to the cart of interest, where the cart distance is defined to be the euclidean distance between the cart embedding representations.

5.4. Neural Network

Finally, we used Sci-Kit's `MLPClassifier` neural network. A neural network operates similarly to the previous approaches, tuning parameters for future predictions by minimizing a global loss over T, but its architecture involves many more computational steps at intermediary nodes, or neurons. In practice, we found a ReLU activation to work better than a logistic activation, so we only include results for the former.

6. Experiments and Discussion

6.1. Testing and Evaluation Metrics

We posit the following recommendation scheme, modeled off a realistic shopping process: When a user begins shopping, we offer 10 recommendations randomly selected from the 500 most popular items. After items are added to the user's cart, we offer recommendations of the same size based on predictions from different models. We used two different validation metrics:

- **Exact match**: Count a recommendation as a success if the next item appears in the recommendation set. This is a strict standard where ”banana” is different from ”organic banana.”
- **Approximate match**: Count a recommendation as a success if the cosine similarity between the product embedding of the actual next item and that of any of the 10 recommended items is 0.88 or higher (normalized to a 0-1 range). For example:
  - Similar Products (counted as a success): 'spinach' and 'baby spinach' (0.93), 'organic apple' and 'apple' (0.91), 'onions' and 'bag of onions' (0.90), 'strawberry' and 'strawberries' (0.90)
  - Dissimilar Products: 'beer' and 'wine' (0.81), 'spinach' and 'kale' (0.79), 'cupcake' and 'cereal' (0.69)

Because our recommendation system relies on a classification problem with highly uneven class labels, we also used Receiver Operating Characteristic (ROC) curves measuring the ratio between false positives and true positives to evaluate some of our best-performing models. The area under the curve (AUC) of the ROC curve has been claimed to be a good summary statistic for machine learning algorithms[2], with an AUC of 0.5 roughly meaning that the classifier picks at random, while a value of 1 means the classifier is always correct.

6.2. Results and Discussion

TABLE I
TRAIN AND TEST RECOMMENDATION PRECISIONS AND ROC AUC'S

Model (Training/ Test Set Size)	Exact (Train)	Exact (Test)	Approx. (Train)	Approx. (Test)	ROC AUC (Unweighted)	ROC AUC (w/ Class Weights)
Support (500k, 1k)	0.9%	1.0%	10.1%	10.2%	-	-
Confidence (500k, 1k)	9.6%	9.3%	21.9%	21.1%	-	-
Lift (500k, 1k)	9.9%	9.4%	22.4%	21.6%	-	-
Logistic (500k, 1k)	16.8%	10.3%	36.1%	26.0%	0.64	0.65
Weighted SVM (1000k, 100)	-	1.9%	-	28.2%	-	-
SVM (5000k, 1k)	8.0%	6.5%	27.8%	28.4%	0.53	0.65
Neural Net / ReLU (5000k, 1k)	12.8%	12.2%	30.4%	30.2%	0.68	-

Our baseline metrics resulted in success rates of about 9% and 21% in exact and approximate matches, respectively. (By contrast, offering a random selection of 10 popular items had only a 1% success rate in exact matches.) Clustering helped our confidence ratings, bringing them up to the point where they nearly matched the richer lift recommendations, but it seemed to introduce too much variance in the data to help improve lift recommendations.

Our other models tended to improve on these results, resulting in between 10-12% success rates in exact match and 26-30% in approximate match. The neural net was the clear winner in both metrics.

An example recommendation from the neural network demonstrates some of the typical positive and negative attributes of our model. On the positive side, the recommendation set found the right answer (twice, in the approximate case) out of 50,000 possible next items, and put together a sensible and plausible set of recommendations.

That said, our models sometimes seemed to learn primarily the frequency of items in the training data. Indeed, 'Organic Banana' was in most recommendations given by the neural net, regardless of input features. It is likely that our problem is not related to overfitting, since we used regularization and train/test errors were quite close in most of our experiments (see Table I & Fig. 1). A more plausible explanation is that the quality of our underlying classifier directly influences the quality of the predictions of our recommender. While the 0.68 AUC value for our neural network model is well above the 0.50 level, it is still not in the desirable 0.8-1.0 range. We speculate that a model with higher AUC would have improved recommendation performance.

TABLE II
SAMPLE RECOMMENDATION (FROM NEURAL NET)

Items	Recommendations
In cart: Banana, Organic Cucumber, Dark Chocolate, Feta Cheese Crumbles, Coconut Fruit Bars, Organic Spring Mix, Peach Pear Yogurt	Strawberries Lemon Hummus Guacamole Hass Avocado Organic Banana Organic Strawberries Broccoli Crown Baby Arugula
Actual next purchase: Organic Strawberries	Hass Avocados Original Hummus

SVMs performed relatively well on the approximate metric (28% success), but hopes that a weighted SVM would solve the problem of only recommending popular items did not pan out. This may have been because neural nets automatically learn feature interactions, whereas the class imbalance and large feature vector dimension of the training data may not have been amenable to SVMs. The need to use smaller training sets due to high computational complexity (since a new classifier needs to be trained from scratch for each possible recommendation set) may also have had a negative impact. We suspect the computational requirements may be infeasible for most applications in this area.

Lastly, using class weightings improved the linear SVM model (moving AUC from .53 to .65), but had almost no effect on the logistic regression classifier.

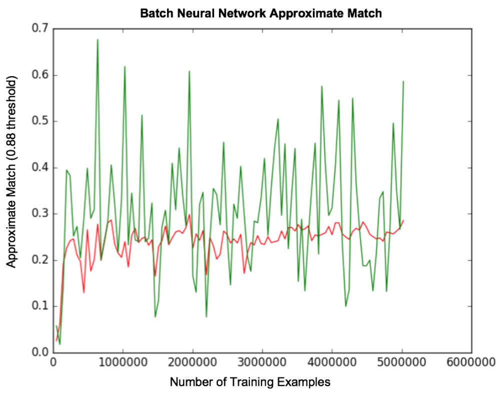

Fig. 1. Neural Network Learning Curve: Green shows approximate accuracy on the train set and red shows the same quantity on the dev set

7. Conclusions and Next Steps

In this report we have presented a form of grocery cart recommendation based on a binary classification problem. We discussed our methods, including our feature extraction work, and how we evaluated both our recommendation system and our underlying classification problem in the face of uneven class labels with ROC AUC scores. The best of our models seemed to learn mostly the frequency of the products in the data, although this model also had the highest AUC value.

Our future work would consist first of evaluating whether our training data indeed captures the information needed for an effective recommendation system; it could be that generating approximate training data where $y^{p,u,o} \in [0, 1]$ works better, for example. In addition, feature reduction techniques like PCA and LDA should be implemented. We would then try to obtain more information about the link between AUC scores and recommender performance. Given a strong link, we could attempt various methods to drive up AUC, starting with feature reduction and kernels. Given a weak link, or if it became too difficult to improve the AUC metric, we could attempt different recommendation schemes altogether, such as graph-based methods or hidden Markov models. Lastly, we could try to frame the problem as a time series question and try a Long Short Term Memory model, for example.

8. Contributions

- Poorvi Bhargava: Data exploration, user clustering, product embeddings and similarity, feature extraction, logistic regression, SVM, neural network, embeddings-based error analysis
- Alan Flores-Lopez: Data exploration, nutrition scoring fuzzy join, generating training set for ML methods, setting up experiment infrastructure, logistic regression, SVM, weighted linear SVM, neural network, ROC AUC scores and analysis
- Skip Perry: Data exploration, generating training set for market basket analysis, market basket analysis, association rules, user cluster testing, error analysis

References

[1] Aggarwal, Charu C. Recommender Systems: The Textbook. , 2016. Internet resource.

[2] Bradley, Andrew P. "The use of the area under the ROC curve in the evaluation of machine learning algorithms." Pattern recognition 30.7 (1997): 1145-1159.

[3] Branco, Paula, Luis Torgo, and Rita Ribeiro. "A survey of predictive modelling under imbalanced distributions." arXiv preprint arXiv:1505.01658 (2015).

[4] Daskalaki, Sophia, Ioannis Kopanas, and Nikolaos Avouris. "Evaluation of classifiers for an uneven class distribution problem." Applied artificial intelligence 20.5 (2006): 381-417.

[5] De Booma, C., Van Canneyta, S., Demeestera, T., & Dhoedta, B. (2016). Representation learning for very short texts using weighted word embedding aggregation. https://arxiv.org/pdf/1607.00570.pdf

[6] Hastie, T., Tibshirani, R., & Friedman, J. H. (2001). The elements of statistical learning: Data mining, inference, and prediction. New York: Springer.

[7] Huang, Yi-Min, and Shu-Xin Du. "Weighted support vector machine for classification with uneven training class sizes." Machine Learning and Cybernetics, 2005. Proceedings of 2005 International Conference on. Vol. 7. IEEE, 2005.

[8] Instacart (2017). The Instacart Online Grocery Shopping Dataset 2017. Retrieved from https://www.instacart.com/datasets/grocery-shopping-2017

[9] Li, Ming & Dias, Malcolm & Jarman, Ian & El-Deredy, Wael & Lisboa, Paulo. (2009). Grocery shopping recommendations based on basket-sensitive random walk. Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. 1215-1224. https://www.researchgate.net/publication/221653590_Grocery_shopping_recommendations_based_on_basket-sensitive_random_walk

[10] McCormick, C. (2016). Google's trained Word2Vec model in Python. http://mccormickml.com/2016/04/12/googles-pretrained-word2vec-model-in-python/

[11] Mild A., Reutterer T. (2001) Collaborative Filtering Methods for Binary Market Basket Data Analysis. In: Liu J., Yuen P.C., Li C., Ng J., Ishida T. (eds) Active Media Technology. AMT 2001. Lecture Notes in Computer Science, vol 2252. Springer, Berlin, Heidelberg. https://link.springer.com/content/pdf/10.1007/3-540-45336-9_35.pdf

[12] Open Food Facts (2017). Retrieved from https://world.openfoodfacts.org/

[13] Sano, Natsuki & Machino, Natsumi & Yada, Katsutoshi & Suzuki, Tomomichi. (2015). Recommendation System for Grocery Store Considering Data Sparsity. Procedia Computer Science. 60. 1406-1413. 10.1016/j.procs.2015.08.216. https://www.researchgate.net/publication/282831785_Recommendation_System_for_Grocery_Store_Considering_Data_Sparsity

[14] Yi-Jing Wu & Wei-Guang Teng (2011). An enhanced recommendation scheme for online grocery shopping. 2011 IEEE 15th International Symposium on Consumer Electronics (ISCE). http://ieeexplore.ieee.org/document/5973860/