ThankBeer: A Beer Recommendation Engine Final Report

Robert Wilson
Department of Electrical Engineering
Stanford University
Email: rwilson4@stanford.edu

***Abstract*—We discuss a beer recommendation engine that predicts whether a user has had a given beer as well as the rating the user will assign that beer based on the beers the user has had and the assigned ratings. k-means clustering is used to group similar users for both prediction problems. This framework may be valuable to bars or breweries trying to learn the preferences of their demographic, to consumers wondering what beer to order next, or to beer judges trying to objectively assess quality despite subjective preferences.**

I. Introduction

The beer drinkers of today face an unprecedented variety of options. This variety poses its own problem for novices and experts alike: what should I drink next? Few things in this world surpass the disappointment of drinking bad beer, yet given the subjectivity of taste, to whom should an unsure patron turn for advice on what to order? Knowing the preferences of the customer can aid a bartender in recommending a drink, but a properly trained machine learning algorithm has the potential to outperform even the beer connoisseur in this task.

Before making a recommendation, a human or machine must first learn the preferences of the patron. This learning process inevitably follows some combination of two approaches, focusing on beers or qualities. In the former, we ask the patron their favorite beers. In the latter, we ask what kinds of beers (hoppy, sour, etc.) the patron likes. These questions address *what* a patron likes and *why*. Both approaches provide insight into consumer behavior and the qualities of a great product, and may be used in conjunction to make better recommendations.

An obvious starting point is to ask the user of a machine learning-based system to provide a list of their favorite beers. A search functionality included in our system permits the user to do just this. Casual users, however, may have trouble freely recalling their favorite beers, so a prompt-based approach may help users remember beers they had forgotten about—provided we can predict which beers someone has had better than they can remember. There is an additional benefit to being able to predict whether a user has had a beer, or at the very least has access to it. If we recommend, e.g. a beer only available in Germany to a person in California, the recommendation is not valuable regardless of its validity. For someone to have had a beer, it necessarily must have been available to them, so predicting availability is correlated with predicting whether a user has actually tried a beer. Even if a person has access to a beer, they may have chosen not to order it due to personal preferences, so predicting whether a user has had a beer has some overlap with predicting their opinion of it.

The rating assigned to a beer depends on the objective quality of the beer, colored by the preferences of the drinker. Averaging the ratings of many may converge to a measure of the objective quality of the beer but does not account for individual taste. Thus any impersonal measure of the quality of a beer is of limited utility. Ideally we should ask only those individuals who like the same beers we do to recommend beers; clustering users based on preferences permits just this.

Gathering data is often a non-trivial task in designing a machine learning algorithm. We set up www.ThankBeer.com to solicit ratings and test predictions. As of this report, we have roughly thirty users and nearly two thousand data points on which to test the algorithms discussed here. Users are presented with a series of beers to rate so their preferences may be identified. They may search for specific beers or breweries (from a database of roughly five thousand beers mostly from the United States[1]). They may also search for a bar, leading to a beer menu which is sorted according to the personalized preferences of the viewer. Sorting beer menus based on personal preferences is one of the major goals of this project.

II. Predicting Past Purchases

For the purposes of predicting past purchases, we define the experience matrix R having elements $r_{ij} \in \{-1, 0, +1\}$ where $r_{ij} = -1$ means user i has told us they have never had beer j, $r_{ij} = +1$ means user i has told us they *have* had beer j, and $r_{ij} = 0$ means we do not know whether the user has had this beer. Of course r_{ij} really is either -1 or $+1$, the goal of this section is to estimate it. Let us first discuss how we initially predicted this, which will form a convenient benchmark against which we may measure the success of more advanced algorithms.

In order to begin gathering data as quickly as possible, we initially implemented a correlation-based approach to predicting whether a user i has had a beer. For appropriate weights $\ell_{ik} \in [-1, 1]$ measuring the correlation of users i and k, we

[1]Courtesy: www.beerme.com

predict

$$\hat{r}_{ij} = \text{sign}\left(\frac{\sum_{k\neq i} \ell_{ik} r_{kj}}{\sum_{k\neq i} |r_{kj}|}\right)$$

In determining the correlation coefficient ℓ_{ik}, we wanted to capture the idea that two users having the same beer says more than two users not having had a beer (given the typical user has had relatively few beers). Thus we defined

$$r'_{ij} = \begin{cases} +2 & r_{ij} = +1 \\ r_{ij} & \text{otherwise} \end{cases}$$

and using the Pearson correlation for ℓ_{ik}:

$$\ell_{ik} = \frac{\sum_{j\in\mathcal{B}}(r'_{ij} - \bar{r}'_i)(r'_{kj} - \bar{r}'_k)}{\sqrt{\sum_{j\in\mathcal{B}}(r'_{ij} - \bar{r}'_i)^2}\sqrt{\sum_{j\in\mathcal{B}}(r'_{kj} - \bar{r}'_k)^2}}$$

with bar denoting average over the absent index. Sums (including in computing the unshown averages) are over only those beers for which r_{ij} and r_{kj} are actually known. In the case these two users have no beers in common, their correlation is undefined.

By considering all data gathered to date, we can apply the Leave One Out Cross Validation process to measure the success of this benchmark algorithm. We remove a single datum, predict the label for that datum based on the residual data, and count the errors. Because we would like to recommend beers the user can easily access without flying around the world, we are concerned primarily with the rate of false positives (cases where we thought a user had had a beer, but in fact had not). This is most important when requesting ratings from the user for the purposes of learning preferences. The user can only rate beers they have had; it is frustrating only to be presented beers the user has never heard of. Thus when learning preferences, a natural strategy is to present the beer we are most confident the user has had, but not yet rated. The error of the algorithm, then, is the number of false positives divided by the total number of positives. Our benchmark algorithm has an error of 49%. This seemingly large error is partially because only 25% of our data points are positive, which in turn reflects the failure of our algorithm to present the user with beers they have had.

Let's try a different approach. We model the event that user i has had beer j as a Bernoulli random variable with parameter ϕ_j. That is, all users are considered the same, and r_{ij} is simply drawn from this distribution, taking on value $+1$ with probability ϕ_j and -1 with probability $(1-\phi_j)$. We can estimate ϕ_j as

$$\hat{\phi}_j = \frac{\sum_i \mathbb{1}\{r_{ij} = +1\}}{\sum_i \mathbb{1}\{r_{ij} \neq 0\}}$$

and predict $\hat{r}_{ij} = 1$ if $\hat{\phi}_j > 0.5$ and -1 otherwise. Simple enough, but the resulting error is 47%. Let's adopt a clustering approach, modeling $r_{ij} \sim \text{Bern}(\phi_{c(i)j})$ with user i in cluster $c(i) \in \{1, \ldots, k\}$. The updated estimate

$$\hat{\phi}_{c(i)j} = \frac{\sum_\ell \mathbb{1}\{c(\ell) = c(i)\}\mathbb{1}\{r_{\ell j} = +1\}}{\sum_\ell \mathbb{1}\{c(\ell) = c(i)\}\mathbb{1}\{r_{\ell j} \neq 0\}}$$

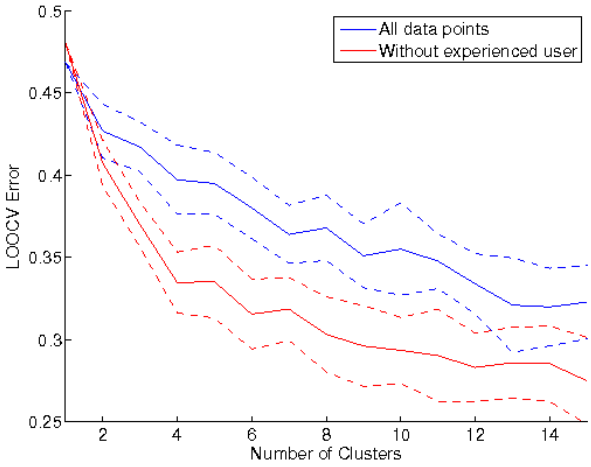

Fig. 1. Error with increasing number of clusters

considers only those users in the same cluster as user i. While user ratings are restricted to be ± 1, cluster centroids are allowed to take on any value, being the average of the user ratings in that cluster. So $c \in \mathbb{R}^{k\times|\mathcal{B}|}$ (where $|\mathcal{B}|$ is the number of beers) has entries

$$c_{ij} = \frac{\sum_\ell r_{\ell j}\mathbb{1}\{c(\ell) = i\}}{\sum_\ell \mathbb{1}\{c(\ell) = i\}}$$

We use the L_1 norm in determining the cluster for a user ℓ:

$$c(\ell) = \arg\min_i \sum_j |r_{\ell j} - c_{ij}|$$

summing over only those beers for which we know $r_{\ell j}$. We use the standard approach, initializing centroids randomly, and iteratively assigning users to clusters and redetermining centroids until convergence. Fig. 1 shows the mean error as the number of clusters increased. Because of the random initialization, the error varied from run to run, so twenty trials were considered for each data point shown in the figure, with the associated standard deviation shown as dashed lines about the mean. This type of graph is displayed throughout this paper; dashed lines always represent one standard deviation above and below the mean, shown as a solid line.

We found something interesting: the error actually diminished if we ignored one of the users. Fig. 1 shows in red the LOOCV error when ignoring said user. This user was unusually experienced; we hypothesize because we have relatively few users, he unduly influenced the clustering. Since he has had more beers than anyone else in this study only one other person was in his cluster (the author, actually) except when the number of clusters was low. Thus any beer he has had the author hasn't led to a false positive. This shows the importance of having numerous persons in each cluster.

We must decide the number of clusters to use. For the purposes of this report, and in light of the number of users, we simply looped over all the options, capping at half the users. In light of the aforementioned finding, fewer clusters having

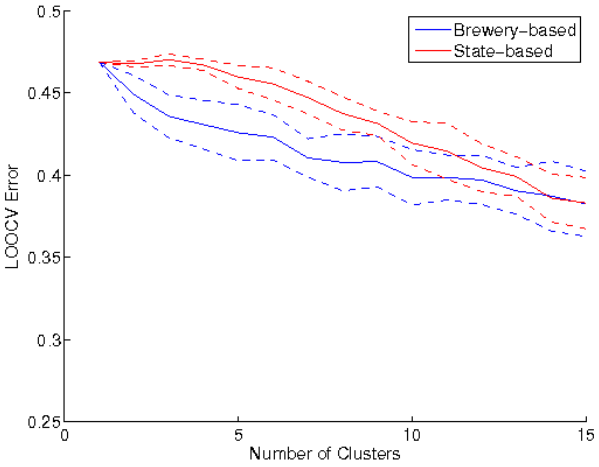

Fig. 2. Brewery and state-based clustering error

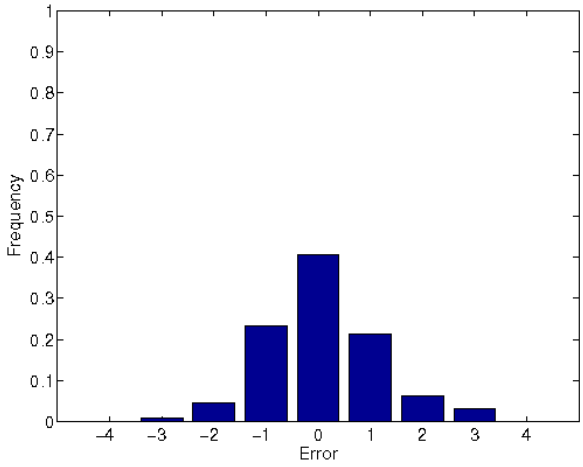

Fig. 3. Prediction error

many users are preferable for robustness. However, our data shows the error continued to decrease as the number of clusters was increased. Intuition sheds some light on the situation. We conjecture based on personal experience there are two types of beer drinkers: aficionados who have tried many of the beers available in their area, and novices, who do not actively seek out good beers and only drink what is available at the typical bar or party. We conjecture further that novices throughout the United States will have had mostly the same beers, while aficionados will exhibit geographical diversity owing to the small distribution region of most craft breweries. Thus the number of clusters, intuitively, should be one (for the novices) plus the number of major brewing centers represented in the database. At this point there is insufficient data to confirm or reject this hypothesis.

One method to make better use of a small data set is to project the features to a smaller space. The problems we encountered are partially due to few people having had a particular beer. Yet in predicting availability, it seems reasonable to conjecture that if a person has access to one beer from a brewery, they likely have access to others. While the flagship beer of a brewery tends to have a wider distribution region than smaller batches such as seasonal beers, it seems reasonable to base our clustering algorithm on breweries instead of beers: if two users have had beers from the same brewery, they are similar. Fig. 2 shows the resulting LOOCV error. This brewery-based strategy did not perform as well as the original k-means approach; clustering based on the city or state of the brewery did still worse. New ideas are needed to make further progress.

III. Predicting Rating

A user may rate a beer out of five stars. The prediction strategy initially implemented (again in the interest of getting something working as quickly as possible) mirrors the correlation-based approach that formed our benchmark strategy for predicting whether a user had had a beer. For this section, $r_{ij} \in \{0, 1, \ldots, 5\}$ represents the rating assigned by user i to beer j, with 0 denoting the user has not rated this beer (perhaps because they have not had it). We would like to predict how a user would rate a beer if they had it.

The correlation-based strategy predicts a weighted average of the ratings other users have assigned the beer under consideration. The rating profile for a user i is normalized according to:

$$r'_{ij} = \frac{r_{ij} - \bar{r}_i}{\sqrt{\sum_j (r_{ij} - \bar{r}_i)^2}} \tag{1}$$

The normalized predicted rating for user i is

$$\hat{r}'_{ij} = \frac{\sum_{k \neq i} \rho_{ik} r'_{kj} \mathbb{1}\{r_{kj} > 0\}}{\sum_{k \neq i} \mathbb{1}\{r_{kj} > 0\}}$$

where ρ_{ik} is the Pearson correlation for the ratings of users i and k. The unnormalized predicted rating is found by inverting Equation 1 and then rounding to the nearest integer. (Rounding was not initially used, but is useful to compare against the multinomial distribution-based methods discussed below.) Via the same Leave One Out Cross Validation procedure discussed above, we may predict the rating for a single datum using the remaining data and compare against the actual value. The errors, defined as the difference between prediction and reality, were distributed as shown in the histogram of Fig. 3. The approximate symmetry of the histogram shows the strategy overrates as often as it underrates. The average of the absolute value of the errors was 0.78.

We can model the rating as being drawn from a multinomial distribution where $r_{ij} = k$ with probability ϕ_{jk}, $k = 1, \ldots, 5$ ($\sum_{k=1}^{5} \phi_{jk} = 1$ for all beers j), which ignores the individual preferences of the users. We estimate

$$\hat{\phi}_{jk} = \frac{\sum_\ell \mathbb{1}\{r_\ell = k\}}{\sum_\ell \mathbb{1}\{r_{\ell j} > 0\}}$$

The predicted rating is $\arg\max_k \hat{\phi}_{jk}$. Alternatively we could simply predict the average over all ratings and round at the end.

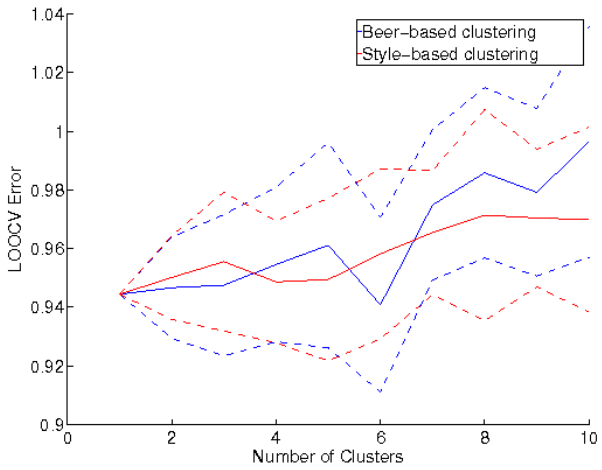

Fig. 4. Beer and style-based clustering prediction error

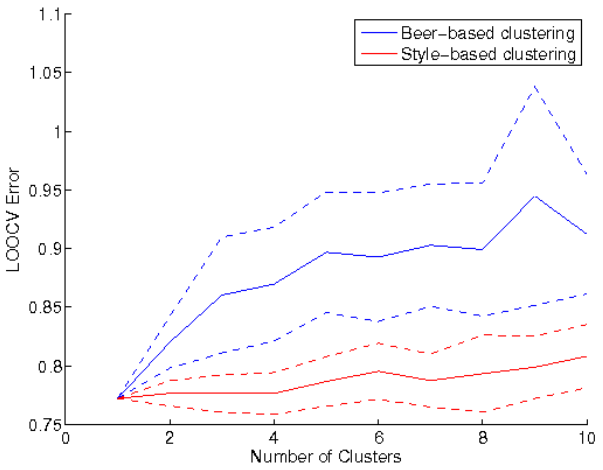

Fig. 5. Beer and style-based clustering prediction error based on normalized ratings

We can cluster users similar to before, but we anticipate, because the factors influencing taste are significantly more complicated than those involving access to a beer, the number of clusters needed will be much higher than before, resulting in fewer data points per cluster and a larger resulting error. We can apply a similar strategy as when clustering via breweries to reduce the state space. We clustered by style: users who like similar styles are similar. Styles (e.g. stout or IPA) were taken from the Beer Judge Certification Program handbook. Fig. 4 compares the performance of beer and style-based clustering. Here, the error actually increases with increasing number of clusters, perhaps due to insufficient data.

Normalizing the ratings improves the error rate, but not the trend, as shown in Fig. 5. Clearly these clustering strategies do not succeed in predicting the available data. Projecting onto still smaller spaces may help. The style-based clustering strategy neglected the relative similarities between styles. For example, a pale ale and an IPA are similar; the styles could be combined. Some styles are sufficiently uncommon they could be neglected altogether for the purposes of clustering. It must be remembered, however, that style popularity differs from region to region: styles common in Germany may be unusual in the United States.

At this point we must consider not only *what* the user likes, but also *why*. Users of www.ThankBeer.com can apply keywords like hoppy or sweet to beers and in the very near future they will be able to assign a degree, letting the community decide that one beer is hoppier than another. We can then project each beer onto this lower-dimension feature space (the number of relevant features is likely much less than the number of beers or styles) and perform the clustering there. This will hopefully result in far fewer clusters, giving more training examples per cluster and lower error. More importantly, it permits the users to tell us directly why they like or dislike the beers they do.

IV. Future Work

The plan laid out for predicting rating based on clustering in feature space is in progress. The best method for predicting user rating is still the correlation coefficient-based approach implemented initially. Although clustering reduced the error associated with predicting whether a user has had a beer, more work is needed since the error rate is still unacceptably high. Combining the two prediction problems may help since personal preferences influence whether a user has had a beer.

Modeling user ratings as a mixture of Gaussians instead of a multinomial distribution provides may provide more flexibility. We would remove the requirement that predicted rating be an integer. The rating would be normally distributed with mean and variance depending on the cluster. Due to the complexity of tastes, a neural network seems like a promising possibility, eliminating the clustering approach and modeling preferences as a continuous spectrum based on some combination of, e.g. sweetness and hoppiness as reported by the users.

One potentially valuable excursion would be to apply these methods to beer judging, which is sometimes criticized for emphasizing adherence to arbitrary style guidelines. This policy is intended to place beers of a common style on an equal footing and reduce the impact of the judges' personal preferences. In practice, beers that do not fit neatly into a particular category are punished, regardless of how enjoyable they are. The effect is to discourage creativity and the evolution of styles that led to the fairly recent explosion of craft brewing in the first place. A system that learns the personal preferences of judges permits them to rate beers as they see fit, knowing their biases can be corrected for and ratings combined to provide an assessment less sensitive to style guidelines.

Acknowledgment

The author would like to thank the early adopters of www.ThankBeer.com who donated ratings and put up with an admittedly klunky interface.