Using Chemical Data to Predict Wine Ratings

Eric Sebastian Soto

12-14-2012

Introduction

Somewhat uncommon for a computer science student, the author of this paper has the dream of someday becoming a world-class chef. But being an excited artificial intelligence student fan, he dreams of running a kitchen with the help of robotic machines. Thinking it would be cool to have a robotic sous chef someday, this paper hopes to begin the process of creating one.

With its latest movement of molecular gastronomy, our society continues to find more and more complex ways to concoct its edible creations, a trend that has resulted in food that is sometimes hard to recognize as being meant for human consumption. With this newly found complexity in food, there begs the question of what makes something so tasty, or what makes a person prefer one food over another. While, we can conjecture about just how exactly we taste food, as Russel and Norvig suggest, it may be better to find a model for taste that relies less on our own understanding of taste, and more on empirical data and any abstractions the data may help us concoct.

A Short Word on Taste

Our understanding of taste continues its constant evolution, as it remains poorly understood by modern science. One of the original models for taste was that our sense of taste came from our taste buds passing on information about which of the five common flavors- sweet, bitter, sour, salty and the recently christened umami, were present in our food. Now modern science has come to find that complicated interactions between protein receptors in taste cells maybe be responsible for our perception of taste. Scientists at UC San Diego found that the same pH protein receptors in the spinal cord were also found in our taste cells, helping us measure the acidity content of food to give us the perception of sourness and acid content [7]. Therefore, it seems a sound idea, that in any attempt to model human taste we seek to include such features in our feature space.

Wine

Since creating a model for taste seems a gargantuan and vaguely defined taste, we can first start with a way of creating a rating system. Such a rating system achieves what we are after with a model for taste, what tastes the best. Since machine learning tasks achieve the best results with a set of great data, wine's rating system is a fantastic place to start. Viticulture comes with a very storied, and tried tradition of ratings its wine. Hence, we can use wine to explore how to predict if something taste good.

Noting that wines vary by region and variety it would be wisest to choose a single type of wine and region. Hence we choose red wine from Portugal's Vinho Verde region, since it was readily available on the UCI machine learning data repository. Due to previous work done on this subject [1], we began the process by choosing a feature set that consists of fixed acidity, volatile acidity, citric acid, residual sugar, chlorides, free sulfur dioxide, total sulfur dioxide, density, pH, sulphites, and alcohol content. Variables related to the sourness of a wine such as fixed acidity, volatile acidity, citric acid, and pH should give us a good indication of how the acidity of these wines affects the taste. While it remains unknown how important or what effects sulfites have on a wine, we include suphate content, amount of free sulfur dioxide, and the total sulfur dioxide to give a measure of the sulphur content, which is a contentious part of the wine making and consumption process. It remains contentious because recent studies indicate that sulfites can lead to a negative experience to the wine drinking process, causing some people to feel sickness [2].

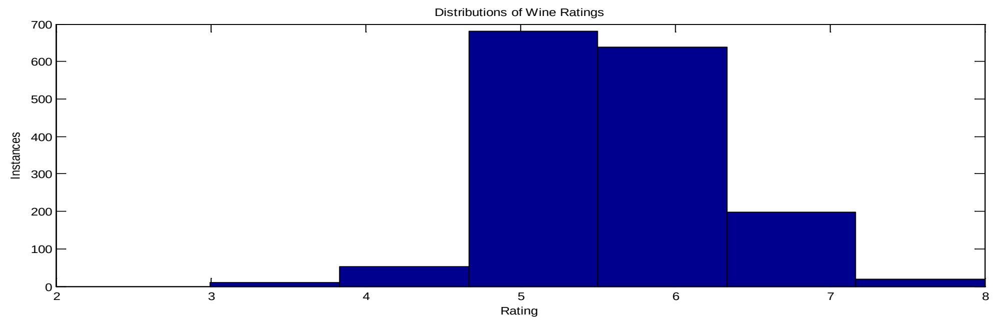
SVM and LDA

As a first attempt to classify our multi-class data, I used the “One-Against-All” variant of the Support Vector Machine. SVM’s have a very good history of being excellent binary classifiers by concentrating on maximizing the geometric margin, rather than the error in relation to all training examples. Using the liblinear library [3], we used the ‘-s 7’ flag when training the SVM model. According to liblinear documentation, the SVM would use the OVA implementation utilizing the logistic kernel. For each data point it makes a separate prediction for all the wine rating and an accompanying probability that represents how confident that prediction is. Once a prediction for each class has been made, the most likely prediction determined by the recorded probabilities gets chosen as the solution to the classification problem. The other implementation of a multiclass SVM from the liblinear library, written by Cramer and Singer, gets called by using the ‘s -4’. This implementation [4] uses a generalized notion of margins to create a compact quadratic optimization problem. The dual gets decomposed into multiple small optimization problems and uses this representation to create a multiclass SVM.

To model our data using Linear Disriminant Analysis, we used a library written by Will Drinnelll. LDA works very similarly to our multiclass SVM, since when classifying an example LDA calculates a value representing its attempt to match the example to a certain wine rating, and then elects the wine rating with the highest value to be its guess. These values come from how far into the space designated for a certain wine rating a data point is determined to be. In order to create a model representation of the data, the LDA divides the data so as to maximize the distance between classes. [5]

Examining the Data

One major difficulty in utilizing the SVM algorithm came from the shape of distribution for wine ratings, shown in the figure below.



The distributions appear to follow a normal distribution, which due to there being very few different ratings assigned leaves there being very few examples that receive high ratings or low ratings. Hence, we could feasibly achieve reasonably good accuracy by only concentrating on doing a good job classifying only the ratings that are in the middle, resulting in poor performance. We do not worry about too much about this when using LDA, since it takes into account a prior, which it takes to be distribution seen in the training examples. On the other hand, running an unweighted version of the SVM on the data results in terrible precision and recall results for the lower and higher wine ratings. Hence, it was decided to use a weighted implementation, where the weights would get manually adjusted. And as expected, by weighting the results we achieved much greater precision and recall results, especially with the OVA SVM. Using Crammer and Singer’s SVM implementation, however, we found that even using the same weights on different training sets would lead to widely varying skewed output distributions when the testing data was run though the SVM. These leads us to say the CS SVM is inappropriate to use with a skewed distribution, possibly because it may converge to local minima due to it trying to represent the dual with several smaller optimizations.

Perceptron

The previously described algorithms were used out of the box. On the other hand, a multiclass perceptron was written to help us classify the data, and support the previous results. The multiclass perceptron we created runs by having a different vector for each type of wine. In training, whenever we incorrectly classify an example we move the correct vector closer to the training example and the incorrect one further away [6, 9]. Ultimately, this multiclass perceptron returned some results that while worse than the SVM and LDA returned, were not so significantly worse we decided that any of the implementations were being used improperly.

Results

The following figures present our precision, recall and f-scores.

Perceptron

Wine Ratings	Precision	Recall	F-Score
3	0.1721	0.2514	0.2044
4	0.1454	0.1928	0.1658
5	0.0673	0.1667	0.0958
6	0.1498	0.1711	0.1598
7	0.0658	0.1667	0.0944
8	0.2184	0.2260	0.2222

One vs All Support Vector Machine

Wine Ratings	Precision	Recall	F-Score
3	0.2424	0.2130	0.2263
4	0.2642	0.2532	0.2586
5	0.2307	0.2069	0.2182
6	0.2651	0.2948	0.2791
7	0.1905	0.2107	0.2001
8	0.3210	0.3826	0.3491

Linear Discriminant Analysis

Wine Ratings	Precision	Recall	F-Score
3	0.2910	0.2870	0.2890
4	0.2837	0.3001	0.2917
5	0.3242	0.4197	0.3658

6	0.2831	0.2943	0.2886
7	0.2705	0.3093	0.2886
8	0.3350	0.4518	0.3847

The multiclass perceptron's strength seems to be in predicting wines with low or high ratings, rather than those in the middle. On the other hand, both the OVA and LDA seem to have an easier time classifying highly weighted wines. LDA recalls almost half of the wines with a maximum score. Furthermore, while LDA seemed to be better at both precision and recall across the board, it is interesting to note that the average error of the SVM when run for fifty trials was 52.06%, while when run the same number of trials the LDA resulted in an error of 52.25%. As mentioned before the multiclass perceptron performed only slightly worse giving an average error 58.86% when run for fifty trials.

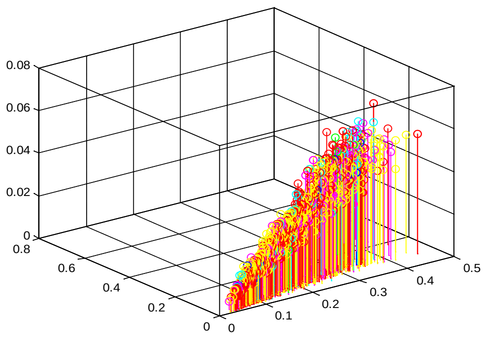
Feature Selection

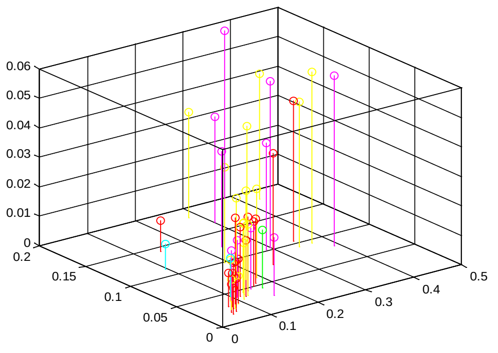
Running a feature selection algorithm on LDA to find which features of a wine contributed the most to a wine's rating gave the following order of importance: citric acid, free sulfur dioxide, alcohol, pH, volatile acidity, density, fixed acidity, sulphates, residual sugar, chlorides, and total sulfur dioxide. One of questions in the world of wine revolves around the presence of sulfur in wines. As a 2009 winter issue of Practical Winery and Vineyard Journal points out, smaller concentrations of sulfur than those found in wine can be identified by human taste and smell [8]. So it would make sense that our feature selection revealed that the amount of free sulfur dioxide would be one of the more important features when a machine tries to learn a model for wine rating. Other more influential features include citric acid content, alcohol levels, and pH level. Finding citric acid to carry more influence than pH is a surprising result, since citric acid, along with other acids is responsible controlling the pH level and the sourness of a wine. Hence, one would expect that the pH would sort of encapsulate the information provided by the citric acid. Therefore, there must be some information that the citric acid provides about taste beyond acidity.

Principal Component analysis

Looking into how the feature space affects the final rating of the wine a little bit more, and the fact that something like citric acid may carry more information about taste than just means that trying to simply group features together would be a difficult task. Hence, when running PCA and trying to couple all the sulfur related variables into one vector and all acid related vectors into another vector, and then adding the sugar and alcohol content as two separate features, the accuracy fell quite dramatically to an overall error mean of 64% when run for 50 trials using the logistic regression OVA SVM.

Reducing everything to three dimensions, by combining sugar and alcohol content, we found the resulting graph that attempts to give us some idea of how the data is mixed together and why the methods encountered such high error.





The figure on the left shows all of our 1200 plus examples and the rightmost figure shows 20 randomly selected examples. The data seems to be very intertwined based off of our features. It is possible that the features we choose do not provide much differentiation.

Conclusion

Looking at the SVM and LDA implementations and their achievement of 52% error, we find that it seems a difficult task to create a model of taste. Part of this error, must be coming from the fact that these wines are subjectively rated by people and are not scientifically grounded. Furthermore, both the SVM and LDA returned very similar results, suggesting our machine learning seem to be reaching some sort of convergence point for accuracy. A suggestion for improvement would be a different set of feature vectors. Since the most important feature was citric acid, it would be wise to include the amount of tartaric acid and malic acid, since they play much bigger roles in the taste of a wine than citric acid.

Citations

1. P. Cortez, A. Cerdeira, F. Almeida, T. Matos and J. Reis. Modeling wine preferences by data mining from physicochemical properties.
In Decision Support Systems, Elsevier, 47(4):547-553, 2009.

2. “A look at suhlphur in wine.” WINEMAG.CO.ZA, September 2, 2009.

3. R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin. LIBLINEAR: A Library for Large Linear Classification, Journal of Machine Learning Research 9(2008), 1871-1874. Software available at http://www.csie.ntu.edu.tw/~cjlin/liblinear

4. On the Algorithmic Implementation of Multiclass Kernel-based Vector Machines. Koby Crammer and Yoram Singer. Journal of Machine Learning Research, 2001

5. “Linear Discriminant Analysis (LDA).” Will Dwinnell. 12-11-2010 Web. 12-13-20112

6. Pieter Abbeel. “Lecture 21: Perceptrons.” UC Berkeley Web. 12-13-2012

7. "Biologists Discover How We Detect Sour Taste", *Science Daily*, August 24, 2006, retrieved 12 September 2010

8. Henderson, Pat. “Science behind this anti-microbial, anti-oxidant, wine additive.” Practical Winery and Vineyard Journal. Januray/February 2009. Magazine

9. “Kernel Perceptron in Python.” Mathieu’s log: Machine Learning, Data Mining, Natural Language Processing … Mathieu Blondel, 10/31/2010 Web. 12-13-12