Cupcakes versus Muffins

Avril Kenney
6.867

6 December 2011

Introduction

The purpose of this project is to determine whether there is a reliable difference between cupcakes and muffins, and if so, what this difference is. There have been various proposed definitions of the two categories and how to distinguish them,[1] ranging from specific ingredient ratios to the sounds that they make when thrown against a wall, but to my knowledge there has been no formal analysis.

In this project, I use several basic machine-learning classification techniques to distinguish cupcakes and muffins based on the amounts of their ingredients. The classifiers have reasonably good performance on a small data set.

Data

The data are vectors of ingredient amounts. Other factors, such as cooking time or methods of mixing things together, were not included. Toppings such as frosting and streusel were also not included.

The data were gathered using Google Recipe Search: the first 50 results for "cupcake" and the first 50 results for "muffin" were used, after excluding recipes that were not actually cupcake or muffin recipes (such as decorating instructions that start with pre-made cupcakes, or recipes for "cupcakes" made of lasagna). The original plan was to automate the data collection process by parsing ingredients directly from the HTML source of the recipes; however, it turned out that this would require a significant amount of language processing and other knowledge, which would make it prohibitively time-consuming to implement. Instead, the data were collected by hand.

For each recipe, the ingredients and their corresponding amounts were noted. Units were all converted to tablespoons, for consistency. There were often multiple slightly different terms used to describe the same or similar ingredients, so in these cases a single term was used to cover all of them (for example, "light brown sugar" was counted as "brown sugar"). When ingredients were listed as optional, they were not included. When multiple options for an ingredient were listed (for example, "butter or margarine"), the first one was used.

[1] For example, `http://www.cupcakeproject.com/2007/05/cupcake-vs-muffin-update.html`.

Normalization

Three different versions of the data were created, to ensure that the classifiers' results were not caused just by effects of scale.

In the first ("regular") version, all amounts in each data point (recipe) were normalized to make the total volume of that recipe equal to 100 tablespoons.

In the second ("scaled") version, the amount of each ingredient in each recipe (in the regular version of the data) was normalized by the median of all non-zero amounts of that ingredient.

In the third ("z-scored") version, the amount of each ingredient in each recipe was represented as its z-score, calculated using the mean and standard deviation of all amounts of that ingredient (in the regular version of the data).

Results were very similar for all versions of the data, so all reported results are for the regular data, unless otherwise noted.

Descriptive Statistics

There were a total of 100 recipes (data points) and 110 ingredients (features). Of these ingredients, the average amount was significantly greater in muffins for seven of them and significantly greater in cupcakes for eight of them, at the .05 significance level (uncorrected), using a two-tailed independent-samples t-test. These ingredients are shown in Table 1.

Ingredient	t	p	**Cupcake amount**	**Muffin amount**	**Cupcake** n	**Muffin** n
bananas	−3.17	0.00205	0(0)	4.54(10.1)	0	9
blueberries	−2.78	0.00656	0(0)	3.28(8.34)	0	7
baking powder	−2.35	0.0207	0.442(0.349)	0.619(0.399)	37	43
apples	−2.24	0.0272	0(0)	1.8(5.67)	0	5
oats	−2.23	0.0281	0(0)	1.81(5.73)	0	5
whole wheat flour	−2.15	0.0342	0.391(2.77)	2.71(7.12)	1	7
orange zest	−2.04	0.0441	0(0)	0.0623(0.216)	0	4
almond extract	2.12	0.0365	0.0216(0.072)	0(0)	5	0
vinegar	2.41	0.0178	0.162(0.373)	0.0231(0.163)	12	1
butter	2.63	0.01	7.06(6.04)	4.22(4.72)	32	25
food coloring	2.73	0.0075	0.436(1.13)	0(0)	8	0
cake flour	2.88	0.00492	8.62(15.2)	1.63(8.05)	13	2
vanilla	4.13	7.56×10^{-5}	0.458(0.367)	0.192(0.272)	40	22
cocoa	4.21	5.62×10^{-5}	2.23(3.75)	0(0)	18	0
sugar	7.69	1.18×10^{-11}	20.4(6.7)	9.3(7.66)	48	35

Table 1: Ingredients with amounts that were significantly different in cupcakes than in muffins (at the .05 significance level (uncorrected), using a two-tailed independent-samples t-test). Amounts are given as means (in tablespoons), with standard deviation in parentheses. The n columns give the number of recipes in which the amount of a given ingredient was greater than zero.

Clustering

For exploring the data, two clustering algorithms were implemented (in Python): EM (which used multivariate Gaussian distributions for the clusters) and k-means.

Because there are a large number of features relative to the number of data points, it seemed possibly useful to group the features (ingredients) into clusters, and treat each cluster as a single feature. To do this, each ingredient was represented as a vector of the mean amounts of all *other* ingredients in recipes in which that ingredient occurred. Both clustering algorithms were run on each version of the data, with various specified numbers of clusters. None of the clusterings they produced made any intuitive sense, so these results were not used.

The clustering algorithms were also run on the data points (all versions of the data), with the number of clusters specified as 2, to see whether they produced groups similar to the cupcake/muffin groups. They did not.

Classifiers

Four basic classifiers were implemented in Python.

Nearest Neighbors

The 1-nearest-neighbor classifier gives each test point the same label as the nearest training point to it, measured by Euclidean distance. In the case of k-nearest-neighbors, the label of each test point is chosen by a majority vote of the k nearest training points. (Results reported are for one-nearest-neighbor.) When there is a tie, a random label is chosen.

Perceptron

The perceptron is trained by repeatedly iterating over the training data and updating a vector of weights (as described in the lecture notes[2]). (An extra dimension is added to the data to allow for an intercept.) The update process is repeated until either all of the training data are classified correctly or a maximum number of iterations has been reached. The reported results use a maximum number of iterations equal to 1000.

Support Vector Machine

Support vector machines were implemented as described in the lecture notes,[3] using a linear (dot-product) kernel. Reported results use a value of 1.0 for the "magic constant" C, but results were very similar for other values.

[2] `http://courses.csail.mit.edu/6.867/lectures/Lecture_2_6.867.PDF`

[3] `http://courses.csail.mit.edu/6.867/lectures/Lecture_3_6.867.PDF`

AdaBoost

This classifier uses the AdaBoost boosting algorithm to create a classifier from a large set of decision stumps. (The implementation is based on the 6.034 notes.[4]) Because of the high dimensionality of the feature space, it was impractical to include a decision stump separating every pair of data points in every dimension, so the algorithm selects a random subset of them to build its classifier from.

Analysis of Classifiers

Cross-Validation Accuracy

The classifiers were tested using 100 runs of k-fold cross-validation, for $k = 5$ and $k = 2$. Results are shown in Table 2.

Classifier	**5-fold accuracy**	**2-fold accuracy**
Nearest Neighbors	0.831	0.805
Perceptron	0.824	0.799
Support Vector Machine	0.799	0.799
AdaBoost	0.804	0.788

Table 2: Accuracy for each of the classifiers in 100 runs of cross-validation.

To ensure that the above-chance accuracy rates were not simply caused by having a small number of data points in a high-dimensional space where any data would be nearly separable, cross-validation was also run using the same data points but with randomly-generated labels. With this input, all of the classifiers performed at chance levels (approximately 50% accuracy). This indicates that the success of the classifiers was because of the structure of the data rather than the flexibility of the space.

Comparing Predictions

As seen in the previous section, all of the classifiers had similar accuracy rates. Furthermore, all of them tended to be relatively consistent: across runs of cross-validation, most points would be classified correctly almost all of the time, and some would be classified incorrectly almost all of the time, but very few would be misclassified around half the time. However, not all of the classifiers tended to misclassify the same data points. Table 3 shows the correlations between each pair of classifiers.

The results of the classifiers were also compared with the results of a likelihood ratio test, which classified each data point as whichever category gave it a higher likelihood, where the categories were assumed to follow Gaussian distributions in each dimension, and the mean and variance were computed for each category not including the data point to be tested. For 36 of the points, the likelihood ratio was 1, usually because the likelihood was zero under

[4] `http://courses.csail.mit.edu/6.034f/ai3/boosting.pdf`

	Nearest Neighbors	Perceptron	Support Vector Machine	AdaBoost
Nearest Neighbors	0.997	0.475	0.477	0.209
Perceptron	-	0.995	0.906	0.313
Support Vector Machine	-	-	0.995	0.377
AdaBoost	-	-	-	0.990

Table 3: Correlation coefficients (Pearson's r) between classifiers on the number of correct classifications for each data point in 100 runs of 5-fold cross-validation. All correlations are significant at the .05 significance level.

both categories' distributions, for example because the recipe contained an ingredient that only occurred in zero or one other recipes. Of the other 64, 55 were correctly classified.

It might be expected that points with very high or very low likelihood ratios would be more typical examples of their categories and therefore be more often classified correctly. Interestingly, however, there was not much correlation between the likelihood ratio of each point and the performance of the classifiers on that point. A plot is shown in Figure 1.

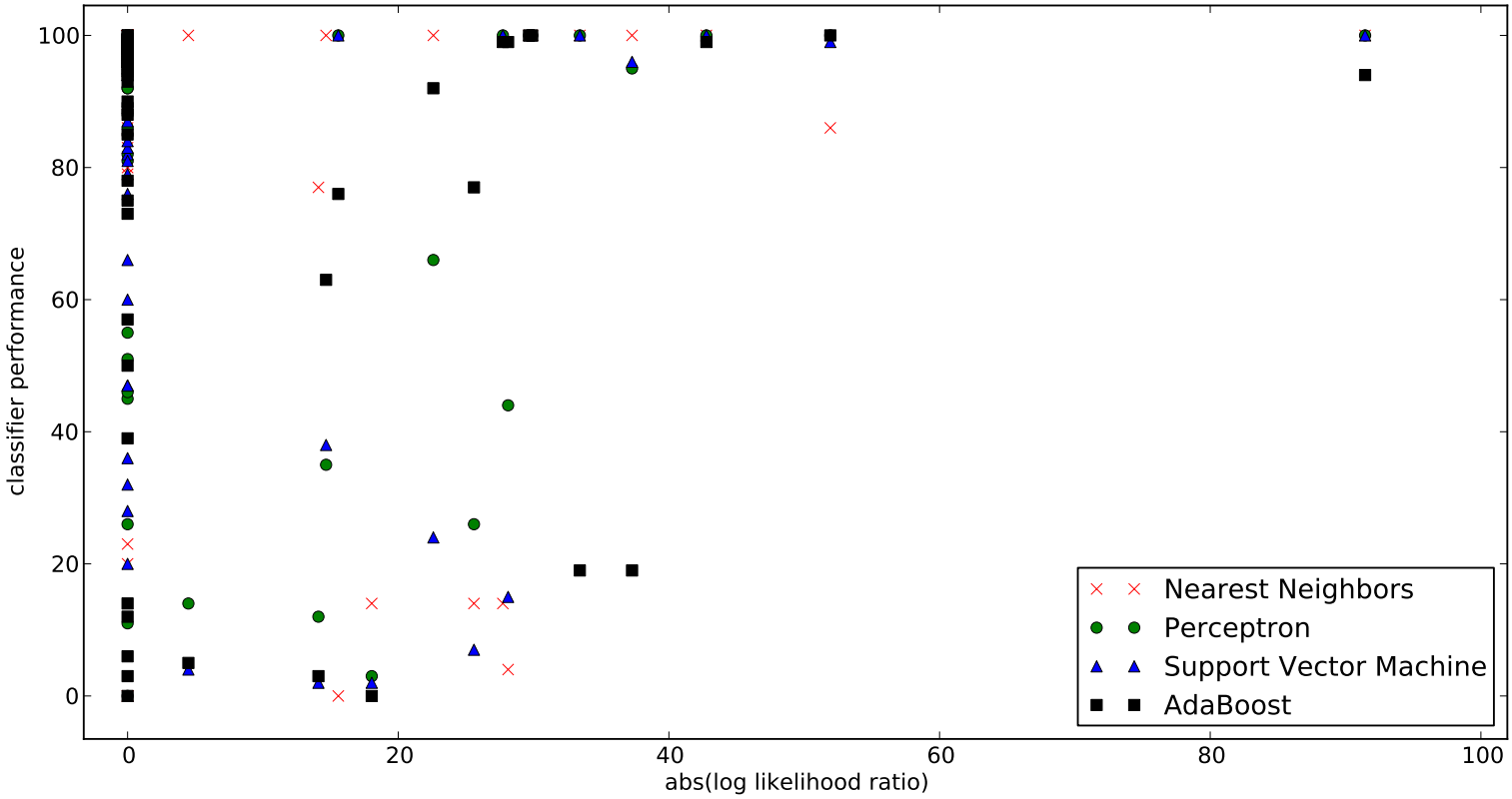

Figure 1: Proportion correct classification of data points in 100 runs of 5-fold cross-validation, plotted against magnitude of log likelihood ratio. (For readability, one point whose log likelihood ratio had a magnitude of 624 is not shown on this graph. Points with log likelihood ratios of $\pm\infty$ are also not shown.)

Feature Selection

To eliminate the problems from high dimensionality, and to determine which features were actually important for classification, feature selection was performed for each of the four classifiers. The algorithm used greedy forward selection, at each iteration adding in the feature that resulted in the largest cross-validation accuracy score (using one run of k-fold cross-validation), until the score stopped improving. Because cross-validation involves choosing random partitions of the data, there is an element of chance in what features get selected: a particular feature might just happen to result in a good cross-validation score on a particular run, without actually being better than the other features. Feature selection was performed using various versions of the data and various values of k for cross-validation, and in almost all cases, the improvement from including more than just the first feature was no more than a few percentage points. The first feature was usually sugar, and there was not much consistency in what features were selected subsequently. Therefore, cross-validation was then run using sugar as the *only* dimension for all the data; results are shown in Table 4. For nearest-neighbors and perceptrons, the accuracies are slightly worse than when using all features, but for support vector machines and AdaBoost, the accuracies are about the same.

Classifier	**5-fold accuracy**	**2-fold accuracy**
Nearest Neighbors	0.690	0.702
Perceptron	0.670	0.678
Support Vector Machine	0.826	0.808
AdaBoost	0.818	0.822

Table 4: Accuracy for each of the classifiers on the data using sugar as the only feature, in 25 runs of cross-validation.

Conclusions

The results indicate that there is definitely a difference in ingredients between cupcakes and muffins. All four of the classifiers used were able to distinguish between the two categories at above-chance levels. As expected, the distinction is not perfect – there is some overlap between the categories. The identifiable differences that they did find, such as the importance of sugar, agree with intuition.