Predicting Wallpaper Preferences

Peter A. Melick
Department of Mechanical Engineering
Stanford University
Stanford, CA 94305
pmelick@stanford.edu

Abstract

An algorithm is proposed for suggesting mobile/tablet wallpapers for a user. Such a method could theoretically be applied to an application which sends users wallpaper reccomendations from a library of images, which are sourced from other users. A method for quickly determining what types of wallpapers a user will like, which involves classification of the entire user base via k-means clustering, is described. In addition to the subject and potentailly the artist who created the image, the model takes into account visual qualities of the image, including dominant colors and it's visual busyness.

1 Data Collection & Processing

1.1 Web Survey

In order to collect data on mobile wallpaper preferences a web survey was created and distributed to the author's friends, family, and classmates. In addition to collecting information on demographics and free-form suggestions for the proposed service, the survey asked users to indicate whether or not they would use an image as their mobile/tablet wallpaper for 105 preselected images. The images were chosen to roughly uniformly represent 11 categories: abstract, aircraft, animals, architecture, cars, popular female media, popular male media, humor, nature, and space. Within each category, the images were selected to represent a range of visual attributes, as will be discussed in section 2. As of this submission, the website is still live at screenflavor.weebly.com.

1.2 Data Set Description

170 total unique users from a wide variety of ages and genders took the survey. Of them, 140 rated all 105 photos. Users who only rated some of the images are elegantly incorporated in the proposed algorithm. The initial form of the data was one .csv file for each photo, containing the IP address and rating ('Yes' or 'No') from each unique user. With a Python script the data in these spreadsheets were loaded into one user data array. Each column of the array represents a user, and each row contains a +1 (Yes), a -1 (No), or a 0 (Did not see/rate). This array was loaded into Matlab for the remainder of the processing.

2 Image Feature Vector Representation

A way to represent proposed wallpapers as vectors of image features was developed. In the free-form suggestions from the survey, the most commonly cited factors in wallpaper preference were the subject of the image, the visual busyness, and the colors (including their intensity and brightness). The vector representation was designed to include this information. The first 11 elements represent the subject of the image. Every image has a 1 in exactly one of those elements. The next three,

numbers between 0 and 1, represent the busyness of the image. The rest of the elements of the image feature vector, again numbers between 0 and 1, contain data on the image's colors.

2.1 Hue, Saturation, and Value

Images were represented in HSV space instead of RGB space since HSV more closely aligns with how humans perceive images.[1] In this space, each image is represented as three arrays, the elements of which represent the hue, saturation, and value of each pixel of the image, all numbers between 0 and 1. The following feature detection algorithms were used on the image data represented in this space.

2.2 Busyness Detection Algorithms

Several methods for detecting the busyness of an image were attempted. At first it was suspected that the 2D FFT of an image would contain information about its busyness, in the form of the strength of frequencies of pixel variations. Several interpretations of data in the 2D FFT were tried, but the central assumption of all was that large amplitude variations at high frequencies indicate busyness. By subjectively assigning a busyness level (0, 1, or 2) and computing the 2D FFT of each image, an attempt was made to classify images according to busyness. Both multi-nomial Naive-Bayes and multi-class SVM approaches were applied to the classification problem, but neither yielded convincing results (60% successful classification). Ultimately, the data extracted 2D FFT was determined not to be a good indicator of busyness. Although it seems to have a correlation on the edge cases (extraordinarily busy images and very uniform images) it does not provide much information on the majority of images.

Another method that did yield good busyness (altough is much slower than 2D FFT) was adapted from one described by D. Ganguly et al.[2] It takes the average value of the the average difference between a pixel and each of the 8 surrounding pixels. Performing this calculation on each of the three image components results in a busyness metric that seems to corresponds well to perceived visual busyness in each of those components. For example, in the images in Figure 1, the busyness algorithm estimates the busyness of the busy color component of each image as greater than 0.5 (1 is the theoretical max), and the busyness of the other two as close to 0.

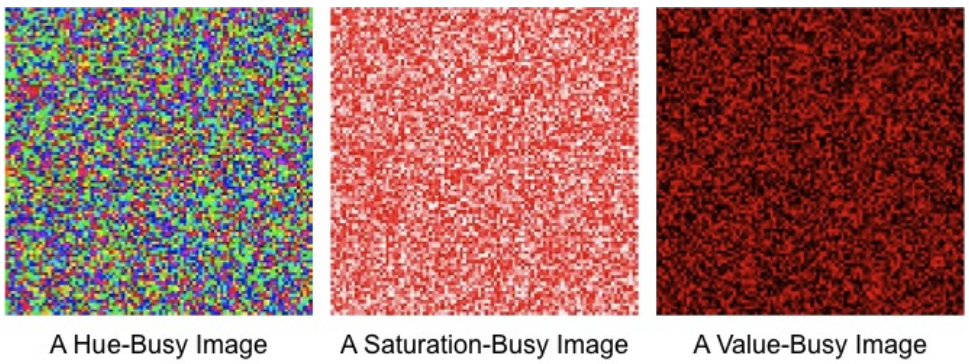

Figure 1: Examples of Hue, Saturation, and Value-Busy Images

As described, the busyness-detection algorithm is quite slow even for fairly small (320x480) images. For this reason, a granularity measure was introduced. Instead of sampling the average difference at every pixel, the algorithm samples at some specified interval. It then samples at a slightly finer interval and compares the two busyness values. If the two are the same within some threshold, no finer sampling is taken. Otherwise (meaning finer sampling found more busy areas) finer and finer samplings are taken until the busyness has converged. The actual busyness returned is that of the finest sampling taken before convergence. With this algorithm, only very busy images have long runtimes, since coarse samplings converge quickly for non-busy images. This worked extremely well, and resulted in very good estimations of the busyness of the non-sampled images with significantly shorter runtime on the set of all the images.

2.3 Colors

The representation of the colors in an image was fairly straightforward. The values in the color histogram of a component of an image indicate how many pixels fall within each of the bins of the histogram. To represent the colors 5 bins were used for hue, and 3 bins were used for each of saturation and value, meaning a total of 11 elements of the feature vector were used to represent color (bringing the total dimension of the feature vector up to 25). The value of the color elements of an image's feature vector are the number of pixels in a bin divided by the total number of pixels.

3 Data Analysis & Results

3.1 User Vector Representation

Each user is represented as a vector that is the sum of the ratings they've given every photo times that photo's feature vector, divided by the sum of the absolute value of all the feature vectors. This can be represented as

$$u = \frac{\sum_{photos} rating * p}{\sum_{photos} abs(rating * p)}$$

where u is a user vector, p is the feature vector representing a photo, and both sums are over all photos. Each element of u is increased when a user upvotes a photo with a certain feature ($rating = 1$), decreases when they downvote ($rating = -1$), and remains the same if they have not seen that photo ($rating = 0$). Therefore, user vector elements have values close to 1 when the user likes that particular feature, values close to -1 when a user dislikes that particular feature, and values close to 0 if the user is indifferent to that particular feature.

Since it is assumed that some photos have an intrinsicly higher probability of being upvoted than others (and since a user up/downvoting a photo that most users down/upvote should count for more than a user up/downvoting a photo that everybody up/downvotes) a photo's mean rating is subtracted from all its ratings before computing the user vectors.

3.2 Principal Components

To ensure that the features selected were actually relevant in the user data, Principal Component Analysis was done on the user data set. The first eigenvector of the covariance of the user matrix (whose columns are user vectors) has by far the largest eigenvalue, and has almost uniformly distributed elements. This is a good indicator that the user data has a good distribution of users who like and dont like every feature, and it is not the case that, for example, all the users don't like busy images.

3.3 Predicting Ratings

A central assumption of the proposed reccomendation system is that all the users could be classified into a small number of types of users, all of whom share similar opinions on which type of photo they would use for the mobile/tablet wallpaper. Users of a particular type are only shown photos that some threshold (for now, more than half) of other users of the same type like. After a small number of up/downvotes (more on how small later) a user can be classified and the system will only show photos with a high likelihood of upvote from that point on.

Another advantage of this type of system (instead of just showing the photo closest to their vector) is that it allows users to see a photos with features they have not yet indicated a preference for, but which similar users have liked. For example, if a user indicates a strong preference for pictures of cars in their first few up/downvotes, they may by classified into the group of users who likes cars, aircraft, and popular male media (and some visual attributes as well). Then they can begin to see photos they will probably like, without ever directly stating (or even neccessarily *knowing*) that they like those features.

3.4 K-Means Clustering & Model Testing

The k-means clustering algorithm was used to find the clusters of similar users from the data. It randomly picks users to be the cluster centroids, assigns each user to the closest centroid, and then re-computes the centroids, and repeats until convergence. In order to test the model (the centroids, assignments, and prediction system) the 70% of the data set was used to train the model and 30% was used to test the predictions. In order to limit the effects of local minima in the k-means clustering algorithm and different accuracies on different splits of data, the accuracy of a model is taken as the average value of twenty accuracies computed on different sets of randomly split data.

By varying the number of centroids and computing accuracies as described, the relationship between number of centroids and accuracy was measured, and is shown in Figure 2, below.

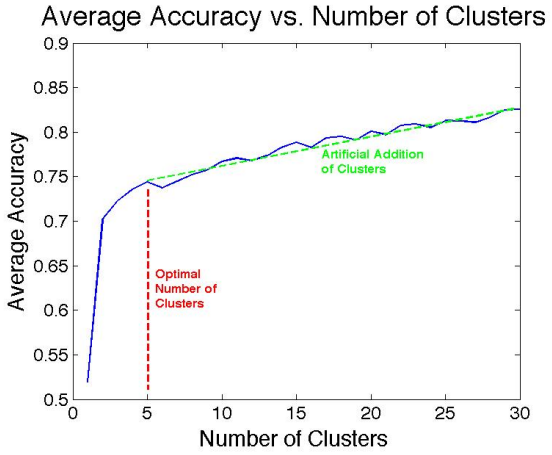

Figure 2: Five Centroids Seems to be an Appropriate Number to Cluster the Data

The described model and testing yields 75% prediction accuracy on average with five clusters. Above five clusters the accuracy slowly increases roughly linearly with the number of clusters, but with this relatively small dataset having many more clusters means having very few users per cluster. Having more users per cluster has the advantage of having more photos with very high likelihood of upvote, and is more likely to bring in alternative features, as decribed in the cars/aircaft example.

3.5 Mixture of Gaussians

As an alternative to k-means, a mixture of gaussians model (with 5 gaussians) was fit to the data using expectation maximization. The predicted ratings were modified to be a weighted average of the ratings predicted by each cluster (the weightings are the probabilies that a user belongs to each cluster). However this model did not provide more accurate results than k-means and predictions were much slower. It's convergence was also dependent on the initialization of the means of the guassians.

3.6 Consensus Threshold

This particular reccomendation system has a nice luxury - it can choose whether to predict or not, by only showing photos that it is very confident a user will like (of course, this is limited by how many very-likely-to-be-upvoted photos are available). By raising the number of similar users who must agree that a photo is good before it is recommended to a new user, the prediction system can achieve must higher successful prediction rates. The 75% success rate (with 5 clusters and (1/2) consensus threshold) can be increased to 83% with a (3/4) consensus thresold, and 96% with a (7/8) consensus threshold.

3.7 Minimum Ratings for Classification

An advantage of the user-clustering approach is that a user can be assigned to his/her cluster given only a few up/downvotes. In order to justify and quantify this assumption, the data set was split and the k-means algorithm was run on it. Then, for each test user, the minimum number of randomly selected photos to successfully classify that user into their eventual cluster was calculated. The results of 1000 trials of this process are shown in Figure 3, below.

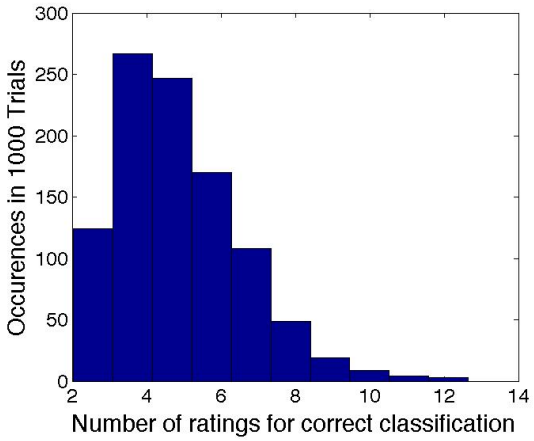

Figure 3: The Number of Up/Downvotes Needed to Correctly Classify a User

Ten ratings was enough to correctly classify a user in 98.7% of the trials. Therefore, it should be possible to begin showing users only photos they will very likely upvote after only about ten photos at most.

4 Future Work

A mixture of another type of probability distributions could be fit to the data, since it appears to not fit a mixture of gaussians very well. Perhaps a mixture of Poisson distributions would be more appropriate. Another good next step would be to implement a live trial of the test described and simulated in determining the minimum ratings for classification. This would involve presenting images to new users and choosing which to present to them based on the current classification of their user vector.

5 Acknowledgements

I would like to thank Professor Ng and the CS229 staff for running an excellent course, and providing me with the framework to perform my analyses. I would also like to acknowledge C.C. Chang and C.J. Li for the use of their LIBSVM.

References

[1] D. Cardani, Adventures in HSV Space, The Advanced Developers Hands on Conference July 2001.

[2] D. Ganguly, A Novel Approach for Edge Detection of Images, International Conference of Computer and Automation Engineering March 2009.