Finding the Best Panoramas

Jeremy Pack

CS 229 Fall 2011

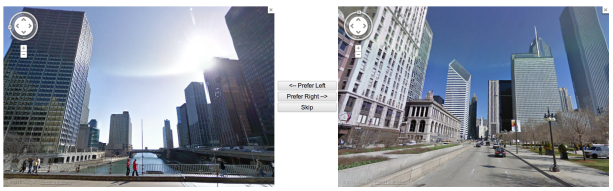
Abstract

Google Maps publishes street level panoramic photographs from around the world in the Street View service. When users request street level imagery in a given area, we would like to show the best or most representative imagery from the region. In order to select the best panorama for a region of any size, I developed a panorama ranking algorithm.

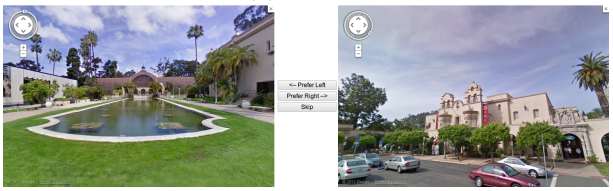
An enhancement to this technique is also described here, leveraging the Alternating Direction Method of Multipliers to create a high throughput distributed online learning algorithm that should allow for instant classification updating based on real-time user traffic.

The ranking algorithm was deployed on maps.google.com on Monday, December 12, 2011.

For more in depth information on the particular difficulties posed by our work on Google Street View, please refer to [1] and [2].



(a) Chicago



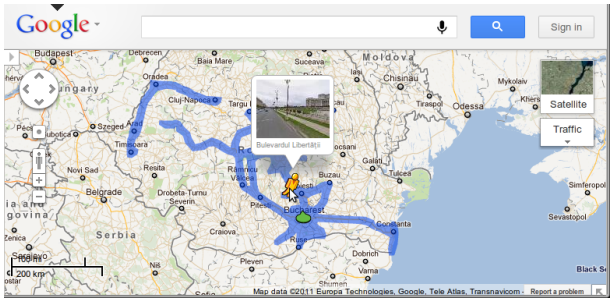
(b) UC San Diego

(c) South Africa

Figure 1: Ranking Server for Different Regions

1 Training data

1.1 Ranking panoramas



The goal of this work is to be able to determine a best panoramic image in any given set of panoramic images. Assuming that any best panorama from a set would also be the best in any subset containing it, and that any subset without it would have a different best panorama implies that there is some total ordering on the set of all panoramic images. As will be discussed later in Section 6.2, this ordering depends on the definition of 'best' for the current user.

In [3], user rankings of photographs are averaged together to estimate image quality. This technique is very prone to scoring bias from individual users, which must then be accounted for. Instead, I use pairwise ranking, where the user selects one of two panoramas as preferred. Another alternative would be listwise ranking, which can exhibit better statistical and convergence characteristics ([4]). Pairwise ranking was selected instead in order to get a variety of rankings as quickly as possible, but listwise ranking may be better for future training.

1.2 Condorcet Criterion

A Condorcet method can be used to find the preferred item in a set of items by considering the pairwise preferences across all items (see [5]). For a method to meet the Condorcet Criterion, it must guarantee that if any single item in the set is preferred in every pairwise comparison to every other item in the set, it must be the selected Condorcet Winner. There are far too many panoramas to perform a comparison between every possible pair. Instead, I use a small number of comparisons to statistically determine machine learning parameters to maximize the likelihood of satisfying the Condercet Criterion, similar to the technique described in [6]. In addition, we

use the extension to the Condorcet Criterion and Kemeny Orders described in [7]. This requires that any item that is 'consistently' preferred over another item should be ahead in the final ranking.

1.3 Preference training data

To estimate the pair-wise rankings for a number of panoramas, an application was constructed to randomly show a pair of images from some location. It may show, for example, two images from Chicago (Figure 1a), UC San Diego (Figure 1a), South Africa (Figure 1c), or some other region of any size. A user would then select one of the two images as the better image to represent the given area.

Users were encouraged to rate images from areas where they had lived, studied, or frequently traveled. Approximately $10,000$ such rankings were used to train the machine learning algorithm. It must be stressed that the goal was not to find just the most likely Condorcet ranking for this small set of panoramic images, but to determine machine learning parameters to automatically generate the most likely Condorcet ranking for all Street View panoramas in the world - including those that were never manually rated.

There are multiple sources of error and ambiguity in the training data:

- Users may accidentally select the wrong pano as the better panorama.
- Different users may have very different preferences for what constitutes a good panorama.
- There were only a small number of users rating a large number of panoramas, meaning that there are certain biases apparent in the results.

2 Model

2.1 Parameters considered (partial list)

To train the machine learning classifier, features including the following were considered:

- Camera type (there are a number of different resolutions in Street View imagery)
- Date of image capture
- Connectedness of panorama (intersections etc.)
- Nearby geo-located user photos
- User photos that match the given panorama (Street View panoramas are feature matched with public user photos)
- Count of unique users to contribute photos
- Nearby businesses and landmarks
- Type and importance of road
- Number of people or cars in the image
- Collection vehicle (trike, car, snowmobile, etc.)

2.2 Simplified model

The parameters in the modeled score include the following:

- θ : non-landmark feature weights
- x_i : some combination of the non-landmark features for panorama i
- ϵ_i : set of nearby landmarks for panorama i - each feature is in one of a small set of super-categories, and also part of a larger set of sub-categories (government buildings, businesses, parks etc.)
- δ_i : distance of the landmarks ϵ_i from panorama i
- ϕ : feature weight for the various types of landmarks in the sub-categories and super-categories.
- $f(\epsilon_i, \delta_i, \phi)$: the weight in the panorama score for panorama i, given the nearby landmarks ϵ_i.
- $\theta^T x_i + f(\epsilon_i, \phi)$: the score for panorama i.

The function $f(\epsilon_i, \delta_i, \phi)$ is assumed to be increasing in δ_i for landmarks with a positive weight in ϕ, or decreasing when the weight in ϕ is negative. A simple function that meets this criteria is the sigmoid function $g(z) = \frac{1}{1+\exp(-z)}$, where z is some linear function of the distance δ_i. f is then the ϕ weighted sum of these sigmoid functions of the distance. Note that this sum is not convex in ϕ and δ_i.

2.3 Binary Classifier

The training data is a set of preferences of the form "Panorama A is better than Panorama B". Using the scoring model described previously, this can be modeled as "The score of Panorama A should be higher than the score of Panorama B". We thus want to develop a classifier that generally selects the same panorama as better as was picked by the users. An error is then the case where the panorama selected as better by the user is considered worse by the algorithm.

As described in Section 1.3, there are numerous sources of error in the training data. Because of this, there is no reason to believe that the data will be separable, no matter how many features are included for consideration. Preference rankings are thus considered to be probabilistic predictors of the true ranking.

3 Method of solution

3.1 Descent method

As mentioned above, the resulting score function is not convex. This necessitated the use of sequential convex optimization to converge to a solution. Once the initial solution was found, the distance function parameters in f were set to be constant while selecting features.

By setting these parameters constant, it was then possible to use the Support Vector Machines algorithm (SVM) to develop a classifier. Many of the features being used were primarily discrete-valued or boolean. Most of the remaining features were usually zero (most panoramas have no nearby post offices, for example). This made SVM very unstable, and it performed worse generally than a basic linear classifier. Stabilizing an SVM with discrete features is described in [8], but further experimentation is needed to determine if their technique would work well with the mixture of features here.

3.2 Feature selection

Being unable to use SVM, I was forced to determine the nonlinear features to include manually. To do this, I split the data into subsets using two splitting techniques. The first technique involved using a Mixture of Gaussians model (as described in [9]) to split the data into sets of similar panoramas. The second technique split across the different discrete features (all highways in one bucket, all sidewalks in one bucket, etc.). I then calculated the classification error for the comparisons between each subset of panoramas. For those pairs of subsets with high classification error, I attempted to determine the most relevant nonlinear features that could be used to distinguish the subsets.

To determine which features to remove, I partitioned the preference set into parts, and trained on each one separately. The feature with the highest probability of being zero was repeatedly removed to determine the final set of features to consider.

3.3 Optimization function

The optimization function was basically as follows:

$$\begin{aligned} &\text{minimize } C\,||e_i||_1 + \lambda\,||\theta||_2^2 + \eta\,||\phi||_2^2 \\ &\text{subject to } g(x_i) - g(x_j) + e_k \geq 1 \text{ for i,j in comparison k} \\ &\qquad\qquad e_i \geq 0 \end{aligned}$$

Convergence was excellent and robust, and for the full training set it gave the same training and generalization errors for a variety of values of C, λ and η – 29.5% and 30% respectively. Since the absolute minimum training error given the cycles in the training data was 10%, and even an over-fit SVM couldn't get better than 29% training error, I found those results quite satisfactory.

3.4 Weighting of training samples

If all of the panoramas in the suburbs of San Jose were perfectly ranked, but the panoramas of the Golden Gate bridge and other Bay Area landmarks were poorly ranked, the algorithm would be of little value. We thus want to weight important panoramas more heavily than less important panoramas in the algorithm.

To make sure high scoring panoramas were weighted more heavily, the random selection of panoramas wasn't uniform. As training samples were added, early iterations of the classification were run to generate approximate scores. Panoramas with higher scores were then included more often in the set of panoramas presented to the people rating.

3.5 Satisfying the Condorcet Criterion

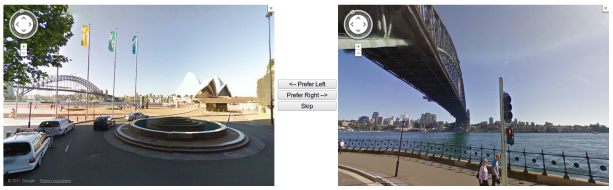
After the weights are found for the scoring function, scores can be generated for all panoramas in the world. Because of the subjectivity of the training data as described in section 1.3, a high percentage of the user preferences cannot be satisfied, even after including a number of nonlinear features. We can still add a manual weight to the panoramas used for training to still satisfy the Condorcet Criterion in cases where we are reasonable certain that one panorama should score more highly than another panorama.

Because we do not have a comparison between every panorama, we can use linear programming to find a Kemeny-like solution without resorting to an NP Hard algorithm, as described in [6]. Given the user preference that panorama A be scored higher than panorama B, for example, we would have that $S(A)+m_A-S(B)-m_B+e_k>0$ where m_A, m_B are the manual scores for A and B respectively, and e_k is the unsatisfied disagreement term. Massaging this into a linear program requires using separate variables for the positive and negative parts of each of these values, and then minimizing the linear sum of all m and e values.

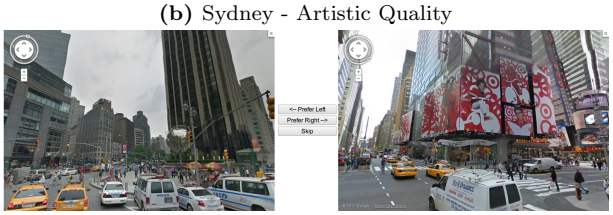
After the initial trial of adding manual weights, it became clear that there was a problem case where low quality panos that were only ranked once or twice could be ranked higher than high quality because of user error in the classification. After the first run one of Paris's fine public restrooms was ranked more highly than the Eiffel Tower. Thus, reasonable certainty requires that the probability of ranking error and user bias be considered in the algorithm.

The simplest way to avoid this type of error proved to be a requirement that any change in the manual score of a panorama be accompanied by a much larger decrease in total disagreement in ranking. This means weighting e more heavily in the optimization function in the linear program. The precise value for the weight of e depends on the expected error rate.

(a) Macau and Hong Kong- Tourist/Event Locations



(b) Sydney - Artistic Quality



(c) New York City - Heading Selection

Figure 2: Difficult Ranking Decisions

4 Distributed online learning

4.1 High volume training data

Though this offline classifier works generally well at finding good panoramas, actually determining which panoramas are best for the users of the Street View service across a variety of use cases will require far more training data. It would be preferable to analyze user traffic in real-time to constantly update the algorithm parameters.

4.2 Training separately on subsets of the data

To perform online training on a large dataset, it will be necessary to distribute the machine learning algorithm. This actually provides an interesting opportunity to train the algorithm separately on different subsets of the data.

It makes sense, for instance, that the machine learning algorithm may result in different parameters in different countries, because of the variability across countries and languages of the quality of the landmark and user photo data sets that are being used as features in the classifier.

Also, there is a discernible difference in the parameters for user photographs and landmarks for panoramas that are on highways or freeways versus those that are on neighborhood streets or major thoroughfares.

Given a subset $p \in P$, it is thus possible to assume some global value for the parameters θ, as well as local modifiers to the global parameters κ_p for each subset p. The scoring for subset p of the panoramas would then depend on the value $\theta + \kappa_p$. The classifier would then be trained separately for each pairwise combination of $p_1, p_2 \in P$. Any pair without enough training samples to train a classifier would not be considered.

The results of the different training runs could then be averaged for each $p \in P$.

4.3 The Alternating Direction Method of Multipliers

It is possible to do far better than simply averaging the results. The Alternating Directions Method of Multipliers (ADMM) is a powerful way to split a convex optimization problem into multiple sub-problems that are solved repeatedly (see [10]). This technique shows far better convergence than simple averaging, and is guaranteed to converge given basic requirements on the functions used in each sub-problem.

I wrote code to perform ADMM consensus (see [11] for information on consensus algorithms) across subsets of the data. ADMM convergence was good, with a few special difficulties. For instance, since a binary classifier was being used, scaling of the scores was arbitrary and they had to be scaled separately in each subproblem.

The great benefit of using ADMM is that it quite naturally allows for online training, and once initially converged, keeps the solution very close to optimal as more data are added over time. This means that true online, distributed real-time training is achievable, even with a high volume of real-time training data.

5 Further work

5.1 Heading selection

The current implementation on maps.google.com does not automatically select a good direction for the user to turn to view the panorama. The optimal direction can be determined by considering the direction of user photos from the current panorama, or by considering the direction of nearby landmarks.

Over time, it may be possible to calculate the optimal directions based on user behavior. As seen in Figure 2c, in a place like the south-west corner of Central Park or Times Square in New York City, it can be difficult to determine a best direction to turn the panoramic view.

5.2 Iconic views

Some landmarks are so famous that we want more than just some view of the image - we want the iconic view. In [12], the authors present techniques for automatically determining from a large set of photographs which view of an object is iconic. This type of technique could be used to select the best of many different views of famous places. Their work also discusses automatically determining which object in a photograph is the "subject",

which could make it easier to determine which direction to point the user initially when viewing the panorama.

6 Conclusions

6.1 Fun facts

There were a number of interesting things learned from the training of the algorithm:

- The number of unique photographers who took photos in an area is more important than the number of unique photos in an area.
- Government buildings are a strong positive predicter of panorama ranking.
- Bars and hotels have are a strong predicter of low quality panoramas.
- Nearly ten percent of user preferences could not be satisfied by *any* ordering.
- People love their sports teams, so when they rate their home town they always rate the stadium highly. This led to a very strong positive weight on stadiums in the rankings.

6.2 Definition of "best" panorama

Users have very different opinions about which panoramas are best. Would a user prefer the view of Sydney Australia that shows both the Harbour Bridge and the Opera House – or the artistically superior shot from below the Harbour Bridge of the Sydney skyline (Figure 2c)?

These differing ideas of what constitutes "best" will actually change even for a given user, depending on the context. As such, a natural extension of this work is to model the behavior of a given user to predict which type of imagery they would be interested in seeing. There could then be a number of separate rankings for the panoramas, to target users interested in seeing the best cafes and restaurants in town; or the best architecture; or the most interesting churches, cathedrals and temples; or the best views of the ocean or mountains.

6.3 Launch on maps.google.com

The initial demonstration of this technique was so convincing that it was determined that the ranking should be made available to the public immediately. The ranking generated using the offline classifier was made available as the default panorama selection method on the website maps.google.com on Monday, December 12th.

To use this new ranking information on Google Maps, move to a map view at the country or state level, and drag and drop the Pegman figure onto a city. The service will attempt to return the best panorama from within a few pixels of the selected location.

References

[1] L. Vincent, "Taking online maps down to street level," *Computer*, vol. 40, no. 12, pp. 118–120, 2007.

[2] D. Anguelov, C. Dulong, D. Filip, C. Frueh, S. Lafon, R. Lyon, A. Ogale, L. Vincent, and J. Weaver, "Google street view: Capturing the world at street level," *Computer*, vol. 43, no. 6, pp. 32–38, 2010.

[3] Y. Ke, X. Tang, and F. Jing, "The design of high-level features for photo quality assessment," in *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, vol. 1, pp. 419–426, Ieee, 2006.

[4] Z. Cao, T. Qin, T. Liu, M. Tsai, and H. Li, "Learning to rank: from pairwise approach to listwise approach," in *Proceedings of the 24th international conference on Machine learning*, pp. 129–136, ACM, 2007.

[5] D. Austen-Smith and J. Banks, "Information aggregation, rationality, and the condorcet jury theorem," *American Political Science Review*, pp. 34–45, 1996.

[6] C. Dwork, R. Kumar, M. Naor, and D. Sivakumar, "Rank aggregation methods for the web," in *Proceedings of the 10th international conference on World Wide Web*, pp. 613–622, ACM, 2001.

[7] M. Truchon, "An extension of the condorcet criterion and kemeny orders," *Cahier*, vol. 9813, 1998.

[8] K. Sadohara, "Learning of boolean functions using support vector machines," in *Algorithmic Learning Theory*, pp. 106–118, Springer, 2001.

[9] J. Bilmes, "A gentle tutorial of the em algorithm and its application to parameter estimation for gaussian mixture and hidden markov models," *International Computer Science Institute*, vol. 4, p. 126, 1998.

[10] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," tech. rep., working paper on line, Stanford, Univ, 2010.

[11] F. Zanella, D. Varagnolo, A. Cenedese, G. Pillonetto, and L. Schenato, "Newton-raphson consensus for distributed convex optimization," 2011.

[12] T. Berg and D. Forsyth, "Automatic ranking of iconic images," *University of California, Berkeley, Tech. Rep*, 2007.