"Combination of Experts" Approach to Image Boundary Detection

David Cohen and Jim Rodgers
{dscohen,jimkr}@cs.stanford.edu

December 13, 2006

1 Introduction and Background

Boundary detection in two-dimensional images is an important problem in computer vision. There are a wide variety of algorithms to accomplish this task, but none have come close to human proficiency. We explore a variety of ways to use machine learning algorithms to combine existing boundary detection algorithms with the goal of exceeding the performance of any particular algorithm. We present methods using Adaboost and linear regression and show favorable results produced by them.

Boundary detection is a classification problem which entails labeling a subset of pixels in an image as part of an edge that separates two objects. However, there is no clear definition for which objects should be separated. The most common approach is to attempt to find a set of edges which a human being would consider reasonable. Nonetheless, for any given image, there may be disagreement among human beings as to what set of edges best captures the image. Additionally, while some boundary detection algorithms impose hard boundaries, others instead provide probabilities that each pixel in an image is part of an edge.

Boundary detection, an active research area within the artificial intelligence field of computer vision, has a variety of applications to higher-level vision tasks. Many object-recognition algorithms use image boundaries as inputs. Furthermore, boundary detection algorithms help show the precise orientation of an object in space, which is useful for robotic manipulation tasks. This approach to boundary detection is particularly interesting to us, since we are working on an MRF-based image segmentation algorithm that takes the output from a boundary detection algorithm as its input. Thus, this project could provide better inputs for image segmentation. Alternatively, it may allow us to combine the image segmentation algorithm's boundary output with other boundary detection algorithms' outputs for even better results.

2 Methodology

Our goal is to predict which pixels in an image fall are edges between objects in that image. Unfortunately, as noted above, there is no clear definition of an edge in an image. Further, even when human beings agree on edges, they often disagree on precisely which pixels in an image correspond to them. To account for this, we use soft boundary maps computed from the edges selected by a number of human subjects as our ground truth. Similarly, we use our algorithms to produce soft edge maps, which we evaluate by applying various threshold values, thinning wide edges, and measuring the quality of each of the resulting thin hard edge maps. We evaluate these edge maps using precision, recall, and f-score, where:

$$\begin{aligned} &\textit{precision} = \\ &\quad P(\textit{point is an edge in the ground truth image}| \\ &\qquad \textit{point is predicted as edge}) \end{aligned} \tag{1}$$

$$\begin{aligned} &\textit{recall} = P(\textit{point is predicted as edge}| \\ &\quad \textit{point is an edge in the ground truth image}) \end{aligned} \tag{2}$$

$$\textit{f-score} = \frac{2 * \textit{precision} * \textit{recall}}{\textit{precision} + \textit{recall}} \tag{3}$$

Plotting precision and recall for each threshold value produces a curve which characterizes an algorithm's behavior, and the maximum f-score of the data points plotted

yields a single number which summarizes the algorithm's success.

3 Experts and Features

We combine the results of currently existing boundary detection algorithms to produce our edge predictions. We assemble a large number of boundary detection algorithms and experiment with methods of combining them to produce a single boundary map. Our set of boundary detection algorithms currently includes six hard-threshold classifiers which produce logical boundary maps, and five probabilistic classifiers which produce soft boundary maps, associating each pixel with the probability it is an edge. Some of the soft edge maps are drawn from [1]. Five of the hard-threshold methods and all of the soft-threshold methods approximate gradients in the image and predict edges at points which correspond to local maxima of the gradient. The two remaining detection methods are a Laplacian of Gaussian algorithm which selects points where the Laplacian of the image changes its sign, and Felzenszwalb's graph-based image segmentation algorithm [2] which merges regions of an image based on their variations in intensity. By varying the parameters of these algorithms, such as threshold values and filter sizes, and by varying the image channel on which these algorithms are run (red, green, blue, or grayscale), we produce 91 different edge maps for each original image. We call each parameterization of a boundary detector on each image channel an *expert*.

In order to learn from our experts, we consider the problem of classifying a single pixel as an edge or non-edge point based on its treatment in the experts' edge maps. Since each expert produces one edge map per image, it contributes exactly one feature for each pixel. We experimented with three different methods for extracting features from experts' edge maps. Let $f_a(x, y, e)$ be the feature extracted by algorithm a from expert e for pixel (x, y), and let M_e be the edge map produced by expert e. In our first feature extraction method, *direct extraction*, $f_{direct}(x, y, e)$ takes the value $M_e(x, y)$. For *Euclidean distance extraction*, $f_{dist}(x, y, e)$ has the value of the Euclidean distance in M_e between (x, y) and the nearest edge pixel in the edge map. In *Gaussian distance extraction*, $f_{gaussian}(x, y, e)$ takes the value $exp(-f_{dist}(x, y, e)^2/v)$ where v is the variance of our Gaussian distribution. We experiment with several such variances.

In constructing training sets, we vary the ratio of edge pixels to non-edge pixels in our training set. We create non-balanced training sets by randomly sampling pixels from our training images, and we label a pixel as an edge if it falls within a one pixel radius of any pixel with a non-zero value in the ground truth edge map. These training sets are composed of approximately 200 edge pixels and 800 non-edge pixels for each image. We also create balanced data sets by sampling 500 edge pixels and 500 non-edge pixels from each image.

4 Learning algorithms

We explore several learning algorithms to combine the predictions of our experts. The two most successful of these was real-valued Adaboost, followed by least-squares linear regression. Real-valued Adaboost is an iterative algorithm that converts features into classifiers by choosing the specific feature and threshold with maximal performance on a weighted version of the training set. For example, one such classifier might be:

$$prediction = \begin{cases} \textit{edge} & \text{if feature } k > \text{threshold } t \\ \textit{non-edge} & \text{otherwise} \end{cases}$$

At each iteration, Adaboost reweights the training set to increase the importance of data instances misclassified by the newly chosen classifier. The result of running Adaboost for n iterations is a linear combination of n classifiers, with coefficients determined by the accuracy of each classifier on the weighted data set. This linear combination of classifiers is then applied to test data.

The other successful algorithm we find is least-squares linear regression, which finds the linear combination of the features that minimizes the squared error of feature-value predictions. Unlike Adaboost, which attempts to directly classify data instances, linear regression estimates an arbitrary real-valued function with a line through d-dimensional space, where d is the number of features. We use linear regression to estimate the same features in the test data that we calculate in the training data: raw pixel value, Euclidean distance, or Gaussian distance.

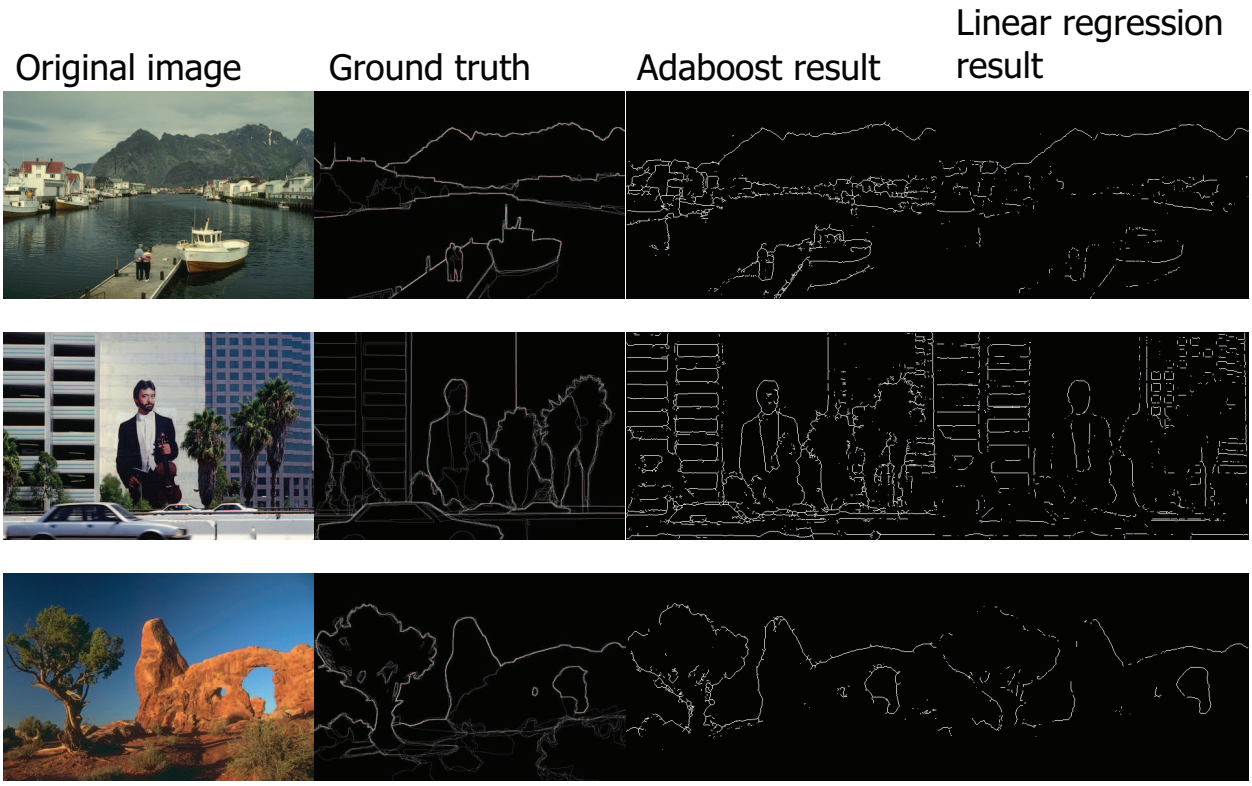


Figure 1: Sample results of running Adaboost and linear regression on the original image, and a comparison with the human-generated ground truth.

5 Results

We train and test our algorithms on twenty images from the training set and test set, respectively, provided by the Berkeley Segmentation Dataset [4]. Figure 1 shows a sample of our results on some of the images. We find that Adaboost performs the best of all the algorithms, both visually and according to the f-score. When trained on a balanced data set based on Gaussian distance features with a standard deviation of 2 pixels, Adaboost achieved an f-score of 0.67. We note that this exceeds the f-score of our best expert by 0.04, but also that this particular expert achieved that score when tested on a 100 image test set. This indicates a small performance gain over the baseline.

We find that the most significant factor in our success with Adaboost was the use of a balanced training set, observing that all of our experiments with different features over balanced datasets scored at or above 0.60, and all Adaboost runs with non-balanced training sets scored at or below 0.56. We believe that this result arises from the ratio of edge points to non-edge points, rather than the absolute numbers of edge or non-edge points. As evidence, we note that our f-score on a balanced Gaussian distance dataset (with standard deviation of 4 pixels) only dropped by 0.01 when we halved the number of training examples used. This highlights the distinction between our objective function, the accuracy, and our evaluation mechanism, the f-score. The distinguishing characteristic of the f-score is that it completely ignores true negatives: the number of non-edge points correctly classified by an algorithm does not affect its f-score, though misclassifying non-edge points as edge-points does lower an algorithm's f-score. The f-score captures the intuition that edges are

more important than non-edges in an edge map, and accounts for the fact that the number of edge points in an image varies linearly with its scale, whereas the number of non-edge points varies quadratically. Adaboost yields similar accuracy on both types of training sets, but both the f-scores and the visual appearance of the results on non-balanced sets are inferior.

Least-squares linear regression produces results that are not as good as those we obtain with Adaboost (see Figure 2), but are nonetheless reasonable. The composition of the training set has little impact on the results. We believe that this is consistent with the non-classification nature of linear regression. The idea of balancing the training set between edges and non-edges is less meaningful when dealing with an algorithm that does not attempt to match edge/non-edge labels, but rather match the real-valued feature that corresponds to any given ground truth pixel. Linear regression performs similarly with Gaussian distance and direct pixel values as features. It performs very poorly with Euclidean distance as a feature, yielding edge maps that predict edges almost everywhere. We achieve the best results, 0.61, with Gaussian features and standard deviation of 4 pixels. Figure 3 shows Precision-Recall curves for Adaboost and linear regression using different training sets and feature composition.

In addition to Adaboost and linear regression, we explore support vector machines (SVMs) and locally weighted linear regression. Neither produce acceptable results. We train SVMs, using the SVMlight package [3], on balanced and unbalanced data sets using pixel values and Euclidean distance features and obtain very poor recall scores. Furthermore, we find a large number of support vectors during SVM training, which hurts SVM performance. Locally weighted linear regression yields no obvious visual differences from standard linear regression when run with a wide variety of bandwidth values, while massively increasing runtime. Whereas standard linear regression produces only one set of parameters, locally weighted linear regression learns new parameters for every pixel in the test set, and thus appears intractable in the context of edge detection.

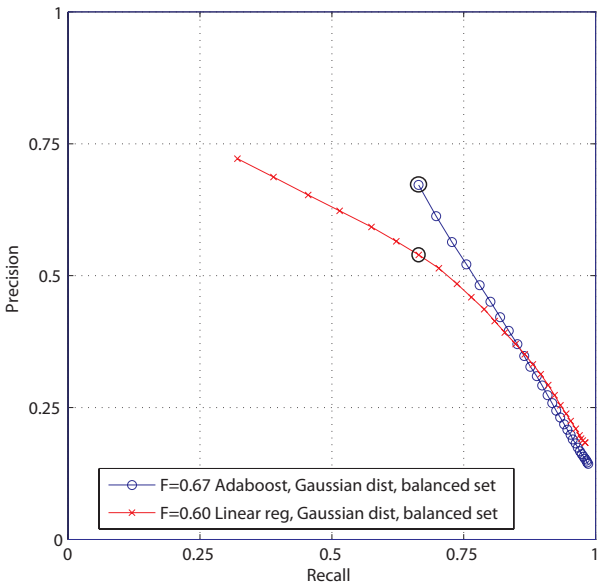
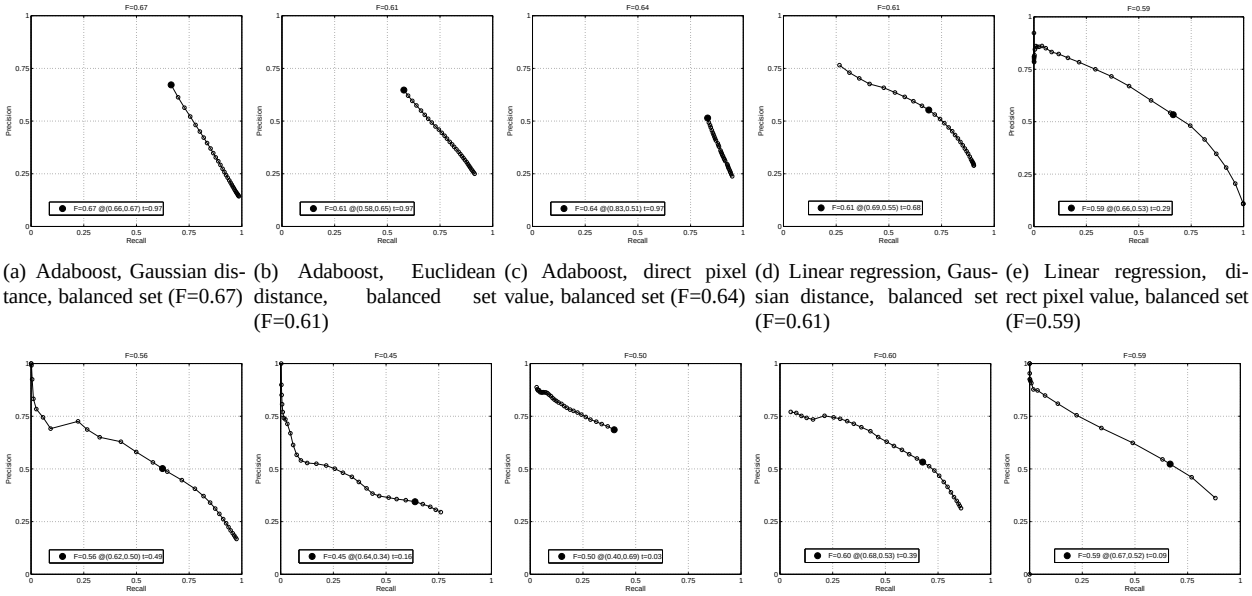


Figure 2: Precision and recall curves for best Adaboost and linear regression results. Adaboost performs better than linear regression

6 Conclusion and Future Work

We conclude that Adaboost is a suitable method for combining expert predictions for the image boundary detection problem. Not only does it appear that our current Adaboost classifier may perform better than all of its component experts (we have yet to confirm this by testing on the full Berkeley test set), our findings also offer several avenues for further improvement. First, we achieved a large improvement by changing the composition of our training set, but have not attempted to find the optimal training set edge to non-edge ratio. Further exploration of training set composition may continue to improve our results. Second, we found significant f-score differences based on the variance used for Gaussian distance features, and would like to find the optimal variance using cross-validation. Furthermore, we currently optimize the accuracy and evaluate performance on the f-score, but boosting may allow us to optimize the f-score directly, and thus improve our results. Also, we wish to incorporate knowledge of the softness of our ground truth edge maps by



(a) Adaboost, Gaussian distance, balanced set (F=0.67)

(b) Adaboost, Euclidean distance, balanced set (F=0.61)

(c) Adaboost, direct pixel value, balanced set (F=0.64)

(d) Linear regression, Gaussian distance, balanced set (F=0.61)

(e) Linear regression, direct pixel value, balanced set (F=0.59)

(f) Adaboost, Gaussian distance, unbalanced set (F=0.56)

(g) Adaboost, Euclidean distance, unbalanced set (F=0.45)

(h) Adaboost, direct pixel value, unbalanced set (F=0.50)

(i) Linear regression, Gaussian distance, unbalanced set (F=0.60)

(j) Linear regression, direct pixel value, unbalanced set (F=0.59)

Figure 3: Precision-Recall curves and their maximum F-score for different combinations of features and training set composition. Best performance is with Adaboost, Gaussian distance, and a balanced training set.

weighting edge pixels by their ground truth pixel values, and thus force our algorithm to favor correct predictions on the stronger edges. Finally, we wish to gather more experts for use in Adaboost and explore more methods for extracting features. Adding more experts and features, especially ones significantly different from those currently used, may improve the resulting edge maps.

7 Acknowledgements

Thanks to Steve Gould, Gal Elidan, and Ben Packer for providing helpful feedback and suggestions.

References

[1] D.Martin, C. Fowlkes, and J. Malik. Learning to detect natural image boundaries using local brightness, color, and texture cues. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(5):530–549, 2004.

[2] P. Felzenszwalb and D. Huttenlocher. Efficient graph-based image segmentation. *International Journal of Computer Vision*, 59(2), 2004.

[3] T. Joachims. Making large-scale support vector machine learning practical. In A. Smola B. Schölkopf, C. Burges, editor, *Advances in Kernel Methods: Support Vector Machines*. MIT Press, Cambridge, MA, 1998.

[4] D. Martin, C. Fowlkes, D. Tal, and J. Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *Proc. 8th Int'l Conf. Computer Vision*, volume 2, pages 416–423, July 2001.