Maximin affinity learning of image segmentation

Srinivas C. Turaga *
MIT

Kevin L. Briggman
Max-Planck Insitute for Medical Research

Moritz Helmstaedter
Max-Planck Insitute for Medical Research

Winfried Denk
Max-Planck Insitute for Medical Research

H. Sebastian Seung
MIT, HHMI

Abstract

Images can be segmented by first using a classifier to predict an affinity graph that reflects the degree to which image pixels must be grouped together and then partitioning the graph to yield a segmentation. Machine learning has been applied to the affinity classifier to produce affinity graphs that are good in the sense of minimizing edge misclassification rates. However, this error measure is only indirectly related to the quality of segmentations produced by ultimately partitioning the affinity graph. We present the first machine learning algorithm for training a classifier to produce affinity graphs that are good in the sense of producing segmentations that directly minimize the Rand index, a well known segmentation performance measure.

The Rand index measures segmentation performance by quantifying the classification of the connectivity of image pixel pairs after segmentation. By using the simple graph partitioning algorithm of finding the connected components of the thresholded affinity graph, we are able to train an affinity classifier to directly minimize the Rand index of segmentations resulting from the graph partitioning. Our learning algorithm corresponds to the learning of maximin affinities between image pixel pairs, which are predictive of the pixel-pair connectivity.

1 Introduction

Supervised learning has emerged as a serious contender in the field of image segmentation, ever since the creation of training sets of images with "ground truth" segmentations provided by humans, such as the Berkeley Segmentation Dataset [15]. Supervised learning requires 1) a parametrized algorithm that map images to segmentations, 2) an objective function that quantifies the performance of a segmentation algorithm relative to ground truth, and 3) a means of searching the parameter space of the segmentation algorithm for an optimum of the objective function.

In the supervised learning method presented here, the segmentation algorithm consists of a parametrized *classifier* that predicts the weights of a nearest neighbor affinity graph over image pixels, followed by a graph *partitioner* that thresholds the affinity graph and finds its connected components. Our objective function is the Rand index [18], which has recently been proposed as a quantitative measure of segmentation performance [23]. We "soften" the thresholding of the classifier output and adjust the parameters of the classifier by gradient learning based on the Rand index.

*sturaga@mit.edu

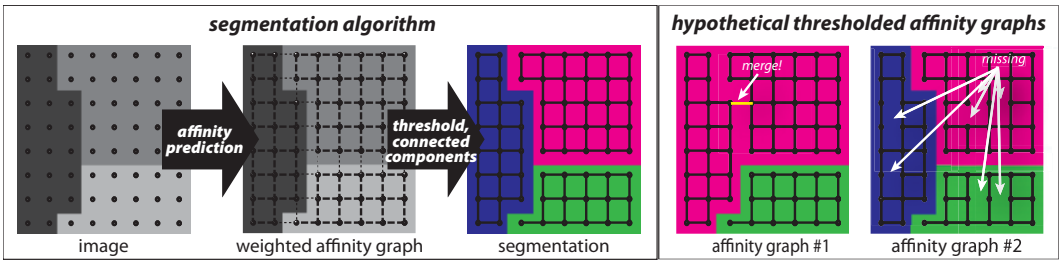

Figure 1: (left) **Our segmentation algorithm.** We first generate a nearest neighbor weighted affinity graph representing the degree to which nearest neighbor pixels should be grouped together. The segmentation is generated by finding the connected components of the thresholded affinity graph. (right) **Affinity misclassification rates are a poor measure of segmentation performance.** Affinity graph #1 makes only 1 error (dashed edge) but results in poor segmentations, while graph #2 generates a perfect segmentation despite making many affinity misclassifications (dashed edges).

Because maximin edges of the affinity graph play a key role in our learning method, we call it *maximin affinity learning of image segmentation*, or MALIS. The minimax path and edge are standard concepts in graph theory, and maximin is the opposite-sign sibling of minimax. Hence our work can be viewed as a machine learning application of these graph theoretic concepts. MALIS focuses on improving classifier output at maximin edges, because classifying these edges incorrectly leads to genuine segmentation errors, the splitting or merging of segments.

To the best of our knowledge, MALIS is the first supervised learning method that is based on optimizing a genuine measure of segmentation performance. The idea of training a classifier to predict the weights of an affinity graph is not novel. Affinity classifiers were previously trained to minimize the number of misclassified affinity edges [9, 16]. This is not the same as optimizing segmentations produced by partitioning the affinity graph. There have been attempts to train affinity classifiers to produce good segmentations when partitioned by normalized cuts [17, 2]. But these approaches do not optimize a genuine measure of segmentation performance such as the Rand index. The work of Bach and Jordan [2] is the closest to our work. However, they only minimize an upper bound to a renormalized version of the Rand index. Both approaches require many approximations to make the learning tractable.

In other related work, classifiers have been trained to optimize performance at detecting image pixels that belong to object boundaries [16, 6, 14]. Our classifier can also be viewed as a boundary detector, since a nearest neighbor affinity graph is essentially the same as a boundary map, up to a sign inversion. However, we combine our classifier with a graph partitioner to produce segmentations. The classifier parameters are not trained to optimize performance at boundary detection, but to optimize performance at segmentation as measured by the Rand index.

There are also methods for supervised learning of image labeling using Markov or conditional random fields [10]. But image labeling is more similar to multi-class pixel classification rather than image segmentation, as the latter task may require distinguishing between multiple objects in a single image that all have the same label.

In the cases where probabilistic random field models have been used for image parsing and segmentation, the models have either been simplistic for tractability reasons [12] or have been trained piecemeal. For instance, Tu et al. [22] separately train low-level discriminative modules based on a boosting classifier, and train high-level modules of their algorithm to model the joint distribution of the image and the labeling. These models have never been trained to minimize the Rand index.

2 Partitioning a thresholded affinity graph by connected components

Our class of segmentation algorithms is constructed by combining a classifier and a graph partitioner (see Figure 1). The classifier is used to generate the weights of an affinity graph. The nodes of the graph are image pixels, and the edges are between nearest neighbor pairs of pixels. The weights of the edges are called affinities. A high affinity means that the two pixels tend to belong to the same

segment. The classifier computes the affinity of each edge based on an image patch surrounding the edge.

The graph partitioner first thresholds the affinity graph by removing all edges with weights less than some threshold value θ. The connected components of this thresholded affinity graph are the segments of the image.

For this class of segmentation algorithms, it's obvious that a single misclassified edge of the affinity graph can dramatically alter the resulting segmentation by splitting or merging two segments (see Fig. 1). This is why it is important to learn by optimizing a measure of segmentation performance rather than affinity prediction.

We are well aware that connected components is an exceedingly simple method of graph partitioning. More sophisticated algorithms, such as spectral clustering [20] or graph cuts [3], might be more robust to misclassifications of one or a few edges of the affinity graph. Why not use them instead? We have two replies to this question.

First, because of the simplicity of our graph partitioning, we can derive a simple and direct method of supervised learning that optimizes a true measure of image segmentation performance. So far learning based on more sophisticated graph partitioning methods has fallen short of this goal [17, 2].

Second, even if it were possible to properly learn the affinities used by more sophisticated graph partitioning methods, we would still prefer our simple connected components. The classifier in our segmentation algorithm can also carry out sophisticated computations, if its representational power is sufficiently great. Putting the sophistication in the classifier has the advantage of making it learnable, rather than hand-designed.

The sophisticated partitioning methods clean up the affinity graph by using prior assumptions about the properties of image segmentations. But these prior assumptions *could be incorrect*. The spirit of the machine learning approach is to use a large amount of training data and minimize the use of prior assumptions. If the sophisticated partitioning methods are indeed the best way of achieving good segmentation performance, we suspect that our classifier will learn them from the training data. If they are not the best way, we hope that our classifier will do even better.

3 The Rand index quantifies segmentation performance

Image segmentation can be viewed as a special case of the general problem of clustering, as image segments are clusters of image pixels. Long ago, Rand proposed an index of similarity between two clusterings [18]. Recently it has been proposed that the Rand index be applied to image segmentations [23]. Define a segmentation S as an assignment of a segment label s_i to each pixel i. The indicator function $\delta(s_i, s_j)$ is 1 if pixels i and j belong to the same segment ($s_i = s_j$) and 0 otherwise. Given two segmentations S and $\hat{S}$ of an image with N pixels, define the function

$$1 - RI(\hat{S}, S) = \binom{N}{2}^{-1} \sum_{i<j} \left| \delta(s_i, s_j) - \delta(\hat{s}_i, \hat{s}_j) \right| \tag{1}$$

which is the fraction of image pixel pairs on which the two segmentations *disagree*. We will refer to the function $1 - RI(\hat{S}, S)$ as the Rand index, although strictly speaking the Rand index is $RI(\hat{S}, S)$, the fraction of image pixel pairs on which the two segmentations *agree*. In other words, the Rand index is a measure of similarity, but we will often apply that term to a measure of dissimilarity.

In this paper, the Rand index is applied to compare the output $\hat{S}$ of a segmentation algorithm with a ground truth segmentation S, and will serve as an objective function for learning. Figure 1 illustrates why the Rand index is a sensible measure of segmentation performance. The segmentation of affinity graph #1 incurs a huge Rand index penalty relative to the ground truth. A single wrongly classified edge of the affinity graph leads to an incorrect merger of two segments, causing many pairs of image pixels to be wrongly assigned to the same segment. On the other hand, the segmentation corresponding to affinity graph #2 has a perfect Rand index, even though there are misclassifications in the affinity graph. In short, the Rand index makes sense because it strongly penalizes errors in the affinity graph that lead to split and merger errors.

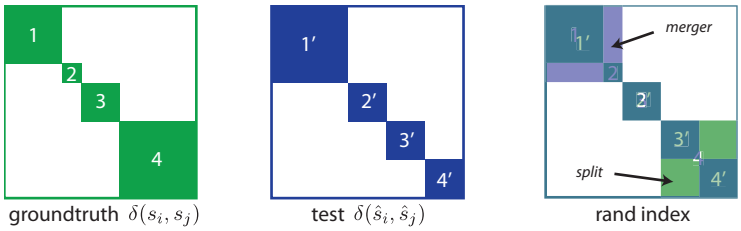

Figure 2: **The Rand index quantifies segmentation performance by comparing the difference in pixel pair connectivity between the groundtruth and test segmentations.** Pixel pair connectivities can be visualized as symmetric binary block-diagonal matrices (s_i, s_j). Each diagonal block corresponds to connected pixel pairs belonging to one of the image segments. The Rand index incurs penalties when pixels pairs that must not be connected are connected or vice versa. This corresponds to locations where the two matrices disagree. An erroneous merger of two groundtruth segments incurs a penalty proportional to the product of the sizes of the two segments. Split errors are similarly penalized.

4 Connectivity and maximin affinity

Recall that our segmentation algorithm works by finding connected components of the thresholded affinity graph. Let $\hat{S}$ be the segmentation produced in this way. To apply the Rand index to train our classifier, we need a simple way of relating the indicator function $(\hat{s}_i, \hat{s}_j)$ in the Rand index to classifier output. In other words, we would like a way of characterizing whether two pixels are connected in the thresholded affinity graph.

To do this, we introduce the concept of maximin affinity, which is defined for any pair of pixels in an affinity graph (the definition is generally applicable to any weighted graph). Let A_{kl} be the affinity of pixels k and l. Let $\mathcal{P}_{ij}$ be the set of all paths in the graph that connect pixels i and j. For every path P in $\mathcal{P}_{ij}$, there is an edge (or edges) with minimal affinity. This is written as $\min_{k,l \quad P} A_{kl}$, where $k, l \quad P$ means that the edge between pixels k and l are in the path P.

A maximin path P_{ij} is a path between pixels i and j that maximizes the minimal affinity,

$$P_{ij} = \arg\max_{P \ \mathcal{P}_{ij}} \min_{k,l \quad P} A_{kl} \tag{2}$$

The maximin affinity of pixels i and j is the affinity of the maximin edge, or the minimal affinity of the maximin path,

$$A_{ij} = \max_{P \ \mathcal{P}_{ij}} \min_{k,l \quad P} A_{kl} \tag{3}$$

We are now ready for a trivial but important theorem.

Theorem 1. *A pair of pixels is connected in the thresholded affinity graph if and only if their maximin affinity exceeds the threshold value.*

Proof. By definition, a pixel pair is connected in the thresholded affinity graph if and only if there exists a path between them. Such a path is equivalent to a path in the unthresholded affinity graph for which the minimal affinity is above the threshold value. This path in turn exists if and only if the maximin affinity is above the threshold value. □

As a consequence of this theorem, pixel pairs can be classified as connected or disconnected by thresholding maximin affinities. Let $\hat{S}$ be the segmentation produced by thresholding the affinity graph A_{ij} and then finding connected components. Then the connectivity indicator function is

$$(\hat{s}_i, \hat{s}_j) = H(A_{ij} \quad) \tag{4}$$

where H is the Heaviside step function.

Maximin affinities can be computed efficiently using minimum spanning tree algorithms [8]. A maximum spanning tree is equivalent to a minimum spanning tree, up to a sign change of the weights.

Any path in a maximum spanning tree is a maximin path. For our nearest neighbor affinity graphs, the maximin affinity of a pixel pair can be computed in $O(|E| \cdot \alpha(|V|))$ where $|E|$ is the number of graph edges and $|V|$ is the number of pixels and $\alpha(\cdot)$ is the inverse Ackerman function which grows sub-logarithmically. The full matrix A^*_{ij} can be computed in time $O(|V|^2)$ since the computation can be shared. Note that maximin affinities are required for training, but not testing. For segmenting the image at test time, only a connected components computation need be performed, which takes time linear in the number of edges $|E|$.

5 Optimizing the Rand index by learning maximin affinities

Since the affinities and maximin affinities are both functions of the image I and the classifier parameters W, we will write them as $A_{ij}(I;W)$ and $A^*_{ij}(I;W)$, respectively. By Eq. (4) of the previous section, the Rand index of Eq. (1) takes the form

$$1 - RI(S,I;W) = \binom{N}{2}^{-1} \sum_{i<j} \Big| \delta(s_i,s_j) - H(A^*_{ij}(I;W) - \theta) \Big|$$

Since this is a discontinuous function of the maximin affinities, we make the usual relaxation by replacing $|\delta(s_i,s_j) - H(A^*_{ij}(I;W) - \theta)|$ with a continuous loss function $l(\delta(s_i,s_j), A^*_{ij}(I;W))$. Any standard loss such as the such as the square loss, $\frac{1}{2}(x-\hat{x})^2$, or the hinge loss can be used for $l(x,\hat{x})$. Thus we obtain a cost function suitable for gradient learning,

$$\begin{aligned} E(S,I;W) &= \binom{N}{2}^{-1} \sum_{i<j} l(\delta(s_i,s_j), A^*_{ij}(I;W)) \\ &= \binom{N}{2}^{-1} \sum_{i<j} l(\delta(s_i,s_j), \max_{P \in \mathcal{P}_{ij}} \min_{\langle k,l \rangle \in P} A_{kl}(I;W)) \end{aligned} \tag{5}$$

The max and min operations are continuous and differentiable (though not continuously differentiable). If the loss function l is smooth, and the affinity $A_{kl}(I;W)$ is a smooth function, then the gradient of the cost function is well-defined, and gradient descent can be used as an optimization method.

Define $(k,l) = mm(i,j)$ to be the maximin edge for the pixel pair (i,j). If there is a tie, choose between the maximin edges at random. Then the cost function takes the form

$$E(S,I;W) = \binom{N}{2}^{-1} \sum_{i<j} l(\delta(s_i,s_j), A_{mm(i,j)}(I;W))$$

It's instructive to compare this with the cost function for standard affinity learning

$$E_{standard}(S,I;W) = \frac{2}{cN} \sum_{\langle i,j \rangle} l(\delta(s_i,s_j), A_{ij}(I;W))$$

where the sum is over all nearest neighbor pixel pairs $\langle i,j \rangle$ and c is the number of nearest neighbors [9]. In contrast, the sum in the MALIS cost function is over all pairs of pixels, whether or not they are adjacent in the affinity graph. Note that a single edge can be the maximin edge for multiple pairs of pixels, so its affinity can appear multiple times in the MALIS cost function. Roughly speaking, the MALIS cost function is similar to the standard cost function, except that each edge in the affinity graph is weighted by the number of pixel pairs that it causes to be incorrectly classified.

6 Online stochastic gradient descent

Computing the cost function or its gradient requires finding the maximin edges for all pixel pairs. Such a batch computation could be used for gradient learning. However, online stochastic gradient

learning is often more efficient than batch learning [13]. Online learning makes a gradient update of the parameters after each pair of pixels, and is implemented as described in the box.

Maximin affinity learning	**Standard affinity learning**
1. Pick a random pair of (not necessarily nearest neighbor) pixels i and j from a randomly drawn training image I.	1. Pick a random pair of nearest neighbor pixels i and j from a randomly drawn training image I
2. Find a maximin edge $mm(i,j)$ 3. Make the gradient update: $W \leftarrow W + \eta \frac{d}{dW} l(\delta(s_i, s_j), A_{mm(i,j)}(I;W))$	2. Make the gradient update: $W \leftarrow W + \eta \frac{d}{dW} l(\delta(s_i, s_j), A_{ij}(I;W))$

For comparison, we also show the standard affinity learning [9]. For each iteration, both learning methods pick a random pair of pixels from a random image. Both compute the gradient of the weight of a single edge in the affinity graph. However, the standard method picks a nearest neighbor pixel pair and trains the affinity of the edge between them. The maximin method picks a pixel pair of arbitrary separation and trains the minimal affinity on a maximin path between them.

Effectively, our connected components performs spatial integration over the nearest neighbor affinity graph to make connectivity decisions about pixel pairs at large distances. MALIS trains these global decisions, while standard affinity learning trains only local decisions. MALIS is superior because it truly learns segmentation, but this superiority comes at a price. The maximin computation requires that on each iteration the affinity graph be computed for the whole image. Therefore it is slower than the standard learning method, which requires only a local affinity prediction for the edge being trained. Thus there is a computational price to be paid for the optimization of a true segmentation error.

7 Application to electron microscopic images of neurons

7.1 Electron microscopic images of neural tissue

By 3d imaging of brain tissue at sufficiently high resolution, as well as identifying synapses and tracing all axons and dendrites in these images, it is possible in principle to reconstruct connectomes, complete "wiring diagrams" for a brain or piece of brain [19, 4, 21]. Axons can be narrower than 100 nm in diameter, necessitating the use of electron microscopy (EM) [19]. At such high spatial resolution, just one cubic millimeter of brain tissue yields teravoxel scale image sizes. Recent advances in automation are making it possible to collect such images [19, 4, 21], but image analysis remains a challenge. Tracing axons and dendrites is a very large-scale image segmentation problem requiring high accuracy. The images used for this study were from the inner plexiform layer of the rabbit retina, and were taken using Serial Block-Face Scanning Electron Microscopy [5]. Two large image volumes of 100^3 voxels were hand segmented and reserved for training and testing purposes.

7.2 Training convolutional networks for affinity classification

Any classifier that is a smooth function of its parameters can be used for maximin affinity learning. We have used convolutional networks (CN), but our method is not restricted to this choice. Convolutional networks have previously been shown to be effective for similar EM images of brain tissue [11].

We trained two identical four-layer CNs, one with standard affinity learning and the second with MALIS. The CNs contained 5 feature maps in each layer with sigmoid nonlinearities. All filters in the CN were $5 \times 5 \times 5$ in size. This led to an affinity classifier that uses a $17 \times 17 \times 17$ cubic image patch to classify a affinity edge. We used the square-square loss function $l(x, \hat{x}) = x \cdot \max(0, 1 - \hat{x} - m)^2 + (1 - x) \cdot \max(0, \hat{x} - m)^2$, with a margin $m = 0.3$.

As noted earlier, maximin affinity learning can be significantly slower than standard affinity learning, due to the need for computing the entire affinity graph on each iteration, while standard affinity training need only predict the weight of a single edge in the graph. For this reason, we constructed a proxy training image dataset by picking all possible $21 \times 21 \times 21$ sized overlapping sub-images

from the original training set. Since each $21 \times 21 \times 21$ sub-image is smaller than the original image, the size of the affinity graph needed to be predicted for the sub-image is significantly smaller, leading to faster training. A consequence of this approximation is that the maximum separation between image pixel pairs chosen for training is less than about 20 pixels. A second means of speeding up the maximin procedure is by pretraining the maximin CN for 500,000 iterations using the fast standard affinity classification cost function. At the end, both CNs were trained for a total of 1,000,000 iterations by which point the training error plateaued.

7.3 Maximin learning leads to dramatic improvement in segmentation performance

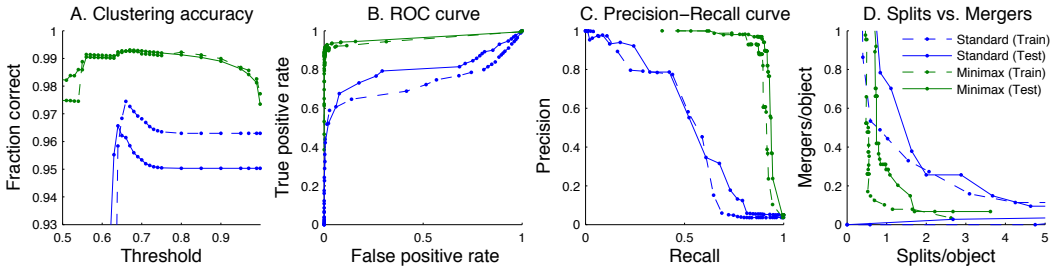

Figure 3: **Quantification of segmentation performance on 3d electron microscopic images of neural tissue.** A) Clustering accuracy measuring the number of correctly classified pixel pairs. B) and C) ROC curve and precision-recall quantification of pixel-pair connectivity classification shows near perfect performance. D) Segmentation error as measured by the number of splits and mergers.

We benchmarked the performance of the standard and maximin affinity classifiers by measuring the the pixel-pair connectivity classification performance using the Rand index. After training the standard and MALIS affinity classifiers, we generated affinity graphs for the training and test images. In principle, the training algorithm suggests a single threshold for the graph partitioning. In practice, one can generate a full spectrum of segmentations leading from over-segmentations to under-segmentations by varying the threshold parameter. In Fig. 3, we plot the Rand index for segmentations resulting from a range of threshold values.

In images with large numbers of segments, most pixel pairs will be disconnected from one another leading to a large imbalancing the number of connected and disconnected pixel pairs. This is reflected in the fact that the Rand index is over 95% for both segmentation algorithms. While this imbalance between positive and negative examples is not a significant problem for training the affinity classifier, it can make comparisons between classifiers difficult to interpret. Instead, we can use the ROC and precision-recall methodologies, which provide for accurate quantification of the accuracy of classifiers even in the presence of large class imbalance. From these curves, we observe that our maximin affinity classifier dramatically outperforms the standard affinity classifier.

Our positive results have an intriguing interpretation. The poor performance of the connected components when applied to a standard learned affinity classifier could be interpreted to imply that 1) a local classifier lacks the context important for good affinity prediction; 2) connected components is a poor strategy for image segmentation since mistakes in the affinity prediction of just a few edges can merge or split segments. On the contrary, our experiments suggest that when trained properly, thresholded affinity classification followed by connected components can be an extremely competitive method of image segmentations.

8 Discussion

In this paper, we have trained an affinity classifier to produce affinity graphs that result in excellent segmentations when partitioned by the simple graph partitioning algorithm of thresholding followed by connected components. The key to good performance is the training of a segmentation-based cost function, and the use of a powerful trainable classifier to predict affinity graphs. Once trained, our segmentation algorithm is fast. In contrast to classic graph-based segmentation algorithms where

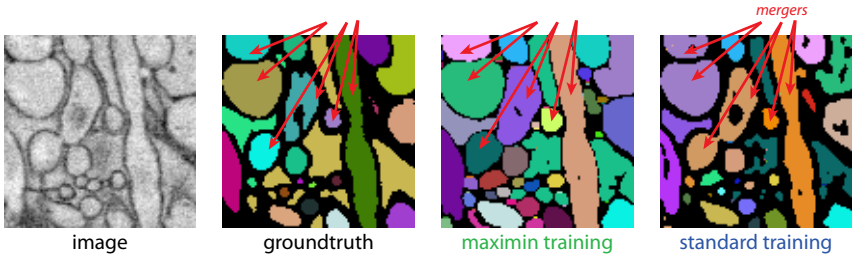

Figure 4: **A 2d cross-section through a 3d segmentation of the test image.** The maximin segmentation correctly segments several objects which are merged in the standard segmentation, and even correctly segments objects which are missing in the groundtruth segmentation. Not all segments merged in the standard segmentation are merged at locations visible in this cross section. Pixels colored black in the machine segmentations correspond to pixels completely disconnected from their neighbors and represent boundary regions.

the partitioning phase dominates, our partitioning algorithm is simple and can partition graphs in time linearly proportional to the number of edges in the graph. We also do not require any prior knowledge of the number of image segments or image segment sizes at test time, in contrast to other graph partitioning algorithms [7, 20].

The formalism of maximin affinities used to derive our learning algorithm has connections to single-linkage hierarchical clustering, minimum spanning trees and ultrametric distances. Felzenszwalb and Huttenlocher [7] describe a graph partitioning algorithm based on a minimum spanning tree computation which resembles our segmentation algorithm, in part. The Ultrametric Contour Map algorithm [1] generates hierarchical segmentations nearly identical those generated by varying the threshold of our graph partitioning algorithm. Neither of these methods incorporates a means for learning from labeled data, but our work shows how the performance of these algorithms can be improved by use of our maximin affinity learning.

Acknowledgements

SCT and HSS were supported in part by the Howard Hughes Medical Institute and the Gatsby Charitable Foundation.

References

[1] P. Arbelaez. Boundary extraction in natural images using ultrametric contour maps. *Proc. POCV*, 2006.

[2] F. Bach and M. Jordan. Learning spectral clustering, with application to speech separation. *The Journal of Machine Learning Research*, 7:1963–2001, 2006.

[3] Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 23(11):1222–1239, 2001.

[4] K. L. Briggman and W. Denk. Towards neural circuit reconstruction with volume electron microscopy techniques. *Curr Opin Neurobiol*, 16(5):562–70, 2006.

[5] W. Denk and H. Horstmann. Serial block-face scanning electron microscopy to reconstruct three-dimensional tissue nanostructure. *PLoS Biol*, 2(11):e329, 2004.

[6] P. Dollár, Z. Tu, and S. Belongie. Supervised learning of edges and object boundaries. In *CVPR*, June 2006.

[7] P. Felzenszwalb and D. Huttenlocher. Efficient Graph-Based Image Segmentation. *International Journal of Computer Vision*, 59(2):167–181, 2004.

[8] B. Fischer, V. Roth, and J. Buhmann. Clustering with the connectivity kernel. In *Advances in Neural Information Processing Systems 16: Proceedings of the 2003 Conference*. Bradford Book, 2004.

[9] C. Fowlkes, D. Martin, and J. Malik. Learning affinity functions for image segmentation: combining patch-based and gradient-based approaches. *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*, 2, 2003.

[10] X. He, R. Zemel, and M. Carreira-Perpinan. Multiscale conditional random fields for image labeling. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2. IEEE Computer Society; 1999, 2004.

[11] V. Jain, J. Murray, F. Roth, S. Turaga, V. Zhigulin, K. Briggman, M. Helmstaedter, W. Denk, and H. Seung. Supervised learning of image restoration with convolutional networks. *ICCV 2007*, 2007.

[12] S. Kumar and M. Hebert. Discriminative random fields: a discriminative framework for contextual interaction in classification. *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*, pages 1150–1157, 2003.

[13] Y. LeCun, L. Bottou, G. Orr, and K. Müller. Efficient backprop. *Lecture notes in computer science*, pages 9–50, 1998.

[14] M. Maire, P. Arbelaez, C. Fowlkes, and J. Malik. Using contours to detect and localize junctions in natural images. In *IEEE Conference on Computer Vision and Pattern Recognition, 2008. CVPR 2008*, pages 1–8, 2008.

[15] 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. Eighth Int'l Conf. Computer Vision*, volume 2, pages 416–423, 2001.

[16] D. R. Martin, C. C. Fowlkes, and J. Malik. Learning to detect natural image boundaries using local brightness, color, and texture cues. *IEEE Trans Pattern Anal Mach Intell*, 26(5):530–549, May 2004.

[17] M. Meila and J. Shi. Learning segmentation by random walks. *ADVANCES IN NEURAL INFORMATION PROCESSING SYSTEMS*, pages 873–879, 2001.

[18] W. Rand. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical association*, pages 846–850, 1971.

[19] H. Seung. Reading the Book of Memory: Sparse Sampling versus Dense Mapping of Connectomes. *Neuron*, 62(1):17–29, 2009.

[20] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000.

[21] S. J. Smith. Circuit reconstruction tools today. *Curr Opin Neurobiol*, 17(5):601–608, Oct 2007.

[22] Z. Tu, X. Chen, A. Yuille, and S. Zhu. Image parsing: Unifying segmentation, detection, and recognition. *International Journal of Computer Vision*, 63(2):113–140, 2005.

[23] R. Unnikrishnan, C. Pantofaru, and M. Hebert. Toward objective evaluation of image segmentation algorithms. *IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE*, pages 929–944, 2007.