E4 project : Maximin Affinity Learning of Image Segmentation

Quentin Garrido, Tiphanie Lamy Verdin, Josselin Lefèvre, Annie Lim

January 2020

Contents

1 Introduction

2 Theoretical background

3 Maximin Affinity learning

3.1 Method's presentation

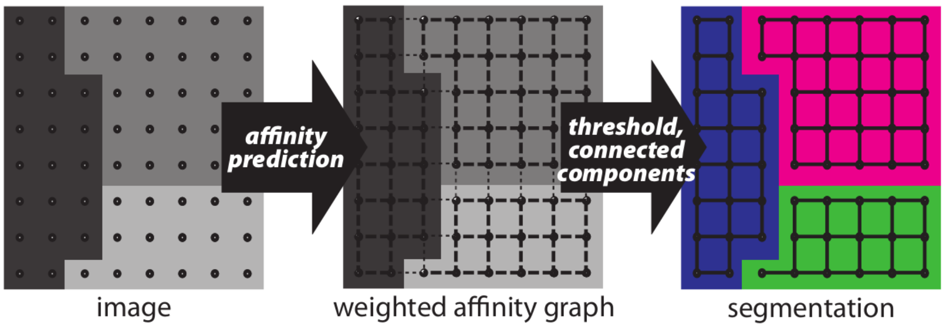

Figure 1: Description of the method, from [1]

The method, as illustrated in 1 works in two steps.
The first step is to compute an affinity graph G, with nodes V representing each pixel in the original image and edges E weighted by the affinity between neighbouring pixels. The graph G will be 4-connected and for every pair of pixels $i, j \in V \times V$ there exist and edge $(i, j) \in E$ iff i and j are neighbours. We will note the affinity between neigbouring pixels A_{ij}.
An affinity of 1 between i and j means that they belong to the same object, and an affinity of 0 means that they belong to different objetcs.

Once we have our affinity graph, with affinities between 0 and 1, we will threshold it and remove edges under a certain affinity to obtain connected components that will be our objects, and this thresholded affinity graph will be our final segmentation.

3.2 Optimizing the Rand Index

There exist various methods of evaluating an image segmentation, one of them being the Rand Index which is defiend as follow :

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

With S our groundtruth segmentatiin $\hat{S}$ our predicted segmentation and $\delta(s_i, s_j)$ the indicator function taking value 1 if $s_i = s_j$ (if pixels i and j are in the same segment/object) and 0 otherwise.
Intuitively, this can be seen as the fraction of image pixels where both segmentations agree.
This gives us a way to evaluate an image segmentation that penalizes when two objects are merged or split in our final segmentation as we can see in figure 2. On the left, only one edge is misclassified but this merges two objects and will be heavily penalized by the Rand Index (we would see the same results if an object was split). On the contrary if we have misclassified edges inside of objects that do not create any splits, this will not be penalized at all by the loss since our objects are still correctly delimited.

As such the Rand Index seems a good measure of segmentation quality for our problem (edge-based segmentation). We will have to see how exactly can we optimise this metric in our case.

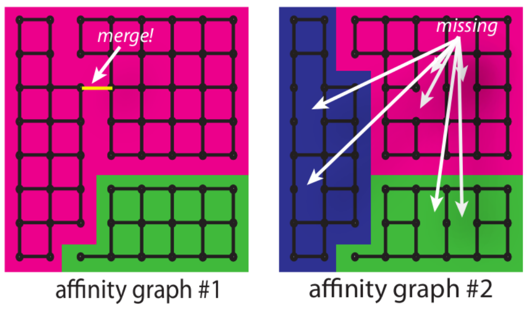

Figure 2: Exemple image segmentations and their affinity graph. On the left we have a merging of two objects due to a misclassified edge and on the right we have edges that should be of value 1 that have been removed. From [1]

3.3 Maximin affinity and the maximin edge

In order to optmize the Rand Index, we must find a way to compute $\delta(\hat{s}_i, \hat{s}_j)$. To to this we will use the concept of maximin affinity, as described in [1].
Let $\mathcal{P}_{ij}$ the set of all paths between i and j in our image. For every path $P \in \mathcal{P}_{ij}$ there is an edge(s) (k, l) with minimal affinity.

This allows us to define the maximin path, which is th path which maximizes the mininal affinity. It is defined as follow :

$$P^*_{ij} = \arg \max_{P \in \mathcal{P}_{ij}} \min_{(k,l) \in P} A_{kl}$$

The edge of minimal affinity in the maximin path will be called the maximin edge and will be writter $mm(i, j)$.
It allows to define the maximin affinity which is simply the affinity of the maximin edge.

$$A^*_{ij} = \max_{P \in \mathcal{P}_{ij}} \min_{(k,l) \in P} A_{kl}$$

The most important consequence of this is that a pair of pixels i, j in connected in the thresholded affinity graph iff A^*_{ij} is greater than the threshold value, as shown in [1].

This means that if we can find the maximin affinity efficiently, we will be able to compute the Rand Index efficiently.
In practice the maximin affinities can bee computed efficiently using a maximum spanning tree (MST), since any path in the MST is a maximin path.

3.4 Computing an affinity graph

As we can see, once we have our affinty graph, obtaining the segmentation is straightforward, but the issue is : How to obtain an affinity graph?
As stated before, an affinity graph is equivalent to the contours in the image, and as such any method to find the contours in an image can be used to find the affinity graph (with relative success).
The first idea that we could have would be to use a contour detector, such as the Canny filter for example, however this would lead to an obvious over segmentation, which is not desirable here.

An other idea (the idea used in [1]) would be to use a neural network to obtain our affinity graph. Neural networks, and particularly convolutionnal neural networks (CNNs) have proven to be an extremely powerful tool in image processing and especially in image classification and segmentation, which makes them a good candidate for MALIS.

The architecture used originally is a fully convolutionnal neural network (FCNN) which is a CNN with only convolutionnal layers. As such they can work with various input sizes which is always a nice feature.

Now that we have seen how we can compute an affinity graph, let's see how we can train this classifier, and if it even possible to train it with the Rand Index as a loss function.

3.5 Training a classifier

As stated before, the goal of this method is to optimise the Rand Index, defined as:

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

with S our groudtruth and $\hat{S}$ our predicted segmentation.
We can define $A^*_{ij}(I, \theta)$ the maximin affinity of pixels i and j in the predicted affinty graph of our classifier on I with parameters θ. We can rewrite the previous equation as :

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

However this function is not differentiable everywhere so we cannot use it directly for our training. We will replace the absolute value with a smooth loss function l such as the mean squared error. This relaxation will allow us to use gradient descent to train our classifier(refer to [1] for more details on this).

Thus our final loss function will be :

$$L(I, \theta, S) = \binom{N}{2}^{-1} \sum_{i<j} l(\delta(s_i, s_j), A^*_{ij}(I, \theta)) = \binom{N}{2}^{-1} \sum_{i<j} l(\delta(s_i, s_j), A_{mm(i,j)}(I, \theta))$$

We can now look at what the training loop will look like.

Algorithm 1 MALIS training loop, from [1]

```
1: repeat
2:     Predict the affinity graph for an image $I$
3:     Pick two random pixels $i, j$ from $I$
4:     Find the maximin edge $mm(i, j)$
5:     Update our parameters $\theta$ with gradient
           $\nabla l(\delta(s_i, s_j), A_{mm(i,j)}(I, \theta)$
6: until convergence is reached
```

In practice we won't train on the whole image since it would be too long to compute the maximin edge in the whole image. We will instead train on 21x21 patches of the image. We have to choose these patches carefully as most of them will mostly be in an object and not around a border.
Since borders between objects are far less common than the "inside" of an object, we may most of the time train only for pixels inside a same object.
This would result in class imbalance between our "borders" (affinity of value 0) and or "inside" (affinity of value 1). Training with such imbalance would lead to worsened results since borders would not be present most of the time in the chosen image.(one example of such pathological behaviour would be the prediction of affinty 1 everywhere, far from what we desire)

As such when training we need to be careful about what training image we select.

4 Implementation

4.1 Computing the affinity graph

4.2 Training the neural network

5 Results

5.1 Evaluation method

5.2 CREMI

5.3 ISBI 2012

Layer	Kernel	Strides	Features	BN	Activation	Output shape
Input						(21, 21, 1)
Convolution	(5, 5)	(1, 1)	8	Y	ReLU	(21, 21, 8)
Convolution	(5, 5)	(1, 1)	32	Y	ReLU	(21, 21, 32)
Convolution	(5, 5)	(1, 1)	32	Y	ReLU	(21, 21, 32)
Convolution	(5, 5)	(1, 1)	32	Y	ReLU	(21, 21, 32)
Convolution	(5, 5)	(1, 1)	8	Y	ReLU	(21, 21, 8)
Convolution	(5, 5)	(1, 1)	2	N	sigmoid	(21, 21, 2)

6 Going forward

7 Teamwork

7.1 Team organization

7.2 Task distribution

7.3 Obstacles and overcoming them

8 Conclusion

References

[1] S. C. Turaga, K. L. Briggman, M. Helmstaedter, W. Denk, and H. S. Seung, "Maximin affinity learning of image segmentation," *arXiv:0911.5372 [cs]*, Nov. 2009. arXiv: 0911.5372.

[2] L. Najman, J. Cousty, and B. Perret, "Playing with Kruskal: Algorithms for Morphological Trees in Edge-Weighted Graphs," in *Mathematical Morphology and Its Applications to Signal and Image Processing*, vol. 7883, pp. 135–146, Berlin, Heidelberg: Springer Berlin Heidelberg, 2013.

[3] G. Chierchia and B. Perret, "Ultrametric Fitting by Gradient Descent," *arXiv:1905.10566 [cs, stat]*, Oct. 2019. arXiv: 1905.10566.

[4] A. Challa, S. Danda, B. S. D. Sagar, and L. Najman, "Watersheds for Semi-Supervised Classification," *IEEE Signal Processing Letters*, vol. 26, pp. 720–724, May 2019.

[5] J. Funke, F. Tschopp, W. Grisaitis, A. Sheridan, C. Singh, S. Saalfeld, and S. C. Turaga, "Large Scale Image Segmentation with Structured Loss Based Deep Learning for Connectome Reconstruction," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 41, pp. 1669–1680, July 2019.

[6] S. Wolf, L. Schott, U. Köthe, and F. Hamprecht, "Learned Watershed: End-to-End Learning of Seeded Segmentation," *arXiv:1704.02249 [cs]*, Sept. 2017. arXiv: 1704.02249.

[7] S. Turaga, "Learning image segmentation and hierarchies by learning ultrametric distances," Apr. 2010.