Graphical Models: Discrete Inference and Learning
Graph-cut algorithm for image segmentation
Alpha-expansion extension

Maxence Gélard
CentraleSupélec
MVA
maxence.gelard@student-cs.fr

1. Introduction

For the Graphical Models course, I have chosen to work on a graph cut algorithm that aims at solving the image segmentation problem. Such application consists in being able to assign a given label to every pixel of an image, thus allowing to divide an image in semantic regions (e.g separate an object in the foreground from the background of an image). Such segmentation problem can be expressed as an optimisation problem which corresponds to an energy minimisation.

Under certain hypothesis, namely described in [3], such energy minimisation can be performed through graph cut techniques, which are "training-free" methods. Thus, this supposes to transform a given image into a graph and define a weighted set of edges in order to perform graph cut.

In this project, we will be working first on the binary segmentation problem, which aims at solving the object / background separation. More precisely, we will be using an interactive segmentation technique that uses graph cut, i.e some pixels will be manually chosen to be respectively part of the object and part of the background. We will thus in the next part formalise the problem of binary segmentation and formulate it as an energy minimisation problem that can be solved through graph cut.

We will then extend the binary segmentation graph cut algorithm to the case of multi label segmentation. To do so, an iterative algorithm called *alpha expansion*, derived from the binary case, will be presented. The main idea is to perform successive binary decision for each pixels to determine whether it gets a new label or keep its current one.

After the presentation of the different methods, we will evaluate the algorithms on a sample of the *COCO* dataset from *Microsoft* [7]. A comparison with a deep learning model will also be performed.

2. Problem formalisation

2.1. Graph cut - Energy minimisation

First, let's derive the setting to perform a binary segmentation of an image using graph cut. First, we aim at representing an image as a directed graph $\mathcal{G} = (\mathcal{P}, \mathcal{N})$, with $\mathcal{P}$ a set of vertices corresponding to each pixel of the image, and $\mathcal{N}$ the set of edges in the graph. For the problem we are tackling, we will be using a *4-connected* graph, i.e each pixel will be connected to at most 4 others pixels (up, down, left, right), handling the boundary cases.

Let's now motivate the need of the edges weighting that will allow us in the end to perform graph cut. Let's consider a labeling matrix A, i.e a matrix with the same shape as the image, with for example an entry being 0 for an object and 1 for the background. Solving the binary segmentation issue can expressed as the minimisation of the following energy, which correspond to a first order approximation Markov Random Field formulation (we keep the same notation as in [3] to have easier references to it):

$$E(A) = \sum_{p \in \mathcal{V}} R_p(A_p) + \sum_{p,q \in \mathcal{N}} B_{p,q}(A_p, A_q) \mathbb{1}_{A_p \neq A_q}$$

In this expression, the unary term R_p can be seen as a data attachment term, as it will aim at penalising a pixel label if it disagrees with the observed data (we will see later that it can be expressed as a distance with the pixel and a set of pixels considered to have a given label). The binary term $B_{p,q}$ is a penalty term for similar pixels that would have different labels (that will force a form a smoothness in the label assignment A).

Such unary and binary terms will be used later to set weights on the graph edges. Graph cut techniques can be used to find the global minimum for this kind of energy function. To this end, we first need to recall the equivalence theorem that in a graph, finding a minimum cut is equiva-

lent to solve a maximum flow problem. Thus, we will be introducing two nodes T and S (sink and source nodes), that will respectively represent the object/foreground and the background regions of the image.

So the set of edges $\mathcal{E}$ that will be finally be used is $\mathcal{N}$, to which are added for every node $v \in \mathcal{P}$ two edges (S, v) and (T, v). The binary terms will thus correspond to edges weights between nodes of the graph that are pixels of the images, and the unary terms will the used to defined the weights between every pixel and the sink and source nodes.

The last required step will be to define pixels that are considered to represented respectively the object and the background. To this end, we will manually pick a set $\mathcal{O}$ of pixels for the foreground and a set $\mathcal{B}$ of pixels for the background.

This final setting is well summarized in the following table which is extracted from [3]:

edge	weight (cost)	for
$\{p, q\}$	$B_{\{p,q\}}$	$\{p, q\} \in \mathcal{N}$
$\{p, S\}$	$\lambda \cdot R_p$("bkg")	$p \in \mathcal{P},\ p \notin \mathcal{O} \cup \mathcal{B}$
	K	$p \in \mathcal{O}$
	0	$p \in \mathcal{B}$
$\{p, T\}$	$\lambda \cdot R_p$("obj")	$p \in \mathcal{P},\ p \notin \mathcal{O} \cup \mathcal{B}$
	0	$p \in \mathcal{O}$
	K	$p \in \mathcal{B}$

Figure 1. Edges weights for graph cut

In this table, we namely define a parameter λ which will control the importance of the data attachment term. We also define the unary weight K between source/sink nodes and the manually chosen pixels by:

$$K = 1 + max_{p \in \mathcal{P}} \sum_{q \in \mathcal{P}:(p,q) \in \mathcal{N}} B_{p,q}$$

Given this graph setting, we end up with the following s-t graph (from [3]:

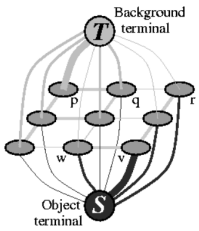

Figure 2. s-t graph for binary segmentation

2.2. Weights functions

In the previous subsection, we defined the binary segmentation problem as a graph cut problem by considering general unary and binary weights. In this section, we define the form of the weights that I have been trying in this project.

Unary potentials

In order to define the unary potentials, we recall that we have been interactively selecting two sets of pixels $\mathcal{O}$ and $\mathcal{B}$ respectively corresponding to the foreground (object) and the background. Then, our image, given in *RGB* format, was first converted to the *YUV* format as it is generally a more efficient coding. Then from $\mathcal{O}$ and $\mathcal{B}$, we were able to compute the mean and the covariance matrix of each set of pixels. From that, two type of unary potentials have been tried:

- L_2**-distance**: $R_p(label) = ||p - \mu_{label}||^2$, with p the current pixel, *label* either designated the foreground or the background, and μ_{label} the mean of the distribution of the pixels assigned to this label.
- **Normal density**: $R_p(label) = \mathcal{N}(p; \mu_{label}, \Sigma_{label})$, with $\mathcal{N}$ the normal distribution, and Σ_{label} the covariance matrix of the distribution of the pixels assigned to this label.

Binary potentials

The binary potentials must be positive weights and such that two similar pixels will be assigned a smaller weight and conversely. Therefore, I have chosen the following form for the binary potentials:

$$B_{p,q}(A_p, A_q) = \omega e^{-\frac{||p-q||^2}{2\sigma^2}} \mathbb{1}_{A_p \neq A_q}$$

with, for a pixel p, x_p corresponding to its position in the image so a tuple $(x_{coordinate}, y_{coordinate})$; w a weighting parameters (same as for λ) ; and σ a band-width.

One should pay attention, as described in [3], that the weights should be positive and define a submodular function, which is the case here. Indeed:

$$0 = B_{p,q}(0,0) + B_{p,q}(1,1) \leq B_{p,q}(0,1) + B_{p,q}(1,0)$$

These conditions allow to find a global optimum of the energy through the graph cut method.

2.3. Running max-flow and get results

Until then, we have constructed an s-t graph by setting on the edges the unary and binary terms. To find a minimal cut and thus labels assignments, we will use the equivalence min-cut / max-flow. Thus, we will be using the Ford-Fulkerson to solve the maximum flow problem. This algorithm, for example described in [1], can be summarized as follow (taken from the lecture by K. Alahari):

- Start with a flow of 0 on all edges
- Until the sink and the source are disjoint in the residual graph:
  - Find an s-t path in the residual graph
  - On this path, use all maximum allowable flow
  - Subtract this flow from the inverse arcs and add it to the forward arcs.

With the residual graph being the graph that only contain the arcs whose current flow is strictly less than its capacity.

Once we have solved the max-flow problem, thanks to the max-flow / min-cut correspondence, we get a cut $\mathcal{C} \subset \mathcal{E}$ that will separate the graph in two subsets of the nodes of the graph in S and T. We then derive the labeling for a given $v \in \mathcal{N}$

- If $v \in S$, $A_s = 1$ (background)
- If $v \in T$, $A_s = 0$ (foreground)

3. Alpha expansion extension

So far, the graph cut method we have been presenting only allows to perform binary image segmentation, for example to separate an object from the rest of the image (foreground / background separation). In this section, we present a extension of the graph cut method in the case of multi-label image segmentation. This method, called **alpha expansion** is presented in [1].

The alpha-expansion algorithm is an iterative process which can be summarized as follows:

- Initialize the labels (random / minimum unary cost through the labels ...)
- Select a class α
- Solve a binary segmentation problem: for each pixel, the graph-cut output will decide whether it keeps its current label or changes to class α.

Multiple iterations of this algorithm can be done until the energy stops decreasing for example.

Globally, the structure of the graph will remain the same. After we have found the cut, the sink and source node, that we previously connected to "foreground" and "background" pixels, will now be connected to pixels that will take label α and the one that will keep their label (often referred as $\bar{\alpha}$).

However, a slight change is provided in the paper [2] by introducing an auxiliary node between two neighbors that have different labels so that the new edges created between the two neighbors and the auxiliary nodes (respectively $e_{p,a}$ and $e_{q,a}$) will have a weight corresponding to the binary decision "α or $\bar{\alpha}$". Formally, for two nodes $p, q \in \mathcal{P}$, if $A_p \neq A_q$, we will introduce a new node a_{pq}, so that, for a current label α, we get the following new set of nodes:

$$\mathcal{V}_\alpha = \left\{\mathcal{P}, \alpha, \bar{\alpha}, \{a_{pq}\}_{A_p \neq A_q}\right\}$$

The last step after having introduced the auxiliary nodes and added the corresponding edges is to add an edge between all auxiliary node and the terminal node $\bar{\alpha}$. An equivalent table as the one given in Figure 1 is provided by [2] to summarize the weights given to all the different types of edges:

edge	weight	for
$t_p^{\bar{\alpha}}$	∞	$p \in \mathcal{P}_\alpha$
$t_p^{\bar{\alpha}}$	$D_p(f_p)$	$p \notin \mathcal{P}_\alpha$
t_p^{α}	$D_p(\alpha)$	$p \in \mathcal{P}$
$e_{\{p,a\}}$	$V(f_p, \alpha)$	$\{p,q\} \in \mathcal{N},\ f_p \neq f_q$
$e_{\{a,q\}}$	$V(\alpha, f_q)$	
$t_a^{\bar{\alpha}}$	$V(f_p, f_q)$	
$e_{\{p,q\}}$	$V(f_p, \alpha)$	$\{p,q\} \in \mathcal{N},\ f_p = f_q$

Figure 3. Edges weights for graph cut (alpha-expansion

4. Evaluation

In order to evaluate both algorithm, I have chosen to use samples from *COCO* dataset from *Microsoft* [7]. More precisely, I have been using 25 samples from the validation split, taking images that contains either cats, dogs or both. I have thus constituted a datasets suitable for binary segmentation (images where there is only cats or only dogs) and for multi labels segmentation (image where there are cats and dogs). The restrictions to cats and dogs was only a choice for engineering purposes to handle easily the change from binary to the multi-label cases for the interactive annotations. Below are given samples from this dataset:

Figure 4. Samples from the *COCO* dataset

Each of theses samples comes with segmentation masks and bounding boxes (to locate these masks on the original image). This allows to build a segmentation matrix whose entries correspond to the label of each pixel. Then, after running either the binary segmentation or its alpha-expansion extension, we can compare the ground truth given by the annotations and our segmentation results. To this end, for each class, we will consider the *Intersection over Union* metric, i.e for a given label L:

- We form a set $\mathcal{G}$ of pixels from the ground truth that have label L
- We form a set $\mathcal{P}$ of pixels from the predicted segmentation that have label L
- We compute the *Intersection over Union* as:

$$IoU_L = \frac{|\mathcal{G} \cap \mathcal{P}|}{|\mathcal{G} \cup \mathcal{P}|}$$

Then, we average over all labels to get our final score:

$$IoU = \frac{1}{n_{labels}} \sum_{L \in labels} IoU_L$$

5. Results

In this section, we will present the results we got by applying graph cut both for binary and multi label segmentation. We recall that the first step was to perform an interactive annotation of the image, in order to provide a set of pixels that belongs to each class. I thus have been implementing an interface that allows to annotate images in such way. In Figure 5, we display an annotated image.

In the section on problem formalisation, we introduced unary weights that were related to the mean and optionally to the covariance matrix of the distribution of pixels for the different classes. For visualization purposes, I have been fitting a Gaussian density for each set of pixels and sampled from it to have a plot of the densities. In Figure 6 are given the corresponding densities for the annotated image, with here 3 distribution for the dog, the background and the cat.

Figure 5. Annotation of an image

As mentioned previously, the image has been converted to the *YUV* format, and the densities that are represented on the figure only correspond to the first channel (Y). Notice that values for densities are artificially greater than 1, it is just a normalization induced by the *seaborn* library:

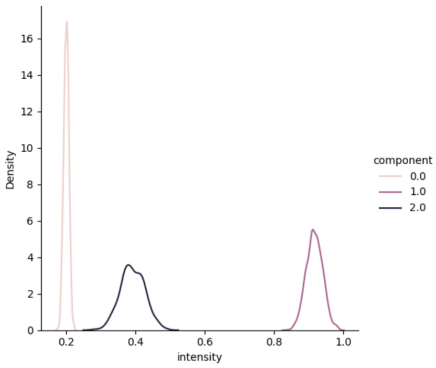

Figure 6. Pixels intensity density for each class

We then run either the binary segmentation or the alpha-expansion algorithm depending on the number of labels, and computed the *IoU* metric for each image. In the following table, we provide the mean and maximum *IoU* for the binary segmentation as well the mean and maximum *IoU* for the multi-class segmentation (in this case a 3 labels segmentation: dog, cat, background), using the L_2 distance as unary potentials.

Segmentation results (L_2 potentials)		
Algorithm	Mean *IoU*	Max *IoU*
Binary	0.50	0.72
α-expansion	0.28	0.49

Additionally, we provide the same results but using the normal distribution as unary potentials:

Segmentation results (normal potentials)		
Algorithm	Mean *IoU*	Max *IoU*
Binary	0.59	0.79
α-expansion	0.33	0.57

One can see that, namely for the α-expansion algorithm, the normal unary potentials perform slightly better. Finally, we provide an example of result (here for α-expansion):

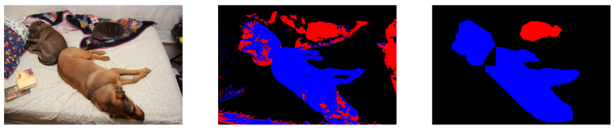

Figure 7. Segmentation results (left: image, center: result, right: ground truth

In order to have an idea of the performance of such approach, I have compared the graph cut method with a Deep Learning one, which is more what is done for segmentation in the state of the art. More precisely, I have been using a Mask R-CNN which is a model that combines image segmentation with Fast-RCNN [5] whose main idea is to propose relevant regions to be classified by the Convolutional Neural Network. More details can be found here [6]. Below is given the table containing the results on the same samples used for the graph-cut methods:

Mask RCNN Segmentation results		
Algorithm	Mean *IoU*	Max *IoU*
Binary	0.58	0.95
α-expansion	0.54	0.92

In these results, not only the segmentation mask is taken into account (i.e we do not only look at if the mask if well located), but the label also. In the deep learning model case, we noticed a few examples on which the mask was correct but the label wrong, hence a sometimes a drop in some *IoU* results.

In addition, we provide a segmentation result for this model (blue mask for dogs and blue mask for cats):

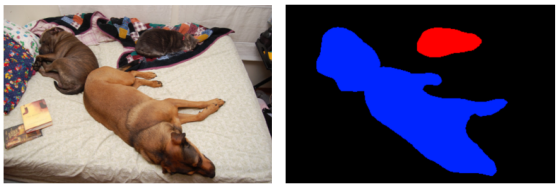

Figure 8. Segmentation prediction Mask RCNN

One can observe that the deep learning model provides better results in terms of *IoU*, with the advantage of directly being able to deal with multiple labels (the separation between binary and α-expansion was just to be able to compare with the graph-cut methods).

6. Conclusion

Finally, I was able to implement a binary segmentation as well as multi-labels segmentation based on the Ford-Fulkerson maximum flow algorithm that allows to find a minimum cut that minimizes an energy function. These algorithms provide quite correct results, and have the advantage of not requiring any training such as segmentation method based on Deep Learning. Nevertheless, this training-free method has the cost of being quite slow as the size of the image increasing, which is namely due to the complexity of the Ford-Fulkerson algorithm which run in a time complexity of $O(m^2U)$ with m the number of arcs in the graph and U the maximum arc length. Another limitation that we found, namely using the COCO dataset, was that the object / foreground and the background often shared similar colors, and thus a trade-off between data attachment (unary terms) and smoothness (binary terms) need to be found and didn't always allow to provide a perfect segmentation. Therefore, such technique seems to be very sensitive to the interactive annotation part.

Possible continuation of the project could be to look at different maximum flow algorithms such that *Edmonds Karp* [8] or *Dinitz* [4] algorithms.

References

[1] Y. Boykov and V. Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *IEEE transactions on pattern analysis and machine intelligence*, 26(9):1124–1137, 2004.

[2] Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. *IEEE Transactions on pattern analysis and machine intelligence*, 23(11):1222–1239, 2001.

[3] Y. Y. Boykov and M.-P. Jolly. Interactive graph cuts for optimal boundary & region segmentation of objects in nd images. In *Proceedings eighth IEEE international conference on computer vision. ICCV 2001*, volume 1, pages 105–112. IEEE, 2001.

[4] Y. Dinitz. Dinitz'algorithm: The original version and even's version. In *Theoretical computer science*, pages 218–240. Springer, 2006.

[5] R. Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015.

[6] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017.

[7] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[8] N. Zadeh. Theoretical efficiency of the edmonds-karp algorithm for computing maximal flows. *Journal of the ACM (JACM)*, 19(1):184–192, 1972.