Fast and Accurate Image Segmentation using Fully Connected Conditional Random Fields

Leonard Bruns <leonardb@kth.se>

January 7, 2020

1 Introduction

In this tutorial we will introduce an efficient method for accurate image segmentation based on conditional random fields. We will start by formally introducing the required definition of conditional random fields in general terms. Afterwards we go over formulating image segmentation as an inference problem in CRFs. We will show that inference in such a graph quickly becomes intractable, which motivates the use of approximate inference methods. In Section 3 we introduce the mean field approximation derive an inference algorithm for fully connected CRFs. As we will see that a straight forward implementation of this equation is still not feasible. Using insights from signal theory we will further simplify the inference algorithm as introduced by Krähenbühl [1].

1.1 Conditional Random Fields

Conditional random fields (CRFs) are undirected probabilistic graphical models. They are by definition discriminative, that is, they model a distribution $p(\boldsymbol{Y}|\boldsymbol{X})$.

Definition 1. Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ in which each vertex $v \in \mathcal{V}$ is assigned to a variable $Y_v \in \boldsymbol{Y}$. Additionally let $\boldsymbol{X}$ be the observed variables and $\mathrm{N}(v)$ and $\mathrm{N}[v]$ denote the open and closed neighborhood of v, respectively. $(\boldsymbol{X}, \boldsymbol{Y})$ is called a *conditional random field* if it obeys the Markov property with respect to the graph $\mathcal{G}$, that is,

$$Y_v \perp \boldsymbol{Y} \setminus \mathrm{N}[v] \,|\, X, \mathrm{N}(v) \quad \forall\, v \in \mathcal{V}. \tag{1}$$

In simple terms, each $Y_v \in \boldsymbol{Y}$ is independent of all its non-neighbors, given its neighbors and $\boldsymbol{X}$. The connectivity of $\mathcal{G}$ implies certain constraints on possible sets of cliques $\mathcal{C}_\mathcal{G}$. To get an actual distribution a set of factors $\Phi = \{\phi_{\boldsymbol{c}} : \boldsymbol{c} \in \mathcal{C}_\mathcal{G}\}$ is required. Each of these factors is a function $\phi_{\boldsymbol{c}} : \boldsymbol{c} \to \mathbb{R}^+$. Note, that the graph alone, does not define a distribution, rather it only constrains the possible sets of cliques $\mathcal{C}_\mathcal{G}$ due to its connectivity.

Given the set of factors Φ and the corresponding set of cliques $\mathcal{C}_\mathcal{G}$ we can write out the Gibbs distribution

$$P(\boldsymbol{Y}|\boldsymbol{X}) = \frac{1}{Z(\boldsymbol{X})} \exp\left(-\sum_{\boldsymbol{c} \in \mathcal{C}_\mathcal{G}} \phi_{\boldsymbol{c}}(\boldsymbol{Y}_{\boldsymbol{c}}|\boldsymbol{X})\right). \tag{2}$$

2 Image Segmentation as Inference in CRFs

Now we can frame the problem of image segmentation as an inference problem in CRFs. Clearly, the observation $\boldsymbol{X}$ in this case are all the pixels of the image, that is, for an image of N pixels $\boldsymbol{X} = \{X_i : 1 \leq i \leq N\}$. The output $\boldsymbol{Y}$ is the segmentation of the image. The domain of each pixelwise label $Y_i, 1 \leq i \leq N$ is $\mathcal{L} = \{l_i, 1 \leq i \leq L\}$, when there are L different labels. Next,

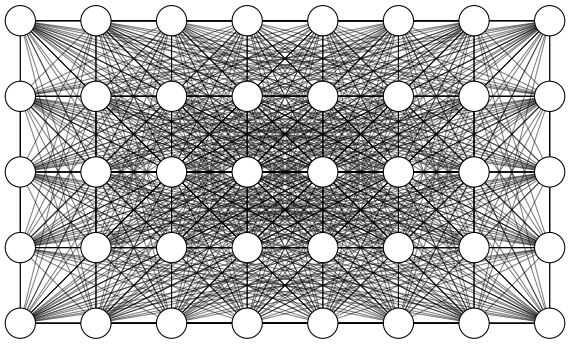


Figure 1: Visualization of the fully connected CRF model. Note, the large number of edges even for such a small image. In general the number of edges is quadratic in the number of pixels N.

we need to define the neighborhood of a pixel. Generally, the more edges we add, the more complex, but also powerful the model will become. Here we will use a fully connected CRF model, so all vertices v have an edge to all other vertices $w \in \mathcal{V} \setminus v$. This allows to capture long distance relations between the pixels. Note, that other approaches to semantic segmentation use different assumptions like 4 or 8 neighborhoods (see, e.g., [2, 3]) to make the inference task feasible. Here we start with a complex model and use approximate inference techniques to make inference tractable. Figure 1 visualizes the model for a small 8×5 image.

Next, we need to define the cliques and factors defining the distribution. Since we use a fully connected CRF, generally all possible subsets of $\boldsymbol{Y}$ could be part of $\mathcal{C}_\mathcal{G}$. Here, we use only cliques comprised of one and two variables

$$\mathcal{C}_\mathcal{G} = \{Y_i : 1 \leq i \leq N\} \cap \big\{\{Y_i, Y_j\} : 1 \leq i \leq N \wedge i < j \leq N\big\}. \tag{3}$$

Hence, the set of factors Φ is comprised of unary (i.e., over a single variable) potentials and pairwise potentials

$$\Phi = \{\phi_{\mathrm{u},i} : \mathcal{L} \to \mathbb{R}^+, 1 \leq i \leq N\} \cap \{\phi_{\mathrm{p},i,j} : \mathcal{L}^2 \to \mathbb{R}^+, 1 \leq i \leq N \wedge i < j \leq N\}. \tag{4}$$

We assume that the unary potentials $\phi_{\mathrm{u},i}$ are given, by manual labeling or any other segmentation method (i.e., a neural network). For the pairwise potentials we assume a sum of weighted Gaussians

$$\phi_{\mathrm{p},i,j}(Y_i, Y_j) = \mu(Y_i, Y_j) \sum_{k=1}^{K} w_k g_k(\boldsymbol{f}_i, \boldsymbol{f}_j). \tag{5}$$

Where $w_k \in \mathbb{R}^+$ are the weights, g_k are Gaussians defined by

$$g_k(\boldsymbol{f}_i, \boldsymbol{f}_j) = \exp\left(-\frac{1}{2}(\boldsymbol{f}_i - \boldsymbol{f}_j)^T \boldsymbol{\Sigma}_k^{-1} (\boldsymbol{f}_i - \boldsymbol{f}_j)\right), \tag{6}$$

where $\boldsymbol{f}_i$ are feature vectors extracted for the pixel $\boldsymbol{X}_i$ (this is why $\boldsymbol{X}$ is implicitly assumed given for all factors). $\mu(Y_i, Y_j)$ is a compatibility function, which models the relation between different labels.

These assumption fully define the Gibbs distribution as given by equation (2)

$$P(\boldsymbol{Y}|\boldsymbol{X}) = \frac{1}{Z(\boldsymbol{X})} \exp\left(-\underbrace{\left(\sum_{i=1}^{N} \phi_{\mathrm{u},i}(Y_i) + \sum_{i=1}^{N} \sum_{j=i+1}^{N} \phi_{\mathrm{p},i,j}(Y_i, Y_j)\right)}_{E(\boldsymbol{Y})}\right). \tag{7}$$

Based on this we can state the actual problem of image segmentation as MAP inference in this distribution. The problem we are trying to solve is given by

$$\boldsymbol{Y}^* = \arg\max P(\boldsymbol{Y}|\boldsymbol{X}). \tag{8}$$

Equivalently we can also minimize the energy $E(\boldsymbol{Y})$

$$\boldsymbol{Y}^* = \arg\min E(\boldsymbol{Y}). \tag{9}$$

Task 1. Is exact MAP inference for this task feasible? What is the complexity of this problem? See, for example, Chapter 13 of [4].

These are all the assumptions that we make for the distribution. Before we introduce an approximate inference algorithm, we will introduce one possible instantiation of such a model which allows easier interpretation and which we will use for the implementation.

2.1 Potts Model and Edge Potentials

To fully define the Gibbs distribution (7) we require the number of kernels K, definition of the features $\boldsymbol{f}_i$, the covariance matrices $\boldsymbol{\Sigma}_k$ and the compatibility function μ. Clearly all of these have to work together such that the Gibbs distribution qualitatively models our intuition of how labels should behave. That is, we want to find a compatibility function and parameters, such that,

- if the color of close pixels is similar, they likely belong to the same class,
- if different labels are close to each other, the pixel colors are likely to be different, and
- segments should not be very small and isolated (i.e., no single pixels with separate class).

First we need to define the features vector $\boldsymbol{f}$. It should be fast to compute, while being able to capture the information required to fulfill these intuitions. Here we choose a 5-dimensional feature vector, comprised of the pixel position $\boldsymbol{p}_i = \begin{bmatrix} x_i & y_i \end{bmatrix}^T$ and the color at the pixel $\boldsymbol{c}_i = \begin{bmatrix} r_i & g_i & b_i \end{bmatrix}^T$, that is,

$$\boldsymbol{f}_i = \begin{bmatrix} x_i \\ y_i \\ r_i \\ g_i \\ b_i \end{bmatrix} = \begin{bmatrix} \boldsymbol{p}_i \\ \boldsymbol{c}_i \end{bmatrix}. \tag{10}$$

Note, that computing this feature vector amounts to a simple lookup in the image. We will use two kernels to fulfill the three intuitions. The first is the *appearance kernel*, defined by a diagonal covariance matrix $\Sigma_1 = \mathrm{diag}(\theta_\alpha, \theta_\alpha, \theta_\beta, \theta_\beta, \theta_\beta)$, that is,

$$\begin{aligned} g_1(\boldsymbol{f}_i, \boldsymbol{f}_j) &= \exp\left(-\frac{1}{2}\left(\frac{(x_i - x_j)^2 + (y_i - y_j)^2}{\theta_\alpha^2} + \frac{(r_i - r_j)^2 + (g_i - g_j)^2 + (b_i - b_j)^2}{\theta_\gamma^2}\right)\right) \\ &= \exp\left(-\frac{1}{2}\left(\frac{|\boldsymbol{p}_i - \boldsymbol{p}_j|^2}{\theta_\alpha^2} + \frac{|\boldsymbol{c}_i - \boldsymbol{c}_j|^2}{\theta_\gamma^2}\right)\right). \end{aligned} \tag{11}$$

The second is the *smoothness kernel*, defined by a diagonal covariance matrix $\Sigma_2 = \mathrm{diag}(\theta_\gamma, \theta_\gamma, 0, 0, 0)$, that is,

$$g_2(\boldsymbol{f}_i, \boldsymbol{f}_j) = \exp\left(-\frac{|\boldsymbol{p}_i - \boldsymbol{p}_j|^2}{2\theta_\gamma^2}\right). \tag{12}$$

The appearance kernel gives high values for close and similar pixels, while the smoothness kernel gives high values for close pixels. Finally we need to define the label compatibility. Here we use a very simple label compatibility function, called *Potts model*. It is given by

$$\mu(Y_i, Y_j) = \begin{cases} 1 & \text{if } Y_i \neq Y_j \\ 0 & \text{if } Y_i = Y_j \end{cases}. \tag{13}$$

Task 2. Take a moment to understand these decisions and why they realize the three intuitions stated before. Which kernel belongs to which intuition? What is the qualitative effect of increasing and decreasing each of the parameters w_1, w_2, θ_α, θ_β, θ_γ when keeping the others fixed?

3 Mean Field Approximation

To make the MAP inference feasible, we have to fall back to approximate inference methods. Here, we are going to use the mean field approximation, which is a special case of structured variational inference. A thorough introduction into the topic can be found in Chapter 11.5 of [4]. In structured variational inference, we choose a class of simple distributions $\mathcal{Q}$ and try to find the distribution $Q \in \mathcal{Q}$ that minimizes the Kullback-Leibler divergence $D(Q||P)$ between the simple distribution Q and the complicated distribution P. As shown in [4], minimizing the Kullback-Leibler divergence is equivalent to maximizing the energy functional

$$F[P, Q] = \sum_{\phi \in \Phi} \mathbb{E}_Q \left[\ln \phi\right] + \mathbb{H}_Q(\boldsymbol{Y}) = \sum_{\phi \in \Phi} \mathbb{E}_Q \left[\ln \phi\right] + \mathbb{E}_Q \left[\ln Q(\boldsymbol{Y})\right]. \tag{14}$$

For mean field approximation we constrain Q to be a product of marginals over single variables, that is,

$$Q(\boldsymbol{Y}) = \prod_{i=1}^{N} Q_i(Y_i). \tag{15}$$

To maximize the energy functional, normally Lagrange multipliers are used.

Task 3. Why is MAP inference in distributions as defined by equation (15) trivial?

We next need to find an algorithm to maximize the energy functional. The general derivation can be found in [4] and uses the method of Lagrange multipliers. Here we will start with the simplified, general result from Corollary 11.6 from [4]. It states that $Q(Y_i)$ is locally optimal only if

$$Q(Y_i) = \frac{1}{Z_i} \exp\left(\sum_{\phi : Y_i \in \text{Scope}[\phi]} \mathbb{E}_{U_\phi = \{Scope[\phi] \setminus Y_i\} \sim Q} \Big[\ln \phi(U_\phi, Y_i) \Big] \right). \tag{16}$$

Task 4. Show that the update equation can be reformulated as

$$Q_i(Y_i = l) = \frac{1}{Z_i} \exp\left(-\phi_{u,i}(Y_i) - \sum_{l'=1}^{L} \mu(l, l') \sum_{k=1}^{K} w_k \sum_{j \neq i} g_k(\boldsymbol{f}_i, \boldsymbol{f}_j) Q_j(l') \right). \tag{17}$$

Based on this update equation we can now write down an update algorithm shown in Algorithm 1, which updates all marginals in parallel (i.e., each line performs the operation for all $1 \leq i \leq N$ and $Y_i \in \mathcal{L}$). Note, that the theory does not require this, but as we will see later, this allows efficient implementation and gives good results.

Algorithm 1 Mean field approximation

$Q_i(Y_i) \leftarrow \frac{1}{Z_i} \exp\left(-\phi_{\mathrm{u},i}(Y_i = l)\right)$ ▷ Initialize
while not converged **do**
  $\tilde{Q}_i^k(Y_i = l) \leftarrow \sum_{j \neq i} g_k(\boldsymbol{f}_i, \boldsymbol{f}_j) Q_j(Y_j = l)$ ▷ Message passing
  $\hat{Q}_i(Y_i = l) \leftarrow \sum_{l' \in \mathcal{L}} \mu(l, l') \sum_{k=1}^{K} w_k \tilde{Q}_i^k(l')$ ▷ Compatibility transform
  $Q_i(Y_i) \leftarrow \exp\left(-\phi_{\mathrm{u},i} - \hat{Q}_i(Y_i)\right)$ ▷ Local update
  $Q_i(Y_i) \leftarrow \frac{1}{Z_i} Q_i(Y_i)$ ▷ Normalize
end while

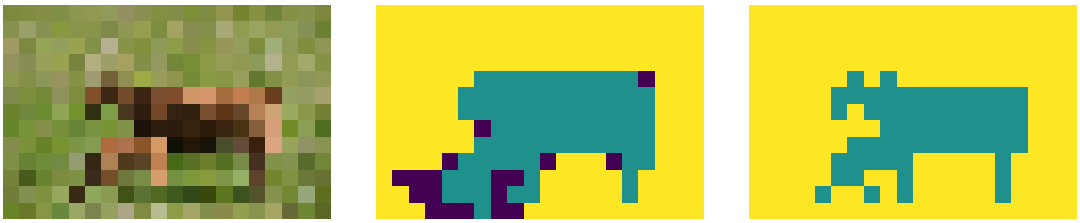


Figure 2: Result on heavily downsampled images. The downsampling is necessary to make the naive implementation fast enough. It can be seen that the segmentation is adjusted to better match the contours of the image.

Task 5. What is the computational complexity of each of the lines of Algorithm 1 in terms of number of pixels N, number of labels L and number of kernels K? Given that $N \gg L, K$, which is the most problematic line?

Task 6. Implement the missing parts in the file `crf/crf.py` for the functions

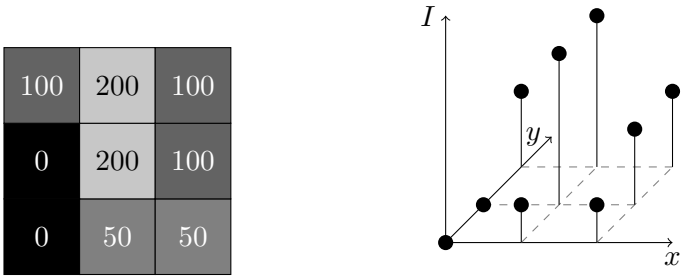
- `normalize`,
- `message_passing`,
- `compatibility_transform`, and
- `local_update`.

Each of these corresponds to the respective line in Algorithm 1 and should be straight forward to implement. Try to understand the remaining code by going through the README and reading the documentation of the functions. You can also have a look at the main inference loop (function `inference`) to understand how the other functions are used.

Task 7. Try out the algorithm. Most likely your implementation will be too slow for full resolution images. You can use the script `downsample.py` to produce images and segmentation of low resolution to see the result of your implementation. The output that you get should be similar to Figure 2.

3.1 Efficient Inference using Signal Processing Interpretation

Clearly the algorithm so far is not practicable for images of normal resolution. The problem is the quadratic complexity in the number of pixels of the message passing step. The key to speeding up the algorithm is to reformulate the message passing as a convolution.



(a) Gray-scale image (b) 3D representation of the image

Figure 3: Visualization of the sparse, high-dimensional interpretation of an image.

Let us start with an easier task, using only the smoothness kernel. In that case we can write the message passing as

$$\tilde{Q}_i(Y_i = l)^2 = \sum_{j \neq i} \exp\left(-\frac{|\boldsymbol{p}_i - \boldsymbol{p}_j|^2}{2\theta_\gamma^2}\right) Q_j(Y_j = l). \tag{18}$$

Task 8. Show that equation (18) can be reformulated as a 2D convolution. *Hint: Start by rewriting the sum as a sum over all i and then note how this sum corresponds to a convolution with a Gaussian kernel when interpreting Q as a 2D image with probabilities at each pixel.*

The problem with the appearance kernel is that the kernel now depends on the color of both pixels, making it impossible to formulate it as a 2D convolution. Instead we can formulate it as a 5D convolution, which is done by lifting the 2D image into a 5D space, by introducing three new axis r, g and b. An image in such a space will have one point for each x and y at r, g, b of the corresponding pixel. Thus we get a sparse 5D interpretation of the image. Figure **??** visualizes this idea for a gray scale image and 3D space.

Here, we do the same but with Q. Let $Q^5(\boldsymbol{Y})$ denote the 5D interpretation of $Q(\boldsymbol{Y})$, that is, for each pixel i $Q^5_{\boldsymbol{f}_i}(Y_i) = Q_i(Y_i)$. Now, we can compute a 5D Gaussian kernel and perform message passing as a 5D convolution. To get back to the desired value of $Q_i(Y_i)$, we just need to lookup the values at the corresponding $\boldsymbol{f}_i = \begin{bmatrix} x_i & y_i & r_i & g_i & b_i \end{bmatrix}^T$ position in the 5D space, that is,

$$\begin{aligned} \tilde{Q}^1_i(Y_i = l) &= \sum_{j=1}^{N} g_1(\boldsymbol{f}_i, \boldsymbol{f}_j)\tilde{Q}_j(Y_j = l) - \tilde{Q}_i(Y_i = l) \\ &= \left[G_1 * \tilde{Q}^5(Y_i)\right](\boldsymbol{f}_i) - \tilde{Q}_i(Y_i = l). \end{aligned} \tag{19}$$

Finally a naive implementation of this operation will still be very slow, that is, a naive 5D convolution will cause a complexity of $\mathcal{O}(N256^3)^2LK)$ (assuming 0-255 for each colors). Hence, much worse than the previous complexity of $\mathcal{O}(N^2LK)$. We can perform a number of simplifications though. First, Gaussian filters are separable, which means we can apply one-dimensional filters across each dimension, which reduces the complexity to $\mathcal{O}(N256^3ALK)$ with A being the size of the Gaussian kernel. Note, that without doing any additional approximations A will be as big as the largest dimension, hence we still have a complexity of $\mathcal{O}(N^2)$ with an additional overhead due to the additional dimension.

But we can do another simplification, based on the observation that a Gaussian filter is a low-pass filter, which will cut off high frequency components. If we downsample the 5D representation, for example using a box filter[1], we do not lose much accuracy, since we mostly

[1]We use a box filter as it prevents a good trade-off between speed and preventing alias effects.

Figure 4: Results achieved using the convolution based implementation. The segmentation in the middle is used to generate the unary potentials, which are then used to produce the final segmentation on the right.

remove high frequency information, that will be removed by the Gaussian anyway. Hence, we can perform convolution in a much smaller space. Furthermore, we can cut off the Gaussian filter outside of one or two standard deviations, since the values become negligible. Going back to the complexity, this will cause $A \ll N$ and hence we reach $\mathcal{O}(N)$, which is the complexity of applying downsampling and converting Q to its low resolution, 5-dimensional representation Q^5. Finally, note that the we need to adjust the parameters of the Gaussian filter to account for the change in size[2].

Task 9. A skeleton that performs efficient message passing is already given in file `crf/crf.py`. Go through the code and implement the missing parts. To prevent parameter tuning the given code already provides parameters, which were used to achieve the results in Figure 4[a].

[a] These parameters are very similar to the ones stated by Krähenbühl [1].

4 Summary

We showed that by starting from an intractable graphical model we can perform theoretically sound approximations which allowed us to reach an inference algorithm that yields good results in seconds. Few parameters have to be set for this method and good results can be obtained without costly parameter fitting. Clearly one could also try to find optimal parameters using accurate training and validation data, we skip this learning step as it is not at the core of this approach. Krähenbühl [1] furthermore demonstrated that by approximation of the convolution in a permutohedral lattice [6] and implementation in C++, inference times of well under a

[2] The general idea of using a downsampled 5D representation of an image stems from bilateral filters, which turn out to be very similar to the message passing step in fully connected CRFs [5].

second can be achieved. While we used unary potentials given from annotation, the approach is straightforward to use as a post-processing step for any kind of semantic segmentation technique (i.e., deep neural networks as in [7]). Furthermore we only used the simple Potts model, while a more sophisticated compatibility function, that models how likely certain classes are to neighbor was successfully used by Krähenbühl [1].

References

[1] Philipp Krähenbühl and Vladlen Koltun. Efficient inference in fully connected crfs with gaussian edge potentials. In *Advances in Neural Information Processing Systems*, pages 109–117, 2011.

[2] Yuri 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.

[3] Pushmeet Kohli, Philip HS Torr, et al. Robust higher order potentials for enforcing label consistency. *International Journal of Computer Vision*, 82(3):302–324, 2009.

[4] Daphne Koller and Nir Friedman. *Probabilistic graphical models: principles and techniques*. 2009.

[5] Sylvain Paris and Frédo Durand. A fast approximation of the bilateral filter using a signal processing approach. In *European conference on computer vision*, pages 568–580. Springer, 2006.

[6] Andrew Adams, Jongmin Baek, and Myers Abraham Davis. Fast high-dimensional filtering using the permutohedral lattice. In *Computer Graphics Forum*, volume 29, pages 753–762. Wiley Online Library, 2010.

[7] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(4):834–848, 2017.