Efficient Linear Programming for Dense CRFs

Thalaiyasingam Ajanthan[1] Alban Desmaison[2] Rudy Bunel[2] Mathieu Salzmann[3] Philip H. S. Torr[2] M. Pawan Kumar[2,4]

[1]Australian National University & Data61, CSIRO [2]Department of Engineering Science, University of Oxford [3]CVLab, EPFL [4]Alan Turing Institute

Introduction

AIM: Efficient and accurate energy minimization in fully connected CRFs.

Dense CRF energy: Defined on a set of n random variables $\mathcal{X} = \{X_1, \dots, X_n\}$, where each random variable X_a takes a label $x_a \in \mathcal{L}$.

$$E(\mathbf{x}) = \sum_{a=1}^{n} \underbrace{\phi_a(x_a)}_{\text{unary potentials}} + \sum_{a=1}^{n} \sum_{b=1, b\neq a}^{n} \underbrace{\psi_{ab}(x_a, x_b)}_{\text{pairwise potentials}} .$$

Sparse CRF Dense CRF

- Captures long-range interactions and provides fine grained segmentations [5].

Difficulty: There are $O(n^2)$ number of variables ($n \approx 10^6$) $\Rightarrow$ Even the energy computation is Intractable.

Gaussian pairwise potentials:

$$\psi_{ab}(x_a, x_b) = \mathbb{1}[x_a \neq x_b] \underbrace{\exp\left(\frac{-\|\mathbf{f}_a - \mathbf{f}_b\|^2}{2}\right)}_{K_{ab}} \quad \text{where } \mathbf{f}_a \in \mathbb{R}^d .$$

$$\begin{bmatrix} v'_a \end{bmatrix}_{n\times 1} = \begin{bmatrix} K_{ab} \end{bmatrix}_{n\times n} \begin{bmatrix} v_b \end{bmatrix}_{n\times 1}$$

- Approximate the above computation using the filtering method [1] $\Rightarrow O(n)$ computations.

Existing efficient algorithms:

- Mean Field (MF) [5], Quadratic Programming (QP) [3] and Difference of Convex (DC) Programming [3].
- All these algorithms rely on the efficient filtering method and have linear time complexity per iteration.

Drawback: No multiplicative bound $\Rightarrow$ The solution obtained by these algorithms can be far away from the optimum.

Linear Programming (LP) relaxation provides the best multiplicative bound but the existing algorithm is too slow.

Linear Programming (LP) Relaxation

- Introduce indicator variables: $y_{a;i} = 1 \Rightarrow x_a = i$.

$$\min_{\mathbf{y}} \quad \tilde{E}(\mathbf{y}) = \sum_a \sum_i \phi_{a;i}\, y_{a;i} + \sum_{a,b\neq a} \sum_i K_{ab} \frac{|y_{a;i} - y_{b;i}|}{2} ,$$

$$\text{s.t.} \quad \mathbf{y} \in \mathcal{M} = \left\{ \mathbf{y} \;\middle|\; \begin{matrix} \sum_i y_{a;i} = 1, \; a \in \{1 \dots n\} \\ y_{a;i} \geq 0, \; a \in \{1 \dots n\}, \; i \in \mathcal{L} \end{matrix} \right\} .$$

- For integer labellings $\tilde{E}(\mathbf{y}) = E(\mathbf{x})$.

E.g., $\mathcal{L} = \{0, 1, 2, 3, 4\}$
$x_a = 2 \Rightarrow y_a = [0, 0, 1, 0, 0]$
$x_b = 3 \Rightarrow y_b = [0, 0, 0, 1, 0]$
$\mathbb{1}[x_a \neq x_b] = \sum_i \frac{|y_{a;i} - y_{b;i}|}{2}$

- LP provides an integrality gap of 2 [4], and no better relaxation can be designed due to the UGC hardness result [7].

Difficulty: Standard LP solvers would require $O(n^2)$ variables $\Rightarrow$ Intractable.

Require auxiliary variables: $|y_{a;i} - y_{b;i}| = \max\{y_{a;i} - y_{b;i}, y_{b;i} - y_{a;i}\} \;\Rightarrow\; z_{ab;i} \geq y_{a;i} - y_{b;i}$ and $z_{ab;i} \geq y_{b;i} - y_{a;i}$.

- Therefore the non-smooth LP objective has to be tackled directly.

Existing algorithm [3]: Projected subgradient descent $\Rightarrow$ Too slow.

- Linearithmic time per iteration (due to a divide-and-conquer strategy).
- Expensive line search to obtain the step size.
- Requires large number of iterations.

E.g., $n = 4$

$$\begin{bmatrix} v'_1 \\ v'_2 \\ v'_3 \\ v'_4 \end{bmatrix} = \left[\begin{array}{cccc} 0 & K_{12} & K_{13} & K_{14} \\ 0 & 0 & K_{23} & K_{24} \\ 0 & 0 & 0 & K_{34} \\ 0 & 0 & 0 & 0 \end{array}\right] \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

LP subgradient computation.

Contribution: LP in linear time per iteration $\Rightarrow$ An order of magnitude speedup ($n = 10^6 \Rightarrow$ 20 times speedup).

- The first LP minimization algorithm for dense CRFs that maintains linear scaling in both time and space complexity.

Our Approach: Proximal Minimization of LP

Algorithm overview:
Proximal minimization using block-coordinate descent.
- **One block:** Significantly smaller subproblems.
- **The other block:** Efficient *conditional gradient descent*.
  - Linear time complexity per iteration.
  - Optimal step size.
- Gurantees optimality and converges faster.

Proximal problem: Let $\lambda > 0$ and $\mathbf{y}^k$ be the current estimate

$$\min_{\mathbf{y}} \quad \tilde{E}(\mathbf{y}) + \frac{1}{2\lambda}\left\|\mathbf{y} - \mathbf{y}^k\right\|^2 ,$$
$$\text{s.t.} \quad \mathbf{y} \in \mathcal{M} .$$

- The quadratic regularization makes the dual problem smooth
  $\Rightarrow$ Efficient optimization of the dual.

Dual of the Proximal Problem

Dual variables:

- $\boldsymbol{\alpha} = \left\{\alpha^1_{ab:i}, \alpha^2_{ab:i} \mid a, b \neq a, i \in \mathcal{L}\right\}$, $\quad \alpha^1_{ab:i}: \; z_{ab:i} \geq y_{a:i} - y_{b:i}$, $\quad \alpha^2_{ab:i}: \; z_{ab:i} \geq y_{b:i} - y_{a:i}$.
- $\boldsymbol{\beta} = \{\beta_a \mid a \in \{1 \dots n\}\}$, $\quad \beta_a: \; \sum_i y_{a:i} = 1$.
- $\boldsymbol{\gamma} = \{\gamma_{a:i} \mid a \in \{1 \dots n\}, i \in \mathcal{L}\}$, $\quad \gamma_{a:i}: \; y_{a:i} \geq 0$.
- The matrices A and B are used to compactly write the dual function.
- $\boldsymbol{\alpha}$ always appear in the product $A\boldsymbol{\alpha}$ $\Rightarrow$ Space complexity is linear.

$$\min_{\boldsymbol{\alpha},\boldsymbol{\beta},\boldsymbol{\gamma}} g(\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\gamma}) = \frac{\lambda}{2}\|A\boldsymbol{\alpha} + B\boldsymbol{\beta} + \boldsymbol{\gamma} - \boldsymbol{\phi}\|^2 + \left\langle A\boldsymbol{\alpha} + B\boldsymbol{\beta} + \boldsymbol{\gamma} - \boldsymbol{\phi}, \mathbf{y}^k \right\rangle - \langle \mathbf{1}, \boldsymbol{\beta} \rangle ,$$

$$\text{s.t.} \quad \gamma_{a:i} \geq 0 \quad \forall\, a \in \{1 \dots n\} \quad \forall\, i \in \mathcal{L} ,$$

$$\boldsymbol{\alpha} \in C = \left\{ \boldsymbol{\alpha} \;\middle|\; \begin{matrix} \alpha^1_{ab:i} + \alpha^2_{ab:i} = \frac{K_{ab}}{2}, \; a, b \neq a, \; i \in \mathcal{L} \\ \alpha^1_{ab:i}, \alpha^2_{ab:i} \geq 0, \; a, b \neq a, \; i \in \mathcal{L} \end{matrix} \right\} .$$

Optimizing over β and γ

- $\boldsymbol{\beta}$: Unconstrained $\Rightarrow$ Set derivative to zero.
- $\boldsymbol{\gamma}$: Unbounded and separable $\Rightarrow$ A m dimensional QP for each pixel, where m is the number of labels.
- Efficiently optimized using the multiplicative iteration algorithm of [8] ($\approx$ 2 milliseconds).

For each pixel a, the QP has the form
$$\min_{\boldsymbol{\gamma}_a \geq \mathbf{0}} \; \tfrac{1}{2}\boldsymbol{\gamma}_a^T Q \boldsymbol{\gamma}_a + \left\langle \boldsymbol{\gamma}_a, Q\left((A\boldsymbol{\alpha}^t)_a - \boldsymbol{\phi}_a\right) + \mathbf{y}^k_a \right\rangle ,$$
where $Q \in \mathbb{R}^{m\times m}$.

Optimizing over α

Conditional gradient descent:

$$\min_{\boldsymbol{\alpha} \in C} g(\boldsymbol{\alpha}) ,$$

g is differentiable and C is convex and compact.

Conditional gradient computation:
- Minimize the first order Taylor approximation.

Step size computation:
- $\delta = 2/(t+2)$ or by line search.

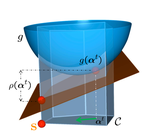


Image from [6].

In our case:

- The conditional gradient has the same form as the subgradient of LP [2].
- The optimal step size can be computed analytically.
- These benefits are due to our choice of quadratic regularization.

$$\forall\, a \in \{1 \dots n\}, \quad v'_a = \sum_b K_{ab}\, v_b \; \mathbb{1}[y_a \geq y_b] ,$$

where $y_a, y_b \in [0, 1]$.

E.g., $n = 3$

$$\begin{bmatrix} v'_1 \\ v'_2 \\ v'_3 \end{bmatrix} = \left[\begin{array}{ccc} K_{11} & K_{12} & K_{13} \\ 0 & K_{22} & K_{23} \\ 0 & 0 & K_{33} \end{array}\right] \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix}$$

Linear time conditional gradient computation:

Difficulty: The original permutohedral lattice based filtering method of [1] cannot handle the ordering constraint.

Existing method [3]: Repeated application of the filtering method using a divide-and-conquer strategy $\Rightarrow O(d^2 n \log(n))$ computations ($d = 2$ or 5).

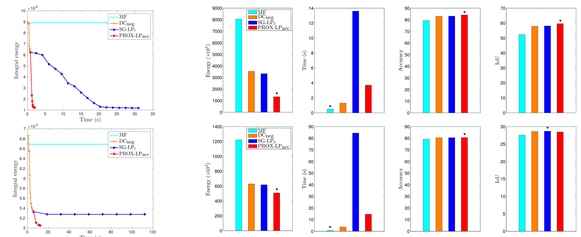
Our idea: Discretize the interval $[0, 1]$ to H levels and instantiate H permutohedral lattices $\Rightarrow O(Hdn)$ computations ($H = 10$).

Modified filtering method:
- Creates single permutohedral lattice and reuses it H times $\Rightarrow$ Cache efficient.
- Substantial speedup (> 40) compared to the state-of-the-art method of [3].

- Our modified filtering method has wide applicability beyond what is shown in this work.

Segmentation Results

Datasets: Pascal VOC 2010 and MSRC.



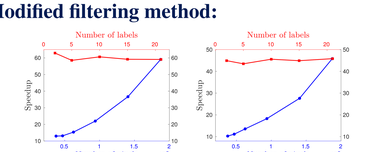
Energy vs time plots for an image in MSRC and Pascal.

*Energy, time, segmentation accuracy and IoU score of different algorithms on (**top**) MSRC and Pascal (**bottom**) datasets.*

- Both LP minimization algorithms are initialized with DC_{neg}.

Image MF DC_{neg} SG-LP_f $\text{PROX-LP}_{\text{acc}}$ Ground truth

Modified filtering method:



Speedup of our modified filtering method over that of [3].

Speedup is around 45 – 65 on a Pascal image.

Discussion

- We have introduced an efficient and accurate algorithm for energy minimization in dense CRFs.
- Our algorithm can be incorporated in an end-to-end learning framework to further improve the accuracy of deep semantic segmentaion architectures.

Code: `https://github.com/oval-group/DenseCRF`

[1] Adams *et al.*, *Computer Graphics Forum*, 2010.
[2] Bach F, *SIAM*, 2015.
[3] Desmaison *et al.*, *ECCV*, 2016.
[4] Kleinberg and Tardos, *ACM*, 2002.
[5] Krähenbühl and Koltun, *NIPS*, 2011.
[6] Lacoste-Julien *et al.*, *ICML*, 2012.
[7] Manokaran *et al.*, *STOC*, 2008.
[8] Xiao *et al.*, *Numerical Linear Algebra*, 2014.