Max-norm optimization and strict optimizers

Filip Malmberg

Introduction

Many of the optimization problems we have seen so far have objective functions that can be described as consisting of two parts:

- A set of *local* error measurements (e.g. unary or pairwise terms)
- A way to aggregate the local error measurements into a single scalar value (e.g., sum, maximum, . . .)

The aggregation function determines how the error is “distributed” across the domain (e.g., the image).

Recall: P-norms

- The sum and the maximum are special cases of *p-norms*.
- Let $p \geq 1$ be a real number. The *p-norm* of a vector x is defined as

$$\|\mathrm{x}\|_p = \left(\sum_{i=1}^{n} |x_i|^p \right)^{1/p} \tag{1}$$

- For $p = 1$, this is the sum of the elements of the vector. For $p \to \infty$, it approaches the maximum of the elements.

Recall: P-norms

- For low p, the L_p norm emphasizes the decrease in average error, but allows arbitrarily high local error.
- On the opposite end of the spectrum, L_∞-norm ensures the tightest possible control on the worst-case error. below the maximal error, however, it does not distinguish between solutions with just one or all elements with high error.

Strict minimizers

The concept of *strict minimizers* was proposed by Levi et al. [2].

- In this framework, two solutions are compared by ordering all elements non-increasingly by their local error value and then performing their lexicographical comparison.
- A solution is optimal (a *strict optimizer*) if it compares as better than or equal to all other solutions.

Strict minimizers and L_p norms

The definition of strict minimizers in the previous slide does not use an explicit objective functions, but it can be shown that is tightly connected to the limit of L_p norms as p goes to infinity.

- A strict minimizer minimizes the L_∞-norm (max-norm) of the error. (But the opposite does not generally hold)
- Strict minimizers are the *limits* of L_p norm minimizers, as $p \to \infty$.

Letting p go to ∞ – a toy example

Example: Fix the values of the pixels marked in red to 0 and 1, respectively. Assign all other values so that the gradients (local errors) are minimized, for some given aggolmeration function.

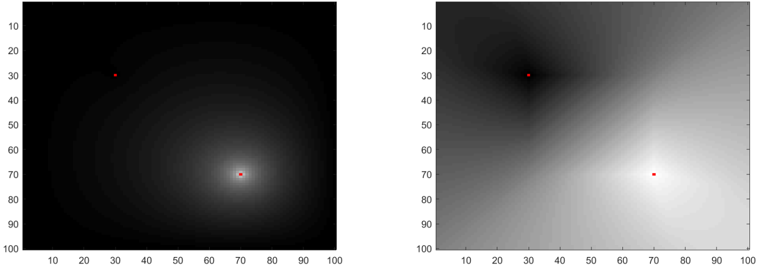


Figure 1: Left: L_2.norm, Right: L_∞-norm/Strict minimizer.

Uniqueness of strict minimizers

- Consider a combinatorial optimization problem where all local errors have a finite number of possible configurations/values.
- If all local errors have distinct (unique) values, then the strict minimizer is unique. Otherwise, it is not. (Why?)

Rest of this lecture

We will now have a look at two papers that concern finding strict minimizers for different optimization problems in image analysis.

- "The Mutex Watershed and its Objective: Efficient, Parameter-Free Graph Partitioning", Wolf et al 2021. [7]
- "Two Polynomial Time Graph Labeling Algorithms Optimizing Max-Norm-Based Objective Functions", Malmberg and Ciesielski, 2020 [3]

A common theme for these papers is that they show that certain optimization problems that are NP-hard under commonly used norms become solvable with the strict optimization approach.

Paper 1: The mutex watershed

Consider, again, graph partitioning by MSF-cuts.

- We can think of the edge weights as *attractive* forces. "How high is the preference for two adjacent pixels to be grouped together".
- In Kruskal's algorithm, we keep grouping pixels in order of decreasing edge weights (attractive force).
- Regions are only prevented from merging by the seedpoints.
- An "oracle" is required to decide good seed points (algorithm of human)

Extending Kruskal's algorithm with repulsive forces

- The main idea of Wolf et al. is to avoid having to define seedpoints by adding *repulsice forces* to the process.
- This leads to an algorithm where the number of clusters does not need to known beforehand!

The algorithm

The algorithm operates on a graph $G = (V, E)$ where each edge has a signed, real valued weight. Positive edge weights are attractive, negative edge weights are attractive.

- Sort all edges in descending order by *absolute* value.
- For every edge:
  - If the edge is attractive, and there is no *mutex* between the regions spanned by the edge, then merge those regions.
  - If the edge is repulsive, and the edge spans two distinct regions, then add a *mutex* between these regions

Implementation

- Just like Kruskal's algorithm, an efficient implementation of the mutex watershed can be achieved using a disjoint-set data structure.
- A hash table (or other efficient set datastructure) can be used to store information about mutex constraints for each region.
- Theoretically, the worst cas run-time is $\mathcal{O}(|E|^2)$, but empirically the runtime is very close to the $\mathcal{O}(|E| \log |E|)$ runtime of Kruskal's algorithm.
- Publicly available implementation: `https://github.com/hci-unihd/mutex-watershed`

Experiments

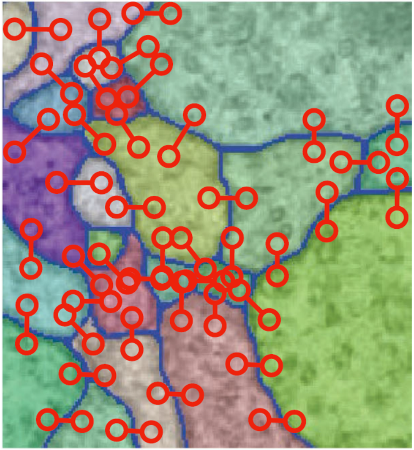


Figure 2: Mutex watershed segmentation of image from an ISBI neuron segmentation challenge

Experiments

- The mutex watershed algorithm performed very well in the highly competitive ISBI EM challenge on neuron segmentation.
- Some noteworthy details:
    - Use of "long-range" connections. "The strength of such edges can often be estimated with greater certainty than is achievable for the local edges".
    - Using a CNN to estimate edge weights

Optimality of Mutex watersheds

- The mutex watershed is closely related to a graph partitioning problem called the *multicut*-problem, which is known to be NP-hard
- It is shown by Wolf et al. [7] that if all edge weights are unique, then the mutex watershed solves a variant of the multicut problem in which all edge weight are raised to some sufficiently large power p. (A "dominant power")
- In our terminology: The mutex watershed solves the strict minimization version of the otherwise NP-hard multicut problem! (When the edge weights are unique)

Optimality of regular watersheds/MSF cuts

- We have established earlier that MSF-cuts are optimal according to the max-norm.
- A corollary of the result by Wolf et al. is that regular MSF-cuts do in fact also cuts that are strict optimizers, when the edge weights are distinct! (MSF-cuts/regular watersheds are a special case of the Mutex Watershed, see [6] for an explanation of how mutex constraints can be translated to seed-point constraints)

What if the edge weights are not distinct?

- Requiring that the edge weights are distinct seems restrictive! What if this condition is violated?
- Even if the edge weights are not unique, it is straightforward to define new unique edge weights:
  - Establish any increasing order of the edge weights (e.g., using a sorting algorithm)
  - Replace the weight of each edge with the value corresponding to its position in this ordering (1,2,3,. . .).
- This is in fact what happens during the sorting step of the mutex watershed algorithm!
- Even if the edge weights are unique, the algorithm will return a result that is strictly optimal according to *some* ordering of the edge weights!

Semi-strict minimizers

- We say that a solution is a *semi-strict* optimizer if there exists some increasing/decreasing ordering of the local errors such that the solution is a strict optimizer w.r.t. this ordering.
- If the local errors are distinct, the (unique) semi-strict optimizer is also the strict optimizer.
- The mutex watershed returns a semi-strict optimizer even if the edge-weights are not unique.

Paper 2: Two Polynomial Time Graph Labeling Algorithms Optimizing Max-Norm-Based Objective Functions

Here, we consider the same "canonical" pixel labeling problem that we studied in the minimal graph cut lecture. We seek a label assignment configuration x that minimizes a given objective function E, written as follows:

$$E(\mathrm{x}) = \sum_{i \in \mathcal{V}} \phi_i(x_i) + \sum_{i,j \in \mathcal{E}} \phi_{ij}(x_i, x_j) \ , \tag{2}$$

where:

- $\mathcal{V}$ is the set of pixels in the image.
- $\mathcal{E}$ is the set of all adjacent pairs of pixels in the image.
- x_i denotes the label of vertex i, belonging to a finite set of integers $\{0, 1 \ldots, K-1\}$.

Recall: Optimization by minimal graph cuts

- In the general case, global optimization of this labeling problem is NP-hard, but in special cases globally optimal solutions can be found efficiently.
- For the binary labeling problem, with $K = 2$, a globally optimal solution can be computed by solving a max-flow/min-cut problem on a suitably constructed graph. This requires all pairwise terms to be *submodular* ($\approx$ convex).
- A pairwise term ϕ_{ij} is said to be submodular if

$$\phi_{ij}(0,0) + \phi_{ij}(1,1) \leq \phi_{ij}(0,1) + \phi_{ij}(1,0)\ . \tag{3}$$

Multi-label problems

- At first glance, the restriction to binary labeling may appear very limiting.
- The multi-label problem can, however, be reduced to a sequence of binary valued labeling problems using, e.g., the *expansion move* algorithm (Boykov et al. 2001, Kolmogorov et al. 2004)
- Thus, the ability to find optimal solutions for problems with two labels has high relevance also for the multi-label case.
- These approaches have been very succesful, and have made graph cuts a standard tool for solving general optimization problem in image processing.

Generalized objective functions

Looking again at the labeling problem described above, we can view the objective function E as consisting of two parts:

- A *local* error measure, in our case defined by the unary and pairwise terms.
- A *global* error measure, aggregating the local errors into a final score. In the case of E, the global error measure is obtained by summing all the local error measures.

$$E(\mathrm{x}) = \sum_{i \in \mathcal{V}} \phi_i(x_i) + \sum_{i,j \in \mathcal{E}} \phi_{ij}(x_i, x_j) \tag{4}$$

L_p-norm objective functions

If we assume all terms to be non-negative, minimizing E can be seen as minimizing the l_1-norm of the vector containing all unary and pairwise terms. A natural generalization is to consider minimization of arbitrary l_p-norms, $p \geq 1$, i.e., minimizing:

$$E_p(\mathrm{x}) = \left(\sum_{i \in \mathcal{V}} \phi_i^p(x_i) + \sum_{i,j \in \mathcal{E}} \phi_{ij}^p(x_i, x_j) \right)^{1/p} \tag{5}$$

Minimizing L_p-norm objective functions via grah cuts

- It is straightfoward to show that similar submodularity requirements hold also for the generalized objective functions E_p for any finite p.

$$(\phi_{st}^p(0,0) + \phi_{st}^p(1,1))^{1/p} \leq (\phi_{st}^p(0,1) + \phi_{st}^p(1,0))^{1/p}. \tag{6}$$

- We say that a pairwise term that satisfies this condition is *p-submodular*.
- Binary L_p norm labeling problems of the form (5) can be globally optimized using graph cuts, if all pairwise terms are p-submodular.
- To use the graph cut approach, we must first verify that all pairwise terms satisfy the appropriate submodularity conditions. Otherwise, we have to resort to approximate methods.

The case when $p \to \infty$

In the limit case when $p \to \infty$, the objective function converges to:

$$E_\infty(\ell) := \max\Big\{\max_{s \in V} \phi_s(\ell(s)), \max_{s,t \in} \phi_{st}(\ell(s), \ell(t))\Big\}. \tag{7}$$

Similarly, the p-submodularity condition converges to:

$$\max\{\phi_{st}(0,0), \phi_{st}(1,1)\} \leq \max\{\phi_{st}(1,0), \phi_{st}(0,1)\}, \tag{8}$$

We say that a pairwise term that satisfies this inequality is ∞-submodular.

A helpful theorem

For optimization of L_p-norm labeling problems with graph cuts, the following theorem can be helpful for proving p-submodularity:

- If a binary term is n-submodular (for some $n \geq 1$) and ∞-submodular, then it is also p-submodular for any real $p \geq n$. [5]

Main result

- We have shown that in the limit as p goes to infinity, *the requirement for submodularity of the pairwise terms disappears*!
- Thus, even when the local costs are such that the problem of minimizing E_p is NP-hard for some or all finite p, a labeling minimizing E_∞ can be found in low order polynomial time! (In practice: linearithmic)

Direct optimization of max-norm problems

- In two recent papers [3, 4], we present two different algorithms for optimizing binary labeling problems with the max-norm E_∞ objective function:
  - A linearithmic time algorithm for optimizing E_∞ under the condition that all pairwise terms are ∞-*submodular*.
  - An algorithm for optimizing *any* function E_∞, submodular or not. The theoretical runtime for this algorithm is quadratic, but empirically it is also linearithmic.
- A pairwise term is said to be ∞-*submodular* if:

$$\max\{\phi_{ij}(0,0), \phi_{ij}(1,1)\} \leq \max\{\phi_{ij}(1,0), \phi_{ij}(0,1)\}. \tag{9}$$

Outline of our proposed algorithms

- To describe the optimization methods, we introduce the notion of unary and binary solution *atoms*.
- A *unary* atom represents one possible label configuration for a single vertex.
- A *binary* atom represent a possible label configuration for a pair of adjacent vertices.
- Thus, for a binary labeling problem, there are two unary atoms associated with every pixel and four binary atoms for every pair of adjacent pixels.
- Each atom has a *weight* given by the corresponding unary or binary term of the objective function.

Outline of our proposed algorithm

The algorithm works as follows:

- Start with a set S consisting of all possible atoms.
- For each atom A, in order of decreasing weight:
  - If $S \setminus \{A\}$ is consistent, remove A from S.

A set of atoms is said to be *consistent* if it is possible to construct at least one valid labeling from the atoms in the set.

At the termination of this algorithm, the atoms remaining in S define a unique labeling. This labeling is globally optimal according to the objective function E_∞.

Checking consistency

The key issue is to determine, at each step of the algorithm, whether the remaining set of atoms is consistent.

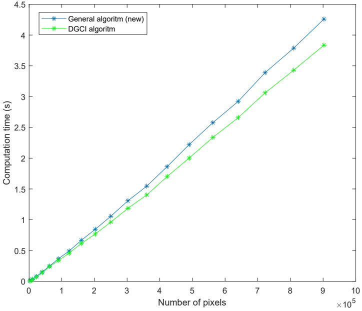
- When the all pairwise terms are ∞-submodular, we show that this check can be performed efficiently via "local" conditions. This leads to the pseudo-linear algorithm.
- In the general case, we show that the problem of determining the consistency can be phrased as a *boolean 2-satisfiability problem*, solvable in linear time. This leads to the quadratic algorithm.

The 2-SAT problem

- Consider a set of boolean variables (*true* or *false*) and a set of constraints on these variables, such that each constraint involves at most two variables. The *2-SAT problem* consists of answering the question: *Is there an assignment of truth values (i.e.,*0 *or* 1*) to the variables that satisfies all given constraints?*
- Solvable in linear time using e.g., Aspvall's algorithm [1].

An efficient version of the general algorithm

- Running Aspvall's algorithm for every atom we want to remove is inefficient.
- Each satisfiability problem, however, is very similar to the previous one. We have found a (yet unpublished) way to utilize this redundancy to formulate a practically efficient algorithm!



Strict minimization

- As shown in [3], the output of the labeling algorithm described above is not only L_∞ optimal, but is in fact a strict minimizer if the local costs have unique weights, i.e., just like the Mutex Watershed is returns semi-strict minimizers.
- Note that the algorithm is structurally very similar to both Kruskal's algorithm and the Mutex watershed algorithm!

Multi-label optimization

- The algorithm for solving binary labeling problems relies on the fact that the 2-SAT problem is solvable in polynomial (linear) time.
- The n-SAT problem for $n > 2$ is unfortunately NP-hard, and it follows that strict optimization of multilabel problems is also NP-hard.
- (But just as with graph cuts we can still make use of the 2-label case to do move-making/local search)
- (Does this contradict the fact that the Mutex Watershed can solve multi-region segmenation? No!)

Examples: Inpainting

Inpainting by minimizing L_∞-norm of partial derivatives (finite difference approximation) across unknown region.

Left: 4 -neighbors. Right: 8-neighbors with weights

Examples: Image matting (soft segmentation)

Image matting by solving the (L_∞) Poisson equation across the gray region.

Left to right: Image, Right: Trimap

Examples: Image matting (soft segmentation)

Poisson matting result. (Recreation of an example from the paper "Poisson Matting", Sun et al., SIGGRAPH 2004, but under the L_∞ norm instead of the L_2 norm.)

Conclusions

- Strict optimization is an alternative framework for defining optimization problems, but with close connections to L_p norm optimization in the limit case where p goes to infinity.
- Many important optimization problems that are NP-hard under other p-norms can be solved very efficiently under the max-norm/as strict optimizers!
- Lots of open questions left to be explored!

References

[1] Bengt Aspvall, Michael F Plass, and Robert Endre Tarjan.
A linear-time algorithm for testing the truth of certain quantified boolean formulas.
Information processing letters, 8(3):121–123, 1979.

[2] Zohar Levi and Denis Zorin.
Strict minimizers for geometric optimization.
ACM Transactions on Graphics (TOG), 33(6):1–14, 2014.

[3] Filip Malmberg and Krzysztof Chris Ciesielski.
Two polynomial time graph labeling algorithms optimizing max-norm-based objective functions.
Journal of Mathematical Imaging and Vision, 62(5):737–750, 2020.

[4] Filip Malmberg, Krzysztof Chris Ciesielski, and Robin Strand.
Optimization of max-norm objective functions in image processing and computer vision.
In *International Conference on Discrete Geometry for Computer Imagery*, pages 206–218. Springer, 2019.

[5] Filip Malmberg and Robin Strand.
When can $$l_p$$-norm objective functions be minimized via graph cuts?
In Reneta P. Barneva, Valentin E. Brimkov, and João Manuel R.S. Tavares, editors, *Combinatorial Image Analysis*, pages 112–117, Cham, 2018. Springer International Publishing.

[6] Filip Malmberg, Robin Strand, and Ingela Nyström.
Generalized hard constraints for graph segmentation.
In *Scandinavian Conference on Image Analysis*, pages 36–47. Springer, 2011.

[7] Steffen Wolf, Alberto Bailoni, Constantin Pape, Nasim Rahaman, Anna Kreshuk, Ullrich Köthe, and Fred A Hamprecht.
The mutex watershed and its objective: Efficient, parameter-free graph partitioning.
IEEE transactions on pattern analysis and machine intelligence, 2020.