Minimizing Convex Piecewise-affine Functions by Local Consistency Techniques

Author: Tomáš Dlask | Supervisor: Tomáš Werner | FEE, Czech Technical University in Prague

Problem

We consider the problem of minimizing a convex piecewise-affine function, given as the sum of pointwise maxima of affine functions, i.e.

$$f(\mathbf{x}) = \sum_{i=1}^{l} \max_{j=1}^{m_i} (\mathbf{a}_{i,j}^T \mathbf{x} + b_{i,j}), \tag{1}$$

where $\mathbf{a}_{1,1}, ..., \mathbf{a}_{l,m_l}, \mathbf{x} \in \mathbb{R}^n$.

This problem can be formulated as a linear program, however, for very large sparse instances, on which we focus, solving this LP might be in practice impossible because of the space and time complexity of general LP solvers. Alternatively, minimizing such function can be seen as an instance of convex non-differentiable minimization and thus one could apply subgradient methods which have linear space complexity, but these methods have been experimentally observed to be very slow. Additionally, we are not looking for its global minimum, but for local minima (not local in topological meaning) that we denote as **locally consistent points**.

Contribution

We designed a **novel algorithm** that searches for the locally consistent points. The algorithm is iterative and in each iteration

1. decides whether it is in a locally consistent point (if it is, it terminates),
2. finds a decreasing direction from this point,
3. performs line search to find a step size,
4. shifts current point in the direction by the step size.

Additionally, we have also introduced local ϵ - consistency generalization that further relaxes the notion of locally consistent points and provides a significant speed-up that is based on the idea of **capacity scaling** (i.e. gradually tightening the conditions on the result until satisfactory solution is reached). This relaxation improves the results and also allows us to **prove the correctness** of the algorithm – it always reaches a locally ϵ-consistent point after a finite amount of iterations under the assumption of precise arithmetic.

We have also introduced an integer version of the algorithm, whose correctness was justified using only the assumption of **finite-precision arithmetic**. We implemented it in C++ and made our experiments with it.

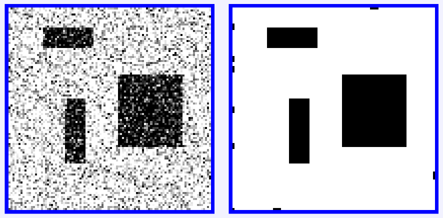
Finally, we formulated **theorems** that in some cases allow us to determine the optimality of the found solution.

Experiments and Results

We tested the algorithm on instances that correspond to **Schlesinger's upper bound of a binary max-sum problem**, which was also the original motivation for solving this problem. The upper bound is a function that can be expressed in form (1) and the mentioned max-sum problem was based on two-dimensional grammars and searching the closest image generated by a given grammar.

For example, we could be given an image that is in the left part of the figure below – it is an image with noise and we would like to find the closest black and white image to it that is generated by *rectangles* grammar. Such images contain only black rectangles that do not touch or overlap on white background. The optimal solution (and also the output of our algorithm) is on the right side of the figure below. Similar experiments were done with 4 different grammars and various amounts of noise on 24 images in total.

The minimized function in the form (1) in the largest instance consisted of 8736000 subfunctions (i.e. different vectors $\mathbf{a}_{i,j}$) that were of dimension 6986000 (which is also the amount of real scalar variables in the problem).



The relaxation used by the algorithm showed to be suitable for such problems because it **often reached the true optimum**. Moreover, our algorithm had significantly shorter runtime on all the instances when compared to an optimal LP solver (i.e. approximately **10–10^4 times faster**, depending on the instance). Note that comparing our solver with a general LP solver is reasonable, since any LP can be in linear time transformed into minimizing a function in form (1). However, for LPs that are not sparse, the locally consistent points would not be good local minima.

On the generated instances, we also tried re-optimizing the solution after a small change in the input instance was done, which could become useful in ill-conditioned or ambiguous problems.

Future Work

Continuation of this topic is going to be a part of the PhD work of the author. We have encountered a number of open questions that might be resolved in the future – for example whether the algorithm would work also with subfunctions that are not only affine, its usage in branch-and-bound methods, and generalizing further the local consistency notion.