Exploiting Structure for Tractable Nonconvex Optimization

Abram L. Friesen AFRIESEN@CS.WASHINGTON.EDU
Pedro Domingos PEDROD@CS.WASHINGTON.EDU
Department of Computer Science and Engineering, University of Washington, Seattle, WA 98195 USA

Abstract

MAP inference in continuous probabilistic models has largely been restricted to convex density functions in order to guarantee tractability of the underlying model, since high-dimensional nonconvex optimization problems contain a combinatorial number of local minima, making them extremely challenging for convex optimization techniques. This choice has resulted in significant computational advantages but a loss in model expressivity. We present a novel approach to nonconvex optimization that overcomes this tradeoff by exploiting local structure in the objective function, greatly expanding the class of tractable, continuous probabilistic models. Our algorithm optimizes a subset of variables such that near the minimum the remaining variables decompose into approximately independent subsets, and recurses on these. Finding the global minimum in this way is exponentially faster than using convex optimization with restarts.

1. Introduction

MAP inference in continuous probabilistic models has typically been restricted to (or approximated by) convex functions in order to guarantee that inference remains tractable. Nonconvex functions are generally intractable, with much of the difficulty arising from a combinatorial explosion of modes in the objective function. For example, protein folding (Anfinsen, 1973; Baker, 2000) corresponds to finding the MAP assignment of a continuous pairwise Markov random field, e.g., (Yanover et al., 2006), where the protein conformations follow a Boltzmann distribution. The goal is to find the minimum-energy configuration of a chain of amino acids, and although the amino acids only interact in pairs, the combination of these interactions results in an extremely intricate energy landscape. State-of-the-art solvers for this problem use convex optimization with randomization and other refinements, but are insufficient for most proteins. In effect, attempting to solve protein folding by gradient descent is akin to attempting to solve a jigsaw puzzle by sliding each piece straight to its correct position, ignoring that it bumps into other pieces along the way.

In reality, the way we solve puzzles is by decomposition: find approximately independent subproblems, solve each separately, and combine the results; if the combination fails, try a new decomposition. Problem decomposition has a long history in combinatorial optimization, starting with the DPLL satisfiability solver (Davis et al., 1962), which finds an assignment of values to Boolean variables that makes the objective formula true. More recently, developments like recursive conditioning (Darwiche, 2001), MPE-SAT (Sang et al., 2007), and sum-product networks (Gens & Domingos, 2013) showed that similar ideas can be used to find modes of probability distributions. In this paper, we extend these ideas to nonconvex optimization, thereby greatly extending the class of models for which inference is tractable. The main feature we require of the objective function is that it be a sum of terms, each of which depends on only a subset of the variables, similar to satisfiability formulas and probabilistic graphical models. Objective functions of this form are very common (e.g., regression, graphical models, energy functions), and our method should therefore be widely applicable.

The key step in our algorithm is to dynamically identify a subset of variables that, once optimized, decomposes the remaining variables into approximately independent subsets. These can then be separately optimized, and we do so recursively, finding within each a subset that further decomposes it, etc. The separating variables are then re-optimized given the solutions to the subproblems, and the process repeats until convergence. Solving subproblems separately leads to an exponential reduction in run time. The local optimization subroutine can be any method; in our experiments, we use conjugate gradient with restarts.

Our approach has similarities with block coordinate descent (Tseng & Yun, 2009) and alternating minimization methods (O'Sullivan, 1998), but with the key difference that our decomposition is recursive and dynamic, changing from one iteration to the next as dictated by the properties

Proceedings of the 31^{st} International Conference on Machine Learning, Beijing, China, 2014. JMLR: W&CP volume 32.

of the objective function in the current subspace. One interpretation of our algorithm is that we generate and solve an approximation to the objective function, where this approximation can be compiled into a sum-product network. However, where sum-product networks learn functions that are guaranteed to be efficiently optimizable, our algorithm approximates arbitrary functions for optimization.

2. Local Structure

In unconstrained optimization the goal is to minimize an objective function, $f(x)$, over the values of the variables, $x \in \mathbb{R}^n$. We focus on functions, $f : \mathbb{R}^n \to \mathbb{R}$, that are continuously differentiable, have a nonempty optimal set x^* with optimal value f^*, and are *partially separable* (Griewank & Toint, 1981; Nocedal & Wright, 2006), such that $f(x) = \sum_i t_i(x)$, where each *term* t_i depends only on a subset of x. A function is *fully separable* if it can be expressed as $f(x) = \sum_i t_i(x_i)$. Such functions are significantly easier to optimize, since they *decompose* with respect to minimization; i.e., $\min_x f(x) = \sum_i \min_{x_i} t_i(x_i)$.

Similarly, partially separable functions can exhibit *partial decomposition*. For example, rewriting $\min_{x,y,z} g(x,y,z) = \min_{x,y,z} t_0(x,z) + t_1(y,z)$ as $\min_z \{\min_x t_0(x,z) + \min_y t_1(y,z)\}$, reveals that the minimization within the braces has decomposed. Thus, if we were to assign z and perform the minimization *conditioned* on $z = a$, the resulting function, $g(x,y|z=a) = \min_x t_0(x,a) + \min_y t_1(y,a)$, is fully separable and decomposes into two independent minimizations. Thus, partial decomposition refers to a decomposition that occurs as a result of conditioning on *any* value of another variable. However, this is a strict assumption, as even simple functions such as $t_0(x,y) + t_1(y,z) + t_2(z,x)$ are not partially decomposable.

Extending these concepts, we introduce the idea of *local decomposition*, which occurs as a result of conditioning on *specific* values of another variable. A trivial example of this would occur if we were optimizing $\min_{x,y,z} h(x,y,z) = \min_{x,y,z} t_0(x) + t_1(x,y,z) + t_2(y)$, where $t_1(x,y,z) = xyz$. Here, if we assign $z = 0$, then $t_1 = 0$, for any values of x and y. As a result, $h(x,y,0)$ decomposes as $\min_x t_0(x) + 0 + \min_y t_2(y)$. This decomposition is local because it depends on the specific value that z is assigned and would (potentially) not occur for other values of z. Note that local decomposition subsumes partial decomposition so, in the remainder of the paper, references to local decomposition also imply partial decomposition. In addition, locally independent subspaces are those that can be optimized independently due to local decomposition.

Finally, since the above definition has limited applicability, we introduce *approximate local decomposition*. In the previous example, t_1 had to become identically 0 to allow decomposition. We relax this by introducing an error tolerance $\epsilon \geq 0$ and considering the upper and lower bounds on $t_1(x,y,z)$, such that $t_1(x,y,z) \in \left[\underline{t_1}(x,y,z), \overline{t_1}(x,y,z)\right]$. Similarly, we can consider the bound on the conditioned term $t_1(x,y|z) \in \left[\underline{t_1}(x,y|z), \overline{t_1}(x,y|z)\right]$. Dropping the variables for clarity, if we know that $t_1 \in \left[\underline{t_1}, \overline{t_1}\right]$ and that $\overline{t_1} - \underline{t_1} \leq 2\epsilon$, then we can replace t_1 with a constant $k = \frac{1}{2}\left(\overline{t_1} + \underline{t_1}\right)$ and guarantee that $|h^* - h_{\min}| \leq \epsilon$, where $h_{\min}$ is the optimum of h with $t_1 = k$ and h^* is the true optimum of h. Extending this to the case where m terms are assigned, the maximum possible error is simply $m\epsilon$. We refer to the process of assigning terms to constants when their bounds are sufficiently narrow as *simplification*.

3. An Algorithm for Nonconvex Optimization

In this section, we develop our optimization algorithm (RDIS), which globally minimizes an objective function by (R)ecursively (D)ecomposing it into locally (I)ndependent (S)ubspaces. Pseudocode is presented in Algorithm 1.

The main idea underlying RDIS is that decomposed functions require exponentially less exploration to find the global optimum than non-decomposed functions. Thus, the goal of RDIS is to find and exploit decomposition in the objective function, in order to significantly reduce the amount of search necessary to globally optimize a nonconvex function. While most interesting problems are not fully separable, many are locally separable. Referring to the function $g(x,y,z)$ from Section 2, we see that conditioning on z caused the remaining subproblem to decompose. Following this same mechanism, RDIS dynamically chooses variables (or blocks of variables) to condition on, in order to achieve separability and decomposition in the remaining sub-function. It exploits full, partial, and (approximate) local separability to realize this decomposition.

Concretely, if we are optimizing $f(x) = \sum_i t_i(x)$, then RDIS chooses a block of variables $x_c \in x$, where the remaining variables are contained in x_u such that $x = \{x_c, x_u\}$. This choice induces a partition on the terms such that $t = \{t_c, t_b, t_u\}$, where t_c are those terms containing variables *only* in x_c, while t_u is the set of terms containing variables only in x_u and t_b contains the remaining terms, each of which contains variables in both x_c and x_u. In general, t_b is nonempty, otherwise the problem would be fully separable. Now, RDIS chooses and assigns a set of values, $x_c = v_c$. Conditioning on x_c causes x_u to decompose into k subsets of variables $x_u^1, \ldots, x_u^k$. In the worst case $k = 1$ and no decomposition occurs, while in the best case $k = |x_u|$ and x_u fully decomposes. In general, $1 \leq k \leq |x_u|$, resulting in the above decomposition into k subsets. Similarly, the resulting minimization also decomposes: $\min_{x_u} f(x_u|x_c = v_c) = \sum_{j=1}^{k} \min_{x_u^j} f_{u_j}(x_u^j|x_c = v_c)$, where $f_{u_j}(x_u^j|x_c = v_c) =$

Algorithm 1 Approximate minimization of an objective function by recursive decomposition into locally independent subspaces.

input The function $f(x) = \sum_i t_i(x)$ over variables x, contained in terms $t = \{t_i\}$, each of which is over a subset of the variables.
input Approximation error, ϵ.

1: **function** $MinRDIS(f, x, t, \epsilon)$
2: choose a block of variables x_c from x
3: $t_c \leftarrow$ terms only containing x_c
4: $x_u \leftarrow x \backslash x_c$ and $t_{ub} \leftarrow t \backslash t_c$
5: **while** domain of x_c not sufficiently explored **do**
6: choose values v_c for x_c
7: $f_{x_c=v_c} \leftarrow \text{simplify}(t_{ub}, \epsilon \mid x_c = v_c)$
8: $\left\{f_{u_k}(x_u^k)\right\} \leftarrow \text{decompose}(f_{x_c=v_c}, x_u, t_{ub})$
9: $f_{\min}(v_c, x_u) = \sum_k MinRDIS(f_{u_k}, x_u^k, t_{ub}, \epsilon)$
10: $f_{\min}(x) \leftarrow \min\left(f_{\min}(x), f_{\min}(v_c, x_u)\right)$
11: **end while**
12: **return** $f_{\min}(x)$
13: **end function**

$\sum t_{u_j}(x_u^j) + \sum t_b(x_u^j | x_c = v_c)$. RDIS tracks this conditional decomposition as well as any local decompositions that occur. It then performs independent optimizations on each of the k separate sub-functions, and sums their solutions to get the solution of $\min_{x_u} f(x_u | x_c = v_c)$.

The separability structure that RDIS exploits exists at many levels of detail within the objective function. To capture this structure, RDIS optimizes each of the independent sub-functions $f_{u_1}(x_u^1 | x_c = v_c), \ldots, f_{u_k}(x_u^k | x_c = v_c)$ by recursing on it as if it were the entire objective function. When $|x_u^j|$ drops below a user-provided size, RDIS conditions entirely on x_u^j, choosing values in the same manner as for any conditioning set, x_c, and the recursion halts.

After minimizing each sub function f_{u_j}, RDIS uses the optimal values found, x_u^*, to choose new values for x_c, by optimizing $f(x_c | x_u = x_u^*)$. Given new values $x_c = v_c'$, RDIS optimizes $f(x_u | x_c = v_c')$, a new sub-function that results in novel decompositions and simplifications. This is because RDIS exploits (approximate) local decomposition and the separability structure at each level of recursion changes as the conditioning variables x_c and their values v_c change. Thus, the entire structure of the optimization changes over time as blocks of variables and their values are chosen based on the local information in different parts of space. This allows RDIS to flexibly adapt to the local structure in the objective function.

A wide array of possible methods exist for choosing blocks of variables (e.g., MOMS (Freeman, 1995)). In this paper, we focus on a heuristic that achieves decomposition whenever possible: hypergraph partitioning. Choosing the smallest block of variables that, when conditioned on, decomposes the remaining variables provides maximum decomposition. RDIS constructs a hypergraph with a vertex for each term and a hyperedge for each variable, where each hyperedge connects to all vertices for which the corresponding term contains that vertex's variable. The cutset defined by the best partition is the smallest set of variables that need to be removed in order to decompose the hypergraph and is exactly the set of variables that RDIS chooses on line 2. This variable selection heuristic is local because the variables and terms at each level of the recursion vary depending on prior decompositions and simplifications.

For value selection (or, equivalently, subspace optimization), we use a very simple method: conjugate gradient descent with random restarts. Other optimization methods are also possible (e.g., Monte Carlo search, trust-region methods, simulated annealing) and we intend to investigate their effect in future work. However, our successes with this basic technique indicate that our underlying mechanism, recursive decomposition using local structure, is effective.

As discussed in Section 2, simplification of the objective function is the process of assigning terms to constant values when their bounds become sufficiently narrow. RDIS maintains the bounds of each term, $t_i \in \left[\underline{t_i}, \overline{t_i}\right]$, as the state of the variables in t_i change (we assume that the terms have bounded range within the region of interest). At each level of recursion, we examine the bounds of each term and simplify those with $\overline{t_i} - \underline{t_i} \leq 2\epsilon$. If there are m terms, then the maximum possible error that simplification can introduce into the objective function is $m\epsilon$, when all m terms are simplified. Techniques such as interval arithmetic (Hansen & Walster, 2003) can be used to compute bounds on arbitrary terms; however, better bounds and thus more simplification and decomposition can typically be achieved by considering the specific problem being optimized.

4. Analysis

If $f(x) = \sum_i t_i(x)$ is continuously differentiable and has a nonempty optimal set x^* with optimal value f^* then we can state the following theorems.

Theorem 1. *Algorithm 1 (RDIS) returns a stationary point of f when $\epsilon = 0$.*

Corollary 1. *Algorithm 1 returns the global optimum f^* when $\epsilon = 0$ and f is convex.*

The intuition behind the proof of Theorem 1 is that, at each level of recursion, RDIS partitions the variables into the two sets x_c and x_u, chooses initial values for x_c, and then performs a sequence of iterative updates to the values of these sets. When $\epsilon = 0$, this process satisfies the conditions for convergence of an inexact Gauss-Seidel method (Bonettini, 2011).

For $\epsilon > 0$, if there are m terms and the algorithm converges, then the error due to simplification is at most $m\epsilon$. However, we do not have an explicit proof that RDIS converges, even in the convex case. Our preliminary analysis indicates that there may be rare corner cases in which the alternating minimization aspect of RDIS combined with the simplification of terms can potentially result in a nonconverging sequence of values. Nonetheless, we have not experienced any issues with convergence during our experimental evaluations of RDIS. Furthermore, our experiments show $\epsilon > 0$ to be extremely beneficial in practice, especially for large and highly-connected problems.

At each level of recursion, RDIS uses hypergraph partitioning to choose the smallest conditioning set of variables that results in a decomposition of the remaining variables. Let $n_c = |x_c|$ be the size of the conditioning set and $n_u^1, \ldots, n_u^k = n_u$ be the sizes of the independent sets resulting from the decomposition of x_u, then the dimensionality of the space that RDIS explores is $k \cdot n_u \cdot n_c$. In decomposition-free algorithms, however, the dimensionality of the state space is $(n_u)^k \cdot n_c$, which is exponentially larger. This examination highlights one of the major advantages that RDIS has over other algorithms that do not exploit local and partial decomposition.

5. Experimental Evaluation

We have preliminary results for Algorithm 1 on two difficult nonconvex optimization problems: a highly multimodal test function we constructed and continuous sidechain placement (Yanover et al., 2006) in protein folding (Anfinsen, 1973; Baker, 2000). We have compared RDIS to conjugate gradient descent (CGD) and block-coordinate descent (BCD), both with random restarts.

The test function is defined as follows. Given a height h, a branching factor k, and a maximum arity A, we define a complete k-ary tree of variables of the specified height, with x_0 as the root. For all paths $p_j \in P$ of length $l_j \leq A$, with l_j even, we define a term $t_{p_j} = \prod_{x_i \in p_j} \sin(x_i)$. We now define the test function $f_{h,k,A}(x_0, \ldots, x_n) = \sum_{i=1}^{n}(c_0 x_i + c_1 x_i^2) + c_2 \sum_P t_{p_j}$. It is a multidimensional sinusoid placed in the basin of a quadratic function parameterized by c_1, with slope c_0.

RDIS finds significantly better minima than both CGD and BCD on functions of arity 8 and 12 while reaching an equivalent minimum faster for arity 4 (a larger arity results in more complex dependencies between variables and more terms in the function). An ablated version of RDIS where blocks of variables are instead chosen randomly performs on par with CGD, indicating that the performance of RDIS degrades gracefully when decomposition does not occur.

Protein folding is the process by which a protein, consisting of a long chain of amino acids, assumes its functional shape. The computational problem is to predict this final conformation given a known sequence of amino acids and can be modeled as MAP inference in a graphical model. This requires minimizing an energy function consisting mainly of a sum of pairwise distance-based terms representing chemical bonds, hydrophobic interactions, electrostatic forces, etc., where, in the simplest case, the variables are the relative angles between the atoms. Each amino acid is composed of a backbone segment and a sidechain and the sidechain placement task is to predict the conformation of the sidechains when the backbone atoms are fixed in place.

Our results indicate that RDIS is able to take advantage of the large amounts of local structure present in this task in order to return consistently better minimas than either conjugate gradient descent or block coordinate descent. In all but the smallest protein, RDIS significantly outperforms the other algorithms and never performs worse than either of the other algorithms. Furthermore, the amount by which RDIS improves on CGD and BCD increases with the size of the proteins, demonstrating the significant reduction in the size of the search space that RDIS explores.

6. Conclusion

This paper proposes a new approach to solving hard nonconvex optimization problems, greatly expanding the class of tractable continuous probabilistic models. RDIS recursively decomposes the state space into approximately locally independent subspaces, enabling them to be optimized separately and exponentially reducing the time required to find the global optimum (or a good local one). In our experiments, RDIS systematically outperforms conjugate gradient and block coordinate descent.

Directions for future research include defining and learning models that exhibit the structure exploited by RDIS, applying RDIS to a wide variety of nonconvex optimization and inference problems, further analyzing its theoretical properties, developing new variable and value selection methods, extending RDIS to handle constrained optimization, and using similar ideas for high-dimensional integration.

Acknowledgments

This research was partly funded by ARO grant W911NF-08-1-0242, ONR grants N00014-13-1-0720 and N00014-12-1-0312, and AFRL contract FA8750-13-2-0019. The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of ARO, ONR, AFRL, or the United States Government.

References

Anfinsen, Christian B. Principles that govern the folding of protein chains. *Science*, 181(4096):223–230, 1973.

Baker, David. A surprising simplicity to protein folding. *Nature*, 405:39–42, 2000.

Bonettini, Silvia. Inexact block coordinate descent methods with application to non-negative matrix factorization. *IMA journal of numerical analysis*, 31(4):1431–1452, 2011.

Darwiche, Adnan. Recursive conditioning. *Artificial Intelligence*, 126(1-2):5–41, 2001.

Davis, Martin, Logemann, George, and Loveland, Donald. A machine program for theorem-proving. *Communications of the ACM*, 5(7):394–397, 1962.

Freeman, Jon William. *Improvements to propositional satisfiability search algorithms*. PhD thesis, University of Pennsylvania, 1995.

Gens, Robert and Domingos, Pedro. Learning the Structure of Sum-Product Networks. In *Proceedings of the Thirtieth International Conference on Machine Learning*. Omnipress, 2013.

Griewank, Andreas and Toint, Ph L. On the unconstrained optimization of partially separable functions. *Nonlinear Optimization*, 1982:247–265, 1981.

Hansen, Eldon and Walster, G William. *Global optimization using interval analysis: revised and expanded*, volume 264. CRC Press, 2003.

Nocedal, Jorge and Wright, Stephen J. *Numerical Optimization*. Springer, 2006.

O'Sullivan, Joseph A. Alternating minimization algorithms: From blahut-arimoto to expectation-maximization. In Vardy, Alexander (ed.), *Codes, Curves, and Signals*, volume 485 of *The Springer International Series in Engineering and Computer Science*, pp. 173–192. Springer US, 1998.

Sang, Tian, Beame, Paul, and Kautz, Henry A. A dynamic approach for MPE and weighted MAX-SAT. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 173–179, 2007.

Tseng, Paul and Yun, Sangwoon. A coordinate gradient descent method for nonsmooth separable minimization. *Mathematical Programming*, 117(1-2):387–423, 2009.

Yanover, Chen, Meltzer, Talya, and Weiss, Yair. Linear programming relaxations and belief propagation – an empirical study. *The Journal of Machine Learning Research*, 7:1887–1907, 2006.