INTERNATIONAL JOURNAL OF
NUMERICAL ANALYSIS AND MODELING
Volume 17, Number 4, Pages 557–591

TRANSIENT DYNAMICS OF BLOCK COORDINATE DESCENT IN A VALLEY

MARTIN MOHLENKAMP, TODD R. YOUNG*, AND BALÁZS BÁRÁNY

Abstract. We investigate the transient dynamics of Block Coordinate Descent algorithms in valleys of the optimization landscape. Iterates converge linearly to a vicinity of the valley floor and then progress in a zig-zag fashion along the direction of the valley floor. When the valley sides are symmetric, the contraction factor to a vicinity of the valley floor appears to be no worse than 1/8, but without symmetry the contraction factor can approach 1. Progress along the direction of the valley floor is proportional to the gradient on the valley floor and inversely proportional to the "narrowness" of the valley. We quantify narrowness using the eigenvalues of the Hessian on the valley floor and give explicit formulas for certain cases. Progress also depends on the direction of the valley with respect to the blocks of coordinates. When the valley sides are symmetric, we give an explicit formula for this dependence and use it to show that in higher dimensions nearly all directions give progress similar to the worst case direction. Finally, we observe that when starting the algorithm, the ordering of blocks in the first few steps can be important, but show that a greedy strategy with respect to objective function improvement can be a bad choice.

Key words. Block Coordinate Descent, alternating Least Squares, tensor Approximation, swamp, diagonal Valley.

1. Introduction

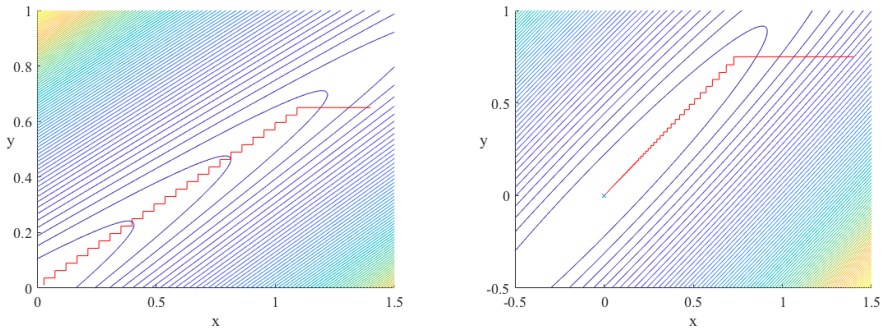
Consider the generic problem of trying to find minimum points of a non-negative, differentiable objective function $f : \mathbb{R}^n \to \mathbb{R}_+$. The argument of f can be considered as a column vector $\mathbf{x}$, which can then be broken into blocks as $\mathbf{x} = (\mathbf{x}_1; \mathbf{x}_2; \ldots; \mathbf{x}_d)$. Minimization with respect to a single block $\mathbf{x}_i$ while holding the other blocks fixed is often much easier than minimizing with respect to the full $\mathbf{x}$. This suggests a minimization algorithm: starting from an initial $\mathbf{x}$,

loop: until some convergence criteria is met:
  loop: through $i = 1, \ldots, d$:
    update: $\mathbf{x}_i$ to minimize f with respect to $\mathbf{x}_i$.

We call the minimization with respect to one block of coordinates a **micro-step** and one loop through i a **pass**. This algorithm is the simplest form of block coordinate descent (BCD) (see e.g. [52, 23, 48, 2, 41, 53] and many textbooks). When the blocks consist of single coordinates, BCD is called alternating coordinate [22] or coordinate descent (CD) (e.g. [23, 30]).

Despite (or perhaps due to) their simplicity, BCD methods are widely used and promising for many applications such as high-dimensional data analysis [30], machine learning [36, 24], image processing [53] and others [15, 49]. In the context of low rank tensor approximation problems, BCD is known as alternating least-squares (ALS) (e.g. [38, 32, 3, 4, 50, 27, 29, 19, 20, 7, 16, 6, 18, 45, 51, 40, 17, 46]). A micro-step of ALS reduces to a linear least squares problem and so is extremely

Received by the editors September 25, 2018 and, in revised form, April 22, 2020.
2000 *Mathematics Subject Classification.* 37N30, 65K10.
Corresponding author.



(A) (B)

FIGURE 1. Illustrations of the behavior of a coordinate descent (CD) method in valleys in $\mathbb{R}^2$. The first panel shows iterations along a straight diagonal valley far from any local minimum and the valley floor is at an angle $\pi/6$ from the x-axis. In the second panel iterations are shown near a local minimum point that is in a diagonal valley that is $\pi/4$ from the coordinate directions. Contour curves are shown in both panels. In both cases the trajectories zig-zag slowly downhill along the valley floor.

efficient and precise. Overall, ALS is observed to converge rapidly in many cases. In other cases, however, it exhibits long periods of very slow progress, in what is informally called a **swamp**. Swamps can be classified as **terminal** or **transient**. In a terminal swamp, the slow progress continues to a (local) minimum point. In a transient swamp, progress eventually accelerates and the iterates exit the swamp. Analysis of the causes of swamps can be broken into two questions:

(1) What features of a generic f can cause BCD algorithms to progress slowly?
(2) What aspects of the tensor approximation problem cause such features to occur so frequently and strongly?

Narrow valleys (thin ridges in maximization problems) have been recognized as a challenge for optimization algorithms for many years (see [34, 35, 42, 11, 14]) and one of the classical test functions for optimization, the Rosenbrock function [42], has a minimum in a narrow valley that is diagonally oriented at the minimum. Thus, narrow valleys are a natural candidate as a cause for swamps and answer for Question 1. In [13] we developed a rudimentary quantitative theory for measuring the effect of narrow valleys on algorithms, based on the gradient descent with line search (GDLS) algorithm. For BCD methods, the orientation of the valley is important. In Figure 1 we illustrate iterations of a CD method in transient and terminal valleys in $\mathbb{R}^2$. Roughly speaking, the problem is that iterations of a BCD method zig-zag slowly in a narrow valley. It is also clear that a valley will attract a non-trivial open set of points under a BCD method.

For the tensor approximation problem, in [13] we found that nonhyperbolic sinks and saddles can occur for certain parameters values and these cause swamps. For nearby parameter values the weakly hyperbolic sinks and saddles create narrow valleys that also cause swamps. See Figure 2 for an illustration of such narrow valleys.