Feature Clustering for Accelerating Parallel Coordinate Descent

Chad Scherrer
Independent Consultant
Yakima, WA
chad.scherrer@gmail.com

Ambuj Tewari
Department of Statistics
University of Michigan
Ann Arbor, MI
tewaria@umich.edu

Mahantesh Halappanavar
Pacific Northwest National Laboratory
Richland, WA
mahantesh.halappanavar@pnnl.gov

David J Haglin
Pacific Northwest National Laboratory
Richland, WA
david.haglin@pnnl.gov

Abstract

Large-scale ℓ_1-regularized loss minimization problems arise in high-dimensional applications such as compressed sensing and high-dimensional supervised learning, including classification and regression problems. High-performance algorithms and implementations are critical to efficiently solving these problems. Building upon previous work on coordinate descent algorithms for ℓ_1-regularized problems, we introduce a novel family of algorithms called block-greedy coordinate descent that includes, as special cases, several existing algorithms such as SCD, Greedy CD, Shotgun, and Thread-Greedy. We give a unified convergence analysis for the family of block-greedy algorithms. The analysis suggests that block-greedy coordinate descent can better exploit parallelism if features are clustered so that the maximum inner product between features in different blocks is small. Our theoretical convergence analysis is supported with experimental results using data from diverse real-world applications. We hope that algorithmic approaches and convergence analysis we provide will not only advance the field, but will also encourage researchers to systematically explore the design space of algorithms for solving large-scale ℓ_1-regularization problems.

1 Introduction

Consider the ℓ_1-regularized loss minimization problem

$$\min_{\mathbf{w}} \ \frac{1}{n}\sum_{i=1}^{n} \ell(\mathbf{y}_i, (\mathbf{X}\mathbf{w})_i) + \lambda\|\mathbf{w}\|_1 \ , \tag{1}$$

where $\mathbf{X} \in I\!R^{n\times p}$ is the design matrix, $\mathbf{w} \in I\!R^p$ is a weight vector to be estimated, and the loss function ℓ is such that $\ell(y,\cdot)$ is a convex differentiable function for each y. This formulation includes ℓ_1-regularized least squares (Lasso) (when $\ell(y,t) = \frac{1}{2}(y-t)^2$) and ℓ_1-regularized logistic regression (when $\ell(y,t) = \log(1+\exp(-yt))$). In recent years, coordinate descent (CD) algorithms have been shown to be efficient for this class of problems [Friedman et al., 2007; Wu and Lange, 2008; Shalev-Shwartz and Tewari, 2011; Bradley et al., 2011].

Motivated by the need to solve large scale ℓ_1 regularized problems, researchers have begun to explore parallel algorithms. For instance, Bradley et al. [2011] developed the Shotgun algorithm. More recently, Scherrer et al. [2012] have developed "GenCD", a generic framework for expressing

parallel coordinate descent algorithms. Special cases of GenCD include Greedy CD [Li and Osher, 2009; Dhillon et al., 2011], the Shotgun algorithm of [Bradley et al., 2011], and Thread-Greedy CD [Scherrer et al., 2012].

In fact, the connection between these three special cases of GenCD is much deeper, and more fundamental, than is obvious under the GenCD abstraction. As our first contribution, we describe a general randomized *block-greedy* that includes all three as special cases. The block-greedy algorithm has two parameter: B, the total number of feature blocks and P, the size of the *random subset* of the B blocks that is chosen at every time step. For each of these P blocks, we greedily choose, in parallel, a single feature weight to be updated.

Second, we present a non-asymptotic convergence rate analysis for the randomized block-greedy coordinate descent algorithms for general values of $B \in \{1, \ldots, p\}$ (as the number of blocks cannot exceed the number of features) and $P \in \{1, \ldots, B\}$. This result therefore applies to stochastic CD, greedy CD, Shotgun, and thread-greedy. Indeed, we build on the analysis and insights in all of these previous works. Our general convergence result, and in particular its instantiation to thread-greedy CD, is novel.

Third, based on the convergence rate analysis for block-greedy, we optimize a certain "block spectral radius" associated with the design matrix. This parameter is a direct generalization of a similar spectral parameter that appears in the analysis of Shotgun. We show that the block spectral radius can be upper bounded by the maximum inner product (or correlation if features are mean zero) between features in distinct blocks. This motivates the use of correlation-based feature clustering to accelerate the convergence of the thread-greedy algorithm.

Finally, we conduct an experimental study using a simple clustering heuristic. We observe dramatic acceleration due to clustering for smaller values of the regularization parameter, and show characteristics that must be paid particularly close attention for heavily regularized problems, and that can be improved upon in future work.

2 Block-Greedy Coordinate Descent

Scherrer et al. [2012] describe "GenCD", a generic framework for parallel coordinate descent algorithms, in which a parallel coordinate descent algorithm can be determined by specifying a *select* step and an *accept* step. At each iteration, features chosen by *select* are evaluated, and a proposed increment is generated for each corresponding feature weight. Using this, the *accept* step then determines which proposals are to be updated.

In these terms, we consider the *block-greedy* algorithm that takes as part of the input a partition of the features into B *blocks*. Given this, each iteration *select*s features corresponding to a set of P randomly selected blocks, and *accept*s a single feature from each block, based on an estimate of the resulting reduction in the objective function.

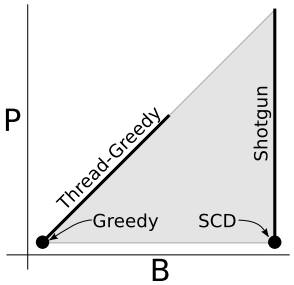


Figure 1: The design space of block-greedy coordinate descent.

The pseudocode for the randomized block-greedy coordinate descent is given by Algorithm 1. The algorithm can be applied to any function of the form $F + R$ where F is smooth and convex, and R is convex and separable across coordinates. Our objective function (1) satisfies these conditions. The greedy step chooses a feature within a block for which the *guaranteed descent* in the objective function (if that feature alone were updated) is maximized. This descent is quantified by $|\eta_j|$, which is defined precisely in the next section. To arrive at an heuristic understanding, it is best to think of $|\eta_j|$ as being proportional to the absolute value $|\nabla_j F(\mathbf{w})|$ of the jth entry in the gradient of the smooth part F. In fact, if R is zero (no regularization) then this heuristic is exact.

The two parameters, B and P, of the block-greedy CD algorithm have the ranges $B \in \{1, \ldots, p\}$ and $P \in \{1, \ldots, B\}$. Setting these to specific values gives many existing algorithms. For instance when $B = p$, each feature is a block on its own. Then, setting $P = 1$ amounts to randomly choosing a single coordinate and updating it which gives us the stochastic CD algorithm of Shalev-Shwartz and Tewari [2011]. Shotgun [Bradley et al., 2011] is obtained when B is still p but $P \geq 1$. Another

Algorithm 1 Block-Greedy Coordinate Descent

Parameters: B (no. of blocks) and $P \leq B$ (degree of parallelism)
while not converged **do**
  Select a random subset of size P from the B available blocks
  Set J to be the features in the selected blocks
  Propose increment $\eta_j, j \in J$ // parallel
  Accept $J' = \{j \ : \ \eta_j$ has maximal absolute value in its block$\}$
  Update weight $w_j \leftarrow w_j - \eta_j$ for all $j \in J'$ // parallel

extreme is the case when all the features constitute a single block. That is, $B = 1$. Then block-greedy CD is a deterministic algorithm and becomes the greedy CD algorithm of Li and Osher [2009]; Dhillon et al. [2011]. Finally, we can choose non-trivial values of B that lie strictly between 1 and p. When this is the case, and we choose to update all blocks in parallel each time ($P = B$), we arrive at the thread-greedy algorithm of Scherrer et al. [2012]. Figure 1 shows a schematic representation of the parameterization of these special cases.

3 Convergence Analysis

Of course, there is no reason to expect block-greedy CD to converge for all values of B and P. In this section, we give a sufficient condition for convergence and derive a convergence rate assuming this condition.

Bradley et al. express the convergence criteria for Shotgun algorithm in terms of the spectral radius (maximal eigenvalue) $\rho(\mathbf{X}^T\mathbf{X})$. For block-greedy, the corresponding quantity is a bit more complicated. We define

$$\rho_{\text{block}} = \max_{M \in \mathcal{M}} \rho(M)$$

where $\mathcal{M}$ is the set of all $B \times B$ submatrices that we can obtain from $\mathbf{X}^T\mathbf{X}$ by selecting exactly one index from each of the B blocks. The intuition is that if features from different blocks are almost orthogonal then the matrices M in $\mathcal{M}$ will be close to identity and will therefore have small $\rho(M)$. Highly correlated features *within* a block do not increase ρ_{block}.

As we said above, we will assume that we are minimizing a "smooth plus separable" convex function $F + R$ where the convex differentiable function $F : \mathbb{R}^p \to \mathbb{R}$ satisfies a second order upper bound

$$F(\mathbf{w} + \Delta) \leq F(\mathbf{w}) + \nabla F(\mathbf{w})^T \Delta + \frac{\beta}{2} \Delta^T \mathbf{X}^T \mathbf{X} \Delta$$

In our case, this inequality will hold as soon as $\ell''(y, t) \leq \beta$ for any y, t (where differentiation is w.r.t. t). The function R is separable across coordinates: $R(\mathbf{w}) = \sum_{j=1}^{p} r(w_j)$. The function $\lambda \|\mathbf{w}\|_1$ is clearly separable.

The quantity η_j appearing in Algorithm 1 serves to quantify the guaranteed descent (based on second order upper bound) if feature j alone is updated and is obtained as a solution of the one-dimensional minimization problem

$$\eta_j = \operatorname*{argmin}_{\eta} \ \nabla_j F(\mathbf{w})\eta + \frac{\beta}{2}\eta^2 + r(w_j + \eta) - r(w_j) \ .$$

Note that if there is no regularization, then η_j is simply $-\nabla_j F(\mathbf{w})/\beta = -g_j/\beta$ (if we denote $\nabla_j F(\mathbf{w})$ by g_j for brevity). In the general case, by first order optimality conditions for the above one-dimensional convex optimization problem, we have $g_j + \beta\eta_j + \nu_j = 0$ where ν_j is a subgradient of r at $w_j + \eta_j$. That is, $\nu_j \in \partial r(w_j + \eta_j)$. This implies that $r(w_j + \eta_j) - r(w') \leq \nu_j(w_j + \eta_j - w')$ for any w'.

Theorem 1. *Let P be chosen so that*

$$\epsilon = \frac{(P-1)(\rho_{\text{block}} - 1)}{(B-1)}$$

is less than 1. *Suppose the randomized block-greedy coordinate algorithm is run on a smooth plus separable convex function* $f = F + R$ *to produce the iterates* $\{\mathbf{w}_k\}_{k\geq 1}$. *Then the expected accuracy after* k *steps is bounded as*

$$\mathbb{E}\left[f(\mathbf{w}_k) - f(\mathbf{w}^\star)\right] \leq C\frac{B\,R_1^2}{(1-\epsilon)P}\cdot\frac{1}{k}\,.$$

Here the constant C *only depends on (Lipschitz and smoothness constants of) the function* F, R_1 *is an upper bound on the norms* $\{\|\mathbf{w}_k - \mathbf{w}^\star\|_1\}_{k\geq 1}$, *and* $\mathbf{w}^\star$ *is any minimizer of* f.

Proof. We first calculate the expected change in objective function following the Shotgun analysis. We will use w_b to denote w_{j_b} (similarly for ν_b, g_b etc.)

$$\begin{aligned}\mathbb{E}\left[f(\mathbf{w}') - f(\mathbf{w})\right] = P\mathbb{E}_b\left[\eta_b g_b + \frac{\beta}{2}(\eta_b)^2 + r(w_b+\eta_b) - r(w_b)\right]\\ + \frac{\beta}{2}P(P-1)\mathbb{E}_{b\neq b'}\left[\eta_b\cdot\eta_{b'}\cdot A_{j_b}^T A_{j_{b'}}\right]\end{aligned}$$

Define the $B\times B$ matrix M (that depends on the current iterate $\mathbf{w}$) with entries $M_{b,b'} = A_{j_b}^T A_{j_b}$. Then, using $r(w_b+\eta_b) - r(w_b) \leq \nu_b\eta_b$, we continue

$$\leq \frac{P}{B}\left[\eta^T g + \frac{\beta}{2}\eta^T\eta + \nu^T\eta\right] + \frac{\beta P(P-1)}{2B(B-1)}\left[\eta^\top M\eta - \eta^T\eta\right]$$

Above (with some abuse of notation), η, ν and g are B length vectors with components η_b, ν_b and g_b respectively. By definition of ρ_{block}, we have $\eta^\top M\eta \leq \rho_{\text{block}}\eta^T\eta$. So, we continue

$$\leq \frac{P}{B}\left[\eta^T g + \frac{\beta}{2}\eta^T\eta - g^T\eta - \beta\eta^T\eta\right] + \frac{\beta P(P-1)}{2B(B-1)}(\rho_{\text{block}}-1)\eta^T\eta$$

where we used $\nu = -g - \beta\eta$. Simplifying we get

$$\mathbb{E}\left[f(\mathbf{w}') - f(\mathbf{w})\right] \leq \frac{P\beta}{2B}\left[-1+\epsilon\right]\|\eta\|_2^2$$

where

$$\epsilon = \frac{(P-1)(\rho_{\text{block}}-1)}{(B-1)}$$

should be less than 1.

Now note that $\|\eta\|_2^2 = \sum_b |\eta_{j_b}|^2 = \|\eta\|_{\infty,2}^2$ where the "infinity-2" norm $\|\cdot\|_{\infty,2}$ of a p-vector is, by definition, as follows: take the ℓ_∞ norm within a block and take the ℓ_2 of the resulting values. Note that in the second step above, we moved from a B-length η to a p length η.

This gives us

$$\mathbb{E}\left[f(\mathbf{w}') - f(\mathbf{w})\right] \leq -\frac{(1-\epsilon)P\beta}{2B}\|\eta\|_{\infty,2}^2\,.$$

For the rest of the proof, assume $\lambda = 0$. In that case $\eta = -g/\beta$. Thus, convexity and the fact that the dual norm of the "infinity-2" norm is the "1-2" norm, give

$$f(\mathbf{w}) - f(\mathbf{w}^\star) \leq \nabla f(\mathbf{w})^T(\mathbf{w}-\mathbf{w}^\star) \leq \|\nabla f(\mathbf{w})\|_{\infty,2}\cdot\|\mathbf{w}-\mathbf{w}^\star\|_{1,2}$$

Putting the last two inequalities together gives (for any upper bound R_1 on $\|\mathbf{w}-\mathbf{w}^\star\|_1 \geq \|\mathbf{w}-\mathbf{w}^\star\|_{1,2}$)

$$\mathbb{E}\left[f(\mathbf{w}') - f(\mathbf{w})\right] \leq -\frac{(1-\epsilon)P}{2\beta BR_1^2}(f(\mathbf{w}) - f(\mathbf{w}^\star))^2\,.$$

Defining the accuracy $\alpha_k = f(\mathbf{w}_k) - f(\mathbf{w}^\star)$, we translate the above into the recurrence

$$\mathbb{E}\left[\alpha_{k+1} - \alpha_k\right] \leq -\frac{(1-\epsilon)P}{2\beta BR_1^2}\mathbb{E}\left[\alpha_k^2\right]$$

and by Jensen's we have $(\mathbb{E}[\alpha_k])^2 \leq \mathbb{E}\left[\alpha_k^2\right]$ and therefore

$$\mathbb{E}[\alpha_{k+1}] - \mathbb{E}[\alpha_k] \leq -\frac{(1-\epsilon)P}{2\beta B R_1^2}(\mathbb{E}[\alpha_k])^2$$

which solves to (up to a universal constant factor)

$$\mathbb{E}[\alpha_k] \leq \frac{2\beta B R_1^2}{(1-\epsilon)P} \cdot \frac{1}{k}$$

Even when $\lambda > 0$, we can still relate $\|\eta\|_{\infty,2}$ to $f(\mathbf{w}) - f(\mathbf{w}^\star)$ but the argument is a little more involved. We refer the reader to the supplementary for more details. □

In particular, consider the case where all blocks are updated in parallel as in the thread-greedy coordinate descent algorithm of Scherrer et al. [2012]. Then $P = B$ and there is no randomness in the algorithm, yielding the following corollary.

Corollary 2. *Suppose the block-greedy coordinate algorithm with $B = P$ (thready-greedy) is run on a smooth plus separable convex function $f = F + R$ to produce the iterates $\{\mathbf{w}_k\}_{k\geq 1}$. If $\rho_{\text{block}} < 2$, then*

$$f(\mathbf{w}_k) - f(\mathbf{w}^\star) = O\left(\frac{1}{(2-\rho_{\text{block}})k}\right) .$$

4 Feature Clustering

The convergence analysis of section 3 shows that we need to minimize the block spectral radius. Directly finding a clustering that minimizes ρ_{block} is a computationally daunting task. Even with equal-sized blocks, the number of possible partitions is $p!/\left(\frac{p}{B}\right)^B$. In the absence of an efficient search strategy for this enormous space, we find it convenient to work instead in terms of the inner product of features from distinct blocks. The following proposition makes the connection between these approaches precise.

Proposition 3. *Let $S \in \mathbb{R}^{B\times B}$ be positive semidefinite, with $S_{ii} = 1$, and $|S_{ij}| < \varepsilon$ for $i \neq j$. Then the spectral radius of S has the upper bound*

$$\rho(S) \leq 1 + (B-1)\,\varepsilon .$$

Proof. Let x be the eigenvector corresponding to the largest eigenvalue of S, scaled so that $\|x\|_1 = 1$. Then

$$\rho(S) = \|Sx\|_1 = \sum_i \left| x_i + S_{ij} \sum_{j\neq i} x_j \right| \leq \sum_i \left(|x_i| + \varepsilon \sum_{j\neq i} |x_j| \right) = 1 + (B-1)\,\varepsilon$$

□

Proposition 3 tells us that we can partition the features into clusters using a heuristic approach that strives to minimize the maximum absolute inner product between the features (columns of the design matrix) $\mathbf{X}_i$ and $\mathbf{X}_j$ where i and j are features in different blocks.

4.1 Clustering Heuristic

Given p features and B blocks, we wish to distribute the features evenly among the blocks, attempting to minimize the absolute inner product between features across blocks. Moreover, we require an approach that is efficient, since any time spent clustering could instead have been used for iterations of the main algorithm. We describe a simple heuristic that builds uniform-sized clusters of features.

To construct a given block, we select a feature as a "seed", and assign the nearest features to the seed (in terms of absolute inner product) to be in the same block. Because inner products with very sparse features result in a large number of zeros, we choose at each step the most dense unassigned feature as the seed. Algorithm 2 provides a detailed description. This heuristic requires computation of $O(Bp)$ inner products. In practice it is very fast—less than three seconds for even the large KDDA dataset.

Algorithm 2 A heuristic for clustering p features into B blocks, based on correlation

$U \leftarrow \{1, \cdots, p\}$
for $b = 1$ to $B - 1$ **do**
  $s \leftarrow \arg\max_{j \in U} \mathrm{NNZ}(\mathbf{X}_j)$
  for $j \in U$ **do** // parallel
    $c_j \leftarrow |\langle \mathbf{X}_s, \mathbf{X}_j \rangle|$
  $J_b \leftarrow \{j$ yielding the $\lceil p/B \rceil$ largest values of $c_j\}$
  $U \leftarrow U \backslash J_b$
$J_B \leftarrow U$
return $\{J_b | b = 1, \cdots, B\}$

Name	# Features	# Samples	# Nonzeros	Source
NEWS20	$1,355,191$	$19,996$	$9,097,916$	Keerthi and DeCoste [2005]
REUTERS	$47,237$	$23,865$	$1,757,800$	Lewis et al. [2004]
REALSIM	$20,958$	$72,309$	$3,709,083$	RealSim
KDDA	$20,216,830$	$8,407,752$	$305,613,510$	Lo et al. [2011]

Table 1: Summary of input characteristics.

5 Experimental Setup

Platform All our experiments are conducted on a 48-core system comprising of 4 sockets and 8 banks of memory. Each socket is an AMD Opteron processor codenamed Magny-Cours, which is a multichip processor with two 6-core chips on a single die. Each 6-core processor is equipped with a three-level memory hierarchy as follows: (i) 64 KB of `L1` cache for data and 512 KB of `L2` cache that are private to each core, and (ii) 12 MB of `L3` cache that is shared among the 6 cores. Each 6-core processor is linked to a 32 GB memory bank with independent memory controllers leading to a total system memory of 256 GB (32×8) that can be globally addressed from each core. The four sockets are interconnected using HyperTransport-3 technology[1].

Datasets A variety of datasets were chosen[2] for experimentation; these are summarized in Table 1. We consider four datasets: (i) NEWS20 contains about $20,000$ UseNet postings from 20 newsgroups. The data was gathered by Ken Lang at Carnegie Mellon University circa 1995. (ii) REUTERS is the RCV1-v2/LYRL2004 Reuters text data described by Lewis et al. [2004]. In this term-document matrix, each example is a training document, and each feature is a term. Nonzero values of the matrix correspond to term frequencies that are transformed using a standard tf-idf normalization. (iii) REALSIM consists of about $73,000$ UseNet articles from four discussion groups: simulated auto racing, simulated aviation, real auto racing, and real aviation. The data was gathered by Andrew McCallum while at Just Research circa 1997. We consider classifying real vs simulated data, irrespective of auto/aviation. (iv) KDDA represents data from the KDD Cup 2010 challenge on educational data mining. The data represents a processed version of the training set of the first problem, algebra_2008_2009, provided by the winner from the National Taiwan University. These four inputs cover a broad spectrum of sizes and structural properties.

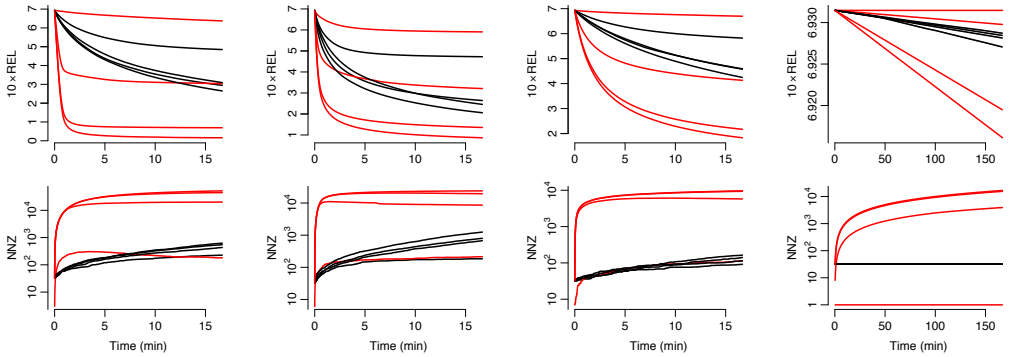
Implementation For the current work, our empirical results focus on thread-greedy coordinate descent with 32 blocks. At each iteration, a given thread must step through the nonzeros of each of its features to compute the proposed increment (the η_j of Section 3) and the estimated benefit of choosing that feature. Once this is complete, the thread (without waiting) enters the line search phase, where it remains until all threads are being updated by less than the specified tolerance. Finally, all updates are performed concurrently. We use OpenMP's atomic directive to maintain consistency.

Testing framework

We compare the effect of clustering to randomization (i.e. features are randomly assigned to blocks), for a variety of values for the regularization parameter λ. To test the effect of clustering for very

[1]Further details on AMD Opteron can be found at `http://www.amd.com/us/products/embedded/processors/opteron/Pages/opteron-6100-series.aspx`.

[2]from `http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/`



(a) NEWS20, $\lambda_0 = 10^{-4}$ (b) REUTERS, $\lambda_0 = 10^{-4}$ (c) REALSIM, $\lambda_0 = 10^{-4}$ (d) KDDA, $\lambda_0 = 10^{-6}$

Figure 2: Convergence results. For each dataset, we show the regularized expected loss (top) and number of nonzeros (bottom), using powers of ten as regularization parameters. Results for randomized features are shown in black, and those for clustered features are shown in red. Note that the allowed running time for KDDA was ten times that of other datasets.

	$\lambda = 10^{-4}$		$\lambda = 10^{-5}$		$\lambda = 10^{-6}$	
	Randomized	Clustered	Randomized	Clustered	Randomized	Clustered
Active blocks	**32**	6	32	32	32	32
Iterations per second	**153**	12.9	**152**	12.3	**136**	12.3
NNZ @ 1K sec	184	215	797	8592	1248	19473
Objective @ 1K sec	**0.472**	0.591	**0.264**	0.321	0.206	**0.136**
NNZ @ 10K iter	74	203	82	8812	110	19919
Objective @ 10K iter	**0.570**	0.593	0.515	**0.328**	0.472	**0.141**

Table 2: The effect of feature clustering, for REUTERS.

sparse weights, we first let λ_0 be the largest power of ten that leads to any nonzero weight estimates. This is followed by the next three consecutive powers of ten. For each run, we measure the regularized expected loss and the number of nonzeros at one-second intervals. Times required for clustering and randomization are negligible, and we do not report them here.

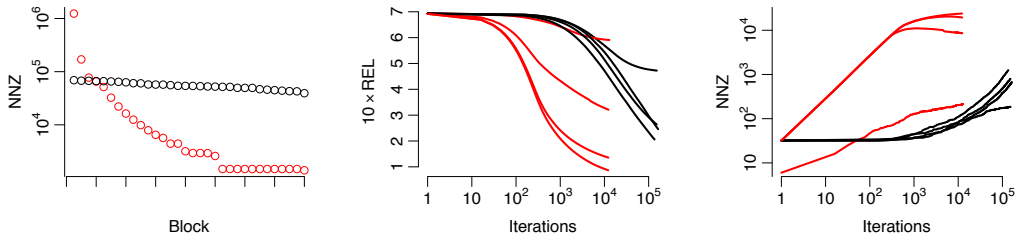
6 Results

Figure 2 shows the regularized expected loss (top) and number of nonzeros (bottom), for several values of the regularization parameter λ. Black and red curves indicate randomly-permuted features and clustered features, respectively. The starting value of λ was 10^{-4} for all data except KDDA, which required $\lambda = 10^{-6}$ in order to yield any nonzero weights.

In the upper plots, within a color, the order of the 4 curves, top to bottom, corresponds to successively decreasing values of λ. Note that a larger value of λ results in a sparser solution with greater regularized expected loss and a smaller number of nonzeros. Thus, for each subfigure of Figure 2, the order of the curves in the lower plot is *reversed* from that of the upper plot.

Overall, results across datasets are very consistent. For large values of λ, the simple clustering heuristic results in slower convergence, while for smaller values of λ we see considerable benefit. Due to space limitations, we choose a single dataset for which to explore results in greater detail.

Of the datasets we tested, REUTERS might reasonably lead to the greatest concern. Like the other datasets, clustered features lead to slow convergence for large λ and fast convergence for small λ. However, REUTERS is particularly interesting because for $\lambda = 10^{-5}$, clustered features seem to provide an initial benefit that does not last; after about 250 seconds it is overtaken by the run with randomized features.



(a) Block density (b) Regularized expected loss (c) Number of nonzeros

Figure 3: A closer look at performance characteristics for REUTERS.

Table 2 gives a more detailed summary of the results for REUTERS, for the three largest values of λ. The first row of this table gives the number of *active blocks*, by which we mean the number of blocks containing any nonzeros. For an inactive block, the corresponding thread repeatedly confirms that all weights remain zero without contributing to convergence.

In the most regularized case $\lambda = 10^{-4}$, clustered data results in only six active blocks, while for other cases every block is active. Thus in this case features corresponding to nonzero weights are colocated within these few blocks, severely limiting the advantage of parallel updates.

In the second row, we see that for randomized features, the algorithm is able to get through over ten times as many iterations per second. To see why, note that the amount of work for a given thread is a linear function of the number of nonzeros over all of the features in its block. Thus, the block with the greatest number of nonzeros serves as a bottleneck.

The middle two rows of Figure 2 summarize the state of each run after 1000 seconds. Note that for this test, randomized features result in faster convergence for the two largest values of λ.

For comparison, the final two rows of Figure 2 provide a similar summary based instead on the number of iterations. In these terms, clustering is advantageous for all but the largest value of λ.

Figure 3 shows the source of this problem. First, Figure 3a shows the number of nonzeros in all features for each of the 32 blocks. Clearly the simple heuristic results in poor load-balancing. For comparison, Figures 3b and 3c show convergence rates as a function of the number of iterations.

7 Conclusion

We have presented convergence results for a family of randomized coordinate descent algorithms that we call *block-greedy coordinate descent*. This family includes Greedy CD, Thread-Greedy CD, Shotgun, and Stochastic CD. We have shown that convergence depends on ρ_{block}, the maximal spectral radius over submatrices of $\mathbf{X}^T\mathbf{X}$ resulting from the choice of one feature for each block.

Even though a simple clustering heuristic helps for smaller values of the regularization parameter, our results also show the importance of considering issues of load-balancing and the distribution of weights for heavily-regularized problems.

A clear next goal in this work is the development of a clustering heuristic that is relatively well load-balanced and distributes weights for heavily-regularized problems evenly across blocks, while maintaining good computational efficiency.

Acknowledgments

The authors are grateful for the helpful suggestions of Ken Jarman, Joseph Manzano, and our anonymous reviewers.

Funding for this work was provided by the Center for Adaptive Super Computing Software - MultiThreaded Architectures (CASS-MT) at the U.S. Department of Energy's Pacific Northwest National Laboratory. PNNL is operated by Battelle Memorial Institute under Contract DE-ACO6-76RL01830.

References

J Friedman, T Hastie, H Höfling, and R Tibshirani. Pathwise coordinate optimization. *Annals of Applied Statistics*, 1(2):302–332, 2007.

T Wu and K Lange. Coordinate descent algorithms for lasso penalized regression. *Annals of Applied Statistics*, 2:224–244, 2008.

S Shalev-Shwartz and A Tewari. Stochastic methods for ℓ_1-regularized loss minimization. *Journal of Machine Learning Research*, 12:1865–1892, 2011.

J K Bradley, A Kyrola, D Bickson, and C Guestrin. Parallel Coordinate Descent for L1-Regularized Loss Minimization. In *Proceedings of the 28th International Conference on Machine Learning*, pages 321–328, 2011.

C Scherrer, A Tewari, M Halappanavar, and D Haglin. Scaling up Parallel Coordinate Descent Algorithms. In *International Conference on Machine Learning*, 2012.

Y Li and S Osher. Coordinate Descent Optimization for ℓ_1 Minimization with Application to Compressed Sensing ; a Greedy Algorithm Solving the Unconstrained Problem. *Inverse Problems and Imaging*, 3:487–503, 2009.

I S Dhillon, P Ravikumar, and A Tewari. Nearest neighbor based greedy coordinate descent. In *Advances in Neural Information Processing Systems 24*, pages 2160–2168, 2011.

D Lewis, Y Yang, T Rose, and F Li. RCV1 : A New Benchmark Collection for Text Categorization Research. *Journal of Machine Learning Research*, 5:361–397, 2004.

S S Keerthi and D DeCoste. A modified finite Newton method for fast solution of large scale linear SVMs. *Journal of Machine Learning Research*, 6:341–361, 2005.

RealSim. Document classification data gathered by Andrew McCallum., circa 1997. `URL:http://people.cs.umass.edu/~mccallum/data.html`.

Hung-Yi Lo, Kai-Wei Chang, Shang-Tse Chen, Tsung-Hsien Chiang, Chun-Sung Ferng, Cho-Jui Hsieh, Yi-Kuang Ko, Tsung-Ting Kuo, Hung-Che Lai, Ken-Yi Lin, Chia-Hsuan Wang, Hsiang-Fu Yu, Chih-Jen Lin, Hsuan-Tien Lin, and Shou de Lin. Feature engineering and classifier ensemble for KDD Cup 2010, 2011. To appear in JMLR Workshop and Conference Proceedings.