Optimization Algorithms for Analyzing Large Datasets

Michael W. Mahoney[1]

ICSI and Dept of Statistics, University of California at Berkeley

PCMI Summer School on the Mathematics of Data, July 2016

[1]Slides are due to Stephen Wright, UW-Madison (Winedale, October, 2012), who will be giving the optimization course in a few weeks.

1. Learning from Data: Problems and Optimization Formulations
2. First-Order Methods
3. Stochastic Gradient Methods
4. Nonconvex Stochastic Gradient
5. Sparse / Regularized Optimization
6. Decomposition / Coordinate Relaxation
7. Lagrangian Methods
8. Higher-Order Methods

Outline

Give some examples of data analysis and learning problems and their formulations as optimization problems.

Highlight common features in these formulations.

Survey a number of optimization techniques that are being applied to these problems.

- Just the basics - give the flavor of each method and state a key theoretical result or two.
- All of these approaches are being investigated at present; rapid developments continue.

Data Analysis: Learning

Typical ingredients of optimization formulations of data analysis problems:

- Collection of data, from which we want to learn to make inferences about future / missing data, or do regression to extract key information.
- A model of how the data relates to the meaining we are trying to extract. Model parameters to be determined by inspecting the data and using prior knowledge.
- Objective that captures prediction errors on the data and deviation from prior knowledge or desirable structure.

Other typical properties of learning problems are huge underlying data set, and requirement for solutions with only low-medium accuracy.

In some cases, the optimization formulation is well settled. (e.g. Least Squares, Robust Regression, Support Vector Machines, Logistic Regression, Recommender Systems.)

In other areas, formulation is a matter of ongoing debate!

LASSO

Least Squares: Given a set of feature vectors $a_i \in \mathbb{R}^n$ and outcomes b_i, $i = 1, 2, \ldots, m$, find weights x on the features that predict the outcome accurately: $a_i^T x \approx b_i$.

Under certain assumptions on measurement error, can find a suitable x by solving a least squares problem

$$\min_x \frac{1}{2}\|Ax - b\|_2^2,$$

where the rows of A are a_i^T, $i = 1, 2, \ldots, m$.

Suppose we want to identify the most important features of each a_i — those features most important in predicting the outcome.

In other words, we seek an approx least-squares solution x that is sparse — and x with just a few nonzero components. Obtain

$$\textbf{LASSO:} \quad \min_x \frac{1}{2}\|Ax - b\|_2^2 \text{ such that } \|x\|_1 \leq T,$$

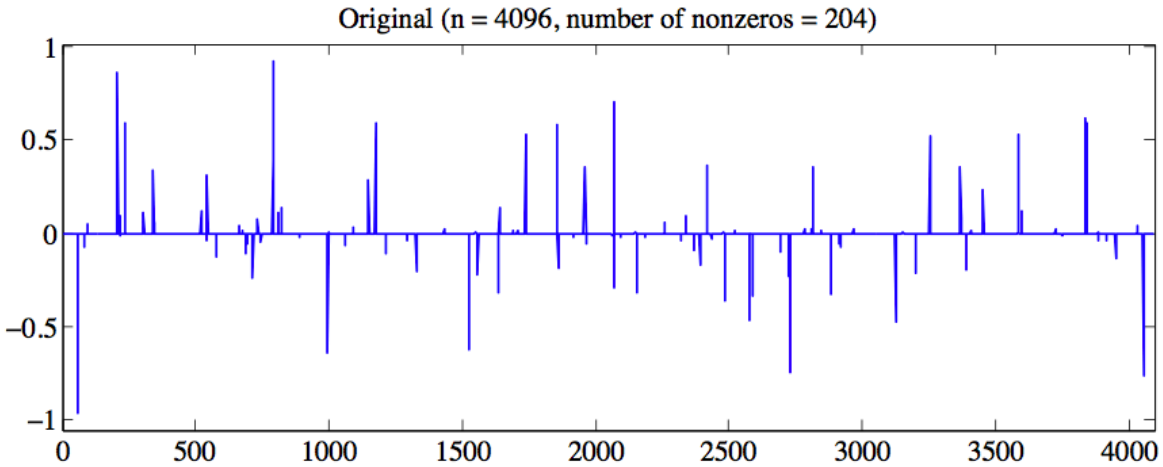
for parameter $T > 0$. (Smaller T implies fewer nonzeros in x.)

LASSO and Compressed Sensing

LASSO is equivalent to an "ℓ_2-ℓ_1" formulation:

$$\min_x \frac{1}{2}\|Ax - b\|_2^2 + \tau\|x\|_1, \qquad \text{for some } \tau > 0.$$

This same formulation is common in compressed sensing, but the motivation is slightly different. Here x represents some signal that is known to be (nearly) sparse, and the rows of A are probing x to give measured outcomes b. The problem above is solved to reconstruct x.



Group LASSO

Partition x into groups of variables that have some relationship — turn them "on" or "off" as a group, not as individuals.

$$\min_x \frac{1}{2}\|Ax - b\|_2^2 + \tau \sum_{g \in G} \|x_{[g]}\|_2,$$

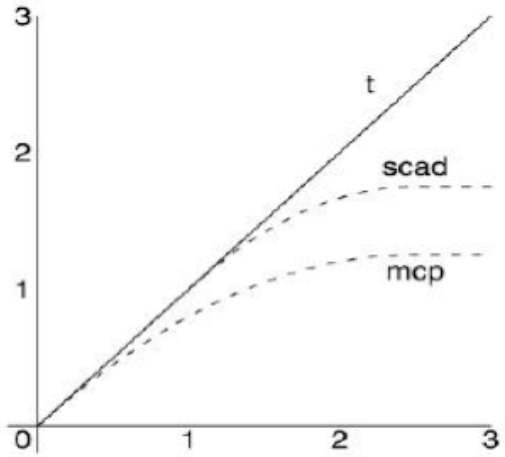
with each $[g] \subset \{1, 2, \ldots, n\}$.

- Easy handle when groups $[g]$ are disjoint.
- Still easy when $\|\cdot\|_2$ is replaced by $\|\cdot\|_\infty$. (Turlach, Venables, Wright, 2005)
- When the groups form a hierarchy, the problem is slightly harder but similar algorithms still work.
- For general overlapping groups, algorithms are more complex (see Bach, Mairal, et al.)

Least Squares with Nonconvex Regularizers

Nonconvex element-wise penalties have become popular for variable selection in statistics.

- SCAD (smoothed clipped absolute deviation) (Fan and Li, 2001)
- MCP (Zhang, 2010).



Properties: Unbiased and sparse estimates, solution path continuous in regularization parameter τ.

SparseNet (Mazumder, Friedman, Hastie, 2011): coordinate descent.

Support Vector Classification

Given many data vectors $x_i \in \mathbb{R}^n$, for $i = 1, 2, \ldots, m$, together with a label $y_i = \pm 1$ to indicate the class (one of two) to which x_i belongs.

Find z such that (usually) we have

- $x_i^T z \geq 1$ when $y_i = +1$;
- $x_i^T z \leq -1$ when $y_i = -1$.

SVM with hinge loss:

$$f(z) = C \sum_{i=1}^{N} \max(1 - y_i(z^T x_i), 0) + \frac{1}{2}\|z\|^2,$$

where $C > 0$ is a parameter. **Dual formulation** is

$$\min_{\alpha} \frac{1}{2}\alpha^T K \alpha - \mathbf{1}^T \alpha \quad \text{subject to } 0 \leq \alpha \leq C\mathbf{1},\ y^T \alpha = 0,$$

where $K_{ij} = y_i y_j x_i^T x_j$. Subvectors of the gradient $K\alpha - \mathbf{1}$ can be computed economically.

(Many extensions and variants.)

(Regularized) Logistic Regression

Seek odds function parametrized by $z \in \mathbb{R}^n$:

$$p_+(x; z) := (1 + e^{z^T x})^{-1}, \quad p_-(x; z) := 1 - p_+(z; w),$$

choosing z so that $p_+(x_i; z) \approx 1$ when $y_i = +1$ and $p_-(x_i; z) \approx 1$ when $y_i = -1$. Scaled, negative log likelihood function $\mathcal{L}(z)$ is

$$\mathcal{L}(z) = -\frac{1}{m} \left[\sum_{y_i=-1} \log p_-(x_i; z) + \sum_{y_i=1} \log p_+(x_i; z) \right]$$

To get a sparse z (i.e. classify on the basis of a few features) solve:

$$\min_z \mathcal{L}(z) + \lambda \|z\|_1.$$

Multiclass Logistic Regression

M **classes**: $y_{ij} = 1$ if data point i is in class j; $y_{ij} = 0$ otherwise. $z_{[j]}$ is the subvector of z for class j.

$$f(z) = -\frac{1}{N} \sum_{i=1}^{N} \left[\sum_{j=1}^{M} y_{ij} (z_{[j]}^T x_i) - \log(\sum_{j=1}^{M} \exp(z_{[j]}^T x_i)) \right] + \lambda \sum_{j=1}^{M} \|z_{[j]}\|_2^2.$$

Useful in speech recognition, to classify phonemes.

Matrix Completion

Seek a matrix $X \in \mathbb{R}^{m \times n}$ with low rank that matches certain observations, possibly noisy.

$$\min_X \frac{1}{2}\|\mathcal{A}(X) - b\|_2^2 + \tau\psi(X),$$

where $\mathcal{A}(X)$ is a linear mapping of the components of X (e.g. element-wise observations).

Can have ψ as the nuclear norm = sum of singular values. Tends to promote low rank (in the same way as $\|x\|_1$ tends to promote sparsity of a vector x).

Alternatively: X is the sum of sparse matrix and a low-rank matrix. The element-wise 1-norm $\|X\|_1$ is useful in inducing sparsity.

Useful in recommender systems, e.g. Netflix, Amazon.

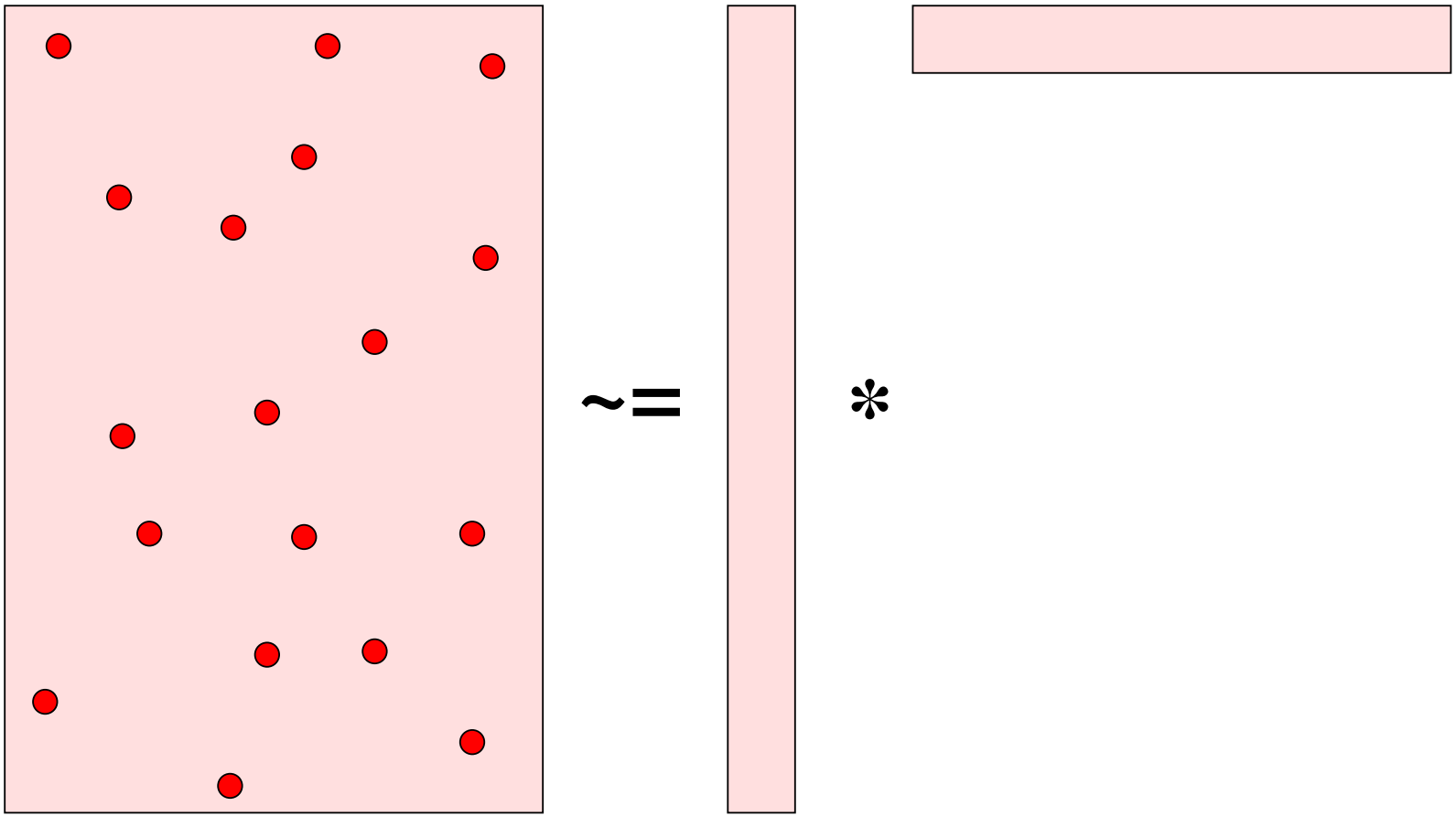


Image Processing: TV Regularization

Natural images are not random! They tend to have large areas of near-constant intensity or color, separated by sharp edges.

Denoising: Given an image in which the pixels contain noise, find a "nearby natural image."

Total-Variation Regularization applies an ℓ_1 penalty to gradients in the image.

$$u : \Omega \to \mathbb{R}, \qquad \Omega := [0,1] \times [0,1],$$

To denoise an image $f : \Omega \to \mathbb{R}$, solve

$$\min_u \int_\Omega (u(x) - f(x))^2 \, dx + \lambda \int_\Omega \|\nabla u(x)\|_2^2 \, dx.$$

(Rudin, Osher, Fatemi, 1992)

(a) Cameraman: Clean

(b) Cameraman: Noisy

(c) Cameraman: Denoised

(d) Cameraman: Noisy

Optimization: Regularization Induces Structure

Formulations may include regularization functions to induce structure:

$$\min_x f(x) + \tau\psi(x),$$

where ψ induces the desired structure in x. Often ψ is nonsmooth.

- $\|x\|_1$ to induce sparsity in the vector x (variable selection / LASSO);
- SCAD and MCP: nonconvex regularizers to induce sparsity, without biasing the solution;
- Group regularization / group selection;
- Nuclear norm $\|X\|_*$ (sum of singular values) to induce low rank in matrix X;
- low "total-variation" in image processing;
- generalizability (Vapnik: "...tradeoff between the quality of the approximation of the given data and the complexity of the approximating function").

Optimization: Properties of f.

Objective f can be derived from Bayesian statistics + maximum likelihood criterion. Can incorporate prior knowledge.

f have distinctive properties in several applications:

- Partially Separable: Typically

$$f(x) = \frac{1}{m} \sum_{i=1}^{m} f_i(x),$$

  where each term f_i corresponds to a single item of data, and possibly depends on just a few components of x.
- Cheap Partial Gradients: subvectors of the gradient ∇f may be available at proportionately lower cost than the full gradient.
- These two properties are often combined.

Batch vs Incremental

Considering the partially separable form

$$f(x) = \frac{1}{m} \sum_{i=1}^{m} f_i(x),$$

the size N of the training set can be very large. There's a fundamental divide in algorithmic strategy.

Incremental: Select a single $i \in \{1, 2, \ldots, m\}$ at random, evaluate $\nabla f_i(x)$, and take a step in this direction. (Note that $E[\nabla f_i(x)] = \nabla f(x)$.) *Stochastic Approximation (SA) a.k.a. Stochastic Gradient (SG).*

Batch: Select a subset of data $X \subset \{1, 2, \ldots, m\}$, and minimize the function $\tilde{f}(x) = \sum_{i \in X} f_i(x)$. *Sample-Average Approximation (SAA).*

Minibatch is a kind of compromise: Aggregate the component functions f_i into small groups, consisting of 10 or 100 individual terms, and apply incremental algorithms to the redefined summation. (Gives lower-variance gradient estimates.)

First-Order Methods

$\min f(x)$, with smooth convex f. First-order methods calculate $\nabla f(x_k)$ at each iteration, and do something with it.

Usually assume

$$\mu I \preceq \nabla^2 f(x) \preceq LI \text{ for all } x,$$

with $0 \leq \mu \leq L$. (L is thus a Lipschitz constant on the gradient ∇f.) Function is strongly convex if $\mu > 0$.

But some of these methods are extendible to more general cases, e.g.

- nonsmooth-regularized f;
- simple constraints $x \in \Omega$;
- availability of just an *approximation* to ∇f.

Steepest Descent

$$x_{k+1} = x_k - \alpha_k \nabla f(x_k), \qquad \text{for some } \alpha_k > 0.$$

Choose step length α_k by doing line search, or more simply by using knowledge of L and μ.

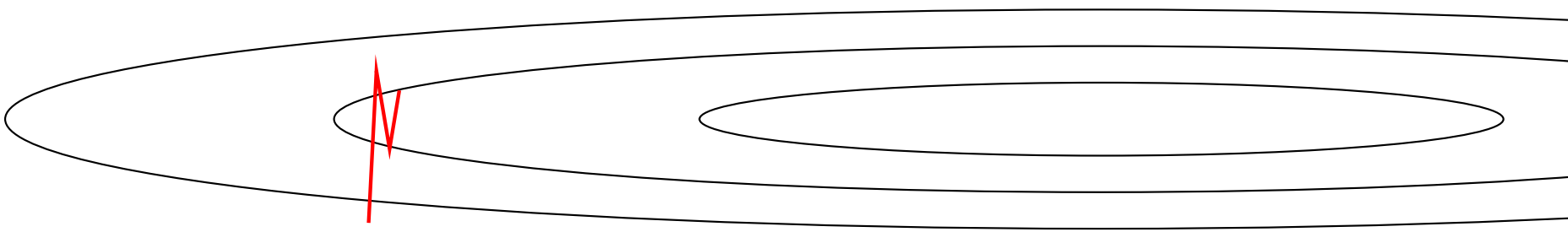
$\alpha_k \equiv 1/L$ yields sublinear convergence

$$f(x_k) - f(x^*) \leq \frac{2L\|x_0 - x^*\|^2}{k}.$$

In strongly convex case, $\alpha_k \equiv 2/(\mu + L)$ yields linear convergence, with constant dependent on problem conditioning $\kappa := L/\mu$:

$$f(x_k) - f(x^*) \leq \frac{L}{2}\left(1 - \frac{2}{\kappa + 1}\right)^{2k} \|x_0 - x^*\|^2.$$

We can't improve much on these rates by using more sophisticated choices of α_k — they're a fundamental limitation of searching along $-\nabla f(x_k)$.



steepest descent, exact line search

Momentum

First-order methods can be improved dramatically using momentum:

$$x_{k+1} = x_k - \alpha_k \nabla f(x_k) + \beta_k (x_k - x_{k-1}).$$

Search direction is a combination of previous search direction $x_k - x_{k-1}$ and latest gradient $\nabla f(x_k)$. Methods in this class include:

- Heavy-ball;
- Conjugate gradient;
- Accelerated gradient.

Heavy-ball sets

$$\alpha_k \equiv \frac{4}{L} \frac{1}{(1 + 1/\sqrt{\kappa})^2}, \quad \beta_k \equiv \left(1 - \frac{2}{\sqrt{\kappa} + 1}\right)^2.$$

to get a linear convergence rate with constant approximately $1 - 2/\sqrt{\kappa}$.

Thus requires about $\sqrt{\kappa} \log \epsilon$ to achieve precision of ϵ, vs. about $\kappa \log \epsilon$ for steepest descent.

Conjugate Gradient

Basic step is

$$x_{k+1} = x_k + \alpha_k p_k, \qquad p_k = -\nabla f(x_k) + \gamma_k p_{k-1}.$$

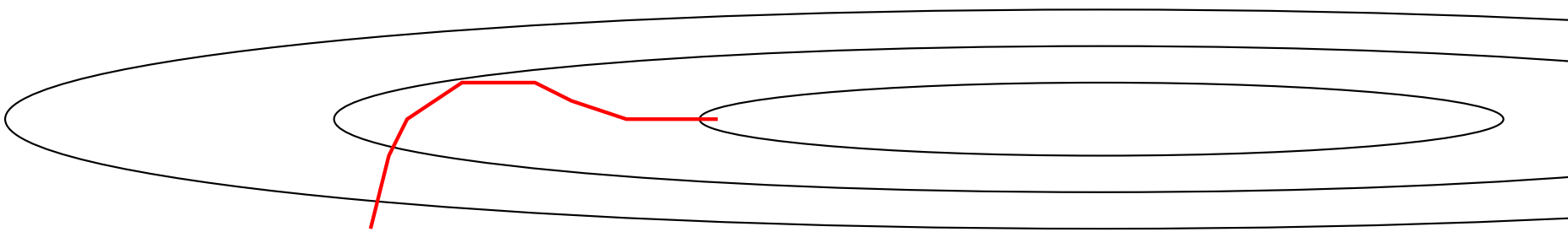
Put it in the "momentum" framework by setting $\beta_k = \alpha_k \gamma_k / \alpha_{k-1}$.
However, CG can be implemented in a way that doesn't require knowledge (or estimation) of L and μ.

- Choose α_k to (approximately) miminize f along p_k;
- Choose γ_k by a variety of formulae (Fletcher-Reeves, Polak-Ribiere, etc), all of which are equivalent if f is convex quadratic. e.g.

$$\gamma_k = \|\nabla f(x_k)\|^2 / \|\nabla f(x_{k-1})\|^2.$$

Similar convergence rate to heavy-ball: linear with rate $1 - 2/\sqrt{\kappa}$.

CG has other properties, particularly when f is convex quadratic.



first–order method with momentum

Accelerated Gradient Methods

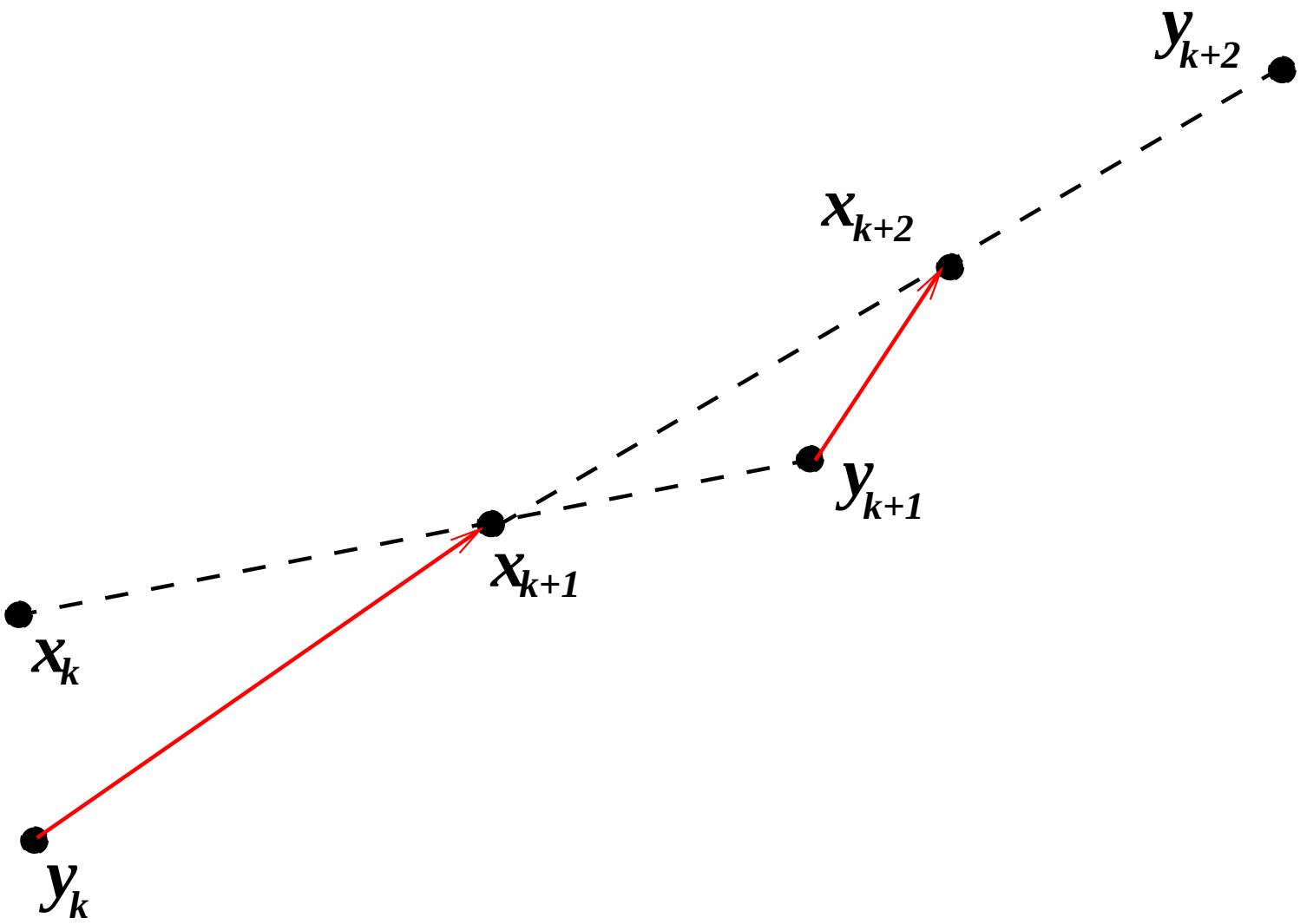
Accelerate the rate to $1/k^2$ for weakly convex, while retaining the linear rate (based on $\sqrt{\kappa}$) for strongly convex case.

Nesterov (1983, 2004) describes a method that requires κ.

0: Choose x_0, $\alpha_0 \in (0,1)$; set $y_0 \leftarrow x_0$./

k: $x_{k+1} \leftarrow y_k - \frac{1}{L}\nabla f(y_k)$; (*short-step gradient*)
solve for $\alpha_{k+1} \in (0,1)$: $\alpha_{k+1}^2 = (1-\alpha_{k+1})\alpha_k^2 + \alpha_{k+1}/\kappa$;
set $\beta_k = \alpha_k(1-\alpha_k)/(\alpha_k^2 + \alpha_{k+1})$;
set $y_{k+1} \leftarrow x_{k+1} + \beta_k(x_{k+1} - x_k)$. (*update with momentum*)

- Separates "steepest descent" step from "momentum" step, producing two sequences $\{x_k\}$ and $\{y_k\}$.
- Still works for weakly convex ($\kappa = \infty$).
- FISTA is similar.


y_{k+2}
x_{k+2}
y_{k+1}
x_{k+1}
x_k
y_k

Extension to Sparse / Regularized Optimization

Accelerated gradient can be applied with minimal changes to the regularized problem

$$\min_x \ f(x) + \tau\psi(x),$$

where f is convex and smooth, ψ convex and "simple" but usually nonsmooth, and τ is a positive parameter.

Simply replace the gradient step by

$$x_k = \arg\min_x \frac{L}{2} \left\| x - \left[y_k - \frac{1}{L}\nabla f(y_k) \right] \right\|^2 + \tau\psi(x).$$

(This is the shrinkage step. Often cheap to compute. More later...)

Stochastic Gradient Methods

Still deal with (weakly or strongly) convex f. But change the rules:

- Allow f nonsmooth.
- Don't calculate function values $f(x)$.
- Can evaluate cheaply an unbiased estimate of a vector from the subgradient ∂f.

Common settings are:

$$f(x) = E_\xi F(x, \xi),$$

where ξ is a random vector with distribution P over a set Ξ. A useful special case is the partially separable form

$$f(x) = \frac{1}{m} \sum_{i=1}^{m} f_i(x),$$

where each f_i is convex.

"Classical" Stochastic Gradient

For the sum:

$$f(x) = \frac{1}{m} \sum_{i=1}^{m} f_i(x),$$

Choose index $i_k \in \{1, 2, \ldots, m\}$ uniformly at random at iteration k, set

$$x_{k+1} = x_k - \alpha_k \nabla f_{i_k}(x_k),$$

for some steplength $\alpha_k > 0$.

When f is strongly convex, the analysis of convergence of $E(\|x_k - x^*\|^2)$ is fairly elementary - see Nemirovski et al (2009).

Convergence depends on careful choice of α_k, naturally.
Line search for α_k is not really an option, since we can't (or won't) evaluate f!

Convergence of Classical SG

Suppose f is strongly convex with modulus μ, there is a bound M on the size of the gradient estimates:

$$\frac{1}{m}\sum_{i=1}^{m}\|\nabla f_i(x)\|^2 \leq M^2$$

for all x of interest. Convergence obtained for the *expected square error*:

$$a_k := \frac{1}{2}E(\|x_k - x^*\|^2).$$

Elementary argument shows a recurrence:

$$a_{k+1} \leq (1 - 2\mu\alpha_k)a_k + \frac{1}{2}\alpha_k^2 M^2.$$

When we set $\alpha_k = 1/(k\mu)$, a neat inductive argument reveals a $1/k$ rate:

$$a_k \leq \frac{Q}{2k}, \qquad \text{for } Q := \max\left(\|x_1 - x^*\|^2, \frac{M^2}{\mu^2}\right).$$

But... What if we don't know μ? Or if $\mu = 0$?

The choice $\alpha_k = 1/(k\mu)$ requires strong convexity, with knowledge of μ. Underestimating μ can greatly degrade performance!

Robust Stochastic Approximation approach uses primal averaging to recover a rate of $1/\sqrt{k}$ in f, works even for weakly convex nonsmooth functions, and is not sensitive to choice of parameters in the step length.

At iteration k:

- set $x_{k+1} = x_k - \alpha_k \nabla f_{i_k}(x_k)$ as before;
- set

$$\bar{x}_k = \frac{\sum_{i=1}^k \alpha_i x_i}{\sum_{i=1}^k \alpha_i}.$$

For any $\theta > 0$ (not critical), choose step lengths to be $\alpha_k = \theta/(M\sqrt{k})$. Then

$$E[f(\bar{x}_k) - f(x^*)] \sim \frac{\log k}{\sqrt{k}}.$$

Constant Step Size: $\alpha_k \equiv \alpha$

We can also get rates of approximately $1/k$ for the strongly convex case, *without* performing iterate averaging and without requiring an accurate estimate of μ. Need to specify the desired threshold for a_k in advance.

Setting $\alpha_k \equiv \alpha$, we have

$$a_{k+1} \leq (1 - 2\mu\alpha)a_k + \frac{1}{2}\alpha^2 M^2.$$

We can show by some manipulation that

$$a_k \leq (1 - 2\mu\alpha)^k a_0 + \frac{\alpha M^2}{4\mu}.$$

Given threshold $\epsilon > 0$, we aim to find α and K such that $a_k \leq \epsilon$ for all $k \geq K$. Choose so that both terms in the bound are less than $\epsilon/2$:

$$\alpha := \frac{2\epsilon\mu}{M^2}, \qquad K := \frac{M^2}{4\epsilon\mu^2} \log\left(\frac{a_0}{2\epsilon}\right).$$

A kind of $1/k$ rate!

Parallel Stochastic Gradient

Several approaches tried for parallel stochastic approximation.

- **Dual Averaging:** Average gradient estimates evaluated in parallel on different cores. Requires message passing / synchronization (Dekel et al, 2011; Duchi et al, 2010).
- **Round-Robin:** Cores evaluate ∇f_i in parallel and update centrally stored x in round-robin fashion. Requires synchronization (Langford et al, 2009).
- **Asynchronous:** HOGWILD!**:** Each core grabs the centrally-stored x and evaluates $\nabla f_e(x_e)$ for some random e, then writes the updates back into x (Niu et al, 2011). **Downpour SGD:** Similar idea for cluster (Dean et al, 2012).

HOGWILD!: Each processor runs independently:

1. Sample i_k from $\{1, 2, \ldots, m\}$;
2. Read current state of x from central memory, evalute $g := \nabla f_{i_k}(x)$;
3. **for** nonzero components g_v **do** $x_v \leftarrow x_v - \alpha g_v$;

HOGWILD! Convergence

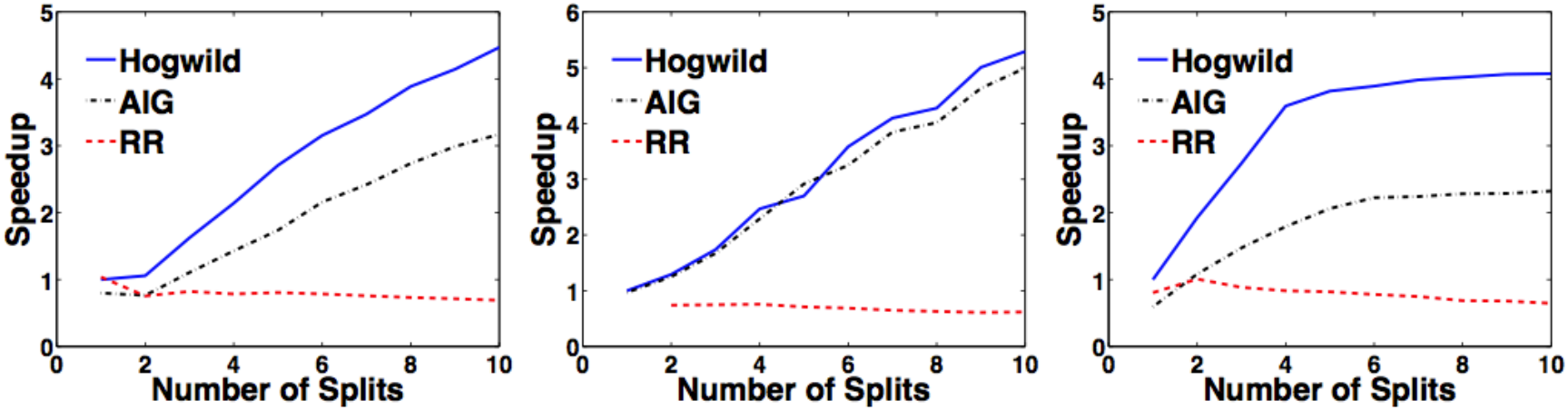
- Updates can be old by the time they are applied, but we assume a bound τ on their age.
- Processors can overwrite each other's work, but sparsity of ∇f_e helps — updates to not interfere too much.

Analysis of Niu et al (2011) simplified / generalized by Richtarik (2012).

Rates depend on τ, L, μ, initial error, and other quantities that define the amount of overlap between nonzero components of ∇f_i and ∇f_j, for $i \neq j$.

For a constant-step scheme (with α chosen as a function of the quantities above), essentially recover the $1/k$ behavior of basic SG. (Also depends linearly on τ and the average sparsity.)

HOGWILD! Performance



HOGWILD! compared with averaged gradient (AIG) and round-robin (RR). Experiments run on a 12-core machine. (10 cores used for gradient evaluations, 2 cores for data shuffling.)

HOGWILD! Performance

	data set	size (GB)	ρ	Δ	time (s)	speedup
SVM	RCV1	0.9	4.4E-01	1.0E+00	10	4.5
	Netflix	1.5	2.5E-03	2.3E-03	301	5.3
MC	KDD	3.9	3.0E-03	1.8E-03	878	5.2
	JUMBO	30	2.6E-07	1.4E-07	9,454	6.8
	DBLife	0.003	8.6E-03	4.3E-03	230	8.8
CUTS	Abdomen	18	9.2E-04	9.2E-04	1,181	4.1

Stochastic Gradient for Nonconvex Objectives

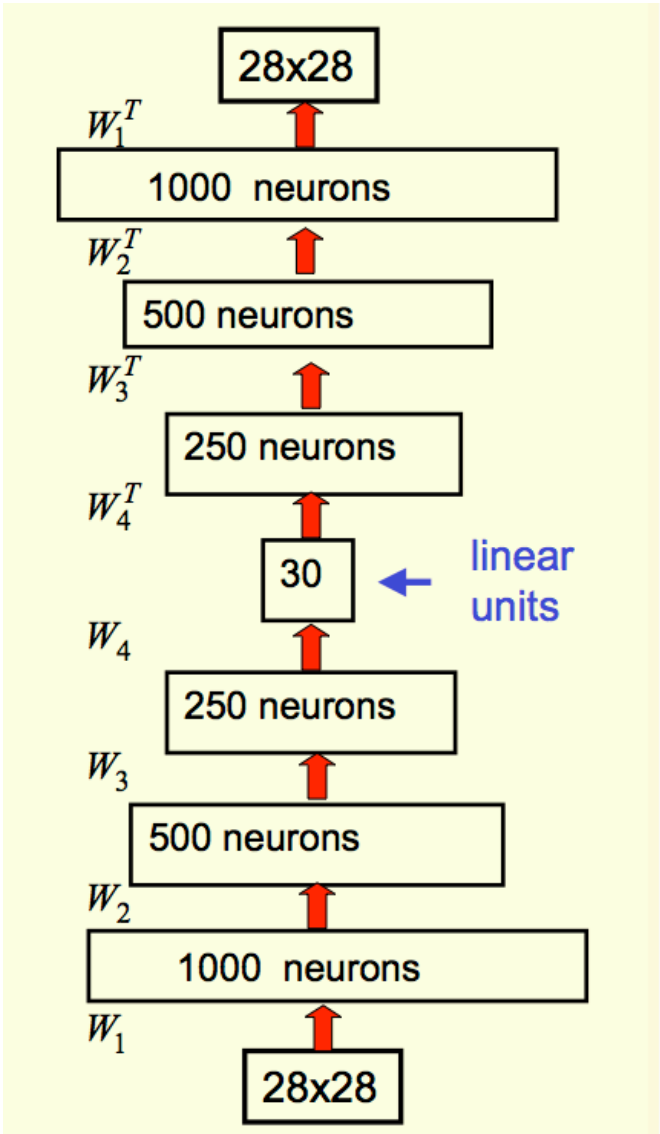
Derivation and analysis of SG methods depends crucially on convexity of f — even strong convexity. However there are data-intensive problems of current interest (e.g. in deep belief networks) with separable nonconvex f.

For nonconvex problems, standard convergence results for full-gradient methods are that "accumulation points are stationary," that is, $\nabla f(\bar{x}) = 0$.

Are there stochastic-gradient methods that achieve similar properties?

Yes! See Ghadimi and Lan (2012).

Deep Belief Networks



Example of a deep belief network for autoencoding (Hinton, 2007). Output (at top) depends on input (at bottom) of an image with 28×28 pixels. Transformations parametrized by W_1, W_2, W_3, W_4; output is a highly nonlinear function of these parameters.

Learning problems based on structures like this give rise to separable, nonconvex objectives.

Nonconvex SG

Describe for the problem

$$f(x) = \frac{1}{m} \sum_{i=1}^{m} f_i(x),$$

with f_i smooth, possibly nonconvex. L is Lipschitz constant for ∇f; M defined as before.

Approach of Ghadimi and Lan (2012): SG with a random number of steps.

- Pick an iteration limit N; Pick stepsize sequence α_k, $k = 1, 2, \ldots, N$;
- Pick R randomly and uniformly from $\{1, 2, \ldots, N\}$;
- Do R steps of SG: For $k = 1, 2, \ldots, R-1$:

$$x_{k+1} = x_k - \alpha_k \nabla f_{i_k}(x_k)$$

  with i_k chosen randomly and uniformly from $\{1, 2, \ldots, m\}$;
- Output x_R.

Convergence for Nonconvex SG

Constant-step variant:

$$\alpha_k \equiv \min\left\{\frac{1}{L}, \frac{\tilde{D}}{M\sqrt{N}}\right\},$$

for some user-defined $\tilde{D} > 0$.

Define D_f so that $D_f^2 = 2(f(x_1) - f^*)/L$ (where f^* is the optimal value of f). Then

$$\frac{1}{L}E[\|\nabla f(x_R)\|^2] \leq \frac{LD_f^2}{N} + \left(\tilde{D} + \frac{D_f^2}{\tilde{D}}\right)\frac{M}{\sqrt{N}}.$$

In the case of convex f, get a similar bound:

$$E[f(x_R) - f^*] \leq \frac{LD_X^2}{N} + \left(\tilde{D} + \frac{D_X^2}{\tilde{D}}\right)\frac{M}{\sqrt{N}},$$

where $D_X = \|x_1 - x^*\|$.

Two-Phase Nonconvex SG

Ghadimi and Lan extend this method to achieve complexity like $1/k$, by running multiple processes in parallel, and picking the solution with the best sample gradient. Prove a large-deviation convergence bound: user specifies required threshold ϵ for $\|\nabla f(x)\|^2$, and required likelihood $1 - \Lambda$ (for small positive Λ).

- Choose trial set $\{j_1, j_2, \ldots, j_T\}$ of indices from $\{1, 2, \ldots, m\}$.
- Set $S = \log(2/\Lambda)$;
- For $s = 1, 2, \ldots, S$, run SG with random number of iterates, to produce $\bar{x}_s$;
- For $s = 1, 2, \ldots, S$, calculate

$$g_s = \frac{1}{T} \sum_{k=1}^{T} \nabla f_{j_k}(\bar{x}_s).$$

- Output $\bar{x}_s$ for which $\|g_s\|$ is minimized.

Sparse Optimization: Motivation

Many applications need structured, approximate solutions of optimization formulations, rather than exact solutions.

- More Useful, More Credible
  - Structured solutions are easier to understand.
  - They correspond better to prior knowledge about the solution.
  - They may be easier to use and actuate.
  - Extract just the *essential* meaning from the data set, not the less important effects.
- Less Data Needed
  - Structured solution lies in lower-dimensional spaces $\Rightarrow$ need to gather / sample less data to capture it.

The structural requirements have deep implications for how we formulate and solve these problems.

Regularized Formulations: Shrinking Algorithms

Consider the formulation

$$\min_x \ f(x) + \tau\psi(x).$$

Regularizer ψ is often nonsmooth but "simple." Often the problem is easy to solve when f is replaced by a quadratic with diagonal Hessian:

$$\min_z \ g^T(z-x) + \frac{1}{2\alpha}\|z-x\|_2^2 + \tau\psi(z).$$

Equivalently,

$$\min_z \ \frac{1}{2\alpha}\|z-(x-\alpha g)\|_2^2 + \tau\psi(z).$$

Define the shrink operator as the arg min:

$$S_\tau(y,\alpha) := \arg\min_z \ \frac{1}{2\alpha}\|z-y\|_2^2 + \tau\psi(z).$$

Interesting Regularizers and their Shrinks

Cases for which the subproblem is simple:

- $\psi(z) = \|z\|_1$. Thus $S_\tau(y, \alpha) = \text{sign}(y)\max(|y| - \alpha\tau, 0)$. When y complex, have

$$S_\tau(y, \alpha) = \frac{\max(|y| - \tau\alpha, 0)}{\max(|y| - \tau\alpha, 0) + \tau\alpha} y.$$

- $\psi(z) = \sum_{g \in G} \|z_{[g]}\|_2$, where $z_{[g]}$, $g \in G$ are non-overlapping subvectors of z. Here

$$S_\tau(y, \alpha)_{[g]} = \frac{\max(|y_{[g]}| - \tau\alpha, 0)}{\max(|y_{[g]}| - \tau\alpha, 0) + \tau\alpha} y_{[g]}.$$

- $\psi(x) = I_\Omega(x)$: Indicator function for a closed convex set Ω. Then $S_\tau(y, \alpha)$ is the projection of y onto Ω.

Basic Prox-Linear Algorithm

(Fukushima and Mine, 1981) for solving $\min_x f(x) + \tau\psi(x)$.

0: Choose x_0

k: Choose $\alpha_k > 0$ and set

$$\begin{aligned} x_{k+1} &= S_\tau(x_k - \alpha_k \nabla f(x_k); \alpha_k) \\ &= \arg\min_z \nabla f(x_k)^T (z - x_k) + \frac{1}{2\alpha_k}\|z - x_k\|_2^2 + \tau\psi(z). \end{aligned}$$

This approach goes by many names, including forward-backward splitting, shrinking / thresholding.

Straightforward, but can be fast when the regularization is strong (i.e. solution is "highly constrained").

Can show convergence for steps $\alpha_k \in (0, 2/L)$, where L is the bound on $\nabla^2 f$. (Like a short-step gradient method.)

Enhancements

Alternatively, since α_k plays the role of a steplength, can adjust it to get better performance *and* guaranteed convergence.

- Backtracking: decrease α_k until sufficient decrease condition holds.
- Use Barzilai-Borwein strategies to get nonmonotonic methods. By enforcing sufficient decrease every 10 iterations (say), still get global convergence.
- Embed accelerated first-order methods (e.g. FISTA) by redefining the linear term g appropriately.
- Do continuation on the parameter τ. Solve first for large τ (easier problem) then decrease τ in steps toward a target, using the previous solution as a warm start for the next value.

Decomposition / Coordinate Relaxation

For min $f(x)$, at iteration k, choose a subset $\mathcal{G}_k \subset \{1, 2, \ldots, n\}$ and take a step d_k only in these components. i.e. fix $[d_k]_i = 0$ for $i \notin \mathcal{G}_k$.

Fewer variables $\Rightarrow$ smaller and cheaper to deal with than the full problem. Can

- take a reduced gradient step in the $\mathcal{G}_k$ components;
- take multiple "inner iterations"
- actually solve the reduced subproblem in the space defined by $\mathcal{G}_k$.

Example: Decomposition in SVM

Decomposition has long been popular for solving the dual (QP) formulation of support vector machines (SVM), since the number of variables (= number of training examples) may be very large.

SMO, LIBSVM: Each $\mathcal{G}_k$ has two components, different heuristics for choosing $\mathcal{G}_k$.

LASVM: Again $|\mathcal{G}_k| = 2$, with focus on online setting.

SVM-light: Small $|\mathcal{G}_k|$ (default 10).

GPDT: Larger $|\mathcal{G}_k|$ (default 400) with gradient projection solver as inner loop.

Deterministic Results

Generalized Gauss-Seidel condition requires only that each coordinate is "touched" at least once every T iterations:

$$\mathcal{G}_k \cup \mathcal{G}_{k+1} \cup \ldots \cup \mathcal{G}_{k+T-1} = \{1, 2, \ldots, n\}.$$

Can show global convergence (e.g. Tseng and Yun, 2009; Wright, 2012).

There are also results on

- global linear convergence rates;
- optimal manifold identification (for problems with nonsmooth regularizer);
- fast local convergence for an algorithm that takes reduced steps on the estimated optimal manifold.

Stochastic Coordinate Descent

Randomized coordinate descent (RCDC)

(Richtarik and Takac, 2012; see also Nesterov, 2012)

For min $f(x)$.

- Partitions the components of x into subvectors;
- Makes a random selection of one partition to update at each iteration;
- Exploits knowledge of the partial Lipschitz constant for each partition in choosing the step.
- Allows parallel implementation.

At each iteration, pick a partition at random, and take a short-step steepest descent step on the variables in that component.

RCDC Details

Partition components $\{1, 2, \ldots, n\}$ into m blocks with block $[i]$ with corresponding columns from the $n \times n$ identity matrix denoted by U_i.

Denote by L_i the partial Lipschitz constant on partition $[i]$:

$$\|\nabla_{[i]} f(x + U_i t) - \nabla_{[i]} f(x)\| \leq L_i \|t\|.$$

Fix probabilities of choosing each partition: p_i, $i = 1, 2, \ldots, m$.

Iteration k:

- Choose partition $i_k \in \{1, 2, \ldots, m\}$ with probability p_i;
- Set $d_{k,i} = -(1/L_{i_k}) \nabla_{[i_k]} f(x_k)$;
- Set $x_{k+1} = x_k + U_{i_k} d_{k,i}$.

For convex f and uniform weights $p_i = 1/m$, can prove high probability convergence of f to within a specified threshold of $f(x^*)$ in $O(1/k)$ iterations.

RCDC Result

$$\|x\|_L := \left(\sum_{i=1}^{m} L_i \|x_{[i]}\|^2 \right)^{1/2}$$

Weighted measure of level set size:

$$\mathcal{R}_L(x) := \max_y \max_{x^* \in X^*} \{ \|y - x^*\|_L \, : \, f(y) \leq f(x) \}.$$

Assuming that desired precision ϵ has

$$\epsilon < \min(\mathcal{R}_L^2(x_0), f(x_0) - f^*)$$

and defining

$$K := \frac{2n\mathcal{R}_L^2(x_0)}{\epsilon} \log \frac{f(x_0) - f^*}{\epsilon\rho},$$

we have for $k \geq K$ that

$$P(f(x_k) - f^* \leq \epsilon) \geq 1 - \rho.$$

Augmented Lagrangian Methods and Splitting

Consider linearly constrained problem:

$$\min f(x) \text{ s.t. } Ax = b.$$

Augmented Lagrangian is

$$\mathcal{L}(x, \lambda; \rho) := f(x) + \lambda^T (Ax - b) + \frac{\rho}{2} \|Ax - b\|_2^2,$$

where $\rho > 0$. Basic augmented Lagrangian / method of multipliers is

$$x_k = \arg\min_x \mathcal{L}(x, \lambda_{k-1}; \rho_k);$$
$$\lambda_k = \lambda_{k-1} + \rho_k (Ax_k - b);$$
$$(\text{choose } \rho_{k+1}).$$

Extends to inequality constraints, nonlinear constraints.

Features of Augmented Lagrangian

- If we set $\lambda = 0$, recover the quadratic penalty method.
- If $\lambda = \lambda^*$, the minimizer of $\mathcal{L}(x, \lambda^*; \rho)$ is the true solution x^*, for any $\rho \geq 0$.
- AL provides successive estimates λ_k of λ^*, thus can achieve solutions with good accuracy without driving ρ_k to ∞.

History of Augmented Lagrangian

- Dates from 1969: Hestenes, Powell.
- Developments in 1970s, early 1980s by Rockafellar, Bertsekas,....
- Lancelot code for nonlinear programming: Conn, Gould, Toint, around 1990.
- Largely lost favor as an approach for general nonlinear programming during the next 15 years.
- Recent revival in the context of sparse optimization and associated domain areas, in conjunction with splitting / coordinate descent techniques.

Separable Objectives: ADMM

Alternating Directions Method of Multipliers (ADMM) arises when the objective in the basic linearly constrained problem is separable:

$$\min_{(x,z)} f(x) + h(z) \text{ subject to } Ax + Bz = c,$$

for which

$$\mathcal{L}(x, z, \lambda; \rho) := f(x) + h(z) + \lambda^T(Ax + Bz - c) + \frac{\rho}{2}\|Ax - Bz - c\|_2^2.$$

Standard augmented Lagrangian would minimize $\mathcal{L}(x, z, \lambda; \rho)$ over (x, z) jointly — but these are coupled through the quadratic term, so the advantage of separability is lost.

Instead, minimize over x and z separately and sequentially:

$$x_k = \arg\min_x \mathcal{L}(x, z_{k-1}, \lambda_{k-1}; \rho_k);$$

$$z_k = \arg\min_z \mathcal{L}(x_k, z, \lambda_{k-1}; \rho_k);$$

$$\lambda_k = \lambda_{k-1} + \rho_k(Ax_k + Bz_k - c).$$

ADMM

- Each iteration does a round of block-coordinate descent in (x, z).
- The minimizations over x and z add only a quadratic term to f and h, respectively. This does not alter the cost much.
- Can perform these minimizations inexactly.
- Convergence is often slow, but sufficient for many applications.
- Many recent applications to compressed sensing, image processing, matrix completion, sparse PCA.

ADMM has a rich collection of antecendents. For an excellent recent survey, including a diverse collection of machine learning applications, see (Boyd et al, 2011).

ADMM for Consensus Optimization

Given

$$\min \sum_{i=1}^{m} f_i(x),$$

form m copies of the x, with the original x as a "master" variable:

$$\min_{x,x^1,x^2,\ldots,x^m} \sum_{i=1}^{m} f_i(x^i) \text{ subject to } x^i = x,\ i = 1, 2, \ldots, m.$$

Apply ADMM, with $z = (x^1, x^2, \ldots, x^m)$, get

$$x_k^i = \arg\min_{x^i} f_i(x^i) + (\lambda_{k-1}^i)^T (x^i - x_{k-1}) + \frac{\rho_k}{2} \|x^i - x_{k-1}\|_2^2,\ \forall_i,$$

$$x_k = \frac{1}{m} \sum_{i=1}^{m} \left(x_k^i + \frac{1}{\rho_k} \lambda_{k-1}^i \right),$$

$$\lambda_k^i = \lambda_{k-1}^i + \rho_k (x_k^i - x_k),\ \forall_i$$

Can do the x^i updates in parallel. Synchronize for x update.

ADMM for Awkward Intersections

The feasible set is sometimes an intersection of two or more convex sets that are easy to handle separately (e.g. projections are easily computable), but whose intersection is more difficult to work with.

Example: Optimization over the cone of doubly nonnegative matrices:

$$\min_X f(X) \;\text{ s.t. }\; X \succeq 0, \;\; X \geq 0.$$

General form:

$$\min f(x) \;\text{ s.t. }\; x \in \Omega_i, \;\; i = 1, 2, \ldots, m$$

Just consensus optimization, with indicator functions for the sets.

$$x_k = \arg\min_x \; f(x) + \sum_{i=1}^{m} (\lambda_{k-1}^i)^T (x - x_{k-1}^i) + \frac{\rho_k}{2} \|x - x_{k-1}^i\|_2^2,$$

$$x_k^i = \arg\min_{x_i \in \Omega_i} \; (\lambda_{k-1}^i)^T (x_k - x^i) + \frac{\rho_k}{2} \|x_k - x^i\|_2^2, \;\; \forall_i$$

$$\lambda_k^i = \lambda_{k-1}^i + \rho_k (x_k - x_k^i), \;\; \forall_i.$$

Splitting+Lagrangian+Hogwild!

Recalling Hogwild! for minimizing

$$f(x) = \sum_{i=1}^{m} f_i(x),$$

using parallel SG, where each core evaluates a random partial gradient $\nabla f_i(x)$ and updates a centrally stored x. Results were given showing good speedups on a 12-core machine.

The machine has two sockets, so need frequent cross-socket cache fetches.

Ré et al (2012) use the same 12-core, 2-socket platform, but

- each socket has its own copy of x, each updated by half the cores;
- The copies "synchronize" periodically by consensus, or by updating a common multiplier λ.

Details

Replace $\min f(x)$ by the split problem

$$\min_{x,z} f(z) + f(z) \text{ s.t. } x = z.$$

Min-max formulation is

$$\min_{x,z} \max_{\lambda} f(x) + f(z) + \lambda^T (x - z).$$

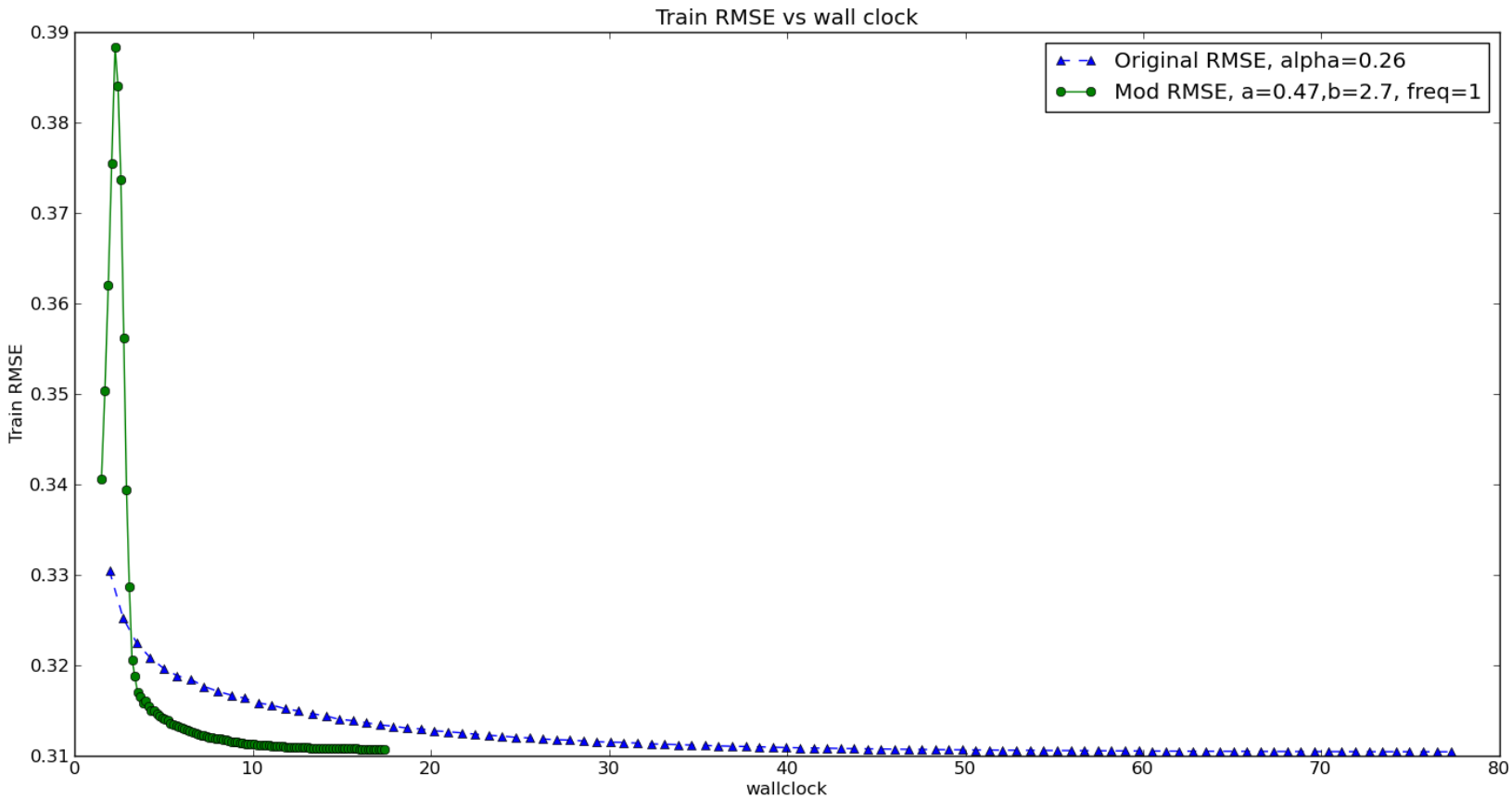
Store x on one socket and z on the other. Algorithm step k:

- Apply Hogwild independently on the two sockets, for a certain amount of time, to

$$\min_x f(x) + \lambda_k^T x, \quad \min_z f(z) - \lambda_k^T z,$$

  To obtain x_k and z_k.
- Swap values x_k and z_k between sockets and update

$$\lambda_{k+1} := \lambda_k + \gamma(x_k - z_k), \text{ for some } \gamma > 0.$$



Over 100 epochs for the SVM problem RCV1, without permutation, speedup factor is 4.6 (.77s vs .17s).

Partially Separable: Higher-Order Methods

Consider again

$$f(x) = \frac{1}{m} \sum_{i=1}^{m} f_i(x)$$

Newton's method obtains step by solving

$$\nabla^2 f(x_k) d_k = -\nabla f(x_k),$$

and setting $x_{k+1} = x_k + d_k$. We have

$$\nabla^2 f(x) = \frac{1}{m} \sum_{i=1}^{m} \nabla^2 f_i(x), \quad \nabla f(x) = \frac{1}{m} \sum_{i=1}^{m} \nabla f_i(x).$$

The "obvious" implementation requires scan through the complete data set, and formation and solution of an $n \times n$ linear system. This is usually impractical for large m, n.

Sampling

(Martens, 2010; Byrd et al, 2011)

Approximate the derivatives by sampling over a subset of the m terms.

Choose $X \subset \{1, 2, \ldots, m\}$ and $S \subset X$, and set

$$H_S^k = \frac{1}{|S|} \sum_{i \in S} \nabla^2 f_i(x_k), \quad g_X^k = \frac{1}{|X|} \sum_{i \in X} \nabla f_i(x_k).$$

Approximate Newton step satisfies

$$H_S^k d_k = g_X^k.$$

Typically, S is just 1% to 10% of the terms.

Conjugate Gradient and "Hessian-Free" Implementation

We can use conjugate gradient (CG) to find an *approximate* solution. Each iteration of CG requires one matrix-vector multiplication with H_S^k, plus some vector operations ($O(n)$ cost).

Hessian-vector product

$$H_S^k v = \frac{1}{|S|} \sum_{i \in S} \nabla^2 f_i(x_k) v.$$

Since $\nabla^2 f_i$ often is sparse, each product $\nabla^2 f_i(x_k)v$ is cheap.

We can even avoid second-derivative calculations altogether, by using finite-difference approximation to the matrix-vector product:

$$\nabla^2 f_i(x_k)v \approx [\nabla f_i(x_k + \epsilon v) - \nabla f_i(x_k)]/\epsilon.$$

Sampling and coordinate descent can be combined: see Wright (2012) for application of this combination to regularized logistic regression.

Conclusions

I've given an incomplete survey of optimization techniques that may be useful in analysis of large datasets, particularly for learning problems.

Some important topics omitted, e.g.

- quasi-Newton methods (L-BFGS),
- nonconvex regularizers,
- manifold identification / variable selection.

For further information, see slides, videos, and reading lists for tutorials that I have presented recently, on some of these topics. Or mail me.

FIN