Optimization in Learning and Data Analysis

Stephen Wright

University of Wisconsin-Madison

August 2013

Big Picture

- Optimization provides a powerful toolbox for solving data analysis and learning problems.
- The particular requirements of data analysis problems are driving new research in optimization — much of it being done by machine learning researchers.
- Some old lines of optimization research are suddenly new again!
- Interesting intersections with systems — multicore and clusters.

Outline

I. Sketch some canonical formulations of data analysis / machine learning problems as optimization problems.

II. An optimization toolbox: Fundamental formulation and algorithmic techniques from optimization that are featuring strongly in data analysis.

II+I. Show how the optimization tools are mixed and matched to address data analysis tasks.

III. Illustrating new work at the intersection of optimization, systems, and big data: asynchronous multicore algorithms.

- Stochastic gradient (HOGWILD!);
- Coordinate descent;
- Application to extreme linear programming.

I. Canonical Formulations

- Linear regression
- + variable selection (LASSO)
- Compressed sensing
- Support vector machines
- Logistic regression
- Matrix completion
- Inverse covariance estimation
- Deep belief networks
- Image processing
- Data assimilation.

Linear Regression

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

Least Squares: Under certain assumptions on measurement error / noise, can find a suitable x by solving a least squares problem

$$\min_x \frac{1}{2}\|Ax - b\|_2^2 = \frac{1}{2}\sum_{i=1}^m (a_i^T x - b_i)^2$$

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

Robust Regression: Can replace the sum-of-squares with loss functions that are less sensitive to outliers. Objectives are still separable, one term per data element.

$$\ell_1: \quad \min_x \|Ax - b\|_1 = \sum_{i=1}^m |a_i^T x - b_i|,$$

$$\text{Huber:} \quad \min_x \sum_{i=1}^m h(a_i^T x - b_i), \quad (h \text{ is Huber loss}).$$

Feature Selection: LASSO

Can modify least-squares for feature selection by adding a LASSO regularizer (Tibshirani, 1996):

$$\textbf{LASSO:} \quad \min_x \frac{1}{2}\|Ax - b\|_2^2 + \lambda \|x\|_1,$$

for some parameter $\lambda > 0$. This identifies an approximate minimizer of the least-squares loss with few nonzeros (sparse).

Can state equivalently in constrained form:

$$\min_x \frac{1}{2}\|Ax - b\|_2^2 \text{ subject to } \|x\|_1 \le T,$$

for parameter $T > 0$.

Compressed Sensing

The ℓ_2-ℓ_1 formulation of compressed sensing is identical to LASSO:

$$\min_x \frac{1}{2}\|Ax - b\|_2^2 + \lambda\|x\|_1,$$

but dimensions and properties of A are typically different.

- There is an approximate solution to $Ax \approx b$ that is known to be sparse. (x could be coefficients of a signal in some basis.)
- A is $m \times n$ with $m \ll n$, with narrow column submatrices well conditioned (*restricted isometry* or *incoherence*). Usually random.
- b is a vector of observations.

Can recover the sparse x^* from this convex formulation, despite undersampling x. Number of observations m needs to be only a modest multiple of the number of nonzeros in x. (Candès et al., 2006)

Support Vector Classification

Given data vectors $x_i \in \mathbb{R}^n$, for $i = 1, 2, \dots, m$ and labels $y_i = \pm 1$ to indicate the class to which x_i belongs.

Seek z such that (usually) we have

$$x_i^T z \geq 1 \text{ when } y_i = +1 \text{ and } x_i^T z \leq -1 \text{ when } y_i = -1.$$

SVM with hinge loss to penalize misclassifications. Objective is separable (as in regession):

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

where $C > 0$ is a parameter. Define $K_{ij} = y_i y_j x_i^T x_j$ for dual:

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

Extends to nonlinear kernel: $K_{ij} := y_i y_j k(x_i, x_j)$ for kernel function $k(\cdot, \cdot)$.

(Boser et al., 1992; Vapnik, 1999)

(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

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

Add regularizer $\lambda \|z\|_1$ to select features.

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.

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

Matrix Completion

Seek a matrix $X \in \mathbb{R}^{m \times n}$ with some desired structure (e.g. low rank) that matches certain observations, possibly noisy.

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

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

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

Can impose other structures, e.g. X is the sum of sparse matrix and a low-rank matrix. (Element-wise 1-norm $\|X\|_1$ is useful for sparsity.)

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

(Recht et al., 2010)

Inverse Covariance Estimation

Given m samples $y_1, y_2, \ldots, y_m$ of a Gaussian random variable $Y \sim \mathcal{N}(\mu; C)$, the log-likelihood of the inverse covariance P is

$$L(P) = \log p(y_1, ..., y_n | P) = \log\det(P) - \langle S, P \rangle + \text{constant}$$

where $S = \frac{1}{m}\sum_{i=1}^m (y_i - \mu)(y_i - \mu)^T$ is the sample covariance.

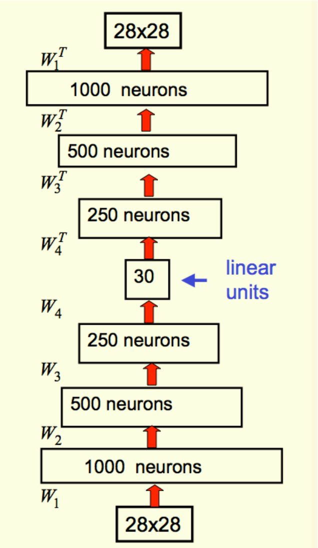
Zeros in P reveal conditional independencies between components of Y:

$$P_{ij} = 0 \quad \Leftrightarrow \quad Y_i \perp\!\!\!\perp Y_j | \{Y_k,\ k \neq i, j\}$$

Sparsity in P is encouraged by adding an element-wise ℓ_1 regularizer:

$$\min_{P \succ 0} -\log\det(P) + \langle S, P \rangle + \tau \|P\|_1.$$

Sparsity reveals only the important dependencies i.e. structure of the network. (Friedman et al., 2008)



Deep Belief Nets / Neural Nets transform feature vectors prior to classification.

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. The unknowns are parameters of the matrices W_1, W_2, W_3, W_4; output is nonlinear in these parameters.

Deep Belief Networks

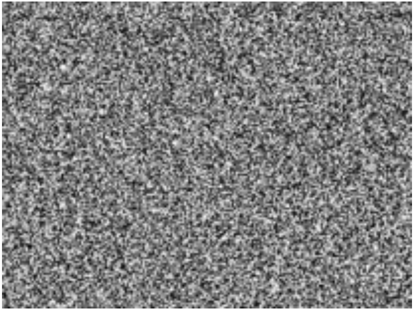
Output of a DBN can form the input to a classifier (e.g. SVM, or something simpler, like a max of the output features).

Objectives in learning problems based on DBNs are

- separable: objective is composed of terms that each depend on one item of data (e.g. one utterance, one character, one image) and possibly its neighbors in space or time.
- nonlinear, nonconvex: each layer is simple (linear transformation, sigmoid, softmax), but their composition is not.
- possibly regularized with terms that impose structure. e.g. phoneme class depends on sounds that came before and after.

Image Processing

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



Denoising / Deblurring: Given an image with noise or blur, seek a "nearby natural image."

Total Variation Regularization

Apply an ℓ_1 penalty to spatial gradients in the 2D image, defined by

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

Given a noisy image $f : \Omega \to \mathbb{R}$, solve for u: (Rudin et al., 1992)

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

Data Assimilation

There are thriving communities in computational science that study PDE-constrained optimization and in particular data assimilation. The latter is the basis of weather forecasting.

These are based on parametrized partial differential equation models, whose parameters are determined from

- data (huge, heterogeneous): observations of the system state at different points in space and time;
- statistical models of noise, in both the PDE model and observations;
- prior knowledge about the solution, such as a guess of the optimal value and an estimate of its reliability.

Needs models (meteorology and oceanography), statistics, optimization, scientific computing, physics, applied math,...

There is active research on better noise models (better covariances).

II. Optimization Formulations: Typical Properties

- Data, from which we want to extract key information, make inferences about future / missing data, or guide decisions.
- Parametrized model that captures the relationship between the data and the meaning we are trying to extract.
- Objective that measures the mismatch between current model / parameters and observed data; also deviation from prior knowledge or desired structure.

Specific typical properties of learning problems are

- Big data;
- Often, need only low-medium accuracy;
- Have some prior knowledge about the model parameters;
- Have some desired structure for the parameters. Regularization.

In some cases, the optimization formulation is well settled: See above.

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

Optimization Toolbox

A selection of fundamental optimization techniques that feature strongly in the applications above.

Most have a long history, but the slew of interesting new applications and contexts has led to new twists and better understanding.

- Accelerated Gradient (and its cousins)
- Stochastic Gradient
- Coordinate Descent
- Shrinking techniques for regularized formulations
- Higher-order methods
- Augmented Lagrangians, Splitting, ADMM.

Describe each briefly, then show how they are deployed to solve the applications in Part I.

Gradient Methods: Steepest Descent

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

Compare these methods on the smooth convex case:

$$\mu I \preceq \nabla^2 f(x) \preceq LI \ \text{ for all } x \qquad (0 \le \mu \le L).$$

Steepest Descent sets

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

When $\mu > 0$, set $\alpha_k \equiv 2/(\mu + L)$ to get linear convergence, at rate depending on conditioning $\kappa := L/\mu$:

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

Need $O(\kappa \log \epsilon)$ iterations to reduce the error by a factor ϵ.

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

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 $O(\sqrt{\kappa} \log \epsilon)$ to achieve precision of ϵ, vs. about $O(\kappa \log \epsilon)$ for steepest descent. (Polyak, 1987)

For $\kappa = 100$, heavy-ball is 10 times faster than steepest descent.

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.

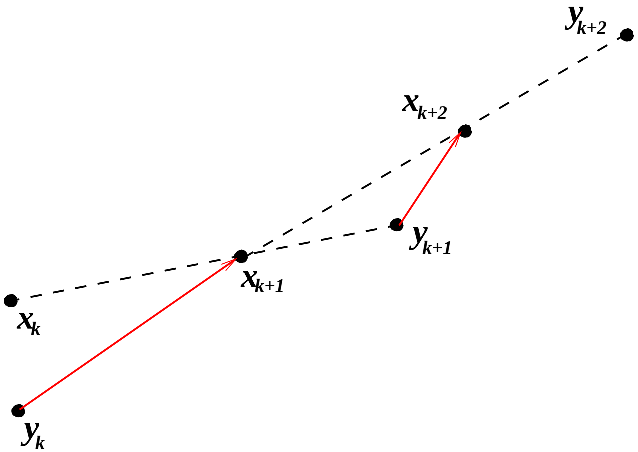
One of Nesterov's methods (Nesterov, 1983, 2004) is:

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" contribution from "momentum" contribution, producing two sequences $\{x_k\}$ and $\{y_k\}$.
- Still works for weakly convex ($\kappa = \infty$).
- FISTA (Beck and Teboulle, 2009) is similar.

Extends easily to problems with convex constraints, regularization, etc.


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

Stochastic Gradient

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.

Consider the finite sum:

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

where each f_i is convex and m is huge. Often, each f_i is a loss function associated with ith data item (SVM, regression, ...), or a mini-batch.

Classical SG: 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 $\alpha_k > 0$. (Robbins and Munro, 1951)

Classical SG

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

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

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:

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

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

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

SG Variants

Constant Stepsize. Set a target ϵ for a_k, and figure out how to choose constant step $\alpha_k \equiv \alpha$ and number of iterations K to hit this target.

Robust SG: Primal Averaging. Choose $\alpha_k = \theta/(M\sqrt{k})$ (for some parameter θ), generate x_k as above, and form a weighted average of all iterates:

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

Convergence can be slower, but more stable than the classical approach. Also works for $\mu = 0$, and less sensitive to estimates of parameters.

Dual Averaging. Take a step based on the average of all gradient estimates seen so far. Again, more stable behavior.

SG is not a descent method, so please don't call it SGD!

Coordinate Descent

Again consider unconstrained minimization for smooth $f : \mathbb{R}^n \to \mathbb{R}$:

$$\min_{x \in \mathbb{R}^n} f(x).$$

Iteration k of coordinate descent (CD) picks one index i_k and takes a step in the i_k component of x to decrease f.

It may be proportional to the gradient w.r.t. x_{i_k}:

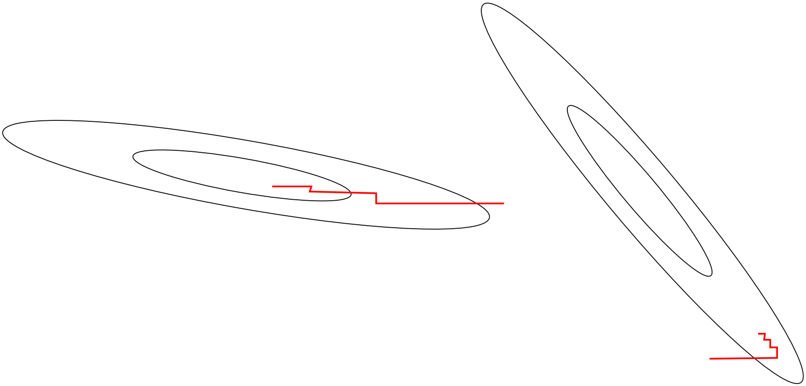
$$x_{k+1,i} = x_{k,i} - \alpha[\nabla f(x_k)]_{i_k},$$

or we may actually minimize f along the i_k direction.

- Deterministic CD: choose i_k in some fixed order e.g. cyclic;
- Stochastic CD: choose i_k at random from $\{1, 2, \ldots, n\}$.

CD is a reasonable choice when it's cheap to evaluate individual elements of $\nabla f(x)$ (at $1/n$ of the cost of a full gradient, say).

Coordinate Descent Illustrated



Coordinate Descent: Extensions and Convergence

Block variants of CD choose a subset $[k] \subset \{1, 2, \ldots, n\}$ of components at iteration k, and take a step in those.

Can also apply coordinate descent when there are bounds on components of x. Or, more generally, constraints that are separable with respect to the blocks in a block CD method.

Similar extensions to separable regularization functions (see below).

Convergence: Deterministic (Luo and Tseng, 1992; Tseng, 2001), linear rate (Beck and Tetruashvili, 2013). Stochastic, linear rate: (Nesterov, 2012).

Regularization

Often have prior knowledge of structure in the solution.

- Most famously: sparsity of the unknown vector x (few nonzeros).
- Low-rank and/or sparsity of an unknown matrix X.
- "Naturalness" of an image vector u_{ij}, for $i, j = 1, 2, \ldots, N$. (Large areas of constant intensity/color separated by sharp edges.)
- Group sparsity: Nonzeros in x appear in predefined clusters, arising for example as subtrees of a tree.

Enforcing these requirements in the obvious way gives rise to intractable optimization problems.

But often, can add regularization functions to the objective or constraints to obtain convex formulations whose solutions have the desired sparsity.

ℓ_1 and Sparsity

It had been known for some time (since the 1970s?) that adding an ℓ_1 norm in the formulation of $\min_x f(x)$ tended to produce sparse solutions.

$$\min_x f(x) + \lambda \|x\|_1,$$

$$\min_x f(x) \text{ subject to } \|x\|_1 \le T.$$

When f is a convex function, these formulations are also convex.

Candès et al. (2006) showed that when $f(x) = (1/2)\|Ax - b\|_2^2$, the convex ℓ_1 formulation has the same solution x^* as the (intractable) cardinality-constrained formulation. Compressed Sensing.

Requires certain conditions on A (e.g. restricted isometry; see above). The number of observations (rows of A) needed to recover x is related to the number of nonzeros — depends only logarithmically on the full dimension of x.

Other Structures

For other types of structure, the trick becomes to define a regularization function $\psi(x)$ that induces the desired structure.

Many of these ψ are obvious generalizations of ℓ_1. Examples:

- Low rank of matrix X: $\psi(X) = \|X\|_* =$ sum of singular values of X. Nuclear norm.
- Sparse plus low-rank X: Split $X = S + T$ and use regularizer $\|T\|_* + \|S\|_1$, where $\|S\|_1$ is element-wise 1-norm.
- Natural image: Total Variation:

$$\|u\|_{\mathrm{TV}} := \sum_{i=1}^{N-1} \sum_{j=1}^{N-1} \left\| \begin{bmatrix} u_{i+1,j} - u_{i,j} \\ u_{i,j+1} - u_{i,j} \end{bmatrix} \right\|.$$

- Group sparse: Sum of ℓ_2:

$$\psi(x) = \sum_{j=1}^{m} \|x_{[j]}\|_2,$$

where $[j] \subset \{1, 2, \dots, n\}$ are the groups of components.

Shrink Operators

The regularizers ψ are usually simple, convex, nonsmooth, separable.

Shrink operators are useful in extending algorithms from smooth setting

$$\min_x f(x)$$

to the regularized setting

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

In smooth setting, step is

$$x_+ \leftarrow x - \alpha g,$$

which is the solution of

$$x_+ = \arg\min_z \frac{1}{2} \|z - (x - \alpha g)\|_2^2$$

Can extend this to the regularized case by simply adding $\psi(x)$: Shrink!

$$x_+ = \arg\min_z \frac{1}{2} \|z - (x - \alpha g)\|_2^2 + \alpha\lambda\psi(x) = S_{\alpha\lambda}(x - \alpha g).$$

Using Shrinks

For many regularizers ψ, the shrink operation is cheap: $O(n)$ operations. Often, can even solve it it in closed form.

Constraints can also be enforced via shrinks. For

$$\min_{x \in \Omega} f(x) \quad \text{with } \Omega \text{ closed, convex}$$

can define

$$\psi(x) = i_\Omega(x) = \begin{cases} 0 & \text{if } x \in \Omega \\ \infty & \text{otherwise} \end{cases}.$$

The shrink operator becomes a projection onto Ω.

Steepest descent, accelerated gradient, stochastic gradient, higher-order can all be extended to regularized case by replacing the line step with a shrink operation.

Newton's Method

Higher-order methods are founded in Newton's method, which takes the step to be the minimizer of a second-order approximation to a smooth function f:

$$d = \arg\min_d \; f(x) + \nabla f(x)^T d + \frac{1}{2} d^T \nabla^2 f(x) d,$$

and sets $x_+ = x + d$. When $\nabla^2 f(x)$ is positive definite, can compute d as the solution of

$$\nabla^2 f(x) d = -\nabla f(x).$$

The usual drawback is the difficulty of computing $\nabla^2 f(x)$. But there are many variants of Newton for which this is not necessary.

Higher-Order Methods

Modify Newton's method in various ways:

- quasi-Newton (e.g. L-BFGS): use gradient information to maintain an approximation to $\nabla^2 f(x)$;
- inexact Newton: Use a method for iterative linear equations to solve for d. Requires only matrix-vector multiplications with $\nabla^2 f(x)$, which can be approximated by finite differences.
- approximate Newton: compute a cheap approximation to $\nabla^2 f(x)$, perhaps by sampling (Byrd et al., 2011):

$$\nabla^2 f(x) \approx \sum_{i \in \mathcal{S}} \nabla^2 f_i(x)$$

- Take higher-order steps only on a reduced space. Example: for sparse x (ℓ_1 regularization), take a reduced Newton step on the nonzero components of x — once these are identified.

Augmented Lagrangian

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

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

Extends in a straightforward way to inequality and nonlinear constraints.

Dates to 1969: Hestenes, Powell, Rockafellar, Bertsekas, Conn / Gould / Toint.

ADMM

Alternating Directions Method of Multipliers (ADMM) exploits a partitioning of the objective and/or constraints. Given

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

we have Lagrangian

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

Minimize over x and z separately — *not jointly, as standard augmented Lagrangian would require*:

$$\begin{aligned} x_k &= \arg\min_x \mathcal{L}(x, z_{k-1}, \lambda_{k-1}; \rho_k); \\ z_k &= \arg\min_z \mathcal{L}(x_k, z, \lambda_{k-1}; \rho_k); \\ \lambda_k &= \lambda_{k-1} + \rho_k(Ax_k + Bz_k - c). \end{aligned}$$

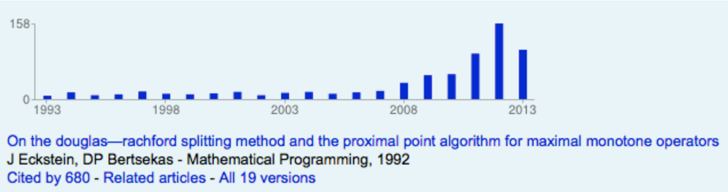
(Useful when minimizations w.r.t x and z are much easier than joint minimization.)

ADMM

Many recent applications to compressed sensing, image processing, matrix completion, sparse principal components analysis, etc.

(Eckstein and Bertsekas, 1992; Boyd et al., 2011)

The surge of interest in ADMM is clear from the citation index for Eckstein and Bertsekas' 1992 paper:



ADMM for Awkward Intersections

$$\min\ f(x)\ \text{ s.t. }\ x \in \Omega_1 \cap \Omega_2 \cap \ldots \cap \Omega_m.$$

Reformulate with master variable x and copies $x_1, x_2, \ldots, x_m$:

$$\min_{x,x^1,x^2,\ldots,x^m}\ f(x)\ \text{ s.t. }\ x^i \in \Omega_i,\ \ x^i - x = 0,\ \ i = 1, 2, \ldots, m.$$

Minimizations over Ω_i can be done independently:

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

Optimize over the master variable (unconstrained)

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

Update multipliers:

$$\lambda_k^i = \lambda_{k-1}^i + \rho_k (x_k - x_k^i),\ \ i = 1, 2, \ldots, m.$$

II+I: Matching Tools to Applications

Return to the applications in Part I and mention how the optimization tools of Part II have been used to solve them. The tools are often combined in different ways.

Linear Regression.

- Linear algebra for $\| \cdot \|_2$. (Traditional!)
- Stochastic gradient for $m \gg n$ (e.g. parallel version described in Raghu's keynote yesterday).

Variable Selection & Compressed Sensing.

- Shrink algorithms (for ℓ_1 term) (Wright et al., 2009).
- Accelerated Gradient (Beck and Teboulle, 2009).
- ADMM (Zhang et al., 2010).
- Higher-order: reduced inexact Newton (Wen et al., 2010); interior-point (Fountoulakis and Gondzio, 2013)
- (Also homotopy in λ, LARS, ...) (Efron et al., 2004)

Support Vector Machines.

- Coordinate Descent (Platt, 1999; Chang and Lin, 2011).
- Stochastic gradient (Bottou and LeCun, 2004; Shalev-Shwartz et al., 2007).
- Higher-order methods (interior-point) (Ferris and Munson, 2002; Fine and Scheinberg, 2001); (on reduced space) (Joachims, 1999).
- Shrink Algorithms (Duchi and Singer, 2009; Xiao, 2010).
- Stochastic gradient + shrink + higher-order (Lee and Wright, 2012).

Logistic Regression (+ Regularization).

- Shrink algorithms + reduced Newton (Shevade and Keerthi, 2003; Shi et al., 2008).
- Newton (Lin et al., 2008; Lee et al., 2006)
- Stochastic gradient (many!)
- Coordinate Descent (Meier et al., 2008)

Matrix Completion.

- (Block) Coordinate Descent (Wen et al., 2012).
- Shrink (Cai et al., 2008; Lee et al., 2010).
- Stochastic Gradient (Lee et al., 2010).

Inverse Covariance.

- Coordinate Descent (Friedman et al., 2008)
- Accelerated Gradient (d'Aspremont et al., 2008)
- ADMM (Goldfarb et al., 2012; Scheinberg and Ma, 2012)

Deep Belief Networks.

- Stochastic Gradient (Le et al., 2012)
- Higher-order (LBFGS, approximate Newton) (Martens, 2010).
- Shrinks
- Coordinate descent (pretraining) (Hinton et al., 2006).

Image Processing.

- Shrink algorithms, gradient projection (Figueiredo and Nowak, 2003; Zhu et al., 2010)
- Higher-order methods: interior-point (Chan et al., 1999), reduced Newton.
- Augmented Lagrangian and ADMM (Bregman) Yin et al. (2008)

Data Assimilation.

- Higher-order methods (L-BFGS, inexact Newton)
- + many other tools from scientific computing.

III. Multicore Asynchronous Methods

Stochastic gradient, coordinate descent: Aren't these **slow, simple, old algorithms?**

- Often good for learning / big-data applications — Can make progress using a small subset of data.
- **Slow?** Often a good fit for modern computers (multicore, NUMA, clusters) — parallel, asynchronous versions are possible.
- **Simple?** What's wrong with that? There's interesting new analysis, tied to plausible models of parallel computation and data.
- **Old?** Yes, but now they are retooled for asynchronous implementation — new computational model and new analysis.

"Asynchronicity is the key to speedup on modern architectures," says Bill Gropp (SIAM CS&E Plenary, 2013).

Parallel Stochastic Gradient

Recall the objective $f(x) = (1/m) \sum f_i(x)$, and basic (serial) SG step:

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

where $i_k \in \{1, 2, \dots, m\}$ is chosen at random. We consider a constant-step variant with $\alpha_k \equiv \alpha$.

Parallel versions tried:

- **Dual Averaging (AIG):** Average gradient estimates evaluated in parallel on different cores. Requires message passing / synchronization (Dekel et al., 2012; Duchi et al., 2010)
- **Round-Robin (RR):** 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 $\nabla f_i(x)$ for some random i, then writes the updates back into x (Niu et al., 2011). **Downpour SGD:** Similar idea for cluster (Le et al., 2012).

Asynchronous Stochastic Gradient: HOGWILD!

HOGWILD!: Each processor runs independently (without synchronization):

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

- Updates can be "old" by the time they are applied, but we assume that they are at most τ cycles old.
- Processors can overwrite each other's work, but sparsity of the ∇f_i helps — updates don't interfere too much with each other.

Define quantities that capture the interconnectedness of the functions f_i:

- $\rho_i =$ number of indices j such that f_i and f_j depend on overlapping components of x.
- $\bar{\rho} = \sum_{i=1}^{m} \rho_i / m^2$: average rate of overlapping subvectors.

HOGWILD! Convergence

Given $\epsilon \in (0, a_0 L)$, and setting

$$\alpha_k \equiv \frac{\mu\epsilon}{(1+2\tau\bar{\rho})LM^2m^2}$$

we have for

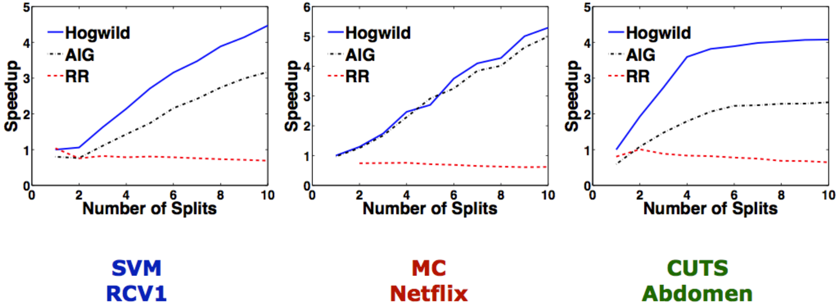
$$k \geq \frac{(1+2\tau\bar{\rho})LM^2m^2}{\mu^2\epsilon} \log\left(\frac{2La_0}{\epsilon} - 1\right)$$

that

$$\min_{0 \leq j \leq k} E(f(x_j) - f(x^*)) \leq \epsilon,$$

Recovers the sublinear $1/k$ convergence rate seen in regular SGD, with the delay τ and overlap measure ρ both appearing linearly.

(Niu et al., 2011; Richtarik, 2012)



HOGWILD! compared with averaged gradient (AIG) and round-robin (RR). Experiments run on a 12-core machine in 2011. (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
MC	Netflix	1.5	2.5E-03	2.3E-03	301	5.3
	KDD	3.9	3.0E-03	1.8E-03	878	5.2
	JUMBO	30	2.6E-07	1.4E-07	9,454	6.8
CUTS	DBLife	0.003	8.6E-03	4.3E-03	230	8.8
	Abdomen	18	9.2E-04	9.2E-04	1,181	4.1

Asynchronous Stochastic Coordinate Descent (ASCD)

Consider $\min f(x)$, where $f : \mathbb{R}^n \to \mathbb{R}^n$ is smooth and convex.

Each processor (independently, without synchronization) does:

1. Choose $i \in \{1, 2, \dots, n\}$ uniformly at random;
2. Read x and evaluate $g = [\nabla f(x)]_i$;
3. Update $x_i \leftarrow x_i - \frac{\gamma}{L_{\max}} g$;

Here γ is a steplength (more below) and $L_{\max}$ is a bound on the diagonals of the Hessian $\nabla^2 f(x)$.

Assume that not more than τ cycles pass between when x is read (step 2) and updated (step 3).

How to choose γ to achieve good convergence?

Constants and "Diagonalicity"

Several constants are critical to the analysis.

- τ: maximum delay;
- $L_{\max}$: maximum diagonal of $\nabla^2 f(x)$;
- L_{res}: maximum row norm of Hessian;
- μ: lower bound on eigenvalues of $\nabla^2 f(x)$ (assumed positive).

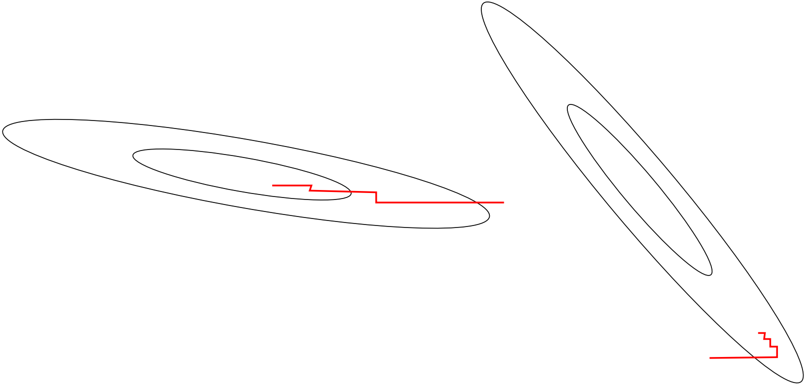
The ratio $L_{\text{res}}/L_{\max}$ is particularly important — it measures the degree of diagonal dominance in the Hessian $\nabla^2 f(x)$ (Diagonalicity).

By convexity, we have

$$1 \le \frac{L_{\text{res}}}{L_{\max}} \le \sqrt{n}.$$

Closer to 1 if Hessian is nearly diagonally dominant (eigenvectors close to principal coordinate axes). Smaller is better for parallelism.

Diagonalicity Illustrated



Left figure is better. It can tolerate a higher delay parameter τ and thus more cores working asynchronously.

How to choose γ?

Choose some $\rho > 1$ and pick γ small enough to ensure that

$$\rho^{-1} \leq \frac{\mathbb{E}(\|\nabla f(x_{j+1})\|^2)}{\mathbb{E}(\|\nabla f(x_j)\|^2)} \leq \rho.$$

Not too much change in gradient over each iteration, so not too much price to pay for using old information, in the asynchronous setting.

Choose γ small enough to satisfy this property but large enough to get a linear rate.

Assuming that

$$\tau + 1 \leq \frac{\sqrt{n} L_{\max}}{2 e L_{\text{res}}},$$

and choosing $\rho = 1 + \frac{2eL_{\text{res}}}{\sqrt{n} L_{\max}}$, we can take $\gamma = 1$. Then have

$$\mathbb{E}(f(x_j) - f^*) \leq \left(1 - \frac{\mu}{2nL_{\max}}\right)^j (f(x_0) - f^*).$$

(Liu and Wright, 2013)

ASCD Discussion

Linear rate is close to the rate attained by short-step steepest descent.

Bound on τ is a measure of potential parallelization. When ratio $L_{\text{res}}/L_{\max}$ is favorable, get $\tau = O(\sqrt{n})$. Thus, expect near-linear speedup on to $O(\sqrt{n})$ cores running asynchronously in parallel.

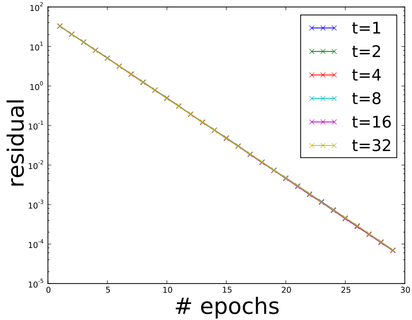
Can extend algorithm and analysis to

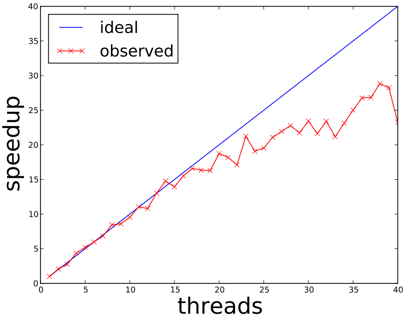
- "Essentially strongly convex" and "weakly convex" cases;
- Separable constraints. Have $\tau = O(n^{1/4})$ — less potential parallelism.

Implemented on 4-socket, 40-core Intel Xeon

$$\min_x \quad \|Ax - b\|^2 + 0.5\|x\|^2$$

where $A \in \mathbb{R}^{m \times n}$ is a Gaussian random matrix ($m = 6000$, $n = 20000$, columns are normalized to 1). $L_{\text{res}}/L_{\max} \approx 2.2$. Choose $\gamma = 1$.





(Thanks: C. Ré, V. Bittorf, S. Sridhar)

Extreme Linear Programming

$$\text{LP}: \qquad \min_x c^T x \ \text{ s.t. } \ Ax = b, \ \ x \geq 0.$$

State-of-the-art solvers for large (LPs) are based on simplex and interior-point methods. An alternative approach based on

- augmented Lagrangian / proximal-point
- iterative solvers for the bounded-QP subproblems (SOR, CG)

were studied in the late 1980s:

- O. L. Mangasarian and R. DeLeone, "Serial and Parallel Solution of Large-Scale Linear Program by Augmented Lagrangian Successive Overrelaxations," 1987.
- S. J. Wright, "Implementing Proximal-Point Methods for Linear Programming," JOTA, 1990

These showed some promise on random, highly degenerate problems, but were terrible on the netlib test set and other problems arising in practice.

But this approach has potential appeal for:

- Cases in which only crude approximate LP solutions are needed.
- No matrix factorizations or multiplications are required. (Thus may be good for special problems, at extreme scale.)
- Multicore implementation is easy, when asynchronous solver is used on the QP subproblems.

Given some estimate $\bar{x}$ of the primal solution and $\bar{u}$ of the dual, get better approximation $x(\beta)$ by solving a convex quadratic program with bounds:

$$x(\beta) := \arg\min_{x \geq 0} \; c^T x - \bar{u}^T (Ax - b) + \frac{\beta}{2} \|Ax - b\|^2 + \frac{1}{2\beta} \|x - \bar{x}\|_2^2,$$

where β is a penalty parameter.

Can update $\bar{x}$ and $\bar{u}$ and repeat. (Augmented Lagrangian.)

(Sridhar et al., 2013)

LP Rounding Approximations

There are numerous NP-hard problems for which approximate solutions can be found using linear programming followed by rounding.

- Construct an integer programming formulation;
- Relax to an LP (replace binary variables by $[0,1]$ intervals);
- Solve the LP approximately;
- Round the LP solution to a feasible integer solution.

Example: **Vertex Cover** Given a graph with edge set E, vertex set V, seek a subset of vertices such that every edge touches the subset. Cost to select a vertex v is c_v. Integer programming form:

$$\min \sum_{v \in V} c_v x_v \;\text{ s.t. }\; x_u + x_v \geq 1 \;\text{ for } (u,v) \in E; \;\; x_v \in \{0,1\} \text{ for all } v \in V.$$

Relax the binary constraint to $x_v \in [0,1]$ to get an LP. Large, but matrix A is highly sparse and structured.

Sample Results

instance	vertex cover		multiway cuts	
	n	nonzeros(A)	n	nonzeros(A)
frb59-26-1	126K	616K	1.3M	3.6M
Amazon	203K	956K	6.8M	21.3 M
DBLP	146K	770K	10.7M	33.7M
Google+	82K	1.5M	7.6M	24.1M
LiveJournal	1.63M	34.6M		

Table: Problem Sizes (after Presolve)

Computation Times (Seconds)

Run on 32 cores Intel machine for max of one hour. Compared with Cplex IP and LP solvers. LP time are for solutions of similar quality. IP solutions are often better. ("$-$" $=$ timeout after 3600s)

instance	type	Cplex IP	Cplex LP	Us
frb59-26-1	VC	-	5.1	0.65
Amazon	VC	44	22	4.7
DBLP	VC	23	21	3.2
Google+	VC	-	62	6.2
LiveJournal	VC	-	-	934
frb59-26-1	MC	54	360	29
Amazon	MC	-	-	131
DBLP	MC	-	-	158
Google+	MC	-	-	570

(Cplex IP sometimes faster than LP because the IP preprocessing can drastically simplify the problem, for some data sets.)

Conclusions

We've discussed

- Canonical problems in data analysis and machine learning
- Some fundamental optimization tools
- Indicated which tools are being used to solve which problems.

Not exhaustive! But enough to show that the optimization toolkit is a vital resource in data analysis and learning.

THANK YOU!

References I

Beck, A. and Teboulle, M. (2009). A fast iterative shrinkage-threshold algorithm for linear inverse problems. *SIAM Journal on Imaging Sciences*, 2(1):183–202.

Beck, A. and Tetruashvili, L. (2013). On the convergence of block coordinate descent methods. Technical report, Technion-Israel Institute of Technology.

Boser, B. E., Guyon, I. M., and Vapnik, V. N. (1992). A training algorithm for optimal margin classifiers. In *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, pages 144–152.

Bottou, L. and LeCun, Y. (2004). Large-scale online learning. In *Advances in Neural Information Processing Systems*.

Boyd, S., Parikh, N., Chu, E., Peleato, B., and Eckstein, J. (2011). Distributed optimization and statistical learning via the alternating direction methods of multipliers. *Foundations and Trends in Machine Learning*, 3(1):1–122.

Byrd, R. H., Chin, G. M., Neveitt, W., and Nocedal, J. (2011). On the use of stochastic hessian information in unconstrained optimization. *SIAM Journal on Optimization*, 21:977–995.

Cai, J.-F., Candès, E., and Shen, Z. (2008). A singular value thresholding algorithm for matrix completion. Technical report, Applied and Computational Mathematics, California Institute of Technology.

Candès, E., Romberg, J., and Tao, T. (2006). Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information. *IEEE Trans. on Inform. Theory*, 52(2):489–509.

References II

Chan, T. F., Golub, G. H., and Mulet, P. (1999). A nonlinear primal-dual method for total variation based image restoration. *SIAM Journal of Scientific Computing*, 20:1964–1977.

Chang, C.-C. and Lin, C. (2011). LIBSVM: a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2.

d'Aspremont, A., Banerjee, O., and El Ghaoui, L. (2008). First-order methods for sparse covariance selection. *SIAM Journal on Matrix Analysis and Applications*, 30:56–66.

Dekel, O., Gilad-Bachrach, R., Shamir, O., and Xiao, L. (2012). Optimal distributed online prediction using mini-batches. *Journal of Machine Learning Research*, 13:165–202.

Duchi, J., Agarwal, A., and Wainwright, M. J. (2010). Distributed dual averaging in networks. In *Advances in Neural Information Processing Systems*.

Duchi, J. and Singer, Y. (2009). Efficient learning using forward-backward splitting. In *Advances in Neural Information Processing Systems*. http://books.nips.cc.

Eckstein, J. and Bertsekas, D. P. (1992). On the Douglas-Rachford splitting method and the proximal point algorithm for maximal monotone operators. *Mathematical Programming*, 55:293–318.

Efron, B., Hastie, T., Johnstone, I., and Tibshirani, R. (2004). Least angle regression. *Annals of Statistics*, 32(2):407–499.

Ferris, M. C. and Munson, T. S. (2002). Interior-point methods for massive support vector machines. *SIAM Journal on Optimization*, 13(3):783–804.

References III

Figueiredo, M. A. T. and Nowak, R. D. (2003). An EM algorithm for wavelet-based image restoration. *IEEE Transactions on Image Processing*, 12:906–916.

Fine, S. and Scheinberg, K. (2001). Efficient svm training using low-rank kernel representations. *Journal of Machine Learning Research*, 2:243–264.

Fountoulakis, K. and Gondzio, J. (2013). A second-order method for strongly convex ell_1-regularization problems. Technical Report ERGO-13-011, University of Edinburgh.

Friedman, J., Hastie, T., and Tibshirani, R. (2008). Sparse inverse covariance estimation with the graphical lasso. *Biostatistics*, 9(3):432–441.

Goldfarb, D., Ma, S., and Scheinberg, K. (2012). Fast alternating linearization methods for minimizing the sum of two convex functions. *Mathematical Programming, Series A*. DOI 10.1007/s10107-012-0530-2.

Hinton, G. E., Osindero, S., and Teh, Y.-W. (2006). A fast learning algorithm for deep belief nets. *Neural Computation*, 18(7):1527–1554.

Joachims, T. (1999). Making large-scale support vector machine learning practical. In Schölkopf, B., Burges, C., and Smola, A., editors, *Advances in Kernel Methods - Support Vector Learning*, chapter 11, pages 169–184. MIT Press, Cambridge, MA.

Langford, J., Li, L., and Zhang, T. (2009). Sparse online learning via truncated gradient. *Journal of Machine Learning Research*, 10:777–801.

References IV

Le, Q. V., Ranzato, M., Monga, R., Devin, M., Chen, K., Corrado, G. S., Dean, J., and Ng, A. Y. (2012). Building high-level features using large scale unsupervised learning. In *Proceedings of the Conference on Learning Theory*. arXiv:1112.6209v5.

Lee, J., Recht, B., Salakhutdinov, R., Srebro, N., and Tropp, J. (2010). Practical large-scale optimization for max-norm regularization. In *Advances in Neural Information Processing Systems*. NIPS.

Lee, S. and Wright, S. J. (2012). Manifold identification in dual averaging methods for regularized stochastic online learning. *Journal of Machine Learning Research*, 13:1705–1744.

Lee, S. I., Lee, H., Abbeel, P., and Ng, A. Y. (2006). Efficient ℓ_1 regularized logistic regerssion. In *Proceedings of the National Conference of the American Association for Artificial Intelligence*.

Lin, C., Weng, R. C., and Keerthi, S. S. (2008). Trust-region newton method for logistic regression. *Journal of Machine Learning Research*, 9:627–650.

Liu, J. and Wright, S. J. (2013). An asynchronous parallel stochastic coordinate descent method. In preparation.

Luo, Z. Q. and Tseng, P. (1992). On the convergence of the coordinate descent method for convex differentiable minimization. *Journal of Optimization Theory and Applications*, 72(1):7–35.

Martens, J. (2010). Deep learning via Hessian-free optimization. In *Proceedings of the 27th International Conference on Machine Learning*.

References V

Meier, L., van de Geer, S., and Buhlmann, P. (2008). The group lasso for logistic regression. *Journal of the Royal Statistical Society B*, 70(1):53–71.

Nesterov, Y. (1983). A method for unconstrained convex problem with the rate of convergence $o(1/k^2)$. *Doklady AN SSSR*, 269:543–547.

Nesterov, Y. (2004). *Introductory Lectures on Convex Optimization: A Basic Course*. Kluwer Academic Publishers.

Nesterov, Y. (2012). Efficiency of coordinate descent methods on huge-scale optimization problems. *SIAM Journal on Optimization*, 22:341–362.

Niu, F., Recht, B., Ré, C., and Wright, S. J. (2011). HOGWILD!: A lock-free approach to parallelizing stochastic gradient descent. In *Advances in Neural Information Processing Systems*.

Platt, J. C. (1999). Fast training of support vector machines using sequential minimal optimization. In Schölkopf, B., Burges, C. J. C., and Smola, A. J., editors, *Advances in Kernel Methods — Support Vector Learning*, pages 185–208, Cambridge, MA. MIT Press.

Polyak, B. T. (1987). *Introduction to Optimization*. Optimization Software.

Recht, B., Fazel, M., and Parrilo, P. (2010). Guaranteed minimum-rank solutions to linear matrix equations via nuclear norm minimization. *SIAM Review*, 52(3):471–501.

Richtarik, P. (2012). HOGWILD! and other hybrid beasts. Notes.

Robbins, H. and Munro, S. (1951). A stochastic approximation method. *Annals of Mathematical Statistics*, 22(3).

References VI

Rudin, L., Osher, S., and Fatemi, E. (1992). Nonlinear total variation based noise removal algorithms. *Physica D*, 60:259–268.

Scheinberg, K. and Ma, S. (2012). Optimization methods for sparse inverse covariance selection. In Sra, S., Nowozin, S., and Wright, S. J., editors, *Optimization for Machine Learning*, Neural Information Processing Series. MIT Press.

Shalev-Shwartz, S., Singer, Y., and Srebro, N. (2007). Pegasos: Primal Estimated sub-GrAdient SOlver for SVM. In *Proceedings of the 24th International Confernece on Machine Learning*.

Shevade, S. K. and Keerthi, S. S. (2003). A simple and efficient algorithm for gene selection using sparse logistic regression. *Bioinformatics*, 19(17):2246–2253.

Shi, W., Wahba, G., Wright, S. J., Lee, K., Klein, R., and Klein, B. (2008). LASSO-Patternsearch algorithm with application to opthalmology data. *Statistics and its Interface*, 1:137–153.

Sridhar, S., Ré, C. M., Wright, S. J., Bittorf, V., and Liu, J. (2013). Large-scale combinatorial optimization using LP rounding. Submitted.

Tibshirani, R. (1996). Regression shrinkage and selection via the LASSO. *Journal of the Royal Statistical Society B*, 58:267–288.

Tseng, P. (2001). Convergence of a block coordinate descent method for nondifferentiable minimization. *Journal of Optimization Theory and Applications*, 109(3):475–494.

Vapnik, V. N. (1999). *The Nature of Statistical Learning Theory*. Statistics for Engineering and Information Science. Springer, second edition.

References VII

Wen, Z., Yin, W., Goldfarb, D., and Zhang, Y. (2010). A fast algorithms for sparse reconstruction based on shrinkage, subspace optimization, and continuation. *SIAM Journal on Scientific Computing*, 32(4):1832–1857.

Wen, Z., Yin, W., and Zhang, Y. (2012). Solving a low-rank factorization model for matrix completion by a nonlinear successive over-relaxation algorithm. *Mathematical Programming Computation*, 4:333–361.

Wright, S. J., Nowak, R. D., and Figueiredo, M. A. T. (2009). Sparse reconstruction by separable approximation. *IEEE Transactions on Signal Processing*, 57:2479–2493.

Xiao, L. (2010). Dual averaging methods for regularized stochastic learning and online optimization. *Journal of Machine Learning Research*, 11:2543–2596.

Yin, W., Osher, S., Goldfarb, D., and Darbon, J. (2008). Bregman iterative algorithms for ℓ_1-minimization with applications to compressed sensing. *SIAM Journal on Imaging Sciences*, 1(1):143–168.

Zhang, Y., Yang, J., and Yin, W. (2010). User's guide for YALL1: Your algorithms for L1 optimization. CAAM Technical Report TR09-17, CAAM Department, Rice University.

Zhu, M., Wright, S. J., and Chan, T. F. (2010). Duality-based algorithms for total variation image restoration. *Computational Optimization and Applications*, 47(3):377–400. DOI: 10.1007/s10589-008-9225-2.