Optimization in Machine Learning

Stephen Wright

University of Wisconsin-Madison

Singapore, 14 Dec 2012
1. Learning from Data: Applications and Models
2. First-Order Methods
3. Stochastic Gradient Methods (Parallel)
4. Identifying Subspaces from Partial Observations
5. Atomic-Norm Regularization
6. Optimization in Deep Learning
Give some examples of data analysis and learning problems and their formulations as optimization problems.

Highlight common features in these formulations.

Survey some optimization techniques that are being applied to these problems. [Give a basic result or two for each.]

Highlight several applications and algorithms from recent work.

Optimization in learning and data analysis is a large and growing area — omit many topics of interest here.
Typical ingredients of optimization formulations of data analysis problems:

- A collection of data, from which we want to extract key information and/or make inferences about future / missing data.
- Parametrized Model of how the data relates to the meaning we are trying to extract.
- Objective that measures quality of information learned: includes model / data discrepancies and deviation from prior knowledge or desirable structure. Model parameters are observed from data via the objective.

Other typical properties of learning problems are huge underlying data set, and need 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!
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 \( a_i^T \), \( i = 1, 2, \ldots, m \).

An approximate sparse solution (few nonzeros in \( x \)) can be found from

**LASSO:** \( \min_x \frac{1}{2} \| Ax - b \|_2^2 \) such that \( \| x \|_1 \leq T \),

for parameter \( T > 0 \). (Smaller \( T \) gives fewer nonzeros in \( x \).)
LASSO and Compressed Sensing

LASSO is equivalent to an \( \ell_2-\ell_1 \) formulation:

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

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 \( x \).

![Graph](image.png)
Group LASSO

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

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

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

- Easy to handle when groups $[g]$ are disjoint.
- Use $\| \cdot \|_2$ or $\| \cdot \|_\infty$. (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 (Bach, Mairal, et al.)
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 $\tau$.

SparseNet (Mazumder, Friedman, Hastie, 2011): coordinate descent.
Linear Support Vector Classification
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

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

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: $\ell_1$ penalty, robust, other loss functions,...)
(Regularized) Logistic Regression

Seek *odds* of class membership rather than a bald prediction. Binary, linear case: linear function with unknown weights $z \in \mathbb{R}^n$:

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

Seek $z$ 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

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

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

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

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

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

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

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

Can have $\psi$ as the nuclear norm $= \text{sum of singular values}$. This regularizer 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.
Example of a deep belief network for autoencoding (Hinton, 2007). Output (at top) depends on input (at bottom) of an image with $28 \times 28$ 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.
Deep Learning in Speech Processing

(Sainath et al, 2012). Break a stream of audio data into phonemes and aim to learn how to identify them from a labelled sample. May use context (phonemes before and after).

Every second layer has \( \approx 10^3 \) inputs and outputs; the parameter is the transformation matrix from input to output (\( \approx 10^6 \) parameters).
Deep learning networks typically consist of layers mapping inputs (form below) to outputs (to above).

Feature vectors are the inputs at the bottom layer. Classification is applied to outputs from the top layer (e.g. SVM, multiclass logistic regression).

Individual layers are simple. Common examples:

- Input vector $x \in \mathbb{R}^n$ maps to output vector $y \in \mathbb{R}^m$ as $y = Wx$. The $mn$ entries of $W$ are the parameters.
- Parametrized sigmoid mapping single input to single output: $y_i = 1/(1 + e^{-\beta_i x_i})$.
- Softmax: $y_i = e^{x_i} / \sum_{j=1}^{n} e^{x_j}$.
- Total-variation (spatial gradient) terms to emphasize edges in image processing

When composed, the result is a complicated nonconvex mapping.
Neural networks were popular and well studied in the early days of machine learning. The equivalence between “back-propagation” and steepest descent was recognized early and studied by several optimization researchers (Mangasarian, Tseng, Luo,...)

Structure of deep networks is motivated by arrangement of cortical neurons in the brain.

They lost popularity because of the difficulty of “training”. Numerous tricks discovered since about 2007 have led to new success, and there has been success in new big-data applications, in speech and image processing. Two *New York Times* front-page stories in recent months.

Optimization techniques are front and center. More later...
Formulations may include regularization functions to induce structure:

$$\min_x f(x) + \tau \psi(x) \quad \text{OR} \quad \min_x f(x) \ \text{s.t.} \ \phi(x) \leq M,$$

where $\psi$ induces the desired structure in $x$. $\psi$ often 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”).
Properties of $f$ in Data Analysis Problems

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 $\nabla f$ may be available at proportionately lower cost than the full gradient.

- These two properties are often combined.
** First-Order Methods

Fundamental first-order methods are the basis of many algorithms. They can typically be extended for

- nonsmooth-regularized \( f \);
- simple constraints \( x \in \Omega \);
- availability of just a random approximation to \( \nabla f \).

Machine learning people love theorems about convergence behavior on convex problems:

- linear / geometric.
- sublinear: \( 1/k \), \( 1/k^2 \), etc.
- dependence of rate on dimension \( n \) and amount of data \( T \).

Sometimes improvements in theoretical rates are reflected in practice e.g. steepest-descent vs accelerated methods.

The question: "What can we say about the nonconvex case?" is especially relevant in deep learning.
Steepest Descent

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

Known to have linear convergence in strongly convex case, sublinear for weakly convex (e.g. Nesterov, 2004).

Fundamental and useful as a basis for algorithms and is extendible to settings more general than convex smooth optimization.

- \( \nabla f(x_k) \) can be replaced by a sparse, cheap approximation.
  (Stochastic Gradient)
- The sparsity may be lost in more sophisticated methods e.g. accelerated (see below...)
Momentum

Can get dramatic improvements in convergence 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 first-order methods are very similar. They typically separate the steepest descent step from the momentum step to generate two or three interleaved sequences.

Momentum term stores information from previous iterates and gradually phases it out.
Momentum Methods

Denoting $\kappa = \text{condition number}$, heav-ball sets

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

to get a linear convergence rate with constant approximately $1 - 2/\sqrt{\kappa}$. Contrast with rate of $1 - 1/\kappa$ for steepest descent. Results in a factor of $\sqrt{\kappa}$ improvement in work to achieve a given accuracy $\epsilon$.

Conjugate gradient is motivated quite differently, but has the same form and achieves a similar convergence rate. CG makes a more adaptive choice of $\alpha_k$ and $\beta_k$. 
** 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 $\partial f$.

A useful form of $f$ for these methods in learning problems is

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

where each $f_i$ is convex.

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 $\alpha_k > 0$.

When $f$ is strongly convex, the analysis of convergence of $E(\|x_k - x^*\|^2)$ is elementary (e.g. Nemirovski et al, 2009).
Stochastic gradient can be smoothed by averaging in primal or dual.

**Primal Averaging:** Report weighted average of all iterates:

\[
\bar{x}_T := \sum_{t=1}^{T} \gamma_t x_t.
\]

(For suitable choices of \(\gamma_t\), can keep a running tally cheaply.)

**Dual averaging:** Define search direction

\[
g_t := \frac{1}{T} \sum_{j=1}^{T} \nabla f_j(x_j).
\]

Average of all gradient estimates seen so far.

In regularized or constrained problems, dual-averaging iterates typically identify the optimal manifold finitely with high probability (Lee and Wright, 2012). (Not true for standard or primal-averaged SG.)
Parallel Stochastic Approximation

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 $\nabla f_i$ 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, evaluate $g := \nabla f_{i_k}(x)$;
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 $\tau$ on their age.
- Processors can overwrite each other’s work, but sparsity of $\nabla f_e$ helps — updates to not interfere too much.


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

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

<table>
<thead>
<tr>
<th>data set</th>
<th>size (GB)</th>
<th>$\rho$</th>
<th>$\Delta$</th>
<th>time (s)</th>
<th>speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVM</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RCV1</td>
<td>0.9</td>
<td>4.4E-01</td>
<td>1.0E+00</td>
<td>10</td>
<td>4.5</td>
</tr>
<tr>
<td>Netflix</td>
<td>1.5</td>
<td>2.5E-03</td>
<td>2.3E-03</td>
<td>301</td>
<td>5.3</td>
</tr>
<tr>
<td>KDD</td>
<td>3.9</td>
<td>3.0E-03</td>
<td>1.8E-03</td>
<td>878</td>
<td>5.2</td>
</tr>
<tr>
<td>JUMBO</td>
<td>30</td>
<td>2.6E-07</td>
<td>1.4E-07</td>
<td>9,454</td>
<td>6.8</td>
</tr>
<tr>
<td>Cuts</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DBLife</td>
<td>0.003</td>
<td>8.6E-03</td>
<td>4.3E-03</td>
<td>230</td>
<td>8.8</td>
</tr>
<tr>
<td>Abdomen</td>
<td>18</td>
<td>9.2E-04</td>
<td>9.2E-04</td>
<td>1,181</td>
<td>4.1</td>
</tr>
</tbody>
</table>
Identifying Subspaces from Partial Observations

Often we observe a certain phenomenon on a high-dimensional ambient space, but the phenomenon lies on a low-dimension subspace. Moreover, our observations may not be complete: “missing data.”

Can we recover the subspace of interest?

- **Matrix completion, e.g. Netflix.** Observe partial rows of an \( m \times n \) matrix; each row lies (roughly) in a low-d subspace of \( \mathbb{R}^n \).
- **Structure from Motion:** Observe a 3-d object from different camera angles, noting the location of reference points on the object’s surface on the (2-d) photo taken at each camera angle.
  - Object is solid, so some references points occluded in each photo: missing data.
  - Matrix of reference point locations in the photos has rank three.
  - Range subspace reveals 3-d location of reference points.
(Kennedy, Balzano, Taylor, Wright, 2012)
Euclidean Subspace Identification

- Seek subspace $S \subset \mathbb{R}^n$ of known dimension $d \ll n$.
- Given vectors $v_t \in S$, $t = 1, 2, \ldots$ for which only the elements on support $\Omega_t \subset \{1, 2, \ldots, n\}$ are revealed.

If the full vectors are revealed (full data: $\Omega_t \equiv \{1, 2, \ldots, n\}$), and vectors $v_t$ are randomly drawn from $S$, we obtain the solution after $d$ steps.

- Use an SVD to get a spanning orthonormal matrix of size $n \times d$.

For the general case consider online algorithms that

- Maintain an estimate $U_t$ (orthonormal $n \times d$) whose range is the current estimate of $S$;
- Update $U_t \rightarrow U_{t+1}$ when the next $(v_t)_{\Omega_t}$ is received.
Incremental SVD

Naive SVD approach: Estimate $U_t$ by
- Assembling $(v_t)_{\Omega_t}$ into an $n \times t$ matrix, with unknown elements $= 0$.
- Define $U_t$ as leading $d$ singular vectors

performs poorly. (The zeros confuse it.)

A more successful \textit{incremental SVD} approach:
- Impute the unknown elements $(v_t)_{\Omega_t^c}$ from the current estimate $U_t$ and the known elements $(v_t)_{\Omega_t}$:

$$w_t := \arg \min_w \| [U_t w - v_t]_{\Omega_t} \|_2^2; \quad \tilde{v}_t := \begin{bmatrix} [v_t]_{\Omega_t} \\ [U_t]_{\Omega_t^c} w_t \end{bmatrix}.$$ 

- Append the completed vector $\tilde{v}_t$ to $U_t$ and take the SVD of the resulting $n \times (d + 1)$ matrix;
- Define $U_{t+1}$ to be the leading $d$ singular vectors of $[U_t : \tilde{v}_t]$ 

(Balzano & Wright, in preparation)
Define an incremental algorithm that generates a sequence of \( n \times d \) orthonormal \( U_t \), applying a rank-one update to get \( U_{t+1} \), based on information in \( (v_t)_{\Omega_t} \).

\[
\begin{align*}
  w_t &:= \arg \min_w \| [U_t w - v_t]_{\Omega_t} \|^2_2 \\
  p_t &:= U_t w_t, \\
  r_t &:= v_t - U_t w_t, \\
  \alpha_t &:= \text{positive step length parameter}; \\
  z_t &:= \cos \left( \alpha_t \frac{r_t}{\| p_t \|} \right) \frac{p_t}{\| p_t \|} + \sin \left( \alpha_t \frac{r_t}{\| p_t \|} \right) \frac{r_t}{\| r_t \|} \quad \text{(note: } \| z_t \| = 1) \\
  U_{t+1} &:= U_t + z_t \frac{w_t^T}{\| w_t \|}
\end{align*}
\]

For a particular choice of \( \alpha_t \) (close to 1) the update is equivalent to a step of incremental SVD. (Balzano and Wright, 2012)
Convergence

Measure convergence of quantity

\[
\sum_{i=1}^{d} \sin^2(\phi_{t,i}) = d - \|A_t\|_F^2, \quad \text{where } A_t := U_t^T \bar{U},
\]

where \(\bar{U}\) is an orthonormal \(n \times d\) matrix that spans \(S\) and \(\phi_{t,i}\) are the angles between the subspaces \(R(U_t)\) and \(S\).

In the full-data case, for reasonable choice of \(\alpha_t\), this quantity increases by

\[
\frac{\sin((1/2) \sin 2\theta_t)}{\sin^2 \theta_t} \sin(2\theta_t - (1/2) \sin 2\theta_t) \left(1 - \frac{w_t^T A_t A_t^T w_t}{w_t^T w_t}\right),
\]

where \(\theta_t\) is the angle between the sampled vector \(v_t\) and \(R(U_t)\).

Under reasonable conditions, the expected value of this quantity is at least a fraction of the current error \(d - \|A_t\|_F^2\), so we get an expected linear decrease. But full-data is not interesting — SVD is better in this case.
Convergence for the missing-data case is more challenging. We have partial results (highly technical) for the case of i.i.d choices of revealed set $\Omega_t$ and sample vector $v_t$. Some highlights:

- Need roughly $|\Omega_t| \approx d \log(d/\delta)$ to ensure that $(U_t)_{\Omega_t}$ has all singular values close to $\sqrt{|\Omega_t|/n}$ with probability at least $1 - \delta$.

- There is a practical way to skip updates where $(U_t)_{\Omega_t}$ is poorly conditioned; they are unlikely to make good progress.

- The expected improvement in $\|A_t\|_F^2$ is asymptotically
  \[ \approx \|r_t\|^2/\|w_t\|^2, \]
  which has an expected value of about
  \[ \frac{1}{d} \sum_{i=1}^{d} \sin^2 \phi_{t,i}, \]
  suggesting linear convergence in expectation with rate $(1 - 1/d)$.

- High-probability improvement rate is similar.

(Balzano, Wright)
**Atomic-Norm Regularization**

Chandrasekaran, Parrilo, Recht, Willsky (2010)

General framework for structured / sparse optimization: Seek $x$ expressible in terms of a few *atoms*: “building block” objects in variable space.

Atomic set $\mathcal{A}$ may have large or infinite cardinality. Examples:

- In $\mathbb{R}^n$, atoms $a \in \mathcal{A}$ could be unit vectors $\pm e_i$, $i = 1, 2, \ldots, n$ (sparse).
- In $\mathbb{R}^n$, given groups $[g] \subset \{1, 2, \ldots, n\}$, each atom $a$ could be a unit vector with support $[g]$, for some $g \in \mathcal{G}$ (group sparse).
- In $\mathbb{R}^{m \times n}$, each atom $a$ could be a rank-one $m \times n$ matrix.
- In the space of functions $\phi : [0, 1] \to \mathbb{C}$, each atom could have the form $\exp(2\pi ift)$, for a given frequency $f$.

Given an objective $f$, seek an approximate minimizer $x$ of the form

$$x = \sum_{a \in \mathcal{A}} c_a a$$

such that $c_a \geq 0$ for all $a$, and few of these coefficients are nonzero.
Atomic-Norm Regularized Formulations

Define the *atomic norm* $\|x\|_A$

$$\|x\|_A := \min \left\{ \sum_{a \in A} c_a \mid x = \sum_{a \in A} c_a a, \ c_a \geq 0 \right\}.$$  

(i.e. gauge defined by the convex hull of $A$.) The formulation

$$\min f(x) \text{ subject to } \|x\|_A \leq M$$

(for given $M > 0$) tends to recover $x$ with sparse atomic representation.

Can formulate algorithms for the various special cases — but is a general approach available?

We describe a greedy approach similar to matching pursuit, but allows culling of the working basis: A “Forward-Backward” greedy algorithm optimization (FOBA). (Rao, Shah, Wright, Nowak, 2012)
Forward-Backward Greedy

At iteration $t$, given $x_t$ and a current working basis of atoms $A_t \subset A$ with $x_t = \sum_{a \in A_t} c_a a$ and $\sum_{a \in A_t} c_a \leq M$.

FORWARD:
- Choose new atom $a_t := \arg \min_{a \in A} \langle \nabla f(x_t), a \rangle$; Add to basis: $A_{t+1} := A_t \cup \{a_t\}$;
- Find $\gamma_{t+1} \in (0, 1)$ that minimizes $f(x_t + \gamma(Ma_t - x_t))$, and update $x$ and its representation accordingly;
- Possibly improve $x$ (e.g. by some steps of gradient projection);

BACKWARD:
- Find atom in current basis $A_{t+1}$ whose removal has the least predicted effect on $f$;
- If degradation of $f$ is less than $\eta$ times improvement on the last forward step (for some $\eta \in (0, 1)$), remove this term, and adjust $x$ accordingly.
- Possibly improve $x$ (e.g. by some more gradient projection steps).
Implementation and Convergence

When $f$ is linear least squares:

$$f(x) = \frac{1}{2} \| \Phi x - y \|_2^2,$$

the **backward step** can be implemented efficiently. **Enhancement steps** require approximate solution of

$$\min_{c_a, a \in A_{t+1}} \frac{1}{2} \| \Phi \left( \sum_{a \in A_{t+1}} c_a a \right) - y \|_2^2 \quad \text{s.t.} \quad c_a \geq 0, \quad \sum c_a \leq M,$$

which is convex QP over a scaled simplex. Use **projected gradient**.

**Convergence** follows immediately from the forward-only result: (Tewari, Ravikumar, Dhillon, 2012); see also (Zhang, 2012) and (Jaggi, 2011)

$$f(x_T) - f(x^*) \leq 4R/T,$$

where

$$L := \sup_{\|x\|_A \leq M} \| \nabla^2 f(x) \|, \quad \| A \| := \sup_{a \in A} \| a \|, \quad R := 2LM^2 \| A \|^2,$$
Convergence Proof

Denote $\delta_t := f(x_t) - f(x^*)$. We have

$$f(x_t + \gamma(Ma_t - x_t)) \leq f(x_t) + \gamma \langle \nabla f(x_t), Ma_t - x_t \rangle + \frac{1}{2} \gamma^2 L \|Ma_t - x_t\|^2$$

$$\leq f(x_t) + \gamma \langle \nabla f(x_t), Ma_t - x_t \rangle + R$$

and

$$-\delta_t = f(x^*) - f(x_t) \geq \langle \nabla f(x_t), x^* - x_t \rangle \geq \langle \nabla f(x_t), Ma_t - x_t \rangle.$$

By substituting into previous expression we get

$$f(x_t + \gamma(Ma_t - x_t)) \leq f(x_t) - \gamma \delta_t + R$$

Subtract $f(x^*)$ from both sides, minimize over $\gamma \in [0, 1]$ to get

$$\delta_{t+1} \leq \delta_t - \delta_t^2/(4R).$$

Result follows from an elementary recursive argument:

$$\delta_t \leq \frac{4R}{t} \Rightarrow \delta_{t+1} \leq 4R \left( \frac{1}{t} - \frac{1}{t^2} \right) = 4R \frac{t - 1}{t^2} \leq 4R \frac{t - 1}{t^2 - 1} = \frac{4R}{t + 1}.$$
** Deep Learning Strategies

Optimization techniques used in deep learning (in various combinations!) include:

- stochastic gradient;
- inexact Newton methods
  - with damping
  - “Hessian-free” conjugate gradient implementation.
- quasi-Newton methods e.g. L-BFGS;
- block coordinate descent.

**Pretraining:** A variant of block-coordinate descent in which all parameters except those for a single layer are fixed. (Single-layer problems are often convex.) After one or more cycles, revert to training on the full space.

**Minibatch.** Random sample of data used to calculate approximate gradients and Hessians. Complete pass through the data is impractical and unnecessary, as big data sets typically have a great deal of redundancy.
Little is known about convergence properties under nonconvexity.

No evidence that the iterates terminate near a critical points / local minimum. But solution quality (as measured by classification performance on held-out data sets) improves well as a function of training time, on many applications.

Sometimes used as a first phase, before applying a Newton-like method. (Sainath et al, 2012)

Parallel SG used successfully on deep learning e.g. by Google group: Downpour SGD.
Hessian-free methods have been successful in speech processing applications (Martens, 2010, 2011)

Basic idea: Given $f(x) = (1/m) \sum_{i=1}^{m} f_i(x)$, choose $X \subset \{1, 2, \ldots, m\}$ and $S \subset X$, and set

$$H^k_S = \frac{1}{|S|} \sum_{i \in S} \nabla^2 f_i(x_k), \quad g^k_X = \frac{1}{|X|} \sum_{i \in X} \nabla f_i(x_k).$$

Approximate Newton step satisfies

$$H^k_S d_k = g^k_X.$$

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

Possibly add a damping term $\lambda I$, possibly use a Gauss-Newton approximation instead of $H^k_S$. 
Use conjugate gradient (CG) to find an approximate solution of the Newton-like equations. Each CG iteration requires one matrix-vector multiplication with $H^k_S$, plus some vector operations ($O(n)$ cost).

Hessian-vector product

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

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

Can use an extra gradient evaluation (finite difference to approximate this product). Martens (2010) suggests using adjoint techniques instead, to compute $\nabla^2 f_i(x_k) v$ exactly. Similar to automatic differentiation — require an “upwards sweep” followed by a “downwards sweep.”
No need to iterate CG to convergence, since $H_k^S$ is only an approximation to the Hessian (and indeed the quadratic Taylor series expansion is itself an approximation).

Typically, 2 to 50 CG iterations.

Line search along approximate Newton direction $d_k$ usually not needed; unit step suffices. The approximate Hessian sets the right scale for $d_k$.

(Byrd et al, 2012) provide an optimization analysis of the sampled inexact Newton approach.

Sampling and coordinate descent can be combined e.g for regularized logistic regression Wright (2012)
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.

- coordinate descent,
- augmented Lagrangian methods / alternating direction,
- quasi-Newton methods (L-BFGS),
- algorithms for sparse / regularized optimization,
- nonconvex regularizers,
- manifold identification / variable selection.

FIN