First-order and zero-order algorithms are drawing renewed attention, in part because of their usefulness in data analysis.

*Everything old is new again.*

I’ll touch on some recent thrusts in optimization algorithms, biased by my own perspectives.

I promise to be biased and short-sighted!

- Optimization formulations from data analysis.
- Algorithms: First-order, accelerated, coordinate descent, stochastic gradient, conditional gradient, regularized.
Data Analysis: A Driving Application

- Extract meaning from data: Understand statistical properties, important features, fundamental structures in the data. Inference, Data Analysis.
- Use this knowledge to make predictions about other, similar data. Machine Learning.

Typical Setup: After cleaning and formatting, obtain \( m \) data objects:
- Vectors \( a_j, j = 1, 2, \ldots, m \) whose elements are “features.”
- One outcome / observation / label \( y_j \) for each feature vector \((j = 1, 2, \ldots, m)\).

Seek a function \( \phi \) that:
- approximately maps \( a_j \) to \( y_j \) for each \( j \): \( \phi(a_j) \approx y_j \) for \( j = 1, 2, \ldots, m \).
- satisfies some additional properties — generalizability, simplicity, structure — that make it “plausible” in the context of the application and robust to small changes in the data.
What’s the use of $\phi$?

Often parametrize $\phi$ e.g. by a vector or matrix $x$, so the data analysis problem becomes a data fitting problem in which $x$ is the variable.

- **Analysis**: $\phi$ — especially the parameter $x$ that defines it — reveals structure in the data. Examples:
  - which components (features) in vectors $a_j$ are most important in determining the outputs $y_j$.
  - quantify how the output depend on the features.
  - show how each $a_j$ can be expressed in terms of a small number of basis vectors. ($X$ defines these basis vectors.)

- **Prediction**: Given a new data vector $a_k$, predict the output $y_k$ to be $\phi(a_k)$.

Many complications:

- $a_j$ and $y_j$ can contain noise or errors.
- **Missing Data**: Vectors $a_j$ may be missing elements.
- Some or all $y_j$ may be missing: semi-supervised / unsupervised.
- **Online learning**: Data $(a_j, y_j)$ is arriving in a stream.
Data Analysis Problems

**Least Squares:**

\[
\min_x f(x) := \frac{1}{2} \sum_{j=1}^{m} (a_j^T x - y_j)^2 = \frac{1}{2} \|Ax - y\|_2^2.
\]

(Gauss, 1799; Legendre, 1805.)

**Support Vector Machines:** Seek \((x, \beta)\) such that

\[
\begin{align*}
    a_j^T x - \beta &\geq 1 \quad \text{when } y_j = +1; \\
    a_j^T x - \beta &\leq -1 \quad \text{when } y_j = -1.
\end{align*}
\]

Design an objective so that the \(j\)th loss term is zero when classification is correct, positive otherwise. A popular one is **hinge loss**:

\[
\frac{1}{m} \sum_{j=1}^{m} \max(1 - y_j(a_j^T x - \beta), 0).
\]

Add a regularization term \((\lambda/2)\|x\|_2^2\) for some \(\lambda > 0\) to maximize the margin between the classes.
Regularize for Plausibility (Generalizability)
Regularize for Plausibility (Generalizability)
Regularize for Plausibility (Generalizability)
Nonlinear SVM
Data Analysis Problems

**Logistic Regression:** Similar to SVM, except that we find odds that data vector \( a \) is in \(+\) or \(-\) class, rather than making a simple prediction. Seek odds function parametrized by \( x \in \mathbb{R}^n \):

\[
p_+(a; x) := (1 + e^{a^T x})^{-1}, \quad p_-(a; x) := 1 - p_+(a; x).
\]

Choose \( x \) so that \( p_+(a_j; x) \approx 1 \) when \( y_j = +1 \) and \( p_-(a_j; x) \approx 1 \) when \( y_j = -1 \).

Minimize a negative log likelihood function:

\[
\min_x \mathcal{L}(x) := -\frac{1}{m} \left[ \sum_{y_j = -1} \log p_-(a_j; x) + \sum_{y_j = 1} \log p_+(a_j; x) \right].
\]

**Deep Learning:** Extend multiclass logistic regression to multiple layers. Composition of simple functions: linear mappings, hinge losses, logistic.
Inputs are the vectors $a_j$, outputs are odds of $a_j$ belonging to each class (as in multiclass logistic regression).

At each layer, inputs are converted to outputs by a linear transformation composed with an element-wise function:

$$a^{l+1} = \sigma(W^l a^l + h^l),$$

where $a^l$ is node values at layer $l$, $(W^l, h^l)$ are parameters in the network, $\sigma$ is the element-wise function.
Deep Learning

The element-wise function $\sigma$ makes simple transformations to each element:

- **Logistic function**: $t \rightarrow \frac{1}{1 + e^{-t}}$;
- **Hinge**: $t \rightarrow \max(t, 0)$;
- **Bernoulli**: $t \rightarrow 1$ with probability $1/(1 + e^{-t})$ and $t \rightarrow 0$ otherwise (inspired by neuron behavior).

The example depicted shows a completely connected network — but more typically networks are engineered to the application (speech processing, object recognition, ...).

- **local aggregation of inputs**: pooling;
- **restricted connectivity + constraints on weights (elements of $W^l$ matrices)**: convolutions.
GoogLeNet
Data Analysis Problem: Matrix Completion

Regression over a structured matrix: Observe a matrix \( X \) by probing with linear operators \( \mathcal{A}_j(X) \), giving observations \( y_j, j = 1, 2, \ldots, m \). Solve for \( X \):

\[
\min_X \frac{1}{2} \sum_{j=1}^{m} (\mathcal{A}_j(X) - y_j)^2 = \frac{1}{2} \| \mathcal{A}(X) - y \|_2^2.
\]

The \( \mathcal{A}_j \) typically observe a single element of \( X \), or a linear combination of elements.

Typically assume some structure on \( X \) e.g. low-rank, or low-rank plus sparse. A nuclear-norm (sum-of-singular-values) regularization term induces low rank on \( X \):

\[
\min_X \frac{1}{2} \| \mathcal{A}(X) - y \|_2^2 + \lambda \| X \|_* , \quad \text{for some } \lambda > 0.
\]
Fundamentals: First-Order Methods

\[ \min f(x), \text{ with smooth nonlinear } f. \]

Steepest descent generate iterates \( \{x^k\}_{k=0,1,2,...} \):

\[ x^{k+1} = x^k - \alpha_k \nabla f(x^k), \quad \text{for steplength } \alpha_k > 0. \]

For strongly convex \( f \), typical convergence is linear:

\[ f(x^k) - f(x^*) \leq C \tau^k, \quad \text{for some } \tau \in (0, 1). \]

For weakly convex \( f \), can get sublinear rates, e.g.

\[ f(x^k) - f(x^*) \leq C/k. \]

Can get faster rates, at no extra effort, by using momentum. Continue to move in direction of the previous step, with a tweak in the direction of negative gradient:

\[ x^{k+1} = x^k - \alpha_k \nabla f(x^k) + \beta_k (x^k - x^{k-1}), \]

for some steps \( \alpha_k > 0, \beta_k > 0 \). Heavy-ball, Nesterov acceleration, conjugate gradient are all variants of the momentum idea.
Coordinate Descent

For some $f$ it costs much less to work with a single component $\nabla_i f(x)$ of the gradient than the full gradient — difference of $1/n$ in cost.

Coordinate descent has become popular again: At iteration $k$:

- Choose index $i_k \in \{1, 2, \ldots, n\}$;
- Take step $x_{k+1} \leftarrow x_k - \alpha_k \nabla_{i_k} f(x_k)e_{i_k}$.

Stochastic version: choose $i_k$ uniformly at random at each iteration. Get similar convergence rates in theory to steepest descent. Can be significantly faster in practice.

Parallel, asynchronous versions are particularly effective.
Stochastic Gradient

Functions of summation form are common in data analysis:

\[ f(x) = \frac{1}{m} \sum_{i=1}^{m} f_i(x). \]

(Least squares, deep learning, SVM.)

**Stochastic gradient:** 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 \( \alpha_k > 0 \).

Parallel asynchronous versions are popular (**HOGWILD!**, 2011).

Typically sublinear convergence — \( 1/k \) or \( 1/\sqrt{k} \).

Recent versions: SAG and SAGA (2013-). Maintain an estimate of each term \( \nabla f_i(x), i = 1, 2, \ldots, m \). Update one of them at each iteration. These are related to full-gradient methods and typically have **linear** convergence.
Augmented Lagrangian

Consider linearly constrained problem:

\[
\begin{align*}
\min & \ f(x) \quad \text{s.t.} \quad Ax = c.
\end{align*}
\]

Augmented Lagrangian is

\[
L(x, \lambda; \rho) := f(x) + \lambda^T (Ax - c) + \frac{\rho}{2} \|Ax - c\|_2^2,
\]

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

\[
x_k := \arg \min_x L(x, \lambda_{k-1}; \rho_k); \quad \lambda_k := \lambda_{k-1} + \rho_k (Ax_k - c); \quad (\text{choose } \rho_{k+1}).
\]


Precursor is quadratic penalty approach, which omits the \( \lambda \) term and is attributed to Courant (1943).
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.

(Eckstein and Bertsekas, 1992; Boyd et al, 2011).
Sparse Optimization & Regularization

Objective often includes a **regularization** term:

\[
\min f(x) + \lambda \psi(x),
\]

where \( f \) is convex and smooth, \( \psi \) convex and “simple” but usually nonsmooth, and \( \lambda \) is a positive parameter.

Account for \( \psi \) by replacing the gradient step with:

\[
x^{k+1} = \arg \min_x \nabla f(x^k)^T(x - x^k) + \frac{1}{2\alpha_k} \|x - x^k\|^2 + \lambda \psi(x).
\]

This is the **shrinkage** step. Often cheap to compute, e.g. when \( \psi(x) = \|x\|_1 \), it is separable in the components of \( x \).

This device of shrinkage gives a way to extend many algorithms to regularized objectives: accelerated gradient, coordinate descent, stochastic gradient.
Frank-Wolfe / Conditional Gradient

Minimize a smooth convex function over a compact convex set $\Omega$:

$$\min_{x \in \Omega} f(x).$$

Frank-Wolfe finds a search direction by minimizing a linear approximation to $f$ over the full constraint set $\Omega$, then takes a step toward this point

$$z^{k+1} = \arg \min_{z \in \Omega} f(x^k) + \nabla f(x^k)^T (z - x^k),$$

$$x^{k+1} = x^k + \alpha_k (z^{k+1} - x^k),$$

where the classical step size is $\alpha_k = 1/(k + 2)$. Sublinear ($1/k$) convergence.

Useful when the linear subproblem is much easier to formulate and solve than the original.

We used it recently as a framework for atomic-norm-regularized minimization, a paradigm of widespread interest.
Conclusions

- Optimization algorithms are proving to be useful in more and more applications.
- The basic approaches are often elementary, but many new angles:
  - Parallelism
  - Randomness
  - Nonsmooth regularization
  - Low-accuracy solutions.
- Algorithm and Formulation “Tools” can be customized to the application. Often requires involvement of both domain specialists and people trained in optimization.
- Mismatches between theory and practical performance in many areas – much remains to be understood.