Optimization in Data Analysis

Stephen Wright

University of Wisconsin-Madison

IMA New Directions, August 2016

Outline

- Data Analysis and Machine Learning
  - The Setup
  - About Ten Applications / Examples
- Optimization in Data Analysis
  - Basic Formulations
  - Relevant Algorithms

Data Analysis

Also: Machine Learning, Statistical Inference, Data Mining.

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

Foundations in Statistics, then Computer Science (AI, Machine Learning, Databases, Parallel Systems), more recently Optimization.

For a recent Statistics perspective on past, present, and future, see David Donoho: *50 Years of Data Science*, September 2015.

Modeling and domain-specific knowledge is vital: "80% of data analysis is spent on the process of cleaning and preparing the data."
[Dasu and Johnson, 2003].

(Most academic research deals with the other 20%.)

"Big Data"

New "Data Science Centers" at many institutions, new degree programs (e.g. Masters in Data Science), new funding initiatives.

- Huge amounts of data are being collected, routinely.
  - Consumer and citizen data: phone logs (call and text), email, surveillance cameras, web activity, online shopping activity,...
  - Scientific data (particle colliders, satellites, biological / genomic, astronomical,...)
- Affects everyone directly!
- Powerful computers make it possible to handle larger data sets and analyze more thoroughly.
- Methodological innovations in some areas. e.g. Deep Learning.
  - Speech recognition in smart phones
  - AlphaGo: AI playing Go

Typical Setup

After cleaning and formatting, obtain a data set of m objects:

- Vectors of features: a_j, $j = 1, 2, \ldots, m$.
- Outcome / observation / label y_j for each feature vector.

The outcomes y_j could be:

- a real number: **regression**
- a label indicating that a_j lies in one of M classes (for $M \geq 2$): **classification**
- multiple labels: classify a_j according to multiple criteria.
- no labels (y_j is null):
  - **subspace identification:** Locate low-dimensional subspaces that approximately contain the (high-dimensional) vectors a_j;
  - **clustering**: Partition the a_j into a few clusters.

  (Structure may reveal which features in the a_j are important / distinctive, or enable predictions to be made about new vectors a.)

Fundamental Data Analysis Task

Seek a function ϕ that:

- approximately maps a_j to y_j for each j: $\phi(a_j) \approx y_j$ for $j = 1, 2, \ldots, m$, if labels y_j are available.
- if there are no labels y_j, or if some labels are missing, seek ϕ that does something useful with the data $\{a_j\}$, e.g. assigns each a_j to an appropriate cluster or subspace.
- satisfies some additional properties — simplicity, structure — that make it "plausible" for the application, robust to perturbations in the data, generalizable to other data samples.

Can usually define ϕ in terms of some parameter vector x — thus identification of ϕ becomes a data-fitting problem.

Objective function in this problem often built up of m terms that capture mismatch between predictions and observations for each (a_j, y_j).

The process of finding ϕ is often called learning or training.

Why?

What's the use of ϕ?

- Analysis: ϕ — especially the parameter x that defines it — reveals structure in the data. Examples:
  - Feature selection: reveal the components of vectors a_j that are most important in determining the outputs y_j, and quantifies the importance of these features.
  - Uncovers some hidden structure, e.g.
    - ⋆ finds some low-dimensional subspaces that contain the a_j;
    - ⋆ find clusters that contain the a_j;
    - ⋆ find a decision tree that builds intuition about how y_j depend on a_j.
- Prediction: Given new data vectors a_k, predict outputs $y_k \leftarrow \phi(a_k)$.

Complications

- **noise or errors** in a_j and y_j. Would like ϕ (and x) to be robust to this. Thus, generalization / regularization.
- **avoid overfitting**: Observed data is viewed as an empirical, sampled representation of some underlying reality. Want to avoid overfitting to the particular sample. (Ideally, the training process should produce a similar result for other samples from the same data set.) Again, generalization / regularization.
- **missing data**: Vectors a_j may be missing elements (but may still contain useful information).
- **missing labels:** Some or all y_j may be missing or null — semi-supervised or unsupervised learning.
- **online learning**: Data (a_j, y_j) is arriving in a stream rather than all known up-front.

Application I: 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]; see [Stigler, 1981].

Here the function mapping data to output is linear: $\phi(a_j) = a_j^T x$.

- ℓ_2 regularization reduces sensitivity of the solution x to noise in y.

$$\min_x \frac{1}{2}\|Ax - y\|_2^2 + \lambda\|x\|_2^2.$$

- ℓ_1 regularization yields solutions x with few nonzeros:

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

  Feature selection: Nonzero locations in x indicates important components of a_j.

- Nonconvex separable regularizers (SCAD, MCP) have nice statistical properties, but lead to nonconvex optimization formulations.

Application II: Matrix Completion

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

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

Each $\mathcal{A}_j$ may observe a single element of X, or a linear combination of elements. Can be represented as a matrix A_j, so that $\mathcal{A}_j(X) = \langle A_j, X \rangle$.

Seek the "simplest" X that satisfies the observations, e.g. low-rank, or low-rank plus sparse. A nuclear-norm (sum-of-singular-values) regularization term induces low rank on X:

$$\min_X \frac{1}{2m} \|\mathcal{A}(X) - y\|_2^2 + \lambda \|X\|_*, \quad \text{for some } \lambda > 0.$$

[Recht et al., 2010]

Explicit Low-Rank Parametrization

Compact, nonconvex formulation is obtained by parametrizing X directly:

$$X = LR^T, \quad \text{where } L \in \mathbb{R}^{m\times r},\ R \in \mathbb{R}^{n\times r},$$

where r is known (or suspected) rank.

$$\min_{L,R} \frac{1}{2m}\sum_{j=1}^{m}(\mathcal{A}_j(LR^T) - y_j)^2.$$

For symmetric X, write $X = ZZ^T$, where $Z \in \mathbb{R}^{n\times r}$, so that

$$\min_{Z} \frac{1}{2m}\sum_{j=1}^{m}(\mathcal{A}_j(ZZ^T) - y_j)^2.$$

(No need for regularizer — rank is hard-wired into the formulation.)

Despite the nonconvexity, near-global minima can be found when $\mathcal{A}_j$ are incoherent. Use appropriate initialization [Candès et al., 2014], [Zheng and Lafferty, 2015] or the observation that all local minima are near-global [Bhojanapalli et al., 2016].

Application III: Nonnegative Matrix Factorization

Given $m \times n$ matrix Y, seek factors L $(m \times r)$ and R $(n \times r)$ that are element-wise positive, such that $LR^T \approx Y$.

$$\min_{L,R} \frac{1}{2}\|LR^T - Y\|_F^2 \text{ subject to } L \geq 0,\ R \geq 0.$$

Applications in computer vision, document clustering, chemometrics, ...

Could combine with matrix completion, when not all elements of Y are known, if it makes sense on the application to have nonnegative factors.

If positivity constraint were not present, could solve this in closed form with an SVD, since Y is observed completely.

Application IV: Sparse Inverse Covariance

Let $Z \in \mathbb{R}^p$ be a (vector) random variable, whose entries have zero mean. Let $z_1, z_2, \ldots, z_N$ be samples of Z. Sample covariance matrix (estimates covariance between components of Z):

$$S := \frac{1}{N-1} \sum_{\ell=1}^{N} z_\ell z_\ell^T.$$

Seek a sparse inverse covariance matrix: $X \approx S^{-1}$.

X reveals dependencies between components of Z: $X_{ij} = 0$ if the i and j components of Z are conditionally independent.

(Nonzeros in X indicate arcs in the dependency graph.)

Obtain X from the regularized formulation:

$$\min_X \langle S, X \rangle - \log\det(X) + \lambda \|X\|_1, \quad \text{where } \|X\|_1 = \textstyle\sum_{i,j} |X_{ij}|.$$

[Friedman et al., 2008, Scheinberg and Ma, 2012].

Application V: Subspace Identification

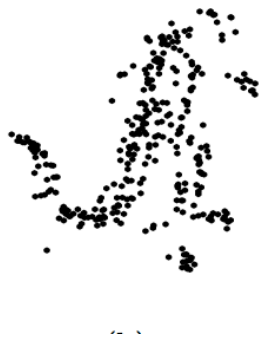
Given vectors $a_j \in R^n$ with missing entries, find a subspace of $\mathbb{R}^n$ such that all "completed" vectors a_j lie approximately in this subspace.

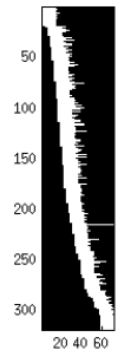
If $\Omega_j \subset \{1, 2, \ldots, n\}$ is the set of observed elements in a_j, seek $X \in \mathbb{R}^{n \times d}$ such that

$$[a_j - Xs_j]_{\Omega_j} \approx 0,$$

for some $s_j \in \mathbb{R}^d$ and all $j = 1, 2, \ldots$.
[Balzano et al., 2010, Balzano and Wright, 2014].

Application: Structure from motion. Reconstruct opaque object from planar projections of surface reference points.





Application VI: Linear Support Vector Machines

Each item of data belongs to one of two classes: $y_j = +1$ and $y_j = -1$.

Seek (x, β) such that

$$\begin{aligned} a_j^T x - \beta &\geq 1 \quad \text{when } y_j = +1; \\ a_j^T x - \beta &\leq -1 \quad \text{when } y_j = -1. \end{aligned}$$

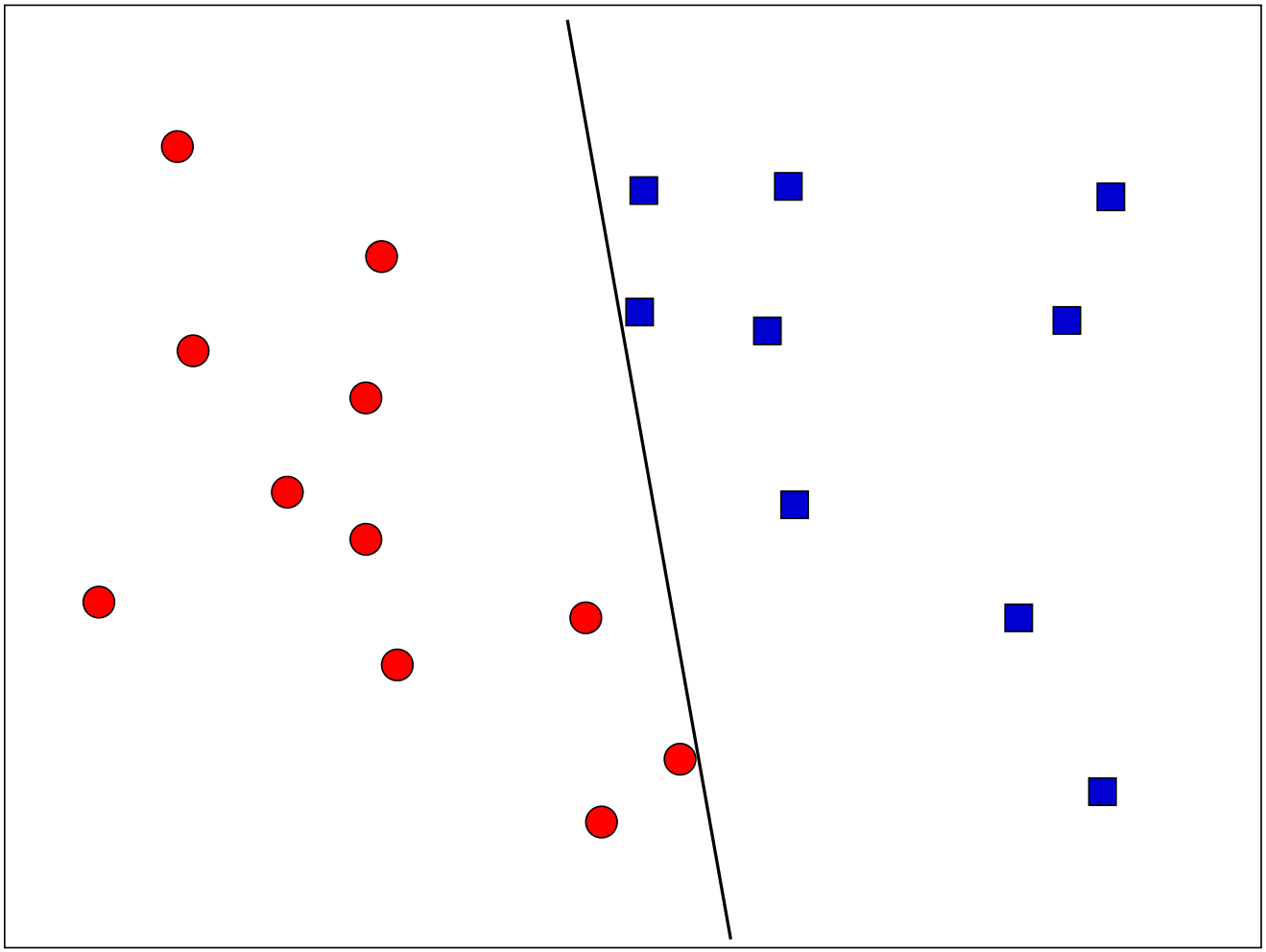
The mapping is $\phi(a_j) = \text{sign}(a_j^T x - \beta)$.

Design an objective so that the jth loss term is zero when $\phi(a_j) = y_j$, positive otherwise. A popular one is hinge loss:

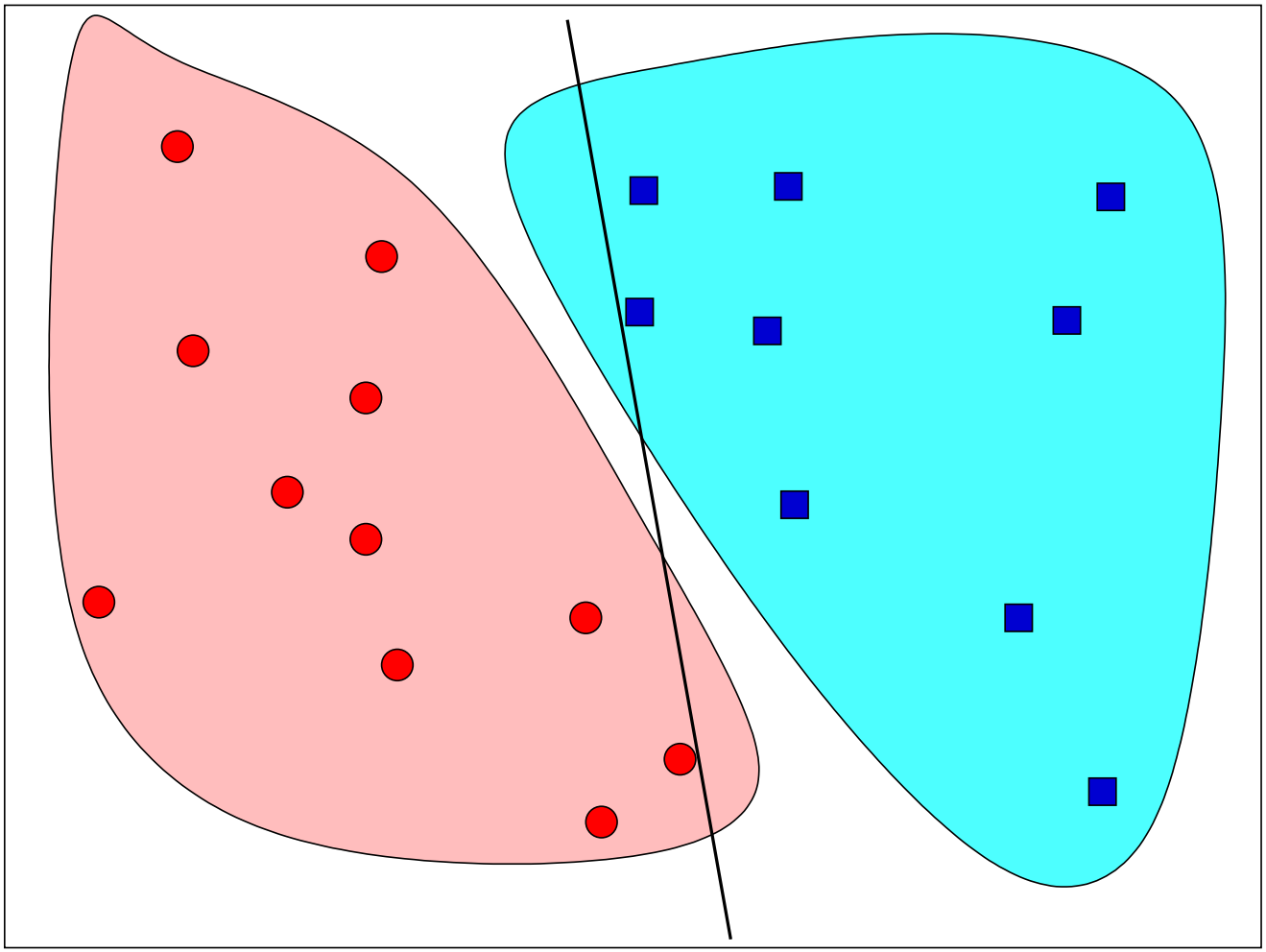
$$H(x) = \frac{1}{m} \sum_{j=1}^{m} \max(1 - y_j(a_j^T x - \beta), 0).$$

Add a regularization term $(\lambda/2)\|x\|_2^2$ for some $\lambda > 0$ to maximize the margin between the classes.

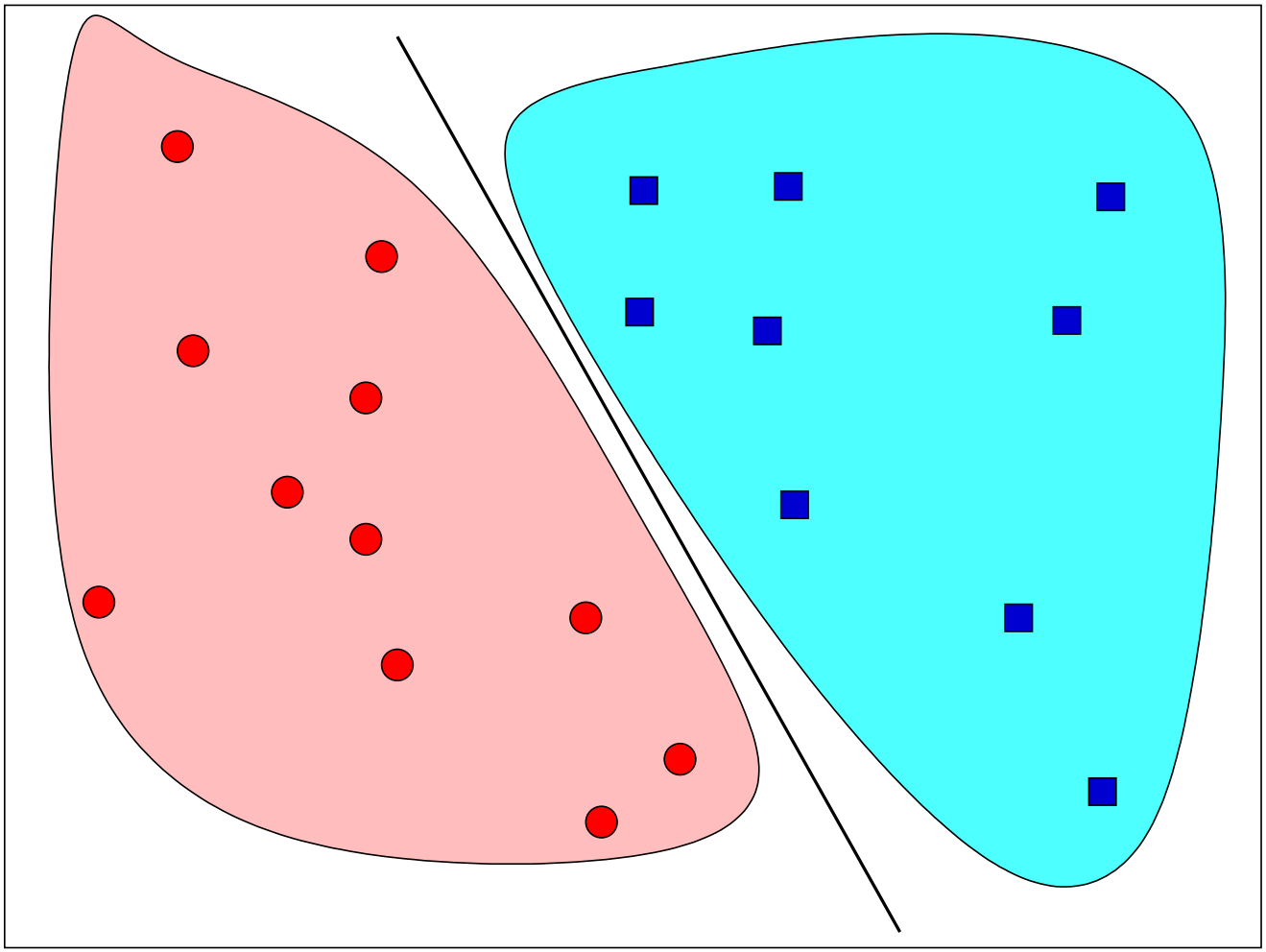
Regularize for Generalizability



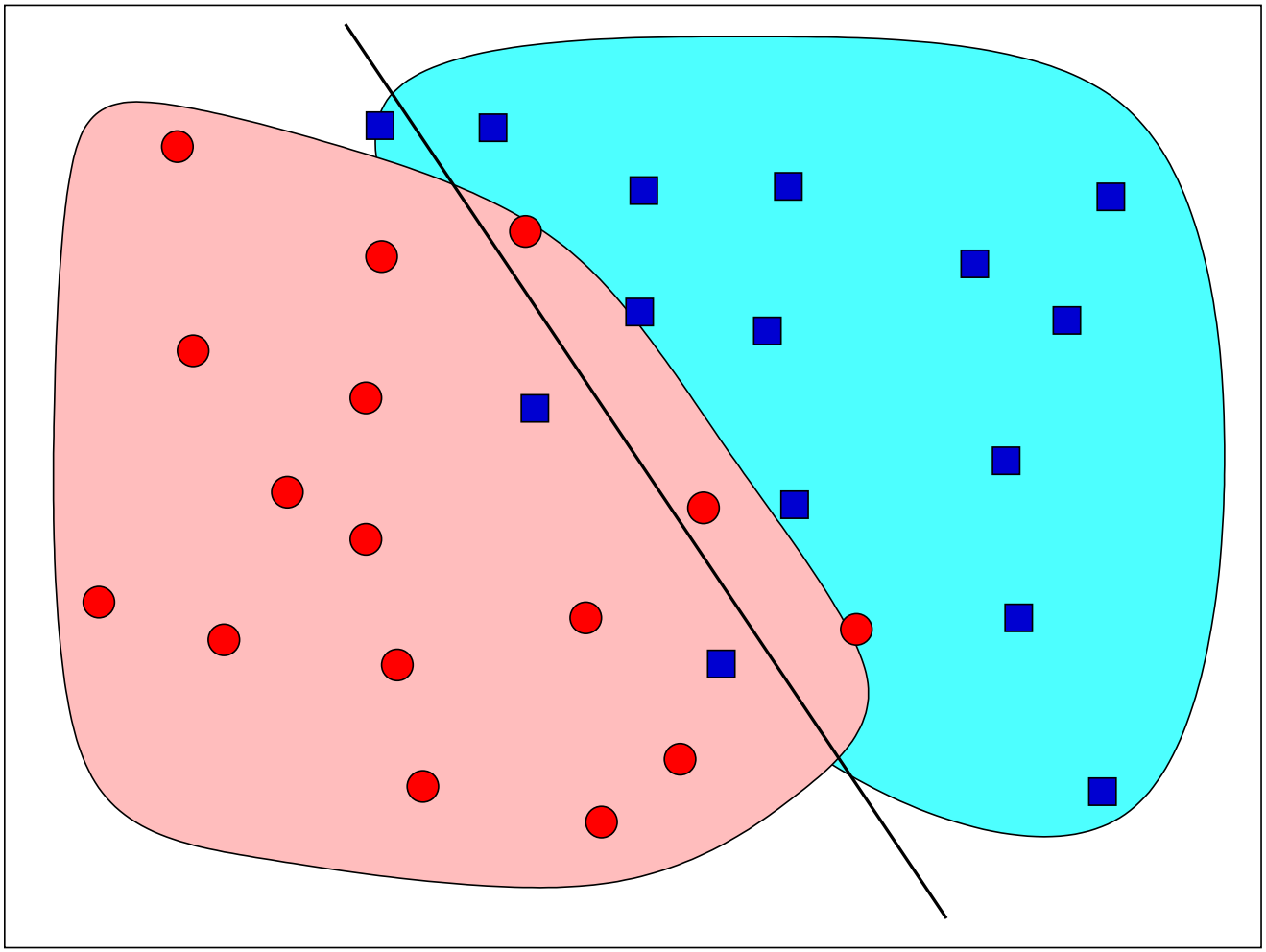
Regularize for Generalizability



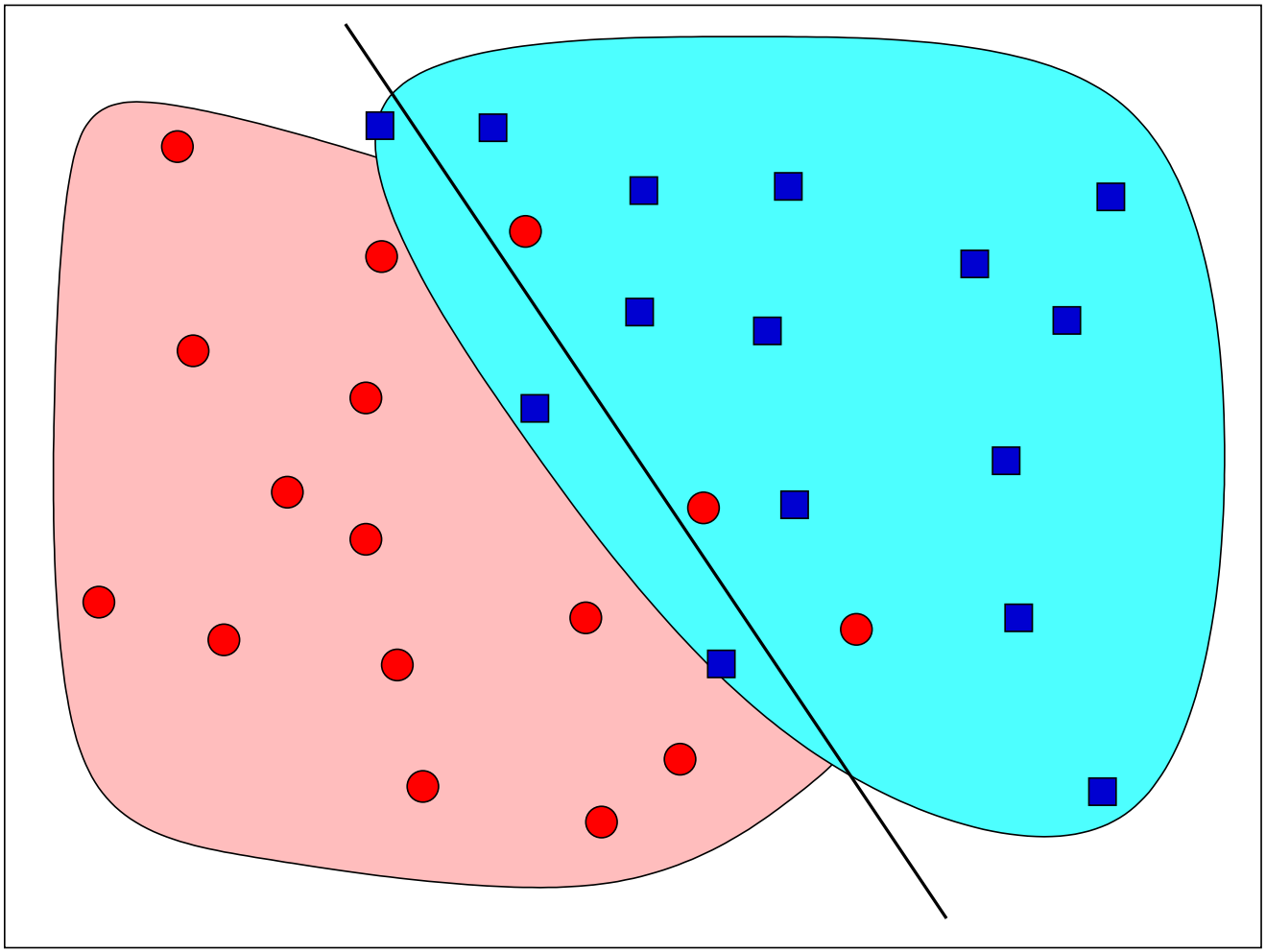
Regularize for Generalizability



Regularize for Generalizability



Regularize for Generalizability



Application VII: Nonlinear SVM

Data a_j, $j = 1, 2, \ldots, m$ may not be *separable* neatly into two classes $y_j = +1$ and $y_j = -1$. Apply a nonlinear transformation $a_j \to \psi(a_j)$ ("lifting") to make separation more effective. Seek (x, β) such that

$$\begin{aligned} \psi(a_j)^T x - \beta &\geq 1 \quad \text{when } y_j = +1; \\ \psi(a_j)^T x - \beta &\leq -1 \quad \text{when } y_j = -1. \end{aligned}$$

Leads to the formulation:

$$\frac{1}{m} \sum_{j=1}^{m} \max(1 - y_j(\psi(a_j)^T x - \beta), 0) + \frac{1}{2}\lambda \|x\|_2^2.$$

Can avoid defining ψ explicitly by using instead the dual of this QP.

Nonlinear SVM: Dual

Dual is a quadratic program in m variables, with simple constraints:

$$\min_{\alpha \in \mathbb{R}^m} \frac{1}{2}\alpha^T Q\alpha - e^T\alpha \text{ s.t. } 0 \leq \alpha \leq (1/\lambda)e, \quad y^T\alpha = 0.$$

where $Q_{k\ell} = y_k y_\ell \psi(a_k)^T\psi(a_\ell)$, $y = (y_1, y_2, \ldots, y_m)^T$, $e = (1, 1, \ldots, 1)^T$.

No need to choose $\psi(\cdot)$ explicitly. Instead choose a kernel K, such that

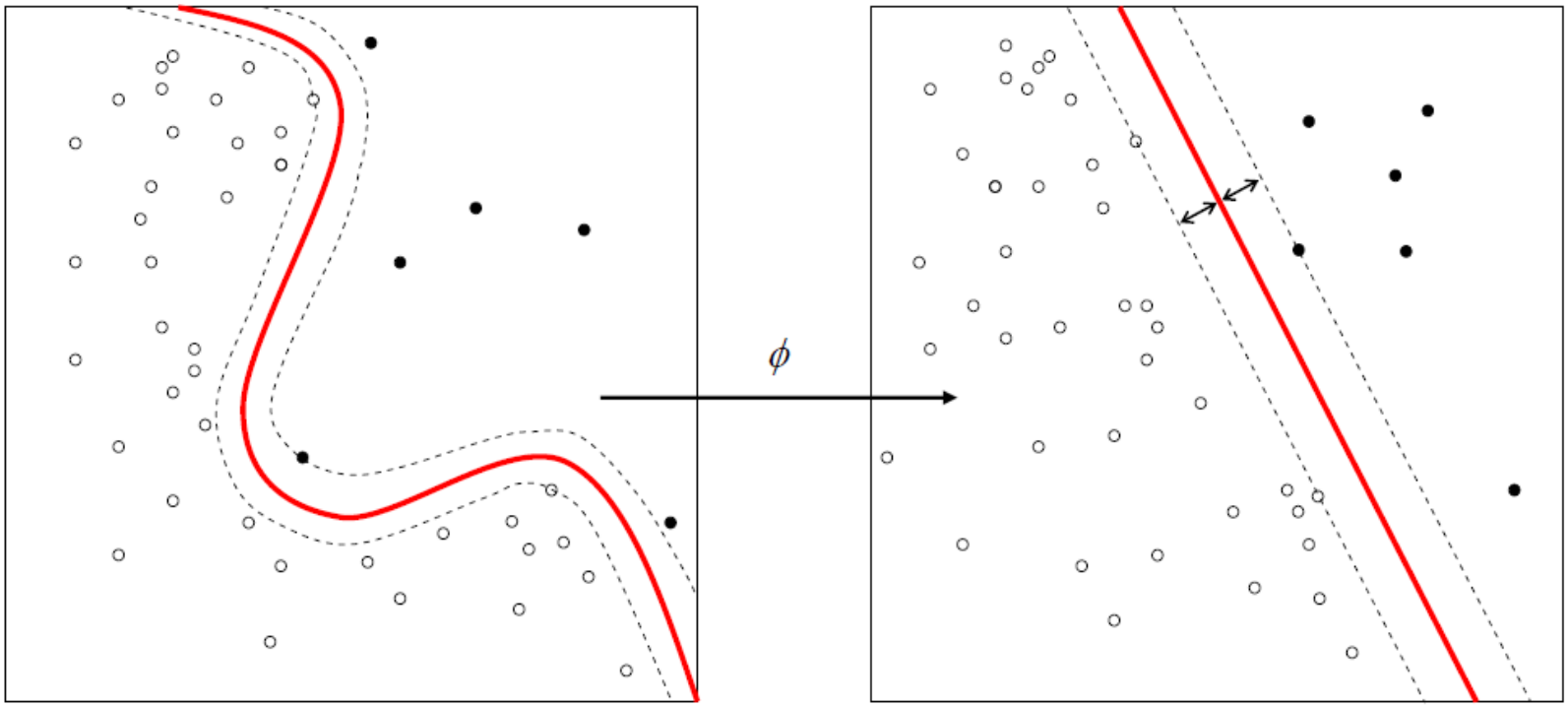
$$K(a_k, a_\ell) \sim \psi(a_k)^T\psi(a_\ell).$$

[Boser et al., 1992, Cortes and Vapnik, 1995]. "Kernel trick."

Gaussian kernels are popular:

$$K(a_k, a_\ell) = \exp(-\|a_k - a_\ell\|^2/(2\sigma)), \quad \text{for some } \sigma > 0.$$

Nonlinear SVM



Application VIII: Logistic Regression

Binary logistic regression is similar to binary SVM, except that we seek a function p that gives odds of data vector a being in class 1 or class 2, rather than making a simple prediction.

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

$$p(a; x) := (1 + e^{a^T x})^{-1}.$$

Choose x so that $p(a_j; x) \approx 1$ when $y_j = 1$ and $p(a_j; x) \approx 0$ when $y_j = 2$.

Choose x to minimize a negative log likelihood function:

$$\mathcal{L}(x) = -\frac{1}{m}\left[\sum_{y_j=2} \log(1 - p(a_j; x)) + \sum_{y_j=1} \log p(a_j; x)\right]$$

$$\text{mapping function:} \quad \phi(a; x) = \text{index of max element of } \begin{bmatrix} p(a; x) \\ 1 - p(a; x) \end{bmatrix}.$$

Sparse solutions x are interesting because the indicate which components of a_j are critical to classification. Can solve: $\min_z \mathcal{L}(z) + \lambda \|z\|_1$.

Application IX: Multiclass Logistic Regression

Have M classes instead of just 2. M can be large e.g. identify phonemes in speech, identify line outages in a power grid.

Labels $y_{j\ell} = 1$ if data point j is in class ℓ; $y_{j\ell} = 0$ otherwise; $\ell = 1, \ldots, M$.

Find subvectors $x_{[\ell]}$, $\ell = 1, 2, \ldots, M$ such that if a_j is in class k we have

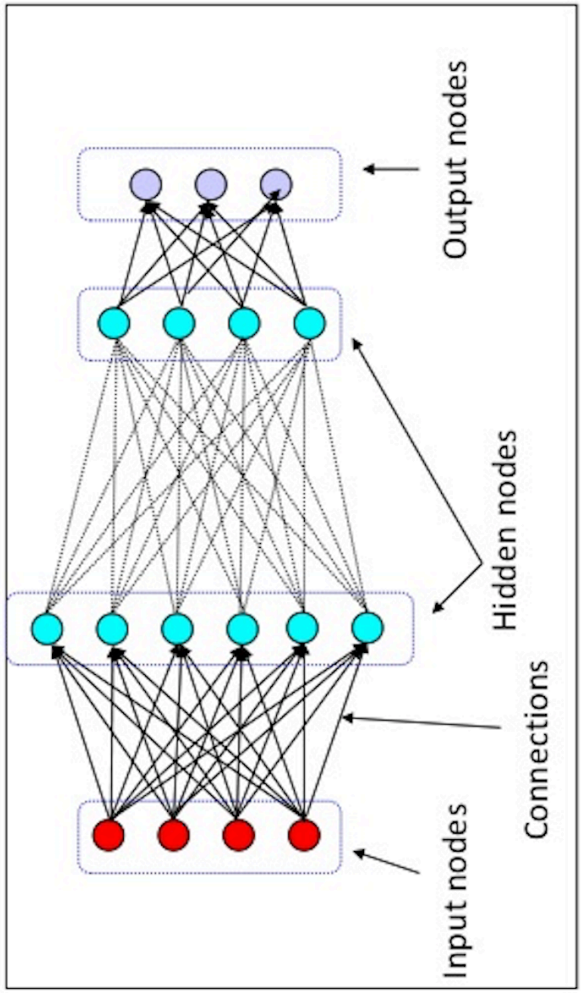
$$a_j^T x_{[k]} \gg a_j^T x_{[\ell]} \quad \text{for all } \ell \neq k.$$

Find $x_{[\ell]}$, $\ell = 1, 2, \ldots, M$ by minimizing a negative log-likelihood function:

$$f(x) = -\frac{1}{m} \sum_{j=1}^{m} \left[\sum_{\ell=1}^{M} y_{j\ell} (a_j^T x_{[\ell]}) - \log \left(\sum_{\ell=1}^{M} \exp(a_j^T x_{[\ell]}) \right) \right]$$

Can use group LASSO regularization terms to select important features from the vectors a_j, by imposing a common sparsity pattern on all $x_{[\ell]}$.

Application X: Deep Learning



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:

$$a^{\ell+1} = \sigma(W^\ell a^\ell + g^\ell),$$

where a^ℓ is node values at layer ℓ, (W^ℓ, g^ℓ) are *parameters* in the network, σ is the element-wise function.

Deep Learning

The element-wise function σ makes transformations to scalar input:

- Logistic function: $t \to 1/(1+e^{-t})$;
- Hinge: $t \to \max(t, 0)$;
- Bernoulli: random! $t \to 1$ with probability $1/(1+e^{-t})$ and $t \to 0$ 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.

Training Deep Learning Networks

The network contains many parameters — (W^ℓ, g^ℓ), $\ell = 1, 2, \ldots, L$ in the notation above — that must be selected by training on the data (a_j, y_j), $j = 1, 2, \ldots, m$.

Objective has the form:

$$\sum_{j=1}^{m} h(x; a_j, y_j)$$

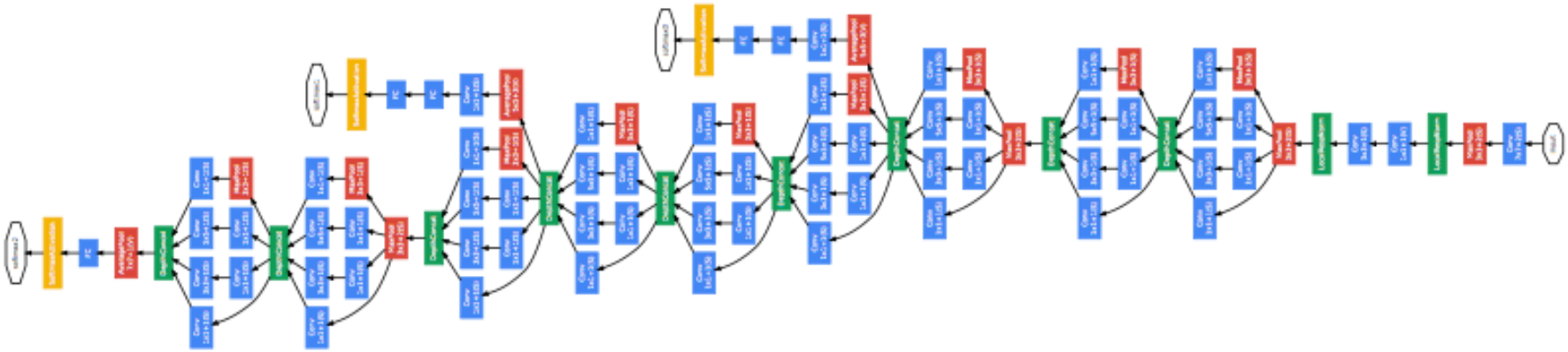
where $x = (W^1, g^1, W^2, g^2, \ldots)$ are the parameters in the model and h measures the mismatch between observed output y_j and the outputs produced by the model (as in multiclass logistic regression).

Nonlinear, Nonconvex. It's also random in some cases (but then we can work with expectation).

Composition of many simple functions.

GoogLeNet

Visual object recognition: Google's State of the Art network from 2014 [Szegedy et al., 2015].



A Basic Paradigm

Many optimization formulations in data analysis have this form:

$$\min_x \frac{1}{m} \sum_{j=1}^{m} h_j(x) + \lambda \Omega(x),$$

where

- h_j depends on parameters x of the mapping ϕ, and data items (a_j, y_j);
- Ω is the regularization term, often nonsmooth, convex, and separable in the components of x (but not always!).
- $\lambda \geq 0$ is the regularization parameter.
- (Ω could also be an indicator for simple set e.g. $x \geq 0$.)

Alternative formulation:

$$\min_x \frac{1}{m} \sum_{j=1}^{m} h_j(x) \quad \text{s.t. } \Omega(x) \leq \tau.$$

Structure in $h_j(x)$ and Ω strongly influences the choice of algorithms.

Optimization in Data Analysis: Some Background

Optimization formulations of data analysis / machine learning problems were popular, and becoming more so... but not universal:

- Heuristics are popular: k-means clustering, random forests.
- Linear algebra approaches suffice in some applications.
- Greedy feature selection.

Algorithmic developments have sometimes happened independently in the machine learning and optimization communities, e.g. stochastic gradient from 1980s-2009.

Occasional contacts in the past:

- Mangasarian solving SVM formulations [Mangasarian, 1965, Wohlberg and Mangasarian, 1990, Bennett and Mangasarian, 1992]
- Backpropagation in neural networks equivalent to gradient descent [Mangasarian and Solodov, 1994]
- Basis pursuit [Chen et al., 1998].

...but now, crossover is much more frequent and systematic.

Optimization in Learning: Context

Naive use of "standard" optimization algorithms is usually not appropriate.

Large data sets make some standard operations expensive: Evaluation of gradients, Hessians, functions, Newton steps, line searches.

The optimization formulation is viewed as a sampled, empirical approximation of some underlying "infinite" reality.

- Don't need or want an exact minimizer of the stated objective — this would be overfitting the empirical data. An approximate solution suffices.
- Choose the regularization parameter λ so that ϕ gives best performance on some "holdout" set of data: validation, tuning.

Low-Dimensional Structure

Generalizability often takes the form of low-dimensional structure, when the function is parametrized appropriately.

- Variable vector x may be sparse (feature selection, compressed sensing).
- Variable matrix X may be low-rank.
- Data objects a_j lie approximately in a low-dimensional subspace.

Example: Identify genetic risk factors for a disease by identifying the crucial locations on the genome (alleles).

- Convex formulations with regularization are often tractable and efficient in practice.
- Discrete or nonlinear formulations are more natural, but harder to solve and analyze. Recent advances make discrete formulations more appealing (Bertsimas et al.)

ALGORITHMS

Seek optimization algorithms — mostly elementary — that can exploit the structure of the formulations above.

- Full-gradient algorithms.
  - with projection and shrinking to handle Ω
- Accelerated gradient
- Stochastic gradient
  - and hybrids with full-gradient
- Coordinate descent
- Conditional gradient
- Augmented Lagrangian / ADMM

Everything Old is New Again

"Old" approaches from the optimization literature have become popular:

- Nesterov acceleration of gradient methods [Nesterov, 1983].
- Alternating direction method of multipliers (ADMM) [Eckstein and Bertsekas, 1992].
- Parallel coordinate descent and incremental gradient algorithms [Bertsekas and Tsitsiklis, 1989]
- Stochastic gradient [Robbins and Monro, 1951]
- Frank-Wolfe / conditional gradient [Frank and Wolfe, 1956, Dunn, 1979].

Many extensions have been made to these methods and their convergence analysis. Many variants and adaptations proposed.

Gradient Methods

Basic formulation, without regularization:

$$\min_x H(x) := \frac{1}{m} \sum_{j=1}^{m} h_j(x).$$

Steepest descent takes steps in the negative gradient direction:

$$x^{k+1} \leftarrow x^k - \alpha_k \nabla H(x^k).$$

Classical analysis applies for H smooth, convex, and

Lipschitz: $\|\nabla H(x) - \nabla H(z)\| \leq L\|x - z\|$, for some $L > 0$,

Lojasiewicz: $\|\nabla H(x)\|^2 \geq 2\mu[H(x) - H^*]$, for some $\mu \geq 0$.

$\mu = 0$: sublinear convergence for $\alpha_k \equiv 1/L$: $H(x^k) - H^* = O(1/k)$.
$\mu > 0$: linear convergence with $\alpha_k \equiv 1/L$:

$$H(x^k) - H^* \leq \left(1 - \frac{\mu}{L}\right)^k [H(x^0) - H^*].$$

Shrinking, Projecting for Ω

In the regularized form:

$$\min_x \; H(x) + \lambda \Omega(x),$$

the gradient step in H can be combined with a shrink operation in which the effect of Ω is accounted for exactly:

$$x^{k+1} = \arg\min_z \; \nabla H(x^k)^T (z - x^k) + \frac{1}{2\alpha_k} \|z - x_k\|^2 + \lambda \Omega(z).$$

When Ω is the indicator function for a convex set, x^{k+1} is the projection of $x^k - \alpha_k \nabla H(x^k)$ onto this set: gradient projection.

For many Ω of interest, this problem can be solved quickly (e.g. $O(n)$).

Algorithms and convergence theory for steepest descent on smooth H usually extend to this setting (for convex Ω).

Accelerated Gradient and Momentum

Accelerated gradient methods [Nesterov, 1983] highly influential.

Fundamental idea: **Momentum!** Search direction at iteration k depends on the latest gradient $\nabla H(x^k)$ and also the search direction at iteration $k-1$, which encodes gradient information from earlier iterations.

Heavy-ball & conjugate gradient (incl. nonlinear CG) also use momentum.

Heavy-Ball for $\min_x H(x)$:

$$x^{k+1} = x^k - \alpha \nabla H(x^k) + \beta(x^k - x^{k-1}).$$

Nesterov's optimal method:

$$x^{k+1} = x^k - \alpha_k \nabla H(x^k + \beta_k(x^k - x^{k-1})) + \beta_k(x^k - x^{k-1}).$$

Typically $\alpha_k \approx 1/L$ and $\beta_k \approx 1$.

Accelerated Gradient and Momentum

Accelerated gradient methods [Nesterov, 1983] highly influential.

Fundamental idea: **Momentum!** Search direction at iteration k depends on the latest gradient $\nabla H(x^k)$ and also the search direction at iteration $k-1$, which encodes gradient information from earlier iterations.

Heavy-ball & conjugate gradient (incl. nonlinear CG) also use momentum.

Heavy-Ball for $\min_x H(x)$:

$$x^{k+1} = x^k - \alpha \nabla H(x^k) + \beta(x^k - x^{k-1}).$$

Nesterov's optimal method:

$$x^{k+1} = x^k - \alpha_k \nabla H(x^k + \beta_k(x^k - x^{k-1})) + \beta_k(x^k - x^{k-1}).$$

Typically $\alpha_k \approx 1/L$ and $\beta_k \approx 1$.

Accelerated Gradient Convergence

Typical convergence:

Weakly convex $\mu = 0$: $H(x^k) - H^* = O(1/k^2)$;

Strongly convex $\mu > 0$: $H(x^k) - H^* \leq M \left(1 - \sqrt{\frac{\mu}{L}}\right)^k [H(x^0) - H^*]$.

- Approach can be extended to regularized functions $H(x) + \lambda\Omega(x)$ [Beck and Teboulle, 2009].
- Partial-gradient approaches (stochastic gradient, coordinate descent) can be accelerated in similar ways.

Are Gradient Methods Useful?

They're useful for some problems in which X is a matrix, in which full gradients are practical to compute.

- Matrix completion, including explicitly parametrized problems with $X = LR^T$ or $X = ZZ^T$; and Nonnegative matrix factorization.
- Subspace identification;
- Sparse covariance estimation;

They are less appealing when m is large. To calculate

$$\nabla H(x) = \frac{1}{m} \sum_{j=1}^{m} \nabla h_j(x),$$

generally need to make a full pass through the data.

Often not practical for massive data sets. But can be hybridized with stochastic gradient methods (see below).

Stochastic Gradient (SG)

For $H(x) = (1/m)\sum_{j=1}^{m} h_j(x)$, iteration k has the form:

- Choose $j_k \in \{1, 2, \ldots, m\}$ uniformly at random;
- Set $x^{k+1} \leftarrow x^k - \alpha_k \nabla h_{j_k}(x^k)$.

$\nabla h_{j_k}(x^k)$ is a proxy for $\nabla H(x^k)$ but it depends on just one data item a_{j_k} and is much cheaper to evaluate.

Unbiased — $\mathbb{E}_j \nabla h_j(x) = \nabla H(x)$ — but the variance may be very large.

- Average the iterates for more robust convergence:

$$\bar{x}^k = \frac{\sum_{\ell=1}^{k} \gamma_\ell x^\ell}{\sum_{\ell=0}^{k} \gamma_\ell}, \text{ where } \gamma_\ell \text{ are positive weights.}$$

- Minibatch: Use a set $J_k \subset \{1, 2, \ldots, m\}$ rather than a single item. (Smaller variance in the gradient estimate.)

See [Nemirovski et al., 2009] and many other works.

Stochastic Gradient (SG)

Convergence results for h_j convex require bounds on the variance of the gradient estimate:

$$\frac{1}{m}\sum_{j=1}^{m}\|\nabla h_j(x)\|_2^2 \leq B^2 + L_g\|x - x^*\|^2.$$

Analyze expected convergence, e.g. $\mathbb{E}(f(x^k) - f^*)$ or $\mathbb{E}(f(\bar{x}^k) - f^*)$, where the expectation is over the sequence of indices $j_0, j_1, j_2, \ldots$.

Sample results:

- H strongly convex, $\alpha_k \sim 1/k$: $\mathbb{E}(H(x^k) - H^*) = O(1/k)$;
- H weakly convex, $\alpha_k \sim 1/\sqrt{k}$: $\mathbb{E}(H(\bar{x}^k) - H^*) = O(1/\sqrt{k})$.
- $B = 0$, H strongly convex, $\alpha_k =$ const: $\mathbb{E}(\|x^k - x^*\|_2^2) = O(\rho^k)$ for some $\rho \in (0, 1)$.

Generalizes beyond finite sums, to $H(x) = \mathbb{E}_\xi f(x; \xi)$, where ξ is random.

Stochastic Gradient Applications

SG fits well the summation form of H (with large m), so has widespread applications:

- SVM (primal formulation).
- Logistic regression: binary and multiclass.
- **Deep Learning.** The Killer App! (Nonconvex) [LeCun et al., 1998]
- Subspace Identification (GROUSE): Project stochastic gradient searches onto subspace [Balzano and Wright, 2014].

Incremental Gradient (Finite Sum, Cyclic Order)

$$H(x) = \frac{1}{m} \sum_{j=1}^{m} h_j(x).$$

Steps: $x^{k+1} \leftarrow x^k - \alpha_k \nabla h_{j_k}(x^k)$, where $j_k = 1, 2, \ldots, m, 1, 2, \ldots$.

Typical requirements on step lengths α_k, for convergence:

$$\sum_{k=1}^{\infty} \alpha_k = \infty, \quad \sum_{k=1}^{\infty} \alpha_k^2 < \infty.$$

(The last condition required only for convergence of iterates $\{x^k\}$ as well as function values $\{f(x^k)\}$.)

Sublinear convergence rates proved in [Nedić and Bertsekas, 2000] for steplength $\alpha_k = C/(k+1)$.

Random Reshuffling [Gürbüzbalaban et al., 2015] (reordering order of functions between cycles) still yields sublinear convergence, but with constants reduced (sometimes dramatically) below worst-case values.

Hybrids of Full Gradient and Stochastic Gradient

Stabilize SG by hybridizing with steepest descent (full-gradient). Typically get *linear convergence* for strongly convex functions, sublinear for weakly convex.

SAG: [LeRoux et al., 2012] Maintain approximations g_j to ∇h_j, use search direction $-(1/m)\sum_{j=1}^m g_j$. At iteration k, choose j_k at random, and update $g_{j_k} = \nabla h_{j_k}(x^k)$.

SAGA: [Defazio et al., 2014] Similar to SAG, but use search direction

$$-\nabla h_{j_k}(x^k) + g_{j_k} - \frac{1}{m}\sum_{j=1}^m g_j.$$

SVRG: [Johnson and Zhang, 2013] Similar again, but periodically do a full gradient evaluation to refresh all g_j.

Too much storage, **BUT** when h_j have the "ERM" form $h_j(a_j^T x)$ (linear least squares, linear SVM), all gradients can be stored in a scalar:

$$\nabla_x h_j(a_j^T x) = a_j h_j'(a_j^T x).$$

Coordinate Descent (CD) Framework

... for smooth unconstrained minimization: $\min_x H(x)$:

Set Choose $x^1 \in \mathbb{R}^n$;
for $\ell = 0, 1, 2, \ldots$ **do**
  for $j = 1, 2, \ldots, n$ **do**
    Define $k = \ell n + j$
    Choose index $i = i(\ell, j) \in \{1, 2, \ldots, n\}$;
    Choose $\alpha_k > 0$;
    $x^{k+1} \leftarrow x^k - \alpha_k \nabla_i H(x^k) e_i$;
  end for
end for

where

- $e_i = (0, \ldots, 0, 1, 0, \ldots, 0)^T$: the ith coordinate vector;
- $\nabla_i H(x) = i$th component of the gradient $\nabla H(x)$;
- $\alpha_k > 0$ is the step length.

Economics of Partial Gradients (Least Squares, SVM)

$$H(x) = \frac{1}{m}\sum_{j=1}^{m} h_j(a_j^T x) + \lambda \sum_{i=1}^{n} \Omega_i(x_i)$$

and define A as the $m \times n$ matrix with jth row a_j^T.

Maintain $g = Ax$ and $\nabla h_j(g_j)$ for $j = 1, 2, \ldots, m$. A CD step on coordinate i_k proceeds as follows:

- Compute the i_k element of the gradient of the summation term:

$$\nabla_{i_k} H(x) = \frac{1}{m} \sum_{j: A_{j,i_k} \neq 0} A_{j,i_k} \nabla h_j(g_j);$$

- Use this information, along with $\lambda \Omega_{i_k}$, to update: $x_{i_k} \leftarrow x_{i_k} + d_{i_k}$;
- Update $g_j \leftarrow g_j + A_{j,i_k} d_{i_k}$ and $\nabla h_j(g_j)$ (only for j s.t. $A_{j,i_k} \neq 0$);

Cost: $O(\text{nonzeros in } A_{\cdot i_k})$, vs $O(\text{nonzeros in } A)$ for a full-gradient method.

Cheaper (by factor $\approx 1/n$) for one element than the full gradient.

CD Variants

- CCD (Cyclic CD): $i(\ell, j) = j$.
- RCD (Randomized CD a.k.a. Stochastic CD): $i(\ell, j)$ is chosen uniformly at random from $\{1, 2, \ldots, n\}$.
- RCCD (Randomized Cyclic CD):
  - At the start of epoch ℓ, we choose a random permutation of $\{1, 2, \ldots, n\}$, denoted by π_ℓ.
  - Index $i(\ell, j)$ is chosen to be the jth entry in π_ℓ.

Important quantities in analysis:

- $L_{\max}$: componentwise Lipschitz constant for ∇H:

$$|\nabla_i H(x + te_i) - \nabla_i H(x)| \leq L_i |t|, \quad L_{\max} = \max_{i=1,2,\ldots,n} L_i.$$

- L: usual Lipschitz constant: $|\nabla H(x + d) - \nabla H(x)| \leq L\|d\|$.
- Lojasiewicz constant μ: $\|\nabla H(x)\|^2 \geq 2\mu[H(x) - H^*]$

Randomized CD Convergence

Of the three variants, convergence of the randomized form has by far the most elementary analysis [Nesterov, 2012].

Get convergence rates for quantity $\phi_k := \mathbb{E}(H(x^k) - x^*)$.

$$\mu > 0: \quad \phi_{k+1} \leq \left(1 - \frac{\mu}{nL_{\max}}\right)\phi_k, \quad k = 1, 2, \ldots,$$

$$\mu = 0: \quad \phi_k \leq \frac{2nL_{\max}R_0^2}{k}, \qquad k = 1, 2, \ldots,$$

where R_0 bounds distance from x^0 to solution set.

If the economics of evalauting gradient components are right, this can be a factor $L/L_{\max}$ faster than full-gradient steepest descent!

This ratio is in range $[1, n]$. Maximized by $H(x) = (\mathbf{1}\mathbf{1}^T)x$.

Functions like this are good cases for RCD and RCCD, which are much faster than CCD or steepest descent.

Cyclic and Randomized Cyclic CD Convergence

Analysis of [Beck and Tetruashvili, 2013] treats CCD as an approximate form of Steepest Descent, bounding improvement in f over one cycle in terms of the gradient at the start of the cycle.

Get linear and sublinear rates that are slower than both RCD and Steepest Descent. This analysis is fairly tight — recent analysis of [Sun and Ye, 2016] confirms slow rates.

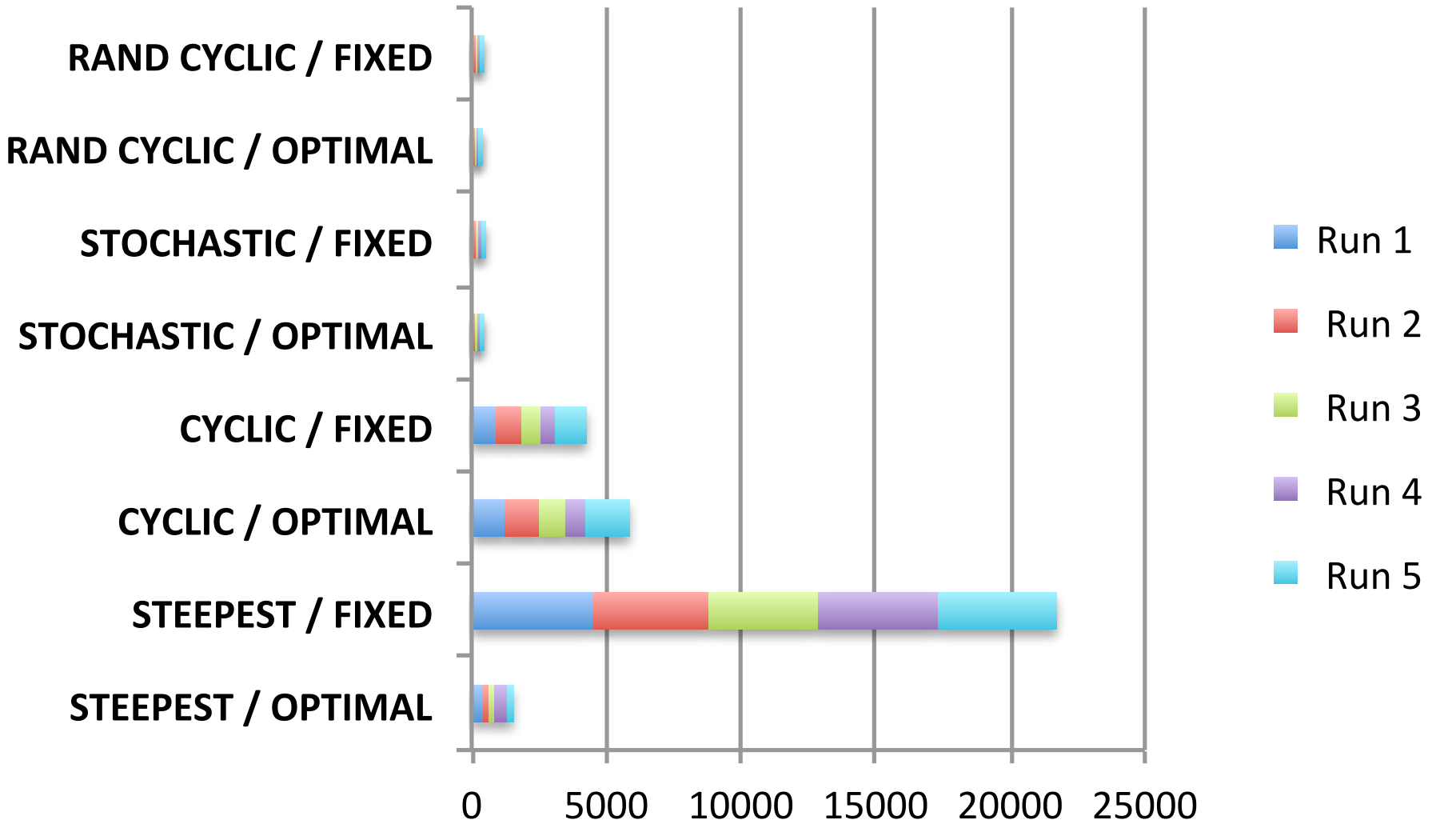
Same analysis applies to Randomized Cyclic (RCCD), but practical results for RCCD are much better, and usually at least as good as RCD. Results on quadratic function

$$H(x) = (1/2)x^T Ax$$

demonstrate this behavior (see [Wright, 2015] and my talks in 2015).

We can explain good behavior of RCCD now! [Lee and Wright, 2016]

Computations: Large $\mathbf{1}\mathbf{1}^T$ component in A ($\zeta = .3$)



$\text{cond}(A) \approx 3000$, $L_{\max} = 1$, $L \approx 50$. Steepest Descent and CCD are poor.

CD Extensions

- Block CD: Replace single component i by block $I \subset \{1, 2, \ldots, n\}$.
- Dual nonlinear SVM [Platt, 1999]. Choose *two* components of α per iteration, to stay feasible w.r.t. constraint $y^T \alpha = 0$.
- Can be accelerated (efficiently) using "Nesterov" techniques: [Nesterov, 2012, Lee and Sidford, 2013].
- Adaptable to the separable regularized case $H(x) + \lambda\Omega(x)$.
- Parallel asynchronous variants, suitable for implementation on shared-memory multicore computers, have been proposed and analyzed. [Bertsekas and Tsitsiklis, 1989, Liu and Wright, 2015, Liu et al., 2015]

Conditional Gradient / "Frank-Wolfe"

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

where f is a convex function and Ω is a closed, bounded, convex set.

Start at $x_0 \in \Omega$. At iteration k:

$$v_k := \arg\min_{v \in \Omega} v^T \nabla f(x_k);$$

$$x_{k+1} := x_k + \alpha_k (v_k - x_k), \quad \alpha_k = \frac{2}{k+2}.$$

- Potentially useful when it is easy to minimize a linear function over the *original* constraint set Ω;
- Admits an elementary convergence theory: $1/k$ sublinear rate.
- Same convergence rate holds if we use a line search for α_k.

Revived by [Jaggi, 2013].

Smooth Nonlinear Optimization: Avoiding Saddle Points

Much recent interest in methods for smooth nonlinear nonconvex problems $\min_x f(x)$ with guaranteed convergence to points satisfying approximate second-order optimality conditions:

$$\nabla f(x) = O(\epsilon^2), \quad \nabla^2 f(x) \geq -\epsilon I, \quad \text{for small } \epsilon > 0.$$

Useful for matrix optimization problems (see earlier).

Also potentially useful for deep learning. (It's believed that saddle points are much more plentiful than local minima, so methods that can escape saddle points are interesting.)

A fundamental approach: cubic regularization: Given a Lipschitz constant L on the Hessian of f, form the following upper bounding cubic approx:

$$T_L(z;x) := f(x) + \nabla f(x)^T(z-x) + \frac{1}{2}(z-x)^T\nabla^2 f(x)(z-x) + \frac{L}{6}\|z-x\|^3.$$

Basic algorithm:

$$x^{k+1} := \arg\min_z \; T_L(z;x^k).$$

Convergence to Second-Order Necessary Points

[Nesterov and Polyak, 2006]; see also [Griewank, 1981], [Cartis et al., 2011a, Cartis et al., 2011b].

For the cubic-regularized Newton method, we have

$$\|\nabla f(x^{k+1})\| = O(\|x^{k+1} - x^k\|^2), \quad \nabla^2 f(x^{k+1}) \geq -\|x^{k+1} - x^k\| I.$$

[Nesterov and Polyak, 2006, Theorem 1]: Given lower bound f^* on f and small $\epsilon > 0$, have

$$\begin{aligned} \|\nabla f(x^k)\| \leq \epsilon \qquad & \text{within } k = O(\epsilon^{-3/2}) \text{ iterations;} \\ \nabla^2 f(x^k) \geq -\epsilon I \quad & \text{within } k = O(\epsilon^{-3}) \text{ iterations,} \end{aligned}$$

where the constants in $O(\cdot)$ depend on $[f(x^0) - f^*]$ and L.

Variants and Extensions

- Trust-region methods (second-order model with constraint $\|z - x^k\|_2^2 \leq \Delta$) have similar complexity properties, when modified.
- Standard first-order method that evaluates Hessian only when $\nabla f(x^k)$ is small has slightly worse complexity.
- Extensions using third-derivative tensor can find third-order minima. Finding a fourth-order minimum is NP-hard. [Ananakumar and Ge, 2016]

But are saddle points really an issue in practice?

First-order methods are unlikely to converge to such points. An elementary application of the stable manifold theorem [Lee et al., 2016] shows that convergence of standard gradient descent converges to a strict saddle point only for x^0 in a measure-zero set. (No complexity guarantees.)

Augmented Lagrangian

Consider the linearly constrained problem,

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

where $f : \mathbb{R}^n \to \mathbb{R}$ is convex.

Define the Lagrangian function:

$$\mathcal{L}(x, \lambda) := f(x) + \lambda^T (Ax - b).$$

x^* is a solution if and only if there exists a vector of Lagrange multipliers $\lambda^* \in \mathbb{R}^m$ such that

$$-A^T \lambda^* \in \partial f(x^*), \quad Ax^* = b,$$

or equivalently:

$$0 \in \partial_x \mathcal{L}(x^*, \lambda^*), \quad \nabla_\lambda \mathcal{L}(x^*, \lambda^*) = 0.$$

Augmented Lagrangian

The augmented Lagrangian is (with $\rho > 0$)

$$\mathcal{L}(x, \lambda; \rho) := \underbrace{f(x) + \lambda^T(Ax - b)}_{\text{Lagrangian}} + \underbrace{\frac{\rho}{2}\|Ax - b\|_2^2}_{\text{“augmentation”}}.$$

Basic Augmented Lagrangian (a.k.a. method of multipliers) is

$$x_k = \arg\min_x \mathcal{L}(x, \lambda_{k-1}; \rho);$$
$$\lambda_k = \lambda_{k-1} + \rho(Ax_k - b);$$

[Hestenes, 1969, Powell, 1969]

Some constraints on x (such as $x \in \Omega$) can be handled explicitly:

$$x_k = \arg\min_{x \in \Omega} \mathcal{L}(x, \lambda_{k-1}; \rho);$$
$$\lambda_k = \lambda_{k-1} + \rho(Ax_k - b);$$

Alternating Direction Method of Multipliers (ADMM)

$$\min_{(x\in\Omega_x, z\in\Omega_z)} f(x) + h(z) \quad \text{s.t.} \quad Ax + Bz = c,$$

for which the Augmented Lagrangian is

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

Standard AL would minimize $\mathcal{L}(x, z, \lambda; \rho)$ w.r.t. (x, z) jointly. However, since coupled in the quadratic term, separability is lost.

In ADMM, minimize over x and z separately and sequentially:

$$x_k = \arg\min_{x\in\Omega_x} \mathcal{L}(x, z_{k-1}, \lambda_{k-1}; \rho);$$

$$z_k = \arg\min_{z\in\Omega_z} \mathcal{L}(x_k, z, \lambda_{k-1}; \rho);$$

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

Extremely useful framework for many data analysis / learning settings. Major references: [Eckstein and Bertsekas, 1992, Boyd et al., 2011]

ADMM for Consensus Optimization

Given the unconstrained (but separable) problem

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

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

$$\min_{x,x^1,x^2,\ldots,x^m} \sum_{i=1}^{m} f_i(x^i) \text{ subject to } x^i - x = 0,\ i = 1,2,\ldots,m.$$

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

$$\mathcal{L}(x, x^1, x^2, \ldots, x^m, \lambda^1, \ldots, \lambda^m; \rho) = \sum_{i=1}^{m} f_i(x^i) + (\lambda^i)^T(x^i - x) + \frac{\rho}{2}\|x^i - x\|_2^2.$$

The minimization w.r.t. $z = (x^1, x^2, \ldots, x^m)$ is separable!

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

Can be implemented in parallel.

Consensus, continued

The minimization w.r.t. x can be done explicitly — averaging:

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

Update to λ^i can also be done in parallel, once the new x_k is known (and communicated):

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

If the initial λ_0^i have $\sum_{i=1}^m \lambda_0^i = 0$, we have $\sum_{i=1}^m \lambda_k^i = 0$ at all iterations k. Can simplify the update for x_k:

$$x_k = \frac{1}{m} \sum_{i=1}^{m} x_k^i.$$

Not Discussed!

Many interesting topics not mentioned, including

- quasi-Newton and approximate Newton methods.
- Linear equations $Ax = b$: Kaczmarz algorithms.
- Image processing: denoising and deblurring using regularization.
- Graphs: detect structure and cliques, consensus optimization, ...
- Integer and combinatorial formulations.
- Parallel variants: synchronous and asynchronous.
- Online learning.

Conclusions: Optimization in Data Analysis

- HUGE interest across multiple communities.
- Ongoing challenges because of increasing scale and complexity of data analysis problems, and the computational platforms on which they are being solved.
- Optimization techniques are meshed with the applications.
- The optimization / data analysis / machine learning research communities are becoming "meshed" too!

FIN

References I


Ananakumar, A. and Ge, R. (2016).
Efficient approaches for escaping higher-order saddle points in nonconvex optimization.
JMLR: Workshop and Conference Proceedings, 49:1–22.


Balzano, L., Nowak, R., and Recht, B. (2010).
Online identification and tracking of subspaces from highly incomplete information.
In *48th Annual Allerton Conference on Communication, Control, and Computing*, pages 704–711.
http://arxiv.org/abs/1006.4046.


Balzano, L. and Wright, S. J. (2014).
Local convergence of an algorithm for subspace identification from partial data.
Foundations of Computational Mathematics, 14:1–36.
DOI: 10.1007/s10208-014-9227-7.


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 type methods.
SIAM Journal on Optimization, 23(4):2037–2060.

References II


Bennett, K. P. and Mangasarian, O. L. (1992).
Robust linear programming discrimination of two linearly inseparable sets.
Optimization Methods and Software, 1(1):23–34.


Bertsekas, D. P. and Tsitsiklis, J. N. (1989).
Parallel and Distributed Computation: Numerical Methods.
Prentice-Hall, Inc., Englewood Cliffs, New Jersey.


Bhojanapalli, S., Neyshabur, B., and Srebro, N. (2016).
Global optimality of local search for low-rank matrix recovery.
Technical Report arXiv:1605.07221, Toyota Technological Institute.


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.


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.


Candès, E., Li, X., and Soltanolkotabi, M. (2014).
Phase retrieval via a Wirtinger flow.
Technical Report arXiv:1407.1065, Stanford University.

References III


Cartis, C., Gould, N. I. M., and Toint, P. L. (2011a).
Adaptive cubic regularisation methods for unconstrained optimization. Part I: motivation, convergence and numerical results.
Mathematical Programming, Series A, 127:245–295.


Cartis, C., Gould, N. I. M., and Toint, P. L. (2011b).
Adaptive cubic regularisation methods for unconstrained optimization. part ii: worst-case function-and derivative-evaluation complexity.
Mathematical Programming, Series A, 130(2):295–319.


Chen, S. S., Donoho, D. L., and Saunders, M. A. (1998).
Atomic decomposition by basis pursuit.
SIAM Journal on Scientific Computing, 20(1):33–61.


Cortes, C. and Vapnik, V. N. (1995).
Support-vector networks.
Machine Learning, 20:273–297.


Dasu, T. and Johnson, T. (2003).
Exploratory Data Mining and Data Cleaning.
John Wiley & Sons.

References IV

Defazio, A., Bach, F., and Lacoste-Julien, S. (2014).
SAGA: a fast incremental gradient method with support for non-strongly composite convex objectives.
In Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N. D., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems 27*, pages 1646–1654. Curran Associates, Inc.

Dunn, J. C. (1979).
Rates of convergence for conditional gradient algorithms near singular and nonsingular extremals.
SIAM Journal on Control and Optimization, 17(2):187–211.

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.

Frank, M. and Wolfe, P. (1956).
An algorithm for quadratic programming.
Naval Research Logistics Quarterly, 3:95–110.

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

References V


Griewank, A. (1981).
The modification of Newton's method for unconstrained optimization by bounding cubic terms.
Technical Report NA/12, DAMTP, Cambridge University.


Gürbüzbalaban, M., Ozdaglar, A., and Parrilo, P. A. (2015).
Why random reshuffling beats stochastic gradient descent.
Technical Report arXiv:1510.08560, MIT.


Hestenes, M. R. (1969).
Multiplier and gradient methods.
Journal of Optimization Theory and Applications, 4:303–320.


Jaggi, M. (2013).
Revisiting Frank-Wolfe: Projection-free sparse convex optimization.
In *Proceedings of the 30th International Conference on Machine Learning*.


Johnson, R. and Zhang, T. (2013).
Accelerating stochastic gradient descent using predictive variance reduction.
In Burges, C. J. C., Bottou, L., Welling, M., Ghahramani, Z., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems 26*, pages 315–323. Curran Associates, Inc.

References VI


LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998).
Gradient-based learning applied to document recognition.
Proceedings of the IEEE, 86(11):2278–2324.


Lee, C.-p. and Wright, S. J. (2016).
Random permutations fix a worst case for cyclic coordinate descent.
Technical report, Computer Sciences Department, University of Wisconsin-Madison.


Lee, J. D., Simchowitz, M., Jordan, M. I., and Recht, B. (2016).
Gradient descent converges to minimizers.
Technical Report arXiv:1602.04915, University of California-Berkeley.


Lee, Y. T. and Sidford, A. (2013).
Efficient accelerated coordinate descent methods and faster algorihtms for solving linear systems.
In *54th Annual Symposium on Foundations of Computer Science*, pages 147–156.


LeRoux, N., Schmidt, M., and Bach, F. (2012).
A stochastic gradient methods with an exponential convergence rate for finite training sets.
In Pereira, F., Burges, C., Bottou, L., and Weinberger, K., editors, *Advances in Neural Information Processing Systems 25*, pages 2663–2671. Curran Associates, Inc.

References VII


Liu, J. and Wright, S. J. (2015).
Asynchronous stochastic coordinate descent: Parallelism and convergence properties.
SIAM Journal on Optimization, 25(1):351–376.


Liu, J., Wright, S. J., Ré, C., Bittorf, V., and Sridhar, S. (2015).
An asynchronous parallel stochastic coordinate descent algorithm.
Journal of Machine Learning Research, 16:285–322.
arXiv:1311.1873.


Mangasarian, O. L. (1965).
Linear and nonlinear separation of patterns by linear programming.
Operations Research, 13(3):444–452.


Mangasarian, O. L. and Solodov, M. V. (1994).
Serial and parallel backpropagation convergence via nonmonotone perturbed minimization.
Optimization Methods and Software, 4:103–116.


Nedić, A. and Bertsekas, D. P. (2000).
Convergence rate of incremental subgradient algorithms.
In Uryasev, S. and Pardalos, P. M., editors, *Stochastic Optimization: Algorithms and Applications*, pages 263–304. Kluwer Academic Publishers.


Nemirovski, A., Juditsky, A., Lan, G., and Shapiro, A. (2009).
Robust stochastic approximation approach to stochastic programming.
SIAM Journal on Optimization, 19(4):1574–1609.

References VIII

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

Nesterov, Y. and Polyak, B. T. (2006).
Cubic regularization of Newton method and its global performance.
Mathematical Programming, Series A, 108:177–205.

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.

Powell, M. J. D. (1969).
A method for nonlinear constraints in minimization problems.
In Fletcher, R., editor, *Optimization*, pages 283–298. Academic Press, New York.

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.

References IX


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


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.


Stigler, S. M. (1981).
Gauss and the invention of least squares.
Annals of Statistics, 9(3):465–474.


Sun, R. and Ye, Y. (2016).
Worst-case complexity of cyclic coordinate descent: $o(n^2)$ gap with randomized version.
Technical Report arXiv:1604.07130, Department of Management Science and Engineering, Stanford University, Stanford, California.


Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. (2015).
Going deeper with convolutions.
In *CVPR*.

References X

Wohlberg, W. H. and Mangasarian, O. L. (1990).
Multisurface method of pattern separation for medical diagnosis applied to breast cytology.
Proceedings of the National Academy of Sciences, 87(23):9193–9196.

Wright, S. J. (2015).
Coordinate descent algorithms.
Mathematical Programming, Series B, 151(arXiv:1502.04759):3–34.

Zheng, Q. and Lafferty, J. (2015).
A convergent gradient descent algorithm for rank minimization and semidefinite programming from random linear measurements.
Technical Report arXiv:1506.06081, Statistics Department, University of Chicago.