Sparse and Structured Optimization

Stephen Wright

University of Wisconsin-Madison

CSMP, Shanghai, May 2010

Sparse Optimization: Motivation

Look for simple approximate solution of optimization problem, rather than a (more complicated) exact solution.

- Occam's Razor: Simple explanations of the observations are preferable to complicated explanations.
- Noisy or sampled data doesn't justify solving the problem exactly. Prefer a solution that is more "generalizable" to the range of possible underlying problems and does not "overfit" this particular instance.
- Simple solutions sometimes more robust to data inexactness.
- Often easier to actuate / implement / store / explain simple solutions.
- May conform better to prior knowledge.

Often, when the solution is represented in an appropriate basis, simplicity or structure shows up as sparsity in x (i.e. few nonzero components).

Example: Compressed Sensing

Given $k \times n$ matrix A and observation vector y, find *sparse* x with

$$Ax \approx y.$$

We can reconstruct x from A and y, even when $k \ll n$ and when noise is present in y, provided:

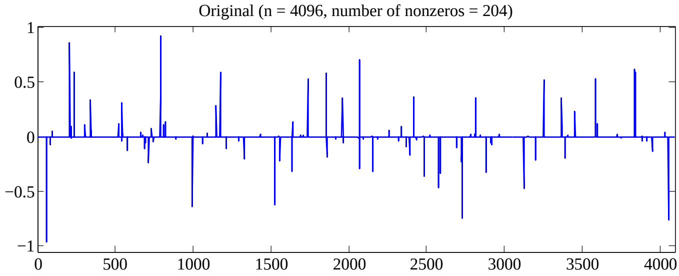
- We know that x is sparse (or nearly so);
- There are enough observations k, relative to sparsity of x;
- A satisfies restricted isometry properties (RIP) that ensure that for all sparse vectors x^1 and x^2, we have $\|A(x^1 - x^2)\|_2 \approx \|x^1 - x^2\|_2$.

If A is a projection from $\mathbb{R}^n$ onto a random k-dimensional subspace, it will have such properties. (Johnson-Lindenstrauss)

Reconstruction: Given A and y, and possibly some knowledge of sparsity level and noise type, recover x.

5% of the 4096 components are nonzero.

- Conventional signal processing indicates that you would need at least 4096 measurements (e.g. an FFT, a component-by-component sample) to determine x.
- Using compressed sensing, it can be reconstructed exactly from 1000 random linear combinations of the components of x.



Example: Image Denoising

Given a rectangular array of pixel intensities $f = [f_{ij}]$, $i, j = 1, 2, \dots, N$, find a denoised array $u = [u_{ij}]$ that is close to f but has smaller total variation (more cartoon-like).

Formulate as a data-fitting problem with a regularization term that penalizes the discrete spatial gradient of u:

$$\min_u P(u) := \frac{\lambda}{2}\|u - f\|_2^2 + \sum_{i,j} \left\| \begin{bmatrix} u_{i+1,j} - u_{i,j} \\ u_{i,j+1} - u_{i,j} \end{bmatrix} \right\|_2$$

Tends to filter out random noise in pixels of f. As λ increases, u is closer to the measured image f.

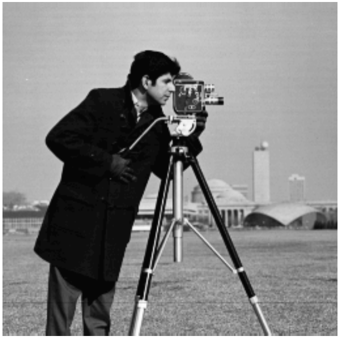
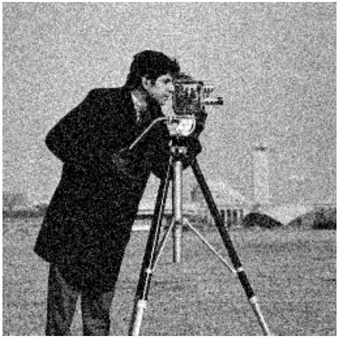



Figure: CAMERAMAN: original (left) and noisy (right)

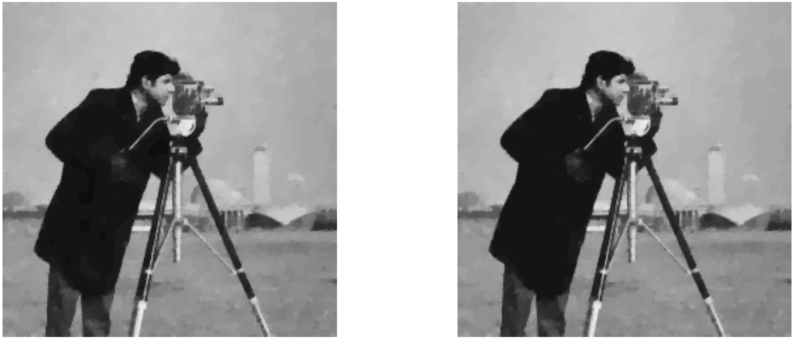


Figure: Denoised CAMERAMAN: Tol=10^{-2} (left) and Tol=10^{-4} (right).

Example: Matrix Completion

Seek low-rank matrix $X \in \mathbb{R}^{n_1 \times n_2}$ such that $X_{ij} \approx M_{ij}$ for $(i,j) \in \Omega$, where

- Ω is a set of index pairs in $\{1, 2, \dots, n_1\} \times \{1, 2, \dots, n_2\}$;
- M_{ij} are given observations.

Examples: Netflix Prize, Covariance Estimation.

More generally, seek low-rank X such that $\mathcal{A}(X) \approx b$, where $\mathcal{A}$ is a linear mapping on elements of X and b is the vector of observations.

In some sense, an extension of compressed sensing to matrix variables.

- "Simplicity" $\sim$ "low rank" rather than sparsity.
- Many algorithmic ideas extend, and new ones arise.
- Linear algebra issues are more complicated and more central.

Recent work on other forms of simplicity, e.g. low-rank + sparse.

Example: Tensor Decompositions

Given an N-dimensional tensor X, the CP decomposition expresses X approximately as an outer product of F rank-1 tensors:

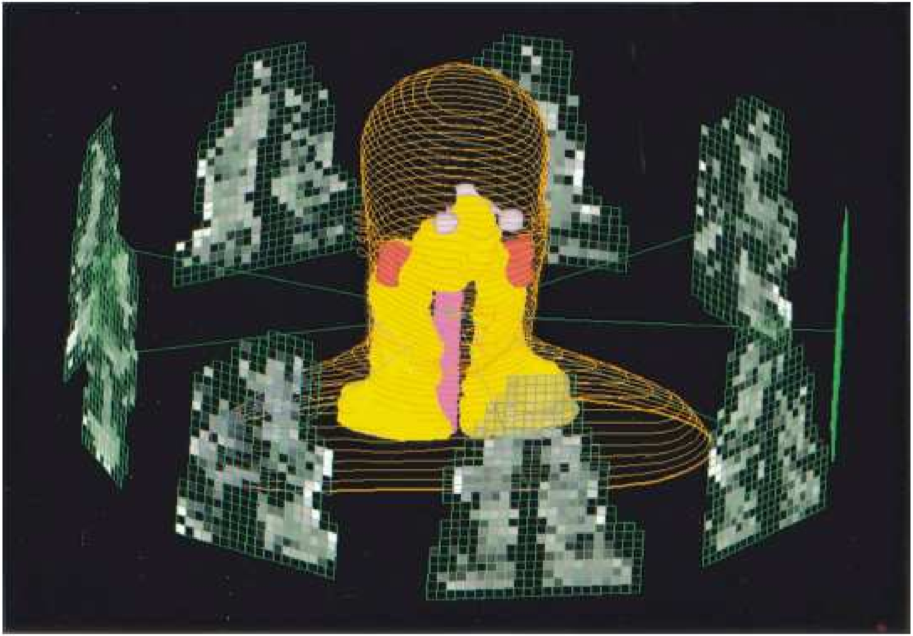
$$X_{i_1,i_2,\dots,i_N} \approx \sum_{f=1}^{F} a^{(1)}_{i_1,f} a^{(2)}_{i_2,f} \dots a^{(N)}_{i_N,f}.$$

Rank of a tensor is the smallest F for which exact equality holds. However things are much more complicated than in the matrix case ($N = 2$):

- F may be different over $\mathbb{R}$ and $\mathbb{C}$.
- Finding F is NP-hard.
- *Maximum* and *typical* ranks of random tensors may be different.
- Can have a sequence of rank-F tensors approaching a rank-$(F + 1)$ tensor.

Example: Radiotherapy for Cancer

- Deliver radiation from an external device to an internal tumor.
- Shape radiation beam, choose angles of delivery so as to deliver prescribed radiation dose to tumor while avoiding dose to surrounding tissue and organs.
- Use just a few different beam shapes and angles, from many possible choices, to simplify the treatment.
- Avoids spending too much time in setup, reduce the likelihood of treatment errors, and avoid over-optimizing to unreliable data.



Learning from Data

Learn how to make inferences from data.

Related Fields: Data Mining, Machine Learning, Support Vector Machines, Classification, Regression.

Given a (possibly huge) number of examples ("training data") and the known inferences for each data point, seek rules that can be used to make inferences about *future* instances.

Among many possible rules that explain the examples, seek simple ones.

- Provide insight into the most important features of the data: *needles in the haystack*.
- Easy and inexpensive to apply to future instances.
- More generalizable to the underlying problem - don't over-fit to the particular set of examples used.

Example: Sparse Least Squares

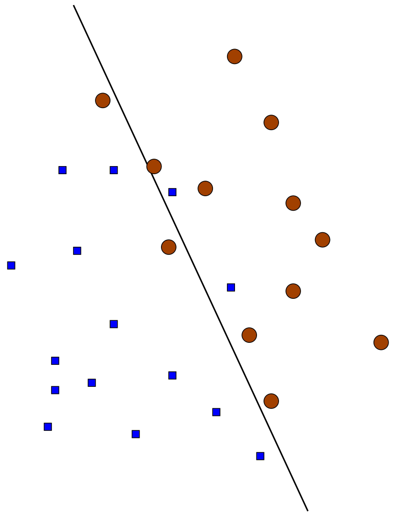
Seek a vector x with few nonzeros that approximately minimizes $\frac{1}{2}\|Ax - b\|_2^2$ for given A, b. (Variable Selection)

(LASSO: Tibshirani, 1997) Select the nonzeros by solving

$$\min_x \frac{1}{2}\|Ax - b\|_2^2 \text{ s.t. } \|x\|_1 \le T$$

for some parameter $T > 0$. (Generally, larger $T \Rightarrow$ more nonzeros in x.)

Example: Support Vector Machines (Linear)

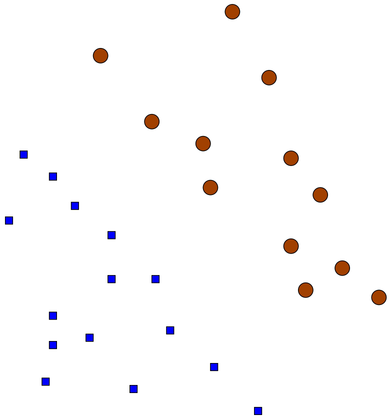


- Have feature vectors $x_1, x_2, \dots, x_n \in \mathbb{R}^m$ (real vectors) and binary labels $y_1, y_2, \dots, y_n = \pm 1$.
- Seek a hyperplane $w^T x + b$ defined by coefficients (w, b) that separates the points according to their classification:

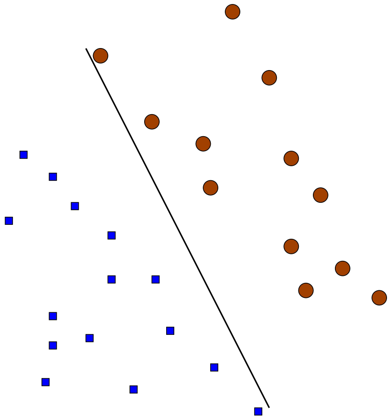
$$w^T x_i + b \geq 1 \Rightarrow y_i = 1, \qquad w^T x_i + b \leq -1 \Rightarrow y_i = -1$$

  (for most training examples $i = 1, 2, \dots, n$).
- Choose (w, b) to balance between
  - fitting this particular set of training examples,
  - ... but not over-fitting — so that it would not change much if presented with other training examples following the same (unknown) underlying distribution.

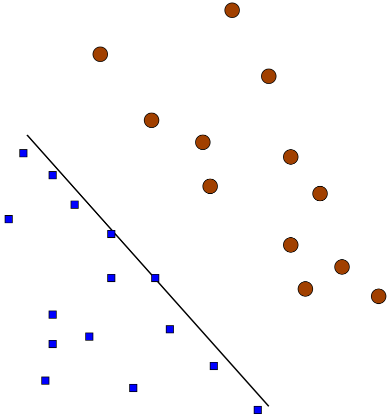
Separable Data Set: Possible Separating Planes



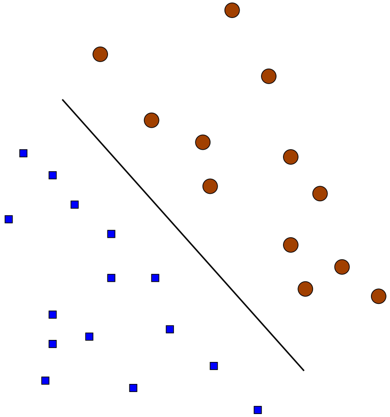
Separable Data Set: Possible Separating Planes



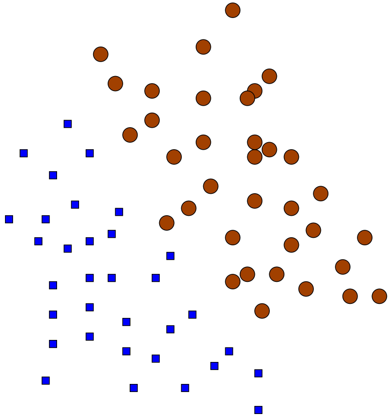
Separable Data Set: Possible Separating Planes



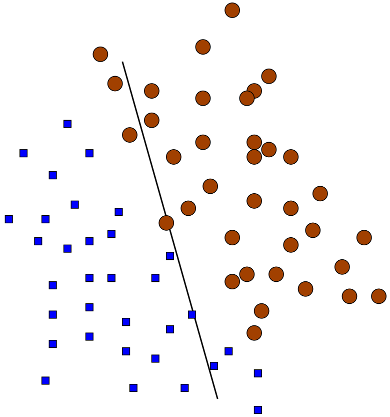
Separable Data Set: Possible Separating Planes



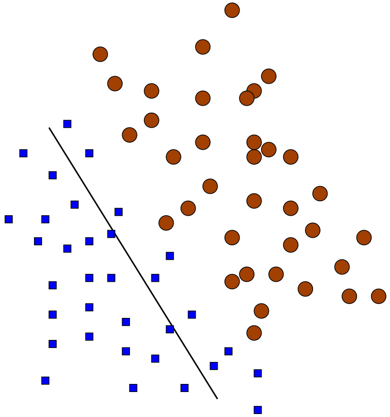
More Data Shows Max-Margin Separator is Best



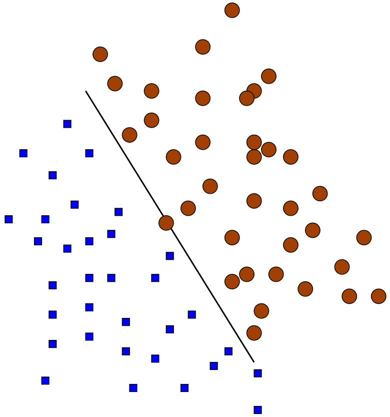
More Data Shows Max-Margin Separator is Best



More Data Shows Max-Margin Separator is Best



More Data Shows Max-Margin Separator is Best



- For separable data, find maximum-margin classifier by solving

$$\min_{(w,b)} \|w\|_2^2 \text{ s.t. } \begin{cases} w^T x_i + b_i \ge 1, & \text{if } y_i = +1 \\ w^T x_i + b_i \le -1, & \text{if } y_i = -1 \end{cases}$$

- Penalized formulation: for suitable $\lambda > 0$, solve

$$\min_{(w,b)} \frac{\lambda}{2} w^T w + \frac{1}{m} \sum_{i=1}^{m} \max\left(1 - y_i[w^T x_i + b], 0\right).$$

(Also works for non-separable data.)

- Dual formulation:

$$\max_{\alpha} e^T \alpha - \frac{1}{2} \alpha^T Y^T K Y \alpha \text{ s.t. } \alpha^T y = 0, \;\; 0 \le \alpha \le \frac{1}{\lambda m} \mathbf{1},$$

where $y = (y_1, y_2, \dots, y_m)^T$, $Y = \text{diag}(y)$, $K_{ij} = x_i^T x_j$ is the kernel.

Formulating Sparse Optimization Problems

Two basic ingredients:

- Underlying optimization problem: data-fitting or max-likelihood;
- Regularization term or constraints that promote sparsity or structure.

Many tools and techiques needed:

- Large-scale optimization: optimal first-order, gradient projection, second-order, continuation, coordinate relaxation, interior-point, ...
- Nonsmooth optimization: cutting planes, subgradient methods, successive approximation, ...
- Duality
- Numerical linear algebra
- Stochastics: Sampling
- Heuristics

Also a LOT of domain-specific knowledge: Problem and Context.

Convex Reformulations

Imposing the desired structured explicitly often leads to an intractable optimization problem. But in many cases we can define convex formulation that yields a useful solution.

Under some assumptions, it can even be shown that the convex formulation exactly solves the original problem.

Example:

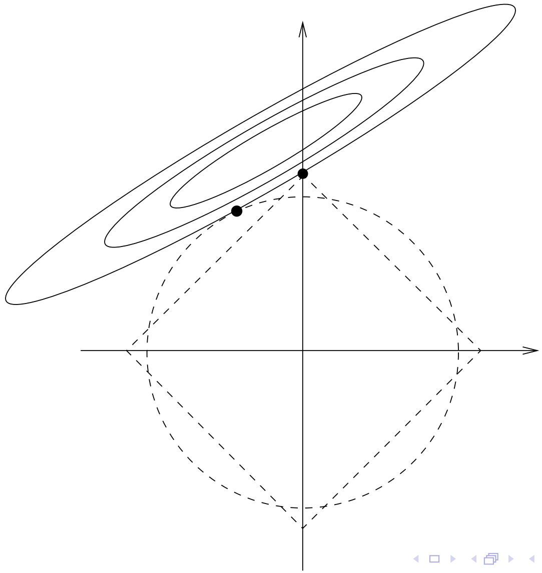
$$\min \frac{1}{2}\|Ax-b\|_2^2 \text{ subject to } \operatorname{card}(x) \le K,$$

where card(x) is the cardinality (number of nonzeros) in x.

The set of vectors satisfying card$(x) \le K$ is nonconvex. (e.g. when $K = 1$, it is the principal axes.) Could replace it by convex constraints:

$$\|x\|_1 \le T.$$

Norm Constraints and Sparse Solutions



Formulation

Given objective $f(x)$ and regularizer $P(x)$, can formulate in different ways:

$$\begin{array}{rl} I. & \min f(x) \text{ s.t. } P(x) \le T_P, \\ II. & \min P(x) \text{ s.t. } f(x) \le T_f, \\ III. & \min f(x) + \lambda P(x), \end{array}$$

for given parameters T_P, T_f, and λ.

Closely related formulations. Often want solutions for a grid or range of parameters, not just one in isolation.

Many variations, e.g. weighting and multiple "tuning" parameters λ.

Designing Algorithms

Consider for example the weighted formulation III:

$$\min f(x) + \lambda P(x).$$

Typically f is smooth and P is nonsmooth with simple structure. (Important exception: Support Vector Classification.)

Q: Can a general algorithmic framework be proposed that's efficient on many specific problems?

A: Yes and No. Some formulation / algorithm tools are useful for a variety of interesting f and P. But for practicality, need to customize carefully to the application and its requirements.

One Useful Tool

Considering the formulation $f(x) + \lambda P(x)$, many algorithms generate steps by solving prox-linear subproblems of the form:

$$\min_d g^T d + \frac{\alpha}{2} d^T d + \lambda P(x + d),$$

to obtain a step d. Useful when this subproblem is trivial to solve — as happens for many P. e.g. when $P(\cdot) = \| \cdot \|_1$, can solve in $O(n)$.

- g can be $\nabla f(x)$, or some averaged or sampled gradient approx.
- When P is null, reduces to $d = -(1/\alpha)g$. When P is indicator function for a convex set, reduces to gradient projection.
- Can use a more general second-order term — $(1/2)d^T \nabla^2 f(x) d$, or a quasi-Newton approximation — but harder to solve.
- Get block coordinate relaxation variant by setting $d_i = 0$ for most i.

Used in compressed sensing, logistic regression, matrix completion.

Compressed Sensing

Seek x with few nonzeros such that $Ax \approx y$, where $A \in \mathbb{R}^{n\times k}$, $n \ll k$.

Given sparsity level $S \le k$, A satisfies RIP with isometry constant $\delta_S < 1$ if for any column submatrix $A_{\cdot T}$ of A with at most S columns, we have

$$(1-\delta_S)\|c\|_2^2 \le \|A_{\cdot T}c\|_2^2 \le (1+\delta_S)\|c\|_2^2, \qquad \text{for all } c \in \mathbb{R}^S.$$

That is, every column submatrix with S columns is nearly orthonormal.

If δ_{2S} is somewhat less than 1, then A can distinguish clearly between any two vectors in $\mathbb{R}^n$ with sparsity level S or below.

Random matrices often have good RIP:

- elements of A drawn i.i.d. from $N(0,1)$;
- columns of A uniformly distributed on the unit sphere in $\mathbb{R}^k$;
- row submatrix of discrete cosine transform.

RIP and Convex Formulations

Candes, Romberg, Tao (2004-2005) and Donoho (2004) show that if there is an underlying solution x^* with fewer than S nonzeros, it can be recovered by using $\| \cdot \|_1$ as a surrogate for cardinality and solving

$$\min \tfrac{1}{2}\|Ax - y\|_2^2 + \lambda\|x\|_1$$

for some $\lambda > 0$, provided δ_{2S} is small.

- When y has no noise, can solve instead $\min \|x\|_1$ s.t. $Ax = y$ to get x^* *exactly*.
- When i.i.d. noise present in y, recover x^* to within accuracy allowed by the noise from the weighted formulation above.

Issues

Many algorithms have been proposed. Besides sparsity of x, the problem has several properties that drive algorithmic choices:

- n very large, possibly also k.
- A often dense, can't store substantial submatrices explicitly (but a small column submatrix may be OK).
- Can often do fast matrix-vector multiplications with A and A^T.
- Often want to solve for a selection of λ values; best value not known a priori.

Can we save in practice by operating with just *submatrices* of A and A^T? Can we use *inexpensive approximations* to these matrices?

For the random matrices that have been the focus of most analysis and testing, probably not. However, in practical situations, such things may be possible (and may lead to customized algorithms).

Algorithms

- Interior-point: Primal-dual: SparseLab / PDCO [Saunders et al 98, 02], `l1_ls` [Kim et al 07]. SOCP: ℓ_1-magic [Candès, Romberg 05]
- Gradient projection on QP formulation: GPSR [Figueiredo, Nowak, Wright 07].
- Pivoting / Homotopy a la LARS: SparseLab / SolveLasso
- Iterative shrinking-thresholding / Forward-backward splitting / Fixed-point: [Daubechies, Defriese, DeMol 04], [Combettes, Wajs 05], FPC [Hale, Yin, Zhang 07], SpaRSA [Wright, Figueiredo, Nowak 08].
- Augmented Lagrangian / Bregman [Yin et al 08], SALSA [Afonso et al 09]
- Matching pursuit: OMP [Pati, Rezaiifar, Krishnaprasad 93] [Davis, Mallat, Avellaneda 97], CoSaMP [Needell, Tropp 08].
- Optimal first-order: [Nesterov 07], FISTA [Beck, Teboulle 08].
- Randomized first-order [Juditsky, Karzan, Nemirovski 10].
- Message passing [Donoho, Maleki, Montanari 09]

SpaRSA

$$\min \phi(x) := \frac{1}{2}\|Ax - y\|_2^2 + \lambda\|x\|_1.$$

Follows the prox-linear framework defined earlier. Define $q(x) := (1/2)\|Ax - y\|_2^2$. From iterate x^k, solve:

$$\min_d \nabla q(x^k)^T d + \frac{1}{2}\alpha_k d^T d + \lambda\|x^k + d\|_1.$$

Can view the α_k term as an approximation to the Hessian: $\alpha_k I \approx A^T A$.

(RIP $\Rightarrow$ approximate the *projected* Hessian well, on *small* sets of nonzero components.)

Solve in $O(n)$ operations using the shrinkage operator. Define $z := x^k + d$ and solve

$$\min_z \frac{1}{2}\|z - u^k\|_2^2 + \frac{\lambda}{\alpha_k}\|z\|_1, \qquad \text{with } u^k := x^k - \frac{1}{\alpha_k}\nabla q(x^k).$$

SpaRSA Details

- By choosing $\alpha_k \geq \bar{\alpha} := \frac{1}{2}\|A\|_2^2$ for all k, can guarantee convergence, but slowly.
- Can use adaptive choices of α_k to accelerate convergence. Increase / decrease α_k to get descent at each iteration.
- Nonmonotone stategies based on Barzilai-Borwein also useful: choose α_k to capture curvature of Hessian $A^T A$ over the step just taken. Cyclic variants update α_k e.g. on only every 3rd iteration.
- Each iterations requires one multiplication each with A and A^T.

Problems are harder to solve for small λ (less regularization, more nonzeros in x). Use continuation:

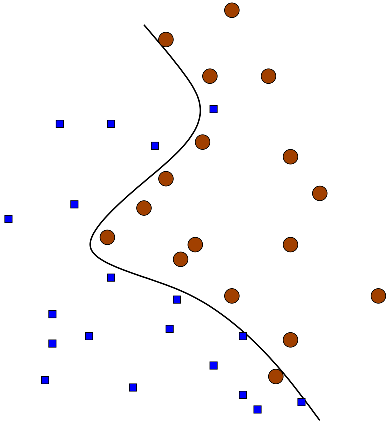
- Choose initial $\lambda_0 \leq \|A^T y\|_\infty$ and
- Solve for a decreasing sequence of λ_i, using previous solution as warm start, ending at the "target" value of λ.
- Greatly improves performance, but not well understood.

Codes for SpaRSA, GPSR also available through my web page.

For suitable matrices A (e.g. random rows of a discrete cosine transform), SpaRSA can be implemented very efficiently on a graphical processing unit (GPU). Code available from
`www.cs.wisc.edu/~swright/GPUreconstruction/`

For more information on compressed sensing (theory, algorithms, applications) see `http://dsp.rice.edu/cs`

Nonlinear Support Vector Machines



Nonlinear Separators

Reminder of the linear classification problem:

$$\text{Primal:} \qquad \min_{(w,b)} \frac{\lambda}{2} w^T w + \sum_{i=1}^{m} \max\left(1 - y_i[w^T x_i - b], 0\right).$$

$$\text{Dual:} \qquad \max_{\alpha} e^T \alpha - \frac{1}{2} \alpha^T Y^T K Y \alpha \text{ s.t. } \alpha^T y = 0, \ \ 0 \le \alpha \le \frac{1}{\lambda} \mathbf{1},$$

where $y = (y_1, y_2, \dots, y_m)^T$, $Y = \text{diag}(y)$, $K_{ij} = x_i^T x_j$.

To get a *nonlinear* classifier, map x into a higher-dimensional space $\phi : \mathbb{R}^n \to \mathcal{H}$, and do linear classification in $\mathcal{H}$ to find $w \in \mathcal{H}$, $b \in \mathbb{R}$.

When the hyperplane is projected back into $\mathbb{R}^n$, gives a nonlinear surface (often not contiguous).

Primal Formulation

In "lifted" space, primal problem is

$$\min_{(w,b)} \frac{\lambda}{2} w^T w + \sum_{i=1}^{m} \max\left(1 - y_i[w^T \phi(x_i) + b], 0\right).$$

By optimality conditions (and a representation theorem), optimal w has the form

$$w = \sum_{i=1}^{m} \alpha_i y_i \phi(x_i).$$

Hence we can recast as a finite-dimensional problem in $(\alpha, b) \in \mathbb{R}^{m+1}$:

$$\min_{\alpha,b} \frac{\lambda}{2} \alpha^T \Psi \alpha + \frac{1}{m} \sum_{i=1}^{m} \max\left(1 - \Psi_{i\cdot}\alpha - y_i b, 0\right),$$

where $\Psi_{ij} = y_i y_j \phi(x_i)^T \phi(x_j)$. WLOG can impose bounds $\alpha_i \in [0, 1/(\lambda m)]$.

Don't need to define ϕ explicitly! Instead define the kernel function $k(s,t)$ to measure distance between s and t in $\mathcal{H}$. Implicitly, $k(s,t) = \langle \phi(s), \phi(t) \rangle$.

Usually use the Gaussian kernel $k^G(s,t) := \exp(-\|s-t\|_2^2/(2\sigma^2))$.

Thus define $\Psi_{ij} = y_i y_j k(x_i, x_j)$ in the problem above.

Given a solution (α, b) we can classify a new point x by evaluating

$$\sum_{i=1}^{m} \alpha_i y_i k(x, x_i) + b,$$

and checking whether it is positive (thus classified as $+1$) or negative (class -1).

Difficulty: Ψ is generally large ($m \times m$) and dense. Specialized techniques needed to solve the classification problem for (α, b).

Approximate Kernel

Here we propose an algorithm that replaces Ψ by a low-rank approximation and then uses stochastic approximation to solve it.

Using a Nystrom method [Drineas & Mahoney 05], choose c indices from $\{1, 2, \dots, m\}$ and evaluate those rows/columns of Ψ. By factoring this submatrix, can construct a rank-r approximation $\Psi = VV^T$, where $V \in \mathbb{R}^{m \times r}$ (with $r \le c$).

Replace $\Psi \leftarrow VV^T$ in the problem and change variables $\gamma = V^T\alpha$, to get

$$\min_{(\gamma,b)} \frac{\lambda}{2}\gamma^T\gamma + \frac{1}{m}\sum_{i=1}^{m} \max\left(1 - v_i^T\gamma - y_i b, 0\right),$$

where v_i^T is the ith row of V.

Same form as linear SVM, with feature vectors $y_i v_i$, $i = 1, 2, \dots, m$.

Stochastic Approximation

Can use any linear SVM method to solve it. We use Stochastic approximation (e.g. [Nemirovski et al 09]) or incremental subgradient.

Basic step at iteration k:

- Choose index $i_k \in \{1, 2, \ldots, m\}$;
- Choose steplength $\eta_k > 0$ and take step:

$$\begin{bmatrix} \gamma_{k+1} \\ b_{k+1} \end{bmatrix} \leftarrow \begin{bmatrix} \gamma_k \\ b_k \end{bmatrix} - \eta_k \begin{bmatrix} \lambda \gamma_k + d_k v_{i_k} \\ d_k y_{i_k} \end{bmatrix},$$

where $d_k = -1$ if $1 - v_{i_k}^T \gamma - y_{i_k} b > 0$ and $d_k = 0$ otherwise.

(These techniques proposed for linear SVM in machine learning community by Bottou, Srebro and others.)

Steplengths and Averaging

When intercept b is omitted, objective is strongly convex with modulus λ. Use steplengths $\eta_k = 1/(\lambda k)$ to get convergence in expectation with rate $1/k$:

$$E\left[f(\gamma_k) - f(\gamma^*)\right] \le \frac{Q}{k},$$

for some Q depending on $\|\gamma_0 - \gamma^*\|$, λ.

When b is present, the problem is only weakly convex. Here use steplengths of the form $\alpha_k = \theta/\sqrt{k}$ for some $\theta > 0$, and form a *weighted average* of the iterates $\{(\gamma_k, b_k)\}$ to get an estimate of (γ^*, b^*). The function value of this estimate converges like $1/\sqrt{k}$.

A Sparse Classifier

Cost of performing classification of new data with kernel machines is often overlooked. For this method, the solution (γ, b) can be used to recover a "sparse," inexpensive approximate classifier.

The "true" classifier would be $\sum_{i=1}^{m} \alpha_i y_i k_{\text{approx}}(x_i, x) + b$ for the approximate kernel — which is unattainable for general x.

- Using the *original* kernel: $\sum_{i=1}^{m} \alpha_i y_i k(x_i, x) + b$.
- Choose α to be a solution of $V^T \alpha = \gamma$ with *just r nonzeros*.

Then need just r kernel evaluations to evaluate the classifier for general x.

Details (including computational tests) appear in

LW10 S. Lee and S. Wright, "Sparse nonlinear support vector machines via stochastic approximation," Technical Report, Feb. 2010.

Conclusions

- Sparse optimization draws on many areas of optimization, linear algebra, and statistics as well as the underlying application areas.
- There is some commonality across different areas that can be abstracted and analyzed.
- Much interdisciplinary work remains to be done on individual application areas.