Optimization Algorithms for Data Analysis

Stephen Wright

University of Wisconsin-Madison

Fields Institute, June 2010

Introduction: Data Analysis

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*.
- Simple rules are inexpensive to apply to new instances.
- Simple rules can be more generalizable to the underlying problem - don't over-fit to the particular set of examples used.
- Need to setting parameters that trade off between data fitting and generalizability (tuning/validation data useful).

Important Tool: Sparse Optimization

Optimization has been a key technology in data analysis for many years. (Least squares, robust regression, support vector machines.)

The need for simple, approximate solutions that draw essential insights from large data sets motivates *sparse* optimization.

In sparse optimization, we look for a 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. Simple solutions sometimes more robust to data inexactness.
- Often easier to actuate / implement / store / explain simple solutions.
- May conform better to prior knowledge.

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

Optimization Tools Needed

Biological and biomedical applications use many tools from large-scale optimization: quadratic programming, integer programming, semidefinite programming.

The extreme scale motivates the use of other tools too, e.g. stochastic gradient methods.

Sparsity requires additional algorithmic tools. (It often introduces structured nonsmooth functions into the objective or constraints.)

Effectiveness depends critically on exploiting the structure of the application class.

This Talk

We discuss sparse optimization and other optimization techniques relevant to problems in biological and medical sciences.

1. Optimization in classification (SVM); sparse optimization in sparse classification.
2. Regularized logistic regression.
3. Tensor decompositions for multiway data arrays.
4. Cancer treatment planning.
5. Semidefinite programming for cluster analysis.
6. Integer programming for genetically optimal captive breeding programs.

(More time for some topics than others!)

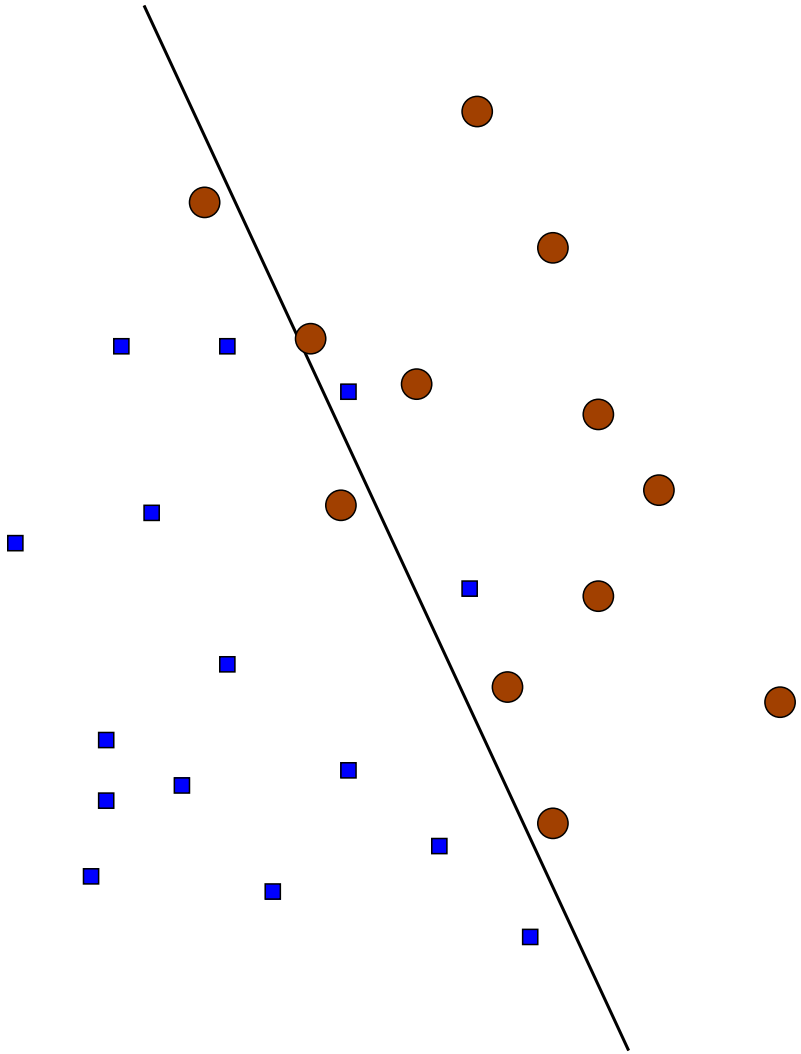
1. Optimization in Classification

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

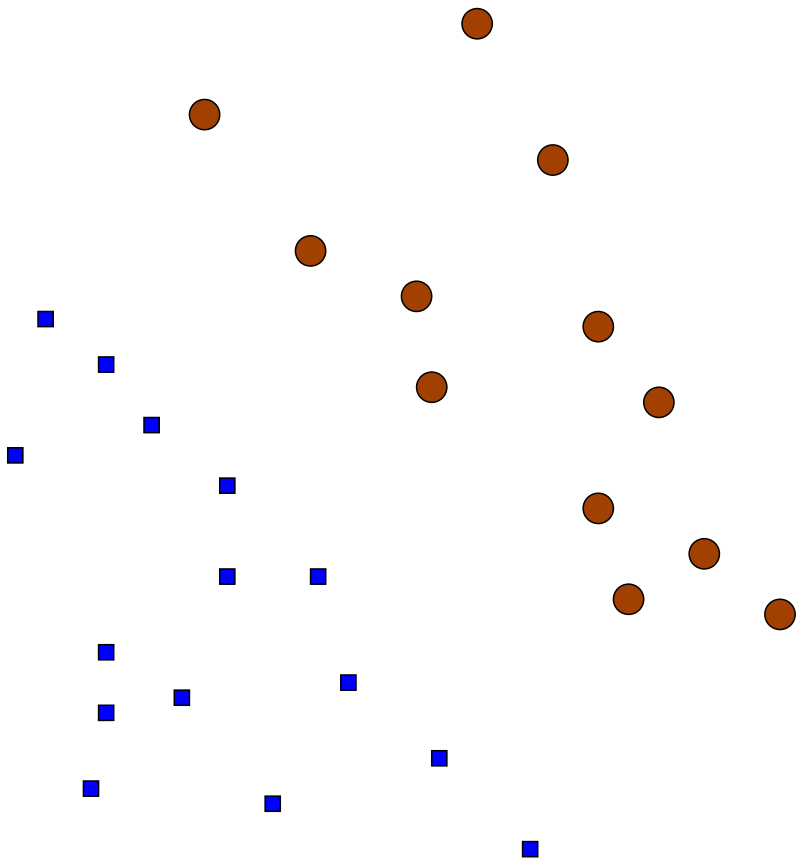
$$w^T x_i + b \geq 1 \Rightarrow y_i = 1, \qquad w^T x_i + b \leq -1 \Rightarrow y_i = -1$$

  for most training examples $i = 1, 2, \ldots, n$.
- 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.

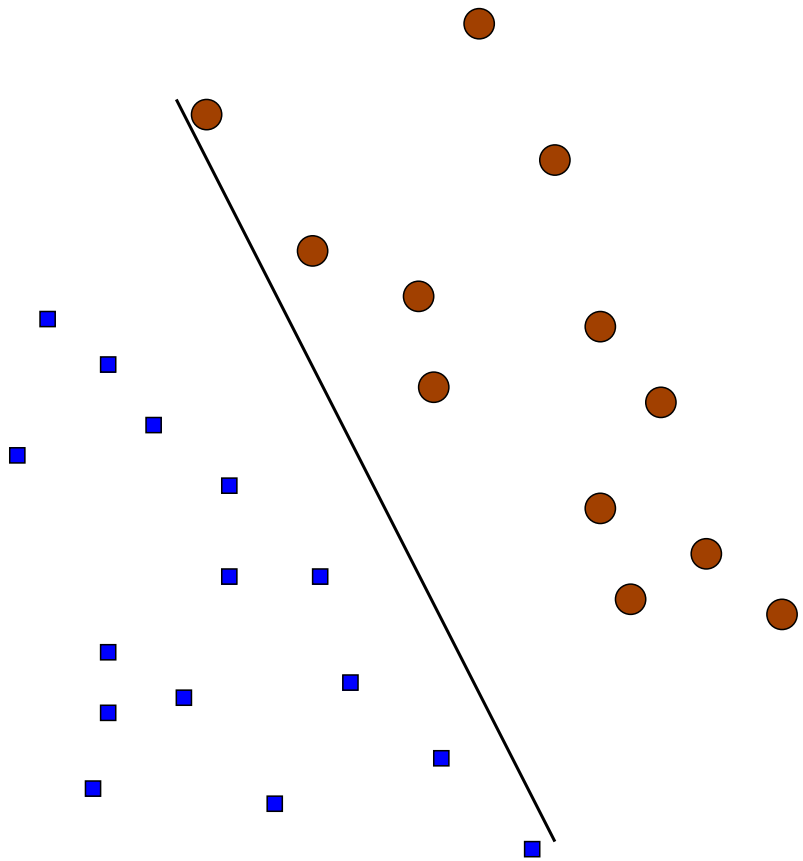
Linear SVM Classifier



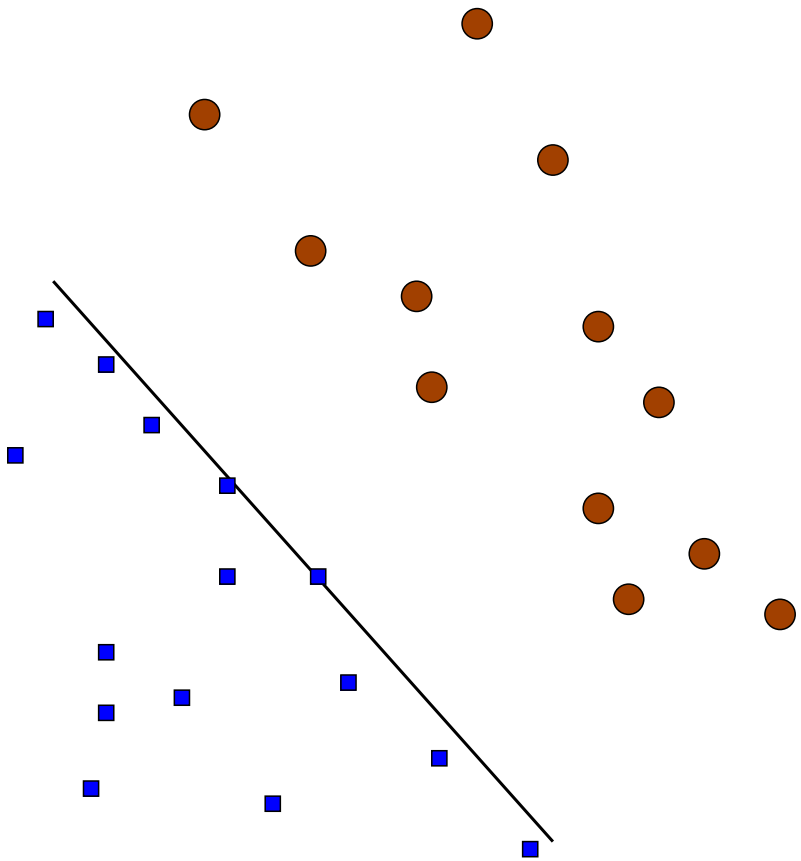
Separable Data Set: Possible Separating Planes



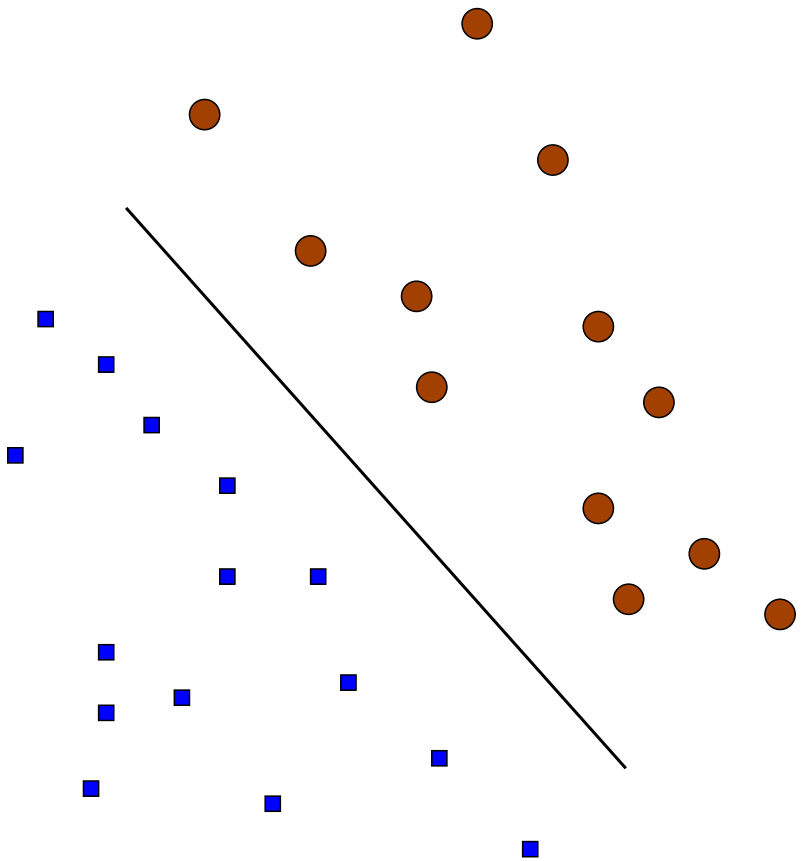
Separable Data Set: Possible Separating Planes



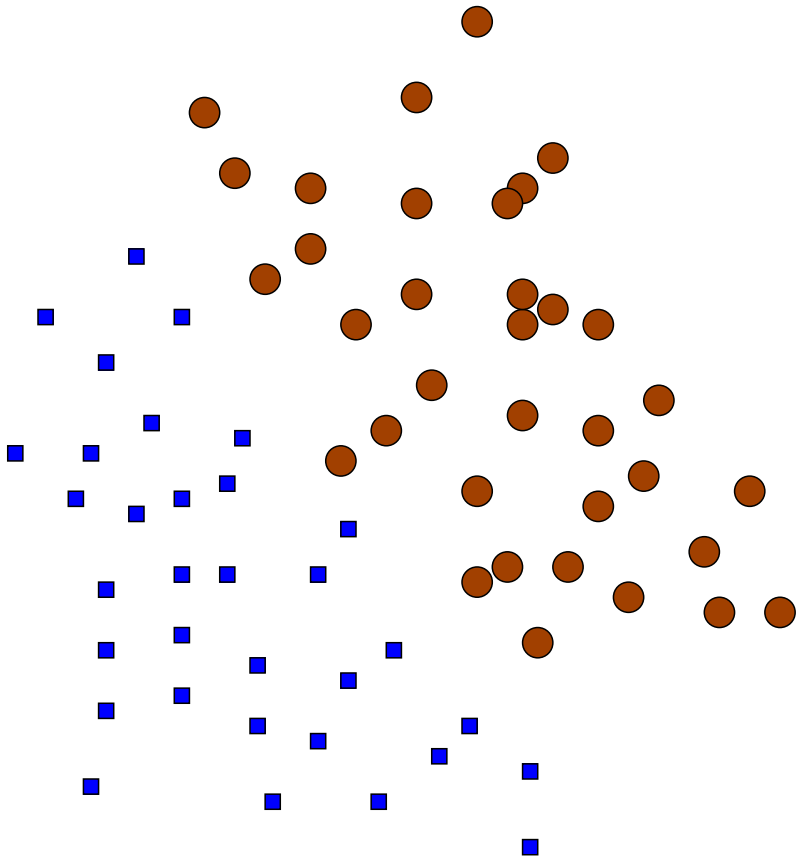
Separable Data Set: Possible Separating Planes



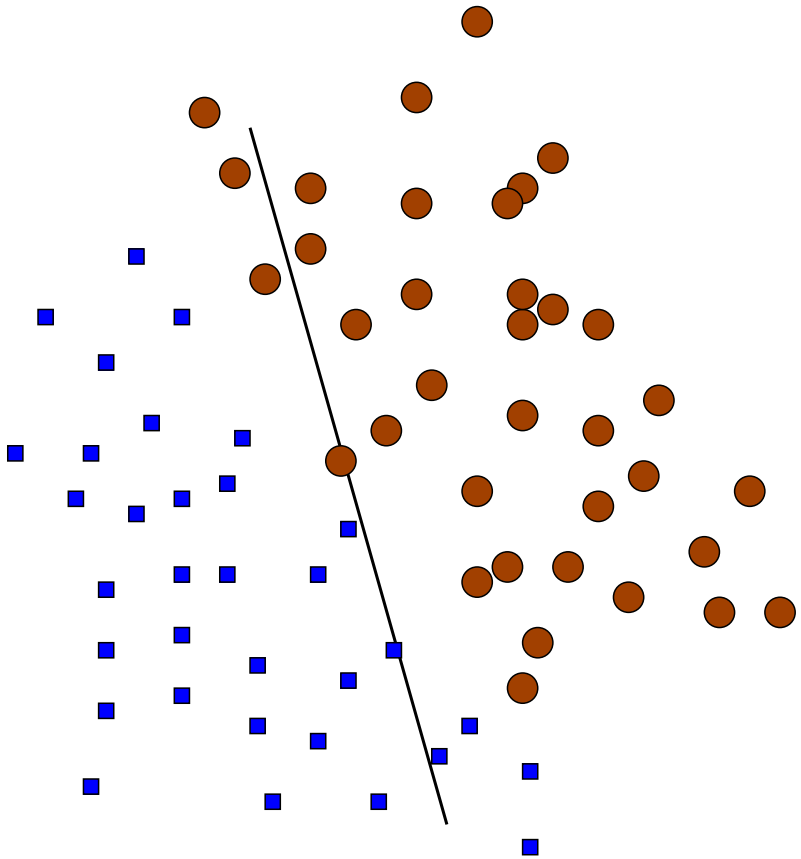
Separable Data Set: Possible Separating Planes



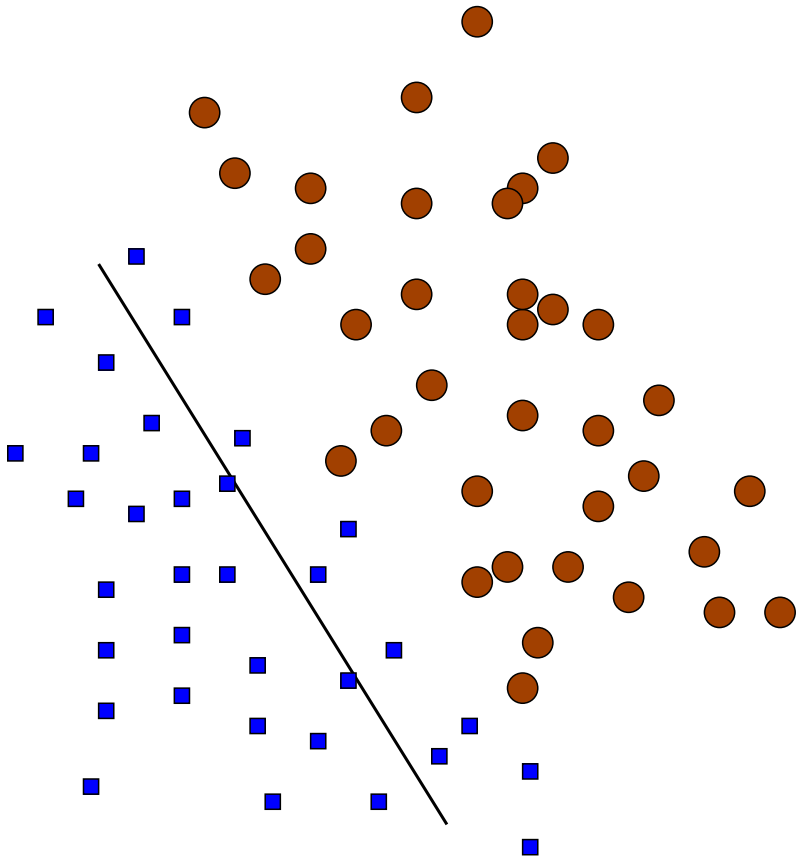
More Data Shows Max-Margin Separator is Best



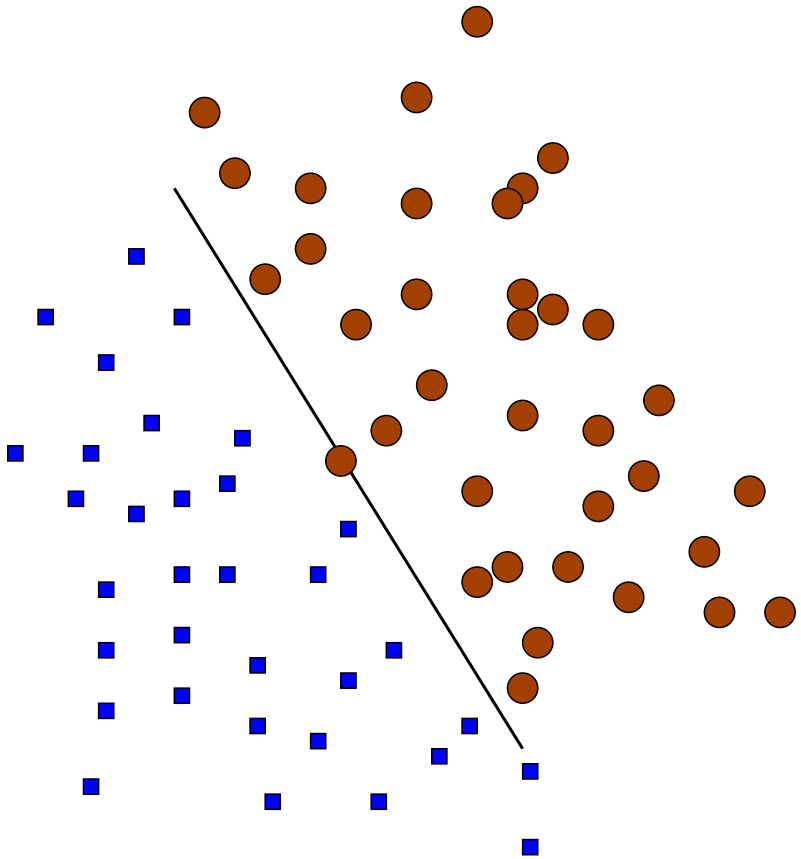
More Data Shows Max-Margin Separator is Best



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

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

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

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

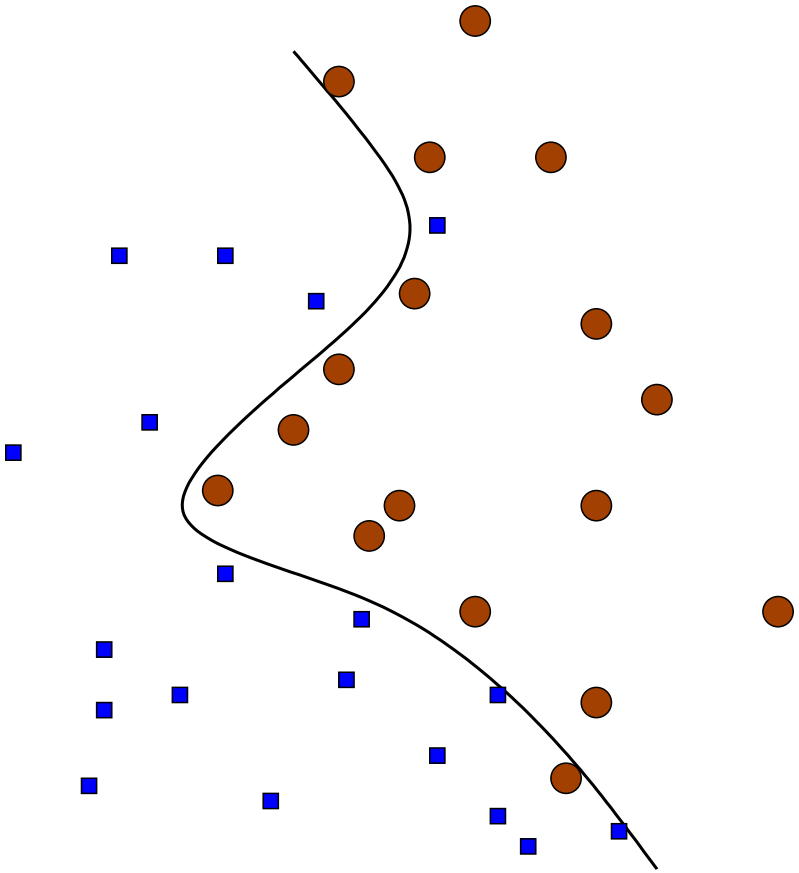
(Also works for non-separable data.)

- Dual formulation:

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

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

Nonlinear Support Vector Machines



Nonlinear SVM

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).

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

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

Kernel

By substitution, obtain 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 indicate distance between s and t in $\mathcal{H}$.

Implicitly, $k(s,t) = \langle \phi(s), \phi(t) \rangle$.

The Gaussian kernel $k^G(s,t) := \exp(-\|s-t\|_2^2/(2\sigma^2))$ is popular.

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

The Classifier

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).

Difficulties: Ψ is generally large ($m \times m$) and dense. Specialized techniques needed to solve the classification problem for (α, b). Classifier can be expensive to apply (it requires m kernel evaluations).

Many specialized algorithms proposed since about 1998, drawing heavily on optimization, but also exploiting the structure heavily.

Approximate Kernel

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 $\{1, 2, \ldots, m\}$ and evaluate those rows/columns of Ψ. By factoring this submatrix, can construct a rank-r approximation $\Psi \approx VV^T$, where $V \in \mathbb{R}^{m \times r}$ (with $r \leq c$).

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 $y_i v_i$, $i = 1, 2, \ldots, m$.

Stochastic Approximation

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

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. The step vector is an unbiased estimate of the subgradient.

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

Similar to incremental subgradient developed by Bertsekas and collaborators for objectives of the form $\sum_{i=1}^m f_i(x)$.

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 $1/k$:

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

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 $\eta_k = \theta/\sqrt{k}$ for some $\theta > 0$, and form a *weighted average* of the iterates $\{(\gamma_k, b_k)\}$.

The function value of this weighted average 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. This is unattainable for general x, as we don't know the kernel k_{approx} that corresponds to the approximate kernel matrix VV^T.

- Use instead 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.

2. Regularized Logistic Regression

Given n feature vectors $x_i \in \mathbb{R}^m$, $i = 1, 2, \ldots, n$ and binary labels $b_i = \pm 1$.

Seek to learn from them a weight vector $z \in \mathbb{R}^m$ such that the following functions give the odds of a new feature vector x belonging to class $+1$ and -1, resp.:

$$p_+(x; z) = \frac{1}{1 + e^{z^T x}}, \qquad p_-(x; z) = \frac{1}{1 + e^{-z^T x}}.$$

Denote $L_+ := \{i \mid b_i = +1\}$, $L_- := \{i \mid b_i = -1\}$.

- For $x_i \in L_+$, want $z^T x_i \ll 0$, so that $p_+(x_i; z) \approx 1$.
- For $x_i \in L_-$, want $z^T x_i \gg 0$, so that $p_-(x_i; z) \approx 1$.

ℓ_1 regularization

Negative, scaled a posteriori log likelihood function is

$$\mathcal{L}(z) = -\frac{1}{n}\left[\sum_{i\in L_-} \log p_-(x_i; z) + \sum_{i\in L_+} \log p_+(x_i; z)\right]$$
$$= -\frac{1}{n}\left[\sum_{i\in L_-} z^T x_i - \sum_{i=1}^{n} \log(1 + e^{z^T x_i})\right].$$

We seek a solution z with few nonzeros, so add a regularization term $\lambda\|z\|_1$:

$$\min_z \; T_\lambda(z) := \mathcal{L}(z) + \lambda\|z\|_1.$$

Smaller $\lambda \Rightarrow$ more nonzeros in solution z.

Functions, Derivatives and What They Cost

Denote by X the $n \times m$ matrix $[x_i^T]_{i=1}^n$. Main cost in evaluating function $\mathcal{L}$ is Xz. This can be cheap if z is sparse.

Gradient is

$$\nabla \mathcal{L}(z) = \frac{1}{n} X^T y, \quad \text{where} \quad y_i = \begin{cases} -(1 + e^{z^T x_i})^{-1}, & i \in L_-, \\ (1 + e^{-z^T x_i})^{-1}, & i \in L_+. \end{cases}$$

In block coordinate descent or similar schemes, may need only a subset $\mathcal{G} \subset \{1, 2, \ldots, m\}$ of components of this vector.

Cost: Assuming that Xz is already known from the $\mathcal{L}$ evaluation, need $O(n)$ (to calculate y) plus the cost of a matrix-vector product involving column submatrix $X_{\cdot \mathcal{G}}$.

This is about a fraction $|\mathcal{G}|/n$ of the cost of a full gradient.

Hessian and Sampling

$$\nabla^2 \mathcal{L}(z) = \frac{1}{n} X^T \text{diag}(f) X, \quad \text{where} \quad f_i = \frac{e^{z^T x_i}}{(1 + e^{z^T x_i})^2}.$$

Costs: Assuming Xz known, main cost is forming (weighted) product of $X_{\cdot\mathcal{C}}$ and its transpose, where $\mathcal{C} \subset \{1, 2, \ldots, m\}$ is the subset of variables for which we want to evaluate the reduced Hessian $\nabla^2 \mathcal{L}_{\mathcal{C}\mathcal{C}}$.

Can use *sampling* (Nocedal et al., 2010) to approximate the projected Hessian: take a subset $\mathcal{S} \subset \{1, 2, \ldots, n\}$ and use $X_{\mathcal{S}\mathcal{C}}$ in place of $X_{\cdot\mathcal{C}}$. Reduces evaluation cost by a factor $|\mathcal{S}|/n$.

Strategy at Step k

- Choose a subset $\mathcal{G}_k \in \{1, 2, \ldots, n\}$ by taking the current nonzeros and a random subset of the rest.
- Evaluate $\nabla \mathcal{L}_{\mathcal{G}_k}$ and solve (in closed form):

$$\min_d \nabla \mathcal{L}(z^k)^T d + \frac{\alpha_k}{2} d^T d + \lambda \|z^k + d\|_1, \quad \text{s.t.} \quad d_i = 0 \ \text{ for } i \notin \mathcal{G}_k.$$

- Define $\mathcal{C}_k \subset \mathcal{G}_k$ by $\mathcal{C}_k := \{i \,|\, (z^k + d)_i \neq 0\}$ and calculate a reduced Newton-like step on this subspace.
- Replace $\mathcal{C}_k$ components of d by reduced Newton step (giving a two-metric direction) and do a cursory line search if necessary.
- If two-metric step fails, try first-order step.
- Increase α_k as needed to satisfy sufficient decrease condition: require improvement of at least $c_1(\alpha_k/2)\|d\|^2$ for some $c_1 \in (0, 1)$.

Continuation, Other Enhancements

Problems with small λ are often much harder to optimize. Here use a continuation strategy of solving for a decreasing sequence $\lambda_0 > \lambda_1 > \lambda_2 > \cdots > \lambda_T$, where λ_T is the target value, and using the solution for λ_{t-1} as a starting point for the problem with λ_t.

Effective in practice; still working on the theory.

Various other enhancements in progress, including extension to group-separable regularizers $P(z) = \sum_{g=1}^{G} \lambda_g \|z_{[g]}\|_2$, where the $z_{[g]}$ are disjoint subvectors of z.

Application: Eye Study

- W. Shi, G. Wahba, S. J. Wright, K. Lee, R. Klein, and B. Klein, "LASSO-Patternsearch algorithm with application to opthalmology data," *Statistics and its Interface* 1 (2008), pp. 137-153. Code: `http://pages.cs.wisc.edu/ swright/LPS/`

Beaver Dam Eye Study. Examined 876 subjects for myopia.

- 7 risk factors identified: gender, income, juvenile myopia, cataract, smoking, aspiring, vitamin supplements.
- Bernoulli model: Chose a cutpoint for each factor, assign 1 for above cutpoint and 0 for below.
- Examine all $2^7 = 128$ interacting factors.

The four most significant factors are:

- cataracts (2.42)
- smoker, don't take vitamins (1.11)
- male, low income, juvenile myopia, no aspirin (1.98)
- male, low income, cataracts, no aspirin (1.15)

plus an intercept of -2.84.

A much larger application about genetic risk factors for rheumatoid arthritis also studied ($> 400,000$ variables).

Application: Predicting Splice Sites

Problem from

- V. Roth and B. Fischer, "The Group-Lasso for generalized linear models: Uniqueness of solutions and efficient algorithms," Proceedings of the 25th ICML, 2008.

Splice: region between coding and noncoding region in DNA segments (introns and extrons)

The idea is to look at a sequence of nine base pairs (e.g. CAGGTAAGT) and decide whether it has the "signature" of a splice site.

Represent each location by four binary variables e.g. $A = 1000$, $T = 0100$, $C = 0010$, $G = 0001$. Also consider posisble interactions between base paris at different locations — all possible pairs, triples, quads, quints.

Can leave out locations 3 and 4 which are alwats G and T. Need 33,068 binary variables to capture the remaining possible effects.

Training and Validation

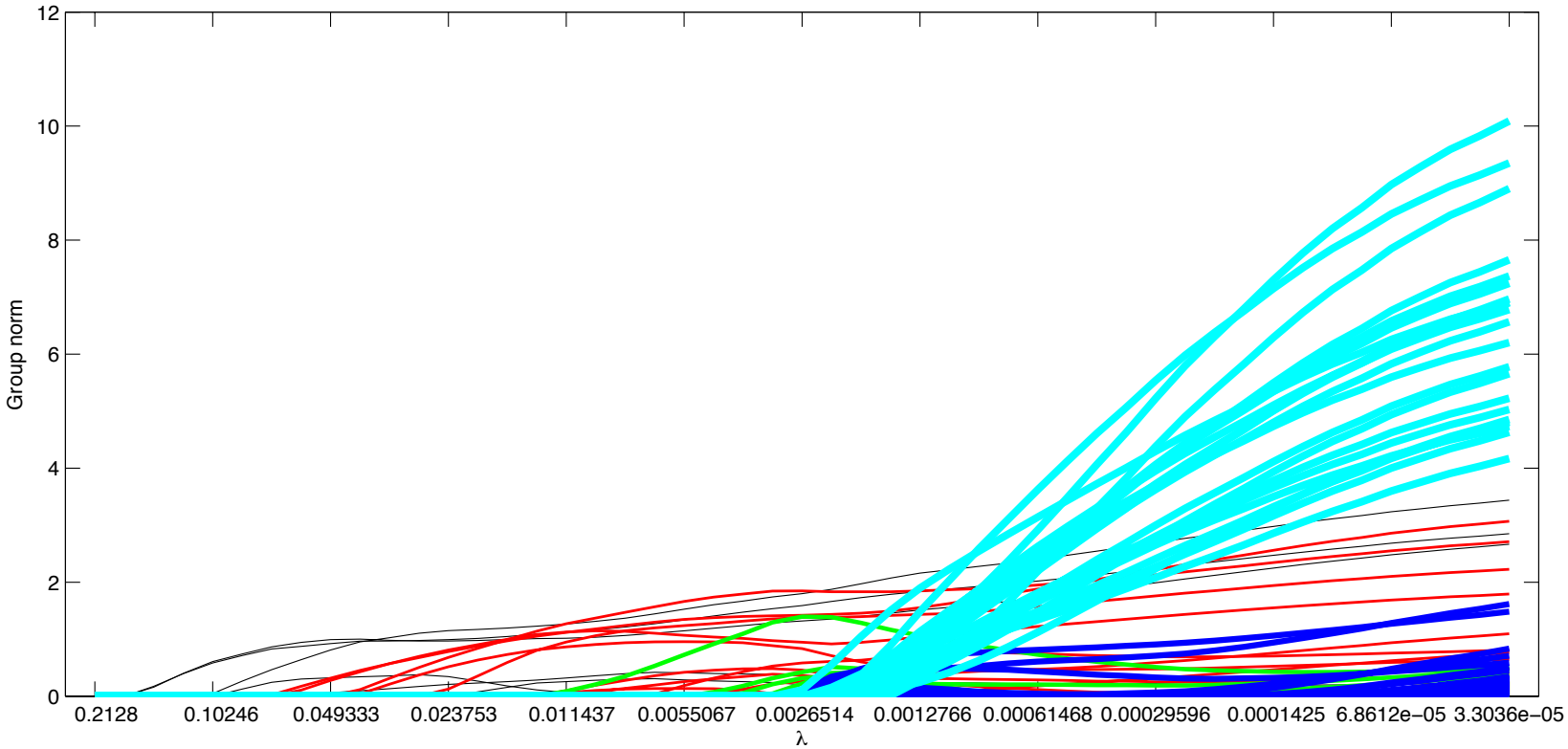
Have a data set of 8415 positive and 179458 negative examples. Select from these an equal number of each for training.

Solve a *group-regularized* logistic regression problem, with groups corresponding to the main effects and the various combinations.

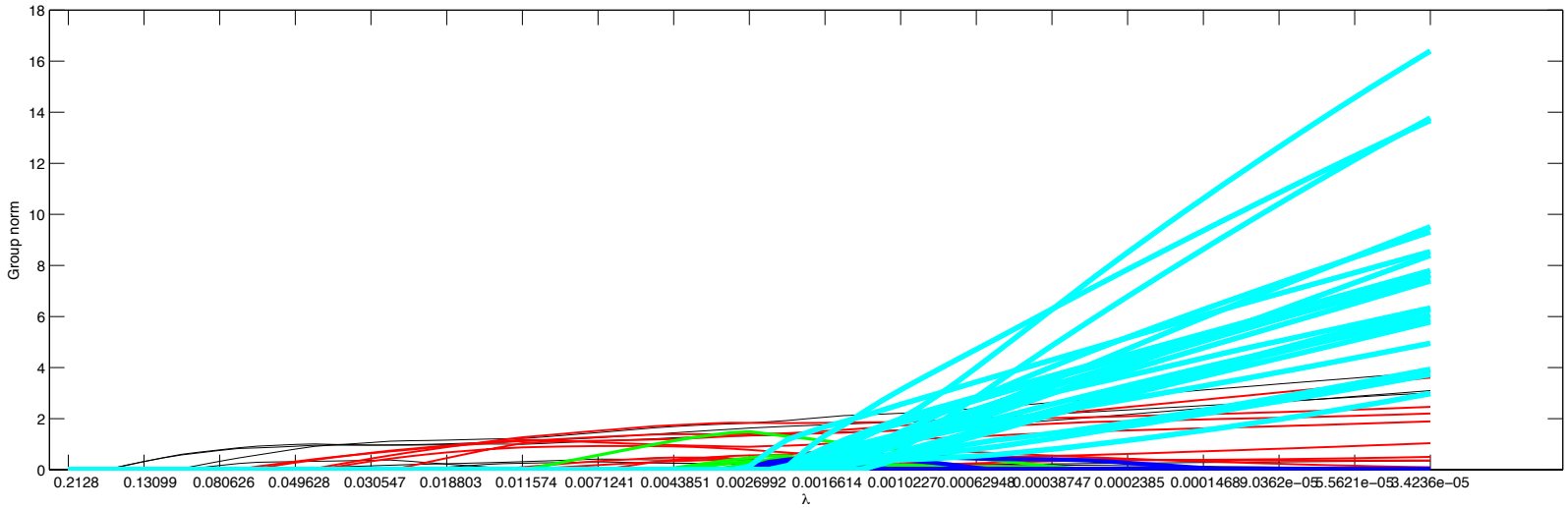
Solve for range of values of λ. Use a validation set to choose the most appropriate value.

Termination criterion is critical to the actual solution but may not make much difference to predictive power. Preliminary plots follow:

Convergence Tolerance 10^{-4}



Convergence Tolerance 10^{-6}



3. Tensor Decompositions

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

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

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

- Smallest F may be different over $\mathbb{R}$ and $\mathbb{C}$.
- Finding smallest F is NP-hard.
- *Maximum* and *typical* ranks of random tensors may be different.
- Minimum-rank decompositions are nonunique for matrices, but often unique for tensors.
- Can have a sequence of rank-F tensors approaching a rank-$(F+1)$ tensor.

There is interest in solving "tensor completion" problems where we find a rank-F tensor that closely approximates the observations in a given tensor.

Tensors in Practice

Tensor problems arise in chemometrics (fluroescence excitation-emission), processing auditory signals, psychometrics, sensor array processing, neuroscience, EEG, functional MRI, image compression, data mining.

A low-rank approximate factorization allows the principal effects to be identified — allows interpretation — and condenses the data without significant loss of information.

Like PCA for matrix analysis, but better suited to situations in which the data is "naturally" multidimensional.

Tensor: Fluorescense Example

27 sample solutions containing 4 known fluorophores. Excite each sample with each of 24 wavelengths (from 250-315 nm) and measure emissions at each of 121 wavelengths (from 241-481 nm).

Measurements are assembled in a $27 \times 121 \times 24$ array.

From a physical law the measurement should be given by a physical law involving 4 terms (for the 4 fluorophores):

$$X_{ijk} = \sum_{f=1}^{4} \zeta_{kf} \varepsilon_{if} \eta_{jf}$$

- ζ_{kf} is concentration of element f in solution k,
- ε_{if} is excitation factor for element f at excitation wavelength i,
- η_{jf} is emission factor at emission wavelength j.

These terms can be estimated by finding a rank-4 tensor that best approximates the measured data matrix X.

Can fit X in a least-squares sense. Can still do this if some data is missing!

Algorithms for Tensor Decompositions

Despite the theoretical difficulties, plow ahead! Given 3D tensor X of dimensions $I \times J \times K$, and rank F, seek vectors a_f, b_f, c_f, and scalars λ_f, $f = 1, 2, \ldots, F$, such that

$$X \approx \sum_{f=1}^{F} \lambda_f a_f \circ b_f \circ c_f.$$

Alternating Least Squares is an old technique but one that has not yet been improved on much. At each step, solve three linear least squares problems. In the first of these, hold b_f and c_f constant and solve for λ_f and a_f, $f = 1, 2, \ldots, F$:

$$\min_{\lambda, A} \left\| X - \sum_{f=1}^{F} \lambda_f a_f \circ b_f \circ c_f \right\|_F^2.$$

(Calibrate so that $\|a_f\| = 1$ for all f and $\lambda_f \geq 0$.)

Problems for (λ_f, b_j) and (λ_f, c_f) are similar.

ALS Properties

Least-squares subproblems highly structured: more expensive to form the right-hand side of the normal equations than the coefficient matrix.

ALS sometimes performs well enough in practice. Typically large reductions in early iterations, then very slow.

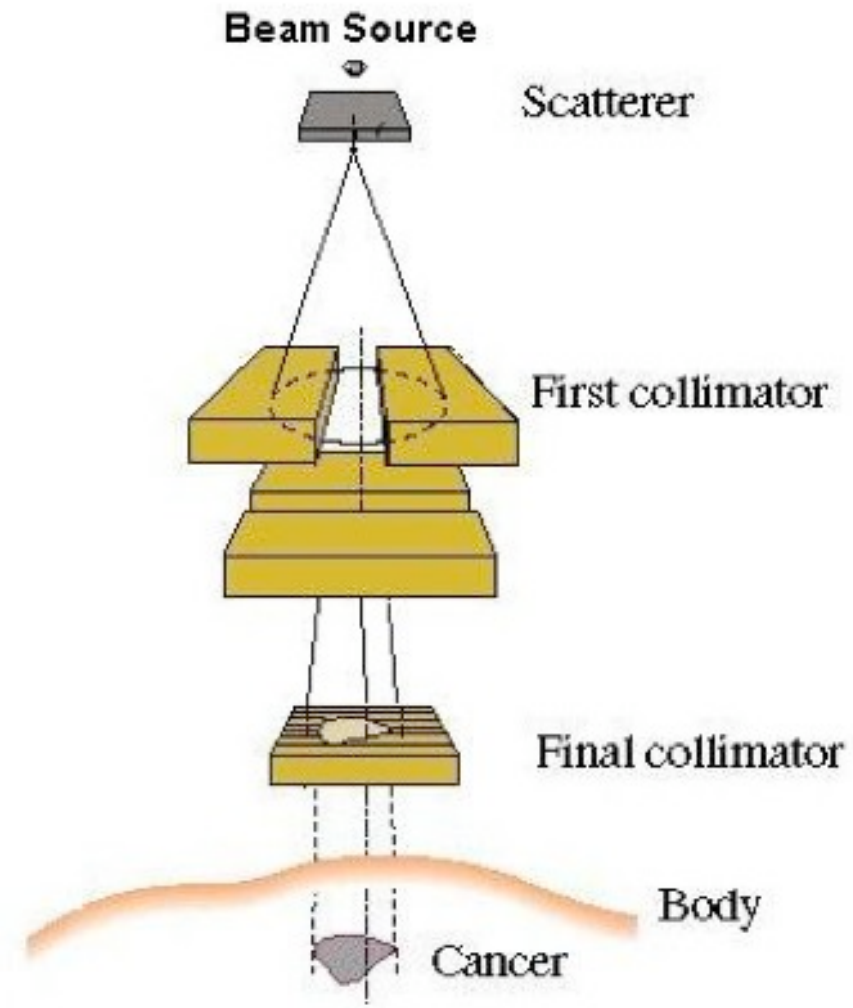
Theoretically, ALS is not well understood. It cycles on some examples, and may approach local minima (which may exist, by nonconvexity).

Many enhancements / alternatives proposed (e.g. in an April 2010 workshop at AIM, Palo Alto) but few yet tried:

- sampling for the right-hand side in ALS;
- alternative steplengths and non-monotone ALS;
- more general "full space" optimization methods;
- alternative loss functions to $\|\cdot\|_F^2$;
- imposing additional structure, e.g. nonnegativity of factors.

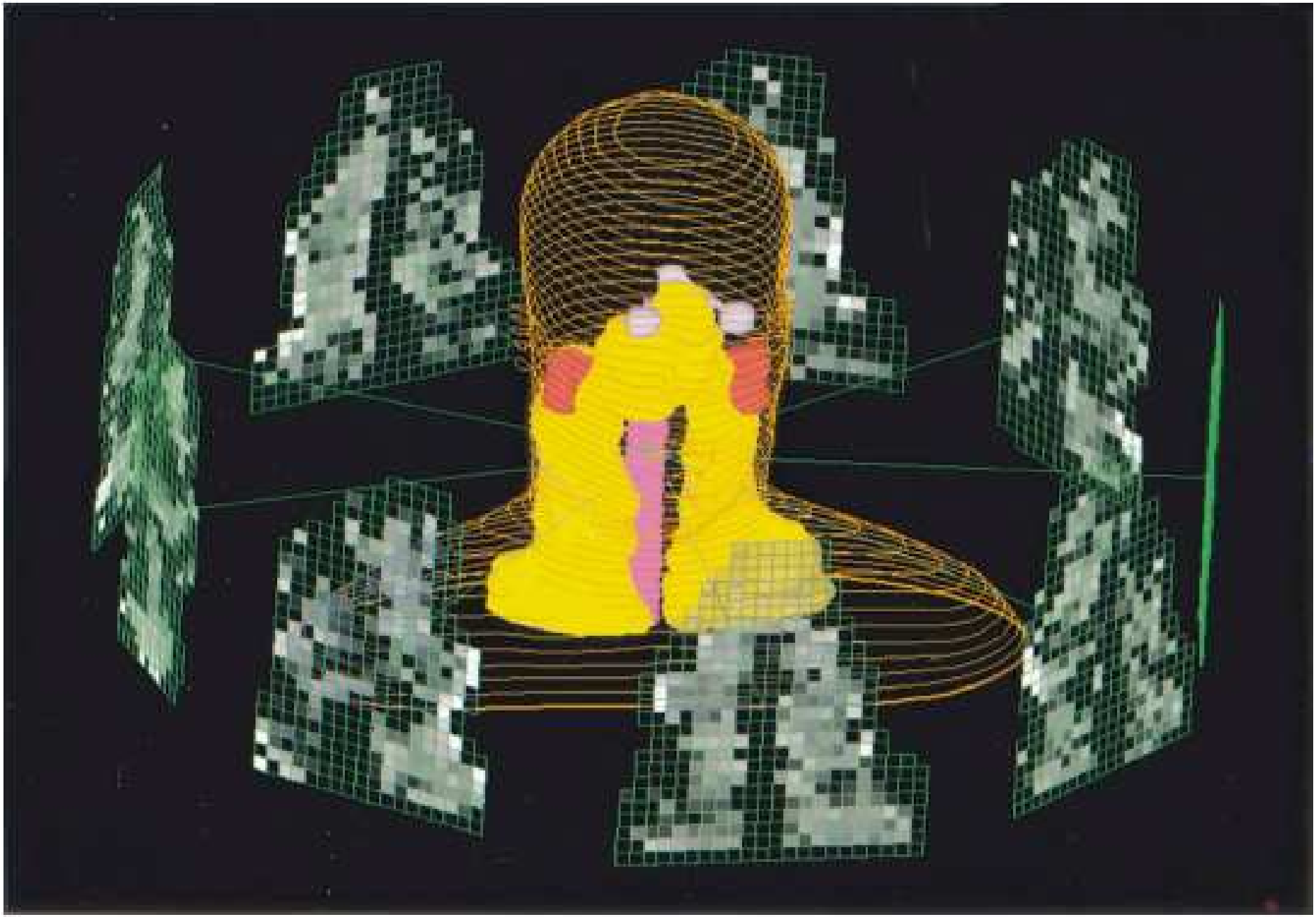
4. Cancer Treatment Planning

- 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.



Linear accelerator, showing cone and collimators

Multileaf collimator. Leaves move up and down to shape the beam.



5. Kernel Estimation and Clustering via SDP

Application: Given a collection of N objects a set of distances d_{ij} between certain pairs of objects (possibly redundant, incomplete, inexact).

Seek to find a Euclidean vector representation $x_i \in \mathbb{R}^N$ for each $i = 1, 2, \ldots, N$, such that $\|x_i - x_j\|_2 \approx d_{ij}$ for the observed distance pairs (i, j).

Could also seek a more compact representation, say $x_i \in \mathbb{R}^3$.

From the set of x_i so defined, can

- perform cluster analysis;
- given a new object and distances to some of the original objects $i = 1, 2, \ldots, N$, can "place" the new object in the same space as the x_i, and possibly assign it to an existing cluster.

Formulation

Get the $x_i \in \mathbb{R}^N$ indirectly, by seeking a "distance matrix" K whose (i,j) element represents $\langle x_i, x_j \rangle$.

Distances induced by K are thus

$$d_{ij}(K) = \|x_i - x_j\|_2^2 = K_{ii} + K_{jj} - 2K_{ij},$$

and K is $N \times N$ positive semidefinite ($K \succeq 0$).

If Ω is the set of observed distance pairs, and the operator $\bullet$ is defined by $Y \bullet Z = \sum_{i,j=1}^{N} Y_{ij} Z_{ij}$, the fitting problem can be written as

$$\min_{K \succeq 0} \sum_{(i,j) \in \Omega} |d_{ij} - B_{ij} \bullet K|,$$

where B_{ij} is the $N \times N$ symmetric matrix with four nonzero elements:

$$B_{ij}(i,j) = B_{ij}(j,i) = -1, \qquad B_{ij}(i,i) = B_{ij}(j,j) = 1.$$

Note that $B_{ij} \bullet K = K_{ii} + K_{jj} - 2K_{ij} = \|x_i - x_j\|_2^2$.

Semidefinite Program

The fitting problem is a semidefinite program. General form of SDP is

$$\min_X \; C \bullet X \;\; \text{subject to} \;\; X \succeq 0, \;\; A_i \bullet X = b_i, \;\; i = 1, 2, \ldots, m,$$

where C and A_i are all symmetric.

Usually solved with interior-point methods. Good codes are available: SeDuMi, SDPT3, others.

Recovering x_i: Having obtained K, perform an eigen-decomposition $K = \Lambda \Gamma \Lambda^T$ (where Λ is orthogonal and Γ is diagonal), and define $X = \Lambda \Gamma^{1/2}$. Take x_i to be the ith row of X.

Regularized Fitting

Can suppress the rank of K (and thus the dimension of each x_i) by adding this regularization term to the objective:

$$\tau \text{trace}(K) = \tau \sum_{i=1}^{N} \lambda_i(K).$$

(Sparse optimization again!)

Larger $\tau \Rightarrow$ forces more of the eigenvalues $\lambda_i(K)$ to zero. Often a "cutoff" is revealed, e.g. the three largest eigenvalues dominate. This would yield $x_i \in \mathbb{R}^3$.

Dimension matters!

This problem is challenging for SDP software because of the many constraint: $|\Omega|$ in total, potentially up to $N^2/2$. (In our data set below, $N \approx 280$ and $|\Omega| \approx 14000$.)

Alternative "incremental" approach: Place a subset $\bar{N}$ of the objects as above, to derive $K \in \mathbb{R}^{\bar{N} \times \bar{N}}$ symmetric and rank r, and thus $x_i \in \mathbb{R}^r$, $i = 1, 2, \ldots, \bar{N}$.

For each remaining object j, place x_j the space $\mathbb{R}^r$ to best fit the distances to the points already placed (without moving those points). Can formulate this problem approximately as a conic problem of smaller dimension.

Get a conic program with a 2×2 SDP variable, a SOC variable of dimension rank(K), and $2|\Psi_j|$ linear terms, where Ψ_j is the number of measured distance pairs involving the new point j.

Application: Protein Clustering

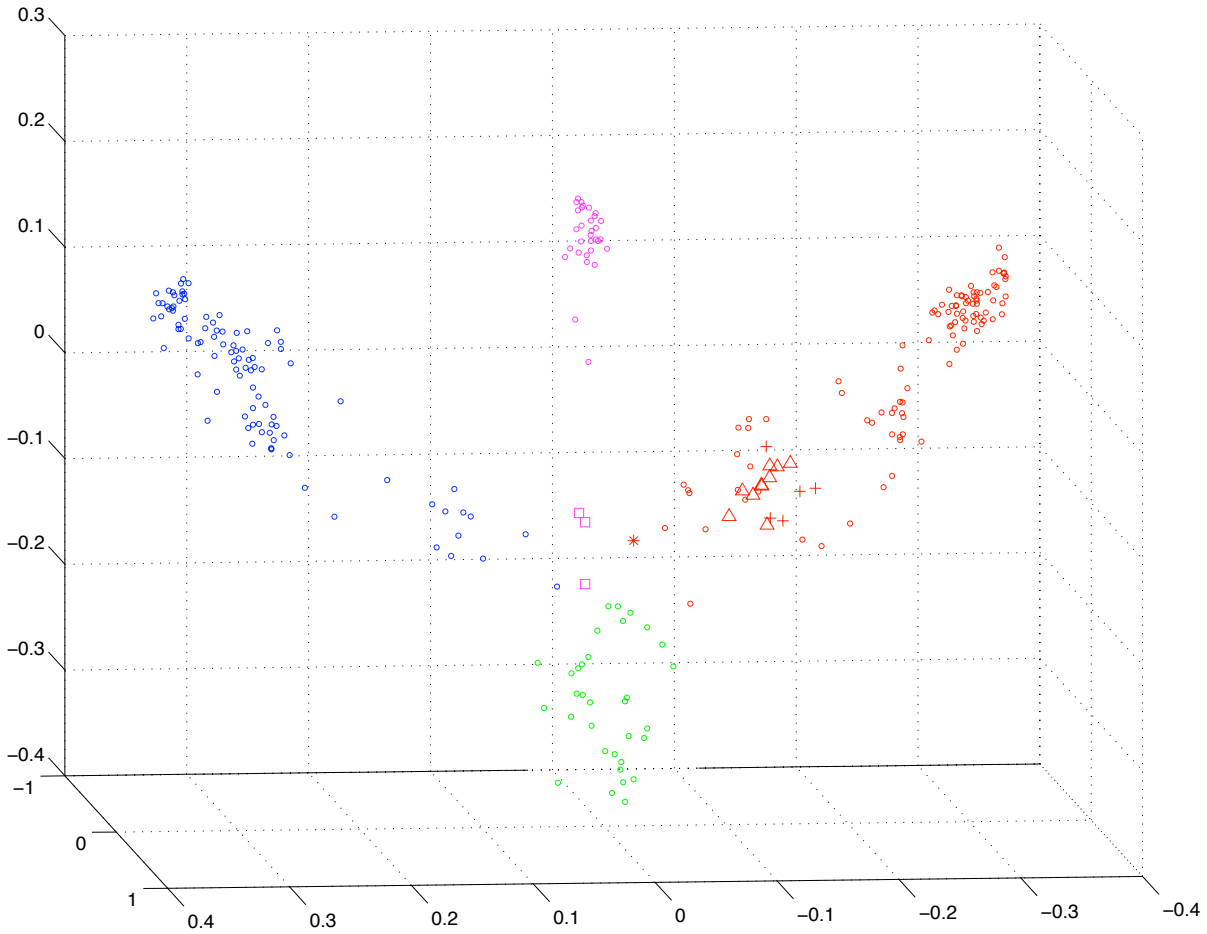
- Lu, F., Keles, S., Wright, S. J., and Wahba, G. "Framework for kernel regularization with application to protein clustering," *Proceedings of the National Academy of Sciences* 102 (2005), pp. 12332–12337.

Infer protein function from sequence similarity.

- Use SDP to represent proteins in low-dimension space, then cluster and classify.
- Assigning new unannotated proteins to the nearest class.

Choose 280 proteins from a databse of 630. Four classes: alpha-globins, beta-globins, myglobins, globins (heterogeneous).

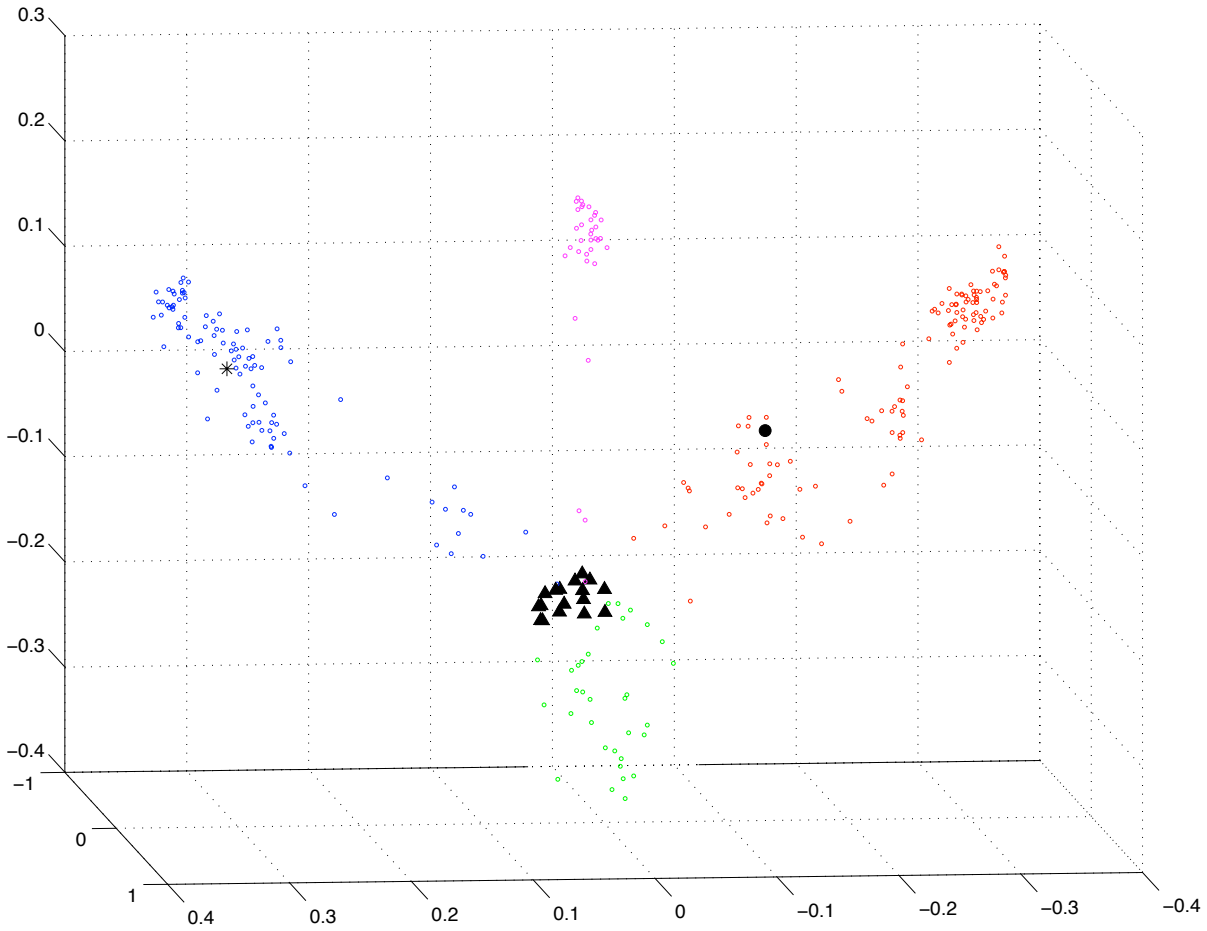
Solve the SDP formulation and reduce to 3 dimensions. The four classes appear as distinct clusters. (The three fish proteins are slightly removed from the other myglobins.)


0.3
0.2
0.1
0
−0.1
−0.2
−0.3
−0.4
−1
0
1
0.4
0.3
0.2
0.1
0
−0.1
−0.2
−0.3
−0.4

Add three sets of test proteins to the previous set, solve the incremental problem for each.

- Hemoglobin zeta chain from a goat
- Hemoglobin theta chain from a pig
- 17 Leghemoglobins.

Each of the 3 classes fits neatly within one of the existing clusters.
Consistent with results of previous studies based on hidden Markov models.


0.3
0.2
0.1
0
−0.1
−0.2
−0.3
−0.4
−1
0
1
0.4
0.3
0.2
0.1
0
−0.1
−0.2
−0.3
−0.4

6. Genetically optimal captive breeding programs.

(with Webb Miller, Penn State)

- A pool of n animals with known genetic information.
- Select k of them to breed in captivity, for later release into the wild.
- Choose the subset to optimize some goal, e.g. achieve a target genetic profile.

$$\min \frac{1}{2}\|Ax - b\|_2^2 \text{ s.t. } Cx = d, \quad x \in \{0,1\}^n.$$

- n is number of animals. x_j indicates whether animal j is selected for breeding or not.
- A_{ij} = number of reference alleles (0,1,2) for animal j at SNP i.
- b_i = target total allele representation at SNP i.
- $Cx = d$ includes a constraint on total number of animals selected, possibly other knapsack constraints based on age and gender.

Solving as an Integer Program

Our data has $n \approx 172$ and includes constraint $\sum_j x_j = 50$. Hard!

We used CPLEX's MIP solvers on various formulations:

- Integer QP (above). After about 2 days of CPU time, finds incumbent with objective 3.8113 and lower bound of 1.8878. Visits 13.5M nodes. Most progress made in the first minutes, but steady improvements throughout.
- Redefined objective as $\|Ax - b\|_1$ and call CPLEX MILP solver with various options. Cuts are important; only 200 nodes examined. Best incumbent was 2.2150 with lower bound .4717. Most progress occurs in first seconds.

These problems have notoriously weak QP relaxations (Bienstock, 2008). Special techniques can be used to find lower bounds at each node:

- Compute distance from relaxed QP solution to nearest feasible point;
- Use curvature of Hessian to raise the lower bound.

Customized lower bounding is hard to implement in CPLEX, however.

QCQP and SDP Relaxation

Can formulate as a quadratically constrained quadratic program (QCQP) by rewriting $x_j \in \{0, 1\}$ as linear and quadratic constraints:

$$\min \frac{1}{2}\|Ax - b\|_2^2 \text{ s.t. } C_i.x = d_i, \quad 0 \leq x_j \leq 1, \quad x_j(1 - x_j) = 0.$$

SDP can be used to solve relaxations of QCQP. General form of QCQP is

$$\min x^T A_0 x + b_0^T x \text{ s.t. } x^T A_k x + b_k^T x + c_k \leq 0, \ k = 1, 2, \ldots, m.$$

where A_k are symmetric $n \times n$ matrices, possibly indefinite, for $k = 0, 1, \ldots, m$.

Reformulation and Relaxation

Define

$$B_k := \begin{bmatrix} A_k & b_k/2 \\ b_k^T/2 & c_k \end{bmatrix}$$

and redefine $x := (x; x_{n+1})$ (add a single component). Then rewrite QP as

$$\min x^T B_0 x \;\text{ s.t. } x_{n+1} = 1, \;\; x^T B_k x \le 0, \; k = 1, 2, \ldots, m.$$

Defining $X = xx^T$, and inner product $Y \bullet Z := \sum_{i,j} Y_{ij} Z_{ij}$, can rewrite as

$$\min B_0 \bullet X \;\text{ s.t. } X_{n+1,n+1} = 1, \;\; B_k \bullet X \le 0, \; k = 1, 2, \ldots, m, \;\; \text{rank}(X) = 1.$$

Get SDP relaxation by dropping the rank constraint.

Obviously $V_{\text{SDP}} \le V_{\text{QCQP}}$, where V are the respective value functions.
Other issues include:

- recovering a feasible solution for QCQP from the SDP solution (possibly randomly) with provably good (expected) quality.
- finding other bounds e.g. $V_{\text{QCQP}} \le \alpha V_{\text{SDP}}$ for some $\alpha \in (0, 1)$.

SDP Relaxation Results

In the relaxation, the constraint $x_j = x_j^2$ (which holds if and only if x_j is either zero or one) can be expressed as

$$(X_{j,n+1} + X_{n+1,j})/2 - X_{jj} = 0.$$

Various "tricks" can be applied to strengthen the relaxation.

- add m constraints $(C_i.x)^2 = d_i^2$.
- add mn constraints $x_j(C_i.x - d_i) = 0$.

Results: SeDuMi produces lower bounds of 3.367 to 3.409 depending on which constraints are enforced, what scalings are used. Run times: 10 seconds to a few minutes.

Element-wise nonnegativities in X

Too expensive to enforce $X_{jl} \geq 0$ for all (j, l) are there are $n^2/2$ of these.

Can add these in "constraint generation" fashion, solving multiple SDPs in which violations of these bounds are successively added to the formulation. Slow, yields gradual increase in objective.

Better: Use Burer's code DNN for doubly nonnegative matrices (2009).

- Represents X by two different variables $\tilde{X}$ and $\hat{X}$, and enforces $\tilde{X} \succeq 0$ and $\hat{X} \geq 0$.
- Uses augmented Lagrangian to enforce $\hat{X} = \tilde{X}$.
- Alternates between gradient projection steps in $\hat{X}$ and $\tilde{X}$. (Easy to project onto the feasible set for these two variables separately.)

Results: Lower bound of 3.6111 (6% optimality gap).

Conclusions

- Optimization is relevant to any areas of bioinformatics, biology, medicine.
- Applications in these and other related areas are driving developments in optimization algorithms and motivating new lines of work.
- The possibilities for further interactions seem endless!