Disciplined Convex Programming

Stephen Boyd Michael Grant

Electrical Engineering Department, Stanford University

University of Pennsylvania, 3/30/07

Outline

- convex optimization
- checking convexity via convex calculus
- convex optimization solvers
- efficient solution via problem transformations
- disciplined convex programming
- examples
  - bounding portfolio risk
  - computing probability bounds
  - antenna array beamforming
  - ℓ_1-regularized logistic regression

Optimization

opitmization problem with variable $x \in \mathbf{R}^n$:

$$\begin{array}{ll} \text{minimize} & f_0(x) \\ \text{subject to} & f_i(x) \leq 0, \quad i = 1, \ldots, m \\ & h_i(x) = 0, \quad i = 1, \ldots, p \end{array}$$

- our ability to solve varies widely; depends on properties of f_i, h_i
- for f_i, h_i affine (linear plus constant) get linear program (LP); can solve very efficiently
- even simple looking, relatively small problems with nonlinear f_i, h_i can be intractable

Convex optimization

convex optimization problem:

$$\begin{array}{ll} \text{minimize} & f_0(x) \\ \text{subject to} & f_i(x) \leq 0, \quad i = 1, \ldots, m \\ & Ax = b \end{array}$$

- objective and inequality constraint functions f_i are convex:
  for all x, y, $\theta \in [0, 1]$,

  $$f_i(\theta x + (1-\theta)y) \leq \theta f_i(x) + (1-\theta) f_i(y)$$

  roughly speaking, graphs of f_i curve upward

- equality constraint functions are affine, so can be written as $Ax = b$

Convex optimization

- a subclass of optimization problems that includes LP as special case
- convex problems can look very difficult (nonlinear, even nondifferentiable), but like LP can be solved very efficiently
- convex problems come up more often than was once thought
- many applications recently discovered in control, combinatorial optimization, signal processing, communications, circuit design, machine learning, statistics, finance, . . .

General approaches to using convex optimization

- pretend/assume/hope f_i are convex and proceed
  - easy on user (problem specifier)
  - but lose many benefits of convex optimization
- verify problem is convex before attempting solution
  - but verification for general problem description is hard, often fails
- construct problem as convex from the outset
  - user needs to follow a restricted set of rules and methods
  - convexity verification is automatic

each has its advantages, but we focus on 3rd approach

How can you tell if a problem is convex?

need to check convexity of a function

approaches:

- use basic definition, first or second order conditions, *e.g.*, $\nabla^2 f(x) \succeq 0$
- via convex calculus: construct f using
  - library of basic examples or atoms that are convex
  - calculus rules or transformations that preserve convexity

Convex functions: Basic examples

- x^p for $p \geq 1$ or $p \leq 0$; $-x^p$ for $0 \leq p \leq 1$
- e^x, $-\log x$, $x \log x$
- $a^T x + b$
- $x^T x$; $x^T x/y$ (for $y > 0$); $(x^T x)^{1/2}$
- $\|x\|$ (any norm)
- $\max(x_1, \ldots, x_n)$, $\log(e^{x_1} + \cdots + e^{x_n})$
- $\log \Phi(x)$ (Φ is Gaussian CDF)
- $\log \det X^{-1}$ (for $X \succ 0$)

Calculus rules

- *nonnegative scaling*: if f is convex, $\alpha \geq 0$, then αf is convex
- *sum*: if f and g are convex, so is $f + g$
- *affine composition*: if f is convex, so is $f(Ax + b)$
- *pointwise maximum*: if $f_1, \ldots, f_m$ are convex, so is $f(x) = \max_i f_i(x)$
- *partial minimization*: if $f(x, y)$ is convex, and C is convex, then $g(x) = \inf_{y \in C} f(x, y)$ is convex
- *composition*: if h is convex and increasing, and f is convex, then $g(x) = h(f(x))$ is convex (there are several other composition rules)

. . . and many others (but rules above will get you quite far)

Examples

- piecewise-linear function: $f(x) = \max_{i=1,\ldots,k}(a_i^T x + b_i)$
- ℓ_1-regularized least-squares cost: $\|Ax - b\|_2^2 + \lambda\|x\|_1$, with $\lambda \geq 0$
- sum of largest k elements of x: $f(x) = x_{[1]} + \cdots + x_{[k]}$
- log-barrier: $-\sum_{i=1}^m \log(-f_i(x))$ (on $\{x \mid f_i(x) < 0\}$, f_i convex)
- distance to convex set C: $f(x) = \mathbf{dist}(x, C) = \inf_{y \in C} \|x - y\|_2$

note: except for log-barrier, these functions are nondifferentiable . . .

How do you solve a convex problem?

- use someone else's ('standard') solver (LP, QP, SDP, . . .)
  - easy, but your problem *must* be in a standard form
  - cost of solver development amortized across many users
- write your own (custom) solver
  - lots of work, but can take advantage of special structure
- transform your problem into a standard form, and use a standard solver
  - extends reach of problems that can be solved using standard solvers
  - transformation can be hard to find, cumbersome to carry out

this talk: methods to formalize and automate the last approach

General convex optimization solvers

subgradient, bundle, proximal, ellipsoid methods

- mostly developed in Soviet Union, 1960s–1970s
- are 'universal' convex optimization solvers, that work even for nondifferentiable f_i
- ellipsoid method is 'efficient' in theory ($i.e.$, polynomial time)
- all can be slow in practice

Interior-point convex optimization solvers

- rapid development since 1990s, but some ideas can be traced to 1960s
- can handle smooth f_i ($e.g.$, LP, QP, GP), and problems in conic form (SOCP, SDP)
- are extremely efficient, typically requiring a few tens of iterations, almost independent of problem type and size
- each iteration involves solving a set of linear equations (least-squares problem) with same size and structure as problem
- *method of choice* when applicable

What if interior-point methods can't handle my problem?

- example: ℓ_1-regularized least-squares (used in machine learning):

$$\text{minimize} \quad \|Ax - b\|_2^2 + \lambda \|x\|_1$$

- a convex problem, but objective is nondifferentiable, so cannot directly use interior-point method (IPM)

- **basic idea**: transform problem, possibly adding new variables and constraints, so that IPM can be used

- even though transformed problem has more variables and constraints, we can solve it very efficiently via IPM

Example: ℓ_1-regularized least-squares

- original problem, with n variables, no constraints:

$$\text{minimize} \quad \|Ax - b\|_2^2 + \lambda \|x\|_1$$

- introduce new variable $t \in \mathbf{R}^n$, and new constraints $|x_i| \leq t_i$:

$$\begin{array}{ll} \text{minimize} & x^T(A^TA)x - (A^Tb)^Tx + \lambda \mathbf{1}^Tt \\ \text{subject to} & x \leq t, \quad -t \leq x \end{array}$$

- a problem with $2n$ variables, $2n$ constraints, but objective and constraint functions are *smooth* so IPM can be used

- key point: problems are *equivalent* (if we solve one, we can easily get solution of other)

Efficient solution via problem transformations

- start with convex optimization problem $\mathcal{P}_0$, possibly with nondifferentiable objective or constraint functions
- carry out a sequence of equivalence transformations to yield a problem $\mathcal{P}_K$ that can be handled by an IP solver

$$\mathcal{P}_0 \rightarrow \mathcal{P}_1 \rightarrow \cdots \rightarrow \mathcal{P}_K$$

- solve $\mathcal{P}_K$ efficiently
- transform solution of $\mathcal{P}_K$ back to solution of original problem $\mathcal{P}_0$
- $\mathcal{P}_K$ often has more variables and constraints than $\mathcal{P}_0$, but its special structure, and efficiency of IPMs, more than compensates

Convex calculus rules and problem transformations

- for most of the convex calculus rules, there is an associated problem transformation that 'undoes' the rule

- example: when we encounter $\max\{f_1(x), f_2(x)\}$ we
  - replace it with a new variable t
  - add new (convex) constraints $f_1(x) \leq t$, $f_2(x) \leq t$

- example: when we encounter $h(f(x))$ we
  - replace it with $h(t)$
  - add new (convex) constraint $f(x) \leq t$

- these transformations look trivial, but are not

From proof of convexity to IPM-compatible problem

$$\begin{array}{ll} \text{minimize} & f_0(x) \\ \text{subject to} & f_i(x) \leq 0, \quad i = 1, \ldots, m \\ & Ax = b \end{array}$$

- when you construct f_i from atoms and convex calculus rules, you have a *mathematical proof* that the problem is convex
- the same construction gives a sequence of problem transformations that yields a problem containing only atoms and equality constraints
- if the atoms are IPM-compatible, our constructive proof automatically gives us an equivalent problem that is IPM-compatible

Disciplined convex programming

- specify convex problem in natural form
  - declare optimization variables
  - form convex objective and constraints *using a specific set of atoms and calculus rules*
- problem is convex-by-construction
- easy to parse, automatically transform to IPM-compatible form, solve, and transform back
- implemented using object-oriented methods and/or compiler-compilers

Example (cvx)

convex problem, with variable $x \in \mathbf{R}^n$:

$$\begin{array}{ll} \text{minimize} & \|Ax - b\|_2 + \lambda \|x\|_1 \\ \text{subject to} & Fx \leq g \end{array}$$

cvx specification:

```
cvx_begin
    variable x(n)      % declare vector variable
    minimize ( norm(A*x-b,2) + lambda*norm(x,1) )
    subject to F*x <= g
cvx_end
```

when `cvx` processes this specification, it

- verifies convexity of problem
- generates equivalent IPM-compatible problem
- solves it using `SDPT3` or `SeDuMi`
- transforms solution back to original problem

the `cvx` code is easy to read, understand, modify

The same example, transformed by 'hand'

transform problem to SOCP, call `SeDuMi`, reconstruct solution:

```
% set up big matrices
[m,n] = size(A); [p,n] = size(F);
AA = [ speye(n), - speye(n), speye(n), sparse(n,p+m+1) ; ...
         F, sparse(p,2*n), speye(p), sparse(p,m+1) ; ...
         A, sparse(m,2*n+p), speye(m), sparse(m,1) ];
bb = [ zeros(n,1) ; g ; b ];
cc = [ zeros(n,1) ; gamma * ones(2*n,1) ; zeros(m+p,1) ; 1 ];
K.f = m; K.l = 2*n+p; K.q = m + 1;        % specify cone
xx = sedumi( AA, bb, cc, K );              % solve SOCP
x = x(1:n);                                % extract solution
```

History

- general purpose optimization modeling systems AMPL, GAMS (1970s)
- systems for SDPs/LMIs (1990s): `sdpsol` (Wu, Boyd), `lmilab` (Gahinet, Nemirovsky), `lmitool` (El Ghaoui)
- `yalmip` (Löfberg 2000–)
- automated convexity checking (Crusius PhD thesis 2002)
- disciplined convex programming (DCP) (Grant, Boyd, Ye 2004)
- `cvx` (Grant, Boyd, Ye 2005)
- `cvxopt` (Dahl, Vandenberghe 2005)
- `ggplab` (Mutapcic, Koh, et al 2006)

Summary

the bad news:

- you can't just call a convex optimization solver, hoping for the best; convex optimization is not a 'plug & play' or 'try my code' method
- you can't just type in a problem description, hoping it's convex (and that a sophisticated analysis tool will recognize it)

the good news:

- by learning and following a modest set of atoms and rules, you can specify a problem in a form very close to its natural mathematical form
- you can simultaneously verify convexity of problem, automatically generate IPM-compatible equivalent problem

Examples

- bounding portfolio risk
- computing probability bounds
- antenna array beamforming
- ℓ_1-regularized logistic regression

Portfolio risk bounding

- portfolio of n assets invested for single period
- w_i is amount of investment in asset i
- returns of assets is random vector r with mean $\overline{r}$, covariance Σ
- portfolio return is random variable $r^T w$
- mean portfolio return is $\overline{r}^T w$; variance is $V = w^T \Sigma w$

value at risk & probability of loss are related to portfolio variance V

Risk bound with uncertain covariance

now suppose:

- w is known (and fixed)
- have only partial information about Σ, *i.e.*,

$$L_{ij} \le \Sigma_{ij} \le U_{ij}, \quad i,j = 1,\ldots,n$$

problem: how large can portfolio variance $V = w^T \Sigma w$ be?

Risk bound via semidefinite programming

can get (tight) bound on V via semidefinite programming (SDP):

$$\begin{array}{ll} \text{maximize} & w^T \Sigma w \\ \text{subject to} & \Sigma \succeq 0 \\ & L_{ij} \leq \Sigma_{ij} \leq U_{ij} \end{array}$$

variable is matrix $\Sigma = \Sigma^T$; $\Sigma \succeq 0$ means Σ is positive semidefinite

many extensions possible, $e.g.$, optimize portfolio w with worst-case variance limit

cvx specification

```
cvx_begin
    variable Sigma(n,n) symmetric
    maximize ( w'*Sigma*w )
    subject to
    Sigma == semidefinite(n);    % Sigma is positive semidefinite
    Sigma >= L;
    Sigma <= U;
cvx_end
```

Example

portfolio with $n = 4$ assets

variance bounding with sign constraints on Σ:

$$w = \begin{bmatrix} 1 \\ 2 \\ -.5 \\ .5 \end{bmatrix}, \qquad \Sigma = \begin{bmatrix} 1 & + & + & ? \\ + & 1 & - & - \\ + & - & 1 & + \\ ? & - & + & 1 \end{bmatrix}$$

($i.e.$, $\Sigma_{12} \geq 0$, $\Sigma_{23} \leq 0$, $\ldots$)

Result

(global) maximum value of V is 10.1, with

$$\Sigma = \begin{bmatrix} 1.00 & 0.79 & 0.00 & 0.53 \\ 0.79 & 1.00 & -.59 & 0.00 \\ 0.00 & -.59 & 1.00 & 0.51 \\ 0.53 & 0.00 & 0.51 & 1.00 \end{bmatrix}$$

(which has rank 3, so constraint $\Sigma \succeq 0$ is active)

- $\Sigma = I$ yields $V = 5.5$
- $\Sigma = [(L+U)/2]_+$ yields $V = 6.75$ ($[\cdot]_+$ is positive semidefinite part)

Computing probability bounds

random variable $(X, Y) \in \mathbf{R}^2$ with

- $\mathcal{N}(0,1)$ marginal distributions
- X, Y uncorrelated

question: how large (small) can $\mathbf{Prob}(X \leq 0, Y \leq 0)$ be?

if $(X, Y) \sim \mathcal{N}(0, I)$, $\mathbf{Prob}(X \leq 0, Y \leq 0) = 0.25$

Probability bounds via LP

- discretize distribution as p_{ij}, $i,j=1,\ldots,n$, over region $[-3,3]^2$
- $x_i = y_i = 6(i-1)/(n-1) - 3$, $i = 1,\ldots,n$

$$
\begin{array}{ll}
\text{maximize (minimize)} & \sum_{i,j=1}^{n/2} p_{ij} \\
\text{subject to} & p_{ij} \geq 0, \quad i,j=1,\ldots,n \\
& \sum_{i=1}^{n} p_{ij} = a e^{-y_i^2/2}, \quad j=1,\ldots,n \\
& \sum_{j=1}^{n} p_{ij} = a e^{-x_i^2/2}, \quad i=1,\ldots,n \\
& \sum_{i,j=1}^{n} p_{ij} x_i y_j = 0
\end{array}
$$

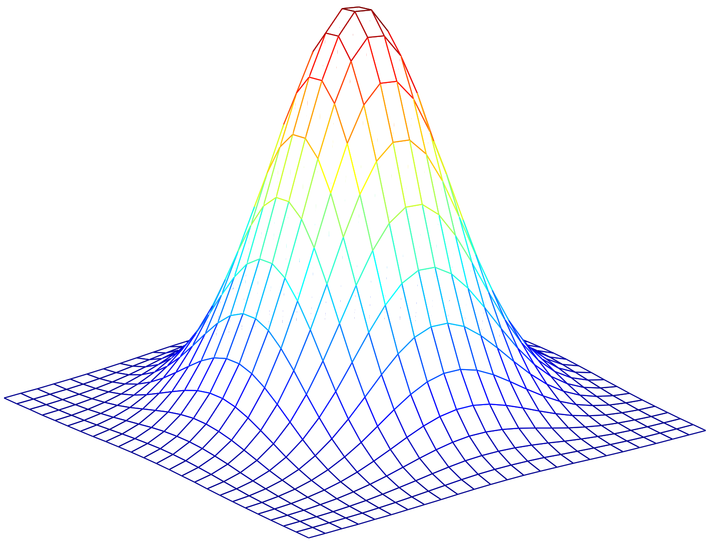
with variable $p \in \mathbf{R}^{n\times n}$, $a = 2.39/(n-1)$

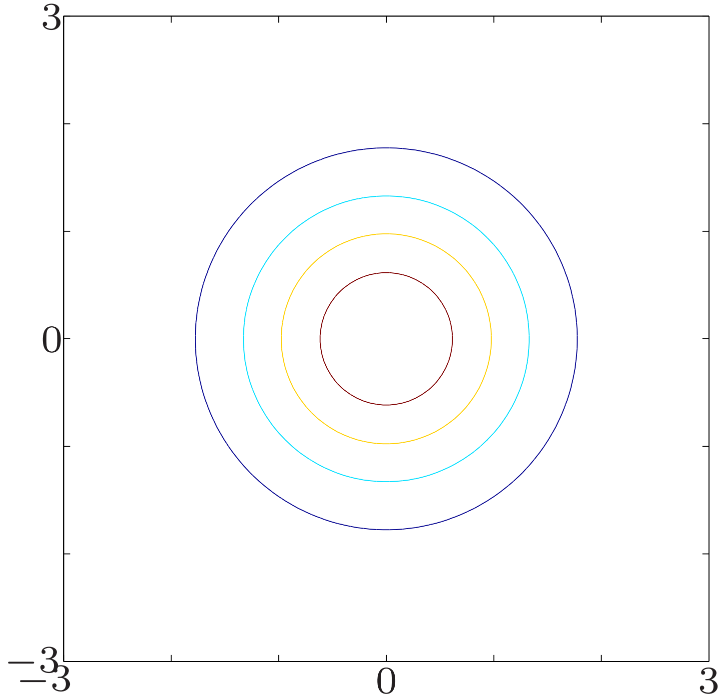
cvx specification

```
cvx_begin
    variable p(n,n)
    maximize ( sum(sum(p(1:n/2,1:n/2))) )
    subject to
        p >= 0;
        sum( p,1 ) == a*exp(-y.^2/2)';
        sum( p,2 ) == a*exp(-x.^2/2)';
        sum( sum( p.*(x*y') ) ) == 0;
cvx_end
```

Gaussian

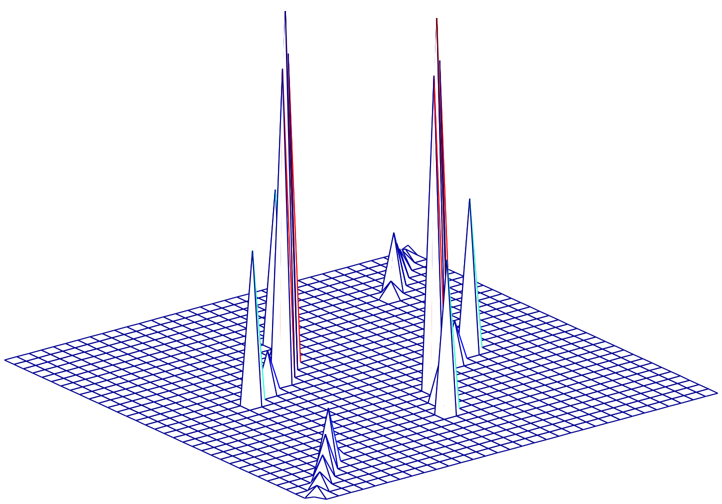
$(X, Y) \sim \mathcal{N}(0.I)$; $\mathbf{Prob}(X \leq 0, Y \leq 0) = 0.25$

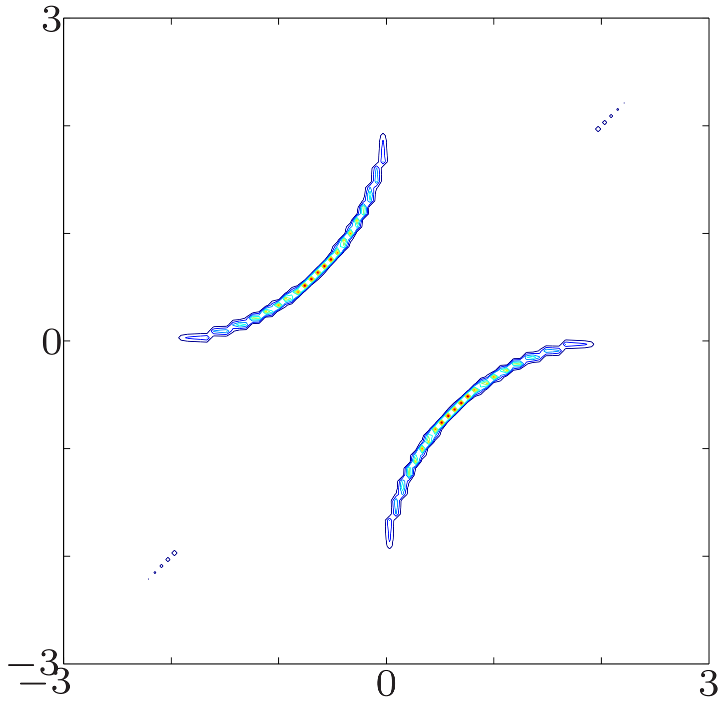




Distribution that minimizes $\mathbf{Prob}(X \leq 0, Y \leq 0)$

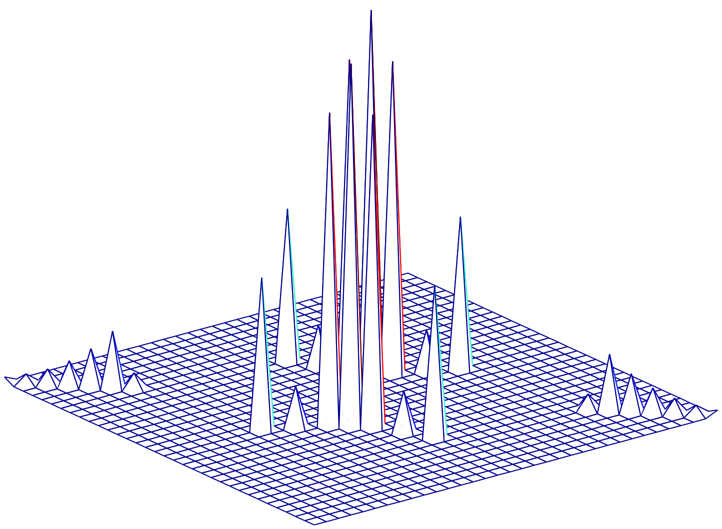
$\mathbf{Prob}(X \leq 0, Y \leq 0) = 0.03$

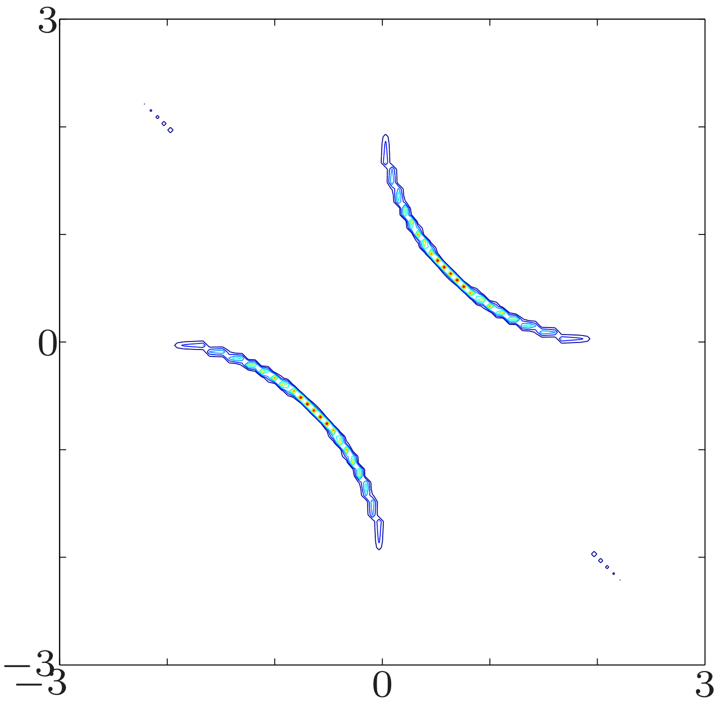




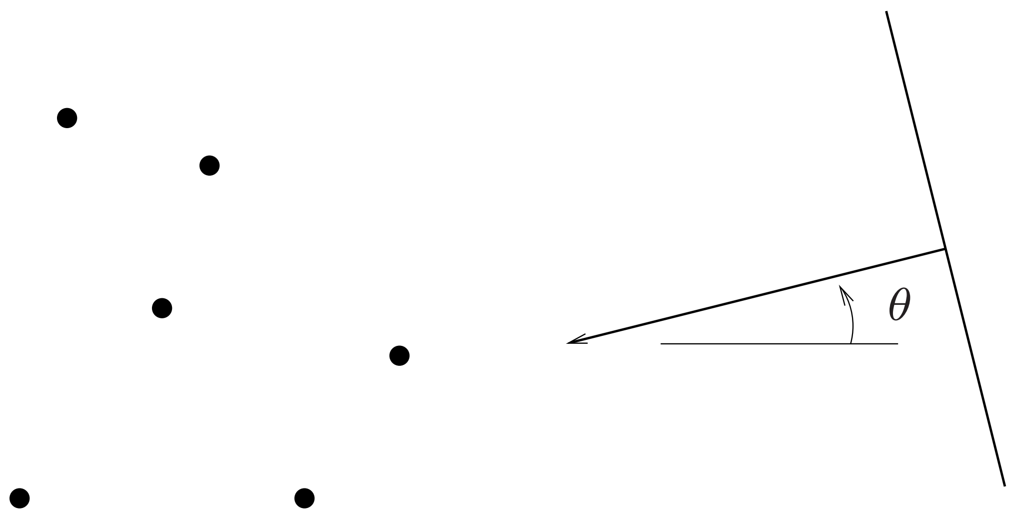
Distribution that maximizes $\mathbf{Prob}(X \leq 0, Y \leq 0)$

$\mathbf{Prob}(X \leq 0, Y \leq 0) = 0.47$





Antenna array beamforming



- n omnidirectional antenna elements in plane, at positions (x_i, y_i)
- unit plane wave ($\lambda = 1$) incident from angle θ
- ith element has (demodulated) signal $e^{j(x_i \cos\theta + y_i \sin\theta)}$ ($j = \sqrt{-1}$)

- combine antenna element signals using complex weights w_i to get antenna array output

$$y(\theta) = \sum_{i=1}^{n} w_i e^{j(x_i \cos\theta + y_i \sin\theta)}$$

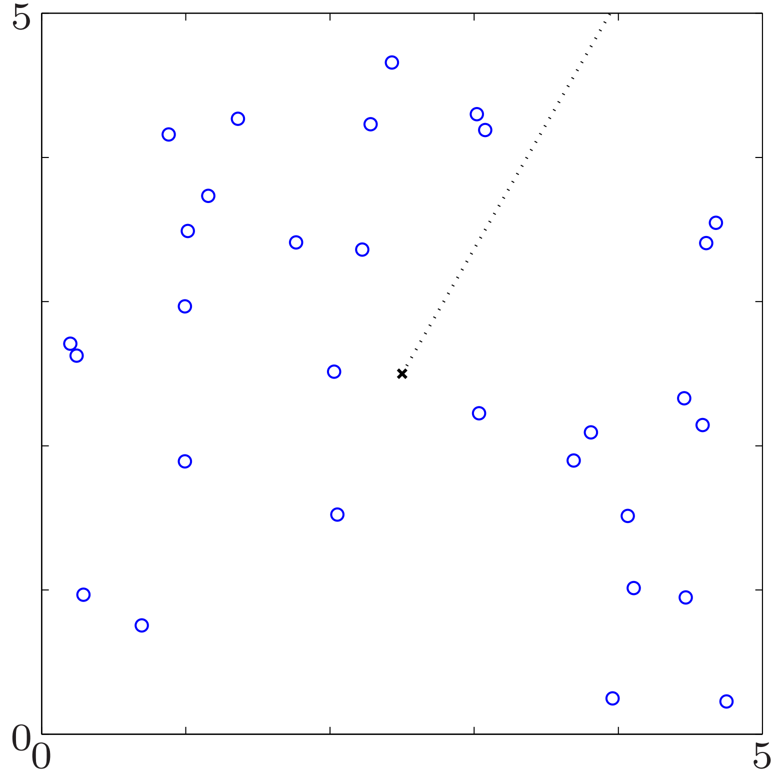
typical design problem:

choose $w \in \mathbf{C}^n$ so that

- $y(\theta_{\rm tar}) = 1$ (unit gain in target or look direction)
- $|y(\theta)|$ is small for $|\theta - \theta_{\rm tar}| \geq \Delta$ (2Δ is beamwidth)

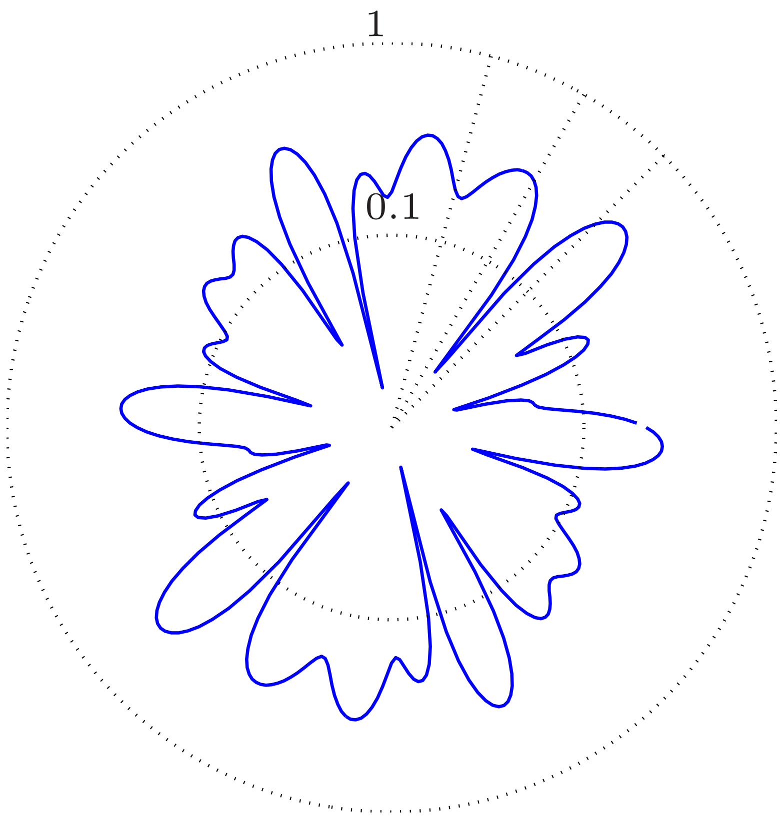
Example

$n = 30$ antenna elements, $\theta_{\text{tar}} = 60^\circ$, $\Delta = 15^\circ$ (30° beamwidth)



Uniform weights

$w_i = 1/n$; no particular directivity pattern



Least-squares (ℓ_2-norm) beamforming

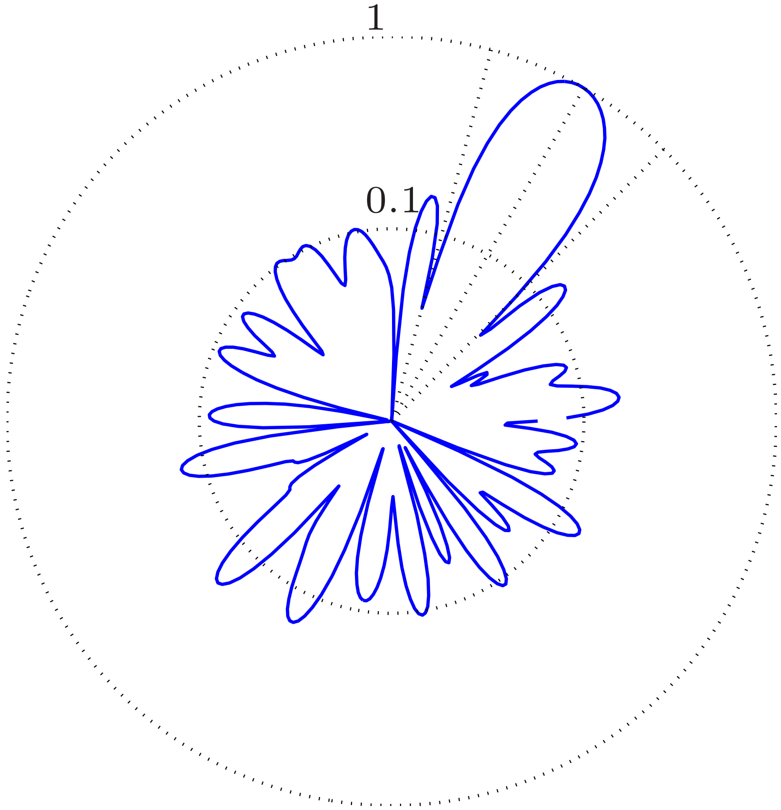
discretize angles outside beam ($i.e.$, $|\theta - \theta_{\rm tar}| \geq \Delta$) as $\theta_1, \ldots, \theta_N$;

solve least-squares problem

$$\begin{array}{ll} \mbox{minimize} & \left(\sum_{i=1}^N |y(\theta_i)|^2\right)^{1/2} \\ \mbox{subject to} & y(\theta_{\rm tar}) = 1 \end{array}$$

```
cvx_begin
    variable w(n) complex
    minimize ( norm( A_outside_beam*w ) )
    subject to
      a_tar'*w == 1;
cvx_end
```

Least-squares beamforming



Chebyshev beamforming

solve minimax problem

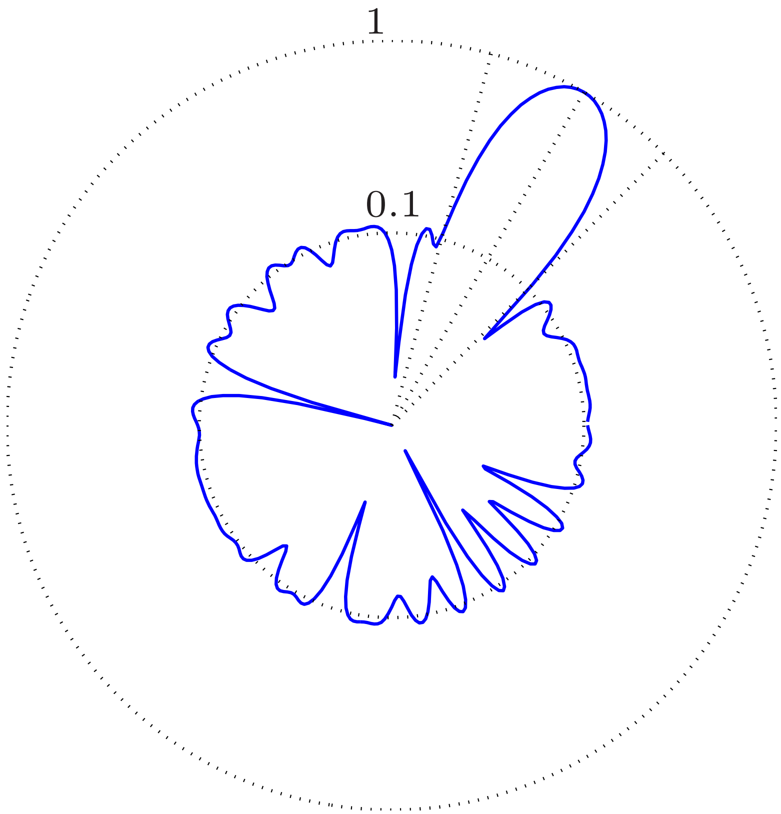
$$\begin{array}{ll} \text{minimize} & \max_{i=1,\ldots,N} |y(\theta_i)| \\ \text{subject to} & y(\theta_{\rm tar}) = 1 \end{array}$$

(objective is called sidelobe level)

```
cvx_begin
    variable w(n) complex
    minimize ( max( abs( A_outside_beam*w ) ) )
    subject to
      a_tar'*w == 1;
cvx_end
```

Chebyshev beamforming

(globally optimal) sidelobe level 0.11



ℓ_1-regularized logistic regression

logistic model:

$$\mathbf{Prob}(y = 1) = \frac{\exp(a^T x + b)}{1 + \exp(a^T x + b)}$$

- $y \in \{-1, 1\}$ is Boolean random variable (outcome)
- $x \in \mathbf{R}^n$ is vector of explanatory variables or features
- $a \in \mathbf{R}^n$, b are model parameters
- $a^T x + b = 0$ is neutral hyperplane
- linear classifier: given x, $\hat{y} = \mathrm{sgn}(a^T x + b)$

Maximum likelihood estimation

a.k.a. logistic regression

given observed (training) examples $(x_1, y_1) \ldots, (x_m, y_m)$, estimate a, b

maximum likelihood model parameters found by solving (convex) problem

$$\text{minimize} \quad \sum_{i=1}^{n} \operatorname{lse}\left(0, -y_i(x_i^T a + b)\right)$$

with variables $a \in \mathbf{R}^n$, $b \in \mathbf{R}$, where

$$\operatorname{lse}(u) = \log\left(\exp u_1 + \cdots + \exp u_k\right)$$

(which is convex)

ℓ_1-regularized logistic regression

find $a \in \mathbf{R}^n$, $b \in \mathbf{R}$ by solving (convex) problem

$$\text{minimize} \quad \sum_{i=1}^{n} \operatorname{lse}\left(0, -y_i(x_i^T a + b)\right) + \lambda \|a\|_1$$

$\lambda > 0$ is regularization parameter

- protects against over-fitting
- heuristic to get sparse a ($i.e.$, simple explanation) for $m > n$
- heuristic to select relevant features when $m < n$

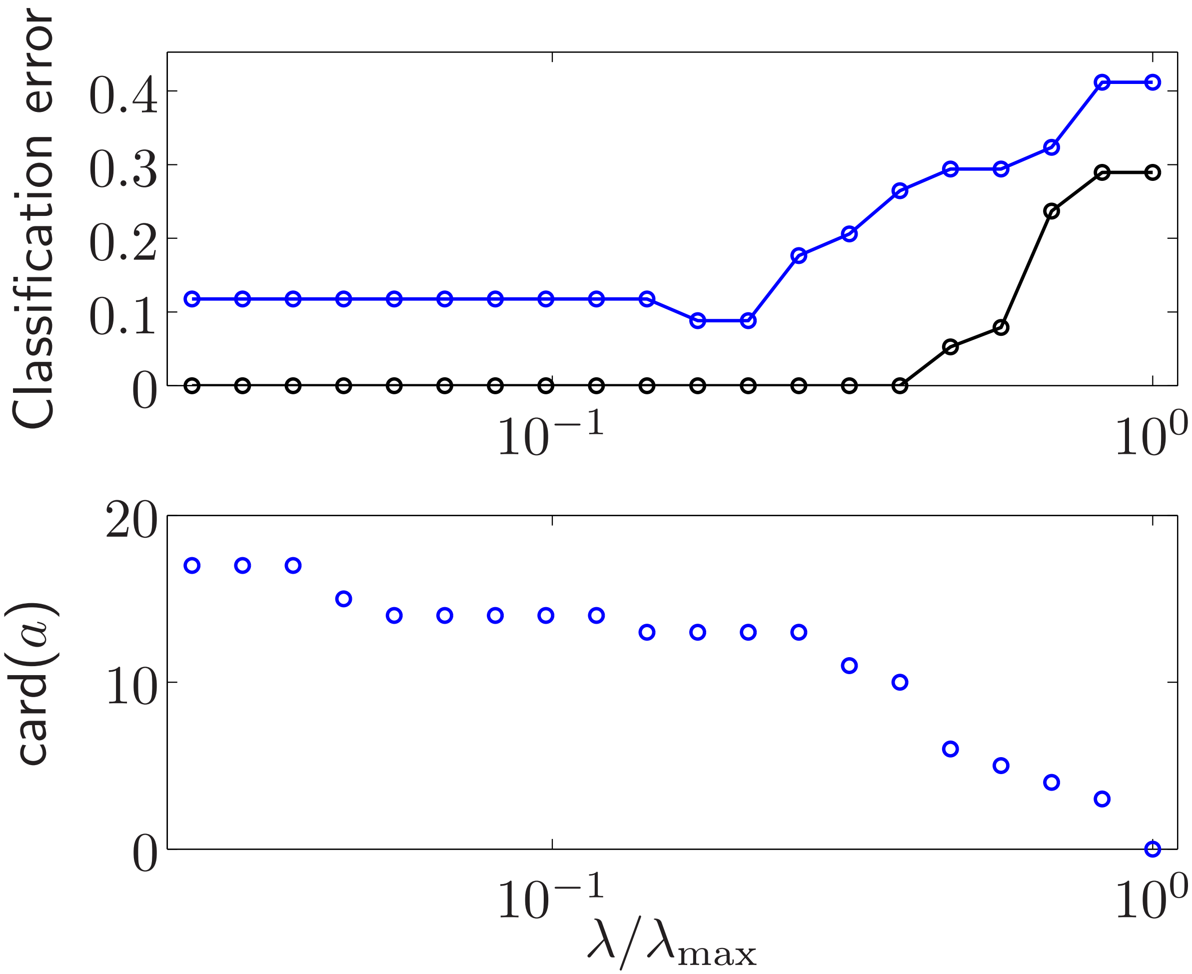
cvx code

```
cvx_begin
    variables a(n) b
    tmp = [zeros(m,1) -y.*(X'*a+b)];
    minimize ( sum(logsumexp(tmp')) + lambda*norm(a,1) )
cvx_end
```

Leukemia example

- taken from Golub et al, *Science* 1999
- $n = 7129$ features (gene expression data)
- $m = 72$ examples (acute leukemia patients), divided into training set (38) and validation set (34)
- outcome: type of cancer (ALL or AML)
- ℓ_1-regularized logistic regression model found using training set; classification performance checked on validation set

Results



Classification error
0.4
0.3
0.2
0.1
0
10^{-1}
10^{0}
20
card(a)
10
0
10^{-1}
10^{0}
$\lambda/\lambda_{\max}$

Final comments

- DCP formalizes the way we think convex optimization modeling should be done
- CVX makes convex optimization model development & exploration quite straightforward

References

- `www.stanford.edu/~boyd`
- `www.stanford.edu/~boyd/cvx`
- `www.stanford.edu/class/ee364`

or just google convex optimization, convex programming, `cvx`, . . .