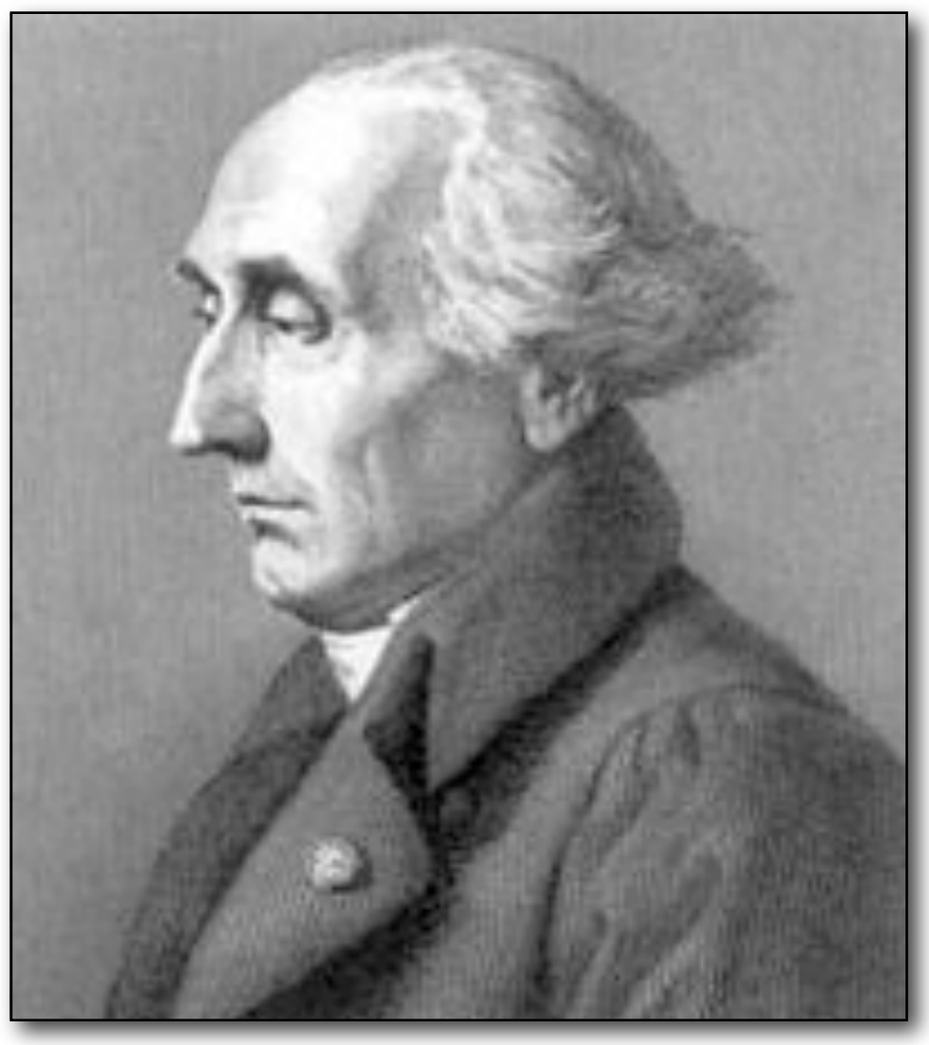
Lagrange multipliers

- Technique for turning constrained optimization problems into unconstrained ones
  - for intuition or for algorithm
- Useful in general
  - but in particular, leads to a famous ML method: the support vector machine



Recall: Newton's method

- $\min_x f(x) \quad \rightarrow \quad H(x)\Delta x + g(x) = 0$
  - $f: R^d \rightarrow R$
- Why: $f(x+\Delta x) \sim f + g^T\Delta x + \Delta x^T H \Delta x/2$
  - $f = f(x), g = g(x), H = H(x)$
  - set derivative wrt Δx to 0

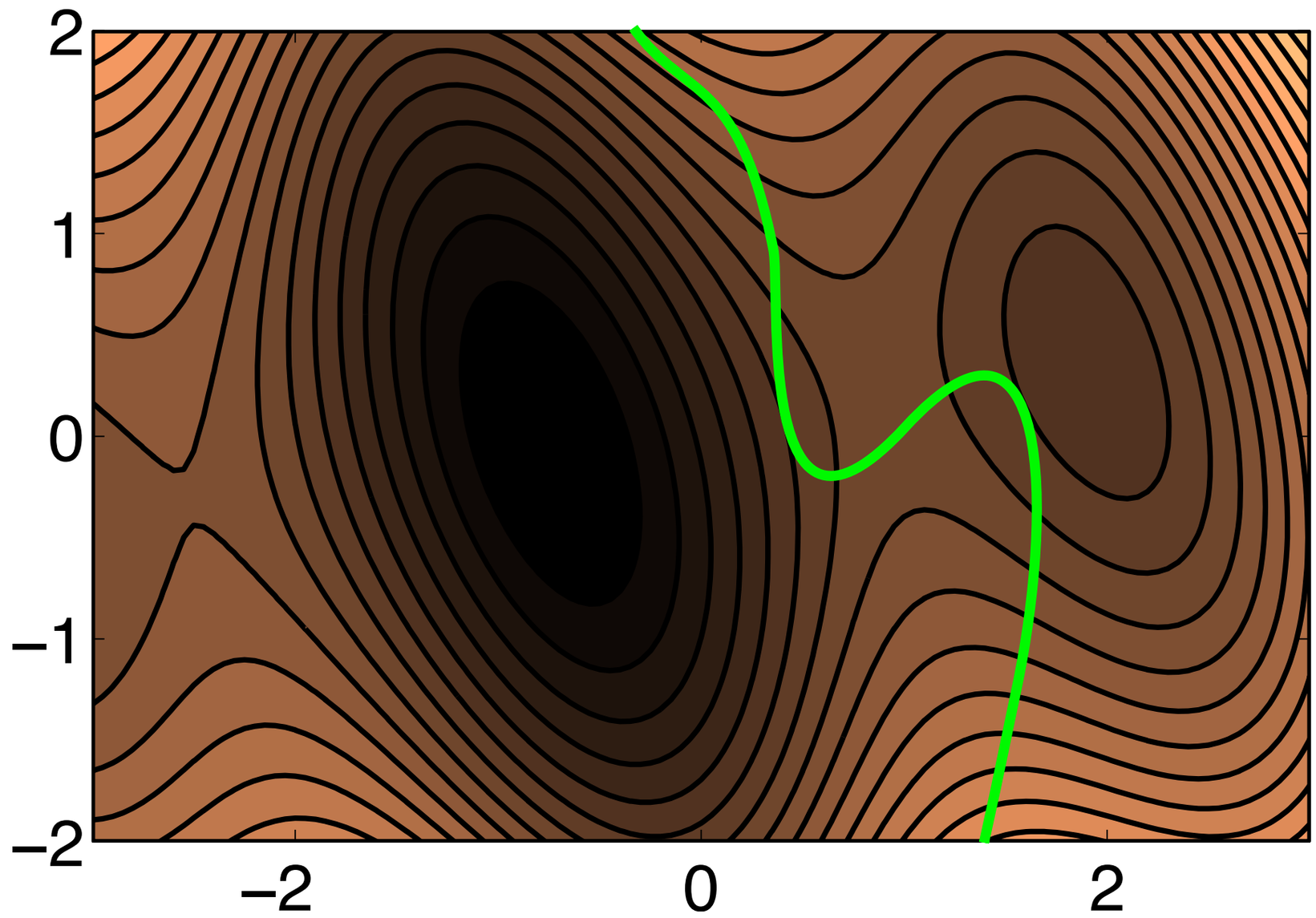
w/o constr: H(x) Δx + g(x) = 0
g(x) = gradient R^d->R^d
H(x) = Hessian R^d->R^{d*d}

why: f(x+Δx) ~ f+g'Δx+Δx'HΔx/2
f = f(x), g = g(x), H = H(x)
set derivative wrt Δx to 0:
0 = g + HΔx

ie, HΔx is predicted change in gradient, so use it to cancel g

Equality constraints

- min f(x) s.t. p(x) = 0



f(x): contours (darker = smaller)
p(x)=0: line

quiz: where are the local optima?
A: places where gradient f'(x) is normal to the curve (can't slide L or R to decrease fn)
i.e., f'(x) = lambda p'(x)
draw: they are places where contours of f are tangent to p=0 [except these include local maxima and saddle points as well as the desired minima]

lambda = "Lagrange multiplier" -- multiplies constraint normal to scale it to match gradient

Optimality w/ equality

- min f(x) s.t. p(x) = 0
  - f: $R^d \to R$ p: $R^d \to R^k$ ($k \le d$)
  - g: $R^d \to R^d$ (g = df/dx gradient of f)
- Useful special case: min f(x) s.t. Ax = b

def: C = {x | Ax = b} p(x) = Ax−b

How do we express g(x) ⊥ C?
  z ⊥ C iff z'(x−y) = 0 forall x,y in C
  idea: z = A' λ
  then z'(x−y) = λ' A(x−y)
   = λ'(b−b) = 0.
necessary & sufficient (count dimensions)

So, want g(x) = A' λ.
  ie, gradient = linear combo of rows of A
  one Lagrange multiplier per constraint

===
How do we know A' λ is a full basis? A' λ is a space of rank(A) dimensions; Ax = 0 is a space of nullity (A) dimensions; rank + nullity is the full dimension of the space, so we've accounted for every dimension as either free to vary under the constraint or orthogonal to the constraint.

More generally

- f: $R^d \to R$ p: $R^d \to R^k$ J: $R^d \to R^{k \times d}$ J = dp/dx

want p(x) = 0 (satisfy constraint)
and $g(x) = J(x)^T \lambda$ where $J_{ij} = dp_i/dx_j$
ie, gradient = lin. comb. of constraint normals

if p(x) = Ax – b then J(x) = A

These are “KKT conditions” or “first-order optimality conditions” [for equality-constrained optimization]

===

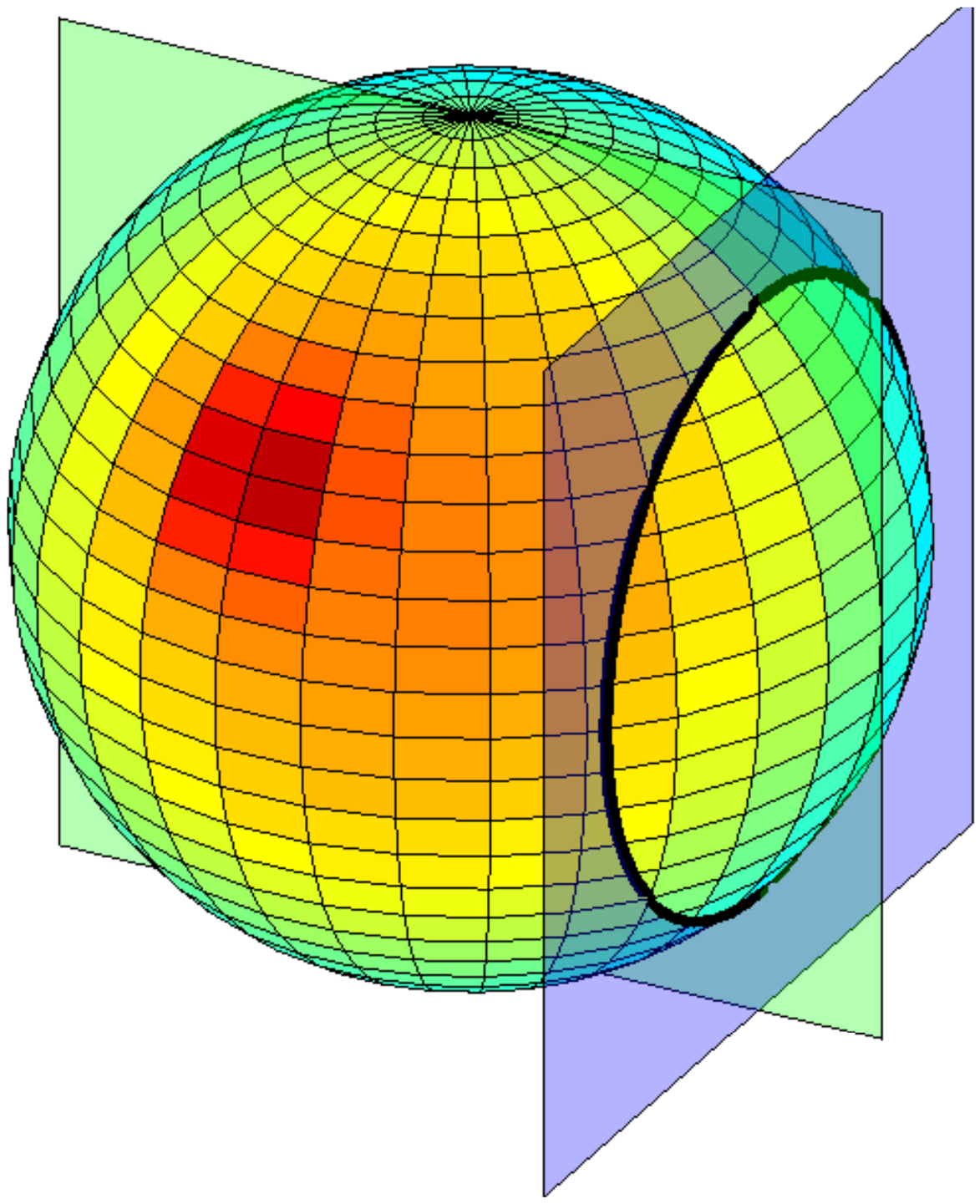
another way to think of it: cancel out the portion of gradient orthogonal to p(x)=0 using best λ. Remainder is projection of gradient onto constraint.

Picture

$$\max c^\top \begin{bmatrix} x \\ y \\ z \end{bmatrix} \text{ s.t.}$$

$$x^2 + y^2 + z^2 = 1$$

$$a^\top x = b$$



c: pointing up
constraints: sphere, blue plane
(intersection = dark circle)

Constraint normals: 2[x y z], a
So, at opt:
c = 2 lam1 [x y z] + lam2 a
x^2 + y^2 + z^2 = 1
a'x = b

(green plane = span of normals @ optimum)

===

max z s.t.
x^2 + y^2 + z^2 = 1
x = .7

opt: x = .7, y = 0, z = √.51
constraint normals: 2*[.7 0 √.51], [1 0 0]
lam2 = -lam1
lam2 = 1/(2√.51)

===

>> [x, y, z] = sphere(30); h = surfl(x, y, z); axis equal off; set(gca, 'fontsize', 24); h = patch(.7*[1 1 1 1], [1 1 -1 -1], [1 -1 -1 1], 'b'); set(h, 'facealpha', .3)

>> [ex, ey] = ellipse([0;0], eye(2), 50); r = sqrt(1-.7^2); line(.7*ones(size(ex)), ex*r, ey*r, 'linewidth', 3, 'color', 'k');

>> h = patch([1 1 -1 -1], [0 0 0 0], [1 -1 -1 1], 'g'); set(h, 'facealpha', .3)

Newton w/ equality

- min f(x) s.t. p(x) = 0
  - f: $R^d \to R$, p: $R^d \to R^k$

df/dx = g(x) p(x) = 0
dp/dx = J(x) $g(x) = J(x)^T\lambda$

- Now suppose:
  - dg/dx =
- Newton step:

Now suppose
dg/dx = H(x) [Jacobian of g = Hessian of f]
[still: dp/dx = J(x) [Jacobian of p], df/dx=g]
[sizes: H is d*d, J is k*d, g is d*1]
Taylor approx of p(x):
$p(x) + J(x)\Delta x = 0$
Taylor approx of g(x):
$H(x)\,\Delta x + g(x) = J(x)^T\lambda$
LHS: predicted gradient after update Δx
RHS: orthogonal to (approx) constraint

Newton step:
[H –J'; J 0] [Δx; λ] = [–g; –p]
N = [H –J'; J 0] is (k+d)*(k+d), PSD if H is

Useful special case

- Exact for quadratic:
  min $x^T Hx/2 + c^T x$ s.t. $Ax + b = 0$
  - $g(x) = Hx+c$, $J(x) = A$
  - so:

$p(x) = 0$
$g(x) = J(x)^T \lambda$

KKT becomes: $Ax+b=0$, $Hx+c = A' \lambda$
want $p(x) = 0$ (satisfy constraint)
and $g(x) = J(x)^T \lambda$ where $J_{ij} = dp_i/dx_j$

Newton becomes:

$H(x+\Delta x) + c'x = A'\lambda$
$A(x+\Delta x) + b = 0$

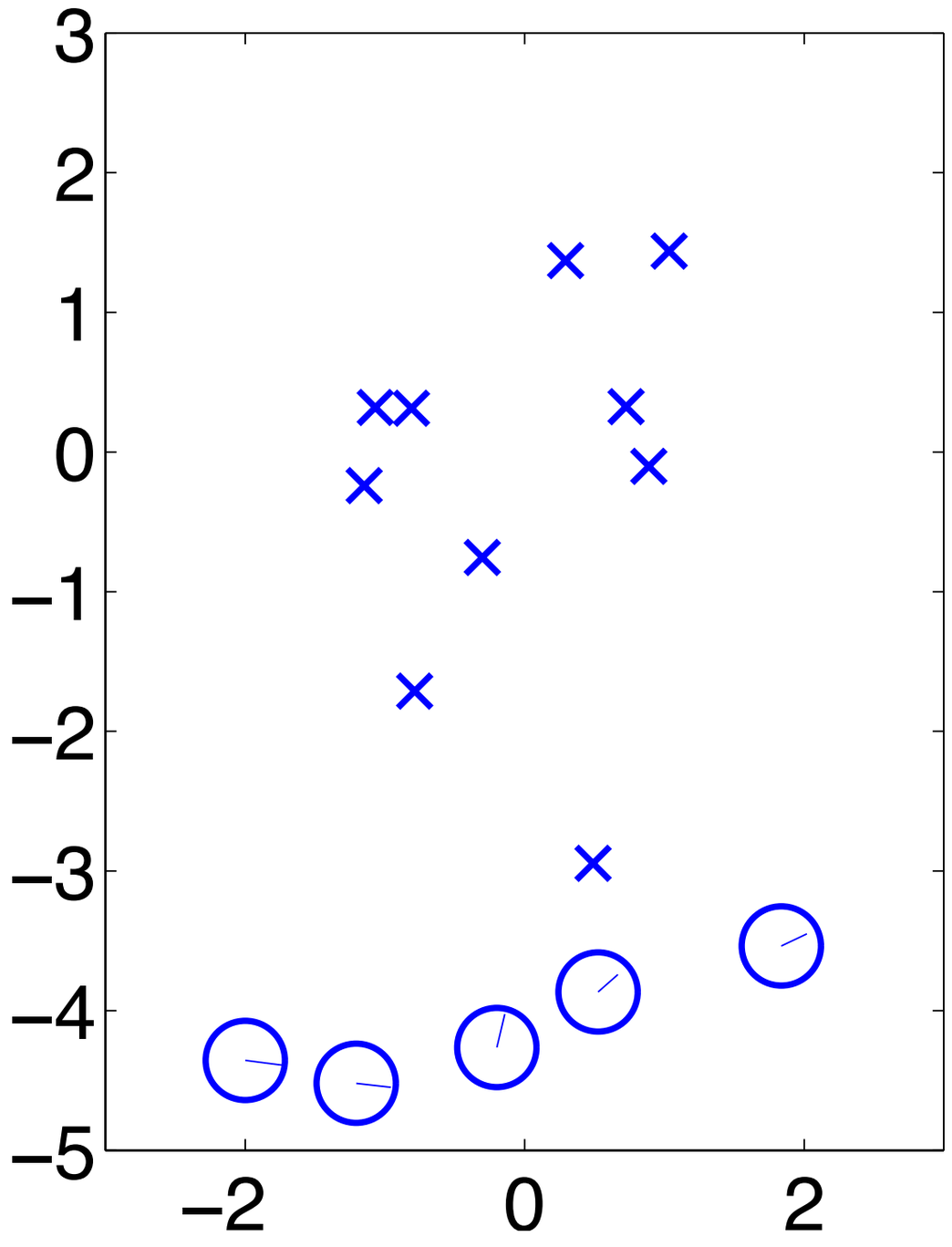
or

$[H\ -A';\ A\ 0]\ [\Delta x;\ \lambda] = [-(Hx+c);\ -(Ax+b)]$

(one step of Newton is enough to minimize a quadratic s.t. linear constraints)

Ex: bundle adjustment for SLAM

- Solve for:
  - Robot positions x_t, θ_t
  - Landmark positions y_k
- Given: odom., radar, vision, …
- Constraints:
  - observations consistent w/ inferred landmark/robot positions



xt, yk in R^2
theta_t in [-pi,pi]

example: distance measurements d_{kt}
||x_t - y_k||^2 = d_{kt}^2 + noise
(min |noise| goes in objective)

Lagrange for inequalities

- Even more useful than for equalities
- Review LPs/QPs first

Linear or quadratic programs

- n variables: $x = (x_1, x_2, \ldots, x_n)^T$
  - ranges: $[l_i, u_i]$
- Objective: min or max $\text{sum}_i\ c_i\ x_i$
  - optionally: ... + $\text{sum}_{ij}\ q_{ij}\ x_i\ x_j$
- m constraints
  - for j=1..m, $\text{sum}_i\ a_{ij}\ x_i = b_j$ (or ≥ or ≤)
- Example:

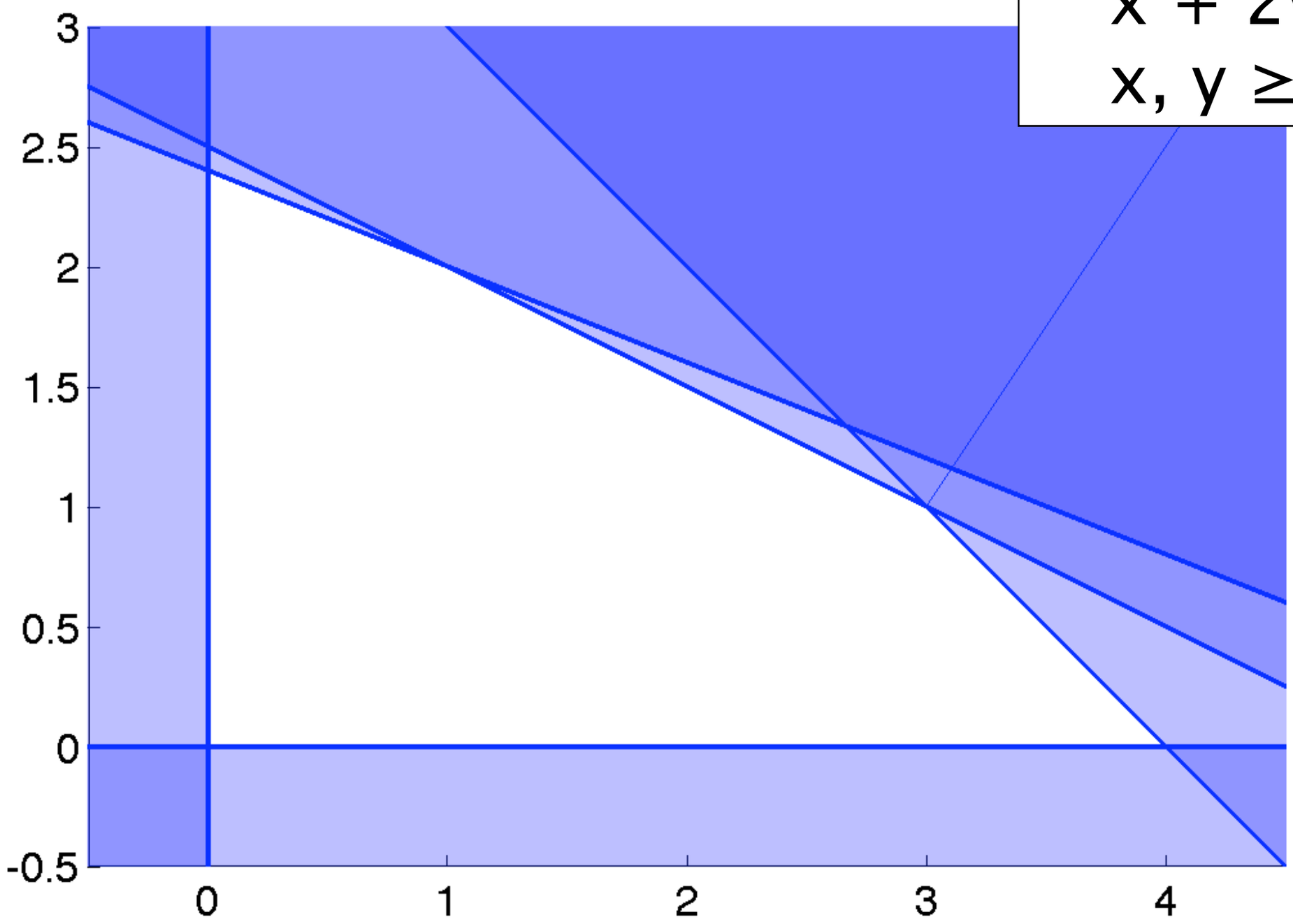
$$\begin{aligned}\max\ & 2x+3y \text{ s.t.}\\ & x + y \le 4\\ & 2x + 5y \le 12\\ & x + 2y \le 5\\ & x, y \ge 0\end{aligned}$$

variables x_i
ranges x_i in [l_i, u_i]
or x_i in [l_i, infty)
or x_i in (infty, u_i]
or x_i in (-infty, infty) (called “free” or “not bounded”)
obj: min or max sum_i c_i x_i
optionally + (1/2)sum_ij q_ij x_ix_j
only difference LP vs QP
m constraints
sum_i a_ij x_i = b_j
or ≤ or ≥

Ex:
max 2x+3y s.t
x + y ≤ 4
2x + 5y ≤ 12
x + 2y ≤ 5
x, y ≥ 0
this form is *inequality form* (all constraints are inequalities, all variables are unbounded -- note trivial-but-useful distinction between x in [0, infty] and x ≥ 0)

Sketching an LP

max 2x+3y s.t.
x + y ≤ 4
2x + 5y ≤ 12
x + 2y ≤ 5
x, y ≥ 0



terminology: feasible, optimal, active/tight

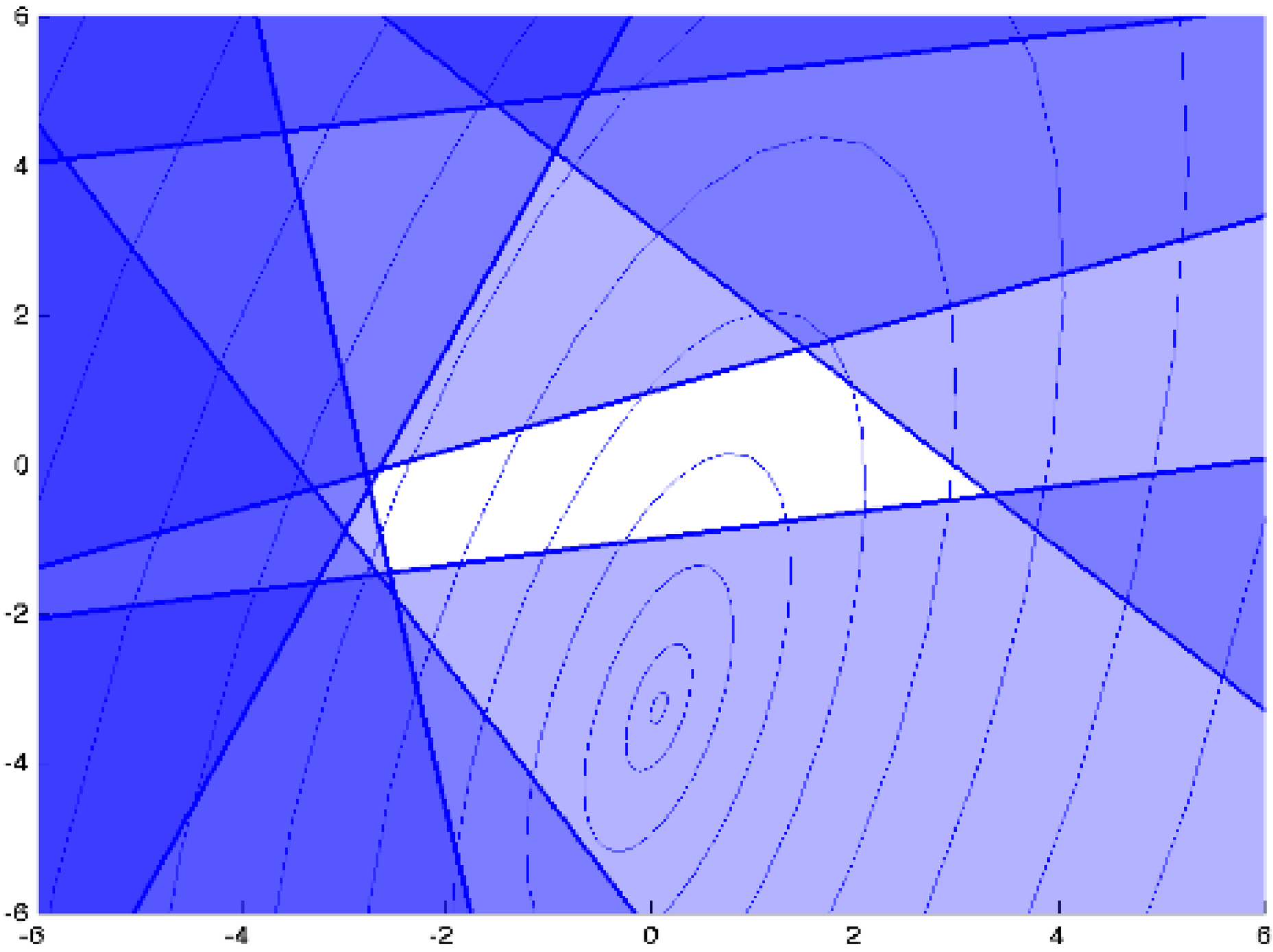
infeasible, suboptimal, inactive/loose
infeasible points are neither optimal nor subopt

for sketch:

x + y ≤ 4	intercepts Y 4, X 4
2x + 5y ≤ 12	Y 2.4, X 6
x + 2y ≤ 5	Y 2.5, X 5
x, y ≥ 0	

```
>> clf; drawhalfspace([1 0; 0 1; -1 -1; -2 -5; -1 -2], [0; 0; 4; 12; 5], 10); axis equal; axis([-0.5 4.5 -0.5 3]); set(gca, 'FontSize', 16); line([3 5], [1 4])
```

Sketching a QP



note objective, constraints

e.g., top long constr bounding feas region:

$.4x - y \leq 0.97$

mark optimum -- note not at corner

Matrix notation

- For a vector of variables v and a constant matrix A and vector b,
  - Av ≤ b [componentwise]
- Objective: $c^T v + v^T Q v/2$
- E.g.:

max 2x+3y s.t.
x + y ≤ 4
2x + 5y ≤ 12
x + 2y ≤ 5
x, y ≥ 0

$$A = \begin{pmatrix} 1 & 1 \\ 2 & 5 \\ 1 & 2 \\ -1 & 0 \\ 0 & -1 \end{pmatrix} \quad b = \begin{pmatrix} 4 \\ 12 \\ 5 \\ 0 \\ 0 \end{pmatrix} \quad c = \begin{pmatrix} 2 \\ 3 \end{pmatrix}$$

Write: v = [x;y] Av ≤ b

similarly ≥, =

===

```
A = \left(
\begin{array}{rr}
1 & 1\\
2 & 5\\
1 & 2\\
-1 & 0\\
0 & -1
\end{array}
\right)
\quad
b = \left(
\begin{array}{r}
4 \\ 12 \\ 5 \\ 0 \\ 0
\end{array}
\right)
\quad
c = \left(
\begin{array}{rr}
2 \\ 3
\end{array}
\right)
```

Transforming linear constraints

- Getting rid of inequalities (except variable bounds)

$$x + y \leq 4$$

- Getting rid of equalities

$$x + 2y = 4$$

slack
$x + y \leq 4$

double ineq.
$x + 2y = 4$

Transforming linear constraints

- Getting rid of free vars

$$\max x + y \text{ s.t.}$$
$$2x + y \le 3$$
$$y \ge 0$$

- Getting rid of bounded vars

$$x \in [2, 5]$$

free vars: diff of 2 bounded ones
max x + y s.t.
2x + y ≤ 3
y ≥ 0
max x – x’ + y s.t.
2(x – x’) + y ≤ 3
x, x’, y ≥ 0

bounded: x in [2, 5] ==> 2 ineqs

Normalizing LPs

- Standard form:
  - all variables are nonnegative
  - all constraints are equalities
  - max or min $c^T x$ s.t. $Ax = b, x \geq 0$
  - (optionally: A has full row rank)
- Inequality form:
  - all variables are free
  - all constraints are inequalities
  - max or min $c^T x$ s.t. $Ax \geq b$
  - (optionally: A has full column rank)

std form: A has more columns than rows, full row rank (else some constraints are redundant, drop)

ineq form: A has more rows than columns, full column rank (else some variables are redundant, fix to 0)

Now back to Lagrange

- Develop Lagrange multiplier technique for LP/QP
- Leads to an important idea: ***duality***
  - transform one optimization problem ("primal") to an equivalent but often-very-different-looking one ("dual")
  - taking dual twice gets back to primal
- Why do we care?
  - dual can be much easier (or harder) to solve than primal
  - if easier: profit!
  - upcoming example: SVMs

Suppose we're lazy

- A "hard" LP (in inequality form):
  - min $x + y$ s.t. $x + y \geq 2$ $x \geq 0$ $y \geq 0$

Don't want to find the exact solution to this "hard" LP, just get an approximate solution with bounds on its quality

Suppose we have a feasible solution (say (2, 2)). How do we know it's good?

Upper bound on opt is easy: our feasible point gives 2+2=4

How do we get a lower bound? Look at first constraint.

OK, we got lucky

- What if it were:
  - min $x + 3y$ s.t. $x + y \geq 2$ $x \geq 0$ $y \geq 0$

$(x+y) \geq 2$ $y \geq 0$

mult latter by 2, add to get

$x + 3y \geq 2$

done!

How general is this?

- What if it were:
  - min $px + qy$ s.t. $x + y \geq 2$ $x \geq 0$ $y \geq 0$

pick: a, b, c ≥ 0 “Lagrange multiplier” or “dual variable”
write a (x + y - 2) + bx + cy ≥ 0
collect: (a+b) x + (a + c) y ≥ 2a

to get a bound, need a + b = p a + c = q
any +ve abc satisfying these give bound. What is bound? 2a
Which bound do we want? Tightest (i.e., largest).
So, max 2a -- this is an LP! Called “dual” LP

note: it’s in standard form
  always true (dual of ineq form is in standard form)
  reverse is also true (dual of std is ineq)
  this is special case of “take dual twice, get back where we started”

Equality constraints

- Note = constraint
  - min $x - 2y$ s.t. $x + y \geq 2$ $x \geq 0$ $y \geq 0$ $3x + y = 2$

For sketch: intercepts x = 2/3 and y = 2

a (x + y - 2) + b x + c y + d (3x + y - 2) ≥ 0
a, b, c ≥ 0
d free

collect: (a + b + 3d) x + (a + c + d) y ≥ 2a + 2d
so: (a+b+3d) = 1, (a+c+d)=–2, max 2a+2d

Quadratic program

- min $c^T x + x^T H x/2$ s.t. $Ax \geq b$

$\lambda \geq 0$ ==> $\lambda^T(b-Ax) \leq 0$
suppose we guess:
$A^T\lambda = c + Hx$
Then: $\lambda^T b - (c+Hx)^T x \leq 0$
or: $\lambda^T b - x^T H x/2 \leq c^T x + x^T H x/2$
bound on objective!
==> dual is
max $-x^T H x/2 + \lambda^T b$
s.t. $A^T\lambda = c + Hx$, $\lambda \geq 0$

===

How could we have guessed $A^T\lambda = c + Hx$?
obj $\geq c^T x + x^T H x/2 + \lambda^T(b-Ax)$
$\geq \min_x c^T x + x^T H x/2 + \lambda^T(b-Ax)$
set gradient wrt x to 0:
$0 = c + Hx - A^T\lambda$
other guesses could still lead to bound, but wouldn't be as tight

Duality summary

- One Lagrange multiplier per constraint
  - =, ≥ yield multipliers that are free, ≥0
- Use "multiplier * constraint ≤ 0" to get a bound on objective of primal program
  - to get bound, constrain vector of multipliers
- Dual program: search for best bound

KKT

- Bound was: $\lambda^T b - x^T H x/2 \le c^T x + x^T H x/2$
- Suppose: $\lambda^T b - x^T H x/2 = c^T x + x^T H x/2$
  - and $A^T\lambda = c + Hx$, $\lambda \ge 0$, $Ax \ge b$
- For LP have H=0:

In this case we found a lower bound that's tight
i.e., we know the optimal solution
these are KKT conditions (= first-order optimality) for inequality-constrained optimization

special case: for LP, H=0
$A^T\lambda = c$, $\lambda \ge 0$, $Ax \ge b$
$\lambda^T b = c^T x$
rearrange:
$\lambda^T b = \lambda^T A x$
$\lambda^T(Ax-b) = 0$
note: λ and $Ax-b$ are componentwise positive
so, for each i, at most one of λ_i, $[Ax-b]_i$ is +ve
"complementary slackness"

Connect back to Newton

- If we start from a QP with nothing but equality constraints and free variables, KKT is same system of equations we had for Newton

min x'Hx/2 + c'x s.t. Ax = b
KKT gives: $A^T\lambda$ = c + Hx, Ax = b

or can derive directly:
x'Hx/2 + c'x =
    x'Hx/2 + c'x + lam'(b-Ax)
grad_x: Hx + c - A'lam = 0
grad_lam: Ax = b
[H -A'; A 0][x; lam] = [-c; b]

Even better: cone programs, etc.

- Epigraph of a function

- Sublevel set ("below set") of a function

- f convex ⇔ epi f convex
- f convex ⇒ below(f, c) convex
  - is ⇐ true?

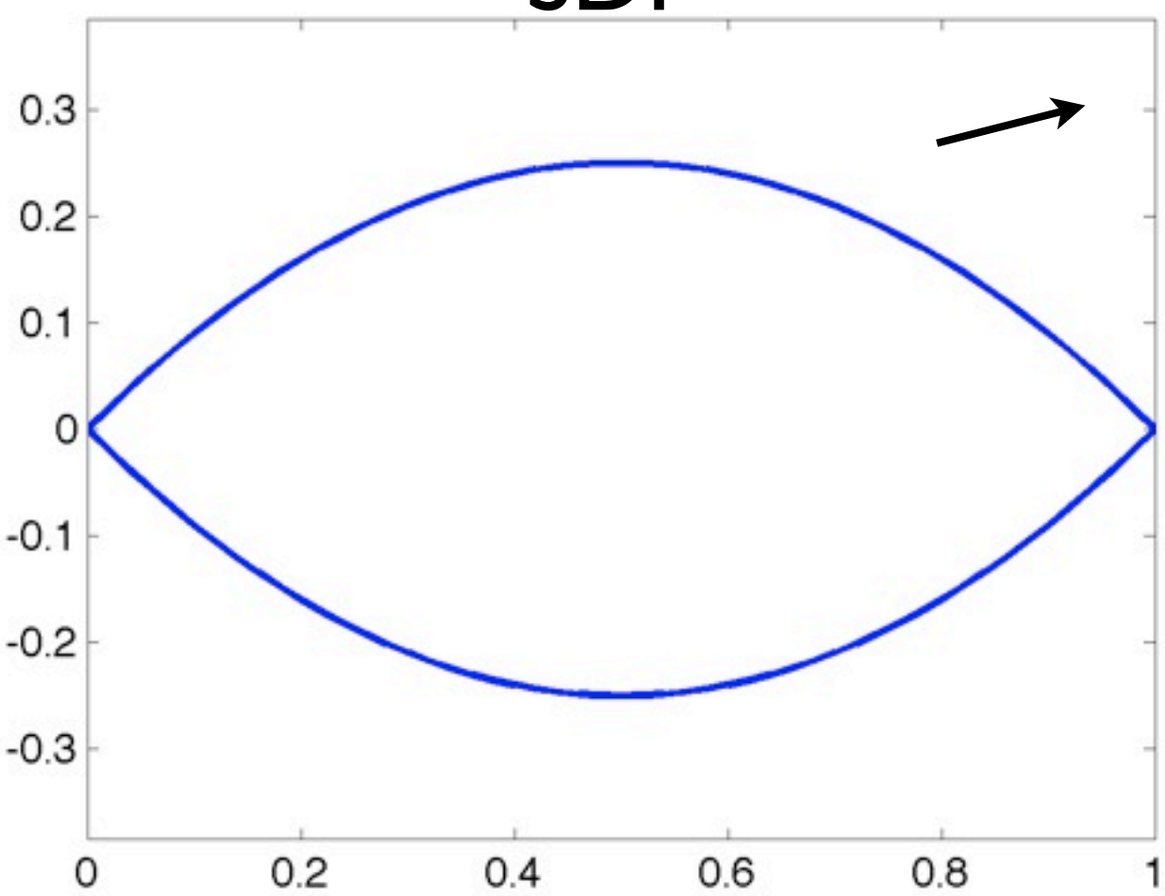
epi f = { (x, y) | y ≥ f(x) }

{ x | f(x) ≤ c }
intersect halfspace w/ epi f, forget about y coordinate

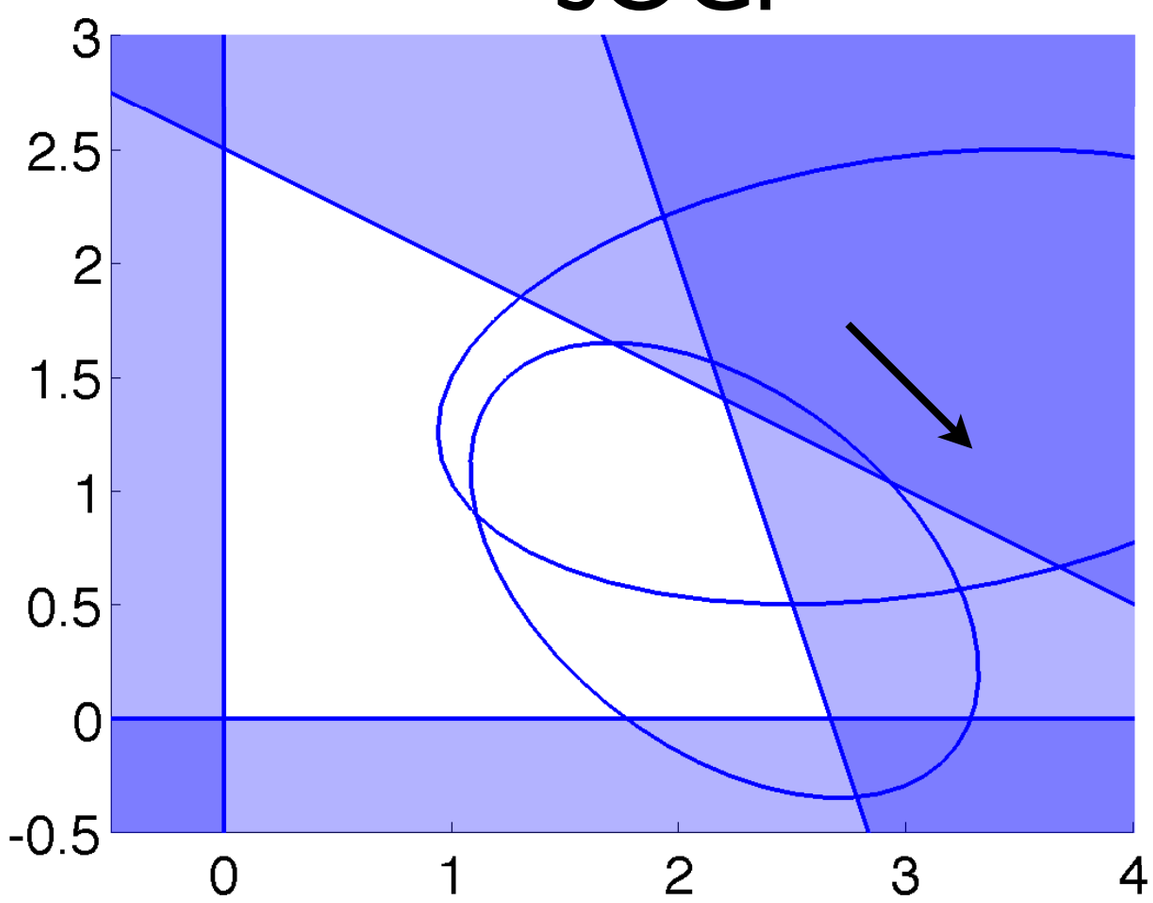
Q: no! this is quasiconvexity

Cone programs

SDP



SOCP



min $c^T x$ s.t. $Ax = b$, x in K_1, x in K_2, …

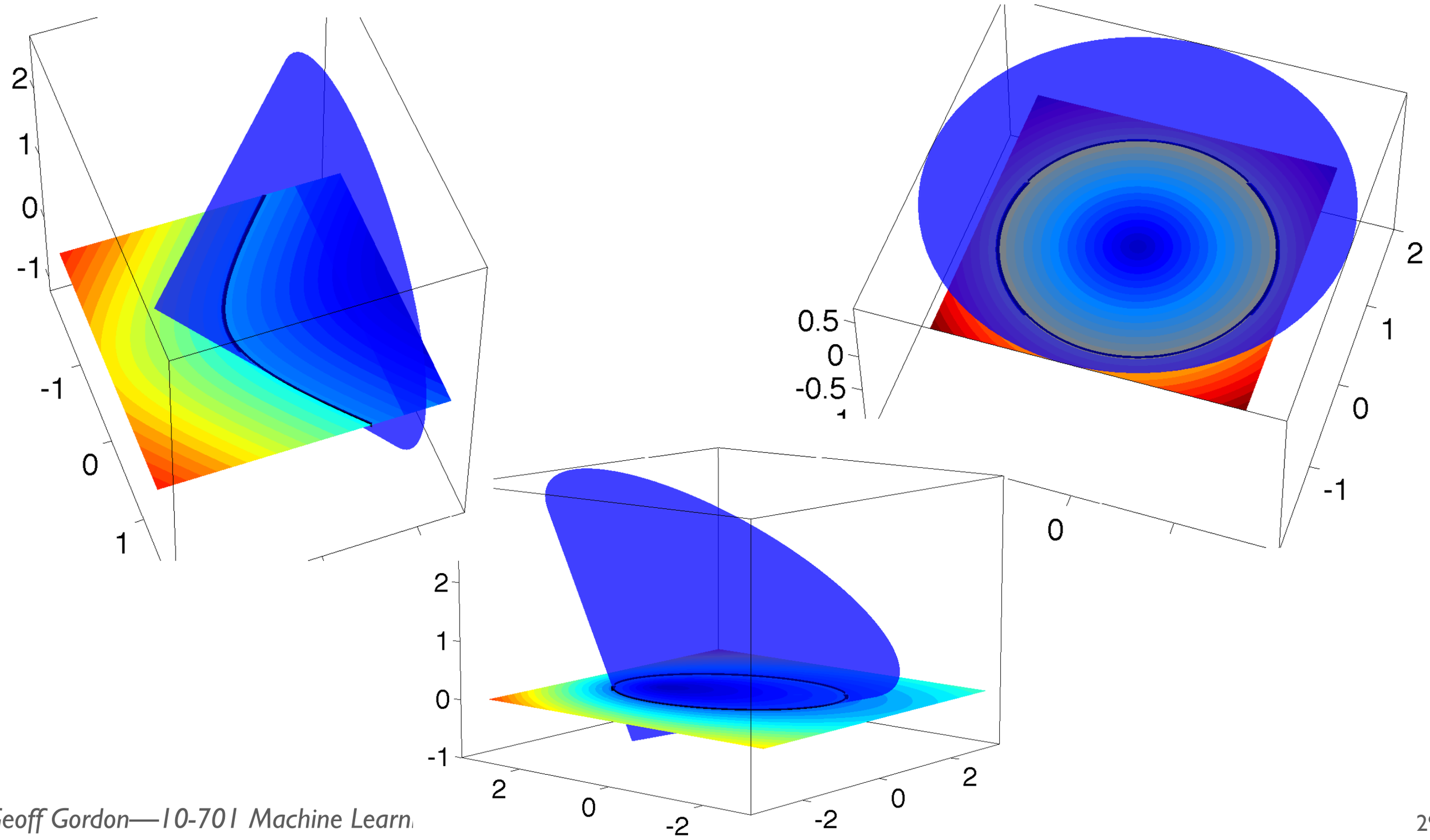
K_i a convex cone: orthant, SOC, PSD cone, …

may have more than one cone constraint

each cone can optionally constrain subset of vars

$R \times SOC \times R_+ \times R \times R_+$

Conic sections



can get: circle, ellipse, hyperbola (just one branch), parabola (not shown)

General convex programs

more generally, objective/constraints may be convex fns

min f(x) s.t. g_i(x) ≤ 0, Ax = b

we know constraints are convex because sublevel sets are convex (and intersection of convex sets is convex)

there is a duality theory for cone and general convex programs, analogous to LP/QP duality -- we won't use it, but be aware of it

Convex programs in ML

- LP:
  - single-source shortest paths
  - maximal matching
  - MDP planning
- QP:
  - LASSO
  - SVM
  - Robust (Huber) regression
- other:
  - matrix completion
  - group lasso
  - graph lasso
  - manifold learning
  - max-cut relaxation
  - portfolio optimization

Algorithms for LP/QP/SDP/…

- {accelerated, stochastic, ∅} proximal gradient
- simplex (LP only)
- interior point
- Avoid implementing simplex or IP by yourself if at all possible! CMU has CPLEX licenses…

algorithmic support is best for cone programs and their special cases

there are other choices besides these too (e.g, ADMM, mirror descent); above are just the main categories