Administrivia

- [http://groups.google.com/group/10725-f12](http://groups.google.com/group/10725-f12)
Administrivia

• Prerequisites: no formal ones, but class will be fast-paced
• Algorithms: basic data structures & complexity
• Programming: we assume you can do it
• Linear algebra: matrices are your friends
• ML/stats: source of motivating examples
• Most important: formal thinking
Administrivia

- Coursework: 5 HWs, scribing, midterm, project
- Project: use optimization to do something cool!
  - groups of 2–3 (no singletons please)
  - proposal, milestone, final poster session, final paper
- Final poster session: Tue or Wed, Dec 11 or 12, starting at about 3PM in NSH atrium, lasting 3 hrs
Administrivia

• Scribing
  ▸ multiple scribes per lecture (coordinate one writeup); required to do once during term
  ▸ sign up now to avoid timing problems

• Late days: you have 5 to use wisely
  ▸ in lieu of any special exceptions for illness, travel, holidays, etc.—your responsibility to allocate
  ▸ some deadlines will be non-extendable
Administrivia

- Working together
  - great to have study groups
  - always write up your own solutions, **closed** notes
  - disclose collaborations on front page of HW
Administrivia

• Office hours
• Recitations: none this week
• Audit forms: please audit r.t. just sitting in
  ‣ except: postdocs & faculty welcome to sit in
• Waitlist: there shouldn’t be one
• Videos
Most important

- Work hard, have fun!
Optimization example

• Simple economy: m agents, n goods
  ‣ each agent: production $p_i \in \mathbb{R}^n$, consumption $c_i \in \mathbb{R}^n$

• Cost of producing $p$ for agent $i$:

• Utility of consuming $c$ for agent $i$: 

$d_i(c)$

$s_i(p)$
Walrasian equilibrium

\[
\max \sum_i [d_i(c_i) - s_i(p_i)] \quad \text{s.t.} \quad \sum_i p_i = \sum_i c_i
\]

- Idea: put price \( \lambda_j \) on good \( j \); agents optimize production/consumption independently
  - high price \( \rightarrow \) produce \( \uparrow \), consume \( \downarrow \)
  - low price \( \rightarrow \) produce \( \downarrow \), consume \( \uparrow \)
  - “just right” prices \( \rightarrow \) constraint satisfied
Algorithm: tâtonnement

\[
\max \sum_i [d_i(c_i) - s_i(p_i)] \text{ s.t. } \sum_i p_i = \sum_i c_i
\]

\[
\lambda \leftarrow [0 \ 0 \ 0 \ ...]^T
\]

for \( k = 1, 2, \ldots \)

- each agent solves for \( p_i \) and \( c_i \) at prices \( \lambda \)
- \( \lambda \leftarrow \lambda + t_k(c - p) \)
Results for a random market

produces/consumes

prices
Why is tâtonnement cool?

• Algorithm is nearly obvious, given setup
  ‣ Leon Walras (1874), based on ideas of Antoine Augustin Cournot (1838)

• But analysis (Arrow and Debreu, 1950s) is subtle: needs concepts from later in this course
  ‣ duality, dual decomposition, convergence rates of gradient descent

• Variants need even more subtlety
“Typical” problem

- Minimize s.t.

- e.g.: $f()$ and $g_i()$ all linear:
- e.g.: $f()$ and $g_i()$ all convex:
- e.g.: $f()$ linear, $g_i()$ is $-\min(\text{eig}(\text{reshape}(x, k, k)))$: 
Ubiquitous (and pretty cool)

- LPs at least as old as Fourier
- first practical algorithm: simplex (Dantzig, 1947)
  - for a long time, best runtime bounds were exponential, but practical runtime observed good
- many thought LPs were NP-hard
- Spielman & Teng (2002): simplex solves “most” LPs in poly time
- LPs are P-complete: “hardest” poly-time problem
Optimization for ML & stats

• Lots of ML & stats based on optimization
  
• Exceptions?
  
• Advantages
Choices

- Set up problem
- Transformations: duality, relaxations, approximations
- Algorithms:
  - first order, interior point, ellipsoid, cutting plane
  - smooth v. nonsmooth v. some combination
  - eigensystems
  - message passing / relaxation

usually many choices, widely different performance (runtime, solution quality, ...)}
Consequences

- First order (gradient descent, FISTA, Nesterov’s method) v. higher order (Newton, log barrier, ellipsoid, affine scaling)
  - # iters poly in $1/\epsilon$ vs. in $\log(1/\epsilon)$
  - cost of each iteration: $O(n)$ or less, vs. $O(n^3)$ or so
- Balanced (#constrs $\approx$ #vars) or not?
  - e.g., ellipsoid handles #constrs $= \infty$
Consequences

• Sparsity? Locality? Other special structure?
  ‣ in solution, in active constraints, in matrices describing objective or constraints
• E.g., $Ax = b$: how fast can we compute $Ax$?
• E.g., simplex vs. log barrier
Consequences

- What degree of "niceness"?
  - differentiable, strongly convex, self-concordant, submodular

- Can we split $f(x) = g(x) + h(x)$?

- Is $f(x)$ "close to" a smooth fn?

- Care more about practical implementation or analysis?
Some more examples

- Image segmentation
- Perceptron, SVM
- MPE in graphical model
- Linear regression
- Lasso (group, graphical, …)
- Parsing, grammar learning
- Sensor placement in a sensor network

- Equilibria in games (CE, EFCE, polymatrix)
- Maximum entropy
- Network flow
- TSP
- Experimental design
- Compressed sensing
- …
Example: playing poker


- Problem: compute a minimax equilibrium

- Even this simple game has $2^{26}$ strategies/player

- We reduce to an LP with ~100 variables

- Similar methods have been used for competition-level 2-player limit Texas Hold’em
  - abstract the game by clustering information sets
  - buy a really big workstation, run for days
Dynamic walking

http://groups.csail.mit.edu/locomotion/movies/LittleDog/MIT_dynamic_short.f4v

[Schkolnik, Levashov, Manchester, Tedrake, 2010]