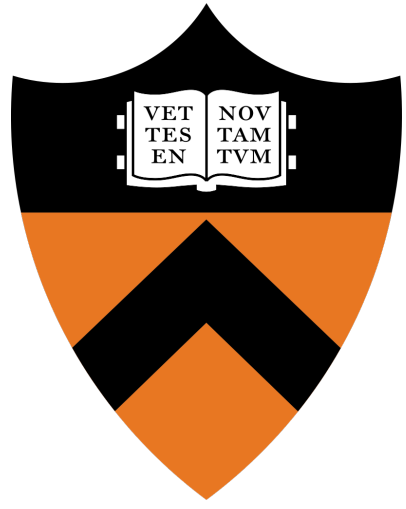
Introduction



Yuxin Chen

Princeton University, Fall 2019

Surge of data-intensive applications

Widespread applications in large-scale data science and learning

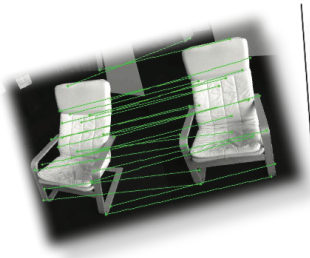
2.5 exabytes of data
are generated every day (2012)

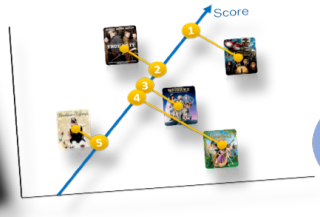
exabyte $\rightarrow$ *zettabyte* $\rightarrow$ *yottabyte...??*

limited processing ability
(computation, storage, ...)

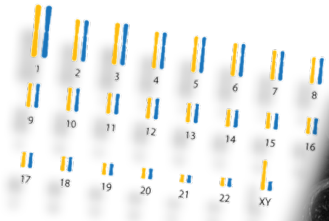
Optimization has transformed algorithm design

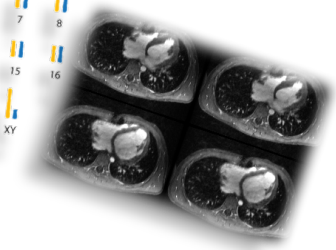
Optimization has transformed algorithm design

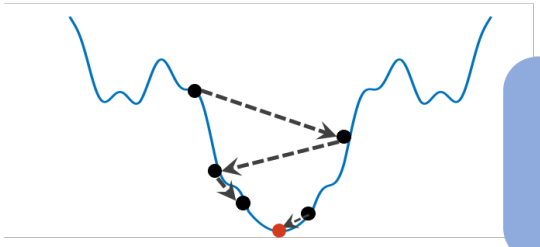




Learning / Inference

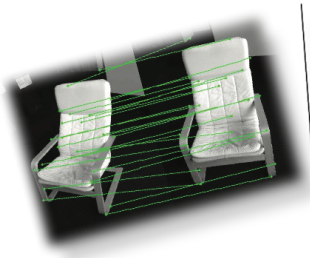




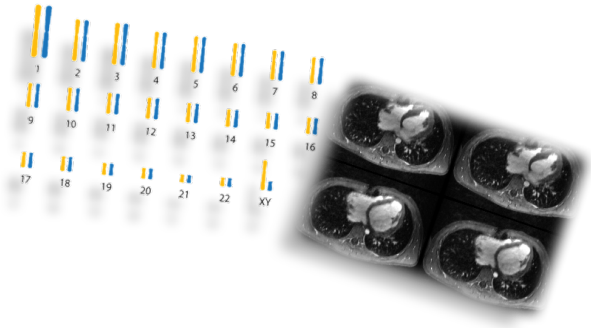


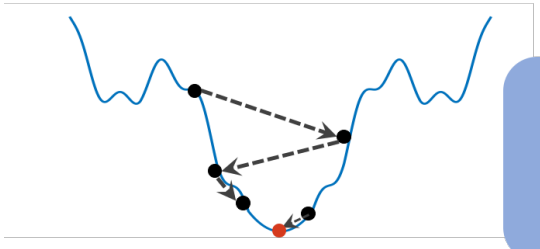
Optimization problems
(e.g. empirical risk minimization, maximum likelihood estimation)

Optimization has transformed algorithm design



Learning / Inference

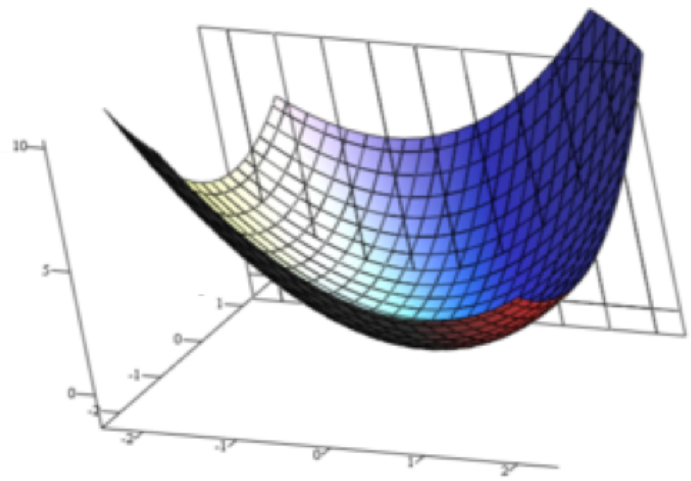


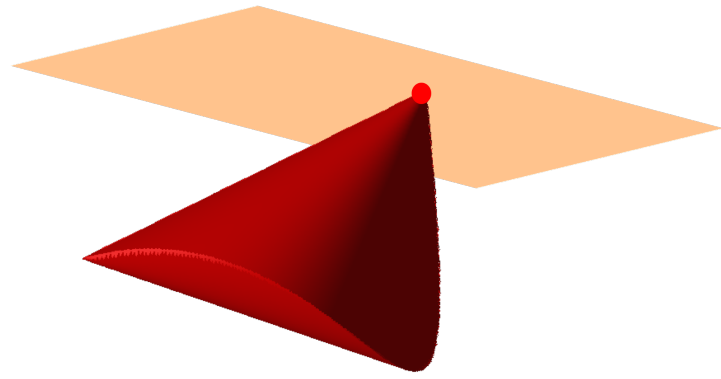


Optimization problems
(e.g. empirical risk minimization, maximum likelihood estimation)

(Convex) optimization is almost a tool

Solvability / tractability





"... the great watershed in optimization isn't between linearity and nonlinearity, but convexity and nonconvexity"

— R. Rockafellar '1993

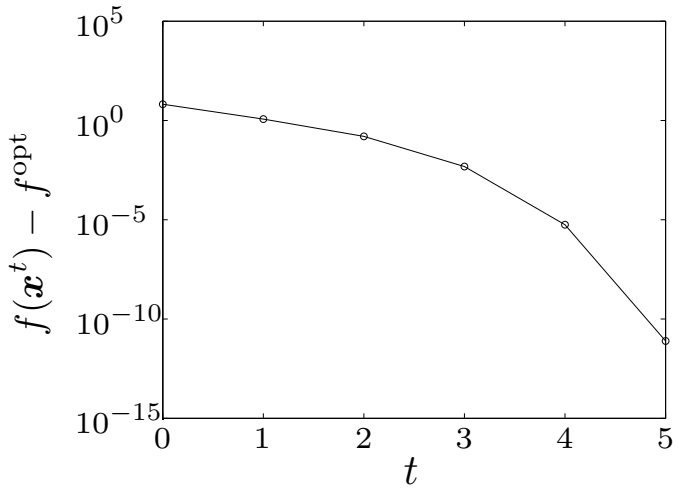
Polynomial-time solvability $\neq$ scalability

Even polynomial-time algorithms might be useless in large-scale applications

Example: Newton's method

$$\text{minimize}_{\boldsymbol{x}\in\mathbb{R}^n} \quad f(\boldsymbol{x})$$

$$\boldsymbol{x}^{t+1} = \boldsymbol{x}^t - (\nabla^2 f(\boldsymbol{x}^t))^{-1} \nabla f(\boldsymbol{x}^t)$$

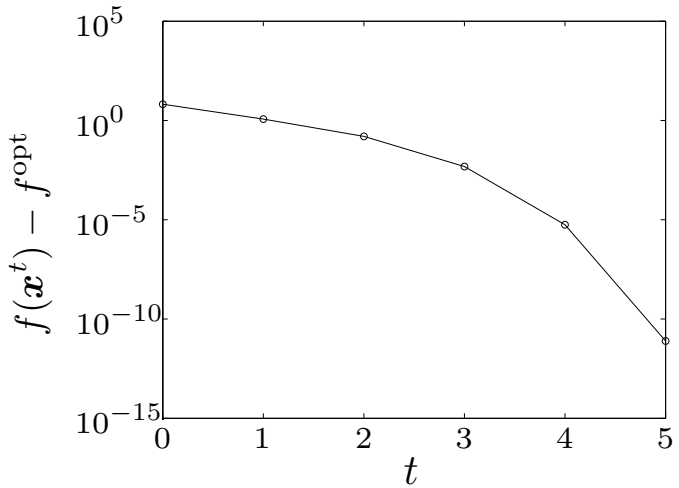


- attains ε accuracy within $O(\log\log\frac{1}{\varepsilon})$ iterations

Example: Newton's method

$$\text{minimize}_{\boldsymbol{x}\in\mathbb{R}^n} \quad f(\boldsymbol{x})$$

$$\boldsymbol{x}^{t+1} = \boldsymbol{x}^t - (\nabla^2 f(\boldsymbol{x}^t))^{-1} \nabla f(\boldsymbol{x}^t)$$

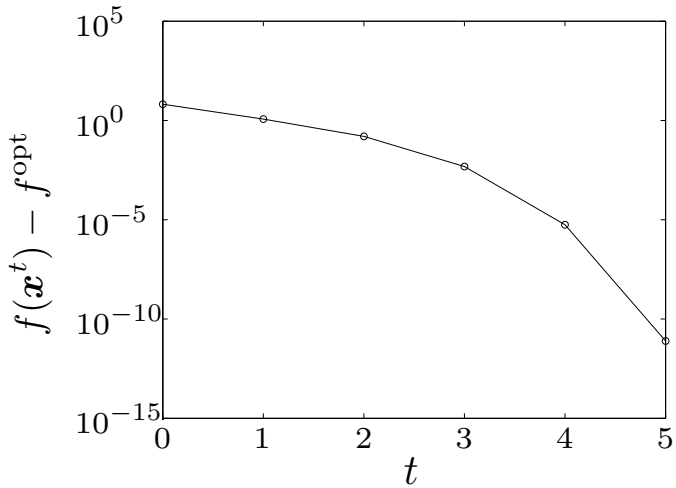


- attains ε accuracy within $O(\log\log\frac{1}{\varepsilon})$ iterations
- typically requires Hessian information $\nabla^2 f(\boldsymbol{x}) \in \mathbb{R}^{n\times n}$

Example: Newton's method

$$\text{minimize}_{\boldsymbol{x}\in\mathbb{R}^n} \quad f(\boldsymbol{x})$$

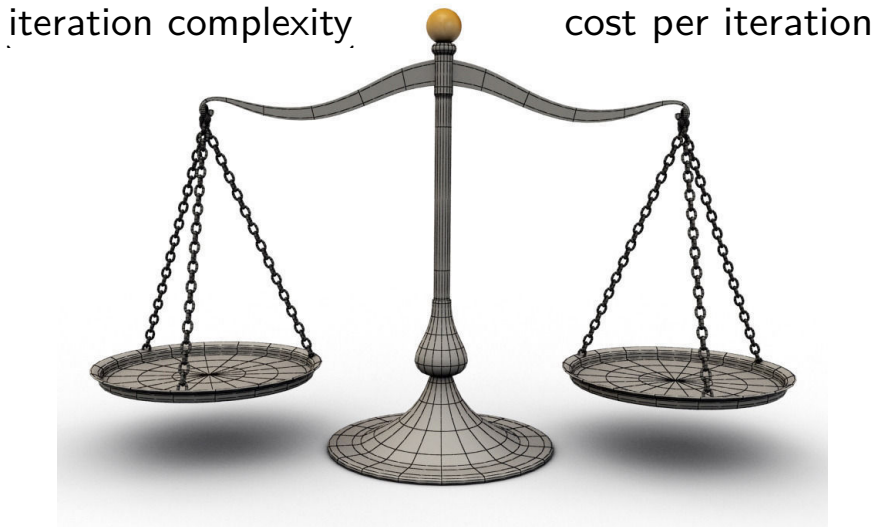
$$\boldsymbol{x}^{t+1} = \boldsymbol{x}^t - (\nabla^2 f(\boldsymbol{x}^t))^{-1}\nabla f(\boldsymbol{x}^t)$$



- attains ε accuracy within $O(\log\log\frac{1}{\varepsilon})$ iterations
- typically requires Hessian information $\nabla^2 f(\boldsymbol{x}) \in \mathbb{R}^{n\times n}$
- a single iteration may last forever; prohibitive storage requirement

Iteration complexity vs. per-iteration cost

$$\text{computational cost} = \underbrace{\text{iteration complexity}}_{\#\text{iterations needed}} \times \text{cost per iteration}$$



Large-scale problems call for methods with cheap iterations

Methods of choice

First-order methods: methods that exploit only information on function values and (sub)gradients (without using Hessian information)

- cheap iterations
- low memory requirement
- solve large-scale problems to acceptable accuracy

What this course will NOT cover

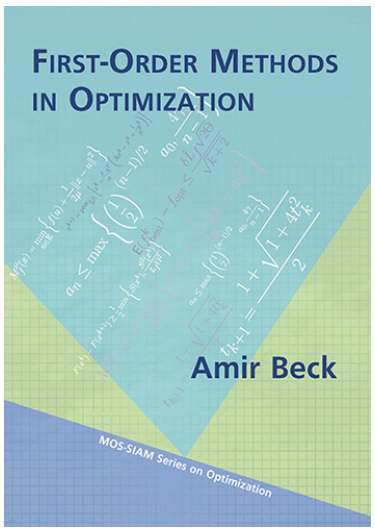
- second-order methods
  - check ORF 522 (by M. Wang) and COS 598D (by E. Hazan)
- convex analysis
  - check ORF 522 (by M. Wang) and ORF 523 (by A. Ahmadi)
- sum of squares programming
  - check ORF 523 (by A. Ahmadi)
- approximation algorithms for NP hard problems
  - check ORF 523 (by A. Ahmadi)
- computational hardness
  - check ORF 523 (by A. Ahmadi)
- online optimization
  - check COS 511 (by E. Hazan)

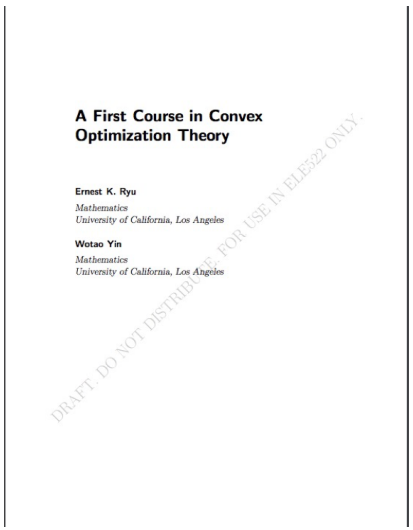
What this course will cover (hopefully)

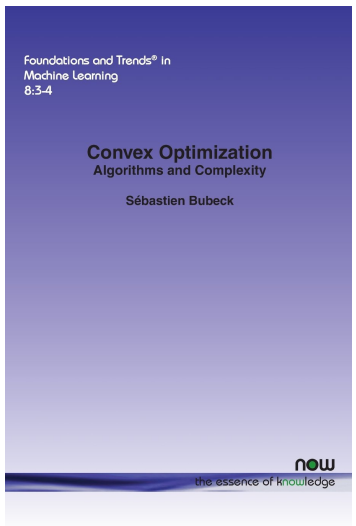
- gradient methods
- Frank-Wolfe and projected gradient methods
- subgradient methods
- proximal gradient methods
- accelerated proximal gradient methods
- mirror descent
- smoothing
- dual and primal dual methods
- stochastic gradient methods
- variance reduction
- quasi-Newton methods (BFGS)
- alternating direction method of multipliers (ADMM)
- distributed optimization

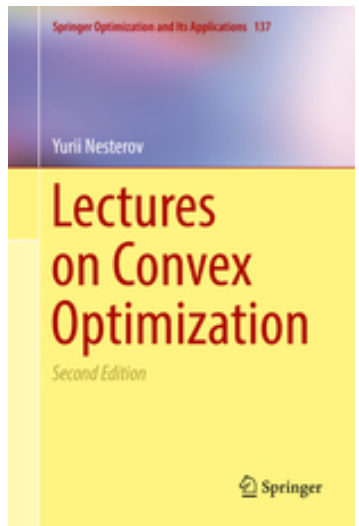
Textbooks

We recommend these books, but will not follow them closely ...









WARNING

- There will be quite a few THEOREMS and PROOFS ...
- May be somewhat disorganized

Prerequisites

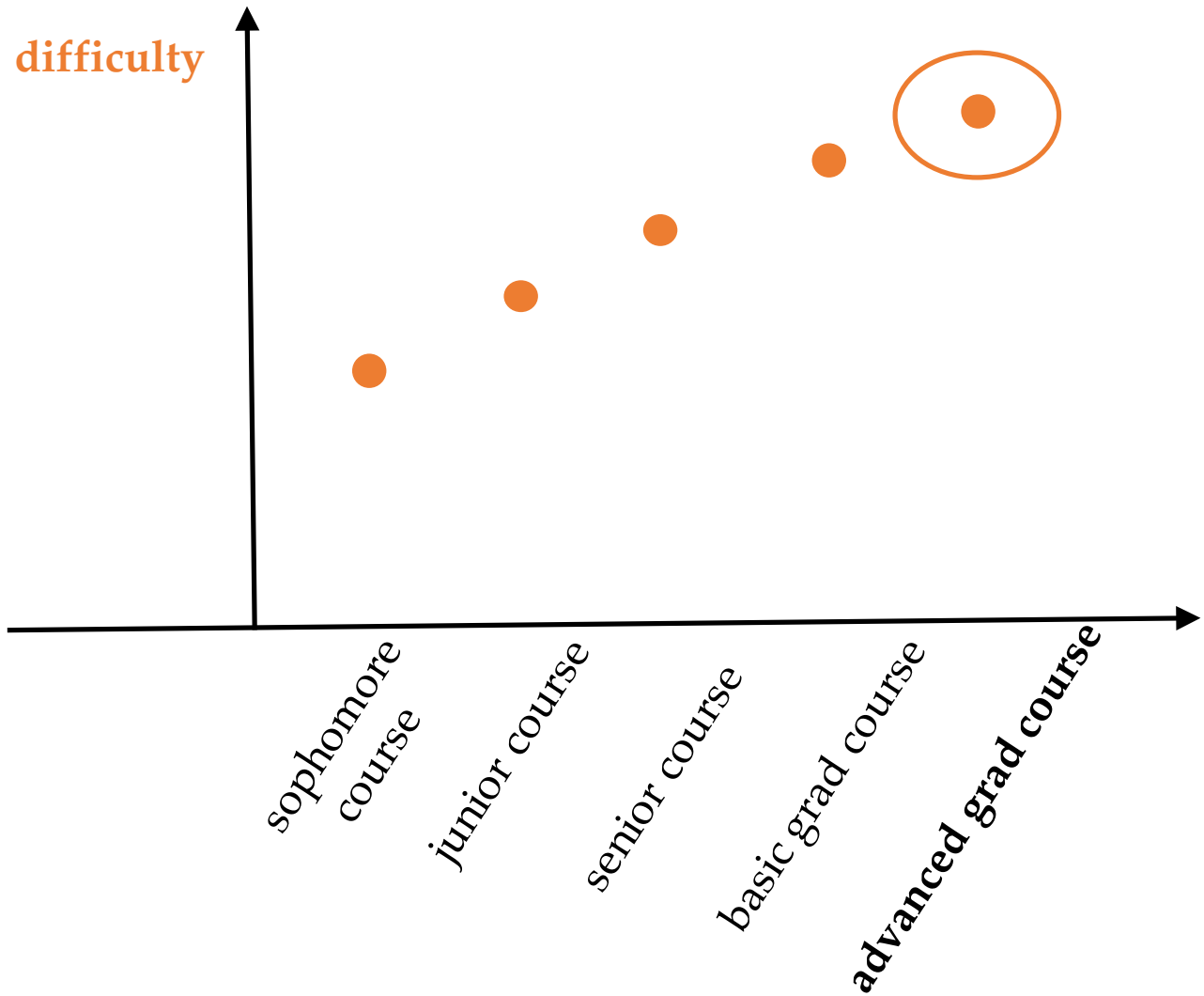
- basic linear algebra
- basic probability
- a programming language (e.g. Matlab, Python, ...)
- *knowledge in basic convex optimization*

Prerequisites

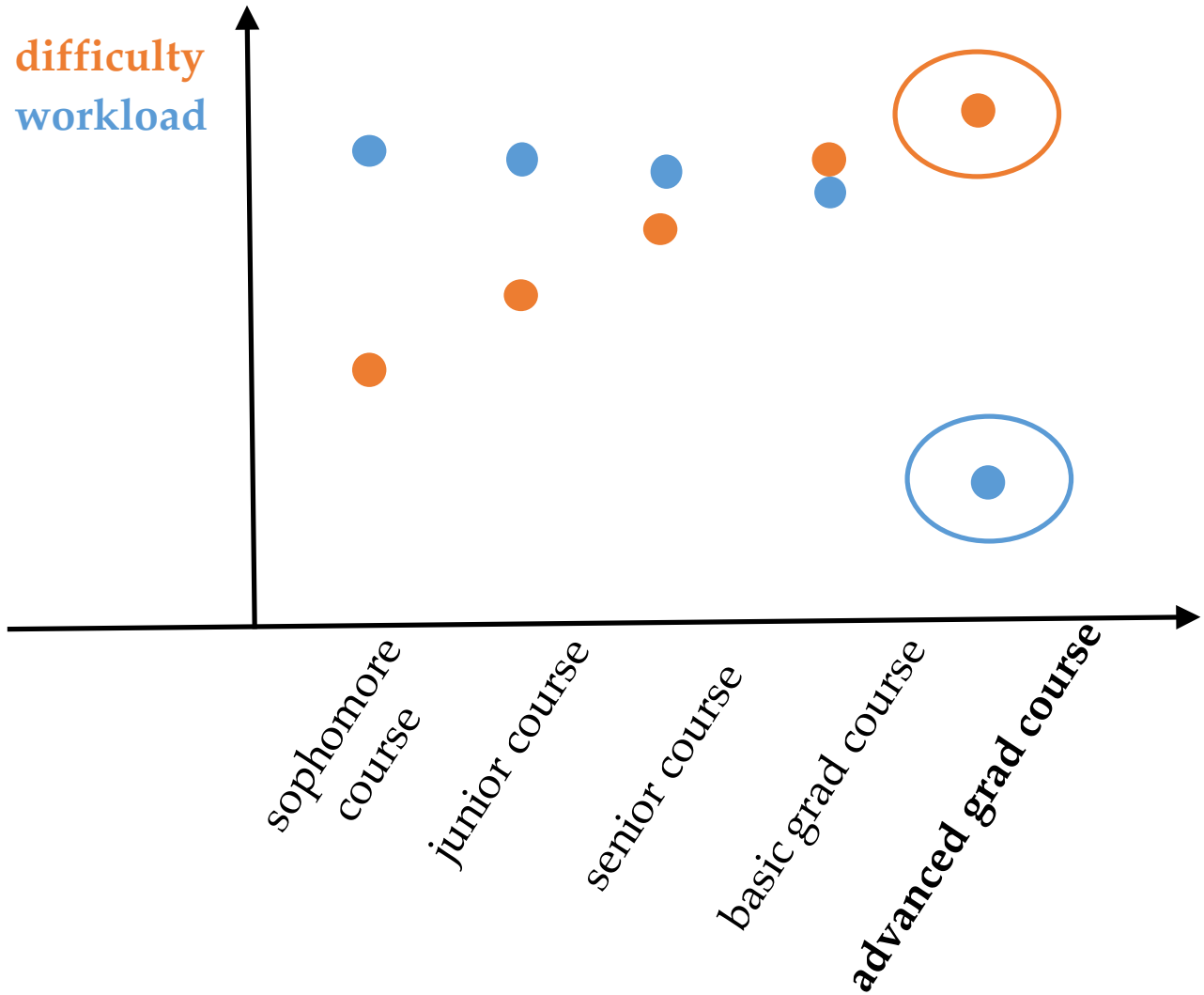
- basic linear algebra
- basic probability
- a programming language (e.g. Matlab, Python, ...)
- *knowledge in basic convex optimization*

Somewhat surprisingly, most proofs rely only on basic linear algebra and elementary recursive formula

Grading


difficulty
sophomore course
junior course
senior course
basic grad course
advanced grad course

Grading


difficulty
workload
sophomore course
junior course
senior course
basic grad course
advanced grad course

Grading

- Homeworks: ≤ 3 problem sets
  - use Piazza as the main mode of electronic communication; please post (and answer) questions there!
- Final exam (take-home)
- Term project
  - either individually or in groups of two

Grading

- Homeworks: ≤ 3 problem sets
  - use Piazza as the main mode of electronic communication; please post (and answer) questions there!
- Final exam (take-home)
- Term project
  - either individually or in groups of two

$$\text{grade} \;=\; \max\{(H+E+P)/3,\; (H+P)/2,\; P\}$$

where H: homework; P: project; E: takehome exam

Grading

- Homeworks: ≤ 3 problem sets
  - use Piazza as the main mode of electronic communication; please post (and answer) questions there!
- Final exam (take-home)
- Term project
  - either individually or in groups of two

$$\text{grade} \;=\; \max\{(H+E+P)/3,\; (H+P)/2,\; P\}$$

where H: homework; P: project; E: takehome exam

Term project

Two forms

- literature review
- original research
  - *You are strongly encouraged to combine it with your own research*

Term project

Two forms

- literature review
- original research
  - *You are strongly encouraged to combine it with your own research*

Three milestones

- Proposal (Oct. 23): up to 1 page
- Presentation (last week of class)
- Report (Jan. 13): up to 5 pages with unlimited appendix (submitted by email)