Final Review

Coverage–comprehensive, slight emphasis post-midterm

- pre-mid: Ross ch 1-5
- post-mid: Ross ch 6-8, hypothesis testing, mle em. DPV algs & NP
- everything in slides, homework, non-supplemental reading as noted on "Schedule & Reading" web page

Mechanics

- closed book, aside from one page of notes (8.5 x 11, both sides, handwritten)
- I'm more interested in setup and method than in numerical answers, so concentrate on giving a clear approach, perhaps including a terse english outline of your reasoning.
- Corollary: calculators are probably irrelevant, but bring one to the exam if you want, just in case.

Format–similar to midterm:

- T/F, multiple choice, problem-solving, explain, …
- Story problems

Hell's library

what to expect on the final in more detail ...

see midterm review slides

- tail bounds
  - Markov
  - Chebyshev
  - Chernoff
- limit theorems
  - weak/strong laws of large numbers
  - central limit theorem

likelihood, parameter estimation, MLE

likelihood

- "likelihood" of observed data given a model
- usually just a product of probabilities (independence assumption)
- a function of (unknown?) parameters of the model

parameter estimation

- if you know/assume the *form* of the model (e.g. normal, poisson,...), can you estimate the *parameters* based on observed data
- many ways

maximum likelihood estimators

- one way to do it–choose values of the parameters that maximize likelihood of observed data
- method (usually) – solve
  - "derivative (wrt parameter/s) of (log) likelihood = 0"

expectation maximization (EM)

EM

- iterative algorithm trying to find MLE in situations that are analytically intractable
- usual framework: there are 0/1 *hidden variables* (e.g. from which component was this datum sampled) & problem much easier if they were known
- E-step: given rough parameter estimates, find expected values of hidden variables
- M-step: given expected values of hidden variables, find (updated) parameter estimates to maximize likelihood
- Algorithm: iterate above alternately until convergence

hypothesis testing

I have data, and 2 (or more) hypotheses about the process generating it. Which hypothesis is (more likely to be) correct?

Again, a very rich literature on this.

One of the many approaches: the "Likelihood Ratio Test"

calculate: $\frac{\text{likelihood of data under alternate hypothesis}}{\text{likelihood of data under null hypothesis}}$

ratio > 1 favors alternate, < 1 favors null, etc.

(type 1, type 2 error , α, β, etc.)

probability & statistics, broadly

noise, uncertainty & variability are pervasive

learning to model it, derive knowledge and compute despite it are critical

E.g., knowing the mean is valuable, but same mean, different variances can behave very differently in practice

complexity summary

Big-O – good

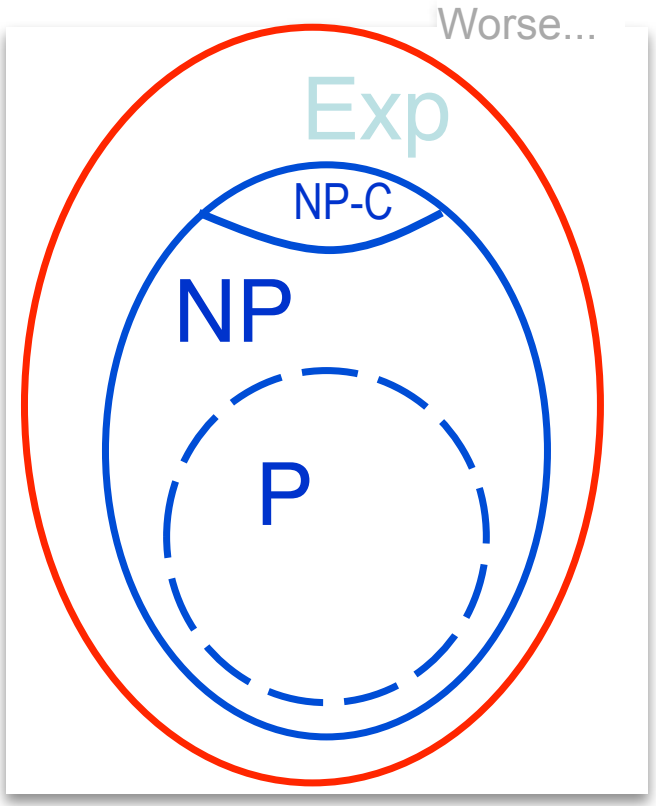
P – good

Exp – bad

Exp, but hints help? NP

NP-hard, NP-complete – bad (I bet)

To show NP-complete – reductions



NP-complete = hopeless? – no, but you need to lower your expectations: heuristics, approximations and/or small instances.

algorithms and complexity

Decades of work on many computational problems has lead to remarkable improvements in speed/memory/solution quality/etc.

Algorithmic progress dwarfs Moore's law in many cases

Some broadly applicable generic approaches like "divide and conquer" and "dynamic programming" have emerged

"Polynomial time" is a good first approximation to "feasibly computable"

Unfortunately, for many problems no polynomial time algorithm is known, we are not able to routinely solve large, arbitrary instances, and progress on doing so has been slow and erratic.

Most such problems are NP-hard; many are NP-complete (a subset)

Key characteristics:

- searching for a "solution" among exponentially many possibilities
- each solution can be described by polynomially many bits
- a potential solution can be verified in polynomial time

Key technique: reduction

algorithms and complexity – questions to expect?

P:

Big picture

What's P? Why does it matter? In principle, what do you have to do to show that a problem is in P? Broadly speaking, what is "divide & conquer" about? How much can it help? Ditto for dynamic programming?

Details

Know the algorithms: closest points, integer multiply, string alignment. Be able to "hand simulate" one on some data.

NP:

Big picture

What's NP? Why does it matter? In principle, what do you have to do to show that a problem is in NP? Is NP-hard? Practical consequences?

Details

Show some new problem is in NP. Mechanics of vertex-cover and subset-sum reductions (e.g., like on the homework, how do you go between assignments & covers; how many edges does the graph have; ...) I will NOT ask you to create reductions you haven't seen.

want more?

Stat 390/1	probability & statistics
CSE 421	algorithms
CSE 431	computability and complexity
CSE 427/8	computational biology
CSE 440/1	human/computer interaction
CSE 446	machine learning
CSE 473	artificial intelligence

and others!

Thanks and Good Luck!