1. Logistics

2. Overview of the class and
   a few definitions and examples

3. Issues with real computers:
   floating point
   memory hierarchy.

Material in Gray was not covered

S&DS 631, <u>Dan</u> Spielman (or Prof. Dan)
is a graduate course.
Compared to undergrad courses is
- cruder
- less well organized
- assumes more maturity
  (students can survive my mistakes)

Goes double because this is the first offering
Plans will change.

Pre reqs: Multivariate Calc, Linear Algebra
Probability (probably)
Exposure to programming

Materials: will draw on free materials,
and produce notes like this.

Examples in Julia. But no programming assignments.

TF: Tracey (Xinyi) Zhong

Work: Semi-regular homework (math problems)
Midterm (in class on Tuesday, March 2)
Final (May 6 at 2pm - last time slot)

Why I created this course:

A lot of work in S&DS involves
optimization and numerical algorithms.

Many advanced classes assume a basic understanding.

So, want to create an elementary, systematic
introduction.

If we can't compute it, we can't do it.

Need to know about what we can and can not
solve efficiently.

NP - completeness.

How to measure running times of algorithms.

Other S&DS courses will eventually assume this one.

Might turn it in to "continuous CS 365/366"

Programming? Wanted to but couldn't fit it.

Motivating optimization problems

Least squares: $\min_x \|Ax-b\|_2 \quad \|v\|_2 = \left(\sum_i v(i)^2\right)^{1/2}$

Should say "arg min" because usually want x

Ridge regression: $\min_x \|Ax-b\|_2^2 + \lambda\|x\|_2^2 \quad , \quad \lambda > 0$

also least squares, but faster.

Linear Programming $\min_x \|Ax-b\|_\infty \quad \|v\|_\infty = \max_i |v(i)|$

$\min_x \|Ax-b\|_1 \quad \|v\|_1 = \sum_i |v(i)|$

l_1-penalized $\min_x \|Ax-b\|_2^2 + \lambda\|x\|_1$ LASSO
Conic programming

l_0/sparse $\min_x \|Ax-b\|_2^2 + \lambda\|x\|_0 \quad \|x\|_0 = |\{i : v(i) \neq 0\}|$

Is NP hard. So approximate by l_1

Regularized Empirical risk minimization (logistic / Neural Nets)

$$\min_x R(x) + \sum_i f_i(x)$$

Or, constrained: $\min_x \sum_i f_i(x) \quad \text{s.t.} \quad g(x) \leq 0$

Convex problems - can always solve.

Will cover linear programming and its geometry.

Will explain simplex method

and first order optimization methods.

Not the fastest, but an introduction

SVD and Semidefinite programming

Nonconvex - often NP-Hard

2 weeks on hard problems

Sometimes solvable - when inputs satisfy

nice properties we will study.

Examples: l_0 regularized least squares

Matrix Completion by Nuclear Norm minimization.

How you know when you've solved a problem:

Duality, Lagrange Multipliers, KKT conditions.

Measuring run time: count # of operations

+, ×, /, −, <, >, branch, goto, memory reference

For many numerical algs just count arithmetic ops, when their cost dominates.

But, memory access takes much longer.

Modern computers employ multiple caches:
fast small memories with a speed / size tradeoff

Result is that recently accessed memory is faster.

First measure run-time as a function of the size of the input, n.

Size = # of parameters, numbers, or bits.

Worry most about scaling.

That allows you to extrapolate from small trials.

Express using "O" notation.

If time is $T(n)$, I write $T(n) \le O(f(n))$

if $\exists\, c$ s.t. $\forall\, n \ge 1$ $T(n) \le c f(n)$.

Warning: the standard is to write "$= O(f(n))$".

I am trying to fix that.

But, other parameters are important, like accuracy ε.

Say want and $\tilde{x}$ s.t. $\|A\tilde{x} - b\|_2 \le \varepsilon$.

Might write $T(n, \varepsilon) \le O(f(n, \varepsilon))$

meaning $\exists\, c, \varepsilon_0 > 0, n_0 > 0$ s.t.

$\forall\, n > n_0, \varepsilon > \varepsilon_0$ $T(n, \varepsilon) \le c \cdot f(n, \varepsilon)$

Here n would be length of b + product of dimensions of A.

Will often measure run time in terms of parameters of the input, like the condition number of the problem.

Might examine how random noise impacts these parameters.

Usually want f to be a polynomial in n and $1/\varepsilon$.

NP = "nondeterministic polynomial".

probably not polynomial time.

Accuracy. Say our problem is to compute $f(x)$, and our code outputs $\tilde{f}(x)$.

(absolute) <u>Forward error</u> is $\|\tilde{f}(x) - f(x)\|$.

For now, I leave the norm unspecified.
if $f(x) \in \mathbb{R}$ then $|f(x)|$ is a good choice

(relative) Forward error is $\dfrac{\|\tilde{f}(x) - f(x)\|}{\|f(x)\|}$

Error could result from time/accuracy tradeoff
Or floating point:
rep numbers as $\pm 2^e \cdot b$ $\quad |e| \leq 1023$
$b \leq 2^{52}$

So, $1 - 10^{-17} = 1$
Smallest u s.t. $1 - u \neq 1$ is machine precision

<u>Backward error</u> is $\|x - \tilde{x}\|$ s.t. $f(\tilde{x}) = \tilde{f}(x)$
the closest problem for which $f(\tilde{x})$ is the right answer.

Relative Backwards error is $\dfrac{\|x - \tilde{x}\|}{\|x\|}$

Example Fix matrix A, and let $f_A(b) = \{y : Ay = b\}$

Assume A invertible, so $f_A(b) = A^{-1}b$

If our alg returns $\tilde{y}$,

forward error is $\|y - \tilde{y}\|$

backward error is $\|b - A^{-1}\tilde{y}\|$ ← can measure this one!

The condition numbers

Measure how output changes when make a small change in the input.

View problem as a function $f(x)$

let δx be small change, and

$$\delta f = f(x + \delta x) - f(x)$$

Absolute condition number at x is

$$\hat{\kappa} = \lim_{\varepsilon \to 0} \sup_{\|\delta x\| \le \varepsilon} \frac{\|\delta f\|}{\|\delta x\|}$$

$$= \frac{\text{forwards error}}{\text{backwards error}}$$

In our example, $\hat{\kappa} = \dfrac{\|y - \tilde{y}\|}{\|A^{-1}y - A^{-1}\tilde{y}\|} = \dfrac{\|\delta y\|}{\|A^{-1}\delta y\|}$

Setting $\delta b = A^{-1} \cdot \delta y$, this is $\dfrac{\|A \cdot \delta b\|}{\|\delta b\|}$

The operator norm of a matrix A is

$$\|A\|_2 \stackrel{\text{def}}{=} \max_{x \neq 0} \frac{\|Ax\|_2}{\|x\|_2}$$ how much A can blow up a vector.

So, $\hat{K} = \|A\|_2$

This norm is different from the Frobenius norm of A = $\|A\|_F = \left(\sum_{i,j} A(i,j)^2 \right)^{1/2}$

which treats the matrix as a vector.

If $f : \mathbb{R}^n \to \mathbb{R}$ is differentiable, then

$\hat{K} = \|\nabla f\|$ where $\nabla f = \left(\frac{\partial f}{\partial x_1}, \dots, \frac{\partial f}{\partial x_n} \right)$

More on this later.

We will usually work with the Relative Condition Number, K

$$= \lim_{\varepsilon \to 0} \sup_{\|\delta x\| \leq \varepsilon} \frac{\|\delta f\| / \|f\|}{\|\delta x\| / \|x\|}$$

The relative condition number of multiplication by A at x is thus $\kappa = \hat{\kappa} \cdot \frac{\|x\|}{\|f\|}$

$$= \frac{\|A\,\delta b\|}{\|\delta b\|} \frac{\|b\|}{\|Ab\|} = \frac{\|A\,\delta b\|}{\|\delta b\|} \cdot \frac{\|A^{-1} y\|}{\|y\|}$$

$$\leq \|A\| \cdot \|A^{-1}\|$$

This is tight if we choose

δb s.t. $\|A\,\delta b\| = \|A\| \cdot \|\delta b\|$ and

y s.t. $\|A^{-1} y\| = \|A^{-1}\| \cdot \|y\|$.