Matrix Completion comes from the Netflix Prize '07-'09

Given ratings of movies by people (person, movie, score)

Try to predict ratings by those people on other movies.

Data was very sparse

$\approx$ 500k people, 18k movies, on average 200 ratings/person

And, very irregular: many people rate few movies.

some movies get few ratings

Idea: Consider matrix

$$\begin{array}{c|c} \text{movies} & \begin{pmatrix} o & & o & & o \\ & 6 & \text{ratings} & & o \\ & o & o & & o \end{pmatrix} \\ \hline & \text{people} \end{array}$$

Knowing only a few entries in the matrix, can we guess the rest.

Need to assume matrix has some structure, like approximately low rank (low-rank + noise).

Matrix Completion problem: given $m \times n$ matrix M, $\Omega \subseteq \{1..m\} \times \{1,..,n\}$ and an integer r

$$\min_{X} \sum_{(i,j)\in\Omega} (M(i,j) - X(i,j))^2 \quad \text{s.t. } \operatorname{rank}(X) \le r.$$

or, given $\varepsilon > 0$,

$$\min_{X} \operatorname{rank}(X) \quad \text{s.t. } \|M - X\|_{\Omega}^2 \le \varepsilon$$

Both forms are NP-hard, even with $\varepsilon = 0$.

So, replace $\mathrm{rank}(X)$ with something convex.

If $\sigma_1, \dots, \sigma_m$ are singular values of X,

$$\mathrm{rank}(X) = \|(\sigma_1, \dots, \sigma_m)\|_0$$

So, by analogy with ℓ_1, use the nuclear norm of X

$$\|X\|_* = \|(\sigma_1, \dots, \sigma_m)\|_1 = \sum_i \sigma_i \qquad \text{(Fazel)}$$

Contrast with

$$\|X\|_F = \|(\sigma_1, \dots, \sigma_m)\|_2 = \left(\sum \sigma_i^2\right)^{1/2}$$

$$\|X\|_2 = \max_i \sigma_i$$

vec	matrix
$\|v\|_0$	$\mathrm{rank}(M)$
$\|v\|_\infty$	$\|M\|_2$
$\|v\|_2$	$\|M\|_F$
$\|v\|_1$	$\|M\|_*$

Exact recovery:

$$\min_X \|X\|_* \quad \text{s.t.} \quad \|M - X\|_\Omega = 0$$

Approximate recovery:

$$\min_X \|X\|_* \quad \text{s.t.} \quad \|M - X\|_\Omega \le \varepsilon$$

Work well when Ω is random (big assumption)

X is nice: no big entries,

singular vectors incoherent.

If M is low rank, can actually find M.

The nuclear norm (also called trace norm)
because trace when symmetric.

Is the dual of the operator norm, $\|X\|_2$.

Thm $\underline{\|X\|_*} \overset{?}{=} \underline{\max_{\|Y\|_2 = 1}} \underline{\mathrm{Tr}(X^T Y)} = \sum_{(i,j)} X(i,j)\,Y(i,j)$

proof $X = USV^T$

$Y = UV^T \leftarrow$ all $\sigma_i = 1$, $\|Y\|_2 = 1$

$\mathrm{Tr}(X^T Y) = \mathrm{Tr}(VS\underbrace{U^T U}V^T) = \mathrm{Tr}(VSV^T)$

$= \mathrm{Tr}(SV^T V) = \mathrm{Tr}(S) = \|X\|_*$

$Y = \hat{U}\hat{V}^T$ $\quad \mathrm{Tr}(X^T Y) = \mathrm{Tr}(VSU^T \hat{U}\hat{V}^T)$

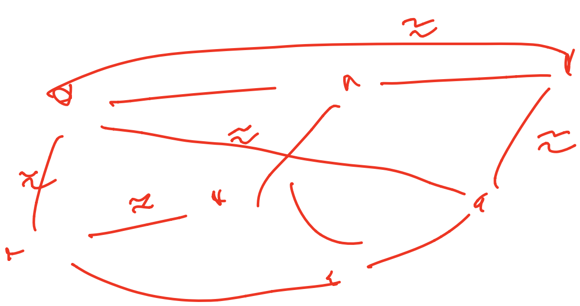
$= \mathrm{Tr}(S\underbrace{U^T \hat{U}\hat{V}^T V}_{Q\,=\,\text{orthogonal}})$

$= \mathrm{Tr}(SQ)$

$= \sum_i \sigma_i Q(i,i) \qquad Q(i,i) \le 1, \forall i$

$\le \sum_i \sigma_i = \mathrm{Tr}(S) = \|X\|_*$

Now we know $\|X\|_*$ is convex



approx compute distances . . .

$$\|X\|_* = \min_{W_1, W_2} \frac{1}{2}\left(Tr(W_1) + Tr(W_2)\right) \leftarrow \min \|H\|_*$$

$$\text{s.t.} \begin{pmatrix} W_1 & X^T \\ X & W_2 \end{pmatrix} \succeq 0 \quad \text{PSD}$$

So, can minimize nuclear norms using
semidefinite programming.

Original Story: approx low-rank
by nuclear norm.

Alternate Story: Use complexity to guide
problem and algorithm selection.

Exactly solve tractable problem

Approximately solve hard problem

or just try to solve hard problem. (heuristic).

Alternating minimization:

$$M = \underbrace{L R^T}_{\text{rank } r}$$

0. pick $L \in \mathbb{R}^{m \times r}$ <u>at random</u>

1. Until $\|M - LR^T\|_\Omega$ is small

$$\underline{R} = \arg\min_{R \in \mathbb{R}^{n \times r}} \underline{\|M - LR^T\|_\Omega}$$

$$\underline{L} = \arg\min_{L \in \mathbb{R}^{m \times r}} \|M - LR^T\|_\Omega$$

How hard are NP-hard problems?

For any, time less than $\boxed{2^{n^{\varepsilon}}}$ would be shocking $\forall \varepsilon > 0$

For most, time $\leq 2^{n^{1-\varepsilon}}$, any ε, would be surprising

Circuit SAT: time $\leq \boxed{2^{(1-\varepsilon)n}}$ unlikely. $n = \#$ inputs.

Strong Exponential time hypothesis (SETH)

$\Rightarrow$ lower bounds for many problems in P.

Are other notions of hard, from

very hard – uncomputable, to pretty hard - Unique Games Conj.

When we can solve NP-hard problems,

they are probably special:

random

structured

structure + noise (smoothed analysis) SPE Condition

well conditioned

other nice properties

Probably Certifiably Correct – Bandeira

Random problems that can usually solve

and certify optimality.

Ex. relax, and certify by computing solution to dual of relaxation hard

$\max_{\{\pm 1\}^n}$ $\leq$ $\max_{\|v_1\|=1,\ldots,\|v_n\|=1}$ $\leq$ dual of that

min ....

1 2

can provide certificate

In nice cases, can quickly solve dual from solution to original – complementary slackness.

Other notions of complexity:

Information complexity of Black-Box algorithms.

See: Bubeck's monograph, Convex Optimization: Algorithms and Complexity, or Lectures on Modern Convex Optimization by Ben-Tal and Nemirovski

Consider an optimization algorithm that minimizes f only using

- evaluations of f
- a few properties of f, like convexity.

Think of accessing f through a black-box or an oracle.

Can prove lower bounds on the # of calls necessary

Can even allow evaluations of gradients of f

(or sub-gradients: $f(y) \geq f(x) + g_x^T(y-x)$, $\forall y$)

For example, consider minimizing convex functions over $B(0,1) = \{x : \|x\|_2 \le 1\} \subseteq \mathbb{R}^n$

Can prove that for every (deterministic) algorithm, A, that makes $k < n$ oracle calls, $\exists$ a convex f s.t.

$$\|x_k - x_*\| \geq \text{const}/\sqrt{k}$$

where x_* = minimum of f and x_k is output of algorithm.

Are many results like this for many function classes.

To prove faster convergence of an algorithm, must use more about the function.

Black-Box solvers usually arise in non-convex optimization.

These are heuristics people use to try to minimize or maximize functions.

I use them to find counter-examples.

For example, here's a conjecture:

<u>Conj</u> For all orthogonal matrices Q

$\exists\, x \in \{\pm 1\}^n$ such that $\|Qx\|_\infty \le 2$.

It's false. I know because I found a Q.

(might be true with 2.1)

Typical approaches: BFGS L-BFGS

1. Gradient descent from many random starts.

2. Nelder-Mead. (the other simplex method)

   keep $n+1$ vectors in $\mathbb{R}^n$, $x_1, \ldots, x_{n+1}$

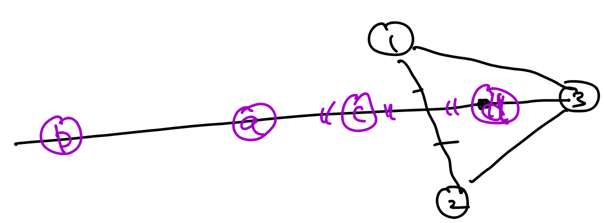
   adjust them until all approach the solution.

   Order so that $f(x_1) \le f(x_2) \le \cdots \le f(x_{n+1})$

   x_{n+1} is worst, so try to improve it by

   moving on line through centroid $= \frac{1}{n+1} \sum_i x_i$

<u>Ex.</u>

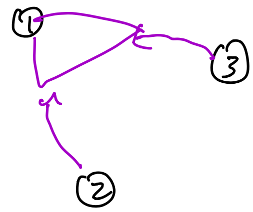


a better than 3? swap

/ a better than 1? Try b

→ try c, d if better than 3, swap

Contraction



3. Line searches in random directions.

4. Simulated annealing: g decreasing with t.

$\hat{x} = x + \tau$, τ random (usually discrete) small

if $f(\hat{x}) \le f(x) + g(t)$

$x = \hat{x}$

5. Differential Evolution ☆

Generate a population $x_1, \dots, x_{cn}$ $c > 0$

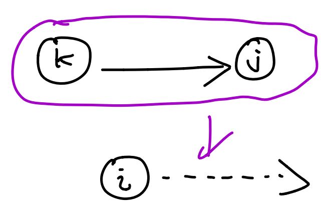
Pick i, j, k at random

make sure $f(x_j) \le f(x_k)$ (swap otherwise)

$\hat{x} = x_i + (x_j - x_k)$

if $f(\hat{x}) \le f(x_i)$

$x_i = \hat{x}$



Integer Linear Programming

$$\min c^T x \text{ s.t. } Ax \le b, \quad x \in \mathbb{Z}^n$$
$$\text{or } x \in \{0,1\}^n$$

Is NP-hard, but is good code
that often works in practice.
(CPLEX, Gurobi, Knitro, Mosek)

Might take a long time,
but can get certificates of optimality.

SAT solvers.

Solve problems like circuit satisfiability.
(usually SAT, which is slightly simpler)
Are competitions.

Conclusions of unsatisfiability are
key technology in program verification
and secure kernels.

SOS (Sums of Squares) Hot topic

Any polynomial inequality like $\boxed{f(\bar{x}) \geq 0}$

has a proof $\underline{h(x)^2} f(x) = \underline{\sum_i g_i(x)^2}$

where h and g_i are polynomials.

Ex. $x^2 + y^2 - 2xy \geq 0$

$$x^2 + y^2 - 2xy = (x-y)^2$$

Find them by Semidefinite Programming.

Modern Machine Learning

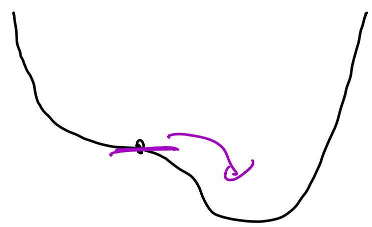
Fit models to data by non-convex programming

Given vectors $x_1, \dots, x_n$ and labels $y_1, \dots, y_n$
try to choose parameters Θ (e.g. weights in neural net)
to minimize $F(\Theta) = \sum_i (f_\Theta(x_i) - y_i)^2$

find Θ

$\sum_i \ell(f_\Theta(x_i), y_i)$

Often use gradient-based methods to find local minima.
Need to escape saddle points!



Many theorems for special problems
Improvements in Black-Box algorithms, assuming more.

But, are often many local minima,
different algorithms produce different Θ.
Θ MATTERS MORE THAN $F(\Theta)$