L17 -- Orthogonal Matching Pursuit
[Jeff Phillips - Utah - Data Mining]

What is compressed sensing?
--> regress to "sparse" explanation
try to encode data with small number of variables

----

single pixel camera:
10 Gigapixels of images, but jpg still 2MB? why is that?
compression.
many cameras compress image even before storing it.
What if we can get same resolution jpq with only 2megapixel sensor?
or 2 mega - repeated measurements at "single pixel"
Incredible resolution with 10 Gigapixels??? (ignoring lens quality...)

hash (sketch) of data:
Hubble telescope: incredibly clarity, but
- communication with Earth expensive
- sensing (battery) expensive
Sense and encode pictures, let Earth decode

Data often sparse+noise:
Very few actual events of interest, but readings not exactly 0 since noise
Decode sparse measurements filtering out noise

Formal Problem Set-up:

Data is S = d-dimensional vector with m << d non-zero elements
"m-sparse"

example:
S = [0 1 0 0|0 1 0 0|0 0 0 0|0 0 0 1#0 0 0 1|1 0 1 0|0 0 1 0|0 1 0 0]
d = 32
m = 8 << 32

(if noisy, maybe 1 actually large value, and 0 actually very small value <= .05)

recover in N = O(m log (d/m) (random) measurements:
x_i = d random vector (e.g. Gaussians or {-1,0,+1})

example:
x_i = [-1 0 1 0 0|1 1 -1 1|0 -1 0 0|1 -1 -1 1#0 1 0 1|-1 -1 -1 0|1 0 0 -1|0 1

0 0]
y_1 =   0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 -1 0 -1 0 0 0 0 0 0 1
0 0 = 2

y_i = <S, x_i>
each element of S "hit" by 0-mean, random variable

- only really lose "log factor"
  + since sparse storage requires log(d) to store location of each 1
  + each measurement requires about log(d) storage to get correct precision.
  + if S is not 0/1, but 0/x, then don't even really lose log-factor, just
constant

How do we "recover" S from Y?
** we know X (used to measure), an N x d matrix**
random, but seed known

Simplest form of data recovery : "Orthogonal Matching Pursuit" (OMP)

* Find measurement column j (not row used to measure)
  x_j = argmax_{x_j in X} |<Y, x_j>|
  represents single index of S that explains most about Y
* Find weight
  gamma = argmin_{gamma in |R} || Y - x_j gamma ||
  our "guess" of s_j is gamma

now what? Don't want to find part already explained by gamma = s_j

Let "residual" r_0 = Y
r_1 = Y - x_{j_1} gamma_1
...
r_t = r_{t-1} - x_{j_t} gamma_{j_t}

Use rounds on t
1: * Find measurement index
j_t = argmax_{j in [N]} |<r_{t-1}, x_j>|
2: * Find weight
gamma_t = argmin_{gamma in |R} || r_{t-1} - x_{j_t} gamma ||
3: * Set new residual
r_t = r_{t-1} - x_{j_t} gamma_t
if (||r_t|| = 0) stop

NOTES:
- can add regularization term into loss-function in step 2 (implicit in
step 1)

i.e. || Y - x_{j_t} gamma || + |gamma|
- can re-solve optimal LS in step 2
  2: gamma_[1..t] = argmin_{gamma in |R^t} || T - x_{1..j_t} gamma_{1..t} ||
  3: r_t = Y - x_{1..j_t} gamma_{1..t}
- can speed up LS @ 2 by maintaining partial decomposition of Y
  (QR decomposition)
- Converges: always ||r_t|| < ||r_{t-1}||
  coordinate descent (Frank-Wolfe algorithm, shows 1/eps steps apx within eps)
- new x_{j_t} always "linearly independent" of X_{1..j_{t-1}}
  adding new type of "explanation" towards Y

N = O(m log d) sufficient: like "Coupon Collector"
key to analysis: <x_i,x_i'> is small for all i,i' in [N]

Similarity to "random projection" for dimensionality reduction.
Says (roughly): if original data is sparse (most points only along a few axes)
- can recover data exactly after projecting to linear subspace

----

SVD algorithm works like this!
we consider all "measurement indices" directions in S^d
each subsequent one is orthogonal on data
decomposes since ||P||^2 = sum_{i=1}^d ||p_i||^2
--> works exactly for any k

Explanatory Variables
if OMP is stopped early, then X_{j_1...j_t} are few explanatory variable.
avoids over fitting if not all error is recovered.

stop at ||r_t||_2 < constant, get all large indices (loses noise)
stop at ||r_t||_infty < constant, only get large indices (none with noise)