UNIVERSITY OF MINNESOTA TWIN CITIES

High performance numerical linear algebra: trends and new challenges.

Yousef Saad

Department of Computer Science and Engineering

University of Minnesota

HPC days in Lyon

April 8, 2016

First: A personal tribute

➤ Grenoble [not far off from here] was the place to be in scientific computing in the 1960's and 1970's

Jean Kuntzmann

Noël Gastinel

➤ Jean Kuntzmann [1912-1992] and Noel Gastinel [1925-1984] played a huge role in Grenoble's pre-eminence in Numerical Analysis and Computer Science in France and Europe

Introduction: Numerical Linear Algebra

Numerical linear algebra has always been a "universal" tool in science and engineering. Its focus has changed over the years to tackle "new challenges"

1940s–1950s: Major issue: the flutter problem in aerospace engineering. Focus: eigenvalue problem.

➤ Triggered discoveries of the LR and QR algorithms, and the package Eispack followed a little later

1960s: Problems related to the power grid promoted what we know today as general sparse matrix techniques.

1970s: Finite Element methods, Computational Fluid Dynamics, reinforced need for general sparse techniques

Late 1980s – 1990s: Focus on parallel matrix computations.

Late 1990s: Big spur of interest in "financial computing" (After which the stock market collapsed ...)

- Computational Mechanics (e.g., Fluid Dynamics, structures) has been a driving force in past few decades
- But new forces are reshaping numerical linear algebra

Recent/Current: Google page rank, data mining, problems related to internet technology, knowledge discovery, bio-informatics, nano-technology, ...

- Major factor: Synergy between disciplines.

Example: Discoveries in materials (e.g. semi conductors replacing vacuum tubes in 1950s) lead to faster computers, which in turn lead to better physical simulations..

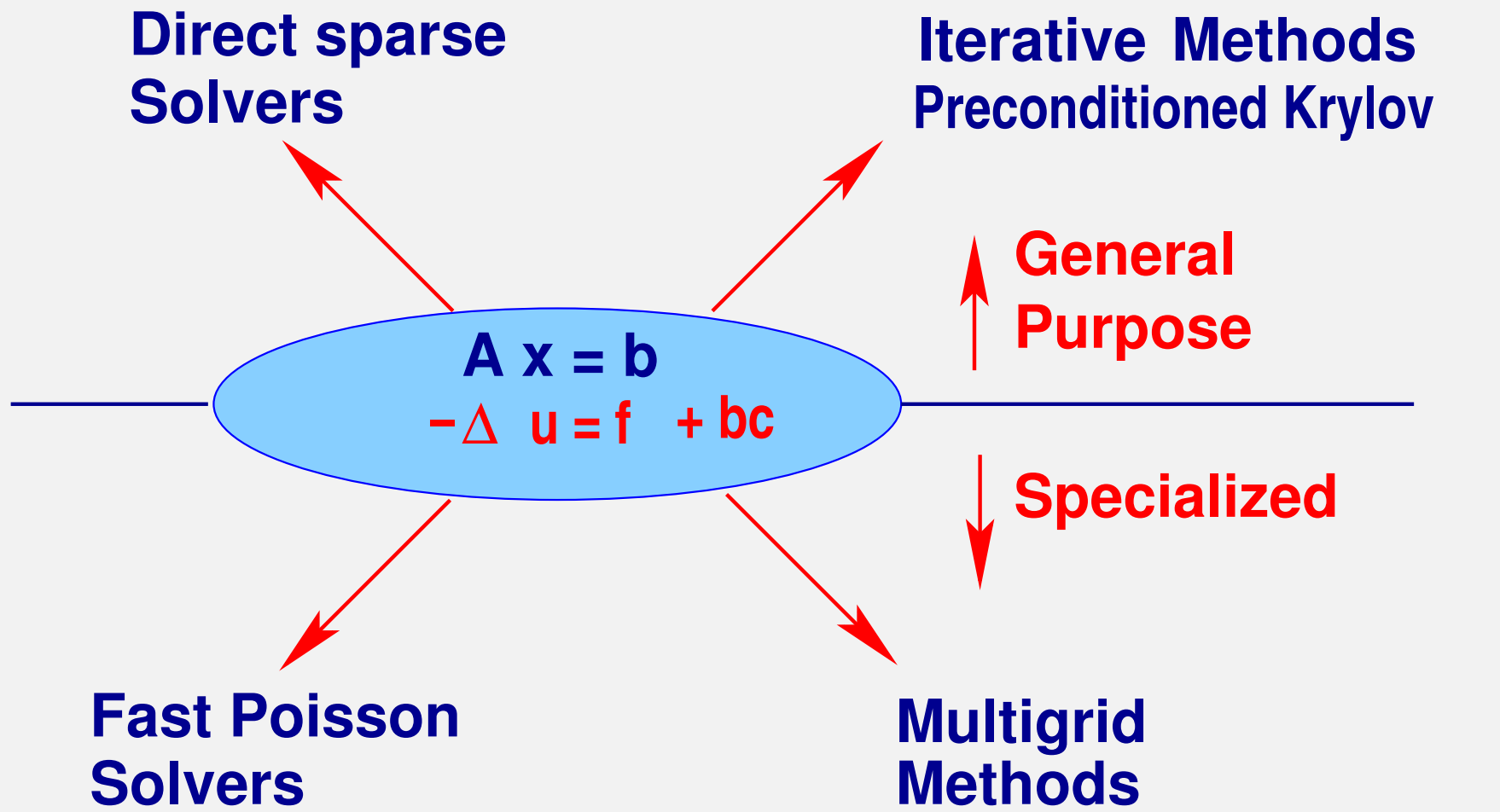
- What about data-mining and materials?
- Potential for a perfect union
- A lot of recent interest

The plan:

1. The traditional: Sparse iterative solvers
   - a brief tutorial
   - recent research
2. The challenging: Materials science
   - a brief tutorial
   - solving very large eigenvalue problems
3. The new: Data Mining/ Machine learning
   - a brief overview
   - dimension reduction

PART 1: SPARSE ITERATIVE SOLVERS

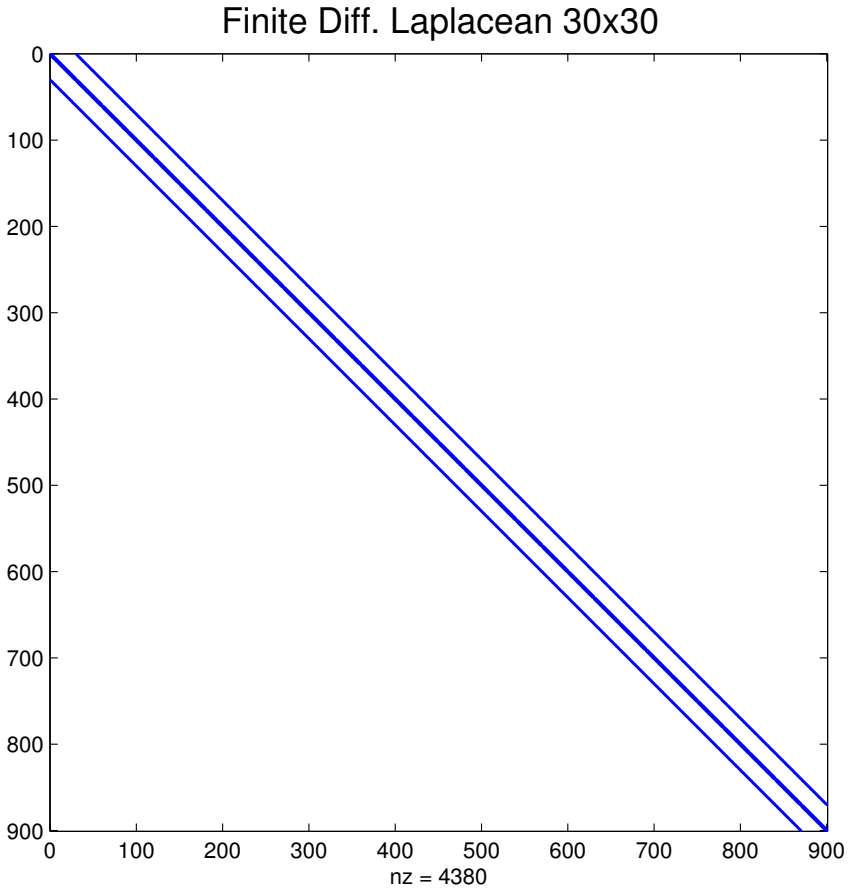
Introduction: Linear System Solvers

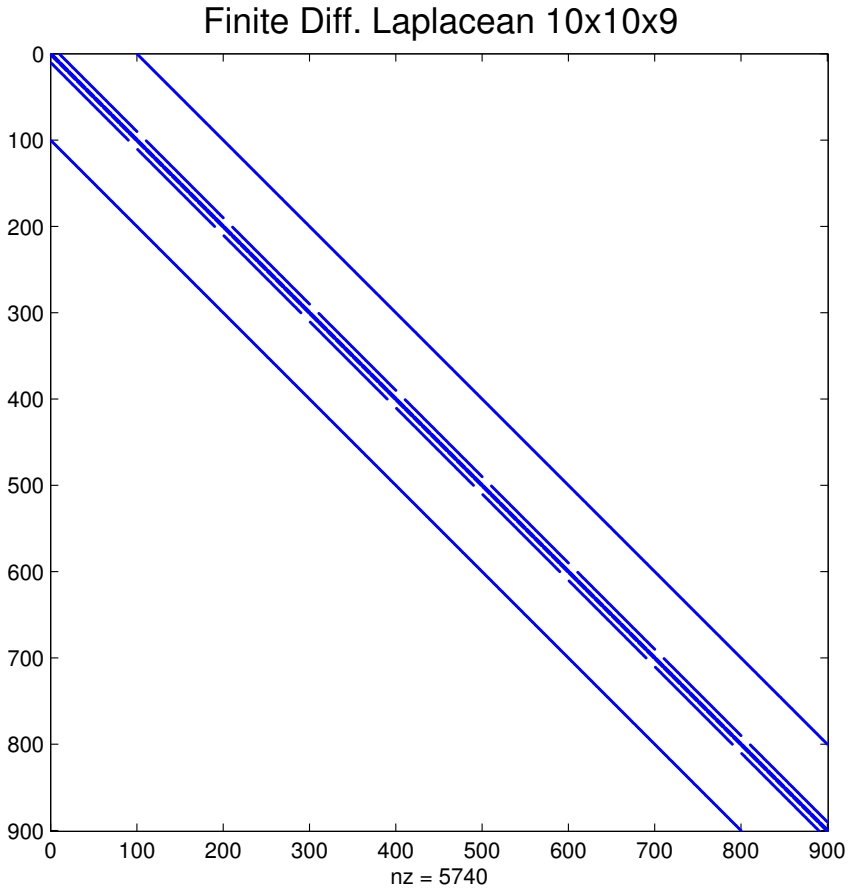


Long standing debate: direct vs. iterative

- Starting in the 1970's: huge progress of sparse direct solvers
- Iterative methods - much older - not designed for 'general systems'. Big push in the 1980s with help from 'preconditioning'
- General consensus now: Direct methods do well for 2-D problems and some specific applications [e.g., structures, ...]
- Usually too expensive for 3-D problems
- Huge difference between 2-D and 3-D case
- Test: Two Laplacean matrices of same dimension $n = 122,500$. First: on a 350×350 grid (2D); Second: on a $50 \times 50 \times 49$ grid (3D)

Pattern of a similar [much smaller] coefficient matrix





A few observations

➤ Problems are getting harder for Sparse Direct methods
(more 3-D models, much bigger problems,..)

➤ Problems are also getting difficult for iterative methods
Cause: more complex models - away from Poisson

➤ Researchers on both camps are learning each other's tricks to develop preconditioners.

Current Challenges:

(1) Scalable (HPC) performance [for general systems]

(2) Robustness, general purpose preconditioners

Background: Preconditioned Krylov subspace methods

Two ingredients:

- An accelerator: Conjugate gradient, BiCG, GMRES, BICGSTAB,.. ['Krylov subspace methods']
- A preconditioner: makes the system easier to solve by accelator, e.g. Incomplete LU factorizations; SOR/SSOR; Multigrid, ...

One viewpoint:

➤ Goal of preconditioner: generate good basic iterates.. [Gauss-Seidel, ILU, ...]

➤ Goal of accelerator: find best combination of these iterates

Acceleration: Krylov subspace methods

- Let x_0 = initial guess, and $r_0 = b - Ax_0$ = initial residual
- Define $K_m = \mathbf{span}\{r_0, Ar_0, \cdots, A^{m-1}r_0\}$ ...
- ... L_m another subspace of dim. m

Basic Krylov step: seek
$x_m = x_0 + \delta;\ \delta \in K_m$ such that
$b - Ax_m \perp L_m$

Projection method on K_m orthogonally to L_m

- Approximation theory viewpoint:

- $\boxed{x_m = x_0 + p_m(A)r_0}$ where p_m = polynomial of deg. $m-1$

Two common and imporant choices

1. $L_m = K_m$ $\rightarrow$ class of Galerkin or orthogonal projection methods (e.g., Conjugate Gradient Method). When A is SPD:

$$\|x^* - \tilde{x}\|_A = \min_{z \in K} \|x^* - z\|_A.$$

2. $L_m = AK_m$ $\rightarrow$ class of minimal residual methods: CR, GCR, ORTHOMIN, GMRES, CGNR, ... x_m satisfies:

$$\|b - A\tilde{x}\|_2 = \min_{z \in K} \|b - Az\|_2$$

➤ Key to success of Krylov methods: *Preconditioning*

Preconditioning – Basic principles

Use Krylov subspace method on a modified system, e.g.:

$$M^{-1}Ax = M^{-1}b.$$

- The matrix $M^{-1}A$ need not be formed explicitly; only need to solve $Mw = v$ whenever needed.
- Requirement : 'easy' to compute $M^{-1}v$ for arbitrary v
- Effect of preconditioner: spectrum of $M^{-1}A$ more favorable for Krylov subspace accelerators

Both A and M are SPD: Preconditioned CG (PCG)

ALGORITHM : 1 *Preconditioned Conjugate Gradient*

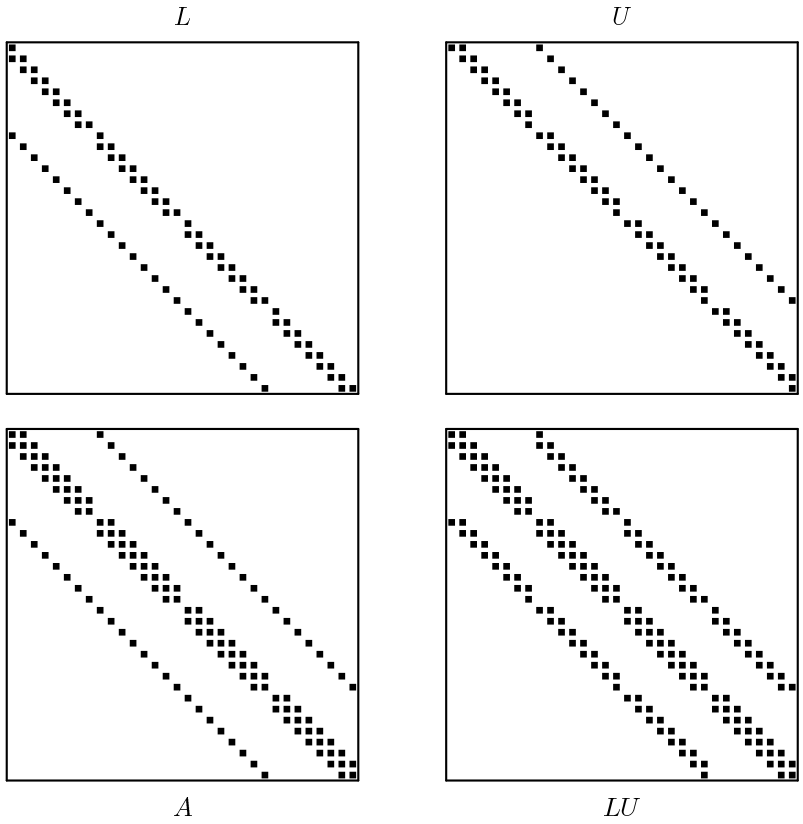
1. *Compute* $r_0 := b - Ax_0$, $z_0 = M^{-1}r_0$, *and* $p_0 := z_0$
2. *For* $j = 0, 1, \ldots,$ *until convergence Do:*
3. $\alpha_j := (r_j, z_j)/(Ap_j, p_j)$
4. $x_{j+1} := x_j + \alpha_j p_j$
5. $r_{j+1} := r_j - \alpha_j A p_j$
6. $z_{j+1} := M^{-1} r_{j+1}$
7. $\beta_j := (r_{j+1}, z_{j+1})/(r_j, z_j)$
8. $p_{j+1} := z_{j+1} + \beta_j p_j$
9. *EndDo*

➤ Krylov method for $M^{-1}Ax = M^{-1}b$ but use M-inner product to preserve self-adjointness.

Background: Incomplete LU (ILU) preconditioners

ILU: $A \approx LU$

Simplest Example: ILU(0) $\rightarrow$



Common difficulties of ILUs:

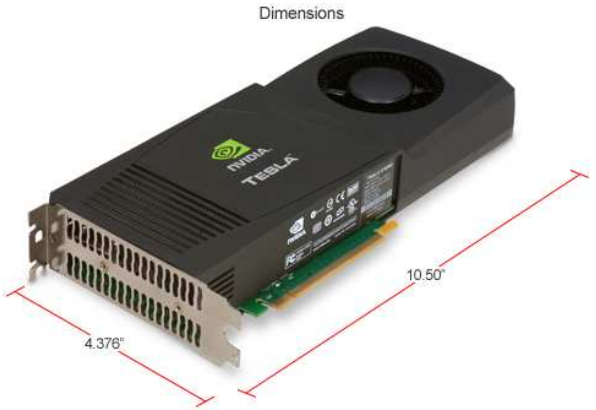
Often fail for indefinite problems

Not too good for highly parallel environments

Sparse matrix computations with GPUs **

- Very popular approach to: inexpensive supercomputing
- Can buy $\sim$ one Teraflop peak power for around $1,000

Tesla C1060 240 cores; 930GF peak



- Next: Fermi; followed by Kepler; then Maxwell
- Tesla K 80 : $2 \times 2{,}496 \rightarrow \mathbf{4992}$ GPU cores. 24 GB Mem.; Peak: $\approx$ 2.91 TFLOPS double prec. [with clock Boost].

The CUDA environment: The big picture

- A host (CPU) and an attached device (GPU)

Typical program:

1. Generate data on CPU
2. Allocate memory on GPU

```
cudaMalloc(...)
```

3. Send data Host → GPU

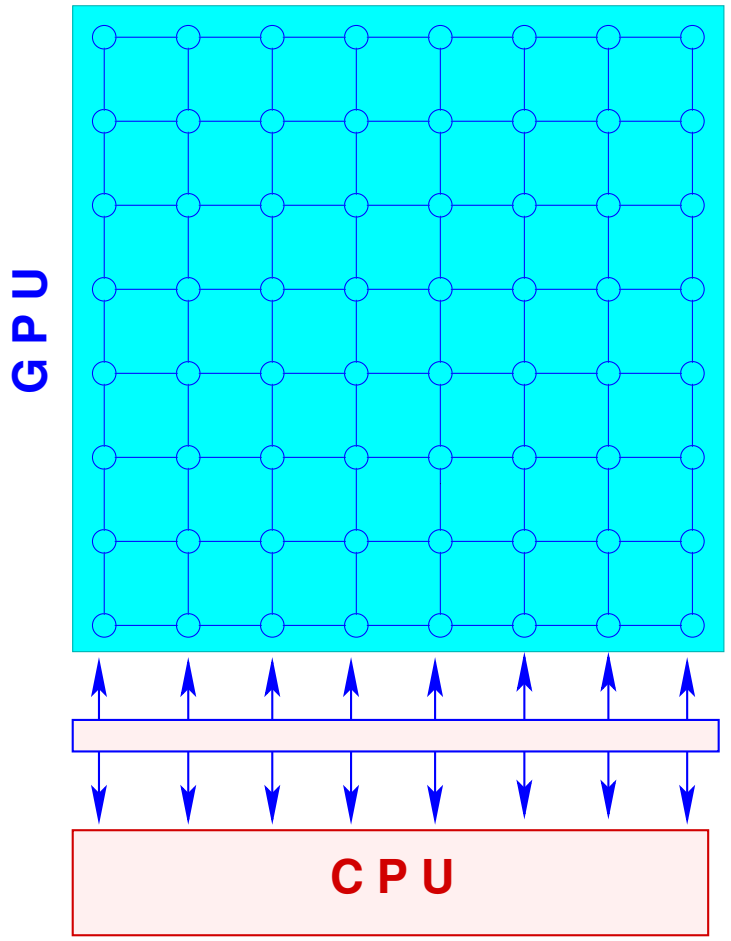
```
cudaMemcpy(...)
```

4. Execute GPU 'kernel':

```
kernel <<<(...)>>>(..)
```

5. Copy data GPU →CPU

```
cudaMemcpy(...)
```



Sparse matrix computations on GPUs

Main issue in using GPUs for sparse computations:

- Huge performance degradation due to 'irregular sparsity'

➤ Matrices:

Matrix -name	N	NNZ
FEM/Cantilever	62,451	4,007,383
Boeing/pwtk	217,918	11,634,424

➤ Performance of Mat-Vecs on NVIDIA Tesla C1060

	Single Precision			*Double Precision*		
Matrix	CSR	JAD	DIA+	CSR	JAD	DIA+
FEM/Cantilever	9.4	10.8	25.7	7.5	5.0	13.4
Boeing/pwtk	8.9	16.6	29.5	7.2	10.4	14.5

- More recent tests: NVIDIA M2070 (Fermi), Xeon X5675
- Double precision in Gflops

MATRIX	Dim. N	*CPU*	CSR	JAD	HYB	DIA
rma10	46,835	*3.80*	10.19	12.61	8.48	-
cfd2	123,440	*2.88*	8.52	11.95	12.18	-
majorbasis	160,000	*2.92*	4.81	11.70	11.54	13.06
af_shell8	504,855	*3.13*	10.34	14.56	14.27	-
lap7pt	1,000,000	*2.59*	4.66	11.58	12.44	18.70
atmosmodd	1,270,432	*2.09*	4.69	10.89	10.97	16.03

- *CPU* SpMV: Intel MKL, parallelized using OpenMP
- HYB: from CUBLAS Library. [Uses ellpack+csr combination]

(*) Thanks: all matrices from the Univ. Florida sparse matrix collection

Sparse Forward/Backward Sweeps

- Next major ingredient of precond. Krylov subs. methods

- ILU preconditioning operations require L/U solves: $x \leftarrow U^{-1}L^{-1}x$
- Sequential outer loop.

```
for i=1:n
    for j=ia(i):ia(i+1)
            x(i) = x(i) - a(j)*x(ja(j))
    end
end
```

- Parallelism can be achieved with level scheduling:

  - Group unknowns into levels
  - Compute unknowns $x(i)$ of same level simultaneously
  - $1 \leq nlev \leq n$

ILU: Sparse Forward/Backward Sweeps

- Very poor performance [relative to CPU]

Matrix	N	CPU	GPU-Lev	
		Mflops	#lev	Mflops
Boeing/bcsstk36	23,052	627	4,457	43
FEM/Cantilever	62,451	653	2,397	168
COP/CASEYK	696,665	394	273	142
COP/CASEKU	208,340	373	272	115

Prec: miserable :-)

GPU Sparse Triangular Solve with Level Scheduling

➤ Very poor performance when #levs is large

➤ A few things can be done to reduce the # levels but perf. will remain poor

So...

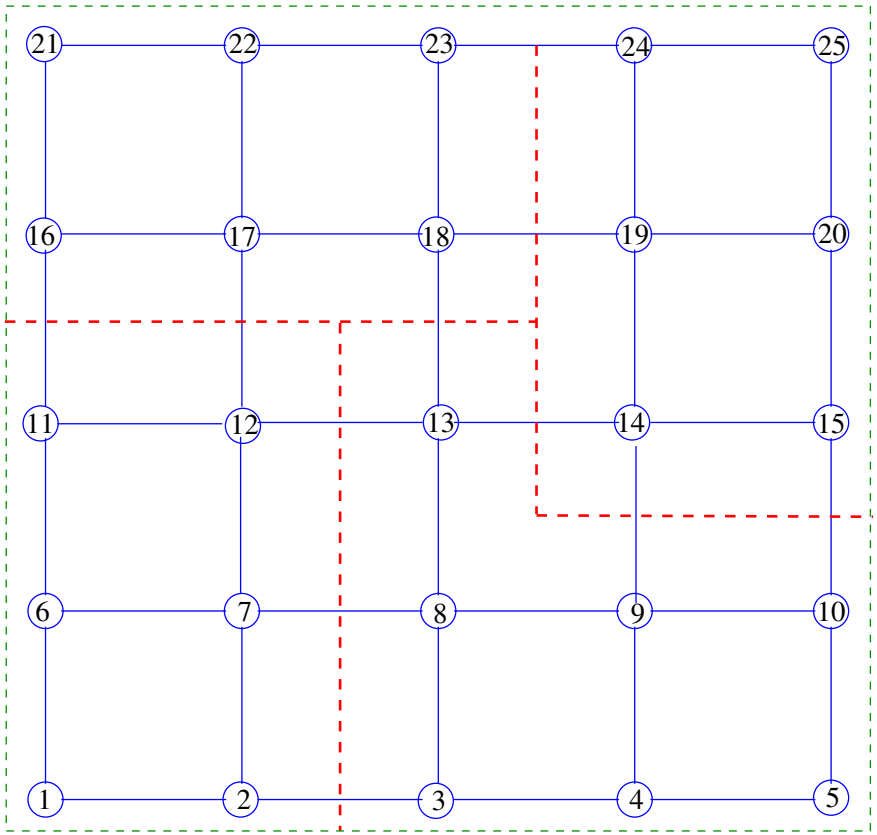
... prepare for the demise of the GPUs...

... or the demise of the ILUs ?

Alternative: Low-rank approximation preconditioners

- Goal: use standard Domain Decomposition framework
- Exploit Low-rank corrections
- Consider a domain partitioned in p sub-domains using vertex- based partitioniong (edge-separator)

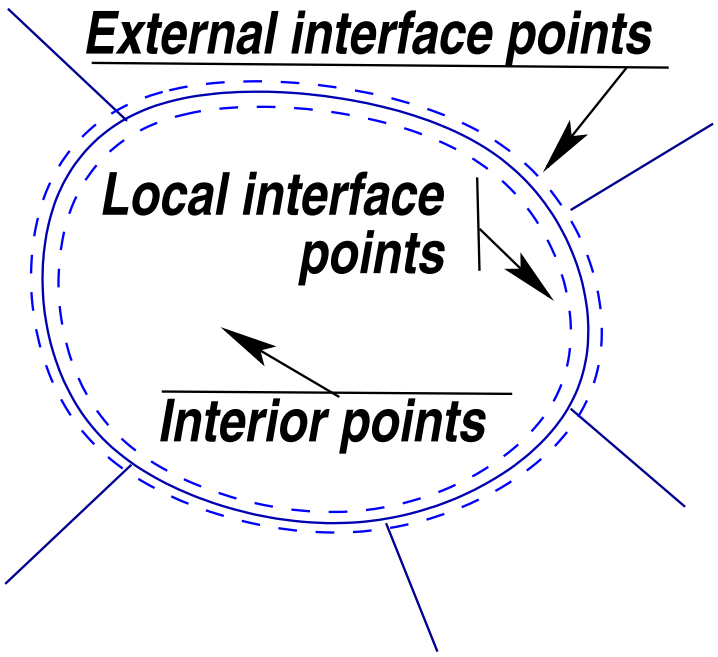
➤ Interface nodes in each domain are listed last.



The global system: Global view

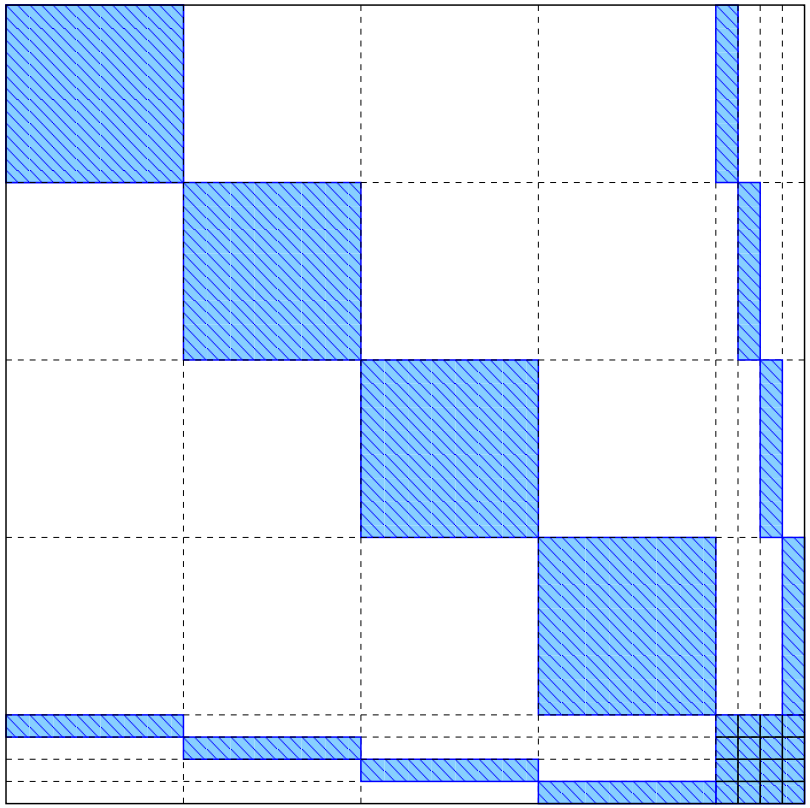
- Global system can be permuted to the form →
- u_i's internal variables
- y interface variables

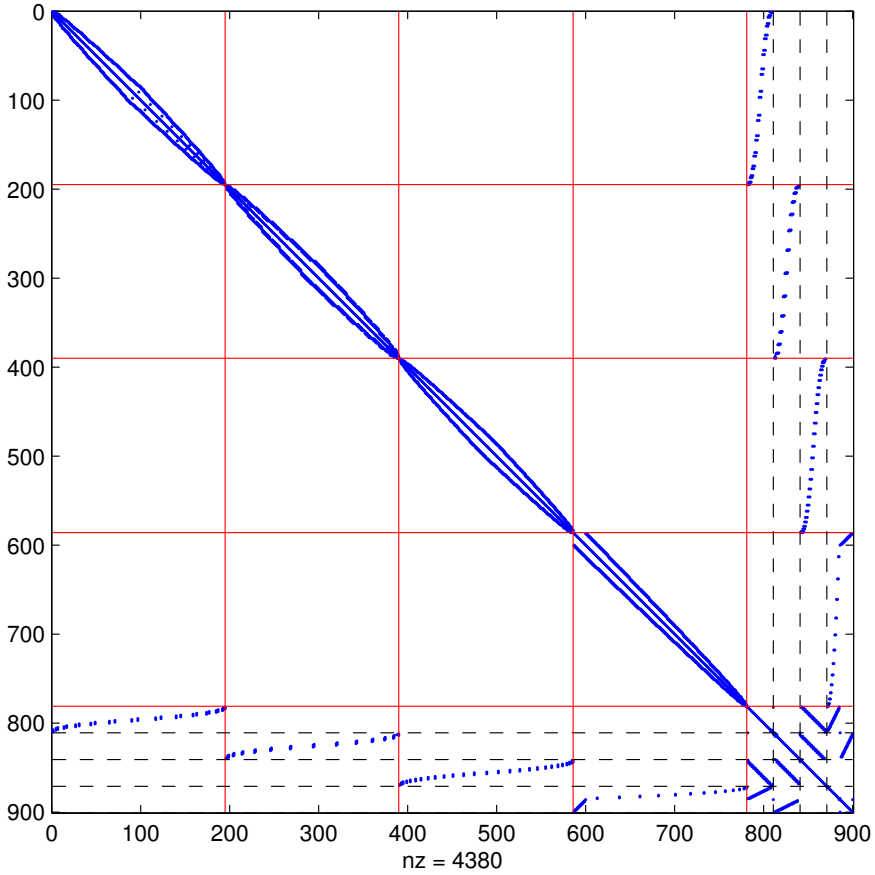
$$\begin{pmatrix} B_1 & & & \hat{F}_1 \\ & B_2 & & \hat{F}_2 \\ & & \ddots & \vdots \\ & & B_p & \hat{F}_p \\ \hat{E}_1^T & \hat{E}_2^T & \dots & C \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \\ \vdots \\ u_p \\ y \end{pmatrix} = b$$



- $\hat{F}_i$ maps local interface points to interior points in domain Ω_i
- $\hat{E}_i^T$ does the reverse operation

Example:




0
100
200
300
400
500
600
700
800
900
0
100
200
300
400
500
600
700
800
900
nz = 4380

Splitting

- Split as: $$A \equiv \begin{pmatrix} B & \hat{F} \\ \hat{E}^T & C \end{pmatrix} = \begin{pmatrix} B & \\ & C \end{pmatrix} + \begin{pmatrix} & \hat{F} \\ \hat{E}^T & \end{pmatrix}$$

- Define: $F \equiv \begin{pmatrix} \hat{F} \\ -I \end{pmatrix}$; $E \equiv \begin{pmatrix} \hat{E} \\ -I \end{pmatrix}$ Then:

$$\left[\begin{array}{c|c} B & \hat{F} \\ \hline \hat{E}^T & C \end{array}\right] = \left[\begin{array}{c|c} B + \hat{F}\hat{E}^T & 0 \\ \hline 0 & C + I \end{array}\right] - FE^T.$$

- Property: $\hat{F}\hat{E}^T$ is 'local', i.e., no inter-domain couplings $\rightarrow$

$$A_0 \equiv \left[\begin{array}{c|c} B + \hat{F}\hat{E}^T & 0 \\ \hline 0 & C + I \end{array}\right] = \text{block-diagonal}$$

Low-Rank Approximation DD preconditioners

Sherman-Morrison $\longrightarrow$

$$A^{-1} = A_0^{-1} + A_0^{-1} F G^{-1} E^T A_0^{-1}$$
$$G \equiv I - E^T A_0^{-1} F$$

Options: (a) Approximate $A_0^{-1}F, E^T A_0^{-1}, G^{-1}$
(b) Approximate only G^{-1} [this talk]

➤ (b) requires 2 solves with A_0.

Let $G \approx G_k$
Preconditioner $\longrightarrow$

$$M^{-1} = A_0^{-1} + A_0^{-1} F G_k^{-1} E^T A_0^{-1}$$

Symmetric Positive Definite case

- Recap: Let $\boxed{G \equiv I - E^T A_0^{-1} E \equiv I - H}$. Then

$$A^{-1} = A_0^{-1} + A_0^{-1} E G^{-1} E^T A_0^{-1}$$

- Approximate G^{-1} by $G_k^{-1} \rightarrow$ preconditioner:

$$M^{-1} = A_0^{-1} + (A_0^{-1} E) G_k^{-1} (E^T A_0^{-1})$$

- Matrix A_0 is SPD
- Can show: $0 \leq \lambda_j(H) < 1$.

- Now take rank-k approximation to H:

$$H \approx U_k D_k U_k^T \qquad G_k = I - U_k D_k U_k^T \quad \rightarrow$$

$$G_k^{-1} \equiv (I - U_k D_k U_k^T)^{-1} = I + U_k[(I - D_k)^{-1} - I]U_k^T$$

- Observation: $A^{-1} = M^{-1} + A_0^{-1} E[G^{-1} - G_k^{-1}] E^T A_0^{-1}$
- G_k: k largest eigenvalues of H matched – others set == 0
- Result: AM^{-1} has
  - $n - s + k$ eigenvalues == 1
  - All others between 0 and 1

Alternative: reset lowest eigenvalues to constant

- Let $H = U\Lambda U^T$ = exact (full) diagonalization of H
- We replaced Λ by:

$$\begin{pmatrix} \lambda_1 & & & & & & \\ & \lambda_2 & & & & & \\ & & \ddots & & & & \\ & & & \lambda_k & & & \\ & & & & 0 & & \\ & & & & & \ddots & \\ & & & & & & 0 \end{pmatrix}$$

- Alternative: replace Λ by

$$\begin{pmatrix} \lambda_1 & & & & & & \\ & \lambda_2 & & & & & \\ & & \ddots & & & & \\ & & & \lambda_k & & & \\ & & & & \theta & & \\ & & & & & \ddots & \\ & & & & & & \theta \end{pmatrix}$$

- Interesting case: $\theta = \lambda_{k+1}$
- Question: related approximation to G^{-1}?

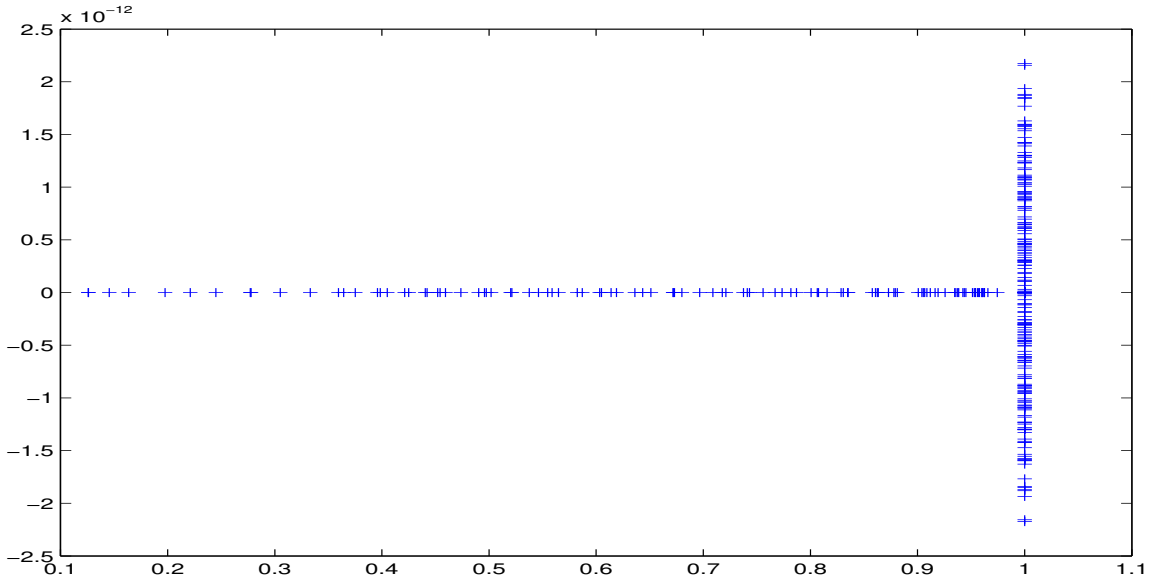
- Result: Let $\gamma = 1/(1-\theta)$. Then approx. to G^{-1} is:

$$G_{k,\theta}^{-1} \equiv \gamma I + U_k[(I - D_k)^{-1} - \gamma I]U_k^T$$

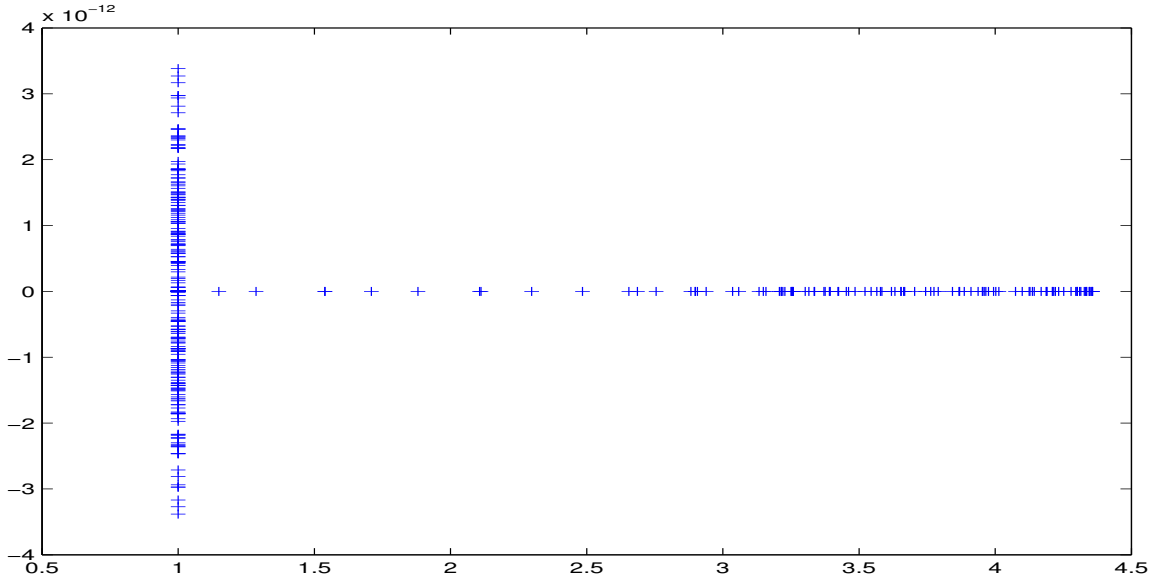
- G_k: k largest eigenvalues of G matched – others set == θ
- $\theta = 0$ yields previous case
- When $\lambda_{k+1} \leq \theta < 1$ we get
- Result: AM^{-1} has
  - $n - s + k$ eigenvalues == 1
  - All others ≥ 1
- Next: An example for a 900×900 Laplacean, 4 domains, $s = 119$.

Eigenvalues of AM^{-1}. Used: $k = 5$. Two cases

$\theta = 0$



$\theta = \lambda_{k+1}$



Proposition Assume θ is so that $\boxed{\lambda_{k+1} \leq \theta < 1}$. Then the eigenvalues η_i of AM^{-1} satisfy:

$$1 \leq \eta_i \leq 1 + \frac{1}{1-\theta} \|A^{1/2} A_0^{-1} E\|_2^2.$$

- Can Show: For the Laplacean (FD)

$$\|A^{1/2} A_0^{-1} E\|_2^2 = \|E^T A_0^{-1} A A_0^{-1} E\|_2 \leq \frac{1}{4}$$

regardless of the mesh-size.

- Best upper bound for $\theta = \lambda_{k+1}$
- Set $\theta = \lambda_{k+1}$. Then $\kappa(AM^{-1}) \leq$ constant, if k large enough so that $\lambda_{k+1} \leq$ constant.
- i.e., need to capture sufficient part of spectrum

The symmetric indefinite case

- Appeal of this approach over ILU: approximate inverse $\rightarrow$ Not as sensitive to indefiniteness
- Part of the results shown still hold
- But $\lambda_i(H)$ can be > 1 now.
- Can change the setting slightly [by introducing a parameter α] to improve diagonal dominance
- Details skipped.

Parallel implementations

- Recall :

$$M^{-1} = A_0^{-1}\left[I + EG_{k,\theta}^{-1}E^T A_0^{-1}\right]$$
$$G_{k,\theta}^{-1} = \gamma I + U_k[(I - D_k)^{-1} - \gamma I]U_k^T$$

- Steps involved in applying M^{-1} to a vector x :

ALGORITHM : 2 *Preconditioning operation*

1. $z = A_0^{-1}x$ // $\hat{B}_i$-solves and C_* solve ($C_* \equiv C + I$)
2. $y = E^T z$ // Interior points to interface (Loc.)
3. $y_k = G_{k,\theta}^{-1}y$ // Use Low-Rank approx.
4. $z_k = Ey_k$ // Interface to interior points (Loc.)
5. $u = A_0^{-1}(x + z_k)$ // $\hat{B}_i$-solves and C_*- solve

A_0 Solves Note:

$$A_0 = \begin{pmatrix} \hat{B}_1 & & & & \\ & \hat{B}_2 & & & \\ & & \ddots & & \\ & & & \hat{B}_p & \\ & & & & C_* \end{pmatrix}$$

- Recall $\hat{B}_i = B_i + E_i E_i^T$
- A solve with A_0 amounts to all p $\hat{B}_i$-solves and a C_*-solve
- Can replace C_*^{-1} by a low degree polynomial [Chebyshev]
- Can use any solver for the $\hat{B}_i$'s

Parallel tests: Itasca (MSI)

- HP linux cluster- with Xeon 5560 ("Nehalem") processors
- Simple Poisson equation on regular grid

2-D

Mesh	Nproc	Rank	#its	Prec-t	Iter-t
256×256	2	8	29	2.30	.343
512×512	8	16	57	2.62	.747
1024×1024	32	32	96	3.30	1.32
2048×2048	128	64	154	4.84	2.38

3-D

Mesh	Nproc	Rank	#its	Prec-t	Iter-t
$32 \times 32 \times 32$	2	8	12	1.09	.0972
$64 \times 64 \times 64$	16	16	31	1.18	.381
$128 \times 128 \times 128$	128	32	62	2.42	.878

PART 2: ALGORITHMS FOR ELECTRONIC STRUCTURE

Approximations/theories used

- Original Schrödinger equation very complex
- Approximations developed started the 1930s reached excellent level of accuracy. Among them...

Density Functional Theory: (DFT) observable quantities are uniquely determined by ground state charge density.

Kohn-Sham equation:

$$\left[-\frac{\nabla^2}{2} + V_{ion} + V_H + V_{xc}\right]\Psi = E\Psi$$

The three potential terms

$$\left[-\frac{\nabla^2}{2} + V_{ion} + V_H + V_{xc}\right]\Psi(r) = E\Psi(r)$$ With:

- Charge density: $\rho(r) = \sum_{i=1}^{occup} |\psi_i(r)|^2$
- Hartree potential (local) $\nabla^2 V_H = -4\pi\rho(r)$
- V_{xc} depends on functional. For LDA: $V_{xc} = f(\rho(r))$
- V_{ion} = nonlocal – does not explicitly depend on ρ $V_{ion} = V_{loc} + \sum_a P_a$

➤ Note: V_H and V_{xc} depend nonlinearly on eigenvectors

Self Consistence

➤ The potentials and/or charge densities must be self-consistent: Can be viewed as a nonlinear eigenvalue problem. Can be solved using different viewpoints

- Nonlinear eigenvalue problem: Linearize + iterate to self-consistence
- Nonlinear optimization: minimize energy [again linearize + achieve self-consistency]

The two viewpoints are more or less equivalent

➤ Preferred approach: Broyden-type quasi-Newton technique

➤ Typically, a small number of iterations are required

Self-Consistent Iteration

➤ Most time-consuming part = computing eigenvalues / eigenvectors.

Characteristic : Large number of eigenvalues /-vectors to compute [occupied states].

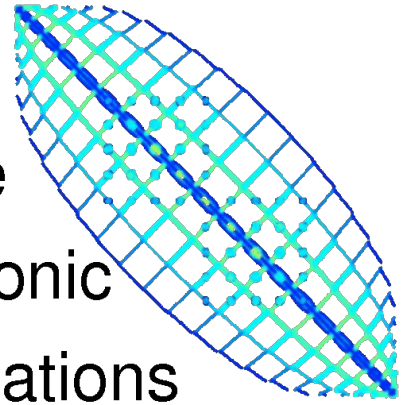
➤ Self-consistent loop takes a few iterations (say 10 or 20 in easy cases).

Challenge: Compute a large number of eigenvalues ($n_{ev} \approx 10^4$–10^5) of a large Hamiltonian matrix ($N \approx 10^7$–10^8)

PARSEC

- Represents ≈ 15 years of effort by a multidisciplinary team
- Real-space Finite Difference;
- Efficient diagonalization, ...
- Exploits symmetry, ...
- PARSEC Released in ∼ 2005.

Pseudopotential
Algorithm for
Real–
Space
Electronic
Calculations



DIAGONALIZATION: CHEBYSHEV FILTERING

Subspace iteration with Chebyshev filtering

Given a basis $[v_1, \ldots, v_m]$, 'filter' each vector as

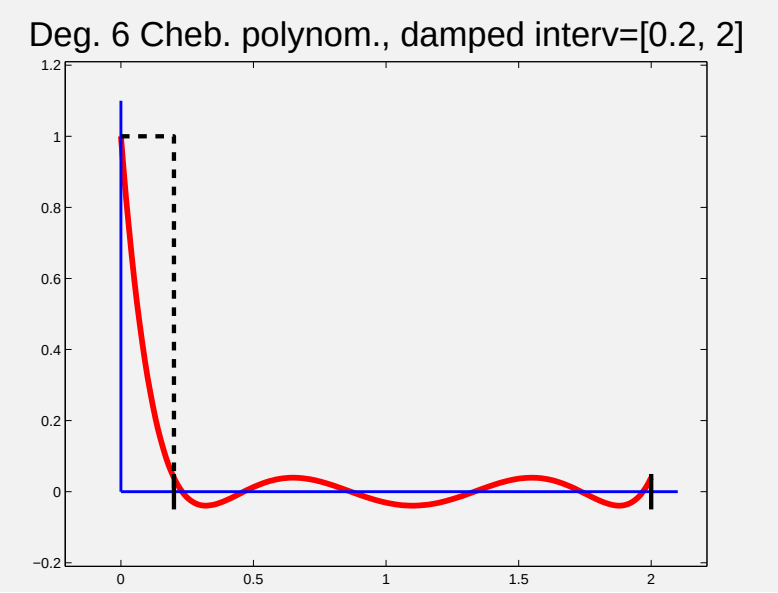
$$\hat{v}_i = p_k(A) v_i$$

➤ p_k = Low deg. polynomial. Enhances wanted eigencomponents

The filtering step is not used to compute eigenvectors accurately ➤

SCF & diagonalization loops merged

Important: convergence still good and robust



Main step:

$$
\begin{array}{ll}
\text{Previous basis} & V = [v_1, v_2, \cdots, v_m] \\
& \downarrow \\
\text{Filter} & \hat{V} = [p(A)v_1, p(A)v_2, \cdots, p(A)v_m] \\
& \downarrow \\
\text{Orthogonalize} & [V, R] = qr(\hat{V}, 0)
\end{array}
$$

- The basis V is used to do a Ritz step (basis rotation)

  $$C = V^T A V \rightarrow [U, D] = eig(C) \rightarrow V := V * U$$

- Update charge density using this basis.
- Update Hamiltonian — repeat

➤ In effect: Nonlinear subspace iteration

➤ Main advantages: (1) very inexpensive, (2) uses minimal storage (m is a little $\geq$ # states).

➤ Filter polynomials: if $[a,\ b]$ is interval to dampen, then

$$p_k(t) = \frac{C_k(l(t))}{C_k(l(c))}; \quad \text{with} \quad l(t) = \frac{2t - b - a}{b - a}$$

- $c \approx$ eigenvalue farthest from $(a + b)/2$ – used for scaling

➤ 3-term recurrence of Chebyshev polynommial exploited to compute $p_k(A)v$. If $B = l(A)$, then $C_{k+1}(t) = 2tC_k(t) - C_{k-1}(t) \rightarrow$

$$w_{k+1} = 2Bw_k - w_{k-1}$$

Tests with Silicon and Iron clusters (old)

Legend:

- n_{state} : number of states
- n_H : size of Hamiltonian matrix
- # $A * x$: number of total matrix-vector products
- # SCF : number of iteration steps to reach self-consistency
- $\frac{total\_eV}{atom}$: total energy per atom in electron-volts
- 1st CPU : CPU time for the first step diagonalization
- total CPU : total CPU spent on diagonalizations

Reference: Y. Zhou, Y.S., M. L. Tiago, and J. R. Chelikowsky, Phy. Rev. E, vol. 74, p. 066704 (2006).

A comparison: $Si_{525}H_{276}$

method	# $A * x$	SCF its.	CPU(secs)
ChebSI	124761	11	5946.69
ARPACK	142047	10	62026.37
TRLan	145909	10	26852.84

Polynomial degree $= 8$. Total energies agree to within 8 digits

A larger system: $Si_{9041}H_{1860}$

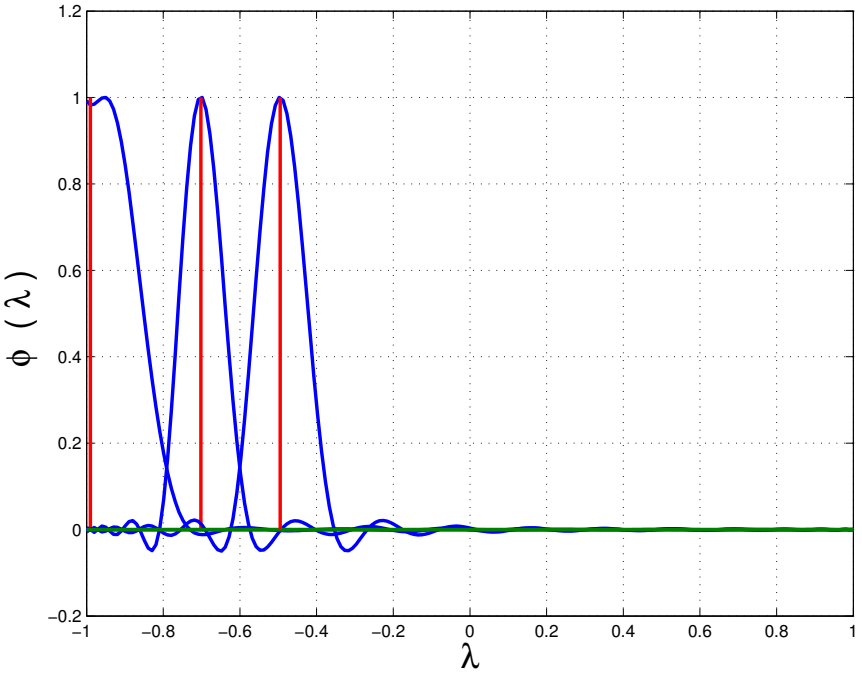
n_{state}	# $A * x$	# SCF	$\frac{total\_eV}{atom}$	1st CPU	total CPU
19015	4804488	18	-92.00412	102.12 h.	294.36 h.

PEs = 48; n_H =2,992,832. Pol. Deg. $= 8$.

Spectrum Slicing and the EVSL project

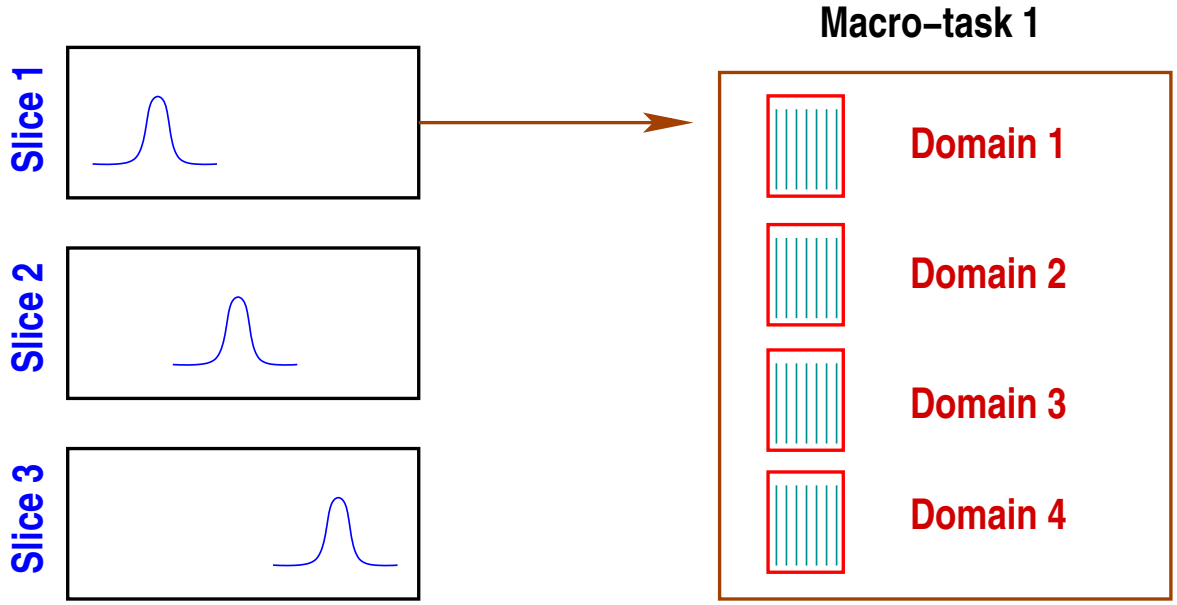
- Part of our DOE project on excited states

Conceptually simple idea: cut the overall interval containing the spectrum into small sub-intervals and compute eigen-pairs in each sub-interval *inde-pendently.*
Tool: polynomial filtering



- To avoid repeting eigenpairs, keep only the computed eigen-values / vectors that are located in support interval

Levels of parallelism



The two main levels of parallelism in EVSL

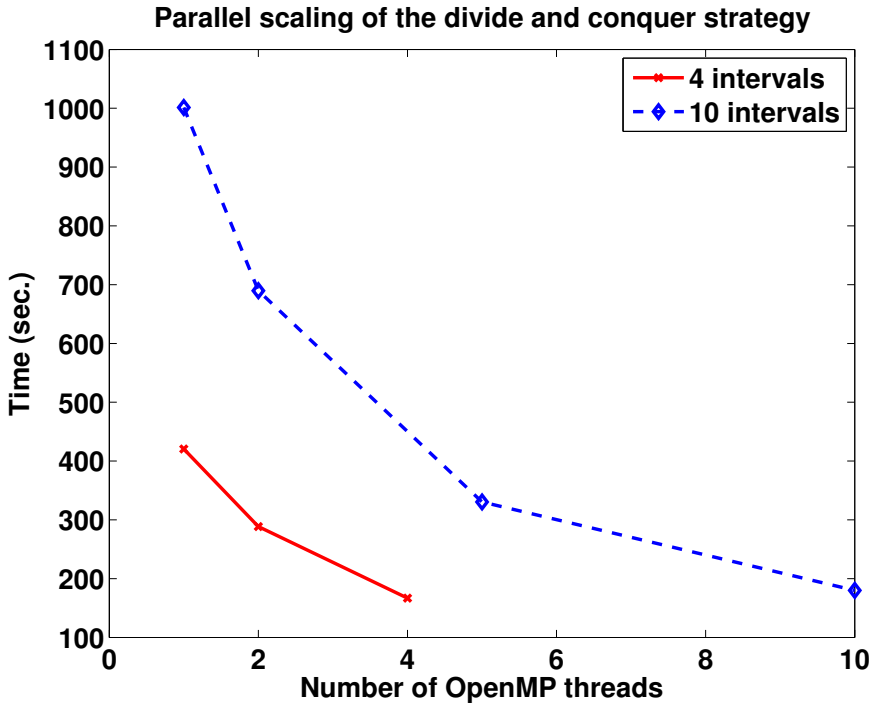
Simple Parallelism: across intervals

➤ Use OpenMP to illustrate scaling: Compute 1002 lowest eigenpairs of matrix `Si0` ($n = 33,401, nnz = 1,317,655$.)

➤ Use 4 and then 10 spectral slices

➤ Near optimal scaling observed as # cores increases

➤ Note: 2nd level of parallelism [re-orthogonalization + MatVecs] not exploited.



➤ Faster solution times with 4 slices (≈ 250 evals per slice) than with 10 slices (≈ 100 evals per slice)

PART 3: DATA MINING

Introduction: a few factoids

- Data is growing exponentially at an "alarming" rate:
  - 90% of data in world today was created in last two years
  - Every day, 2.3 Million terabytes (2.3×10^{18} bytes) created
- Mixed blessing: Opportunities & big challenges.
- Trend is re-shaping & energizing many research areas ...
- ... including my own: numerical linear algebra

Introduction: What is data mining?

Set of methods and tools to extract meaningful information or patterns from (big) datasets. Broad area : data analysis, machine learning, pattern recognition, information retrieval, ...

➤ Tools used: linear algebra; Statistics; Graph theory; Approximation theory; Optimization; ...

➤ This talk: brief introduction – emphasis on linear algebra viewpoint

➤ + our initial work on materials.

➤ Focus on "Dimension reduction methods"

Lots of data: can be hugely beneficial (e.g., Health sciences)

Home → Medical Computing → IBM's Watson Could Diagnose Cancer Better Than Doctors

IBM's Watson Could Diagnose Cancer Better Than Doctors

36

Posted in Medical Computing by Qmed Staff on October 22, 2013

Several years ago, IBM's Watson supercomputer gained fame after beating some of the world's top Jeopardy! players. To accomplish that feat, researchers fed thousands of points of information into Watson's database, allowing it to retrieve information presented through natural language. While winning Jeopardy! might be an exciting challenge for researchers, Watson's next goal could revolutionize oncology. IBM is currently working on the third-generation of the Watson platform, which has the power to debate and reason, according to IBM CEO Ginni Rometty.

The third generation of IBM's Watson platform will be able to actively reason.

The latest version of Watson can absorb and analyze vast amounts of data, allowing it to make diagnoses that are more accurate than human doctors. If a Watson-style computer was deployed through a cloud interface, healthcare facilities may be able to improve diagnosis accuracy, reduce costs and minimize patient wait times.

In combination with the Memorial Sloan-Kettering Cancer Center and Wellpoint, a private healthcare company

Relat

Medt
Reco
by Ste

Edge
by Ton

Targe
by Ton

Test
by Ton

Incre
by Ton

Major tool of Data Mining: Dimension reduction

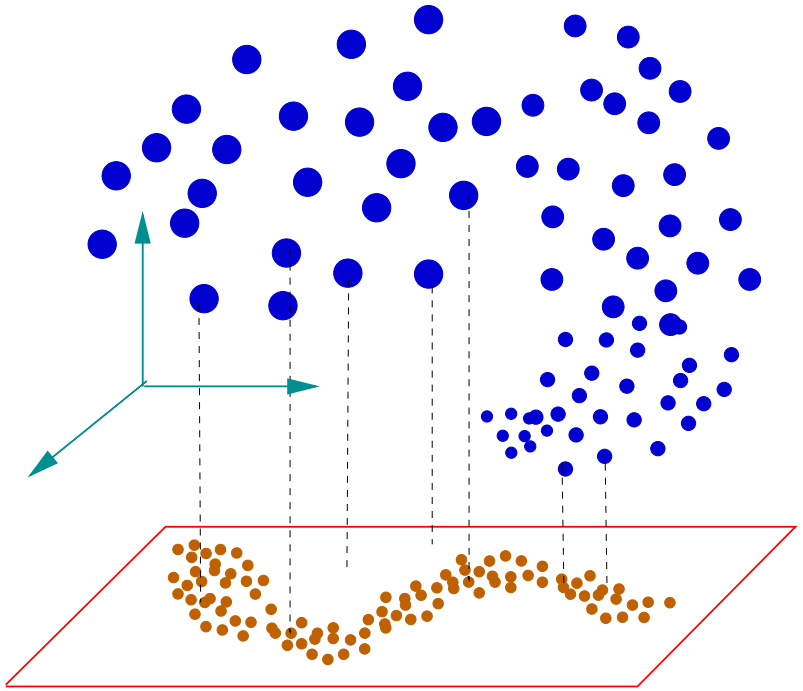
➤ Goal is not as much to reduce size (& cost) but to:

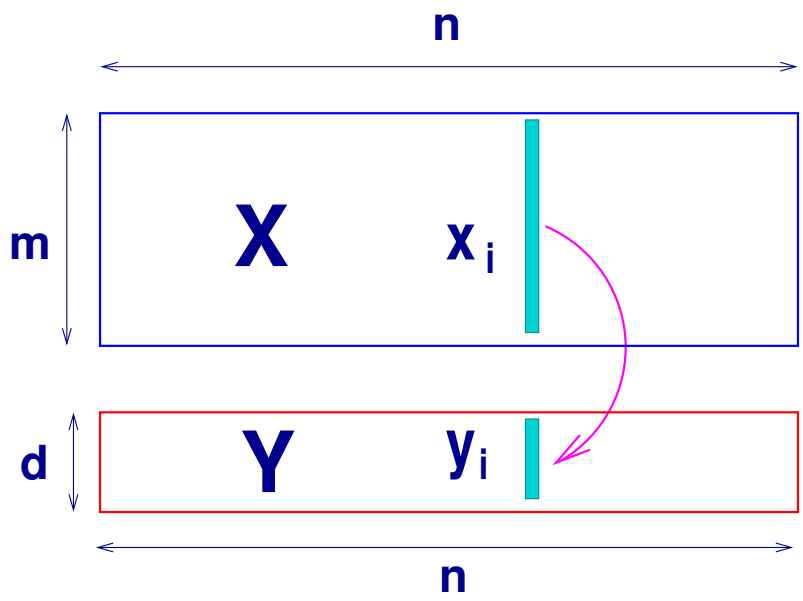
- Reduce noise and redundancy in data before performing a task [e.g., classification as in digit/face recognition]
- Discover important 'features' or 'paramaters'

The problem: Given: $\boldsymbol{X} = [x_1, \cdots, x_n] \in \mathbb{R}^{m \times n}$, find a low-dimens. representation $\boldsymbol{Y} = [y_1, \cdots, y_n] \in \mathbb{R}^{d \times n}$ of $\boldsymbol{X}$

➤ Achieved by a mapping $\Phi : x \in \mathbb{R}^m \longrightarrow y \in \mathbb{R}^d$ so:

$$\phi(x_i) = y_i, \quad i = 1, \cdots, n$$





- Φ may be linear : $y_i = W^\top x_i$, i.e., $Y = W^\top X$, ..
- ... or nonlinear (implicit).
- Mapping Φ required to: Preserve proximity? Maximize variance? Preserve a certain graph?

Example: Principal Component Analysis (PCA)

In *Principal Component Analysis* $\boldsymbol{W}$ is computed to maximize variance of projected data:

$$\max_{\boldsymbol{W}\in\mathbb{R}^{m\times d};\boldsymbol{W}^\top\boldsymbol{W}=\boldsymbol{I}} \sum_{i=1}^{n}\left\|\boldsymbol{y}_i-\frac{1}{\boldsymbol{n}}\sum_{j=1}^{n}\boldsymbol{y}_j\right\|_2^2,\ \boldsymbol{y}_i=\boldsymbol{W}^\top\boldsymbol{x}_i.$$

- Leads to maximizing

$$\text{Tr}\left[\boldsymbol{W}^\top(\boldsymbol{X}-\boldsymbol{\mu}\boldsymbol{e}^\top)(\boldsymbol{X}-\boldsymbol{\mu}\boldsymbol{e}^\top)^\top\boldsymbol{W}\right],\quad \boldsymbol{\mu}=\tfrac{1}{n}\Sigma_{i=1}^{n}\boldsymbol{x}_i$$

- Solution $\boldsymbol{W}=\{$ dominant eigenvectors $\}$ of the covariance matrix $\equiv$ Set of left singular vectors of $\bar{\boldsymbol{X}}=\boldsymbol{X}-\boldsymbol{\mu}\boldsymbol{e}^\top$

SVD:

$$\bar{\boldsymbol{X}} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top, \quad \boldsymbol{U}^\top\boldsymbol{U} = \boldsymbol{I}, \quad \boldsymbol{V}^\top\boldsymbol{V} = \boldsymbol{I}, \quad \boldsymbol{\Sigma} = \text{Diag}$$

- Optimal $\boldsymbol{W} = \boldsymbol{U}_d \equiv$ matrix of first $\boldsymbol{d}$ columns of $\boldsymbol{U}$
- Solution $\boldsymbol{W}$ also minimizes 'reconstruction error' ..

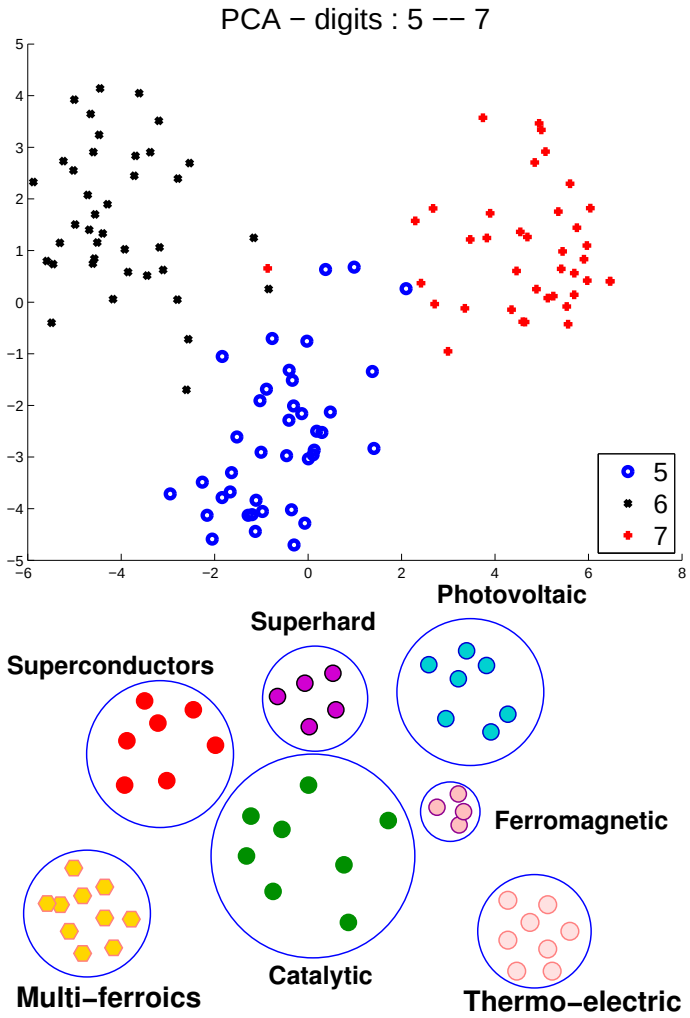
$$\sum_i \|\boldsymbol{x}_i - \boldsymbol{W}\boldsymbol{W}^T\boldsymbol{x}_i\|^2 = \sum_i \|\boldsymbol{x}_i - \boldsymbol{W}\boldsymbol{y}_i\|^2$$

- In some methods recentering to zero is not done, i.e., $\bar{\boldsymbol{X}}$ replaced by $\boldsymbol{X}$.

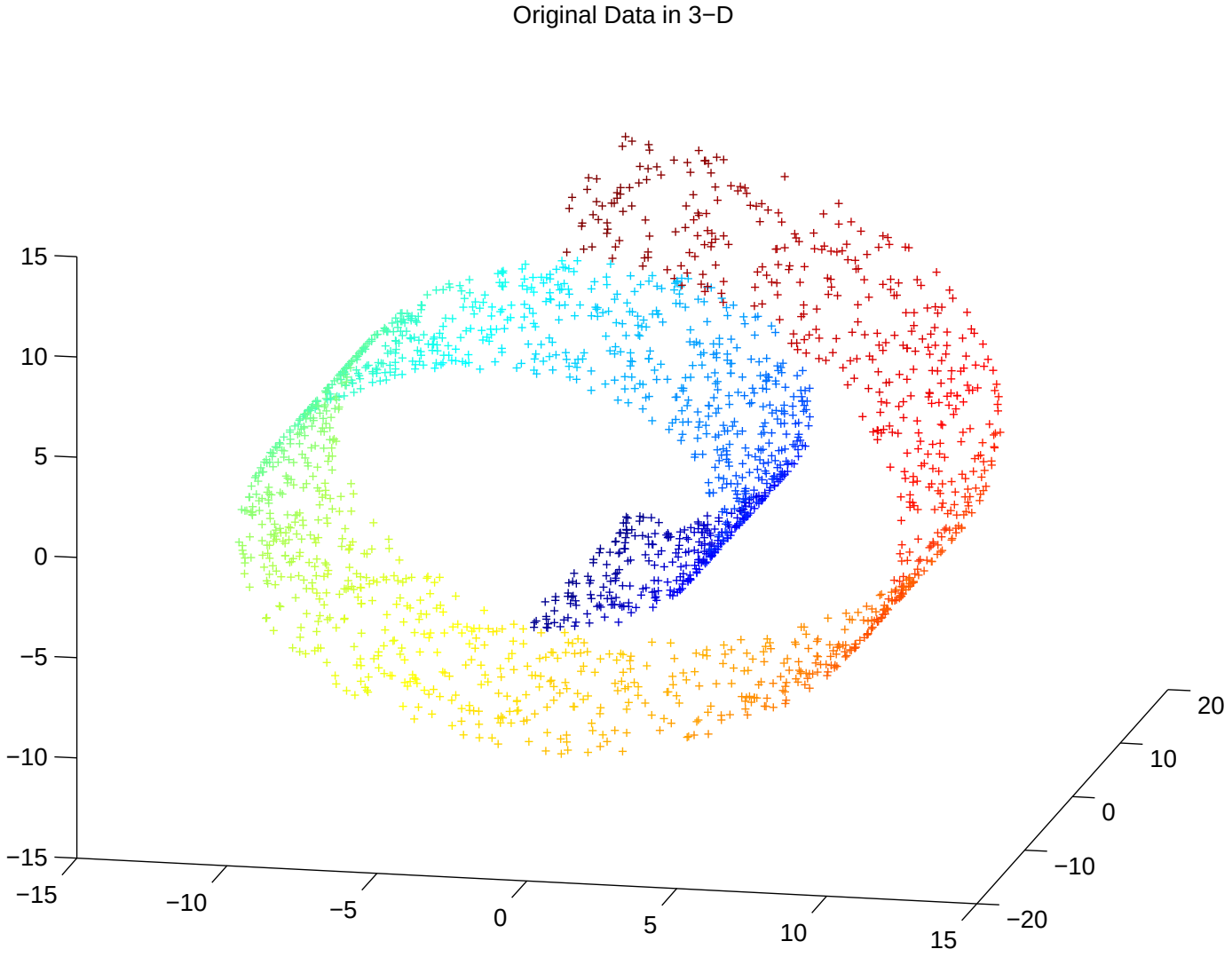
Unsupervised learning

"Unsupervised learning" : methods that do not exploit known labels

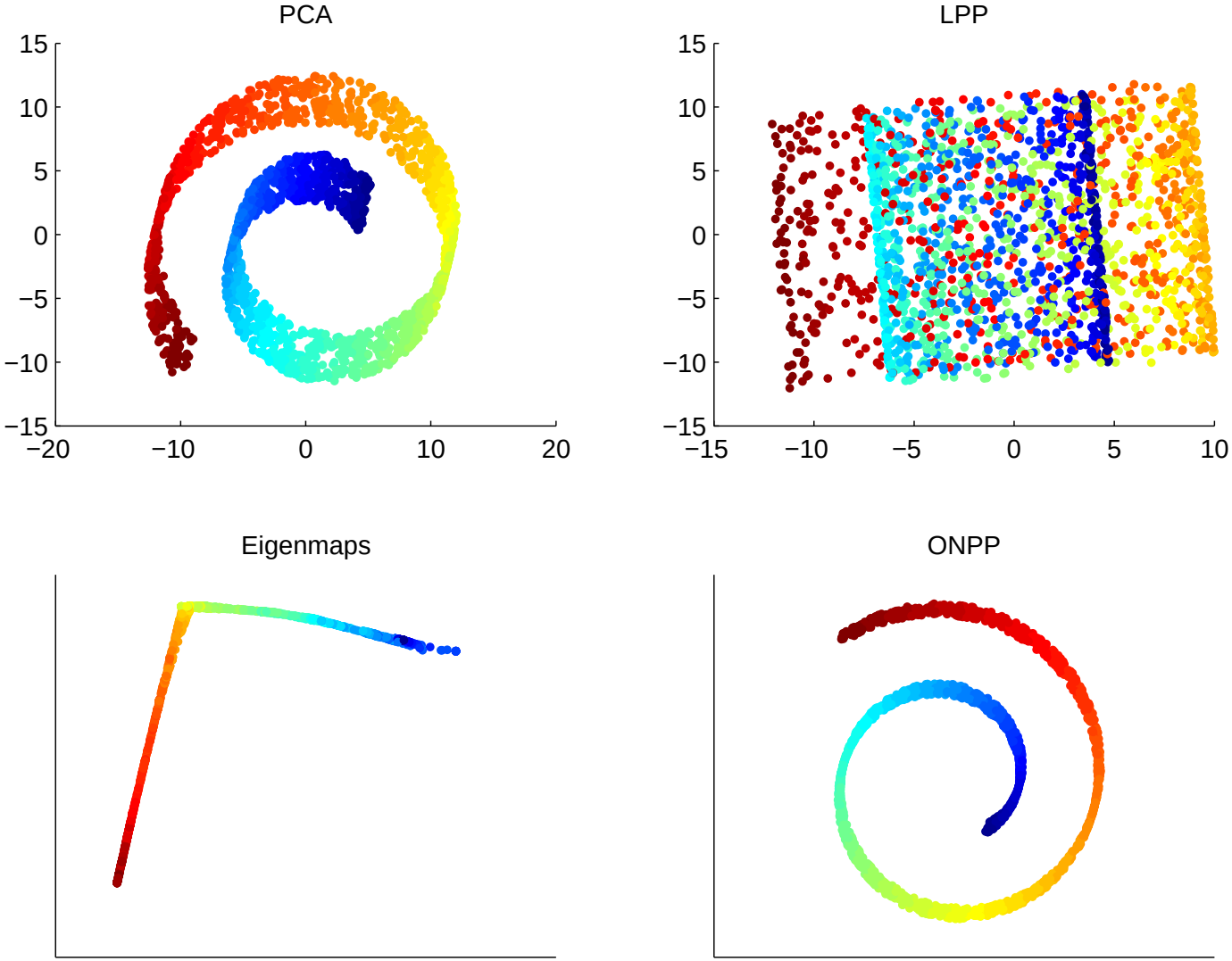
- Example of digits: perform a 2-D projection
- Images of same digit tend to cluster (more or less)
- Such 2-D representations are popular for visualization
- Can also try to find natural clusters in data, e.g., in materials
- Basic clusterning technique: K-means



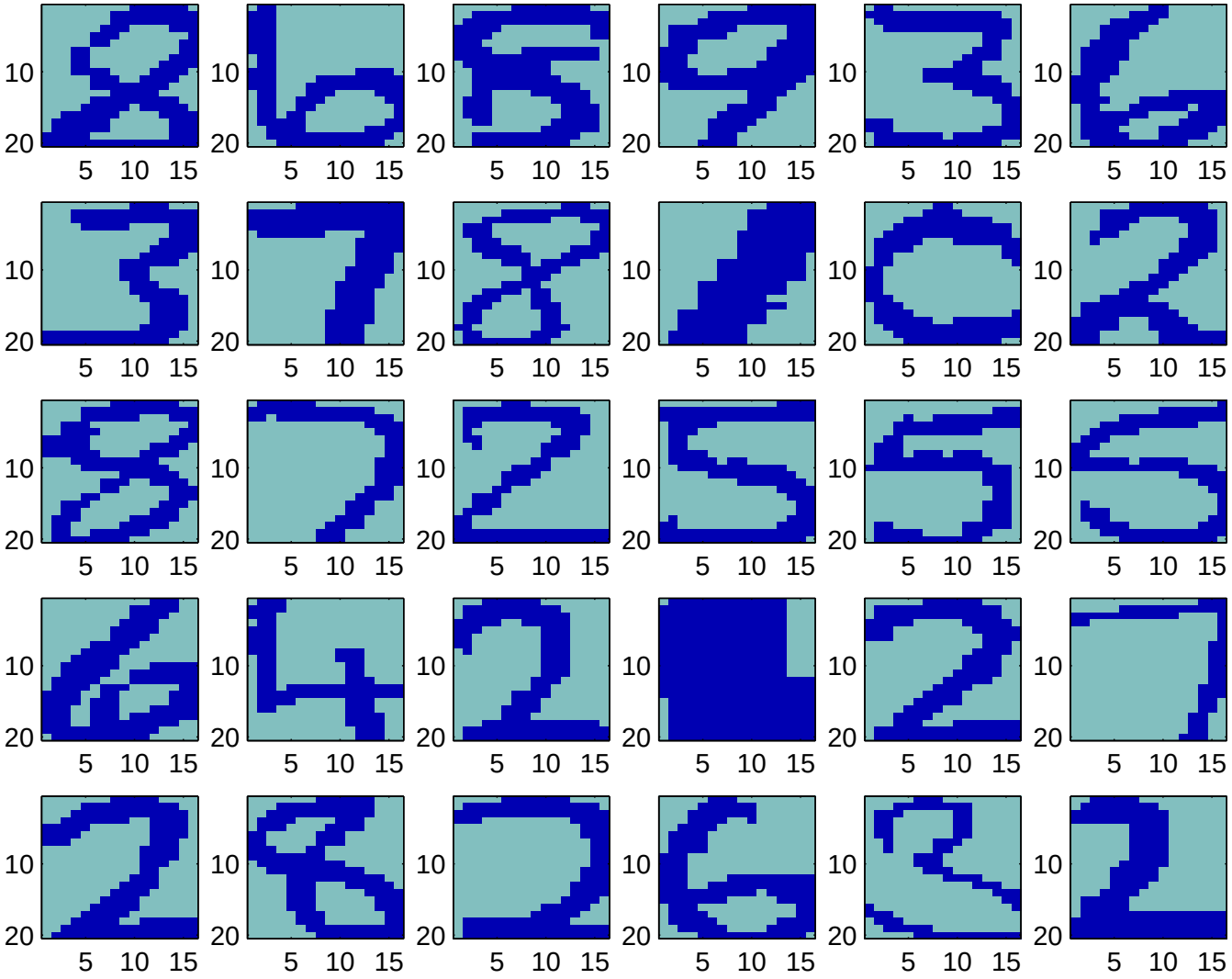
Example: The 'Swill-Roll' (2000 points in 3-D)



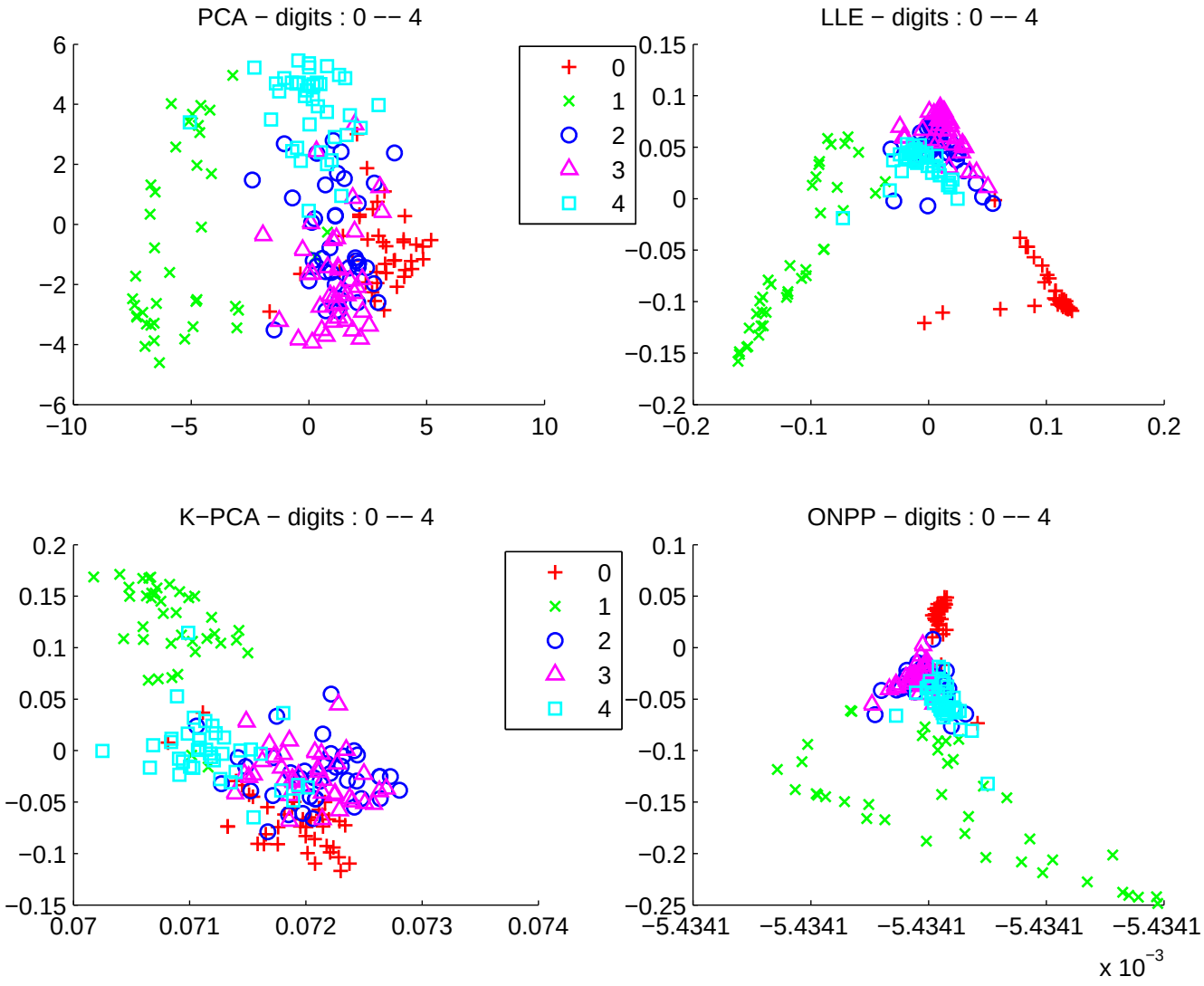
2-D ‘reductions’:



Example: Digit images (a random sample of 30)

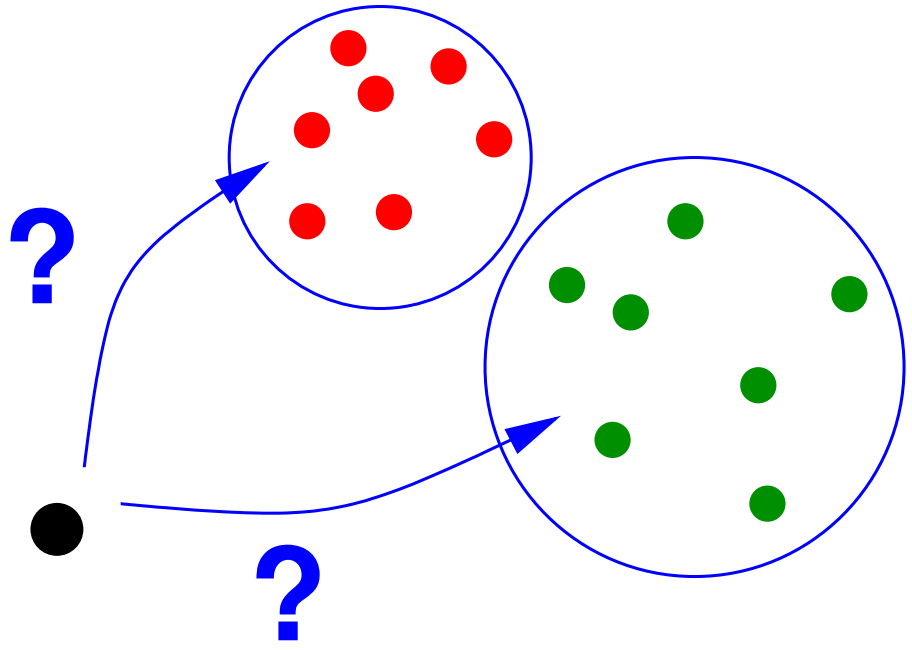


2-D 'reductions':



Supervised learning: classification

Problem: Given labels (say “A” and “B”) for each item of a given set, find a mechanism to classify an unlabelled item into either the “A” or the “B" class.



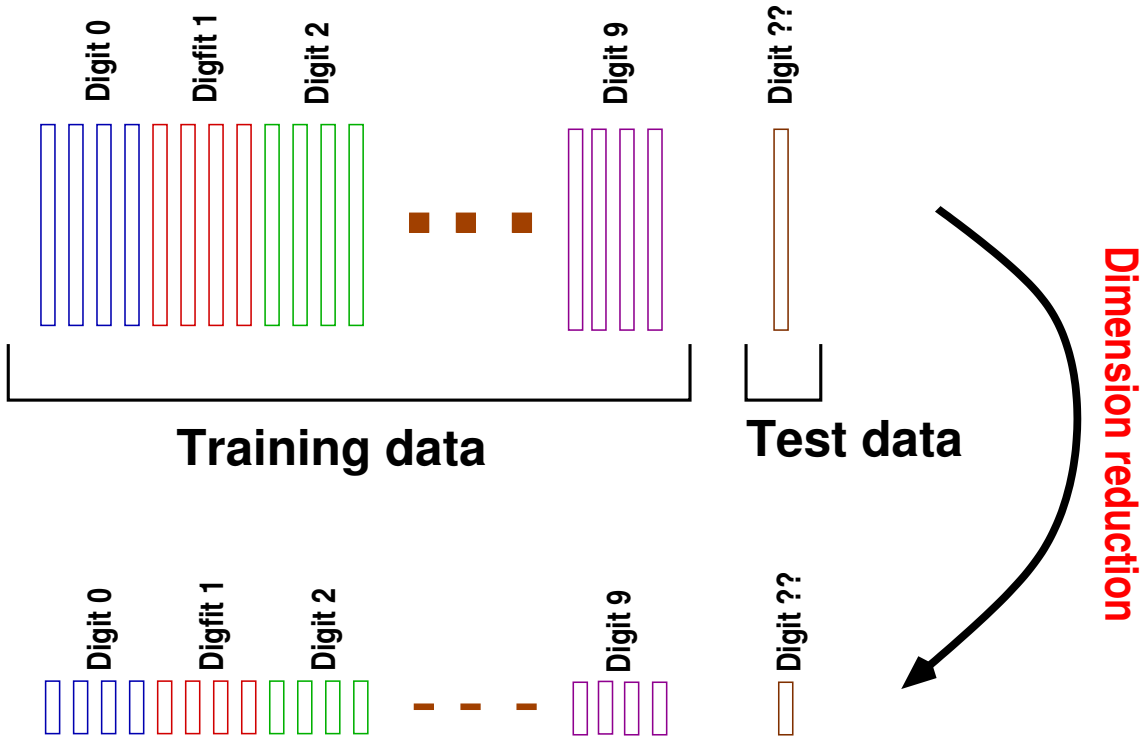
- Many applications.
- Example: distinguish SPAM and non-SPAM messages
- Can be extended to more than 2 classes.

Supervised learning: classification

- Best illustration: written digits recognition example

Given: a set of labeled samples (training set), and an (unlabeled) test image.

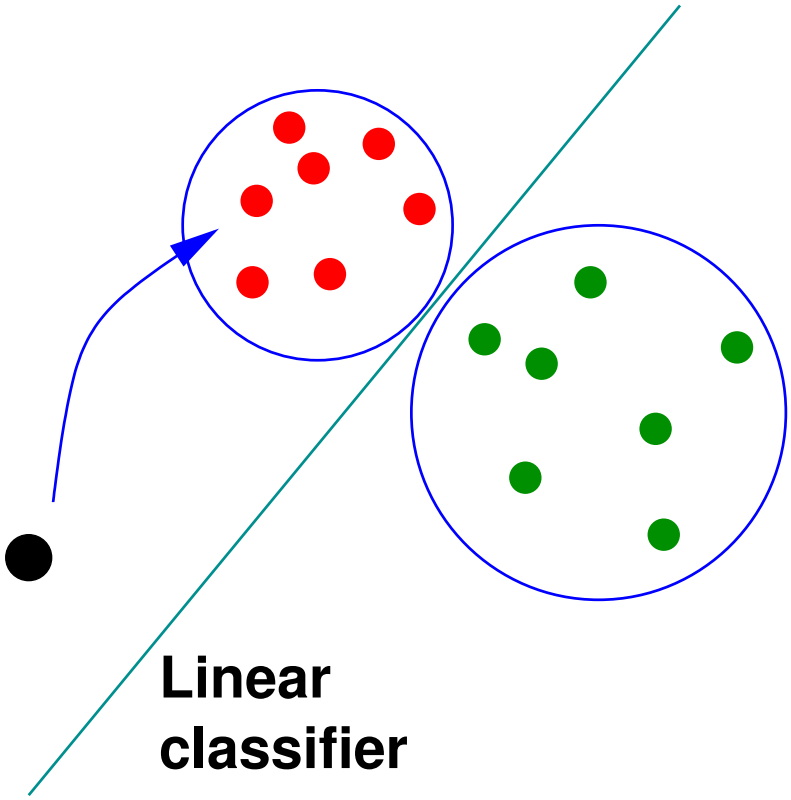
Problem: find label of test image



- Roughly speaking: we seek dimension reduction so that recognition is 'more effective' in low-dim. space

Supervised learning: Linear classification

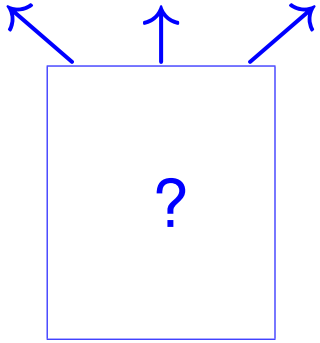
Linear classifiers: Find a hyperplane which best separates the data in classes A and B.



➤ Note: The world in non-linear. Often this is combined with Kernels – amounts to changing the inner product

Face Recognition – background

Problem: We are given a database of images: [arrays of pixel values]. And a test (new) image.



Question: Does this new image correspond to one of those in the database?

➤ Techniques used in words:

"Build a (linear) projector that does well on some training data. Use that same projector to predict class of new item"

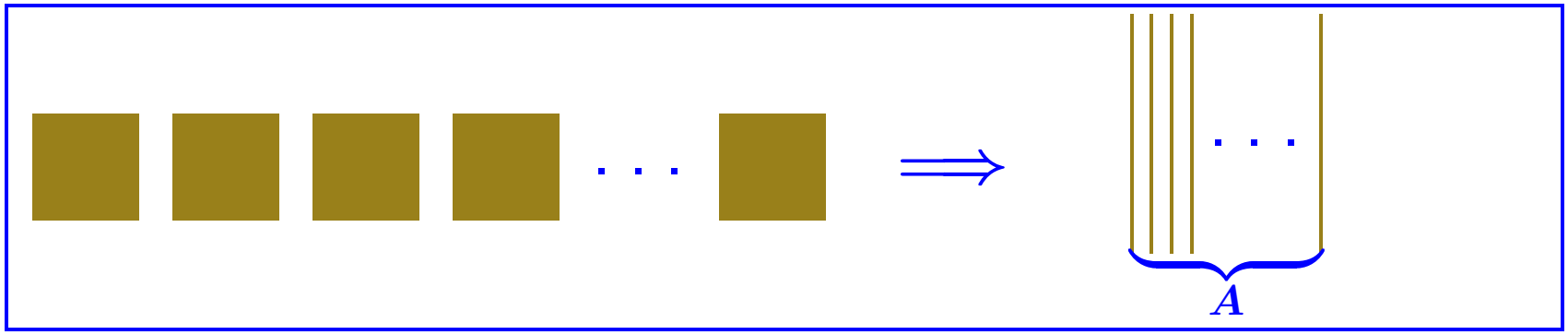
- Some methods use a graph - e.g., neighborhood graph
- Some methods use kernels (change inner products).

Example: Eigenfaces [Turk-Pentland, '91]

➤ Idea identical with the one we saw for digits:

– Consider each picture as a (1-D) column of all pixels

– Put together into an array A of size $\#\_pixels \times \#\_images$.

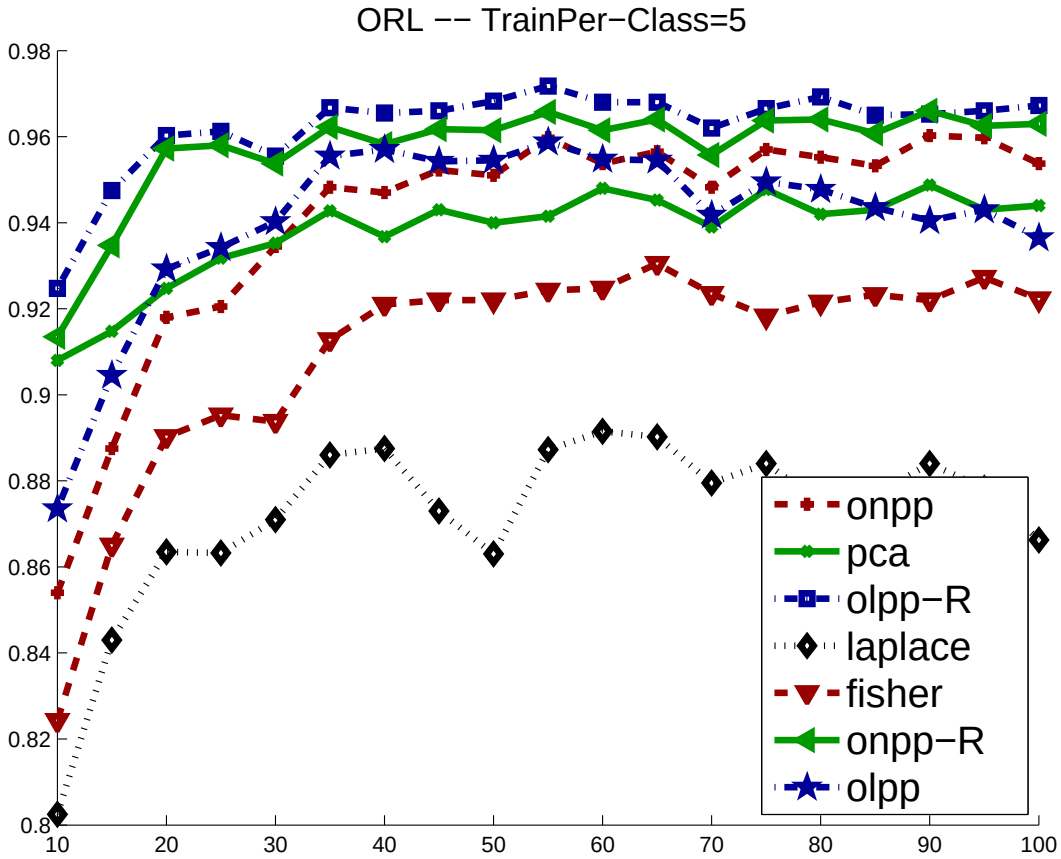


– Do an SVD of A and perform comparison with any test image in low-dim. space

– Similar to LSI in spirit – but data is not sparse.

Graph-based methods in a supervised setting

Test: ORL 40 subjects, 10 sample images each – sample shown earlier # of pixels : 112×92; TOT. # images : 400



ESTIMATING MATRIX RANKS

What dimension to use?

- Important question – but a hard one.
- Often, dimension k is selected in an ad-hoc way.
- k = intrinsic rank of data.
- Can we estimate it?

Two scenarios:

1. We know the magnitude of the noise, say τ. Then, ignore any singular value below τ and count the others.

2. We have no idea on the magnitude of noise. Determine a good threshold τ to use and count singular values $> \tau$.

Use of Density of States [Lin-Lin, Chao Yang, YS]

- Formally, the Density Of States (DOS) of a matrix A is

$$\phi(t) = \frac{1}{n}\sum_{j=1}^{n} \delta(t - \lambda_j),$$

where

- δ is the Dirac δ-function or Dirac distribution
- $\lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$ are the eigenvalues of A

- Term used by mathematicians: Spectral Density
- Note: number of eigenvalues in an interval $[a, b]$ is

$$\mu_{[a,b]} = \int_a^b \sum_j \delta(t - \lambda_j)\, dt \equiv \int_a^b n\phi(t) dt\,.$$

- $\phi(t)$ == a probability distribution function == probability of finding eigenvalues of A in a given infinitesimal interval near t.
- In Solid-State physics, λ_i's represent single-particle energy levels.
- So the DOS represents # of levels per unit energy.
- Many uses in physics

The Kernel Polynomial Method

- Used by Chemists to calculate the DOS – see Silver and Röder'94 , Wang '94, Drabold-Sankey'93, + others
- Basic idea: expand DOS into Chebyshev polynomials
- Use trace estimators [discovered independently] to get traces needed in calculations
- Assume change of variable done so eigenvalues lie in $[-1,\ 1]$.
- Include the weight function in the expansion so expand:

$$\hat{\phi}(t) = \sqrt{1-t^2}\phi(t) = \sqrt{1-t^2} \times \frac{1}{n}\sum_{j=1}^{n} \delta(t-\lambda_j).$$

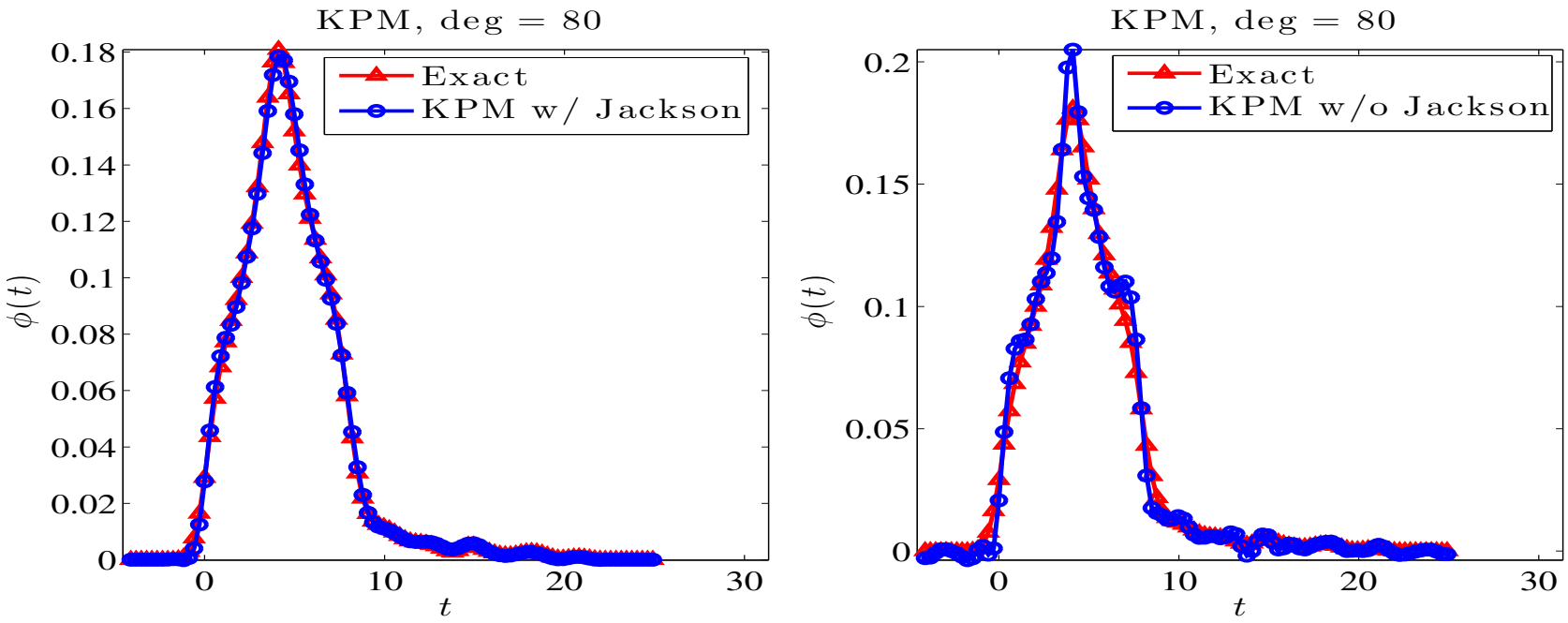
Then, (full) expansion is: $\hat{\phi}(t) = \sum_{k=0}^{\infty} \mu_k T_k(t)$.

- Expansion coefficients μ_k are formally defined by:

$$\begin{aligned}\mu_k &= \frac{2-\delta_{k0}}{\pi}\int_{-1}^{1}\frac{1}{\sqrt{1-t^2}}T_k(t)\hat{\phi}(t)dt \\ &= \frac{2-\delta_{k0}}{\pi}\int_{-1}^{1}\frac{1}{\sqrt{1-t^2}}T_k(t)\sqrt{1-t^2}\phi(t)dt \\ &= \frac{2-\delta_{k0}}{n\pi}\sum_{j=1}^{n}T_k(\lambda_j).\end{aligned}$$

- Note: $\sum T_k(\lambda_i) = Tr[T_k(A)]$
- Estimate this, e.g., via stochastic estimator
- Generate random vectors $v^{(1)}, v^{(2)}, \cdots, v^{(n_{\rm vec})}$ (normal distribution, zero mean)
- Some calculations similar to those of eigenvalue counts

An example with degree 80 polynomials



Left: Jackson damping; right: without Jackson damping.

Integrating to get eigenvalue counts

- As already mentioned

$$\mu_{[a,b]} = \int_a^b \sum_j \delta(t - \lambda_j)\, dt \equiv \int_a^b n\phi(t)dt$$

- If we use KPM to approximate $\phi(t) = \hat{\phi}(t)/\sqrt{1-t^2}$ then

$$\mu_{[a,b]} \approx \sum_{k=0}^{m} \mu_k \int_a^b \frac{T_k(t)}{\sqrt{1-t^2}} dt$$

- A little calculation shows that the result obtained in this way is identical with that of the eigenvalue count by Cheb expansion

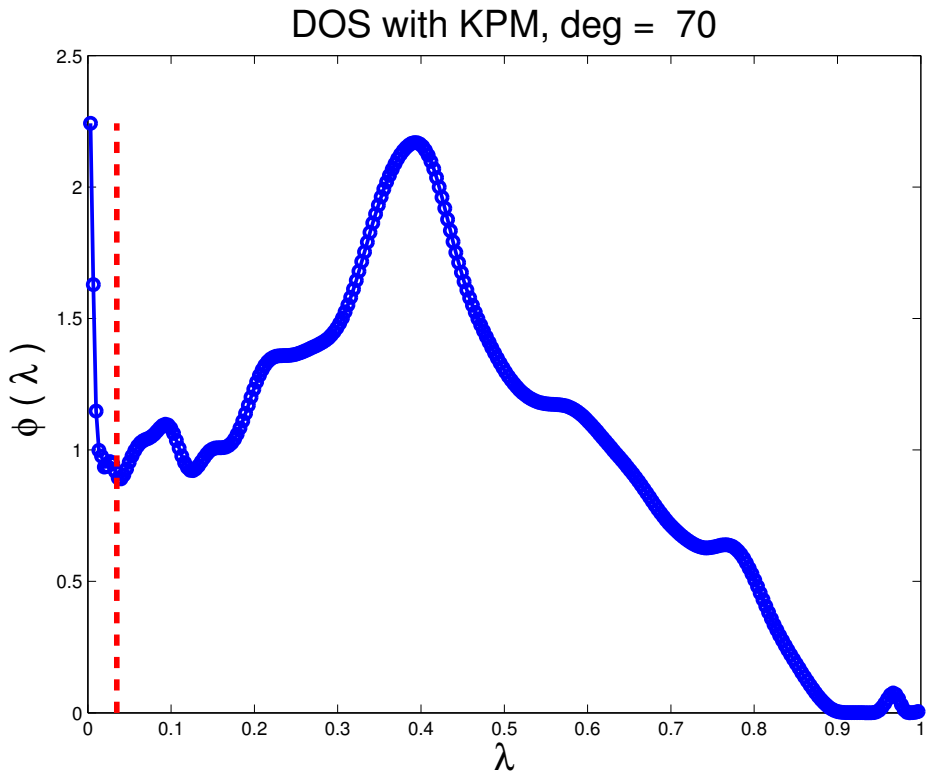
Application: Estimating a threshold for rank estimation

➤ When no other information is available, DOS can be used to find a good cut-off value θ to use to estimate ranks

➤ Common behavior: DOS decreases then increases (or stabilizes).

➤ If there is a significant gap use this to set-up θ.

➤ Often gives good guess for cut-off between 'noise' and 'real' sing. values



Updating the partial SVD

- In applications, data matrix $\boldsymbol{X}$ often updated

Challenge: Update the partial SVD as fast as possible [e.g. for 'online' applications..]

- Example: Information Retrieval (IR), can add documents, add terms, change weights, ..

- Methods based on projection techniques – developed in E. Vecharynski and YS'13. Details skipped

MATERIALS INFORMATICS

Data mining for materials: Materials Informatics

Definition: "The application of computational methodologies to processing and interpreting scientific and engineering data concerning materials" [Editors. 2006 MRS bulletin issue on materials informatics]

- Huge potential in exploiting two trends:

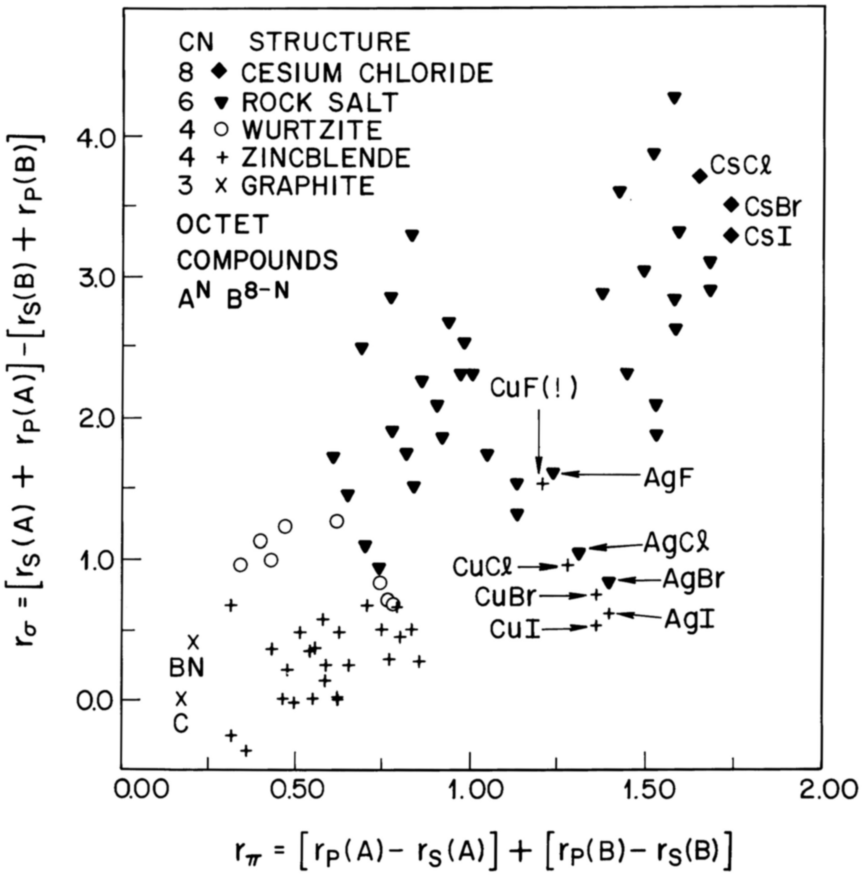
1 Improvements in efficiency and capabilities in computational methods for materials

2 Recent progress in data mining techniques

- Current practice: "One student, one alloy, one PhD" [see special MRS issue on materials informatics] $\rightarrow$ Slow ..

- Data Mining: can help speed-up process

Unsupervised learning



- 1970s: Unsupervised learning "by hand": Find coordinates that will cluster materials according to structure
- 2-D projection from physical knowledge
- 'Anomaly Detection': helped find that compound Cu F does not exist

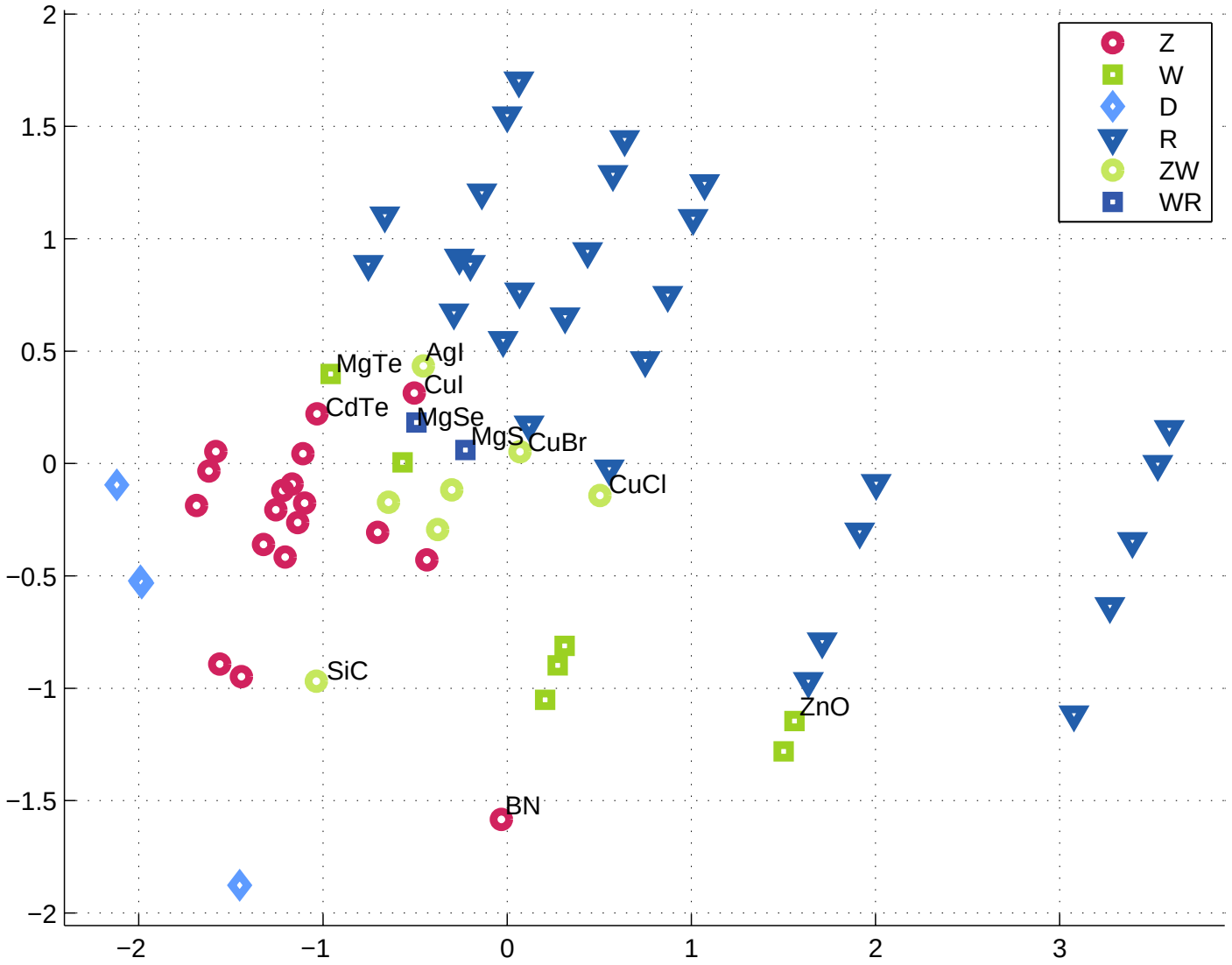
see: J. R. Chelikowsky, J. C. Phillips, Phys Rev. B 19 (1978).

Question: Can modern data mining achieve a similar diagrammatic separation of structures?

- Should use only information from the two constituent atoms
- Experiment: 67 binary 'octets'.
- Use PCA – exploit only data from 2 constituent atoms:

1. Number of valence electrons;
2. Ionization energies of the s-states of the ion core;
3. Ionization energies of the p-states of the ion core;
4. Radii for the s-states as determined from model potentials;
5. Radii for the p-states as determined from model potentials.

Result:


Z
W
D
R
ZW
WR
MgTe
AgI
CuI
CdTe
MgSe
MgS
CuBr
CuCl
SiC
ZnO
BN
2
1.5
1
0.5
0
−0.5
−1
−1.5
−2
−2
−1
0
1
2
3

Supervised learning: classification

Problem: classify an unknown binary compound into its crystal structure class

- 55 compounds, 6 crystal structure classes
- "leave-one-out" experiment

Case 1: Use features 1:5 for atom A and 2:5 for atom B. No scaling is applied.

Case 2: Features 2:5 from each atom + scale features 2 to 4 by square root of # valence electrons (feature 1)

Case 3: Features 1:5 for atom A and 2:5 for atom B. Scale features 2 and 3 by square root of # valence electrons.

Three methods tested

1. PCA classification. Project and do identification in space of reduced dimension (Euclidean distance in low-dim space).

2. KNN K-nearest neighbor classification –

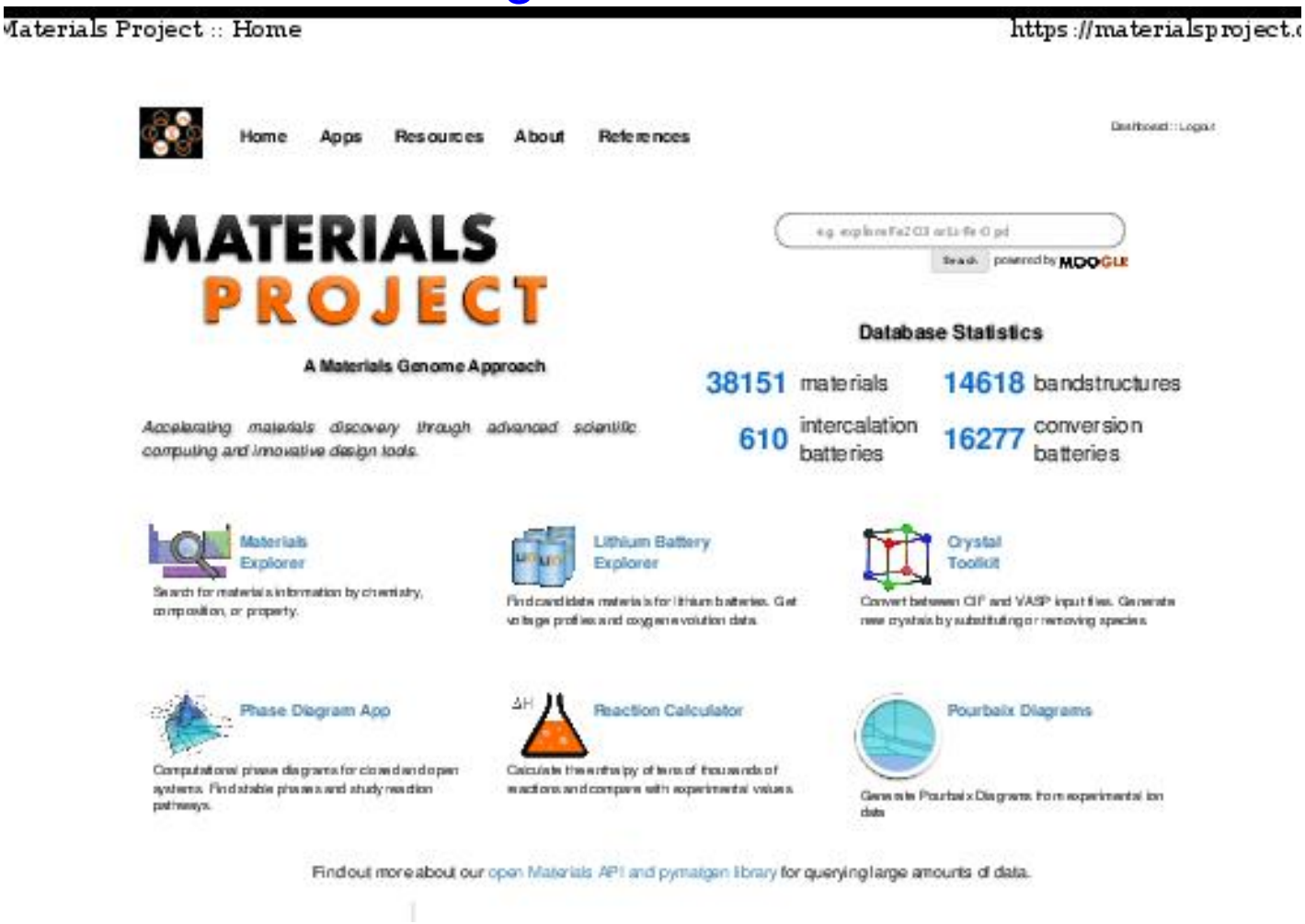
3. Orthogonal Neighborhood Preserving Projection (ONPP) - a graph based method - [see Kokiopoulou, YS, 2005]

Recognition rates for 3 different methods using different features

Case	KNN	ONPP	PCA
Case 1	0.909	0.945	0.945
Case 2	0.945	0.945	1.000
Case 3	0.945	0.945	0.982

Current work

- Some data is becoming available



Current work

➤ Huge recent increase of interest in the physics/chemstry community

1. Predicting enthalpy of formation for a certain class of compounds.. [current]

2. Application to Orthophosphates of lanthanides (LnPO4) [current – with Edoardo di Napoli TRWH]

3. Project: Predict Band-gaps

Conclusion

- Many, interesting new matrix problems in areas that involve the effective mining of data
- Among the most pressing issues is that of reducing computational cost - [SVD, SDP, ..., too costly]
- Many online resources available
- Huge potential in areas like materials science though inertia has to be overcome
- On the + side: materials genome project is starting to energize the field
- To a researcher in computational linear algebra : big tide of change on types or problems, algorithms, HPC, culture,..

- But change should be welcome

Two quotes:

> *"When one door closes, another opens; but we often look so long and so regretfully upon the closed door that we do not see the one which has opened for us.*
>
> – Alexander Graham Bell

> *"Life is like riding a bicycle. To keep your balance, you must keep moving"*
>
> – Albert Einsten