UNIVERSITY OF MINNESOTA TWIN CITIES

Numerical Linear Algebra for data-related applications

Yousef Saad

Department of Computer Science and Engineering

University of Minnesota

CEDYA/CMA 2021

Gijón, June 15, 2021

Introduction: a historical perspective

In 1953, George Forsythe published a paper titled:
"Solving linear systems can be interesting".

- Survey of the state of the art linear algebra at that time: direct & iterative methods, conditioning, preconditioning, the Conjugate Gradient method, acceleration methods, ...

➤ An amazing paper in which the author was urging researchers to start looking at solution methods for linear systems.

Introduction: a historical perspective

In 1953, George Forsythe published a paper titled:
"Solving linear systems can be interesting".

- Survey of the state of the art linear algebra at that time: direct & iterative methods, conditioning, preconditioning, the Conjugate Gradient method, acceleration methods, ...

➤ An amazing paper in which the author was urging researchers to start looking at solution methods for linear systems.

➤ Almost 7 decades later – we can similarly state that:

"Linear Algebra problems in Machine Learning can be interesting"

Focus of numerical linear algebra changed many times over the years

1940s–1950s: Major issue: the flutter problem in aerospace engineering $\rightarrow$ eigenvalue problem [cf. Olga Taussky Todd]

➤ Then came the discoveries of the LR and QR algorithms. The package Eispack followed a little later

1960s: Problems related to the power grid promoted what we would call today general sparse matrix techniques

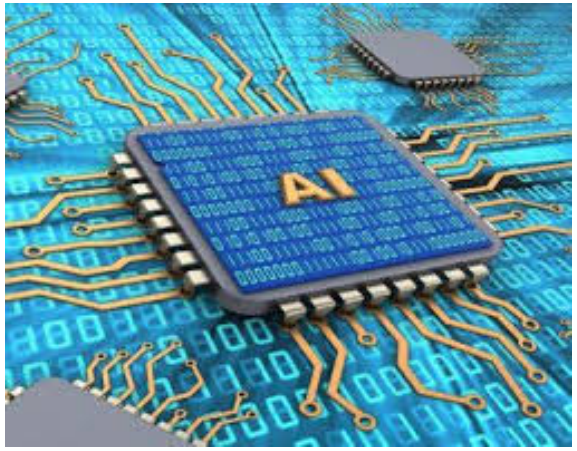
Early-late 1990: Thrust on parallel matrix computations.

Early 2000: Spur of interest in "financial computing"

Current: Machine learning, data-centered computing

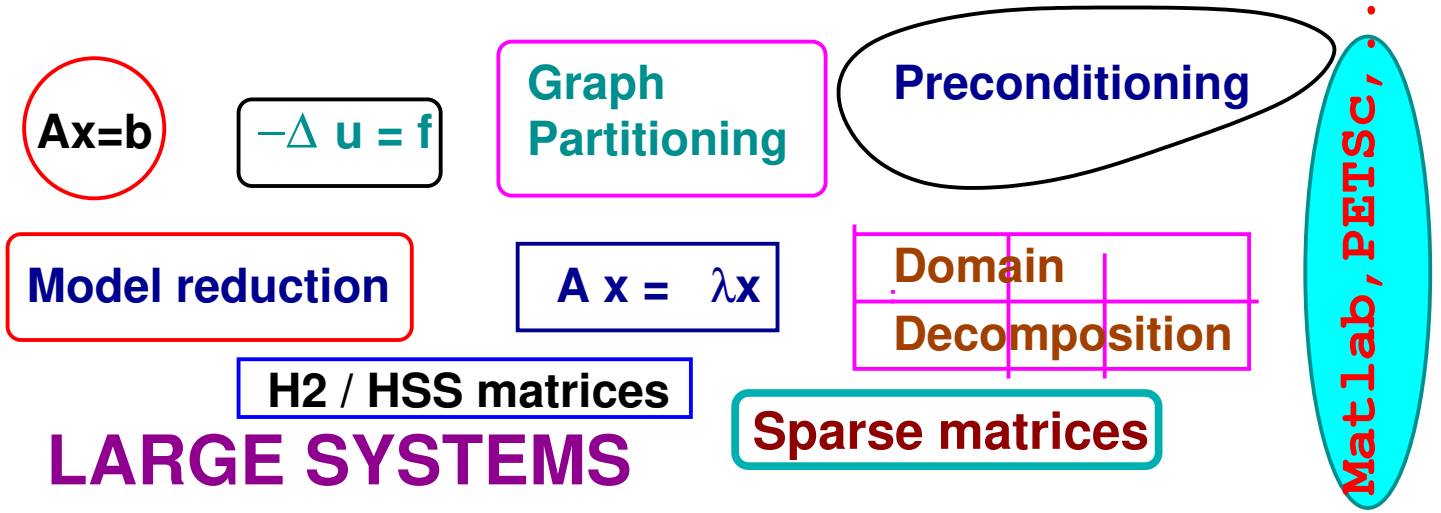
Solution of PDEs (e.g., Fluid Dynamics) and problems in mechanical eng. (e.g. structures) major force behind numerical linear algebra algorithms in the past few decades.

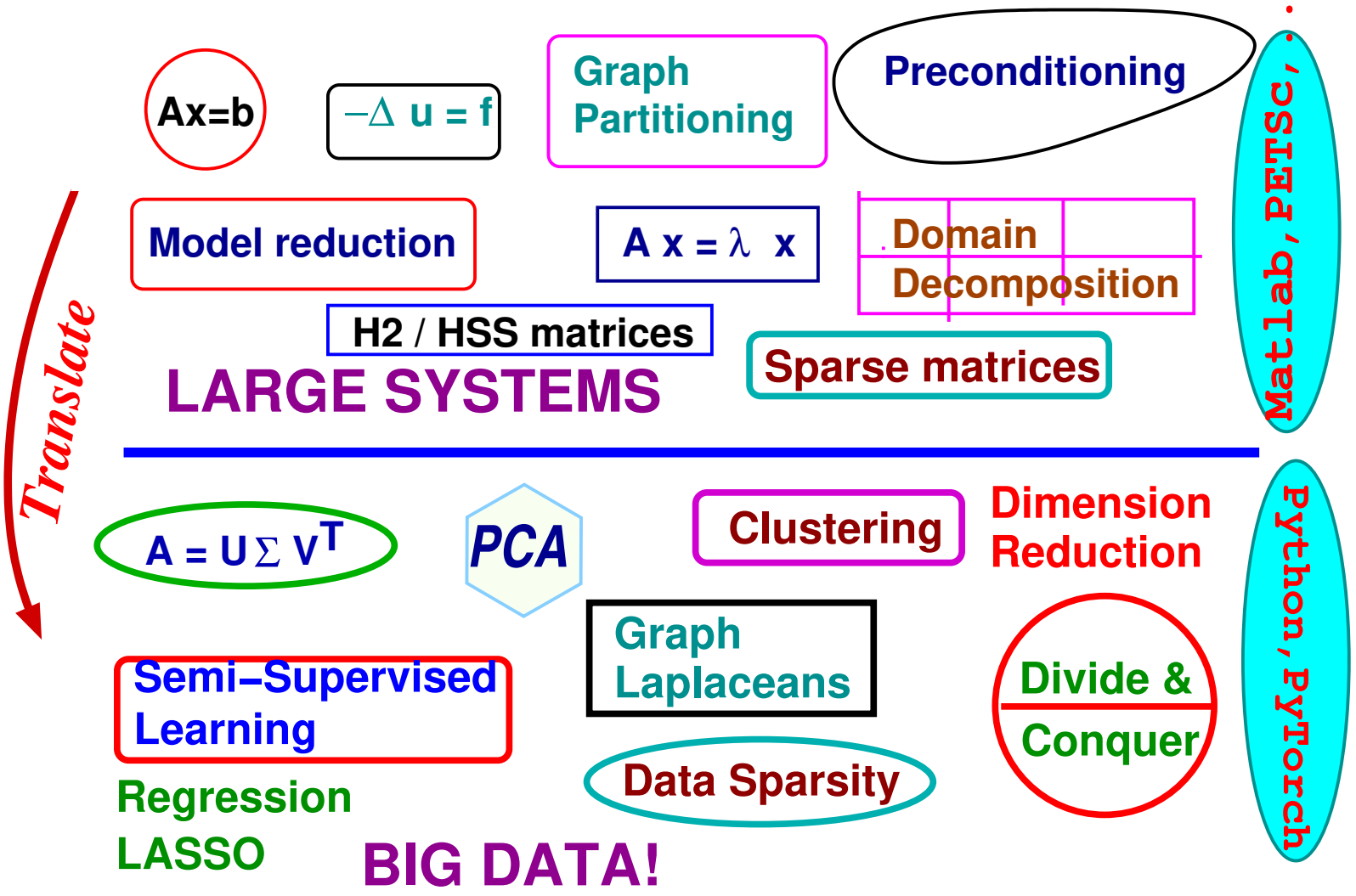
- Strong new forces are now reshaping the field
- Machine learning is appearing everywhere:



  - Design of materials, drugs, ...
  - Machine learning in geophysics
  - Self-driving cars, ..
  - .. Even: solving PDEs
  - ...

- Look at what you are doing under new lenses: DATA
- Big impact on the economy .. and on jobs:


Ax=b
$-\Delta u = f$
Graph Partitioning
Preconditioning
Matlab, PETSc, ...
Model reduction
A x = λx
Domain Decomposition
H2 / HSS matrices
Sparse matrices
LARGE SYSTEMS


Ax=b
$-\Delta u = f$
Graph Partitioning
Preconditioning
Matlab, PETSc, ...
Model reduction
$A x = \lambda x$
Domain Decomposition
H2 / HSS matrices
Sparse matrices
LARGE SYSTEMS
Translate
$A = U \Sigma V^T$
PCA
Clustering
Dimension Reduction
Python, PyTorch
Graph Laplaceans
Semi–Supervised Learning
Divide & Conquer
Data Sparsity
Regression
LASSO
BIG DATA!

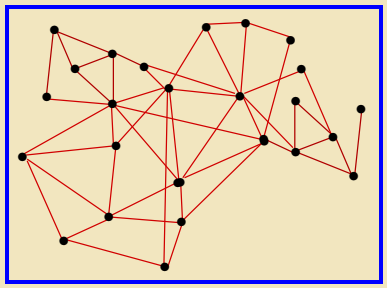
Plan:

1. A mini-tutorial: machine learning
2. Focus: Graph methods ...
3. ... and Graph coarsening.

INTRODUCTION & BACKGROUND: GRAPH LAPLACIANS

Graph Laplacians - Definition

- "Laplace-type" matrices associated with general undirected graphs –



$\longrightarrow L = \begin{bmatrix} \quad ? \quad \end{bmatrix}$

➤ Given a graph $G = (V, E)$ define

- A matrix W of weights w_{ij} for each edge with:

$$\boxed{w_{ij} \geq 0, \quad w_{ii} = 0, \quad \text{and} \quad w_{ij} = w_{ji} \; \forall (i,j)}$$

- The diagonal matrix $D = diag(d_i)$ with $d_i = \sum_j w_{ij}$

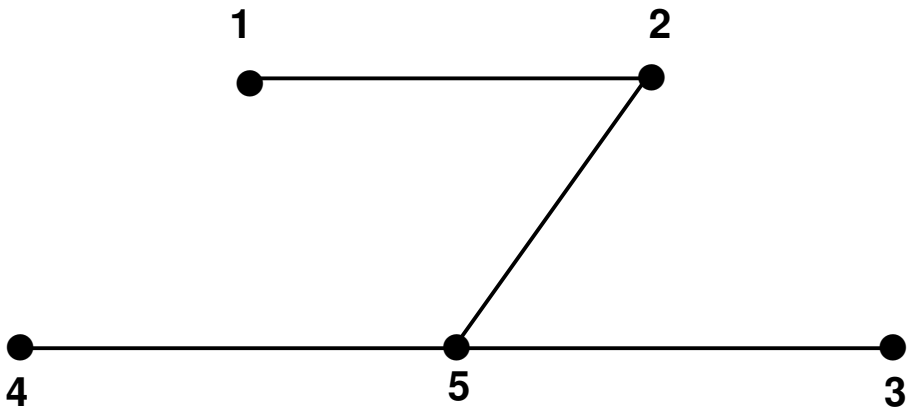
➤ Corresponding *graph Laplacian* of G is $\rightarrow$

$$L = D - W$$

➤ Simplest case:

$$w_{ij} = \begin{cases} 1 \text{ if } (i,j) \in E \ \& \ i \neq j \\ 0 \quad \text{else} \end{cases}$$

Example:



$$L = \begin{pmatrix} 1 & -1 & 0 & 0 & 0 \\ -1 & 2 & 0 & 0 & -1 \\ 0 & 0 & 1 & 0 & -1 \\ 0 & 0 & 0 & 1 & -1 \\ 0 & -1 & -1 & -1 & 3 \end{pmatrix}$$

Basic results on graph Laplacians

Proposition:

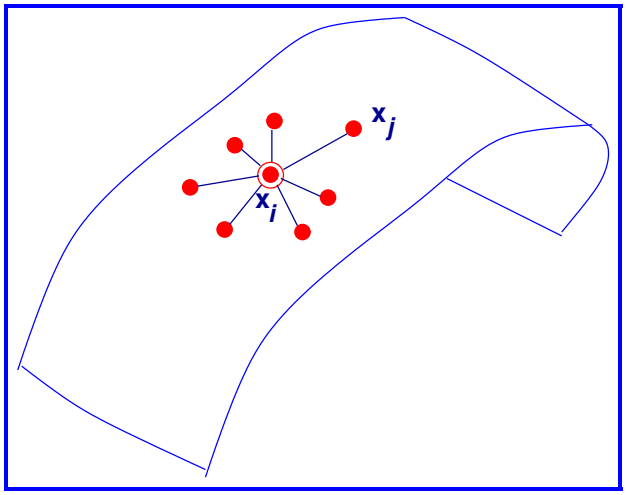
1. L is symmetric semi-positive definite.

2. L is singular with $\mathbb{1}$ as a null vector. If G is connected, then $\mathbf{Null}(L) = \mathbf{span}\{\mathbb{1}\}$

3. If G has $k > 1$ connected components $G_1, G_2, \cdots, G_k$, then the nullity of L is k and $\mathbf{Null}(L)$ is spanned by the vectors $z^{(j)}$, $j = 1, \cdots, k$ defined by:

$$(z^{(j)})_i = \begin{cases} 1 \text{ if } i \in G_j \\ 0 \text{ if not.} \end{cases}$$

A few properties of graph Laplacians



Strong relation between $x^T L x$ and local distances between entries of x

➤ Let L = a graph Laplacian. Then:

Property 1: for any $x \in \mathbb{R}^n$:

$$x^\top L x = \sum_{j>i} w_{ij} |x_i - x_j|^2$$

Property 2: (Generalization) for any $Y \in \mathbb{R}^{n \times d}$:

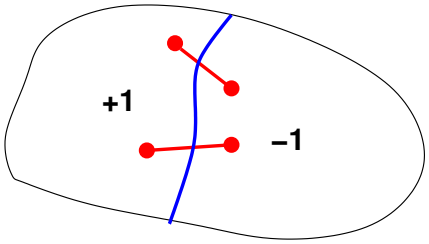
$$\text{Tr}\,[Y^\top L Y] = \sum_{j>i} w_{ij} \|y_{i,:} - y_{j,:}\|^2$$

➤ Note: $y_{j,:}$ = j-th row of Y. Each row can represent a data sample.

Property 3: (Graph partitioning) Consider situation when $w_{ij} \in \{0, 1\}$. If x is a vector of signs (± 1) then

$$x^\top L x = 4 \times \text{('number of edge cuts')}$$

- Edge-cut $\equiv$ pair (i, j) with $x_i \neq x_j$
- Can be used to partition graphs....



➤ Minimize (Lx, x) s.t. $x \in \{-1, 1\}^n$ and $\mathbb{1}^T x = 0$. $\rightarrow$ Hard

$$\min_{x \in \{-1,1\}^n;\ \mathbb{1}^T x = 0} \frac{(Lx, x)}{(x, x)}$$

➤ Instead solve a relaxed form of problem. Solution = u_2 2nd smallest eigenvector of L (Fiedler vector)

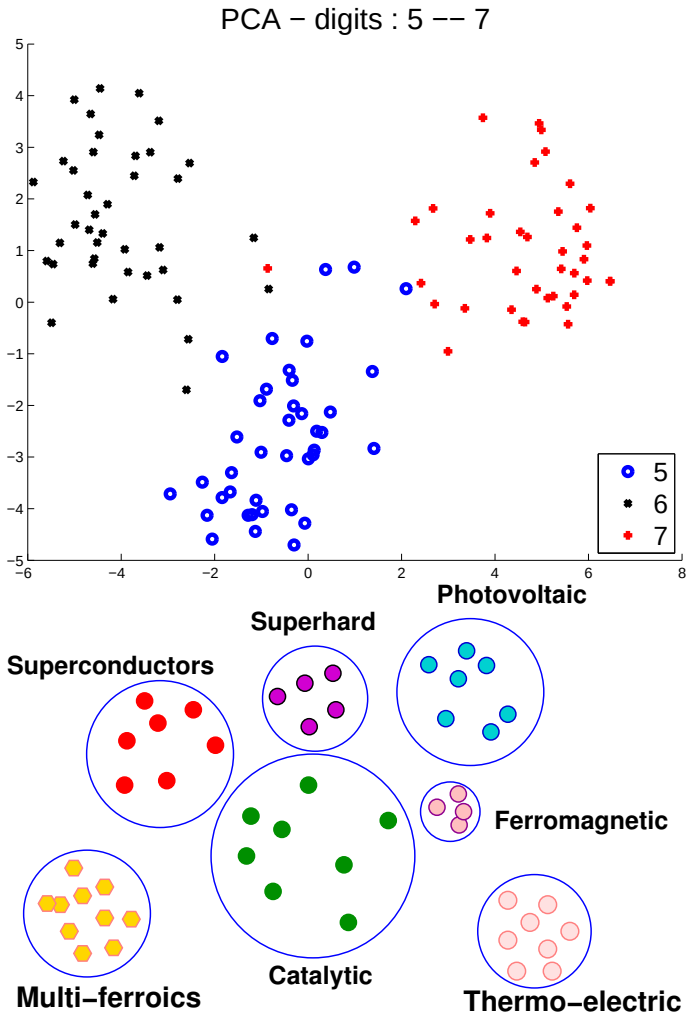
$$\min_{x \in \mathbb{R}^n;\ \mathbb{1}^T x = 0} \frac{(Lx, x)}{(x, x)}$$

UNSUPERVISED LEARNING & CLUSTERING

Unsupervised learning

Data is not labeled

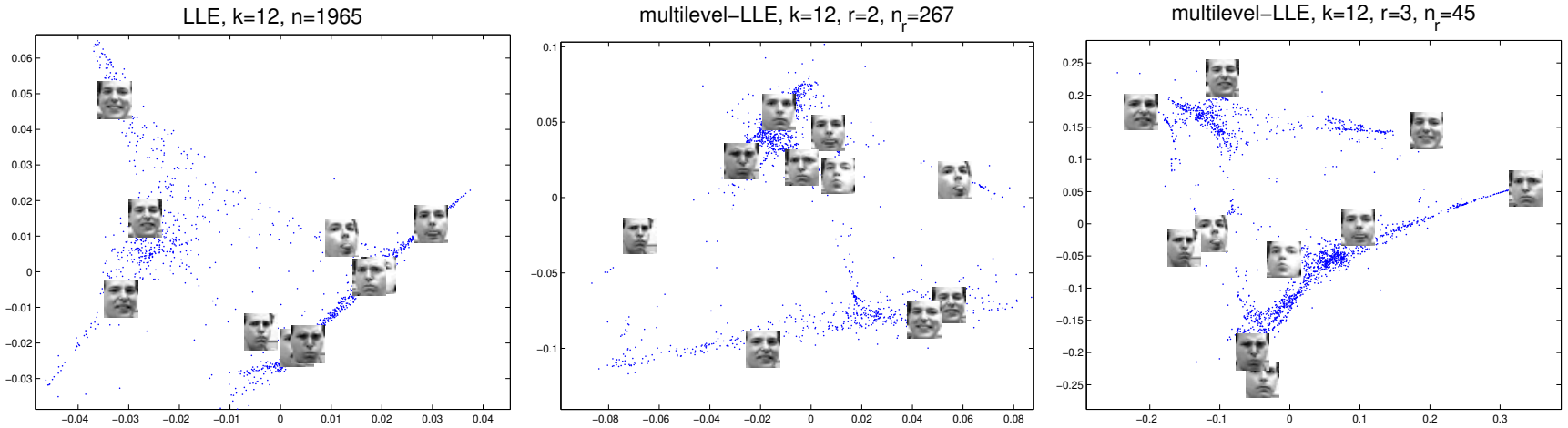
- Example of digits: perform a 2-D projection. Images of same digit tend to cluster (more or less)
- Such 2-D representations are popular for visualization
- Problem: find natural clusters in data, e.g., in materials



"Manifold Learning" Example: projection of face images

- Frey Dataset: 1,965 images of an individual – different expressions. Each image: 20×28 grey-scale pixels

Various projections [see H-R Fang, S. Sakellaridi, YS '10]

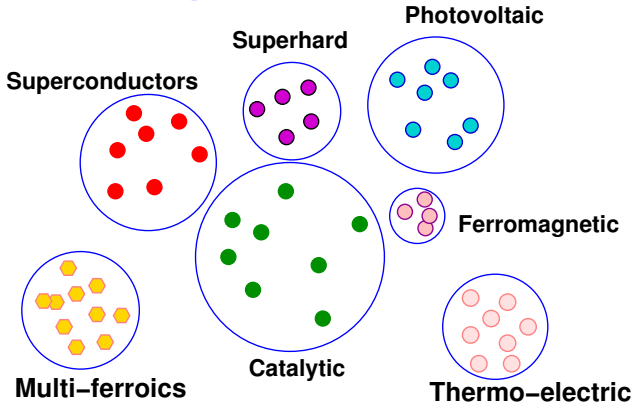


2D mappings of `Frey Face` *database using LLE and multilevel-LLE.*

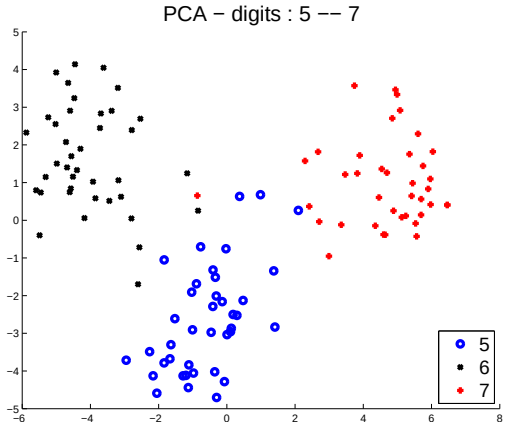
Clustering

➤ Problem: we are given n data items: $x_1, x_2, \cdots, x_n$. Would like to *'cluster'* them, i.e., group them so that each group or cluster contains items that are similar in some sense.

➤ Example: materials



➤ Example: Digits

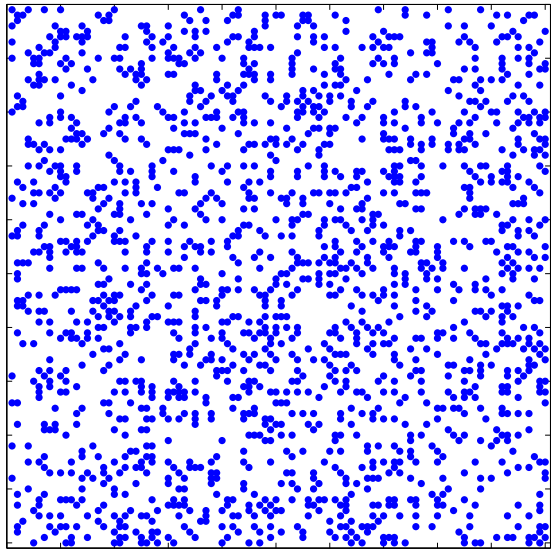


➤ Refer to each group as a 'cluster' or a 'class'

➤ 'Unsupervised learning' : Methods do not exploit labeled data

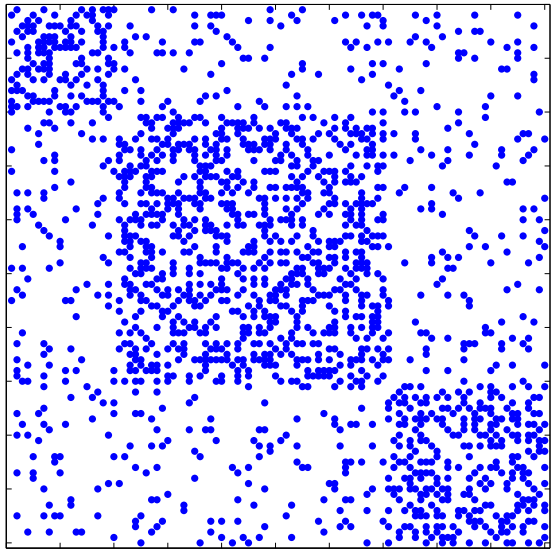
Example: Community Detection

- Communities modeled by an 'affinity' graph [e.g., 'user A sends frequent e-mails to user B']
- Adjacency Graph represented by a sparse matrix



← Original matrix

Goal: Find ordering so blocks are as dense as possible →



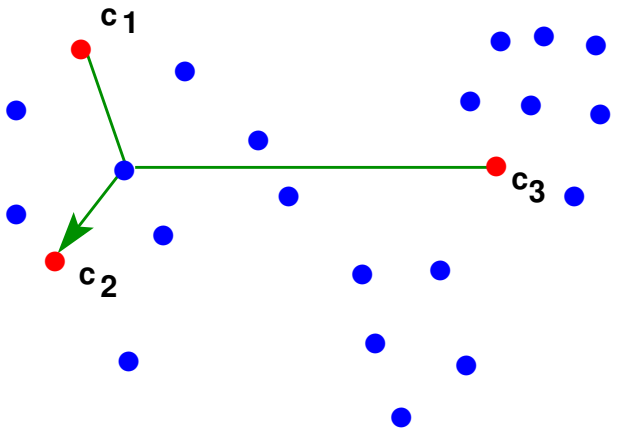
- Use 'blocking' techniques for sparse matrices
- Advantage of this viewpoint: need not know # of clusters.

[data: `www-personal.umich.edu/~mejn/netdata/`]

A basic clustering method: K-means (Background)

➤ A basic algorithm that uses Euclidean distance

1. Select p initial centers: $c_1, c_2, \ldots, c_p$ for classes $1, 2, \cdots, p$
2. For each x_i do: determine *class* of x_i as $\text{argmin}_k \|x_i - c_k\|$
3. Redefine each c_k to be the centroid of class k
4. Repeat until convergence

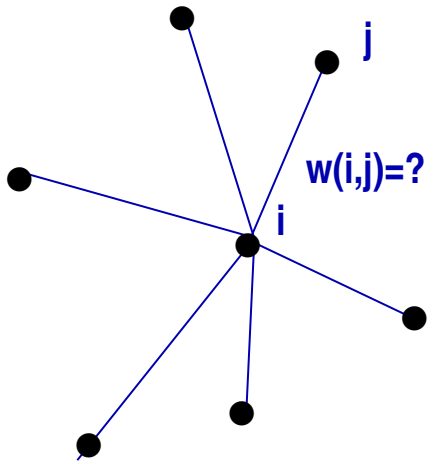


- Simple algorithm
- Works well but can be slow
- Performance depends on initialization

Spectral clustering: General approach

1. Given: Collection of data samples $\{x_1, x_2, \cdots, x_n\}$

2. Build a similarity graph between items



3. Compute (smallest) d eigenvectors of resulting graph Laplacian [this *'embeds'* graph to $\mathbb{R}^d$]

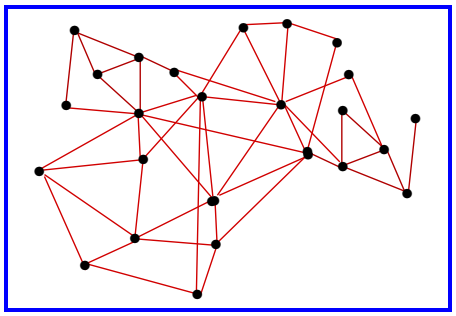
4. Use k-means on eigenvector (s) of Laplacean

GRAPH EMBEDDINGS

Graph embeddings

- In Similarity Graphs: we build a graph to represent data
- *Graph embedding*: We do the opposite, i.e., map a graph to vectors

Vertex embedding: map every vertex x_i to a vector $y_i \in \mathbb{R}^d$



$\longrightarrow$ Data: $Y = [y_1, y_2, \cdots, y_n]$ in $\mathbb{R}^d$

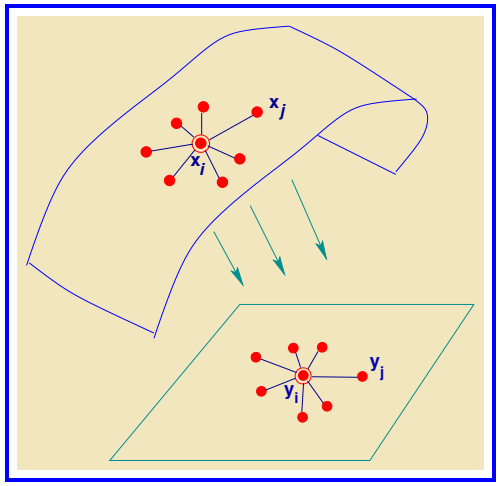
➤ Trivial use: visualize a graph ($d = 2$)

Graph embedding: map whole graph G to a vector $y_G \in \mathbb{R}^d$

➤ Many applications [clustering, finding missing link, semi-supervised learning, community detection, ...]

➤ Embeddings are central to Graph Neural Networks (GNNs)

- Graph built to captures similarities in data
- Goal of the embedding is to preserve these similarities.
- Done via the Graph (e.g., Laplacian)



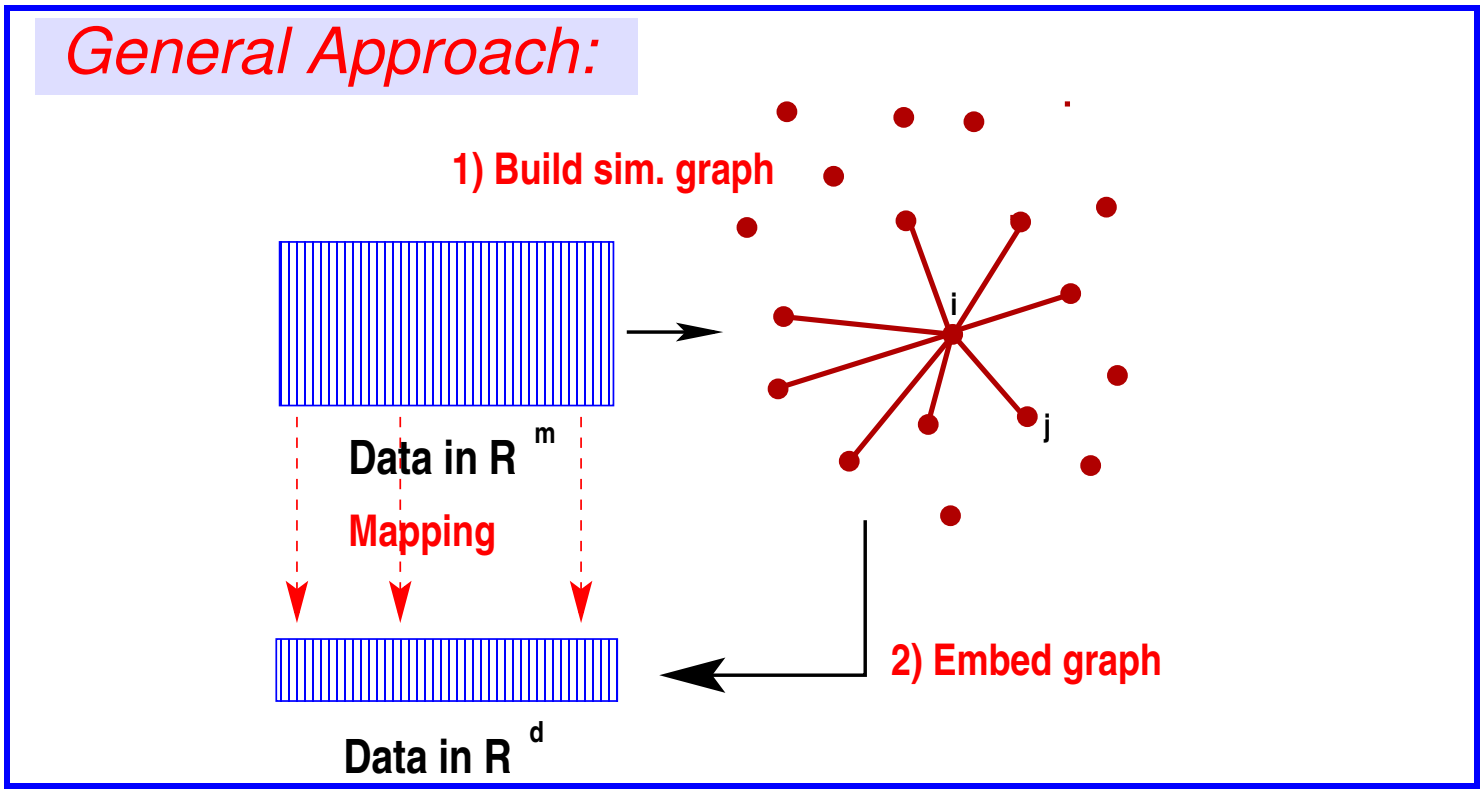
➤ Many methods do this. Examples:

Eigenmaps , *Isomap* , *LLE*

➤ Used in earlier illustration with Frey dataset

Graph-based dimension reduction

- A class of methods that exploit graphs to perform dimensionality reduction [eigenmaps, LLE, isomap, LLP, ..]

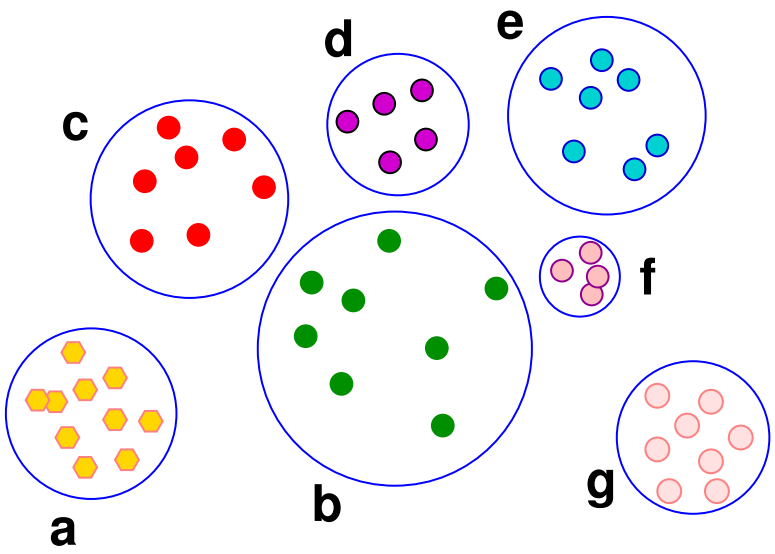


SUPERVISED LEARNING

Supervised learning

Now: *data is 'labeled'*

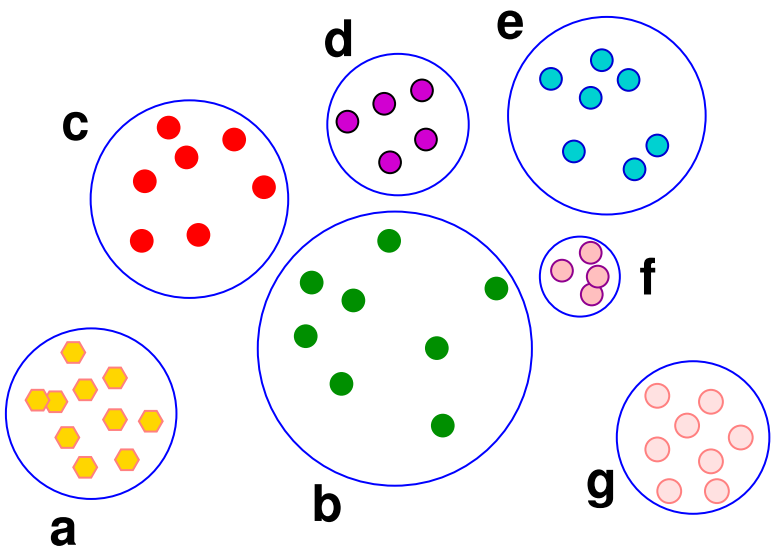
- Example: (health sciences) 'malignant'- 'non malignant'
- Example: (materials) 'photovoltaic', 'hard', 'conductor', ...
- Example: (Digit recognition) Digits '0', '1', ..., '9'

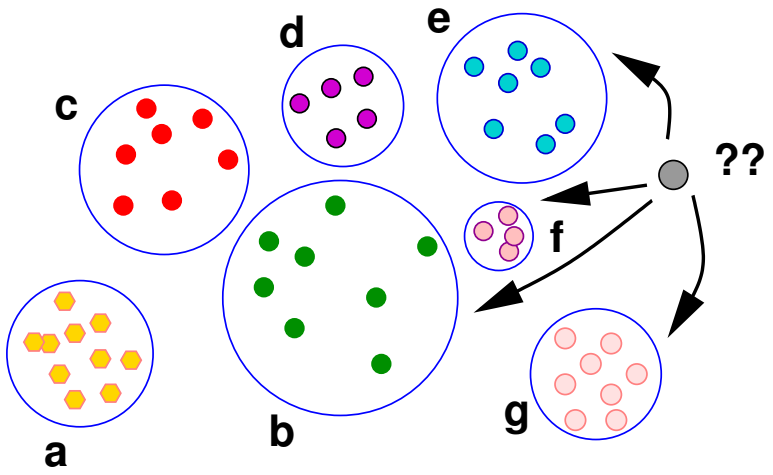


Supervised learning

We now have data that is ‘labeled’

- Example: (health sciences) ‘malignant’- ’non malignant’
- Example: (materials) ’photovoltaic’, ’hard’, ’conductor’, ...
- Example: (Digit recognition) Digits ’0’, ’1’, ..., ’9’



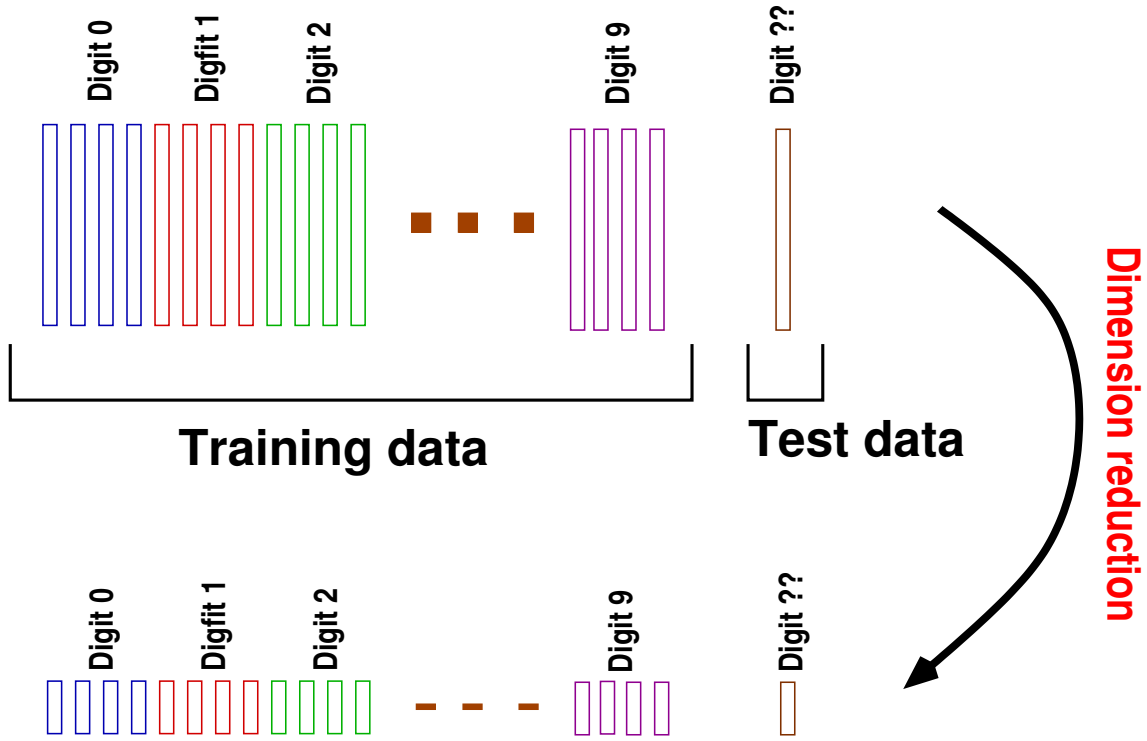


Supervised learning: classification

➤ Example: written digits recognition

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

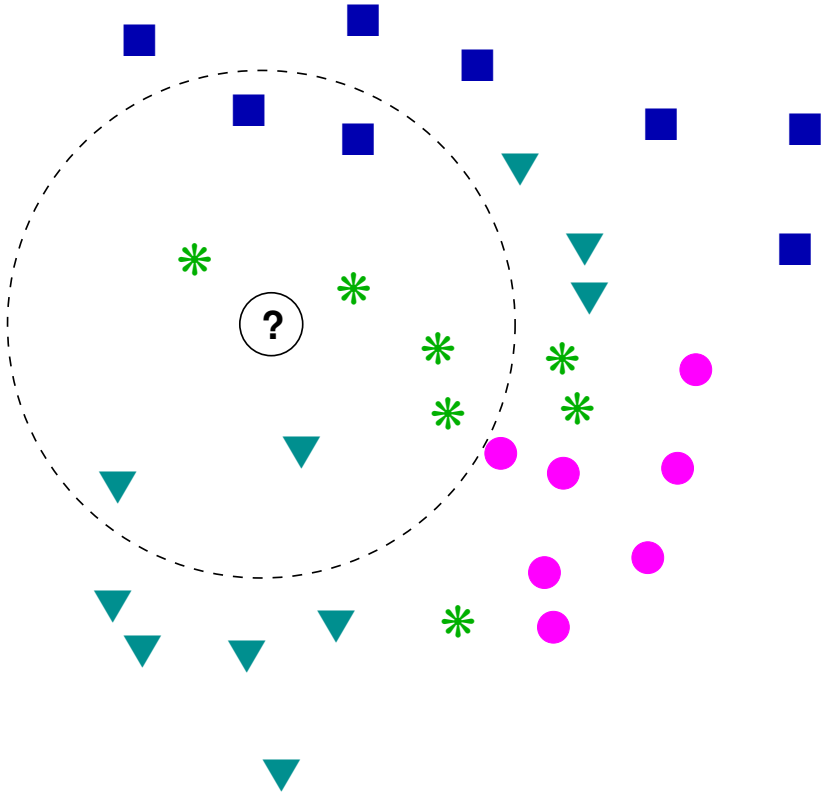
Problem: find label of test image



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

Basic method: K-nearest neighbors (KNN) classification

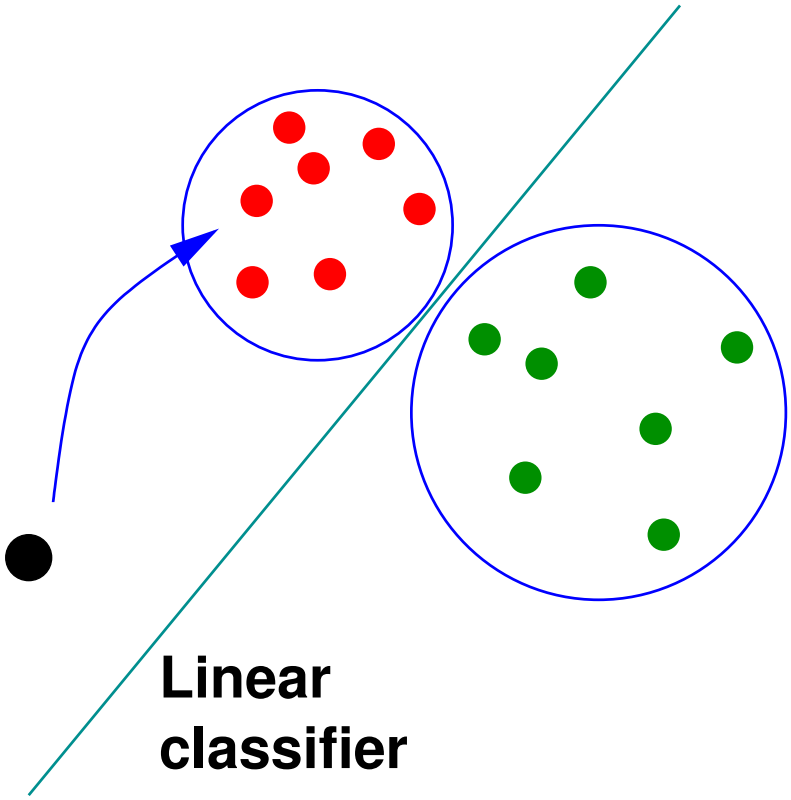
- Idea of a voting system: get distances between test sample and training samples
- Get the k nearest neighbors (here $k = 8$)
- Predominant class among these k items is assigned to the test sample ("$*$" here)



Supervised learning: Linear classification

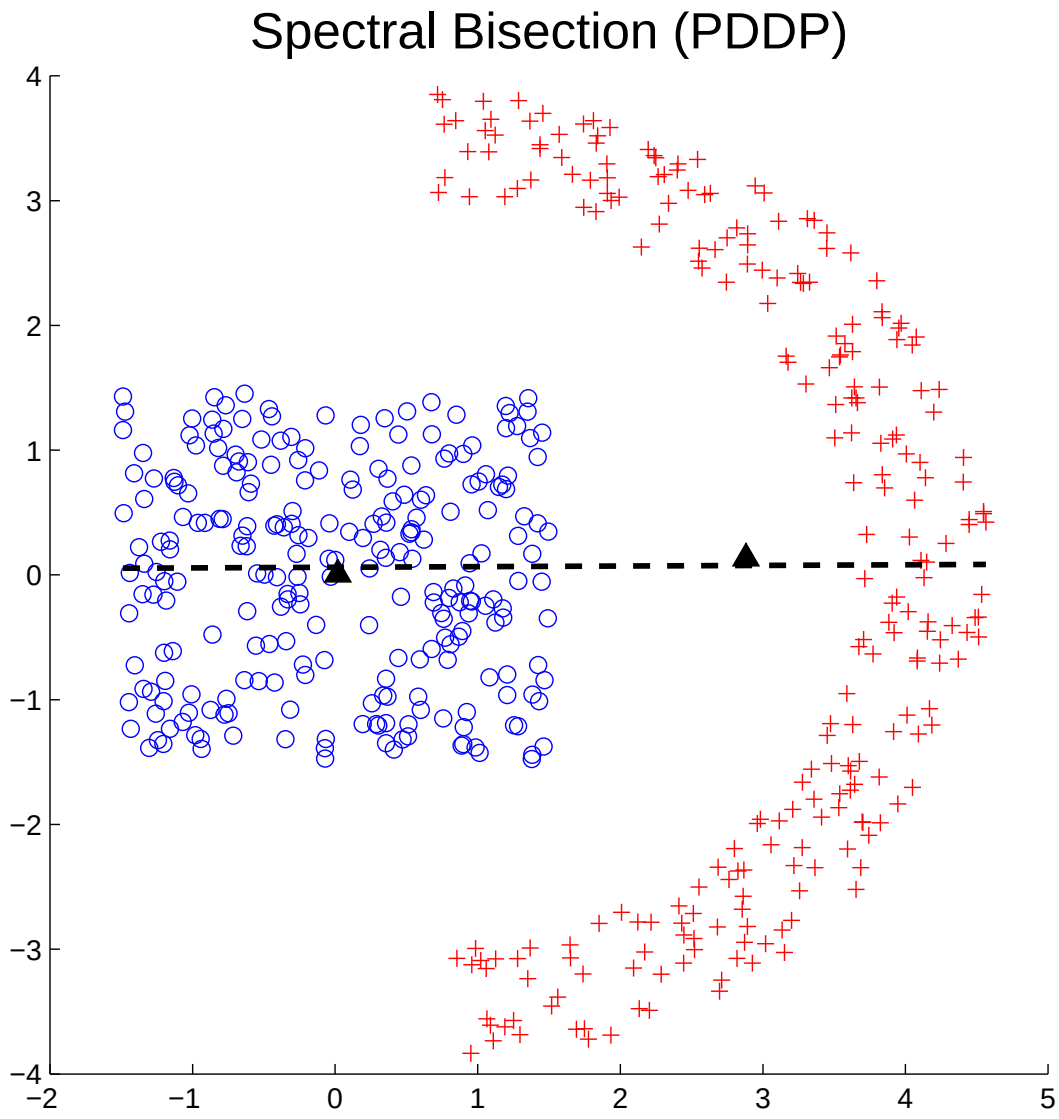
Linear classifiers: Find a hyperplane that best separates data in two classes. Examples:

- Fisher's Linear Discriminant Analysis (LDA)
- Support Vector Machines (SVM)

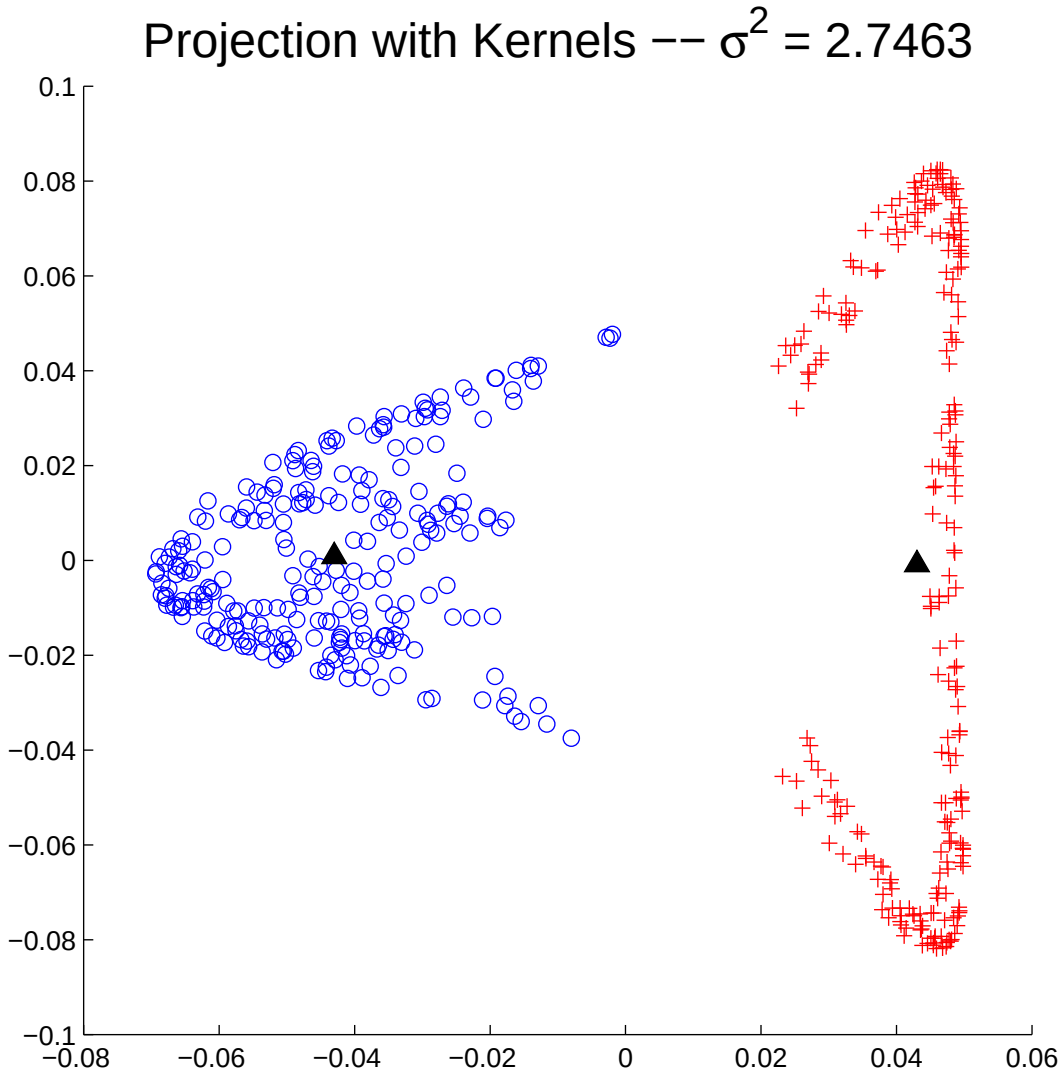


➤ Note: The world in non-linear. Often this is combined with Kernels – amounts to changing the inner product

A harder case:



- Use kernels to transform



Transformed data with a Gaussian Kernel

DEMO

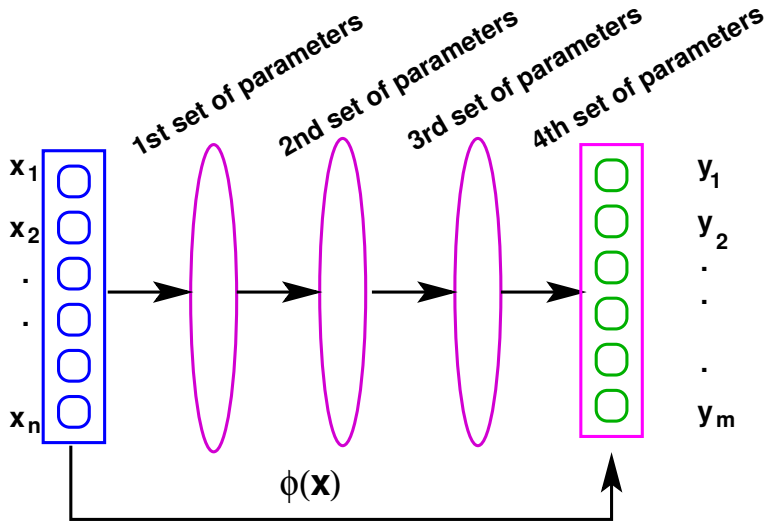
A few words on Deep Neural Networks (DNNs)

➤ Ideas of neural networks goes back to the 1960s - were popularized in early 1990s – then laid dormant until recently.

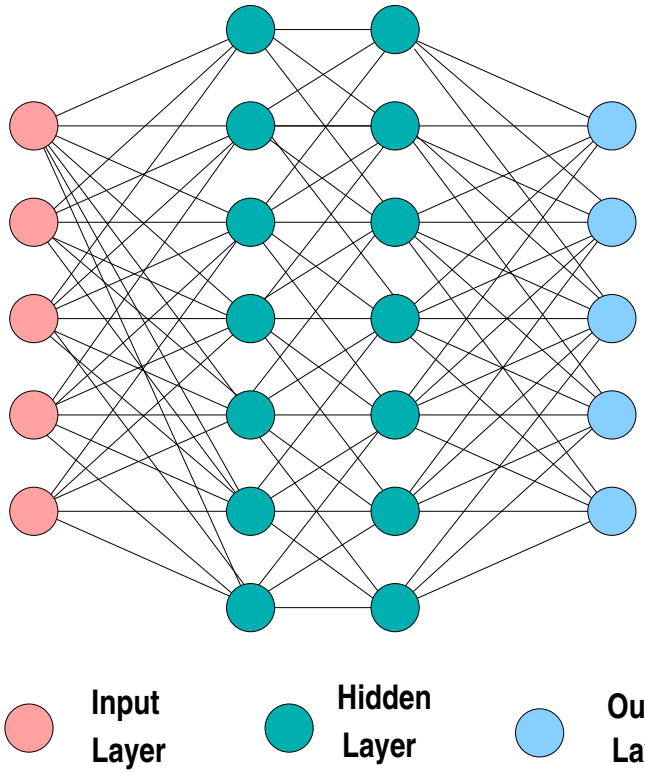
➤ Training a neural network amounts to approximating a function ϕ which is defined via sets ('layers') of parameters:

Problem:

Find sets of parameters such that $\phi(x) \approx y$



Input: x, **Output:** y
Set: $z_0 = x$
For $l = 1 :$ L+1 **Do:**
$z_l = \sigma(W_l{}^T z_{l-1} + b_l)$
End
Set: $y = \phi(x) := z_{L+1}$



- layer # 0 = input layer
- layer # $(L+1)$ = output layer

➤ Matrix W_l associated with layer l for $l = 1, 2, \cdots, L+1$

➤ Problem: Find ϕ (i.e., matrices W_l) s.t. $\phi(x) \approx y$

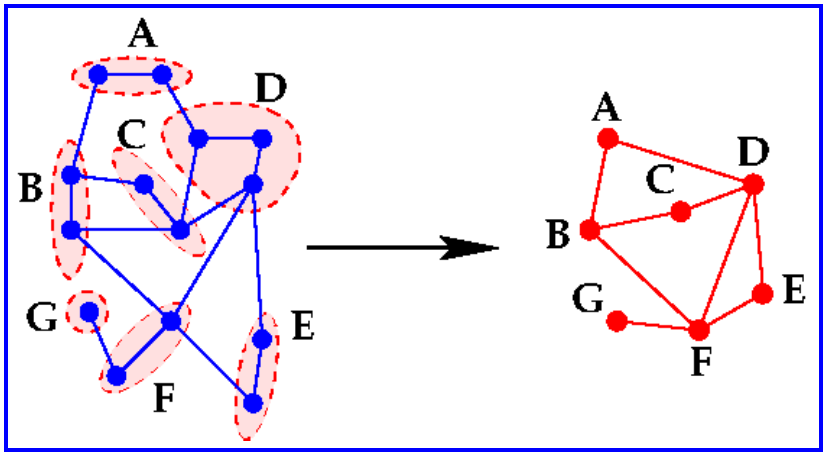
DNN (continued)

- Problem is not convex and it is highly over-parameterized
- Main method used: Stochastic gradient descent [basic]
- It all works like alchemy... but great results for certain applications
- Training is still quite expensive – GPUs can help
- *Very* active area of research

FOCUS: GRAPH COARSENING

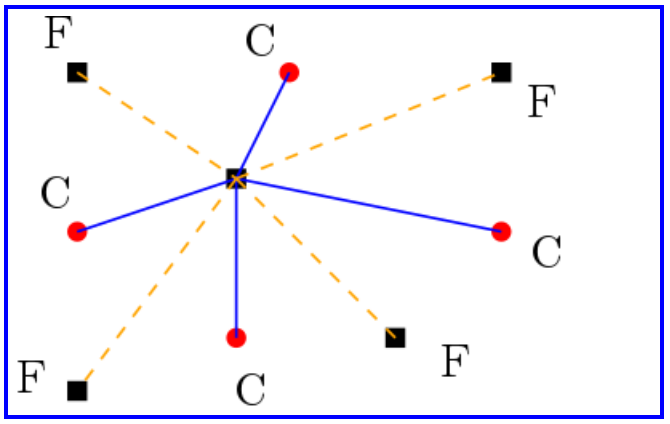
Graph Coarsening in scientific computing

- Goal : exploit coarse representation of problem



- Fewer nodes so: cheaper
- Can be used recursively

- Success story: *Multigrid, Algebraic Multigrid*
- *AMG:* Define coarse / fine nodes based on 'strength of coupling' $\rightarrow$



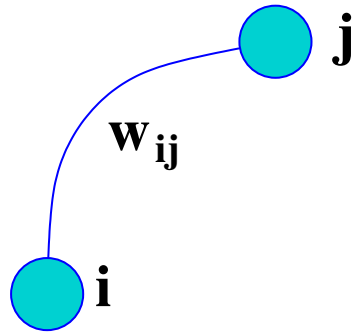
Example: Multilevel ILU [D. Osei-Kuffuor, R. Li, YS, '15]

Goal: Form of ILU preconditioning with improved robustness

➤ To define coarse nodes: traverse edges $(i,j) \in Nz(A)$ in decreasing order of the weights:

$$w_{ij} = \min \left\{ \frac{|a_{ij}|}{\delta_r(i)}, \frac{|a_{ij}|}{\delta_c(j)} \right\} \text{ where:}$$

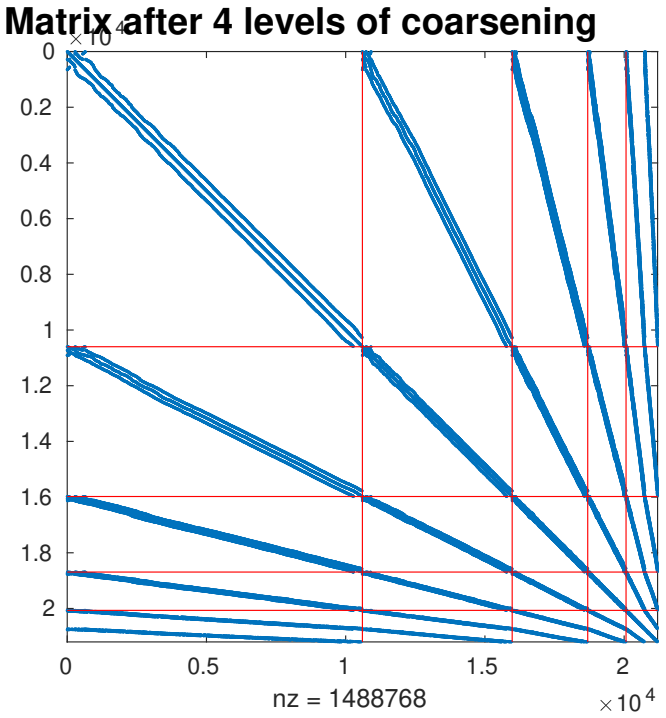
$$\delta_r(i) = \frac{\|A_{i,:}\|_1}{nz(A_{i,:})} \quad \text{and} \quad \delta_c(j) = \frac{\|A_{:,j}\|_1}{nz(A_{:,j})}$$

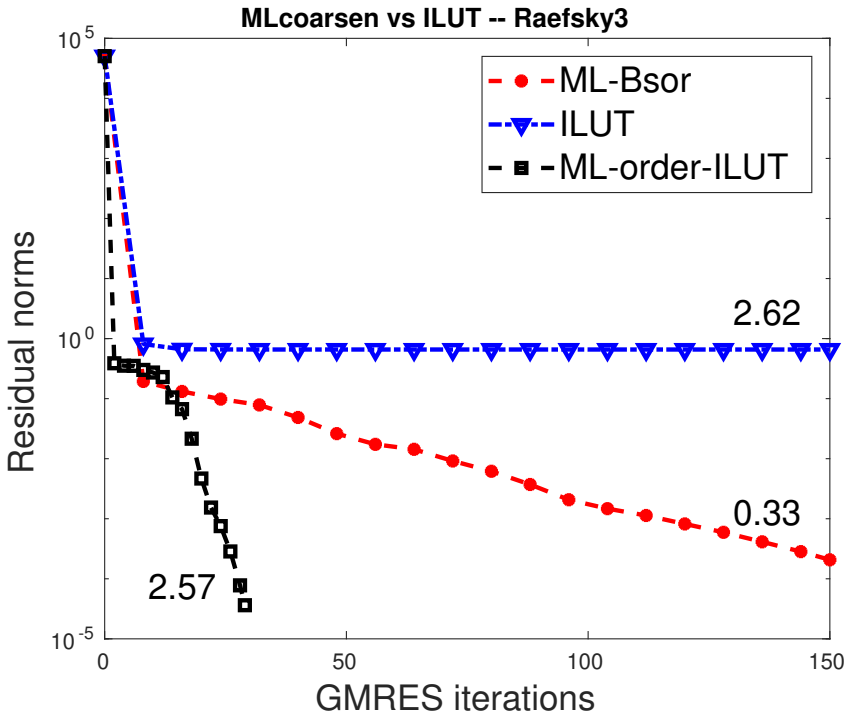


➤ Select i as 'coarse' if $\sigma_i > \sigma_j$ and j otherwise, where $\rightarrow$

$$\sigma_k = \frac{|a_{kk}|}{\delta_r(k)\delta_c(k)}$$

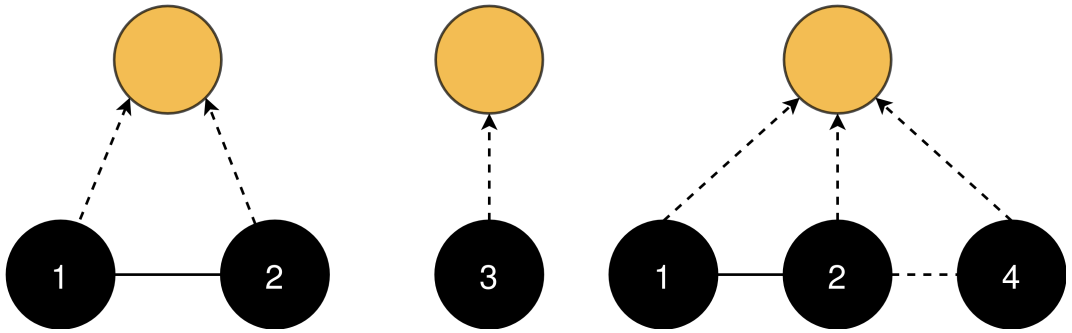
- (Matlab) Test with matrix *Raefsky3* [1]
- 4 levels of coarsening. Then reorder matrix and:
- Solve with ILUT- GMRES(50) or BSOR - GMRES(50)





[1] SparseSuite collection. $n = 21,200, nnz \approx 1,5M$, Turbulence model.

Coarsening approaches by matching: Pairwise aggregation



1. Visit edges (i, j) in decreasing value of their weight $w_{i,j}$
2. If both i and j have no parents yet (left), create a new coarse node ('new'). Set parents of i and j to be new.
3. When loop is completed deal with unassigned nodes: Either (middle) add as a coarse nodes if disconnected ("singleton") or (right) lump as a child to an existing coarse node

➤ We will refer to this as: *Heavy Edge Matching (HEM)*

Coarsening by independent sets

Recall: An independent set $\mathcal{S} \subseteq V$ consists of vertices that are not adjacent to each other: $\boxed{i, j \in \mathcal{S} \implies a_{ij} = 0}$

- $\mathcal{S}$ is maximal if it cannot be augmented into another IS
- Can take $V_c = \mathcal{S}$ as a coarse set. Need to define edges.
- Let L = reordered graph Laplacian (n_c vertices of V_c listed first): (note: D_c is *diagonal*)

$$L = \begin{pmatrix} D_c & -F \\ -F^T & B \end{pmatrix}$$

- Replace B by $D_f = F^T \mathbb{1}$ and define G_c = graph of $S_c \rightarrow$

$$S_c = D_c - F D_f^{-1} F^T$$

Property: S_c = Graph Laplacian of coarse graph G_c

Coarsening by 'algebraic distance'

- Motivated by "bootstrap algebraic multigrid" (BAMG) [Brandt'01]
- In BAMG notion of closeness (used for coarsening) defined from a few steps of Gauss-Seidel for solving $\boldsymbol{Ax = 0}$
- Speed of convergence of the iterate determines an 'algebraic distance' between variables.
- Exploited to aggregate the unknowns and define restriction and interpolation operators. Analysis in [Chen-Safro'11]

Coarsening by 'kron' decomposition

- Kron reduction of networks proposed back in 1939 by Kron
- Revived by Dorfler and Bullo(2013) and Shuman et al. (2016)

Main idea:

- Select a coarse set V_1: Shuman-al use eigenvectors
- Reorder matrix so that nodes of V_1 come 1st. Laplacean becomes $\rightarrow$

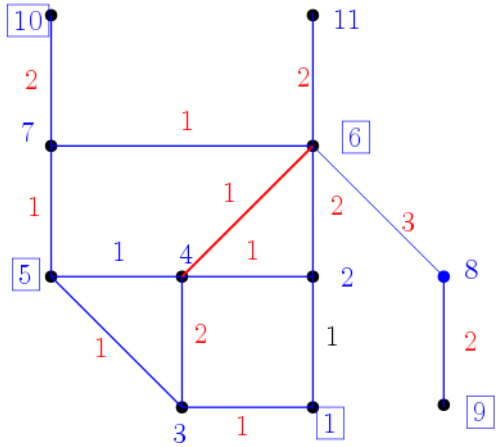
$$L = \begin{bmatrix} L_{11} & L_{12} \\ L_{12}^T & L_{22} \end{bmatrix}$$

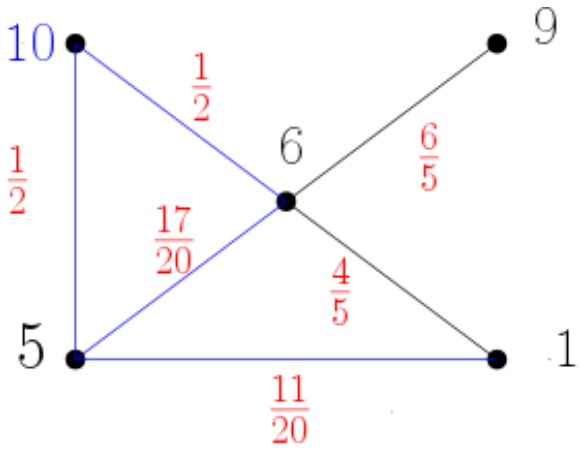
- Kron reduction of L defined as the Schur complement:

$$L(V_1) = L_{11} - L_{12} L_{22}^{-1} L_{12}^T$$

Property $L(V_1)$ == graph Laplacian of V_1 [Dorfler-Bullo]

Example:





Kron coarsening

Independent set coarsening

Two ways of using independent sets for coarsening.

D_I	$-F$
$-F^T$	B

$$L_c = D_I - FB^{-1}F^T$$

D_I	$-F$
$-F^T$	D_f

$$L_c = D_I - FD_f^{-1}F^T$$

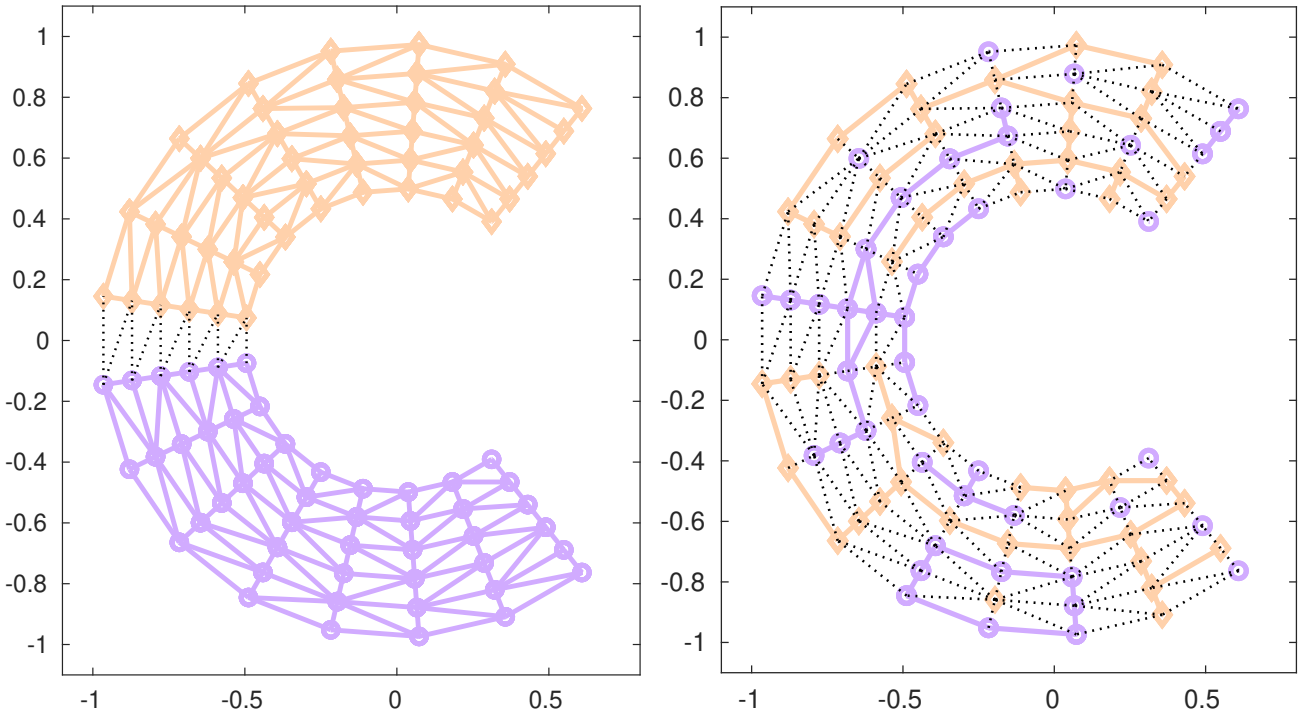
Q. 1: How to deal with 'denser' graph?

A Sparsify - using spectral sparsificaition

Q. 2: How to select V_1?

A Use signs of largest eigenvector of original Laplacian L

- If $u_1 = [\xi_1, \xi_2, \cdots, \xi_n]^T$ = the largest eigenvector.
- Define $V_+ = \{i | \xi_i \geq 0\}$ and $V_- = \{i | \xi_i < 0\}$
- Then select one of V_+, V_- as V_1.
- Opposite of what is done in spectral graph partitioning



Left side: spectral graph partitioning. Right: Coarsening with-largest eigenvector

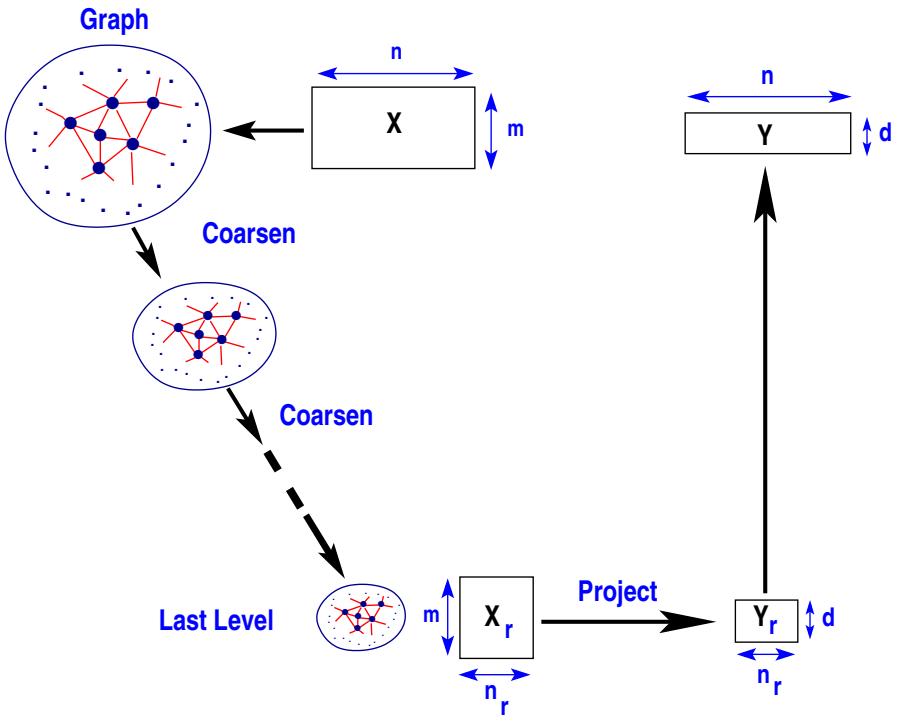
➤ Easy to show: (under mild condition on eigenvector) Each node of $\boldsymbol{V}_+$ (resp. $\boldsymbol{V}_-$) must have at least one nearest neighbor node from $\boldsymbol{V}_-$ (resp. $\boldsymbol{V}_+$).

GRAPH COARSENING IN MACHINE LEARNING

Multilevel Dimension Reduction

Idea:

Coarsen for a few levels. Use resulting data set $\hat{\boldsymbol{X}}$ to find a projector $\boldsymbol{P}$ from $\mathbb{R}^m$ to $\mathbb{R}^d$. Use this $\boldsymbol{P}$ to project data items.



- Gain: Dimension reduction is done with a much smaller set.
- Wish: not much loss compared to using whole data

Multilevel Dimension Reduction (for sparse data- e.g., text)

➤ Use Hypergraph Coarsening with *column matching* – similar to a common one used in graph partitioning

➤ Compute the non-zero inner product $\langle a^{(i)}, a^{(j)} \rangle$ between two vertices i and j, i.e., the ith and jth columns of A.

➤ Note: $\langle a^{(i)}, a^{(j)} \rangle = \|a^{(i)}\| \|a^{(j)}\| \cos \theta_{ij}$

Modif. 1: Parameter: $0 < \epsilon < 1$. Match columns i & j only if angle satisfies:

$$\tan \theta_{ij} \leq \epsilon$$

Modif. 2: Re-Scale. If i and j match and $\|a^{(i)}\|_0 \geq \|a^{(j)}\|_0$ replace $a^{(i)}$ and $a^{(j)}$ by

$$c^{(\ell)} = \left(1 + \cos^2 \theta_{ij}\right)^{\frac{1}{2}} a^{(i)}$$

➤ Call C the coarsened matrix obtained from A using the approach just described

Lemma: Let $C \in \mathbb{R}^{m \times c}$ be the coarsened matrix of A obtained by one level of coarsening of $A \in \mathbb{R}^{m \times n}$, with columns $a^{(i)}$ and $a^{(j)}$ matched if $\tan \theta_i \leq \epsilon$. Then

$$|x^T A A^T x - x^T C C^T x| \leq 3\epsilon \|A\|_F^2,$$

for any $x \in \mathbb{R}^m$ with $\|x\|_2 = 1$.

➤ Very simple bound for Rayleigh quotients for any x.

➤ Implies some bounds on singular values and norms - skipped.

➤ See details + experiments in [Ubaru-YS '19]

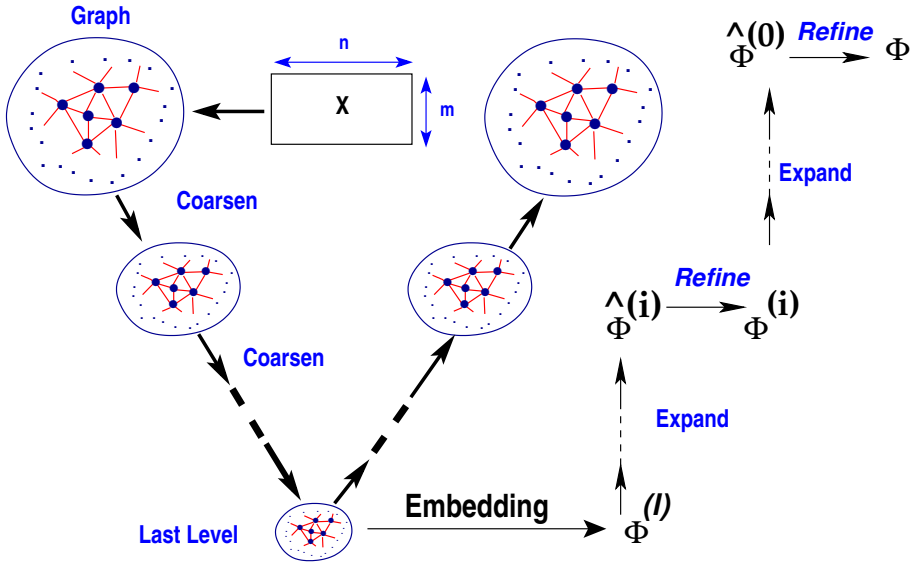
Graph coarsening for graph embeddings: HARP and MILE

- Vertex embedding: Given $G = (V, E)$ find mapping Φ:

$$\Phi : v \in V \longrightarrow \Phi(v) \in \mathbb{R}^d$$

d is small: $d \ll n$

Hierarchical Representation Learning for Networks (HARP): (Chen et al. '18) coarsen for a few levels. Find embedding $\Phi^{(\ell)}$ for coarsest graph (level ℓ). Then a succession of expansions to higher level + refinement.



- Gain: Embedding done with a much smaller set.

➤ MILE approach [Liang et al. ’18] very similar (difference in refinement).

Experiment to evaluate the effectiveness of HARP.

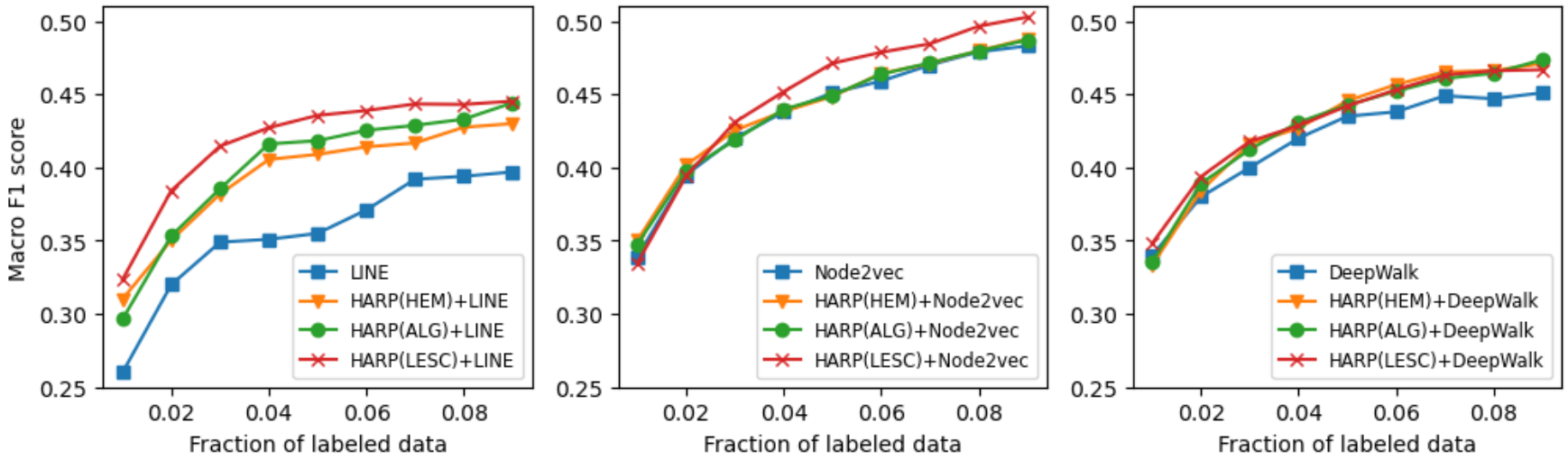
➤ Baseline. Three embedding algorithms: *DeepWalk* [Perozzi-al’14], *LINE* [Tang-al’15] and *Node2vec* [Grover-Leskovec’16]

➤ Combined with Coarsening methods:

1. Heavy Edge Matching (HEM) - sketched earlier
2. Algebraic distance (ALG) - sketched earlier
3. Leverage Score Coarsening (LESC) – variant of HEM

➤ Problem: Multilabel classification with dataset *Citeseer*

[Citation network. Publications in computer science consisting of 3.3K nodes and 4.5K edges. Label (zeros and ones) indicates research areas to which a paper belongs.]



Multi-label classification results. x-axis == portion of nodes randomly sampled for training. y-axis == Macro F_1 *score*

Coarsening with eigenvectors

- It is possible to coarsen a graph with the goal of exactly preserving a few eigenvectors.
- This has turned out not to be too useful in practice.
- Instead we use eigenvectors to define 'importance of nodes' for the graph traversal

Leverage Scores

➤ $A = U\Sigma V^T$ (ran (A) = ran (U))

➤ Leverage score of i-th row $\longrightarrow$

$$\eta_i = \|U_{i,:}\|_2^2$$

• Used to measure importance of row i in random sampling methods [e.g. El-Aloui & Mahonney '15]

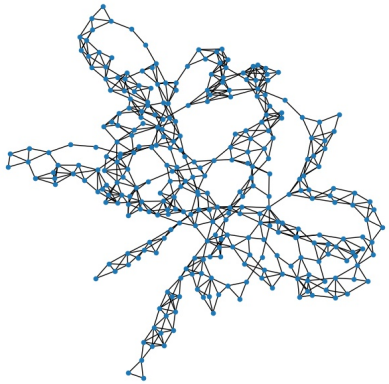
- Let A now be a graph Laplacian and $A = U\Lambda U^T$ with $\lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$

In *Leverage-score coarsening (LESC)* we dampen lower sing. vectors $\rightarrow$

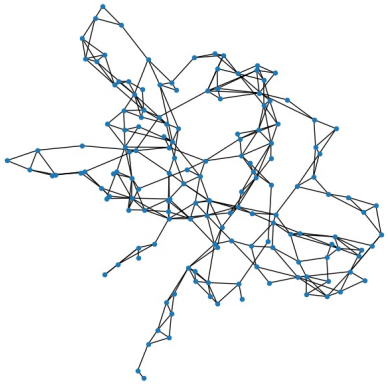
$$\eta_i = \sum_{k=1}^{r} (e^{-\tau\lambda_k} U_{ik})^2$$

- Use η_i to decide order of traversal in coarsening algorithm
- Slightly different way of handling left-over nodes ('singletons')
- Next: visualization with 5 different coarsening methods on a graph with $n = 312$ nodes and $ne = 761$ edges

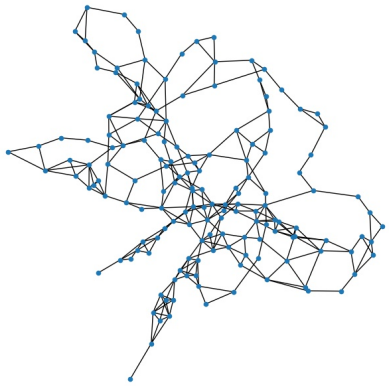
Original, $ne = 761$



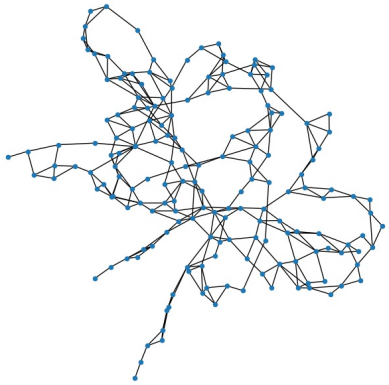
HEM $ne = 340$



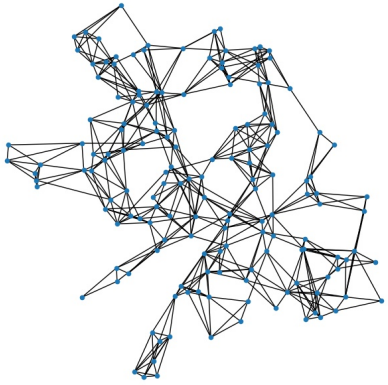
LV[1],$ne = 321$



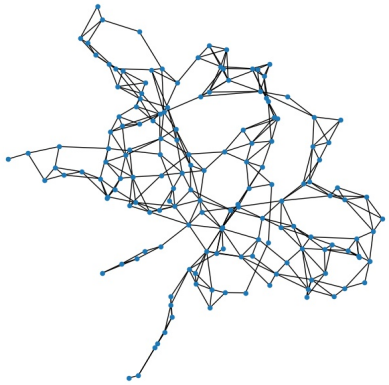
ALG,$ne = 327$



Kron, $ne = 485$



LESC, $ne = 362$



1. Local Variation (Loukas'2019)

Consider case when $r = n$ (or simply r is large)

$$\eta_i = \sum_{k=1}^{n} (e^{-\tau\lambda_k} U_{ik})^2 = \sum_{k=1}^{n} e^{-2\tau\lambda_k} |U_{ik}|^2 = e_i^T e^{-2\tau L} e_i.$$

➤ η_i equals the i-th diagonal entry of the matrix $H \equiv \exp(-2\tau L)$

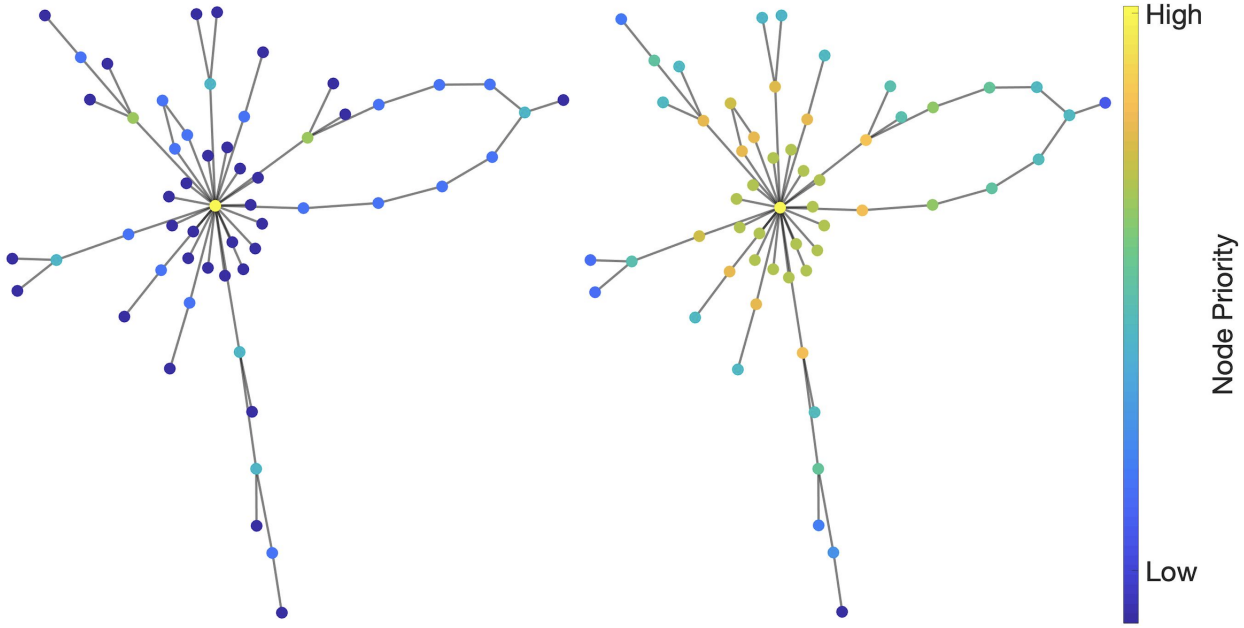
Alternative definition

➤ We consider the following alternative - related to $L^\dagger$

$$\eta_i = \sum_{j=2}^{n} \left(\frac{1}{\sqrt{\lambda_j}} U_{ij} \right)^2 \rightarrow \eta_i \approx \sum_{j=2}^{r} \left(\frac{1}{\sqrt{\lambda_j}} U_{ij} \right)^2$$

Property:

$$L^\dagger_{ii} = \sum_{j=2}^{n} \frac{U_{ij}}{\sqrt{\lambda_j}} \frac{U_{ij}}{\sqrt{\lambda_j}} = \eta_i$$



Traversal order: HEM (left) and LESC (right) on a small graph.

Analysis

- $L^\dagger$ has long been used to define node importance
- The nonzero entries of $L^\dagger$ define *resistance distance*. Its trace is the *Effective graph resistance*. Related to *betweenness* centrality measure... + many other links.
- Important fact: η helps measure the change in $L^\dagger$

Let the graph be connected. The magnitude of the difference between $L^\dagger$ and $L^\dagger_\infty$ caused by assigning the $+\infty$ edge weight to an edge $e(i,j)$ is bounded by

$$||\Delta L^\dagger||_F^2 \leq \kappa(L)(L^\dagger_{ii} + L^\dagger_{jj}),$$

where $\kappa \equiv$ effective condition number.

[Adapted from a result of Hermsdorff and Gunderson'19]

Application: Graph classification

Problem: determine the label of a graph [e.g., graph of a molecule in chemistry applications].

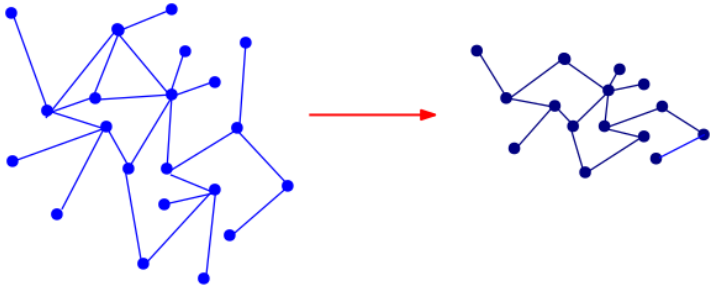
Method: Graph Neural Networks [GNN]

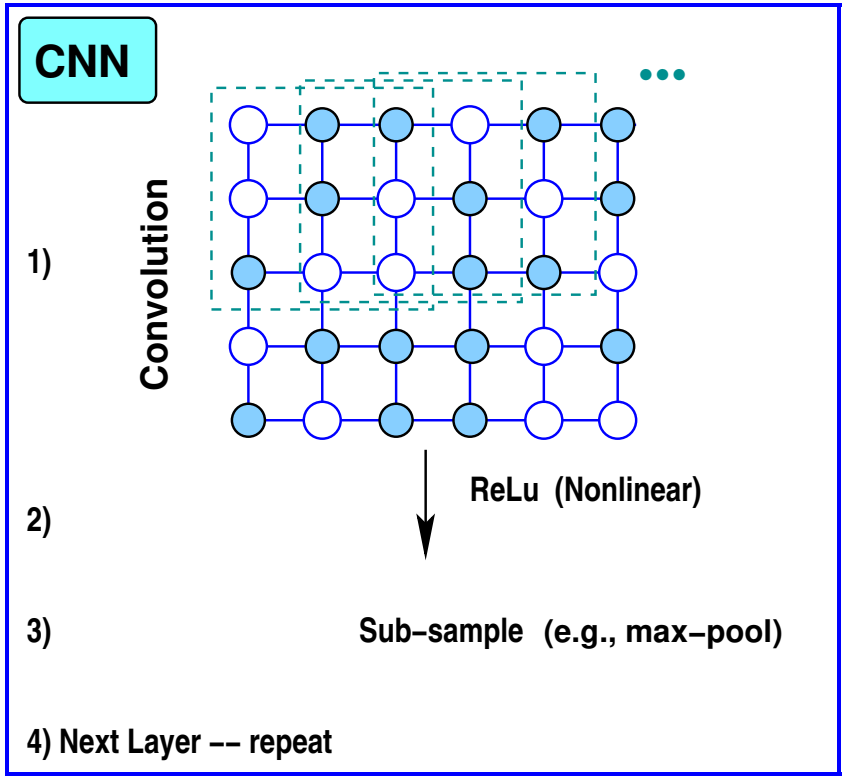
➤ GNNs find an embedding of a graph by using several 'pooling' layers of a neural network. We use:

1. **SortPool**
2. **DiffPool**
3. **TopKPool**
4. **SAGPool**

What are these 'pooling' methods?

Aim: generalize the convolution and subsampling layers of Convolutional Neural Networks to graphs:





➤ End result : embedding of a graph.

Datasets:

D&D protein data set (predict protein functions from structure)

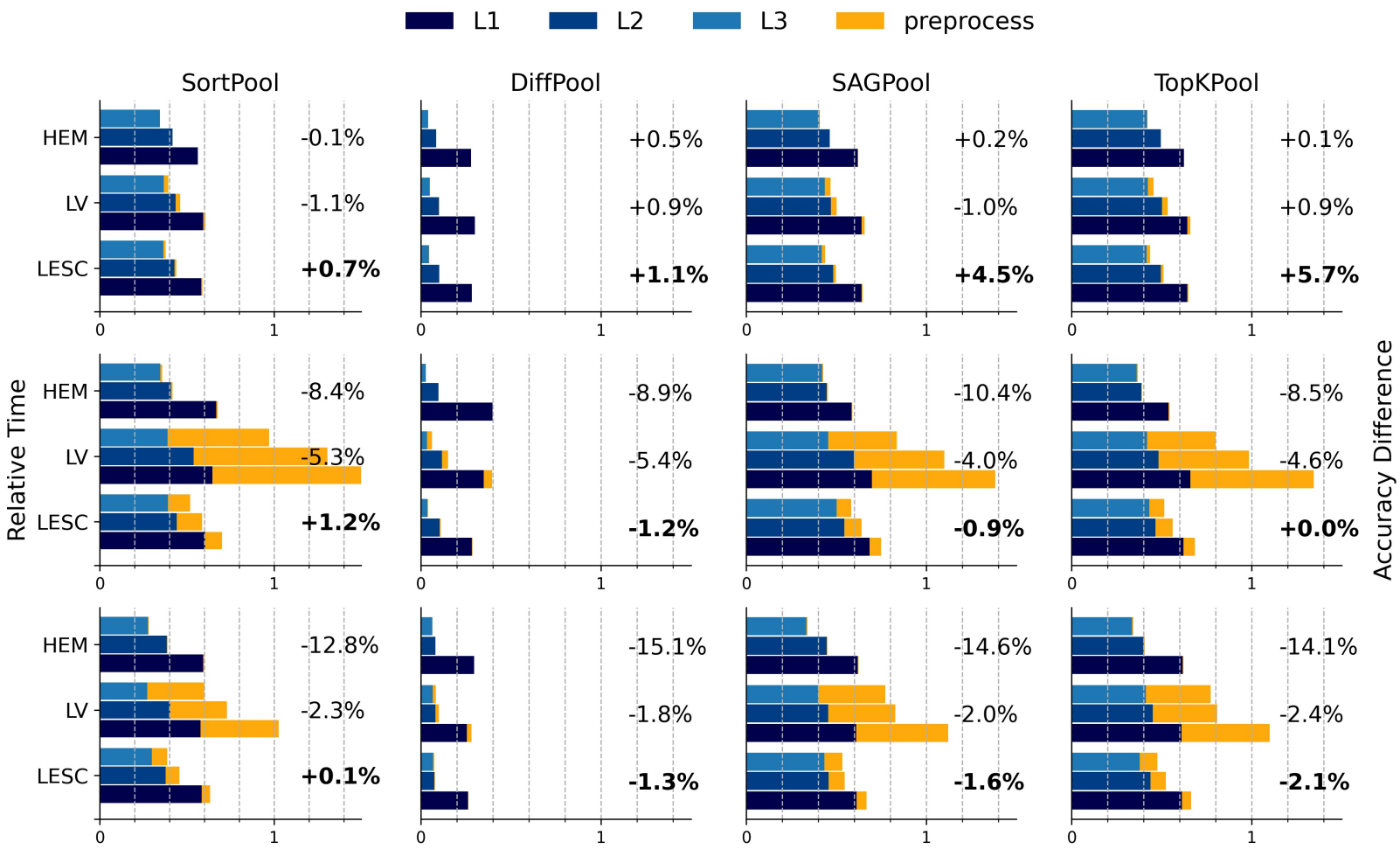
REDDIT-BINARY (REBI) and

REDDIT-MULTI-5K (RE5K) social network data sets from the discussion forum *Reddit* [Graph: discussion threads]

Stats.

	DD	REBI	RE5K
#GRAPHS	1178	2000	4999
#CLASSES	2	2	5
AVG.#NODES	284.32	429.63	508.52
AVG.#EDGES	715.66	497.75	594.87

➤ Method: preprocess (coarsen) each graph prior to using it.



Relative times vs. original (No coarsening). Percentates on right of each figure: gain (loss) in accuracy

Conclusion

- *Many* interesting new matrix problems in areas that involve the effective exploitation of data
- Many online resources available
- Huge potential in scientific areas like materials science
- To a researcher in computational linear algebra : Tsunami of change on types or problems, algorithms, frameworks, culture,..
- But change should be welcome :

➤ From “Who Moved My Cheese?” [Spencer Johnson ’02]:

“The quicker you let go of old cheese, the sooner you find new cheese.”

“If you do not change, you can become extinct!”

Thank you !

➤ Visit my web-site at `www.cs.umn.edu/~saad`