SPECTRAL DENSITIES

Spectral Densities - Introduction

- Spectral density == function that provides a global representation of the spectrum of a Hermitian matrix
- Known in solid state physics as *‘Density of States’* (DOS)
- Very useful in physics
- Almost unknown (as a tool) in numerical linear algebra

Density of States

➤ 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
- $\lambda_1 \le \lambda_2 \le \cdots \le \lambda_n$ are the eigenvalues of A

➤ DOS is also referred to as the spectral density

➤ Note: number of eigenvalues in an interval $[a, b]$ is

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

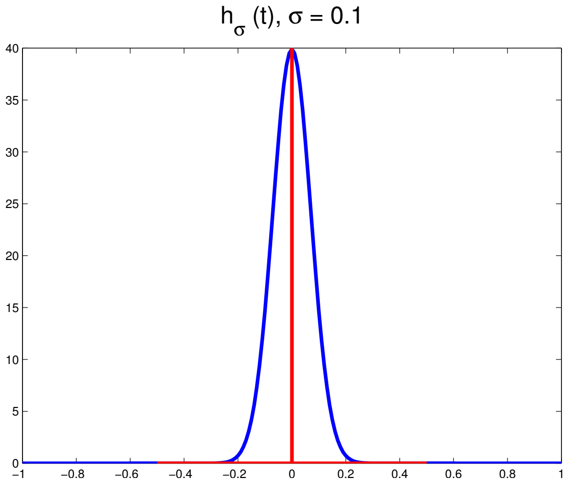
Issue: How to deal with distributions?

➤ Highly 'discontinuous', not easy to handle numerically

➤ Solution for practical and theoretical purposes: replace ϕ by a regularized ('blurred') version ϕ_σ:

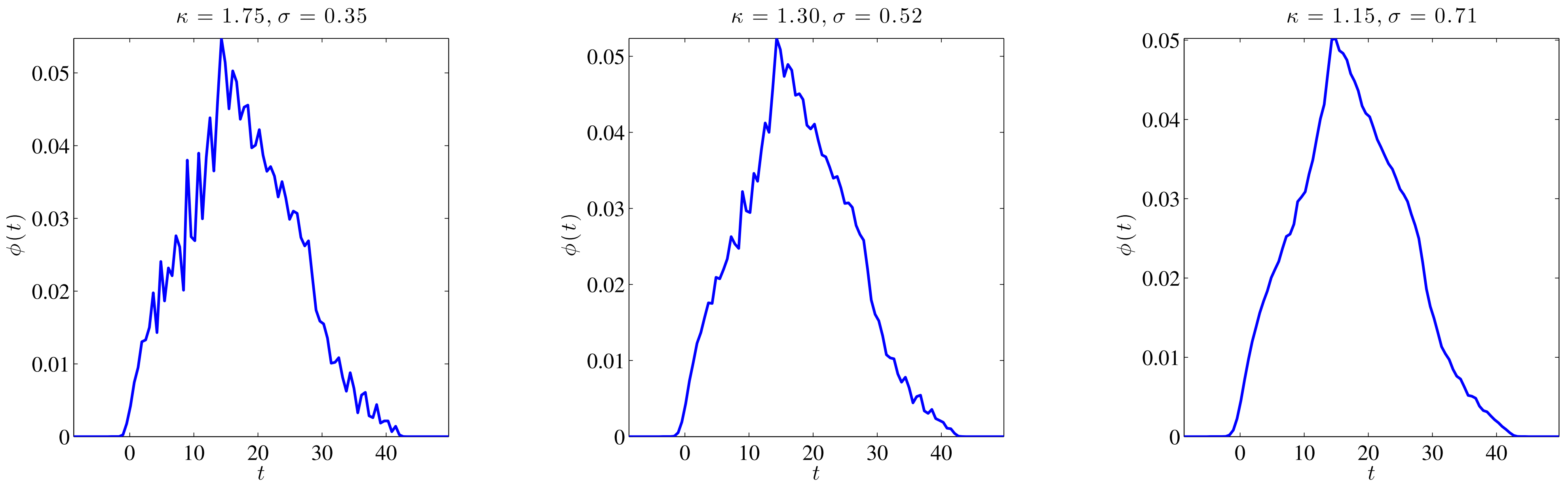
$$\phi_\sigma(t) = \frac{1}{n}\sum_{j=1}^{n} h_\sigma(t-\lambda_j),$$

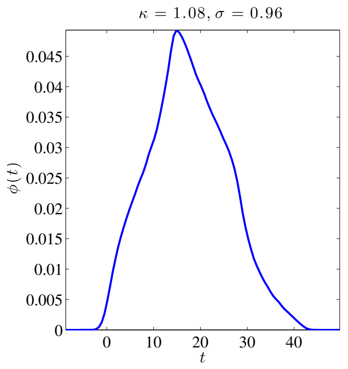
Where, for example: $h_\sigma(t) = \frac{1}{(2\pi\sigma^2)^{1/2}} e^{-\frac{t^2}{2\sigma^2}}$.



➤ Smoothed $\phi(t)$ can be viewed as a probability distribution function for the spectrum

➤ How to select smoothing parameter σ? Example for Si_2





- ➤ Higher $\sigma \rightarrow$ smoother curve
- ➤ But loss of detail ..
- ➤ Compromise: $\sigma = \frac{h}{2\sqrt{2\log(\kappa)}}$,
- ➤ h = resolution, κ = parameter > 1

Computing the DOS: 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 estimator to get traces needed in calculations ➤ Assume change of variable done so eigenvalues lie in $[-1,\ 1]$.

- To avoid weight function expand $\sqrt{1-t^2}\phi \rightarrow$

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

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

- Expansion coefficients μ_k are formally defined by:

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

- Here $2-\delta_{k0}$ == 1 when $k=0$ and == 2 otherwise.

- Note: $\sum T_k(\lambda_i) = \text{Trace}[T_k(A)]$ $\longrightarrow$ Estimate this, e.g., via stochastic estimator

- Generate random vectors $v^{(1)}, v^{(2)}, \cdots, v^{(n_{\text{vec}})}$

- Each vector is normalized so that $\|v^{(l)}\| = 1, l = 1, \dots, n_{\text{vec}}$.

- Estimate the trace of $T_k(A)$ with stochastisc estimator:
$$\text{Trace}(T_k(A)) \approx \frac{1}{n_{\text{vec}}} \sum_{l=1}^{n_{\text{vec}}} \left(v^{(l)}\right)^T T_k(A) v^{(l)}.$$

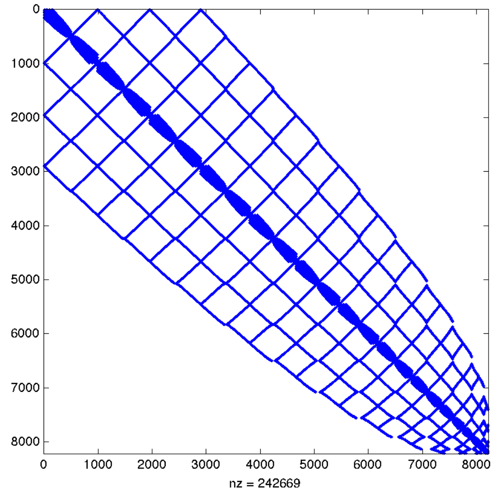
- Will lead to the desired estimate:
$$\boxed{\mu_k \approx \frac{2 - \delta_{k0}}{n\pi n_{\text{vec}}} \sum_{l=1}^{n_{\text{vec}}} \left(v^{(l)}\right)^T T_k(A) v^{(l)}.}$$

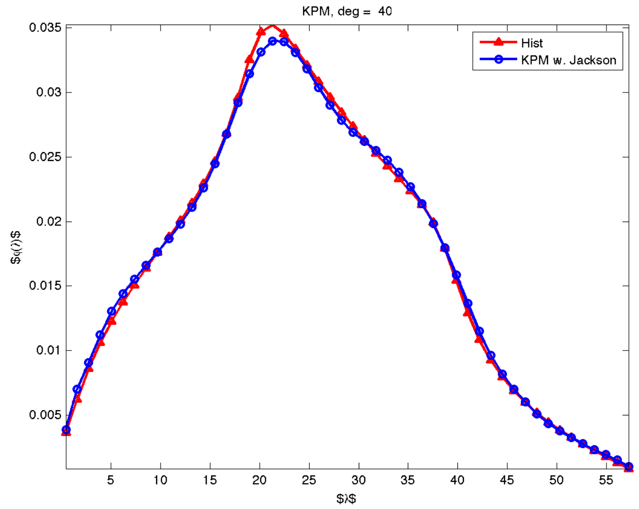
- To compute scalars of the form $v^T T_k(A) v$, exploit 3-term recurrence of the Chebyshev polynomial: $\boxed{T_{k+1}(A)v = 2AT_k(A)v - T_{k-1}(A)v}$

- If we let $v_k \equiv T_k(A)v$, we have
$$\boxed{v_{k+1} = 2Av_k - v_{k-1}}$$

An example: The Benzene matrix

```
>> TestKpmDos
 Matrix Benzene n =8219  nnz = 242669
Degree =  40  # sample vectors =  10
Elapsed time is 0.235189 seconds.
```





Use of the Lanczos Algorithm

➤ Recall: The Lanczos algorithm generates an orthonormal basis $V_m = [v_1, v_2, \cdots, v_m]$ for the Krylov subspace:

$$\boxed{\mathbf{span}\{v_1, Av_1, \cdots, A^{m-1}v_1\}}$$

➤ ... such that:
$V_m^H A V_m = T_m$ - with

$$T_m = \begin{pmatrix} \alpha_1 & \beta_2 & & & & \\ \beta_2 & \alpha_2 & \beta_3 & & & \\ & \beta_3 & \alpha_3 & \beta_4 & & \\ & & . & . & . & \\ & & & . & . & . \\ & & & & \beta_m & \alpha_m \end{pmatrix}$$

➤ Lanczos process builds orthogonal polynomials wrt to dot product:

$$\int p(t)q(t)dt \equiv (p(A)v_1, q(A)v_1)$$

➤ Let $\theta_i, \ i = 1 \cdots, m$ be the eigenvalues of T_m [Ritz values]

➤ y_i's associated eigenvectors; Ritz vectors: $\{V_m y_i\}_{i=1:m}$

➤ Ritz values approximate eigenvalues

➤ Could compute θ_i's then get approximate DOS from these

➤ Problem: θ_i not good enough approximations – especially inside the spectrum.

Better idea: exploit relation of Lanczos with (discrete) orthogonal polynomials and related Gaussian quadrature:

$$\int p(t)dt \approx \sum_{i=1}^{m} a_i p(\theta_i) \quad a_i = \left[e_1^T y_i\right]^2$$

- ➤ See, e.g., Golub & Meurant '93, and also Gautschi'81, Golub and Welsch '69.
- ➤ Formula exact when p is a polynomial of degree $\leq 2m+1$

➤ Consider now $\int p(t)dt = < p, 1 > =$ (Stieljes) integral $\equiv$

$$(p(A)v, v) = \sum \beta_i^2 p(\lambda_i) \equiv < \phi_v, p >$$

➤ Then $\langle \phi_v, p \rangle \approx \sum a_i p(\theta_i) = \sum a_i \langle \delta_{\theta_i}, p \rangle \rightarrow$

$$\phi_v \approx \sum a_i \delta_{\theta_i}$$

➤ To mimick the effect of $\beta_i = 1, \forall i$, use several vectors v and average the result of the above formula over them..

● *Approximating spectral densities of large matrices*, Lin Lin, YS, and Chao Yang - SIAM Review '16. Also in:
[arXiv: http://arxiv.org/abs/1308.5467]

Application 1: Eigenvalue counts

Problem: Given A (Hermitian) find an estimate of the number $\mu_{[a,b]}$ of eigenvalues of A in $[a, \quad b]$.

Standard method: Sylvester inertia theorem $\rightarrow$ expensive!

First alternative: integrate the Spectral Density in $[a, \ b]$.

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

Second method: Estimate trace of the related spectral projector P ($\rightarrow u_i$'s = eigenvectors $\leftrightarrow \lambda_i$'s)

$$P = \sum_{\lambda_i \ \in \ [a \ \ b]} u_i u_i^T .$$

➤ It turns out that the 2 methods are identical.

Application 3: Estimating the rank

➤ Very important problem in signal processing applications, machine learning, etc.

➤ Often: a certain rank is selected ad-hoc. Dimension reduction is application with this “guessed” rank.

➤ Can be viewed as a particular case of the eigenvalue count problem - but need a cutoff value..

Approximate rank, Numerical rank

- Notion defined in various ways. A common one:

$$r_\epsilon = \min\{rank(B) : B \in \mathbb{R}^{m\times n}, \|A - B\|_2 \le \epsilon\},$$

$$r_\epsilon = \text{ Number of sing. values} \ge \epsilon$$

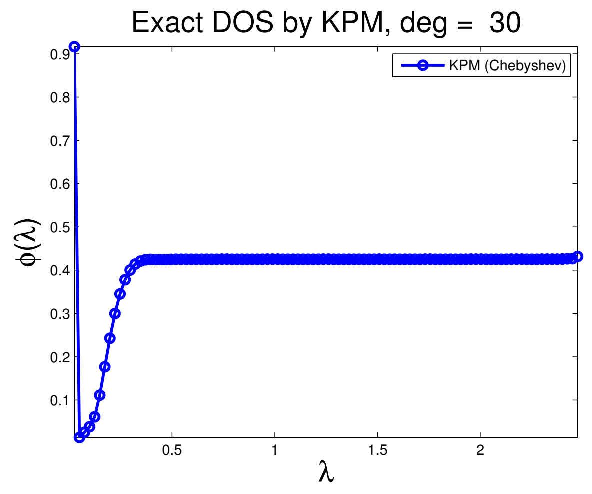
- Two distinct problems:

1. Get a good ϵ 2. Estimate number of sing. values $\ge \epsilon$

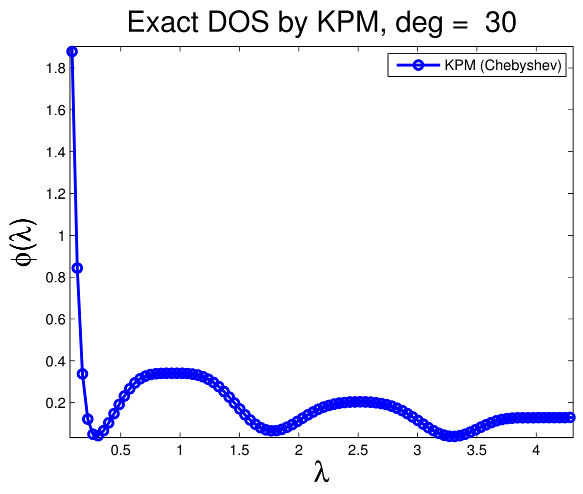
- We will need a cut-off value ('threshold') ϵ.
- Could use 'noise level' for ϵ, but not always available

Threshold selection

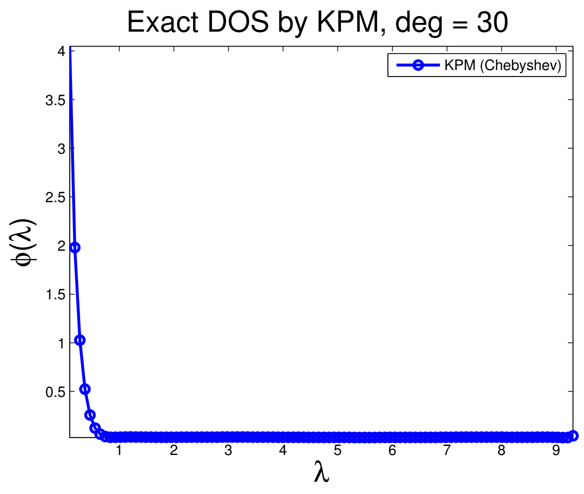
- How to select a good threshold?
- Answer: Obtain it from the DOS function



(A)



(B)



(C)

Exact DOS plots for three different types of matrices.

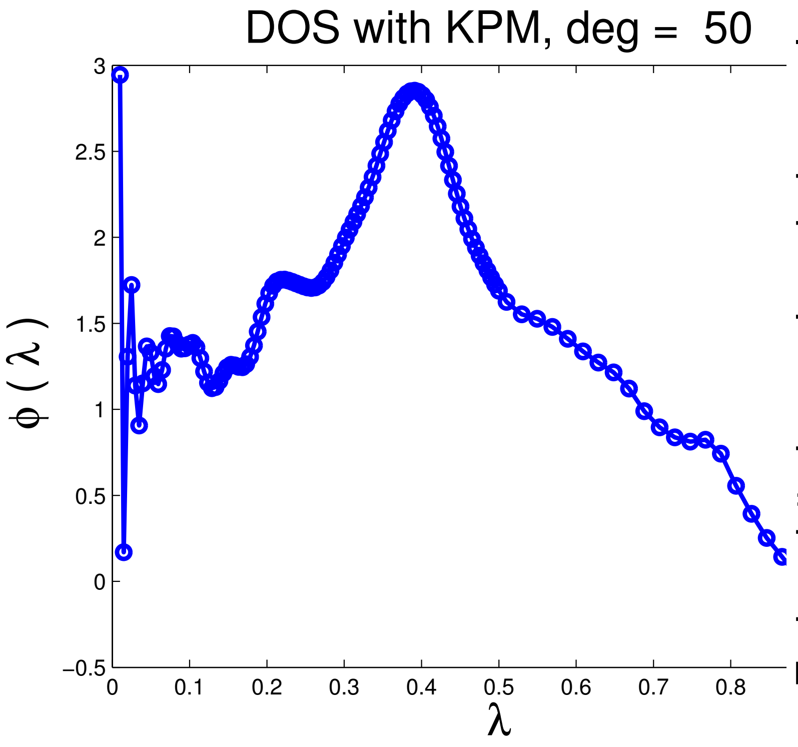
- To find: point immediatly following the initial sharp drop observed.
- Simple idea: use derivative of DOS function ϕ
- For an $n \times n$ matrix with eigenvalues $\lambda_n \le \lambda_{n-1} \le \cdots \le \lambda_1$:

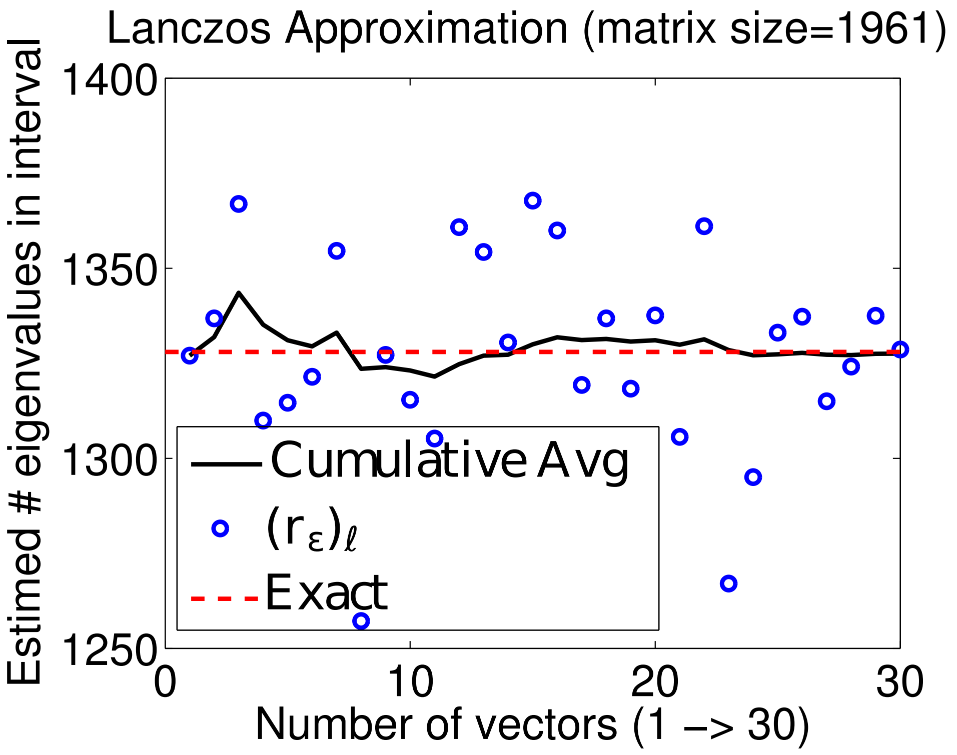
$$\boxed{\epsilon = \min\{t : \lambda_n \le t \le \lambda_1, \phi'(t) = 0\}.}$$

- In practice replace by

$$\boxed{\epsilon = \min\{t : \lambda_n \le t \le \lambda_1, |\phi'(t)| \ge \text{tol}\}}$$

Experiment: estimated rank by Lanczos for matrix `netz4504`.





(A) (B)

(A) The DOS found by KPM. (B) Approximate rank estimation by Lanczos

Tests with Matérn covariance matrices for grids

- Important in statistical applications

Approximate Rank Estimation of Matérn covariance matrices

Type of Grid (dimension)	Matrix Size	# λ_i's $\geq \epsilon$	r_ϵ KPM	r_ϵ Lanczos
1D regular Grid (2048×1)	2048	16	16.75	15.80
1D no structure Grid (2048×1)	2048	20	20.10	20.46
2D regular Grid (64×64)	4096	72	72.71	72.90
2D no structure Grid (64×64)	4096	70	69.20	71.23
2D deformed Grid (64×64)	4096	69	68.11	69.45

- For all test $M(deg) = 50, n_v$=30

Application 4: The LogDeterminant

Evaluate the Log-determinant of A:

$$\log\det(A) = \text{Trace}(\log(A)) = \sum_{i=1}^{n} \log(\lambda_i).$$

A is SPD.

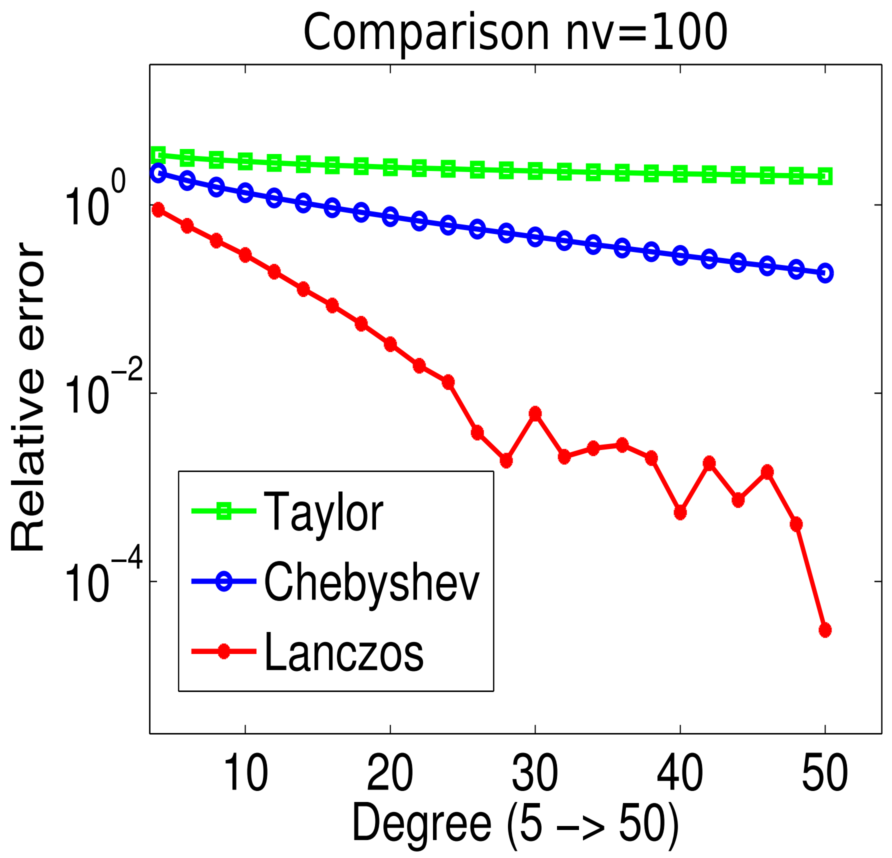
➤ Estimating the log-determinant of a matrix equivalent to estimating the trace of the matrix function $f(A) = \log(A)$.

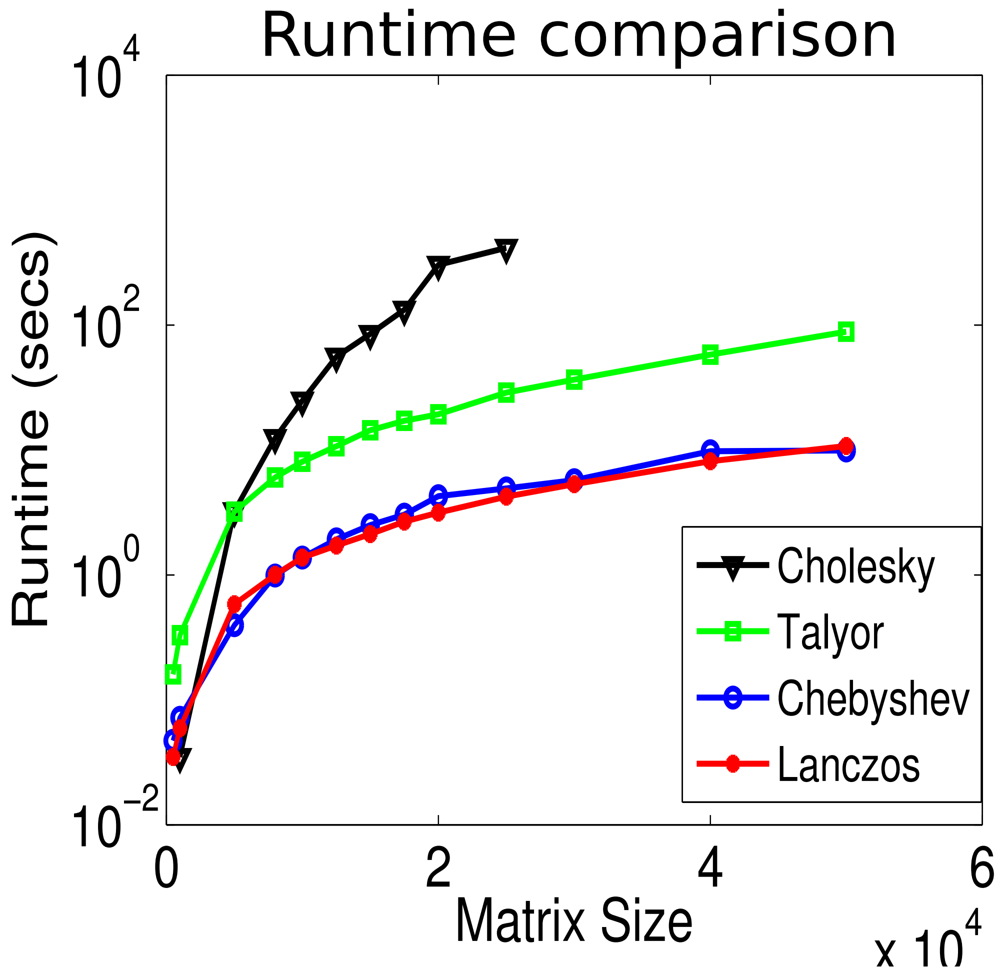
➤ Can invoke Stochastic Lanczos Quadrature (SLQ) to estimate this trace.

Numerical example: A graph Laplacian `california` of size 9664×9664, $nz \approx 10^5$ from the Univ. of Florida collection.

Rel. error vs degree

- 3 methods: Taylor Series, Chebyshev expansion, SLQ
- # starting vectors $nv = 100$ in all three cases.





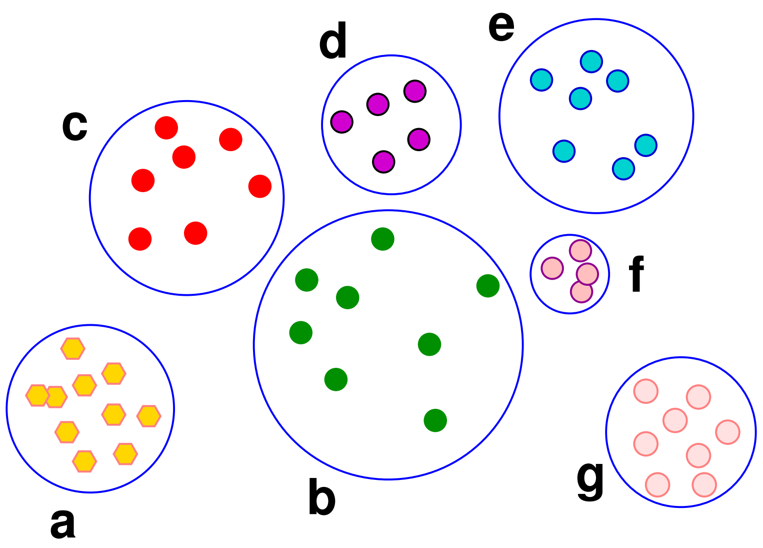
➤ Many more applications!

SUPERVISED LEARNING

Supervised learning

➤ We now have data that is ‘labeled’

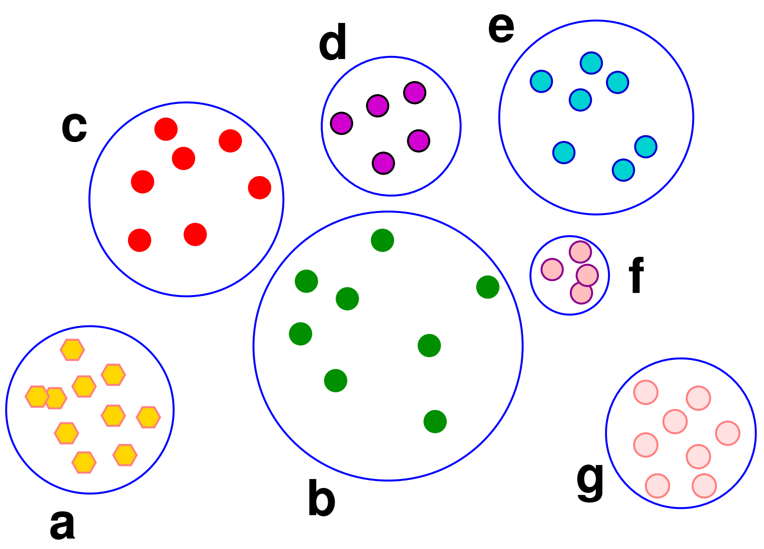
Examples: Health Sciences (‘malignant’- ’non malignant’) ; Materials (’photovoltaic’, ’hard’, ’conductor’, ...) ; Digit Recognition (’0’, ’1’, ..., ’9’)

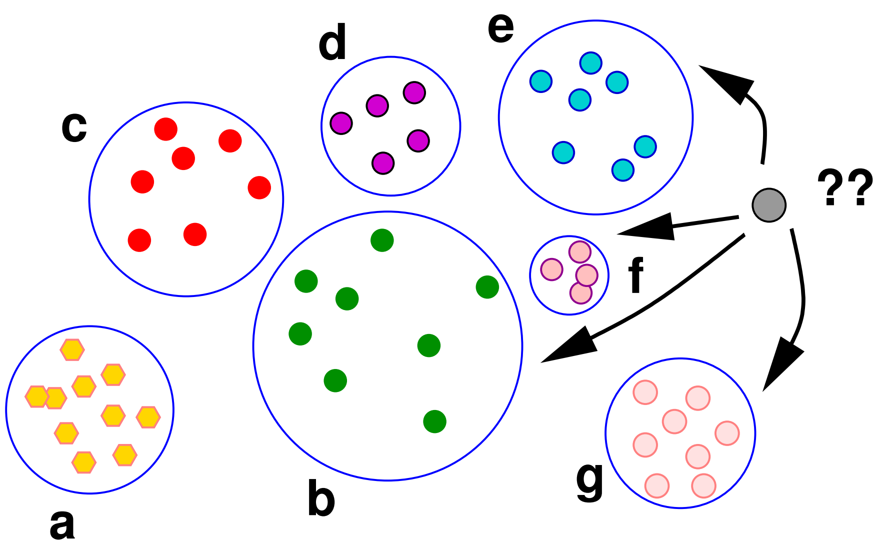


Supervised learning

- We now have data that is ‘labeled’

Examples: Health Sciences (‘malignant’- ’non malignant’) ; Materials (’photovoltaic’, ’hard’, ’conductor’, ...) ; Digit Recognition (’0’, ’1’, ..., ’9’)

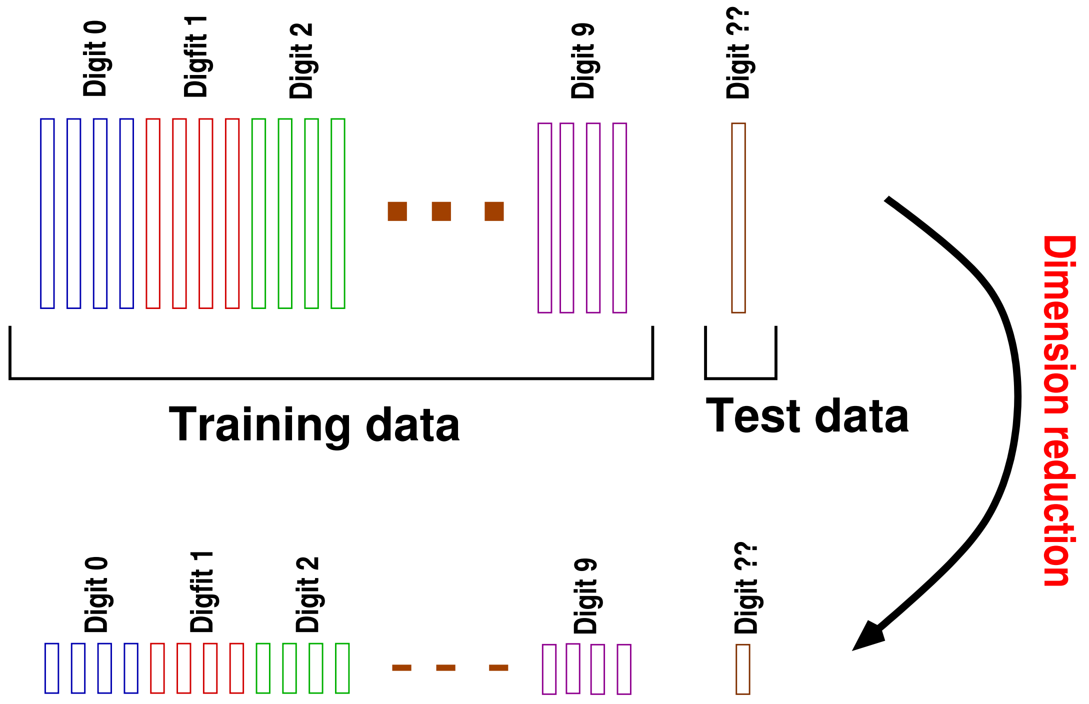




Supervised learning: classification

➤ Best illustration: written digits recognition example

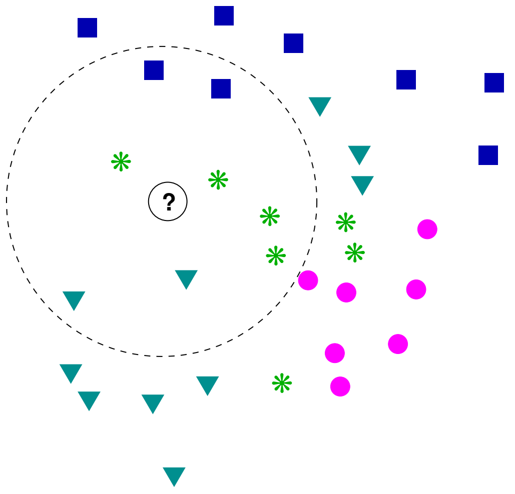
> *Given:* set of labeled samples (training set), and an (unlabeled) test image x.
> *Problem:* label of x =?



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

Basic method: K-nearest neighbors (KNN) classification

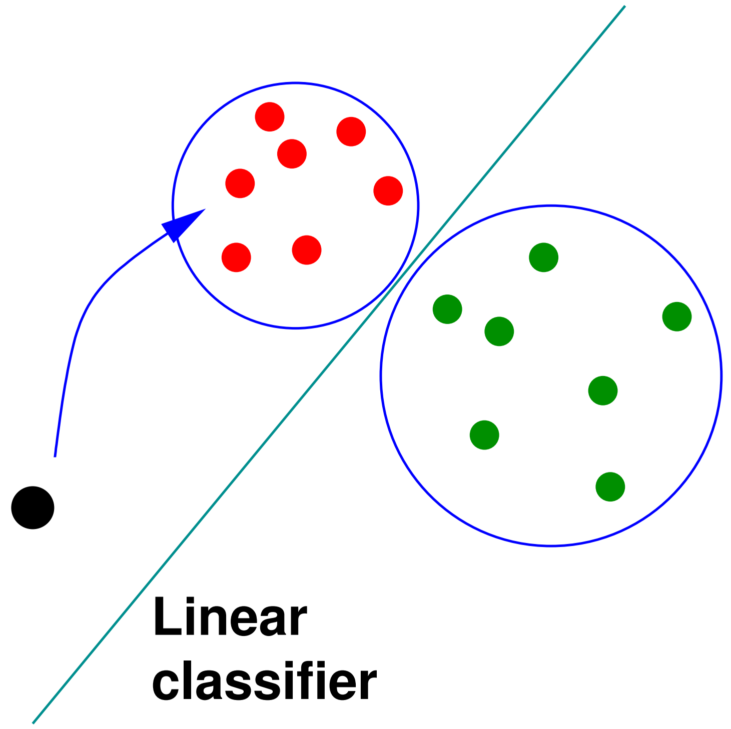
- ➤ Idea of a voting system: get distances between test sample and training samples
- ➤ Get the k nearest neighbors (here $k = 8$)
- ➤ Predominant class among these k items is assigned to the test sample ("$*$" here)



Supervised learning: Linear classification

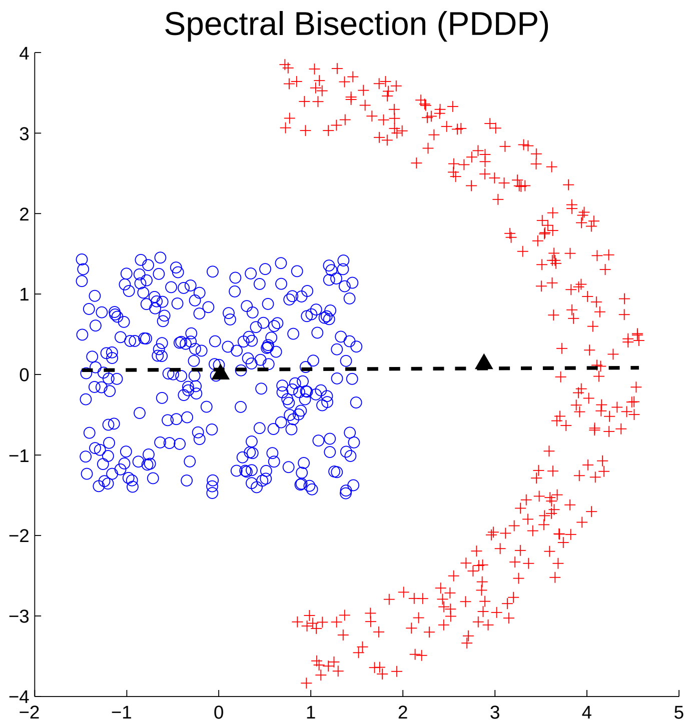
Linear classifiers: Find a hyperplane which best separates the data in classes A and B.

➤ Example of application: Distinguish between SPAM and non-SPAM e-mails

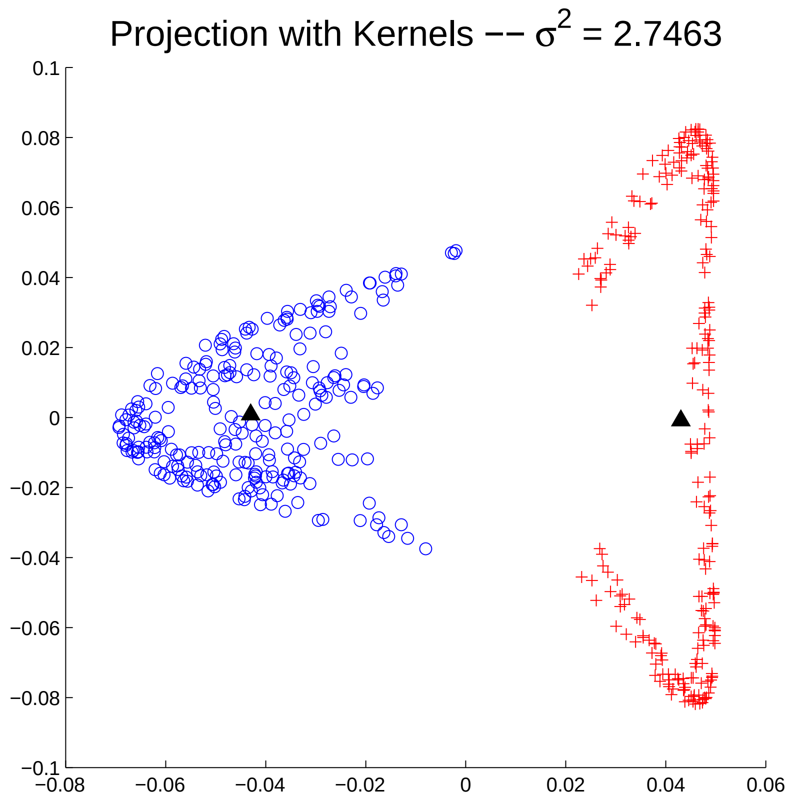


➤ Note: The world in non-linear. Often this is combined with Kernels – amounts to changing the inner product

A harder case:



- Use kernels to transform

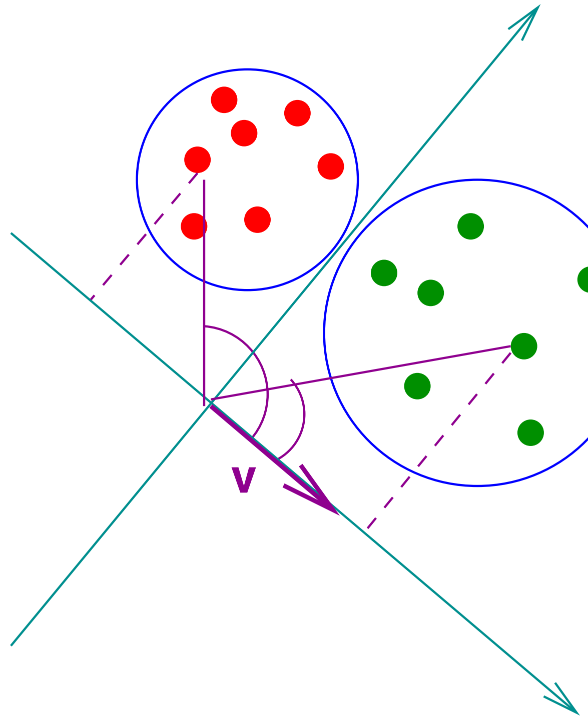


Transformed data with a Gaussian Kernel

Simple linear classifiers

- Let $\boldsymbol{X} = [\boldsymbol{x}_1, \cdots, \boldsymbol{x}_n]$ be the data matrix.
- and $\boldsymbol{L} = [\boldsymbol{l}_1, \cdots, \boldsymbol{l}_n]$== labels. $\boldsymbol{l}_i = \pm 1$
- 1st Solution: Find a vector $\boldsymbol{u}$ such that $\boldsymbol{u}^T \boldsymbol{x}_i$ close to $\boldsymbol{l}_i$, $\forall \boldsymbol{i}$
- Common solution: SVD to reduce dimension of data [e.g. 2-D] then do comparison in this space. e.g.

A: $\boldsymbol{u}^T \boldsymbol{x}_i \geq 0$, B: $\boldsymbol{u}^T \boldsymbol{x}_i < 0$



[For clarity: principal axis $\boldsymbol{u}$ drawn below where it should be]

Fisher's Linear Discriminant Analysis (LDA)

Principle: Use label information to build a good projector, i.e., one that can 'discriminate' well between classes

➤ Define "between scatter": a measure of how well separated two distinct classes are.

➤ Define "within scatter": a measure of how well clustered items of the same class are.

➤ Objective: make "between scatter" measure large and "within scatter" small.

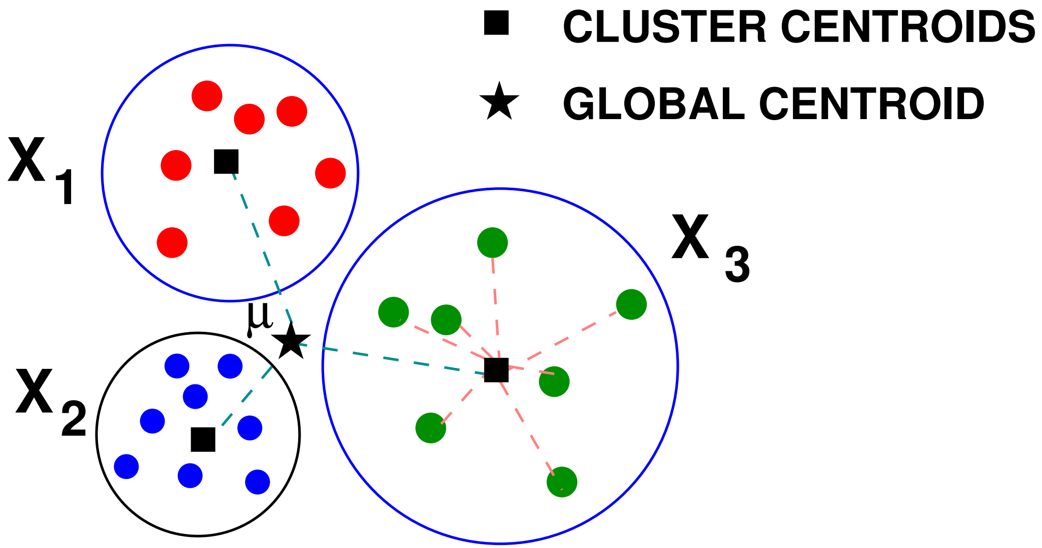
Idea: Find projector that maximizes the ratio of the "between scatter" measure over "within scatter" measure

$$S_B = \sum_{k=1}^{c} n_k(\mu^{(k)} - \mu)(\mu^{(k)} - \mu)^T,$$

$$S_W = \sum_{k=1}^{c} \sum_{x_i \in X_k} (x_i - \mu^{(k)})(x_i - \mu^{(k)})^T$$

$\boxed{\text{where:}}$

- μ = mean (X)
- $\mu^{(k)}$ = mean (X_k)
- X_k = k-th class
- $n_k = |X_k|$



- Consider 2nd moments for a vector $\boldsymbol{a}$:

$$\boldsymbol{a}^T S_B \boldsymbol{a} = \sum_{i=1}^{c} n_k \, |\boldsymbol{a}^T(\boldsymbol{\mu}^{(k)} - \boldsymbol{\mu})|^2,$$

$$\boldsymbol{a}^T S_W \boldsymbol{a} = \sum_{k=1}^{c} \sum_{x_i \in X_k} |\boldsymbol{a}^T(\boldsymbol{x}_i - \boldsymbol{\mu}^{(k)})|^2$$

- $\boldsymbol{a}^T S_B \boldsymbol{a} \equiv$ weighted variance of projected $\boldsymbol{\mu}_j$'s

- $\boldsymbol{a}^T S_W \boldsymbol{a} \equiv$ w. sum of variances of projected classes X_j's

- LDA projects the data so as to maximize the ratio of these two numbers:

$$\boxed{\max_{a} \frac{\boldsymbol{a}^T S_B \boldsymbol{a}}{\boldsymbol{a}^T S_W \boldsymbol{a}}}$$

- Optimal $\boldsymbol{a}$ = eigenvector associated with top eigenvalue of:

$$S_B u_i = \lambda_i S_W u_i \ .$$

LDA – Extension to arbitrary dimensions

- Criterion: maximize the ratio of two traces: $\frac{\text{Trace}[U^T S_B U]}{\text{Trace}[U^T S_W U]}$

- Constraint: $U^T U = I$ (orthogonal projector).

- Reduced dimension data: $Y = U^T X$.

Common viewpoint: hard to maximize, therefore ...

- ... alternative: Solve instead the ('easier') problem: $\max_{U^T S_W U = I} \text{Trace}[U^T S_B U]$

- Solution: largest eigenvectors of $S_B u_i = \lambda_i S_W u_i$.

In Brief: Support Vector Machines (SVM)

- Similar in spirit to LDA. Formally, SVM finds a hyperplane that best separates two training sets belonging to two classes.

- If the hyperplane is: $\boxed{w^T x + b = 0}$

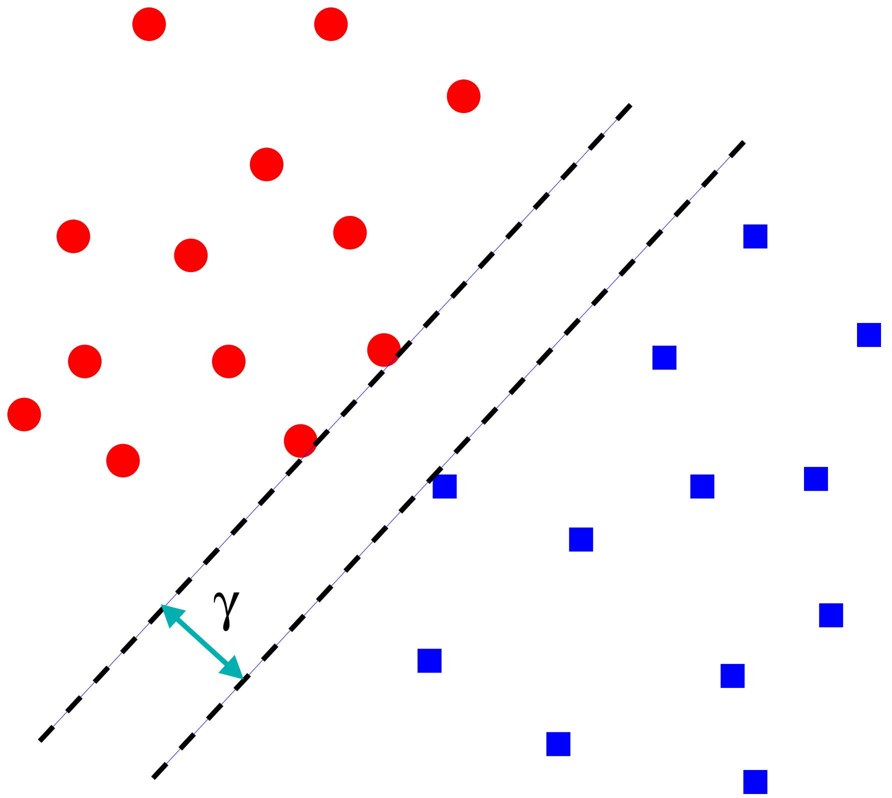
- Then the classifier is $f(x) = \text{sign}(w^T x + b)$: assigns $y = +1$ to one class and $y = -1$ to other

- Normalize parameters w, b by looking for hyperplanes of the form $w^T x + b \geq 1$ to include one set and $w^T x + b \leq -1$ to include the other.

- With $y_i = +1$ for one class and $y_i = -1$ for the other, we can write the constraints as $y_i(w^T x_i + b) \geq 1$.

- The margin is the maximum distance between two such planes: goal find w, b to maximize margin.

- Maximize margin subject to the constraint $y_i(w^T x_i + b) \geq 1$.



- As it turns out the margin is equal to: $\boxed{\gamma = \frac{2}{\|w\|_2}}$

✍ Prove it.

➤ Need to solve the constrained quadratic programming problem:

$$\boxed{\begin{array}{ll} \min_{w.b} & \frac{1}{2}\|w\|_2^2 \\ \text{s.t.} & y_i(w^T x_i + b) \ge 1, \quad \forall x_i. \end{array}}$$

Modification 1: Soft margin. Consider hinge loss: $\max\{0, 1 - y_i[w^T x_i + b]\}$

➤ Zero if constraint satisfied for pair x_i, y_i. Otherwise proportional to distance from corresponding hyperplane. Hence we can minimize

$$\lambda\|w\|^2 + \frac{1}{n}\sum_{i=1}^{n} \max\{0, 1 - y_i[w^T x_i + b]\}$$

✍ Suppose $y_i = +1$ and let $d_i = 1 - y_i[w^T x_i + b]$. Show that the distance between x_i and hyperplane $\boxed{w^T x_i + b = +1}$ is $d_i/\|w\|$.

Modification 2: Use in combination with a Kernel to improve separability

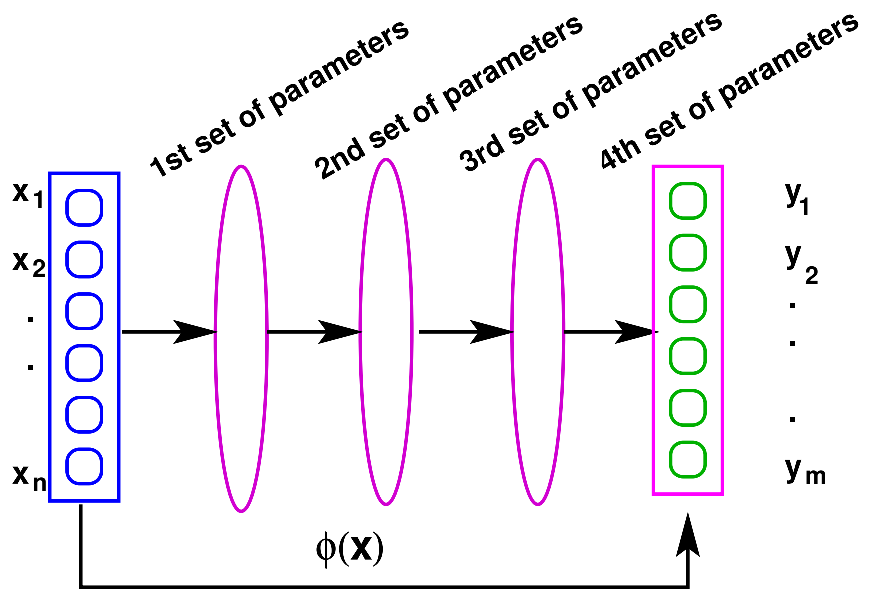
A few words on Deep Neural Networks (DNNs)

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

➤ Two reasons for the come-back:

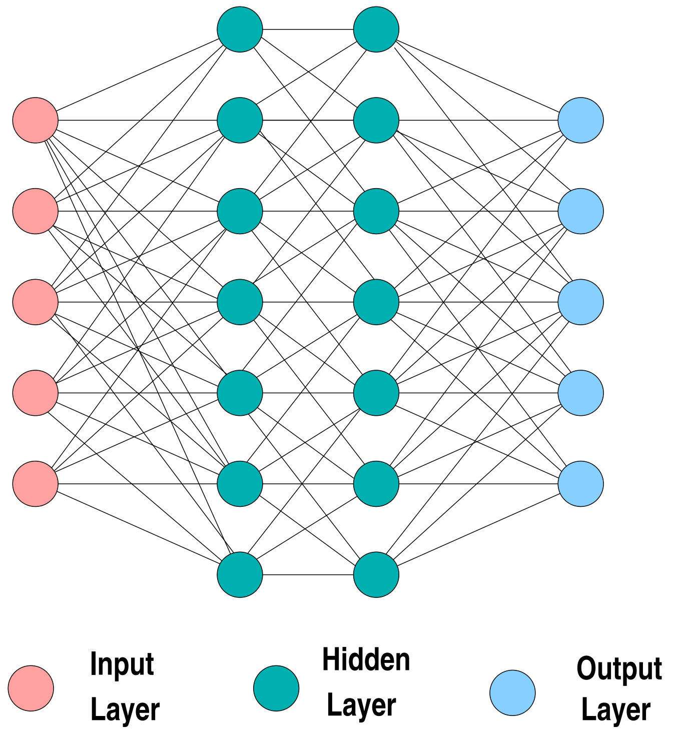
- DNN are remarkably effective in some applications
- big progress made in hardware [$\rightarrow$ affordable ‘training cost’]

➤ Training a neural network can be viewed as a problem of approximating a function ϕ which is defined via sets of parameters:



Problem: find sets of parameters such that $\phi(x) \approx y$

```
Input: $x$, Output: $y$
Set: $z_0 = x$
For $l = 1 : \mathtt{L+1}$ 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

➤ A matrix W_l is associated with layers 1,2, $L + 1$.

➤ Problem:

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

DNN (continued)

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

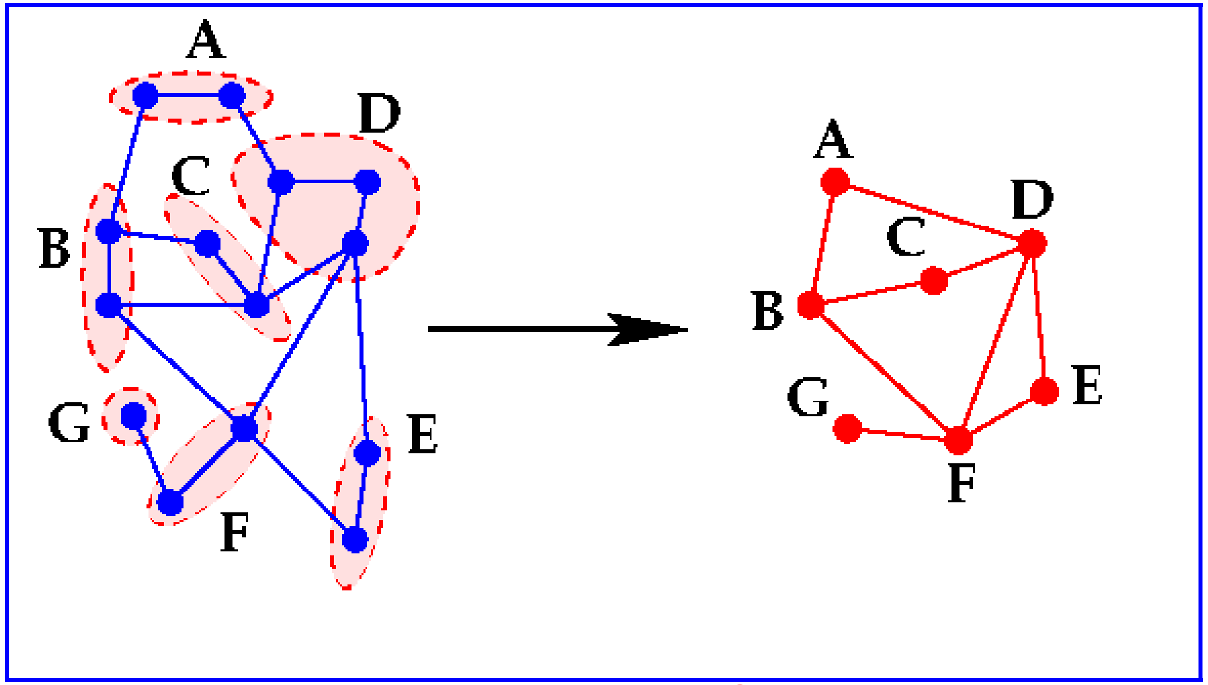
GRAPH COARSENING

Graph coarsening

Given a graph $G = (V, E)$, goal of graph coarsening is to find a smaller graph $G_c = (V_c, E_c)$ with n_c nodes and m_c edges, where $n_c < n$, which is a faithful approximation of G in some sense.

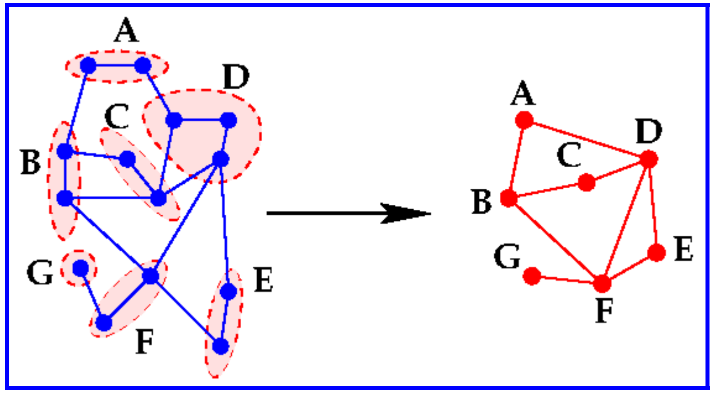
Notation:

- A_c = adjacency matrix of G_c;
- L_c = graph Laplacian of G_c



Graph Coarsening in scientific computing

➤ Goal : exploit coarse representation of problem to solve linear systems

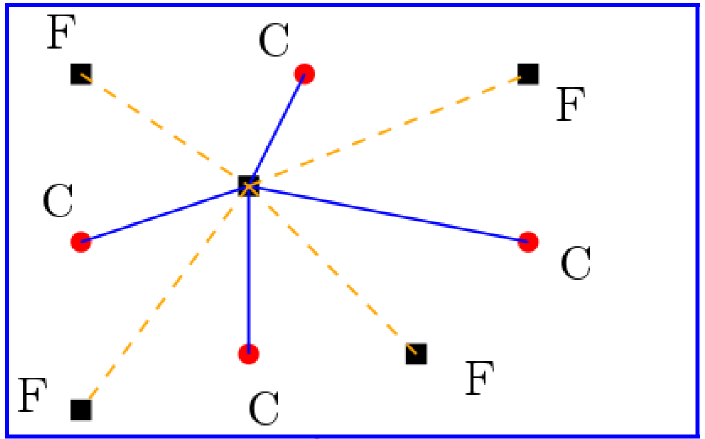


➤ Fewer nodes so: cheaper

➤ Can be used recursively

➤ Success story: *Multigrid, Algebraic Multigrid*

➤ *AMG:* Define coarse / fine nodes based on 'strength of coupling' $\rightarrow$



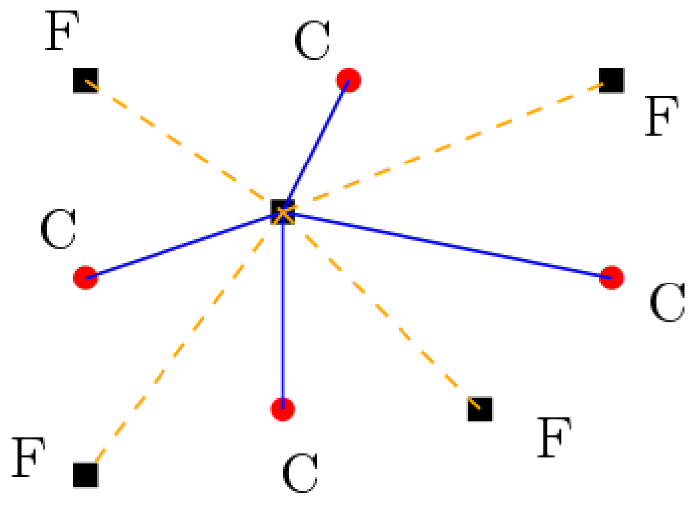
Graph coarsening in scientific computing: (A) MG

Algebraic multigrid Main idea: generalize the *interpolation* and *restriction* operations of standard MG.

➤ For each fine node select a set of nearest neighbors from the coarse set

➤ Then express a fine node i as a linear combination of a selected number of nearest neighbors that form a set C_i:

Fine nodes: ■. Coarse: ● In coarsening: central fine node is expressed as a combination of its coarse neighbors.



- Classical Ruge-Stüben strategy: selection based on ‘strong connections’ of node (i and j are strongly connected if a_{ij} has a large magnitude relative to others)
- Let $C ==$ set of coarse nodes; $F ==$ set of fine nodes
- Can define ‘interpolation operator’ P:

$$[Px]_i = \begin{cases} x_i & \text{if } i \in C, \\ \sum_{j \in C_i} p_{ij} x_j & \text{otherwise.} \end{cases}$$

- Expand into a multilevel framework by repeating the process on the graph of coarse set.

- Let $G_0 \equiv G$ (orig.) and $G_1, G_2, \ldots, G_h$ be sequence of coarse graphs: $G_\ell = (V_\ell, E_\ell)$ is obtained by coarsening on $G_{\ell-1}$ for $1 \leq \ell < L$.

- Let $A^{(0)} \equiv A$ and $A^{(\ell)} \equiv$ matrix associated of ℓ-th level.

- Linear system at the ℓ-th level, can be reordered as:

$$A^{(\ell)} = \begin{bmatrix} A_{CC}^{(\ell)} & A_{CF}^{(\ell)} \\ A_{FC}^{(\ell)} & A_{FF}^{(\ell)} \end{bmatrix}, \quad f^{(\ell)} = \begin{bmatrix} f_C^{(\ell)} \\ f_F^{(\ell)} \end{bmatrix}.$$

- AMG: exploit different levels to building approximate solution. An AMG scheme depends entirely on defining a sequence of interpolation operators P_ℓ for $\ell = 0, 1, \ldots$

- Once the P_ℓ's are defined, one can design various 'cycles' : processes of going back and forth between levels

Multilevel ILU preconditioner based on coarsening

- Method: find a good ordering for ILU preconditioner
- Example: technique presented in [D. Osei-Kuffuor et al, ’06]:
- Ingredient: ordering based on coarsening + apply recursively
- Matrix is ordered in block form - then $A_{22}^{(0)}$ is in turn reordered:

$$\begin{bmatrix} A_{11}^{(0)} & A_{12}^{(0)} \\ A_{21}^{(0)} & A_{22}^{(0)} \end{bmatrix} \quad \rightarrow \quad \left[\begin{array}{c|cc} A_{11}^{(0)} & \multicolumn{2}{c}{A_{12}^{(0)}} \\ \hline A_{21}^{(0)} & \begin{array}{c|c} A_{11}^{(1)} & A_{12}^{(1)} \\ \hline A_{21}^{(1)} & A_{22}^{(1)} \end{array} \end{array}\right] .$$

- Repeat with $A_{22}^{(1)}$ and further down for a few levels.
- Do ILU factorization of the resulting reordered system.

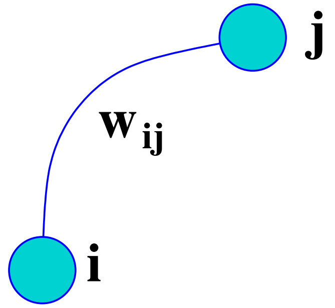
Example: Multilevel ILU [D. Osei-Kuffuor, R. Li, YS, '15]

Goal: Form of ILU preconditioning with improved robustness

➤ 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 $\rightarrow$

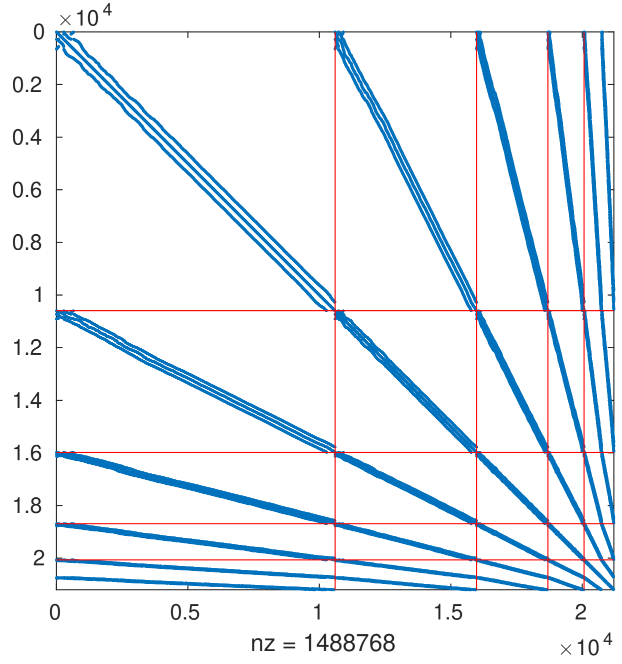
$$\boxed{\sigma_k = \frac{|a_{kk}|}{\delta_r(k)\delta_c(k)}}$$

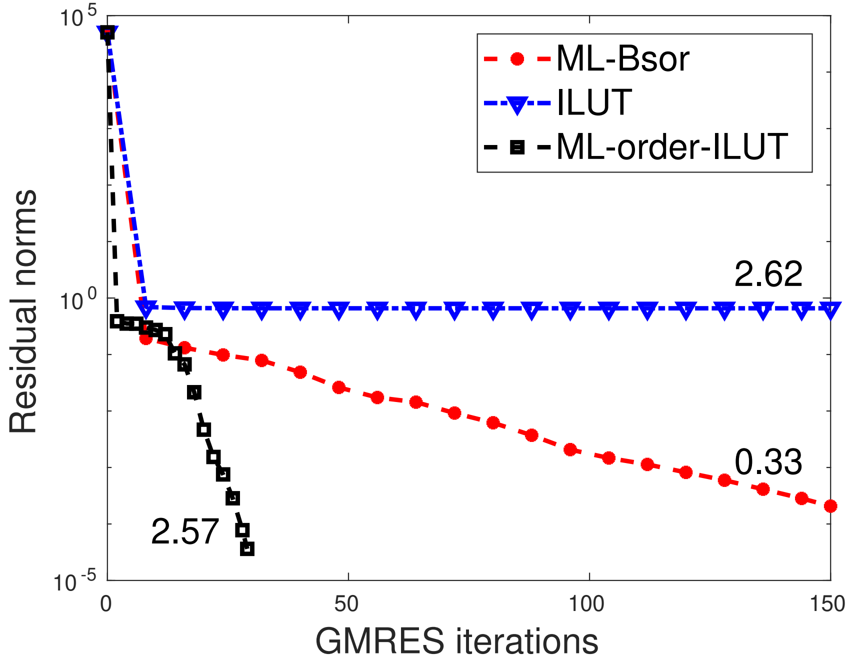
- Goal: to put large entries in the blocks $(A^{(\ell)}_{CF})$ and $(A^{(\ell)}_{FC})$

$$\begin{bmatrix} A^{(\ell)}_{CC} & A^{(\ell)}_{CF} \\ A^{(\ell)}_{FC} & A^{(\ell)}_{FF} \end{bmatrix}$$

- Model: very rough approximation of Gaussian Elimination.
- Next: (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.





Left: The matrix Raefsky3 after the reordering obtained from four levels of coarsening. Right: Performance of various coarsening based preconditioners for solving a linear system with the matrix.

COARSENING APPROACHES

Coarsening by matching: Pairwise aggregation

- Strategy: coalesce (collapse) two adjacent nodes in a graph into a single node, based on some measure of nearness or similarity.
- A *matching* of a graph $G = (V, E)$ is a set of edges $\widetilde{E}$, $\widetilde{E} \subseteq E$, such that no two edges in $\widetilde{E}$ have a node in common.
- Matching is maximal if it cant be augmented by additional edges
- Edge collapsing: usually selected using maximal matching
- Such edge matching techniques are common in AMG literature

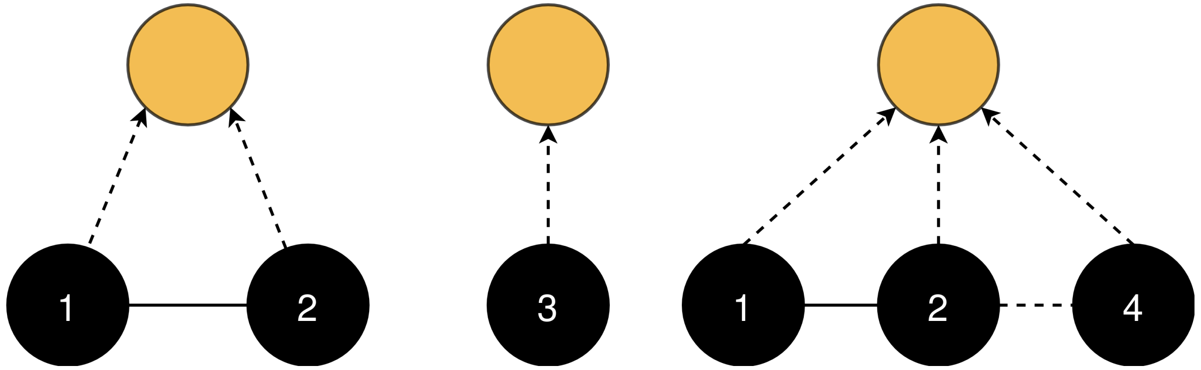
- For each node i, build a set S_i of nodes that are 'strongly connected' to i
- Traverse graph nodes in a certain order of preference
- Next unmarked node in this order, say j, selected as a *coarse* node.
- Priority measure of traversal updated after each insertion of a coarse node

Heavy-edge matching (HEM) : matches a node i with its largest off-diagonal neighbor j_{max};

$$\boxed{|a_{ij_{max}}| = \max_{j \in adj(i), j \neq i} |a_{ij}|}$$

- There will be left-over nodes - called 'singletons'

Heavy Edge Matching (HEM)



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. At completion of traversal: 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

ALGORITHM : 9. Heavy Edge Matching (HEM)

```
Input: Weighted graph $G = (V, E, A)$
Output: Coarse nodes; $Prnt$ list
Init: $Prnt(i) = 0\ \forall i \in V$; $new = 0$
for $\max$ to $\min$ edge $(i, j)$ do
    if $Prnt(i) == 0$, $Prnt(j) == 0$ then
        $new = new + 1$
        $Prnt(i) = Prnt(j) = new$
    end if
end for
for Node $v$ with $Prnt(v) == 0$ do
    if $v$ has no neighbor then
        $new = new + 1$; $Prnt(v) = new$
    else
        $Prnt(v) = Prnt(j)$ where $j = \text{argmax}_i(a_{iv})$
    end if
end for
```

Coarsening by independent sets

Recall: Independent set: $\mathcal{S} \subseteq V$ is a set of vertices that are not adjacent to each other: $\boxed{i, j \in \mathcal{S} \implies a_{ij} = 0}$. It is maximal if it can't be augmented

➤ 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 $Ax = 0$
- 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 $\rightarrow$

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

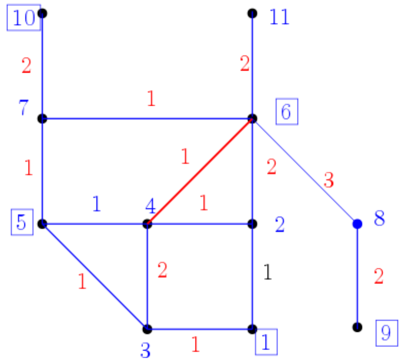
- Kron reduction of L defined as the

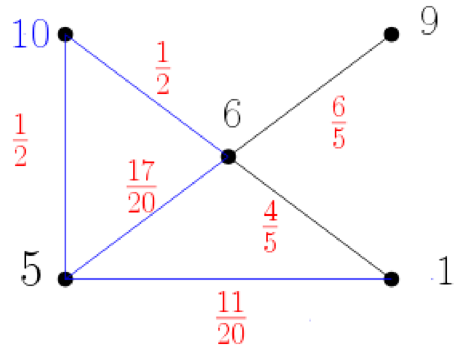
Schur complement:

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

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

Example:





Two ways of using independent sets for coarsening.

Kron coarsening

Independent set coarsening

$$\left[\begin{array}{c|c} D_I & -F \\ \hline -F^T & B \end{array}\right] \longrightarrow \left[\begin{array}{c|c} D_I & -F \\ \hline -F^T & D_f \end{array}\right]$$

$$\downarrow \qquad\qquad\qquad \downarrow$$

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

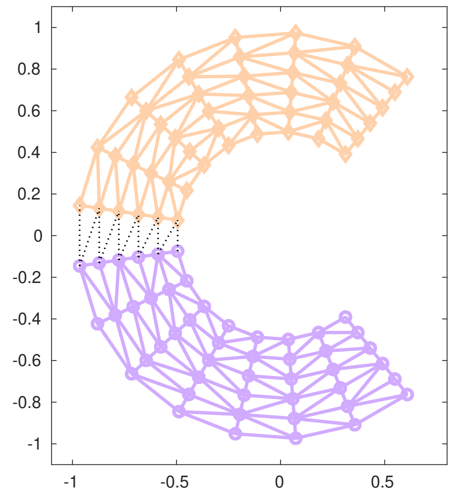
Q. 1: How to deal with ‘denser’ graph?

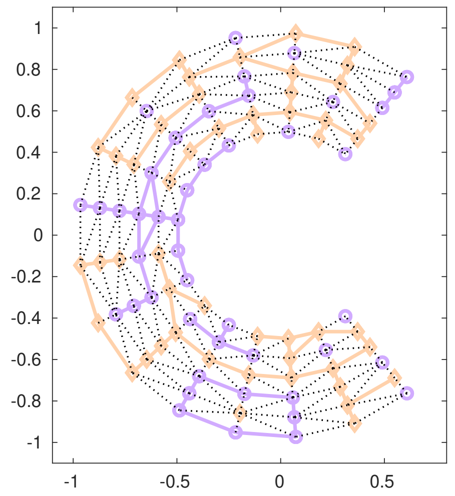
A Sparsify - using spectral sparsificaition

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 withlargest eigen-vector

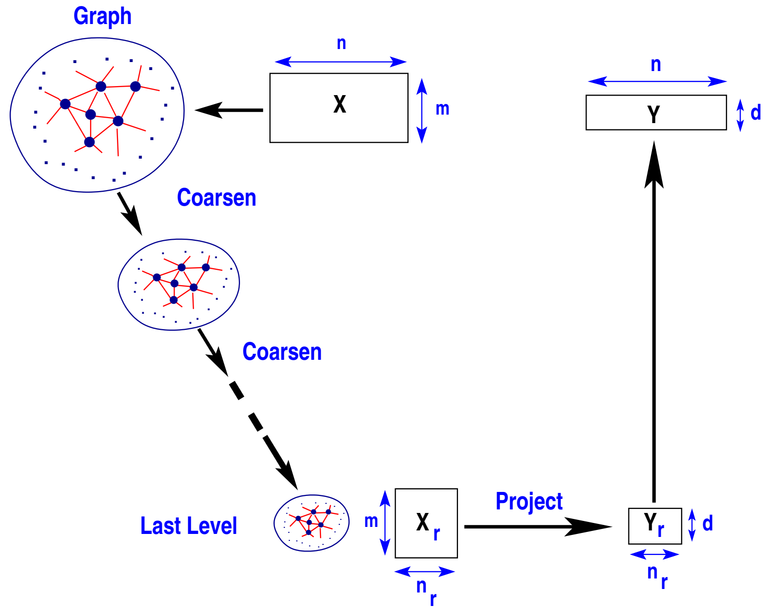
➤ Easy to show: (under mild condition on eigenvector) Each node of V_+ (resp. V_-) must have at least one nearest neighbor node from V_- (resp. V_+).

GRAPH COARSENING IN MACHINE LEARNING

Multilevel Dimension Reduction

Idea:

Coarsen for a few levels. Use resulting data set $\hat{X}$ to find a projector P from $\mathbb{R}^m$ to $\mathbb{R}^d$. Use this 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 $\tan\theta_i \le \epsilon$. Then

$$|x^T AA^T x - x^T CC^T x| \le 3\epsilon \|A\|_F^2,$$

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]

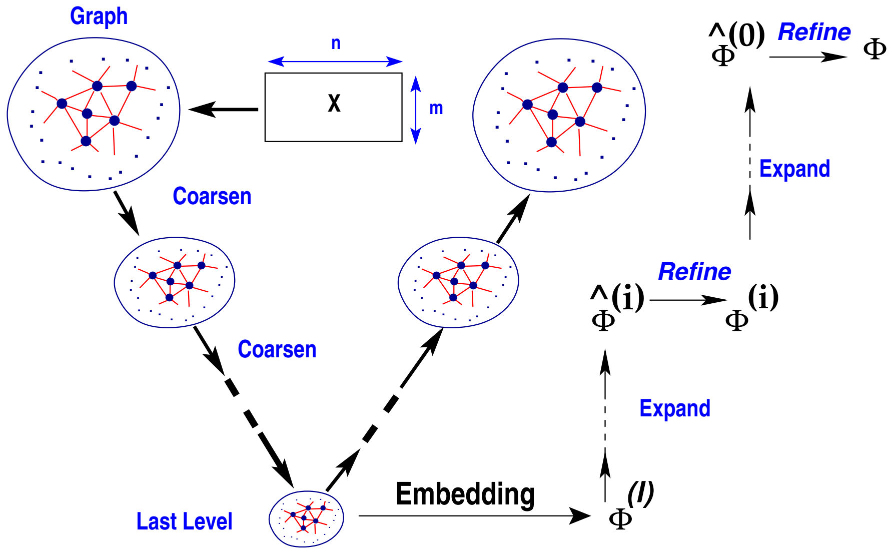
Graph coarsening for graph embeddings: HARP and MILE

- Recall 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

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

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

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

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

● 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 $\lambda_1 \le \lambda_2 \le \cdots \le \lambda_n$

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

Note: 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 $H \equiv \exp(-2\tau L)$

- Next: visualization with 5 different coarsening methods on a graph with $n = 312$ nodes and $ne = 761$ edges

Final words

- *Many* interesting new matrix problems in areas that involve the effective exploitation of data

- Unlike in Forsythe's time: change happens fast - because we are better connected

- In particular: many many resources available online.

- Huge potential for making a good impact by looking at a topic from new perspective

- To a researcher in computational linear algebra : Tsunami of change on types or problems, algorithms, frameworks, culture,..

- My favorite quote. Alexander Graham Bell (1847-1922) said:

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.

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

- More complete version of this material will available in course csci-8314 (S23) - notes (and more) are open to all.

Thank you !