Compressive Sensing, Low Rank models, and Low Rank Submatrix

NICTA Computer Vision Short Course 2012

Yi Li
yi.li@nicta.com.au
http://users.cecs.anu.edu.au/~yili

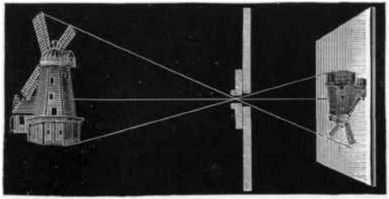
Sep 12, 2012
ver. 1.8
http://tinyurl.com/brl89pk

Outline

Traditional camera

Pinhole model

- Geometry
- Image formation
- Pixelated images
  - Discretization
  - Interpolation

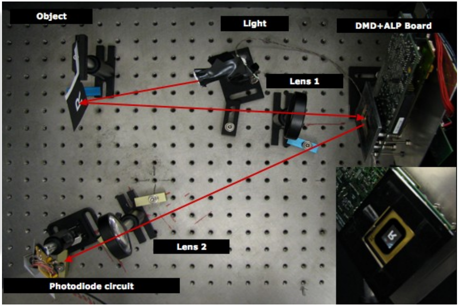


http://chestofbooks.com/arts/photography/Telephotographic-Lens/images/The-Formation-Of-Images-By-The-Pinhole-Camera-And-4.jpg

Single pixel camera

Experimental setting

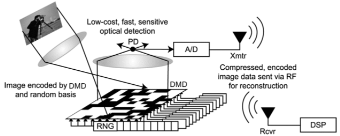
- Random sampling
- Reconstruction
  - # of samples
  - Reconstruction algorithm
- http://dsp.rice.edu/cscamera



How does it work?

Principles

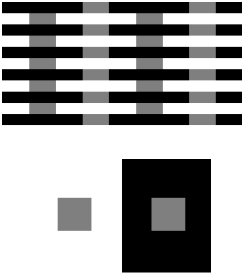
- Random basis $(y_i = sum(R_i(:).*u(:)))$
  - $.*$ is a componentwise multiplication
  - Linear operation $y = Ru$
  - Each row of R is random
- if $u = Dx$ for some D
- Reconstruct x from $y = RDx$



Food for thought

Optical illusion

- Error in reconstruction
  - Can it be used for explaining illusion?
  - Does the explanation fit into human model?
- Implication to visual neuroscience?
- X. Tang and Y. Li, ICIP 2012



Important concepts

- Linear equations
- Rank, trace, and norms
- Eigenvalues/Eigenvectors and Singular Value Decomposition

$y = Ax$

```
%% solve y=Ax

A = rand(3);        %% creating a random matrix.
r = rank(A);        %% full rank?

y = rand(3,1);

x = A\y;            %% one way of solving this: least square
x = inv(A)*y;       %% unique solution if A is full rank

y - Ax              %% verify the correctness
```

Underdetermined and overdetermined

`%% solve y=Ax`
 	
`A = rand(3, 4);`
`r = rank(A);`
 	
`y = rand(3,1);`
 	
`x = A\y;`
`x = inv(A)*y;   %% ??`
 	
`y-A*x`
	`%% solve y=Ax`
	
	`A = rand(4, 3);`
	`r = rank(A);`
	
	`y = rand(4,1);`
	
	`x = A\y;`
	`x = inv(A)*y;   %% ??`
	
	`y-A*x`

Rank: the concept

- Matrix $A_{m \times n}$
- Column rank
  - the maximum number of linearly independent column vectors of A
- Row rank
  - the maximum number of linearly independent row vectors of A
- Column rank $==$ row rank
  - $\leq$min(m,n)

Properties

Two matrices A and B

- rank(AB)≤min(rank(A),rank(B))
- rank(A+B)≤rank(A)+rank(B)
- rank(A^TA)=rank(AA^T)=rank(A)=rank(A^T)
- row-echelon forms
  - Ae = rref(A) in matlab

Questions

Implication in computer vision

- Background pixels over time
- Multiple part tracking
- Image matching
- Your nomination?

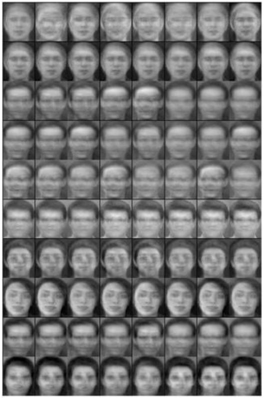
Eigenvalue and Eigenvector

- Matrix $A_{n \times n}$
- $Av = \lambda v$
  - Same eigenvalue may have multiple eigen vectors
  - zero eigenvalue?
- Matlab
  - eig(A)
- Each column in A can be represented by a linear combination of eigenvectors

PCA in computer vision

Eigenface

- Each face is a linear vector
  - Concatenate columns
  - Faces are usually aligned
- Eigenvector = basis
- http://www.umiacs.umd.edu/~knkim/



Singular vector decomposition

- Matrix $A_{m\times n}$
- Factorize A to $A = U\sum V^T$, where
  - U is $m \times m$ unitary matrix
  - $\sum$ is a $m \times n$ diagonal matrix
  - V is $n \times n$ unitary matrix
- Matlab
  - [u d v] = svd(A)
- Why we need Singular Value, if we already have Eigenvalue?

SVD

- SVD works for arbitrary matrix $A_{m \times n}$
- $A = U \sum V^T$ means:
  - U and V are orthonormal basis
  - $\sum$ is the singular value of A
  - Can be used for pseudo-inverse: proof $A^{-1} = V \sum^{-1} U^T$
  - Proof columns of V are the eigen vector of $A^T A$ (homework)
  - Consequently, $\sum$ is the eigenvalue of $A^T A$
  - How about U ?
- Low rank approximation

Trace

- Matrix $A_{n \times n}$
- Simple definition
  - $tr(A) = a_{11} + .. + a_{ii} + ... + a_{nn}$
- Linkage to Eigenvalue
  - $tr(A) = sum(eig(A))$
- Invariant to the change of basis!

Properties

- $\mathrm{tr}(A+B) \leq \mathrm{tr}(A) + \mathrm{tr}(B)$
- $\mathrm{tr}(A) = \mathrm{tr}(A^T)$
- $\mathrm{tr}(A^T B) = \mathrm{tr}(BA^T)$
- $\mathrm{tr}(A^T B)$, "inner product" of A and B
- $\mathrm{tr}(ABC) = \mathrm{tr}(BCA) = \mathrm{tr}(CBA)$

L_2 norm

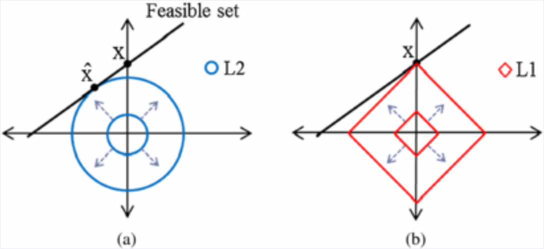
"Least square"

- Case 1: Line fitting
  - a few pairs (x_i, y_i), or simply (x, y)
  - $\beta = (x^T x)^{-1} x^T y$
- Case 2: Signal-to-noise rate in signal processing
  - decibel (dB)
- Matlab: norm(x,2)

L_1 norm

Sum of absolute values

- In many cases: $Dist = \sum |x|$
- Example: $Dist = |x_1 - x_2| + |y_1 - y_2|$ ("Manhattan Distance")
- Why are the differences?:

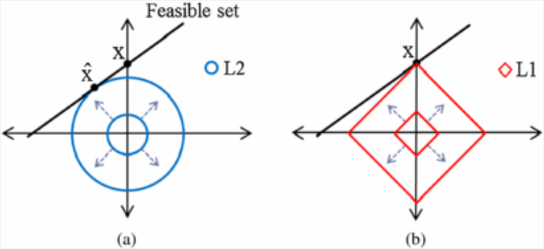


- online figure

L_1 norm

Sum of absolute values

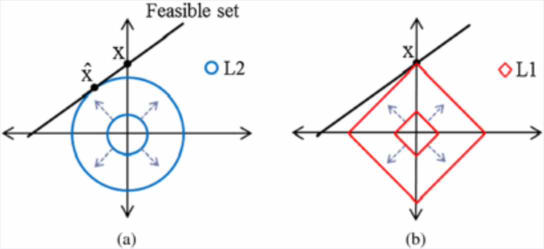
- In many cases: $Dist = \sum |x|$
- Example: $Dist = |x_1 - x_2| + |y_1 - y_2|$ ("Manhattan Distance")
- Why are the differences?:



online figure

L_1 norm

Sum of absolute values

- In many cases: $Dist = \sum |x|$
- Example: $Dist = |x_1 - x_2| + |y_1 - y_2|$ ("Manhattan Distance")
- Why are the differences?:
- 

- online figure

L_0 norm

"Count of non-zero values"

- Ideal definition for measuring the sparseness of a vector
- Problem:
  - Very difficult optimization, NP complete
  - Card(x) in constraints, or minimize the set size of the non-zero variable
  - Need approximation in many practical problems

Matlab practice (and Q/A)

Practice

- Generate two 2×2 random matrices A and B
- Use bilinear interpolation to resize them to 10×10
- Calculate the rank(A)
- trace($A^T B$)=trace($B^T A$)=sum of A.*B
- show norm(A(:), 2), norm(A(:), 1), and norm(A(:), 0)
- Generate a 10×10 random matrix C
- Compare eig(A) and eig(C)

Matlab practice (and Q/A)

Practice

- Generate two 2×2 random matrices A and B
- Use bilinear interpolation to resize them to 10×10
- Calculate the rank(A)
- trace($A^T B$)=trace($B^T A$)=sum of A.*B
- show norm(A(:), 2), norm(A(:), 1), and norm(A(:), 0)
- Generate a 10×10 random matrix C
- Compare eig(A) and eig(C)

Matlab practice (and Q/A)

Practice

- Generate two 2×2 random matrices A and B
- Use bilinear interpolation to resize them to 10×10
- Calculate the rank(A)
- trace($A^T B$)=trace($B^T A$)=sum of A.*B
- show norm(A(:), 2), norm(A(:), 1), and norm(A(:), 0)
- Generate a 10×10 random matrix C
- Compare eig(A) and eig(C)

Matlab practice (and Q/A)

Practice

- Generate two 2×2 random matrices A and B
- Use bilinear interpolation to resize them to 10×10
- Calculate the rank(A)
- trace($A^T B$)=trace($B^T A$)=sum of A.*B
- show norm(A(:), 2), norm(A(:), 1), and norm(A(:), 0)
- Generate a 10×10 random matrix C
- Compare eig(A) and eig(C)

Matlab practice (and Q/A)

Practice

- Generate two 2×2 random matrices A and B
- Use bilinear interpolation to resize them to 10×10
- Calculate the rank(A)
- trace($A^T B$)=trace($B^T A$)=sum of A.*B
- show norm(A(:), 2), norm(A(:), 1), and norm(A(:), 0)
- Generate a 10×10 random matrix C
- Compare eig(A) and eig(C)

Matlab practice (and Q/A)

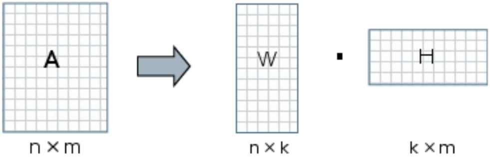
Practice

- Generate two 2×2 random matrices A and B
- Use bilinear interpolation to resize them to 10×10
- Calculate the rank(A)
- trace($A^T B$)=trace($B^T A$)=sum of A.*B
- show norm(A(:), 2), norm(A(:), 1), and norm(A(:), 0)
- Generate a 10×10 random matrix C
- Compare eig(A) and eig(C)

Matlab practice (and Q/A)

Practice

- Generate two 2×2 random matrices A and B
- Use bilinear interpolation to resize them to 10×10
- Calculate the rank(A)
- trace($A^T B$)=trace($B^T A$)=sum of A.*B
- show norm(A(:), 2), norm(A(:), 1), and norm(A(:), 0)
- Generate a 10×10 random matrix C
- Compare eig(A) and eig(C)

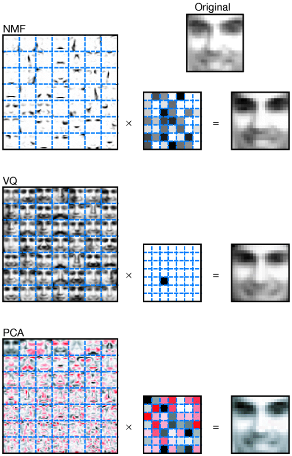
- Goal: $X = WH$
- If k $<<$ min(m,n)
  - Extreme case: rank(W)=1
  - Meaning?
- Constraints:
  - 

Interpretation

- Rewrite $X = WH$ as $X_{:,i} = \sum_{j=1}^{n} H_{ji} W_{:,j}$
  - $W_{:,j}$ can be considered as a basis function
  - $X_{:,i}$ is in the space spanned by W
  - Columns of W are not necessarily orthogonal
- Recall PCA
  - What are the similarities?
  - What are the differences?

Application
Face recognition

- Decomposing faces into parts
- Basis of objects
  - Orthogonal (Eigenface)
  - non-orthogonal (NMF)
- W can be regarded as face parts
- H can be regarded as weights for combing basis functions



Approach

- L_2 norm between X and WH
  - $\min |X - WH|_2$
- Subject to?
  - $W \geq 0,\ H \geq 0$
- Non-convex

Quick detour

Coordinated Descent

- $z = f(x)g(y)$
  - maybe non-convex -> local minima
- Fix $x = x_0$, $z = f(x_0)g(y) = \bar{g}(y)$
  - If $z = \bar{g}(y)$ is convex, unique solution y_1
- Do the same thing for $z = g(y_1)f(x)$ until converge or after certain number of iterations.

Solution

Coordinated Descent

- Random initialize W and H
- Iteratively solve $|X - WH|_2$
    - $|X - WH|_2$ given H
    - $|X - WH|_2$ given W
- Update until converge

Iterative Update Rules

Coordinated Descent

- Random initialize W and H
- The Euclidean distance $|X - WH|_2$ is nonincreasing under the update rules
  - $W_{ia} = W_{ia} \sum \frac{X_{i\mu}}{(WH)_{i\mu}} H_{a\mu}$ and normalize W for each column.
  - $H_{a\mu} = H_{a\mu} \sum W_{ia} \frac{X_{i\mu}}{(WH)_{i\mu}}$
- Update until converge

Interpretation

- L_2 norm between X and WH
  - Gaussian distribution.
  - KL divergence ($D(p||q) = \sum p_i \ln \frac{p_i}{q_i}$)
  - L_1 distance
- Basis functions (W) are not orthogonal
  - Good or not?
- Variations
  - $X = WSH$, where S can be used for controlling smoothness

Matlab experiments (15 mins)

Practice

- Generate a 2×2 random matrices A
- Resize it to 10×10
- Random initialize matrix $W_{10\times3}$ and $H_{10\times3}$
- Use iterative update rule $W_{ia} = W_{ia} \sum \frac{X_{i\mu}}{(WH)_{i\mu}} H_{a\mu}$ and $H_{a\mu} = H_{a\mu} \sum W_{ia} \frac{X_{i\mu}}{(WH)_{i\mu}}$
- Define your convergence criteria.

Matlab experiments (15 mins)

Practice

- Generate a 2×2 random matrices A
- Resize it to 10×10
- Random initialize matrix $W_{10\times 3}$ and $H_{10\times 3}$
- Use iterative update rule $W_{ia} = W_{ia} \sum \frac{X_{i\mu}}{(WH)_{i\mu}} H_{a\mu}$ and $H_{a\mu} = H_{a\mu} \sum W_{ia} \frac{X_{i\mu}}{(WH)_{i\mu}}$
- Define your convergence criteria.

Matlab experiments (15 mins)

Practice

- Generate a 2×2 random matrices A
- Resize it to 10×10
- Random initialize matrix $W_{10\times 3}$ and $H_{10\times 3}$
- Use iterative update rule $W_{ia} = W_{ia} \sum \frac{X_{i\mu}}{(WH)_{i\mu}} H_{a\mu}$ and $H_{a\mu} = H_{a\mu} \sum W_{ia} \frac{X_{i\mu}}{(WH)_{i\mu}}$
- Define your convergence criteria.

Matlab experiments (15 mins)

Practice

- Generate a 2×2 random matrices A
- Resize it to 10×10
- Random initialize matrix $W_{10\times 3}$ and $H_{10\times 3}$
- Use iterative update rule $W_{ia} = W_{ia} \sum \frac{X_{i\mu}}{(WH)_{i\mu}} H_{a\mu}$ and $H_{a\mu} = H_{a\mu} \sum W_{ia} \frac{X_{i\mu}}{(WH)_{i\mu}}$
- Define your convergence criteria.

Matlab experiments (15 mins)

Practice

- Generate a 2×2 random matrices A
- Resize it to 10×10
- Random initialize matrix $W_{10\times3}$ and $H_{10\times3}$
- Use iterative update rule $W_{ia} = W_{ia} \sum \frac{X_{i\mu}}{(WH)_{i\mu}} H_{a\mu}$ and $H_{a\mu} = H_{a\mu} \sum W_{ia} \frac{X_{i\mu}}{(WH)_{i\mu}}$
- Define your convergence criteria.

Discussion

- How to use NMF in your projects?
- Do you buy it?
- Yes or No, what did you learn?

Recall $y = Ax$

Detail matters

- If A is orthonormal (e.g., in PCA)
- if A is full rank, $x = A^{-1}y$
- However, $y = Ax$ can be underdetermined
  - Well known in undergrad studies: many solutions
  - Less known: minimizing $\sum |x|_2$
- What happens if we minimizing $\sum |x|_0$ or $\sum |x|_1$?
  - Discussion: meaning?

Dictionary: the concept

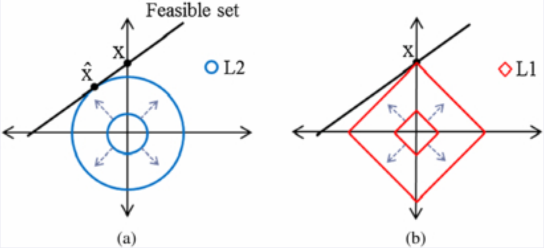
- In sparse representation, we call A a "dictionary"
- Assume A is given
- We further call x as coefficients
- Now we want to solve $y = Ax$ s.t. minimizing $\sum |x|_0$

Minimizing the cardinality of coefs

Discussion

- What are the advantages?
  - An ideal solution of many problems
- Problems?
  - NP complete
  - We need approximation

Approximation: Minimizing L_1 norm

- Sparseness
- Convex problem
- Recall
  - 

  - Stable and accurate results (for cases where coefs are sparse)

Solver 0: Orthogonal Matching Persuit (OMP)

OMP

- Idea: sequentially pick the basis.
  - Greedy algorithm
- For each basis function, calculate the error
  - $v_i = \arg\min y - A_{:,i} v_i$ for each i, where v_i denotes an all zero vector except the i^{th}element
  - Pick the coefs with minimal fitting error
- let $y = y - A_{:,j} v_j$ repeat the procedure for the remaining basis

Solver 1: Softthresholding

- Solve $y = Ax$ in L_2 norm
- Soft thresholding
  - $S_\lambda(x) = x - 0.5\lambda$ if $x > 0.5\lambda$
  - $S_\lambda(x) = x + 0.5\lambda$ if $x < -0.5\lambda$
  - $S_\lambda(x) = 0$, o.w.

Solver 1: Pros and Cons

Does it solve all problems?

- Very efficient
- Only solves $\sum |x|_1 + \lambda |y - Ax|_2$
- How about $\sum |Bx|_1 + \lambda |y - Ax|_2$?

Solver 2: Bregman iteration

- $\min J(x) + H(x)$
  - $J(x)$ is continuous but not differentiable
  - $H(x)$ is continuous and differentiable
- Introduce $|d - Bx|_2$ and $E(x, d) = |d|_1 + H(x)$
  - $\min E(x, d) + \frac{\lambda}{2}|d - Bx|_2$

Solver 2: Iterative update

- $x^{k+1} = \arg\min H(x) + \lambda/2|d^k - Bx - p^k|_2$
  - L_2norm: least square
- $d^{k+1} = \arg\min |d|_1 + \lambda/2|d - Bx^{k+1} - p^k|_2$
  - Soft thresholding
- $p^{k+1} = p^k + Bx^{k+1} - d^{k+1}$
  - Simple numerical operation

Matlab experiment (20 minutes)

- Randomly generate an orthonormal matrix $A_{10\times 10}$ (how?)
- Randomly generate $y_{10\times 1}$
- hint: each column of A is a basis function
- Assuming we want x that has only 3 non-zero coefs to approximate $y = Ax$
  - Use OMP
  - Use soft thresholding

How about unknown A?

Sparse coding

- $Y = AX$, where both A and X are unknown.
  - Y is a matrix, because we need more than one *observation* to learn the underlying dictionary
- Coordinated descent
  - Given A , solve X (we know!)
  - Given X , solve A

What does Low Rank mean?

Idea: correlation

- Redundancy
- Accurate representation
- Reduce the problems caused by noise

Modeling

Matrix A can be approximated by X+E

- Low rank X
  - Example: human motion capture data
- Sparse noise E
  - Example: occlusion
- Formulation
  - $\min rank(L) + \lambda |E|_1$
  - s.t. $A = L + E$

Norm: trace norm

Definition

- Recall: trace is the sum of eigenvalue
- minimizing trace norm
- $|A|_* = tr((A^T A)^{1/2})$

Putting everything together

Formulation

- min $tr((L^T L)^{1/2}) + \lambda |E|_1$
- s.t. $A = L + E$

Solver: Alternating Direction Method of Multiplier (ADMM)

Method of multipliers

- $\min f(x)$ s.t. $Ax = b$
- Lagrangian: $L(x, y) = f(x) + y^T(Ax - b)$
- Augmented $L_\rho(x, y) = f(x) + y^T(Ax - b) + \frac{\rho}{2}|Ax - b|_2$
  - $x^{k+1} = \arg\min L_\rho(x, y^k)$
  - $y^{k+1} = y^k + \rho(Ax^{k+1} - b)$
- Problem: how about $f(x) = \sum |Bx|_1 + \lambda|y - Ax|_2$?

ADMM

ADMM

- $\min f(x) + g(z)$ s.t. $Ax + Bz = c$
- Augmented
  $L_\rho(x, z, y) = f(x) + g(z) + y^T(Ax + Bz - c) + \frac{\rho}{2}|Ax + Bz - c|_2$
  - $x^{k+1} = \arg\min L_\rho(x, z^k, y^k)$
  - $z^{k+1} = \arg\min L_\rho(x^{k+1}, z, y^k)$
  - $y^{k+1} = y^k + \rho(Ax^{k+1} + Bz^{k+1} - c)$
- Key idea: separate x and z
- Problem: This is a so called "two term admm". It is not clear any separation higher than 2 terms will converge
  - empirically yes!

Problem of Low Rank matrix?

X is the low rank version of A

- Only a subset of features correlated
  - DNA
  - Data mining
- Noise is not sparse

Solution: finding LR submatrix directly

Random projection

- "Binarization" of a matrix A
- $B = sign(A - mean(A(:)))$

An (extremely fast) method for detecting lr submatrix

Loop: the concept

- Take any 2×2 submatrix $[B_{ij}, B_{ij'}; B_{i'j}, B_{i'j'}]$ of B
- Take the product $p = B_{ij}B_{ij'}B_{i'j}B_{i'j'}$
  - p=-1 if $[B_{ij}, B_{ij'}; B_{i'j}, B_{i'j'}]$ is rank 2
  - p=1 if $[B_{ij}, B_{ij'}; B_{i'j}, B_{i'j'}]$ is rank 1
- Fix i, and test its "similarity" with other rows
  - Sum all loops $Z = \sum_j \sum_{j'} \sum_{i'} B_{ij}B_{ij'}B_{i'j}B_{i'j'} = [BB^TBB^T]_{ii}$
  - Practice: verify $\sum_j \sum_{j'} \sum_{i'} B_{ij}B_{ij'}B_{i'j}B_{i'j'} = [BB^TBB^T]_{ii}$

Procedure

Algorithm

- Calculate $Z_{row} = [BB^TBB^T]_{ii}$
- Sort Z_{row}
- Truncate the bottom $p\%$ rows
- Calculate $Z_{col} = [B^TBB^TB]_{jj}$
- Sort Z_{col}
- Truncate the bottom $p\%$ cols
- until max number of iterations

Does it work? Matlab experiments (20 mins)

LR Submatrix

- Generate a 2×2 random matrices A_1
- Use bilinear interpolation to resize it to 20×20
- Generate a 50×50 random matrices A_2
- Randomly embed A_1 to A_2
- binarize A_2 to 1/-1 and run the procedure
- Visualize the results for each iteration

Discussion: multiple submatrix?

- How can we find multiple submatrices?

Take home message

Take home message

- Linear algebra is important
- Sparseness is useful
- Low rank models are effective

Homework

- Download a face dataset from http://tinyurl.com/bpdduaj
  - Each column is a face (165×120), and each row is a pixel location
  - Visualize the first 10 faces in this dataset (hint: reshape()).
- Problem 1: Use all the faces to compute the Eigenfaces of this dataset
  - You define the number of eigenvectors
  - You must visualize all the eigenfaces and the reconstruction errors
- Problem 2: Use the eigenfaces as the dictionary, use L_1 minimization to approximate each face
  - You define the number of non-zero coefs
  - You must visualize the reconstruction errors and compare them to the errors in the Problem 1
- Problem 3: Find the first low rank submatrix in this dataset
  - Truncate rows only, for simplicity
  - Recall that each row is a pixel location, visualize the submatrix in the original image space for the first 10 faces
  - Explain what is the common feature in this dataset

Requirement

- You must use MATLAB and / or C++
- No team work
- You must hand in a zip file that has
  - Your code and a readme file, explaining how to run it
  - a report that
    - 1) describes your experiments comprehensively;
    - 2) presents your results neatly; and
    - 3) must include reasonable discussion;