It Is All Based on Linear Algebra !

Matrix Decomposition Techniques for Image Analysis

Désiré Sidibé

Assistant Professor - Université de Bourgogne

LE2I - UMR CNRS 6306

dro-desire.sidibe@u-bourgogne.fr

27/04/2016

Outline

Outline

- An image is represented as a matrix

$$I = \begin{bmatrix} \cdots & \cdots \\ \vdots & \vdots \\ \cdots & \cdots \end{bmatrix}_{m \times n}$$

- A video can either be represented as a set of matrices or a 3D tensor

Importance

Linear algebra (Matrix properties and calculations) is a fundamental tool

Introduction

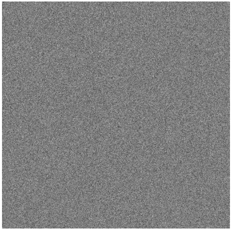
- Consider the image restauration problem :
  - Given an observed noisy image I_n, we want to decompose it into a noise-free image I corrupted by a degradation function G, and a noise component N

$$I_n = GI + N$$

- If, we can solve this decomposition problem, we can get the noise free image.
  - The difficulty is to find the best such decomposition (under reasonable constraints)
- Useful tools includes : PCA, SVD, etc.

Introduction

- Consider the image denoising problem :

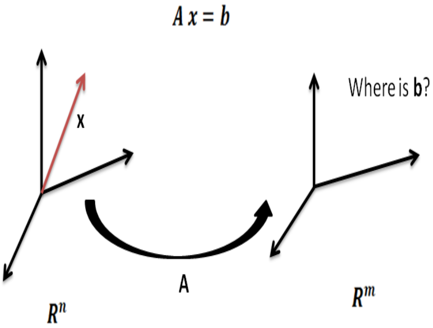


$$I_n \quad = \quad I \quad + \quad N$$

Outline

What is a matrix ?

- A matrix is one way of describing (or representing) a linear transformation between two vector spaces.
- A general $m \times n$ matrix A represents a linear transformation from $\mathbb{R}^n$ to $\mathbb{R}^m$.



The matrix acts on vectors $\mathbf{x} \in \mathbb{R}^n$ to produce vectors $\mathbf{y} \in \mathbb{R}^m$ as $\mathbf{y} = A\mathbf{x}$.

Basic questions

- Does the system $A\mathbf{x} = \mathbf{b}$ has a solution ?
- If yes, how many solution(s) ?
- How to find the solution(s) ?

For example, can we solve the following system ?

$$\begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} \mathbf{x} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

How many solutions, if any ?

Column space and nullspace

Column space

The *column space* of A, denoted by $C(A)$ and also called *range* or *span* of A, is the subspace of $\mathbb{R}^m$ such that :
$y \in C(A)$ if and only if $y = Ax$ for some $x \in \mathbb{R}^n$.

Nullspace

The *nullspace* of A, denoted by $N(A)$ and also called *kernel*, is the subspace of $\mathbb{R}^n$ such that :
$x \in N(A)$ if and only if $Ax = 0$.

- $C(A)$ is equals to the set of all linear combinations of the columns of A
- $N(A)$ is exactly the set of vectors which are orthogonal to all the row vectors of A.

Rank of a matrix

Rank

The *rank of a matrix* is the dimension of its column space.

$$rank(A) \doteq dim(C(A)).$$

- The *rank* is the most fundamental notion about a matrix
- The rank of A is equal to the maximum number of linearly idependent columns (or rows) of A
- What are the rank of the following matrices ?

$$\begin{bmatrix} 1 & 2 \\ 2 & 1 \end{bmatrix} ; \begin{bmatrix} 1 & 2 \\ 1 & 2 \end{bmatrix}$$

Rank of a matrix

Rank theorem

if A is an $m \times n$ matrix, then $rank(A) + \dim N(A) = n$.

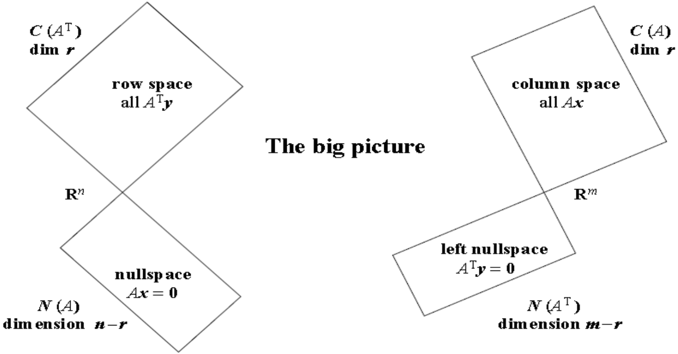


FIGURE : The big picture of linear algebra (from G. Strang)

Solving linear systems

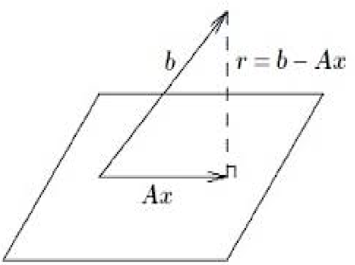
The main problem in linear algebra : solve $A\mathbf{x} = \mathbf{b}$

- One can solve $A\mathbf{x} = \mathbf{b}$ iff $\mathbf{b} \in C(A)$
- The rank of A tells everything

TABLE : A is $m \times n$ matrix of rank r

$r = m = n$	$A\mathbf{x} = \mathbf{b}$ has a unique solution
$r = n < m$	$A\mathbf{x} = \mathbf{b}$ has either 0 or a unique solution
$r = m < n$	$A\mathbf{x} = \mathbf{b}$ has ∞ many solutions
$r < m,\ r < n$	$A\mathbf{x} = \mathbf{b}$ has either 0 or ∞ solutions

What if b $\notin C(A)$?



Find $\mathbf{x} \in \mathbb{R}^n$ such that
$\| r \|^2 = \| A\mathbf{x} - \mathbf{b} \|^2$ is minimum.

Linear Least Squares (LLS)

- Project **b** onto $C(A)$, and solve $A\hat{\mathbf{x}} = p$
- The "best" (minimum mean square error) is solution to the **normal equation :** $A^T A\hat{\mathbf{x}} = A^T\mathbf{b}$
- If $A^T A$ is invertible, then the LLS solution is given by

$$\hat{\mathbf{x}} = (A^T A)^{-1} A^T \mathbf{b}$$

Eigen-decomposition

Eigenvalues/Eigenvectors

Given a square $n \times n$ matrix A, we say that $\lambda \in \mathbb{C}$ is an *eigenvalue* of A and $\mathbf{x} \in \mathbb{C}$ in the corresponding *eigenvector* if

$$A\mathbf{x} = \lambda\mathbf{x}, \ \mathbf{x} \neq 0.$$

Properties of eigenvalues

- The rank of A is equal to the number of non-zero eigenvalues.
- If A is a non-singular matrix (all of its eigenvalues are non-zero) then $1/\lambda_i$ is an eigenvalue of A^{-1} with associated eigenvector $\mathbf{x}_i$.

Properties of eigenvalues

- The sum of the eigenvalues of A is equal to its trace

$$\text{trace}(A) = \sum_{i=1}^{n} A_{ii} = \sum_{i=1}^{n} \lambda_i.$$

- The determinant of A is equal to the product of its eigenvalues

$$\det(A) = |A| = \prod_{i=1}^{n} \lambda_i.$$

Eigen-decomposition

Properties of eigenvalues

- Different eigenvalues ⇒ linearly independent eigenvectors

$$\lambda_i \neq \lambda_j \Rightarrow \mathbf{x}_i \text{ and } \mathbf{x}_j \text{ are independent}$$

- If A has n different eigenvalues, then A can be diagonalized as

$$A = S\Lambda S^{-1} = [\mathbf{x}_1, \ldots, \mathbf{x}_n] \begin{pmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_n \end{pmatrix} [\mathbf{x}_1, \ldots, \mathbf{x}_n]^{-1}$$

- Powers of A are easily obtained as $A^k = S\Lambda^k S^{-1}$
  - useful to solve recurrent equations such as $u_{k+1} = Au_k$
  - useful to exponentiate the matrix : $e^A = \sum_{k=0}^{\infty} \frac{A^k}{k!}$
- If A is symmetric, then we can write $A = S\Lambda S^T$
- If the eigenvalues of A are not all different, it may or may not be possible to diagonalize A.

Singular value decomposition

SVD : generalization of eigenvalues/eigenvectors concept for non-square matrices

Any general $m \times n$ matrix A of rank r can be decomposed as

$$A = U \Sigma V^T$$

with

- U an orthogonal $m \times m$ matrix : $UU^T = I$
- Σ a diagonal $m \times r$ matrix : $\Sigma = \begin{pmatrix} \sigma_1 & & & \\ & \ddots & & \\ & & \sigma_r & \\ & & & 0 \end{pmatrix}$
- V an orthogonal $n \times n$ matrix : $VV^T = I$

Singular value decomposition

Any general $m \times n$ matrix A can be decomposed as : $A = U\Sigma V^T$

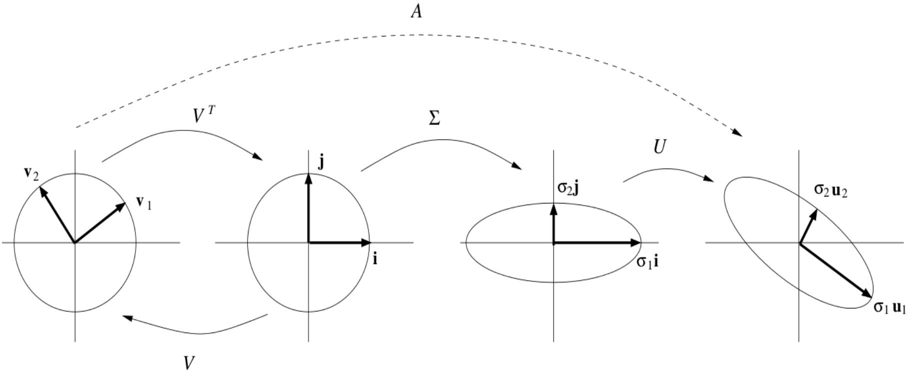


FIGURE : Geometric interpretation of SVD.

The usefulness of SVD

Probably the most important tool.

$$A = U\Sigma V^T$$

- **Solving linear systems :** $A\mathbf{x} = b$
  $\widehat{\mathbf{x}} = A^{\pm}b$, where $A^{\pm}$ is the pseudo-inverse of A given by

  $$A^{\pm} = V \,\mathrm{diag}(1/\sigma_1, \ldots, 1/\sigma_r)\; U^T$$

- **Solving homogeneous systems** : $A\mathbf{x} = 0$
  $\widehat{\mathbf{x}}$ = the right singular vector corresponding to the smallest sigular value.
  $\widehat{\mathbf{x}} =$ V(:, end), in MATLAB notation.
- **Approximating a matrix** A
  The best rank k approximation of A is $\widehat{A} = \sum_{i=1}^{k} \sigma_i u_i v_i^T$.
- Many more ...

SVD applications

SVD is a fundamental tool for data analysis and is often used in computer vision and machine learning applications

- Image compression
- Image denoising
- Pattern classification
- Transformations estimations
- etc

Image denoising

- A noisy image X can be decomposed as : $A = U\Sigma V^T = \sum_{i=1}^{r} \sigma_i u_i v_i^T$, where each $u_i v_i^T$ is a rank one approximation of X.
- A noiseless approximation of X is obtained by truncating the sum at k terms : $\widehat{X} = \sum_{i=1}^{k} \sigma_i u_i v_i^T$.

$k = 10$

$k = 50$

$k = 100$

FIGURE : Image denoising with SVD.

Image denoising

- It is better to work with local patches
- Denoise each local patch with SVD

$k = 1$ $k = 2$ $k = 10$

FIGURE : Image denoising with SVD on local patches.

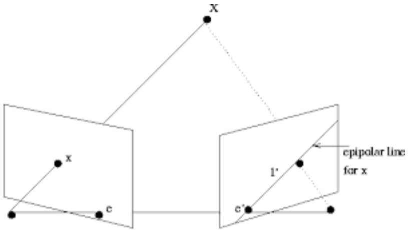
Epipolar geometry

- Epipolar geometry gives a constraint between corresponding points
- if a 3D point $\mathbf{X}$ of the scene is projected onto $\mathbf{x}$ and $\mathbf{x}'$ in the two views, then the image points $\mathbf{x}$ and $\mathbf{x}'$ must satisfy the epipolar constraint :

$$\mathbf{x}^T F \mathbf{x}' = 0,$$

where $\mathbf{x} = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$, $\mathbf{x}' = \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix}$ and $F = \begin{bmatrix} f_{11} & f_{12} & f_{13} \\ f_{21} & f_{22} & f_{23} \\ f_{31} & f_{32} & f_{33} \end{bmatrix}$.

- F is called the fundamental matrix.



Epipolar geometry

- Each pair of points $(\mathbf{x}, \mathbf{x}')$ yields one equation : $\mathbf{x}^T F \mathbf{x}' = 0$
- The epipolar constraint equation is linear in the entries of F and it can be rewritten as :

$$\begin{bmatrix} & \cdots & & & & & & \cdots & \\ & \vdots & & & & & & \vdots & \\ xx' & xy' & x & yx' & yy' & y & x' & y' & 1 \\ & \vdots & & & & & & \vdots & \\ & \cdots & & & & & & \cdots & \end{bmatrix} \begin{bmatrix} f_{11} \\ f_{12} \\ \vdots \\ \vdots \\ f_{33} \end{bmatrix} = 0$$

- With a sufficient number of correspondences in general position it is possible to determine F.
- No knowledge about the cameras or scene structure is necessary to find F.

SVD applications

Homography estimations

- Following the same idea as in the case of fundamental matrix estimation,
- each pair of points $(\mathbf{x}, \mathbf{x}')$ yields one equation : $\mathbf{x}' \times (H\mathbf{x}) = 0$

Outline

What is PCA ?

- Most common answer would be '*an algorithm for dimensionality reduction*'
- Yes, but :
  - Where does the algorithm comes from ?
  - What's the underlying model ?
- PCA is actually many different things (models)
  - latent variable model (Hotelling, 1930s)
  - variance maximization directions (Pearson, 1901)
  - optimal linear reconstruction (Kosambi-Karhunen-Loève transform in signal processing)
- It just turns out that these different models lead to the same algorithm (in the linear Gaussian case)

What is PCA ?

Goal of PCA

The main goal of PCA is to express a complex data set into a new set a basis vectors that 'best' explain the data

- So, PCA is essentially a **change of basis**
- We want to find the most meaningful basis to re-express the data such that
  - the new basis **reveals hidden structure**
  - the new basis **removes redundancy**
- Most of the time, we would like a lower dimensional space.

The algorithm

Given a set of set of N data samples $\mathbf{x}_i \in \mathbb{R}^d$ such that $\sum_i \mathbf{x}_i = 0$

1. Compute the sample covariance matrix $\mathbf{C} = \frac{1}{N}\sum_{i=1}^{N} \mathbf{x}_i\mathbf{x}_i^T$ Note that $\mathbf{C}$ is a $d \times d$ matrix.
2. Compute eigen-decomposition of $\mathbf{C}$: $\mathbf{C} = \mathbf{U}\Lambda\mathbf{U}^T$
   $\mathbf{U}$ is an orthogonal $d \times d$ matrix : $\mathbf{U} = [\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_d]$
   Λ is a diagonal matrix : $\Lambda = \text{diag}(\lambda_1, \lambda_2, \cdots, \lambda_d)$.
3. Since $\mathbf{C}$ is symmetric, its eigenvectors $\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_d$ form a basis of $\mathbb{R}^d$.
   - The eigenvectors $\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_d$ are called principal components
   - The corresponding eigenvalues $\lambda_1 > \lambda_2 > \cdots > \lambda_d$ give the importance of each principal axis.

PCA algorithm

The PCA algorithm is pretty simple

- First, center the data (if it is not) $\sum_i \mathbf{x}_i = 0$
- Then, compute the sample covariance matrix and its eigenvectors
- Finally, each sample point $\mathbf{x}_i$ can be represented in the new basis (projection onto the eigenspace) as

$$\mathbf{y}_i = \mathbf{U}^T \mathbf{x}_i$$

- We claim that the new representation makes the data un-correlated, i.e. $Cov(\mathbf{y}_i, \mathbf{y}_j) = 0$ if $i \neq j$.

PCA algorithm

We claim that the new representation makes the data un-correlated

Why ?

The sample covariance of the transformed data is

$$
\begin{aligned}
\mathbf{C}_{new} &= \frac{1}{N}\sum_{i=1}^{N}\mathbf{y}_i\mathbf{y}_i^T = \frac{1}{N}\sum_{i=1}^{N}(\mathbf{U}^T\mathbf{x}_i)(\mathbf{U}^T\mathbf{x}_i)^T \\
&= \frac{1}{N}\sum_{i=1}^{N}\mathbf{U}^T\mathbf{x}_i\mathbf{x}_i^T\mathbf{U} = \mathbf{U}^T\left(\frac{1}{N}\sum_{i=1}^{N}\mathbf{x}_i\mathbf{x}_i^T\right)\mathbf{U} \\
&= \mathbf{U}^T\mathbf{C}\mathbf{U} = \mathbf{U}^T(\mathbf{U}\Lambda\mathbf{U}^T)\mathbf{U} = (\mathbf{U}^T\mathbf{U})\Lambda(\mathbf{U}^T\mathbf{U}) \\
&= \Lambda
\end{aligned}
$$

Hence, when projected onto the principal components, the data is decorreletad.

PCA algorithm

Dimensionality reduction

- We usually want to represent our data in a lower dimensional space $\mathbb{R}^k$, with $k \ll d$.
- We achieve this by projecting onto the k principal axes which preserve most of the variance in the data
- From the previous analysis, we see that those axes correspond to the eigenvectors associated with the k largest eigenvalues

$$\mathbf{U} = \begin{bmatrix} | & | & & | \\ \mathbf{u}_1 & \mathbf{u}_2 & \dots & \mathbf{u}_d \\ | & | & & | \end{bmatrix}_{d\times d} \Rightarrow \mathbf{U}_k = \begin{bmatrix} | & | & & | \\ \mathbf{u}_1 & \mathbf{u}_2 & \dots & \mathbf{u}_k \\ | & | & & | \end{bmatrix}_{d\times k}$$

- The projected data is then $\mathbf{y}_i = \mathbf{U}_k^T \mathbf{x}_i$, $\mathbf{y}_i \in \mathbb{R}^k$.

Dual PCA

- Suppose we are working with images, each of size $M \times N$
- We represent an image as a vector $\mathbf{x} \in \mathbf{R}^d$, with $d = MN$
- The sample covariance is given $\mathbf{C} = \frac{1}{N}\mathbf{X}\mathbf{X}^T$
- $\mathbf{C}$ is a $d \times d$ matrix
- When the images have high resolution, d is large and so is $\mathbf{C}$
- Imagine computing the eigenvalues/eigenvectors of a 1000000×1000000 matrix with MATLAB !
- Moreover, the number N of images is usually much smaller then d.
- The dual PCA algorithm is a **small size trick**.

PCA algorithm

Dual PCA

- Let $\mathbf{X}$ be the $d \times N$ data matrix $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N]$, $\mathbf{x}_i \in \mathbb{R}^d$
- The sample covariance can be computed as $\mathbf{C} = \frac{1}{N}\mathbf{X}\mathbf{X}^T$
- If $N \ll d$, then it is better to work with $\mathbf{C}' = \frac{1}{N}\mathbf{X}^T\mathbf{X}$
  - $\mathbf{C}'$ is an $N \times N$ matrix
  - Let $\mathbf{C}' = \mathbf{U}'\Lambda'\mathbf{U}'^T$ be the eigen-decomposition of $\mathbf{C}'$
  - We have $\Lambda = \Lambda'$, i.e. eigenvalues of $\mathbf{C}$ and $\mathbf{C}'$ are equal
  - We have $\mathbf{u}_i = \mathbf{X}\mathbf{u}'_i$, for all i
- Working with $\mathbf{C}'$ is computationally less expensive if $N \ll d$.
  - We get eigenvectors of $\mathbf{C}'$: $\mathbf{u}'_i, i = 1, \ldots, N$
  - And those of $\mathbf{C}$, the principal components we care about, are given as $\mathbf{u}_i = \mathbf{X}\mathbf{u}'_i$.

The matrix $\mathbf{C}' = \frac{1}{N}\mathbf{X}^T\mathbf{X}$ is called the Gram (or Gramian) matrix.

PCA algorithm

Connection with SVD

PCA & SVD

There is a direct link between PCA and SVD

- Let $\mathbf{X}$ be the $d \times N$ data matrix $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N]$
- The sample covariance can be computed as $\mathbf{C} = \frac{1}{N}\mathbf{X}\mathbf{X}^T$
  - The eigenvectors of $\mathbf{C}$ are the principal components
- The SVD of $\mathbf{X}$ is given as $\mathbf{X} = \mathbf{U}\Sigma\mathbf{V}^T$,
  where $\mathbf{U}$ is orthogonal $d \times d$ and $\mathbf{V}$ is orthogonal $N \times N$.
  - The columns of $\mathbf{U}$ are eigenvectors of $\mathbf{X}\mathbf{X}^T$
  - So, the columns of $\mathbf{U}$ are the principal components
  - The sigular values of $\mathbf{X}$ are ordered as the eigenvalues of $\mathbf{C}$, since $\sigma_i^2 = \lambda_i$
  - The columns of $\mathbf{V}$ are the 'dual' principal components
- **SVD gives it all !**

Other facts about PCA

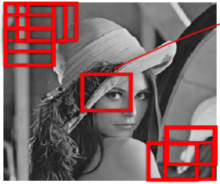
- It can be shown that the principal axes found as described above (i.e. the matrix $\mathbf{U}$) form the best set of orthogonal basis vectors which minimizes the **average reconstruction error**

$$\mathbf{U} = \underset{\mathbf{W}}{\operatorname{argmin}} \frac{1}{N} \sum_{i=1}^{N} \|\mathbf{x}_i - \mathbf{W}^T \mathbf{x}_i\|_F$$

- For each data point $\mathbf{x}_i$, the projection $\mathbf{y}_i = \mathbf{U}_k^T \mathbf{x}_i$ is the best k-dimensional approximation to $\mathbf{x}_i$ (best in the mean square error sense)
- The principal axes are axes of maximum variance

PCA based image denoising

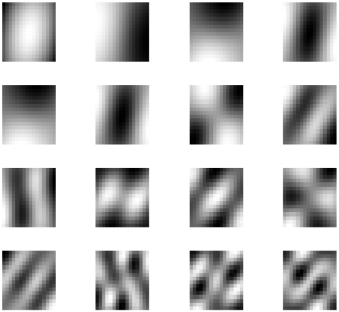
- Assume the noise is uniformly spread out over all directions
- Assume the image lies in a low dimensional subspace
- Extract local patches from the image and compute an orthogonal basis using PCA
- Can denoise each patch by projection onto the first K principal components



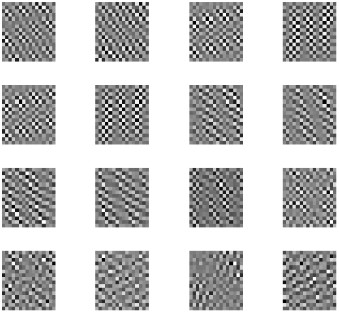
$$X = [\ldots | \ldots]$$

PCA based image denoising

- First K principal components (PCs) capture data image structures
- Similar to wavelet based denoising



First 16 PCs



Last 16 PCs

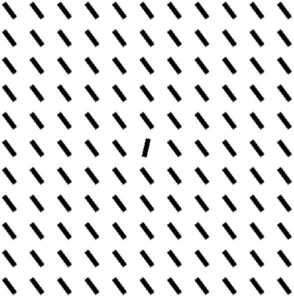
PCA based image denoising

Input image

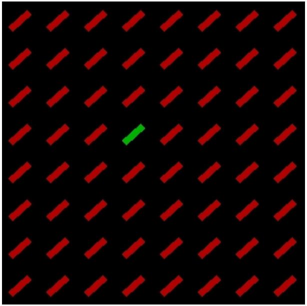
Denoised image

PCA based saliency detection

- Visual saliency is an attention mechanism that helps to focus on ROI rather than processing the entire image
- It is a widely studied problem in computer vision :
  - An image region is considered *salient* if it differs from its neighbour
  - Features used can be : color, edge, torientation, exture, motion, etc.



Orientation



Color

FIGURE : Popout effect.

PCA based saliency detection

PCA provides a very simple and effective solution (Margolin *et al.* 2013)

- The saliency of a patch is computed as the L_1 norm of the pacth projected onto the PCA axes :

$$P(\mathbf{x}) = \sum_{k=1}^{K} |\alpha_{\mathbf{x}}^{k}|.$$

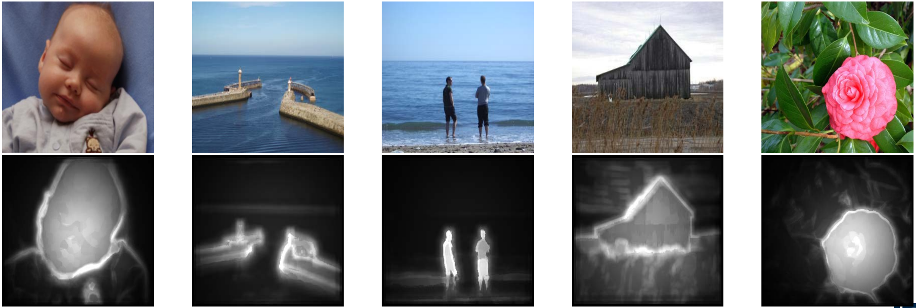


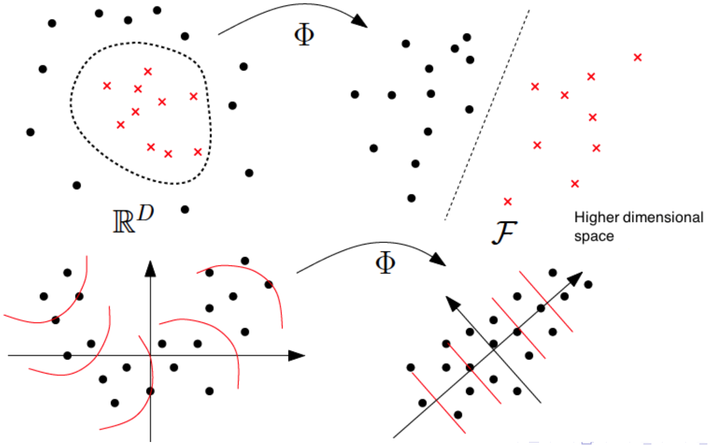
FIGURE : PCA-based saliency detection (images from Margolin et al. CVPR 2013).

Outline

Kernel PCA

Kernel methods

General idea : Map the data to a higher dimensional space (features space) in which, we hope, we can use a linear method



A word about kPCA

- Introduced by Schoelkopf, Smola and Mueller in 1999.
- The key observation is that the eigenvectors of $\mathbf{C}$ can be written as a linear combination of the sample data points $\mathbf{u}_k = \sum_i \alpha_i^{(k)} \mathbf{x}_i$, with $\alpha^{(k)} \in \mathbb{R}^N$.
- The second key observation is that, the coefficients of the linear combination are solutions to the eigenvalue problem $\mathbf{K}\alpha^{(k)} = \lambda^{(k)}\alpha^{(k)}$ where $\mathbf{K}$ is the $N \times N$ Gram matrix defined by $\mathbf{K}_{ij} = \mathbf{x}_i^T \mathbf{x}_j$.
  - $\mathbf{K}$ is sometimes called the inner product matrix or the kernel matrix
- Kernel PCA corresponds to dual-PCA in the features space

A word about kPCA

Given a set of set of N data samples $\mathbf{x}_i \in \mathbb{R}^d$

1. Compute the Gram matrix $\mathbf{K}_{ij} = \mathbf{x}_i^T \mathbf{x}_j$
2. Find the eigenvectors of $\mathbf{K}$: $\mathbf{K}\alpha^{(k)} = \lambda^{(k)}\alpha^{(k)}$
3. The principal components are given by $\mathbf{u}_k = \sum_i \alpha_i^{(k)} \mathbf{x}_i$
4. Each data point $\mathbf{x}_i$ is projected onto the eigenspace as

$$\mathbf{u}_k^T \mathbf{x}_i = \sum_j (\alpha_j^{(k)} \mathbf{x}_j)^T \mathbf{x}_i = \sum_j \alpha_j^{(k)} (\mathbf{x}_j^T \mathbf{x}_i) = \sum_j \alpha_j^{(k)} K_{ji}$$

Kernel PCA

A word about kPCA

kPCA

We only need the Gram matrix $\mathbf{K}$

- We can replace $\mathbf{x}_i \rightarrow \phi(\mathbf{x}_i)$ (mapping)
- And define $\mathbf{K}_{ij} = \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j)$
- And do the same calculations

Kernel Trick

- ϕ can be any mapping function (usually mapping the data to higher dimension)
- **The kernel trick** is we don't need to map the data explicitly as long as we can compute the matrix $\mathbf{K}$ using some well defined kernel !

Kernel PCA

Example

- Assume data in $\mathcal{R}^2$, i.e. $\mathbf{x}_i = [x_1, x_2]^T$
- We wish to map the data into a higher dimensional space ($\mathbb{R}^6$) and find the principal axes in that space. We use

$$\phi(\mathbf{x}_i) = [1,\ \sqrt{2}x_1,\ \sqrt{2}x_2,\ \sqrt{2}x_1x_2, x_1^2, x_2^2]^T$$

- Now let define a polynomial kernel as $k(\mathbf{x}, \mathbf{y}) = (1 + \mathbf{x}^T\mathbf{y})^2$;
  then $k(\mathbf{x}, \mathbf{y}) = \phi(\mathbf{x})^T\phi(\mathbf{y})$.
- By defining $\mathbf{K}$ such that $\mathbf{K}_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$, we don't need to explicitly map each data point in $\mathbb{R}^6$.
  We can work with the point in $\mathbb{R}^2$ and still get the eigenvectors in the mapped space
- That's the power of the kernel trick

A word about kPCA

- Thus, kPCA allows us to compute eigenvectors is a higher dimensional space without visiting it ♦
- Another common kernel is the radial basis function (RBF) which maps data to an infinite dimensional space

$$k(\mathbf{x}, \mathbf{y}) = exp(-\gamma||\mathbf{x} - \mathbf{y}||^2)$$

- Mapping data to higer dimensional space can be useful for classification purposes.
- However, the choice of the kernel is delicate.

A word about PPCA

- Standard PCA (and kPCA) does not provide a probabilistic interpretation
- PPCA is a probabilistic formulation of PCA from a Gaussian latent variable model
  - We seek $\mathbf{W}$, σ^2 and μ such that

$$\mathbf{x} = \mathbf{W}\mathbf{y} + \mu + \epsilon,$$

    with $\mathbf{y} \sim \mathcal{N}(0, \mathbf{I})$ and $\epsilon \sim \mathcal{N}(0, \sigma^2\mathbf{I})$
  - We have, from this model, that

$$\mathbf{x} \sim \mathcal{N}(\mu, \mathbf{W}\mathbf{W}^T + \sigma^2\mathbf{I})$$

- Introduced by Tipping and Bishop in 1999.

A word about PPCA

- The parameters of the model are obtained via maximum likelihood (ML) estimation
- The ML estimate of μ is given by the mean of the data :

$$\mu_{ML} = \frac{1}{N}\sum_{i=1}^{N}\mathbf{x}_i$$

- The ML estimate for σ^2 is given by

$$\sigma^2_{ML} = \frac{1}{d-k}\sum_{j=k+1}^{d}\lambda_j$$

- The ML estimate for $\mathbf{W}$ is given by

$$\mathbf{W}_{ML} = \mathbf{U}_k(\Lambda_k - \sigma^2\mathbf{I})^{1/2}\mathbf{R}$$

A word about PPCA

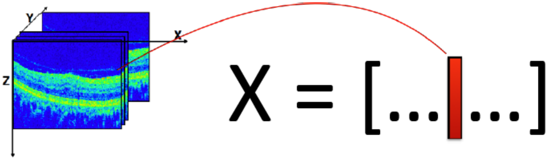
The ML estimate for $\mathbf{W}$ is given by

$$\mathbf{W}_{ML} = \mathbf{U}_k(\Lambda_k - \sigma^2\mathbf{I})^{1/2}\mathbf{R}$$

- columns of $\mathbf{U}_k$ are the k dominant eigenvectors of the sample covariance
- Λ_k is diagonal and contains the corresponding k largest eigenvalues
- $\mathbf{R}$ is an arbitrary orthogonal matrix
- When $\mathbf{R} = \mathbf{I}$ and $\sigma^2 \to 0$, PPCA = PCA
- PPCA is derived iteratively (using EM algorithm) and can deal with missing data

Why multidimensional PCA ?

- Applying PCA to multidimensional data, e.g. 2D data



  - The 2D image is vectorized
- Results in high dimensional vectors to work with
  - An image of size 512×512 becomes a vector of size $262,144$
  - A 3D volume of size $512 \times 512 \times 128 \rightarrow 28.10^6$-D vector !
- The natural spatial correlation is removed ▲

Multidimensional PCA

MPCA

MPCA uses the full 2D or 3D nature of the data

2D-PCA (in PAMI 2004)

- Given a set of images $A_1, A_2, \ldots, A_M$ of size $m \times n$
- Compute the *image covariance matrix*

$$\mathbf{G} = \frac{1}{M} \sum_i (A_i - \bar{A}_i)^T (A_i - \bar{A}_i)$$

- $\mathbf{G}$ is an nonnegative $n \times n$ matrix and its d largest eigenvectors are used to extract features from A as $Y_k = AX_k$, $k = 1, \ldots, d$.
- The set of projected features vectors are used to form an $m \times d$ matrix which represents image A

$$B = [Y_1, Y_2, \cdots, Y_d]$$

2D-PCA (in PAMI 2004)

- Find d dominant eigenvectors of $\mathbf{G}$: X_k, $k = 1, \ldots, d$
- Project image image A onto the eigenspace : $Y_k = AX_k$
- Use the obtained features to approximate the image : $B = [Y_1, Y_2, \cdots, Y_d]$
- If $U = [X_1, X_2, \ldots, X_d]$, then $B = AU$.
- Note A is $m \times n$ and B is $m \times d$, $d \ll n$.
- The image can be reconstructed as $\bar{A} = VU^T = \sum_{k=1}^{d} Y_k X_k^T$

Multidimensional PCA

- 2DPCA was shown to be better than PCA (using vectorized images) for face recognition
- However, it does not use full 2D structure of the data
  - It projects the 2D image only in one direction and ignore the other one
- MPCA uses tensor representation and projects a 2D (3D) tensor as a 2D (3D) tensor of smaller size.

Multidimensional PCA

Tensors

- An *Nth*-order tensor is an *N*-dimensional array with *N* modes
  - The number of dimensions of a tensor is its **order**
  - Each dimension of the tensor is called a **mode**

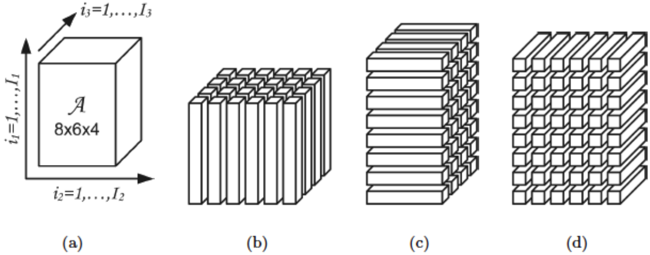


FIGURE : An 3rd order tensor and its three modes (from Lu *et al.* 2008).

Multidimensional PCA

- Thus MPCA find N projections matrices, one in each mode of the tensor
  - MPCA is solved by performing PCA in each mode of the tensor iteratively
- For dimensionality reduction, the projection axes are sorted (weighted) and features are extracted using the 'best' axes.
- The method is appealing
  - But, requires lot of memory for large size data
  - It is not computationaly expensive (not much more than PCA)
  - A Matlab package exists (http ://www.comp.hkbu.edu.hk/~haiping/)

Video saliency with MPCA

- How to extend the PCA-based saliency method (Margolin *et al.* 2013) to deal with videos ?

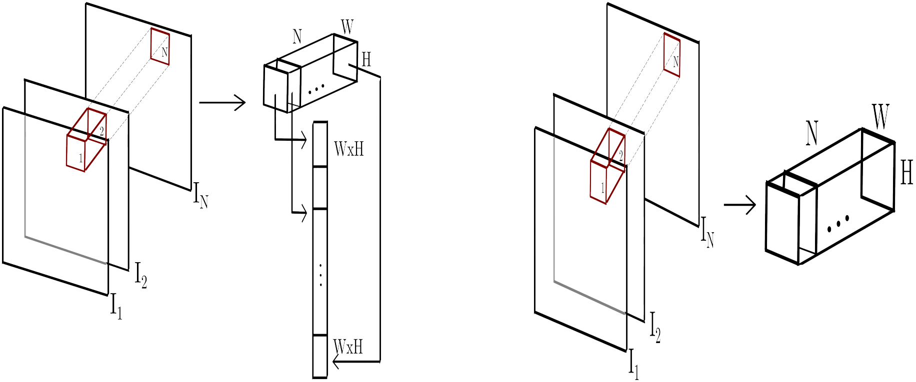


FIGURE : Different options.

Multidimensional PCA

Video saliency with MPCA

- MPCA takes into account the spatio-temporal structure of the video and provides good results

FIGURE : Sidibé *et al.* 2016

Outline

Outline

Bag-of-Words

A bit of history

- The Bag-of-Words (BoW) concept comes from text/documents retrieval community
- Assume you have to organize web pages into categories
  - Categories include **Sports**, **Movies**, **Cooking**
  - Your goal is to asssign each new webpage to one of these categories
  - You look for certain **words** in the webpages
  - For example, you might count how many times the word '*game*' appears in the webpage, or how many times the word '*recipe*' appears.
  - Then, you can assign a category based on the frequency of the words
- The set of words is called a **dictionary**
- And each webpage is represented by a **bag of words** from the dictionary

A bit of history

- Analysing a set of N documents, each represented by

$$\mathbf{x}^n = [x_1^n, \ldots, x_D^n]^T,$$

  where x_i^n counts how many times word i appears in document n
- D is typically very large and $\mathbf{x}$ will be very sparse
- The term-frequency (TF) is defiend as

$$tf_i^n = \frac{x_i^n}{\sum_i x_i^n}$$

- The inverse-document frequency (IDF)is given by

$$idf_i = \log \frac{N}{\text{\# of documents that contain term } i}$$

A bit of history

- Analysing a set of N documents, each represented by

$$\mathbf{x}^n = [x_1^n, \ldots, x_D^n]^T,$$

  where x_i^n counts how many times word i appears in document n
- The term-frequency - inverse document frequency (TF-IDF) is given by

$$x_i^n = tf_i^n \times idf_i$$

- TF-IDF gives high weight to terms that appear often in a document, but rarely amongst documents.

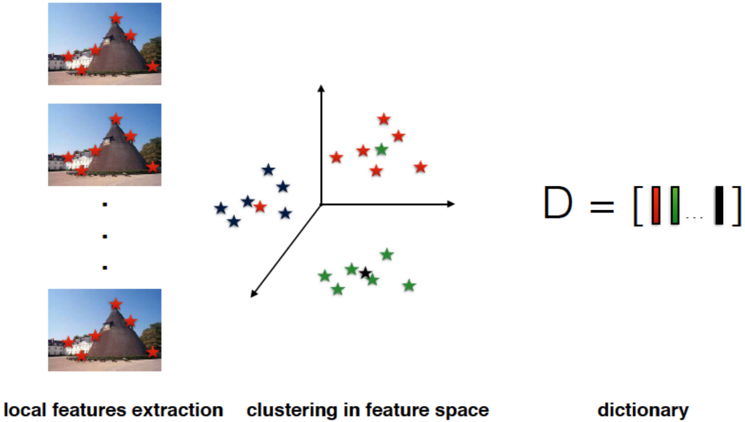
A bit of history

- This is the idea that was introduced to the computer vision community in the context of image category recognition
- The two seminal papers are :
  1. "*Video Google : a text retrieval approach to object matching in videos*", Sivic and Zisserman, ICCV 2003
  2. "*Visual categorization with bag of keypoints*", Csurka et al., ECCV Workshop 2004
- Paper 1 introduced the concept of visual vocabulary and used TF-IDF for retrieval
- Paper 2 introduced the concept of bag of features (later commonly used as BoW)

Bag-of-Words

Key issues

- How to construct a visual dictionary ?



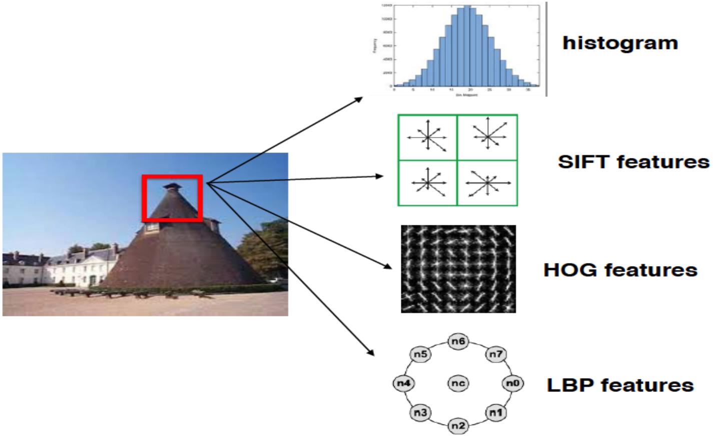
Bag-of-Words

Key issues

- Vocabulary size ?
- Sampling strategy ?
- Clustering/Quantization ?
- Unsupervised vs Supervised ?

Local Features

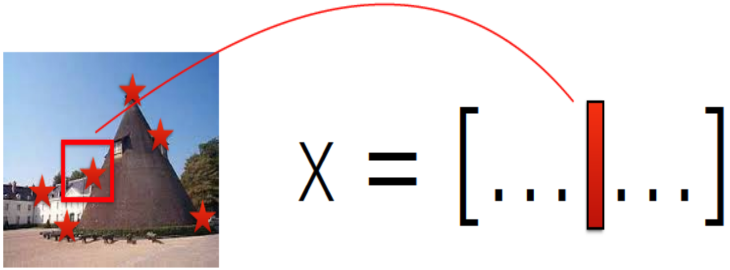
Many local features can be used



Sampling strategy

Keypoints detection

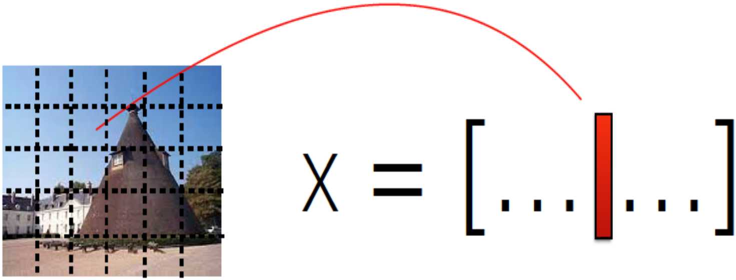
- Detect a set of keypoints (Harris, SIFT, etc)
- Extract local descriptors around each keypoint



Sampling strategy

Dense sampling

- Divide image into local patches
- Extract local features from each patch



Clustering/Quantization

- For each image I_i we extract a set of low level descriptors and represent them as a feature matrix $\mathbf{X}_i$:

$$\mathbf{X}_i = \begin{bmatrix} | & | & & | \\ \mathbf{f}_i^1 & \mathbf{f}_i^2 & \dots & \mathbf{f}_i^{N_i} \\ | & | & & | \end{bmatrix},$$

where $\mathbf{f}_i^1, \dots, \mathbf{f}_i^{N_i}$ are the N_i descriptors extracted from I_i.

- We then put together all descriptors from all training images to form a big training matrix $\mathbf{X}$:

$$\mathbf{X} = \begin{bmatrix} \mathbf{X}_1 & \dots & \mathbf{X}_N \end{bmatrix}.$$

$\mathbf{X}$ is a matrix of size $d \times M$, with $M = \sum_{i=1}^{N} N_i$ and d the dimension of the descriptor.

Clustering/Quantization

- To simplify the notation, we will just write the set of descriptors from the training images as

$$\mathbf{X} = \begin{bmatrix} | & | & & | \\ \mathbf{f}_1 & \mathbf{f}_2 & \dots & \mathbf{f}_M \\ | & | & & | \end{bmatrix}.$$

- Create a dictionary by solving the following optimization problem

$$\min_{\mathbf{D}} \sum_{m=1}^{M} \min_{k=1\dots K} ||\mathbf{f}_m - \mathbf{d}_k||^2,$$

  where $\mathbf{D} = [\mathbf{d}_1, \dots, \mathbf{d}_K]$ are the K clusters centers to be found and $||.||$ is the L_2 norm of vectors.
- **D** is the visual dictionary or codebook.

Clustering/Quantization

- The optimization problem

$$\min_{\mathbf{D}} \sum_{m=1}^{M} \min_{k=1...K} \|\mathbf{f}_m - \mathbf{d}_k\|^2,$$

is solved iteratively with K-means algorithm.

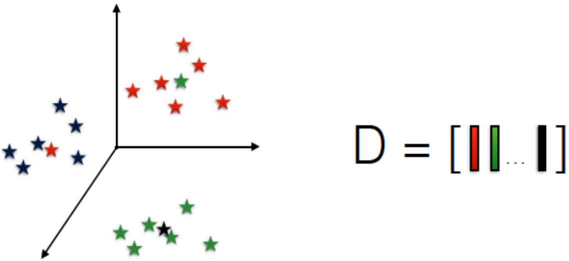
K-means

1. Initialize the K centers (randomly)
2. Assign each data point to one of the K centers
3. Update the centers
4. Iterate until convergence

Clustering/Quantization

- K-means algorithm results in a set of K cluster centers which form the dictionary

$$\mathbf{D} = \begin{bmatrix} | & | & & | \\ \mathbf{d}_1 & \mathbf{d}_2 & \dots & \mathbf{d}_K \\ | & | & & | \end{bmatrix}_{d \times K}$$



Features coding

- Given the dictionary $\mathbf{D}$
- Given a set of low-level features $\mathbf{X}_i$ from image I_i

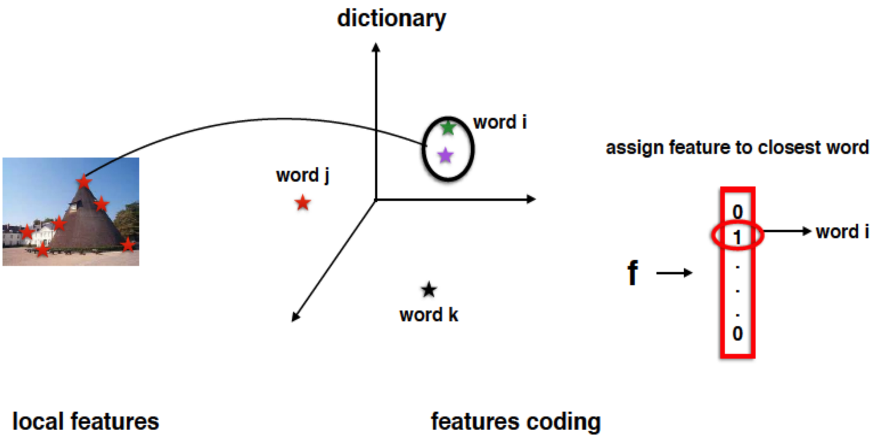
$$\mathbf{X}_i = \begin{bmatrix} | & | & & | \\ \mathbf{f}_i^1 & \mathbf{f}_i^2 & \dots & \mathbf{f}_i^{N_i} \\ | & | & & | \end{bmatrix}$$

- Encode each local descriptor $\mathbf{f}_i^l$ using the dictionary $\mathbf{D}$
  - Find $\mathbf{a}_l$ such that

$$\min_{\mathbf{a}_l} ||\mathbf{f}_i^l - D\mathbf{a}_l||^2 \; s.t. \; ||\mathbf{a}_l||_0 = 1, \mathbf{a}_l \geq 0$$

Features coding

- Encode each local descriptor $\mathbf{f}_i^l$ using the dictionary $\mathbf{D}$



Features pooling

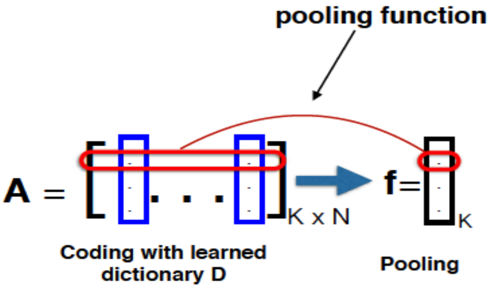
- The coding of image I_i results in a matrix of codes $\mathbf{A}$

$$\mathbf{A} = \begin{bmatrix} | & | & & | \\ \mathbf{a}_1 & \mathbf{a}_2 & \dots & \mathbf{a}_K \\ | & | & & | \end{bmatrix}_{K \times N_i} ,$$

  where each $\mathbf{a}_l$ satisfies $||\mathbf{a}_l||_0 = 1, \ \mathbf{a}_l \geq 0$
- The pooling step transforms $\mathbf{A}$ into a single signature vector $\widehat{\mathbf{x}}_i$

$$\widehat{\mathbf{x}}_i = \text{pooling}(\mathbf{A})$$

Features pooling



- A popular choice for pooling is to compute a histogram

$$\widehat{\mathbf{x}}_i = \frac{1}{N_i} \sum_{l=1}^{N_i} \mathbf{a}_l$$

- The final vector just encodes the frequency of occurrence of each visual words.

BoW representation

Summary : Basic BoW framework

1. Extract a set of local features from all images

$$\mathbf{X} = \begin{bmatrix} | & | & & | \\ \mathbf{f}_1 & \mathbf{f}_2 & \dots & \mathbf{f}_M \\ | & | & & | \end{bmatrix}_{d \times M}$$

2. Create a visual dictionary by clustering of the set of local features

$$\mathbf{D} = \begin{bmatrix} | & | & & | \\ \mathbf{d}_1 & \mathbf{d}_2 & \dots & \mathbf{d}_K \\ | & | & & | \end{bmatrix}_{d \times K}$$

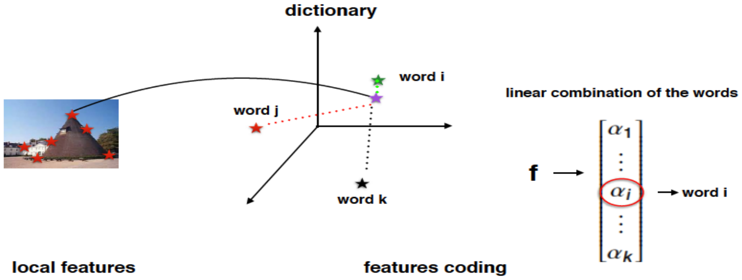
3. Given $\mathbf{D}$, encode each local feature from an image I_i, by assigning it to its closest word : $\mathbf{A} = \begin{bmatrix} | & | & & | \\ \mathbf{a}_1 & \mathbf{a}_2 & \dots & \mathbf{a}_K \\ | & | & & | \end{bmatrix}_{K \times N_i}$

4. Finally, compute the final representation of I_i : $\widehat{\mathbf{x}}_i = \frac{1}{N_i} \sum_{l=1}^{N_i} \mathbf{a}_l$

Improvements : Features coding

- Represent each local feature $\mathbf{f}_i^l$ as a linear combination of the words.

$$\mathbf{f}_i^l = \sum_{p=1}^{K} \alpha_i^p \mathbf{d}_p \qquad s.t.\ \sum_{p=1}^{K} \alpha_i^p = 1,\ \alpha_i^p \geq 0.$$



Improvements : Features coding

Hard assignment

- Assign each local feature $\mathbf{f}_i^l$ to its closest word

$$\mathbf{a}_l = \begin{bmatrix} 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \end{bmatrix}, \qquad \sum_p \mathbf{a}_l^p = 1$$

Soft assignment

- Write each local feature $\mathbf{f}_i^l$ as a linear combination (weighted sum) of the words

$$\mathbf{a}_l = \begin{bmatrix} \alpha_l^1 \\ \vdots \\ \alpha_l^p \\ \vdots \\ \alpha_l^K \end{bmatrix}, \qquad \sum_p \alpha_l^p = 1,\ \alpha_l^p \geq 0$$

Improvements : Features pooling

- **average**

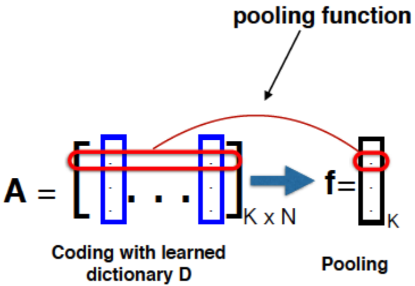
$$\widehat{\mathbf{x}}_i = \frac{1}{N_i}\sum_{l=1}^{N_i} \mathbf{a}_l$$

- **max**

$$\widehat{\mathbf{x}}_i^j = \max_j(\mathbf{a}_l^j)$$

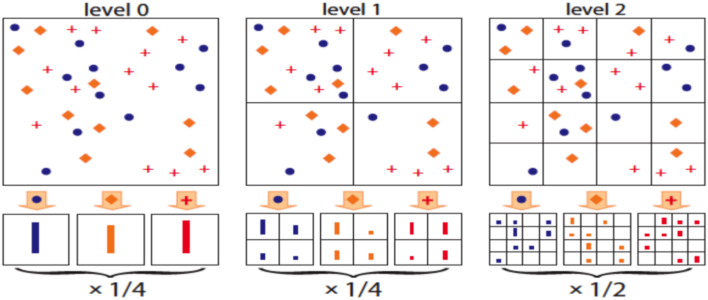
- **mean absolute value**

$$\widehat{\mathbf{x}}_i = \frac{1}{N_i}\sum_{l=1}^{N_i} |\mathbf{a}_l|$$



Improvements : Including spatial information

- BoW model ignores the spatial layout of the features in the image
- Does not take into account the regularities in image composition



Spatial pyramid : Lazebnik et al. CVPR 2006

Outline

Another view of the problem

Representation over a dictionary

- The BoW method can be seen as representing the input images over a given dictionary.
- We represent each image as a **linear combination** of the elements of the dictionary.

$$\underbrace{\begin{bmatrix} | & | & & | \\ \mathbf{x}_1 & \mathbf{x}_2 & \dots & \mathbf{x}_N \\ | & | & & | \end{bmatrix}_{d\times N}}_{\text{Input vectors}} = \underbrace{\begin{bmatrix} | & | & & | \\ \mathbf{d}_1 & \mathbf{d}_2 & \dots & \mathbf{d}_K \\ | & | & & | \end{bmatrix}_{d\times K}}_{\text{Dictionary}} \underbrace{\begin{bmatrix} -- & \alpha_1^T & -- \\ & \vdots & \\ -- & \alpha_K^T & -- \end{bmatrix}_{K\times N}}_{\text{Coefficients of representation}}$$

$$\forall i,\ \mathbf{x}_i = \sum_{k=1}^{K} \alpha_k^{(i)} \mathbf{d}_k.$$

Another view of the problem

$$\underbrace{\begin{bmatrix} | & | & & | \\ \mathbf{x}_1 & \mathbf{x}_2 & \dots & \mathbf{x}_N \\ | & | & & | \end{bmatrix}}_{\mathbf{X}} = \underbrace{\begin{bmatrix} | & | & & | \\ \mathbf{d}_1 & \mathbf{d}_2 & \dots & \mathbf{d}_K \\ | & | & & | \end{bmatrix}}_{\mathbf{D}} \underbrace{\begin{bmatrix} -- & \alpha_1^T & -- \\ & \vdots & \\ -- & \alpha_K^T & -- \end{bmatrix}}_{\mathbf{A}}$$

Representation over a dictionary

We want to solve $\mathbf{X} = \mathbf{DA}$

- We need to constrain the problem (many solutions)
- We can impose constraints on
  - The dictionary **D**
    - For example : a set of orthogonal vectors
  - The representation (matrix of coefficients) **A**
    - For example : only a few non-zero elements
- **Constraints** $\equiv$ **prior information**

Why Sparsity ?

- Consider a simple problem

$$\min_{\mathbf{x}} (A\mathbf{x} - b)^2$$

$$\underbrace{\begin{bmatrix} | & | & & | \\ \mathbf{a}_1 & \mathbf{a}_2 & \dots & \mathbf{a}_N \\ | & | & & | \end{bmatrix}}_{N} \begin{bmatrix} x_1 \\ \cdots \\ x_N \end{bmatrix} = \begin{bmatrix} | \\ \mathbf{b} \\ | \end{bmatrix} \in \mathcal{R}^d$$

- Assuming A is full rank and $N > d$, there is no unique solution
  - Many $\mathbf{x}$ can achieve the minimum

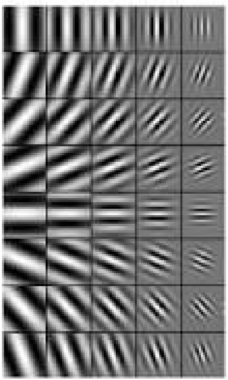
$$\min_{\mathbf{x}} (A\mathbf{x} - b)^2$$

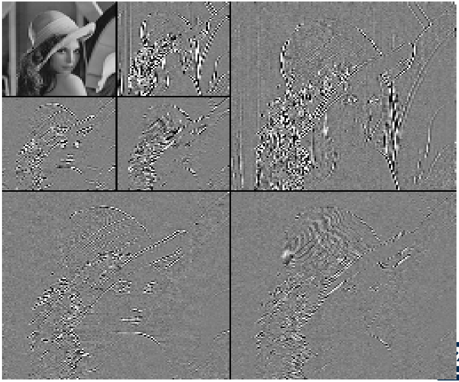
  - Which one do you want ?
- We need to impose some constraints on $\mathbf{x}$
  For instance, choose the $\mathbf{x}$ with the least nonzero elements

$$\boxed{arg \min_{\mathbf{x}} \|\mathbf{x}\|_0, \ s.t.(A\mathbf{x} - b)^2 = 0}$$

Why Sparsity ?

- The more **concise**, the better (Ockham's razor)
- Sparsity is a **good prior** for image representation
  - Images are compressible signals with a compressible representation in DCT or wavelets bases
  - JPEG, JPEG 200





Why Sparsity ?

The image denoising example

$$\min_{\mathbf{x}} f(\mathbf{x}) = \frac{1}{2}||\mathbf{y} - \mathbf{x}||^2 + G(\mathbf{x})$$

$\mathbf{x} \rightarrow$ unknown signal to be recovered
$\mathbf{y} \rightarrow$ given measurement (noisy image)
$G(\mathbf{x}) \rightarrow$ prior or regularization term

- This is a Bayesian point of view : MAP estimation
- The choice of the prior if fundamental

energy	$G(\mathbf{x}) = \lambda\|\|\mathbf{x}\|\|^2$
smoothness	$G(\mathbf{x}) = \lambda\|\|L(\mathbf{x})\|\|^2$
robust statistics	$G(\mathbf{x}) = \lambda\rho(L(\mathbf{x}))$
total variation	$G(\mathbf{x}) = \lambda\|\|\nabla\mathbf{x}\|\|_1$
sparse prior	$G(\mathbf{x}) = \lambda\|\|\mathbf{x}\|\|_0$ for $\mathbf{x} = \mathbf{D}\mathbf{x}$

Sparse coding

Sparse coding

The objective of sparse coding is to reconstruct an input vector (e.g. an image patch) as a **linear combination** of a **small number of vectors** picked from a large **dictionary**

$$\underbrace{\begin{bmatrix} | & | & & | \\ \mathbf{d}_1 & \mathbf{d}_2 & \dots & \mathbf{d}_K \\ | & | & & | \end{bmatrix}}_{\text{Dictionary}} \begin{bmatrix} \\ \alpha \\ \\ \end{bmatrix} = \begin{bmatrix} \\ \mathbf{x} \\ \\ \end{bmatrix}$$

- Every column of **D** is called an atom
- The vector α is the representation of **x** w.r.t. **D**
- α has few non-zero elements (sparsity)

- Every signal is built as a linear combination of few atoms from **D**

Sparse coding

Signal model

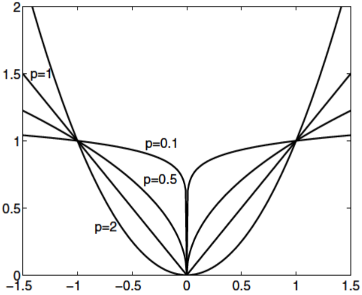
- Every signal is built as a linear combination of few atoms from **D**
- $\mathbf{x} = \mathbf{D}\alpha$ where α is sparse

How to model sparsity ?

- L_p norm :

$$||\alpha||_p^p = \sum_{i=1}^{k} |\alpha_i|^p$$

- As $p \to 0$, we get a count of the nonzero elements of the vector α



- So our model is

$$\boxed{\mathbf{x} = \mathbf{D}\alpha \quad \text{s.t.} \quad ||\alpha||_0^0 < L}$$

Back to the image denoising example

The problem

$$\min_{\mathbf{x}} f(\mathbf{x}) = \frac{1}{2}||\mathbf{y} - \mathbf{x}||^2 + G(\mathbf{x})$$

can be re-written as

$$\min_{\alpha} \frac{1}{2}||\mathbf{D}\alpha - \mathbf{y}||_2^2 \quad \text{s.t.} \quad ||\alpha||_0^0 < L$$

- The vector α is the representation of $\mathbf{x}$: $\hat{\mathbf{x}} = \mathbf{D}\hat{\alpha}$
- Few atoms ($L < K$) can be combined to form the true signal, the noise cannot be fitted well
- Denoising ≡ projection of the noisy image onto a low dimensional space (as with SVD or PCA)

Few issues

Assume we build a signal by the relation $\mathbf{D}\alpha = \mathbf{x}$

$$\begin{bmatrix} | & | & & | \\ \mathbf{d}_1 & \mathbf{d}_2 & \dots & \mathbf{d}_K \\ | & | & & | \end{bmatrix} \begin{bmatrix} \\ \alpha \\ \\ \end{bmatrix} = \begin{bmatrix} \\ \mathbf{x} \\ \\ \end{bmatrix}$$

We want to find the signal's represntation

$$\min_{\alpha} ||\alpha||_0^0 \quad \text{s.t.} \quad \mathbf{x} = \mathbf{D}\alpha$$

- **Uniqueness** ?
  - Why should we necessary get $\hat{\alpha} = \alpha$?
  - It might happen that eventually $||\hat{\alpha}||_0^0 < ||\alpha||_0^0$?

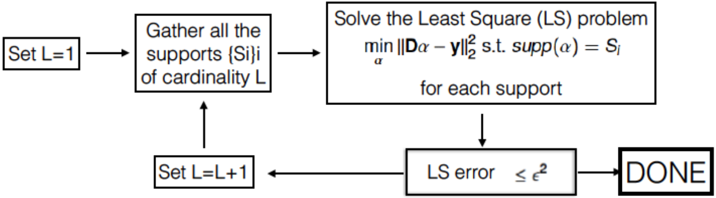
How to compute α ?

- Assume we know the dictionary **D** and **x** and want to recover α
- Solve

$$\min_{\alpha} ||\alpha||_0^0 \quad \text{s.t.} \quad ||\mathbf{D}\alpha - \mathbf{x}||_2^2 < \epsilon^2$$

- This happens to be a combinatorial NP hard problem

Pourquoi ? Recipe for solving this problem



Assume $K = 1000$ and $L = 10$ (kwown !), and 1 nano-sec per each LS
We would need ~8e+6 years to solve this problem ! ! !

Illustration based on G. Sapiro.

Sparse coding

How to compute α ?

$$\min_{\alpha} ||\alpha||_0^0 \quad \text{s.t.} \quad ||\mathbf{D}\alpha - \mathbf{x}||_2^2 < \epsilon^2$$

We have seen it is an NP hard problem : let's approximate.

Relaxation methods

Smooth the L_0 norm and use continuous optimization techniques

Greddy algorithms

Build the solution one nonzero element at a time

How to compute α ?

Relaxation methods : Replace L_0 by L_1 norm
Instead of solving

$$\min_{\alpha} \|\alpha\|_0^0 \quad \text{s.t.} \quad \|\mathbf{D}\alpha - \mathbf{x}\|_2^2 < \epsilon^2$$

Solve

$$\boxed{\min_{\alpha} \|\alpha\|_1^1 \quad \text{s.t.} \quad \|\mathbf{D}\alpha - \mathbf{x}\|_2^2 < \epsilon^2}$$

- The new problem is known as Basis-Pursuit (BP)
- The new problem is convex (quadratic programing) and can be solved efficiently
- Under certain conditions (on **D** and L) both problems are equivalent ! (Candes et al. 2006)

Sparse coding

How to compute α ?

Greedy algorithms : Find one atom at a time

- Step 1 : find the atom of $\mathbf{D}$ that best matches the signal $\mathbf{x}$
- Next step : Given previously found atoms, find the next atom to best fit the residual
- The algorithm stops when $||\mathbf{D}\alpha - \mathbf{x}||_2 < \epsilon$

Note : each of the steps just involves solving a least square problem.
Greedy algorithms are known as Matching-Pursuit (MP)

Sparse coding

- We now know how to solve the sparse coding problem

  Given the dictionary **D** and a signal **x**, find the sparse vector α

$$\begin{bmatrix} | & | & & | \\ \mathbf{d}_1 & \mathbf{d}_2 & \dots & \mathbf{d}_K \\ | & | & & | \end{bmatrix} \begin{bmatrix} \\ \alpha \\ \\ \end{bmatrix} = \begin{bmatrix} \\ \mathbf{x} \\ \\ \end{bmatrix}$$

- The next question is : how is the dictionary **D** obtained ?

Dictionary learning

Assumption : good behaved images have a sparse representation
⇒ **D** should be chosen such that it sparsifies the representation

Two options :

1. Choose **D** from a kwown set of transformation
   - DCT, wavelet, curvelet, steerable, bandlets, etc
2. Use a universal dictionary
   - obtained from a large dataset of images (ImageNet)
3. Learn the dictionary from examples
   - Training

Learning the dictionary from examples

- We are given a set of training examples $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N]$
- We want to find a dictionary **D** and a sparse codes matrix **A** such that

$$\underbrace{\begin{bmatrix} & & \\ & \mathbf{X} & \\ & & \end{bmatrix}_{d\times N}}_{\text{training data matrix}} = \underbrace{\begin{bmatrix} & & \\ & \mathbf{D} & \\ & & \end{bmatrix}_{d\times K}}_{\textit{dictionary}} \underbrace{\begin{bmatrix} & & \\ & \mathbf{A} & \\ & & \end{bmatrix}_{K\times N}}_{\text{sparse codes matrix}}$$

Dictionary learning

Learning the dictionary from examples

Our goal is to solve

$$\min_{\mathbf{A},\mathbf{D}} \sum_{j=1}^{N} ||\mathbf{D}\alpha_j - \mathbf{x}_j||_2^2 \quad \text{s.t.} \quad \forall j \; ||\alpha_j||_0^0 \leq L$$

The K-SVD [1] algorithm is one effective technique for dictionary learning

- It is an unsupervised dictionary learning technique
- It is a generalization of K-means clustering method

1. Aharon, et al., "The K-SVD : An Algorithm for Designing of Overcomplete Dictionaries for Sparse Representation", IEEE Trans. On Signal Processing, 54(11), pp. 4311-4322, 2006.

Dictionary learning

K-SVD algorithm

K-SVD is an extension of K-means algorithm

1. **Initialize the dictionary D**
   - with random K signals from $\mathbf{X}$ ($K < N$)
2. **Given D, find A by sparse coding each column of X**
   - we can use any pursuit algorithm : MP, OMP or BP
3. **Update D one atom at a time**
   - $\forall \mathbf{d}_k \in \mathbf{D}$ select the signals $\mathbf{x}_j \in \mathbf{X}$ that use that atom ($\mathbf{X}^k$)
   - compute the residual for all the examples that use $\mathbf{d}_k$, without taking into account $\mathbf{d}_k$ itself

     $$\mathbf{E}^k = \mathbf{X}^k - \mathbf{DA} + \mathbf{d}_k \alpha_k$$

   - find $\mathbf{d}_k$ to better fit the residual :

     $$\min_{\mathbf{d}_k, \alpha_k} \|\alpha_k \mathbf{d}_k^T - \mathbf{E}^k\|^2$$

     this linear system is solved using SVD
4. **Go to step 2 and iterate until convergence**

K-SVD vs K-means

K-means

- Initialize the K centers
- Assign each data point to one of the K centers
- Update the centers
- Iterate

K-SVD

- Initialize the K atoms of $\mathbf{D}$
- Sparse code each example with $\mathbf{D}$
- Update the dictionary $\mathbf{D}$
- Iterate

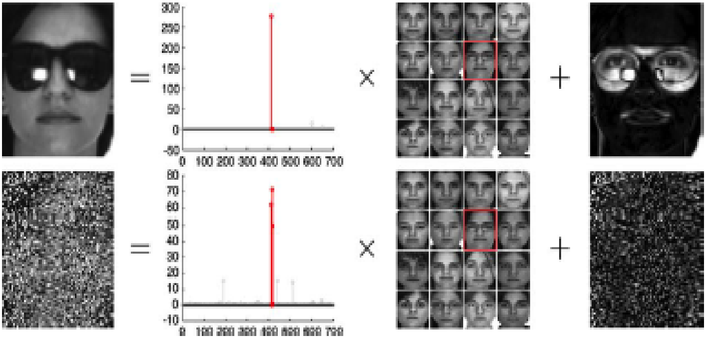
Some applications

Sparse representations have achieved state-of-art results in several applications

- Image denoising
- Image super-resolution
- Image impainting
- Face recognition
- PASCAL challenge (image recognition)
- Activity recognition in videos
- Speech recognition and NLP
- etc

Some applications

Face recognition



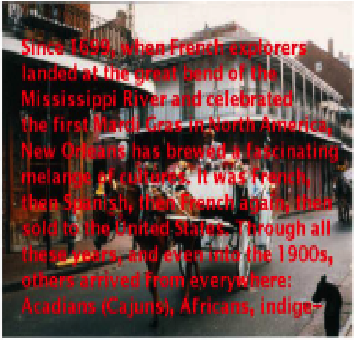
From Wright et al., PAMI 2010

Image restoration

From Mairal et al., TIP 2009

Image restoration



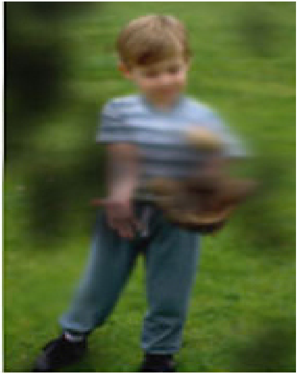
From Mairal et al., TIP 2009

Outline

What is Diabetic Retinopathy ?

- The most common diabetic eye disease
- A leading cause of blindness in Europe and America
- > 300 millions people will be affected by 2025 worldwide

Normal vision



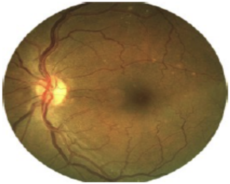
Vision with DR

What is Diabetic Retinopathy ?

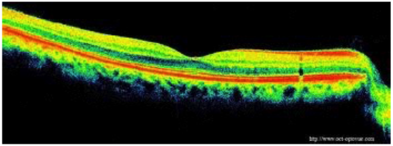
- Diabetic Retinopathy (DR) damages the retinal blood vessels
- It is suggested that 80% of people which have diabetes for more than 10 years are affected by DR.
- 90% of DR cases can be prevented through early detection and treatment
- **Early detection of clinical signs is important**

DR diagnosis tools

Fundus camera



OCT camera



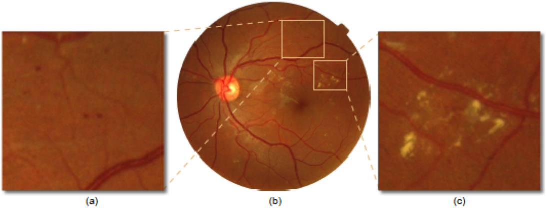
DR detection

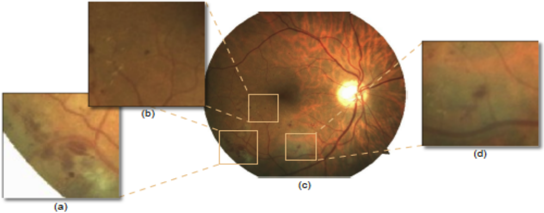
- DR may not be perceived until it reaches severe stage

- Early DR symptoms include :
  - Microaneurysms (MAs)
  - Cotton wool spots
  - Hemorrhages
  - Exudates
  - Drusens
  - Etc

DR symptoms

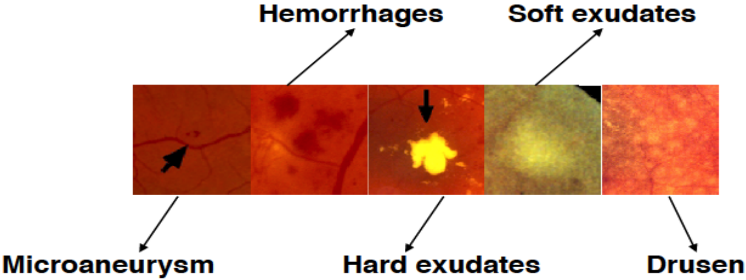
Several lesions may be present in the same image



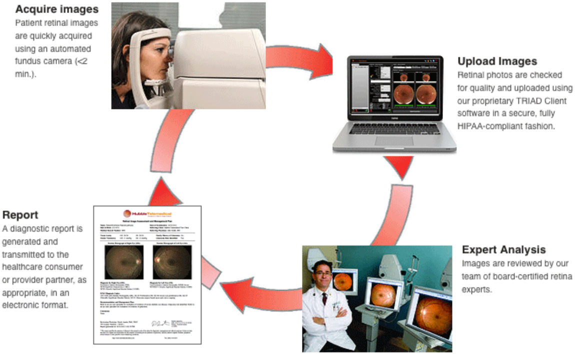


DR symptoms

Several lesions may be present in the same image



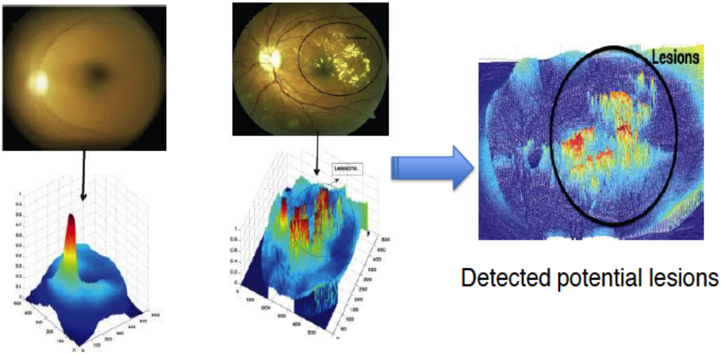
Telemedical Retinal Image Analysis and Diagnosis (TRIAD) project



University of Tennessee Health Science Center (UTHSC) & Oak Ridge National Laboratory (ORNL)

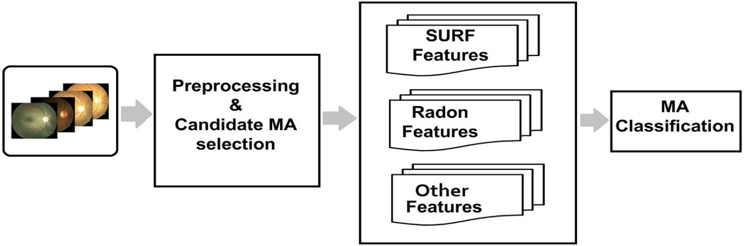
Exudates detection

An atlas based exudates detection method [2]



2. S. Ali, D. Sidibé, K. Adal, L. Giancardo, E. Chaum, T. P. Karnowski, F. Mériaudeau, "*Statistical atlas based exudate segmentation*", **Computerized Medical Imaging and Graphics, vol. 37(5), pp. 358-368, 2013**

A semi-supervised approach for MA detection [3]



3. K. Adal, D. Sidibé, S. Ali, E. Chaum, T. Karnowski, F. Mériaudeau,"*Automated Detection of Microaneurysms Using Scale-Adapted Blob Analysis and Semi-Supervised Learning*", **Computer Methods and Programs in Biomedicine, 114(1), pp. 1-10, 2014**

Drusen vs Exudates

- Diabetic Macular Edema (DME) is a complication of DR
  - blurred vision due to swelling of the macula
  - assessed by detecting **exudates**
- Age related macular degeneration (AMD or ARMD) is a eye condition related to age
  - loss of vision in the macula
  - assessed by detecting **drusen**

Drusen vs Exudates

Exudates

- small white or yellowish white deposits of lipid
- sign of DME

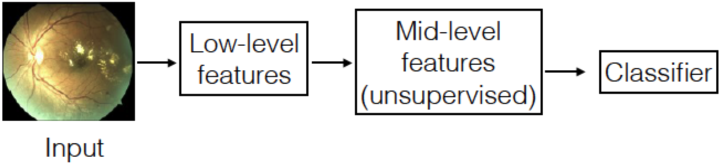
Drusen

- variable size yellowish white deposits of lipid
- earliest signs of ARMD

Distinguishing between exudates and drusen is important

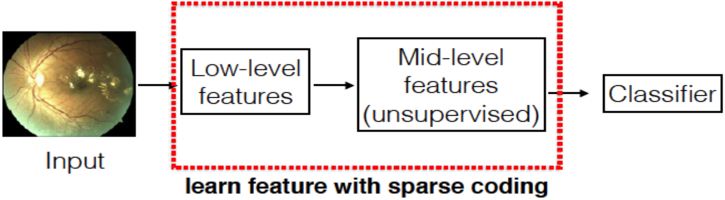
Retinal images classification

Main framework used in literature



- Pre-processing
  - vessels segmentation, optic disc removal, etc
- Low-level features
  - Color, texture, edges, etc
- Mid-level representation
  - Clustering, Bag-of-visual-words (BoW)

What we would like to do

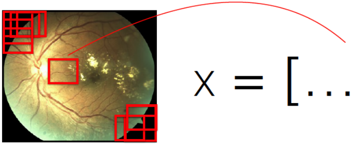


Extract discriminative features for retinal images classification

- No complex pre-processing

Sparse features extraction

- Extract local patches form the images
- Put each patch as column of the matrix **X**
- Learn a dictionary **D** and a matrix **A** such that **X** $\simeq$ **DA** (using K-SVD algorithm)



$$X = [\ldots|\ldots]$$

$$\underbrace{\begin{bmatrix} & & \\ & \mathbf{X} & \\ & & \end{bmatrix}_{d\times N}}_{\text{training data matrix}} = \underbrace{\begin{bmatrix} & & \\ & \mathbf{D} & \\ & & \end{bmatrix}_{d\times K}}_{\textit{dictionary}} \underbrace{\begin{bmatrix} & & \\ & \mathbf{A} & \\ & & \end{bmatrix}_{K\times N}}_{\text{sparse codes matrix}}$$

Sparse features extraction

1. **Coding**
   For a given set of features $\mathbf{X}$ from an image I, find $\mathbf{A}$

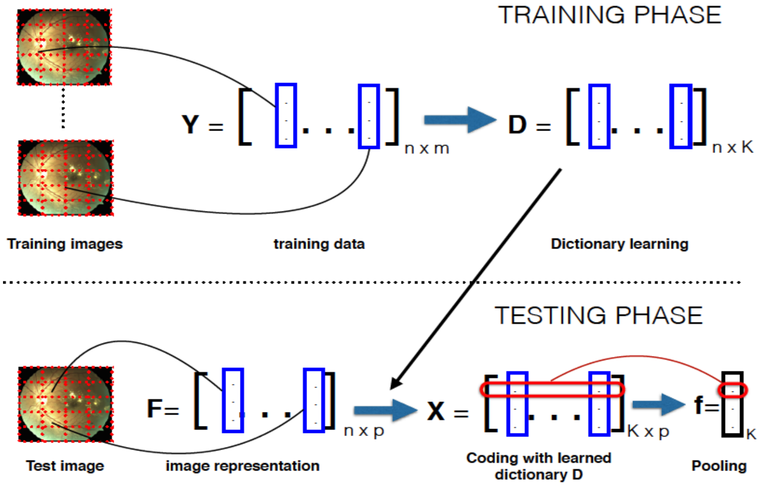
$$\begin{bmatrix} & & \\ & \mathbf{X} & \\ & & \end{bmatrix}_{d\times N} = \begin{bmatrix} & & \\ & \mathbf{D} & \\ & & \end{bmatrix}_{d\times K} \begin{bmatrix} & & \\ & \mathbf{A} & \\ & & \end{bmatrix}_{K\times N}$$

2. **Pooling**
   From $\mathbf{A}$ find a single feature vector $\mathbf{f}$

$$\begin{bmatrix} & & \\ & \mathbf{A} & \\ & & \end{bmatrix}_{K\times N} \Longrightarrow \begin{bmatrix} \vdots \\ \mathbf{f_i} \\ \vdots \end{bmatrix}_{K\times 1} \qquad \forall i,\ \mathbf{f_i} = g(\mathbf{A_{i,:}})$$

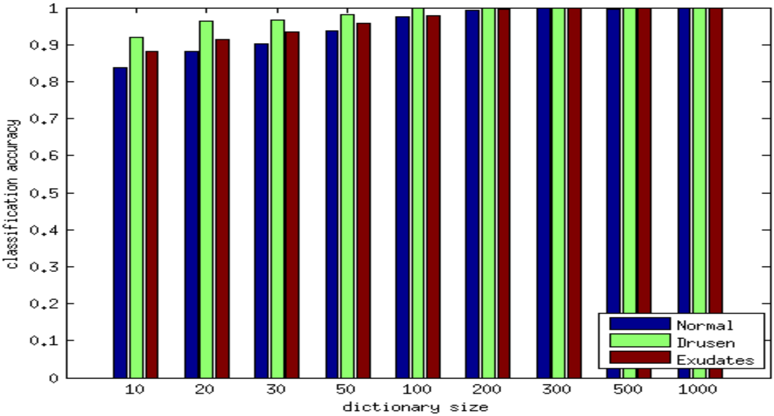
g can be *max* or *average*

Sparse features extraction



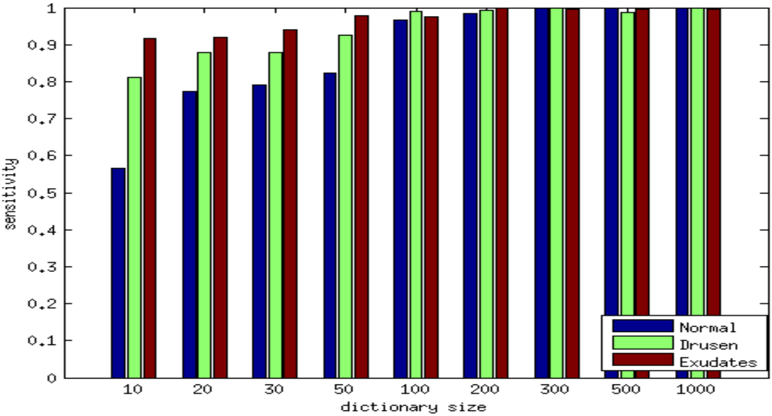
Classification results

Accuracy

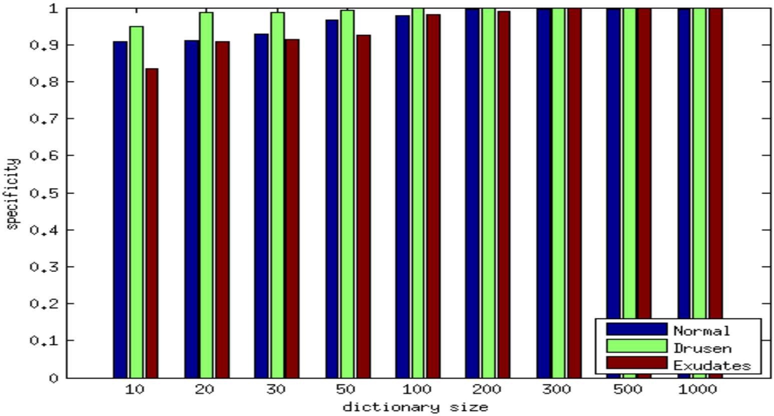


Classification results

Sensitivity



Specificity



Comparison with Bag of Words approach

		Dictionary size			
		50	100	500	1000
Proposed method	Acc	93.70 (±3.71)	97.50 (±2.84)	99.40 (±0.97)	99.80 (±0.63)
	Sens	92.40 (±5.33)	96.50 (±5.76)	98.50 (±3.17)	100 (±0)
	Spec	96.60 (±3.17)	97.70 (±3.50)	99.70 (±0.95)	99.70 (±0.95)
Bag-of-Words	Acc	93.70 (±2.58)	95.30 (±2.06)	97.20 (±2.04)	97.70 (±2.06)
	Sens	90.20 (±8.11)	87.30 (±12.59)	92.50 (±6.57)	92.20 (±12.04)
	Spec	94.60 (±3.50)	96.60 (±3.50)	98.20 (±1.55)	98.80 (±1.55)

More results in Sidibé *et al*. Computers in Biology an Medicine, 2015.

Outline

Conclusions

About PCA

- PCA is a key technique that everyone should know and understand :)
- It is useful in many areas
- Many extensions exist :
  - kPCA : widely used in classification
  - PPCA : can be used online (streming data) and handle missing data
  - MPCA : interesting for multi-dimensional data
- PCA is closely related to SVD
- MPCA is closely related to higher order SVD

Another view of PCA

- PCA can also be viewed as an unsupervised dictionary learning technique
- Given a set of features $\mathbf{X}$, we find a set of vectors (the dictionary) $\mathbf{V}$ such that the data is un-correlated when represented in $\mathbf{V}$

$$\mathbf{V} = \begin{bmatrix} | & | & & | \\ \mathbf{v}_1 & \mathbf{v}_2 & \dots & \mathbf{v}_K \\ | & | & & | \end{bmatrix}_{d \times K}$$

- In general, $K \ll d$, so that we reduce the dimensionality of the data
- Each feature $\mathbf{x}_i$ is represented by $\mathbf{V}^T \mathbf{x}_i$

Conclusions

About dictionaries

- PCA finds a set of K vectors such that $K \leq d$
  - When $K < d$, we say that we have an **under-complete** dictionary
  - When $K = d$, we say that we have a **complete** dictionary
- With the BoW approach, we will usually have large dictionaries, $K > d$
  - When $K > d$, we say that we have an **over-complete** dictionary

Conclusions

About Sparse Coding

- Sparse coding has shown excellent results in various applications
- It relates to current understanding of visual information processing in HVS
- It forms the basis of deep learning architectures (sparse auto-encoders, etc)
- It is been widely used in computer vision and pattern recognition
  - The concept has been extended to 3D : shape descriptors and object recognition
- Improvements
  - Structured dictionary learning
  - Fast optimization algorithms
  - Other sparsity priors (other than L_1 norm)

A word about compressive sensing

- Compressed sensing (CS) is based on the same concepts as sparse coding but with a different goal
- Assume $\mathbf{x}$ has been created by $\mathbf{x} = \mathbf{D}\alpha$ with α very sparse

$$\mathbf{Q}\left(\begin{bmatrix} & \mathbf{D} & \end{bmatrix}\begin{bmatrix}\alpha\end{bmatrix} = \begin{bmatrix}\mathbf{x}\end{bmatrix}\right) \Rightarrow \widehat{\mathbf{D}}\alpha = \widehat{\mathbf{x}}$$

- $\mathbf{Q}$ is called the sensing matrix
- The goal is to recover α from $\widehat{\mathbf{D}}$ and $\widehat{\mathbf{x}}$
- CS focuses on conditions for the recovery to be perfect

Conclusions

From a broader perspective

Matrix factorization

Decomposing each input example as a linear combination of basis vectors

$$\mathbf{X} \approx \mathbf{DA}$$

PCA	variance maximization
ICA	non-Gaussianity (kurtosis) maximization
NMF	non-negativity constraints
Sparse coding	sparsity constraints
...	

TABLE : Different approaches

References

G. Strang (2009).
Introduction to linear algebra.
Wellesley-Cambridge Press and SIAM.

B. Scholkopf, A. Smola, K.R. Müller (1997).
Kernel principal component analysis.
Proc. ICANN 97.

M.E. Tipping, C.M. Bishop (1999).
Probabilistic principal component analysis.
Journal of Royal Statistical Society B, 61 (3), pp. 611-622.

J. yang, D. Zhang, A. Frangi, J. Yang (2004).
Two-dimensional PCA : A new approach to appearance-based face representation and recognition.
IEEE Trans. PAMI, 26(1), pp : 131-37.

H. Lu, K.N. Plataniotis, A.N. Venetsanopoulos (2008).
MPCA : Multilinear principal component analysis of tensor objetcs
IEEE Trans. Neural Networks, 19(1), pp : 18-39.

M. Elad (2010).
Sparse and Redundant Representations : From Theory to Applications in Signal and Image Processing.
Springer.

J. Wright, Y. Ma, J. Mairal, G. Sapiro, A. Zisserman (2010).
Sparse Representation for Computer Vision an Pattern Recognition.
Proceedings of the IEEE, 98(6), pp : 1031-1044.