Embeddings II: Sparse & Kernel PCA, Dictionary Learning

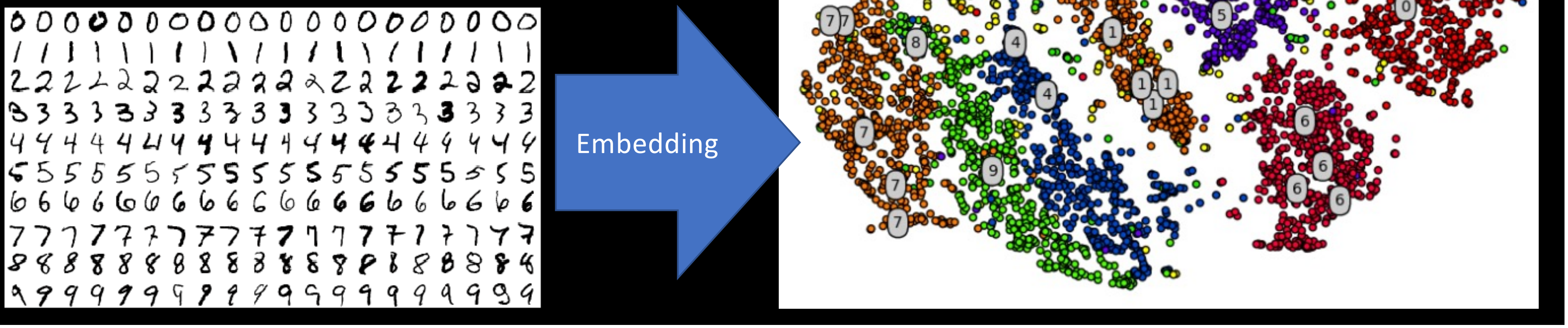
CSCI 4360/6360 Data Science II

Embeddings

- What is an *embedding*?

$$f : X \to Y$$

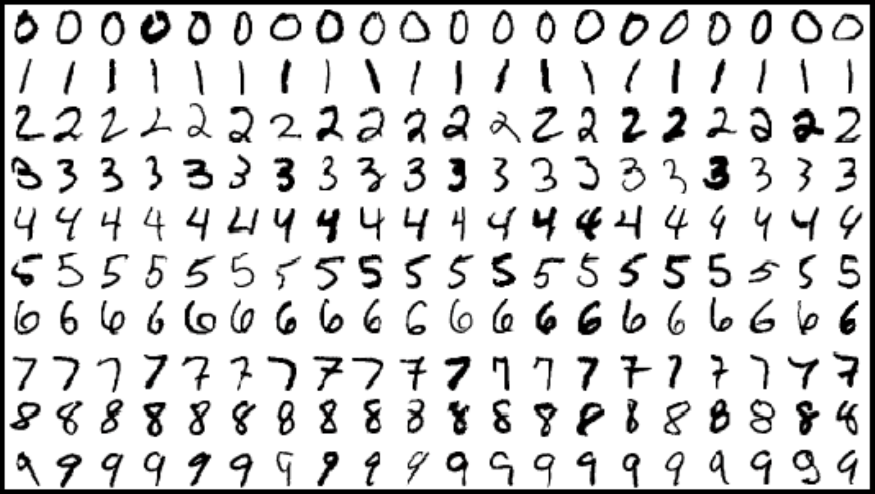
- **Mapping**
- **Transformation**
- **Reveals / preserves "structure"**



Embedding

Embeddings

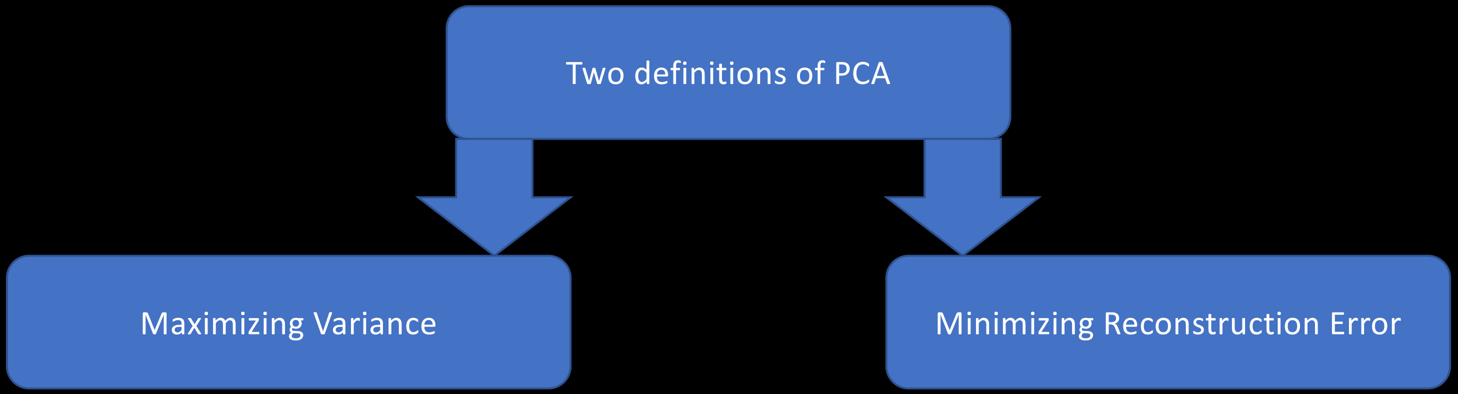
- “Degrees of freedom” versus “intrinsic dimensionality”



- Despite 64x64 pixels, only so many ways to draw a 9
- **Low-dimensional manifold**

Principal Component Analysis (PCA)

1. Orthogonal projection of data
2. Lower-dimensional linear space known as the *principal subspace*
3. Variance of the projected data is maximized



Two definitions of PCA

Maximizing Variance

Minimizing Reconstruction Error

Maximizing Variance

- We start with the idea of projection from *D*-dimensions *x* to *M*-dimensions *u*
  - *u* is a unit vector, so $u^T u = 1$.
- Mean of projected data is $u^T \bar{x}$, where $\bar{x} = \frac{1}{N} \sum_{n=1}^{N} x_n$
- Variance of the projected data $$\frac{1}{N} \sum_{n=1}^{N} \{\vec{u}_1^T \vec{x}_n - \vec{u}_1^T \overline{x}\}^2 = \vec{u}_1^T S \vec{u}_1$$
- where S is the sample covariance matrix of the data $$S = \frac{1}{N} \sum_{n=1}^{N} (\vec{x}_n - \overline{x})(\vec{x}_n - \overline{x})^T$$

Maximizing Variance

- We want to maximize projected variance $u_1^T S u_1$ with respect to u_1
- Obvious problem: needs to be constrained, or else $||u_1||$ -> ∞

- Appropriate constraint: $u_1^T u_1 = 1$, enforced with Lagrange multiplier

$$\vec{u}_1^T S \vec{u}_1 + \lambda_1 (1 - \vec{u}_1^T \vec{u}_1)$$

- Set derivative with respect to $u_1 = 0$, and a stationary point appears

$$S\vec{u}_1 = \lambda_1 \vec{u}_1$$

- Means u_1 must be an eigenvector of S! Left-multiply by u_1^T

$$\vec{u}_1^T S \vec{u}_1 = \lambda_1$$

- Variance will be max when these are 1st eigenvalue & eigenvector

Minimizing Error

- We want the reconstruction error using the first *M* < *D* principal components to be minimal

$$J = \frac{1}{N}\sum_{n=1}^{N} ||\vec{x}_n - \tilde{x}_n||^2$$

We want to minimize *J*

- This can be rewritten purely in terms of eigenvectors u_i

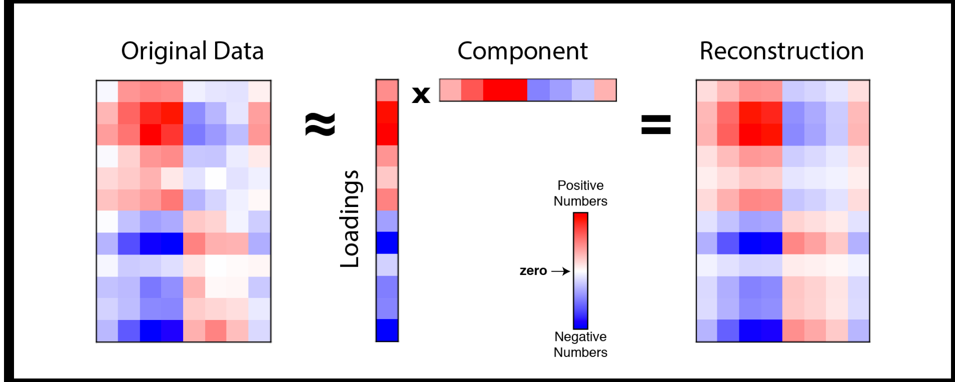
$$J = \sum_{n=M+1}^{D} \vec{u}_i^T S \vec{u}_i$$

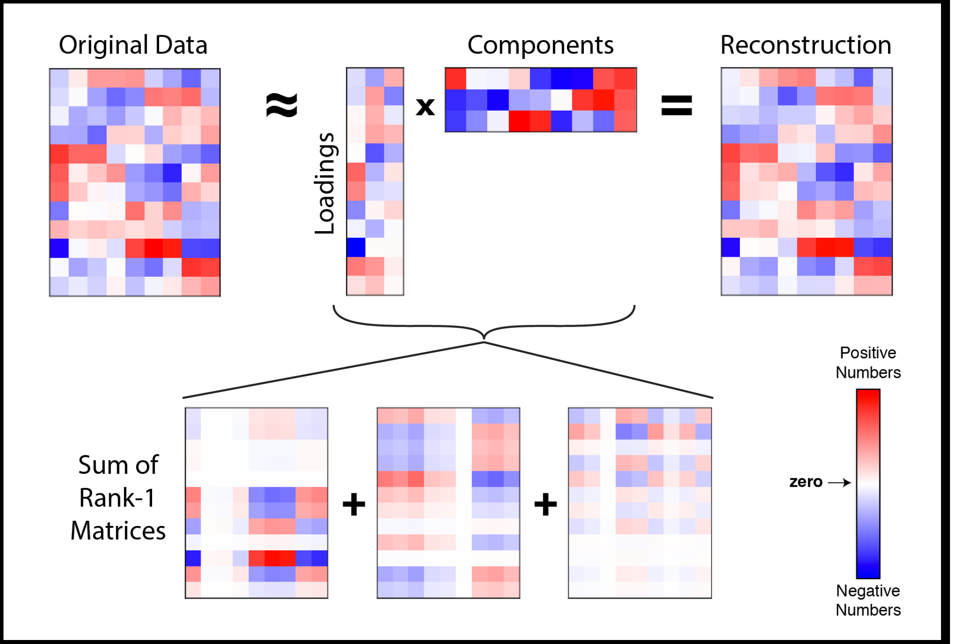
Eigenvectors u_i come out of equation for $\tilde{x}$

- Therefore, the distortion measure of reconstruction using the *M* eigenvectors of the largest eigenvalues is the sum of the remaining *D* - *M* eigenvalues

$$J = \sum_{n=M+1}^{D} \lambda_i$$

Minimizing Error





Principal Component Analysis

Advantages

- Optimal low-rank approximation in terms of squared reconstruction error
- Completely unsupervised
- Endless applications

Disadvantages

- Principal components are linear combinations (cannot generate nonlinear PCs; struggles to determine PCs in geodesic spaces)
- Basis vectors are dense and sometimes difficult to interpret

Kernel PCA

- Whenever we compute a *kernel*, we rely on a scalar (dot) product of the form $x^T x$
- Conventional PCA is an outer product (covariance), $X^T X$
- What if we replaced this with an inner product, XX^T
  - This “Gram matrix” is what we compute eigenvectors of in PCA anyway
- If anything, Kernel PCA is a *generalization* of PCA to arbitrary similarity (kernel) functions!
- **First step**: express conventional PCA such that data vectors *x* appear only in the form of scalar products

Kernel PCA

- Recall that the principal components are defined by eigenvectors of the covariance matrix

$$S\vec{u}_i = \lambda_i \vec{u}_i$$

i is the dimensional index

- and sample covariance matrix defined by

$$S = \frac{1}{N}\sum_{n=1}^{N} \vec{x}_n \vec{x}_n^T$$

N is the number of data points

- and eigenvectors are normalized such that

$$\vec{u}_i^T \vec{u}_i = 1$$

Kernel PCA

- In kernel PCA, we consider data that have already undergone a nonlinear transformation:

$$\vec{x} \in \mathcal{R}^D \quad \Longrightarrow \quad \phi(\vec{x}) \in \mathcal{R}^M$$

- **We now perform PCA on this new *M*-dimensional feature space**

Kernel PCA

- Sample covariance matrix *C* (now *MxM*)

$$C = \frac{1}{N}\sum_{n=1}^{N}\phi(\vec{x}_n)\phi(\vec{x}_n)^T$$

$$\boxed{C\vec{v}_i = \lambda_i\vec{v}_i}$$

- **Goal:** solve the eigenvector/eigenvalue equation **without having to explicitly operate in the *M*-dimensional feature space**
- Combining the two equations: $\frac{1}{N}\sum_{n=1}^{N}\phi(\vec{x}_n)\phi(\vec{x}_n)^T\vec{v}_i = \lambda_i\vec{v}_i$
- This reduces to

$$\vec{v}_i = \sum_{n=1}^{N} a_{in}\phi(\vec{x}_n)$$

Kernel PCA

- Substitute back into eigenvector equation and we get a royal mess

$$\frac{1}{N}\sum_{n=1}^{N}\phi(\vec{x}_n)\phi(\vec{x}_n)^T\sum_{m=1}^{N}a_{im}\phi(\vec{x}_m)=\lambda_i\sum_{n=1}^{N}a_{in}\phi(\vec{x}_n)$$

- Remember our goal: work only in terms of $k(x_n, x_m) = \phi(x_n)^T\phi(x_m)$
- Multiply both sides by $\phi(x_l)$

$$\frac{1}{N}\sum_{n=1}^{N}k(\vec{x}_l,\vec{x}_n)\sum_{m=1}^{N}a_{im}k(\vec{x}_n,\vec{x}_m)=\lambda_i\sum_{n=1}^{N}a_{in}k(\vec{x}_l,\vec{x}_n)$$

Kernel PCA

$$\frac{1}{N}\sum_{n=1}^{N} k(\vec{x}_l,\vec{x}_n)\sum_{m=1}^{N} a_{im}k(\vec{x}_n,\vec{x}_m)=\lambda_i\sum_{n=1}^{N} a_{in}k(\vec{x}_l,\vec{x}_n)$$

- Look familiar?

$$K^2\vec{v}_i=\lambda_i K\vec{v}_i$$

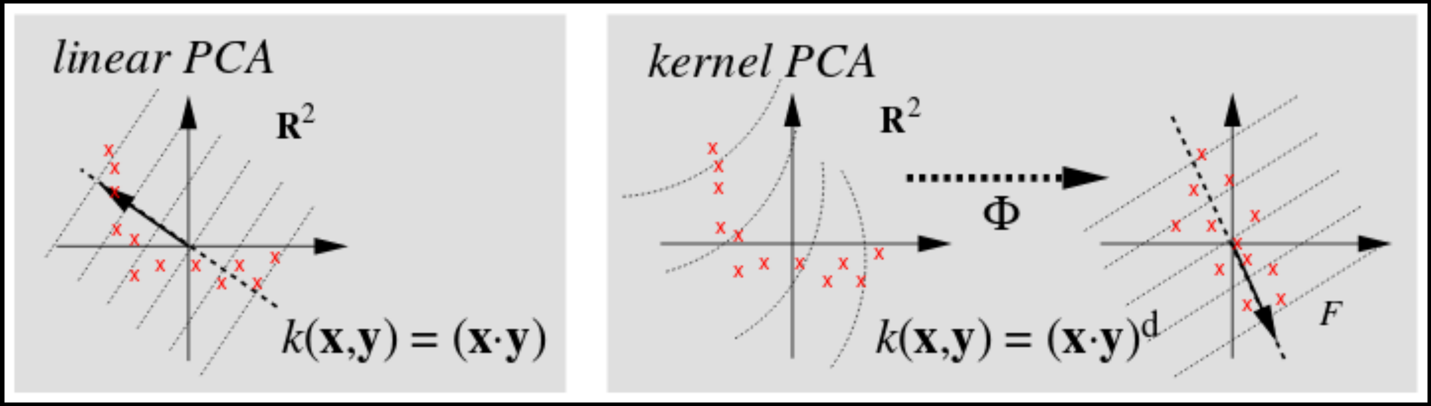
- Which reduces to

$$\boxed{K\vec{v}_i=\lambda_i\vec{v}_i}$$

- (there’s some normalization magic that has to happen but we’re skipping that for now)

Kernel PCA

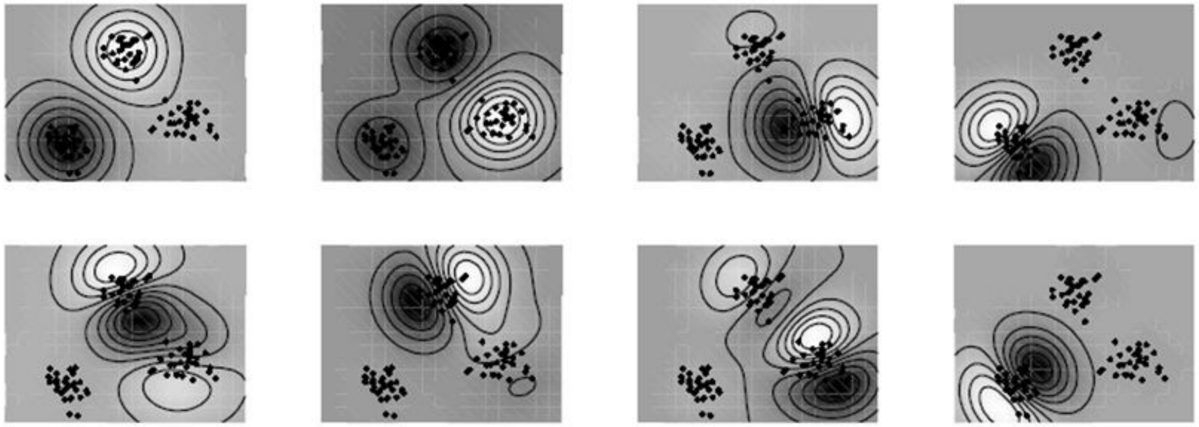
- Data in original data space (right panel, left subpanel) projected by nonlinear transformation into feature space (right subpanel). By performing PCA on feature space, PCs correspond to nonlinear projections in original data space.


linear PCA
$\mathbf{R}^2$
$k(\mathbf{x},\mathbf{y}) = (\mathbf{x}\cdot\mathbf{y})$
kernel PCA
$\mathbf{R}^2$
Φ
$k(\mathbf{x},\mathbf{y}) = (\mathbf{x}\cdot\mathbf{y})^d$
F

Kernel PCA

- Gaussian kernel applied to 2D data
- First 8 kernel PCs
- Contours are lines along which the projection onto the corresponding PC is constant

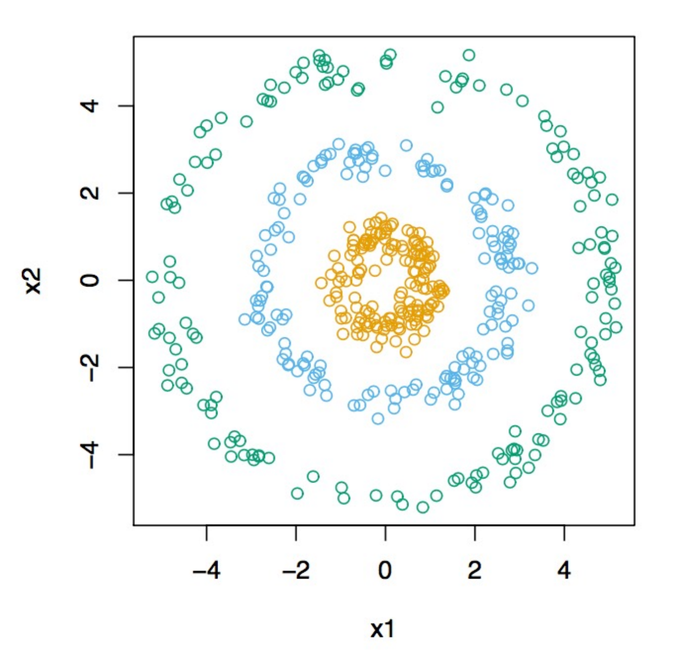
$$k(\mathbf{x}, \mathbf{y}) = \exp\left(-\gamma \|\mathbf{x} - \mathbf{y}\|^2\right)$$



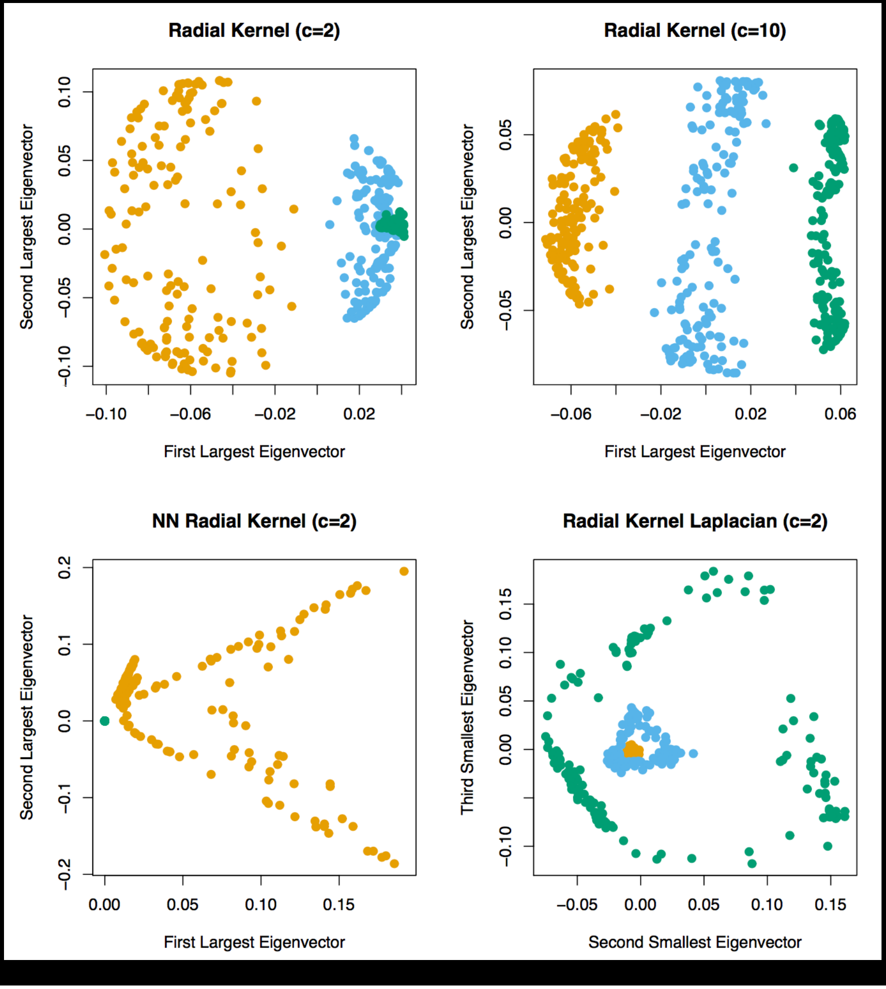
(courtesy of Bernhard Schölkopf)

Kernel PCA

Data



Kernelized PCs



Kernel PCA

Advantages

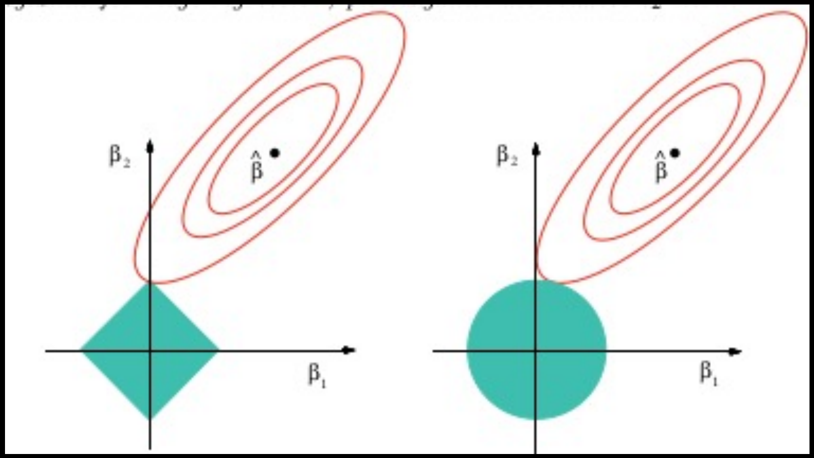
- Allows for nonlinear principal components
- Infinitely flexible in terms of allowed kernel functions

Disadvantages

- Requires finding eigenvectors and eigenvalues of *NxN* matrix, instead of *DxD* (large *N* is problematic)
- Cannot project new, unobserved data onto *L*-dimensional manifold of kernel

Sparse PCA

- Anyone remember lasso regularization?



- Regularization, in general, is a penalty to encourage small weights (remember Assignment 2)
- Lasso (or L_1) forces weights to 0 so they become *sparse*

Sparse PCA

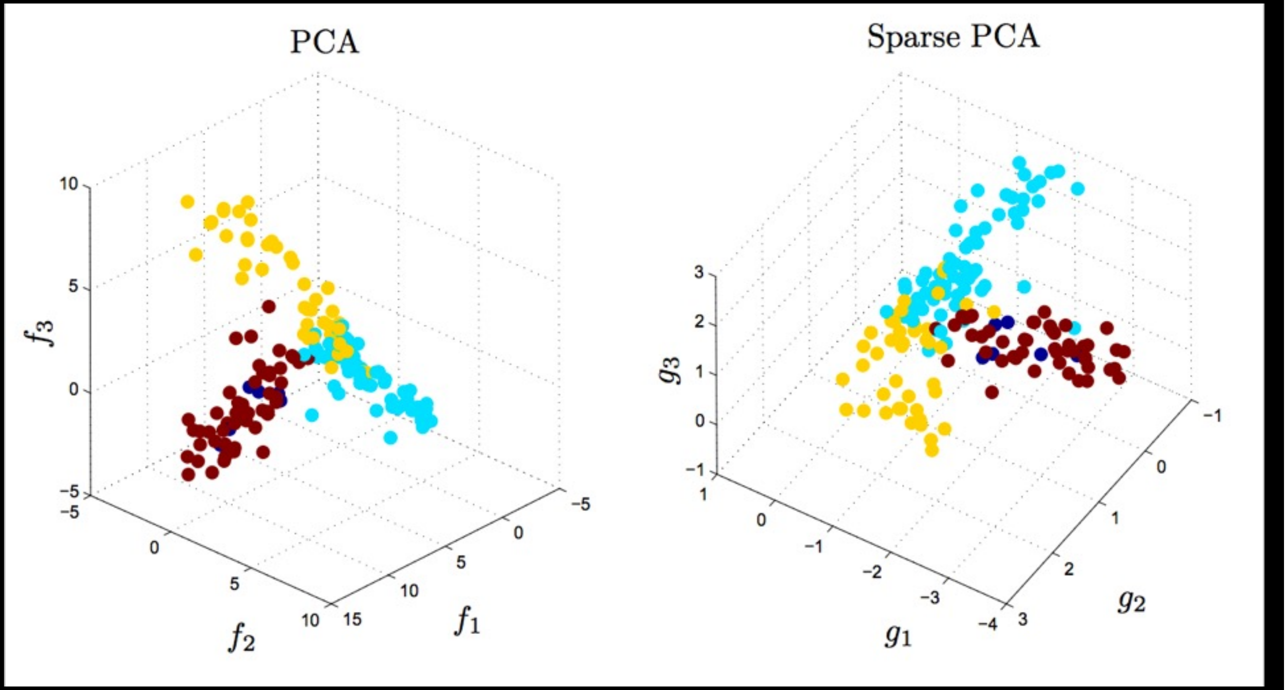
- We still want to maximize $u_i^T S u_i$, subject to $u_i^T u_i = 1$
- …and one more constraint: we want to *minimize* $||u_i||_1$

- Formalize these constraints using Lagrangian multipliers

$$\min_{W,U} ||X - WU^T||_F^2 + \gamma \sum_{n=1}^{N} ||\vec{w}_i||_1 + \gamma \sum_{i=1}^{D} ||\vec{u}_i||_1$$

Sparse PCA

- Qualitatively similar to PCA, but with lots more zeros



Sparse PCA

Advantages

- Simpler and more interpretable components
- Resulting components are very similar to “standard” PCA

Disadvantages

- Optimization procedure is non-convex (often use some version of alternating least-squares)

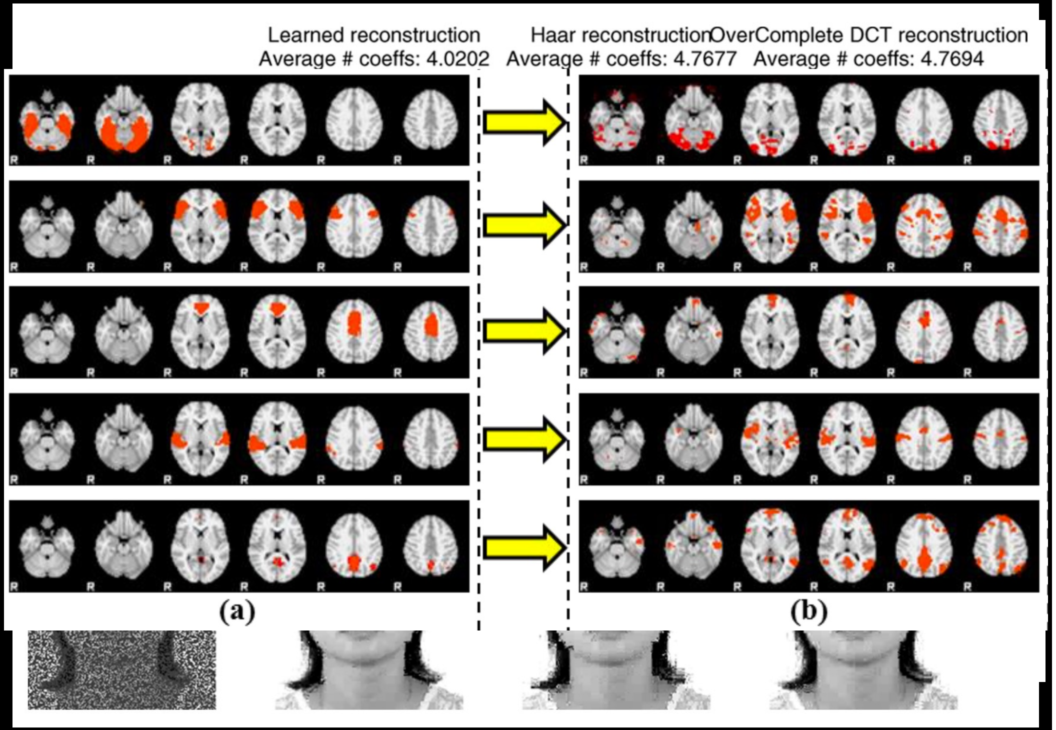
Dictionary Learning

- *"Given a set of signals belonging to a certain class, one wishes to extract the relevant information by identifying the generating causes; that is, recovering the elementary signals (atoms) that efficiently represent the data."*
  - *Regularization, Optimization, Kernels, and Support Vector Machines*, Ch. 2

- Every embedding strategy ever?

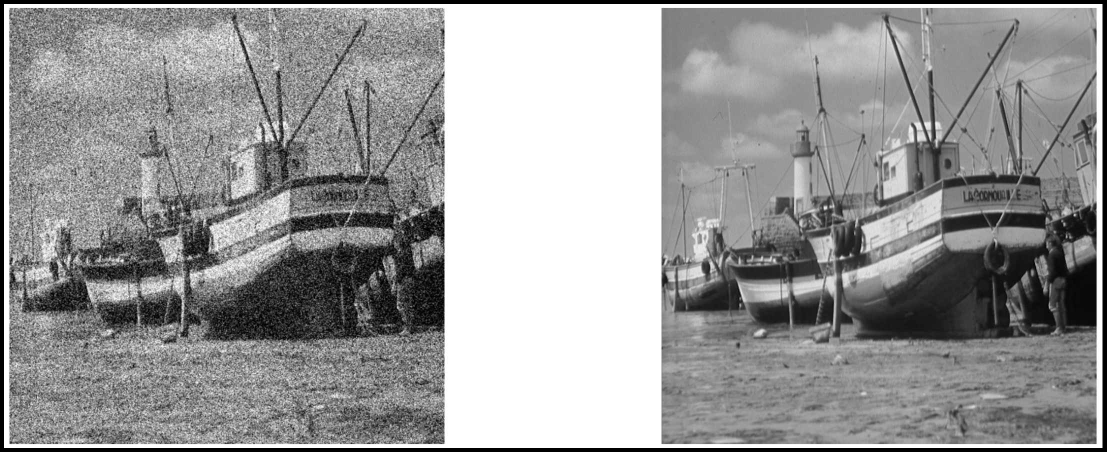
Dictionary Learning

- Sparse coding
- *l*^{p} sparsity
- Hierarchical sparse coding
- K-SVD
- Elastic net



Motivations

- Dictionary learning is ideally formulated for image denoising (and is indeed a major application of dictionary learning)



$$y = x_{orig} + w$$

Measurements (image)

Original image

Noise

Motivations

$$y = x_{orig} + w$$

- Easily converted to an energy minimization problem

$$E(\vec{x}) = ||\vec{y} - \vec{x}||_2^2 + Pr(\vec{x})$$

Energy minimization becomes a MAP estimation!

- Some classical priors

• Smoothness	$\lambda||\mathcal{L}\vec{x}||_2^2$
• Total variation	$\lambda||\nabla\vec{x}||_1^2$
• Wavelet sparsity	$\lambda||W\vec{x}||_1$
• Lasso	$\lambda||\vec{x}||_1$
• …	

Dictionary Learning

- We have our data *X*
- and wish to represent it using some small number *k* atoms (*k* <<< *n*)
- When combined with coefficients, the linear combinations with the atoms should yield a nearly complete representation of *X*

$$X = [\vec{x}_1, \vec{x}_2, \ldots, \vec{x}_n]^T \in \mathbb{R}^{n \times m}$$

$$\vec{x}_i \cong \sum_{j=1}^{k} \theta_{ji} \vec{b}_j, \forall i = 1, \ldots, n$$

$$B = \left[\vec{b}_1, \vec{b}_2, \ldots, \vec{b}_k\right]^T \in \mathbb{R}^{k \times m}$$

$$\Theta = \left[\vec{\theta}_1, \vec{\theta}_2, \ldots, \vec{\theta}_n\right]^T \in \mathbb{R}^{n \times k}$$

Dictionary Learning

- This gives the minimization

$$\min_{B,\Theta} \sum_{i=1}^{n} \left(||\vec{x}_i - B\vec{\theta}_i||_q^q + h(\vec{\theta}_i) \right)$$

  where *h* promotes sparsity in the coefficients, and *B* is chosen from a constraint set

- The general dictionary learning problem then follows

$$\phi(\Theta, B) = \frac{1}{2}||X - B\Theta||_F^2 + h(\Theta) + g(B)$$

  where specific choices of *h* and *g* are what differentiate the different kinds of dictionary learning (e.g. hierarchical, K-SVD, etc)

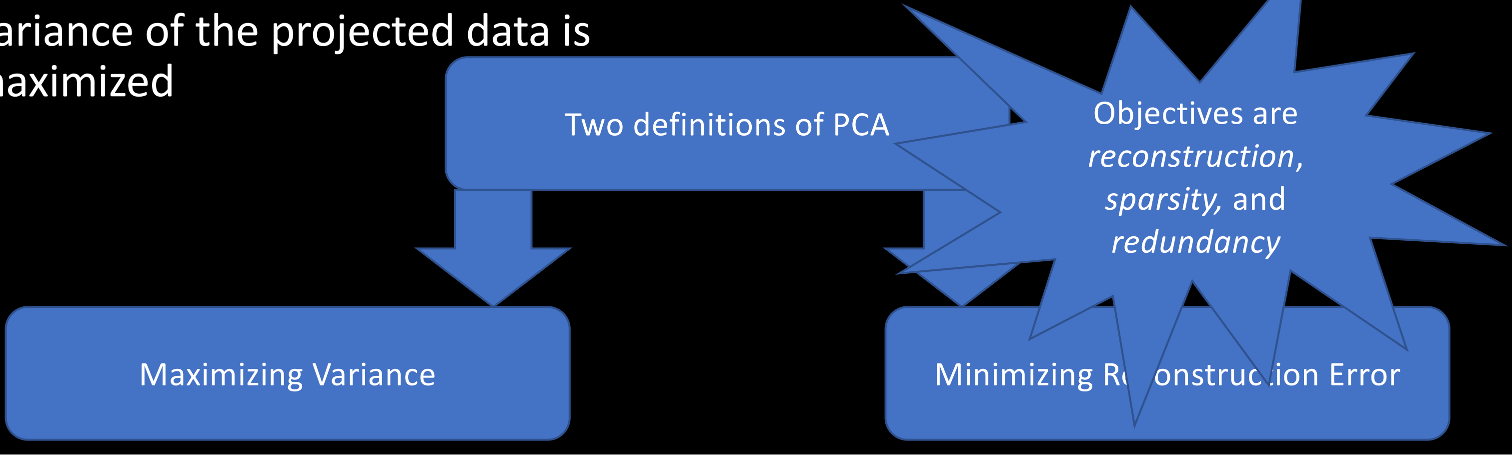
Dictionary Learning vs PCA

- Remember the operational definition of PCA?

1. Orthogonal projection of data
2. Lower-dimensional linear space known as the *principal subspace*
3. Variance of the projected data is maximized

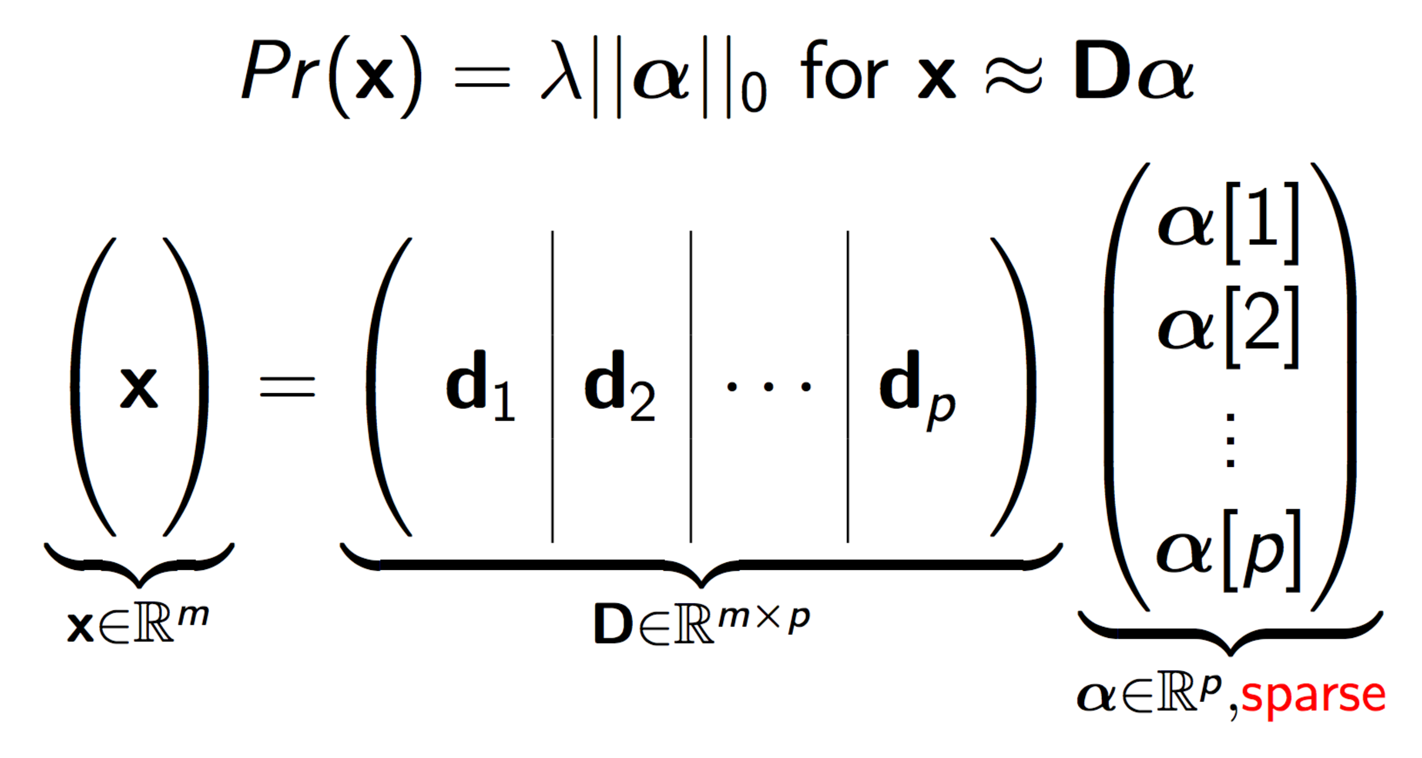
- Dictionary Learning (**sparse** coding)

  1. Minimize reconstruction error
  2. Linear combination of *atoms*
  3. Sparse, overcomplete basis



Objectives are *reconstruction*, *sparsity,* and *redundancy*

Dictionary Learning

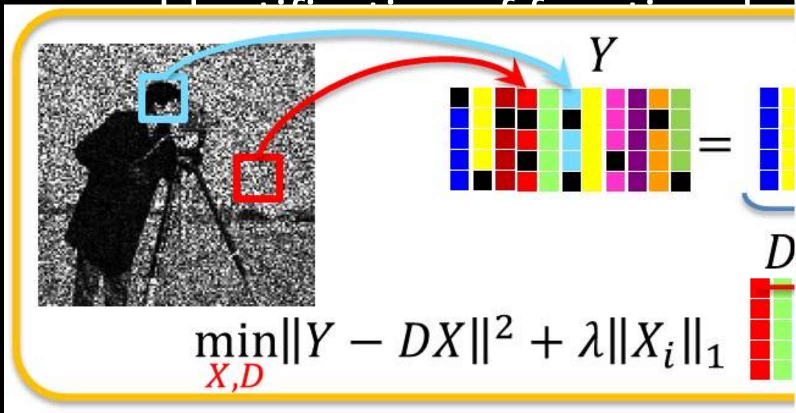


$$Pr(\mathbf{x}) = \lambda||\boldsymbol{\alpha}||_0 \text{ for } \mathbf{x} \approx \mathbf{D}\boldsymbol{\alpha}$$

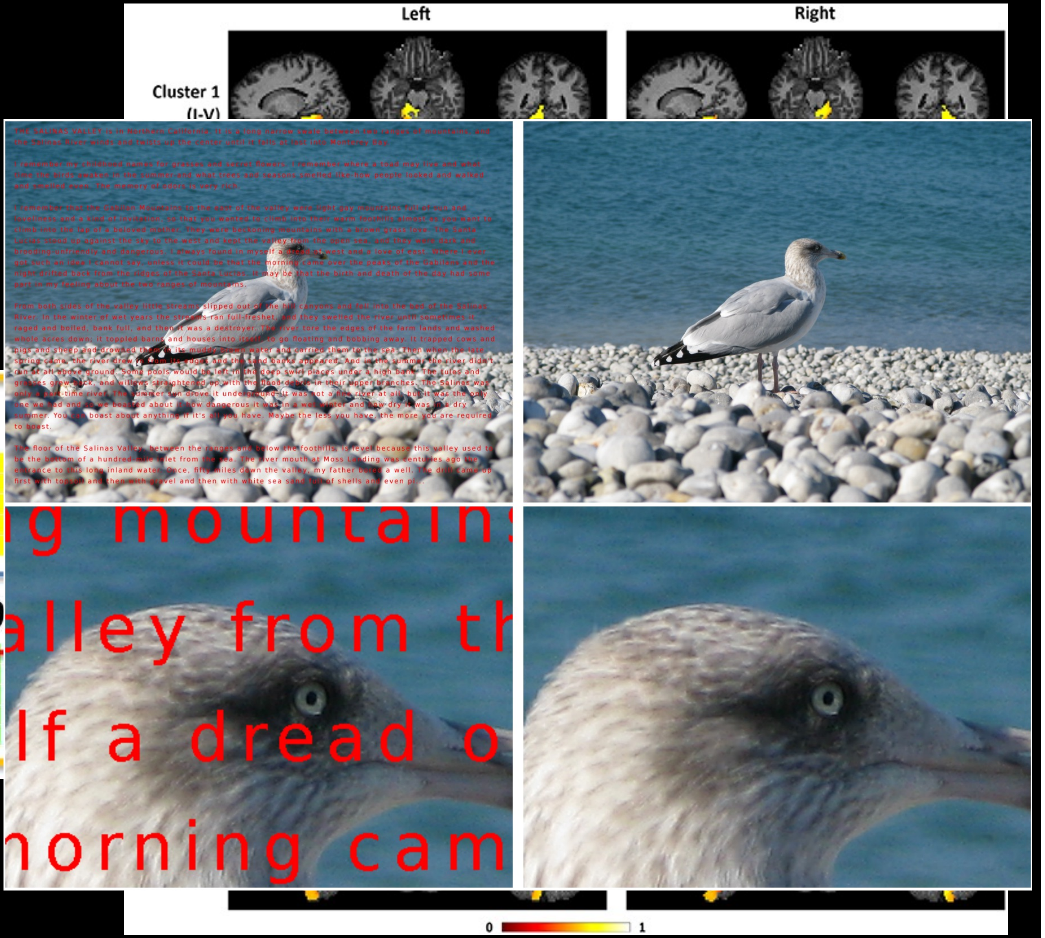
$$\underbrace{\begin{pmatrix} \mathbf{x} \end{pmatrix}}_{\mathbf{x}\in\mathbb{R}^m} = \underbrace{\left(\begin{array}{c|c|c|c} \mathbf{d}_1 & \mathbf{d}_2 & \cdots & \mathbf{d}_p \end{array}\right)}_{\mathbf{D}\in\mathbb{R}^{m\times p}} \underbrace{\begin{pmatrix}\boldsymbol{\alpha}[1]\\ \boldsymbol{\alpha}[2] \\ \vdots \\ \boldsymbol{\alpha}[p]\end{pmatrix}}_{\boldsymbol{\alpha}\in\mathbb{R}^p,\text{sparse}}$$

Applications

- Image denoising
  - Sparse basis forces out noise
- Identification of functional
  object categorization
- Image restoration & inpainting



$$\min_{X,D} \|Y - DX\|^2 + \lambda \|X_i\|_1$$



Dictionary Learning

B is typically implicitly constrained to fall within a *convex set* C of the *k x m* reals, to make optimization tractable

- General formulation

$$\phi(\Theta, B) = \frac{1}{2}||X - B\Theta||^2 + h(\Theta) + g(B)$$

- More common: set *g* to identity*, and *h* to L$_1$ norm

$$\phi(\Theta, B) = \min_{\boxed{B \in \mathcal{C}}} \frac{1}{2} \sum_{i=1}^{n} ||\vec{x}_i - B\vec{\theta}_i||_2^2 + \lambda||\vec{\theta}_i||_1$$

Optimization

- Problems with the objective function?

$$\phi(\Theta, B) = \min_{B \in \mathcal{C}} \frac{1}{2} \sum_{i=1}^{n} ||\vec{x}_i - B\vec{\theta}_i||_2^2 + \lambda ||\vec{\theta}_i||_1$$

- Squared loss is convex
- Regularization is convex

- Squared loss + regularization is **not convex**
- Even worse, often **non-smooth**

Optimization

- Alternating minimization algorithm
  - Two-block Gauss-Seidel

$$A\vec{x} = \vec{b} \qquad A = L_* + U$$

$$\boxed{L_*\vec{x}^{(k+1)} = \vec{b} - U\vec{x}^{(k)}}$$

- Streaming online learning
  - At iteration (or minibatch) *t*, signal x_t and sparse code θ_t are computing using the current dictionary

$$\vec{\theta}_t = \arg\min_{\vec{\theta}} \frac{1}{2}||\vec{x}_t - B_{t-1}\vec{\theta}||_2^2 + \lambda||\vec{\theta}||_1$$

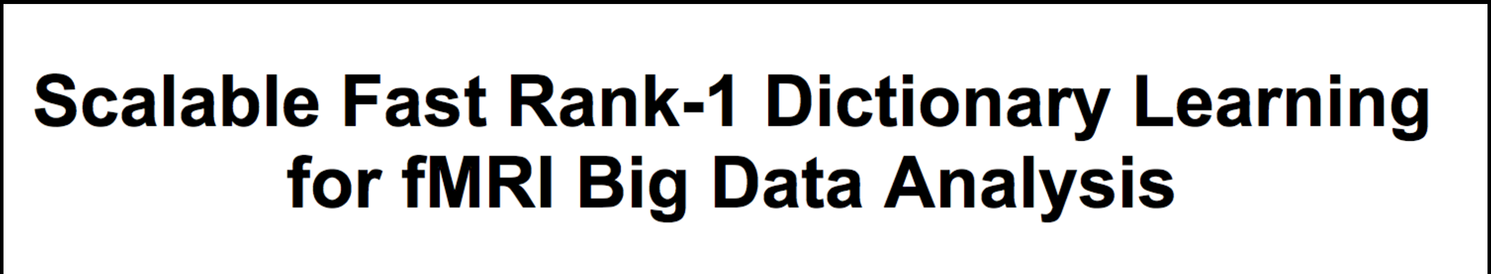
  - Which can then be used to update the dictionary

$$g_t(B) = \frac{1}{t}\sum_{i=1}^{t}\frac{1}{2}||\vec{x}_i - B\vec{\theta}_i||_2^2 + \lambda||\vec{\theta}_i||_1$$

  - *g* can be efficiently solved using block coordinate descent on columns of *B*

Rank-1 Dictionary Learning (R1DL)

- KDD 2016

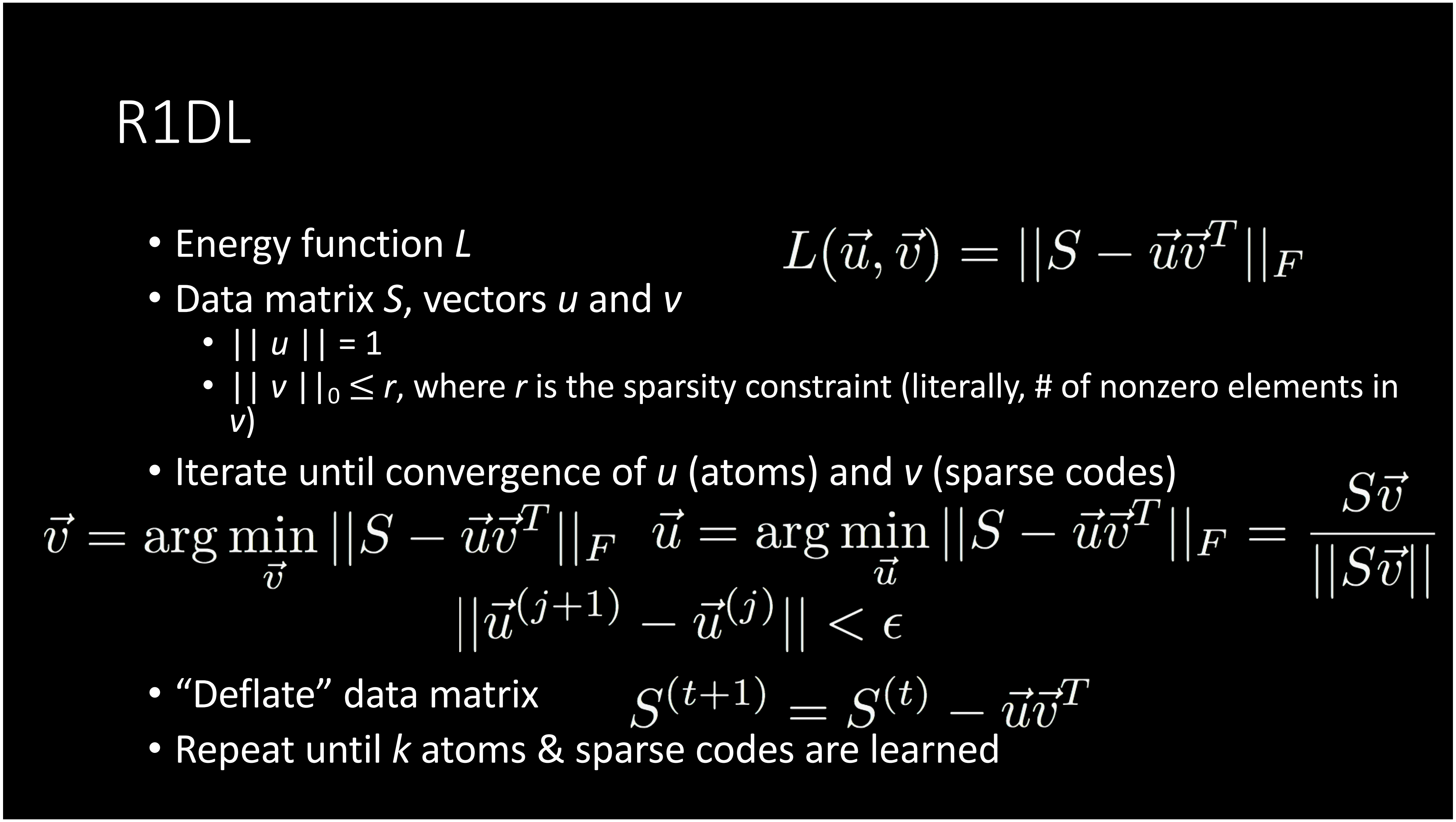


Scalable Fast Rank-1 Dictionary Learning for fMRI Big Data Analysis

- “Scalable fast”

R1DL

- Reformulates dictionary learning as an alternating least-squares problem
    - (embraces the optimization procedure)
- Uses 0-”norm” instead of L_1
    - Given rank-1 formulation, this is an inexpensive way of guaranteeing sparsity
- Iteratively learns rank-1 dictionary atoms until *k* have been found
    - ”Deflates” data matrix on each iteration



R1DL

- Energy function *L* $$L(\vec{u}, \vec{v}) = ||S - \vec{u}\vec{v}^T||_F$$
- Data matrix *S*, vectors *u* and *v*
  - $|| u || = 1$
  - $|| v ||_0 \leq r$, where *r* is the sparsity constraint (literally, # of nonzero elements in *v*)
- Iterate until convergence of *u* (atoms) and *v* (sparse codes)

$$\vec{v} = \arg\min_{\vec{v}} ||S - \vec{u}\vec{v}^T||_F \quad \vec{u} = \arg\min_{\vec{u}} ||S - \vec{u}\vec{v}^T||_F = \frac{S\vec{v}}{||S\vec{v}||}$$

$$||\vec{u}^{(j+1)} - \vec{u}^{(j)}|| < \epsilon$$

- "Deflate" data matrix $$S^{(t+1)} = S^{(t)} - \vec{u}\vec{v}^T$$
- Repeat until *k* atoms & sparse codes are learned

Summary

- Principal Components Analysis
  - Classic dimensionality reduction technique
- Kernel PCA
  - Introduces nonlinearities into component vectors
  - Permits use of arbitrary similarity functions
  - Can capture much richer and more complex interactions in data
  - Much more expensive to compute than PCA
- Sparse PCA
  - Qualitatively similar results to PCA
  - Components are sparse, improving interpretability
  - Learning procedure is non-convex, typically requiring ALS

Summary

- Dictionary learning is focused on developing a basis of *atoms* and *coefficients*
    - Coefficients are *sparse*
    - Atoms form an *overcomplete* representation of the data
    - Chosen to minimize *reconstruction error*
- Explicitly factorizes out noise
    - Can be customized in the form of a prior
- Optimization is often non-convex and non-smooth, requiring alternating minimization strategies or online learning
- R1DL focuses on leveraging optimization strategies to iteratively learn the basis, one atom at a time
- Other variants include K-SVD, Hierarchical DL, and Elastic Net

Questions?

Resources

- http://alexhwilliams.info/itsneuronalblog/2016/03/27/pca/
- *Elements of Statistical Learning*, Chapter 14 http://statweb.stanford.edu/~tibs/ElemStatLearn/printings/ESLII_print10.pdf
- *Pattern Recognition and Machine Learning,* Chapter 12
- *Machine Learning: A Probabilistic Perspective*, Chapter 14
- *An Introduction to Statistical Learning,* Chapter 10 http://www-bcf.usc.edu/~gareth/ISL/ISLR%20Seventh%20Printing.pdf