EECS 545: Machine Learning

Lecture 19. Unsupervised Learning: Nonlinear latent variable models

Honglak Lee

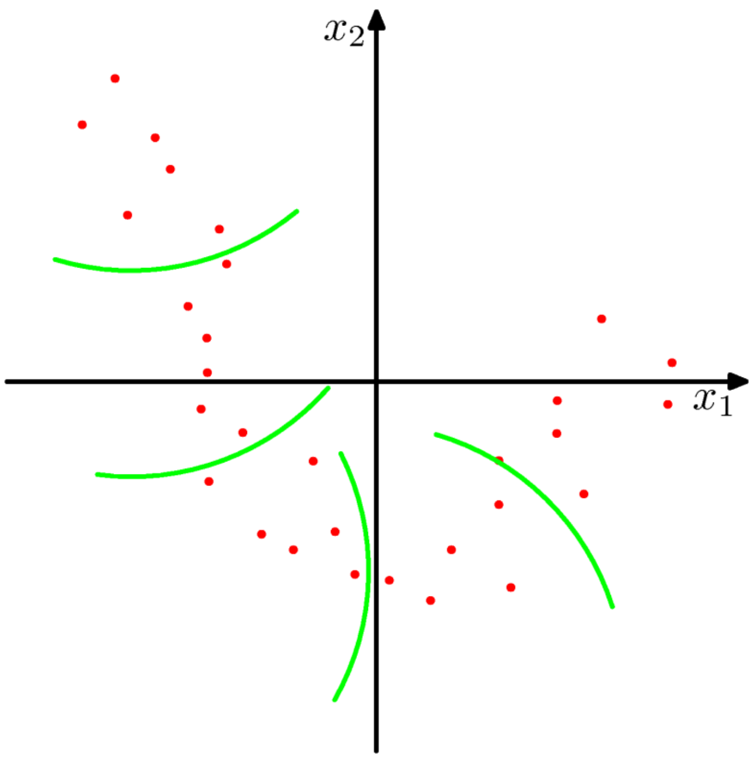
3/21/2011

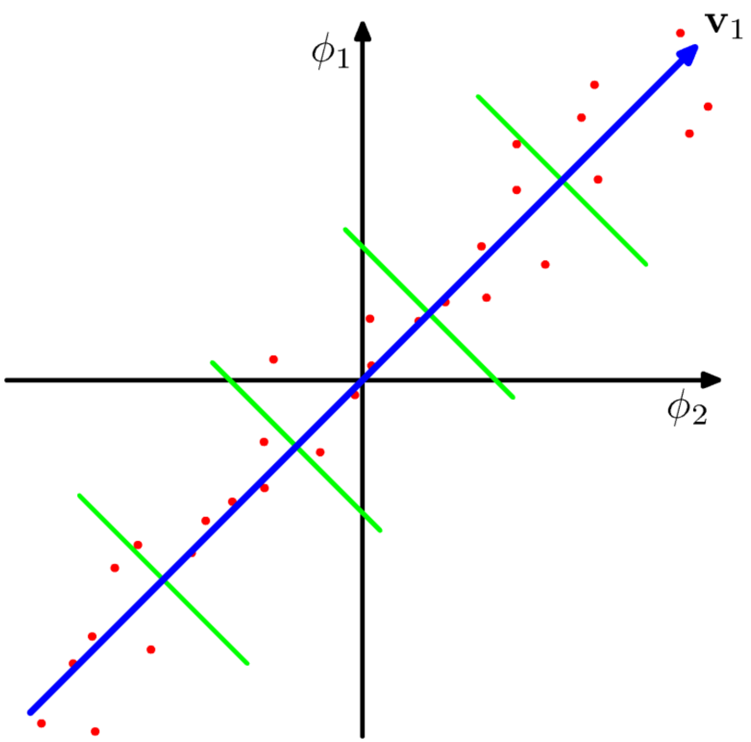
Outline

- Kernel PCA
- Isomap
- LLE
- Independent Component Analysis
- Autoassociative Networks

Kernel PCA

- Suppose the regularity that allows dimensionality reduction is non-linear.





PCA

$\lambda v = \left(\sum_n x_n x_n^T\right) v$

KPCA

$\lambda v = \left(\sum_n \phi(x_n) \phi(x_n)^T\right) v$ $\leftarrow$ $V = \sum \alpha_j \phi(x_j)$

Kernel PCA

- As with regression and classification, we can transform the raw input data $\{x_n\}$ to a set of feature values

$$\{\mathbf{x}_n\} \longrightarrow \{\phi(\mathbf{x}_n)\}$$

K

- Linear PCA gives us a linear subspace in the feature value space, corresponding to nonlinear structure in the data space.

Kernel PCA

- Define a kernel, to avoid having to evaluate the feature vectors explicitly.

$$k(\mathbf{x}, \mathbf{x}') = \phi(\mathbf{x})^T \phi(\mathbf{x}')$$

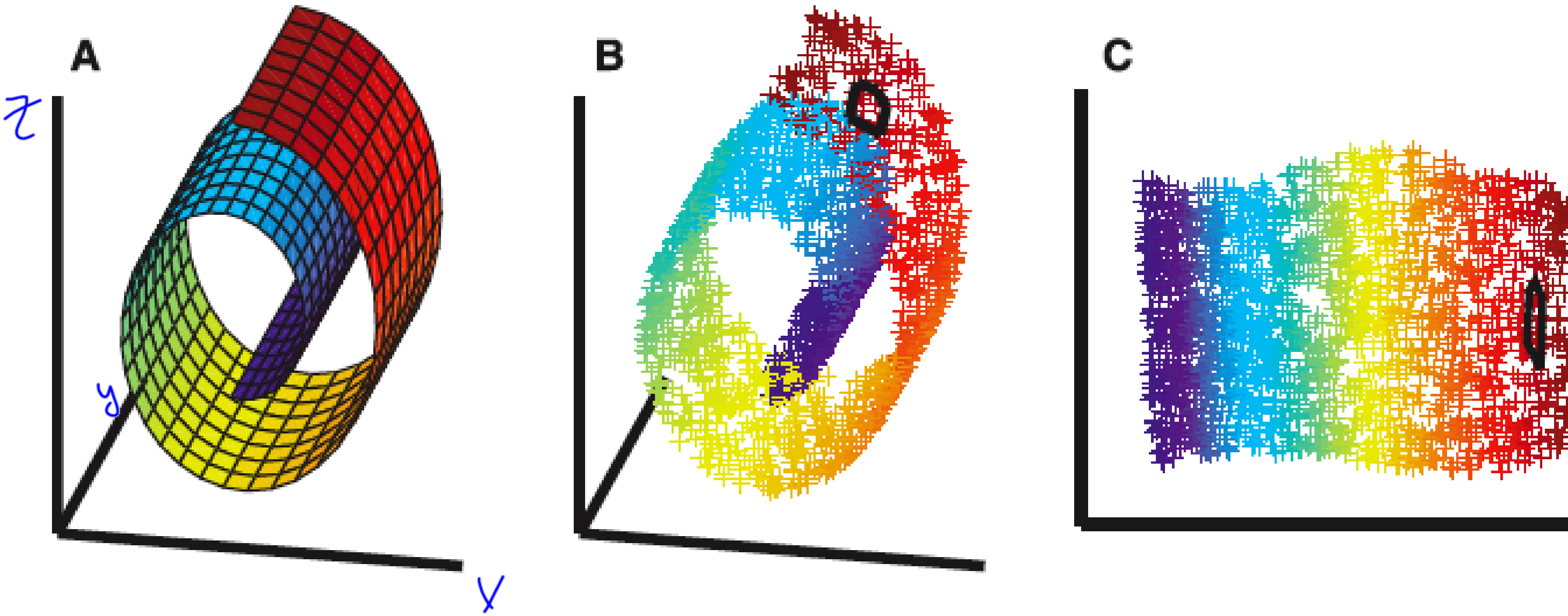
- Define the Gram matrix K of pairwise similarities among the data points:

$$K_{nm} = \phi(\mathbf{x}_n)^T \phi(\mathbf{x}_m) = k(\mathbf{x}_n, \mathbf{x}_m)$$

- Express PCA in terms of the kernel,
  – Some care is required to centralize the data.

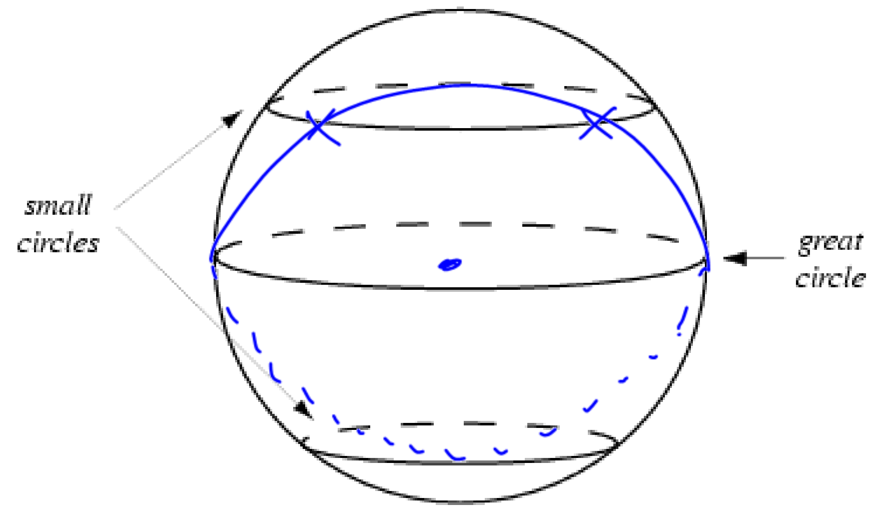
Nonlinear Manifolds

- PCA fails on seriously nonlinear manifolds like the “Swiss roll”.



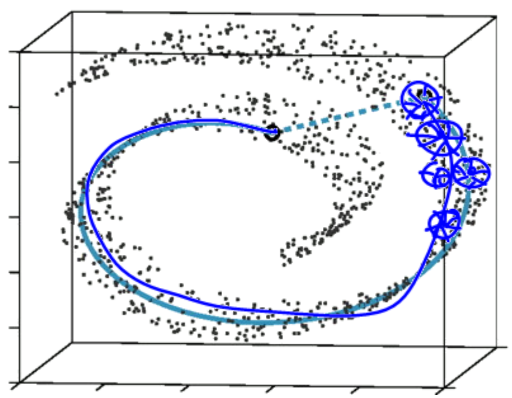
Isometric Feature Mapping (ISOMAP)

- Geodesic :the shortest curve on a manifold that connects two points on the manifold
  - e.g. on a sphere, geodesics are great circles



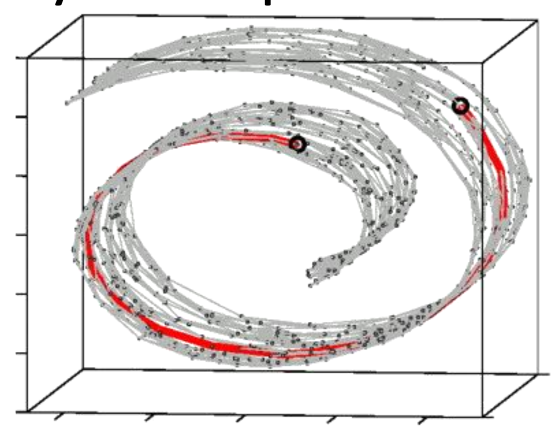
- Geodesic distance: length of the geodesic

- Points far apart measured by geodesic dist. appear close measured by Euclidean dist.



ISOMAP

- Take a distance matrix as input
- Construct a weighted graph G based on neighborhood relations
- Estimate pairwise geodesic distance by “a sequence of short hops” on G

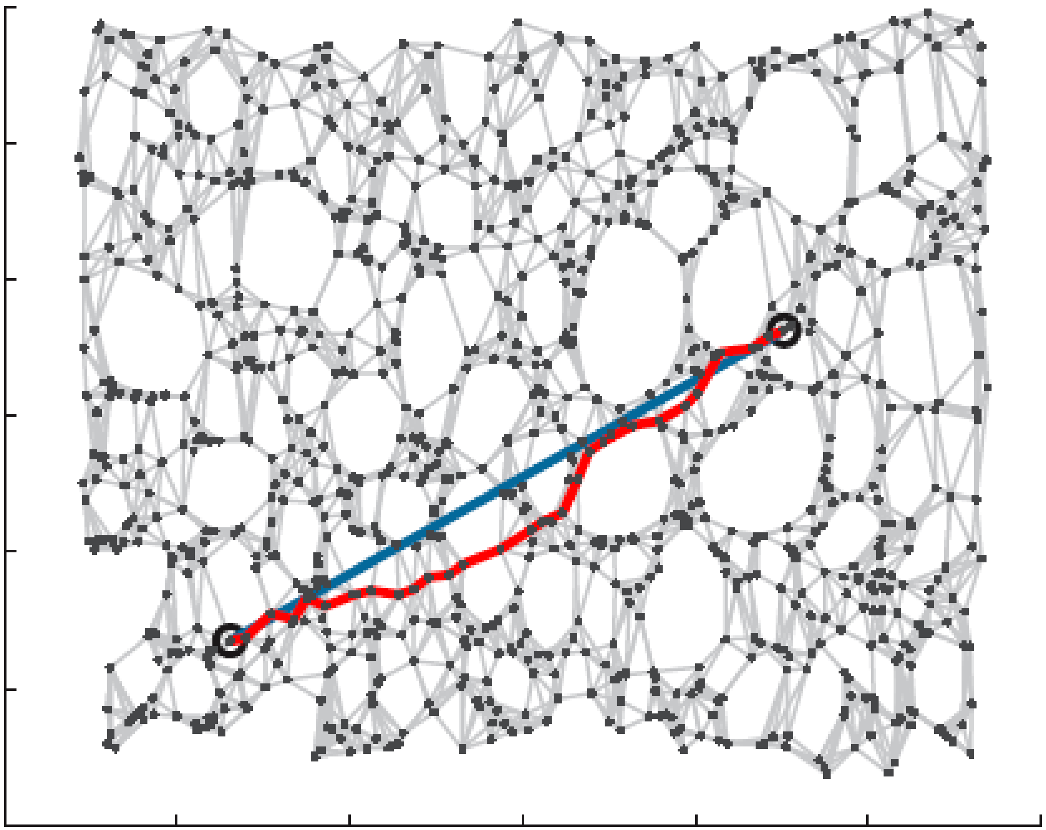


- Apply MDS to the geodesic distance matrix

Multi dimensional Scaling : D (pairwise distance) → projection $W_1, \dots, W_K$

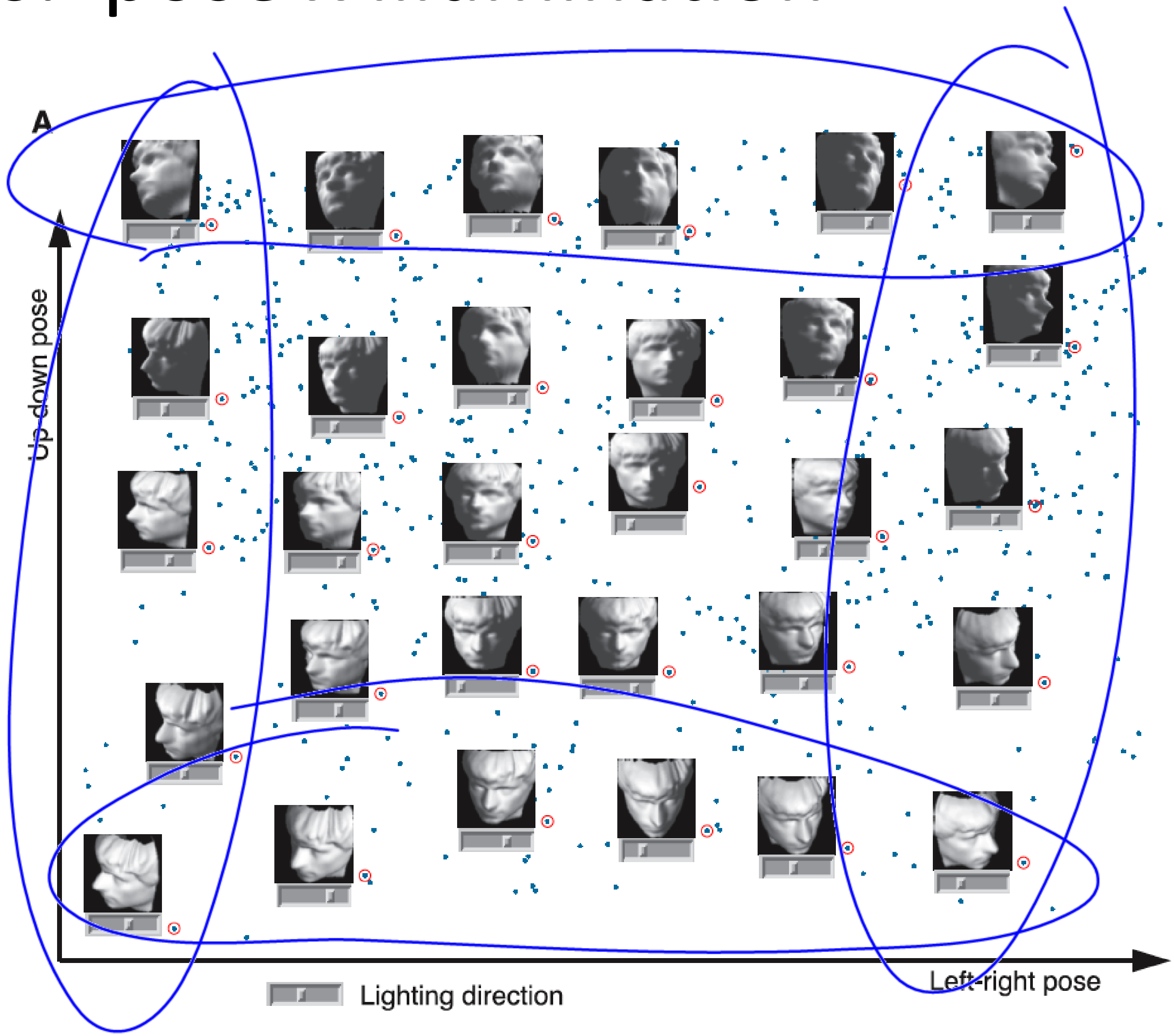
Unrolling the Swiss Roll

- The resulting 2D structure reflects the geodesic distances along the manifold.



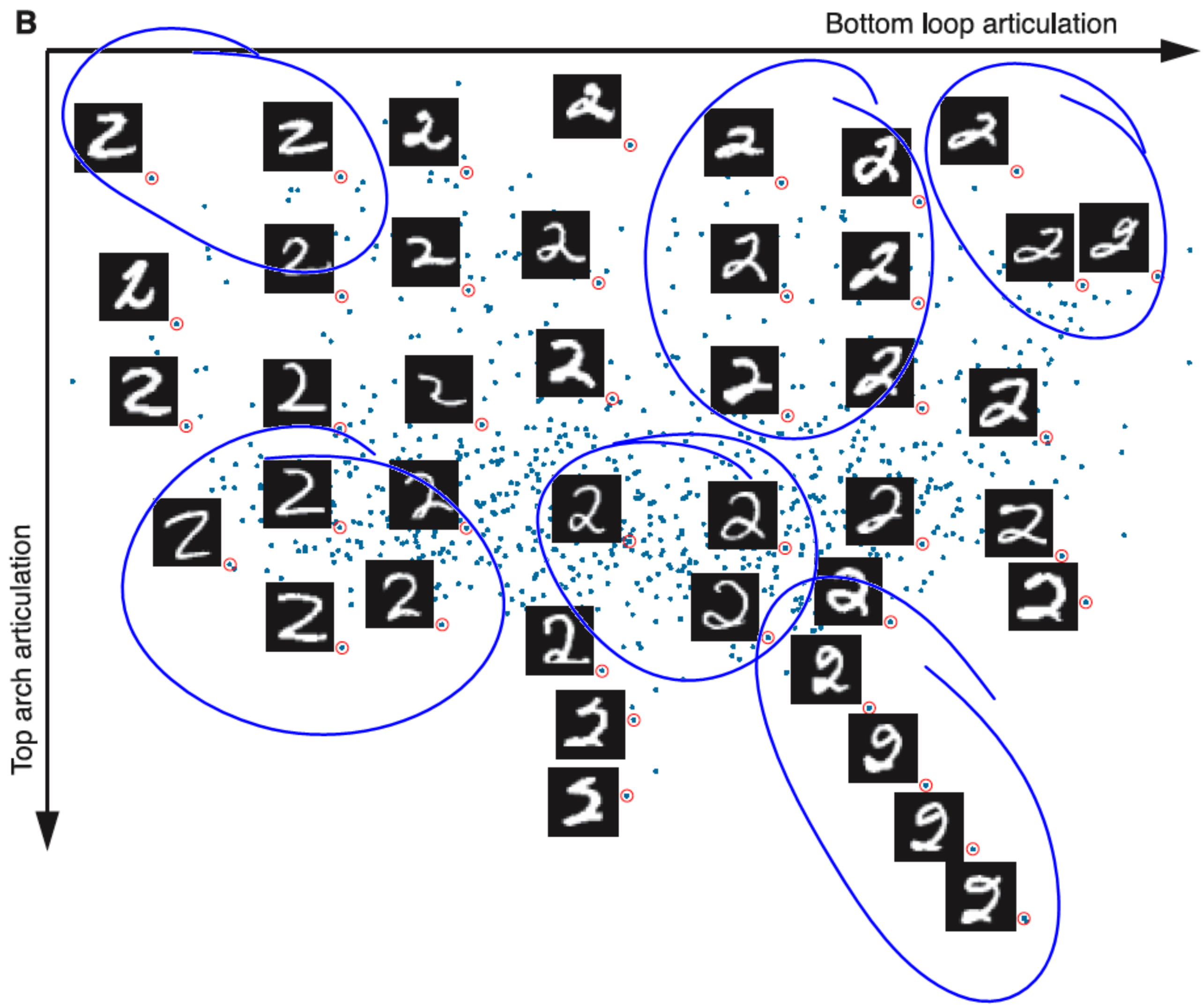
Faces: pose x illumination

- 64x64 = 4096 dimensions



Hand-written “2”s

- Mapping groups the digits by “style”



Locally Linear Embedding (LLE)

- LLE finds the subspace that best preserves the local linear structure of the data

- Assumption: manifold is locally "linear"
  - Each sample in the input space is a linearly weighted average of its neighbors.

- A good projection should best preserve this geometric locality property

Locally Linear Embedding (LLE)

- W: a linear representation of every data point by its neighbors
- Choose W by minimized the reconstruction error

$$\text{minimizing} \sum_{\mathrm{i}=1}^{\mathrm{n}} \left\| \mathrm{x}_{\mathrm{i}} - \sum_{j=1}^{K} \mathrm{W}_{\mathrm{ij}} x_{ij} \right\|^2$$

$$\text{s.t.} \ \sum_{j=1}^{n} \mathrm{W}_{\mathrm{ij}} = 1, \forall \mathrm{x}_{\mathrm{i}} \ ; \ \ \mathrm{W}_{\mathrm{ij}} = 0 \text{ if } \mathrm{x}_{\mathrm{j}} \text{ is not a neighbor of } \mathrm{x}_{\mathrm{i}}$$

- Calculate a neighborhood preserving mapping Y, by minimizing the reconstruction error

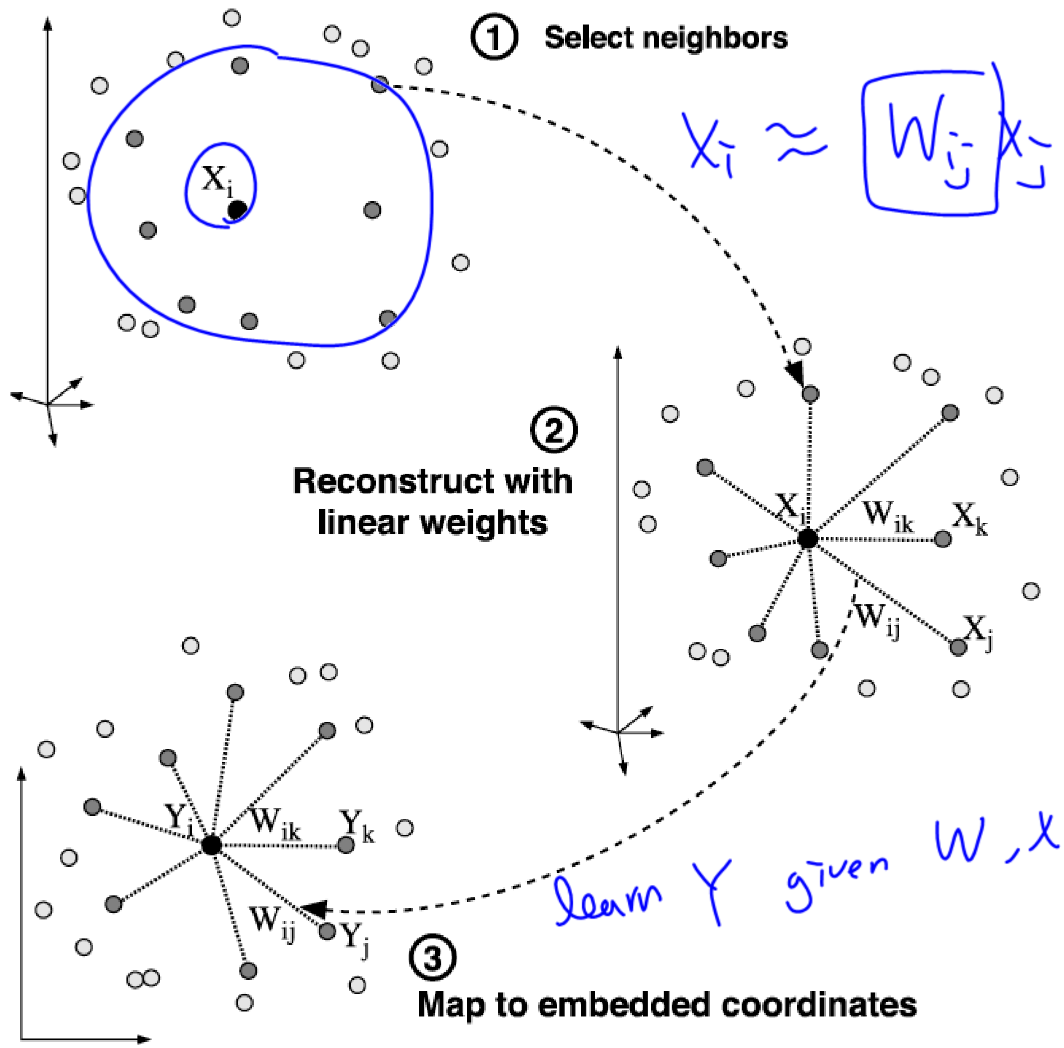
$$\Phi(\mathrm{Y}) = \left\| \mathrm{y}_{\mathrm{i}} - \sum_{i=1}^{K} W_{ij}^{*} y_{ij} \right\|, \text{ where } \mathrm{W}^{*} = \arg\min_{\mathrm{W}} \varphi(\mathrm{W})$$

LLE: Local Linear Embedding

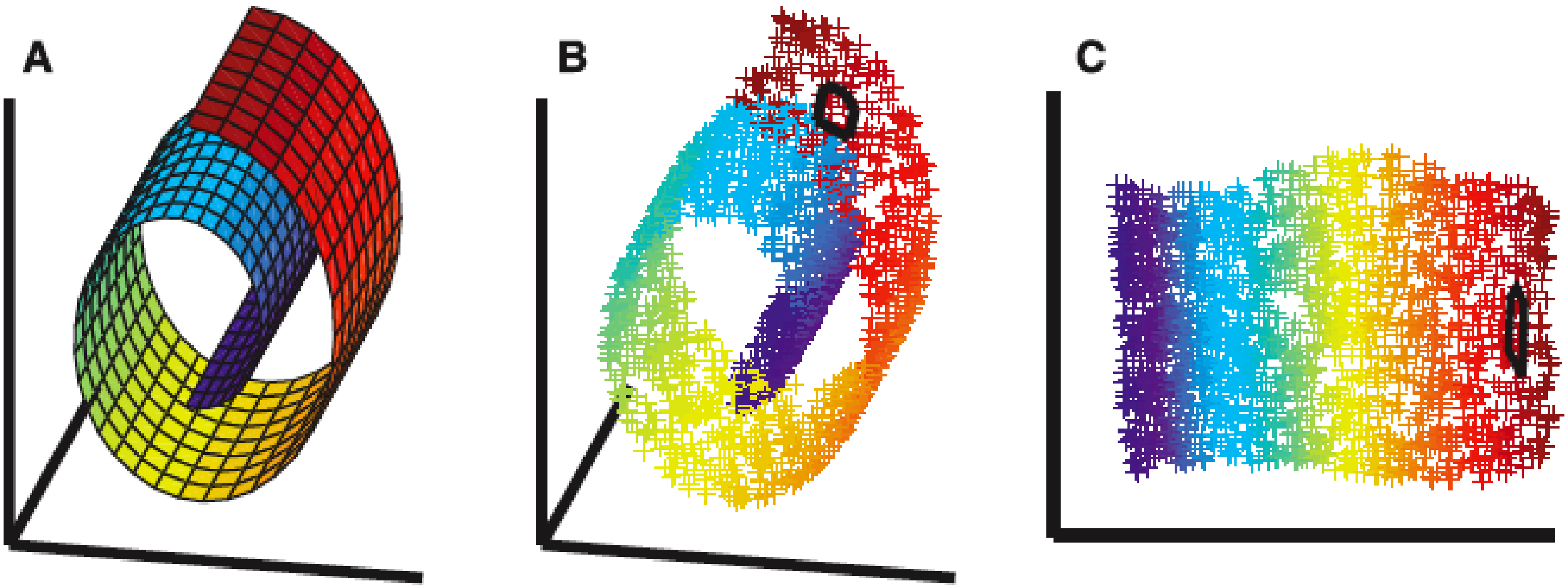
- For each point X_i, define a local neighborhood by *k*-nearest-neighbors.
- Find the weights W_{ij} that best reconstruct X_i from its neighbors X_j.
- Select the dimensionality *d* of the manifold.
- Fix the weights W_{ij} and solve for the points Y_i in the d-dimensional space that are best reconstructed with the same weights.

LLE: Local Linear Embedding

- Uses only local properties of the manifold.
- No computing of geodesics.



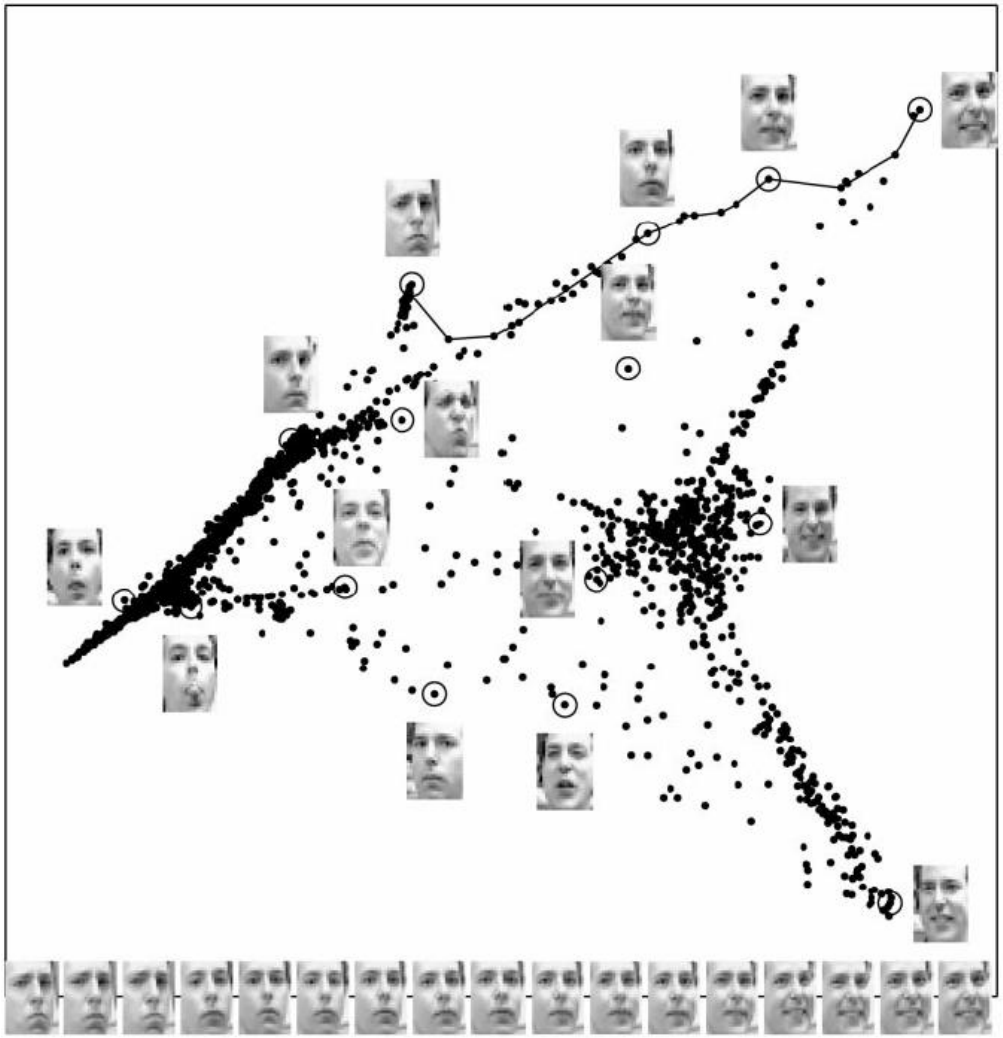
LLE on the Swiss Roll



- Both Isomap and LLE depend on the neighborhood being local to the manifold.

LLE and the manifold of faces

- Separates orientation and facial expression, in spite of image representation in pixel space.



Independent Component Analysis

$$\underbrace{X}_{\text{data}} \approx \overbrace{A}^{\text{mixing matrix}}\underbrace{S}_{\text{source}}$$

- Independent Component Analysis (ICA)
  - Also called: "blind source separation"
- Suppose *N* independent signals are mixed, and sensed by *N* independent sensors.
  - Cocktail party with speakers and microphones.
  - EEG with brain wave sources and sensors.
  - etc.
- Can we reconstruct the original signals, given the mixed data from the sensors?
  - Latent variables from data.

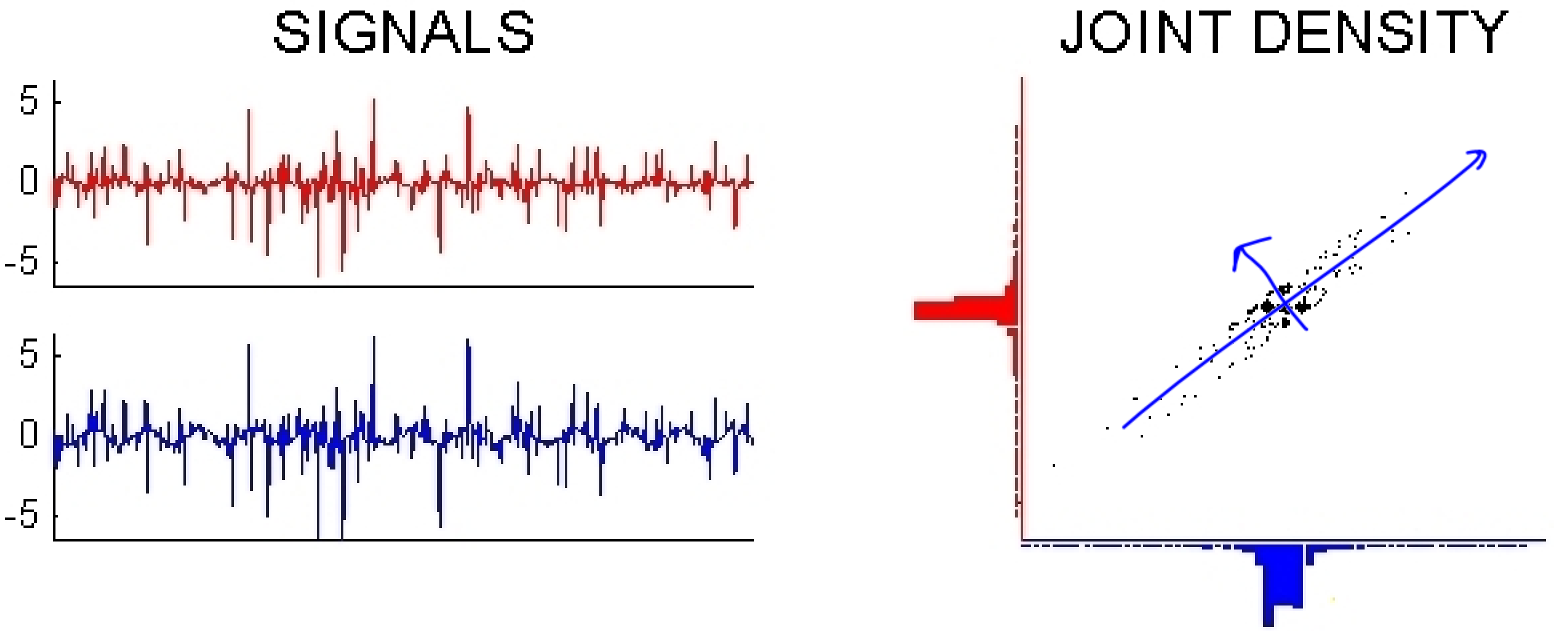
Independent Component Analysis

- The sources s must be independent.
  - And they must be non-Gaussian.
- Linear mixing to get the sensor signals x.
  - $\boxed{\mathbf{x} = \mathbf{A}\mathbf{s}}$ + s are independent. $s = A^{-1}x = Wx$
- We will use an example with — Find W s.t.
  - Two source **s**. Two sensors **x**. — $w_1^T x, w_2^T x, \ldots, w_k^T x$ become independent
  - 2x2 mixing matrix **A**.

$\max_{A} p(x)$

Independent Component Analysis

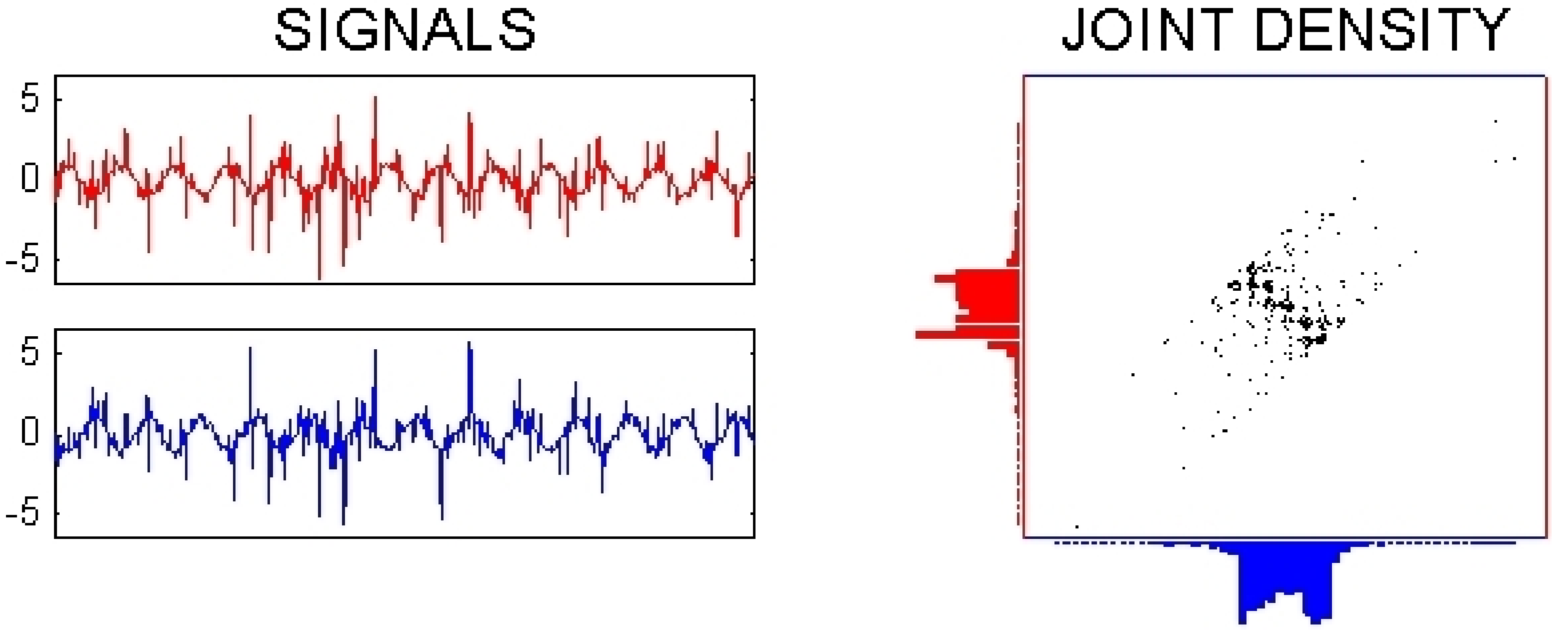
- Mixture example.



Input signals and density

Independent Component Analysis

- Remove correlations by whitening (sphering) the data.



Whitened signals and density

Independent Component Analysis

- To whiten the input data,
  - We want a linear transformation $\mathbf{y} = \mathbf{V}\mathbf{x}$
  - So the components are uncorrelated: $E[\mathbf{y}\mathbf{y}^T] = \mathbf{I}$
  - Given the original covariance $\mathbf{C} = E[\mathbf{x}\mathbf{x}^T]$

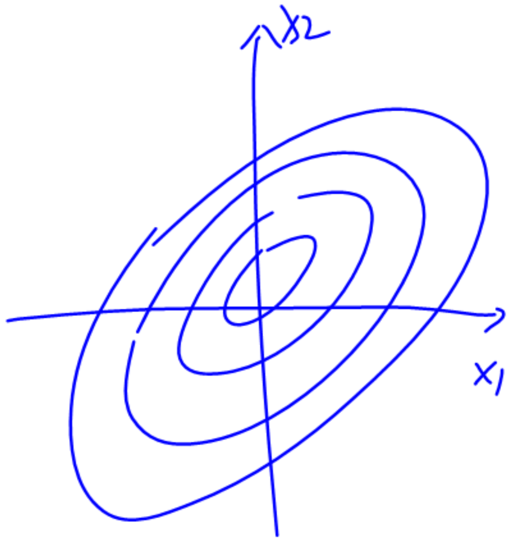
    $= UDU^T = U\sqrt{D}\sqrt{D}^T U^T$
  - We can use $\mathbf{V} = \mathbf{C}^{-1/2}$

    $= (\sqrt{D})^{-1} U^T$
  - Because

$$E[\mathbf{y}\mathbf{y}^T] = E[\mathbf{V}\mathbf{x}\mathbf{x}^T\mathbf{V}^T] = \mathbf{C}^{-1/2}\mathbf{C}\mathbf{C}^{-1/2} = \mathbf{I}$$

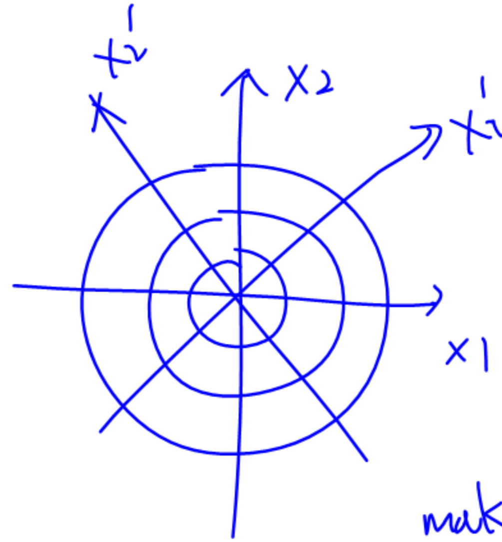
$Vx(Vx)^T$

$= Vxx^T V^T$

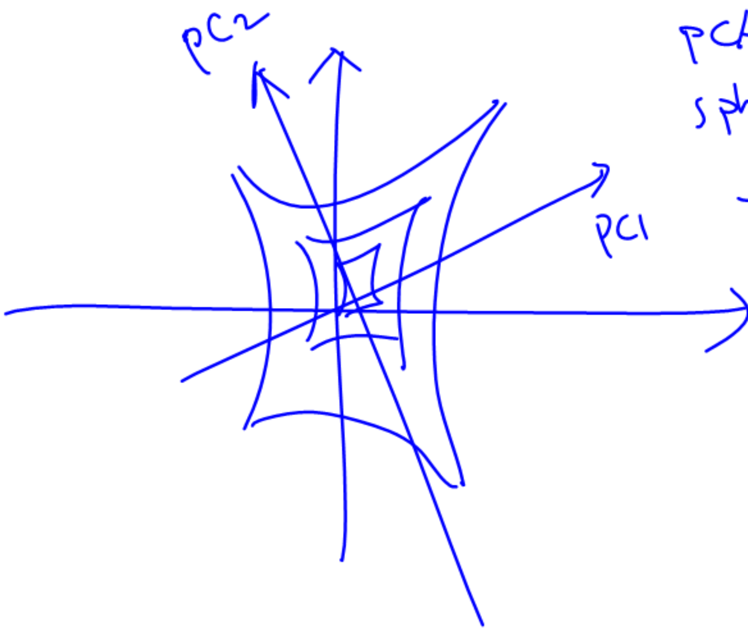


sphering
(PCA whitening)

$y = Vx$

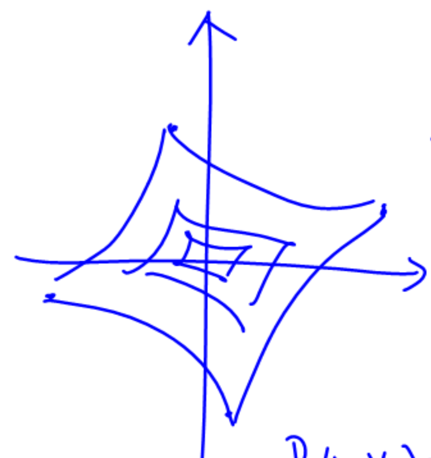


make wx non-gaussian
=
maximize kurtosis of wx

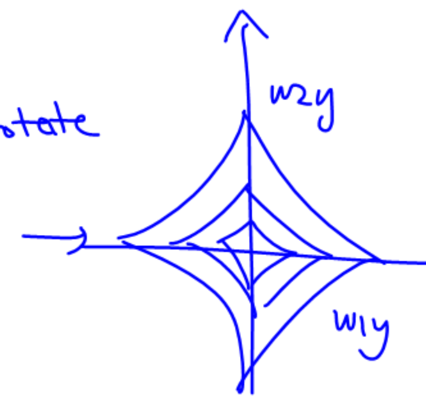


PCA whitening
sphering

$y = Vx$



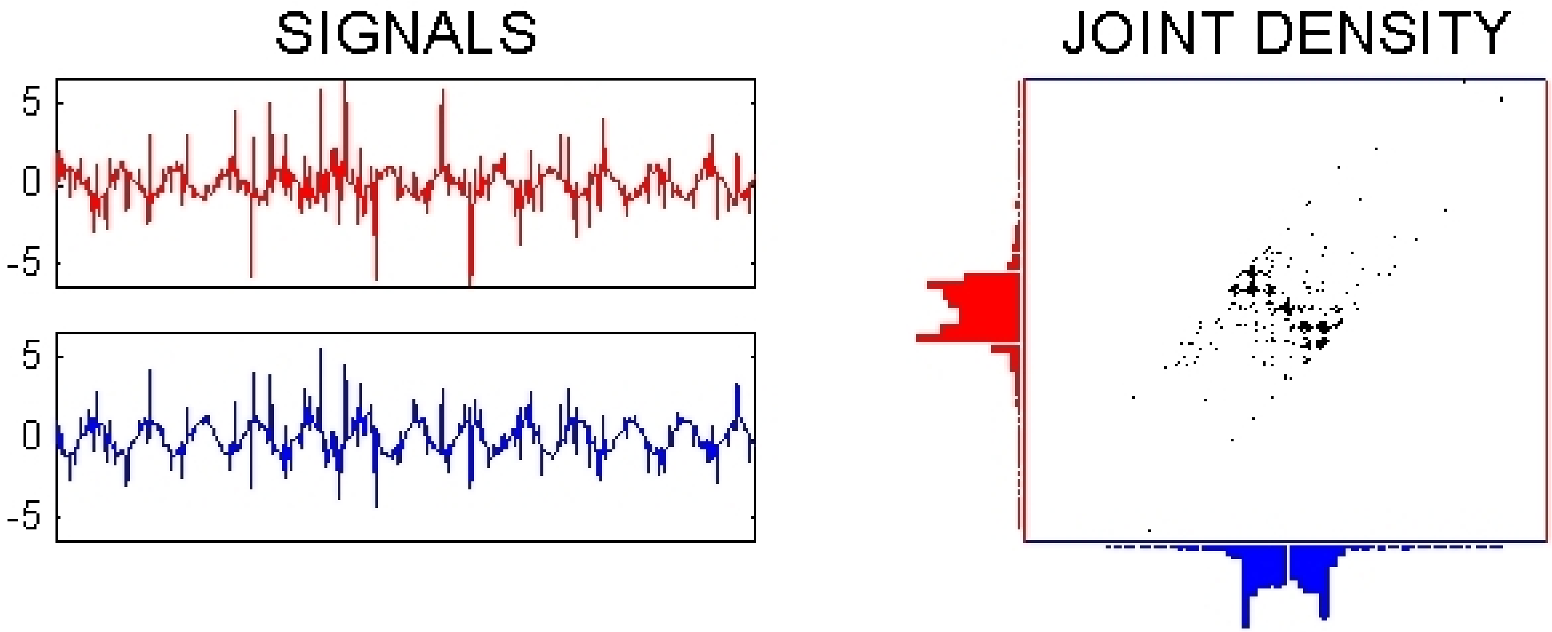
rotate



$$P(x_1, x_2) = \exp(-\lambda|x_1| - \lambda|x_2|)$$
$$= \exp(-\lambda|x_1|)\exp(-\lambda|x_2|)$$
$$\propto P(x_1)\,P(x_2)$$

Independent Component Analysis

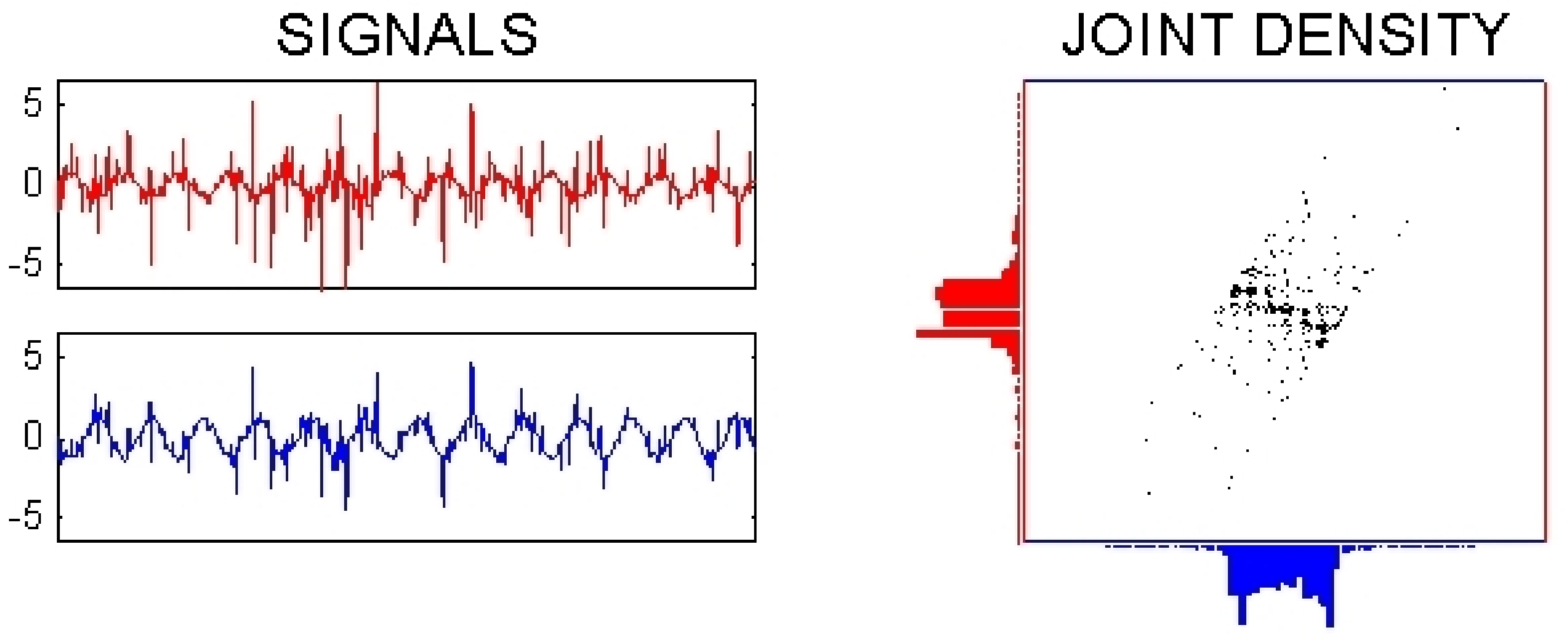
- Step 1 of FastICA



Separated signals after 1 step of FastICA

Independent Component Analysis

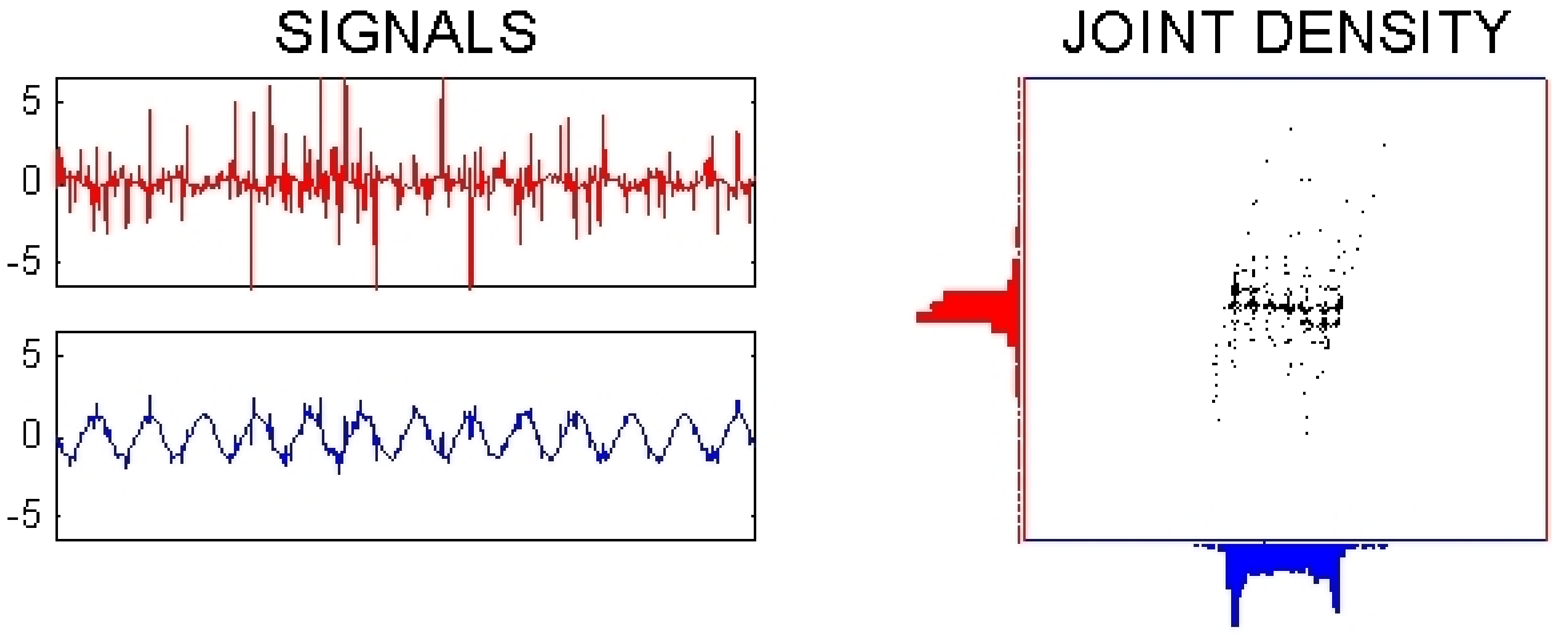
- Step 2 of FastICA



Separated signals after 2 steps of FastICA

Independent Component Analysis

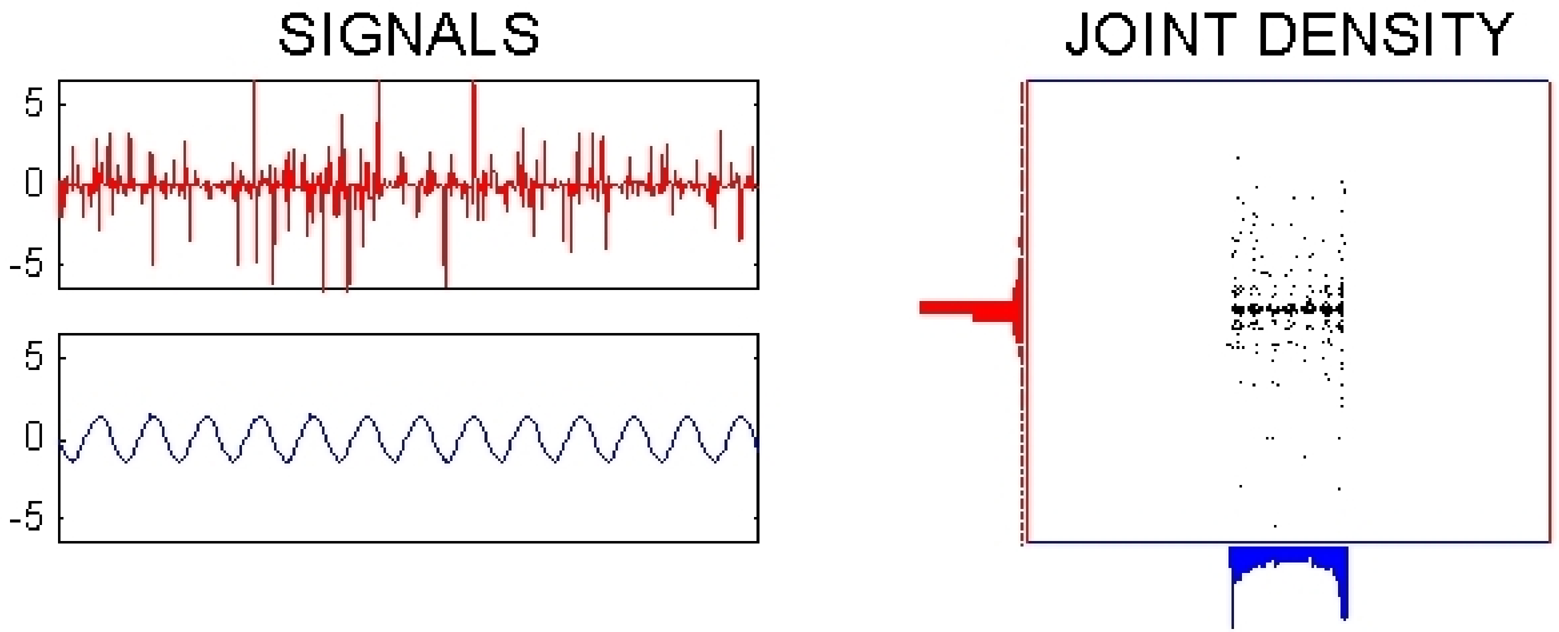
- Step 3 of FastICA



Separated signals after 3 steps of FastICA

Independent Component Analysis

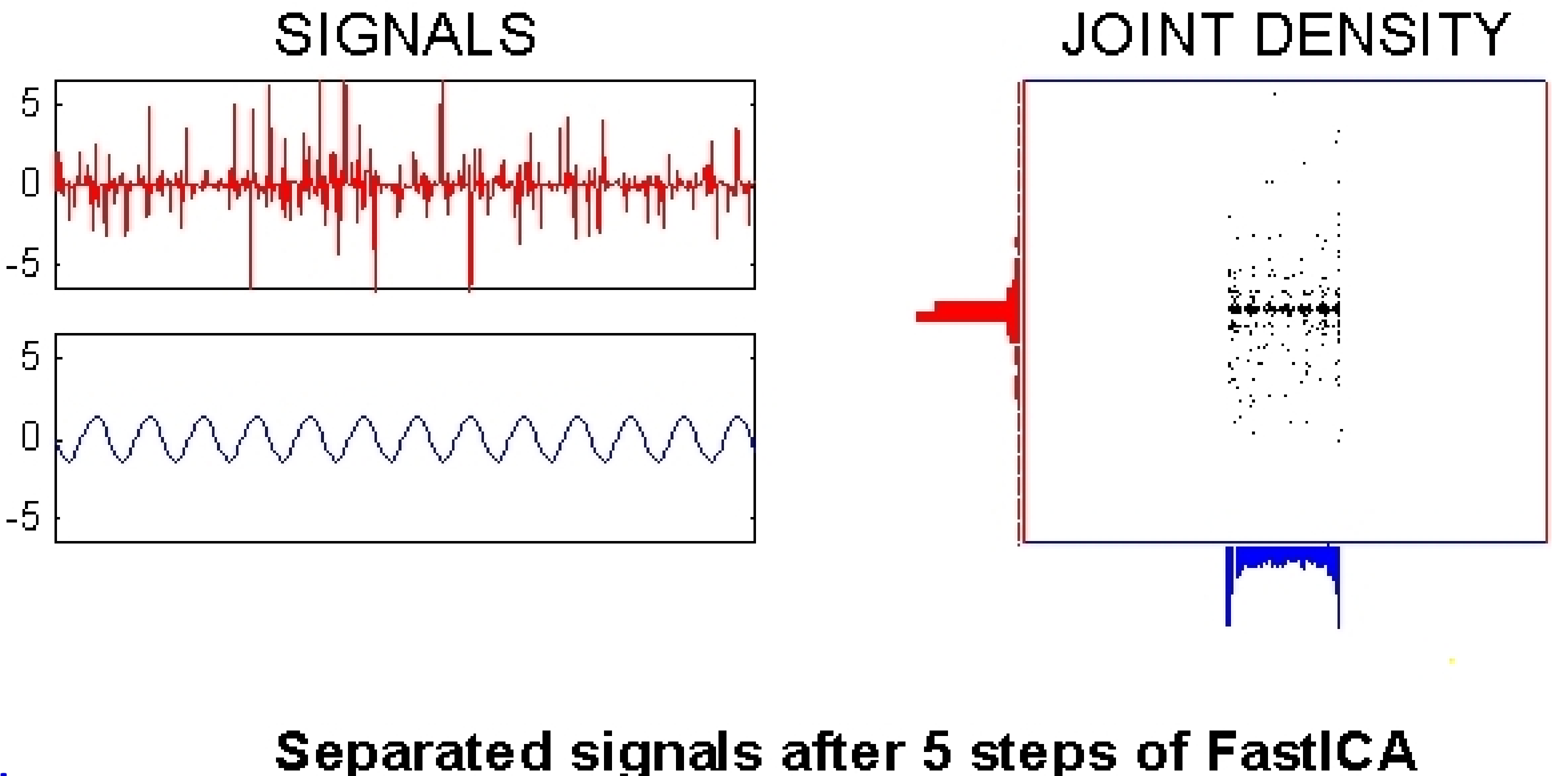
- Step 4 of FastICA



Separated signals after 4 steps of FastICA

Independent Component Analysis

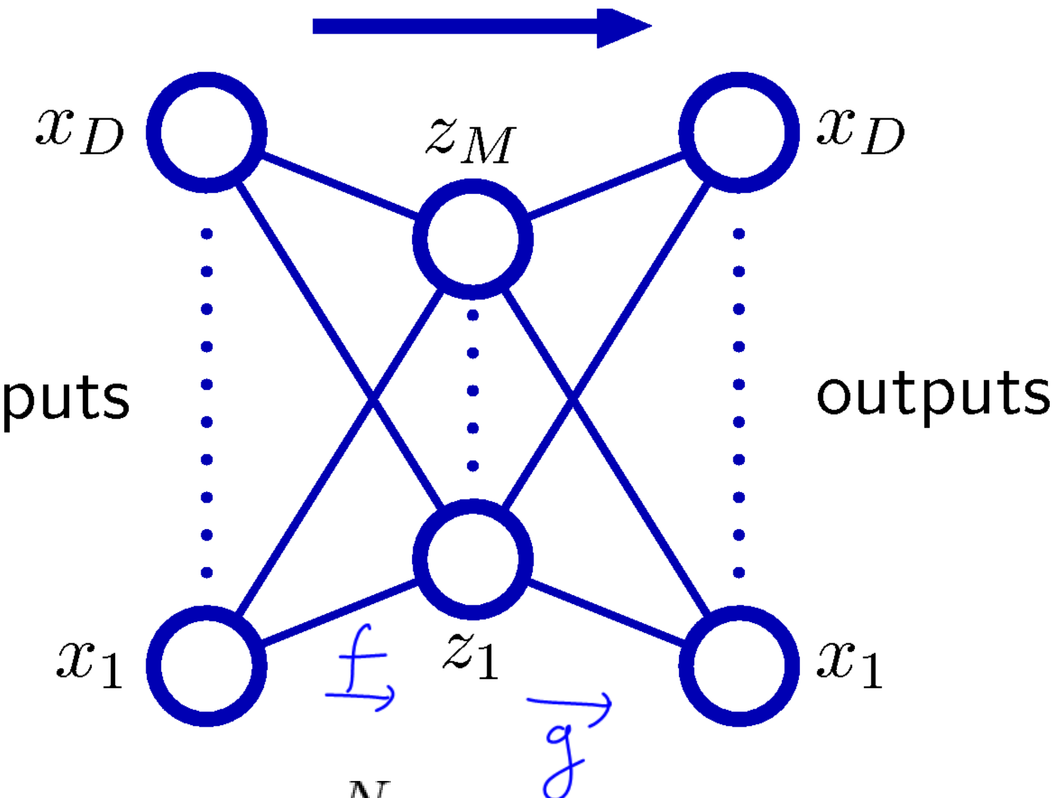
- Step 5: note that $p(y_1,y_2) = p(y_1)\, p(y_2)$



ICA demo
http://cnl.salk.edu/~tewon/Blind/blind_audio.html

Autoassociative neural networks

- A neural network with *D* inputs, *D* outputs, and *M*<<*D* hidden nodes.
- Trained to reproduce the input on the output.



$$E(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N} ||\mathbf{y}(\mathbf{x}_n, \mathbf{w}) - \mathbf{x}_n||^2$$

$\|g(f(x_n)) - x_n\|^2$

if f,g are linear, $g \circ f$ is linear.

$\sum_n \|Wx_n - x_n\|^2$

Autoassociative neural networks

- The hidden layer learns an M-dimensional approximation to the input.
- The trained network performs a projection onto the space spanned by the first M eigenvectors.
  - A form of Principal Component Analysis.
  - (Even with nonlinear (sigmoidal) activation functions, gives the linear PCA subspace.)

Sparse coding

- Sparse coding [Olshausen and Field, 1997]
  - Objective: Given input data {x}, search for a set of bases $\{b_j\}$ such that

$$\vec{x} = \sum_j a_j \vec{b}_j$$

where a_j are mostly zeros.

- Main intuition:
  - Build compact/succinct representations.
  - Learn interpretable and discriminative features.

$$X \approx \sum_j a_j \vec{b}_j$$

PCA
ICA: a are independent
Autoencoder (linear decoder)
$a = f(x)$
Sparse Coding: a are sparse
Kmeans: a is multinomial

Two objectives in sparse coding

- Preserve information
  - Minimize the reconstruction error

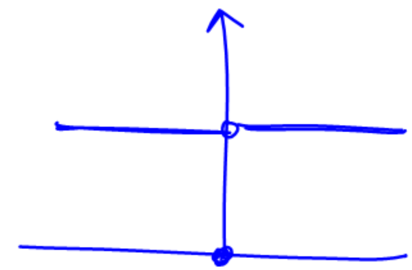
$$||x^{(i)} - \sum_j b_j s_j^{(i)}||^2$$

- Sparseness of coefficients
  - Minimize the sparsity penalty

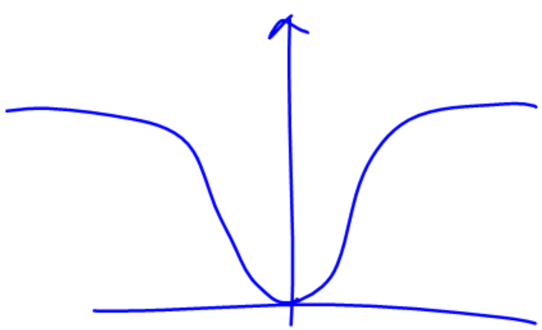
$$\sum_{ij} \Psi(s_j^{(i)})$$

Sparsity penalty

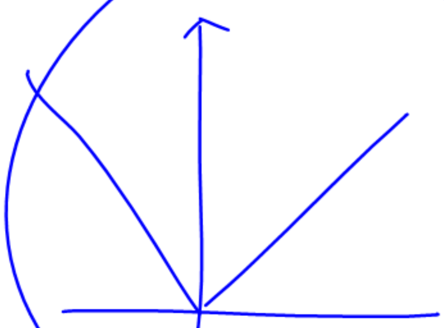
- Many possibilities



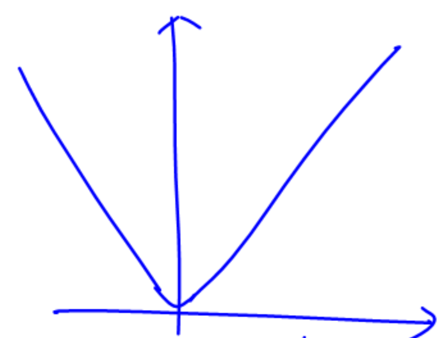
$$\Psi(s) = \begin{cases} I(s \neq 0) & \text{L}_0 \text{ penalty} \\ \log(1+s^2) & \text{log penalty} \\ |s| & \text{L}_1 \text{ penalty} \\ \sqrt{s^2+\epsilon} & \text{epsilon L}_1 \text{ penalty} \end{cases}$$



$\log(1+s^2)$



$|s|$



$\sqrt{s^2+\epsilon}$

Learning bases: optimization

Given input data {x$^{(1)}$, …, x$^{(m)}$}, we want to find good bases {b_1, …, b_n}:

$$\min_{b,a} \sum_i \underbrace{\| x^{(i)} - \sum_j s_j^{(i)} b_j \|_2^2}_{\text{Reconstruction error}} + \beta \underbrace{\sum_i \| s^{(i)} \|_1}_{\text{Sparsity penalty}}$$

$$\forall j: \ \| b_j \| \leq 1$$

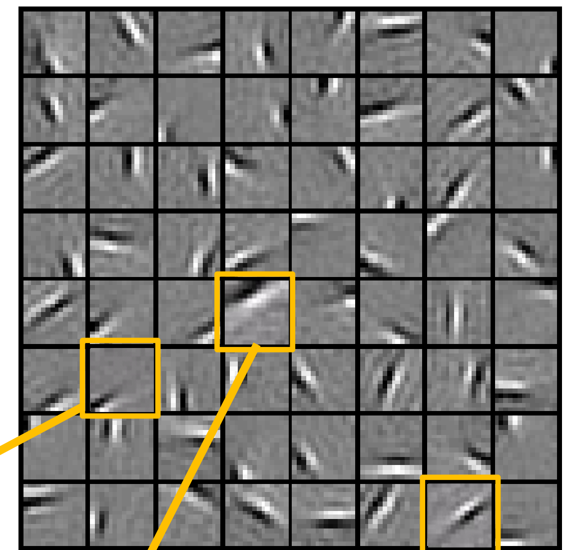
Normalization constraint

Tradeoff between "quality of approximation" and "sparsity" (compactness).

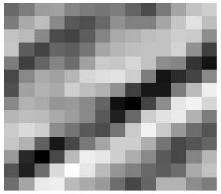
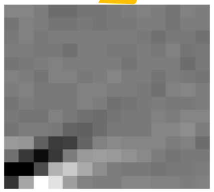
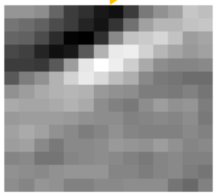
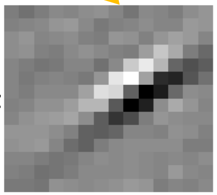
Sparse coding for images

Natural Images

Learned bases: "Edges"



Test example

 $\sim 0.8 *$  $+ 0.3 *$  $+ 0.5 *$ 

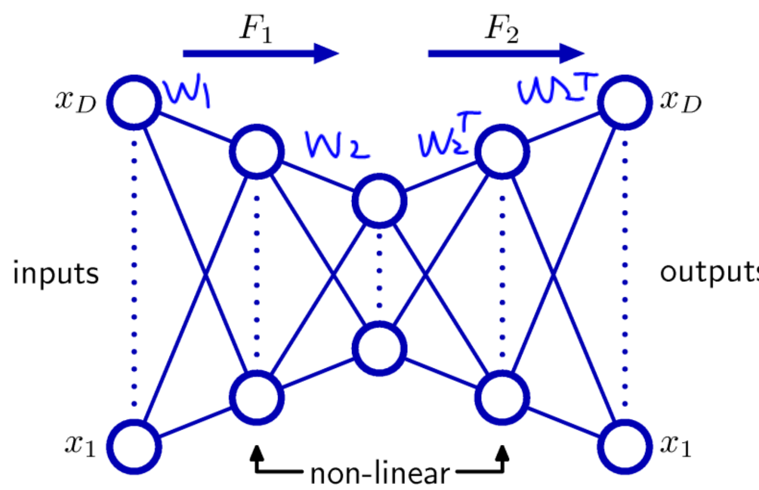
$$x \sim 0.8 * b_{36} + 0.3 * b_{42} + 0.5 * b_{65}$$

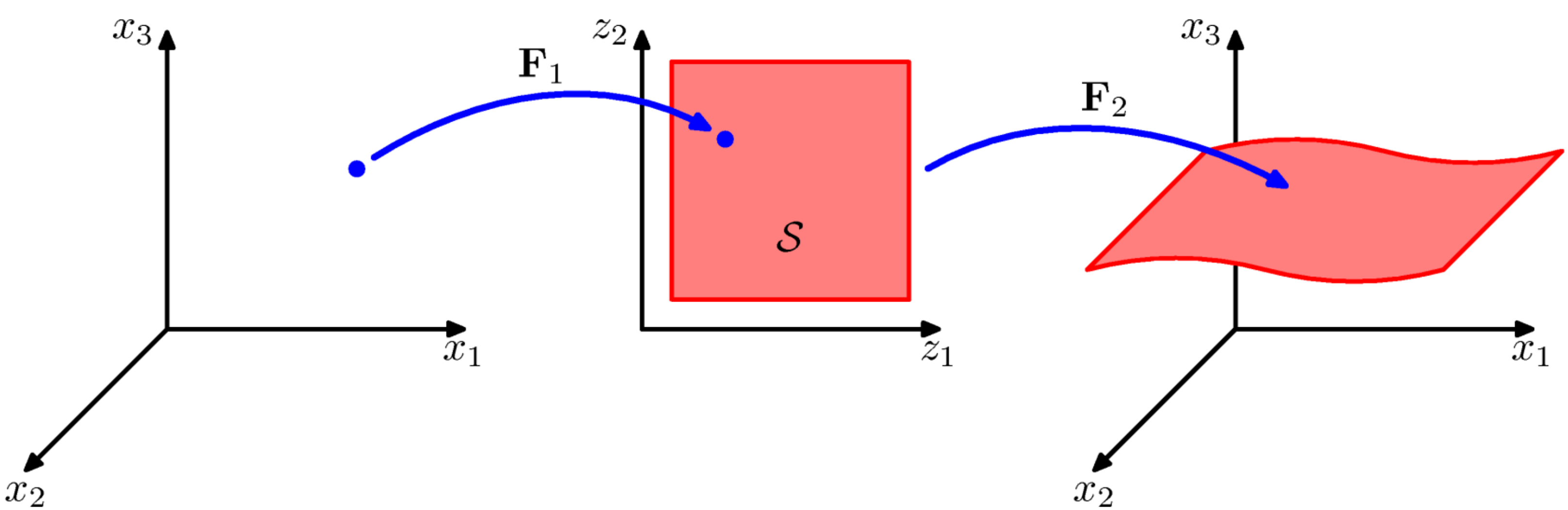
[0, 0, …, 0, **0.8**, 0, …, 0, **0.3**, 0, …, 0, **0.5**, …]
= coefficients (feature representation)

Compact & easily interpretable

Autoassociative neural networks

- Extra hidden layers allows nonlinear PCA.



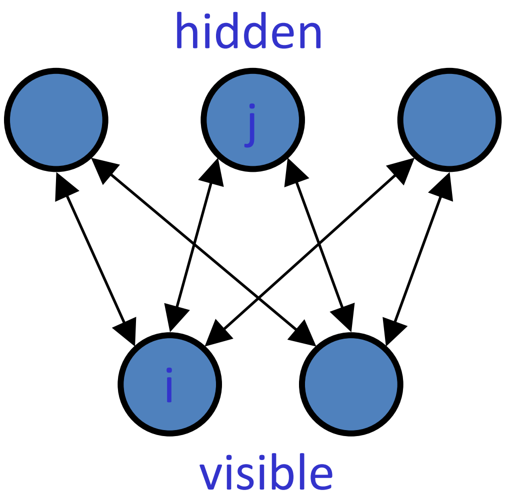


Autoassociative neural networks

- Significance:
  - Simple neural networks can implement PCA, even nonlinear PCA.
  - Thus, unsupervised methods, without prior knowledge of what they are looking for, can identify low-dimensional structure in high-dimensional data.

Restricted Boltzmann Machines

- Representation
  - V: observed (visible) binary variables
  - H: hidden binary variables
  - Bipartite undirected graph (MRF)



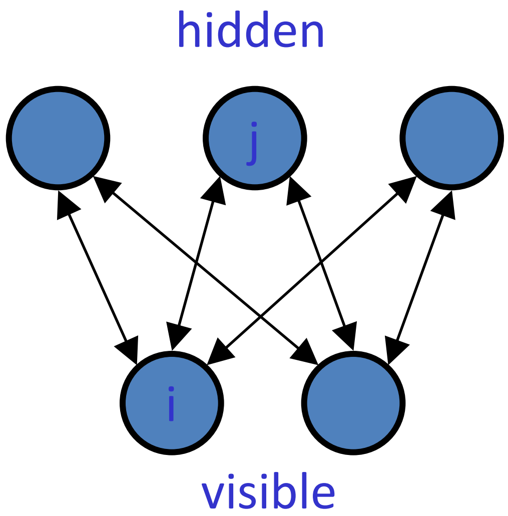
$$P(\mathbf{v},\mathbf{h}) = \frac{1}{Z}\exp(\sum_{i,j}\mathbf{h}_j\mathbf{W}_{ij}\mathbf{v}_i)$$

- Training and inference
  - Trained by Contrastive Divergence (CD) $\max_{\theta} P_{\theta}(V)$
  - Generate samples by Gibbs sampling

More about RBMs

- Inference is easy
  - Given $\mathbf{h}$, all the $\mathbf{v}_i$ are conditionally independent
  - Given $\mathbf{v}$, all the $\mathbf{h}_j$ are conditionally independent

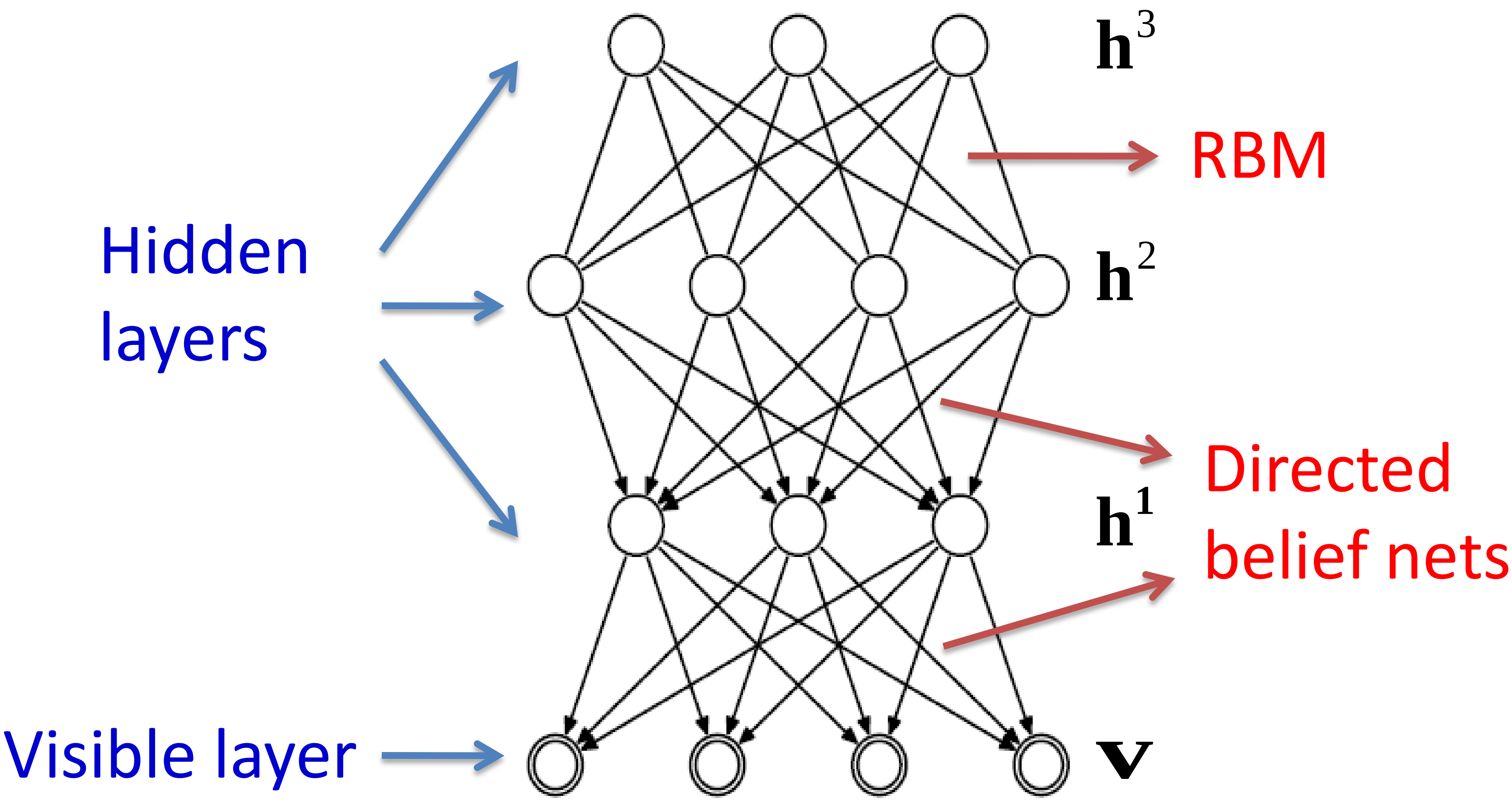
- After training the RBM, the posterior P(h|v) can be used as nonlinear feature mapping of v



Deep Belief Networks(DBNs)

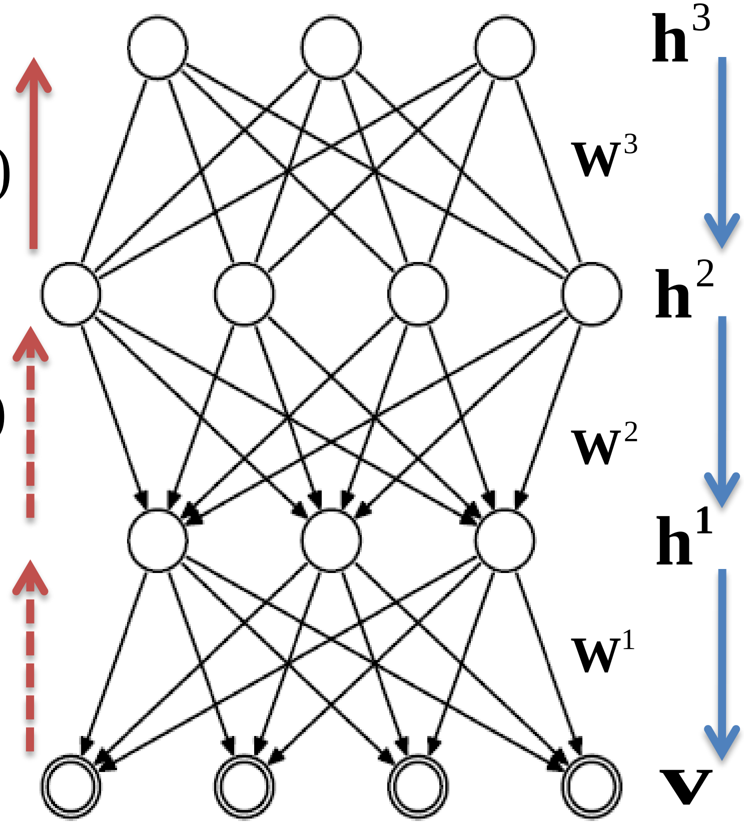
- Probabilistic generative model
- Deep architecture – multiple layers
- Unsupervised learning
  (Maximizing the log-likelihood of the data)
- Can be used as unsupervised pre-training

DBN structure



$$P(\mathbf{v},\mathbf{h}^1,\mathbf{h}^2,\ldots,\mathbf{h}^l) = P(\mathbf{v}\,|\,\mathbf{h}^1)P(\mathbf{h}^1\,|\,\mathbf{h}^2)\ldots P(\mathbf{h}^{l-2}\,|\,\mathbf{h}^{l-1})P(\mathbf{h}^{l-1},\mathbf{h}^l)$$

DBN structure



$Q(\mathbf{h}^3 \mid \mathbf{h}^2) = P(\mathbf{h}^3 \mid \mathbf{h}^2)$

$P(\mathbf{h}^2 \mid \mathbf{h}^3)$

$Q(\mathbf{h}^2 \mid \mathbf{h}^1)$

$P(\mathbf{h}^1 \mid \mathbf{h}^2)$

$Q(\mathbf{h}^1 \mid \mathbf{v})$

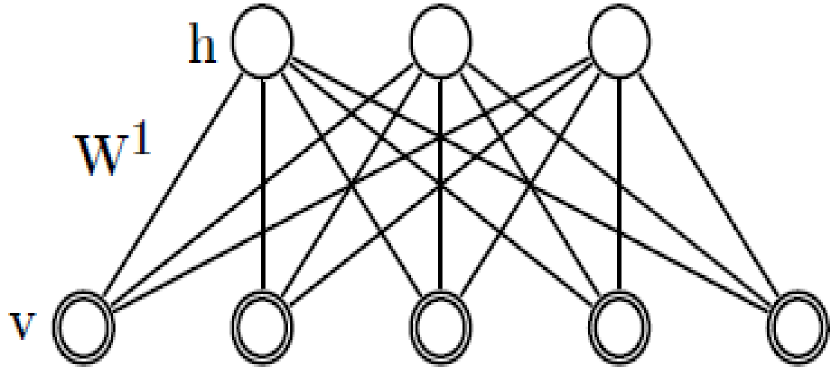
$P(\mathbf{v} \mid \mathbf{h}^1)$

$$P(\mathbf{v}, \mathbf{h}^1, \mathbf{h}^2, \ldots, \mathbf{h}^l) = P(\mathbf{v} \mid \mathbf{h}^1) P(\mathbf{h}^1 \mid \mathbf{h}^2) \ldots P(\mathbf{h}^{l-2} \mid \mathbf{h}^{l-1}) P(\mathbf{h}^{l-1}, \mathbf{h}^l)$$

$$Q(\mathbf{h}^i \mid \mathbf{h}^{i-1}) = \prod_j sigm(\mathbf{b}_j^{i-1} + \mathrm{W}_j^i \mathbf{h}^{i-1}) \qquad P(\mathbf{h}^{i-1} \mid \mathbf{h}^i) = \prod_j sigm(\mathbf{b}_j^i + \mathrm{W}_{.j}^i{}' \mathbf{h}^i)$$

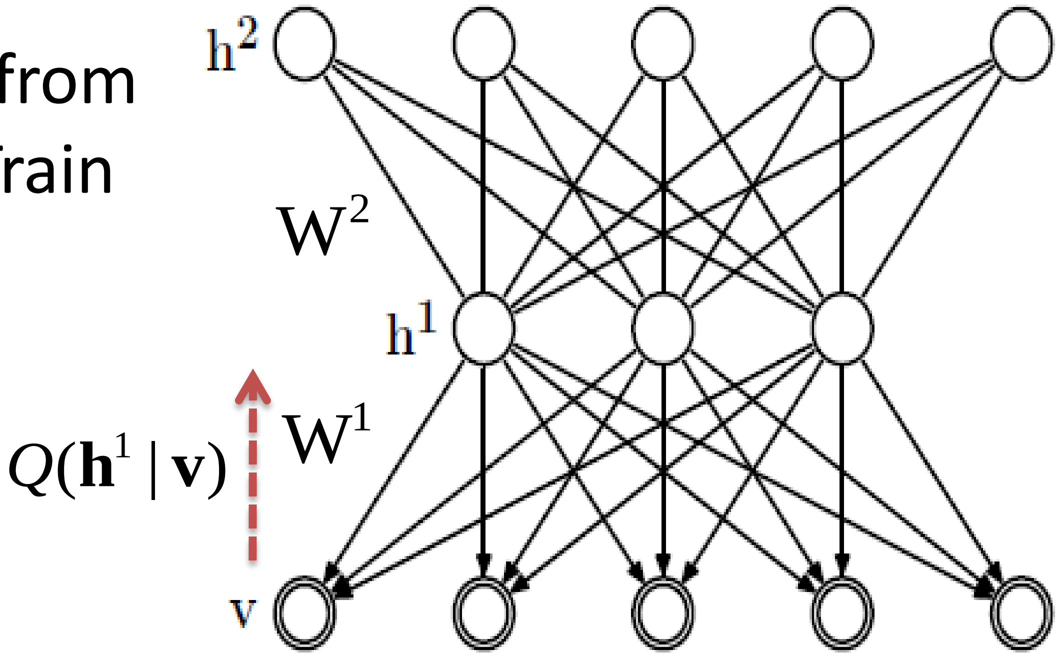
DBN Greedy training

- First step:
  - Construct an RBM with an input layer **v** and a hidden layer **h**
  - Train the RBM (using CD)



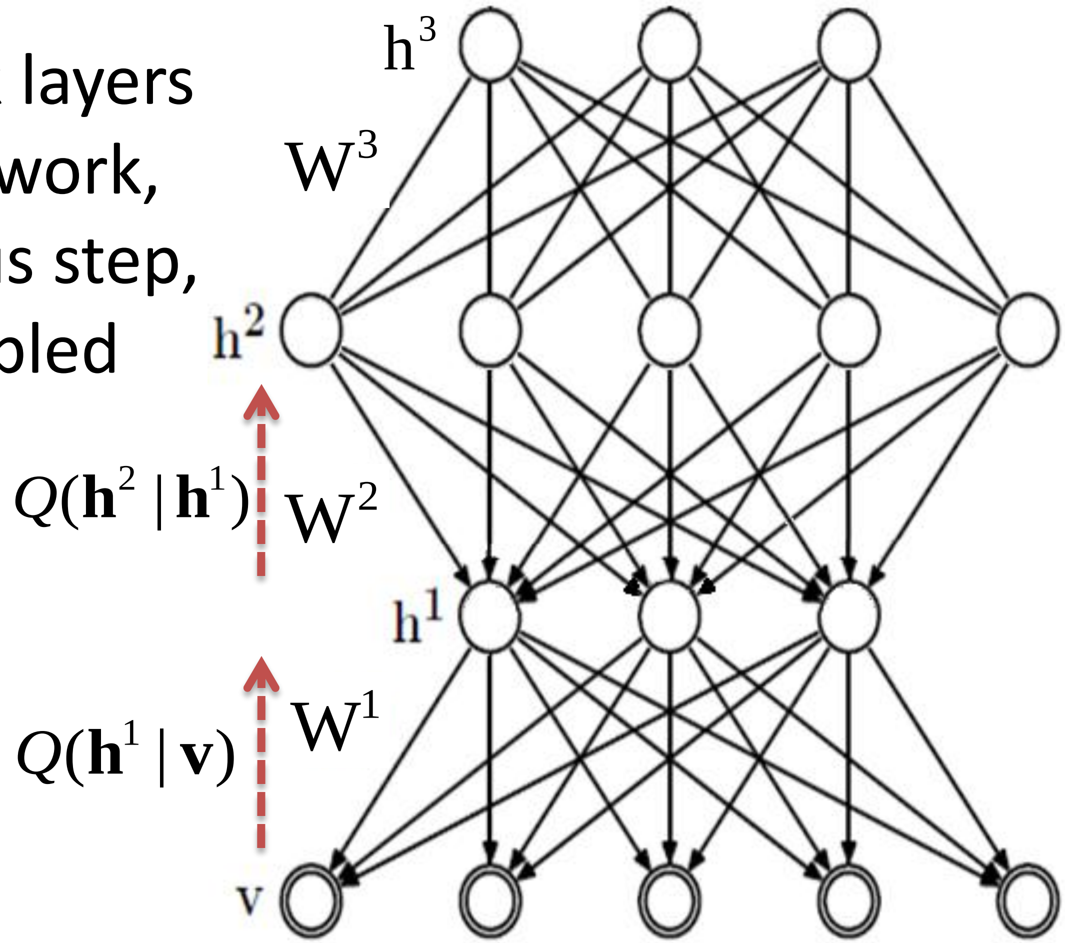
DBN Greedy training

- Second step:
  - Stack another hidden layer on top of the RBM to form a new RBM
  - Fix W^1, sample $\mathbf{h}^1$ from $Q(\mathbf{h}^1 | \mathbf{v})$ as input. Train W^2 as RBM.



DBN Greedy training

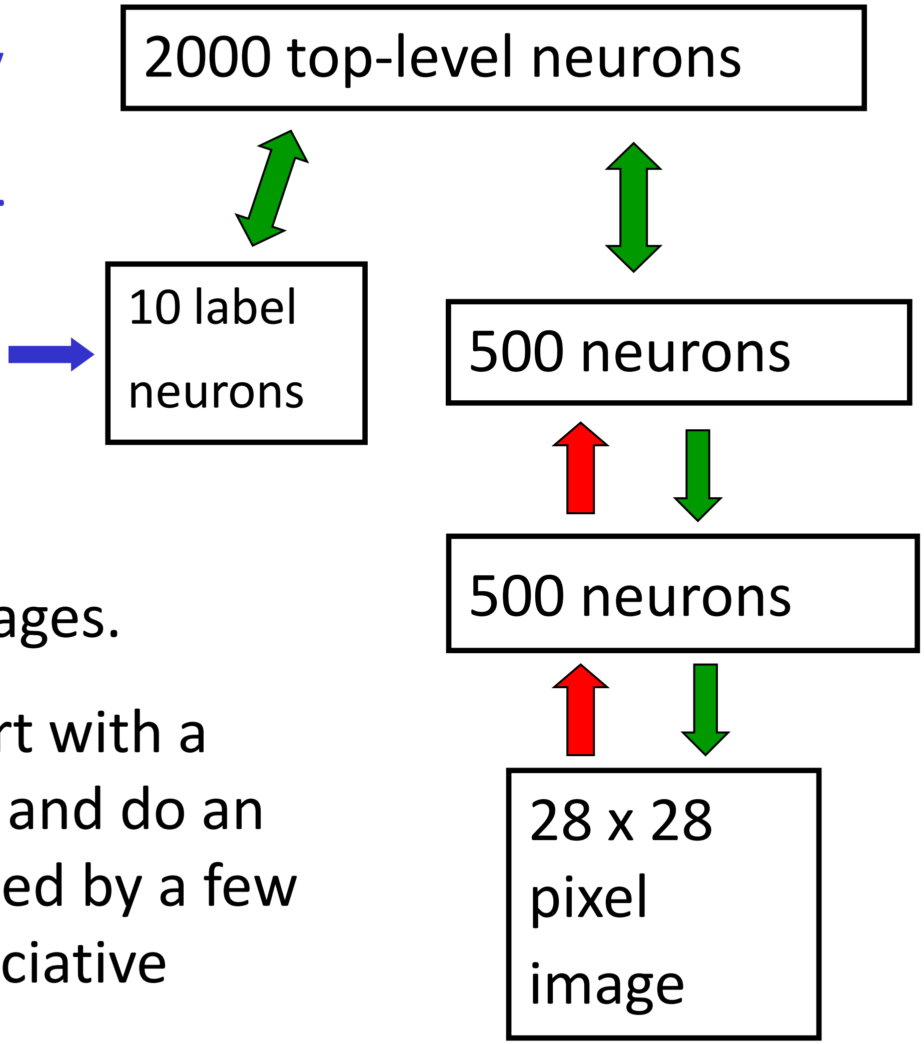
- Third step:
  - Continue to stack layers on top of the network, train it as previous step, with sample sampled from $Q(\mathbf{h}^2 \mid \mathbf{h}^1)$
- And so on...



A model of digit recognition

The top two layers form an associative memory whose energy landscape models the low dimensional manifolds of the digits.

The energy valleys have names



The model learns to generate combinations of labels and images.

To perform recognition we start with a neutral state of the label units and do an up-pass from the image followed by a few iterations of the top-level associative memory.

http://www.cs.toronto.edu/~hinton/nipstutorial/nipstut3.ppt

- More details on up-down algorithm:
  - Hinton, G. E., Osindero, S. and Teh, Y. (2006) "A fast learning algorithm for deep belief nets", Neural Computation, 18, pp 1527-1554. http://www.cs.toronto.edu/~hinton/absps/ncfast.pdf

- Handwritten digit demo:
  - http://www.cs.toronto.edu/~hinton/digits.html

Next

- Next, from Bishop:
  - Hidden Markov Models, Dynamical Systems
- Then, Reinforcement Learning
  - From the Sutton & Barto book