Beyond projections

CSE 250B

Beyond projections

PCA and SVD find informative linear projections. Given a data set in $\mathbb{R}^p$, and a number $k < p$, they:

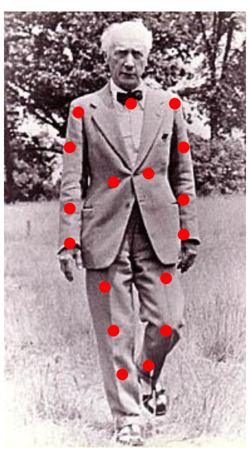
- Find orthogonal directions $u_1, \ldots, u_k \in \mathbb{R}^p$
- Approximate points in $\mathbb{R}^p$ by their projection into the subspace spanned by these directions

Two ways in which we'd like to generalize this.

1. Manifold learning
   What if the data lies on (or near) a nonlinear surface?
2. Dictionary learning
   What if we want the basis vectors $u_1, \ldots, u_k$ to have other special properties: for instance, that the data points x have a *sparse representation* in terms of these directions?

Low dimensional manifolds

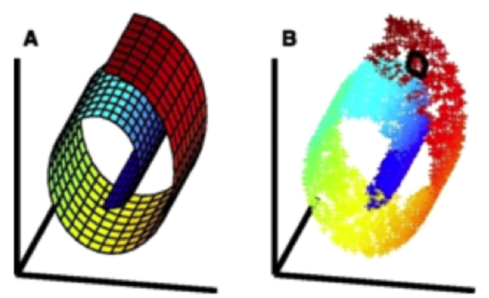
Sometimes data in a high-dimensional space $\mathbb{R}^p$ in fact lies close to a k-dimensional manifold, for $k \ll p$



1. Motion capture
   M markers on a human body yields data in $\mathbb{R}^{3M}$
2. Speech signals
   Representation can be made arbitrarily high dimensional by applying more filters to each window of the time series

This whole area: "Manifold learning"

The ISOMAP algorithm

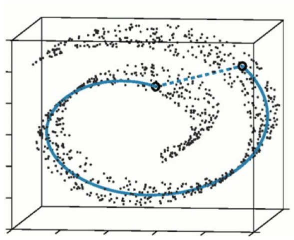


ISOMAP (Tenenbaum et al, 1999): given data $x^{(1)}, \ldots, x^{(n)} \in \mathbb{R}^p$,

1. Estimate *geodesic distances* between the data points: that is, distances along the manifold.
2. Then embed these points into Euclidean space so as to (approximately) match these distances.

How can these two steps be achieved?

Estimating geodesic distances



Key idea: for **nearby** pairs of points, Euclidean distance and geodesic distance are approximately the same.

1. Construct the **neighborhood graph.**
   Given data $x^{(1)}, \ldots, x^{(n)}$, construct a graph $G = (V, E)$ with
   - Nodes $V = \{1, 2, \ldots, n\}$ (one per data point)
   - Edges $(i, j) \in E$ whenever $x^{(i)}$ and $x^{(j)}$ are close together
2. Compute distances in this graph.
   Set the length of any $(i, j) \in E$ to $\|x^{(i)} - x^{(j)}\|$. Compute all pairwise distances between nodes, using a shortest-paths algorithm.

Distance-preserving embeddings

The algorithmic task:

- *Input:* An $n \times n$ matrix of pairwise distances

$$D_{ij} = \text{desired distance between points } i \text{ and } j,$$

  as well as an integer k.
- *Output:* an embedding $z^{(1)}, \ldots, z^{(n)} \in \mathbb{R}^k$ that realizes these distances as closely as possible.

Most widely-used algorithm: **classical multidimensional scaling.**

- Let $D \in \mathbb{R}^{n \times n}$ be the matrix of desired *squared* interpoint distances.
- Schoenberg (1938): D can be realized in Euclidean space if and only if $B = -\frac{1}{2}HDH$ is positive semidefinite, where $H = I_n - \frac{1}{n}11^T$.
- In fact, looking at this matrix B suggests an embedding even if it is not positive semidefinite.

The Gram matrix

For points in Euclidean space, it is easy to express squared distances in terms of dot products.

$$\|x - x'\|^2 \;=\; \|x\|^2 + \|x'\|^2 - 2x \cdot x' \;=\; x \cdot x + x' \cdot x' - 2x \cdot x'.$$

What about expressing dot products in terms of distances?

Let $z^{(1)}, \ldots, z^{(n)}$ be points in Euclidean space.

- Let $D_{ij} = \|z^{(i)} - z^{(j)}\|^2$ be the squared interpoint distances.
- Let $B_{ij} = z^{(i)} \cdot z^{(j)}$ be the dot products. This is the **Gram matrix**.

Moving between D and B:

- We've seen: $D_{ij} = B_{ii} + B_{jj} - B_{ij} - B_{ji}$. That is, D is linear in B.
- A little algebra also shows that for $H = I_n - \frac{1}{n}11^T$,

$$B = -\frac{1}{2}HDH.$$

The Gram matrix is convenient: we can read off an embedding from it.

Quick quiz

Consider the two points $x_1 = (1, 0)$ and $x_2 = (-1, 0)$ in $\mathbb{R}^2$.

1. What is the matrix D of squared interpoint distances?
2. Write down $H = I_n - \frac{1}{n}11^T$.
3. Compute $B = -\frac{1}{2}HDH$. Is this the correct Gram matrix?

In general, how might one recover an embedding from the Gram matrix?

Recovering an embedding based only on distances

Let $D \in \mathbb{R}^{n \times n}$ be a matrix of desired *squared* interpoint distances. Suppose these are realizable in Euclidean space: that is, there exist vectors $z^{(1)}, \ldots, z^{(n)}$ such that $D_{ij} = \|z^{(i)} - z^{(j)}\|^2$.

We have seen that we can easily obtain the Gram matrix, $B_{ij} = z^{(i)} \cdot z^{(j)}$.

- B is p.s.d. (why?). Thus its eigenvalues are nonnegative.
- Compute spectral decomposition:

$$B = U \Lambda U^T = YY^T,$$

  where Λ is the diagonal matrix of eigenvalues and $Y = U\Lambda^{1/2}$.
- Denote the rows of Y by $y^{(1)}, \ldots, y^{(n)}$. Then

$$y^{(i)} \cdot y^{(j)} = (YY^T)_{ij} = B_{ij} = z^{(i)} \cdot z^{(j)}.$$

  If dot products are preserved, so are distances.

Result: an embedding $y^{(1)}, \ldots, y^{(n)}$ that exactly replicates distances D.

What is the dimensionality of this embedding?

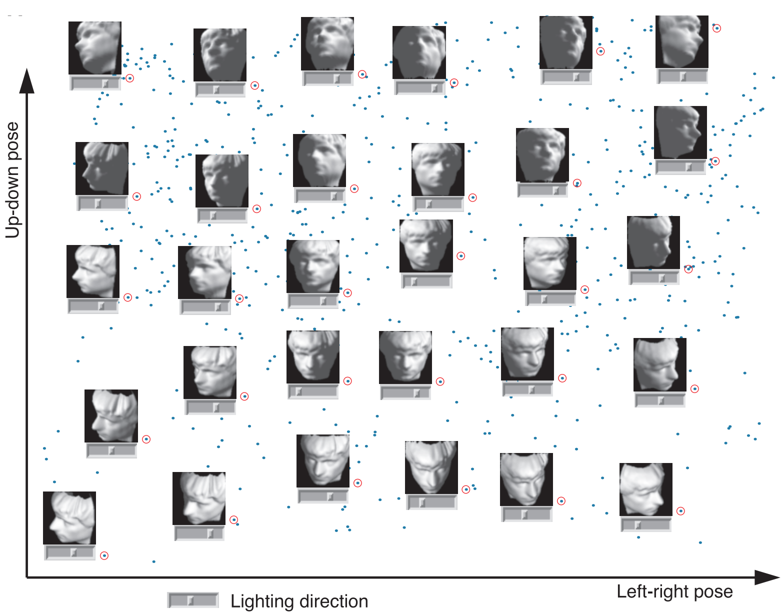
Classical multidimensional scaling

A slight generalization works even when the distances cannot necessarily be realized in Euclidean space.

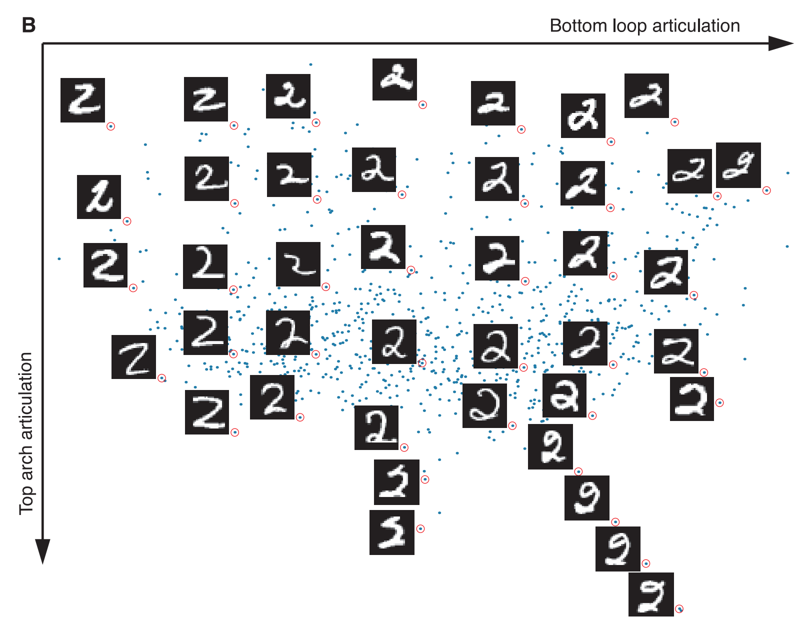
Given $n \times n$ matrix D of squared interpoint distances, and target dimension k:

1. Compute $B = -\frac{1}{2}HDH$.
2. Compute the spectral decomposition $B = U\Lambda U^T$, where the eigenvalues in Λ are arranged in decreasing order.
3. Zero out any negative entries of Λ to get Λ_+.
4. Set $Y = U\Lambda_+^{1/2}$.
5. Set Y_k to the first k columns of Y.
6. Let the embedding of the n points be given by the rows of Y_k.

ISOMAP: examples



ISOMAP: examples



More manifold learning

1. Other good algorithms, such as
   - Locally linear embedding
   - Laplacian eigenmaps
   - Maximum variance unfolding
2. Notions of intrinsic dimensionality
3. Statistical rates of convergence for data lying on manifolds
4. Capturing other kinds of topological structure

Dictionary learning

Given data points $x^{(1)}, \ldots, x^{(n)} \in \mathbb{R}^p$, and an integer m:

- Choose m dictionary vectors $\phi_1, \ldots, \phi_m \in \mathbb{R}^p$.
- Approximate each $x^{(i)}$ by a linear combination of these dictionary elements.

$$\underbrace{\begin{pmatrix} \uparrow & & \uparrow \\ x^{(1)} & \cdots & x^{(n)} \\ \downarrow & & \downarrow \end{pmatrix}}_{\text{data matrix } X} \approx \underbrace{\begin{pmatrix} \uparrow & \uparrow & & \uparrow \\ \phi_1 & \phi_2 & \cdots & \phi_m \\ \downarrow & \downarrow & & \downarrow \end{pmatrix}}_{\text{dictionary } \Phi} \underbrace{\begin{pmatrix} \uparrow & & \uparrow \\ s^{(1)} & \cdots & s^{(n)} \\ \downarrow & & \downarrow \end{pmatrix}}_{\text{encoding } S}$$

- Principal component analysis: the ϕ_i are orthogonal and $m \leq p$
- Independent component analysis: the rows of S are approximately statistically independent and $m \leq p$
- Sparse coding: the columns of S are sparse and often $m > p$ ("overcomplete basis")

Sparse coding

Given $x^{(1)}, \ldots, x^{(n)} \in \mathbb{R}^p$, find dictionary vectors $\phi_1, \ldots, \phi_m$ and sparse representations $s^{(1)}, \ldots, s^{(n)} \in \mathbb{R}^m$ such that

$$x^{(i)} \approx \Phi s^{(i)}.$$

Optimization problem: find matrices Φ, S that minimize

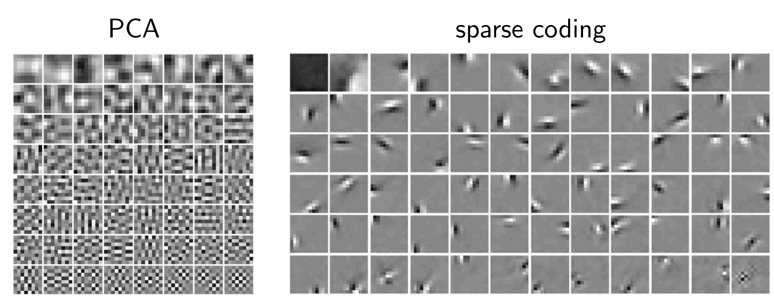
$$\begin{aligned} L(\Phi, S) &= \|X - \Phi S\|_F^2 - \lambda \cdot \text{sparsity}(S) \\ &= \sum_{i=1}^n \left(\|x^{(i)} - \Phi s^{(i)}\|^2 - \lambda \cdot \text{sparsity}(s^{(i)}) \right) \end{aligned}$$

Alternating minimization procedure:

- Initialize Φ somehow
- Repeat until convergence:
  - Fixing Φ, minimize $L(\cdot)$ over S
  - Fixing S, minimize $L(\cdot)$ over Φ

Example: image patches

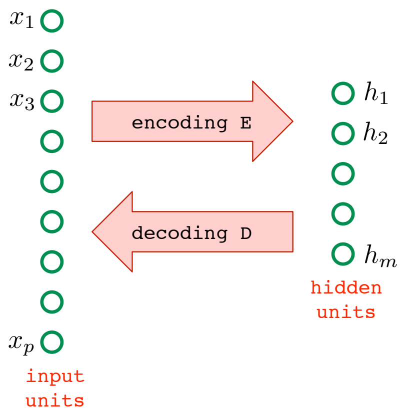
Olshausen-Field (1996), Lewicki-Olshausen (1999): PCA versus sparse coding for natural image patches.



Sparse coding does a much better job at finding a basis that resembles the receptive fields of simple cells in visual cortex.

Autoencoders

A generalization of dictionary learning.

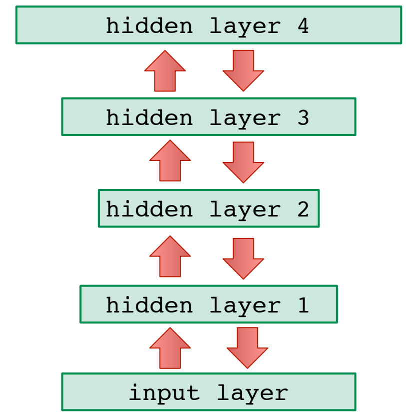


Here E and D might be probabilistic maps. Fit them so that

$$x \approx D(E(x)) \;\text{ on data points } x \in \mathbb{R}^p.$$

Stacked autoencoders

Successively higher-level representations



One way to fit these models (using unlabeled data):

- Fit one layer at a time to the previous layer's activations
- Then fine-tune the whole structure to minimize reconstruction error

Example: MNIST

Hinton-Salakhutdinov (2006):

- First row: original images
- Second row: reconstruction using stacked autoencoder with layers of size 784-1000-500-250-30
- Third and fourth rows: reconstruction using two variants of PCA, each with 30 components