Advanced course in machine learning
582744
Lecture 6

Arto Klami

Outline

Mixtures of linear models

All of the FA models have marginal density that is a multivariate normal distribution with some low-rank covariance matrix

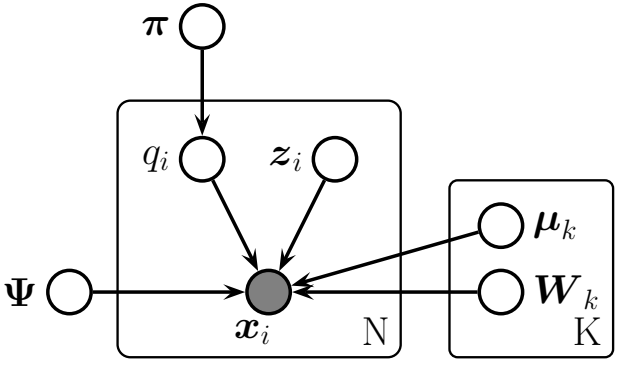
Hence, we can use them as component densities in a mixture model

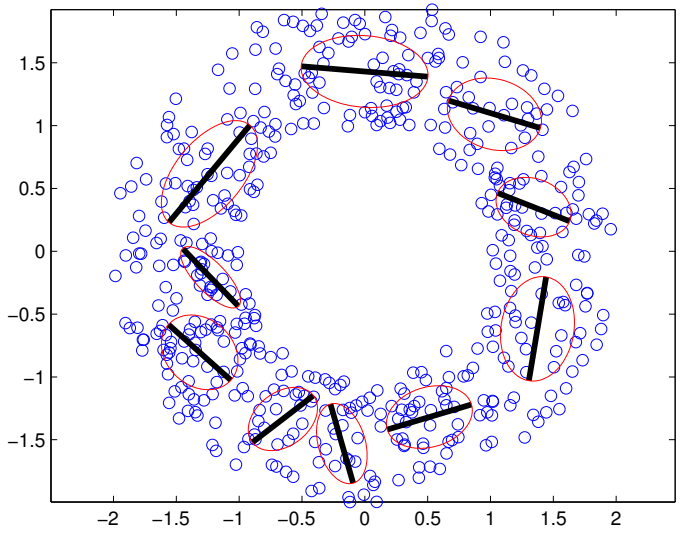
Can either be interpreted as constrained mixture of Gaussians, or as piecewise-linear dimensionality reduction (though we do not get global coordinates)

All mixtures of Gaussians are actually special cases of mixture of factor analyzers; with zero factors we get spherical/diagonal covariance, etc.

...and mixtures with one cluster naturally give PCA/FA/CCA

Mixtures of linear models





Section 12.1.5. presents the EM algorithm for this general case

Recommender systems: Collaborative filtering

Given triplets (user, item, rating), predict the rating for new (user, item) -pairs

$$\begin{bmatrix} 5 & 4 & & & & \\ & 1 & & & 4 & 5 \\ & & 3 & & 5 & 1 \\ 4 & 5 & & 1 & & \end{bmatrix}$$

Recommender systems: Collaborative filtering

Given triplets (user, item, rating), predict the rating for new (user, item) -pairs

$$\begin{bmatrix} 5 & 4 & & & & \\ & 1 & & & 4 & 5 \\ & & 3 & & 5 & 1 \\ 4 & 5 & & 1 & & \end{bmatrix} \approx \begin{bmatrix} u_{11} & u_{12} \\ u_{21} & u_{22} \\ u_{31} & u_{32} \\ u_{41} & u_{42} \end{bmatrix} \begin{bmatrix} v_{11} & v_{21} & v_{31} & v_{41} & v_{51} & v_{61} \\ v_{12} & v_{22} & v_{32} & v_{42} & v_{52} & v_{62} \end{bmatrix}$$

Recommender systems

Prototypical application for matrix factorization with missing data $\mathbf{X} \approx \mathbf{U}\mathbf{V}^T$: Learn factorization based on the observed entries, predict the missing ones

The approximation presents all users and items in a lower dimensional space, such that similar representations correspond to high rating

The Netflix challenge: 100M ratings for 480k users and 18k movies (99% missing)

Probabilistic matrix factorization

$$\mathbf{X}_{ij} = p_X(\sum_k \mathbf{U}_{ik}\mathbf{V}_{jk})$$
$$\mathbf{U} = p_U(\theta_U)$$
$$\mathbf{V} = p_V(\theta_V)$$

Probabilistic PCA/FA is one example, using normal distributions as the densities

For recommender systems we usually need to replace the likelihood with Bernoulli (for binary ratings) or, for example, Poisson distribution ("how many times one listened this song")

The learning algorithm for extremely sparse binary matrices is too complex for this course, but is pretty much state-of-the-art for recommeder systems

Recommender systems: Collaborative filtering

Alternative solutions based on user- or item-similarity:

- Find other users with similar preferences, recommend items they liked
- Find items similar to the ones the user liked (Amazon et al.)

Algorithmically: Nearest neighbor is often quite good, just need to choose the distance measure

Cold-start problem: Recommending anything for new users or products is extremely difficult, since no (or little) data exists

Content-based recommendation instead of collaborative filtering helps for that case

Non-negative matrix factorization

If the data is positive, then we might want to model it with positive factors as well

Easier to interpret and encourages sparsity

Probabilistic formulation for example using Poisson likelihood and Gamma priors for the factors (Section 27.2.4)

Non-negative matrix factorization (Section 13.8.1)

Non-probabilistic formulation by minimizing $\|\mathbf{X} - \mathbf{WH}\|^2$ subject to the non-negativity constraint

Assuming we start with positive $\mathbf{W}$ and $\mathbf{H}$, we can use gradient updates with small steps:

$$\mathbf{H}_{t+1} = \mathbf{H}_t + \eta_H[(\mathbf{W}^T\mathbf{X}) - (\mathbf{W}^T\mathbf{W}\mathbf{H}_t)]$$
$$\mathbf{W}_{t+1} = \mathbf{W}_t + \eta_W[(\mathbf{X}\mathbf{H}^T) - (\mathbf{W}_t\mathbf{H}\mathbf{H}^T)]$$

Non-negative matrix factorization (Section 13.8.1)

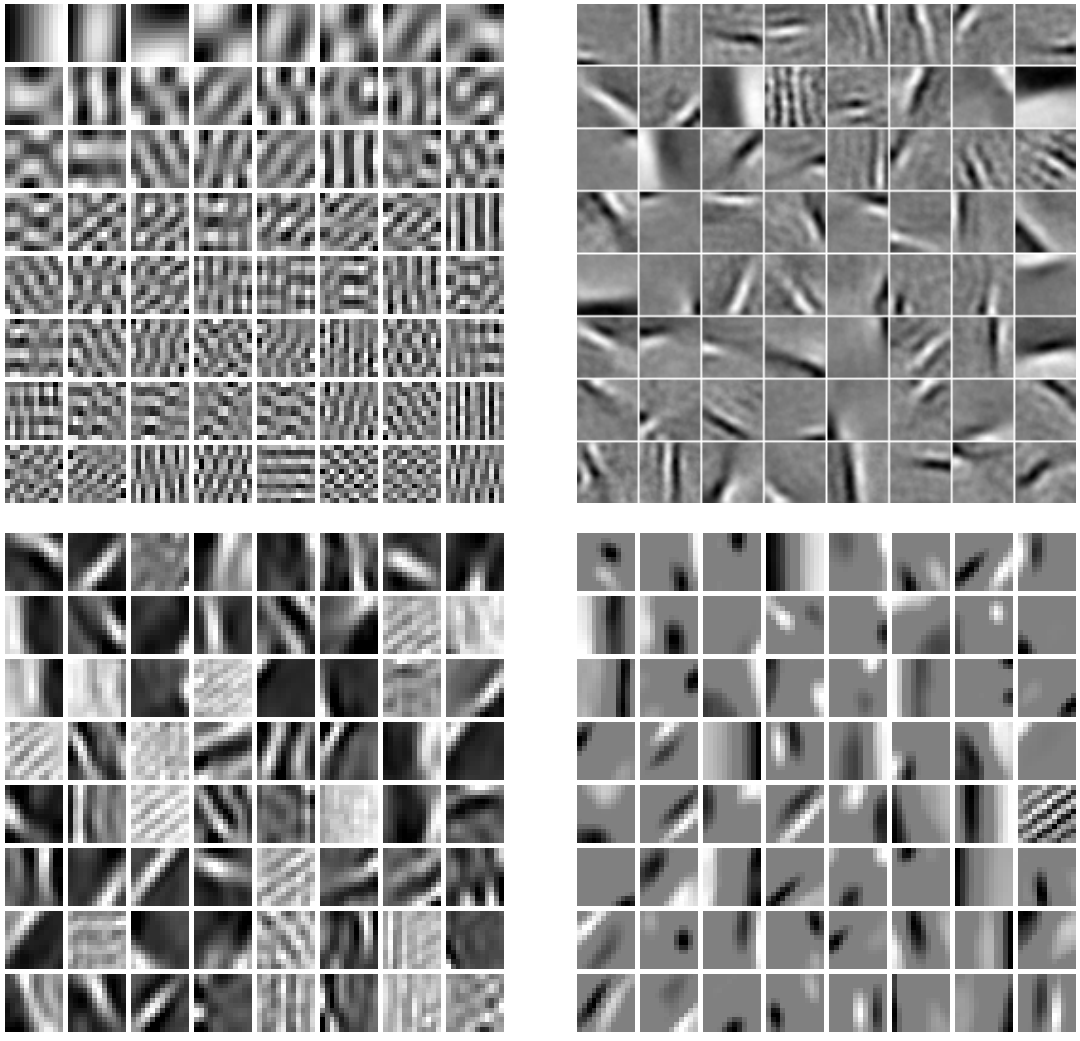
Remember how we can consider other descent directions as well? Or that scaling the gradient separately for each dimension is fine (Adagrad)?

Plug in $\eta_H = \frac{\mathbf{H}_t}{(\mathbf{W}_t^T \mathbf{W}_t \mathbf{H}_t)}$ as the learning rate for the gradient $\mathbf{H}_{t+1} = \mathbf{H}_t + \eta_H[(\mathbf{W}^T\mathbf{X}) - (\mathbf{W}^T\mathbf{W}\mathbf{H}_t)]$ and simplify to get $\mathbf{H}_{t+1} = \mathbf{H}_t \times \frac{(\mathbf{W}_t^T\mathbf{X})}{(\mathbf{W}_t^T\mathbf{W}_t\mathbf{H}_t)}$

Similarly use $\eta_W = \frac{\mathbf{W}_t}{\mathbf{W}_t\mathbf{H}_{t+1}\mathbf{H}_{t+1}^T}$ to get $\mathbf{W}_{t+1} = \mathbf{W}_t \times \frac{\mathbf{X}\mathbf{H}_{t+1}^T}{\mathbf{W}_t\mathbf{H}_{t+1}\mathbf{H}_{t+1}^T}$

The resulting updates are multiplicative, even though we started from gradients

Sparse dictionaries



Top row: PCA and ICA

Bottom row: NMF and sparse PCA

Non-linear dimensionality reduction

Linear models are efficient and often easy to interpret, but linearity is a strong assumption

Plenty of techniques for non-linear dimensionality reduction as well

We will go through a few alternatives not discussed in the book

Laplacian/spectral eigenmaps

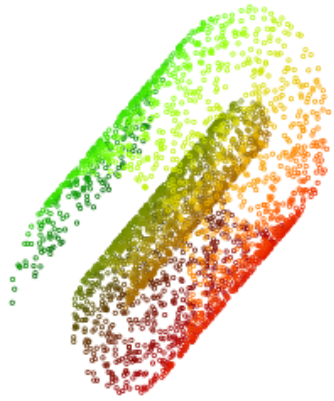
We already know one method for this: Compute the graph laplacian of some graph connecting the observations, and represent the data with the first M eigenvectors

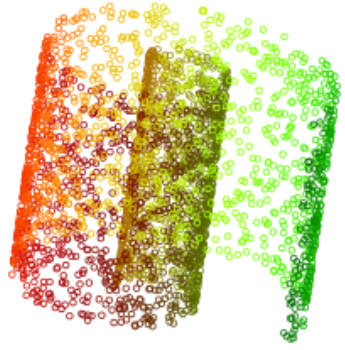
Then use, for example, PCA to drop into even smaller dimensionality

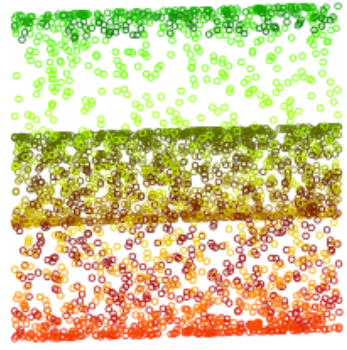
Emphasizes cluster structure due to the properties of the graph Laplacian

Manifold structure in data

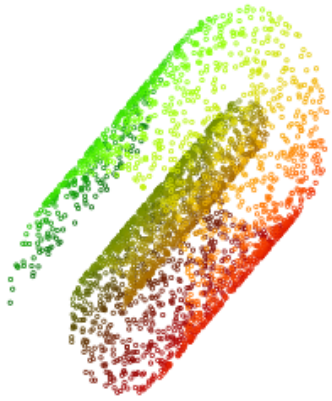
Manifold structure in data

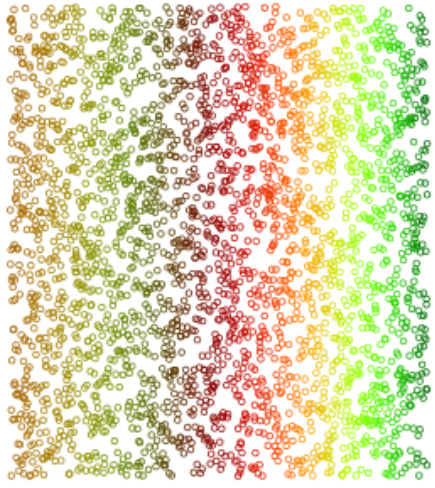


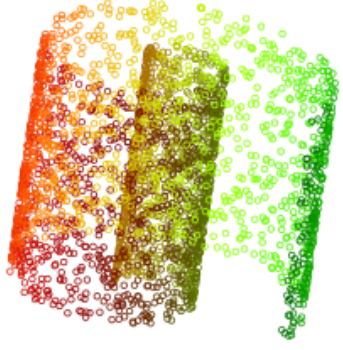


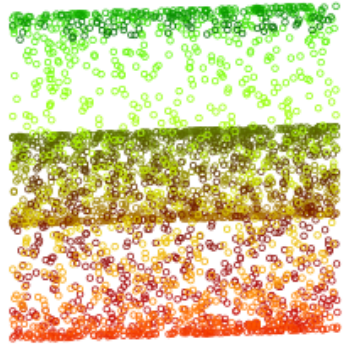


Manifold structure in data









Multidimensional scaling (MDS)

MDS is a general framework for non-linear dimensionality reduction techniques, starting with pairwise distances d_{ij}

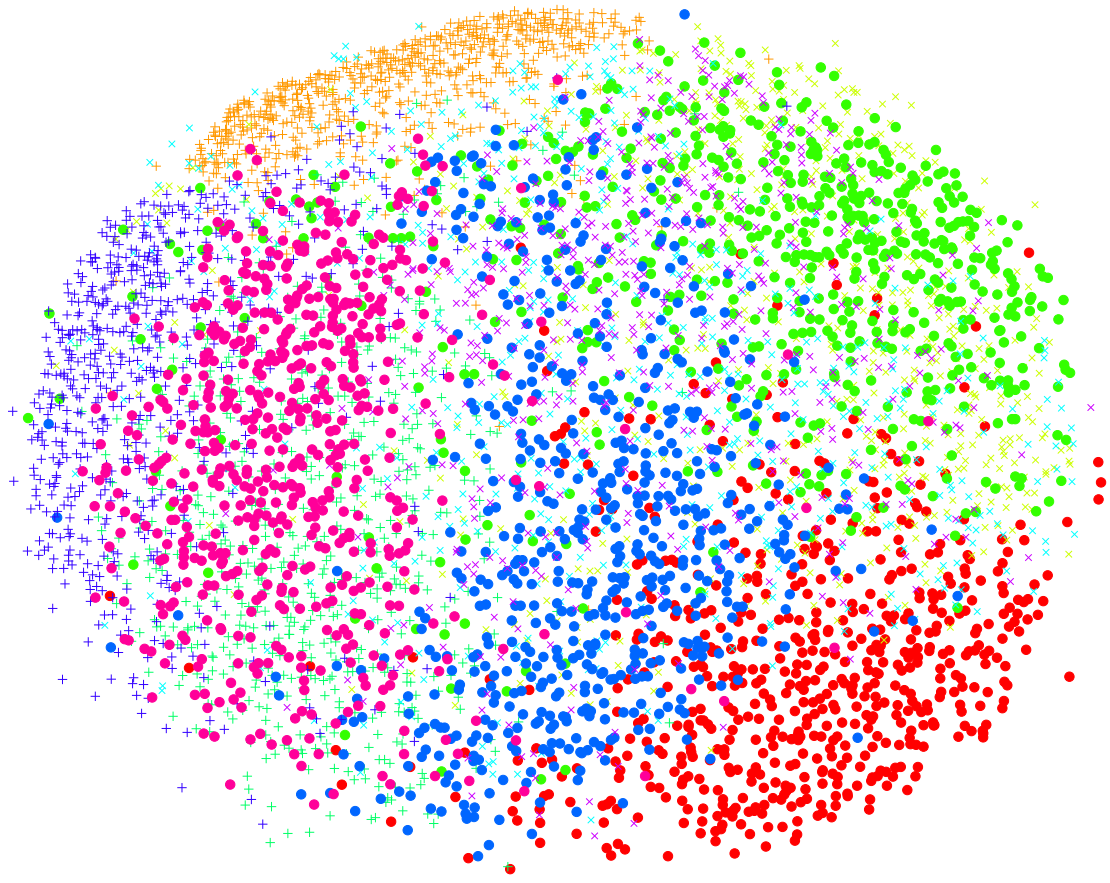
Construct a lower-dimensional representation $\mathbf{x}$ for which we can compute $\hat{d}_{ij}$ using Euclidean distance between $\mathbf{x}_i$ and $\mathbf{x}_j$

Sammons's mapping: Loss $L(\hat{d}_{ij}) = \sum_{i<j} \frac{(\hat{d}_{ij} - d_{ij})^2}{d_{ij}}$ – we want representation for which the distances are as close to the original ones as possible

Gradient-based optimization over $\mathbf{x}$ possible, but the loss function is not very nice

Computed for a fixed collection of N data points, and does not define a closed-form mapping for new points

Sammon's mapping



(b) Visualization by Sammon mapping.

From van der Maaten & Hinton, JMLR 2008

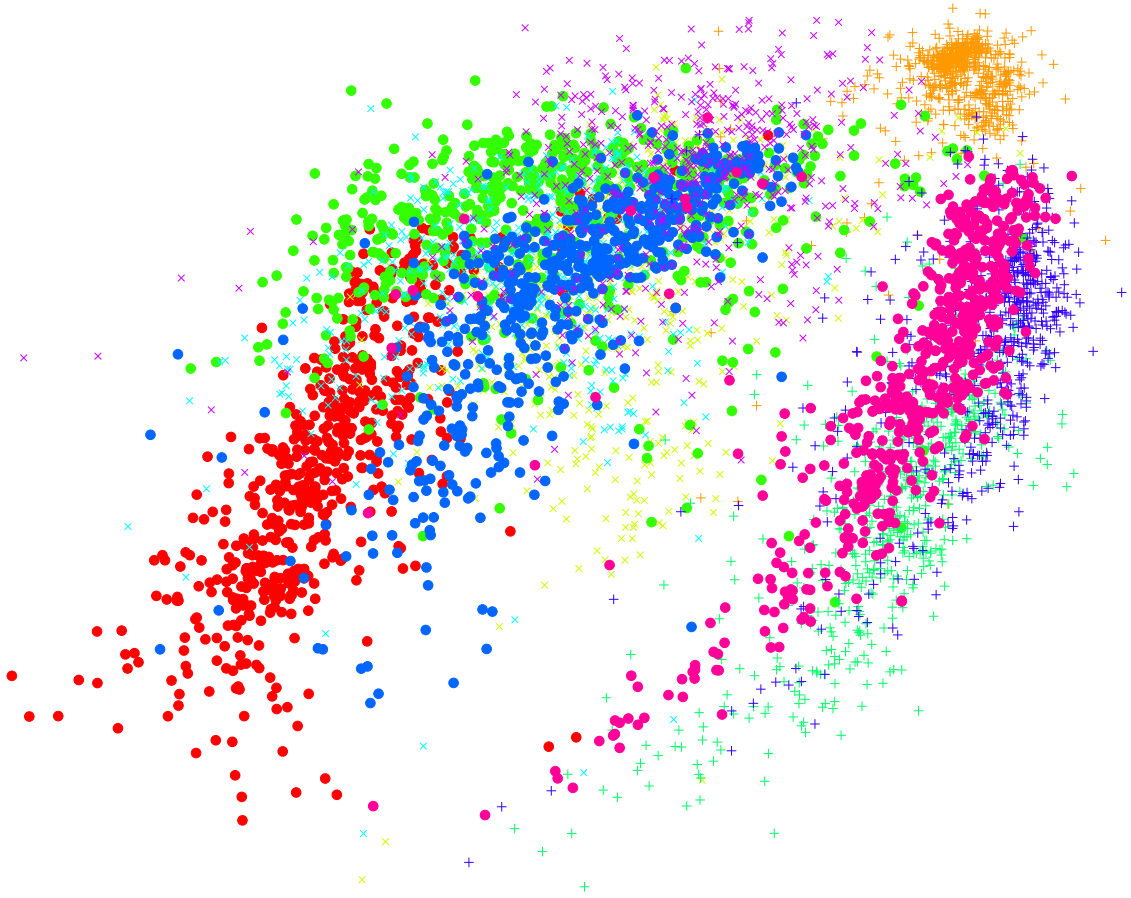
Isomap

One practical MDS algorihm is Isomap

- Compute the k nearest neighbor graph (the same we used in spectral clustering)
- Compute geodesic distances between all pairs of samples using the Floyd's algorithm
- Perform MDS on the resulting graph to find the low-dimensional representation; in practice PCA of the similarity matrix is often used

Attempts to preserve geodesic distances along the manifold, solving the swiss roll example

Isomap



(a) Visualization by Isomap.

From van der Maaten & Hinton, JMLR 2008

Stochastic neighbor embedding (SNE)

Another method that is commonly used today is SNE (or its more robust variant t-SNE)

The core idea is to match neighborhoods in the projection space with those in the original data space

Neighborhood: Probability distribution over all other samples that gives high probability for close-by samples

Low-dimensional space:

$$q_{ij} = \frac{e^{-\|\mathbf{z}_i - \mathbf{z}_j\|^2}}{\sum_{k \neq j} e^{-\|\mathbf{z}_i - \mathbf{z}_k\|^2}}$$

Original space:

$$p_{ij} = \frac{e^{-\|\mathbf{x}_i - \mathbf{x}_j\|^2}}{\sum_{k \neq j} e^{-\|\mathbf{x}_i - \mathbf{x}_k\|^2}}$$

Stochastic neighbor embedding (SNE)

The loss measures how dissimilar the two neighborhoods are, averaged over all pairs

$$L(\mathbf{z}) = KL(p, q) = \sum_{i,j} p_{ij} \log \frac{p_{ij}}{q_{ij}},$$

The Kullback-Leibler divergence is zero if and only if $p = q$

Again we can use gradient-based optimization:

$$\frac{\partial L}{\partial \mathbf{z}_i} = 2 \sum_j (\mathbf{z}_i - \mathbf{z}_j)(p_{ij} - q_{ij} + p_{ji} - q_{ji})$$

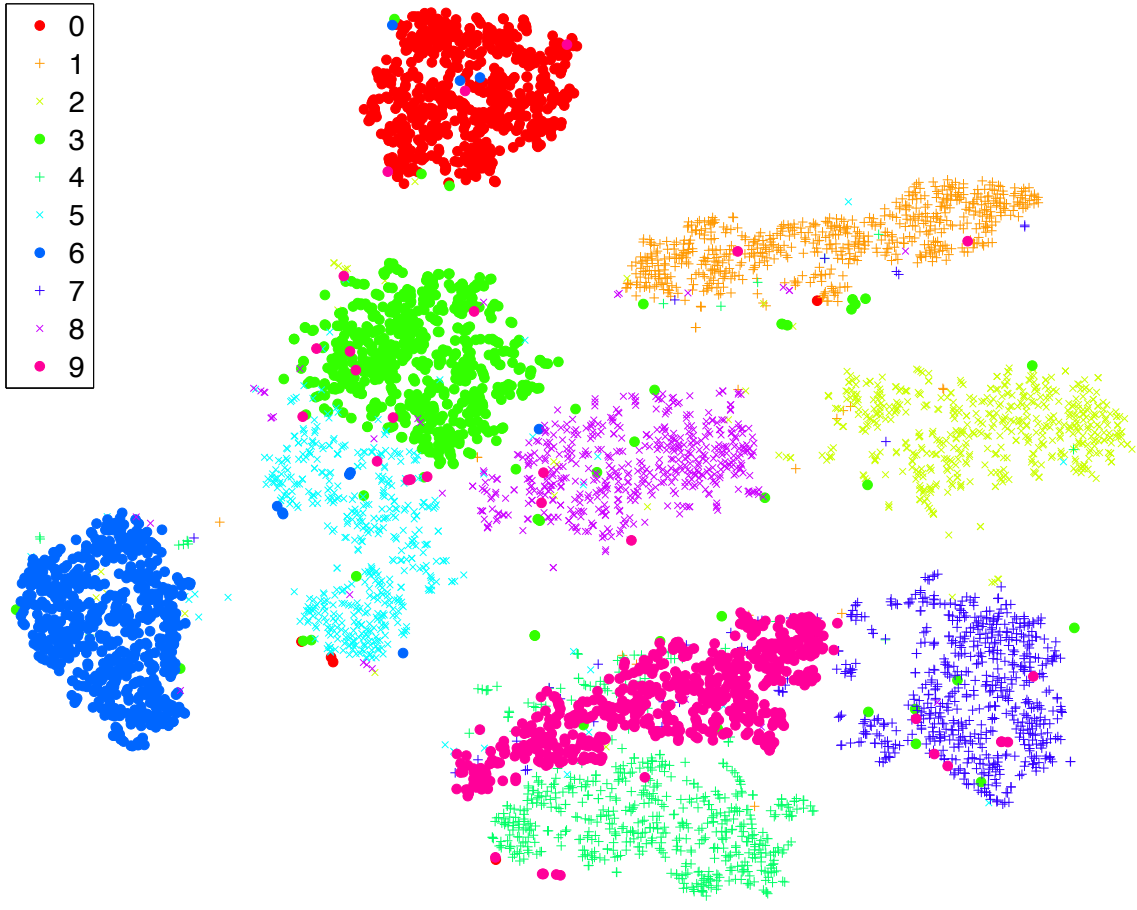
Stochastic neighbor embedding (SNE)

Attempts to capture the local structure as well as possible

Even better version, t-SNE, uses t-distributions in the embedding space to further discard global structure, often resulting in rather pretty visualizations

Evaluating the gradient is slow and the loss has numerous local optima; already the original authors propose several heuristics and afterwards more efficient algorithms have been proposed

Stochastic neighbor embedding



(a) Visualization by t-SNE.

From van der Maaten & Hinton, JMLR 2008

Other non-linear dimensionality reduction methods

Kernel-PCA:

- PCA after a non-linear data transformation represented as a kernel; we will talk about kernel methods when discussing supervised learning
- Quite close to explicitly transforming the input data and then performing PCA
- See Section 14.4.4, but note that this is not very widely used

Self-organizing map:

- Approximately k-means clustering where the clusters are organized into a grid
- Iterative updates where the closest centroid is pulled towards the sample and some of its closest neighbor in the grid are pulled with it
- We can then visualize the grid itself

Self-organizing map