

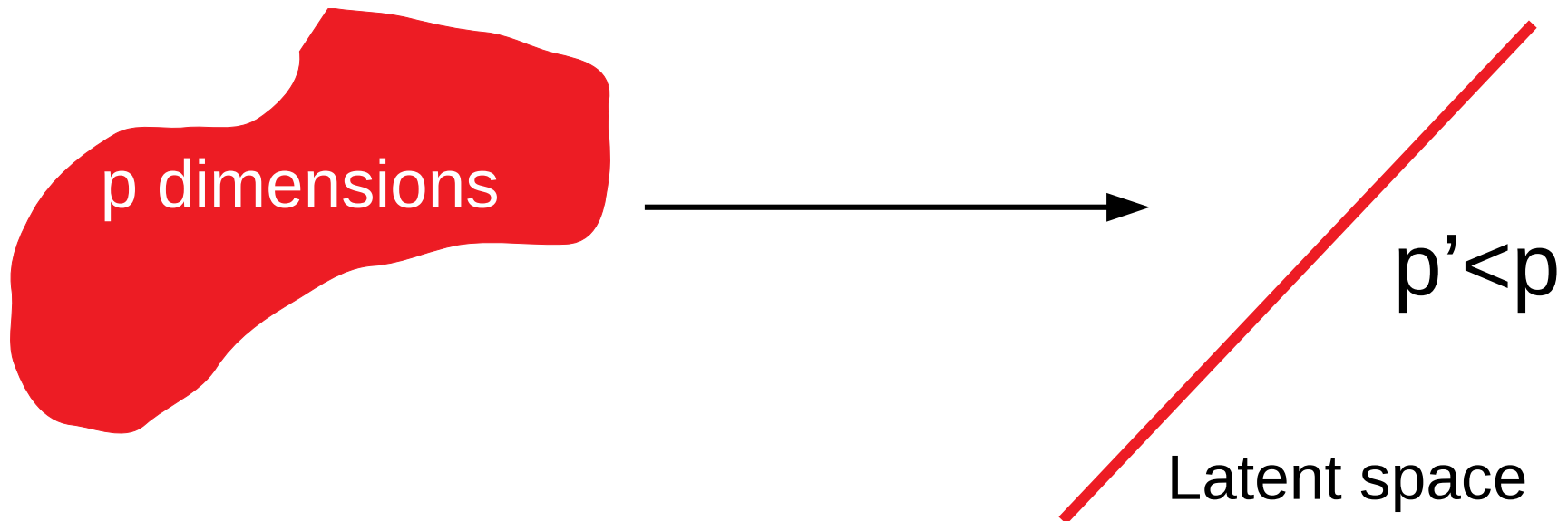
# DATA VISUALIZATION



# Data visualization

- To identify
  - correlated features
  - redundant features
  - irrelevant features (noise)
- Improves modeling
- Often: dimensional reduction

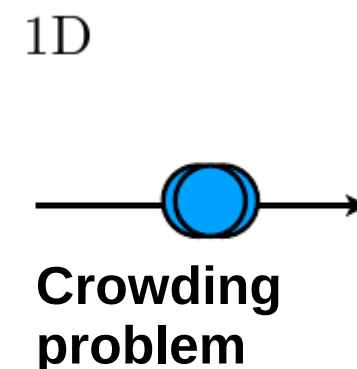
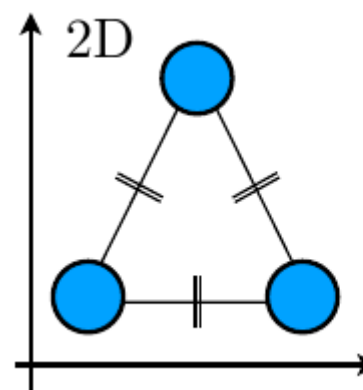
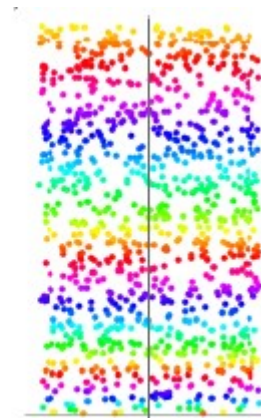
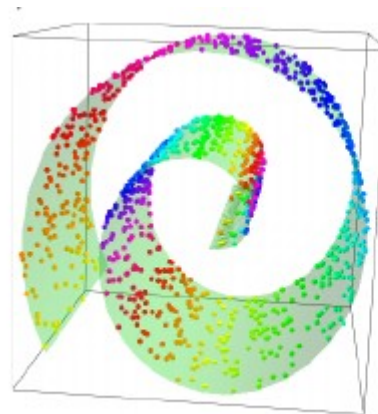
# Data visualization



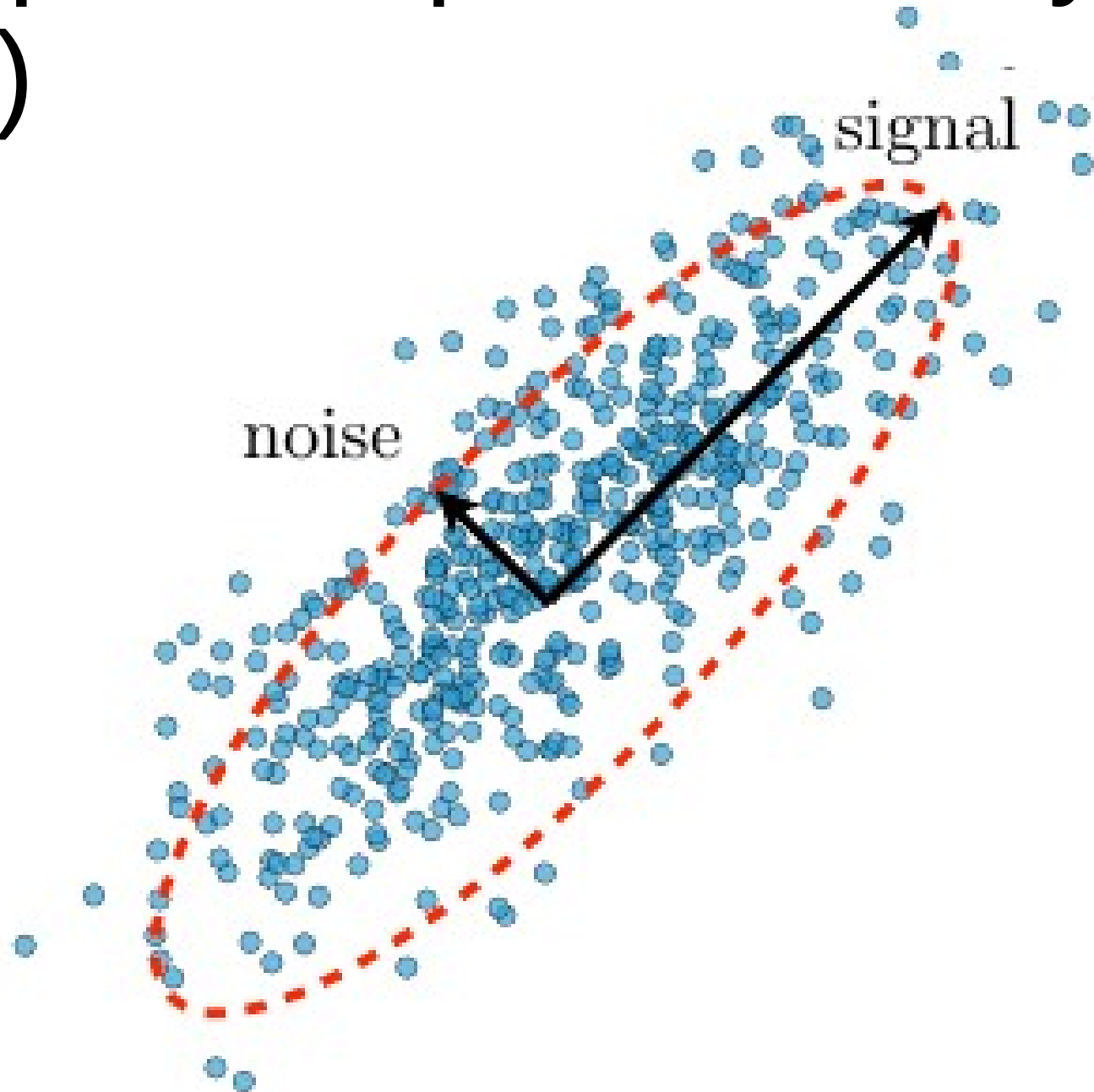
- Often: dimensional reduction

# Data visualization

- Low-dimensional representation
  - Usually data with  $p$  features lives in a manifold with smaller dimension  $p' \ll p$
- Attempts to
  - Emphasize most relevant features
  - Keep neighboring data points together

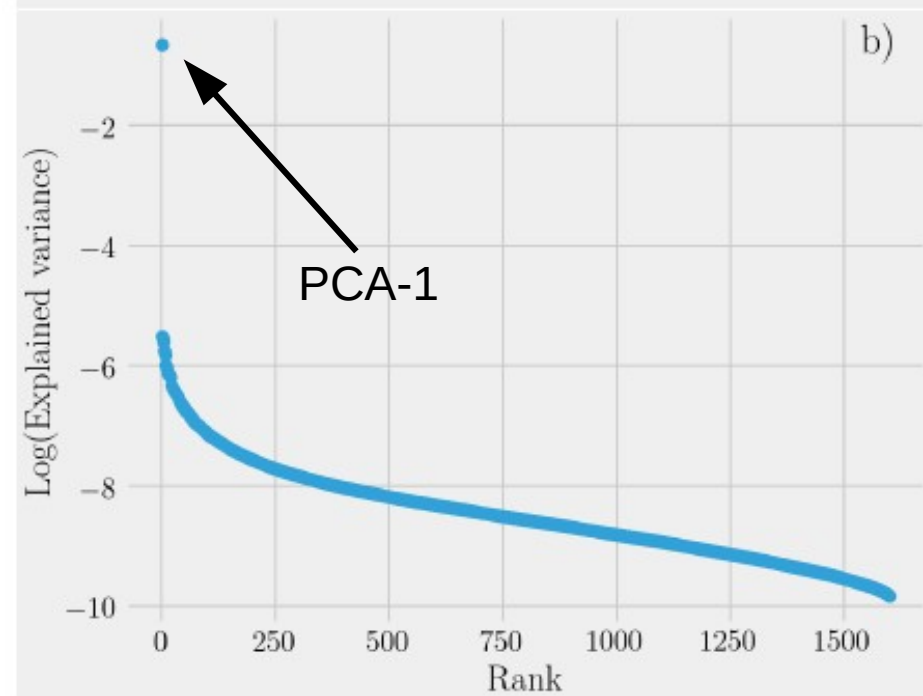
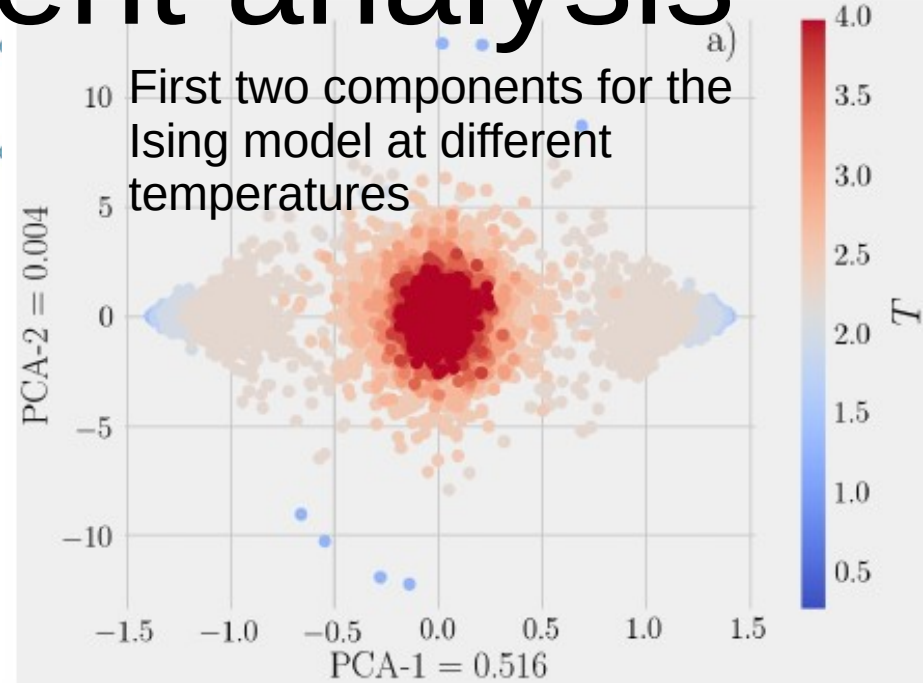
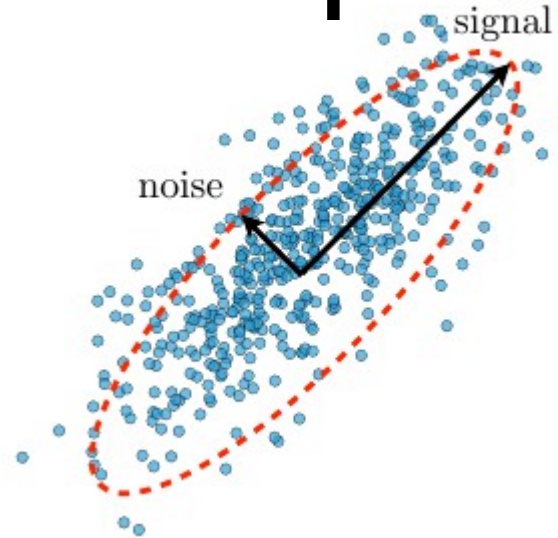


# Principal component analysis (PCA)



# Principal component analysis (PCA)

- Finds linear combinations of data components (PCA-1, PCA-2,...) with most informative content
- e.g., for distinguishing phases of the Ising model, the sum of spins = magnetization  $\sim$  PCA-1 gives the most info



# t-SNE

t-stochastic neighbor embedding

- Preserves local structure
- Non-parametric
- Nonlinear embedding
- associate a probability distribution to the neighborhood of each data  $x_i$

$$p_{i|j} = \frac{\exp(-\|x_i - x_j\|^2 / 2\sigma_i^2)}{\sum_{k \neq i} \exp(-\|x_i - x_k\|^2 / 2\sigma_i^2)},$$



- associate a probability distribution to the neighborhood of each data  $x_i$

$$p_{i|j} = \frac{\exp(-||x_i - x_j||^2 / 2\sigma_i^2)}{\sum_{k \neq i} \exp(-||x_i - x_k||^2 / 2\sigma_i^2)}, \quad (132)$$

where  $p_{i|j}$  can be interpreted as the likelihood that  $x_j$  is  $x_i$ 's neighbor (thus we take  $p_{i|i} = 0$ ).  $\sigma_i$  are free bandwidth parameters that are usually determined by fixing the local entropy  $H(p_i)$  of each data point:

$$H(p_i) \equiv - \sum_j p_{j|i} \log_2 p_{j|i}. \quad (133)$$

The local entropy is then set to equal a constant across *all data points*  $\Sigma = 2^{H(p_i)}$ , where  $\Sigma$  is called the *perplexity*. The perplexity constraint determines  $\sigma_i \forall i$  and implies that points in regions of high-density will have smaller  $\sigma_i$ .






- symmetrized probability distribution

$$p_{ij} \equiv (p_{i|j} + p_{j|i}) / (2N)$$

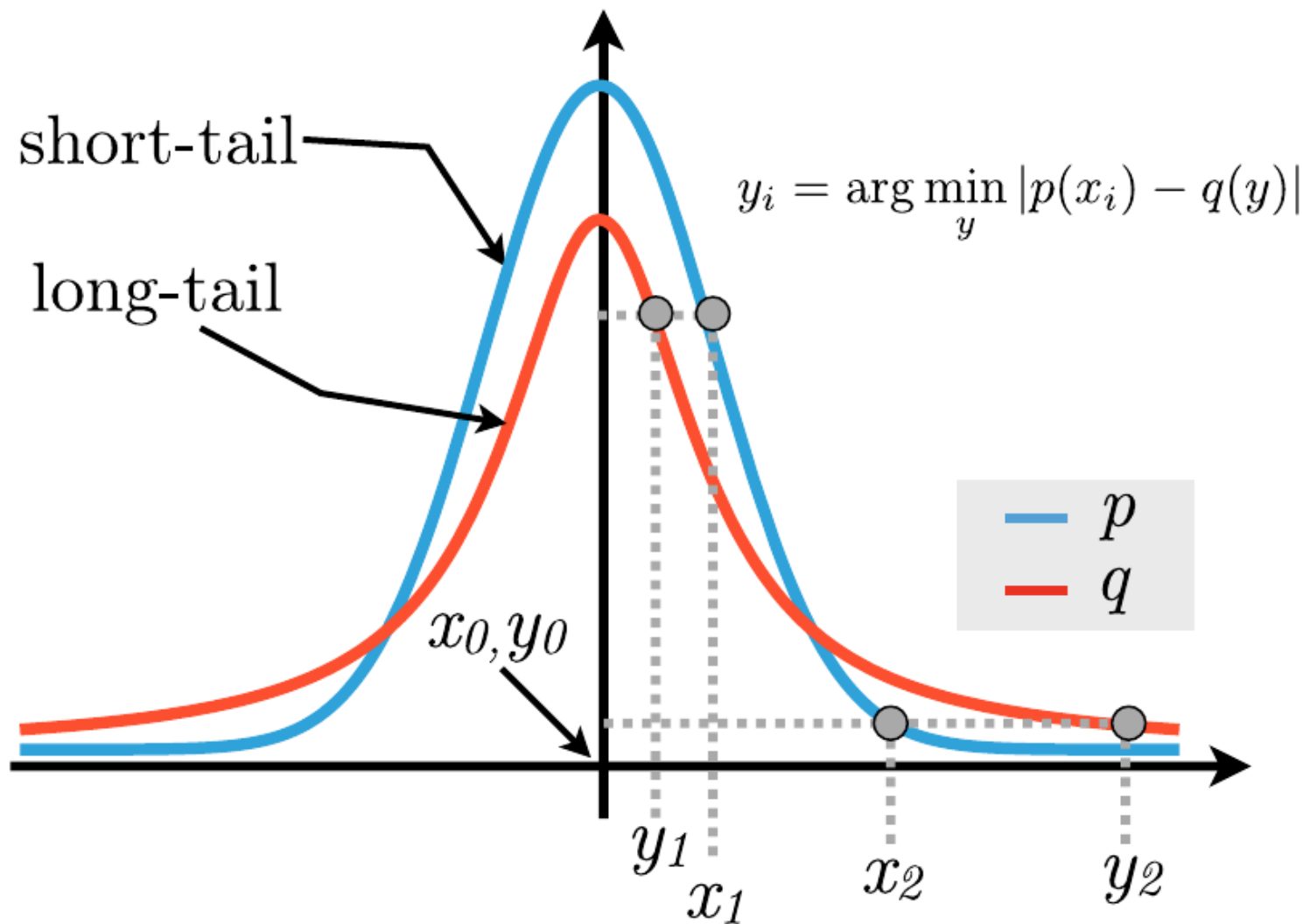
t-SNE constructs a *similar* probability distribution  $q_{ij}$  in a low dimensional latent space (with coordinates  $Y = \{y_i\}$ ,  $y_i \in \mathbb{R}^{p'}$ , where  $p' < p$  is the dimension of the latent space):


$$q_{ij} = \frac{(1 + \|y_i - y_j\|^2)^{-1}}{\sum_{k \neq i} (1 + \|y_i - y_k\|^2)^{-1}}. \quad (134)$$

The crucial point to note is that  $q_{ij}$  is chosen to be a long tail distribution. This preserves short distance information (relative neighborhoods) while strongly repelling two points that are far apart in the original space (see FIG.



# p vs q



52). In order to find the latent space coordinates  $y_i$ , t-SNE minimizes the Kullback-Leibler divergence between  $q_{ij}$  and  $p_{ij}$ :

💬 
$$\mathcal{C}(Y) = D_{KL}(p||q) \equiv \sum_{ij} p_{ij} \log \left( \frac{p_{ij}}{q_{ij}} \right). \quad (135)$$

This minimization is done via gradient descent (see section IV). We can gain further insights on what the embedding cost-function  $\mathcal{C}$  is capturing by computing the gradient of (135) with respect to  $y_i$  explicitly:

$$\begin{aligned} \partial_{y_i} \mathcal{C} &= \sum_{j \neq i} 4p_{ij}q_{ij}Z_i(y_i - y_j) - \sum_{j \neq i} 4q_{ij}^2Z_i(y_i - y_j), \\ &= F_{\text{attractive},i} - F_{\text{repulsive},i}, \end{aligned} \quad (136)$$

where  $Z_i = 1/(\sum_{k \neq i} (1 + \|y_k - y_i\|^2)^{-1})$ . We have separated the gradient of point  $y_i$  into an attractive  $F_{\text{attractive}}$  and repulsive term  $F_{\text{repulsive}}$ . Notice that  $F_{\text{attractive},i}$  in-

**Minimizing a KL divergence is a classic in unsupervised learning**

Originally KL introduced a symmetric metrics  $D(q|p)+D(p|q)$

# t-SNE, check that it...

- gives **stochastic** results
- can **rotate** data
- generally **preserves short distance** information
- **deforms** scales
- is computationally intensive

( $O(N^2)$ , with Barnes-Hut method  $O(N \ln N)$  )



# Other nonlinear methods

- Check the scikit webpage for examples comparing performances of several methods (Hessian, etc...)
- Consider writing another python notebook that visualises data with all these methods in parallel

[Previous](#)  
Comparison  
of...

[Next](#)  
Manifold  
lear...

[Up](#)  
Examples

**scikit-learn v0.20.3**
[Other versions](#)

Please [cite us](#) if you use  
the software.

**Manifold Learning methods on a  
severed sphere**

**Note:** Click [here](#) to download the full example code

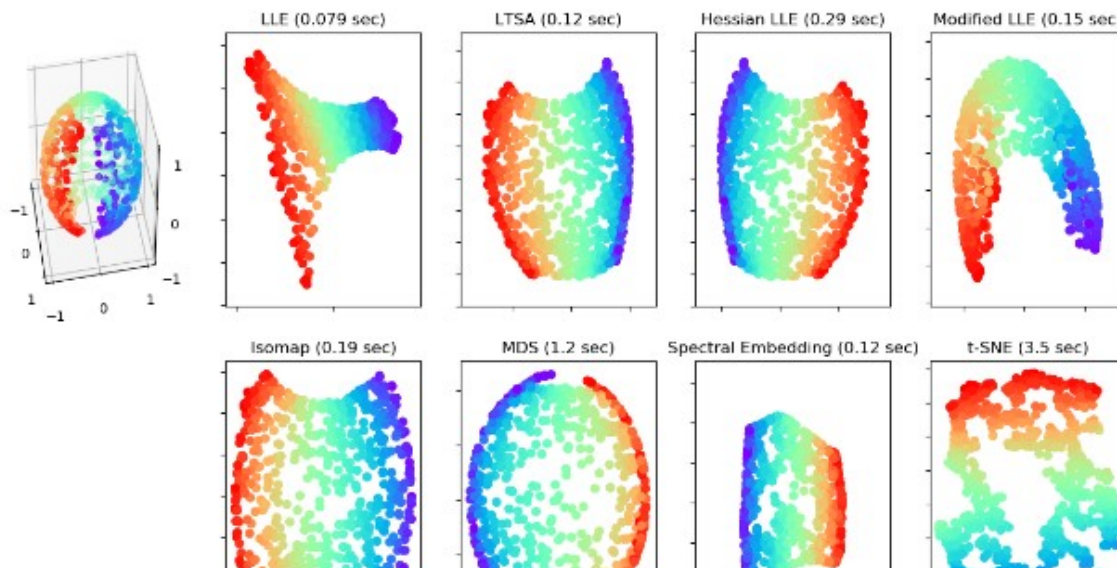
## Manifold Learning methods on a severed sphere

An application of the different [Manifold learning](#) techniques on a spherical data-set. Here one can see the use of dimensionality reduction in order to gain some intuition regarding the manifold learning methods. Regarding the dataset, the poles are cut from the sphere, as well as a thin slice down its side. This enables the manifold learning techniques to 'spread it open' whilst projecting it onto two dimensions.

For a similar example, where the methods are applied to the S-curve dataset, see [Comparison of Manifold Learning methods](#)

Note that the purpose of the [MDS](#) is to find a low-dimensional representation of the data (here 2D) in which the distances respect well the distances in the original high-dimensional space, unlike other manifold-learning algorithms, it does not seeks an isotropic representation of the data in the low-dimensional space. Here the manifold problem matches fairly that of representing a flat map of the Earth, as with [map projection](#)

Manifold Learning with 1000 points, 10 neighbors



**Points on a sphere  
cannot be projected  
by PCA**

# Material: download



[Home](#) [Installation](#) [Documentation](#) [Examples](#)

Google Custom Search



[Previous](#)  
Multi-  
dimensi...

[Next](#)  
Comparison  
of...

[Up](#)  
Examples

scikit-learn v0.20.3  
[Other versions](#)

Please [cite us](#) if you use  
the software.

**t-SNE: The effect of various  
perplexity values on the shape**

**Note:** Click [here](#) to download the full example code

## t-SNE: The effect of various perplexity values on the shape

An illustration of t-SNE on the two concentric circles and the S-curve datasets for different perplexity values.

We observe a tendency towards clearer shapes as the perplexity value increases.

The size, the distance and the shape of clusters may vary upon initialization, perplexity values and does not always convey a meaning.

As shown below, t-SNE for higher perplexities finds meaningful topology of two concentric circles, however the size and the distance of the circles varies slightly from the original. Contrary to the two circles dataset, the shapes visually diverge from S-curve topology on the S-curve dataset even for larger perplexity values.

For further details, "How to Use t-SNE Effectively" <http://distill.pub/2016/misread-tsne/> provides a good discussion of the effects of various parameters, as well as interactive plots to explore those effects.

