Projections

Dimensionality Reduction

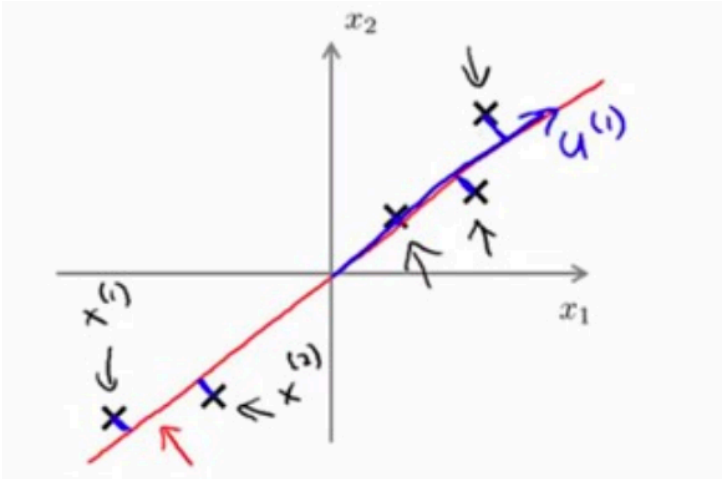
Dimensionality reduction: taking high dimensional data and embedding it in a lower dimensional space while preserving the “characteristics” of the data

Projections

Low variance projection Maximal Variance Projection

Figure 4. Dimensionality reduction from 3D to 2D.

Figure Source: Opinion space: *A scalable tool for browsing online comments*, Bitton et al., CHI 2010



Reduce data from 2D to 1D

Figure Source: Slides by Andrew Ng

Mathematically, the above projections boil down to matrix-vector/matrix multiplication.

Matrix Factorization/Decomposition

Non Negative Matrix Factorization

Idea: NMF will separate the matrix into two matrices. In which one can interpreted as topics. While the other shows how to different topics to reconstruct the word distribution of the original document.

Drawback: NP-Hard, not guaranteed unique solution

Singular Value Decomposition

Idea: Generalization of eigendecomposition for nonsymmetric matrices.

Drawback: Expensive to compute

DataScience@SMU

Linear Dimensionality Reduction

Linear Dimensionality Reduction

Principal Component Analysis (PCA): express the data as linear independent vectors (called principal components) that have the largest possible variance (hence, approximate the data with minimal reconstruction error).

Low variance projection Maxima variance Projection

Figure 4. Dimensionality reduction from 3D to 2D.

X: $n \times d$ data matrix, **D**: $d \times d$ covariance matrix; K: the desired # of dimensions

Algorithm 37 PCA($\mathbf{D}$, K)

1: $\mu \leftarrow \text{MEAN}(\mathbf{X})$ // compute data mean for centering

2: $\mathbf{D} \leftarrow \left(\mathbf{X} - \mu\mathbf{1}^\top\right)^\top \left(\mathbf{X} - \mu\mathbf{1}^\top\right)$ // compute covariance, $\mathbf{1}$ is a vector of ones

3: $\{\lambda_k, u_k\} \leftarrow$ top K eigenvalues/eigenvectors of $\mathbf{D}$

4: **return** $(\mathbf{X} - \mu\mathbf{1})\,\mathbf{U}$ // project data using $\mathbf{U}$

Figure source: *A Course in Machine Learning*, Ha Daume III (http://ciml.info/)

DataScience@SMU

Nonlinear Dimensionality Reduction

Nonlinear Dimensionality Reduction

One example is **Laplacian eigenmaps**, which assumes that the data lies on a low dimensional manifold in high dimensions:

- → build a k-nn graph of the data points on your dataset $\mathrm{D} = \{x_1, x_2, \ldots, x_m\}$ (the graph is effectively an approximation of the manifold)
- → compute the eigenfunctions of the resulting $m \times m$ Laplacian
- → project the data points onto the eigenfunctions

Many other similar types of approaches

Nonlinear Dimensionality Reduction

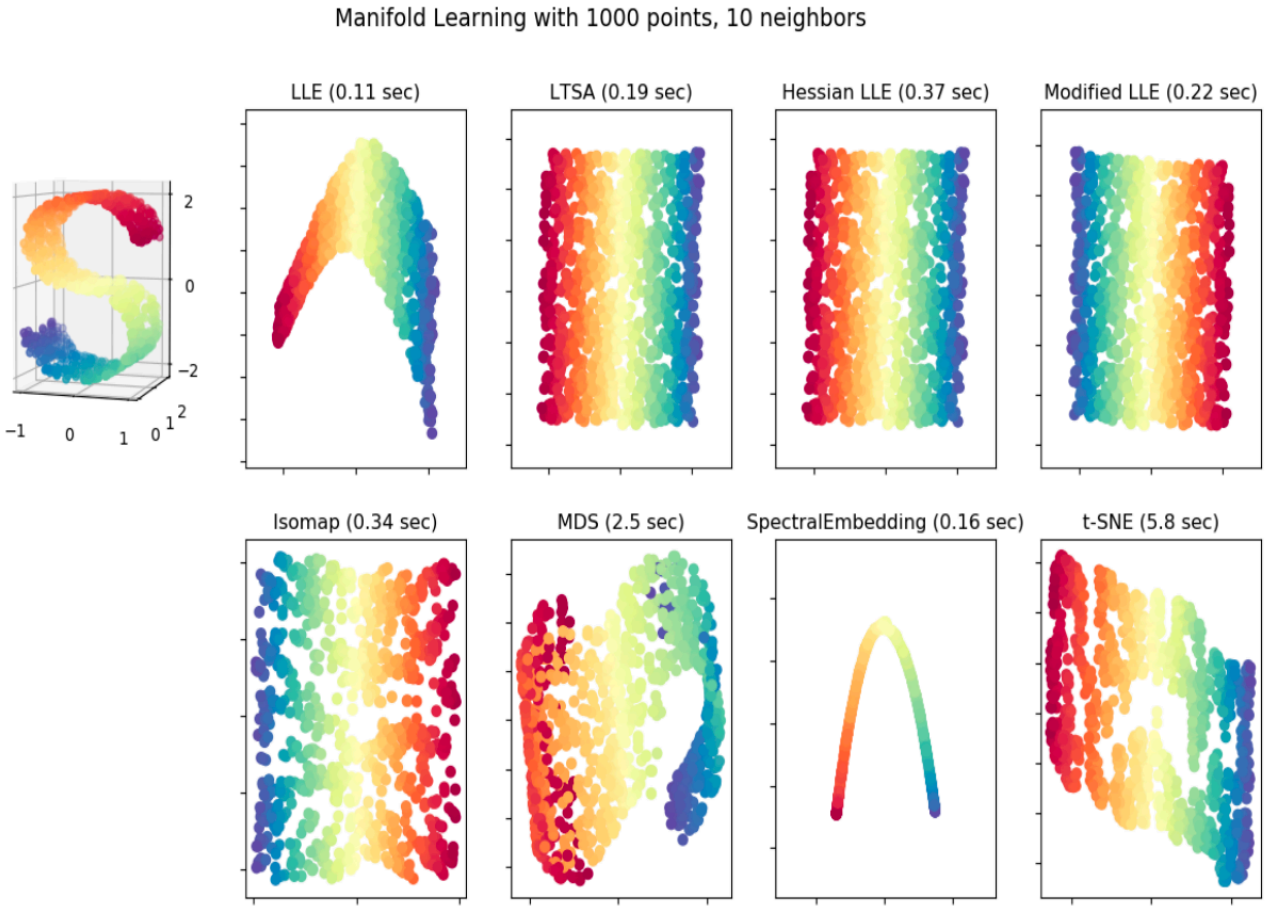


Figure source: https://scikit-learn.org/stable/modules/manifold.html

DataScience@SMU

Dimensionality “Expansion”

Kernels and the Kernel Trick

Kernel Methods

Linear models are easy to understand and easy to optimize but can only learn very simple decision boundaries. One way of getting a linear model to behave nonlinearly is to transform the input—for instance, by adding feature pairs as additional inputs. Learning a linear model on such a representation is convex but is computationally prohibitive. **Kernel techniques** allow us to implicitly expand the feature space.

Source: Chapter 11, *A Course in Machine Learning*, Ha Daume III (http://ciml.info/)

Kernel Methods

- A kernel is a form of generalized dot product.
- If you can't separate + from − in a low-dimensional space using a hyperplane, map the points to a higher dimensional space where you can separate them.
- Examples: polynomial, Gaussian, sigmoid

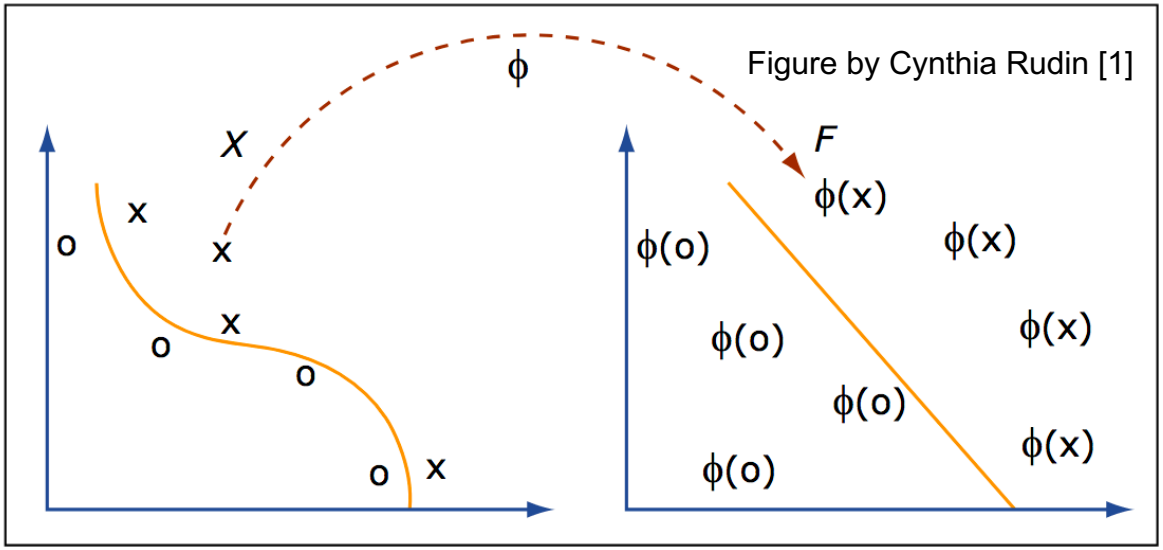


Figure by Cynthia Rudin [1]

Image by MIT OpenCourseWare.

The (Kernel Trick) SVM

The kernel trick is that if you have an algorithm (like SVM) where the examples appear only in inner products, you can freely replace the inner product with a different one. (And it works just as well as if you designed the SVM with some map $\mathbf{\Phi}(\mathbf{x})$ in mind in the first place, and took the inner product in $\mathbf{\Phi}'$s space instead.)

Remember the optimization problem for SVM?

$$K(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{\|\mathbf{x} - \mathbf{x}'\|^2}{2\sigma^2}\right)$$

$$\max_{\alpha} \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,k=1}^{m} \alpha_i \alpha_j y_i y_k \, \mathbf{x}_i^T \mathbf{x}_k$$

← Inner product

$$s.t.\ 0 \leq \alpha_i \leq C, i = 1, \ldots, m \ and \sum_{i=1}^{m} \alpha_i y_i = 0$$

You can replace this inner product with another one, *without even knowing* $\mathbf{\Phi}$. In fact there can be many different feature maps that correspond to the same inner product.

References

[1] Slides by Cynthia Rudin,
https://ocw.mit.edu/courses/sloan-school-of-management/15-097-prediction-machine-learning-and-statistics-spring-2012/lecture-notes/MIT15_097S12_lec13.pdf

DataScience@SMU