Introduction to Machine Learning Methods in Condensed Matter Physics

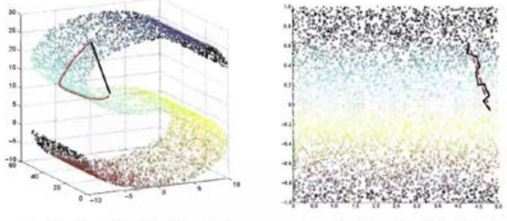
LECTURE 10 PART 2, FALL 2021

Pei-Lin Zheng (郑沛林)
Yi Zhang (张亿)
International Center for Quantum Materials, School of Physics
Peking University, Beijing, 100871, China

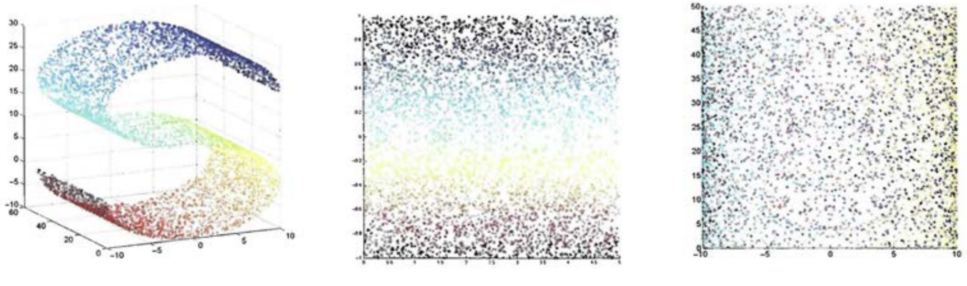
Email: frankzhangyi@pku.edu.cn

Stochastic Neighbor Embedding

- Nonlinear dimensionality reduction algorithm
- Manifold learning:
- Method:

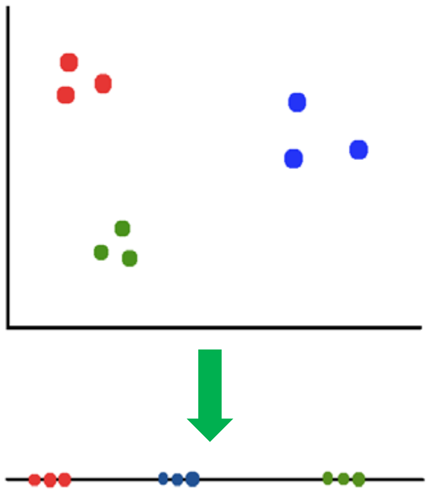


(a) 测地线距离与高维直线距离 (b) 测地线距离与近邻距离

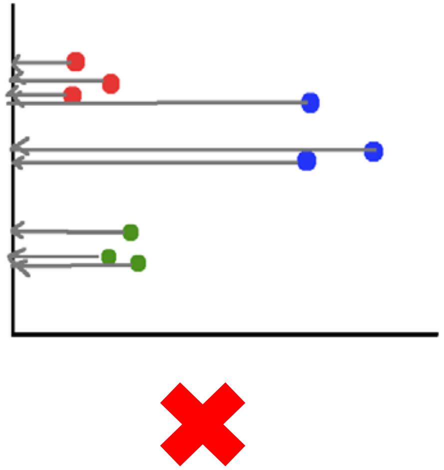


(a) 三维空间中的观察 (b) 本真二维结构 (c) PCA 降维结果

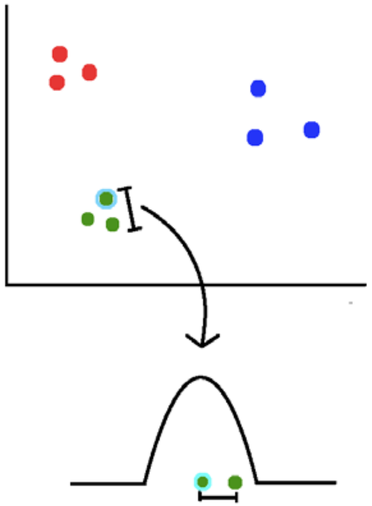
Task

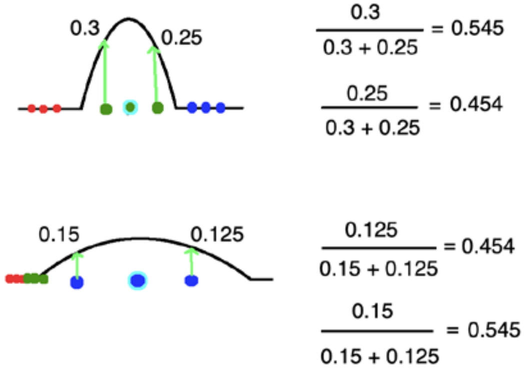


PCA



Solution





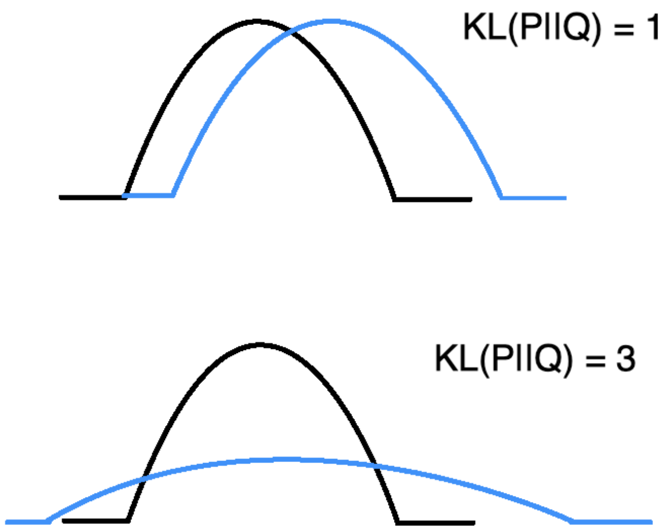
https://towardsdatascience.com/t-sne-python-example-1ded9953f26

Stochastic Neighbor Embedding

- Original space: $p_{j|i} = \frac{\exp\left(-\|x_i - x_j\|^2 / 2\sigma_i^2\right)}{\sum_{k \neq i} \exp\left(-\|x_i - x_k\|^2 / 2\sigma_i^2\right)}$
- Target space: $q_{j|i} = \frac{\exp\left(-\|y_i - y_j\|^2\right)}{\sum_{k \neq i} \exp\left(-\|y_i - y_k\|^2\right)}$
- Loss function: $C = \sum_i KL(P_i || Q_i) = \sum_i \sum_j p_{j|i} \log \frac{p_{j|i}}{q_{j|i}}$
- Gradient: $\frac{\delta C}{\delta y_i} = 2 \sum_j \left(p_{j|i} - q_{j|i} + p_{i|j} - q_{i|j}\right)\left(y_i - y_j\right)$
- Update: $Y^{(t)} = Y^{(t-1)} + \eta \frac{\delta C}{\delta Y} + \alpha(t)\left(Y^{(t-1)} - Y^{(t-2)}\right)$
- One more question: how to determine σ_i?
  - Give perplexity, use binary search to find.

$$\begin{cases} Perp(P_i) = 2^{H(P_i)} \\ H(P_i) = -\sum_j p_{j|i} \log_2\left(p_{j|i}\right) \end{cases}$$

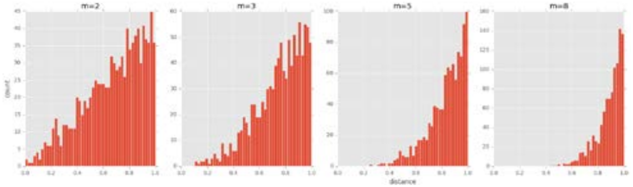
Kullback-Leibler divergences



https://towardsdatascience.com/t-sne-python-example-1ded9953f26

t-Distributed Stochastic Neighbor Embedding

- Disadvantage of SNE:
  - Crowding problem: the clusters are clustered together and indistinguishable.
  - Slow: $O(N^2 i)$, where N is the number of samples, i is iteration times.
  - Cannot be generalized.



- How to improve $\Longrightarrow$ t-Distributed Stochastic Neighbor Embedding
  - Symmetrization
    - Replace the conditional probability with the joint probability,

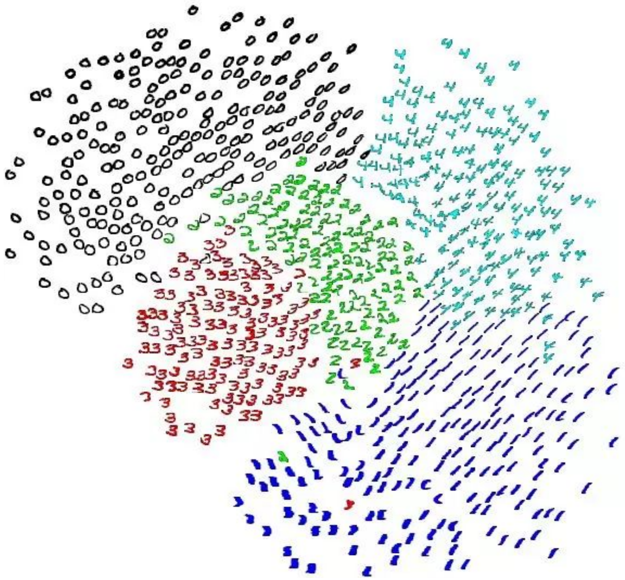
$$p_{ij} = \frac{\exp\left(-\|x_i - x_j\|^2 / 2\sigma_i^2\right)}{\sum_{k \neq l} \exp\left(-\|x_k - x_l\|^2 / 2\sigma_i^2\right)} \qquad q_{ij} = \frac{\exp\left(-\|y_i - y_j\|^2\right)}{\sum_{k \neq l} \exp\left(-\|y_k - y_l\|^2\right)},$$

    but it introduces the problem of outliers, especially in high-dimensional space.

    - In the original space, adopt $p_{ij} = \frac{p_{i|j} + p_{j|i}}{2N}$ in practice, $\sum_{i,j} p_{ij} = 1$, $\sum_j p_{ij} > \frac{1}{2N}$,

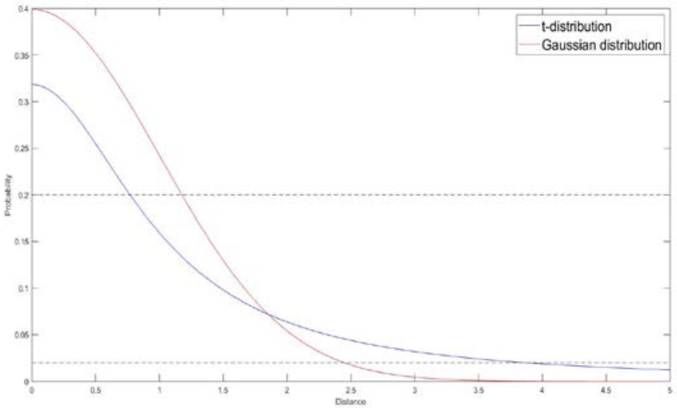
    gradient will be simplified:

$$\frac{\delta C}{\delta y_i} = 4 \sum_j (p_{ij} - q_{ij})(y_i - y_j)$$



t-Distributed Stochastic Neighbor Embedding

- Use t-distribution instead of Gaussian distribution in the target space
  - $q_{ij} = \frac{\left(1+\|y_i - y_j\|^2\right)^{-1}}{\sum_{k \neq l}(1+\|y_k - y_l\|^2)^{-1}}$
  - $\frac{\delta C}{\delta y_i} = 4\sum_j (p_{ij} - q_{ij})(y_i - y_j)\left(1 + \|y_i - y_j\|^2\right)^{-1}$



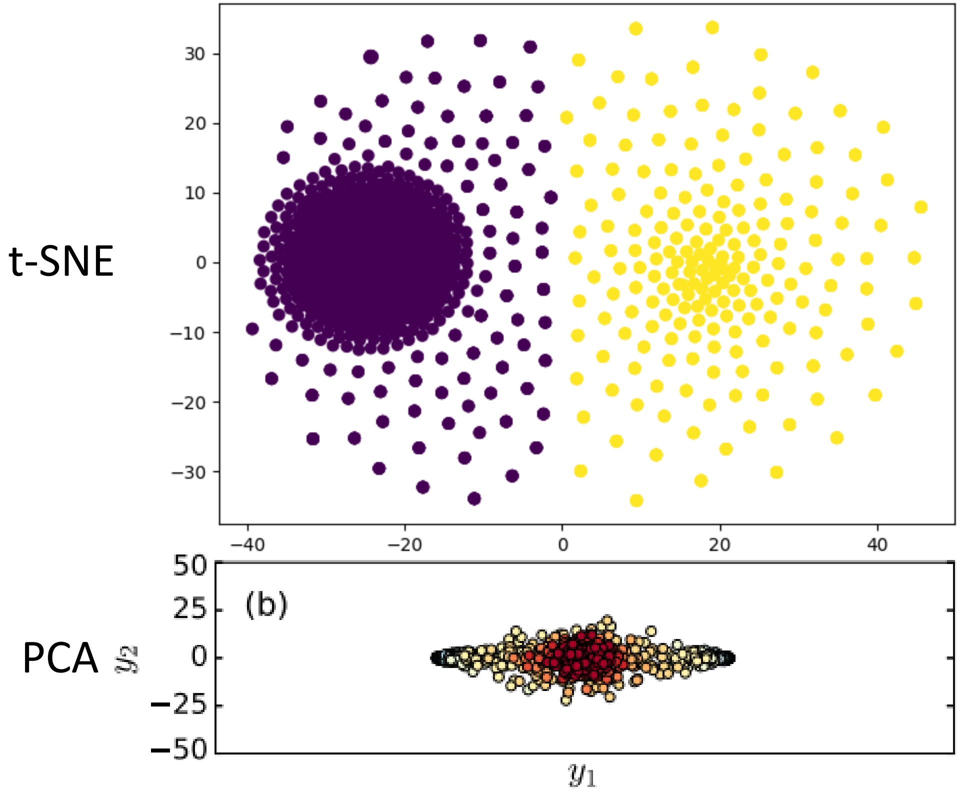
- Algorithm
  - Step 1: (optional) preprocess the data with PCA;
  - Step 2: search σ_i by given perplexity and compute original distribution p_{ij};
  - Step 3: initialize the coordinates y_i of the target space after dimension reduction;
  - Step 4: compute target distribution q_{ij} and optimize the loss function $C = \sum_i KL(P_i||Q_i)$

https://lvdmaaten.github.io/tsne/

https://lvdmaaten.github.io/publications/papers/JMLR_2008.pdf

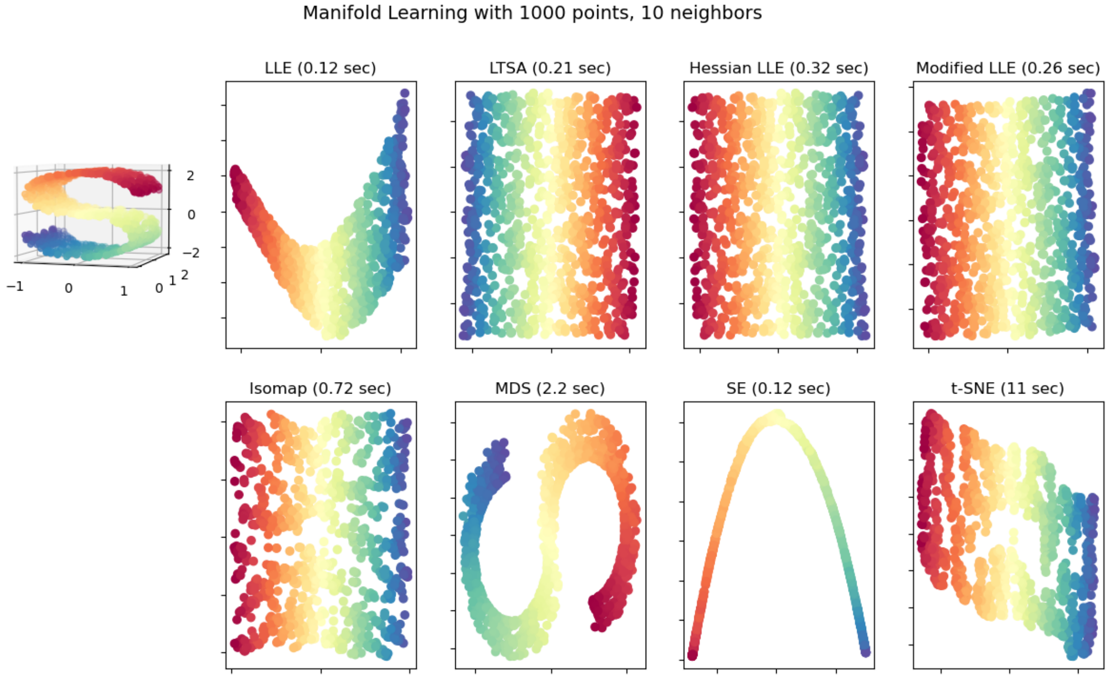
t-Distributed Stochastic Neighbor Embedding

- Example: t-SNE on 2D classical Ising model



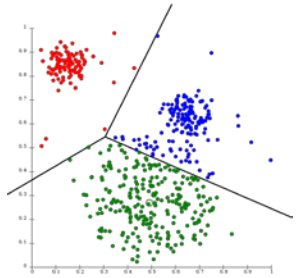
*Lei Wang, Phys. Rev. B **94**, 195105 (2016).*

- Disadvantage of t-SNE:
  - Slow in practice, complexity is still $O(N^2 i)$.
  - Still cannot be generalized.



https://lvdmaaten.github.io/publications/misc/Supplement_JMLR_2008.pdf

k-means

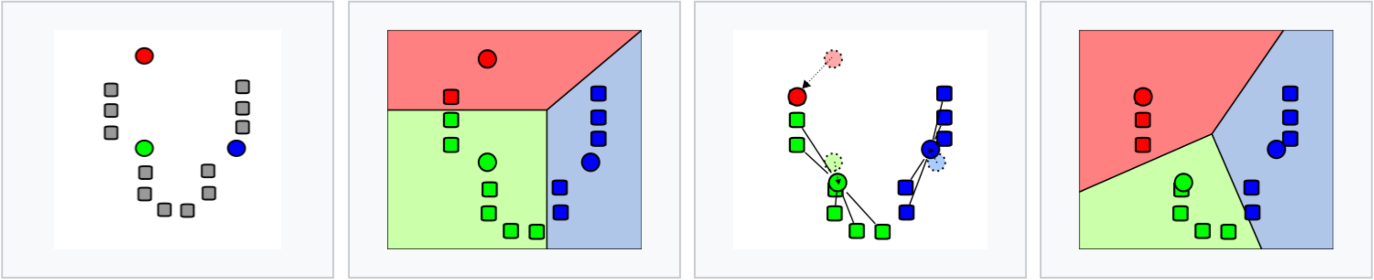


- Clustering: group a set of objects in such a way that objects in the same group.
- After clustering, the obtained model can be used for prediction.
- k-means:
  - k cluster: $\mathcal{C} = \{C_1, C_2, \cdots, C_k\}$
  - k mean of points in $\mathcal{C}$:

    $$\boldsymbol{\mu} = \{\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \cdots, \boldsymbol{\mu}_k\}$$
  - Objective :

    $$\arg\min_{C} \sum_{i=1}^{k} \sum_{\boldsymbol{x} \in C_i} \|\boldsymbol{x} - \boldsymbol{\mu}_i\|_2^2$$
  - NP-hard
  - Greedy strategy
    - Iterative optimization
    - Complexity: $O(Ndki)$
      - N: number of samples
      - d: dimension of the sample
      - i: iteration times

- Algorithm:



1. k initial "means" (in this case k=3) are randomly generated within the data domain (shown in color).

2. k clusters are created by associating every observation with the nearest mean. The partitions here represent the Voronoi diagram generated by the means.

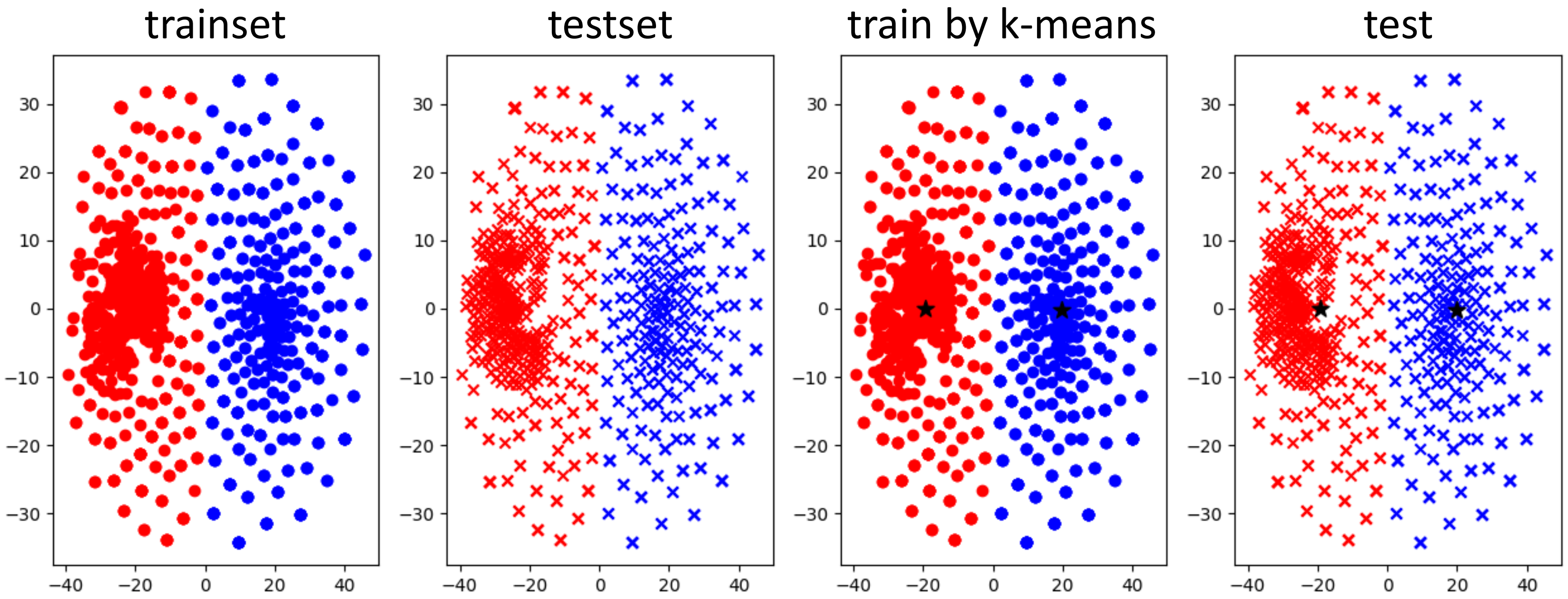
3. The centroid of each of the k clusters becomes the new mean.

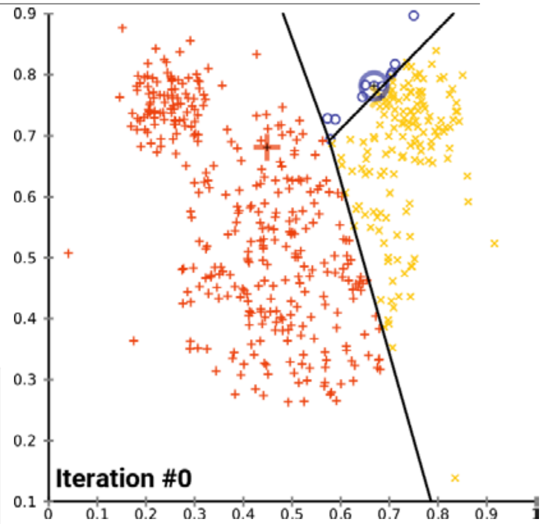
4. Steps 2 and 3 are repeated until convergence has been reached.

https://en.wikipedia.org/wiki/K-means_clustering

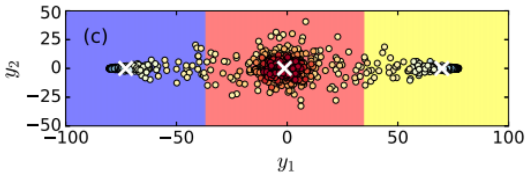
k-means

- Example: k-means on 2D classical Ising model
  - Divide the data after t-SNE dimensionality reduction into trainset and testset
  - Run k-means on the trainset, and predict on the testset
  - Accuracy: 100%





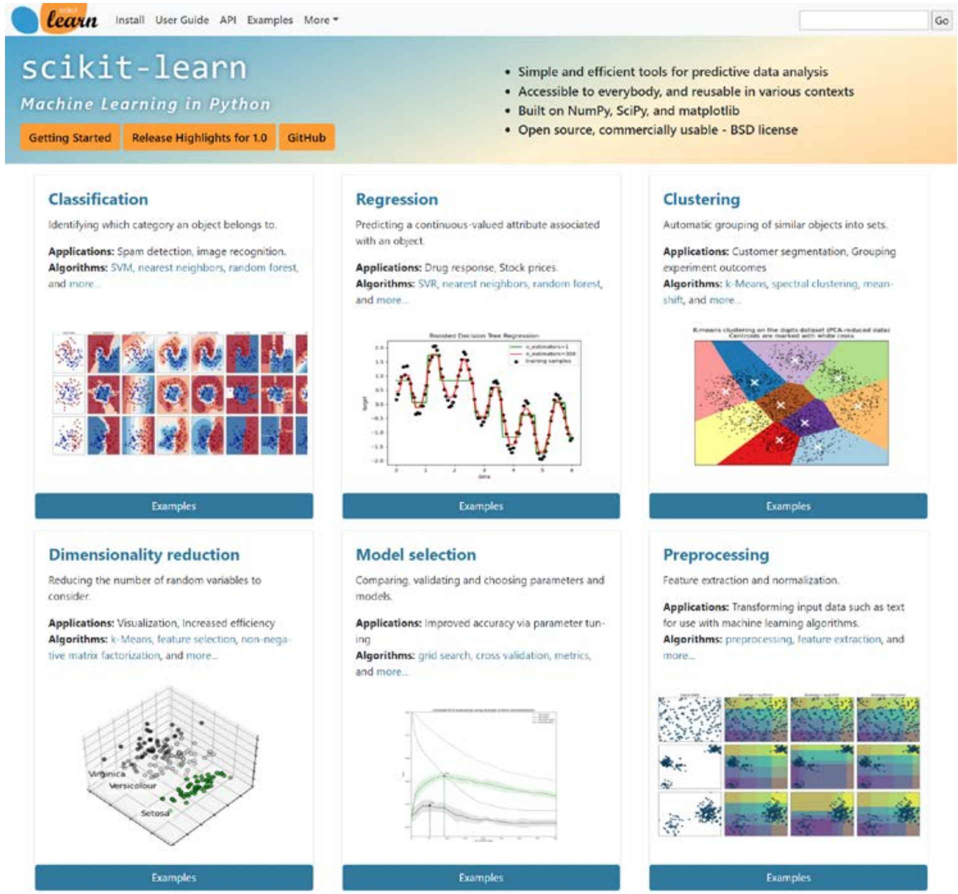
- k-means for the data after PCA dimensionality reduction



*Lei Wang, Phys. Rev. B **94**, 195105 (2016).*

Scikit-learn

- https://scikit-learn.org/stable/
- sklearn.neural_network
  - BernoulliRBM
  - MLPClassifier
  - MLPRegressor
- sklearn.svm
- sklearn.decomposition
  - PCA
- sklearn.manifold
  - TSNE
- sklearn.cluster
  - KMeans



Scikit-learn

- Example: t-SNE and k-means on 2D classical Ising model