Technische Universität Braunschweig

Machine learning

Mini-project: NCA - Neighborhood Components Analysis

Cui Xiao

Catalog

- Introduction of NCA
- Performance of NCA
- Compare with other similar algorithms
- My Code

NCA - Neighborhood Components Analysis

Neighbourhood components analysis is a supervised learning method for classifying high dimension data into distinct dimension data according to **a distance metric** learning from data.

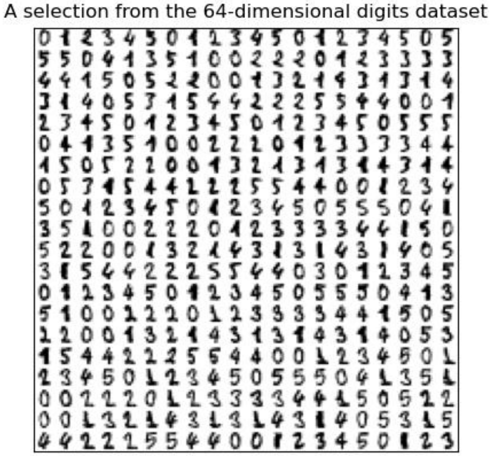
Some keyword：

- Dimension reduction und Feature selection
- Mapping to a new space
- Boosting the effect of KNN

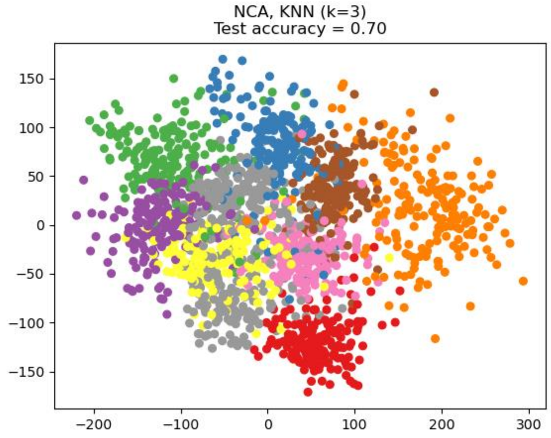
What is the NCA?

Dataset: MNIST

Dimension reduction



Features: 28*28 = 784

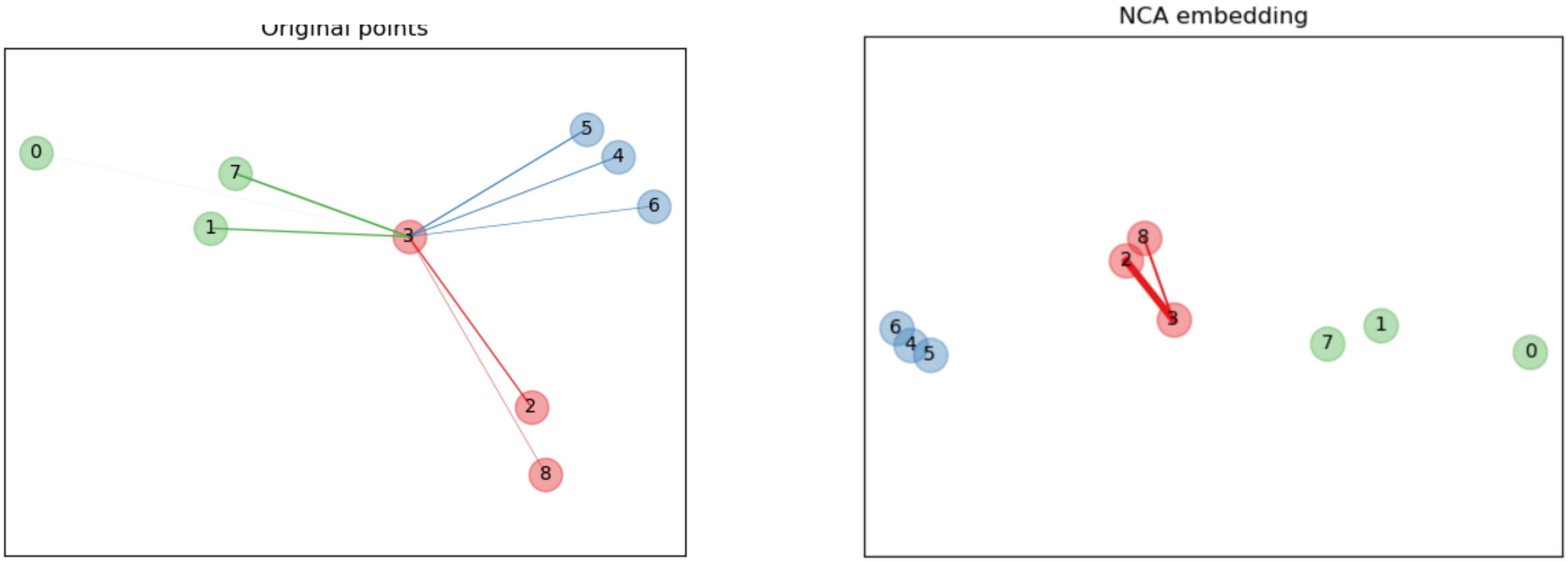


Features: 2

- Reduce the dimension to make the visualization much easy.
- Less feature means saving calculation resource.

[2]

What is the NCA?



- Data category interval becomes larger

[1]

NCA and KNN

Process of KNN

- Calculate the Euclidean distance between sample i and all the remaining samples $d_{ij} = \|x_i - x_j\|_2$
- Select the k samples with the smallest distance
- The predictions are obtained by voting using the labels of these k samples

$$Vote(y_{j_1}, y_{j_2}, \cdots, y_{j_k})$$

How to improve KNN ?

How to make KNN faster ?

How to make KNN much more accurate?

NCA and KNN

From lecture:

- Accuracy：
  - find a more suitable k using cross-validation
  - change the distance function
- Speed：
  - ？？？？

NCA:

- to get a new distance function, that is designed especially for each datasets
- to reduce the dimension of datasets to speed up

How does the NCA work?

In NCA, our aim is to obtain the distance matrix by machine learning

training datasets $\xrightarrow{\text{NCA}}$ Distance matrix

training datasets $\xrightarrow{\text{distance matrix}}$ new training datasets

test datasets $\xrightarrow{\text{distance matrix}}$ new test datasets

1. NCA

2.KNN

How does the NCA work?

the probability $P(x_j)$ of points j being selected according to their distance.

$$P(x_j) \propto k(d_w(x, x_j))$$

where k is an equation on distance. $$k(z) = \exp\left(-\frac{z}{\sigma}\right)$$

Set a new distance function

$$d_w(x_i, x_j) = \sum_{r=1}^{p} w_r^2 |x_{ir} - x_{jr}|,$$

with w_r is the feature weight.

[1-3]

How does the NCA work?

Thus for point i, the probability of selecting point j is

$$p_{ij} = \frac{\exp(-d_w(x_i,x_j))}{\sum_{k\neq i}\exp(d_w(x_i,x_k))}, p_{ii} = 0$$

Set $i = 1 \; if \; y_i = y_j \; esle \; i = 0$, the probability of selecting the right point is

$$p_i = \sum i * \mathrm{p_{ij}}$$

[1-3]

How does the NCA work?

to make the accuracy rate improve, sum all possibilities together

$$f = \sum_{i=1}^{n} p_i$$

Optimization of f by gradient descent

$$\frac{\partial f}{\partial A} = 2A \sum_i \left(p_i \sum_{k \neq i} p_{ik} \left(x_i - x_k\right) \left(x_i - x_k\right)^T - \sum_{j \in C_i} p_{ij} \left(x_i - x_j\right) \left(x_i - x_j\right)^T \right)$$

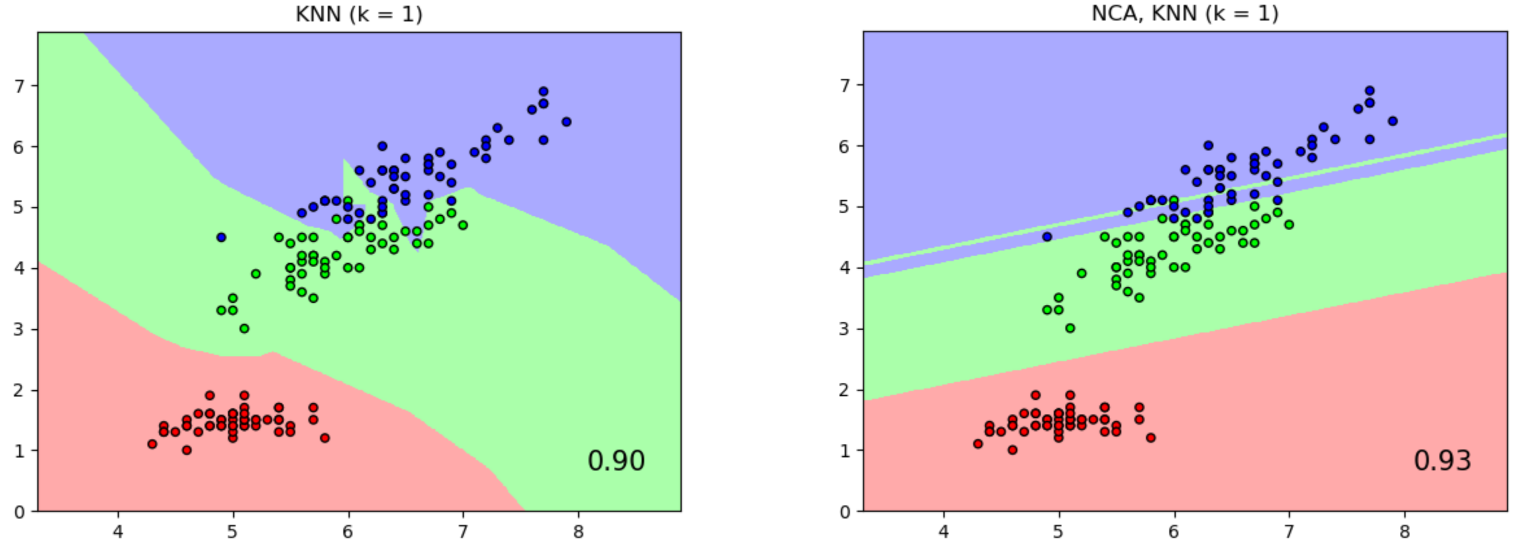
[1-3]

NCA on iris datasets

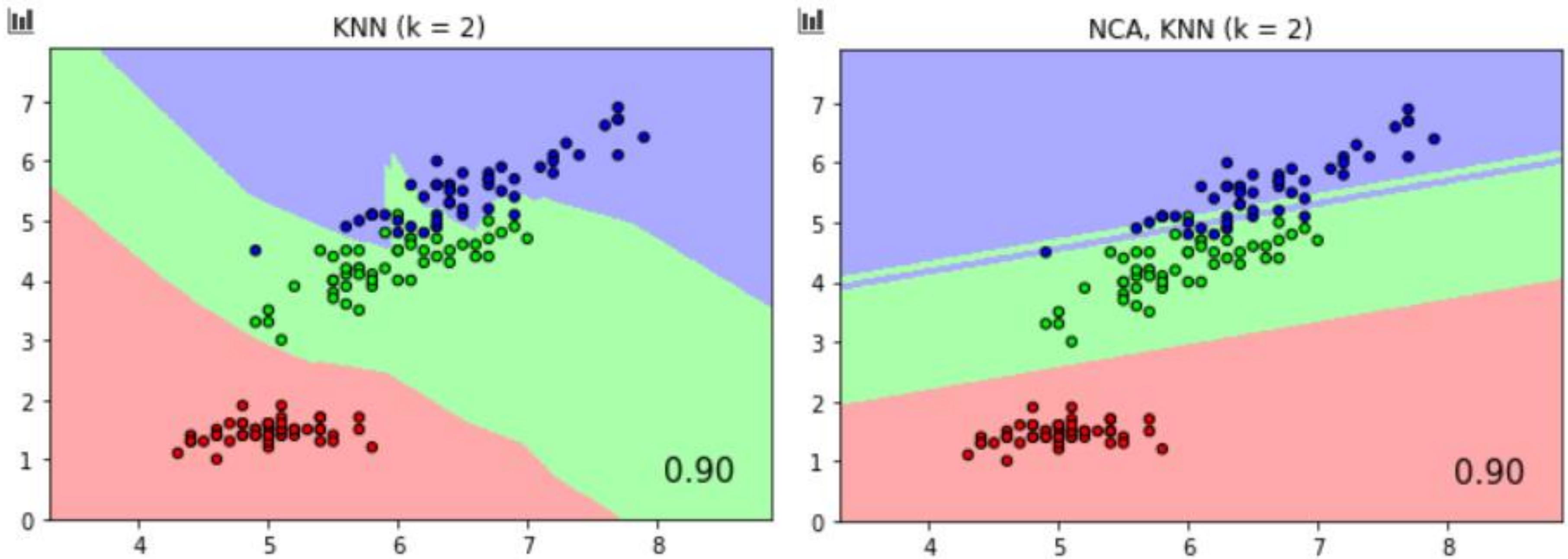
Improvment of KNN performence

```
the score of the KNN on iris datasets is 93.333%
the score of the KNN + NCA on iris datasets without Dimensionality Reduction is 96.19%
the score of the KNN + NCA on iris datasets with Dimensionality Reduction to 3 is 95.238%
the score of the KNN + NCA on iris datasets with Dimensionality Reduction to 2 is 98.095%
```

a new distance function for each datasets



Drawback of NCA

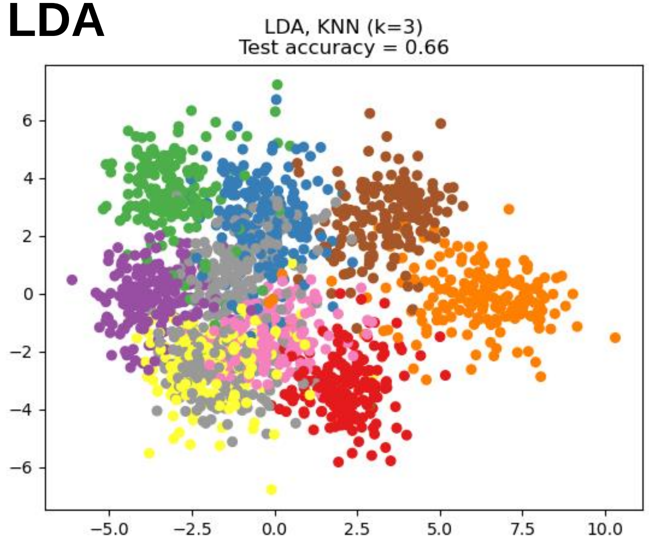


Other dimension reduction algorithms

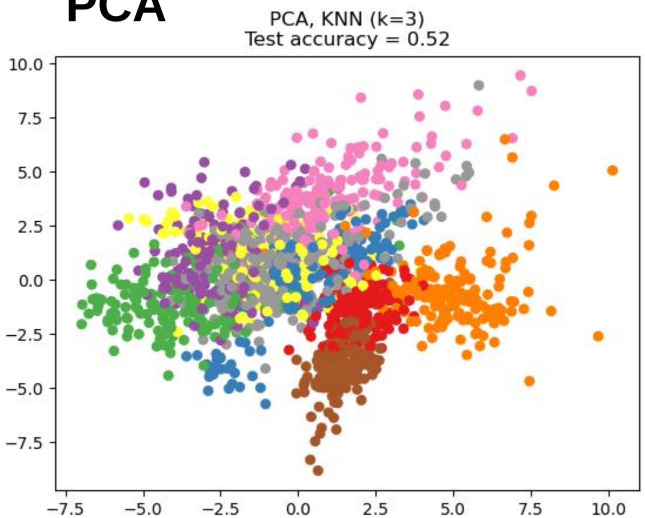
Dataset: MNIST

NCA Neighborhood Components Analysis
PCA Principal Component Analysis
LDA Linear Discriminant Analysis

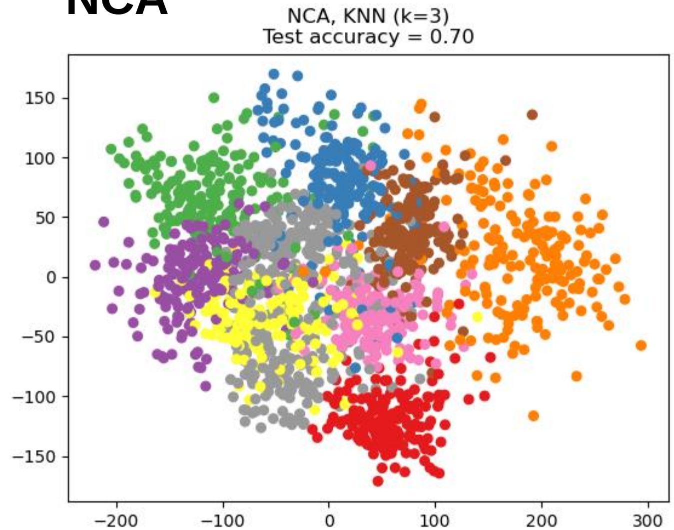
LDA



PCA



NCA



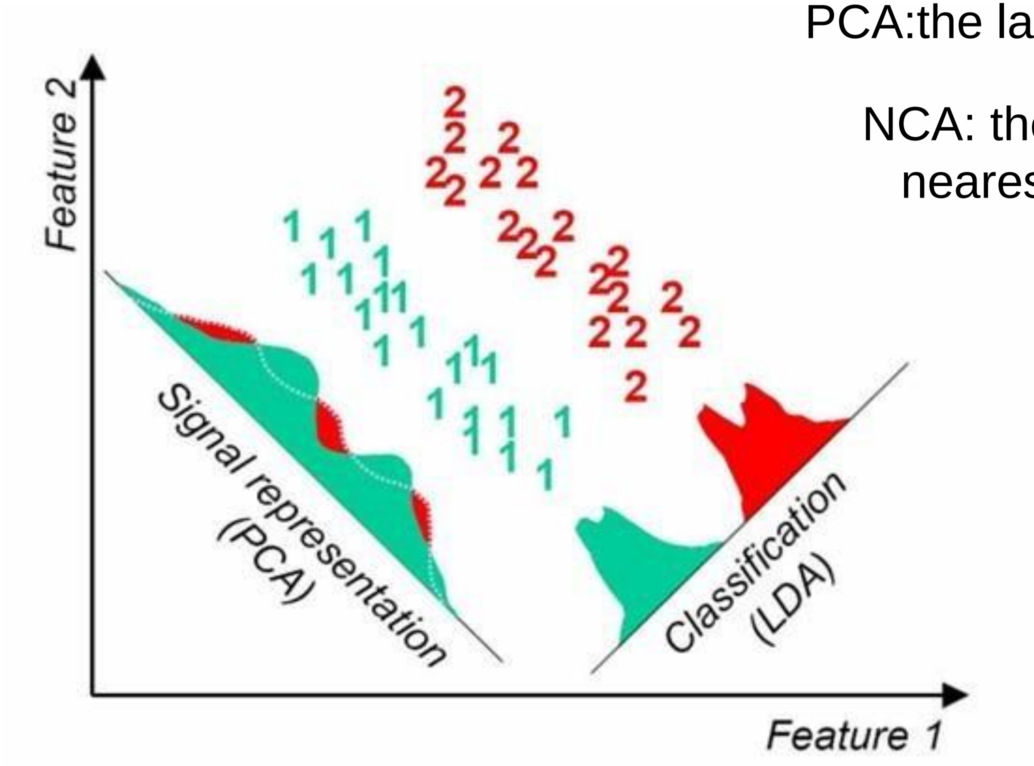
[5]

Other dimension reduction algorithms

LDA: the most variance between classes in projection direction

PCA:the largest variance in projection direction

NCA: the best performance of the stochastic nearest neighbor algorithm in new space



[6]

Other dimension reduction algorithms

Table 1 : compare of the different algorithms

characteristics	PCA	LDA	NCA
Un-/supervised learning	Unsupervised learning	Supervised learning	Supervised learning
Function	Dimensionality reduction	Dimensionality reduction/classification	Dimensionality reduction/classification /regression
Downscaling Limits	no limit	max to k-1	no limit
Data Distribution	Gaussian distribution	Gaussian distribution	no limit

[3,7]

My code

$$\frac{\partial f}{\partial A} = 2A \sum_i \left(p_i \sum_{k \neq i} p_{ik} \left(x_i - x_k\right)\left(x_i - x_k\right)^T - \sum_{j \in C_i} p_{ij} \left(x_i - x_j\right)\left(x_i - x_j\right)^T \right)$$

```
# gradients for-loop
gradients = np.zeros((self.high_dims,
self.high_dims))
# for i
for i in range(self.n_samples):
    k_sum = np.zeros((self.high_dims,
self.high_dims))
    k_same_sum = np.zeros((self.high_dims,
self.high_dims))
    # for k
    for k in range(self.n_samples):
        out_prod = np.outer(X[i] - X[k],
X[i] - X[k])
        k_sum += pij_mat[i][k] * out_prod
        if Y[k] == Y[i]:
            k_same_sum += pij_mat[i][k] *
out_prod
    gradients += pi_row[i] * k_sum -
k_same_sum
gradients = 2 * np.dot(gradients, self.A)
```

[8]

```
# gradient #1 matrix
part_gradients = np.zeros((self.high_dims,
self.high_dims))
for i in range(self.n_samples):
    xik = X[i] - X
    prod_xik = xik[:, :, None] * xik[:,
None, :]
    pij_prod_xik = pij_mat[i][:, None, None]
* prod_xik
    first_part = pi_row[i] *
np.sum(pij_prod_xik, axis=0)
    second_part = np.sum(pij_prod_xik[Y ==
Y[i], :, :], axis=0)
    part_gradients += first_part -
second_part
gradients = 2 * np.dot(part_gradients,
self.A)c
```

[9]

	Loop-style	Vector-style
Runtime	98.824s	9.84363s
Accuracy	0.9619	0.9619

My code

$$\frac{\partial f}{\partial A} = 2A\sum_i \left(p_i \sum_{k\neq i} p_{ik}\left(x_i - x_k\right)\left(x_i - x_k\right)^T - \sum_{j\in C_i} p_{ij}\left(x_i - x_j\right)\left(x_i - x_j\right)^T\right)$$

set $\quad pmask_{ij} = p_{ij}, if\, j \in C_i, else\, 0 \quad W_{ij} = p_i p_{ij} - pmask_{ij}$

$$\frac{\partial f}{\partial A} = 2(XA^T)^T\left(diag\left(sum(W, axis=0)\right) - W - W^T\right)X$$

Benefit :

- Reducing the number of matrix multiplications
- Reducing the size of the matrix

[7]

My code

$$\frac{\partial f}{\partial A} = 2(XA^T)^T \left(diag\left(sum(W, axis=0)\right) - W - W^T\right) X$$

```
# gradients #2
part_gradients = np.zeros((self.high_dims,
self.high_dims))
for i in range(self.n_samples):
    xik = X[i] - X
    prod_xik = xik[:, :, None] * xik[:, None,
:]
    pij_prod_xik = pij_mat[i][:, None, None] *
prod_xik
    first_part = pi_row[i] *
np.sum(pij_prod_xik, axis=0)
    second_part = np.sum(pij_prod_xik[Y ==
Y[i], :, :], axis=0)
    part_gradients += first_part - second_part
gradients = 2 * (part_gradients @ self.A)
```

	Loop-style	Vector-style	W-style	Sk-learn
Runtime	98.824s	9.84363s	6.8411	0.075s
Accuracy	0.9619	0.9619	0.9619	0.9809

[7]

References

- [1] J. Goldberger, G. Hinton, S. Roweis, R. Salakhutdinov. (2005) Neighbourhood Components Analysis. Advances in Neural Information Processing Systems. 17, 513-520, 2005.
- [2] Matlab tutorial - Neighborhood Component Analysis (NCA) Feature Selection
- [3] Data dimension reduction| LDA PCA NCA
- [4] https://scikit-learn.org/stable/auto_examples/neighbors/plot_nca_classification.html#sphx-glr-auto-examples-neighbors-plot-nca-classification-py
- [5] https://scikit-learn.org/stable/auto_examples/neighbors/plot_nca_illustration.html#sphx-glr-auto-examples-neighbors-plot-nca-illustration-py
- [6] https://scikit-learn.org/stable/auto_examples/neighbors/plot_nca_dim_reduction.html#sphx-glr-auto-examples-neighbors-plot-nca-dim-reduction-py
- [7] Wold, Svante, Kim Esbensen, and Paul Geladi. "Principal component analysis." *Chemometrics and intelligent laboratory systems* 2.1-3 (1987): 37-52.
- [8] Neighbourhood Component Analysis
- [9] RoIT / NCA-python