A connection between sparse and low-rank matrices and its application to manifold learning

Stanislav Pyatkin
Danil Ivanov
Danil Gusak

Skoltech

Problem statement

Sparse nonnegative matrix S can be recovered from a real-valued matrix L of **significantly lower rank**. Of particular interest is the setting where the positive elements of S encode the similarities of nearby points on a low dimensional manifold.

$$\mathbf{S} = \max(0, \mathbf{L})$$

$$\begin{bmatrix} 10 & 0 & 0 & 0 & 0 & 15 \\ 0 & 13 & 0 & 16 & 24 & 0 \\ 0 & 0 & 34 & 0 & 24 & 0 \\ 0 & 16 & 0 & 29 & 16 & 0 \\ 0 & 24 & 24 & 16 & 64 & 0 \\ 15 & 0 & 0 & 0 & 0 & 25 \end{bmatrix} \qquad \begin{bmatrix} 10 & -11 & -4 & -11 & -24 & 15 \\ -11 & 13 & -1 & 16 & 24 & -15 \\ -4 & -1 & 34 & -19 & 24 & -15 \\ -11 & 16 & -19 & 29 & 16 & -10 \\ -24 & 24 & 24 & 16 & 64 & -40 \\ 15 & -15 & -15 & -10 & -40 & 25 \end{bmatrix}$$

$\mathrm{rank}(\mathbf{S}) = 6$ $\qquad$ $\mathrm{rank}(\mathbf{L}) = 2$

A geometrical connection between sparse and low-rank matrices and its application to manifold learning

https://openreview.net/forum?id=p8gncJbMit

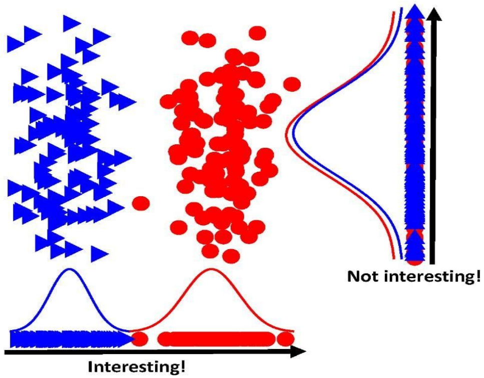
Relevance

$$\mathbf{S} \approx \max(0, \mathbf{L}) \quad (1)$$

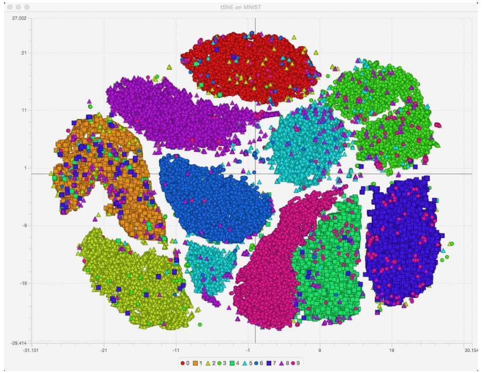
The computation of (1) is interesting as it can be implemented by a neural network with one layer of rectified linear units, so it suggests many possibilities for high dimensional data analysis, especially for data that is naturally represented as a sparse nonnegative matrix.

Available solutions and their limitations

1) PCA: the PCA operates by finding the direction of the largest variance in the data. If there is no effect in projecting in that direction, PCA fails.



2) t-SNE: It is not recommended for use in analysis such as clustering or outlier detection since it does not necessarily preserve densities or distances well.



3) Non-negative matrix factorization (NMF): non-convex, in case the nonnegative rank of V is equal to its actual rank, the problem of finding the NRF of V, if it exists, is known to be NP-hard.

$$\min_{B,C\geq 0} ||A - BC||_F$$

Solution

Let {x1, x2, ..., xn} be a data set of high-dimensional inputs. Our goal is to discover a corresponding set of outputs {y1, y2, ..., yn} that provide a **faithful** **but much lower-dimensional representation** (or embedding).

(i) Magnitudes are preserved: $\|\boldsymbol{y}_i\| = \|\boldsymbol{x}_i\|$ for all i.

(ii) Small angles are preserved: $\cos(\boldsymbol{y}_i, \boldsymbol{y}_j) = \cos(\boldsymbol{x}_i, \boldsymbol{x}_j)$ whenever $\cos(\boldsymbol{x}_i, \boldsymbol{x}_j) > \tau$.

(iii) Large angles remain large: $\cos(\boldsymbol{y}_i, \boldsymbol{y}_j) \leq \tau$ whenever $\cos(\boldsymbol{x}_i, \boldsymbol{x}_j) \leq \tau$.

$$\boxed{\max(0, \boldsymbol{x}_i \cdot \boldsymbol{x}_j - \tau\|\boldsymbol{x}_i\|\|\boldsymbol{x}_j\|) = \max(0, \boldsymbol{y}_i \cdot \boldsymbol{y}_j - \tau\|\boldsymbol{y}_i\|\|\boldsymbol{y}_j\|).}$$

We implement an algorithm for the problem based on the nonlinear low-rank decomposition of sparse matrices.

Algorithm: Sparse similarity matching

Step 1. Compute the sparse matrix S.

$$S_{ij} = \max(0, \boldsymbol{x}_i \cdot \boldsymbol{x}_j - \kappa \|\boldsymbol{x}_i\| \|\boldsymbol{x}_j\|)$$

Step 2. Compute the low-rank matrix L.

Two updates - one for matrix L, and another for an auxiliary matrix Z of the same size. The updates attempt to solve the constrained optimization.

$$\min_{\mathbf{L},\mathbf{Z}} \|\mathbf{L} - \mathbf{Z}\|_F^2 \quad \text{such that} \quad \begin{cases} d = \operatorname{rank}(\mathbf{L}), \\ \mathbf{S} = \max(0, \mathbf{Z}) \end{cases}$$

L is fixed:

$$Z_{ij} = \begin{cases} S_{ij} & \text{if } S_{ij} > 0, \\ \min(0, L_{ij}) & \text{otherwise.} \end{cases}$$

Z is fixed:

$$\mathbf{L} = \arg\min_{\mathbf{M}} \|\mathbf{M} - \mathbf{Z}\|_F^2 \quad \text{such that} \quad \operatorname{rank}(\mathbf{M}) = d.$$

Algorithm: Sparse similarity matching

Step 3. Compute the embedding.

Recover the outputs y_i from the known elements of L.

$$L_{ij} = \boldsymbol{y}_i \cdot \boldsymbol{y}_j - \kappa \|\boldsymbol{y}_i\| \|\boldsymbol{y}_j\|$$

$$G_{ij} = L_{ij} + \frac{\tau}{1-\tau} \sqrt{L_{ii} L_{jj}}$$

{λ_α} - eigenvalues of the matrix G, and {$\mathbf{e}_\alpha$} – corres-ponding eigenvectors

$$y_{\alpha i} = \sqrt{\max(0, \lambda_\alpha)}\, e_{i\alpha}$$

Algorithm: Locally linear extension

Step 1. Compute a sparse matrix W of reconstruction weights.

For each input we compute a set of k weights that reconstruct the input from its k-nearest neighbors. Let Ki denote the set of k-nearest neighbors for input x_i . Weights can be computed by minimizing the regularized sum of reconstruction errors:

$$E_{\mathcal{X}}(\mathbf{W}) = \left\| \boldsymbol{x}_i - \sum_{j \in \mathcal{K}_i} W_{ij} \boldsymbol{x}_j \right\|^2 + \varepsilon \sum_{j \in \mathcal{K}_i} \|\boldsymbol{x}_j\|^2 W_{ij}^2$$

Step 2. Extend the embedding.

The embedding is extended by assuming that nearby outputs should be linearly related in the same way as nearby inputs.

$$E_{\mathbf{W}}(\mathcal{Y}) = \left\| \boldsymbol{y}_i - \sum_{j} W_{ij} \boldsymbol{y}_j \right\|^2 \longrightarrow$$ least-squares problems for the coordinates of unknown outputs

As W is sparse, these problems can be solved very efficiently using preconditioned conjugate gradients methods.

Summary

Algorithm #1: Sparse similarity matching

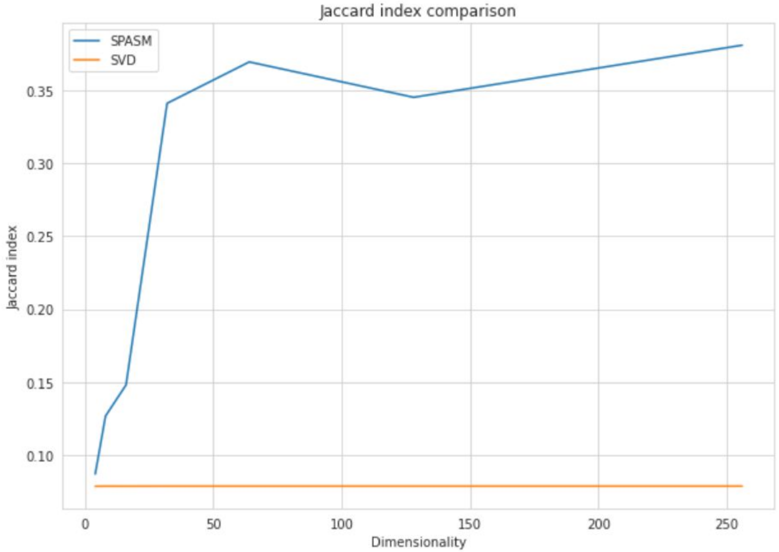
Main idea: given this sparse matrix S, the model's embeddings are obtained by solving a sequence of eigenvalue problems for the corresponding low-rank matrix L.

Algorithm #2: Locally linear extension

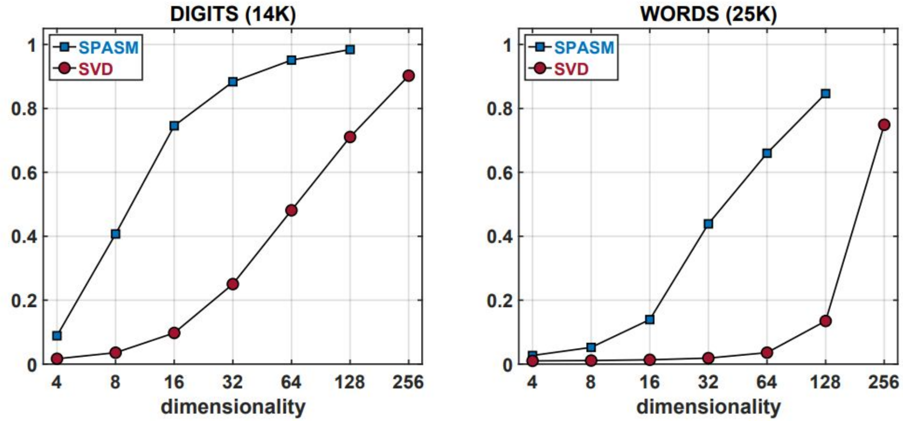
Main idea: extend the aforementioned model to larger data sets where it is too expensive to construct and solve eigenvalue problems directly. This extension requires only the solution of a sparse least-squares problem.

In both of these algorithms we apply the model to images of handwritten digits.

Results: Sparse similarity matching (algorithm #1)



JACCARD INDEX



$$J(\Omega_S, \Omega_L) = \frac{|\,\Omega_S \cap \Omega_L\,|}{|\,\Omega_S \cup \Omega_L\,|}$$

$$\Omega_L = \{(i,j)\,|\,L_{ij} > 0\}$$
$$\Omega_S = \{(i,j)\,|\,S_{ij} > 0\}$$

Skoltech

Implementation: Locally linear extension (algorithm #2)

$$\mathbf{I} - \mathbf{W} = \begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{bmatrix}$$

```python
# Taken from https://github.com/JAVI897/LLE-and-its-variants/blob/master/LLE/example%20LLE.ipynb
# with two corrections: 1) No weight normalization 2) cosine distances instead of the ordinary one
from sklearn.metrics import pairwise_distances
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
import scipy as sp
import sklearn as skl
from sklearn.metrics.pairwise import cosine_distances

class LLE:
    def __init__(self, X, k_n, dim, reg = 1e-03, verbose = False, sparsity = False):
        """
        LLE object
        Parameters
```

$$\left[\mathbf{A}^\top \mathbf{A} + \mathbf{C}^\top \mathbf{C}\right] \mathbf{Y}_{\mathrm{u}}^\top = -\left[\mathbf{A}^\top \mathbf{B} + \mathbf{C}^\top \mathbf{D}\right] \mathbf{Y}_{\mathrm{k}}^\top$$

```python
def PCG(y,A,B,C,D,om=1):
  CTAC = A.T@A + C.T@C
  # M = 1/om*(np.eye(A.shape[0]) * np.diag(
  M = np.eye(A.shape[0])
  f = np.random.randn(C.shape[1])
  r = -(A.T@B + C.T@D) @ y - CTAC @ f
  z = np.linalg.inv(M) @ r
  p = r
```

Main Results

- We explored a geometrical connection between sparse and low-rank matrices.
- The algorithms for Sparse similarity matching and Locally linear extension were implemented from scratch.
- On MNIST dataset the algorithm discovers much lower dimensional representations which preserve meanings.