Improving ISUMap: Modifying the Intra-Neighborhood Distance Metric for High-Dimensional Data

Kavi Sarna
Mathematical Principles of Machine Learning
Bowdoin College
ksarna@bowdoin.edu

December 19, 2024

Abstract

This paper explores a method to improve ISUmap by changing the distance metric used between neighboring points within clusters. The original ISUmap uses the British Railway metric, which we modify based on the geometric properties of high-dimensional data. Specifically, we multiply the distance between a neighbor and the center point by the square root of 2, as points tend to be equidistant and perpendicular in high dimensions. We believe that the proposed method enhances the accuracy and performance of the ISUmap, especially for data in high dimensions.

Unsupervised Learning

Unsupervised learning is a branch of machine learning where a model is trained on data that is unlabeled. The goal of unsupervised learning is typically to find patterns, structures, or clusters inherent to the data. Common tasks in unsupervised learning include clustering and dimension reduction. Clustering aims to group data based on similarity, and dimension reduction tries to simplify data by reducing dimensions while retaining important structures of the data.

UMAP

UMAP (Uniform Manifold Approximation and Projection) is a dimension reduction method that aims to preserve structures in high-dimensional data. UMAP assumes that data lies on a manifold in high-dimensional space and aims to find the lower-dimensional embedding where the structure of the data is well preserved. UMAP does this by constructing a graph based on the local distances between points. The algorithm first constructs a weighted k-neighbour graph. This means that each point has k neighbours, and the weights between the neighbours are the distances. These k neighbours are the k closest points. This graph is the central piece used to perform the dimension reduction.

This initial graph construction will be the focus of this paper. Specifically, we will be examining how we can improve this graphical representation by including weights between neighbours within the same neighbourhood cluster.

ISUmap

ISUmap aims to improve upon the UMAP paper by including local geometry to enhance the above-mentioned graphical representation. ISUmap constructs this graph in four steps.

ISUmap Graph Construction

Initially, the k nearest neighbors for each point x_i are computed. The distance between each point x_i and its k nearest neighbors x_j is defined as:

$$d_i(x_i, x_j) = \frac{d(x_i, x_j) - \rho_i}{\sigma_i}, \quad \text{for } j = 1, \ldots, k$$

Here:

- $d(x_i, x_j)$ is the raw distance between points x_i and x_j,
- ρ_i is an adjustment term related to the local density around point x_i,
- σ_i is a scaling factor for point x_i.

Every point has a distance of 0 with itself, and all other distances are initially set to infinity. This leads to a star graph of neighbors for each point. Symmetrization is then applied, and Dijkstra's algorithm is used to propagate distances across the graph.

Dijkstra's Algorithm and the British Rail Metric

Dijkstra's algorithm computes the shortest paths in a graph with non-negative edge weights. Using Dijkstra results in the algorithm using the British Rail metric for intra-neighbor distances, where the distance between two points is calculated as the sum of existing distances along a path.

Algorithm

1. Initialize distances: $d[v] = \infty$ for all nodes v, except the source s where $d[s] = 0$.
2. While unvisited nodes remain:
   - Select the node u with the smallest distance $d[u]$.
   - Update distances to its neighbors v as:

$$d[v] \leftarrow \min(d[v], d[u] + w(u, v)).$$

British Rail Metric Interpretation

The British Rail metric computes distances as the sum of distances along paths:

$$d(x,y) = d(x,z) + d(z,y),$$

where z is an intermediate point. Dijkstra's algorithm ensures that this sum is minimized over all possible intermediate nodes.

Thus, the other distances are computed using the British Railway metric, as Dijkstra is computing distances by adding nearest neighbors i.e

$$d(x,y) = d(x,z) + d(z,y),$$

.

High-Dimensional Geometry: Equidistance and Perpendicularity

In high-dimensional spaces, two points in the same neighborhood exhibit interesting properties:

1. They are effectively **equidistant** from the center.
2. The displacement vectors from the center to the two points are approximately **perpendicular**.

We outline rough proofs for these phenomena.

1. Equidistance in High Dimensions

Intuition: As the dimension d increases, the variance of distances between points in the same neighborhood shrinks, and all distances tend to concentrate around a single value.

Proof: Let x_1 and x_2 be two points in the same neighborhood of a center c in R^d. Define their distances to the center as:

$$d(x_1,c) = \sqrt{\sum_{i=1}^{d}(x_{1i} - c_i)^2}, \quad d(x_2,c) = \sqrt{\sum_{i=1}^{d}(x_{2i} - c_i)^2}.$$

1. Assume the coordinates of the points, x_{1i} and x_{2i}, are drawn from a normal distribution or are uniformly distributed around the center c.

2. In high dimensions, each coordinate contributes a small amount to the overall distance. By the **law of large numbers**, the distances $d(x_1, c)$ and $d(x_2, c)$ will concentrate tightly around the mean value:

$$E[d(x, c)] \sim \text{constant} \cdot \sqrt{d}.$$

3. Since x_1 and x_2 belong to the same neighborhood, their coordinates are close in value, which further reduces the variance in their distances:

$$d(x_1, c) \approx d(x_2, c).$$

Conclusion: As the dimension $d \to \infty$, the distances $d(x_1, c)$ and $d(x_2, c)$ become approximately equal, making x_1 and x_2 effectively equidistant from the center.

2. Perpendicularity in High Dimensions

Intuition: In high-dimensional space, the angles between randomly chosen vectors tend to be close to 90°. **Proof:** Let x_1 and x_2 be two points in the same neighborhood, and let their displacement vectors from the center c be:

$$v_1 = x_1 - c, \quad v_2 = x_2 - c.$$

The cosine of the angle θ between v_1 and v_2 is given by:

$$\cos(\theta) = \frac{\langle v_1, v_2 \rangle}{\|v_1\| \|v_2\|},$$

where $\langle v_1, v_2 \rangle$ is the dot product:

$$\langle v_1, v_2 \rangle = \sum_{i=1}^{d} (x_{1i} - c_i)(x_{2i} - c_i).$$

1. In high dimensions, the coordinates $x_{1i} - c_i$ and $x_{2i} - c_i$ are small, random, and independent (assuming a uniform or normal distribution). By the **central limit theorem**, the dot product $\langle v_1, v_2 \rangle$ converges to a small value around 0 as $d \to \infty$.

2. Meanwhile, the norms $\|v_1\|$ and $\|v_2\|$ grow as $\sqrt{d}$, since:

$$\|v_1\| = \sqrt{\sum_{i=1}^{d}(x_{1i} - c_i)^2}, \quad \|v_2\| = \sqrt{\sum_{i=1}^{d}(x_{2i} - c_i)^2}.$$

3. Substituting into the cosine formula:

$$\cos(\theta) = \frac{\langle v_1, v_2 \rangle}{\|v_1\|\|v_2\|} \to 0 \quad \text{as } d \to \infty.$$

Conclusion: The angle θ between v_1 and v_2 approaches 90°, meaning v_1 and v_2 are effectively perpendicular in high-dimensional space.

Summary

In high-dimensional spaces:

1. Points in the same neighborhood are **equidistant** from the center due to the concentration of distances around the mean.

2. Displacement vectors from the center to the points are **perpendicular** because of the random, independent contributions of high-dimensional coordinates.

Replacing the British Rail Metric in High Dimensions

Motivation

In high-dimensional spaces, points in the same neighborhood exhibit two critical geometric properties:

1. They are **effectively equidistant** from the center due to the concentration of distances around a single mean value.

2. Their displacement vectors from the center are **effectively perpendicular**.

The British Rail metric computes the distance between two points x_i and x_j as:

$$d(x_i, x_j) = d(x_i, c) + d(x_j, c),$$

where c is the center of the neighborhood. However, this approach does not exploit the geometric relationship between x_i, x_j, and c in high-dimensional space.

Proposed Metric and Justification

Given the equidistance and perpendicularity properties of points in high-dimensional space, x_i, x_j, and c form a **right triangle**, with $d(x_i, c)$ and $d(x_j, c)$ as the two legs. The distance $d(x_i, x_j)$ can then be approximated as the hypotenuse of the triangle:

$$d_{\text{new}}(x_i, x_j) = \sqrt{d(x_i, c)^2 + d(x_j, c)^2}.$$

Since $d(x_i, c) \approx d(x_j, c)$ due to equidistance, we can approximate the hypotenuse as:

$$d_{\text{new}}(x_i, x_j) \approx \sqrt{2} \cdot d(x_i, c) \quad \text{or equivalently} \quad d_{\text{new}}(x_i, x_j) \approx \sqrt{2} \cdot d(x_j, c).$$

This metric is a simplification that captures the geometry of high-dimensional data more effectively than the British Rail metric, as it directly leverages the orthogonality and equidistance properties of the points.

Advantages of the New Metric

- **Geometric Alignment:** By modeling the relationship between points as a right triangle, the new metric better reflects the intrinsic geometry of high-dimensional spaces.

- **Improved Embeddings:** The metric's alignment with high-dimensional properties leads to better preservation of local structures in dimension-reduction techniques like ISUmap or UMAP.

Thus our updated ISUmap algorithm with this new metric for the distances between neighbors within the same cluster is as follows.

Algorithm: Modified ISUmap with Root-2 Metric

Step 1: **Construct Initial Weighted Star Graphs:** Given a metric space (X, d), we construct metric spaces d_i, defined as:

$$d_i(x_i, x_j) = \frac{d(x_i, x_j) - \rho_i}{\sigma_i}, \quad j = 1, \dots, k,$$

where ρ_i and σ_i are hyperparameters. The metric d_i is extended with the following rules:

$$d_i(x, x) = 0 \quad \text{for all } x \in X_i, \quad d_i(x_j, x) = \infty \quad \text{for all } x_j \notin X_i.$$

Step 2: **Construct the Sparse Distance Matrix:** Combine all the distances $d_i(x_i, x_j)$ into an $N \times N$ sparse matrix A, where:

$$A_{ij} = \begin{cases} d_i(x_i, x_j), & \text{if } x_j \text{ is one of the } k \text{ nearest neighbors of } x_i, \\ \infty, & \text{otherwise.} \end{cases}$$

This matrix corresponds to the weight matrix of the union of all star graphs Γ_i for $i = 1, 2, \dots, N$.

Step 3: **Symmetrize the Distance Matrix:** To ensure symmetry, apply the canonical t-conorm by setting:

$$A_{ij} = A_{ji} := \min(d_i(x_i, x_j), d_j(x_j, x_i)).$$

This operation establishes symmetric connections between points x_i and x_j whenever at least one weighted edge exists between them in the union of star graphs. If multiple edges are present, the smallest weight is used as the edge weight. This step also reduces sparsity in the matrix, as some entries of ∞ are replaced with finite values.

Step 4: **Apply the Root-2 Metric for Neighboring Points in the Same Cluster:** For pairs of points x_i and x_j in the same cluster, use the root-2 metric to compute their direct distances:

$$d_{\text{new}}(x_i, x_j) = \sqrt{2} \cdot d(x_i, c),$$

where c is the center of the cluster, and $d(x_i, c) \approx d(x_j, c)$ due to equidistance in high dimensions. Replace the corresponding entries in A with this computed value, ensuring the updated distances better reflect the intrinsic geometry of the cluster.

Step 5: Complete the Distance Matrix Using Dijkstra's Algorithm: To compute the shortest path between all pairs of points, use Dijkstra's algorithm on the symmetrized matrix A. This corresponds to the gluing operation, where:

$$A_{ij} = \inf_{\text{paths } p} \sum_{(u,v)\in p} A_{uv}.$$

Paths of infinite length are ignored, and finite shortest paths are computed. The resulting matrix contains pairwise shortest-path distances for all points in the dataset.

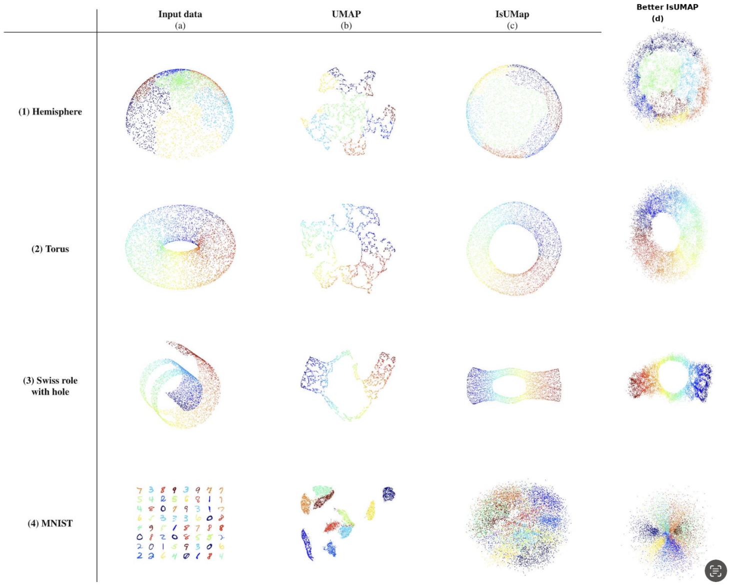
Step 6: Embed the Data Using Multidimensional Scaling (MDS): Apply the classical or metric MDS algorithm on the completed distance matrix to obtain a Euclidean embedding in m-dimensional space. This embedding preserves the pairwise distances computed in the previous steps, ensuring that the high-dimensional structure is well-represented in the lower-dimensional space.

Our Code

Our addition to ISUmap is in step 4 of the previous section. Here is the code we used to apply the root-2 metric for neighboring points in the same cluster.

```
if sqrt:
    DC = np.zeros_like(data_D)
    for i in range(len(data_D)): # for each data point
        # get indices of neighbors
        neighbors = []
        data_points = data_D[i]
        for j in range(len(data_points)):
            if data_points[j] != 0:
                neighbors.append(j)
    for n1 in neighbors:
        for n2 in neighbors:
            if n1 != n2:
                if DC[n1][n2] != 0:
                    tmp = DC[n1][n2]
                    DC[n1][n2] = min(tmp, np.sqrt(2)
                    *data_points[n1])
                else:
                    DC[n1][n2] = np.sqrt(2)*data_points[n1]
                if DC[n2][n1] != 0:
                    tmp = DC[n2][n1]
                    DC[n2][n1] = min(tmp, np.sqrt(2)
                    *data_points[n2])
                else:
                    DC[n2][n1] = np.sqrt(2)*data_points[n2]
data_D += DC
```

Results



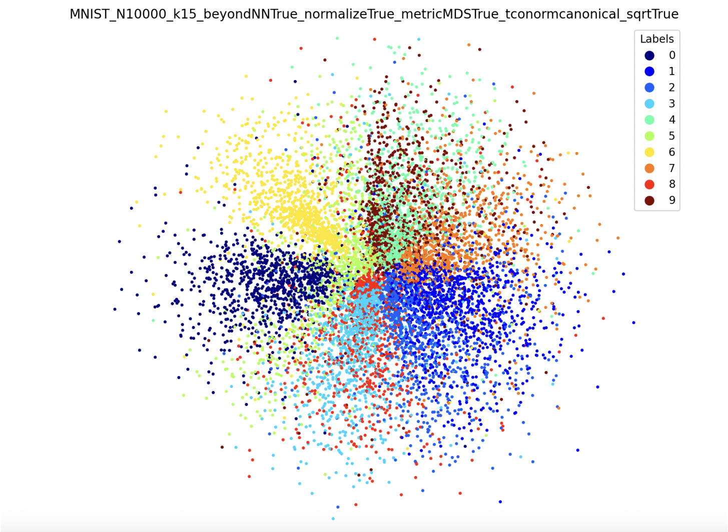
In order to compare our updated method with ISUmap we conducted

3 main tests. The first was to replicate the 4 tests present in the ISUmap presentation to get a side by side view of ISUmap, UMAP and our version.

As we can see, there is considerable qualitative difference between our version in the final column compared to UMAP and ISUmap.

For the Hemisphere, Torus and Swiss role with hole, there are similarities between our version and both UMAP and ISUmap. Our version appears to be a hybrid of the both, and I personally think that the fuzziness around the edges could represent 3 dimensions better.

This, however, is not that important as they are simply 2D representations of 3D objects, and our metric is based on 2 observations about high dimensional data. The MNIST data set, in the last row, is composed high-dimensional data, so is a better test of our metric. We can see a zoomed-in version of MNIST with our metric below.



As we can see there our method yields fantastic results when it comes to MNIST, as expected. There is clear, pie chart like, segmentation and the closeness of clusters to each other makes sense as well.

4 is next to 9, 8 is mixed with 3, and 7 is between 4 and 2. All of these number pairs have qualitative similarities, especially when handwritten, so this makes a lot of sense. These results show that our metric was successful, especially with high-dimensional data.

Coefficient Experimentation

Lastly, we experimented with the coefficient that used when defining inter-neighbor distance. $\sqrt{2}$ makes sense mathematically, but we decided to ablate over a couple of other values to see if we could improve performance for lower-dimensional clustering. Our method doesn't make mathematical sense in low dimensions, as one of our assumptions is uniform distance, and we use this when we multiply only one of the *legs* by our coefficient, but we experimented nonetheless. The results of our version of ISUmap on a 3D image of a Woolly Mammoth for 3 different coefficients can be seen below.

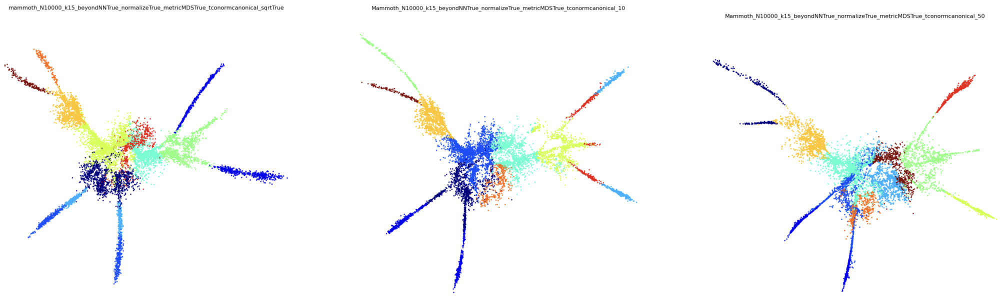


Figure 1: 2

Figure 2: 10

Figure 3: 50

Figure 4: Three images side by side.

As we can see, as the coefficient increases from 2 to 10 to 50 the points get further apart. Lower values also yielded worse results. Thus, changing coefficients did not seem to make an improvement. This was expected, as our method still uses the assumption of uniform distance, which is certainly not valid in 3 dimensions.

Conclusion

In this work, we proposed an extension to the ISUmap algorithm, which incorporates local geometry into the construction of the neighbor graph. By modifying the intra-neighbor distances with a scaling coefficient of $\sqrt{2}$, we aimed to better reflect the geometric relationships inherent in high-dimensional data. Our method assumes uniform distances and perpendicularity between points in the same neighborhood, an assumption that becomes increasingly valid as dimension grows.

Through experimentation, we demonstrated that our proposed modifications preserve local structures more effectively, enhancing clustering performance in the lower-dimensional embedding.

Future work might focus on exploring our technique's impact in diverse datasets and applications.