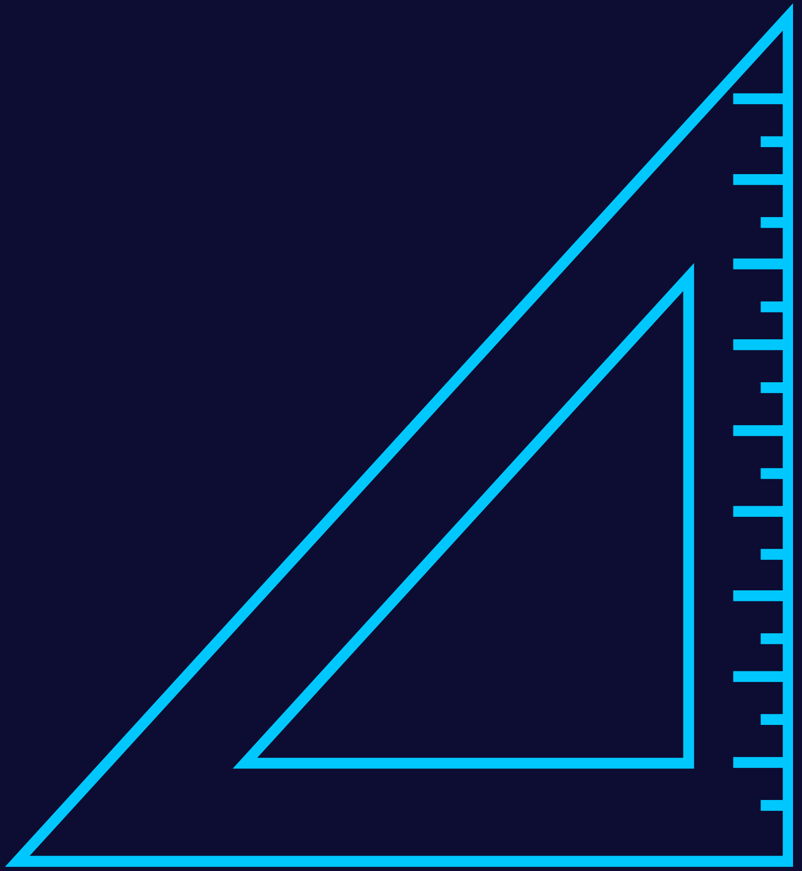


5th Week
The Bridge

Manifold distances
02/02 - 02/09

Josefa Fernández

ToDo Meeting

- ✓ Move Data to new folder
- ✓ Distribution between manifold
  - Dimensionality estimation
    - Search for a dimensionality estimation - better if it is for face recognition
    - Compare to this found method to the PCA method
  - ✓ A single value for intrinsic dimension
  - Filter number of img K, whose intrinsic dimension it is K-1
  - Between same intrinsic dimension, compare angle the distances
    - orthogonal?
    - distribution angles close to 1 and if they are really small WHY?
    - angle between subpaces

ToDo Meeting

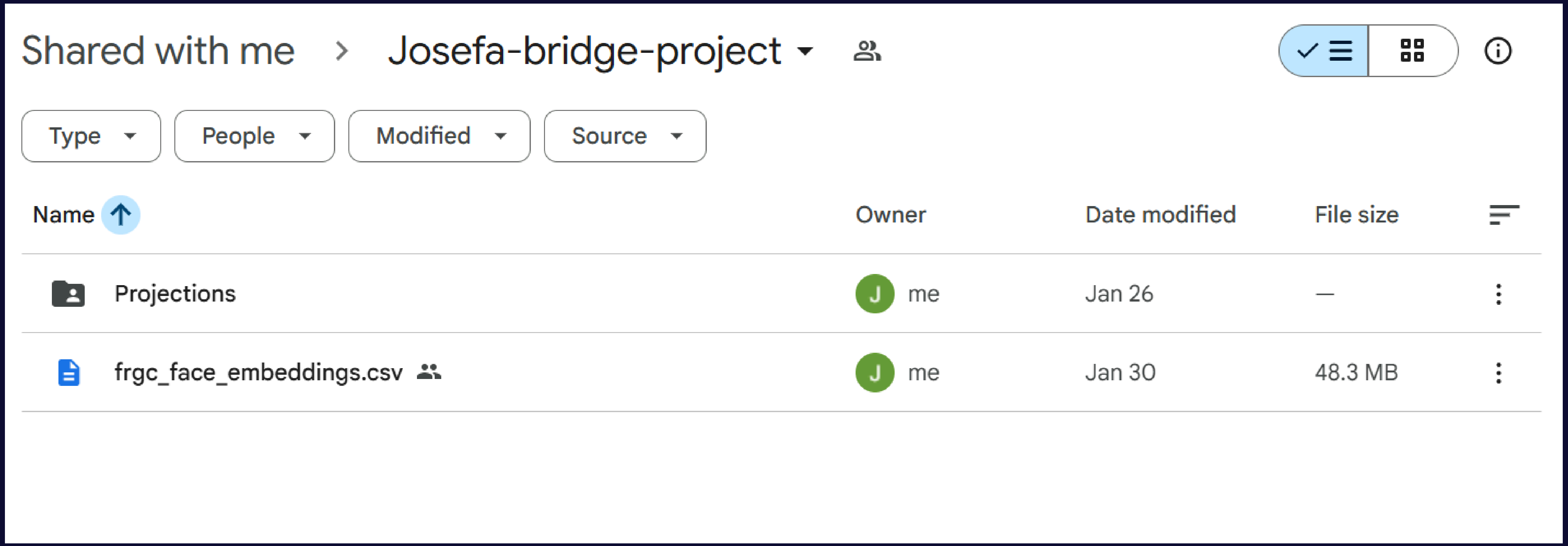
- Distribution of embeddings
  - Geometry structure
- Representation proximity
- Compare to face networks
  - ArcFace
  - MagFace
  - AdaFace
  - Facenet
  - Dlib

Main Info

- **ENTER TERMINAL:** ssh jferna27@cvrl-flynn-ws1.cse.nd.edu
- **GITHUB:** https://github.com/J0SEF4/The-Bridge/tree/main
- **DATA:** https://drive.google.com/drive/folders/1WSC-posibf0VGm1CYAcUhi8m5KqJyNWJ?usp=sharing

Save Data in a new folder

https://drive.google.com/drive/folders/1WSC-posibf0VGm1CYAcUhi8m5KqJyNWJ



Shared with me > Josefa-bridge-project

Name	Owner	Date modified	File size
Projections	me	Jan 26	—
frgc_face_embeddings.csv	me	Jan 30	48.3 MB

Distribution between Manifolds

Papers for dimensionality estimation

Paper

On the Intrinsic Dimensionality of Image Representations - Sixue Gong, et al.

- https://openaccess.thecvf.com/content_CVPR_2019/papers/Gong_On_the_Intrinsic_Dimensionality_of_Image_Representations_CVPR_2019_paper.pdf

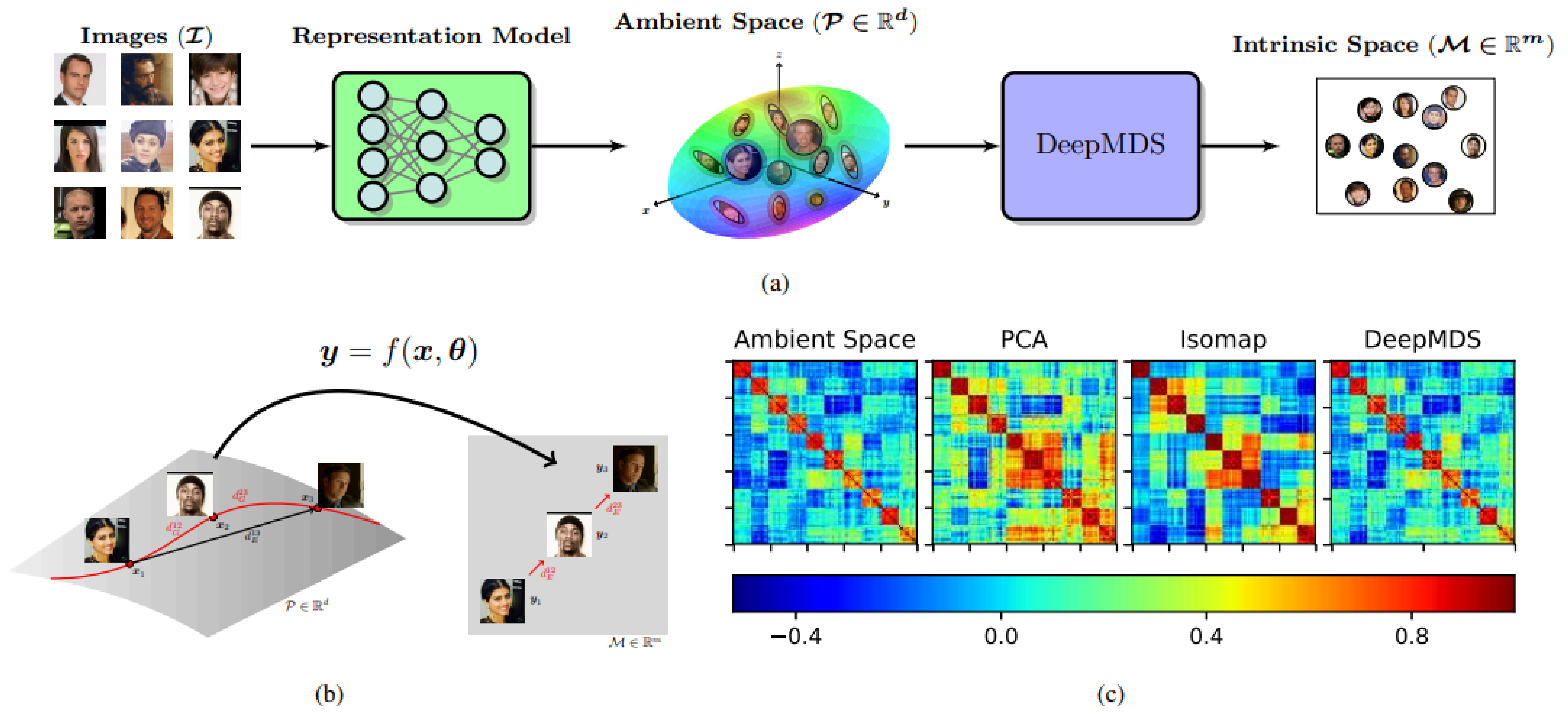
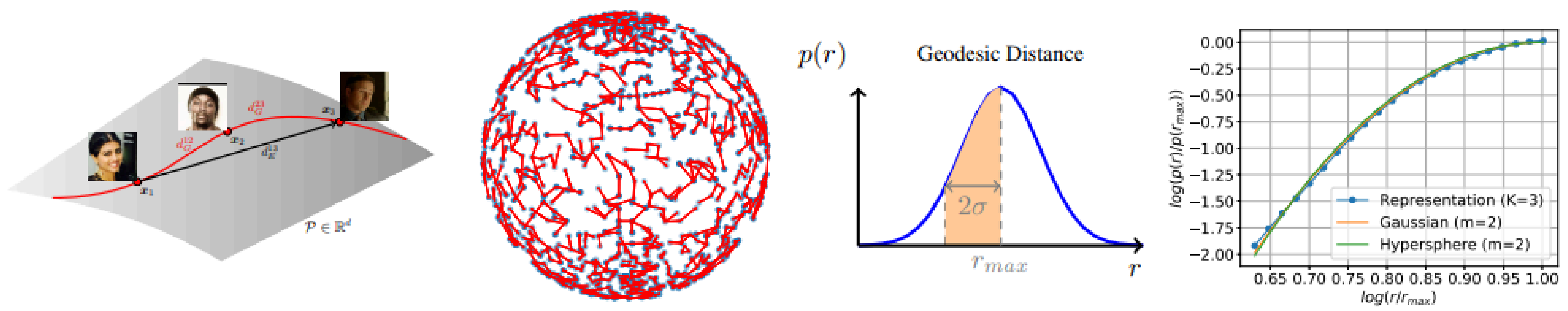

Images ($\mathcal{I}$)
Representation Model
Ambient Space ($\mathcal{P} \in \mathbb{R}^d$)
DeepMDS
Intrinsic Space ($\mathcal{M} \in \mathbb{R}^m$)
(a)
$y = f(x, \theta)$
$\mathcal{P} \in \mathbb{R}^d$
$\mathcal{M} \in \mathbb{R}^m$
(b)
Ambient Space
PCA
Isomap
DeepMDS
−0.4
0.0
0.4
0.8
(c)

Figure 1: **Overview:** This paper studies the manifold of feature vectors of images $\mathcal{I}$ obtained from a given representation model. (a) We estimate the intrinsic dimensionality (ID) of the ambient space $\mathcal{P}$ and propose DeepMDS, an unsupervised method, to map $\mathcal{P}$ to a low-dimensional intrinsic space $\mathcal{M}$. (b) Illustration of the ambient space $\mathcal{P}$ and intrinsic manifold $\mathcal{M}$ of a face representation. Here, while the ambient and linear dimension of the representation is three, its ID is only two. (b) Heatmaps of similarity scores between face pairs of 10 classes with 10 images per class for a representation with ID of 10-dim. The similarity is computed in four different spaces, the 512-dim ambient space $\mathcal{P}$, 10-dim space of linear dimensionality (PCA), 10-dim intrinsic space $\mathcal{M}$ estimated by Isomap [40] and by our DeepMDS model. The class separability, as shown by the diagonal blocks, is better maintained by DeepMDS.

Paper

On the Intrinsic Dimensionality of Image Representations - Sixue Gong, et al.

- https://openaccess.thecvf.com/content_CVPR_2019/papers/Gong_On_the_Intrinsic_Dimensionality_of_Image_Representations_CVPR_2019_paper.pdf



(a) Graph Induced Geodesic Distance

(b) Topological Similarity

Figure 2: **Intrinsic Dimension:** Our approach is based on two observations: (a) Graph induced geodesic distance between images is able to capture the topology of the image representation manifold more reliably. As an illustration, we show the graph edges for the surface of a unitary hypersphere and a face manifold of ID two, embedded within a 3-dim space. (b) The distribution of the geodesic distances (for distance $r_{max} - 2\sigma \leq r \leq r_{max}$, where r_{max} is the distance at the mode) has been empirically observed [13] to be similar across different topological structures with the same intrinsic dimensionality. The plot shows the distance distribution for a face representation, unitary hypersphere and a Gaussian distribution of ID two embedded within 3-dim space.

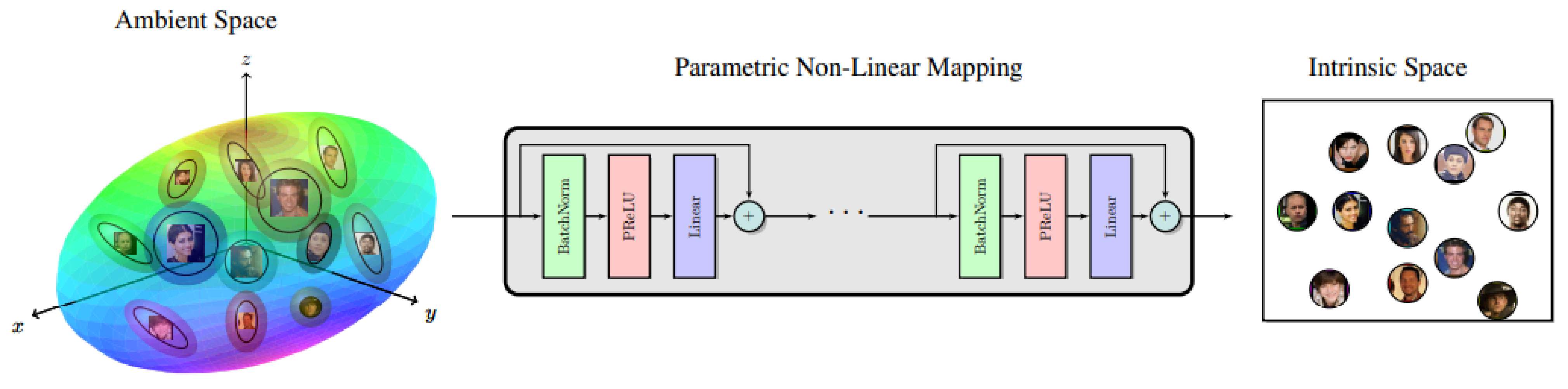


Figure 3: **DeepMDS Mapping:** A DNN based non-linear mapping is learned to transform the ambient space to a plausible intrinsic space. The network is optimized to preserve distances between pairs of points in the ambient and intrinsic space.

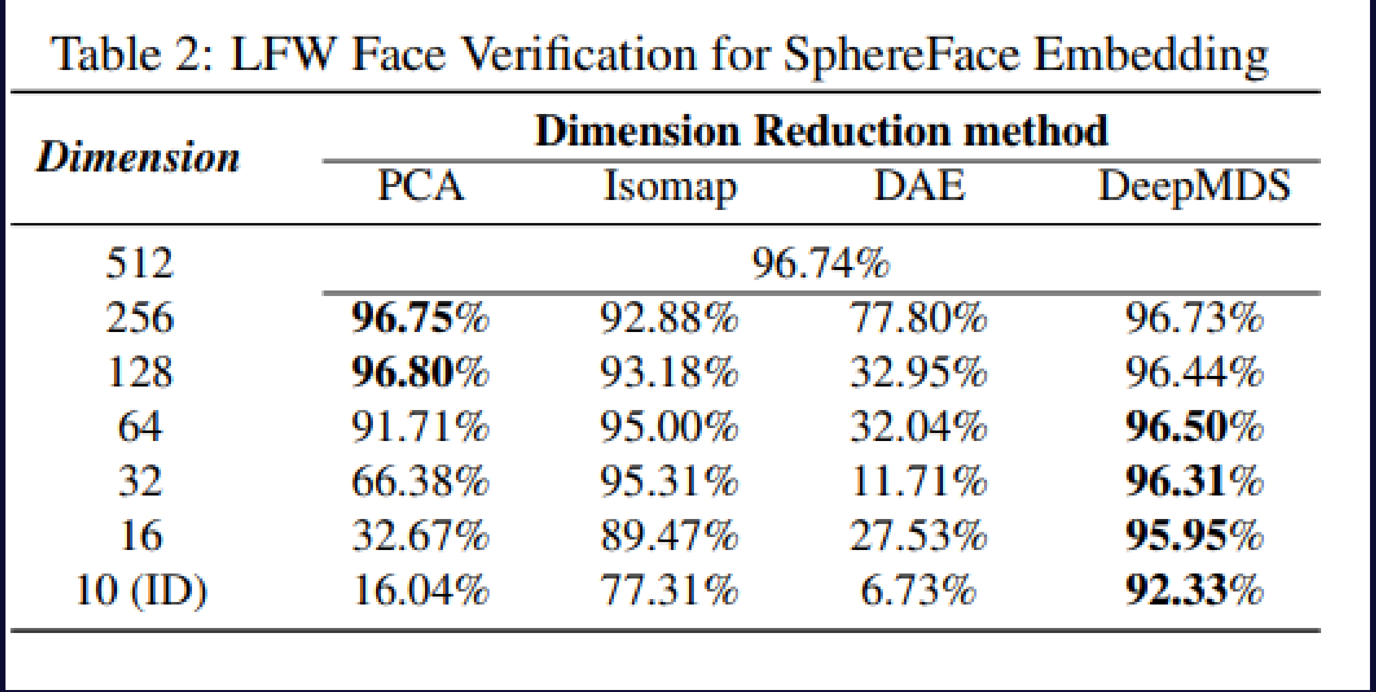


Table 2: LFW Face Verification for SphereFace Embedding

Dimension	**Dimension Reduction method**			
	PCA	Isomap	DAE	DeepMDS
512	96.74%			
256	**96.75%**	92.88%	77.80%	96.73%
128	**96.80%**	93.18%	32.95%	96.44%
64	91.71%	95.00%	32.04%	**96.50%**
32	66.38%	95.31%	11.71%	**96.31%**
16	32.67%	89.47%	27.53%	**95.95%**
10 (ID)	16.04%	77.31%	6.73%	**92.33%**

Paper

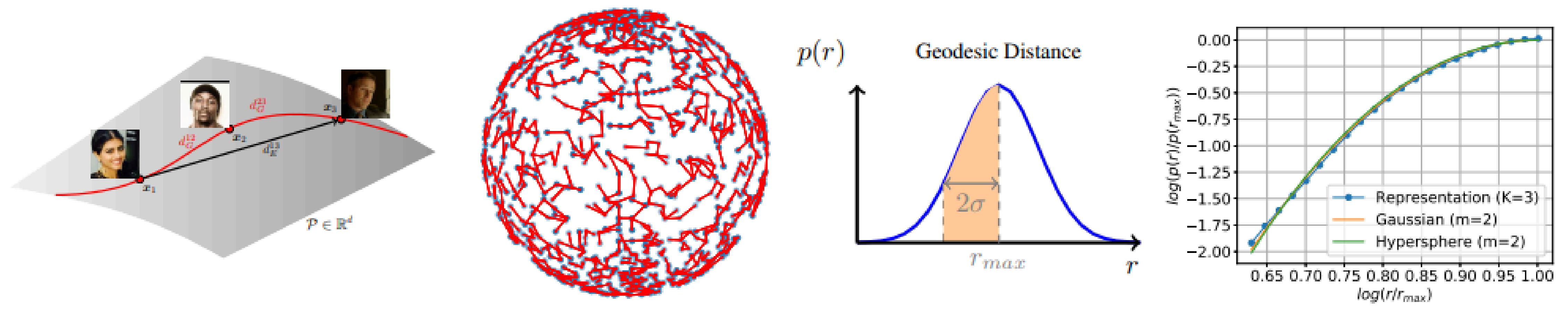
“The intrinsic dimensionality (ID) refers to the dimensionality of the m-dimensional manifold M embedded within the d-dimensional ambient (representation) space P where <=d”

1. **Global Notation**

- Ambient space P (d = dimensional space of embeddings) $\mathcal{P} \in \mathbb{R}^d$

- Normalized ArcFace Embeddings

- ArcFace Angular Distance



(a) Graph Induced Geodesic Distance

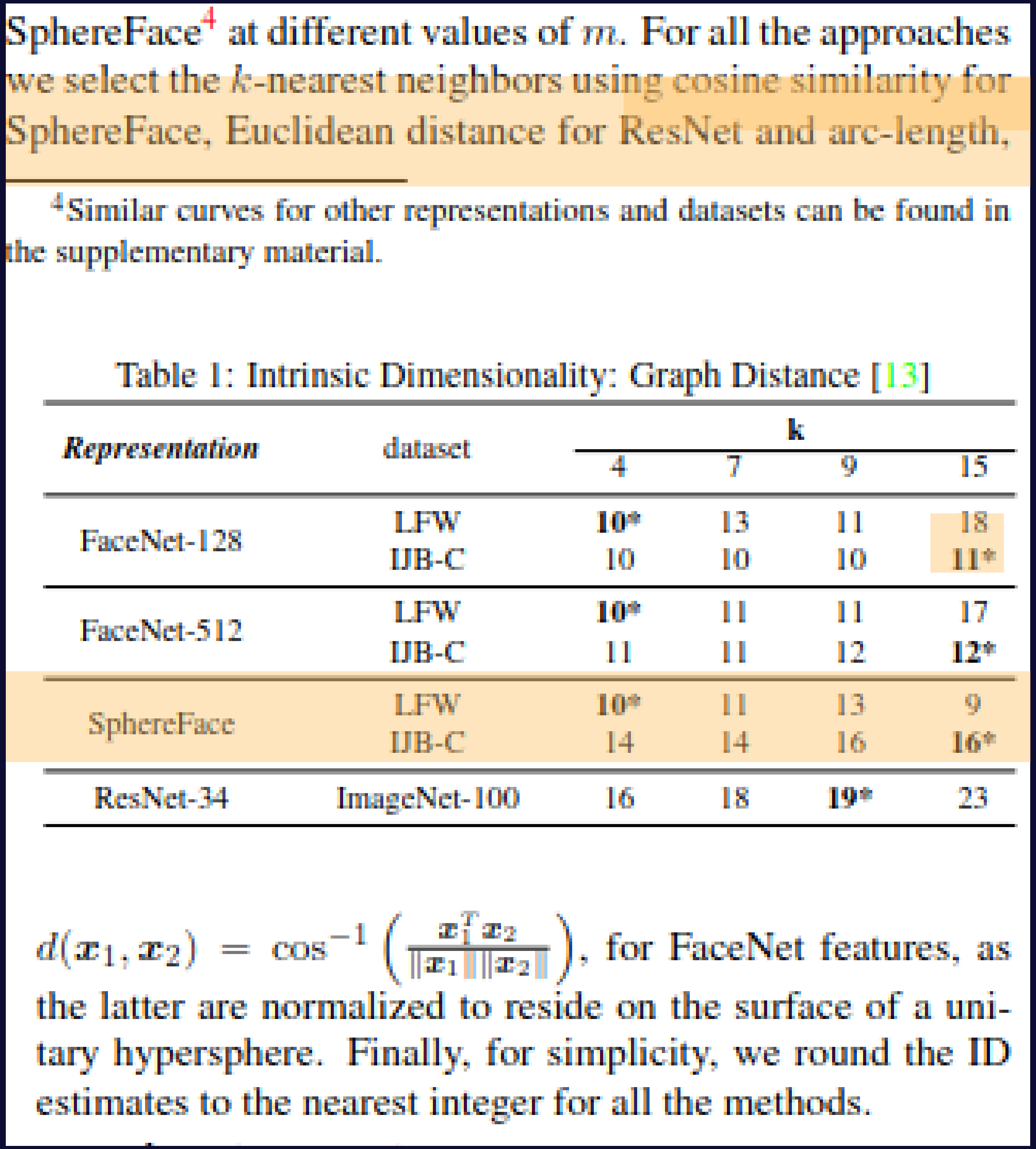
(b) Topological Similarity

Figure 2: **Intrinsic Dimension:** Our approach is based on two observations: (a) Graph induced geodesic distance between images is able to capture the topology of the image representation manifold more reliably. As an illustration, we show the graph edges for the surface of a unitary hypersphere and a face manifold of ID two, embedded within a 3-dim space. (b) The distribution of the geodesic distances (for distance $r_{max} - 2\sigma \leq r \leq r_{max}$, where r_{max} is the distance at the mode) has been empirically observed [13] to be similar across different topological structures with the same intrinsic dimensionality. The plot shows the distance distribution for a face representation, unitary hypersphere and a Gaussian distribution of ID two embedded within 3-dim space.

Paper

Intrinsic Dimensionality

3.1. Estimating Intrinsic Dimension = m



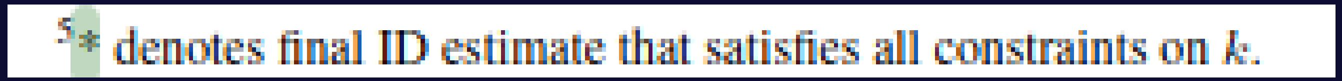
SphereFace[4] at different values of m. For all the approaches we select the k-nearest neighbors using cosine similarity for SphereFace, Euclidean distance for ResNet and arc-length,

[4]Similar curves for other representations and datasets can be found in the supplementary material.

Table 1: Intrinsic Dimensionality: Graph Distance [13]

Representation	dataset	k: 4	7	9	15
FaceNet-128	LFW	**10***	13	11	18
	IJB-C	10	10	10	**11***
FaceNet-512	LFW	**10***	11	11	17
	IJB-C	11	11	12	**12***
SphereFace	LFW	**10***	11	13	9
	IJB-C	14	14	16	**16***
ResNet-34	ImageNet-100	16	18	**19***	23

$d(\boldsymbol{x}_1, \boldsymbol{x}_2) = \cos^{-1}\left(\frac{\boldsymbol{x}_1^T \boldsymbol{x}_2}{\|\boldsymbol{x}_1\|\|\boldsymbol{x}_2\|}\right)$, for FaceNet features, as the latter are normalized to reside on the surface of a unitary hypersphere. Finally, for simplicity, we round the ID estimates to the nearest integer for all the methods.



[5]* denotes final ID estimate that satisfies all constraints on k.

- **ArcFace live in a hypersphere using normalized embeddings**
  - SphereFace → use cosine similarity
  - ArcFace similar SphereFace
    - Same geometry embedding =
      - L2 normalized (embedding live in a hypersphere)
      - angular discrimination (additive/multiplicative)

- k > m
- k << N

- X image representation manifold = embeddings
- k size of the neighborhood, here k=15 best

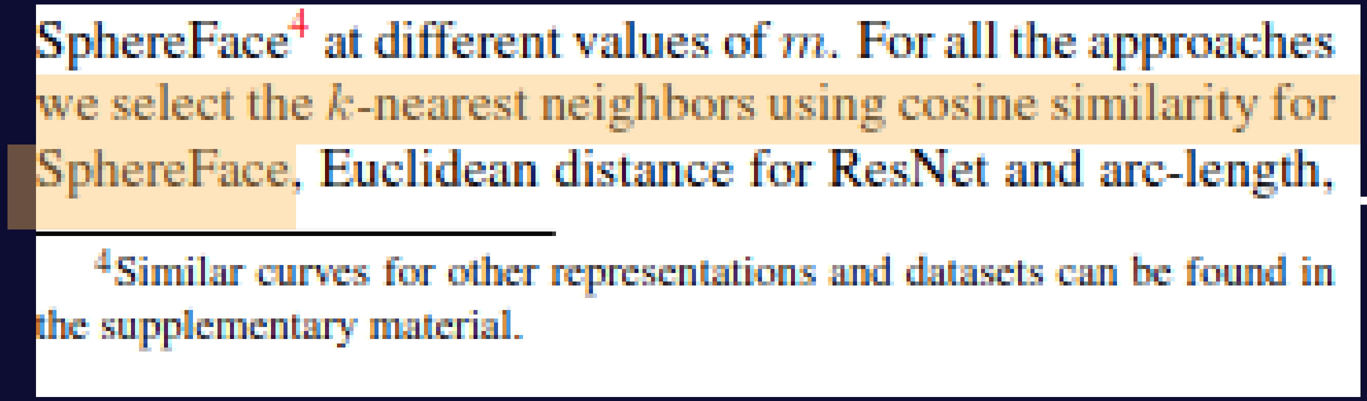
- * IJB-C more noise and variation than LFW

Paper

Intrinsic Dimensionality

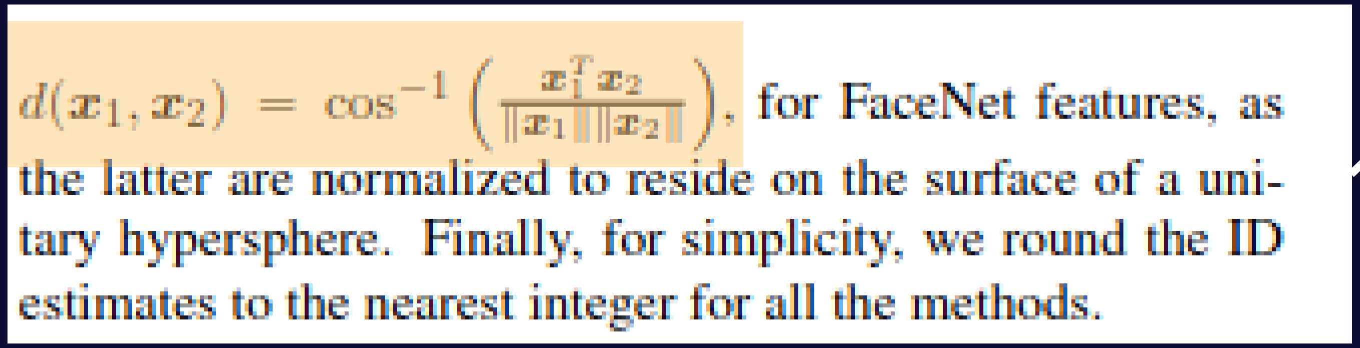
3.1. Estimating Intrinsic Dimension =m

- L2 Normalization



SphereFace[4] at different values of m. For all the approaches we select the k-nearest neighbors using cosine similarity for SphereFace, Euclidean distance for ResNet and arc-length,

[4] Similar curves for other representations and datasets can be found in the supplementary material.



$d(\boldsymbol{x}_1, \boldsymbol{x}_2) = \cos^{-1}\left(\frac{\boldsymbol{x}_1^T \boldsymbol{x}_2}{\|\boldsymbol{x}_1\|\|\boldsymbol{x}_2\|}\right)$, for FaceNet features, as the latter are normalized to reside on the surface of a unitary hypersphere. Finally, for simplicity, we round the ID estimates to the nearest integer for all the methods.

- Each embedding xii → xii_norm = x_i / ||x_i||
  - each vector norm 1
  - live on hypersphere
  - Cosine similarity is a angular distance in a sphere
  - geodesic distance in the manifold that is an hypersphere to
  - ArcFace

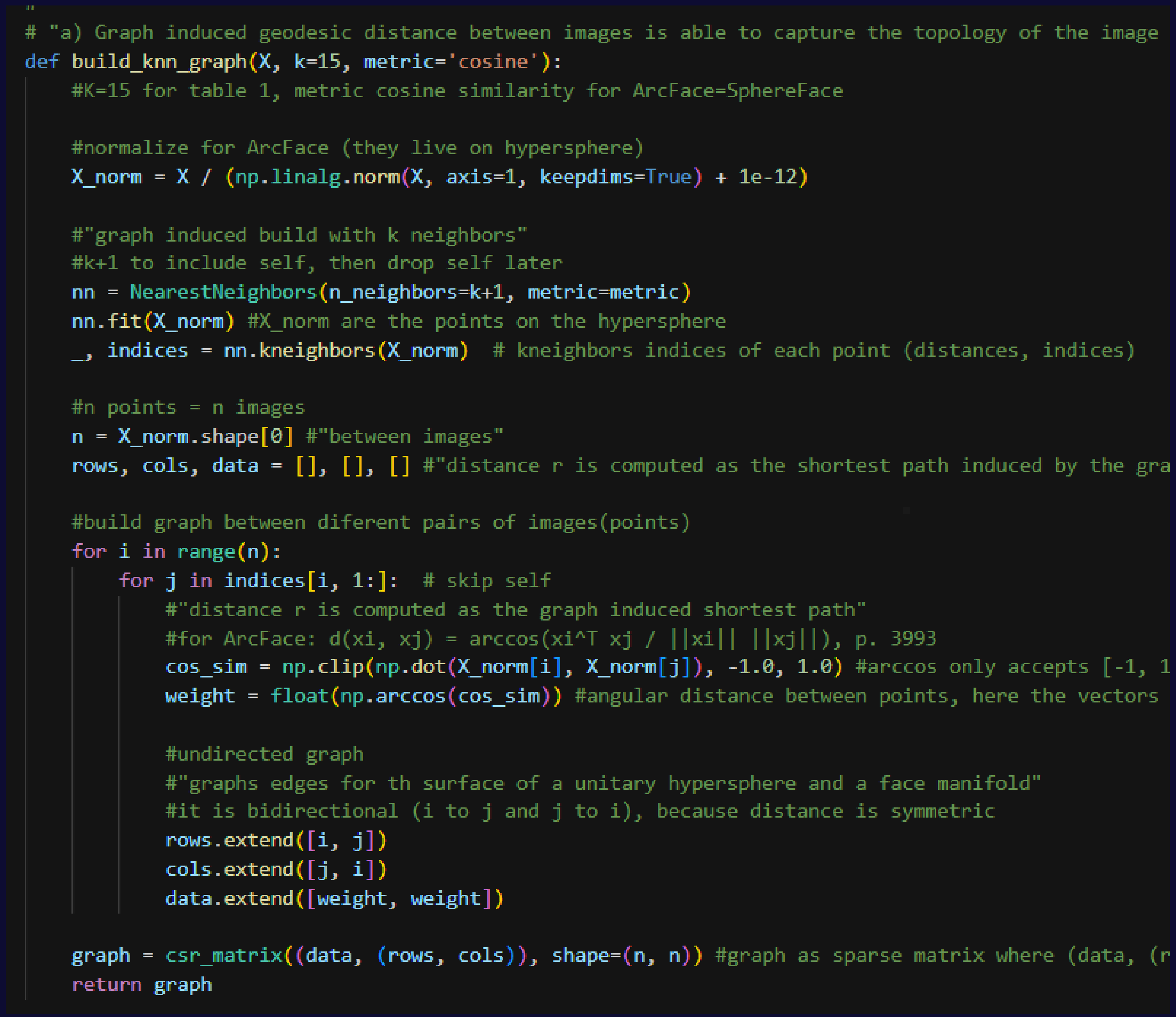
- +1e-12, prevents division by zero

Paper

Intrinsic Dimensionality

3.1. Estimating Intrinsic Dimension

- K neighbors to a - graph induced



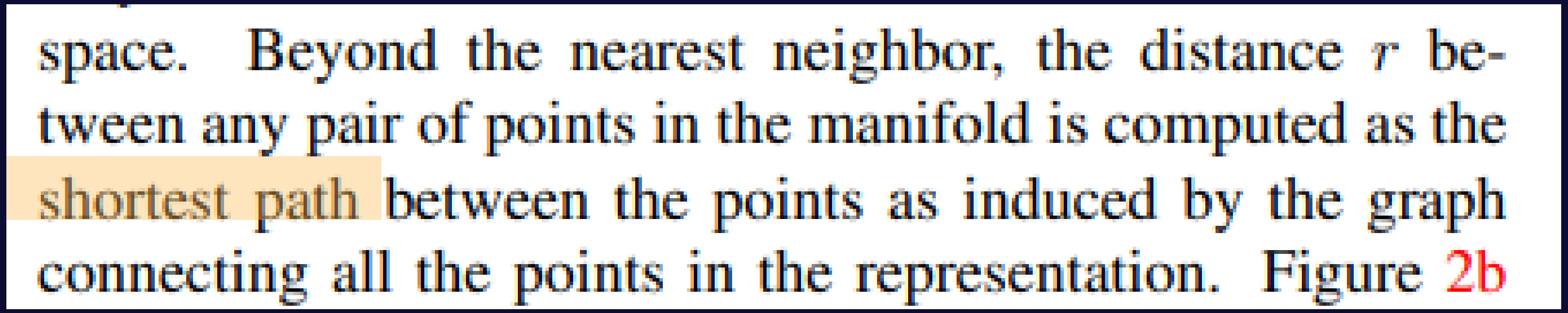
```python
# "a) Graph induced geodesic distance between images is able to capture the topology of the image
def build_knn_graph(X, k=15, metric='cosine'):
    #K=15 for table 1, metric cosine similarity for ArcFace=SphereFace

    #normalize for ArcFace (they live on hypersphere)
    X_norm = X / (np.linalg.norm(X, axis=1, keepdims=True) + 1e-12)

    #"graph induced build with k neighbors"
    #k+1 to include self, then drop self later
    nn = NearestNeighbors(n_neighbors=k+1, metric=metric)
    nn.fit(X_norm) #X_norm are the points on the hypersphere
    _, indices = nn.kneighbors(X_norm)  # kneighbors indices of each point (distances, indices)

    #n points = n images
    n = X_norm.shape[0] #"between images"
    rows, cols, data = [], [], [] #"distance r is computed as the shortest path induced by the gra

    #build graph between diferent pairs of images(points)
    for i in range(n):
        for j in indices[i, 1:]:  # skip self
            #"distance r is computed as the graph induced shortest path"
            #for ArcFace: d(xi, xj) = arccos(xi^T xj / ||xi|| ||xj||), p. 3993
            cos_sim = np.clip(np.dot(X_norm[i], X_norm[j]), -1.0, 1.0) #arccos only accepts [-1, 1
            weight = float(np.arccos(cos_sim)) #angular distance between points, here the vectors

            #undirected graph
            #"graphs edges for th surface of a unitary hypersphere and a face manifold"
            #it is bidirectional (i to j and j to i), because distance is symmetric
            rows.extend([i, j])
            cols.extend([j, i])
            data.extend([weight, weight])

    graph = csr_matrix((data, (rows, cols)), shape=(n, n)) #graph as sparse matrix where (data, (r
    return graph
```

Paper

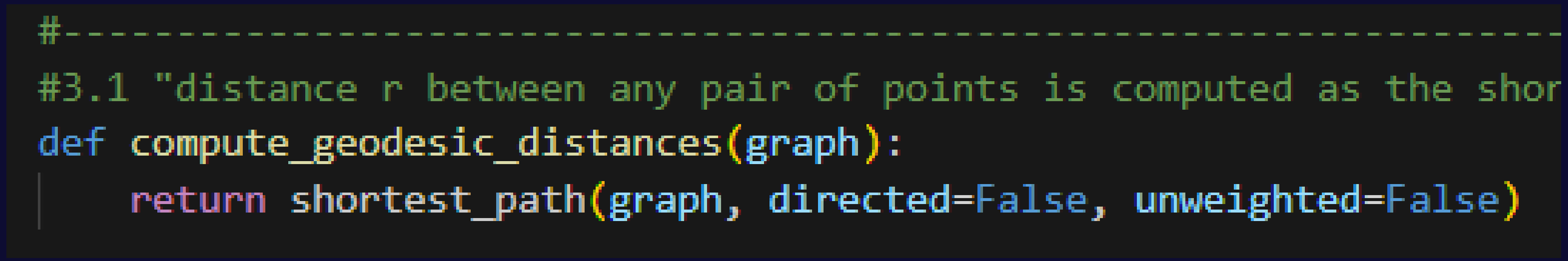
Intrinsic Dimensionality

3.1. Estimating Intrinsic Dimension

- Geodesic Distance



space. Beyond the nearest neighbor, the distance r between any pair of points in the manifold is computed as the shortest path between the points as induced by the graph connecting all the points in the representation. Figure 2b



```
#------------------------------------------------------------------
#3.1 "distance r between any pair of points is computed as the shor
def compute_geodesic_distances(graph):
    return shortest_path(graph, directed=False, unweighted=False)
```

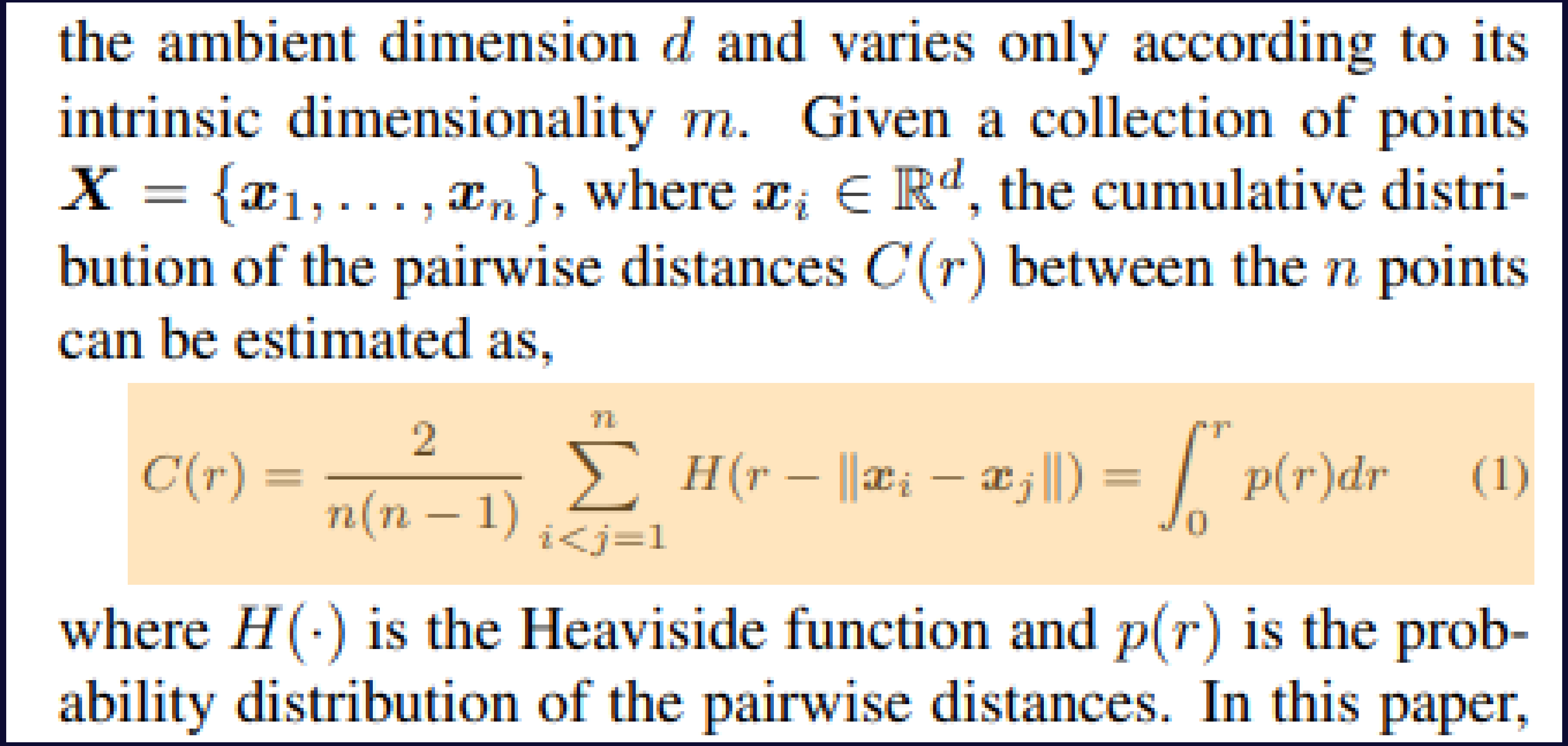
- **shortest path**:
  - min distance in graph represent on diperse matrix (CSR)
- **directed=false:**
  - no directed graph
- **unweighted=false:**
  - shortest_path uses the actual edge weights (angular distances) instead of treating every edge as weight 1.

Paper

Intrinsic Dimensionality

3.1. Estimating Intrinsic Dimension

- Distance Distributuion and C(r)



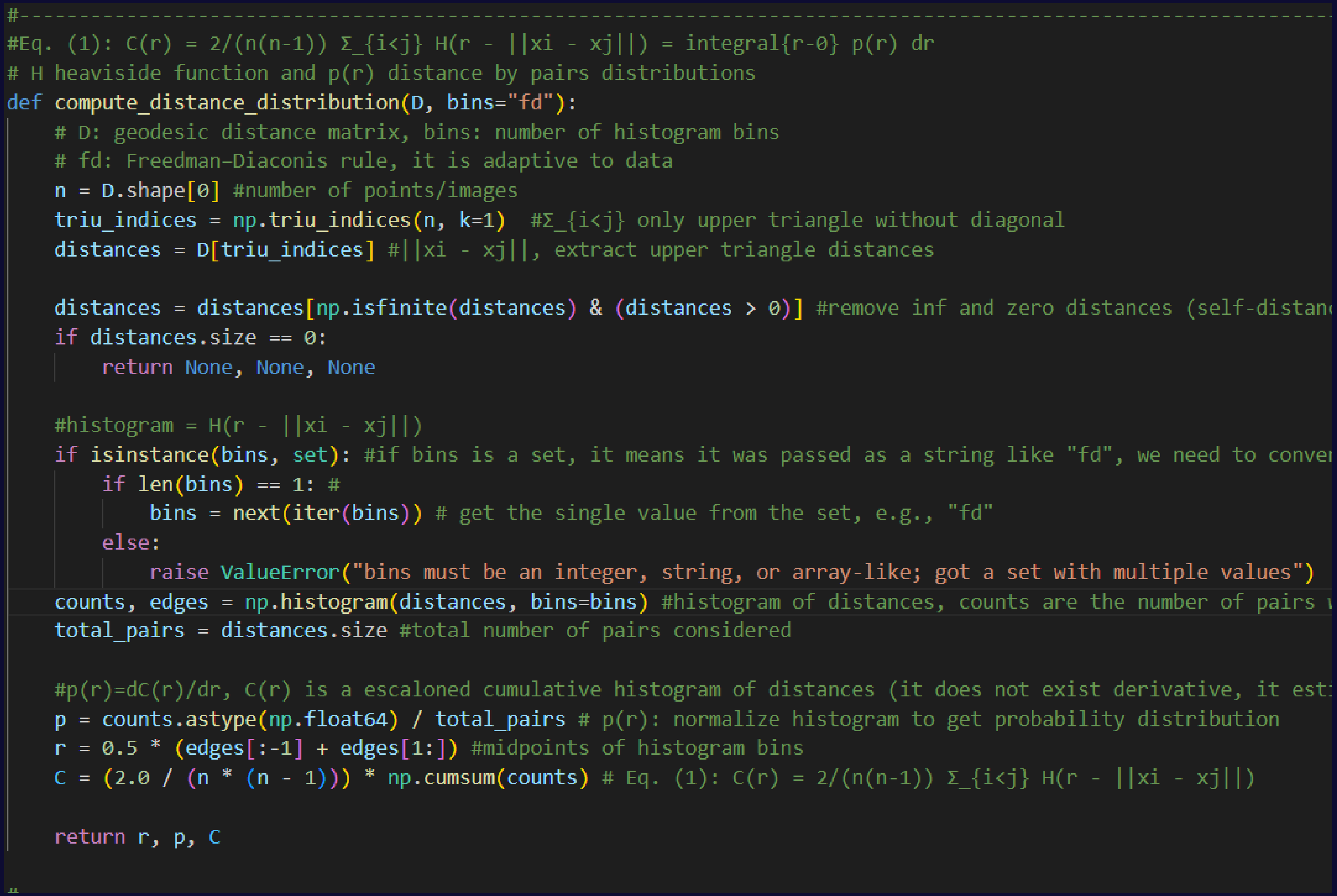
the ambient dimension d and varies only according to its intrinsic dimensionality m. Given a collection of points $\boldsymbol{X} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$, where $\boldsymbol{x}_i \in \mathbb{R}^d$, the cumulative distribution of the pairwise distances $C(r)$ between the n points can be estimated as,

$$C(r) = \frac{2}{n(n-1)} \sum_{i<j=1}^{n} H(r - \|\boldsymbol{x}_i - \boldsymbol{x}_j\|) = \int_0^r p(r)dr \qquad (1)$$

where $H(\cdot)$ is the Heaviside function and $p(r)$ is the probability distribution of the pairwise distances. In this paper,

- Σ_{i<j}: np.triu_indices(n,k=1)
- distances = D[triu_indices]   #||xi - xj||
- Histogram p(r), Cummulative on pairs C(r)
  - bins Fredman-Diaconis
  - p(r) distance distribution between pairs with finite data, not with a continuos density.
  - p(r)=dC/dr, in code it approximates with the geodesic distances histograms
  - So the histogram it is the estimation of the probability densities of r.
  - r, is the geodesic distance between 2 points in the manifold. In the code, is the center of the bins of the histogram of the distances

Paper



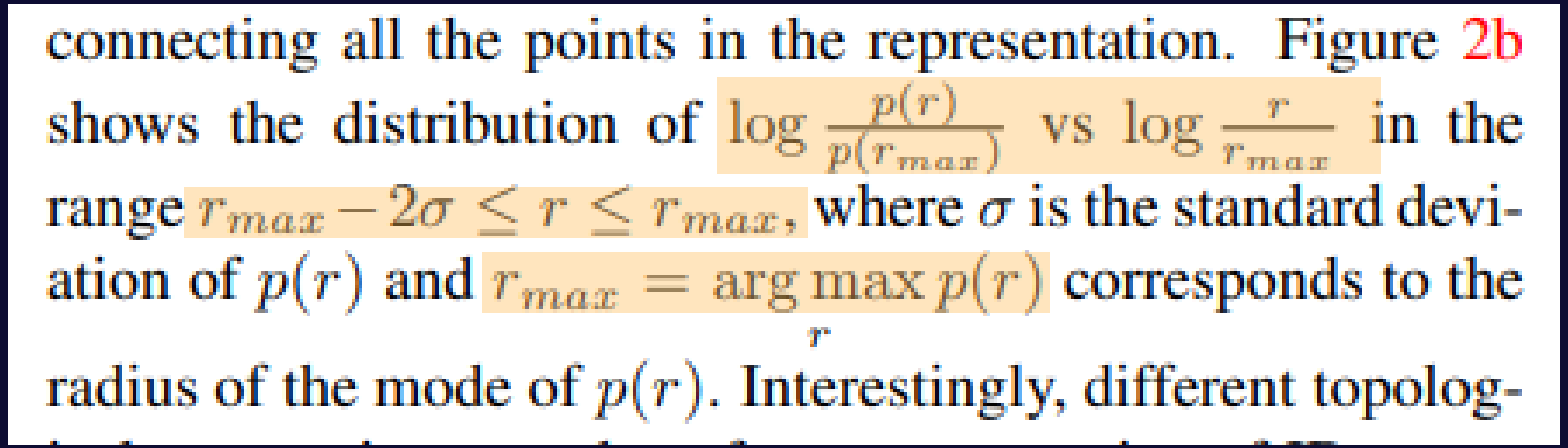
```
#--------------------------------------------------------------------------------------------------------------
#Eq. (1): C(r) = 2/(n(n-1)) Σ_{i<j} H(r - ||xi - xj||) = integral{r-0} p(r) dr
# H heaviside function and p(r) distance by pairs distributions
def compute_distance_distribution(D, bins="fd"):
    # D: geodesic distance matrix, bins: number of histogram bins
    # fd: Freedman-Diaconis rule, it is adaptive to data
    n = D.shape[0] #number of points/images
    triu_indices = np.triu_indices(n, k=1)  #Σ_{i<j} only upper triangle without diagonal
    distances = D[triu_indices] #||xi - xj||, extract upper triangle distances

    distances = distances[np.isfinite(distances) & (distances > 0)] #remove inf and zero distances (self-distanc
    if distances.size == 0:
        return None, None, None

    #histogram = H(r - ||xi - xj||)
    if isinstance(bins, set): #if bins is a set, it means it was passed as a string like "fd", we need to conver
        if len(bins) == 1: #
            bins = next(iter(bins)) # get the single value from the set, e.g., "fd"
        else:
            raise ValueError("bins must be an integer, string, or array-like; got a set with multiple values")
    counts, edges = np.histogram(distances, bins=bins) #histogram of distances, counts are the number of pairs
    total_pairs = distances.size #total number of pairs considered

    #p(r)=dC(r)/dr, C(r) is a escaloned cumulative histogram of distances (it does not exist derivative, it esti
    p = counts.astype(np.float64) / total_pairs # p(r): normalize histogram to get probability distribution
    r = 0.5 * (edges[:-1] + edges[1:]) #midpoints of histogram bins
    C = (2.0 / (n * (n - 1))) * np.cumsum(counts) # Eq. (1): C(r) = 2/(n(n-1)) Σ_{i<j} H(r - ||xi - xj||)

    return r, p, C

#
```

Paper

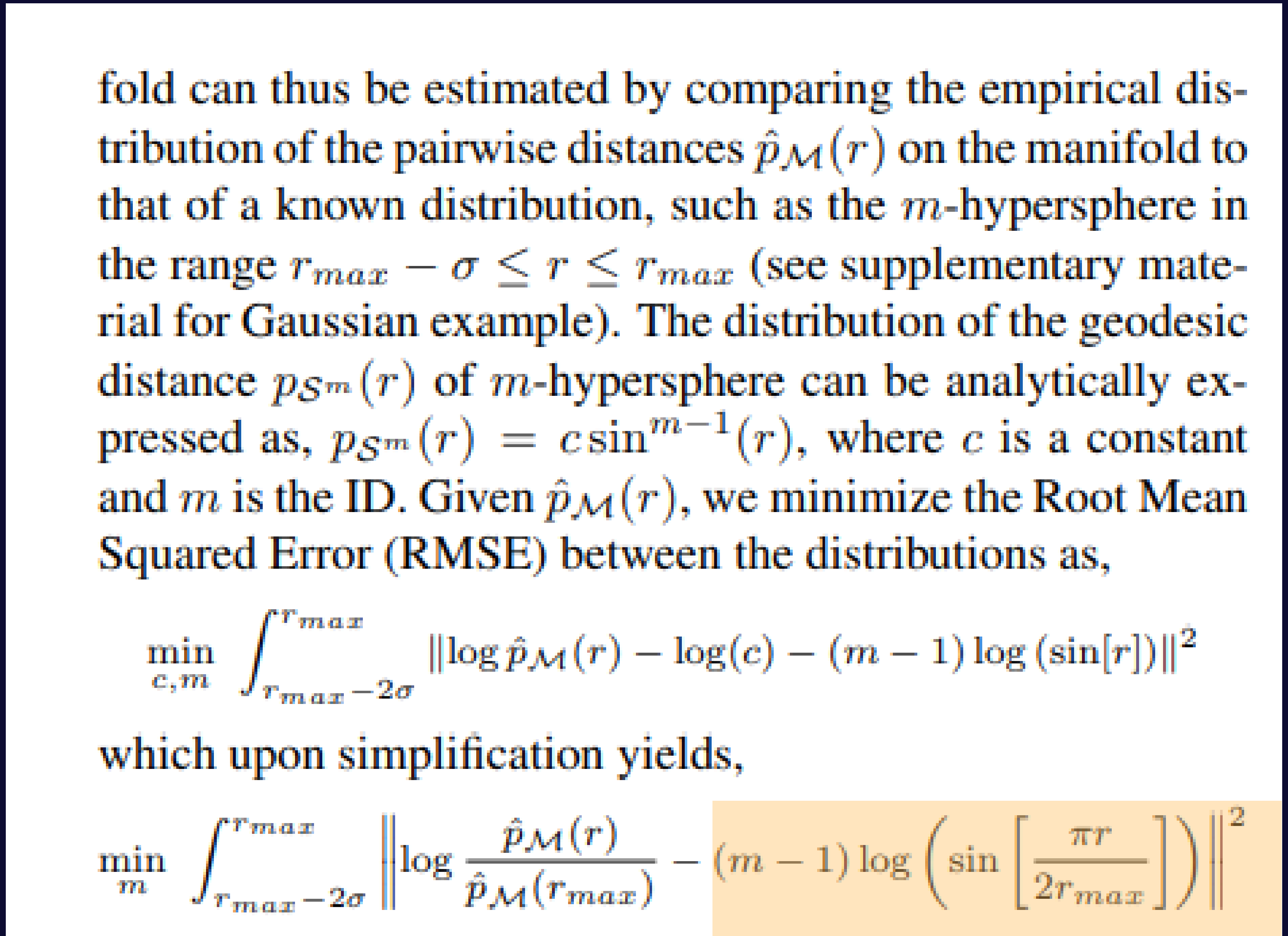
Intrinsic Dimensionality

3.1. Estimating Intrinsic Dimension =m

- Estimate Intrinsic Dimension



connecting all the points in the representation. Figure 2b shows the distribution of $\log \frac{p(r)}{p(r_{max})}$ vs $\log \frac{r}{r_{max}}$ in the range $r_{max} - 2\sigma \leq r \leq r_{max}$, where σ is the standard deviation of $p(r)$ and $r_{max} = \arg\max_r p(r)$ corresponds to the radius of the mode of $p(r)$. Interestingly, different topolog-



fold can thus be estimated by comparing the empirical distribution of the pairwise distances $\hat{p}_{\mathcal{M}}(r)$ on the manifold to that of a known distribution, such as the m-hypersphere in the range $r_{max} - \sigma \leq r \leq r_{max}$ (see supplementary material for Gaussian example). The distribution of the geodesic distance $p_{\mathcal{S}^m}(r)$ of m-hypersphere can be analytically expressed as, $p_{\mathcal{S}^m}(r) = c\sin^{m-1}(r)$, where c is a constant and m is the ID. Given $\hat{p}_{\mathcal{M}}(r)$, we minimize the Root Mean Squared Error (RMSE) between the distributions as,

$$\min_{c,m} \int_{r_{max}-2\sigma}^{r_{max}} \|\log \hat{p}_{\mathcal{M}}(r) - \log(c) - (m-1)\log(\sin[r])\|^2$$

which upon simplification yields,

$$\min_{m} \int_{r_{max}-2\sigma}^{r_{max}} \left\|\log \frac{\hat{p}_{\mathcal{M}}(r)}{\hat{p}_{\mathcal{M}}(r_{max})} - (m-1)\log\left(\sin\left[\frac{\pi r}{2r_{max}}\right]\right)\right\|^2$$

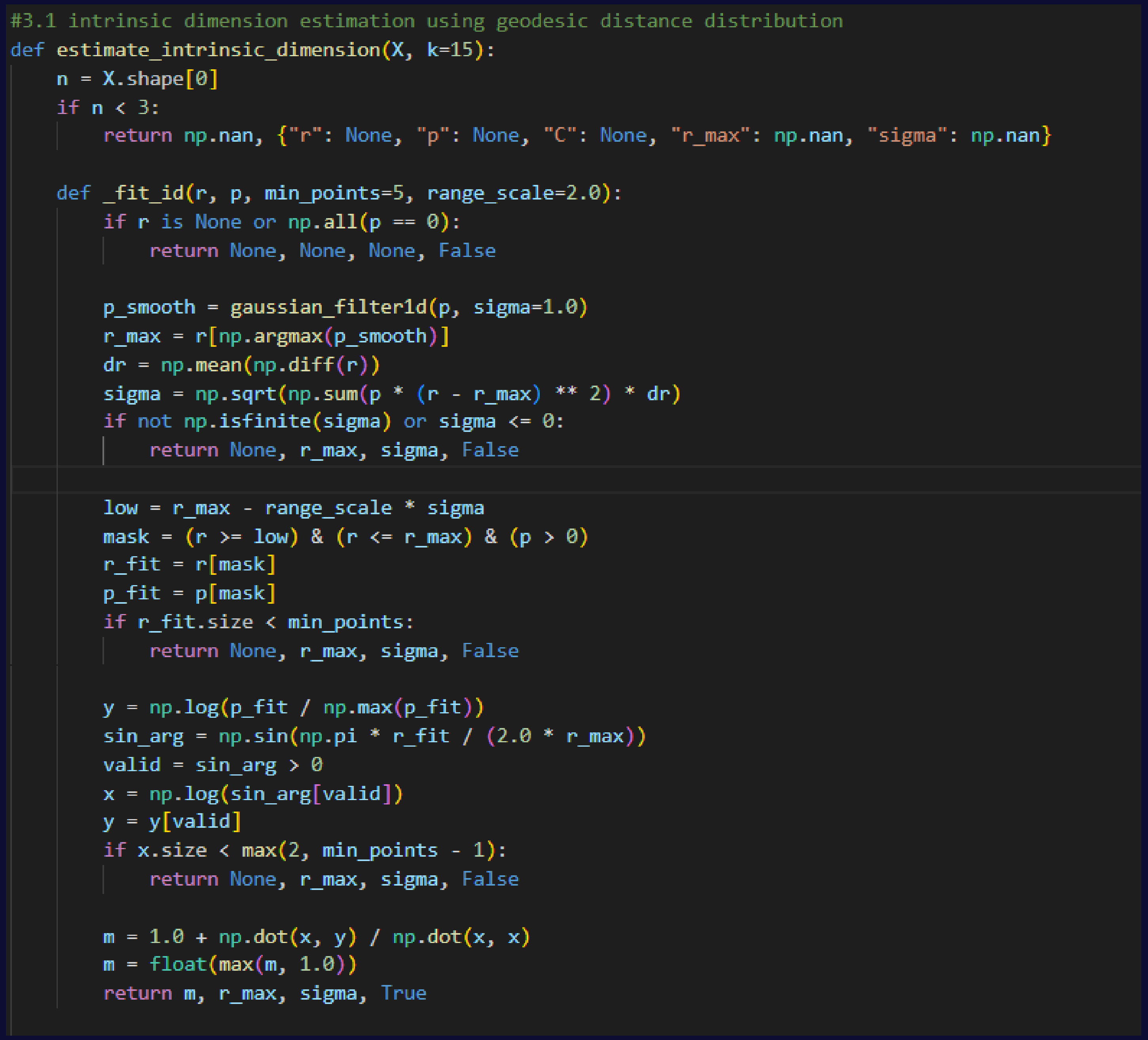
- fit id
  - Calculates r_max and σ from p(r)
  - Range from [r_max- 2σ, r_max],
  - 2= range_scale
    - In try fit use other values to broaden the range if the adjust fails (few poits, invalid values)
  - Adjust linear model in logs to obtain m (ID)
    - build x and y
      - y= log(p(r)/p(r_max))
      - x= log(sin(pi*r/2r_max))
      - So y is a straight line in x and can estimate the slope (m-1) with least squares
    - Adjust y=(m-1) by least squares
      - m=1+x*y/x*x
    - If m less than 1 it approx to 1 the ID cannot be less than 1 in decimal
  - Use gaussian_filter1d to smooth p, r_max = argmax, σ

Paper

Intrinsic Dimensionality

3.1. Estimating Intrinsic Dimension

- Estimate Intrinsic Dimension



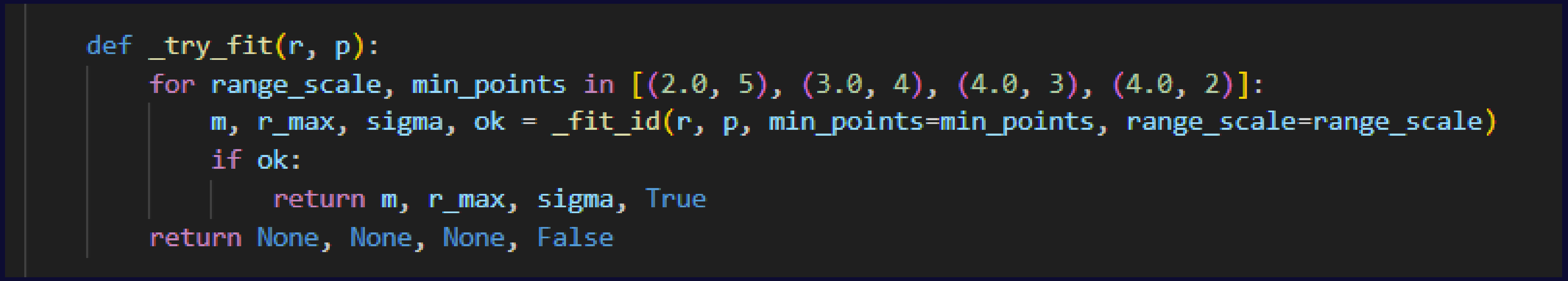
```python
#3.1 intrinsic dimension estimation using geodesic distance distribution
def estimate_intrinsic_dimension(X, k=15):
    n = X.shape[0]
    if n < 3:
        return np.nan, {"r": None, "p": None, "C": None, "r_max": np.nan, "sigma": np.nan}

    def _fit_id(r, p, min_points=5, range_scale=2.0):
        if r is None or np.all(p == 0):
            return None, None, None, False

        p_smooth = gaussian_filter1d(p, sigma=1.0)
        r_max = r[np.argmax(p_smooth)]
        dr = np.mean(np.diff(r))
        sigma = np.sqrt(np.sum(p * (r - r_max) ** 2) * dr)
        if not np.isfinite(sigma) or sigma <= 0:
            return None, r_max, sigma, False

        low = r_max - range_scale * sigma
        mask = (r >= low) & (r <= r_max) & (p > 0)
        r_fit = r[mask]
        p_fit = p[mask]
        if r_fit.size < min_points:
            return None, r_max, sigma, False

        y = np.log(p_fit / np.max(p_fit))
        sin_arg = np.sin(np.pi * r_fit / (2.0 * r_max))
        valid = sin_arg > 0
        x = np.log(sin_arg[valid])
        y = y[valid]
        if x.size < max(2, min_points - 1):
            return None, r_max, sigma, False

        m = 1.0 + np.dot(x, y) / np.dot(x, x)
        m = float(max(m, 1.0))
        return m, r_max, sigma, True
```

Paper

Intrinsic Dimensionality

3.1. Estimating Intrinsic Dimension

- Estimate Intrinsic Dimension



```
def _try_fit(r, p):
    for range_scale, min_points in [(2.0, 5), (3.0, 4), (4.0, 3), (4.0, 2)]:
        m, r_max, sigma, ok = _fit_id(r, p, min_points=min_points, range_scale=range_scale)
        if ok:
            return m, r_max, sigma, True
    return None, None, None, False
```

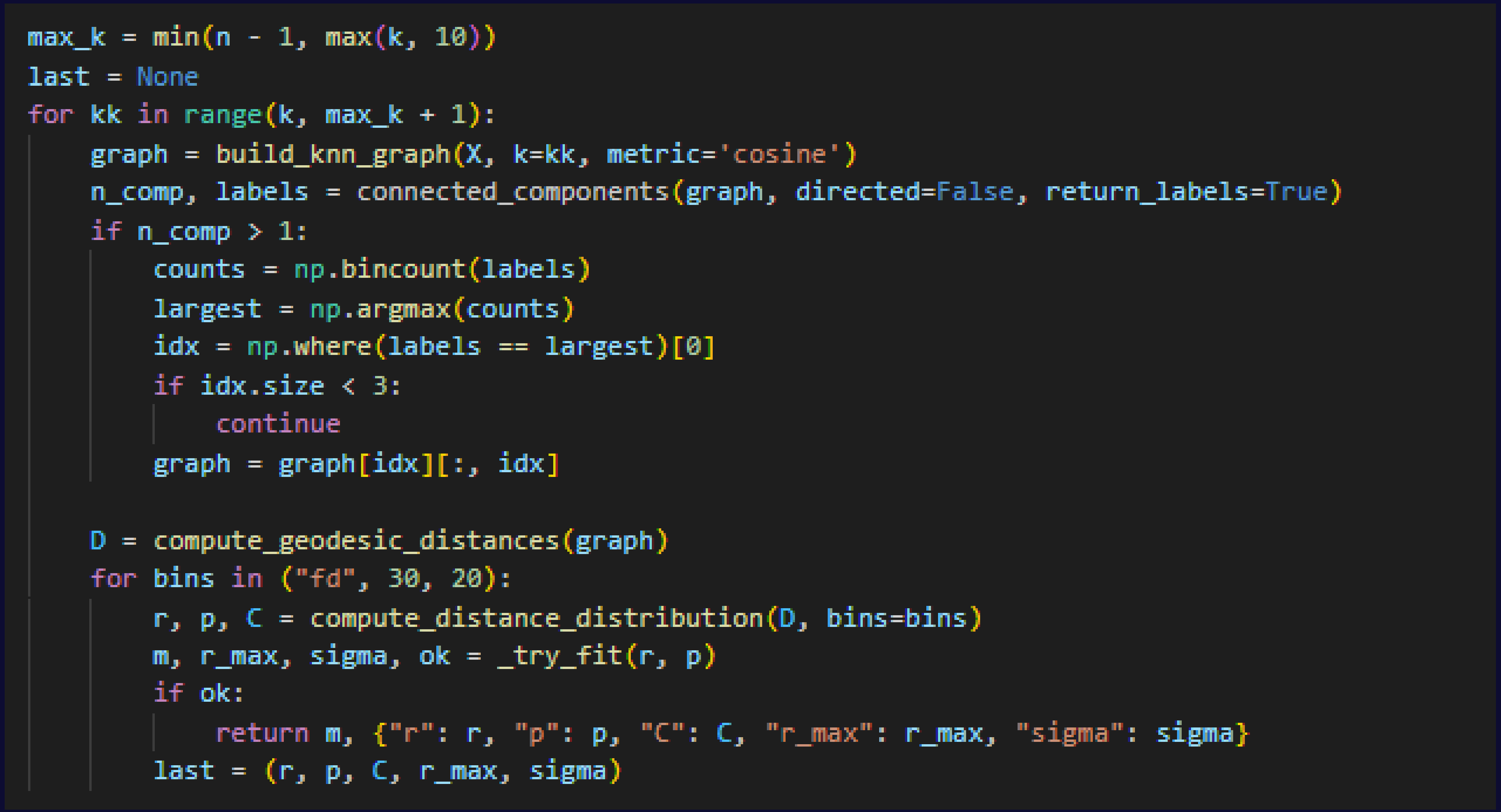
- try fit
  - it is a fallback, adjust fit id with broader ranges and less points required.
  - If one adjust function, returns that.
  - If none works, it returns false.

Paper

Intrinsic Dimensionality

3.1. Estimating Intrinsic Dimension

- Estimate Intrinsic Dimension



```
max_k = min(n - 1, max(k, 10))
last = None
for kk in range(k, max_k + 1):
    graph = build_knn_graph(X, k=kk, metric='cosine')
    n_comp, labels = connected_components(graph, directed=False, return_labels=True)
    if n_comp > 1:
        counts = np.bincount(labels)
        largest = np.argmax(counts)
        idx = np.where(labels == largest)[0]
        if idx.size < 3:
            continue
        graph = graph[idx][:, idx]

    D = compute_geodesic_distances(graph)
    for bins in ("fd", 30, 20):
        r, p, C = compute_distance_distribution(D, bins=bins)
        m, r_max, sigma, ok = _try_fit(r, p)
        if ok:
            return m, {"r": r, "p": p, "C": C, "r_max": r_max, "sigma": sigma}
        last = (r, p, C, r_max, sigma)
```

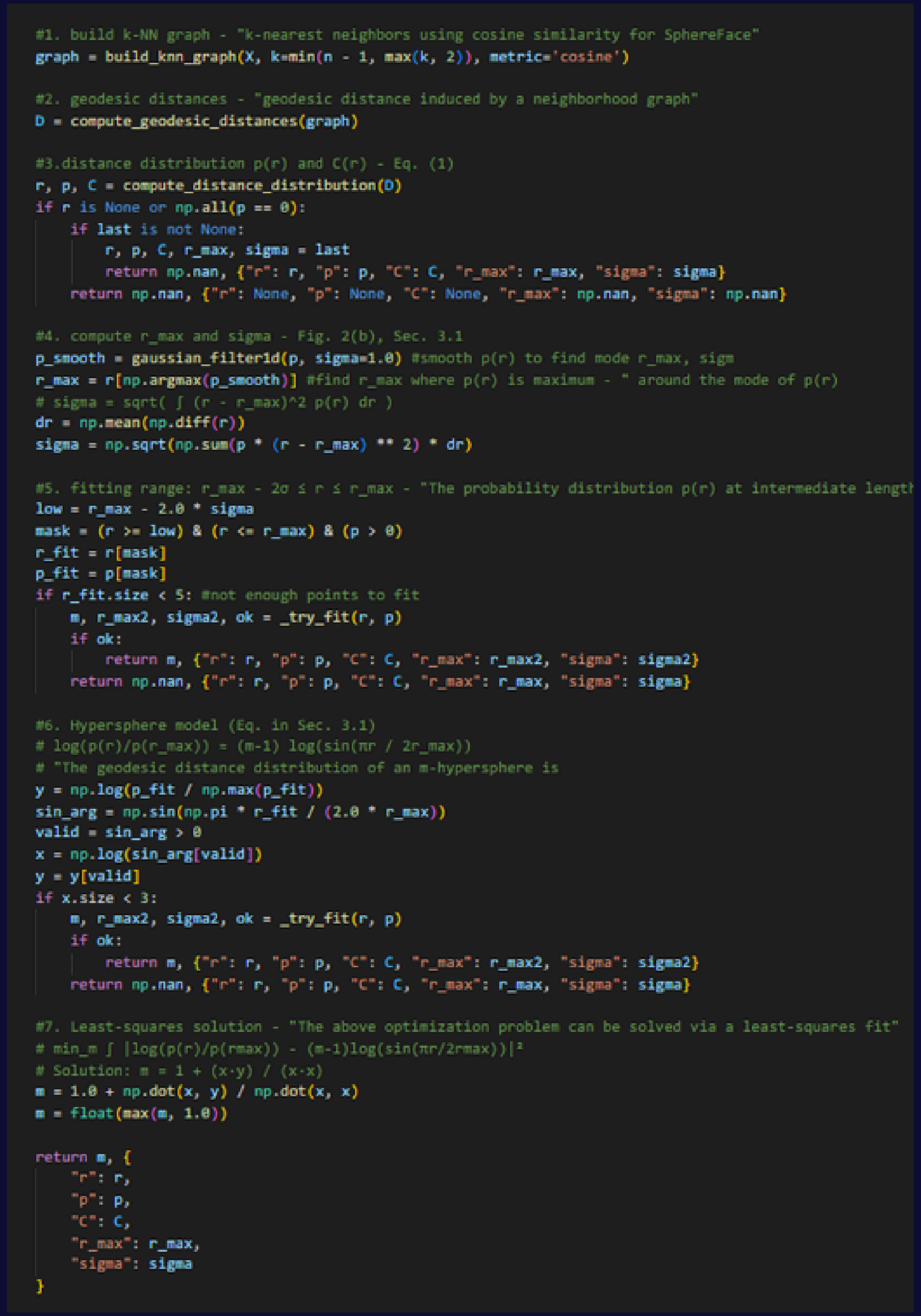
- The greater k to test at least 10, but never bigger than n-1 (because cannot have more neighbors than points)
- Try different k to build knn graph
- Calculate geodesic distances
- Use try fit to find a valid adjust

Paper

Intrinsic Dimensionality

3.1. Estimating Intrinsic Dimension

- Estimate Intrinsic Dimension



```python
#1. build k-NN graph - "k-nearest neighbors using cosine similarity for SphereFace"
graph = build_knn_graph(X, k=min(n - 1, max(k, 2)), metric='cosine')

#2. geodesic distances - "geodesic distance induced by a neighborhood graph"
D = compute_geodesic_distances(graph)

#3.distance distribution p(r) and C(r) - Eq. (1)
r, p, C = compute_distance_distribution(D)
if r is None or np.all(p == 0):
    if last is not None:
        r, p, C, r_max, sigma = last
        return np.nan, {"r": r, "p": p, "C": C, "r_max": r_max, "sigma": sigma}
    return np.nan, {"r": None, "p": None, "C": None, "r_max": np.nan, "sigma": np.nan}

#4. compute r_max and sigma - Fig. 2(b), Sec. 3.1
p_smooth = gaussian_filter1d(p, sigma=1.0) #smooth p(r) to find mode r_max, sigm
r_max = r[np.argmax(p_smooth)] #find r_max where p(r) is maximum - " around the mode of p(r)
# sigma = sqrt( ∫ (r - r_max)^2 p(r) dr )
dr = np.mean(np.diff(r))
sigma = np.sqrt(np.sum(p * (r - r_max) ** 2) * dr)

#5. fitting range: r_max - 2σ ≤ r ≤ r_max - "The probability distribution p(r) at intermediate length
low = r_max - 2.0 * sigma
mask = (r >= low) & (r <= r_max) & (p > 0)
r_fit = r[mask]
p_fit = p[mask]
if r_fit.size < 5: #not enough points to fit
    m, r_max2, sigma2, ok = _try_fit(r, p)
    if ok:
        return m, {"r": r, "p": p, "C": C, "r_max": r_max2, "sigma": sigma2}
    return np.nan, {"r": r, "p": p, "C": C, "r_max": r_max, "sigma": sigma}

#6. Hypersphere model (Eq. in Sec. 3.1)
# log(p(r)/p(r_max)) = (m-1) log(sin(πr / 2r_max))
# "The geodesic distance distribution of an m-hypersphere is
y = np.log(p_fit / np.max(p_fit))
sin_arg = np.sin(np.pi * r_fit / (2.0 * r_max))
valid = sin_arg > 0
x = np.log(sin_arg[valid])
y = y[valid]
if x.size < 3:
    m, r_max2, sigma2, ok = _try_fit(r, p)
    if ok:
        return m, {"r": r, "p": p, "C": C, "r_max": r_max2, "sigma": sigma2}
    return np.nan, {"r": r, "p": p, "C": C, "r_max": r_max, "sigma": sigma}

#7. Least-squares solution - "The above optimization problem can be solved via a least-squares fit"
# min_m ∫ |log(p(r)/p(rmax)) - (m-1)log(sin(πr/2rmax))|²
# Solution: m = 1 + (x·y) / (x·x)
m = 1.0 + np.dot(x, y) / np.dot(x, x)
m = float(max(m, 1.0))

return m, {
    "r": r,
    "p": p,
    "C": C,
    "r_max": r_max,
    "sigma": sigma
}
```

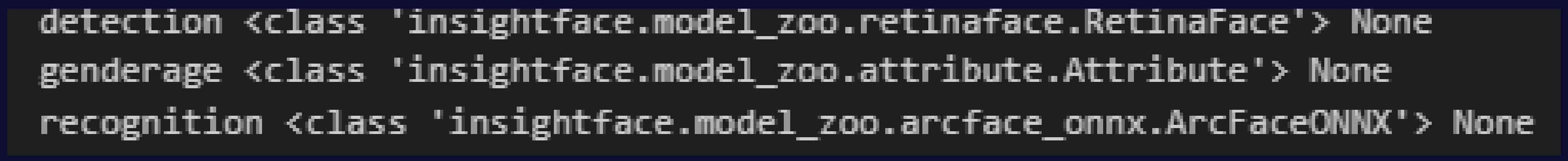
- build knn graph
- calculate geodesic distances
- obtains p(r) and C(r)
- calculates r_max and sigma
- selections the range [r_max-2sigma, r_max]
- adjust hipersphere model in log-log
- resolves m by least-squares
- get the results:
  - m, p,r,C,r_max, sigma

Distribution between Manifolds

RESULTS

Results

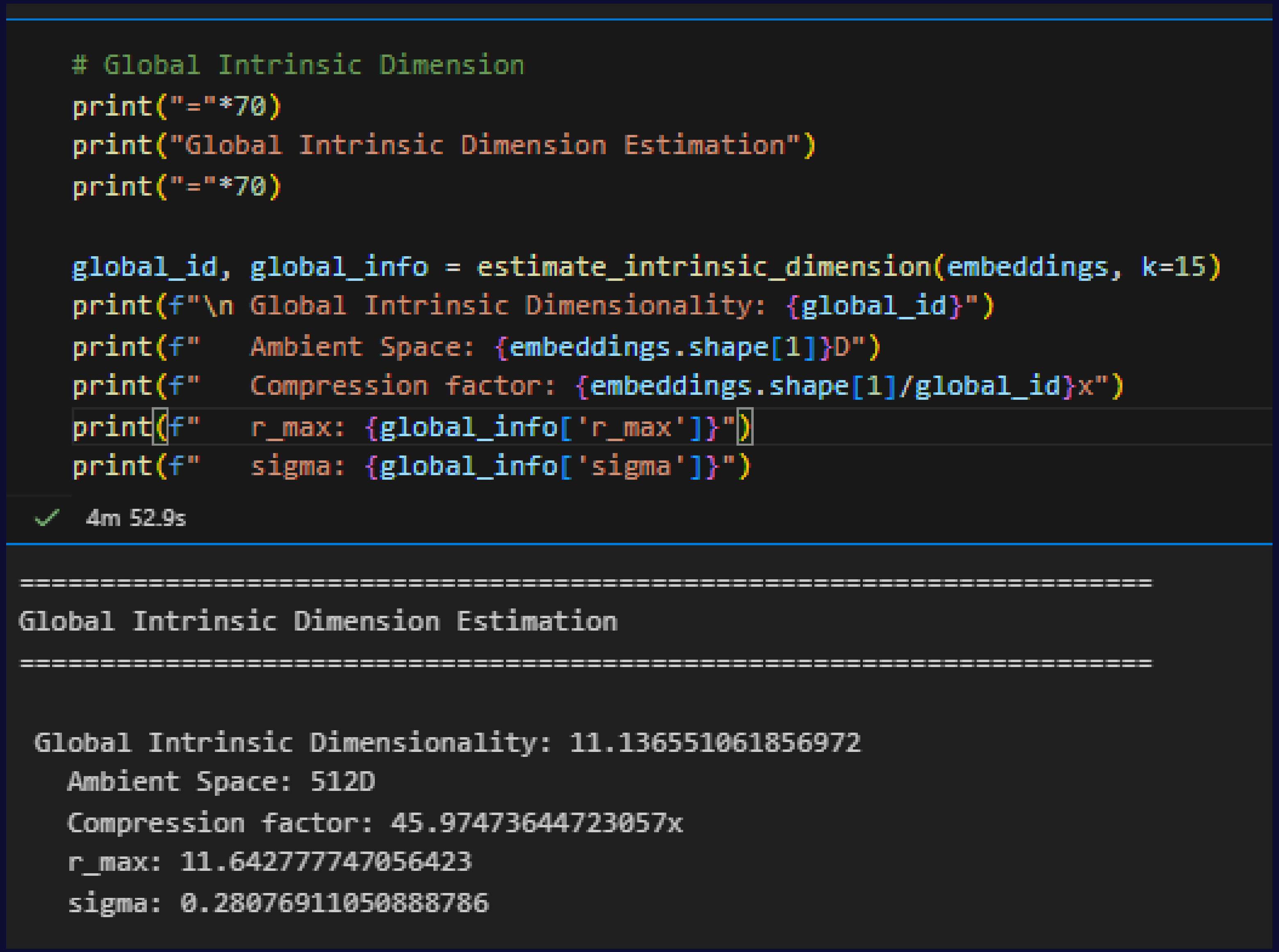
1. Use FRGC dataset
2. Face Detection
    a. RetinaFace via FaceAnalysis
        i. app = FaceAnalysis(name="buffalo_l")
3. Extract Embeddings
    a. ArcFaceONNX
4. Cluster by identity - 568 identities
5. Intrinsic dimensionality



```
detection <class 'insightface.model_zoo.retinaface.RetinaFace'> None
genderage <class 'insightface.model_zoo.attribute.Attribute'> None
recognition <class 'insightface.model_zoo.arcface_onnx.ArcFaceONNX'> None
```

Results

Global Instrinsic Dimension Estimation



```
# Global Intrinsic Dimension
print("="*70)
print("Global Intrinsic Dimension Estimation")
print("="*70)

global_id, global_info = estimate_intrinsic_dimension(embeddings, k=15)
print(f"\n Global Intrinsic Dimensionality: {global_id}")
print(f"   Ambient Space: {embeddings.shape[1]}D")
print(f"   Compression factor: {embeddings.shape[1]/global_id}x")
print(f"   r_max: {global_info['r_max']}")
print(f"   sigma: {global_info['sigma']}")
```

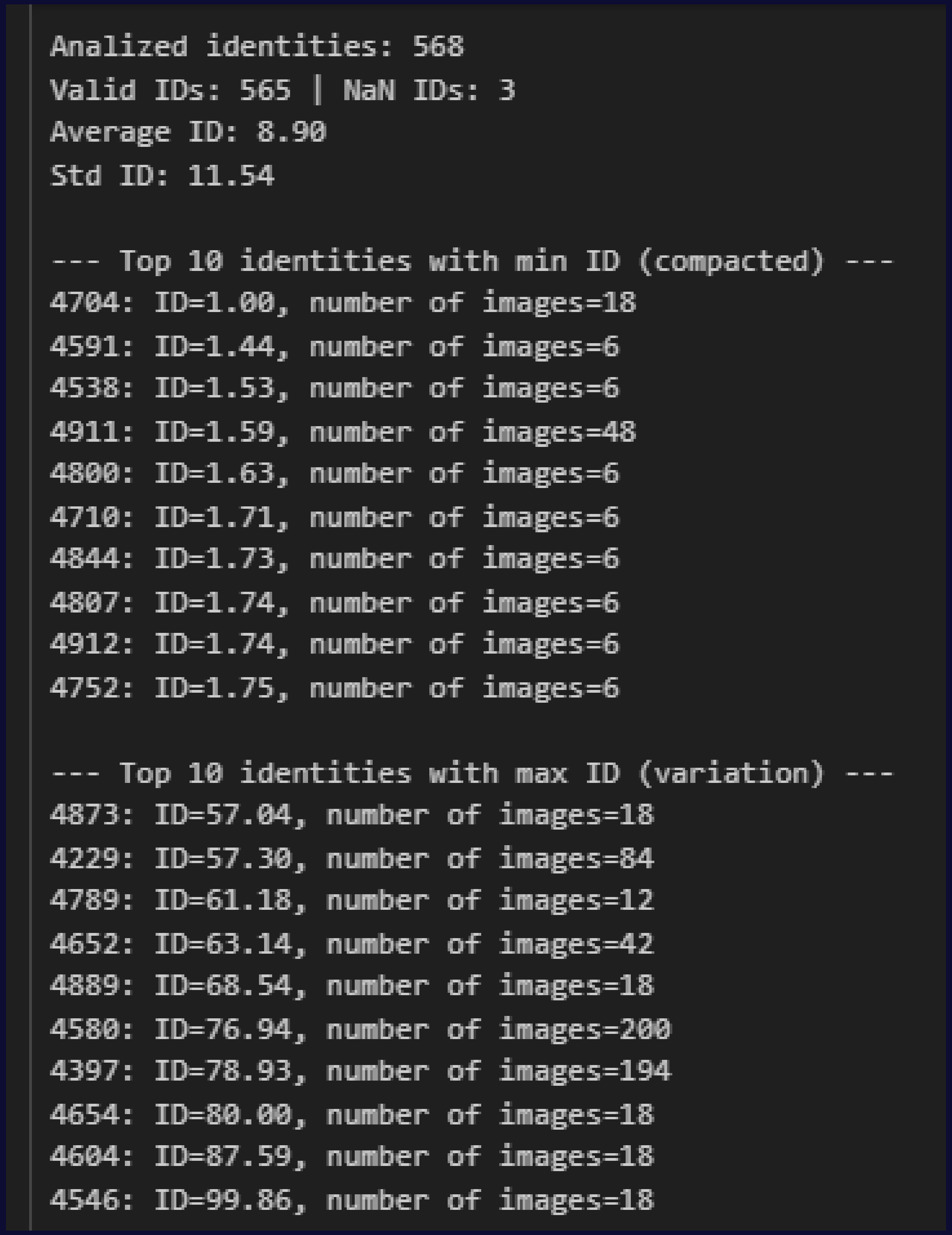
✓ 4m 52.9s

```
======================================================================
Global Intrinsic Dimension Estimation
======================================================================

 Global Intrinsic Dimensionality: 11.136551061856972
   Ambient Space: 512D
   Compression factor: 45.97473644723057x
   r_max: 11.642777747056423
   sigma: 0.28076911050888786
```

- Global ID for all the embeddings
- Original dimension of embeddings= 512D
- How much can compress d/m = compression factor = 45.97
  - d= ambient space = 512
  - m= ID = 11.14

Results

Intrinsic Dimension per Identity


```
Analized identities: 568
Valid IDs: 565 | NaN IDs: 3
Average ID: 8.90
Std ID: 11.54

--- Top 10 identities with min ID (compacted) ---
4704: ID=1.00, number of images=18
4591: ID=1.44, number of images=6
4538: ID=1.53, number of images=6
4911: ID=1.59, number of images=48
4800: ID=1.63, number of images=6
4710: ID=1.71, number of images=6
4844: ID=1.73, number of images=6
4807: ID=1.74, number of images=6
4912: ID=1.74, number of images=6
4752: ID=1.75, number of images=6

--- Top 10 identities with max ID (variation) ---
4873: ID=57.04, number of images=18
4229: ID=57.30, number of images=84
4789: ID=61.18, number of images=12
4652: ID=63.14, number of images=42
4889: ID=68.54, number of images=18
4580: ID=76.94, number of images=200
4397: ID=78.93, number of images=194
4654: ID=80.00, number of images=18
4604: ID=87.59, number of images=18
4546: ID=99.86, number of images=18
```

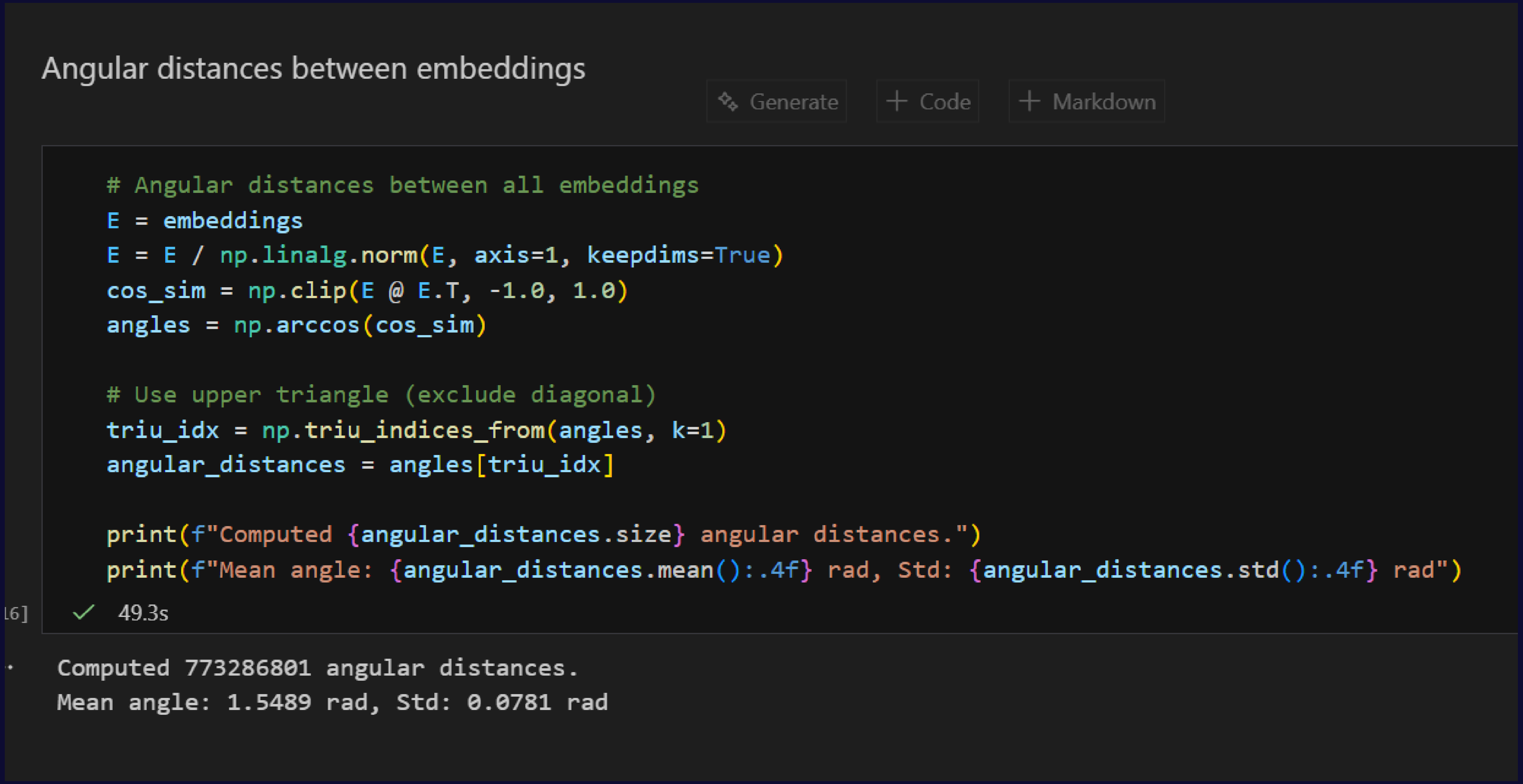
- ID per identity
- 3 NaN ID, maybe:
  - Distances can be empty or invalid
  - The distance distribution can be too flat or degenerate, making sigma non-finite or ≤0.
  - There may be too few valid bins in the fit range, so the hypersphere fit fails.
  - The kNN graph can be disconnected; if the largest component is too small, the fit never succeeds.
  - Any of those leads to m_id=None, which converts to NaN.
- ** approx the ID ≤1?

Distribution of Angles

Angular distances between embeddings

Results

- Using FRGC dataset
- RetinaFace + ArcFace



Angular distances between embeddings

```python
# Angular distances between all embeddings
E = embeddings
E = E / np.linalg.norm(E, axis=1, keepdims=True)
cos_sim = np.clip(E @ E.T, -1.0, 1.0)
angles = np.arccos(cos_sim)

# Use upper triangle (exclude diagonal)
triu_idx = np.triu_indices_from(angles, k=1)
angular_distances = angles[triu_idx]

print(f"Computed {angular_distances.size} angular distances.")
print(f"Mean angle: {angular_distances.mean():.4f} rad, Std: {angular_distances.std():.4f} rad")
```

```
Computed 773286801 angular distances.
Mean angle: 1.5489 rad, Std: 0.0781 rad
```

Future Work

- Distribution between manifold
  - dimensionality estimation, distribution angles, angles between subpaces
  - same intrinsic dimension, compare angle distances
  - filter: number of img K, whose intrinsic dimension is K-1
- Distribution embeddings - geometry structure
- Compare to face networks
  - Arcface, MagFace, AdaFace, Facenet, Dlib
- Organize better documentation in LaTEX

Thank You