به نام خدا

فاطمه محمدی

شماره دانشجویی: ۴۰۲۱۳۵۰۳۹۶

تمرین کاهش ابعاد

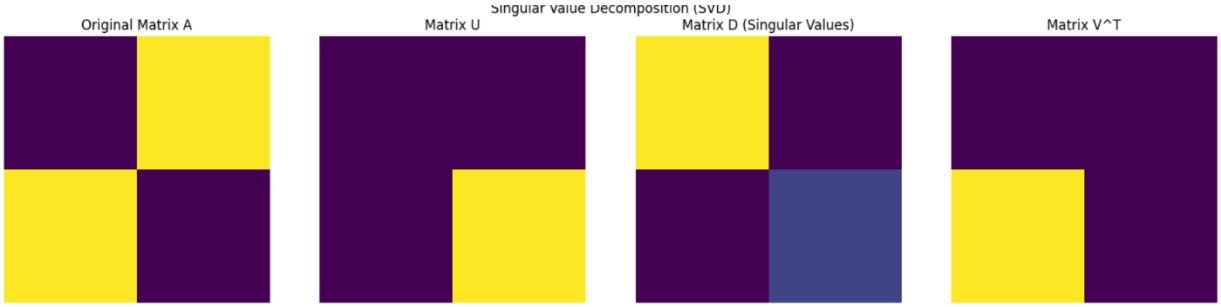
تمرین اول: خروجی نرم افزار

```
Matrix A:
[[2 3]
 [3 2]]

Matrix U:
[[-0.70710678 -0.70710678]
 [-0.70710678  0.70710678]]

Matrix D (Diagonal Singular Values):
[[5. 0.]
 [0. 1.]]

Matrix V^T:
[[-0.70710678 -0.70710678]
 [ 0.70710678 -0.70710678]]
```

تحلیل خروجی:

۱. مقادیر منفرد (D)مقادیری مثبت هستند که در ماتریس قطری قرار دارند.
۲. ماتریس Uو VTماتریس‌های متعامد هستند که ساختار اصلی ماتریس Aرا تجزیه می‌کنند.
۳. با استفاده از این ماتریس‌ها می‌توان Aرا به صورت زیر بازسازی کرد:

$$A = U.D.V^T$$



حل به روش دستی:

در فایل ضمیمه پی دی اف قرار داده شد.

تمرین دوم:

Manifold learning مجموعه ای از تکنیک ها برای کاهش ابعاد است که هدف آن شناسایی یک Manifold (خم) با ابعاد پایین تر درون داده هایی با ابعاد بالا می باشد.

کاهش ابعاد غیرخطی با Manifold learning روش های unsupervised را پوشش می دهند که تلاش می کنند Manifold های کم بعد را در فضای P-بعد اصلی شناسایی کنند که نشان دهنده چگالی داده بالا است

سپس آن روش ها یک Map از فضای با ابعاد بالا به Embeddingبا ابعاد پایین ارائه می دهد.

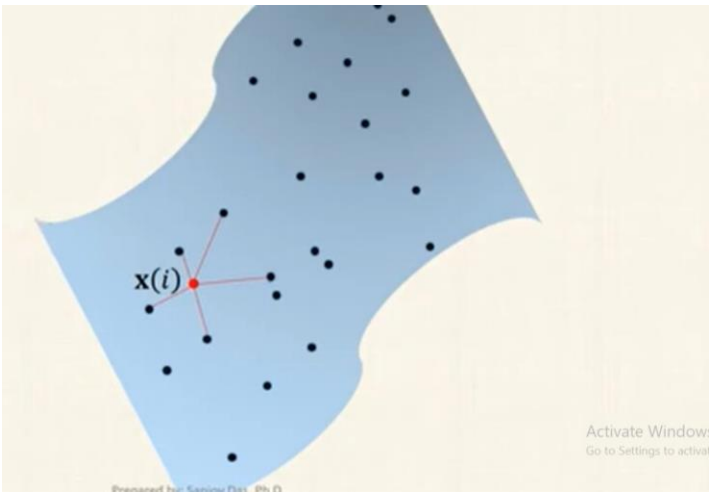
دو روش محبوب در این زمینه عبارتند از:

1.Locally linear Embedding

2.Isomap

Locally Linear Embedding

Roweis & Saul: Nonlinear Dimensionality Reduction by locally linear embedding .



Inputs: X(i) : D × 1 vectors (i=1,2,….,N)

Parameters : K

Output : y(i) : M× 1 vectors

N points: D dims ⟶ M dims

(M < D)

Locally Linear Embedding

Step 1 : Local Linearization

Define K-nearest neighborhood $\aleph_i$

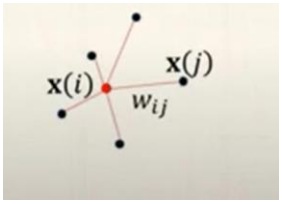
Separately around each point X(i)

Express X(i) as weighted sum of neighbors X(j)

Minimize w_{ij} :

$$\left\| X(i) - \sum_{j \in \aleph_i} w_{ij} X(j) \right\|^2$$

Hence: X(i) ≈ $\sum_{j \in \aleph_i} w_{ij} X(j)$



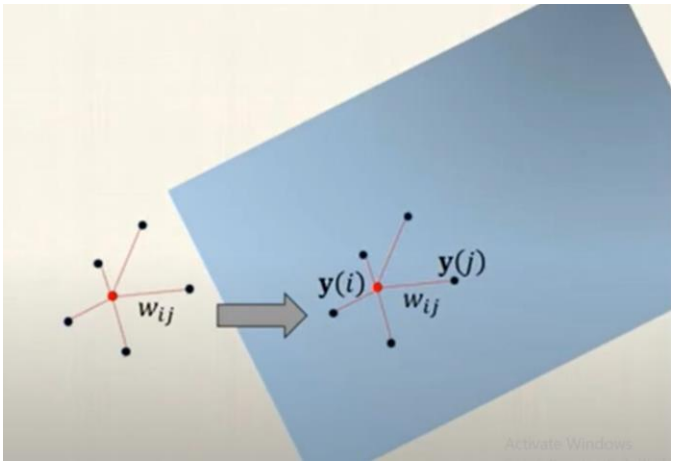
Step 2: Low Dim.Embedding

Find Y(i) for all points

Minimize Y(i):

$$\sum_i \left\| Y(i) - \sum_{j \in \aleph_i} w_{ij}\, Y(j) \right\|^2$$

Hence: Y(i) $\approx \sum_{j \in \aleph_i} w_{ij}\, Y(j)$



Isomap

Isomapیک روش کاهش ابعاد غیرخطی بر پایه (Manifold Learning)است.

این روش با محاسبه فاصله‌های هندسی (فاصله روی سطح خم یا "ژئودزیک") بین نقاط داده کار می‌کند.

سپس از الگوریتم‌های تحلیل مؤلفه‌های اصلی (MDS) برای نمایش داده‌ها در یک فضای بُعد پایین‌تر استفاده می‌کند.

روشی را برای محاسبه ماتریس فاصله با MDS ترکیب می کند

محاسبه فواصل بر اساس فواصل ارزیابی شده بر روی گراف همسایگی است:

۱. همسایگان هر نقطه را مشخص کنید همه نقاط در برخی از شعاع ثابت یا K نزدیک ترین همسایگان
۲. یک گراف همسایگی بسازید هر نقطه اگر نزدیک ترین همسایه K باشد به نقطه دیگر متصل شود.طول یال برابر با فاصله اقلیدسی است
۳. برای ساختن ماتریس فاصله D، کوتاه ترین مسیر را بین دو نقطه d_{ij} محاسبه کنید
۴. MDSرا روی D پیاده کنید.

کدها در پایتون و توضیح نمودارها

فراخوانی کتابخانه های مورد نیاز برای فراخوانی داده های رقم های دست نویس و کتابخانه های مورد نیاز برای به کارگیری روش های Manifold learning از جمله Locally linear Embedding و Isomap

فرق داده های handwritten digits با MNIST

داده‌های digitsاز مجموعه داده‌های کتابخانه **Scikit-learn**هستند، در حالی که **MNIST**یک مجموعه داده متفاوت و بزرگ‌تر از ارقام دست نویس است.

داده‌های digitsکوچکتر و ساده‌تر هستند و برای الگوریتم‌های کلاسیک مانند LLE، Isomapیا PCAمناسب‌تر هستند.

داده‌های MNISTبزرگ‌تر و پیچیده‌تر بوده و معمولاً برای شبکه‌های عصبی و الگوریتم‌های یادگیری عمیق استفاده می‌شود.

برای استفاده از داده های MNIST می توان کتابخانه های Keras وTensorflow یا Pytorchبارگذاری کرد

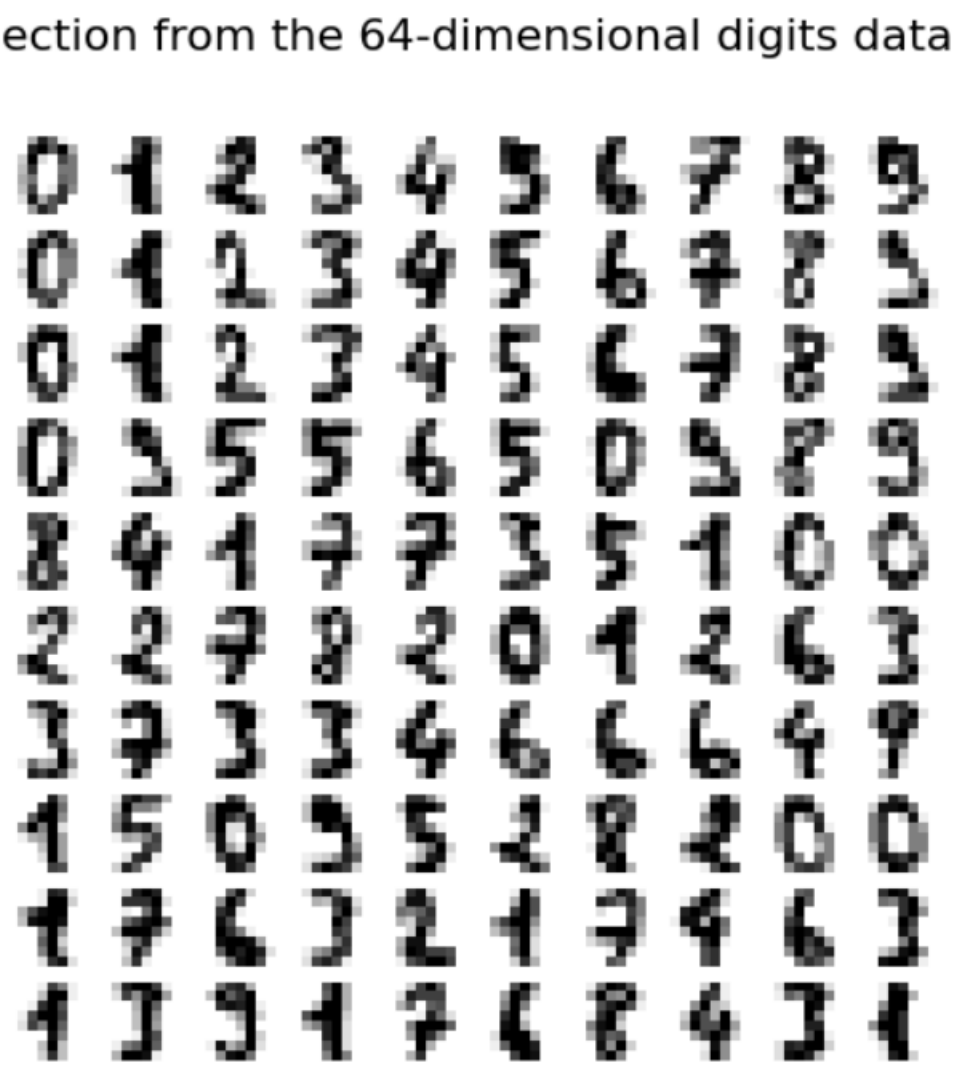
```
from sklearn.datasets import load_digits    #فراخوانی داده های ارقام دست نویس
import matplotlib.pyplot as plt             #رسم نمودار
import numpy as np
from matplotlib import offsetbox
from sklearn.preprocessing import MinMaxScaler
from sklearn.manifold import Isomap,LocallyLinearEmbedding
```

فراخوانی داده ها

```
[4]  #فراخوانی مجموعه داده

     digits = load_digits()
     X, y = digits.data, digits.target
     n_samples, n_features = X.shape
     n_neighbors = 30
```

رسم داده ها

```
[5]  # رسم داده ها

     fig, axs = plt.subplots(nrows=10, ncols=10, figsize=(6, 6))
     for idx, ax in enumerate(axs.ravel()):
         ax.imshow(X[idx].reshape((8, 8)), cmap=plt.cm.binary)
         ax.axis("off")
     _ = fig.suptitle("A selection from the 64-dimensional digits dataset", fontsize=16)
```



کدهای زیر داده های Embedding (مانند خروجی روش های کاهش ابعاد مانند LLE یا Isomap) را به همراه تصاویر واقعی ارقام دست نویس نمایش می دهد.این کد امکان این را فراهم می کند که ببینیم هر نقطه در فضای کاهش یافته به چه تصویر یا برچسبی مربوط می شود.این کار امکان بررسی بهتر بر روی داده ها را فراهم میکند.

```
def plot_embedding(X, title)
```

این تابع داده های Embedding(مانند Isomapیا Locally linear Embeddig) را دریافت کرده و خروجی بصری ایجاد می کند.

X خروجی های الگوریتم هایی مانند LLE یاIsomap با ابعاد دو بعدی

```
for digit in digits.target_names:
        ax.scatter(
            *X[y == digit].T,
            marker=f"${digit}$",
            s=60,
            color=plt.cm.Dark2(digit),
            alpha=0.425,
            zorder=2,
        )
```

با استفاده از scatter نقاط Embedding(خروجی های الگوریتم های کاهش بعد غیرخطی) برای هر رقم به صورت مجزا با رنگ های متفاوت رسم می شود.

رنگ ها از colormap (Dark2) استفاده می کنند.

علامت marker هر نقطه، برچسب عدد مربوطه (مثلا ۰و ۱ و...) است

```
 shown_images = np.array([[1.0, 1.0]])  # just something
big        #اضافه کردن تصاویر کوچک (Annotation Boxes)
    for i in range(X.shape[0]):
        # plot every digit on the embedding
        # show an annotation box for a group of digits
        dist = np.sum((X[i] - shown_images) ** 2, 1)
        if np.min(dist) < 4e-3:
            # don't show points that are too close
            continue
        shown_images = np.concatenate([shown_images, [X[i]]], axis=0)
```

```
        imagebox = offsetbox.AnnotationBbox(
            offsetbox.OffsetImage(digits.images[i], cmap=plt.cm.gray_r),
X[i]
        )
        imagebox.set(zorder=1)
        ax.add_artist(imagebox)
```

برای هر نقطه در نمودار:

فاصله نقطه با نقاط دیگر بررسی می شود

اگر فاصله از حد معینی کمتر باشد، تصویر نشان داده نمی شود (برای جلوگیری از نمایش تصاویر نزدیک بهم)

در غیر این صورت تصویر کوچک (thumbnail) مربوط به رقم روی نمودار درج می شود.

تصاویر از مجموعه داده های digits.images خوانده می شود و بااستفاده از AnnotationBbox نمایش داده می شوند.

این کد برای تحلیل بصری داده‌های کاهش ابعادشده استفاده می‌شود تا بتوان:

ساختار داده‌ها را در فضای دو بُعدی مشاهده کرد.

بررسی کرد که کدام نقاط (تصاویر) نزدیک به هم قرار گرفته‌اند و آیا الگوریتم توانسته ارقام مشابه را کنار هم قرار دهد یا خیر.

```
[9]  def plot_embedding(X, title):
         _, ax = plt.subplots()
         X = MinMaxScaler().fit_transform(X) #مقیاس بندی داده ها برای اینکه داده ها در محدوده مناسبی برای رسم قرار گیرند
     #رسم نقاط با برچسب هر عدد
         for digit in digits.target_names:
             ax.scatter(
                 *X[y == digit].T,
                 marker=f"${digit}$",
                 s=60,
                 color=plt.cm.Dark2(digit),
                 alpha=0.425,
                 zorder=2,
             )
         shown_images = np.array([[1.0, 1.0]])  # just something big        #اضافه کردن تصاویر کوچک (Annotation Boxes)
         for i in range(X.shape[0]):
             # plot every digit on the embedding
             # show an annotation box for a group of digits
             dist = np.sum((X[i] - shown_images) ** 2, 1)
             if np.min(dist) < 4e-3:
                 # don't show points that are too close
                 continue
             shown_images = np.concatenate([shown_images, [X[i]]], axis=0)
             imagebox = offsetbox.AnnotationBbox(
                 offsetbox.OffsetImage(digits.images[i], cmap=plt.cm.gray_r), X[i]
             )
             imagebox.set(zorder=1)
             ax.add_artist(imagebox)

         ax.set_title(title) # عنوان نمودار
         ax.axis("off")  #محوکردن اسم محورهای افقی و عمودی برای خلوت کردن نمودار
```

به کارگیری الگوریتم های کاهش بعد غیر خطی

```
[ ]  # Dimensionality reduction to 2D using Locally linear Embedding and Isomap
```

```
embeddings = {

    "Isomap embedding": Isomap(n_neighbors=n_neighbors, n_components=2),
    "Standard LLE embedding": LocallyLinearEmbedding(
        n_neighbors=n_neighbors, n_components=2, method="standard"
    ),
    "Modified LLE embedding": LocallyLinearEmbedding(
        n_neighbors=n_neighbors, n_components=2, method="modified"
    ),
    "Hessian LLE embedding": LocallyLinearEmbedding(
        n_neighbors=n_neighbors, n_components=2, method="hessian"
    ),
    "LTSA LLE embedding": LocallyLinearEmbedding(
        n_neighbors=n_neighbors, n_components=2, method="ltsa"
    ),

}
```

کد زیر کمک می‌کند تا عملکرد الگوریتم‌های مختلف کاهش ابعاد را از نظر **زمان اجرا** مقایسه شوند

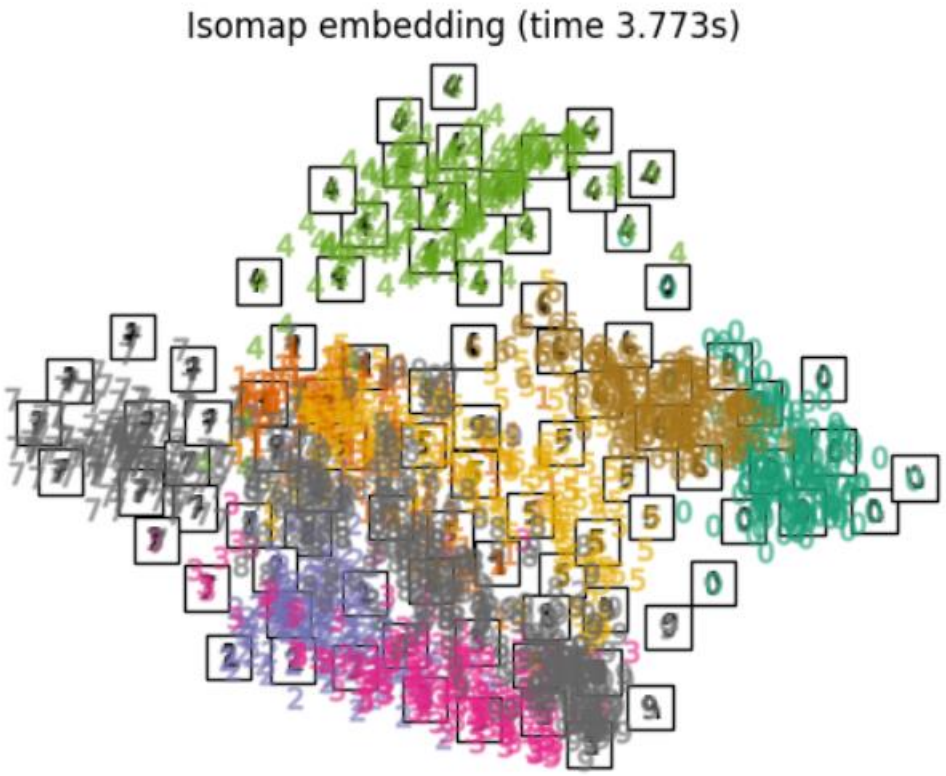
هنگامی که همه روش های مورد علاقه را روی داده های اصلی اجرا کردیم داده های پیش بینی شده و همچنین زمان محاسباتی مورد نیاز برای انجام هر طرح را ذخیره می کنیم.

```
[10] from time import time

     projections, timing = {}, {}
     for name, transformer in embeddings.items():
         if name.startswith("Linear Discriminant Analysis"): # بـررسی شرط LDA
             data = X.copy()
             data.flat[:: X.shape[1] + 1] += 0.01  # Make X invertible
         else:   # نـبـاشد LDA اگـر الـگـوریـتم
             data = X

         print(f"Computing {name}...")
         start_time = time()
         projections[name] = transformer.fit_transform(data, y)
         timing[name] = time() - start_time
```

```
Computing Isomap embedding...
Computing Standard LLE embedding...
Computing Modified LLE embedding...
Computing Hessian LLE embedding...
Computing LTSA LLE embedding...
```

رسم نمودارها

```
4s
for name in timing:
    title = f"{name} (time {timing[name]:.3f}s)"
    plot_embedding(projections[name], title)

plt.show()
```



این تصویر نتیجه **(Embedding)**داده‌های دست‌نویس ارقام با استفاده از **روش Isomap** در یک فضای دوبعدی است.

ساختار نمودار

محورها :فضای دو بُعدی کاهش‌یافته‌ای است که داده‌ها در آن نمایش داده می‌شوند.

نقاط داده‌ها :هر نقطه روی نمودار نماینده یک نمونه از رقم دست‌نویس است.

رنگ‌بندی:

هر رنگ نشان‌دهنده یک برچسب کلاس (رقم ۰ تا ۹)است.

رنگ‌ها کمک می‌کنند تا گروه‌های مختلف (مانند ارقام مشابه) در فضای جدید شناسایی شوند.

تصاویر درج‌شده:

در بعضی نقاط تصویر واقعی رقم (نمونه دست‌نویس) به عنوان یک Annotationدرج شده است.

این تصاویر کمک می‌کنند تا ببینیم هر نقطه مربوط به چه عددی است.

تحلیل داده‌ها در فضای Embedding

خوشه‌بندی:(Clustering)

نقاط مربوط به هر رقم در خوشه‌های جداگانه متمرکز شده‌اند.

مثلاً رقم ۴ در ناحیه بالا و رقم ۲ در ناحیه پایین متمرکز شده‌اند.

ساختارManifold :

داده‌ها به صورت غیرخطی در فضای دوبعدی تعبیه شده‌اند.

نقاط مشابه (مثلاً ارقام یکسان) در این فضای کاهش‌یافته نزدیک به هم قرار گرفته‌اند.

فاصله‌ها:

الگوریتم Isomap فاصله‌ها را به شکلی حفظ می‌کند که ساختار هندسی داده‌ها به درستی در بُعد پایین‌تر منعکس شود.

زمان محاسبه

زمان اجرای روش Isomap برابر ۳.۷۷۳ ثانیه است (همانطور که در عنوان نمودار نشان داده شده است).

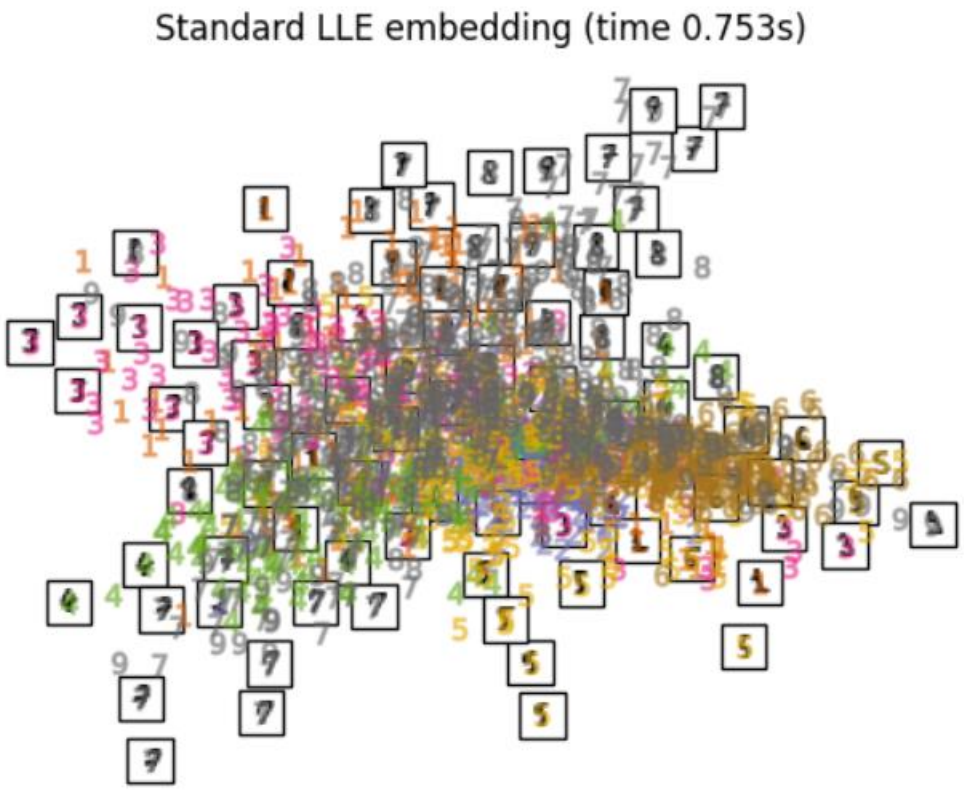
این نمودار نشان می‌دهد که الگوریتم Isomap توانسته:

داده‌های مربوط به ارقام دست‌نویس را به طور مؤثری در دو بُعد تعبیه کند.

ارقام مشابه را در فضای کاهش‌یافته در خوشه‌های مجزا مرتب کند.

ساختار اصلی داده‌ها را با حفظ فاصله‌های ژئودزیک بازسازی کند.

این روش می‌تواند برای تحلیل بصری داده‌های پیچیده و همچنین به عنوان پیش‌پردازش برای الگوریتم‌های یادگیری ماشین مورد استفاده قرار گیرد.



این تصویر مربوط به کاهش ابعاد داده‌های دست‌نویس ارقام با استفاده از **روش استاندارد** (Locally Linear Embedding) **LLE** در یک فضای **دو بُعدی** است.

روش استانداردLLE

LLE یکی از روش‌های (Manifold Learning)است که برای کاهش ابعاد غیرخطی استفاده می‌شود.

این روش سعی می‌کند ساختار محلی داده‌ها را با فرض خطی بودن در همسایگی نقاط حفظ کند.

در واقع LLE از ترکیب خطی وزن‌های همسایگی برای بازسازی داده‌ها در بُعد پایین‌تر استفاده می‌کند.

ساختار نمودار

محورها :فضای دو بُعدی کاهش یافته با استفاده از LLE.

نقاط داده‌ها :هر نقطه نماینده نمونه‌ای از ارقام دست‌نویس (۰تا ۹)است.

رنگ‌ها:

هر رنگ نشان‌دهنده یک کلاس (رقم خاص)است.

ارقام مشابه با یک رنگ نمایش داده می‌شوند.

تصاویر درج‌شده:

برخی نقاط دارای تصاویر کوچک از ارقام دست‌نویس هستند.

این تصاویر کمک می‌کنند تا مشخص شود هر نقطه مربوط به کدام رقم است.

تحلیل نمودار

خوشه‌بندی ضعیف‌تر :در مقایسه با نمودارIsomap ، نقاط مربوط به ارقام مختلف به طور کامل از هم جدا نشده‌اند.

به عنوان مثال، ارقام ۳ (صورتی)و ۱ (نارنجی) تا حدی با هم هم‌پوشانی دارند.

حفظ ساختار محلی:

نقاط داده‌ها که در فضای اصلی (داده‌های اولیه) نزدیک به هم بودند، در فضای کاهش‌یافته نیز به هم نزدیک باقی مانده‌اند.

تراکم داده‌ها:

بخش مرکزی نمودار متراکم است، که نشان می‌دهد در این بخش داده‌ها به سختی تفکیک‌پذیر هستند.

فاصله‌ها و نواحی:

نقاط داده‌ای که در حاشیه قرار دارند (مانند ارقام ۴ و ۵) بهتر تفکیک شده‌اند.

زمان محاسبه

زمان محاسبه برای روش LLE برابر ۰.۷۵۳ ثانیه است که نسبت به روش Isomap سریع‌تر است.

Standard LLE ساختار محلی داده‌ها را به خوبی حفظ می‌کند، اما در مقایسه با روش‌هایی مانند Isomap قدرت تفکیک بین کلاس‌ها (ارقام مختلف) کمتری دارد.

به دلیل وجود هم‌پوشانی در خوشه‌های مختلف، این روش برای داده‌های پیچیده ممکن است کمتر مؤثر باشد.

با این حال سرعت بالای LLE یک مزیت مهم برای کاربردهای عملی با داده‌های بزرگ محسوب می‌شود.

Isomap ساختار کلی داده‌ها و تفکیک بین کلاس‌ها را بهتر نشان می‌دهد.

LLE ساختار محلی داده‌ها را حفظ می‌کند ولی تفکیک بین کلاس‌ها ضعیف‌تر است.

به شکل دیگری هم میتوان کد را اجرا کرد:

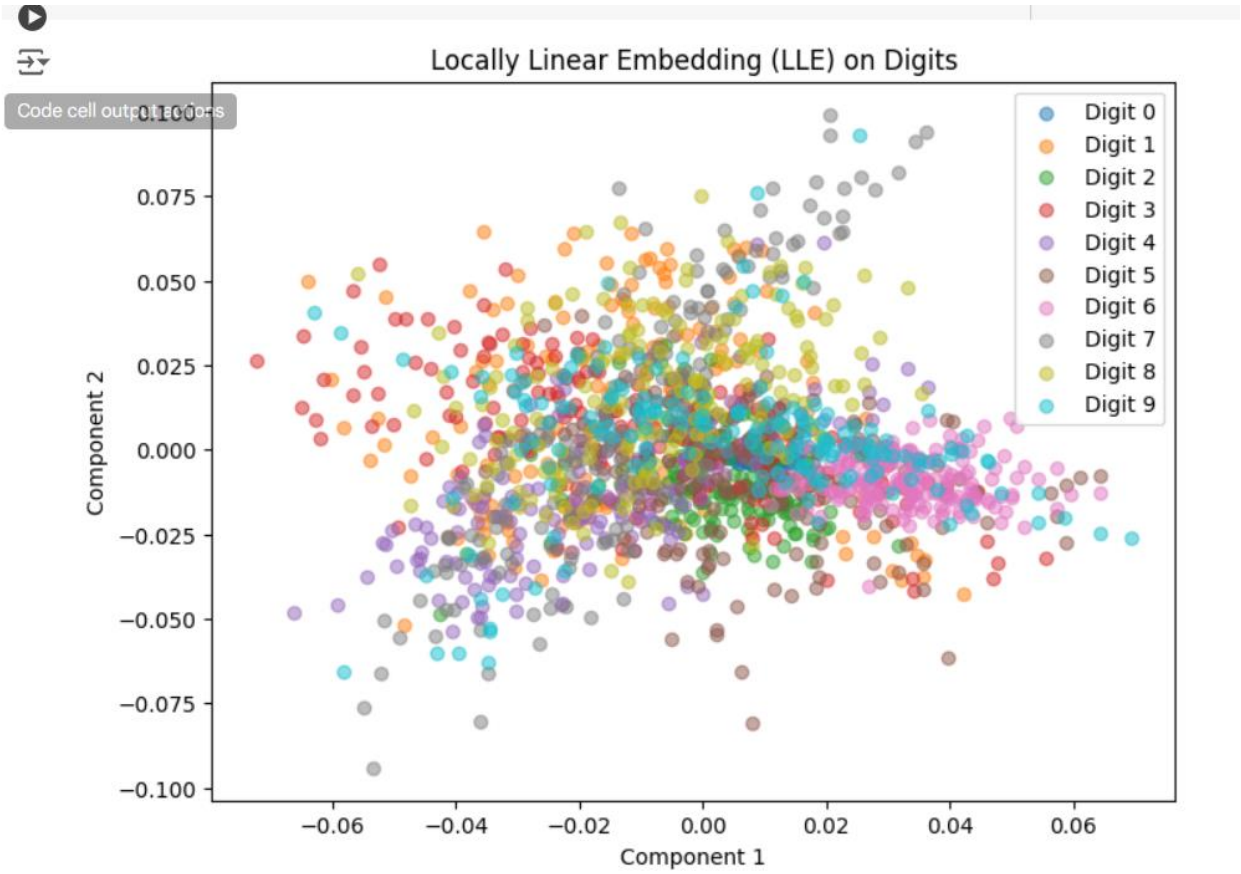
```
# Dimensionality reduction to 2D using Locally linear Embedding and Isomap
```

```
[13] lle=LocallyLinearEmbedding(n_components=2,n_neighbors=n_neighbors,method='standard')
     print("Running LLE...")
     X_lle = lle.fit_transform(X)



     isomap=Isomap(n_components=2,n_neighbors=n_neighbors)
     print("Running Isomap...")
     X_isomap=isomap.fit_transform(X)
```

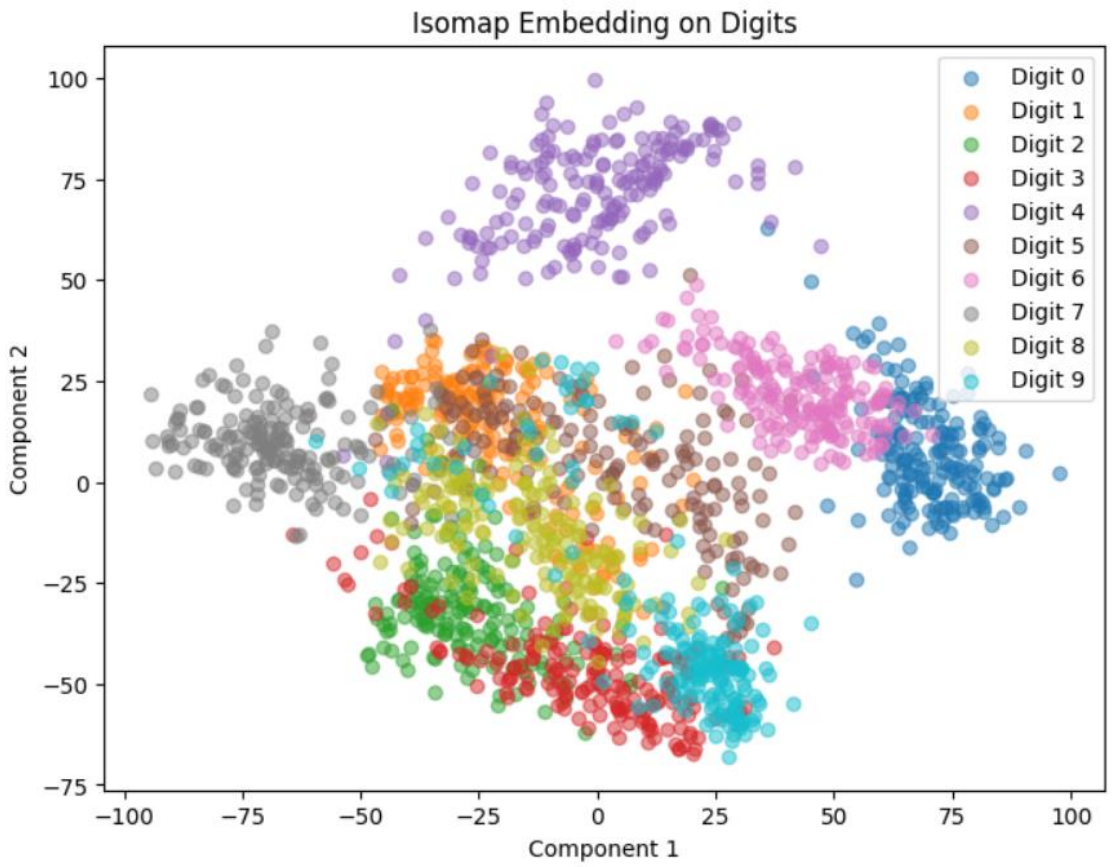
```
Running LLE...
Running Isomap...
```

رسم نمودارها

```
[14] #رسم نتایج

     #تابع کمکی برای رسم نمودار

     def plot_embedding(X, y, title):
         plt.figure(figsize=(8, 6))
         for i in range(10):  # (۰ تا ۹) برای هر کلاس
             plt.scatter(X[y == i, 0], X[y == i, 1], label=f"Digit {i}", alpha=0.5)
         plt.legend()
         plt.title(title)
         plt.xlabel("Component 1")
         plt.ylabel("Component 2")
         plt.show()

     # رسم نتایجLLE
     plot_embedding(X_lle, y, "Locally Linear Embedding (LLE) on Digits")

     #رسم نتایج Isomap
     plot_embedding(X_isomap, y, "Isomap Embedding on Digits")
```


Locally Linear Embedding (LLE) on Digits
Digit 0
Digit 1
Digit 2
Digit 3
Digit 4
Digit 5
Digit 6
Digit 7
Digit 8
Digit 9
Component 2
Component 1
0.100
0.075
0.050
0.025
0.000
−0.025
−0.050
−0.075
−0.100
−0.06
−0.04
−0.02
0.00
0.02
0.04
0.06

[14]


Isomap Embedding on Digits
Digit 0
Digit 1
Digit 2
Digit 3
Digit 4
Digit 5
Digit 6
Digit 7
Digit 8
Digit 9
Component 2
Component 1
100
75
50
25
0
−25
−50
−75
−100
−75
−50
−25
0
25
50
75
100