Office Hour

UMAP

Markus Hohle

University California, Berkeley

Machine Learning Algorithms

MSSE 277B, 3 Units

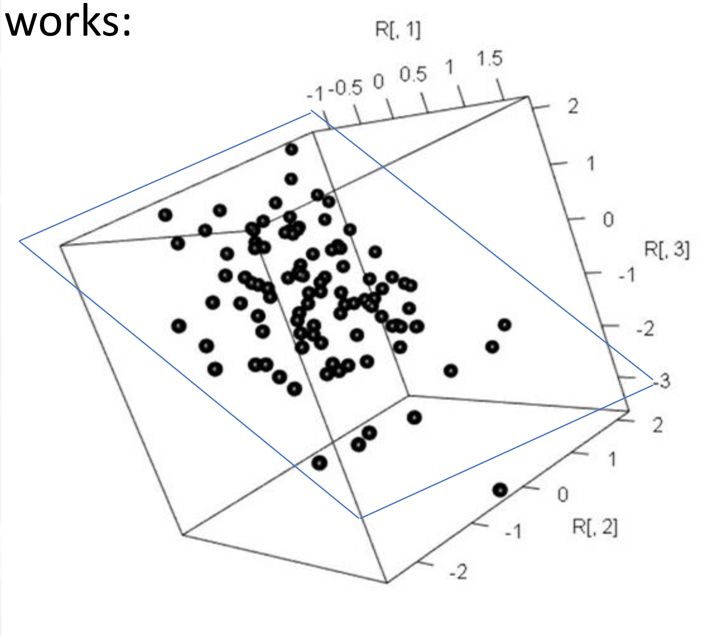
Fall 2024

- PCA is a very common tool for dimension reduction
- simple and fast
- well interpretable (eigenvalue spectrum of the covariance matrix)

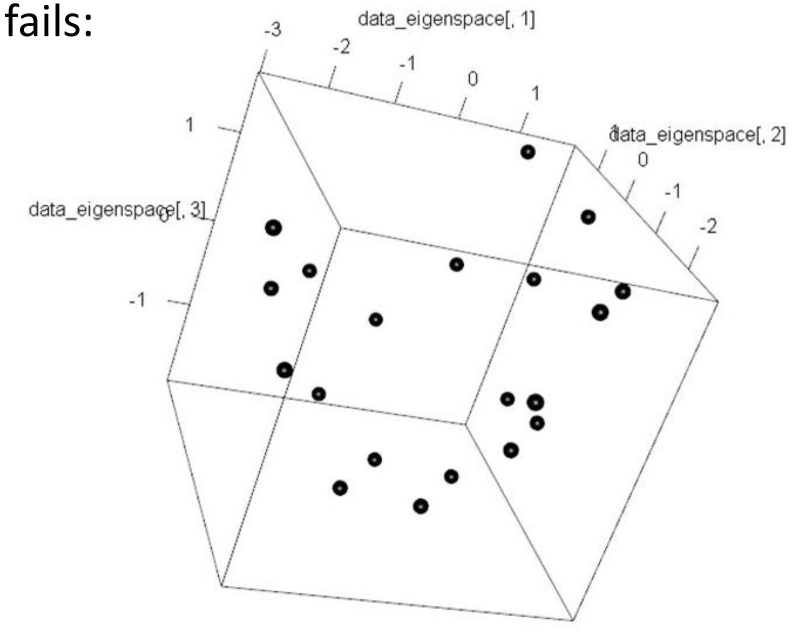
but...

- only on flat manifolds → fails if data clouds have unusual shapes

works:

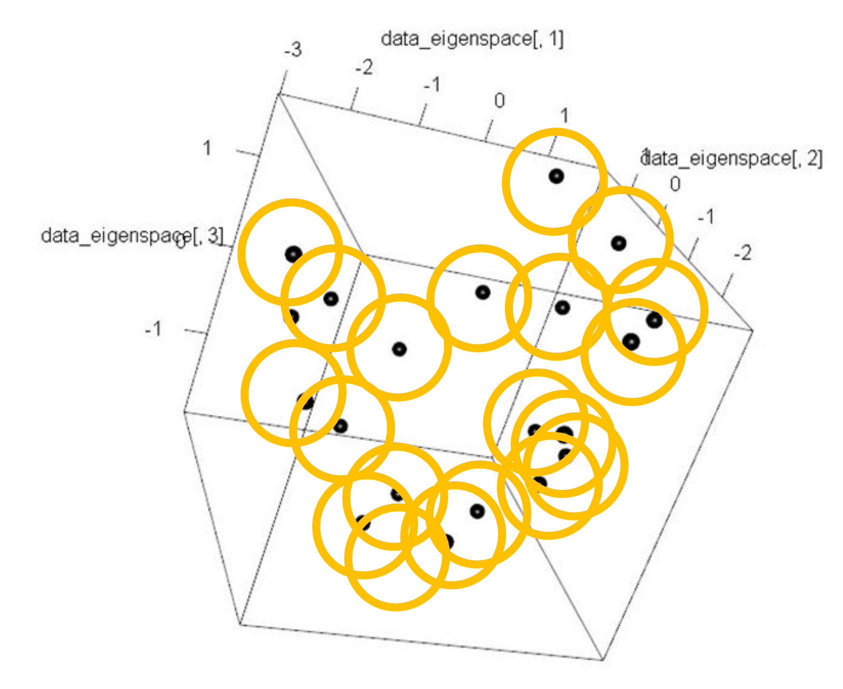


fails:



Is there a tool for more complex data cloud shapes?

time for **U**niform **M**anifold **A**pproximation and **P**rojection



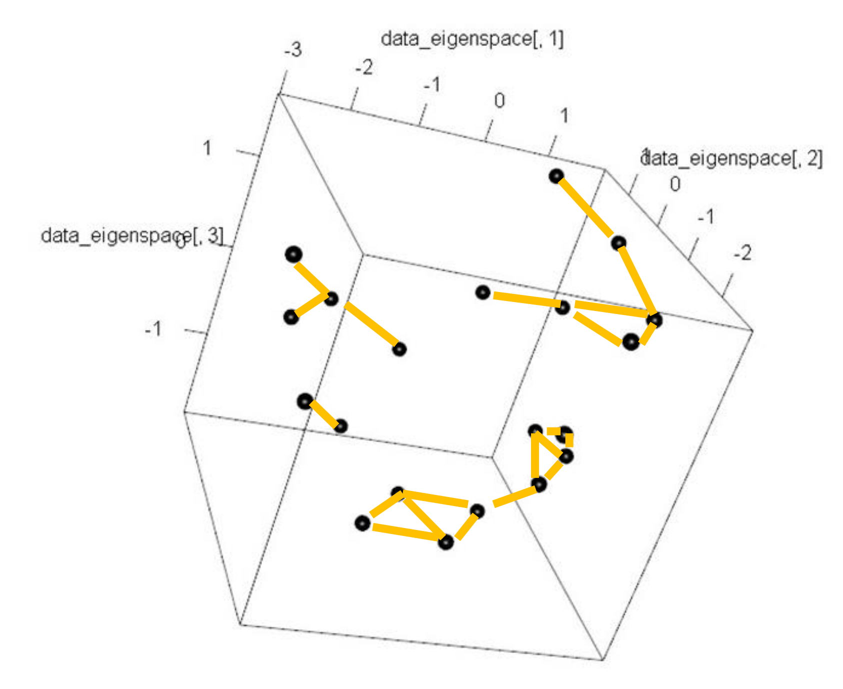
- defining a *unit* distance

...and so on

Machine Learning Algorithms:

Is there a tool for more complex data cloud shapes?

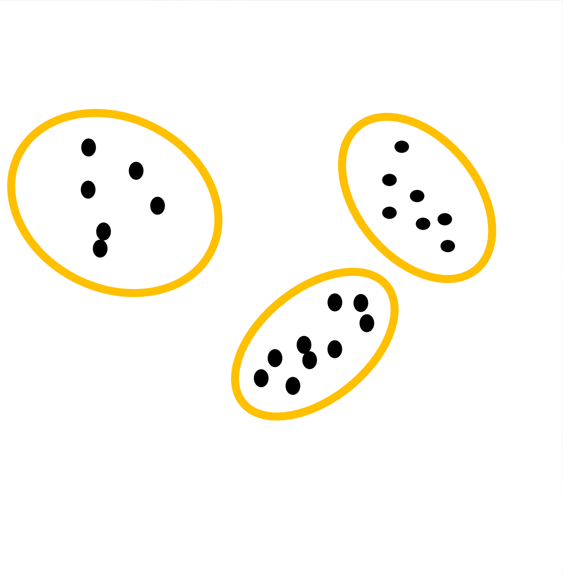
<u>time for **U**niform **M**anifold **A**pproximation and **P**rojection</u>



- defining a *unit* distance
- next neighbors and connectivity
- interpreting the result as graphs
- transferring into eigen coordinates (*like* PCA, graph Laplacian)
- note: it all happens in N-D of course

Is there a tool for more complex data cloud shapes?

time for **U**niform **M**anifold **A**pproximation and **P**rojection



- defining a *unit* distance
- next neighbors and connectivity
- interpreting the result as graphs
- transferring into eigen coordinates (*like* PCA, graph Laplacian)
- note: it all happens in N-D of course
- projection on 2D manifold

Machine Learning Algorithms:

time for **U**niform **M**anifold **A**pproximation and **P**rojection

Now we are ready for the UMAP analysis:

```
#pip install umap-learn
import umap.umap_ as umap
```

the UMAP package is not part of sklearn

```
newXY = umap.UMAP().fit_transform(XS)
```

performing the actual UMAP transformation

time for **U**niform **M**anifold **A**pproximation and **P**rojection

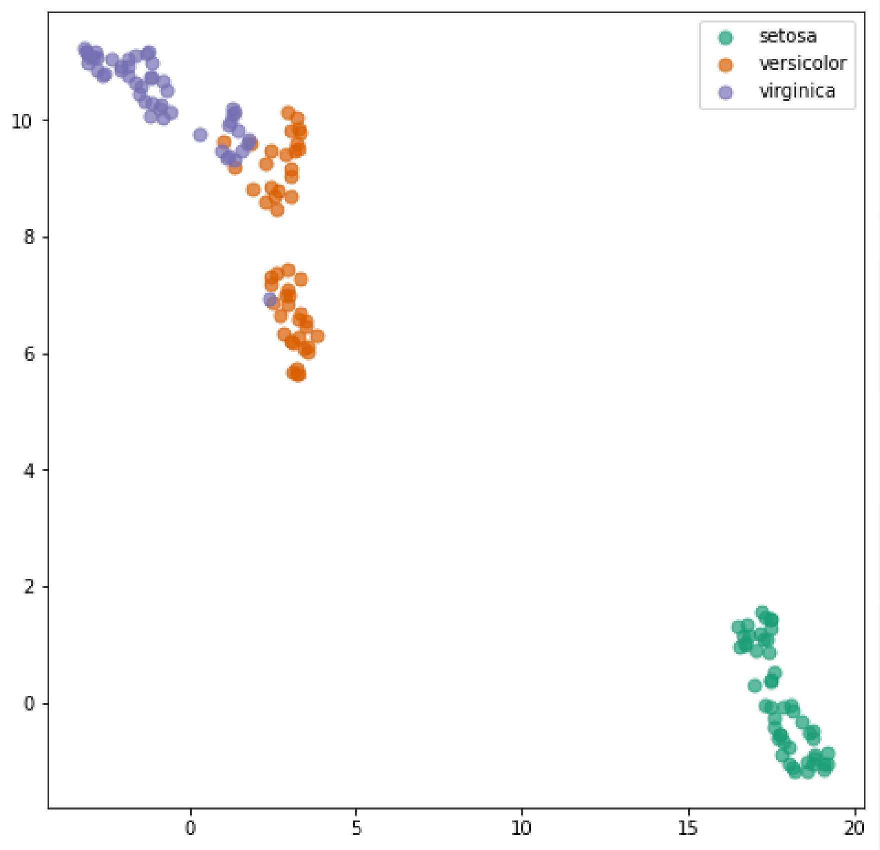
```
import numpy as np
import matplotlib.pyplot as plt
#pip install umap-learn
import umap.umap_ as umap
from sklearn import datasets
from draw_umap import draw_umap


iris   = datasets.load_iris()
Labels = iris.target_names
X      = iris.data


Color  = ["#1B9E77", "#D95F02", "#7570B3"]
```

time for **U**niform **M**anifold **A**pproximation and **P**rojection

```
iris   = datasets.load_iris()
Labels = iris.target_names
X      = iris.data

Color  = ["#1B9E77", "#D95F02", "#7570B3"]


newXY  = umap.UMAP().fit_transform(X)


fig, ax = plt.subplots(figsize = (8,8))
i = 0
for species, color in zip(Labels, Color):
    idxs = np.arange(0,50) + 50*i
    i += 1
    ax.scatter(newXY[idxs, 0], newXY[idxs, 1], label = species,\
                s = 50, color = color, alpha = 0.7)
ax.legend()
plt.show()
```

time for **U**niform **M**anifold **A**pproximation and **P**rojection

```
iris   = datasets.load_iris()
Labels = iris.target_names
X      = iris.data
Color  = ["#1B9E77", "#D95F02", "#7570B3"]

newXY  = umap.UMAP().fit_transform(X)

fig, ax = plt.subplots(figsize = (8,8))
i = 0
for species, color in zip(Labels, Color):
    idxs = np.arange(0,50) + 50*i
    i += 1
    ax.scatter(newXY[idxs, 0], newXY[idxs, 1], label = species,\
                    s = 50, color = color, alpha = 0.7)
ax.legend()
plt.show()
```



time for **U**niform **M**anifold **A**pproximation and **P**rojection

```
for i in range(10):
    nn = i + 5
    draw_umap(X, Color, Labels, n_neighbors = nn, title = str(nn) + ' neighbors')

for i in range(10):
    dist = i/10
    draw_umap(X, Color, Labels, min_dist = dist, title = 'dist = ' + str(dist))

for i in range(3):
    i += 1
    draw_umap(X, Color, Labels, n_components = i, title = str(i) + ' components')
```