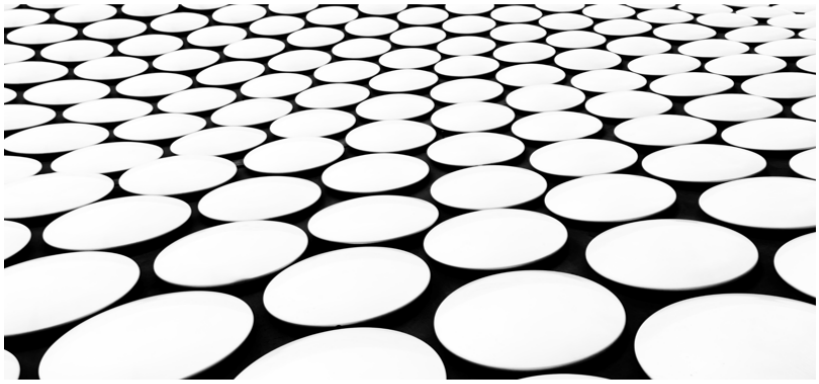


SCHOOL OF AI
ITALIA

MACHINE LEARNING AVANZATO DA ZERO

ANTONIO DI CECCO - SCHOOL OF AI

Unsupervised Learning

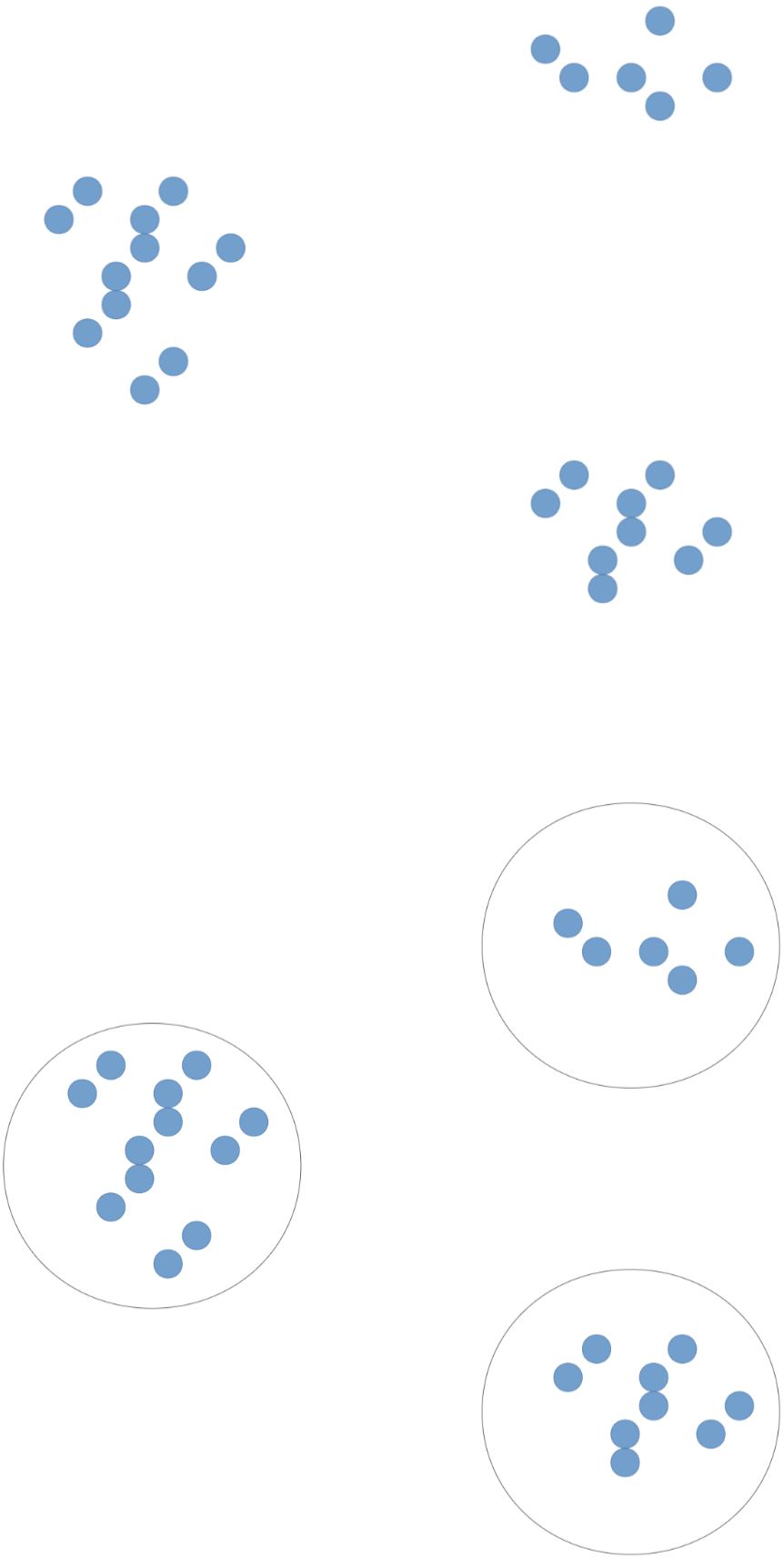
- è il vero apprendimento?

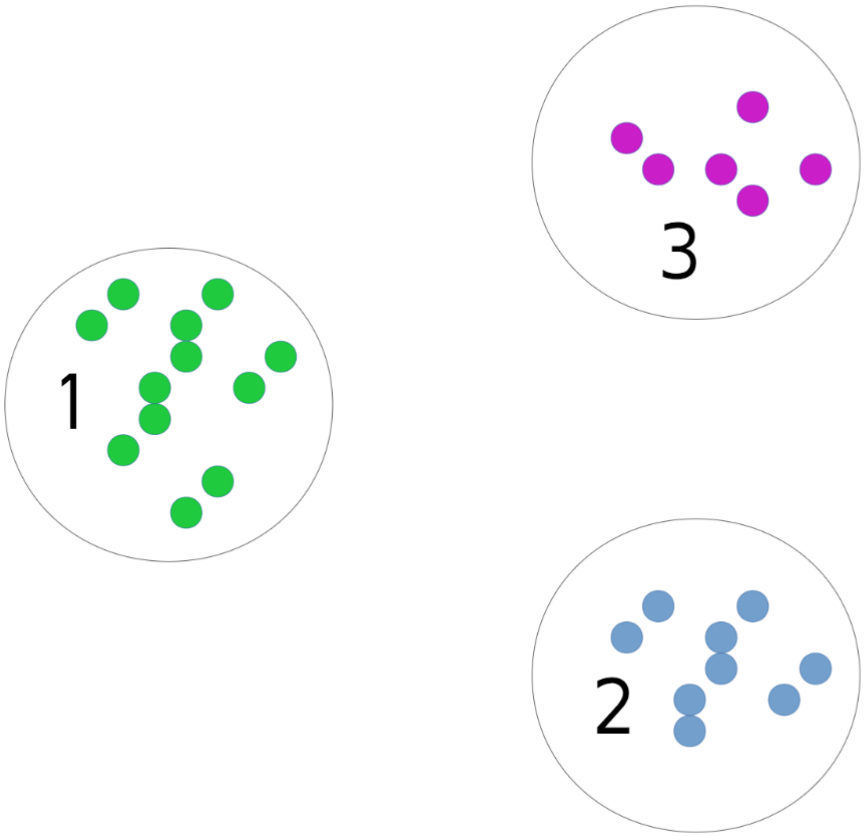
  X -> y y = f(X)

  X -> X f(X, y) = 0

- quante immagini deve vedere un bambino per distinguere cani e gatti?

Clustering





- Divide i dati in gruppi (i cluster)
- Punti nello stesso cluster dovrebbero essere “simili”
- Punti in differenti cluster dovrebbero essere “differenti”

Obiettivi del clustering

Data exploration

- Are there coherent groups ?
- How many groups are there ?

Data Partitioning

- Divide data by group before further processing

Unsupervised feature extraction

- Derive features from clusters or cluster distances

Evaluation and parameter tuning

- Quantitative measures of limited use
- Usually qualitative measures used
- Best: downstream tasks
- What is your goal?

Clustering Algorithms

K-Means

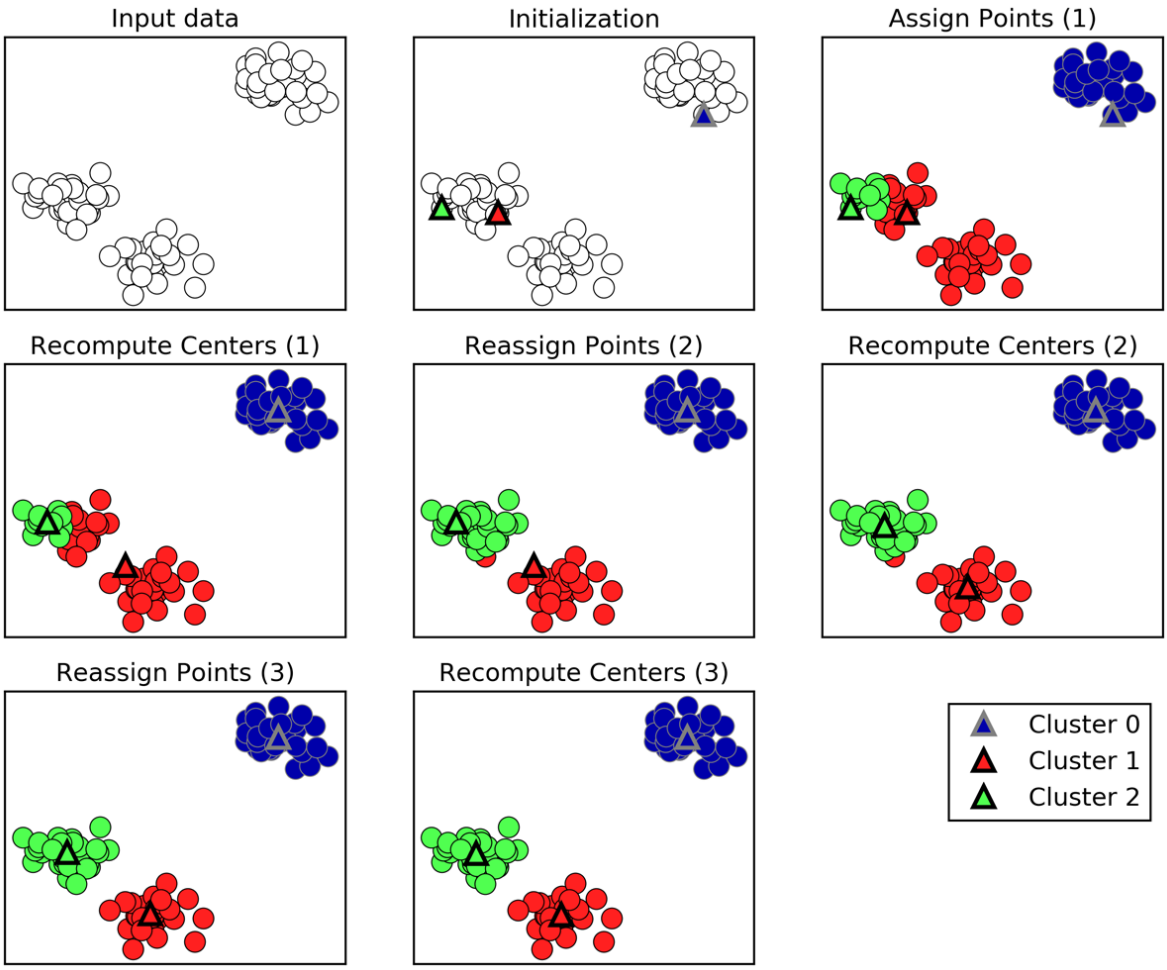
Objective function for K-Means

$$\min_{\mathbf{c}_j \in \mathbf{R}^p, j=1,..,k} \sum_i ||\mathbf{x}_i - \mathbf{c}_{x_i}||^2$$

c_j è il centroide

K-Means algorithm (Lloyd's)

- SCELGO UN NUMERO DI CLUSTER k.
- Pick k random points as “cluster center”
- While cluster centers change:
    1. Assign each data point to it’s closest cluster center
    2. Recompute cluster centers as the mean of the assigned points.



K-Means API

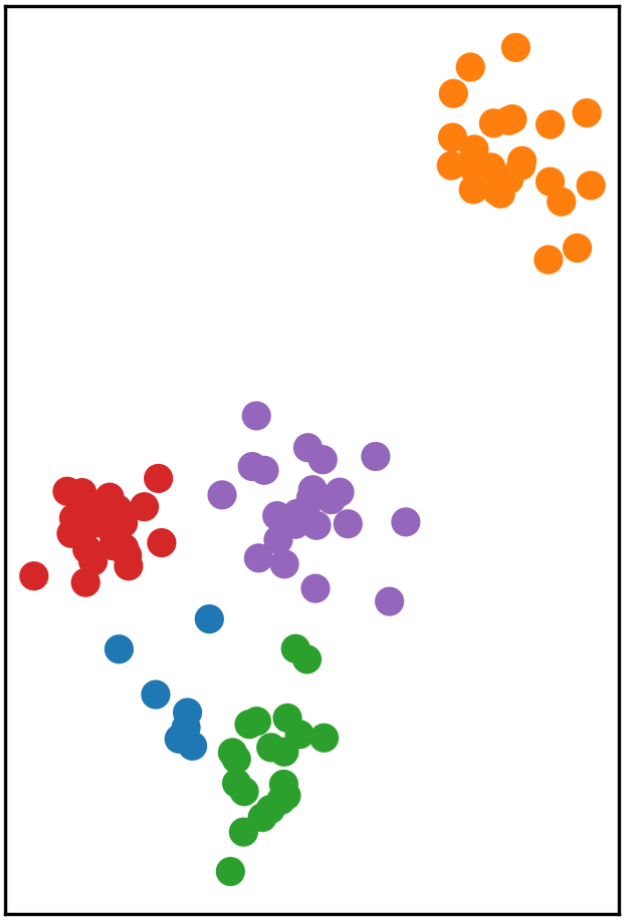
```
X, y = make_blobs(centers=4, random_state=1)

km = KMeans(n_clusters=5, random_state=0)
km.fit(X)
print(km.cluster_centers_.shape)
print(km.labels_.shape)
```

```
(5, 2)
(100,)
```

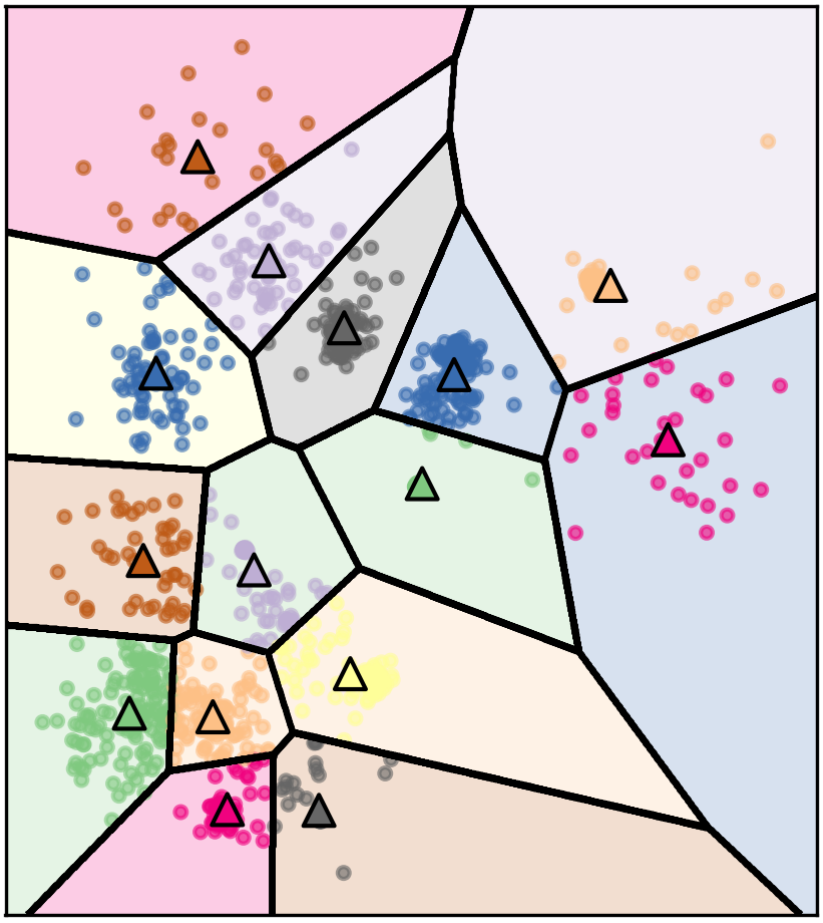
```
# predict is the same as labels_ on training data
# but can be applied to new data
print(km.predict(X).shape)
```

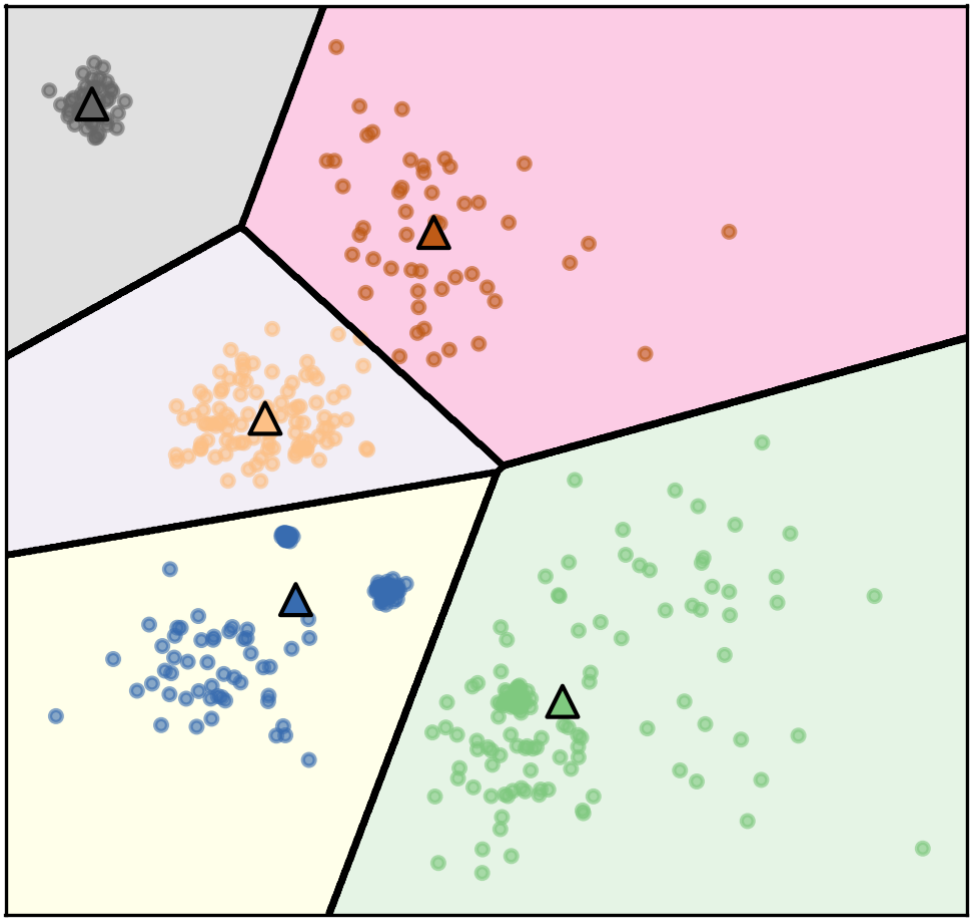
```
(100,)
```

```
plt.scatter(X[:, 0], X[:, 1], c=km.labels_)
```



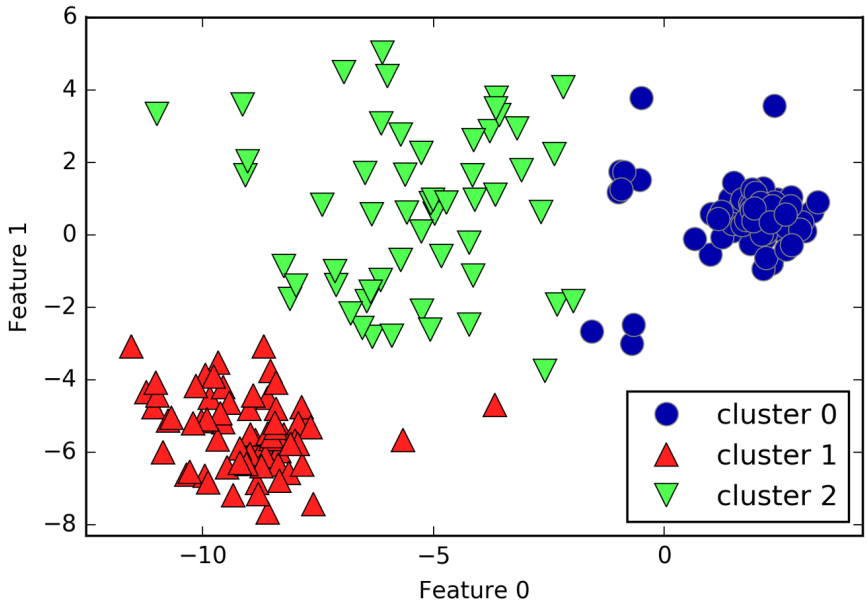
Restriction of Cluster Shapes



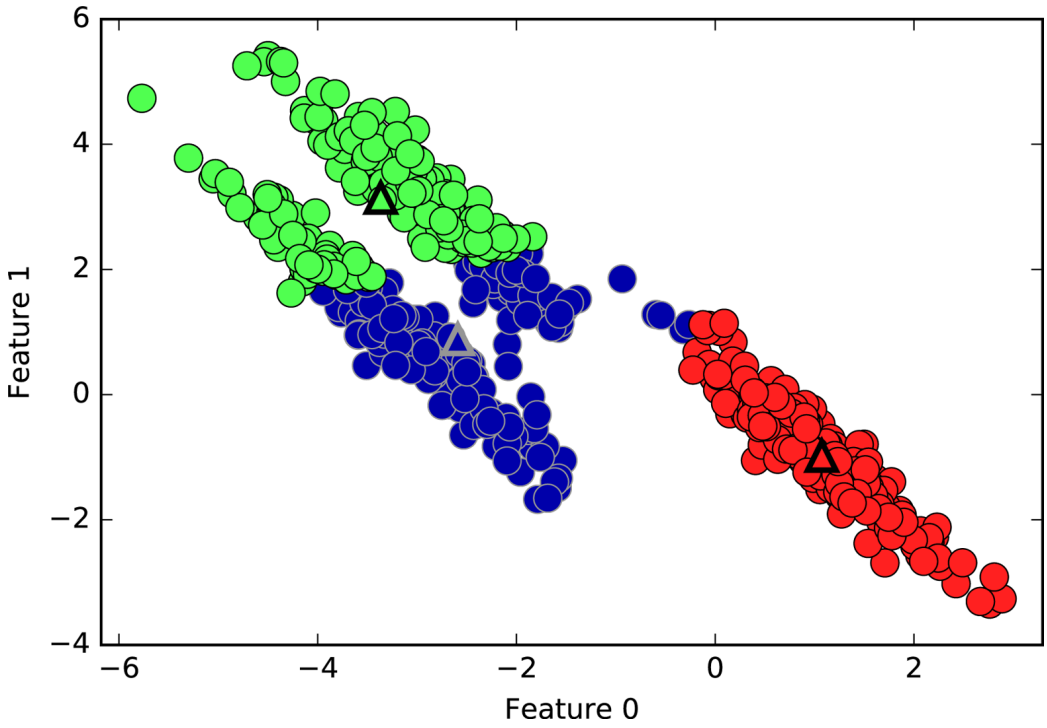


Clusters are Voronoi-diagrams of centers

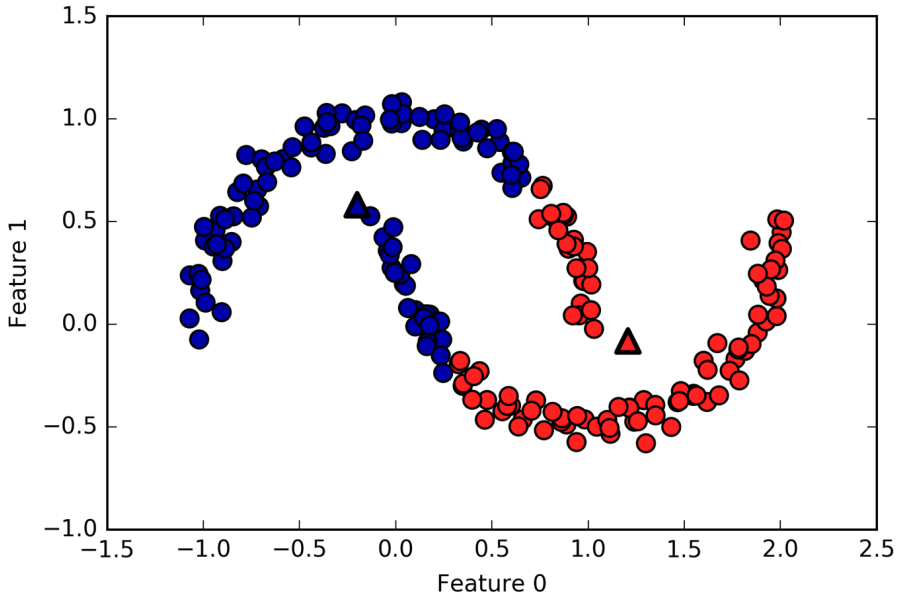
Limitations of K-Means



Cluster boundaries equidistant to centers: CONVESSITA'



Can't model covariances well



Only simple cluster shapes

Computational Properties

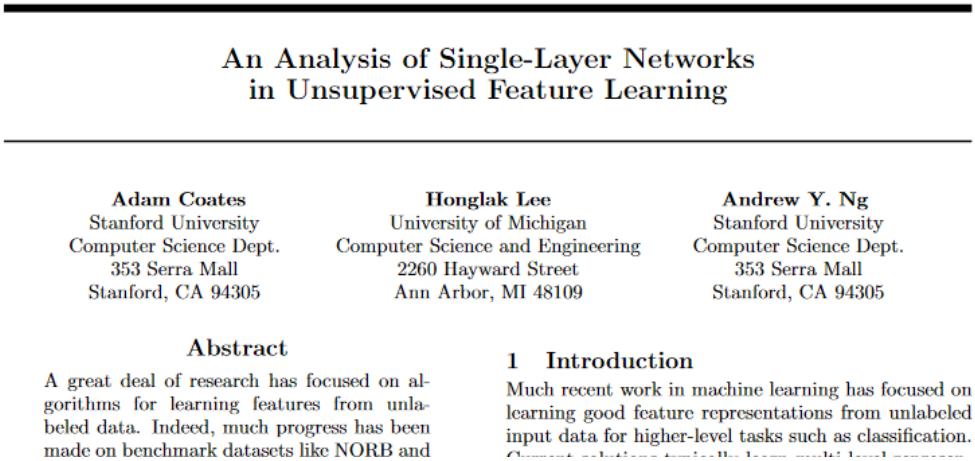
- Naive implementation (Lloyd's):
  - n_cluster * n_samples distance calculations per iteration
- Fast "exact" algorithms:
  - Elkan's, Ying-Yang, ...
- Approximate algorithms:
  - minibatch K-Means

Initialization

- Random centers fast.
- K-means++ (default): Greedily add “furthest way” point
- By default K-means in sklearn does 10 random restarts with different initializations.
- K-means++ initialization may take much longer than clustering.

Feature Extraction using K-Means

- Cluster membership → categorical feature
- Cluster distanced → continuous feature
- Examples:
  - Partitioning low-dimensional space (similar to using basis functions)
  - Extracting features from high-dimensional spaces, for image patches, see http://ai.stanford.edu/~acoates/papers/coatesleeng_aistats_2011.pdf



An Analysis of Single-Layer Networks in Unsupervised Feature Learning

Adam Coates
Stanford University
Computer Science Dept.
353 Serra Mall
Stanford, CA 94305

Honglak Lee
University of Michigan
Computer Science and Engineering
2260 Hayward Street
Ann Arbor, MI 48109

Andrew Y. Ng
Stanford University
Computer Science Dept.
353 Serra Mall
Stanford, CA 94305

Abstract

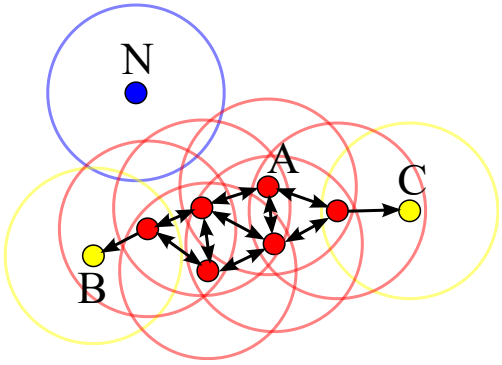
A great deal of research has focused on algorithms for learning features from unlabeled data. Indeed, much progress has been made on benchmark datasets like NORB and

1 Introduction

Much recent work in machine learning has focused on learning good feature representations from unlabeled input data for higher-level tasks such as classification.

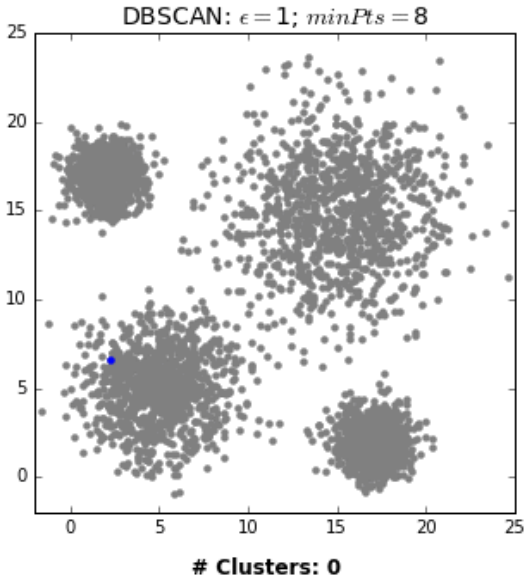
DBSCAN

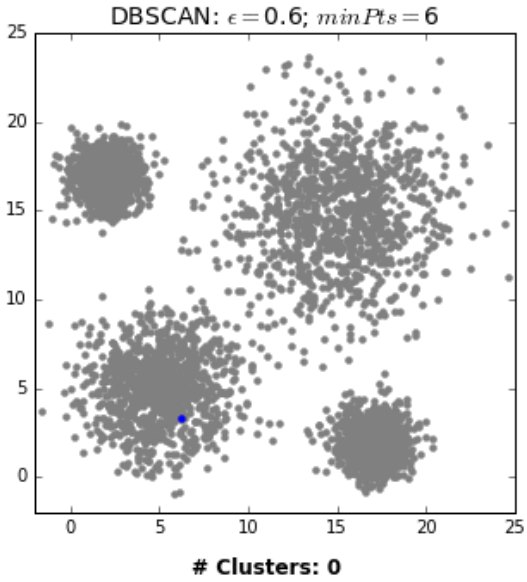
Algorithm



- eps: neighborhood radius
- min_samples: 4

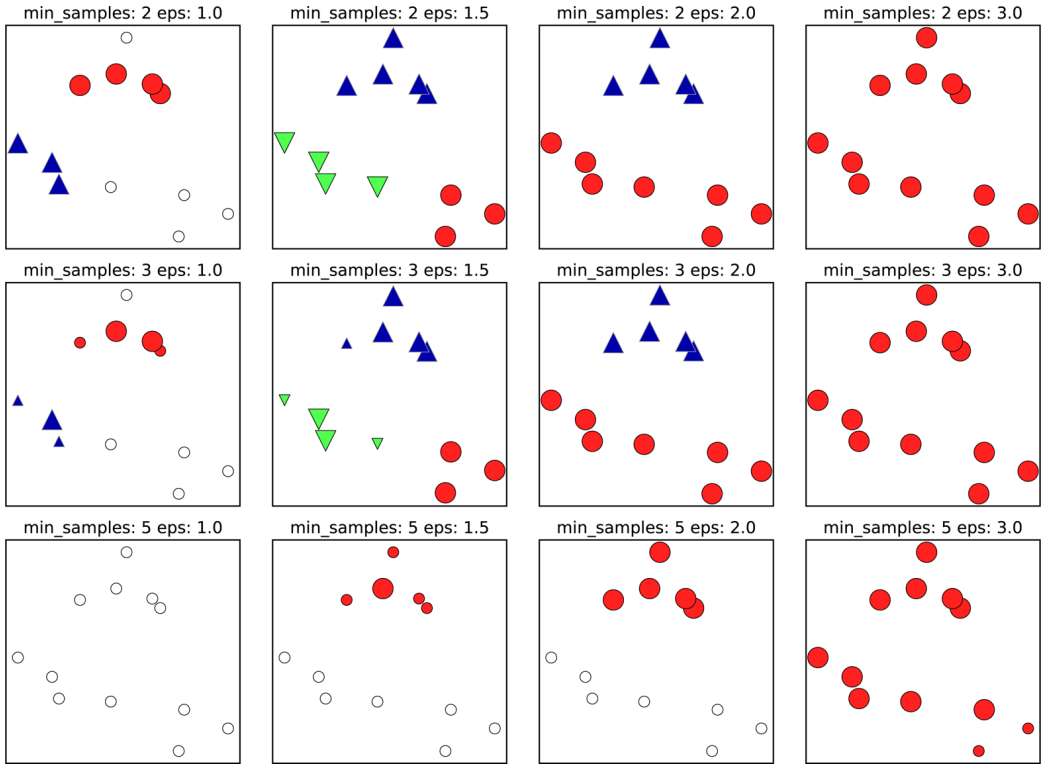
- A: Core
- B, C: not core
- N: noise





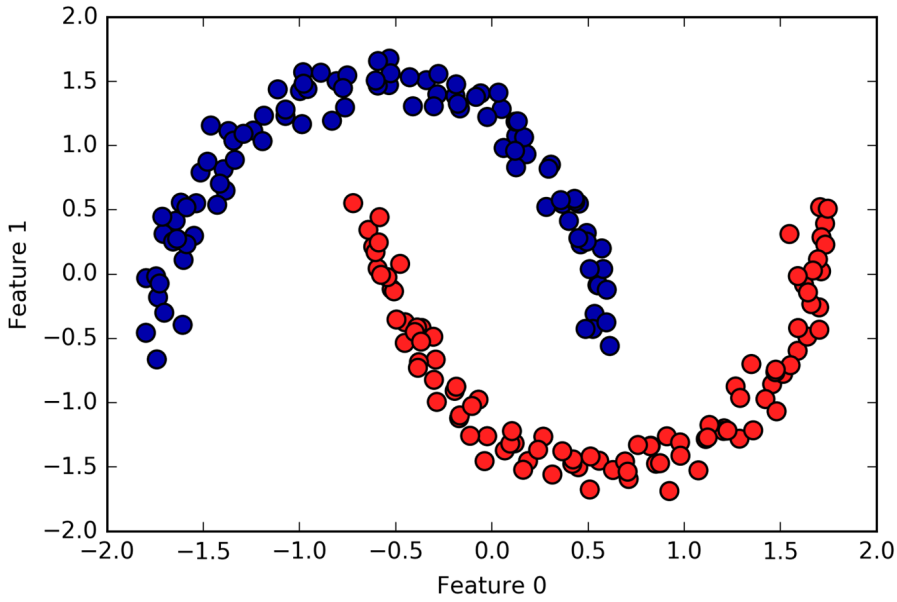
by David Sheehan dashee87.github.io

Illustration of Parameters



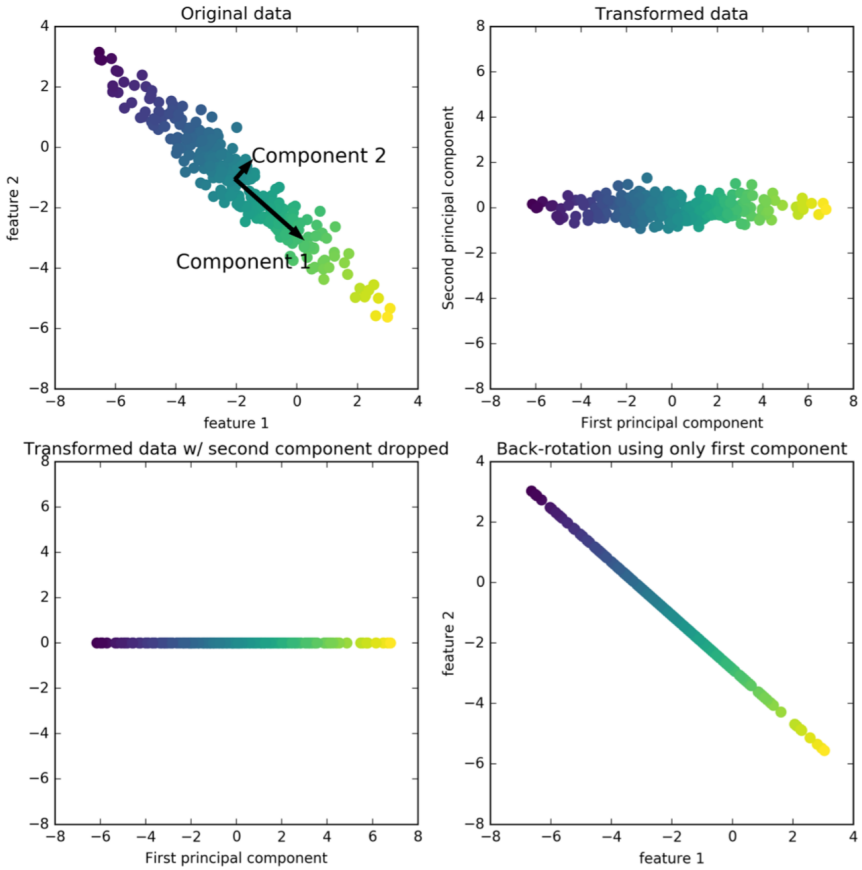
Pros and Cons

- Pro: Can learn arbitrary cluster shapes
- Pro: Can detect outliers
- Con: Needs two (non-obvious?) parameters to adjust
- Improved version: HDBSCAN
- Kmeans + DBSCAN (mio)



Riduzione dimensionale

Principal Component Analysis



$$\min_{X', \mathrm{rank}(X')=r} \|X - X'\|$$

equivalente a

$$\max_{u_1 \in R^p, \|u_1\|=1} \mathrm{var}(Xu_1)$$

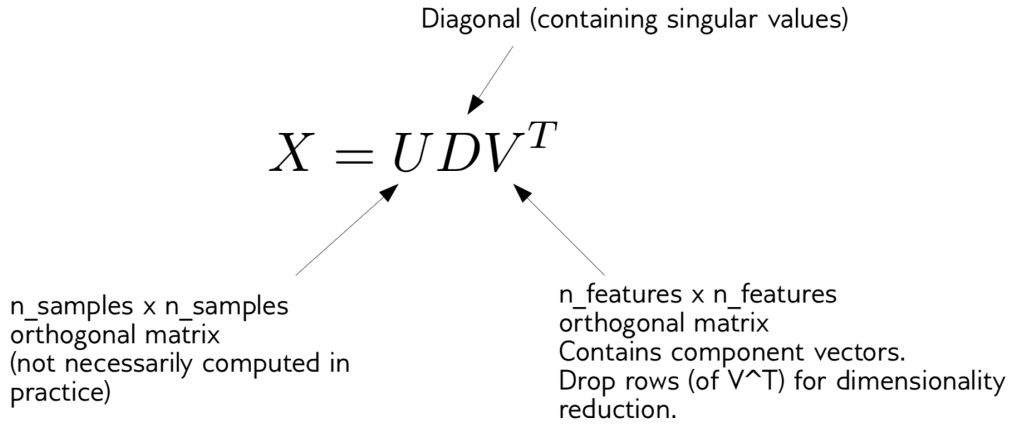
$$\max_{u_1 \in R^p, \|u_1\|=1} u_1^T \mathrm{cov}(X) u_1$$

PCA Computation

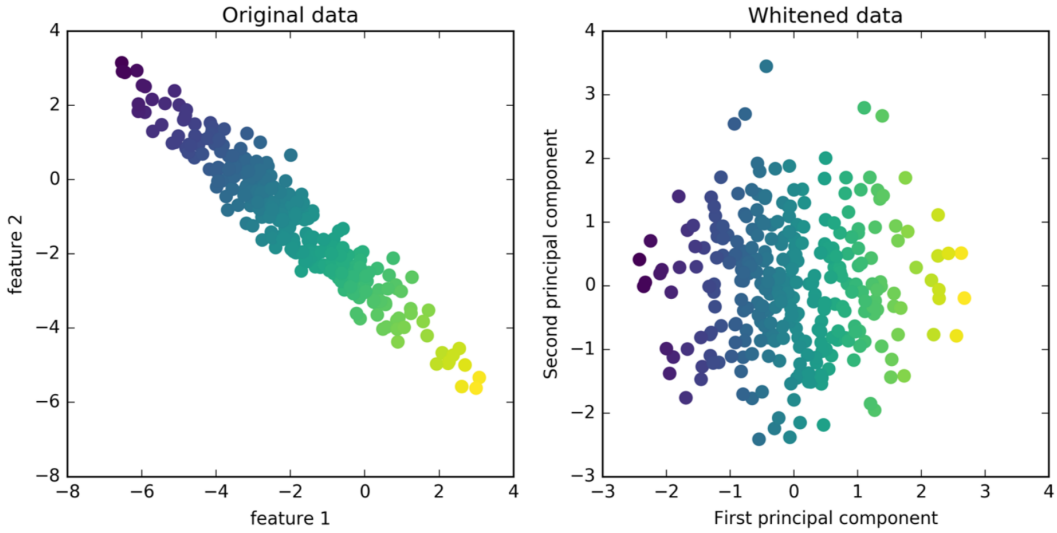
Center X (subtract mean).

In practice: Also scale to unit variance.

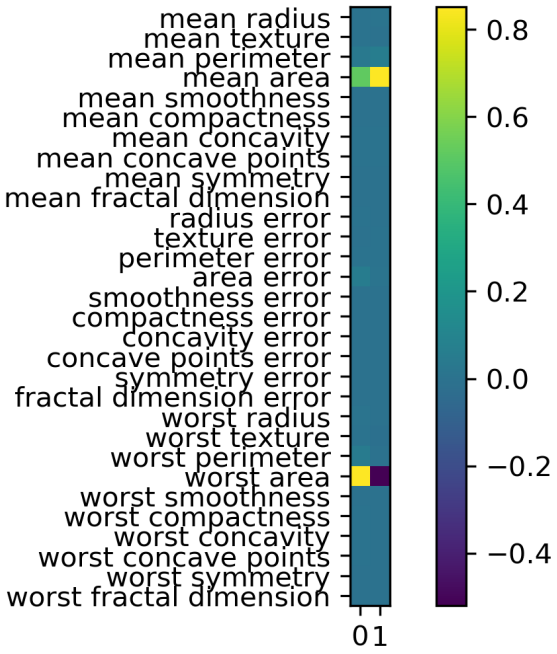
Compute **singular value decomposition**:

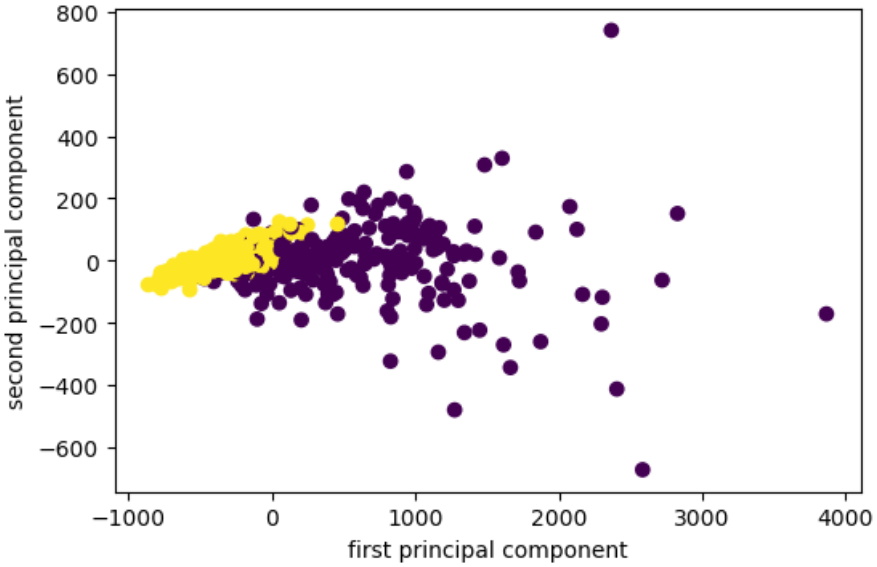


Whitening

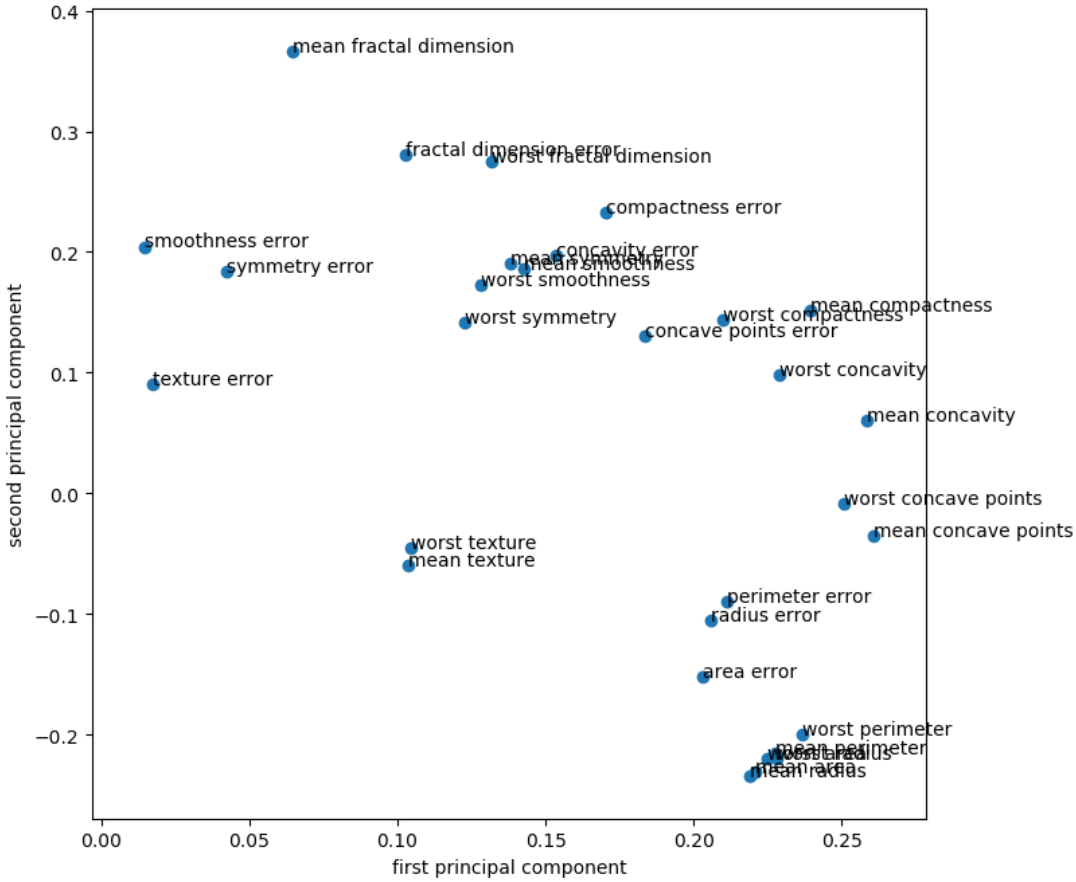



```
from sklearn.decomposition import PCA
print(cancer.data.shape)
pca = PCA(n_components=2)
X_pca = pca.fit_transform(cancer.data)
print(X_pca.shape)
plt.scatter(X_pca[:, 0], X_pca[:, 1], c=cancer.target)
plt.xlabel("first principal component")
plt.ylabel("second principal component")
components = pca.components_
plt.imshow(components.T)
plt.yticks(range(len(cancer.feature_names)), cancer.feature_names)
plt.colorbar()
```






```
components = pca_scaled.named_steps['pca'].components_
plt.imshow(components.T)
plt.yticks(range(len(cancer.feature_names)), cancer.feature_names)
plt.colorbar()
```



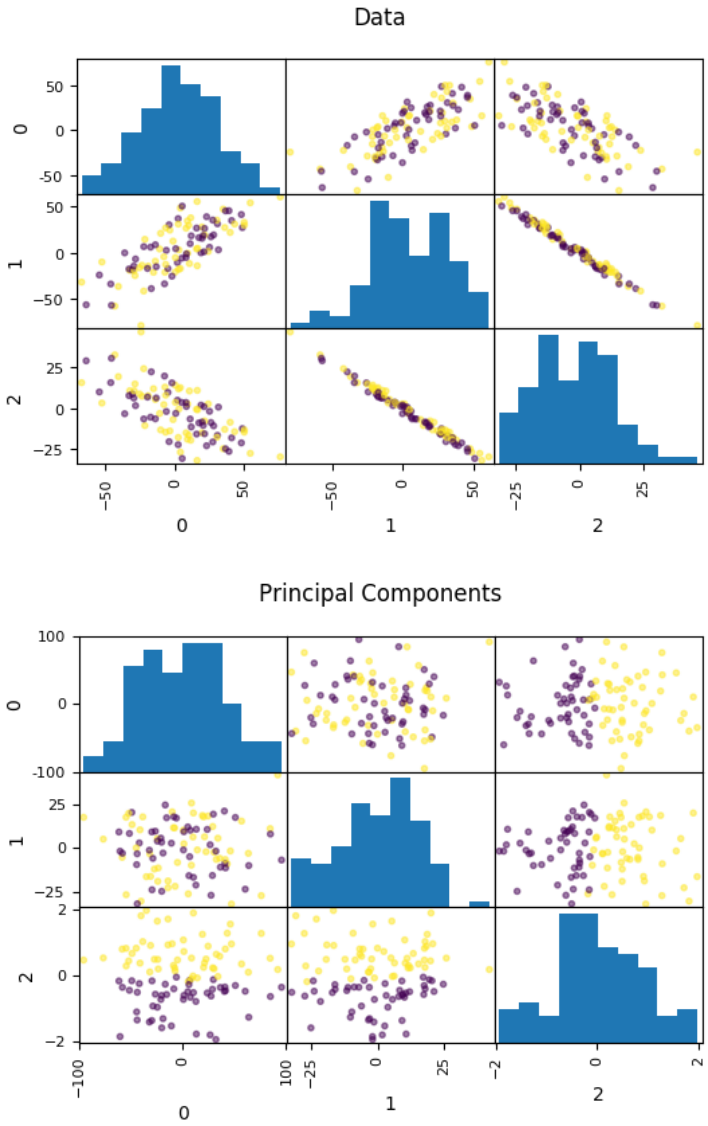
```
from sklearn.linear_model import LogisticRegression
X_train, X_test, y_train, y_test = train_test_split(cancer.data, cancer.target,
stratify=cancer.target, random_state=0)
lr = LogisticRegression(C=10000).fit(X_train, y_train)
print(lr.score(X_train, y_train))
print(lr.score(X_test, y_test))
```

```
0.993
0.944
```

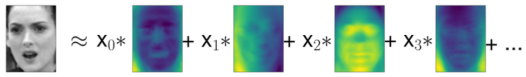
```
pca_lr = make_pipeline(StandardScaler(), PCA(n_components=2),
LogisticRegression(C=10000))
pca_lr.fit(X_train, y_train)
print(pca_lr.score(X_train, y_train))
print(pca_lr.score(X_test, y_test))
```

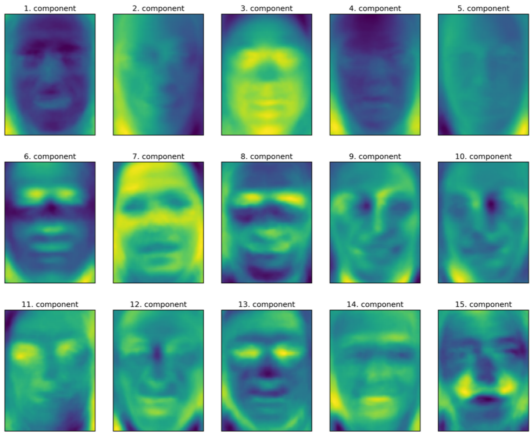
```
0.961
0.923
```

PCA is Unsupervised!



PCA for feature extraction



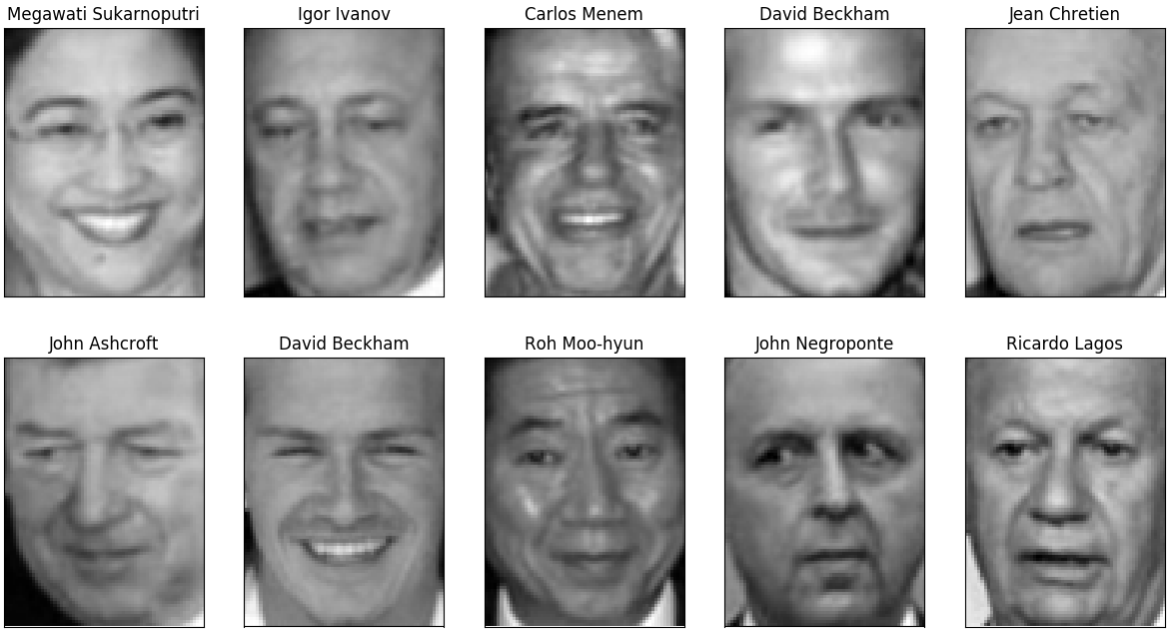


Remember:
Signs don't mean anything!

Reconstruction

PCA for outlier detection

```
pca = PCA(n_components=100).fit(X_train)
reconstruction_errors = np.sum((X_test -
pca.inverse_transform(pca.transform(X_test))) ** 2, axis=1)
```



Ricostruzioni migliori

ricostruzioni peggiori

Manifold Learning (in generale non lineare)

- For visualization only
- Axes don’t correspond to anything in the input space.
- Often can’t transform new data.
- Pretty pictures!

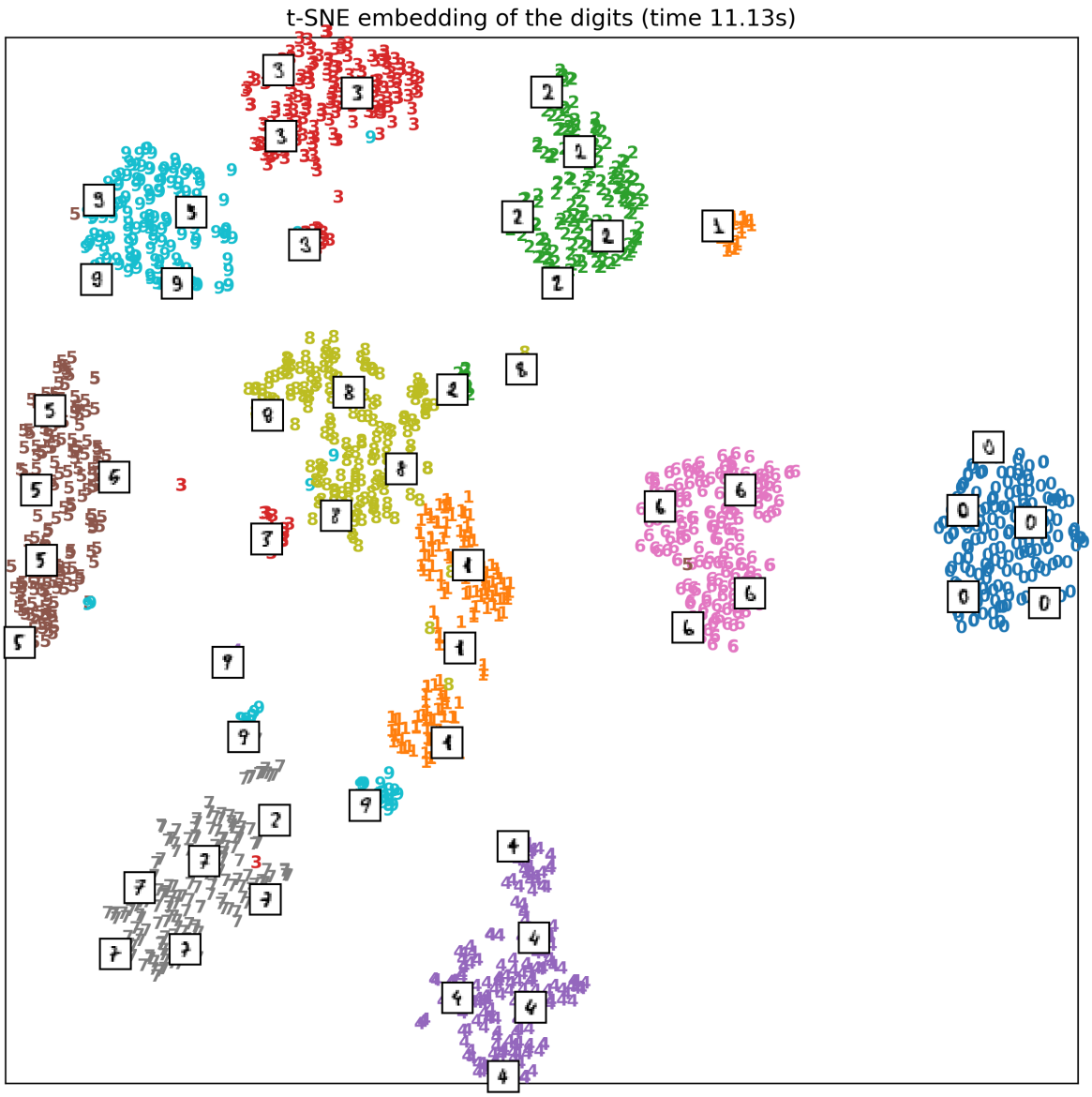
t-SNE

$$p_{j|i} = \frac{\exp(-\|\mathbf{x}_i - \mathbf{x}_j\|^2/2\sigma_i^2)}{\sum_{k\neq i}\exp(-\|\mathbf{x}_i - \mathbf{x}_k\|^2/2\sigma_i^2)}$$

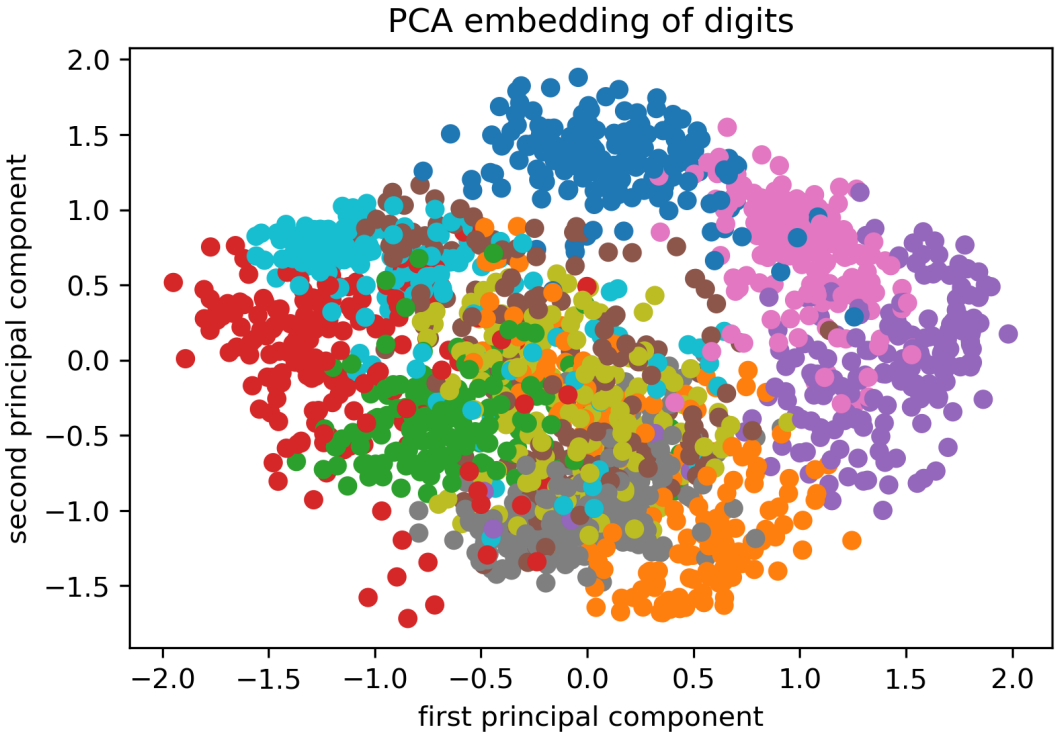
$$p_{ij} = \frac{p_{j|i} + p_{i|j}}{2N}$$

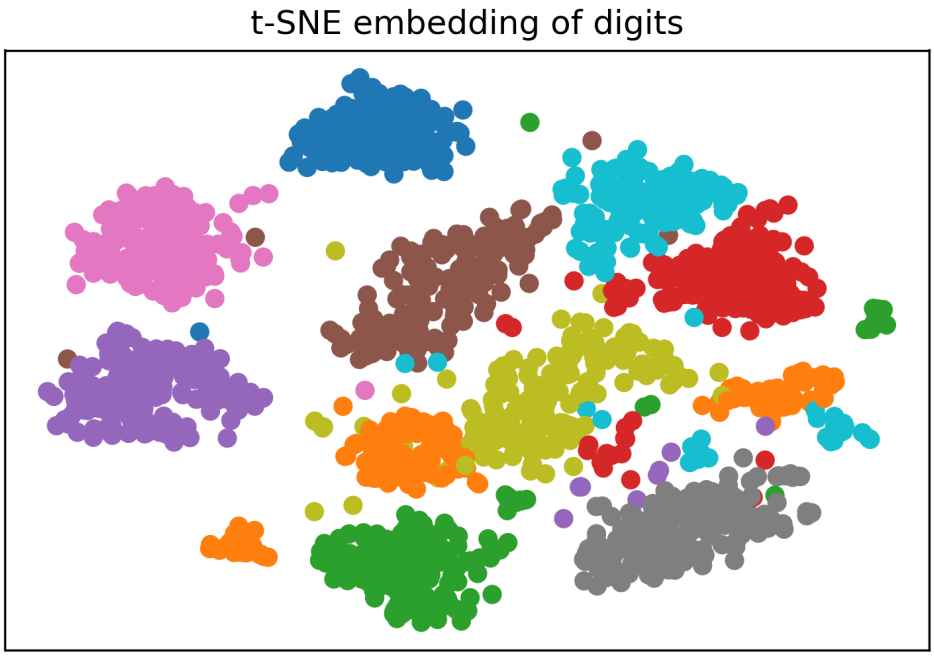
$$q_{ij} = \frac{(1 + \|\mathbf{y}_i - \mathbf{y}_j\|^2)^{-1}}{\sum_{k\neq i}(1 + \|\mathbf{y}_i - \mathbf{y}_k\|^2)^{-1}}$$

$$KL(P||Q) = \sum_{i\neq j} p_{ij}\log\frac{p_{ij}}{q_{ij}}$$



```
from sklearn.manifold import TSNE
from sklearn.datasets import load_digits
digits = load_digits()
X = digits.data / 16.
X_tsne = TSNE().fit_transform(X)
X_pca = PCA(n_components=2).fit_transform(X)
```



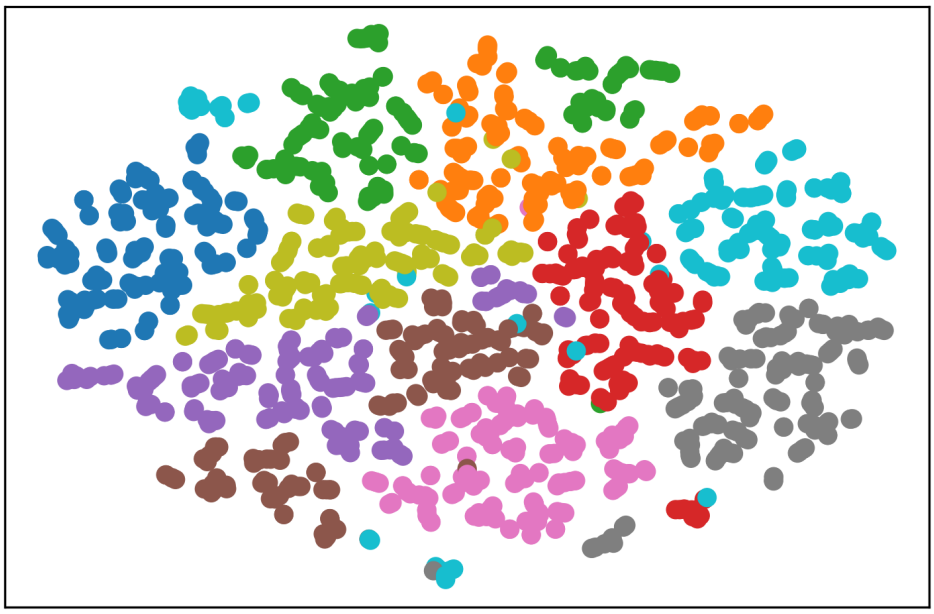


9

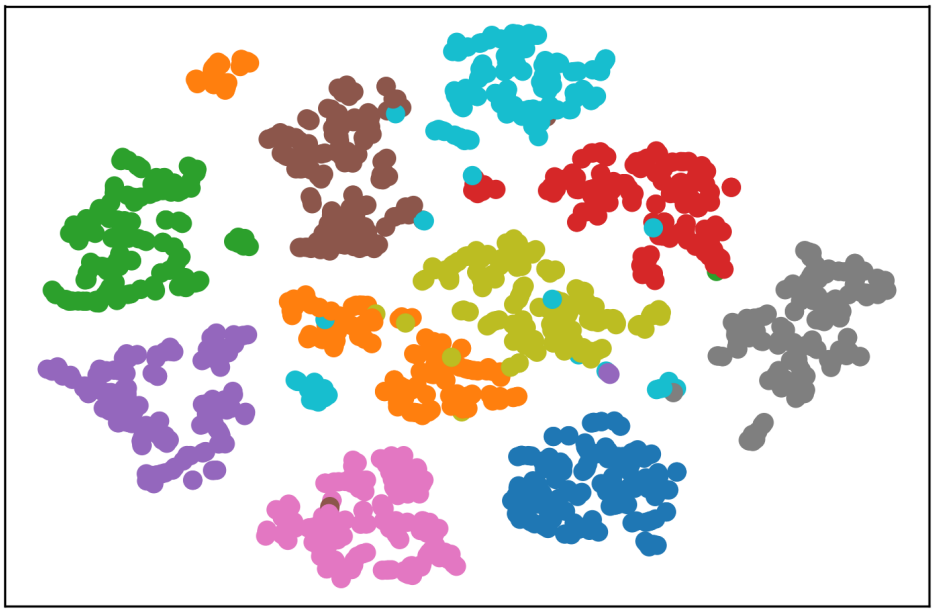
https://github.com/oreillymedia/t-SNE-tutorial

Tuning t-SNE perplexity

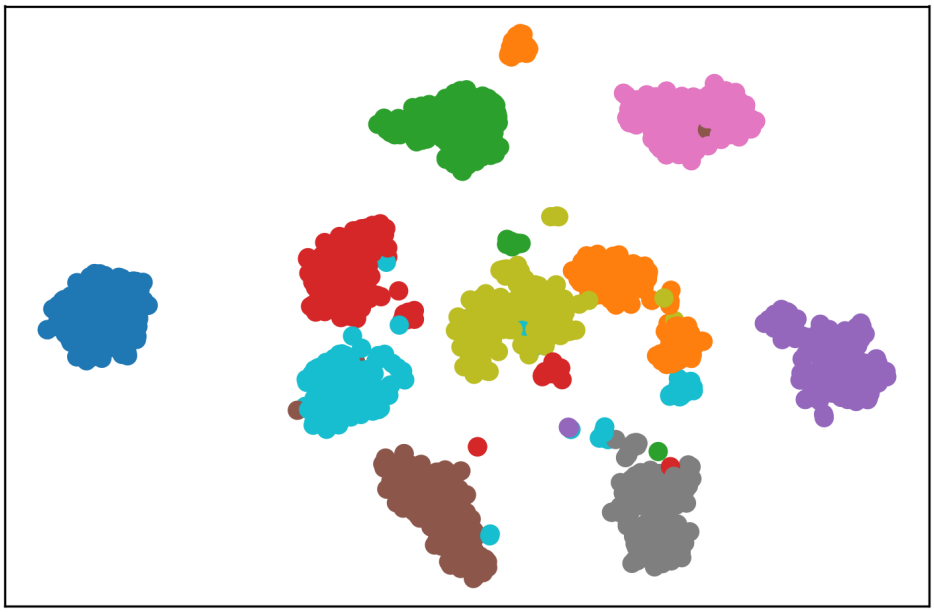
perplexity = 2



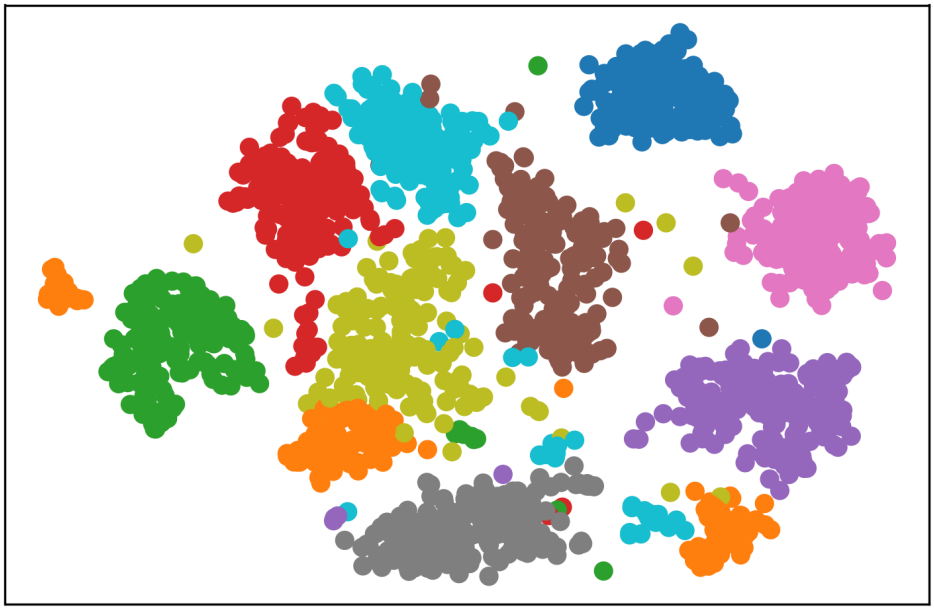
perplexity = 5

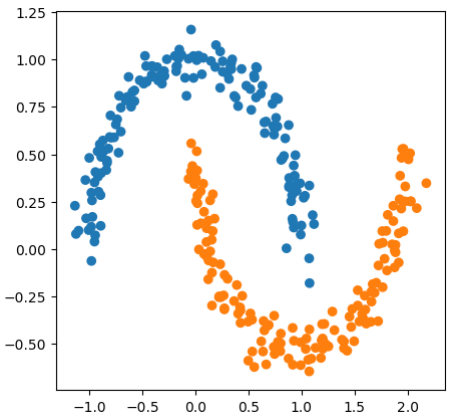


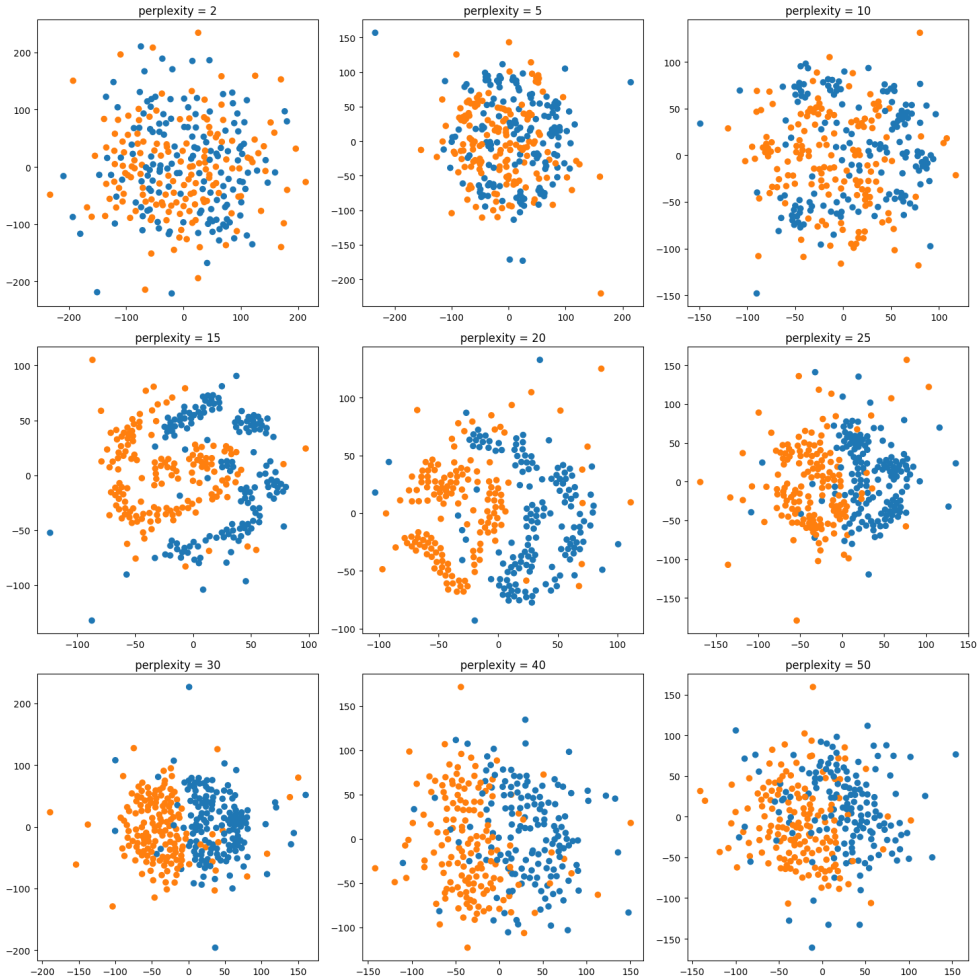
perplexity = 30



perplexity = 300





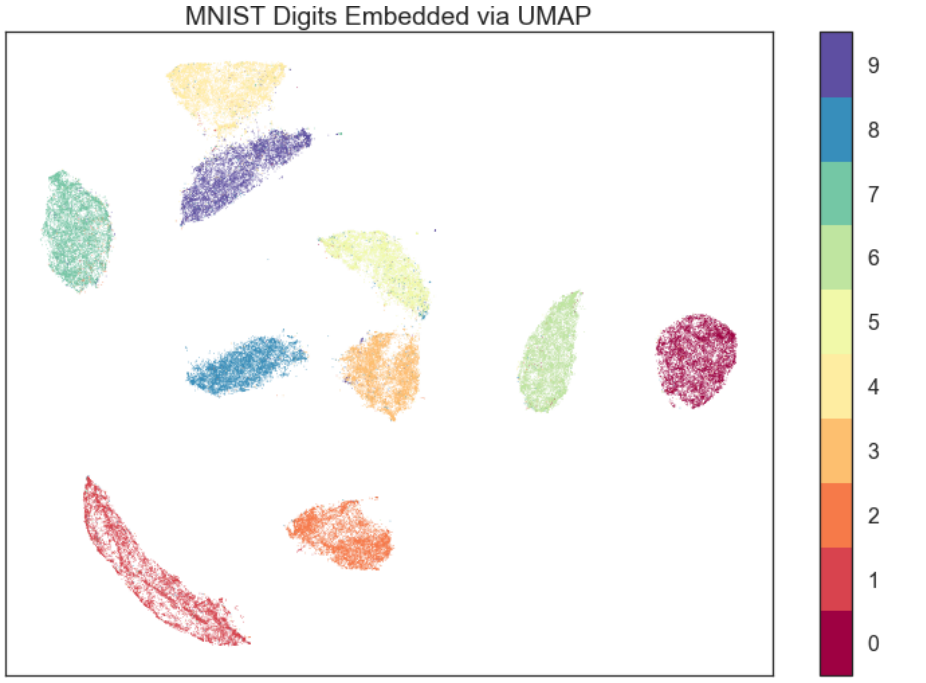


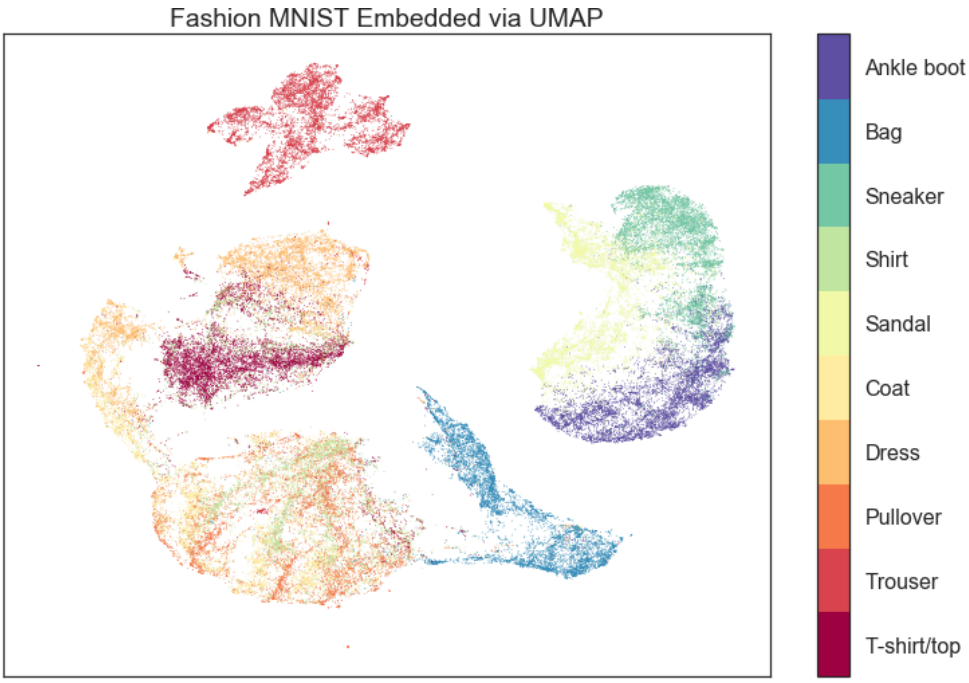
http://distill.pub/2016/misread-tsne/

UMAP

https://github.com/lmcinnes/umap

- Construct graph from simplices, use graph layout algorithm
- Much faster than t-SNE: MNIST takes 2 minutes vs 45 minutes for t-SNE

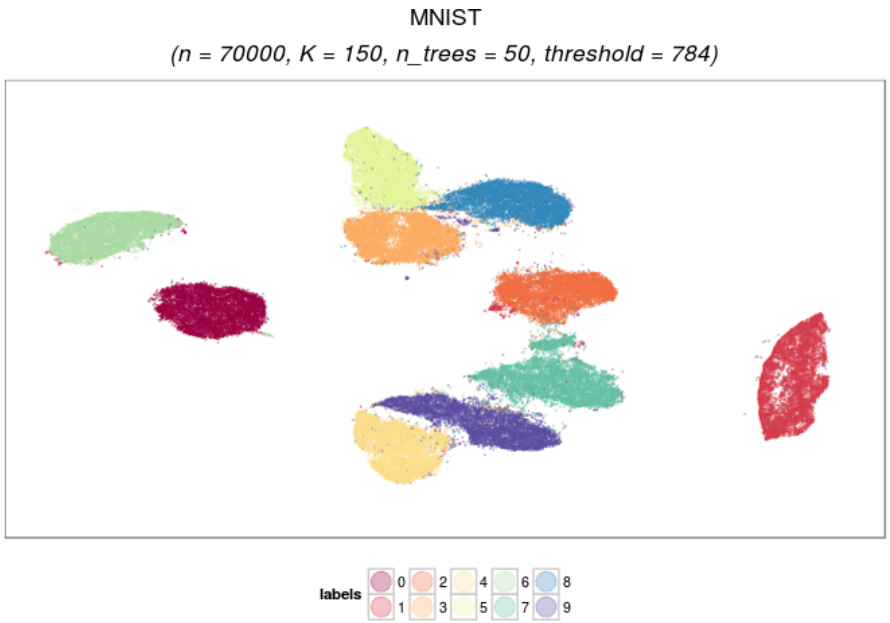


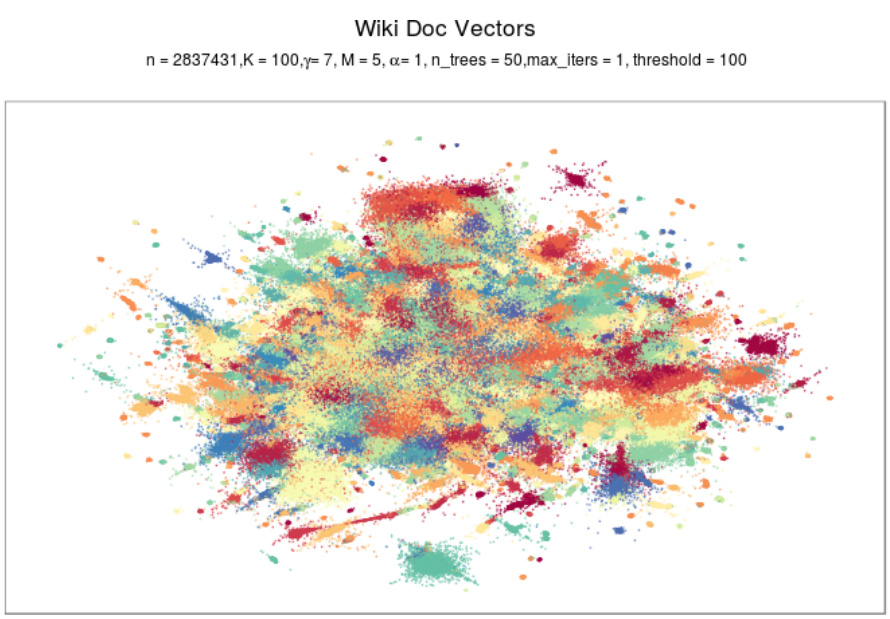


LargeVis

https://github.com/elbamos/largeVis

- Use nearest neighbor graph constructed very efficiently (and approximately)
- Much faster than t-SNE





K-means e HDBSCAN

https://hypertools.readthedocs.io/en/latest/tutorials/cluster.html

pca

https://hypertools.readthedocs.io/en/latest/auto_examples/plot_digits.html#sphx-glr-auto-examples-plot-digits-py

t-sne

https://hypertools.readthedocs.io/en/latest/auto_examples/plot_TSNE.html#sphx-glr-auto-examples-plot-tsne-py

umap

https://hypertools.readthedocs.io/en/latest/auto_examples/plot_UMAP.html#sphx-glr-auto-examples-plot-umap-py