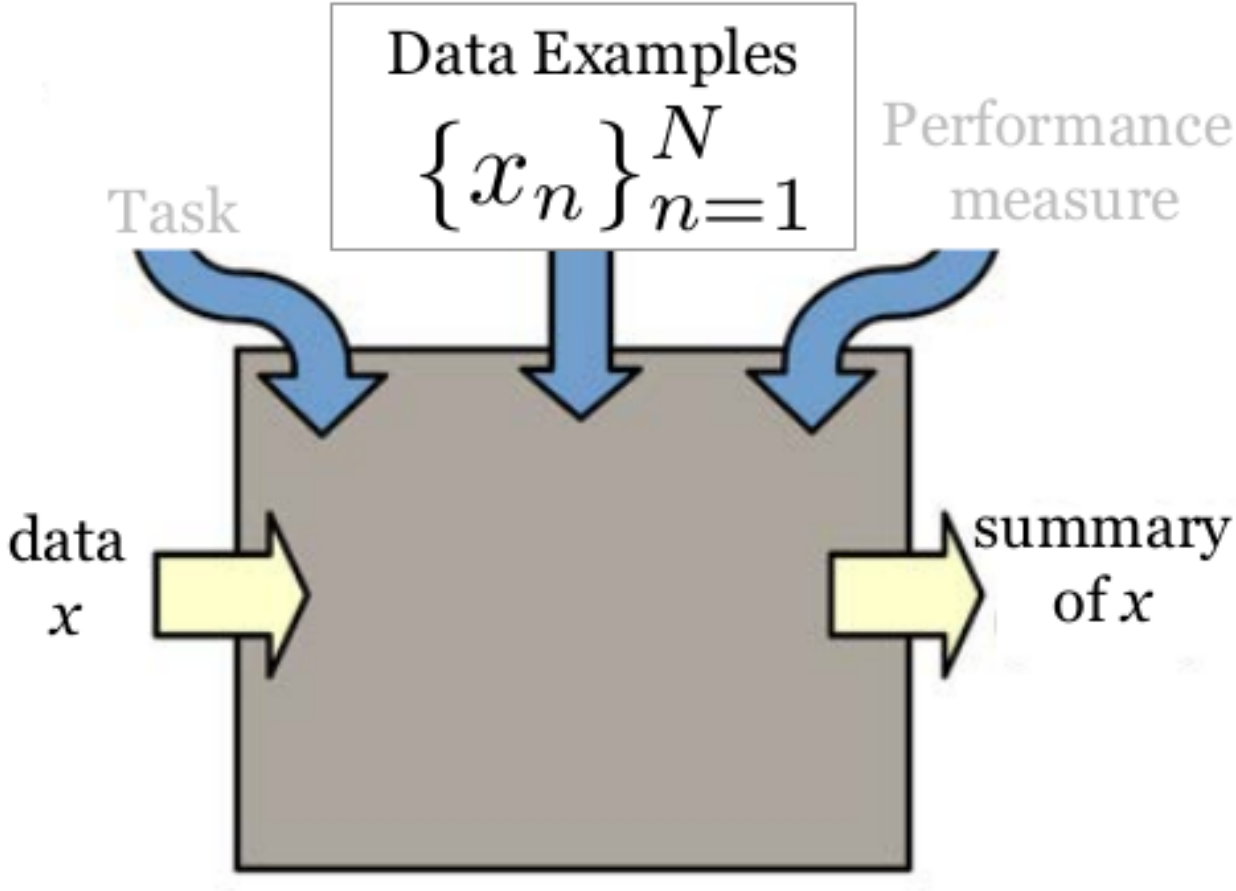
Clustering

Machine Learning and Computational Statistics (DSC6135)

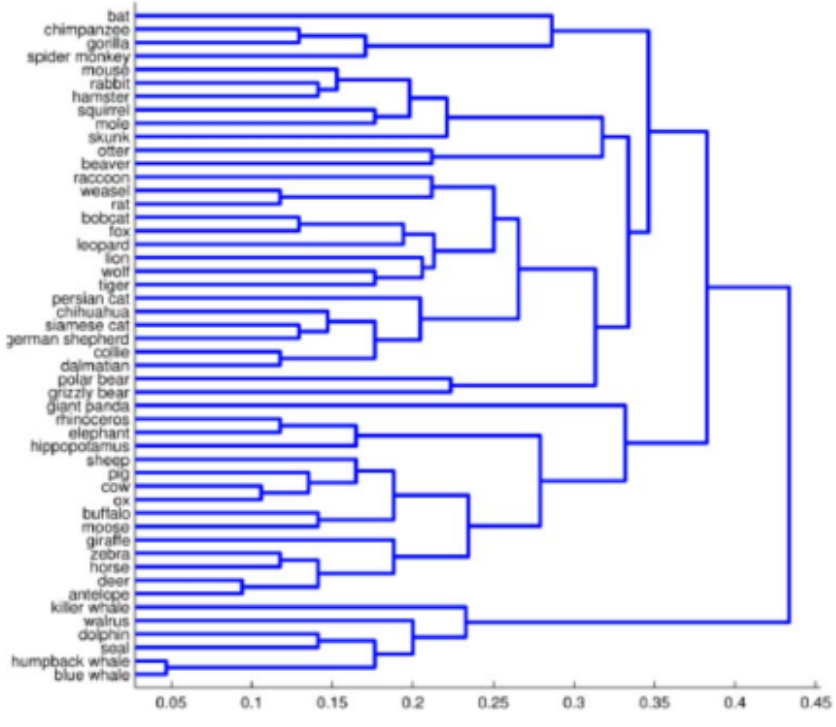
Instructors: Weiwei Pan (Harvard), Javier Zazo (Harvard), Melanie F. Pradier (Harvard)

- Supervised Learning
- **Unsupervised Learning**



Some Application Examples

Data set of 50 animals, 85 binary features (e.g., longneck, water, smelly)

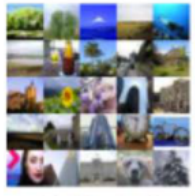


(a) Cluster Centers

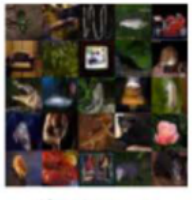
(b) Cluster 1

(c) Cluster 2

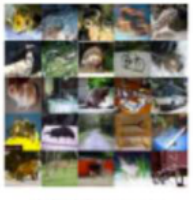
(d) Cluster 3



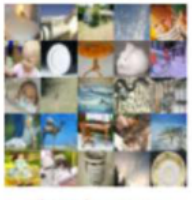
(e) Cluster 4



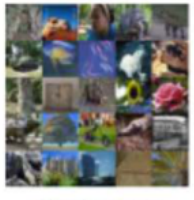
(f) Cluster 5



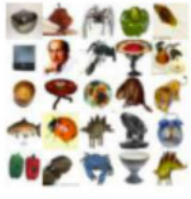
(g) Cluster 6



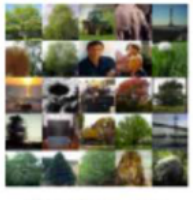
(h) Cluster 7



(i) Cluster 8

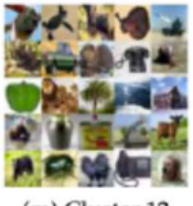


(j) Cluster 9

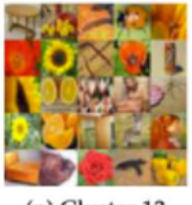


(k) Cluster 10

(l) Cluster 11



(m) Cluster 12



(n) Cluster 13

(o) Cluster 14

(p) Cluster 15

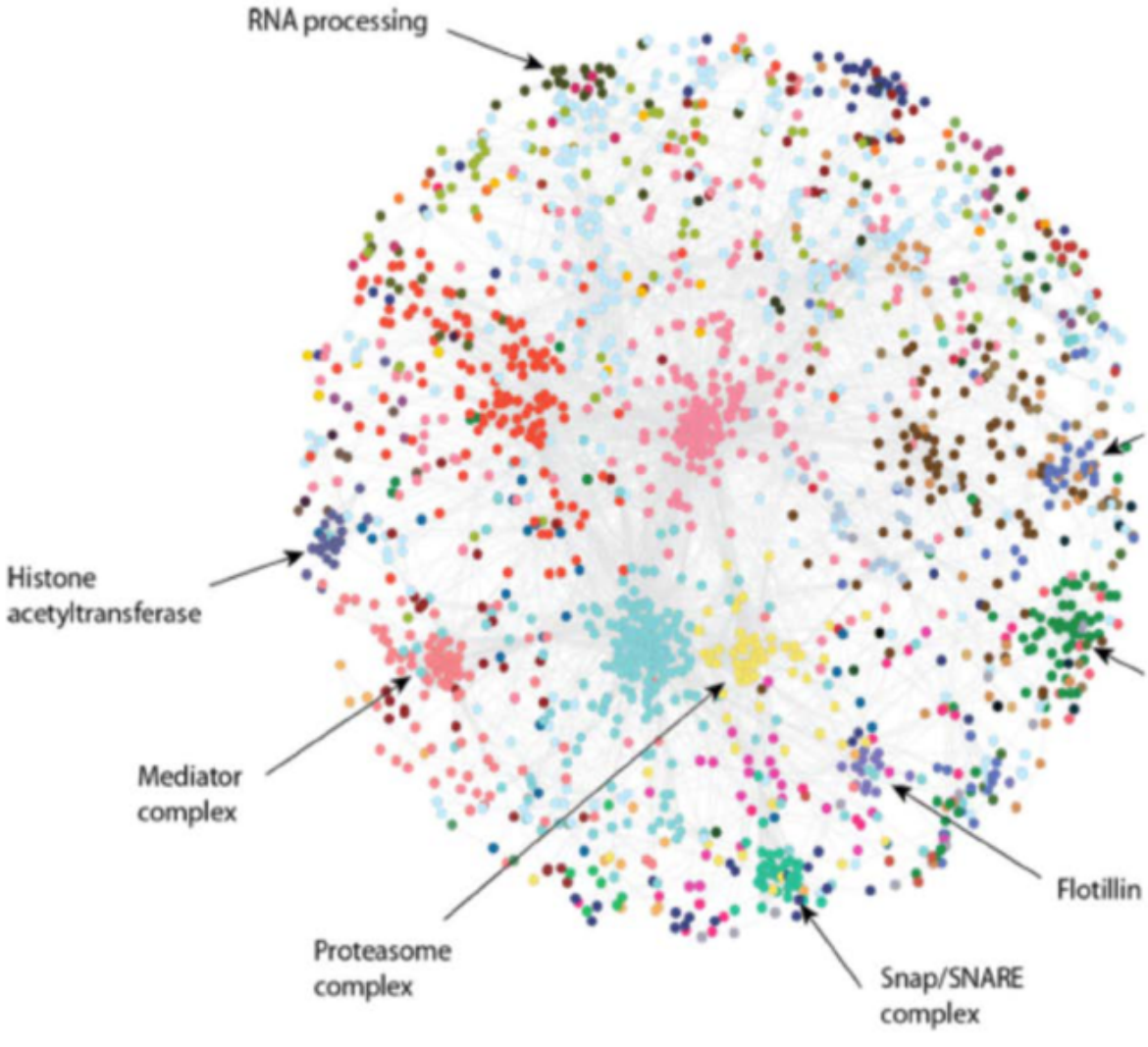
(q) Cluster 16

Original Image

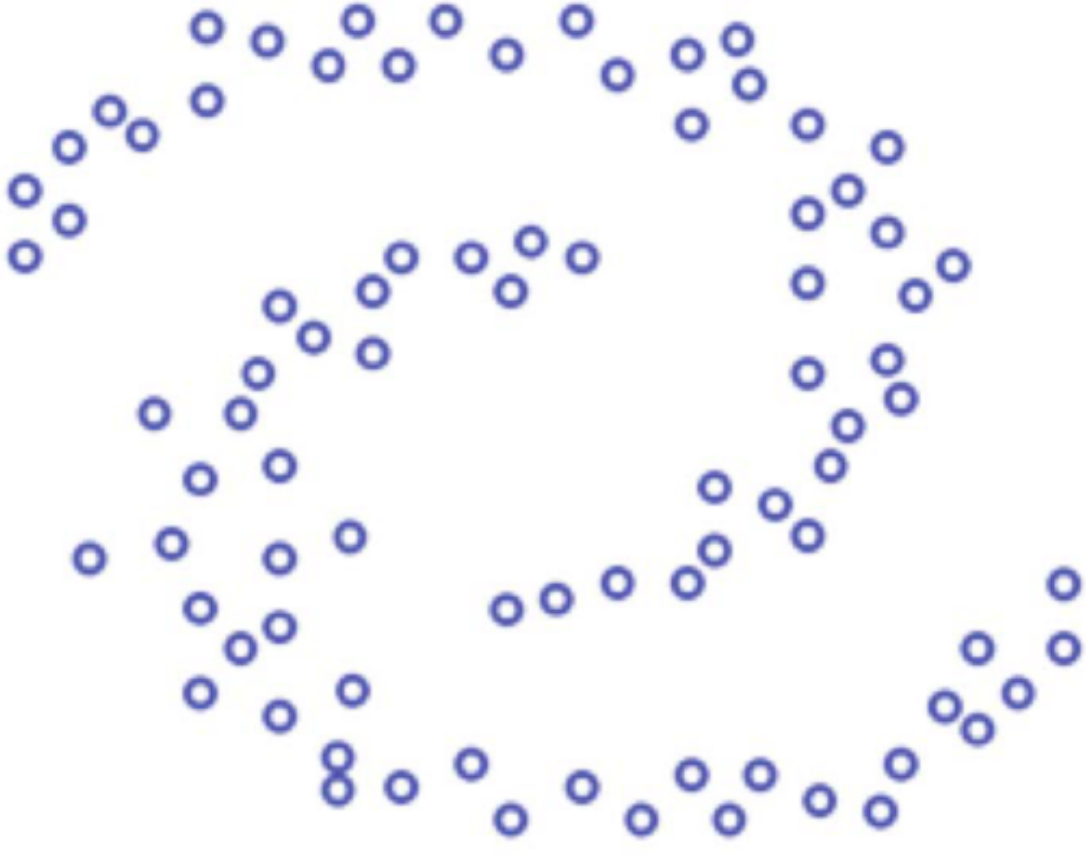
Possible pixel values (R, G, B):
255 * 255 * 255 = 16 million

16-color Image

Possible pixel values:
One of 16 fixed (R,G,B) values


RNA processing
Histone
acetyltransferase
Mediator
complex
Proteasome
complex
Snap/SNARE
complex
Flotillin

How to cluster these points?



How to cluster these points?

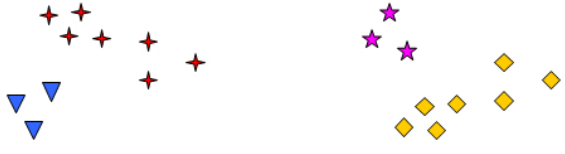
Key Questions

- How many clusters?

How many clusters?

Six Clusters

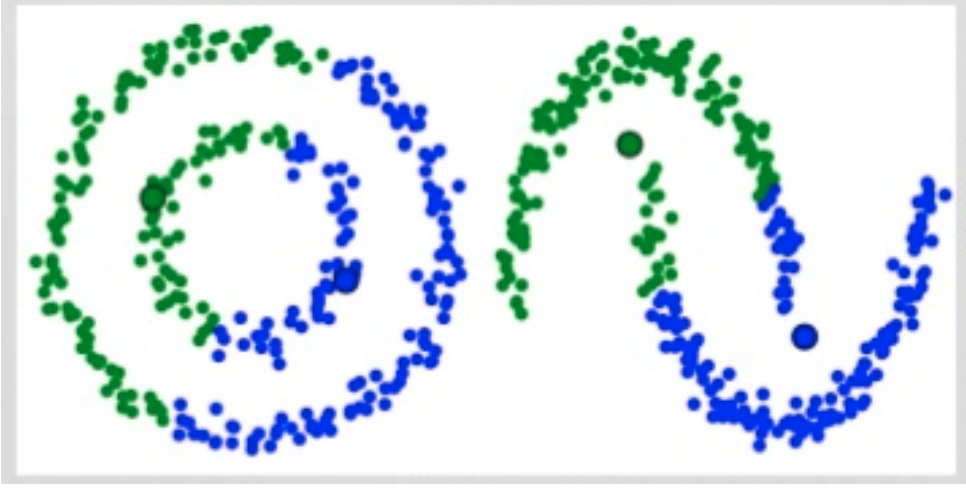
Two Clusters

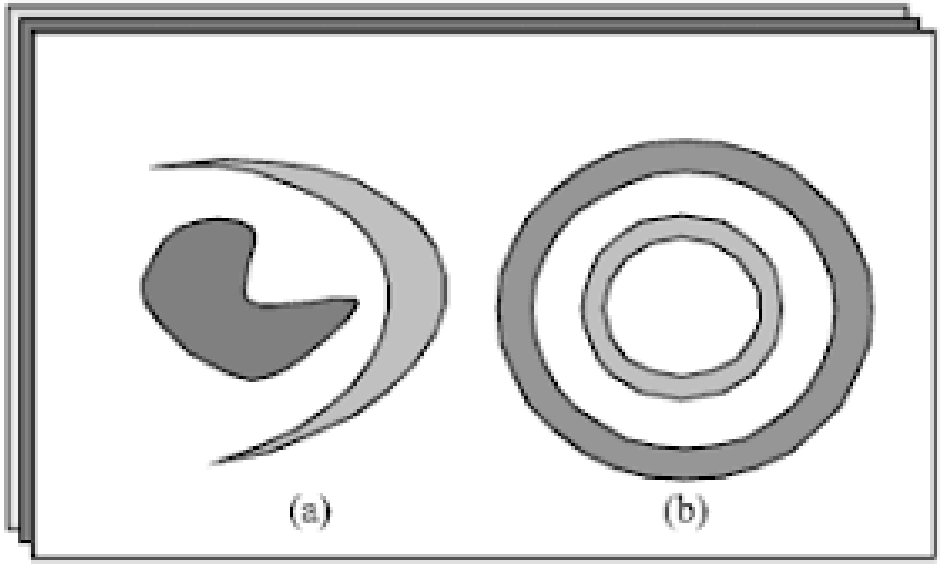


Four Clusters

Key Questions

- What shape for the clusters?





K-Means

- **Input**:
  - N datapoints: $x_1, x_2, \ldots, x_N$
  - parameters K (number of clusters)

- **Goals of K-means**:

    1. Assign each datapoint to one of K clusters
       (*Assumption*: clusters are exclusive)
    2. Minimize Euclidean distance from datapoints to cluster centers
       (*Assumption*: isotropic Euclidean - all features weighted equally - is a good metric)

K-means output:

- Centroid vectors (one per cluster $k = 1, \ldots, K$)

$$\mu_k = [\mu_{k1}, \mu_{k2}, \ldots, \mu_{kF}]$$

- Cluster Assignments (one per datapoint $n = 1, \ldots, N$)

$$z_n = [z_1, z_2, \ldots, z_{nK}] = [0\ 1\ 0\ 0\ 0]$$

K-means Optimization problem

$$\min_{z,\mu} \sum_{n=1}^{N} \sum_{k=1}^{K} z_{nk} \text{ distance}(x_n, \mu_k)$$

where

$$\text{distance}(x_n, \mu_k) = ||x_n - \mu_k||^2$$

Directly optimizing this expression is problematic!!

Iterative Algorithm (Expectation-Maximization)

- Initialize cluster means randomly
- Repeat until converged

1) Update per-example assignment

For each n in 1:N:
Find cluster k* that minimizes $\mathrm{dist}(x_n, \mu_k)$
Set z_n to indicate k*

2) Update per-cluster centroid

For each k in 1:K:
Set μ_k to mean of data vectors assigned to k

Iterative Algorithm (Expectation-Maximization)

- Initialize cluster means randomly
- Repeat until converged

1) Update per-example assignment

$$z_{nk} = \begin{cases} 1 & \text{if } \mathrm{dist}(x_n, \mu_k) \leq \mathrm{dist}(x_n, \mu_j) \quad \forall j \neq k \\ 0 & \text{o.w.} \end{cases}$$

2) Update per-cluster centroid

$$\mu_k = \frac{\sum_n z_{nk} x_n}{\sum_n z_{nk}}$$

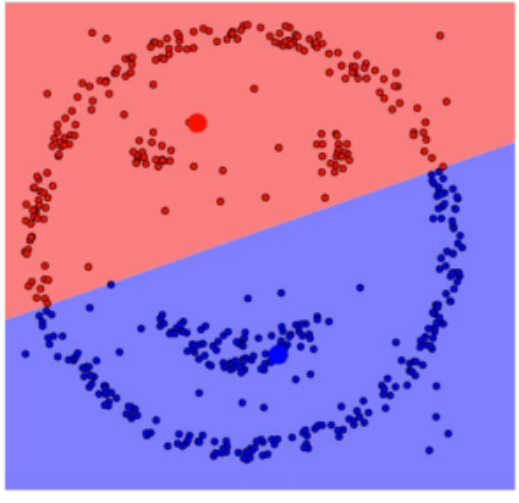
Demo!

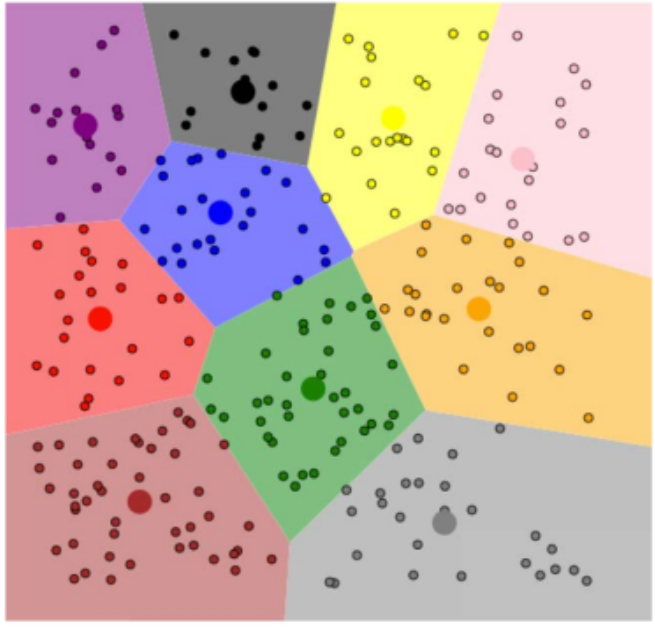
https://www.naftaliharris.com/blog/visualizing-k-means-clustering/ (https://www.naftaliharris.com/blog/visualizing-k-means-clustering/)

- For the same data and number of clusters, can you find different clustering configurations?
- What does this mean about the objective function?

Interesting Notes

- K-means boundaries are linear





- Each step in EM-algorithm leads to equal or lower cost than previous one
- Initialization important (leads to different solutions)
- Use cost to decide among multiple runs of K-means

In [ ]:

```
from sklearn.cluster import KMeans
kmeans = KMeans(n_clusters=4)
kmeans.fit(X)
y_kmeans = kmeans.predict(X)
```

In [ ]:

```python
# K-means algorithm from scratch (doing Expectation-Maximization)
from sklearn.metrics import pairwise_distances_argmin

def find_clusters(X, n_clusters, rseed=2):
    # 1. Randomly choose clusters
    rng = np.random.RandomState(rseed)
    i = rng.permutation(X.shape[0])[:n_clusters]
    centers = X[i]

    while True:
        # 2a. Get cluster assignments based on closest center
        assignments = pairwise_distances_argmin(X, centers)

        # 2b. Find new centers from means of points
        new_centers = np.array([X[assignments == i].mean(0)
                                for i in range(n_clusters)])

        # 2c. Check for convergence
        if np.all(centers == new_centers):
            break
        centers = new_centers

    return centers, assignments
```

How to initialize the centroids

- Step 1: choose an example uniformly at random as first centroid
- Repeat for k = 2, 3, … K:
    - Choose example based on distance from nearest centroid:

$$P(\mu_k) \propto \min_{1,2,\ldots,k-1} \mathrm{dist}(x_n, \mu_j)$$

    - In `sklearn` this is called **k-means++**.

- This procedure initializes clusters far from each other

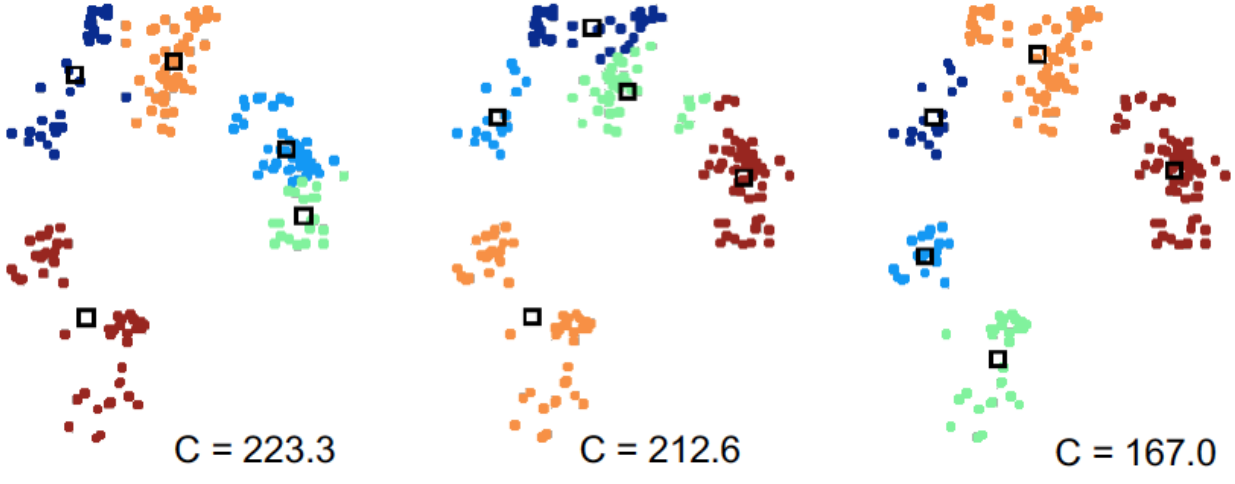
- There are some theoretical guarantees of worst-case performance...
- ... but the problem is hard!

How do we address this difficulties?

- Multiple initializations --> choose based on some score evaluation / metrics

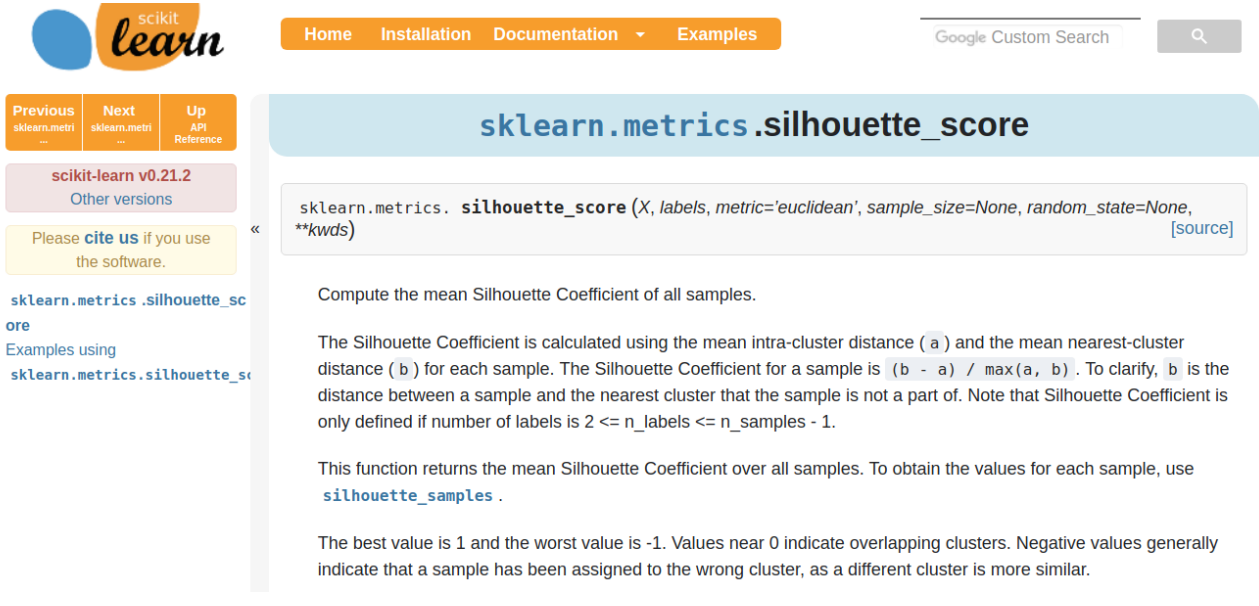
- There are no guarantees that your cluster results will be good with a single one!

Multiple initializations: scoring



How do we evaluate the score?

- Many possibile metrics: based on true labels, distance considerations between clusters...
- Here, we can use shiloutte coefficient or the inertia.



scikit learn

Home Installation Documentation Examples

Google Custom Search

Previous sklearn.metri... | Next sklearn.metri... | Up API Reference

scikit-learn v0.21.2
Other versions

Please cite us if you use the software.

sklearn.metrics.silhouette_score
Examples using sklearn.metrics.silhouette_sc

sklearn.metrics.silhouette_score

sklearn.metrics. **silhouette_score** (*X*, *labels*, *metric='euclidean'*, *sample_size=None*, *random_state=None*, ***kwds*) [source]

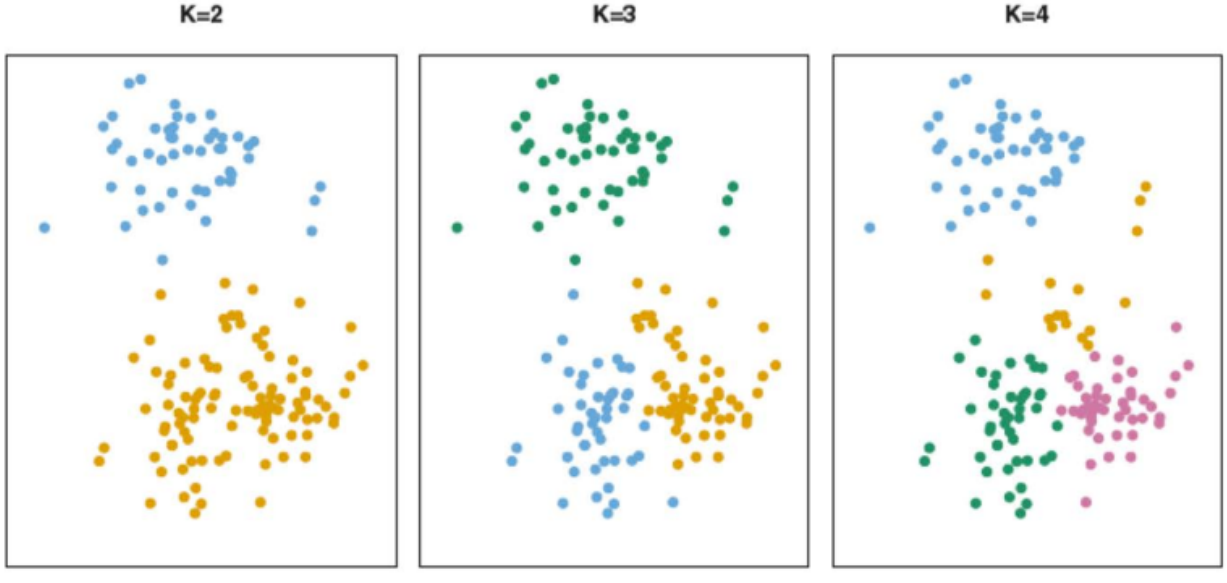
Compute the mean Silhouette Coefficient of all samples.

The Silhouette Coefficient is calculated using the mean intra-cluster distance (a) and the mean nearest-cluster distance (b) for each sample. The Silhouette Coefficient for a sample is (b - a) / max(a, b) . To clarify, b is the distance between a sample and the nearest cluster that the sample is not a part of. Note that Silhouette Coefficient is only defined if number of labels is 2 <= n_labels <= n_samples - 1.

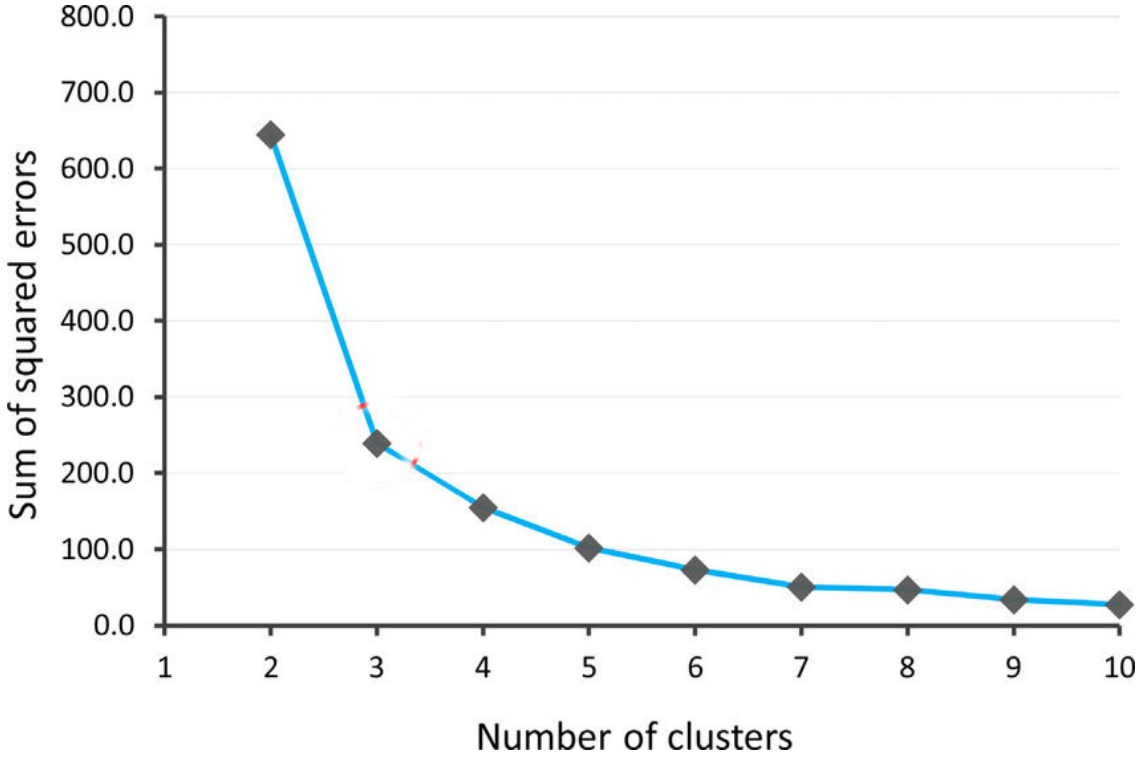
This function returns the mean Silhouette Coefficient over all samples. To obtain the values for each sample, use silhouette_samples .

The best value is 1 and the worst value is -1. Values near 0 indicate overlapping clusters. Negative values generally indicate that a sample has been assigned to the wrong cluster, as a different cluster is more similar.

How to choose number of clusters K?



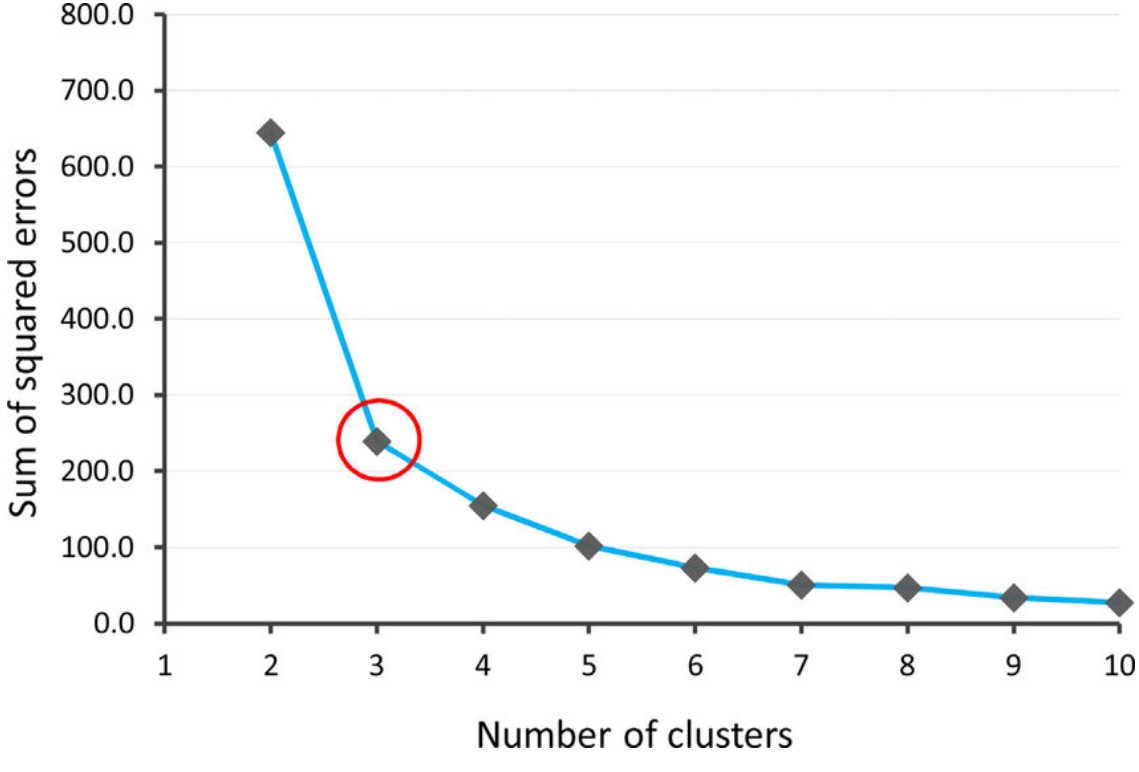
Can we use the cost function?



- We should not!! Instead, regularization

$$\text{loss function} = \text{cost} + \lambda\,\text{penalty}(K)$$

- Or... based on elbow:



Disadvantages of K-means

- Hard assignments: clusters are exclusive
- Weight equally each feature (isotropic Euclidean distance)

Gaussian Mixture Models

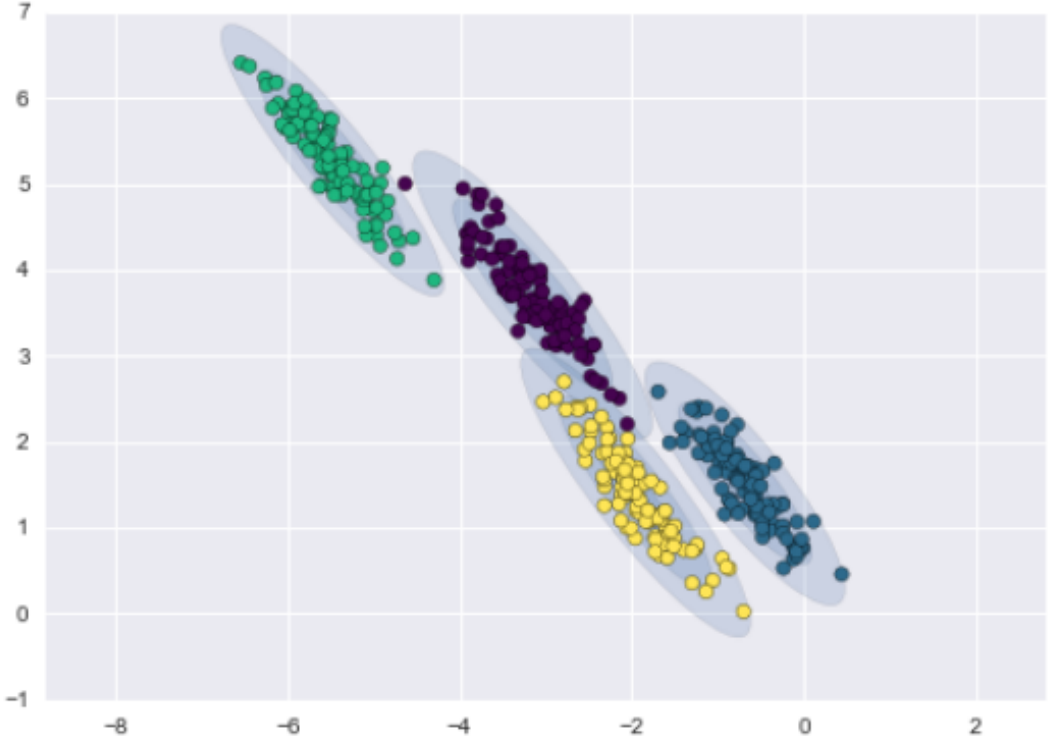
Improving K-means:

1. Assign each datapoint to one of K clusters
   (*Assumption*: clusters are exclusive)
   Improvement: soft-probabilistic assignments

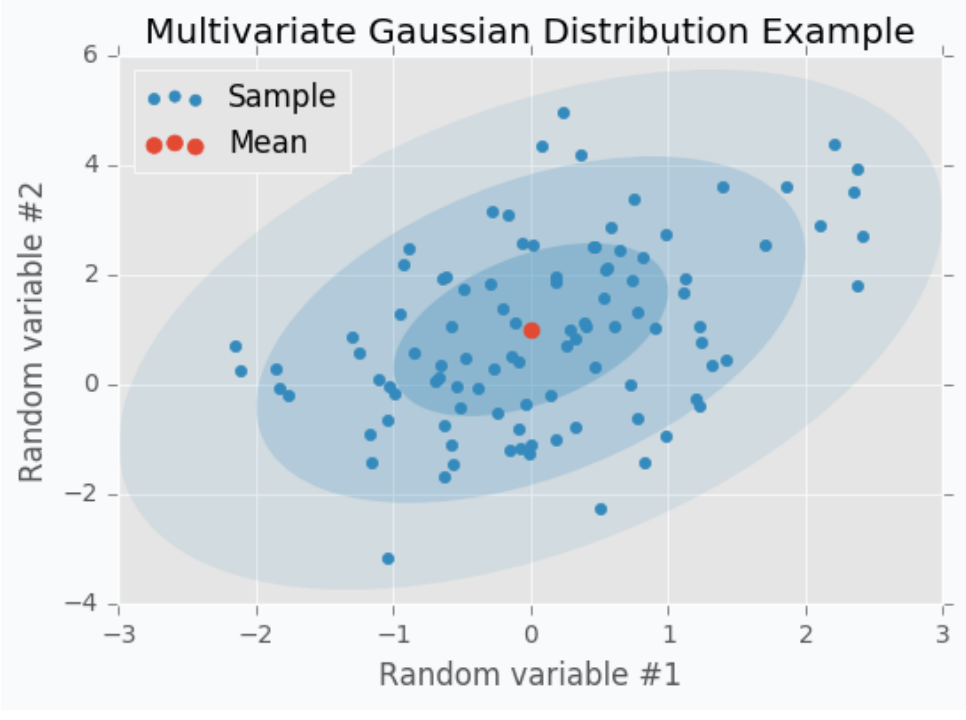
1. Minimize Euclidean distance from datapoints to cluster centers
   (*Assumption*: isotropic Euclidean - all features weighted equally - is a good metric)
   Improvement: model cluster covariance

```
gmm = GMM(n_components=4, covariance_type='full')
```

```
gmm = GMM(n_components=4, covariance_type='full', random_state=42)
plot_gmm(gmm, X_stretched)
```



Multivariate Gaussian

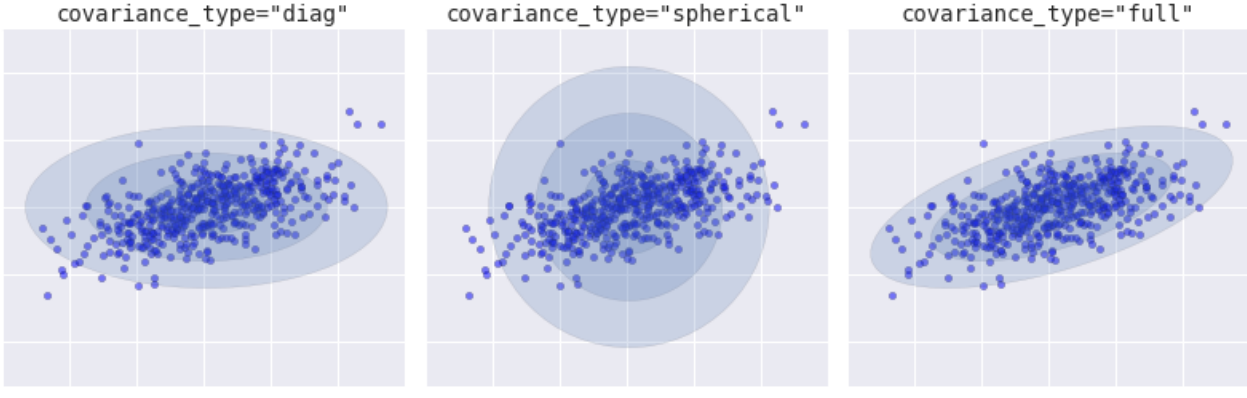


- Probability density function:

$$f_Y(x) = \frac{1}{\sqrt{(2\pi)^n |\mathbf{\Sigma}|}} \exp\left(-\frac{1}{2}(x-\mu)^T \mathbf{\Sigma}^{-1}(x-\mu)\right)$$

- Mean: $\mu \in \mathbb{R}^F$
- Covariance: $\Sigma \in \mathbb{R}^{F \times F}$

Covariance models



Parameters for Gaussian Mixture Models:

- mean vectors --> one per cluster k.

$$\mu_k = [\mu_{k1}, \mu_{k2}, \ldots, \mu_{kF}]$$

- covariance matrix --> one per cluster k.

$$\Sigma_k \in \mathbb{R}^{F \times F}, \qquad \forall k \in \{1, \ldots, K\}$$

- soft assignments --> one per example n in $1, \ldots, N$

$$r_n = [r_{n1}, r_{n2}, \ldots, r_{nK}] \qquad \forall n \in \{1, \ldots, N\}$$

Multivariate Gaussian parameter estimation Training

- How to fit the parameters of a multivariate Gaussian from data?

$$X = x_1, x_2, \ldots, x_N.$$

- Maybe with a log-maximum likelihood estimate?

- Mean is the average of the samples:

$$\mu = \frac{1}{N} \sum_{i=1}^{N} x_i$$

- The covariance matrix:

$$\Sigma = \frac{1}{N} \sum_{i=1}^{N} (x_i - \mu)(x_i - \mu)^T$$

Gaussian Mixture Model training

- We can estimate the parameters of multivariate Gaussians easily...
- ... but we don't know to which cluster each data point belongs to.

- We can, once more, maximize log-likelihood!

$$\max_{\mu_k, \Sigma_k} \quad \sum_{i=1}^{N} \sum_{k=1}^{K} \log p(x_i, z_{ik} = 1 | \mu_k, \Sigma_k)$$

- How do we optimize?
- We don't have cluster assignments, and we don't have mean and covariance estimates --> but we will use soft assignments!

Expectation - Maximization

- We repeat the following procedure until convergence
- On each step, we assume the other variables fixed.

---

1. E-step: Update soft assignments r (and normalize):

$$r_{ik} \propto p(x_i, z_k = 1 | \mu_k, \Sigma_k) \quad \forall k, i$$

1. M-step: Update means and covariances:

$$\mu = \frac{1}{N} \sum_{i=1}^{N} r_{ik} x_i$$

$$\Sigma = \frac{1}{N} \sum_{i=1}^{N} r_{ik} (x_i - \mu)(x_i - \mu)^T$$

K-means is a GMM with:

- hard assignments --> probabilities r are one-hot encodings.
- spherical covariances

$$\Sigma_k = \lim_{\sigma \to 0} \sigma^2 \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$