Applied Machine Learning

Unsupervised machine learning

Kevyn Collins-Thompson

Associate Professor of Information & Computer Science
University of Michigan

Introduction: Unsupervised Learning

- Unsupervised learning involves tasks that operate on datasets without labeled responses or target values.
- Instead, the goal is to capture interesting structure or information.

Applications of unsupervised learning:

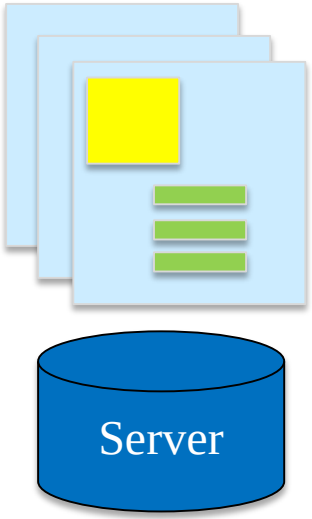
- Visualize structure of a complex dataset.
- Density estimation to predict probabilities of events.
- Compress and summarize the data.
- Extract features for supervised learning.
- Discover important clusters or outliers.

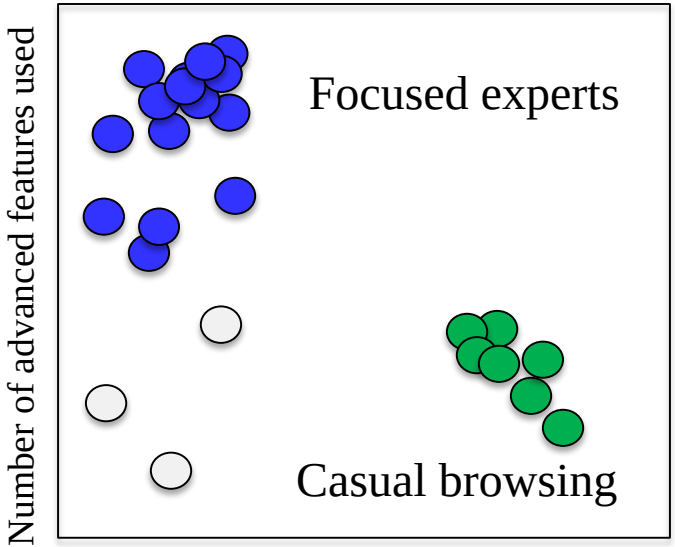
Web Clustering Example

User population

Interaction signals

- Site features used
- Pages browsed

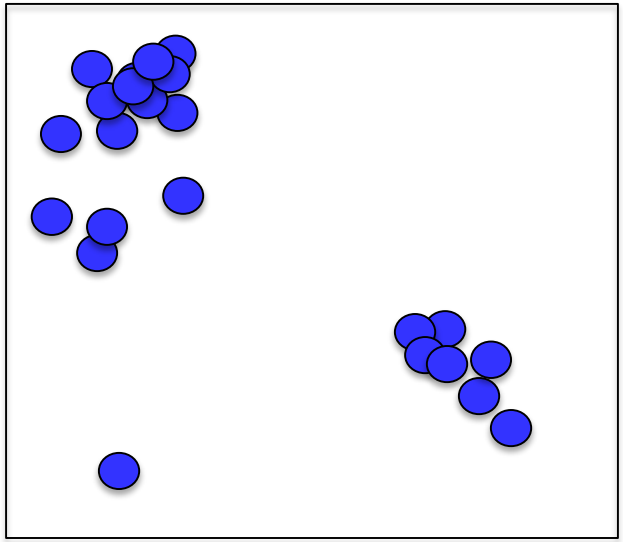




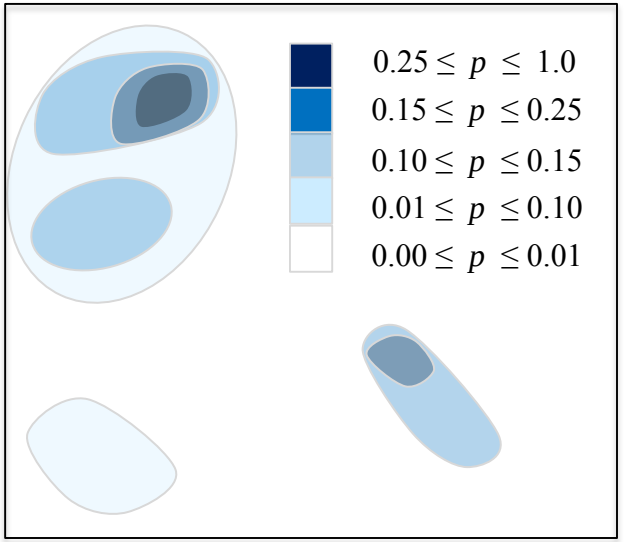
Two major types of unsupervised learning methods

- **Transformations**
  - *Processes that extract or compute information*
- **Clustering**
  - *Find groups in the data*
  - *Assign every point in the dataset to one of the groups*

Transformations: Density Estimation

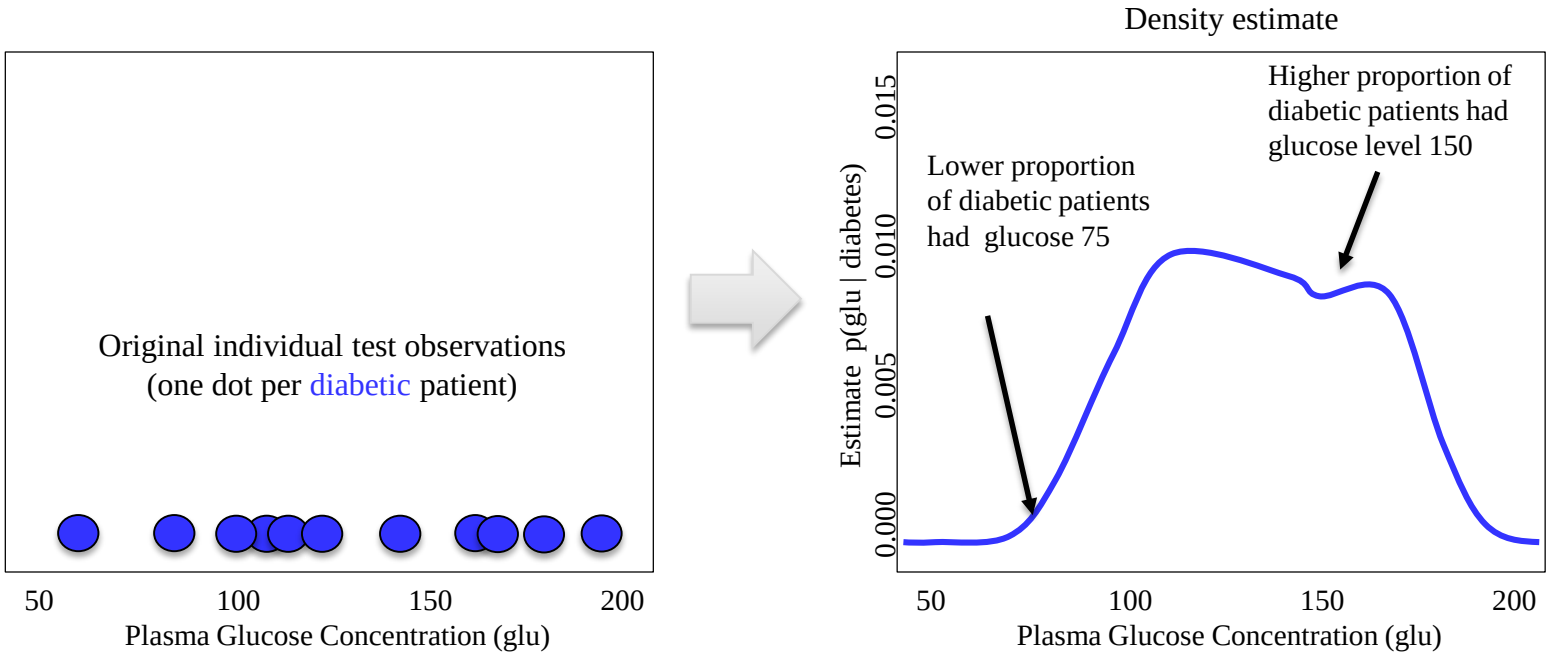


Individual measurements

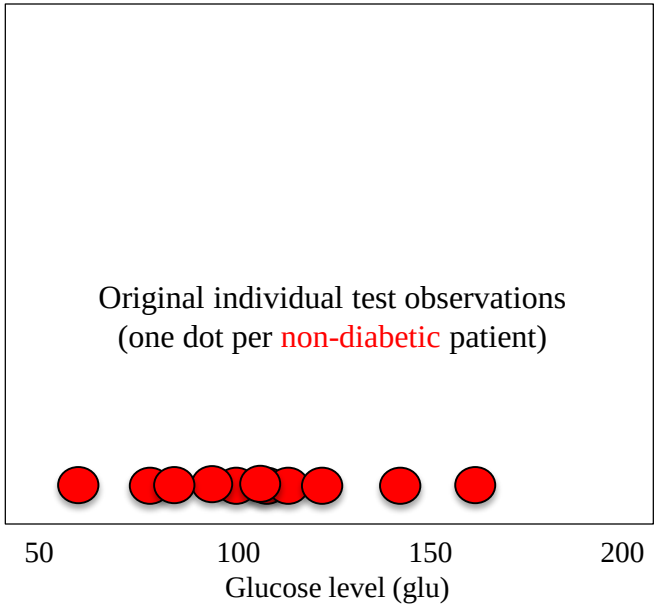


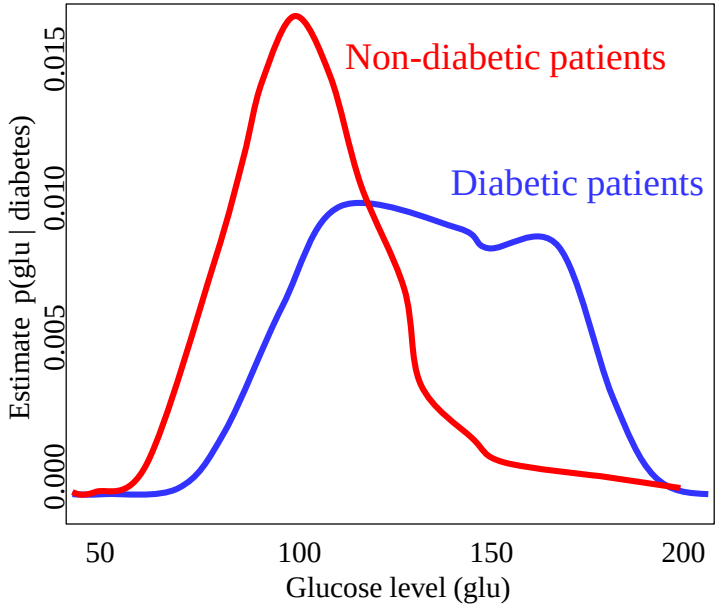
Density estimate
(Estimated probability p of observing a measurement at a given location)

Density Estimation Example

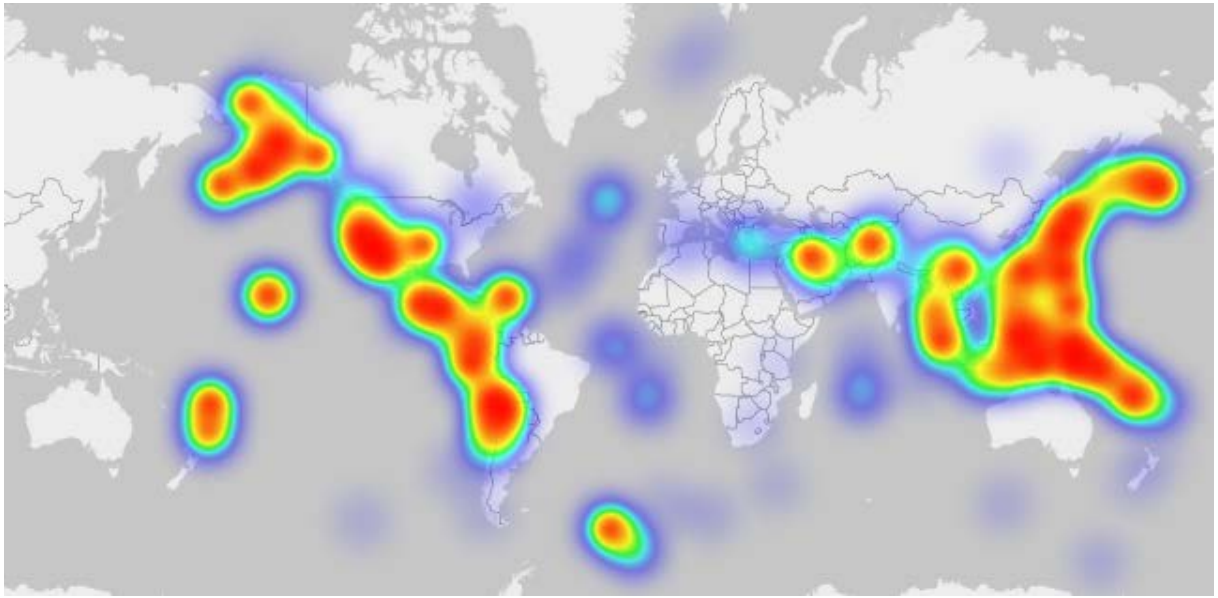


Density Estimation Example





Kernel Density Example

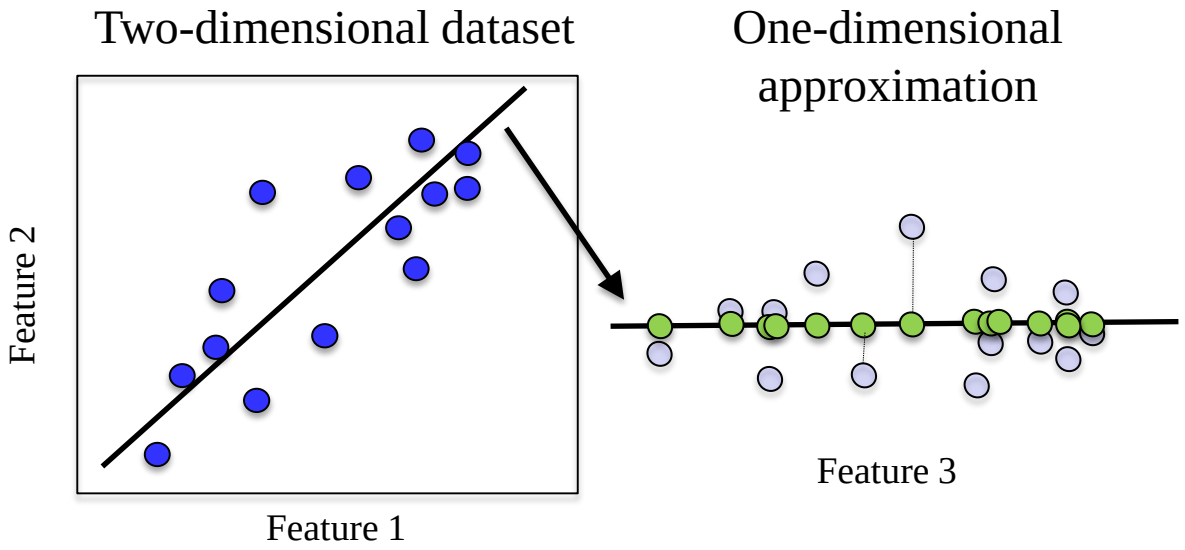


Recent global earthquake activity (U.S. Geological Survey data)

Source: http://www.digital-geography.com/csv-heatmap-leaflet/

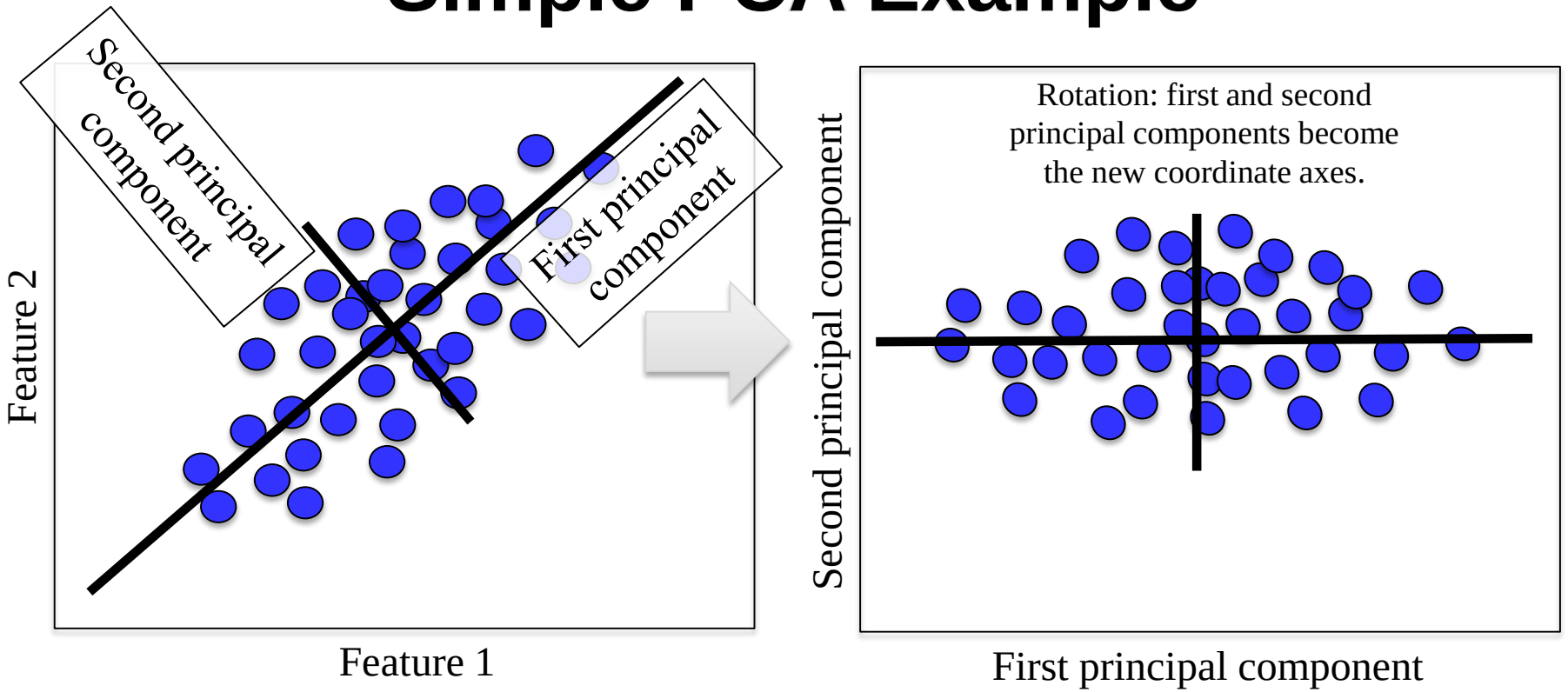
Dimensionality Reduction

- **Finds an approximate version of your dataset using fewer features**
- **Used for exploring and visualizing a dataset to understand grouping or relationships**
- **Often visualized using a 2-dimensional scatterplot**
- **Also used for compression, finding features for supervised learning**



The one-dimensional approximation is obtained by projecting the original points onto the diagonal line and using their position on that line as the new single feature.

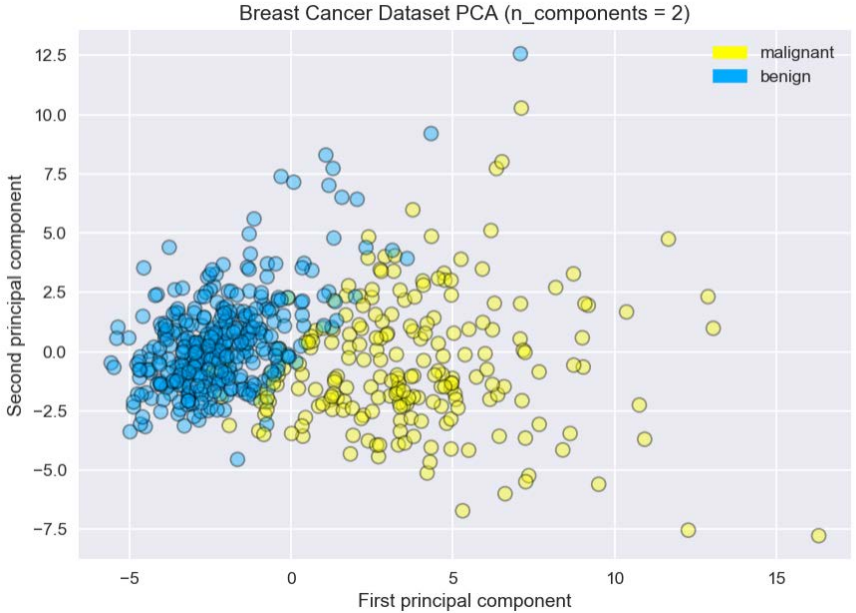
Simple PCA Example



Dimensionality Reduction with PCA in scikit-learn

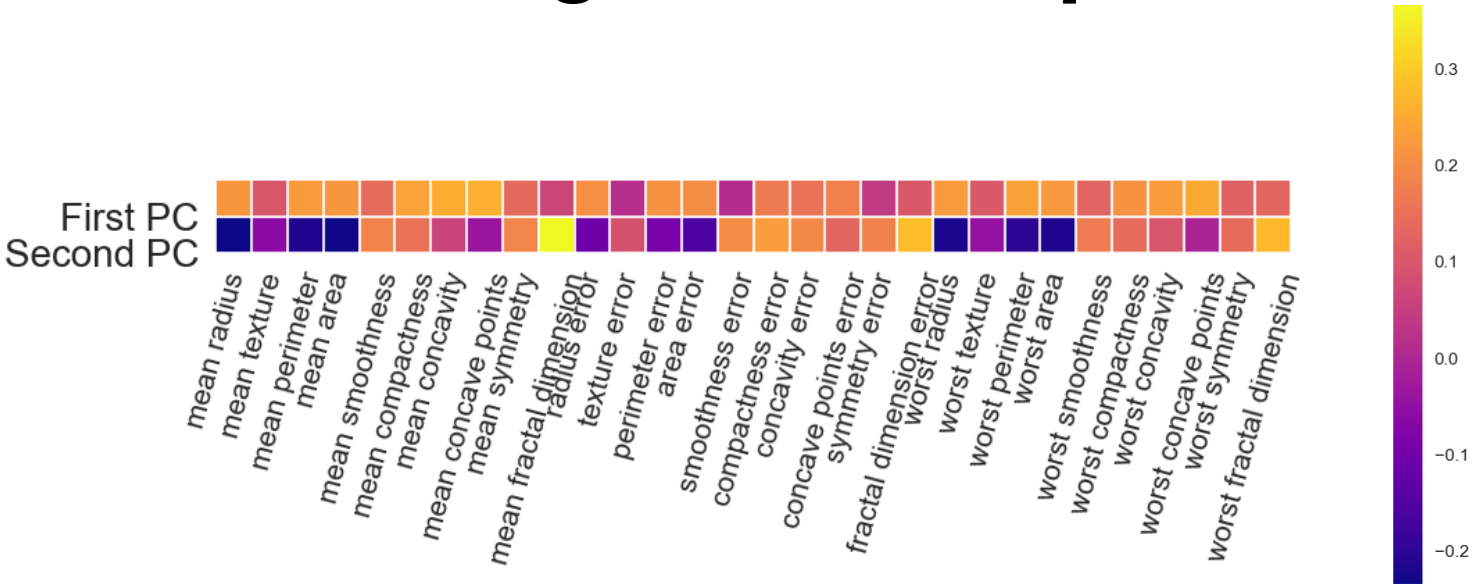
```
# PCA
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.datasets import load_breast_cancer

cancer = load_breast_cancer()
(X_cancer, y_cancer) = load_breast_cancer(return_X_y = True)

# each feature should be centered (zero mean) and with unit variance
X_normalized = StandardScaler().fit(X_cancer).transform(X_cancer)

pca = PCA(n_components = 2).fit(X_normalized)

X_pca = pca.transform(X_normalized)
print(X_cancer.shape, X_pca.shape)
```

```
(569, 30) (569, 2)
```

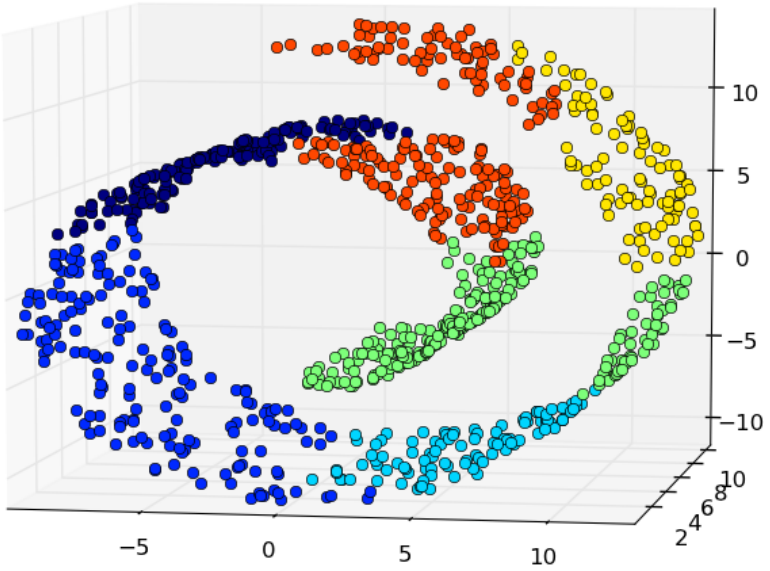



```
from adspy_shared_utilities import plot_labelled_scatter
plot_labelled_scatter(X_pca, y_cancer, ['malignant', 'benign'])

plt.xlabel('First principal component')
plt.ylabel('Second principal component')
plt.title("Breast Cancer Dataset PCA (n_components = 2)")
```

Visualizing PCA Components



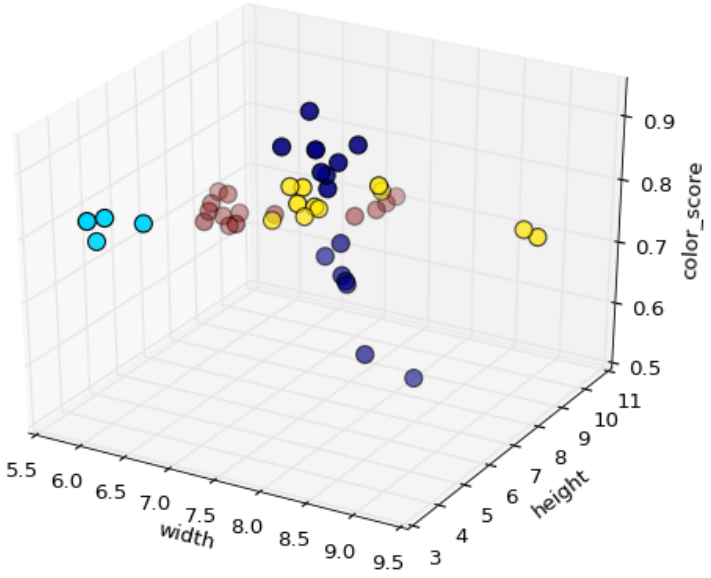
The ”Swiss Roll" Dataset



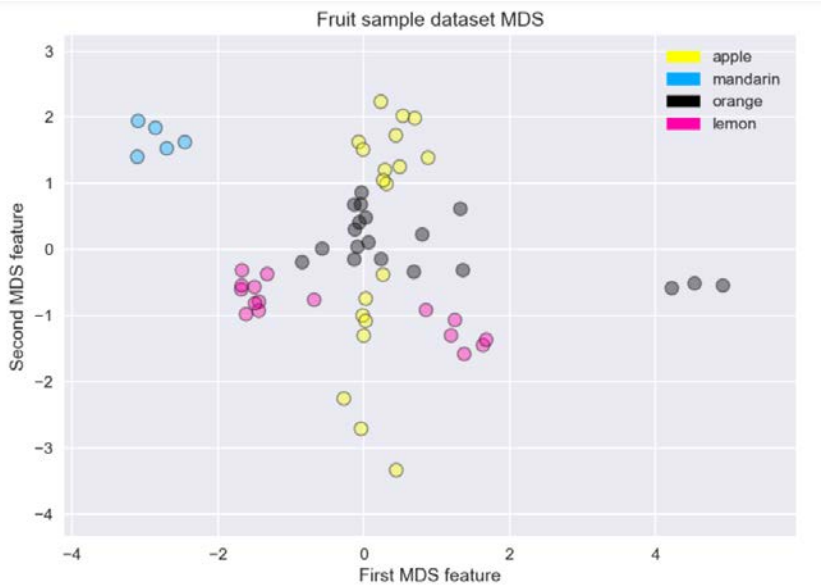
`sklearn.datasets.make_swiss_roll(n_samples=1500, noise=0.05)`

See: `http://scikit-learn.org/stable/modules/clustering.html#hierarchical-clustering`

Multidimensional scaling (MDS) attempts to find a distance-preserving low-dimensional projection



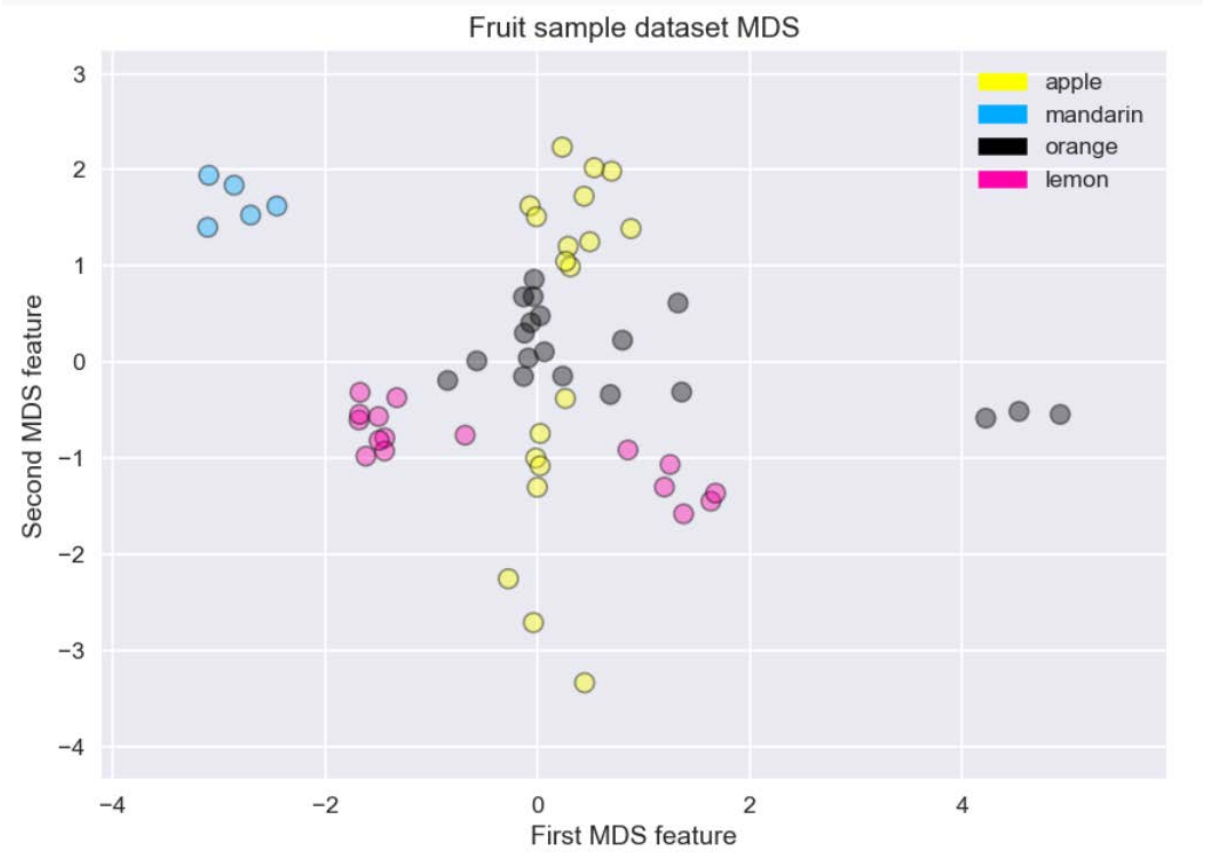
High-dimensional dataset



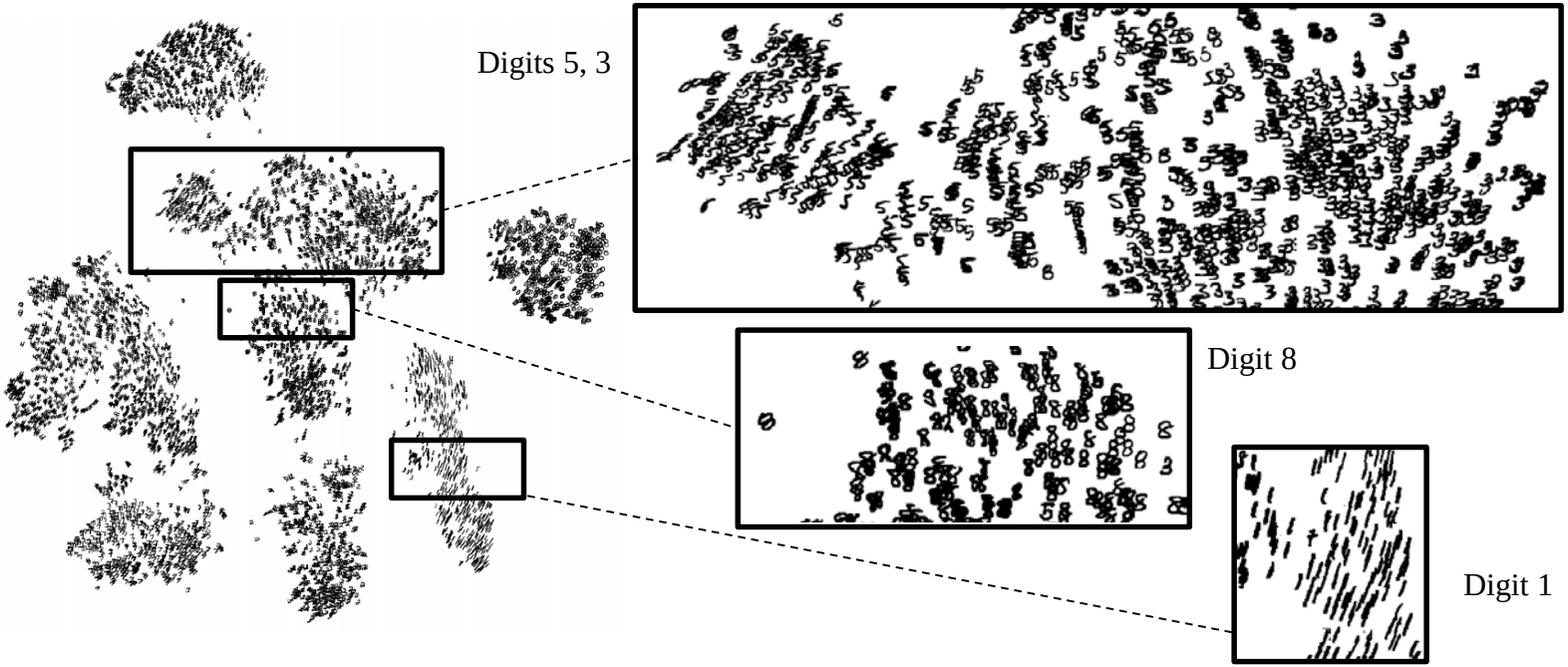
Two-dimensional MDS projection

Notebook: MDS on the Fruit Dataset

```
# Multidimensional scaling
from adspy_shared_utilities import plot_labelled_scatter
from sklearn.preprocessing import StandardScaler
from sklearn.manifold import MDS

# each feature should be centered (zero mean) and with unit variance
X_fruits_normalized = StandardScaler().fit(X_fruits).transform(X_fruits)

mds = MDS(n_components = 2)

X_fruits_mds = mds.fit_transform(X_fruits_normalized)

plot_labelled_scatter(X_fruits_mds, y_fruits, ['apple', 'mandarin', 'orange', 'lemon'])
plt.xlabel('First MDS feature')
plt.ylabel('Second MDS feature')
plt.title("Fruit sample dataset MDS")
```


Fruit sample dataset MDS
apple
mandarin
orange
lemon
Second MDS feature
First MDS feature
3
2
1
0
−1
−2
−3
−4
−4
−2
0
2
4

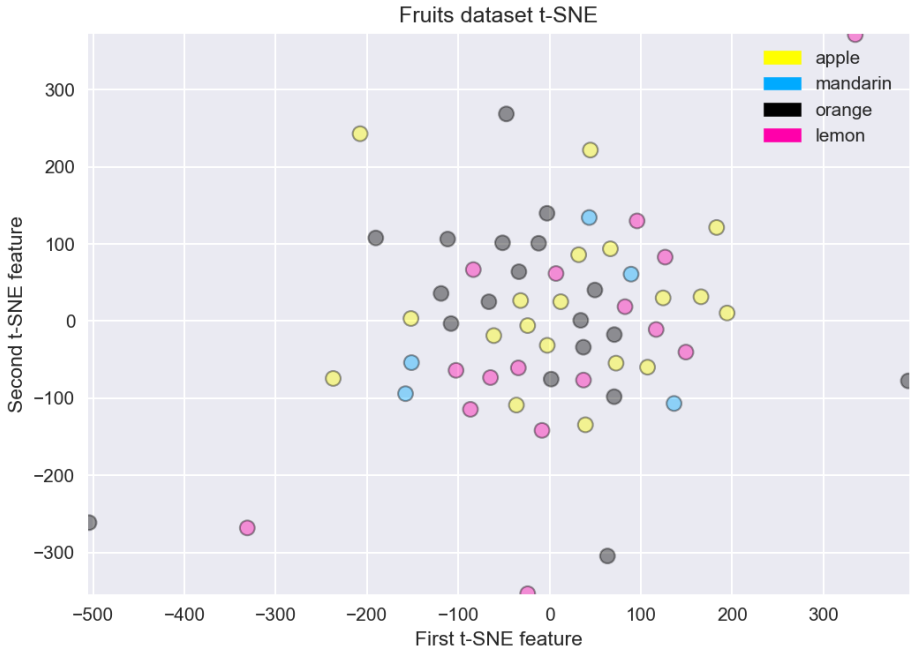
t-SNE: a powerful manifold learning method that finds a 2D projection preserving information about neighbors



Source: van der Maaten & Hinton https://lvdmaaten.github.io/tsne/

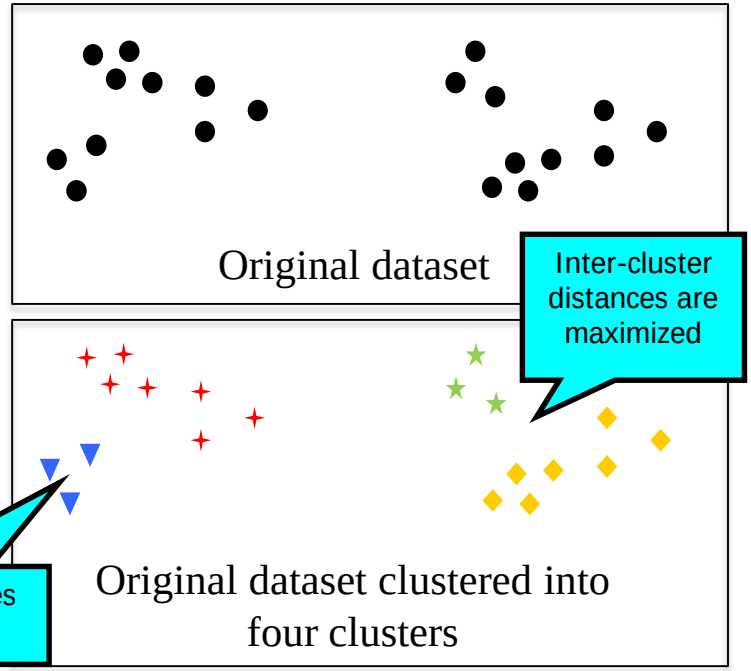
Notebook: t-SNE on the Fruit Dataset

```
from sklearn.manifold import TSNE

tsne = TSNE(random_state = 0)

X_tsne = tsne.fit_transform(X_fruits_normalized)

plot_labelled_scatter(X_tsne, y_fruits,
    ['apple', 'mandarin', 'orange', 'lemon'])
plt.xlabel('First t-SNE feature')
plt.ylabel('Second t-SNE feature')
plt.title("Fruits dataset t-SNE")
```



Clustering:
Finding a way to divide a dataset into groups ('clusters')

- **Data points within the same cluster should be 'close' or 'similar' in some way.**
- **Data points in different clusters should be 'far apart' or 'different'**
- **Clustering algorithms output a cluster membership index for each data point:**
  - *Hard clustering: each data point belongs to exactly one cluster*
  - *Soft (or fuzzy) clustering: each data point is assigned a weight, score of membership for each clu*



K-means Clustering

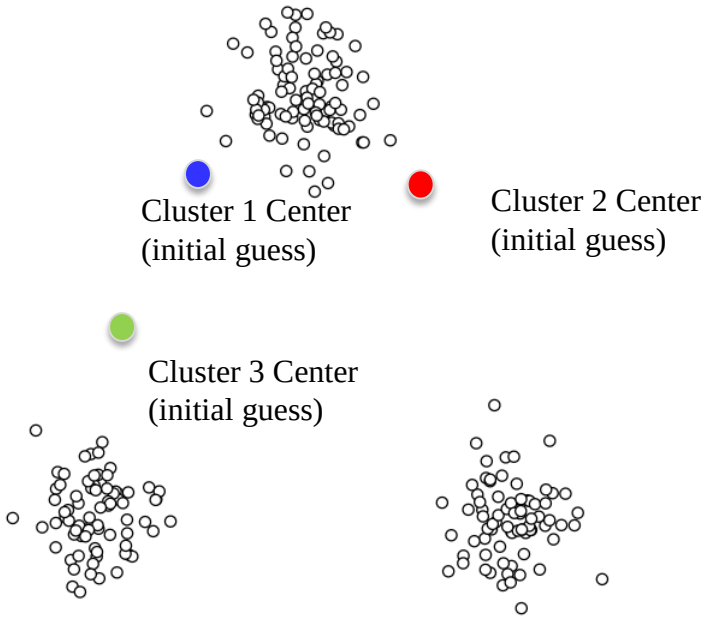
The k-means algorithm

Initialization Pick the number of clusters *k* you want to find. Then pick *k random* points to serve as an initial guess for the cluster centers.

Step A Assign each data point to the nearest cluster center.

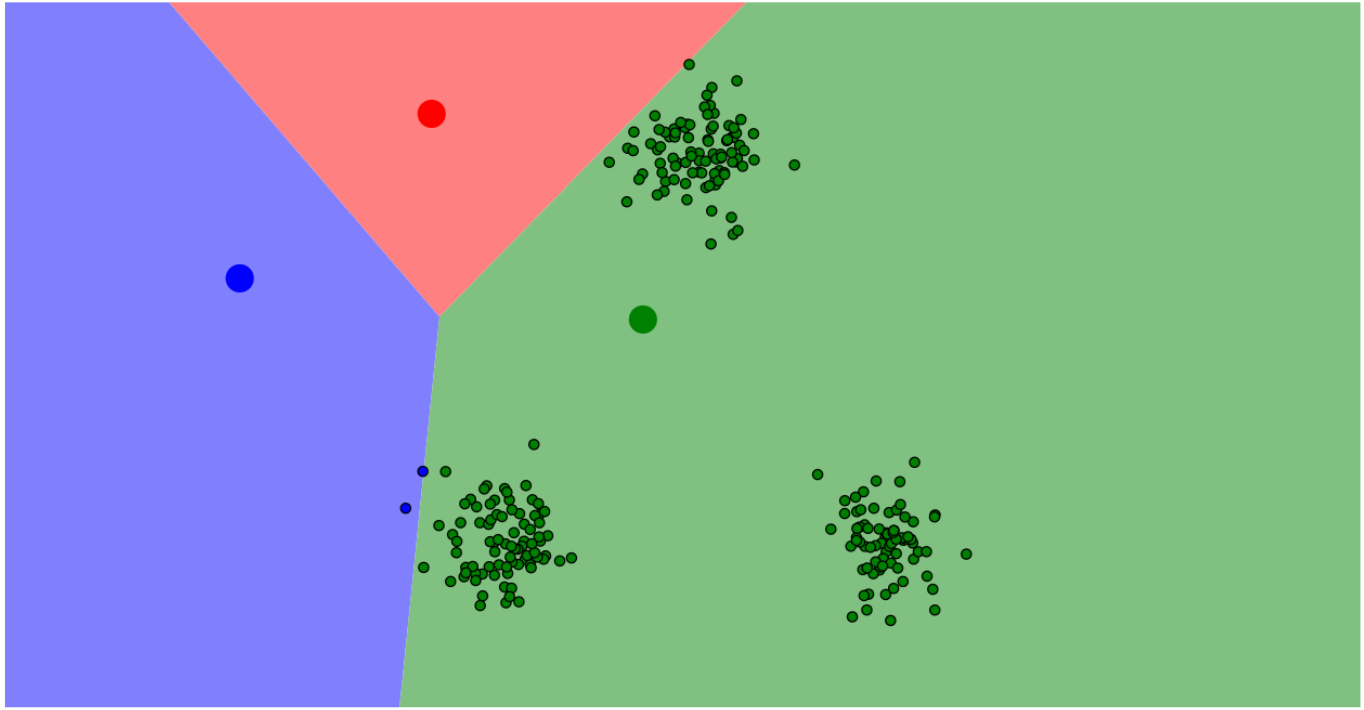
Step B Update each cluster center by replacing it with the mean of all points assigned to that cluster (in step A).

Repeat steps A and B until the centers converge to a stable solution.



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

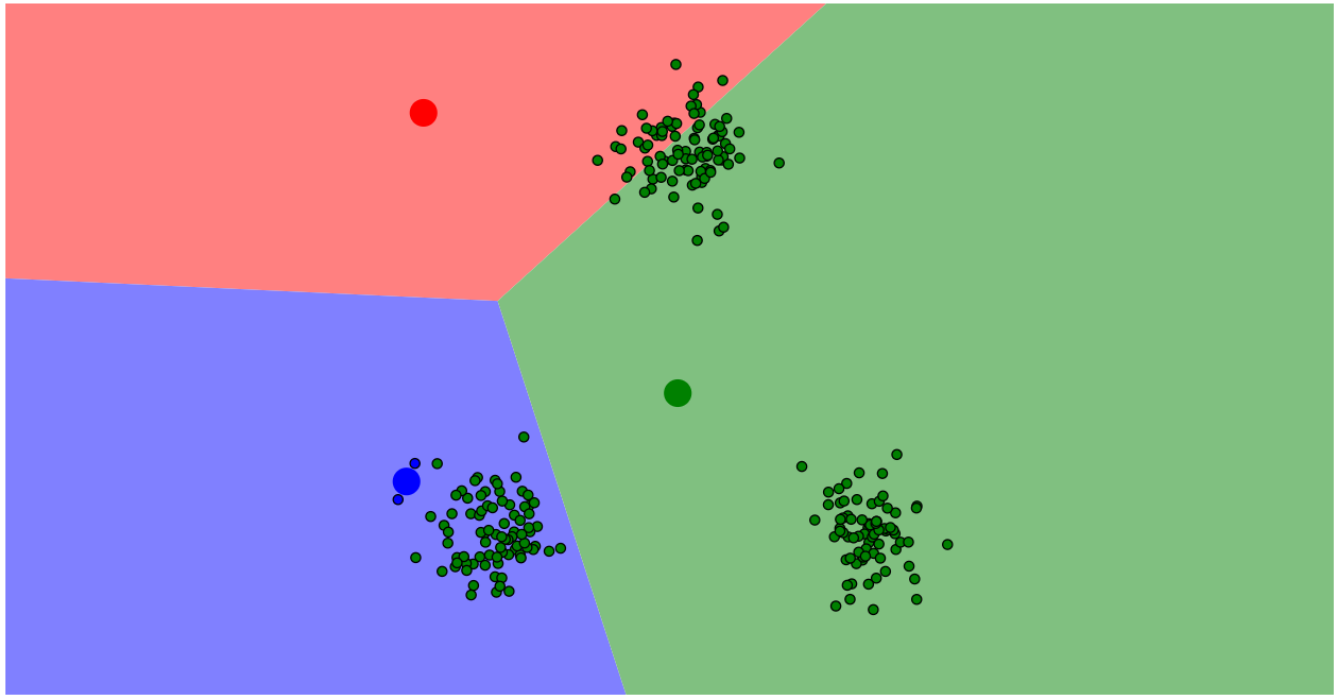
K-means Example: Step 1A



We want three clusters, so three centers are chosen randomly.

Data points are colored according to the closest center.

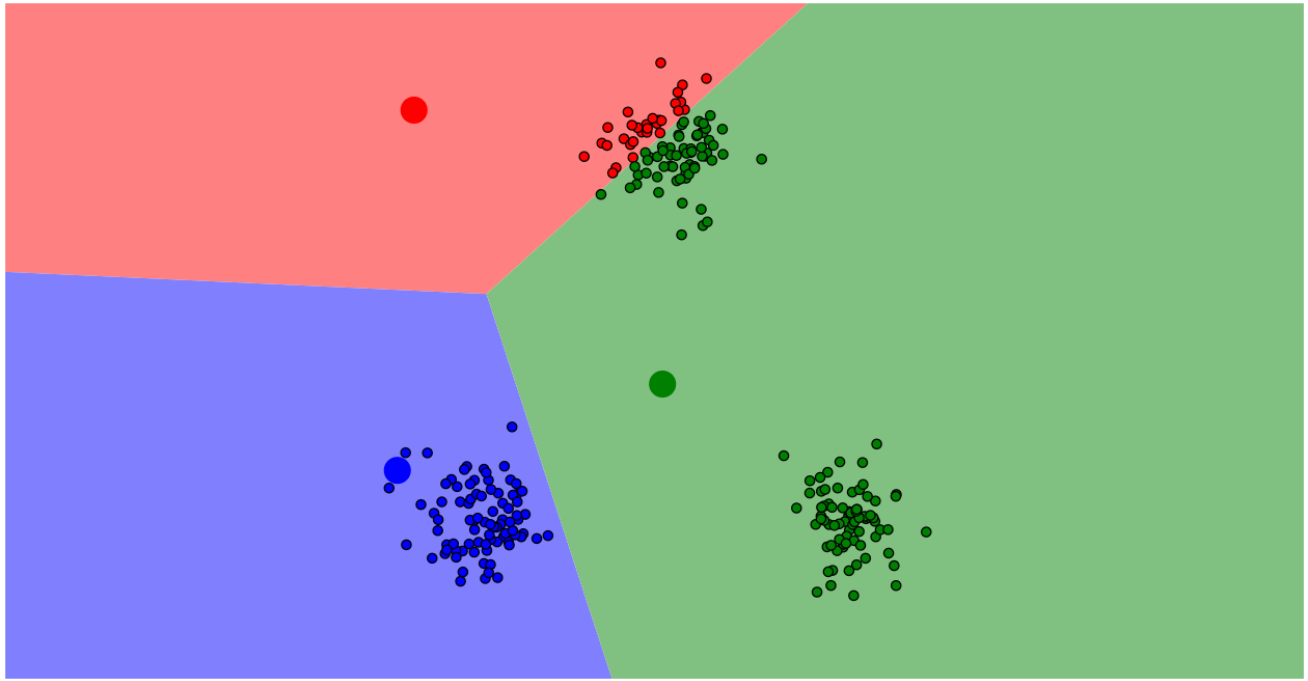
K-means Example: Step 1B



Each center is then updated…

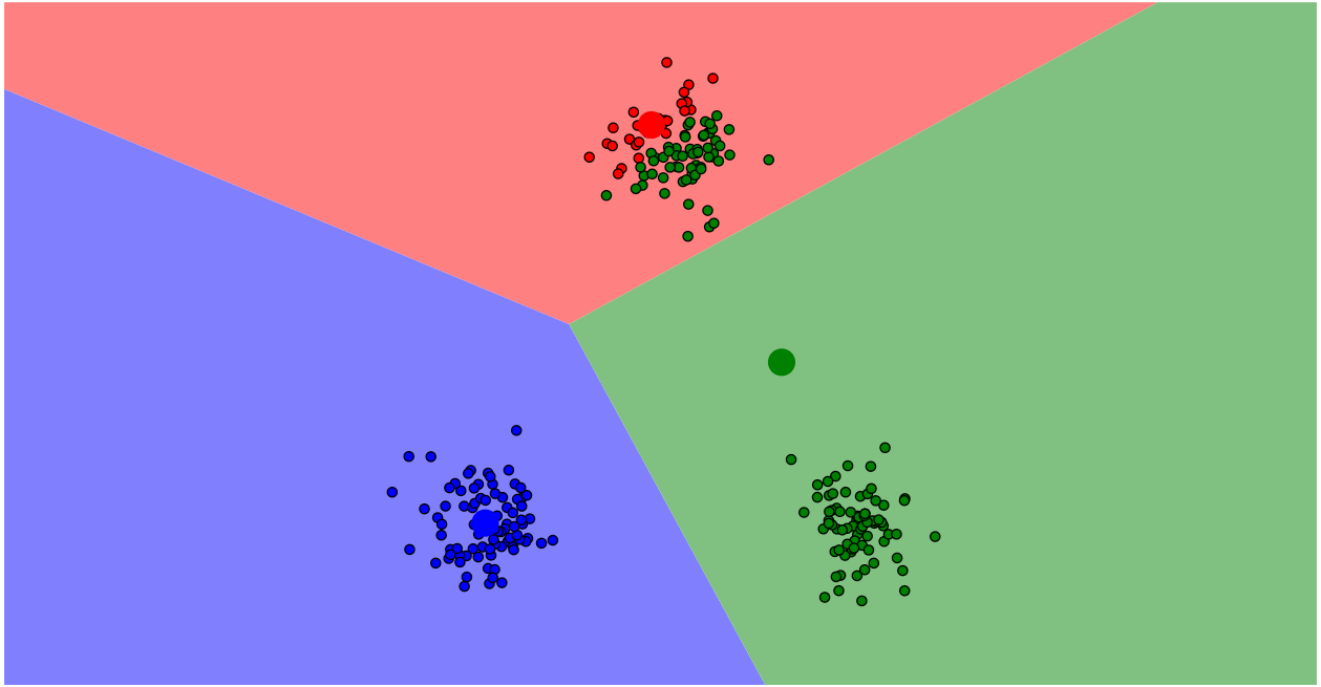
… using the mean of all points assigned to that cluster.

K-means Example: Step 2A



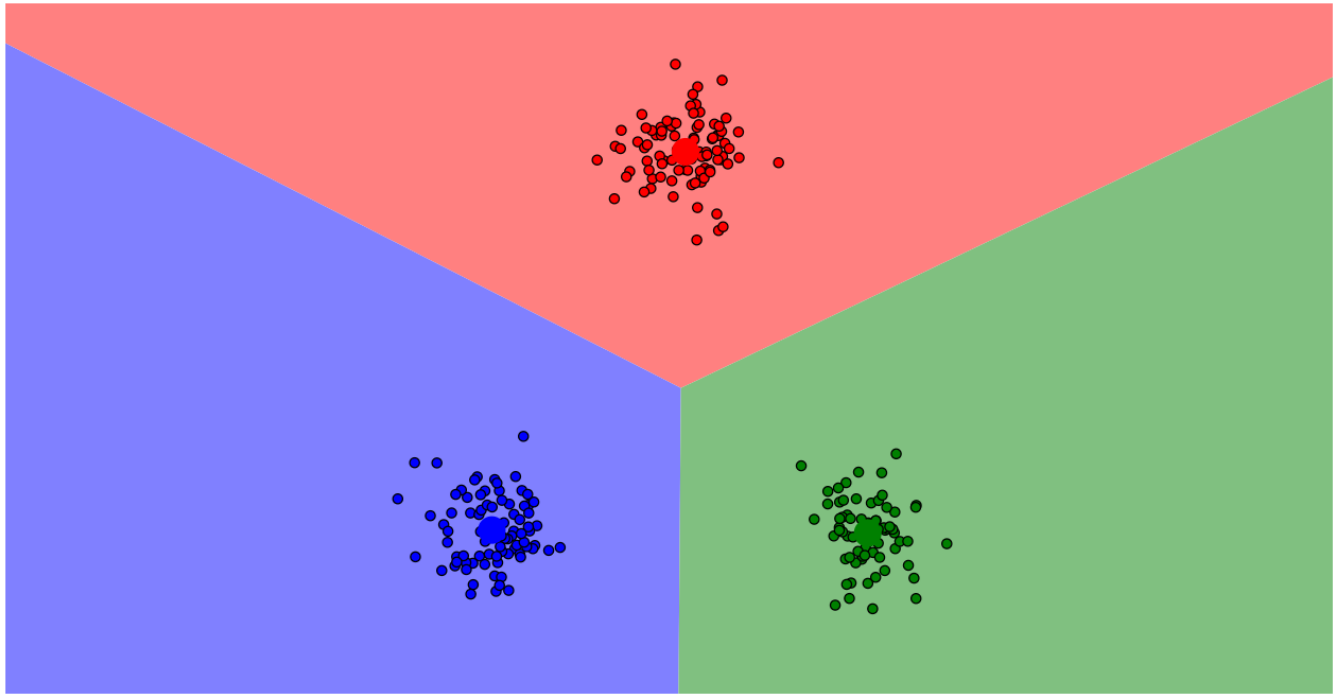
Data points are colored (again) according to the closest center.

K-means Example: Step 2B



Re-calculate all cluster centers.

K-means Example: Converged



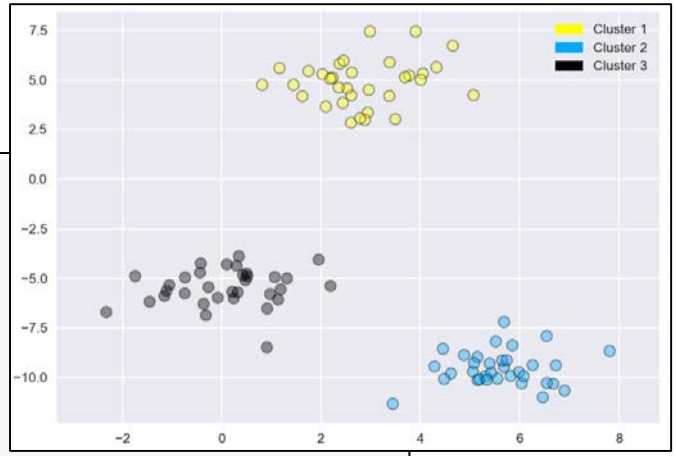
After repeating these steps for several more iterations…

The centers converge to a stable solution!

These centers define the final clusters.

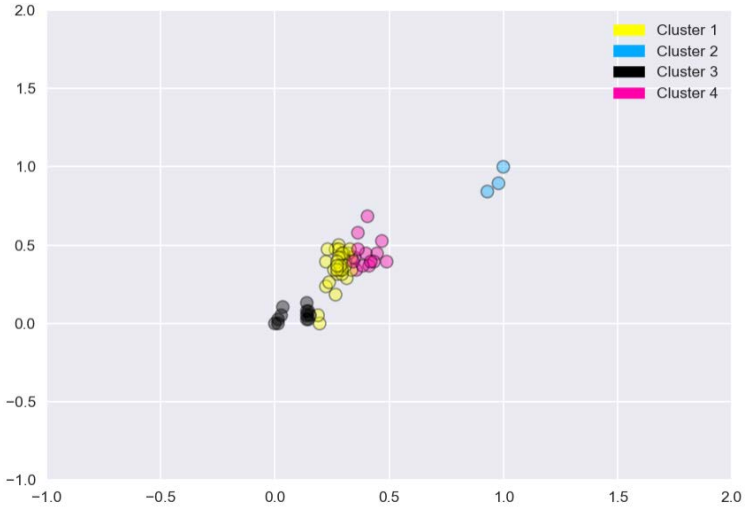
k-means Example in Scikit-Learn

```
from sklearn.datasets import make_blobs
from sklearn.cluster import KMeans
from adspy_shared_utilities import plot_labelled_scatter

X, y = make_blobs(random_state = 10)

kmeans = KMeans(n_clusters = 3)
kmeans.fit(X)

plot_labelled_scatter(X, kmeans.labels_, ['Cluster 1', 'Cluster 2', 'Cluster 3'])
```



k-means Output on the Fruits Dataset

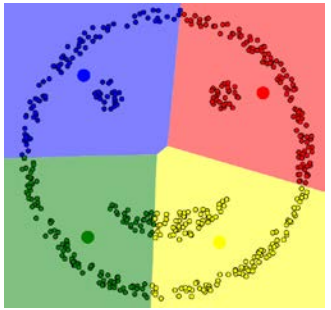
```
from sklearn.datasets import make_blobs
from sklearn.cluster import KMeans
from adspy_shared_utilities import plot_labelled_scatter
from sklearn.preprocessing import MinMaxScaler

fruits = pd.read_table('fruit_data_with_colors.txt')
X_fruits = fruits[['mass','width','height', 'color_score']].as_matrix()
y_fruits = fruits[['fruit_label']] - 1

X_fruits_normalized = MinMaxScaler().fit(X_fruits).transform(X_fruits)

kmeans = KMeans(n_clusters = 4, random_state = 0)
kmeans.fit(X_fruits)

plot_labelled_scatter(X_fruits_normalized, kmeans.labels_,
                      ['Cluster 1', 'Cluster 2', 'Cluster 3', 'Cluster 4'])
```



Can you interpret how these clusters correspond with the true fruit labels?

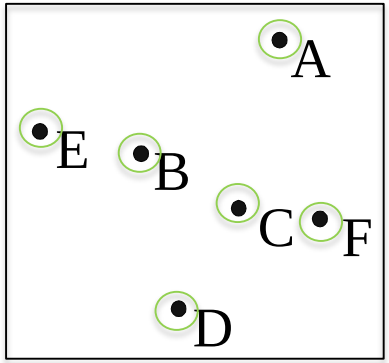
Limitations of k-means

- **Works well for simple clusters that are same size, well-separated, globular shapes.**
- **Does not do well with irregular, complex clusters.**
- **Variants of k-means like k-medoids can work with categorical features.**

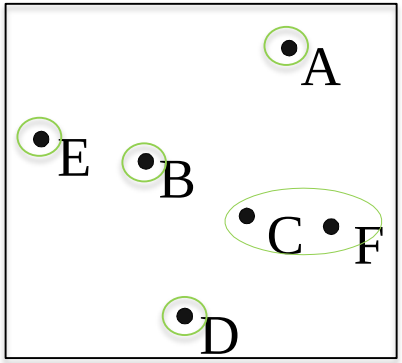


K-means typically performs poorly with data having complex, irregular clusters.

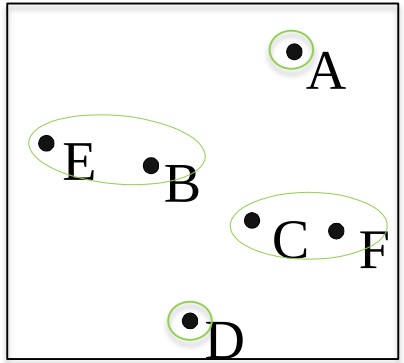
Agglomerative Clustering Example



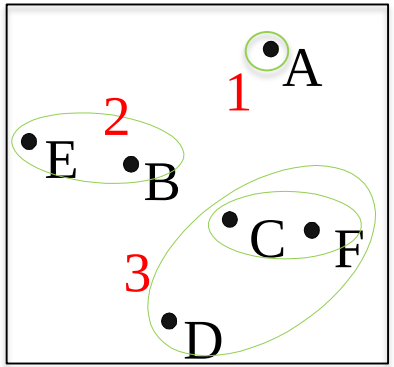
Stage 1



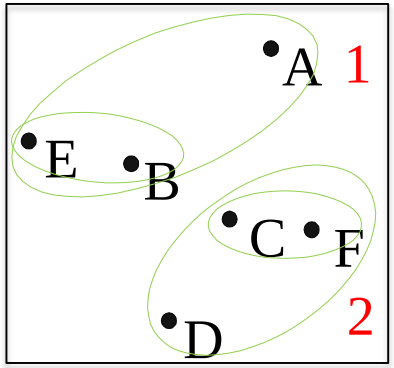
Stage 2



Stage 3



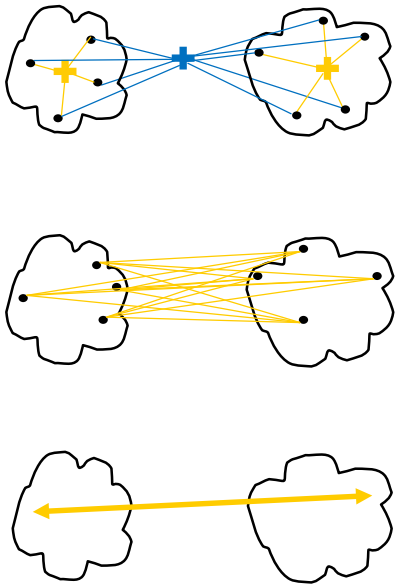
Stage 4
(three top-level clusters)

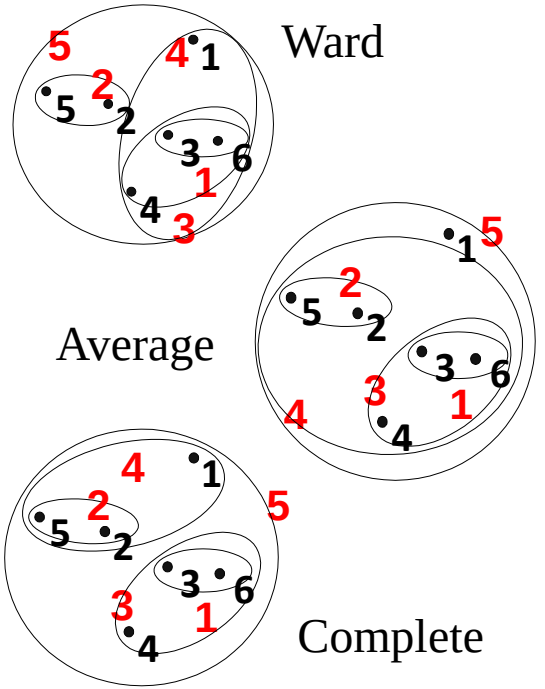


Stage 5 (two top-level clusters)

Linkage Criteria for Agglomerative Clustering

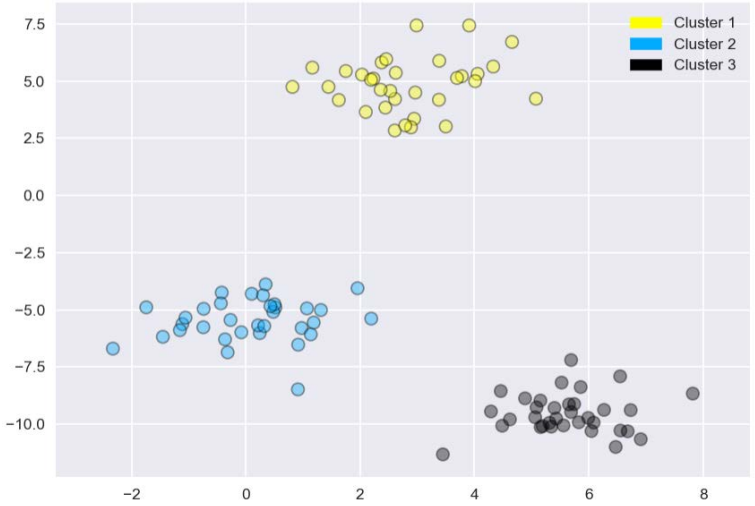
- **Ward's method**
  - *Least increase in total variance (around cluster centroids)*
- **Average linkage**
  - *Average distance between clusters*
- **Complete linkage**
  - *Max distance between clusters*



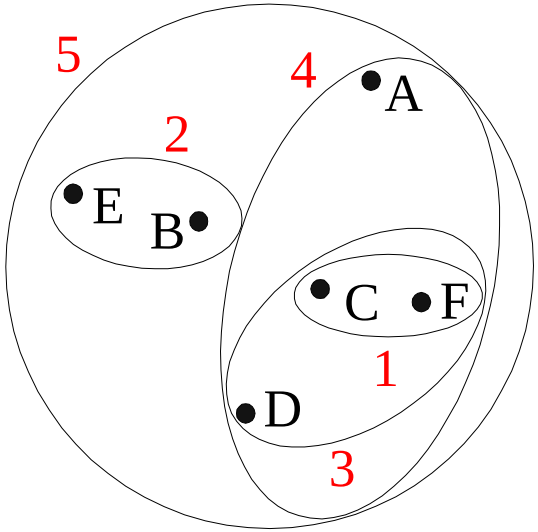


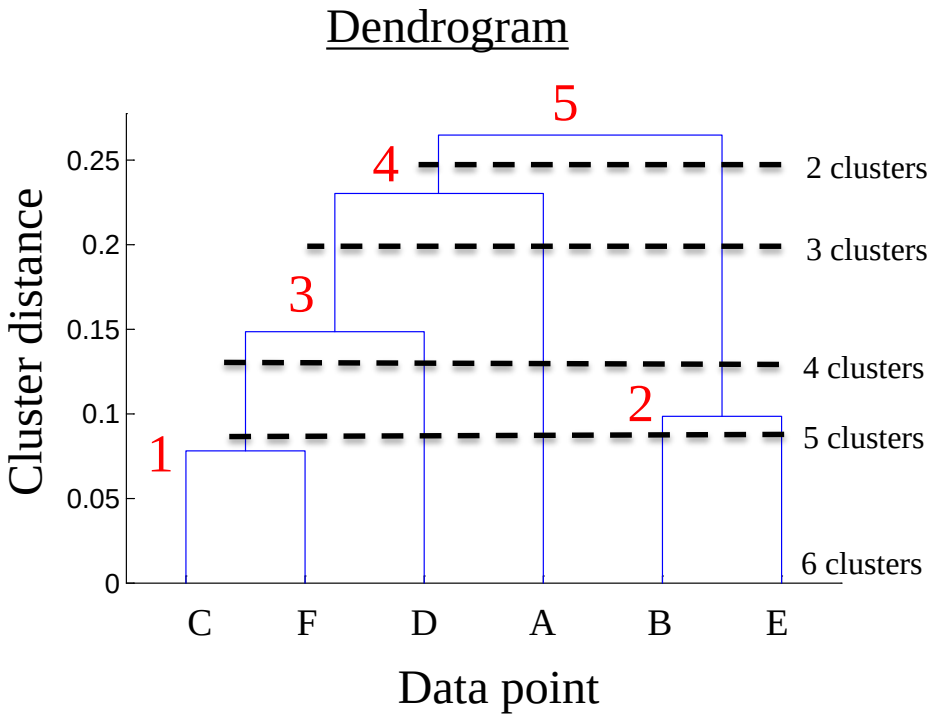
Agglomerative Clustering in Scikit-Learn

```
from sklearn.datasets import make_blobs
from sklearn.cluster import AgglomerativeClustering
from adspy_shared_utilities import plot_labelled_scatter

X, y = make_blobs(random_state = 10)

cls = AgglomerativeClustering(n_clusters = 3)
cls_assignment = cls.fit_predict(X)

X, y = make_blobs(random_state = 10)
plot_labelled_scatter(X, cls_assignment,
        ['Cluster 1', 'Cluster 2', 'Cluster 3'])
```



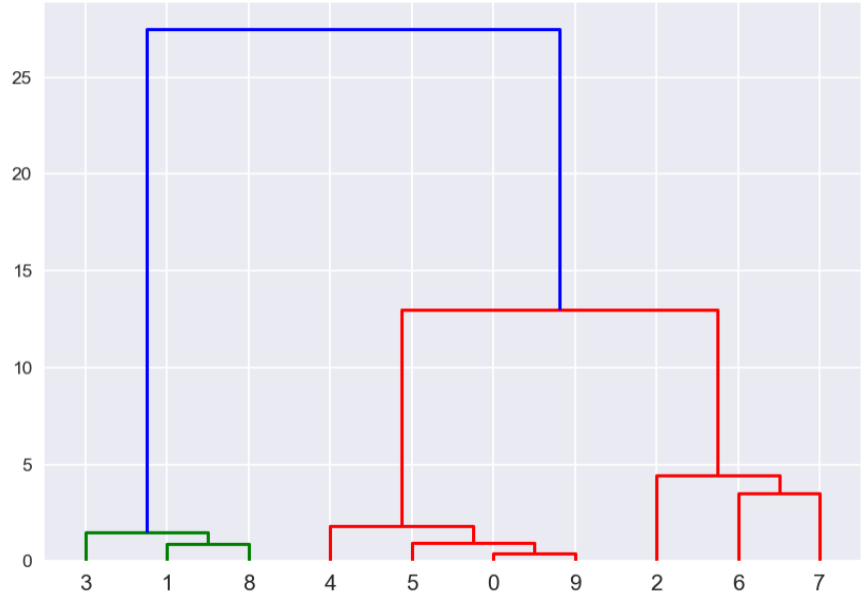
Hierarchical Clustering





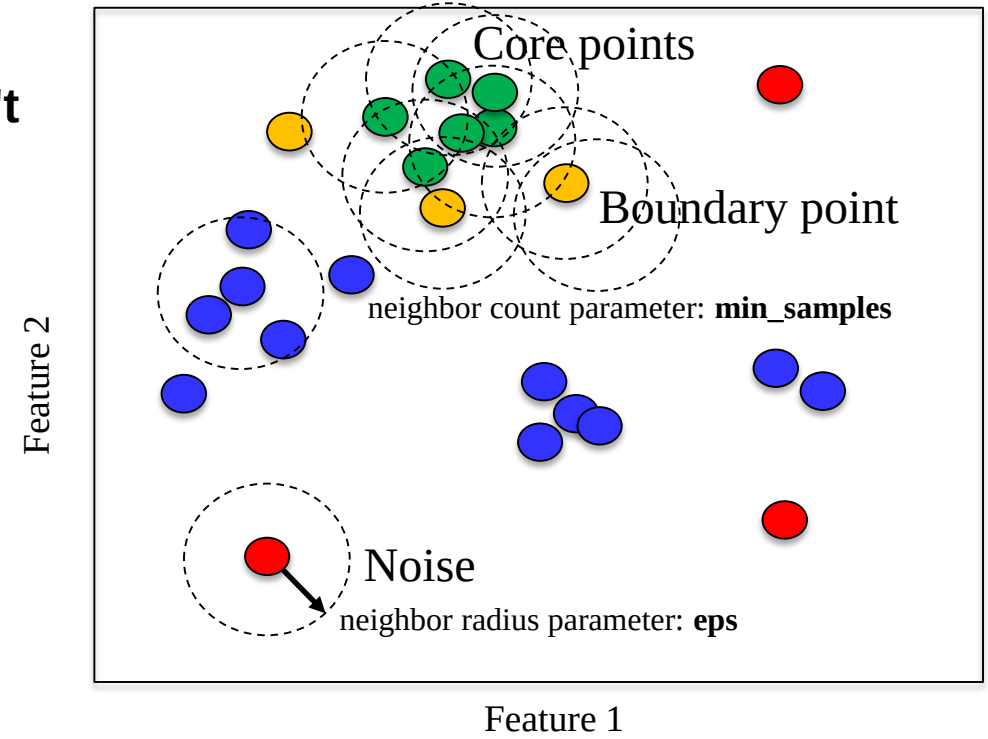
Dendrogram Example

```
from scipy.cluster.hierarchy import ward, dendrogram
from sklearn.datasets import make_blobs
from sklearn.cluster import AgglomerativeClustering

X, y = make_blobs(random_state = 10, n_samples = 10)

plt.figure()
dendrogram(ward(X))
plt.show()
```



DBSCAN Clustering

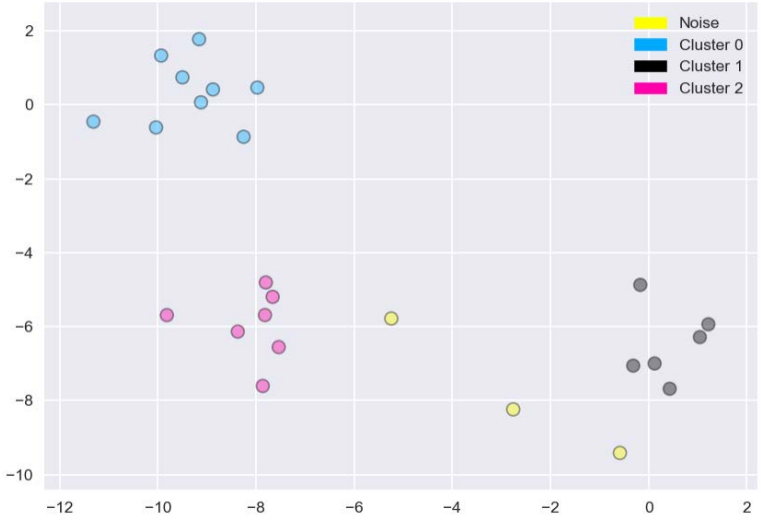
- **Unlike k-means, you don't need to specify # of clusters**
- **Relatively efficient – can be used with large datasets**
- **Identifies likely noise points**



DBSCAN Example in Scikit-Learn

```
from sklearn.cluster import DBSCAN
from sklearn.datasets import make_blobs

X, y = make_blobs(random_state = 9, n_samples = 25)

dbscan = DBSCAN(eps = 2, min_samples = 2)

cls = dbscan.fit_predict(X)
print("Cluster membership values:\n{}", format(cls))

plot_labelled_scatter(X, cls + 1,
        ['Noise', 'Cluster 0', 'Cluster 1', 'Cluster 2'])
```

```
Cluster membership values:
{} [ 0  1  0  2  0  0  0  2  2 -1  1  2  0  0 -1  0  0  1 -1  1  1  2  2  2  1]
```



Clustering Evaluation

- With ground truth, existing labels can be used to evaluate cluster quality.
- Without ground truth, evaluation can difficult: multiple clusterings may be plausible for a dataset.
- Consider task-based evaluation: Evaluate clustering according to performance on a task that does have an objective basis for comparison.
- Example: the effectiveness of clustering-based features for a supervised learning task.
- Some evaluation heuristics exist (e.g. silhouette) but these can be unreliable.