# INF391 - Reconocimiento de Patrones en Minería de Datos


# Tarea 1: Técnicas de *Clustering*

**Francisca Ramírez**

**Juan Pablo Muñoz**

***17 de abril del 2019***

---

## Introducción

En esta tarea se exploran distintas técnicas de reconocimiento de patrones basadas en *clustering* vistas en cátedra. Para ello, se cuenta con tres pequeños *datasets* con distintas características, que servirán para contrastar la aptitud que cada técnica posee para cada caso.

Luego de la experimentación, se responden las dos preguntas conceptuales planteadas en el enunciado.

---

## Parte I

Primero, se prepara la ingesta de datos.

In [1]:
```
import os.path
import numpy as np

def ingest_dataset(txt_dir):
    dataset = list()
    if os.path.exists(txt_dir):
        with open(txt_dir, 'r') as f:
            for line in f.readlines():
                data_point = line.split()
                x_coord, y_coord = float(data_point[0]), float(data_point[1])
                dataset.append([x_coord, y_coord])
    return np.array(dataset)
```

Y se instancian los tres datasets.

In [2]:
```
smile = ingest_dataset('smile.txt')
mouse = ingest_dataset('mouse.txt')
spiral = ingest_dataset('spiral.txt')
```

---

A continuación, se procede a aplicar las técnicas de *clustering*.

#### 1. K-Means

In [5]:

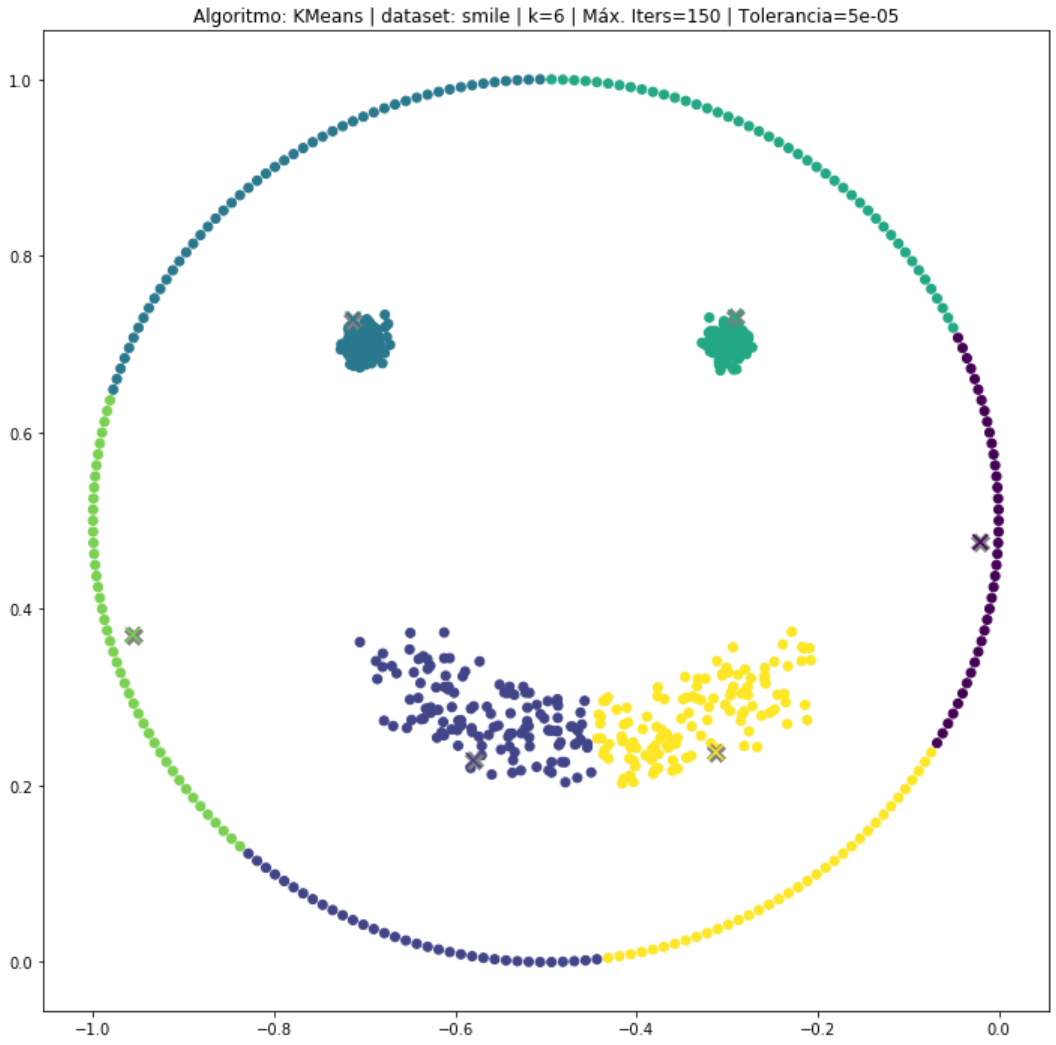
```
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt
from ipywidgets import interact
from ipywidgets import FloatSlider

def apply_kmeans(dataset, k, max_iterations=300, tolerance=1e-4):
    kmeans = KMeans(
        n_clusters=k,
        init='random',
        n_init=1,
        max_iter=max_iterations,
        tol=tolerance,
        random_state=0,
    )
    kmeans.fit(dataset)
    return kmeans.cluster_centers_, kmeans.labels_

@interact(
    dataset_name=['smile', 'mouse', 'spiral'],
    k=(2,10, 1),
    max_iterations=(10, 300, 10),
    tolerance=FloatSlider(min=5e-5, max=5e-4, step=5e-5, continuous_update=False),
)
def plot_kmeans(dataset_name, k, max_iterations, tolerance):
    if dataset_name == 'smile':
        dataset = smile
    elif dataset_name == 'mouse':
        dataset = mouse
    elif dataset_name == 'spiral':
        dataset = spiral
    centroids, labels = apply_kmeans(dataset, k, max_iterations, tolerance)
    plt.figure(figsize=(12,12))
    plt.scatter(dataset[:, 0], dataset[:, 1], c=labels)
    plt.scatter(centroids[:, 0], centroids[:, 1], marker='X', s=150, linewidths=.5, c='gray')
    plt.scatter(centroids[:, 0], centroids[:, 1], marker='x', s=100, linewidths=2, c=list(range(len(centr
    plt.title('Algoritmo: KMeans | dataset: {} | k={} | Máx. Iters={} | Tolerancia={}'.format(dataset_nam

```

dataset_na… smile

k 6

max_iterati… 150

tolerance 0.00




**Análisis K-Means**

Bla...

---

**2. Agglomerative Hierarchical Clustering**

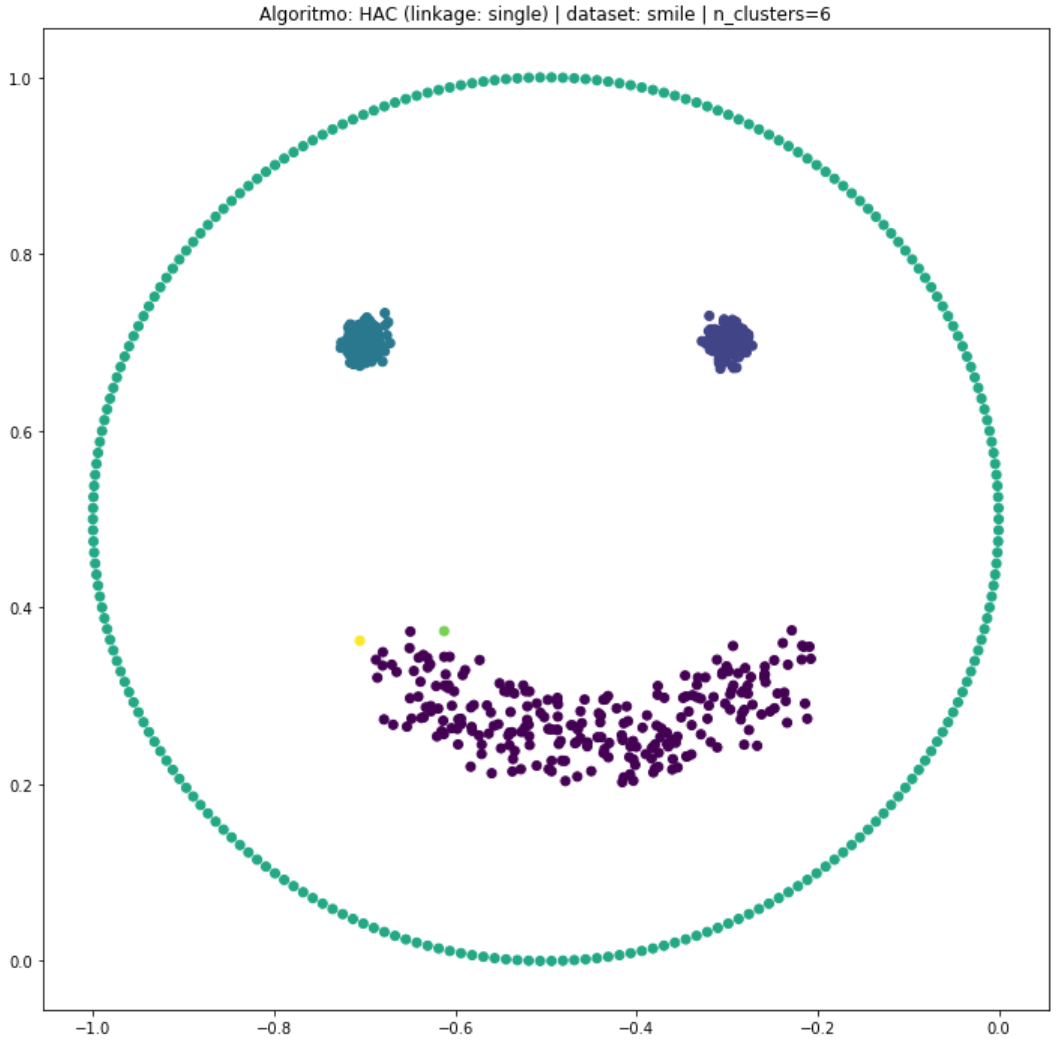
In [4]:

```
from sklearn.cluster import AgglomerativeClustering

def apply_hac(dataset, linkage, n_clusters):
    hac = AgglomerativeClustering(n_clusters=n_clusters, linkage=linkage)
    hac.fit(dataset)
    return hac.labels_

@interact(
    dataset_name=['smile', 'mouse', 'spiral'],
    linkage=['single', 'complete'],
    n_clusters=(2,10, 1),
)
def plot_hac(dataset_name, linkage, n_clusters):
    if dataset_name == 'smile':
        dataset = smile
    elif dataset_name == 'mouse':
        dataset = mouse
    elif dataset_name == 'spiral':
        dataset = spiral

    labels = apply_hac(dataset, linkage, n_clusters)
    plt.figure(figsize=(12,12))
    plt.scatter(dataset[:, 0], dataset[:, 1], c=labels)
    plt.title('Algoritmo: HAC (linkage: {}) | dataset: {} | n_clusters={}'.format(linkage, dataset_name,
```

dataset_na… smile

linkage single

n_clusters 6




**Análisis Agglomerative Hierarchical Clustering**

Bla...

In [ ]:
```
1
```