

# Reducción de Dimensionalidad & Aprendizaje No Supervisado

PCA | K-Means, DBSCAN

Dr. Jesus Alvarado Huayhuaz

ISIL  
BCP

# Outline

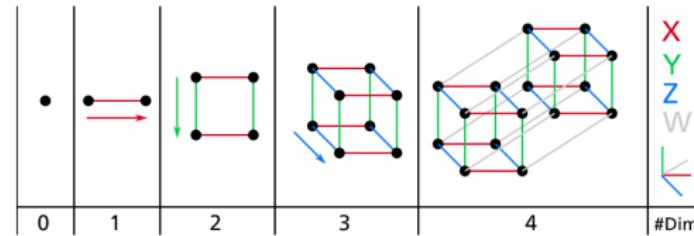
## ① Reducción de la dimensionalidad

## ② PCA



# La maldición de la dimensionalidad<sup>1</sup>

- 1 Frase atribuida por Richard Bellman (1956) para referirse al problema de tener un exceso de atributos en un dataset.
- 2 La distancia media entre dos puntos aleatorios en un cuadrado unitario es aproximadamente 0.52, en un cubo unitario 3D es 0.66 ¿Y en un hipercubo de 1 millón de dimensiones?
- 3 Los conjuntos de datos de alta dimensión corren el riesgo de estar muy dispersos, es decir, es probable que una instancia nueva esté alejada de cualquier instancia de entrenamiento (extrapolaciones grandes)
- 4 Cuantas más dimensiones tenga el conjunto de entrenamiento mayor será el riesgo de sobreajustarlo.

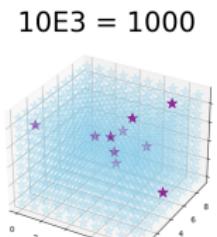
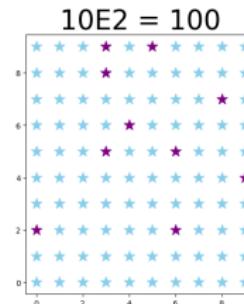
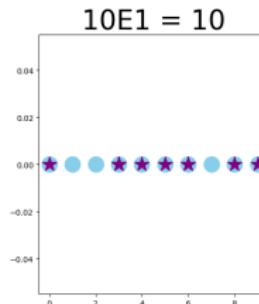


<sup>1</sup>Géron, A. (2019). Hands-on machine learning with Scikit-Learn, Keras and TensorFlow: concepts, tools, and techniques to build intelligent systems (2nd ed.)

Revisar el COLAB: tiempo estimado 5 minutos

# La maldición de la dimensionalidad<sup>2</sup>

- 1 En el ejemplo del colab adjunto, de las 10 posibles configuraciones en 1D, prácticamente el 70 % está ocupada.
- 2 Cuando el espacio de configuraciones crece, como en el ejemplo (10, 100 y 1000), es más difícil el reconocimiento de patrones, pero cuán difícil.
- 3 Una solución podría ser aumentar el tamaño del conjunto de entrenamiento, para mejorar la densidad de instancias de entrenamiento, sin embargo, estas crecen exponencialmente.



<sup>2</sup>[https://github.com/inefable12/upch\\_rp\\_2023/](https://github.com/inefable12/upch_rp_2023/)

# Enfoques principales para la reducción de la dimensionalidad

- 1 Proyección: En la mayoría de problemas del mundo real, las instancias de entrenamiento no se extienden de manera uniforme a través de todas las dimensiones.
- 2 Muchas características pueden presentar alta correlación o ser casi constantes.
- 3 En general, las instancias de entrenamiento pueden quedar dentro o cerca de un subespacio con muchas menos dimensiones del espacio de alta dimensión.

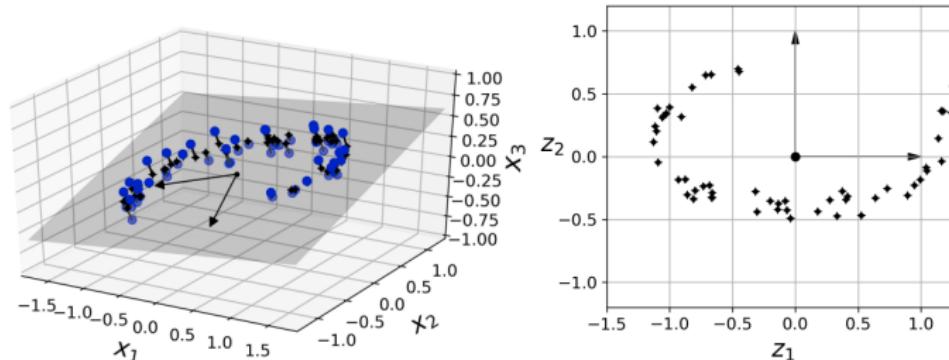


Figura 1: Conjunto de datos 3D (izquierda) y su proyección en 2D (derecha)

# Enfoques principales para la reducción de la dimensionalidad

- 1 Aprendizaje de variedades: Variedad 2D  $\leftarrow$  Swiss roll con  $d=2$ ,  $n=3$

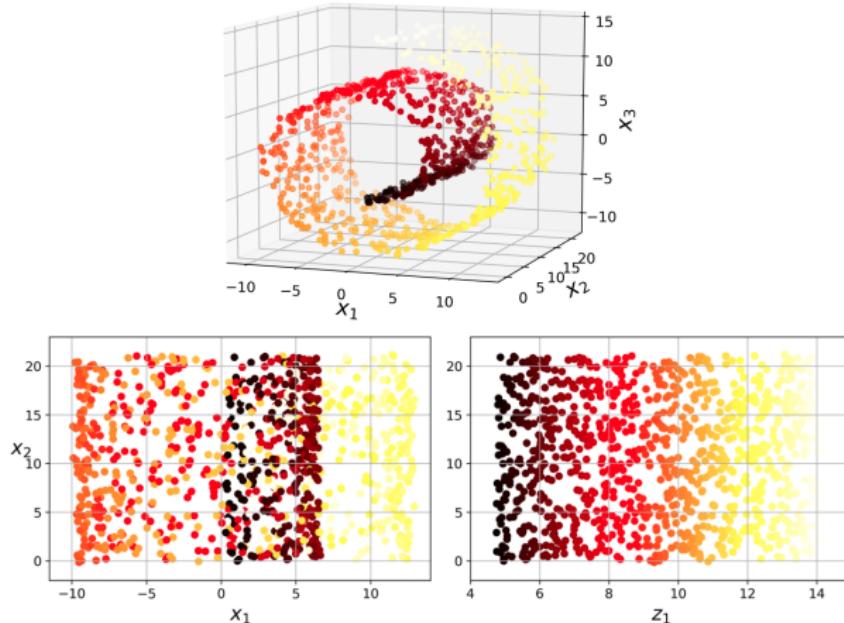


Figura 2: (Arriba) Datos con estructura Swiss roll. (abajo) Proyección 2D versus desenrollado

MNIST contiene como clases a números del 1 al 9 escritos a mano

## Enfoques principales para la reducción de la dimensionalidad



- 1 Hipótesis de variedades: Sostiene que la mayoría de datasets de alta dimensión del mundo real quedan cerca de una variedad con muchas menos dimensiones (observación empírica).
- 2 MNIST: todas son líneas conectadas, centradas y con bordes blancos. Y si generásemos imágenes aleatorias?
- 3 Reducir la dimensionalidad del conjunto de entrenamiento antes de entrenar el modelo suele acelerar el entrenamiento, pero puede que no siempre lleve a una solución mejor o más simple.

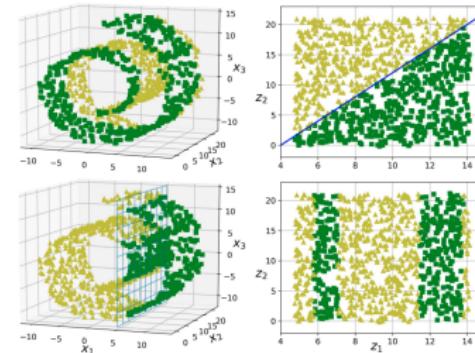


Figura 3: El límite de decisión no siempre será más simple con menos dimensiones

# Preservar la varianza

- ① ¿Cómo encontrar aquella superficie que proyecte mejor mi dataset?
- ② La proyección en la línea continua C1 (hiperplano 1D) es más amplia o dispersa que en C1' y C2.
- ③ Podemos decir que C1 preserva la varianza máxima y que lo más probable es que pierda menos información que las otras proyecciones.
- ④ Otra justificativa es que C1 minimiza la distancia cuadrática media entre el dataset original y su proyección en ese eje.

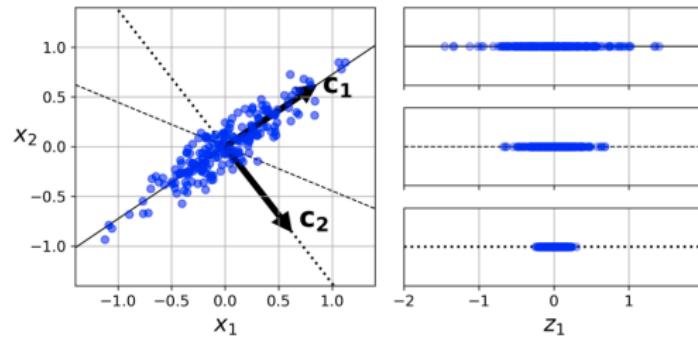


Figura 4: Seleccionar el subespacio en el que proyectar

# Análisis de Componentes Principales <sup>3</sup>

- ① PCA identifica el eje que representa la cantidad más grande de varianza en el dataset de entrenamiento.
- ② En la figura 4, C2 representa la cantidad más grande de varianza restante y es perpendicular a C1 (son linealmente independientes). Si existiese otra dimensión, el PCA encontraría un tercer eje C3, perpendicular a los dos anteriores, y así sucesivamente.

$$\mathbf{V} = \begin{pmatrix} | & | & | \\ \mathbf{c}_1 & \mathbf{c}_2 & \cdots & \mathbf{c}_n \\ | & | & | \end{pmatrix}$$

Singular Value Decomposition (SVD), disponible en NumPy, es una técnica de factorización de matrices estándar que puede descomponer la matriz X del conjunto de entrenamiento en la multiplicación de matrices U,  $\Sigma$ , y V, que contiene los vectores unitarios que definen los componentes principales.

```
X_centered = X - X.mean(axis=0)
U, s, Vt = np.linalg.svd(X_centered)
c1 = Vt.T[:, 0]
c2 = Vt.T[:, 1]
```

<sup>3</sup>Karl Pearson F.R.S. (1901). On lines and planes of closest fit to systems of points in space.

# Reducción de Dimensionalidad & Aprendizaje No Supervisado

PCA | K-Means, DBSCAN

Dr. Jesus Alvarado Huayhuaz

ISIL  
BCP

## Proyección para bajar a $d$ dimensiones

- 1 Una vez que se ha identificado todos los componentes principales, puedes reducir la dimensionalidad del conjunto de datos a  $d$  dimensiones proyectándolo en el hiperplano definido por los primeros  $d$  componentes principales.

$$\mathbf{X}_{d\text{-proj}} = \mathbf{X} \mathbf{W}_d$$

El siguiente código Python proyecta el conjunto de entrenamiento en el plano definido por los dos primeros componentes principales.

```
W2 = Vt.T[:, :2]
X2D = X_centered.dot(W2)
```

La clase PCA de Scikit-Learn utiliza SVD para implementar el PCA (centraliza los datos automáticamente)

```
from sklearn.decomposition import PCA

pca = PCA(n_components = 2)
X2D = pca.fit_transform(X)
```

## Ratio de varianza explicada

- ① Otra información útil es el ratio de la varianza explicada de cada componente principal, disponible mediante la variable *explained\_variance\_ratio\_*.
- ② El ratio indica la proporción de la varianza del conjunto de datos que queda a lo largo de cada componente principal.
- ③ Por ejemplo, en la Fig. 1, los dos primeros componentes presentan 84.2 % y 14.6 % de la varianza del conjunto de datos, respectivamente.

```
>>> pca.explained_variance_ratio_
array([0.84248607, 0.14631839])
```

## Elegir el número adecuado de dimensiones

- 1 Podemos realizar el análisis de componentes principales sin reducir la dimensionalidad y calcular el número de dimensiones requeridas para preservar el 95 % de la varianza.

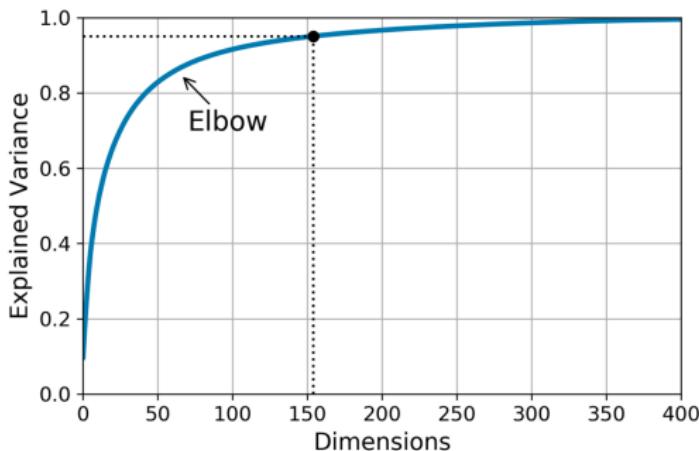


Figura 1: Varianza explicada como función del número de dimensiones

# PCA para compresión<sup>1</sup>

- ① Tras la reducción de dimensionalidad el conjunto de entrenamiento ocupa mucho menos espacio.
- ② Por ejemplo, MNIST conserva 150 características (en lugar de las 784 originales) para preservar el 95 % de su varianza.
- ③ Ahora el dataset es menor al 20 % de su tamaño original.
- ④ La descompresión requiere una transformación inversa y tiene asociado un *error de construcción* con leve pérdida en la calidad de imagen.
- ⑤ PCA aleatorizado, gradual, Kernel PCA, Locally Linear Embedding, etc.

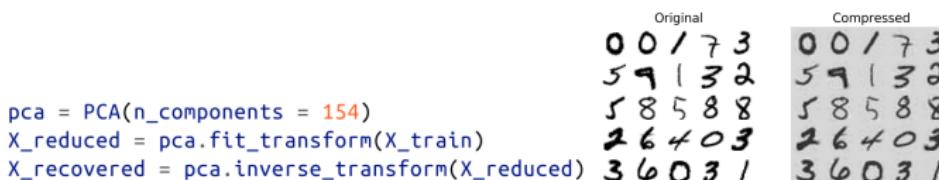
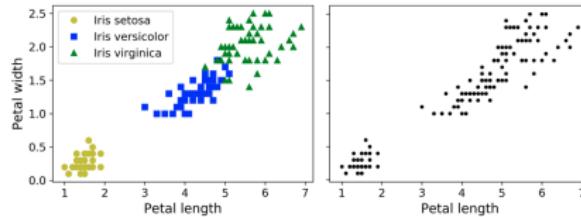


Figura 2: Varianza explicada como función del número de dimensiones

<sup>1</sup>[github.com/inefable12/ML\\_no\\_supervisado/blob/main/Kmeans\\_Auder\\_PZAdler\\_Curcuminoideas.ipynb](https://github.com/inefable12/ML_no_supervisado/blob/main/Kmeans_Auder_PZAdler_Curcuminoideas.ipynb)

# Técnicas de aprendizaje no supervisado

- ① Agrupamiento: Tiene como objetivo juntar instancias similares en grupos. Tiene aplicaciones en genética de poblaciones, detección de moléculas con potencial farmacológico, segmentación de mercado, sistemas de recomendación, motores de búsqueda, etc.
- ② Detección de anomalías: Tiene como objetivo aprender qué aspecto tienen los datos 'normales' y, después, utilizar eso para detectar instancias anormales, como por ejemplo, artículos defectuosos en una cadena de producción.
- ③ Estimación de la densidad: Se utiliza con frecuencia en la detección de anomalías. Es probable que las instancias ubicadas en regiones de densidad muy baja sean anomalías. También resulta útil para la visualización y el análisis de datos.



# K-Medias

- ① Lo propuso Stuart Lloyd en Bell Labs en 1957 pero no se publicó fuera de la empresa hasta 1982.
- ② En 1965, Edward W. Forgy había publicado un algoritmo muy parecido, así que se conoce también como Lloyd-Forgy.

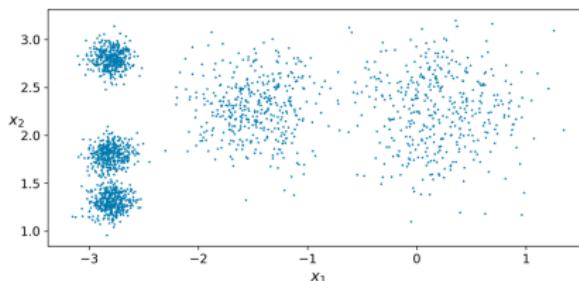


Figura 3: Dataset sin etiquetar compuesto por 5 masas de instancias

```
from sklearn.cluster import KMeans
k = 5
kmeans = KMeans(n_clusters=k)
y_pred = kmeans.fit_predict(X)
>>> kmeans.cluster_centers_
array([[-2.80389616,  1.80117999],
       [ 0.20876306,  2.25551336],
       [-2.79290307,  2.79641063],
       [-1.46679593,  2.28585348],
       [-2.80037642,  1.30082566]])
```

## K-Medias<sup>2</sup>

- Podemos asignar con facilidad nuevas instancias al grupo cuyo centroide esté más cerca:

```
>>> X_new = np.array([[0, 2], [3, 2], [-3, 3], [-3, 2.5]])
>>> kmeans.predict(X_new)
array([1, 1, 2, 2], dtype=int32)
```

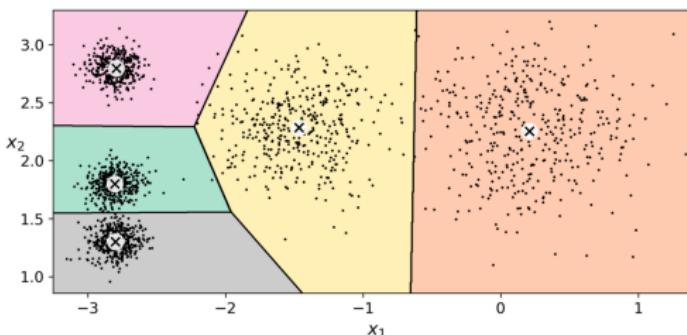


Figura 4: Límites de decisión de K-Medias

<sup>2</sup>[https://github.com/inefable12/ML\\_no\\_supervisado/blob/main/Kmeans\\_jesus.ipynb](https://github.com/inefable12/ML_no_supervisado/blob/main/Kmeans_jesus.ipynb)

# Reducción de Dimensionalidad & Aprendizaje No Supervisado

PCA | K-Means, DBSCAN

Dr. Jesus Alvarado Huayhuaz

ISIL  
BCP

- ① Agrupamiento duro: asignar cada instancia a un solo grupo
- ② Agrupamiento suave: asignar una puntuación por grupo.
- ③ Esta puntuación puede ser la distancia entre la instancia y el centroide.
- ④ El método transform() mide la distancia entre cada instancia a todos los centroides.

```
>>> kmeans.transform(X_new)
array([[2.81093633, 0.32995317, 2.9042344 , 1.49439034, 2.88633901],
       [5.80730058, 2.80290755, 5.84739223, 4.4759332 , 5.84236351],
       [1.21475352, 3.29399768, 0.29040966, 1.69136631, 1.71086031],
       [0.72581411, 3.21806371, 0.36159148, 1.54808703, 1.21567622]])
```

La primera instancia en X\_new se encuentra a 2.81 del primer centroide, 0.33 del 2do, 2.90 del 3ro, etc. Si empleamos esto en un dataset de alta dimensión obtenemos un conjunto de datos de K dimensiones, por ello, **K-Medias también es empleado como técnica de reducción de la dimensionalidad muy eficiente.**

# El algoritmo K-Medias

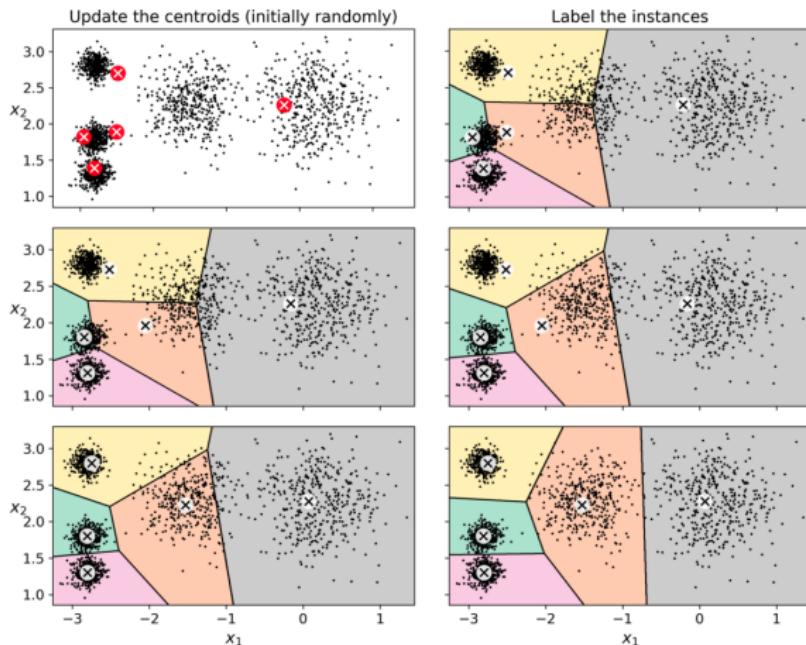


Figura 1: El algoritmo K-Medias

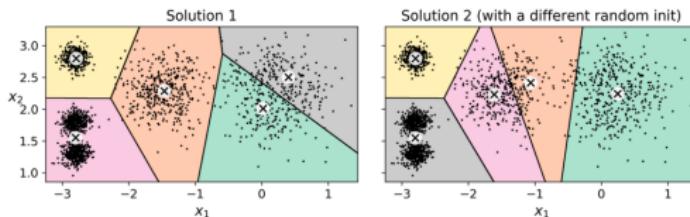


Figura 2: Soluciones que no son óptimas debido a inicializaciones poco afortunadas

- 1 Si se tiene una idea aproximada de donde deberían estar los centroides, se puede usar *init*.

```
good_init = np.array([[-3, 3], [-3, 2], [-3, 1], [-1, 2], [0, 2]])
kmeans = KMeans(n_clusters=5, init=good_init, n_init=1)
```

- 2 Otra solución es ejecutar el algoritmo varias veces con diferentes inicializaciones y mantener la mejor solución, esto se consigue con *n\_init*.
- 3 Por defecto *n\_init* es 10, es decir, se ejecuta 10 veces cuando llamamos a fit y Scikit-Learn selecciona la mejor solución empleando una métrica de rendimiento llamada *inercia* (o distancia cuadrática media entre cada instancia y su centroide más próximo). Se selecciona la menor inercia.

- ① La inercia de un modelo es accesible a través de la variable de instancia:

```
>>> kmeans.inertia_
211.59853725816856
```

- ② El método score() devuelve la inercia negativa, con lo cual se facilitan la interpretación *cuanto más grande mejor*.
- ③ Es decir, si un método es mejor que otro, entonces su método score() debería devolver una puntuación mayor.

```
>>> kmeans.score(X)
-211.59853725816856
```

- ④ David Arthur y Sergei Vassilvitskii introdujeron K-Medias++ en el 2006, que emplea centroides alejados entre sí, reduciendo las probabilidades de converger en una solución no óptima.

---

<sup>1</sup> David Arthur and Sergei Vassilvitskii, k-Means++: The Advantages of Careful Seeding, Proceedings of the 18th Annual ACM-SIAM Symposium on Discrete Algorithms (2007)

- ① Otra mejora fue realizada por Charles Elkan en el 2003, en la cual emplea desigualdad triangular para reducir el número de cálculos de distancia innecesarios.
- ② Este es el algoritmo que la clase KMeans utiliza por defecto.
- ③ Otra variante, por David Sculley (2010) consiste en no emplear todo el dataset en cada iteración, sino minilotes, moviendo los centroides lentamente pero acelerando el algoritmo y permitiendo agrupar conjuntos de dataset enormes.

```
from sklearn.cluster import MiniBatchKMeans  
  
minibatch_kmeans = MiniBatchKMeans(n_clusters=5)  
minibatch_kmeans.fit(X)
```

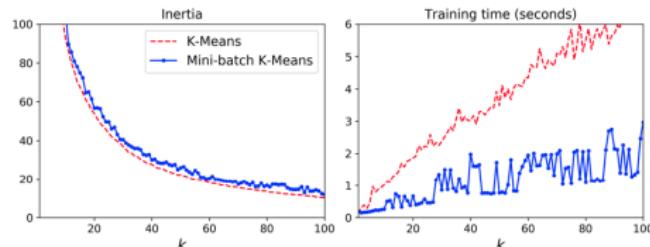
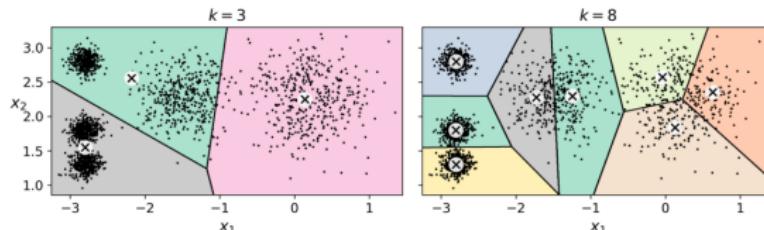


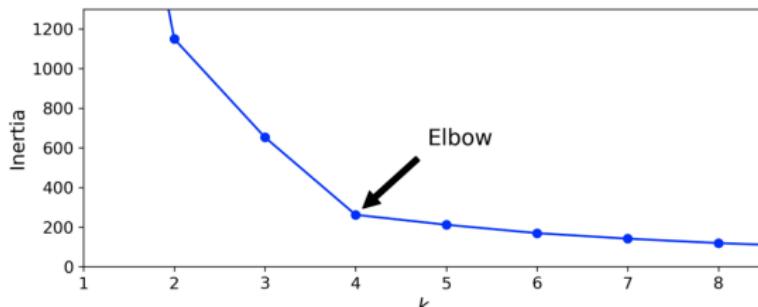
Figura 3: Mini-batch tiene mayor inercia pero es más rápido que K-Medias

# Encontrar el número de grupos óptimo

- 1 Por visualización escogimos  $k=5$  pero si cambiamos?



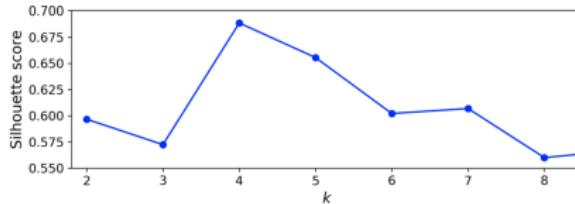
- 2 La inercia para  $k=3$  es 653 y para  $k=5$  es 212, pero para  $k=8$  es 119, por ello, no es una buena métrica para la elección de  $k$ , ya que cuantos más grupos hayan, más cerca estará cada instancia de su centroide más próximo.



# Encontrar el número de grupos óptimo

- ① El método del codo es una técnica bastante tosca para elegir k.
- ② El coeficiente de silueta de una instancia es igual a  $(b-a)/\max(a,b)$ , donde  $a$  es la distancia media a las otras instancias en el mismo grupo (es decir, la distancia intragrupo media) y  $b$  es la distancia media al grupo más cercano (es decir, la distancia media a las instancias del siguiente grupo más cercano).
- ③ El coeficiente de silueta puede variar entre -1 y +1.
- ④ Un coeficiente cercano a +1 significa que la instancia está muy metida en su propio grupo y lejos de otros grupos.
- ⑤ 0 indica que se encuentra en una frontera y -1 que la instancia se encuentra en un grupo equivocado.

```
>>> from sklearn.metrics import silhouette_score  
>>> silhouette_score(X, kmeans.labels_)  
0.655517642572828
```



# Reducción de Dimensionalidad & Aprendizaje No Supervisado

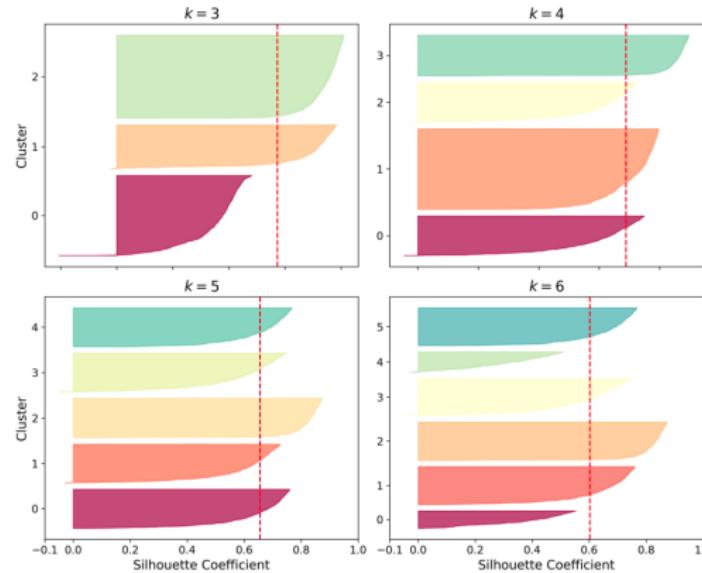
PCA | K-Means, DBSCAN

Dr. Jesus Alvarado Huayhuaz

ISIL  
BCP

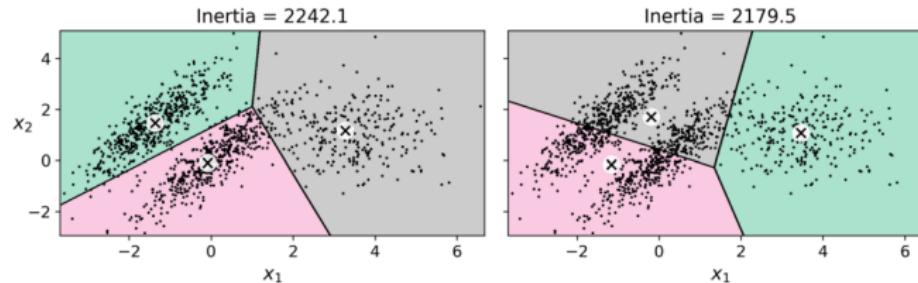
# Encontrar el número de grupos óptimo

- 1 Si graficamos el coeficiente de silueta de cada instancia por cada grupo obtenemos el siguiente diagrama de silueta.
- 2 El anchura representa los coeficientes de silueta de las instancias, cuanto más ancho, mejor. La línea discontinua es el coeficiente de silueta medio.



# Límites de K-Medias

- 1 K-Medias requiere ejecutar varias veces para evitar soluciones no óptimas y también requiere conocer previamente k.
- 2 Además, no funciona bien con grupos de tamaños variados, densidades diferentes o forma no esféricas.



- Define grupos como regiones continuas de alta densidad.
- Para cada instancia, se agrupan los puntos a  $\epsilon$  de distancia. Esta región se denomina "vecindad  $\epsilon$ "\_de la instancia.
- Si una instancia tiene al menos *min\_samples* instancias en su vecindad  $\epsilon$  (incluida ella misma) se considera una instancia central.
- Todas las instancias de la vecindad de una instancia central pertenecen al mismo grupo. Esta vecindad puede incluir otras instancias centrales, para formar un grupo.
- Cualquier instancia que no sea central y no tenga una en su vecindad se considera una anomalía.

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

X, y = make_moons(n_samples=1000, noise=0.05)
dbscan = DBSCAN(eps=0.05, min_samples=5)
dbscan.fit(X)

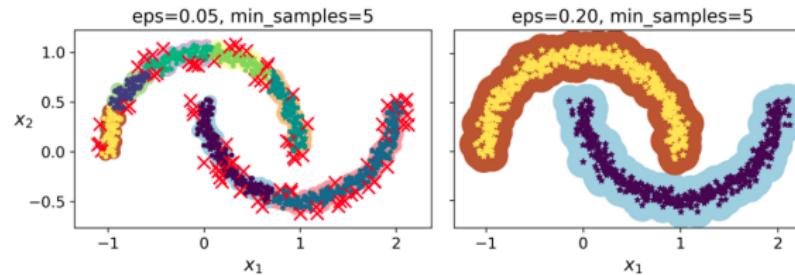
>>> dbscan.labels_
array([ 0,  2, -1,  1,  0,  0,  0, ...,  3,  2,  3,  3,  4,  2,  6,
       3])
```

- -1 indica anomalías

## DBSCAN

```
>>> len(dbSCAN.core_sample_indices_)
808
>>> dbSCAN.core_sample_indices_
array([ 0,  4,  5,  6,  7,  8, 10, 11, ..., 992, 993, 995, 997, 998,
999])
>>> dbSCAN.components_
array([[ -0.02137124,  0.40618608],
       [-0.84192557,  0.53058695],
       ...
       [ -0.94355873,  0.3278936 ],
       [  0.79419406,  0.60777171]])
```

- `core_sample_indices_` contiene los índices de las instancias centrales y `components_` da las propias instancias centrales (coordenadas).
- Se identificaron 7 grupos, sin embargo, modificando  $\epsilon$  a 0.2 obtenemos 2 grupos.

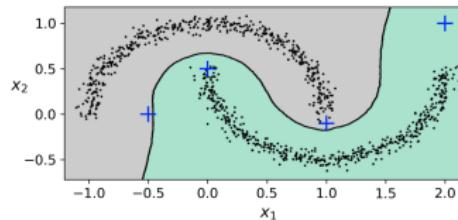


DBSCAN<sup>1</sup>

```
from sklearn.neighbors import KNeighborsClassifier

knn = KNeighborsClassifier(n_neighbors=50)
knn.fit(dbSCAN.components_, dbSCAN.labels_[dbSCAN.core_sample_indices_])

>>> X_new = np.array([[-0.5, 0], [0, 0.5], [1, -0.1], [2, 1]])
>>> knn.predict(X_new)
array([1, 0, 1, 0])
>>> knn.predict_proba(X_new)
array([[0.18, 0.82],
       [1. , 0. ],
       [0.12, 0.88],
       [1. , 0. ]])
```



- DBSCAN es sencillo (requiere solo eps y min\_samples) y sólido en la detección de valores atípicos, sin embargo, si la densidad varía de manera significativa de unos grupos a otros, puede resultarle imposible capturar todos los grupos de manera adecuada.

<sup>1</sup>[https://github.com/inefable12/upch\\_rp\\_2023/blob/main/DBSCAN\\_cl2.ipynb](https://github.com/inefable12/upch_rp_2023/blob/main/DBSCAN_cl2.ipynb)

-  D Arthur and S Vassilvitskii.  
*k-means++: The advantages of careful seeding: Conference.-proceedings of the eighteenth annual acm-siam symposium on discrete algorithms.*  
2007.
-  Jesús Bobadilla.  
*Machine learning y deep learning: usando Python, Scikit y Keras.*  
Ediciones de la U, 2021.
-  Aurélien Géron.  
*Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow.*  
O'Reilly Media, Inc., 2022.
-  Karl Pearson.  
Liii. on lines and planes of closest fit to systems of points in space.  
*The London, Edinburgh, and Dublin philosophical magazine and journal of science*, 2(11):559–572, 1901.