Dan Beninson
Instituto de Tecnología Nuclear

Universidad Nacional de San Martín

Comisión Nacional de Energía Atómica

Nociones de Aprendizaje Automático para Aplicaciones Nucleares

Aprendizaje No Supervisado

Aprendizaje no supervisado

- Aunque la mayoría de las aplicaciones actuales del aprendizaje automático se basan en el aprendizaje supervisado, la mayoría de los datos disponibles (y más en esta área) son datos no etiquetados.
- La tarea de etiquetar datos es costosa, compleja, y a menudo demanda mucho más tiempo y esfuerzo que el dedicado a probar algoritmos de aprendizaje automático y desarrollar una solución.
- A esto se suma que, en términos generales, mientras más complejo sea el problema que deseamos abordar mayor será la cantidad de datos necesarios. La demanda de datos incrementa tanto con la complejidad del problema como con la sofisticación de los métodos utilizados.
- Acá es donde entra el **aprendizaje no supervisado.**

Aprendizaje no supervisado

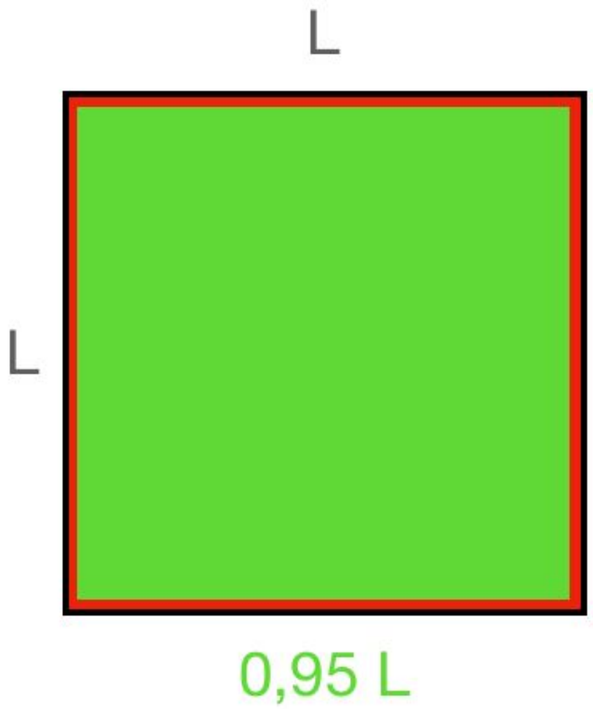
- **Reducción de la dimensionalidad:** Simplificación de datos eliminando características redundantes o poco informativas manteniendo la mayor cantidad de información posible.
- ***Clustering*** (agrupamiento): Busca agrupar instancias similares en *clusters*. Piensen en cosas como segmentación de clientes, sistemas de recomendación, agrupar las fotos de nuestras mascotas en el celular, etc.
- **Detección de anomalias**: Si se aprende la generalidad de como son los datos “normales” se pueden identificar los que no lo sean.
- **Estimación de densidad**: Se busca estimar la función de densidad de probabilidad del proceso aleatorio que generó el conjunto de datos.

Estos métodos se suelen usar como herramientas en el análisis de datos o como un paso de preprocesamiento previo a aplicar otros algoritmos de AA

Reducción de la dimensionalidad

- Muchos problemas de aprendizaje automático implican miles o incluso millones de características para cada instancia de entrenamiento.
- Todas estas características no sólo pueden hacer que el entrenamiento sea extremadamente lento, sino que también pueden hacer que sea mucho más difícil encontrar una buena solución.
- Este problema se conoce como **la maldición de la dimensionalidad**.

Reducción de la dimensionalidad

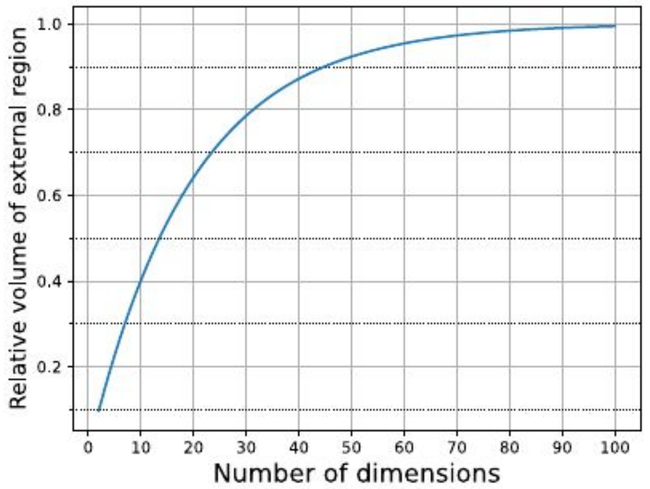


2D

$$\frac{V}{V} = \frac{V - V}{V} = 1 - \frac{V}{V} = 1 - \frac{0,95^2 L^2}{L^2} = 1 - 0,95^2 = 0,0975$$

3D

$$\frac{V}{V} = \frac{V - V}{V} = 1 - \frac{V}{V} = 1 - \frac{0,95^3 L^3}{L^3} = 1 - 0,95^3 = 0,143$$



Rodrigo F. Díaz, Aprendizaje Automático (2021)

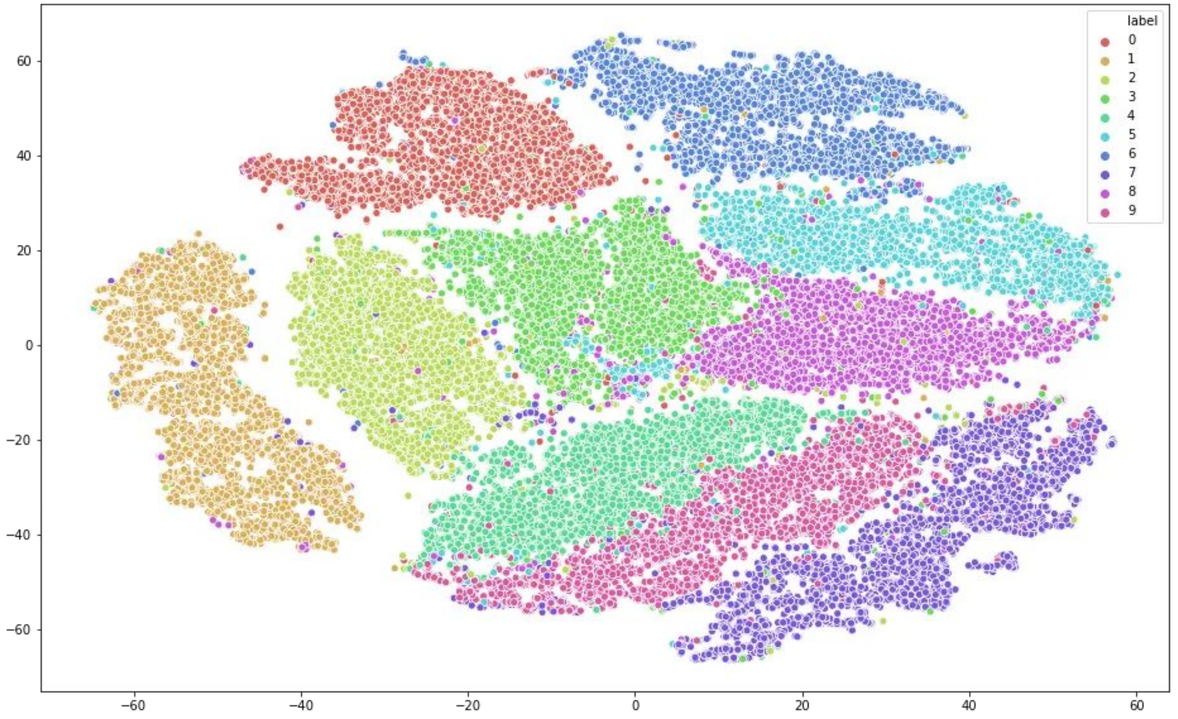
Reducción de la dimensionalidad

- La distancia media entre puntos aumenta, los datos tienden a ser muy esparzos.
- Una nueva instancia va a estar muy lejos de cualquier instancia de entrenamiento, las predicciones van a ser menos confiables porque van a estar basadas en extrapolaciones muy grandes.
- Mayor riesgo de *overfitting*.
- Entrenamientos más costosos, necesidad de una cantidad enorme (*) de datos con sus problemas de obtención y manipulación (o directamente, imposibilidad práctica).

(*)Con 100 características, todas entre 0 y 1, se necesitarían más instancias de entrenamiento que átomos en el universo observable para que cada punto de entrenamiento esté dentro de 0,1 de media, suponiendo que estuvieran repartidas uniformemente en todas las dimensiones. (Gerón, Capítulo 8)

Reducción de la dimensionalidad

- Las ventajas de reducir las dimensiones de los datos son claras:
  - Alejarnos de la *maldición de la dimensionalidad*.
  - Mejora el rendimiento (tiempo de cómputo, memoria).
  - Mejora la calidad de los datos eliminando el ruido de dimensiones irrelevantes o redundantes.
  - Herramienta contra el *overfitting*.
  - Facilita la visualización, podemos representar datos complejos en dos o tres dimensiones para entender patrones.



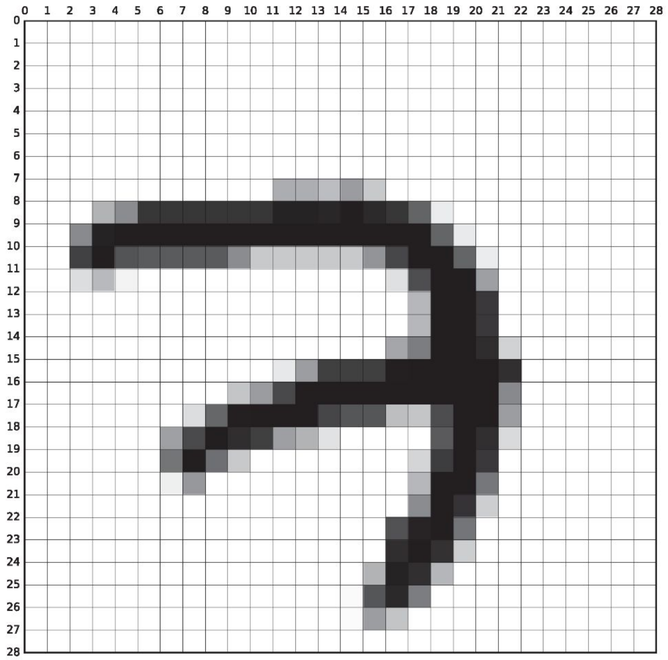
Scatter plot 2D de los datos MNIST tras aplicar PCA (50 componentes) y luego t-SNE (Link)

Reducción de la dimensionalidad

- En la mayoría de los problemas del mundo real, las instancias de entrenamiento no están repartidas uniformemente por todas las dimensiones.
- Muchas características son casi constantes, mientras que otras están muy correlacionadas.
- Se puede pensar, entonces, que todas las instancias de entrenamiento se encuentran dentro (o cerca) de un subespacio de dimensiones mucho más bajas que las del espacio original.
- Pensemos el caso de los datos de MNIST.

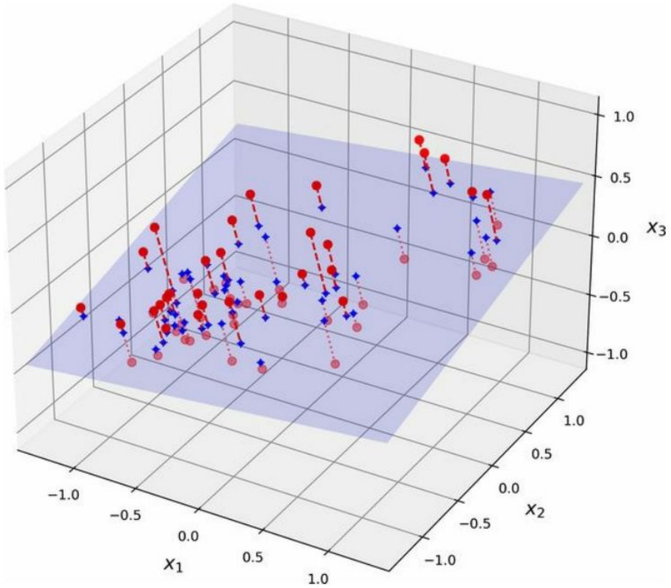
Reducción de la dimensionalidad

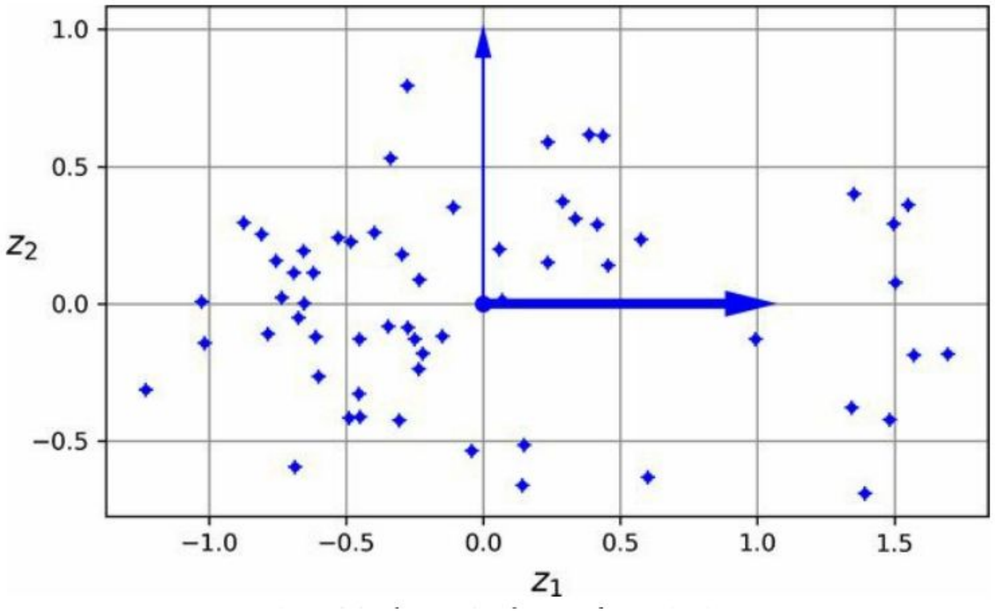
- Cada imagen de MNIST “vive” en un espacio de 784 dimensiones.
- ¿Le parece que todas estas dimensiones son igual de importantes al transmitir información?



Reducción de la dimensionalidad

- Pensemos en proyecciones, podemos simplemente proyectar los datos en hiperplanos de dimensiones inferiores.

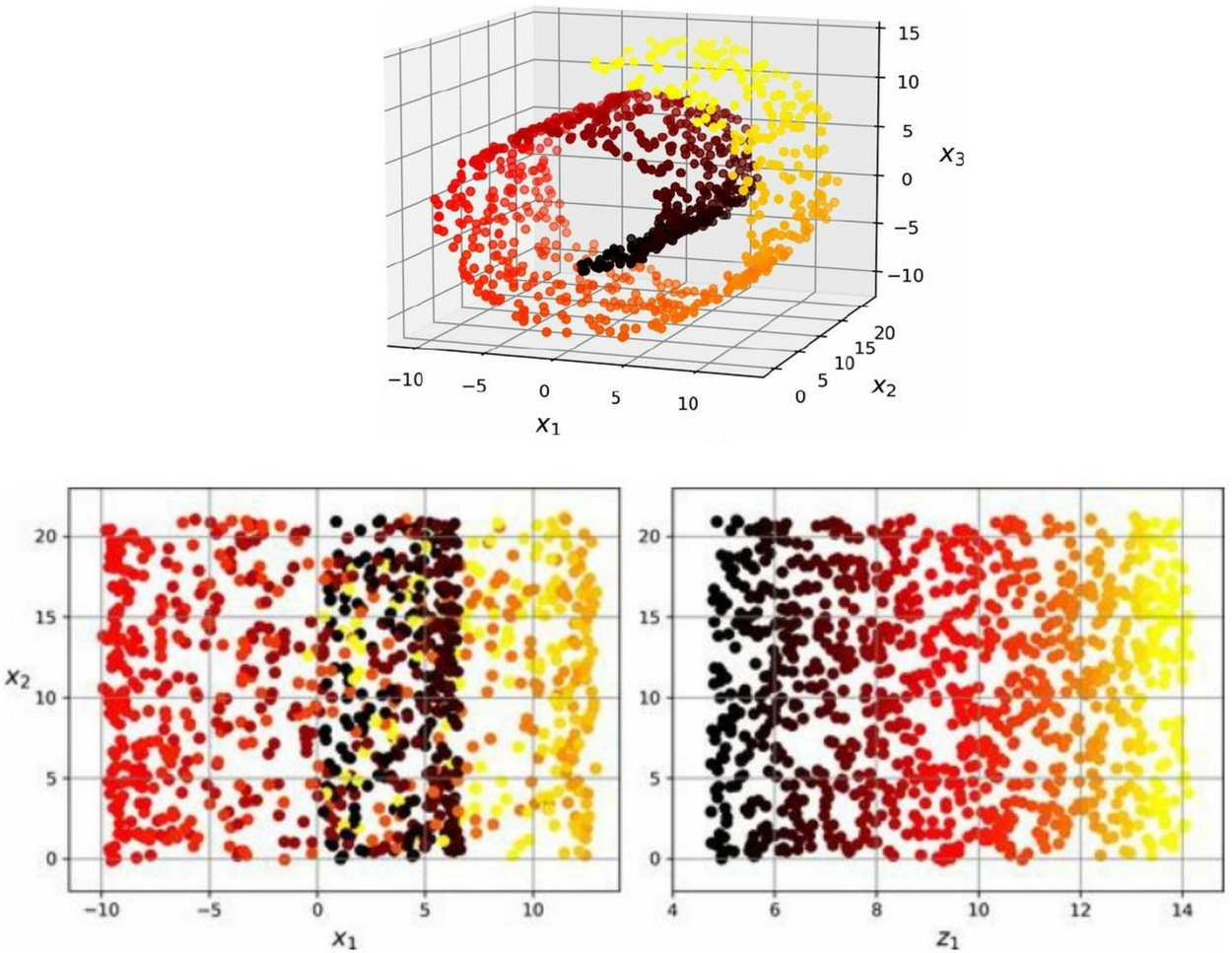




Imágen del Gerón

Reducción de la dimensionalidad

- Otra opción más avanzada puede ser lo que se conoce como *Manifold Learning*.
- Es una técnica que busca reducir la dimensionalidad de los datos preservando sus propiedades geométricas y estructurales más importantes.
- Se basa en la idea de que, aunque los datos puedan estar en un espacio de alta dimensión, a menudo se distribuyen en torno a un *manifold* (variedad) de dimensión mucho menor.
- En este ejemplo del *Swiss roll* se pueden ver las diferencias entre proyectar y desenrollar.



Aplastando con proyecciones (izq.) Vs. *Unrolling*

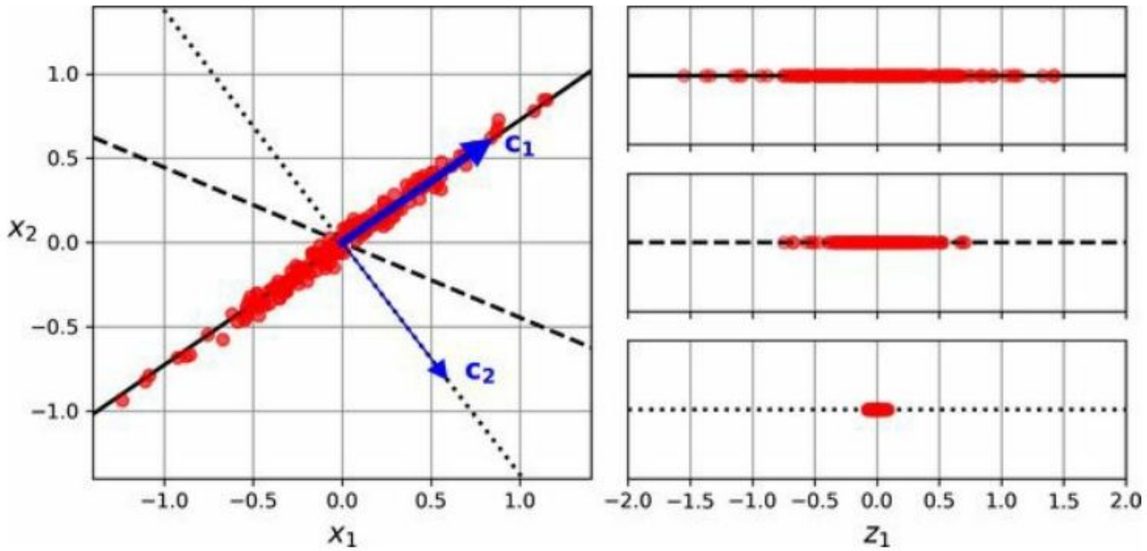
Imágen del Gerón

Reducción de la dimensionalidad

- De nuevo, pensemos en MNIST: todas las imágenes de dígitos manuscritos tienen algunas similitudes. Están formadas por líneas conectadas, los bordes son blancos y están más o menos centradas.
- Si se generan imágenes al azar, sólo una fracción ridículamente pequeña se parecería a los dígitos manuscritos.
- En otras palabras, los grados de libertad que tenemos si queremos crear una imagen de dígitos son drásticamente inferiores a los grados de libertad que hay si se le permite generar cualquier imagen que desee. Estas restricciones tienden a comprimir el conjunto de datos en una variedad de dimensiones inferiores.

Preservando la varianza al proyectar

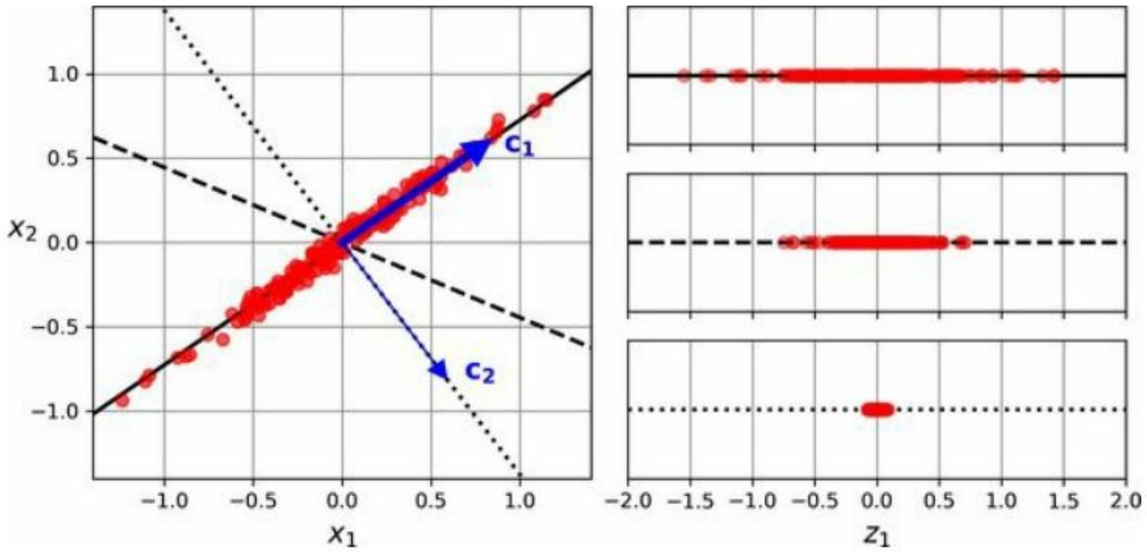
- Antes de poder proyectar el conjunto de entrenamiento en un hiperplano de dimensiones inferiores, primero hay que elegir el adecuado.
- Las diferentes proyecciones conservan diferentes porcentajes de la varianza (en el ejemplo: máxima, media y baja)
- Parece razonable seleccionar el eje que conserva la máxima cantidad de varianza, ya que lo más probable es que pierda menos información que las demás proyecciones.
- Otra forma de justificar esta elección es que **se trata del eje que minimiza la distancia cuadrática media entre el conjunto de datos original y su proyección sobre ese eje**. Esta es la sencilla idea que subyace a PCA.



Imágen del Gerón

Análisis de Componentes Principales (PCA)

- PCA identifica el eje que representa la mayor cantidad de varianza en el conjunto de entrenamiento (la línea continua).
- También encuentra un segundo eje, **ortogonal al primero**, que representa la mayor parte de la varianza restante (en este ejemplo 2D no hay más opciones que la línea punteada).
- Si hay más dimensiones, PCA también encontrará un tercer eje, ortogonal a los dos ejes anteriores, y un cuarto, un quinto, y así sucesivamente. Tantos ejes como dimensiones tengan los datos.

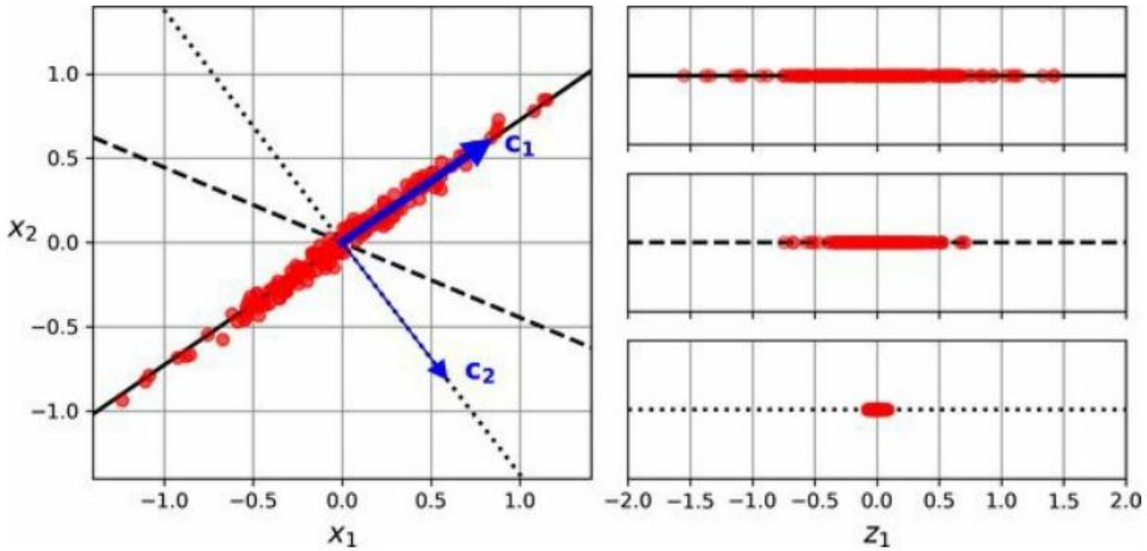


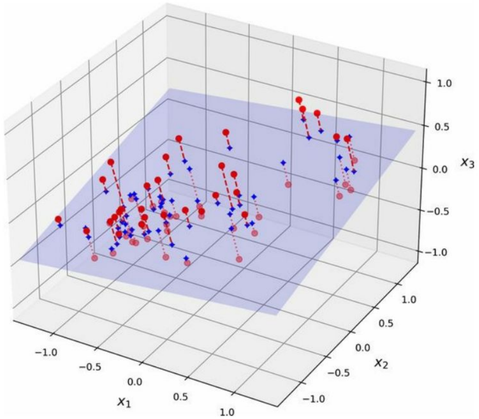
Imágen del Gerón

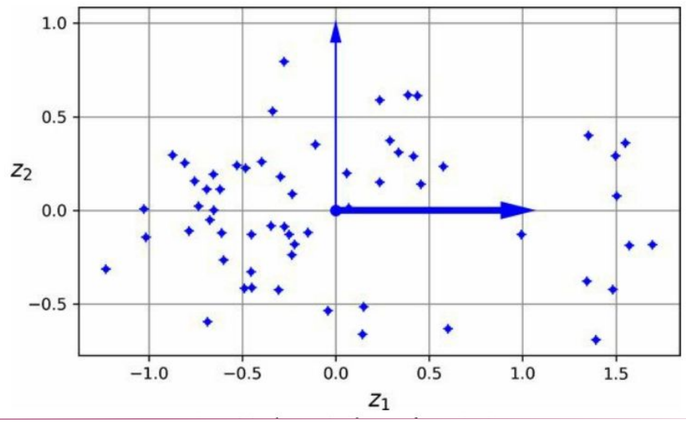
LINK: StatQuest: PCA - Practical Tips

Análisis de Componentes Principales (PCA)

- El *i*-ésimo eje se llama el *i*-ésimo componente principal (CP) de los datos.
- En el ejemplo de la derecha el primer CP es el eje sobre el que se encuentra c1, el segundo es c2.
- En la figura de abajo, los dos primeros están en el plano y el tercero al eje ortogonal.





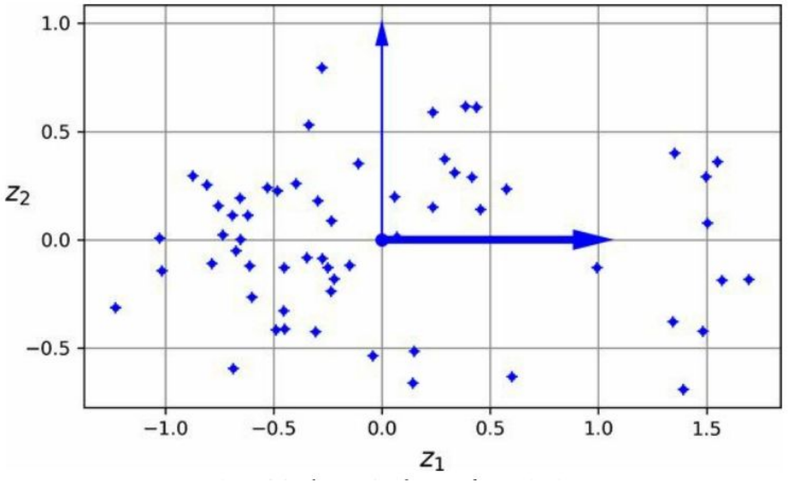


- Luego de la proyección el primer CP corresponde al eje Z_1 y el segundo a Z_2.

Imágen del Gerón

Explained Variance Ratio

- Una información útil de este proceso es el *explained variance ratio*.
- Este ratio indica la proporción de la varianza del conjunto de datos que se encuentra en cada componente principal.
- En scikit-learn está accesible a traves de la variable `explained_variance_ratio_`.
- En el ejemplo, alrededor del 76% de la varianza de los datos están en la primera CP y un 15% en la segunda, dejando un 9% para la tercera, por lo que es razonable pensar que contiene muy poca información.

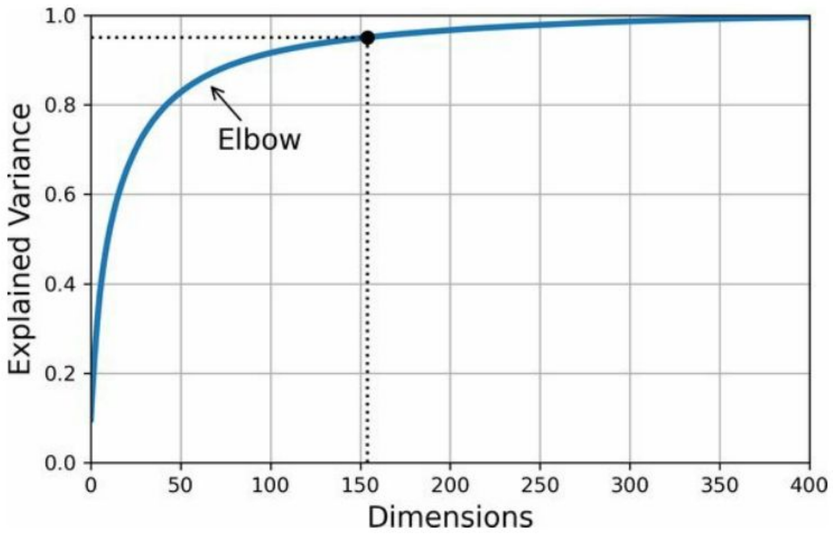



```
>>> pca.explained_variance_ratio_
array([0.7578477 , 0.15186921])
```

Imágen del Gerón

Elegir la cantidad de dimensiones

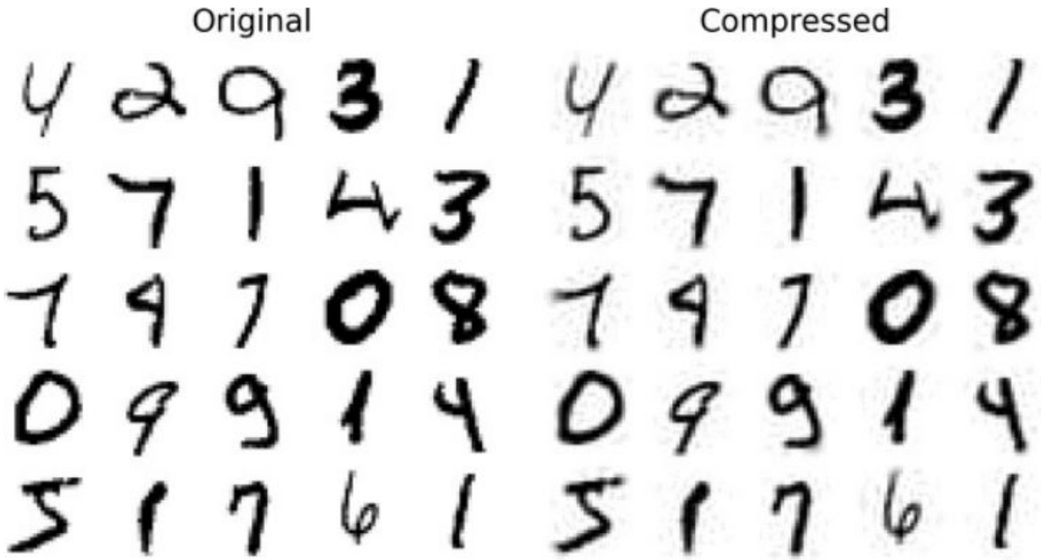
- Si queremos visualizar datos es sencillo, vamos a elegir dos o tres CP.
- Si, en cambio, estamos buscando reducir para poder seguir trabajando con los datos podemos buscar que número de componentes explican una porción suficientemente grande de la varianza (por ejemplo un 95%).
- Otra opción es plotear la varianza explicada en función del número de dimensiones, buscando *el "codo"* en la curva donde la varianza deja de "crecer rápido".
- Finalmente, también se puede pensar como un hiperparámetro más y probar diferentes combinaciones junto al algoritmo con el que estamos haciendo la clasificación o regresión posterior.



Imágen del Gerón

PCA para compresión

- Luego de reducir sus dimensiones, el conjunto de entrenamiento ocupa mucho menos.
- Por ejemplo, tras aplicar PCA al conjunto de datos MNIST y conservar el 95% de su varianza, nos quedamos con 154 características, en lugar de las 784 originales.
- Ahora tenemos un 20% del tamaño original habiendo perdido solamente un 5% de su varianza.
- Es posible hacer la transformación inversa al espacio original, obviamente no nos va a dar los datos originales, pero se va a parecer bastante.
- La distancia media al cuadrado entre los datos originales y los datos reconstruidos se denomina **error de reconstrucción**.



Imágen del Gerón

Clustering

- El *Clustering* o “agrupamiento” son un conjunto de técnicas y métodos no supervisados (porque estamos en las diapositivas de “Aprendizaje no supervisado” :P) para clasificar o particionar nuestros datos en grupos o *clusters*.
- Al particionar la idea es que las observaciones dentro de cada grupo sean lo más *similares* posibles entre sí, mientras que son distintas a las observaciones de los otros grupos.
- Se puede usar para la detección de anomalías, para clasificar, para hacer un análisis exploratorio de los datos, etc.
- Se suele usar en conjunción con las técnicas de reducción de la dimensionalidad para optimizar tiempos, resultados o hacer un análisis similar al ejemplo de la diapositiva 7.
- Pensemos cómo es que se puede hacer esto de manera no supervisada.

Clustering

- Puede que no sepamos **qué** tipo de árbol es cada uno, pero podemos saber **cuáles** son del mismo tipo (más o menos).
- Depende de las **características** que elijamos ver.
- ¿Color?, ¿Forma de la copa?, ¿Diámetro?, ¿Altura?, ¿Una combinación de ambos?.
- Si elegimos pocos, no podemos agrupar bien; si elegimos muchos, perdemos generalidad.
- Es un balance..

Clustering

- Cada árbol puede representarse con un vector de características
- (altura, diámetro, tono de verde, ...).

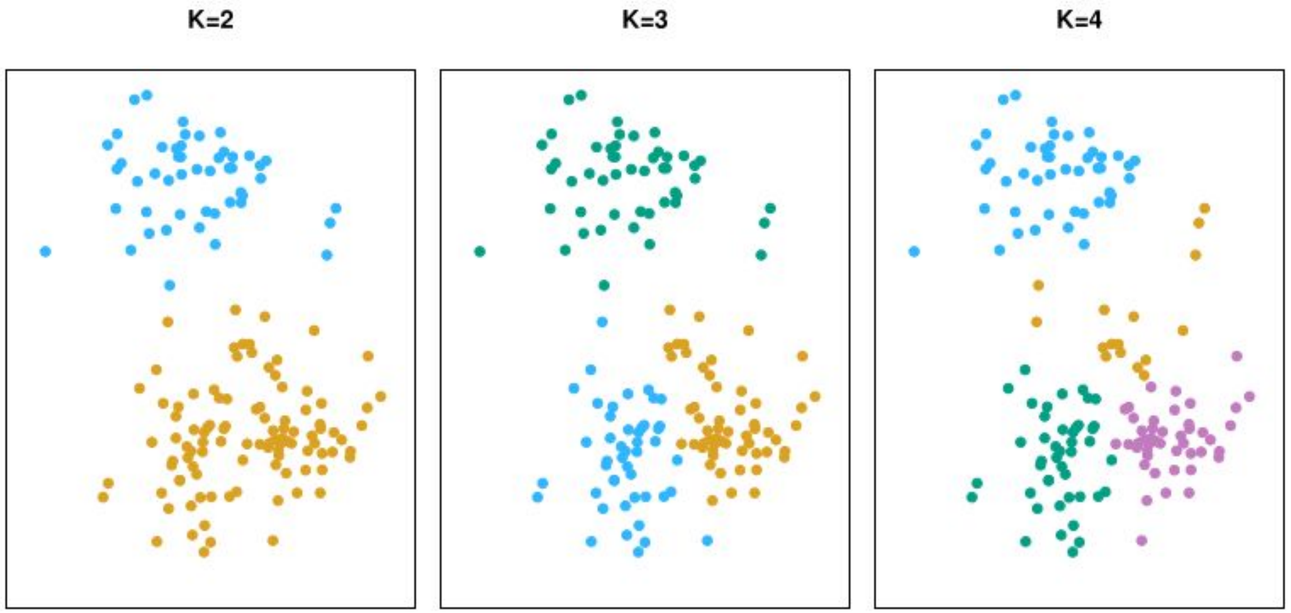
(25, 120, 3A5311, ...)

(10, 30, 597D35, ...)

- Luego podemos ver que los árboles más similares son los que están más cerca en este espacio de características.

K-Means

- K-Means es uno de los métodos más populares, si bien no es de los más poderosos, pero su rapidez y simplicidad lo convierte en una excelente opción como punto de partida para explorar problemas de *clustering*.
- Simplemente se le dan los datos y se le indica un número *k* que representa la cantidad de *clusters* a buscar y el algoritmo le asigna un grupo a cada punto de nuestro dataset.

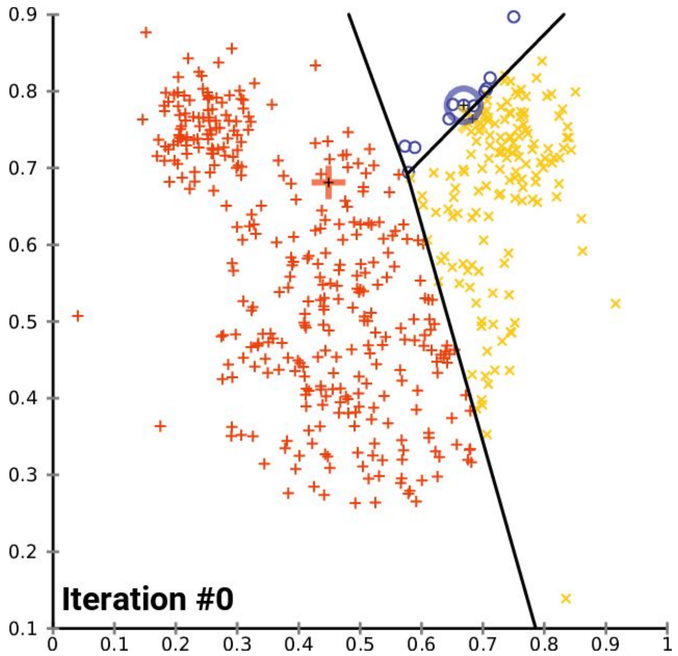


Para pensar:
¿Qué es un cluster?

Imágen de An Introduction to Statistical Learning

K-Means

- Encuentra los *k* centroides que minimizan la distancia entre cada punto del conjunto de datos y su centroide correspondiente.



Source: Wikipedia

1. Selecciona *k* puntos del conjunto de datos como centroides iniciales.
2. Asigna cada punto del conjunto de datos al centroide más cercano.
3. Recalcula los centroides de cada grupo utilizando la distancia media de todos los puntos que están asociados a ese centroide.
4. Repite los pasos 2 y 3 hasta que no haya cambios o se alcance un número máximo de iteraciones.

LINK: StatQuest: K-means clustering

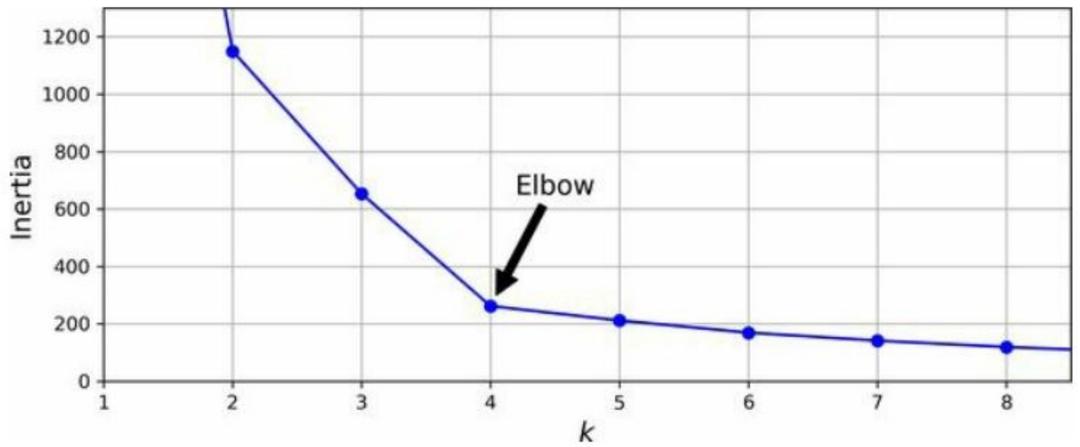
K-Means

- Si se conocen aproximadamente donde los centroides deberían estar (por ejemplo, se usó algún otro algoritmo de clustering antes) se puede inicializar el algoritmo con esos puntos como centroides iniciales.
- Dependiendo de los datos, los clusters encontrados pueden depender de las inicializaciones al azar de los centroides, se puede ejecutar varias veces con varias inicializaciones aleatorias (hiperparámetro `n_init`) y devolver el “mejor agrupamiento”.
- Esto nos lleva a pensar cómo elegimos un *k* adecuado y que significa un “buen agrupamiento”...

K-Means

- Un criterio para definir el mejor agrupamiento usa como métrica la **inercia**: que es la distancia media de cada dato a su centroide más cercano. Cuanto menor es la inercia, más concentrados están los puntos alrededor de los clusters.

$$I = \sum_{j=1}^{K} \sum_{i=1}^{n_j} (x_i - \mu_j)^2$$



Imágen del Gerón

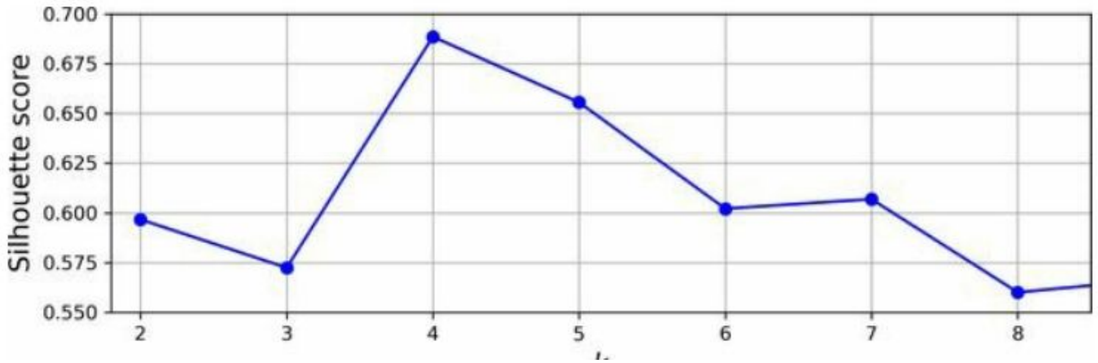
- Se prueba con diferentes *k* buscando el punto en el que la inercia *deja de disminuir tan rápido* (formando el famoso "*codo*").
- Es una forma rápida de calcular pero un poco tosca (además de que siempre va a disminuir cuando aumenta *k*), otra forma más costosa computacionalmente, pero más precisa, es usar la métrica de **la Silueta**.

K-Means

- Silueta: para cada punto *i*, ***a(i)*** *es la distancia media a otros datos del mismo cluster (es la distancia media intra-cluster), y* ***b(i)*** *es la distancia media al cluster más cercano, es decir, la distancia media a datos del siguiente cluster más cercano:*

$$S(i) = \frac{(b(i) - a(i))}{\max(a(i), b(i))}$$

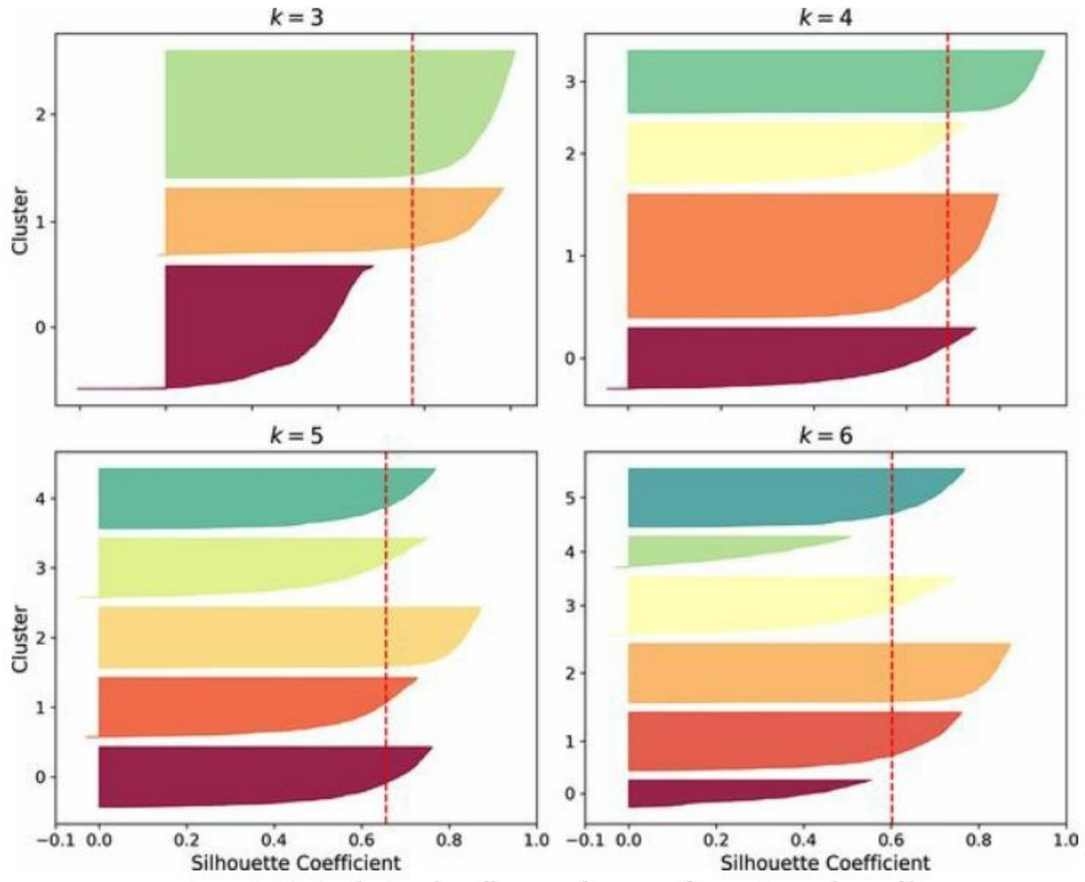
- Se calcula para todos los puntos y se hace el promedio.
- Puede variar entre -1 y +1: un coeficiente cercano a +1 significa que el dato está bien asignado dentro de su propio cluster y lejos de otros clusters, mientras que un coeficiente cercano a 0 significa que un dato está cerca del límite de un cluster.
- Finalmente coeficientes negativos indican que ese dato puede haber sido asignado al cluster equivocado.



Imágen del Gerón

K-Means

- Otra visualización es representar el coeficiente de silueta de cada instancia, ordenado por los clusters a los que está asignado y por el valor del coeficiente. Esto se denomina **diagrama de siluetas.**
- Para cada forma la altura indica el número de instancias en el cluster, y su anchura representa los coeficientes de silueta ordenados. Las líneas verticales indican la puntuación media de la silueta para cada número de clusters.
- Si muchas instancias de un cluster tienen un coeficiente inferior a esta puntuación, el cluster se considera malo, ya que indica que están demasiado cerca de otros clusters.



(En este caso k=3 o k=6 son peores que k=4 o k=5)

Imágen del Gerón

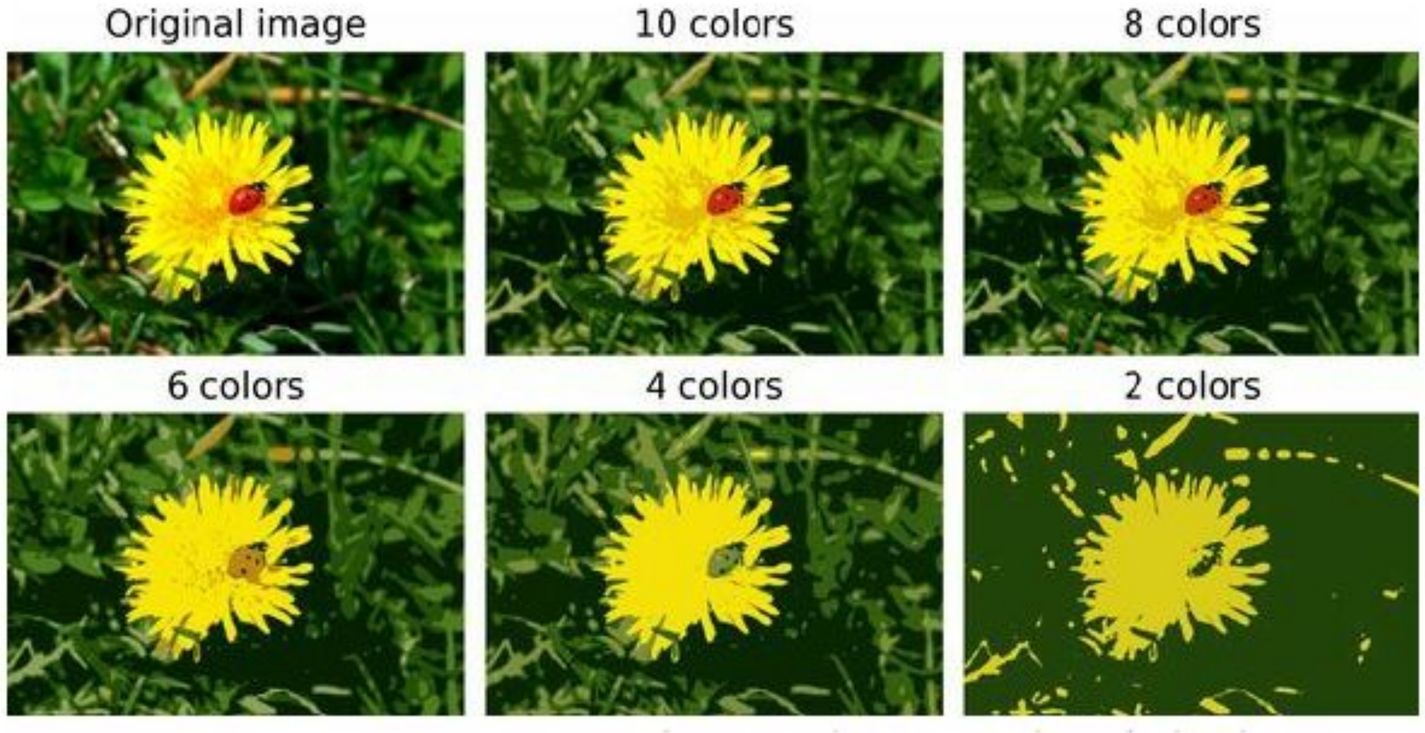
K-Means

- Si bien K-Means es un método rápido y escalable tiene algunas limitaciones:
  - Es necesario ejecutar el algoritmo varias veces para evitar soluciones subóptimas.
  - Requiere especificar el número de clusters, lo cual puede ser complicado.
  - Tiene problemas con clusters de tamaños variables, densidades diferentes o formas no esféricas (para atacar algunos de estos problemas se puede probar con modelos de mixturas gaussianas).
- A pesar de estas limitaciones tiene algunos aplicaciones que vale la pena mencionar.

K-Means

- Segmentación de imágenes: es la tarea de particionar una imagen en varios segmentos, tiene algunas variantes:
  - **Segmentación por colores**, los píxeles con un color similar se asignan al mismo segmento.
  - **Segmentación semántica**, todos los píxeles que forman parte del mismo tipo de objeto se asignan al mismo segmento. Por ejemplo, en el sistema de visión de un coche autónomo, todos los píxeles que forman parte de la imagen de un peatón podrían asignarse al segmento “peatón”
- El estado del arte en segmentación se alcanza con arquitecturas complejas de redes convolucionales, pero para problemas simples podría ser una buena solución.

K-Means

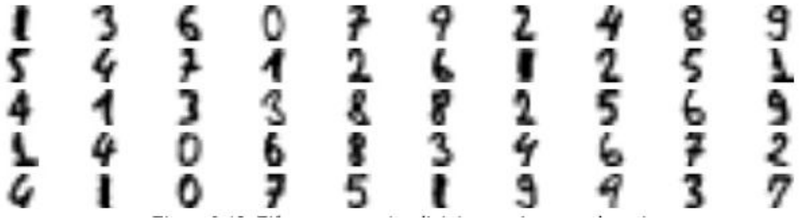


Ejemplo de segmentación usando k-means con varios números de clusters de color

Imágen del Gerón

K-Means

- Aprendizaje semi-supervisado
  - Supongamos que tenemos el conjunto MNIST, pero no tenemos recursos para etiquetar todas las imágenes.
  - Podemos hacer un clustering de, por ejemplo, 50 clusters.
  - Para cada uno de estos clusters buscamos las imágenes que están más cerca de su respectivo centroide, llamaremos a estas imágenes “imágenes representativas”, las etiquetamos a mano y entrenamos con esto. Los resultados son mejores con imágenes representativas que con instancias al azar.
  - ¡Se puede ir un paso más allá!, se puede etiquetar automáticamente las instancias más cercanas a los centroides y también ignorar las más lejanas. (Ver cap 8 del Gerón para detalles).



```
y_representative_digits = np.array([1, 3, 6, 0, 7, 9, 2, ..., 5, 1, 9, 9, 3, 7])
```

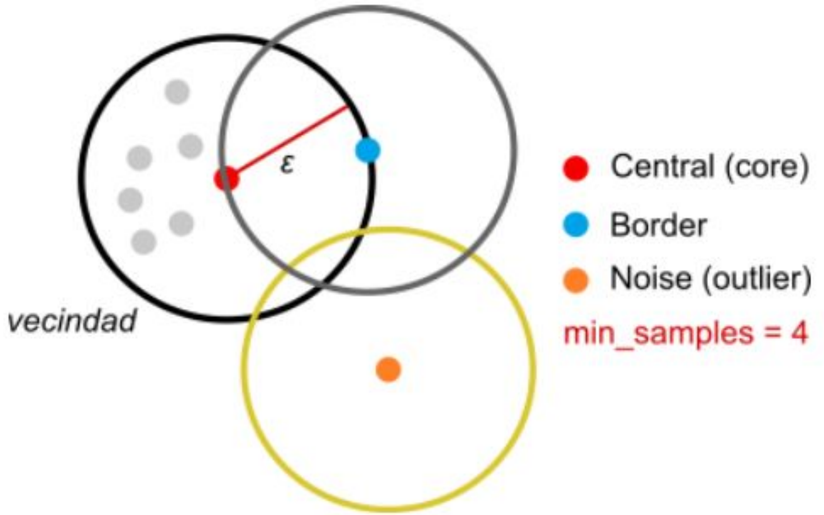
Detección de anomalías

- Como en K-Means los puntos se asignan a clusters basados en su proximidad al centroide, los puntos que estén más lejos podrían considerarse anómalos. Se los puede analizar uno por uno o tratar de definir un umbral de distancia.
- El coeficiente de silueta puede ayudar a identificar puntos que tienen un bajo valor de cohesión con su cluster o que están más cerca de otros clusters. Los puntos con un valor de silueta bajo o negativo pueden ser indicativos de anomalías, ya que están en el borde o fuera de su cluster asignado.
- Otra estrategia es considerar la distancia entre el punto y el cluster más cercano, los puntos ubicados lejos de todos los clusters podrían representar anomalías, ya que no se agrupan naturalmente con ningún conjunto de puntos en los datos.
- Y para terminar, mencionemos otro algoritmo de *clustering* un poco más sofisticado y que además nos detalla los *outliers*.

DBSCAN

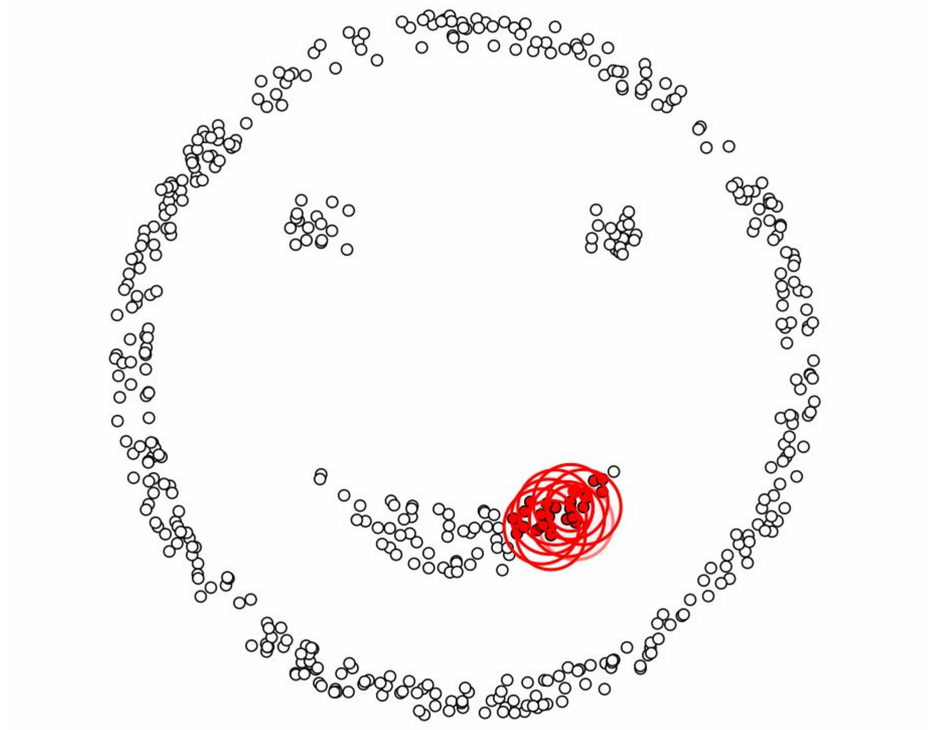
- DBSCAN (Density-Based Spatial Clustering of Applications with Noise): Es un algoritmo de clustering que se basa en la **densidad de los datos** para formar grupos, no necesita conocer el número de clusters a priori y puede encontrar clusters de formas arbitrarias.

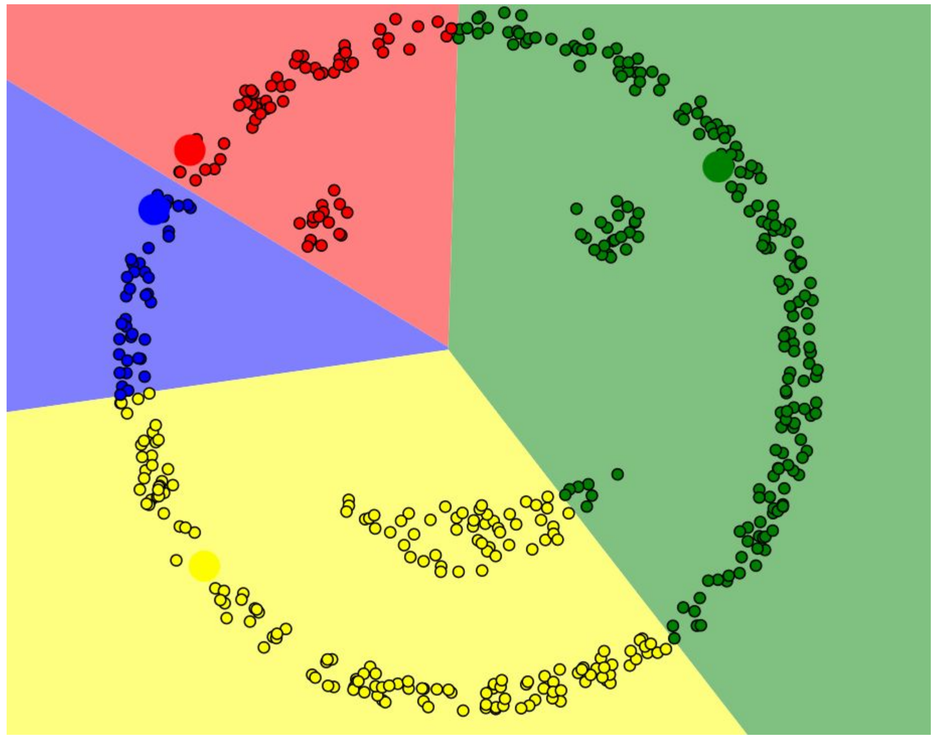
- Para cada instancia, el algoritmo cuenta cuántas instancias se encuentran a una distancia ε de ella. Esta región se denomina vecindad ε de la instancia.
- Si una instancia tiene al menos *min_samples* de instancias en su vecindad ε (incluida ella misma), se considera una instancia núcleo.
- Todas las instancias de la vecindad de una instancia núcleo pertenecen al mismo cluster. Esta vecindad puede incluir otras instancias núcleo; por lo tanto, una larga secuencia de instancias núcleo vecinas **forma un único clúster**.
- Cualquier instancia que no sea un núcleo y no tenga ninguno en su vecindario se considera una anomalía.



LINK: Clustering with DBSCAN, Clearly Explained!!!

Comparación con K-Means





Fuente: Understanding DBSCAN: A Practical Guide for Beginners with Business Applications

Parada técnica: Resumen hasta ahora

- **Aprendizaje no supervisado:** Búsqueda de patrones en datos para los que no tenemos etiquetas.
- **Reducción de la dimensionalidad:** Tener datos en muchas dimensiones puede tener sus complicaciones, tenemos algunas técnicas para buscar reducir esta dimensionalidad para usarlas como entradas para otros algoritmos o para visualizar.
- **Clustering:** Algoritmos que buscan agrupar datos similares, útiles para detección de anomalías, etiquetado, análisis de datos o incluso segmentación de imágenes.