Conceptos Fundamentales

Índice

Referencia principal

Cap. 1 de "Hands-On Machine Learning with Scikit-Learn and TensorFlow"
`(Todas las imágenes pertenecen a dicha referencia salvo que se indique lo contrario)`

Al terminar este tema conoceremos los conceptos más importantes del lenguaje empleado habitualmente en Reconocimiento de Patrones.

1. ¿Qué es el *Reconocimiento de patrones*?

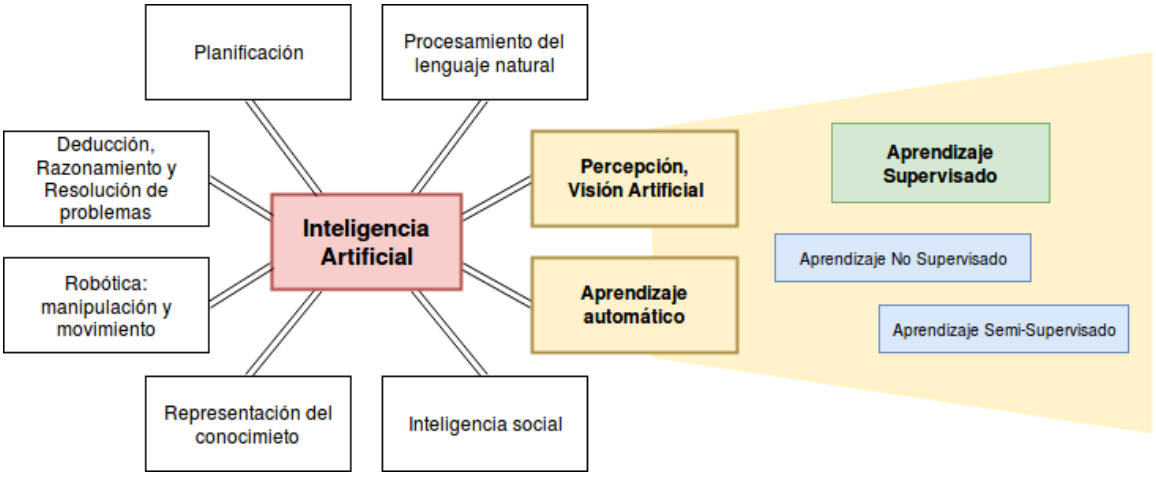

Figura 1: Diferentes ramas de la IA [Fuente: Original de A. Cuesta]

El **Reconocimiento de Patrones** es el conjunto de técnicas que tienen el objetivo de encontrar, de manera automática, elementos comunes y característicos de la clase bajo la cual se agrupa un conjunto de datos. Mediante la utilización de estos patrones se pretende automatizar la estimación de la clase a la que pertenecen otros nuevos ejemplos.

El término *Reconocimiento de Patrones* (RP de aquí en adelante) fue la primera terminología empleada para una de las grandes tareas de lo que hoy en día se conoce como **Aprendizaje Automático** (*Machine Learning*, o ML de aquí en adelante): el aprendizaje supervisado.

En ML el objetivo es ser capaz de que la máquina aprenda a reaccionar a partir de un conjunto de ejemplos, de manera que responda acertadamente antes ejemplos nuevos.

Cuando cada ejemplo pertenece (al menos) a una clase, y dicha clase es asignada por un humano se denomina **Aprendizaje supervisado**. El objetivo es lograr que la máquina aprenda a clasificar correctamente los ejemplos.

Cuando el humano no interviene, y por tanto los ejemplos no están etiquetados, el objetivo es lograr que la máquina los agrupe en un cierto número de conjuntos o *clusters*. Supuestamente, los miembros de un cluster tendrán mayor similitud entre ellos que con miembros de otro cluster. En este caso se habla de Aprendizaje no supervisado .

Entre ambos podemos encontrar el **Aprendizaje Semi-supervisado**, donde el objetivo es el mismo que en el aprendizaje supervisado, pero sólo unos pocos ejemplos están etiquetados. .

Hay muchas otras tareas de ML, como aprendizaje por refuerzo, filtrado colaborativo, detección de anomalías o regresión.

El ML es a su vez una de las facetas en el campo, aún más amplio, de la Inteligencia Artificial (IA). La Figura 1, muestra algunas de las más importantes, entre las que destaca la visión artificial. En este curso vamos a profundizar en las técnicas de aprendizaje supervisado, poniendo el interés en ejemplos relacionados con la visión artificial. En otras palabras, aprenderemos:

- Un conjunto de métodos para encontrar patrones en conjuntos de imágenes etiquetadas con el objetivo de construir clasificadores automáticos.
- A automatizar, sistematizar y evaluar los sistemas para obtener resultados fiables.

.

Tipos de aprendizaje

En la Figura 2 se muestran diferentes soluciones o *frameworks* de aprendizaje atendiendo a tres preguntas esenciales.

Los frameworks no son excluyentes, es decir que podemos enfrentarnos a un problema supervisado con aprendizaje en línea y finalmente para hacer predicciones se necesiten todos los ejemplos proporcionados.

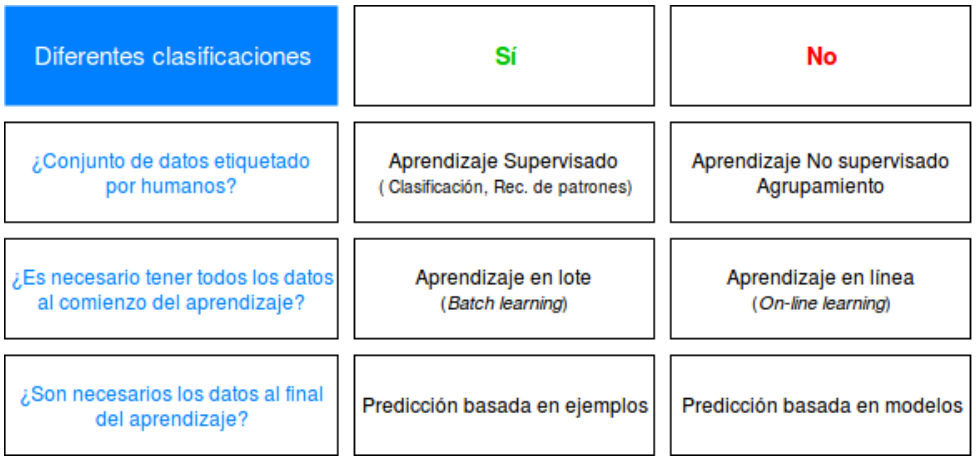

Figura 2: Las cajas centrales son diferentes soluciones (*frameworks*) según la pregunta (columna izquierda) y su respuesta (fila de arriba) [Fuente: Original de A. Cuesta]

2. Elementos esenciales para RP

2.1. Conjunto de datos

El **conjunto de datos** (*Data set*) se puede ver como una tabla en la que las filas son los diferentes ejemplos que se presentan a la máquina, y las columnas son los atributos . que los caracterizan. En la Figura 4 se muestran algunos de los conjuntos de datos más populares para visión artificial.

Figura 3: MNIST: 60K dígitos manuscritos. CIFAR-100: 60K imágenes de 100 clases distintas. ImageNet: 1500K imágenes de 1000 clases distintas. [Fuente: Anónimo @ internet]

Por ejemplo, el conjunto de datos MNIST recoge imágenes en escala de grises de tamaño 28×28 pixeles. Si escribimos todos los pixels en fila, uno tras otro, empezando por la esquina superior izquierda y terminando en la esquina inferior derecha, obtenemos un array o vector de 784 bytes (es decir, enteros entre 0 y 255). Este array caracteriza perfectamente al ejemplo (no es más que una reordenación de sus valores)-

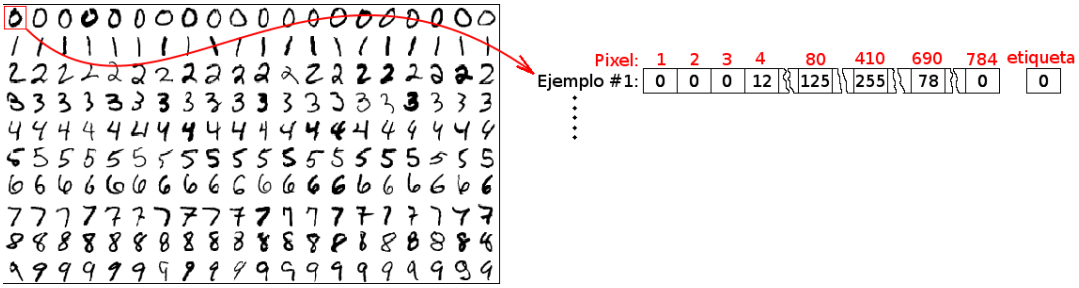

Figura 4: Muestra del conjunto de datos MNIST (izq.) y su versión como tabla de ejemplos y atributos. Además, a la derecha, se muestra la etiqueta de este ejemplo, en este caso "0". [Fuente: Original de A. Cuesta]

Los atributos o **características** de los ejemplos pueden ser medidas que se toman sobre los mismos, como por ejemplo en aplicaciones de biometría. En la Figura 5 se muestran varias de estas medidas. Así cada ejemplo estaría determinado por un vector de $37 \times 2 = 74$ elementos que se corresponde con las coordenadas x e y de cada punto azul. Por cada vector además podríamos tener una etiqueta que indicara si se trata de un varón o una mujer, o el ID de la persona de la imagen.

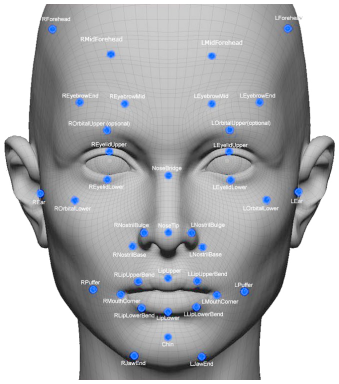

Figura 5: Características para RP de rostros. [Fuente: eForensics Magazine]

Aplicando la misma idea de hacer mediciones sobre la imagen en vez de usarla directamente, podemos plantear un ejemplo de juguete en el que se desea aprender a distinguir entre el 0 y el 1. Para ello se realizan tres mediciones: h_a es distancia entre el primer y el último pixel mayor de un cierto umbral, en vertical, h_b es similar a h_a pero en horizontal, y d es la distancia, medida desde la izquierda, del pixel superior a un cierto úmbral que ocupe la posición más alta (ver la Figura 6). A continuación reducimos estas 3 mediciones a 2 características $x_1 = h_a/h_b$ es la relación de aspecto, y $x_2 = d$ es la posición horizontal del pixel más alto por encima de un umbral dado. Por tanto el **vector de características** que representa a cada ejemplo consta sólo de dos valores, (x_1, x_2).

Esto permite poder representar cada ejemplo como un punto en un plano, tal y como se muestra en la Figura 6, donde los cuadrados serían ejemplos de '1' y los círculos de '0'.

En otras palabras, el vector de características vive en un espacio bidimensional, es decir con una dimensión por cada coordenada. Pero, como hemos visto, es frecuente tener muchas características. Por tanto, generalizando, hablaremos de vectores de características **d-dimensionales**.

En este último ejemplo, además, hemos creado la característica x_1 a partir de otras dos, h_a y h_b. Las características, por tanto se pueden elegir en función del problema y de la experiencia, es decir se realiza una **selección de atributos**. Pero también se pueden obtener mediante **extracción de**

características(*feature extraction*) relacionando unos atributos con otros mediante expresiones matemáticas. Al proceso de obtener características frecuentemente se le conoce como **ingeniería de características**(*feature engineering*).

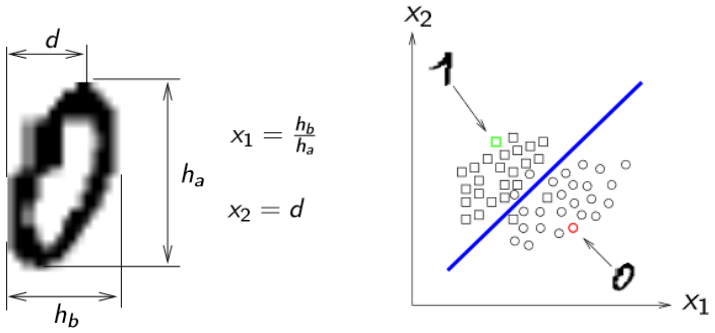

Figura 6: Dos características para distinguir un '0' de un '1' pueden ser suficientes. Además permite visualizar el conjunto de datos en un gráfico. [Fuente: J.M. Buenaposada]

En definitiva:

- El conjunto de datos se puede representar en una tabla con n filas y d columnas.
- Cada fila es un ejemplo del conjunto de datos.
- Cada columna es una característica.
- Cada fila es por tanto un vector de características d-dimensional.
- Las características seleccionadas se pueden combinar para crear otras nuevas.
- En tareas de clasificación cada fila tendrá (al menos) una etiqueta
- El objetivo será encontrar la función discriminante.

2.2. Clasificadores

En la Figura 6 se puede apreciar también una línea azul que separa de manera *general* (observar que se produce algún fallo) los ejemplos del conjunto de datos en dos clases. Dicha línea, o en general hipersuperficie para cualquier dimensionalidad, se conoce como **Superficie de decisión**.

Se define la **Función discriminente** $f(x)$ como aquella que recibe un vector de características, x y devuelve una estimación de la clase a la que pertenece, $\hat{y} = f(x)$.

Cuando la función discriminante es **lineal** la superficie de decisión será una recta, un plano o un hiperplano, según el número de dimensiones. Un discriminante lineal es el más sencillo que se puede construir, y si funciona bien entonces las clases son **linealmente separables**. En otro caso tendremos que construir clasificadores no-lineales.

Algunos métodos de aprendizaje están diseñados para construir directamente la función discriminante. Estos métodos se denominan **Métodos discriminativos**. Otros métodos, sin embargo, intentan construir la distribución de todos los ejemplos que pertenecen a una misma clase. Estos se denominan **Métodos generativos** porque, una vez disponemos de esta, no sólo podemos clasificar nuevas muestras, sino también generarlas simplemente muestreando de la distribución.

Es importante entender esta diferencia desde el punto de vista probabilístico. Sea $X = \{x_i\}$ el conjunto de datos, y sea $Y = \{y_i\}$ el conjunto de etiquetas asociadas, con $i = 1 \ldots n$, entonces los métodos discriminativos construyen la función de distribución de probabilidad $p(y_i|x_i)$ para todo i. Los modelos generativos construyen la función de distribución de probabilidad $p(x_i|y_i)$.

Los métodos generativos son más complejos porque normalmente los ejemplos son multidimensionales, mientras que las etiquetas son unidimiensionales. Dicho de otro modo, podemos pensar fácilmente en una distribución de probabilidad de números entre 0 y 9, pero ¿¿cuál es la distribución de probabilidad de las imágenes 28×28 de ceros manuscritos??.

Los métodos generativos a su vez pueden ser paramétricos y no-paramétricos. Los paramétricos se llaman así porque el modelo de los datos pertenecerá a una familia paramétrica de funciones de distribución de probabilidad, y se buscarán aquellos parámetros con los que el modelo mejor se ajusta a los datos. El los no-paramétricos no se utiliza ninguna familia de funciones, pero eso no significa que no haya parámetros. De hecho el número de parámetros aumenta con el número de ejemplos. Por contra son más sencillos de utilizar y frecuentemente ajustan mejor.

3. ¿Cómo lograr buenos resultados?

3.1. Buenos datos

3.1.1. Cantidad

- "The Unreasonable Effectiveness of Data", P. Norvig *et al*, 2009.
- Si se tienen suficientes (muchísimos) datos todos los métodos dan un resultado similar.
- No siempre es fácil (o barato) obtener conjuntos de datos de ese tamaño.

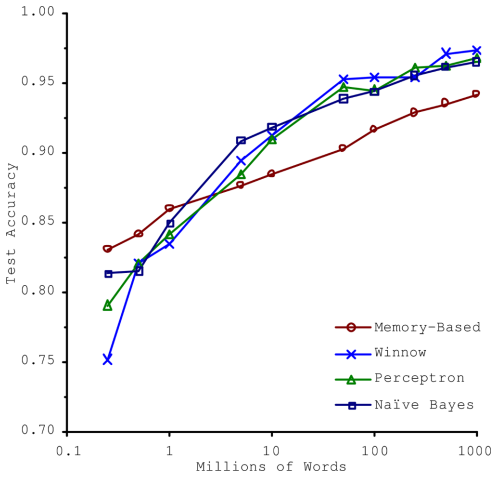

Figura 7: "Learning Curves for Confusion Set Disambiguation", M. Banko y E. Brill, 2001

3.1.2. Calidad

- Los datos deben ser representativos. Esto se consigue con ejemplos que capturen la mayor diversidad posible de casos y con características que describan fielmente los ejemplos.
- Ejemplos atípicos o a los que les faltan características son datos de mala calidad.

3.2. Buenos algoritmos

- Clasificadores demasiado sencillos pueden generalizar demasiado → Subajuste.
- Clasificadores demasiado complejos pueden memorizar → Sobreajuste.

3.3. Buenas prácticas

"...uno comienza a deformar los hechos para hacerlos encajar en las teorías en lugar de encajar las teorías en los hechos.", A. Conan Doyle, "El sabueso de los Baskerville", 1902.

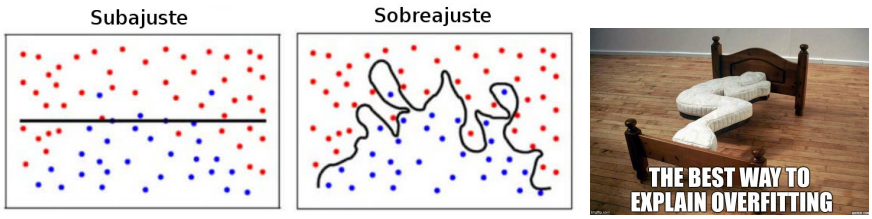

Figura 8: Subajuste vs. Sobreajuste [Fuente: Anónimo @ internet]

Tal y como recomendaba Sherlock Holmes en la cita de arriba, la primera buena práctica es dejar que los datos guíen el proceso de reconocimiento de patrones. Sin embargo hoy en día hay una cierta sensación de que el RP, y en general el ML, consiste en recopilar un montón de información y lanzar algoritmos contra ellos hasta que se logre lo que se está buscando (ver Figura 9).

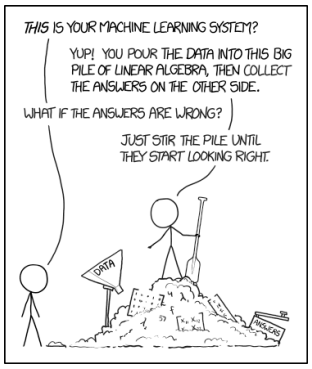

Figura 9: '*¿Este es tu sistema de ML?*','Sip, echas los datos en esta pila de algebra lineal y recoges las respuestas al otro lado','*¿y si las respuestas están mal?*','Agitas la pila hasta que empiecen a salir bien'. [Fuente: Anónimo @ internet]

El ML debería ser algo más sistemático que esto. En primer lugar conviene estudiar el problema. En muchas ocasiones esto da lugar a una buena selección de características y de algoritmos o modelos.

El entrenamiento y evaluación no debe sesgar la confianza en los resultados. Debemos intentar conocer el margen de error de la solución obtenida y averiguar si es aceptable. En caso contrario debemos volver al principio y comenzar de nuevo.

Además es frecuente que los datos sean cambiantes en el tiempo, unas veces más rápido y otras más lento. Esto da lugar a un modelo que debe ser dinámico puesto que dejará de ser válido cuando los datos sean muy diferentes a los que había cuando se creo.

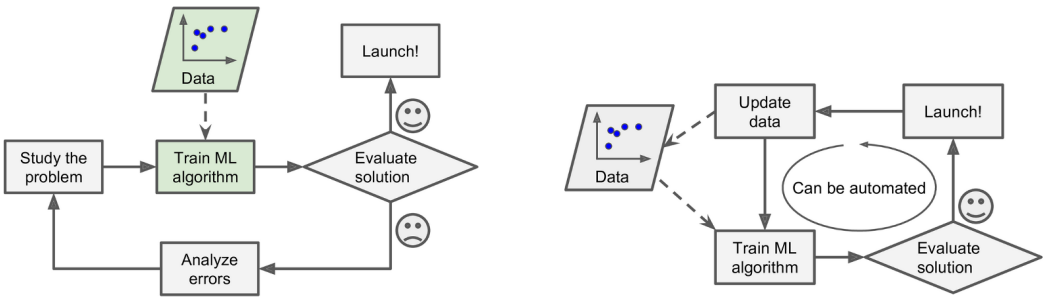

Figura 10: (Izq.) Ciclo estandar. (Der.) Ciclo dinámico

En el próximo tema aprenderemos a desarrollar un proyecto de ML completo.

Índice alfabético