Proyecto ML de principio a fin

Índice

Referencia principal

Cap. 2 de "Hands-On Machine Learning with Scikit-Learn and TensorFlow"

Al terminar este tema habremos ejecutado la mayor parte de las fases de un proyecto de ML sobre un conjunto de datos real.

1. Secuencia para un proyecto ML

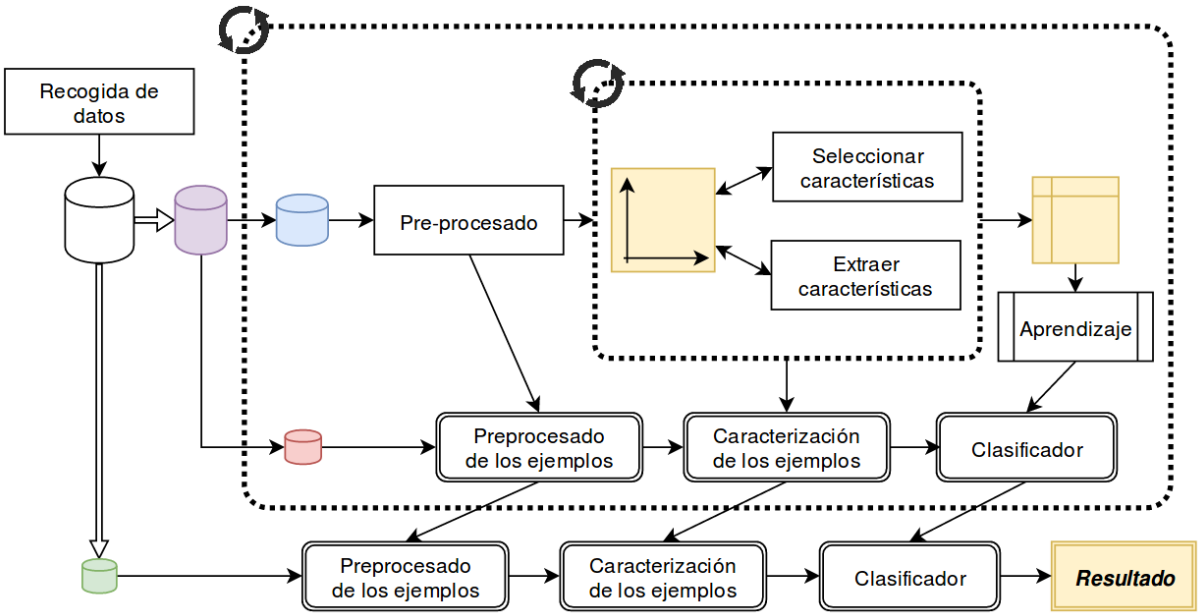


Figura 1: Diagrama de trabajo para la construcción de un clasificador [Fuente: Original de A. Cuesta]

1. Recoger los datos que van a conformar el conjunto de ejemplos (en color blanco)
2. Reservar un subconjunto de datos para *Test* (en color verde).
3. Separar el resto de los datos en dos:
   - *Entrenamiento* (en color azul)
   - *Validación* (en color rojo)
4. Casi siempre es necesario preprocesar los datos de entrenamiento.

   El preprocesado suele consistir en una normalización de los datos (cambio de escala)

   En ese caso, es importante guardar tanto el método empleado como los parámetros que se utilizaron ya que será necesario para procesar los datos de validación y test.
5. Realizar *ingeniería de características* para lograr una representación de los datos con mayor potencial discriminante.

   Esto se consigue mediante selección de características directamente de los atributos de los datos o mediante extracción, realizando operaciones con ellas.

   Es recomendable reducir la dimensionalidad a 2, que puede ser representado en un gráfico.

   Esta tarea se puede realizar varias veces hasta llegar a un representación adecuada.
6. El resultado es el conjunto de datos sobre el que se puede aplicar el algoritmo de aprendizaje.

   Este conjunto es una tabla donde las filas son ejemplos y las columnas son características
7. El método de aprendizaje suele tener parámetros e hiperparámetros, por lo que es necesario ejecutar varias iteraciones del mismo hasta conseguir un buen clasificador.

   En estas iteraciones entra en juego el conjunto de datos de validación, al que se le deben aplicar los mismos métodos de preprocesado y caracterización que al de entrenamiento.
8. El resultado es un clasificador con el que ya podemos probar los datos del conjunto de test, que serán los que den la medida del verdadero rendimiento.

   Dicho clasificador puede ser también implementado en el sistema final y ponerlo a funcionar desde ese momento. Aunque es importante hacer un seguimiento de su funcionamiento y sus resultados para actualizarlo cuando sea necesario.

2. Exploración de los datos

Esta asignatura está enfocada a problemas supervisados, es decir que el objetivo es construir clasificadores. Por tanto, de ahora en adelante tomaremos como punto de partida un conjunto de datos donde cada ejemplo tiene asociada una etiqueta (en inglés se usa indistintamente *label* y *tag*) que nos es dado y sobre el cual se desea, como mínimo, realizar una clasificación binaria, es decir que al menos habrá ejemplos de dos clases distintas y además disjuntas (un ejemplo no puede pertenecer a las dos clases a la vez). Además debemos suponer que las dos clases están más o menos equilibradas en el número de ejemplos. Según avancemos en la asignatura iremos relajando estas imposiciones para aprender más técnicas.

2.1. Obtener los datos

Para explicar la secuencia completa de un proyecto de ML vamos a utilizar el conjunto de datos *de juguete* que se encuentra en el aula virtual comprimido en el archivo `0vs1.zip`

Este conjunto consiste en 1000 imágenes de 28×28 ceros y otras tantas de unos obtenidas del MNIST. Cada uno se presenta en una tabla 1000×784 grabada en formato CSV.

Como todos los '0' están en la mismo fichero, y lo mismo con todos los '1', no hay una etiqueta explícita para cada ejemplo; es algo implícito al fichero que se está leyendo. Como hemos dicho, el objetivo será construir un clasificador capaz de distinguir nuevas imágenes de ceros y unos.

2.2. Visualizar los datos

No importa cuan grande sea la tabla, que siempre la podremos representar como una *imágen*. En la Figura 2 se muestra la imagen correspondiente a los dos conjuntos del ejemplo-guía, donde cada fila es un ejemplo de '0' (a la izquierda) o de '1' (a la derecha).

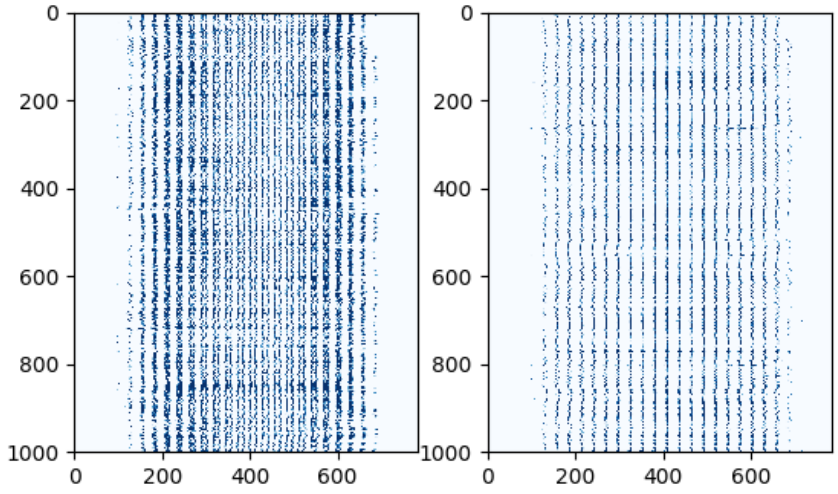


Figura 2: Conjunto de '0' (izq.) y '1' (der.) proporcionado [Fuente: Original de A. Cuesta]

Podemos comprobar que cada fila efectivamente es una imagen 28×28 seleccionando alguna y reordenando sus valores en una matriz 28×28 para después mostrar dicha matriz como una imagen. El resultado de la fila #1 de cada conjunto se muestra en la Figura 3

La visualización de los datos, junto con una exploración de las estadísticas más sencillos, permite estimar su distribución (valores máximo y mínimo, media, desviación, moda, mediana, ...), así como suele revelar la existencia de datos perdidos (o faltantes, en inglés *missing*), anómalos (*outliers*) o erróneos (*mistakes*).

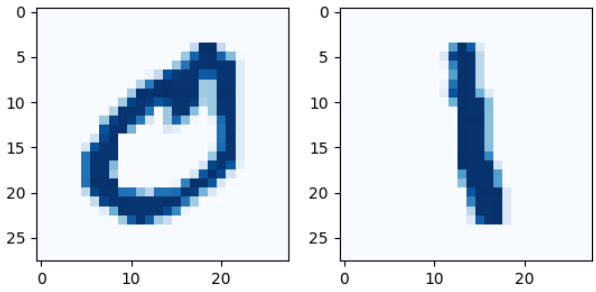


Figura 3: Fila #1 reordenada como matriz 28×28 del conjunto '0' (izq.) y del conjunto '1' (der.)
[Fuente: Original de A. Cuesta]

En el ejemplo-guía:

- Todas las imágenes están en escala de grises de 8 bits, lo cual significa que cada píxel tiene una intensidad mínima de 0 y una intensidad máxima de 255. En otras palabras todos los datos del conjunto están en el rango $[0, 255]$.
- No hay datos perdidos. Si los hubiera significaría que en algunas imágenes faltan píxeles.

  No es lo mismo que falte un píxel a que el píxel sea 0. Si por ejemplo en el archivo CSV, tuviéramos `...,128,180,220,...` y se perdiera el píxel de intensidad 180, entonces habría un dato perdido, y el archivo CSV sería `...,128,,220,...`
- No hay datos erróneos. Los habría si por ejemplo encontráramos datos inferiores a 0 o superiores a 255.

  Esto se puede detectar consultando el valor máximo y mínimo del conjunto.
- Es difícil saber si hay anómalos porque las imágenes presentan mucha variabilidad. Un modo de detectarlos es mirando las imágenes. Por ejemplo si estamos utilizando el MNIST completo y por inspección visual encontrásemos un 7 que parece un 1, o un 3 que parece un 8, serían candidatos a anómalo.

Los datos erróneos y perdidos hay que corregirlos cuando es posible o bien aplicar algún tratamiento para 'avisar' al algoritmo de aprendizaje de su existencia.

Algunos algoritmos funcionan 'igual de bien' cuando hay datos perdidos que cuando no, pero ninguna funciona bien con datos erróneos.

Los datos anómalos, por el contrario, son correctos, y por tanto no deben ser corregidos.

3. Preparar los datos

3.1. Separación del conjunto de Test

El primer paso importante es separar un cierto número de ejemplos del resto.

- Se suele utilizar entre de 10 % a un 20 % como máximo del conjunto original.
- Conviene que la selección de los ejemplos separados sea 'estratificada'.
- Este subconjunto se denomina 'de Test'.
- El conjunto de Test sólo se utilizará cuando se obtenga el clasificador definitivo.

En nuestro ejemplo-guía tenemos 1000 ceros y 1000 unos, en total 2000 imágenes. Si reservamos un 10 % para test y hacemos una selección estratificada, deberíamos elegir al azar 100 ceros y 100 unos; es decir mantenemos la proporción de ceros frente a unos del conjunto original en el conjunto de Test.

3.2. Validación cruzada

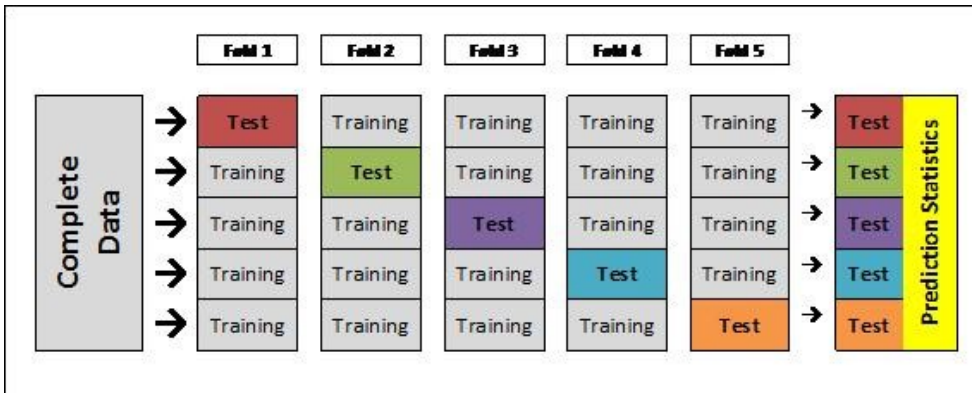


Figura 4: Ejemplo de 5-*fold* [Fuente: Anónimo @ internet]

A continuación debemos separa otra vez un cierto número de ejemplos del resto. Pero a diferencia de antes, este proceso se realizara K veces, se conoce como 'validación cruzada' (K-*fold*) y consiste en el siguiente bucle:

1. Dividir el subconjunto de datos restante de la etapa anterior en K bloques los más estratificados posible.
2. Agrupar $K-1$ bloques para formar el conjunto de 'Entrenamiento'.

   El bloque restante conforma el conjunto de 'Validación'.

   Utilizaremos el conjunto de Entrenamiento para aprender un clasificador y el conjunto de Validación para evaluarlo.
3. Repetir el paso 2 pero eligiendo un bloque diferente a los anteriores para validación.

   Cuando hayamos usado todos los bloques para validación tendremos K clasificadores.
   La Figura 4 representa el proceso para 5 bloques o iteraciones de validación cruzada (5-*fold*)

   Podemos calcular la media y desviación de su rendimiento así como el 'clasificador medio' que, si es satisfactorio, será el que tomemos como definitivo.

3.3. Preprocesado del conjunto de Entrenamiento

Esta etapa involucra la limpieza de los datos y frecuentemente un cambio de escala para que todas las columnas tengan el mismo orden de magnitud.

La limpieza de datos consiste en corregir los datos perdidos y erróneos en caso de que los haya. Cuando los ejemplos son imágenes normalmente no hay datos perdidos o erróneos salvo por corrupción de los ficheros. En su lugar se suelen aplicar técnicas como ecualización de histogramas, realce, enfocado o eliminación de ruido, que son el objetivo de otras asignaturas.

El cambio de escala es importante para que los algoritmos no se vean sesgados por el orden de magnitud de las diferentes columnas. Vamos a ver dos métodos, escalado lineal y estandarización.

Antes de pasar a explicar cada uno, es importante recordar que:

1. Cada atributo (cada columna) se escala por separado respecto de las demás
2. Se debe guardar los parámetros y el método de escalado utilizado en cada atributo, ya que estos dependen frecuentemente de los datos de entrenamiento
3. Hay que aplicar el mismo método de escalado, con los mismos parámetros, al atributo correspondiente en el conjunto de validación y test.

Escalado lineal: Supongamos que nuestro conjunto de entrenamiento consta de 3 atributos: X_1, X_2 y X_3. Cada uno de ellos tendrá un valor máximo $\overline{X}_i$ y un valor mínimo $\underline{X}_i$, siendo $i = \{1, 2, 3\}$. Mediante escalado lineal los valores de cada atributo X_i pasarán de estar en el intervalo $[\underline{X}_i, \overline{X}_i]$ a estar en el intervalo $[L_i, U_i]$ (*Lower* y *Upper*), cuyos valores decidimos nosotros, y que generalmente son iguales para todo i. Para ello debemos realizar alguna de las siguiente dos conversiones (son equivalentes):

$$Z_i = m \cdot (X_i - \overline{X}_i) + U_i \ , \quad \text{o bien} \quad Z_i = m \cdot (X_i - \underline{X}_i) + L_i,$$

donde $m = (U_i - L_i)/(\overline{X}_i - \underline{X}_i)$, y Z_i es el atributo escalado correspondiente a X_i.
Naturalmente esto es válido para cualquier número de atributos.

Estandarización: Supongamos ahora que nuestro conjunto de entrenamiento consta de 3 atributos: X_1, X_2 y X_3. Cada uno de ellos tendrá una media μ_i y una desviación estándar σ_i, para $i = \{1, 2, 3\}$. Entonces la estandarización realiza la siguiente conversión:

$$Z_i = (X_i - \mu_i)/\sigma_i,$$

donde, igual que antes, Z_i es el atributo estandarizado correspondiente a X_i y naturalmente esto es válido para cualquier número de atributos. Como resultado, los valores de Z_i estarán, en su gran mayoría, en el intervalo $[-3\sigma, +3\sigma]$.

	Datos medidos			Datos escalados en [-1,+1]			Datos estandarizados		
Marca tiempo	Sensor 1	Sensor 2	Sensor 3	Sensor 1	Sensor 2	Sensor 3	Sensor 1	Sensor 2	Sensor 3
0	3.200	17,5	2,25	-0,25	0,13	0,68	-1,45	0,28	1,00
250	4.050	14	2,18	0,57	-0,20	0,59	0,93	-0,74	0,74
500	4.050	11,33	1,83	0,57	-0,45	0,22	0,93	-1,52	-0,47
119500	3.700	19	2,20	0,23	0,27	0,62	-0,05	0,72	0,33
119750	3.875	16,75	2,10	0,40	0,06	0,51	0,44	0,06	0,48
Media =	3.718	16,53	1,96				0,00 ✓✓	0,00 ✓✓	0,00 ✓✓
Desv.=	358	3,43	0,29				1,00 ✓	1,00 ✓	1,00 ✓
Máx=	4.500	26,75	2,55	1,00 ✓	1,00 ✓	1,00 ✓			
Mín=	2.425	5,50	0,70	-1,00 ✓	-1,00 ✓	-1,00 ✓			

Figura 5: Datos medidos, escalados a [-1,+1] y normalizados. [Fuente: Original de A. Cuesta]

En el ejemplo-guía sabemos que todas las imágenes van a tener un valor mínimo de 0 y un valor máximo de 255. Todos los datos están en la misma escala, por lo que, en principio, no sería necesario ningún tipo de cambio de escala. Sin embargo, si reescalamos al intervalo $[0, 1]$ y usamos `float` tendremos mayor rango de representación que si utilizamos `uint8`.

4. Ingeniería de características

En este punto el conjunto de entrenamiento consiste en una serie de ejemplos descritos por un vector de atributos tal cual nos ha sido proporcionado.

En el ejemplo-guía, y muy frecuentemente en visión artificial, el vector de atributos simplemente consiste en una lista con el valor de todos los píxeles de la imagen.

¿Son todos los datos igualmente necesarios? ¿Es posible encontrar un conjunto de atributos a partir de estos, de menor tamaño, pero que tengan la misma capacidad de caracterización?

Generalmente los algoritmos de ML no se aplican directamente sobre los atributos de los datos, sino sobre una descripción alternativa de cada ejemplo que consiste en un conjunto de *características* (*features*), que en ocasiones son seleccionadas por alguien experto en los datos que se están utilizando y en ocasiones se extraen a partir de relacionar los atributos entre sí mediante expresiones matemáticas.

Antes de continuar es importante resaltar la notación que estamos siguiendo en este curso. Los datos proporcionados se pueden representar como una tabla a cuyas columnas les hemos llamado inicialmente **atributos**. Sin embargo el algoritmo siempre se aplica sobre las **características**, que son seleccionadas o extraídas. Nada impide utilizar los datos proporcionados directamente, sería el caso particular en el que hemos seleccionado todos los atributos para convertirlos en características.

El motivo de esta aclaración es que en inglés frecuentemente se encuentra la palabra *characteristics* para lo que aquí hemos denominado atributos, y se utiliza *features* para las características sobre las que se trabaja; pero ambas palabras en inglés tienen la misma traducción al español.

En nuestro ejemplo-guía vamos a comenzar creando algunas características a partir de las imágenes. Suponiendo que previamente hemos escalado linealmente la intensidad de los píxeles al rango $[0, 1]$, seguimos el siguiente proceso:

1. Sumar la intensidad de los píxeles en horizontal y en vertical. De esta operación obtenemos una proyección horizontal de 28 *valores* en horizontal y otra proyección similar en vertical.
2. Llamamos W a la *anchura* del dígito, y la definimos como el número de valores seguidos en la proyección horizontal superiores a un cierto umbral θ.
3. De manera similar definimos la altura H con la proyección vertical.
4. Anotamos la posición de los 3 valores más altos de la proyección horizontal y los llamamos w_1, w_2, w_3.
5. Anotamos la posición de los 3 valores más altos de la proyección vertical y los llamamos h_1, h_2, h_3.
6. Construimos el siguiente vector de características de cada imagen: $[W, w_1, w_2, w_3, H, h_1, h_2, h_3]$

En la Figura 6 se puede ver un ejemplo de la obtención del vector de características.

4.1. Selección de características

Un comienzo recomendable consiste en enfrentar dos características en un gráfico, donde los puntos de una misma etiqueta tienen un mismo aspecto, diferente al de los puntos de cualquier otra etiqueta. La Figura 7 muestra 3 opciones: W vs. H (arriba), w_1 vs. h_1 (abajo izquierda) W vs. w_1 (abajo derecha). Además en todas excepto la de arriba izquierda se ha aplicado un **jitter**. El *jitter* es una agitación que se añade a cada coordenada cuyo dominio sea discreto para distinguir todos los puntos entre sí, por ejemplo sumando un valor aleatorio dos o tres órdenes de magnitud menores que la escala de dicha coordenada. Esto hace que sean más visibles cuando se repiten los valores con frecuencia.

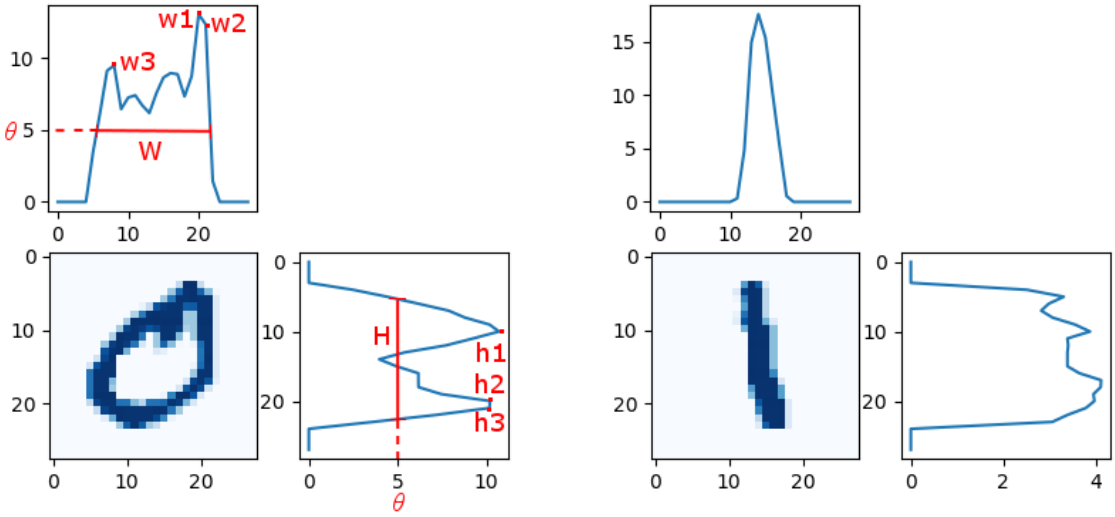
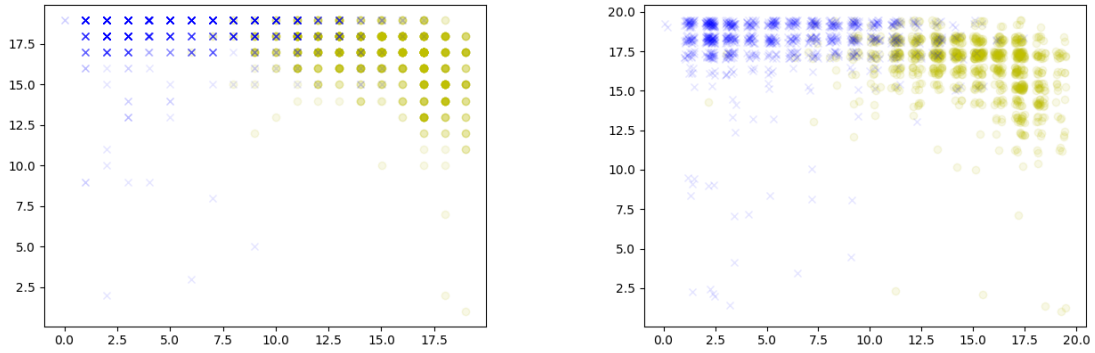
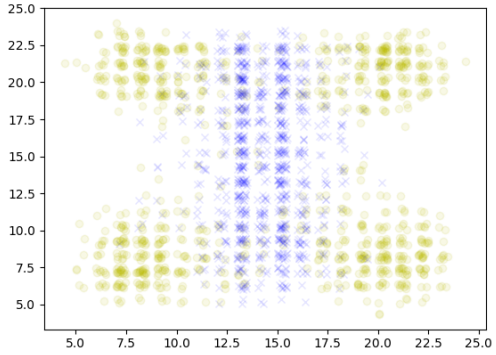


Figura 6: Proyección horizontal y vertical de un '0' y un '1'. [Fuente: Original de A. Cuesta]

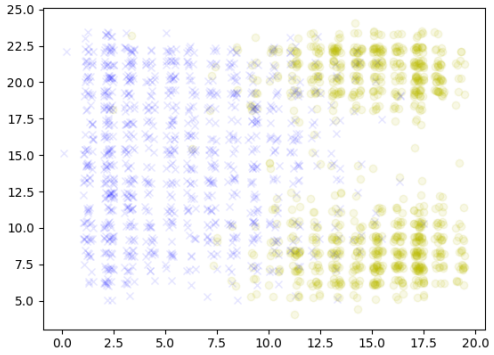


(a) W vs. H sin *jitter*

(b) W vs. H con *jitter*



(c) w_1 vs. h_1 con *jitter*

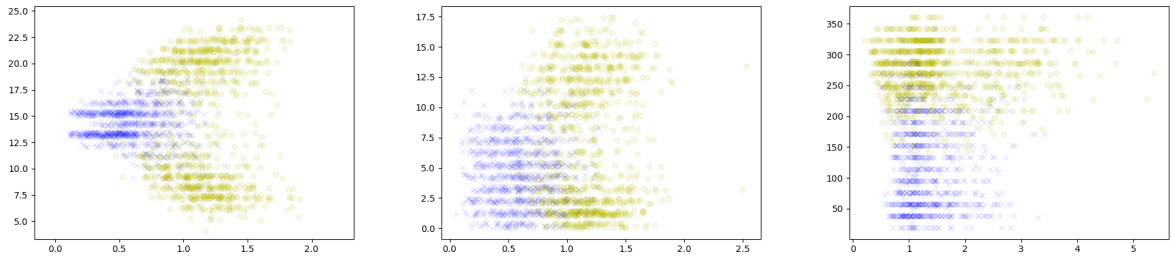


(d) W vs. w_1 con *jitter*

Figura 7: Conjunto de entrenamiento seleccionando 2 características. [Fuente: Original de A. Cuesta]

4.2. Extracción de características

La extracción de características es la creación de nuevas características mediante expresiones matemáticas. En la Figura 8 se muestran 3 opciones. A la izquierda se representa w_1/W vs. h_1/H; en el centro w_1/w_2 vs. h_1/h_2; y a la izquierda W^2 vs. H^2.



(a) w_1/W vs. h_1/H (b) w_1/w_2 vs. h_1/h_2 (c) W^2 vs. H^2.

Figura 8: Conjunto de entrenamiento extrayendo 2 características. [Fuente: Original de A. Cuesta]

4.3. Discusión

En esta sección hemos utilizado dos características, con 7 variantes distintas. Esto nos da la posibilidad de asociar cada ejemplo i a un punto x_i de un espacio vectorial 2D, donde cada característica es una dimensión.

La ventaja es que podemos representar los ejemplos en un gráfico, lo cual es una importante ayuda para los pasos siguientes. Por ejemplo podemos localizar características que separen linealmente el conjunto o detectar formas en las clases que nos guíen hacia el mejor método de clasificación.

La desventaja es que seguramente hayamos perdido información discriminante, ya que hemos 'comprimido' nuestro conjunto de datos a dos columnas. Esto se puede apreciar en aquellos lugares del gráfico donde aparecen dos puntos de diferentes clases superpuestos o muy cercanos.

Aunque 'ver' el conjunto de datos es interesante tener tan pocas características no suele ser una buena decisión. Como hemos visto, cada característica aporta una dimensión al espacio en el que viven los puntos. Sobre el papel sólo podemos representar eficientemente 2 dimensiones, y con algo de imaginación 3 y hasta 4. Pero es habitual tener bastantes más. Por ejemplo, si consideramos que la intensidad de cada píxel es una característica, las imágenes del MNIST viven en un espacio de ¡ 784 dimensiones !

Por otro lado, las características que hemos usado han sido creadas para el caso concreto de distinguir '0' de '1'. Pero ¿serían válidas para distinguir otros dos números? ¿Y si quiero distinguir todos los dígitos? Sería interesante contar con un método más independiente del problema, que funcione con unos criterios objetivos basados en los datos y de manera automática. Nosotros estudiaremos el Análisis de Componentes Principales (*Principal Component Analysis*, PCA).

Finalmente, al igual que en el preprocesado, hay que guardar el método con el que se han obtenido las características que vamos a aprender, ya que sólo hemos usado los ejemplos del conjunto de entrenamiento, para poder transformar del mismo modo los conjuntos de validación y test.

5. Selección del método de entrenamiento y ejecución

Cada clasificador tiene una **expresividad** diferente. Es decir, genera superficies de decisión con un cierto carácter. Así, unos se adaptan mejor al conjunto de datos que otros, y por tanto logran mayor poder discriminante.

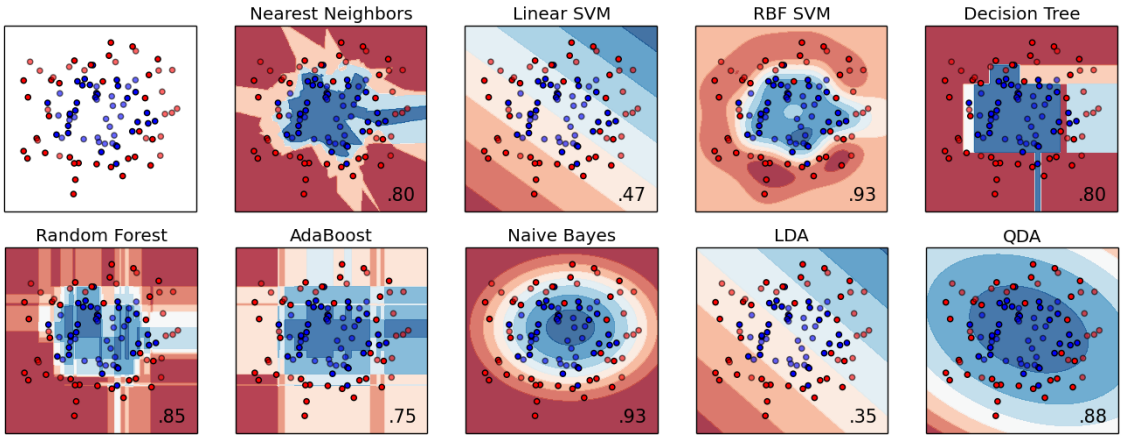


Figura 9: Expresividad de diferentes métodos de clasificación. [Fuente: SciKit-Learn.org]

En nuestro ejemplo-guía vamos a tratar de separar linealmente el conjunto de entrenamiento. Evidentemente, entre las selecciones de características mostradas en la Figura 7 la (c) no parece apropiada para este propósito. Entre la (a,b) y la (d), la que tiene las muestras de las cada clase más compactada es la (a,b) (es la misma sin/con *jitter*), por lo que elegimos esta.

5.1. Clasificador lineal

En este caso la superficie de clasificación será un hiperplano. Pero en nuestro ejemplo-guía, al ser únicamente bidimensional, colapsa a una simple recta, con ecuación implícita

$$w_0 + w_x x + w_y y = 0,$$

donde cada ejemplo es un punto del este espacio 2D con coordenadas (x, y); y $\{w_0, w_x, w_y\}$ son los pesos o parámetros del clasificador.

Es decir que, una vez finalizado el proceso de aprendizaje, tendremos los pesos óptimos $\{w_0^*, w_x^*, w_y^*\}$. Si queremos pintar la recta de decisión debemos recordar que los puntos de la recta son aquellos que satisfacen la ecuación implícita. Por tanto despejando y tenemos

$$y = -\frac{w_x}{w_y} x - \frac{w_0}{w_y}.$$

Así, dando valores a x obtenemos los y correspondientes a dicha recta.

Se han utilizado dos métodos de clasificación para modelos lineales: Descenso del gradiente estocástico (SGD) y Clasificador basado en Vectores Soporte (SVC)[1]. En el módulo SciKit-Learn de Python, cada uno de ellos se instancia de su clase respectiva. En ambas el atributo de clase `coef_` guarda los pesos de cada dimensión en una lista, en este caso serían $[[w_x, w_y]]$, y `intercep_` el término independiente en otra, aquí $[w_0]$.

5.2. Resultado

Puesto que ambos son lineales, su expresividad es similar. Sin embargo, al ejecutar dos veces cada uno de ellos, debido a la naturaleza estocástica de SGD se puede apreciar a simple vista

[1] El objetivo de este tema es tener una panorámica. Más adelante estudiaremos en detalle estos y otros métodos.

en la Figura 10 que la superficie de decisión obtenida varía más que con SVC; aunque cuantitativamente podemos comprobar que los pesos obtenidos también son diferentes. Por tanto es necesario introducir medidas para estimar (y mejorar) la fiabilidad del clasificador.

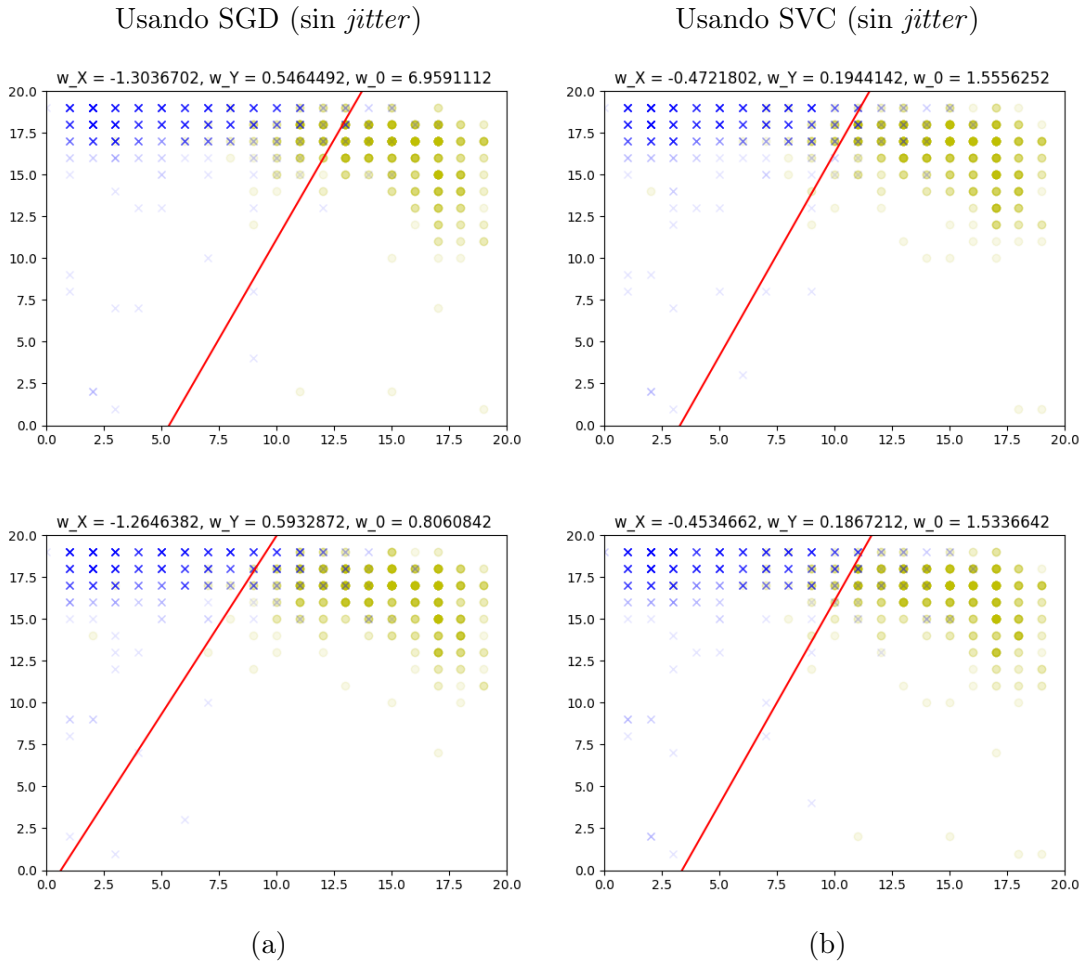


Figura 10: Superficie de decisión lineal con dos métodos de clasificación ejecutado dos veces (arriba y abajo). (a) Descenso de gradiente estocástico (SGD) (b) Clasificador basado en vectores soporte (SVC). Se puede apreciar que SGD varía más que SVC. [Fuente: Original de A. Cuesta]

En el tema siguiente estudiaremos los elementos necesarios para realizar una clasificación lineal.

Índice alfabético