Evaluación de los resultados del aprendizaje

TÉCNICAS ESTADÍSTICAS PARA EL APRENDIZAJE II

Máster Universitario en Estadística Computacional
y Ciencia de Datos para la Toma de Decisiones

Evaluación de los resultados del aprendizaje

Introducción

Métricas de evaluación

- Precisión y tasa de error
- Predicciones cuantitativas

Métodos de evaluación

- Conjuntos de entrenamiento, validación y prueba
- Muestreo sin reemplazo: Validación cruzada
- Muestreo con reemplazo: *Bootstrapping*

Introducción

Métricas de evaluación

- Precisión y tasa de error
- Predicciones cuantitativas

Métodos de evaluación

- Conjuntos de entrenamiento, validación y prueba
- Muestreo sin reemplazo: Validación cruzada
- Muestreo con reemplazo: *Bootstrapping*

Introducción

Cuando queremos **resolver un problema** utilizando técnicas de aprendizaje en Inteligencia Artificial, lo que hacemos es **recopilar datos** que recojan las variables que creamos que pueden ser relevantes para nuestro problema. Estos datos describen **casos particulares del problema** que pretendemos modelar.

Una vez que dispongamos de **suficientes datos**, los utilizaremos para **construir un modelo** mediante un proceso de entrenamiento.

Ese entrenamiento dará como resultado un **modelo aplicable a datos diferentes de los utilizados en su entrenamiento**, y será el que le permita al modelo ser capaz de **generalizar** a partir de ejemplos concretos.

Cuantos **más datos** obtengamos, más fácilmente podremos diferenciar patrones válidos de patrones debidos a irregularidades o errores en el conjunto de entrenamiento, siendo **mejores** los **resultados.**

Introducción

Sea cual sea el tipo de técnica de aprendizaje que decidamos emplear, el **modelo** obtenido como resultado del proceso de aprendizaje, no será más que una **simplificación de la realidad** y estará **sujeto a errores**.

Por tanto, debemos **evaluar la calidad del modelo** obtenido para ver si realmente nos puede servir en la práctica.

Para ello, diferenciaremos dos aspectos complementarios:

Métricas de evaluación: cómo medir la calidad de un modelo obtenido mediante el uso de técnicas de aprendizaje automático.

Métodos de evaluación: cómo realizar una estimación fiable de las métricas que nos indican la calidad de un modelo.

Evaluación de los resultados del aprendizaje

Introducción

Métricas de evaluación

- Precisión y tasa de error
- Predicciones cuantitativas

Métodos de evaluación

- Conjuntos de entrenamiento, validación y prueba
- Muestreo sin reemplazo: Validación cruzada
- Muestreo con reemplazo: *Bootstrapping*

Métricas de evaluación

Comencemos por el caso más habitual y sencillo: un **problema de clasificación binaria**.

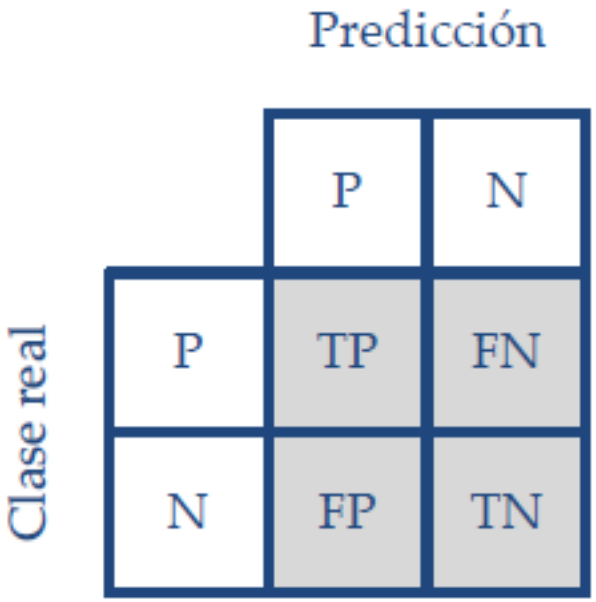
Deseamos construir un modelo de clasificación que nos permita discriminar entre dos clases diferentes: la **clase positiva (P)**, que corresponde a los ejemplos que deseamos ser capaces de identificar, y la **clase negativa (N)**, con la que denominamos a los demás ejemplos.

Cuando aplicamos un modelo de clasificación a un conjunto de datos previamente etiquetado, podemos ver **cómo etiqueta los diferentes ejemplos nuestro clasificador**.

Sólo existen cuatro posibilidades, que podemos representar en una **matriz de contingencia** de tamaño 2x2, también llamada **matriz de confusión**:

Verdaderos positivos [*TP: True Positive*]: Los ejemplos de la clase positiva que nuestro clasificador es capaz de clasificar correctamente.

Falsos positivos [*FP: False Positive*]: Los ejemplos que, aun no siendo de la clase positiva, nuestro clasificador predice que sí lo son.



Falsos negativos [*FN: False Negative*]: Los errores que comete nuestro clasificador en sentido contrario, diciéndonos que no son de la clase positiva cuando en realidad sí que lo son.

Verdaderos negativos [*TN: True Negative*]: Los ejemplos de la clase negativa que nuestro clasificador clasifica correctamente.

Si nos encontramos ante un problema de **clasificación con múltiples clases**, la matriz de confusión tendría una dimensión de K×K, donde K es el número de clases del problema.

Analizando las filas y columnas de la matriz de confusión podemos **ver dónde acierta y dónde se equivoca nuestro modelo** de clasificación, lo que nos puede indicar posibles problemas y sugerir qué es lo que deberíamos intentar cambiar.

Ahora bien, aparte de poder analizar con detalle cómo clasifica nuestro modelo de clasificación los ejemplos de las diferentes clases, nos gustaría poder disponer de **medidas numéricas** que nos ayudasen a resumir el contenido de la matriz de contingencia y **comparar modelos** de clasificación alternativos.

Precisión y tasa de error

La **métrica más utilizada** para resumir el rendimiento de un modelo de aprendizaje supervisado es su **precisión [*accuracy*]**.

La precisión nos indica la **proporción de ejemplos que un clasificador es capaz de clasificar correctamente**, indicada habitualmente en forma de tanto por ciento:

$$precisión = accuracy = \frac{TP + TN}{TP + TN + FP + FN} = \frac{aciertos}{n}$$

donde n es el número total de ejemplos del conjunto de datos.

Si tenemos un problema con **K clases diferentes**, simplemente **sumaríamos** los elementos correspondientes a la **diagonal** de la matriz de confusión, para obtener los aciertos de nuestro clasificador.

Precisión y tasa de error

De forma alternativa, podríamos contabilizar los fallos de nuestro clasificador y resumir su rendimiento mediante una **tasa de error**:

$$error = \frac{FP + FN}{n} = \frac{errores}{n} = 1 - accuracy$$

Por lo general, nunca conseguiremos eliminar por completo el error en nuestros modelos. Entonces, ¿cómo sabemos si nuestro **clasificador** es lo **suficientemente bueno**?

Para responder, nos puede venir bien disponer de algún punto de referencia con el que establecer una **comparación**.

Es habitual utilizar un modelo de clasificación simple y ver hasta qué punto nuestro modelo mejora los resultados del **clasificador base**.

Precisión y tasa de error

¿Existe la posibilidad de **mejorar el rendimiento** de nuestro clasificador?

En este caso, también nos sería de utilidad encontrar un punto de **referencia** que nos sirva de objetivo.

Si, por ejemplo, podemos conseguir una estimación de la tasa de error cometida por el **ser humano** en el problema que pretendemos automatizar, nos puede servir para fijar un sistema de referencia.

En ocasiones se utiliza este valor para **determinar cuándo un problema ha sido resuelto en I.A.** Hasta hace muy poco, los sistemas de reconocimiento de voz eran peores que los seres humanos. Algo que cambió gracias al uso de redes neuronales y que posibilitó el despegue de los asistentes virtuales de los smartphones.

Precisión y tasa de error

Medidas de calidad como la precisión [*accuracy*] o la tasa de error no resultan adecuadas en problemas de clasificación con clases poco equilibradas, los **problemas de clasificación no balanceada**.

Ejemplo: supongamos que, para un problema determinado, disponemos de un conjunto de datos con 10000 ejemplos. De esos, sólo 10 son de la clase P, mientras que los 9990 restantes son de la clase N.

En este caso, podríamos construir un **clasificador** por defecto, **que siempre diga que los ejemplos son de la clase más frecuente**. Ese clasificador, aunque totalmente inútil para resolver nuestro problema de clasificación, obtendrá una **precisión del 99.9%**, difícilmente mejorable. Eso sí, esa elevada precisión es totalmente engañosa, ya que será incapaz de detectar un solo ejemplo de la clase P.

Precisión y tasa de error

Estas situaciones son muy habituales en la práctica, principalmente en los **sistemas de detección de anomalías**:

- Realizar el control de calidad en una fábrica
- Detectar fallos en sistemas mecánicos
- Marcar como sospechosas operaciones financieras fraudulentas
- Detectar intrusiones en un sistema
- Identificar el comportamiento anómalo de un programa que podría indicar la presencia de virus informáticos

En todas esas aplicaciones, es necesario afinar y no resumir la calidad de nuestro modelo con una simple medida de precisión o error.

Predicciones cuantitativas

En ocasiones, hemos de construir modelos capaces de **predecir el valor de una variable numérica**, que toma valores dentro de un rango continuo.

En la predicción de valores de tipo numérico, nuestro **modelo** de aprendizaje será de la forma $\hat{\boldsymbol{y}} = \boldsymbol{f}(\boldsymbol{x})$. A partir de un conjunto de datos x, intentaremos estimar el valor de y por medio de una función $f(x)$.

En un **modelo de regresión**, los valores asociados a x incluirán todas aquellas variables que hayamos decidido considerar en nuestro modelo.

En un **modelo de predicción de series temporales**, los valores x corresponden a los valores que la serie temporal tomó en el pasado y cuyo futuro queremos predecir utilizando $f(x)$.

Predicciones cuantitativas

La **medida de error más utilizada** para estimar la calidad de un modelo de predicción cuantitativa es el **error cuadrático medio [MSE: *Mean Squared Error*]**, que aplicamos sobre las n muestras del conjunto de datos sobre el que estamos estimando la calidad de nuestro modelo cuantitativo:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(f(x) - y)^2$$

¿Por qué se utiliza el cuadrado del error y no simplemente la diferencia entre la estimación y el valor observado?

Porque usando directamente las diferencias $f(x) - y$, los errores por encima y por debajo se compensarían, con lo que las medidas de error no servirían para nada.

Predicciones cuantitativas

En ocasiones, también se utiliza la **suma de los errores al cuadrado [SSE: *Sum of Squared Errors*]**:

$$SSE = n * MSE = \sum_{i=1}^{n} (f(x) - y)^2$$

A la hora de comparar alternativas, si el valor de n no cambia, nos da igual emplear MSE o SSE.

Predicciones cuantitativas

En el **análisis estadístico de datos**, se suele emplear la **desviación típica** σ en lugar de la varianza σ^2 porque la desviación se mide en las **mismas unidades que la variable** que estamos analizando, lo que facilita la interpretación de los resultados de nuestro análisis.

Análogamente, podemos calcular la raíz cuadrada del error cuadrático medio para obtener una medida de error denominada **RMSE [*Root Mean Squared Error*]**, que representa la desviación de las diferencias entre los valores $f(x)$ e y:

$$RMSE = \sqrt{MSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(f(x) - y)^2}$$

Predicciones cuantitativas

SSE, MSE y RMSE sirven para **agregar en una única medida las magnitudes de los errores cometidos** sobre un conjunto de datos, por lo que desempeñan un valor similar a la precisión o la tasa de error en los problemas de clasificación.

Nos permiten comparar los errores cometidos por modelos alternativos para un mismo conjunto de predicciones, pero no para comparar predicciones sobre distintos conjuntos de datos, al ser **dependientes de la escala de los datos**.

Estas medidas de error se pueden **normalizar** para comparar modelos sobre conjuntos de datos que se representen sobre escalas diferentes.

Predicciones cuantitativas

Aunque no hay una normalización particular que se considere estándar, podemos **normalizar** una medida como **RMSE** utilizando, o bien la media aritmética de la variable sobre la que estamos midiendo el error :

$$NRMSE_{mean} = \frac{RMSE}{\bar{y}}$$

o bien el rango de los valores de dicha variable:

$$NRMSE_{range} = \frac{RMSE}{y_{max} - y_{min}}$$

Todas las medidas vistas hasta ahora incorporan el **cuadrado del error** para los errores cometidos. El problema de esto es que hace que las medidas sean **sensibles** a la presencia de anomalías u ***outliers*** en los datos, porque el efecto de cada error individual es proporcional al cuadrado del tamaño del error.

Predicciones cuantitativas

Una alternativa es utilizar el **error absoluto medio o MAE [*Mean Absolute Error*]**:

$$MAE = \frac{1}{n}\sum_{i=1}^{n} |f(x) - y|$$

MAE no es más que la media de las diferencias, en valor absoluto, entre las predicciones $f(x)$ y las observaciones y.

La contribución directa de cada error individual es proporcional al valor absoluto del error, lo que hace al error absoluto medio más robusto frente a la presencia de ruido y *outliers* en el conjunto de datos.

Predicciones cuantitativas

Igual que sucede con RMSE, existen distintas formas de **normalizar el error absoluto medio** para poder establecer comparaciones entre los resultados obtenidos sobre diferentes conjuntos de datos.

La medida normalizada de error absoluto más conocida tal vez sea el **error absoluto medio porcentual MAPE [*Mean Absolute Percentage Error*]**:

$$MAPE = \frac{100}{n} \sum_{i=1}^{n} \frac{|f(x) - y|}{|y|}$$

MAPE nos da el error medio cometido en la estimación $f(x)$ expresado como un tanto por ciento sobre el valor observado de y.

Otras medidas

Aunque usualmente se está interesado en conseguir modelos de aprendizaje lo más precisos posible, la precisión de un modelo no siempre será el único criterio que utilizaremos para evaluar la calidad de los modelos construidos. **Otros criterios** suelen ser:

- Interpretabilidad
- Complejidad
- Estabilidad
- Eficiencia
- Escalabilidad
- …

Además, se debe valorar también si hay restricciones en cuanto al uso que se puede hacer de distintos tipos de recursos, como tiempo, espacio, uso de energía o dinero.

Evaluación de los resultados del aprendizaje

Introducción

Métricas de evaluación

- Precisión y tasa de error
- Predicciones cuantitativas

Métodos de evaluación

- Conjuntos de entrenamiento, validación y prueba
- Muestreo sin reemplazo: Validación cruzada
- Muestreo con reemplazo: *Bootstrapping*

Métodos de evaluación

En aprendizaje automático, el modelo lo construimos a partir de un conjunto de **datos.**

Dado que no disponemos de otros datos, ese conjunto de datos tendremos que utilizarlo para, además de **construir el modelo**, **estimar su calidad**, es decir, para evaluar, en media, cuál será el comportamiento del modelo una vez que nos pongamos a utilizarlo.

En vez de utilizar todos los datos disponibles para entrenar el modelo, **una parte no la utilizaremos en el entrenamiento**.

Será la que nos sirva para estimar cómo se comportará el modelo con datos diferentes a los datos con los que se ha construido. Es el llamado **conjunto de prueba**.

Conjuntos de entrenamiento, validación y prueba

Nunca se debe utilizar el **conjunto de entrenamiento para evaluar la calidad de un modelo** obtenido por técnicas de aprendizaje automático. Bajo ninguna circunstancia.

Una vez construido un modelo a partir del conjunto de entrenamiento, se usa dicho modelo sobre los datos del **conjunto de prueba**, que hasta ese momento **nunca habrá visto**.

Comparando las etiquetas asociadas a los casos del conjunto de prueba con el resultado de aplicar el modelo, se obtiene una medida de su **precisión**. Si la precisión es **aceptable** sobre el conjunto de prueba, y sólo entonces, podremos utilizar el modelo en la práctica para **clasificar nuevos casos de los que realmente desconocemos su clase**.

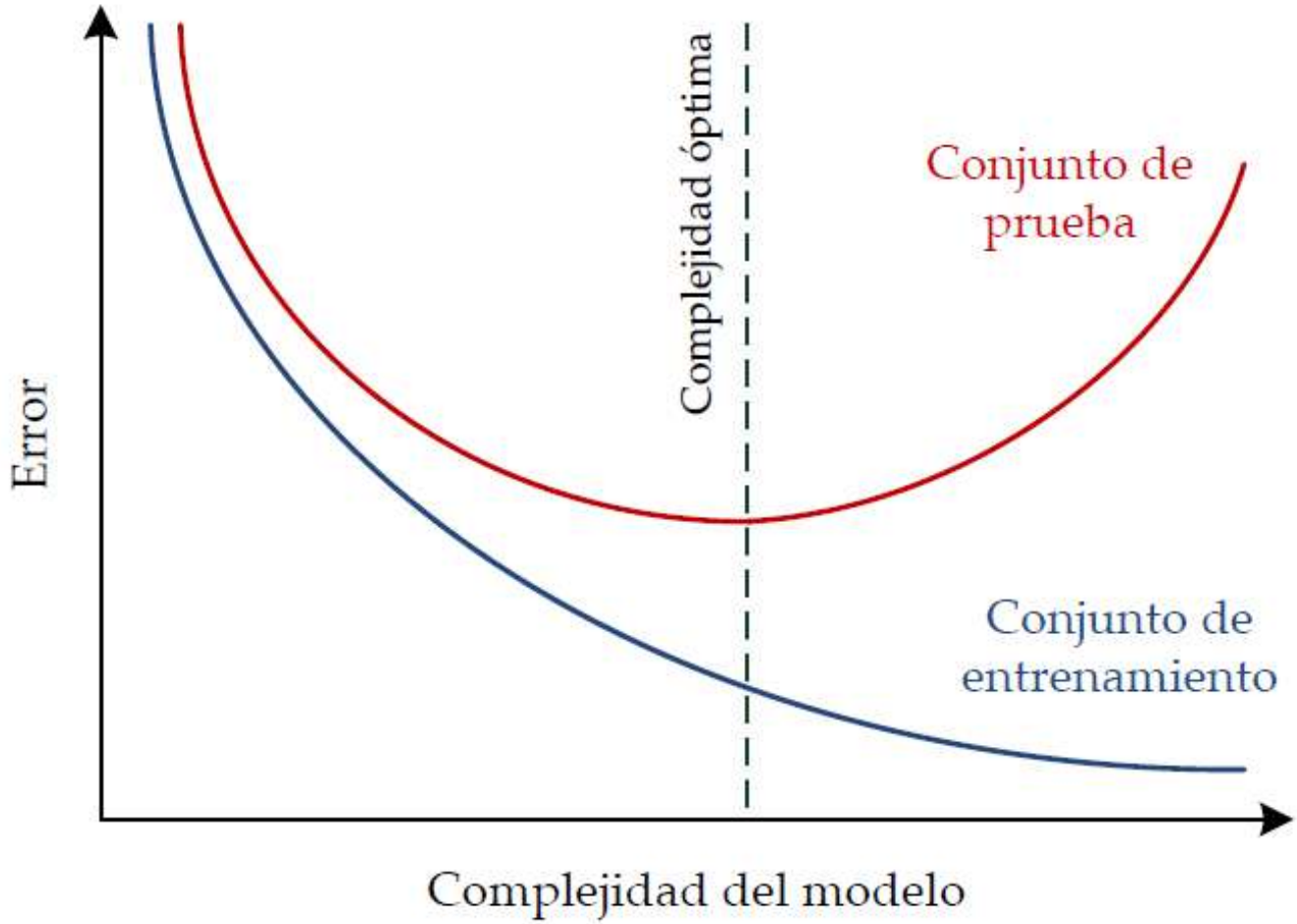
Conjuntos de entrenamiento, validación y prueba

Un problema muy habitual en aprendizaje automático es el problema del **sobreaprendizaje [*overfitting*]**. En general, cuanto mayor sea la complejidad del modelo aprendido, más tenderá a ajustarse al conjunto de datos de entrenamiento utilizado en su construcción.

Sobre datos reales, ese sobreajuste puede hacer que el **modelo** sea **menos útil** para trabajar con datos nuevos, como los que se encontrará cuando empecemos a utilizarlo, es decir, **generalizará peor**.

Esa es la causa de que el error en el conjunto de entrenamiento no sea un buen estimador de la precisión del modelo.

Métodos de evaluación



El problema del sobreaprendizaje: error en el conjunto de entrenamiento vs. error en el conjunto de prueba

Conjuntos de entrenamiento, validación y prueba

Por ejemplo, podemos reservar **dos tercios** de los datos disponibles para entrenar el modelo y **el tercio restante** lo utilizaríamos para estimar su calidad. También es habitual optar por una descomposición **70%-30%** o **80%-20%.**

Lo importante es que **no nos olvidemos de dividir el conjunto de datos disponible** en un conjunto de entrenamiento (para construir el modelo) y un conjunto de prueba (para evaluar el modelo).

En un contexto ideal, **cuantos más datos tengamos** a nuestra disposición, **mejor podremos entrenar** nuestro modelo y cuanto mayor sea el tamaño del conjunto de test, **más precisas** serán nuestras **estimaciones** de la calidad del modelo.

Conjuntos de entrenamiento, validación y prueba

Para que nuestra estimación de la calidad de un modelo sea realista, los conjuntos de datos de entrenamiento y prueba deben ser, además de **independientes**, **ejemplos representativos** de los datos asociados al problema que pretendemos resolver.

Por ejemplo, en **problemas de clasificación no balanceada**, al descomponer el conjunto de datos en conjunto de entrenamiento y conjunto de prueba deberíamos evitar que una clase poco frecuente pueda no estar representada en el conjunto de prueba.

La **tasa de error en el conjunto de prueba** es sólo una estimación de la tasa de error real del modelo. Esa tasa real es desconocida, ya que nos hemos limitado a utilizar una (pequeña) muestra de datos para estimar su valor: el conjunto de prueba.

Conjuntos de entrenamiento, validación y prueba

Al no utilizar todos los datos disponibles para construir el modelo, puede que no sea todo lo bueno que podría llegar a ser si hubiésemos empleado el conjunto de datos completo en su construcción.

Entonces, **¿tenemos que reservar datos para evaluar el modelo pero el modelo sería mejor si no reservamos nada y usamos todos los datos a nuestra disposición?**

En la práctica, primero se realiza el experimento para estimar la calidad del modelo que podemos obtener. Una vez finalizada esa estimación, se desechan los modelos usados durante la fase de estimación y se construye un **modelo final**, que construimos a partir del **conjunto de datos completo**.

Conjuntos de entrenamiento, validación y prueba

Un **error muy común** es **reutilizar el mismo conjunto de prueba para múltiples experimentos** en los que se van variando los parámetros que admiten las distintas técnicas de aprendizaje.

¡El conjunto de prueba, deja de ser independiente!

Se podría decir que estamos entrenando modelos particulares con la ayuda de un conjunto de entrenamiento pero, y aquí viene lo grave, también **estamos entrenando los parámetros del algoritmo de aprendizaje utilizando el conjunto de prueba**, que debería ser siempre independiente si queremos utilizarlo como estimación no sesgada de la calidad de los modelos obtenidos.

Conjuntos de entrenamiento, validación y prueba

Los **hiperparámetros** del algoritmo de aprendizaje, obviamente tendremos que **ajustarlos**. Pero, para ello, debemos siempre emplear un **conjunto de datos independiente del conjunto de prueba**.

Para ello, el conjunto de datos disponible se divide en:

- **Conjunto de entrenamiento**: para construir/entrenar modelos.
- **Conjunto de validación**: para probar distintos parámetros del algoritmo de aprendizaje.
- **Conjunto de prueba**: para evaluar la calidad del modelo. Se mantiene aparte y usualmente se reporta la eficacia del modelo según los resultados en este conjunto.

Métodos de evaluación

La **estimación del error** mediante un conjunto de prueba independiente se puede hacer más fiable si **repetimos el proceso** con diferentes conjuntos de prueba.

En cada prueba, podríamos seleccionar aleatoriamente el conjunto de datos de entrenamiento y evaluar las tasas de error (o cualquier otra medida de interés) **promediando los resultados** obtenidos.

Inconvenientes de este proceso:
- Los **conjuntos** de entrenamiento y de prueba **se solaparían** en los distintos experimentos.
- Existiría la posibilidad de que **algunos ejemplos no se usasen nunca** como parte del conjunto de entrenamiento de un modelo.

Para solventar ambas limitaciones, lo habitual es recurrir a la **validación cruzada**.

Muestreo sin reemplazo: Validación cruzada

La validación cruzada **crea y evalúa múltiples modelos de forma sistemática**, cada uno de ellos sobre diferentes subconjuntos del conjunto de entrenamiento.

La **validación cruzada de k iteraciones** o **k-CV [*k-fold Cross-Validation*]** se realiza de la siguiente manera:

- En primer lugar, se divide aleatoriamente el conjunto de entrenamiento en **k subconjuntos disjuntos** del mismo tamaño.
- A continuación, **para cada uno** de los k subconjuntos **se construye un modelo** utilizando dicho subconjunto como conjunto de validación y empleando los $k-1$ restantes como conjunto de entrenamiento.

Es habitual usar $k=5$ o $k=10$.

Muestreo sin reemplazo: Validación cruzada

Al usar validación cruzada, en la **iteración** i, obtendremos una **medida** $\hat{m}_i$ **de la calidad del modelo** obtenido, evaluada siempre sobre el conjunto de validación correspondiente.

Finalmente, realizamos una **estimación** $\hat{m}_{cv}$ **de la calidad del modelo promediando** las medidas individuales:

$$\hat{m}_{cv} = \frac{1}{k} \sum_{i=1}^{k} \hat{m}_i$$

Muestreo sin reemplazo: Validación cruzada

Existen distintas **variantes de la validación cruzada**. Algunas de las más conocidas son:

- *"Leave one out"*: se realiza una validación cruzada con k particiones del conjunto de datos, donde k coincide con el número de ejemplos disponibles. Se utiliza con conjuntos de datos pequeños.
- *"Leave p out"*: como la anterior, pero el tamaño de los conjuntos de validación es p.
- **Validación cruzada estratificada:** las particiones se realizan intentando mantener en todas ellas la misma proporción de clases que aparece en el conjunto de datos completo.

Ejemplo

Supongamos un problema de clasificación para el que tenemos un conjunto de datos con 10000 registros. Realizamos una **separación** 80-20 para los conjuntos de **entrenamiento (train)** y conjunto de **prueba (test)**, con lo que tenemos:

x_train con 8000 registros para entrenar
y_train con las etiquetas de x_train
x_test con 2000 registros para test
y_test con las etiquetas de x_test

Entrenamos nuestro modelo con los 8000 registros del conjunto de entrenamiento y obtenemos un *accuracy* del 85% sobre dicho conjunto (y asumimos que ese porcentaje nos parece bien).

Ejemplo

Los 2000 registros del conjunto de prueba todavía no han pasado por el modelo. Realizamos **predicción** sobre esos registros y **obtenemos su *accuracy*** haciendo uso de y_test (en este proceso el modelo no está entrenando ni recibiendo conocimiento, únicamente ve la entrada y nos devuelve una salida). Puede pasar:

- Que el ***accuracy* en test** sea **cercano al de entrenamiento**: nuestro modelo está generalizando bien y lo podemos dar por bueno (siempre y cuando estemos conformes con las métricas obtenidas).

- Que el ***accuracy* en test** sea **muy distinto al de entrenamiento**, tanto por encima como por debajo: es un indicador de que nuestro modelo no ha entrenado bien y no nos sirve.

Ejemplo

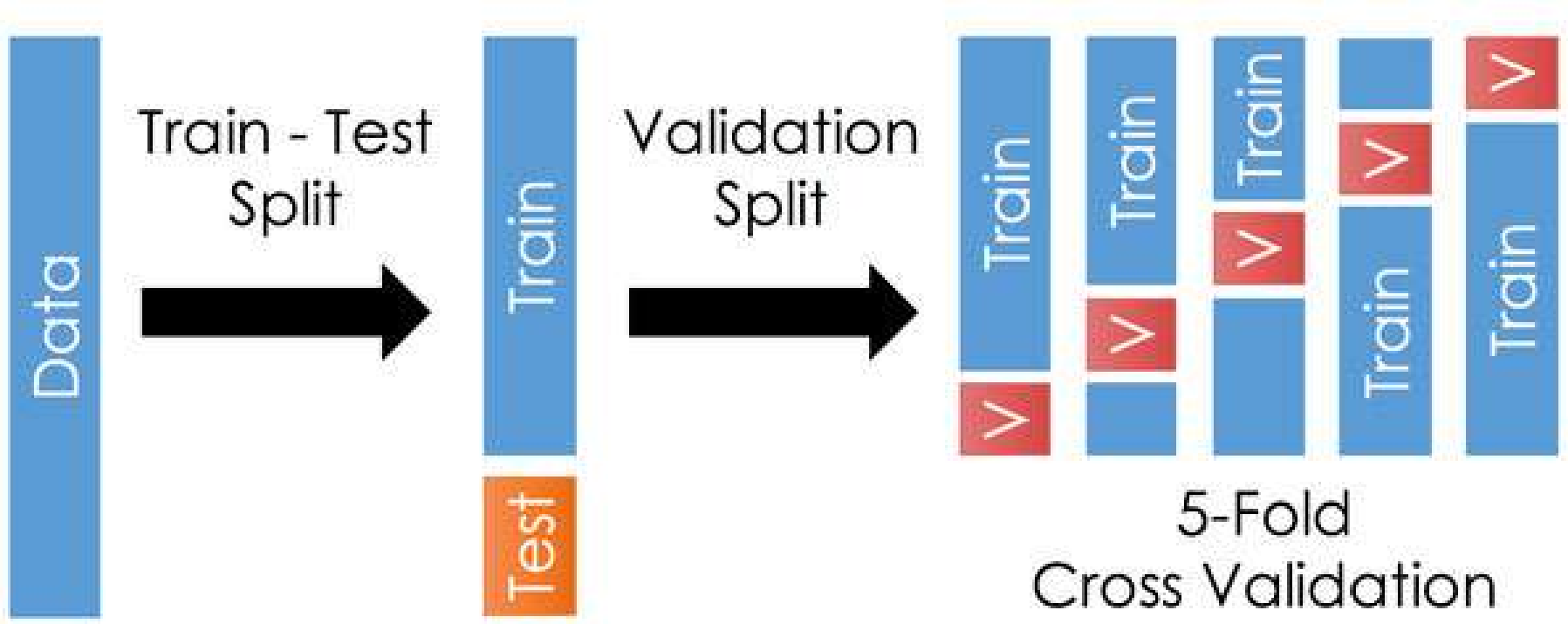
Si nos encontramos en el segundo caso, para mejorar el modelo podemos pensar en **ajustar sus hiperparámetros y volver a entrenar**.

Aquí es donde aparece el **conjunto de validación**. Hay que tener un cuenta que este no es realmente un tercer conjunto si no que **forma parte del conjunto de entrenamiento** y nos ayuda a obtener el mejor modelo de entre los distintos que probaremos, por ejemplo, haciendo uso de **validación cruzada**, que nos ayudará a medir el comportamiento de los modelos creados.

El **conjunto de prueba (test)** seguirá tratándose como antes: lo apartamos y lo usaremos al final, una vez entrenemos el modelo.

Ejemplo

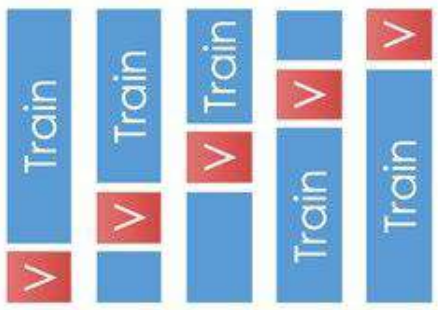
Recordemos que nuestro conjunto de entrenamiento estaba formado por 8000 registros. Si usamos validación cruzada, por ejemplo, con ***k*=5, iteramos 5 veces:**



Ejemplo

Recordemos que nuestro conjunto de entrenamiento estaba formado por 8000 registros. Si usamos validación cruzada, por ejemplo, con ***k*=5, iteramos 5 veces:**

- Apartamos 1/5 de la muestra, es decir 1600 registros.
- Entrenamos al modelo con el restante 4/5 (6400 registros).
- Medimos el *accuracy* sobre las 1600 que habíamos apartado.



Después de los 5 **entrenamientos independientes**, el *accuracy* final será el promedio de los 5 *accuracies* (que deberían ser similares).

Finalmente, se probará el modelo con el **conjunto de prueba**.

Ejemplo

Supongamos que el entrenamiento haciendo validación cruzada y predicción en el conjunto de prueba (test) nos ha dado buenos *accuracies* (y similares) y **damos por bueno nuestro modelo**.

El modelo seleccionado después de todas nuestras iteraciones, mejoras de modelo, ajuste de hiperparámetros y comparando con el conjunto de test, **deberíamos entrenarlo** haciendo uso de los 10000 registros que teníamos, es decir, agregamos el conjunto de prueba al modelo para aprovechar el **100% de nuestros datos**, y ese modelo final, será el que utilizaremos en la práctica para realizar predicciones.

Esta **última iteración** debería mejorar el modelo final, aunque es algo que no podemos contrastar, excepto con su comportamiento en el entorno real.

Muestreo con reemplazo: *Bootstrapping*

La **validación cruzada** utiliza **muestreo sin reemplazo**: el mismo ejemplo, una vez seleccionado, no puede volver a utilizarse. Por esto, no aparecen ejemplos duplicados en los conjuntos de entrenamiento y validación (más allá de los inevitables 'gemelos').

El ***bootstrapping*** es una técnica alternativa que recurre a técnicas de **muestreo con reemplazo**:

Se muestrea un conjunto de datos con n ejemplos para formar un nuevo conjunto de datos de n ejemplos. Al realizar el muestreo con reemplazo, algunos de los ejemplos aparecerán más de una vez en la muestra seleccionada. Esa muestra, del mismo tamaño que el conjunto de datos original, es la que se utiliza para entrenar un modelo.

Muestreo con reemplazo: *Bootstrapping*

¿Cómo se evalúa ese modelo? Recurriendo a los ejemplos del conjunto de datos original que no han sido incluidos en el conjunto de datos de entrenamiento.

Este método también se conoce como ***0.632-Bootstrap***:

Cuando se escoge una muestra, se escoge siempre con probabilidad $1/n$. Entonces, la probabilidad de que una muestra no sea escogida es $1-1/n$. Como se construye un conjunto de entrenamiento con n muestras, el proceso hay que repetirlo n veces. Por tanto, la probabilidad de que una muestra no sea escogida para formar parte del conjunto de entrenamiento será $(1-1/n)^n \approx e^{-1} = 0.368$. En otras palabras, **el 36.8% de las muestras no se escogerá nunca y servirán para evaluar el modelo**. El 63.2% restante formará parte del conjunto de entrenamiento, de ahí el nombre.

Muestreo con reemplazo: *Bootstrapping*

Si queremos una estimación más fiable, **podemos repetir el proceso varias veces**, seleccionando distintos conjuntos de muestras, y promediar los resultados obtenidos en las distintas ejecuciones.

Aunque no se utiliza tan a menudo como la validación cruzada, puede resultar útil si necesitamos estimar el rendimiento de un modelo cuando sólo disponemos de un **conjunto de datos muy pequeño**.

Si tenemos muy pocos datos disponibles para construir un modelo, *bootstrapping* nos ofrece un mecanismo para construir un conjunto de entrenamiento del mismo tamaño que nuestro conjunto de datos original y, además, evaluarlo con la ayuda de un conjunto de prueba independiente del conjunto de entrenamiento.

Algunos **consejos**:

Underfitting: si obtenemos un modelo con un error elevado en el conjunto de entrenamiento, similar al error sobre el conjunto de prueba, es probable que nuestro modelo tenga un **número insuficiente de parámetros** como para aprender correctamente todos los matices asociados a nuestro conjunto de datos. En este caso, intentar **ampliar el conjunto de características** utilizado puede que también nos ayude.

Overfitting: si obtenemos un modelo capaz de ajustar muy bien el conjunto de entrenamiento pero no funciona tan bien con el conjunto de prueba, tal vez tenga **demasiados parámetros ajustables**, y no disponga de los datos suficientes para poder estimar adecuadamente cuáles deberían ser sus valores correctos. **Aumentar** el número de casos de nuestro **conjunto de entrenamiento** o, si esto no es posible, **reducir el conjunto de características**, puede ser efectivo.

Bibliografía

•[1] Fernando Berzal. Redes Neuronales & Deep Learning. Edición independiente.

•[2] François Chollet. Deep learning with Python. Manning Shelter Island.

•[3] Ian Goodfellow, Yoshua Bengio & Aaron Courville. Deep Learning . MIT Press.