

Laboratorio de Datos

Entrenamiento y testeo

Primer Cuatrimestre 2024
Turnos tarde y noche

Facultad de Ciencias Exactas y Naturales, UBA

¿Cómo elegir entre distintos modelos?

Toda tarea de «aprendizaje automático», «machine learning» o «inteligencia artificial», consiste en:

- 1 Tomar un problema relevante del mundo material.
- 2 Elegir un modelo matemático que lo represente. El modelo en general va a depender de ciertos parámetros β_1, \dots, β_s . Por ejemplo en un modelo de regresión lineal, estos parámetros son los coeficientes de cada variable.
- 3 Definir una forma de medir qué tan bueno es un modelo, en relación a la realidad (vía los datos disponibles). Esto suele hacerse mediante una *función de pérdida*.
- 4 ...

¿Cómo elegir entre distintos modelos?

Función de pérdida: Si fijamos los parámetros de nuestro modelo β_1, \dots, β_s y aplicamos el modelo resultante a un conjunto de datos X , la función de pérdida mide cuánto “perdemos” utilizando el modelo en lugar de los datos reales.

En regresión lineal, la función de pérdida más común es el *error cuadrático medio* (*MSE*).

Vamos a llamar $L(\beta \mid X)$ a la función de pérdida para un conjunto de parámetros $\beta = (\beta_1, \dots, \beta_s)$ y un conjunto de datos X .

¿Cómo elegir entre distintos modelos?

Toda tarea de «aprendizaje automático», «machine learning» o «inteligencia artificial», consiste en:

- 1 Tomar un problema relevante del mundo material.
- 2 Elegir un modelo matemático que lo represente. El modelo en general va a depender de ciertos parámetros β_1, \dots, β_s . Por ejemplo en un modelo de regresión lineal, estos parámetros son los coeficientes de cada variable.
- 3 Definir una *función de pérdida* $L(\beta | X)$.
- 4 “Aprender” los coeficientes β_1, \dots, β_s . Es decir, encontrar valores β_1, \dots, β_s que minimicen la pérdida.

Nivel 1: Entrenar y evaluar sobre todo el conjunto de datos

Como primer acercamiento, entrenamos nuestros modelos con un conjunto de datos X y evaluamos la performance sobre *los mismos datos* X .

En este contexto, si un modelo M_1 es *más complejo* que un modelo M_0 (es decir tiene más variables o parámetros), entonces necesariamente el valor de la función de pérdida será menor o igual.

Nivel 1: Entrenar y evaluar sobre todo el conjunto de datos

Ejemplo:

- M_0 : gastos mensuales $\sim b_0 + b_1 \cdot \text{suelo} + b_2 \cdot \text{cantidad de hijos}$
- M_1 : gastos mensuales $\sim b_0 + b_1 \cdot \text{suelo} + b_2 \cdot \text{cantidad de hijos} + b_3 \cdot \text{altura}$

El modelo M_1 es más complejo que M_0 . Podemos ver a M_0 como un caso particular de M_1 tomando $b_3 = 0$.

El valor mínimo de pérdida de M_1 es menor o igual que el valor mínimo de pérdida de M_0 :

$$L_{M_1}(\beta_0^*, \beta_1^*, \beta_2^*, \beta_3^* \mid X) \leq L_{M_0}(\beta_0^*, \beta_1^*, \beta_2^* \mid X)$$

Nivel 1: Entrenar y evaluar sobre todo el conjunto de datos

En este esquema, un valor de pérdida menor no nos asegura que el modelo M_1 sea mejor que M_0 (tenga mejor capacidad predictiva).

Que un modelo alcance un error muy pequeño durante el entrenamiento, puede ser tanto mérito propio del modelo como un síntoma de una excesiva parametrización, que le permite “interpolarse” o “memorizar” los datos.

Ejemplo: Si tenemos n observaciones y tomamos un polinomio de grado $n - 1$ o un modelo con n variables (independientes), vamos a poder obtener un modelo exacto, la pérdida será 0.

Nivel 2: Separar en conjuntos de entrenamiento («train») y prueba («test»)

Para evitar el *sobreajuste* (overfitting) de los modelos, es habitual dividir el conjunto de datos en dos partes mutuamente excluyentes (y conjuntamente exhaustivas, ¡nada se tira!).

Entrenaremos cada modelo con los mismos datos de train para obtener los β óptimos, pero seleccionaremos como “mejor” a aquel modelo que minimice la pérdida sobre el conjunto de test.

Nivel 3: Split entrenamiento - validación - prueba

Como antes argumentamos que el modelo que minimiza el error de entrenamiento puede estar sobreajustándose a los datos, es igualmente posible que aquél que minimiza el error de prueba esté sobreajustándose a los datos de prueba: al fin y al cabo, así fue como definimos nuestra regla de selección (minimizar el error de prueba).

Nivel 3: Split entrenamiento - validación - prueba

Para evitar este problema, se suele dividir el conjunto de datos en tres partes: entrenamiento, validación y test:

- 1 Entrenamos los modelos minimizando L en X_{train} (los datos de entrenamiento).
- 2 Una vez entrenados los modelo, medimo la pérdida L en X_{val} y seleccionamos el modelo con menor pérdida.
- 3 Evaluamos su performance “en el mundo real” con L en X_{test} .

Nivel 4: Validación Cruzada en k Pliegos (“K-fold CV”)

En un esquema tripartito «train-val-test», el error de test sólo sirve para reporte, y achica el tamaño efectivo de la muestra.

Más aún, como hay un único conjunto de test, y todo el proceso está atravesado por ruido estocástico, la selección de modelos sigue teniendo un fuerte componente de azar.

Para evitar posibles sesgos en la selección de los conjuntos de entranamiento y validación, resulta conveniente realizar varias repeticiones del experimento con distintos conjuntos de datos. ¿Pero de dónde sacamos los datos para ello?

¡Pues los reutilizamos!

Nivel 4: Validación Cruzada en k Pliegos (“K-fold CV”)

En validación cruzada de k pliegos (“k-fold cross-validation”), dividimos primero el conjunto de datos sólo en train y test.

Luego, partimos train en k partes iguales, que se rotarán el papel de validación: entrenamos y evaluamos el modelo k veces, cada vez dejando uno distinto de los k pliegos como *val* y el resto para *train*.

