Tema 2: Regularización y Selección de Modelos

Aprendizaje automático

Héctor Lacueva Sacristán

Índice

Generalización

Es la capacidad de predecir la salida para nuevos datos.

En el apartado 1.2.3 del libro Probabilistic Machine Learning se explica mejor el concepto y sus consecuencias.

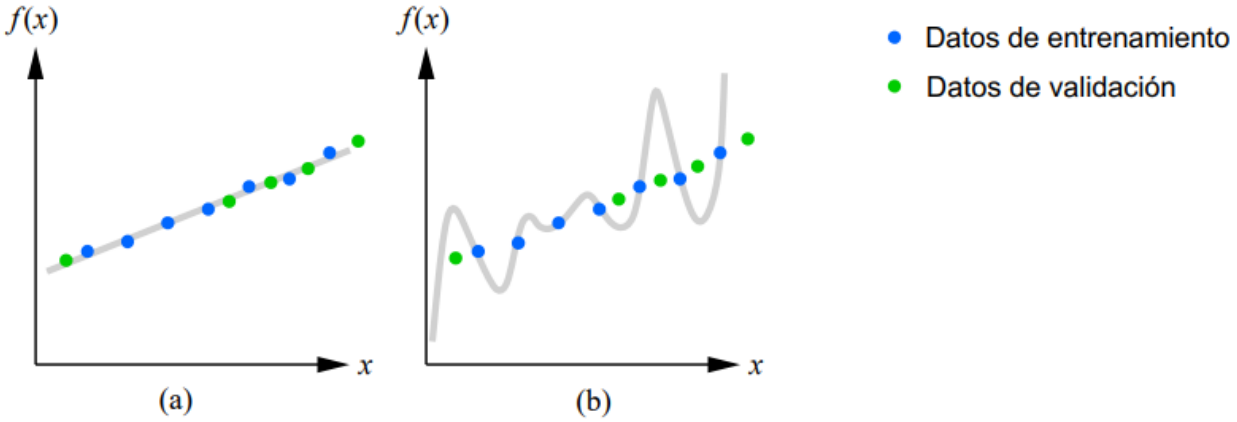


Figure 1: Ejemplo de ajuste polinómico. En esta imagen se observa como ajustar un modelo simple a unos datos vs. un modelo complejo. Se puede ver que el modelo simple no ajusta perfectamente los datos de entrenamiento, pero hace un trabajo decente para nuevos datos. En el segundo caso, se ajusta perfectamente a los datos de entrenamiento pero hace un trabajo pésimo para los nuevos datos. Por tanto, el segundo modelo generaliza peor que el primero. IMPORTANTE[2]

Sobreajuste y Subajuste

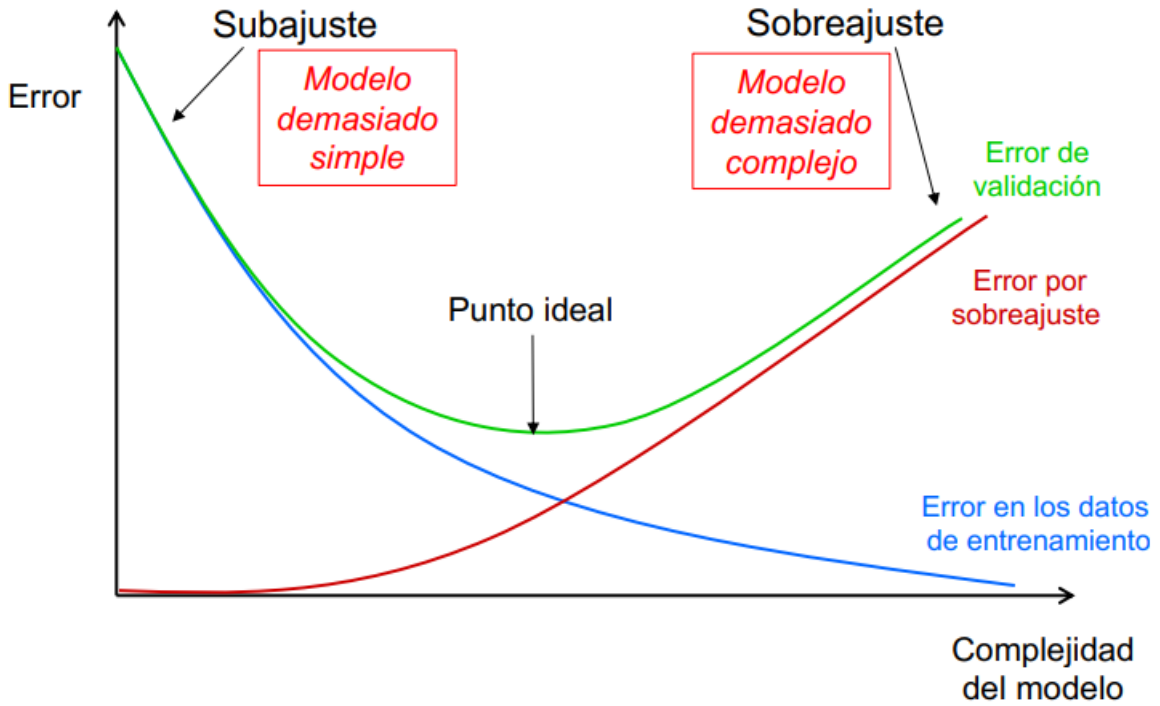


Figure 2: Representación gráfica del sobreajuste, subajuste y punto ideal

Definiciones

- Incluir definición de **Sesgo**.
- Incluir definición de **Varianza**.

[2]**Principio de la navaja de Occam**: En igualdad de condiciones, **elegir la hipótesis más simple**.

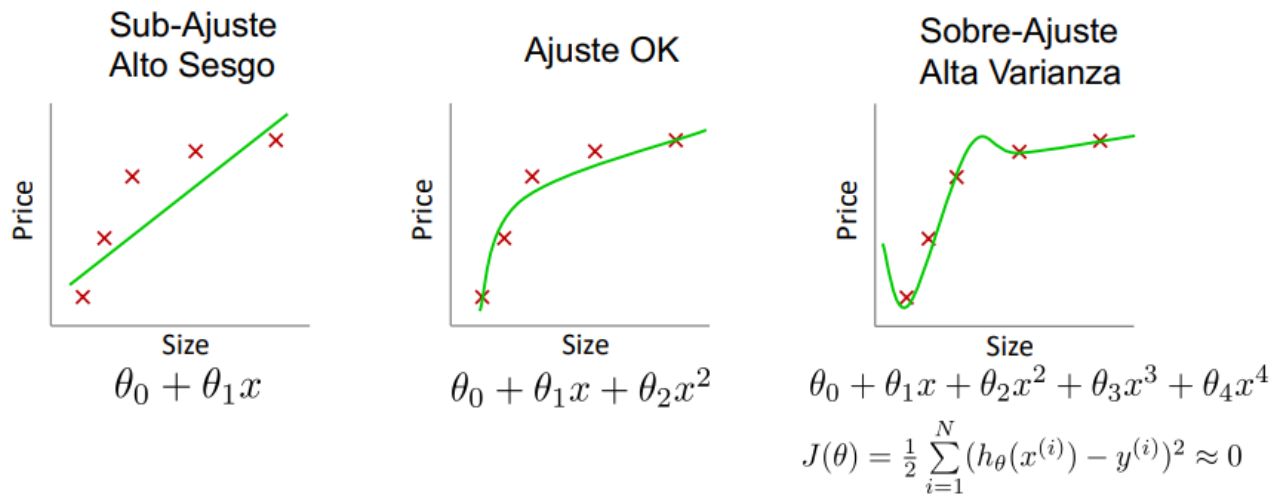


Figure 3: Ejemplo en **regresión**. Para una función muy simple se produce subajuste (por alto sesgo), para una función muy compleja se produce sobreajuste (por alta varianza, si la complejidad es la correcta el resultado será el deseado.)

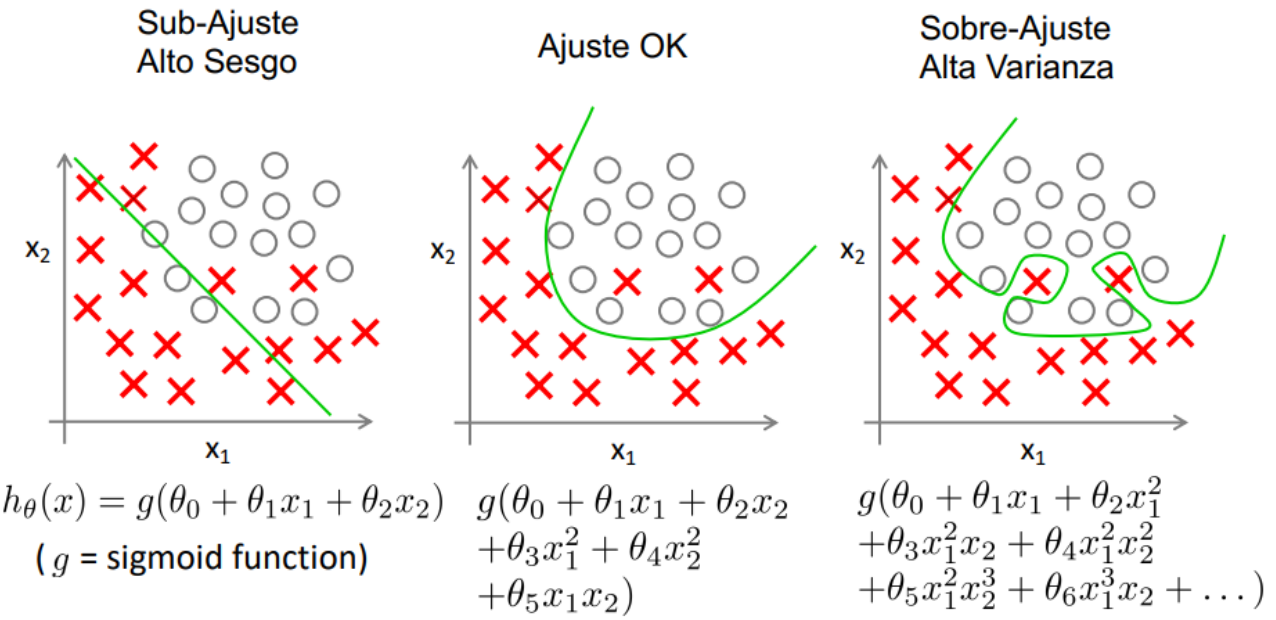


Figure 4: Ejemplo en **clasificación**. Para una función muy simple se produce subajuste (por alto sesgo), para una función muy compleja se produce sobreajuste (por alta varianza, si la complejidad es la correcta el resultado será el deseado.)

Sobreajuste

Si hay demasiados atributos, la hipótesis puede **ajustarse muy bien a los datos de entrenamiento**, pero **puede no generalizar bien a nuevos ejemplos**.

Como evita el Sobreajuste

Hay varias opciones:

Reducir el número de atributos

- Seleccionar manualmente los atributos a mantener.
- Selección de modelos (validación cruzada, puede ser K-Folding-validation).

Regularización:

- Mantener los atributos, pero reducir la magnitud de los pesos.
- Funciona bien si hay muchos atributos, y cada uno contribuye un poco a la predicción.

Evaluación de hipótesis

Como evaluar una hipótesis

Se evalua con datos de test distintos de los de entrenamiento.

Como evaluar varias hipótesis

Imaginemos que queremmos elegir entre estos modelos:

1. $h_\theta(x) = \theta_0 + \theta_1 x$
2. $h_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2$
3. $h_\theta(x) = \theta_0 + \theta_1 x + \cdots + \theta_3 x^3$
4. $h_\theta(x) = \theta_0 + \theta_1 x + \ldots + \theta_1 0 x^1 0$

Supongamos que el menor error se da para: $J_{test}(\theta^(5))$. Por tanto, elegimos el polinomio de orden 5: $\theta_0 + \cdots + \theta_5 x^5$.

El **problema** es que $J_{test}(\theta^(5))$ es una estimación optimista del error de generalización, porque el parámetro extra d[3] se ha ajustado con los datos de test.

Selección de modelos: Validación Cruzada

División de datos

- **Entrenamiento**: sirven para entrenar cada modelo.
- **Validación**: sirven para comparar modelos.
- **Test**: **bajo llave hasta el final**.

Proceso de aprendizaje

- **Aprender los parámetros** con los datos de **entrenamiento**.
- **Ajustar los hyper-parámetros** con los datos de **validación**.
- **SOLO UNA VEZ, AL FINAL**, calcular la precisión con los datos de **test**.

Es importante no espiar nunca los datos de test.

Datos de Validación y de Test:

- Datos de **Validación**
  - Misma distribución que los datos de entrenamiento (p.e. escoger 20% de los datos de entrenamiento).
  - los datos de validación se gastan (cambiar cada cierto tiempo).
  - Si hay pocos datos, y el entrenamiento no es muy costoso, usar K-fold.
- Datos de **Test**
  - Representativos de lo que esperamos encontrar en el futuro.

[3] d: grado del polinomio.

K-Fold Cross-Validation

Partir los datos en **k pliegues**. Cada dato entra en el conjunto de validación una vez. Valores típicos de $K \to 5, 10$.

Leave-one-out Cross Validation

Cuando **hay pocas muestras de entrenamiento N**, tomar **K=N**. En cada iteración:

- N-1 muestras para el entrenamiento.
- 1 muestra para la validación.

Errores

Habitualmente usaremos $RMSE$ ya que trabaja en las mismas unidades que el valor predicho y es independiente del número de muestras.

Los errores de entrenamiento y validación sirven para comprobar sobreajuste o subajuste. El error de validación sirve para seleccionar el mejor modelo y el de test sirve para la evaluación final del modelo elegido.

Como detecto el sobre-ajuste o sub-ajuste

Sub-ajuste

$$E_{train}(\theta) \text{ es alto y } E_{validation}(\theta) \approxeq E_{train}(\theta)$$

Sobre-ajuste

$$E_{train}(\theta) \text{ es bajo y } E_{validation}(\theta) > E_{train}(\theta)$$

Cómo encuentro el mejor modelo

Búsqueda exhaustiva (grid search)

- Probar todas las combinaciones posibles de los hiper-parámetros.
- Factible si son pocos y el entrenamiento es rápido.

$$\text{6 hyper-parámetros, 10 valores} \to \text{1.000.000 pruebas}$$

Búsqueda heurística

- Probar valores para el hyper-parámetro más importante y fijarlo
- Repetir con los hyper-parámetros siguientes

$$\text{6 hyper-parámetros, 10 valores} \to \text{60 pruebas}$$

Otras variaciones

- Refinamiento sucesivo: grid basta + grid fina.
- Heurística + grid para afinar.

  SIEMPRE SE DEBE ANOTAR EN UNA TABLA LOS RESULTADOS

Regularización

La idea es penalizar la complejidad del modelo con tal de reducir el sobre-ajuste.

Regulación L_2

Valores pequeños para los parámetros $\theta_1, \cdots, \theta_D$.

- Hipótesis "más simples"
- Menos propenso al sobre-ajuste

Penaliza más a los pesos que se vuelven más grandes en magnitud.

El peso θ_0 no se penaliza ya que solo afecta a la media global de la salida y no contribuye al sobre-ajuste.

Regresión Ridge

$$J(\theta) = \frac{1}{2}(\sum_{i=1}^{N}(h_\theta(x^{(i)} - y^{(i)})^2) + \lambda \sum_{j=1}^{D} \theta_j^2)$$

$$\widehat{\theta} = argmin_{theta} J(\theta)$$

Necesita datos escalados.

Valores mayores de lambda, disminuye el valor de los θ (sub-ajuste). Si uso valores pequeños de lambda puede no afectar a los valores de los pesos.

Se puede solucionar también por medio de **Descenso de Gradiente**.

Escoger el regularizador

Escoger un número finito de λ, usar cross validation y elegir el mejor. Si el problema es muy grande, usar warm start para mejorar la eficiencia.

Regularización L_1

Valores nulos para algunos parámetros $\theta_1, \cdots, \theta_D$.

- Selección automática de atributos importantes.
- **Proporciona modelos más simples**.

Regresión Lasso

$$J(w) = \frac{1}{2}\sum_{i=1}^{N}(w^T x^{(i)} - y^{(i)})^2 + \lambda \sum_{j=1}^{D} |w_j|$$

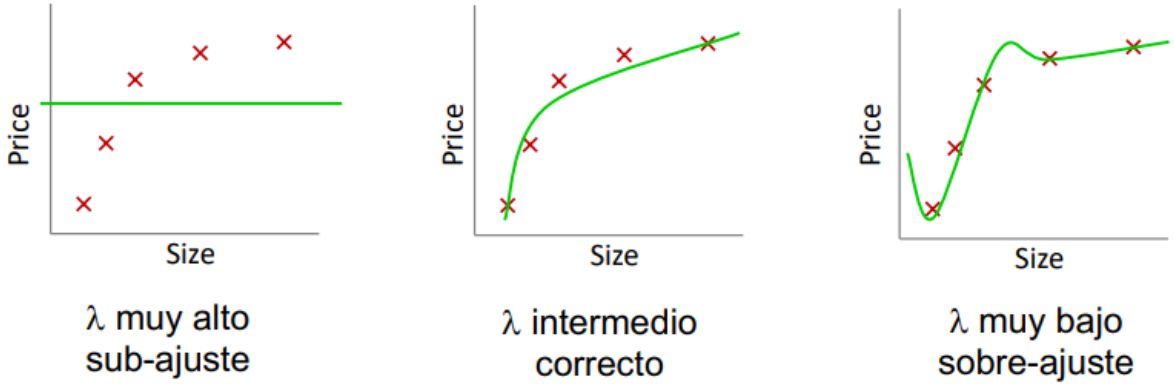
$$\widehat{w} = argmin_w J(w)$$

Necesita datos escalados.

Valores mayores de lambda, vuelve nulos los pesos w (sub-ajuste). Si uso valores pequeños de lambda puede no afectar a los valores de los pesos o puede disminuir algunos y haciendo nulos otros.

Como ajustar el parámetro regularizador λ

Diferencia de ajuste de una regresión para diferentes valores de λ



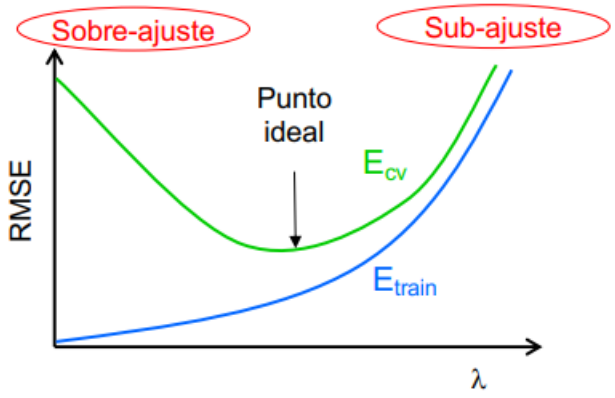
Se explica mejor en el apartado 4.5.4 del libro Probabilistic Machine Learning.

Validación cruzada

Probar valores de λ en progresión geométrica: $[0.0001, 0.001, 0.01, 0.1, \cdots]$

Se puede hacer un plot de una gráfica que relacione el $RMSE$ y el λ.

Ajuste de lambda



Ver numpy.logspace y matplotlib.pyplot.semilogx.

Regularización en RN

No parece que haya sobre-ajuste, o no se manifiesta.

Regularización L1 y L2

Dropout

Apagado aleatorio de neuronas.

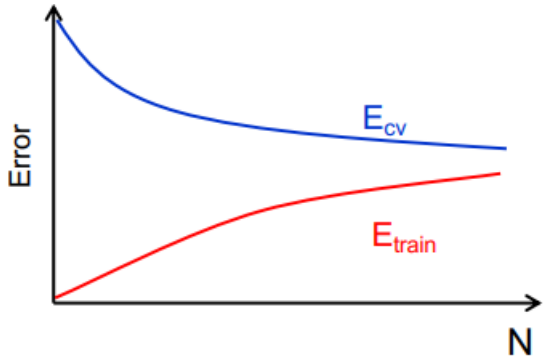
Double descent (no se comprende muy bien)

Muchas veces no hace falta.

Curvas de aprendizaje

Al aumentar las muestras de entrenamiento, la curva de aprendizaje converge.

Curva de aprendizaje



Usar **muchos datos**:

- Puede reducir el sobre-ajuste.
- Si hay sub-ajuste, no mejora.

¿Cómo depurar el aprendizaje?

Supongamos que hemos resuelto la regresión lineal regularizada. Pero probamos con datos nuevos y comete errores inaceptables. **¿Qué hacemos a continuación?**:

Si hay **sobre-ajuste**:

- Más muestras de entrenamiento.
- Usar menos atributos.
- Aumentar λ.

Si hay **sub-ajuste**:

- Conseguir atributos nuevos.
- Atributos polinómicos y otros.
- Disminuir λ.

Siempre que se tome una decisión se debe de escribir.