Introducción al Aprendizaje Automático

Aprenentatge Automàtic

APA/GEI/FIB/UPC - 2025/2026 1Q

/ Javier Béjar

Introducción

- Para resolver una tarea con una computadora, generalmente escribimos un algoritmo especificando todos los pasos para resolverlo
- Cuanto más difícil es la tarea, menos factible es definir el algoritmo
- Hay tareas que son tan difíciles que es imposible hacerlo bien
- Nuestra habilidad para codificar tareas complejas es limitada

- un robot que pueda caminar
- un sistema que identifique una cara en una fotografía
- un programa que siga un objeto en un video
- un traductor de un idioma a otro idioma
- un programa que prediga el precio de una acción
- un recomendador de películas/libros/canciones

- Para estas tareas, podemos recurrir al uso de datos en lugar de nuestra comprensión y conocimiento del problema
- Podemos:
  - recopilar cientos/miles/millones de ejemplos de la tarea
  - definir un algoritmo capaz de procesar esos datos
  - dejarle construir un modelo para la tarea por sí mismo,
- El modelo no será perfecto, pero será una aproximación útil que podremos aplicar

- No podemos escribir un programa capaz de reconocer una cara
- Sí podemos recopilar muchas imágenes y decirle a una computadora dónde está la cara en la imagen
- No podemos saber qué hace que un cliente esté más ansioso por comprar un producto
- Sí podemos reunir millones de registros de compras que un algoritmo puede explorar para encontrar patrones

- El Aprendizaje Automático es el área de la inteligencia artificial que estudia algoritmos que permiten que un agente se adapte a su entorno
- El Aprendizaje Automático trata de obtener modelos
  - para ayudar a agentes a ir más allá de su programación inicial
  - para resolver automáticamente tareas complejas
  - para hacer predicciones sobre nuevos problemas
  - para descubrir nuevos patrones que podrían ser útiles
- El objetivo de los algoritmos de aprendizaje automático es usar experiencias (datos) para generar modelos capaces de generalizar bien a nuevas experiencias

- Aprendizaje supervisado
- Aprendizaje no supervisado
- Aprendizaje por refuerzo

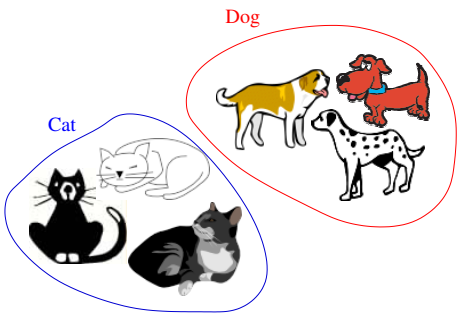
- La tarea a aprender es conocida
- La tarea es descrita por:
  - Un conjunto de ejemplos de entrada generalmente representados en forma tabular
  - Un valor de salida asociado a cada entrada (el valor esperado)
- **Objetivo:** Obtener un modelo para predecir la salida que corresponde a la entrada
- El modelo generaliza los datos (aprende, no memoriza), por lo que también puede predecir ejemplos que no ha visto
- Dos tareas diferenciadas:
  - Clasificación: La salida a predecir es un conjunto de valores discretos (etiquetas)
  - Regresión: La salida a predecir es un valor continuo

- La tarea a aprender es desconocida (descubrimiento)
- La tarea es descrita por:
  - Un conjunto de ejemplos de entrada generalmente representados en forma tabular
- Varias metas/tareas posibles:
  - Descubrir patrones en los datos que representan diferentes grupos (clustering)
  - Descubrir asociación entre los atributos de los datos (asociación/causalidad/dependencia)
  - Obtener transformaciones que simplifican la representación de los datos (reducción de dimensionalidad)

- Especializado en tareas de resolución de problemas
- Un agente tiene un conjunto de acciones disponibles para resolver una tarea compleja en un entorno
- El agente explora el entorno utilizando las acciones
- El entorno retroalimenta al agente en función de su desempeño (refuerzo)
- Aprendemos a asociar un valor a las acciones para un estado del entorno según su ventaja para alcanzar el objetivo de la tarea
- Después de aprender el modelo, se utiliza para tomar decisiones en el entorno

Aprendizaje supervisado

- **Objetivo:** Descubrir conceptos generales a partir de un conjunto limitado de ejemplos (experiencia)
- Se basa en la búsqueda de características similares entre ejemplos (patrones comunes)
- Todos sus métodos se basan en razonamiento inductivo: Genera conocimiento general a partir de información específica



- **Datos:** Experiencia que utilizamos para realizar el aprendizaje, una muestra aleatoria recolectada del problema que queremos resolver, descrita por un conjunto de atributos y su respuesta asociada

- **Modelos:** Descripción, utilizando un lenguaje específico, de cómo se generan o se comportan los datos de manera general, por ejemplo, fórmulas lógicas, funciones matemáticas o distribuciones de probabilidad. Un modelo tiene parámetros.

- **Aprendizaje:** El proceso que decide cómo se deben ajustar las características específicas de un modelo para generalizar los datos. Encontrar los parámetros.

- Asumiremos que los datos se representan como tablas
- Las filas son ejemplos
- Las columnas son los atributos (características) que describen los ejemplos
- Los atributos pueden ser numéricos o categóricos
- Una de las columnas corresponde a la respuesta supervisada (numérica o etiqueta)
- Consideraremos cada ejemplo como un vector D-dimensional, un conjunto de datos estará compuesto por un conjunto de pares (ejemplo, respuesta)

$$\mathcal{X} = \{(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)\}$$

- También podemos referirnos a ella como la **matriz de datos** $\mathcal{X} \in \mathbb{R}^{N \times D}$

1. Codificación de atributos (discretización, recodificación, binarización)
2. Detección de valores atípicos (*outliers*)
3. Valores faltantes *(missing values)* (imputación)
4. Normalización (rango, distribución)
5. Selección de características (*feature selection*)
6. Extracción de características (ingeniería de características, *feature engineering*)
7. Reducción de dimensionalidad y transformación
   - Maldición de la dimensionalidad = Todo es similar en espacios con alta dimensionalidad

Más sobre esto en el primera sesión de laboratorio

- Un modelos define:
  - Cómo se aborda el aprendizaje
  - Cuál será la descripción de la solución
  - Qué/Cómo se puede inferir del modelo
- Dos grupos de modelos:
  - Funciones (determinista)
  - Distribuciones de probabilidad (probabilista)

- El resultado del aprendizaje es una función predictiva/discriminativa
- La función asigna vectores (ejemplos) a un valor real (regresión) o a una etiqueta (clasificación)

$$f : \mathbb{R}^D \to \mathbb{R} \;\; / \;\; f : \mathbb{R}^D \to \{C_1, C_2, \ldots, C_L\}$$

- Por ejemplo, podemos usar funciones lineales:

$$f(x) = w^\top x + w_0$$

donde w y w_0 son los parámetros que debemos aprender usando los datos

- Los datos son observaciones ruidosas de un proceso y los modelos deben representar el efecto del ruido (incertidumbre)
- Modelamos una distribución de funciones posibles
- Un modelo se describe como una distribución de probabilidad multivariante
- Podemos usar dos enfoques diferentes:
  - **Modelos discriminativos:** Distribuciones de probabilidad que modelan la respuesta condicionada a los datos observados $p(y|x)$
  - **Modelos generativos**: Distribuciones de probabilidad que modelan la distribución conjunta de la respuesta y los datos $p(y, x)$

- El aprendizaje es el proceso que permite encontrar para un conjunto de datos los mejores parámetros para un modelo y una estimación de su rendimiento
- Hay tres procesos/fases involucradas en el aprendizaje:
  1. Entrenamiento o estimación de parámetros
  2. Ajuste de hiperparámetros o selección de modelo
  3. Predicción o inferencia

- Durante esta fase se ajustan los parámetros del modelo con el conjunto de datos (datos de entrenamiento)
- Aplicamos diferentes metodologías dependiendo del tipo de modelo
  - **Funciones:**
    - Minimización del riesgo empírico (Empirical Risk Minimization)
  - **Distribuciones de probabilidad:**
    - Máxima verosimilitud (Maximum Likelihood Estimation, MLE)
    - Máximo a posteriori (Maximum a Posteriori, MAP)
    - Inferencia Bayesiana, Inferencia Variacional...

- No solo nos interesa el mejor rendimiento en los datos de entrenamiento sino en datos no vistos (generalización)
- Dado que no sabemos qué datos futuros existen, debemos aplicar métodos de muestreo para simular el rendimiento con datos no vistos
- Se construyen varios modelos utilizando parte de los datos y se promedia el rendimiento para obtener su estimación
  - Validación Cruzada, Leave-one-out, Bootstrapping

- La complejidad elegida para el modelo puede conducir a
  - Subajuste (underfitting), el modelo es poco expresivo para la complejidad de los datos, alto error de entrenamiento
  - Sobreajuste (overfitting), el modelo es demasiado expresivo y captura la señal y el ruido, el rendimiento con nuevos datos es peor que para los datos de entrenamiento (memoriza)
- Dada suficiente capacidad en el modelo, la optimización de los parámetros tenderá a sobre ajustarse a los datos de entrenamiento

- A veces, un modelo tiene características que deben ser ajustadas antes del entrenamiento que cambian su estructura o complejidad
- Estas características se denominan hiperparámetros, por ejemplo
  - Podemos elegir entre diferentes distribuciones de probabilidad para nuestro modelo ¿cuál es la mejor?
  - Una distribución de probabilidad compleja que se compone de un número de distribuciones de probabilidad simples ¿cuántas usar?
  - Una regresión con un polinomio ¿cuántos términos usar?
- El problema de elegir entre diferentes modelos se llama selección de modelo

Aprendizaje funciones

- Aplicaremos **Minimización del riesgo empírico**
- Para aprender necesitamos decidir cuatro elementos:
  1. ¿Cuál es el conjunto de funciones que permitiremos para el predictor? (espacio de hipótesis)
  2. ¿Cómo medir el rendimiento del predictor en los datos de entrenamiento? (función de pérdida)
  3. ¿Cómo aprendemos predictores que también funcionan bien con datos no vistos? (regularización)
  4. ¿Cómo buscamos en el espacio de funciones posibles? (ajuste/selección de modelos)

- Asumimos que tenemos N ejemplos $x_n \in \mathbb{R}^D$ y su respuesta correspondiente $y_n \in \mathbb{R}$
- Queremos estimar una función parametrizada (predictor)

$$f(\cdot, w) : \mathbb{R}^D \to \mathbb{R}$$

siendo w los parámetros de la función

- Una vez que ajustamos los parámetros de la función (w^*) esperamos que

$$\hat{y}_n = f(x_n, w^*) \approx y_n \quad \forall n = 1, \ldots, N$$

- Decidimos aproximar funciones de una dimensión usando polinomios
- La familia de funciones que queremos usar tiene la forma:

$$f(x, w) = \sum_{i=0}^{M} w_i x^i$$

- Los ejemplos están compuestos por vectores de la forma:

$$x_n = \{x_n^0, x_n^1, \ldots, x_n^M\}$$

- Queremos optimizar los parámetros $w = \{w_0, w_1, \ldots, w_M\}$
- M es un hiperparámetro, grado máximo del polinomio a ajustar

- Debemos definir cómo medir el error que corresponde a una parametrización específica del predictor
- La **función de pérdida** $\ell(y_n, \hat{y}_n)$ toma las predicciones y los valores reales para calcular un valor no negativo (pérdida) que mide qué tan lejos están
- El objetivo es encontrar un vector de parámetros w^* que minimice la pérdida promedio sobre los N ejemplos
- Suponiendo que los ejemplos son independientes e idénticamente distribuidos (iid), la media empírica de la pérdida es una buena estimación de la pérdida de la población

- Dados los datos de entrenamiento X y sus salidas y podemos definir el riesgo empírico (función de coste) como:

$$R_{emp}(f, X, y) = \frac{1}{N} \sum_{i=1}^{N} \ell(y_i, \hat{y}_i)$$

donde $\hat{y}_i = f(x_n, w)$

- Minimizar esta función para encontrar el parámetro vectorial w^* se denomina Minimización del riesgo empírico

- Para funciones complejas, obtener los parámetros con la mínima pérdida empírica puede sobreespecializar
- Queremos un predictor que también prediga bien los datos nuevos (generalice)
- La minimización del riesgo empírico introduce el concepto de **regularización**

- La regularización es un término adicional agregado a la función de pérdida penalizando la complejidad
- Buscamos que la función aprendida sea tan suave como sea posible entre los datos de entrenamiento
- La búsqueda está sesgada para que los mejores parámetros correspondan a un compromiso entre el error y la complejidad
- Este término se pondera, agregando un nuevo hiperparámetro para ajustar la selección del modelo

$$R_{emp}(f, X, y) = \frac{1}{N} \sum_{n=1}^{N} \ell(y_n, \hat{y}_n) + \lambda \cdot Compl(f)$$

- Para aproximar el error de generalización de un modelo necesitamos datos que no se hayan utilizado para el ajuste de parámetros
- Debemos reservar una parte del conjunto de datos (conjunto de validación)
- Cuando el conjunto de datos es pequeño, la muestra validación es demasiado pequeña, lo que resulta en una estimación ruidosa del error (varianza alta)
- El enfoque para obtener una estimación del error esperado es usar una técnica de remuestreo
- La más habitual es la validación cruzada de k-particiones (**k-fold cross validation**)

- La validación cruzada es una estrategia de estimación por remuestreo
  - Los datos se dividen en K particiones independientes
  - K-1 particiones se usan como conjunto de entrenamiento ($\mathcal{R}$), la petición restante es el conjunto de validación ($\mathcal{V}$)
  - El entrenamiento se repite K veces con todas las combinaciones de particiones como conjunto de entrenamiento y validación
- El valor de K puede oscilar entre 2 y N, un valor común es $K = 10$
- Con $K = N$ tenemos Leave One Out Cross Validation (LOOCV)

- El número de particiones corresponde a un compromiso entre el sesgo y la varianza del error
  - Cuanto menor sea, mayor será el sesgo en las estimaciones de error y menos varianza
  - Cuando $K = N$, la estimación del error tiene un sesgo muy bajo, pero tiene la posibilidad de una varianza alta
- Cuando el tamaño del conjunto de datos (N) es grande, podemos reducir K
- La principal desventaja es que tenemos que entrenar múltiples modelos, pero es una tarea que se puede paralelizar

- Para unos hiperparámetros del modelo, la validación cruzada permite calcular los parámetros con mejor el error de generalización
- Para calcular el modelo final necesitamos datos adicionales no usados en el entrenamiento
- Los datos deben dividirse inicialmente en un conjunto de entrenamiento y un conjunto de prueba/test que no usaremos hasta el final
- Los mejores modelos se reentrenan con el conjunto de entrenamiento inicial y el error final se calcula con el conjunto del test
- Un mucho menor error de validación cruzada que error de test indica sobre especialización

Queremos aprender los parámetros para el ajuste de modelo polinómico

$$f(x, w) = \sum_{i=0}^{M} w_i x^i$$

donde $x_n = \{x_n^0, x_n^1, x_n^2, \ldots, x_n^M\} = \{1, x_n, x_n^2, \ldots, x_n^M\}$ y $w = \{w_0, w_1, \ldots, w_M\}$

Usamos el error cuadrático como función de pérdida para encontrar los mejores parámetros, usando el riesgo empírico tendríamos:

$$R_{emp}(f, X, y) = \frac{1}{N} \sum_{n=1}^{N} \ell(y_n, \hat{y}_n) = \frac{1}{2N} \sum_{n=1}^{N} \left(y_n - \sum_{i=0}^{M} w_i x_n^i \right)^2$$

Necesitamos minimizar el error medio para encontrar los parámetros, quitando $1/N$ (no afecta al mínimo), calculando la derivada del error respecto a los parámetros w_j e igualando a 0 tenemos:

$$\frac{\partial R_{emp}}{\partial w_j} = \sum_{n=1}^{N} x_n^j \left(y_n - \sum_{i=0}^{M} w_i x_n^i \right) = 0 \;\; \forall j \in [0 \dots M]$$

Eso corresponde a un sistema de ecuaciones lineales ($N \times (M+1)$) y podemos obtener una solución analítica para este sistema (no hace falta optimización)

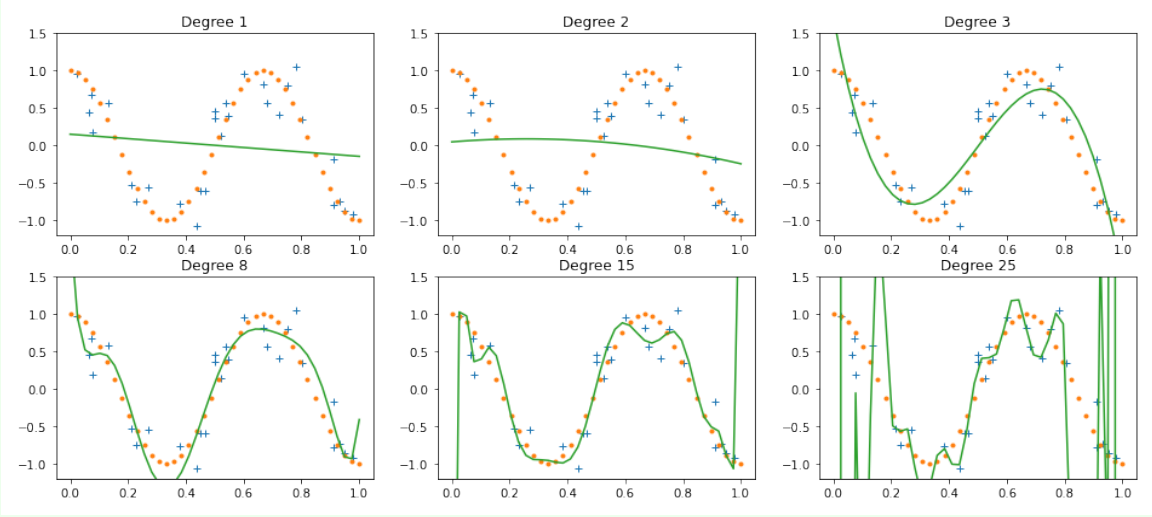
Necesitamos decidir un valor para M; cuanto mayor sea M, más complejo será el modelo. Cuando $N = M + 1$ el sistema tiene una solución única, que sobre ajustará los datos (error = 0), debemos usar regularización

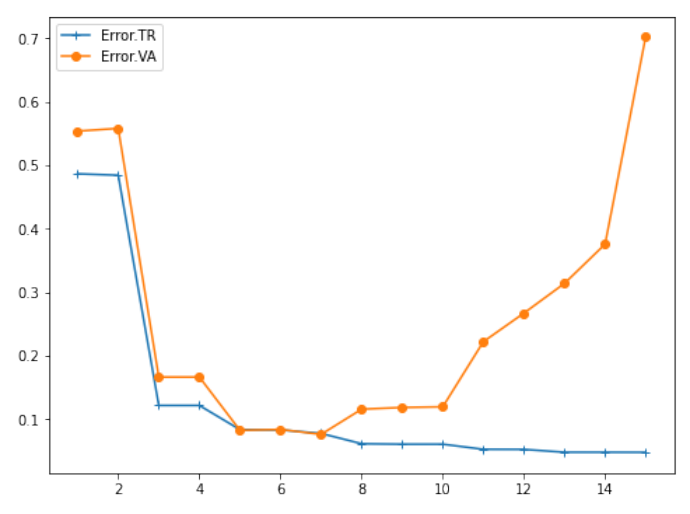
Para este caso, el problema de usar demasiados términos en el polinomio es que los parámetros tienen valores muy grandes haciendo que la función cambie mucho entre los datos observados. Sobre ajusta los datos y predice muy mal datos no vistos

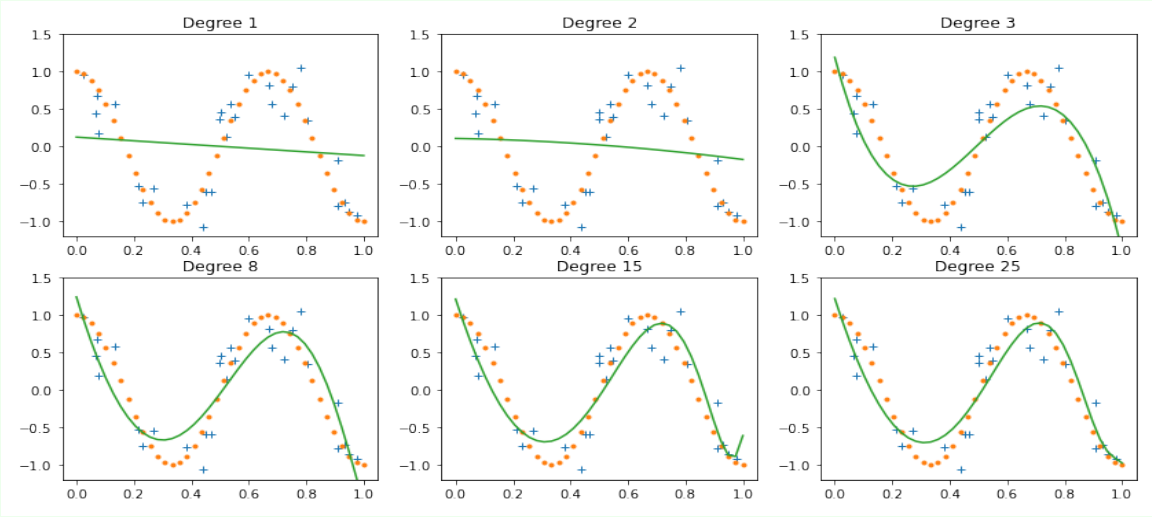
Podríamos penalizar parámetros con valores muy grandes, por ejemplo usando su suma o la suma de sus valores cuadrados, modificando el error empírico así:

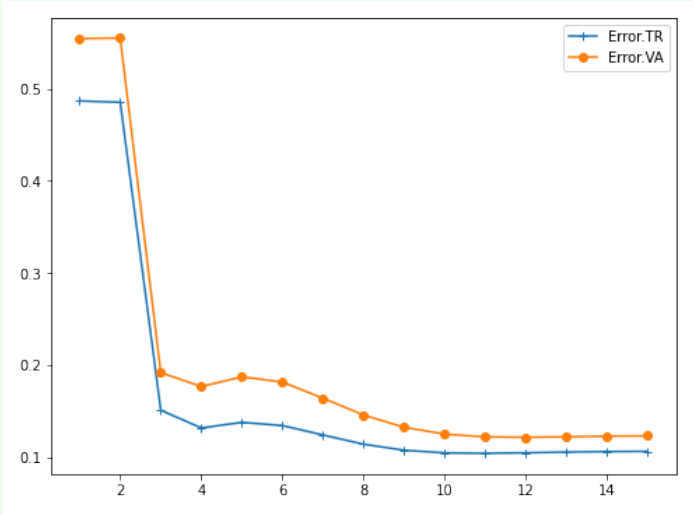
$$R_{emp}(f, X, y) = \frac{1}{N} \sum_{n=1}^{N} \left(y_n - \sum_{i=0}^{M} w_i x_n^i \right)^2 + \frac{\lambda}{2} \cdot \sum_{i=1}^{M} w_i^2$$

Podemos elegir el valor para λ usando validación cruzada


Degree 1
Degree 2
Degree 3
Degree 8
Degree 15
Degree 25


Error.TR
Error.VA
0.7
0.6
0.5
0.4
0.3
0.2
0.1
2
4
6
8
10
12
14


Degree 1
Degree 2
Degree 3
Degree 8
Degree 15
Degree 25

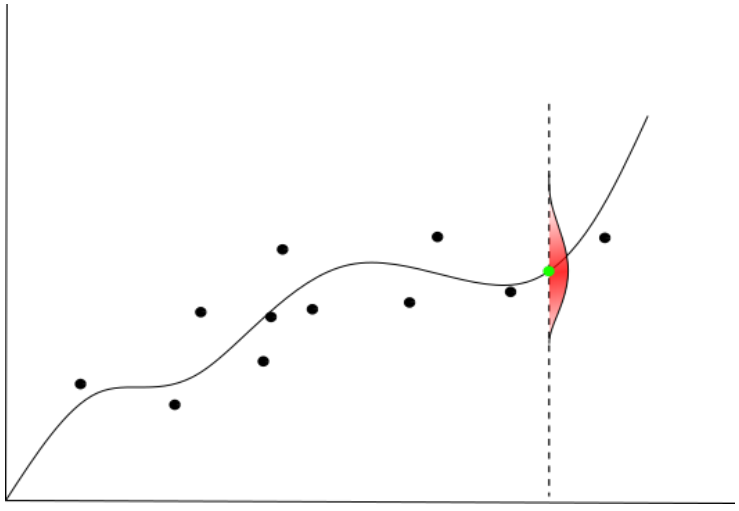

Error.TR
Error.VA
0.5
0.4
0.3
0.2
0.1
2
4
6
8
10
12
14

Podéis jugar con este ejemplo en este notebook

También podéis ver un video explicando el contenido del notebook

Aprender distribuciones de probabilidad

- Usando una función como modelo, ignoramos la incertidumbre de
  - Las medidas que corresponden a los ejemplos y sus respuestas
  - Los parámetros del modelo
- Podemos usar distribuciones de probabilidad para modelar estas incertidumbres
- Para estos modelos, la función de verosimilitud es la función de pérdida
- La regularización se define a partir de distribuciones a priori (*priors*)



- La Estimación de Máxima Verosimilitud (MLE) define una función de los parámetros que permite encontrar el mejor valor que se ajuste a los datos
- La estimación de parámetros se centra en la función de verosimilitud de una distribución de probabilidad
- Por razones numéricas (sus valores son demasiado pequeños) se utiliza la función logaritmo de la verosimilitud (log-likelihood)
- Por razones históricas se usa el negativo de esta función (generalmente minimizamos las funciones)

- Dado un conjunto de datos representado por una función de densidad de distribución de probabilidad $p(x|w)$, el negativo del logaritmo de la verosimilitud (*negative log likelihood*) es:

$$\mathcal{L}_x(w) = -\log p(x|w)$$

- La función de verosimilitud se puede leer de dos maneras:
  - Para un conjunto fijo de parámetros w, nos da la probabilidad de observar los datos x
  - Para datos observados (fijos) x, nos da qué tan probables son los diferentes valores para los parámetros w
- En aprendizaje supervisado definiremos un predictor que, dado un vector de ejemplo x_n y un conjunto de parámetros ajustados w, devuelva una distribución sobre la respuesta y_n

Supondremos que queremos obtener un predictor para el polinomio de regresión como en el ejemplo anterior, donde el vector de datos x_n es $\{x_n^0, x_n^1, x_n^2, \ldots, x_n^M\}$ y la respuesta corresponde a un valor real y_n más un error ϵ_n (la incertidumbre) que se modela como una distribución gaussiana $\mathcal{N}(0, \sigma^2)$ (σ^2 es sabido)

$$f(x_n, w) = w^\top x_n + \epsilon_n = y_n$$

Podemos definir una función de densidad de probabilidad como:

$$p(y_n|x_n, w) = \mathcal{N}(y_n|w^\top x_n, \sigma^2)$$

Dado un conjunto de datos de ejemplos iid $\{(x_1, y_1), \ldots, (x_N, y_N)\}$ donde $\mathcal{X}$ son los valores de entrada e $\mathcal{Y}$ son las respuestas, podemos definir la probabilidad del conjunto de datos como el producto de las probabilidades individuales (porque son independientes):

$$p(\mathcal{Y}|\mathcal{X}, w) = \prod_{n=1}^{N} p(y_n|x_n, w)$$

Si calculamos el negativo del logaritmo de la verosimilitud:

$$\mathcal{L}(w) = -\log p(\mathcal{Y}|\mathcal{X}, w) = -\sum_{n=1}^{N} \log p(y_n|x_n, w)$$

Sustituyendo la distribución gaussiana que estamos asumiendo:

$$\begin{aligned}
\mathcal{L}(w) &= -\sum_{n=1}^{N} \log \mathcal{N}(y_n | w^\top x_n, \sigma^2) \\
&= -\sum_{n=1}^{N} \log \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y_n - w^\top x_n)^2}{2\sigma^2}\right) \\
&= -\sum_{n=1}^{N} \log \exp\left(-\frac{(y_n - w^\top x_n)^2}{2\sigma^2}\right) - \sum_{n=1}^{N} \log \frac{1}{\sqrt{2\pi\sigma^2}} \\
&= \frac{1}{2\sigma^2} \sum_{n=1}^{N} (y_n - w^\top x_n)^2 - \sum_{n=1}^{N} \log \frac{1}{\sqrt{2\pi\sigma^2}}
\end{aligned}$$

Como σ es dado, minimizar la log-verosimilitud significa minimizar el primer término, que corresponde a minimizar la pérdida cuadrática (mínimos cuadrados)

Comprensión de los modelos ML

- El construir un modelo para hacer predicciones no es el último paso del proceso de aprendizaje automático
- Entender cómo se hacen las predicciones es importante porque:
  - Ayuda a comprender el proceso de generación de datos
  - Ayuda a comprender qué características son más importantes y cómo interactúan
  - Las predicciones se pueden explicar/justificar a los usuarios finales, por lo que el modelo es más confiable
  - ¡La GDPR[1] y el futuro reglamento de la IA de la UE (EU AI Act) lo dicen!

[1] General Data Protection Regulation

- La **Interpretabilidad** asume que el modelo se define en términos que son fáciles de entender (modelos de caja blanca)
  - El modelo se basa en los atributos originales usando combinaciones lineales
  - Cada atributo tiene asignado un peso o un umbral
  - Los atributos/características adicionales (ingeniería de características) son combinaciones simples de las características originales

- La **Explicabilidad** (XAI) asume que el modelo es demasiado complejo para ser interpretado (modelos de caja negra)
  - El modelo se basa en combinaciones no lineales de las características originales
  - Cada atributo tiene múltiples pesos con múltiples combinaciones con otros atributos
  - Las características adicionales se obtienen aplicando no linealidades complejas o son aprendidas directamente por el modelo

- **Alcance:**
  - Modelo Global, cómo se comporta el modelo para cualquier ejemplo
  - Modelo Local, cómo se comporta alrededor de ejemplos específicos, generalmente usando un modelo sustituto/subrogado (aproximación lineal)
  - Justificación de ejemplos, por qué un ejemplo tiene una predicción específica

- **Tipo:**
  - Importancia/prominencia de las características, qué atributos tienen más importancia para el comportamiento del modelo o la predicción de un ejemplo
  - Dependencia de las características, cómo los atributos influyen/interactúan/se relacionan entre sí
  - Basada en ejemplos, contrafactuales (si eso no sucede, entonces eso no sucede), prototipos (tendencia central), ejemplos adversarios (este ejemplo es similar, pero se predijo incorrectamente), ejemplos influyentes (eliminar el ejemplo cambia drásticamente el modelo)