Aprendizaje Automático

Resumen

Esta asignatura se basa en la formación de paradigmas avanzados de aprendizaje automático y cómo funcionan por dentro las inteligencias artificiales. Se estudiarán algoritmos capaces de generalizar comportamientos y reconocer patrones a partir de información suministrada en forma de ejemplos. Las técnicas a emplear en cada una de las fases de un típico problema de aprendizaje automático son la formalización del problema, identificación de las variables relevantes, pre-procesamiento de datos, construcción de modelos, entrenamiento de los modelos y validación y estimación de la capacidad de generalización de los mismos ante nuevos datos.

Sandra Mingo Ramírez

UAM - 2024/25

11 de marzo de 2025 18:03

Universidad Autónoma de Madrid
Bioinformática y Biología Computacional

Código en Github

Índice general

Capítulo I

Introducción

I.1. Introducción al Aprendizaje Automático o Machine Learning

I.1.1. Contexto histórico

Hay muchas definiciones de aprendizaje automático. Según Wikipedia, machine learning es la construcción y estudio de sistemas que pueden aprender de datos. Arthur Samuel lo definía como un campo de estudio que confiere a los ordenadores la capacidad de aprender sin ser programados explícitamente. En el aprendizaje automático, no se diseña el algoritmo para que resuelva una tarea con unas reglas fijas, si no para que con una serie de datos pueda aprender a resolver la tarea.

Arthur Samuel, en la década de 1950, escribió un programa para jugar a las damas que era capaz de aprender las mejores posiciones del tablero analizando miles de partidas. El sistema aprendió por sí mismo a jugar a las damas cada vez mejor. El 11 de mayo de 1997, el gran maestro de ajedrez Garry Kasparov renuncia tras 19 movimientos en una partida contra Deep Blue, un ordenador ajedrecista desarrollado por científicos de IBM. En 2016, Google (AlphaGo) derrotó al campeón mundial de Go. Este juego fue considerado durante décadas uno de los grandes retos de la IA. [1] En 2020, Google (AlphaFold) predice la estructura de las proteínas. Aquí se basa de **aprendizaje por refuerzo**. Se utilizó AlphaGo como base para generar otros modelos similares: AlphaChess, AlphaFold, etc. Las damas, el ajedrez, el go y las estructuras de las proteínas tienen en común ser problemas con unas reglas bien definidas. A partir de reglas sencillas, se generan estructuras complejas. Por ello, son campos donde se puede predecir o estimar muchas combinaciones y posibles variaciones. Estos algoritmos funcionan por prueba y error, por lo que no tiene sentido aplicarlo en otros campos donde los errores tienen consecuencias graves, como puede ser el diagnóstico de enfermedades o la conducción autónoma de coches. En general, todo el comportamiento humano es imposible de describir en reglas; cada paciente es muy

[1] Para el ajedrez, no se trata realmente de una inteligencia artificial, si no una máquina que calcula probabilidades. Cada movimiento proporciona una probabilidad de vencer al contrincante. Hay aperturas del ajedrez que facilitan un poco la victoria. Esto para el Go no existe. La máquina pudo encontrar una táctica para el Go nunca antes descrita, abriendo el debate de si se trata de creatividad.

complejo en sí mismo, siendo difícil generalizar en poblaciones grandes por procesos moleculares, comportamiento, epigenética, etc.

La IA se ha democratizado mucho con los softwares open-source. Tecnológicamente no hay secretos a día de hoy, solo diferencias en los datos y el hardware.

I.1.2. De programación clásica al aprendizaje automático

Los humanos adquieren con el tiempo experiencias que les hace aprender, causando respuestas concretas a distintas situaciones. Los ordenadores y las máquinas obtienen reglas predefinidas y datos, y con programación clásica llegan a su respuesta. No obstante, actualmente se utilizan datos y respuestas para, mediante aprendizaje automático, poder inferir las reglas. Esto invierte la forma de funcionar los ordenadores, y ha sido lo revolucionario del campo. Esas reglas inferidas se utilizarán para nuevos datos y poder ser cada vez más precisas. Así, el algoritmo encuentra las mejores reglas para resolver un problema. Estas reglas son ecuaciones matemáticas, pudiendo ser propensas a sesgos en base a los datos.

I.1.3. De comportamiento humano a comportamiento computacional: el modelo estándar

Una tarea humana se realiza a través de unos objetivos mediante abstracción. El proceso de aprendizaje está guiado por objetivos predefinidos (es decir, la simplificación de comportamientos complejos). En el caso de las máquinas, el proceso de aprendizaje consta de etapas de optimización para llegar al objetivo. Al final se trata de una reducción y optimización de la abstracción humana. No obstante, no hay una visión directa entre el comportamiento de la máquina y el comportamiento humano. No se puede esperar que un algoritmo sea justo o generoso por naturaleza si no se especifica en sus objetivos. Por ello, es muy fácil que aparezcan sesgos en el aprendizaje automático. La toma de decisión es muy diferente entre un humano y una máquina.

I.1.4. Cómo ve la IA: Un ejemplo con ataques adversarios

Tenemos una imagen de un cerdo (figura I.1), y no necesitamos el ruido para saber lo que es. Si le añadimos ruido a la imagen, aunque sea en una baja cantidad, la imagen no cambia para los humanos. No obstante, el sistema de reconocimiento de imágenes lo reconoce como una aerolínea. El ruido no es aleatorio, si no adversario. Esto quiere decir que la entrada al modelo ha sido modificada ligeramente de forma intencional, haciendo que el modelo genere una salida incorrecta. Se manipula para confundir a la máquina.

I.1.5. Cognición humana: sistema 1 vs sistema 2

Los conceptos del libro Thinking, Fast and Slow de Daniel Kahneman se han aplicado en el campo del aprendizaje automático. Se habla de dos sistemas y categorías de tareas cognitivas. El **sistema 1** es intuitivo, rápido, inconsciente, no lingüístico y

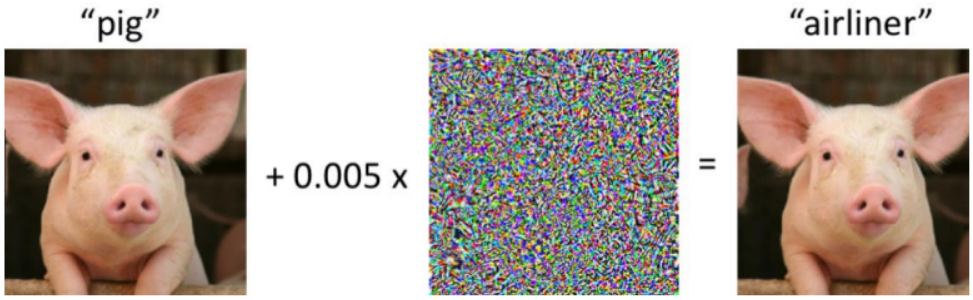


Figura I.1: *Reconocimiento de imágenes con ataque adversario.*

habitual. Se decía que el aprendizaje profundo estaba en ese sistema. El **sistema 2** es lento, lógico, secuencial, consciente, lingüístico, algorítmico, y es donde estaría el deep learning futuro. Esto sirvió para el aprendizaje automático de estructuras de datos: redes de cápsulas, aprendizaje automático neurosintáctico, razonamiento conceptual, bases de experiencia, reglas lógicas, etc. *El sistema 1 sirve para reconocer formas, colores y posiciones, mientras que el sistema 2 ayuda en la predicción de interacciones.*

I.1.6. Tarea de aprendizaje

Se dice que un programa informático aprende de la experiencia E con respecto a alguna tarea T y alguna medida de rendimiento P, si su rendimiento en T, medido por P, mejora con la experiencia E. Si el rendimiento es perfecto desde el principio, no hay aprendizaje, ya que requiere una optimización o mejora del estado. Ejemplos son:

- T: Jugar a las damas
  P: Porcentaje de partidas ganadas contra un contrincante arbitrario
  E: Jugar partidas de práctica contra uno mismo
- T: Reconocer palabras escritas a mano
  P: Porcentaje de palabras clasificadas correctamente
  E: Base de datos de imágenes de palabras manuscritas etiquetadas por humanos
- T: Conducción en autopistas de cuatro carriles mediante sensores de visión
  P: Distancia media recorrida antes de un error apreciado por el ser humano
  E: Secuencia de imágenes y comandos de dirección grabados mientras se observa a un conductor humano.
- T: clasificar los mensajes de correo electrónico como spam o legítimos.
  P: Porcentaje de mensajes de correo electrónico clasificados correctamente.
  E: Base de datos de correos electrónicos, algunos con etiquetas dadas por humanos.

I.1.7. Aprendizaje automático en contexto

En el núcleo de la IA, el aprendizaje automático es simplemente una forma de conseguir IA. En lugar de codificar rutinas de software con instrucciones específicas

para realizar una tarea concreta, el ML es una forma de «entrenar» un algoritmo para que aprenda a hacerlo. El «entrenamiento» consiste en introducir grandes cantidades de datos en el algoritmo y permitir que éste se ajuste y mejore.

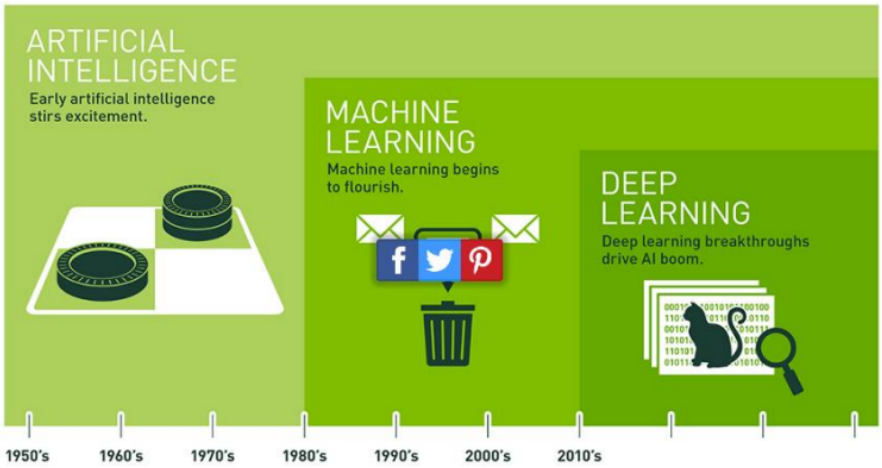


Figura I.2: *Mapa temporal del desarrollo de las inteligencias artificiales. Ya está algo desfasado, faltaría añadir después del Deep Learning los Modelos Generativos.*

No se trata de comparar el aprendizaje humano vs aprendizaje automático, si no combinar ambos para sacar lo mejor de los dos mundos. Habrá tareas que se irán automatizando.

I.1.8. Diseño de sistema de aprendizaje

Muchos métodos de aprendizaje implican formación. La formación es la adquisición de conocimientos, destrezas y competencias como resultado de la enseñanza de aptitudes o conocimientos prácticos relacionados con una competencia útil. La formación requiere escenarios o ejemplos (datos). Existen varios tipos de sistemas de aprendizaje:

- **Aprendizaje no supervisado:** No se proporcionan respuestas o retroalimentación explícita. El sistema debe encontrar patrones o estructuras en los datos por sí mismo.
- **Aprendizaje supervisado:** Se utiliza un conjunto de datos etiquetados, es decir, se proporcionan ejemplos con las respuestas correctas. El sistema aprende a mapear las entradas (x) a las salidas (y) basándose en estos ejemplos.
- **Aprendizaje de refuerzo:** La retroalimentación es indirecta y se recibe después de varias acciones o decisiones. El sistema aprende a través de la interacción con un entorno, recibiendo recompensas o penalizaciones.

I.1.8.1. Supervisado vs no supervisado

Supongamos una función desconocida $y_\Theta(\mathbf{x}) = h_\Theta(\mathbf{x})$, donde x es un ejemplo de entrada y y la salida deseada. En el **aprendizaje supervisado**, se proporciona un

conjunto de pares de entrenamiento (x,y), donde x es la entrada y y es la salida deseada. El objetivo es aprender una función $h_\Theta(\mathbf{x})$ que mapee las entradas a las salidas. En el **aprendizaje no supervisado**, solo se proporcionan las entradas x, y el sistema debe encontrar patrones o estructuras en los datos sin conocer las salidas. Θ hace referencia a los parámetros que tiene el modelo y que hay que entrenar. Por tanto, cuantos menos parámetros haya, más rápido va a ser el modelo.

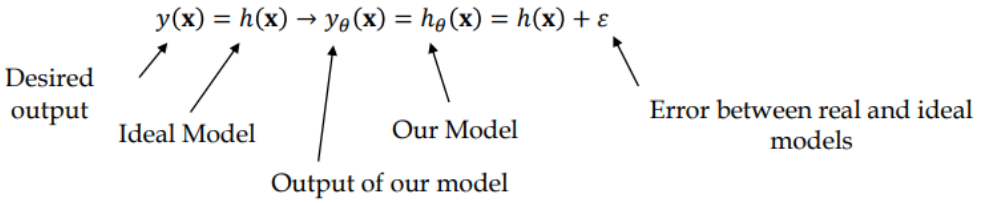
I.1.8.2. Fases de un algoritmo de aprendizaje

Las fases de un algoritmo de aprendizaje son:

1. **Hipótesis y datos**: Los datos son representados como vectores[2] $\mathbf{x}_n = (x_{n1}...x_{nD})^T$, donde D es la dimensión del vector. En el aprendizaje supervisado, también se tienen etiquetas y_n que representan la salida deseada. Los datos pueden ser de diferentes tipos: numéricos, categóricos, texto, series temporales, etc. La información puede estar estructurada (datos genéticos, metereológicos, etc) o no estructurada (imágenes, audio, texto).

2. **Selección del modelo**

   Se elige un modelo $h_\Theta(\mathbf{x})$ que intenta aproximar la relación entre las entradas y las salidas. Por ejemplo, si se elige un modelo lineal, $h_\Theta(\mathbf{x}) = a\mathbf{x} + b$, donde a y b son parámetros que se deben optimizar (correspondientes a Θ_1 y a Θ_2).

   La función de coste $E(y_\Theta - y)^2$ mide el error entre la predicción del modelo y la salida real. Este error se divide en un coste reducible que se puede minimizar optimizando los parámetros del modelo, y un coste irreducible que no puede reducirse con los parámetros actuales, requiriendo un cambio en el modelo o la hipótesis. La función del coste se resume en:

   $$E(y_\Theta - y)^2 = [h_\Theta(\mathbf{x}) - h(\mathbf{x})]^2 + \varepsilon$$

   siendo $[h_\Theta(\mathbf{x}) - h(\mathbf{x})]^2$ el coste reducible y ε el irreducible.

3. **Entrenamiento o aprendizaje**: En esta fase, el modelo se ajusta a los datos de entrenamiento optimizando los parámetros Θ para minimizar la función de coste.

   Por ejemplo, si tenemos un set de datos logarítmico, habría que cambiar de la hipótesis de modelo lineal $h_\Theta(x) = ax + b$ que solo valdría para una recta, por un modelo polinómico $h_\Theta(x) = ax^3 + bx^2 + cx + d$ para poder disminuir el

[2] Aunque no haya una notación general, vamos a utilizar la negrita no itálica para denominar que la variable es un vector.

error ε. El problema es que es muy costoso matemáticamente cuando aumenta el tamaño de los datos, y puede llevar a un sobreajuste del modelo a los datos de entrenamiento. De una información discreta (un set de valores) se busca obtener una solución continua (la función). Todo esto no solo permite obtener una aproximación de los datos intermedios del set de valores dado, si no también una predicción de los datos futuros.

Los errores se suelen representar al cuadrado para que no se compensen los errores negativos con los positivos.

4. **Testeo o inferencia**: Una vez entrenado, el modelo h_Θ se evalúa con datos nuevos (no vistos durante el entrenamiento) para ver cómo generaliza a situaciones no vistas. Así, se predice un nuevo $y(\mathbf{x})$ a un nuevo $\mathbf{x}$. Esto normalmente se hace separando un set de datos en un set de entrenamiento y un set de evaluación de forma aleatoria.

   Una vez en este punto, si el error es muy elevado, se vuelve a la selección del modelo y se establece una nueva hipótesis. Esto es un proceso iterativo en el que se evalúa el rendimiento del sistema hasta que se observe un modelo con un buen ajuste tanto a los valores de entrenamiento como a los valores de test.

I.1.9. Tipos de problemas o tareas

I.1.9.1. Aprendizaje supervisado

Regresión El objetivo de la regresión es predecir el valor de una variable continua. La salida es un valor numérico, y se busca modelar una función continua que relacione las variables de entrada con la salida. Este proceso implica métodos estadísticos para estimar las relaciones entre las variables. Por ejemplo, predecir el precio de una casa en función de su tamaño, ubicación y otras características.

Clasificación En la clasificación, el objetivo es asignar una etiqueta categórica a cada instancia de datos. La salida es una etiqueta discreta, como "maligno" o "benigno". El límite de decisión es una hipersuperficie que divide el espacio de características en regiones, cada una asociada a una clase. Por ejemplo, en la clasificación de un tumor, se utilizan características como el tamaño y la tasa de crecimiento para determinar si el tumor es maligno o benigno. Aquí, las características (tamaño y tasa de crecimiento) definen el espacio de entrada, y la etiqueta (maligno/benigno) es la salida binaria.

I.1.9.2. Aprendizaje no supervisado

Clustering El clustering es una técnica de aprendizaje no supervisado que busca agrupar un conjunto de objetos (o datos) de manera que aquellos que pertenecen al mismo grupo (clúster) sean más similares entre sí que con los objetos de otros grupos. La similitud se mide utilizando métricas de distancia (como la distancia euclidiana) o similitud, dependiendo del tipo de datos y del algoritmo utilizado. Un ejemplo común es la agrupación de clientes en segmentos basados en su comportamiento de compra, donde cada clúster representa un grupo de clientes con características similares.

I.2. Reducción de dimensionalidad

La reducción de dimensionalidad es una técnica fundamental en el aprendizaje automático que permite representar datos multidimensionales en un espacio de menor dimensión, preservando la mayor cantidad posible de información relevante. Esto es especialmente útil para visualización, mejora del rendimiento y manejo de la maldición de la dimensionalidad.

- **Visualización**: Permite visualizar datos en 2D o 3D, incluso cuando los datos originales tienen muchas más dimensiones.
- **Mejora del rendimiento**: Reduce el tiempo de entrenamiento y el uso de memoria al trabajar con menos dimensiones. Además, elimina ruido y redundancia en los datos.
- **Maldición de la dimensionalidad**: Cuando el número de dimensiones es muy alto en comparación con el número de muestras, los modelos pueden volverse ineficientes o sobreajustarse. Ejemplo: No tiene sentido ajustar un modelo con 2.000 parámetros si solo se dispone de 10 datos. Los parámetros representan grados de libertad, y en este caso, el modelo no generalizaría bien. La reducción de dimensionalidad ayuda a eliminar información redundante y mejorar el rendimiento.

No obstante, no siempre es necesario o beneficioso reducir la dimensionalidad. Por ejemplo, si las dimensiones originales ya son interpretables y no hay redundancia, o si la pérdida de información al reducir dimensiones afecta negativamente al modelo.

El objetivo es reducir el número de variables (dimensiones) en un conjunto de datos, manteniendo la estructura y la información más importante. La herramienta más común es **PCA (Principal Component Analysis)**, un algoritmo basado en álgebra lineal que transforma los datos originales en un nuevo sistema de coordenadas, donde las dimensiones (componentes principales) capturan la mayor varianza posible.

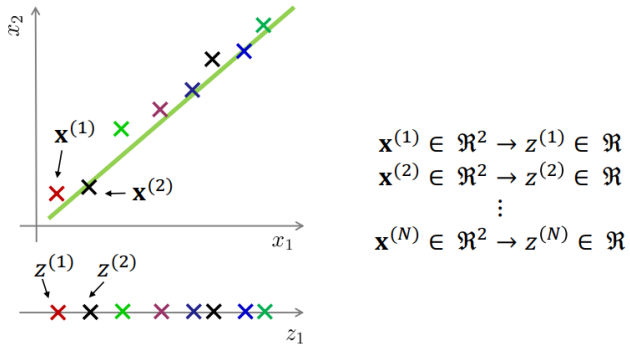
I.2.1. Proyección de datos en dimensiones inferiores

La idea central de la reducción de dimensionalidad es proyectar datos de un espacio de alta dimensión a uno de menor dimensión, preservando la estructura subyacente.

2D a 1D Supongamos que tenemos datos en un espacio bidimensional ($\mathbb{R}^2$). Queremos proyectarlos en una recta unidimensional ($\mathbb{R}$). La recta es una combinación lineal de las dos dimensiones originales (desde la recta, solo nos podemos mover en una dirección, hacia delante o hacia detrás), y la proyección de los datos a esa recta no conlleva pérdida de información si la recta captura la dirección de máxima varianza. Matemáticamente:

$$\mathbf{x}^{(1)} \in \mathbb{R}^2 \rightarrow z^{(1)} \in \mathbb{R}$$
$$\mathbf{x}^{(2)} \in \mathbb{R}^2 \rightarrow z^{(2)} \in \mathbb{R}$$
$$\mathbf{x}^{(N)} \in \mathbb{R}^2 \rightarrow z^{(N)} \in \mathbb{R}$$

El superíndice sirve para anotar el dato, y el subíndice para la dimensión.



Extensión a más dimensiones Esta idea se puede generalizar a espacios de mayor dimensión. Por ejemplo, al pasar de 3D ($\mathbb{R}^3$) a 2D ($\mathbb{R}^2$), se proyectan los datos en un plano bidimensional:

$$\mathbf{x}^{(i)} \in \mathbb{R}^3 \rightarrow z^{(i)} \in \mathbb{R}^2$$

I.2.2. PCA: Análisis de Componentes Principales

El Análisis de Componentes Principales (PCA) es una técnica de reducción de dimensionalidad que transforma datos de alta dimensión en un espacio de menor dimensión, preservando la mayor cantidad posible de información (varianza). Es especialmente útil cuando se trabaja con datos multidimensionales, como en el caso de una **tabla de expresión génica**, donde las filas representan pacientes y las columnas corresponden a genes individuales.

Ejemplo: Tabla de expresión génica Cada paciente puede describirse como un punto en un espacio de 2000 dimensiones: $\mathbf{x}^{(i)} \in \mathbb{B}^{2000}$, donde cada componente del vector representa la expresión de un gen. PCA permite reducir estas 2000 dimensiones a un espacio de menor dimensión, por ejemplo, a dos dimensiones: $\mathbf{z}^{(i)} \in \mathbb{R}^2$.

Nota: Al proyectar los datos, los valores transformados pierden su significado biológico directo (ya no representan expresiones génicas específicas). Sin embargo, la proximidad entre puntos en el espacio bidimensional puede interpretarse como una medida de similitud genética entre pacientes.

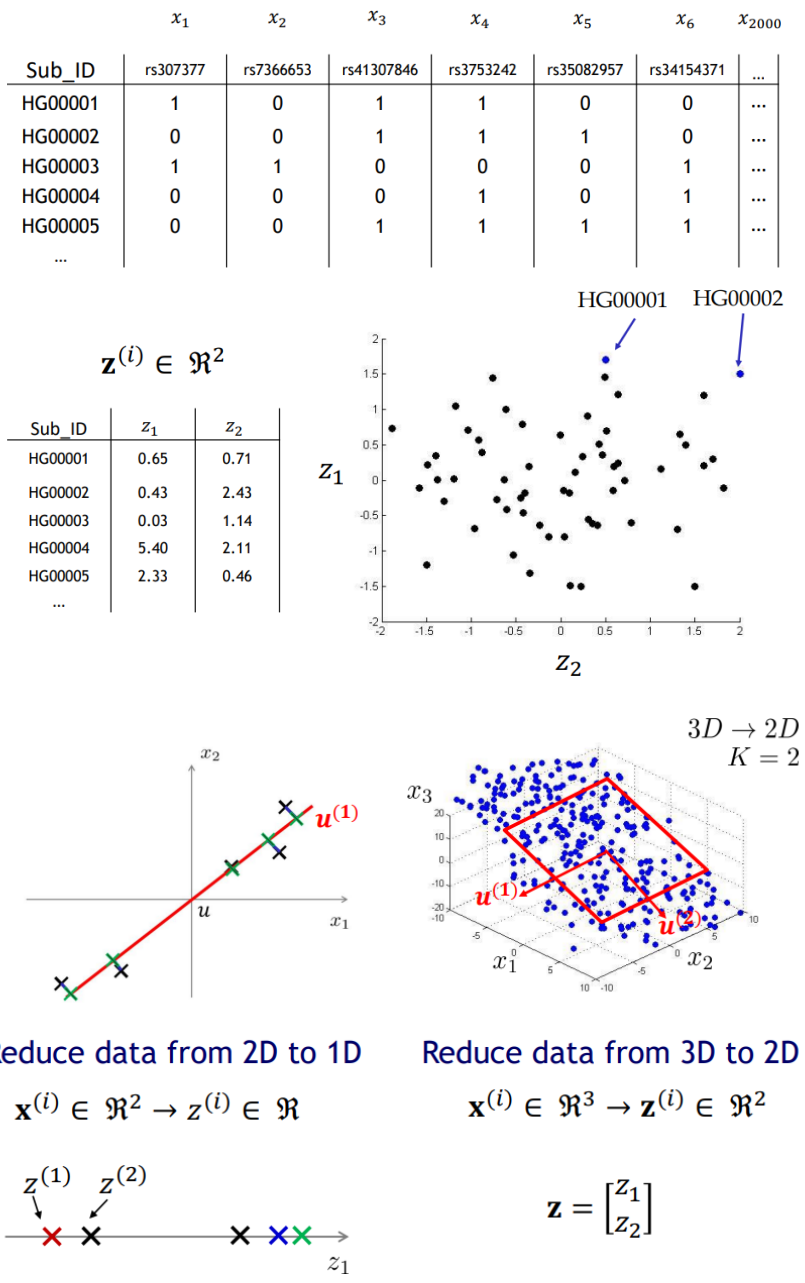
PCA busca minimizar el error de proyección de los datos sobre un subespacio de menor dimensión. Para reducir un espacio de n dimensiones a k dimensiones, PCA determina k vectores ortogonales $\mathbf{u}^{(k)}$ que definen el subespacio óptimo para proyectar los datos:

$$\mathbf{x} \in \mathbb{R}^n \rightarrow \mathbf{z} \in \mathbb{R}^k$$

I.2.2.1. Algoritmo de PCA

1. **Preprocesado de datos**

   Antes de aplicar PCA, es común normalizar los datos para que todas las variables tengan la misma escala. Esto se hace restando la media y dividiendo por la



desviación estándar:

$$\mathbf{z}_1 = a\mathbf{x}_1 + b\mathbf{x}_2$$

Nota: Este paso no es estrictamente necesario si las variables ya están en escalas comparables.

2. **Matriz de covarianza**

Dada una matriz de datos $\mathbf{x} \in \mathbb{R}^{M \times N}$ donde M es el número de muestras y N el número de características, se calcula la matriz de covarianza $\Sigma \in \mathbb{R}^{n \times n}$.

$$\Sigma = \frac{1}{M}\mathbf{x}^T\mathbf{x}$$

La matriz de covarianza mide la correlación entre las variables. La diagonal principal contiene las varianzas de cada variable.

3. **Autovalores y autovectores**

   Los **autovectores** $\breve{U}$ representan las direcciones en las que los datos varían más (direcciones de máxima varianza). Tiene la dimensión: $\breve{U} \in \mathbb{R}^{N \times N}$

   Los **autovalores** λ indican la cantidad de varianza capturada en cada dirección. Tiene la dimensión: $\lambda \in \mathbb{R}^{N \times 1}$.

   La matriz de autovectores $\breve{U}$ y el vector de autovalores λ se obtienen resolviendo:

$$\Sigma \breve{U} = \lambda \breve{U}$$

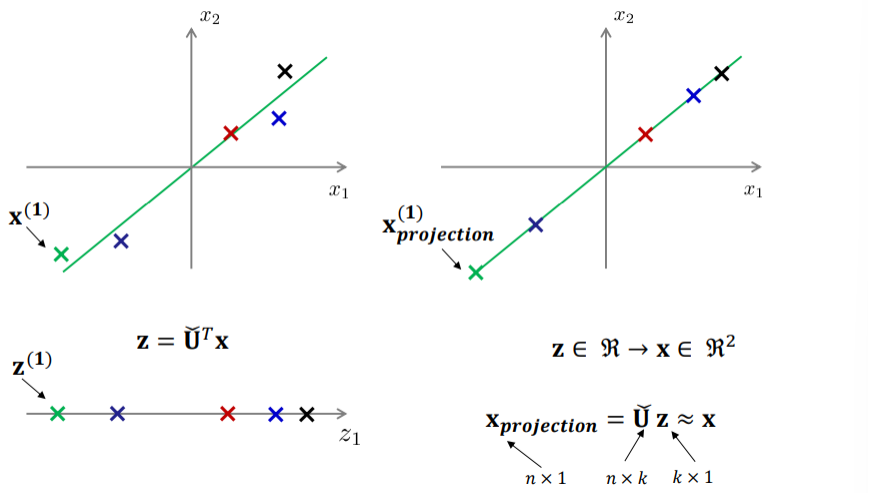
4. **Ordenación y selección de componentes**

   Los autovalores se ordenan de mayor a menor, y los autovectores se reordenan en consecuencia. Para reducir a k dimensiones, se seleccionan los k autovectores asociados a los k autovalores más grandes.

5. **Proyección de los datos** Los datos originales $\mathbf{x}$ se proyectan en el nuevo espacio de k dimensiones multiplicando por la matriz de autovectores seleccionados:

$$\mathbf{z} = \breve{U}_k^T \mathbf{x}$$

   La matriz de autovectores seleccionados es $\breve{U}_k \in \mathbb{R}^{N \times k}$, mientras que los datos proyectados en el espacio reducido son $\mathbf{z} \in \mathbb{R}^{k \times 1}$.



Ejemplo práctico Supongamos que queremos reducir datos de 2D a 1D:

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \rightarrow \lambda = \begin{bmatrix} \lambda_1 \\ \dots \\ \lambda_n \end{bmatrix} \in \mathbb{R}^{n \times 1}$$

Si el autovector es

$$U = \begin{bmatrix} -1 & 2 \\ 3 & 0 \end{bmatrix}$$

Se debe seleccionar solo

$$\mathbf{U} = \begin{pmatrix} \mathbf{u}^{(i)} & \dots & \mathbf{u}^{(n)} \end{pmatrix} \in \mathbb{R}^{n \times n} \rightarrow \breve{U} = \begin{bmatrix} -1 \\ 3 \end{bmatrix}$$

Así, la proyección queda de la siguiente forma:

$$z = \breve{U}_1^T \mathbf{x} = \begin{bmatrix} -1 & 3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = -x_1 + 3x_2$$

En caso de no reducir dimensiones:

$$z = \begin{bmatrix} -1 & 3 \\ 2 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} -1x_1 + 3x_2 \\ 2x_1 + 0x_2 \end{bmatrix} \sim \begin{bmatrix} z_1 \\ z_2 \end{bmatrix}$$

I.2.2.2. Escoger el número de componentes principales

La varianza en cada componente de PCA está definida por los autovalores λ. La varianza explicada de un componente i se explica mediante la siguiente fórmula:

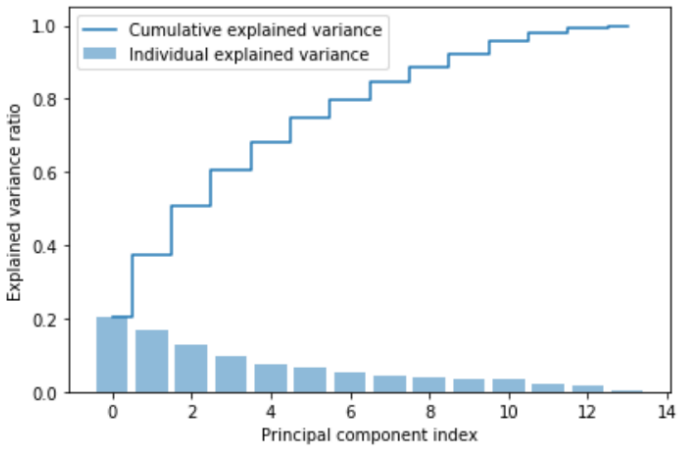
$$i = \frac{\lambda_i}{\sum_{i=1}^{n} \lambda_i}$$

Normalmente se escoge que k sea el valor más pequeño posible de forma que:

$$\frac{\sum_{j=1}^{k} \lambda_j}{\sum_{i=1}^{n} \lambda_i} \geq \tau$$

Si $\tau = 0.99$, entonces el 99 % de la varianza está retenida.

Esto se suele representar con una curva de varianza explicada sobre el número de componentes (el gráfico está indexado con Python, por lo que 1 componente se representa con un 0). Los primeros componentes sí se analizan de forma individual al explicar bastante varianza, pero llega un punto en el que se busca el número de componentes que expliquen el x % de la varianza (por ejemplo, el 95 %).



I.2.3. Otros métodos de reducción de dimensionalidad

- **Multidimensional Scaling (MDS)**: define un espacio de baja dimensión que preserva la distancia entre casos en el espacio original de alta dimensión.
- **Discriminant Analysis (LDA)**: calcular una función que maximice la capacidad de discriminación entre 2 o más grupos. Es una especie de PCA supervisado.
- **Manifold Learning (p.ej. Isomap)**: descubrir representaciones de baja dimensión en variedades lisas localmente euclidianas. Hay estructuras de datos en las que se deben cumplir una serie de leyes. En el espacio de representación, puede haber una distribución de datos que impida ir de un punto a otro de forma directa, sin pasar por la estructura definida (por ejemplo, una espiral).
- **t-SNE**: reduce la dimensionalidad preservando la similitud local, se ha construido heurísticamente y se utiliza habitualmente para visualizar representaciones.

Capítulo II

Aprendizaje no supervisado - Clustering

II.1. Clustering

En el **aprendizaje supervisado**, se dispone de etiquetas (valores de y) que permiten medir el rendimiento del modelo. En cambio, en el **aprendizaje no supervisado**, no se tienen etiquetas, y el objetivo es descubrir patrones o estructuras ocultas en los datos. Una de las técnicas más comunes es el **clustering**, que agrupa los datos en función de sus características o similitudes.

El clustering se basa en la **proximidad** entre datos o ítems. Esta proximidad puede medirse de diversas formas, dependiendo del tipo de datos y del problema:

- **Distancia euclídea:** La distancia más corta entre dos puntos en un espacio euclídeo.
- **Distancia de Hamming:** Utilizada en datos binarios, mide el número de bits que difieren entre dos vectores.
- **Distancia de Manhattan:** Mide la distancia entre dos puntos moviéndose solo en direcciones horizontales o verticales (no en diagonal).

En casos más complejos, como con palabras o distribuciones estadísticas, la proximidad puede medirse utilizando técnicas avanzadas, como las empleadas en modelos de lenguaje (LLMs) o métricas específicas para distribuciones.

II.1.1. Definición de Clustering

El clustering es la organización de una colección de patrones en grupos (clústeres) basados en su similitud (Jain, Murty, et al. 1999). Mientras que clustering es aprendizaje no supervisado, la clasificación sí lo es, ya que se utilizan las etiquetas de los datos para, a datos nuevos, asignarles esas etiquetas. Un clúster puede definirse como:

- conjunto de objetos similares.

- un grupo de puntos donde la distancia entre dos puntos del mismo clúster es menor que la distancia entre cualquier punto del clúster y cualquier punto de otros clústeres.
- regiones densamente conectadas en un espacio multidimensional separadas por puntos o regiones poco conectados.

II.1.2. Distancia

La distancia entre dos instancias $x^{(i)}$ y $x^{(j)}$ es una métrica si satisface las siguientes propiedades:

1. **Desigualdad triangular**

$$d(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) \leq d(\mathbf{x}^{(i)}, \mathbf{x}^{(k)}) + d(\mathbf{x}^{(k)}, \mathbf{x}^{(j)}), \forall \mathbf{x}^{(i)}, \mathbf{x}^{(j)}, \mathbf{x}^{(k)} \in \mathbb{R}^n$$

2. **Identidad de los indiscernibles**

$$d(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) = 0 \rightarrow \mathbf{x}^{(i)} = \mathbf{x}^{(j)}, \forall \mathbf{x}^{(i)}, \mathbf{x}^{(j)} \in \mathbb{R}^n$$

II.2. Algoritmo K-means

K-means es uno de los algoritmos más utilizados en clustering. Su objetivo es dividir un conjunto de datos en K clústeres, donde cada dato pertenece al clúster cuyo centroide (punto central) está más cerca.

II.2.1. Pasos del algoritmo K-means

1. **Inicialización**: Se elige el número de clústeres K. Se inicializan K centroides de forma aleatoria.
2. **Asignación**: Cada dato se asigna al clúster cuyo centroide está más cerca (según una métrica de distancia, como la euclídea).
3. **Actualización**: Se recalcula la posición de cada centroide como el promedio (centro de masas) de todos los datos asignados a su clúster.
4. **Iteración**: Los pasos de asignación y actualización se repiten hasta que los centroides ya no cambian significativamente.

II.2.2. Inicialización aleatoria y óptimos locales

La inicialización aleatoria de los centroides puede llevar a soluciones subóptimas. Para mitigar esto, se pueden utilizar técnicas como:

Input:
- K: number of clusters
- $\{\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots, \mathbf{x}^{(m)}\}$ training set $\mathbf{x}^{(i)} \in \Re^n$

STEP 0: Random initialization K centroids $\mu_1, \mu_2, \ldots, \mu_K \in \Re^n$

Repeat {

STEP 1: for i = 1 to m

$c^{(i)}$:= index of the cluster (from 1 to K) closest to $\mathbf{x}^{(i)}$

$$c^{(i)} := \arg\min_k \|\mathbf{x}^{(i)} - \mu_k\|^2$$

STEP 2: for k = 1 to K

μ_k := mean value of the data assigned to the cluster k

}

$$\mu_k := \frac{\sum_{i=1}^{m} 1\{c^{(i)} = k\}\mathbf{x}^{(i)}}{\sum_{i=1}^{m} 1\{c^{(i)} = k\}}$$

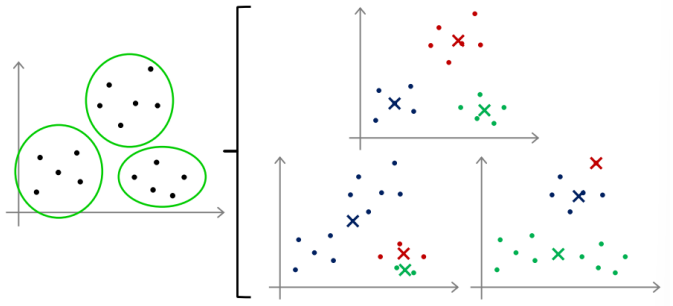
- **Inicialización basada en datos**: Seleccionar K datos aleatorios como centroides iniciales.
- **Repetición del algoritmo**: Ejecutar K-means varias veces y seleccionar la solución con la menor inercia (suma de las distancias al cuadrado entre cada punto y su centroide).

Al hacer esto, el algoritmo no es determinista, ya que los resultados no van a ser siempre los mismos, aunque pueda haber resultados más y menos estables.

II.2.3. Superposición entre clústeres

En algunos casos, los clústeres pueden superponerse, lo que da lugar a clústeres difusos. Por ejemplo, en la clasificación de tallas de ropa (S, M, L), algunos puntos pueden estar en la frontera entre dos clústeres, lo que dificulta su asignación clara.

Cuando no hay tanta separación entre los datos, se pueden dar soluciones dispares. Esto se puede resolver mediante la repetición y medir el rendimiento de las distintas ejecuciones para ver qué solución es la mejor. Se calcula la distancia de cada punto a su centroide y la solución que minimice esa distancia es la que obtiene los centroides más óptimos.



II.2.4. Función de coste

K-means no garantiza encontrar el mínimo global de la función de coste, ya que el resultado depende de la inicialización. La función de coste (o función de distorsión) se

define como J y tiene el siguiente aspecto:

$$J(c^{(1)}, \dots, c^{(m)}, \mu_1, \dots \mu_k) = \frac{1}{m}\Sigma_{i=1}^{m}||\mathbf{x}^{(i)} - \mu_{c^{(i)}}||^2$$

donde:

- $c^{(i)}$ es el índice del clúster al que se ha asignado $\mathbf{x}^{(i)}$
- μ_k es el centroide del clúster k ($\mu_k \in \mathbb{R}^n$)
- $\mu_{c^{(i)}}$ es el centroide del clúster al que se ha asignado $\mathbf{x}^{(i)}$. Es una matriz de m elementos.

STEP 0: Random initialization of K centroids $\mu_1, \mu_2, \dots, \mu_K \in \mathbb{R}^n$

Repeat {

STEP 1: for i = 1 to m
$c^{(i)}$:= index of the cluster (from 1 a K) closest to $\mathbf{x}^{(i)}$

$$\min_{c^{(1)},\dots,c^{(m)}} J(c^{(1)}, \dots, c^{(m)})$$

STEP 2: for k = 1 to K
μ_k := mean value of the data assigned to the cluster k

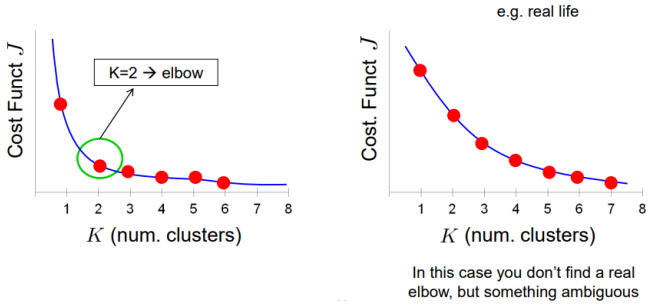
$$\min_{\mu_1,\dots,\mu_K} J(c^{(1)}, \dots, c^{(m)}, \mu_1, \dots, \mu_K)$$

}

Como se ha descrito antes, esto se repite durante varios ciclos para escoger aquel clustering que dé el menor coste.

II.2.5. Escoger el número de centroides

Si el objetivo único es minimizar el coste, la forma de obtener $J = 0$ sería poniendo para cada punto un clúster, pero esto no agrupa los datos según patrones. Por ello, es importante escoger k. Si conocemos nuestros datos, podemos saber qué número de k puede ser el óptimo. Otra opción es con el método del codo: se ejecuta el algoritmo de K-means varias veces cambiando k y se ve cuándo la función de coste se ve significativamente reducida.



II.3. Bonus track: otros algoritmos

El algoritmo Gaussian Mixture Models (GMM) permite adaptar los clústers a los datos. De esa forma, en lugar de ser clústeres redondos, se ajustan a la distribución que presenten, pudiendo adoptar otras formas.

Capítulo III

Aprendizaje supervisado

III.1. Introducción

El aprendizaje automático se divide en dos grandes categorías: aprendizaje supervisado y aprendizaje no supervisado. En el aprendizaje supervisado, el objetivo es aprender una **función de hipótesis** desconocida que permita predecir una salida y a partir de una entrada x. Este enfoque se conoce como sistema **data-driven**, ya que el modelo aprende patrones a partir de datos etiquetados.

III.1.1. Datos etiquetados

En el aprendizaje supervisado, cada dato tiene una etiqueta asociada, lo que se representa como pares $(x^{(i)}, y^{(i)})$. Partimos de un **conjunto de datos de entrenamiento**:

$$D_{train} = \{(\mathbf{x}_n, y_n)\}_{n=1}^{N_{train}}$$

siendo:

- N_{train}: número de muestras de entrenamiento.
- $\mathbf{x}_n$: vector de atributos (inputs, características, variables independientes).
- y_n: etiqueta o valor objetivo.

El objetivo es **inducir o aprender** a partir de los datos de entrenamiento un **predictor** h que permita predecir y para nuevos datos. La clave no es memorizar los datos de entrenamiento, sino **generalizar** para hacer predicciones precisas en situaciones similares pero no idénticas (capacidad de generalización).

III.1.2. Función de hipótesis y predictor

El algoritmo de aprendizaje L toma los datos de entrenamiento y busca una función h que sirva como predictor:

$$L : D_{train} \to h$$

- h: función de hipótesis que mapea entradas x a salidas y.
- El predictor debe ser capaz de generalizar, es decir, funcionar bien con datos no vistos durante el entrenamiento.

III.1.3. Tipos de problemas en aprendizaje supervisado

Clasificación En un problema de clasificación, hay un espacio que se divide en las distintas etiquetas que adquieren los datos. No se trata de buscar la línea divisoria entre los distintos elementos, si no la caracterización de los espacios en los que se encuentran. Esa línea divisoria, o frontera de decisión, es importante para dicha caracterización. La salida es discreta, ya que se trata de etiquetas categóricas. Puede ser binaria (dos clases) o multiclase, pero nunca un dato podrá estar entre dos etiquetas. Además, las etiquetas pueden seguir un orden (etiqueta 1 $<$ etiqueta 2 $<$ etiqueta 3) o no.

Regresión Un ejemplo es la predicción del tamaño de un tumor en función del tiempo que lleva desarrollándose mediante la descripción de la tasa de crecimiento. En este caso, los datos son el tiempo (distinto número de semanas) y las etiquetas el tamaño del tumor, y se busca describir el tamaño para datos nuevos, como pueden ser semanas no descritas. Aun así, para una misma semana, puede haber distintas tasas de crecimiento dependiendo de la persona. En este caso sí se busca encontrar la recta. Puede haber varias soluciones independientes, y en este caso la salida es continua (un valor numérico).

El número máximo de clases a partir de que un problema pasa de clasificación a regresión depende de la naturaleza del problema. Si el problema es de naturaleza continua, entonces la aproximación será mediante regresión, mientras que si el problema es discreto, entonces utilizaremos la clasificación. Si existe una relación entre muestras consecutivas, se espera que una distancia entre muestras esté correlacionada o haya consecuencialidad.

En resumen, en problemas de clasificación, el espacio de características se divide en regiones según las etiquetas. Por ejemplo, en un problema binario, se busca una frontera de decisión que separe las dos clases. En regresión, se busca una función (como una recta) que ajuste los datos.

III.1.4. Función de pérdida

Hasta ahora, seguíamos la siguiente notación:

- x, y: son variables aleatorias con una distribución conjunta, es decir, hay un patrón que permite predecir una variable en función de la otra.
- Predictor h: permite obtener y a partir de x.

La **función de pérdida** L mide la discrepancia entre la predicción $h(x)$ y la etiqueta real y. El objetivo es minimizar esta pérdida. Existen dos tipos comunes de funciones de pérdida:

1. Clasificación
   - Pérdida 0-1:
     $$L(h(x), y) = \mathbb{I}(h(x) \neq y)$$
     donde $\mathbb{I}$ es una función indicadora que vale 1 si la predicción es incorrecta y 0 si es correcta. Así, el error se calcula como la suma de las predicciones incorrectas. Un sistema que acierte mucho tendrá un indicador $\mathbb{I}$ cercano a 0, mientras que un sistema que falle mucho tendrá un $\mathbb{I}$ alto que habrá que normalizar por los intentos realizados.
2. Regresión
   - Error cuadrático medio (MSE):
     $$MSE = \mathbb{E}[|h(\mathbf{X}) - Y|^2]$$
   - Error absoluto medio (MAE):
     $$MAE = \mathbb{E}[|h(\mathbf{X}) - Y|]$$
   - El MSE magnifica los errores grandes debido al cuadrado, mientras que el MAE es más robusto y mantiene el sentido físico-biológico de la variable.

III.1.5. Pérdida esperada y generalización

La pérdida esperada es un estimador que mide el rendimiento del modelo en toda la población, no solo en los datos de entrenamiento. Esto se debe a que los datos de entrenamiento se definen en una subregión o espacio finito de la región de todos los posibles valores. Se busca que el entrenamiento tenga un rendimiento bueno con datos conocidos y desconocidos. Se define como:

$$Error = \mathbb{E}[\mathbb{I}(h(\mathbf{X}) \neq Y)]$$

El objetivo es minimizar la pérdida esperada, lo que implica que el modelo generalice bien a datos nuevos.

III.1.6. Entrenamiento y optimización

El entrenamiento consiste en ajustar los **parámetros** Θ del modelo para minimizar la pérdida en los datos de entrenamiento. La pérdida promedio en el conjunto de entrenamiento se define como:

$$L_{train}(\Theta) = \frac{1}{N_{train}} \sum_{n=1}^{N_{train}} L(h(\mathbf{x}_n^{train}; \Theta), y_n^{train})$$

- Θ: parámetros del modelo (por ejemplo, coeficientes en regresión lineal).
- L_{train}: pérdida promedio en el conjunto de entrenamiento.

Sin embargo, minimizar L_{train} no garantiza un buen rendimiento en datos nuevos. Esto se debe al **sobreajuste** (overfitting), donde el modelo memoriza los datos de entrenamiento pero no generaliza bien a datos no vistos.

Para evitar el sobreajuste, se introduce una **penalización de complejidad** en la función de pérdida. Esto limita el número de parámetros o la magnitud de los mismos, favoreciendo modelos más simples y generalizables. Ejemplos comunes incluyen la regularización L1 (Lasso) y L2 (Ridge) que penalizan la suma absoluta o cuadrática de los parámetros respectivamente.

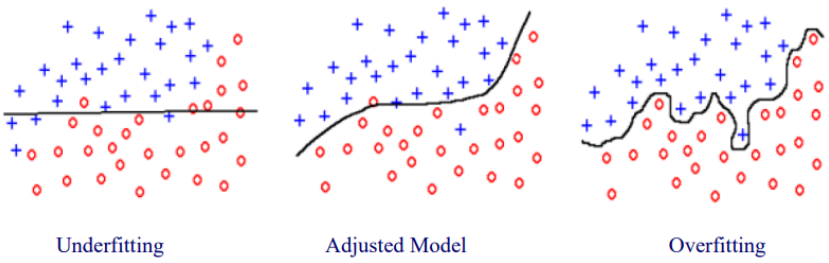
III.1.6.1. Sesgos en el entrenamiento

La pérdida de entrenamiento suele ser menor que la pérdida en datos nuevos, ya que el modelo se optimiza específicamente para los datos de entrenamiento. Esto introduce un **sesgo** en el proceso de entrenamiento. Para evaluar el rendimiento del modelo en datos no vistos, se utiliza un **conjunto de test**:

Los datos de test no se usan para ajustar Θ, solo para evaluar el rendimiento. La pérdida en el conjunto de test se calcula de manera similar a L_{train}, pero sin modificar Θ.

Underfitting El tipo de predictor considerado tiene **poca capacidad expresiva** o pocos grados de libertad. En consecuencia, no es capaz de capturar las dependencias entre los atributos y la variable a predecir. La pérdida esperada del predictor es demasiado elevada. Esto suele ocurrir en modelos rígidos, como pueden ser los modelos lineales.

Overfitting El tipo de predictor considerado es **demasiado flexible** (demasiados grados de libertad) y aprende patrones espurios que no son relevantes para la predicción (por ejemplo, fluctuaciones de muestreo, ruido, valores atípicos, etc.). La estimación de entrenamiento de la pérdida esperada es demasiado optimista y subestima la pérdida esperada real. Esto puede ocurrir en modelos flexibles, como pueden ser las redes neuronales.



III.1.7. Compensación entre sesgo y varianza

Se tiene un sesgo bajo en modelos flexibles que se ajustan bien a los datos de entrenamiento, pero pueden tener alta varianza (sobreajuste). El sesgo alto se

suele dar en modelos rígidos que no se ajustan bien a los datos, pero tienen baja varianza (underfitting). Por ello, se describe la compensación. A medida que aumenta la complejidad del modelo, el error de entrenamiento disminuye, pero el error de test puede aumentar después de un punto óptimo debido al sobreajuste.

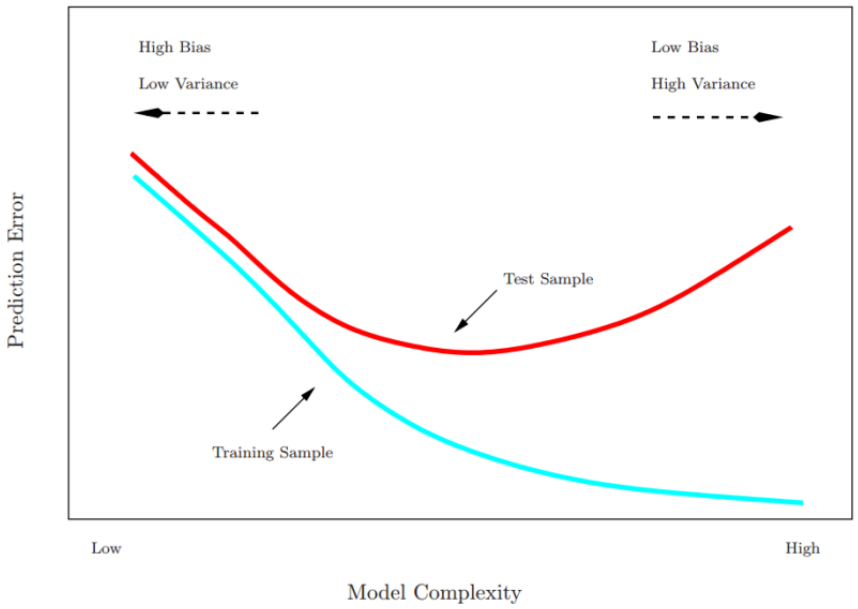


Figura III.1: *En el eje x está la complejidad del modelo, o el orden del polinomio. En el eje y está la predicción del error. Se empieza con un error alto, y se va adaptando, mediante modelos más complejos, para disminuir el error. No obstante, llega un momento en el que, aunque el error de entrenamiento siga bajando con modelos cada vez más complejos, el error de test vuelve a aumentar, indicando un modelo sobreajustado.*

III.1.8. Arquitectura, parámetros

- **Arquitectura, configuración**: Especificación de la forma funcional del predictor (por ejemplo, número y tipo de capas ocultas y neuronas en una red neuronal, elección del núcleo en una SVM, etc.). En el caso de la regresión polinomial, es el polinomio.
- **Parámetros del modelo** Θ: Valores necesarios para especificar el sistema de predicción (por ejemplo, pesos sinápticos en una red neuronal, vectores de soporte en una SVM, etc.). Se determinan entrenando el modelo con datos etiquetados.
- **Configuración del algoritmo de aprendizaje**: Función a optimizar (por ejemplo, verosimilitud, probabilidad posterior, error cuadrático medio, etc.). Términos de regularización

Hay distintos tipos de modelos predictivos: vecinos cercanos, árboles de decisión, redes neuronales, Support Vector Machine.

III.1.9. Preprocesado de datos

El preprocesado es crucial para preparar los datos antes de entrenar un modelo. Incluye:

1. **Selección de características**: identificar las características más relevantes
2. **Manejo de valores atípicos (outliers)**: detectar y tratar datos anómalos
3. **Manejo de datos faltantes**: imputar o eliminar valores faltantes
4. **Normalización**: centrar y escalar los datos para que tengan media 0 y desviación estándar 1. La normalización basada en la media y desviación estándar es más robusta que usar valores mínimos y máximos.

El preprocesado se puede realizar programáticamente con el módulo de scikit-learn.

III.1.10. Medición del rendimiento

III.1.10.1. Error de clasificación

El error de clasificación se estima como la proporción de veces que el modelo predice incorrectamente las etiquetas. Es una métrica básica para evaluar la efectividad del modelo.

III.1.10.2. Matriz de confusión

La matriz de confusión es una herramienta fundamental para evaluar el rendimiento en problemas de clasificación. Muestra las combinaciones entre las etiquetas reales y las predichas. En clasificación binaria, se divide en:

- **True Positive (TP)**: correctamente clasificados como positivos.
- **True Negative (TN)**: correctamente clasificados como negativos.
- **False Positive (FP)**: incorrectamente clasificados como positivos.
- **False Negative (FN)**: incorrectamente clasificados como negativos.

En problemas con más de dos clases, la matriz de confusión se extiende, y la diagonal principal indica las predicciones correctas, mientras que las demás celdas muestran los errores. Esto es útil para identificar qué clases se confunden con mayor frecuencia.

III.1.10.3. Métricas de rendimiento

A partir de la matriz de confusión, se derivan varias métricas para evaluar el rendimiento del modelo:

- **Accuracy (exactitud)**: mide la efectividad general del modelo.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

- **Error**: estimación de la proporción de clasificación incorrectas.

$$Error = 1 - Accuracy = \frac{FP + FN}{TP + TN + FP + FN}$$

- **Sensitividad (Recall)**: ratio de positivos correctamente identificados.

$$Sensitividad = \frac{TP}{TP + FN}$$

- **Especifidad**: ratio de negativos correctamente identificados.

$$Especificidad = \frac{TN}{TN + FP}$$

- **Precisión**: ratio de predicciones positivas que son correctas.

$$Precision = \frac{TP}{TP + FP}$$

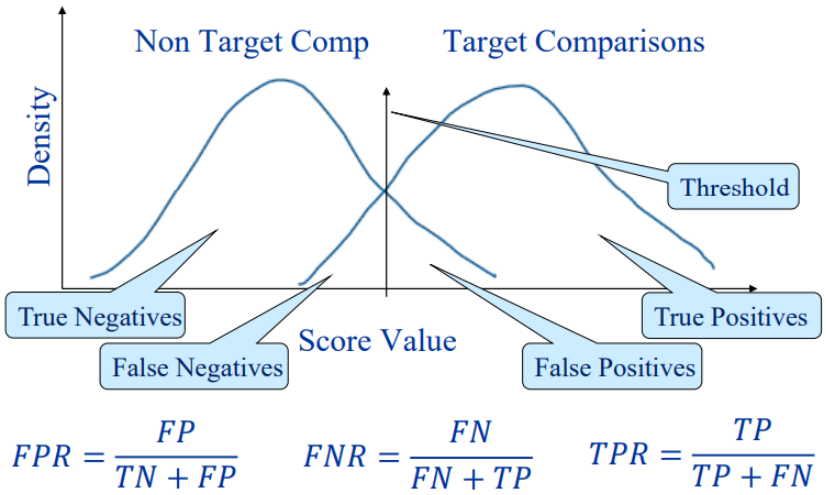
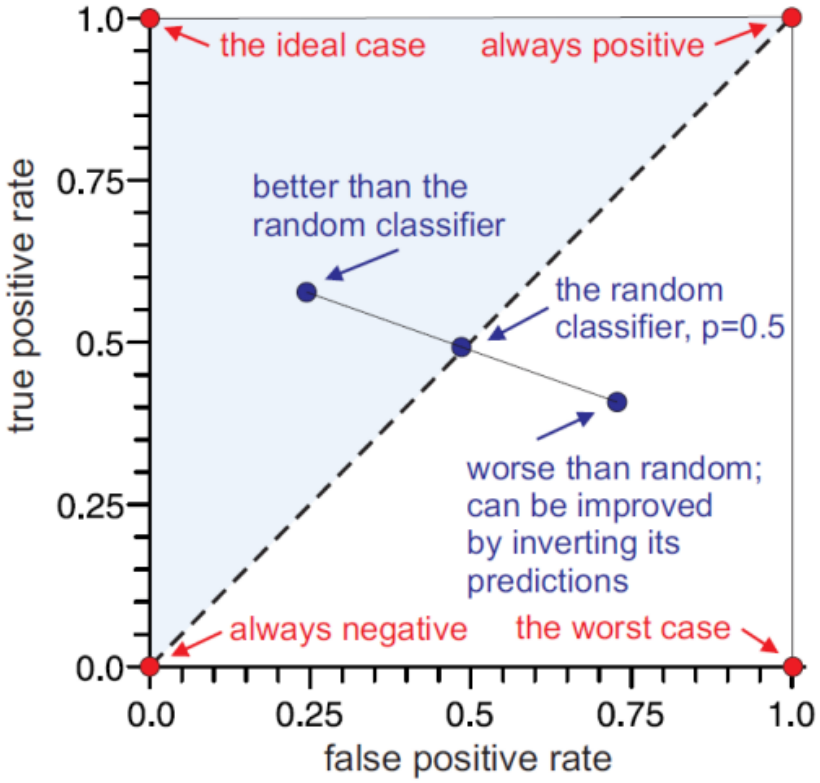


Figura III.2: *En la literatura se habla de Non Target Comparison en lugar de Negativos por la connotación. Normalmente hay un cierto solape entre las distribuciones target y non-target, produciendo así un error. El umbral de decisión se coloca dependiendo del error que se quiera priorizar (o si no se quiere priorizar ninguno).*

III.1.10.4. Curva ROC y AUC

La curva ROC es una gráfica la tasa de verdaderos positivos (TPR; sensitividad) frente a la tasa de falsos positivos (FPR) para diferentes umbrales de decisión. La



matriz de confusión solo tiene sentido cuando ya están las etiquetas y se ha definido un umbral, pero esto no es necesario para la curva ROC.

El AUC (Área bajo la curva) mide el rendimiento general del clasificador. Un AUC cercano a 1 indica un buen rendimiento, mientras que un AUC cercano a 0.5 sugiere un rendimiento aleatorio.

La curva ROC es útil para decidir el umbral de decisión óptimo, especialmente cuando se quiere priorizar la minimización de falsos negativos o falsos positivos.

III.1.10.5. Métricas para regresión

En problemas de regresión, las métricas comunes incluyen:

- **Error cuadrático medio (MSE)**:

$$MSE = \mathbb{E}[(h(\mathbf{X};\Theta) - Y)^2] \approx \frac{1}{N}\sum_{n=1}^{N}(h(\mathbf{x}_n;\Theta) - y_n)^2$$

- **Error absoluto medio (MAE)**:

$$NMSE = \frac{MSE}{Var}$$

$$Var = \frac{1}{N}\sum_{n=1}^{N}(y_n - \bar{y})^2$$

- **Error cuadrático medio normalizado (NMSE)**:

$$MAE = \mathbb{E}[|h(\mathbf{X};\Theta) - Y|] \approx \frac{1}{N}\sum_{n=1}^{N}|h(\mathbf{x}_n;\Theta) - y_n|$$

III.1.11. Protocolos experimentales

En general, tenemos un conjunto de datos que debemos dividir nosotros en dos conjuntos, uno de train y uno de test.

Holdout Se divide el conjunto de datos en dos partes: una para entrenamiento (por ejemplo, 70 %) y otra para test (30 %). Problema: La partición puede no ser representativa si los datos no están bien distribuidos.

K-Fold Cross Validation El conjunto de datos se divide en K subconjuntos (folds). El modelo se entrena K veces, utilizando $K-1$ folds para entrenamiento y 1 fold para validación. El rendimiento final es el promedio de las K iteraciones. Ventaja: Reduce la dependencia de una partición específica y proporciona una estimación más robusta del rendimiento.

Leave-One-Out Caso extremo de K-Fold, donde $K = N$ (número de muestras). En cada iteración, se deja una muestra fuera para validación y se entrena con las $N-1$ restantes. Ventaja: Útil cuando el conjunto de datos es muy pequeño. Desventaja: Computacionalmente costoso y puede estar sesgado si hay outliers.

III.2. Teoría de decisión

En un problema de aprendizaje automático, trabajamos con datos observados (atributos) y etiquetas de clase. Por ejemplo:

- **Atributos**: una radiografía de un paciente.
- **Etiquetas de clase**: si el paciente está sano o enfermo.

El objetivo es realizar **predicciones**: dado un nuevo paciente, asignarle una etiqueta (enfermo o sano) basándose en sus atributos. Para ello, asumimos que los datos observados se generan a partir de una **distribución de probabilidad conjunta** sobre los atributos y las etiquetas de clase. Una vez observados los atributos, utilizamos esta distribución para tomar decisiones sobre la asignación de etiquetas.

A partir de los datos observados, **inferimos** la distribución de probabilidad conjunta. Utilizamos la distribución inferida para asignar etiquetas a nuevos datos.

III.2.1. Probabilidad condicional y regla de Bayes

Para asignar una etiqueta a un nuevo dato, nos interesa la probabilidad condicional de la etiqueta dado los atributos observados, $p(C|x)$, donde:

- C: etiqueta de clase (por ejemplo, enfermo o sano).
- x: atributos observados (por ejemplo, radiografía).

La **regla de Bayes** permite calcular esta probabilidad condicional:

$$p(C|x) = \frac{p(x|C) \cdot p(C)}{p(x)}$$

donde

- $p(x|C)$: probabilidad de observar los atributos x dada la clase C (verosimilitud).
- $p(C)$: probabilidad a priori de la clase C (por ejemplo, la fracción de la población que está enferma).
- $p(x)$: probabilidad marginal de los atributos x (constante de normalización).

Entre las reglas de probabilidad hay dos fundamentales:

- **Regla del producto**: la probabilidad conjunta de x y C es el producto de la verosimilitud y la probabilidad a priori.

$$p(x, C) = p(x|C) \cdot p(C)$$

- **Regla de la suma**: la probabilidad marginal de x se obtiene sumando (marginalizando) sobre todas las clases.

$$p(x) = p(x, C_1) + p(x, C_2)$$

III.2.1.1. Ejemplo de probabilidades condicionales

Consideremos un ejemplo con dos atributos: color de ojos y color de pelo. La siguiente tabla muestra las frecuencias observadas:

		Pelo	
		Rojo	Rubio
Ojos	Marrón	5	10
	Azul	10	20

A partir de esta tabla, podemos calcular varias probabilidades

- Probabilidad marginal:

$$p(\text{ojos marrones}) = \frac{15}{45}$$

- Probabilidad conjunta:

$$p(\text{ojos marrones, pelo rubio}) = \frac{10}{45}$$

- Probabilidad condicional:

$$p(\text{ojos marrones}|\text{pelo rubio}) = \frac{10}{30}$$

III.2.1.2. Probabilidad posterior para dos clases

En un problema con dos clases (C_1 y C_2) se cumple que:

$$p(C_1|x) + p(C_2|x) = 1$$

Esto se debe a que, según la regla de Bayes:

$$p(C_1|x) = \frac{p(x|C_1)p(C_1}{p(x)}$$

$$p(C_2|x) = \frac{p(x|C_2)p(C_2}{p(x)}$$

Sumando ambas probabilidades:

$$p(C_1|x) + p(C_2|x) = \frac{p(x|C_1) \cdot p(C_1) + p(x|C_2) \cdot p(C_2)}{p(x)} = \frac{p(x)}{p(x)} = 1$$

III.2.1.3. Ejercicios teorema de Bayes

Estamos trabajando para una empresa que ha desarrollado una nueva prueba para detectar una enfermedad. La prueba se caracteriza por tener una alta probabilidad de dar positivo en personas enfermas. Sin embargo, a veces puede fallar como se indica en la siguiente tabla:

Test output	Is the person sick?	Probability
+	Sick	0.9
-	Sick	0.1
+	Healthy	0.05
-	Healthy	0.95

Sean las dos etiquetas de clase $\mathcal{C}_0 \equiv$ Healty y $\mathcal{C}_1 \equiv$ Sick y x el resultado de la prueba, es decir, las variables observadas. Por lo tanto, la tabla anterior proporciona todos los valores potenciales para las distribuciones condicionales de clase $p(x|\mathcal{C}_k)$.

Escenario 1 Supongamos que tenemos la misma probabilidad a priori de que una persona esté enferma o sana, es decir, $p(\mathcal{C}_1) = p(\mathcal{C}_0) = 0,5$.

Utiliza el teorema de Bayes para calcular la fracción de personas que están realmente enfermas a partir del número total de personas que son positivas según la prueba. Es decir, se pide calcular $p(\mathcal{C}_1|x = +)$.

$$p(sick|+) = \frac{p(+|sick) \cdot p(sick)}{p(+)} =$$

$$\frac{p(+|sick)p(sick)}{p(+, sick) + p(+, healthy)} = \frac{p(+, sick)p(sick)}{p(+|sick)p(sick) + p(+|healthy)p(healthy)}$$

Ahora rellenamos con los datos del enunciado y la tabla

$$\frac{0.9 \cdot 0.5}{0.9 \cdot 0.5 + 0.05 \cdot 0.5} = 0.95$$

Según los resultados obtenidos, ¿es muy buena la prueba desarrollada? ¿Por qué? Parece un buen test, ya que si es positivo, con un 95 % de probabilidad la persona realmente está enferma.

Escenario 2 Supongamos ahora que la enfermedad es muy rara y que sólo afecta al 1 % de la población. Es decir, $p(\mathcal{C}_1) = 0,01$.

Utilice el teorema de Bayes para calcular la fracción de personas que están realmente enfermas a partir del número total de personas que son positivas según la prueba. Es decir, se le pide que calcule $p(\mathcal{C}_1|x = +)$.

$$p(sick|+) = \frac{p(+|sick) \cdot p(sick)}{p(+)} =$$

$$\frac{p(+|sick)p(sick)}{p(+, sick) + p(+, healthy)} = \frac{p(+, sick)p(sick)}{p(+|sick)p(sick) + p(+|healthy)p(healthy)}$$

Ahora rellenamos con los datos del enunciado y la tabla:

$$\frac{0.9 \cdot 0.01}{0.9 \cdot 0.01 + 0.05 \cdot 0.99} = 0.15$$

Según los resultados obtenidos, ¿es muy buena la prueba desarrollada? ¿Por qué? ¿Qué es lo que falla en la prueba en este escenario?

Este test es malo, ya que con un test positivo, la probabilidad de estar enfermo es bastante baja. El problema está en los falsos positivos. Para mejorar el test, hay que reducir la probabilidad de los falsos positivos.

Calcula cuál es la tasa de falsos positivos necesaria, es decir, $p(x = +|\mathcal{C}_0)$*, de la prueba para que al menos el 90 % de las personas que den positivo en la prueba estén realmente enfermas.*

Ahora queremos calcular $p(+|healthy)$: Sabemos que

$$p(sick|+) = \frac{p(+|sick) \cdot p(sick)}{p(+)} = 0.9$$

y que:

$$p(+) = p(+, sick) + p(+, healthy)$$

Esto es igual a :

$$p(+) = p(+|sick)p(sick) + p(+|healthy)p(healthy)$$

Sustituyendo en función de la tabla del enunciado:

$$p(+) = 0.9 \cdot 0.01 + p(+|healthy) \cdot 0.99$$

Insertando esto en la primera fórmula:

$$0.9 = \frac{p(+|healthy) \cdot p(sick)}{p(+)} = \frac{0.9 \cdot 0.01}{0.9 \cdot 0.01 + p(+|healthy) \cdot 0.99}$$

Se puede pasar la parte de abajo de la fracción al otro lado:

$$0.9 \cdot 0.9 \cdot 0.01 + 0.9 \cdot p(+|healthy) \cdot 0.99 = 0.9 \cdot 0.01$$

Simplificando

$$0.9 \cdot p(+|healthy) \cdot 0.99 = 0.9 \cdot 0.01 - 0.9 \cdot 0.9 \cdot 0.01$$

$$p(+|healthy) = \frac{0.9 \cdot 0.01 - 0.9 \cdot 0.9 \cdot 0.01}{0.9 \cdot 0.99} = 10^{-3}$$

Escenario 3 *Supongamos ahora que disponemos de T pruebas independientes para diagnosticar la enfermedad, pero que tienen las mismas características que las descritas en la tabla anterior. Como las pruebas son independientes, tenemos que*

$$p(x_1 = +, x_2 = +, \ldots, x_T = +|\mathcal{C}_k) = \prod_{i=1}^{T} p(x_i = +|\mathcal{C}_k)$$

¿En cuántas de estas pruebas tiene que dar positivo una persona para que al menos el 90 % de las personas diagnosticadas como positivas estén realmente enfermas?

$$p(sick|x_1 = +, ..., x_T = +) = 0.9$$

Se busca T. Tenemos la siguiente pista: $log(a^b) = b \cdot log(a)$ y $log(ab) = log(a) + log(b)$

$$p(sick|x_1 = +, ..., x_T = +) = p(+|sick)^T = 0.9^T$$

$$p(sick|x_1 = +, ..., x_T = +) = 0.9$$

$$0.9 = \frac{p(x_1 = +, ..., x_T = +|sick) \cdot p(sick)}{p(x_1 = +, ..., x_T = +|sick) \cdot p(sick) + p(x_1 = +, ..., x_T = +|healthy) \cdot p(healthy)}$$

$$p(healthy|x_1 = +, ..., x_T = +) = p(+|healthy)^T = 0.05^T$$

$$0.9 = \frac{0.9^T \cdot 0.01}{0.9^T \cdot 0.01 + 0.05^T \cdot 0.99}$$
$$0.9 \cdot 0.9^T \cdot 0.01 + 0.9 \cdot 0.05^T \cdot 0.99 = 0.9^T \cdot 0.01$$
$$0.9 \cdot 0.05^T \cdot 0.99 = 0.9^T \cdot 0.01 - 0.9 \cdot 0.9^T \cdot 0.01$$
$$0.9 \cdot 0.05^T \cdot 0.99 = 0.9^T [0.01 - 0.9 \cdot 0.01]$$

Aplicando los logaritmos:

$$log0.9 + T \cdot log0.05 + log0.99 = T \cdot log0.9 + log(0.01 - 0.9 \cdot 0.01)$$
$$Tlog0.05 - Tlog0.9 = log(0.01 - 0.9 \cdot 0.01) - log0.9 - log0.99$$
$$T(log0.05 - log0.9) = \frac{log(0.01 - 0.9 \cdot 0.01) - log0.9 - log0.99}{log0.05 log0.9} = 2.3$$

III.2.2. Minimizar el porcentaje de errores de clasificación

Una regla de clasificación dividirá el espacio de entrada (de atributos x) en regiones $\mathcal{R}_k$. Todos los $\mathbf{x} \in \mathcal{R}_k$ se asignan a $\mathcal{C}_k$. Los límites se denominan superficies de decisión o fronteras de clasificación. La probabilidad de error es:

$$\begin{aligned} p(mistake) &= p(\mathbf{x} \in \mathcal{R}_1, \mathcal{C}_2) + p(\mathbf{x} \in \mathcal{R}_2, \mathcal{C}_1) \\ &= \int_{\mathcal{R}_1} p(\mathbf{x}, \mathcal{C}_2) d\mathbf{x} + \int_{\mathcal{R}_2} p(\mathbf{x}, \mathcal{C}_1) d\mathbf{x} \end{aligned}$$

Si el ejemplo cae en la región 1, pero pertenece a la clase 2, se produce un error de clasificación. Lo mismo al contrario: si un dato cae en la región 2 y realmente viene de la clase 1, también se produce un error de clasificación. El error que se comete es el área debajo de las funciones de arriba. Para minimizar la probabilidad de error, hay que construir la siguiente regla de predicción:

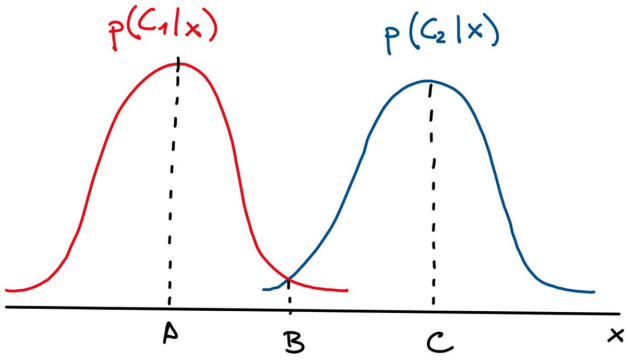
$$\pi(\mathbf{x}) = \begin{cases} \mathcal{C}_1 if p(\mathbf{x}, \mathcal{C}_1) \geq p(\mathbf{x}, \mathcal{C}_2) \\ \mathcal{C}_2 if p(\mathbf{x}, \mathcal{C}_2) \geq p(\mathbf{x}, \mathcal{C}_1) \end{cases}$$

Esta regla es justo la que predice la clase que tiene la mayor probabilidad posterior $p(\mathcal{C}_k|\mathbf{x})$.

Para minimizar el error de predicción y conseguir el clasificador con menor error, se debe predecir la etiqueta de clase con mayor probabilidad posterior. A este clasificador se le conoce como **clasificador de Bayes** al utilizar el teorema de Bayes.

El problema es que las probabilidades posteriores no se conocen, y se deben inferir de los datos. Se deben utilizar distintos métodos para estimar estas probabilidades posteriores.

Consideramos que tenemos un problema de clasificación en cuyo eje x se encuentra el espacio posible de los datos. Hay dos distribuciones posibles: $p(\mathcal{C}_1|x)$ y $p(\mathcal{C}_2|x)$. Tenemos tres puntos: A, B y C. ¿Dónde tendríamos que poner la frontera de clasificación? Respuesta: es donde se cortan las probabilidades posteriores, que en este caso es el punto B.



III.2.3. Minimizar la pérdida esperada

A veces, tomar decisiones equivocadas puede tener costes asimétricos.

- Paciente que no tiene cáncer al que se le diagnostica cáncer: estrés
- Paciente con cáncer diagnosticado que no tiene cáncer: muerte prematura

Estas cuestiones pueden formalizarse mediante una matriz de pérdidas L:

$$L_{kj} \equiv \text{Cost of assigning } \mathcal{C}_j \text{ to } \mathbf{x} \text{ with true class } \mathcal{C}_k$$

$$\mathbf{L} = \begin{pmatrix} 0 & 1 \\ 10 & 0 \end{pmatrix}$$

Las filas contienen la clase verdadera, y las columnas representan la clase etiquetada. La diagonal tiene 0, ya que es donde estaríamos acertando (se predice la etiqueta de clase correcta). Como en ese caso no se comete ningún error, se pone 0, y los elementos fuera de la diagonal indican los errores que se cometen. En este caso, tiene más peso que a un paciente con cáncer se le etiquete como no cáncer.

Nuestro objetivo es minimizar la pérdida esperada:

$$\mathbb{E}[loss] = \sum_k \sum_j L_{kj} p(\mathbf{x} \in \mathcal{R}_j, \mathcal{C}_k) = \sum_k \sum_j L_{kj} \int_{\mathcal{R}_j} p(\mathbf{x}, \mathcal{C}_k) d\mathbf{x}$$

Debemos elegir $\mathcal{R}_j$ para minimizar $\sum_k L_{kj} p(\mathbf{x}, \mathcal{C}_k)$. La regla de decisión es asignar $\mathbf{x}$ a $\mathcal{C}_j$ para el que $\sum_k L_{kj} p(\mathcal{C}_k|\mathbf{x})$ sea el mínimo.

Si tenemos acceso a las probabilidades posteriores, podemos tomar decisiones de manera óptima y tener en cuenta posibles asimetrías de los errores. Si tuviéramos el mismo peso para los errores, la frontera de clasificación esperada sería la calculada anteriormente (0.5).

III.2.4. Opción de rechazo

A veces es mejor evitar las decisiones sobre los casos difíciles, es decir, regiones en las que la mayor $p(\mathcal{C}_k|\mathbf{x})$ es significativamente menor que 1. En el caso de las radiografías, dejamos que un humano clasifique los casos más ambiguos.

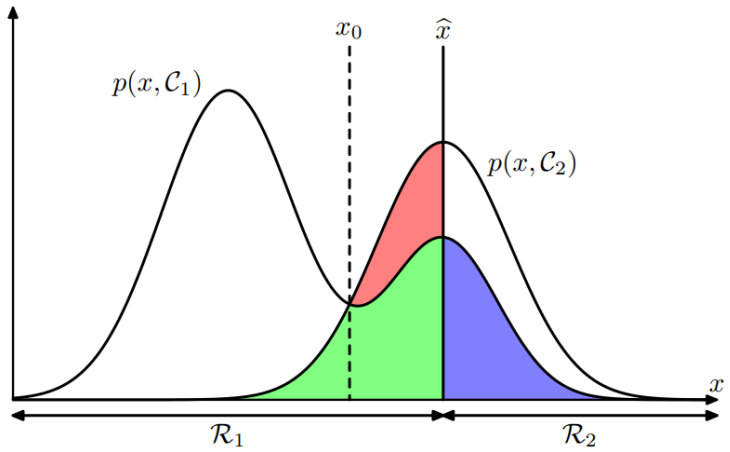


Figura III.3: *Imaginamos que tenemos un problema de clasificación de una dimensión. Tenemos dos clases: la positiva y la negativa. Queremos especificar un clasificador para resolver el problema. Definimos las regiones $\mathcal{R}_1$ y $\mathcal{R}_2$ para las clases $\mathcal{C}_1$ y $\mathcal{C}_2$ respectivamente. Para un dato que se observe en la región 1, pero pertenezca a la clase 2, se debe calcular el área debajo de la curva representado en rojo y verde. Si x cae en la región 2 y realmente viene de la clase 1, el error se calcula como el área azul. El clasificador no es óptimo, ya que la frontera de clasificación no es muy correcta. Si se cambiase al punto marcado con las líneas discontinuas, el error se vería minimizado. Con el cambio de frontera, si x viene de la clase 2 y cae en la región 1, el error sería el área verde que se sale de la frontera. Si x cae en la región 2 y pertenece a la clase 1, el error es el área verde de la derecha sumado al área azul. Generalmente, el clasificador óptimo se encuentra cuando las probabilidades posteriores de clase tomen el mismo valor. En la línea discontinua se cortan las dos curvas de distribución de ambas clases. Las probabilidades posteriores de clase es igual a la probabilidad conjunta dividida por una constante. Si las probabilidades conjuntas toman el mismo valor, las probabilidades posteriores también son iguales. En ese punto donde las probabilidades posteriores valen 0.5 se encuentra la frontera de clasificación.*

Se introduce un umbral θ y se rechazan todos aquellos datos $\mathbf{x}$ para los cuales $max(p(\mathcal{C}_k|\mathbf{x}) < \theta$. Si $\theta = 1$, todos los ejemplos se rechazan. Si $\theta < 1/K$, ningún ejemplo se rechaza.

III.2.5. Inferencia y decisión

Hay dos procesos diferentes:

- Inferencia: utiliza datos de entrenamiento para estimar las probabilidades posteriores de clase, es decir, modelar $p(\mathcal{C}_k|\mathbf{x})$
- Decisión: utilizar las probabilidades posteriores para hacer asignaciones óptimas de clase

Generalmente, los distintos clasificadores son distintos mecanismos para estimar las probabilidades posteriores. Hay tres tipos de clasificadores en función de su complejidad. Por orden descendente, son:

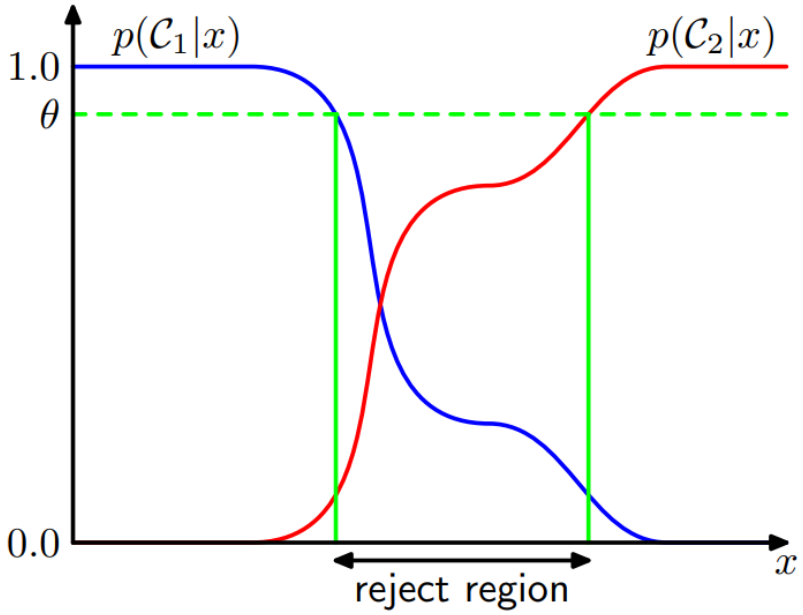


Figura III.4: *La región de rechazo se encuentra alrededor de la frontera de decisión.*

- Modelos generativos: coge los ejemplos de cada clase y estima las distribuciones condicionales (de la clase positiva y de la clase negativa). Una vez estimadas, y sabiendo la probabilidad a priori de cada clase en base a los datos de entrenamiento, se utiliza la regla de Bayes para clasificar los datos nuevos. El problema es que $\mathbf{x}$ puede ser muy dimensional (sobre todo en biología y salud, como muchos genes u otros atributos) y es posible que necesitemos muchos datos para determinar las densidades con una precisión razonable. Esto se debe a la maldición de la dimensionalidad. Conforme haya más dimensiones, el espacio está cada vez más vacío, por lo que es más difícil estimar la densidad. Permite generar nuevos datos. No obstante, estima p(x), lo que resulta útil para detectar valores atípicos. Es caro si sólo queremos decisiones de clasificación.

- Modelos discriminativos: se estima directamente las probabilidades posteriores de clase. Tiene un enfoque mucho más sencillo que el anterior al no tener que estimar una probabilidad posterior en múltiples dimensiones. Las densidades condicionales de clase pueden contener mucha estructura que tiene poco efecto en las probabilidades posteriores.

- Discriminantes: se estima una función que mapea $\mathbf{x}$ al espacio de etiquetas. Su enfoque es aún más sencillo, combinando el paso de inferencia y decisión en uno solo. La pega es que ya no tenemos acceso a $p(\mathcal{C}_k|\mathbf{x})$, por lo que la minimización de la función de pérdida y la opción de rechazo no son posibles.

Las densidades condicionales de clase complicadas pueden dar lugar a probabilidades posteriores de clase simples.

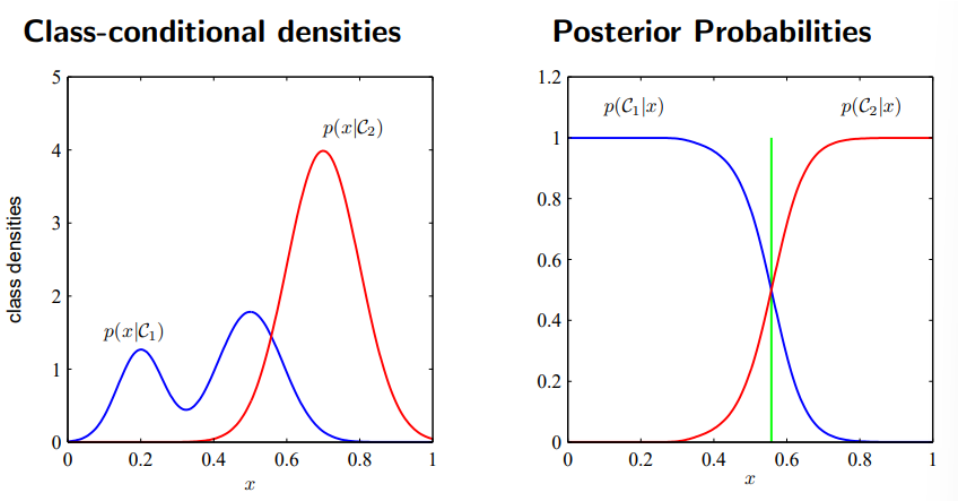


Figura III.5: *Suponemos que tenemos una dimensión con una clase positiva y una clase negativa. La distribución es complicada al tener dos modas. Si esto se utiliza en la regla de Bayes, se pueden calcular las probabilidades posteriores, las cuales tienen menos complejidad que las densidades. Generalmente, las probabilidades posteriores son funciones relativamente sencillas y más fáciles de calcular que las distribuciones condicionales.*

III.2.6. Máxima verosimilitud

Cuando un clasificador era un modelo generativo, se estiman las probabilidades a priori. La clave del uso de modelos generativos para la clasificación es especificar una forma adecuada para las densidades condicionales de clase. Elegiremos una distribución con parámetros θ. La tarea consiste en estimar el mejor valor de θ a partir de los datos observados.

Supongamos que $x \in [1, 2, 3, \ldots, 100]$ y consideramos dos posibles distribuciones:

$$\theta_{two} \equiv \text{distribución uniforme entre las potencias de dos}$$

$$\theta_{even} \equiv \text{distribución uniforme entre los números pares}$$

Observamos los valores de x en $\mathcal{D} = [16, 8, 2, 64]$. Estos números son pares y también se pueden generar por potencias de dos. ¿Qué distribución se elegiría? La de potencias de dos, ya que es más probable que esa sea la distribución verdadera.

III.2.6.1. Principio de máxima verosimilitud

Los datos se han generado independientemente. La observación de los datos para cada distribución es:

$$p(\mathcal{D}|\theta_{two}) = \prod_{i=1}^{N} p(x_i|\theta_{two}) = (1/6)^N = 7.7 \cdot 10^{-4}$$

$$p(\mathcal{D}|\theta_{even}) = \prod_{i=1}^{N} p(x_i|\theta_{even}) = (1/50)^N = 1.6 \cdot 10^{-7}$$

La probabilidad es mayor en la distribución de las potencias de dos. El principio de máxima verosimilitud elige el modelo que maximice la probabilidad de los datos observados.

$$\hat{\theta}_{ML} = argmax_\theta \qquad p(\mathcal{D}|\theta) = \prod_{i=1}^{N} p(\mathbf{x}_i|\theta)$$
$$= argmax_\theta \qquad p(\mathcal{D}|\theta) = \sum_{i=1}^{N} p(\mathbf{x}_i|\theta)$$

Para estimar los parámetros, se utiliza la máxima verosimilitud, eligiendo la media y varianza para ello. Los valores de los parámetros que maximizan la probabilidad de los datos es mediante los estimadores de máxima verosimilitud, que es la media empírica y la varianza empírica.

Estos estimadores son asintóticamente insesgados. Convergen a los valores reales a medida que tenemos más y más datos.

Ejemplo: suponemos que los datos observados han surgido de una distribución gaussiana de la cual no conocemos la media ni la varianza. Hemos observado los datos de x1 a xn. Se puede escribir la probabilidad de los datos observados dado la media y la varianza:

$$p(\mathcal{D}|\mu, \sigma^2) = \prod_{i=1}^{N} p(x_i|\mu, \sigma^2)$$

Esto se puede convertir en:

$$\log p(\mathcal{D}|\mu, \sigma^2) = \sum_{i=1}^{N} p(x_i|\mu, \sigma^2)$$
$$= \sum_{i=1}^{N} -0.5 \log 2\pi\sigma^2 - \frac{1}{2 \cdot \sigma^2}(x_i - \mu)^2$$

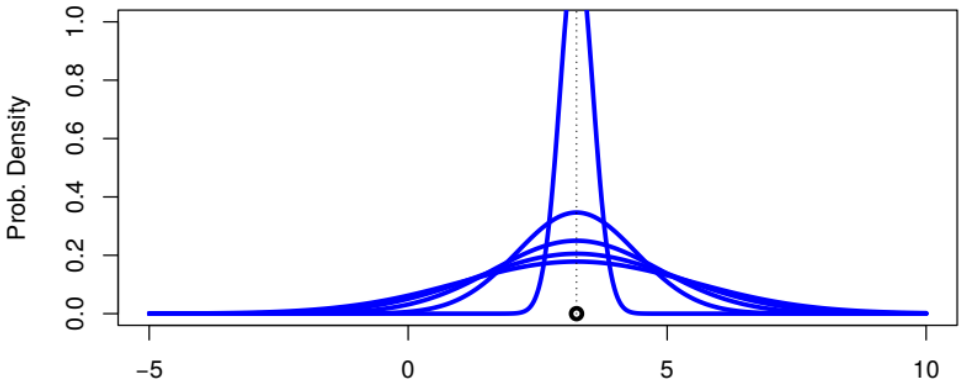
$$\log p(D|\mu) = \sum_{i=1}^{N} -\frac{1}{2\sigma^2}\frac{(x_i - \mu)^2}{\sigma^2} + cte$$

$$\frac{\delta \log p(D|\mu)}{\delta \mu} = \sum_{i=1}^{N} \frac{-2(x_i - \mu)}{2\sigma^2} \cdot -1 + 0 = 0$$
$$= \sigma_{i=1}^{N} \frac{x_i \mu}{\sigma^2} = 0$$
$$= \sum_{i=1}^{N} (x_i) - \mu) = 0$$
$$= \sum_{i=1}^{N} (x_i) - N\mu = 0$$
$$= \sum_{i=1}^{N} x_i = N\mu$$

$$\mu = \frac{1}{N}\sum_{i=1}^{N} x_i$$

En múltiples dimensiones, en lugar de tener una varianza, hay una matriz de covarianzas. Esto se puede ver mejor en el notebook de MaximumLikelihoodGaussian (carpeta ejercicios).

Sobreajuste El principio de máxima verosimilitud puede llevar a un **sobreajuste severo** cuando hay pocos datos. El sobreajuste $p(x|\hat{\theta}_{ML}$ captura propiedades de los datos que no generaliza bien. Esto se traduce en que la probabilidad para los datos de entrenamiento es muy alta, pero para datos nuevos muy baja. Por ejemplo, si en una distribución gausiana solo tenemos un valor observado, el estimador de la media que maximiza la probabilidad es el mismo. La media se pondría donde está la observación, y la varianza sería 0, ya que la diferencia entre la media y nuestro dato es 0. Esto se traduce en que la gaussiana es muy picuda y se centra en el dato observado. Si la varianza es 0, la probabilidad de los datos observados es infinita (siendo esto lo mejor que hay). Esto parece muy bueno según el principio de máxima verosimilitud. No obstante, para un dato nuevo, la densidad de probabilidad de un dato nuevo, diferente al dato observado, sería 0.



III.2.6.2. Ejercicios

1. Un problema de clasificación binaria tiene probabilidades condicionales de clase:

$$p(\mathcal{C}_1|x) = \frac{1}{1+exp(-wx)} \qquad p(\mathcal{C}_2|x) = \frac{1}{1+exp(wx)}$$

siendo w algún parámetro que puede valer cualquier número excepto 0. El objetivo está en encontrar la frontera de decisión óptima.

$$\begin{aligned} \frac{1}{1+exp(-wx)} &= \frac{1}{1+exp(wx)} && |\text{invertir fracción} \\ 1+exp(-wx) &= 1+exp(wx) && |-1 \\ exp(-wx) &= exp(wx) && |\log \\ -wx &= wx \end{aligned}$$

$$wx + wx = 0$$
$$2wx = 0$$
$$x = 0$$

2. Dadas las probabilidades anteriores, sólo queremos tomar decisiones con un 90 % de confianza. Encuentra la región de rechazo correspondiente.

3. La función de masa de probabilidad de una variable aleatoria Bernoulli es:

$$p(x \mid p) = p^x(1-p)^{1-x} \text{with} x \in [0,1]$$

Halla el estimador de máxima verosimilitud del parámetro $p \in [0,1]$. Pista: maximizar el logaritmo de la probabilidad en lugar de la probabilidad.

$$p(D \mid p) = \prod_{i=1}^{N} p(x_i \mid p) = \prod_{i=1}^{N} p^{xi}(1-p)^{1-xi}$$
$$\log p(D \mid p) = \sum_{i=1}^{N} \log p(x_i \mid p) = \sum_{i=1}^{N} x_i \log p + (1-x_i)\log(1-p)$$

Queremos obtener el valor máximo. Para ello, derivamos e igualamos la derivada a 0.

$$\frac{\delta \log p(D \mid p)}{\delta p} = \sum_{i=1}^{N} x_i \frac{1}{p} + (1-x_i)\frac{1}{1-p} \cdot (-1) = 0$$
$$\frac{\delta \log p(D \mid p)}{\delta p} = \sum_{i=1}^{N} \frac{N_1}{p} - \frac{N_0}{1-p} = 0$$
$$\frac{N_1}{p} = \frac{N_0}{1-p}$$

Despejamos de ahí el valor de p:

$$\frac{p}{N_1} = \frac{1-p}{N_0} = \frac{1}{N_0} - \frac{p}{N_0}$$
$$\frac{p}{N_1} + \frac{p}{N_0} = \frac{1}{N_0}$$
$$p(\frac{1}{N_1} + \frac{1}{N_0}) = \frac{1}{N_0}$$
$$p = \frac{1}{N_0} \cdot [\frac{1}{N_1 + \frac{1}{N_0}}]^{-1} = \frac{1}{N_0} \cdot [\frac{N_0 + N_1}{N_1 N_0}]^{-1} = \frac{1}{N_0} \cdot [\frac{N}{N_1 N_0}]^{-1} = \frac{1}{N_1}\frac{N_1 N_0}{N} = \frac{N_1}{N}$$

III.2.7. Conocimiento a priori

Se puede utilizar una probabilidad a priori para asignar una probabilidad baja a conceptos poco naturales que no se esperan en la práctica y viceversa. La probabilidad a priori es subjetiva y refleja cualquier información adicional que podamos tener sobre el concepto desconocido que intentamos inferir a partir de los datos. Consideremos las distribuciones potenciales:

$$\begin{aligned}
\theta_{two} &\equiv \text{distribución uniforme entre las potencias de dos} \\
\theta_{even} &\equiv \text{distribución uniforme entre los números pares} \\
\theta_{odd} &\equiv \text{distribución uniforme entre los números impares} \\
\theta_{prime} &\equiv \text{distribución uniforme entre números primos} \\
\theta_{9} &\equiv \text{distribución uniforme entre números terminados en 9} \\
\theta_{two^{\setminus 32}} &\equiv \text{distribución uniforme entre las potencias de dos excepto 32} \\
\theta_{two}^{\cup 37} &\equiv \text{distribución uniforme entre las potencias de dos más 37}
\end{aligned}$$

Introducimos una a priori uniforme excepto para θ_{odd} y θ_{even} que son más probables a priori y $\theta_{two^{\setminus 32}}$ y $theta_{two}^{\cup 37}$, que son menos probables.

III.2.8. Conocimiento a posterior

La posterior es la probabilidad multiplicada por la anterior, normalizada:

$$p(\theta \mid \mathcal{D}) = \frac{p(\mathcal{D} \mid \theta)p(\theta)}{p(\mathcal{D})} = \frac{p(\mathcal{D} \mid \theta)p(\theta)}{\sum_{\theta \in \mathcal{H}} p(\mathcal{D}, h)}$$

En la posterior se combinan la verosimilitud y la anterior:

- Probabilidad: favorece los conceptos explicados por los datos.
- Prior: los conceptos no naturales reciben una importancia menor.

El prior resume nuestras creencias sobre cada concepto antes de ver los datos y el posterior actualiza esas creencias después de ver los datos. Utilizamos probabilidades para asignar grados de creencia a cada concepto potencial que intentamos aprender de los datos.

III.2.9. Estimación máxima a posteriori (MAP)

Cuando tenemos suficientes datos, la estimación a posteriori $p(\theta \mid \mathcal{D})$ alcanza su punto máximo en un único concepto, la estimación MAP.

$$p(\theta \mid \mathcal{D}) \to \delta_{\hat{\theta}_{MAP}}(\theta)$$

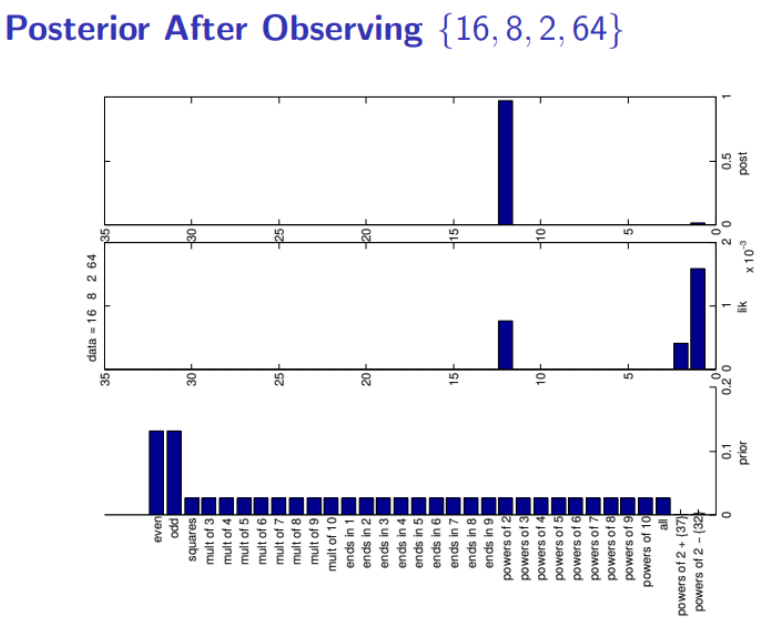


Figura III.6: *Habiendo observado los valores 16, 8, 2 y 64, según la máxima verosimilitud, aceptaríamos la distribución de números de potencias de dos salvo el 32. Como esa distribución es muy rara, utilizamos el conocimiento prior y multiplicamos eso con la máxima verosimilitud, normalizando. De esa forma, la distribución con la que nos quedamos (la más probable) es la distribución de las potencias de 2.*

III.2.9.1. Media de la distribución gaussiana

Se considera un conjunto de datos $\mathcal{D} = \{x_1, \ldots, x_N\}$ bajo la hipótesis de que los datos siguen una distribución normal:

$$x_i \sim \mathcal{N}(\mu, \sigma^2)$$

Además, se asume un conocimiento previo sobre μ, modelado mediante una distribución normal:

$$p(\mu) = \mathcal{N}(\mu \mid 5, 1)$$

Nuestro objetivo es encontrar la estimación MAP (Máxima A Posteriori) de μ, es decir, el valor de μ que maximiza la distribución posterior $p(\mu \mid \mathcal{D})$.

La distribución posterior está dada por:

$$\log p(\mu \mid \mathcal{D}) = \log p(\mathcal{D} \mid \mu) + \log p(\mu) + \text{constante}$$

donde:

- $p(\mathcal{D} \mid \mu)$ es la verosimilitud de los datos dada μ.
- $p(\mu)$ es la distribución a priori de μ.

La función de verosimilitud para una normal es:

$$\log p(\mathcal{D} \mid \mu) = \sum_{i=1}^{N} -\frac{(x_i - \mu)^2}{2\sigma^2} + \text{constante}$$

Para la distribución previa:

$$\log p(\mu) = -\frac{1}{2}(\mu - 5)^2 + \text{constante}$$

Al combinar ambas ecuaciones y maximizar con respecto a μ, se obtiene el estimador MAP:

$$\hat{\mu}_{MAP} = \hat{\mu}_{ML}\frac{N}{N+\sigma^2} + 5\frac{\sigma^2}{N+\sigma^2}$$

donde $\hat{\mu}_{ML}$ es el estimador de máxima verosimilitud (ML), que corresponde a la media muestral.

Si N es grande ($N \gg \sigma^2$), la estimación MAP se aproxima a la media muestral $\hat{\mu}_{ML}$. Si N es pequeño ($N \ll \sigma^2$), la estimación MAP está más influenciada por la media de la distribución previa (5 en este caso).

Esto muestra cómo la información previa se combina con los datos observados para obtener una mejor estimación del parámetro μ.

III.2.9.2. Varianza de la distribución gaussiana

Se considera un conjunto de datos $\mathcal{D} = \{x_1, \ldots, x_N\}$ bajo la hipótesis de que los datos siguen una distribución normal:

$$x_i \sim \mathcal{N}(\mu, \sigma^2)$$

Además, se asume un conocimiento previo sobre la varianza σ^2, modelado mediante una distribución a priori:

$$p(\sigma^2) \propto \frac{\exp(-1/\sigma^2)}{\sigma^2}$$

Nuestro objetivo es encontrar la estimación MAP (Máxima A Posteriori) de σ^2, dado el estimador de máxima verosimilitud $\hat{\mu}_{ML}$.

La distribución posterior está dada por:

$$\log p(\sigma^2 \mid \mathcal{D}) = \log p(\mathcal{D} \mid \sigma^2) + \log p(\sigma^2) + \text{constante}$$

donde:

- $p(\mathcal{D} \mid \sigma^2)$ es la verosimilitud de los datos dada σ^2.
- $p(\sigma^2)$ es la distribución a priori de σ^2.

La función de verosimilitud para una normal es:

$$\log p(\mathcal{D} \mid \sigma^2) = \sum_{i=1}^{N} -\frac{(x_i - \hat{\mu}_{ML})^2}{2\sigma^2} - \frac{N}{2}\log\sigma^2 + \text{constante}$$

Para la distribución previa:

$$\log p(\sigma^2) = -\frac{1}{\sigma^2} - \log\sigma^2 + \text{constante}$$

Al combinar ambas ecuaciones y maximizar con respecto a σ^2, se obtiene el estimador MAP:

$$\hat{\sigma}^2_{MAP} = \hat{\sigma}^2_{ML}\frac{N}{N+2} + 1 \cdot \frac{2}{N+2}$$

donde $\hat{\sigma}^2_{ML}$ es el estimador de máxima verosimilitud (ML) de la varianza.

Si N es grande ($N \gg 2$), la estimación MAP se aproxima a la estimación ML $\hat{\sigma}^2_{ML}$. Si N es pequeño, la estimación MAP está más influenciada por la distribución previa, lo que reduce el sobreajuste.

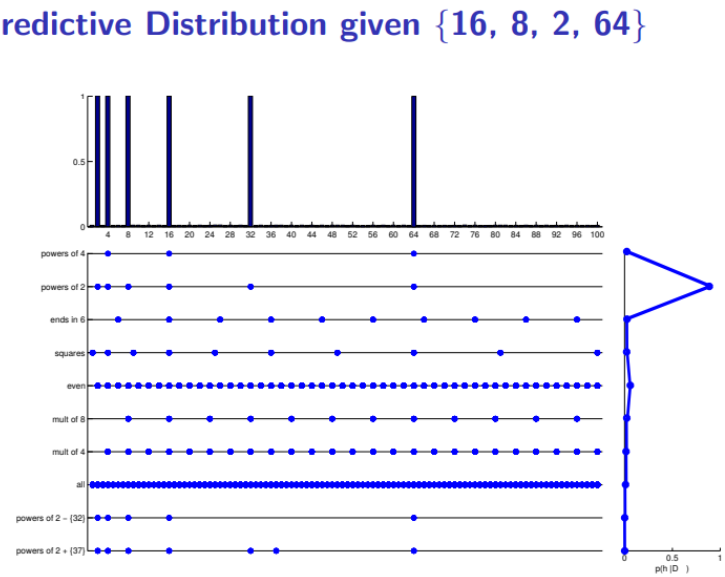
Este resultado muestra cómo la información previa se combina con los datos observados para obtener una mejor estimación del parámetro σ^2.

III.2.10. Aprendizaje bayesiano y distribución predictiva

Cuando tenemos un conjunto de datos pequeño, la posterior es amplia, lo que indica que muchos conceptos potenciales son compatibles con los datos observados. En ese caso, puede ser mejor considerar $p(\theta \mid \mathcal{D})$ en lugar de $\hat{\theta}_{MAP}$.

La distribución predictiva posterior es:

$$p(x \mid \mathcal{D}) = \int p(x \mid \theta)p(\theta \mid \mathcal{D})d\theta$$



III.2.10.1. Selección de modelos bayesianos

La distribución posterior bayesiana también es útil para realizar la selección de modelos sin tener que recurrir a la validación cruzada. Obtenemos la probabilidad posterior de cada modelo utilizando la regla de Bayes:

$$p(m \mid \mathcal{D} = \frac{p(\mathcal{D} \mid m)p(m)}{p(\mathcal{D})} = \frac{p(\mathcal{D} \mid m)p(m)}{\sum_{m \in \mathcal{M}} p(m, \mathcal{D})}$$

Si la probabilidad a priori sobre los modelos es uniforme, simplemente elegimos el modelo con la probabilidad marginal más alta:

$$p(\mathcal{D} \mid m) = \int p(\mathcal{D} \mid \theta) p(\theta \mid m) d\theta$$

Penaliza los modelos demasiado simples o demasiado complejos para los datos.

Las mixturas de gaussianas (GMM, por sus siglas en inglés: Gaussian Mixture Models) son un modelo probabilístico que representa la distribución de datos como una combinación de múltiples distribuciones normales (gaussianas). Se utilizan ampliamente en problemas de agrupamiento (clustering), densidad de probabilidad y reducción de dimensionalidad. La idea principal de un GMM es que los datos provienen de una combinación de varias distribuciones normales. Cada muestra es generada por una de las gaussianas, pero no sabemos cuál. En lugar de asignar cada punto a un único cluster como en k-means, GMM asigna probabilidades a cada punto de pertenecer a diferentes clusters.

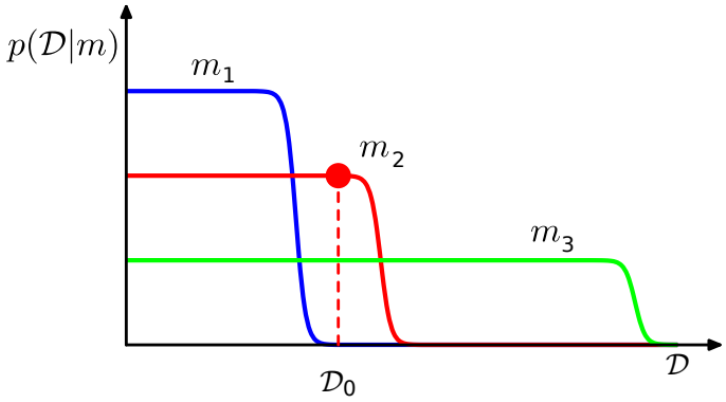


Figura III.7: *m_1 es muy simple para explicar $\mathcal{D}_0$, m_3 es muy complejo, pero m_2 es perfecto.*

III.2.10.2. Ejercicios

1. Considera dos modelos m_1 y m_2 con los siguientes valores asociados:

$$p(\mathcal{D} \mid m_1) = 0,05 \qquad p(\mathcal{D} \mid m_2) = 0,01$$
$$p(m_1) = 0,1 \qquad p(m_2) = 0,9$$

   ¿Cuál es el modelo MAP y cuál el ML?

   El modelo m_1 es el de máxima verosimilitud, ya que tiene una mayor probabilidad de los datos dado el modelo $(0,05 > 0,01)$. En cuanto al modelo MAP, se trata del modelo m_2, ya que tiene su prior favorecido $(0,9 > 0,1)$.

III.2.11. Resumen

- La teoría de la decisión permite realizar asignaciones de clase óptimas en presencia de incertidumbre. Sólo requiere probabilidades de clase posteriores.

- Estas probabilidades pueden estimarse directamente (modelos discriminativos) u obtenerse mediante la regla de Bayes (modelos generativos).
- Las probabilidades estimadas dependerán de varios parámetros. Éstos pueden estimarse utilizando enfoques ML o MAP.
- A diferencia del ML, que puede llevar a un sobreajuste cuando los datos son escasos, el enfoque MAP puede ayudar a reducir el sobreajuste introduciendo conocimiento previo.
- Un enfoque bayesiano en el que se marginen los parámetros puede ser aún más robusto. Sin embargo, los cálculos suelen ser inviables.

III.3. Modelos generativos

Los modelos generativos estiman las densidades condicionales de clase $p(x \mid \mathcal{C}_k)$ para luego utilizar el teorema de Bayes para calcular una probabilidad posterior de clase.

$$p(\mathcal{C}_k \mid \mathbf{x}) = \frac{p(\mathbf{x} \mid \mathcal{C}_k)p(\mathcal{C}_k)}{p(\mathbf{x})}$$

Esto permite generar nuevos datos muestreando de $p(\mathbf{x} \mid \mathcal{C}_k)$.

No obstante, esto tiene varios problemas:

- Estimar $p(\mathcal{C}_k)$ es fácil, pero estimar $p(\mathbf{x} \mid \mathcal{C}_k)$ puede ser complicado.
- El vector $\mathbf{x}$ puede ser multidimensional, aplicando entonces la maldición de la dimensionalidad.
- El vector $\mathbf{x}$ puede tener valores continuos y discretos.
- En los casos discretos, no vamos a ver las instancias C^D.

El clasificador Naive Bayes ofrece una solución razonable a estos problemas.

III.3.1. Clasificador Naive Bayes

La clave está en simplificar las densidades condicionales de clase suponiendo independencia.

$$p(\mathbf{x} \mid \mathcal{C}_k) \approx \prod_{j=1}^{D} p(x_j \mid \theta_{jk})$$

donde θ_{jk} son parámetros.

Los tipos de densidades condicionales de clase en función del tipo de característica son:

- **De valor real:** distribución gaussiana
- **Binaria**: distribución Bernoulli
- **Categórica:** distribución multinoulli

III.3.1.1. Estimación de la prior (probabilidad de clase)

La primera parte trata sobre la probabilidad a priori de cada clase k:

$$p(y_i \mid \pi) = \prod_{k=1}^{K} \pi_k^{I(y_i=k)}$$

Esto significa que la probabilidad de que una muestra pertenezca a una clase k se modela como un producto de probabilidades pi_k, donde solo la clase correcta contribuye (usando la variable indicadora $I(y_i = k)$. Tomando logaritmos:

$$\log p(\mathcal{D} \mid \pi) = \sum_{k=1}^{K} N_k \log \pi_k$$

siendo N_k el número de muestras en la clase k.

La estimación de máxima verosimilitud (MLE) para la probabilidad de la clase es:

$$\hat{\pi}_k^{ML} = \frac{N_k}{N}$$

Es decir, la probabilidad estimada de la clase k es simplemente el número de instancias en esa clase dividido por el total de muestras.

III.3.1.2. Estimación de probabilidades de los atributos (feature probabilities)

Cuando las características son categóricas, el modelo necesita estimar la probabilidad condicional $p(x_j \mid y_k)$, es decir, la probabilidad de observar un valor de un atributo dado que pertenece a la clase k.

La función de verosimilitud para los atributos se modela como:

$$\log p(\mathcal{D}_j^k \mid \mu_{jk}) = \sum_{c=1}^{C} N_{jk}^c \log \mu_{jkc}$$

donde

- N_{jk}^c es el número de veces que el valor c del atributo j aparece en la clase k.
- μ_{jkc} es la probabilidad de que el atributo j tome el valor c en la clase k.

La estimación ML para esta probabilidad es:

$$\hat{\mu}_{jkc}^{ML} = \frac{N_{jk}^c}{N_k}$$

Esto significa que la probabilidad estimada de un atributo específico es simplemente la frecuencia relativa de ese valor dentro de la clase k.

Sólo tenemos que contar las ocurrencias entre las instancias de formación.