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

13 de mayo de 2025 18:08

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

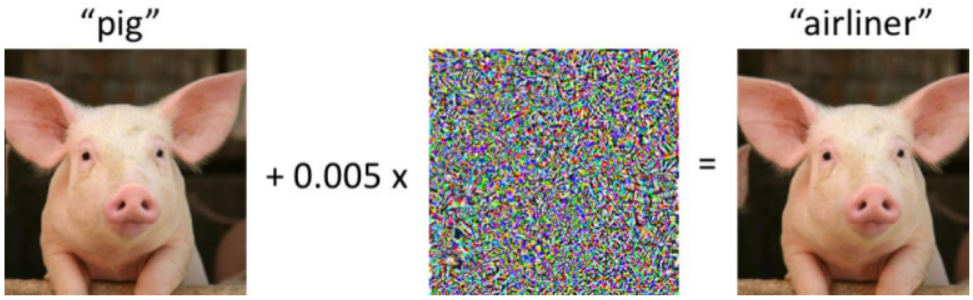


Figura I.1: *Reconocimiento de imágenes con ataque adversario.*

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.

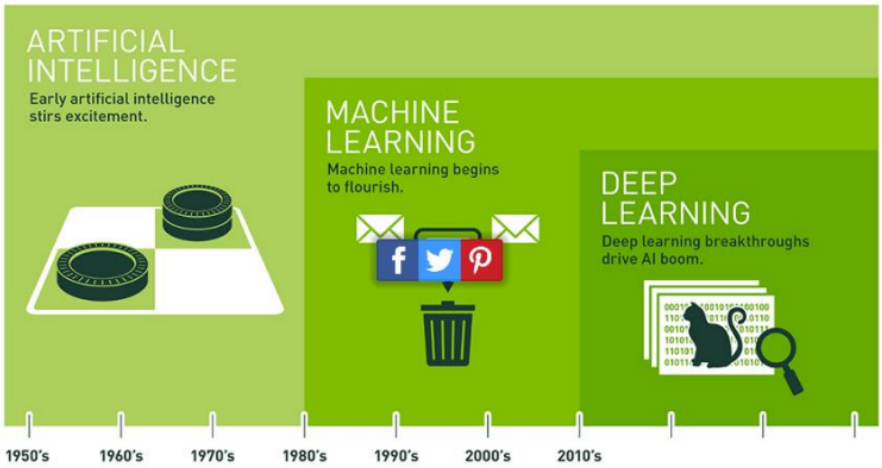


Figura I.2: *Mapa temporal del desarrollo de las inteligencias artificiales. Ya está algo desfasado, faltaría añadir después del Deep Learning los Modelos Generativos.*

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.

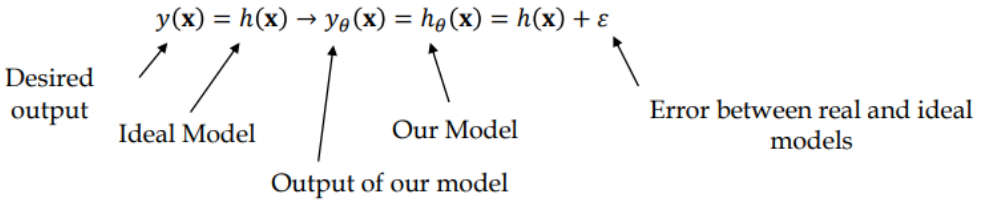
I.1.8.2. Fases de un algoritmo de aprendizaje

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.

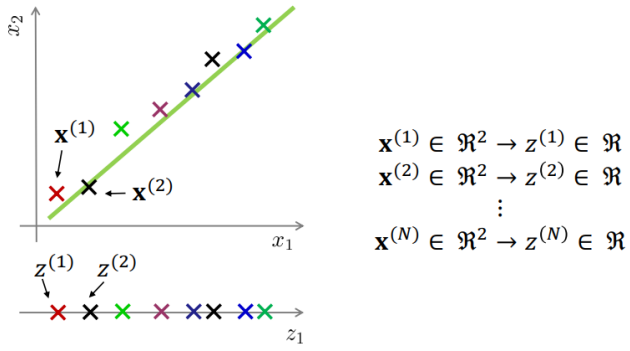
I.2.1. Proyección de datos en dimensiones inferiores

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.

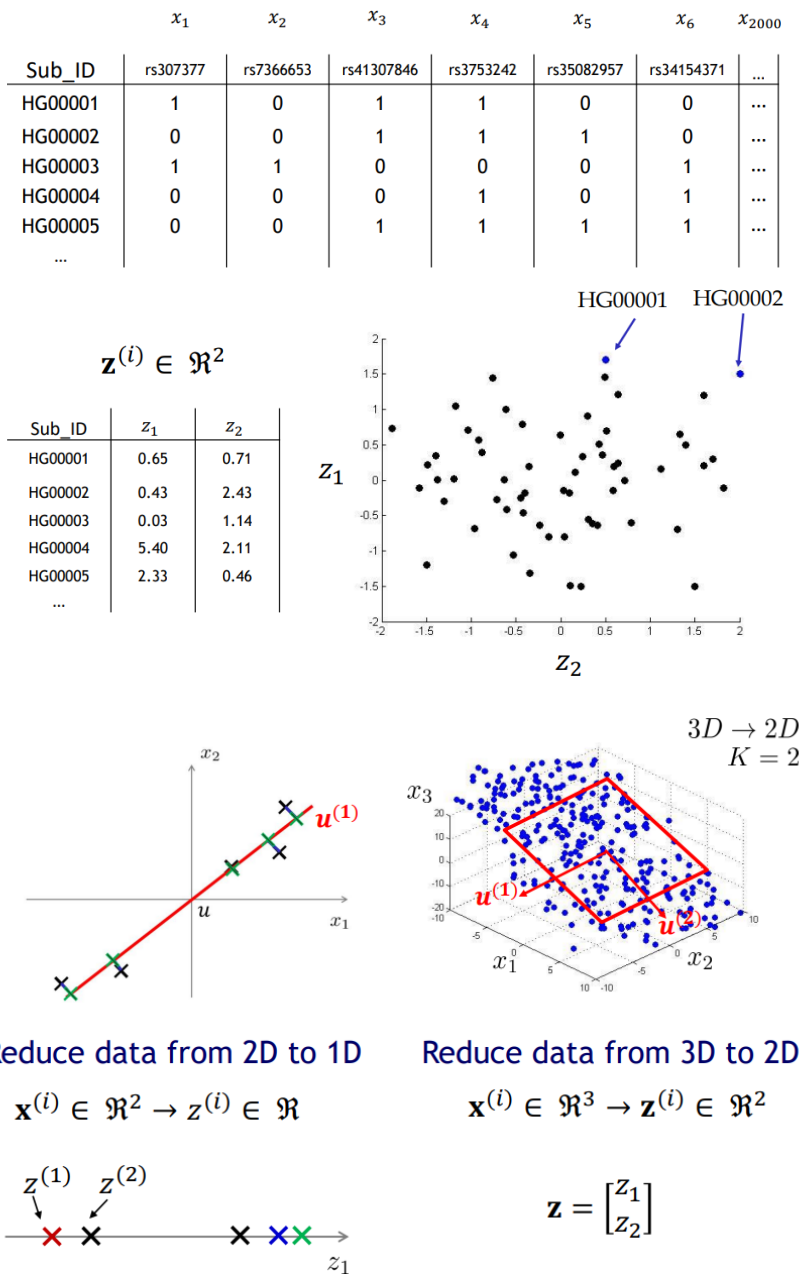
PCA busca minimizar el error de proyección de los datos sobre un subespacio de menor dimensión. Para reducir un espacio de n dimensiones a k dimensiones, PCA determina k vectores ortogonales $\mathbf{u}^{(k)}$ que definen el subespacio óptimo para proyectar los datos:

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

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:

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

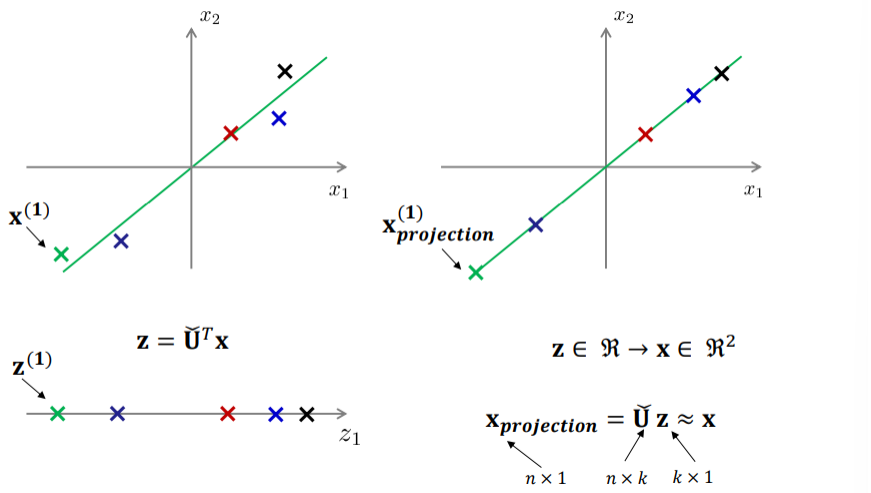
4. **Ordenación y selección de componentes**

   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:

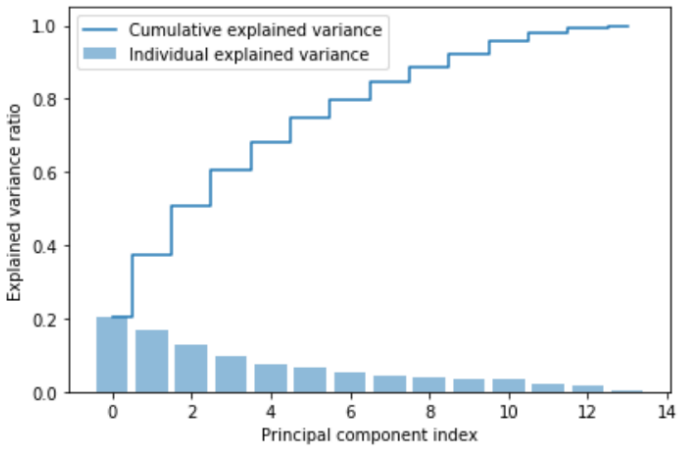
$$i = \frac{\lambda_i}{\sum_{i=1}^{n} \lambda_i}$$

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

}

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

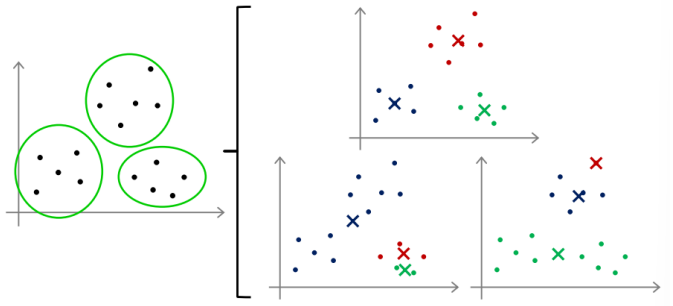
- **Inicialización basada en datos**: Seleccionar K datos aleatorios como centroides iniciales.
- **Repetición del algoritmo**: Ejecutar K-means varias veces y seleccionar la solución con la menor inercia (suma de las distancias al cuadrado entre cada punto y su centroide).

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

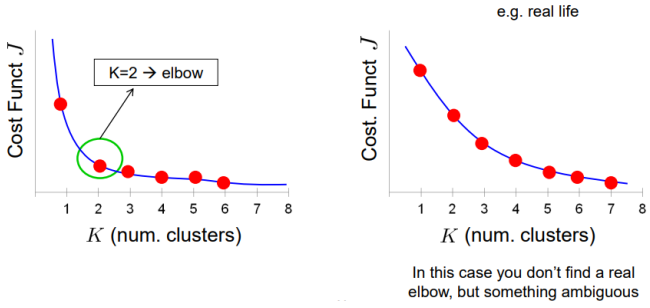
$$\min_{\mu_1,\dots,\mu_K} J(c^{(1)}, \dots, c^{(m)}, \mu_1, \dots, \mu_K)$$

}

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.

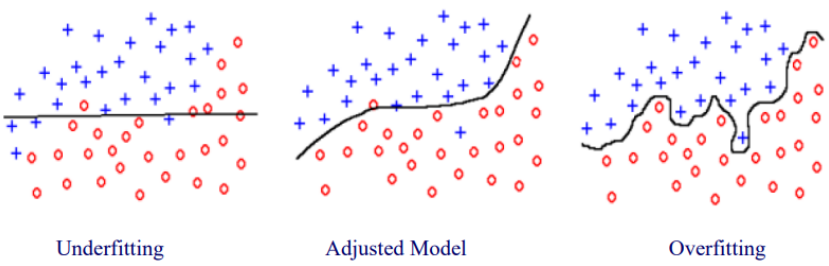
III.1.6.1. Sesgos en el entrenamiento

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.

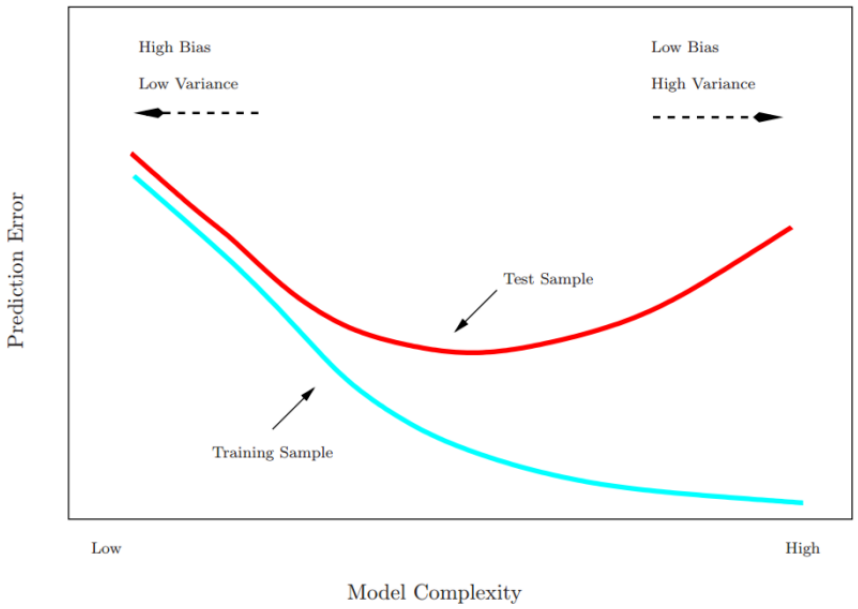


Figura III.1: *En el eje x está la complejidad del modelo, o el orden del polinomio. En el eje y está la predicción del error. Se empieza con un error alto, y se va adaptando, mediante modelos más complejos, para disminuir el error. No obstante, llega un momento en el que, aunque el error de entrenamiento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

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

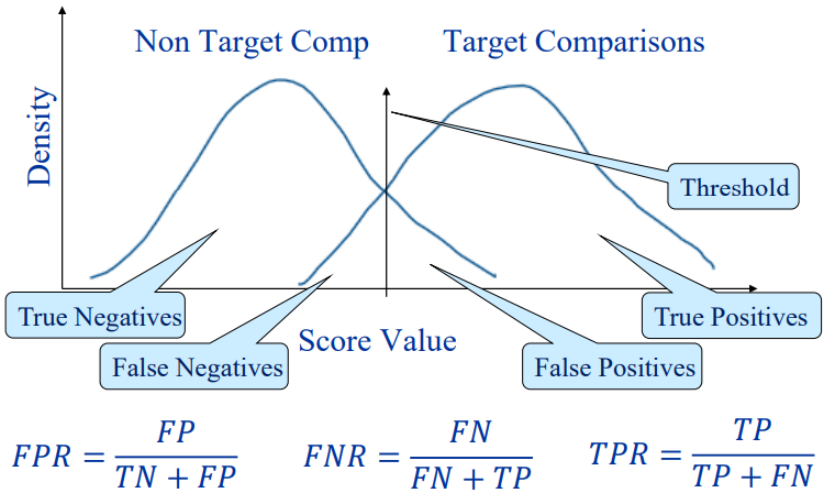
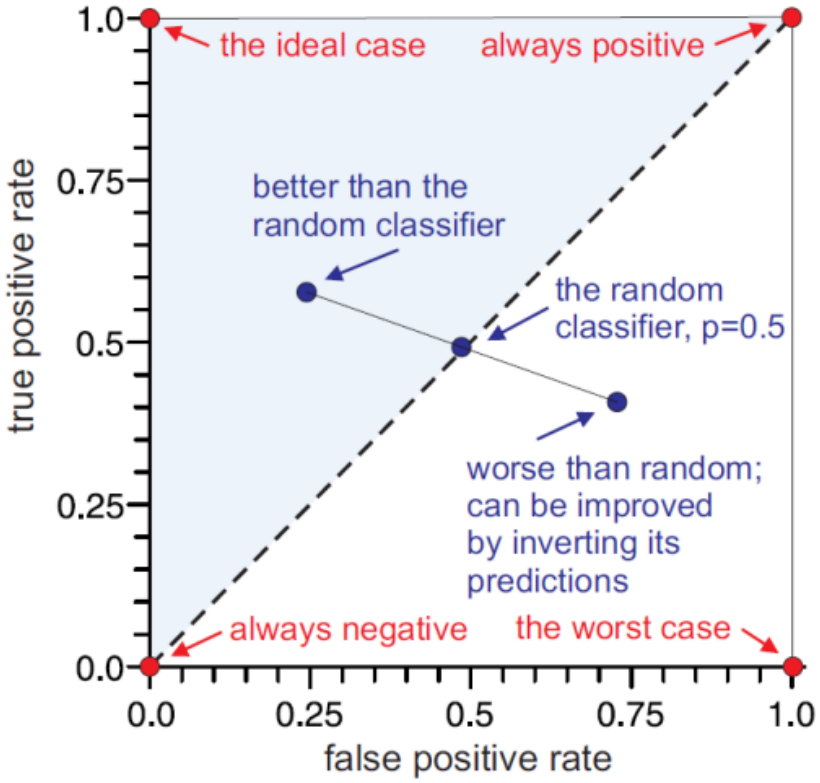


Figura III.2: *En la literatura se habla de Non Target Comparison en lugar de Negativos por la connotación. Normalmente hay un cierto solape entre las distribuciones target y non-target, produciendo así un error. El umbral de decisión se coloca dependiendo del error que se quiera priorizar (o si no se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

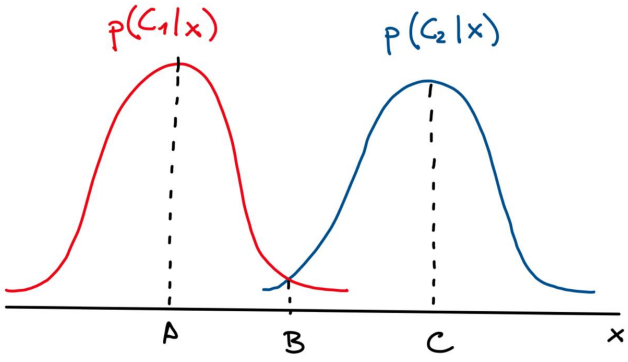
$$\pi(\mathbf{x}) = \begin{cases} \mathcal{C}_1 if p(\mathbf{x}, \mathcal{C}_1) \geq p(\mathbf{x}, \mathcal{C}_2) \\ \mathcal{C}_2 if p(\mathbf{x}, \mathcal{C}_2) \geq p(\mathbf{x}, \mathcal{C}_1) \end{cases}$$

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.

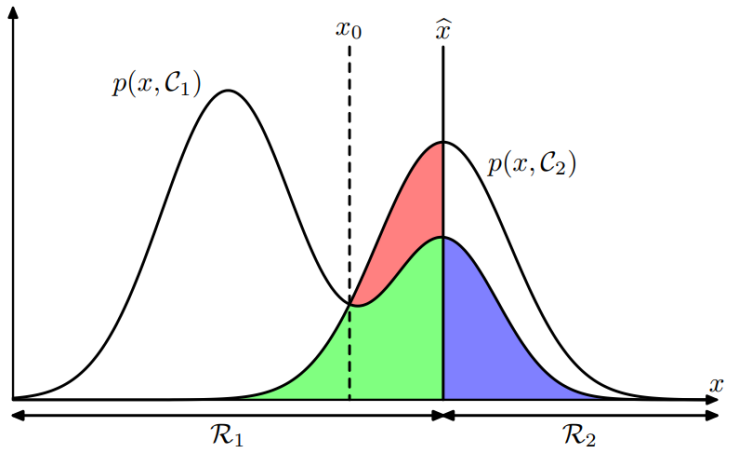


Figura III.3: *Imaginamos que tenemos un problema de clasificación de una dimensión. Tenemos dos clases: la positiva y la negativa. Queremos especificar un clasificador para resolver el problema. Definimos las regiones $\mathcal{R}_1$ y $\mathcal{R}_2$ para las clases $\mathcal{C}_1$ y $\mathcal{C}_2$ 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:

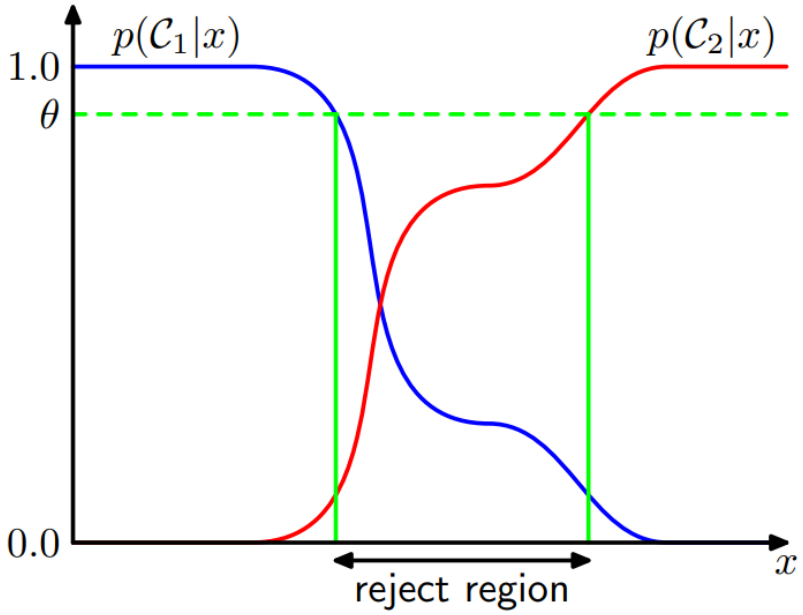


Figura III.4: *La región de rechazo se encuentra alrededor de la frontera de decisión.*

- 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.

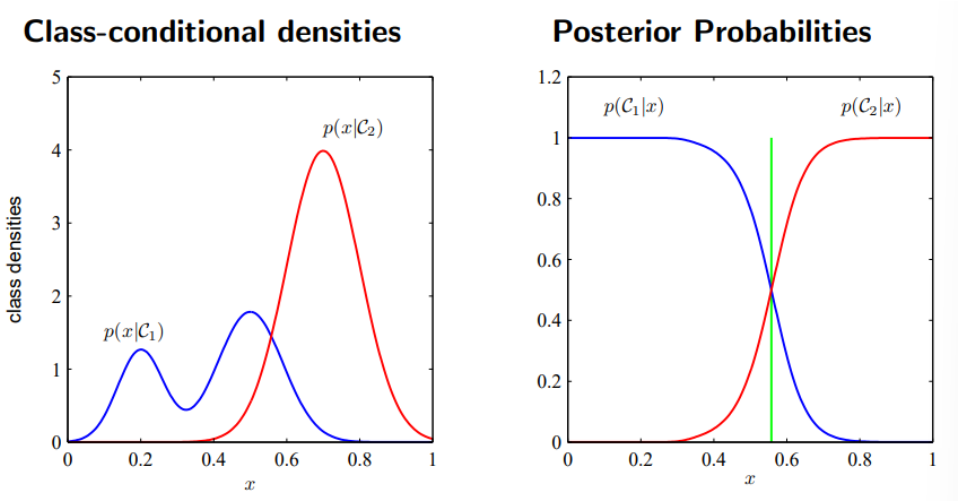


Figura III.5: *Suponemos que tenemos una dimensión con una clase positiva y una clase negativa. La distribución es complicada al tener dos modas. Si esto se utiliza en la regla de Bayes, se pueden calcular las probabilidades posteriores, las cuales tienen menos complejidad que las densidades. Generalmente,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

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

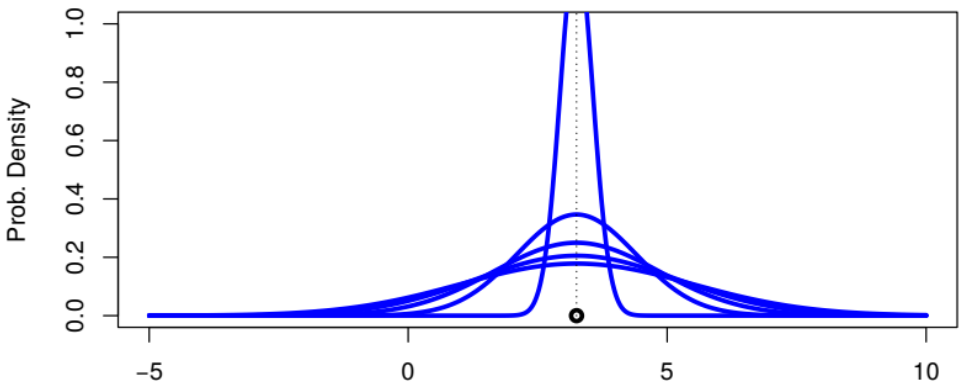
$$\log p(D|\mu) = \sum_{i=1}^{N} -\frac{1}{2\sigma^2}\frac{(x_i - \mu)^2}{\sigma^2} + cte$$

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

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

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.

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

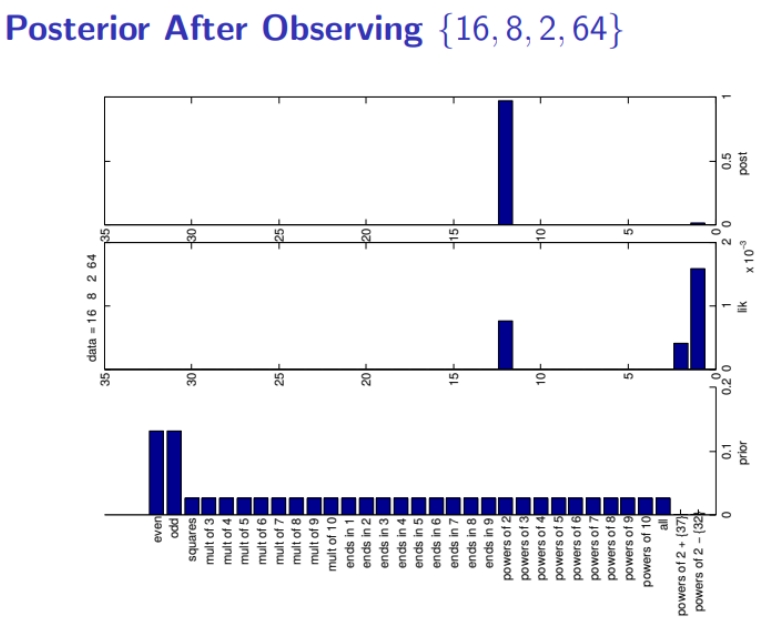


Figura III.6: *Habiendo observado los valores 16, 8, 2 y 64, según la máxima verosimilitud, aceptaríamos la distribución de números de potencias de dos salvo el 32. Como esa distribución es muy rara, utilizamos el conocimiento prior y multiplicamos eso con la máxima verosimilitud, normalizando. De esa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

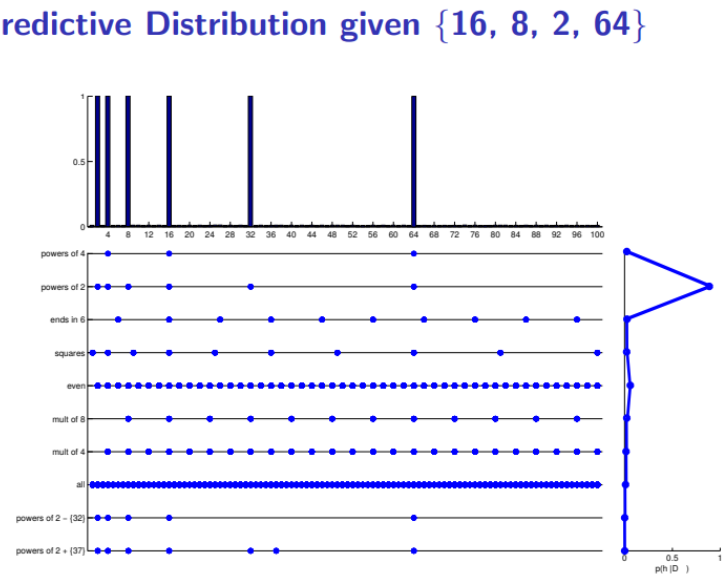
Este resultado muestra cómo la información previa se combina con los datos observados para obtener una mejor estimación del parámetro σ^2.

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.

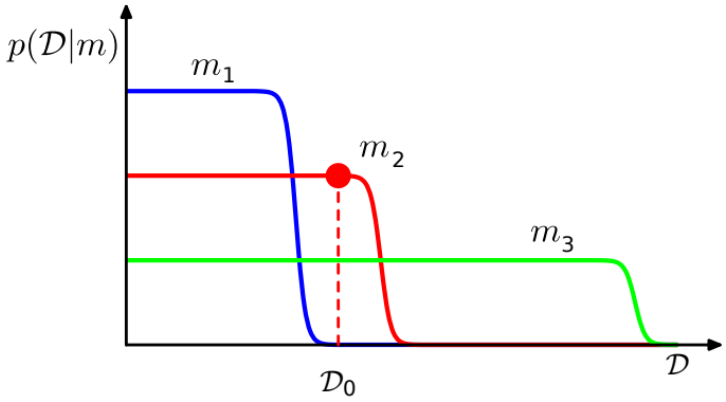


Figura III.7: *m_1 es muy simple para explicar $\mathcal{D}_0$, m_3 es muy complejo, pero m_2 es perfecto.*

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, y es difícil modelar ambos atributos.
- 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:

- **Atributo de valor real:** distribución gaussiana (media y varianza)

- **Atributo binario**: distribución Bernoulli (probabilidad del atributo 1 o 0)
- **Atributo categórico:** distribución Multinoulli

Con K clases y D dimensiones, solo $\mathcal{O}(KD)$ parámetros implican robustez sobre sobreajuste.

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.

III.3.1.3. Breast Cancer Dataset

Este es un problema binario, ya que las muestras pueden ser positivas (tumor maligno) o negativas (tumor benigno). Hay 3 atributos: área, smoothness y simetría. De los tres, simetría es binaria (distribución de Bernoulli), mientras que las otras dos son categóricas.

Recibimos una muestra con los valores large, medium y yes. Para calcular la probabilidad posterior, utilizamos la regla de Bayes para la clase 1:

$$p(C_1 \mid x) = \frac{p(x \mid C_1)p(C_1)}{p(x)} \rightarrow \frac{p(x \mid C_1)p(C_1)}{p(x \mid C_0)p(C_0) + p(x \mid C_1)p(C_1)}$$

Como previamente teníamos 5 casos positivos y 9 negativos, las probabilidades prior son:

$$p(malignant) = 5/14 = 0,36$$
$$p(benign) = 9/14 = 0,64$$

En nuestro caso tenemos tres atributos, por lo que hay que estimar cada uno de los factores. Para el atributo área, tenemos una distribución categórica con 3 categorías, al igual que para smoothness, mientras que para simetría tenemos una distribución de Bernoulli. Se estiman los parámetros para calcular la probabilidad de los datos. Debemos calcular la probabilidad de que área sea large para las clases positivas y negativas. Para ello, calculamos $p(area = large \mid malignant) = 4/5 = 0,8$. De los 5 casos de tumores benignos, 4 de ellos tienen el atributo large en área. Para la otra clase: $p(area = large \mid benign) = 1/9 = 0,1$. Esto también se hace para los atributos smoothness y simetría:

!!! Este ejercicio podría entrar en el examen final: "Utiliza el estimador de Naive Bayes para este ejemplo"

$$p(smoothness = medium \mid malignant) = 1/5 = 0,2$$

$$p(smoothness = medium \mid benign) = 5/9 = 0,6$$
$$p(symmetry = yes \mid malignant) = 1/5 = 0,2$$
$$p(symmetry = yes \mid benign) = 5/9 = 0,6$$

Multiplicando estos valores por el prior, ya tenemos el numerador para Bayes. Para el denominador hay que poner lo mismo sumado a la probabilidad de la clase contraria:

$p(malignant) \cdot p(area = large \mid malignant) \cdot p(smoothness = medium \mid malignant) \cdot (symmetry = ye$

$p(benign) \cdot p(area = large \mid benign) \cdot p(smoothness = medium \mid benign) \cdot (symmetry = yes \mid benign) =$

$$p(malignant \cdot \mathbf{x}_new) = \frac{0,0115}{0,0115 + 0,0213} = 0,35$$

$$p(benign \cdot \mathbf{x}_new) = \frac{0,0213}{0,0115 + 0,0213} = 0,65$$

Con estos datos, predeciríamos la clase **benigno** al tener mayor probabilidad posterior. Esto es equivalente a minimizar el error de predicción, por lo que el paciente no está enfermo.

III.3.1.4. Sobreaprendizaje

La estimación ML de las probabilidades puede sobreajustarse. Si una característica nunca se activa en ambas clases, la estimación de probabilidad correspondiente será cero. En ese caso, dada una nueva instancia con esa característica activada, la probabilidad de tener esa característica activada será cero para cada clase, por lo que el proceso de decisión colapsará (sin importar los otros atributos, el valor final será 0). Posibles soluciones son:

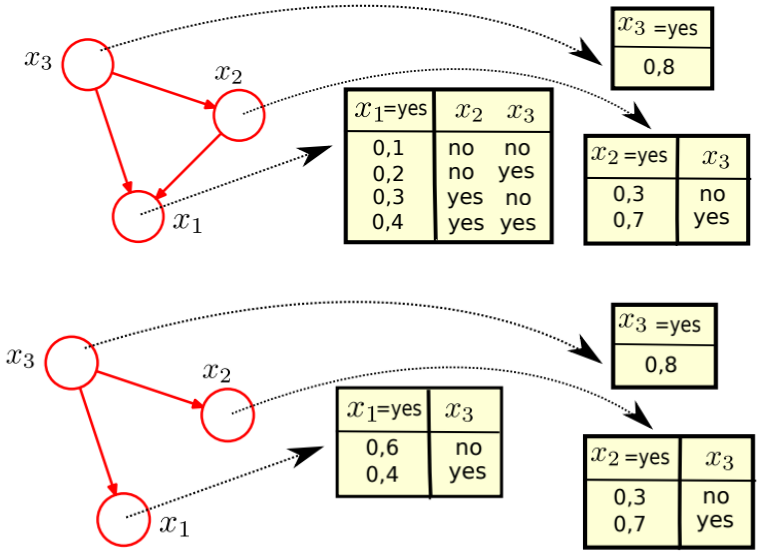
- Utilizar la estimación MAP en lugar de ML e introducir conocimiento previo para imponer estimaciones de probabilidad diferentes de cero o uno.
- Sea totalmente bayesiano y calcule una distribución predictiva para cada x_j marginalizando μ_{jkc} (manejable bajo determinadas premisas).

Aunque es barato de entrenar y robusto frente al sobreajuste, puede ser subóptimo si en la práctica no se cumple el supuesto de independencia. Las superficies de decisión pueden ser muy distintas de las óptimas.

III.3.2. Redes bayesianas

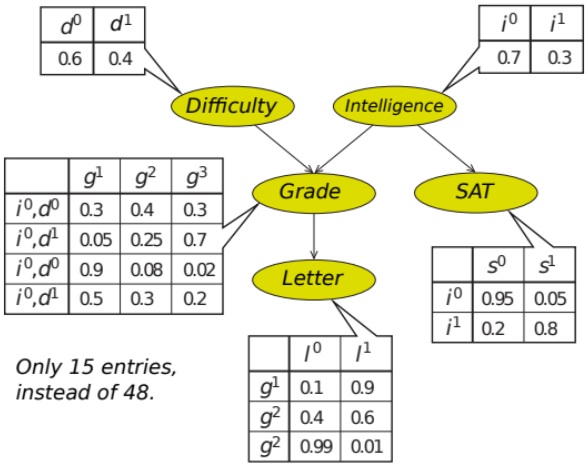
La hipótesis Naive de Bayes tiene varias ventajas: simplifica la estimación de probabilidades cuando x tiene muchas dimensiones, y en caso contrario K valores y D características en x conducen a una tabla de tamaño K^D. En la práctica, es poco probable que se cumpla la hipótesis. Para ello está la red bayesiana, un grafo con un nodo por cada dato. De esta forma, describe gráficamente la distribución de probabilidad conjunta especificando supuestos de independencia condicional. En el grafo, se conectan los nodos con flechas que van de los nodos origen a aquellos que dependen de ellos. Así, tenemos una tabla de probabilidad (tantas tablas como variables hay en la red) con los posibles valores que puede tener.

En este ejemplo, hay que completar 7 valores. Suponiendo que tenemos tres atributos binarios, habría 2^3 combinaciones de atributos, teniendo así 8 parámetros. Como las probabilidades deben sumar 1, solo es necesario calcular 7 (siendo el último 1 - los demás). Si eliminamos las conexiones (aristas) de la red, la distribución se simplifica. Normalmente será una persona experta en los datos la cual decida qué conexiones eliminar sin impactar en los datos. De esta forma, la probabilidad de un parámetro depende de la probabilidad de su parámetro padre.



La red bayesiana de la hipótesis de Naive Bayes asume independencia, por lo que no habría ninguna arista y ningún padre. La idea sería tener algo intermedio: ni todas las aristas ni ninguna. Al quitar todas, se simplifica demasiado.

Un ejemplo para una red bayesiana sería el siguiente:



The corresponding factorization is:

$$p(\text{Diff.}, \text{Int.}, \text{SAT}, \text{Letter}, \text{Grade}) = p(\text{Diff.})p(\text{Int.})p(\text{Grade}|\text{Diff.}, \text{Int.})$$
$$p(\text{Letter}|\text{Grade})p(\text{SAT}|\text{Int.})$$

Con Naive Bayes tendríamos que estimar 6 valores, con la red bayesiana 15, y sin simplificación 48. Se puede utilizar una red bayesiana para obtener algo intermedio entre la hipótesis de independencia total (naive bayes) y la no simplificación.

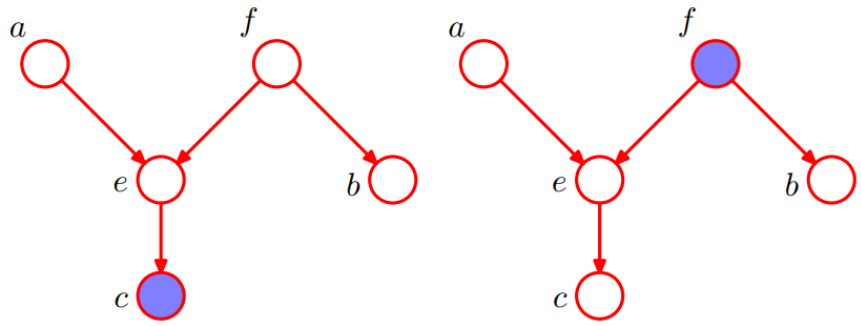
III.3.2.1. Independencia condicional

El proceso de eliminar aristas introduce simplificaciones, las cuales son independencias condicionales. Dos variables son independientes condicionalmente si

su distribución factoriza dada otra variable. x1 es independiente de x2 dado x3, de forma que: $p(x_1, x_2 \mid x_3) = p(x_1 \mid x_3)p(x_2 \mid x_3)$.

Las independencias condicionales de la distribución conjunta pueden leerse a partir del gráfico utilizando la **separación d**: Sean A, B y C conjuntos de nodos no intersecantes, entonces $A \perp B|C$ si todos los caminos desde cualquier nodo de A a cualquier nodo de B están bloqueados. Un camino está bloqueado si contiene un nodo n que o bien:

- las flechas se encuentran cabeza con cola o cola con cola en n y $n \in C$, o
- las flechas se encuentran cabeza con cabeza en n, y ni n, ni ninguno de sus descendientes, está en C.



Note that $a \perp b|c$ does **not** follow from the first network. However, $a \perp b|f$ is implied by the second network.

Figura III.8: *Queremos llegar desde el nodo A al nodo B. Izquierda: se observa C, por lo que el camino no está bloqueado. No se cumple ninguno de los dos criterios anteriores, por lo que A no es independiente de B dado C, son dependientes. Esto se debe a que el nodo E tiene cabeza-con-cabeza, pero su descendiente se ha observado. Derecha: el camino de A a B está bloqueado, por lo que sí se cumple la condición y A es independiente.*

El grafo bayesiano es un filtro en el que se permite el paso de $p(x)$ si y sólo si $p(x)$ satisface la propiedad de factorización dirigida. Podemos utilizar el grafo para filtrar distribuciones en función de si respetan las independencias condicionales implícitas en la separación d. El conjunto de datos final coincide entre ambos (mediante la factorización y mediante la separación).

III.3.2.2. Aprendizaje en redes bayesianas

Aprender del gráfico es muy difícil en la práctica. A menudo lo proporciona un experto con conocimientos previos sobre el proceso de generación de datos.

El aprendizaje de las tablas de probabilidad se basa en observar todas las variables, al igual que en el modelo Naive de Bayes. El aprendizaje se realiza por estimación de máxima verosimilitud o MAP. También se pueden observar solo algunas variables, pero esto requiere el uso práctico del algoritmo expectation-maximization (EM).

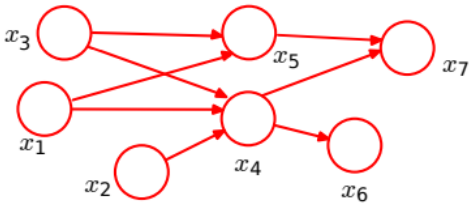
III.3.2.3. Ejercicios

- En un clasificador Naive de Bayes, $D = 5$. Los dos primeros atributos toman valores reales. El tercer atributo toma valores binarios. Los dos últimos atributos toman valores categóricos con 3 y 4 categorías diferentes. Escribe la expresión para $p(x|\mathcal{C}_k)$.

  Los primeros dos factores siguen una distribución gaussiana, mientras que el restante toma una distribución de Bernoulli.

$$p(x \mid \mathcal{C}_k) = \mathcal{N}(x_1 \mid \mu_{1k}, \sigma^2_{1k})\mathcal{N}(x_2 \mid \mu_{2k}, \sigma^2_{2k}) p_{3k}^{x3}(1-p_{3k})^{1-x3} \prod_{i=1}^{3} \mu_{4k}^{I(x4=i)} \prod_{i=1}^{4} \mu_{5k}^{I(x5=i)}$$

- Dada la siguiente red bayesiana, hay que buscar los caminos de x1 a x3 y si están bloqueados por x6. Es decir, hay que ver si se cumple $x_1 \perp x_3 | x_6$.



Is $x_1 \perp x_3 | x_6$ satisfied by this network?

  Al buscar los caminos, no nos importan los sentidos de las flechas. Los caminos son:

  - x1-x5-x3
  - x1-x5-x7-x4-x3
  - x1-x4-x7-x5-x3
  - x1-x4-x3: no aparece x6, pero sí el padre de x6. Este padre tiene cabeza con cabeza, y como x6 está en el condicional, el camino no está bloqueado, y no podemos decir que x1 es independiente de x3 dado x6.

  En el momento en que un camino no esté bloqueado, ya no podemos decir que haya independencia condicional, tienen que estar todos bloqueados.

III.3.3. Resumen

- Los modelos generativos de clasificación estiman $p(x|\mathcal{C}_k)$ y $p(\mathcal{C}_k)$ y utilizan el teorema de Bayes para calcular las probabilidades posteriores de clase.
- Estimar $p(x|\mathcal{C}_k)$ puede ser un problema difícil. El clasificador Naive Bayes lo resuelve suponiendo que $p(x|\mathcal{C}_k) = \prod_{j=1}^{D} p(xj|\mathcal{C}_k)$.
- A menudo, la hipótesis del Naive Bayes no es realista y no se cumple, lo que da lugar a decisiones y resultados de clasificación subóptimos.
- Una red bayesiana ofrece una solución intermedia (simplicidad y precisión) cuando se modela $p(x|\mathcal{C}_k)$, o cualquier otra distribución.
- Las simplificaciones se obtienen introduciendo independencias condicionales en $p(x)$ mediante la eliminación de aristas en la red.

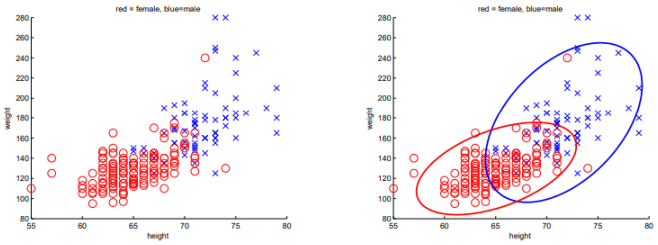
III.4. Análisis discriminante

Como hemos dicho antes, es poco probable que en la práctica se cumpla la hipótesis Naive de Bayes .Una red bayesiana permite simplificar este supuesto y conduce a un modelo de probabilidad más flexible cuando se trabaja con atributos categóricos. En el caso de los atributos continuos, una distribución gaussiana multivariante permite tener en cuenta las dependencias entre los atributos.

III.4.1. Distribución gaussiana multivariante

Esto permite introducir algo más flexible que la hipótesis Naive de Bayes. Las entradas de la diagonal representan la varianza de cada componente. Las entradas fuera de la diagonal son las dependencias entre variables. Si son negativas, es que existe correlación negativa entre variables. Cuando recibe el valor 0, es que no hay dependencias.

Esto permite describir una serie de métodos para resolver problemas de clasificación: análisis discriminante. Utiliza una gaussiana multivariante para describir las condiciones multivariante. Aunque se llame análisis discriminante, se trata de un modelo generativo. Se utiliza una gaussiana para estimar las probabilidades a priori de cada clase y mediante el teorema de Bayes se puede calcular la probabilidad posterior para predecir. Si la matriz es cuadrada y la diagonal es 0, se trata de Naive Bayes. Para el prior se utilizan los estimadores de máxima verosimilitud (media empírica y promediar el vector con la media y él mismo transpuesto).



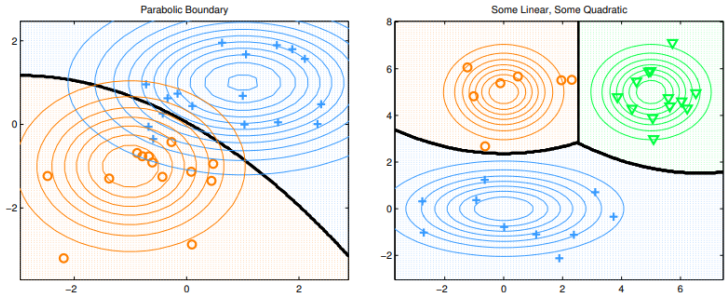
Height/weight data. Visualization of 2d Gaussians fit to each class. 95% of the probability mass is inside the ellipse.

III.4.2. Análisis discriminante cuadrático

Se utilizan matrices de covarianza distintas para cada clase. Al enchufar las distribuciones en la regla de Bayes, en el numerador queda el prior de la clase y la probabilidad de x dada la etiqueta de clase (densidad gaussiana multivariante). En el denominador se pone la suma del numerador para cada etiqueta de clase.

$$p(y_i = \mathcal{C}_k \mid \mathbf{x}) = \frac{\pi_k \det(2\pi\sigma_k)^{-1/2} \exp\left\{-\frac{1}{2}(\mathbf{x}-\mu_k)^T\sigma_k^{-1}(\mathbf{x}-\mu_k)\right\}}{\sum_{k'} \pi_{k'} \det(2\pi\sigma_{k'})^{-1/2} \exp\left\{-\frac{1}{2}(\mathbf{x}-\mu_{k'})^T\sigma_{k'}^{-1}(\mathbf{x}-\mu_{k'})\right\}}$$

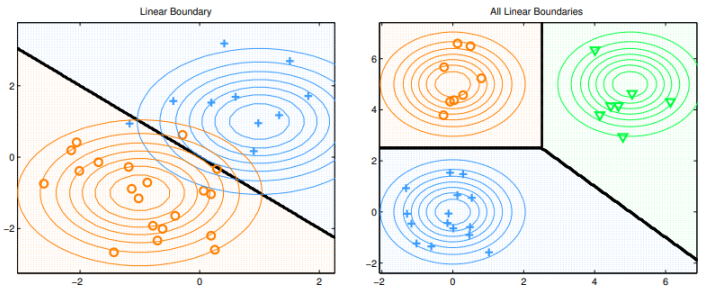
En un problema de clasificación, la frontera está donde la probabilidad posterior para ambas clases es igual (0,5). El resultado es el análisis discriminante cuadrático, ya que la superficie de decisión es una función cuadrática (parábola).



Quadratic decision boundaries of QDA in 2D for the 2 and 3 class case.

III.4.3. Análisis discriminante lineal

Cuando las matrices de covarianza para las clases se comparte (pero el vector de medias no), podemos simplificar. Así, el término cuadrático que depende de x por la matriz de covarianza invertida por x, se puede meter en un término constante. La frontera de clasificación va a ser lineal.



Linear decision boundaries of LDA in 2D for the 2 and 3 class case. Note that all densities have equal covariance matrices.

Lo difícil de este método es estimar las matrices de covarianza. Para ello, se maximiza la probabilidad de los datos observados, donde la media depende de la etiqueta de clase de donde venga. Se puede utilizar el estimador de máxima verosimilitud, quedando una mezcla de la máxima verosimilitud para las dos gaussianas.

Entre estos dos análisis (el cuadrático y el lineal), es el cuadrático el que tiene más términos y también el más flexible. Además, el cuadrático es el que tiene mayor riesgo de sobreajuste. El lineal es más robusto: tiene menos parámetros, es menos flexible, pero tiene menos riesgo de sobreajustar a los datos de entrenamiento.

La única diferencia entre los dos análisis es la estimación de la matriz de covarianza (para el cuadrático hay dos matrices, para el lineal es compartida entre las clases). Hay que maximizar la matriz de covarianza observada con respecto a la matriz sigma. Se

obtiene el estimador de máxima verosmilitud para la matriz de covarianza, siendo el promedio de las matrices de covarianzas de las distintas etiquetas de clase.

III.4.3.1. Sobreaprendizaje

El análisis discriminante cuadrático es más sensible al sobreaprendizaje, pudiendo haber problemas cuando hay pocos datos. A la hora de calcular las probabilidades posteriores de clase, hay que calcular las inversas de las matrices de covarianza. Si el número de datos es menor que el de las observaciones, no va a ser posible la inversión, siendo un problema para estos clasificadores. Si ocurre eso, la matriz de covarianza no se puede invertir, por lo que se hace diagonal. En ese caso, iríamos al clasificador de Naive Bayes. Otra técnica podría ser igualar todas las matrices de covarianza, juntando los datos de todas las clases, aplicando el análisis discriminante lineal. Alternativamente, se puede forzar que las matrices de covarianza sean iguales y forzar que sean diagonales, utilizando el clasificador del análisis discriminante lineal diagonal. Por último, se puede utilizar la estimación máxima a posteriori para ajustar la matriz de covarianza.

III.4.4. Análisis discriminante regularizado (RDA)

Se puede introducir como prior para la matriz de covarianzas la distribución de probabilidad de Wishart. Tiene dos parámetros: la media de la distribución (una matriz de covarianzas alrededor de la que estarían nuestras matrices) y un escalar que define la dispersión alrededor de la media. Con esta distribución definimos que esperamos que la matriz de covarianzas esté cerca de la matriz de identidad. El parámetro de dispersión es equivalente a λ, que afectaría al parámetro máximo a posteriori. λ tomaría valores entre 0 y 1, mientras que MAP sería una combinación entre la matriz prior y máxima verosimilitud. Lo ideal sería poner algo intermedio entre los dos valores. Para algunos valores de λ podemos obtener algunas matrices invertibles, mejores que las de MAP. Esto introduce un hiperparámetro al modelo que se debe ajustar. Para ello, se utilizan técnicas de comparación de modelos mediante la validación cruzada.

Hay que tener cuidado a la hora de elegir hiperparámetros. Cuando se ajusta, no se pueden utilizar datos de tests, ya que se utilizarían para construir el clasificador y ya no sería independiente a ellos. La estimación sería sesgada. Para ajustar los hiperparámetros, se haría validación cruzada sobre los datos de entrenamiento. Se obtendría el valor de λ con el que funcione mejor el modelo y se evalúa con los datos de test.

III.4.5. Análisis discriminante lineal diagonal

En el caso de que la matriz de covarianza sea diagonal y compartida entre todas las clases, se conoce el modelo como análisis discriminante lineal diagonal. Para cuando hay muchas dimensiones, esto funciona mejor que otros análisis. Esto es menos flexible al tener menos parámetros, por lo que es bueno con pocos datos de entrenamiento y muchas dimensiones, ya que se limita el riesgo de sobreaprendizaje.

III.4.6. Clasificador Nearest Shrunken Centroids

Este clasificador es una variante del anterior en el que solo algunos atributos son relevantes. En lugar de hacer estimación máxima a posteriori para la matriz, se utiliza MAP para las medias. La media de cada clase es igual a una media común igual para todas las clases y un término diferenciador que va a ser el que diferencie las etiquetas de clase. A eso se le conoce como centroide. Se fuerza que ese término se aproxime a 0. Cuando el término se hace exactamente a 0, la media para todas las clases es la misma, siendo la media para ese atributo. Esto se traduce en que el atributo es irrelevante para calcular la probabilidad a posteriori, al contribuir de la misma manera para cada una de las distintas etiquetas de clase, sin introducir ningún cambio en el cálculo de las probabilidades posteriores. Así, se hace una selección de atributos.

Para que los centroides se aproximen a 0, hay un hiperparámetro entre 0 y el máximo valor de los centroides, siendo esa la cantidad que se sustrae a los centroides. Si se sustituye para que los centroides sean 0, lo que queda es el prior en la regla de Bayes.

Este clasificador tiene un hiperparámetro que indica lo que se encogen los centroides. Si no se encogen, tenemos un análisis discriminante lineal, y si se encogen mucho, el prior de las clases. Por ello, se debe estimar el hiperparámetro con validación cruzada.

III.4.7. Ejercicios

1. ¿Qué ventajas y desventajas tienen QDA y LDA?

   QDA tiene fronteras más flexibles al ser parábolas en comparación con las fronteras lineales de LDA. No obstante, es más propenso al sobreaprendizaje con un peor error de generalización.

2. Imagina que aplicas QDA a un problema de clasificación con más atributos que datos. La matriz es singular y no inverda. ¿Qué técnicas se pueden utilizar para resolver este problema?

   Se puede utilizar la estimación máxima a posteriori y ajustar λ que interpola entre una matriz igual a la de identidad y una matriz diagonal. Esta es una de las soluciones. Otra solución sería coger la diagonal, quedando así Naive Bayes. También se podrían forzar las matrices de covarianza para que fueran iguales (y por tanto, compartidas).

III.4.8. Resumen

- Es poco probable que se cumpla la hipótesis Naive Bayes. Una distribución gaussiana multivariante puede tener en cuenta las dependencias.
- El análisis discriminante surge utilizando el teorema de Bayes y una gaussiana multivariante para estimar las densidades condicionales de clase.

- QDA considera diferentes matrices de covarianza para cada densidad condicional de clase, que pueden estimarse utilizando ML. Los límites de decisión de QDA son funciones cuadráticas.
- LDA considera matrices de covarianza compartidas para cada densidad condicional de clase. Los límites de decisión de LDA son funciones lineales. Es menos flexible pero más robusto al sobreajuste.
- DLDA es una variante de LDA que asume una matriz de covarianza diagonal.
- Para mejorar la robustez de los métodos discriminantes se puede utilizar la estimación MAP. En algunos casos, esto conduce a la selección de atributos.

III.5. Vecinos próximos y árboles de clasificación

III.5.1. Vecinos más próximos y estimación de la densidad

Hasta ahora, hemos creado modelos generativos con probabilidades condicionales calculadas con el teorema de Bayes para hacer predicciones. Calcular las estimaciones condicionales es complicado de hacer en la práctica.

La hipótesis de gaussianidad es restrictiva al ser unimodal, lo cual no siempre coincide. Los vecinos próximos es más flexible que la distribución gaussiana. Se estiman densidades de distribuciones (cada etiqueta de clase). La probabilidad de que los datos caigan en una región en el espacio depende del área bajo la curva de la densidad.

Imaginemos que tenemos datos observados, teniendo que estimar la densidad de probabilidad en dos regiones. La probabilidad de que los datos caigan en la región 1 es el área debajo de la curva de la densidad de la región 1.

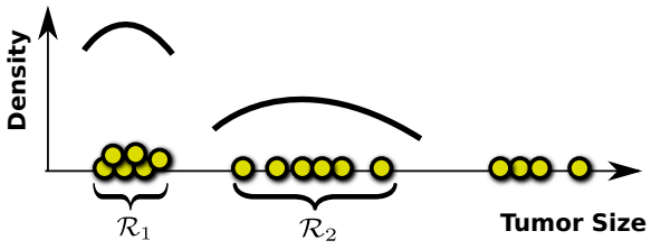
We want to **estimate** $f(x)$ so that $\Pr(X \in \mathcal{R}) = \int_{\mathcal{R}} f(x)dx$.

How can we approximate $\Pr(x \in \mathcal{R}_1) \approx 6/16$ and $\Pr(x \in \mathcal{R}_2) \approx 6/16$?

Should the density curves above $\mathcal{R}_1$ and $\mathcal{R}_2$ be equally high?

$$\Pr(x \in \mathcal{R}_1) = \int_{\mathcal{R}_1} f(x)dx\,, \qquad \Pr(x \in \mathcal{R}_2) = \int_{\mathcal{R}_2} f(x)dx$$

No, since $\mathcal{R}_1$ is smaller than $\mathcal{R}_2$.



La estimación de densidad se debe normalizar en cuanto al tamaño de la región. Suponemos que la densidades son constantes (que la curva es plana), el área bajo la curva es un rectángulo, altura por longitud, es decir, densidad de probabilidad por anchura de la región.

De esta forma, la densidad se puede aproximar como:

$$f(x) \approx \frac{\text{number of samples in region}}{\text{total number of samples}} \cdot \frac{1}{\text{volume of the region}} = \frac{\kappa}{N}\frac{1}{V}$$

Esto se debe estimar para cada etiqueta de clase. Si hacemos esto, para la estimación de la distribución condicional queda:

$$p(\mathbf{x} \mid \mathcal{C}_k) \approx \frac{N_k(\mathbf{x}}{N_k V(\mathbf{x})}$$

Este estimador es para la distribución condicional, y se utiliza en la regla de Bayes para las probabilidades posteriores:

$$p(y_i = \mathcal{C}_k \mid \mathbf{x}) = \frac{N_k(\mathbf{x})}{\sum_{k'} N_{k'}(\mathbf{x})} = \frac{N_k(\mathbf{x})}{\kappa}$$

Así, en la clasificación de vecinos κ-próximos, simplemente hay que predecir la clase más común entre los κ vecinos de x. Una κ pequeña produce muchas regiones pequeñas de cada clase, mientras que una κ grande da lugar a menos regiones grandes.

Las ventajas de este clasificador son:

- Puede aplicarse a los datos de cualquier distribución.
- Muy sencillo e intuitivo.
- No paramétrico: más expresivo a medida que tenemos más muestras.
- Buen rendimiento si el número de muestras es grande.

No obstante, también tiene algunas desventajas:

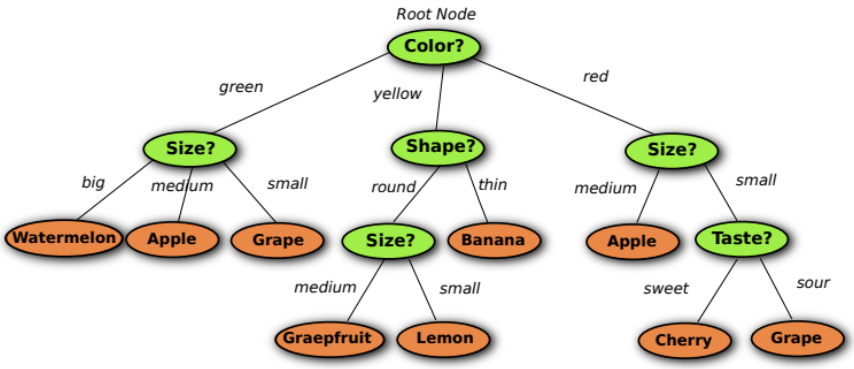
- Elegir el mejor κ puede ser difícil, realizándose mediante búsqueda en validación cruzada. Una κ pequeña provoca un sobreajuste, y una κ pequeña un infraajuste.
- Es computacionalmente pesado al necesitar almacenar todos los datos de entrenamiento.
- Se necesita un gran número de muestras para obtener una buena precisión.

III.5.2. Árboles de clasificación

Es natural clasificar un patrón mediante una secuencia de preguntas, en la que la siguiente pregunta depende de la respuesta a la pregunta actual. La principal ventaja es la interpretabilidad. Es sencillo transformar el árbol en un conjunto de reglas de clasificación.

El aprendizaje de árboles de decisión suele ser más adecuado si

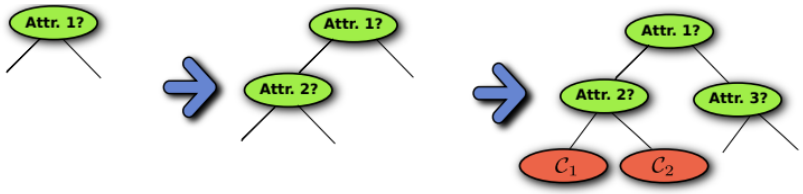
- Varios atributos categóricos, por ejemplo, el tamaño puede ser pequeño, mediano o grande.



- El problema de clasificación tiene varias etiquetas, es decir, es multiclase.
- Los datos de entrenamiento pueden contener errores (atributos o etiquetas).
- Los datos de entrenamiento pueden contener valores perdidos (missing values).
- Nos interesa la interpretabilidad del clasificador.

La mayoría de los algoritmos emplean una búsqueda codiciosa descendente:

1. Se determina qué atributo debe comprobarse en la raíz.
2. A continuación, se crean descendientes para cada valor posible.
3. Los datos de entrenamiento se clasifican en el descendiente apropiado.
4. El proceso se repite con los datos asociados a cada descendiente.
5. El proceso se detiene, por ejemplo, cuando todos los casos pertenecen a la misma clase.



Example Algorithms: ID3, C4.5, CART

III.5.2.1. Selección de test en cada nodo

¿Qué atributo debemos seleccionar en cada nodo? La idea clave es seleccionar el atributo que sea más útil para clasificar. Esto dará lugar a árboles más pequeños y compactos, y requiere una medida de impureza de los datos. Esta impureza se mide con la entropía, el índice de Gini o la tasa de error. Todas las medidas son altas en el 50 % de cada clase.

Dada una determinada medida de impureza, ¿qué atributo debemos seleccionar para dividir en el nodo del árbol parcial? Una heurística elige el atributo que disminuye más la impureza. Se trata de una regla codiciosa que intenta generar árboles pequeños.

Como ejemplo, retomamos el dataset de la clasificación tumoral (con size, smoothness y symmetry). Utilizando la entropía, hay 5 ejemplos de la clase positiva y 9 de la negativa. La impureza de los datos es la entropía de un conjunto de datos con 9 ejemplos de la clase negativa y 5 de la positiva. Los cálculos serían

$$5/9 \cdot log_2(5/14) - 9/14 \cdot log_2(9/14) = 0.940286$$

Ahora queremos ver la reducción esperada de la impureza de los datos al utilizar el atributo symmetry para separar los datos. Es un atributo binario con 6 datos yes y 8 no. De los 6, hay 5 ejemplos de la clase negativa y 1 de la positiva, mientras que de los 8 son 4 de la clase positiva y 4 de la negativa. Así:

$$0.94 - 6/14 log_2(6/14) - 8/14 log_2(8/14) = 0.09$$

Ahora consideramos el tamaño:

$$0.94 - 5/14 log_2(5/14) - 5/14 log_2(5/14) - 4/14 \cdot 0 = 0.424$$

Comparando los dos resultados, la entropía de tamaño es mayor, por lo que es un mejor atributo para la raíz. También habría que tener en cuenta smoothness para determinar la raíz. Los resultados son:

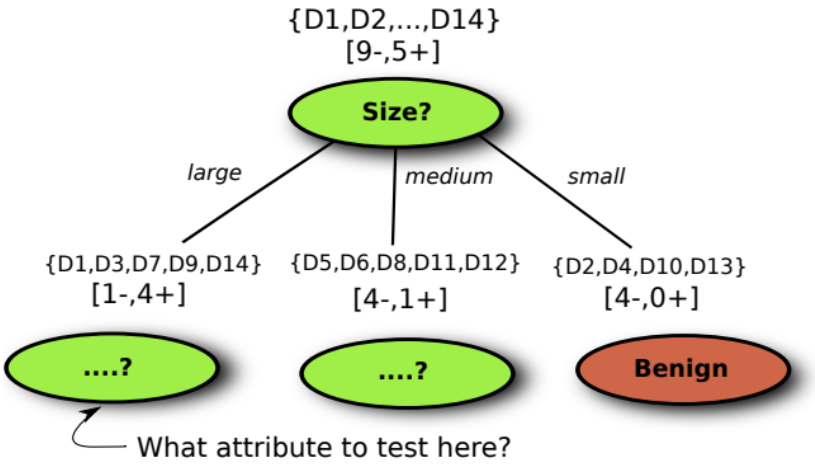
- Smoothness = 0.119
- Symmetry = 0.09
- Size = 0.424

La mayor disminución de la impureza de los datos corresponde a Tamaño. El nodo raíz comprueba el atributo Tamaño y el proceso se repite para cada hijo hasta que se cumple un criterio de parada.

III.5.2.2. Tamaño del árbol

Si el tamaño del árbol es muy grande, puede ajustarse en exceso a los datos. Por el contrario, si el tamaño es demasiado pequeño, puede no ajustarse lo suficiente a los datos. Existen diferentes métodos para evitar que crezcan árboles demasiado grandes:

- Mantener un conjunto de validación (10 % de los datos). Continúa dividiendo nodos hasta que se minimice el error en los datos de validación.
- La división se detiene si la mejor división reduce la impureza en menos de una cantidad preestablecida β, es decir, si $max_a Decrease(\mathcal{D}, a) \leq \beta$.
- Se detiene cuando un nodo tiene menos de algún número umbral de puntosN_{min} o algún porcentaje del conjunto total de entrenamiento, por ejemplo, 5 %.
- Intercambia complejidad por precisión de la prueba



$$\mathcal{D}_{\text{large}} = \{D1, D3, D7, D9, D14\}$$

$$\text{Decrease}(\mathcal{D}_{\text{large}}, \text{Symmetry}) = 0.722 - \frac{1}{5}0.0 - \frac{4}{5}0.0 = 0.722$$

$$\text{Decrease}(\mathcal{D}_{\text{large}}, \text{Smoothness}) = 0.722 - \frac{3}{5}0.92 - \frac{1}{5}0.0 - \frac{1}{5}0.0 = 0.17$$

III.5.2.3. Estrategias post-pruning

La división detenida adolece de falta de suficiente visión de futuro, por lo que es preferible la realizar post-pruning.

Hay que hacer crecer un árbol hasta que los nodos de las hojas tengan una impureza mínima. A continuación, se considera cada nodo del árbol como candidato a la poda:

- Eliminar el subárbol completo enraizado en ese nodo.
- Convertir el nodo eliminado en una hoja.
- Asignar la clase más común dentro de los datos que llegan a ese nodo.

Los nodos se podan de forma iterativa, eligiendo el nodo cuya eliminación más aumenta la precisión en el conjunto de validación. La poda continúa hasta que una nueva poda resulte perjudicial.

III.5.2.4. Incorporación de atributos continuos

Se ordenan los atributos continuos de menor a mayor y se observa en qué rango hay un cambio de la etiqueta.

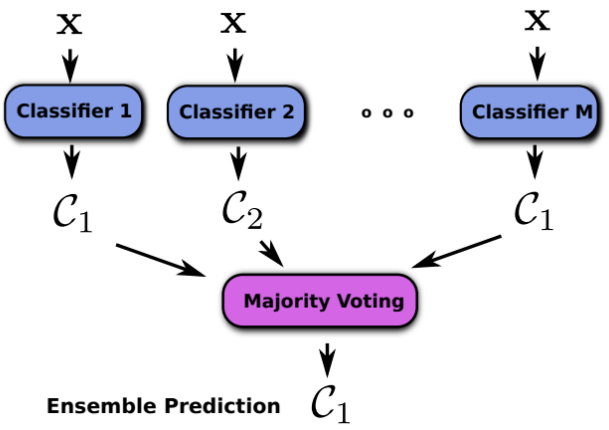
III.5.3. Resumen

- Los vecinos más próximos son un método de clasificación no paramétrico muy sencillo cuya capacidad expresiva crece con el tamaño del conjunto de datos.
- El número de vecinos κ especifica la complejidad del modelo. Un valor alto de κ conduce a un ajuste insuficiente y κ pequeño conduce a un ajuste excesivo.

- Los árboles son modelos interpretables que dividen los datos en cada nodo. Tratan atributos continuos, categóricos y ausentes.
- Los árboles se generan mediante un algoritmo codicioso que trata de encontrar la mejor división en cada nodo para reducir al máximo la impureza de los datos.
- Los árboles completamente desarrollados suelen ajustarse en exceso a los datos de entrenamiento, por lo que las estrategias de poda a posteriori, en las que se reduce el tamaño del árbol, son necesarias.
- Los árboles de clasificación tienden a funcionar un poco peor que otros modelos.

III.6. Combinación de clasificadores: métodos ensemble

La predicción del conjunto de los clasificadores individuales será la clase mayoritaria. De esta forma, se puede probar si los errores de los clasificadores son independientes, teniendo así una clasificación mejorada.



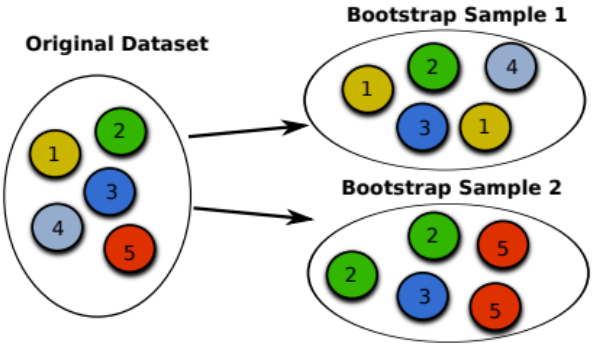
El ensemble puede reducir significativamente el error de un único clasificador simplemente considerando varias versiones del mismo. Por desgracia, los errores cometidos por los clasificadores no son independientes en la práctica. Sólo existe un único conjunto de datos de entrenamiento D, y hay que introducir variabilidad entre los clasificadores utilizando diferentes técnicas:

- Entrenamiento de cada clasificador en una muestra bootstrap.
- Inyección de ruido en el conjunto de entrenamiento (cambiar algunas etiquetas).
- Introduciendo cierta aleatoriedad en el algoritmo de entrenamiento.

Los errores suelen estar sólo ligeramente correlacionados, y la reducción del error global suele ser significativa.

III.6.1. Bagging, agregación de bootstrap

Cada clasificador base se entrena en una muestra bootstrap de los datos, y se utiliza la votación por mayoría para combinar los resultados de los clasificadores. Al construir una muestra bootstrap, se pueden repetir algunos datos en una misma muestra, y algunos datos pueden faltar. En torno a un tercio de los datos de una muestra de bootstrap van a estar repetidos, y por tanto en torno a un tercio de los datos totales no van a estar presentes.



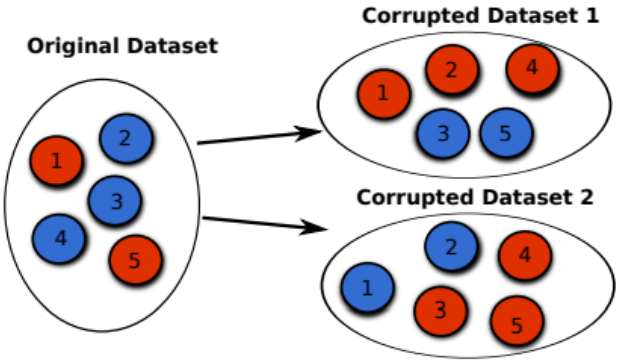
Las características del Bagging son:

- Puede aplicarse a cualquier algoritmo de clasificación (normalmente árboles).
- El error de un clasificador suele ser mayor (debido al muestreo bootstrap).
- El error del conjunto disminuye con M y suele ser mucho mejor.
- Funciona mejor con clasificadores inestables (reduce la varianza). Estos clasificadores tienen grandes cambios en los límites de decisión y la predicción con pequeños cambios en el conjunto de entrenamiento. Por ejemplo, los árboles de clasificación no podados serían un clasificador inestable.
- Puede degradar el rendimiento de los clasificadores estables, es decir, clasificadores en los cuales grandes cambios en el conjunto de entrenamiento producen pequeños en los límites de decisión (predicciones), como pueden ser los K-vecinos más próximos.

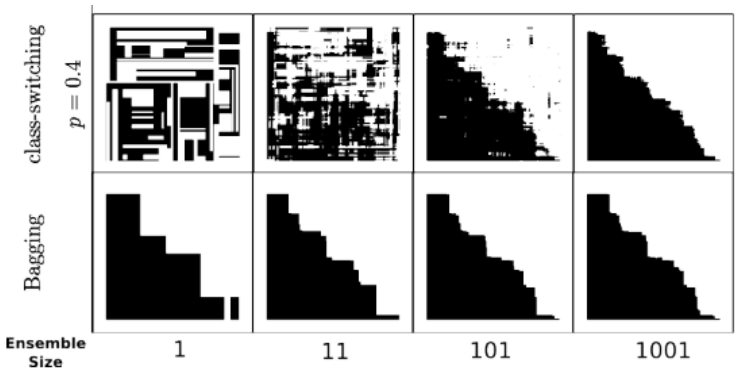
El bagging es peor al principio que utilizar un solo clasificador debido al bootstrap (se utilizarían menos datos de entrenamiento), pero al final es mejor que un solo árbol. No obstante, es más costoso computacionalmente, ya que hay que generar varios clasificadores. Otra desventaja es que la interpretabilidad se pierde, pero al menos se mejora el error de clasificación.

III.6.2. Cambio de clases

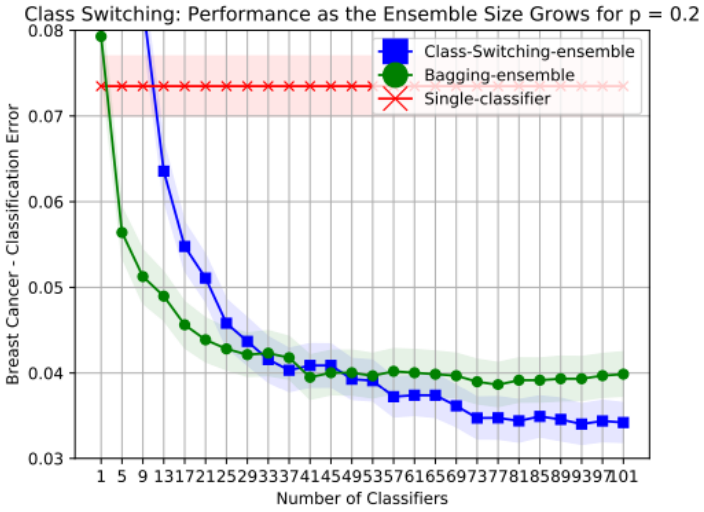
Cada clasificador base se entrena en una versión corrupta de los datos. Cada vez, se cambia aleatoriamente la etiqueta de una fracción p de los datos. Se utiliza la votación por mayoría para combinar los resultados de los clasificadores.



Este cambio de clase obliga a cometer errores independientes de forma más agresiva que el bagging. Requiere ajustar un (hiper)parámetro adicional p que depende del problema. A menudo se requieren conjuntos de mayor tamaño que en el bagging. Un p grande tiende a funcionar mejor, pero también requiere valores más grandes de M.



Comparando el cambio de clases con el uso de un solo clasificador y el bagging, al principio es el peor método, pero al final es el mejor.



Cuando p aumenta, el error del conjunto tarda más en converger, por lo que pueden ser necesarios conjuntos más grandes.

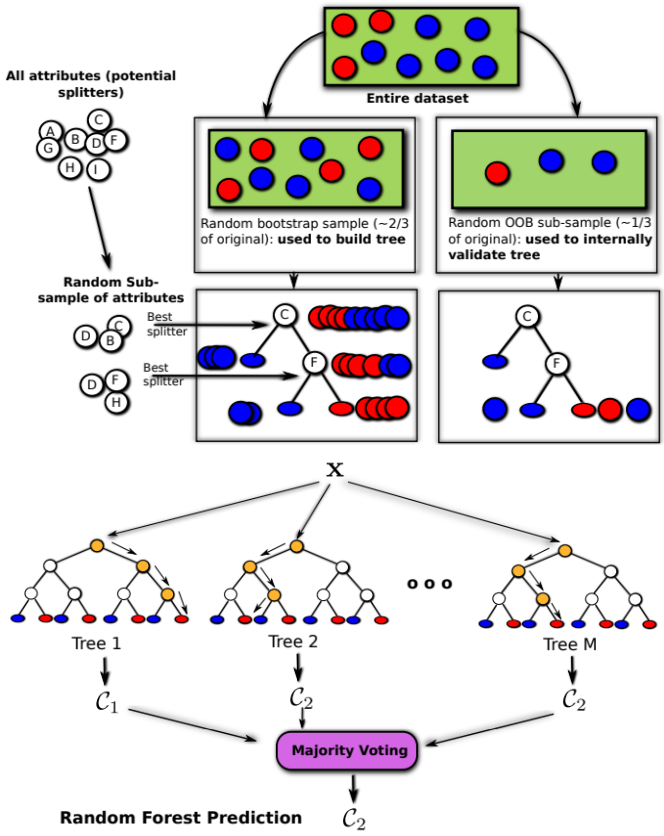
III.6.3. Random forest

Cada clasificador de árbol se entrena en una muestra bootstrap de los datos. Se introduce una cierta aleatoriedad en el algoritmo de crecimiento de los árboles, y se permite que éstos crezcan al completo, sin poda. Se utiliza la votación por mayoría para combinar los resultados de los clasificadores.

La aleatoriedad se introduce en el proceso de crecimiento del árbol:

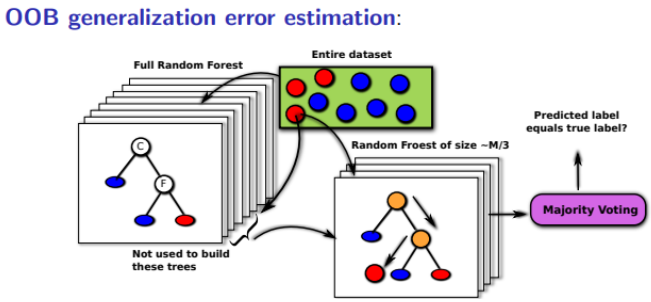
- En cada nodo, se elige un subconjunto aleatorio de m $<$ D atributos.
- Se calcula la disminución de impurezas para cada uno de los m atributos.
- Se dividen los datos en ese nodo utilizando el atributo con la mayor disminución.

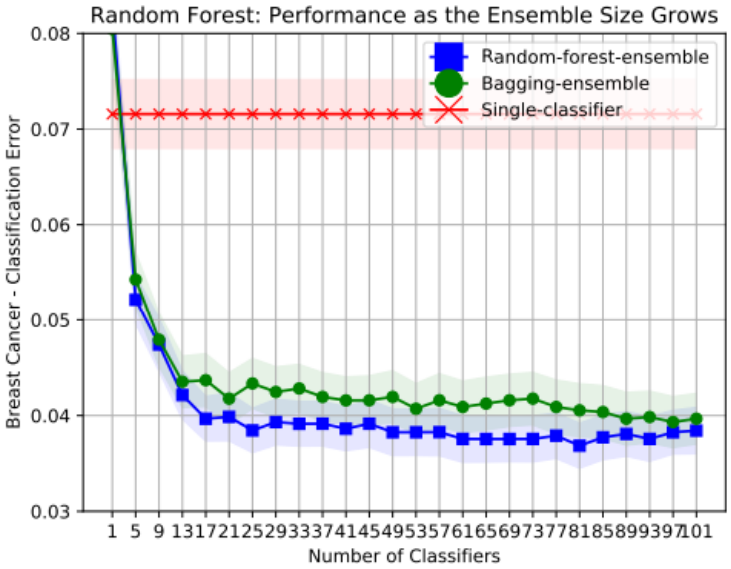
Este proceso introduce variabilidad en los clasificadores y decorrelaciona los errores cometidos.



Random forest es más eficaz que bagging a la hora de generar árboles diversos. El proceso de construcción de cada árbol es más rápido, y debido a la aleatorización de las divisiones, los árboles son más grandes. Los datos fuera de bootstrap (OOB) se utilizan para estimar el error de prueba de forma gratuita. Además, es bastante insensible al parámetro m (a menudo, $m = \sqrt{D}$).

Random forest ofrece mejores resultados que el bagging y el árbol único.





III.6.4. Resumen

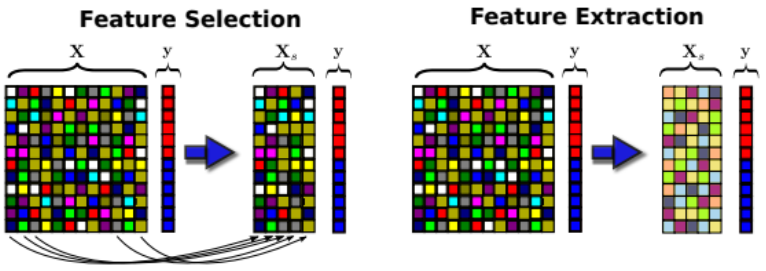
- Los métodos de ensamblaje pueden mejorar significativamente el rendimiento de un clasificador base, aunque pueden reducir la interpretabilidad.
- Un buen ensemble debe contener clasificadores que cometan errores independientes pero que también sean relativamente precisos (mejores que los aleatorios).
- El bagging introduce diversidad en los clasificadores de base mediante el ajuste de cada clasificador a una muestra bootstrap de los datos.
- El cambio de clase invierte aleatoriamente las etiquetas de los datos.
- El random forest añade más aleatoriedad al proceso de crecimiento del árbol.

El método por defecto basado en conjuntos de clasificadores es random forest.

III.7. Selección de atributos

La selección de atributos se realiza con técnicas que identifican unos pocos atributos relevantes del conjunto total de atributos que son más útiles para abordar una tarea

de clasificación. También se denomina como selección de variables o selección de atributos. Es diferente a las técnicas de extracción de características, que crean nuevas características a partir de funciones de las características originales, como por ejemplo: PCA.



La selección de atributos se realiza por los siguientes motivos:

- Simplificación de los clasificadores para facilitar su interpretación.
- Tiempos de entrenamiento más cortos.
- Aliviar la maldición de la dimensionalidad.
- Mejorar la generalización reduciendo el sobreajuste.
- Simplificar la recogida de datos.

En las técnicas de selección de rasgos, el objetivo es identificar qué rasgos son

- **Relevantes:** Características que son estrictamente necesarias para obtener buenos resultados en la tarea de predicción. Sin ellas, el error de generalización aumenta.
- **Irrelevantes:** Características innecesarias para la tarea de clasificación. El error de generalización no se reduce al considerarlas.
- **Redundantes:** Características que se vuelven irrelevantes en presencia de otras características relevantes para la clasificación.

El objetivo de las técnicas de selección de características es identificar las características relevantes y excluir las irrelevantes y redundantes.

Hay dos posibles escenarios que pueden surgir:

1. **Gran número de datos, pero con pocas dimensiones/atributos:** al utilizar un método de selección de atributos, en esta situación va a ayudar algo, pero no demasiado a la hora de mejorar el error de clasificación.
2. **Número pequeño de datos y muchas dimensiones:** en este escenario, la selección de atributos será clave. Esto se puede dar en estudios clínicos con pocos pacientes, pero muchos genes de los que se mide la expresión. La selección de atributos puede ser interesante para identificar un número reducido de genes que sean relevantes para predecir la enfermedad, obteniendo un subset interpretable que ayude a los médicos.

En función del número de variables consideradas conjuntamente, se distinguen:

- **Métodos univariantes, clasificación de variables**: Consideran las variables de entrada una a una. Puede ignorar variables importantes.
- **Métodos multivariantes, selección de subconjuntos de variables**: considera grupos de variables juntos. No es posible probar todas las combinaciones

Basado en el uso de un clasificador en la selección de características:

- **Filtro**: selecciona un subconjunto de variables independientemente del clasificador que luego las utilizará. A menudo, métodos univariantes.
- **Wrapper**: selecciona un subconjunto de variables teniendo en cuenta la precisión del clasificador que las utilizará.
- **Embebido**: el método de selección de características se incorpora al proceso de entrenamiento del clasificador (por ejemplo, árboles de clasificación o centroides encogidos).

III.7.1. Métodos de filtrado

Este método se basa en encontrar un subconjunto de variables de las que se espera un buen rendimiento utilizando únicamente los datos observados X e y. En el **ranking de variables**, se calcula el estadístico J para cada atributo y se mantienen los valores por encima de un umbral. Este método es univariable al mirarse cada atributo de forma individual.

Uno de los scores que se puede utilizar en variable ranking es ANOVA. Se realiza el análisis de la varianza como prueba estadística para evaluar si las medias de varios grupos son iguales o no. Se obtiene el F-score. F será grande si la variabilidad entre grupos es grande en relación con la variabilidad dentro de los grupos (poco probable si todas las medias son iguales).

La ventaja de los filtros es que son rápidos de computar, y permite la selección genérica de variables sin estar condicionado por ningún clasificador. Requiere un paso de preprocesado para reducir el espacio de dimensionalidad y el sobreajuste. No obstante, filtros simples pueden ignorar variables que solo son relevantes en conjunto con otras.

III.7.2. Wrappers

Los wrappers permiten evaluar la calidad de un subconjunto de características en función de la precisión de un clasificador concreto. Implica evaluar muchos subconjuntos diferentes. Es un proceso interativo en el que, en cada iteración, se generan y evalúan varios subconjuntos de atributos. El éxito de cada subconjunto determina qué subconjuntos se probarán a continuación. La selección de características pasa a formar parte del entrenamiento del clasificador.

La ventaja es que puede utilizarse con cualquier clasificador, y la selección de variables está sintonizada para un clasificador dado. Además, puede encontrar atributos que sean relevantes conjuntamente. No obstante, es muy costoso computacionalmente, y requiere dejar de lado los datos de validación. Las características D dan 2^D posibles subconjuntos, por lo que es necesario una estrategia de búsqueda: selección secuencial hacia delante, eliminación recursiva hacia atrás, algoritmos genéticos, annealing simulado, etc.

III.7.3. Métodos embebidos

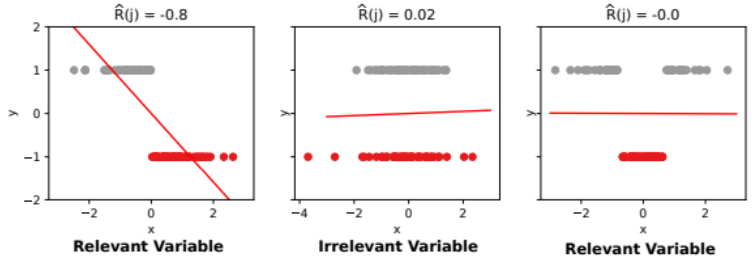
En este caso, el proceso de selección de características se realiza durante el entrenamiento del clasificador. Ejemplos son árboles de decisión, centroides encogidos más próximos o LASSO generalizado. Este último es un modelo discriminativo que contiene el hiperparámetro w.

La selección de variables se realiza de forma gratuita mientras se entrena el clasificador. De esta forma, pueden encontrarse atributos que sean relevantes conjuntamente y utilizar conocimientos sobre el algoritmo de clasificación específico. Además, suele ser más barato que los métodos wrappers y tiene un buen rendimiento empírico en general. La desventaja es que suele requerir el ajuste de hiperparámetros mediante validación cruzada, y las características seleccionadas son específicas del clasificador.

III.7.4. Medidas de dependencia: correlación lineal

Es importante medir la dependencia de un atributo con la etiqueta de clase. Si las dependencias son grandes, es bastante probable que el atributo sea relevante.

La correlación entre dos variables toma valores entre -1 y 1. Esos valores indican que la dependencia es fuerte. El coeficiente al cuadrado toma valores entre 0 y 1, e indica así la fracción de varianza total que se explica mediante una relación lineal simple entre el atributo y la etiqueta de clase.

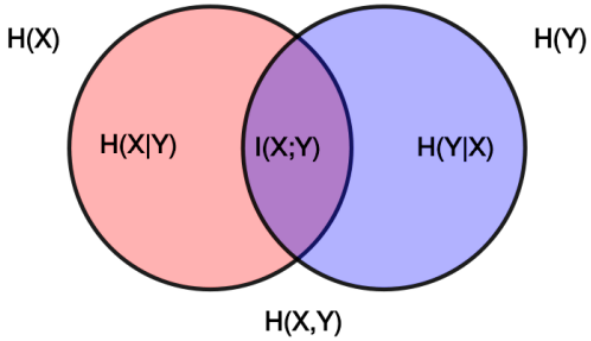


La correlación lineal solo tiene sentido en problemas de clasificación cuando el problema es binario. Además, solo puede captar las dependencias lineales y puede subestimar variables importantes.

III.7.5. Medidas de dependencia: información mutua

La información mutua mide la información que comparten x e y: En qué medida el conocimiento de una de estas variables reduce la incertidumbre sobre la otra. Esto se puede definir de muchas formas, entre las que se incluye la entropía.

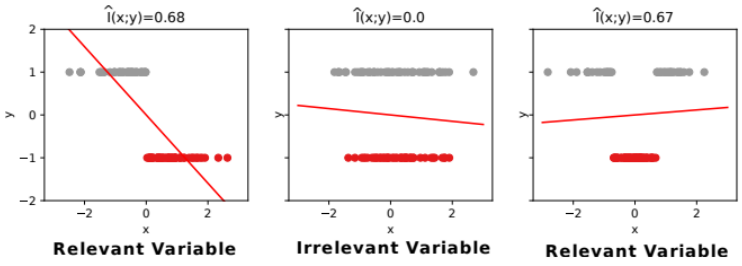
La entropía condicional es la incertidumbre en y después de conocer x.



$$I(x;y) = I(y;x), \quad I(x;y) \geq 0, \quad I(x;y) = 0 \Longleftrightarrow p(x,y) = p(x)p(y)$$

$I(x;y)$ is a natural measure of dependence.

En la práctica, $I(x;y)$ capta las dependencias no lineales, a diferencia de la correlación lineal. y no tiene por qué ser binario (asignado a un conjunto entero). En el caso discreto, I(x; y) puede estimarse simplemente contando. En el caso continuo, es necesario estimar la densidad (muy difícil). A menudo, las variables se discretizan o se utilizan técnicas basadas en los vecinos más próximos para su estimación.



III.7.6. Métodos de filtrado avanzado

Un enfoque heurístico común es una búsqueda secuencial que considera las características una por una para añadirlas/eliminarlas. Dos enfoques secuenciales populares son:

- **Selección hacia delante:** añade la característica que maximiza $J(x., j)$
- **Eliminación hacia atrás:** elimina la característica que minimiza J.

III.7.6.1. Minimum-Redundancy Maximum-Relevance

Éste es un criterio que favorece las características relevantes que no son redundantes

$$J(\mathbf{x}., j) = Relevance - Redundancy$$

Se favorecen atributos relevantes, penalizando aquellos que, aunque sean relevantes, sean redundantes, estando altamente correlacionados con los atributos ya elegidos. De esta forma, una característica que sea muy relevante pero también muy redundante con las características ya seleccionadas en S recibirá una puntuación baja.

Variable ranking utiliza $I(x., j; y)$. En MRMR, el ranking de variables se utiliza en primer lugar para conservar solo el 20 % de las características y reducir el coste computacional.

Para el mismo número de características, MRMR ofrece resultados similares o mejores que un enfoque de clasificación de variables simple.

III.7.6.2. Información mutua conjunta

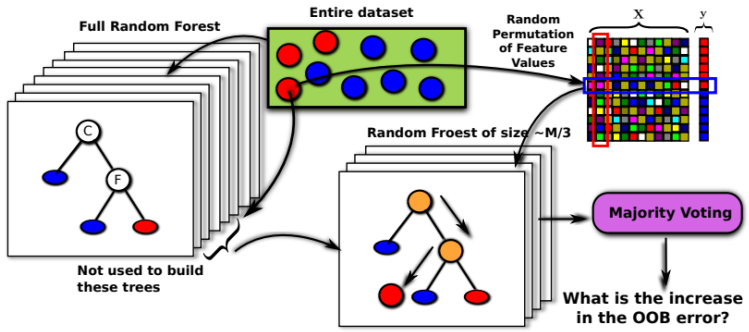
El criterio de información mutua conjunta favorece las características que son relevantes cuando se consideran junto con las características ya seleccionadas.

En JMI, el ranking de variables se utiliza en primer lugar para conservar solo el 20 % de las características y reducir el coste computacional.

Para el mismo número de características, JMI ofrece resultados similares o mejores que una simple clasificación de variables.

III.7.7. Random forest para selección de atributos

Se puede utilizar random forest para calcular la importancia de cada atributo. Si un atributo es muy importante, su aleatoriedad va a incrementar el error de OOB en una gran cantidad.



La ventaja es que se obtiene gratuitamente tras construir un random forest sobre los datos. Se pueden identificar los atributos que solo son relevantes cuando se consideran juntos, teniendo en cuenta las interacciones multivariantes. Además, devuelve una medida cuantitativa de la importancia (aumento del error), siendo relativamente barato de obtener. No obstante, puede sobreestimar la importancia de las variables predictoras

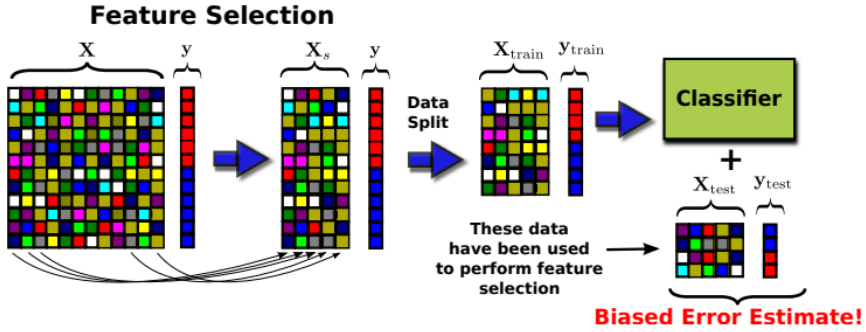
correlacionadas debido a la preferencia de las variables predictoras correlacionadas en las primeras divisiones. También puede sobreestimar la importancia de variables categóricas con un gran número de categorías debido a su preferencia para dividir los datos.

III.7.8. Sesgos en el proceso de selección de atributos

Considerar un número reducido de características mejora la interpretabilidad, pero puede reducir el rendimiento de la predicción.

Muchos trabajos presentan estimaciones de rendimiento demasiado optimistas debido al uso de datos de prueba para llevar a cabo la selección de características.

Los datos de prueba no deben utilizarse en ninguna de las etapas de construcción del clasificador, incluidas la selección de características o la normalización de datos.



III.7.9. Ejercicios

1. *En un conjunto de datos con 100.000 características y unos cientos de instancias, es demasiado costoso ejecutar Random Forest para la selección de características. Indique qué enfoques puede seguir para resolver este problema.*

   Se podría hacer una preselección de atributos con variable ranking. De 100.000 podríamos quedarnos con 10.000 y después aplicar ya Random Forest.

III.7.10. Resumen

- La selección de atributos puede utilizarse para identificar un pequeño conjunto de características más adecuadas para abordar una determinada tarea de clasificación.
- Mejora la interpretabilidad del clasificador, simplifica la adquisición de datos y, en ocasiones, también mejora el rendimiento de la predicción.
- La mayoría de las técnicas de selección de características pueden clasificarse en filtros, wrappers y métodos embebidos.
- Los métodos de filtro son sencillos, rápidos y, la mayoría de las veces, suficientes para encontrar un buen conjunto de características para una tarea de clasificación concreta.

- Existe un trade-off entre el número de características consideradas para la predicción y la precisión de la clasificación resultante.
- Hay que tener cuidado para evitar sesgos optimistas en la estimación del rendimiento de predicción de un conjunto de características no utilizando datos de test para la selección de atributos.

Capítulo IV

Modelos lineales, métodos de Kernel y redes neuronales

IV.1. Modelos lineales de regresión

El aprendizaje supervisado es la creación de un modelo que prediga la salida en base a unas entradas, partiendo de un histórico con parejas entrada-salida. Dentro de esto aparece la regresión, donde se quiere predecir un valor continuo.

Los elementos de un problema de aprendizaje supervisado son:

- Datos: parejas entrada-salida
- Características: atributos
- Objetivo: la salida real, las etiquetas, la salida asociada
- Modelo: una función matemática de x a y que proporciona la salida. Puede tener varios parámetros Θ.
- Algoritmo de aprendizaje: procedimiento para la obtención del modelo en base a los datos.

El enfoque más sencillo de la regresión es un modelo constante, ignorando la entrada. Así, se define la salida como una combinación lineal de entradas, creando un modelo lineal. La ventaja es que es simple, robusto, interpretable, fácil de entrenar y fácil de predecir. La desventaja es que sea demasiado limitado y un subajuste a los datos.

IV.1.1. Regresión lineal 1-dimensional

El caso más simple tiene un solo dato de valores reales. El modelo lineal simple sería una línea recta con parámetros $\Theta = \{b, w\}$, siendo b el sesgo y w la pendiente de la recta. Así, el modelo se definiría como

$$f_\Theta = b + wx$$

Ejercicio: dado este modelo lineal con $b = 1$ y $w = 2$, computa la salida del modelo para $x = 2$ y para $x = -1$.

$$1 + 2 \cdot 2 = 1 + 4 = 5$$

$$1 + 2 \cdot -1 = 1 - 2 = -1$$

Se necesita un procedimiento que determine el sesgo y la pendiente para optimizar la calidad del modelo. Se pueden utilizar dos perspectivas para definir la calidad:

- En base al error: medida sobre cómo de bien se ajusta el modelo a los datos de entrenamiento
- En base a la complejidad: un término de regulación que penaliza la complejidad del modelo

Se define el residuo como la diferencia entre lo que se debería haber predicho y lo que el modelo ha predicho. El error cuadrático sería la esperanza de los residuos al cuadrado. Es decir, sobre el conjunto de datos de entrenamiento, se mide el error, se suma y se eleva al cuadrado. Esto es una medida automática de cuánto se confunde el modelo. También se puede calcular el error absoluto con el valor absoluto en lugar del error cuadrado.

Ejercicio: calcular MAE y MSE del modelo anterior con $b = 1$ y $w = 2$ con los pares (2,4) y (-1,1).

$$f_\Theta = 1 + 2 \cdot 2 = 1 + 4 = 5 \neq 4$$

$$f_\Theta = 1 + 2 \cdot -1 = 1 - 2 = -1 \neq 1$$

$$r = -1, 2$$

$$MAE = 1/2 \cdot (1 + 2) = 1/2 \cdot 3 = 1.5$$

$$MSE = 1/2 \cdot (1^2 + 2^2) = 1/2 \cdot (1 + 4) = 2.5$$

IV.1.1.1. Optimización

Optimizar hace referencia a encontrar la mejor solución posible respecto a algún criterio.

En un modelo lineal, hay que elegir una métrica que generalmente es el error cuadrático. Esto se debe a que es diferenciable (se puede calcular la derivada) y tiene sentido geométricamente. En el algoritmo, la optimización sería los valores de b y w que hacen que el error cuadrático sea mínimo.

En resumen, la línea de regresión por mínimos cuadrados se obtiene resolviendo

$$\min_{b,w \in \mathbb{R}} \left\{ \frac{1}{N} \sum_{i=1}^{N} \big(y_i - (b + w x_i)\big)^2 \right\}$$

Los elementos auxiliares son:

$$\bar{y} = \frac{1}{N} \sum_{i=1}^{N} y_i \quad \text{(Mean Target)}, \quad \bar{x} = \frac{1}{N} \sum_{i=1}^{N} x_i \quad \text{(Mean Feature)},$$

$$\hat{y}_i = y_i - \bar{y} \quad \text{(Centred Target)}, \quad \hat{x}_i = x_i - \bar{x} \quad \text{(Centred Feature)}.$$

Ejercicio: dado los siguientes pares de datos (0,4), (1,6), (2,4), (3,6), (4,8), computa:

- El valor de $\bar{x}$ y $\bar{y}$:

$$\bar{x} = (0+1+2+3+4)/5 = 10/5 = 2$$

$$\bar{y} = (4+6+4+6+8)/5 = 28/5 = 5.6$$

- El valor de $\hat{x}_i$ y $\hat{y}_i$:

$$\hat{x}_1 = 0-2 = -2 \quad \hat{y}_1 = 4-5.6 = -1.6$$

$$\hat{x}_2 = 1-2 = -1 \quad \hat{y}_2 = 6-5.6 = 0.4$$

$$\hat{x}_3 = 2-2 = 0 \quad \hat{y}_3 = 4-5.6 = -1.6$$

$$\hat{x}_4 = 3-2 = 1 \quad \hat{y}_4 = 6-5.6 = 0.4$$

$$\hat{x}_5 = 4-2 = 2 \quad \hat{y}_5 = 8-5.6 = 2.4$$

- El valor de w^* y b^*:

$$w^* = \frac{\sum_{i=1}^N \hat{x}_i \hat{y}_i}{\sum_{i=1}^N \hat{x}_i \hat{x}_i} = \frac{3.2 - 0.4 + 0 + 0.4 + 4.8}{4+1+0+1+4} = \frac{8}{10} = 0.8$$

$$b^* = \bar{y} - w^* \bar{x} = 5.6 - 0.8 \cdot 2 = 4$$

- El valor de MSE:

$$MSE = \frac{1}{N} \sum_{i=1}^{N} (y_i - (b + w x_i))^2$$

$$\frac{1}{5} \cdot ((4-(4+0.8\cdot 0)^2 + 6-(4+0.8\cdot 1)^2 + 4-(4+0.8\cdot 2)^2 + 6-(4+0.8\cdot 3)^2 + 8-(4+0.8\cdot 4))^2$$

$$\frac{1}{5} \cdot (0 + 1.44 + 2.56 + 0.16 + 0.64) = \frac{1}{5} \cdot 4.8 = 0.96$$

IV.1.2. Regresión lineal múltiple

Por simplicidad, $\mathcal{X} = \mathbb{R}^d$. Los datos se representan como $D = \{(x_i, y_i)\}_{i=1}^N$, donde $x_i = (x_{i,1}, x_{i,2}, \ldots, x_{i,d}) \in \mathbb{R}^d$ y $y_i \in \mathbb{R}$.

El modelo lineal correspondiente es un hiperplano, con parámetros $\theta = \{b, w\}$.

- $b \in \mathbb{R}$ es el término de intercepción o sesgo.
- $w = (w_1, w_2, \ldots, w_d) \in \mathbb{R}^d$ es el vector normal del hiperplano.
- El modelo se define como:

$$f_\theta(\mathbf{x}) = b + w^\top \mathbf{x} = b + \sum_{i=1}^{d} w_i x_i.$$

El algoritmo de aprendizaje determinará b y w utilizando D.

Ejercicio: dado un modelo lineal bidimensional con parámetros $\Theta = \{b, \mathbf{w}\}$ con $b = 1$ y $\mathbf{w} = (1, 2)^T$. Computa la salida del modelo para $\mathbf{x} = (1, 1)^T$:

$$1 + (1 \cdot 1 + 1 \cdot 2) = 4$$

Computa la salida del modelo para $\mathbf{x} = (-1, 0)^T$:

$$1 + (1 \cdot -1 + 0 \cdot 2) = 0$$

IV.1.2.1. Ecuaciones lineales

Se necesita un procedimiento para determinar el sesgo b y el vector $\mathbf{w}$.

Un primer enfoque es intentar igualar todos los pares entrada-salida (x_i, y_i), $i = 1, \ldots, N$. Específicamente:

$$\begin{cases} b + \mathbf{w}^T x_1 = y_1 \\ b + \mathbf{w}^T x_2 = y_2 \\ \vdots \\ b + \mathbf{w}^T x_N = y_N \end{cases} \equiv \begin{cases} b + w_1 x_{1,1} + w_2 x_{1,2} + \cdots + w_d x_{1,d} = y_1 \\ b + w_1 x_{2,1} + w_2 x_{2,2} + \cdots + w_d x_{2,d} = y_2 \\ \vdots \\ b + w_1 x_{N,1} + w_2 x_{N,2} + \cdots + w_d x_{N,d} = y_N \end{cases}$$

La siguiente notación matricial puede simplificar las ecuaciones:

$$\mathbf{X} = \begin{pmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,d} \\ x_{2,1} & x_{2,2} & \cdots & x_{2,d} \\ \vdots & \vdots & \ddots & \vdots \\ x_{N,1} & x_{N,2} & \cdots & x_{N,d} \end{pmatrix}; \quad \tilde{\mathbf{X}} = \begin{pmatrix} 1 & x_{1,1} & \cdots & x_{1,d} \\ 1 & x_{2,1} & \cdots & x_{2,d} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{N,1} & \cdots & x_{N,d} \end{pmatrix}; \quad \mathbf{y} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{pmatrix}; \quad \tilde{\mathbf{w}} = \begin{pmatrix} b \\ w_1 \\ \vdots \\ w_d \end{pmatrix},$$

donde $\mathbf{X} \in \mathbb{R}^{N \times d}$ es la matriz de datos, $\tilde{\mathbf{X}} \in \mathbb{R}^{N \times (d+1)}$ es la matriz de datos con un término constante, $\mathbf{y} \in \mathbb{R}^N$ es el vector de objetivos y $\tilde{\mathbf{w}} \in \mathbb{R}^{d+1}$ es el vector de pesos que incluye el intercepto.

El sistema de ecuaciones se convierte en $\tilde{\mathbf{X}}\tilde{\mathbf{w}} = \mathbf{y}$. Como $\tilde{\mathbf{X}} \in \mathbb{R}^{N \times (d+1)}$, $\tilde{\mathbf{w}} \in \mathbb{R}^{d+1}$ y $\mathbf{y} \in \mathbb{R}^N$, tenemos N ecuaciones y $d + 1$ incógnitas. Normalmente, $N >> d + 1$ y el sistema está **sobredeterminado**.

IV.1.2.2. Calidad del modelo

Se define con el término de error y de complejidad, al igual que en el caso de una sola dimensión. Los residuos son la desviación de la salida con respecto a la predicción. El error cuadrático y el error absoluto se definen igual que antes.

Para el i-ésimo patrón, el **residual** se define como:

$$r_i = y_i - f_\theta(\mathbf{x}_i) = y_i - (b + \mathbf{w}^\intercal \mathbf{x}_i).$$

El **Error Cuadrático Medio** se calcula como:

$$\text{MSE}(b, \mathbf{w}) = \mathbb{E}\left[R^2\right] \approx \frac{1}{N}\sum_{i=1}^{N}\left(y_i - (b + \mathbf{w}^\intercal \mathbf{x}_i)\right)^2.$$

$$\text{MAE}(b, \mathbf{w}) = \mathbb{E}\left[|R|\right] \approx \frac{1}{N}\sum_{i=1}^{N}\left|y_i - (b + \mathbf{w}^\intercal \mathbf{x}_i)\right|.$$

Ejercicio: dado un modelo lineal bidimensional con parámetros $\Theta = \{b, \mathbf{w}\}$, con $b = 1$ y $\mathbf{w} = (1, 2)^T$, y los datos siguientes datos para $x_{i,1}$, $x_{i,2}$ y y_i, computa MAE y MSE. Para un modelo lineal bidimensional:

$$f(x_1, x_2) = b + w_1 x_1 + w_2 x_2$$

En el enunciado nos dicen b y w, por lo que la fórmula queda en:

$$f(x_1, x_2) = 1 + 1x_1 + 2x_2$$

Ahora tenemos dos datos, por lo que calculamos las salidas predichas:

$$f_1 = 1 + 1 + 2 = 4$$

$$f_2 = 1 - 1 + 0 = 0 \neq 2$$

Los residuos son 0 y 2. Por tanto, los residuos cuadrados son 0 y 4. Calculando la media, $MAE = 1$ y $MSE = 2$.

IV.1.2.3. Optimización basada en gradiente: problemas multidimensionales

En muchas dimensiones, la derivada se generaliza con el gradiente, que es un vector de derivadas parciales. El gradiente define el hiperplano tangente.

IV.1.2.4. Entrenamiento de un modelo lineal

Usualmente se utiliza como función de error el MSE al ser diferenciable y corresponderse a la distancia entre los vectores de predicción y de salida esperada. Se busca minimizar este error, calculando para cada entrada la salida correspondiente y viendo el error.

El problema de optimización para encontrar los parámetros óptimos se formula como:

$$\min_{\substack{b\in\mathbb{R}\\ \mathbf{w}\in\mathbb{R}^d}} \left\{\text{MSE}(b, \mathbf{w})\right\} = \min_{\substack{b\in\mathbb{R}\\ \mathbf{w}\in\mathbb{R}^d}} \left\{\frac{1}{N}\sum_{i=1}^{N}\left(y_i - (b + \mathbf{w}^\intercal \mathbf{x}_i)\right)^2\right\} \equiv \min_{\tilde{\mathbf{w}}\in\mathbb{R}^{d+1}} \left\{(\mathbf{y} - \tilde{\mathbf{X}}\tilde{\mathbf{w}})^\top(\mathbf{y} - \tilde{\mathbf{X}}\tilde{\mathbf{w}})\right\}.$$

Para encontrar el punto óptimo $\tilde{\mathbf{w}}^*$, calculamos el gradiente e igualamos a cero:

$$\nabla_{\tilde{\mathbf{w}}}\text{MSE}(\tilde{\mathbf{w}})\big|_{\tilde{\mathbf{w}}=\tilde{\mathbf{w}}^*} = \mathbf{0} \implies 2\tilde{\mathbf{X}}^\top(\mathbf{y} - \tilde{\mathbf{X}}\tilde{\mathbf{w}}^*) = \mathbf{0}$$

Esto conduce al sistema de ecuaciones normales:

$$\tilde{\mathbf{X}}^\top\mathbf{y} - \tilde{\mathbf{X}}^\top\tilde{\mathbf{X}}\tilde{\mathbf{w}}^* = \mathbf{0} \implies \tilde{\mathbf{X}}^\top\tilde{\mathbf{X}}\tilde{\mathbf{w}}^* = \tilde{\mathbf{X}}^\top\mathbf{y}$$

Finalmente, la solución óptima (si $\tilde{\mathbf{X}}^\top\tilde{\mathbf{X}}$ es invertible) es:

$$\tilde{\mathbf{w}}^* = (\tilde{\mathbf{X}}^\top\tilde{\mathbf{X}})^{-1}\tilde{\mathbf{X}}^\top\mathbf{y} \quad \text{(Mínimos Cuadrados)}$$

Caso especial: cuando $\tilde{\mathbf{X}}^\top\tilde{\mathbf{X}}$ es invertible y $N = d + 1$, se simplifica a:

$$\tilde{\mathbf{w}}^* = \tilde{\mathbf{X}}^\top\mathbf{y}$$

En resumen, el modelo de mínimos cuadrados lineales es la solución del siguiente problema de optimización:

$$\min_{\substack{b\in\mathbb{R}\\ \mathbf{w}\in\mathbb{R}^d}} \left\{ \frac{1}{N}\sum_{i=1}^{N} \left(y_i - (b + \mathbf{w}^\intercal\mathbf{x}_i)\right)^2 \right\}.$$

La solución óptima viene dada por:

$$\begin{pmatrix} b^* \\ \mathbf{w}^* \end{pmatrix} = \tilde{\mathbf{w}}^* = \tilde{\mathbf{X}}^\dagger\mathbf{y} = \begin{bmatrix} \mathbf{1} & \mathbf{X} \end{bmatrix}^\dagger \mathbf{y},$$

donde:

- $\tilde{\mathbf{X}} = \begin{bmatrix} \mathbf{1} & \mathbf{X} \end{bmatrix}$ es la matriz de diseño aumentada
- $\dagger$ denota la pseudoinversa de Moore-Penrose
- $\mathbf{1}$ es un vector columna de unos

IV.1.3. Resumen: modelos lineales para regresión

Un problema de regresión es un problema supervisado con objetivos continuos. Un modelo de regresión sencillo pero útil es el modelo lineal. La predicción es una combinación lineal de las características. Para entrenar el modelo lineal, se suele resolver un problema de optimización. El MSE suele utilizarse para medir la calidad del modelo. Es una elección natural. El problema resultante puede resolverse de forma cerrada utilizando el pseudoinverso de la matriz de datos.

IV.2. Modelos lineales de clasificación

Un problema de clasificación es un problema de aprendizaje supervisado en el que las salidas son discretas. Ejemplos son predecir si un paciente padece o no una determinada enfermedad en función de datos médicos, distinguir la especie de un pez capturado utilizando los datos proporcionados por varios sensores o discernir el tipo de objeto que aparece en una imagen.

Los elementos de un problema de aprendizaje supervisado son:

- Datos: conjunto de pares entrada-salida
- Características: vector de atributos (variables independientes/de entrada, covariables...)
- Etiquetas: objetivo (variable dependiente, resultado...)
- Modelo: asignación del espacio de entrada al de salida
- Algoritmo de aprendizaje: procedimiento para obtener un modelo basado en los datos

IV.2.1. Clasificación binaria lineal

El escenario de clasificación más importante es cuando M = 2 (clasificación binaria). Si M > 2, existen técnicas de codificación para transformar el problema en varios subproblemas binarios. Las clases suelen denominarse C0 y C1, y se representan con una codificación 0/1 (o -1/1). Las etiquetas se transforman en:

$$t_i = \begin{cases} 0 & \text{if } y_i = \mathcal{C}_0 \\ 1 & \text{if } y_i = \mathcal{C}_1 \end{cases}$$

Los enfoques más simples de clasificación son:

- Ignorar la entrada: modelo constante (normalmente, clase mayoritaria).
- Definir la salida como una combinación lineal de las entradas más una transformación: modelo lineal. Es simple, robusto (varianza pequeña), interpretable, fácil de entrenar y fácil de predecir. No obstante, tiene flexibilidad limitada y un ajuste insuficiente (gran sesgo).

Por simplicidad, $\mathcal{X} = \mathbb{R}^d$. Los datos se convierten en $\mathcal{D} = \{(x_i, t_i)\}_{i=1}^N$, donde $x_i = (x_{i,1}, x_{i,2}, \ldots, x_{i,d}) \in \mathbb{R}^d$ y $t_i \in \{0, 1\}$. El modelo lineal correspondiente es un hiperplano, con parámetros $\boldsymbol{\theta} = \{b, \mathbf{w}\}$.

- $b \in \mathbb{R}$ es el término de intercepción o sesgo.
- $\mathbf{w} = (w_1, w_2, \ldots, w_d) \in \mathbb{R}^d$ es el vector normal del hiperplano.

- El modelo se define como:

$$f_{\boldsymbol{\theta}}(\mathbf{x}) = \begin{cases} 0 & \text{si } b + \mathbf{w}^\intercal \mathbf{x} < 0, \\ 1 & \text{si } b + \mathbf{w}^\intercal \mathbf{x} \geq 0. \end{cases}$$

- El hiperplano divide el espacio en dos mitades, una para la clase $\mathcal{C}_0$ y la otra para la clase $\mathcal{C}_1$.

El **algoritmo de aprendizaje** determinará b y $\mathbf{w}$ utilizando $\mathcal{D}$.

Ejercicio: dado un modelo de clasificación lineal binario bidimensional con parámetros $\theta = \{b, \mathbf{w}\}$, con $b = 1$ y $\mathbf{w} = (1, 2)^T$:

- Calcula la salida del modelo para $\mathbf{x}_1 = (1, 1)$:

$$1 + 1 \cdot 1 + 2 \cdot 1 = 4 \rightarrow f_\theta(\mathbf{x}) = \mathcal{C}_1$$

- Calcula la salida del modelo para $\mathbf{x}_2 = (1, -2)$:

$$1 + 1 \cdot 1 + 2 \cdot (-2) = -2 \rightarrow f_\theta(\mathbf{x}) = \mathcal{C}_0$$

- Calcula la salida del modelo para $\mathbf{x}_3 = (0, 0)$:

$$1 + 1 \cdot 0 + 2 \cdot 0 = 1 \rightarrow f_\theta(\mathbf{x}) = \mathcal{C}_1$$

IV.2.1.1. Calidad del modelo

Se necesita un procedimiento para determinar el sesgo b y el vector w, optimizando la calidad del modelo. Hay que definir la calidad del modelo. Normalmente desde dos puntos de vista:

- Ajuste: Un término de aptitud $\mathcal{F}_\mathcal{D}(\theta)$ mide lo bien que el modelo se ajusta a los datos de entrenamiento.
- Complejidad: Un término de regularización $\mathcal{R}(\theta)$ penaliza la complejidad del modelo

Predicción Correcta Para el i-ésimo patrón,

$$c_i = \begin{cases} 0 & \text{si } t_i \neq f_0(\mathbf{x}_i) \\ 1 & \text{si } t_i = f_0(\mathbf{x}_i) \end{cases} = \begin{cases} 0 & \text{si } (t_i = 0, b + \mathbf{w}^\intercal \mathbf{x}_i \geq 0) \text{ o } (t_i = 1, b + \mathbf{w}^\intercal \mathbf{x}_i < 0), \\ 1 & \text{si } (t_i = 0, b + \mathbf{w}^\intercal \mathbf{x}_i < 0) \text{ o } (t_i = 1, b + \mathbf{w}^\intercal \mathbf{x}_i \geq 0). \end{cases}$$

Precisión $\text{Acc}(b, \mathbf{w}) = \mathbb{E}[C] \approx \frac{1}{N} \sum_{i=1}^{N} c_i$.

Ejercicio: dado un modelo de clasificación binario lineal bidimensional con los parámetros $\theta = (b, \mathbf{w})$ con $b = 1$ y $\mathbf{w} = (1, 2)^T$ y los siguientes datos, computa la precisión:

$$(1, 1; 1) \rightarrow 1 + 1 \cdot 1 + 1 \cdot 2 = 4 \rightarrow 1 = 1$$

$$(1, -2; 0) \rightarrow 1 + 1 \cdot 1 + 2 \cdot -2 = -2 \rightarrow 0 = 0$$
$$(0, 0; 0) \rightarrow 1 + 1 \cdot 0 + 2 \cdot 0 = 1 \rightarrow 1 \neq 0$$
$$Accuracy = 2/3 \approx 66.67$$

La opción más habitual para evaluar el modelo es la **precisión**. Es una medida sensata e intuitiva. No es convexa, no es diferenciable y es discontinua. Optimizar la precisión es un problema que (en general) no puede abordarse directamente.

Una idea alternativa podría ser entrenar un **modelo de regresión lineal** con etiquetas -1/1. La etiqueta predicha se determina tomando el signo de la salida. El problema de esto es que con datos asimétricos, el modelo minimiza el error cuadrático, pudiendo mover la frontera. Se penaliza tener valores demasiado positivos o demasiado negativos, intentando ajustarse siempre a los valores 1 y -1.

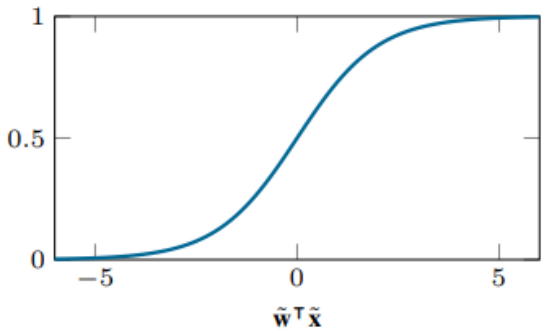
Por tanto, se necesita una medida de calidad diferente que debería ser más fácil de optimizar que la precisión. No debería penalizar los puntos alejados del límite de decisión (pero en el lado correcto). Un enfoque probabilístico puede ser útil. En particular, el marco principal es la regresión logística. El modelo lineal se utiliza para estimar la probabilidad posterior de una clase. Se utiliza una transformación sigmoidea.

Denotando por $\tilde{x} = [1, x]$ y por $\tilde{w} = [b, w]$, las probabilidades posteriores se definen como:

$$p(C_1 \mid \tilde{x}; \tilde{w}) = \sigma(\tilde{w}^T \tilde{x}) = \frac{1}{1 + e^{-\tilde{w}^T \tilde{x}}},$$

$$p(C_0 \mid \tilde{x}; \tilde{w}) = 1 - p(C_1 \mid \tilde{x}; \tilde{w}) = 1 - \frac{1}{1 + e^{-\tilde{w}^T \tilde{x}}} = \frac{e^{-\tilde{w}^T \tilde{x}}}{1 + e^{-\tilde{w}^T \tilde{x}}} = \frac{1}{1 + e^{\tilde{w}^T \tilde{x}}} = \sigma(-\tilde{w}^T \tilde{x}).$$

- Si $\tilde{w}^T \tilde{x} < 0 \Rightarrow p(C_1 \mid \tilde{x}; \tilde{w}) < 0.5$: Se predice la clase C_0.
- Si $\tilde{w}^T \tilde{x} \geq 0 \Rightarrow p(C_1 \mid \tilde{x}; \tilde{w}) \geq 0.5$: Se predice la clase C_1.



Ejercicio: dado un modelo de clasificación binario lineal bidimensional con los parámetros $\theta = (b, \mathbf{w})$ con $b = 1$ y $\mathbf{w} = (1, 2)^T$ y los siguientes datos, computa la precisión:

- Probabilidad de $\mathbf{x}_1$ pertenecer a la clase $\mathcal{C}_1$ para $\mathbf{x}_1 = (1, 1)^T$.

$$1 + 1 \cdot 1 + 2 \cdot 1 = 4$$
$$1/(1 + e^{-4}) = 0.98$$

- Probabilidad de $\mathbf{x}_2$ pertenecer a la clase $\mathcal{C}_1$ para $\mathbf{x}_2 = (1, -2)^T$.

$$1 + 1 \cdot 1 + 2 \cdot -2 = -2$$
$$1/(1 + e^2) = 0.119$$

- Probabilidad de $\mathbf{x}_3$ pertenecer a la clase $\mathcal{C}_0$ para $\mathbf{x}_3 = (0, 0)^T$.

$$1 + 1 \cdot 0 + 2 \cdot 0 = 1$$
$$1 - (1/(1 + e^{-1})) = 0.2689$$

IV.2.1.2. Optimización: Máxima verosimilitud y entropía cruzada

La verosimilitud de los datos es una elección común para cuantificar la calidad de un modelo probabilístico:

$$\mathcal{L}(\mathcal{D}; \tilde{\mathbf{w}}) = \prod_{i=1}^{N} p(t_i|\tilde{\mathbf{x}}_i; \tilde{\mathbf{w}}) = \prod_{i=1}^{N} \underbrace{p(C_0|\tilde{\mathbf{x}}_i; \tilde{\mathbf{w}})^{1-t_i} p(C_1|\tilde{\mathbf{x}}_i; \tilde{\mathbf{w}})^{t_i}}_{\begin{cases} p(C_0|\tilde{\mathbf{x}}_i; \tilde{\mathbf{w}}) & \text{si } t_i = 0, \\ p(C_1|\tilde{\mathbf{x}}_i; \tilde{\mathbf{w}}) & \text{si } t_i = 1. \end{cases}}$$

El error de Entropía Cruzada (CE) se define como el logaritmo negativo de la verosimilitud:

$$CE(\tilde{\mathbf{w}}) = -\log \mathcal{L}(\mathcal{D}; \tilde{\mathbf{w}})$$
$$= \sum_{i=1}^{N} \left(-(1 - t_i) \log(p(C_0|\tilde{\mathbf{x}}_i; \tilde{\mathbf{w}})) - t_i \log(p(C_1|\tilde{\mathbf{x}}_i; \tilde{\mathbf{w}})) \right)$$
$$= \sum_{i=1}^{N} \left(-(1 - t_i) \log(1 - \sigma(\tilde{\mathbf{w}}^T \tilde{\mathbf{x}}_i)) - t_i \log(\sigma(\tilde{\mathbf{w}}^T \tilde{\mathbf{x}}_i)) \right).$$

Ejercicio: dado un modelo de clasificación binario lineal bidimensional con los parámetros $\theta = (b, \mathbf{w})$ con $b = 1$ y $\mathbf{w} = (1, 2)^T$ y los siguientes datos, computa la precisión:

$$(1, 1; 1) \rightarrow 1 + 1 \cdot 1 + 1 \cdot 2 = 4$$
$$p1 = 1/(1 + e^{-4}) = 0.98$$
$$p0 = 1 - 0.98 = 0.02$$
$$\mathcal{L} = (0.98)^1 \cdot (0.02)^0 = 0.98$$

$$(1, -2; 0) \rightarrow 1 + 1 \cdot 1 + 2 \cdot -2 = -2$$
$$p1 = 1/(1 + e^2) = 0.119$$
$$p0 = 1 - 0.119 = 0.881$$
$$\mathcal{L} = (0.881)^1 \cdot (0.119)^0 = 0.881$$

$$(0,0;0) \rightarrow 1 + 1 \cdot 0 + 2 \cdot 0 = 1$$
$$p1 = 1/(1 + e^{-1}) = 0.7311$$
$$p0 = 1 - 0.2689 = 0.2689$$
$$\mathcal{L} = (0.2689)^1 \cdot (0.7311)^0 = 0.2689$$

$$ML = 0.98 \cdot 0.881 \cdot 0.2689 = 0.23$$

El CE minimizado es el máximo de ML. El algoritmo de aprendizaje para entrenar un modelo de Regresión Logística Lineal consiste en resolver el problema:

$$\min_{\tilde{\mathbf{w}} \in \mathbb{R}^{d+1}} \{\mathrm{CE}(\tilde{\mathbf{w}})\} = \min_{\tilde{\mathbf{w}} \in \mathbb{R}^{d+1}} \left\{ \sum_{i=1}^{N} \left(-(1 - t_i) \log(1 - \sigma(\tilde{\mathbf{w}}^\intercal \tilde{\mathbf{x}}_i)) - t_i \log(\sigma(\tilde{\mathbf{w}}^\intercal \tilde{\mathbf{x}}_i)) \right) \right\}.$$

Es convexa: no existen mínimos locales. Es diferenciable: los óptimos se caracterizan por los ceros del gradiente.

$$\nabla_{\tilde{\mathbf{w}}} \mathrm{CE}(\tilde{\mathbf{w}}) = \sum_{i=1}^{N} (\sigma(\tilde{\mathbf{w}}^T \tilde{\mathbf{x}}_i) - t_i) \tilde{\mathbf{x}}_i = \mathbf{0}$$

El problema es que no siempre se puede igualar el gradiente a 0.

El Descenso Gradiente es un algoritmo de optimización simple (pero útil) que se utiliza a menudo en Aprendizaje Automático para encontrar el mínimo local.

Gradient Descent: Algorithm

Require: Objective function $\mathcal{F}$, starting point $\boldsymbol{\theta}^{(0)}$ ▷ Start at a random point.
Ensure: $\boldsymbol{\theta}^{(t-1)} \in \mathbb{R}^d$ an approximate local minimum of $\mathcal{F}(\boldsymbol{\theta})$ ▷ Arrive to a valley.
  for $t = 1, 2, \cdots$ **do** ▷ Several steps.
    $\mathbf{g} \leftarrow \nabla_{\boldsymbol{\theta}} \mathcal{F}\left(\boldsymbol{\theta}^{(t-1)}\right)$ ▷ Stop, look for the direction most downhill.
    if $\mathbf{g} \approx \mathbf{0}$ **then**
      return $\boldsymbol{\theta}^{(t-1)}$ ▷ If there is not direction downhill, stop.
    end if
    $\boldsymbol{\theta}^{(t)} \leftarrow \boldsymbol{\theta}^{(t-1)} - \eta^{(t)} \mathbf{g}$ ▷ Take a step in that direction.
  end for

El tamaño del paso $\eta^{(t)}$ debe fijarse en cada iteración t. Se trata de una cuestión crucial: Si el tamaño del paso es demasiado pequeño, el algoritmo necesitará demasiadas épocas (iteraciones) para converger y puede quedar atrapado en mínimos locales más fácilmente. Si el tamaño del paso es grande, la convergencia también será lenta. Si el tamaño del paso es demasiado grande, el descenso de gradiente sobrepasará los mínimos y divergirá. En algunos casos, se pueden calcular tamaños de paso óptimos. Existen heurísticos que garantizan la convergencia, pero sólo a mínimos locales, y normalmente de forma lenta y zigzagueante.

IV.3. Modelos lineales regulados

IV.3.1. Introducción

Ejercicio: dado un modelo tridimensional con los siguientes datos, define un modelo lineal $\{b, w_1, w_2, w_3\}$ con el menor MSE posible. ¿Se puede obtener una predicción perfecta?

$$(1, 0, 1; 2) \rightarrow 1 + 0 + 1 = 2$$

$$(1, 1, 1; 3) \rightarrow 1 + 1 + 1 = 3$$

Es posible obtener una predicción perfecta, y hay varias posibilidades. Primero, se pueden evaluar todos los parámetros por igual con el mismo peso, o un parámetro se puede compensar con los demás.

La complejidad del modelo se debe controlar, ya que no todas las entradas o características van a ser relevantes. Por tanto, un modelo basado en un subconjunto de características parece ser una opción sensible.

IV.3.1.1. Sesgo y varianza

Suponemos un problema de regresión. Las entradas y salidas se relacionan con la ecuación $y = f^*(\mathbf{x}) + \varepsilon$, siendo f^* la función perfecta a la que no tenemos acceso. El modelo debe intentar de aproximar la función verdadera: $f \approx f^*$. La distancia entre ambas funciones se formaliza como **sesgo**. Un sesgo pequeño se puede conseguir con modelos muy flexibles y con muchos parámetros. No obstante, el modelo depende de una muestra particular de entrenamiento, $f_{\mathcal{D}}$. El modelo debe ser estable para distintos datasets $\mathcal{D}$ y $\mathcal{D}'$, de forma que $f_{\mathcal{D}} \approx f_{\mathcal{D}'}$. La estabilidad se define como **varianza**. Una varianza pequeña se consigue con modelos simples con pocos parámetros. Así, se encuentra un **trade-off**:

- **Error debido al sesgo:** diferencia entre la predicción esperada del modelo y el valor correcto a predecir.
- **Error debido a la varianza:** variabilidad de la predicción del modelo para un punto de datos dado.

La regularización suele denotar el conjunto de técnicas que intentan mejorar las estimaciones desviándolas de sus valores basados en la muestra hacia valores que se consideran más "físicamente plausibles". La varianza del modelo se reduce a expensas de un sesgo potencialmente mayor.

IV.3.1.2. Sobreajuste y subajuste

En el sobreajuste, el modelo resultante es excesivamente complejo para describir los datos estudiados. Se puede dar con un número limitado de datos de entrenamiento o una máquina de aprendizaje demasiado compleja (muchos parámetros libres). La varianza será grande y el sesgo pequeño.

En el subajuste, el modelo resultante es demasiado simple para describir los datos estudiados. Puede darse cuando la máquina de aprendizaje es demasiado rígida. El sesgo es grande y varianza pequeña.

La regularización viene buen cuando hay más entradas que patrones, ya que hay que asumir algo del modelo. Esto se debe a varios estimadores óptimos. Desde el punto de vista computacional, evita las inestabilidades numéricas y se evita el sobreajuste. Finalmente, el modelo sencillo mejora la parsimonia e interpretabilidad.

IV.3.1.3. Aprendizaje regulado

El aprendizaje regularizado consiste en modelos entrenados optimizando funciones objetivo de la forma:

$$\mathcal{S} = \mathcal{E}_{\mathcal{D}} + \gamma\mathcal{R}$$

El término principal de la función objetivo es un término de error $\mathcal{E}_{\mathcal{D}}$. Representa lo bien que el modelo se ajusta a los datos de entrenamiento $\mathcal{D}$, por ejemplo el error cuadrático medio (MSE) en regresión y la log verosimilitud negativa en clasificación. El término adicional es un término de regularización $\mathcal{R}$. Penaliza la complejidad del modelo con varios propósitos: evitar el sobreajuste, introducir conocimiento previo, y cumplir ciertas propiedades deseables. γ es un parámetro de regularización, siendo responsable del equilibrio entre precisión y complejidad. Un γ bajo favorece el sobreajuste.

IV.3.2. Funciones reguladas

Existen diferentes funciones de regularización $\mathcal{R}(\theta)$ que asignan a cada conjunto de parámetros θ una medida de su complejidad. Dependiendo de la función elegida, cambiará el efecto sobre θ. La influencia de las funciones de regularización es especialmente clara en los modelos lineales:

- Cada coeficiente de w corresponde a una característica de entrada.
- Si wi = 0, se ignora la característica i-ésima.
- Si wi = wj, la característica i-ésima es similar a la característica j-ésima.

IV.3.2.1. ℓ_2 norm

Término clásico, conocido como regularización de Tikhonov, corresponde a la suma de los cuadrados de las entradas:

$$\mathcal{R}(\mathbf{w}) = ||\mathbf{w}||_2^2 = \sum_{i=1}^{d} w_i^2$$

Controla la complejidad del modelo, siendo diferenciable y fácil de optimizar. Desplaza las entradas hacia 0.

Ejercicio: Dados los siguientes modelos lineales tridimensionales, computa su ℓ_2 norma cuadrada para comprobar cuál es más simple en base a este criterio.

$$\{w_1 = 1, w_2 = 1, w_3 = 1\} \rightarrow 1^2 + 1^2 + 1^2 = 3$$

$$\{w_1 = 3, w_2 = 0, w_3 = 0\} \rightarrow 3^2 + 0^2 + 0^2 = 9$$

$$\{w_1 = 2, w_2 = 2, w_3 = 0\} \rightarrow 2^2 + 2^2 + 0 = 8$$

Así, el primer modelo es el más simple y el segundo el más complejo.

IV.3.2.2. ℓ_1 norm

Corresponde a la suma de los valores absolutos de las entradas:

$$\mathcal{R}(\mathbf{w}) = ||\mathbf{w}||_1 = \sum_{i=1}^{d} |w_i|$$

Controla la complejidad del modelo. El valor absoluto es no-diferenciable alrededor de 0, por lo que este término es más complicado de optimizar. Empuja las entradas hacia cero obligando a que algunas de ellas sean idénticamente cero. Refuerza la dispersión (sparsity).

Ejercicio: Dados los siguientes modelos lineales tridimensionales, computa su ℓ_1 norma para comprobar cuál es más simple en base a este criterio.

$$\{w_1 = 1, w_2 = 1, w_3 = 1\} \rightarrow 1 + 1 + 1 = 3$$

$$\{w_1 = 3, w_2 = 0, w_3 = 0\} \rightarrow 3 + 0 + 0 = 3$$

$$\{w_1 = 2, w_2 = 2, w_3 = 0\} \rightarrow 2 + 2 + 0 = 4$$

Así, el tercer modelo es el más complejo.

IV.3.3. Modelos lineales regulados

El problema de optimización para entrenar un modelo regularizado puede formularse como:

$$\min_{\boldsymbol{\theta}} \left\{ \mathcal{E}_{\mathcal{D}}(\boldsymbol{\theta}) + \gamma \mathcal{R}(\boldsymbol{\theta}) \right\}.$$

Existe una equivalencia entre este modelo sin restricciones y la siguiente formulación con restricciones:

$$\min_{\boldsymbol{\theta}} \left\{ \mathcal{E}_{\mathcal{D}}(\boldsymbol{\theta}) \right\} \text{ s.t. } \mathcal{R}(\boldsymbol{\theta}) \leq c.$$

En el caso de un modelo de regresión lineal:

$$\min_{\mathbf{w} \in \mathbb{R}^d} \left\{ ||\mathbf{y} - \mathbf{X}\mathbf{w}||_2^2 + \gamma \mathcal{R}(\mathbf{w}) \right\} \equiv \min_{\mathbf{w} \in \mathbb{R}^d} \left\{ ||\mathbf{y} - \mathbf{X}\mathbf{w}||_2^2 \right\} \text{ s.t. } \mathcal{R}(\mathbf{w}) \leq c.$$

En caso de un modelo de clasificación lineal:

$$\min_{\mathbf{w} \in \mathbb{R}^d} \left\{ \mathrm{CE}(\mathbf{w}) + \gamma \mathcal{R}(\mathbf{w}) \right\} \equiv \min_{\mathbf{w} \in \mathbb{R}^d} \left\{ \mathrm{CE}(\mathbf{w}) \right\} \text{ s.t. } \mathcal{R}(\mathbf{w}) \leq c.$$

IV.3.3.1. Regresión de Ridge

Este modelo lineal utiliza la regularización de Tikhonov (ℓ_2):

$$\mathcal{R}(\mathbf{w}) = \frac{1}{2}\|\mathbf{w}\|_2^2 = \frac{1}{2}\sum_{i=1}^{d}\mathbf{w}_i^2.$$

Función objetivo:

$$S(\mathbf{w}) = \mathsf{MSE}(\mathbf{w}) + \frac{\gamma}{2}\|\mathbf{w}\|_2^2.$$

Las propiedades clave son:

- **Control de complejidad:** La regularización suaviza los pesos $\mathbf{w}$.
- **Robustez ante ruido:** Para una entrada con ruido $\mathbf{x} + \varepsilon$:

  $$\mathbf{w}^\top(\mathbf{x} + \varepsilon) \approx \mathbf{w}^\top\mathbf{x},$$

  ya que $|\mathbf{w}^\top\varepsilon| \leq \|\mathbf{w}\|_2\|\varepsilon\|_2 \approx 0$.
- **Sin estructura:** Todos los pesos contribuyen al modelo (no hay selección de características).
- **Convexidad:** El problema es diferenciable y convexo.

Con el modelo creado, se debe entrenar y optimizar para hacer los pesos más pequeños posibles.

$$\min_{\mathbf{w}\in\mathbb{R}^d}\left\{\frac{1}{2}\|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2 + \frac{\gamma}{2}\|\mathbf{w}\|_2^2\right\}.$$

Para encontrar la solución de w de optimización se debe derivar e igualar a 0.

$$\begin{aligned}
\nabla_{\mathbf{w}} S(\mathbf{w})|_{\mathbf{w}=\mathbf{w}^*} = 0 &\implies -\mathbf{X}^\intercal(\mathbf{y} - \mathbf{X}\mathbf{w}^*) + \gamma\mathbf{w}^* = 0 \\
&\implies -\mathbf{X}^\intercal\mathbf{y} + \mathbf{X}^\intercal\mathbf{X}\mathbf{w}^* + \gamma\mathbf{w}^* = 0 \\
&\implies (\mathbf{X}^\intercal\mathbf{X} + \gamma\mathbf{I})\mathbf{w}^* = \mathbf{X}^\intercal\mathbf{y} \\
&\implies \boxed{\mathbf{w}^* = (\mathbf{X}^\intercal\mathbf{X} + \gamma\mathbf{I})^{-1}\mathbf{X}^\intercal\mathbf{y}}.
\end{aligned}$$

IV.3.3.2. Modelo de Lasso

Este modelo lineal utiliza como regularizador la norma ℓ_1:

$$\mathcal{R}(\mathbf{w}) = \|\mathbf{w}\|_1 = \sum_{i=1}^{d}|w_i|.$$

Función objetivo:

$$S(\mathbf{w}) = \mathsf{MSE}(\mathbf{w}) + \gamma\|\mathbf{w}\|_1.$$

Las características clave son:

- **Selección automática/implícita de características:** Algunos coeficientes w_i se vuelven exactamente cero, descartando características irrelevantes.
- **Prevención de sobreajuste:** Reduce la complejidad del modelo eliminando variables innecesarias.
- **Propiedades matemáticas:** El problema es convexo pero no diferenciable en $w_i = 0$, por lo que no se puede optimizar fácilmente.

IV.3.3.3. Elastic-Net

Este modelo lineal combina las ventajas de la norma ℓ_1 con las de la norma ℓ_2. Es más estable que Lasso en la selección de características. El regularizador es una combinación de ambos:

$$\mathcal{R}(\mathbf{w}) = \|\mathbf{w}\|_1 + \frac{\gamma_2}{2}\|\mathbf{w}\|_2^2.$$

La función objetivo resulta:

$$S(\mathbf{w}) = \text{MSE}(\mathbf{w}) + \gamma_1\|\mathbf{w}\|_1 + \frac{\gamma_2}{2}\|\mathbf{w}\|_2^2.$$

El problema es convexo pero no diferenciable por tener la norma ℓ_1 de Lasso.

IV.3.4. Resumen

- La regularización suele ser necesaria en problemas reales para controlar la complejidad o inducir estructura.
- Los modelos regularizados se entrenan minimizando tanto un término de error como un término de regularización.
- Existen diferentes opciones para las funciones de regularización, dos de las más importantes son:
  - La norma ℓ_2, que controla la complejidad.
  - La norma ℓ_1, que controla la complejidad e induce la dispersión (sparsity; simplificación del vector eliminando las características que sean 0).
- Los modelos lineales regularizados resultantes son:
  - Regresión Ridge, basada en la norma ℓ_2.
  - Lasso, basado en la norma ℓ_1.
  - Elastic-Net, basado en la combinación de los dos regularizadores.

IV.4. Modelos no lineales y SVMs

IV.4.1. Introducción: limitaciones de modelos lineales

Los modelos lineales se basan en una hipótesis sólida sobre los datos:

- Regresión: existe una relación lineal entre los datos de entrada y los de salida.
- Clasificación: las clases son linealmente separables.

Si dicha relación es real, son una buena opción, pero la flexibilidad de los modelos lineales es muy limitada. El número de grados de libertad corresponde al número de características de entrada d. Son suficientemente complejos si d es grande, o si el número de patrones N es pequeño. En muchas situaciones, su hipótesis subyacente no es cierta y su expresividad no es suficiente.

No siempre es fácil determinar si los modelos lineales son apropiados o no para un conjunto de datos concreto. En un contexto multidimensional, no basta con representar gráficamente el conjunto de datos. Incluso si $N >> d$, puede que exista una relación lineal quizás enmascarada por el ruido. Aunque $d >> N$, tal vez haya mucho ruido y la dimensión efectiva sea pequeña. Siempre es una buena idea empezar con un modelo lineal y comprobar el rendimiento.

IV.4.2. Modelos lineales generalizados

La idea clave es, en lugar de construir el modelo sobre las características originales, ampliar los datos de forma no lineal.

Se utiliza un mapeo no lineal $\phi : \mathbb{R}^d \to \mathbb{R}^D$.

Se construye un modelo lineal usando como patrones $\phi(x_i)$ en lugar de x_i.

Formalmente, el modelo se convierte en:

$$f(x) = w^T \phi(x) = \sum_{i=1}^{D} w_i \phi_i(x),$$

con $w \in \mathbb{R}^D$ y $x \in \mathbb{R}^d$, donde $\phi_i : \mathbb{R}^d \to \mathbb{R}$ es la i-ésima componente del mapeo ϕ.

Ejercicio: dado los siguientes dados de entrada $(x_i = 1, 4)$, computa las características extendidas para el mapeo $\phi(x) = (x, x^2, \sqrt{x})$:

$$x = 1 \to 1, 1^2, \sqrt{1} = (1, 1, 1)$$

$$x = 4 \to 4, 4^2, \sqrt{4} = (4, 16, 2)$$

Computa la salida de un modelo lineal generalizado definido con el mapeo anterior y los pesos $\{w_1 = 1, w_2 = 1, w_3 = 2\}$:

$$w_1 \cdot x + w_2 \cdot x^2 + w_3 \cdot \sqrt{x}$$

$$x = 1 \to 1 \cdot 1 + 1 \cdot 1 + 2 \cdot 1 = 4$$

$$x = 4 \to 1 \cdot 4 + 1 \cdot 16 + 2 \cdot 2 = 24$$

IV.4.2.1. Matriz de datos y optimización

La matriz de datos se convierte en $\Phi \in \mathbb{R}^{N \times D}$:

$$\Phi = \begin{pmatrix} \phi_1(x_1) & \phi_2(x_1) & \cdots & \phi_D(x_1) \\ \phi_1(x_2) & \phi_2(x_2) & \cdots & \phi_D(x_2) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_1(x_N) & \phi_2(x_N) & \cdots & \phi_D(x_N) \end{pmatrix}$$

Por tanto, el resultado del problema de optimización es:

$$\min_{w \in \mathbb{R}^D} \left\{ \frac{1}{2} \|y - \Phi w\|_2^2 \right\},$$

con la solución:

$$w^* = (\Phi^\top \Phi)^{-1} \Phi^\top y$$

El mapeo será crucial para el rendimiento del modelo.

IV.4.2.2. Construcción de características

Las características son cuidadosamente elaboradas por expertos (es decir, se crean características nuevas "a mano"). Si existen conocimientos de expertos, este enfoque puede mejorar el rendiminedo. Así, el modelo no depende (necesariamente) de d o N, sino de la naturaleza del problema. No obstante, se requiere de conocimientos de expertos y una intuición sobre el problema, lo cual es difícil para d grandes.

IV.4.2.3. Conjunto de funciones básicas

Otro enfoque consiste en definir un mapeo suficientemente general para cualquier problema. Existe un conjunto de funciones de base: polinómicas, gaussianas, sigmoidales, de Fourier, wavelets, splines... Entre las ventajas se incluye que es un método automático y no requiere ningún conocimiento experto ni intuición. No obstante, el número de funciones base requeridas crece rápidamente debido a la maldición de la dimensionalidad, puede generar una elevada redundancia, y la dimensión resultante D puede ser mucho mayor de lo necesario.

IV.4.2.4. Otros enfoques de extensión de características

En las **funciones adaptativas básicas**, el mapeo también se aprende y se adapta automáticamente a los datos. Un ejemplo de esto es en las redes neuronales.

Otro enfoque es el truco de Kernel, en el que no es necesario conocer explícitamente ϕ.

IV.4.3. Regresión Kernel Ridge

La regresión Ridge aplicada sobre un espacio de características extendido se formula como:

$$\min_{\mathbf{w}\in\mathbb{R}^D}\left\{\frac{1}{2}\|\mathbf{y}-\mathbf{\Phi w}\|_2^2+\frac{\gamma}{2}\|\mathbf{w}\|_2^2\right\}$$

Un caso particular es el modelo lineal generalizado.

La Regresión Ridge admite una **formulación dual**. Resulta que la solución puede expresarse utilizando únicamente productos escalares entre los vectores.

La solución estándar de Regresión Ridge puede usarse para resolver el problema de optimización:

$$w^* = (\Phi^T\Phi+\gamma I)^{-1}\Phi^T y.$$

Procedimiento:

- Definir el mapeo $\phi:\mathbb{R}^d\to\mathbb{R}^D$.
- Transformar explícitamente la matriz de datos de $X\in\mathbb{R}^{N\times d}$ a $\Phi\in\mathbb{R}^{N\times D}$.
- Resolver el problema estándar de Regresión Ridge invirtiendo una matriz $D\times D$.
- Predecir usando $(w^*)^T\phi(x)$.

Una solución alternativa puede derivarse mediante una formulación restringida del problema y la Dualidad Lagrangiana.

IV.4.3.1. Problema dual: dualidad lagrangiana

La dualidad Lagrangiana puede utilizarse para obtener un problema alternativo en la Regresión Ridge con Kernel. El punto de partida es una formulación restringida del problema original:

$$\min_{\mathbf{w}\in\mathbb{R}^D}\left\{\frac{1}{2}\|\mathbf{y}-\Phi\mathbf{w}\|_2^2+\frac{\gamma}{2}\|\mathbf{w}\|_2^2\right\}\equiv\min_{\mathbf{w}\in\mathbb{R}^D}\left\{\frac{1}{2\gamma}\sum_{i=1}^N e_i^2+\frac{1}{2}\|\mathbf{w}\|_2^2\right\}\text{ s.a. } e_i=y_i-\mathbf{w}^T\phi(\mathbf{x}_i)$$

Mediante la dualidad Lagrangiana, el problema restringido es equivalente al siguiente problema dual:

$$\max_{\alpha\in\mathbb{R}^N}\{D(\alpha)\}=\max_{\alpha\in\mathbb{R}^N}\left\{-\frac{\gamma}{2}\|\alpha\|_2^2-\frac{1}{2}\alpha^T\Phi\Phi^T\alpha+\alpha^T\mathbf{y}\right\}$$

La solución se caracteriza por los ceros del gradiente, y puede relacionarse con la solución primal:

$$\nabla_\alpha D(\alpha)\Big|_{\alpha^*}=\gamma\alpha^*-\Phi\Phi^T\alpha^*+\mathbf{y}=0\implies\begin{cases}\alpha^* &= (\Phi\Phi^T+\gamma I_N)^{-1}\mathbf{y},\\ \mathbf{w}^* &= \sum_{i=1}^N\alpha_i^*\phi(\mathbf{x}_i).\end{cases}$$

La formulación dual conduce a un enfoque alternativo. La formulación dual puede ser más conveniente computacionalmente cuando el número de muestras N es menor que la dimensión del espacio de características D, ya que evita operar directamente en un espacio de alta dimensión.

Procedimiento:

1. Definir el mapeo $\phi : \mathbb{R}^d \to \mathbb{R}^D$.
2. Transformar explícitamente la matriz de datos de $X \in \mathbb{R}^{N \times d}$ a $\Phi \in \mathbb{R}^{N \times D}$.
3. Resolver el problema dual de Kernel Ridge Regression invirtiendo una matriz $N \times N$:
$$\alpha^* = (\Phi\Phi^T + \gamma I_N)^{-1} y$$
4. Reconstruir la solución primal:
$$w^* = \Phi^T \alpha^*$$
5. Realizar predicciones usando:
$$(w^*)^T \phi(x)$$

IV.4.3.2. Truco del núcleo (kernel)

La solución del problema dual es:

$$\boldsymbol{\alpha}^* = \left(\boldsymbol{\Phi}\boldsymbol{\Phi}^\mathsf{T} + \gamma \mathbf{I}_N\right)^{-1} \mathbf{y}$$

Los datos solo aparecen en forma de $\mathbf{K} = \boldsymbol{\Phi}\boldsymbol{\Phi}^\mathsf{T} \in \mathbb{R}^{N \times N}$, donde cada elemento se define como:

$$k_{i,j} = \mathcal{K}(\mathbf{x}_i, \mathbf{x}_j) = \boldsymbol{\phi}(\mathbf{x}_i)^\mathsf{T} \boldsymbol{\phi}(\mathbf{x}_j)$$

La función $\mathcal{K} : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ se conoce como **función kernel**. Esta función calcula el producto interno en un espacio de Hilbert determinado. Nota importante: La función kernel puede definirse directamente, sin necesidad de una forma explícita para $\boldsymbol{\phi}$.

Queríamos obtener α para obtener w y poder calcular el modelo para hacer predicciones. Así, el hiperplano primal puede recuperarse como:

$$\mathbf{w}^* = \boldsymbol{\Phi}^\top \boldsymbol{\alpha}^* = \sum_{i=1}^{N} \alpha_i^* \boldsymbol{\phi}(\mathbf{x}_i)$$

La predicción se realiza mediante:

$$f(\mathbf{x}) = (\mathbf{w}^*)^\top \boldsymbol{\phi}(\mathbf{x}) = \sum_{i=1}^{N} \alpha_i^* \boldsymbol{\phi}(\mathbf{x}_i)^\top \boldsymbol{\phi}(\mathbf{x}) = \sum_{i=1}^{N} \alpha_i^* \mathcal{K}(\mathbf{x}_i, \mathbf{x})$$

No es necesario calcular explícitamente $\mathbf{w}^*$, y no se requiere conocer $\boldsymbol{\phi}$ directamente, basta con conocer la función kernel $\mathcal{K}$. Así, tras haber calculado α, cada vez que tengamos un dato nuevo se puede incorporar directamente con la matriz $\mathbf{K}$.

Procedimiento:

1. **Definir la función kernel**:

$$\mathcal{K} : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$$

2. **Resolver el problema dual**: Invertir una matriz $N \times N$ en la Regresión Ridge dual.

3. **Realizar predicciones**:

$$f(\mathbf{x}) = \sum_{i=1}^{N} \alpha_i^* \mathcal{K}(\mathbf{x}_i, \mathbf{x})$$

El cálculo de $\mathcal{K}$ debe ser **eficiente**, y no debe requerir la aplicación explícita del mapeo ϕ.

IV.4.3.3. Construcción de funciones de Kernel

Una función kernel $\mathcal{K} : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ es una función simétrica y definida positiva.

Dados dos kernels $\mathcal{K}_1(\mathbf{x}, \mathbf{x}')$ y $\mathcal{K}_2(\mathbf{x}, \mathbf{x}')$, y $c \in \mathbb{R}$, se pueden definir nuevos kernels mediante:

- $\mathcal{K}_1(\mathbf{x}, \mathbf{x}') + c$
- $c\mathcal{K}_1(\mathbf{x}, \mathbf{x}')$, para $c > 0$
- $\mathcal{K}_1(\mathbf{x}, \mathbf{x}') + \mathcal{K}_2(\mathbf{x}, \mathbf{x}')$
- $\mathcal{K}_1(\mathbf{x}, \mathbf{x}')\mathcal{K}_2(\mathbf{x}, \mathbf{x}')$

Ejemplos de kernels comunes:

- **Lineal**:

$$\begin{aligned}\mathcal{K}(\mathbf{x}, \mathbf{x}') &= \mathbf{x}^T\mathbf{x}' \\ \mathcal{K}(\mathbf{x}, \mathbf{x}') &= c + \mathbf{x}^T\mathbf{x}' \\ \mathcal{K}(\mathbf{x}, \mathbf{x}') &= (\mathbf{x} - \boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\mathbf{x}' - \boldsymbol{\mu})\end{aligned}$$

- **Polinomial** (grado d): $\mathcal{K}(\mathbf{x}, \mathbf{x}') = (\mathbf{x}^T\mathbf{x}' + c)^d$
- **Gaussiano (RBF)**: $\mathcal{K}(\mathbf{x}, \mathbf{x}') = \exp\left(-\gamma\|\mathbf{x} - \mathbf{x}'\|_2^2\right)$
- **Exponencial**: $\mathcal{K}(\mathbf{x}, \mathbf{x}') = \exp\left(-\gamma\|\mathbf{x} - \mathbf{x}'\|_2\right)$
- Otros kernels: Gamma Exponencial, Sigmoide, Matérn, Kernel Periódico...

La selección del kernel (y sus hiperparámetros) debe realizarse cuidadosamente según el problema específico.

IV.4.4. Clasificadores de vectores de soporte

IV.4.4.1. Múltiples hiperparámetros

Las máquinas de vectores soporte (SVM) surgen en el marco de los problemas de clasificación linealmente separables. Existen múltiples hiperplanos que separan perfectamente los datos, aunque elgunos de ellos generalizarán mejor que otros. En el caso de la regresión logística, un enfoque probabilístico selecciona el mejor hiperplano, pero se pueden utilizar otras interpretaciones geométricas.

IV.4.4.2. Margen de un modelo lineal

La intuición geométrica puede formalizarse con el concepto de **margen**.

El **margen** en un problema de clasificación binaria linealmente separable se define como la distancia entre el hiperplano y el punto de datos más cercano:

$$m = \min_{1 \leq i \leq N} \left\{ \frac{|\mathbf{w}^\intercal \mathbf{x}_i + b|}{\|\mathbf{w}\|_2} \right\}$$

Cuando el problema es linealmente separable y considerando $y_i \in \{-1, 1\}$, el margen puede expresarse alternativamente como:

$$m = \min_{1 \leq i \leq N} \left\{ \frac{y_i(\mathbf{w}^\intercal \mathbf{x}_i + b)}{\|\mathbf{w}\|_2} \right\}$$

- $\mathbf{w}$: Vector normal al hiperplano de separación
- b: Término de sesgo (bias)
- $\|\mathbf{w}\|_2$: Norma euclídea del vector $\mathbf{w}$
- y_i: Etiqueta de clase (-1 o 1)

Ejercicio: dado un modelo de clasificación lineal bidimensional con pesos $\{b = 0, w_1 = 1, w_2 = 0\}$ y el siguiente dataset, computa las distancias entre cada punto y el hiperplano usando $|\mathbf{w^t x}_i + b|/||\mathbf{w}||_2$ y el margen del modelo. Como la norma de **w** es 1, se podría simplificar.

$$-1, -1; -1 \rightarrow (-1 \cdot 1) + (-1 \cdot 0) + 0 = -1 < 0 \checkmark \rightarrow d = |-1|/1 = 1$$
$$-2, 1; -1 \rightarrow (-2 \cdot 1) + (1 \cdot 0) + 0 = -2 < 0 \checkmark \rightarrow d = |-2|/1 = 2$$
$$1, 0; 1 \rightarrow (1 \cdot 1) + (0 \cdot 0) + 0 = 1 > 0 \checkmark \rightarrow d = |1|/1 = 1$$

El modelo sí separa las dos clases. El margen del modelo es 1 (la distancia mínima calculada).

La idea es encontrar un hiperplano que maximice m. El hiperplano definido por $(\mathbf{w}, b)$ es el mismo que el definido por $(c\mathbf{w}, cb)$, para $c > 0$. Debe aplicarse algún tipo de normalización, y hay dos enfoques diferentes: Fijar la norma de w o hacer que los puntos más cercanos pertenezcan a los hiperplanos de apoyo ($\mathbf{w}^T\mathbf{x} + b = \pm 1$).

$$y_i(\mathbf{w}^\intercal \mathbf{x}_i + b) \geq 1$$
$$m = \frac{1}{\|\mathbf{w}\|_2}$$

IV.4.4.3. Clasificador de vectores de soporte de margen duro

El problema de clasificación se define para problemas de clasificación binaria de la siguiente forma:

$$\max_{\mathbf{w}\in\mathbb{R}^d}\left\{\frac{1}{\|\mathbf{w}\|_2}\right\} \equiv \min_{\mathbf{w}\in\mathbb{R}^d}\left\{\frac{1}{2}\|\mathbf{w}\|_2^2\right\}$$

sujeto a:

$$y_i(\mathbf{w}^\top\mathbf{x}_i + b) \geq 1, \quad \text{for } 1 \leq i \leq N$$

o alternativamente:

$$\text{s.t. } \begin{cases} y_i(\mathbf{w}^\top\mathbf{x}_i + b) \geq 1, \\ 1 \leq i \leq N. \end{cases}$$

Para poder hacer esto, el problema debe ser linealmente separable.

La función objetivo es convexa y diferenciable. El problema tiene restricciones lineales. Puede resolverse utilizando la dualidad lagrangiana. Las variables duales se relacionan con las primarias como:

$$\mathbf{w} = \sum_{i=1}^{N} \alpha_i y_i \mathbf{x}_i$$

El problema dual resultante es:

$$\max_{\alpha\in\mathbb{R}^N}\left\{-\frac{1}{2}\alpha^\top\tilde{\mathbf{X}}\tilde{\mathbf{X}}^\top\alpha + \alpha^\top\mathbf{1}\right\}$$

sujeto a:

$$\begin{cases} \alpha^\top\mathbf{y} = 0, \\ \alpha \geq \mathbf{0}, \end{cases}$$

que es equivalente a:

$$\min_{\alpha\in\mathbb{R}^N}\left\{\frac{1}{2}\alpha^\top\tilde{\mathbf{X}}\tilde{\mathbf{X}}^\top\alpha - \alpha^\top\mathbf{1}\right\}$$

sujeto a:

$$\begin{cases} \alpha^\top\mathbf{y} = 0, \\ \alpha \geq \mathbf{0}. \end{cases}$$

Esto es un problema cuadrático restringido para el cual hay varios algoritmos ad hoc que lo resuelven. Los datos solo aparecen como productos internos, y como consecuencia de la dualidad lagrangiana, hay dos casos:

- Si $\alpha_i > 0, y_i(\mathbf{w}^\intercal\mathbf{x}_i + b = 1$, este punto sobre el hiperplano de apoyo es un vector de apoyo.
- Si $\alpha_i = 0, y_i(\mathbf{w}^\intercal\mathbf{x}_i + b \geq 1$, este punto no tiene ningún impacto sobre el modelo.

Este modelo es sparse en cuanto a los patrones de entrenamiento.

IV.4.4.4. Clasificador de vectores de soporte de margen blando

La mayoría de los problemas no son linealmente separables. Incluso si lo son (por ejemplo, porque d es grande), puede que no sea conveniente clasificar perfectamente los datos. Esto puede dar lugar a un sobreajuste. Los clasificadores de vectores de soporte de margen suave (SM-SVC) permiten errores de entrenamiento introduciendo variables slack (variables de holgura). Estas variables cuantifican la violación del margen de cada patrón. Las restricciones se modifican a $y_i(\mathbf{w}^\intercal \mathbf{x}_i + b) \geq 1 - \xi_i$, siendo $\xi_i \geq 0$ la distancia al hiperplano de soporte correspondiente. Las variables slack se penalizan para que sean lo más pequeñas posible.

$$\min_{\substack{\mathbf{w}\in\mathbb{R}^d \\ b\in\mathbb{R} \\ \boldsymbol{\xi}\in\mathbb{R}^N}} \left\{ \frac{1}{2}\|\mathbf{w}\|_2^2 + C\sum_{i=1}^{N} \xi_i \right\}$$

$$\text{s.t.} \begin{cases} y_i(\mathbf{w}^\intercal \mathbf{x}_i + b) \geq 1 - \xi_i, \\ \xi_i \geq 0, \\ 1 \leq i \leq N. \end{cases}$$

Este problema se define para problemas de clasificación binaria. No es necesario que el problema sea linealmente separable. El hiperparámetro C controla el equilibrio entre precisión y complejidad.

De forma equivalente, se puede escribir el problema sin restricciones de la siguiente forma:

$$\min_{\substack{\mathbf{w}\in\mathbb{R}^d \\ b\in\mathbb{R}}} \left\{ \frac{1}{2}\|\mathbf{w}\|_2^2 + C\sum_{i=1}^{N} \left[1 - y_i(\mathbf{w}^\intercal \mathbf{x}_i + b)\right]_+ \right\}$$

donde $|\chi|_+$ denota la parte positiva (función bisagra). La medida resultante se conoce como función de pérdida bisagra.

Ejercicio: dado un modelo de clasificación bidimensional con los pesos $\{b = 0, w_1 = 0.25, w_2 = -0.5\}$ y los siguientes datos, indicar si el modelo separa las dos clases.

$$(-1, -1; -1) \to 0 + (0.25 \cdot -1) + (-0.5 \cdot -1) = 0.25 \to 1 \neq -1$$

$$(-2, 1; -1) \to 0 + (0.25 \cdot -2) + (-0.5 \cdot 1) = -1 \to -1\checkmark$$

$$(1, 0; 1) \to 0 + (0.25 \cdot 1) + (-0.5 \cdot 0) = 0.25 \to 1\checkmark$$

Las dos clases no se separan bien, ya que el primer conjunto de datos no predice la salida correcta.

Computar la función de pérdida de bisagra para cada patrón usando $|1 - y_i(\mathbf{w}^\intercal \mathbf{x}_i + b)|_+$:

$$(-1, -1; -1) \to |1 - (-1)(0.25 \cdot (-1) - 0.5 \cdot (-1) + 0)|_+ = 1 + 1 \cdot 0.25 = 1.25$$

$$(-2, 1; -1) \to |1 - (-1)(0.25 \cdot (-2) - 0.5 \cdot 1 + 0)|_+ = 1 + 1 \cdot -1 = 0$$

$$(1, 0; 1) \to |1 - 1(0.25 \cdot 1 - 0.5 \cdot 0 + 0)|_+ = 1 - 1 \cdot 0.25 = 0.75$$

Para la optimización de los clasificadores de vectores soporte con margen blando se calcula

$$\begin{aligned} \min_{\substack{\mathbf{w}\in\mathbb{R}^d \\ b\in\mathbb{R} \\ \boldsymbol{\xi}\in\mathbb{R}^N}} & \left\{ \frac{1}{2}\|\mathbf{w}\|_2^2 + C\sum_{i=1}^{N}\xi_i \right\} \\ \text{s.t.} \quad & y_i(\mathbf{w}^\mathsf{T}\mathbf{x}_i + b) \geq 1 - \xi_i, \\ & \xi_i \geq 0, \\ & 1 \leq i \leq N. \end{aligned}$$

El problema resultante dual es:

$$\begin{aligned} \min_{\boldsymbol{\alpha}\in\mathbb{R}^N} & \left\{ \frac{1}{2}\boldsymbol{\alpha}^\mathsf{T}\widetilde{\mathbf{X}}\widetilde{\mathbf{X}}^\mathsf{T}\boldsymbol{\alpha} - \boldsymbol{\alpha}^\mathsf{T}\mathbf{1} \right\} \\ \text{s.t.} & \begin{cases} \boldsymbol{\alpha}^\mathsf{T}\mathbf{y} = 0, \\ 0 \leq \alpha_i \leq C \quad \forall i = 1, \ldots, N. \end{cases} \end{aligned}$$

Se trata de nuevo de un problema cuadrático restringido. Los coeficientes duales tienen una cota superior adicional C. Si C es mayor que un valor determinado, se recupera el SVC de margen duro. Existen diferentes algoritmos ad hoc para resolverlo. Los datos sólo aparecen en forma de productos internos. Como consecuencia de la dualidad Lagrangiana, se cumplen las condiciones:

$$\alpha_i\big(1 - \xi_i - y_i(\mathbf{w}^\top\mathbf{x}_i + b)\big) = 0 \quad \text{y} \quad \beta_i\xi_i = 0, \quad \text{para } i = 1, \ldots, N$$

- Si $\alpha_i = 0$, entonces $y_i(\mathbf{w}^\top\mathbf{x}_i + b) \geq 1$ y el punto no influye en el modelo.
- Si $0 < \alpha_i < C$, entonces $y_i(\mathbf{w}^\top\mathbf{x}_i + b) = 1$ (vectores de soporte).
- Si $\alpha_i = C$, entonces $y_i(\mathbf{w}^\top\mathbf{x}_i + b) = 1 - \xi_i \leq 1$ (vectores con error).
- El modelo es **disperso** en términos de los patrones de entrenamiento (solo los vectores soporte contribuyen).

IV.4.5. Regresión por vectores de soporte

Los modelos de clasificación de vectores de soporte (SVC) tienen ciertas propiedades deseables:

- Pueden entrenarse mediante un problema dual.
- Son dispersos en términos de patrones de entrenamiento.
- Controlan de forma natural la complejidad.

Estas propiedades motivan su extensión a un entorno de regresión. El origen de estas buenas propiedades provienen de maximizar el margen (minimizar la complejidad del modelo) y tener un término de error disperso. Para extender esto a un marco de regresión, se hace parcialmente en (Kernel) Ridge Regression y se necesita una nueva función de pérdida.

IV.4.5.1. Pérdida insensible de ε

La pérdida insensible de ε de un modelo lineal sobre un patrón se define como:

$$|\mathbf{w}^\intercal \mathbf{x} + b - y|_\varepsilon = \max\{0, |\mathbf{w}^\intercal \mathbf{x} + b - y| - \varepsilon\}.$$

Los errores más pequeños que ε se ignoran, y los más grandes se penalizan linealmente. Esto evita el sobreajuste ignorando los errores pequeños, pero el hiperparámetro ε se debe ajustar.

Ejercicio: dado un modelo de regresión lineal bidimensional con los pesos $\{b = 0, w_1 = 1, w_2 = 1\}$ y los siguientes datos, computa la predicción para cada patrón.

$$(-1, -1; -1.9) \to 0 + 1 \cdot (-1) + 1 \cdot (-1) = -2$$

$$(-2, 1; -1) \to 0 + 1 \cdot (-2) + 1 \cdot 1 = -1 \checkmark$$

$$(1, 0; 2) \to 0 + 1 \cdot 1 + 1 \cdot 0 = 1$$

Ahora se computa la pérdida insensible de ε para cada patrón utilizando $\max\{0, |\mathbf{w}^\intercal \mathbf{x} + b - y| - \varepsilon\}$ con $\varepsilon = 0.25$.

$$(-1, -1; -1.9) \to -2 - (-1.9) = 0.1 < 0.25 \to \varepsilon_i = 0$$

$$(-2, 1; -1) \to -1 - (-1) = 0 < 0.25 \to \varepsilon_i = 0$$

$$(1, 0; 2) \to 1 - 2 = 1 > 0.25 \to \varepsilon_i = 1 - 0.25 = 0.75$$

Hay un 0.25 del error que se ignora. Si es más pequeño, se ignora, si el error es mayor, se resta 0.25 a lo que se ha equivocado el modelo

El Regresor de Vectores Soporte (SVR) se define como la solución del problema:

$$\min_{\substack{\mathbf{w} \in \mathbb{R}^d \\ b \in \mathbb{R}}} \left\{ \frac{1}{2} \|\mathbf{w}\|_2^2 + C \sum_{i=1}^{N} |\mathbf{w}^\intercal \mathbf{x}_i + b - y_i|_\varepsilon \right\}$$

Que equivale a la formulación con variables de holgura:

$$\min_{\substack{\mathbf{w} \in \mathbb{R}^d \\ b \in \mathbb{R} \\ \xi, \xi^* \in \mathbb{R}^N}} \left\{ \frac{1}{2} \|\mathbf{w}\|_2^2 + C \sum_{i=1}^{N} (\xi_i + \xi_i^*) \right\}$$

$$\text{s.a.} \quad \begin{cases} \mathbf{w}^\intercal \mathbf{x}_i + b - y_i \leq \varepsilon + \xi_i, \\ y_i - \mathbf{w}^\intercal \mathbf{x}_i - b \leq \varepsilon + \xi_i^*, \\ \xi_i, \xi_i^* \geq 0, \\ 1 \leq i \leq N. \end{cases}$$

donde:

- $\mathbf{w}$ es el vector de pesos

- b es el término de sesgo
- ξ_i, ξ_i^* son variables de holgura
- C es el parámetro de penalización
- ε define la zona insensible

El problema dual resultante es:

$$\min_{\boldsymbol{\alpha},\boldsymbol{\alpha}^* \in \mathbb{R}^N} \left\{ \frac{1}{2}(\boldsymbol{\alpha}^* - \boldsymbol{\alpha})^\top \mathbf{X}\mathbf{X}^\top (\boldsymbol{\alpha}^* - \boldsymbol{\alpha}) + \varepsilon(\boldsymbol{\alpha}^* + \boldsymbol{\alpha})^\top \mathbf{1} - (\boldsymbol{\alpha}^* - \boldsymbol{\alpha})^\top \mathbf{y} \right\}$$

sujeto a:

$$\begin{cases} (\boldsymbol{\alpha}^* - \boldsymbol{\alpha})^\top \mathbf{1} = 0, \\ 0 \leq \alpha_i, \alpha_i^* \leq C \quad \forall i = 1, \ldots, N. \end{cases}$$

El hiperplano primal se recupera como:

$$\mathbf{w} = \sum_{i=1}^{N} (\alpha_i - \alpha_i^*) \mathbf{x}_i.$$

Propiedades Clave:

- Es un **problema cuadrático con restricciones**.
- Existen diferentes algoritmos especializados para resolverlo.
- Los datos solo aparecen en forma de **productos internos**.

Como consecuencia de la dualidad Lagrangiana:

- Si $\alpha_i - \alpha_i^* = 0$, el punto está dentro del tubo ε-insensible y no afecta al modelo.
- En caso contrario, el punto está fuera del tubo (o en el borde) y es un **vector soporte**.

IV.4.6. Máquinas de vectores soporte no lineales (SVMs)

Los modelos lineales no son suficientes en muchos problemas. En los problemas de optimización para entrenar SVM, los datos sólo aparecen como productos internos. Además, la predicción para un nuevo punto de datos, $\mathbf{w}^\intercal \mathbf{x} + b$, también se puede calcular utilizando sólo productos internos. Las SVM pueden extenderse a un marco no lineal utilizando un mapeo $\phi : \mathbb{R}^d \to \mathbb{R}^D$. Gracias al truco del kernel, en lugar de definir explícitamente ϕ, se utiliza una función kernel $\mathcal{K} : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$. La selección del kernel, y sus hiperparámetros, es crucial. Una de las opciones más comunes es el kernel RBF $\mathcal{K}(\mathbf{x}, \mathbf{x}') = \exp(-\gamma ||\mathbf{x} - \mathbf{x}'||_2^2)$. Los patrones $\mathbf{x}_i$ se sustituyen por $\phi(\mathbf{x}_i)$. La matriz $\mathbf{X}\mathbf{X}^\intercal$ se sustituye por la matriz kernel $\mathbf{K} = \Phi\Phi^\intercal$.