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

6 de febrero de 2025 15:37

Universidad Autónoma de Madrid
Bioinformática y Biología Computacional

Código en Github

Índice general

Capítulo I

Introducción

I.1. Repaso: Álgebra lineal

I.1.1. Notación general

Se denota un **vector** $\mathbf{x}$ como $x \in \mathbb{R}^n$ con n entradas, donde $x_i \in \mathbb{R}$ es la entrada i-ésima. Un vector se puede ver como una matriz de dimensiones $n \times 1$ y se denomina también como vector-columna.

$$\mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \\ ... \\ x_n \end{pmatrix} \in \mathbb{R}^n$$

Se denota una **matriz** $\mathbf{A}$ como $A \in \mathbb{R}^{m \times n}$ con n columnas y m filas, donde $A_{i,j} \in \mathbb{R}$ es la entrada en la fila i-ésima y columna j-ésima.

$$\mathbf{A} = \begin{pmatrix} A_{1,1} & ... & A_{1,n} \\ ... & & ... \\ A_{m,1} & ... & A_{m,n} \end{pmatrix} \in \mathbb{R}^{m \times n}$$

Una **matriz de identidad** $\mathbf{I} \in \mathbb{R}^{n \times n}$ es una matriz cuadrada con 1 en la diagonal principal y 0 en el resto. Para cualquier matriz $\mathbf{A} \in \mathbb{R}^{n \times n}$, se cumple que $\mathbf{A} \times \mathbf{I} = \mathbf{I} \times \mathbf{A} = \mathbf{A}$.

$$\mathbf{I} = \begin{pmatrix} 1 & 0 & ... & 0 \\ 0 & 1 & ... & ... \\ ... & ... & 1 & 0 \\ 0 & ... & 0 & 1 \end{pmatrix}$$

I.1.2. Operaciones de matrices

Multiplicación vector-vector Hay dos tipos de productos vector-vector:

- **Producto interno (inner product):** Dados dos vectores $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$ de la misma dimensión, el producto interno es un escalar (un sólo número). Se usa en cálculos que involucran proyecciones, determinación de ortogonalidad, etc. El producto interno puede aplicarse en cualquier dimensión.

$$\mathbf{x}^T y = \sum_{i=1}^{n} x_i y_i \in \mathbb{R}$$

- **Producto externo (outer product):** Dados dos vectores, no necesariamente de la misma dimensión, $\mathbf{x} \in \mathbb{R}^m, \mathbf{y} \in \mathbb{R}^n$, el producto externo es una matriz de $m \times n$.

$$xy^T = \begin{pmatrix} x_1 y_1 & \dots & x_1 y_n \\ \dots & & \dots \\ x_m y_1 & \dots & x_m y_n \end{pmatrix} \in \mathbb{R}^{m \times n}$$

  El producto externo tiene las siguientes aplicaciones en bioinformática:

  - **Álgebra lineal**
    - **Matrices de Covarianza**: Se utiliza en la construcción de matrices de covarianza, que son fundamentales en estadística y análisis de datos.
    - **Representación de Transformaciones**: Ayuda a representar transformaciones lineales y rotaciones en el espacio.
  - **Análisis de Datos y Machine Learning**
    - **Modelos de Regresión**: En algunos métodos de regresión, como la regresión de mínimos cuadrados, se utiliza el producto externo para construir matrices de diseño.
    - **Métodos de Factorización**: Se aplica en técnicas como la factorización de matrices y la descomposición en valores singulares (SVD), que son esenciales en la reducción de dimensionalidad y análisis de componentes principales (PCA).

Multiplicación matriz-vector Dada una matriz $\mathbf{A} \in \mathbb{R}^{m \times n}$ y un vector $\mathbf{x} \in \mathbb{R}^n$, el producto es un vector del tamaño $\mathbb{R}^m$. El proceso consiste en multiplicar cada fila de la matriz $\mathbf{A}$ por el vector $\mathbf{x}$ y sumar los resultados.

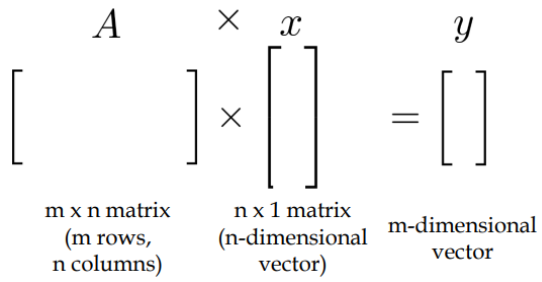
$$Ax = \begin{pmatrix} A_{1,1}x_1 + A_{1,2}x_2 + \dots + A_{1,n}x_n \\ \dots \\ A_{m,1}x_1 + A_{m,2}x_2 + \dots + A_{m,n}x_n \end{pmatrix} \in \mathbb{R}^m$$

Entre las aplicaciones de la multiplicación matriz-vector se encuentran:

- **Álgebra lineal y matemáticas puras**
  - **Sistemas de Ecuaciones Lineales**: Resolver sistemas de ecuaciones de la forma

$$Ax = b \tag{I.1}$$

.



  - **Transformaciones Lineales**: Representar y aplicar transformaciones lineales como rotaciones, escalamientos y reflexiones.

- **Computación y Algoritmos**
  - **Algoritmos de Optimización**: Implementar métodos de optimización como gradiente descendente.
  - **Análisis de Gráficos**: Procesar datos en grafos y redes, como algoritmos de PageRank.
  - **Compresión de Datos**: Utilizar en algoritmos de compresión de datos y análisis de componentes principales (PCA).
- **Machine Learning e Inteligencia Artificial**
  - **Regresión Lineal**: Resolver problemas de regresión lineal para ajustar modelos a datos.
  - **Redes Neuronales**: Calcular activaciones y actualizar pesos en redes neuronales.
- **Bioinformática**
  - **Análisis de Datos Genómicos**: Procesar y analizar grandes volúmenes de datos genómicos y de secuenciación.

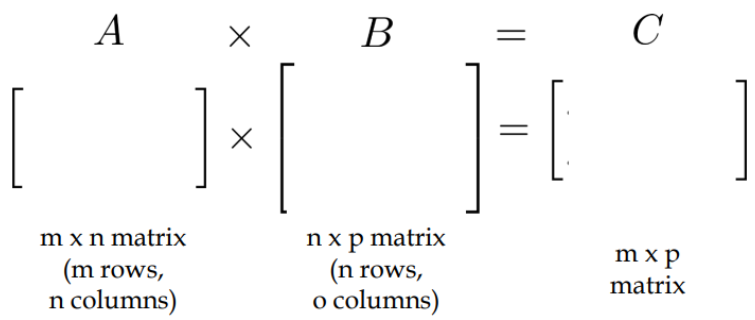
Multiplicación matriz-matriz Dadas dos matrices $\mathbf{A} \in \mathbb{R}^{m\times n}$ y $\mathbf{B} \in \mathbb{R}^{n\times p}$, el producto es una matriz de tamaño $\mathbb{R}^{m\times p}$.

$$\mathbf{AB} = \begin{pmatrix} \sum_{k=1}^{n} A_{1,k}B_{k,1} & \dots & \sum_{k=1}^{n} A_{1,k}B_{k,p} \\ \dots & \dots & \dots \\ \sum_{k=1}^{n} A_{m,1}B_{k,1} & \dots & \sum_{k=1}^{n} A_{m,k}B_{k,p} \end{pmatrix} \in \mathbb{R}^{m\times p}$$

I.1.3. Propiedades de la multiplicación de matrices

No conmutatividad: En general, la multiplicación de matrices no es conmutativa, es decir, $\mathbf{A} \times \mathbf{B} \neq \mathbf{B} \times \mathbf{A}$. Por ejemplo:

$$\begin{bmatrix} 1 & 1 \\ 0 & 0 \end{bmatrix} \times \begin{bmatrix} 0 & 0 \\ 2 & 0 \end{bmatrix} = \begin{bmatrix} 2 & 0 \\ 0 & 0 \end{bmatrix}$$



$$\begin{bmatrix} 0 & 0 \\ 2 & 0 \end{bmatrix} \times \begin{bmatrix} 1 & 1 \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 2 & 2 \end{bmatrix}$$

En el caso de multiplicar una matriz con una matriz de identidad, sí es conmutativa.

Matriz inversa Si $\mathbf{A}$ es una matriz cuadrada $m \times m$, y tiene inversa, entonces

$$AA^{-1} = A^{-1}A = I$$

Transposición de matriz Dada una matriz $A \in \mathbb{R}^{m \times n}$, su transpuesta A^T: es una matriz $n \times m$ donde $\forall i, j, \quad (A^T)_{ij} = A_{ji}$. Por ejemplo:

$$A = \begin{bmatrix} 1 & 2 & 0 \\ 3 & 5 & 9 \end{bmatrix} \qquad A^T = \begin{bmatrix} 1 & 3 \\ 2 & 5 \\ 0 & 9 \end{bmatrix}$$

I.2. Introducción al Aprendizaje Automático o Machine Learning

I.2.1. 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 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.2.2. 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.2.3. 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.

[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.

I.2.4. 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.

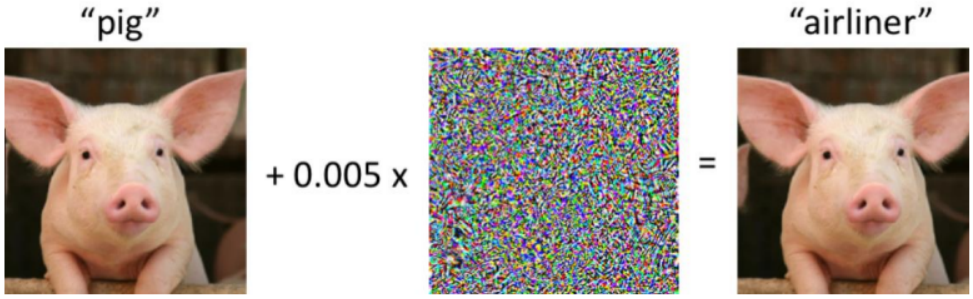


Figura I.1: *Reconocimiento de imágenes con ataque adversario.*

I.2.5. 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 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.2.6. 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.2.7. 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.

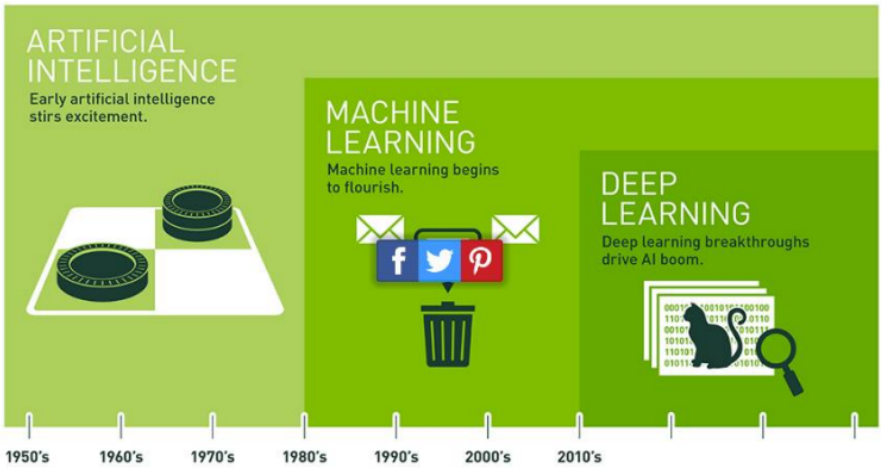


Figura I.2: *Mapa temporal del desarrollo de las inteligencias artificiales. Ya está algo desfasado, faltaría añadir después del Deep Learning los Modelos Generativos.*

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.2.8. 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.2.8.1. Supervisado vs no supervisado

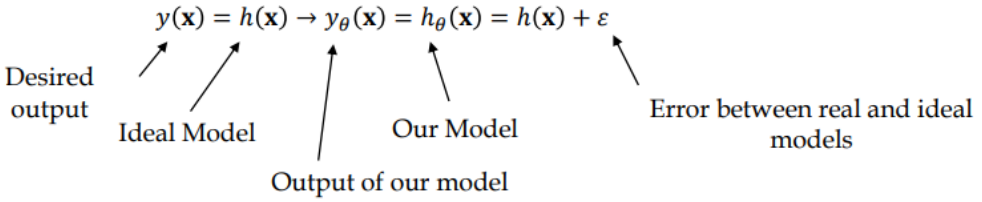
Supongamos una función desconocida $y_\Theta(\mathbf{x}) = h_\Theta(\mathbf{x})$, donde x es un ejemplo de entrada y y la salida deseada. En el **aprendizaje supervisado**, se proporciona un conjunto de pares de entrenamiento (x,y), donde x es la entrada y y es la salida deseada. El objetivo es aprender 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.2.8.2. 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).



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

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 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.2.9. Tipos de problemas o tareas

I.2.9.1. 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.2.9.2. 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.3. 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.

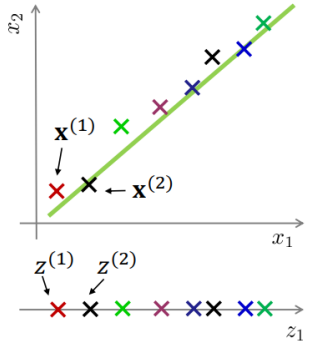
I.3.1. Proyección de datos en dimensiones inferiores

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.



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

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.3.2. PCA: Análisis de Componentes Principales

En el caso de una tabla de expresión génica donde las filas representan pacientes y las columnas corresponden a genes individuales, cada paciente puede describirse como un punto en un espacio de 2000 dimensiones: $\mathbf{x}^{(i)} \in \mathbb{B}^{2000}$, es decir, un vector de 2000 componentes binarios.

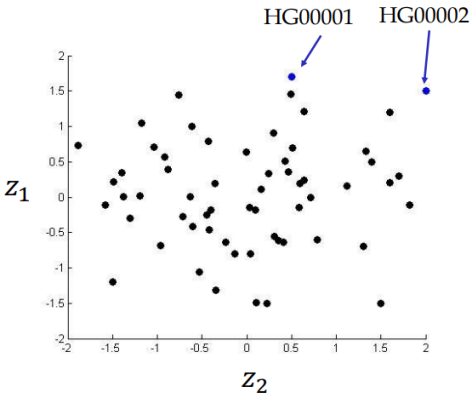
El Análisis de Componentes Principales (PCA) permite reducir estas 2000 dimensiones a un espacio de menor dimensión, por ejemplo, a dos dimensiones: $\mathbf{z}^{(i)} \in \mathbb{R}^2$

Sin embargo, al proyectar los datos, el tipo de variable puede cambiar. En este caso, los valores transformados pierden su significado biológico directo (es decir, ya no tienen una interpretación genética específica), pero la proximidad entre puntos en el

	x_1	x_2	x_3	x_4	x_5	x_6	x_{2000}
Sub_ID	rs307377	rs7366653	rs41307846	rs3753242	rs35082957	rs34154371	...
HG00001	1	0	1	1	0	0	...
HG00002	0	0	1	1	1	0	...
HG00003	1	1	0	0	0	1	...
HG00004	0	0	0	1	0	1	...
HG00005	0	0	1	1	1	1	...
...							

$\mathbf{z}^{(i)} \in \Re^2$

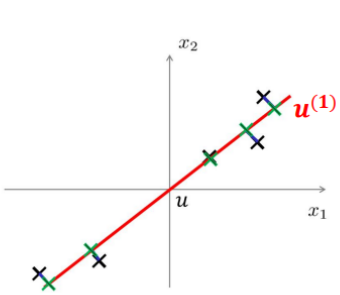
Sub_ID	z_1	z_2
HG00001	0.65	0.71
HG00002	0.43	2.43
HG00003	0.03	1.14
HG00004	5.40	2.11
HG00005	2.33	0.46
...		



espacio bidimensional puede interpretarse como una medida de similitud genética entre sujetos.

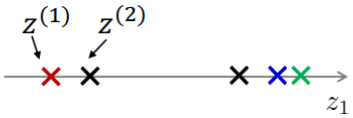
El algoritmo de PCA busca minimizar el error de proyección de los datos sobre un subespacio de menor dimensión. Para la reducción de dos dimensiones a una, esto equivale a encontrar un vector $u^{(1)} \in \mathbb{R}^n$ que minimice la distancia entre los puntos originales y sus proyecciones. Más generalmente, para reducir un espacio de n dimensiones a k dimensiones, PCA busca determinar k vectores ortogonales $u^{(k)}$ que definan el subespacio óptimo para proyectar los datos, minimizando la pérdida de información:

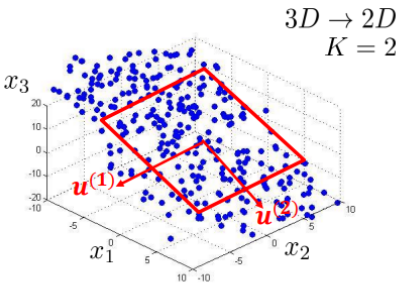
$$\mathbf{x} \in \mathbb{R}^n \rightarrow \mathbf{z} \in \mathbb{R}^k$$



Reduce data from 2D to 1D

$\mathbf{x}^{(i)} \in \Re^2 \rightarrow z^{(i)} \in \Re$





Reduce data from 3D to 2D

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

$$\mathbf{z} = \begin{bmatrix} z_1 \\ z_2 \end{bmatrix}$$

I.3.2.1. Algoritmo de PCA