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

24 de febrero de 2025 19:15

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

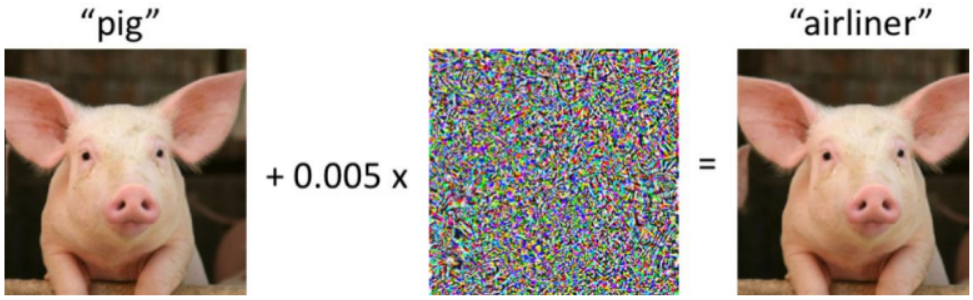


Figura I.1: *Reconocimiento de imágenes con ataque adversario.*

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.

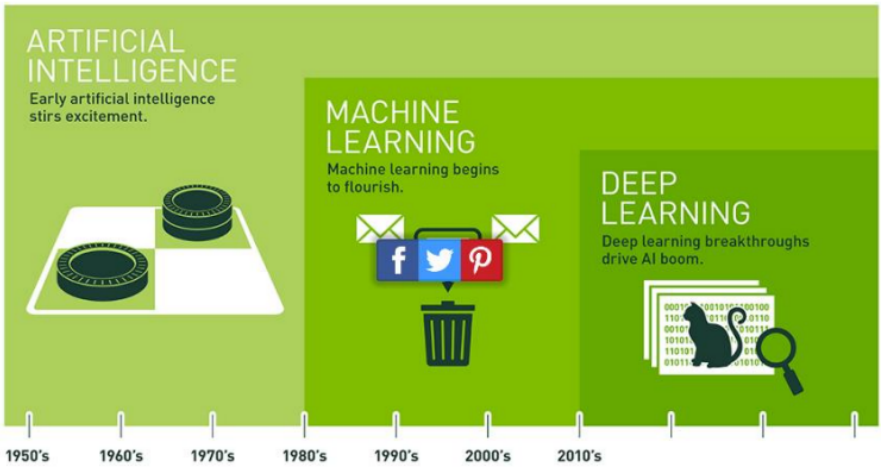


Figura I.2: *Mapa temporal del desarrollo de las inteligencias artificiales. Ya está algo desfasado, faltaría añadir después del Deep Learning los Modelos Generativos.*

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.

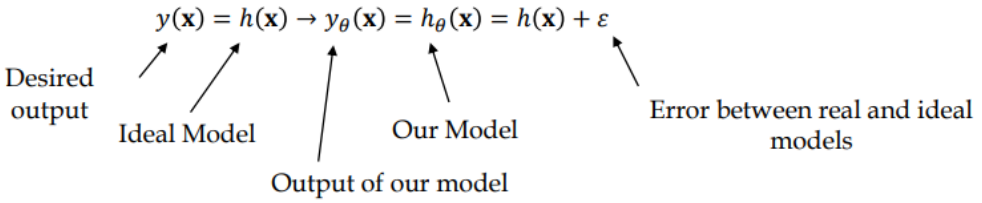
I.1.8.2. Fases de un algoritmo de aprendizaje

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.

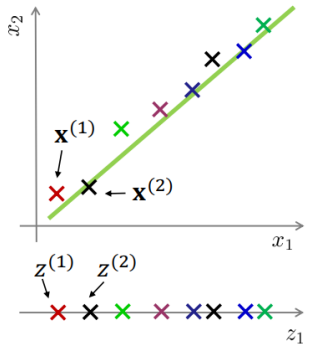
I.2.1. Proyección de datos en dimensiones inferiores

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.



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

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.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.

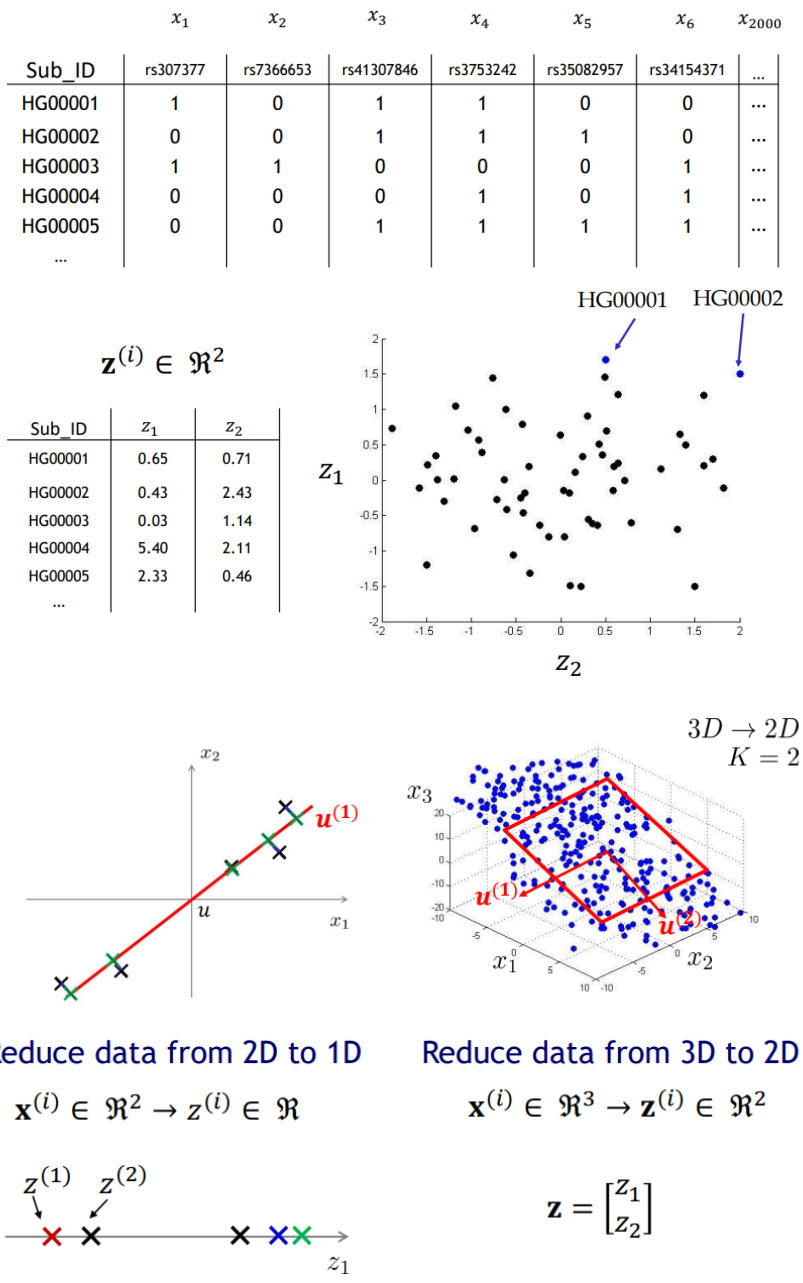
PCA busca minimizar el error de proyección de los datos sobre un subespacio de menor dimensión. Para reducir un espacio de n dimensiones a k dimensiones, PCA determina k vectores ortogonales $\mathbf{u}^{(k)}$ que definen el subespacio óptimo para proyectar los datos:

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

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:

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

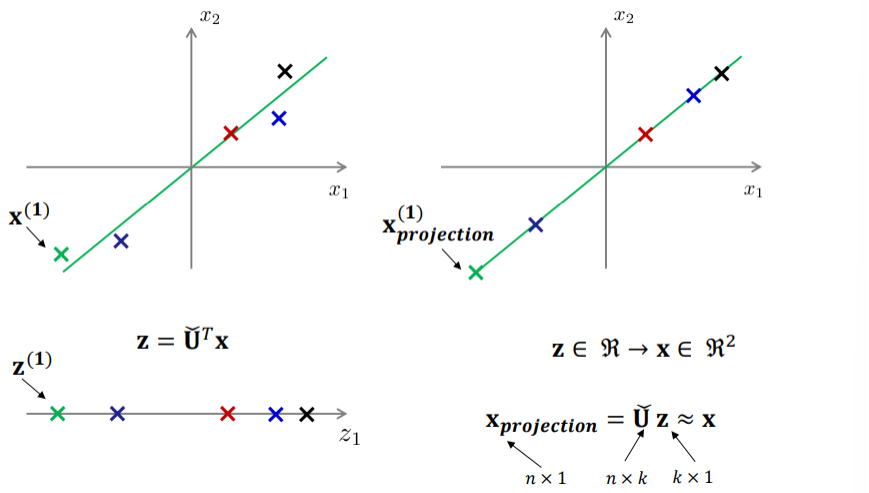
4. **Ordenación y selección de componentes**

   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:

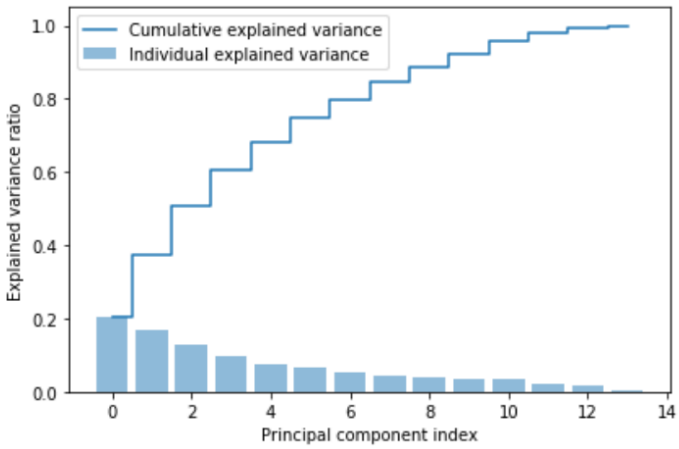
$$i = \frac{\lambda_i}{\sum_{i=1}^{n} \lambda_i}$$

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

}

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

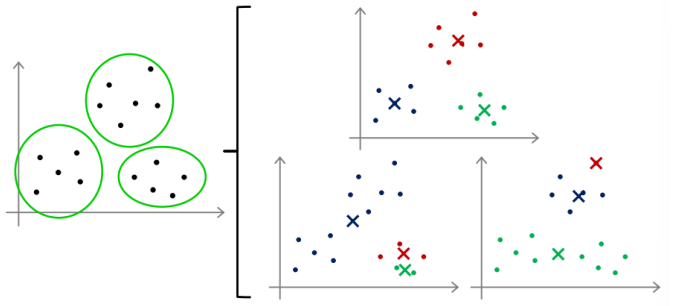
- **Inicialización basada en datos**: Seleccionar K datos aleatorios como centroides iniciales.
- **Repetición del algoritmo**: Ejecutar K-means varias veces y seleccionar la solución con la menor inercia (suma de las distancias al cuadrado entre cada punto y su centroide).

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

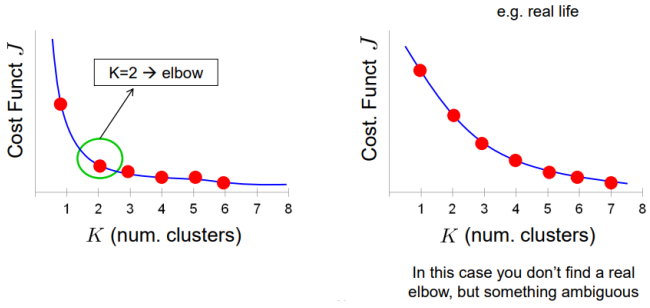
$$\min_{\mu_1,\dots,\mu_K} J(c^{(1)}, \dots, c^{(m)}, \mu_1, \dots, \mu_K)$$

}

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.

Hay un espacio dividido por dos características. No se busca la recta que separa los dos espacios de características, pero es importarte al ser el punto en el que los datos tienen una u otra etiqueta. Otro problema típico de aprendizaje supervisado es la regresión, como puede ser por ejemplo la predicción del tamaño del tumor. Los datos ya están etiquetados, pero se busca para semanas no etiquetadas. Aun así, para una misma semana, puede haber distintas tasas de crecimiento. En este caso sí se busca encontrar la recta. Puede haber varias soluciones independientes.

La clasificación tiene una salida discreta, mientras que en la regresión la salida es continua. El número máximo de clases a partir de que un problema pasa de clasificación a regresión depende de la naturaleza del problema. Si existe una relación entre muestras consecutivas, se espera que una distancia entre muestras esté correlacionada o haya consecuencialidad.

El vector de x es D-dimensional. En regresión, y es una variable continua y real. En clasificación, y existe en un conjunto finito de posibilidades, pudiendo haber problemas multiclase o multietiqueta, pero no hay nada entre las distintas salidas, y éstas son independientes entre sí. Pueden tener o no un orden.

X e Y son variables aleatorias con una distribución conjunta. Hay una probabilidad o patrón que permita predecir Y a partir de X. El predictor es h, el cual permite pasar de x a Y. La función de pérdida L va a ser el que mida la distancia entre la salida del modelo y la salida deseada. Es una función que mide la distancia entre h e Y. Idealmente, queremos que L sea lo menor posible. Las pérdidas esperadas incorporan a la función de pérdida la esperanza, es decir, estimarlo para toda la población. X son todos los posibles valores del problema. Los datos de entrenamiento van a ser una subregión o ejemplo finito de esa región. Entrenamos en base a lo que se conoce, esperando que al enfrentarnos a datos no vistos, el rendimiento sea bueno. Por ello se habla de estimación, ya que no queremos replicar el entrenamiento. Debe funcionar con los datos de entrenamiento, pero funcionar bien con el set de entrenamiento no implica que funcione bien con datos nuevos. La función de pérdida, en la mayoría de casos, no es muy compleja. Tiene un indicador en el que se ve si h(x) es igual o diferente que y, añadiendo 1 cuando no haya igualdad, sumando así los errores. Un sistema que acierta mucho tendrá un indicador I cercano a 0, mientras que un sistema que falle mucho tendrá un I alto. El error es la esperanza o estimador de I. Esto se puede hacer en un problema de clasificación, pero es más complicado en uno de regresión; ahí no se suma 0 o 1, si no que se tiene en cuenta la distancia entre el valor real y la predicción. Se tiene en cuenta el MSE (error cuadrático medio) y MAE (error absoluto medio). Al elevar al cuadrado el error de cada muestra, se pierde el sentido físico biológico (por ejemplo, tasa de crecimiento), pero exageran o magnifican las desviaciones.

Se hacen algoritmos de aprendizaje L que a partir de unos datos se aprende una distribución h. h no solo depende de x, si no de los parámetros Θ de aprendizaje. Hay un conjunto de datos de entrenamiento con etiquetas y la pérdida esperada, que es un estimador para las pérdidas de muchos datos. Introducimos Θ, y dependiendo de Θ tenemos unas pérdidas u otras. Se diseña un algoritmo variando Θ y se busca disminuir

L. La penalización de complejidad pretende limitar la introducción de parámetros, cuyo número es infinito, para limitar la búsqueda de parámetros y permitir que el algoritmo sea generalizable a datos nuevos. El entrenamiento resume todo lo anterior. Busca las pérdidas entre h e y mediante la siguiente fórmula matemática.

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

En general, la pérdida de entrenamiento será menor a la pérdida general. Esto se debe a que se optimiza Theta con respecto a L de entrenamiento, por lo que será mejor para los datos utilizados que para datos nuevos. Esto significa que, de por sí, es un proceso sesgado.

Lo que se busca entonces es tener un subconjunto para poder evaluar el rendimiento del modelo en ese subconjunto. A esto se le conoce como datos de test. Toda la ecuación es igual, pero sin tocar Theta. Se debe optimizar Theta con los datos de entrenamiento, y con los datos de prueba solo se evalúa.