Clase 7: Otros Métodos Supervisados

Matías Leoni

Aprendizaje Automático I
Maestría en IA - Universidad de San Andrés

29 de Julio de 2025

Agenda de la Clase

- **Introducción:** Un vistazo a modelos no paramétricos y probabilísticos.

Agenda de la Clase

- **Introducción:** Un vistazo a modelos no paramétricos y probabilísticos.
- **Bloque 1:**
  - k-Nearest Neighbors (k-NN): Intuición y mecanismo.
  - Naive Bayes: Fundamentos en el Teorema de Bayes.

Agenda de la Clase

- **Introducción:** Un vistazo a modelos no paramétricos y probabilísticos.
- **Bloque 1:**
  - k-Nearest Neighbors (k-NN): Intuición y mecanismo.
  - Naive Bayes: Fundamentos en el Teorema de Bayes.
- **Pausa (15 minutos)**
- **Bloque 2:**
  - Redes Neuronales Simples: El Perceptrón Multicapa (MLP).
  - Comparativa de los métodos.

Agenda de la Clase

- **Introducción:** Un vistazo a modelos no paramétricos y probabilísticos.
- **Bloque 1:**
  - k-Nearest Neighbors (k-NN): Intuición y mecanismo.
  - Naive Bayes: Fundamentos en el Teorema de Bayes.
- **Pausa (15 minutos)**
- **Bloque 2:**
  - Redes Neuronales Simples: El Perceptrón Multicapa (MLP).
  - Comparativa de los métodos.
- **Taller Práctico:** Aplicación en Jupyter Notebook.

Contexto: ¿Por qué otros métodos?

- Hasta ahora hemos visto modelos que asumen una forma funcional específica (lineales, SVM) o que parten el espacio con reglas jerárquicas (árboles).

Contexto: ¿Por qué otros métodos?

- Hasta ahora hemos visto modelos que asumen una forma funcional específica (lineales, SVM) o que parten el espacio con reglas jerárquicas (árboles).
- Hoy exploraremos enfoques con supuestos diferentes:

Contexto: ¿Por qué otros métodos?

- Hasta ahora hemos visto modelos que asumen una forma funcional específica (lineales, SVM) o que parten el espacio con reglas jerárquicas (árboles).
- Hoy exploraremos enfoques con supuestos diferentes:
  - **Basados en instancia (k-NN):** Usan la "memoria" del dataset completo para predecir. No aprenden un "modelo" explícito.

Contexto: ¿Por qué otros métodos?

- Hasta ahora hemos visto modelos que asumen una forma funcional específica (lineales, SVM) o que parten el espacio con reglas jerárquicas (árboles).
- Hoy exploraremos enfoques con supuestos diferentes:
  - **Basados en instancia (k-NN):** Usan la "memoria" del dataset completo para predecir. No aprenden un "modelo" explícito.
  - **Probabilísticos (Naive Bayes):** Modelan la probabilidad de pertenencia a una clase.

Contexto: ¿Por qué otros métodos?

- Hasta ahora hemos visto modelos que asumen una forma funcional específica (lineales, SVM) o que parten el espacio con reglas jerárquicas (árboles).
- Hoy exploraremos enfoques con supuestos diferentes:
  - **Basados en instancia (k-NN):** Usan la "memoria" del dataset completo para predecir. No aprenden un "modelo" explícito.
  - **Probabilísticos (Naive Bayes):** Modelan la probabilidad de pertenencia a una clase.
  - **Inspirados en la biología (Redes Neuronales):** Aprenden representaciones complejas de los datos a través de capas de neuronas interconectadas.

Contexto: ¿Por qué otros métodos?

- Hasta ahora hemos visto modelos que asumen una forma funcional específica (lineales, SVM) o que parten el espacio con reglas jerárquicas (árboles).
- Hoy exploraremos enfoques con supuestos diferentes:
  - **Basados en instancia (k-NN):** Usan la "memoria" del dataset completo para predecir. No aprenden un "modelo" explícito.
  - **Probabilísticos (Naive Bayes):** Modelan la probabilidad de pertenencia a una clase.
  - **Inspirados en la biología (Redes Neuronales):** Aprenden representaciones complejas de los datos a través de capas de neuronas interconectadas.
- Estos métodos ofrecen nuevas perspectivas para resolver problemas de clasificación y regresión.

k-Nearest Neighbors (k-NN): La Intuición

- Principio fundamental: *"Dime con quién andas y te diré quién eres"*.

k-Nearest Neighbors (k-NN): La Intuición

- Principio fundamental: *"Dime con quién andas y te diré quién eres"*.
- Para clasificar una nueva observación, k-NN mira las k observaciones más cercanas (sus "vecinos") en el conjunto de entrenamiento.

k-Nearest Neighbors (k-NN): La Intuición

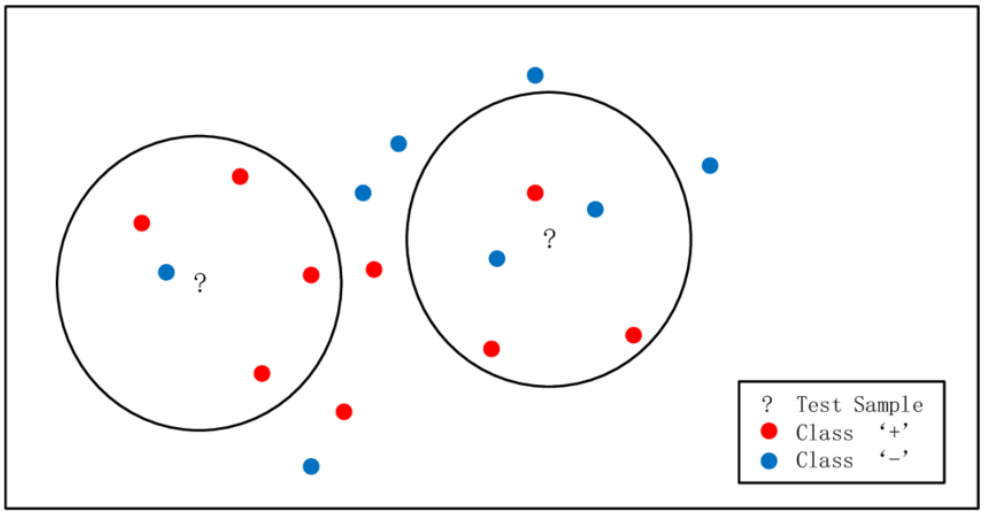
- Principio fundamental: *"Dime con quién andas y te diré quién eres"*.
- Para clasificar una nueva observación, k-NN mira las k observaciones más cercanas (sus "vecinos") en el conjunto de entrenamiento.
- La clase de la nueva observación se decide por "votación": la clase más común entre sus k vecinos.

k-Nearest Neighbors (k-NN): La Intuición

- Principio fundamental: *"Dime con quién andas y te diré quién eres"*.
- Para clasificar una nueva observación, k-NN mira las k observaciones más cercanas (sus "vecinos") en el conjunto de entrenamiento.
- La clase de la nueva observación se decide por "votación": la clase más común entre sus k vecinos.
- Es un método no paramétrico: no hace suposiciones sobre la forma funcional del límite de decisión.

k-Nearest Neighbors (k-NN): La Intuición

- Principio fundamental: *"Dime con quién andas y te diré quién eres"*.
- Para clasificar una nueva observación, k-NN mira las k observaciones más cercanas (sus "vecinos") en el conjunto de entrenamiento.
- La clase de la nueva observación se decide por "votación": la clase más común entre sus k vecinos.
- Es un método no paramétrico: no hace suposiciones sobre la forma funcional del límite de decisión.
- Es un algoritmo "perezoso" (lazy learner): no construye un modelo, simplemente almacena los datos de entrenamiento. La computación real ocurre en el momento de la predicción.



k-NN: El Algoritmo en Pasos

- **Paso 1: Elegir el número de vecinos (k).** Este es un hiperparámetro clave.

k-NN: El Algoritmo en Pasos

- **Paso 1: Elegir el número de vecinos (k).** Este es un hiperparámetro clave.
- **Paso 2: Calcular Distancias.** Para una nueva observación x_0, calcular la distancia a cada observación x_i en el dataset de entrenamiento.

k-NN: El Algoritmo en Pasos

- **Paso 1: Elegir el número de vecinos (k).** Este es un hiperparámetro clave.
- **Paso 2: Calcular Distancias.** Para una nueva observación x_0, calcular la distancia a cada observación x_i en el dataset de entrenamiento.
- **Paso 3: Identificar los k Vecinos.** Ordenar las distancias de menor a mayor y seleccionar las k observaciones de entrenamiento más cercanas a x_0.

k-NN: El Algoritmo en Pasos

- **Paso 1: Elegir el número de vecinos (k).** Este es un hiperparámetro clave.
- **Paso 2: Calcular Distancias.** Para una nueva observación x_0, calcular la distancia a cada observación x_i en el dataset de entrenamiento.
- **Paso 3: Identificar los k Vecinos.** Ordenar las distancias de menor a mayor y seleccionar las k observaciones de entrenamiento más cercanas a x_0.
- **Paso 4: Votación de Mayoría.**
  - **Para Clasificación:** Asignar a x_0 la clase que es más frecuente entre los k vecinos.
  - **Para Regresión:** Asignar a x_0 el promedio de los valores de la variable respuesta de los k vecinos.

k-NN: La Noción de “Cercanía”

- La definición de “cercanía” depende de la métrica de distancia utilizada.

k-NN: La Noción de "Cercanía"

- La definición de "cercanía" depende de la métrica de distancia utilizada.
- La elección de la métrica es crucial y debe ser apropiada para los datos.

k-NN: La Noción de "Cercanía"

- La definición de "cercanía" depende de la métrica de distancia utilizada.
- La elección de la métrica es crucial y debe ser apropiada para los datos.
- **Distancia Euclidiana (L2):** La más común, representa la distancia en línea recta entre dos puntos. $d(x, x') = \sqrt{\sum_{j=1}^{p}(x_j - x'_j)^2}$

k-NN: La Noción de "Cercanía"

- La definición de "cercanía" depende de la métrica de distancia utilizada.
- La elección de la métrica es crucial y debe ser apropiada para los datos.
- **Distancia Euclidiana (L2):** La más común, representa la distancia en línea recta entre dos puntos. $d(x, x') = \sqrt{\sum_{j=1}^{p}(x_j - x'_j)^2}$
- **Distancia de Manhattan (L1):** Suma de las diferencias absolutas de las coordenadas. Útil cuando las diferentes características no son directamente comparables. $d(x, x') = \sum_{j=1}^{p} |x_j - x'_j|$

k-NN: La Noción de "Cercanía"

- La definición de "cercanía" depende de la métrica de distancia utilizada.
- La elección de la métrica es crucial y debe ser apropiada para los datos.
- **Distancia Euclidiana (L2):** La más común, representa la distancia en línea recta entre dos puntos. $d(x, x') = \sqrt{\sum_{j=1}^{p}(x_j - x'_j)^2}$
- **Distancia de Manhattan (L1):** Suma de las diferencias absolutas de las coordenadas. Útil cuando las diferentes características no son directamente comparables. $d(x, x') = \sum_{j=1}^{p} |x_j - x'_j|$
- **Importancia del escalado:** Si las variables tienen escalas muy diferentes, la variable con la escala más grande dominará la distancia. Es fundamental estandarizar los datos antes de aplicar k-NN.

k-NN: La Elección de k y su Poder Teórico

- La elección de k controla el balance sesgo-varianza del modelo.

k-NN: La Elección de k y su Poder Teórico

- La elección de k controla el balance sesgo-varianza del modelo.
- **k pequeño (ej. k=1):**
  - **Bajo sesgo**, pero **Alta varianza**. Propenso al sobreajuste.

k-NN: La Elección de k y su Poder Teórico

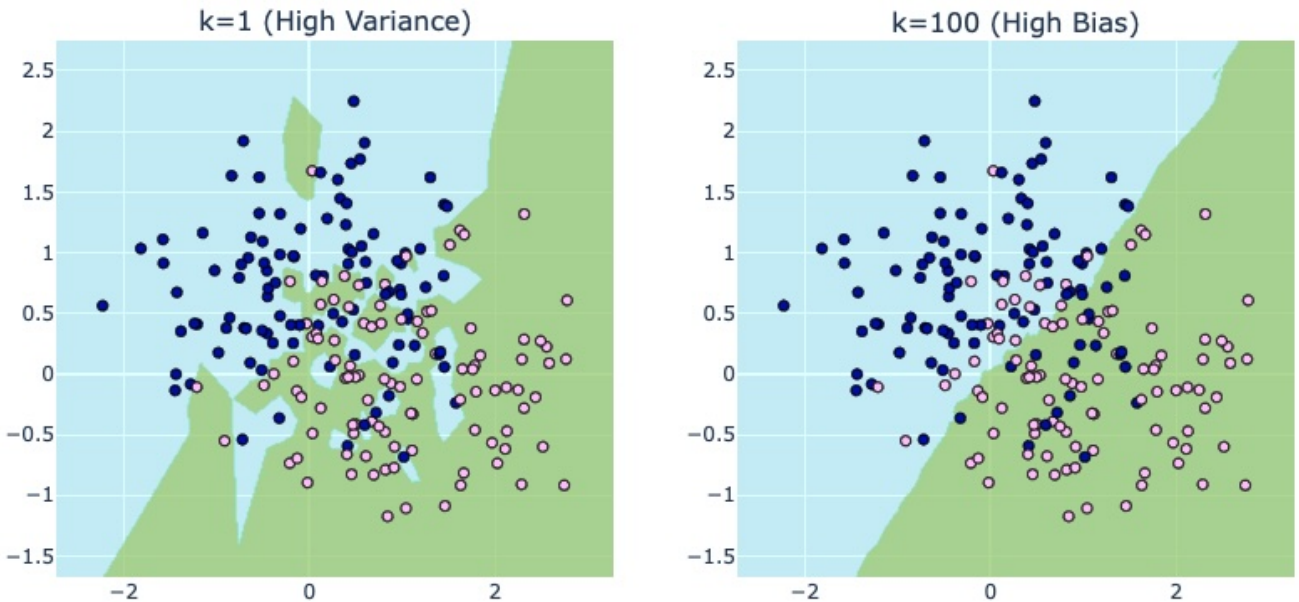
- La elección de k controla el balance sesgo-varianza del modelo.
- **k pequeño (ej. k=1):**
  - **Bajo sesgo**, pero **Alta varianza**. Propenso al sobreajuste.
- **k grande (ej. k=100):**
  - **Alto sesgo**, pero **Baja varianza**. Excesivamente suave.

k-NN: La Elección de k y su Poder Teórico

- La elección de k controla el balance sesgo-varianza del modelo.
- **k pequeño (ej. k=1):**
  - **Bajo sesgo**, pero **Alta varianza**. Propenso al sobreajuste.
- **k grande (ej. k=100):**
  - **Alto sesgo**, pero **Baja varianza**. Excesivamente suave.
- El valor óptimo de k se elige típicamente mediante validación cruzada.

k-NN: La Elección de k y su Poder Teórico

- La elección de k controla el balance sesgo-varianza del modelo.
- **k pequeño (ej. k=1):**
  - **Bajo sesgo**, pero **Alta varianza**. Propenso al sobreajuste.
- **k grande (ej. k=100):**
  - **Alto sesgo**, pero **Baja varianza**. Excesivamente suave.
- El valor óptimo de k se elige típicamente mediante validación cruzada.
- **Resultado Potente (para el pizarrón):** A pesar de su simplicidad, k-NN tiene garantías teóricas muy fuertes.



Naive Bayes: Clasificación Probabilística

- Cambiamos de enfoque: en lugar de un límite de decisión, modelamos la probabilidad de que una observación pertenezca a cada clase.

Naive Bayes: Clasificación Probabilística

- Cambiamos de enfoque: en lugar de un límite de decisión, modelamos la probabilidad de que una observación pertenezca a cada clase.
- Se basa en el **Teorema de Bayes**, que relaciona la probabilidad condicional de dos eventos.

Naive Bayes: Clasificación Probabilística

- Cambiamos de enfoque: en lugar de un límite de decisión, modelamos la probabilidad de que una observación pertenezca a cada clase.
- Se basa en el **Teorema de Bayes**, que relaciona la probabilidad condicional de dos eventos.
- El objetivo es calcular la probabilidad posterior $P(Y = c|X_1, ..., X_p)$ para cada clase c.

Naive Bayes: Clasificación Probabilística

- Cambiamos de enfoque: en lugar de un límite de decisión, modelamos la probabilidad de que una observación pertenezca a cada clase.
- Se basa en el **Teorema de Bayes**, que relaciona la probabilidad condicional de dos eventos.
- El objetivo es calcular la probabilidad posterior $P(Y = c|X_1, ..., X_p)$ para cada clase c.
- La regla de decisión es simple: asignar la observación a la clase con la probabilidad posterior más alta.

Naive Bayes: Clasificación Probabilística

- Cambiamos de enfoque: en lugar de un límite de decisión, modelamos la probabilidad de que una observación pertenezca a cada clase.
- Se basa en el **Teorema de Bayes**, que relaciona la probabilidad condicional de dos eventos.
- El objetivo es calcular la probabilidad posterior $P(Y = c|X_1, ..., X_p)$ para cada clase c.
- La regla de decisión es simple: asignar la observación a la clase con la probabilidad posterior más alta.
- Es un modelo "generativo", ya que modela la distribución de las características para cada clase.

Recordando el Teorema de Bayes

- El teorema nos permite "invertir" las probabilidades condicionales:

$$P(Y = c|X) = \frac{P(X|Y = c) \cdot P(Y = c)}{P(X)}$$

Recordando el Teorema de Bayes

- El teorema nos permite "invertir" las probabilidades condicionales:

$$P(Y = c|X) = \frac{P(X|Y = c) \cdot P(Y = c)}{P(X)}$$

- Traducido a nuestro problema:
  - $P(Y = c|X)$: **Probabilidad Posterior**. La que queremos calcular.

Recordando el Teorema de Bayes

- El teorema nos permite "invertir" las probabilidades condicionales:

$$P(Y = c|X) = \frac{P(X|Y = c) \cdot P(Y = c)}{P(X)}$$

- Traducido a nuestro problema:
  - $P(Y = c|X)$: **Probabilidad Posterior**. La que queremos calcular.
  - $P(X|Y = c)$: **Verosimilitud (Likelihood)**. Probabilidad de observar los predictores X dada la clase c.

Recordando el Teorema de Bayes

- El teorema nos permite "invertir" las probabilidades condicionales:

$$P(Y = c|X) = \frac{P(X|Y = c) \cdot P(Y = c)}{P(X)}$$

- Traducido a nuestro problema:
  - $P(Y = c|X)$: **Probabilidad Posterior**. La que queremos calcular.
  - $P(X|Y = c)$: **Verosimilitud (Likelihood)**. Probabilidad de observar los predictores X dada la clase c.
  - $P(Y = c)$: **Probabilidad a Priori (Prior)**. Probabilidad de la clase c antes de ver los datos.

Recordando el Teorema de Bayes

- El teorema nos permite "invertir" las probabilidades condicionales:

$$P(Y = c|X) = \frac{P(X|Y = c) \cdot P(Y = c)}{P(X)}$$

- Traducido a nuestro problema:
  - $P(Y = c|X)$: **Probabilidad Posterior**. La que queremos calcular.
  - $P(X|Y = c)$: **Verosimilitud (Likelihood)**. Probabilidad de observar los predictores X dada la clase c.
  - $P(Y = c)$: **Probabilidad a Priori (Prior)**. Probabilidad de la clase c antes de ver los datos.
  - $P(X)$: **Evidencia**. Probabilidad de los predictores. Actúa como un factor de normalización.

Recordando el Teorema de Bayes

- El teorema nos permite "invertir" las probabilidades condicionales:

$$P(Y = c|X) = \frac{P(X|Y = c) \cdot P(Y = c)}{P(X)}$$

- Traducido a nuestro problema:
  - $P(Y = c|X)$: **Probabilidad Posterior**. La que queremos calcular.
  - $P(X|Y = c)$: **Verosimilitud (Likelihood)**. Probabilidad de observar los predictores X dada la clase c.
  - $P(Y = c)$: **Probabilidad a Priori (Prior)**. Probabilidad de la clase c antes de ver los datos.
  - $P(X)$: **Evidencia**. Probabilidad de los predictores. Actúa como un factor de normalización.
- Para clasificar, no necesitamos calcular $P(X)$, ya que es la misma para todas las clases. Buscamos el máximo de:

$$\underset{c}{\text{argmax}} \left(P(X|Y = c) \cdot P(Y = c)\right)$$

El Supuesto "Ingenuo" de Naive Bayes

- Calcular $P(X|Y = c) = P(X_1, ..., X_p|Y = c)$ es muy difícil, ya que requiere modelar las dependencias entre todas las variables.

El Supuesto "Ingenuo" de Naive Bayes

- Calcular $P(X|Y = c) = P(X_1, ..., X_p|Y = c)$ es muy difícil, ya que requiere modelar las dependencias entre todas las variables.
- El supuesto **ingenuo** (naive) simplifica radicalmente el problema:
  - **Asume independencia condicional de las características, dada la clase.**

El Supuesto "Ingenuo" de Naive Bayes

- Calcular $P(X|Y = c) = P(X_1, ..., X_p|Y = c)$ es muy difícil, ya que requiere modelar las dependencias entre todas las variables.
- El supuesto **ingenuo** (naive) simplifica radicalmente el problema:
  - **Asume independencia condicional de las características, dada la clase.**
- Matemáticamente:

$$P(X_1, ..., X_p|Y = c) \approx P(X_1|Y = c) \cdot P(X_2|Y = c) \cdots P(X_p|Y = c)$$

El Supuesto "Ingenuo" de Naive Bayes

- Calcular $P(X|Y = c) = P(X_1, ..., X_p|Y = c)$ es muy difícil, ya que requiere modelar las dependencias entre todas las variables.
- El supuesto **ingenuo** (naive) simplifica radicalmente el problema:
  - **Asume independencia condicional de las características, dada la clase.**
- Matemáticamente:

$$P(X_1, ..., X_p|Y = c) \approx P(X_1|Y = c) \cdot P(X_2|Y = c) \cdots P(X_p|Y = c)$$

- Este supuesto es raramente cierto en la práctica, pero el clasificador funciona sorprendentemente bien.

El Supuesto "Ingenuo" de Naive Bayes

- Calcular $P(X|Y=c) = P(X_1, ..., X_p|Y=c)$ es muy difícil, ya que requiere modelar las dependencias entre todas las variables.
- El supuesto **ingenuo** (naive) simplifica radicalmente el problema:
  - **Asume independencia condicional de las características, dada la clase.**
- Matemáticamente:

$$P(X_1, ..., X_p|Y=c) \approx P(X_1|Y=c) \cdot P(X_2|Y=c) \cdots P(X_p|Y=c)$$

- Este supuesto es raramente cierto en la práctica, pero el clasificador funciona sorprendentemente bien.
- La tarea ahora se reduce a estimar $P(X_j|Y=c)$ para cada característica j y cada clase c por separado, lo cual es mucho más manejable.

Aplicación Típica: Clasificación de Texto

- Naive Bayes es muy popular para la clasificación de texto (ej. filtros de spam).

Aplicación Típica: Clasificación de Texto

- Naive Bayes es muy popular para la clasificación de texto (ej. filtros de spam).
- **Objetivo:** Clasificar un email como “Spam” o “No Spam”.

Aplicación Típica: Clasificación de Texto

- Naive Bayes es muy popular para la clasificación de texto (ej. filtros de spam).
- **Objetivo:** Clasificar un email como "Spam" o "No Spam".
- **Características (X_j):** La presencia o ausencia de cada palabra del diccionario en el email (modelo "bolsa de palabras" o Bag-of-Words).

Aplicación Típica: Clasificación de Texto

- Naive Bayes es muy popular para la clasificación de texto (ej. filtros de spam).
- **Objetivo:** Clasificar un email como "Spam" o "No Spam".
- **Características (X_j):** La presencia o ausencia de cada palabra del diccionario en el email (modelo "bolsa de palabras" o Bag-of-Words).
- El algoritmo aprende $P(\text{palabra}_j|\text{Spam})$ y $P(\text{palabra}_j|\text{No Spam})$ a partir de un corpus de emails etiquetados.

Aplicación Típica: Clasificación de Texto

- Naive Bayes es muy popular para la clasificación de texto (ej. filtros de spam).
- **Objetivo:** Clasificar un email como "Spam" o "No Spam".
- **Características (X_j):** La presencia o ausencia de cada palabra del diccionario en el email (modelo "bolsa de palabras" o Bag-of-Words).
- El algoritmo aprende $P(\text{palabra}_j|\text{Spam})$ y $P(\text{palabra}_j|\text{No Spam})$ a partir de un corpus de emails etiquetados.
- Para un nuevo email, se calcula la probabilidad posterior de ser spam usando las palabras que contiene.

Aplicación Típica: Clasificación de Texto

- Naive Bayes es muy popular para la clasificación de texto (ej. filtros de spam).
- **Objetivo:** Clasificar un email como "Spam" o "No Spam".
- **Características (X_j):** La presencia o ausencia de cada palabra del diccionario en el email (modelo "bolsa de palabras" o Bag-of-Words).
- El algoritmo aprende $P(\text{palabra}_j|\text{Spam})$ y $P(\text{palabra}_j|\text{No Spam})$ a partir de un corpus de emails etiquetados.
- Para un nuevo email, se calcula la probabilidad posterior de ser spam usando las palabras que contiene.
- ¿Por qué funciona bien a pesar del supuesto de independencia? El supuesto es incorrecto (ej. "viagra" y "oferta" no son independientes), pero las probabilidades calculadas a menudo son suficientes para discriminar correctamente.

Pausa

15 minutos

Redes Neuronales: El Perceptrón

- Inspiradas en el funcionamiento del cerebro humano.

Redes Neuronales: El Perceptrón

- Inspiradas en el funcionamiento del cerebro humano.
- La unidad fundamental es la **neurona** o **perceptrón**.

Redes Neuronales: El Perceptrón

- Inspiradas en el funcionamiento del cerebro humano.
- La unidad fundamental es la **neurona** o **perceptrón**.
- Un perceptrón:
  - Recibe una o más entradas $(X_1, ..., X_p)$.

Redes Neuronales: El Perceptrón

- Inspiradas en el funcionamiento del cerebro humano.
- La unidad fundamental es la **neurona** o **perceptrón**.
- Un perceptrón:
  - Recibe una o más entradas $(X_1, ..., X_p)$.
  - Calcula una suma ponderada de las entradas: $z = w_0 + \sum_{j=1}^{p} w_j X_j$.

Redes Neuronales: El Perceptrón

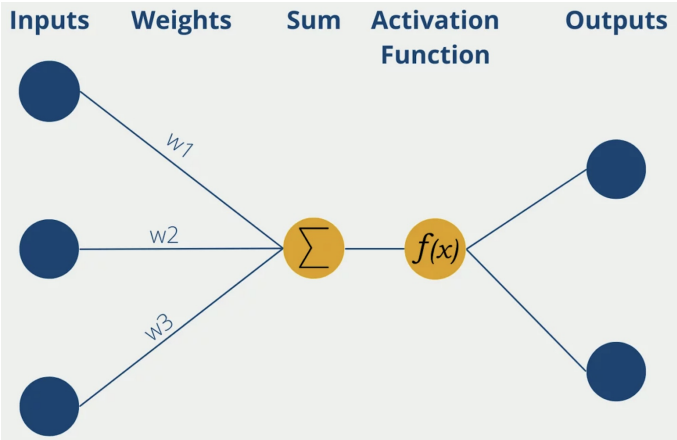
- Inspiradas en el funcionamiento del cerebro humano.
- La unidad fundamental es la **neurona** o **perceptrón**.
- Un perceptrón:
  - Recibe una o más entradas $(X_1, ..., X_p)$.
  - Calcula una suma ponderada de las entradas: $z = w_0 + \sum_{j=1}^{p} w_j X_j$.
  - Aplica una **función de activación** no lineal $g(z)$ para producir una salida.

Redes Neuronales: El Perceptrón

- Inspiradas en el funcionamiento del cerebro humano.
- La unidad fundamental es la **neurona** o **perceptrón**.
- Un perceptrón:
  - Recibe una o más entradas $(X_1, ..., X_p)$.
  - Calcula una suma ponderada de las entradas: $z = w_0 + \sum_{j=1}^{p} w_j X_j$.
  - Aplica una **función de activación** no lineal $g(z)$ para producir una salida.
- La función de activación introduce no linealidad, permitiendo al modelo aprender relaciones complejas.

Redes Neuronales: El Perceptrón

- Inspiradas en el funcionamiento del cerebro humano.
- La unidad fundamental es la **neurona** o **perceptrón**.
- Un perceptrón:
  - Recibe una o más entradas $(X_1, ..., X_p)$.
  - Calcula una suma ponderada de las entradas: $z = w_0 + \sum_{j=1}^{p} w_j X_j$.
  - Aplica una **función de activación** no lineal $g(z)$ para producir una salida.
- La función de activación introduce no linealidad, permitiendo al modelo aprender relaciones complejas.
- Por sí solo, un perceptrón solo puede aprender límites de decisión lineales.



De una Neurona a una Red: Perceptrón Multicapa (MLP)

- Para modelar relaciones más complejas, conectamos neuronas en capas.

De una Neurona a una Red: Perceptrón Multicapa (MLP)

- Para modelar relaciones más complejas, conectamos neuronas en capas.
- **Capa de Entrada (Input Layer):** Recibe las características originales.

De una Neurona a una Red: Perceptrón Multicapa (MLP)

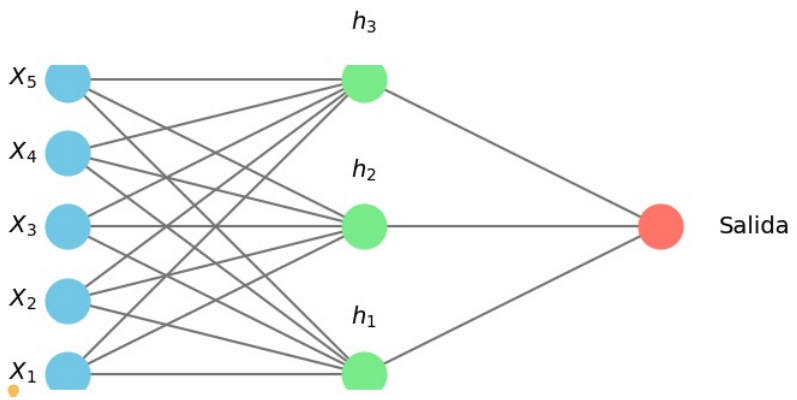
- Para modelar relaciones más complejas, conectamos neuronas en capas.
- **Capa de Entrada (Input Layer):** Recibe las características originales.
- **Capas Ocultas (Hidden Layers):** Una o más capas intermedias de neuronas. Cada neurona en una capa está conectada a todas las neuronas de la capa anterior.

De una Neurona a una Red: Perceptrón Multicapa (MLP)

- Para modelar relaciones más complejas, conectamos neuronas en capas.
- **Capa de Entrada (Input Layer):** Recibe las características originales.
- **Capas Ocultas (Hidden Layers):** Una o más capas intermedias de neuronas. Cada neurona en una capa está conectada a todas las neuronas de la capa anterior.
- **Capa de Salida (Output Layer):** Produce la predicción final. El número de neuronas depende del problema (una para regresión, C para clasificación con C clases).

De una Neurona a una Red: Perceptrón Multicapa (MLP)

- Para modelar relaciones más complejas, conectamos neuronas en capas.
- **Capa de Entrada (Input Layer):** Recibe las características originales.
- **Capas Ocultas (Hidden Layers):** Una o más capas intermedias de neuronas. Cada neurona en una capa está conectada a todas las neuronas de la capa anterior.
- **Capa de Salida (Output Layer):** Produce la predicción final. El número de neuronas depende del problema (una para regresión, C para clasificación con C clases).
- Al apilar capas, la red aprende representaciones de los datos cada vez más abstractas y complejas.



MLP: Funciones de Activación

- La elección de la función de activación es clave para el rendimiento de la red.

MLP: Funciones de Activación

- La elección de la función de activación es clave para el rendimiento de la red.
- **Función Sigmoide:** Transforma la entrada a un valor entre 0 y 1. Históricamente popular, pero puede sufrir de "gradientes evanecentes". $g(z) = (1 + e^{-z})^{-1}$

MLP: Funciones de Activación

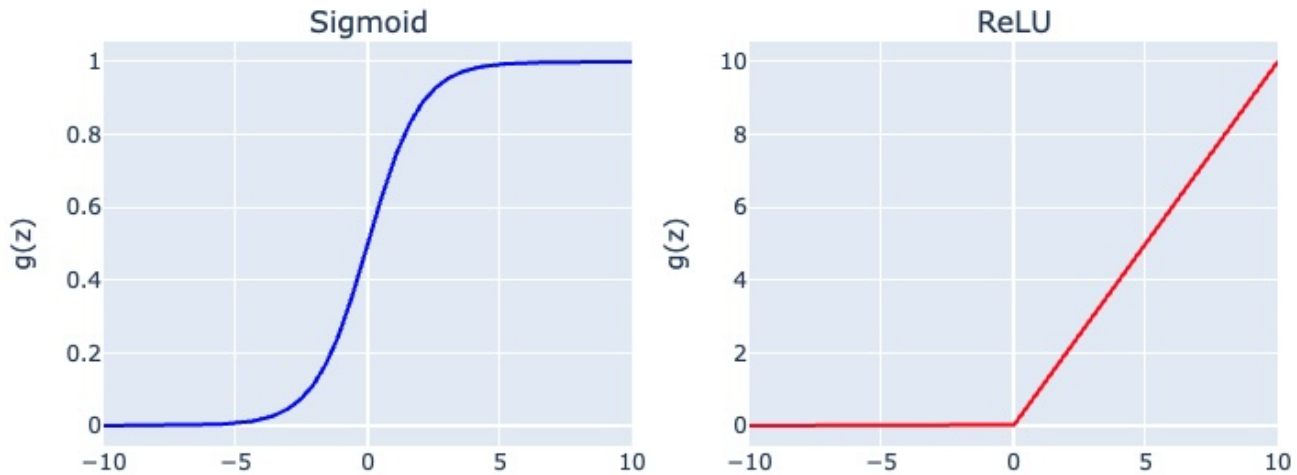
- La elección de la función de activación es clave para el rendimiento de la red.
- **Función Sigmoide:** Transforma la entrada a un valor entre 0 y 1. Históricamente popular, pero puede sufrir de "gradientes evanecentes". $g(z) = (1 + e^{-z})^{-1}$
- **Función ReLU (Rectified Linear Unit):** Muy popular actualmente por su simplicidad y eficiencia computacional. $g(z) = \text{máx}(0, z)$

MLP: Funciones de Activación

- La elección de la función de activación es clave para el rendimiento de la red.
- **Función Sigmoide:** Transforma la entrada a un valor entre 0 y 1. Históricamente popular, pero puede sufrir de "gradientes evanecentes". $g(z) = (1 + e^{-z})^{-1}$
- **Función ReLU (Rectified Linear Unit):** Muy popular actualmente por su simplicidad y eficiencia computacional. $g(z) = \text{máx}(0, z)$
- **Función Softmax:** Usada en la capa de salida para problemas de clasificación multiclase. Convierte las salidas en una distribución de probabilidad.

MLP: Funciones de Activación

- La elección de la función de activación es clave para el rendimiento de la red.
- **Función Sigmoide:** Transforma la entrada a un valor entre 0 y 1. Históricamente popular, pero puede sufrir de "gradientes evanecentes". $g(z) = (1 + e^{-z})^{-1}$
- **Función ReLU (Rectified Linear Unit):** Muy popular actualmente por su simplicidad y eficiencia computacional. $g(z) = \text{máx}(0, z)$
- **Función Softmax:** Usada en la capa de salida para problemas de clasificación multiclase. Convierte las salidas en una distribución de probabilidad.



MLP: Entrenamiento con Backpropagation

- El objetivo es encontrar los pesos (w) de todas las conexiones que minimicen una **función de pérdida** (ej. error cuadrático medio para regresión).

MLP: Entrenamiento con Backpropagation

- El objetivo es encontrar los pesos (w) de todas las conexiones que minimicen una **función de pérdida** (ej. error cuadrático medio para regresión).
- El algoritmo de entrenamiento más común es **Backpropagation** (retropropagación del error).

MLP: Entrenamiento con Backpropagation

- El objetivo es encontrar los pesos (w) de todas las conexiones que minimicen una **función de pérdida** (ej. error cuadrático medio para regresión).
- El algoritmo de entrenamiento más común es **Backpropagation** (retropropagación del error).
- **Pasos generales:**
  1. **Forward Pass:** Se presenta una observación a la red, se calcula la salida y se compara con el valor real para obtener el error.

MLP: Entrenamiento con Backpropagation

- El objetivo es encontrar los pesos (w) de todas las conexiones que minimicen una **función de pérdida** (ej. error cuadrático medio para regresión).
- El algoritmo de entrenamiento más común es **Backpropagation** (retropropagación del error).
- **Pasos generales:**
  1. **Forward Pass:** Se presenta una observación a la red, se calcula la salida y se compara con el valor real para obtener el error.
  2. **Backward Pass:** El error se propaga hacia atrás a través de la red, desde la capa de salida hasta la de entrada.

MLP: Entrenamiento con Backpropagation

- El objetivo es encontrar los pesos (w) de todas las conexiones que minimicen una **función de pérdida** (ej. error cuadrático medio para regresión).
- El algoritmo de entrenamiento más común es **Backpropagation** (retropropagación del error).
- **Pasos generales:**
  1. **Forward Pass:** Se presenta una observación a la red, se calcula la salida y se compara con el valor real para obtener el error.
  2. **Backward Pass:** El error se propaga hacia atrás a través de la red, desde la capa de salida hasta la de entrada.
  3. Se calcula el gradiente de la función de pérdida con respecto a cada peso.

MLP: Entrenamiento con Backpropagation

- El objetivo es encontrar los pesos (w) de todas las conexiones que minimicen una **función de pérdida** (ej. error cuadrático medio para regresión).
- El algoritmo de entrenamiento más común es **Backpropagation** (retropropagación del error).
- **Pasos generales:**
  1. **Forward Pass:** Se presenta una observación a la red, se calcula la salida y se compara con el valor real para obtener el error.
  2. **Backward Pass:** El error se propaga hacia atrás a través de la red, desde la capa de salida hasta la de entrada.
  3. Se calcula el gradiente de la función de pérdida con respecto a cada peso.
  4. Se actualizan los pesos en la dirección que minimiza el error (usando un optimizador como el Descenso de Gradiente).

MLP: Entrenamiento con Backpropagation

- El objetivo es encontrar los pesos (w) de todas las conexiones que minimicen una **función de pérdida** (ej. error cuadrático medio para regresión).
- El algoritmo de entrenamiento más común es **Backpropagation** (retropropagación del error).
- **Pasos generales:**
  1. **Forward Pass:** Se presenta una observación a la red, se calcula la salida y se compara con el valor real para obtener el error.
  2. **Backward Pass:** El error se propaga hacia atrás a través de la red, desde la capa de salida hasta la de entrada.
  3. Se calcula el gradiente de la función de pérdida con respecto a cada peso.
  4. Se actualizan los pesos en la dirección que minimiza el error (usando un optimizador como el Descenso de Gradiente).
- Este proceso se repite para muchas observaciones (o lotes) durante muchas épocas.

Comparación de Métodos: k-NN, Naive Bayes, MLP

Criterio	**k-NN**	**Naive Bayes**	**MLP (Simple)**
Supuestos	Datos similares están cerca; las características son comparables.	Independencia condicional de las características.	Aprende la estructura a través de capas.

Comparación de Métodos: k-NN, Naive Bayes, MLP

Criterio	**k-NN**	**Naive Bayes**	**MLP (Simple)**
Supuestos	Datos similares están cerca; las características son comparables.	Independencia condicional de las características.	Aprende la estructura a través de capas.
Complejidad	Alta en predicción (calcula todas las distancias).	Baja, solo conteo de frecuencias/parámetros.	Muy alta en entrenamiento (backpropagation).

Comparación de Métodos: k-NN, Naive Bayes, MLP

Criterio	**k-NN**	**Naive Bayes**	**MLP (Simple)**
Supuestos	Datos similares están cerca; las características son comparables.	Independencia condicional de las características.	Aprende la estructura a través de capas.
Complejidad	Alta en predicción (calcula todas las distancias).	Baja, solo conteo de frecuencias/parámetros.	Muy alta en entrenamiento (backpropagation).
Interpretabilidad	Baja. La predicción depende de un subconjunto local.	Moderada. Se puede ver la influencia de cada característica.	Muy baja (caja negra).

Comparación de Métodos: k-NN, Naive Bayes, MLP

Criterio	k-NN	Naive Bayes	MLP (Simple)
Supuestos	Datos similares están cerca; las características son comparables.	Independencia condicional de las características.	Aprende la estructura a través de capas.
Complejidad	Alta en predicción (calcula todas las distancias).	Baja, solo conteo de frecuencias/parámetros.	Muy alta en entrenamiento (backpropagation).
Interpretabilidad	Baja. La predicción depende de un subconjunto local.	Moderada. Se puede ver la influencia de cada característica.	Muy baja (caja negra).
Overfitting	Riesgo alto con k pequeño.	Riesgo bajo-moderado.	Riesgo muy alto. Necesita regularización.

Comparación de Métodos: k-NN, Naive Bayes, MLP

Criterio	k-NN	Naive Bayes	MLP (Simple)
Supuestos	Datos similares están cerca; las características son comparables.	Independencia condicional de las características.	Aprende la estructura a través de capas.
Complejidad	Alta en predicción (calcula todas las distancias).	Baja, solo conteo de frecuencias/parámetros.	Muy alta en entrenamiento (backpropagation).
Interpretabilidad	Baja. La predicción depende de un subconjunto local.	Moderada. Se puede ver la influencia de cada característica.	Muy baja (caja negra).
Overfitting	Riesgo alto con k pequeño.	Riesgo bajo-moderado.	Riesgo muy alto. Necesita regularización.
Ideal para...	Problemas con límites de decisión complejos y no muchos datos/dims.	Clasificación de texto, problemas con muchas características.	Problemas complejos (imágenes, voz) con muchos datos.

Ventajas y Desventajas Clave

- **k-NN:**
  - **Ventajas:** Simple de entender e implementar. No requiere entrenamiento. Flexible.
  - **Desventajas:** Computacionalmente caro en la predicción. Sensible a características irrelevantes y a la escala de los datos ("maldición de la dimensionalidad").

Ventajas y Desventajas Clave

- **k-NN:**
  - **Ventajas:** Simple de entender e implementar. No requiere entrenamiento. Flexible.
  - **Desventajas:** Computacionalmente caro en la predicción. Sensible a características irrelevantes y a la escala de los datos ("maldición de la dimensionalidad").
- **Naive Bayes:**
  - **Ventajas:** Muy rápido y eficiente. Funciona bien con muchas características y pocos datos.
  - **Desventajas:** El supuesto de independencia es a menudo irreal.

Ventajas y Desventajas Clave

- **k-NN:**
  - **Ventajas:** Simple de entender e implementar. No requiere entrenamiento. Flexible.
  - **Desventajas:** Computacionalmente caro en la predicción. Sensible a características irrelevantes y a la escala de los datos ("maldición de la dimensionalidad").
- **Naive Bayes:**
  - **Ventajas:** Muy rápido y eficiente. Funciona bien con muchas características y pocos datos.
  - **Desventajas:** El supuesto de independencia es a menudo irreal.
- **MLP (Simple):**
  - **Ventajas:** Capacidad de aprender relaciones muy complejas y no lineales. Es la base de los modelos de Deep Learning.
  - **Desventajas:** Difícil de interpretar. Propenso al sobreajuste. Requiere muchos datos y ajuste de hiperparámetros.

Ventajas y Desventajas Clave

- **k-NN:**
  - **Ventajas:** Simple de entender e implementar. No requiere entrenamiento. Flexible.
  - **Desventajas:** Computacionalmente caro en la predicción. Sensible a características irrelevantes y a la escala de los datos ("maldición de la dimensionalidad").
- **Naive Bayes:**
  - **Ventajas:** Muy rápido y eficiente. Funciona bien con muchas características y pocos datos.
  - **Desventajas:** El supuesto de independencia es a menudo irreal.
- **MLP (Simple):**
  - **Ventajas:** Capacidad de aprender relaciones muy complejas y no lineales. Es la base de los modelos de Deep Learning.
  - **Desventajas:** Difícil de interpretar. Propenso al sobreajuste. Requiere muchos datos y ajuste de hiperparámetros.

Resumen y Conclusiones

- Exploramos tres familias de algoritmos muy diferentes a las vistas previamente: basados en instancia, probabilísticos y redes neuronales.

Resumen y Conclusiones

- Exploramos tres familias de algoritmos muy diferentes a las vistas previamente: basados en instancia, probabilísticos y redes neuronales.
- Cada uno tiene un conjunto único de supuestos, fortalezas y debilidades.

Resumen y Conclusiones

- Exploramos tres familias de algoritmos muy diferentes a las vistas previamente: basados en instancia, probabilísticos y redes neuronales.
- Cada uno tiene un conjunto único de supuestos, fortalezas y debilidades.
- No hay un "mejor" algoritmo universal. La elección depende del problema, la cantidad y dimensionalidad de los datos, y el objetivo (predicción vs. interpretabilidad).

Resumen y Conclusiones

- Exploramos tres familias de algoritmos muy diferentes a las vistas previamente: basados en instancia, probabilísticos y redes neuronales.
- Cada uno tiene un conjunto único de supuestos, fortalezas y debilidades.
- No hay un "mejor" algoritmo universal. La elección depende del problema, la cantidad y dimensionalidad de los datos, y el objetivo (predicción vs. interpretabilidad).
- k-NN y Naive Bayes son excelentes herramientas, a menudo subestimadas, que pueden servir como buenos modelos de base (baselines).

Resumen y Conclusiones

- Exploramos tres familias de algoritmos muy diferentes a las vistas previamente: basados en instancia, probabilísticos y redes neuronales.
- Cada uno tiene un conjunto único de supuestos, fortalezas y debilidades.
- No hay un "mejor" algoritmo universal. La elección depende del problema, la cantidad y dimensionalidad de los datos, y el objetivo (predicción vs. interpretabilidad).
- k-NN y Naive Bayes son excelentes herramientas, a menudo subestimadas, que pueden servir como buenos modelos de base (baselines).
- El MLP simple es nuestra puerta de entrada al mundo del Deep Learning, una de las áreas más activas en Machine Learning hoy en día.

Resumen y Conclusiones

- Exploramos tres familias de algoritmos muy diferentes a las vistas previamente: basados en instancia, probabilísticos y redes neuronales.
- Cada uno tiene un conjunto único de supuestos, fortalezas y debilidades.
- No hay un "mejor" algoritmo universal. La elección depende del problema, la cantidad y dimensionalidad de los datos, y el objetivo (predicción vs. interpretabilidad).
- k-NN y Naive Bayes son excelentes herramientas, a menudo subestimadas, que pueden servir como buenos modelos de base (baselines).
- El MLP simple es nuestra puerta de entrada al mundo del Deep Learning, una de las áreas más activas en Machine Learning hoy en día.
- **A continuación:** Taller práctico para implementar y comparar estos modelos.