Para estudiantes de Educación Básica y Media.
UNIVERSIDAD DE CHILE

# Clase 12: Modelos Clasicos de Aprendizaje Supervisado

FM849 - Programación Científica para Proyectos de Inteligencia Artificial (IA)

# Contenidos de esta clase

Algunos de los modelos mas utilizados en Machine Learning:

- Recuerdo sobre Aprendizaje Supervisado.
- Support Vector Machine (SVM).
- k-Nearest Neighbors (k-NN)
- Naïve Bayes.

# Recuerdo de Aprendizaje Supervisado

**Idea General**: Tenemos pares features y etiquetas $\{\overrightarrow{x}_i, y_i\}$. Queremos encontrar un modelo $h$, al cual le entregaremos features de un ejemplo $\overrightarrow{x}_i$ y retornara la etiqueta $y_i$.

Esto es lo mismo que decir

$$h(\overrightarrow{x}_i) = y_i$$

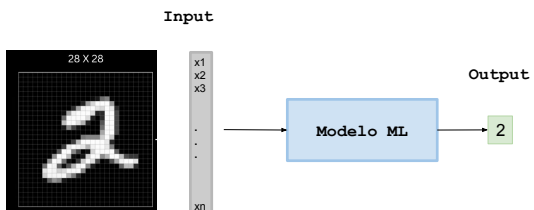



Figura 1: Esquema de un modelo ML en aprendizaje supervisado.

# Recuerdo de Aprendizaje Supervisado

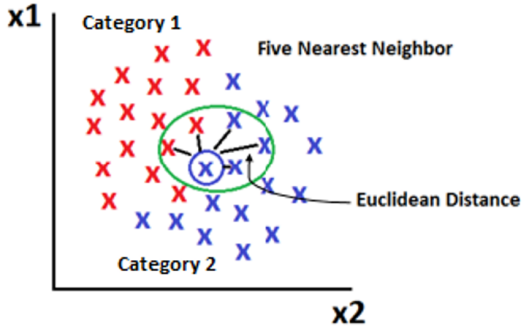
Estudiaremos diferentes tipos de modelos utilizados para resolver el problema de clasificación. Empezaremos dando una idea general, luego un ejemplo de como utilizar el modelo para clasificar y finalmente hablaremos de detalles que existen en cada modelo.

- k-Nearest Neighbors (k-NN)
- Naïve Bayes.
- Support Vector Machine (SVM).

# k-Nearest Neighbors (kNN)

**Idea:** Clasificamos un dato según que tan similar es todo nuestro dataset.

- Dado un ejemplo desconocido a clasificar, busca los $k$ ejemplos conocidos más cercanos. Por defecto, cada clase suma 1 voto si un ejemplo suyo está entre los $k$ más cercanos.
- El nuevo ejemplo es clasificado con la clase de mayor votación.



# k-Nearest Neighbors (kNN)

Tenemos que **buscar** los ejemplos más cercanos. **Como hacemos esto ?** Necesitamos una métrica de distancia para buscarlos ejemplos mas cercanos.

- Distancia euclidiana.

$$d_{eucli}(x, z) = \sqrt{\sum_{i=1}^{d}(x_i - z_i)^2}$$

**Problema:** Es importante escalar los datos para esta métrica, pues ciertas dimensiones en los datos pueden tener mayores magnitudes.

# Nuestro problema a resolver

Supongamos que tenemos una tabla con datos de mensajes (Spam y no Spam). Calculamos dos features (características) para cada mensaje.

| **Oración** | **Etiqueta** |
|---|---|
| Oferta exclusiva, paga y gana dinero | Spam |
| Confirma el pago para recibir dinero | Spam |
| Gran oferta disponible hoy | Spam |
| Hola amigo, ¿cómo está tu hijo? | No spam |
| Amigo, te paso el dinero mañana | No spam |
| Amiga amiga amiga | No spam |

Tabla 1: Mensajes originales

- $x_1$: Número de veces que ocurre [oferta, pago, dinero].
- $x_2$: Número de veces que ocurre [hijo, hija, amigo].

# Nuestro problema a resolver

Supongamos que tenemos una tabla con datos de mensajes (Spam y no Spam). Calculamos dos features (características) para cada mensaje.

| **Oración** | **Etiqueta** |
|---|---|
| Oferta exclusiva, paga y gana dinero | Spam |
| Confirma el pago para recibir dinero | Spam |
| Gran oferta disponible hoy | Spam |
| Hola amigo, ¿cómo está tu hijo? | No spam |
| Amigo, te paso el dinero mañana | No spam |
| Amiga amiga amiga | No spam |

Tabla 1: Mensajes originales

- $x_1$: Número de veces que ocurre [oferta, pago, dinero].
- $x_2$: Número de veces que ocurre [hijo, hija, amigo].

# Nuestro problema a resolver

Supongamos que tenemos una tabla con datos de mensajes (Spam y no Spam). Calculamos dos features (características) para cada mensaje.

| **Oración** | $x_1$ | $x_2$ | **Etiqueta** |
|---|---|---|---|
| Oferta exclusiva, paga y gana dinero | 3 | 0 | Spam |
| Confirma el pago para recibir dinero | 2 | 0 | Spam |
| Gran oferta disponible hoy | 1 | 0 | Spam |
| Hola amigo, ¿cómo está tu hijo? | 0 | 2 | No spam |
| Amigo, te paso el dinero mañana | 1 | 1 | No spam |
| Amiga amiga amiga | 0 | 3 | No spam |

Tabla 1: Mensajes con features

- $x_1$: Número de veces que ocurre [oferta, pago, dinero].
- $x_2$: Número de veces que ocurre [hijo, hija, amigo].

# KNN: problema de escalamiento

Supongamos que usamos KNN con dos variables:

- $x_1$: edad (años)
- $x_2$: ingresos (pesos)

Punto a clasificar:

$$x = (25,\ 900\,000)$$

Dos ejemplos del conjunto de entrenamiento:

$$z_1 = (20,\ 1\,000\,000), \quad z_2 = (40,\ 850\,000)$$

**Distancia euclidiana sin escalar:**

$$d(x, z_1) \approx \sqrt{(25 - 20)^2 + (900000 - 1000000)^2}$$

# KNN: problema de escalamiento (solución)

Para evitar que una variable domine a las demás, normalizamos cada característica usando **min–max**:

$$x_i^{\text{norm}} = \frac{x_i - \text{mín}(x_i)}{\text{máx}(x_i) - \text{mín}(x_i)}$$

Supongamos que, tras normalizar a $[0, 1]$, obtenemos:

$$x = (0.3,\ 0.4), \quad z_1 = (0.2,\ 0.6), \quad z_2 = (0.6,\ 0.35)$$

$$d(x, z_1) = \sqrt{(0.3 - 0.2)^2 + (0.4 - 0.6)^2}$$

$$d(x, z_2) = \sqrt{(0.3 - 0.6)^2 + (0.4 - 0.35)^2}$$

# KNN: Problema de la dimensionalidad

En altas dimensiones, las funciones de distancia pierden la capacidad de distinguir elementos que están muy cerca o muy lejos. Las distancias se empiezan a concentrar alrededor de un valor.

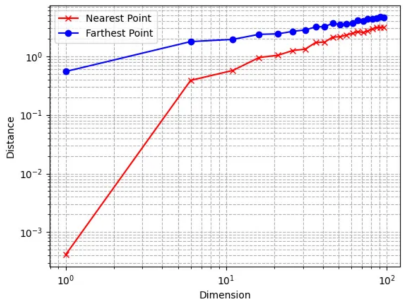



Figura 2: Variación entre distancia minima y maxima para datos aleatorios con diferente numero de dimensiones. Fuente: https://medium.com/data-science/the-math-behind-the-curse-of-dimensionality-cf8780307d74

# Naive Bayes

**Idea:** Dado un conjunto de features vamos a generar una probabilidad de ocurrencia para cada clase $y_i$.

Dado un input $x$ de $n$ features $x = [x_1, x_2, ..., x_n]$ queremos obtener $P(y_i|x_1, x_2, ..., x_n)$. Para hacer esto deberiamos buscar los casos dado $x$ ocurre $y_i$, lo cual es intratable computacionalmente

La idea sera entonces convertir la expresión $P(y_i|x)$ en algo mas simple y calculable. Usamos dos cosas: Teorema de Bayes e Independencia Condicional.

- Teorema de Bayes: $P(A|B) = \frac{P(B|A) \cdot P(A)}{P(B)}$
- Independencia Condicional: Dada una clase $y_i$, los features $x_1, ..., x_n$ ocurren de manera independiente.

# Naive Bayes

Entonces usando el Teorema de Bayes tenemos que:

$$P(y_i|x_1, x_2, ..., x_n) = \frac{P(x_1, x_2, ..., x_n|y_i)P(y_i)}{P(x_1, x_2, ..., x_n)}$$

Usando la independencia condicional entre los features llegamos a:

$$P(y_i|x_1, x_2, ..., x_n) = \frac{P(x_1|y_i) \cdot P(x_2|y_i) \cdots P(x_n|y_i) \cdot P(y_i)}{P(x_1, x_2, ..., x_n)}$$

Podemos ver que el denominador es una constante para todas las clases del problema, entonces:

$$\hat{y} = \arg \max_{y_i \in Y} P(x_1|y_i) \cdot P(x_2|y_i) \cdots P(x_n|y_i) \cdot P(y_i)$$

# Naive Bayes: Ejemplo

| | Outlook | Temperature | Humidity | Windy | Play Golf |
|---|---|---|---|---|---|
| 0 | Rainy | Hot | High | False | Yes |
| 1 | Rainy | Hot | High | True | No |
| 2 | Overcast | Hot | High | False | Yes |
| 3 | Sunny | Mild | High | False | Yes |
| 4 | Sunny | Cool | Normal | False | Yes |
| 5 | Sunny | Cool | Normal | True | No |
| 6 | Overcast | Cool | Normal | True | Yes |
| 7 | Rainy | Mild | High | False | No |
| 8 | Rainy | Cool | Normal | False | Yes |
| 9 | Sunny | Mild | Normal | False | Yes |
| 10 | Rainy | Mild | Normal | True | Yes |
| 11 | Overcast | Mild | High | True | Yes |
| 12 | Overcast | Hot | Normal | False | Yes |
| 13 | Sunny | Mild | High | True | No |

Que predice el modelo cuando $X =$ (Sunny,Hot,Normal,False) ?

# Support Vector Machine

**Idea:** Generar un espacio de separación (hiperplano) perpendicular a los datos que separe las clases.

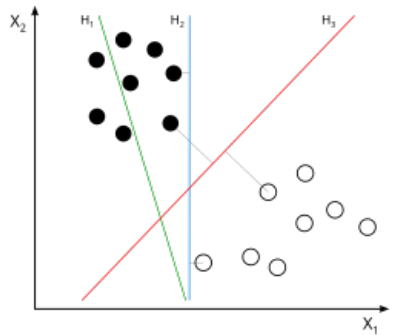



Figura 3: Ejemplo de hiperplanos. Fuente: https://en.wikipedia.org/wiki/Support_vector_machine.

# Support Vector Machine

- Corresponde a un método que busca separar las clases con un hiperplano (en 2D, una recta).
- El hiperplano elegido es el que maximiza el margen que se genera con los vectores de soporte.
- Cada vez que evaluamos un nuevo punto en la ecuación del hiperplano $w \cdot x + b = 0$, clasificamos según el signo de $w \cdot x + b$.

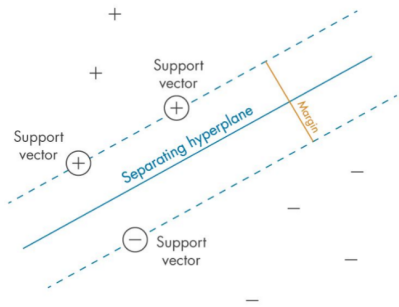



Figura 4: Ilustración de SVM.

# Problema de Outliers en Support Vector Machine

Los **outliers** no permiten generar una separación perfecta cuando usamos SVM.

- En esos casos conviene generar un modelo mas flexible, es decir, le permitimos equivocarse en algunos ejemplos.
- La idea sera hacer la çalle"tan larga como sea posible con la menor cantidad de ejemplos errados. Esto se controla con un hiperparametro $C$ del modelo.
- Esta técnica es comúnmente llamada *soft Margin*.

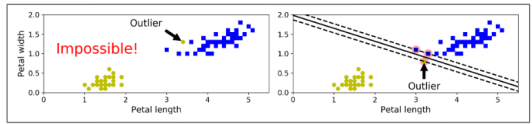



Figura 5: Ejemplos de datos con outliers.

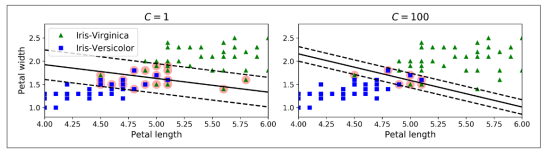
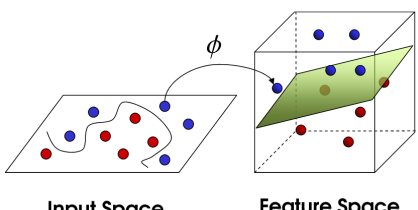



Figura 6: Modelo con *soft Margin*.

# Support Vector Machine: Kernel

Existen casos en los cuales los datos no son linearmente separables (por ejemplo Figura 16). En estos casos, SVM utiliza una herramienta matematica llamada **kernel**.

El kernel transforma el espacio original de input features en un nuevo espacio vectorial, de tal forma de poder separar los datos.



# Resumen de Ventajas y Desventajas de cada algoritmo

| **Modelo** | **Ventaja** | **Desventaja** |
|---|---|---|
| kNN | No requiere entrenamiento explícito. | Sensible a la dimensionalidad. |
| NB | Eficiente incluso en alta dimensión. | Independencia condicional poco realista. |
| SVM | Data efficient ($\sim$ 1000). | Costoso y sensible a hiperparámetros. |

# Referencias:


- https://medium.com/data-science/the-math-behind-the-curse-of-dimensionality-cf8780307