Machine learning y deep learning

Rolando Gonzales Martinez, PhD

Fellow postdoctoral Marie Skłodowska-Curie

Universidad de Groningen (Países Bajos)

Investigador (researcher)

Iniciativa de Pobreza y Desarrollo Humano de la Universidad de Oxford (UK)

Contenido del curso

(3) Fundamentos de Machine Learning

- Preprocesamiento de datos: detección, limpieza y tratamiento de datos.
- Selección y extracción de características.
- Modelos de regresión y clasificación: árboles de decisión, bosques aleatorios, máquinas de soporte vectorial, redes elásticas y otros modelos y algoritmos
- Laboratorio: Implementación de modelos de machine learning básicos.

Preprocesamiento de datos: detección, limpieza y tratamiento de datos

Imputación de missings (datos perdidos):

- Hot deck: seleccionar un valor fijo de otra observación con las mismas covariables, en la misma base de datos
- Cold deck: seleccionar un valor fijo de otra observación con las mismas covariables, en una base externa diferente
- Imputación con la media, la mediana u otra medida de tendencia central
- Imputación con regresión: se utiliza el ajuste de una regresión para imputar el valor perdido.
- Imputación múltiple con ecuaciones encadenadas (MICE): asume los criterios MCAR y MAR de Rubin.
- …

Métodos sin imputación pero que consideran valores missing: Máxima Verosimilitud de Información Completa (FIML). Asume MCAR y MAR.

Preprocesamiento de datos: detección, limpieza y tratamiento de datos

Detección de outliers:

- Metodos clasicos: basados en la desviación estándar.
- No-parametricos: en base al rango intercuartil.
- Inferenciales: test de Grubbs, el test de Dixon (nula: no hay outliers).
- Métodos modernos en el contexto de ML: isolation forests, LOF

$$X = \{x_1, x_2, \ldots, x_n\}$$

$$G = \frac{\max\left(|x_i - \bar{x}|\right)}{s}$$

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i \quad s = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2}$$

$$G_{\text{crítico}} = \frac{(n-1)}{\sqrt{n}} \cdot \sqrt{\frac{t^2_{\alpha/(2n),n-2}}{n-2+t^2_{\alpha/(2n),n-2}}}$$

Preprocesamiento de datos: detección, limpieza y tratamiento de datos

Tratamiento de outliers:

- Eliminar los outliers (trimming)
- Transformar la variables
- Winsorizacion: limitar los valores extremos a un percentil específico
- Imputación con la tendencia central
- Utilizar modelos que sean robustos a outliers (RF, SVM)

$$X = \{x_1, x_2, \ldots, x_n\}$$

$$G = \frac{\max\left(|x_i - \bar{x}|\right)}{s}$$

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i \quad s = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2}$$

$$G_{\text{crítico}} = \frac{(n-1)}{\sqrt{n}} \cdot \sqrt{\frac{t^2_{\alpha/(2n),n-2}}{n-2+t^2_{\alpha/(2n),n-2}}}$$

Preprocesamiento de datos: detección, limpieza y tratamiento de datos

Transformaciones de datos:

- Escalamiento min-max y normalización
- Transformación logarítmica
- Transformación de Box-Cox
- Binarización de variables categóricas
- Transformación de variables categóricas y continuas en WoE

$$x' = \frac{x - \min(x)}{\max(x) - \min(x)}$$

$$z = \frac{x - \mu}{\sigma}$$

$$x' = \log(x + c)$$

$$x' = \begin{cases} \frac{(x^\lambda - 1)}{\lambda} & \text{si } \lambda \neq 0, \\ \log(x) & \text{si } \lambda = 0. \end{cases}$$

$$\text{Si } x_i = k, \text{ entonces } \mathbf{x'}_i = (0, 0, ..., 1, ..., 0)$$

$$\text{WoE}_i = \ln\left(\frac{\text{Proporción de Buenos}_i}{\text{Proporción de Malos}_i}\right)$$

Preprocesamiento de datos: detección, limpieza y tratamiento de datos

Reducción de la dimensionalidad:

- Reduce el ruido en los datos
- Reduce la multicolinealidad (mejorando la precisión)
- Reduce la complejidad computacional (acelera el entrenamiento)
- Puede mejorar el desempeño de los modelos ML al eliminar caracteristicas (features) redundantes o irrelevantes.
- Facilita la visualización

$$\mathbf{X}_{centrada} = \mathbf{X} - \text{media}(\mathbf{X})$$

$$\mathbf{C} = \frac{1}{n-1}\mathbf{X}_{centrada}^{\top}\mathbf{X}_{centrada}$$

$$\mathbf{C}\mathbf{v}_i = \lambda_i \mathbf{v}_i$$

$$\mathbf{Z} = \mathbf{X}_{centrada}\mathbf{V}$$

Machine learning y deep learning: bosques aleatorios

Rolando Gonzales Martinez, PhD

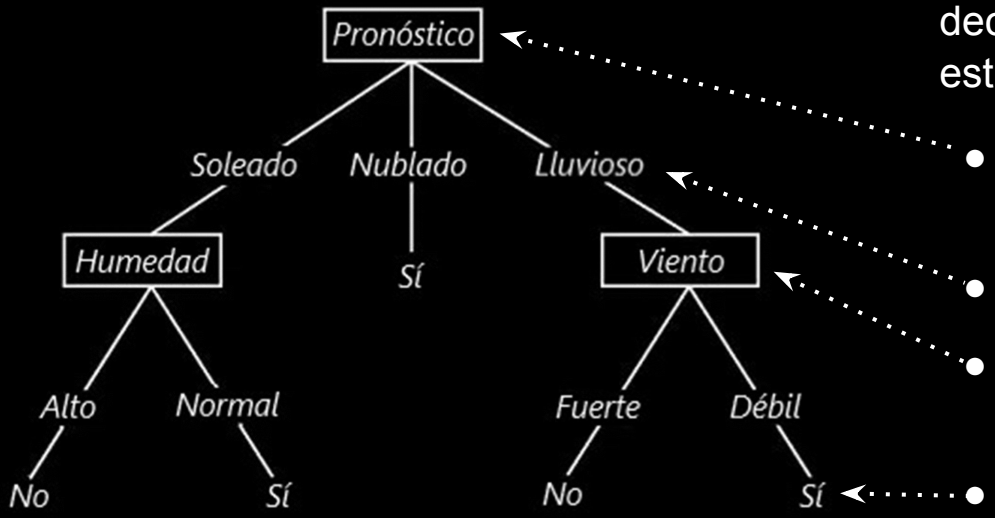
Fellow postdoctoral Marie Skłodowska-Curie

Universidad de Groningen (Países Bajos)

Investigador (researcher)

Iniciativa de Pobreza y Desarrollo Humano de la Universidad de Oxford (UK)

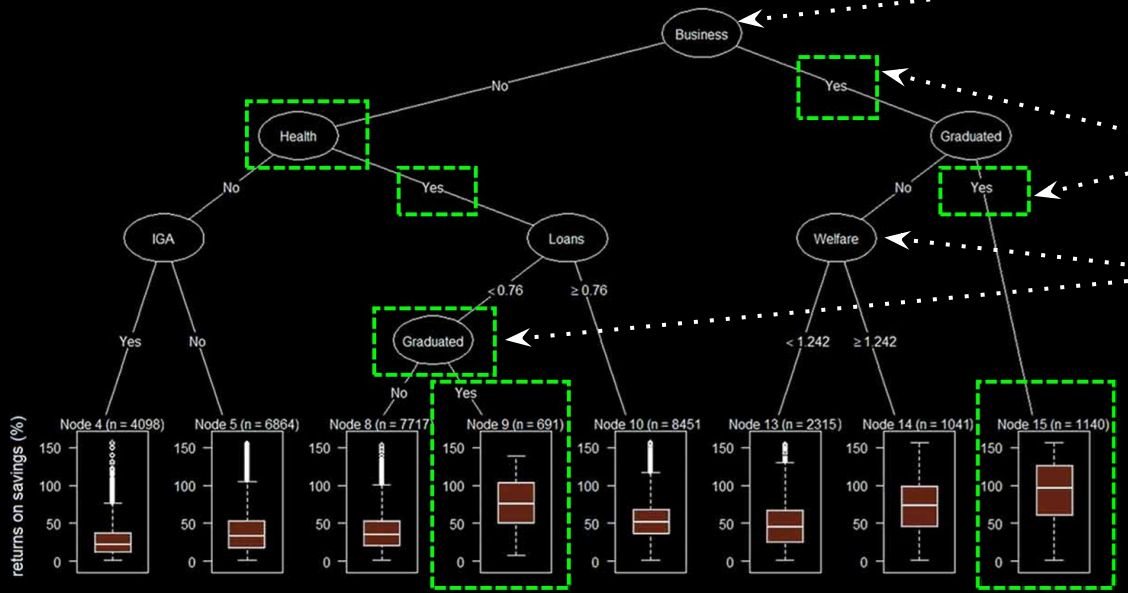
Árbol de decisión



Un árbol de decisión es una estructura jerárquica en la que se realiza una serie de decisiones secuenciales. TIene la siguiente estructura:

- **Raíz:** El nodo superior del árbol donde comienza la división de los datos.
- **Ramas:** que representan las divisiones
- **Nodos:** representan opciones de división para realizar divisiones adicionales
- **Hojas:** Representan las decisiones finales o las clasificaciones.

Árbol de decisión



Raíz: El nodo superior del árbol donde comienza la división de los datos.

Ramas: que representan las divisiones

Nodos: representan opciones de división para realizar divisiones adicionales

Hojas: Representan las decisiones finales o las clasificaciones.

Árbol de decisión: estadigrafo de Gini

- El Gini es una medida de pureza utilizada en la construcción de árboles de decisión.
- Se usa para evaluar la calidad de una división o partición en los nodos del árbol. El objetivo del índice de Gini es determinar cuán mezclados están los datos
- El Gini mide la proporción de clases en un nodo particular.

$$Gini = 1 - (p_1^2 + p_2^2)$$

El índice de Gini oscila entre 0 y 0.5.

- **Gini = 0**: Indica que todas las instancias en el nodo pertenecen a una sola clase, es decir, el nodo es *puro*.
- **Gini cercano a 0.5**: Indica que las instancias en el nodo están distribuidas equitativamente entre las clases, lo que significa que el nodo es *impuro*.

Árbol de decisión: Nodo Raiz

El nodo inicial del árbol de decisión se escoge mediante una evaluación de todas las características (features) y posibles umbrales, seleccionando la combinación que minimice la impureza de los nodos hijos resultantes:

$$D_{\text{izq}} \text{ con las instancias donde } x_i \leq umbral.$$

$$D_{\text{der}} \text{ con las instancias donde } x_i > umbral.$$

$$I_{\text{hijos}}(umbral) = \frac{|D_{\text{izq}}|}{|D|} \cdot I(D_{\text{izq}}) + \frac{|D_{\text{der}}|}{|D|} \cdot I(D_{\text{der}})$$

$$x^*, umbral^* = \arg \min_{x_i, umbral} (I_{\text{hijos}}(umbral))$$

Árbol de decisión: Algoritmo

1. Selección de la Característica y el Umbral de partición:

- **Para cada característica**: El algoritmo evalúa todos los posibles puntos de división (umbrales).
- **Para cada umbral posible**: Se separan las muestras en dos grupos: uno donde la característica es menor o igual al umbral, y otro donde es mayor.
- **Cálculo del Gini**: Se calcula el Gini para los ramas resultantes de la división.

2. Elección del Mejor Umbral de partición:

- **Comparación de la pureza**: Se comparan los Ginis de las posibles divisiones. El objetivo es minimizar la impureza (Gini < 0.5) en los nodos hijos.
- **Selección del umbral**: Se selecciona el umbral que produce la mayor reducción del Gini (reducción impureza). Esto significa que se selecciona el umbral que resulta en los nodos hijos más puros posibles.

Árbol de decisión: Algoritmo

3. División del Nodo:

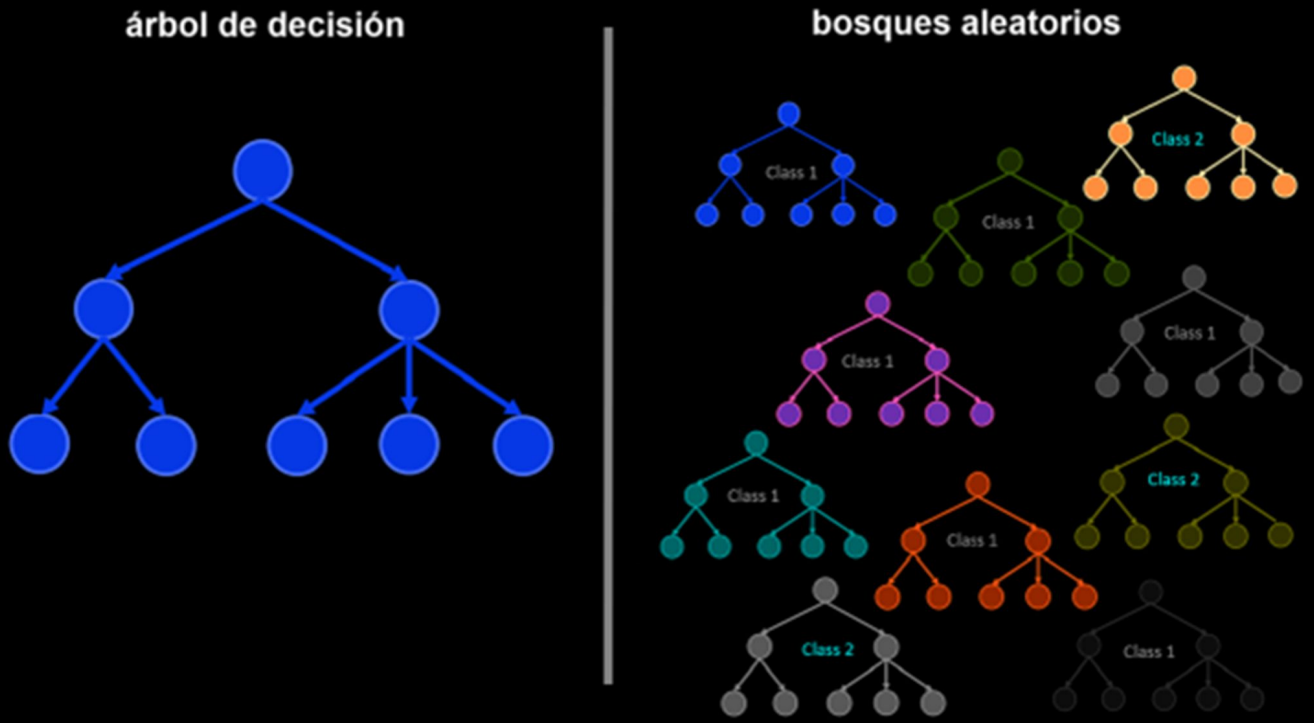
- Una vez seleccionado el umbral óptimo, el nodo se divide en dos ramas: una para las instancias que cumplen con la condición (es decir, donde el valor de la característica es menor o igual al umbral) y otra para las que no (donde el valor es mayor al umbral).

4. Repetición del Proceso:

- Este proceso se repite recursivamente para cada nodo hijo hasta que se cumpla un criterio de detención (por ejemplo, cuando todos los nodos son puros, o cuando no se puede reducir significativamente la impureza).

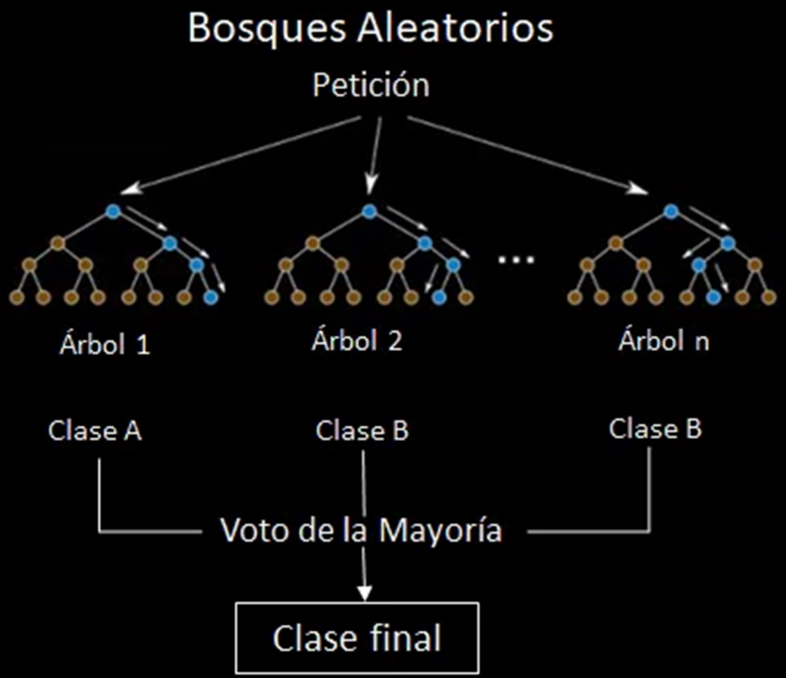
Bosques aleatorios

Los bosques aleatorios son algoritmos de aprendizaje supervisado que combina la salida de múltiples árboles de decisión.



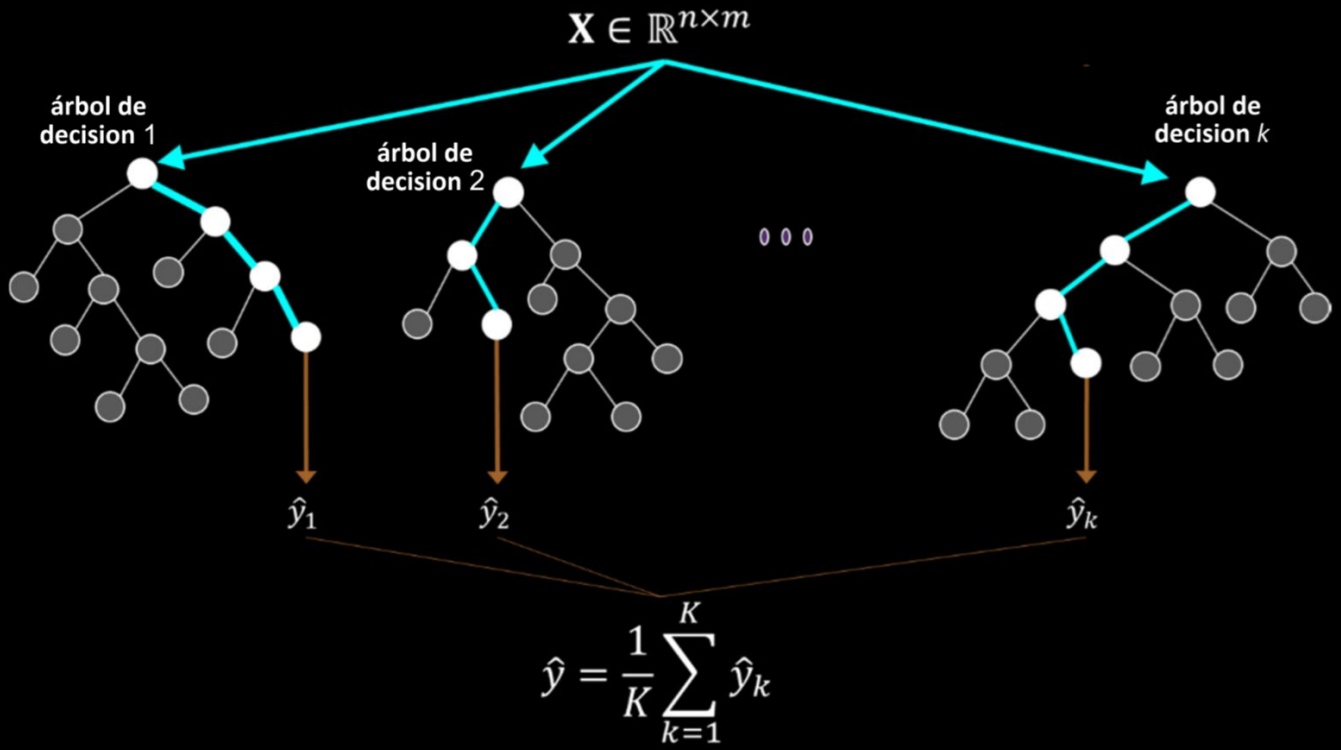
Bosques aleatorios

La clasificación final es el resultado de la agregación de las clasificaciones individuales de los árboles de decisión individuales:



Bosques aleatorios

Por ejemplo: en el caso de datos continuos, el resultado final del bosque aleatorio es el promedio de las predicciones individuales de cada árbol:



Bosques aleatorios

- Cada árbol en el bosque se entrena usando una **muestra diferente** de los datos original. Estas muestras se obtienen mediante muestreo con reemplazo.
- En cada árbol, se considera solamente un subconjunto aleatorio de características (features) para decidir la mejor división.

$$D = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$$

$$\{T_1, T_2, \ldots, T_B\} \quad \begin{matrix} D_b \subseteq D \\ \mathcal{X}_b \subseteq \mathcal{X} \end{matrix}$$

$$\hat{y}_b(x') = T_b(x') \quad T_b : \mathcal{X} \to \mathcal{Y}$$

$$\hat{y}(x') = \frac{1}{B} \sum_{b=1}^{B} \hat{y}_b(x')$$

$$\hat{y}(x') = \arg\max_{c \in \mathcal{Y}} \sum_{b=1}^{B} \mathbb{I}\left[\hat{y}_b(x') = c\right]$$

$$\mathcal{Y} = \{c_1, c_2, \ldots, c_k\}$$

$$\hat{y}(x') = \text{moda}\left(\hat{y}_1(x'), \hat{y}_2(x'), \ldots, \hat{y}_B(x')\right)$$

Bosques aleatorios

Ventajas

- Se aplica con inputs y outputs categóricos y continuos
- Es paralelizable
- Es robusto ante valores atípicos
- Funciona bien con datos desbalanceados
- Cuanto mayor es la cantidad de árboles, más preciso es el resultado, pero hay que tener cuidado con el sobreajuste
- Puede generalizar mejor debido al uso de múltiples árboles de decisión

Desventajas:

- Interpretabilidad: los modelos de bosque aleatorios no son fácilmente interpretables
- Para conjuntos de datos muy grandes, el tamaño de los árboles puede consumir mucha memoria.
- Puede tender al sobreajuste, por lo que es necesario ajustar los hiperparámetros.

Bosques aleatorios: medidas de evaluación

- El soporte para una clase C es simplemente el número de ejemplos en el conjunto de prueba que pertenecen a esa clase. Un soporte alto en una clase permite calcular métricas más robustas y confiables para esa clase.
- El Promedio Macro y el Promedio Ponderado son dos formas de promediar las métricas de rendimiento (como precisión, recall, F1-score) a través de todas las clases en un problema de clasificación.

$$\text{Soporte}(C_i) = N_i$$

$$\text{Promedio Ponderado} = \frac{\sum_{i=1}^{N} (\text{Soporte}_i \times \text{Métrica}_i)}{\sum_{i=1}^{N} \text{Soporte}_i}$$

$$\text{Promedio Macro} = \frac{1}{N} \sum_{i=1}^{N} \text{Métrica}_i$$

Bosques aleatorios: medidas de evaluación

- El Promedio Macro trata a todas las clases por igual, independientemente de cuántas instancias haya en cada clase. Es útil cuando cuando se busca que el modelo tenga un rendimiento equilibrado en todas las clases.
- El Promedio Ponderado da más importancia a las clases con más instancias. Es útil en clases desbalanceadas porque refleje el rendimiento en las clases más representadas cuando estas son de interés.

$$\text{Soporte}(C_i) = N_i$$

$$\text{Promedio Ponderado} = \frac{\sum_{i=1}^{N} (\text{Soporte}_i \times \text{Métrica}_i)}{\sum_{i=1}^{N} \text{Soporte}_i}$$

$$\text{Promedio Macro} = \frac{1}{N} \sum_{i=1}^{N} \text{Métrica}_i$$

Bosques aleatorios: cálculo de importancia de features

El cálculo de la importancia de las variables en un modelo de **Random Forests** se basa en la **reducción de la impureza** (Gini < 0.5) en los t-nodos de los árboles de decisión.

$$\text{Gini}(t) = 1 - \sum_{i=1}^{C} p_i^2$$

$$\Delta\text{Impureza} = \text{Impureza}(t) - \left(\frac{N_L}{N} \times \text{Impureza}(t_L) + \frac{N_R}{N} \times \text{Impureza}(t_R) \right)$$

$$\text{Importancia}(x_j) = \sum_{t \in \text{nodos donde } x_j \text{ es utilizado}} \Delta\text{Impureza}(t) \qquad \text{árbol } k$$

$$\text{Importancia}(x_j) = \frac{1}{T} \sum_{k=1}^{T} \text{Importancia}(x_j, \text{árbol } k)$$

Machine learning y deep learning: maquinas de soporte vectorial

Rolando Gonzales Martinez, PhD

Fellow postdoctoral Marie Skłodowska-Curie

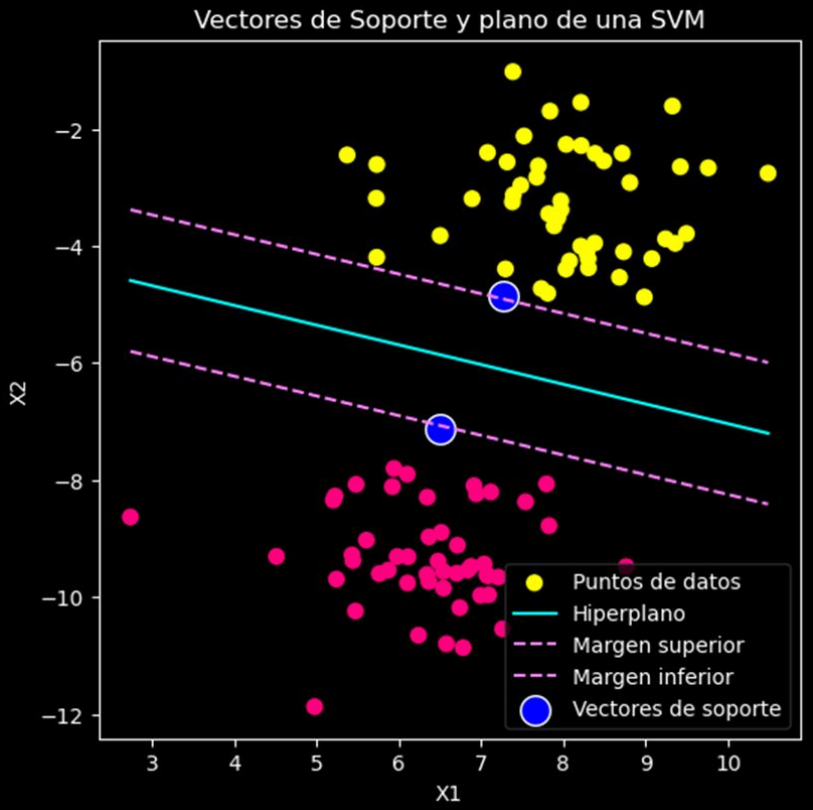
Universidad de Groningen (Países Bajos)

Investigador (researcher)

Iniciativa de Pobreza y Desarrollo Humano de la Universidad de Oxford (UK)

Máquinas de soporte vectorial

- En lugar de simplemente encontrar una línea que separa las clases, SVM busca la línea que no solo las separa, sino que lo hace de tal manera que la distancia desde la línea hasta los puntos más cercanos (de ambas clases) sea lo amplia posible.
- El margen es la banda alrededor del {hiper)plano dentro de la cual ningún punto de datos se encuentra (idealmente).
- Los vectores de soporte son los puntos en el margen que definen la posición del (hiper)plano.



Máquinas de soporte vectorial

Formalmente:

- El objetivo del SVM es maximizar el margen, sujeto a restricciones.
- Las restricciones se incorporan en la función objetivo mediante multiplicadores de Lagrange para n-datos, con condiciones Karush-Kuhn-Tucker
- La función objetivo tiene forma dual, sujeta a restricciones, y se resuelve con métodos numéricos.

$\{(\mathbf{x}_i, y_i)\}_{i=1}^n$, donde $\mathbf{x}_i \in \mathbb{R}^d \quad y_i \in \{-1, 1\}$

$\mathbf{w} \cdot \mathbf{x} + b = 0 \qquad \frac{1}{\|\mathbf{w}\|}$

$$\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2$$

$$y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1, \quad \text{para todo } i$$

$$L(\mathbf{w}, b, \alpha) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{n} \alpha_i \left[y_i(\mathbf{w} \cdot \mathbf{x}_i + b) - 1\right]$$

$$\frac{\partial L}{\partial \mathbf{w}} = \mathbf{w} - \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i = 0 \quad \Rightarrow \quad \mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i$$

$$\frac{\partial L}{\partial b} = -\sum_{i=1}^{n} \alpha_i y_i = 0 \quad \Rightarrow \quad \sum_{i=1}^{n} \alpha_i y_i = 0$$

$$\alpha_i \left[y_i(\mathbf{w} \cdot \mathbf{x}_i + b) - 1\right] = 0 \qquad \alpha_i \geq 0$$

$$\max_{\alpha} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j (\mathbf{x}_i \cdot \mathbf{x}_j)$$

$$\sum_{i=1}^{n} \alpha_i y_i = 0, \quad \text{y} \quad \alpha_i \geq 0$$

Máquinas de soporte vectorial

- Los valores de α definen cuáles de los puntos de datos se convierten en vectores de soporte, se utilizan para calcular el margen, el sesgo, y para clasificar los datos en clases.
- En problemas no linealmente separables, se utiliza el “truco de kernel”: se reemplaza una función kernel no-lineal en la función de optimización dual
- En problemas en que existe un cierto grado de superposición de clases, se agrega un término de penalización controlado por el hiperparámetro C.

$$\mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i \qquad \text{Márgen} = \frac{1}{\|\mathbf{w}\|}$$

$$b = y_i - \mathbf{w} \cdot \mathbf{x}_i$$

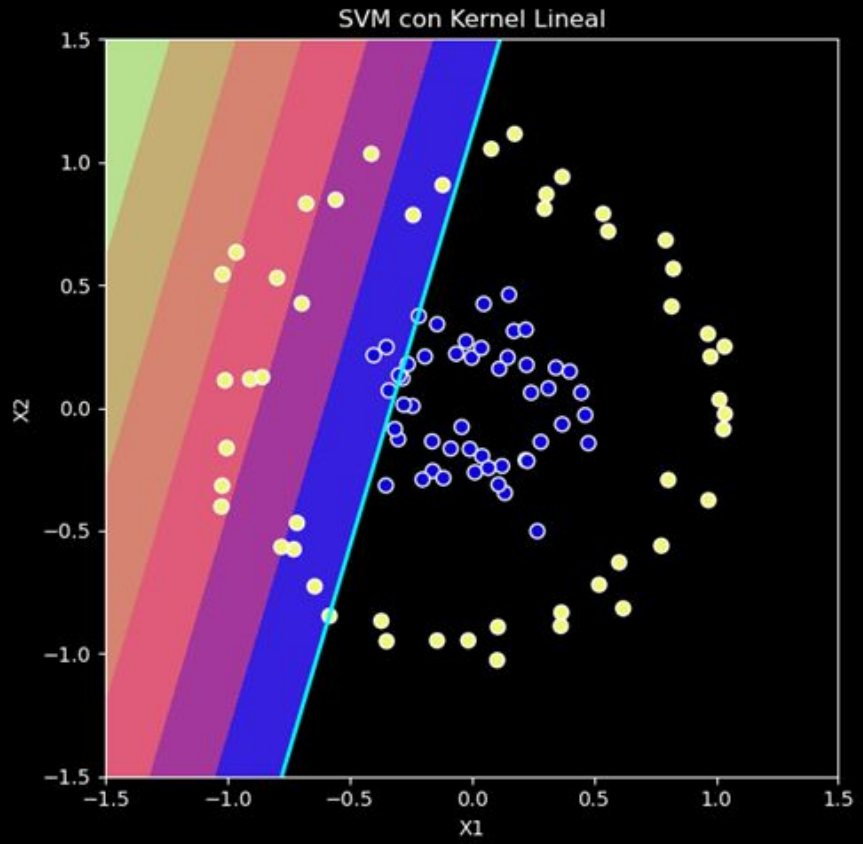
$$f(\mathbf{x}) = \operatorname{sign}\left(\sum_{i=1}^{n} \alpha_i y_i (\mathbf{x}_i \cdot \mathbf{x}) + b\right)$$

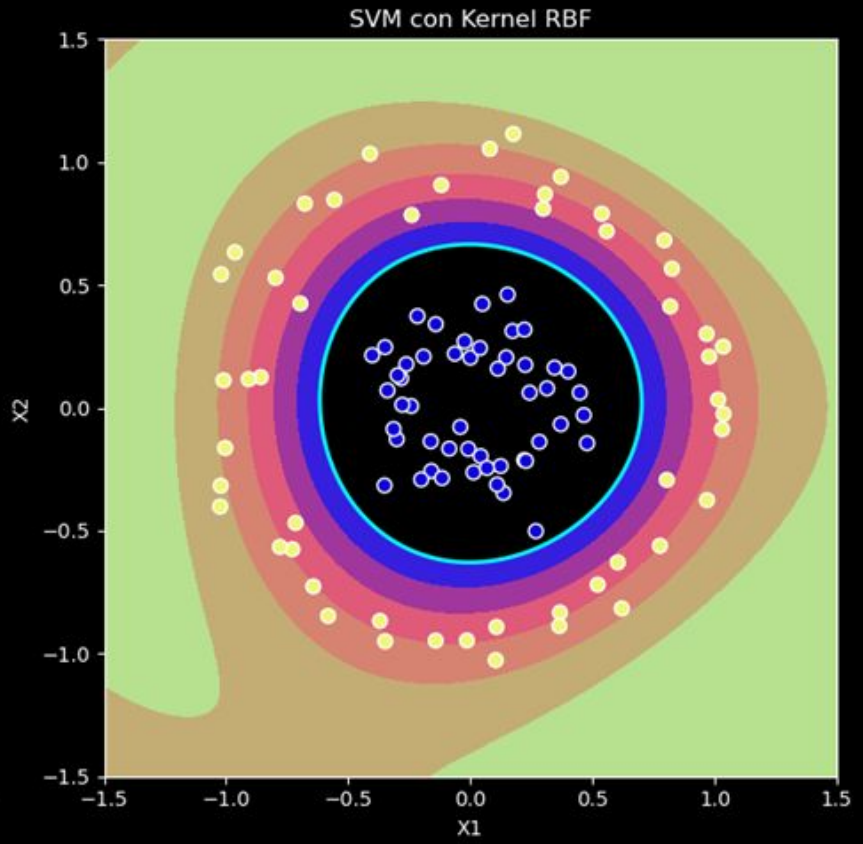
$$\max_{\alpha} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j$$

$$K(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\gamma \|\mathbf{x}_i - \mathbf{x}_j\|^2\right)$$

$$\min_{\mathbf{w},b,\xi} \frac{1}{2}\|\mathbf{w}\|^2 + C \sum_{i=1}^{n} \xi_i$$

Máquinas de soporte vectorial





Máquinas de soporte vectorial suave (soft SVM)

El hiperparámetro C controla el trade-off entre maximizar el margen y minimizar los errores de clasificación:

C alto:

- El margen será mas estrecho.
- Mejor clasificación
- Podría resultar en **sobreajuste**.

C bajo:

- El margen será más amplio.
- Algunos puntos de datos quedarán mal clasificados o dentro del margen
- Un C bajo introduce más **tolerancia a los errores** y puede mejorar la capacidad del modelo para generalizar a datos no vistos
- Podría resultar en un **subajuste** si C es demasiado bajo.

Machine learning y deep learning: Redes elásticas

Rolando Gonzales Martinez, PhD

Fellow postdoctoral Marie Skłodowska-Curie

Universidad de Groningen (Países Bajos)

Investigador (researcher)

Iniciativa de Pobreza y Desarrollo Humano de la Universidad de Oxford (UK)

Redes elásticas

- Elastic Nets son útiles en situaciones donde las variables predictoras están altamente correlacionadas o cuando el número de variables es mayor que el número de observaciones.
- La regularización Elastic Net busca encontrar una solución que minimice la función de pérdida con penalización combinada de L1 (Lasso) y L2 (Ridge).

$$\min_{\beta} \left\{ \frac{1}{2N} \sum_{i=1}^{N} \left(y_i - \sum_{j=1}^{p} X_{ij} \beta_j \right)^2 + \lambda_1 \sum_{j=1}^{p} |\beta_j| + \lambda_2 \sum_{j=1}^{p} \beta_j^2 \right\}$$

Regularización y redes elásticas

Lasso (Least Absolute Shrinkage and Selection Operator, L1), Ridge (L2), y Elastic Nets son técnicas de regularización que permiten reducir el sobreajuste y lidiar con multicolinealidad y variables redundantes:

$$\hat{\beta}_{\text{ridge}} = \arg\min_{\beta} \left(\sum_{i=1}^{m} (y_i - \hat{y}_i)^2 + \lambda \sum_{j=1}^{n} \beta_j^2 \right)$$

$$\hat{\beta}_{\text{lasso}} = \arg\min_{\beta} \left(\sum_{i=1}^{m} (y_i - \hat{y}_i)^2 + \lambda \sum_{j=1}^{n} |\beta_j| \right)$$

$$\hat{\beta}_{\text{elastic}} = \arg\min_{\beta} \left(\sum_{i=1}^{m} (y_i - \hat{y}_i)^2 + \lambda_1 \sum_{j=1}^{n} |\beta_j| + \lambda_2 \sum_{j=1}^{n} \beta_j^2 \right)$$

Redes elásticas Bayesianas

$$J(\beta) = \frac{1}{2N} \sum_{i=1}^{N} \left(y_i - \sum_{j=1}^{p} X_{ij} \beta_j \right)^2 + \lambda_1 \sum_{j=1}^{p} |\beta_j| + \lambda_2 \sum_{j=1}^{p} \beta_j^2$$

$$p(\beta_j \mid \lambda_1) = \frac{\lambda_1}{2} \exp(-\lambda_1 |\beta_j|)$$

$$p(\beta_j \mid \lambda_2) = \sqrt{\frac{\lambda_2}{2\pi}} \exp\left(-\frac{\lambda_2}{2} \beta_j^2 \right)$$

$$p(\beta_j \mid \lambda_1, \lambda_2) \propto \exp\left(-\lambda_1 |\beta_j| - \lambda_2 \beta_j^2 \right)$$

$$p(y \mid X, \beta, \sigma^2) = \prod_{i=1}^{N} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y_i - X_i \beta)^2}{2\sigma^2} \right)$$

$$p(\beta \mid X, y, \lambda_1, \lambda_2, \sigma^2) \propto p(y \mid X, \beta, \sigma^2) \times p(\beta \mid \lambda_1, \lambda_2)$$

$$p(\beta \mid X, y, \lambda_1, \lambda_2, \sigma^2) \propto \exp\left(-\frac{1}{2\sigma^2} \sum_{i=1}^{N} (y_i - X_i \beta)^2 - \lambda_1 \sum_{j=1}^{p} |\beta_j| - \lambda_2 \sum_{j=1}^{p} \beta_j^2 \right)$$

Ridge Bayesiano

$$y = \mathbf{X}\mathbf{w} + \epsilon$$

$$\hat{\mathbf{w}} = \underset{\mathbf{w}}{\operatorname{argmin}} \left\{ \|\mathbf{y} - \mathbf{X}\mathbf{w}\|^2 + \lambda \|\mathbf{w}\|^2 \right\}$$

$$\mathbf{w} \sim \mathcal{N}(\mathbf{0}, \lambda^{-1}\mathbf{I})$$

$$p(\mathbf{w}|\mathbf{X}, \mathbf{y}) \propto p(\mathbf{y}|\mathbf{X}, \mathbf{w}) p(\mathbf{w})$$

$$p(\mathbf{w}|\mathbf{X}, \mathbf{y}) \sim \mathcal{N}(\mathbf{w}_{\text{posterior}}, \mathbf{\Sigma}_{\text{posterior}})$$

$$\mathbf{w}_{\text{posterior}} = \mathbf{\Sigma}_{\text{posterior}}(\mathbf{X}^{T}\mathbf{y})$$

$$\mathbf{\Sigma}_{\text{posterior}} = (\lambda\mathbf{I} + \mathbf{X}^T\mathbf{X})^{-1}\sigma^2$$

$$p(y_*|\mathbf{x}_*, \mathbf{X}, \mathbf{y}) \sim \mathcal{N}(\mathbf{x}_*^T \mathbf{w}_{\text{posterior}}, \sigma^2 + \mathbf{x}_*^T \mathbf{\Sigma}_{\text{posterior}} \mathbf{x}_*)$$

$$\lambda = \frac{M}{\mathbf{w}^T\mathbf{w}} \qquad \hat{y} = X\mathbf{w}_{\text{posterior}}$$

Machine learning y deep learning: XGBoost

Rolando Gonzales Martinez, PhD

Fellow postdoctoral Marie Skłodowska-Curie

Universidad de Groningen (Países Bajos)

Investigador (researcher)

Iniciativa de Pobreza y Desarrollo Humano de la Universidad de Oxford (UK)

XGBoost (Extreme Gradient Boosting)

- XGBoost busca minimizar una función objetivo que combina la función de pérdida con un término de regularización en los t = 1,2,...,T árboles del modelo, para obtener estimaciones de Θ (parámetros e hiper parámetros como los pesos de las hojas y las penalizaciones)
- Una expansión de Taylor de segundo orden se emplea para aproximar la función de pérdida, utilizando un gradiente g (derivada parcial respecto a la predicción) y un hessiano (segunda derivada parcial respecto a la predicción):

$$\text{Obj}(\Theta) = \sum_{i=1}^{n} L(y_i, \hat{y}_i) + \sum_{t=1}^{T} \Omega(f_t)$$

$$L(y_i, \hat{y}_i^{(t)}) \approx L(y_i, \hat{y}_i^{(t-1)}) + g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i)$$

$$g_i = \frac{\partial L(y_i, \hat{y}_i^{(t-1)})}{\partial \hat{y}_i^{(t-1)}} \qquad h_i = \frac{\partial^2 L(y_i, \hat{y}_i^{(t-1)})}{\partial (\hat{y}_i^{(t-1)})^2}$$

$$\text{Obj}^{(t)} = \sum_{i=1}^{n} \left[g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i) \right] + \Omega(f_t)$$

XGBoost (Extreme Gradient Boosting)

- En la función de regularización T es el número de árboles, w es el peso asignado a las hojas, *γ* penaliza la profundidad del árbol y la penalización λ evita que las hojas crezcan demasiado
- Durante la construcción del árbol, cada nodo se divide de manera que maximice la ganancia.
- En las predicciones *η* es la tasa de aprendizaje que escala la contribución de cada nuevo árbol para evitar el sobreajuste.

$$\Omega(f_t) = \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} w_j^2$$

$$\text{Ganancia} = \frac{1}{2}\left[\frac{G_L^2}{H_L+\lambda} + \frac{G_R^2}{H_R+\lambda} - \frac{(G_L+G_R)^2}{H_L+H_R+\lambda}\right] - \gamma$$

$$\hat{y}_i^{(t)} = \hat{y}_i^{(t-1)} + \eta f_t(x_i)$$

Laboratorios

- AIMLDL_0301.ipynb: árboles de decisión y bosques aleatorios aplicados al análisis de enfermedades cardiacas.
- AIMLDL_0302.ipynb: bosques de aislamiento, local outlier factors y tratamiento de datos
- AIMLDL_0303.ipynb: máquinas de soporte vectorial, el truco kernel y máquinas de soporte vectorial suaves.
- AIMLDL_0304.ipynb: máquinas de soporte vectorial y máquinas de soporte vectorial suaves aplicadas al análisis de enfermedades cardiacas.
- AIMLDL_0305.ipynb: Estimación y comparación de múltiples modelos de machine learning: máquinas de soporte vectorial, bosques aleatorios, ridge, LASSO, redes elasticas, Bayesian ridge, XGBoost