Introducción

Data Scientist

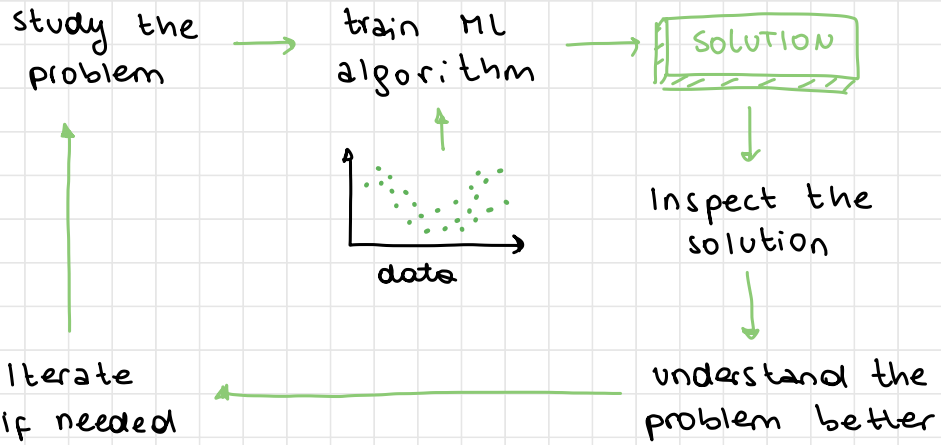
- obtener, interpretar, procesar y filtrar los datos
- llegar a conclusiones
- construir soluciones para los problemas que quiere solucionar

Data Scientist != Data Engineer

Data Scientist → interpretación y análisis de los datos mediante el análisis estadístico y matemático

Data Engineer → creación de la infraestructura y arquitectura para la generación, soporte y extracción de los datos (Big Data)

Machine Learning (Aprendizaje Automático)



Metodología

1° → Entender el problema

2° → Recolectar los datos

3° → Procesar los datos

4° → Explorar los datos

5° → Analizar los datos

6° → Comunicar los resultados

Visualización de Datos

Para ML

- Análisis inicial de los datos
  - ↳ examinar si satisfacen los supuestos requeridos
  - ↳ tienen complicaciones inesperadas (valores atípicos o no linealidad)
- Evaluar el ajuste del modelo
  - ↳ predicho vs observado
  - ↳ análisis de residuos

Plots

- distribución continua
- distribución discreta
- de relación
- series de tiempo

Numeros Útiles

Media: Promedio

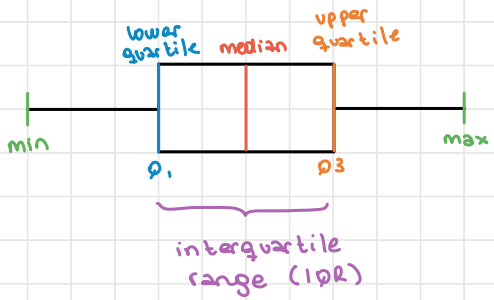
Mediana: valor que está en la mitad de la población

Cuartil: valor límite que deja al 25% de la población

Rango Intercuartílico: rango entre el cuartil 1 y 3

Box Plot

Muestra la distribución de los datos numéricos y la asimetría mediante la visualización de los cuartiles (o percentiles) y los promedios de los datos.



Scatter Plot (de dispersión)

Utiliza las coordenadas cartesianas para mostrar los valores de dos variables.

Correlación de Pearson: función que mide cuán relacionada están 2 variables de forma lineal.

- 0 → no existe relación
- 1 → relacionadas linealmente de forma perfecta
- -1 → correlación negativa perfecta

Heat map

Comparar distribuciones en donde ambos ejes son discretos y un tercero de "profundidad" numérico.

Violin Plots

Similar a los diagramas de caja excepto que también muestran la densidad de probabilidad de los datos en diferentes valores.

Introducción a la Ciencia de Datos

Variables

- Independientes

→ Cualitativas → Texto / Numérica (nominal / ordinal) → problema de clasificación

→ Cuantitativas → discreta / continua → problema de regresión

→ si no hay → problema de agrupamiento

Correlación de Variables

Dos variables están correlacionadas cuando varían de igual forma sistemáticamente.

Puede ser
- positiva
- negativa
- sin correlación

No implica causalidad.

Varianza

Promedio de la diferencia respecto de su media.

Covarianza

Grado de variación conjunta de 2 variables aleatorias respecto a sus medias.

Permite determinar si existe una dependencia entre ambas variables y estimar otros parámetros básicos como el coeficiante de correlación lineal o la recta de regresión.

Correlación de Pearson

Cuantifica la variación o dispersión de un conjunto de datos numéricos. Una desviación estándar baja indica que la mayor

parte de los datos de una muestra tienden a estar agrupados cerca de su media. Una desviación estándar alta indica que los datos se extienden sobre un rango de valores más amplio.

Métodos de Regresión

Regresión Lineal ó Ajuste Lineal:

- Ecuación Normal → Problemas: alto nivel computacional por la inversión de matrices.
- Métricas →
  - Root Mean Square Error (RMSE) ó Raíz del Error Cuadrático Medio
  - Mean Absolute Error (MAE) ó Error Medio Absoluto
  - Mean Square Error ó Error Cuadrático Medio

Métodos de Clasificación

Buscamos una categoría c de un conjunto C de categorías posibles. Estas categorías no sólo son finitas, sino que además son conocidas de antemano.

Regresión Logística

Dado una serie de puntos queremos encontrar una función que separe los puntos en dos conjuntos.

Una vez que la encontré puedo determinar, para cualquier valor x futuro, el conjunto al cual pertenecerá.

Métodos de Clusterización

Clustering: Agrupar los datos de tal forma que queden definidos N conjuntos distinguibles, aunque no necesariamente se sepa qué signifiquen esos conjuntos. El agrupamiento será por características similares.

K-Means:

- El usuario decide la cantidad de grupos
- K-Means decide al azar k centroides
- Decide qué grupos están más cerca de cada centroide
- K-Means recalcula los centroides al centro de el grupo
- K-Means vuelve a reasignar los puntos usando los nuevos centroides
- K-Means repite los 2 anteriores hasta que los

puntos no cambien de grupo

¿Cómo saber cuántos conjuntos elegir?

Regla del codo: Elegimos un rango y para cada valor, para cada centroide calculamos la distancia promedio

Método Silhouette: Elegimos un rango y para cada valor, para cada valor de k graficamos la silhouette. El mejor valor posible es 1. El peor valor posible es -1.

Estadística de Hopkins: Para evaluar la tendencia de agrupación de un conjunto de datos midiendo la probabilidad de que un conjunto de datos dado sea generado por una distribución de datos uniforme. Prueba la aleatoriedad espacial de los datos.

- **Hipótesis Nula:** el conjunto de datos se distribuye uniformemente (no hay clusters significativos)
- **Hipótesis Alternativa:** el conjunto de datos no está uniformemente distribuido (contiene clusters significativos)

Análisis de Valores Atípicos

Outlier: observación que se desvía tanto de las otras como para despertar sospechas que fue generado por un mecanismo diferente.

Tipos:
- Global Outlier
- Contextual Outlier
- Collective Outlier

Outlier Univariado

Valores atípicos que podemos encontrar en una simple variable. Son buenos para detección de extremos.

Outlier Multivariado

Valores atípicos que podemos encontrar en un espacio n-dimensional. Para detectarlos es necesario ajustar un modelo

Efectos:

- **Enmascaramiento**: un grupo de outliers esconden a otros. Se harán visibles cuando se eliminen los outliers que los esconden.

- **Inundación**: una observación sólo es outlier en presencia de otras observaciones si se quitaran las últimas, la primera dejaría de ser outlier.

Métodos Univariados

Z-Score: métrica que indica cuántas desviaciones estándar tiene una observación de la media muestral, asumiendo una distribución gaussiana fijamos un umbral de $Z > 3$.

Z-Score Modificado: la media de la muestra y la desviación estándar de la muestra, pueden verse afectados por los valores extremos presentes en los datos.

Análisis de Box-Plot: permiten visualizar valores extremos univariados. la regla de decisión suele ser: $\pm 1.5 * IQR$ (outliers moderados) y $\pm 3 * IQR$ (outliers severos).

Métodos Multivariados

Distancia de Mahalanobis: medida de distancia entre el punto x y el conjunto de observaciones con media μ y una matriz de covarianza S.

Lof - local Outlier factor: valora puntos en un conjunto de datos multivariados. Método basado en densidad que utiliza la búsqueda de vecinos más cercanos. Calcula los scores para cada punto y se debe definir un umbral de corte.

Isolation forest: Algoritmo no supervisado y no paramétrico basado en árboles de decisión. Los datos anómalos se pueden aislar los datos normales mediante particiones recursivas del conjunto de datos.

Limpieza de Datos

Datos faltantes

Missing completely at random (MCAR): la razón de la falta de datos es ajena a los datos mismos. No existen relaciones con la variable donde se encuentran los datos faltantes o con las restantes variables en el data set que expliquen por qué faltan.

Missing not at random (MNAR): la razón por la cual faltan los datos depende de los mismos datos recolectados.

Missing at random (MAR): punto intermedio entre los dos anteriores. La causa de los datos faltantes no depende de estos mismos datos faltantes pero puede estar relacionada con otras variables del dataset.

Estrategias

Eliminar registros ó variables: si la eliminación de un subconjunto disminuye significativamente la utilidad de los datos, la eliminación del caso puede no ser efectiva. No se recomienda en situaciones que no sean MCAR.

Imputar datos: Utilizar métodos de relleno de faltantes.

Imputación de datos

Sustitución de casos: se reemplaza con valores no observados. Debería ser realizado por un experto.

Sustitución por Media ó Mediana: se reemplaza utilizando la medida calculada de los valores presentes.

Desventajas:
- la varianza estimada de la nueva variable no es válida.
- se distorsiona la distribución.

- Las correlaciones que se observen estarán deprimidas debido a la repetición de un solo valor constante.

Cold Deck: selecciona valores o usa relaciones obtenidas de fuentes distintas de la base de datos actual.

Hot Deck: se reemplazan los faltantes con valores obtenidos de registros que son los más similares.

Por Regresión. el dato faltante es reemplazado con el valor predicho por un modelo de regresión.

Multivariate Imputation by Chad Equations (MICE): Trabaja bajo el supuesto de que el origen de los faltantes es MAR Es un proceso iterativo en el cual cada valor faltante de cada variable se predice en función de las variables restantes. 10 iteraciones suele ser suficiente.

Transformación de Datos

Técnicas

- Normalización
- Discretización
- lograr normalidad
- Imaginación

Normalización

se aplica sobre valores numéricos. Escala los features de manera que puedan ser mapeados a un rango más pequeño.

se utiliza cuando:

- las unidades de medida dificultan la comparación de features
- se quiere evitar que atributos con mayores magnitudes tengan pesos muy diferentes al resto

Normalización - Min Max:

Funciona al ver cuánto más grande es el valor actual del valor mínimo del feature y escala esta diferencia por el rango.

los valores van de 0 a 1.

Normalización Z-Score.

los valores para un atributo se normalizan en base a su media y desvío estándar.

Es útil cuando el verdadero mínimo y máximo del atributo no son conocidos, o cuando hay valores atípicos que dominan la normalización min-max.

Normalización Decimal Scaling:

Asegura que cada valor normalizado se encuentre entre -1 y 1.

Discretización

Permite dividir el rango de una variable continua en intervalos.

Se reducen los valores de una variable continua a un número reducido de etiquetas.

Discretización Binning:

Se divide a la variable en un número específico de bins.

Los criterios de agrupación pueden ser:

- Igual-Frecuencia. misma cantidad de observaciones en un bin
- Igual-Ancho. rangos o intervalos de clases para cada bin
- Cuantiles: separar en intervalos utilizando mediana, cuantiles y Percentiles.

Para cada uno de los agrupamientos podemos hacer:

→ reemplazo por media ó mediana

→ reemplazo por una etiqueta o valor entero

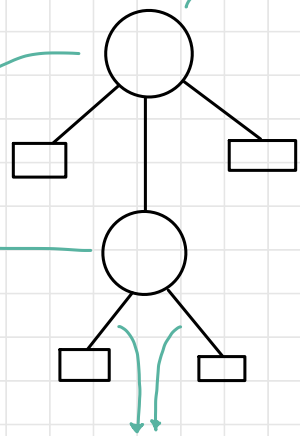
Variables Dummies - One Hot Encoding

Se puede recodificar la variable categórica en una o más dummies.

Árboles

ID3 : Iterative Dichotomiser 3

Genera un árbol de decisión a partir de un conjunto de ejemplos.



RAÍZ : nodo principal

TERMINAL : nodo donde termina el flujo pues no es raíz de ningún otro. contiene la respuesta (la clasificación a que pertenece el objeto que ha conducido hasta él).

representa preguntas con respecto al valor de uno de los atributos

posibles respuestas que los atributos pueden tomar

Entropía (de la información): Medida de la impureza o aleatoriedad en los datos.

- Para una muestra homogénea la entropía es igual a 0.
- la máxima entropía es 1 → máxima incertidumbre

→ $\text{Entropía}(S) = \sum_{i=1}^{n} -p_i \log_2 p_i$

- S: lista de valores posibles
- p_i: probabilidad de los valores
- i: cada uno de los valores

Ganancia de la Información: Se aplica para cuantificar qué característica, de un conjunto de datos, proporciona la máxima información sobre la clasificación.

$$\text{Gan Inf}(S,A) = \text{Entropia}(S) - \sum_{v \in V(A)} \underbrace{\frac{|S_v|}{|S|}}_{\text{probabilidad de un valor para el atributo A}} \text{Entropia}(S_v)$$

- S: lista de valores posibles para un atributo dado A
- $V(A)$: conjunto de valores que A puede tomar
- $\text{Entropia}(S_v)$: entropía calculada para el valor v de A

Algoritmo básico:

1) Calcular entropía para todas las clases
2) Calcular entropía para cada valor posible de cada atributo
3) Seleccionar el mejor atributo basado en la reducción de la entropía, usando el cálculo de ganancia de la información
4) Iterar, para cada sub-nodo, excluyendo el nodo raíz que ya fue usado

Impureza de Gini

Medida de cuán a menudo un elemento elegido aleatoriamente del conjunto sería etiquetado incorrectamente si fue etiquetado de manera aleatoria de acuerdo a la distribución de las etiquetas

en el subconjunto.

Algunas implementaciones de árboles de decisión (ej: scikit-learn) utilizan la impureza de Gini en lugar de la ganancia de la información ya que es más fácil de calcular.

→ $Gini = 1 - \sum_{i=1}^{c} (p_i)^2$

C4.5

se hicieron mejoras a ID3: campos numéricos, rangos continuos, datos faltantes y poda.

Datos faltantes: Permite valores de los atributos para ser marcados como "?" para faltantes. los valores faltantes de los atributos simplemente no se usan en los cálculos de la ganancia y la entropía.

Campos numéricos o rangos continuos: El algoritmo crea un campo booleano tal que si $A < C$, $A_C = TRUE$, si no $A_C = FALSE$.

El umbral (c) se encuentra cortando el rango de forma tal que C nos quede con la mayor ganancia de información:

- ordenamos A de menor a mayor

- identificamos los valores adyacentes (de la clase que es nuestra salida)
- detectamos cuando hay un cambio de valor de salida, entonces en esos límites seguramente están nuestros C_i candidatos
- creamos varios C_i que dividen en dos el rango Para cada uno de estos calculamos la ganancia de información
- nos quedamos con el | los que dan el mejor resultado

Poda: El método consiste en

- generar el árbol
- analizar recursivamente, y desde las hojas, qué preguntas (nodos interiores) se pueden eliminar sin que se incremente el error de clasificación con el conjunto de test

→ si hay ruido, el árbol tendrá un error

→ $\text{Error} = \dfrac{\text{casos bien clasificados}}{\text{casos totales}}$

- se elimina un nodo interior cuyos sucesores son todos nodos hoja

- se vuelve a calcular el error que se comete con este nuevo árbol sobre el conjunto de test
- si este error es menor que el anterior, se elimina el nodo y todos sus sucesores
- se repite

Random Forest

Bootstrap aggregating: Técnica o meta-algoritmo que dice:

→ dado un conjunto de entrenamiento D, de tamaño n, la técnica de bagging generará m nuevos conjuntos de entrenamiento D_1, D_i, D_m cada uno de tamaño n' tomando muestras aleatorias de D. En general $n' < n$ siendo $n' \cong \frac{2}{3} n$.

- Atribute bagging (or random subspace): Para cada una de las m tablas escogemos algunos atributos de forma aleatoria y nos quedamos con la raíz cuadrada del número de atributos de columnas.
- Una vez que tenemos m tablas reducidas en atributos y para cada una de ellas entrenamos un árbol: para cada árbol calculamos su matriz de confusión

- calculamos la tasa del error de cada árbol:

$$\frac{\text{FALSOS POSITIVOS + FALSOS NEGATIVOS}}{\text{TOTAL EJEMPLOS CLASIFICADOS}} = \text{Tasa de error}$$

- nos quedamos con el mejor árbol de los m y repetimos el proceso k veces
- al final vamos a tener k árboles

→ Clasificación: ejecutamos el caso que queremos probar por cada uno de los k árboles. si la mayoría de los casos devolvió que la categoría final es A entonces nos quedamos con ese valor.

Reducción de la dimensionalidad

- Visualización de datos para entender su distribución (detección de patrones inherentes a simple vista)
- Reducción del ruido
- Aceleración de los tiempos de entrenamiento de un modelo
- Compresión de la información
- Presentación de resultados a interesados

PCA: Análisis de Componentes Principales

Busca la recta que mejor ajusta maximizando la suma de las distancias al cuadrado, medidas desde los puntos proyectados sobre la recta hasta el origen de coordenadas, es decir: c^2.

→ suma distancias cuadradas de PC1 = autovalor de PC1

→ suma distancias cuadradas de PC2 = autovalor de PC2

→ variación de PC1 = $\frac{\text{autovalor de PC1}}{n-1}$

→ variación de PC2 = $\frac{\text{autovalor de PC2}}{n-1}$

- Scree plot: gráfico donde se representa el porcentaje de variación de cada PC (componente principal)
- Pasos:
  - centramos los datos en el eje de coordenadas
  - buscamos la recta que mejor ajuste que pase por el eje de coordenadas → obtenemos PC1
  - buscamos la recta que pasa por el origen y sea perpendicular a PC1 → obtenemos PC2

3 variables = 3 dimensiones: los pasos son los mismos sólo que en este caso la recta tiene 3 coordenadas.

1. centramos los datos en el eje de coordenadas
2. buscamos la recta que mejor ajuste que pase por el eje de coordenadas → obtenemos PC1
3. buscamos la recta que pase por el origen, sea perpendicular a PC1 y tenga el mejor ajuste maximizando su autovector → obtenemos PC2
4. buscamos la recta que pase por el origen y sea perpendicular a PC1 y PC2 → obtenemos PC3

Resumen:

- Nos sirve para identificar si hay agrupamiento de datos en el espacio de entrada
- Podemos identificar correlaciones, clusters o entender cuán dispersos están los datos y sobre qué ejes o variables
- Es útil cuando no podemos representar el espacio de entrada sobre un eje cartesiana

MDS: Multidimensional Scaling & PCoA: Principal Coordinate Analysis

Busca preservar las distancias entre puntos y ubicarlos en una

dimensión menor tal que las distancias se parezcan lo más posible.

- Pasos: ① Calculamos las distancias entre todos los valores

  ② Buscamos una matriz M con la misma cantidad de columnas pero menos filas (según las dimensiones que se quieran visualizar) usando la función Stress.

→ $$\text{Stress}_D(\vec{x}_1, \vec{x}_2, \dots, \vec{x}_N) = \sqrt{\sum_{\substack{i \neq j \\ 1 \dots N}} (d_{ij} - \|\vec{x}_i - \vec{x}_j\|)^2}$$

$\vec{x}_1, \vec{x}_2, \dots, \vec{x}_N$: cada uno de los vectores que representan a las columnas en el nuevo espacio

d_{ij}: distancia original calculada

$\|\vec{x}_i - \vec{x}_j\|$: nueva distancia que deberíamos calcular

si esto es 0 la distancia es la misma en ambos espacios. Por ello debemos minimizar este valor

Para minimizar la función:

★ Método de descenso empinado de Kruskal: descenso por gradiente

★ Método Iterativo de mayorización de De Leeuw, también llamado SMACOF

⇒ ambos son enfoques iterativos que pueden dar resultados ≠

- Tratar de formar clusters con la técnica de minimizar las distancias lineales entre los puntos es lo mismo que maximizar la correlación lineal

Fortalezas:

- soporta varios tipos de distancias
- permite transformaciones no lineales

Debilidades:

- optimización iterativa con mínimos locales
- difícil determinar qué distancia a usar es mejor

t-SNE: t-distributed stochastic neighbor embedding

Toma datos de un espacio de alta dimensión y los proyecta en un espacio de menor dimensión para que puedan ser representados.

- Determinar la similitud de todos los puntos en el gráfico
  a) tomamos el primer punto y medimos la distancia al siguiente
  b) proyectamos la distancia sobre una curva Normal centrada en el punto de interés
- El ancho de la curva normal (su desvío estándar) depende

de la densidad de los datos en la cercanía del punto de interés

- Cuando los anchos de las curvas no sean iguales hay que normalizar la similitud. puntaje (score) = similitud escalada = $\frac{\text{similitud}}{\sum \text{todas similitudes}}$

→ **Perplejidad**: parámetro que es igual a la densidad esperada para un punto. la desviación estándar se define por este valor que corresponde al número de vecinos alrededor de cada punto. Es establecido por el usuario y permite estimar la desviación estándar de las distribuciones gaussianas definidas para cada punto Cuanto mayor es, mayor es la variación.

- Una vez que calculamos todos los puntajes de similitud, armamos la matriz de similitud.
- Proyectamos los puntos del espacio en una recta de forma aleatoria.
- Calculamos el puntaje de similitud en el nuevo espacio. tomamos el primer punto, medimos la distancia al siguiente punto. luego proyectamos en una curva centrada

en el punto de interés y vemos su valor → distribución t-Student.

→ t-Student: permite una mejor visualización de los datos.

- Calculamos y normalizamos todos los nuevos puntajes y construimos una matriz (que no está ordenada)

Fortalezas:

- De lo mejor para visualizar datos
- Conserva estructuras no lineales globales y locales

Debilidades:

- Es estocástico (no determinista)
- Escala mucho en tiempo con dimensiones y puntos
- No se puede usar para nuevos puntos

ISOMAP

Permite aproximar la forma de los datos y calcular la distancia Geodésica.

- Línea Geodésica: línea de mínima longitud que une dos puntos en una superficie dada y está contenida en esta

superficie.

Para intentar aproximar la distancia Geodésica:

1. Construiremos un gráfico pesado

   a) Para cada punto en el espacio de entrada, tomamos k vecinos más cercanos (usando la distancia euclidiana)

   b) Trazamos aristas que conectan cada punto con sus vecinos más cercanos y ponderamos esas aristas según la distancia Euclidiana calculada.

2. Construimos una matriz de distancias entre todos los puntos, y usamos un algoritmo para encontrar la distancia más corta entre dos nodos en un grafo (Dijkstra ó Floyd-Warshall).

3. Usamos MDS con la matriz de distancia calculada.

Debilidades:

- Es lento, se nota a partir de las 1000 observaciones
- Puede crear una proyección errónea si k es demasiado grande con respecto a la estructura de la variedad o si hay ruido en los datos
- Si k es demasiado pequeño, el gráfico de vecinidad

puede volverse demasiado escaso para aproximar las trayectorias geodésicas con precisión

Landmark ISOMAP: se aplica en la matriz de distancias calculadas para encontrar una proyección euclidiana de todos los puntos en el conjunto original de datos cuando tenemos N datos especiales, siendo que N es menor que el número total de datos (sólo tomamos las distancias que involucran a los datos especiales).

KNN: k-nearest neighbors

- Método de Clasificación
- Sensible a conjuntos de datos no balanceados
- Sensible a outliers
- En scikit-learn $k=5$ por defecto

→ Usa distancias como la Euclídea, Manhattan, etc.

→ los vecinos pueden estar ponderados, siendo que los más cercanos tengan más peso en la votación y los más lejanos menos

SVM: Support Vector Machines

- **Margen**: distancia mínima entre cualquier observación y el umbral. Es el más grande posible ⇒ el clasificador es de Margen Máximo (Maximal Margin Classifier) → son súper sensibles a los outliers y pueden generar clasificaciones erróneas:

  → Higher bias: umbral poco sensible al conjunto de entrenamiento

  → low Variance: desempeño bueno con los nuevos datos

  → low bias: umbral muy sensible al conjunto de entrenamiento

  → High Variance: desempeño pobre con nuevos datos

- **Soft Margin**: distancia entre las observaciones y el umbral cuando permitimos clasificaciones erróneas. Es calculado por el algoritmo que usemos utilizando validación cruzada:
  - Parte el conjunto en 2 (70%-30% ó 10 subconjuntos)
  - Utiliza pares de observaciones similares para calcular la distancia media del umbral y validará la clasificación

con el segundo conjunto

- Define cuántas clasificaciones erróneas soportará en su margen blando y cuántas observaciones

→ Los problemas que no son linealmente separables no pueden ser clasificados por Maximal Margin Classifiers y Support Vector Classifiers.

SVM Kernel Polinómico

El algoritmo detrás de SVM usa funciones kernel que buscan clasificadores de tipo Support Vector Classifiers en dimensiones más altas:

- $d = 1$: para encontrar el Support Vector Classifier.
- $d = 2$: obtiene una segunda dimensión elevando al cuadrado las observaciones. Luego computa la relación entre cada par de observaciones y calcula el Support Vector Classifier
- $d = 3$: obtiene una segunda dimensión elevando al cubo las observaciones. Calcula el algoritmo nuevamente las relaciones entre todas las observaciones y busca un plano que parta el conjunto en 2.
- $d = n$ se repite el proceso agregando n dimensiones adicionales.

Resumen:

1. El kernel Polinómico de SVM incrementa las dimensiones seteando d, desde 1 hasta el valor del parámetro.
2. Calcula en cada caso las relaciones entre las observaciones para encontrar un hiper-plano que parta el conjunto en 2.
3. Usando validación cruzada, en cada caso calcula el error y se queda con el mejor d.

SVM kernel Radial

Soporta Support Vectors Classifiers en infinitas dimensiones.

Funciona como un Nearest Neighbor model ponderado: usa las observaciones más cercanas para clasificar la nueva observación.

SVM. The kernel Trick

Las kernel functions calculan la relación entre pares de observaciones como si estuviesen en otra dimensión, pero no realizan una verdadera transformación en el espacio.

Se reduce la cantidad de tiempo de cómputo necesario.

Permite cálculos de kernel Radial en infinitas dimensiones.

Ensambles de Modelos

- Entrenar varios modelos, c/u sobre datos distintos
- Cada modelo sobre-ajusta de manera diferente

bajo sesgo, alta varianza

Bias vs. Variance

Bias:

- El error debido al Bias de un modelo es la diferencia entre el valor esperado del estimador (predicción media del modelo) y el valor real.
- Bias muy alto: el modelo es muy simple y no se ha ajustado a los datos de entrenamiento, por lo que produce un error alto en todas las muestras (entrenamiento, validación y test).

underfitting

Variance:

- Es cuánto varía la predicción según los datos que utilicemos para el entrenamiento.
- Varianza baja: indica que cambiar los datos de entrenamiento produce cambios pequeños en la estimación.

- Varianza alta · pequeños cambios en el dataset conlleva a grandes cambios en el output (overfitting).

Bagging

Construir nuevos conjuntos de entrenamiento usando bootstrap (muestras aleatorias con reemplazo) para entrenar distintos modelos y luego combinarlos

1. Dividimos el conjunto de entrenamiento en distintos sub-conjuntos, obteniendo como resultado diferentes muestras aleatorias
   a) Las muestras son uniformes (misma cantidad de individuos)
   b) Son muestras con reemplazo (individuos pueden repetirse)
2. Entrenamos un modelo con cada subconjunto
3. Construimos un único modelo predictivo a partir de los anteriores

- Disminuye la varianza en el modelo final
- Muy efectivo en conjuntos de datos con varianza alta
- Puede reducir el overfitting
- Puede reducir el ruido de los outliers
- Puede mejorar levemente con el voto ponderado

Problemas:

- Si pocos atributos son predictores fuertes, todos los árboles se van a parecer entre sí
- Esos atributos terminarán cerca de la raíz, para todos los conjuntos generados con bootstrap

Bagging: Random Forest: igual a bagging tradicional, pero en cada nodo, considerar sólo un subconjunto de m atributos elegidos al azar.

Boosting

Alternativa a Bagging.

Buscar nuevos modelos para las instancias mal clasificadas por los anteriores.

1. Comenzar con un modelo (simple) entrenado sobre todos los datos: h_0
2. En cada iteración, entrenar h_i dando mayor importancia a los datos mal clasificados por las iteraciones anteriores.
3. Terminar al conseguir cierto cubrimiento o luego de un número de iteraciones.

④ Clasificar nuevas instancias usando una votación ponderada de todos los modelos construidos.

Modelos Exitosos:

- ★ AdaBoost
- ★ Gradient Boosting
- ★ XGBoost

XGBoost vs Gradient Boosting:

- La velocidad de entrenamiento de XGBoost es mucho menor por su implementación y por usar el GPU.
- El accuracy (métrica adecuada) también es mejor debido a que XGBoost maneja mejor el overfitting mediante regularizaciones.

AdaBoost

Entrenar un predictor, un clasificador base, verificar los errores que comete y entrenar luego otro predictor que corrija estos errores.

Se repite este proceso hasta disminuir el error o encontrar un clasificador perfecto.

Para verificar el error utiliza el mismo conjunto de entrenamiento. Antes de entrenar el siguiente predictor, pondera las

instancias mal clasificadas aumentando su peso relativo. Así, el siguiente predictor entrenado estará focalizado en corregir los errores del primero.

Contras.

- El entrenamiento no puede hacerse en paralelo ⇒ es poco escalable

- La implementación más común es con árboles.
- A diferencia de Random Forest (donde hay N árboles completos), en AdaBoost hay un bosque de tocones (stumps).
- stump. árboles con nodo raíz y 2 hojas No son buenos clasificando.

Resumen:

- Combina muchos weak learners (estimadores pobres) para hacer clasificaciones. Son, generalmente, stumps.
- Algunos stumps tienen más peso que otros en la votación final.
- Cada stump está construido teniendo en cuenta los errores del anterior. Usando una función de impureza de Gini ó simplemente regenerando los datos.

Gradient Boost

1. Crea una cadena de árboles de profundidad fija
2. Comienza con un sólo valor (nodo hoja) y luego calcula árboles para calcular el error cometido por el anterior.
3. Cada árbol está ponderado por un factor constante llamado learning rate (tasa de aprendizaje). Los árboles están restringidos en su crecimiento.

XGBoost

1. Predicción inicial: puede ser cualquier valor pero por defecto se toma 0,5. El error que comete el estimador para cada medición (distancia entre la predicción inicial y la medición) es el residuo.
2. Construir un árbol para los residuos: primero se crea un nodo hoja y se ponen allí todos los residuos.
3. Calcular el Similarity Score para los residuos. $SC = \frac{(\Sigma \text{ residuos})^2}{\#\text{ residuos} + \lambda}$

parámetro de REGULARIZACIÓN → determina qué tan severa es la penalización al método tradicional (Regresión lineal) → con λ reducimos el error por varianza

④ Calcular la ganancia total: para ver cuál será el siguiente nodo.

op. 1: tomar como umbral la distancia intermedia entre las primeras 2 observaciones.

op. 2: tomar como umbral la distancia intermedia entre las segundas 2 observaciones.

op. 3: tomar como umbral la distancia intermedia entre las últimas 2 observaciones.

Nos quedamos con la ganancia más alta.

⑤ Repetir el ④ hasta alcanzar la profundidad del árbol estipulada.

Por defecto XGBoost trabaja con 6 niveles.

⑥ Poda:

i) elegimos un número al azar (γ)

ii) calculamos la diferencia entre el gain del nodo más bajo y γ

iii) si la diferencia es < 0 removemos el nodo, si no el nodo se queda y terminó la poda

⑦ Repetir el paso ③ usando $\lambda = 1$.

- con $\lambda > 0$ los similarity scores son mucho más chicos

- esta disminución es proporcional a la cantidad de residuos en el nodo

8. Calcular el gain de cada nodo ($\lambda > 0$ hace que el gain sea menor)

9. Poda: repetir paso 5

- con $\lambda > 0$ es más probable tener que podar un árbol pues los Gain son menores
- con $\lambda = 0$ igual podemos tener que podar nodos pues el Gain puede ser negativo
- con $\lambda = 1$ prevenimos el sobreentrenamiento o sobreajuste del modelo en el conjunto de entrenamiento

10. Calcular respuesta: $\text{output} = \dfrac{\sum \text{residuos}}{\#\text{residuos} + \lambda}$, $\lambda = 0$ (valor por defecto)

11. Calcular todos los residuos (distancia entre predicción y output): si siguen siendo altos construimos un nuevo árbol y repetimos todo desde el paso 2

→ Repetimos todo hasta que los residuos sean prácticamente 0 ó alcancemos el número máximo de árboles.

Ensambles Híbridos

Combinan clasificadores de distinto tipo.

★ VOTING

Construir N modelos utilizando los mismos datos y luego tomar la predicción mayoritaria

datos nuevos → regresión logística → 1
datos nuevos → SVM → 0
datos nuevos → árbol de decisión → 1

→ 1

modelos ya entrenados

★ STACKING

- Entrenar diferentes modelos (base) y uno más que decide qué modelos usar
- Concepto de meta-aprendizaje para reemplazar el mecanismo del voto → se puede interpretar como mejora
- Pueden apilarse tantas capas de modelos como se desee

- los modelos para el meta-aprendizaje suelen ser: árboles, NB, SVM o Perceptrón
- Menos popular que boosting o bagging
  - → dificultad de análisis teórico (caja negra)
  - → múltiples variantes
- Si los clasificadores base pueden generar medidas de certeza, suele funcionar mejor

★ CASCADING

- Se pasa sucesivamente los datos de un modelo a otro
- Cada capa tiene un sólo modelo
- El input de cada modelo son las instancias predichas con poca certeza por el modelo anterior
- Suele utilizarse cuando se necesita una alta certeza en la predicción

① Entrenamiento

→ Entrenamos N modelos ≠ utilizando set de entrenamiento

→ Cada modelo lo entrenamos sobre las instancias predichas con baja certeza del modelo anterior

→ Se empieza por modelos simples y a medida que se entrenan nuevos los mismos son de mayor complejidad

② Predicción

→ El primer modelo recibe los datos de la transacción

→ Si la probabilidad de que no sea fraude es menor a 0.99, se pasa al siguiente modelo. Si no, se descarta la posibilidad de fraude

→ Si ningún modelo descarta el fraude, hay que confirmar a mano

- Enfoque "Multi-estado"
- Inicialmente creado para computer vision
- Cascadas muy profundas pueden producir overfitting

Resumen

- La combinación de clasicadores permite generar clasificadores más precisos a cambio de menor compresión

Ventajas

- Logran mejores predicciones
- Buen trade-off sesgo varianza

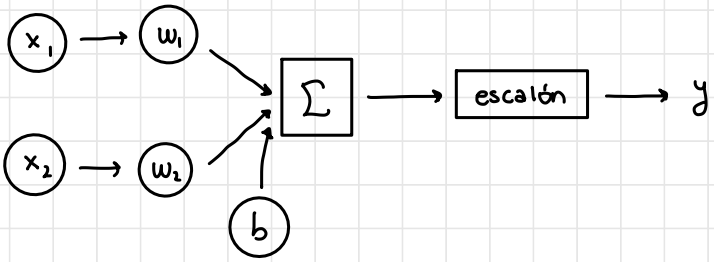
Desventajas

- Se pierde interpretabilidad
- Tienen una complejidad computacional mayor

Redes Neuronales

Perceptrón simple

★ **AND**



$$x_1 w_1 + x_2 w_2 + b \geq 1 \Rightarrow y = 1$$

$$x_1 w_1 + x_2 w_2 + b < 1 \Rightarrow y = 0$$

★ **CONJUNTO ETIQUETADO**

x_1	x_2	y
0	0	0
0	1	0
1	0	0
1	1	1

Datos de Entrenamiento

Etiquetas: valores esperados de salida

★ FUNCIÓN DEL ERROR

$$\text{Escalon}\ (w_1 + w_2 - 1) - 1 = \text{error}$$

$$\text{Escalon}\ (w_1 + w_2 - 1) = \begin{cases} w_1 + w_2 - 1 < 0 \Rightarrow 0 \\ w_1 + w_2 - 1 \leq 0 \Rightarrow 1 \end{cases}$$

★ ACTUALIZACIÓN DE LOS PESOS

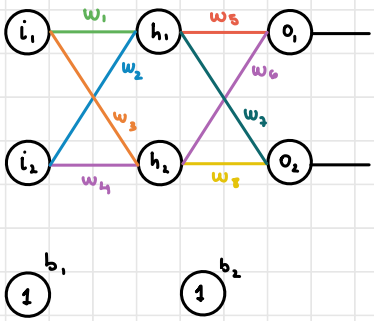
α: tasa de crecimiento, $\in [0,1]$, suele ser 0 5

$$\left.\begin{aligned} w_1' &= w_1 + \alpha * \text{erorr} \\ w_2' &= w_2 + \alpha * \text{error} \end{aligned}\right\}$$ cuanto más cerca de 0 esté el error más pequeña será la actualización

$b = -1$

Backpropagation

Perceptrón Multicapa (fully connected):



- i: inputs
- h: neuronas primera etapa
- o: neuronas de salida
- b_1 y b_2: bias (umbrales)

Función de activación: $f(x) = \dfrac{1}{1 + e^{-x}}$

① Calculo la entrada para h_1: $net_{h_1} = w_1 i_1 + w_2 i_2 + b_1 \cdot 1$

2. Calculo la salida para h_1: $out_{h_1} = f(net_{h_1})$

3. Calculo la entrada para h_2: $net_{h_2} = w_3 i_1 + w_4 i_2 + b_1 \cdot 1$

4. Calculo la salida para h_2: $out_{h_2} = f(net_{h_2})$

5. Calculo la salida para o_1: $out_{o_1} = f(w_5 h_1 + w_6 h_2 + b_2 \cdot 1)$

6. Calculo la salida para o_2: $out_{o_2} = f(w_7 h_1 + w_8 h_2 + b_2 \cdot 1)$

7. Calculo el error cuadrático medio de cada neurona de salida: $E = \sum \frac{1}{2} (\underbrace{target}_{\text{el valor que tenía inicialmente}} - output)^2$

8. Calculo el error total: $E_T = E_{o_1} + E_{o_2}$

→ Aplico Backpropagation

9. Calculo los nuevos valores de los pesos:

$$w_5' = w_5 - \alpha \frac{\partial E_T}{\partial w_5}, \quad \frac{\partial E_T}{\partial w_5} = \frac{\partial E_T}{\partial out_{o_1}} \frac{\partial out_{o_1}}{\partial net_{o_1}} \frac{\partial net_{o_1}}{\partial w_5}$$

y repito para w_6, w_7 y w_8

10. Calculo la capa anterior:

$$w_1' = w_1 - \alpha \frac{\partial E_T}{\partial w_1}, \quad \frac{\partial E_T}{\partial w_1} = \frac{\partial E_T}{\partial out_{h_1}} \frac{\partial out_{h_1}}{\partial net_{h_1}} \frac{\partial net_{h_1}}{\partial w_1}$$

y repito para w_2, w_3 y w_4

Implementación

Función de activación para la última capa con una clasificación multi clase

- Regresión → sigmoidea, lineal, ReLu, etc
- Clasificación de clases excluyentes → sigmoidea
- Clasificación de N clases simultáneas → Early stopping,

MÉTODOS DE REGULARIZACIÓN ← Regularización: L2, L1, Dropout, Data Argumentation

Funciones de Activación: Softmax

- Función exponencial normalizada
- Se utiliza para la capa de salida en modelos de clasificación
- según el modelo, se interpreta como scoring

Métodos de Regularización:

soluciona el overfitting que ocurre cuando una RN es muy compleja. Ayuda a que el modelo funcione en datos que nunca vio.

- **Regularización L1 y L2:**
  - Penalizan el valor de los pesos de la red para que no se le dé más relevancia a una característica que a otra

- Si es proporcional al módulo de los pesos se llama L1
- Si es proporcional al módulo cuadrado se llama L2

→ $\text{Loss} = \text{Error}(y, \hat{y}) + \lambda \sum_{i=1}^{N} |w_i|$ → Regularización L1

→ $\text{loss} = \text{Error}(y, \hat{y}) + \lambda \sum_{i=1}^{N} w_i^2$

- **Dropout.**
  - Evita codependencias en las conexiones de la red
  - El buen funcionamiento de la red no depende de unas pocas neuronas
- **Early Stopping:**
  - Evita el sobreajuste parando el entrenamiento antes de que el error del set validación empiece a aumentar
  - Busca quedarse con los pesos en la instancia óptima
- **Data Agumentation:**
  - Agrega datos usando los que se tienen y aplicarles transformaciones que los conviertan en nuevos datos, de manera que sean verosímiles
  - Útil cuando se trabaja con imágenes

Optimizadores: implementación concreta del algoritmo de backpropagation.

- **SGD: Stochastic Gradient Descent**

  Backpropagation simple.

- **Momentum:**

  Implementa backpropagation con pasos cada vez más rápidos

- **Nesterov.**

  Variante de Momentum, en vez de calcular el gradiente del error en el punto actual, lo calcula un poco más adelante (en la dirección del momento).

- **Adagrad:**

  Reduce el vector gradiente a lo largo de las dimensiones más empinadas.

  Cuanto más alto es el gradiente, más empinada es la función de error. Al suavizarla se evita caer en valles locales.

  ✓ Buen desempeño para problemas cuadráticos simples y tareas sencillas como regresión lineal

  ✗ Suele detenerse demasiado pronto cuando entrena RNs.

- **RMS Prop:**

Soluciona el principal problema de AdaGrad al ir olvidando las pendientes anteriores a medida que sigue avanzando (sólo acumula los gradientes de las iteraciones más recientes).

Suele ser mejor que AdaGrad y era el preferido antes de Adam.

- **Adam: Adaptive Moment Estimation**

Combina las ideas de Momentum y RMSProp.

Hace un seguimiento de una medida de decaimiento exponencial de gradientes pasados y de gradientes cuadrados pasados.

- **Ada Max:**

Modificación de Adam

En general, Adam da mejores resultados pero depende del conjunto de datos.

- **Nadam:**

Es Adam + Nesterov. Suele converger más rápido que Adam

- **Ada Delta:**

Variación de Ada Grad: en vez de calcular el escalado del factor de entrenamiento de cada dimensión teniendo en cuenta el gradiente acumulado desde el principio de la ejecución, se restringe a una ventana de tamaño fijo de los últimos n grandientes.

Similar a RMSProp que va olvidando gradientes.

Número de Capas.

- Un PMC con una sola capa oculta puede modelizar funciones complejas y suele ser suficiente
- Con problemas complejos, las redes profundas tendrán un mejor desempeño. Con 2 capas ocultas, para cientos de neuronas, se alcanza una cobertura del 98%.
- Para reconocimiento de imágenes o del discurso requieren decenas o cientos de capas, conectadas como en PMC.

Número de neuronas por capa: El número de neuronas de la capa de entrada y de salida está determinado por el

problema a resolver:

→ # datos = # neuronas_entrada

→ # salida_esperada = # neuronas_salida

Entrenamiento de la red:

arquitectura + hiperparámetros + optimizador + función de pérdida + funciones de activación