Sesión 5: Modelos de regresión y clasificación

ML-IA: Módulo de Machine learning

Jacob Muñoz-Castro

Universidad de los Andes

17 de noviembre de 2020

Agenda

1. Recap
2. SVM
3. Árboles de clasificación
4. Métodos de agrupación

Clasificación, kNN y LDA

Support Vector Machine

- SVM trata de encontrar un **hiperplano** en nuestro **espacio de atributos** que clasifique nuestros datos.
- ¿Qué es un espacio de atributos? Es un espacio p-dimensional donde los registros de los datos pueden ser mapeados (matriz de datos).
- ¿Qué es un hiperplano? Para un espacio de atributos p-dimensional, un hiperplano es un subespacio de $p-1$ dimensiones.
- En general, la ecuación de un hiperplano es:

$$\beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p = 0$$

- Entonces, ¿hay muchos hiperplanos posibles para un espacio de atributo? Sí

Hiperplanos separadores

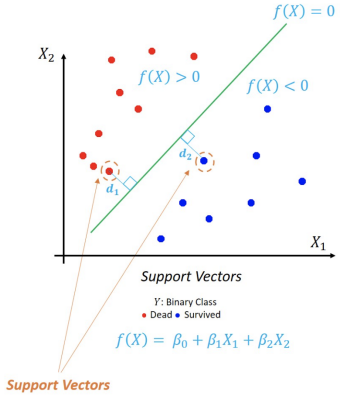
- Definamos la función $f(X)$

$$f(x) = \beta_0 + \beta_1 X_1 + \beta_2 X_2$$

- Para todos los hiperplanos, tenemos $f(X) > 0 (< 0)$ para todos los puntos por encima (debajo) del hiperplano.
- ¿Qué criterio usar para definir nuestro hiperplano separador?

Vector de soporte

- Los elementos más cercanos al hiperplano los llamaremos **vectores de soporte**.
- ¿Cómo medimos la distancia? Geométricamente sabemos que la mínima distancia de un punto a un hiperplano es el que se define por su vector ortogonal.

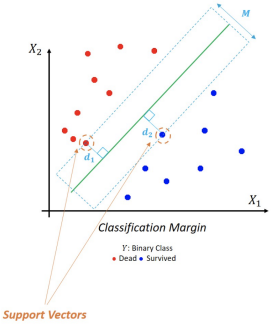


Margen de clasificación

- La distancia entre los vectores de soporte lo conocemos como **margen de clasificación** o margen. Es decir,

$$M = d_1 + d_2$$

- El margen define la separación o brecha entre nuestras clases.



Margen maximal

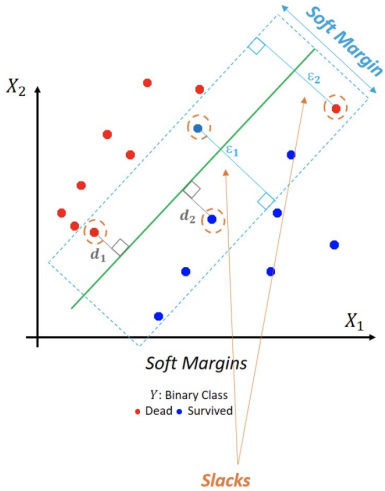
- Si nuestro objetivo es diferenciar lo que más podamos, entonces nuestro hiperplano debe ser el de mayor margen.
- A este hiperplano lo llamaremos **margen de clasificación maximal**.
- Entonces, el nombre vector de soporte reulta ahora más intuitivo: son los vectores que "soportan" el margen maximal.
- Es decir, si movemos los vectores de soporte, nuestro maximal debe moverse también.

Datos no separables

- ¿Qué pasa si nuestros datos no son separables por un hiperplano? Podemos observar superposición. También podemos tener muchas dimensiones (sparse data). En el mejor de los casos nuestros datos son separables pero ruidosos.
- Esto nos puede llevar a soluciones precarias para nuestro clasificador de margen maximal.
- ¿Solución?

Margen suave

- Podemos tratar nuestros datos no separables relajando los requerimientos del margen maximal.
- A esta relajación la llamaremos **slack**.
- Slack implica permitirle a algunos puntos de nuestros datos mantenerse en el lado equivocado de su margen.
- Nuestras variables slack ε_i permiten la clasificación errónea de ejemplos difíciles o ruidosos.

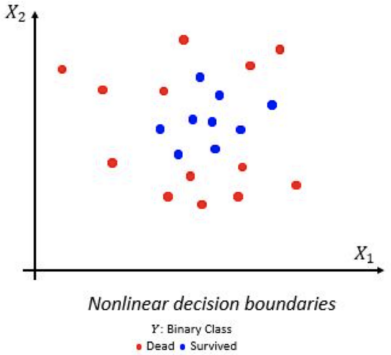


clasificadores de Vectores de soporte

- Nuestras variables slack cambian el problema de optimización
- Un parámetro de ajuste C se introduce y controla que tan amplio debe ser el margen suave comparado con el margen máximo.
- Entonces: Support vector classifier es el clasificador de margen máximo que optimiza el margen suave.

Decisión no lineal

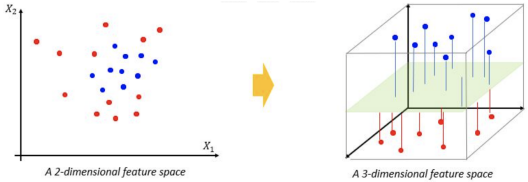
- Tenemos casos en los que la separación que queremos lograr no es lineal



- Nuestro clasificador de vectores de soporte intentará encontrar un límite lineal que mejor ajuste los datos, pero va a fallar independiente del valor de C que elija.

Expansión de atributos

- Una solución la vimos antes con regresiones polinomiales.
- Como les mencioné, el polinomio de un atributo se puede representar en un vector de tantas entradas como grados polinomiales tenga.
- En esto consiste la expansión de atributos: generar y por tanto expandir nuestro espacio de atributos de p dimensiones a M dimensiones ($M > p$) añadiendo $M - p$ transformaciones de nuestros p atributos iniciales.



Kernels y SVM

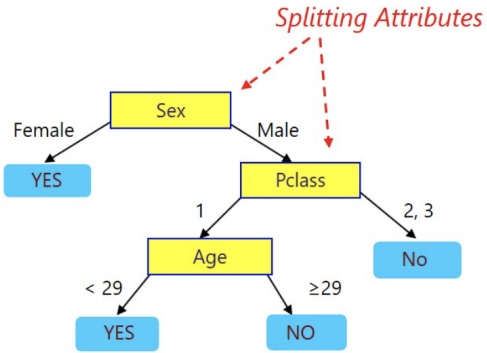
- Las transformaciones polinomiales pueden ser computacional-mente costosas.
- Si podemos computar productos internos entre observaciones, podemos ajustar un SVC
- Las funciones kernel pueden computar esto para polinomios d-dimensionales y por lo tanto agrandar el espacio.

$$K(x_i, x_i') = \left(1 + \sum_{j=1}^{p} x_i x_i'\right)^d$$

- **Conclusión:** SVM es una extensión de SVC que resulta de agrandar el espacio de variables (como p.e. funciones kernel).

Árboles de decisión

Pid	Sex	Age	Pclass	Survived
2	Female	38	1	Yes
3	Female	26	3	Yes
5	Male	35	3	No
7	Male	54	1	No
13	Male	20	3	No
14	Male	39	3	No
21	Male	35	2	No
24	Male	28	1	Yes
34	Male	66	1	No
54	Female	29	2	Yes



¿Cómo plantar árboles?

- Encontrar el árbol absolutamente óptimo es inviable.
- Nos concentramos en estrategias ambiciosas: encontrar el árbol del camino más corto. Es decir, dividir un nodo basado en pruebas de variables que optimiza un criterio para cada división.
- Preguntas que surgen:
  - ¿Qué atributo elige primero?
  - ¿Qué umbral de división elegir?
  - ¿Cómo determinar el mejor división?
  - ¿Cuándo paramos de dividir?

Especificación de la condición de prueba

- Tipos de atributos: nominal, ordinal, continuo
- Orden de división: binaria o múltiple.
- *Pensemos nominla y ordinal*
- *Ahora pensemos continuo...¿cómo hacemos divisiones con variables continuas?*
- R: Discretización! Esta puede ser estática o dinámica, equal interval bucketing, equal frecuency bucketing, sweep, ... pero lo realmente importante es entender cuál es el mejor split!

¿Cuál es el mejor split?

- Tenemos dos clases C_1 (morir) y C_2 (sobrevivir).
- Tenemos 10 registros en cada clase
- Diferentes condiciones de prueba son:
  - Edad < 23: $C^S = (6, 4)$ y $C^{No} = (4, 6)$
  - Clase: $C^1 = (1, 3)$, $C^2 = (8, 0)$ y $C^3 = (1, 7)$
  - Identificación (NIT): $C^{NIT1} = (0, 1)$, $C^{NIT2} = (1, 0)$, ..., $C^{NIT20} = (0, 1)$

¿Cuál es el mejor split?

- Estrategia ambiciosa: distribución homogénea de clases es preferido. ¿Qué hacemos?
- Para esto necesitamos una medida de impureza de nodo
- P.E.: $C = (5, 5)$ vs $C = (9, 1)$
- Medidas de impureza?

Medidas de impureza

- Índice de Gini

$$GINI(t) = 1 - \sum_j p(j|t)^2$$

- Entropía

$$H(t) = -\sum_j p(j|t) \log_2 p(j|t)$$

- Error de clasificación errónea

$$E(t) = 1 - \text{máx}(p(j|t)$$

donde $p(j|t)$ es la frecuencia relativa de la clase j en el nodo t. Máximo cuando los registros se distribuyen igual en todas las clases (split menos útil).

Ganancia de información o reducción de impureza

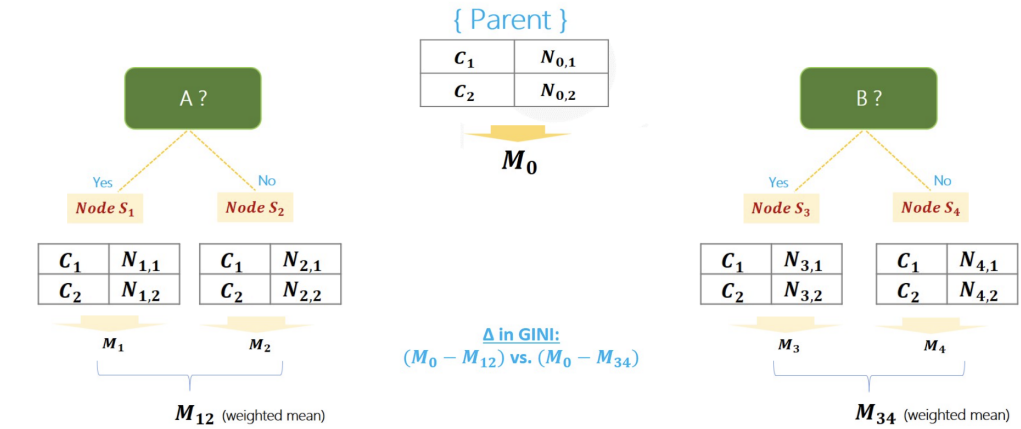
Cuando un nodo p se divide en k particiones (ramas o hijas), la calidad de la división d es:

- $$G_d^{GINI}(p) = \sum_{i=1}^{k} \frac{n_i}{n} GINI(i)$$
- $$G_d^{H}(p) = H(p) - \sum_{i=1}^{k} \frac{n_i}{n} H(i)$$
  - ¿Qué identificamos? Prefiere splits con muchas particiones! ¿Cómo corregir?
  - Information Gain Ratio

$$g(p) = \frac{G(p)}{I(p)}, \; I(p) = -\sum_{i=1}^{k} \frac{n_i}{n} \log \frac{n_i}{n}$$

donde I es la información intrínseca de la división

Ganancia de información o reducción de impureza



Criterio de suspensión

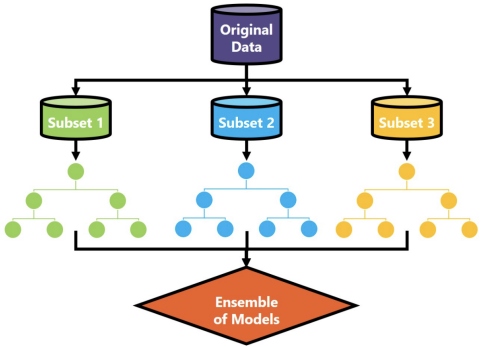
- Todos los registros pertenecen a una clase
- Todos los registros tienen valores de atributos similares
- Terminación fija o podado (como un árbol)
  - Número de niveles del árbol
  - Número de nodos-hojas
  - Número mínimo de registros por hoja

Pros y contras

- Pros
  - Intuitivo
  - Fácil interpretación de árboles pequeños
  - No paramétrico (incorpora atributos numéricos y categóricos, nos deja de importar la colinealidad)
  - Rápido
  - Robusto a outliers
- Contras
  - Sobreajuste: ajustar con cuidado, elección de parámetros es crítico
  - Clasificación rectangular (parece más primitivo que los problemas lineales ya vistos)

Métodos de agrupación

- Mejora el desempeño del modelo al computar múltiples modelos
- Agrupación se le puede aplicar a cualquier algoritmo de aprendizaje (incluyendo ambos regresión y clasificación)



Distribución binomial

- Consideremos un lanzamiento de moneda 3 veces, cada una independiente entre ellas, y justa (50/50).
- Sabemos de la distribución binomial que es una distribución conocida, intuitiva, no podemos decir mucho del comportamiento de los resultados individuales, pero sí podemos decir y predecir el comportamiento del agregado!

$$f(k; n, p) = P(X = k) = \binom{n}{k} p^k (1-p)^{n-k}$$

Plug in

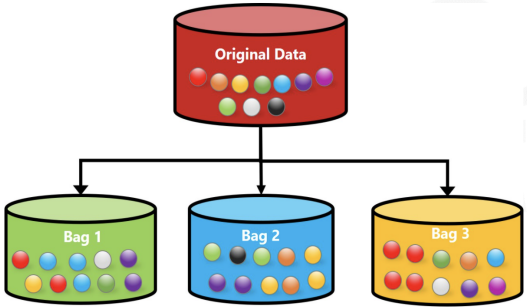
- ¿Cómo unimos esta idea con los métodos de agrupación?
- Suponga que tenemos 25 clasificadores (modelos)
- Cada clasificador tiene error de $\varepsilon = 0{,}35$
- Suponga que los clasificadores son independientes
- Entonces, la probabilidad de que nuestro clasificador agrupado haga un error de predicción es (equivalente a que la mayoría de modelos se equivoquen):

$$\sum_{i=13}^{25} \binom{25}{i} \varepsilon^i (1-\varepsilon)^{25-i}$$

- Hoy veremos dos métodos: bagging (todos los clasificadores son creados iguales) y boosting (no todos los clasificadores son creados iguales)

Bagging

- Muestreos con reemplazos
- Cada bolsa (bag) contiene variaciones de los datos originales
- Esto resulta en diferentes clasificadores



Bagging

- Con esto reducimos varianza en la estimación
- Previene el sobreajuste que tanto tememos
- Es robusto a outliers

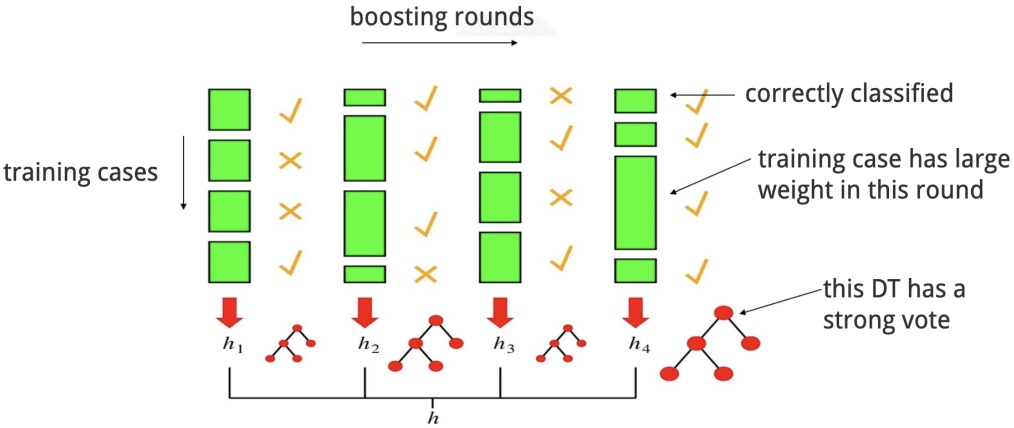
Random Forest

- Ahora pensemos en los árboles
- Si pensamos en el concepto de agrupación, un bosque aleaotrio (random forest) tiene mucho sentido!
- ¿Cómo funciona?
  - Diferentes subconjuntos de datos se seleccionan con reemplazo para entrenar árboles
  - Lo que queda fuera de la bolsa (OOB) es usado para estimar el error y la importancia de variable
  - La asignación de clase se hace por el número de votos de cada árbol. Para regresión, se usa el promedio

Random Forest

- Se comparan los ginis de cada split y se elige la varible que tenga la mayor reducción de gini.
- Se elige un subconjunto de variables que van a hacer los demás splits. Este número lo elige el usuario.
- Reglas de dedo:
  - Dado: N registros, M variables y m variables elegidas aleatoriamente para cada nodo,
  - se hace un muestreo con reemplazos N veces para construir la muestra de entrenamiento de cada árbol
  - $m < M$: para clasificación $m = \sqrt{M}$ y para regresión $m = M/3$

Boosting


boosting rounds
training cases
correctly classified
training case has large weight in this round
this DT has a strong vote
h_1
h_2
h_3
h_4
h

Boosting

- Es un proceso interactivo para cambiar adaptativamente las distribución de los datos de entrenamiento enfocándose en las clasificaciones erróneas previas
- Inicialmente todos los registros tienen el mismo peso
- Diferente a bagging, los pesos pueden cambiar al final de la ronda de boosting.
- Los registros mal (bien) clasificados tendrán el peso incrementado (reducido)
- Las clasificaciones erróneas serán más probables en salir en rondas subsecuentes.
- **Intuición:** para un examen uno estudia el tema en el que se siente más crudo
- Por último, los clasificadores con mejor error de predicción son los que pesan más en la votación