FAMAF Facultad de Matemática, Astronomía, Física y Computación
UNC Universidad Nacional de Córdoba

Aprendizaje Supervisado 2024

Docentes:

- **Dra. Ing. Karim Nemer Pelliza**
- **Dr. Ing. Diego González Dondo**

Tercera Clase

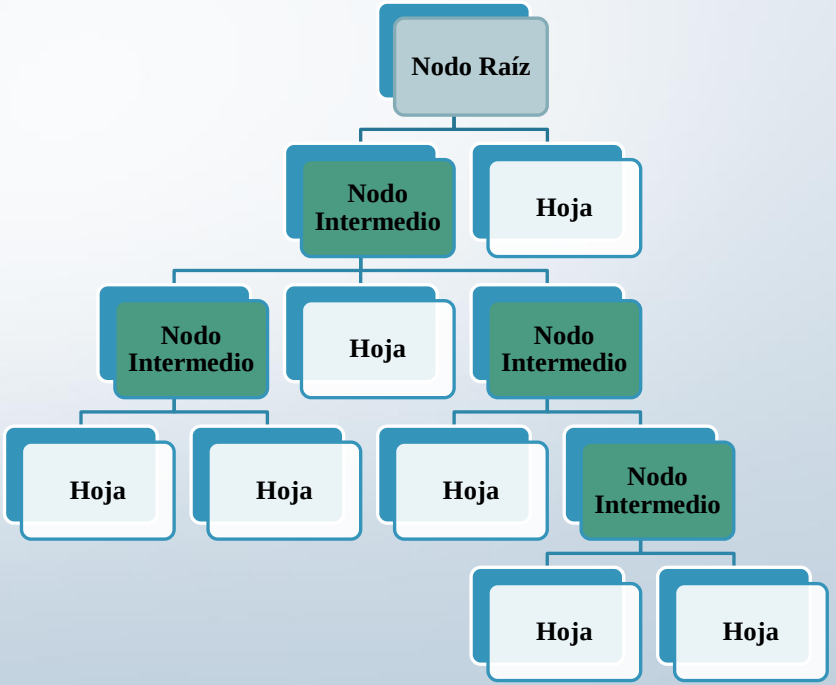
Contenido de la Tercera Clase

- ✓ Introducción al ML
- ✓ Etapas en la aplicación del ML
- ✓ Repaso:
  - ✓ Regresión Lineal y Polinomial,
  - ✓ Regresión Logística
  - ✓ Perceptrón.
- ✓ Support Vector Machines.
- ✓ SVC/SVR.
  - ✓ Datos no linealmente separables.
  - ✓ Función de costo.
- ✓ Redes neuronales multicapa
- ✓ Naive Bayes
- ✓ Repaso:
  - ✓ Árboles de decisión
- ✓ Ensemble learning.
  - ✓ Random Forest,
  - ✓ Bagging, Boosting y Voting.
- ✓ Sistemas de recomendación.
  - ✓ Filtrado colaborativo.
- ✓ “Buenas” prácticas en la aplicación de ML

Árboles de decisión

Árboles de decisión

- Aprende a diferenciar los datos en base a reglas de decisión.
- Los nodos del árbol representan las reglas. Las hojas asignan la clase o el valor.
- El árbol se particiona recursivamente. Para obtener el resultado, simplemente se siguen los nodos de decisión de acuerdo a los datos y se asigna la clase final.
- Es un algoritmo de “caja blanca”, ya que puede visualizarse fácilmente e interpretarse por los humanos (a diferencia de un algoritmo de “caja negra” como son las redes neuronales).
- Son buenos con datos de mucha dimensionalidad (high dimensional data)



Construyendo árboles de decisión

- Se selecciona la característica “más predictivo”
- Se calculan las “impurezas” de los nodos mediante la “entropía”
  - $Entropía(S) = \sum_{i=1}^{n} -p_i \cdot \log_2(p_i)$ donde p_i es la proporción de una categoría
  - Se calcula la entropía de las características
  - Se compara la entropía del nodo actual con todas las entropías posibles siguientes y se selecciona la de mayor diferencia
- Repetir con cada nodo hijo.
- El algoritmo se detiene si:
  - Todos los ejemplos del subconjunto son de la misma clase.
  - Todos los elementos del subconjunto son constantes con respecto al atributo/s de interés del nodo actual.

Árboles de decisión: Métodos de decisión

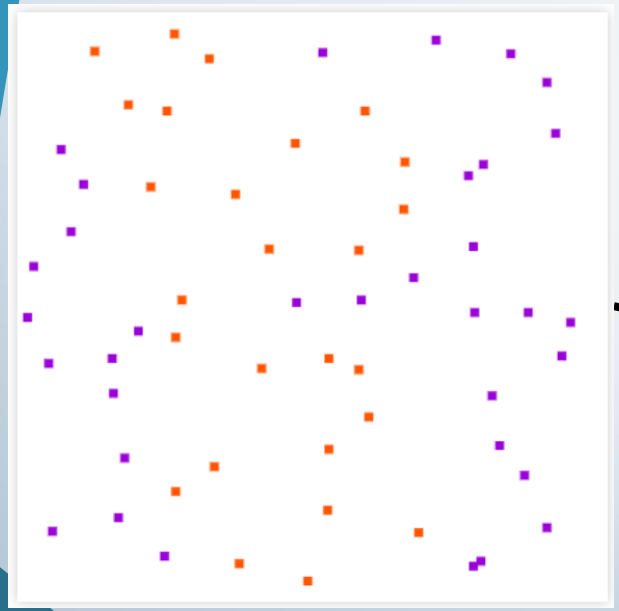
1. Definen la forma de particionar el conjunto de datos (es una heurística).
2. Buscan rankear cada atributo de acuerdo a ciertos parámetros.
3. Los criterios más populares son:

Information Gain: calcula la diferencia entre la entropía del conjunto antes de dividirse y la entropía media luego de dividir el conjunto de datos, para cada atributo.

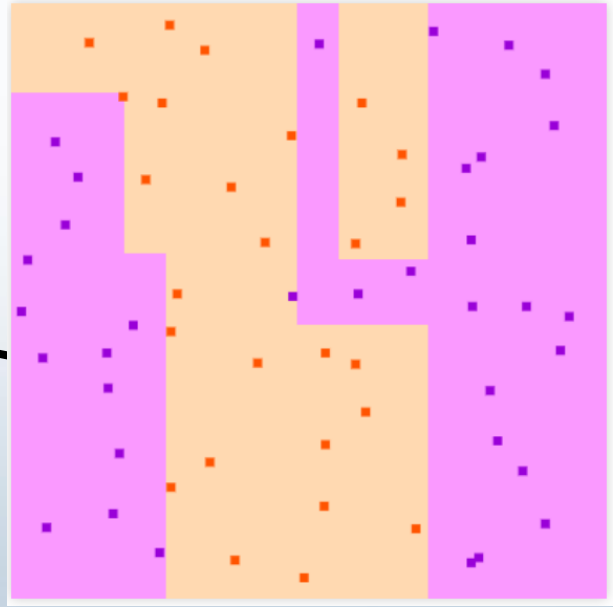
Gini Index Es una métrica que mide cuánto un elemento del subconjunto elegido al azar sería identificado incorrectamente. Busca los atributos que tienen menor índice de Gini.

Mean Squared Error Este método sirve para casos de regresión (en lugar de clasificación). Busca minimizar el error cuadrático medio en los nodos hijos a la hora de particionar los datos.

Árboles de decisión: Cómo son las fronteras



"Frontera de Decisión"



Link

Imagen de Yisong Yue

Demo Time (demo_8_trees)

Ensemble Learning

Ensemble Learning: Motivación

Se basan en la idea de que el trabajo en conjunto debería dar mejores resultados.

De tal forma, un conjunto de modelos, al combinarse, deberían tener mejor performance.

Consideremos el caso de tres modelos: M1, M2 y M3:

Cuando las predicciones son iguales, es sencillo definir cómo predecir; qué hacemos en el caso en que difieran?

- **Le creemos al mejor de los modelos?**
- **Votamos por mayoría?**

Ensemble Learning

Un modelo "***ensemble***" se constituye como un conjunto de diferentes modelos.

Habitualmente, un modelo "***ensemble***" es más preciso que los modelos que lo constituyen. Intuitivamente, esto se debe a que "dos aprenden mejor que uno" (Sinergia).

Ensemble Learning: Aproximación de justificación

Asumimos que tenemos un modelo M, formado por n modelos: M_1, M_2,... M_n.

Cuando un modelo recibe un dato *x*, el modelo predice M(*x*) a partir de las predicciones $M_i(x)$, a partir de votación (clase más votada).

¿Cómo determinamos cuán bien funciona el modelo?

- **Todos los clasificadores M_1, M_2,... M_n son igualmente precisos (precisión p)**
- **Los errores en la clasificación hecha por cada clasificador son independientes: $P(M_j \text{ erróneo} \mid M_k \text{ erróneo}) = P(M_j \text{ erróneo})$**

Ensemble Learning: Aproximación de justificación

Para hacer el ejemplo más concreto, consideremos:

- **p** = 0.8 (probabilidad de acertar de todos los modelos)
- **n** = 5 (cantidad de modelos)

Entonces, (aplicando el cálculo de la binomial)

$$Pr(x \geq k) = \sum_{i=k}^{n} \binom{n}{i} P^i (1-P)^{n-i}$$

Pr (la predicción de M es correcta) =

= Pr(al menos 3 de los 5 M_i predicen correctamente)

$$= \binom{5}{3} 0{,}8^3\ 0{,}2^2 + \binom{5}{4} 0{,}8^4 0{,}2^1 + \binom{5}{5} 0{,}8^5\ 0{,}2^0 = 0{,}942$$

Ensemble Learning: Bagging

Bagging

Es un método para hacer aprendizaje por "ensemble".

Si usamos el mismo modelo sobre los mismos datos, obtendremos los mismos resultados (*salvo inicializaciones aleatorias*). Entonces, de cierta forma, debemos introducir variaciones en los datos:

Si D es el dataset inicial; repetimos k veces lo siguiente:

- Generar D_i a partir de las entradas de D, seleccionando aleatoriamente y con reposición |D| instancias de D.
- Entrenamos el modelo M_i a partir de D_i.

Nuestro modelo M selecciona las predicciones más frecuentes de $\{M_i\}_i$.

Bagging gráficamente (para árboles de decisión)

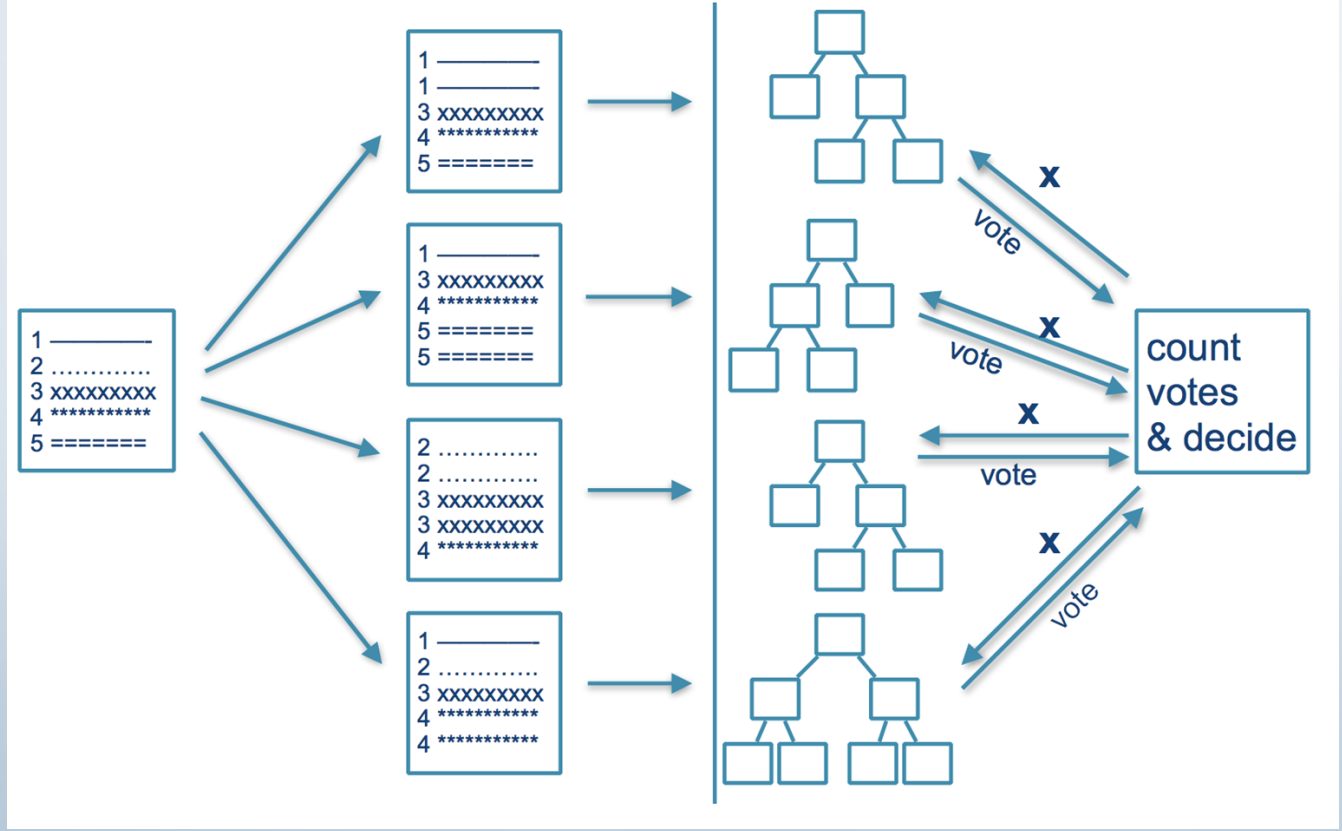


Imagen de Hendrik Blockeel

Bagging para árboles de decisión

Bagging generalmente funciona bien para algoritmos "inestables":

- *Un algoritmo es inestable si pequeñas variaciones en el dataset pueden generar modelos muy diferentes.*

Resulta que los árboles de decisión son inestables.

Si bien lo anterior incentiva al uso de bagging para árboles de decisión, tenemos una desventaja importante: entrenar ***k*** árboles es ***k*** veces más caro que entrenar uno solo.

Random Forests

Random Forests

Las Random Forests son una modificación a Bagging para Árboles de Decisión. Para evitar la sobrecarga, se simplifican los modelos:

Como vimos antes, cuando se crean los árboles (cuando se entrenan), todas las características son consideradas o evaluadas al crear cada nodo.

Para los Random Forests, en cada nodo se consideran sólo M atributos elegidos aleatoriamente (parámetro *max_features* en sklearn).

Usualmente, se toma:

- $M = \sqrt{número\ de\ atributos}$

Random Forests

"Los Random Forests son uno de los métodos de aprendizaje más eficientes y precisos a la fecha" (2008): Caruana: *An empirical evaluation of supervised learning in high dimensions. ICML 2008.*

El algoritmo es sencillo, fácil de implementar, fácil de usar y requiere de poco ajuste de parámetros.

Es relativamente sencillo debuggear los Random Forests; aunque comparado con los Árboles de Decisión, pueda resultar menos interpretable.

Demo Time (demo_9_random_forest)

Boosting

Boosting

- Método para hacer aprendizaje por "ensemble".
- En 1989 Vilian y Kerns plantean un algoritmo al que estimulan “boost” mediante el análisis de los errores cometidos.
- Combina un conjunto de clasificadores débiles para obtener un clasificador más poderoso.
- Los datos se siguen eligiendo en forma aleatoria.
- Es un método iterativo.
- Una vez obtenidos todos los resultados, se establece una jerarquía basada en el nivel de error, teniendo más peso el resultado del árbol que cometió el menor error.

Algoritmo de Boosting

Se definen las entradas, determinando cada una de las variables implicadas: (x_i, y_i)

Donde x_i son el conjunto de datos del árbol i y y_i su clasificación, tal que $y_i \in Y/Y = \{-1, 1\} \approx \{\text{Error}, \text{Acierto}\}$

Se inicializa de forma uniforme, de tal manera que: $D_i = \frac{1}{m} \; \forall \; i = 1, \dots, m$

Durante la primera iteración se entrena el clasificador "débil", utilizando el conjunto de pesos D_i.

Se obtiene la primera hipótesis $h_t: Y$. Se pretende obtener un error bajo de predicción

$\varepsilon_t = Pr_{i\, D_i}[h_t(y) \neq y_i]$

Con los errores obtenidos se actualizan los D_i según lo siguiente:

$\alpha_t = \frac{1}{2} \ln\left(\frac{1-\varepsilon_t}{\varepsilon_t}\right)$

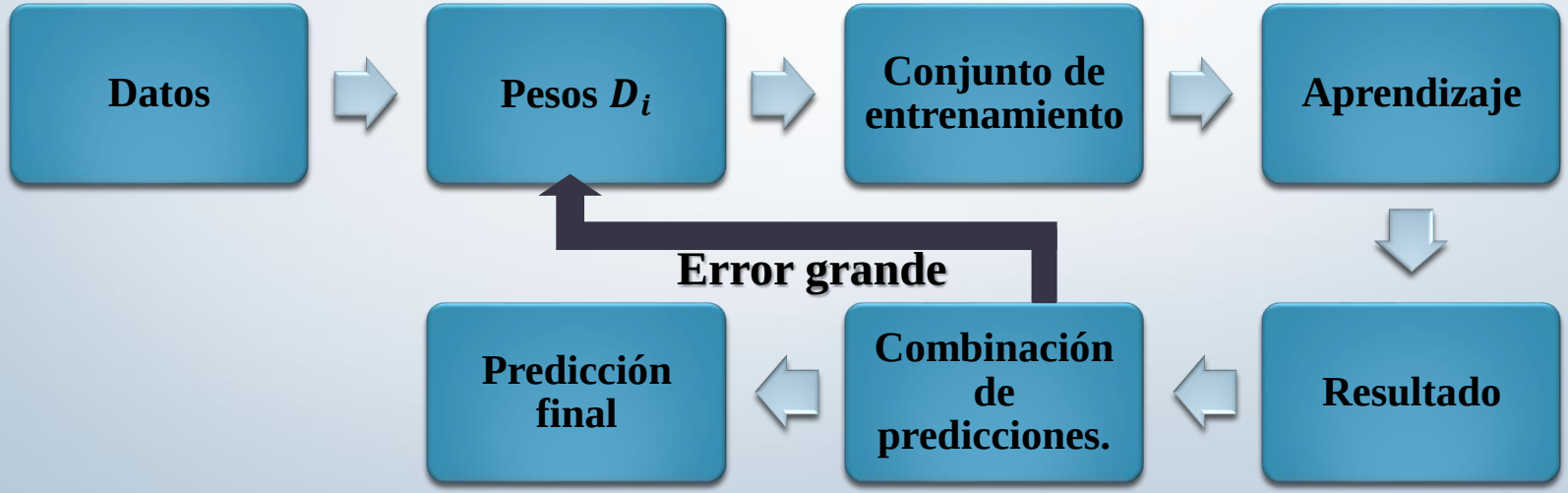
$D_{t+1}(i) = \frac{D_t(i) \exp(\alpha_t \, x_i \, h_t(y_i))}{Z_t}$

Donde Z_t es un factor de normalización.

La hipótesis final quedaría como:

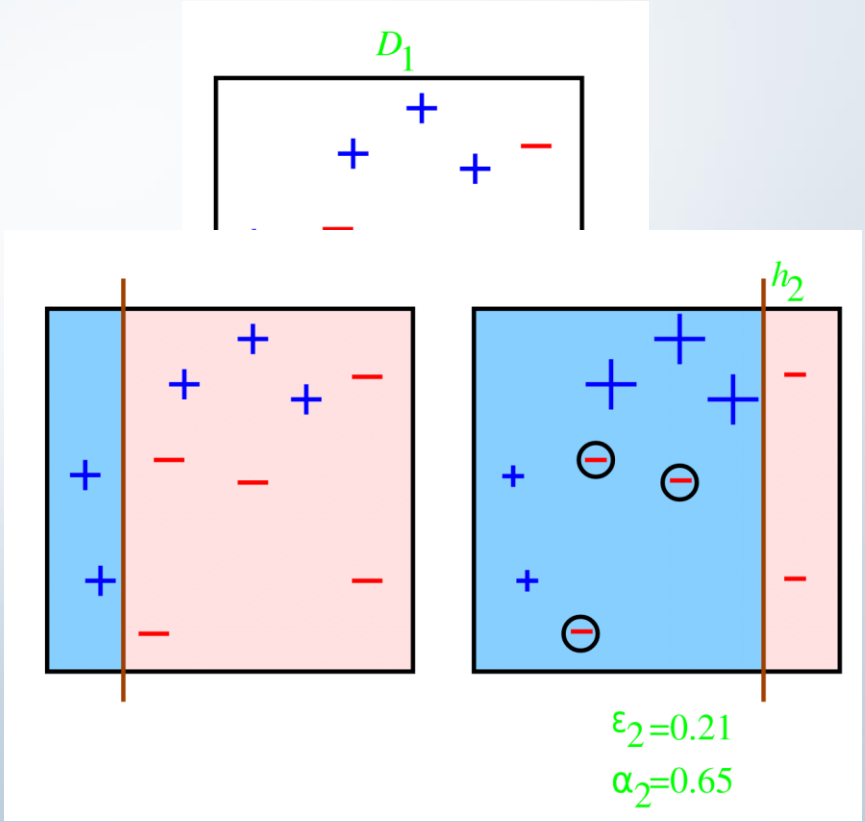
$$H(y) = signo\left(\sum_{t=1}^{T} \alpha_t \, h_t(x)\right)$$

Boosting


Datos
Pesos D_i
Conjunto de entrenamiento
Aprendizaje
Resultado
Combinación de predicciones.
Predicción final
Error grande

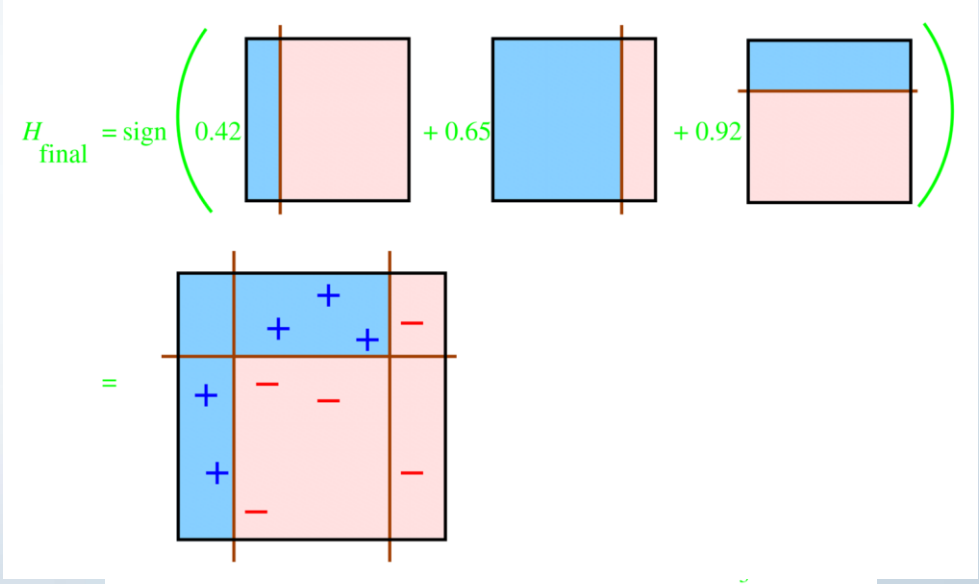
Boosting: Ejemplo (Parte 1)

- Tiene sus variantes, pero en forma general el Boosting tiene los siguientes pasos:
- Se entrenan varios árboles de decisión con conjuntos aleatorios de datos. De forma individual.
- Se le asigna la misma ponderación a cada set de datos de entrenamiento y se aplica al primer modelo de ML, llamado “Modelo Base”.
- El modelo hace predicciones a cada conjunto de datos.
- El algoritmo de Boosting evalúa las predicciones y aumenta la ponderación de las muestras que presenten un error más significativo



Boosting: Ejemplo (Parte 2)

- También asigna una ponderación basada en el rendimiento del modelo
- Un modelo con pocos o ningún error tendrá una ponderación mucho más alta que los demás
- El algoritmo pasa los datos ponderados al siguiente árbol de decisión
- El algoritmo repite los pasos del 4 al 7 hasta que el error de entrenamiento esté por debajo de lo aceptado



Source: http://media.nips.cc/Conferences/2007/Tutorials/Slides/schapire-NIPS-07-tutorial.pdf

Tipos de boosting: Boosting Adaptativo

- Los principales tipos de Boosting son
- Boosting Adaptativo
  - Fue uno de los primeros desarrollados
  - Se adapta y autocorrige en cada iteración
  - Es menos sensible que otros tipos de Boosting.
  - No funciona bien cuando existe correlación entre las features
  - Apto para problemas de clasificación

Tipos de boosting: Boosting por Gradiente

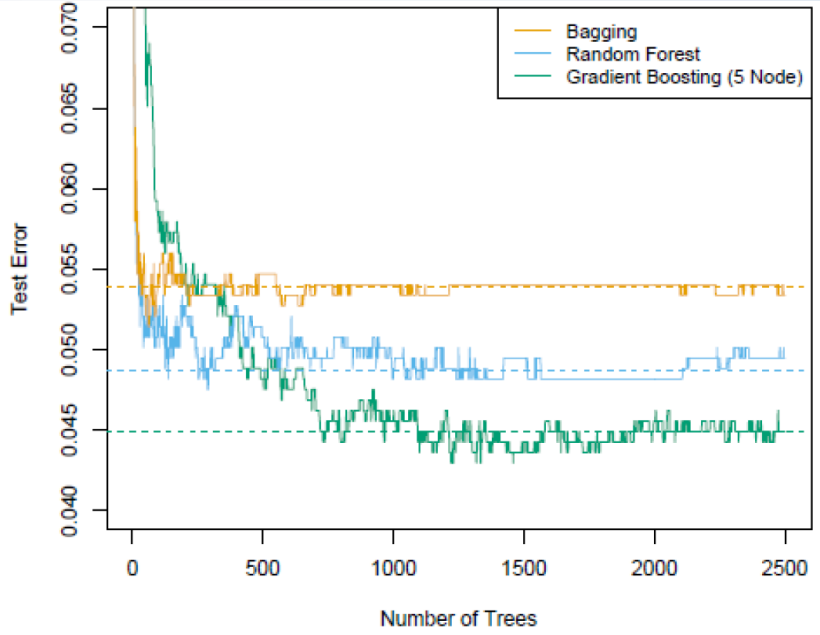
- Boosting por gradiente
  - Similar al adaptativo.
  - No asigna ponderación a los que clasifican de forma incorrecta
  - Se utiliza una función de pérdida para que el nuevo paso sea más eficiente que el anterior
  - Presenta soluciones más efectivas que el anterior
  - Sirve para problemas de clasificación y regresión

Tipos de boosting: Boosting por Gradiente Extremo

- Boosting por Gradiente Extremo
  - Similar al por gradiente.
  - Incrementa la velocidad computacional del anterior.
  - Se realiza procesamiento en paralelo.
  - Se puede utilizar para manejar grandes conjuntos de datos.
  - Sus características clave son la paralelización, la computación distribuida, la optimización de la memoria caché y el procesamiento fuera del núcleo.
  - Sirve para problemas de clasificación y regresión

Ventajas y desventajas del Boosting

Ventajas:	*Desventajas*
Poca variabilidad	No estudia todas las características.
Eficacia computacional	Se necesitan muchos árboles
Sesgo reducido	Implementación en tiempo real
Fácil de implementación	Vulnerabilidad a datos atípicos



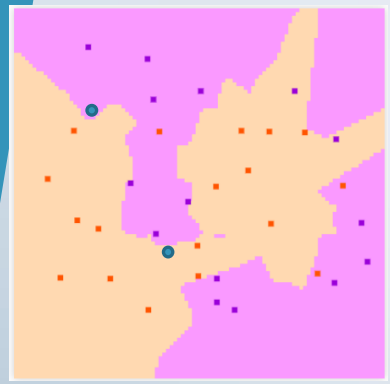
Demo time (demo_10_boosting)

Voting

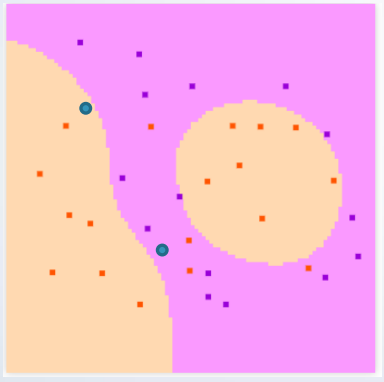
Voting

- Clasificador compuesto de varios clasificadores.
- Cada clasificador que lo compone se entrena sobre el conjunto de datos.
- El clasificador de “voting” final simplemente elige la clase que tuvo “más votos” de parte de los clasificadores que lo componen.
- La votación puede ser “hard” (simplemente se cuenta la cantidad de votos para una clase) o “soft” (se usa la probabilidad).

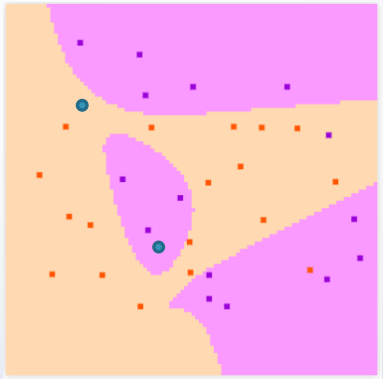
Voting



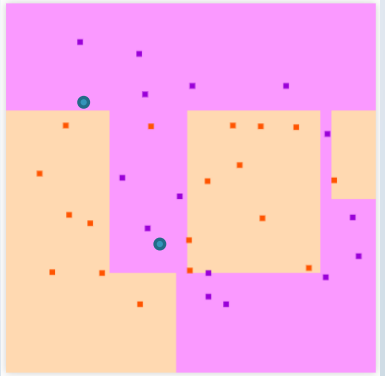
Vecino más cercano
k=3



SVM: c=10
RBF Kernel



ANN



Árbol de decisión

Link

Fin de la tercera clase