BoostAODE: selección de SPODEs guiada por boosting para ensembles bayesianos compactos

Ricardo Montañana, Jose A. Gámez y Jose M. Puerta

Departamento de Sistemas Informáticos, Universidad de Castilla-La Mancha, Albacete, España
Ricardo.Montanana@alu.uclm.es, {Jose.Gamez,Jose.Puerta}@uclm.es

Resumen AODE (*Average One-Dependence Estimators*) es un ensemble bayesiano que promedia los n SPODEs que se obtienen al tomar cada variable como superpadre. Su tamaño, fijo y proporcional al número de variables, lo hace redundante y costoso en problemas de alta dimensionalidad. En este trabajo presentamos BoostAODE, un ensemble que adapta el mecanismo de pesos por instancia del *boosting* para seleccionar de forma incremental un subconjunto de $k \leq n$ SPODEs guiado por el error. Los pesos de *boosting* cumplen una triple función: ponderan el voto de cada SPODE, reorientan la atención hacia las instancias mal clasificadas y, de forma distintiva, recalculan en cada iteración la información mutua que ordena la selección del siguiente superpadre. Un mecanismo de tolerancia progresiva evita la convergencia prematura ante la evolución no monótona de la tasa de acierto. Proponemos además la métrica CLC_α, que combina tasa de acierto y simplicidad del modelo mediante un parámetro de compromiso. Sobre 40 conjuntos de datos, BoostAODE iguala la tasa de acierto de AODE (0,877 frente a 0,870, sin diferencia significativa) empleando una media de 20 SPODEs frente a 72.

Palabras clave: Clasificadores bayesianos · AODE · Boosting · Selección de modelos · Información mutua.

1. Introducción

Los clasificadores bayesianos *semi-naive* relajan la hipótesis de independencia condicional de *naive* Bayes admitiendo dependencias limitadas entre las variables predictoras. Entre ellos, AODE (*Average One-Dependence Estimators*) [13] ha demostrado una notable combinación de precisión y eficiencia. Un SPODE (*SuperParent One-Dependence Estimator*) con superpadre X_i modela la distribución conjunta como $P(y, X_i) \prod_{j \neq i} P(X_j \mid y, X_i)$, esto es, todas las variables dependen de la clase Y y de un único superpadre X_i. AODE construye exactamente un SPODE por cada variable, tomándola como superpadre, y promedia las estimaciones de los n SPODEs resultantes. Esta estrategia evita la costosa búsqueda de estructura, produce un clasificador de baja varianza y obtiene resultados competitivos con técnicas más complejas.

Sin embargo, el tamaño del *ensemble* de AODE es fijo y proporcional al número de variables. En datasets de alta dimensionalidad, muchos de los SPODEs pueden ser redundantes o incluso contraproducentes, al construirse a partir

de variables poco informativas o altamente correlacionadas. Un *ensemble* más reducido es, además, más compacto e interpretable y de menor coste de inferencia, ya que el tiempo de clasificación de AODE escala linealmente con el número de SPODEs. El argumento se intensifica con la familia AnDE (*Average n-Dependence Estimators*) [14], que generaliza AODE admitiendo hasta m superpadres por clasificador base: el número de modelos crece como $O(n)$ para A1DE, $O(n^2)$ para A2DE y, en general, $O(n^m)$ para AmDE, de modo que la reducción activa del *ensemble* deja de ser una mejora cosmética para convertirse en un requisito de tratabilidad.

Proponemos BoostAODE, un *ensemble* de SPODEs que toma prestado del *boosting* [9,11] su mecanismo de pesos por instancia para seleccionar, de forma incremental y guiada por el error, un subconjunto de $k \leq n$ SPODEs combinados mediante voto ponderado. Conviene precisar que BoostAODE no es una aplicación directa del *boosting* clásico: los SPODEs son estimadores de baja varianza, poco proclives a beneficiarse de los esquemas de *ensemble* concebidos para reducir la varianza de estimadores inestables. La clave de nuestra propuesta es que los pesos de *boosting*, además de regular el voto de cada hipótesis y reorientar la atención hacia las instancias mal clasificadas, asumen una tercera función: en cada iteración se recalcula la información mutua entre cada variable candidata y la clase ponderada por los pesos actuales de las instancias, y el ranking resultante decide qué superpadre genera el siguiente SPODE. Los pesos, por tanto, no solo regulan con qué intensidad participa cada hipótesis en la votación, sino que guían activamente qué hipótesis se construyen.

Las contribuciones de este trabajo son: (i) el algoritmo BoostAODE, que integra el manejo de errores del *boosting* con una heurística de selección de superpadres basada en información mutua y un mecanismo de tolerancia progresiva ante la convergencia; (ii) la métrica CLC_α, que permite evaluar conjuntamente tasa de acierto y complejidad del modelo; y (iii) una evaluación empírica frente a AODE sobre 40 conjuntos de datos. El estudio se guía por las siguientes preguntas de investigación: **RQ1**: considerando simultáneamente tasa de acierto y complejidad, ¿ofrece BoostAODE un mejor *trade-off* que AODE? **RQ2**: ¿en qué escenarios de dimensionalidad es más ventajoso? **RQ3**: ¿cuál es la reducción típica de complejidad? **RQ4**: ¿cuándo es AODE suficiente? **RQ5**: ¿cuál es el coste computacional adicional de BoostAODE?

El resto del artículo se organiza como sigue. La Sección 2 revisa el trabajo relacionado. La Sección 3 presenta el algoritmo BoostAODE y la métrica CLC_α. La Sección 4 describe el diseño experimental y la Sección 5 analiza los resultados. Finalmente, la Sección 6 recoge las conclusiones y líneas futuras.

2. Trabajo relacionado

Debido al número de modelos incluidos en AODE, $O(n)$, y especialmente en AmDE, $O(n^m)$, el principal inconveniente para estos algoritmos es que el ensemble resultante puede dejar de ser tratable debido a la memoria necesaria para almacenar sus estadísticos suficientes. Este fenómeno ha sido descrito como una transición desde un problema de *big data* hacia un problema de *big model* [1].

Esta limitación ha motivado distintas aproximaciones destinadas a reducir el número de submodelos incluidos en el ensemble. Una primera línea de trabajo sustituye el ensemble explícito por un modelo probabilístico más compacto. HODE (*Hidden One-Dependence Estimator*) [8], por ejemplo, introduce una variable oculta que actúa como superpadre, con el objetivo de resumir en un único modelo compacto el comportamiento en forma de mixtura del ensemble de SPODEs. HODE persigue reducir tanto los requisitos de memoria como el tiempo de clasificación, manteniendo la capacidad predictiva de AODE.

Una segunda familia de métodos consiste en seleccionar únicamente un subconjunto de los SPmDEs candidatos. La selección puede realizarse mediante distintos criterios, incluyendo medidas de teoría de la información, error de clasificación o procedimientos aleatorizados [10,4,3]. Los enfoques de tipo *wrapper*, que evalúan directamente distintos subconjuntos de submodelos en función del error de clasificación, optimizan directamente una medida estrechamente alineada con el objetivo final del clasificador; sin embargo, su coste computacional es prohibitivo salvo en dominios de muy baja dimensionalidad.

Por este motivo, la mayoría de propuestas prácticas recurren a criterios de tipo *filter*, especialmente aquellos basados en información mutua (condicional). Selective AnDE y Sample-based Attribute Selective AnDE son ejemplos representativos de esta estrategia [4,3]. Estos métodos reducen el número de superpadres considerados al construir el modelo AnDE, permitiendo obtener clasificadores sujetos a restricciones de memoria que pueden aplicarse a dominios de mayor tamaño. No obstante, el uso de medidas de teoría de la información como criterio principal de selección descansa sobre la hipótesis de que dichas medidas constituyen un buen *proxy* del error de clasificación. Sin embargo, esta hipótesis no siempre se cumple, y seleccionar SPmDEs con una alta información mutua condicional respecto a la clase no implica necesariamente una reducción proporcional del error del ensemble. Así, experimentos previos han mostrado que, en determinados escenarios, la selección basada exclusivamente en información mutua puede comportarse de forma similar a una selección aleatoria de SPmDEs [2]. Esto limita la capacidad de los métodos *filter* puros para identificar los submodelos más útiles desde el punto de vista predictivo.

3. El clasificador BoostAODE

3.1. Relación estructural con AODE

Ambos clasificadores son *ensembles* de SPODEs que comparten el mismo grafo subyacente: dado un superpadre X_i, los restantes $n-1$ atributos son hijos conjuntos de Y y de X_i. En consecuencia, cada SPODE queda unívocamente determinado por la elección de su superpadre, y la construcción del *ensemble* BoostAODE se reduce a decidir qué superpadres se incorporan y en qué orden. La diferencia fundamental entre ambos clasificadores se resume en la Tabla 1. Esta relación permite definir una métrica de complejidad natural basada en el ratio entre el número de SPODEs seleccionados y el total de variables disponibles (Sección 3.6).

Tabla 1. Comparación estructural entre AODE y BoostAODE.

Aspecto	AODE	BoostAODE
SPODEs en el *ensemble*	Todos (n)	Subconjunto ($k \leq n$)
Pesos de los SPODEs	Uniformes	Ponderados por *boosting*
Selección del superpadre	Ninguna (todos)	Información mutua ponderada
Complejidad del modelo	Fija: proporcional a n	Variable: proporcional a k

3.2. Esquema general del algoritmo

BoostAODE construye el *ensemble* de forma incremental. Tras inicializar un peso uniforme $w_i^{(0)} = 1/N$ para cada una de las N instancias de entrenamiento y a partir de un *ensemble* vacío, cada iteración repite los siguientes pasos: (1) se calcula la información mutua entre cada variable candidata y la clase, teniendo en cuenta los pesos actuales de las instancias, y se selecciona como superpadre la variable más relevante todavía no utilizada; (2) el nuevo SPODE se entrena sobre los datos ponderados y se calcula su error ϵ, del que se obtiene su importancia o coeficiente de voto $\alpha_t = \frac{1}{2} \ln \frac{1-\epsilon}{\epsilon}$, de forma análoga a AdaBoost; (3) los pesos de las instancias mal clasificadas se incrementan y los de las correctamente clasificadas se reducen, orientando la siguiente iteración hacia los ejemplos aún no resueltos; y (4) si la incorporación del nuevo SPODE no mejora el rendimiento del *ensemble*, se activa el mecanismo de tolerancia progresiva; de persistir la falta de mejora, se declara la convergencia y se eliminan los últimos SPODEs improductivos. Las dos subsecciones siguientes detallan las decisiones de diseño no triviales: el orden en que se procesan las variables (Sección 3.3) y la gestión de la evolución no monótona de la tasa de acierto (Sección 3.4).

3.3. Orden de procesamiento de variables

En AdaBoost, el espacio de clasificadores débiles se explora de forma implícita mediante el repesado, sin un criterio explícito sobre qué clasificador construir en cada iteración. En BoostAODE, sin embargo, cada SPODE queda determinado por la elección de su superpadre, lo que convierte la construcción del *ensemble* en un problema de selección y ordenación de variables. Como el algoritmo incorpora SPODEs de forma incremental y puede detenerse antes de agotar las variables, el orden de procesamiento influye directamente en la calidad del *ensemble* resultante. BoostAODE ordena las variables candidatas en cada iteración según su información mutua con la clase, ponderada por los pesos actuales de las instancias. Esta decisión se fundamenta en tres razones. *Priorización de variables informativas*: un SPODE cuyo superpadre tiene alta información mutua con la clase tiende a producir predicciones más precisas, lo que permite alcanzar un buen rendimiento con pocos SPODEs y favorece una convergencia temprana. *Adaptación dinámica al repesado*: al recalcular la información mutua en cada iteración con los pesos actualizados, el ranking refleja qué variables son más relevantes para las instancias que el *ensemble* aún clasifica mal, de forma análoga a cómo AdaBoost redirige la atención hacia los errores. *Interacción con la convergencia*: cuando la tolerancia se activa y se construyen varios SPODEs en una sola

iteración, el ranking garantiza que se empleen las variables más prometedoras del conjunto restante.

3.4. Mecanismo de convergencia

En AdaBoost los clasificadores base son débiles e independientes, lo que favorece que cada nuevo clasificador aporte una mejora marginal. En BoostAODE, en cambio, los SPODEs son modelos relativamente expresivos que comparten estructura y se diferencian solo en el superpadre. Esta interdependencia provoca que la evolución de la tasa de acierto al incorporar SPODEs no sea necesariamente monótona: un nuevo SPODE puede solaparse parcialmente con los ya incluidos y no mejorar el rendimiento inmediato, pero la redistribución de pesos que provoca puede hacer que el siguiente SPODE capture un patrón complementario que sí aporte una mejora significativa. Por ejemplo, en una secuencia típica el *ensemble* puede estancarse en 0,7465 al añadir un SPODE y saltar a 0,7830 con el siguiente: un criterio de parada estricto detendría el proceso prematuramente y perdería esa mejora.

Esta observación motiva el esquema de **tolerancia progresiva**: en lugar de detenerse al primer signo de estancamiento, el algoritmo amplía progresivamente la exploración construyendo $2^0, 2^1, 2^2, 2^3$ SPODEs en iteraciones sucesivas sin mejora, antes de declarar la convergencia. Este diseño equilibra el riesgo de sobreajuste por incluir SPODEs innecesarios con el riesgo de convergencia prematura por descartar SPODEs potencialmente útiles.

3.5. Algoritmo

El Algoritmo 1 presenta el pseudocódigo de BoostAODE. El método toma prestado de AdaBoost únicamente su aparato de contabilidad de pesos, esto es, el cálculo del coeficiente de voto α_t a partir del error ponderado y la actualización de la distribución de pesos, e incorpora un elemento ausente en AdaBoost y central en su diseño: la ordenación de las variables candidatas por información mutua ponderada, que sustituye la exploración implícita del espacio de hipótesis por una selección explícita del superpadre del siguiente SPODE. El mecanismo de convergencia, con su esquema de tolerancia progresiva (incremento del número de SPODEs construidos por iteración en potencias de 2), detiene el algoritmo cuando la incorporación de nuevos SPODEs no mejora el rendimiento del *ensemble*.

3.6. La métrica CLC_α

Dado que BoostAODE busca expresamente reducir la complejidad del *ensemble*, una evaluación justa debe considerar simultáneamente precisión y complejidad. Definimos la métrica CLC_α (*Complexity-adjusted Learning Criterion*) como:

$$\mathrm{CLC}_\alpha = \alpha \cdot \mathrm{accuracy} + (1 - \alpha) \cdot \mathrm{simplicity}, \tag{1}$$

donde $\mathrm{accuracy} \in [0, 1]$ es la tasa de acierto, $\mathrm{simplicity} = 1 - n_{\mathrm{spodes}}/n_{\mathrm{features}} \in [0, 1)$ mide la fracción de SPODEs que el modelo *no* necesita, y $\alpha \in [0, 1]$ es el parámetro de *trade-off*. Para AODE, $n_{\mathrm{spodes}} = n_{\mathrm{features}}$, luego $\mathrm{simplicity} = 0$ y $\mathrm{CLC}_\alpha = \alpha \cdot \mathrm{accuracy}$: AODE recibe la penalización máxima de complejidad.

Algoritmo 1 BoostAODE

Require: Conjunto de entrenamiento $D = \{(x_1^i, \ldots, x_n^i, y^i)\}_{i=1}^N$, donde x^i es el vector de variables e $y^i \in \{0, 1, ..., C-1\}$ es la etiqueta de clase.
Require: Variables $\mathcal{X} = \{X_1, \ldots, X_n\}$
Ensure: *ensemble* de SPODEs con sus coeficientes de voto

▷ Inicialización de variables del algoritmo
$W \leftarrow (w_1, \ldots, w_N)$ con $w_i \leftarrow \frac{1}{N}$
$Vars \leftarrow \mathcal{X}$
$\delta \leftarrow 10^{-4}$
$maxTolerance \leftarrow 3$
$finished \leftarrow false$
$ensemble \leftarrow \emptyset$
$tolerance \leftarrow 0$
$numItemsPack \leftarrow 0$
$maxAccuracy \leftarrow -1$
▷ Bucle principal
while $\neg finished$ **do**
    $\pi \leftarrow SortFeatures(Vars, D[W])$ ▷ Orden por información mutua ponderada por W
    $z \leftarrow 2^{tolerance}$
    if $tolerance == 0$ **then**
        $numItemsPack \leftarrow 0$
    end if
    $P \leftarrow Head(\pi, z)$ ▷ Primeras z variables en orden
    $spodes \leftarrow \emptyset$
    $i \leftarrow 0$
    while $i < size(P)$ **do** ▷ Crea un SPODE para cada variable en P
        $X \leftarrow P[i]$
        $i \leftarrow i + 1$
        $numItemsPack \leftarrow numItemsPack + 1$
        $Vars.remove(X)$
        $spode \leftarrow BuildSpode(X, \mathcal{X}, D[W])$
        $\hat{y}[] \leftarrow spode.Predict(D)$
        $\epsilon \leftarrow error(\hat{y}[], y[], W)$ ▷ Error ponderado por W
        if $\epsilon > 0.5$ **then**
            $finished \leftarrow true$
            break
        end if
        $\alpha_t \leftarrow \frac{1}{2} ln\left(\frac{1-\epsilon}{\epsilon}\right)$
        $spodes.add((spode, \alpha_t))$
        $W \leftarrow UpdateWeights(W, \alpha_t, y[], \hat{y}[])$ ▷ Actualización de pesos tipo AdaBoost
    end while
    $ensemble.add(spodes)$ ▷ Añade el grupo de SPODEs creado
    if $\neg finished$ **then**
        $\hat{y}[] \leftarrow ensemble.Predict(D)$
        $actualAccuracy \leftarrow accuracy(\hat{y}[], y[])$
        if $(actualAccuracy - maxAccuracy) < \delta$ **then** ▷ No hay mejora suficiente
            $tolerance \leftarrow tolerance + 1$
        else
            $tolerance \leftarrow 0$
            $numItemsPack \leftarrow 0$
            $maxAccuracy \leftarrow actualAccuracy$
        end if
    end if
    if $Vars == \emptyset \vee tolerance > maxTolerance$ **then**
        $finished \leftarrow true$
    end if
end while
if $tolerance > maxTolerance$ **then** ▷ Termina por falta de convergencia
    $removeModels(ensemble, numItemsPack)$
end if

Para BoostAODE, $n_{\text{spodes}} = k \leq n_{\text{features}}$, luego simplicity ≥ 0 y recibe un *bonus* proporcional a los SPODEs ahorrados. El caso $\alpha = 1{,}0$ corresponde a la comparación de tasa de acierto pura y $\alpha = 0{,}5$ a un peso equilibrado.

El α de *break-even* es el valor de α a partir del cual AODE supera a BoostAODE en CLC_α. Igualando ambas métricas se obtiene:

$$\alpha_{\text{break-even}} = \frac{\text{simplicity}_{\text{Boost}}}{\text{simplicity}_{\text{Boost}} + \text{acc}_{\text{AODE}} - \text{acc}_{\text{Boost}}}. \tag{2}$$

Si $\text{acc}_{\text{Boost}} \geq \text{acc}_{\text{AODE}}$, BoostAODE domina en todos los valores de α y no existe *break-even*.

4. Diseño experimental

Clasificadores. Se comparan dos clasificadores: AODE (*baseline*), que utiliza todas las variables como superpadres generando n SPODEs; y BoostAODE (propuesto), que aplica *boosting* para seleccionar un subconjunto de $k \leq n$ SPODEs. Ambos[1] se ejecutaron con sus hiperparámetros por defecto, sin ajuste, siguiendo el enfoque estándar en estudios comparativos amplios [7].

Conjuntos de datos. Se emplearon 40 conjuntos de datos (Tabla 2) de los repositorios UCI [6] y OpenML [12], seleccionados para cubrir un amplio espectro de dimensionalidad, tamaño y número de clases.

Tabla 2. Resumen de los 40 conjuntos de datos por categoría de dimensionalidad. n: variables, N: instancias, C: clases.

Categoría	# datasets	rango n	rango N	rango C
Pequeños ($n < 10$)	13	3–9	150–12.960	2–10
Medianos ($10 \leq n \leq 99$)	22	10–72	178–130.064	2–26
Grandes ($n \geq 100$)	5	167–617	1.593–10.299	2–26

Configuración. Se empleó validación cruzada estratificada de 5 particiones con 5 repeticiones (semillas fijas: 42, 123, 456, 789, 1024), generando 25 evaluaciones por par clasificador-dataset y un total de 2.000 evaluaciones ($40 \times 2 \times 5 \times 5$). El resultado por dataset es la media de las 25 evaluaciones.

Métricas y metodología. La métrica principal es CLC_α (ec. 1), evaluada para $\alpha \in \{0{,}5; 0{,}6; 0{,}7; 0{,}8; 0{,}9; 1{,}0\}$; adicionalmente se recogen los tiempos de entrenamiento y predicción (RQ5). El test de Shapiro-Wilk sobre las diferencias pareadas (BoostAODE − AODE) rechaza la normalidad en todas las métricas ($p < 0{,}05$), lo que justifica el uso del test no paramétrico de Wilcoxon *signed-rank* [5], con nivel de significación $\alpha_{\text{test}} = 0{,}05$. Para responder a RQ2 y RQ4 se repite el test sobre cada categoría de dimensionalidad. Se reportan además estadísticos de victoria/empate/derrota, ratios de compresión y el α de *break-even*.

[1] https://github.com/Doctorado-ML/BayesNet

5. Resultados y análisis

5.1. Visión de conjunto

BoostAODE obtuvo una tasa de acierto media de 0,8772 frente a 0,8701 de AODE, con 22 victorias, 1 empate y 17 derrotas en comparación directa por dataset. Esta diferencia no es significativa ($W = 312{,}0$, $p = 0{,}19$), pero al considerar CLC_α, que incorpora la simplicidad del modelo, BoostAODE es estadísticamente superior a AODE para todos los valores evaluados de $\alpha \leq 0{,}9$, con 32 victorias de 40 para $\alpha \leq 0{,}8$ y 31 para $\alpha = 0{,}9$ (Tabla 3). La ventaja de BoostAODE, por tanto, reside en la capacidad de lograr rendimientos comparables con modelos mucho más simples: una media de 20 submodelos frente a los 72 de AODE.

Para aislar la contribución de cada componente comparamos, con el mismo diseño experimental, BoostAODE con tres variantes ablacionadas: sin pesos de *boosting*, con selección aleatoria de superpadres en lugar de información mutua y con ambas ablaciones. El componente crítico resulta ser el esquema de pesos: al anularlo, la tasa de acierto cae de 0,8772 a 0,8591, por debajo de AODE (0,8701), con diferencias significativas; la aportación de la selección por información mutua es marginal con respecto a seleccionar el superpadre de forma aleatoria, tanto si se usan los pesos, como si no. La compresión se mantiene en todas las variantes, pues la produce el mecanismo de convergencia: los pesos son, por tanto, los que convierten esa compactación en una ventaja sin coste en tasa de acierto. Los resultados completos están publicados en la página web del proyecto[2].

Tabla 3. Test de Wilcoxon *signed-rank* para cada métrica (BoostAODE vs. AODE). V, E y D: número de datasets en los que BoostAODE obtiene un resultado superior, equivalente o inferior a AODE.

Métrica	W	p-valor	Sig.	V	E	D
$\mathrm{CLC}_{0,5}$	53,0	$8{,}20 \times 10^{-8}$	✓	32	1	7
$\mathrm{CLC}_{0,6}$	54,0	$9{,}23 \times 10^{-8}$	✓	32	1	7
$\mathrm{CLC}_{0,7}$	57,0	$1{,}30 \times 10^{-7}$	✓	32	1	7
$\mathrm{CLC}_{0,8}$	64,0	$2{,}82 \times 10^{-7}$	✓	32	1	7
$\mathrm{CLC}_{0,9}$	84,0	$2{,}04 \times 10^{-6}$	✓	31	1	8
$\mathrm{CLC}_{1,0}$ (Accuracy)	312,0	0,1922		22	1	17

5.2. Tasa de acierto y CLC_α

La Figura 1 muestra la comparación directa de tasa de acierto por dataset: la mayoría de los puntos se agrupan cerca de la diagonal. Es notable la asimetría en las magnitudes: las mayores victorias de BoostAODE (p. ej., `kr-vs-kp` con +7,2 pp, `tic-tac-toe` con +6,2 pp) son considerablemente más amplias que las mayores pérdidas, que apenas alcanzan −2,3 pp (`balance-scale`). La ventaja de BoostAODE en CLC_α se reduce gradualmente a medida que α se acerca a 1,0

[2] https://rmontanana.github.io/boostaode

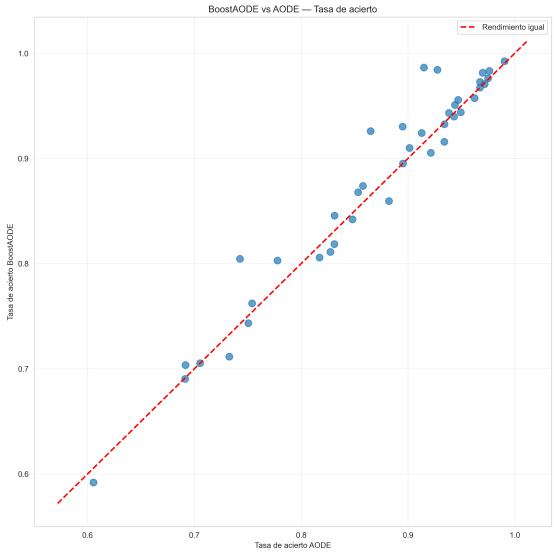


Figura 1. Comparación de tasa de acierto por dataset. Los puntos por encima de la diagonal indican datasets donde BoostAODE supera a AODE y viceversa.

y la complejidad pierde peso relativo: la diferencia media pasa de +0,1426 para $\alpha = 0{,}5$ a +0,0071 para $\alpha = 1{,}0$, lo que confirma que la principal contribución de BoostAODE proviene de la simplicidad del modelo.

5.3. Análisis de complejidad

BoostAODE utiliza una media de 20,0 SPODEs frente a los 72,0 de AODE. Definimos el *ratio de compresión* como el cociente $n_{\text{spodes}}/n_{\text{features}}$: valores próximos a 1 indican ausencia de compresión (BoostAODE reproduce la estructura de AODE) y valores próximos a 0 indican compresión agresiva. El ratio medio es 0,72 y la mediana 0,89, con una distribución *bimodal*: 13 datasets (32,5 %) sin compresión (ratio = 1,0, en su mayoría con menos de 10 variables) y 6 datasets (15 %) con compresión muy alta (ratio $< 0{,}3$, todos con 60 o más variables). En estos la pérdida de tasa de acierto es mínima o inexistente: `har` (561 variables) supera a AODE en +1,2 pp usando solo 77,5 de los 561 SPODEs posibles (ratio 0,14), y `musk` (167 variables) mantiene la tasa de acierto con solo el 25 % de los SPODEs. La compresión crece con la dimensionalidad: los datasets con más variables presentan ratios significativamente menores, lo que es coherente con la mayor redundancia esperada entre SPODEs en espacios de alta dimensionalidad.

5.4. Análisis segmentado y *break-even*

La Tabla 4 desglosa el test de Wilcoxon por categoría de dimensionalidad. La mayor ventaja se concentra en los **datasets grandes** (≥ 100 variables), con una diferencia media de +0,4219 en $\text{CLC}_{0,5}$ y victoria en los 5 datasets para $\alpha \leq 0{,}9$ (el reducido tamaño del segmento impide la significación formal). En

los **datasets medianos** (10–99 variables), BoostAODE gana en 21 de 22 para $\alpha = 0{,}5$ ($p = 2{,}38 \times 10^{-6}$); un hallazgo destacado es que la ventaja es significativa *incluso en tasa de acierto pura* ($\alpha = 1{,}0$, $p = 0{,}0425$), lo que indica que el *boosting* no solo simplifica el modelo sino que mejora la precisión cuando hay variables candidatas suficientes. En **datasets pequeños** ($n < 10$) no hay diferencias significativas ($p = 0{,}41$ para $\alpha = 0{,}5$): con tan pocos SPODEs candidatos, el proceso de selección apenas tiene margen y ambos clasificadores son equivalentes.

Tabla 4. Análisis segmentado por tamaño de datasets (test de Wilcoxon) para valores representativos de α. V, E, D: victorias, empates y derrotas de BoostAODE vs AODE.

Segmento	N	α	p-valor	Sig.	V	E	D
Todos	40	0,5	$8{,}20 \times 10^{-8}$	✓	32	1	7
		0,8	$2{,}82 \times 10^{-7}$	✓	32	1	7
		1,0	0,1922		22	1	17
Pequeños ($n < 10$)	13	0,5	0,4143		6	1	6
		0,8	0,5417		6	1	6
		1,0	1,0000		5	1	7
Medianos (10−99)	22	0,5	$2{,}38 \times 10^{-6}$	✓	21	0	1
		0,8	$2{,}38 \times 10^{-6}$	✓	21	0	1
		1,0	0,0425	✓	15	0	7
Grandes ($n \geq 100$)	5	0,5	—		5	0	0
		0,8	—		5	0	0
		1,0	—		2	0	3

El análisis del α de *break-even* refuerza esta lectura. De los 40 datasets, en 22 (55 %) BoostAODE domina en todo α (tasa de acierto igual o superior a AODE); en 7 (17,5 %) el *break-even* es 0 (sin compresión, clasificadores idénticos); y en los 11 restantes (27,5 %) existe un *break-even* positivo con mediana 0,979. Es decir, en la mitad de los datasets con *break-even* positivo sería necesario otorgar un peso superior al 97,9 % a la tasa de acierto para que AODE resultase preferible.

5.5. Coste computacional

Los tiempos absolutos (54,75 s de media en entrenamiento y 0,19 s en predicción) son prácticos. El mayor coste de entrenamiento se debe al proceso iterativo de *boosting*, que evalúa el error del *ensemble* y actualiza los pesos en cada iteración e impide la paralelización trivial de AODE. En predicción, BoostAODE tiene coste $\mathcal{O}(k \cdot n \cdot N \cdot C)$ frente a $\mathcal{O}(n^2 \cdot N \cdot C)$ de AODE, de modo que es *estrictamente más rápida* cuando $k < n$, con ganancia proporcional al ratio n/k.

5.6. Discusión

Ordenamos la discusión del trabajo en base a las preguntas de investigación formuladas al inicio.

RQ1: ¿Ofrece BoostAODE mejor *trade-off*? Sí, de forma estadísticamente significativa. BoostAODE supera a AODE en CLC_α para todo α entre 0,5 y 0,9

($p < 0{,}05$). En tasa de acierto pura muestra una ligera ventaja no significativa (0,8772 vs. 0,8701, $p = 0{,}19$).

RQ2: ¿En qué escenarios es más ventajoso? En datasets de alta dimensionalidad (≥ 100 variables; diferencia media +0,42 en $\mathrm{CLC}_{0,5}$ con compresión muy alta) y de dimensionalidad media (10–99 variables; +0,16 en $\mathrm{CLC}_{0,5}$ y ventaja significativa incluso en tasa de acierto pura).

RQ3: ¿Cuál es la reducción típica de complejidad? Depende de la dimensionalidad: nula en datasets pequeños, variable (ratios 0,3–0,9) en medianos y muy alta (ratios $< 0{,}25$) en grandes.

RQ4: ¿Cuándo es AODE suficiente? Problemas con pocas variables (igual accuracy y AODE más rápido) y en aplicaciones con restricciones estrictas de tiempo de entrenamiento, donde AODE es entre 64× y 127× más rápido.

RQ5: ¿Cuál es el coste computacional? Superior al de AODE en entrenamiento y, por efecto de la vectorización, también en predicción; pero con tiempos absolutos prácticos y una predicción asintóticamente más rápida que AODE bajo igualdad de implementaciones cuando $k < n$.

Recomendación práctica. Usar BoostAODE en problemas de dimensionalidad media-alta; usar AODE directamente para datasets de baja dimensionalidad o cuando el tiempo de entrenamiento sea crítico.

6. Conclusiones y trabajo futuro

Hemos presentado BoostAODE, un *ensemble* de SPODEs que adapta el mecanismo de pesos por instancia del *boosting* para seleccionar de forma incremental y guiada por el error un subconjunto compacto de clasificadores base, combinando el manejo de errores de AdaBoost con una heurística de selección de superpadres basada en información mutua ponderada y un mecanismo de tolerancia progresiva. Junto al algoritmo, hemos propuesto la métrica CLC_{α} para evaluar conjuntamente precisión y complejidad. La evaluación sobre 40 datasets muestra que BoostAODE iguala la tasa de acierto de AODE con una media de 20 SPODEs frente a 72, y resulta significativamente superior en CLC_{α} para $\alpha \in [0{,}5,\ 0{,}9]$, con la mayor ventaja en datos de dimensionalidad media y alta.

Como trabajo futuro, la estrategia de selección es directamente extrapolable a órdenes superiores de la familia AnDE, donde el crecimiento $O(n^m)$ del número de clasificadores base convierte la reducción del *ensemble* en un requisito de tratabilidad. También planeamos investigar criterios de ordenación alternativos a la información mutua y de inicialización no vacía del subconjunto de SPODEs.

Agradecimientos Este trabajo ha sido parcialmente financiado por MCIU/AEI/10.13039/501100011033 y el FEDER a través del proyecto PID2022-139293NB-C32; y por la Universidad de Castilla-La Mancha y el «FEDER Una manera de hacer Europa» en el marco del proyecto 2025-GRIN-38476.

Referencias

1. Arias, J., Gámez, J.A., Puerta, J.M.: Learning distributed discrete Bayesian network classifiers under MapReduce with Apache Spark. Knowledge-Based Systems **117**, 16–26 (2017). https://doi.org/10.1016/j.knosys.2016.06.013
2. Arias, J., Gámez, J.A., Puerta, J.M.: Bayesian network classifiers under the ensemble perspective. In: Proceedings of the Ninth International Conference on Probabilistic Graphical Models (PGM 2018). Proceedings of Machine Learning Research, vol. 72, pp. 1–12. PMLR (2018)
3. Chen, S., Martínez, A.M., Webb, G.I., Wang, L.: Sample-based attribute selective AnDE for large data. IEEE Transactions on Knowledge and Data Engineering **29**(1), 172–185 (2017). https://doi.org/10.1109/TKDE.2016.2608881
4. Chen, S., Martínez, A.M., Webb, G.I., Wang, L.: Selective AnDE for large data learning: A low-bias memory constrained approach. Knowledge and Information Systems **50**(2), 475–503 (2017). https://doi.org/10.1007/s10115-016-0937-9
5. Demšar, J.: Statistical comparisons of classifiers over multiple data sets. Journal of Machine Learning Research **7**, 1–30 (2006)
6. Dua, D., Graff, C.: UCI machine learning repository (2017), http://archive.ics.uci.edu/ml
7. Fernández-Delgado, M., Cernadas, E., Barro, S., Amorim, D.: Do we need hundreds of classifiers to solve real world classification problems? Journal of Machine Learning Research **15**(90), 3133–3181 (2014)
8. Flores, M.J., Gámez, J.A., Martínez, A.M., Puerta, J.M.: HODE: Hidden one-dependence estimator. In: Symbolic and Quantitative Approaches to Reasoning with Uncertainty. LNCS, vol. 5590, pp. 481–492. Springer (2009)
9. Freund, Y., Schapire, R.E.: A decision-theoretic generalization of on-line learning and an application to boosting. Journal of Computer and System Sciences **55**(1), 119–139 (Aug 1997). https://doi.org/10.1006/jcss.1997.1504
10. Martínez, A.M., Webb, G.I., Chen, S., Zaidi, N.A.: Scalable learning of Bayesian network classifiers. J. Mach. Learn. Res. **17**, 44:1–44:35 (2016)
11. Schapire, R.E.: The Boosting Approach to Machine Learning: An Overview, pp. 149–171. Springer NY (2003). https://doi.org/10.1007/978-0-387-21579-2_9
12. Vanschoren, J., van Rijn, J.N., Bischl, B., Torgo, L.: OpenML: Networked science in machine learning. SIGKDD Explorations **15**(2), 49–60 (2014)
13. Webb, G.I., Boughton, J.R., Wang, Z.: Not so naive Bayes: Aggregating one-dependence estimators. Machine Learning **58**(1), 5–24 (Jan 2005)
14. Webb, G.I., Boughton, J.R., Zheng, F., Ting, K.M., Salem, H.: Learning by extrapolation from marginal to full-multivariate probability distributions: decreasingly naive Bayesian classification. Machine Learning **86**(2), 233–272 (2012)