IADEM-0: Un Nuevo Algoritmo Incremental *

Gonzalo Ramos-Jiménez, Rafael Morales-Bueno y José del Campo-Ávila

Departamento de Lenguajes y Ciencias de la Computación
E.T.S. Ingeniería Informática. Universidad de Málaga
Málaga, 29071, España

Resumen El aprendizaje incremental es un buen método para la clasificación cuando los conjuntos de experiencias que se van a clasificar son muy grandes o cuando las experiencias pueden llegar en cualquier momento. Para reducir los requisitos de memoria de los algoritmos es necesario olvidar las experiencias que se van usando y conservar únicamente los conocimientos que se van aprendiendo. Para realizar este olvido es necesario contar con una representación probabilística del conocimiento que se está extrayendo. El algoritmo que presentamos, y al que llamamos IADEM-0, ha sido desarrollado teniendo en cuenta estos enfoques y para llevarlo acabo se han usado las cotas de concentración de Chernoff y Hoeffding. Este nuevo algoritmo cuenta con distintas características muy positivas, pero la fundamental es que está preparado para extraer conocimiento, en forma de árbol de decisión, de conjuntos de experiencias muy grandes sin que le afecte el tamaño de dicho conjunto. Esto se consigue al no almacenar ninguna experiencia de las que explora. Otro aspecto novedoso es que el usuario puede fijar, con la confianza deseada, el error máximo que se requiere para el árbol que induzca IADEM-0. Además, nuestro algoritmo da información detallada sobre el vector de predicción, mostrando los valores estimados para cada clase y proporcionando un vector con los márgenes de error para cada valor estimado, todo ello con la confianza requerida.

1. Introducción

Los sistemas de aprendizaje automático se usan para trabajar sobre bases de datos que cada vez son más grandes. Además, recientemente están surgiendo los denominados flujos de datos [1], cuya principal característica es que los datos llegan constantemente sin parar. Así, el tamaño de los conjuntos de experiencias puede crecer continuamente a un ritmo bastante alto cada día. Existe una gran cantidad de información en estos conjuntos de datos, ya sean bases de datos o flujos de datos, y es muy deseable poder inducir conocimiento a partir de ellos.

El proceso de extraer conocimiento de dichos conjuntos de experiencias se convierte en un problema inabordable para los algoritmos tradicionales como el ID3 [2][3] o el C4.5 [4] debido a sus requisitos de memoria.

* Este trabajo ha sido parcialmente financiado por el proyecto MOISES, número TIC-2002-04019-C03-02 del Ministerio de Ciencia y Tecnología.

Existen otras aproximaciones basadas en disco que mantienen listas de atributos para cada registro, como pueden ser SLIQ [5] o SPRINT [6], pero siguen teniendo limitaciones a pesar de hacer un uso más eficiente de la memoria. Necesitan dar múltiples pasadas al conjunto de experiencias almacenado en el disco, por lo que necesitan conocer todas las experiencias antes de ejecutarse y no pueden tratar flujos de datos continuos. Tampoco pueden tratar bases de datos mayores que el tamaño del disco.

La representación probabilística [7] nos permite explorar los datos mientras mantenemos la información más relevante extraída de las experiencias exploradas [8]. De esta forma, los requisitos de memoria no tienen que depender del número de experiencias en el conjunto de datos, sino de la estructura de conocimiento que se esté extrayendo. Algunos algoritmos usan este enfoque de uno u otro modo: ID4 [9], ID5 [10], ITI [11] o VFDT [12]. A pesar del potencial que representa este enfoque probabilístico, los algoritmos que lo usan pueden estar limitados dependiendo del modelo de memoria [13][14] que se emplee. Por ejemplo, ID5 o ITI son algoritmos de memoria completa que almacenan todas las experiencias y, aunque usan representación probabilística, no pueden tratar conjuntos de experiencias tan grandes como ID4 o VFDT, que no usan memoria de experiencias. De este modo, los modelos sin memoria de experiencias o los de memoria parcial con mecanismos de olvido son los modelos que mejor se ajustan a nuestras necesidades.

Los métodos de aprendizaje incremental (también conocidos como métodos on-line o secuenciales) presentan ciertas características que los hacen muy apropiados para trabajar con grandes cantidades de datos. El concepto de aprendizaje incremental ha sido ampliamente usado con distintas interpretaciones, desde unas más relajadas (como la aproximación *revolucionaria* de Michalski [15]) a otras más restrictivas (como la propuesta por Polikar [16]). A pesar de estas interpretaciones, dos características destacan como propias de los algoritmos incrementales: su capacidad de incorporar nuevas experiencias a la base de conocimiento [9] y su capacidad de evolucionar esta base de conocimiento desde una estructura sencilla hacia otra más compleja [4]. En este contexto ubicaremos nuestro algoritmo.

IADEM-0 es un algoritmo incremental sin memoria de experiencias. El hecho de olvidar todas las experiencias y usar una representación probabilística permite que el algoritmo que presentamos sea capaz de trabajar sobre conjuntos de datos independientemente del tamaño. El conocimiento extraído por IADEM-0 es representado usando árboles de decisión, permitiendo así que el conocimiento adquirido se puede comprender fácilmente. Además, la inducción de árboles de decisión se considera un buen método para clasificar grandes conjuntos de datos [17].

El algoritmo que presentamos puede trabajar, al igual que BOAT [18], con cualquier medida de desorden para elegir el atributo por el que es mejor expandir un nodo. En este sentido IADEM-0 ofrece una gran capacidad de configuración, puesto que, si se conoce que alguna medida de desorden funciona mejor para determinados conjuntos de datos, basta con usarla. Una característica original de

IADEM-0 es que estima, con un soporte estadístico, el error estimado del árbol, dando así la posibilidad de fijar el error (ε) y la confianza ($1 - \delta$) que quiere alcanzarse con el árbol generado. Usamos para ello las cotas de concentración de Chernoff [19] y Hoeffding [20] [21]. Éstas han sido usadas recientemente en algunas áreas del aprendizaje automático [12] [22] [23] y proporcionan una serie de ventajas a los algoritmos que los usan. En nuestro caso las cotas nos permiten conocer los márgenes de error estadístico que han sido calculados después de procesar un determinado número de experiencias. Gracias a estos intervalos podemos calcular el error estimado del árbol con la confianza establecida. Además, para el proceso de expansión del árbol también se tienen en cuenta estas cotas.

La predicción con IADEM-0 también es algo novedoso. Los algoritmos suelen proporcionar un vector de predicción con los valores estimados. IADEM-0 también proporciona un vector con márgenes de error para cada uno de los valores del vector de predicción teniendo en cuenta la confianza requerida por el usuario. Con esta información, más completa que el simple valor estimado, se pueden diseñar funciones de predicción más complejas que el método de elegir el mayoritario.

En la Sección 2 describiremos el algoritmo IADEM-0 y a continuación, en la Sección 3, hablaremos de los resultados obtenidos después de la experimentación realizada. Para terminar, en la última sección recogeremos las conclusiones.

2. IADEM-0: Un Enfoque Descriptivo

La Figura 1 muestra el núcleo del algoritmo de una forma sencilla y fácilmente comprensible. Esta descripción a alto nivel incluye cuatro predicados (Condición_Parada, Condición_Completamente_Expandido, Condición_Expansión y Condición_Es_Expansible) y tres procedimientos (Inicialización, Muestreo_y_Recálculo y Expandir_Árbol).

```
1. Inicialización
2. mientras ¬Condición_Parada ∧ ¬Condición_Completamente_Expandido hacer:
3.          Muestrear_y_Recalcular
4.          si ¬Condición_Parada ∧ Condición_Expansión
5.          entonces si Condición_Es_Expansible(peor_nodo)
6.                  entonces Expandir_Árbol
```

Figura 1. Algoritmo IADEM-0

La salida del algoritmo será un árbol de decisión con error. Este error (*error*) será menor que el error establecido por el usuario (ε) y tendrá la confianza deseada ($1 - \delta$). Debemos hacer notar que, si el error deseado (ε) es menor que el ruido que presentan los datos o que el no determinismo del problema, el algoritmo no se detendría hasta que el árbol esté completamente expandido.

La entrada del algoritmo son experiencias extraídas del conjunto de experiencias. Estas experiencias pasan al algoritmo, son usadas y se les extrae la información necesaria, pero nunca son almacenadas en ninguna estructura. Las experiencias pueden venir de bases de datos o de flujos de datos. Si tenemos una base de datos, se realizan muestreos con reposición para extraer las experiencias que se van a usar. Si tenemos un flujo de datos, podemos usar las experiencias conforme lleguen o podemos almacenarlas para realizar después muestreos con reposición.

Antes de comentar el funcionamiento del algoritmo nombraremos los parámetros que intervienen en él:

- Factor de expansión: $\gamma \in (0, 1)$. Para determinar la frontera de expansión que se comentará más adelante.
- Medida de desorden: $medida : [0, 1]^k \rightarrow \mathbb{R}^+ \cup \{0\}$. Al poder configurar este parámetro, si conocemos que alguna medida de desorden en concreto funciona bien para el problema, la podemos usar.
- Diferenciación de atributos: $d \in (0, 1)$. Permite regular la calidad de las expansiones.
- Número de experiencias en cada muestreo: $n \in \mathbb{N}^+$. Fija el número de experiencias que se exploran en cada paso del algoritmo.

Todos los parámetros intervienen en el cálculo de ciertos valores estimados y en el de sus márgenes correspondientes. Para realizar dichos cálculos se usan las cotas de Chernoff y Hoeffding. Estas cotas tienen una propiedad bastante atractiva y es su independencia de la distribución de probabilidad que genera las experiencias mientras esta distribución de probabilidad permanezca estática durante el proceso de generación de experiencias. Un problema derivado de esta independencia es que estas cotas son bastante conservativas y requieren un mayor número de experiencias para converger (aunque esto siempre depende de los problemas particulares). Lo que conseguimos con las cotas de concentración son unas estimaciones del error cometido en diversos cálculos. Así, para estimar el error cometido por el árbol que se está generando, contamos con dos valores: su límite superior ($sup(error)$) y su límite inferior ($inf(error)$).

Cuando se inicializa el algoritmo, se ponen todas las estructuras y contadores a sus valores iniciales. El árbol de decisión tendrá un solo nodo y sus contadores estarán a cero. Una vez se ha inicializado la estructura se puede iniciar el proceso de inducción.

El funcionamiento básico del algoritmo consiste en repetir un bucle en el que se reciben conjuntos de experiencias, cuyo tamaño depende del parámetro n, y en el que se puede modificar la estructura de conocimiento que se tiene almacenada. Cuando termina la ejecución del bucle, termina también el algoritmo y el árbol inducido hasta ese momento será la salida. Existen dos razones para salir del bucle:

- satisfacer los requisitos impuestos por el usuario. Es decir, que el límite superior del error estimado del árbol que se está induciendo ($sup(error)$) sea menor que el error deseado (ε) con cierta confianza ($1 - \delta$).

- no poder expandir más el árbol. Cuando el árbol no se pueda seguir expandiendo porque no queden más nodos con atributos libres.

La primera acción dentro del bucle es recibir un nuevo conjunto de experiencias (extraído por muestreo con reposición) y realizar los cálculos necesarios. Con cada nuevo muestreo, los contadores se actualizan y determinados valores se recalculan. Es, después de cada recálculo, cuando se estudia si el conocimiento almacenado se ajusta al conocimiento procesado. En caso de que sea necesario, IADEM-0 puede intentar expandir algún nodo del árbol para ajustarse más. Antes de expandir deben darse una serie de condiciones:

- no haber satisfecho ya las exigencias del usuario.
- que la frontera inferior del error estimado ($inf(error)$) sea superior a la frontera de expansión que está determinada por el error solicitado (ε) y un parámetro (γ) usando la fórmula $(1-\gamma)\cdot\varepsilon$. El objetivo de esta frontera es evitar comportamientos asintóticos de *error* (limitado entre las cotas $inf(error)$ y $sup(error)$).
- que la expansión del nodo sea conveniente, en función de dos parámetros: medida de desorden (*medida*) y diferenciación de atributos (d). Antes de que se realice dicha expansión es necesario encontrar cuál es el nodo que introduce mayor error en el árbol (que será el que se expanda). A ese nodo lo llamamos *peor_nodo*. Una vez sea determinado, hay que encontrar para ese nodo, usando la medida de desorden (*medida*), qué atributo es el mejor para realizar la expansión. Como último paso antes de aceptar la expansión hay que asegurarse de que el mejor atributo es mejor que el resto en una distancia d (parámetro de diferenciación de atributos).

3. Resultados

Hemos realizado distintos tipos de experimentos para comprobar el comportamiento de este nuevo algoritmo. Hemos usado conjuntos de experiencias reales (extraídos del repositorio de la UCI [24]) y conjuntos generados de forma sintética por nosotros. Los experimentos se han realizado comparando los resultados obtenidos por IADEM-0 con los obtenidos por otros algoritmos que también inducen árboles de decisión y que son bastante conocidos como C4.5 o ITI. Dichos experimentos han consistido siempre en una validación cruzada por deciles (10-cross validation).

La razón de usar conjuntos de experiencias reales es que queríamos comparar nuestro algoritmo con otros considerando experimentos bastante utilizados. Los resultados indicaron que, para los experimentos realizados, el algoritmo IADEM- 0 tenía un comportamiento similar a otros algoritmos.

Pero el objetivo de un algoritmo incremental y sin memoria de experiencias, como es IADEM-0, no es tratar pequeños conjuntos de experiencias (como ocurre con los experimentos de la UCI), sino tratar grandes conjuntos de experiencias. Para estudiar el comportamiento con dichos conjuntos de datos generamos algunos experimentos. En ellos el número de experiencias era bastante

considerable (entre 1 y 10 millones). En vista de los resultados que obtuvimos, pudimos concluir que IADEM-0 siempre es capaz de inducir árboles de decisión (independientemente del tamaño del conjunto de datos) y que los que obtiene son muy buenos atendiendo a criterios de precisión y tamaño.

4. Conclusiones

Este artículo introduce brevemente el algoritmo IADEM-0: un algoritmo incremental para extraer conocimiento a partir de grandes conjuntos de datos o flujos de datos. Como núcleo del algoritmo se usan la representación probabilística y las cotas de Chernoff y Hoeffding.

Entre sus características más destacables se encuentran las siguientes:

- no hay dependencia del tamaño de la fuente de datos. Los requisitos de memoria tan sólo dependen de la estructura de conocimiento que se esté almacenando.
- en todo momento el algoritmo mantiene información del error estimado para el árbol que se está induciendo. Así puede detener su ejecución cuando se satisfagan los requisitos del usuario.
- se da una información más detallada para realizar la predicción, puesto que los valores estimados del vector de predicción van acompañados de sus respectivos márgenes de error (siempre con la confianza requerida por el usuario).

Otra característica bastante útil del algoritmo es la posibilidad de realizar una implementación distribuida. La forma en la que está diseñado el algoritmo permite que la expansión de cada nodo se realice de forma independiente y que después pueda ser integrada en el árbol.

Referencias

1. Wang, H., Fan, W., Yu, P.S., Han, J.: Mining concept-drifting data streams using ensemble classifiers. In: Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM Press (2003) 226– 235
2. Quinlan, J.R.: Learning efficient classification procedures and their application to chess end games. In: Machine Learning: An Artificial Intelligence Approach. Tioga (1983) 463– 482
3. Quinlan, J.R.: Induction of decision trees. Machine Learning **1** (1986) 81– 106
4. Quinlan, J.R.: C4.5: Programs for Machine Learning. Morgan Kaufmann (1993)
5. Mehta, M., Agrawal, R., Rissanen, J.: SLIQ: A fast scalable classifier for data mining. Lecture Notes in Computer Science **1057** (1996) 18– 32
6. Shafer, J.C., Agrawal, R., Mehta, M.: SPRINT: A scalable parallel classifier for data mining. In: Proceedings of the 22th International Conference on Very Large Data Bases, September 3–6, 1996, Mumbai (Bombay), India, Morgan Kaufmann Publishers (1996) 544– 555

7. Smith, E.E., Medin, D.L.: Categories and Concepts. Harvard University Press (1981)
8. Fisher, D.H., Schlimmer, J.C.: Models of incremental concept learning: A coupled research proposal. Technical Report CS-88-05, Vanderbilt University (1998)
9. Schlimmer, J.C., Fisher, D.H.: A case study of incremental concept induction. In: Proceedings of the 5th National Conference on Artificial Intelligence, Philadelphia, Morgan Kaufmann (1986) 496– 501
10. Utgoff, P.E.: ID5: an incremental ID3. In: Proceedings of the 5th International Conference on Machine Learning, Morgan Kaufmann (1988) 107– 120
11. Utgoff, P.E., Berkman, N.C., Clouse, J.A.: Decision tree induction based on efficient tree restructuring. Machine Learning **29** (1997) 5– 44
12. Domingos, P., Hulten, G.: Mining high-speed data streams. In: Proceedings of the 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-00), ACM Press (2000) 71–80
13. Reinke, R.E., Michalski, R.S.: Incremental learning of concept descriptions: A method and experimental results. In: Machine Intelligence. Volume 11., Oxford University Press (1988) 435– 454
14. Maloof, M.A., Michalski, R.S.: Selecting examples for partial memory learning. Machine Learning **41** (2000) 27– 52
15. Michalski, R.S.: Knowledge repair mechanisms: Evolution vs. revolution. In: Proceedings of the 3rd International Workshop on Machine Learning, Skytop, PA. (1985) 116– 119
16. Polikar, R., Udpa, L., Udpa, S., Honavar, V.: Learn++: An incremental learning algorithm for supervised neural networks. IEEE Transactions on Systems, Man, and Cybernetics **31** (2001) 497– 508
17. Perlich, C., Provost, F., Simonoff, J.S.: Tree induction vs. logistic regression: A learning-curve analysis. Journal of Machine Learning Research **4** (2003) 211– 255
18. Gehrke, J., Ganti, V., Ramakrishnan, R., Loh, W.: Boat – optimistic decision tree construction. In: Proceedings of the 1999 ACM SIGMOD International Conference on Management of Data, ACM Press (1999) 169– 180
19. Chernoff, H.: A measure of asymptotic efficiency for tests of a hypothesis based on the sums of observations. Annals of Mathematical Statistics **23** (1952) 493– 507
20. Hoeffding, W.: Probability inequalities for sums of bounded random variables. Journal of the American Statistical Association **58** (1963) 13– 30
21. Maron, O., Moore, A.W.: Hoeffding races: Accelerating model selection search for classification and function approximation. In: Advances in Neural Information Processing Systems. Volume 6., Morgan Kaufmann Publishers, Inc. (1994) 59– 66
22. Yang, J., Wang, W., Yu, P.S., Han, J.: Mining long sequential patterns in a noisy environment. In: Proceedings of the ACM SIGMOD International Conference on Management of Data, ACM Press (2002) 406– 417
23. Auer, P.: Using confidence bounds for exploitation-exploration trade-offs. Journal of Machine Learning Research **3** (2002) 397– 422
24. Blake, C., Merz, C.J.: UCI repository of machine learning databases. University of California, Department of Information and Computer Science (2000)