Divide y vencerás - clasificación mediante árboles de decisión y reglas, parte II

Comprensión de las reglas de clasificación

Las reglas de clasificación representan el conocimiento en forma de declaraciones if-else lógicas que asignan una clase a ejemplos no etiquetados. Se especifican en términos de un **antecedente** y un **consecuente**, que forman una declaración que dice que "si esto sucede, entonces eso sucede". El antecedente comprende ciertas combinaciones de valores de características, mientras que el consecuente especifica el valor de clase que se asignará si se cumplen las condiciones de la regla. Una regla simple podría decir, "si la computadora está haciendo un sonido de clic, entonces está a punto de fallar".

Los aprendices de reglas (*rule learners*) son hermanos estrechamente relacionados de los aprendices de árboles de decisión y a menudo se utilizan para tipos similares de tareas. Al igual que los árboles de decisión, se pueden utilizar para aplicaciones que generan conocimiento para acciones futuras, como:

- Identificar condiciones que conducen a fallas de hardware en dispositivos mecánicos
- Describir las características clave de grupos de personas para la segmentación de clientes
- Encontrar condiciones que preceden a grandes caídas o aumentos en los precios de las acciones en el mercado de valores

Los aprendices de reglas tienen algunos contrastes distintivos en relación con los árboles de decisión. A diferencia de un árbol, que debe seguirse a través de una serie de decisiones ramificadas, las reglas son proposiciones que pueden leerse como declaraciones independientes de hechos.

Además, por razones que se analizarán más adelante, los resultados de un aprendiz de reglas pueden ser más simples, directos y fáciles de entender que un árbol de decisiones construido sobre los mismos datos.

Es posible que ya te hayas dado cuenta de que las ramas de los árboles de decisiones son casi idénticas a las declaraciones if-else de los algoritmos de aprendizaje de reglas y, de hecho, las reglas pueden generarse a partir de árboles. Entonces, ¿por qué molestarse con un grupo separado de algoritmos de aprendizaje de reglas? Sigue leyendo para descubrir los matices que diferencian los dos enfoques.

Los aprendices de reglas generalmente se aplican a problemas donde las características son principalmente o completamente nominales. Son buenos para identificar eventos raros, incluso si el evento raro ocurre solo para una interacción muy específica entre valores de características.

Separar y conquistar

Los algoritmos de clasificación de aprendizaje de reglas utilizan una heurística conocida como **separar y conquistar**. El proceso implica identificar una regla que cubra un subconjunto de ejemplos en los datos de entrenamiento y luego separar esta partición de los datos restantes. A medida que se agregan reglas, se separan subconjuntos adicionales de datos hasta que se cubre todo el conjunto de datos y no quedan más ejemplos.

Aunque separar y vencer es en muchos sentidos similar a la heurística dividir y vencer que se trató anteriormente, difiere en formas sutiles que pronto se aclararán.

Una forma de imaginar el proceso de aprendizaje de reglas de separar y conquistar es imaginar que se profundiza en los datos creando reglas cada vez más específicas para identificar valores de clase. Supongamos que se te encomendó la tarea de crear reglas para identificar si un animal es o no un mamífero. Podrías representar el conjunto de todos los animales como un espacio grande, como se muestra en el siguiente diagrama:

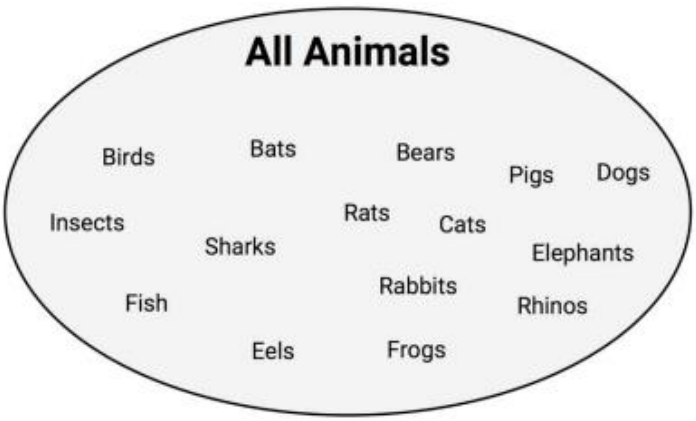

Figura 6: Un algoritmo de aprendizaje de reglas puede ayudar a dividir a los animales en grupos de mamíferos y no mamíferos.

Un aprendiz de reglas comienza utilizando las características disponibles para encontrar grupos homogéneos. Por ejemplo, utilizando una característica que indica si la especie viaja por tierra, mar o aire, la primera regla podría sugerir que todos los animales terrestres son mamíferos:

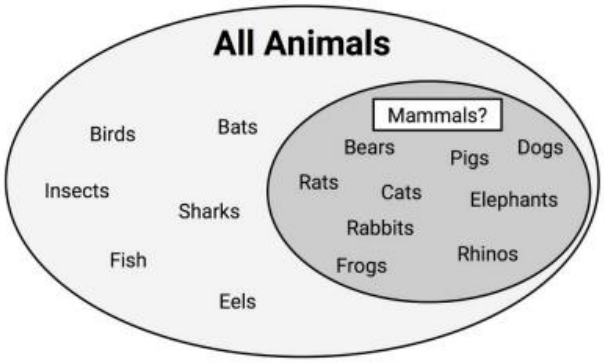

Figura 7: Una regla potencial considera a los animales que se desplazan por tierra como mamíferos.

¿Notas algún problema con esta regla? Si eres amante de los animales, es posible que te hayas dado cuenta de que las ranas son anfibios, no mamíferos. Por lo tanto, nuestra regla debe ser un poco más específica. Profundicemos más en el tema sugiriendo que los mamíferos deben caminar sobre la tierra y tener cola:

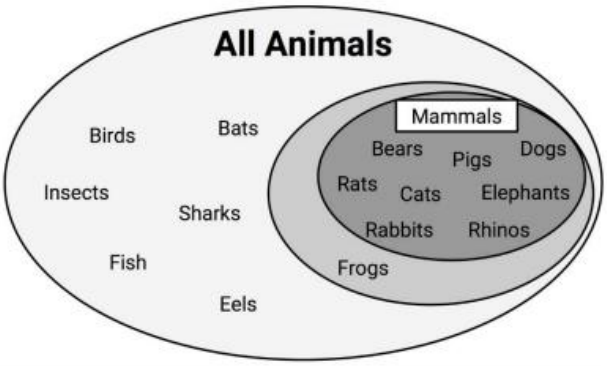

Figura 8: Una regla más específica sugiere que los animales que caminan sobre la tierra y tienen cola son mamíferos.

Como se muestra en la figura anterior, la nueva regla da como resultado un subconjunto de animales que son completamente mamíferos.

Por lo tanto, el subconjunto de mamíferos se puede separar de los demás datos y las ranas se devuelven al grupo de animales restantes (sin juego de palabras!).

Se puede definir una regla adicional para separar a los murciélagos, el único mamífero restante. Una característica potencial que distinguiría a los murciélagos de los animales restantes sería la presencia de pelo. Usando una regla construida alrededor de esta característica, hemos identificado correctamente todos los animales:

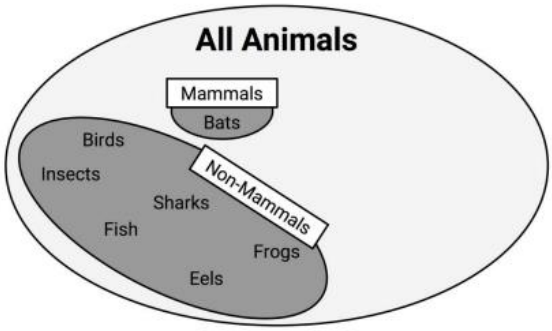

Figura 9: Una regla que establece que los animales con pelaje son mamíferos clasifica perfectamente a los animales restantes.

En este punto, dado que todas las instancias de entrenamiento han sido clasificadas, el proceso de aprendizaje de reglas se detendría. Aprendimos un total de tres reglas:

- Los animales que caminan sobre la tierra y tienen cola son mamíferos
- Si el animal no tiene pelaje, no es un mamífero
- De lo contrario, el animal es un mamífero

El ejemplo anterior ilustra cómo las reglas consumen gradualmente segmentos de datos cada vez más grandes para finalmente clasificar todas las instancias. Como las reglas parecen cubrir partes de los datos, los algoritmos de separación y conquista también se conocen como **algoritmos de cobertura**, y las reglas resultantes se denominan reglas de cobertura.

En la siguiente sección, aprenderemos cómo se aplican las reglas de cobertura en la práctica examinando un algoritmo de aprendizaje de reglas simple. Luego examinaremos un aprendiz de reglas más complejo y aplicaremos ambos algoritmos a un problema del mundo real.

El algoritmo 1R

Supongamos que un programa de juegos de televisión tiene un animal escondido detrás de una gran cortina. Se te pide que adivines si se trata de un mamífero y, si aciertas, ganas un gran premio en efectivo. No se te dan pistas sobre las características del animal, pero sabes que una porción muy pequeña de los animales del mundo son mamíferos. En consecuencia, adivina que es “no mamífero”. ¿Qué piensas sobre tus posibilidades de ganar?

Al elegir esta opción, por supuesto, maximizas tus probabilidades de ganar el premio, ya que es el resultado más probable suponiendo que el animal fue elegido al azar. Claramente, este programa de juegos es un poco ridículo, pero demuestra el clasificador más simple, **ZeroR**, que es un aprendiz de reglas que no considera ninguna característica y literalmente no aprende ninguna regla (de ahí el nombre).

Para cada ejemplo sin etiquetar, independientemente de los valores de sus características, predice la clase más común. Este algoritmo tiene muy poca utilidad en el mundo real, excepto que proporciona una línea de base simple para comparar con otros aprendices de reglas más sofisticados.

El algoritmo 1R (también conocido como **One Rule or OneR**), mejora a ZeroR al seleccionar una sola regla. Aunque esto puede parecer demasiado simplista, tiende a funcionar mejor de lo que se podría esperar.
Como se ha demostrado en estudios empíricos, la precisión de este algoritmo puede acercarse a la de algoritmos mucho más sofisticados para muchas tareas del mundo real.

Para una mirada en profundidad al sorprendente desempeño de 1R, consulta el artículo de Holte, RC. Very Simple Classification Rules Perform Well on Most Commonly Used Datasets, Machine Learning, 1993, Vol. 11, pp. 63-91.

Las fortalezas y debilidades del algoritmo 1R se muestran en la siguiente tabla:

Fortalezas	**Debilidades**
• Genera una única regla fácil de entender y legible para humanos • A menudo funciona sorprendentemente bien • Puede servir como punto de referencia para algoritmos más complejos	• Utiliza una única característica • Probablemente demasiado simplista

El funcionamiento de este algoritmo es simple. Para cada característica, 1R divide los datos en grupos con valores similares de la característica. Luego, para cada segmento, el algoritmo predice la clase mayoritaria.

Se calcula la tasa de error para la regla basada en cada característica y se elige la regla con menos errores como la única regla.

Las siguientes tablas muestran cómo funcionaría esto para los datos de animales que analizamos anteriormente:

Para la característica Viajar por, el conjunto de datos se dividió en tres grupos: Aire, Tierra y Mar. Se predijo que los animales de los grupos Aire y Mar no serían mamíferos, mientras que los animales del grupo Tierra serían mamíferos. Esto dio como resultado dos errores: murciélagos y ranas.

La función Tiene pelaje dividió a los animales en dos grupos. Se predijo que los que tenían pelaje serían mamíferos, mientras que los que no lo tenían no lo eran. Se contaron tres errores: cerdos, elefantes y rinocerontes.

Animal	Travels By	Has Fur	Mammal
Bats	Air	Yes	Yes
Bears	Land	Yes	Yes
Birds	Air	No	No
Cats	Land	Yes	Yes
Dogs	Land	Yes	Yes
Eels	Sea	No	No
Elephants	Land	No	Yes
Fish	Sea	No	No
Frogs	Land	No	No
Insects	Air	No	No
Pigs	Land	No	Yes
Rabbits	Land	Yes	Yes
Rats	Land	Yes	Yes
Rhinos	Land	No	Yes
Sharks	Sea	No	No

Conjunto completo de datos

Travels By	Predicted	Mammal
Air	No	Yes ✖
Air	No	No
Air	No	No
Land	Yes	Yes
Land	Yes	Yes
Land	Yes	Yes
Land	Yes	Yes
Land	Yes	No ✖
Land	Yes	Yes
Land	Yes	Yes
Land	Yes	Yes
Land	Yes	Yes
Sea	No	No
Sea	No	No
Sea	No	No

Regla para "Viajar Por"
Tasa de error = 2/15

Has Fur	Predicted	Mammal
No	No	No
No	No	No
No	No	Yes ✖
No	No	No
No	No	No
No	No	No
No	No	Yes ✖
No	No	Yes ✖
No	No	No
Yes	Yes	Yes
Yes	Yes	Yes
Yes	Yes	Yes
Yes	Yes	Yes
Yes	Yes	Yes
Yes	Yes	Yes

Regla para "Tiene Pelaje"
Tasa de error = 3/15

Figura 10: El algoritmo 1R elige la única regla con la tasa de clasificación errónea más baja.

Como la función Viajar por dio como resultado menos errores, el algoritmo 1R devolvería lo siguiente:

- Si el animal viaja por aire, no es un mamífero
- Si el animal viaja por tierra, es un mamífero
- Si el animal viaja por mar, no es un mamífero

El algoritmo se detiene aquí, habiendo encontrado la regla más importante.

Obviamente, este algoritmo de aprendizaje de reglas puede ser demasiado básico para algunas tareas. ¿Querrías que un sistema de diagnóstico médico considerara solo un síntoma, o que un sistema de conducción automática detuviera o acelerara tu automóvil basándose en un solo factor? Para este tipo de tareas, un aprendiz de reglas más sofisticado podría ser útil. Aprenderemos sobre uno en la siguiente sección.

El algoritmo RIPPER

Los primeros algoritmos de aprendizaje de reglas se vieron afectados por un par de problemas. En primer lugar, eran conocidos por ser lentos, lo que los hacía ineficaces para la cantidad cada vez mayor de conjuntos de datos grandes. En segundo lugar, a menudo eran propensos a ser imprecisos con datos ruidosos.

Un primer paso hacia la solución de estos problemas fue propuesto en 1994 por Johannes Furnkranz y Gerhard Widmer. Su algoritmo de poda de error reducida incremental (IREP, *Incremental Reduced Error Pruning*) utiliza una combinación de métodos de poda previa y posterior que generan reglas muy complejas y las podan antes de separar las instancias del

conjunto de datos completo. Aunque esta estrategia mejoró el rendimiento de los aprendices de reglas, los árboles de decisión a menudo seguían teniendo un mejor rendimiento.

Para obtener más información sobre IREP, consulta el artículo de Furnkranz, J y Widmer, G. Incremental Reduced Error Pruning, Proceedings of the 11th International Conference on Machine Learning, 1994, pp. 70-77.

Los aprendices de reglas dieron otro paso adelante en 1995 cuando William W. Cohen introdujo el algoritmo de poda incremental repetida para producir reducción de errores (RIPPER, *Repeated Incremental Pruning to Produce Error Reduction*), que mejoraba el IREP para generar reglas que igualaran o superaran el rendimiento de los árboles de decisión.

Para obtener más detalles sobre RIPPER, consulta el artículo de Cohen, WW. Fast Effective Rule Induction, Proceedings of the 12th International Conference on Machine Learning, 1995, pp. 115-123.

Como se describe en la siguiente tabla, las fortalezas y debilidades de RIPPER son generalmente comparables a las de los árboles de decisión. El principal beneficio es que pueden dar como resultado un modelo ligeramente más parsimonioso.

Fortalezas	**Debilidades**
• Genera reglas fáciles de entender y legibles para humanos • Es eficiente en conjuntos de datos grandes y ruidosos • Generalmente, produce un modelo más simple que un árbol de decisiones comparable	• Puede generar reglas que parecen desafiar el sentido común o el conocimiento de los expertos • No es ideal para trabajar con datos numéricos • Puede no funcionar tan bien como modelos más complejos

El algoritmo RIPPER, que evolucionó a partir de varias iteraciones de algoritmos de aprendizaje de reglas, es un mosaico de heurísticas eficientes para el aprendizaje de reglas. Debido a su complejidad, una discusión de los detalles de implementación está fuera del alcance de este curso. Sin embargo, se puede entender en términos generales como un proceso de tres pasos:

1. Crecimiento
2. Podar
3. Optimizar

La fase de crecimiento utiliza la técnica de separar y conquistar para agregar condiciones a una regla con ‘avidez’ (greedy) hasta que clasifique perfectamente un subconjunto de datos o se quede sin atributos para dividir. Al igual que los árboles de decisión, el criterio de ganancia de información se utiliza para identificar el siguiente atributo de división. Cuando

aumentar la especificidad de una regla ya no reduce la entropía, la regla se poda inmediatamente. Los pasos uno y dos se repiten hasta llegar a un criterio de detención, momento en el que se optimiza todo el conjunto de reglas utilizando una variedad de heurísticas.

El algoritmo RIPPER puede crear reglas mucho más complejas que el algoritmo 1R, ya que puede considerar más de una característica. Esto significa que puede crear reglas con múltiples antecedentes como “si un animal vuela y tiene pelo, entonces es un mamífero”. Esto mejora la capacidad del algoritmo para modelar datos complejos, pero al igual que los árboles de decisión, significa que las reglas pueden volverse rápidamente difíciles de comprender.

La evolución de los aprendices de reglas de clasificación no se detuvo con RIPPER. Se están proponiendo nuevos algoritmos de aprendizaje de reglas rápidamente. Un estudio de la literatura muestra algoritmos llamados IREP++, SLIPPER, TRIPPER, entre muchos otros.

Reglas de árboles de decisión

Las reglas de clasificación también se pueden obtener directamente de los árboles de decisión. Comenzando en un nodo de hoja y siguiendo las ramas hasta la raíz, se obtiene una serie de decisiones. Estas reglas se pueden combinar en una sola regla. La siguiente figura muestra cómo se pueden construir reglas a partir del árbol de decisiones para predecir el éxito de una película:

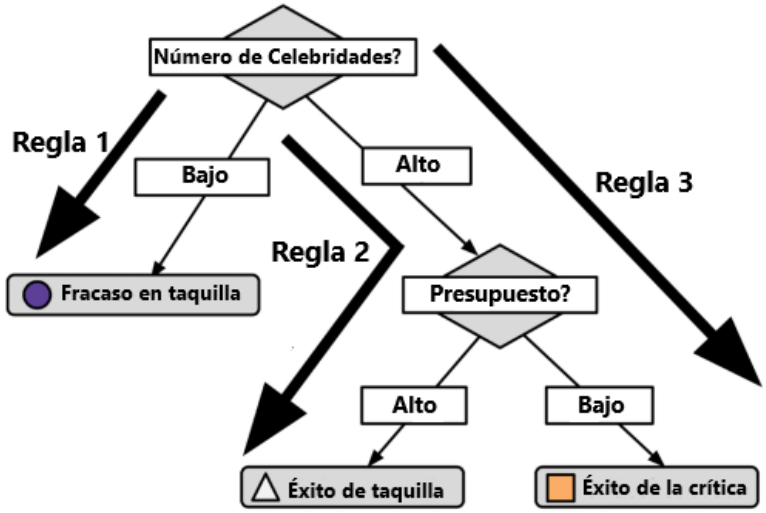

Figura 5.11: Se pueden generar reglas a partir de árboles de decisiones siguiendo las rutas desde el nodo raíz hasta cada nodo hoja.

Siguiendo las rutas desde el nodo raíz hasta cada hoja, las reglas serían:

1. Si el número de celebridades es bajo, entonces la película será un fracaso de taquilla
2. Si el número de celebridades es alto y el presupuesto es alto, entonces la película será un éxito de taquilla
3. Si el número de celebridades es alto y el presupuesto es bajo, entonces la película será un éxito de crítica

Por razones que se explicarán en la siguiente sección, la principal desventaja de usar un árbol de decisiones para generar reglas es que las reglas resultantes suelen ser más complejas que las aprendidas por un algoritmo de aprendizaje de reglas.

La estrategia de dividir y conquistar empleada por los árboles de decisiones sesga los resultados de manera diferente a la de un algoritmo de aprendizaje de reglas. Por otro lado, a veces es más eficiente computacionalmente generar reglas a partir de árboles.

La función C5.0() del paquete C50 generará un modelo utilizando reglas de clasificación si especificas rules = TRUE al entrenar el modelo.

¿Qué hace que los árboles y las reglas sean glotones (greedy)?

Los árboles de decisión y los aprendices de reglas se conocen como **aprendices codiciosos** (avariciosos, glotones) porque utilizan los datos por orden de llegada. Tanto la heurística de dividir y conquistar que utilizan los árboles de decisión como la heurística de separar y conquistar que utilizan los aprendices de reglas intentan hacer particiones de a una por vez, encontrando primero la partición más homogénea, seguida de la siguiente mejor, y así sucesivamente, hasta que se hayan clasificado todos los ejemplos.

La desventaja del enfoque codicioso es que no se garantiza que los algoritmos codiciosos generen la cantidad óptima, más precisa o menor de reglas para un conjunto de datos en particular.

Al elegir la fruta más fácil al principio, un aprendiz codicioso puede encontrar rápidamente una sola regla que sea precisa para un subconjunto de datos; sin embargo, al hacerlo, el aprendiz puede perder la oportunidad de desarrollar un conjunto de reglas más matizado con una mejor precisión general en todo el conjunto de datos.

Sin embargo, sin utilizar el enfoque voraz para el aprendizaje de reglas, es probable que, para todos los conjuntos de datos, excepto los más pequeños, el aprendizaje de reglas sea computacionalmente inviable.

Figura 5.12: Tanto los árboles de decisión como los aprendices de reglas de clasificación son algoritmos voraces.

Aunque tanto los árboles como las reglas emplean heurísticas de aprendizaje voraz, existen diferencias sutiles en la forma en que construyen reglas. Tal vez la mejor manera de distinguirlos es notar que una vez que divide y conquista divide en una característica, las particiones creadas por la división no pueden ser reconquistadas, solo subdivididas. De esta manera, un árbol está limitado permanentemente por su historial de decisiones pasadas.

Por el contrario, una vez que separa y conquista encuentra una regla, cualquier ejemplo que no esté cubierto por todas las condiciones de la regla puede ser reconquistado.

Para ilustrar este contraste, considera el caso anterior en el que construimos un aprendiz de reglas para determinar si un animal era un mamífero. El aprendiz de reglas identificó tres reglas que clasifican perfectamente a los animales del ejemplo:

1. Los animales que caminan sobre la tierra y tienen cola son mamíferos (osos, gatos, perros, elefantes, cerdos, conejos, ratas y rinocerontes)
2. Si el animal no tiene pelo, no es un mamífero (pájaros, anguilas, peces, ranas, insectos y tiburones)
3. De lo contrario, el animal es un mamífero (murciélagos)

En cambio, un árbol de decisión construido sobre los mismos datos podría haber llegado a cuatro reglas para lograr la misma clasificación perfecta:

1. Si un animal camina sobre la tierra y tiene cola, entonces es un mamífero (osos, gatos, perros, elefantes, cerdos, conejos, ratas y rinocerontes)
2. Si un animal camina sobre la tierra y no tiene cola, entonces no es un mamífero (ranas)
3. Si el animal no camina sobre la tierra y tiene pelo, entonces es un mamífero (murciélagos)

4. Si el animal no camina sobre la tierra y no tiene pelo, entonces no es un mamífero (pájaros, insectos, tiburones, peces y anguilas)

El resultado diferente en estos dos enfoques tiene que ver con lo que les sucede a las ranas después de que se las separa por la decisión de "caminar sobre la tierra". Cuando el aprendiz de reglas permite que las ranas sean reconquistadas por la decisión de "no tener pelo", el árbol de decisión no puede modificar las particiones existentes y, por lo tanto, debe colocar a la rana en su propia regla.

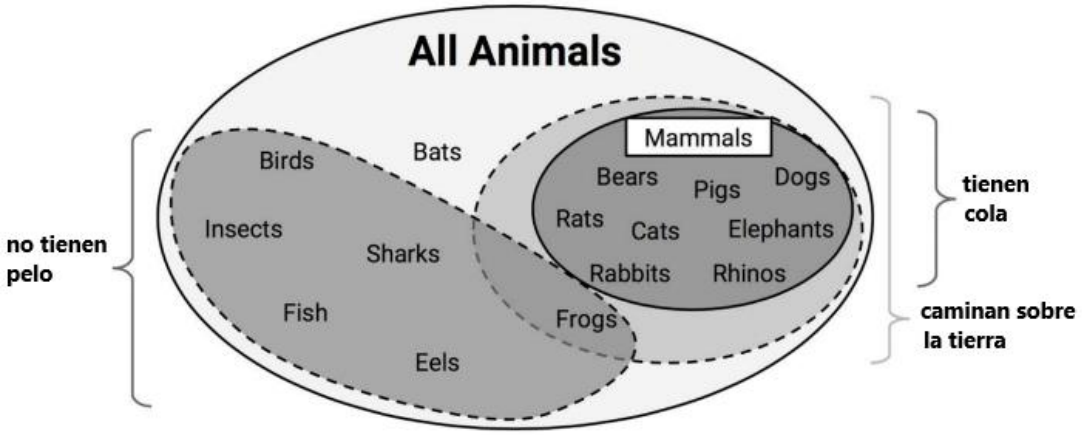

Figura 5.13: El manejo de las ranas distingue las heurísticas de dividir y conquistar y separar y conquistar. El último enfoque permite que las ranas sean reconquistadas por reglas posteriores.

Por un lado, debido a que los aprendices de reglas pueden reexaminar casos que se consideraron pero que finalmente no se cubrieron como parte de reglas anteriores, los aprendices de reglas a menudo encuentran un conjunto de reglas más parsimonioso que los generados por árboles de decisión. Por otro lado, esta reutilización de datos significa que el costo computacional de los aprendices de reglas puede ser algo más alto que para los árboles de decisión.