Parte III: La salida (Representación del Conocimiento)

Jhon J. Padilla

Introducción

- Antes de entender cómo funcionan las técnicas para generar el conocimiento a partir de los datos, es importante entender cómo se puede representar el conocimiento en forma de patrones estructurales.

1. Arboles de Decisión

- Es una forma de aprendizaje que utiliza la aproximación de “Divide y Vencerás”.
- Cada nodo en un árbol de decisión prueba un atributo particular.
- Usualmente se compara el valor de un atributo con una constante. Sin embargo, se pueden comparar dos atributos entre sí, o una comparación con una función de varios atributos.

Arboles de Decisión: Reglas

- Las hojas del árbol dan una clasificación que aplica para todas las instancias que alcanzan esa hoja.
- Para clasificar una entrada particular, se enruta hacia abajo por el árbol de acuerdo a los valores de los atributos probados en nodos sucesivos, hasta que alcanza una hoja y puede ser asignada a la clase de esa hoja.

Arboles de Decisión: Reglas

- Si el atributo a probar es nominal, el número de hijos es usualmente el número de posibles valores del atributo. Es decir, hay una ramificación por cada posible valor, por tanto, el mismo atributo no será probado posteriormente en el árbol.
- Algunas veces los valores del atributo se dividen en dos subconjuntos, por lo que en ramificaciones siguientes se podría probar el mismo atributo para otros valores dentro del subconjunto.

Arboles de Decisión: Reglas

- Si el atributo es numérico, la prueba en un nodo determina si su valor es mayor o menor que una constante predeterminada, dando una división en dos ramas.
- Si se trata un valor perdido, se puede crear una tercera ramificación

Valores perdidos

- Los valores perdidos pueden ser tratados en una forma especial en lugar de ser considerados como otro posible valor que el atributo podría tomar.
- Una solución simple es almacenar el número de elementos que en el conjunto de entrenamiento se fueron por cada rama y usar la rama más popular si el valor para una prueba está perdido.

Valores perdidos

- Una solución más sofisticada es dividir la instancia en piezas y enviar cada pieza por la rama adecuada en el árbol.
- La división está acompañada usando un peso numérico entre 0 y 1, y el peso para una rama es elegido de manera proporcional al número de instancias de entrenamiento que se fueron por esa rama. Todos los pesos deben sumar 1.

Valores perdidos

- Una instancia con un peso podría ser posteriormente dividida en un nodo inferior. Por tanto, las diferentes partes de la instancia alcanzarán un nodo hoja, y las decisiones en estos nodos hojas deben ser recombinadas usando los pesos que han usado al bajar hacia las hojas.

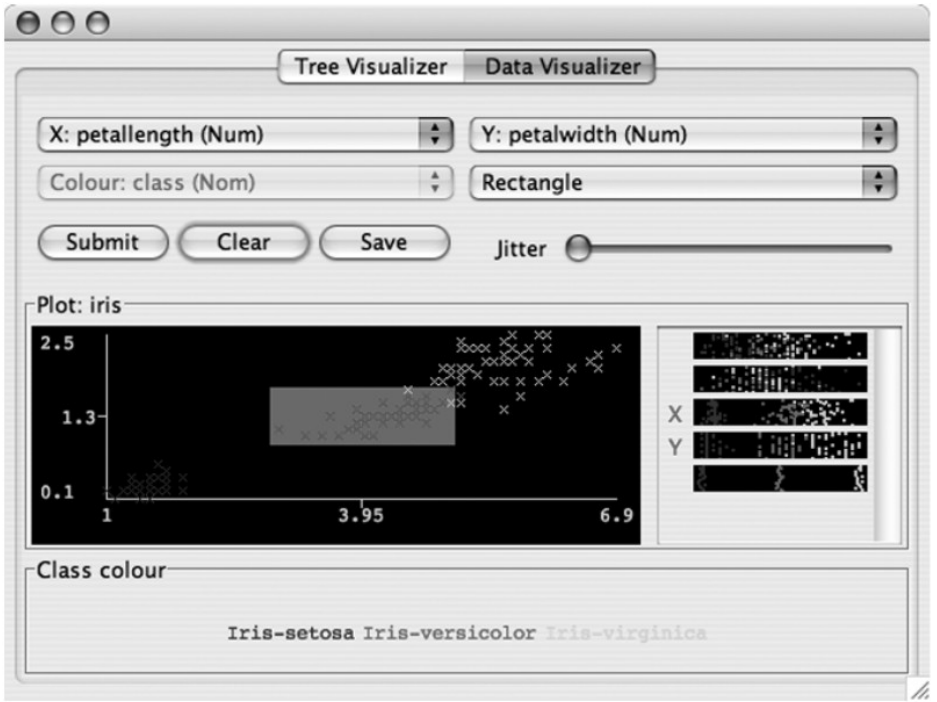
Ejemplo

- Es instructivo construir un árbol de decisiones para un conjunto de datos de forma manual.
- Para hacerlo, usted necesita una buena forma de visualizar los datos para que usted pueda decidir cuáles son los mejores atributos a probar y, qué tan apropiada puede ser una prueba.

Explorador Weka

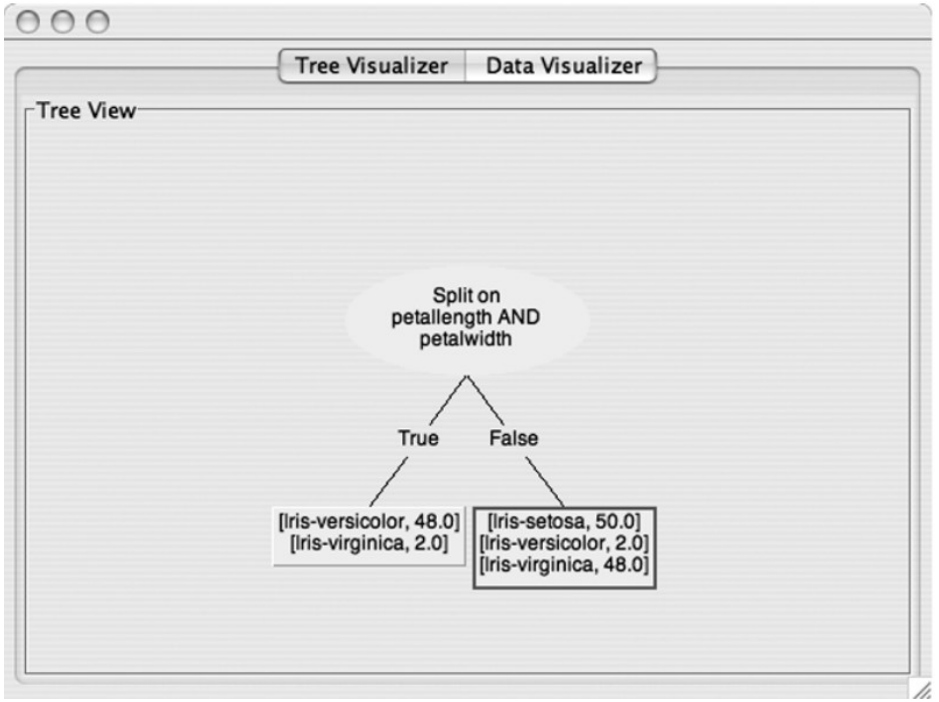
- El explorador Weka tiene una facilidad de clasificación para los usuarios, que les permite construir un árbol de decisión interactivamente.
- Este muestra un gráfico de puntos de los datos contra dos atributos de su elección.
- Cuando usted encuentra un par de atributos que discrimina las clases bien, usted puede crear una división para trazar un polígono alrededor de los puntos de datos apropiados en el gráfico de puntos.

Explorador Weka-Clasificación



- El usuario tiene un conjunto de datos (iris dataset) con 3 clases
- Se usaron dos atributos: longitud y ancho de pétalos, que permiten una buena división en clases.
- Se pudo trazar un rectángulo manualmente para separar una de las clases (iris versicolor)

Explorador Weka-Clasificación



- El usuario puede visualizar el árbol de decisión para verlo detenidamente.
- La rama izquierda contiene predominantemente plantas de un tipo: iris versicolor, contaminada solamente por dos virginicas.
- La rama derecha contiene predominantemente dos tipos: iris setosa y virgínica, contaminadas solamente por dos versicolor).
- El usuario probablemente seleccionará la rama derecha para subdividirla luego basado en otro rectángulo con otros dos atributos, aunque los dos actuales lucen bastante bien.

2. Reglas de Clasificación

- Son una alternativa popular a los árboles de decisión.
- El antecedente, o precondición, de una regla es una serie de pruebas, igual que las pruebas de los nodos en los árboles de decisión.
- Como consecuencia, o conclusión, se obtiene la clase o clases que aplican a instancias cubiertas por esa regla. También puede dar una distribución de probabilidad sobre las clases.

Reglas de Clasificación

- Generalmente, las precondiciones están unidas por AND, y todas las pruebas deben ser exitosas si la regla está encendida.
- Sin embargo, para algunas formulaciones de reglas, las precondiciones son expresiones lógicas generales en lugar de simples conjunciones.
- A menudo pensamos en las reglas individuales como si estuvieran unidas por OR: si alguna de ellas aplica, la clase dada en su conclusión es aplicada a la instancia.
- Sin embargo, ocurren conflictos cuando aparecen diferentes reglas con diferentes conclusiones.

Ventajas/Desventajas de un árbol de decisión

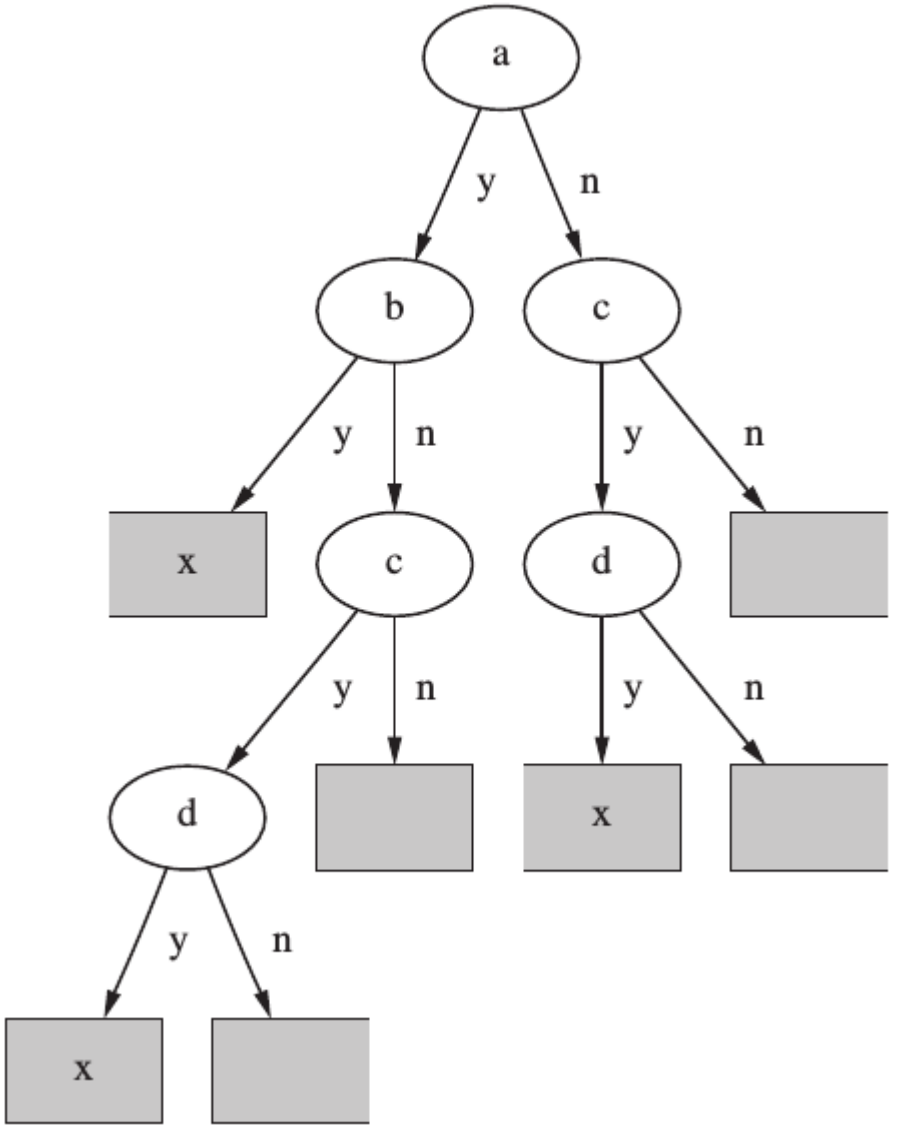
- Es fácil leer directamente un conjunto de reglas de un árbol de decisiones. Para cada hoja se genera una regla, y las reglas no son ambiguas, por lo que el orden en que se ejecuten es irrelevante.
- Sin embargo, las reglas que se obtienen de un árbol de decisión son más complejas de lo necesario, y por lo general se deben probar para eliminar pruebas redundantes

Comparación reglas vs. árbol

- Debido a que los árboles de decisión no pueden expresar la disyunción implicada en un conjunto de reglas, transformar un conjunto de reglas en un árbol no es directo.
- Un ejemplo de esto es cuando las reglas tienen la misma estructura pero diferentes atributos.

Comparación reglas vs. árbol

```
If a and b then x
If c and d then x
```



Ventajas de las reglas de clasificación

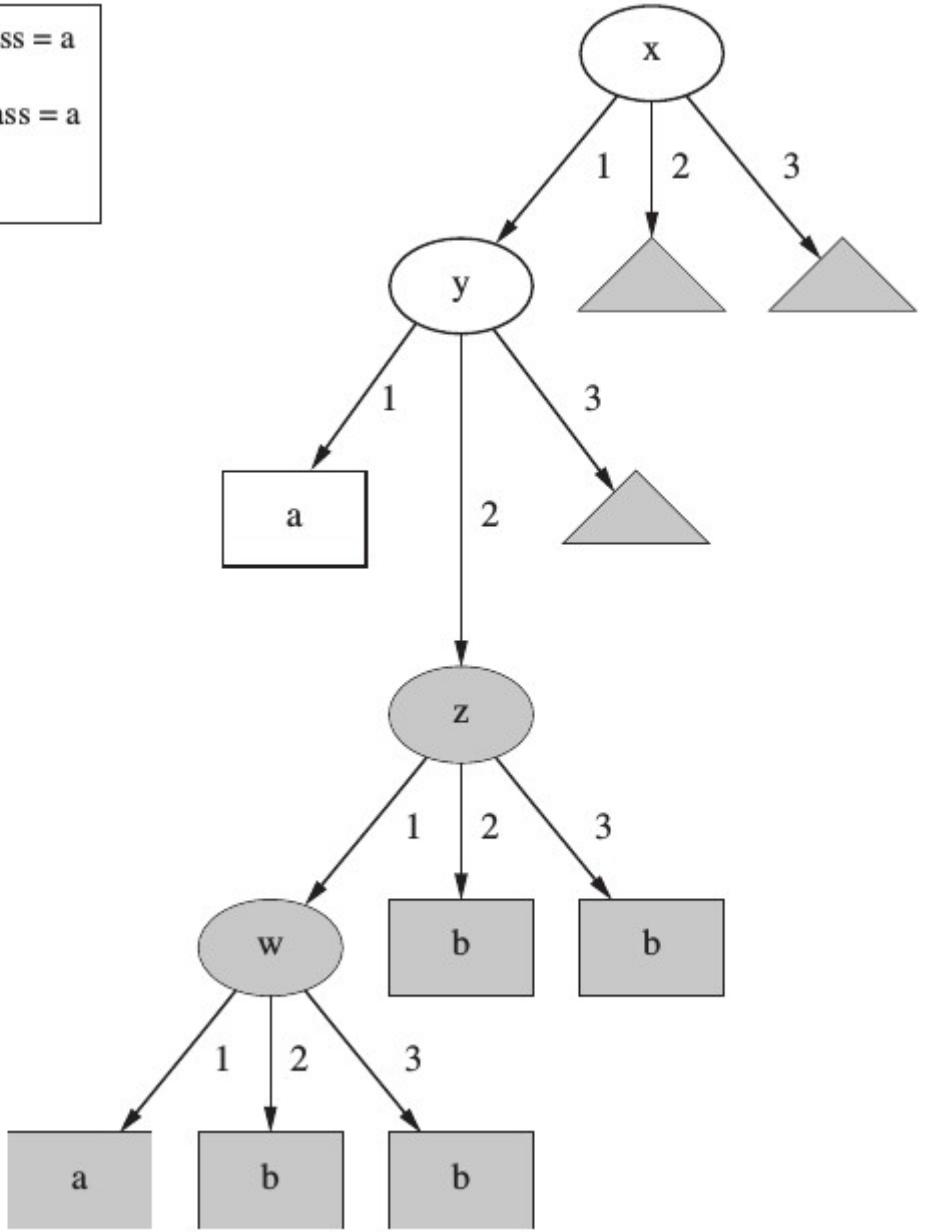
- Las reglas de clasificación suelen ser más compactas que los árboles.
- Suele ser útil tener una regla por defecto que cubra los casos no especificados por otras reglas.

Comparación reglas vs. árboles

If x=1 and y=1 then class = a

If z=1 and w=1 then class = a

Otherwise class = b



Ventajas de las reglas

- Una razón por la que las reglas son populares es que cada regla representa una píldora de conocimiento.
- Pueden agregarse nuevas reglas a un conjunto de reglas existente sin perturbas las que ya están, mientras que agregar una regla a un árbol puede requerir redibujar el árbol completo.

- Sin embargo, esta independencia es una ilusión, porque se ignora cómo se ejecuta el conjunto de reglas.
- Las reglas deben ejecutarse en orden de aparición como una lista de decisión, sino, podrían ser interpretadas incorrectamente si están fuera de contexto.
- Las reglas que se deducen de un árbol de decision no tienen este problema porque contemplan la redundancia y previenen ambigüedad en la interpretación.

Multiples clasificaciones

- Si una regla da múltiples clasificaciones para un ejemplo particular, una solución es no dar una conclusión.
- Otra solución es contar cuán frecuente se dispara una regla en los datos de entrenamiento y enviar dicha entrada a la más popular.
- Estas estrategias dan resultados radicalmente diferentes.

Instancia que no tiene clasificación

- Esto no puede ocurrir con los árboles de decisión, o con reglas leídas directamente de ellos, pero puede ocurrir con conjuntos de reglas generales.
- Una solución es sencillamente decir que se falló con esa entrada.
- Otra solución es enviarla a la clase más frecuentemente usada.
- Estas estrategias dan resultados radicalmente diferentes.

Resultados booleanos

- Una situación particular ocurre cuando las reglas llevan a una clase que es booleana (sí, o no), y cuando solo se expresan reglas que llevan a una sola salida: sí.
- El supuesto es que si una instancia particular no está en la clase Sí, entonces debe estar en la clase No. Si este es el caso, las reglas no pueden tener conflicto, no hay ambigüedad.
- Estas reglas pueden escribirse como una expresión lógica que es llamada forma normal disyuntiva: es decir, como una OR (disyunción) de condiciones AND (conjunción).

Advertencia

- Este caso especial seduce la gente para asumir reglas que son fáciles de obtener, porque cada regla realmente opera como una pieza nueva e independiente de información que contribuye directamente a la disyunción.
- Desafortunadamente, esto sólo aplica para salidas booleanas y requiere de un supuesto de mundo cerrado. Estos dos supuestos son poco realistas en la mayoría de situaciones.
- Los algoritmos de Machine Learning que generan reglas, producen conjuntos de reglas ordenadasen situaciones multiclase, y esto sacrifica cualquier posibilidad de modularidad porque el orden de ejecución es crítico.

4. Reglas de Asociación

- Realmente no son diferentes de las reglas de clasificación, excepto que pueden predecir cualquier atributo, no solo la clase.
- Esto da libertad para predecir también combinaciones de atributos.
- Las reglas de asociación no pretenden usarse como un conjunto (las reglas de clasificación sí deben usarse como conjunto)

Reglas de Asociación

- Diferentes reglas de asociación expresan diferentes regularidades que subyacen en el conjunto de datos.
- Generalmente predicen diferentes cosas.
- Se pueden generar muchas reglas de asociación de conjuntos de datos pequeños, por lo que solo interesan las que pueden aplicarse a un número grande de entradas y tienen una alta precisión en las instancias en las que se aplican.

Cobertura y Confianza

- La ***cobertura*** de una regla de asociación es el número de instancias para las cuales hace una predicción correcta. También se le llama el **soporte**.
- Su ***exactitud*** (o confianza) es el número de instancias que ella predice correctamente, expresada como una proporción de todas las instancias a las cuales se aplica.

Ejemplo

- Para la regla:
  - If Temperatura = fresco Then humedad = normal
- La cobertura es el número de dias que son ambos: frescos y con humedad normal (4 días en la tabla 1.2)
- La exactitud es la proporción de días frescos que tienen humedad normal (100% en este caso: 4 de 4).
- Es usual especificar los valores mínimos de cobertura y exactitud, y buscar sólo aquellas reglas cuya cobertura y exactitud cumplen ambas con sus mínimos.
- En los datos del clima, por ejemplo, hay 58 reglas cuya cobertura y exactitud son al menos 2 y 95% respectivamente.
- Puede ser conveniente expresar la cobertura como un porcentaje del número total de instancias.

Datos del Clima

Table 1.2 The weather data.

Outlook	Temperature	Humidity	Windy	Play
sunny	hot	high	false	no
sunny	hot	high	true	no
overcast	hot	high	false	yes
rainy	mild	high	false	yes
rainy	cool	normal	false	yes
rainy	cool	normal	true	no
overcast	cool	normal	true	yes
sunny	mild	high	false	no
sunny	cool	normal	false	yes
rainy	mild	normal	false	yes
sunny	mild	normal	true	yes
overcast	mild	high	true	yes
overcast	hot	normal	false	yes
rainy	mild	high	true	no

Precaución

- Las reglas de asociación que predicen múltiples consecuencias, deben ser interpretadas cuidadosamente.
- En la tabla 1.2 aparece esta regla:
  - If windy = false and play = no then outlook = sunny and humidity = high
- Esta es una expresión corta para las dos reglas separadas:
  - If windy=false and play=no then outlook= sunny
  - If windy=false and play=no then humidity=high

Precaución

- Esto implica que estos exceden las cifras mínimas de cobertura y precisión, pero también implica más.
- La regla original significa que el número de ejemplos que son nonwindy, nonplaying, con sunny outlook y high humidity, es al menos tan grande como el mínimo de cobertura especificado.
- Esto también significa que el número de tales días, expresado como una proporción de días nonwindy, nonplaying, es al menos la exactitud mínima.

Precaución

- Esto implica que la regla:
  - If humidity = high and windy = false and play = no

    then outlook = sunny
- También es válida, porque tiene la misma cobertura que la regla original, y su exactitud debe ser al menos tan grande como la de la regla original porque el número de días highhumidity, nonwindy, nonplaying, es necesariamente menor que el número de días nonwindy, nonplaying, lo cual hace que la exactitud sea mayor.

Concluyendo

- Hay relaciones entre reglas de asociación particulares, algunas reglas implican otras.
- Para reducir el número de reglas que se producen, en casos donde hay varias reglas relacionadas, toma sentido presentar solo la más fuerte al usuario.
- En el ejemplo anterior, sólo la primera regla debe ser impresa.

5. Reglas de Clasificación con excepciones

- Se pueden hacer modificaciones incrementales a un conjunto de reglas por medio de excepciones a reglas existentes en lugar de volver a crear un conjunto nuevo.

Ejemplo

- Consideremos el problema del iris descrito anteriormente.
- Suponga que se encontró una nueva flor con las dimensiones dadas en la tabla 3.1, y un experto declaró que es una instancia de Iris setosa.

Table 3.1 A new iris flower.

Sepal length (cm)	Sepal width (cm)	Petal length (cm)	Petal width (cm)	Type
5.1	3.5	2.6	0.2	?

Datos de la Flor Iris

Table 1.4 The iris data.

	Sepal length (cm)	Sepal width (cm)	Petal length (cm)	Petal width (cm)	Type
1	5.1	3.5	1.4	0.2	*Iris setosa*
2	4.9	3.0	1.4	0.2	*Iris setosa*
3	4.7	3.2	1.3	0.2	*Iris setosa*
4	4.6	3.1	1.5	0.2	*Iris setosa*
5	5.0	3.6	1.4	0.2	*Iris setosa*
. . .					
51	7.0	3.2	4.7	1.4	*Iris versicolor*
52	6.4	3.2	4.5	1.5	*Iris versicolor*
53	6.9	3.1	4.9	1.5	*Iris versicolor*
54	5.5	2.3	4.0	1.3	*Iris versicolor*
55	6.5	2.8	4.6	1.5	*Iris versicolor*
. . .					
101	6.3	3.3	6.0	2.5	*Iris virginica*
102	5.8	2.7	5.1	1.9	*Iris virginica*
103	7.1	3.0	5.9	2.1	*Iris virginica*
104	6.3	2.9	5.6	1.8	*Iris virginica*
105	6.5	3.0	5.8	2.2	*Iris virginica*
. . .					

Reglas de clasificación obtenidas para Iris

If sepal width < 2.55 and petal length < 4.95 and
   petal width < 1.55 then Iris versicolor
If petal length ≥ 2.45 and petal length < 4.95 and
   petal width < 1.55 then Iris versicolor
If sepal length ≥ 6.55 and petal length < 5.05 then Iris versicolor
If sepal width < 2.75 and petal width < 1.65 and
   sepal length < 6.05 then Iris versicolor
If sepal length ≥ 5.85 and sepal length < 5.95 and
   petal length < 4.85 then Iris versicolor
If petal length ≥ 5.15 then Iris virginica
If petal width ≥ 1.85 then Iris virginica
If petal width ≥ 1.75 and sepal width < 3.05 then Iris virginica
If petal length ≥ 4.95 and petal width < 1.55 then Iris virginica

Ejemplo

- Si la flor fue clasificada por las reglas dadas en el capitulo 1 para este problema, podría ser clasificada erróneamente por dos de ellas:

```
If petal length ≥ 2.45 and petal length < 4.45 then Iris versicolor
If petal length ≥ 2.45 and petal length < 4.95 and
   petal width < 1.55 then Iris versicolor
```

  Estas reglas requieren una modificación para que la nueva instancia puede ser tratada correctamente.

- Cambiar los límites de los atributos en las reglas no es suficiente porque se clasificarían mal las instancias originales con que se creó la regla.
- Por tanto, modificar una regla no es tan simple.

Ejemplo

- En lugar de cambiar los límites de las reglas existentes, se podría consultar un experto para explicar por qué las nuevas floras violan estos límites, recibiendo explicaciones que podrían ser usadas para extender las reglas relevantes solamente.
- Por ejemplo, la primera de estas dos reglas clasifica mal la nueva instancia de iris setosa como iris versicolor. En lugar de alterar los límites en la regla, puede usarse una excepción basada en algún otro atributo:

```
If petal length ≥ 2.45 and petal length < 4.45 then
   Iris versicolor EXCEPT if petal width < 1.0 then Iris setosa
```

Ejemplo

```
If petal length ≥ 2.45 and petal length < 4.45 then
   Iris versicolor EXCEPT if petal width < 1.0 then Iris setosa
```

- Esta regla dice que una flor es Iris versicolor si la longitud del pétalo está entre 2.45cm y 4.45cm, excepto cuando su ancho de pétalo es menor que 1 cm, en cuyo caso es Iris setosa.

Precaución

- Por supuesto, podríamos tener excepciones a las excepciones y así sucesivamente, haciendo que el conjunto de reglas tome la forma de un árbol.

Reglas expresadas como excepciones

```
Default: Iris-setosa                                                    1
except if petal-length ≥ 2.45 and petal-length < 5.355                  2
          and petal-width < 1.75                                        3
       then Iris-versicolor                                             4
            except if petal-length ≥ 4.95 and petal-width < 1.55        5
                   then Iris-virginica                                  6
                   else if sepal-length < 4.95 and sepal-width ≥ 2.45   7
                        then Iris-virginica                             8
       else if petal-length ≥ 3.35                                      9
            then Iris-virginica                                         10
                 except if petal-length < 4.85 and sepal-length < 5.95  11
                        then Iris-versicolor                            12
```

- Las reglas de la figura clasifican correctamente todos los ejemplos del conjunto de datos de iris dado inicialmente.

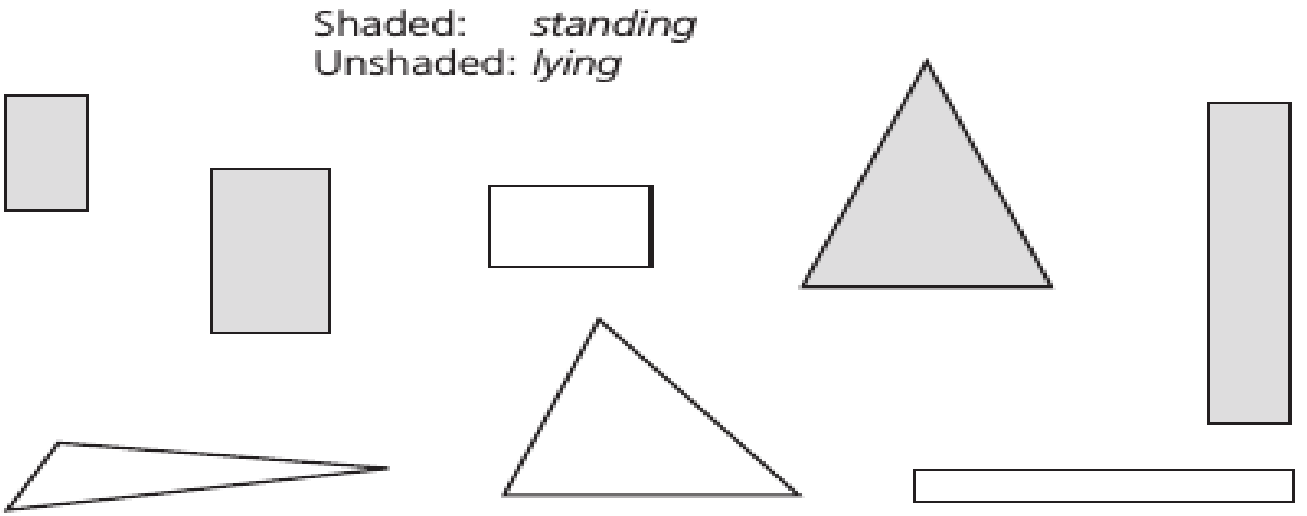
Reglas expresadas como excepciones

- Las reglas de clasificación expresadas como excepciones, pueden ser fácilmente pasadas a términos de if...then...else
- Lo que se gana con la formulación en términos de excepciones no es en la forma lógica, sino en términos psicológicos.
- Se asume que los valores por defecto y las pruebas que se hacen primero aplican más ampliamente que las excepciones que van apareciendo más abajo.

6. Reglas que involucran Relaciones

- Hasta ahora hemos asumido que las condiciones en las reglas involucran pruebas de atributos contra valores constantes.
- Estas reglas se conocen como ***reglas proposicionales*** porque el lenguaje usado para definirlas tiene el mismo poder que en lógica se llama cálculo proposicional.
- En muchas tareas de clasificación, las reglas proposicionales son suficientemente expresivas y aproximadas a las descripciones de los conceptos.
- Sin embargo, hay otras situaciones donde se requiere de encontrar relaciones entre las instancias.

Ejemplo



- Suponga que tenemos un conjunto de ocho bloques de construcción de varias formas y tamaños, como se observa en la figura.
- Se desea aprender el concepto de “parado”
- Este es un clásico problema con dos clases: “parado” (figuras grises) y “recostado” (figuras blancas)

Ejemplo

Table 3.2 Training data for the shapes problem.

Width	Height	Sides	Class
2	4	4	standing
3	6	4	standing
4	3	4	lying
7	8	3	standing
7	6	3	lying
2	9	4	standing
9	1	4	lying
10	2	3	lying

- Para aprender el concepto se tienen los datos de entrenamiento de la tabla: ancho, altura, número de lados y la clase a la que pertenece (parado, recostado)

Ejemplo

- Una regla proposicional que puede salir de esta tabla es:
  - if width ≥ 3.5 and height < 7.0 then lying
  - if height ≥ 3.5 then standing
- Se escogió el valor de 3.5 como límite para el ancho porque está a mitad entre el ancho de los bloques recostados más delgados, (es decir, 4), y el ancho de los bloques parados más gordos cuya altura es menor que 7, (es decir, 3)
- 7 es el límite para la altura porque está a mitad de camino entre la altura de los bloques recostados más altos (6), y los bloques parados más cortos cuyo ancho es mayor que 3.5 (es decir, 8).

Ejemplo

- Es común ubicar umbrales numéricos a mitad entre los valores que delimitan los límites de un concepto.
- Aunque estas dos reglas trabajan bien con los ejemplos dados, ellas no son muy buenas porque nuevos bloques podrían no quedar bien clasificados (ej. ancho=1 y altura=2).
- Una persona que clasifica los 8 bloques probablemente notará que “los bloques parados son aquellos que son más altos que anchos”.
- Esta regla no compara valores de atributos con constantes, sino que compara atributos entre sí, quedando expresadas como:
  - if width > height then lying
  - if height > width then standing

Reglas relacionales

- Los valores de los atributos ancho y altura no son importantes; sólo el resultado de la comparación de los dos.
- Las reglas que tienen esta forma se llaman ***reglas relacionales***, porque ellas expresan relaciones entre los atributos.

- Aunque las comparaciones entre atributos podrían ubicarse como nodos en árboles de decisión, suele ser común expresar estas comparaciones entre atributos mediante reglas.
- Sin embargo, la mayoría de métodos de Machine Learning no consideran las reglas relacionales porque hay un considerable costo en hacerlo.

- Una manera de permitir un método proposicional para hacer uso de relaciones es adicionar atributos secundarios que dicen si los atributos primarios son iguales o no, o dada una diferencia entre ellos si son numéricos.
- P.ej., podríamos agregar un atributo binario: *es ancho<alto?* A la tabla 3.2.
- Tales atributos se agregan a menudo como parte de un proceso de ingeniería de datos.

Reglas como programas lógicos

- Se puede usar un truco aparentemente simple, pero potente: expresar las reglas de forma que cumplen el rol de la instancia explícita:
  - if width(block) > height(block) then lying(block)
  - if height(block) > width(block) then standing(block)
- Aunque esto no se ve como una extensión, lo es si las instancias pueden ser descompuestas en partes.
- P.ej. Si una torre es una pila de bloques, uno sobre otro, el hecho de que el bloque de más arriba esté parado puede expresarse como:
  - if height(tower.top) > width(tower.top) then standing(tower.top)

Programas lógicos

- Este tipo de reglas, se pueden aplicar sucesivamente sobre partes constitutivas en escalas menores dentro de cada bloque, de forma que se obtiene una recursividad.
- La expresión:
  - if height(tower.top) > width(tower.top) and standing(tower.rest) then standing(tower)
- Es recursiva porque aplica sobre tower.rest, es decir sobre los demás bloques constitutivos.
- A este tipo de reglas se les llama ***programas lógicos*** y el área del Machine Learning que los aborda se conoce como ***programación lógica inductiva***. Esta temática no es abordada en este curso.

7. Uso de árboles para predicción numérica

- Hasta ahora hemos usado los árboles de decisión para predecir categorías en lugar de cantidades.
- Sin embargo, se pueden obtener nodos hojas del árbol con valores numéricos.
- Debido a que estos árboles tratan de predecir valores numéricos, similar a lo que se hace en los métodos de regresión, a estos árboles se les conoce como ***árboles de regresión***.

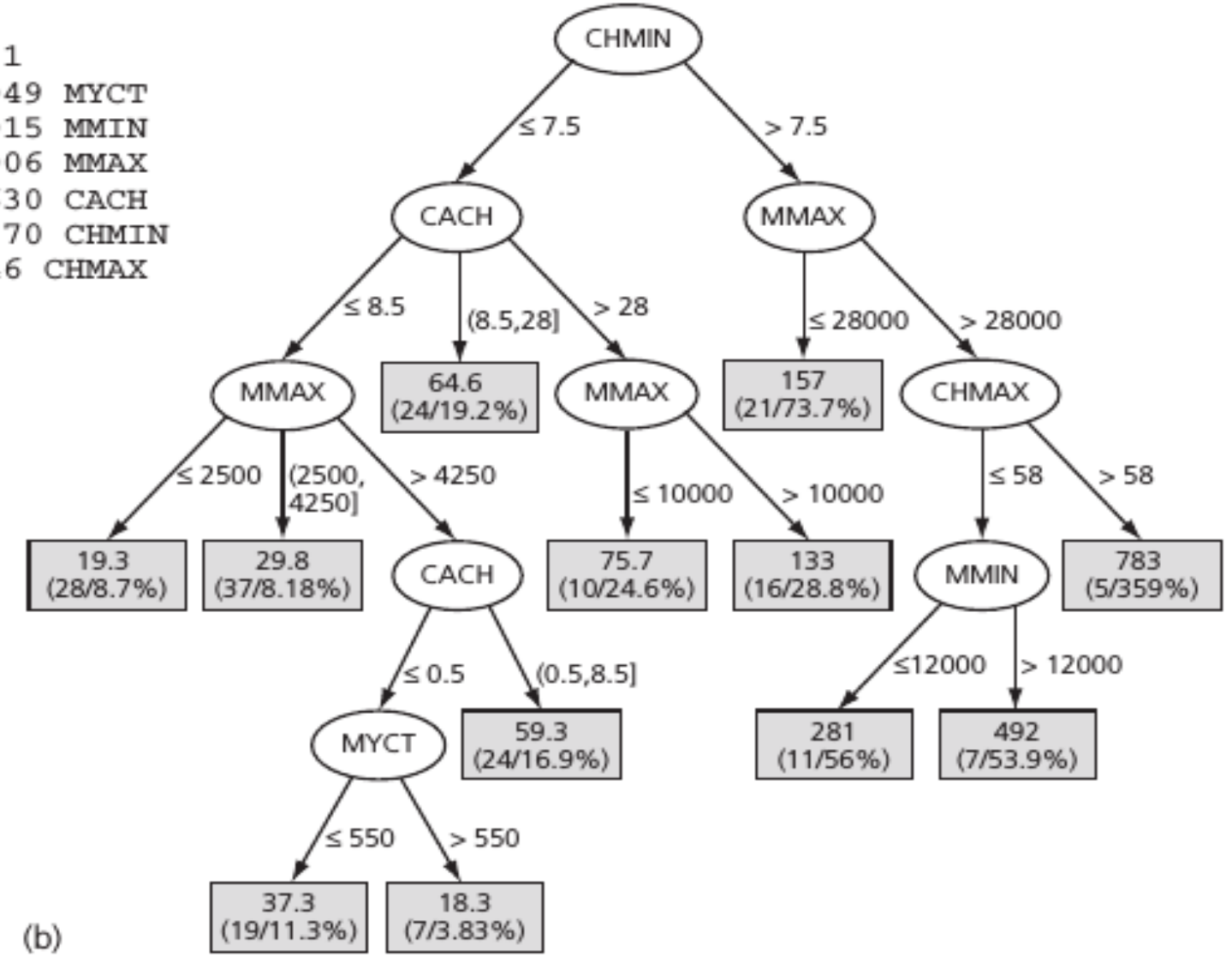
Árboles de regresión

- La figura siguiente muestra un árbol de regresión (b) y un a expresión de regresión lineal (a).
- La expresión de regresión lineal es más simple, mientras que el árbol es más complejo de entender.
- Los resultados del árbol son más aproximados que los de la expresión lineal.
- En la figura (c) se observa una combinación de un árbol de regresión con expresiones lineales en los nodos. A este híbrido se le llama ***árbol modelo***.

Árboles de regresión vs. Expresiones de regresión lineal

```
PRP =
  -56.1
  +0.049 MYCT
  +0.015 MMIN
  +0.006 MMAX
  +0.630 CACH
  -0.270 CHMIN
  +1.46 CHMAX
```

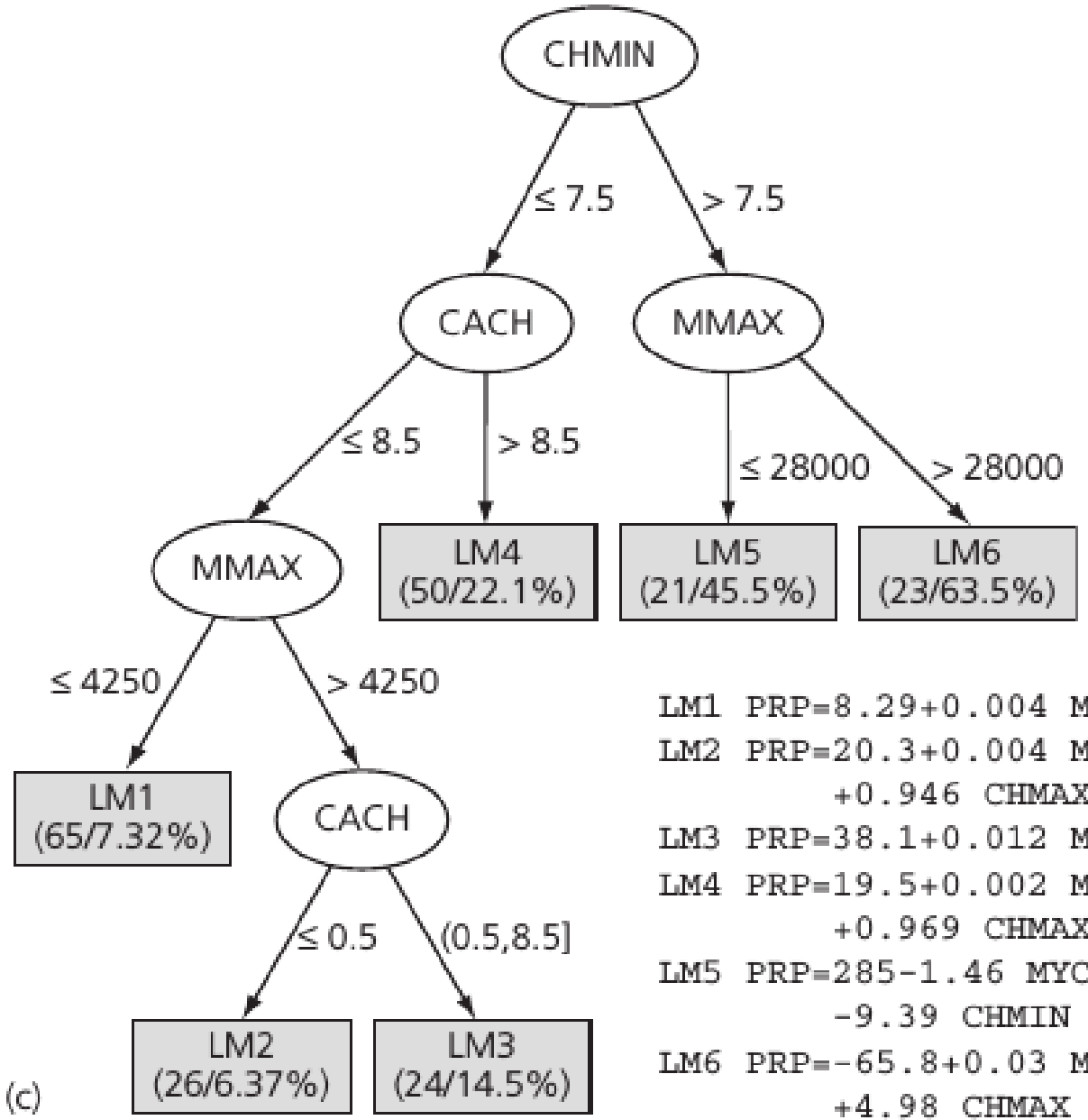
(a)



(b)

Árbol modelo

LM son expresiones lineales puestas como parches en el árbol. Se usan cuando se llega al nodo por haber cumplido las restricciones de los nodos anteriores.
Es más complejo que un árbol de regresión.
El error medio es menor que en el árbol de regresión




```
LM1 PRP=8.29+0.004 MMAX+2.77 CHMIN
LM2 PRP=20.3+0.004 MMIN-3.99 CHMIN
        +0.946 CHMAX
LM3 PRP=38.1+0.012 MMIN
LM4 PRP=19.5+0.002 MMAX+0.698 CACH
        +0.969 CHMAX
LM5 PRP=285-1.46 MYCT+1.02 CACH
        -9.39 CHMIN
LM6 PRP=-65.8+0.03 MMIN-2.94 CHMIN
        +4.98 CHMAX
```

8. Representación basada en instancias

- La forma más simple de aprender es la memorización plana (aprendizaje de memoria)
- Así, cuando llega una nueva instancia, se busca en la memoria la instancia más parecida a la nueva.
- El problema es interpretar “parecida”
- Nótese que esta es una forma diferente de representar el conocimiento extraído del conjunto de instancias. Se busca una instancia similar a la nueva entre las instancias de entrenamiento cuya clase es conocida.
- Esto se conoce como ***aprendizaje basado en instancias***.

Aprendizaje basado en instancias

- Los anteriores métodos en cierto sentido también son basados en instancias porque siempre se inicia el aprendizaje con un conjunto de instancias de entrenamiento.
- El aprendizaje basado en instancias usa las instancias mismas para representar lo que se aprende, en lugar de inferir un conjunto de reglas o un árbol de decisión.
- En lugar de crear reglas, trabaja directamente con las instancias mismas.
- El trabajo sucede al clasificar una nueva instancia, en lugar de cuando se procesa el conjunto de entrenamiento.

Aprendizaje basado en instancias

- La diferencia está en el momento en que el aprendizaje tiene lugar.
- El aprendizaje basado en instancias es perezoso, postergando el trabajo lo más posible.
- Los otros métodos son ansiosos, producen una generalización tan pronto como ven los datos.
-

K-nearest-neighbor

- En el aprendizaje basado en instancias cada nueva instancia es comparada con las otras existentes usando una métrica de distancia. La distancia más cercana marca cuál será la clase de la nueva instancia.
- A esto se le llama método de clasificación del vecino más cercano.
- Algunas veces hay más de un vecino cercano, la clase de la mayoría de los k vecinos más cercanos (o la media del peso de la distancia, si la clase es numérica) es la que se asigna a la nueva instancia. A este se le llama el método de los k vecinos más cercanos (k-nearest-neighbor).

- Calcular la distnacia con dos ejemplos es trivial con un solo atributo. Sólo es la diferencia entre los dos valores.
- Para dos o más atributos a comparar, se utiliza comunmente la distancia euclidiana estándar.
- Esto asume que los atributos están normalizados y tienen igual importancia.
- Uno de los principales problemas en el aprendizaje es determinar cuales atributos son más importantes.

Qué hacer con atributos nominales

- Cuando se tienen atributos nominales como colores por ejemplo, cómo calcular la distancia entre ellos?
- Una opción es asignar una distancia cero cuando son iguales y uno cuando son diferentes.
- Pero esto no es bueno cuando tenemos muchos valores intermedios como el naranja, que no es un color básico. Habría que usar una métrica para los matices, que muestren que se acercan más a un color base que a otro.

Atributos con diferente importancia

- Algunos atributos son más importantes que otros. Esto se puede reflejar mediante un peso asignado al atributo.
- Obtener los pesos de los atributos suele ser un problema clave en el aprendizaje basado en instancias.

Uso de recursos de cómputo

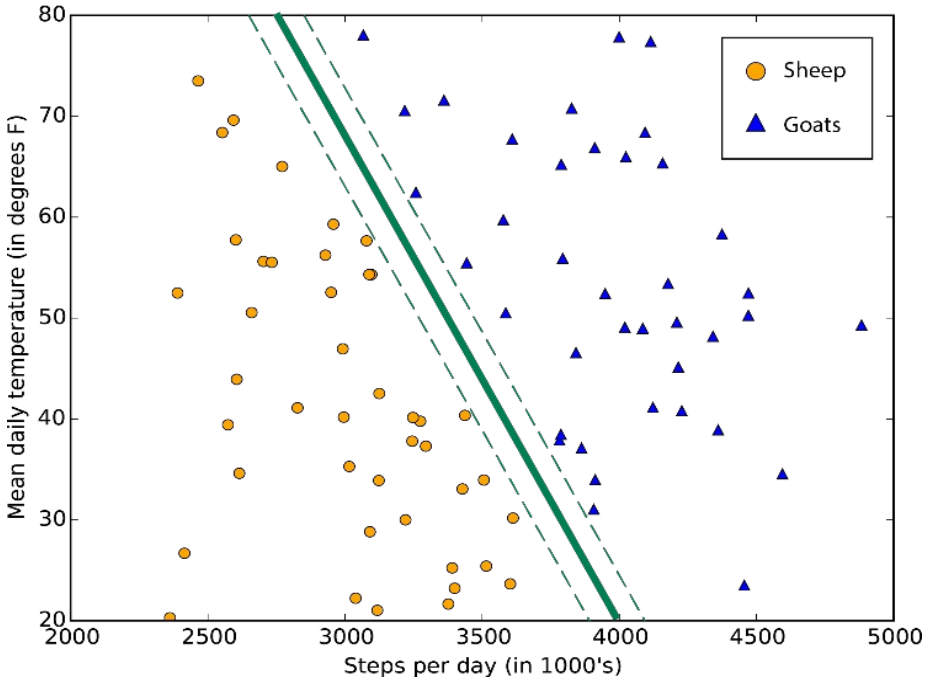
- Almacenar todas las instancias podría hacer que el cálculo de l vecino más cercano sea insoportablemente lento.
- Además se requeriría una gran cantidad de almacenamiento.
- Generalmente, algunas regiones del espacio de atributos son más estables que otras con respecto a la clase.
- Sólo se requieren pocos ejemplos para las regiones estables. Se espera que se necesiten menos ejemplos para las regiones estables dentro de los límites que para los límites de las clases.
- El decidir qué instancias almacenar y cuáles instancias eliminar es otro problema del aprendizaje basado en instancias.

Representación del conocimiento en aprendizaje basado en instancias

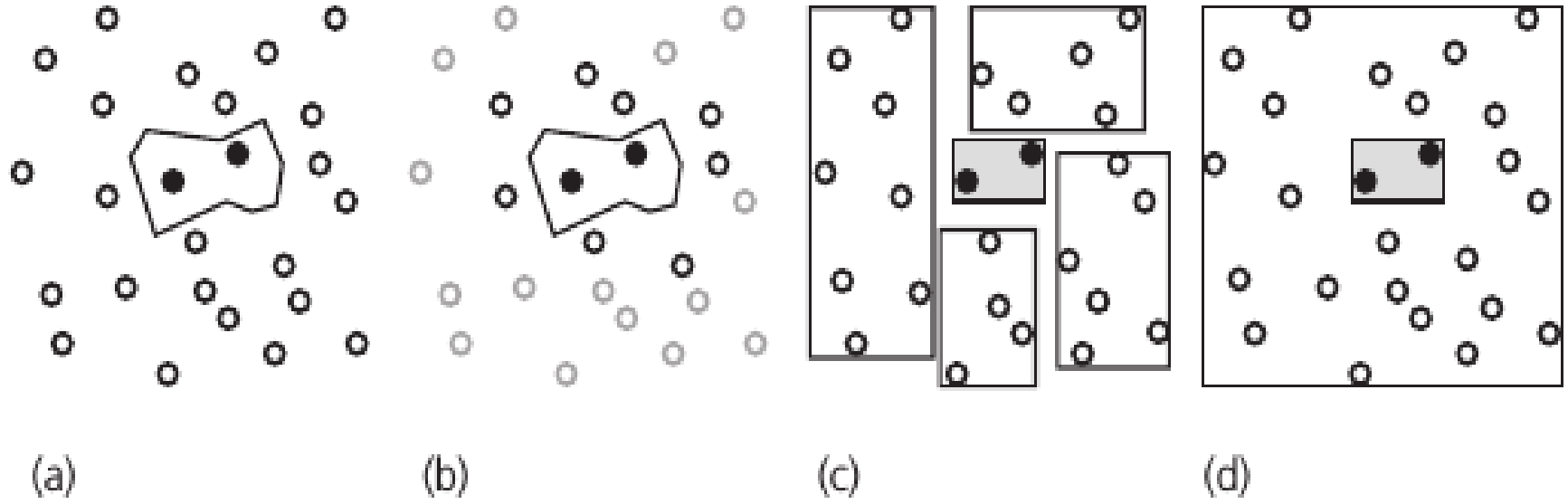
- Aparentemente en el aprendizaje basado en instancias no hay estructuras que describan lo aprendido, lo que va en contra de la definición de aprendizaje dada en la introducción del curso.
- Sin embargo, la combinación de distancias e instancias para forjar límites que distinguen una clase de otra, también son una forma de representación explícita del conocimiento.

Ejemplo

- Dada una instancia de cada una de dos clases, el límite que las separa es un bisector perpendicular a la línea recta que une las dos instancias.



Diferentes maneras de particionar un espacio



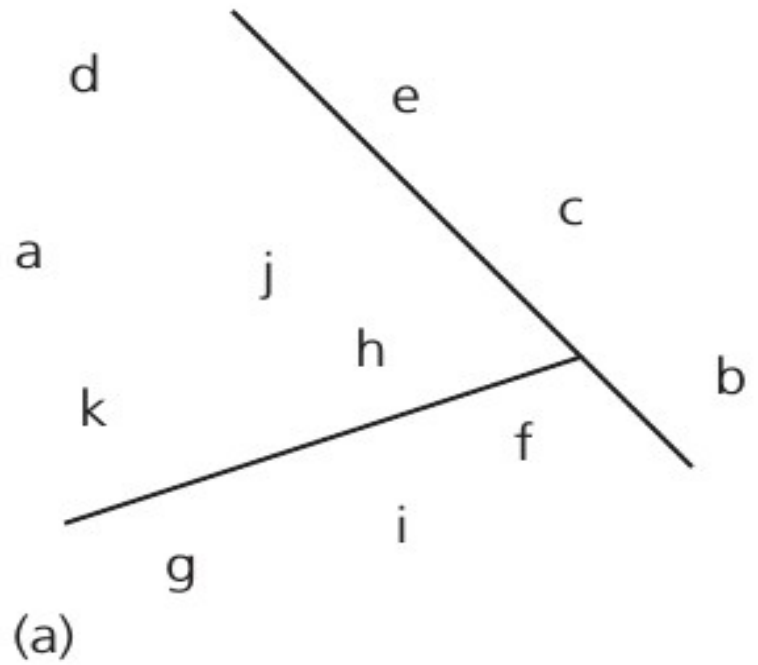
Particiones como estructuras de conocimiento

- Los ejemplos introducidos pueden ser clasificados con reglas más simples que dividen el espacio en rectángulos.
- Estos rectángulos se pueden representar con reglas if...then...else...que manejan límites superiores e inferiores de una variable (en la regla es una operación AND de dos condiciones “mayor que” y “menor que”)
- Es decir, que un rectángulo representa una regla de clasificación como las ya estudiadas, siendo entonces una estructura de conocimiento.

- Una clase más compleja de generalización es permitir regiones rectangulares anidadas una dentro de otra.
- Así, una región que es una clase, puede contener otra región interior de una clase diferente.
- Las regiones interiores podrían ser excepciones a las clases externas.

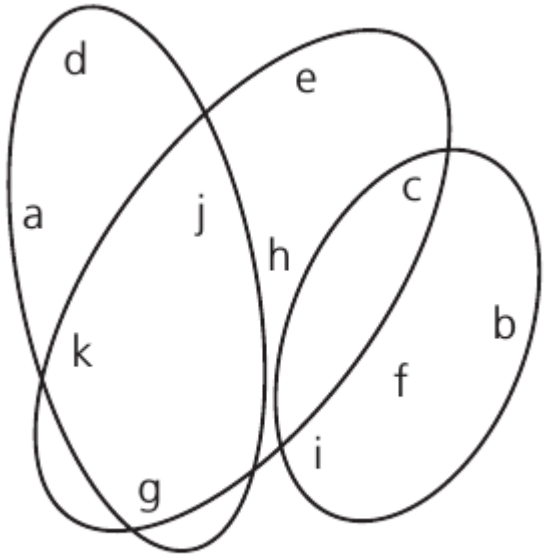
CLUSTERS

- Cuando se aprenden clusters en lugar de clasificadores, la salida toma la forma de un diagrama que muestra como caen las intancias dentro de los clusters.



CLUSTERS

- Algunos algoritmos de clusterización permiten a una instancia pertenecer a mas de un cluster.
- Se pueden usar diagramas de Venn para expresar estas intersecciones.



(b)

Clusters

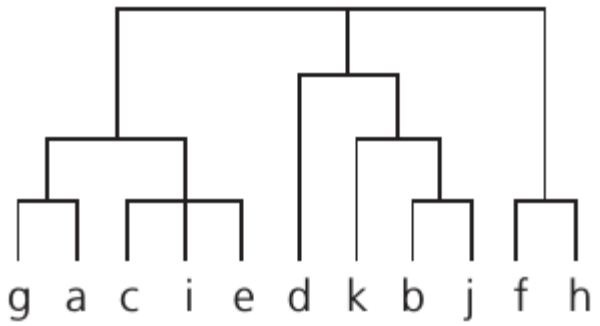
- Algunos algoritmos asocian de forma probabilística las instancias con los clusters, en lugar de hacerlo categóricamente.
- En este caso, para cada instancia hay una probabilidad o grado de membrecía con el cual esta pertenece a cada uno de los clusters. La suma de las probabilidades para cada instancia es 1 (suma de la fila).

	1	2	3
a	0.4	0.1	0.5
b	0.1	0.8	0.1
c	0.3	0.3	0.4
d	0.1	0.1	0.8
e	0.4	0.2	0.4
f	0.1	0.4	0.5
g	0.7	0.2	0.1
h	0.5	0.4	0.1

(c)

Clusters

- Otros algoritmos producen una estructura jerárquica de clusters.
- En el nivel más alto se divide el espacio de instancias en solo unos pocos clusters, cada uno de los cuales se divide en subclusters en el siguiente nivel de más abajo, y así sucesivamente.
- Los elementos que están juntos en niveles más bajos están más cerca que aquellos juntos en niveles más altos.



(d)

Clusters

- A estos diagramas se les llama dendogramas (otra forma de llamar a un árbol)
- La Clusterización es seguida a menudo por una etapa en la cual se infiiere un un árbol de decisión o un conjunto de reglas que asignan cada instancia al cluster al que pertenece. Por tanto, la operación de clusterización es sólo un paso en la vía de una descripción estructural.