# Re: Revisión de ROLLER

Hola,

Con respecto a la conferencia, como os comenté ayer, creo que enviar al ICMLC no nos merece la pena al tratarse de una conferencia CORE C.  Creo que es mucho más interesante intentarlo en la ECML PKDD que se celebra en Lisboa y que sí es un CORE A o CORE A\*.  Esto ya sabéis que incrementa enormemente las posibilidades de rechazo, pero creo que también el reto y la satisfacción si lo superamos.  Va a estar difícil, pero palante; y si nos chingan, buscamos un CORE B un poco más asequible, pero de entrada vamos a por los mejores.

Yo lo veo bien. El ECML es CORE A y los plazos son en quincenas. Yo creo que podríamos intentar tenerlo terminado para la entrega del 19 de Octubre o para la del 2 de Noviembre.

  Usar MDL como medida de calidad de las reglas.   Podemos meter una columna final en la que decimos si las reglas producidas cumplen con el principio MDL o no.  Evidentemente una técnica que es muy eficiente y efectiva no es realmente tan buena si resulta que no cumple con el principio MDL.  PATRICIA: por favor, apunta esta idea para nuestro artículo sobre validación.  Creo que esto tiene mucho sentido y es una idea original, junto con la que se me ocurrió para calcular la entropía de un clasificador y de un conjunto de training.

Hecho.

  Implementar MDL para cortar reglas que empiezan a ser muy complejas.  Esto ya está implementado en ROLLER 1, pero el problema es que nunca se activó este corte.  Además, si se hubiera activado, lo que ROLLER 1 habría hecho habría sido decir que no podía encontrar regla… pero realmente tendría que haber recuperado un savepoint y seguir por él.  El tema de los savepoints no llegué a implementarlo.

Está claro que MDL se puede usar 1) para medir la complejidad de las reglas, 2) para usarlo como criterio de poda. Si lo usamos como criterio de poda necesitamos implementar backtracking en la técnica.

  Usar un beam search para llevar k caminos de exploración en paralelo.

Ya estuvimos comentando esta idea y me transmitiste que no era tan fácil implementarlo. Yo creo que esta la deberíamos dejar para más adelante, si acaso. Pero no creo que vaya a mejorar enormemente los resultados en comparación con el esfuerzo.

  Usar k lookahead.  Experimentalmente hemos llegado a la conclusión de que con explorar 3-4 nodos es más que suficiente para obtener una buena regla.  La cuestión es que podríamos realizar una exploración con k lookaheads; es decir, en lugar de explorar una expansión con una única característica relacional, explorar una expansión con k características relacionales a la vez.  Es decir, en lugar de explorar el conjunto de características relacionales R, exploramos R^k en cada paso del algoritmo.  No estudiamos por ejemplo el binding (node2, parent, node) y después (node3, next, node2), sino que exploramos todas las combinaciones de 2, 3, 4, … k predicados a la vez.

Ya te comenté que a mi esta idea me parece muy interesante y creo que puede vender porque el lookahead es una idea de la que se habla mucho en Machine Learning.

  PATRICIA, TOÑI: En el pasado comentamos sobre usar funciones alternativas de scoring.  Incluso implementé Calcata para compararla con Information Content e Information Gain, pero los resultados eran sólo ligeramente mejores en tiempo.  El problema es que con ROLLER se exploran realmente muy pocos nodos.  En la experimentación que tenemos, se han explorado entre 2-3-4 nodos y 45-50 nodos… muy poquitos para que realmente haya una diferencia significativa de tiempo cuando se cambia la heurística de búsqueda (=funcion de scoring y de ganancia).  Por eso creo que de momento no merece la pena explorar esto mucho más.

De acuerdo. Yo el interés que le veo a las funciones de scoring es a que funcionen mejor de cara a la validación. Es decir, que a lo mejor las reglas que produce una función de scoring son más generales y a la hora de validar se comportan mejor. El "problema" aquí es, que nuestros resultados ya son bastante buenos.

  PATRICIA: Comentamos buscar reglas de asociación con el objetivo de eliminar características redundantes.  Las reglas de asociación son de la forma p(a1, a2, …, an) -> q(am) y significan que si los atributos a1, a2, …, an cumplen el predicado p, entonces el atributo am cumple el predicado q.  Esto significa que am es un atributo redundante puesto que se puede calcular sobre la base de los anteriores. Pero esto no lo podemos realmente aplicar por el mismo problema que te comenté en relación a usar cualquier otra técnica de eliminación de atributos.  Suponte que en la primera expansión resulta que a1 = 2 & a2 > 3 => a3 < 7; así que quitamos el atributo a3.  ¿Pero por qué motivo?  La regla sólo dice que cuando a1 y a2 cumplen una determinada propiedad, a3 cumple otra diferente.  Podríamos pensar en eliminar tan sólo cuando las reglas de asociación sean en igualdad, por ejemplo a1 = 3 & a2 = 3 => a3 = 7, pero esto tiene dos problemas: a) este tipo de reglas no son muy habituales; b) ¿qué pasa si en la siguiente expansión a3 y parent\_a9 forman una buena pareja para clasificar o incluso dan lugar a otra regla de asociación?  Pues que ya no tenemos a3.  En cualquier caso, hay que experimentar antes de validar o refutar la hipótesis de que esta técnica puede funcionar, pero a priori le veo estos problemas, por lo que la dejo para el final.

Bueno, yo no me refería exactamente a eso. Porque según lo que comentas, si a1 = 2 & a2 > 3 --> a3 < 7, aquí a3 no sería redundante. Lo que yo digo es que para todos los pares <a1,a2> siempre se da un valor a3, de modo que a3 siempre puede expresarse en términos de a1, a2. Es decir, se debería de obtener un conjunto de reglas de asociación para todos los valores <a1,a2> no sólo para esos que mencionas. Si se encuentra una regla de asociación para cada par, entonces a3 es redundante. Y la que eliminaría sería las características a1 y a 2, no a3. Por ejemplo:

a1                    a2                      a3

0                       0                       0

1                       3                       0

2                       3                       0

3                       3                       0

4                       4                       1

5                       4                       1

Si encontramos reglas de asociación para todos los valores, podríamos considerar que a3 es redundante:

a1 < 4 & a2 < 4 --> a3 = 0

a1 >= 4 & a2 = 4 --> a3 = 1

Si encontramos reglas de asociación para todo el rango de valores, entonces la característica es redundante, en caso contrario pues no lo es. El problema de esto es que posiblemente no encontremos características redundantes.

  RAFAEL: Creo que se te ha olvidado mencionar lo de los predicados inversos. Es decir, next(X,Y) = previous(Y,X). En esos casos sólo deberíamos de usar una característica, o next o previous. Lo mismo ocurriría con parent y children. También hablamos de las combinaciones que no tienen sentido, por ejemplo: (node, null, null) & (node2, next, node) & (node3, previous, node2) porque claramente node3 = node. Esto, si hacemos la mejora de calcular los predicados que son inversos no se daría. Lo que no sé es si se produciría alguna otra combinación sin sentido. Pero lo he estado pensando y creo que no. Supongo que algo del tipo: (node, null, null) & (node2, next, node) & (node3, next, node) dónde node2 = node3,  ¿no puede producirse verdad? Creo que esta alternativa es fácil de implementar.

  RAFAEL: ¿Y qué decís si sólo usamos las características que tenían frecuencia mayor a 1 en FOIL del estudio experimental anterior? Es decir, quitar aquellas que no se usaron nunca o aquellas que no se usaron nunca en ROLLER para la versión del SAC.

Un Saludo.

--

Patricia Jiménez Aguirre

ETSI Informática

Avda. Reina Mercedes, s/n

Sevilla, 41012 (Spain)