Revisión sobre métricas de complejidad en el modelado de clústers de un sistema CBR

Núria Macià, Ester Bernadó, Albert Fornells, Elisabet Golobardes, Josep M. Martorell y Josep M. Garrell

Grup de Recerca en Sistemes Intel·ligents

Enginyeria i Arquitectura La Salle, Universitat Ramon Llull

Quatre Camins 2, 08022 Barcelona (España)

{nmacia,esterb,afornells,elisabet,jmmarto,josepmg}@salle.url.edu

WWW home page: http://www.salle.url.edu/GRSI

Resumen

El análisis de la complejidad de los datos mediante métricas de complejidad es útil para predecir el comportamiento de un sistema de aprendizaje. Este trabajo ilustra cómo las métricas permiten definir un espacio a través del cual puede modelarse la bondad de los clústers creados a partir de un mapa autoorganizativo. Unos clústers que se utilizan para organizar la memoria de casos de un sistema de razonamiento basado en casos con el objetivo de mejorar el tiempo empleado en explorar dicha memoria.

Palabras clave: Complejidad de los datos, Métricas de complejidad, Razonamiento basado en casos, Clusterización.

1. Introducción

Con frecuencia, los sistemas de minería de datos deben manejar grandes volúmenes de datos, lo que implica un coste computacional elevado. Esto es especialmente crítico en sistemas de clasificación como el CBR (*Case-Based Reasoning* [1]), puesto que para predecir la clase de un ejemplo debe comparar el mismo con todos los ejemplos disponibles y recuperar los más semejantes. Una posible solución que reduce el elevado coste computacional consiste en organizar la memoria de casos (MC) en clústers [10]. Se trata de agrupar en patrones los casos que muestran propiedades similares, permitiendo al CBR realizar una recuperación más selectiva. A partir de una MC estructurada, el CBR selecciona el clúster más parecido a la entrada y de éste recupera algunos casos. Así pues, deben fijarse dos criterios: cuántos clústers han de seleccionarse y cuántos casos recuperarse del clúster. Sin embargo, el rendimiento del sistema puede quedar afectado si las agrupaciones no se han definido correctamente. Por lo tanto, es posible que se produzca una mejora en el tiempo de computación a la vez que la precisión de la clasificación disminuye.

Este artículo revisa la aplicación de las métricas de complejidad en el modelado de clústers de un sistema CBR. Sabiendo que la complejidad de los datos influye en la construcción de las agrupaciones, a partir de la información proporcionada por las métricas de complejidad sobre la distribución de los datos se intenta entender el comportamiento de las estrategias de recuperación. Esto ayuda a identificar para qué tipo de problemas es útil organizar la MC en agrupaciones [5] y a determinar qué estrategia alcanza mayor rendimiento según los requisitos establecidos por el usuario [6].

El artículo se estructura de la siguiente manera. La sección 2 describe brevemente el CBR y la organización de la memoria de casos me-

diante mapas autoorganizativos. En la sección 3 se introduce las fuentes de complejidad de un problema y cómo estimar su complejidad geométrica. La sección 4 recoge la experimentación realizada así como los resultados obtenidos. La sección 5 presenta una discusión sobre los resultados. Finalmente, la sección 6 expone las conclusiones y las líneas futuras.

2. CBR y organización de la memoria de casos

2.1. Razonamiento basado en casos

El CBR [1] es una técnica de aprendizaje automático, concretamente de aprendizaje analógico, que trata de resolver un nuevo caso recuperando, de una base de conocimiento, otros casos similares previamente resueltos. De los casos recuperados, se adecúa su solución para ajustarla a la resolución del nuevo caso. Este proceso constituye un ciclo en el que se diferencian cuatro fases: (1) la **fase de recuperación**, en la que se extraen de la base de casos los más parecidos al caso de entrada según una función de similitud, (2) la **fase de adaptación**, donde se ajusta la solución del caso recuperado, (3) la **fase de revisión**, que consiste en evaluar la bondad de la solución obtenida y (4) la **fase de almacenaje**, en la que si el nuevo caso se considera información relevante, se introduce en la memoria de casos.

Así pues, se observa que las cuatro fases mantienen constante relación con la memoria de casos lo que indica que su organización es un factor importante que condiciona el tiempo de las operaciones. El manejo de un gran volumen de datos obliga a configurar estrategias de recuperación para mejorar el rendimiento de los sistemas en términos de tiempo de cómputo.

2.2. Mapa autoorganizativo

El SOM es una técnica de clusterización basada en redes neuronales. La finalidad es generar un mapa topológico que agrupe los casos similares en patrones. Este comportamiento se aprovecha en el CBR para organizar la memoria de casos obteniendo un sistema SOMCBR [7]. La figura ilustra el ejemplo de una memoria de casos estructurada en un mapa de dos dimensiones con $M \times M$ patrones. El SOM está formado por dos capas: (1) una capa de entrada compuesta por N neuronas, donde cada neurona representa un atributo del caso de entrada y (2) una capa de salida compuesta por $M \times M$ neuronas, donde cada neurona contiene un conjunto de casos similares representado por un vector director. Cada neurona de la capa de entrada está conectada con todas las neuronas de la capa de salida. Para cada nuevo caso C que se introduce, la capa de salida computa el grado de similitud entre el caso C y su vector director aplicando una función de similitud. Nuestra función es el complementario de la distancia euclidiana.

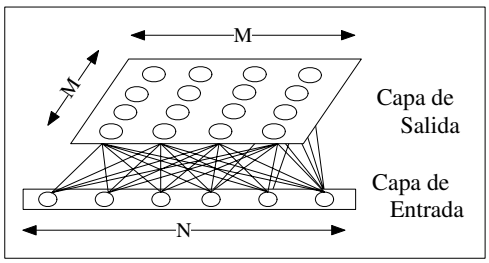

Figura 1: La memoria de casos está organizada por el SOM que distribuye los datos en $M \times M$ clústers, creando grupos de propiedades similares. Esta organización reduce el tiempo de cómputo de la fase de recuperación.

La recuperación consiste en: (1) buscar el patrón más parecido al caso de entrada y (2) compararlo con los casos del patrón seleccionado. En consecuencia, el SOMCBR reduce el tiempo de cómputo puesto que solamente usa los datos de un único clúster. No obstante, los patrones construidos pueden no estar bien definidos debido a la complejidad de los datos y entonces comprometer la precisión de la clasificación.

3. Complejidad de los datos

La complejidad del problema atañe a la distribución de los datos e influye en la precisión

de los sistemas clasificadores. Habitualmente, se estima el error (o precisión) del clasificador como medida para evaluar la calidad del mismo. Sin embargo, esta medida no sólo depende del clasificador sino que también depende de la complejidad inherente al problema que a menudo es la causa del error.

3.1. Causas de la complejidad de los datos

La dificultad de la clasificación está sujeta a tres causas [9][3]:

Ambigüedad de las clases. Algunos problemas de clasificación contienen clases que no pueden distinguirse dado que hay instancias que, perteneciendo a clases opuestas, toman los mismos valores en todos los atributos. Este fenómeno es propio de la ambigüedad intrínseca del problema o de la falta de atributos discriminantes. En el primer caso es necesario conocer el contexto para poder desambiguar y en el segundo se debería redefinir o ampliar el conjunto de atributos.

Complejidad de la frontera. Otros problemas pueden presentar una frontera de separación entre clases compleja que requiere un conjunto de descripción amplio o un algoritmo complejo para representarla. La complejidad de la frontera puede caracterizarse con la complejidad de Kolmogorov o la longitud mínima del programa que se necesita para reproducirla. Esto es independiente de la ambigüedad de las clases y del tamaño del conjunto de entrenamiento, ya que disponiendo de suficientes puntos sin ambigüedad en la clase, la descripción de la frontera puede ser todavía compleja.

Conjunto de ejemplos reducido y dimensión del espacio de atributos. La cantidad de instancias del conjunto de entrenamiento y su representatividad condicionan la capacidad de generalización del clasificador. Con un conjunto de entrenamiento pequeño se puede errar en la estimación de la complejidad del problema y catalogarlo de simple; es probable que los datos no sean representativos. Un número de ejemplos de entrenamiento insuficiente con una dimensionalidad elevada, es decir un número grande de atributos, pueden provocar que, aunque se obtenga un buen resultado de clasificación en entrenamiento, los aciertos en la fase de test con nuevas instancias sean bajos e inestables.

La combinación de estos aspectos fija un umbral mínimo de error. En [9] se propusieron un conjunto de métricas de complejidad que determinan la complejidad de la frontera. La ambigüedad de las clases y la representatividad de los ejemplos son difíciles de estimar. Por este motivo, los estudios sobre la complejidad se centran principalmente en caracterizar la geometría de la frontera entre clases.

3.2. Estimación de la complejidad geométrica

Las métricas de complejidad son una herramienta que permite evaluar la distribución geométrica de los datos con la finalidad de estimar su complejidad. Basu y Ho [2] presentan un conjunto de métricas que aproximan la complejidad de los datos desde diferentes aspectos: analizando los atributos de manera individual, midiendo la separabilidad de las clases o a partir de la topología de los datos. No obstante, cada métrica ofrece información de una característica concreta del problema que no siempre corresponde a la complejidad real, lo que significa que para una definición precisa se requiere la combinación de estas métricas. Incluso es posible que falten otro tipo de métricas para completar la estimación de la complejidad.

Actualmente, su aplicación se centra en: (1) predecir el error cometido por un clasificador para un conjunto de datos concreto [3] y (2) caracterizar la dificultad de un problema y relacionarla con el rendimiento del clasificador. Esto permitiría construir un espacio de complejidad sobre el que se puede representar el dominio de competencia de los distintos esquemas de aprendizaje.

4. Estimación de complejidad en SOMCBR y su aplicación

4.1. Experimentación

Este trabajo revisa la aplicabilidad del análisis de la complejidad de los datos en la clusterización de la memoria de casos. En una primera fase, se quiere estudiar cómo aplicar el análisis de la complejidad para predecir si es posible realizar una clusterización de la memoria que reduzca el coste computacional sin una pérdida significativa de precisión. Para ello, se llevó a cabo un estudio sobre 28 problemas de clasificación [5], de distinto dominio y con diferentes características, extraídos del Repositorio UCI [4]. Debido a que las métricas de complejidad están definidas para problemas de dos clases, el estudio se limitó a este tipo de problemas. De este modo, los problemas formados por m clases se transformaron en m problemas biclase, partiendo cada clase con respecto al resto. Como trabajo futuro, se quiere ampliar el estudio a problemas con más de dos clases.

Para cada conjunto de datos se ha ejecutado por un lado el CBR y por el otro el SOMCBR con diferentes configuraciones, en las que varían dos parámetros: (1) el número de clústers definidos y (2) el número de casos recuperados del clúster seleccionado, siendo un determinado porcentaje de casos o bien la totalidad de ellos. La precisión de cada clasificador se ha estimado utilizando la técnica *stratified 10-fold cross-validation* y las diferencias entre CBR y SOMCBR se han evaluado mediante el test *t Student*. El SOMCBR se ha ejecutado diez veces para cada configuración y el resultado promedio ha sido usado en la comparación con el CBR

Además de la precisión del CBR y SOMCBR, se ha evaluado el porcentaje de reducción en el número de comparaciones en el SOMCBR. Los resultados indican que hay problemas en los que la reducción de operaciones es significativa.

Efectivamente, las técnicas de clusterización favorecen el tiempo de recuperación de casos, pero en cuanto los clústers no se construyen debidamente, la precisión de la clasificación decae.

El objetivo de aplicar las métricas de complejidad es doble. En primer lugar, se pretende predecir a partir de la complejidad de los datos en qué casos es útil la clusterización de la MC, y por consiguiente saber cuando aplicar el SOMCBR. En segundo lugar, se intenta relacionar cuál es la mejor estrategia de recuperación para satisfacer los requisitos establecidos por el usuario en base a la complejidad del problema.

Las métricas más relevantes son:

Eficiencia del atributo (F3). En un problema caracterizado por un gran número de dimensiones, la información relevante se distribuye a través de los diferentes atributos. Esta métrica determina la eficiencia de cada atributo de manera individual y estima en qué medida el atributo participa en la separabilidad de las clases. Se parte de una heurística de continuidad local que asume que, para cada atributo, las instancias de la misma clase oscilan entre el mínimo y el máximo de esa clase. Entonces, cuando dos instancias de clase opuesta toman el mismo valor, se produce un solapamiento y por lo tanto se considera ambigua la región de esa dimensión. Esta ambigüedad se resuelve eliminando los puntos que se sitúan en la zona solapada y la eficiencia se calcula como el cociente entre los puntos restantes y el total de puntos. La métrica corresponde al valor máximo de las eficiencias calculadas para cada dimensión.

Distancia de la frontera (N1). Proporciona el porcentaje de nodos que conectan clases opuestas en un árbol de expansión mínimo (*Minimum Spanning Tree*, MST). El árbol se construye a partir del grafo que generan las relaciones de cada instancia con el resto mediante el cálculo de la distancia euclidiana. Esta métrica es un indicador de la separabilidad de las clases y de la tendencia a los clústers. Cuanto mayor es el valor de la métrica, más se acentúa la presencia de puntos cercanos de clases opuestas. En cambio, cuanto menor es, más agrupadas están las instancias de la misma clase y menos dificultad aparente denota el problema. Esta métrica no es adecuada para identificar la separabilidad lineal del problema, puesto que en un conjunto en

el que las clases están considerablemente entrelazadas, la mayoría de puntos se sitúan en la frontera y lo mismo sucede en un problema linealmente separable con los márgenes más estrechos que la distancia entre los puntos de la misma clase.

Distancia del vecino más cercano dentro y fuera de la clase (N2). Compara la dispersión dentro de la clase respecto a la separación entre clases opuestas. El cálculo se efectúa según la ecuación 1:

$$\frac{(\sum_{r=1}^{n} d_{intra}(x_i, x_j))/n}{(\sum_{r=1}^{n} d_{inter}(x_i, x_j))/n} \quad (1)$$

donde, d_{intra}: es la mínima distancia euclidiana entre dos instancias vecinas x_i y x_j de la misma clase; d_{inter}: es la mínima distancia euclidiana entre dos instancias x_i y x_j de clases opuestas; n: es el número de instancias del problema.

Se calculan las distancias euclidianas de cada punto con el vecino más próximo de su misma clase y con el de fuera de la clase. Se promedian todas las distancias del vecino más próximo entre puntos de la misma clase y también las distancias del vecino más próximo entre puntos de clases opuestas. La relación de las medias constituye la métrica de complejidad. La proximidad de puntos opuestos afecta a la tasa de error de un sistema clasificador K-NN. Es decir, cuanta más dispersión presenta el conjunto de datos, mayor es el error cometido por el clasificador. El valor de la medida puede ser superior a 1 si la distancia entre el vecino más cercano de la misma clase es más grande que la del vecino más cercano de la clase opuesta. Cuanto menor es la medida, más agrupadas están las instancias de la misma clase y más separadas de las de clase contraria.

4.2. Resultados

De la experimentación realizada se obtiene que el rendimiento del SOMCBR se diferencia en dos tipos. **Tipo 1.** El tiempo de cómputo mejora y el rendimiento se mantiene. **Tipo 2.** La precisión de la clasificación depende del número de casos recuperados. Esta situación corresponde al caso en el que el SOMCBR no ha podido construir correctamente los clústers debido a la complejidad y/o calidad de los datos. La figura 2 representa el $p - value$ del test estadístico que compara el rendimiento del SOMCBR respecto al CBR y la reducción del número de operaciones ($\%R$). En este gráfico se aprecia la distinción entre los dos tipos de comportamiento.

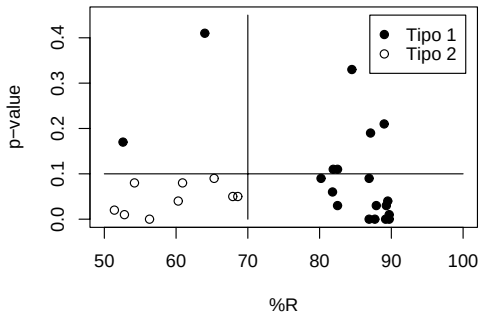

Figura 2: Representación gráfica del rendimiento del SOMCBR sobre un conjunto de 28 problemas de clasificación.

Aplicando las métricas de complejidad detalladas anteriormente sobre el mismo conjunto de problemas, se construye el espacio de la figura 3. La representación gráfica de las métricas, $F3$ en función del producto $N1{\cdot}N2$, constituye un espacio de complejidad que aísla los problemas con un comportamiento de Tipo 2, como en la figura 2. Esta distinción resulta útil para identificar *a priori* los problemas para los cuales el SOMCBR mejorará el tiempo de cómputo manteniendo la precisión de la clasificación. Es decir, permite predecir en qué casos será útil aplicar el sistema SOMCBR. Se observa que el cuadrante inferior derecho engloba los problemas de menor complejidad y es para estos que no parece adecuada nuestra herramienta. Se utiliza el producto de $N1{\cdot}N2$ ya que, debido a la correlación que presentan estas métricas [3], se acentúa el comportamiento en los valores extremos, sobretodo para valores

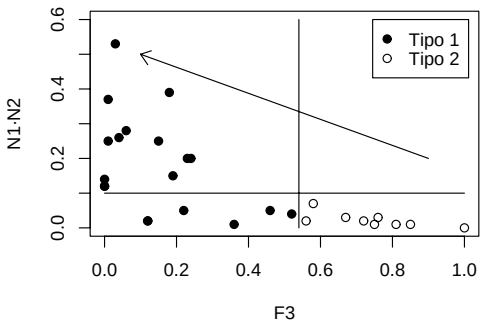

Figura 3: Espacio de complejidad. Cada conjunto de datos se representa en función del valor de $F3$ y $N1{\cdot}N2$. Este mapa permite predecir la aplicabilidad del SOMCBR.

bajos.

Los problemas de Tipo 2 corresponden a problemas en que el porcentaje de reducción de operaciones es bajo y coincide, para todos los casos, que muestran una complejidad baja ($F3$ con un valor elevado que indica que hay atributos discriminantes y un valor bajo de $N1{\cdot}N2$ que simboliza una elevada separabilidad entre clases). En cambio, los problemas de Tipo 1 presentan altos porcentajes de reducción en los que no se refleja una tendencia estable. A veces el SOMCBR es equivalente al CBR, es decir que no hay diferencias significativas en cuanto a porcentaje de aciertos, y en otras es peor. Estas dos situaciones que se producen en el Tipo 1 no se pueden diferenciar mediante las métricas de complejidad. Por lo tanto, sólo cabe concluir que si el problema es fácil, desde el punto de vista de complejidad, no es útil aplicar clusterización en la MC ya que no habrá una reducción de operaciones significativa. Para los problemas de complejidad elevada la aplicabilidad del SOMCBR dependerá del problema en concreto.

Entendiendo que la complejidad de los datos afecta a la construcción de los clústers y por lo tanto al comportamiento de las diferentes estrategias, se considera el análisis de la complejidad de la frontera [9] para evaluar su influencia en las estrategias de recuperación.

Refinando el estudio anterior sobre una experimentación ampliada a 56 conjuntos de datos, se obtiene un nuevo espacio que clasifica los problemas en tres tipos de complejidad (ver Fig. 4). En este espacio el punto (1,0) se corresponde al punto de mínima complejidad (mCP) mientras que el punto (0,1) alcanza la máxima complejidad posible (MCP).

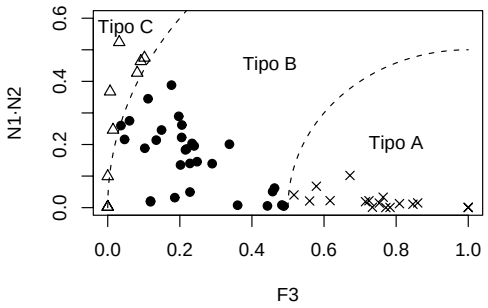

Figura 4: Espacio de complejidad. Divide en tres regiones el grado de complejidad de los datos, siendo C el mayor posible.

La distancia desde el punto mCP divide los conjuntos de datos estudiados en tres grupos: (1) problemas de complejidad baja (Tipo A, la distancia respecto al punto mCP es menor a 0.5), (2) problemas de complejidad elevada (Tipo C, con un valor mayor o igual a 0.9) y (3) problemas de complejidad intermedia (Tipo B). Con esta clasificación se puede describir el rendimiento de cada estrategia con mayor precisión.

Este mapa de complejidad supone una nueva variable que se incorpora en la elección de la configuración de la estrategia de recuperación. El rendimiento es un compromiso entre el tiempo de cómputo y la precisión de clasificación deseada. El resultado del análisis se resume en la tabla 1. Para problemas de complejidad A, es decir fáciles, hay una alta correlación entre el número de operaciones del

SOMCBR respecto al CBR y la precisión de la estrategia de recuperación concreta, que alcanza un coeficiente de correlación de 0.96. Significa que cuantas más operaciones, mayor es el porcentaje de aciertos, lo que se traduce en que la clusterización perjudica la precisión del sistema. En los problemas de complejidad B, complejidad intermedia, se constatan dos efectos: (1) el número de operaciones disminuye en la mayoría de estrategias y (2) el coeficiente de correlación decrece hasta un 0.86. Así pues, se obtiene una mejora en el rendimiento sin perder precisión. Finalmente, para los problemas de tipo C, de mayor complejidad, se pronuncian los efectos enunciados en los problemas de tipo B siendo ya la correlación de 0.76.

Tipo de complejidad	Coeficiente de correlación
A	0.96
B	0.86
C	0.76

Cuadro 1: Coeficientes de correlación entre la precisión y la reducción del número de operaciones según el grado de complejidad.

Analizando cuál es la estrategia más adecuada, se puede decir que para los problemas de tipo A y B, que corresponden a una complejidad baja e intermedia respectivamente, la estrategia All_All es la que ofrece mejor porcentaje de aciertos. Esta configuración equivale a un sistema CBR. En problemas de complejidad elevada, tal como se observa en la figura 5, la mejor estrategia se encuentra en la configuración Eq3_05_All. Esta se basa en recuperar un porcentaje de los casos de cada clúster en función de su bondad para representar el patrón de entrada. La bondad se calcula mediante la diferencia entre el caso de entrada y el vector director del patrón.

5. Discusión

En este trabajo se ha revisado cómo el análisis de la complejidad puede ser útil para estimar la aplicabilidad de la clusterización para sistemas CBR. Por una parte, se ha analizado

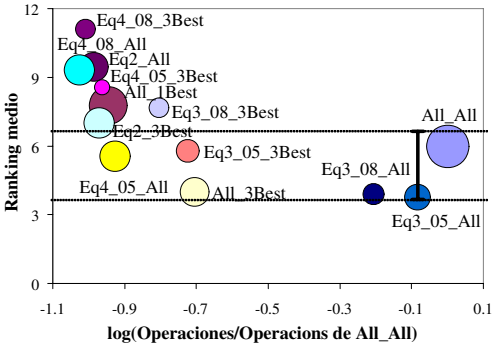

Figura 5: Análisis de las estrategias de recuperación según el tipo de complejidad C.

la posibilidad de predecir cuándo la clusterización puede aportar una mejora importante en el coste computacional sin perder demasiada precisión. Se ha detectado que para problemas de baja complejidad, la clusterización no aporta beneficios. Sin embargo, para problemas de elevada complejidad, la clusterización suele conllevar una reducción importante del número operaciones aunque no se puede predecir si esta reducción implica pérdida de precisión o no. Con las métricas actuales, no se puede predecir qué tipo de problemas mantendrán la precisión con respecto a un sistema sin clusterizar.

Esta limitación puede venir dada por una caracterización insuficiente de la complejidad del problema. En el trabajo presentado, se usaron las métricas más significativas que son F3, N1 y N2. La primera dedica su atención a estimar en qué grado los atributos individualmente intervienen en la separación de las clases y las dos siguientes describen la complejidad de la frontera entre clases teniendo en cuenta la distancia entre puntos. El estudio de nuevas métricas podría aportar más información y aproximarnos mejor a la caracterización del problema. Asimismo sería necesario aumentar el número de problemas a analizar, que en el estudio presentado cuenta con 28 en una primera fase y 56 en una segunda. Algunos estudios sobre la complejidad de los problemas usan alrededor de 300 *datasets* [8] para gene-

ralizar el comportamiento. Podría suceder que nuestra elección de problemas esté sesgada y limite la generalización de los resultados. De hecho, se ha usado un conjunto reducido de problemas debido al elevado coste computacional que supone ejecutar la clusterización en diferentes configuraciones y con 10 repeticiones cada una de ellas.

La segunda parte de nuestro estudio realiza un paso más allá de la mera predicción de la aplicabilidad de la clusterización. En concreto, trata de analizar qué tipo de estrategia de recuperación del ciclo del CBR es más apropiada según el problema. El estudio pretende obtener como resultado una recomendación del tipo de estrategia a usar dado un problema caracterizado por su complejidad. Para ello, se propuso una metodología que parte de una taxonomía de las estrategias de recuperación donde interviene la definición de dos parámetros: el número de clústers y el número de casos que deben recuperarse. A partir de esta clasificación, cada configuración se ejecuta sobre un conjunto de datos del cual se obtiene, mediante una representación gráfica, un mapa del rendimiento de cada estrategia en función de la reducción del número de operaciones y de la precisión de la clasificación. Para cada conjunto de datos se mide su complejidad mediante tres métricas, F3, N1 y N2, cuya combinación deriva en un mapa sobre el que se identifica a qué tipo de complejidad pertenece el problema. Para cada tipo, se construye un mapa de estrategias en función del tiempo computacional y la precisión de la clasificación.

El estudio presentado se ha realizado únicamente para problemas con dos clases. El motivo es que la definición original de las métricas de complejidad se realizó para tal tipo de problemas [9]. Si el problema es multiclase, se puede analizar la complejidad promedio de los problemas biclase que derivarían de comparar cada clase con el resto. Sin embargo, probablemente sería más preciso analizar la complejidad del problema multiclase en conjunto y no con cada clase por separado. Actualmente, ya existe alguna propuesta de métricas de complejidad para problemas multiclase [2], aunque no existen trabajos que lo usen. Creemos que sería útil extender este estudio para problemas multiclase.

En resumen, este trabajo ha revisado una posible aplicabilidad del estudio de la complejidad del problema en el análisis de la clusterización para sistemas CBR. En general, el estudio de la complejidad del problema puede ayudar a comprender el comportamiento de determinadas estrategias de aprendizaje, mejorar el rendimiento de las mismas y predecir la aplicabilidad de éstas *a priori*. Aún así, son necesarios más esfuerzos para estimar mejor la complejidad de los problemas y relacionarla debidamente con el rendimiento de los esquemas de aprendizaje.

6. Conclusiones

La necesidad de mejorar el tiempo de cómputo sin mermar la precisión de la clasificación ha llevado a plantear una metodología que ayude a decidir la estrategia de recuperación de casos apoyándose en la información proporcionada por las métricas de complejidad. Así pues, la complejidad de los datos es una nueva variable que ayuda a predecir en qué casos es útil aplicar el SOMCBR y a un nivel superior cuál debe ser la configuración de la estrategia, en lo que se refiere a número de clústers a seleccionar y número de casos a recuperar. Cabe destacar que el beneficio del estudio de la complejidad de los datos se encuentra en que las decisiones se toman únicamente con los datos y sin la necesidad de aplicar el sistema clasificador. Por lo tanto, la aportación de las métricas de complejidad es la posibilidad de predecir *a priori* el rendimiento del SOMCBR para un problema determinado.

Las líneas de futuro apuntan hacía el estudio de nuevas métricas que puedan completar la definición de la complejidad de los datos, y consolidar así la metodología sobre memorias de casos organizadas por otras técnicas.

Agradecimientos

Este trabajo ha sido soportado en parte por los proyectos TIN2006-15140-C03-03 y TIN2005-

08386-C05-04 gracias al Ministerio de Ciencia y Tecnología, y al Fondo Europeo de Desarrollo Regional (FEDER). También se debe al *Departament d'Universitats, Recerca i Societat de la Informació* (DURSI) por el apoyo proporcionado mediante las becas 2005SGR-302 y 2007FIC-00976. Asimismo, los autores agradecen a *Enginyeria i Arquitectura La Salle*, de la Universitat Ramon Llull, el soporte a nuestro Grupo de Investigación en Sistemas Inteligentes

Referencias

[1] A. Aamodt and E. Plaza. Case-based reasoning: Foundations issues, methodological variations, and system approaches. *AI Communications*, 7:39–59, 1994.

[2] M. Basu and T.K. Ho. *Data Complexity in Pattern Recognition.* Advanced Information and Knowledge Processing. Springer, 2006.

[3] E. Bernadó and T.K. Ho. Domain of competence of XCS classifier system in complexity measurement space. *IEEE Transaction Evolutionary Computation*, 9(1):82–104, 2005.

[4] C.L. Blake and C.J. Merz. UCI repository of machine learning databases, 1998.

[5] A. Fornells, E. Golobardes, J.M. Martorell, J.M. Garrell, E. Bernadó, and N. Macià. Measuring the applicability of self-organization maps in a case-based reasoning system. In *3rd Iberian Conference on Pattern Recognition and Image Analysis*, volume 4478 of *LNCS*, pages 532–539. Springer-Verlag, 2007.

[6] A. Fornells, E. Golobardes, J.M. Martorell, J.M. Garrell, E. Bernadó, and N. Macià. A methodology for analyzing the case retrieval from a clustered case memory. In *7th International Conference on Case-Based Reasoning*, LNAI. Springer-Verlag, 2007. In press.

[7] A. Fornells, E. Golobardes, D. Vernet, and G. Corral. Unsupervised case memory organization: Analysing computational time and soft computing capabilities. In *8th European Conference on Case-Based Reasoning*, volume 4106 of *LNAI*, pages 241–255. Springer-Verlag, 2006.

[8] T.K. Ho. A data complexity analysis of comparative advantages of decision forest constructors. *Pattern Analysis and Applications*, 5:102–112, 2002.

[9] T.K. Ho and M. Basu. Complexity measures of supervised classification problems. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 24(3):289–300, 2002.

[10] T. Kohonen. *Self-Organization and Associative Memory*, volume 8 of *Springer Series in Information Sciences*. Springer, Berlin, Heidelberg, 1984. 3rd ed. 1989.

Remainder Subset Awareness for Feature Subset Selection

Gabriel Prat-Masramon
Faculty of Computer Science
Polytechnical University of Catalonia
Barcelona, Spain
gprat@lsi.upc.edu

Lluís A. Belanche-Muñoz
Faculty of Computer Science
Polytechnical University of Catalonia
Barcelona, Spain
belanche@lsi.upc.edu

2007-05-29

Abstract

Feature subset selection has become more and more a common topic of research. This popularity is partly due to the growth in the number of features and application domains. The family of algorithms known as *plus-l-minus-r* and its immediate derivatives (like *forward selection*) are very popular and often the only viable alternative when used in *wrapper* mode. In consequence, it is of the greatest importance to take the most of every evaluation of the inducer, which is normally the more costly part. In this paper, a technique is proposed that takes into account the inducer evaluation both in the *current* subset and in the *remainder* subset (its complementary set) and is applicable to any sequential subset selection algorithm at a reasonable overhead in cost. Its feasibility is demonstrated on a series of benchmark data sets.

1 Introduction

In the last few years *feature selection* has become a more and more common topic of research, a fact probably due to the introduction of new application domains and the growth of the number of features involved. An example of these new domains is web page categorization, a domain currently of much interest for internet search engines where thousands of terms can be found in a document. Another example is found in appearance-based image classification methods which may use every pixel in the image. Classification problems with many features are also very common in medicine and biology; e.g. molecule classification, gene selection or medical diagnostics.

Feature selection can help in solving a classification problem with irrelevant and/or redundant features for many reasons. First it can make the task of data visualization and understanding easier by eliminating irrelevant features which can mislead the interpretation of the data. It can also reduce costs since the measurement or recording of some of the features can be avoided; this is especially important in domains where some features are very expensive to obtain, e.g., a costly or invasive medical test. In addition, a big benefit of feature selection is in defying the curse of dimensionality to help the induction of good classifiers from the data. When many *unuseful*, i.e. irrelevant or redundant, features are present in the training data, classifiers are prone to finding false regularities in the input features and learn from that instead of learning from the features that really determine the instance class (this is also valid when predicting the instance target value in the case of regression).

This work addresses the problem of *selecting a subset of features* from a given set by introducing a general-purpose modification for feature subset selection algorithms which iteratively select and discard features. An important family of algorithms for feature subset selection perform an explicit search in the space of subsets by iteratively adding and/or removing features one at a time until some stop condition is met. The idea is then to

use the evaluation of the inducer in the so-called *remainder set* (the set complementary to the current subset of selected features) as an additional source of information. This information is used in conjuntion to conventional algorithms, that only use the evaluation on the current subset of selected features. In our experimental results, such simple modification achieves significant improvements in the maximization of the objective function for classification tasks. The modified algorithms obtain similar numbers of selected features in general, or a further reduction if purely *forward* algorithms are used.

The rest of the paper is organized as follows: first we briefly review the feature subset selection problem (Section 2). Section 3 introduces the concept of the remainder set of features and suggests a modification of the previously presented algorithms. In the next section the experimental design is defined and the results are exposed and discussed. Finally, section 5 extracts conclusions about these results and introduces some future work.

2 Feature Subset Selection

There are two main approaches to feature selection: *filter* methods and *wrapper* methods. These two families of methods only differ in the way they evaluate the candidate sets of features. While the former uses a problem independent criterion, the latter uses the performance of the final classifier to evaluate the quality of a feature subset. The basic idea of the filter methods is to select the features according to some prior knowledge of the data. For example, selection of features based on the *conditional probability* that an instance is a member of a certain class given the value of its features [1]. Another criterion commonly used by filter methods is the *correlation* of a feature with the class, i.e. selecting features with high correlation [3]. In contrast, wrapper methods suggest a set of features that is then supplied to a classifier, which uses it to classify the training data and returns the classification accuracy or some other measure thereof [5].

It is common to see feature subset selection in a set Y of size n as an *optimization problem* where the search space is $\mathcal{P}(Y)$ [6]. In this setting, the *feature selection problem* is to find an optimal subset $X^* \in \mathcal{P}(Y)$ which maximizes a given criterion $J : \mathcal{P}(Y) \rightarrow [0, 1]$ as seen in eq. (1). Having $J(X)$ as the evaluation criterion is a common characteristic of many feature selection algorithms. The criterion J may be problem-independent or may be the classifier that will be used to solve a classification problem, and thus this setting is valid either for filter or wrapper algorithms. In any case, we will refer to $J(X)$ as the *usefulness* of feature subset X.

$$X^* = \arg\max_{X \in \mathcal{P}(Y)} J(X) \tag{1}$$

In the literature, several suboptimal algorithms have been proposed for doing this. Among them, a wide family is formed by those algorithms which, departing from an initial solution, iteratively add or delete features by locally optimizing the objective function. The search starts with an arbitrary set of features (e.g. the full set or the empty set) and moves iteratively to neighbor solutions by adding or removing features. These *sequential* algorithms with no backtracking can be cast in the general class of *hill-climbing algorithms*. They leave a sequence of visited states $X_{k_1}, \ldots X_{k_m}$, where the size of every X_{k_i} is k_i and m is the number of visited states. The difficulty of the feature selection problem can be illustrated by the following facts:

1. In general, it is not the case that $J(X_{k_{i+1}}) \geq J(X_{k_i})$ (not even for the simplest algorithms like SFG or SBG).
2. There may exist "ugly" feature subsets along the way X_{k_i} such that $J(X_{k_i}) < J(\emptyset)$, for some $i \geq 0$.
3. Take X_{k_i} and $X_{k_{i+1}}$ such that $X_{k_i} \subset X_{k_{i+1}}$ and $k_{i+1} = k_i + 1$. In general, it is not the case that $J(X_{k_{i+1}}) \geq J(X_{k_i})$.
4. Calling X^*_{k+1} a current optimal solution with $k+1$ features, it is not the case that $J(X^*_{k+1})$ contains $J(X^*_k)$. This is known as the *nesting* problem.

Expressed more succinctly, any partial order with respect to J in the set $\mathcal{P}(X)$ is conceivable. Among the proposed algorithms for attacking this problem are the sequential forward generation (SFG) and sequential backward generation (SBG), the *plus l - take away r* or $PTA(l,r)$ proposed by Stearns [10] or the floating search methods [9]. They both introduce methods for the generation of the sets of features by combining steps of SFG with steps of SBG but keep using a certain $J(X)$ as evaluation criterion.

$X_0 \leftarrow \emptyset$ //Initial subset
$i \leftarrow 0$
repeat
  //Subset generation
  $\vec{S}_{i+1} \leftarrow \{X \mid X = X_i \cup \{x\} \wedge x \in Y \setminus X_i\}$
  //Subset evaluation
  $X_{i+1} \leftarrow \arg\max_{X \in \vec{S}_{i+1}} J(X)$
  $i \leftarrow i + 1$
//Stopping criterion
until $J(X_i) \leq J(X_{i-1}) \vee i = n$
return X_{i-1} //Selected subset

Algorithm 1: SFG

$X_0 \leftarrow Y$ //Initial subset
$i \leftarrow 0$
repeat
  //Subset generation
  $\vec{S}_{i+1} \leftarrow \{X \mid X = X_i \setminus \{x\} \wedge x \in X_i\}$
  //Subset evaluation
  $X_{i+1} \leftarrow \arg\max_{X \in \vec{S}_{i+1}} J(X)$
  $i \leftarrow i + 1$
//Stopping criterion
until $J(X_i) \leq J(X_{i-1}) \vee i = 0$
return X_{i-1} //Selected subset

Algorithm 2: SBG

Algorithm 1 and **Algorithm 2** below describe two of the classic feature selection algorithms using this point of view: sequential forward generation (SFG) and sequential backward generation (SBG). In these algorithms X_0 is the starting set of features of the algorithm, $\vec{S}_k$ the set of sets of features generated during the subset generation phase and X_k the selected set of features at iteration k. It can be seen that the subset evaluation phase in the two algorithms is exactly the same while the initialization and the subset generation phases change. Note that at all times the size of X_k is k and thus $X_n = Y$ and $X_0 = \emptyset$.

3 The Remainder Set of Features

As the goal of feature selection is to find an optimal subset X^* as seen in (1), it seems plausible to choose an X_k for each iteration as in (2) in a stepwise and greedy way, which is exactly what the previously described feature selection algorithms do:

$$X_k = \arg\max_{X \in \vec{S}_k} J(X), \qquad k = 1, \ldots, n \tag{2}$$

In real problems, features are far from independent, thus not always the best feature set in every iteration has to be the best option. Quite possibly there is some combination of features that would be a better choice that the feature which maximizes $J(X)$ in this iteration. In this vain, the forward steps in the previous algorithms are not taking into account some information they could use. Only the *usefulness* of every generated subset of features is measured, as in (2). However, by considering the current set of features X_k another set is implicitly created, the set of *remaining* features or *remainder set* $Y_k = Y \setminus X_k$. This set can also give information about the new variable to be added or removed at every step. It is our conjecture that a way to enhance the detection of feature interactions is to see how the addition of a feature to X_k (a removal, from the point of view of Y_k) affects the *usefulness* of the remainder set. The idea is to add that feature most useful to X_k and whose removal is most harmful to Y_k. An analogous reasoning can be made in backward steps by interchanging the roles of the current and remainder subsets. The general idea is called *Remainder Subset Awareness* for obvious reasons.

With this formulation we have a multiobjective problem, since not always the sub-

set with maximum $J(X_k)$ will coincide with the subset with minimum $J(Y_k)$, so it will not be possible to satisfy both objectives with the same single solution. In this case, either the two solutions have to be explored or a trade-off has to be found that partly optimizes both objectives. If both solutions are chosen for further exploration, then the search space is highly increased over the original version of the algorithm, and the complexity of the algorithm grows from polynomial to exponential, which is unfeasible. A reasonable alternative is to choose the subset which maximizes some predefined function f of the two criteria among the two candidate subsets, as expressed by:

$$\arg\max_{X \in \vec{S}_k} f\left[J(X), J(Y \setminus X)\right], \qquad k = 1, \ldots, n \tag{3}$$

The function $f : (0,1)^2 \rightarrow (0,1)$ has to be chosen to be continuous in both arguments, increasing in the first and decreasing in the second and to permit control on the relative importance of the two arguments (thus it is non-symmetrical). Following this alternative, an algorithm of the sequential kind can be modified by replacing the evaluation function $J(X)$ with the one in Eq. 3. As an example, the following **Algorithm 3** shows the straightforward *Remainder Subset Aware* version of the original SFG presented in **Algorithm 1**. Other forward/backward algorithms would be modified analogously.

$X_0 \leftarrow \emptyset$ //Initial subset
$i \leftarrow 0$
repeat
  //Subset generation
  $\vec{S}_{i+1} \leftarrow \{X \mid X = X_i \cup \{x\} \wedge x \in Y \setminus X_i\}$
  //Subset evaluation
  $X_{i+1} \leftarrow \arg\max_{X \in \vec{S}_{i+1}} f\left[J(X), J(Y \setminus X)\right]$
  $i \leftarrow i + 1$
//Stopping criterion
until $J(X_i) \leq J(X_{i-1}) \vee i = n$
return X_{i-1} //Selected subset

Algorithm 3: Remainder set aware SFG

The chosen evaluation function f, which combines the *usefulness* of the selected subset of features with that of the remaining subset is shown in Eq. 4.

$$f(x, y) = x^k \times (1 - y)^{1-k}, \; k, x, y \in [0, 1] \tag{4}$$

Note that $k = 1$ recovers the conventional algorithms and $k = 0.5$ corresponds to the geometrical mean between x and $1 - y$. In general, lower values of k give more weight to the evaluation of the inducer in the remainder set.

4 Experimental work

The previous idea is first illustrated using the CorrAl problem, a small dataset with some specific characteristics that make it useful to test feature subset selection algorithms in a known environment [4]. This dataset has two classes and six boolean features (A_0; A_1; B_0; B_1; I; C). Feature I is irrelevant, feature C is correlated to the class label 75% of the time, and the other four features are relevant to the boolean target concept: $(A_0 \wedge A_1) \vee (B_0 \wedge B_1)$. SFG will choose C first as it is the best feature when taken all alone [4]. The hypothesis is that the *usefulness* of the remainder set would be so high if C was chosen that the modified version of SFG would not choose it. After running the experiments with CorrAl, the hypothesis was confirmed: a conventional SFG chose the features in the order $\{C, I, A_0, A_1, B_0, B_1\}$, whereas the modified remainder set aware version chose the order $\{A_0, A_1, B_0, B_1, C, I\}$.

4.1 Experimental settings

Experimental work is now presented in order to assess the described modification with a group of four sequential algorithms, using some well known datasets from the UCI repository of machine learning databases [2]. The family $PTA(l, r)$ has been selected to carry out the experiments, comparing their original versions and the modified ones, which are aware of the remainder set of non-selected features. Four different combinations of values for l and r have been tested as seen on Table 1. Note SFG can be seen as a particular case of $PTA(l, r)$ with $l = 1$ and $r = 0$ and referred

Table 1: Tested values for the $PTA(l, r)$ parameters (forward and backward steps)

Algorithm	Fwd st.	Bwd st.
$PTA(0,1) \equiv SBG$	0	1
$PTA(1,0) \equiv SFG$	1	0
$PTA(1,2)$	1	2
$PTA(2,1)$	2	1

to as $PTA(1, 0)$. The same can be done for SBG calling it $PTA(0, 1)$. The value of k in eq. (4) was set to 0.8 after some preliminary experiments and should be taken only as an educated guess.
Each pair of algorithms has been tested with the following datasets:

Ionosphere Classification of radar returns from the ionosphere. There are 2 classes, 351 instances, 34 numeric features. The targets were free electrons in the ionosphere. "Good" radar returns are those showing evidence of some type of structure in the ionosphere. "Bad" returns are those that do not: their signals pass through the ionosphere.

Mammogram Mammography data donated by the Pattern Recognition and Image Modeling Laboratory at University of California, Irvine. There are 86 cases with 65 features each and a binary class indicating benign or malignant.

Spect The dataset describes diagnosing of cardiac Single Proton Emission Computed Tomography (SPECT) images. Each of the patients is classified into two categories: normal and abnormal. There are 22 binary features extracted from the original SPECT images and 267 instances.

Spectf The same data as the previous dataset but this time a continuous feature pattern of size 44 was created for each patient. The same binary class and the same 267 instances.

Sonar There are 208 patterns obtained by bouncing sonar signals off a metal cylinder and rocks at various angles and under various conditions. Each pattern is a set of 60 numbers in the range 0.0 to 1.0. Each number represents the energy within a particular frequency band, integrated over a certain period of time. The class is binary indicating whether the object was a rock or a metal cylinder.

Waveform Artificial dataset where each class is generated from a combination of 2 of 3 "base" waves. There are 5000 instances with 21 features each, all of which include noise, and 3 classes.

Wdbc Breast cancer databases obtained from the University of Wisconsin Hospitals, Madison from Dr. William H. Wolberg [7]. Features 2 through 10 have been used to represent instances. There are 699 instances with 10 features, each has one of 2 possible classes: benign or malignant.

The experiments were carried out by extending the YALE learning environment [8] in order to implement a conventional PTA and the modified remainder set aware version of it. Each experiment consisted of a feature selection chain with a 1-nearest neighbor learner (using Euclidean distance) and 5-fold cross-validation for estimating feature *usefulness*. The quantity reported is the mean classification error in the five test folds. It is important to mention that there was no stopping criterion in the experiments: forward methods run until all the features were selected and backward ones until all of them were removed. Then the best of the obtained sequence of subsets was returned. The results are displayed in Table 4. The table also shows the standard deviation for these values found in the cross-validation runs and the size of the final selected subsets.

Table 2: Summary results of the experiments. The results are from the point of view of the modified algorithms.

	# Feat.			
Error	=	>	<	Total
better	3%	28%	25%	56%
equal	19%	3%		22%
worse		8%	14%	22%
Total	22%	39%	39%	100%

Table 3: Summary results of the experiments for forward algorithms. The results are from the point of view of the modified algorithms.

	# Feat.			
Error	=	>	<	Total
better		44%	17%	61%
equal	17%	6%		22%
worse		6%	11%	17%
Total	17%	56%	28%	100%

4.2 Experimental results

Tables 2 and 3 show the summary results of the experiments for all the 8 data sets ans 4 algorithms, and for the forward versions of the algorithms only, respectively. The tables cross the number of features selected with the classification error.

Upon looking at the summary tables, the first fact to note from the experiment results is that the remainder aware version of the algorithms outperformed the conventional version in most of the cases. It is seen that performance is in general increased (as expressed by the chosen J) while keeping the number of selected features roughly equal (Table 2). The improvement is even greater if we only look at the *forward* versions of the algorithms (Table 3). In this particular case, performance is increased while lowering the number of selected features. This is mainly due to the fact that the forward methods can easyly make wrong decisions at early iterations as (almost) no feature interaction is taken into account when evaluating individual features. A clear example of this has been exposed at the beginning of this section with the CorrAl dataset. There SFG selected the correlated feature while SBG correctly discarded it as the interaction with the other more relevant features was taken into account. Whenever the conventional and the modified algorithm are in ties or very close to, the modified versions offer a solution with a lower number of features, which is also interesting from the point of view of feature selection (there is an exception to this rule for the particular case of the Sonar data set and PTA (2,1)). The detailed experiment results are displayed in Table 4.

5 Conclusions

This paper has presented a modification for feature subset selection algorithms that iteratively evaluate subsets of features, by making them compute not only the *usefulness* of the selected set but also the *usefulness* of the remainder set. A set of experiments have been conducted in order to compare the modified versions of the algorithms with their original versions. Our experimental results indicate a general improvement in performance while keeping the size of the final subset roughly equal or lower. The fact that the modified version does not always improve the results of the original should not be a surprise. According to the *No free lunch* theorems, if an algorithm achieves superior results on some problems, it must pay with inferiority on other problems. However, it is possible to modify a search algorithm to obtain a version that is generally superior in performance to the original version [11]. In the present situation this fact can be explained by the way the modified version selects subsets of features. For instance, given two features: One that makes a significant reduction of the performance of the remainder set and not a big change on the performance of the selected set. And one that increases the performance of the selected set a bit more than the first one but does not make a big change on the remainder one. A conventional algorithm would always select the latter while the mod-

Table 4: Detailed Experiment Results. The error shown is the average of test set error in the 5 folds, while σ is the standard deviation of these values. Figures in boldface correspond to improvements.

Dataset	**Steps**		**Conventional Algorithm**			**Modified Algorithm**		
	fw	bw	error	σ	#Features	error	σ	#Features
corrAl	0	1	0,00%	0,00%	4	0,00%	0,00%	4
corrAl	1	0	3,20%	6,40%	5	**0,00%**	0,00%	4
corrAl	1	2	0,00%	0,00%	4	0,00%	0,00%	4
corrAl	2	1	0,00%	0,00%	4	0,00%	0,00%	4
Ionosphere	0	1	9,68%	2,41%	9	**7,12%**	2,85%	13
Ionosphere	1	0	**6,27%**	2,15%	11	7,70%	1,95%	7
Ionosphere	1	2	**7,11%**	2,33%	12	7,42%	2,12%	7
Ionosphere	2	1	6,26%	1,09%	14	**5,11%**	2,44%	7
Mammogram	0	1	**15,03%**	5,65%	4	16,27%	2,29%	29
Mammogram	1	0	12,75%	6,74%	17	**12,68%**	5,29%	15
Mammogram	1	2	14,97%	7,41%	13	**11,57%**	5,05%	13
Mammogram	2	1	11,57%	3,42%	26	**8,10%**	4,61%	22
Spect	0	1	20,59%	1,07%	1	**20,22%**	1,81%	4
Spect	1	0	23,23%	3,11%	6	**20,59%**	1,07%	1
Spect	1	2	23,21%	1,38%	10	**20,60%**	1,68%	7
Spect	2	1	**21,35%**	1,92%	5	22,09%	2,68%	6
Spectf	0	1	20,98%	4,04%	11	**18,72%**	5,52%	13
Spectf	1	0	19,48%	4,98%	10	**17,97%**	3,85%	6
Spectf	1	2	20,58%	3,00%	22	**16,48%**	4,34%	27
Spectf	2	1	20,59%	3,53%	8	**17,64%**	5,07%	24
Sonar	0	1	13,94%	5,72%	39	**10,10%**	4,12%	28
Sonar	1	0	8,19%	4,70%	42	**7,21%**	2,61%	29
Sonar	1	2	12,02%	3,99%	22	**9,11%**	3,11%	26
Sonar	2	1	**7,20%**	5,42%	36	10,56%	4,87%	42
Waveform	0	1	**20,60%**	0,83%	18	21,32%	1,26%	17
Waveform	1	0	21,16%	0,85%	16	**20,60%**	0,83%	18
Waveform	1	2	21,00%	1,22%	15	21,00%	1,22%	15
Waveform	2	1	21,62%	1,11%	15	**21,14%**	0,48%	17
Wdbc	0	1	5,27%	2,41%	11	**4,39%**	2,66%	25
Wdbc	1	0	3,86%	2,39%	24	**3,34%**	2,17%	13
Wdbc	1	2	**3,51%**	3,04%	13	4,04%	1,89%	27
Wdbc	2	1	3,86%	1,80%	27	3,86%	2,26%	14

ified version would maybe select the former. That could lead the modified version to avoid local maxima by not selecting the best feature in this iteration feature and end with a better subset; but when the algorithm has selected a set close to optimal subset, the modification may cause the algorithm to loose precision in choosing features. This loss of precision can be greater when the remainder set of features is very small compared with the selected set so few feature interactions are taken into account in this remainder set. Thus a future line of work is to make the weight of the remainder set performance vary with its size to compensate for this fact.

References

[1] M. Ben-Bassat. *Use of Distance Measures, Information Measures and Error Bounds in Feature Evaluation*, volume 2, pages 773–791. North Holland, 1982.

[2] C.L. Blake D.J. Newman, S. Hettich and C.J. Merz. UCI repository of machine learning databases, 1998.

[3] Mark A. Hall. *Correlation-based Feature Selection for Machine Learning*. PhD thesis, University of Waikato, 1999.

[4] George H. John, Ron Kohavi, and Karl Pfleger. Irrelevant features and the subset selection problem. In William W. Cohen and Haym Hirsh, editors, *Machine Learning, Procs. of the 11th Intl. Conf.*, pages 121–129, Rutgers University, New Brunswick, NJ, USA, July 10-13 1994. Morgan Kaufmann.

[5] Ron Kohavi and George H. John. Wrappers for feature subset selection. *Artif. Intell.*, 97(1-2):273–324, 1997.

[6] P. Langley. Selection of relevant features in machine learning. In *Procs. of the* AAAI *Fall Symposium on Relevance*, pages 140–144, New Orleans, LA, USA, 1994. AAAI Press.

[7] O. L. Mangasarian and W. H. Wolberg. Cancer diagnosis via linear programming. 23(5):1–18, 1990.

[8] Ingo Mierswa, Michael Wurst, Ralf Klinkenberg, Martin Scholz, and Timm Euler. Yale: rapid prototyping for complex data mining tasks. In Tina Eliassi-Rad, Lyle H. Ungar, Mark Craven, and Dimitrios Gunopulos, editors, *Proceedings of the Twelfth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Philadelphia, PA, USA, August 20-23, 2006*, pages 935–940. ACM, 2006.

[9] Pavel Pudil, Jana Novovicová, and Josef Kittler. Floating search methods in feature selection. *Pattern Recognition Letters*, 15(11):1119–1125, 1994.

[10] S.D. Stearns. On selecting features for pattern classifiers. In *Procs. of the 3rd Intl. Conf. on Pattern Recognition (ICPR 1976)*, pages 71–75, Coronado, CA, 1976.

[11] David Wolpert and William G. Macready. No free lunch theorems for optimization. *IEEE Trans. Evolutionary Computation*, 1(1):67–82, 1997.