Examen Parcial #2- 26/06/2021

GARS: Genethic Algorithm for the identification of a Robust Subset of features in high-dimensional datasets

Cavasin Nicolás - Legajo #143501

Resumen:

La etapa de selección de features (FS de ahora en más) en los modelos de clasificación intenta encontrar las variables que mejor representen los datos (que aporten mayor varianza) con el objetivo de poder construir un clasificador robusto y evitar sobre-entrenarlo. En la actualidad existen varios métodos para realizar FS y se pueden agrupar generalmente en tres grandes categorías:

- Métodos basados en filtros:
  a. Se basan en estadísticas univariadas (de una sola variable) + medidas de correlación o basadas en entropía.

- Métodos *wrapper* o envolventes:
  a. Combinan algoritmos de búsqueda con modelos de clasificación.

- Métodos *embedded* o embebidos:
  a. A diferencia del resto, FS se realiza dentro de la construcción del modelo en lugar de hacerlo en un paso previo de *preprocesamiento*.

Otra manera de clasificarlos es considerando su complejidad computacional como si fueran problemas de búsqueda, lo que permite agruparlos en otras tres nuevas categorías: exhaustivos, heurísticos y métodos híbridos.

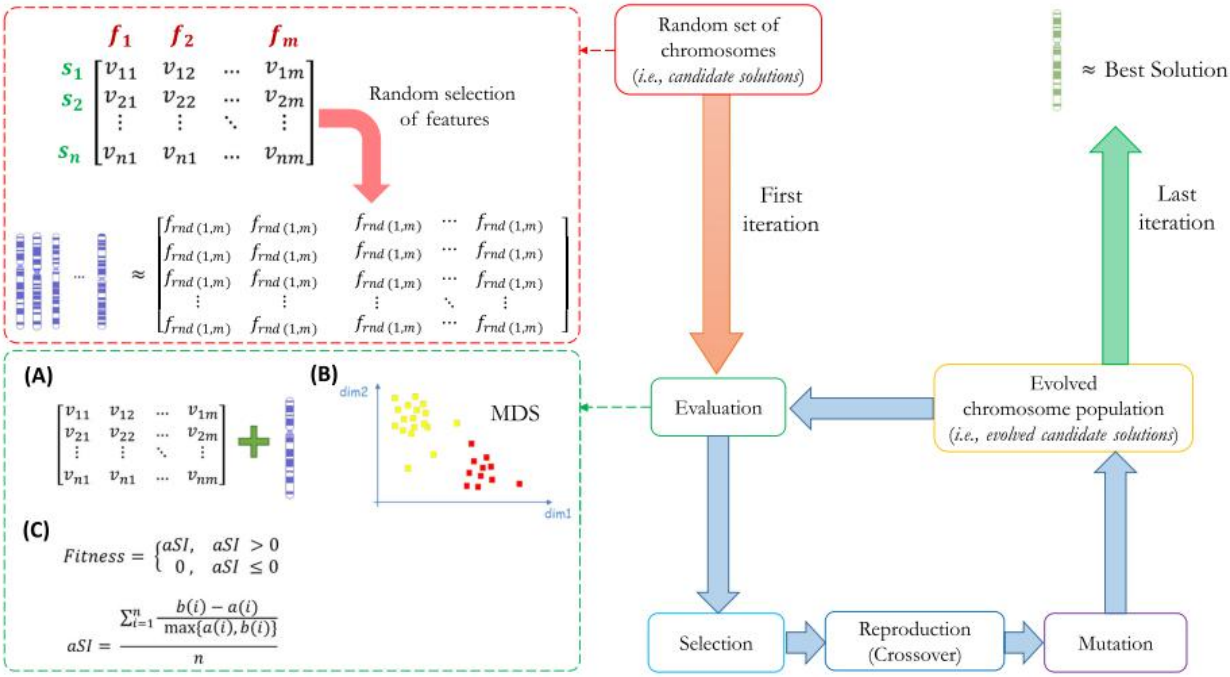
- Métodos exhaustivos:
  a. Demandan de mucha capacidad computacional, pues funcionan con "fuerza bruta", y prueban todas las posibles combinaciones de features hasta encontrar la más óptima. Generalmente los métodos *wrapper* y *embedded* suelen ser exhaustivos.
  b. Un ejemplo de esto es el método *GridSearchCV* provisto por la renombrada librería *scikit-learn* del lenguaje Python.

- Métodos heurísticos:
  a. Dada una función heurística, apuntan optimizar el problema mejorando la solución iterativamente.
  b. Un ejemplo de este método es el caso de los Sistemas Expertos, vistos anteriormente en clase.

- Métodos híbridos:
  a. Son combinaciones secuenciales de alguna de las diferentes categorías mencionadas antes.
  b. El caso más común es el que combina un método basado en filtro con un método *wrapper*.

Los algoritmos genéticos (GA de ahora en más) son algoritmos de búsqueda heurísticos y adaptativos que caen en la categoría de métodos *wrapper*. Intentan aplicar la ley del biólogo Charles Darwin -la supervivencia del más apto- con el fin de obtener la mejor combinación de features y lograr así reducir sustancialmente la dimensionalidad del dataset. Para ello somete al dataset a un proceso de cinco pasos: *population* (1), evaluación de aptitud (2), selección del más apto (3), entrecruzamiento y reproducción (4), mutación (5). Finalizado el proceso, los cromosomas supervivientes se consideran *evolucionados* -si son diferentes respecto a la población anterior- y vuelven a ser sometidos al mismo proceso hasta lograr la *convergencia*. Es decir, el proceso se repite hasta que en los cromosomas supervivientes no se observen diferencias respecto a la población de la iteración anterior.

En este caso, los autores proponen una estrategia que les permite sortear limitaciones como el costo computacional (por ser un método *wrapper*) y el sobre-entrenamiento que suele sufrir no solo un GA sino una gran cantidad de clasificadores de Machine Learning (ML de ahora en más). En *GARS* los pasos evolucionarios -pasos 3, 4 y 5- se realizan con los métodos más populares/tradicionales.



Workflow de GARS

Sin embargo, una característica distintiva de *GARS* es lo que ocurre en el paso 2 de cada iteración, en la evaluación de aptitud, pues es desdoblado en dos sub-tareas: escalado multi-dimensional (MDS) de los cromosomas presentes + promediado del resultado de la técnica Silueta aplicado sobre las primeras dos coordenadas obtenidas por MDS para cada cromosoma. Al valor resultante, que determina el grado de aptitud de cada cromosoma, se lo acota al intervalo [0; 1] donde 1 representa la aptitud máxima.

Para evaluar el rendimiento de *GARS*, los autores implementaron tres métodos de FS basados en ML para testear su estrategia: un método basado en un filtro univariado llamado *SBF*, un método *wrapper* que utiliza una estrategia de eliminación recursiva llamado *RFE* y un método *embedded* de regresión denominado *Least Absolute Shrinkage and Selection Operator* (LASO). Además, agregaron dos GAs a la batería de pruebas cuya función de aptitud es calculada por dos métodos de clasificación: uno mediante *Suppor Vector Machine* (SVM) llamado *svmGA* y otro mediante *Random Forests* (RF) llamado *rfGA*.

Por último seleccionaron datasets de diferentes tipos y densidades, todos relacionados a casos clínicos reales, para ser utilizados de manera acorde a cada método de testeo:

- Dataset binario de dimensionalidad baja:
  - Un dataset de 58 samples y 168 features.
  - Basado en un experimento que investigó las causas de la desregulación de micro secuencias de ARN en tejidos cancerígenos cervicales [1] .
- Dataset binario de dimensionalidad media:
  - Un dataset con 98 samples y "*pretty high number*" features (no indican el número exacto).
  - Basado en un experimento de resonancias magnéticas nucleares donde se estudiaron características metabólicas en pacientes con daños agudos de riñón [2].
- Dataset multi-clase de dimensionalidad alta:
  - Un dataset con samples de tejidos de 11 regiones del cerebro diferentes y 19.162 features que sub-dividió en 5 distintos para testear con diferentes parámetros.

- Basado en la explotación de un proyecto que almacena la características de 53 tipos de tejidos cerebrales distintos obtenidos gracias al aporte de más de 700 donantes[3].

Los autores indicaron que para el primer dataset *GARS* comparte el primer lugar con *LASO* ya que las caraterísticas elegidas son de 1,5 a 6 veces menores en comparación al resto y alcanzaron a representar un 98% del total de los datos. Sin embargo, perdió contra *LASO* en el tiempo de procesamiento utilizado para encontrar dicha cantidad de features.

En el segundo dataset *GARS* alcanzó la mayor precisión de clasificación con un 73% y además obtuvo el mejor balance entre sensitividad y especificidad (es decir, evitó el sobre-entrenamiento) con tan solo 7 features seleccionadas.

Para los sub-datasets derivados del último dataset, *GARS* obtuvo un rendimiento muy bueno el cual mejoró por cada sub-dataset con mayor cantidad de clases que procesaba. *svmGA* y *rfGA* fueron descartados por no poder completar un solo folding pasadas las 24hs de su ejecución. Ante esta situación, se volvió a manipular el dataset para lograr la convergencia de estos dos últimos métodos y se volvió a realizar la comparación. *GARS* no solo logró siempre cubrir el mayor porcentaje de datos totales sino que también fué siempre el que menos features utilizó. Solo fué superado en un 3-6% de precisión por *LASO* (quién utilizó hasta 7 veces más features).

Crítica:

La selección y extracción de features es un tópico de suma relevancia en campos de ML donde se procesan datasets de enormes dimensionalidades diariamente. Sin ir más lejos, participo ad-honorem de un proyecto de nuestra universidad que utiliza un dataset de 24.700 mails y se estudian diferentes técnicas de selección y extracción de features para lograr construir un clasificador. El FS en este caso es crítico porque permite otorgar contexto a cada mail pero a la vez es muy sensible a errores ajenos como por ejemplo las faltas de ortografía o abreviaciones, casos que impiden una selección óptima. Es muy interesante saber que existen este tipo de algoritmos robustos que logran lo mejor de ambos mundos al poder reducir drásticamente dimensionalidades y además seleccionan features que cubren un gran porcentaje de los datos totales.

GARS logra evitar tres puntos clave que generalmente es sufrido por grandes cantidades de clasificadors en el ámbito de ML: simplificación del modelo que limita el sobre-entrenamiento, reducción de capacidad de cómputo, segregación bastante precisa de los samples.

Como comentario, me hubiera resultado divertido observar como se comportaba contra un dataset que esté relacionado a otro ámbito que no sea el de estudios clínicos. Por nombrar alguno, alimentarlo de un dataset de rostros como *YouTube Faces* (YTF) y observar cuánto tiempo demora en converger, cuántas features selecciona y qué porcentaje de datos cubre.

GARS está escrito en R, un lenguaje bastante popular en el espacio del ML y se deja en [5] una referencia a su librería.

Referencias:

[1] - Ultra-high throughput sequencing-based small RNA discovery and discrete statistical biomarker analysis in a collection of cervical tumours and matched controls:
https://bmcbiol.biomedcentral.com/articles/10.1186/1741-7007-8-58

[2] - Analysis of human urine reveals metabolic changes related to the development of acute kidney injury following cardiac surgery:
https://www.researchgate.net/project/NMR-based-metabolomics-studies-of-Acute-Kidney-Injury-after-cardiac-surgery-with-Cardiopulmonary-Bypass-use

[3] - The Genotype-Tissue Expression (GTEx) project: https://www.nature.com/articles/ng.2653

[4] - YouTube Faces: https://www.cs.tau.ac.il/~wolf/ytfaces/

[5] - Librería de GARS: https://bioconductor.org/packages/release/bioc/html/GARS.html