Universidad Simón Bolívar
Decanato de Estudios Profesionales
Coordinación de Ingeniería de la Computación

Título

Por:
Antonio Álvarez

Realizado con la asesoría de:
Emely Arráiz B.

PROYECTO DE GRADO
Presentado ante la Ilustre Universidad Simón Bolívar
como requisito parcial para optar al título de
Ingeniero de Computación

Sartenejas, septiembre de 2018

Resumen

Hola mundo

Índice general

Índice de figuras

Índice de Tablas

Índice de algoritmos

Acrónimos y Símbolos

KDD	Knowledge Discovery in Databases
DM	Data Mining
IS	Instance Selection
PS	Prototype Selection
NN	Nearest Neighbor
NE	Nearest Enemy
CNN	Condensed Nearest Neighbor
ENN	Edited Nearest Neighbor
RSS	Relaxed Selective Subset
GGA	Generational Genetic Algorithm
SGA	Steady-State Genetic Algorithm
CHC	CHC Adaptive Search Algorithm
MEM	Memetic Algorithm

$\in$	Relación de pertenencia, «*es un elemento de*»
$\subseteq$	Subconjunto

Introducción

Introdución

Capítulo 1

Marco teórico

1.1. Descubrimiento de Conocimiento y preprocesamiento de datos

Hoy en día, existe una creciente necesidad de procesar grandes volúmenes de datos, estos datos son producto de la recolección de información de procesos y actividades de distintas índoles y se vuelven un material valioso para extraer información sobre posibles tendencias que puedan existir en dichos procesos. Es aquí donde entra el descubrimiento de conocimiento en bases de datos (KDD por su siglas en inglés) como disciplina encargada del procesamiento de datos para la extracción de información.

KDD es definida por *Smyth, P. et al.* [FSS96] como "el proceso no trivial de identificar patrones en los datos que sean válidos, novedosos, potencialmente útiles y finalmente entendibles". Para este fin, KDD se subdivide en distintas etapas a llevar a cabo para lograr el fin último de identificar patrones, éstas son [GLH16]: especificación del problema, entendimiento del problema, preprocesamiento de los datos, minería de datos, evaluación de los resultados y explotación de los resultados. En este trabajo es de especial interés la etapa de preprocesamiento de datos.

El preprocesamiento de datos consiste en el conjunto actividades destinadas a preparar los datos para ser usado por un algoritmo de minería de datos (DM por sus siglas en ingés). Las actividades realizadas en el preprocesamiento pueden ser clasificadas como actividades para la preparación de los datos y la reducción de los mismos [GLH16].

La preparación de datos es un paso obligatorio en el preprocesamiento, ya que transforma los datos, que inicialmente son inservibles para el algoritmo de DM por asuntos como la presencia de atributos faltantes en instancias, datos erróneos y atributos con formatos no aceptables para el algoritmo a utilizar [GLH16]. Dependiendo del enfoque dado, estas actividades pueden clasificarse en:

- **Limpieza de datos [GLH16, KCH$^+$03]:** incluye el tratamiento de los atributos faltantes y los datos erróneos, que si se dejan sin tratar resulta en un modelo de minería de datos poco confiable. Un atributo faltante en una instancia resulta de no haberlo introducido al momento del registro o por la pérdida en el proceso de almacenamiento. Los datos con atributos faltantes pueden tratarse de 3 maneras [FKP07]: la eliminación de las instancias que presenten el problema, utilizar métodos de estimación de máxima verosimilitud para calcular promedios y variancias con lo cual llenar los atributos faltantes y utilizar algoritmos del repertorio de machine learning como k-nn, k-means o Suport Vector Machine para estimar el valor de los atributos faltantes.

  Por su parte, los datos erróneos (también conocidos como datos ruidosos) pueden venir de dos formas [CAB11]: ruido de clase cuando la instancia está mal clasificada y ruido de atributo cuando uno o más valores de los atributos en una instancia están distorsionados y no representan la realidad. Para tratar los datos ruidosos se puede usar 3 métodos: construir algoritmos de DM que no se vean afectados en cierta medida ante el ruido (sean robustos), pulir los datos [Ten99] de tal manera que se corrijan los errores y por último se puede identificar los datos ruidosos para eliminarlos del conjunto y así quedarse sólo con datos correctos [BF99]. Cada uno de estos métodos tiene sus ventajas y desventajas; si sólo se cuenta con lo robusto del algoritmo de clasificación o regresión se tendrá un nivel de tolerancia del cual, al pasarse los resultados serán inservibles, pulir los datos sólo es aplicable a conjuntos de tamaño pequeño y mediano debido al alto costo computacional que tienen los algoritmos que hacen el trabajo y si se decide filtrar todos los datos ruidosos se puede disminuir considerablemente el conjunto hasta un punto que no sea utilizable por los algoritmos de clasificación y regresión; por lo tanto, lo que se estila es usar una combinación en lo que sea posible de estos 3 métodos para obtener los mejores resultados

- **Transformación de datos [GLH16]:** se centra en aplicar fórmulas matemáticas a los valores de los atributos para así obtener valores sintéticos que pueden proporcionar más información respecto a la instancia y al conjunto que pertenencen. Las transformaciones más comunes son la lineal y la cuadrática, la primera se usa principalmente para combinar distintos atributos y así crear uno sintético para ser usado por el algoritmo de DM, la transformación cuadrática por su parte, es usada cuando una transformación lineal no es suficiente para derivar información útil de los atributos. En este sentido, existen otros tipos de transformaciones como la polinomial, que engloba a la lineal y a la cuadrática y la no polinomial que trata con transformaciones más complejas.

- **Integración de los datos [GLH16, BLN86]:** consiste en la unión de los conjuntos de datos provenientes de distintas fuentes en un único conjunto. La integración tiene que tomar en cuenta algunos aspectos que se pueden presentar durante el proceso, entre ellos están la redundacia de atributos, la cual sucede cuando 2 atributos están fuertemente correlacionados y por lo tanto, con tener uno de ellos se puede derivar el otro. La redundancia de atributos puede traer consigo un sobre ajuste (overfitting) de los modelos predictivos, además de aumentar el tiempo de cómputo de los mismos, es por eso que se debe eliminar esta redundancia y para ello se usa una prueba de correlación χ^2 con el fin de identificar los atributos redundantes y así decidir con cual quedarse.

  Continuando, con los problemas que se pueden presentar al momento de la integración, se tiene también la duplicación de instancias, problema que normalmente trae consigo la inconsistencia en los valores de los atributos, debido a las diferencias con las que se registran los valores. Para solucionar este asunto primero se tiene que identificar las instancias duplicadas usando técnicas que midan la similitud entre ellas, como la propuesta de *Fellegi, I. & Sunter, A.* [FS69] que lo modela como un problema de inferencia bayesiana o como en [CKLS01] donde se usan árboles de clasificación y regresión (CART por sus siglas en inglés) para cumplir este trabajo.

- **Normalización de datos [GLH16]:** busca cambiar la distribución de los datos originales de tal manera que se acoplen a las necesidades de los algoritmos predictivos. Dos de los tipos de normalización más usadas son la normalización

min-max en la cual se aplica la fórmula en la ecuación 1.1, donde max_A es el valor máximo del atributo sobre los valores en el conjunto, min_A es el valor mínimo existente, $nuevo\_max_A$ y $nuevo\_min_A$ son los nuevos rangos para el atributo:

$$v' = \frac{v - min_A}{max_A - min_A}(nuevo\_max_A - nuevo\_min_A) + nuevo\_min_A \tag{1.1}$$

El otro tipo de normalización es la puntuación Z (Z-score) en donde se llevan los datos a promedio 0 y desviación estándar 1 aplicando la fórmula de la ecuación 1.2:

$$v' = \frac{v - \mu_A}{\sigma_A} \tag{1.2}$$

.

Pasando a la reducción de los datos, se tiene que engloba todas las técnicas que reducen el conjunto de datos original para obtener uno representativo con el cual trabajar en los modelos predictivos. La reducción de datos cobra especial importancia cuando se tienen conjuntos muy grandes que retardarían en gran medida el tiempo de cómputo de los algorimtos que los van a usar. Las técnicas de reducción de datos son [GLH16]:

- **Discretización de datos [GLH16, GLS$^+$13]:** es el proceso de transformar datos numéricos en datos categóricos, definiendo un número finito de intervalos que representan rangos entre distintos valores consecutivos con el fin de poder tratarlos como valores nominales. Es de especial importancia conseguir el número correcto de intervalos que mantengan la ínformación original de los datos, ya que muy pocos intervalos puede llegar a ocultar la relación existente entre un rango en específico y una clase dada y muchos intervalos puede llevar a un sobre ajuste [CPSK07]. El principal atractivo de la discretización es que permite utilizar un algoritmo de DM que trabaje principalmente con datos nominales como Naïve Bayes [YW09] a partir de datos numéricos. Para un estudio más completo de la discretización se referencia a [GLS$^+$13].

- **Selección de características [GLH16, LM12]:** busca eliminar atributos que sean redundantes o irrelevantes de tal manera que el subconjunto de características restantes mantenga la distribución original de las clases. El proceso de selección de características tiene ventajas, como mantener e incluso mejorar la precisión de los modelos predictivos, reducir los tiempos de cómputo y reducir la complejidad de los modelos resultantes. La búsqueda de un subconjunto de atributos puede realizarse de 3 maneras: búsqueda exhaustiva, búsqueda heurística y métodos no determinísticos. La búsqueda exhaustiva cubre todo el espacio de soluciones, normalmente van probando todas las combinaciones posibles de atributos para conseguir el que mejor se acople a la métrica a optimizar, entre los métodos exhaustivos están Focus [AD91], Automatic Branch & Bound [LMD98], Best First Search [XYC88], entre otros. Por su parte, la busqueda heurística busca una solución aproximada a la óptima en poco tiempo, entre sus métodos están los propuestos en [DL97, KS96, Bat94]. Por último, están los métodos no determinísticos, de entre los que destacan los algoritmos géneticos, recocido simulado y Las Vegas Filter [LS$^+$96].
- **Selección de instancias [GLH16]:** consiste en elegir un subconjunto de las instancias totales manteniendo las características del conjunto original. Es el problema a tratar en este trabajo y se elabora más sobre el mismo en la siguiente sección.

1.2. Selección de Instancias y Selección de Prototipos

La selección de instancias (IS por sus siglas en inglés) consiste en reducir el conjunto de datos dado a un conjunto reducido que va a ser utilizado con un algoritmo de clasificación o regresión, manteniendo el desempeño del algorimto como si se usara el conjunto original.

Definición 1. Dado un conjunto de datos X, se tiene que una instancia $X_i = (X_i^1, X_i^2, \ldots, X_i^p)$ donde X_i^j es el atributo j para la instancia X_i con $X_i \in X$ y siendo p el número de atributos. La instancia X_i es de clase Y_j donde $Y_j \in Y$, siendo Y el conjunto de todas las clases definidas con $j \in (1 \ldots q)$ donde q es el número de clases

totales. Se divide el conjunto X en un conjunto TR de entrenamiento y un conjunto TS de prueba. El problema de **Selección de Instancias** consiste en conseguir un conjunto $S \subseteq TR$ con el cual, al usarse con el clasificador T se obtengan los mismos valores de precisión o mejores que al usar T con TR [GLH16].

La respuesta óptima de un método de selección de instancias es un conjunto *consistente* y de cardinalidad mínima.

Definición 2. "Un conjunto R es **consistente** con T, *si y solo si* toda instancia $t \in T$ es clasificada correctamente mediante el uso de un clasificador M y las instancias en R como conjunto de entrenamiento." [Ale14]

Sin embargo, conseguir la respuesta óptima es un problema NP-Duro (NP-Hard) como lo demuestra *Zukhba* en [Zuk10]. Por lo tanto, la mayoría de los métodos propuestos hasta la fecha se enfocan en obtener una solución aproximada.

El problema de selección de instancias se puede enfocar como un problema de selección de prototipos (PS por sus siglas en inglés). PS es en esencia IS con el detalle de que el clasificador T usado es un clasificador basado en instancias [GLH16]. De los cuales K Vecinos más Cercanos (KNN por sus siglas en inglés) es el más conocido y será usado como clasificador para este trabajo.

1.2.1. Regla de K vecinos más cercanos

Inicialmente propuesta por *Fix, E. & Hodges, J.* en [FHJ51]. La regla KNN clasifica instancias a partir de los datos adyacentes; esto viene dado bajo el razonamiento de que una instancia probablemente comparta la misma clase que sus vecinos. Formalmente, el algoritmo de clasificación usando KNN se puede definir como:

Definición 3. Sea X un conjunto de datos con $X_i \in X$ una instancia del conjunto, con clase $Y_{X_i} \in Y$ la clase a la cual pertenece, siendo Y el conjunto de las clases presentes en los datos. Sea $\pi_1(X_i) \dots \pi_n(X_i)$ un reordenamiento de las n instancias que conforman el conjunto X de acuerdo a la distancia a la que se encuentren de la instancia X_i, usando una métrica de distancia dada $\rho : \chi \times \chi \to \mathbb{R}$, donde χ es el dominio de las

instancias en X, tal que $\rho(X_i, \pi_k(X_i)) \leq \rho(X_i, \pi_{k+1}(X_i))$. Para clasificar una instancia X_j se usa la clase de la mayoría perteneciente al conjunto $\left\{Y_{\pi_i(X_j)} \mid i \leq k\right\}$ siendo k el número de vecinos que se toma en consideración. [SSBD14]

Lo simple del algoritmo ha impulsado KNN a ser uno de los algoritmos de DM más usados y consigo ha traido numerosos estudios sobre el comportamiento de convergencia y acotaciones sobre el error en la clasificación. Entre dichos trabajos se encuentra el de *Cover, T. & Hart, P.* [CH67] donde muestran que la probabilidad de error R del clasificador NN está acotada por debajo por la probabilidad de error de Bayes R* y acotada por arriba por $R^*(\frac{2-MR^*}{M-1})$ cuando el número de instancias tiende al infinito y además la regla NN es admimsible en la clase de reglas KNN, esto quiere decir que no hay $k \neq 1$ para el cual la probabilidad de error R sea menor que para $k = 1$. Para un estudio más formal de las propiedades de convergencia se refiere a [DGL13].

Una implementación ingenua de KNN dado una instancia a clasificar q, consta de calcular la distantia de todos los puntos con respecto a q y reportar los k puntos más cercanos; esto tiene una complejidad de $O(dn)$ donde d es el número de atributos en una instancia y n es el total de instancias [SDI06]. Es por eso que mucho de los esfuerzos de la investigación de KNN es encontrar estructuras de ordenado y almacenamiento que lleven la complejidad a un orden sublineal o inclusive logarítmico; entre ellas están:

- **Árboles KD [SDI06, Ben75]:** también conocidos como KD Trees en inglés, se construyen de la siguiente manera: dado n puntos en un conjunto P en un espacio d-dimensional, primero se calcula la mediana M de los valores del i-ésimo atributo de los n puntos (inicialmente i = 1) y con este valor M se particiona el conjunto P en P_L como el conjunto con puntos cuyo valor del i-ésimo atributo es menor a M y P_R como el conjunto de puntos cuyo valor del i-ésimo atributo es mayor o igual a M. En la siguiente iteración se elige otro atributo i y se particuionan P_L y P_R en dos cada uno. El proceso se repite hasta que el conjunto de puntos en un nodo del árbol construido llegue a tener cardinalidad 1. El tiempo de construcción del árbol es de O(nlog(n)) y el tiempo de búsqueda en G(d)log(n) dado una función G la cual es exponencial en d, cabe destacar que el tiempo de búsqueda es al lo sumo O(dn).

- **Árboles de esfera [SDI06, Omo89, Uhl91]:** los árboles de esfera (balltrees en inglés) son árboles binarios donde las hojas corresponden a las instancias y cada nodo interior del árbol corresponde a una esfera en el espacio de los datos, cada esfera requiere ser la más pequeña que contenga las esferas asociadas a los nodos hijos. En contraste con los árboles KD, las regiones asociadas entre nodos vecinos en los árboles de esfera pueden intersectarse y no tienen que cubrir la totalidad del espacio, lo que permite una cobertura más flexible que refleje la estructura inherente a los datos.
- **Hashing sensitivo a la localidad [SDI06, Ind04]:** la idea principal detrás del Hashing Sensitivo a la localidad (LSH por sus siglas en inglés) es realizar un hashing con los datos usando varias funciones de hash de tal manera que la probabilidad de colisión entre dos puntos sea mayor mientras más cerca estén uno del otro usando una métrica de distancia. Entonces, una vez construida la tabla de hash, se puede determinar los vecinos más cercanos retornando los elementos en el contenedor correspondiente a su valor calculado por la función de hash.

Un concepto que está presente al momento de clasificar usando KNN es la llamada maldición de dimensionalidad, ésta afecta la clasificación de dos maneras: la primera estipula que un pequeño incremento en las dimensiones de los datos trae consigo un gran aumento en el número de instancias necesarias para mantener la misma precisión del clasificador. La segunda, por su parte, menciona que para metodos de almacenamiento como los árboles KD y los árboles esfera un aumento en las dimensiones tiende a degradar su desempeño a una búsqueda lineal como la implementación ingenua [KM17]. Lo cual afecta la aplicabilidad del algoritmo a datos de muy grandes dimensiones y abre un campo de estudio continuo a formas de optimización del ordenamiento y almacenamiento de las instancias.

1.2.2. Taxonomía del problema de selección de prototipos

En este trabajo se adopta la taxonomía propuesta por *García et al* en [GDCH12]. En ella se definen unas propiedades comunes de todos los algoritmos de PS con los

que se pueden comparar y así establecer la taxonomía. Sea TR el conjunto de entrenamiento y S el conjunto reducido, las propiedades son las siguientes:

1.2.2.1. Dirección de búsqueda

- **Incremental:** se empieza con un conjunto vacío S y se va añadiendo instancias de TR si cumple con cierto criterio. El orden de presentación de las instancias puede llegar a afectar el resultado final para muchos algoritmos, por eso se acostumbra a presentar los datos de manera aleatoria. Una búsqueda incremental tiene la ventaja de que puede seguir agregando instancias una vez finalizado un proceso de selección inicial, lo cual lo hace bastante atractivo para el aprendizaje continuo
- **Decremental:** la búsqueda empieza con S = TR y se va buscando instancias para remover de S. El orden de presentación sigue siendo importante, pero a diferencia de los métodos incrementales, se tiene todo el conjunto desde el inicio. Los algoritmos decrementales tienden a presentar un mayor costo computacional que los incrementales y el aprendizaje no puede continuar luego de terminar el lote inicial.
- **Por lote:** se elige un grupo y se evalúan todos los elementos del mismo para su eliminación, los que no pasen la prueba seleccionada son desechados a la vez. El proceso se repite con distintos lotes hasta terminar.
- **Mixto:** S empieza como un subconjunto pre selecionado (puede ser de manera aleatoria o usando un proceso incremental/decremental) e iterativamente puede añadir o remover instancias que cumplan con criterios en específico.
- **Fijo:** el número final de instancias en S se fija al principio de la fase de aprendizaje y se aplica una búsqueda mixta hasta cumplir con dicha cuota.

1.2.2.2. Tipo de selección

- **Condensación:** se busca mantener los puntos bordes (aquellos que están cercas de las fronteras entre las clases). El razonamiento es que son los puntos bordes

los que realmente determinan las fronteras, siendo más útiles al momento de clasificar una nueva instancia. Estos métodos tienden a reducir bastante el conjunto original ya que hay menos puntos bordes que interiores.

- **Edición:** los métodos de edición en cambio buscan remover los puntos bordes, suavizando las fronteras bajo la idea de que es el lugar donde se concentran la mayor cantidad de puntos ruidosos. Tienden a disminuir en menor medida el conjunto TR en comparación a los métodos de condensación.
- **Híbridos:** su principal objetivo es mantener la precisión del clasificador usando un conjunto lo más reducido posible. Para esto eliminan tanto puntos internos como los ruidosos en el borde, tomando las ideas principales de los métodos de condensación y edición.

1.2.2.3. Evaluación de la búsqueda

- **Filtro:** son los métodos que usan un conjunto parcial de datos para decidir cuáles remover o añadir sin usar un esquema de validación, donde se deja uno por fuera para probar con el resto de los datos en cada iteración del algoritmo. La simplifacación en la validación cruzada y el uso de un subconjunto de TR agiliza los cálculos realizados por el algortimo a costa de precisión.
- **Envolventes:** usan todo el conjunto TR en un proceso de validación cruzada, donde en cada iteración se va excluyendo cada instancia para evaluar si vale la pena eliminarla. Son métodos más costosos que los filtros, pero tienden a obtener una precisión mayor al momento de generalizar usando un algoritmo de DM.

Una vez expuestas las características con las que se pueden comparar distintos métodos de PS. Se presenta en la la figura 1.1 la clasificación que se le puede dar a los algoritmos. Para un estudio más extenso sobre los distintos algoritmos se recomienda leer [GLH16]

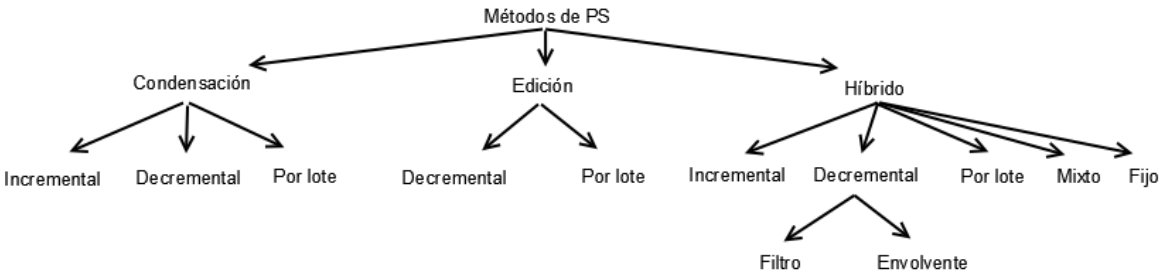

FIGURA 1.1: Taxonomía para los métodos de selección de prototipos

1.2.3. Heurísticas

En esta sección se exponen las heurísticas utilizadas en este trabajo. Citando a *Pearl, J.* en [Pea84]: "Una heurística es un criterio, método o principo para decidir cual, de entre varias alternativas de acciones a seguir, promete ser la más efectiva para alcanzar un objetivo". Para el caso de PS, dicho objetivo es alcanzar un buen aproximado del conjunto de cardinalidad mínima y máxima precisión en la clasificación. Sea $S \subseteq TR$ el conjunto reducido a devolver y TR el conjunto de entrenamiento:

1.2.3.1. Condensed Nearest Neighbor (CNN)

Propuesto inicialmente por *Hart, P.* en [Har68],CNN es un método de condensación incremental. El conjunto S se construye de tal manera que cada elemento de TR está más cerca de un miembro de S de la misma clase que un miembro de S de clase distinta. El algoritmo empieza seleccionando una instancia aleatoria x y se coloca en S (inicialmente vacío), acto seguido se empieza a clasificar todas las instancias de TR sólo usando como referencia las pertenecientes S, si una instancia es clasificada incorrectamente, se agrega a S, asegurando así que en la siguiente vuelta sea clasificada correctamente. Una vez aumentado S se vuelve a probar cada instancia de TR y se agruegan las que sean mal clasificadas. El proceso se repite hasta que no existan instancias en TR que se encuentren mal clasificadas. El algoritmo se presenta en 1.1.

Algoritmo 1.1 CNN

Input: `TR` conjunto de entrenamiento, `k` número de vecinos a ser considerado en la clasificación
Output: `S` conjunto reducido

```
1: S ← instancia aleatoria x
2: flag ← false
3: while ¬flag do
4:     for all x ∈ TR do
5:         Y ← k vecinos más cercanos a x pertenecientes a S // Si | S |< k elegir | S | elementos
6:         Clasificar x con la misma clase que sea mayoría en Y
7:         if x está mal clasificada then
8:             S ← S ∪ {x}
9:             Retornar a 3
10:    if Todas las instancias en TR fueron bien clasificadas then
11:        flag ← true
       return S
```

1.2.3.2. Edited Nearest Neighbor (ENN)

Propuesto por *Wilson, D.* en [Wil72] ENN es un método de edición decremental. Empieza con S = TR y se va iterando sobre las instancias de S, removiendo aquellas que no concuerdan con la clase de la mayoría de sus k vecinos más cercanos. El algoritmo se presenta en 1.2

Algoritmo 1.2 ENN

Input: `TR` conjunto de entrenamiento, `k` número de vecinos a ser considerado en la clasificación
Output: `S` conjunto reducido

```
1: S ← TR
2: for x ∈ S do
3:     Y ← k vecinos más cercanos a x pertenecientes a S
4:     if la clase de x es distinta a la clase mayoritaria en Y then
5:         Se elimina x de S
       return S
```

1.2.3.3. Relaxed Selective Subset (RSS)

Propuesto por *Flores, A. & Mount, D.* en [FM] se tiene que RSS es un algoritmo de condensación incremental con la particularidad de que no es sensible al orden de

presentación de las instancias, porque realiza un ordenamiento inicial de las mismas. El método primero ordena las instancias según la distancia que tengan a su enemigo más cercano (la instancia más cercana con clase distinta) de manera incremental (de la distancia más corta a la más larga). Luego, empezando con un conjunto S vacío, se van presentando las instancias en el orden establecido anteriormente y se agrega a S aquellas para las cuales no exita un punto $r \in S$ que esté a una distancia menor que la distancia que tiene r a su enemigo más cercano. Sea $d_{NE}(p)$ la distancia del punto p a su enemigo más cercano y sea $d(p_i, r)$ la distancia de un punto p_i a un punto r. El algoritmo se presenta en 1.3

Algoritmo 1.3 RSS

Input: `TR` conjunto de entrenamiento
Output: `S` conjunto reducido

1: $S \leftarrow \emptyset$
2: Sea $\{p_i\}_{i=1}^{n}$ los puntos en TR ordenados de manera ascendente respecto a $d_{NE}(p_i)$
3: **for all** $p_i \in TR$ **do**
4: **if** $\neg\exists r \in S$ tal que $d(p_i, r) < d_{NE}(r)$ **then**
5: $S \leftarrow S \cup \{p\}$
return S

Capítulo 2

Marco metodológico

2.1. Descripción general

Marco metodológico

Capítulo 3

Resultados

Resultados

Conclusiones y Recomendaciones

Conclusiones

Bibliografía

[AD91] Hussein Almuallim and Thomas G Dietterich. Learning with many irrelevant features. In *AAAI*, volume 91, pages 547–552, 1991.

[Ale14] Flores Alejandro. *Metaheurísticas Bio-Inspiradas para Selección de Instancias*. PhD thesis, Undergraduate thesis,Departamento de Ciencias de la Computación, Universidad Simón Bolívar, Venezuela, 2014.

[Bat94] Roberto Battiti. Using mutual information for selecting features in supervised neural net learning. *IEEE Transactions on neural networks*, 5(4):537–550, 1994.

[Ben75] Jon Louis Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9):509–517, 1975.

[BF99] Carla E Brodley and Mark A Friedl. Identifying mislabeled training data. *Journal of artificial intelligence research*, 11:131–167, 1999.

[BLN86] Carlo Batini, Maurizio Lenzerini, and Shamkant B. Navathe. A comparative analysis of methodologies for database schema integration. *ACM computing surveys (CSUR)*, 18(4):323–364, 1986.

[CAB11] Cagatay Catal, Oral Alan, and Kerime Balkan. Class noise detection based on software metrics and roc curves. *Information Sciences*, 181(21):4867–4877, 2011.

[CH67] Thomas Cover and Peter Hart. Nearest neighbor pattern classification. *IEEE transactions on information theory*, 13(1):21–27, 1967.

[CKLS01] Munir Cochinwala, Verghese Kurien, Gail Lalk, and Dennis Shasha. Efficient data reconciliation. *Information Sciences*, 137(1-4):1–15, 2001.

[CPSK07] Krzysztof J Cios, Witold Pedrycz, Roman W Swiniarski, and Lukasz Andrzej Kurgan. *Data mining: a knowledge discovery approach*. Springer Science & Business Media, 2007.

[DGL13] Luc Devroye, László Györfi, and Gábor Lugosi. *A probabilistic theory of pattern recognition*, volume 31. Springer Science & Business Media, 2013.

[DL97] Manoranjan Dash and Huan Liu. Feature selection for classification. *Intelligent data analysis*, 1(3):131–156, 1997.

[FHJ51] Evelyn Fix and Joseph L Hodges Jr. Discriminatory analysis-nonparametric discrimination: consistency properties. Technical report, California Univ Berkeley, 1951.

[FKP07] Alireza Farhangfar, Lukasz A Kurgan, and Witold Pedrycz. A novel framework for imputation of missing values in databases. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, 37(5):692–709, 2007.

[FM] Alejandro Flores and David M Mount. Nearest neighbor condensation with guarantees.

[FS69] Ivan P Fellegi and Alan B Sunter. A theory for record linkage. *Journal of the American Statistical Association*, 64(328):1183–1210, 1969.

[FSS96] Usama M Fayyd, Gregory P Shapiro, and Padhraic Smyth. From data mining to knowledge discovery: An overview. 1996.

[GDCH12] Salvador Garcia, Joaquin Derrac, Jose Cano, and Francisco Herrera. Prototype selection for nearest neighbor classification: Taxonomy and empirical study. *IEEE transactions on pattern analysis and machine intelligence*, 34(3):417–435, 2012.

[GLH16] Salvador García, Julián Luengo, and Francisco Herrera. *Data preprocessing in data mining*. Springer, 2016.

[GLS+13] Salvador Garcia, Julian Luengo, José Antonio Sáez, Victoria Lopez, and Francisco Herrera. A survey of discretization techniques: Taxonomy and

empirical analysis in supervised learning. *IEEE Transactions on Knowledge and Data Engineering*, 25(4):734–750, 2013.

[Har68] Peter Hart. The condensed nearest neighbor rule (corresp.). *IEEE transactions on information theory*, 14(3):515–516, 1968.

[Ind04] Piotr Indyk. Nearest neighbors in high-dimensional spaces. 2004.

[KCH+03] Won Kim, Byoung-Ju Choi, Eui-Kyeong Hong, Soo-Kyung Kim, and Doheon Lee. A taxonomy of dirty data. *Data mining and knowledge discovery*, 7(1):81–99, 2003.

[KM17] Eamonn Keogh and Abdullah Mueen. Curse of dimensionality. In *Encyclopedia of Machine Learning and Data Mining*, pages 314–315. Springer, 2017.

[KS96] Daphne Koller and Mehran Sahami. Toward optimal feature selection. Technical report, Stanford InfoLab, 1996.

[LM12] Huan Liu and Hiroshi Motoda. *Feature selection for knowledge discovery and data mining*, volume 454. Springer Science & Business Media, 2012.

[LMD98] Huan Liul, Hiroshi Motoda, and Manoranjan Dash. A monotonic measure for optimal feature selection. In *European conference on machine learning*, pages 101–106. Springer, 1998.

[LS+96] Huan Liu, Rudy Setiono, et al. A probabilistic approach to feature selection-a filter solution. In *ICML*, volume 96, pages 319–327. Citeseer, 1996.

[Omo89] Stephen M Omohundro. *Five balltree construction algorithms*. International Computer Science Institute Berkeley, 1989.

[Pea84] Judea Pearl. Heuristics: intelligent search strategies for computer problem solving. 1984.

[SDI06] Gregory Shakhnarovich, Trevor Darrell, and Piotr Indyk. *Nearest-neighbor methods in learning and vision: theory and practice (neural information processing)*. The MIT press, 2006.

[SSBD14] Shai Shalev-Shwartz and Shai Ben-David. *Understanding machine learning: From theory to algorithms*. Cambridge university press, 2014.

[Ten99] Choh-Man Teng. Correcting noisy data. In *ICML*, pages 239–248. Citeseer, 1999.

[Uhl91] Jeffrey K Uhlmann. Satisfying general proximity/similarity queries with metric trees. *Information processing letters*, 40(4):175–179, 1991.

[Wil72] Dennis L Wilson. Asymptotic properties of nearest neighbor rules using edited data. *IEEE Transactions on Systems, Man, and Cybernetics*, (3):408–421, 1972.

[XYC88] Lei Xu, Pingfan Yan, and Tong Chang. Best first strategy for feature selection. In *Pattern Recognition, 1988., 9th International Conference on*, pages 706–708. IEEE, 1988.

[YW09] Ying Yang and Geoffrey I Webb. Discretization for naive-bayes learning: managing discretization bias and variance. *Machine learning*, 74(1):39–74, 2009.

[Zuk10] AV Zukhba. Np-completeness of the problem of prototype selection in the nearest neighbor method. *Pattern Recognition and Image Analysis*, 20(4):484–494, 2010.

Apéndice A

Apéndice A

Apéndice A