Selección de modelo completo en conjuntos con gran volumen de datos

Ángel Díaz Pacheco, Jesús A. González Bernal, Carlos Alberto Reyes García

Reporte Técnico No. CCC-17-006
17 de Noviembre de 2017

Luis Enrique Erro No. 1
Santa María Tonanzintla,
72840, Puebla, México.

Selección de modelo completo en conjuntos con gran volumen de datos

Angel Díaz Pacheco Jesús A. González Bernal Carlos A. Reyes García

Computer Science Department
National Institute of Astrophysics, Optics and Electronics
Luis Enrique Erro # 1, Santa María Tonantzintla, Puebla, 72840, México
E-mail: {diazpacheco,jagonzalez,kargaxxi}@inaoep.mx

Abstract

Big Data es un término que hace referencia a bases de datos de gran magnitud y/o complejidad las cuales no pueden ser procesadas con los equipos y algoritmos de procesamiento de datos tradicionales y cuyo valor potencial reside en el análisis de su información. Dicha información puede encontrarse en forma estructurada, semi estructurada y no estructurada, además es necesario tener en cuenta la incertidumbre asociada a la variedad y veracidad de ésta. Dentro de las áreas de interés de Big Data se encuentran el reconocimiento de patrones y problemas de clasificación, ámbitos que deben ser abordados mediante mejoras y adaptaciones de los métodos existentes en la literatura. Los algoritmos de clasificación son una parte importante en el proceso de análisis de la información, por lo cual se ha buscado mejorar su precisión predictiva a través de extensiones, hibridaciones, desarrollo de nuevos algoritmos y la configuración óptima de sus hiperparámetros, así como la selección de características. Este último enfoque es conocido como selección de modelo y existen trabajos que lo abordan, pero relativos a bases de datos de pequeñas dimensiones, entendiéndose esto, a que es posible ejecutar dichos algoritmos en computadoras de escritorio convencionales. Por otro lado, el problema de la mejora en la precisión predictiva de un algoritmo de clasificación mediante el enfoque de selección de modelo ha sido escasamente explorado para el ámbito de Big Data, problema con restricciones intrínsecas al gran espacio de búsqueda que representan las grandes colecciones de datos de este ámbito, por lo cual el presente trabajo de investigación tiene como objetivo investigar este problema en dicho ámbito, siendo las principales contribuciones de éste al estado del arte: un conjunto de algoritmos para la selección de modelo completo en Big Data.

Palabras clave: Selección de modelo, híperparámetros, MapReduce.

Contenido

Lista de Tablas

Lista de Figuras

1 Introducción

En muchas de las áreas de la actividad humana existe un gran interés en el análisis y almacenamiento de información y dada la naturaleza del área, es generada en grandes cantidades y a gran velocidad. Tal es el caso de los sistemas de reservación de vuelos de las líneas aéreas o la información de colisiones de partículas que se generan en el gran acelerador de hadrones en Suiza. Detectar tendencias de negocios, prevenir enfermedades y combatir el crimen son algunos de los beneficios de trabajar con grandes conjuntos de datos. Debido a las dimensiones, manipular grandes cantidades de información supone dificultades en la captura, almacenamiento, búsqueda y análisis de la misma. Estos problemas han sido frecuentes en ámbitos como la meteorología, astrofísica, la ingeniería genética, simulaciones físicas, la investigación biológica y la investigación aeroespacial, entre otras (Roebuck, 2012). Con el advenimiento de nuevas tecnologías como las redes sociales, la cantidad y la variedad de la información se incrementó a escalas sin precedentes. Se estima que en 2013 Twitter y Facebook generaron diariamente 7 y 10 terabytes de información respectivamente (Tili & Hamdani, 2014) y para 2014 diariamente eran creados 2.5 quintillones de Bytes de información, lo cual representaba que el 90% de la información existente en el planeta había sido creada en los últimos dos años (Wu et al., 2014), a causa de este fenómeno se popularizó el término Big Data. Este término (ya en uso en el volumen 9 de la revista new scientist de 1980, página 89) es utilizado para referirse a cantidades de datos que son demasiado grandes como para ser procesadas por los algoritmos y hardware convencionales (Cox & Ellsworth, 1997). Por otro lado, Marx (2013) lo define como la cantidad de información que excede las capacidades de procesamiento de un sistema en términos de tiempo y consumo de memoria. Una definición muy extendida en la literatura sobre Big Data describe el concepto por medio de 3 rasgos distintivos de la información. Esta definición es conocida como las 3 V's: Volumen, Velocidad y Variedad (del Rio, et al., 2015; López et al., 2015, entre otros). El Volumen hace referencia a la gran cantidad de datos disponibles para el análisis, la Velocidad define la rapidez con la que deben ser procesados los datos, para poder obtener resultados dentro de un tiempo razonable y la Variedad indica que la información puede encontrarse en diversos formatos como texto, imágenes, video y audio, cumpliendo la estructura de la información que requiere una base de datos (tablas con filas y columnas) o no. Posteriormente se han añadido otras V's a la definición: Veracidad y Valor (Tili & Hamdani, 2014). La Veracidad se refiere a la confiabilidad de la información, mientras que el Valor, a la capacidad de la información de ser transformada en un bien valioso por medio del análisis de la misma. Como se detalló en la definición de las 3 V's, una característica distintiva de los conjuntos de datos de este ámbito es su volumen, para poder comprender esta propiedad podemos hacer uso de la clasificación que Hubber propuso en 1996. De acuerdo a ésta, se clasificó el tamaño de los conjuntos de datos en relación a los mostrados en la Tabla 1.1. Bezdek y Hathaway agregaron la categoría Very Large a esta tabla en 2006. En la actualidad, los proveedores de tecnologías de Big Data contemplan otro tipo de escala para definir qué tan grandes son los conjuntos de datos, siendo este parámetro la capacidad requerida para su almacenamiento, por lo general en el rango de un par de Gigabytes hasta cientos de Terabytes[1].

A pesar de la complejidad de analizar conjuntos de datos de gran volumen, existe una gran cantidad de trabajos donde se ha empleado el paradigma de Big Data. Por ejemplo Tannanhill et al. (2013) propusieron el análisis de información meteorológica proveniente de 70 sensores, imágenes de las condiciones de nubosidad e información sobre la posición solar y niveles de energía para realizar la predicción de la energía fotovoltaica de un sistema de generación distribuida, o el trabajo de Mukkamala et al. (2014) que crearon un modelo alterno al análisis de una Red Social para el análisis de sentimientos mediante información obtenida de 11,348 publicaciones del perfil de Facebook de una compañía. Robertson et al. (2010) realizaron un

[1] https://cloud.google.com/bigquery/pricing

Tabla 1.1: Descripción de tamaños de conjuntos de datos según Huber
Fuente: Havens et al., 2012

Bytes	10^6	10^8	10^{10}	10^{12}	$10^{>12}$
Tamaño	Median	Large	Huge	Monster	Very Large

análisis de patrones de participación en 687,626 mensajes distribuidos en los muros de Facebook de Barack Obama, Hillary Clinton y John McCain durante las elecciones de 2008 y Hofer et al. (2004) investigaron la coincidencia de patrones frecuentes aplicados a la búsqueda de fragmentos de compuestos moleculares en un conjunto de datos consistente en 41,316 compuestos. El paradigma de Big Data también incorpora técnicas de aprendizaje automático para realizar el análisis de la información como en el caso de la clasificación supervisada (Tannahill et al., 2013; López et al., 2014, 2015; del Rio et al., 2014, 2015; Wang et al., 2015; Triguero et al., 2015) y agrupamientos o clustering (Havens et al.,2012). Ha sido reportado en varios trabajos que el desempeño de los algoritmos de clasificación depende en gran medida de la correcta selección de sus parámetros, así como de una apropiada selección de características y métodos de pre procesamiento de los datos. Este enfoque es conocido como selección de modelo completo (Escalante et al., 2009) y puede ser entendido como la selección de los valores de configuración de los parámetros (algoritmo de clasificación y sus parámetros, algoritmo de pre procesamiento y selección de características) que mejor describen a un conjunto de datos. Este problema presenta restricciones únicas de tiempo computacional y espacio en memoria a causa del espacio de búsqueda de los conjuntos de datos de este ámbito. Para hacer frente al análisis de conjuntos de datos de gran volumen se han utilizado procedimientos estadísticos basados en muestreo (Kleiner et al., 2014; Liang et al., 2013; Ma et al., 2013) y el enfoque divide y vencerás (Lin & Xi, 2011; Chen & Xie, 2014), además de técnicas que aprovechan las capacidades del procesamiento masivo en clústeres (Dean & Ghemawat, 2008; Sparks et al., 2013).

1.1 Motivación

De acuerdo a las definiciones sobre Big Data que se detallaron en el apartado anterior, una característica de gran importancia es el **Valor** que la información obtiene a través de su análisis. Las técnicas de aprendizaje automático frecuentemente son utilizadas para esta labor y recientemente se han realizado extensiones a los algoritmos para poder procesar cantidades de información cada vez más grandes en un menor tiempo. En Big Data, los algoritmos de aprendizaje se ven beneficiados por un gran incremento en la cantidad de instancias disponibles para el entrenamiento, lo cual permitirá obtener mejores modelos. Sin embargo, a pesar de esto, no es posible decidir de antemano que modelo elegir, ya que, hasta la fecha no existe un modelo cuyo desempeño sea superior al de los demás en todos los conjuntos de datos, situación conocida como el Teorema de no hay tal cosa como almuerzo gratis (Wolpert, 1996) y debido a esto, encontrar el modelo más adecuado a un determinado conjunto requiere de un largo proceso de búsqueda. Este proceso representa la exploración de la combinación de un algoritmo de aprendizaje automático y los valores de sus hiperparámetros que obtengan el menor error de clasificación en dicho conjunto de datos (Thornton et al., 2013). Otros factores que tienen un gran impacto en la mejora de la capacidad de generalización de un algoritmo de clasificación son: la selección de características y el pre procesamiento de los datos. Estos factores en combinación con los antes mencionados conforman el paradigma de la selección de modelo completo. Bajo este paradigma, cada combinación de los factores antes mencionados representa un con-

junto de transformaciones, así como realizar el proceso de aprendizaje del algoritmo de clasificación con una determinada configuración de sus hiperparámetros utilizando los datos del conjunto de entrenamiento (Guo et al. (2008); Escalante et al., 2009; Thornton et al., 2013; Rosales-Pérez et al., 2014a; entre otros). Si la cantidad de datos es mayor a la que un equipo de cómputo puede almacenar, no se podrán realizar las pruebas de las combinaciones en el conjunto de datos y por tanto no se encontrará el modelo que exhibe el mejor desempeño. **La elección del modelo más adecuado a un conjunto de datos de Big Data tiene una repercusión positiva en muchas áreas del quehacer humano donde es necesario almacenar y analizar grandes cantidades de información. En el apartado siguiente se darán un par de ejemplos de esto. La necesidad de mejorar la capacidad de generalización de los algoritmos de clasificación en entornos de grandes cantidades de datos, a través de la selección de modelo completo bajo condiciones de incertidumbre, es la razón que motiva la creación de algoritmos de selección de modelo completo para conjuntos con gran volumen de datos.**

1.2 Justificación

Como se mencionó con anterioridad, las aplicaciones de Big Data contemplan una gran variedad de ámbitos en los cuales la capacidad de almacenar y analizar grandes cantidades de información ha sido de gran valor, por lo cual empresas, gobiernos y la comunidad científica han comenzado a incorporar las tecnologías necesarias para la gestión de información de esta magnitud. En Estados Unidos, los gobiernos estatales han hecho uso de Big Data para incrementar la seguridad pública, descubrir fraudes y la mejora de los servicios de asistencia pública entre otras. En lo referente al descubrimiento de fraudes, Big Data está ayudando a detectar fraudes fiscales que cuestan al gobierno americano millones de dólares cada año. Connecticut, Georgia, Indiana, Luisiana, Nueva York, Carolina del Sur y Washington son algunos de los estados que han contratado a proveedores de servicios de análisis de información en Big Data para detectar esquemas de pago de impuestos y reducir el número de reembolsos fraudulentos que el estado paga cada año. Por ejemplo, Georgia ha capturado $25 millones de dólares en reembolsos fraudulentos desde que comenzaron a utilizar Big Data en 2012 y en Washington se han recuperado más de $10 millones al año (Bourquard & Kirsch, 2014). Por otro lado, en un gran esfuerzo por mejorar la seguridad pública, los legisladores de Carolina del Norte aprobaron el presupuesto para la creación de un sistema para intercambio de información entre las diferentes agencias de seguridad pública. Al centralizar la información en lugar de hacer las consultas en diferentes bases de datos, las agencias de seguridad ahorrarán una cantidad considerable de tiempo y se estima que un ahorro de tan solo 5 minutos por consulta equivale hasta $18.6 millones de dólares anualmente, según lo indica Kay Meyer, director del proyecto en la oficina de Control del Estado (Bourquard & Kirsch, 2014). En cuanto a servicios humanos, el condado de los Ángeles ha encarado fraudes en el programa de cuidado infantil para familias de escasos recursos. Muchos de los fraudes fueron perpetrados por redes de estafadores que reclutaban tanto a padres como proveedores de cuidado infantil para hacer que las investigaciones fueran complicadas y difíciles de continuar. Para combatir el fraude y asegurar los beneficios para familias que en realidad lo necesiten, el Departamento de Servicios Sociales ha utilizado técnicas de minería de datos para identificar y expedir la revisión de casos sospechosos. El proyecto piloto ha alcanzado un 85% de precisión entre 2008 y 2009 en la identificación de redes de estafadores con un ahorro estimado de $6.8 millones de dólares en pérdidas debidas a fraudes. En lo referente a la contribución a la investigación científica de Big Data podemos mencionar la 2^a exploración de objetos estelares del observatorio Palomar. Dicha exploración tomó alrededor de 6 años para ser completada produciendo 3 terabytes (La inspección fue realizada en 1991) de información, conteniendo alrededor de 2 mil millones de objetos estelares (Fayyad et al., 1996). La clasificación automática de los objetos estelares (Galaxia o Estrella)

permitió aumentar la cantidad de información utilizable para el análisis (información ya clasificada) en un 300% y gracias a esto los investigadores pudieron descubrir 16 nuevos Quásares. Un ejemplo más actual es el Gran Colisionador de Hadrones (LHC), cuyo registro de colisiones genera anualmente alrededor de 30 petabytes de información que debe ser analizada para determinar si existe información de interés (CERN, 2013). Por otra parte, el Instituto de Genómica de Beijing (BGI) ubicado la provincia de Shenzen en China es uno de los más grandes productores de información genómica en el mundo generando 6 terabytes de datos por día. Tal cantidad de información es producto del secuenciamiento del genoma de personas, animales, plantas y microbios. Actualmente tienen la capacidad de secuenciar un genoma humano a la semana (de acuerdo al artículo "The Big Challenges of Big Data" el genoma humano tiene un tamaño de alrededor de 140 Gb), tarea que antes solía tomar meses o incluso años (Marx, 2013). Dentro del ámbito empresarial, Amazon es uno de los mayores exponentes en el uso y comercialización de tecnologías de Big Data. Con sus aproximadamente 150 millones de clientes, Amazon ostenta uno de los mejores servicios al cliente debido a la personalización o sugerencias personalizadas de productos. Estas son obtenidas mediante información de ventas pasadas. Por otro lado, también mediante técnicas de Big Data monitorean alrededor de 1.5 billones de elementos de su inventario (Burgess, 2015). Como se puede notar, las colecciones de datos pertenecientes a Big Data abarcan una gran cantidad de disciplinas donde parte del análisis de los mismos es realizado mediante técnicas de aprendizaje automático. Los análisis de estos conjuntos de datos requieren que el usuario realice la selección del algoritmo de aprendizaje y la configuración de sus hiperparámetros. Comúnmente esta elección es realizada en base a la experiencia del analista o la reputación de un determinado algoritmo de clasificación. En cuanto a la elección de los valores de los hiperparámetros, ésta es realizada por lo general mediante prueba y error de diferentes configuraciones (Aydin et. Al, 2011). Han sido propuestos diversos métodos para abordar este problema dentro de los cuales se encuentra la selección de modelo completo. Esta técnica presenta grandes beneficios en comparación a las técnicas ad-hoc a un algoritmo en concreto o para un determinado dominio (Escalante et al., 2009). Dicha ventaja radica en que puede ser aplicada a cualquier problema de clasificación sin modificaciones, pues no requiere del conocimiento previo, tanto del dominio como de aprendizaje automático. Dicha situación beneficiará tanto a usuarios con conocimiento limitado de aprendizaje automático como a los expertos que podrán obtener información de interés sobre el conjunto de datos al analizar los modelos obtenidos. La elección del o los modelos más adecuados para cada conjunto de datos de este ámbito tendrá un gran impacto en el desempeño de los algoritmos utilizados en los procesos de análisis como los mencionados en los párrafos previos, lo cual puede traducirse en beneficios tanto económicos como sociales.

2 Conceptos relacionados

2.1 Selección de Modelo

La selección de modelo se define como "Una estimación del desempeño de diferentes modelos con el fin de elegir al mejor" [101] en el contexto de la descripción de un conjunto de datos. Hay muchas interpretaciones para "Selección de Modelo", estas incluyen la selección y optimización de los hiperparámetros de un algoritmo de clasificación, selección de características, modelado de redes neuronales incluyendo el aprendizaje de los pesos de la red o incluso, optimización de la arquitectura de ésta. Se han hecho extensiones a la definición introduciendo el concepto de Selección de Modelo Completo (Full Model Selection, FMS) [39, 101], el cual está definido como: dado un conjunto de métodos de pre procesamiento, selección de características y algoritmos de aprendizaje, seleccionar la combinación de aquellos métodos que obtengan el error de clasificación más bajo para un conjunto de datos determinado; esta tarea también involucra la selección de los hiperparámetros para el algoritmo de clasificación elegido. Esta labor no es trivial ya que puede realizarse a través de una búsqueda exhaustiva sobre el espacio de búsqueda de los hiperparámetros o a través de un proceso de optimización que explore solo un subconjunto finito de los valores posibles. Hasta ahora, en la mayoría de las ocasiones un desarrollador simplemente selecciona empíricamente el algoritmo de aprendizaje y los hiperparámetros mediante la prueba de un número finito de valores y quedándose solamente con los que ofrecieron el menor error durante la fase de prueba. Otro enfoque para solucionar el problema es mediante la búsqueda en rejilla. Este tipo de búsqueda investiga todas las combinaciones entre los hiperparámetros [56]. Un intento de definición formal de selección de modelo completo acorde a la definición propuesta por Escalante et al. (2009) se proporciona en la ecuación 1.

$$\begin{gathered} a^*_{wA}, p^*, f^* \in argmin\, \mathcal{L}(a_{wA}, p, f, D^{(i)}_{entrenamiento}, D^{(i)}_{validacion}) \\ a^{(i)} \in A, p^{(i)} \in P, f^{(i)} \in F, w_A \in W_A \end{gathered} \tag{1}$$

Donde:

A	= Conjunto de algoritmos de clasificación disponibles
P	= Conjunto de algoritmos de pre procesamiento disponibles incluyendo $\varnothing$
F	= Conjunto de algoritmos de selección de características disponibles incluyendo $\varnothing$
W_A	= Conjunto de híperparámetros de un algoritmo de aprendizaje automático a
D_{entren}, D_{val}	= Particiones disjuntas en el conjunto de datos bajo análisis
$\mathcal{L}$	= función de perdida (tasa de error de clasificación calculado en la partición de validacion)
$argmin$	= valores de los parámetros de entrada para los cuales $\mathcal{L}$ es mínima

2.2 Híperparámetros

Un algoritmo de aprendizaje puede ser visto como una función que toma un conjunto de datos de entrenamiento como entrada y produce una función o modelo como salida. Sin embargo, en la práctica, muchos algoritmos de aprendizaje involucran hiperparámetros, esto es, parámetros que deben ser establecidos manualmente por el usuario. El hiperparámetro de un algoritmo de aprendizaje "A", es una variable que debe ser establecida antes de iniciar el proceso de aprendizaje de "A" en los datos y que no puede ser establecida directamente por el algoritmo de aprendizaje [7]. Los hiperparámetros son aquellos parámetros que definen a un conjunto de funciones de aprendizaje $\{f(x, w_A), w_A) \in A\}$, donde x es un vector de características de entrada, w_A es un vector de hiperparámetros dependiente de los modelos pertenecientes a A el cual es un conjunto de modelos [57].

2.3 Pre procesamiento

La preparación de datos tiene como objetivo dar a la información una presentación adecuada para la construcción de algoritmos de aprendizaje. Casi cualquier conjunto de datos presente en el ámbito científico no es en principio adecuado para ser modelado. La información del mundo real a menudo es incorrecta y sesgada, existen datos faltantes y es en ocaciones ruidosa. Por tanto es necesario realizar un pre procesamiento de los datos para limpiarla y transformarla de tal forma que resulte más adecuada para realizar un proceso de aprendizaje automático. Esta es una de las etapas que consume una mayor cantidad de tiempo, sin embargo, un modelo simple con información limpia tiene la capacidad de superar el desempeño de un modelo de mayor complejidad con información sin pre procesar [28].

Existen muchas formas de realizar el pre procesamiento de los datos: desde escalamientos lineales y no lineales de las entradas y salidas, homogeneización de la variabilidad, análisis de componentes principales (PCA), entre otras [57]. Esta etapa incluye operaciones de limpieza de los datos (tales como eliminación de ruido y datos inconsistentes), integración de datos (donde múltiples fuentes son combinadas en una), transformación de datos (donde la información es transformada y consolidada de forma que sea más apropiada para los procesos de aprendizaje automático) y reducción de información (incluye la selección de características y de instancias) [43].

2.4 Selección de características

La selección de características consiste en escoger un subconjunto de características que idealmente es necesario y suficiente para describir la clase objetivo. Considerando esta definición, las características redundantes e irrelevantes deberían ser removidas [96]. El problema de la selección de características puede ser definido como encontrar una proyección de X a X', donde X es un vector binario de características $X = \{x_1, x_2, ..., x_n\}$ con $x_i \in \{0, 1\}$ para toda $1 \leq i \leq n$ y $X' \subseteq X$ La selección de características es usualmente realizada en consideración al desempeño de alguna función $p(X')$ [37]. Existen dos metodologías que compiten en la selección del más adecuado subconjunto de características para un algoritmo de aprendizaje supervisado: el enfoque de filtros (filter) y el enfoque envoltura (wrapper), [35] los cuales se definen a continuación:

- Los métodos de envoltura o ***wrapper*** hacen uso de algoritmos de aprendizaje automático para evaluar la calidad de los subconjuntos de características. Debido a que el número de subconjuntos de características es exponencial, éstos son usualmente evaluados de forma incremental o decremental por medio de añadir o remover una característica a la vez.
- Los métodos de tipo filtro o ***filter*** trabajan de forma independiente a los algoritmos de aprendizaje automático ya que proveen su propio criterio para juzgar el merito de una característica, tales como la correlación con el valor a ser predicho. La mayoría de los métodos de tipo filtro evalúan cada característica individualmente. El uso de filtros hace posible realizar un ordenamiento de las características de acuerdo a su importancia de forma independiente al algoritmo de aprendizaje utilizado.

2.5 Selección de instancias

La selección de instancias es en cierto sentido una tarea dual con respecto a la selección de características ya que las instancias están ocultas a diferencia de las características. De la misma forma que algunos atributos pueden ser más relevantes que otros, algunos ejemplos o instancias pueden ser de la misma forma más

relevantes que otros para el proceso de aprendizaje. La selección de instancias ha sido estudiada ampliamente en el campo de la detección de valores atípicos. Los métodos de selección de instancias pueden ser clasificados de la misma forma que los métodos de selección de características en filtros y envolturas [88].

2.6 Complejidad del modelo

El concepto de complejidad de un modelo es la mayoría de las veces explicado en relación al sobre ajuste del modelo en lugar de ser definido explícitamente. Las definiciones varían de acuerdo a los autores, como es el caso de [38] indican que la complejidad de un modelo se incremente en relación al número de características del conjunto de datos (la dimensionalidad de los vectores de entrada) y que por tanto es deseable enfocarse en las características que mejor describen o caracterizan las observaciones. Por otro lado, Godo (2005) define la complejidad del modelo como el número de parámetros requeridos para especificar por completo un modelo. En [57] se describe de forma intuitiva que un modelo sobre ajustado es más complejo y esto se debe en parte a su número de parámetros libres. Dentro de este orden de ideas puede entenderse que entre mayor sea la complejidad del modelo, éste es más adecuado a lo que puede considerarse como ruido en el conjunto de entrenamiento en lugar de lo que puede ser considerado como señal en el conjunto de prueba [80]. El núcleo del problema es determinar un nivel apropiado de complejidad para maximizar la señal y minimizar el ruido. Esta situación, es conocida comúnmente como el acuerdo entre sesgo y varianza, ya que un modelo complejo exhibe una gran varianza, mientras que uno muy simple está fuertemente sesgado [23]. En la Figura 2.1 se muestra como el error disminuye en el conjunto de entrenamiento (línea azul), mientras que la complejidad del modelo se incrementa. La figura muestra también como decrece el error en el conjunto de prueba (línea roja) hasta que la complejidad del modelo se encuentra en el punto marcado por la línea punteada. Posterior a ese punto, el error en el conjunto de prueba aumentará y puede considerarse que el modelo esta sobre ajustado. En este trabajo se medirá y penalizará la complejidad en relación a la contribución de los valores de los híper parámetros sobre la misma.

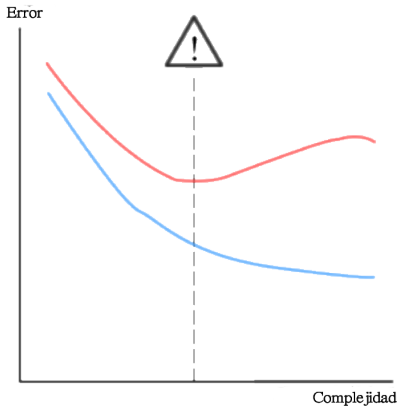

Fig. 2.1: Error en el conjunto de entrenamiento (línea azul) vs error en el conjunto de prueba (línea roja).

2.7 Dimensión Vapnik-Chervonenkis

Una forma de enfrentar el problema del aumento de la complejidad en un modelo, es mediante una métrica que nos permita conocer el nivel de complejidad del mismo. Una alternativa es la dimensión Vapnik-Chervonenkis (VC). La dimensión VC mide la complejidad del espacio de hipótesis por medio del número de instancias que pueden ser discriminadas utilizando una función f. Considerando dicha función, con un vector de parámetros θ, se dice que la función fracciona o divide al conjunto de datos $D = \{\vec{x}^{(1)}, ..., \vec{x}^{(n)}, y^{(n)}\}$ si, para todas las asignaciones de etiquetas a los datos, existe un vector θ tal que la función f s el número máximo de puntos de datos que son separados por f. En otras palabras, la dimensión VC de una función f es h', donde h' es el número máximo de h para algún conjunto de datos de cardinalidad h y que puede ser separado por f [110].

2.8 Regularización

Otra alternativa para controlar la complejidad de un modelo es por medio de técnicas para penalizarla, como la regularización. Generalmente un modelo dado es regularizado por medio de la reducción o aumento del efecto de algunas variables independientes del modelo. La regularización no es exclusiva de la regresión y la mayoría de los algoritmos de aprendizaje utilizan alguna forma de regularización con el fin de crear modelos más acertados a partir del conjunto de entrenamiento [113]. La regularización es una herramienta matemática para imponer información a priori en la estructura de la solución obtenida como consecuencia de un proceso de optimización. En la ecuación 2 se observa la regularización representada por una función de penalización añadida a la función de pérdida. Dos de las funciones de penalización más comunes son conocidas como norma $L1$ y $L2$. La norma L2 consiste en la suma del cuadrado de los parámetros, mientras que en L1 es solo la suma absoluta de estos parámetros [78].

$$argmin\, L(y_i, f(x_i)) + wR(f) \tag{2}$$

Donde:

$L(y_i, f(x_i))$ = función de pérdida
$R(f)$ = regularizador
w = parámetros del modelo

2.9 Terminación temprana

La terminación temprana es una forma de regularización. En esta técnica el conjunto de entrenamiento es dividido en un conjunto de entrenamiento más pequeño y un conjunto de validación. El entrenamiento es realizado en el nuevo conjunto de entrenamiento y el desempeño es evaluado en el conjunto de validación. Posteriormente, el entrenamiento es detenido tan pronto como el error del conjunto de validación aumenta y por tanto se utilizarán los valores de los parámetros obtenidos antes del aumento del error de validación [70]. De manera más formal, la terminación temprana puede entenderse como: dado $k = k^*$ donde k es el número de iteraciones tras las cuales un algoritmo logra converger, si se desea encontrar un cierto $\tilde{k} < k^*$ para el cual el error de generalización EG es mínimo, puede estimarse un EG_k del EG en un conjunto independiente en cada iteración k, para $k = 1, ..., k^*$ durante el proceso de aprendizaje. Entonces es dado por la ecuación 3 [62].

$$\tilde{k} = argmin_{k=1,...,k^*}(EG_k) \tag{3}$$

2.10 Meta-aprendizaje

El meta-aprendizaje es el proceso de "aprender a aprender", de forma informal un algoritmo de aprendizaje utiliza la experiencia para cambiar ciertos aspectos de un algoritmo de aprendizaje o el método de aprendizaje en si mismo, tal que el mecanismo de aprendizaje modificado es mejor que el original al aprender a partir de experiencia adicional [92]. Desde su introducción, el término meta-aprendizaje ha sido utilizado por varios investigadores en formas diferentes, algunas veces en definiciones mutuamente inconsistentes. Por esta razón se introducirá la siguiente terminología. Considerando el dominio D de posibles experiencias $s \in D$, cada una con una probabilidad $p(s)$ asociada. Sea T el conjunto de entrenamiento de experiencia disponible en cualquier momento. El conjunto de entrenamiento de experiencia es un subconjunto de D, por ejemplo $T \in D_T \subset P(D)$ donde $P(D)$ es el conjunto poder de D. Un agente π_θ es parametrizado por $\theta \in \Theta$. Un proceso asociado a una medida de desempeño $\phi : (\Theta, D) \to \mathcal{R}$ con el comportamiento del agente para cada experiencia. El desempeño esperado de un agente D es denotado por Φ tal que:

$$\Phi(\Theta) = \mathbb{E}_{s \in D}[\phi(\Theta, s)] \tag{4}$$

Ahora definiendo a un algoritmo de aprendizaje L_μ: $(\Theta, D_T) \to \Theta$, parametrizado por $\mu \in M$, como una función que cambia a los parámetros Θ del agente basados en el conjunto de entrenamiento de experiencia, de tal forma que su desempeño esperado Φ es incrementado. Más formalmente, se definirá la ganancia del desempeño esperado del algoritmo de aprendizaje δ como:

$$\delta(L_\mu) = (\mathbb{E})_{\theta \in \Theta, T \in D_T}[\Phi(L_\mu(\theta, T)) - \Phi(\theta)] \tag{5}$$

Cualquier algoritmo de aprendizaje debe satisfacer $\delta > 0$ en su dominio, esto es que debe mejorar el desempeño esperado. Los componentes modificables μ de un algoritmo de aprendizaje son denominados sus meta-parámetros. Un algoritmo de aprendizaje (ML) es definido como $(M, D_T) \to M$, donde la función que cambia los meta-parámetros de un algoritmo de aprendizaje, basado en el entrenamiento en el conjunto de datos de experiencia, de tal forma que su ganancia de desempeño δ se incrementa:

$$\mathbb{E}_{\mu \in M, e \in D_T}[\delta(L_{ML(\mu,T)}) - \delta(L_\mu)] > 0 \tag{6}$$

En otras palabras, utilizando $L_{\mu'}$ tiende a guiar a un mayor incremento en el desempeño utilizando L_μ, donde $\mu' = ML(\mu, T)$ son los meta-parámetros actualizados.

2.11 Modelo proxy

El modelado de proxys (palabra en latín que significa substituto) es uno de los métodos más ampliamente utilizados para replicar la funcionalidad de simulaciones numéricas complejas para asistir en procesos como cuantificación de incertidumbres, optimización de diseño operacional y correspondencia histórica. Los modelos proxy han sido utilizados con mayor frecuencia en la industria petrolera y de gas natural para recrear modelos de orden reducido y superficies de respuesta los cuales reducen el tiempo de ejecución mediante la aproximación del problema y/o el espacio de solución. Las superficies de respuesta, las cuales son esencialmente modelos proxy basados en técnicas estadísticas requieren de cientos de ejecuciones de la simulación compleja para ajustarse a las salidas de esta para utilizar el modelo en análisis de incertidumbre y propósitos de optimización [73].

2.12 Algoritmos de búsqueda basados en poblaciones

En los algoritmos de búsqueda basados en poblaciones, la noción de una única solución es remplazada por el mantenimiento de una colección o población de soluciones [75]. Los miembros de esta población son en principio seleccionados para ser las soluciones candidatas actuales y posteriormente cambios son realizados en estas soluciones para producir nuevas soluciones candidatas. Desde que existe una colección de soluciones en lugar de solo una, podemos sacar provecho de esta situación para a partir de dos o más de estas soluciones para crear nuevas soluciones candidatas. El uso de poblaciones, ademas de oportunidades trae consigo algunos inconvenientes como la necesidad de una estrategia para seleccionar las soluciones que se considerarán soluciones candidatas actuales. Por otra parte, cuando una o más soluciones candidatas han sido creadas es necesaria una estrategia para mantenerlas en la población. Esto es, asumiendo que deseamos mantener el tamaño de la población constante, algunas soluciones deben de ser descartadas con el fin de hacer lugar para las nuevas soluciones creadas. Mientras que la estrategia de selección engloba los detalles de como elegir candidatos de la población en si misma, la estrategia de mantenimiento de las soluciones en la población afecta que soluciones son elegibles. En la Fig 2.2 se presenta un algoritmo de búsqueda basado en poblaciones.

Pasos involucrados:
1. Inicio: Incializar y evaluar la solución S(0).
2. Seleccionar: Seleccionar S(t) de S(t-1)
3. Generar: Genera un nuevo espacio de soluciones S'(t) a través de operadores
4. Evaluar: Evaluá la solución S'(t) y reemplaza S(t) con S'(t)
5. Continua: Regresar a 2 (A menos que se cumpla el criterio de terminación)

Fig. 2.2: Algoritmo de búsqueda basado en poblaciones.

Existen casi tantos algoritmos de búsqueda basados en poblaciones como formas de manejar los problemas antes mencionados (creación, reemplazo y mantenimiento de soluciones). Los algoritmos conocidos como evolutivos, son ejemplos de este algoritmos de búsqueda basados en poblaciones pues comparten las características mencionadas anteriormente.

2.12.1 Algoritmo genético

Un algoritmo genético (AG) es una técnica de búsqueda iterativa inspirada en los principios de la selección natural. Los AG no buscan modelar la evolución biológica, sino derivar estrategias de optimización. El concepto se basa en la generación de poblaciones de individuos mediante la reproducción de los padres. Durante el curso de la evolución, los genes con evolución lenta fueron remplazados por genes con mejor estrategia evolutiva (Cruz, 2011). En rigor, los AG se pueden concebir como métodos de optimización. En general, los problemas de optimización se plantean de la siguiente manera:

$$x_0 \in X \wedge f\, es\, máximo\, en\, x_0\, donde\, f : X \to R, \therefore\ f(x_0) = máx_{x \in X} f(x) \tag{7}$$

La transición de una generación a otra en un AG consta de cuatro elementos básicos:

- Selección: mecanismo de selección individual para la reproducción acorde con la función de aptitud (valor de la función objetivo). Los algoritmos de selección serán los encargados de escoger que individuos van a disponer de oportunidades de reproducirse y cuáles no.
- Cruzamiento: método de fusión sobre la información genética de dos individuos. Provee un mecanismo para heredar características a su descendencia; interviene en ambos progenitores.
- Mutación: produce cambios incrementales al azar en la descendencia, efectuando cambios aleatorios en los valores del alelo en algunos genes.
- Reemplazo: procedimiento para calcular una nueva generación de la anterior y sus descendientes.

2.12.2 Optimización de enjambre de partículas

La optimización de enjambre de partículas es un algoritmo de optimización que simula el comportamiento de las bandadas de aves en vuelo en busca de alimento [52]. Este comportamiento es caracterizado por la colaboración colectiva del grupo de aves para lograr el objetivo común de conseguir alimento. En el algoritmo de optimización de enjambre de partículas, cada uno de los individuos es denominado partícula y cada partícula representa una solución potencial. Cada partícula actualiza su posición y velocidad de acuerdo a su mejor desempeño obtenido (pBest) y al mejor desempeño global obtenido (gBest). Sea N el tamaño del enjambre, D el espacio de búsqueda, x_{ij}^t es la j-ésima dimensión de la partícula i en la iteración t; v_{ij}^t es la j-ésim dimensión de la velocidad de la partícula i en la iteración t; $pBest_i^t$ es la j-ésima dimensión de la mejor marca personal de la partícula i en la iteración t; $gBest^t$ es la mejor marca global de toda la población en la iteración t. La nueva velocidad y posición de cada partícula x_{ij}^t son actualizadas por (8) y (9).

$$v_{ij}^{t+1} = wv_{ij}^t + c_1 r_1 (pBest_{ij}^t - x_{ij}^t) + c_2 r_2 (gBest_{1j}^t - x_{ij}^t) \tag{8}$$

$$x_{ij}^{t+1} = x_{ij}^t + v_{ij}^{t+1} \tag{9}$$

Donde C_1 y C_2 son constantes, los cuales son denominados coeficientes de aceleración, r_1 y r_2 son números aleatorios de una distribución uniforme en el rango $[0, 1]$, w es la ponderación de la inercia la cual decrece mediante decremento lineal.

2.13 Lógica difusa

La lógica difusa (fuzzy logic) es considerada una generalización de la teoría general de conjuntos que permite que elementos de un universo tengan grados intermedios de pertenencia a un conjunto por medio de una función característica. Con esta idea modifica el concepto de bivalencia de la lógica booleana, el cual pasa a ser un caso particular de los conjuntos difusos. Acorde a esto, un conjunto difuso está compuesto por dos partes esenciales: sus elementos y la función de pertenencia que asigna el grado de pertenencia de estos elementos a su conjunto. En otras palabras, un conjunto difuso A se define como:

$$A = (x, \mu_A(x) | x \in U) \tag{10}$$

Donde:

A = Conjunto difuso
x = Elemento a probar su grado de pertenencia al conjunto A
$\mu_A(x)$ = Grado de pertenencia del elemento x al conjunto A

La función $\mu_A(x)$ asigna un valor real en el intervalo [0,1] que representa el grado de pertenencia del elemento x al conjunto A. En este sentido, mientras más cerca está el valor de $\mu_A(x)$ al valor unitario, mayor es el grado de pertenencia de x en A. Así, por ejemplo, si x es el conjunto de los números reales y A es el conjunto de los números mayores que 1 se puede dar una caracterización precisa, aunque subjetiva, de A si se especifica $\mu_A(x)$. La forma de la función de pertenencia está abierta a cualquier función de x que modele la pertenencia de los elementos al conjunto que se pretende representar [79].

2.13.1 Reglas difusas

Las reglas difusas controlan el comportamiento de cualquier sistema difuso. Estas son formas de representar estrategias apropiadas cuando el conocimiento proviene de la experiencia. Las reglas utilizan variables lingüísticas como vocabulario. Una base de reglas difusas es un conjunto de reglas "Si – Entonces" que pueden ser expresadas de la siguiente forma:

$$Si \quad x_1 \quad es \quad A_1^k \quad Entonces \quad y_k \quad es \quad C^k \tag{11}$$

Donde:

$i = 1, ..., n; con$ n = número de antecedentes
x_i = Valor de entrada del i-ésimo antecedente de la regla
A_i^k = k-ésimo conjunto difuso del i-ésimo antecedente
y_k = Valor de salida del consecuente
C^k = k-ésimo conjunto difuso del consecuente

Se observa que esta regla tiene además la particularidad de que es una regla multi-antecedente; este tipo de reglas (que combina varias variables en el antecedente) es el más utilizado en el diseño de sistemas difusos. Existen dos caminos para obtener el conjunto de reglas correspondiente a un conjunto de datos numéricos [72]:

1. Dejar que los datos establezcan los conjuntos difusos que aparecen en los antecedentes y consecuentes.
2. Predefinir los conjuntos difusos para antecedentes y consecuentes y luego asociar los datos a esos conjuntos.

2.14 Dimensión intrínseca

En el ámbito del reconocimiento de patrones, la información es representada como vectores de dimensión d. La información es entonces embebida en R^d pero esto no necesariamente implica que su dimensión real sea d. La dimensión de un conjunto de datos es el número mínimo de variables necesarias para representar dicho conjunto sin perdidas de información. Esto puede entenderse como, dado un conjunto de datos $D \subset R^d$, su dimensión intrínseca (ID) es igual a M si sus elementos se encuentran en su totalidad en un sub espacio $M - dimensional$ de (donde $M < d$) [14].

2.15 MapReduce

A medida que la necesidad de procesar grandes colecciones de datos se hace más prevalente en la comunidad científica y de negocios en general, se ha incrementado la demanda por sistemas y modelos de programación que puedan ejecutarse en hardware económico y no tan especializado. Para tomar ventaja de los servidores con múltiples CPU's y múltiples núcleos, los modelos de programación en paralelo pueden utilizarse para reducir el tiempo que toma el cómputo de grandes cargas de datos. En respuesta a esta necesidad surge MapReduce, un modelo de programación y una implementación asociada para el procesamiento de grandes conjuntos de datos. MapReduce posibilita la paralelización y distribución del cómputo a gran escala [29]. Este modelo está basado en la estrategia divide y vencerás, ya que los conjuntos de entrada son divididos y cada sección es procesada en paralelo. Posteriormente las salidas de los nodos son combinadas [68]. Una de las ventajas más significativas de MapReduce es que provee una abstracción que oculta muchos detalles a nivel de sistema al desarrollador, por esta razón, el programador puede enfocarse en que cálculos requieren ser realizados, en contraposición a cómo esos cálculos son llevados a cabo o cómo los datos son enviados al proceso del que dependen. De la misma forma que OpenMP y MPI, MapReduce provee un medio para distribuir los cálculos sin sobrecargar al programador con detalles sobre el cómputo distribuido (pero a un diferente nivel de granularidad). En lugar de mover grandes cantidades de información, es más eficiente, si es posible, mover el código a donde se encuentran los datos. Esto es realizado por medio de la difusión de los datos a través de los discos locales de los nodos en el clúster y ejecutar los procesos en los nodos que almacenan la información. Una de las ideas claves detrás de MapReduce es mover el código y no los datos. Sin embargo, con el fin de que el cómputo ocurra, es necesario proporcionar datos al programa. En MapReduce, este problema es manejado por el sistema de archivos subyacente. Para lograr una buena ubicación de los datos, el planificador de MapReduce inicia el proceso de cómputo en el nodo que almacena un bloque particular de información necesaria para la tarea. Esto tiene como efecto el movimiento del código a la ubicación de los datos. Si esto no es posible (por ejemplo, si un nodo está ejecutando muchos procesos), nuevos procesos serán iniciados en cualquier otro nodo y la información necesaria será movida a través de la red. MapReduce debe cumplir con todos los procesos en un ambiente donde los errores y fallas son la norma y no la excepción. Desde que MapReduce fue explícitamente diseñado para funcionar en equipo de bajo costo, el entorno de ejecución debe ser especialmente resistente [64]. El diseño del modelo MapReduce fue desarrollado considerando los siguientes principios fundamentales [89]:

- Hardware básico y de bajo costo. En lugar de utilizar equipos costosos y de alto desempeño como computadoras de procesamiento masivo en paralelo (MPP) equipadas con subsistemas de almacenamiento y una estructura de red de gama alta, MapReduce fue diseñado para ejecutarse en clústeres de equipos de cómputo básicos.

- Clústeres extremadamente escalables RAIN (Arreglo redundante de nodos independientes y baratos). En vez de utilizar sistemas de almacenamiento redundante (RAID) basados en redes de área de almacenamiento (RAID-SAN) o almacenamiento conectado en red (RAID-NAS), un nodo MapReduce utiliza su disco duro local. Estos nodos pueden interconectarse mediante conexiones de red estándar y pueden ser puestos fuera de servicio casi sin afectar a algún proceso MapReduce en ejecución.

- Tolerantes a fallos. MapReduce aplica mecanismos sencillos para replicar la información con el fin de mantener los procesos en ejecución en caso de fallo. Para manejar nodos colapsados, el administrador del sistema simplemente los pone fuera de servicio. Nuevos nodos serán puestos en ejecución sin grandes complicaciones.

- Altamente paralelo. La contribución más importante del modelo MapReduce es su habilidad para automáticamente soportar la paralelización de tareas. Por tanto, permite a los desarrolladores enfocarse principalmente en el problema a resolver más que los detalles de bajo nivel de la implementación como son: administración de memoria, localización de los archivos, programación paralela o multi hilo en red. Además, la arquitectura no compartida le hace mucho más escalable y listo para la paralelización.

Altamente paralelo. La contribución más importante del modelo MapReduce es su habilidad para automáticamente soportar la paralelización de tareas. Por tanto, permite a los desarrolladores enfocarse principalmente en el problema a resolver más que los detalles de bajo nivel de la implementación como son: administración de memoria, localización de los archivos, programación paralela o multi hilo en red. Además, la arquitectura no compartida le hace mucho más escalable y listo para la paralelización. Comparado a los sistemas de bases de datos en paralelo, MapReduce tiene algunas ventajas [20]:

1. MapReduce se encarga solo del procesamiento de datos y por tanto es independiente del sistema de archivos del sistema operativo subyacente. Esta independencia permite que el almacenamiento pueda ser escalado independientemente de dicho sistema operativo.

2. Los procesos Map y los Procesos Reduce son asignados a los nodos disponibles de acuerdo a la demanda y el número de tareas. Esto es independiente del número de nodos en el clúster. Este esquema de calendarización permite que los usuarios puedan incrementar o decrecer dinámicamente el tamaño del clúster por medio de la asignación de nodos disponibles en la Nube. El sistema administrará y calendarizará los nodos disponibles para realizar tareas sin interrumpir trabajos en ejecución.

3. Las tareas Map y Reduce son ejecutadas independientemente una de otra. Sin embargo, existe comunicación entre procesos Map y procesos Reduce. Este diseño permite que MapReduce sea altamente resistente a fallos en los nodos. Cuando un nodo falla durante la ejecución de una tarea, solo los procesos Map y Reduce de ese nodo deberán ser reiniciadas, sin necesidad de reiniciar todo el trabajo.

2.16 Funcionamiento de MapReduce

En el modelo MapReduce un proceso de cómputo es especificado como una secuencia de etapas ***map, shuffle*** y ***reduce*** que operan en un conjunto de datos $X = \{x_1, x_2, ..., x_n\}$. La etapa ***map*** aplica una función μ a cada valor x_i para producir un conjunto finito de pares (k, v) clave-valor. Para permitir el cómputo paralelo de una función $\mu(x_i)$, dicha función solo debe depender de x_i. La etapa ***shuffle*** recolecta todos los pares clave-valor producidos en la etapa ***map*** ejecutada anteriormente y produce un conjunto de listas $L_k = (k; v_1, v_2, ..., v_n)$, donde cada lista consiste de todos los valores v_i, de tal manera que $k_i = k$ para una determinada clave k asignada en la etapa ***map***. La etapa ***reduce*** aplica una función ρ, a cada lista $L_k = (k; v_1, v_2, ..., v_n)$ formada en la etapa anterior (***shuffle***) para producir un conjunto de valores $y_1, y_2, ..., y_n$. La función de reducción ρ es definida secuencialmente sobre L_k pero siendo independiente de otras listas L'_k donde $k' \neq k$ [45]. En la Figura 2.3 se muestra una representación de las etapas ***map, shuffle*** y ***reduce*** sobre un conjunto de datos representado mediante colores para mayor facilidad de interpretación.

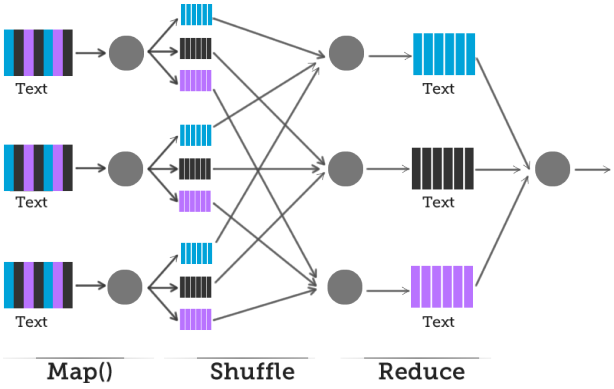

Fig. 2.3: Modelo de programación MapReduce y las etapas que lo conforman: Map, Shuffle y Reduce.

2.16.1 Apache Spark

Apache Spark es un marco de desarrollo de aplicaciones de propósito general para la creación de aplicaciones distribuidas con mecanismos para el procesamiento de datos mayoritariamente en memoria principal con la capacidad para realizar operaciones de análisis, aprendizaje automático y procesamiento de grafos en conjuntos con gran volumen de datos y en flujos de datos. En contraste con otros marcos de desarrollo como Hadoop (donde el procesamiento se realiza con múltiples operaciones de lectura/escritura a disco), la mayoría de las etapas de procesamiento son realizadas en memoria y por tanto, en la mayoría de ocasiones obtiene un mejor desempeño para cierto tipo de aplicaciones como algoritmos iterativos o minería de datos interactiva. La piedra angular de Spark es el conjunto de datos elástico distribuido (traducción de Resilient Distributed Dataset) también denominado RDD. Este es una colección de registros extendidos por una o más particiones. El uso de los RDD permite ocultar las particiones y la distribución de los datos mediante interfaces de programación de alto nivel, permitiendo diseñar aplicaciones paralelas. Las características principales del RDD son:

- **Elástico**, ya que es tolerante a fallos pues sus mecanismos le permiten re computar particiones pérdidas o dañadas a causa de fallos en los nodos de procesamiento.
- **Distribuido**, pues permite ubicar la información en distintos nodos de procesamiento.
- **Conjunto de datos**, ya que es una colección de instancias o tuplas que representan los registros de la información bajo análisis.

Los RDD soportan principalmente dos tipos de operaciones: transformaciones, las cuales obtienen un nuevo RDD y las acciones. Estas ultimas realizan cálculos sobre el conjunto de datos y obtienen valores de salida [61].

2.16.2 Otras plataformas

Las redes entre pares involucran miles de equipos conectados a la red. Es una arquitectura de red descentralizada y distribuida donde los nodos en la red (conocidos como pares) realizan tareas al igual que consumen recursos. Típicamente, MPI (Interface de Paso de Mensajes) es el esquema de comunicaciones utilizado en tal configuración para comunicar e intercambiar la información entre pares. El mayor problema de este

paradigma surge durante la comunicación entre nodos. La difusión general de mensajes entre nodos es simple pero la agregación de datos/resultados es más complicada. MPI es realmente eficiente para el desarrollo de algoritmos para el análisis de Big Data, pero su mayor desventaja es que carece de mecanismos para manejar fallos, la falla de un solo nodo puede causar el colapso de todo el sistema. Por otro lado, los Clústeres de alto desempeño computacional (HPC) son equipos con miles de núcleos de procesamiento. Estos pueden tener una gran variedad de organizaciones de almacenamiento, cache, mecanismos de comunicación, etc. Estos sistemas utilizan hardware de alto desempeño optimizado para obtener mayor velocidad y bajas latencias. A causa de su hardware de gran calidad, la tolerancia a fallos no es un gran problema desde que los fallos en el hardware son extremadamente raros. El costo inicial de estos sistemas puede ser muy alto a causa del uso de hardware de alto desempeño. Esta configuración de clústeres no es tan escalable, pero sigue teniendo la capacidad de procesar terabytes de información, aunque el costo del escalamiento de estos sistemas es bastante alto. Las Unidades de Procesamiento de Gráficos (GPU) son hardware especializado diseñado para acelerar la creación de imágenes en un buffer compartido para la salida de señal por pantalla. A causa de su arquitectura paralela masiva y recientes desarrollos en su hardware y los marcos de desarrollo relacionados, han dado origen a las GPGPU (GPU de propósito general). Las GPU tienen un mayor número de núcleos de procesamiento en comparación a los CPU multi núcleo. Sin embargo y a pesar de que han sido utilizadas para desarrollar algoritmos de aprendizaje automático de alto desempeño, su mayor inconveniente es la memoria limitada que contienen, un máximo de 12GB de memoria por GPU no es adecuado para el proceso de información de la escala de terabytes. Una vez que el tamaño de la información es mayor que el tamaño de la memoria de la GPU, el desempeño decrece significativamente mientras que el acceso a disco se convierte en el mayor cuello de botella [97].

3 Trabajos relacionados

En esta sección se muestra un análisis de algunos de los trabajos reportados en la literatura con mayor relevancia para el desarrollo de este proyecto. Existen en la literatura algunos trabajos que abordan el problema de la selección de modelo. Debido a la falta de estandarización en este término, parte de los trabajos solo abordan la búsqueda de la configuración óptima de los hiperparámetros de un único algoritmo de clasificación, mientras que otros realizan la selección del modelo completo (acorde con la definición establecida en la sección 2.1). Un ejemplo de los trabajos que abordan solo la configuración de los hiperparámetros de un algoritmo de clasificación es el trabajo de Guo et al. (2008). En este trabajo, se realiza la optimización de los hiperparámetros del algoritmo máquina de soporte vectorial de mínimos cuadrados o LS-SVM. El proceso de búsqueda es realizado a través de la optimización de enjambre de partículas (PSO) y la función de evaluación es realizada mediante el promedio de los porcentajes de clasificación de la validación cruzada a 5 pliegos. El algoritmo propuesto no toma en cuenta el tipo de kernel como híper parámetro a ser optimizado por lo cual, para encontrar el modelo más adecuado para un conjunto de datos, el proceso debe ser repetido tantas veces como kernels se contemplen, además, no se toman en cuenta aspectos como el control de la complejidad de los modelos ni extensiones para realizar dicho proceso en conjuntos de datos de gran volumen. En Escalante et al. (2009), se define y se aborda el problema de la selección de modelo completo. El proceso de búsqueda de la configuración óptima de hiperparámetros, selección de características y métodos de pre procesamiento es realizado mediante PSO. Tanto en el trabajo de Guo et al. como en el de Escalante et al. se opta por la utilización de PSO en lugar de métodos evolutivos ya que solo se requiere un operador para realizar la actualización de las soluciones y por la simplicidad de la representación de éstas. Las partículas o soluciones potenciales son evaluadas por medio de la tasa de error balanceado obtenida en la validación cruzada a 2 pliegos, lo cual reduce en gran medida el tiempo de ejecución. Los autores argumentan que los mecanismos proporcionados por la búsqueda empleada ayudan a disminuir los efectos del sobreajuste y el aumento de la complejidad de los modelos empleados, pero no se proporcionan formas de medir la complejidad de los modelos. Para poder hacer frente a conjuntos de datos de gran volumen, se propone el uso de técnicas de muestreo, sin embargo, no se establece cual es el límite en el tamaño de los conjuntos de datos para los cuales funcionará este método. Chatelain et al. (2010) presentaron un método optimización de hiperparámetros del algoritmo SVM multi objetivo. La función de desempeño toma en cuenta tanto la sensibilidad como la especificidad del modelo. Para definir el conjunto de las mejores soluciones obtenidas se utilizó el principio de optimalidad de Pareto. Dicho principio establece que dado un vector de decisión , este domina a otro vector de decisión si no es peor que para alguna función objetivo y es mejor que por al menos en una función objetivo. Tampoco se establecen mecanismos para evaluar la complejidad de los modelos ni extensiones para conjuntos de datos de gran volumen. Por otro lado, este trabajo se diferencia de los métodos analizados de selección de modelo multi objetivo en su capacidad para ordenar las soluciones de acuerdo a su aptitud en ambos objetivos en lugar de solo seleccionar el conjunto de soluciones que pertenecen al frente de Pareto. Aydin et al. (2011) emplean un sistema inmune artificial (AIS) para la selección de los parámetros óptimos del algoritmo de clasificación SVM. La función de evaluación del desempeño de las soluciones es multi objetivo, ya que por un lado busca disminuir el error de clasificación en los modelos producidos y también busca obtener modelos con el menor número de vectores de soporte, por lo cual la búsqueda producirá modelos de baja complejidad. El valor de desempeño final es calculado mediante una suma ponderada de los criterios antes mencionados. Aunque este trabajo toma en cuenta la complejidad de los modelos, podría emplear la dimensión VC que fue creada originalmente para el algoritmo SVM. En Escalante et al. (2012) el método de selección de modelo completo propuesto en 2009 es utilizado para obtener un conjunto de modelos con los cuales crear diferentes ensambles para la

clasificación de los tipos y subtipos de leucemia. El trabajo propone diversas alternativas para aprovechar las soluciones propuestas por métodos de búsqueda que operan sobre poblaciones. Bergstra & Bengio (2012) compararon el uso de la búsqueda aleatoria y la búsqueda en rejilla para la optimización de hiperparámetros de algoritmos de clasificación. Obtuvieron evidencia de que con la búsqueda aleatoria podían encontrarse modelos con igual o mejor capacidad de generalización y en una fracción del tiempo de cómputo requerido por la búsqueda en rejilla. El trabajo no aborda aspectos como el control de la complejidad y métricas para la evaluación de los modelos evaluados durante la búsqueda. Sin embargo, esta técnica es útil para encontrar modelos de buen poder predictivo en presupuestos de tiempo cortos, por lo cual puede ser adecuado para la selección de modelo en conjuntos de datos de gran volumen. Thornton et al. (2013) desarrollaron una herramienta que enfrenta dos aspectos de la selección de modelo completo: la selección de características y la selección de un algoritmo de clasificación con la optimización de sus parámetros. La herramienta puede ser entendida como una optimización jerárquica de los parámetros de un algoritmo de clasificación, siendo un parámetro raíz y de la cual se desprenden los demás, el algoritmo de clasificación a ser evaluado. El método de búsqueda de los parámetros es aleatorio pero guiado por la optimización secuencial basada en modelo (SMBO). Éste es un método de optimización de enfoque Bayesiano que identifica puntos en el espacio de búsqueda que merecen ser investigados. Para evitar modelos sobre ajustados utilizaron una partición en el conjunto de entrenamiento para validación y debido a que la herramienta desarrollada funciona sobre WEKA, es posible que dicha herramienta pueda ser extendida a conjuntos de datos de Big Data debido a la incorporación de librerías MapReduce en las nuevas versiones de WEKA, aunque este detalle no fue mencionado por los autores. Bergstra et al. (2013) desarrollaron un conjunto de librerías para realizar selección de modelo (HyperOpt). Dichas librerías permiten elegir el método de búsqueda (rejilla, optimización bayesiana y búsqueda aleatoria entre otras), escoger o diseñar la función de evaluación y definir el espacio de búsqueda de los hiperparámetros. Este marco de desarrollo también provee los mecanismos necesarios para realizar el proceso de forma paralela, pero no provee las herramientas para realizar selección de modelo para conjuntos de datos de gran escala. Chan et al. (2013), realizaron optimización de hiperparámetros de dos algoritmos de filtrado colaborativo utilizados en sistemas de recomendación de gran escala. La contribución principal de este trabajo se centra en la investigación de los efectos de realizar no solo reentrenamiento en presencia de nuevos datos, sino también realizar selección de modelo. Los resultados reportados muestran un incremento en la precisión de los sistemas de recomendación al cambiar el paradigma de solo reentrenar en presencia de nuevos datos. Los experimentos fueron realizados utilizando MapReduce para hacer frente a las dimensiones de los conjuntos de datos. En este trabajo no se mencionan aspectos como el control de la complejidad y método de selección de la solución final. Oh et al. (2014) proponen un algoritmo para modelado de sistemas difusos complejos con la capacidad de evadir el problema de la explosión combinatoria. Dicho algoritmo optimiza sus parámetros a través de un algoritmo genético paralelo. A pesar de que el algoritmo puede operar en paralelo, solo se aprovecha esta capacidad para explorar con mayor rapidez el espacio de búsqueda, pero no se puede utilizar para la búsqueda de modelos en conjuntos de datos de gran volumen. Valencia-Ramírez et al. (2014) compararon el paradigma de la programación genética contra el método de selección de modelo completo propuesto en Escalante et al. (2009) y el algoritmo de clasificación SVM. Obtuvieron resultados competitivos los cuales sus autores atribuyen a una selección de características implícita en el proceso de creación de las funciones de clasificación. En el trabajo no se definen mecanismos para controlar el sobre ajuste. Rosales-Pérez et al. (2014a) propusieron un método de selección de modelo que contempla la selección de algoritmos de clasificación y la optimización de sus parámetros mediante un algoritmo evolutivo. La función de desempeño es multi objetivo ya que contempla la disminución del error de entrenamiento y la complejidad de los modelos obtenidos mediante la dimensión VC. Para determinar las mejores soluciones se utilizó el principio de optimalidad de Pareto y debido a que el método permite

obtener un conjunto de soluciones óptimas, los autores propusieron un ensamble de las mejores soluciones como clasificador final. El uso de la dimensión VC pese a su utilidad, tiene un gran impacto en el tiempo de cómputo de la función de evaluación. En Rosales-Pérez et al. (2015) se enfrentan los aspectos de selección de características, métodos de pre procesamiento y optimización de los parámetros del algoritmo de clasificación SVM. Se utiliza un algoritmo genético como método de búsqueda y la función de desempeño toma en cuenta el sesgo y la varianza (multi objetivo). Para determinar cuáles son las soluciones óptimas también se utiliza el principio de optimalidad de Pareto. A causa del costo computacional de la función de desempeño, se propuso la utilización de Funciones subrogadas. Dichas funciones son utilizadas para predecir el posible desempeño de un modelo en base a las características o atributos de los sujetos o modelos en la población del algoritmo genético. La disminución del tiempo de cómputo mediante el uso de funciones subrogadas, presenta una buena alternativa para utilizar funciones de evaluación más rigurosas que permitirán obtener modelos de una complejidad moderada, pero disminuyendo el tiempo que se requeriría para la evaluación de todos los individuos de las generaciones del AG. Esta técnica, presenta una gran oportunidad para poder disminuir los tiempos de cómputo que requiere la selección de modelo en Big Data. Sparks et al. (2015), desarrollaron un sistema para automatizar el proceso de entrenamiento y selección de algoritmos de aprendizaje, así como el ajuste de sus hiperparámetros en conjuntos de datos de gran volumen. Este sistema consta de un gestor de recursos para entrenar y probar diversos modelos de forma simultánea en clústeres que funcionen bajo el esquema MapReduce. La aportación de este trabajo no recae en el método de búsqueda o la forma de evaluación de los modelos obtenidos, en lugar de esto, el sistema es capaz utilizar diversos métodos de búsqueda como rejilla, HyperOpt de Bergstra et al. (2013) y Auto-Weka de Thorton et al. (2009) e incluso en futuros desarrollos será extensible a otros. El algoritmo de gestión de recursos, en base al algoritmo de búsqueda de modelo, realiza el entrenamiento simultáneo parcial (un número reducido de iteraciones de los algoritmos) para descartar soluciones poco promisorias en etapas tempranas y realizar el entrenamiento a fondo de los modelos con buen potencial. El número de modelos a evaluar se define de acuerdo a un presupuesto de tiempo de cómputo que el usuario determina. En los experimentos reportados se probaron poco más de 100 modelos para poder encajar dentro del presupuesto de tiempo de cómputo. En el trabajo no se analizan métodos para medir o controlar la complejidad de los modelos obtenidos ni métodos de evaluación del desempeño de los modelos obtenidos. Kaneko & Funatsu (2015), propusieron un método para la selección de parámetros del algoritmo SVR (Regresión de Soporte de Vectores) basado en la técnica de búsqueda en rejilla (GS) y la técnica de decisión teórica de hiperparámetros de Cherkassky y Ma (ThD). Una parte de los parámetros son optimizados mediante la búsqueda en rejilla y los restantes mediante ThD, con lo cual el número de combinaciones a evaluar es mucho menor que el total de las combinaciones en una búsqueda de tipo rejilla convencional. Sin embargo, pese a la utilidad del método, solo está disponible para el algoritmo SVR. Bansal & Sahoo (2015) exploraron el uso del algoritmo murciélago (BAT) para realizar la selección de modelo completo. Dicho algoritmo está basado en el sentido de eco localización de los murciélagos para detectar comida y calcular la distancia hasta ella. La función de desempeño considera la disminución del error de clasificación y el menor subconjunto de características, mediante una suma ponderada de estos criterios. En un sentido estricto, la mayor aportación de este trabajo, es considerar dentro de la función de desempeño el número de características a utilizar en cada modelo, con lo cual de cierta forma se controla la complejidad de los mismos.

En los párrafos anteriores se describió brevemente algunos de los trabajos sobre selección de modelo presentes en la literatura. De lo anteriormente analizado, el lector puede percatarse que son pocos los trabajos que abordan el problema de la selección de modelo (Escalante et al., 2009; Thorton et al., 2009; Rosales-Pérez et al., 2014a; Rosales-Pérez et al., 2015; Bansal & Sahoo, 2015) y el resto (Guo et al.,2008; Chatelain

et al.,2010; Aydin et al.,2011; Oh et al.,2014; Kaneko & Funatsu, 2015) se enfocan en la optimización de los hiperparámetros de un solo algoritmo de aprendizaje. Los trabajos sobre selección de modelo proporcionan evidencia de que este enfoque puede ser empleado en una mayor cantidad de conjuntos de datos, ya que, al tener una mayor cantidad de algoritmos de aprendizaje a evaluar, existen mayores probabilidades de que alguno sea más adecuado a un conjunto de datos en particular. También es necesario notar que el enfoque de selección de modelo, no requiere conocimiento a priori del conjunto de datos a analizar y si estos han sido procesados de forma adecuada para el mejor funcionamiento del algoritmo de aprendizaje. Por otra parte, puede notarse una tendencia hacia las búsquedas multi objetivo, como se aprecia en los trabajos de: Chatelain et al. (2010), Aydin et al. (2011), Rosales-Pérez et al. (2014[a],2015), Bansal & Sahoo (2015). Esto se debe a la poca fiabilidad que puede obtenerse al utilizar un solo estimador del desempeño de un modelo, como es el cálculo del error de clasificación, el cual, es empleado en gran parte de los trabajos. Sin embargo, solo en: Aydin et al. (2011), Rosales-Pérez et al. (2014a) y Bansal & Sahoo (2015), se contempla la complejidad de los modelos como criterio de selección. No obstante, solo en Rosales-Pérez et al. (2014a) presenta una forma de medir la complejidad de los modelos potenciales, extensible a cualquier algoritmo de aprendizaje (dimensión VC) a diferencia del método de Aydin et al. (2011) que solo es aplicable al algoritmo SVM (número de vectores de soporte empleados). También dentro del tópico de los métodos de búsqueda multi objetivo, la mitad de estos hace uso del frente de Pareto, mientras que los restantes resuelven el problema mediante la suma ponderada de los criterios considerados. Aunque la suma ponderada es un método más sencillo que el frente de Pareto, este último provee una mayor variedad de soluciones, de acuerdo a la evidencia presentada en los trabajos de Rosales-Pérez et al. (2014a,2015) y Chatelain et al. (2010). Otro aspecto relevante en lo referido a la selección de modelo, es el tiempo empleado en el entrenamiento y prueba de las soluciones potenciales, las cuales son durante la mayor parte de la búsqueda, soluciones inferiores y solo en unos cuantos casos soluciones aceptables. Sin importar el método de búsqueda empleado, será necesario realizar este proceso una cantidad de veces, por lo cual, se ha propuesto el empleo de funciones subrogadas (Rosales-Pérez et al., 2015; Thorton et al., 2009). El empleo de estas técnicas puede ser de gran utilidad en el ámbito de los conjuntos de datos de gran volumen al disminuir el tiempo empleado en entrenar modelos que pueden ser potencialmente poco eficientes y de esta forma poder emplear métodos de búsqueda más adecuados. En los trabajos de Chan et al. (2013) y Sparks et al. (2015) se enfrenta el problema de la selección de modelo en Big Data por medio del modelo de programación MapReduce, pero el sistema de búsqueda empleado es, con algunas diferencias, similar a la búsqueda en rejilla, la cual, de acuerdo a la evidencia proporcionada en Bergstra & Bengio (2012), es menos eficiente que la búsqueda aleatoria.

Tomando en cuenta todos los trabajos analizados, podemos obtener las propiedades con las cuales debe de contar un algoritmo para selección de modelo en conjuntos de gran volumen de datos. En primera instancia, el método de búsqueda no puede ser basado en rejilla y las capacidades de los algoritmos basados en poblaciones demuestran obtener excelentes resultados. Una buena opción para enfrentar la selección de modelo en conjuntos de gran volumen de datos es por medio del modelo de programación MapReduce debido a su escalabilidad, sin embargo, en lugar de seguir el método propuesto por Sparks et al. (2015) de limitar la búsqueda a un cierto número de modelos, sería más conveniente disminuir el número de evaluaciones de modelos por medio de funciones subrogadas y así asegurar la estabilidad en la ejecución de un algoritmo que podría tomar una considerable cantidad de tiempo en ejecutarse. Para evitar obtener modelos sobre ajustados es necesario medir la complejidad de los modelos analizados y dentro de este mismo orden de ideas los trabajos que emplean métodos de búsqueda multi objetivo obtienen una mayor variedad en las soluciones obtenidas. Por estas razones, dichos aspectos serán considerados en el desarrollo del presente proyecto.

4 Propuesta

4.1 Definición del problema

La selección de modelo puede ser descrita como el proceso de elegir al modelo que mejor describe un conjunto de datos [54]. Este proceso implica la comparación en el desempeño de un conjunto de algoritmos de aprendizaje donde, cada configuración de los hiperparámetros de un algoritmo, representa un modelo potencial que deberá ser comparado con otros modelos con el fin de elegir al más adecuado a un conjunto de datos. En el enfoque de selección de modelo completo, un modelo potencial consta de lo antes mencionado además del pre procesamiento de los datos mediante alguna técnica y la selección de un subconjunto de características a través de algún algoritmo para este fin. Las combinaciones entre los valores de los diferentes factores que involucra la selección de modelo completo, desembocan en un gran espacio de búsqueda. La evaluación del desempeño de una combinación o modelo potencial representa una serie de transformaciones en el conjunto de datos y el proceso de aprendizaje, es decir, la construcción de ese modelo en particular. Dicha construcción conlleva largos tiempos de ejecución, lo cual aumenta en relación al tamaño de los conjuntos de datos utilizados. Para poder encontrar los modelos más adecuados a un conjunto con gran volumen de datos es necesario un algoritmo que pueda afrontar la construcción de los modelos en conjuntos de datos de este ámbito.

4.2 Hipótesis

La selección de modelo completo en conjuntos con gran volumen de datos se logrará a través del uso de técnicas de computo suave, meta-aprendizaje y el paradigma de programación MapReduce obteniendo tasas de clasificación errónea menores a las obtenidas por los algoritmos de la linea base y con un menor número de evaluaciones que dichos algoritmos.

4.3 Objetivo general

Proponer diversas estrategias para realizar selección de modelo completo en conjuntos con gran volumen de datos, haciendo uso del paradigma de programación MapReduce, técnicas de cómputo suave y meta-aprendizaje.

4.4 Objetivos específicos

- Diseñar y adaptar algoritmos para realizar selección de modelo completo al paradigma de programación MapReduce con el fin de efectuar selección de modelo completo en conjuntos con gran volumen de datos.
- Analizar el efecto de la selección de modelo completo en la construcción de modelos proxy (también conocidos como funciones surrogadas) para guiar la búsqueda de los algoritmos para selección de modelo completo en conjuntos con gran volumen de datos y la disminución del número de funciones de evaluación utilizando algoritmos de regresión.
- Analizar el efecto de la selección de modelo completo en la construcción de modelos proxy para dirigir la búsqueda de los algoritmos para selección de modelo completo en conjuntos con gran volumen de datos y la disminución del número de funciones de evaluación utilizando algoritmos de clasificación.

- Analizar el efecto de la selección de modelo completo en la construcción de modelos proxy para guiar la búsqueda de los algoritmos para selección de modelo completo en conjuntos con gran volumen de datos y la disminución del número de funciones de evaluación utilizando algoritmos de clasificación difusos.
- Analizar el efecto del uso del meta-aprendizaje para realizar selección de modelo completo en conjuntos con gran volumen de datos.
- Analizar el efecto del uso del meta-aprendizaje para sugerir puntos iniciales promisorios para un algoritmo de búsqueda para efectuar selección de modelo completo en conjuntos con gran volumen de datos.
- Analizar el efecto del uso del meta-aprendizaje en conjunto con el uso de selección de modelo completo para construir modelos proxy que asistirán a un algoritmo de selección de modelo completo en conjuntos con gran volumen de datos.

4.5 Metodología propuesta

Con el fin de lograr los objetivos antes mencionados se propone la siguiente metodología:

- Crear conjuntos de datos sintéticos para probar el desempeño del algoritmo propuesto que contenga características con poder discriminativo y características no informativas.
- Se analizará la dimensión intrínseca de los conjuntos de datos propuestos, con el fin de tener evidencias de que cada conjunto de datos representa un problema computacional diferente.
- Diseñar e implementar algoritmos para selección de modelo completo en un marco de desarrollo de aplicaciones basado en el modelo de programación MapReduce.
- Se implementará el algoritmo K-NN bajo el modelo de programación MapReduce y se utilizará como línea base. Se determinará experimentalmente la configuración óptima de los parámetros del algoritmo para realizar el proceso de selección de modelo con el fin de establecer cuál configuración tiene un mejor desempeño.
- Se seguirá el protocolo experimental "interno" propuesto por Cawley y Talbot (2010) con el fin de evitar estimaciones optimistas en el desempeño de los modelos obtenidos.
- Se analizará experimentalmente el desempeño de los algoritmos propuestos utilizando: a) métodos de regularización (Cawley & Talbot, 2007), b) terminación temprana (Qi et al., 2004) y c) dimensión Vapnik-Chervonenkis (VC) (Vapik et al.,1994) para penalizar/atenuar los efectos del sobre ajuste de los parámetros de los modelos.
- Se analizará experimentalmente el desempeño de los algoritmos propuestos utilizando: a) modelos proxy, b) meta-aprendizaje, c) hibridación de los métodos antes mencionados con el fin de disminuir el número de funciones de evaluación.
- Se comparará el desempeño de los algoritmos de selección de modelo resultantes con el desempeño de los algoritmos de selección de modelo presentes en Escalante et al. (2009), en los conjuntos de datos presentes en: Rosales-Pérez et al. (2014a, 2015).
- Se realizarán análisis de los resultados mediante los protocolos propuestos en la literatura.

4.6 Contribuciones

- Creación de un algoritmo para realizar selección de modelo completo en conjuntos con gran volumen de datos basado en MapReduce.
- Marco de desarrollo para adaptar algoritmos de búsqueda basados en poblaciones para realizar selección de modelo completo en conjuntos con gran volumen de datos.
- Uso del paradigma de selección de modelo completo para la construcción de modelos proxy.
- Uso del meta-aprendizaje para realizar selección de modelo completo en conjuntos con gran volumen de datos.

4.7 Cronograma de actividades por cuatrimestres

Tabla 4.1: Los cuatrimestres serán en el intervalo $[Enero - Abril]$, $[Mayo - Agosto]$ y $[Sep - Dic]$. Se empezará a contar desde Septiembre.

Actividades	**cuatrimestres**											
	2014	2015			2016			2017			2018	
	1	2	3	4	5	6	7	8	9	10	11	12
Análisis de la literatura												
Aprobar asignaturas obligatorias												
Redacción de propuesta doctoral												
Análisis de conjuntos de datos de gran volumen de datos presentes en la literatura												
Creación de conjuntos de datos sintéticos												
Determinación experimental de los parámetros de un algoritmo genético												
Experimentos para determinar el número optimo de nodos MapReduce												
Experimentos de SMC utilizando la dimensión Vapnik-Chervonenkis												
Experimentos de SMC utilizando regularización y redacción de artículo												
Experimentos de SMC utilizando modelos proxy y redacción de artículo												
Experimentos de SMC utilizando algoritmo de clasificación difuso y redacción de artículo												
Experimentos de SMC utilizando meta-aprendizaje y redacción de artículo												
Experimentos de SMC combinando modelos proxy y meta-aprendizaje												
Periodo de respaldo												
Redacción de tesis doctoral												
Defensa de tesis doctoral												

5 Experimentos y resultados preliminares

5.1 Desarrollo del primer algoritmo de selección de modelo completo

Como se detalló en el apartado anterior, se desarrolló un algoritmo de selección de modelo completo para conjuntos de datos de gran volumen basado en un algoritmo genético. Dicho algoritmo fue desarrollado en Apache Spark 1.3.0, un marco de desarrollo de aplicaciones basado en el modelo de programación MapReduce descrito en la sección 2.15. Este marco de desarrollo fue elegido por su capacidad para lidiar con algoritmos iterativos y la posibilidad de realizar procesamiento de datos en la memoria principal (si existe suficiente capacidad). Tales atributos, descritos anteriormente, hacen de este marco la mejor elección para el desarrollo de este proyecto. Para el proceso de búsqueda se desarrolló un algoritmo genético (AG) basado en la variante propuesta por [41] conocida como CHC (Cross-generational elitist selection, Heterogeneous recombination, Cataclysmic mutation). El algoritmo genético CHC difiere de un algoritmo genético simple debido a su comportamiento altamente elitista, en lugar de reemplazar la generación anterior con la nueva generación de descendientes, se realiza una competencia entre generaciones, esto significa que los descendientes deben competir contra sus padres por su supervivencia, de tal forma que se incrementa la probabilidad de los padres de permanecer dentro de la población. Para evitar una pronta convergencia hacia un óptimo local, el CHC mantiene la diversidad por medio de un operador de cruza que asegura la variedad en la descendencia. Por otro lado, cada vez que en toda una generación no se realiza una cruza, se produce un decremento en un contador que al sobrepasar un umbral producirá que se genere una población nueva [76]. A diferencia del enfoque del algoritmo CHC en su estado puro donde no se utiliza la mutación en su sentido clásico sino como un recurso para generar nuevas poblaciones, en el algoritmo propuesto para selección de modelo completo, se sigue ocupando el operador de mutación en su sentido tradicional y el operador de macro mutación del enfoque CHC como un recurso para evitar el estancamiento en la convergencia del algoritmo. Cada individuo codificado en el AG representa un modelo potencial, esto significa que para cada individuo es necesario aplicar ciertas transformaciones al conjunto de datos. Estas transformaciones desembocarán en un conjunto de datos nuevo que servirá para entrenar/probar al algoritmo de clasificación codificado en el individuo. Es en este punto donde el modelo de programación MapReduce permite hacer frente a las dificultades de efectuar transformaciones en conjuntos de datos de gran magnitud. Como se mencionó con anterioridad, MapReduce permite distribuir las transformaciones de un conjunto de datos en múltiples nodos trabajadores, los cuales trabajarán con su porción de datos asignada. Las porciones de datos transformadas serán unidas finalmente en la fase reduce. En el algoritmo 1 se detalla el algoritmo propuesto en este trabajo denominado Algoritmo Genético para Selección de Modelo Completo Basado en MapReduce en adelante **AGSMCMR**.

Algoritmo 1 AGSMCMR

```
1: procedure OBTIENEMODELO(ConjuntoDatos)
2:     Modelo = ∅
3:     Umbral_Incesto = ObtieneUI()
4:     Población = Crea_Población_Inicial()
5:     evalúaMR(Población)
6:     while ! CondiciónTerminación do
7:         Decendencia = Cruza(Población)
8:         Umbral_Incesto=Actualiza_Umbral_Incesto()
9:         if Umbral_Incesto≤ 0 then
10:            Umbral_Incesto = ObtieneUI()
11:            Decendencia = MutaciónCataclísmica()
12:        end if
13:        evaluaMR(Decendencia)
14:        Población = SelecciónElitista(Descendencia)
15:    end while
16:    Return Modelo(Población)
17: end procedure
```

La evaluación de cada modelo potencial involucra múltiples transformaciones en el conjunto de datos, además de la construcción de los clasificadores. La función de evaluación de desempeño fue desarrollada bajo el paradigma MapReduce, denotada en el algoritmo 1 como evalúaMR(). En la sección 2.1 se detalló que la selección de modelo completo está conformada por los procesos de selección de características, pre procesamiento de los datos y la configuración de los hiperparámetros de los algoritmos de clasificación. La función evalúaMR() está conformada por las sub funciones que llevan a cabo estos procesos, las cuales constan de las etapas Map y Reduce descritas con anterioridad. El algoritmo 2 presentan las etapas Map y Reduce del proceso de pre procesamiento de los datos.

Algoritmo 2 Algoritmo de preprocesamiento de datos

```
1: procedure ETAPA_MAP(ConjuntoDatos)
2:     for Fila en ConjuntoDatos do
3:         Clave = Índice_Fila                          ▷ Identificador dado a esa fila
4:         Valor = Lista(ValoresDeCaracterísticas(1..n))
5:     end for
6:     Return (Clave,Valor)
7: end procedure
8: procedure ETAPA_REDUCE((Clave,Valor))
9:     VectorTransformado =[ ]
10:    for valor en ListaDeValores_Clave do
11:        Aux = AplicaTransformación(valor)
12:        VectorTransformado =[VectorTransformado;Aux]
13:    end for
14:    Return (Nodo.Id,VectorTransformado)              ▷ Identificador del nodo trabajador
15: end procedure
```

En la etapa Map del algoritmo 2 cada fila en el conjunto de datos recibe un identificador dato en función al número de particiones en el conjunto de datos (Ej. 1,1,1,2,2,2,3,3,3 significa que el conjunto de datos será dividido entre tres nodos de trabajo). Este Índice_Fila será usado como clave y el valor (esto bajo el paradigma de pares clave-valor) consiste de un vector columna con los valores del conjunto de datos en esa fila. Por otra parte, en la etapa Reduce todos los subconjuntos serán transformados de acuerdo a la técnica de pre procesamiento de datos utilizada. Con el fin de realizar el proceso de selección de características, se implementó un enfoque de tipo filtro donde las características son ordenadas de acuerdo a su relevancia respecto a algún criterio. Las etapas Map y Reduce del proceso de selección de características se muestran en el algoritmo 3.

Algoritmo 3 Algoritmo de selección de características

```
1: procedure ETAPA_MAP(ConjuntoDatos)
2:     for Fila en ConjuntoDatos do
3:         Clave = Índice_Fila                                   ▷ Identificador dado a esa fila
4:         v = Valor_de_Característica                           ▷ Un vector columna con los valores de las características
5:         c = Valor_de_ClaseObjetivo                            ▷ (-1 o 1)
6:         Valor = Lista(c,v)
7:     end for
8:     Return (Clave,Valor)
9: end procedure
10: procedure ETAPA_REDUCE((Clave,Valor))
11:     for valor en ListaDeValores_Clave do
12:         ConjuntoAuxiliar = [ConjuntoAuxiliar ; (valor.c,valor.v)]
13:     end for
14:     ranking = ObtieneRanking(ConjuntoAuxiliar)                ▷ Obtiene el ranking de cada característica
15:     Return (Nodo.Id,ranking)                                  ▷ Identificador del nodo donde se proceso el subconjunto de datos
16: end procedure
```

En lo referente a la selección de características se optó por el enfoque filter, con lo cual en la etapa Map del algoritmo 3 la clave es otorgada de forma análoga a la etapa Map del algoritmo 2. Adicionalmente, la variable Valor consiste de una lista con los valores de las características y el valor de la clase objetivo (-1 o 1 en un problema de clasificación binaria). En la etapa Reduce, los pares intermedios son procesados y las características ordenadas de acuerdo al criterio del algoritmo de selección de características elegido. Finalmente, el proceso de clasificación es descrito en el algoritmo 4.

Algoritmo 4 Proceso de clasificacíon

```
1: procedure ETAPA_MAP(ConjuntoDatos)
2:     ParticiónConjuntoDatos =DivideConjunto(ConjuntoDatos,L)   ▷ Divide el conjunto de datos en L particiones
3:     for l en L do                                             ▷ Número de nodos disponibles
4:         Clave = l
5:         Valor = ParticiónConjuntoDatos
6:         Return (Clave,Valor)                                  ▷ Asigna una partición del conjunto de datos a un nodo
7:     end for
8: end procedure
9: procedure ETAPA_REDUCE((Clave,Valor))
10:     while !CriterioDeConvergencia do
11:         for l en L do                                         ▷ Número de nodos disponibles
12:             modelo_l = EntrenaModelo(D_l)                     ▷ D_l = Subconjunto en nodo "l"
13:         end for
14:         for l en L do                                         ▷ Número de nodos disponibles
15:             modelo_global = modelo_global ∪ modelo_l
16:         end for
17:     end while
18: end procedure
```

Para la construcción de un modelo para clasificación bajo el paradigma MapReduce, el conjunto de datos es dividido entre los nodos de trabajo disponibles como se muestra en la etapa Map. Posteriormente, en la etapa Reduce cada nodo trabajador utiliza su porción del conjunto de entrenamiento para construir un sub modelo. Estos sub modelos serán unidos en un modelo global para el conjunto de datos completo. Los algoritmos de clasificación empleados en este trabajo se encuentran en la tabla 5.1.

Tabla 5.1: Métodos de pre procesamiento, algoritmos de selección de características y algoritmos de clasificación utilizados en este trabajo.

Métodos de pre procesamiento	Estandarización de características Normalización Análisis de componentes principales (PCA) Escala y desplazamiento Discretización
Algoritmos de selección de características	Información mutua conjunta (JMI) Mínima redundancia - máxima relevancia (mrMR) Interacción bloqueo (ICAP) Maximización de información condicional mutua (CMIM) Fragmentos informativos (IF)
Algoritmos de clasificación	Máquina de soporte vectorial (SVM) Regresión logística (LR) Clasificador Bayes Ingenuo (NB) Árbol de decisión (DT) Bosque aleatorio (RF) Árboles de impulso gradiente (Boo)

Cada individuo en la población del algoritmo representa un modelo potencial para el proceso de clasificación de los conjuntos de datos. Cada modelo será codificado en un vector de longitud 16, y dada la naturaleza de los parámetros que se representarán en los individuos del algoritmo genético, se optó por una codificación con valores reales en lugar de binarios. En la figura 5.1 se detalla la función de cada elemento del individuo.

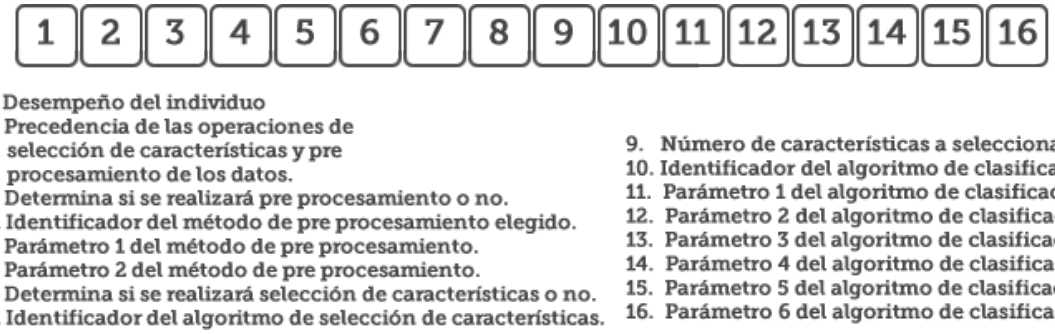

Fig. 5.1: Función de cada elemento de un individuo.

5.2 Experimentos

Con el fin de evaluar la factibilidad del algoritmo propuesto para selección de modelo completo, se utilizaron los conjuntos de datos que aparecen en la tabla 5.2. Dichos conjuntos de datos han sido utilizados en diversos trabajos que abordan problemas de clasificación en Big Data (del Rio, et al., 2014; del Rio, et al., 2015; López et al., 2015; López et al., 2015; entre otros). También se crearon 2 conjuntos de datos sintéticos, los cuales constan de 50,000 instancias y dos clases. El primero fue creado con 5 atributos con poder discriminativo, 25,121 instancias de la clase uno y 24,879 instancias en la clase dos. El segundo

consta de 10 atributos, 5 no informativos y 5 con poder discriminativo, 25,108 instancias de la clase uno y 24,892 instancias de la clase 2.

Tabla 5.2: Conjuntos de datos utilizados en los experimentos.

Conjunto de datos	Ejemplos	Atributos	Muestras por clase
Covtype_2_vs_1	495141	54	(283,301; 211,840)
Census-income	199523	40	(187,141; 12,382)
Fars_Fatal_Inj_vs No_Inj	50164	52	(17,445; 32,719)
Sintético 1	50000	5	(25,121;24,879)
Sintético 2	50000	10	(25,108;24,892)

Los parámetros del algoritmo genético fueron determinados experimentalmente al analizar el desempeño de las combinaciones de los factores: **Tamaño de Población = 4 individuos, 6 individuos, 8 individuos**; **Porcentaje de cruza: 50%, 70%, 90%** y **Porcentaje de mutación = 10%, 20%, 30 %** en la décima parte del conjunto Covtype_2_vs_1 por ser el conjunto con el mayor número de instancias y un número balanceado de características (52). Se realizaron 27 experimentos de selección de modelo fijando el número de generaciones a 25 y el criterio de terminación fue la ejecución del número establecido de generaciones. En esta primera etapa del proyecto se optó por utilizar tamaños de población bajos, ya que un manejo incipiente del paradigma MapReduce producía inestabilidad en la implementación del algoritmo y un número limitado de nodos de procesamiento ocasionó largos tiempos de ejecución. Actualmente, un mejor conocimiento del paradigma ha influido en la mejora de la estabilidad del algoritmo, con lo cual ha podido aumentarse la cantidad de la población hasta 40 individuos. El desempeño de los modelos fue estimado mediante el promedio de la validación cruzada a 10 pliegos del área bajo la curva. La validación cruzada fue empleada en los trabajos de Escalante et al. (2009) y parcialmente en Rosales-Pérez et al. (2014a) para la evaluación de los modelos, por lo cual se optó por esta técnica. Se optó por la métrica del promedio del área bajo la curva por contemplar tanto sensitividad como especificidad en el desempeño de los modelos. De los 27 experimentos realizados, la combinación que obtuvo el mayor desempeño fue: **población = 8 individuos**, porcentaje de **cruza = 50 %** y **porcentaje de mutación = 10 %**, por lo cual, estos son los valores que fueron empleados en los subsecuentes experimentos. También se realizaron pruebas con el algoritmo mínima redundancia máxima relevancia (mrMR) para identificar la dimensión intrínseca de los conjuntos de datos de la tabla 5.2. Esta información es de utilidad para poder juzgar el desempeño de los individuos y disponer evidencias de que los conjuntos de datos empleados representan problemas computacionales diferentes. En la tabla 5.3 se muestran los resultados del análisis.

Tabla 5.3: Análisis de las dimensiones intrínsecas de los conjuntos de datos.

Conjunto de datos	Número total de características	Dimensión intrínseca
Sintético 1	5	5
Sintético 2	10	5
Fars_Fatal_Inj_vs No_Inj	52	14
Census-income	40	20
Covtype_2_vs_1	54	54

Para evaluar el desempeño del algoritmo **AGSMCMR**, se realizaron pruebas en los conjuntos de datos de la Tabla 5.3. Siguiendo el protocolo propuesto por Cawley y Talbot (2010), utilizado también en Rosales-

Pérez et al. (2014a, 2015); se realizaron réplicas de los conjuntos de datos, pero a causa del gran volumen de los mismos se optó por disminuir el número de réplicas a 10, guiados por el precedente de los conjuntos de datos Image y Splice (2,310 y 3,175 instancias respectivamente) donde se realizaron 20 réplicas en lugar de 100. Los conjuntos de datos fueron divididos en 60% de las muestras para el conjunto de entrenamiento y el 40% restante para el conjunto de pruebas. El desempeño del individuo con el mayor grado de aptitud en la última generación del algoritmo genético fue comparado con los algoritmos disponibles en las librerías de aprendizaje automático del marco de desarrollo **Spark 1.3.0** que aparecen en la tabla 2 con sus hiperparámetros por defecto. En la tabla 5 se muestran los resultados obtenidos.

Tabla 5.4: Resultados obtenidos en el conjunto de prueba por el mejor individuo del algoritmo AGSMCMR y los obtenidos por los algoritmos de clasificación en Big Data: SVM, Árbol de decisión (DT), Bosque aleatorio (RF), Bayes Ingenuo (NB), Regresión logística (LR) y Árboles de impulso gradiente (Boo), tras 10 réplicas. Los mejores resultados para cada conjunto de datos están en **negritas**.

Conjunto de datos	AGSMCMR	SVM	DT	RF	NB	LR	Boo
Sintético 1	**0.998 ± 0.000**	0.888 ± 0.003	0.896 ± 0.001	0.911 ± 0.001	0.654 ± 0.003	0.975 ± 0.001	0.954 ± 0.001
Sintético 2	**0.997 ± 0.000**	0.894 ± 0.001	0.896 ± 0.001	0.904 ± 0.002	0.785 ±0.001	0.960 ±0.001	0.954 ± 0.001
Fars_Fatal_Inj_vs No_Inj	**0.998 ± 0.000**	0.589 ± 0.003	0.998 ± 0.000	0.956 ± 0.009	0.585 ± 0.001	0.500	0.998 ± 0.000
Census-income	**0.896 ± 0.003**	0.687 ± 0.055	0.609 ± 0.007	0.539 ± 0.004	0.617 ± 0.003	0.607 ± 0.003	0.675 ± 0.001
Covtype_2_vs_1	**0.898 ± 0.016**	0.542 ± 0.024	0.765 ± 0.002	0.693 ± 0.019	0.500	0.697 ± 0.002	0.813 ± 0.000

Posteriormente los datos fueron analizados con la prueba de normalidad Shapiro-Wilk y Kolmogorov-Smirnov, resultando en todos los casos que los datos tienen una distribución normal. A continuación, se realizó una prueba ANOVA con un 95% de confianza y una prueba Pos hoc Tukey. En la tabla 5.5 se muestran los resultados.

Tabla 5.5: Estadístico F de la prueba ANOVA y valores "q" de la prueba pos hoc Tukey con todas las comparaciones por pares entre el mejor individuo (MI) del algoritmo propuesto y los algoritmos de la línea base. El valor crítico de la prueba ANOVA al nivel de confianza al 95% es 2.246 (F(6,63)) para todos los conjuntos de datos. El valor crítico de la prueba Tukey al nivel de confianza del 95% es de 4.306 con 63 grados de libertad para todos los conjuntos de datos. Los casos que exceden el valor crítico son considerados como que poseen una diferencia estadísticamente significativa y son marcados con *.

Conjunto de datos	ANOVA F	MI vs SVM	MI vs DT	MI vs RF	MI vs NB	MI vs LR	MI vs Boo
Sintético 1	**23,895.97***	**147.856***	**137.736***	**117.623***	**463.389***	**30.030***	**58.626***
Sintético 2	**15,718.64***	**188.535***	**185.229***	**171.591***	**388.782***	**68.414***	**78.685***
Fars_Fatal_Inj_vs No_Inj	**35,887.01***	**334.292***	0.1786	**34.903***	**337.212***	**407.447***	0.2433
Covtype_2_vs_1	**1,111.722***	**83.435***	**31.162***	**47.992***	**93.396***	**47.042***	**19.896***

Como puede apreciarse en la tabla 5.5, el modelo codificado en el mejor individuo del algoritmo AGSMCMR tiene un nivel de desempeño superior al de los algoritmos de la línea base en casi todos los conjuntos de datos analizados. Estos resultados muestran el impacto sobre el desempeño de un proceso de clasificación cuando se realiza una adecuada selección de características y pre procesamiento de datos. Sin embargo, en el conjunto de datos Fars_Fatal_Inj_vs No_Inj, el desempeño del mejor individuo no consiguió diferencias estadísticamente significativas en comparación a los algoritmos DT y Boo. Con el fin de poder mostrar las capacidades del algoritmo AGSMCMR en comparación a las de otros algoritmos de selección de modelo presentes en la literatura, se realizaron experimentos de selección de modelo en los conjuntos de datos uti-

lizados en Rosales-Pérez et al. (2014a, 2015). En la tabla 5.6 se observan los resultados obtenidos por los modelos del algoritmo AGSMCMR en comparación por los resultados reportados de los algoritmos PSMS, MOMTS-S2 y SAMOMS en los trabajos antes mencionados.

Tabla 5.6: Resultados promedio tras 100 o 20 réplicas de los algoritmos de selección de modelo: PSMS, MOMTS-S2, SAMOMS, AGSMCMR. Los mejores resultados están en **negritas**.

Conjunto de datos	PSMS	MOMTS-S2	SAMOMS	AGSMCMR
Banana	11.08±0.083	**10.48± 0.046**	10.65±0.054	12.71±0.840
Breast Cancer	33.01±0.658	25.61±0.593	28.22±0.506	**25.32±4.590**
Diabetes	27.06±0.658	23.08±0.174	24.46±0.212	**22.96±2.220**
Flare Solar	34.81±0.173	34.59±0.189	**33.04±0.236**	33.12±1.820
German	30.10±0.720	**23.67±0.224**	24.55±0.230	24.33±3.110
Heart	20.69±0.634	16.48±0.241	16.19±0.373	**15.30±2.950**
Image	2.90±0.112	**2.24±0.123**	3.73±0.117	2.267±0.760
Ringnorm	7.98±0.660	2.49±0.074	**1.81±0.035**	10.31±6.580
Splice	14.63±0.324	**4.84±0.156**	8.31±0.114	5.142±2.200
Thyroid	4.32±0.235	**4.00±0.194**	5.09±0.240	4.186±2.050
Titanic	24.18±0.193	22.08±0.085	23.19±0.217	**22.07±0.780**
Twonorm	3.09±0.127	3.73±0.179	**2.58±0.034**	2.883±0.730
Waveform	12.80±0.325	**9.93±0.043**	10.56±0.108	12.59±1.770

Puede observarse que el desempeño del algoritmo AGSMCMR es competitivo y que obtiene los mejores resultados en 4 de 13 conjuntos de datos y aunque probablemente las diferencias no son estadísticamente significantes, es un indicador positivo de la calidad del algoritmo al ser comparado con trabajos líderes en el ámbito de selección de modelo. Sin embargo, existe gran varianza en los resultados obtenidos, razón por la cual es necesario seguir refinándolo. Posteriormente se realizaron pruebas de selección de modelo con el algoritmo PSMS en los conjuntos de datos de la tabla 4 reducidos mediante muestreo aleatorio. En este punto, la reducción mediante muestreo aleatorio nos permite conocer la cantidad máxima de instancias que el algoritmo PSMS puede procesar con estabilidad durante la ejecución. Más adelante se utilizarán mejores técnicas de muestreo, pero con el conocimiento del número de instancias adecuado. Para este experimento se utilizaron particiones del 1% y 5% del total de instancias para los conjuntos de entrenamiento y el resto para los conjuntos de prueba. Se realizaron 10 réplicas para cada conjunto de datos en cada porcentaje. En la tabla 5.7 se muestran los resultados.

Tabla 5.7: Resultados obtenidos por el algoritmo PSMS en conjuntos de datos Reducidos mediante muestreo al 1% y 5%

Conjunto de datos	Error al 1%	Error al 5%
Covtype_2_vs_1 (495,141)	N/A	N/A
Census-income (199,523)	18.30±1.57	N/A
Fars_Fatal_Inj_vs No_Inj (50,164)	1.31±0.73	0.5±0.72
Sintético 1 (50,000)	5.62±2.39	3.75±1.61
Sintético 2 (50,000)	3.11±0.82	1.41±0.34

Como puede observarse, en los conjuntos de datos Sintético 2 con un conjunto de entrenamiento del 5%

del total de instancias (2,500 instancias) y en Fars_Fatal_Inj vs No_inj al 1% (500 instancias) y al 5% (2,508 instancias) se obtuvieron resultados similares a los obtenidos por el algoritmo propuesto. En las columnas donde aparece N/A (no aplica), el algoritmo no se ejecutó de manera estable (largos tiempos de ejecución / Matlab no responde) debido a la cantidad de datos. En el caso del conjunto de datos Covtype se realizó una reducción al 0.5% (2,476 instancias) sin obtener una ejecución estable y aun se busca la causa del error. Pese a lo básico del método de muestreo aleatorio, puede notarse que para ciertos conjuntos de datos es suficiente para obtener buenos resultados y permite obtener una estimación del número máximo de instancias que al algoritmo puede procesar, en este caso alrededor de 2,500 instancias. Sin embargo, podemos suponer que 2,500 instancias podrían significar que, no importando la calidad del método de muestreo, el tamaño de la muestra no sea representativa del conjunto de datos. Más adelante se realizarán pruebas con los algoritmos MOMTS-S2 y SAMOMS además de otras técnicas de muestreo para obtener evidencias que puedan dar respuesta a esta pregunta.

6 Conclusiones parciales

En trabajo se abordó el problema de selección de modelo completo para conjuntos de datos pertenecientes al ámbito de los conjuntos con gran volumen de datos, comúnmente denominados Big Data. Se propuso el desarrollo de un algoritmo de selección de modelo basado en un algoritmo genético, desarrollado bajo el modelo de programación MapReduce. Los resultados experimentales muestran que, es posible realizar un algoritmo de selección de modelo bajo el paradigma antes mencionado. Los modelos obtenidos demuestran una importante mejoría en el poder predictivo en comparación con los algoritmos de clasificación en Big Data con los parámetros por defecto. Es importante mencionar que el desempeño del algoritmo propuesto en comparación con otros algoritmos líderes del campo de selección de modelo proporciona un indicador positivo de la calidad del mismo. Por otro lado, los resultados preliminares de selección de modelo mediante la reducción de conjuntos de datos a través de muestreo, revelan que esta alternativa es útil cuando el número de instancias que el algoritmo puede procesar es representativo del conjunto de datos. Como trabajo futuro se buscará reducir el número de funciones de evaluación y se explorarán medidas para atenuar los efectos del sobre ajuste de los modelos obtenidos. Se realizarán pruebas con otras técnicas de muestreo y otros algoritmos de selección de modelo para reunir evidencias que den respuesta a si puede realizarse selección de modelo mediante algoritmos convencionales y muestreo o mediante metodologías basadas en el paradigma MapReduce.

References

[1] Daniel J Abadi, Peter A Boncz, and Stavros Harizopoulos. Column-oriented database systems. *Proceedings of the VLDB Endowment*, 2(2):1664–1665, 2009.

[2] Ben Jemaa Abdelhak, Essounbouli Najib, Hamzaoui Abdelaziz, Faicel Hnaien, and Farouk Yalaoui. Optimum sizing of hybrid pv/wind/battery using fuzzy-adaptive genetic algorithm in real and average battery service life. In *Power Electronics, Electrical Drives, Automation and Motion (SPEEDAM), 2014 International Symposium on*, pages 871–876. IEEE, 2014.

[3] Renzo Angles and Claudio Gutierrez. Survey of graph database models. *ACM Computing Surveys (CSUR)*, 40(1):1, 2008.

[4] Nedjem-Eddine Ayat, Mohamed Cheriet, and Ching Y Suen. Automatic model selection for the optimization of svm kernels. *Pattern Recognition*, 38(10):1733–1745, 2005.

[5] Ilhan Aydin, Mehmet Karakose, and Erhan Akin. A multi-objective artificial immune algorithm for parameter optimization in support vector machine. *Applied soft computing*, 11(1):120–129, 2011.

[6] Bhavna Bansal and Anita Sahoo. Full model selection using bat algorithm. In *Cognitive Computing and Information Processing (CCIP), 2015 International Conference on*, pages 1–4. IEEE, 2015.

[7] Yoshua Bengio. Practical recommendations for gradient-based training of deep architectures. In *Neural networks: Tricks of the trade*, pages 437–478. Springer, 2012.

[8] James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13(Feb):281–305, 2012.

[9] James Bergstra, Dan Yamins, and David D Cox. Hyperopt: A python library for optimizing the hyperparameters of machine learning algorithms. In *Proceedings of the 12th Python in Science Conference*, pages 13–20, 2013.

[10] J Bourquard and C Kirsch. Big data= big benefits: states are using large amounts of data to improve efficiency, fight fraud, and identify savings. *State legislatures*, 40(8):32, 2014.

[11] Damien Brain and G Webb. On the effect of data set size on bias and variance in classification learning. In *Proceedings of the Fourth Australian Knowledge Acquisition Workshop, University of New South Wales*, pages 117–128, 1999.

[12] Jan Burgess. How retailers use big data for product customization. `http://www.ingrammicroadvisor.com/data-center/how-retailers-use-big-data-for-product-customization`, 2017.

[13] James E Burt, Gerald M Barber, and David L Rigby. *Elementary statistics for geographers*. Guilford Press, 2009.

[14] Francesco Camastra and Alessandro Vinciarelli. Machine learning for audio, image and video analysis. *Advanced Information and Knowledge Processing*, pages 83–89, 2008.

[15] Gavin C Cawley and Nicola LC Talbot. Preventing over-fitting during model selection via bayesian regularisation of the hyper-parameters. *Journal of Machine Learning Research*, 8(Apr):841–861, 2007.

[16] Gavin C Cawley and Nicola LC Talbot. On over-fitting in model selection and subsequent selection bias in performance evaluation. *Journal of Machine Learning Research*, 11(Jul):2079–2107, 2010.

[17] CERN. Computing — cern. `http://home.cern/about/computing`, 2017.

[18] Simon Chan, Philip Treleaven, and Licia Capra. Continuous hyperparameter optimization for large-scale recommender systems. In *Big Data, 2013 IEEE International Conference on*, pages 350–358. IEEE, 2013.

[19] Clément Chatelain, Sébastien Adam, Yves Lecourtier, Laurent Heutte, and Thierry Paquet. A multi-model selection framework for unknown and/or evolutive misclassification cost problems. *Pattern Recognition*, 43(3):815–823, 2010.

[20] Lei Chen, Peter Triantafillou, and Torsten Suel. Web information systems engineering-wise 2010. *Lecture Notes in Computer Science*, 6488, 2010.

[21] Xueying Chen and Min-ge Xie. A split-and-conquer approach for analysis of extraordinarily large data. *Statistica Sinica*, pages 1655–1684, 2014.

[22] Jeoung-Nae Choi, Young-Il Lee, and Sung-Kwun Oh. Fuzzy radial basis function neural networks with information granulation and its genetic optimization. *Advances in Neural Networks–ISNN 2009*, pages 127–134, 2009.

[23] Marc Claesen and Bart De Moor. Hyperparameter search in machine learning. *arXiv preprint arXiv:1502.02127*, 2015.

[24] Edgar F Codd. Extending the database relational model to capture more meaning. *ACM Transactions on Database Systems (TODS)*, 4(4):397–434, 1979.

[25] Carlos Coronel and Steven Morris. *Database systems: design, implementation, & management*. Cengage Learning, 2016.

[26] Michael Cox and David Ellsworth. Managing big data for scientific visualization. In *ACM Siggraph*, volume 97, pages 21–38, 1997.

[27] Pedro Ponce Cruz and Alejandro Herrera. *Inteligencia artificial con aplicaciones a la ingeniería*. Marcombo, 2011.

[28] S. Das. *Data Science Using Oracle Data Miner and Oracle R Enterprise: Transform Your Business Systems into an Analytical Powerhouse*. Apress, 2016.

[29] Jeffrey Dean and Sanjay Ghemawat. Mapreduce: simplified data processing on large clusters. *Communications of the ACM*, 51(1):107–113, 2008.

[30] Jeffrey Dean and Sanjay Ghemawat. Mapreduce: simplified data processing on large clusters. *Communications of the ACM*, 51(1):107–113, 2008.

[31] Sara Del Río, Victoria López, José Manuel Benítez, and Francisco Herrera. On the use of mapreduce for imbalanced big data using random forest. *Information Sciences*, 285:112–137, 2014.

[32] Sara del Río, Victoria López, José Manuel Benítez, and Francisco Herrera. A mapreduce approach to address big data classification problems based on the fusion of linguistic fuzzy rules. *International Journal of Computational Intelligence Systems*, 8(3):422–437, 2015.

[33] Janez Demšar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine learning research*, 7(Jan):1–30, 2006.

[34] David DeWitt and Jim Gray. Parallel database systems: the future of high performance database systems. *Communications of the ACM*, 35(6):85–98, 1992.

[35] R.M. Díaz, F. Pichler, and A.Q. Arencibia. *Computer Aided Systems Theory - EUROCAST 2009: 12th International Conference, Las Palmas de Gran Canaria, Spain, February 15-20, 2009, Revised Selected Papers*. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2009.

[36] Jörg Drechsler and Jerome P Reiter. An empirical evaluation of easily implemented, nonparametric methods for generating synthetic datasets. *Computational Statistics & Data Analysis*, 55(12):3232–3243, 2011.

[37] Sašo Džeroski, Panče Panov, Dragi Kocev, and Ljupčo Todorovski. *Discovery Science: 17th International Conference, DS 2014, Bled, Slovenia, October 8-10, 2014, Proceedings*, volume 8777. Springer, 2014.

[38] Issam El Naqa, Ruijiang Li, Martin J Murphy, et al. Machine learning in radiation oncology. *Theory Appl*, pages 57–70, 2015.

[39] Hugo Jair Escalante, Manuel Montes, and Luis Enrique Sucar. Particle swarm model selection. *Journal of Machine Learning Research*, 10(Feb):405–440, 2009.

[40] Hugo Jair Escalante, Manuel Montes-y Gómez, Jesús A González, Pilar Gómez-Gil, Leopoldo Altamirano, Carlos A Reyes, Carolina Reta, and Alejandro Rosales. Acute leukemia classification by ensemble particle swarm model selection. *Artificial intelligence in medicine*, 55(3):163–175, 2012.

[41] Larry J Eshelman. The chc adaptive search algorithm: How to have safe search when engaging. *Foundations of Genetic Algorithms 1991 (FOGA 1)*, 1:265, 2014.

[42] Usama M Fayyad, David Haussler, and Paul E Stolorz. Kdd for science data analysis: Issues and examples. In *KDD*, pages 50–56, 1996.

[43] Salvador García, Julián Luengo, and Francisco Herrera. *Data preprocessing in data mining*. Springer, 2015.

[44] Lluís Godo. *Symbolic and quantitative approaches to reasoning with uncertainty*. Springer, 2005.

[45] Michael T Goodrich, Nodari Sitchinava, and Qin Zhang. Sorting, searching, and simulation in the mapreduce framework. In *ISAAC*, volume 7074, pages 374–383. Springer, 2011.

[46] XC Guo, JH Yang, CG Wu, CY Wang, and YC Liang. A novel ls-svms hyper-parameter selection based on particle swarm optimization. *Neurocomputing*, 71(16):3211–3215, 2008.

[47] Gavin Hackeling. *Mastering Machine Learning with scikit-learn*. Packt Publishing Ltd, 2014.

[48] Alon Halevy, Peter Norvig, and Fernando Pereira. The unreasonable effectiveness of data. *IEEE Intelligent Systems*, 24(2):8–12, 2009.

[49] Randy L Haupt and Sue Ellen Haupt. *Practical genetic algorithms*. John Wiley & Sons, 2004.

[50] Timothy C Havens, James C Bezdek, Christopher Leckie, Lawrence O Hall, and Marimuthu Palaniswami. Fuzzy c-means algorithms for very large data. *IEEE Transactions on Fuzzy Systems*, 20(6):1130–1146, 2012.

[51] Heiko Hofer, Christian Borgelt, and Michael R Berthold. Large scale mining of molecular fragments with wildcards. In *IDA*, volume 2810, pages 376–385. Springer, 2003.

[52] D.S. Huang, P. Gupta, X. Zhang, and P. Premaratne. *Emerging Intelligent Computing Technology and Applications: 8th International Conference, ICIC 2012, Huangshan, China, July 25-29, 2012. Proceedings*. Communications in Computer and Information Science. Springer Berlin Heidelberg, 2012.

[53] Heikki Huttunen and Jussi Tohka. Model selection for linear classifiers using bayesian error estimation. *Pattern Recognition*, 48(11):3739–3748, 2015.

[54] ZQ John Lu. The of statistical learning: data mining, inference, and prediction. *Journal of the Royal Statistical Society: Series A (Statistics in Society)*, 173(3):693–694, 2010.

[55] Willem Jonker and Milan Petković. *Secure Data Management*. Springer, 2011.

[56] Hiromasa Kaneko and Kimito Funatsu. Fast optimization of hyperparameters for support vector regression models with highly predictive ability. *Chemometrics and Intelligent Laboratory Systems*, 142:64–69, 2015.

[57] Mikhail Kanevski, Alexei Pozdnoukhov, and Vadim Timonin. *Machine learning for spatial environmental data: theory, applications, and software*. EPFL press, 2009.

[58] Rohit Khare, Doug Cutting, Kragen Sitaker, and Adam Rifkin. Nutch: A flexible and scalable open-source web search engine. *Oregon State University*, 1:32–32, 2004.

[59] Hyoung-Joong Kim and Stefan Katzenbeisser. Transactions on data hiding and multimedia security.

[60] Ariel Kleiner, Ameet Talwalkar, Purnamrita Sarkar, and Michael I Jordan. A scalable bootstrap for massive data. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 76(4):795–816, 2014.

[61] Jacek Laskowski. Mastering apache spark, 2016.

[62] Fabien Lauer and Gérard Bloch. Ho–kashyap classifier with early stopping for regularization. *Pattern recognition letters*, 27(9):1037–1044, 2006.

[63] Faming Liang, Yichen Cheng, Qifan Song, Jincheol Park, and Ping Yang. A resampling-based stochastic approximation method for analysis of large geostatistical data. *Journal of the American Statistical Association*, 108(501):325–339, 2013.

[64] Jimmy Lin and Chris Dyer. Data-intensive text processing with mapreduce. *Synthesis Lectures on Human Language Technologies*, 3(1):1–177, 2010.

[65] Nan Lin and Ruibin Xi. Aggregated estimating equation estimation. *Statistics and Its Interface*, 4(1):73–83, 2011.

[66] Victoria Lopez, Sara del Rio, Jose Manuel Benitez, and Francisco Herrera. On the use of mapreduce to build linguistic fuzzy rule based classification systems for big data. In *Fuzzy Systems (FUZZ-IEEE), 2014 IEEE International Conference on*, pages 1905–1912. IEEE, 2014.

[67] Victoria López, Sara del Río, José Manuel Benítez, and Francisco Herrera. Cost-sensitive linguistic fuzzy rule based classification systems under the mapreduce framework for imbalanced big data. *Fuzzy Sets and Systems*, 258:5–38, 2015.

[68] Boris Lublinsky, Kevin T Smith, and Alexey Yakubovich. *Professional hadoop solutions*. John Wiley & Sons, 2013.

[69] Ping Ma, Michael Mahoney, and Bin Yu. A statistical perspective on algorithmic leveraging. In *International Conference on Machine Learning*, pages 91–99, 2014.

[70] Stephen Marsland. *Machine learning: an algorithmic perspective*. CRC press, 2015.

[71] Vivien Marx. Biology: The big challenges of big data. *Nature*, 498(7453):255–260, 2013.

[72] Eduardo Massad, Neli Regina Siqueira Ortega, Laecio Carvalho de Barros, and Claudio J Struchiner. *Fuzzy logic in action: Applications in epidemiology and beyond*, volume 232. Springer Science & Business Media, 2009.

[73] S.D. Mohaghegh. *Shale Analytics: Data-Driven Analytics in Unconventional Resources*. Springer International Publishing, 2017.

[74] Raghava Rao Mukkamala, Abid Hussain, and Ravi Vatrapu. Fuzzy-set based sentiment analysis of big social data. In *Enterprise Distributed Object Computing Conference (EDOC), 2014 IEEE 18th International*, pages 71–80. IEEE, 2014.

[75] G.C. Onwubolu and B.V. Babu. *New Optimization Techniques in Engineering*. Studies in Fuzziness and Soft Computing. Springer Berlin Heidelberg, 2013.

[76] Ian C Parmee. Introduction. In *Evolutionary and Adaptive Computing in Engineering Design*, pages 1–15. Springer, 2001.

[77] Hanchuan Peng, Fuhui Long, and Chris Ding. Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy. *IEEE Transactions on pattern analysis and machine intelligence*, 27(8):1226–1238, 2005.

[78] Nick Pentreath. *Machine Learning with Spark*. Packt Publishing Ltd, 2015.

[79] Iván Pérez and Betzabeth León. Lógica difusa para principiantes. *Teoría y práctica. Caracas, Venezuela: UCAB*, 2007.

[80] Petra Perner. *Machine Learning and Data Mining in Pattern Recognition: 9th International Conference, MLDM 2013, New York, NY, USA, July 19-25, 2013, Proceedings*, volume 7988. Springer, 2013.

[81] Yuan Alan Qi, Thomas P Minka, Rosalind W Picard, and Zoubin Ghahramani. Predictive automatic relevance determination by expectation propagation. In *Proceedings of the twenty-first international conference on Machine learning*, page 85. ACM, 2004.

[82] Gunnar Rätsch, Takashi Onoda, and K-R Müller. Soft margins for adaboost. *Machine learning*, 42(3):287–320, 2001.

[83] Scott P Robertson, Ravi K Vatrapu, and Richard Medina. Off the wall political discourse: Facebook use in the 2008 us presidential election. *Information Polity*, 15(1, 2):11–31, 2010.

[84] K Roebuck. Data mining: High-impact strategies-what you need to know: Definitions, adoptions, impact. *Benefits, Maturity, Vendors, Emereo Publishing, Brisbane*, 2012.

[85] Alejandro Rosales-Pérez, Jesus A Gonzalez, Carlos A Coello Coello, Hugo Jair Escalante, and Carlos A Reyes-Garcia. Multi-objective model type selection. *Neurocomputing*, 146:83–94, 2014.

[86] Alejandro Rosales-Pérez, Jesus A Gonzalez, Carlos A Coello Coello, Hugo Jair Escalante, and Carlos A Reyes-Garcia. Surrogate-assisted multi-objective model selection for support vector machines. *Neurocomputing*, 150:163–172, 2015.

[87] Alejandro Rosales-Pérez, Carlos A Reyes-García, Jesus A Gonzalez, Orion F Reyes-Galaviz, Hugo Jair Escalante, and Silvia Orlandi. Classifying infant cry patterns by the genetic selection of a fuzzy model. *Biomedical Signal Processing and Control*, 17:38–46, 2015.

[88] L. Saitta and J.D. Zucker. *Abstraction in Artificial Intelligence and Complex Systems*. SpringerLink : Bücher. Springer New York, 2013.

[89] Sherif Sakr, Anna Liu, and Ayman G Fayoumi. The family of mapreduce and large-scale data processing systems. *ACM Computing Surveys (CSUR)*, 46(1):11, 2013.

[90] AF Salam. *Semantic Web Technologies and E-Business: Toward the Integrated Virtual Organization and Business Process Automation: Toward the Integrated Virtual Organization and Business Process Automation*. IGI Global, 2006.

[91] Javier Sánchez-Monedero, Pedro Antonio Gutiérrez, María Pérez-Ortiz, and César Hervás-Martínez. An n-spheres based synthetic data generator for supervised classification. In *International Work-Conference on Artificial Neural Networks*, pages 613–621. Springer, 2013.

[92] T. Schaul and J. Schmidhuber. Metalearning. *Scholarpedia*, 5(6):4650, 2010. revision #91489.

[93] Jiri Schindler. I/o characteristics of nosql databases. *Proceedings of the VLDB Endowment*, 5(12):2020–2021, 2012.

[94] Marc Seeger and S Ultra-Large-Sites. Key-value stores: a practical overview. *Computer Science and Media, Stuttgart*, 2009.

[95] Jeffrey Shafer, Scott Rixner, and Alan L Cox. The hadoop distributed filesystem: Balancing portability and performance. In *Performance Analysis of Systems & Software (ISPASS), 2010 IEEE International Symposium on*, pages 122–133. IEEE, 2010.

[96] Yun Q Shi. *Transactions on data hiding and multimedia security III*, volume 4920. Springer, 2008.

[97] Dilpreet Singh and Chandan K Reddy. A survey on platforms for big data analytics. *Journal of Big Data*, 2(1):8, 2015.

[98] Evan R Sparks, Ameet Talwalkar, Daniel Haas, Michael J Franklin, Michael I Jordan, and Tim Kraska. Automating model search for large scale machine learning. In *Proceedings of the Sixth ACM Symposium on Cloud Computing*, pages 368–380. ACM, 2015.

[99] Evan R Sparks, Ameet Talwalkar, Virginia Smith, Jey Kottalam, Xinghao Pan, Joseph Gonzalez, Michael J Franklin, Michael I Jordan, and Tim Kraska. Mli: An api for distributed machine learning. In *Data Mining (ICDM), 2013 IEEE 13th International Conference on*, pages 1187–1192. IEEE, 2013.

[100] Shan Suthaharan. Big data classification: Problems and challenges in network intrusion prediction with machine learning. *ACM SIGMETRICS Performance Evaluation Review*, 41(4):70–73, 2014.

[101] EG Talbi. Hybrid metaheuristics, vol. 434 of studies in computational intelligence, 2013.

[102] David Taniar, Clement HC Leung, Wenny Rahayu, and Sushant Goel. *High performance parallel database processing and grid databases*, volume 67. John Wiley & Sons, 2008.

[103] Barnabas K Tannahill, Chris E Maute, Yunus Yetis, Maryam N Ezell, Aldo Jaimes, Roberto Rosas, Azima Motaghi, H Kaplan, and Mo Jamshidi. Modeling of system of systems via data analytics—case for "big data" in sos. In *System of Systems Engineering (SoSE), 2013 8th International Conference on*, pages 177–183. IEEE, 2013.

[104] Chris Thornton, Frank Hutter, Holger H Hoos, and Kevin Leyton-Brown. Auto-weka: Combined selection and hyperparameter optimization of classification algorithms. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 847–855. ACM, 2013.

[105] Mania Tlili and Tarek M Hamdani. Big data clustering validity. In *Soft Computing and Pattern Recognition (SoCPaR), 2014 6th International Conference of*, pages 348–352. IEEE, 2014.

[106] LeiChen PeterTriantafillou TorstenSuel. Web information systems engineering–wise 2010.

[107] Isaac Triguero, Daniel Peralta, Jaume Bacardit, Salvador García, and Francisco Herrera. Mrpr: a mapreduce solution for prototype reduction in big data classification. *neurocomputing*, 150:331–345, 2015.

[108] Muhammet Ünal, Ayça Ak, Vedat Topuz, and Hasan Erdal. *Optimization of PID controllers using ant colony and genetic algorithms*, volume 449. Springer, 2012.

[109] José María Valencia-Ramírez, Julio A Raya, José R Cedeño, Ranyart R Suárez, Hugo Jair Escalante, and Mario Graff. Comparison between genetic programming and full model selection on classification problems. In *Power, Electronics and Computing (ROPEC), 2014 IEEE International Autumn Meeting on*, pages 1–6. IEEE, 2014.

[110] Vladimir Vapnik, Esther Levin, and Yann Le Cun. Measuring the vc-dimension of a learning machine. *Neural computation*, 6(5):851–876, 1994.

[111] Panos Vassiliadis, Alkis Simitsis, and Spiros Skiadopoulos. Conceptual modeling for etl processes. In *Proceedings of the 5th ACM international workshop on Data Warehousing and OLAP*, pages 14–21. ACM, 2002.

[112] Peter Verplaetse, Jan Van Campenhout, and Dirk Stroobandt. On synthetic benchmark generation methods. In *Circuits and Systems, 2000. Proceedings. ISCAS 2000 Geneva. The 2000 IEEE International Symposium on*, volume 4, pages 213–216. IEEE, 2000.

[113] Akhil Wali. *Clojure for Machine Learning*. Packt Publishing Ltd, 2014.

[114] Tom White. *Hadoop: The definitive guide*. " O'Reilly Media, Inc.", 2012.

[115] David H Wolpert. The lack of a priori distinctions between learning algorithms. *Neural computation*, 8(7):1341–1390, 1996.

[116] Xindong Wu, Xingquan Zhu, Gong-Qing Wu, and Wei Ding. Data mining with big data. *IEEE transactions on knowledge and data engineering*, 26(1):97–107, 2014.

[117] Junchang Xin, Zhiqiong Wang, Luxuan Qu, and Guoren Wang. Elastic extreme learning machine for big data classification. *Neurocomputing*, 149:464–471, 2015.

[118] Waleed A Yousef and Subrata Kundu. Learning algorithms may perform worse with increasing training set size: Algorithm–data incompatibility. *Computational Statistics & Data Analysis*, 74:181–197, 2014.

[119] Jianfeng Zhan, Rui Han, and Chuliang Weng. Big data benchmarks, performance optimization, and emerging hardware. In *4th and 5th Workshops, BPOE 2014*. Springer, 2014.

[120] Huai-xiang Zhang, Bo Zhang, and Feng Wang. Automatic fuzzy rules generation using fuzzy genetic algorithm. In *Fuzzy Systems and Knowledge Discovery, 2009. FSKD'09. Sixth International Conference on*, volume 6, pages 107–112. IEEE, 2009.

[121] Huai-xiang Zhang, Bo Zhang, and Feng Wang. Automatic fuzzy rules generation using fuzzy genetic algorithm. In *Fuzzy Systems and Knowledge Discovery, 2009. FSKD'09. Sixth International Conference on*, volume 6, pages 107–112. IEEE, 2009.

[122] Xuguang Zhang, Shuo Hu, Dan Chen, and Xiaoli Li. Fast covariance matching with fuzzy genetic algorithm. *IEEE Transactions on Industrial Informatics*, 8(1):148–157, 2012.