Universidad de las Ciencias Informáticas

Facultad No. 2

Implementación de algoritmos basados en aprendizaje de funciones de distancia para problemas de clasificación en la herramienta WEKA

Trabajo de Diploma

Autores: Gabriela Santos Martínez

Frank Rubén Campos Almarales

Tutor: Msc. Héctor Raúl González Diez

Ciudad de La Habana, 19 de junio del 2015

“La razón más importante para trabajar en la escuela y en la vida es el placer de trabajar, el placer de su resultado y el conocimiento del valor del resultado para la comunidad.”

Albert Einstein

Declaración de autoría

Declaramos ser autores de la presente tesis y reconocemos a la UCI los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmo la presente a los ____ días del mes de _____ del año _____.

_____ _____

Gabriela Santos Martínez Frank Rubén Campos Almarales

_____

Héctor Raúl González Diez

Agradecimientos

A mi mamá, por enseñarme todo lo que sé, por ser lo mejor de mi vida, por ser una madre incondicional.

A mi papá, por ser mi héroe, la roca que me sostiene en los momentos que más lo necesito.

A mi hermano porque cada día me enseña algo nuevo y por ser la fuente de mi inspiración.

A Frank, por ayudarme a llegar a donde estoy ahora y por ser mi más grande amor.

A mis familiares y amigos que de una forma u otra han contribuido en la formación de la persona que soy hoy.

A mi tutor, que nos ha ayudado más allá de lo que puedan expresar las palabras.

A los profesores y especialistas que nos han ayudado a formarnos como profesionales en el transcurso de la carrera

A todos,

¡Gracias!

Gabriela

Agradecimientos

A mi mamá, porque a ella le debo todo lo que soy y por estar a mi lado en todo momento.

A mi abuela, mi segunda madre, por sus consejos y brindarme su apoyo toda mi vida.

A mi hermano, por ser mi compañero de juegos y lo que más quiero.

A mi abuelo, mi papá y mi tío por ser los hombres en los que quisiera convertirme algún día.

A Gaby, por ser tan especial para mí y por haberme soportado todos estos años.

A mis familiares y amigos que de una forma u otra han contribuido en la formación de la persona que soy hoy.

A mi tutor, que se ha convertido en mi mejor amigo y un ejemplo a seguir.

A los profesores y especialistas que nos han ayudado a formarnos como profesionales en el transcurso de la carrera

A todos,

¡Gracias!

Frank

Dedicatoria

A mi mamá por ser mi todo y gracias a ella hoy estoy aquí.

A mi papá por impulsarme cada día a aprender más y porque lo amo mucho.

A mi abuela Nana, que aunque no esté conmigo siempre la tengo presente.

A Frank, por quererme tanto y ayudarme cada vez que lo he necesitado

Gabriela

A mi mamá y mi abuela por ser mis más grandes tesoros.

A mi abuelo y mi papá por su apoyo incondicional.

A mi hermano que me recuerda tanto a mí cuando tenía su edad.

A Gaby, por haber estado a mi lado en los momentos más difíciles de mi vida.

Frank.

Resumen

El aprendizaje de funciones de distancia, no es más que, aprender una función de distancia que transforme el espacio de entrada de los datos, de manera que los elementos de la misma clase se encuentren lo más cerca posible y los elementos de clases diferentes se encuentren lo más lejos posible. En este trabajo se muestra un estudio de los algoritmos de aprendizaje de funciones distancia para problemas de clasificación. Además se describe el proceso de implementación de los algoritmos más representativos del enfoque supervisado LMNN e ITML, y su integración a la herramienta WEKA, así como su evaluación utilizando pruebas estadísticas. Por último se introduce como resultado de esta investigación la extensión teórica del LMNN e ITML para problemas de predicción con salidas compuestas. Los resultados de la evaluación de los algoritmos utilizando la Prueba de Friedman y el *post-hoc* de Bonferroni-Dunn, muestran que existen diferencias significativas entre el LMNN-CV con el algoritmo base de WEKA, kNN. Con relación al *accuracy*, el algoritmo que mostró mejores resultados fue el LMNN-CV en 8 de 15 bases de datos.

Palabras clave: Aprendizaje de funciones de distancia, kNN, clasificación supervisada, LMNN, ITML.

Summary

Distance metric learning, is nothing more than, to learn a distance function that transforms the data input space, in a way that objects of the same class stay closer and the objects of different classes stay as far as possible. This paper presents a study of the existing distance metric learning algorithms for classification problems. Also it is described the implementation process of the most representative algorithms LMNN and ITML from the supervised approach and their integration to WEKA, also their evaluation using statistical processes . Finally the theoretical extensions of these algorithms to multi-target prediction problems. The results of this evaluation using the Friedman test and the post-hoc test of Bonferroni-dunn, showed that there are significant differences between the LMNN-CV and the basic WEKA algorithm kNN. Regarding accuracy, the algorithm that showed the best result was the LMNN-CV on 8 out of 15 datasets.

Keywords: distance metrics learning, kNN, supervised classification, LMNN, ITML.

Introducción

El Aprendizaje Automático (AA) es una rama de la Inteligencia Artificial que abarca un conjunto de técnicas que permiten a una máquina o computadora “aprender” de forma automática (1). El objetivo de este proceso es utilizar información conocida en aras de elaborar una hipótesis y dar solución a nuevos problemas o mejorar la solución de problemas anteriores (2).

Existen muchas aplicaciones en la actualidad para el AA. Ejemplo de ello son las empresas que se dedican a construir objetos (mesas, espejos, piezas, etc.) de forma automatizada donde existe la necesidad de predecir cuándo una determinada maquinaria pudiera fallar mediante el análisis de las lecturas del sensor. Existen expertos que pueden resolver problemas complejos pero que son incapaces de explicar su experiencia. También surge la necesidad de resolver problemas donde los sucesos que los generan cambian constantemente de modo que si existiera algún buen programa que predijera su comportamiento debería ser reescrito frecuentemente. Se encuentran además aplicaciones que necesitan ser ajustadas a un caso específico, o sea, a las características de cada usuario individualmente como es el caso de establecer un filtro para los correos electrónicos no deseados donde cada usuario necesita filtros diferentes y no pueden programar sus reglas. Un sistema de AA puede aprender cuáles correos electrónicos rechaza el usuario y mantener las reglas de filtrado automáticamente (3).

Otras aplicaciones del AA incluyen los Motores de Búsqueda (con el objetivo de mejorar el funcionamiento de las búsquedas basados en la experiencia de los usuarios.), Diagnósticos Médicos (asisten a médicos en el diagnóstico según la historia clínica y síntomas del paciente), Ciencias Biológicas (muy útil en el reconocimiento de tumores, arritmias o patrones en cadenas de ADN), Procesamiento del Lenguaje Natural (en tareas de Extracción de Información, Clasificación Automática de Documentos, Agrupamiento Semántico, entre otras), Finanzas e Industria bancaria (se utilizan modelos de riesgo crediticio que asistan en la calificación de los solicitantes según su nivel de cumplimiento esperado en el pago de las cuotas del crédito, así como modelos de fraude de consumo de tarjetas de crédito y de predicción de comportamiento en el mercado de valores.), Análisis de imágenes (Para reconocer escritura manuscrita y tipos de objetos dentro de una imagen), Juegos, Robótica y Sistemas de Recomendación (en tareas como Extracción de Información, Clasificación Automática de Documentos, Agrupamiento Semántico, Análisis de Sentimiento), por mencionar algunas (4).

Según (3) las tareas del AA pueden ser clasificadas de forma general en dos importantes dimensiones: el Aprendizaje Empírico y el Aprendizaje Analítico. La primera se refiere al tipo de aprendizaje que hace uso de experiencia externa mientras que la segunda no requiere de dichas entradas. La línea divisora entre ambas categorías puede ser algo borrosa.

En la categoría de Aprendizaje Empírico se encuentran contenidas las siguientes tareas:

Aprendizaje supervisado para la clasificación y regresión

Dada una foto de una persona se requiere determinar si es hombre o mujer, esta tarea sería la clasificación. Ahora si también se quiere predecir qué edad tiene, peso y altura entonces se refiere a la regresión. Se identifican en esta categoría los Árboles de Decisión, el Intercambio Triple y el Conocimiento Previo y uso del Bias.

Aprendizaje Supervisado para Secuencias, Series de Tiempo y Datos Espaciales

En este tipo de tarea se pueden enmarcar las funcionalidades de Reconocimiento de Discursos, Predicción del comportamiento de fenómenos (Ejemplo El Niño) y predicción del tipo de cubrimiento de la tierra (árboles, hierba, ríos, etc.).

Aprendizaje no Supervisado

Se identifican cuatro funcionalidades fundamentales.

1. Entendimiento y visualización: dado un conjunto de objetos $\{O_1, \dots, O_n\}$ que, se puede imaginar, constituyen una muestra aleatoria de alguna probabilidad de distribución $P(O)$ subyacente. El proceso de estimación de densidad constituye en aprender la definición de la función de densidad de probabilidad. Este enfoque es muy utilizado para la detección de transacciones de tarjetas de crédito fraudulentas.
2. Completamiento de objetos: involucra la predicción de las partes ausentes de un objeto dada su descripción parcial. Tanto el agrupamiento como la estimación de densidad pueden ser aplicados para realizar esta tarea.
3. Recuperación de información: recuperar objetos relevantes (documentos, imágenes, audio, etc.) de una larga colección de objetos.
4. Compresión de información: identificación y eliminación de los aspectos considerados como irrelevantes de la información.

Aprendizaje para la Toma de Decisiones Secuenciales

Esto quiere decir que las decisiones que se tomen tendrán consecuencia en las acciones futuras. Muchas de las tareas de este tipo surgen en varios dominios donde está presente la necesidad de controlar el sistema (movimientos de un robot, vehículos espaciales, plantas químicas, tratamiento de pacientes en cuidado intensivo).

Por otra parte, en el Aprendizaje Analítico, no se interactúa con la información externa por lo que no puede aprender conocimiento con nuevos contenidos empíricos por lo que se centra en mejorar la velocidad y confiabilidad de las deducciones y decisiones tomadas por el sistema. La tarea que concuerda con esta descripción es el Aprendizaje Acelerado.

Dentro del área de conocimiento y aplicación del AA, se encuentra un tema que ha generado sin número de investigaciones, experimentos y publicaciones: el Aprendizaje de Funciones de Distancia. El aprendizaje de una función de distancia no es más que aprender una transformación del espacio de entrada de los datos de manera que los elementos de la misma clase se encuentren cada vez más cerca y los elementos de clases diferentes lo más lejos posible (5). En estas investigaciones se llega a la conclusión que este enfoque mejora significativamente el poder predictivo de los clasificadores.

Existen un conjunto de herramientas con técnicas y algoritmos de aprendizaje automático donde su uso se ha generalizado por la comunidad internacional, entre las cuales se encuentran las herramientas Matlab, R, KEEL y WEKA.

MATLAB es un entorno de computación y desarrollo de aplicaciones totalmente integrado orientado para llevar a cabo proyectos en donde se encuentren implicados elevados cálculos matemáticos y la visualización gráfica de los mismos. Las desventajas de MATLAB son las siguientes: es privativa, no es de código abierto y no se integra a aplicaciones empresariales (6). R es un software de uso gratuito, para análisis estadístico y gráfico que facilita una amplia gama de técnicas y que se desarrolla a través de paquetes que aportan algunos de sus usuarios para uso de todos. No es fácil de utilizar para un usuario sin conocimientos básicos de programación. Se necesita un conocimiento amplio de estadística para operarlo. Existen programas menos complejos de utilizar, con un producto final muy similar al programa R. No tiene la capacidad de interactuar con Excel (7; 8). KEEL es una herramienta de código abierto, escrita en Java que puede ser usada para distintas tareas de descubrimiento de conocimiento, además asiste en tareas de regresión, clasificación, agrupamiento y minería de patrones. Esta herramienta es una de las menos utilizadas por la comunidad internacional, las estructuras de las bases de datos que se utilizan en ella son complejas, está

generalmente enfocada a los algoritmos evolucionales y en sus últimas versiones todavía no está totalmente estable. Además no es una herramienta de propósito general (9).

En el caso de WEKA es una herramienta especializada para aprendizaje automático desarrollada en el departamento de Ciencias de la Computación de la Universidad de Waikato en Nueva Zelanda. WEKA es una herramienta de código abierto desarrollado sobre tecnología JAVA que funciona en plataforma Windows y Linux. En cuanto a los algoritmos de aprendizaje automático que soporta WEKA esencialmente cuenta con un conjunto de herramientas de pre-procesamiento de datos, alrededor de 76 algoritmos de clasificación/regresión para resolver problemas de clasificación supervisadas, 8 algoritmos de agrupamiento y 3 algoritmos de reglas de asociación para problemas no supervisados. De forma general podemos afirmar que WEKA no cuenta con una implementación de un enfoque de aprendizaje de funciones de distancia que permita a los investigadores que utilicen esta herramienta poder utilizar este enfoque en diferentes experimentos, ya que ha quedado demostrado que mejora significativamente el poder predictivo de la herramienta.

En la actualidad, las aplicaciones existentes relacionadas con el aprendizaje automático, requieren de métodos no convencionales que den solución a problemas cada vez más complejos donde los resultados puedan modelarse de maneras distintas en dependencia del tipo de problema. En los problemas de predicción del mundo real a menudo involucran la predicción simultánea de varias salidas utilizando el mismo conjunto de variables predictoras. A este tipo de problemas se le conoce como problemas de predicción con salidas compuestas.

En este tipo de problemas en vez de obtener una salida simple, se necesitan predecir un conjunto de estas de manera simultánea. Es decir, todas las salidas son igualmente importantes por lo que se predicen simultáneamente con un solo modelo (10).

Existen muchas e interesantes aplicaciones para los problemas de predicción con salidas compuestas, mayormente industriales. Ejemplo de algunas de ellas es reproducir el comportamiento de un operador humano para reducir el tiempo de uso de la maquinaria en una central eléctrica. Es útil además para predecir cuántas veces un tipo de llama solar (normal, moderada, severa) es observada en 24 horas (para propósitos científicos). Otra aplicación es determinar la presencia de animales, plantas, químicos que influyen en la calidad del agua. Finalmente para estimar el número de trabajadores a tiempo completo existentes en determinada área poblacional según el tipo de empleo. Otras de las aplicaciones de este tipo de problemas serían en la Visión por Computadora en las tareas de clasificación y recuperación de imágenes, recuperación de información, bioinformática y minería web, entre otras.

De manera resumida podemos plantear:

- ✓ Los algoritmos basados en distancias, siguen estando considerados como los más efectivos en problemas de clasificación o predicción (k vecinos más cercanos (kNN), K-mean, etc.).
- ✓ El Aprendizaje de funciones de distancia realiza una transformación del espacio de entrada de los datos que mejora el poder predictivo del kNN.
- ✓ En los algoritmos de predicción con salidas compuestas no se ha explotado el uso de kNN (solo kNN-SP, que no hace un análisis estadístico profundo de las configuraciones del kNN para obtener mejores resultados).
- ✓ WEKA no cuenta con una implementación de ningún algoritmo basado en aprendizaje de funciones de distancia.

La problemática antes descrita ha generado la necesidad de desarrollar una investigación que dé respuesta al siguiente **problema a resolver**: ¿Cómo mejorar la clasificación en problemas que utilicen algoritmos basados en distancia en la herramienta WEKA?. Teniendo en cuenta el problema antes propuesto se define como **objeto de estudio**: Algoritmos basados en aprendizaje de funciones de distancia.

Determinándose como **objetivo general**: Implementar, como extensión de WEKA algoritmos de aprendizaje de funciones de distancia para la resolución de problemas de clasificación. Para dar cumplimiento al objetivo general se plantean los siguientes **objetivos específicos**:

1. Caracterizar el marco teórico-conceptual de los algoritmos de aprendizaje de distancia para problemas de predicción o clasificación.
2. Implementar un conjunto de algoritmos de aprendizaje de distancia para problemas de predicción o clasificación integrados a la herramienta WEKA.
3. Validar la solución implementada mediante experimentos para evaluar el desempeño de estos algoritmos y compararlos con los algoritmos del estado del arte.
4. Desarrollar desde el punto de vista teórico una propuesta de extensión de los algoritmos de aprendizaje de funciones de distancia para problemas de predicción con salidas compuestas.

Enmarcándose estos en el **campo de acción**: Algoritmos de aprendizaje de funciones de distancia para problemas de clasificación basados en distancia.

Para la solución del problema planteado se utilizaron los métodos teóricos:

- **Analítico-Sintético**: se utilizó para el estudio y síntesis de los diferentes algoritmos de aprendizaje de funciones de distancia, para problemas de predicción de salidas compuestas y un estudio general de las métricas y metodologías para la evaluación de los clasificadores.
- **Inductivo-Deductivo**: para llegar a conclusiones sobre qué algoritmos de aprendizaje de funciones de distancia implementar como parte de la herramienta WEKA y para la extensión para problemas de predicción con salidas compuestas.
- **Modelación**: para la propuesta teórica del modelo que extiende el aprendizaje de funciones de distancia a problemas de predicción con salidas compuestas; en el diseño de los diagramas de clases, diseño de la implementación, modelación de la arquitectura y la integración dentro de la herramienta WEKA.

También se hizo uso de los métodos empíricos:

- **Medición**: durante la experimentación se utilizaron las métricas para la evaluación de los algoritmos para problemas de clasificación basados en aprendizaje de funciones de distancia y la Prueba de Friedman, con el post-hoc de Bonferroni-Dunn para comparar estos algoritmos con el algoritmo kNN natural de WEKA y sus variantes.

El desarrollo de la investigación trae consigo diversos beneficios. En primer lugar WEKA cuenta con una extensa comunidad internacional de desarrolladores e investigadores que hacen uso de ella; es de código abierto, o sea, se puede acceder y modificar su código; brinda facilidades para la realización de investigaciones y experimentos en el ámbito del AA. Además es la herramienta que se utiliza en la asignatura de Inteligencia Artificial, de la Universidad de las Ciencias Informáticas de Cuba, para la impartición de la docencia, donde uno de los temas de esa asignatura es precisamente el aprendizaje automático. También se desarrollan diferentes actividades tanto teóricas como prácticas. En estas últimas se experimenta con algoritmos de aprendizaje automático. El resultado de la investigación puede ser parte de una metodología de trabajo ya que se describe paso a paso el proceso de evaluación de los algoritmos implementados. Uno de los temas que se podría incluir en la asignatura sería el kNN y el modelado de estos algoritmos que utilizan el enfoque basado en aprendizaje de funciones de distancia. Compartir los resultados de la investigación con la comunidad internacional ayudaría a que investigadores de todo el mundo utilicen esta herramienta, ya modificada, en la construcción de nuevos algoritmos relacionados con la temática.

En aras de estructurar el proceso de desarrollo del presente trabajo de una manera coherente y organizada, se dividió el mismo en un total de 3 capítulos:

Capítulo 1. *Fundamentación teórica de los elementos relacionados con el Aprendizaje de funciones de distancia*: se abordan los elementos teóricos que sirven de base para la realización de la investigación, es decir, se realiza un estado del arte de los algoritmos basados en el aprendizaje de funciones de distancia, los algoritmos para problemas de predicción con salidas compuestas, las herramientas existentes, la metodología estadística de evaluación de clasificadores a utilizar para evaluar los algoritmos basados en aprendizaje de distancia.

Capítulo 2. *Propuesta de Solución a los problemas de clasificación basados en el Aprendizaje de funciones de distancia*: se describe la propuesta de los algoritmos a implementar, tanto el LMNN como el ITML, además se modela la propuesta de la extensión de estos algoritmos para el trabajo con problemas de predicción con salidas compuestas desde el punto de vista teórico. También se muestra el proceso de integración de estos algoritmos en la herramienta WEKA.

Capítulo 3. *Experimentos basados en la aplicación de los algoritmos implementados y resultados de la Prueba de Friedman*: se diseñan un conjunto de experimentos a través de casos de estudio y la evaluación estadística de los algoritmos implementados comparados con el algoritmo kNN natural de WEKA y sus variantes.

Contenido

Capítulo 1

Fundamentación teórica de los elementos relacionados con el Aprendizaje de Funciones de Distancia

1.1. Introducción

El objetivo del aprendizaje de funciones de distancia es, aprender una función de valor real, usando la información obtenida de los ejemplos de entrenamiento. Esta función debe transformar el espacio de entrada de los datos, para que los elementos de clases similares se acerquen lo más posible y los elementos de clases diferentes se encuentren lo más lejos posible. Es necesario además que sea la que mejor satisfaga las restricciones existentes (11).

Esta función debe cumplir las siguientes propiedades:

1. $D(x_i, x_j) + D(x_j, x_k) \geq D(x_i, x_k) \Rightarrow desigualdad\ triangular$
2. $D(x_i, x_j) \geq 0 \Rightarrow no\ negatividad$
3. $D(x_i, x_j) = D(x_j, x_i) \Rightarrow simetría$
4. $D(x_i, x_j) = 0 \Leftrightarrow x_i = x_j \Rightarrow distinguibilidad$ (12)

A una función que satisface las tres primeras propiedades pero no la cuarta, se denomina *seudométrica*. Se pueden identificar las siguientes familias de funciones: lineales, no lineales y locales. Aunque el poder de las lineales es limitado, son más fáciles de modelar a través de un problema de optimización y menos propensas al sobre-entrenamiento. Las no lineales a su vez, conducen a formulaciones no convexas y pueden determinar variaciones no lineales en los datos. Por último las locales operan mejor para problemas que dependen de una cantidad grande de parámetros. Una deficiencia de las dos últimas es que poseen problemas con el sobre-entrenamiento (11).

Los algoritmos basados en el Aprendizaje de funciones de distancia tratan de resolver un problema de optimización con restricciones, cuyo modelo se puede expresar como:

$$\arg\min_{M \pm 0} L(M) = \lambda R(M) + \sum_{i=1}^{m} l_i(M, R_i)$$

Donde R es una función de regularización que depende de M. Esta función previene que el sistema aprenda un modelo sin capacidad de generalización, la función $l_i(M, R_i)$ es la función de pérdida que penaliza la violación de la restricción R_i y λ es el parámetro de regularización (12).

Estas restricciones R_i se pueden presentar en pares o en tripletas:

- Restricciones en pares (*positivo/negativo*)

$$S = \{(x_i, x_j): deben\ ser\ similares\}$$

$$D = \{(x_i, x_j): deben\ ser\ diferentes\}$$

- Tripletas

$$R = \{(x_i, x_j, x_k): x_i\ debe\ ser\ más\ similar\ a\ x_j\ que\ a\ x_k\}$$

La función aprendida podría ser la función de distancia Euclídea, que es utilizada cuando los datos son de tipo numérico. Para determinar esta distancia entre dos instancias del espacio de entrada de los datos, se calcula:

$$d(x_i, x_j) = \sqrt{(x_i - x_j)^T (x_i - x_j)}$$

Un enfoque más general a esta función de distancia, es haciendo uso de pesos. Es decir:

$$d(x_i, x_j) = \sqrt{\sum_{k=1}^{p} w_k (x_i^k - x_j^k)^2}$$

En la mayoría de los casos, la función aprendida es una función de distancia de Mahalanobis:

$$D_M(x_i, x_j) = \sqrt{(x_i - x_j)^T M (x_i - x_j)}$$

entre dos instancias $x_i, x_j \in \mathcal{X}$, y sus clases correspondientes $y_i, y_j \in [0, m]$, usando la información de la clase de los datos del conjunto de entrenamiento. Esta distancia se puede considerar como una generalización de la distancia Euclidiana. En particular, las distancias Euclidianas se recuperan haciendo que M sea igual a la matriz identidad.

Luego del aprendizaje de la métrica la función resultado debe mejorar el funcionamiento de un algoritmo basado en ella.

1.2. Categorías del Aprendizaje de Funciones de Distancia

Tanto para el aprendizaje automático como para el reconocimiento de patrones, el aprendizaje de funciones de distancia ha sido un problema fundamental en el cual la comunidad científica ha logrado avances significativos. Teniendo esto en cuenta, se han sintetizado las cuatro categorías fundamentales.

La primera está referida a los enfoques de aprendizaje de funciones de distancia basados en un margen amplio. Esta categoría es muy útil para resolver el problema centralizado de incremento

del margen. Otra categoría son los métodos de Kernel hacia el aprendizaje de funciones de distancia, donde una buena selección del kernel puede lograr buen desempeño en las atreas de clasificación (5).

El Aprendizaje de Funciones de Distancia Supervisado y el No Supervisado son las categorías donde se enmarcan la mayoría de los algoritmos. La primera intenta aprender las funciones de distancia utilizando información complementaria como la información de las clases y las restricciones asociadas. La segunda no tiene en cuenta esta información (13) (14).

1.3. Métodos de Aprendizaje de Funciones de Distancia

Según se ha avanzado en las investigaciones relacionadas con el tema, se han ido implementando diversos algoritmos que pueden pertenecer a una de las categorías antes descritas.

Los métodos de métricas de distancia basados en el vecino más cercano de margen amplio y los de Programación Semi- Definida (SDP) (15) están contenidos dentro de los enfoques de aprendizaje de métricas de distancia basados en un margen amplio.

Los métodos de Kernel contienen la alineación del kernel (16) con sus enfoques SDP y el trabajo de extensión del aprendizaje del kernel idealizado (17) (14).

Como las dos últimas categorías antes descritas, Aprendizaje Supervisado y No Supervisado, son las más generales y utilizadas por los autores especializados en el tema, a continuación se presenta un estado del arte de los métodos de aprendizaje de funciones de distancia existentes.

1.3.1. Aprendizaje de Distancia Supervisado

Aprendizaje de Distancia Supervisado Global

En este tipo particular de aprendizaje de distancia, se distinguen los métodos globales, estos aprenden la métrica de distancia en un sentido global, o sea, para satisfacer todas las restricciones en pares simultáneamente. Se pueden identificar como globales el Aprendizaje de Métricas de Distancia Global (GDML) (18), el Análisis de los Componentes Relevantes (RCA), el Análisis de los Componentes Discriminantes (DCA) (19), el Kernel DCA, el Aprendizaje de Métricas de Distancia Activo Bayesiano (BADML) (20) y la Métrica de Aprendizaje basada en Teoría de la Información (ITML) (21).

El GDML es un algoritmo que aprende una función de distancia que respeta las relaciones de similaridad entre pares de puntos para un espacio de entrada $\mathbb{R}^n$. Este método se modela como un problema de optimización convexa y se encarga de minimizar la distancia entre los objetos de la misma clase, bajo la restricción de que los objetos de clases diferentes deben estar bien separados.

Teniendo en cuenta un conjunto de puntos $\{x_i\}_{i=1}^{m} \subseteq \mathbb{R}^n$ y conociendo que algunos pares de ellos son "similares": S: $(x_i, x_j) \in S$. La función $d(x, y)$ logra acercar estos objetos similares cada vez más entre sí y se expresa de la forma:

$$d(x, y) = d_A(x, y) = \|x - y\|_A = \sqrt{(x - y)^T A(x - y)}$$

donde se tiene que cumplir que $A \succcurlyeq 0$ (o sea que A sea semidefinida positiva). Como caso particular para $A = I$ se obtiene la distancia Euclideana; si se restringe A para que sea diagonal se parametriza una función de distancia de Mahalanobis sobre $\mathbb{R}^n$.

El GDML puede aprender funciones diagonales o completas; para algunos problemas puede aprenderlas rápidamente con muy poca información complementaria, para otros la función de distancia (particularmente la completa) es más difícil de aprender. Es incapaz de estimar la probabilidad de que cualquier instancia de datos comparta la misma clase. La extensión de este algoritmo por parte de (17) condujo a un caso no lineal por la introducción de kernels.

Por su parte RCA busca minimizar la variabilidad global no deseada dentro de los datos. A través de una transformación lineal global, este algoritmo cambia el espacio de características usado para la representación de los datos. Este proceso va asignando pesos grandes a las "dimensiones relevantes" y pesos pequeños a las "dimensiones no relevantes" (22). Aunque brinda muchas ventajas, RCA no está exento de desventajas: la primera es la falta de inclusión de las restricciones negativas; y la segunda es que solo puede aprender la relación lineal existente entre las distintas instancias de datos. Esto es demasiado restringido e impide descubrir las relaciones no lineales para muchas aplicaciones (19).

DCA y Kernel DCA mejoran RCA al explorar restricciones negativas apuntando a capturar estas relaciones, utilizando información contextual.

El funcionamiento de DCA está basado en aprender una transformación óptima de los datos que conduzca a una métrica de distancia óptima. Esto lo logra al maximizar la varianza total entre los "chunklets" de datos discriminativos y al minimizar la varianza total de las instancias de datos en los mismos "chunklets" (19). Según (23) un chunklet no es más que un subconjunto de puntos que, se conoce, pertenecen a la misma clase a pesar de ser esta desconocida y se pueden obtener a partir de las relaciones de equivalencia resultantes de aplicar cierre transitivo. Un inconveniente de DCA es que cuando se trata de descubrir relaciones no lineales entre la información del mundo real no muestra buenos resultados (19).

Para resolver esta deficiencia del DCA se incluye en su funcionamiento el truco del kernel dando lugar al KDCA. Este método al aplicar el truco del kernel permite el aprendizaje de estructuras complejas no lineales de los datos de entrada (25; 26). KDCA primeramente proyecta los datos

de entrada en un espacio de características implícito mediante el truco del kernel. Luego se aplica el DCA al espacio de características proyectado en aras de encontrar la transformación lineal óptima en el espacio de características. Finalmente, se hayan las estructuras no lineales de la información dada usando la técnica KDCA (19).

Otro de los algoritmos mencionados anteriormente es BADML que, siguiendo el principio de incertidumbre para aprendizaje activo, selecciona pares de datos que son informativos para la función de distancia objetivo. Se selecciona el par de puntos de datos que posean la incertidumbre más grande en decidir si estos se encuentran cerca o no. Esta medida de incertidumbre se calcula por la función $H_{i,j}$ para un par x_i, x_j de datos. Este enfoque aplica primeramente el algoritmo de aprendizaje de una función de distancia bayesiano para encontrar una función dados un par de puntos. Luego evalúa la probabilidad $P_r(\pm|x_i, x_j)$ al usar directamente la media μ_γ (todo esto es parte de la fórmula de $H_{i,j}$). Los pares de ejemplo con mayor entropía serán elegidos para el etiquetado manual (20). A continuación se muestra cómo calcular $H_{i,j}$:

$$H_{i,j} = \begin{cases} -P_r(-|x_i, x_j) \log P_r(-|x_i, x_j) \\ -P_r(+|x_i, x_j) \log P_r(+|x_i, x_j) \end{cases} \quad (19)$$

Finalmente en el ITML se busca lograr un equilibrio entre la distribución gaussiana multivariada y el conjunto de distancias de Mahalanobis, transformando así el problema de aprender una función de distancia óptima al aprendizaje de una gaussiana óptima con respecto a una función objetivo basada en la entropía. ITML acomoda un rango de restricciones y relaciones entre pares de distancia, incluyendo además, información previa con respecto a la función de distancia. El funcionamiento de este algoritmo es equivalente a aplicar por primera vez un PCA estándar sobre el espacio de características y luego calcular las distancias usando la función de distancia Euclídea al cuadrado. AL realizarle pruebas, ITML ha demostrado ser eficaz en problemas de agrupamiento y aprende las métricas más rápido que otros algoritmos de aprendizaje (21).

Aprendizaje de Distancia Supervisado Local

Como se ha mencionado anteriormente, en el Aprendizaje de Funciones de Distancia Supervisado se distingue también el enfoque local. Este enfoque utiliza solamente la información de la clase de los objetos de la vecindad para hallar la función de distancia.

El Análisis Discriminante Lineal Local (LDA) (27), el Análisis Discriminante Local de Fisher (LFDA) (28), el Análisis de los Componentes de la Vecindad (NCA) (29), el Clasificador del Vecino Más Cercano de Margen Amplio (LMNN) (30) y el Aprendizaje de Funciones de Distancia

Local (LDM) (31) son otros de los Algoritmos de Aprendizaje de Distancia Supervisado que se basan en este enfoque Local.

Encontrando los vectores propios de la matriz $T = S_w^{-1}S_b$, LDA transforma linealmente el espacio de representación de los atributos, siendo S_w^{-1} y S_b la covarianza entre las clases y la covarianza inter-clase respectivamente (12). En LDA los datos son primeramente colocados en forma de esfera con respecto a S_b, luego la instancia es clasificado según la clase cuyo centroide se encuentre más cerca (con una corrección para las probabilidades previas de aplicación de la clase). Dado que solo las distancias relativas son relevantes, cualquier instancia en el complemento del sub-espacio extendido por los centroides en forma de esfera puede ser ignorada. Este complemento corresponde al espacio nulo de S_w^{-1} (27).

RCA y DCA pueden ser considerados como extensiones de LDA ya que ambos explotan las restricciones “debe conectar” y “no puede conectar”. Adoptando una aproximación de una colección de extensiones para cada paso de la actualización, (32) provee un método eficiente de aprendizaje incremental para LDA. Un enfoque basado en LDA presentado como un problema eficiente de autovalores es uno de los aportes de (33). (21) expresa el aprendizaje de una función de Mahalanobis como un problema de optimización de Bregman, lo que facilita la minimización de la entropía diferencial relativa entre dos gaussianas multivariadas sujetas a restricciones lineales sobre la función de distancia (14).

El LFDA extiende al LDA asignando mayores pesos a los ejemplos conectados que están cerca en lugar de los que están distantes (14), sin embargo no se puede comprender su funcionamiento si no se tiene conocimiento del Análisis del Discriminante de Fisher (FDA) (34; 35), el cual es un método popular para la reducción de la dimensionalidad lineal. FDA maximiza la dispersión entre clases y la minimiza dentro de las clases. Aunque es conocido por trabajar bien tiende a dar resultados no deseados si los ejemplos en algunas clases forman varios clústers separados (34). Este comportamiento es resultado de la globalidad cuando se evalúa la dispersión entre e intra clases. Por otro lado la Proyección Preservadora de la Localidad (LPP) (36) puede superponer ejemplos de clases diferentes si se encuentran muy cerca en el espacio de alta dimensión original. Para dar solución a estos problemas se combinan FDA y LPP para dar origen a LFDA.

LFDA no impone que se acerquen los objetos que pertenecen a la misma clase, preservando así la estructura local de la información. Tomando $S^{(w)}$y $S^{(b)}$ como las matrices de dispersión intra y entre clases, FDA se formula a través de restricciones por pares de la siguiente manera:

$$S^{(w)} = {}^{1}/_{2} \sum_{i,j=1}^{n} A_{i,j}^{(w)} (x_i - x_j)(x_i - x_j)^T$$

$$S^{(b)} = {}^{1}/_{2} \sum_{i,j=1}^{n} A_{i,j}^{(b)} (x_i - x_j)(x_i - x_j)^T$$

donde

$$A_{i,j}^{(w)} = \begin{cases} {}^{1}/_{n_c} & y_i = y_j = c \\ 0 & y_i \neq y_j \end{cases}$$

$$A_{i,j}^{(b)} = \begin{cases} {}^{1}/_{n} - {}^{1}/_{n_c} & y_i = y_j = c \\ {}^{1}/_{n} & y_i \neq y_j \end{cases}$$

Basadas en esta representación se definen $\hat{S}^{(w)}$ y $\hat{S}^{(b)}$ como las matrices de dispersión local intra y entre clases:

$$\hat{S}^{(w)} = {}^{1}/_{2} \sum_{i,j=1}^{n} \hat{A}_{i,j}^{(w)} (x_i - x_j)(x_i - x_j)^T$$

$$\hat{S}^{(b)} = {}^{1}/_{2} \sum_{i,j=1}^{n} \hat{A}_{i,j}^{(b)} (x_i - x_j)(x_i - x_j)^T$$

donde

$$\hat{A}_{i,j}^{(w)} = \begin{cases} {A_{i,j}^{(w)}}/{n_c} & y_i = y_j = c \\ 0 & y_i \neq y_j \end{cases}$$

$$\hat{A}_{i,j}^{(b)} = \begin{cases} A_{i,j}^{(b)} ({}^{1}/_{n} - {}^{1}/_{n_c}) & y_i = y_j = c \\ {}^{1}/_{n} & y_i \neq y_j \end{cases}$$

Comparado con las contrapartes globales $S^{(w)}$y $S^{(b)}$, los valores para los pares de la misma clase son medidos por su afinidad en $\hat{S}^{(w)}$y $\hat{S}^{(b)}$, lo cual significa que tienen menos influencia los pares de la misma clase que se encuentran distantes. Por tanto la matriz de transformación de LFDA, T_{LFDA}, se define como:

$$T_{LFDA} = \underset{T \in \mathbb{R}^{dxm}}{\operatorname{argmax}}\, t_r((T^T \bar{S}^{(w)})^{-1} T^T \bar{S}^{(b)} T) \quad (28)$$

Este algoritmo puede preservar la estructura multimodo de los datos, es decir, la capacidad de los datos de presentarse en modos diferentes de actividad u ocurrencia. Es muy útil en tareas de visualización de datos.

Otro de los algoritmos a describir es NCA, que es un método no paramétrico muy simple y tiene varias propiedades:

- Las áreas de decisión son no lineales.
- La calidad de las predicciones mejoran automáticamente tanto como incrementa la cantidad de datos de entrenamiento
- Solo tiene un hiper-parámetro a ser estimado (29).

Este algoritmo aprende una distancia de Mahalanobis para el clasificador kNN maximizando la validación cruzada *leave-one-out*. Garantizando que la matriz de distancia aprendida sea simétrica y semidefinida positiva (29) asume a $M = L^T L$ donde L puede ser cualquier matriz, por tanto la distancia entre dos instancias de datos u y v será positiva y se calcula:

$$D_M(u, v) = (u - v)^T M(u - v) = (Lu - Lv)^T (Lu - Lv)$$

La probabilidad de clasificar correctamente x_i se expresa por $p_i = \sum_{j \in c_i} p_{i,j}$ y el número de puntos clasificados correctamente es $f(L) = \sum_{i=1}^{n} p_i$. Es necesario aclarar que $c_i = \{j \mid y_i = y_j\}$ representa el conjunto de instancias que comparten la misma clase con x_i y $p_{i,j}$ la probabilidad para seleccionar x_j como el vecino de x_i que comparte su misma etiqueta de clase:

$$p_{i,j} = \frac{\exp(-\|Lx_i - Lx_j\|^2)}{\sum_{k \neq i} \exp(\|Lx_i - Lx_k\|^2)}$$

Por otro lado (29) sugiere el uso de la validación cruzada, *leave-one-out* de la función objetivo $f(L)$:

$$f(L) = \sum_{i=1}^{n} \log(\sum_{j \in c_i} p_{i,j}) \quad (12)$$

Aparte de las facilidades que brinda, NCA tiene como inconveniente que se debe especificar una función en el espacio de entrada para definir lo que se considera como “el más cercano”. Además, para encontrar los vecinos de un objeto antes de la clasificación se debe guardar y buscar en el conjunto completo de datos de entrenamiento, lo cual provoca que la carga computacional del clasificador sea alta cuando se realizan las pruebas (29).

NCA emplea una función de distancia extendiendo el clasificador k vecino más cercano (kNN). El clasificador LMNN extiende a su vez a NCA a través de un marco de trabajo de margen amplio (14).

LMNN está basado en que cada entrada de entrenamiento $\vec{x}_i$ debe compartir la misma etiqueta y_i que sus k vecinos más cercanos; las entradas de entrenamiento que tienen etiquetas distintas deben ser expulsadas del margen de la vecindad. LMNN intenta aprender una transformación lineal del espacio de entrada de manera que las entradas de entrenamiento cumplan dichas propiedades (37). Es decir, LMNN aprende una matriz de distancia M para mejorar los resultados

del kNN de manera que para cada instancia la función de distancia acerque cada vez más a los k vecinos más cercanos de la misma clase y aleje los de clase diferentes (30).

La función objetivo está compuesta por dos términos: el primero minimiza las distancias entre los vecinos objetivos y cada objeto, mientras que el segundo es una función de pérdida que penaliza la existencia de instancias de clases diferentes en la vecindad definida por los vecinos objetivos más un margen fijo (12).

Antes del aprendizaje, una entrada de entrenamiento tiene en su vecindad local vecinos objetivos e impostores. Durante el aprendizaje, los impostores son sacados del perímetro establecido por los vecinos objetivos. Luego existe un margen finito entre el perímetro y los impostores (37).

Una forma de la distancia de Mahalanobis es empleada por (38) para tratar de atraer a las instancias de la misma clase a un solo punto y al mismo tiempo mantener las instancias de clases diferentes lo más lejos posible (14).

Un algoritmo que dirige distribuciones de datos multimodo en aprendizaje de funciones de distancia es el LDM. La idea esencial del LDM es identificar primeramente un subconjunto de restricciones que solo involucren objetos relativamente cerca unos de otros para luego aprender una función de distancia que lo satisfaga (31). Para ello se emplea un proceso iterativo basado en el algoritmo de optimización de límite de (39). Específicamente, el LDM se inicializa usando la métrica Euclideana con el objetivo de identificar el conjunto inicial de restricciones locales para luego iterar alternativamente entre el paso del aprendizaje de funciones de distancia y el de refinar el subconjunto de las restricciones locales hasta que se alcance la convergencia (31). Este algoritmo optimiza la compactación y separabilidad local en un marco de trabajo probabilístico (14). LDM evita la dificultad computacional al emplear análisis de los auto-vectores. (31)

1.3.2. Aprendizaje de Distancia No Supervisado

En el Aprendizaje de Distancia No Supervisado los algoritmos se pueden dividir en dos clasificaciones: lineales y no lineales. Estas a su vez, al igual que en el Aprendizaje de Distancia Supervisado, se dividen en globales y locales. Dentro de los lineales se encuentran los algoritmos globales Análisis de los Componentes Principales (PCA) (40), Escalado Multidimensional (MDS) (41) y el Análisis de los Componentes Independientes (ICA) (42); y los locales LPP (43) y la Inclusión Preservadora del Vecindario (NPE) (44).

PCA (45) es una técnica estadística que transforma linealmente un conjunto de variables en un conjunto substancialmente menor de variables no relacionadas que representa la mayor parte

de la información del conjunto original (40). En general, PCA encuentra el sub-espacio que mejor preserva la variación de los datos (14).

MDS (42) se refiere a una clase de técnicas que usan las cercanías entre cualquier clase de objeto como entrada. Esta cercanía es un número que indica cuán similares o diferentes son dos objetos, o cómo son percibidos, o cualquier medida de este tipo. MDS es un procedimiento sistemático cuyos cálculos son tan complejos que las versiones más simples no pueden desarrollarse virtualmente sin la ayuda de una computadora (41).

ICA es otros de los algoritmos y su objetivo es buscar una transformación lineal a coordenadas en la cual los datos son estadísticamente independientes al máximo (14). Consiste en buscar una transformación lineal que minimice la dependencia estadística entre sus componentes. La definición de ICA dada por (46) depende de una función de contraste que sirve como criterio de selección de rotación. Está garantizado que alcanzará el máximo local solo en el caso en que las dos fuentes se encuentren en presencia de ruido no Gaussiano. El concepto de ICA puede ser visto como una extensión de PCA, lo cual solo puede imponer independencia hasta el segundo orden y, consecuentemente, definir las direcciones ortogonales (46).

LPP construye un gráfico incorporando información de la vecindad de la base de datos, luego se calcula una matriz de transformación que mapea los objetos a un sub-espacio. Esta transformación lineal preserva óptimamente la información de la vecindad local en cierto sentido. Los mapas son diseñados para minimizar una función objetivo diferente de las técnicas lineales clásicas. LPP puede ser aplicado a cualquier instancia nueva para localizarla en el espacio de representación reducido. (43)

NPE, dado un conjunto de instancias de datos en un espacio ambiental, construye una matriz de pesos que describe la relación entre cada instancia. NPE tiene propiedades similares a LPP: ambos tratan de descubrir la estructura local del conjunto de datos aunque sus funciones objetivo son diferentes (44).

LPP y NPE son las aproximaciones de Eigenmap Laplaciano (LE) (47) y la Inclusión Localmente lineal (LLE) (48) respectivamente, aunque fueron desarrollados originalmente para lograr la reducción de la dimensionalidad sin supervisión. Pueden ser extendidos a la configuración del aprendizaje de funciones de distancia supervisado donde la etiqueta de información es usada para la construcción de la matriz de peso. (14)

Por su parte, los ya antes mencionados, LLE y LE pertenecen al Aprendizaje de Distancia No supervisado lineal con enfoque local junto al Aprendizaje del Manifold Localmente Suavizado (LSML) (49). Con enfoque global se puede apreciar el algoritmo ISOMAP (50).

LLE calcula inclusiones, preservadoras del vecindario de baja dimensión, de entradas altamente dimensionales. Este método mapea sus entradas en un solo sistema de coordenadas global de baja dimensionalidad y sus optimizaciones no implican a los mínimos locales. LLE aprende la estructura global de conjuntos no lineales ya que usa las simetrías locales de reconstrucción lineal (48). Introduciendo múltiples vectores de pesos locales linealmente independientes para cada vecindario se logra una versión mejorada y estable de LLE. (14)

Por otro lado, LE es bastante simple, tiene unos pocos cálculos locales y un problema de auto-valores. Sin embargo, requiere la búsqueda de puntos inmediatos (vecinos) en un espacio de alta dimensionalidad. Este enfoque muestra estabilidad con relación a la inclusión, por tanto, mientras esta sea isométrica, la representación no va a cambiar. LE es capaz de producir representaciones similares independientemente de la resolución. (47)

LSML es muy efectivo no solo para la reducción de ruido Gaussiano sino también para el manejo efectivo de valores atípicos. Su rendimiento depende de dos valores: el número de vecinos usados para el suavizado local y el usado para el cálculo de los pesos. (49)

ISOMAP a su vez, define la conectividad de cada instancia a través de sus vecinos Euclideanos más cercanos en el espacio altamente dimensional, lo cual es vulnerable a errores de cortocircuito. Aún el error de este tipo más pequeño puede alterar muchas entradas en la matriz de distancia geodésica lo que conlleva a una inclusión de dimensión baja drásticamente diferente e incorrecta (50). En otras palabras, la función de ISOMAP es buscar el sub-espacio que mejor preserva estas distancias geodésicas entre dos instancias (14).

1.4. Metodologías estadísticas para la Comparación de los clasificadores

La evaluación estadística de resultados experimentales es una parte esencial para la validación de los métodos de aprendizaje automático. Para la comparación entre múltiples modelos sobre varias bases de datos, (52) y (53) utilizan la Prueba ANOVA y la Prueba de Friedman.

En el diagrama que se muestra a continuación se evidencia qué tipo de metodología escoger en dependencia de los resultados que se quiera obtener y en las características de las bases de datos con las que se debe trabajar. Por ejemplo si la distribución de los datos es normalizada, se cumple la propiedad de esfericidad de los datos y son de baja dimensionalidad, lo recomendable es usar la prueba paramétrica ANOVA, en caso contrario se recomienda la Prueba de Friedman. Ambas pruebas determinan si existen diferencias entre los clasificadores y cuentan con un conjunto de pruebas post-hoc para determinar si estas diferencias son significativas o no.

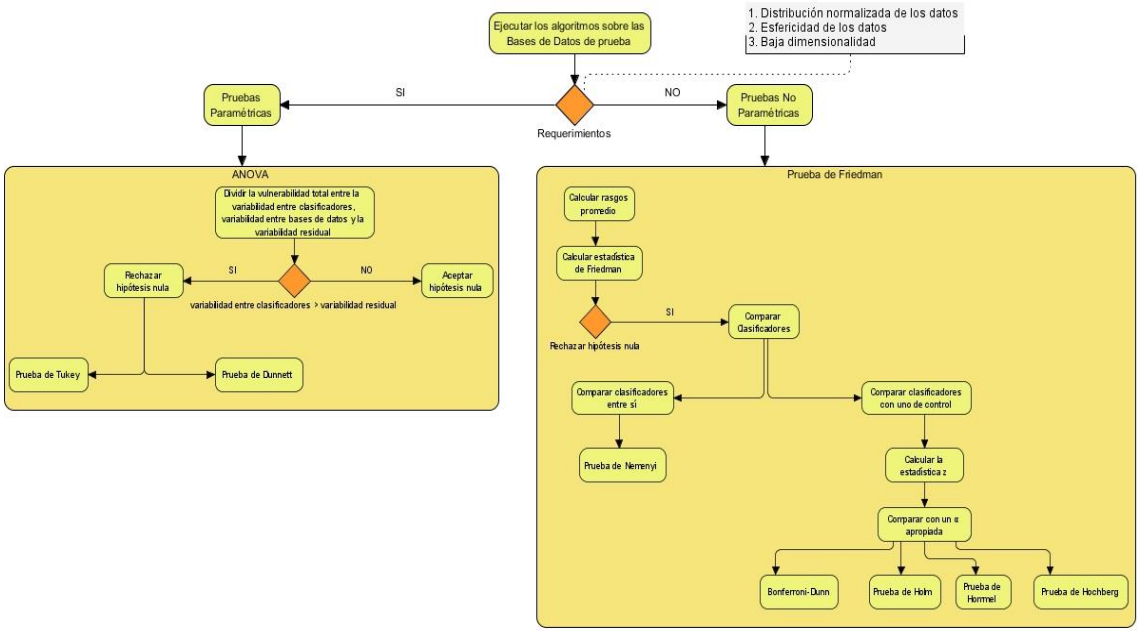

Diagrama 1 Diagrama del Proceso de selección de la metodología

1.4.1. Comparación de Múltiples Clasificadores

El método estadístico común para la comparación de múltiples clasificadores es ANOVA (conocido también como medidas- repetidas) (54). Una de las desventajas de ANOVA es que está basada en asunciones, que al analizar el desempeño de los algoritmos de aprendizaje automático, son posiblemente violadas. Primeramente ANOVA asume que las muestras son extraídas de distribuciones normalizadas cuando no hay garantía de ello, aunque esto no es un gran problema y muchos estadísticos no se oponen a su uso con la excepción de que las distribuciones fueran, para dar ejemplo, claramente bimodales (55). La segunda y más importante asunción de ANOVA es la esfericidad, que es una propiedad que requiere que las variables aleatorias tengan igual varianza. Debido a la naturaleza de los algoritmos de aprendizaje y las colecciones de datos, es una propiedad que no se puede dar por hecho. Violaciones de estas asunciones tiene un efecto aún mayor en las Pruebas Post-hoc, por lo que ANOVA no puede considerarse adecuada para estudios típicos de aprendizaje automático (51).

Un equivalente no paramétrico de ANOVA es la Prueba de Friedman (56; 57), la cual jerarquiza los algoritmos por cada colección de datos separadamente y en caso de empate asigna un rango promedio. Tomando r_i^j como el rango del algoritmo j de k algoritmos en la $i-ésima$ colección de datos de N colecciones de datos, la Prueba de Friedman compara los rasgos promedios de los algoritmos $R_j = 1/N \sum_i r_i^j$. En esta, la hipótesis nula afirma que todos los algoritmos son equivalentes y por eso sus rangos R_j deberían ser iguales. Bajo esta hipótesis nula, la estadística de la Prueba de Friedman se calcula:

$$X^2F = \frac{12N}{k(k+1)}\left[\sum_j R_j^2 - \frac{k(k+1)^2}{4}\right]$$

y es distribuida de acuerdo a X_F^2 con $k-1$ grados de libertad cuando k y N son lo suficientemente grandes ($k > 5$, $N > 10$) y en caso contrario (58; 59) calcularon valores críticos exactos. En (60) demostraron que la X_F^2 de Friedman no es lo deseadamente conservativa y derivaron una mejor estadística:

$$F_F = \frac{(N-1)X_F^2}{N(k-1) - X_F^2}$$

la cual es distribuida de acuerdo a la distribución F con $k-1$ y $(k-1)(N-1)$ grados de libertad. Teóricamente la Prueba de Friedman no paramétrica tiene menos poder que el ANOVA paramétrico cuando las asunciones de ANOVA se encuentran, en caso contrario no necesariamente. (57) experimentalmente los comparó en 56 problemas independientes y demostró que los dos coincidían generalmente. Cuando uno encontraba significancia en $p < 0.01$ el otro la muestra en al menos $p < 0.05$ respectivamente. Solamente en dos casos ANOVA encontró significancia que era insignificante para Friedman, mientras que lo contrario ocurrió en 4 casos. (51)

Si la hipótesis nula es rechazada se puede proceder con una Prueba de Post-hoc. La Prueba de Nemenyi (61) es similar a la de Tukey para ANOVA y es usada cuando los clasificadores son comparados entre sí. El desempeño de los mismos es significantemente diferente si el rango ordinario correspondiente difiere al menos la diferencia crítica (CD) donde los valores críticos q_α están basados en la estadística del rango Studentized dividida por $\sqrt{2}$. Esta diferencia crítica se calcula de la siguiente manera:

$$CD = q_\alpha \sqrt{\frac{k(k+1)}{6N}}$$

Cuando todos los clasificadores son comparados con un clasificador de control, se puede hacer uso de algún procedimiento general en lugar de la Prueba de Nemenyi para controlar el error FamilyWise (FWE) en la prueba de múltiples hipótesis como la corrección de Bonferroni u otros similares. El FWE o como también se conoce la inflación alfa o error acumulativo tipo I representa la probabilidad de que una de un conjunto de comparaciones es un error de tipo I. Aunque estos métodos se pueden considerar generalmente conservativos, en este caso son más efectivos que el de Nemenyi, ya que este ajusta el valor crítico para realizar $k(k-1)/2$ comparaciones

mientras que al comparar con un clasificador de control solo se hacen $k-1$ comparaciones. La estadística para comparar los clasificadores i y j utilizando estos métodos es:

$$z = (R_i - R_j)/\sqrt{k(k+1)/6N}$$

Este valor es usado para encontrar la probabilidad correspondiente para compararla con un α apropiada. Las pruebas difieren en la forma que ajustan el valor de α para compensar por comparaciones múltiples (51).

El test de Bonferroni-Dunn (62) controla la tasa del FWE al dividir α por la cantidad de comparaciones realizadas, o calcular la CD usando la ecuación de Nemenyi pero usando los valores críticos para $\alpha/(k-1)$. Como contraste del test de Bonferroni-Dunn, los procedimientos de paso-adelante y paso-atrás prueban secuencialmente las hipótesis ordenadas por su significancia, denotando los valores ordenados p por $p_1, p_2, \dots, p_{k-1}$ de manera que $p_1 < p_2 < p_{k-1}$. Ambos comparan cada p_i con $\alpha/(k-i)$, pero difieren en el orden de las pruebas. El procedimiento paso-atrás de Holm (63) comienza con el valor p más significante. Si p_1 es menor que $\alpha/(k-1)$ la hipótesis correspondiente es rechazada y se permite la comparación de p_2 y así sucesivamente. En el momento en que alguna hipótesis nula no pueda ser rechazada, las restantes son retenidas también. El procedimiento de paso-adelante de Hochberg trabaja de manera opuesta, comparando el valor más grande de p con α, el segundo con $\alpha/2$, y así consecutivamente hasta que encuentre una hipótesis que pueda rechazar. Todas las hipótesis con valores p más pequeños son rechazadas también (51).

El procedimiento de Hommel (64) encuentra el valor más grande j para el cual $p_{n-j+k} > k\alpha/j$ para todas las $k = 1$. Si tal j no existe se rechazan todas las hipótesis, en caso contrario se rechazan todas para las cuales $p_i \leq \alpha/j$. El procedimiento de Holm es más potente que el de Bonferroni-Dunn y no realiza asunciones adicionales sobre la hipótesis probada. La única ventaja del test de Bonferroni-Dunn es que es más fácil de describir y visualizar porque utiliza la misma CD para todas las comparaciones. Los métodos de Hochberg y Hommel rechazan más hipótesis que el de Holm pero en algunas circunstancias exceden el FWE prescrito ya que se basan en las conjeturas de Simes las cuales todavía se encuentran bajo investigación. Las diferencias entre todos estos métodos en la práctica son pequeñas, así que el método más complejo de Hommel no ofrece gran ventaja sobre el más sencillo de Holm (65). Aunque estos procedimientos se proponen como Pruebas Post-hoc para la Prueba de Friedman, también pueden usarse para controlar el FWE cuando múltiples hipótesis, de posiblemente varios tipos, son probadas. Aunque algunas veces la Prueba de Friedman reporta una diferencia significante, las Pruebas Post-hoc fallan en detectarla debido al poco poder que poseen (51).

1.5. Problemas de predicción con Salidas Compuestas

En la actualidad, existen problemas que requieren cada vez más métodos no convencionales que obtengan su solución y donde los resultados puedan modelarse de maneras distintas en dependencia del tipo de problema. En los problemas de predicción generalmente involucran la predicción simultánea de varias salidas utilizando el mismo conjunto de variables predictoras. A este tipo de problemas se le conoce como problemas de predicción con salidas compuestas. En este tipo de problemas en vez de obtener una salida simple, se necesitan predecir un conjunto de estas de manera simultánea. Es decir, todas las salidas son igualmente importantes por lo que se predicen simultáneamente con un solo modelo (10).

En términos más formales, dados $x = \{x_1, \dots, x_d\}$ y $y = \{y_1, \dots, y_m\}$ vectores aleatorios de variables de entrada y de salida respectivamente, se asume que los objetos (x, y) son generados por algún medio de acuerdo a la distribución de probabilidad acoplada $P(X, Y)$ en $\mathcal{X}\ x\ \mathcal{Y}$ donde $\mathcal{X} = R^d$ y $\mathcal{Y} = R^m$ son los dominios de X y Y, o sea, los espacios de entrada y salida. Tomando un conjunto de datos de entrenamiento $D = \{(x^1, y^1), \dots, (x^n, y^n)\}$ de n cantidad de ejemplos, el objetivo de los problemas de predicción con salidas compuestas es aprender un modelo $h: \mathcal{X} \rightarrow \mathcal{Y}$ que dado un vector de entrada x^q pueda predecir el vector de salida $\hat{y}^q = h(x^q)$ que se aproxime más al vector real de salida y^q. De esta forma, cada salida se predice independientemente y las posibles relaciones entre ellas no pueden ser explotadas (66).

Existen una serie de algoritmos para problemas de predicción con salidas compuestas como el Multi-Target Stacking (MTS), Regressor Chains (RC), el MTS y ERC Corregidos (MTSC y ERCC) (66) y el kNN-SP (67). A continuación se describe la esencia de dichos algoritmos.

1.5.1. Multi-target Stacking (MTS)

El entrenamiento en MTS está compuesto por dos etapas fundamentales: en la primera se aprenden m modelos $\mathrm{h_j}: \mathcal{X} \rightarrow \mathrm{R}$ de salidas simples independientes y en la segunda se aprende un segundo conjunto de m meta-modelos $\mathrm{h_j^*}: \mathcal{X}\ \mathrm{x}\ \mathrm{R^{m-1}} \rightarrow \mathrm{R}$ por cada salida $\mathrm{Y_j}$. Para predecir las salidas de una instancia desconocida $\mathrm{x^q}$ se aplica la primera etapa obteniéndose un vector de salida $\hat{\mathrm{y}}^{\mathrm{q}}$, luego al aplicar la segunda etapa en un vector de una entrada $\mathrm{x_j^{*q}}$ transformado se produce el vector final de salida $\hat{\hat{\mathrm{y}}}_{\mathrm{j}}^{\mathrm{q}}$.

1.5.2. Regressor Chains (RC)

El entrenamiento de RC consiste en la selección de una cadena aleatoria del conjunto de salida y luego construir un modelo de regresión separado para cada una de las salidas. Asumiendo que se selecciona la cadena por defecto $C = \{Y_1, Y_2, \dots, Y_m\}$ donde C es un conjunto ordenado, el primer modelo concerniente a la predicción de Y_1, tiene la forma $h_1: \mathcal{X} \rightarrow R$. Los modelos sub-

siguientes $h_{j,j>1}$ son entrenados en bases de datos transformadas y tienen la forma $h_j: \mathcal{X}\ x\ R^{j-1} \rightarrow R$. Dada tal cadena de modelos, el vector de salida $\hat{y}^q$ de una entrada desconocida x^q se obtiene al aplicar secuencialmente los modelos h_j.

El principal problema de RC es que al usar una sola cadena aleatoria para las salidas que aparecen antes en ella no se pueden modelar relaciones estadísticas potenciales con las que aparecen luego. Además, es casi seguro que se propaguen y amplifiquen errores de predicción a lo largo de la cadena cuando se realice esta tarea para una nueva instancia de prueba.

1.5.3. MTS y ERC Corregidos (MTSC y ERCC)

El MTSC y el ERCC utilizan un enfoque interno *f-fold cross validation* para obtener estimados fuera de la muestra para las variables de salida. El estimado de esta validación cruzada puede ser menos preciso que los obtenidos durante la predicción ya que los modelos son hechos usando el $\frac{f-1}{f}\%$ del conjunto de entrenamiento en su totalidad.

1.5.4. K Nearest Neighbor Structure Prediction (kNN-SP)

Este algoritmo soporta el uso de las distancias Euclideana con pesos, de Mahalanobis y de Chebyshev. Puede manejar valores de atributos ausentes, además de los tres tipos de datos existentes (vectores, jerarquías y series de tiempo).

Cuando predice la salida de un ejemplo nuevo x, toma los k ejemplos de entrenamiento más cercanos y sus salidas, su predicción es el prototipo de las k salidas. La contribución de las salidas es igual para todas por defecto (valor 1) pero puede ser modificada por la proximidad correspondiente a x.

1.6. Métrica de evaluación de los Algoritmos para Problemas de Predicción con Salidas Compuestas

Para la evaluación de estos algoritmos se utiliza el promedio del error cuadrático medio (aRRMSE, según sus siglas en inglés) entre lo que ha predicho el algoritmo y lo medido en cada base de datos para las variables de salidas. Esta métrica se calcula a partir del error cuadrático medio (RRMSE).

Dado un modelo h inducido del conjunto de entrenamiento D_{train}, el RRMSE se estima basado en un conjunto de prueba D_{test} de acuerdo a:

$$RRMSE(h; D_{test}) = \sqrt{\frac{\sum_{(x,y_j)\in D_{test}} (\hat{y}_j - y_j)^2}{\sum_{(x,y_j)\in D_{test}} (\bar{Y}_j - y_j)^2}}$$

donde $\bar{Y}_j$ es el valor medio de las variables de las salidas Y_j sobre D_{train} y $\hat{y}_j$ la estimación de $h(x)$ para Y_j. Esta métrica se estima haciendo uso del enfoque *hold-out* para grandes bases de datos, mientras que en caso contrario se utiliza *10-fold cross validation*.

Tomando lo anteriormente expuesto como base se puede calcular el aRRMSE de la siguiente forma:

$$aRRMSE(h; D_{test}) = \frac{1}{q}\sum_{j=1}^{q}\sqrt{\frac{\sum_{(x,y)\in D_{test}}(h(x_j)-y_j)^2}{\sum_{(x,y)\in D_{test}}(\bar{Y}_j-y_j)^2}}$$

1.7. Herramientas y tecnologías

A continuación se describen las principales características de las herramientas existentes que implementan técnicas de AA. Se muestra una comparación basada en diferentes aspectos entre las herramientas WEKA y Shogun.

La herramienta Shogun fue creada en 1999 y su lenguaje principal es C++. Esta herramienta de aprendizaje automático de propósito general tiene especial enfoque en el aprendizaje a gran escala, en los métodos del núcleo e interfaces con varios idiomas (68). Ofrece un gran número de modelos de aprendizaje automático (69).

WEKA (Waikato Environment for Knowledge Analysis) es una plataforma para el aprendizaje automático y la minería de datos escrito en Java y desarrollado en la Universidad de Waikato. A continuación se muestra una serie de tablas donde se especifican las funcionalidades de cada una de estas herramientas (70). WEKA fue diseñado para ofrecer toda una gama de técnicas de aprendizaje automático en una interfaz común aplicables consistentemente y que sea suficientemente flexible para que permita la incorporación de nuevos esquemas. WEKA es una herramienta útil que logra reducir el nivel de complejidad que conlleva la obtención de datos del mundo real, así como la evaluación de su salida, proporcionado una ayuda flexible para la investigación de aprendizaje automático (71).

Rasgos generales	**SHOGUN**	**WEKA**
Interfaz gráfica para el Usuario	NO	SI
Clasificación	SI	SI
Clasificación multiclase	SI	SI
Regresión	SI	SI
Aprendizaje a gran escala	SI	NO

Aprendizaje semi-supervisado	NO	NO

Tabla 1. *Comparación de las herramientas SHOGUN y WEKA considerando aspectos generales.* **(68)**

Clasificadores	**SHOGUN**	**WEKA**
Redes Bayesianas	NO	SI
kNN	SI	SI

Tabla 2. *Comparación de las herramientas SHOGUN y WEKA según los clasificadores que poseen* (68).

Clasificadores lineales	**SHOGUN**	**WEKA**
Máquina de Programación Lineal	SI	NO
LDA	SI	NO

Tabla 3. *Comparación de las herramientas SHOGUN y WEKA según los clasificadores lineales que poseen* (68).

Reducción de dimensionalidad	**SHOGUN**	**WEKA**
PCA	SI	SI
KPCA	SI	NO
MDS	SI	NO
LLE	SI	NO
LE	SI	NO
ISOMAP	SI	NO

Tabla 4. *Comparación de las herramientas SHOGUN y WEKA según los algoritmos de reducción de dimensionalidad que poseen* (68).

Agrupamiento	**SHOGUN**	**WEKA**
Agrupación jerárquica	SI	SI
k-means	SI	SI

Tabla 5. *Comparación de las herramientas SHOGUN y WEKA teniendo en cuenta los algoritmos de agrupamiento con los que cuenta* (68).

R es un software de uso gratuito, para análisis estadístico y gráfico que facilita una amplia gama de técnicas y que se desarrolla a través de paquetes que aportan algunos de sus usuarios para uso de todos. R es uno de los entornos que más se está desarrollando hoy día. Tiene alrededor de 13 librerías estadísticas definidas en su paquete base y ofrece un buen número de paquetes de rutinas especializadas. Con el uso de R se tiene acceso fácil a una amplia variedad de técnicas estadísticas y gráficas. Las facilidades de programación incluidas en R son muy amplias, lo que hace más eficiente la implementación de nuevos procedimientos, así como el uso reiterado de funciones existentes. Se necesita un conocimiento amplio de estadística para operarlo. Existen programas menos complejos de utilizar, con un producto final muy similar al programa R. No tiene la capacidad de interactuar con Excel (7; 8)

MATLAB es un entorno de computación y desarrollo de aplicaciones totalmente integrado orientado para llevar a cabo proyectos en donde se encuentren implicados elevados cálculos matemáticos y la visualización gráfica de los mismos. MATLAB integra análisis numérico, cálculo matricial, proceso de señal y visualización gráfica en un entorno completo donde los problemas y sus soluciones son expresados del mismo modo en que se escribirían tradicionalmente, sin necesidad de hacer uso de la programación tradicional. Dentro de sus características se encuentran: cálculos intensivos desde un punto de vista numérico, gráficos y visualización avanzada, lenguaje de alto nivel basado en vectores, arreglos y matrices; y una colección muy útil de funciones de aplicación (6). Las desventajas de MATLAB son las siguientes: es privativa, no es de código abierto y no se integra a aplicaciones empresariales

MULAN es una librería de Java para el aprendizaje desde datos multi-etiqueta. Ofrece un exceso de algoritmos para clasificación multi-etiqueta y categorización de etiqueta. Además brinda un marco de trabajo que calcula una gran variedad de medidas de evaluación multi-etiqueta a través de validación cruzada. Algunas de sus desventajas son las siguientes: no posee interfaz gráfica para el usuario y que corre todo en la memoria principal por lo que existen limitaciones cuando se utilizan bases de datos muy extensas. MULAN fue construido sobre WEKA para tomar ventaja de los recursos de esta en lo que concierne al aprendizaje supervisado (72).

KEEL es una herramienta de código abierto, escrita en Java que puede ser usada para distintas tareas de descubrimiento de conocimiento. Contiene variedad de algoritmos utilizados en tareas de extracción de información, pre- procesamiento de datos, algoritmos de aprendizaje basados en inteligencia computacional, modelos híbridos, metodologías estadísticas para contrastar

experimentos, entre otras. KEEL asiste en tareas de regresión, clasificación, agrupamiento y minería de patrones y más. Esta herramienta es una de las menos utilizadas por la comunidad internacional, las estructuras de las bases de datos que se utilizan en ella son complejas, está generalmente enfocada a los algoritmos evolucionales y en sus últimas versiones todavía no está totalmente estable. Además no es una herramienta de propósito general (9).

CLUS es un software libre de árboles de decisión e inducción de reglas que implementa un marco de trabajo de agrupamiento predictivo, el cual contiene técnicas de agrupamiento no supervisado y modelado predictivo. Fue desarrollado por el Grupo de Lenguajes Declarativos e Inteligencia Artificial de la Universidad Católica de Leuven de Bélgica y del Departamento de Tecnologías del Conocimiento del Instituto Jožef Stefan de Eslovenia (73).

1.8. Bases de Datos

Un factor en común que tienen todos los experimentos realizados por los expertos acerca del enfoque de aprendizaje de funciones de distancia para salidas simples, es la fuente de obtención de las bases de datos que utilizaron en ellos: UC Irvine. También, aunque no en gran cantidad, algunos autores utilizaron bases de datos de COREL, del MNIST, del ORL y de la Universidad de Yale.

Base de Datos	**Clases**	**Rasgos**	**Instancias**	**Descripción**
Banknote_authentication	2	5	1372	Características de billetes para determinar si es falso o auténtico
Yeast	10	9	1483	Sitios de la localización celular de las proteínas
Ecoli	8	8	336	Estos datos contienen sitios de localización de proteínas
Glass	7	10	214	Base de datos de identificación de cristal
Hayes_roth	3	5	160	Base de datos de Hayes Roth
Iris	3	5	150	Base de datos de plantas iris
Ionosphere	3	4	351	Base de datos de la Ionosfera de la Universidad Johns Hopkins

Diabetes	2	9	768	Dados los signos determinar si el paciente sufre de diabetes o no
Vehicle	4	18	946	Datos acerca de las siluetas de los vehículos
Vertebral_Columm	2	7	310	Conjunto de datos que contiene los valores de seis características biomecánicas utiliza para clasificar a los pacientes ortopédicos
BUPA	2	7	345	Trastornos hepáticos BUPA
Australian	2	15	690	Aprobación de crédito australiano
Seed	3	8	96	Núcleos examinados que pertenecen a tres diferentes variedades de trigo: Kama, Rosa y Canadá
Fertility	2	10	99	Muestras de semen analizado de acuerdo a los criterios de la OMS 2010
Toracic	2	16	400	Datos de pacientes que se sometieron a grandes resecciones pulmonares para el cáncer de pulmón primario

Tabla 6. Características de las bases de datos más usadas por los expertos en el tema *(74)*

Descripción de las Bases de Datos

Banknote_authentication: Los datos fueron obtenidos a partir de imágenes que fueron tomadas de tipos de billetes auténticos y falsificados. Para la digitalización, se utilizó una cámara industrial normalmente utilizados para la inspección de impresión. Las imágenes finales tienen 400x 400 píxeles. Debido a la lente objetivo y la distancia de los objetos investigados se tomaron imágenes en escala de grises con una resolución de unos 660 dpi. La herramienta Wavelet Transform se utilizó para extraer las características de las imágenes.

Glass: Viña realizó una prueba de comparación de su sistema basado en normas, BEAGLE, el algoritmo del vecino más próximo, y el análisis discriminante. En la determinación de si el vaso fue un tipo de vidrio "float" o no.

Hayes_roth: Esta base de datos contiene 5 atributos numéricos. Sólo un subconjunto de tres se utiliza durante la prueba (el último 3). Por otra parte, sólo 2 de los 3 conceptos son "utilizados" durante las pruebas (es decir, aquellos con los prototipos 000 y 111).

Iris: El conjunto de datos contiene 3 clases de 50 casos cada uno, donde cada clase se refiere a un tipo de planta iris. Una clase es linealmente separable de las otras 2. Estos últimos no son linealmente separables unos de otros.

Ionosphere: Estos datos de radar se recogieron mediante un sistema en Goose Bay, Labrador. Este sistema consiste en una red en fase de 16 antenas de alta frecuencia con una potencia total transmitida en el orden de 6,4 kilovatios. Los objetivos eran electrones libres en la ionosfera. "Buenos" ecos de radar son los que muestran evidencia de algún tipo de estructura en la ionosfera. Devuelve "Malos" a aquellos que no lo hacen; sus señales pasan a través de la ionosfera.

Diabetes: La variable de diagnóstico binaria investigada evalúa si el paciente muestra signos de diabetes de acuerdo a criterios de la Organización Mundial de la Salud (es decir, si la post carga de glucosa plasmática de 2 horas es por lo menos 200 mg / dl en cualquier examen encuesta o si se encuentran durante la rutina atención médica). La población vive cerca de Phoenix, Arizona, EE.UU.

Vehicle: Las características fueron extraídos de las siluetas por el HIPS (Imagen jerárquica Sistema de Procesamiento) BINATTS de extensión, que extrae una combinación de características independientes de escala que utilizan ambos momentos clásicos medidas basadas tales como escalado varianza, asimetría y curtosis sobre los mayores / menores ejes y medidas heurísticos tales como huecos, circularidad, ortogonalidad y la compacidad.

Vertebral_columm: Conjunto de datos biomédica integrada por el Dr. Henrique da Mota durante un período de residencia médica en el Grupo de Investigación Aplicada en Ortopedia (GARO). Los datos se han organizado en dos tareas de clasificación diferentes, pero relacionados. La primera tarea consiste en clasificar a los pacientes como pertenecientes a una de tres categorías: Largo (100 pacientes), disco Hernia (60 pacientes) o espondilolistesis (150 pacientes). Para la segunda tarea, las categorías de disco Hernia y espondilolistesis se fusionaron en una sola categoría denominada como "anormal". Por lo tanto, la segunda tarea consiste en clasificar a los pacientes como pertenecientes a uno de cada dos categorías: Largo (100 pacientes) o anormales (210 pacientes). Proporciona archivos también para su uso en el entorno de WEKA.

BUPA: Los primeros 5 variables son todos los análisis de sangre que se cree que son sensibles a los trastornos hepáticos que puedan surgir por el consumo excesivo de alcohol. Cada línea del archivo *bupa.data* constituye el récord de un solo individuo de sexo masculino.

Australian: Este archivo se refiere a las solicitudes de tarjetas de crédito. Todos los nombres y valores de los atributos se han cambiado para símbolos sin sentido para proteger la confidencialidad de los datos. Este conjunto de datos es interesante porque hay una buena mezcla de atributos (continuos, nominales, con un pequeño número de valores y nominales con un mayor número de valores). También hay unos pocos valores perdidos.

Seed: El grupo compuesto por los núcleos examinados pertenecen a tres diferentes variedades de trigo: Kama, Rosa y Canadá, 70 elementos cada uno, seleccionados al azar para el experimento. Visualización de alta calidad de la estructura interna del núcleo se detectó utilizando una técnica de rayos X blandos. Es no destructivo y considerablemente más barato que otras técnicas de imagen más sofisticados como la microscopía de barrido o la tecnología láser. Las imágenes fueron grabadas en las placas de rayos X KODAK 13x18 cm. Los estudios se llevaron a cabo utilizando combinar grano de trigo cosechado procedentes de campos experimentales, exploradas en el Instituto de Agrophysics de la Academia Polaca de Ciencias en Lublin.

Breast_cancer: Las muestras llegan periódicamente según informa el Dr. Wolberg según sus casos clínicos. La base de datos, por lo tanto, refleja esta agrupación cronológica de los datos.

Toracic: Los datos se recogieron de forma retrospectiva en el Centro de Cirugía Torácica Wroclaw para los pacientes que se sometieron a grandes resecciones pulmonares para el cáncer de pulmón primario en los años 2007-2011. El Centro está asociado a la Cirugía de la Universidad de Medicina de Wroclaw y Baja-Silesia Centro de Enfermedades Pulmonares, Polonia Departamento de torácico, mientras que la base de datos de la investigación constituye una parte del Registro Nacional de Cáncer de Pulmón, administrado por el Instituto de la Tuberculosis y Enfermedades Pulmonares en Varsovia, Polonia.

1.9. Conclusiones parciales del Capítulo

Luego de realizar un estudio de los algoritmos de aprendizaje de funciones de distancia del estado del arte se identificaron dentro de los algoritmos globales al ITML (por su naturaleza sencilla que facilita extenderlo a problemas de predicción con salidas compuestas) y al LMNN (por la obtención de los mejores resultados en ese ámbito cuando se aplica a distintas problemáticas) dentro de los locales como los más representativos, extendidos, utilizados y generalizados por diferentes dominios de aplicación para implementarlos en el lenguaje Java y extenderlos para poder aplicarlos a problemas de predicción con salidas compuestas.

Se seleccionó la herramienta WEKA para añadirle la implementación de estos algoritmos ya que no cuenta con un enfoque de aprendizaje de funciones de distancia y es una herramienta que cuenta con una vasta comunidad de investigadores y desarrolladores que la utilizan por lo que la mayoría de los algoritmos que se encuentran ya desarrollados se pueden encontrar en ella. Además es una herramienta de código abierto que permite la modificación del mismo. Con esto también se aumenta el poder predictivo de la herramienta, que es el objetivo de la investigación.

Se seleccionó para la comparación de los algoritmos implementados con algunos algoritmos del estado del arte la Prueba de Friedman ya que debido a las características de las Bases de Datos con las que se realizará dicha prueba no se puede asegurar que se cumplen las asunciones de ANOVA. Al realizar este tipo de prueba se obtendrán los resultados necesarios para establecer un ranking de los algoritmos y comprobar si la propuesta de la investigación es más eficiente que la solución existente.

Capítulo 2

Propuesta de solución a los problemas de clasificación basados en Aprendizaje de funciones de distancia

2.1. **Introducción**

En este capítulo se explicará la propuesta de solución de la investigación. Esta consiste en la implementación del algoritmo global ITML y del local LMNN para incluirlos entre los clasificadores de WEKA. Luego se formula la extensión teórica de los mismos para problemas de predicción con salidas compuestas. Además se muestra la relación de las clases implementadas con las que ya traen por defecto dichas herramientas.

2.2. **Descripción e Implementación**

2.2.1. Large Margin Nearest Neighbor (LMNN)

Es el objetivo del LMNN (37) intentar aprender una función de distancia de Mahalanobis $d_M(\vec{x}_i, \vec{x}_j)$ haciendo uso de la parametrización de la matriz M semidefinida positiva:

$$M = L^T L$$

Cabe destacar que cualquier matriz M formada de una matriz $\mathbb{R}$ valuada L expresada en esta ecuación será semidefinida positiva. Dicha función de distancia debe permitir calcular la distancia entre pares de objetos utilizando la matriz M como parametrización es posible expresar la función de distancia de la forma:

$$D_M(\vec{x}_i, \vec{x}_j) = (\vec{x}_i - \vec{x}_j)^T M(\vec{x}_i - \vec{x}_j)$$

Similar a como se ha expresado anteriormente sobre los algoritmos de aprendizaje de funciones de distancia el LMNN se modela como un problema de optimización convexa sobre el espacio de matrices semidefinidas positivas, intenta maximizar el margen por el que las instancias etiquetadas del conjunto de entrenamiento son clasificadas correctamente y mejora la veracidad de la clasificación usando el método kNN. El funcionamiento de LMNN se rige por los siguientes prospectos:

1. Cada entrada de entrenamiento $\vec{x}_i$ debe tener la misma etiqueta y_i que sus k vecinos más cercanos.
2. Las entradas de entrenamiento con etiquetas de clase distintas deben ser ampliamente separadas.

El aprendizaje en LMNN requiere información auxiliar más allá de la etiqueta de las instancias de entrenamiento, por lo que identifica a los vecinos objetivos para cada una al comienzo del aprendizaje de modo que al aprender una transformación lineal del espacio de entrada los vecinos más cercanos de $\vec{x}_i$ sean los identificados anteriormente. Estos vecinos objetivos son fijados a priori y no cambian durante el proceso de aprendizaje.

Al mantener un margen amplio (finito) de distancia entre los impostores y los perímetros establecidos por los vecinos objetivos, LMNN es robusto a pequeñas cantidades de ruido en las entradas de entrenamiento. LMNN maneja tres conceptos fundamentales, cuyo conocimiento es necesario para entender su funcionamiento, ellos son: vecinos, vecindad e impostores. Los vecinos son aquellos objetos más cercanos a un objeto $\vec{x}_i$. La vecindad es aquella compuesta por $\vec{x}_i$ y sus vecinos. Se denominan impostores a las entradas etiquetadas distintamente en el conjunto de entrenamiento que invaden este perímetro, es decir, aquellos vecinos de $\vec{x}_i$ que no comparten la misma clase. Matemáticamente un impostor se define por:

$$\|L(\vec{x}_i - \vec{x}_l)\|^2 \leq \left\|L\left(\vec{x}_i - \vec{x}_j\right)\right\|^2 + 1$$

donde para una entrada $\vec{x}_i$ de etiqueta y_i y vecino objetivo $\vec{x}_j$, un impostor es una entrada $\vec{x}_l$ de etiqueta $y_l \neq y_i$ que no cumpla con la inecuación anteriormente expuesta.

Antes del aprendizaje, una entrada de entrenamiento tiene en su vecindad un conjunto de vecinos y a su vez para cada vecino un conjunto de objetos considerados impostores. Durante el aprendizaje, los impostores son sacados del perímetro establecido por los vecinos. Luego existe un margen finito entre el perímetro y los impostores que en este caso toma valor uno.

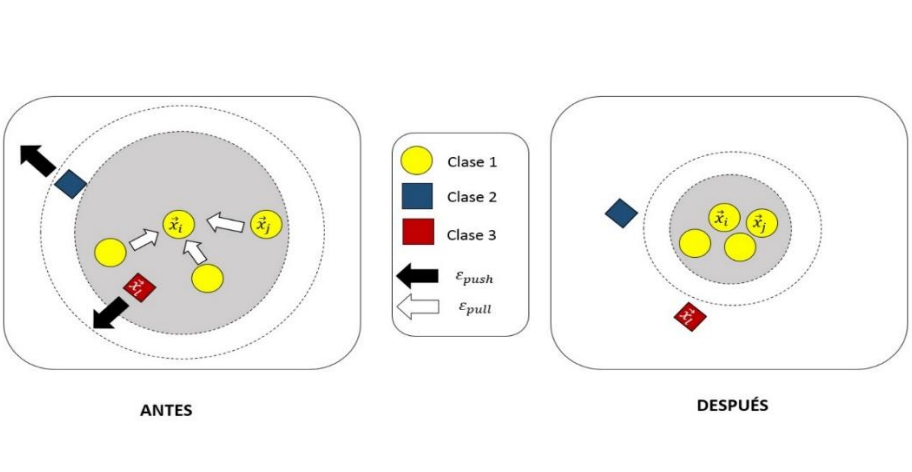

Figura 1. Funcionamiento del algoritmo LMNN

Teniendo en cuenta todo lo antes descrito se puede construir la función de costo para LMNN, la cual consiste de dos términos: uno que une más a los vecinos objetivos y otro que separa más a los impostores.

Este primer elemento es dado en términos de la transformación lineal L del espacio de entrada:

$$\varepsilon_{pull}(L) = \sum_{i,j\rightsquigarrow i} \left\| L\left(\vec{x}_i - \vec{x}_j\right) \right\|^2$$

El gradiente de este término genera una fuerza atrayente que impulsa más cerca a los vecinos objetivos en el espacio de entrada linealmente transformado. Se utiliza la notación $j \rightsquigarrow i$ para indicar que la entrada $\vec{x}_j$ es un vecino objetivo de $\vec{x}_i$, no es simétrica.

El segundo elemento en la función de pérdida penaliza las violaciones a la inecuación antes expuesta y está dada por:

$$\varepsilon_{push}(L) = \sum_{i,j\rightsquigarrow i} \sum_{l} (1 - y_{il})[1 + \left\| L\left(\vec{x}_i - \vec{x}_j\right) \right\|^2 - \|L(\vec{x}_i - \vec{x}_l)\|^2]_+$$

donde $y_{il} = 1$ si y solo si $y_i = y_l$, en caso contrario $y_{il} = 0$, y el término $[z]_+ = \max(z, 0)$ denota la pérdida estándar de Hinge. El sub-gradiente de este término genera una fuerza de empuje que repele a los impostores del perímetro establecido.

Finalmente, al combinar $\varepsilon_{pull}(L)$ y $\varepsilon_{push}(L)$, se obtiene la función de costo $\varepsilon(L)$.

$$\varepsilon(L) = (1 - \mu)\varepsilon_{pull}(L) + \mu\varepsilon_{push}(L)$$

Como $\varepsilon_{pull}(L)$ y $\varepsilon_{push}(L)$ tienen efectos opuestos, se introduce el parámetro $\mu \in [0,1]$ que equilibre sus objetivos. Este parámetro puede obtenerse a través de validación cruzada y no depende sensitivamente de su valor, el resultado de minimizar $\varepsilon(L)$.

LMNN formula $\varepsilon(L)$ como una optimización convexa, sin embargo no es así en la matriz de elementos de la transformación lineal L. Uno de los enfoques utilizados para la solución de este problema es aplicar el gradiente descendente en los elementos de L. Otro enfoque es reformular su optimización como una instancia de programación semidefinida (SDP).
La reformulación de $\varepsilon(L)$ se realiza como una optimización sobre matrices semidefinidas positivas y se trabaja en términos de $M = L^T L$.

$$\varepsilon(M) = (1 - \mu) \sum_{i,j\rightsquigarrow i} D_M\left(\vec{x}_i, \vec{x}_j\right) + \mu \sum_{i,j\rightsquigarrow i} \sum_{l} (1 - y_{il})[1 + D_M\left(\vec{x}_i, \vec{x}_j\right) - D_M(\vec{x}_i, \vec{x}_l)]_+$$

El solucionador diseñado para la clasificación LMNN re-estima iterativamente la distancia de Mahalanobis mientras trata de minimizar la función de costo. La cantidad de cálculo es reducida al mínimo gracias a una cuidadosa contabilidad mantenida de una iteración a otra.
Este solucionador implementa una proyección del sub-gradiente iterativa para optimizar $\varepsilon(M)$ en términos de la matriz semidefinida positiva M. Tomando a M_t como la función de distancia de Mahalanobis y a D_t como la distancia de Mahalanobis cuadrada en la iteración t, en cada

iteración la optimización da un paso junto al sub-gradiente para reducir la función de costo y luego proyecta M_t sobre el conjunto accesible. Este conjunto es el cono de todas las matrices S_+ semidefinidas positivas.

$$C_{ij} = (\vec{x}_i - \vec{x}_j)(\vec{x}_i - \vec{x}_j)^T$$

En particular en la iteración t, $D_t(\vec{x}_i, \vec{x}_j) = tr(M_t C_{ij})$. Tomando lo anterior como base, se puede replantear $\varepsilon(M)$ en función de M_t.

$$\varepsilon(M_t) = (1-\mu)\sum_{i,j\rightsquigarrow i} tr(M_t C_{ij}) + \mu \sum_{i,j\rightsquigarrow i}\sum_{l}(1-y_{il})[1 + tr(M_t C_{ij}) - tr(M_t C_{il})]_+$$

Esta relación es lineal en pares con respecto a M_t. Como lo que realmente interesa saber son los intrusos y $y_{il} = 1$ cuando las entradas comparten la misma etiqueta de clase por tanto $(1 - y_{il}) = 0$, se definen un conjunto de tripletas N^t de modo que $(i, j, l) \in N^t$, replanteando el gradiente de $\varepsilon(M_t)$.

$$G_t = \frac{d\varepsilon}{M_t} = (1-\mu)\sum_{i,j\rightsquigarrow i} C_{ij} + \mu \sum_{(i,j,l)\in N^t}(C_{ij} - C_{il})$$

En la próxima iteración $(t+1)$ se actualiza el gradiente. Como los vecinos objetivos se mantienen, se le sustraen los impostores que ya no están activos y se le añade los que se activaron en esa iteración.

$$G_{t+1} = G_t - \mu \sum_{(i,j,l)\in N^t - N^{t+1}}(C_{ij} - C_{il}) + \mu \sum_{(i,j,l)\in N^{t+1} - N^t}(C_{ij} - C_{il})$$

Tomando todo lo descrito anteriormente, se actualiza la matriz M_t para ir obteniendo la mejor métrica de distancia, que es el objetivo de este algoritmo. Esta forma de actualización es:

$$M_{t+1} \coloneqq P_S(M_t - \alpha G_{t+1})$$

Al calcular N^t en cada iteración requiere chequear cada tripleta (i, j, l) con $j \rightsquigarrow i$ en búsqueda de una violación potencial del margen o perímetro.

Como solo una pequeña fracción de intrusos se encuentran en el espacio de entrada, al comenzar el proceso de entrenamiento se chequean todas las tripletas y se mantiene una lista activa de todas las que violan el perímetro; aunque de 10 a 20 iteraciones se repite este proceso dependiendo de fluctuaciones de N^t.

Para iteraciones intermedias solo se buscan las violaciones del perímetro de esa lista de tripletas. Cuando finaliza la optimización, se verifica que N^t contiene todas las tripletas que incurren en violación del perímetro.

Este chequeo final es para verificar que converge al mínimo correcto. Si no es satisfecho, la optimización vuelve a empezar con el conjunto activo recién expandido.

El pseudo-código de este algoritmo es el que se presenta a continuación:

Algoritmo 1. LMNN

1: $M_0 \coloneqq I$ *//Inicializar con la matriz de identidad*
2: $t \coloneqq 0$ *//Inicializar contador*
3: $\mathcal{N}^{(0)}, \mathcal{N}_0 \coloneqq \{\}$ *//Inicializar conjuntos activos*
4: $G_0 \coloneqq (1-\mu)\sum_{i,j\rightsquigarrow i} C_{ij}$ *//Inicializar el gradiente*
5: **while** (no_converja) **do**
6: **if** mod(t,alg_const) $= 0 \vee$ (casi_converge){usando alg_const $= 10$} **then**
7: calcular $\mathcal{N}_{t+1}$ exactamente
8: $\mathcal{N}^{(t+1)} \coloneqq \mathcal{N}^{(t)} \cup \mathcal{N}_{t+1}$ *//Actualizar conjunto activo*
9: **else**
10: calcular $\mathcal{N}_{t+1} \approx \mathcal{N}_{t+1} \cap \mathcal{N}^{(t)}$ *//Solo buscar en el conjunto activo*
11: $\mathcal{N}^{(t+1)} \coloneqq \mathcal{N}^{(t)}$ *//Mantener conjunto activo sin modificar*
12: **end if**
13: $G_{t+1} \coloneqq G_t - \mu\sum_{(i,j,l)\in\mathcal{N}_t-\mathcal{N}_{t+1}}(C_{ij}-C_{il}) + \mu\sum_{(i,j,l)\in\mathcal{N}_{t+1}-\mathcal{N}_t}(C_{ij}-C_{il})$
14: $M_{t+1} \coloneqq P_S(M_t - \alpha G_{t+1})$ *//Tomar paso del gradiente y proyectarlo en el cono SDP*
15: $t \coloneqq t+1$
16: **end while**
17: Salida M_t

LMNN para problemas de predicción con salidas compuestas

Para un problema de predicción de salidas compuestas, el espacio de salida y es un conjunto de vectores que toman valores continuos. Un conjunto de objetos $O = \{O_1, \dots, O_n\} \in X\ x\ Y$ donde cada objeto $O_i = (\vec{x}_i, \vec{y}_i)$ del conjunto de entrenamiento está compuesto por un par de vectores que representan las variables de entrada y salida respectivamente.

En este sentido es difícil modelar las restricciones en pares por lo que se plantearon en forma de tripletas como el LMNN para salidas simples. Tomando:

$$\{O_i, O_j, O_l\} = \{(\vec{x}_i, \vec{y}_i), (\vec{x}_j, \vec{y}_j), (\vec{x}_l, \vec{y}_l)\}$$

se definen las tripletas como:

$$R = \{(O_i, O_j, O_l) |\ donde\ (\vec{x}_i, \vec{y}_i)\ está\ más\ cerca\ a\ (\vec{x}_j, \vec{y}_j)\ que\ a\ (\vec{x}_l, \vec{y}_l)\}$$

En este enfoque un impostor será aquel objeto que no respete la relación de distancia entre los objetos de la entrada y la salida. O sea, en términos matemáticos, es aquel que cumpla con la siguiente inecuación:

$$f_{ijl} = (d_M(\vec{x}_i, \vec{x}_j) - d_M(\vec{x}_i, \vec{x}_l))(d(\vec{y}_i, \vec{y}_j) - d(\vec{y}_i, \vec{y}_l)) \leq 0$$

El procedimiento a seguir a partir de esto es el mismo que para las salidas simples, con excepción de que al calcular $\varepsilon(M_t)$, se sustituye $(1 - y_{il})$ por δ_{ijl} donde:

$$\delta_{ijl} = \begin{cases} 1 & si\ f_{ijl}\ menor\ q\ 0 \\ 0 & si\ \ otra\ cosa \end{cases}$$

Tomando en consideración lo antes planteado se replantean las ecuaciones donde se hizo uso de este término.

$$\varepsilon_{push}(L) = \sum_{i,j\rightsquigarrow i} \sum_{l} \delta_{ijl}[1 + \left\|L(\vec{x}_i - \vec{x}_j)\right\|^2 - \|L(\vec{x}_i - \vec{x}_l)\|^2]_+$$

$$\varepsilon(M) = (1-\mu) \sum_{i,j\rightsquigarrow i} D_M(\vec{x}_i, \vec{x}_j) + \mu \sum_{i,j\rightsquigarrow i} \sum_{l} \delta_{ijl}[1 + D_M(\vec{x}_i, \vec{x}_j) - D_M(\vec{x}_i, \vec{x}_l)]_+$$

$$\varepsilon(M_t) = (1-\mu) \sum_{i,j\rightsquigarrow i} tr(M_t C_{ij}) + \mu \sum_{i,j\rightsquigarrow i} \sum_{l} \delta_{ijl}[1 + tr(M_t C_{ij}) - tr(M_t C_{il})]_+$$

Luego de las sustituciones hechas se procede con el algoritmo como se ha explicado anteriormente y como indica el pseudo-código del mismo.

2.2.2. Informatio Theoric Metric Learning (ITML)

Muchos de los algoritmos existentes que se han desempeñado bien en algunas tareas de aprendizaje fallan en cumplir requerimientos básicos como:

1. Ser suficientemente flexible para lograr soportar la cantidad de restricciones realizadas a través de diferentes paradigmas de aprendizaje.
2. Ser capaz de aprender una función de distancia generalizadora de los datos de prueba no vistos.
3. Ser rápido y medible.

ITML es un nuevo enfoque al aprendizaje de una clase de funciones de distancia con buen desempeño de generalización. Este algoritmo acomoda un rango de restricciones, incluyendo las de similitud y las de diferencia, y relaciones entre pares de distancias. Además se puede incorporar información previa en relación a la función de distancia en sí. Para algunos problemas la distancia Euclideana estándar puede funcionar bien. En otros, la distancia de Mahalanobis usando la inversa de la covarianza de ejemplo puede brindar resultados razonables. En dichos casos, ITML encuentra una función de distancia que sea la más cercana a una función de distancia inicial mientras que satisfaga las restricciones dadas.

Dado un conjunto de n puntos $\{x_1, \dots, x_n\}$ en $\mathbb{R}^d$, se busca una matriz definida positiva A que parametrice las distancias cuadradas de Mahalanobis:

$$d_A(x_i, x_j) = (x_i - x_j)^T A(x_i, x_j)$$

Asumiendo que lo expuesto anteriormente es conocido en lo relacionado a las distancias entre puntos y considerando relaciones que limitan la similitud o diferencia entre pares de puntos, se puede afirmar que dos puntos son similares si la distancia de Mahalanobis entre ellos es menor que una cota superior dada $d_A(x_i, x_j) \leq u$ para un valor relativamente pequeño de u y son diferentes si $d_A(x_i, x_j) \geq l$ para una l suficientemente grande. Dichas restricciones son consideradas generalmente como entradas para muchos problemas de aprendizaje semi-supervisado y pueden ser también fácilmente inferidos en un ajuste de clasificación donde las etiquetas de clase se conocen para cada instancia: las distancias entre puntos de la misma clase se restringen como similares mientras que las de los puntos de clases diferentes se restringen como diferentes.

Dado un conjunto de restricciones de distancia entre puntos como las descritas anteriormente, el problema consiste en aprender una matriz definida positiva A que parametrice la distancia de Mahalanobis correspondiente. ITML regulariza esta matriz para que esté lo más cerca posible a una función de distancia de Mahalanobis dada parametrizada por A_0. Luego se cuantifica la medida de cercanía entre A y A_0 a través de un enfoque natural de información teórica.

Se mide la distancia entre dos funciones de distancia de Mahalanobis parametrizadas por A_0 y A por la entropía relativa (diferencial) entre sus Gaussianas multivariadas correspondientes. Esto se plantea como una divergencia Kullback-Leibler (KL):

$$KL(p(x; A_0)||p(x; A)) = \int p(x; A_0) \log \frac{p(x; A_0)}{p(x; A)} dx$$

Esta distancia provee una medida de cercanía entre estas dos distancias de Mahalanobis y forma la base del problema dado a continuación. Dados pares de puntos similares S y pares de puntos diferentes D:

$$\min_A KL(p(x; A_0)||p(x; A))$$

Sujeto a:

$$d_A(x_i, x_j) \leq u \qquad (i, j) \in S$$
$$d_A(x_i, x_j) \geq l \qquad (i, j) \in D$$

Según (Davis, Kulis, Sra and Dhillon 2007) la función objetivo se puede expresar como un tipo particular de la función Divergencia Bregman, que se permite adaptar al método de Bregman (Censor 1997) para resolver el aprendizaje de funciones de distancia.

$$D_\gamma(x, y) = \gamma(x) - \gamma(y) - trace(x - y)^T \nabla\gamma(y)$$

Luego la divergencia de Burg la transforma en:

$$D(A, A_0) = -logdet(A) + logdet(A_0) + trace(A - A_0)^T \nabla - logdet(A_0)$$

Para introducir el término actual en una única solución se introduce el término Lagrangiano, que no es más que restarle a esta función las restricciones, o sea:

$$D(A, A_0) = -logdet(A) + logdet(A_0) + trace(A - A_0)^T \nabla - logdet(A_0) - \left(d_A(x_i, x_j) - u\right) - \left(d_A(x_i, x_j) + l\right)$$

La optimización se basa en la proyección Bregman mencionada anteriormente, que proyecta la solución actual en una única restricción a través de la regla de actualización:

$$M_{t+1} = M_t + \beta M_t C_{ij} M_t$$

Una limitación de ITML es que la selección de la matriz M_0 puede tener una influencia importante en la calidad de la métrica de distancia M.

A continuación se muestra el pseudo-código del algoritmo ITML:

Algoritmo 2. Métrica de aprendizaje de Información Teórica

Entrada: X: matriz de entrada dxn; S: conjunto de pares similares;
D: conjunto de pares diferentes; u, l: umbrales de distancia
A_0: matriz de Mahalanobis de entrada; γ: parámetro de holgura;
c: función índice de restricción

Salida:

A: matriz de Mahalanobis de salida

1: $A \leftarrow A_0, \lambda_{ij} \leftarrow 0 \; \forall \; i, j$
2: $\xi_{c(i,j)} \leftarrow u$ for $(i, j) \in S$ de_otra_manera $\xi_{c(i,j)} \leftarrow l$
3: **repeat**
4: obtener restricciones $(i, j) \in S$ o $(i, j) \in D$
5: $p \leftarrow \left(x_i - x_j\right)^T A (x_i - x_j)$
6: $\delta \leftarrow 1$ si $(i, j) \in S$, -1 de_otra_manera
7: $\alpha \leftarrow \min(\lambda_{ij}, \frac{\delta}{2}(\frac{1}{p} - \frac{\gamma}{\xi_{c(i,j)}}))$
8: $\beta \leftarrow \delta\alpha/(1 - \delta\alpha p)$
9: $\xi_{c(i,j)} \leftarrow \gamma\xi_{c(i,j)}/(\gamma + \delta\alpha\xi_{c(i,j)})$
10: $\lambda_{ij} \leftarrow \lambda_{ij} - \alpha$
11: $A \leftarrow A + \beta A\left(x_i - x_j\right)\left(x_i - x_j\right)^T A$
12: **until** convergencia
13: **return** A

ITML para problemas de predicción con salidas compuestas

Tomando en cuenta lo planteado anteriormente sobre las salidas compuestas, para el caso específico del ITML, que trabaja con las restricciones en pares es necesario realizar algunas

modificaciones ya que dado que la salida es un vector no hay manera de saber si dos objetos pertenecen a la misma clase o no. Dicho esto es necesario modificar el parámetro δ que controla si dos objetos son de la misma clase en el ITML para salidas simples que en el caso de salidas compuestas sería si pertenecen al mismo conjunto o no, quedando de la forma siguiente:

$$\delta = \begin{cases} -d_{ij}^{\vec{y}} & si \;\; d_{ij}^{\vec{y}} \geq \tau \\ d_{ij}^{\vec{y}} & si \;\; d_{ij}^{\vec{y}} < \tau \end{cases}$$

Donde el término $d_{ij}^{\vec{y}}$ representa la distancia existente entre dos vectores de salida y $\tau = 0.5$ un umbral que se introduce para saber si dos vectores de entrada pertenecen al mismo conjunto S o D, es decir:

$$d_{ij}^{\vec{y}} \geq \tau \quad entonces \quad \vec{x}_i, \vec{x}_j \in D$$

$$d_{ij}^{\vec{y}} < \tau \quad entonces \quad \vec{x}_i, \vec{x}_j \in S$$

Luego de estos ajustes se prosigue normalmente con el algoritmo, cuyos pasos a seguir indica su pseudo-código.

2.3. Integración del ITML y el LMNN a WEKA

WEKA es una herramienta que desarrollada sobre tecnología JAVA presenta una arquitectura por paquetes de clases. Por ejemplo en el paquete *weka.core* se encuentran implementadas todas las funciones de distancia que se utilizan en la herramienta. Cuenta además con un paquete llamado *weka.classifier* donde se encuentra la clase abstracta *Classifier* de la cual heredan el resto de los clasificadores incluidos en sus respectivos paquetes por ejemplo *weka.classifier.bay*es y *weka.classifier.lazy*. Como se muestra (Figura 2) en este último paquete fueron incluidos los nuevos clasificadores basados en algoritmos de aprendizaje de funciones de distancia que se proponen en este trabajo. Como parte de la integración también se implementaron dos nuevas funciones de distancia incluidas en el *weka.core* que utilizan función de distancia aprendida para el cálculo de la distancia entre instancias del conjunto de datos de entrenamiento. Como parte del aporte se incorporó un nuevo paquete a la herramienta con el nombre de *weka.distance.metric* donde se incluye la implementación de los algoritmos de aprendizaje de funciones de distancia propuestos en este artículo (Figura 3).

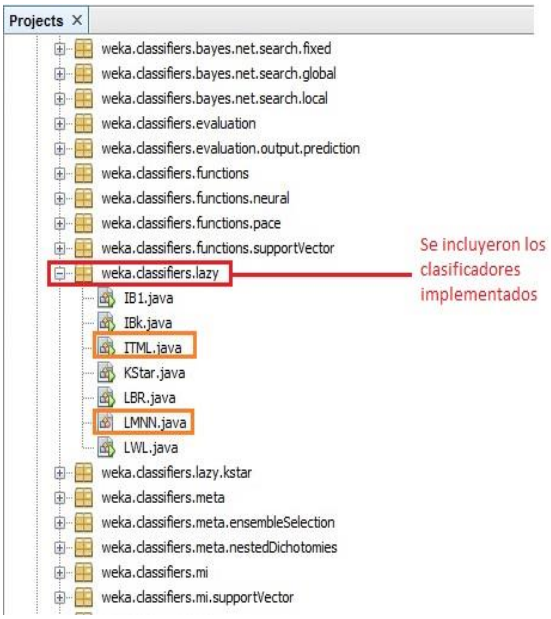

Figura 2 Paquetes de WEKA

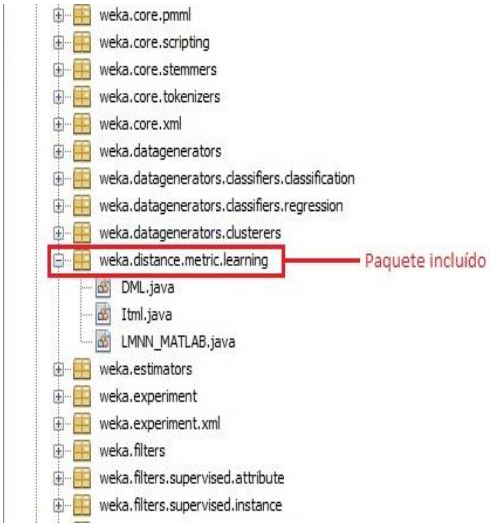

Figura 3 Paquete incluido

En el siguiente diagrama de clases (Diagrama 2) se muestra la relación de los clasificadores implementados y de las clases incluidas en el código de WEKA.

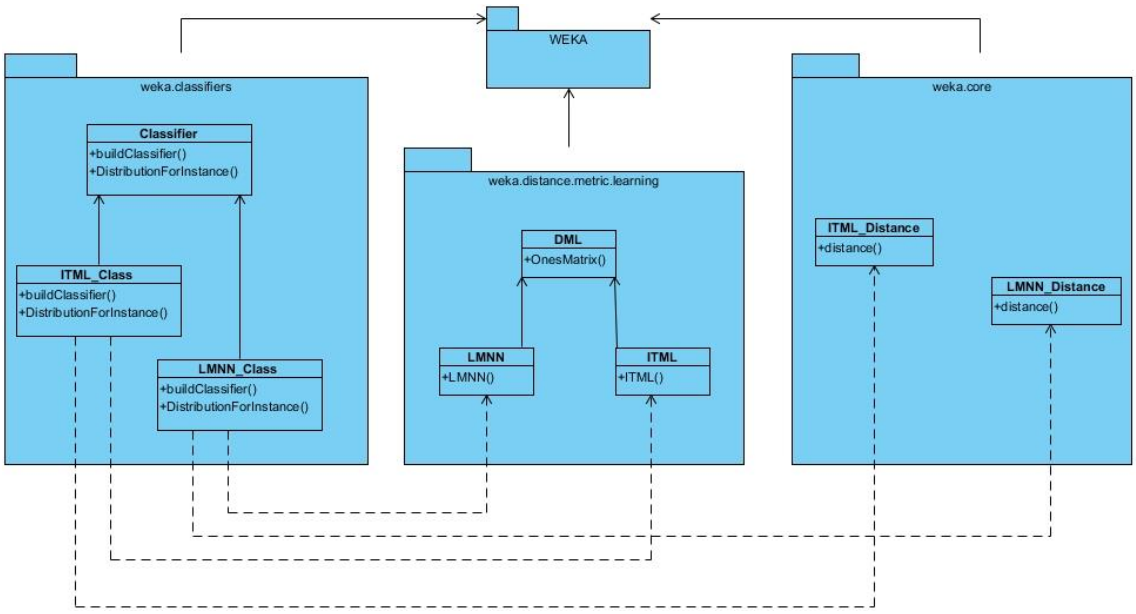

Diagrama 2 Diagrama de clases de WEKA luego de incluir las modificaciones

Como se puede observar se añadió un nuevo paquete llamado *weka.distance.metric.learning* que contiene las nuevas clases implementadas DML, LMNN e ITML (estas últimas heredan de DML). En DML se encuentra el método *OnesMatrix()* que es quien calcula la matriz de identidad y en las clases que contiene se incluye la implementación de dichos algoritmos respectivamente.

En los métodos *LMNN()* e *ITML()* es donde se construye la matriz M. Además se incluyeron en el paquete *weka.core* las nuevas funciones de distancia ya que en este paquete es donde se encuentran todas las funciones de distancia que utiliza WEKA, dichas funciones implementan el método *distance()* el cual calcula la distancia de Mahalanobis entre dos pares de instancias. En el paquete *weka.classifier* se añadieron los nuevos clasificadores que heredan de la clase abstracta Classifier e implementan los algoritmos del paquete añadido utilizando las nuevas funciones de distancia. Es importante señalar que estos clasificadores tienen dos métodos esenciales, el método *buildClassifiers()* que es donde se construye el modelo de aprendizaje que luego se utiliza para predecir las salidas en el método *distributionForInstance()*.

Al compilar nuevamente la herramienta se puede observar cómo los nuevos clasificadores implementados se pueden seleccionar del menú de clasificadores (Figura 4). Luego de seleccionado el clasificador se ajustan sus parámetros (Figura 5). Finalmente se muestran los resultados, es decir, la precisión del clasificador, la matriz de confusión, el error absoluto medio, entre otros (Figura 6).

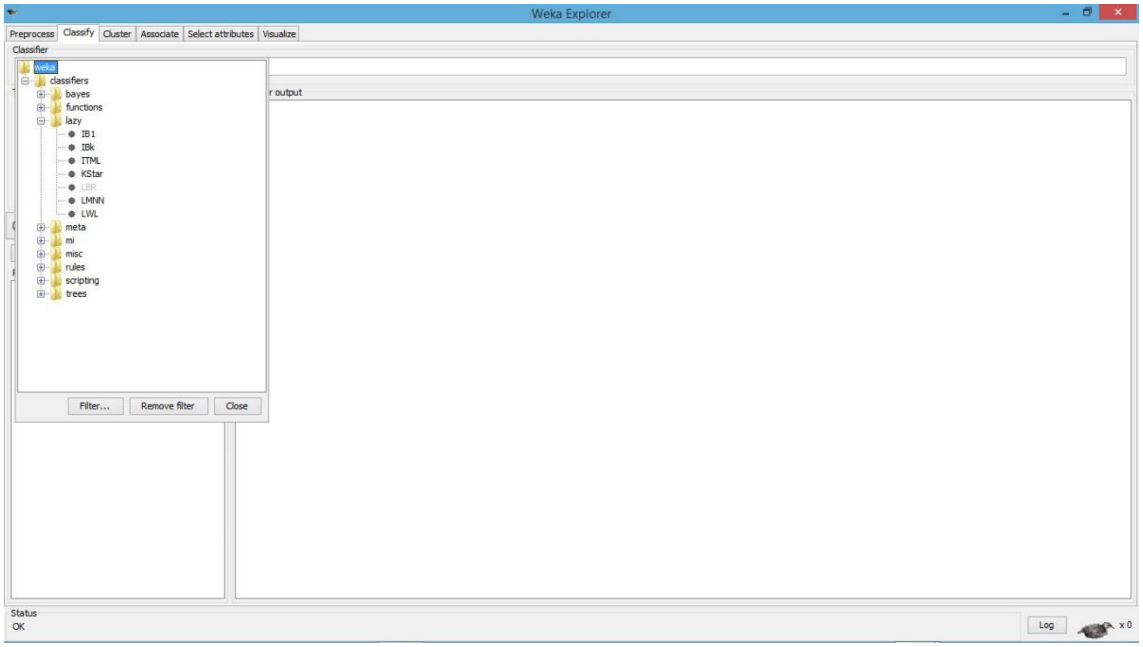

Figura 4 Selección de los clasificadores en la interfaz de WEKA

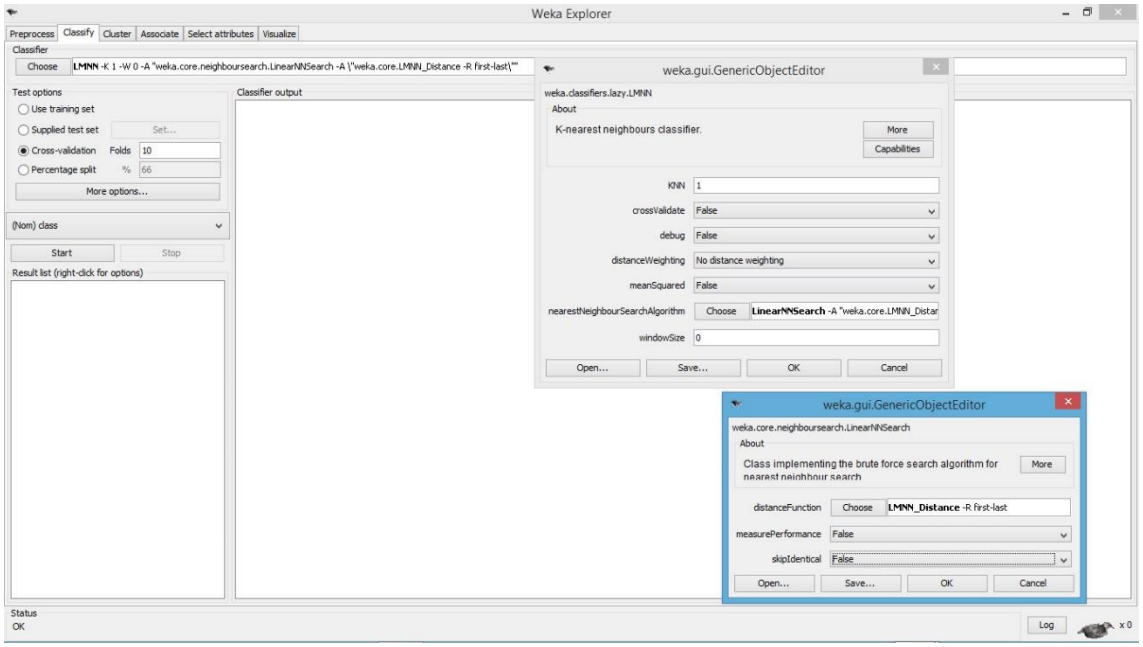

Figura 5 Ajuste de los parámetros del clasificador seleccionado

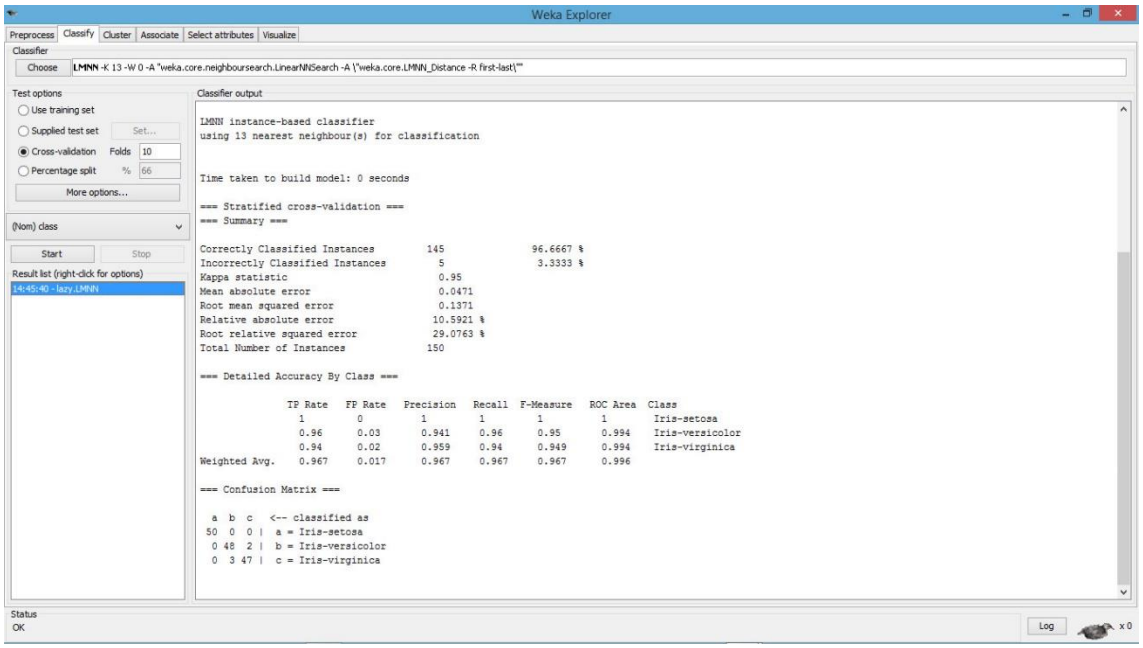

Figura 6 Resultados finales del clasificador seleccionado en la interfaz de WEKA

Como se pudo observar en las imágenes anteriores una vez, inicializada la herramienta WEKA y cargada la base de datos con la que se realizarán los experimentos, el usuario tiene la posibilidad de seleccionar uno de los nuevos clasificadores incluidos en la categoría de *lazy*. Se incluyeron en este paquete ya que es donde se encuentra la implementación del algoritmo base de WEKA, kNN. Luego se accede a la configuración de los parámetros en caso de que no se quiera utilizar la configuración que trae por defecto. Estos parámetros no son más que el número

de k, si se quiere o no realizar validación cruzada, la función de distancia, entre otros. Finalmente una vez terminado el proceso de clasificación, se muestran los resultados obtenidos.

Patrones de diseño

Para la implementación de los algoritmos se utilizaron los patrones de asignación de responsabilidades GRASP, que no son más que principios fundamentales de asignación de responsabilidades expresados como patrones. Los patrones utilizados son:

Creador: cuando una clase crea una instancia de otra, si agrega los objetos de otra, si registra las instancias de los objetos de otra, etc. En el caso de la investigación se utilizó fundamentalmente en la primera funcionalidad planteada. Ejemplo de ello es:

```
Instances train = new Instances(trainset.getDataSet());
```

Bajo Acoplamiento: se decide emplear en las clases con el objetivo de que estén lo menos relacionadas entre sí, para que en caso de producirse una modificación en alguna de ellas, repercuta mínimamente en el resto. Como se puede observar en el diagrama de clases cada clase (***Diagrama 2***) implementada se relaciona únicamente con las que necesita.

Experto: este patrón se selecciona para ser aplicado en todas las clases de manera que contengan la información necesaria para ejecutar una acción específica, siendo cada clase la más indicada para realizar una tarea (***Diagrama 2***).

Capítulo 3

Experimentos basados en la aplicación de los algoritmos implementados y resultados de la Prueba de Friedman.

3.1. Introducción

A continuación se mostrarán los resultados obtenidos al comparar los algoritmos implementados LMNN e ITML con el algoritmo base que trae WEKA por defecto que es el kNN, primeramente se muestra una tabla con algunos datos que se obtienen al ejecutarlos en WEKA y luego los resultados tras aplicar la Prueba de Friedman y como prueba post-hoc la Prueba de Bonferroni-Dunn. Además se muestra un Caso de Estudio por cada algoritmo donde se detalla su funcionamiento con una base de datos de prueba.

3.2. Resultados

A continuación se mostrará una tabla donde se puede apreciar el *accuracy* (la cercanía de una medición al valor verdadero) de los clasificadores implementados y sus variantes utilizando la validación cruzada, en comparación con la del algoritmo base kNN y sus variantes: kNN-CV (kNN utilizando validación cruzada) y kNN-DW (kNN utilizando pesos en la distancia, en este caso la distancia inversa), en las 15 bases de datos mencionadas anteriormente. Los campos en azul representan los mejores resultados para cada base de datos (*para más información consultar en Anexo A*).

La validación cruzada que se utiliza en este caso es la llamada *hold-one-out* , que encuentra el mejor $k$donde la medida del error sea la menor.

Base de Datos	***kNN***	***kNN-CV***	***kNN-DW***	***ITML***	***ITML-CV***	***LMNN***	***LMNN-CV***
Iris	96.666	96.0	96.667	96.667	97.333	96.667	97.333
Glass	70.560	71.495	72.429	71.962	72.897	72.897	76.635
Seeds	94.285	94.285	94.285	94.285	94.285	94.761	94.761
Diabetes	75.651	76.171	75.781	75.520	75.651	75.130	74.869
Vehicle	71.394	72.104	72.695	1.276	72.104	71.158	72.104
Vertebral _column	82.580	82.258	83.871	85.806	84.838	85.806	86.451
Ecoli	86.607	87.202	87.5	86.904	86.904	85.416	85.416
Banknote	99.854	99.854	99.927	100.0	100.0	100.0	100.0

Ionosphere	86.609	90.598	86.609	86.609	90.598	89.173	92.022
Fertility	88.0	88.0	88.0	88.0	89.0	88.0	89.0
Thoracic	85.319	85.319	85.106	85.106	84.893	85.106	85.106
Yeast	59.501	59.029	56.805	60.175	60.175	60.107	59.905
Hayes_roth	70.454	71.969	70.454	70.454	73.484	70.454	73.484
BUPA	65.217	63.768	68.115	71.014	68.985	71.594	69.275
Australian	86.811	86.087	85.942	86.956	86.231	86.087	85.072

Tabla 7 Comparación de los algoritmos basados en el *accuracy*

Como se muestra en la Tabla 7, tomando como dominio específico las 15 bases de datos co las que se efectuaron las pruebas, en solo 4 de estas el algoritmos kNN o sus variantes supera a los algoritmos propuestos basados en aprendizaje de funciones de distancia. Estos algoritmos superan al kNN (o sus variantes) un 73.333% de las veces.

3.3. Prueba de Friedman

Tomando como base la Tabla 7, donde se muestra el *accuracy* de cada algoritmo para cada base de datos, se aplica la Prueba de Friedman para determinar si existen diferencias entre los clasificadores a evaluar. Luego de aplicar esta prueba se obtiene un ranking de los algoritmos. Este ranking se encuentra representado en la siguiente tabla:

Algoritmo	Ranking
LMNN-CV	3.1
ITML-CV	3.16
ITML	3.89
LMNN	3.99
kNN-DW	4.43
kNN-CV	4.56
kNN	4.83

Tabla 8 Rango promedio de los algoritmos

Obtenido este ranking se procedió a realizar la Prueba de Bonferroni-Dunn, que compara los clasificadores con uno de control y utiliza la ecuación de diferencia crítica planteada por Nemenyi. Esta prueba *post-hoc* plantea que el desempeño de los clasificadores es significantemente diferente si el rango ordinario correspondiente difiere al menos la diferencia crítica (CD), obteniéndose los siguientes resultados para los 7 clasificadores en las 15 bases de datos de prueba para $\alpha = 0.1$:

$$q_\alpha = 2.3$$

$$CD = q_\alpha \sqrt{\frac{k(k+1)}{6N}} = 2.3 \sqrt{\frac{7(7+1)}{6 * 15}} = 1.7$$

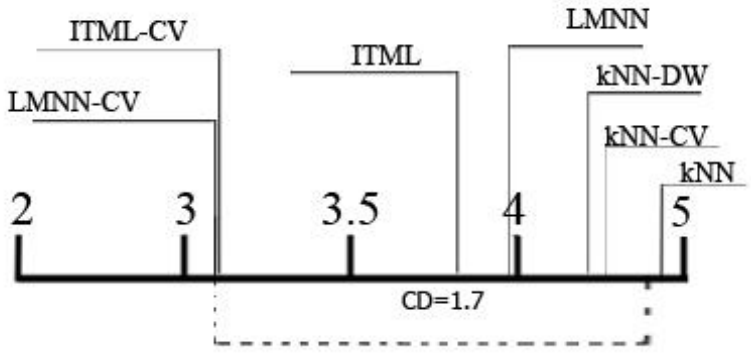

Figura 7 Resultados prueba de Bonferroni-Dunn para $\alpha = 0.1$

Tomando los resultados mostrados en el gráfico anterior, se puede afirmar que existen diferencias significativas entre el algoritmo LMNN-CV y el kNN base de WEKA, con un 90% de confiabilidad. Se demuestra que hacer una transformación lineal del espacio de entrada de los datos ofrece mejoras en el proceso de clasificación. Para el dominio específico de las 15 bases de datos de prueba, los algoritmos propuestos y sus variantes ocupan los primeros lugares en el ranking, superando a aquellos que no transforman el espacio de entrada (kNN y sus variantes).

3.4. Implementación de Caso de Estudio para LMNN

Tomando para este caso de estudio la base de datos Wine que contiene datos útiles para el reconocimiento del vino. Esta base de datos tiene un total de 178 instancias, 14 atributos y 3 clases en las que puede ser clasificado el vino.

Primeramente se inicializa la matriz de identidad M.

```
Matrix M = OnesMatrix(ncol);
```

Se calculan entonces los vecinos más cercanos de cada instancia y además se inicializa C_{ij}.

```
for (int i = 0; i < nRow; i++)
    {
        Instance instanceI = train.instance(i);
        numC = instanceI.value(nCol);
```

```
list = lnn.kNearestNeighbours(instanceI, k,numC,nCol);
indices = (int[]) list.get(0);
neighboursSameClass = (Instances) list.get(1);
int m  = neighboursSameClass.numInstances();
for (int j = 0; j < m; j++)
{
    Instance instanceJ = neighboursSameClass .instance(j);
    for (int z = 0; z < nCol; z++)
     {
       Xi.set(z, 0, instanceI.value(z));
       Xj.set(z, 0, instanceJ.value(z));
     }
   Cij= Cij.plus((Xi.minus(Xj)).times((Xi.minus(Xj)).transpose()));
 }
neighboursMap.put(i,indices);
}
```

Luego se calcula el gradiente.

```
G = InicializarCij(train, kg, ncol, nrow, lnn, neighboursMap).times(1-mu);
```

Después de esto se identifican los impostores y se construye la función objetivo.

```
slack = ActualizarTripletas(nrow,train,lmnnd,M,neighboursMap,ncol, kg,iter,Ytr);
C = (1-mu) * epull + mu * epush;
```

Finalmente se actualiza el gradiente y se proyecta en el semicono de las matrices semidefinidas positivas.

```
G = G.plus(New_Violations(train, ncol, slack, old_slack).times(mu).Minus (Resolved_Violations(train, ncol, slack, old_slack).times(mu)));
M = M.minus(G.times(eta/nrow));
M= ProyectarSemicono(M);
```

Se repite este proceso hasta que converja y se muestran los resultados.

21 vecinos=== Summary ===

Correctly Classified Instances	174	97.7528 %
Incorrectly Classified Instances	4	2.2472 %
Kappa statistic	0.966	
Mean absolute error	0.0642	
Root mean squared error	0.147	
Relative absolute error	14.6225 %	
Root relative squared error	31.3793 %	
Total Number of Instances	178	

=== Detailed Accuracy By Class ===

	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	1	0.017	0.967	1	0.983	0.999	1
	0.944	0	1	0.944	0.971	0.996	2
	1	0.015	0.96	1	0.98	0.999	3
Weighted Avg.	0.978	0.01	0.978	0.978	0.977	0.998	

=== Confusion Matrix ===

a	b	c	<-- classified as
59	0	0	a = 1
2	67	2	b = 2
0	0	48	c = 3

3.5. Implementación de Caso de Estudio para ITML

Manteniendo el uso de la base de datos Wine , para este algoritmo lo primero que se realiza es inicializar la matriz M como la matriz de identidad.

```
Matrix M = OnesMatrix(ncol);
```

Seguidamente se obtienen las matrices dobles X y Y al separar los datos de entrenamiento. Estas matrices están compuestas por las instancias.

```
LLenarMatrices(X, Y, nCol, nRow, workingSet);
```

Luego se calculan los valores l y u que son los umbrales necesarios para las restricciones de similaridad y disimilaridad.

```
double  [] lu = DistanceToPercentile(X, 5, 95, A_O);

 l = lu[0]; u=lu[1];
```

Se calcula entonces el número total de restricciones existentes.

```
int num_contrains = const_factor*kT*(kT-1);
```

Además se construye la matriz C de restricciones.

```
Matrix C = getContrains(Y, num_contrains, l, u,nCol);
```

Se modifican los valores de α, β y λ según la matriz M.

```
if(C.get(i, 2)==1)
      {
        alpha = Math.min(lamda.get(i, 0), gamma_proj* (1/wtw.get(0, 0) -                    1/bhat.get(i,0))
);
        lamda.set(i, 0, lamda.get(i, 0)-alpha);
        beta= alpha/(1-alpha*wtw.get(0, 0));
        bhat.set(i, 0, 1/(1/bhat.get(i, 0) + (alpha/gamma)));
      }
else if (C.get(i, 2)==-1)
     {
     alpha = Math.min(lamda.get(i, 0), gamma_proj* (1/bhat.get(i,0)-1/wtw.get(0, 0)));
     lamda.set(i, 0, lamda.get(i, 0)-alpha);
     beta = -1 * alpha/(1+alpha*wtw.get(0, 0));
     bhat.set(i, 0, 1/(1/bhat.get(i, 0) - (alpha/gamma)));
     }
```

Se continúa actualizando la matriz hasta la convergencia y finalmente se muestran los resultados obtenidos.

```
M = M.plus((((M.times(beta)).times(v)).times(v.transpose())).times(M));
```

A continuación se muestra la salida que brinda WEKA para $k = 21$, en este algoritmo utilizando la base de datos Wine.

21 vecinos=== Summary ===

Correctly Classified Instances	174	97.7528 %
Incorrectly Classified Instances	4	2.2472 %
Kappa statistic	0.966	
Mean absolute error	0.0647	
Root mean squared error	0.1476	
Relative absolute error	14.7442 %	
Root relative squared error	31.5086 %	
Total Number of Instances	178	

=== Detailed Accuracy By Class ===

	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	1	0.017	0.967	1	0.983	0.999	1
	0.944	0	1	0.944	0.971	0.996	2
	1	0.015	0.96	1	0.98	0.999	3
Weighted Avg.	0.978	0.01	0.978	0.978	0.977	0.998	

=== Confusion Matrix ===

a	b	c	<-- classified as
59	0	0	a = 1
2	67	2	b = 2
0	0	48	c = 3

3.6. Conclusiones parciales del Capítulo

Luego de integrados los algoritmos ITML y LMNN a la herramienta WEKA, se le realizaron los experimentos. Primeramente se ejecutaron en un total de 15 bases de datos y se midió el *accuracy* de ellos en la tarea de clasificación, para luego establecer una comparación entre los mismos y el algoritmo base de WEKA kNN y sus variantes. Luego se aplicó una Prueba de Friedman donde se obtuvo un ranking de los algoritmos, para entonces realizar la prueba post-hoc de Bonferroni-Dunn, llegando a la conclusión que para $\alpha = 0.10$ existen diferencias significativas entre los algoritmos LMNN-CV y el kNN. Es decir, que con los algoritmos implementados se obtienen resultados mucho mejores que aplicando el kNN por defecto de WEKA y sus variantes con un 90% de certeza. Además se implementaron Casos de Estudio para cada algoritmo de modo que se observa con claridad su funcionamiento.

Conclusiones

A modo de conclusión se puede afirmar que se realizó un estudio de los algoritmos del estado del arte relacionados con el aprendizaje de funciones de distancia para problemas de clasificación, sobre todo los que emplean un enfoque supervisado, que permitió reconocer al ITML y al LMNN como los más representativos, utilizados, extendidos y generalizados de todos los existentes. Además han sido los que mejores resultados han alcanzado en la resolución de problemas de clasificación o predicción que emplean un enfoque basado en aprendizaje de funciones de distancia.

Se implementaron dichos algoritmos en el lenguaje de programación Java y se integraron a la herramienta WEKA para que puedan ser utilizados por la comunidad que utiliza esta herramienta. Con esto definitivamente aumentó el poder predictivo y clasificativo de la herramienta lo cual brindará resultados mucho mejores en la solución de numerosos problemas, dándole así respuesta al problema de la investigación. Además, se realizó el planteamiento teórico de la extensión de dichos algoritmos para problemas de predicción con salidas compuestas.

Se realizaron un conjunto de experimentos para comprobar la efectividad de los algoritmos implementados, comparándolos con el algoritmo base de WEKA y sus variantes (kNN, kNN-CV, kNN-DW), desde la medición del *accuracy* con que clasifican hasta la realización de pruebas estadísticas, la Prueba de Friedman y la Prueba de Bonferroni-Dunn, para verificar dichos resultados obteniéndose que, con un 90% decerteza, existen diferencias significativas entre los algoritmos LMNN-CV y el kNN. Esto significa que los algoritmos implementados definitivamente muestran mejores resultados que el existente en la herramienta y sus variantes.

Recomendaciones

Se recomienda implementar la extensión de estos algoritmos para problemas de predicción con salidas compuestas. También aplicar los resultados de esta investigación en el plan de estudio de la asignatura de Inteligencia Artificial.

Se recomienda implementar la extensión de estos algoritmos para problemas de predicción con salidas compuestas integrado a la herramienta MULAN.

Desarrollar estos algoritmos y su extensión a problemas de salidas compuestas basados en un enfoque de margen amplio utilizando un solver de propósito general de SVM (Support Vector Machine).

Bibliografía

1. **Alonso Jimenez, Jose A. y Gutiérrez Naranjo , Miguel A.** Introducción al Aprendizaje Automático. Sevilla, España : s.n., 2000.

2. **ARAUJO, Basilio Sierra**. Aprendizaje Automático: conceptos básicos y avanzados. *Aspectos prácticos utilizando el software Weka. ISBN,* 2006, vol. 10, p. 84-8322.

3. **Dietterich, Thomas G.** *Machine Learning.* Oregon : s.n.

4. **Rodríguez, Ricardo.** Aprendizaje Automático - Machine Learning. Buenos Aires, Argentina : s.n., 2011.

5. **Yang, Liu.** Distance Metric Learning: A Comprehensive Survey. 19 de Mayo de 2006.

6. **© Monografias.com S.A.** monografias.com. [En línea] 19 de Abril de 2015. http://www.monografias.com.

7. **Hakim, Syahir.** Agrobiodiversidad en acción. *http://www.agrobiodiversidad.org.* [En línea] 19 de Abril de 2015.

8. **© 2015 1999, Editorial Ciencias Médicas.** Revista Cubana de Investigaciones Biomédicas. [En línea] 19 de Abril de 2015. http://scielo.sld.cu.

9. **Alcalá-Fdez, J., y otros.** *KEEL: A Software Tool to Assess Evolutionary Algorithms to Data Mining Problems.* 2009, Soft Computing 13, págs. 307-318.

10. **Blockeel, Hendrix.** *Top-down Induction of First Order Logical Decision Trees.* Bélgica : s.n., 1998.

11. **Bellet, A, Habrard, A y Sebban, M.** A Survey on Metric Learning for Feature Vectors and Structured Data. 2013.

12. **Nguyen, Bac, Rivero Pérez, Jorge Luis y Morell, Carlos.** *General issues of distance metric learning. Supervised approach.* Revista Cubana de Ciencias Informáticas.

13. **JIN, Rong; WANG, Shijun; ZHOU, Yang**. Regularized distance metric learning: Theory and algorithm. En *Advances in neural information processing systems*. 2009. p. 862-870.

14. **Yang, Liu.** An Overview of Distance Metric Learning. 28 de Octubre de 2007.

15. **Lanckriet, Gert R.G., y otros.** *Learning the kernel matrix with semi-definite programming.* Berkeley : s.n., 2002.

16. **Elisseeff, A, y otros.** *On jernel-target aligment.* 2002, Neural Information Processing Systems.

17. **Kwok, James T y Tsang, Ivor W.** *Learning with idealized kernels.* 2003, Proceedings of International Conference on Machine Learning.

18. **Xing, E, y otros.***Distance Metric Learning with application to clustering with side information.* 2003, Neural Information Processing Systems.

19. **Hoi, S. C.H., y otros.** *Learning Distance metrics with contextual constraits for image retrieval.* 2006, Proceedings of Computer Vision and Pattern Recognition.

20. **Yang, Liu, Jin, Rong y Sukthankar, Rahul.***Bayesian Active Distance Metric Learning.* 2007, Proceedings iof Uncertainty in Artificial Intelligence.

21. **Davies, Jason V., y otros.** *Information Theoretic Metric Learning.* 2007, Proceedings of the 24th International Conference on Machine Learning.

22. **Tenenbaum, J y Freeman, W.** *Separating Style and Content with Bilinear Models.* Neural Computation, 12, págs. 1247-1283.

23. **Bar-Hillel, A, y otros.***Learning distance functions using equivalence relations.* 2003, Proceedings of International Conference on Machine Learning.

24. **Bar-Hillel, Aharon y Weinshall, Daphna.** Learning distnace functions by coding similarity. New York : s.n., 2007.

25. **Vapnik, V. N. y Wiley, John.** *Statistical Learning Theory.* 1998.

26. **Shawe-Taylor, J y Cristianini, N.** Kernel methods for pattern analysis. *Cambridge University Press.* 2004.

27. **Hastie, T y Tibshirani, R.** *Discriminant adaptive nearest neighbor classification.* 1996, IEEE Trans Pattern Analysis Machine Intelligence 18.

28. **Sugiyama, Masashi.** *Local Fisher discriminant analysis for supervised dimensionality.* 2006, Proceedings of International Conference on Machine Learning.

29. **Goldberg, Jacob, y otros.***Neighbourhood components analysis.* 2005, Neural IInformation Processing Systems.

30. **Weinberg, Kilian, Blitzer, John y Saul, Lawrence.***Distance metric learning for large margin nearest neighbor.* 2006, Advances in Neural Information Processing Systems, 18, págs. 1473-1480.

31. **Yang, Liu, y otros.** *An efficient algorithm for local distance metric learning.* 2006, Proceedings of AAAI.

32. **Kim, T, y otros.***Incremental linear discriminant analysis using sufficient spanning set approximations.* 2007.

33. **De Bie, T, Cristianini, N y Momma, M.***Efficiently learning the metric using side information.* 2003.

34. **Fukuraga, K.** Introduction to statistical pattern recognition. *Boston: Academy Press, Inc Second edition.* 1990.

35. **Fisher, R. A.** *The use of multiple measurements in taxonomic problems.* 1936, Anals of Eugenics, 7, págs. 179-188.

36. **He, X y Niyogi, P.** *Locality Preserving Projections.* 2004, Advances in Neural Information Processing Systems, 16.

37. **Weinberg, Kilian y Saul, Lawrence.** *Distance metric learning for large margin nearest neighbor classification.* 2009, Journal of Machine Learning Research, 10, págs. 207-244.

38. **Globerson, Amir y Roweis, Sam.***Metric Learning by collapsing classes.* 2006.

39. **Salakhutdinov, R y Roweis, Sam.** *Adaptive overrelaxed bound optimization methods.* 2003, Proceedings of International Conference on Machine Learning.

40. **Jolliffe, Ian T.** Principal Component Analysis. s.l. : John Wiley & Sons Hd, 2002.

41. **Cox, T. y Cox, M.** Multidimensional Scaling. *CRC Press.* 2000.

42. **Cox, T y Cox, M.** Multidemsional Scaling. s.l. : Chapman and Hall, 1994.

43. **He, X y Niyogi, Partha.** *Locality Preserving Projections.* 2003, Neural Information Processings Systems.

44. **He, Xiaofei, y otros.***Neighbourhood preserving embedding.* 2005, Proceedings of International Conference on Computer Vision.

45. **Gonzalez, R y Woods, R.** Digital Image Processing. s.l. : Addison-Wesley, 1992.

46. **Common, P.** *Independent Component Analysis _ a new concept?* 1994, Signal Processing, 36, págs. 287-314.

47. **Belkin, M. y Niyogi, P.***Laplacian eigenmaps for dimensionality reduction and data representation.* 2002, Neural Computatio, 16.

48. **Roweis, S. y Saul, L.** *Nonlinear dimensionality reduction by loocally linear embedding.* 2000, Science.

49. **Zha, Hongyuan, y otros.** *Local smoothing for manifold learning.* 2004, Computer Visiona and Pattern Recognition, volume 2, págs. 452-459.

50. **Tenenbaum, J.B., de Silva, V. y Langford, J.C.** *A global geometric framework for nonlinear dimensionality reduction.* 2000, Science,290.

51. **Demsar, Janez.** *Statistical Comparisons of Classifiers over multiple datasets.* 2006, Journal of Machine Learning Research, 7, págs. 1-30.

52. **Vazquez, E. G., y otros.***Repeated measures multiple comparison procedures applied to model selection in neural networks.* 2001, Proceedings of the 6th International Conference on Artificial Intelligence and Neural Networks, págs. 88-95.

53. **Pizzarro, J., Guerrero, E. y Galindo, P. L.** *Multiple comparison procedures applied to model selection.* 2002, Neurocomputing 48, págs. 155-173.

54. **Fisher, R.A.** *Statistical methods and scientific inference (2nd edition).* New York : hafner Publushing Co., 1959.

55. **Hamilton, L.C.** *modern Data Analysis. A first course in applied statistics.* s.l. : Wadsworth, 1990.

56. **Friedman, M.***The use of ranks to avoid the assumption of normality implicit in the analysis of variance.* 1937, Journal of the American Statistical Association, 32, págs. 675-701.

57. **Friedman, M.** *A comparison of alternative tests of significance for the problem of m rankings.* 1940, Annals of Mathematical Statistics, 11, págs. 86-92.

58. **Sheskin, D.J.** *Handbook of parametric and nonparametric statistical procedures.* s.l. : Chapman & Hall/CRC, 2000.

59. **Zar, J.H.** *Biostatistical Analysis (4th edition).* New Jersey : s.n., 1998.

60. **Iman, L.R. y Davenport, J.M.***Approximations of the critical region of the Friedman statistic.* 1980, Communications in Statics, págs. 571-595.

61. **Nemenyi, P.B.** Distribution-free multiple comparisons. 1963.

62. **Dunn, O.J.***Multiple comparisons among means.* 1961, Journal of the American Statistical Association, 56, págs. 52-64.

63. **Holm, S.** *A simple sequentially rejective multiple test procedure.* 1979, Scandinavian Journal of Statistics, 6, págs. 65-70.

64. **Hommel, G.** *A stagewise rejective multiple test procedure based on a modified Bonferroni test.* 1988, Biometrika, 75, págs. 383-386.

65. **Holland, B.***On the application of three modified Bonferroni procedures to pairwise multiple comparisons in balanced repeated measures designs.* 1991, Computational Statistics Quarterly, 6, págs. 219-231.

66. **Spyromitros-Xioufis, Eleftherios, y otros.** *Multi-Label Classification Methods for Multi-Target Regression.* 2014.

67. **Pugelj, Mitja y Dzeroski, Saso.** *Predicting Structured Outputs k- Nearest Neighbours Method.* s.l. : T. Elomaa, J. Hollmén and H. Mannila, 2011.

68. **© 2014 Shogun Toolbox Foundation.** The SHOGUN Machine Learning Toolbox. [En línea] 19 de Abril de 2015. http://www.shogun-toolbox.org.

69. **Sonnenburg, S., y otros.** The SHOGUN Machine Learning Toolbox. *Journal of Machine Learning Research, 11 .* s.l. : Cheng Soon Ong, 2010.

70. **University of Waikato.** Machine Learning Group at the University of Waikato. [En línea] 19 de Abril de 2015. http://www.cs.waikato.ac.nz.

71. **Garner, Stephen R.** WEKA: The Waikato Enviroment for Knowledge Analysis. s.l. : Hamilton.

72. **Tsoumakas, Grigorios, y otros.** Mulan: A Java Library for Multi-Label Learning. *Journal of Machine Learning Research, 12.* s.l. : Cheng Soon Ong, 2011.

73. **Leuven, Universidad Católica de y Stefan, Instituto Jozef.** CLUS. [En línea] 27 de Mayo de 2015. https://dtai.cs.kuleuven.be/clus/.

74. **Systems, Center for machine Learning and Intelligent.** UCI Machine Learning Repository. [En línea] [Citado el: 28 de Mayo de 2015.] http://archive.ics.uci.edu/ml/.

75. **Tukey, J.W.** *Comparing individual means in the analysis of variance.* 1949, Biometrics 5, págs. 99-114.

76. **Dunnett, C.W.***A multiple comparison procedure for comparing several treatments with a control.* 1980, Journal of American Statistical Association 50, págs. 1096-1121.

Anexo A

A continuación se muestran las tablas con los resultados completos para cada algoritmo, en las 15 bases de datos de prueba donde se pueden observar los siguientes aspectos: precisión (grado de concordancia entre un conjunto de resultados), recall (fracción de instancias relevantes que han sido recuperadas), accuracy y *f-measure* (medida de precisión que tiene un test).

Algoritmos	Base de datos	Precision	Recall	F-Measure	CCI
KNN	Australian	0.869	0.868	0.868	86.8116
KNN-CV		0.862	0.861	0.861	86.087
KNN-DW		0.86	0.859	0.86	85.942
ITML		0.871	0.87	0.87	86.9565
ITML-CV		0.863	0.862	0.863	86.2319
LMNN		0.861	0.861	0.861	86.087
LMNN-CV		0.852	0.851	0.851	85.0725
KNN	Banknote	0.999	0.999	0.999	99.8542
KNN-CV		0.999	0.999	0.999	99.8542
KNN-DW		0.999	0.999	0.999	99.9271
ITML		1.0	1.0	1.0	100.0
ITML-CV		1.0	1.0	1.0	100.0
LMNN		1.0	1.0	1.0	100.0
LMNN-CV		1.0	1.0	1.0	100.0
KNN	BUPA	0.647	0.652	0.646	65.2174
KNN-CV		0.633	0.638	0.634	63.6681
KNN-DW		0.677	0.681	0.675	68.1159
ITML		0.715	0.71	0.696	71.0145
ITML-CV		0.687	0.69	0.683	68.9855
LMNN		0.721	0.716	0.703	71.5942
LMNN-CV		0.689	0.693	0.688	69.2754
KNN	Diabetes	0.752	0.757	0.742	75.6510
KNN-CV		0.757	0.762	0.758	76.1719
KNN-DW		0.752	0.758	0.746	75.7813
ITML		0.751	0.755	0.74	75.5208
ITML-CV		0.748	0.755	0.746	75.5208
LMNN		0.747	0.751	0.735	75.1302

LMNN-CV		0.743	0.749	0.744	74.8698
KNN	Ecoli	0.857	0.866	0.86	86.6071
KNN-CV		0.866	0.872	0.866	87.2024
KNN-DW		0.869	0.875	0.87	87.5
ITML		0.858	0.869	0.863	86.9048
ITML-CV		0.857	0.869	0.862	86.9048
LMNN		0.846	0.854	0.847	85.4167
LMNN-CV		0.847	0.854	0.848	85.4167
KNN	Fertility	0.774	0.88	0.824	88.0
KNN-CV		0.848	0.88	0.853	88.0
KNN-DW		0.773	0.87	0.819	87.0
ITML		0.774	0.88	0.824	88.0
ITML-CV		0.902	0.89	0.847	89.0
LMNN		0.774	0.88	0.882	88.0
LMNN-CV		0.869	0.89	0.86	89.0
KNN	Glass	0.709	0.706	0.704	70.5607
KNN-CV		0.718	0.715	0.715	71.4953
KNN-DW		0.73	0.724	0.708	72.4299
ITML		0.728	0.72	0.718	71.9626
ITML-CV		0.737	0.729	0.73	72.8972
LMNN		0.731	0.729	0.729	72.8972
LMNN-CV		0.769	0.766	0.766	76.6355
KNN	Hayes_roth	0.726	0.705	0.707	70.4545
KNN-CV		0.739	0.72	0.722	71.9697
KNN-DW		0.726	0.705	0.707	70.4545
ITML		0.726	0.705	0.707	70.4545
ITML-CV		0.752	0.735	0.738	73.4848
LMNN		0.725	0.705	0.708	70.4545
LMNN-CV		0.749	0.735	0.738	73.4848
KNN	Ionosphere	0.879	0.866	0.859	86.6097
KNN-CV		0.907	0.906	0.904	90.5983
KNN-DW		0.879	0.866	0.859	86.6097
ITML		0.874	0.866	0.861	86.6097
ITML-CV		0.907	0.906	0.904	90.5983
LMNN		0.897	0.892	0.888	89.1738

LMNN-CV		0.92	0.92	0.919	92.0228
KNN	Iris	0.968	0.967	0.967	96.6667
KNN-CV		0.96	0.96	0.96	96.0
KNN-DW		0.968	0.967	0.967	96.6667
ITML		0.968	0.967	0.967	96.6667
ITML-CV		0.974	0.973	0.973	97.3333
LMNN		0.968	0.967	0.967	96.6667
LMNN-CV		0.974	0.973	0.973	97.3333
KNN	Seeds	0.943	0.943	0.943	94.2857
KNN-CV		0.943	0.943	0.943	94.9857
KNN-DW		0.943	0.943	0.943	94.2857
ITML		0.943	0.943	0.943	94.2857
ITML-CV		0.943	0.943	0.943	94.2857
LMNN		0.949	0.948	0.947	94.7619
LMNN-CV		0.949	0.948	0.947	94.7619
KNN	Thoracic	0.815	0.853	0.798	85.3191
KNN-CV		0.815	0.853	0.798	85.3191
KNN-DW		0.803	0.851	0.794	85.1064
ITML		0.724	0.851	0.783	85.1064
ITML-CV		0.775	0.849	0.785	84.8936
LMNN		0.8	0.851	0.787	85.1064
LMNN-CV		0.775	0.849	0.785	85.1064
KNN	Vertebral _column	0.824	0.826	0.824	82.5806
KNN-CV		0.828	0.823	0.824	82.2581
KNN-DW		0.838	0.839	0.838	83.8710
ITML		0.861	0.858	0.859	85.8065
ITML-CV		0.85	0.848	0.849	84.8387
LMNN		0.861	0.858	0.859	85.8065
LMNN-CV		0.863	0.865	0.863	86.4516
KNN	Vehicle	0.697	0.714	0.702	71.3948
KNN-CV		0.707	0.721	0.703	72.104
KNN-DW		0.715	0.727	0.72	72.695
ITML		0.696	0.713	0.701	71.2766
ITML-CV		0.707	0.721	0.703	72.104

LMNN		0.694	0.712	0.699	71.1584
LMNN-CV		0.707	0.721	0.703	72.104
KNN	Yeast	0.59	0.595	0.584	59.5013
KNN-CV		0.586	0.59	0.58	59.0296
KNN-DW		0.561	0.568	0.563	56.8059
ITML		0.596	0.601	0.592	60.1752
ITML-CV		0.597	0.602	0.592	60.1752
LMNN		0.595	0.601	0.59	60.1078
LMNN-CV		0.594	0.599	0.589	59.9057

Anexo B

Los resultados de la presente investigación se presentaron en la XIII Jornada Científica Estudiantil en la Universidad de las Ciencias Informáticas, Cuba, donde se obtuvo categoría de Relevante a nivel de Universidad.