SEP TECNOLÓGICO NACIONAL DE MÉXICO

INSTITUTO TECNOLÓGICO DE TIJUANA

DEPARTAMENTO DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA

ANÁLISIS DE APRENDIZAJE TRANSFERIDO EN UN SISTEMA DE PROGRAMACIÓN GENÉTICA

TRABAJO DE TESIS PRESENTADO POR:
ING. JOEL LEE NATION MORALES

PARA OBTENER EL GRADO DE:
MAESTRO EN CIENCIAS DE LA INGENIERÍA

DIRECTOR DE TESIS:
DR. LEONARDO TRUJILLO REYES

CO-DIRECTOR DE TESIS:
DR. LUIS MUÑOZ DELGADO

TIJUANA, BAJA CALIFORNIA, MÉXICO.
DICIEMBRE DE 2021

Tijuana Baja California, 25/Noviembre/2021
Asunto: Autorización de impresión de trabajo de tesis

DR. GUADALUPE HERNÁNDEZ ESCOBEDO
DIVISIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN
PRESENTE

En lo referente al trabajo de tesis, **"Análisis de aprendizaje transferido en un sistema de programación genética"**, presentado por el **Ing. Joel Lee Nation Morales** alumno del programa de Maestría en Ciencias de la Ingeniería, con número de control **M1921008** informamos a usted que después de una minuciosa revisión e intercambio de opiniones, los miembros del comité manifiestan **APROBAR LA TESIS**, en virtud de que satisface los requisitos señalados por las disposiciones reglamentarias, por lo que se autoriza al interesado para que proceda de inmediato a la impresión del mismo.

A T E N T A M E N T E
Excelencia en Educación Tecnológica®
Por una juventud integrada al desarrollo de México®

Dr. Leonardo Trujillo Reyes
DIRECTOR DE TESIS

Dr. Luis Muñoz Delgado
CO-DIRECTOR DE TESIS

Dr. José Ricardo Cárdenas Valdez
MIEMBRO DEL COMITÉ

Dr. Daniel Eduardo Hernández Morales
MIEMBRO DEL COMITÉ

INSTITUTO TECNOLOGICO DE TIJUANA
DEPTO. DE INGENIERIA ELECTRICA Y ELECTRONICA
POSGRADO EN CIENCIAS DE LA INGENIERIA

ccp. Archivo
Dr. José Ricardo Cárdenas Valdez – Coordinador Académico de la Maestría en Ciencias de la Ingeniería.

Tijuana, Baja California,
30/noviembre/2021
OFICIO No. 128/DEPI/2021
Asunto: Autorización de Impresión de Tesis

MARIBEL GUERRERO LUIS
JEFA DEL DEPARTAMENTO DE SERVICIOS ESCOLARES
PRESENTE

En lo referente al trabajo de tesis, "Análisis de aprendizaje transferido en un sistema de programación genética". Presentado por C. **Joel Lee Nation Morales**, alumno de la Maestría en Ciencias de la Ingeniería con número de control **M1921008**; informo a usted que a solicitud del comité de tutorial, tengo a bien **Autorizar la impresión de Tesis**, atendiendo las disposiciones de los Lineamientos para la Operación de Estudios de Posgrado del Tecnológico Nacional de México.

Sin más por el momento le envío un cordial saludo.

A T E N T A M E N T E
Excelencia en Educación Tecnológica®
Por una juventud integrada al desarrollo de México®

EDUCACIÓN
INSTITUTO TECNOLÓGICO DE TIJUANA
DIVISIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN

GUADALUPE HERNANDEZ ESCOBEDO
JEFE DE DIVISIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN

ccp. Archivo
GHE/lap

CARTA DE CESION DE DERECHOS

En la ciudad de Tijuana, Baja California, el día **22** del mes de **noviembre** del año **2021**, el que suscribe **Joel Lee Nation Morales**, con número de control **M1921008**, alumno de **Maestría** del programa de Posgrado en Ciencias de la Ingeniería, manifiesta que es autor intelectual del presente trabajo de Tesis bajo la dirección del **Dr. Leonardo Trujillo Reyes**, cede los derechos del trabajo titulado **'Análisis de aprendizaje transferido en un sistema de programación genética'** al Tecnológico Nacional de México para su difusión, con fines académicos y de investigación en la comunidad estudiantil y científica del país.

Los usuarios de la información no deben reproducir el contenido textual, gráficas, código, fórmulas o datos del trabajo sin permiso expreso del autor o director del trabajo. Este debe ser obtenido escribiendo a cualquiera de las siguientes direcciones de correo electrónico **joel.nation19@tectijuana.edu.mx** y **leonardo.trujillo@tectijuana.edu.mx** o bien, dirigirse a las instalaciones del Instituto Tecnológico de Tijuana en Calzada del Tecnológico S/N Esq. Av. Castillo de Chapultepec y calle Cuauhtemotzin, Fracc. Tomás Aquino C.P. 22414, Tijuana, Baja California, conmutador 664-6078400.

Si se otorga el permiso, el usuario deberá dar el agradecimiento correspondiente y citar la fuente del mismo como lo indique el autor intelectual o el director del trabajo de Tesis.

A T E N T A M E N T E

Joel Lee Nation Morales
ALUMNO DEL POSGRADO EN CIENCIAS DE LA INGENIERÍA

RESUMEN

ANÁLISIS DE APRENDIZAJE TRANSFERIDO EN UN SISTEMA DE PROGRAMACIÓN GENÉTICA

Ing. Joel Lee Nation Morales

El aprendizaje de transferencia (TL, por sus siglas en inglés de *Transfer Learning*) es el proceso mediante el cual se utiliza el conocimiento obtenido por un modelo generado mediante aprendizaje máquina en un problema (problema donador) para mejorar o simplificar el proceso de aprendizaje en un segundo problema (problema receptor). Si bien TL ha logrado resultados prometedores en Deep Learning, apenas se ha estudiado en algoritmos evolutivos y programación genética (GP, por sus siglas en inglés de *Genetic Programming*). Una dificultad que limita la utilización de TL en GP o algún otro método evolutivo es la dificultad para predecir cuando tendrá éxito, esto sigue siendo una pregunta abierta. Este trabajo presenta un primer intento de determinar cuándo dos problemas podrían ser compatibles para que TL tenga éxito entre ellos. Esto se hace analizando la estructura del espacio de características de cada problema, extraído por el método DeepInsight. Se utilizan resultados previos en TL para tareas de clasificación con GP, de modo que cada posible par de problemas se categorizados en uno de dos grupos. El primer grupo son problemas compatibles con TL que incluye todos los pares de problemas en los que TL tuvo mucho éxito, y el segundo son los problemas de TL no compatibles en los que TL tuvo un desempeño inferior al de los métodos de referencia. Los resultados muestran que se pudieron rechazar las hipótesis propuestas con un rango de confianza de 94,50 % para $H0_1$ y 96,27 % para $H0_2$, esto significa que, es posible distinguir entre diferentes grupos de pares de problemas alineando sus respectivos espacios de características representativos y calcular una distancia entre ellas.

Palabras clave: Clasificación, Programación genética, Aprendizaje Transferido, DeepInsight, Inducción constructiva.

ABSTRACT

Transfer Learning (TL) is the process by which the knowledge obtained in a model generated by machine learning is used in a problem (source problem) to improve or simplify the learning process in a second problem (target problem). While TL has achieved promising results in deep learning, it has hardly been studied in evolutionary algorithms and genetic programming (GP). One difficulty that limits the use of TL in GP or some other evolutionary method is the difficulty of predicting when it will be successful, this remains an open question. This work presents a first attempt to determine when two problems could be compatible for TL to work between them. This is done by analyzing the structure of the feature space for each problem, extracted by the DeepInsight method. Previous results in TL are used for classification tasks with GP, so that each possible pair of problems is categorized into one of two groups. The first group are TL compatible problems that includes all pairs of problems where TL was very successful, and the second group is the non-compatible TL problems where TL underperformed the reference methods. The results show that the proposed hypotheses could be rejected with a confidence range of 94,50 % for $H0_1$ and 96,27 % for $H0_2$, this means that, it is possible to distinguish between different groups of pairs of problems by aligning their respective feature spaces representations and calculate a distance between them.

keywords: Classification, Genetic programming, Transferred Learning, DeepInsight, Constructive induction.

DEDICATORIA

A Krizia.

AGRADECIMIENTOS

A mi madre por ayudarme todas las mañanas a estar listo, a mi padre por apoyarme cuando lo necesitaba. A mis compañeros por las buenas risas que bajaban el estrés, a mis asesores por su paciencia y no darse por vencidos para guiarme por el mejor camino.

Y a mi novia por darme la motivación de seguir adelante.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

INTRODUCCIÓN

1.1. ANTECEDENTES

El aprendizaje máquina (ML, por sus siglas en inglés de *Machine Learning*) es el estudio de algoritmos informáticos que pueden mejorar automáticamente a través del aprendizaje y mediante el uso de datos [1]. Existen varios métodos de aprendizaje, pero en general podemos categorizarlos en supervisados y no supervisados. El aprendizaje supervisado utiliza datos de entrenamiento para enseñar a un modelo a producir el resultado deseado, estos datos de entrenamiento incluyen etiquetas, que permiten que el modelo aprenda con el tiempo. Caso contrario, en el aprendizaje no supervisado estos datos de entrenamiento no tienen etiquetas. El aprendizaje supervisado se puede subdividir en dos tipos: clasificación y regresión.

En clasificación, el objetivo es predecir las etiquetas de nuevos registros, con base en observaciones de los datos de entrenamiento. Dependiendo de cuántas etiquetas existan en los datos, se puede decir que la clasificación es binaria o multiclase. Cuando sólo existen dos clases, se trata de clasificación binaria; si existen más de dos clases, entonces es clasificación multiclase. Mientras tanto, la regresión se trata de un proceso estadístico donde el modelo intenta predecir un valor continuo mediante la relación entre variables dependientes e independientes.

El resultado de este entrenamiento genera un modelo con el cual es posible predecir datos nuevos y asignarles una etiqueta, siempre y cuando estos pertenezcan al mismo dominio del problema inicial. Es decir, que puede generalizar los datos de entrenamiento a cualquier dato del problema en cuestión. Desafortunadamente, durante el aprendizaje supervisado se puede producir un "sobreajuste" o "subajuste" [2], lo cual no permite que el modelo pueda generalizar los datos y genere errores al momento de realizar predicciones en nuevos datos.

El subajuste se produce cuando el modelo sólo se enfoca en aprender los casos particulares que se muestran en los datos de entrenamiento, lo cual provoca que sea incapaz de etiquetar nuevos datos de entrada. En algunos casos estos datos de entrenamiento introducen muestras atípicas (o anómalas), con ruido en algún rasgo, o muestras que no son del todo representativas. En cambio, el sobreajuste se da cuando sobre-entrenamos el modelo, ya que éste sólo estará considerando como

válidos los datos idénticos a los datos de entrenamiento, haciéndolo incapaz de distinguir otros datos como fiables si estos se salen un poco de los rangos ya preestablecidos. Estos errores ocasionan que sea necesario volver a entrenar el modelo; reentrenar un modelo es costoso computacionalmente, ya que éste requiere de un reajuste.

En algunos escenarios, es intuitivo suponer que debería ser posible aprender en una tarea, y aplicar lo aprendido en otra. Recientemente, esta idea general se ha unido a lo que se conoce como adaptación de dominio [3], lo que resulta en el área conocida como aprendizaje por transferencia (TL, por sus siglas en inglés *Transfer Learning*) [4], [5]. TL se puede caracterizar de la siguiente manera: un algoritmo de aprendizaje se aplica a un problema (problema donador), y algunos conocimientos o elementos de lo aprendido se utilizan para obtener las solución de un segundo problema (problema receptor).

1.2. PLANTEAMIENTO DEL PROBLEMA

TL aborda varios de los problemas con ML mencionados anteriormente. Primero, permite la reutilización de modelos aprendidos previamente, extendiendo su utilidad más allá del problema original que se pretendía resolver originalmente. En segundo lugar, tiene un impacto en la cantidad de datos y cálculos que se requieren para resolver una nueva tarea, ya que gran parte del *aprendizaje* ya se realizó en el problema donador. Esto ha generado un mayor interés en una variedad de dominios, incluido el procesamiento del lenguaje natural [6] y visión por computadora [7], lo que lo hace bastante útil para abordar nuevos problemas rápidamente con redes neuronales profundas (DNN, por sus siglas en inglés de *Deep Neural Network*) [8], [9].

Sin embargo, una dificultad que existe cuando se utiliza TL para resolver un nuevo problema, es que no hay una forma general de determinar cuándo TL podría tener éxito, o de proporcionar una interpretación objetiva de por qué éste podría ser el caso [10], [11]. Una manera empírica de aplicar TL, es que los dominios del problema donador y receptor deben ser similares, pero esto se expresa principalmente como una característica subjetiva de la configuración de TL que como una propiedad objetivamente verificable [4]. Por ejemplo, en el caso de la visión por computadora, esto a veces se puede argumentar de manera convincente, ya que es natural que un experto humano *vea* las similitudes entre detectar un tipo de enfermedad en un tipo de imagen u otro [8].

1.3. JUSTIFICACIÓN

Por otra parte, existe una cantidad limitada de literatura sobre el desarrollo de un enfoque basado en principios para predecir el éxito o el fracaso de un sistema de TL [10]. Este trabajo presenta un análisis de lo que se conoce como compatibilidad de TL entre un problema de donador y receptor, analizando los resultados presentados en [11]. Por ello la principal motivación de esta tesis es la siguiente pregunta:

¿Cómo se relaciona la compatibilidad de TL con la similitud del espacio de características entre los problemas donador y receptor?

Nuestro objetivo es determinar cuándo dos problemas podrían ser candidatos adecuados para que se produzca un TL exitoso, y proporcionar una explicación posible de por qué es así. Para esto, proponemos analizar el espacio de características de ambos problemas usando una representación estructurada bidimensional [12] (una imagen 2D) en la que se puede definir la similitud del problema.

Se plantea la hipótesis de que, cuando existe una correspondencia adecuada entre los espacios de características de ambos problemas, se puede esperar un alto nivel de compatibilidad con TL, y viceversa. El enfoque propuesto podría usarse para determinar cuándo TL podría ser aplicable a los modelos evolucionados de GP.

Por lo tanto, ampliamos el trabajo presentado en [11], y presentamos el primer estudio que muestra cómo se puede medir la similitud entre problemas, para determinar el nivel de éxito alcanzado al utilizar estos problemas para realizar TL.

1.4. OBJETIVOS

1.4.1 OBJETIVO GENERAL

Analizar los casos de éxito y fracaso que se obtuvieron en [11] al utilizar TL en un sistema de GP, para determinar cuándo dos problemas podrían ser candidatos adecuados para lograr un TL exitoso. Además, proponer una metodología que pueda medir la similitud entre problemas, basada en el espacio de características de ambos problemas.

1.4.2 OBJETIVOS ESPECÍFICOS

- Analizar la similitud entre los problemas de prueba utilizados para evaluar el TL dentro de GP.
- Desarrollar una metodología que permita analizar el espacio de características de los problemas de prueba.

- Validar la metodología propuesta utilizando los resultados generados por [11].

1.5. ORGANIZACIÓN DE LA TESIS

En el Capítulo 2 se presenta una introducción a GP, se describe el funcionamiento, condiciones y decisiones del usuario que debe tomar en consideración al utilizar este algoritmo. Se describe el funcionamiento de M3GP, asimismo su funcionamiento en un entorno de GP y termina con una comparativa de M3GP ante otros trabajos.

En el Capítulo 3 se presenta TL, que se debe considerar para utilizarlo en un problema y cómo es que funciona. Se habla de cómo es utilizado dentro de aprendizaje máquina, su aplicación en otros trabajos donde se utiliza GP y las limitaciones que se tiene al utilizar este método.

En el Capítulo 4 se presenta el caso de estudio del cual se obtuvieron los datos a analizar, se exponen los resultados notables, se explica la metodología de TL dentro de M3GP y, por último, se plantea la explicación de dónde comienza el análisis de los datos y conceptos que se utilizaran a lo largo de esta tesis.

En el Capítulo 5 se presenta el funcionamiento de la herramienta DeepInsight, se describe la metodología para generar imágenes a partir de unos datos de entrada, de las limitaciones y consideraciones que se deben tomar al generar las imágenes. También se presenta cómo será el análisis de las imágenes y cómo se interpretan.

En el Capítulo 6 se presenta la metodología desarrollada para analizar los resultados de la herramienta DeepInsight. A su vez se muestra el análisis entre problemas con transferencias positivas y negativas de aprendizaje. Además se describe cómo es que logra una medición numérica utilizando el método iterativo del punto más cercano.

Por último, en el Capítulo 7 se encuentran las conclusiones y trabajo a futuro.

Capítulo 2

PROGRAMACIÓN GENÉTICA

Dentro de la inteligencia artificial se encuentran los algoritmos evolutivos (AE), que son procedimientos de búsqueda y optimización que tienen sus orígenes e inspiración en el mundo biológico, los cuales cuentan con múltiples utilidades prácticas que contribuyen a encontrar soluciones de forma rápida a procesos y problemas variados [13].

Se caracterizan por imitar procesos adaptativos de los sistemas naturales y se basan en la supervivencia del mejor individuo, siendo un individuo una solución potencial del problema que se implementa como una estructura de datos. Trabajan con una población de individuos, que representan soluciones candidatas a un problema. Esta población se somete a ciertas transformaciones y después a un proceso de selección, que favorece a los mejores. Cada ciclo de transformación y selección constituye una generación, de forma que después de cierto número de generaciones se espera que el mejor individuo de la población esté cerca de la solución buscada. Los AE combinan la búsqueda aleatoria, dada por las transformaciones de la población, con una búsqueda dirigida dada por la selección.

La característica fundamental de los AE radica en los métodos de generación de soluciones: se parte de un conjunto de soluciones iniciales y se van empleando un conjunto de operadores de búsqueda para ir refinando la solución final. Para realizar dicho refinamiento de las soluciones, se pueden utilizar técnicas clásicas como el seguimiento del gradiente (*Hill Climbing*) complementadas con mecanismos biológicos de exploración: población de soluciones, operadores genéticos.

Una técnica que engloba todos estos paradigmas es GP, ya que tiene como objetivo generar programas informáticos a través de la búsqueda evolutiva, también llamada inducción automática de programas. GP puede entenderse como un algoritmo de aprendizaje supervisado que intenta construir una expresión válida utilizando un conjunto finito de funciones básicas y variables de entrada, guiadas por una función objetivo dependiente del dominio. En las siguientes secciones se detallará su uso y funcionamiento.

2.1. ALGORITMO BÁSICO DE GP

GP es una técnica para crear programas de computadora de forma automática, donde se utilizan los principios de la evolución Darwiniana [14]. Se inicia con una población de programas de computadora (individuos) y sólo los mejores pasarán su información, utilizando los principios básicos de cruce, reproducción y mutación, con esto se asegura que cada generación será mejor a la previa. Con el tiempo, los mejores individuos sobrevivirán y eventualmente evolucionarán para tener éxito en el ambiente dado.

El objetivo de GP, es poder darle un problema a la computadora y que realice un programa para resolverlo, sin tener que escribir el código explícitamente.

2.1.1 CREACIÓN DE UN INDIVIDUO

A cada programa generado se le conoce como individuo, esta se representa con una estructura de árbol que se forma de dos componentes esenciales: funciones y terminales, que actuarán como los genes del individuo.

Las funciones serán los nodos, las cuales serán las operaciones matemáticas que realizará el programa, mientras que las terminales serán las hojas y representarán componentes como variables y constantes. El nodo le dirá al programa qué instrucciones debe realizar y las terminales indicarán qué argumentos usar para cada instrucción.

Los genes disponibles para la programación genética se deben seleccionar o crear por el usuario. Este es un paso importante, ya que una selección deficiente puede afectar al sistema y hacerlo incapaz de desarrollar una solución. Un programa simple sería como se muestra en la Figura 2.1.

En formas más avanzadas de programación genética los individuos pueden ser representados de formas diferentes como en presentaciones lineales, en grafos y en pilas [14]. En este caso, la representación utilizada es un conjunto de árboles agrupados bajo un nodo especial llamado raíz, llamaremos ramas a estos sub-árboles. El número y el tipo de las ramas en un programa, junto con otras características de la estructura de las ramas, forman la arquitectura del programa.

2.1.2 GENERANDO UNA POBLACIÓN

A un conjunto de individuos se le llama población, una vez seleccionados los genes se crea una población aleatoria de individuos, la población inicial es generada utilizando una de tres técnicas: *grow*, *full* y *ramped-half-and-half*.

Grow: Se crea aleatoriamente un individuo a la vez, el cual puede ser un árbol de cualquier profundidad hasta un máximo especificado, m. Comenzando desde

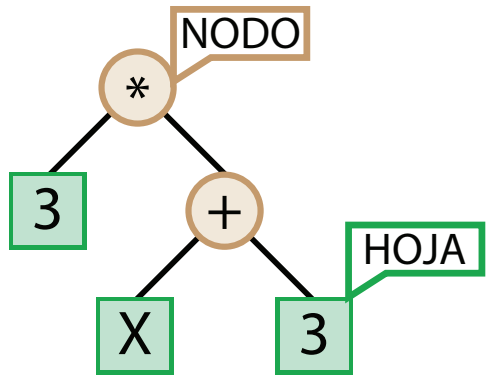


Figura 2.1: Ejemplo de individuo: $3(x+3)$. El cual expresado en notación de árbol. Donde en los nodos se encuentran la operación matemática a realizar y en las hojas las variables que se utilizarán.

la raíz del árbol, cada nodo se elige aleatoriamente como una función o terminal. Si el nodo es una terminal, se elige aleatoriamente. Si el nodo es una función, se elige una función aleatoria, y ese nodo recibe un número de hijos igual a la aridad (número de argumentos) de la función. Para cada uno de los hijos de la función, el algoritmo comienza de nuevo, a menos que el hijo llegue a la profundidad m, lo cual causa que se convierta en una terminal seleccionada aleatoriamente.

Este método no garantiza individuos de cierta profundidad (aunque no serán más profundos que m). En cambio, proporciona una variedad de estructuras en toda la población e incluso puede producir individuos que contienen sólo un nodo. Esos individuos se reproducen rápidamente si el problema no es trivial, por lo que realmente no tienen mucho valor.

Full: El método Full es muy similar al método de Grow, excepto que los terminales tienen una profundidad determinada, lo cual no garantiza una cantidad específica de nodos en un individuo. Este método requiere una profundidad final, d. Cada nodo a partir de la raíz con una profundidad menor que d, se convierte en una función seleccionada al azar. Si el nodo tiene una profundidad igual a d, el nodo se convierte en una terminal seleccionada al azar.

Ramped-half-and-half: Este método se utiliza para aumentar la variedad de la estructura de los individuos, siendo un punto medio de las dos técnicas anteriores, donde sólo se especifica una profundidad máxima, md. El método genera una población con un buen rango de individuos de tamaño y estructura aleatoria. La población está dividida en partes: un total de $md-1$. Mitad de la población se produce usando el método Grow, mientras la otra mitad se produce utilizando el

método Full.

2.1.3 EVALUACIÓN DE LOS INDIVIDUOS

Una vez creada la población, se requiere saber qué tan buenos son los individuos para resolver problemas. Para esto se requiere una medida de aptitud (mejor conocida en inglés como *Fitness*). Para evaluar la aptitud se utiliza una función matemática llamada *función de aptitud* la cual puede evaluar cualquier aspecto del individuo que se quiera mejorar y por lo tanto depende del problema a resolver. Por ejemplo, para problemas de Regresión Lineal se puede utilizar el Error cuadrático medio (RMSE, por sus siglas en inglés de *Root Mean Square Error*) y para problemas de Clasificación, la exactitud (mejor conocido en inglés como *Accuracy*).

RMSE está dado por:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(x_i - y_i)^2}{n}},$$

donde x_i es el valor predicho de la generación, y_i el valor actual de la generación, n el número total de generaciones.

Accuracy está dado por:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \times 100,$$

donde TP son los positivos correctamente clasificados, TN los negativos correctamente clasificados, FP los positivos falsos y FN los negativos falsos clasificados por el algoritmo.

2.1.4 OPERACIONES GENÉTICAS

Cuando se tiene el valor de aptitud de cada individuo, comienza el proceso de evolución para crear una nueva generación de individuos, los cuales se forman utilizando los métodos de: cruce y mutación, a esto se le conoce como operaciones genéticas (OG).

Cruce: es la recombinación de dos individuos de los cuales se eligen partes al azar de su estructura para crear un nuevo individuo, las partes sobrantes son eliminadas, como se muestra en la Figura 2.2.

Mutación: se da cuando parte de la estructura de un individuo es intercambiada aleatoriamente por un sub-árbol generado al azar, el punto de mutación puede ser un nodo o una hoja, como se muestra en la Figura 2.3.

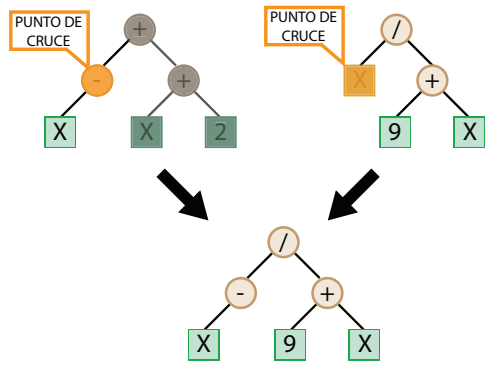


Figura 2.2: Ejemplo de cruce entre dos individuos: $(-x)+(x+2)$ y $x/(9+x)$ forman $(-x)/(9+x)$

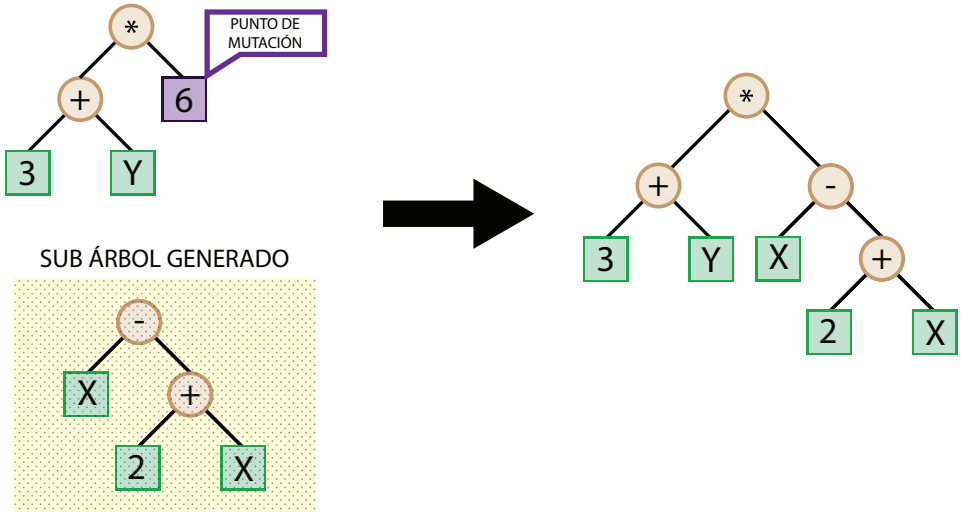


Figura 2.3: Ejemplo de mutación de un individuo: $6(3+y)$ con el punto de mutación en la hoja 6, donde muta a la expresión $(3+y)(x-(2+x))$.

Las OG en GP se aplican a individuos que se seleccionan probabilísticamente en función de su aptitud, es decir, los mejores individuos tienen más probabilidades de ser seleccionados que los individuos inferiores. El método más comúnmente empleado para seleccionar individuos en GP es la selección por torneo [14], sin embargo, se puede utilizar cualquier mecanismo de selección de AE estándar, por ejemplo, la aptitud. En la selección del torneo, se elige al azar un número de in-

dividuos de la población. Estos se comparan entre sí y se elige al mejor de ellos para que sea el padre. Al hacer cruce, se necesitan dos padres y, por lo tanto, se realizan dos torneos de selección. Tenga en cuenta que la selección de torneos sólo considera qué programa es mejor que otro, no hace falta saber cuán mejor. Estas operaciones se repetirán un número de veces hasta que se cumpla la condición o criterio de terminación del algoritmo.

2.1.5 CRITERIO DE TERMINACIÓN

El criterio de terminación puede incluir un número máximo de generaciones a ejecutar, así como una meta basada en el éxito específico del problema o meta del usuario.

2.1.6 DECISIONES DE USUARIO

Antes de comenzar el ciclo evolutivo de GP, es importante que el usuario contemple los genes que se necesitan seleccionar y crear. La selección de genes es importante para realizar una tarea; si no es posible resolver el problema con los genes disponibles, no hay esperanza para que el sistema desarrolle una solución.

Por eso es importante incluir todas las funciones y terminales necesarias, sin embargo, seleccionar más que el conjunto de genes mínimo requerido plantea un problema, ya que los genes que no son útiles quedarán fuera de la población.

El usuario debe especificar el tamaño de los individuos que se van a generar, un límite demasiado grande puede eliminar todas las posibilidades de desarrollar un individuo aceptable. Sin embargo, no restringir suficientemente el espacio de búsqueda tiene sus propios problemas y pueden ser creados individuos ineptos. Otros parámetros a considerar son:

Tamaño de población: Una población grande permite una mayor exploración del espacio problemático en cada generación y aumenta las posibilidades de desarrollar una solución, sin embargo, esto requiere más recursos de procesamiento. La otra opción es tener una población pequeña que se ejecuta por más generaciones y explorar el mismo espacio, lo cual se traduce a un tiempo más largo de ejecución. En general, el tamaño de población puede variar dependiendo del problema que se quiera solucionar.

Número de generaciones: El proceso evolutivo necesita tiempo, cuanto mayor sea el número de generaciones, mayores serán las posibilidades de desarrollar una solución. Sin embargo, una mayor evolución de una población no garantiza que se encuentre una solución. En algunos casos puede ser mejor comenzar de nuevo el ciclo evolutivo con una población inicial diferente. Se puede dar el caso en que después de un número de generaciones definidas por el usuario, los individuos no

se desarrollen lo suficiente para una solución (a esto se le conoce como trampa de óptimo local), por el cual el proceso debe detenerse y comenzar de nuevo.

Probabilidad de OG: Independientemente del tamaño de la población, se debe dejar claro qué porcentajes de reproducción, mutación y cruce se les aplicará. Esta decisión afectará cómo cada nueva población se desarrollará para encontrar una solución.

Criterio de terminación: Éste puede ser determinado por un número *n* de generaciones, por un tiempo limite, cuando la función de *aptitud* ya no mejore una cierta cantidad de generaciones, o definirse por otro parámetro específico del usuario.

Ciclo evolutivo

Una vez que el usuario ha definido las funciones y terminales, el método para crear una población y ajustado el porcentaje de las operaciones genéticas de su programación genética, comienza el ciclo evolutivo como se muestra en la Figura 2.4, el cual se detendrá cuando llegue a un número de generaciones máximas o se encuentre una solución deseada.

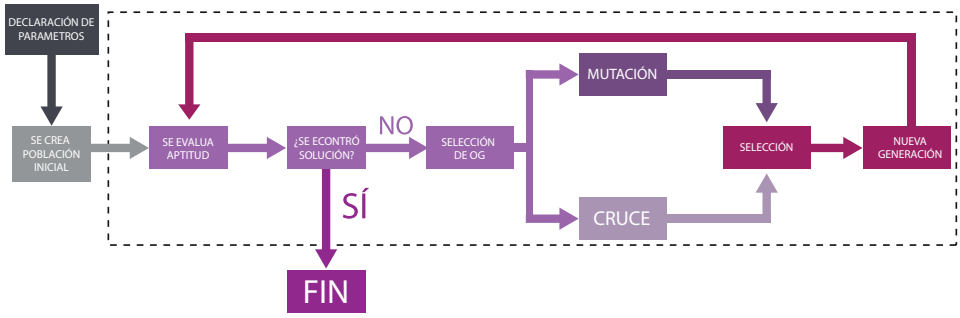


Figura 2.4: Ciclo evolutivo básico para un algoritmo de GP estándar.

2.2. M3GP

2.2.1 Inducción constructiva de características

Cuando las características sin procesar de un problema no son apropiadas, se requiere un paso de construcción o transformación de características. En otras palabras, dado un conjunto de características de entrada, se utiliza un modelo de

transformación para generar un nuevo conjunto de características con un espacio más simple o complejo, de manera que se pueda llevar a cabo el aprendizaje. Básicamente, existen dos formas de construcción de características: compilación de características e Inducción constructiva de características (CI, por sus siglas en inglés de *Constructive Induction*) [15].

La compilación de características consiste en reescribir la representación original en una nueva forma, generalmente más compacta, de modo que las características resultantes sean lógicamente equivalentes a las originales. Por el contrario, CI busca mejorar la precisión de una representación, incluso a expensas de la eficiencia, aumentando a veces la dimensionalidad del espacio de características, utilizando diferentes formas de construcción, modificación y eliminación de características. CI puede aplicarse con GP en algo que se conoce comúnmente como el enfoque basado en *wrappers* para la construcción de características [16], el funcionamiento se puede observar en la Figura 2.5.

Figura 2.5: Enfoque basado en *wrappers* para CI con GP .

2.2.2 FUNCIONAMIENTO DE M3GP

GP se ha utilizado para resolver muchos problemas difíciles en varios dominios, como se puede apreciar en [17]. Los problemas en los que tiene mejor rendimiento GP es en problemas de aprendizaje automático supervisado, en particular la regresión simbólica y la clasificación de datos. Gracias a esto se ha podido implementar en varios problemas de referencia y escenarios del mundo real, sin embargo, presenta un bajo rendimiento en problemas multiclase.

Para contrarrestar esta deficiencia utilizando un método de CI, se propuso un algoritmo llamado *Programación Genética Multidimensional Multiclases* (M2GP, por sus siglas en inglés de *Multidimensional Multiclass Genetic Programming*) [18]. Este algoritmo resuelve efectivamente problemas multiclase, realizando una transformación multidimensional de los datos de entrada. El algoritmo M2GP utiliza un número fijo de nuevas dimensiones de características d, que debe elegirse

y fijarse antes de que comience la ejecución del ciclo evolutivo, por lo tanto, antes de iniciar un ciclo evolutivo M2GP ejecuta un proceso en el cual inicializa de manera iterativa diferentes poblaciones con dimensiones en aumento y verifica cuál de estas poblaciones iniciales tiene un mejor rendimiento comenzando con $d = 1$. Este procedimiento agrega una dimensión e inicializa una población nueva cada que mejore el rendimiento y se detiene cuando el rendimiento baja, lo cual da a entender que ha encontrado el mejor valor d. Esto significa que M2GP es incapaz de agregar o eliminar dimensiones durante el ciclo evolutivo una vez que se seleccione un valor d.

Sin embargo, este proceso de dejar fijo el número de dimensiones al inicio de un ciclo evolutivo puede evitar que el algoritmo encuentre mejores soluciones durante la búsqueda, unas que pueden usar un número diferente de dimensiones. Para eliminar esta incertidumbre en [19] proponen una mejoría del algoritmo M2GP llamado M3GP, donde ahora el algoritmo puede evolucionar una población que puede contener individuos de diferentes dimensiones. Aquí los OG pueden agregar o eliminar dimensiones, y se supone que la selección será suficiente para descartar los peores individuos y mantener los mejores en la población.

2.2.3 Operaciones Geneticas con M3GP

M3GP es un enfoque de evolución estándar utilizando una representación multi-árbol, buscando transformaciones de la forma $k : \mathbb{R}^p \rightarrow \mathbb{R}^d$ con $p, d \in \mathbb{N}$ asociando las características p del problema a un nuevo espacio de características de tamaño d, como se muestra en la Figura 2.6.

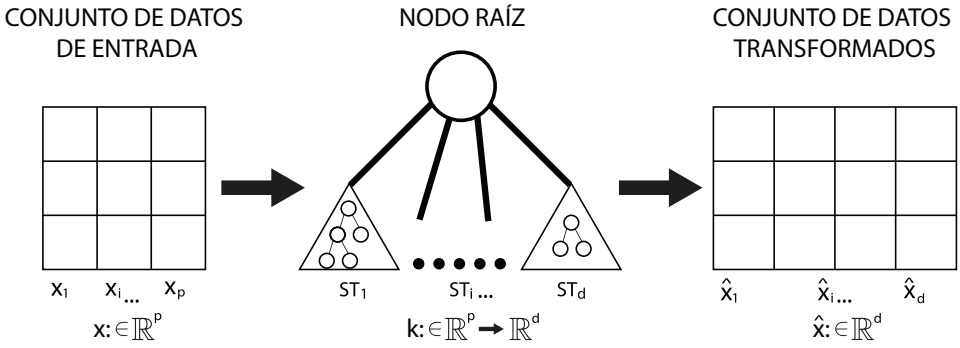


Figura 2.6: Árbol especial con d subárboles ST_i, cada uno definiendo una nueva característica de dimensiones construida con los datos de entrada.

Cada una de las nuevas características se construye mediante una combinación

lineal o no lineal de las características del problema donador. El método incluye operadores genéticos especializados, los cuales se enlistan a continuación:

Población inicial: M3GP comienza el ciclo evolutivo con una población aleatoria donde todos los individuos tienen una sola dimensión. Esto garantiza que la búsqueda evolutiva comience a buscar soluciones simples y unidimensionales, antes de avanzar hacia soluciones de mayor dimensión, que también podrían ser más complejas, como se observa en la Figura 2.7.

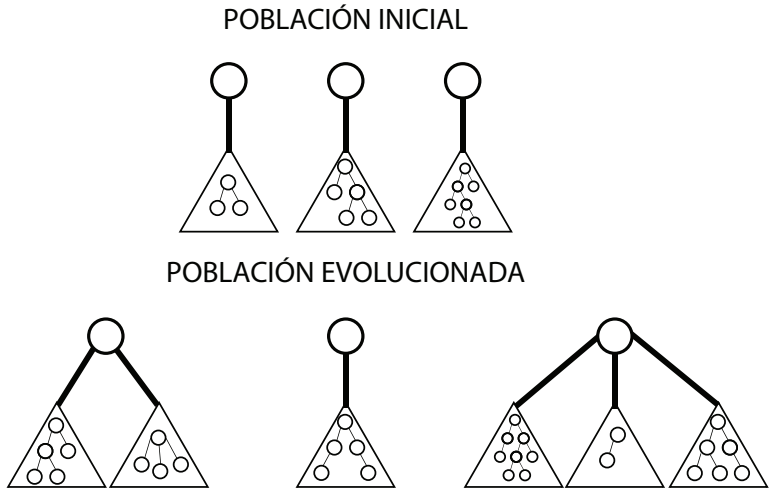


Figura 2.7: M3GP comienza con una población de transformaciones unidimensionales y, a través de la mutación, la búsqueda puede evolucionar a una población multidimensional.

Mutación: cuando este OG es elegido, se realiza una de tres acciones, con la misma probabilidad:

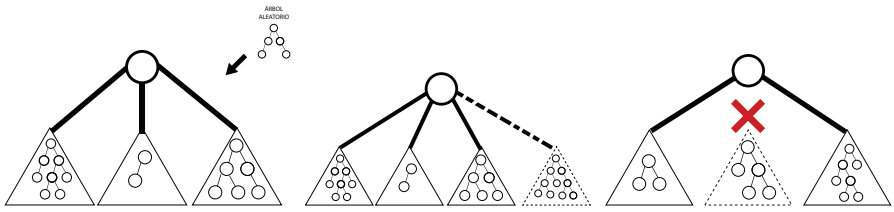
1.- Mutación de subárbol estándar (Figura 2.8:a), donde un nuevo árbol creado aleatoriamente reemplaza una rama elegida aleatoriamente (excluyendo el nodo raíz) del árbol principal;

2.- Se añade un nuevo subárbol al azar como una nueva rama del nodo raíz, agregando una dimensión al árbol principal (Figura 2.8:b).

3.- Se elimina aleatoriamente un subárbol del nodo raíz, eliminando una dimensión del árbol padre (Figura 2.8:c).

Cruce: Cuando este OG es elegido, se realiza una de dos acciones, con la misma probabilidad:

1.- Cruce de subárbol estándar, donde se elige un nodo aleatorio (excluyendo el nodo raíz) en cada uno de los padres, y se intercambian los respectivos brazos

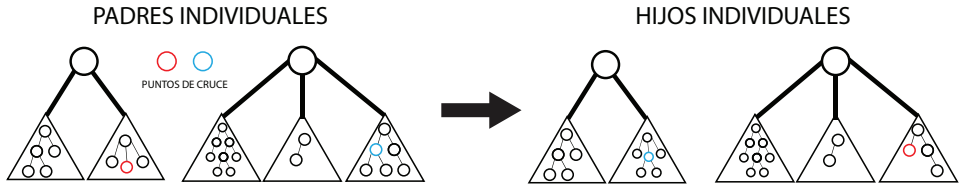


(a) Subárbol estándar (b) Dimensión añadida (c) Eliminación de dimensión

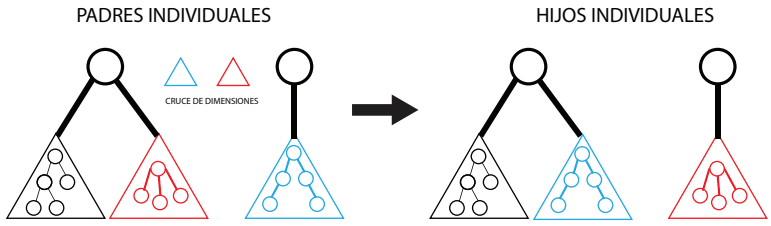
Figura 2.8: Las 3 mutaciones posibles en M3GP.

(Figura 2.9:a).

2.- Intercambio de dimensiones, donde se elige aleatoriamente una ramificación completa del nodo raíz en cada padre, y se intercambian entre sí, intercambiando las dimensiones entre los padres. El segundo evento es el caso particular del primero, donde se garantiza que los nodos de cruce se conectan directamente al nodo raíz (Figura 2.9:b).



(a) Cruce estándar de subárbol



(b) Cruce de dimensiones

Figura 2.9: Los 2 cruces posibles en M3GP.

Podar el árbol: el operador mutación puede agregar dimensiones a un árbol de forma aleatoria, lo cual puede afectar negativamente el rendimiento del nuevo árbol creado. Este operador también puede eliminar dimensiones pero lo hace de una forma aleatoria lo cual no asegura que el nuevo árbol creado sea mejor al anterior, para contrarrestar esta incertidumbre se aplica el procedimiento de recorte.

Este procedimiento elimina la primera dimensión de un individuo y reevalúa su rendimiento, como se observa en la Figura 2.10, si esto mejora el individuo recortado reemplaza al original y pasa a la reducción de la siguiente dimensión. De lo contrario, el individuo recortado se descarta y el individuo original se mantiene, el procedimiento se detiene después de recortar la última dimensión. El recorte sólo se aplica al mejor individuo de la generación.

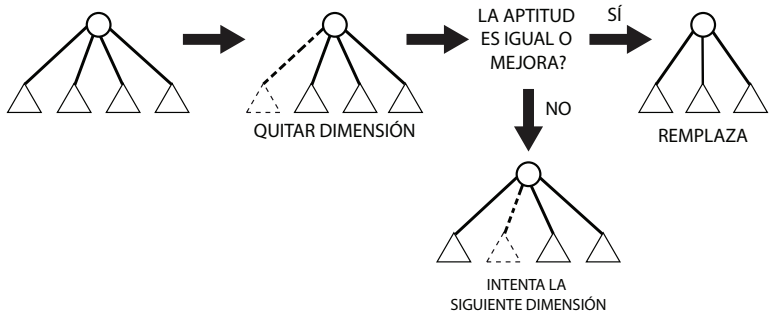


Figura 2.10: El proceso de recorte se detiene hasta que se revisan todas las dimensiones.

Elitismo: con el fin de explorar soluciones de diferentes dimensiones, M3GP se depende en la mutación para agregar y eliminar dimensiones de los individuos, con una probabilidad bastante alta. También tiene que confiar en la selección para mantener las mejores dimensiones en la población y descartar las peores, esto se logra creando un elitismo sobre la supervivencia de los individuos de una generación a otra.

M3GP no permite que se pierda el mejor individuo de cualquier generación, lo cual significa que siempre se mantiene y se asegura que se copie a la siguiente generación. Recordemos que este individuo ya está optimizado en el sentido de que pasó por un recorte.

2.2.4 COMPARATIVA Y EXTENSIONES RECIENTES DE M3GP

Los resultados experimentales de [19] muestran que M3GP logra un rendimiento excelente en comparación con métodos como bosques aleatorios, subespacios aleatorios y perceptrón multicapa, incluso con *benchmakrs* conocidos y problemas del mundo real. M3GP es un método muy prometedor al cual le han agregado varias mejoras como en [20], donde se incluyen hiperfunciones evolucionadas a los conjuntos de datos de referencia y obtuvieron una mejora signifi-

cativa del rendimiento de tres algoritmos de ML de última generación: árboles de decisión, bosques aleatorios y XGBoost.

Mientras en [21] los resultados experimentales muestran que su variante de M3GP, llamado M3GPSpectra, obtiene el mejor rendimiento entre los ocho métodos probados en su estudio. Adicionalmente, verifican que el método propuesto no es sensible al tamaño de las muestras de entrenamiento. Por lo tanto, M3GPSpectra es una herramienta muy prometedora para el análisis cuantitativo espectral de propiedades físicas o químicas de varios materiales.

Una variante competente para M3GP son los métodos que se describen en [22], donde proponen y desarrollan un marco para GP multidimensional llamado Herramienta de automatización de ingeniería de funciones (FEAT, por sus siglas en inglés de *Feature Engineering Automation Tool*). Ellos observan que un operador de cruce semántico basado en regresión por etapas conduce a mejoras significativas en un conjunto de problemas de regresión y que la inclusión del cruce semántico produce resultados de gran impacto en varios problemas de referencia de ML y GP.

Existen métodos que le sacan ventaja a M3GP, como en [23] donde proponen el método GP Ensamblado (eGP, por sus siglas en inglés de *Ensemble GP*). Este método fue probado en ocho problemas de clasificación binaria de varios dominios, con un número diferente de características. Se compararon diferentes variantes de eGP para después compararse con GP y M3GP. Los resultados muestran que eGP evoluciona constantemente modelos más pequeños y precisos que los de GP estándar. A pesar de que M3GP y XGBoost fueron los mejores métodos en general del estudio, en un problema particularmente difícil, las variantes de eGP fueron capaz de alcanzar resultados de generalización excepcionalmente buenos, muy por encima de todos los demás métodos.

Sin embargo, el método M3GP sigue mejorando e inpirando nuevas variantes como la que se observa en [24] llamada M4GP, en el cual se utiliza una representación de programa novedosa (basada en pila), que simplifica la construcción de soluciones multidimensionales e incorpora una técnica de supervivencia y selección de padres multiobjetivo que le permite superar marginalmente a M2GP y M3GP en un amplio conjunto de problemas de prueba.

Capítulo 3

APRENDIZAJE TRANSFERIDO

TL es un proceso donde se reutiliza el conocimiento de un problema (problema donador) para facilitar el proceso de aprendizaje de un problema nuevo (problema receptor). De esta manera se aprovecha el aprendizaje y la experiencia del problema donador para simplificar el entrenamiento del problema receptor. Con esto se reduce el número de ejemplos requeridos para entrenar el algoritmo que se aplique para resolver algún problema [25].

Al aplicar TL, es necesario considerar 3 cosas como lo proponen en [26]:

1.- *¿Qué transferir?* Se determina el tipo aprendizaje que se llevó a cabo en el problema donador para transferirse al problema receptor.

2.- *¿Cómo se transfiere?* Se debe indicar el proceso utilizado para transferir los conocimientos extraídos al problema receptor, ya que las diferencias entre la fuente y el objetivo pueden dificultar la reutilización de algunos aspectos del conocimiento transferido.

3.- *¿Cuándo transferir?* Es importante saber cuándo TL será beneficioso para el problema receptor (transferencia positiva) y cuándo no (transferencia negativa).

Las primeras dos preguntas pueden responderse en la implementación, pero la tercera pregunta es difícil de responder como se verá en 4. Predecir cuándo TL tendrá éxito parece ser que no es una tarea trivial. TL se puede clasificar en cuatro categorías dependiendo de cómo se responde la primera pregunta: *¿Qué se va a transferir?*:

1.- **Basado en instancia**: esto es cuando los datos del problema donador se transfieren al dominio del problema receptor. Qué datos se transfieren se determina en el proceso de aprendizaje.

2.- **Transferencia de conocimiento relacional:** se refiere a TL cuando existe alguna relación entre los dominios del problema donador y problema receptor, al igual sus datos comparten algunas similitudes. Un "*dominio relacional*", se refiere a un entorno donde existen relaciones entre conceptos, como las personas en una institución académica.

3.- **Enfoque de transferencia de parámetros:** éste se utiliza cuando el problema donador y el problema receptor comparten parámetros que pueden transferirse desde los modelos aprendidos.

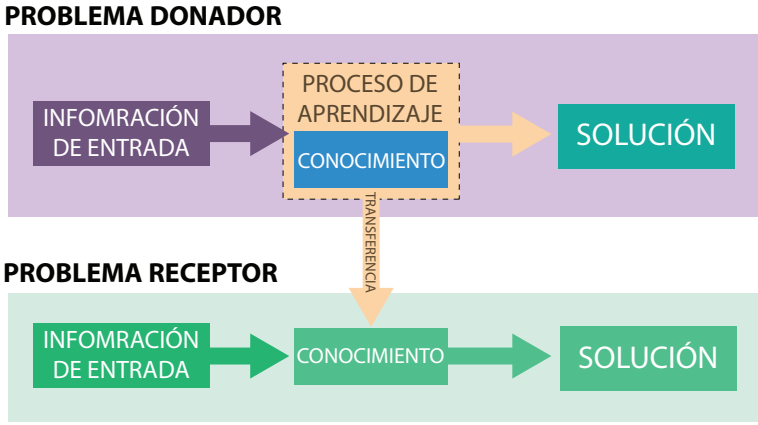


Figura 3.1: Flujo de trabajo de TL.

4.- **Función de transferencia de representación:** el conocimiento transferido entre los problemas de donador y receptor son las representaciones o transformaciones de las funciones aprendidas.

Para GP basado en inducción constructiva, se utiliza el último punto. Básicamente, los operadores de transformación de características que se aprendieron en los datos del problema donador se transfieren al problema receptor.

El proceso de aprendizaje debe repetirse utilizando los datos del problema receptor, reajustando los parámetros del modelo pero sin la necesidad de ejecutar la búsqueda de construcción de características basada en GP.

3.1. APLICACIONES DE APRENDIZAJE TRANSFERIDO

En ML, al entrenar un modelo para resolver algún problema en cierto dominio, se proporcionan un número limitado de datos etiquetados para ese problema, a esto se le conoce como un problema de aprendizaje supervisado.

El resultado de esto genera un modelo de ML con el cual es posible predecir datos nuevos y asignarles una etiqueta, siempre y cuando estos pertenezcan al mismo dominio del problema inicial.

Desafortunadamente, el aprendizaje supervisado falla cuando no se tienen suficientes datos etiquetados para entrenar un modelo confiable dentro del problema o dominio que nos interesa. Entrenar un modelo con pocos datos puede ocasionar que el modelo se sobreajuste a los datos y cometa errores al momento de predecir. Otros factores como la cantidad o calidad de datos también influyen al sobreajuste. A su vez, reentrenar un modelo es costoso computacionalmente.

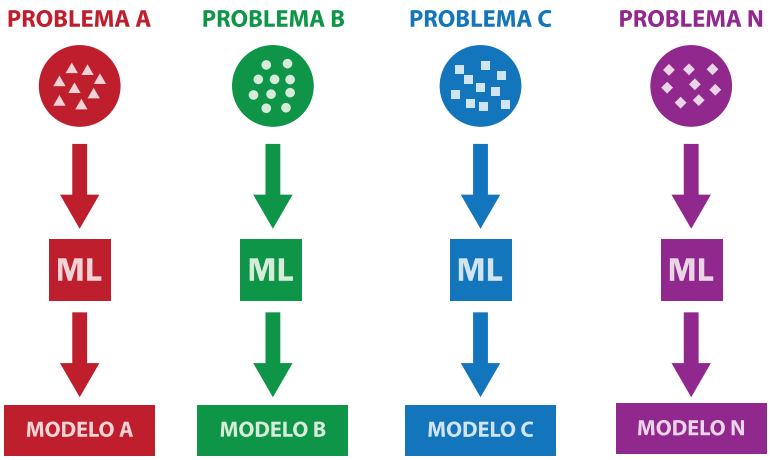


Figura 3.2: Flujo de trabajo tradicional de ML.

A pesar de que TL es un método muy prometedor, no es muy popular, sin embargo, cada vez es más utilizado por la comunidad. La principal razón por la cual no se utiliza es porque actualmente el auge es entrenar modelos utilizando redes neuronales, pero para entrenar estos modelos se necesitan una inmensa cantidad de datos, mucho tiempo y poder de cómputo, además esos datos tienen que estar etiquetados para lograr un buen rendimiento. Para algunos problemas y dominios, estos datos están disponibles, porque existen personas que se dedican a recopilar datos de diversas fuentes, pero para otros problemas existen muy pocos o simplemente no existen. En tales casos, TL es de gran utilidad para transferir el *çonocimiento"* de un problema donador.

El "*conocimiento*" que se transfiere puede ser: características, parámetros o modelos. TL se puede utilizar cuando se quiere reutilizar un modelo previamente entrenado como en [27], donde utilizan problemas de comprensión del lenguaje natural (NLU, por sus siglas en inglés de *Natural Language Understanding*) con problemas de respuesta a preguntas (QA) creando QANLU. Al utilizar este método se dieron cuenta de que con pocos datos el enfoque ofrece mejoras significativas en comparación con otros enfoques de NLU. Esas ganancias podrían incrementarse mediante el aprendizaje de transferencia secuencial a través de problemas de NLU de diferentes dominios y demuestran que su enfoque podría reducir la cantidad de datos necesarios para el mismo rendimiento hasta en un factor de 10.

También se puede utilzar si se quiere reducir el tiempo de entrenamiento como lo hacen en [28]. Este artículo proporciona un marco para seleccionar una red neuronal artificial (RNA) pre-entrenada para aplicar TL en la clasificación de tumores cerebrales en tipos benignos y malignos. Alimentaron varias RNA (Res-

Net101, ResNet50, GoogleNet, AlexNet, SqueezeNet) con imágenes de resonancia magnética para re-entrenarlas. Los resultados muestran que TL a través de AlexNet ofrece el mejor rendimiento con 99,04 % entre todas las RNA utilizadas en ese estudio. Aparte de tener el mejor rendimiento también se entrenó en menos tiempo que todas las demás RNA.

Un enfoque parecido es en [29], donde observaron que el entrenamiento de redes generativas adversarias (GANs, por sus siglas en inglés de *Generative Adversarial Networks*) con imágenes de alta calidad implica gastar muchos recursos informáticos, lo cual representa un cuello de botella para el desarrollo de aplicaciones de GAN. Su técnica de TL para GAN reduce significativamente el tiempo de entrenamiento en comparación con StyleGAN.

En [30] presentan la robustez que tiene TL ante la presencia de degeneración de imágenes. En particular, examinan el nivel de tolerancia cuando hay mala resolución, corrupción de ruido, recorte no deseado o rotación presente en las imágenes. El rendimiento del procesamiento de imágenes puede alcanzar una precisión del 75 % o más en todas las situaciones de degeneración de la imagen, incluido el peor de los casos, en el que el 100 % de las imágenes se corrompen con ruido con una desviación estándar de 0,5.

Sin embargo, la función más predominante de TL es cuando no se tienen datos suficientes en el problema que se quiere resolver, como en el caso de [31]. En su momento de investigación existían muy pocos datos de referencia para COVID-19, especialmente en imágenes de tórax. La idea principal era recopilar todas las imágenes posibles que existen del COVID-19 hasta la redacción de esa investigación y utilizar una técnica de "redes adversas generativas condicionales" para generar más imágenes que ayuden en la detección del COVID-19. Utilizaron varias RNA (AlexNet, VGGNet16, VGGNet19, GoogleNet, ResNet50), donde los resultados mostraron que ResNet50 es el modelo de aprendizaje más apropiado para detectar el COVID-19 a partir de un conjunto de datos limitados.

3.2. APRENDIZAJE TRANSFERIDO EN GP

TL en GP ha recibido un interés marginal en la investigación, particularmente en comparación con aprendizaje profundo (DL por su siglas en inglés *Deep Learning*), ya que normalmente se asume que una expresión simbólica evolucionada se adapta específicamente a los datos de un problema y, por lo tanto, no puede usarse en otros problemas. Sin embargo, esta idea no es del todo correcta como lo muestran en [26].

En [11] presentan una encuesta completa sobre el estado de TL en GP, de la cual podemos destacar lo siguiente. Primero, TL se puede rastrear hasta varias

formas de técnicas de inicialización de poblaciones [32] o enfoques de descomposición de problemas [33]. En segundo lugar, el interés por aplicar TL en GP ha aumentado gradualmente en los últimos años en varios dominios:

En [34], proponen varios métodos de TL para GP. Estos métodos se implementaron transfiriendo un número de buenos individuos o subindividuos del problema donador al problema receptor. Fueron probados en dos familias de problemas de regresión simbólica. Los resultados experimentales mostraron que los métodos de TL ayudan a GP a lograr mejores errores de entrenamiento. Es importante destacar que el rendimiento de GP en datos de entrenamiento cuando se implementó con TL también mejoró considerablemente.

Mientras [35] propone un nuevo método para emplear TL en GP para extraer y transferir conocimiento con el fin de abordar problemas complejos de clasificación de imágenes de textura. Ellos encuentran que el enfoque propuesto produce una mejor precisión de clasificación que la precisión de clasificación final obtenida por el método de referencia después de 50 generaciones.

De manera similar, un sistema de GP utilizando el TL transductiva [36], el sistema GP propuesto extrae automáticamente características de diferentes problemas, y estas características extraídas de GP se combinan para formar nuevos clasificadores que se aplican directamente a un problema nuevo. A partir de los resultados experimentales, el sistema de GP de TL transductiva propuesto puede desarrollar características de los problemas donadores para aplicarlos de manera efectiva a problemas receptores que sean similares a los problemas donadores.

Más recientemente, TL en GP se ha utilizado en el modelado de tráfico urbano [37], donde proponen el método GP con TL (GENTLE, por su acrónimo en inglés de *Genetic Programming with Transfer Learning*). Es un algoritmo capaz para resolver problemas de regresión simbólica, mejorado con un operador de retardo y una memoria para almacenar la población de modelos que se transferirán desde el problema donador al receptor.

También se ha aplicado una variante de TL a la síntesis de programas [38], donde muestran que *PushGP* [39] se beneficia sustancialmente del aprendizaje por transferencia de conjuntos de instrucciones, con un desempeño significativamente mejor en 8 de los 25 problemas de referencia utilizados.

En [11], además, también se muestra un análisis único de TL en GP, utilizando tanto clasificación multiclase como problemas de regresión simbólica. Los resultados muestran que TL puede, de hecho, tener éxito incluso cuando los dominios de los problemas donador y receptor son bastante diferentes. Además, TL no parece ser simétrico; es decir, una transferencia exitosa del origen al destino no garantiza que si los roles se invierten, el rendimiento también sería bueno.

Capítulo 4

CASO DE ESTUDIO

El trabajo realizado por [11], muestra que el TL es posible, particularmente en la inducción constructiva (CI) de características basada en GP, donde GP desarrolla operadores de construcción o transformación de características, y se utiliza un algoritmo de aprendizaje secundario para ajustarse al modelo final.

Utilizando problemas de regresión y clasificación del mundo real, los experimentos muestran que TL es realmente posible en este dominio, con varios resultados notables:

- TL es capaz de generar soluciones que superan, en muchos casos, los métodos de referencia.
- Se observa que algunos problemas son buenos problemas de donadores, mientras que otros son buenos receptores en una configuración TL.
- La transferibilidad de soluciones no siempre es simétrico; es decir, el problema A puede transferir buenas soluciones al problema B, pero lo contrario no es necesariamente cierto.

4.1. PROBLEMAS ANALIZADOS

Los principales elementos del trabajo experimental presentado en [11] se resumen a continuación. El sistema GP utilizado fue M3GP [19], del cual se habló en la Sección 2, mientras que el modelo predictivo final es un algoritmo de ML simple. Para la regresión se utiliza regresión lineal múltiple, para la clasificación se emplea la distancia de Mahalanobbis (MD, por sus siglas en inglés de *Mahalanobis Distance*), dada por:

$$MD = \sqrt{(x-m)^T \cdot C^{-1} \cdot (x-m)},$$

donde x es el vector de la observación (fila en un conjunto de datos), m es el vector de valores medios de variables independientes (la media de cada columna) y C^{-1} es la inversa de la matriz de covarianza de variables independientes.

Las soluciones en M3GP están representadas por individuos formados por múltiples árboles, donde cada subárbol del nodo raíz representa una nueva característica para el problema, de modo que se espera que el nuevo espacio de características sea más adecuado para el aprendizaje. Si bien [11] estudió TL tanto en tareas de regresión como de clasificación, este trabajo se centra en la clasificación. TL se evaluó considerando todas las posibles combinaciones de problemas donador y receptor, utilizando 8 problemas muticlase, estos son: HRT, IM-3, WAV, SEG, IM-10, YST, VOW y ML; los problemas se resumen en la Tabla 4.1.

Problema	HRT	IM-3	WAV	SEG	IM-10	YST	VOW	ML
Clases	2	3	3	7	10	10	11	15
Características	13	6	40	19	6	8	13	90
Muestras	270	322	5000	2310	6798	1484	990	360

Tabla 4.1: Dataset de 'heart' (HRT), 'segmentation' (SEG), 'vowel' (VOW), 'yeast' (YST) y 'movement-libras' (ML) se encuentran en KEEL dataset repository [40], mientras que 'waveform' (WAV) se encuentra en [41]. 'IM-3' y 'IM-10' son los datos satelitales utilizados en [42].

Estos problemas de clasificación presentan distintos rangos de muestras, clases y tipos de características. Algunos problemas contienen una gran cantidad de muestras (IM-10 tiene 6,798 muestras) y otros contienen una cantidad pequeña (HRT tiene alrededor de 270 muestras). Además, estos conjuntos de datos varían entre baja cantidad de características (IM-3 e IM-10 tienen 6 características) a una gran cantidad de características (ML tiene 90 características). Finalmente, los conjuntos de datos incluyen características binarias y de valor real. Por lo tanto, se eligen cuidadosamente estos problemas de referencia para que las evaluaciones realizadas no dependan del dominio del problema.

4.2. METODOLOGÍA DE APRENDIZAJE TRANSFERIDO

Los resultados que se muestran en esta sección fueron los obtenidos por [11], donde utilizaron la configuración que se muestra en la Figura 4.1. En la configuración propuesta, lo que se transfiere son las mejores características del problema donador. En el problema receptor no se realiza ninguna otra evolución, solo el método de aprendizaje.

Para realizar TL con M3GP, se utilizó la transformación de características evolucionadas de un problema para construir un nuevo espacio de características en el problema receptor antes de aplicar el algoritmo de ML utilizando solo los datos del problema receptor. Sin embargo, esto lleva al problema de alineación de

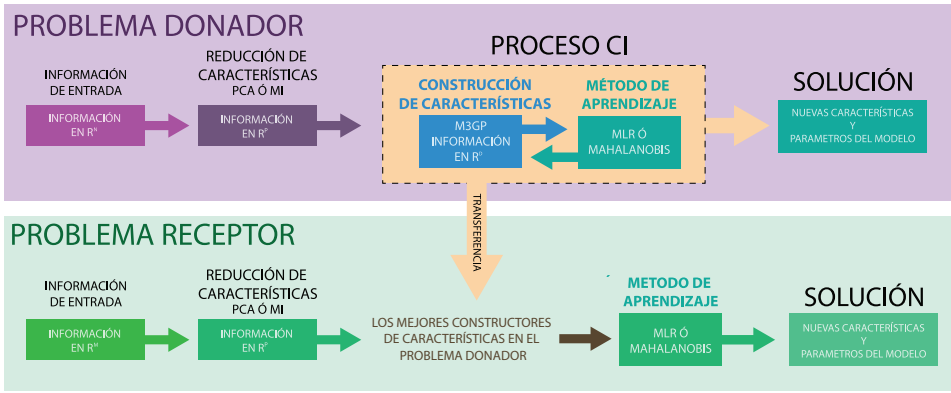


Figura 4.1: Flujo de trabajo de TL con M3GP.

características, dado que los problemas donador y receptor pueden tener un número diferente de características, ya que que no existe una correspondencia *a priori* entre ambos espacios de características para determinar cómo aplicar las transformaciones de características evolucionadas del problema donador al receptor.

Por lo tanto, el rendimiento de TL se evaluó de la siguiente manera. Primero, el espacio de características de todos los problemas se redujo a 6 características usando análisis de componentes principales (PCA, por sus siglas en inglés de *Principal Component Analysis*) o información mutua (MI, por sus siglas en inglés de *Mutual Information*).

PCA: es una técnica para reducir la dimensionalidad de un conjunto de datos, preservando al mismo tiempo la mayor ”variabilidad”(es decir, información estadística) como sea posible, aumentando la interpretabilidad pero al mismo tiempo minimizando la pérdida de información **PCA**. Lo hace creando nuevas variables no correlacionadas que maximizan sucesivamente la varianza. Encontrar estas nuevas variables, los componentes principales, se reduce a resolver un problema de eigenvalores/eigenvectores, dado por

$$Av = \lambda v$$

donde $A \in \mathbb{R}^{m\times m}$ es una matriz cuadrada; $v \in \mathbb{R}^{m\times 1}$ es un vector columna; $\lambda \in \mathbb{R}^{m\times m}$ es una matriz diagonal. v son los eigenvectores y λ son los eigenvalores.

Las nuevas variables se definen mediante el conjunto de datos en cuestión, no *a priori*, por lo que el PCA se convierte en una técnica de análisis de datos adap-

tativo **PCA**.

MI: es utilizada para medir la dependencia mutua de dos variables aleatorias, específicamente, cuantifica la cantidad de información obtenida sobre una variable aleatoria al observar la otra variable aleatoria. Se utiliza una versión normalizada de MI para el caso continuo **MI** para medir la dependencia funcional entre una característica de entrada X y la salida esperada Y. Sea (X,Y) un vector aleatorio normalmente distribuido con el ρ coeficiente de correlación de Pearson, la información mutua se calcula por $I(X;Y) = -1/2\log(1-\rho^2)$. Podemos definir una versión normalizada de MI **MI** como

$$I_c^*(X;Y) := \sqrt{1-\exp[-2I(X;Y)]}$$

los valores de $I_c^*(X;Y)$ estan en el rango de $[0,1]$, donde más cercano a 1 se considera funcionalmente dependiente, y son iguales a $|\rho|$ solo si (X,Y) se distribuye normalmente con correlación coeficiente ρ.

En este procedimiento la reducción de características en todos lo problemas no bajó de 85 % en pérdida de información, lo cual es un rango aceptable. Luego, se probaron todas las combinaciones posibles, tanto de estrategias de reducción de características como de correspondencias de características. Estas son 4 permutaciones de reducción de características para problema donador y receptor (PCA / MI, MI / PCA, PCA / PCA, MI / MI) y 720 posibles alineaciones entre los espacios de características de ambas tareas (usando de 6 características para problema donador y receptor hay un total de 720 alineaciones de características diferentes entre ellos).

4.3. COMPATIBILIDAD DE PROBLEMAS

El éxito o el fracaso de la TL se cuantificó basándose en la idea de una transferencia positiva, esto ocurre cuando el rendimiento de un modelo transferido, aplicado al problema receptor, es mejor que el rendimiento de un método de referencia, basado en datos de prueba. Se consideraron dos métodos de referencia para las tareas de clasificación, MD y M3GP.

Además, la compatibilidad de TL se propone en [11] como una medida que resume el número de transferencias positivas que ocurren cuando se utilizan diferentes estrategias de reducción de características para los problemas donador y receptor. La compatibilidad de TL está entre 0 y 4, igual al número de permutaciones de reducción de características que producen al menos una transferencia positiva (entre todas las posibles alineaciones de características entre ambas tareas).

La Tabla 4.2 resume las puntuaciones de compatibilidad de TL para todas las combinaciones de problemas donador y receptor, considerando ambos métodos de referencia (MD y M3GP). Esto nos da cuatro valores de compatibilidad TL para dos problemas cualesquiera, para los dos métodos de referencia y las dos posibilidades de que cada problema sea el problema donador o receptor. Cada par de problemas se presenta en formato (u,v,w,z), donde u y w representan el problema donador (se encuentran en renglón superior de la Tabla 4.2) mientras que w y z representan el problema receptor (se encuentran en la primera columna de la Tabla 4.2). Además, u y w se basan en en el método de referencia MD, y v y z se basan en M3GP.

	HRT	**IM-3**	**WAV**	**SEG**	**IM-10**	**YST**	**VOW**	**ML**
HRT	-	(4,4,4,2)	(2,2,4,3)	(4,0,4,3)	(3,0,4,3)	(1,0,4,2)	(0,0,4,3)	(1,1,4,2)
IM-3	-	-	(2,2,4,4)	(4,0,4,4)	(3,0,4,4)	(1,0,4,2)	(0,0,4,1)	(1,1,4,2)
WAV	-	-	-	(4,1,2,2)	(3,0,2,2)	(1,0,2,2)	(0,0,2,2)	(2,2,2,2)
SEG	-	-	-	-	(4,0,4,0)	(2,0,2,0)	(0,0,3,0)	(2,2,3,0)
IM-10	-	-	-	-	-	(4,0,1,0)	(2,0,2,0)	(2,2,2,0)
YST	-	-	-	-	-	-	(2,0,1,0)	(1,1,2,0)
VOW	-	-	-	-	-	-	-	(1,1,2,0)
ML	-	-	-	-	-	-	-	-

Tabla 4.2: Puntuaciones de compatibilidad TL para cada par de problemas, en formato (u,v,w,z) donde u y w representan el problema donador mientras que w y z representan el problema receptor. A su vez u y w se basan en en el método de referencia MD, y v y z se basan en M3GP.

4.4. PLANTEAMIENTO DEL PROBLEMA

En este trabajo de tesis el interés es determinar cuándo un par de problemas tendrán éxito al aplicarse un TL. En particular, se quiere información que permita determinar esto sin ninguna información más allá del espacio de características de un problema (después de la reducción de características). En la Figura 4.2, se muestra de dónde salen los datos a analizar para este trabajo, tomando como referencia el flujo de trabajo de [11].

En [10] los autores intentan determinar qué fuente es la más adecuada para un problema receptor en particular, pero para hacer esto se requiere el modelo y es necesario aplicar el modelo en los datos del problema receptor.

De manera similar, en el trabajo previo se requiere conocimiento del desempeño de los métodos de referencia [11]. Dado que no usaremos la transformación de características, ni ningún proceso de aprendizaje, para analizar la compatibili-

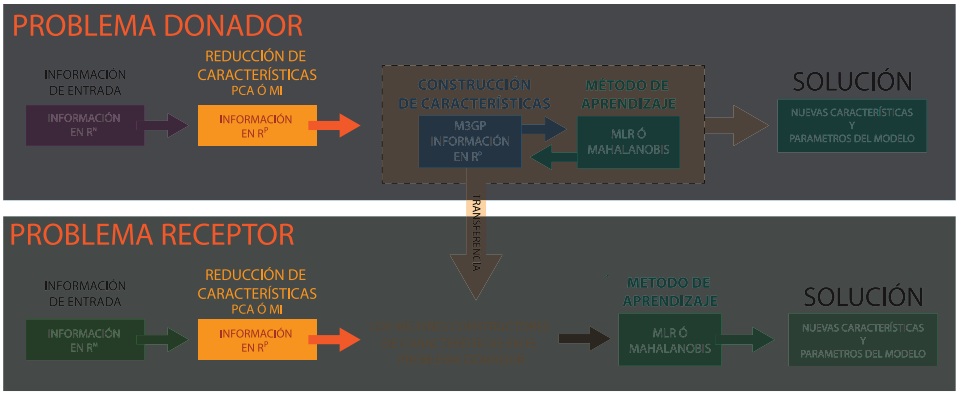


Figura 4.2: Para este análisis veremos las propiedades intrínsecas de un espacio de características, por ello el enfoque del problema es en la zona donde están los problemas con reducción PCA/MI (subrayada de color naranja).

dad de TL observada, podemos estar más seguros de que estamos basando nuestro análisis en propiedades intrínsecas de un espacio de características de los problemas. Lo que se requiere es una medida de distancia o similitud entre dos problemas cualesquiera [10], que pueda explicar la suposición subyacente en cualquier configuración de TL; es decir, que los problemas donador y receptor son similares de alguna manera.

Sin embargo, un aspecto a considerar es que [11] mostró que TL no es simétrico, lo que impone una limitación con respecto a cómo se deben analizar dos problemas. De los resultados presentados en la Tabla 4.2, se hacen las siguientes observaciones. En todos los pares de problemas se obtiene al menos una transferencia positiva. Algunos pares de problemas muestran claramente el resultado no simétrico discutido anteriormente. Además, podemos definir al menos dos grupos de pares de problemas en función de sus cuatro puntuaciones de compatibilidad TL.

Primer grupo, problemas compatibles con TL (PCTL) : Este grupo está compuesto por todos los pares de problemas con al menos tres valores de compatibilidad TL distintos de cero, de los cuatro informados en la Tabla 4.3, resaltados en gris. Esto significa que, independientemente de qué problema sea el origen y cuál sea el objetivo, TL puede producir al menos una transferencia positiva entre estas dos tareas.

	HRT	**IM-3**	**WAV**	**SEG**	**IM-10**	**YST**	**VOW**	**ML**
HRT	-	(4,4,4,2)	(2,2,4,3)	(4,0,4,3)	(3,0,4,3)	(1,0,4,2)	(0,0,4,3)	(1,1,4,2)
IM-3	-	-	(2,2,4,4)	(4,0,4,4)	(3,0,4,4)	(1,0,4,2)	(0,0,4,1)	(1,1,4,2)
WAV	-	-	-	(4,1,2,2)	(3,0,2,2)	(1,0,2,2)	(0,0,2,2)	(2,2,2,2)
SEG	-	-	-	-	(4,0,4,0)	(2,0,2,0)	(0,0,3,0)	(2,2,3,0)
IM-10	-	-	-	-	-	(4,0,1,0)	(2,0,2,0)	(2,2,2,0)
YST	-	-	-	-	-	-	(2,0,1,0)	(1,1,2,0)
VOW	-	-	-	-	-	-	-	(1,1,2,0)
ML	-	-	-	-	-	-	-	-

Tabla 4.3: Puntuaciones de compatibilidad TL para cada par de problemas, con el grupo PCTL en un fondo gris y el grupo PNTL en blanco.

Segundo grupo, problemas no compatibles con TL (PNTL) : Estos son los pares de problemas restantes, donde TL tuvo menos éxito o donde el rendimiento no fue simétrico.

Está claro que ambos grupos son algo estrictos en la forma en que se definen. PCTL requiere que TL tenga éxito independientemente de cuál es el problema receptor y cuál es el problema donador.

Por otro lado, los pares de problemas de PNTL pueden mostrar un rendimiento de TL muy bueno desde un problema donador y receptor, pero no al revés. En la práctica, esto podría no ser un problema si se utiliza la fuente correcta, pero de lo contrario, puede fallar.

Para calcular una medida de similitud, podríamos adoptar varios enfoques. Una forma podría ser usar medidas descriptivas de cómo se distribuyen las diferentes clases en el espacio de características [43], pero estas medidas son difíciles de aplicar en problemas de varias clases, particularmente cuando diferentes problemas tienen un número diferente de clases. En este trabajo, especulamos que la relación relativa entre las características podría ser clave para comprender la compatibilidad de TL.

En otras palabras, estamos interesados en analizar cómo las diferentes características se relacionan entre sí, y planteamos la hipótesis de que cuando dos problemas comparten una estructura común con sus respectivos espacios de características, entonces pueden considerarse compatibles con TL. Para realizar este análisis, empleamos un enfoque de aprendizaje automático explicable de última generación llamado DeepInsight [12].

Capítulo 5

DEEPINSIGHT

DeepInsight, convierte datos que no son imágenes a una forma de imagen bien organizada, esto permite la extracción de características mediante la aplicación de una red neuronal de convolución (CNN, por sus siglas en inglés de *Convolutional Neural Network*) para entender y clasificar la información que contienen estos datos [12].

A partir de un vector de características x, éste se transforma en una matriz de características M mediante una transformación T (Figura 6.1). La ubicación de las características en las coordenadas cartesianas depende de la similitud de las características. Una vez que se determinan las ubicaciones de cada característica en una matriz de características, se mapean los valores de expresión o los valores de característica.

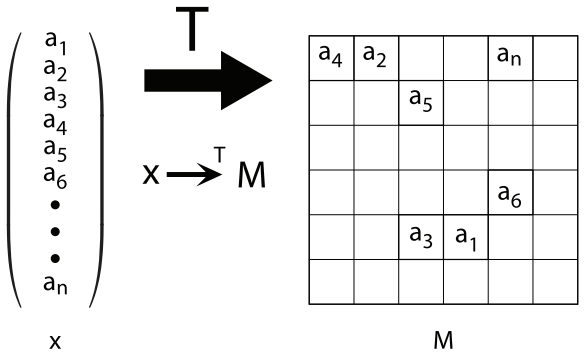


Figura 5.1: Transformación de Vector a Matriz, donde x es el vector de características transpuesto de un renglón del conjunto de datos, T es una transformación dada por una medición de similitud o técnica de reducción de dimensión y M es la matriz de características resultante.

Esto generará una imagen única para cada muestra, N muestras de d características proporcionarán N muestras de mn matrices de características. Esta forma de matriz 2D tendrá todas las características d. Este conjunto de N matrices de

características se procesa en la arquitectura CNN para aprender el modelo y proporcionar predicciones. Para este trabajo no requiere un modelo de red neuronal, simplemente tomamos las imágenes y sobre ellas se realiza el análisis.

Se utiliza un conjunto de entrenamiento para encontrar la ubicación de las funciones. Si el conjunto de entrenamiento que consta de n muestras se define como $X = \{x_1, x_2, \ldots, x_n\}$ donde un vector de características tiene d características o $x \in \mathbb{R}^d$, entonces también podemos definir un gen o conjunto de características $G = g_1, g_2, \ldots, G_d$ donde $g \in \mathbb{R}^n$; es decir, una característica g_j tiene n muestras de entrenamiento. Básicamente, G se puede obtener transponiendo X.

5.1. FUNCIONAMIENTO DE DEEPINSIGHT

Se utiliza el conjunto de características G y se aplica una técnica de medición de similitud o una técnica de reducción de dimensionalidad como incrustación de vecinos estocásticos distribuidos en t (t-SNE, por sus siglas en inglés de *t-distributed stochastic neighbor embedding*) o análisis de componentes principales del núcleo (kPCA, por sus siglas en inglés de *kernel Principal Component Analisis*) para obtener un plano 2D. Los puntos en este plano cartesiano son las características o genes. Estos puntos sólo definen la ubicación de las características, no la característica en sí ni los valores de expresión. Una vez que se define la ubicación de las características, se usa el algoritmo de casco convexo (mejor conocido en inglés como *convex hull*) para encontrar el rectángulo más pequeño que contiene todos los puntos. Dado que la imagen debe estar enmarcada en forma horizontal o vertical para la arquitectura de CNN, se realiza una rotación. A partir de entonces, las coordenadas cartesianas se convierten en píxeles. El funcionamiento de *DeepInsight* se puede observar en la Figura 5.2.

La conversión de coordenadas cartesianas a cuadros de píxeles se realiza promediando algunas características, ya que el tamaño de la imagen tiene una limitación de píxeles. El cuadro de píxeles, por lo tanto, consistirá en las posiciones de las características (o genes) para una muestra x_j (para $j = 1, 2, \ldots, n$). Una vez que se determina la ubicación, el siguiente paso es mapear los valores de la característica (o expresión genética) a estas ubicaciones de píxeles.

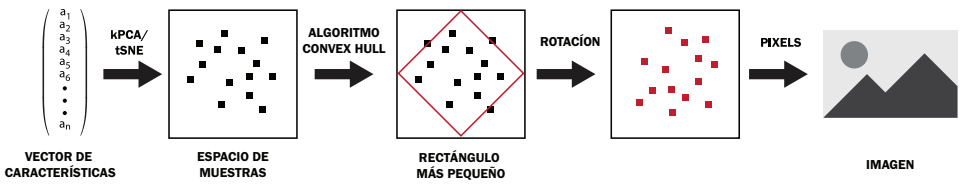


Figura 5.2: Funcionamiento de DeepInsight.

Si más de una característica adquirió la misma ubicación en el cuadro de píxeles, entonces, durante el mapeo de las características, las características respectivas se promediarán y se colocarán en la misma ubicación. Por lo tanto, si la resolución de la imagen o el tamaño de la cuadrícula es muy pequeña (en comparación con la cantidad de características dadas), muchas características se superponen entre sí y la representación de la imagen puede no ser muy precisa. Se debe seleccionar una resolución adecuada dada la capacidad del hardware y el número de funciones necesarias para procesar. Alternativamente, la reducción de la dimensionalidad puede aplicarse *a priori*.

5.2. ANÁLISIS DE PROBLEMAS CON DEEPINSIGHT

Para cada problema definido en la Tabla 4.1 se extrae su representación del espacio de características usando DeepInsight. Esto se hace para ambas versiones del problema, utilizando PCA o MI como reducción de características. En la Figura 5.3 se muestra el flujo de trabajo que se realizó para cada problema.

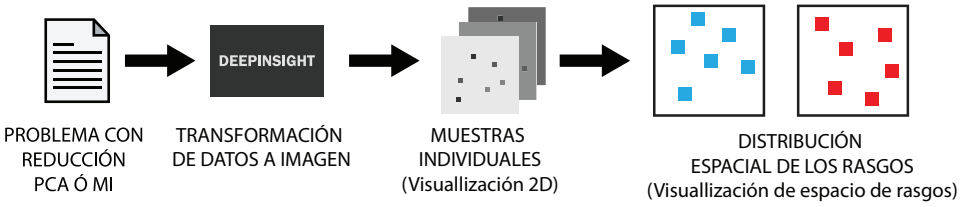


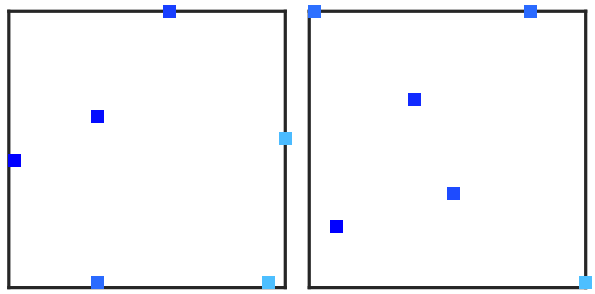
Figura 5.3: Flujo de análisis con DeepInsight.

DeepInsight proyecta las características de un problema en un plano 2D o una imagen digital, en este plano, los únicos píxeles visibles son la características de un problema, su distribución geométrica dentro de la imagen está determinada por sus similitudes relativas, o correlación, basadas en los datos. De esta manera, cada muestra en un conjunto de datos se puede representar mediante una imagen (como se observa en la Sección 5.2.1), donde el valor de intensidad de los píxeles se basa en su representación del espacio de características.

En en este trabajo, sin embargo, lo que interesa es la distribución geométrica del espacio de características de un problema (5.2.2), en lugar de las imágenes de muestras singulares, ya que queremos comparar problemas completos y no instancias individuales. En la Tabla 5.1 se resume como los problemas originales (mostrados en la Tabla 4.1) se traducen a la conversión por DeepInsight

Problema	HRT	IM-3	WAV	SEG	IM-10	YST	VOW	ML
Tipo de imágenes	2	3	3	7	10	10	11	15
Pixeles	6	6	6	6	6	6	6	6
Cantidad de imágenes	270	322	5000	2310	6798	1484	990	360

Tabla 5.1: Problemas convertidos por DeepInsight.



(a) VOW-MI (b) VOW-PCA

Figura 5.4: Espacio de rasgos generado por DeepInsight para el problema VOW con su reducciones: MI (a) y PCA (b).

5.2.1 VISUALIZACIÓN 2D

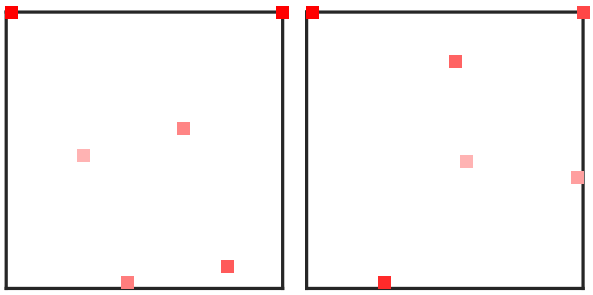
Una muestra individual (como se muestra en la Figura 5.4 y Figura 5.5) se interpreta de la siguiente manera:

- Cada píxel representa una característica de los datos.
- Los píxeles más cercanos, son las características más parecidas entre sí.
- El color del píxel representa el valor de la característica para la muestra en particular que se está observando, rasgos con valores altos tendrán un color saturado, valores bajos un color tenue.

5.2.2 VISUALIZACIÓN DE ESPACIO DE CARACTERÍSTICAS

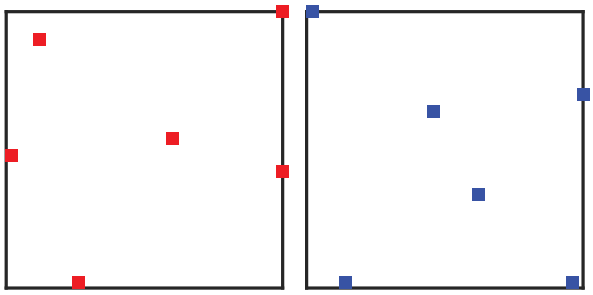
Si mostramos sólo la posición de los rasgos de un problema en particular, como se muestra en la Figura 5.6, se observa la distribución espacial que DeepInsight genera para las características de un problema.

Este tipo de imágenes son las que se analizan en la Sección 6 para determinar transferencias positivas y negativas con base en la Tabla 4.2.



(a) YST-MI (b) YST-PCA

Figura 5.5: Espacio de rasgos generado por DeepInsight para el problema YST con su reducciones: MI (a) y PCA (b).



(a) HRT-MI (b) HRT-PCA

Figura 5.6: Visualización del espacio de características generado por DeepInsigth para el problema HRT después de la reducción de PCA (a) y MI (b).

Capítulo 6

Análisis y Resultados

En el Capítulo 4 se definieron dos grupos de pares de problemas en función de sus cuatro puntuaciones de compatibilidad TL, de acuerdo a lo observado en la Tabla 4.2 definimos el grupo PCTL y PNTL. En el Capítulo 5, se habló de cómo se extrae la representación del espacio de características usando DeepInsight, para ambas versiones de un problema, utilizando PCA o MI como reducción de características. En este capítulo se analizan ambos grupos, mostrando imágenes de referencia y explicando cómo estas ayudaron a formar la metodología del presente trabajo de tesis.

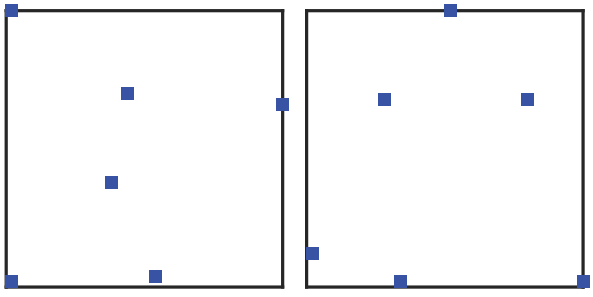
6.1. Análisis de PCTL

El análisis del espacio de características de dos problemas se comienza observando la Tabla 4.2, en teoría los problemas que tienen mejor puntación de compatibilidad (grupo PCTL) tendrán un espacio donde las características se encuentran en un lugar similar. Si observamos los problemas IM3 y WAV que tienen una puntación de compatibilidad (PC) $(2,2,4,4)$, sus espacios de características con reducciones PCA y MI, son los que se observan en la Figura 6.1. Las Figuras 6.1 *a* y 6.1 *d* comparten una gran similitud en 4 de sus 6 características, mientras tanto la Figura 6.1 c y 6.1 b comparten 3 similitudes.

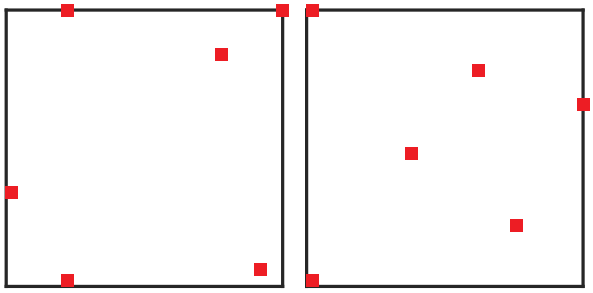
Si observamos otro par de problemas como IM3 y HRT que tienen el PC de $(4,4,4,2)$, se obtiene la Figura 6.2. Las Figuras 6.2 *a* y 6.2 *d* tienen una correlación en la posición de características al igual como las Figuras 6.2 *b* y 6.2 *c*.

Si observamos el problema SEG y WAV que tienen un PC de $(4,0,4,4)$, tenemos la Figura 6.3. A pesar de que las imágenes 6.3 *b* y 6.3 *c* muestran una correlación alta, en el caso de 6.3 *a* y 6.3 *d*, el análisis visual no es suficiente para a hacer una correlación obvia entre las imágenes. Sin embargo, si rotamos una de las dos imágenes (en este caso imagen d de la Figura 6.3), observamos lo siguiente en la Figura 6.4.

Ahora los espacios de características son altamente similares, esto nos da entender que DeepInsight genera imágenes que en realidad no tienen una orientación fija. Esto abre la posibilidad de analizar mejor los espacios de un par de problemas,



(a) IM3-PCA (b) IM3-MI



(c) WAV-MI (d) WAV-PCA

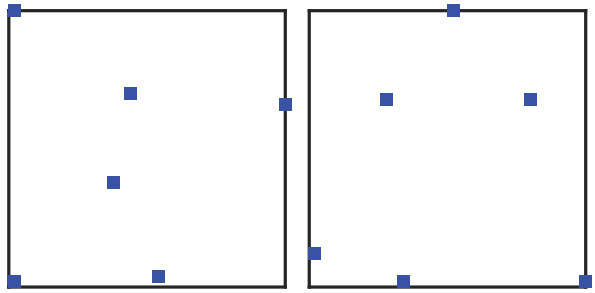
Figura 6.1: Comparativa de distribuciones de características para los problemas IM3 y WAV, donde IM3-PCA (a) y WAC-PCA (d) comparten una similitud en la posición de sus rasgos.

por ejemplo: los problemas IM3 y YST con PC de $(1,0,2,2)$ que se muestran en la Figura 6.5.

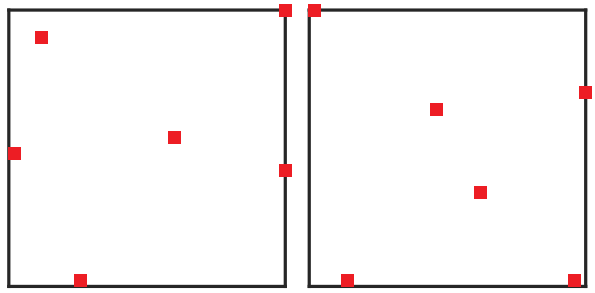
Las Figuras 6.5 *a* y 6.5 *b*, a simple vista no tienen una correlación clara, sin embargo, al rotar YST podemos observar una mejor correlación entre los problemas como se muestra en la Figura 6.6. Ahora, al realizar este análisis no importa cuál de las dos imágenes se rota, si los problemas corresponden al grupo PCTL estas van a concordar sin importar la rotación.

6.2. ANÁLISIS DE PNTL

Si esta vez observamos el grupo PNTL de la Tabla 4.2, se comienzan a notar grandes diferencias entre los problemas, ya que en teoría los problemas con peor puntación de compatibilidad baja tendrán sus características en posiciones muy diferentes. Si observamos el par de problemas VOW y SEG con PC de $(0,0,3,0)$



(a) IM3-PCA (b) IM3-MI



(c) HRT-MI (d) HRT-PCA

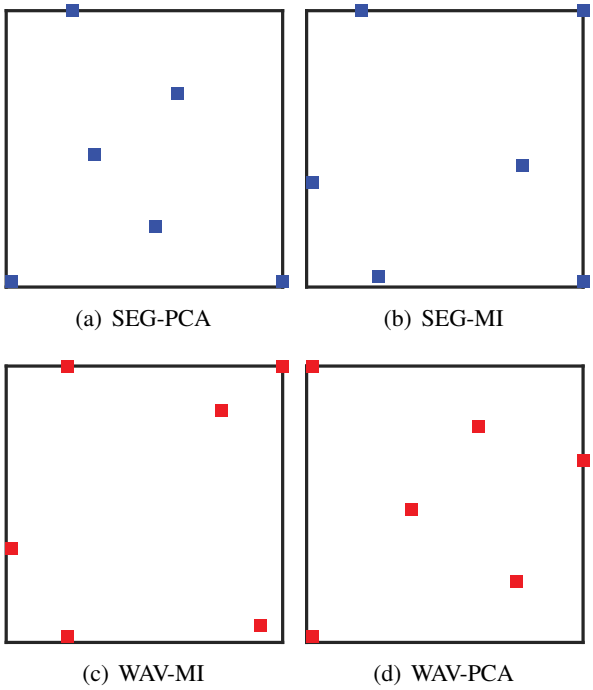
Figura 6.2: Comparativa de distribuciones de características para los problemas IM3 y HRT, donde IM3-PCA (a) y HRT-PCA (d) comparten una similitud en la posición de sus rasgos.

en la Figura 6.6 podemos ver que, a pesar de que 6.6 *a* con 6.6 *c* y 6.6 *d* tienen un punto en común, todos sus demás puntos están desalineados. Si se compara 6.6 *b* con 6.6 *c* y 6.6 *d*, la discrepancia de los puntos es mayor.

Si tomamos otro par de problemas como YST e IM10 con PC de $(4,0,1,0)$ o el par IM10 y SEG con PC de $(4,0,4,0)$, se observa en la Figura 6.8 y 6.9 que los puntos no tienen una correlación clara. Esta discrepancia se sigue observando a lo largo de todos los problemas del grupo PNTL, si aplicamos rotaciones a cualquiera de las dos imágenes los puntos no parecen concordar, sin importar la posición.

6.3. ALINEACIÓN DE ESPACIO DE CARACTERÍSTICAS CON ICP

Una vez realizado el análisis visual el objetivo se torna a comparar la representación espacial de características de dos problemas para determinar su similitud. Esto se hace realizando un registro de las imágenes de DeepInsight de dos proble-



(a) SEG-PCA (b) SEG-MI

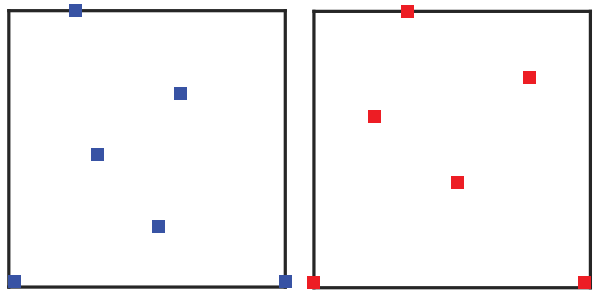
(c) WAV-MI (d) WAV-PCA

Figura 6.3: Comparativa de distribuciones características para los problemas SEG y WAV, donde SEG-MI (b) y WAC-PCA (c) comparten una similitud en la posición de sus rasgos.

mas tomados como nubes de puntos 2D utilizando el método iterativo del punto más cercano (ICP, por sus siglas en inglés de *Iterative Closest Point*). Para ello se necesitan dos nubes de puntos (nube de vértices). Una nube de puntos como referencia u objetivo la cual se mantendrá fija y otra nube (conocida como fuente) que se irá transformando para que coincida mejor con la referencia u objetivo, como ejemplo está la Figura 6.10.

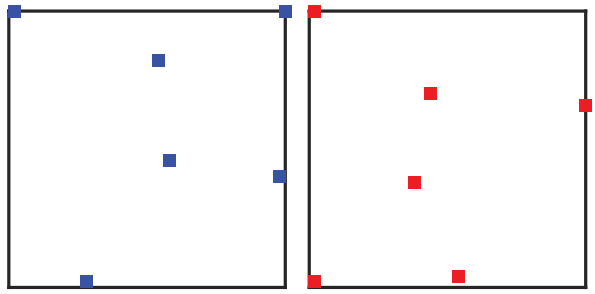
ICP busca la mejor transformación rígida (rotación y traslación) de una nube de puntos a otra (la imagen de referencia) y devuelve la distancia euclidiana promedio entre la nube de puntos resultante y la referencia. En las Figuras 6.11 y 6.12, se observan los resultados que se lograron utilizando ICP.

Esa comparación entre los espacios de características es la clave para determinar su similitud. Por lo tanto, para cualquier par de problemas, se calculan 8 valores, considerando las cuatro combinaciones de reducción de características (PCA-MI, MI-PCA, PCA-PCA y MI-MI), y las dos posibilidades de calcular la



(a) SEG-PCA (b) WAV-PCA

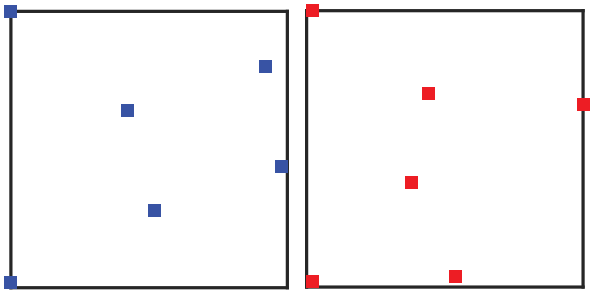
Figura 6.4: Distribución de características SEG-PCA (a) con WAV-PCA (b) rotada a 90 grados.



(a) YST-PCA (b) IM3-PCA

Figura 6.5: Distribución de características YST-PCA (a) e IM3-PCA(b), ambas figuras no parecen compatibles a simple vista a pesar de pertenecer al grupo PCTL.

medida de ICP (alternando qué problema se usa como referencia). La medida de ICP calculada no es simétrica, ya que depende de la condición inicial del proceso de registro; es decir, depende de la imagen que se utilice como referencia. Para ellos nos enfocamos en el mínimo de los 8 valores como una medida representativa de la similitud del problema, a esto lo llamamos distancia de registro mínima (DRM) de las imágenes del espacio de características DeepInstight.



(a) YST-PCA rotada 90° (b) IM3-PCA

Figura 6.6: Distribución de características YST-PCA rotada a 90° (a) para encajar con IM3-PCA (b), demostrando que DeepInSight no genera imágenes con orientación fija.

	HRT	**IM-3**	**WAV**	**SEG**	**IM-10**	**YST**	**VOW**	**ML**
HRT	-	6.42	6.91	6.94	7.54	5.07	6.79	8.17
IM-3	-	-	5.66	7.8	8.69	6.62	6.59	8.24
WAV	-	-	-	5.15	7.72	4.45	7.75	8.51
SEG	-	-	-	-	8.3	6.65	7.25	6.53
IM-10	-	-	-	-	-	8.67	9.66	6.27
YST	-	-	-	-	-	-	9.12	6.09
VOW	-	-	-	-	-	-	-	5.36
ML	-	-	-	-	-	-	-	-

Tabla 6.1: DMR para cada par de problemas, el grupo PCTL está en gris y el grupo PNTL en blanco.

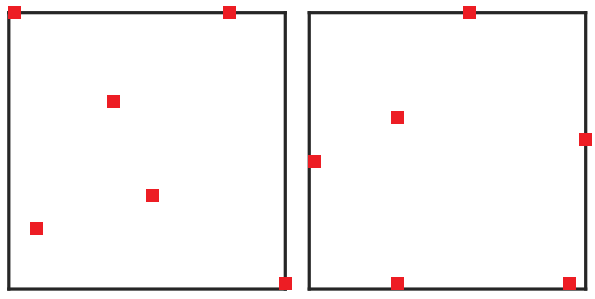
6.4. RESULTADOS

DMR se centra en la distancia mínima porque puede verse como el mejor escenario para la similitud de problemas, un método de referencia para determinar cuándo dos problemas pueden ser una buena solución para que se dé un TL positivo. La distancia mínima de registro se da en la Tabla 6.1 para cada par de problemas. Con esto definimos las siguientes dos hipótesis nulas:

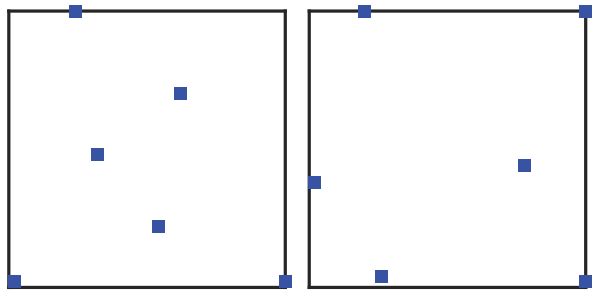
1. $H0_1$: Los grupos PCTL y PNTL tienen la misma mediana de DMR.

2. $H0_2$: Los grupos PCTL y PNTL tienen el mismo promedio de DMR.

Para evaluar estas hipótesis se usaron la prueba *ran ksum* para $H0_1$ y *t-test* para $H0_2$. La DMR mediana y promedio del grupo PCTL son 6,79 y 6,90, mientras que



(a) VOW-PCA (b) VOW-MI



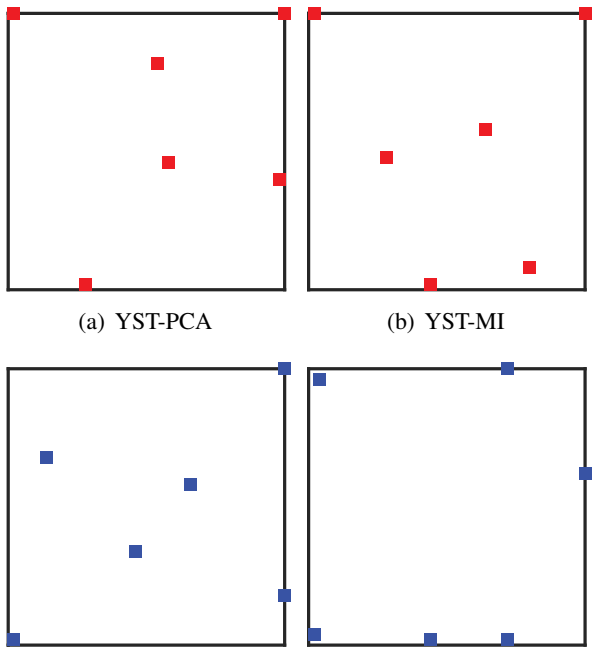
(c) SEG-PCA (d) SEG-MI

Figura 6.7: Comparativa de distribuciones de características para los problemas VOW y SEG, donde VOW-PCA (a), VOW-MI(b), SEG-PCA(c) y SEG-PCA(d) tienen una gran discrepancia en sus rasgos.

	$H0_1$**: Rank sum**	$H0_2$**: T-Test**
Alpha	0.6	0.4
p-value	0.055	0.037
Confidence level	**94.50 %**	**96.27 %**

Tabla 6.2: Resultados de las pruebas estadísticas.

para los problemas PNTL son 7,86 y 7,75. *Rank sum* dio un valor p de 0,055, lo que nos permite rechazar $H0_1$ en el nivel de confianza de 94,50 %. Por otro lado, *t-test* dio un valor p de 0,037, lo que nos permite rechazar el $H0_1$ en el nivel de confianza de 96,27 %. Se puede afirmar, dados estos resultados, que los grupos PCTL y PNTL son distintos y que se pueden detectar pares de problemas de cada grupo basado en la geometría de la representación del espacio de características extraída por DeepInsight.



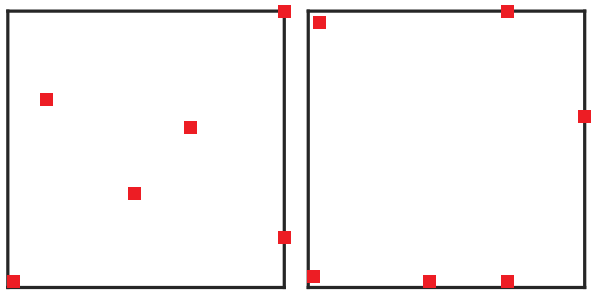
(a) YST-PCA (b) YST-MI

(c) IM10-PCA (d) IM10-MI

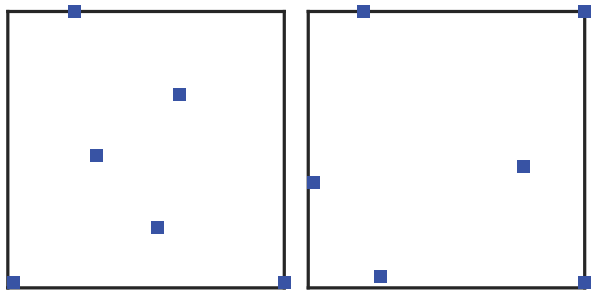
Figura 6.8: Comparación de distribución de características YST con reducción PCA (a) y MI (b) contra IM10 con reducción PCA (a) y MI (b), donde las distribuciones muestran una gran discrepancia.

	TLCP	TLNP
Promedio	**6.79**	7.86
Mediana	**6.90**	7.75
Desviación estándar	1.25	1.13
Q1	**5.97**	6.79
Q3	**7.76**	8.67

Tabla 6.3: Comparativa de valores estadísticos de los grupos TLCP y TLNP.

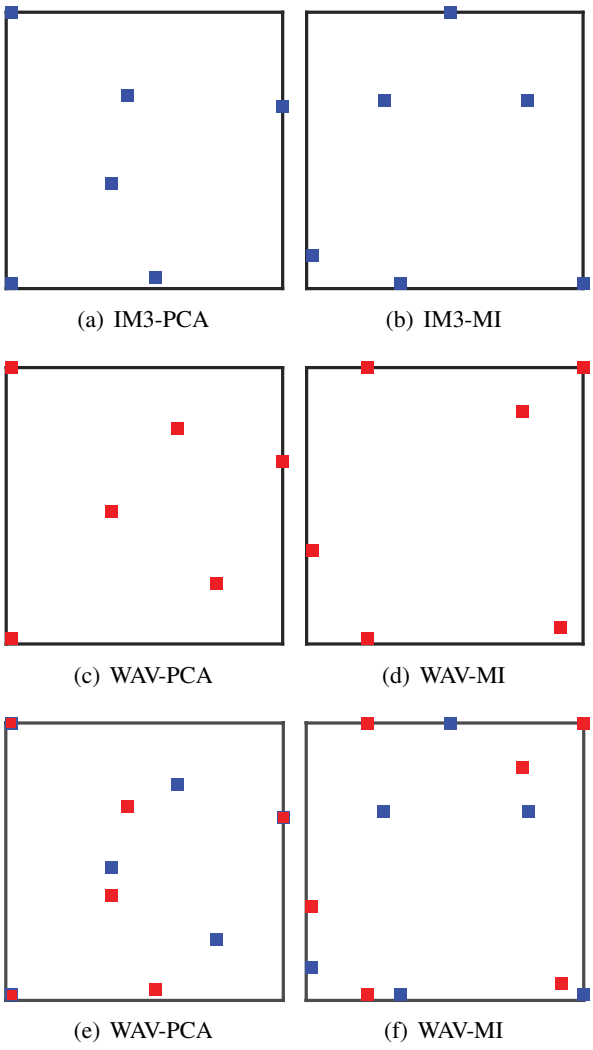


(a) IM10-PCA (b) IM10-MI



(c) SEG-PCA (d) SEG-MI

Figura 6.9: Comparación de distribución de características YST con reducción PCA (a) y MI (b) contra IM10 con reducción PCA (a) y MI (b), donde las distribuciones muestran una gran discrepancia.

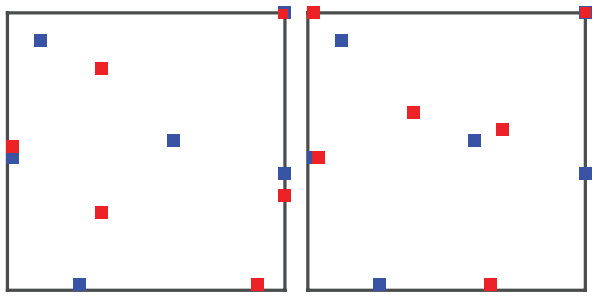


(a) IM3-PCA (b) IM3-MI

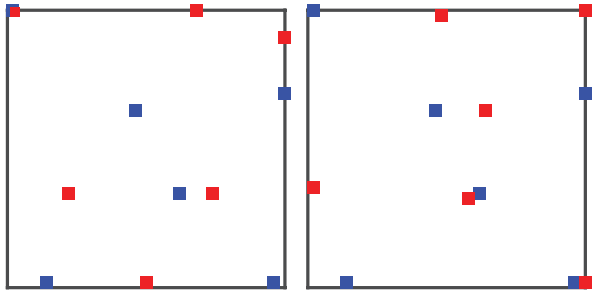
(c) WAV-PCA (d) WAV-MI

(e) WAV-PCA (f) WAV-MI

Figura 6.10: Representación del espacio de características de DeepInsight para los problemas de WAV e IM3 en las dos primeras filas. El ICP para dos casos de ejemplo (de 8 posibilidades) en la última fila. La distancia ICP para (e) es 5,6636 y para (f) es 7,7829, mientras que la DMR para el par problema es 5,66. Se puede observar que, en algunos registros, algunas de las características de diferentes problemas se superponen.

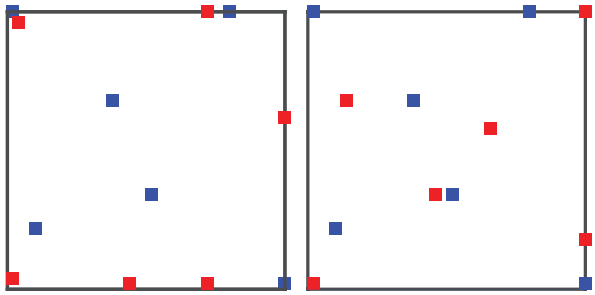


(a) ICP: HRT-MI a IM3-MI (b) ICP: HRT-MI a IM3-PCA



(c) ICP: HRT-PCA a IM3-MI (d) ICP: HRT-PCA a IM3-PCA

Figura 6.11: La distancia ICP para (a) es 7,9256, (b) es 9,6748,(c) es 10,1923 y (d) es 6,7230 mientras que la DMR para el par problema es 6,72. En algunos registros, algunos rasgos de diferentes problemas se superponen.



(a) VOW-PCA to IM10-MI (b) IM10-MI to VOW-PCA

Figura 6.12: La distancia ICP para (a) es 9,9294 y para (b) es 9,6608, mientras que la DMR para el par problema es 9,66, mostrando claramente que el espacio de características de estos dos problemas no pueden ser alineados.

Capítulo 7

Conclusiones y Trabajo Futuro

Este trabajo presenta un análisis de TL para un sistema GP, proponiendo un método para determinar cuándo dos problemas podrían ser buenos candidatos para la aplicación de TL. Esto se hizo analizando la estructura del espacio de características de cada problema (donador y receptor), extraído por la herramienta DeepInsight, la cual generó imágenes que representaban dicho espacio, esta herramienta transformó los resultados obtenidos por [11] de sus experimentos de TL para tareas de clasificación con GP.

Al analizar cada posible par de problemas se observó que estos podían ser clasificados en uno de dos grupos: el primer grupo son problemas compatibles con TL que incluye todos los pares de problemas en los que TL tuvo mucho éxito (PCTL), y el segundo son los problemas de TL no compatibles en los que TL tuvo un desempeño inferior de TL (PNTL).

Alineando sus respectivos espacios de características de estos grupos se pudo calcular una distancia entre ellos utilizando el algoritmo ICP, donde para cada par de problemas se tomó DMR para realizar un análisis estadístico el cual nos indica si estos grupos son diferentes. Al utilizar pruebas estadísticas como *rank sum* y *t-test* los resultados muestran que, de hecho, es posible distinguir entre estos grupos de pares de problemas.

7.1. Conclusiones

Las imágenes que se obtienen por medio de DeepInsight, han mostrado ser de gran utilidad para analizar el desempeño que tendrá TL, dado dos problemas con una reducción de características de dominios diferentes.

Las imágenes generadas de los problemas de [11] nos permiten observar cuando dos problemas serán compatibles, por medio del acomodo de sus rasgos dentro de la imagen generada, a esto le llamamos espacio de características. Estos rasgos corresponden a las características de cada problema de la Tabla 4.1, por lo tanto, entre mayor cantidad de características, mayor será la cantidad de rasgos en una imagen.

Cada problema se caracteriza por una representación del espacio de rasgos extraída mediante DeepInsight, con esto se plantea la hipótesis de que cuando dos problemas comparten una geometría de espacio de características similar, entonces TL se puede aplicar con éxito entre ellos.

Los resultados muestran que alineando la representación espacial de características de ambos problemas, es posible calcular una medida de distancia entre ellos y usar esta distancia para caracterizar el par de problemas. Como se observa en la Tabla 6.1, esa compatibilidad puede ser medida aplicando el algoritmo ICP, donde entre menor sea el error de sus distancias significa que dos problemas tendrán una mayor compatibilidad.

Estos valores ayudaron a distinguir grupos dentro de los problemas, en particular, clasificamos los pares de problemas en dos grupos, TLCP contiene pares de problemas que se consideran altamente compatibles en función del rendimiento de TL entre ellos, mientras que TLNP contiene pares de problemas en los que TL no tuvo éxito o sólo parcialmente.

En las pruebas estadísticas de los grupos PCTL se observan valores numéricos menores a comparación de los del grupo PNTL, como se puede apreciar en la Tabla 6.3. Podemos decir que estos grupos sí tienen una clara distinción gracias a las pruebas estadísticas *rank sum* ($H0_1$) y *t-test* ($H0_2$), con las cuales se pudieron rechazar las hipótesis propuestas con un rango de confianza de 94,50 % para $H0_1$ y 96,27 % para $H0_2$. Las pruebas de hipótesis estadísticas muestran claramente que nuestro enfoque propuesto se puede utilizar para diferenciar entre ambos grupos.

Podemos decir que dos problemas son parecidos cuando los rasgos en el espacio de características se encuentran cercanos del vecindario del problema receptor. Si las muestras comparten rasgos en las mismas áreas podemos suponer que tendrán un cierto grado de compatibilidad que tendrá como resultado un TL favorable.

7.2. TRABAJO A FUTURO

Como trabajo a futuro, se propone realizar el análisis de los problemas de regresión lineal con la misma metodología que este trabajo ha propuesto. En el caso de estudio presentado en el Capítulo 4 utilizan M3GP como un método de CI, utilizando PCA y MI para reducir las dimensiones de los problemas que se ven en la Tabla 4.1; con el método presentado en esta tesis, se pueden analizar problemas con CIs y reducciones diferentes, esto con el fin de comenzar a crear un *benchmark* del método propuesto.

M3GP puede tener una participación aún más útil, esto sería generar transformaciones que puedan garantizar la compatibilidad de TL con la ayuda de DeepIn-

sight, ya que una función de M3GP es encontrar espacios de rasgos que faciliten el aprendizaje. Con ayuda de DeepInsight este nuevo espacio se podría evaluar con el fin de ir configurando el espacio de rasgos para facilitar la transferencia entre dos problemas.

En este trabajo se analizaron los datos en un punto específico como se muestra en la Figura 4.2, esto significa que el enfoque puede fácilmente cambiar a otro punto de la metodología de [11], por ejemplo, se puede comparar el problema después del proceso CI y ver cómo este espacio de características se relaciona con el espacio de características del problema receptor con reducción PCA ó MI.

Sin embargo, el análisis se puede realizar tomando en cuenta otros aspectos de los problemas para medir su similitud, no sólo los dados por la representación del espacio de rasgos generado por DeepInsight, como lo proponen en [11]. Ejemplos de esto serían:

- **Antes de aplicar la reducción de características**: Esto con el fin de observar la distribución de valores de las características, tanto en el problema donador y receptor (por ejemplo, valores mínimos y máximos, media, mediana, varianza y otras medidas estadísticas), tratando de encontrar diferencias o similitudes entre los problemas.

- **Al reducir funciones**: Registrar qué tan grande es esta reducción, considerando la cantidad de características originales y también la cantidad de características que M3GP crea automáticamente. Esto debe hacerse tanto para el problema donador y receptor, buscando diferencias y similitudes. La cantidad de reducción a considerar debe basarse no sólo en el número de características, sino también en la pérdida de información resultante, para todas las clases juntas y también para cada clase por separado.

- **Después de la reducción de características, antes de cualquier transformación:** Uno debe realizar el mismo conjunto de análisis realizado en el punto 1, ahora también buscando diferencias y similitudes entre las medidas obtenidas con las características originales y con el conjunto de características reducido. Esto puede permitir la identificación de características problemáticas resultantes de la reducción de características.

- **Después de evolucionar la transformación:** Se debe observar la distribución de los valores de las características evolucionadas, tanto en el problema donador como en el receptor y medir la distancia entre ellos, no sólo entre sus centroides, sino también las distancias mínima y máxima entre sus elementos. Se deben utilizar medidas de similitud y disimilitud para comparar las agrupaciones obtenidas los problemas donador y receptor.

Otro enfoque sería eliminar un nodo del modelo y generar su imagen utilizando DeepInsight, tanto del problema donador como del problema receptor, para después mapear su distribución de características, observar cuáles características empatan y analizar si con menos características mejora la predicción de una transferencia.

Finalmente, se requiere encontrar una mejor manera de medir las similitudes entre problemas para tener una mejor estimación, ya que el algoritmo ICP ignora cuál problema es el receptor y donador.

Referencias

[1] T. M. Mitchell, *Machine Learning*. New York: McGraw-Hill, 1997.

[2] I. Goodfellow, Y. Bengio y A. Courville, *Deep Learning*. MIT Press, 2016, `http://www.deeplearningbook.org`.

[3] I. Redko, E. Morvant, A. Habrard, M. Sebban e Y. Bennani, "A survey on domain adaptation theory," *CoRR*, vol. abs/2004.11829, 2020.

[4] K. Weiss, T. M. Khoshgoftaar y D. Wang, "A survey of transfer learning," *Journal of Big Data*, vol. 3, n.º 1, pág. 9, 2016.

[5] A. Taylor, I. Dusparic, E. G. López, S. Clarke y V. Cahill, "Accelerating Learning in multi-objective systems through Transfer Learning," en *2014 International Joint Conference on Neural Networks, IJCNN 2014, Beijing, China, July 6-11, 2014*, IEEE, 2014, págs. 2298-2305.

[6] M. Namazifar, A. Papangelis, G. Tür y D. Hakkani-Tür, "Language Model is All You Need: Natural Language Understanding as Question Answering," *CoRR*, vol. abs/2011.03023, 2020.

[7] Y. Lu, L. Luo, D. Huang, Y. Wang y L. Chen, "Knowledge Transfer in Vision Recognition: A Survey," *ACM Comput. Surv.*, vol. 53, n.º 2, abr. de 2020.

[8] M. Loey, G. Manogaran y N. E. M. Khalifa, "A deep transfer learning model with classical data augmentation and CGAN to detect COVID-19 from chest CT radiography digital images," *Neural Computing and Applications*, oct. de 2020.

[9] E. Galván and P. Mooney, *Neuroevolution in Deep Neural Networks: Current Trends and Future Challenges*, arXiv preprint arXiv: 2006.05415, 2020.

[10] B. Bhattacharjee, J. R. Render, M. Hill y col., "P2L: Predicting Transfer Learning for Images and Semantic Relations," en *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2020, págs. 3284-3293.

[11] L. Muñoz, "Transformaciones multidimensionales del espacio de rasgos con programación genética," Tesis doct., TecNM instituto tecnológico de Tijuana departamento de ingeniería eléctrica y electrónica, 2019.

[12] A. Sharma, E. Vans, D. Shigemizu, K. A. Boroevich y T. Tsunoda, "DeepInsight: A methodology to transform a non-image data to an image for convolution neural network architecture," *Scientific Reports*, vol. 9, n.º 1, pág. 11 399, 2019.

[13] I. Hidalgo y C. Cervigon, "Una revisión de los algoritmos evolutivos y sus aplicaciones," ene. de 2004.

[14] R. Poli, W. B. Langdon y N. F. McPhee, *A Field Guide to Genetic Programming*. Lulu Enterprises, UK Ltd, 2008.

[15] L. Muñoz, L. Trujillo y S. Silva, "Transfer learning in constructive induction with Genetic Programming," *Genetic Programming and Evolvable Machines*, vol. 21, dic. de 2020.

[16] I. Kuscu, F. Brighton y B. Qh, "Constructive Induction using Genetic Programming," jun. de 1996.

[17] J. R. Koza, "Human-competitive results produced by genetic programming," *Genetic Programming and Evolvable Machines*, vol. 11, n.º 3, págs. 251-284, 2010.

[18] V. Ingalalli, S. Silva, M. Castelli y L. Vanneschi, "A Multi-dimensional Genetic Programming Approach for Multi-class Classification Problems," en *Genetic Programming*, M. Nicolau, K. Krawiec, M. I. Heywood y col., eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2014, págs. 48-60.

[19] L. Muñoz, S. Silva y L. Trujillo, "M3GP – Multiclass Classification with GP," vol. 9025, abr. de 2015, págs. 78-91.

[20] J. Batista, A. Cabral, M. Vasconcelos, L. Vanneschi y S. Silva, "Improving Land Cover Classification Using Genetic Programming for Feature Construction," *Remote Sensing*, vol. 13, pág. 1623, abr. de 2021.

[21] Y. Yang, X. Wang, X. Zhao, M. Huang y Q. Zhu, "M3GPSpectra: A novel approach integrating variable selection/construction and MLR modeling for quantitative spectral analysis," *Analytica Chimica Acta*, vol. 1160, pág. 338 453, mar. de 2021.

[22] W. La Cava y J. Moore, "Learning feature spaces for regression with genetic programming," *Genetic Programming and Evolvable Machines*, vol. 21, sep. de 2020.

[23] N. Rodrigues, J. Batista y S. Silva, "Ensemble Genetic Programming," en abr. de 2020, págs. 151-166.

[24] W. La Cava, S. Silva, K. Danai, L. Spector, L. Vanneschi y J. Moore, "Multidimensional genetic programming for multiclass classification," *Swarm and Evolutionary Computation*, vol. 44, abr. de 2018.

[25] L. Yang, S. Hanneke y J. Carbonell, "A theory of transfer learning with applications to active learning," *Machine Learning*, vol. 90, n.º 2, págs. 161-189, 2013.

[26] S. J. Pan y Q. Yang, "A Survey on Transfer Learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, n.º 10, págs. 1345-1359, 2010.

[27] M. Namazifar, A. Papangelis, G. Tur y D. Hakkani-Tür, *Language Model is All You Need: Natural Language Understanding as Question Answering*, 2020.

[28] R. Mehrotra, M. Ansari, R. Agrawal y R. Anand, "A Transfer Learning approach for AI-based classification of brain tumors," *Machine Learning with Applications*, vol. 2, pág. 100 003, 2020.

[29] Y. Fregier y J.-B. Gouray, "Mind2Mind: Transfer Learning for GANs," en jul. de 2021, págs. 851-859.

[30] S. Ren y C. Q. Li, "Robustness of transfer learning to image degradation," *Expert Systems with Applications*, vol. 187, pág. 115 877, 2022.

[31] M. Loey, G. Manogaran y N. E. M. Khalifa, "A deep transfer learning model with classical data augmentation and CGAN to detect COVID-19 from chest CT radiography digital images," *Neural Computing and Applications*, 2020.

[32] W. B. Langdon y J. P. Nordin, "Seeding Genetic Programming Populations," en *Genetic Programming*, R. Poli, W. Banzhaf, W. B. Langdon, J. Miller, P. Nordin y T. C. Fogarty, eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2000, págs. 304-315.

[33] P. Stone y M. Veloso, "Layered Learning," en *Machine Learning: ECML 2000 (Proceedings of the Eleventh European Conference on Machine Learning)*, R. L. de Mántaras y E. Plaza, eds., Barcelona,Catalonia,Spain: Springer Verlag, 2000, págs. 369-381.

[34] T. T. H. Dinh, T. H. Chu y N. Q. Uy, "Transfer learning in genetic programming," *2015 IEEE Congress on Evolutionary Computation (CEC)*, págs. 1145-1151, 2015.

[35] M. Iqbal, H. Al-Sahaf, B. Xue y M. Zhang, "Genetic programming with transfer learning for texture image classification," *Soft Computing*, vol. 23, n.º 23, págs. 12 859-12 871, feb. de 2019.

[36] W. Fu, B. Xue, M. Zhang y X. Gao, "Transductive Transfer Learning in Genetic Programming for Document Classification," en *Simulated Evolution and Learning*, Y. Shi y col., eds., Cham: Springer International Publishing, 2017, págs. 556-568.

[37] A. Ekárt, A. Patelli, V. Lush y E. Ilie-Zudor, "Genetic Programming with Transfer Learning for Urban Traffic Modelling and Prediction," en *2020 IEEE Congress on Evolutionary Computation (CEC)*, 2020, págs. 1-8.

[38] T. Helmuth, E. Pantridge, G. Woolson y L. Spector, "Transfer Learning of Genetic Programming Instruction Sets," ép. GECCO '20, Cancún, Mexico: Association for Computing Machinery, 2020, 241–242.

[39] L. Spector, "Autoconstructive evolution: Push, pushGP, and pushpop," en *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001)*, vol. 137, 2001.

[40] J. Alcalá-Fdez, A. Fernández, J. Luengo, J. Derrac y S. García, "KEEL Data-Mining Software Tool: Data Set Repository, Integration of Algorithms and Experimental Analysis Framework.," *Multiple-Valued Logic and Soft Computing*, vol. 17, n.º 2-3, págs. 255-287, 2011.

[41] K. Bache y M. Lichman, *UCI Machine Learning Repository*, 2013.

[42] G. Survey, *U.S. geological survey (usgs) earth resources observation systems (eros) data center (edc)*.

[43] Y. Martínez, L. Trujillo, P. Legrand y E. Galván-López, "Prediction of expected performance for a genetic programming classifier," *Genetic Programming and Evolvable Machines*, vol. 17, n.º 4, págs. 409-449, feb. de 2016.