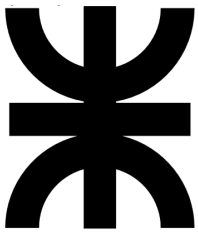

Universidad Tecnológica Nacional

Tesis de Maestría

Generación de datos sintéticos para selección de características con algoritmos genéticos

Autor:
Claudio Sebastian Castillo

Directores:
Matías Gerard y Leandro Vignolo

Tesis presentada en cumplimiento de los requisitos para el grado de Maestría en Minería de Datos

en

UTN
Seccional Paraná

Septiembre, 2024

A mi corazón de 4/4; Morella, Sofía, Joquín y Manuel. A mi musa Verónica.

UNIVERSIDAD TECNOLÓGICA NACIONAL

Seccional Paraná

Maestría en Minería de Datos

Resumen

Generación de datos sintéticos para selección de características con algoritmos genéticos

Claudio Sebastian Castillo

La disponibilidad de datos muestrales afecta los procesos de selección de características, y resulta particularmente condicionante en escenarios de alta dimensionalidad y bajo número de muestras. En el caso de selección de características mediante Algoritmos Genéticos la falta de datos impacta negativamente en la función de aptitud, y de esa forma limita la eficacia del algoritmo. Por eso, la técnica de aumentación de datos mediante Autocodificadores Variacionales plantea una posible solución a este problema, ofreciendo distintas alternativas de implementación en el contexto de los Algoritmos Genéticos.

Introducción

En el campo del aprendizaje automático, la selección de características es una tarea crítica que puede determinar el éxito o fracaso de un modelo predictivo. La alta dimensionalidad y la complejidad inherente a los conjuntos de datos reales hacen que la selección de un subconjunto óptimo de características sea, muchas veces, un paso ineludible para el aprendizaje efectivo.

En este contexto, los Algoritmos Genéticos (AGs) se han consolidado como una herramienta poderosa para resolver problemas de optimización complejos, incluida la selección de características. Estos algoritmos, inspirados en la evolución natural, son capaces de explorar grandes espacios de búsqueda de manera efectiva, proporcionando soluciones cercanas al óptimo en una variedad de escenarios. Por ello, los AGs han sido ampliamente utilizados en problemas de selección, demostrando su eficacia en la identificación de subconjuntos relevantes de características en datos de alta dimensionalidad.

Sin embargo, la eficacia de los AGs depende de la disponibilidad de suficientes datos para evaluar las soluciones en competencia. En contextos donde los datos son limitados, los AGs pueden verse afectados en su capacidad discriminativa, produciendo soluciones subóptimas o inestables. Esta limitación es especialmente crítica en problemas de alta dimensionalidad y bajo número de muestras, donde la función objetivo que guía la búsqueda de soluciones puede degradarse significativamente.

Por esta razón, la investigación de estrategias que mitiguen las limitaciones impuestas por la escasez de datos se ha convertido en un área de interés creciente en el subcampo de la selección de características. Una de las técnicas emergentes en este ámbito es la aumentación de datos mediante Autoencodificadores Variacionales (AVs). Los AVs, como modelos generativos, tienen la capacidad de crear muestras sintéticas que mantienen las propiedades fundamentales de los datos originales, conviertiéndolos en una herramienta prometedora para mejorar la capacidad de los AGs en la selección de características.

El problema central de la tesis que aquí presentamos gira, precísamente, en la restricciones que la escacez de datos impone a los AGs, y cómo superarlas usando AVs. Así, la pregunta central del trabajo es: ¿cómo puede la aumentación de datos mediante autoencodificadores variacionales mejorar el desempeño de los algoritmos genéticos en la selección de características?

Cabe destacar que este desafío y su eventual solución son importantes por varias razones. La selección de características no solo condiciona, como ya se mencionó, la precisión de los modelos predictivos, también afecta la eficiencia computacional y la interpretabilidad de los resultados. En problemas de alta dimensionalidad, la posibilidad de reducir el número de características relevantes sin perder información útil puede marcar la diferencia entre un modelo efectivo y uno ineficaz, entre uno interpretable y uno de caja negra. Por lo tanto, mejorar este proceso mediante la integración de técnicas de aumentación de datos puede tener un impacto significativo en diversas aplicaciones prácticas, desde la biología molecular hasta la ingeniería y las ciencias sociales.

La hipótesis que hemos llevado a prueba ha sido que la aumentación de datos mediante AVs mejora la capacidad discriminativa de los AGs, permitiendo la identificación de subconjuntos de características más relevantes y estables en contextos de escasez muestral. Para evaluarla, hemos propuesto un trabajo experimental que exploró la

integración de estas dos técnicas en un marco unificado. La idea de combinar la generación de datos sintéticos mediante AVs con la selección de características mediante AGs, estaba orientada a buscar combinaciones sinérgicas y eficaces entre modelos que mejorasen la selección de características. A estos fines, trabajamos con cinco conjuntos de datos de referencia, representativos de distintos contextos y niveles de complejidad, para evaluar el desempeño de los modelos propuestos.

A lo largo de este documento, describiremos el proceso de investigación llevado a cabo, desde los estudios iniciales hasta los experimentos finales, pasando por el diseño y construcción de un modelo genérico de AV, y su adaptación a los datasets elegidos y la creación de una estructura combinada de AV + AG para la selección de características. Los resultados obtenidos en cada etapa se analizarán y discutirán en detalle, con el objetivo de identificar las ventajas y limitaciones de la propuesta, así como posibles áreas de mejora y futuros trabajos.

Al finalizar el documento, esperamos poder justificar la eficacia de la aumentación de datos mediante AVs en la selección de características, demostrando que esta técnica puede mejorar significativamente el desempeño de los AGs en contextos de escasez. Además, esperamos identificar las condiciones y contextos en los que esta técnica es más efectiva.

El documento está estructurado de la siguiente manera: en el capítulo 1 se presenta el problema de la selección de características en contextos de alta dimensionalidad, desbalance y escacez de datos. En el capítulo 2 se hace una revisión de modelos clásicos aplicados a los datos que forman parte de nuestra investigación, su evaluación y resultados. En el capítulo 3 se presentan los modelos AVs, sus bases teóricas y los experimentos realizados para construir la arquitectura final empleada en nuestra investigación. En el capítulo 4 se aborda la integración de los modelos AVs y AGs en una estructura combinada, su configuración y los resultados experimentales obtenidos de su aplicación. Finalmente, en el capítulo 5, se presentan las conclusiones de la investigación, así como posibles líneas futuras de trabajo.

Por último, en tiempos de grandes debates sobre los riesgos de los modelos generativos, esperamos poder brindar a nuestros lectores un ejemplo funcional de su estructura, iluminar ciertos principios y evidenciar sus límites. Si nuestro trabajo permite resaltar la sobriedad de sus mecanismos internos, habremos cumplido el anhelo de hacer menos opaca su belleza y más clara sus posibilidades.

Índice

Resumen III

Introducción V

1. El problema de la selección de características **1**
1.1. Estrés, ignorancia y selección de características 1
1.2. Las multiples cara de los datos problemáticos 3
1.3. Distintos enfoques para la selección de características 6
1.4. Selección, *algoritmos genéticos* y datos sintéticos 8

2. Modelos clásicos aplicados al espacio completo de características **13**
2.1. Datos elegidos en nuestro estudio 13
2.2. El desempeño de algoritmos clásicos 15
2.3. Resultados Obtenidos 18

3. Autocodificadores Variacionales y datos sintéticos **23**
3.1. Modelos generativos 23
3.2. Autocodificadores 24
3.3. Autocodificadores y el problema de la generación de datos 25
3.4. Autocodificadores Variacionales 26
3.5. Presentación de nuestros modelos de AV y AVC 29
3.6. Configuración de modelos 33
3.7. Resultados obtenidos en los experimentos 34
3.8. Otras consideraciones emergentes de los experimentos 38
3.9. Conclusiones 39

4. Algoritmos Genéticos **41**
4.1. Elementos básicos de los Algoritmos Genéticos 41
4.2. Implementación de un Algoritmo Genético para la selección de características 42

5. Próximos Pasos: complejización del modelo y exploración de nuevas arquitecturas **47**
5.1. 1. Complejización del modelo AV 47
5.2. 2. Integración AV-AG optimizada 48
5.3. 3. Exploración de encadenamientos y stacks de modelos 49
5.4. 4. Exploración de arquitecturas híbridas y meta-aprendizaje 49
5.5. Conclusión 50

References **51**

Appendices **53**

A. Algoritmo Genético, teoría y implementación **55**
A.1. a) Codificación del espacio de búsqueda 55
A.2. b) Búsqueda por población de soluciones 57
A.3. c) Función de aptitud y evaluación de soluciones 58
A.4. d) Operadores estocásticos y *esquemas* genéticos 59
A.5. Implementación de un Algoritmo Genético 62
A.6. Algoritmo Genético con la librería DEAP 63

Lista de figuras

1.1. Diagrama de enfoques para la selección de características 7

2.1. algoritmosclasicos 19

3.1. Ejemplo esquemático de un autocodificador 24
3.2. Discontinuidad del espacio latente 25
3.3. Autocoficadores Variacionales 26
3.4. MLP con datos sintéticos vs MLP con datos reales 35
3.5. Exactitud según cantidad de muestras sintéticas 39

4.1. Diagrama de un Algoritmo Genético 43

Lista de tablas

Capítulo 1

El problema de la selección de características

En el presente capítulo abordamos el problema de la selección de características, su importancia y desafíos. En ese marco, repasaremos ciertas dificultades asociadas a los datos, como por ejemplo la alta dimensionalidad y el desbalance de clases, y veremos cómo ellas impactan en la selección de características. Luego describiremos brevemente distintos enfoques para la selección de características, como así también ventajas y desventajas de cada uno. Finalmente, vincularemos la selección de características con el aporte fundamental de nuestro trabajo asociado a la generación sintética de datos.

1.1. Estrés, ignorancia y selección de características

Un punto de partida difícil de controvertir en el mundo actual del aprendizaje automático es que la cantidad de información disponible para investigar ha crecido dramáticamente en los últimos veinte años (Li and Zhang 2023). Conforme la vida se digitaliza, y la información almacenada en los sistemas aumenta, la generación de conocimiento parece menos determinada por el viejo problema de la escasez de factores y más por su nueva situación de abundancia.

En este contexto, el desafío de trabajar con datos de alta dimensionalidad ha motivado la aparición de una serie de técnicas dirigidas a generar conocimiento y resultados precisos en escenarios complejos debido a la abundancia de información. Así, bajo el nombre de *selección de características* nació un area de estudio que busca resolver, precisamente, el problema de discernir *sistemáticamente* la información relevante de aquella que no lo es cuando se trabaja con muchas variables.

Según Bolón-Canedo y ot., el origen de este campo se remonta a los años '60, cuando Hughes, usando un modelo paramétrico, estudió la precisión de un clasificador bayesiano en función del número de características utilizado para predecir una variable objetivo (Bolón-Canedo, Sánchez-Maroño, and Alonso-Betanzos 2015). Por nuestra parte, entendemos que incluso se puede ir más lejos, a los años '30, cuando Fisher reprochaba el *estrés* con el que se buscaba conocimiento manipulando variables aisladas en lugar de mirar sus interacciones, proponiendo una *investigación experimental* más *abarcativa* y *eficiente* que permitiera obtener mayor conocimiento con la misma cantidad de observaciones (Fisher 1935).

Más allá de su origen, el poderoso impulso de considerar espacios de búsqueda cada vez más amplios y problemas cada vez más complejos, ha contribuido a que la selección de características se convierta en un campo de estudio activo e importante.

En efecto, hoy no es extraño encontrar investigaciones donde los objetos de estudio superen las decenas de miles de dimensiones, como sucede en la investigación biomédica. Allí, los microarrays de ADN son ejemplos representativos de datos de alta dimensionalidad, que constituyen fuentes vitales de información en problemas que involucran expresión génica (Almugren and Alshamlan 2019). En sentido similar, datos de video -presentes en multiples aplicaciones-, datos financieros, información vinculada a interacciones sociales, entre otros, son ejemplos del tipo de problema que justifican el uso de técnicas de selección de características (El-Hasnony et al. 2020).

Podemos definir a la *selección de características* como *el proceso de detectar las características relevantes y descartar aquellas irrelevantes o redundantes, con el objetivo de obtener un subconjunto que describa adecuadamente un problema dado con una degradación mínima del rendimiento* (Bolón-Canedo, Sánchez-Maroño, and Alonso-Betanzos 2015). Aquí rendimiento se refiere a la medida de evaluación empleada para juzgar los resultados del proceso de selección. Los distintos enfoques para selección de características adhieren a la premisa de que podemos separar la información relevante de aquella que no lo es para predecir una variable objetivo, y por ende mejorar el desempeño de los modelos de aprendizaje.

Nótese aquí que el *proceso* de selección de características es un proceso sistemático. Gran parte de su esfuerzo se centra en disponer de un método para discriminar, de manera precisa y controlada, la información relevante de la irrelevante o redundante. La precisión y el control destacan como dos características fundamentales debido a que la selección de características tiene lugar en un escenarios donde se ignora cual es el aporte de cada variable a la resolución del problema. Por eso, resulta necesario contar con un criterio de evaluación claramente definido que permitan examinar de forma objetiva y cuantificable el valor real de cada variable y sus interacciones.

Bajo esa perspectiva, la selección de características busca determinar un subconjunto de atributos que satisfaga uno de los siguientes criterios(Vignolo and Gerard 2017):

1. El subconjunto con un tamaño especificado que maximice la precisión de la predicción.
2. El subconjunto de menor tamaño que satisfaga un requisito de precisión mínima.
3. El subconjunto que logre el mejor compromiso entre dimensionalidad y precisión.

El criterio a elegir dependerá de los objetivos del estudio y de las características del problema. Mas allá de eso, es fácil advertir que la selección de características supone como ventaja la reducción del espacio de búsqueda, y en cierto sentido es un remedio a la alta dimensionalidad.

En efecto, en el contexto del aprendizaje automático es común enfrentar problemas representados por grandes conjuntos de variables, vicisitud que se asocia con la *maldición de la dimensionalidad* (Bolón-Canedo, Sánchez-Maroño, and Alonso-Betanzos 2015). En general, este fenómeno se presenta por la alta demanda computacional y costos asociados con la optimización de dichos espacios, volviendo a ciertos problemas intratables desde el punto de vista práctico. Para abordarlo, una alternativa posible es la reducción de dimensionalidad, que consiste en encontrar o construir matrices con menor número de columnas que las originales pero la misma carga de información relevante. Esas matrices de menor dimensión, puede usarse de manera más eficiente para modelar el problema objetivo, facilitando su interpretación y comunicación. El proceso de encontrar estas matrices se llama *reducción de dimensionalidad*, y uno de los métodos para lograrlo es la selección de características.

Dicho lo anterior, es preciso reconocer que la abundancia de información trae consigo una serie de desafíos que impactan en la selección de características. En el siguiente apartado describiremos aquellos más relevantes para nuestro estudio.

1.2. Las multiples cara de los datos problemáticos

1.2.1. Alta dimensionalidad y escasez muestral

En primer lugar, un problema frecuente en el aprendizaje automático se presenta cuando disponemos de pocas muestras en un espacio vectorial de alta dimensionalidad. Es decir, cuando el número de muestras (m) es menor que el número de características (n), siendo n particularmente grande. Como vimos, esta situación es común en el campo del análisis genético, el procesamiento de información médica, procesamiento de video, entre otros. Como veremos mas adelante, tres de los datasets elegidos para nuestro estudio -vinculados al ámbito biomédico- se encuentran en esta situación.

La escasez de muestras dificulta el tratamiento de la información, ya que la cantidad de datos disponibles para entrenar un modelo es insuficiente para capturar la variabilidad inherente a los datos. En tales circunstancias, el proceso de aprendizaje tiende a sufrir severas limitaciones, bien sea sobreajustándose a las observaciones disponibles (llevando a modelos poco generalizables), o bien resultando incapaz de capturar la estructura subyacente de los datos (lo que puede llevar a modelos poco precisos).

Por ejemplo, en el campo de la clasificación de perfiles de expresión génica, el problema se considera difícil de resolver debido a la complejidad que supone identificar los genes que contribuyen a la aparición de ciertas condiciones - como por ejemplo: cáncer, diabetes, entre otros (Almugren and Alshamlan 2019). Dada la gran cantidad de genes presentes en estos supuestos (muchos de ellos irrelevantes), entrenar un modelo con todos ellos puede conducir a resultados erróneos. Este desafío, comúnmente se ve acentuado por la baja cantidad de observaciones disponibles y por el hecho de que los genes se encuentran altamente correlacionados. Dichas características dificultan el adecuado funcionamiento de los métodos clásicos de aprendizaje automático, cuyo rendimiento se asocia a la cantidad de muestras disponibles.

Ante estos problemas, la selección de características tiene severas limitaciones, ya que la alta dimensionalidad y bajo número de muestras dificultan la distinción entre información relevante y irrelevante.

1.2.2. Desbalance de clases

El desbalance de clases constituye otro problema importante y frecuente en los desafíos actuales que se presentan para el aprendizaje automático. El mismo sucede cuando la distribución de las clases en el conjunto de datos es altamente desigual: típicamente una clase mayoritaria supera ampliamente las observaciones de una o más clases minoritarias. Este problema puede darse en combinación con alta dimensionalidad y escacez muestral, agravando la dificultad de selección de características.

El desbalance de clases es un problema relevante en aplicaciones donde la o las clases minoritarias son precisamente las de mayor interés, como por ejemplo: el diagnóstico de enfermedades raras, la detección de fraudes bancarios, la identificación de intrusiones en redes y la predicción de fallos en equipos técnicos. En supuestos como estos, los clasificadores que no contemplen un tratamiento explícito del problema tienden

a sesgarse hacia la clase mayoritaria, pudiendo alcanzar una alta precisión global mientras fallan en la detección de los casos más importantes pero menos frecuentes.

Para atacar este problema, se han propuesto diversas soluciones, que pueden categorizarse en tres grupos:

1. **Muestreo de datos**: Este enfoque modifica las muestras de entrenamiento para producir una distribución de clases más balanceada. Las técnicas tradicionales incluyen: submuestreo (*undersampling*) donde se cre un subconjunto del conjunto original eliminando instancias; sobremuestreo (*oversampling*) donde se gnera un superconjunto replicando instancias existentes o creando nuevas, y finalmente, métodos híbridos que combinan ambas técnicas de muestreo.
2. **Modificación algorítmica**: Este enfoque adapta los métodos de aprendizaje para que sean más sensibles a los problemas de desbalance. Por ejemplo, el uso de la distancia de Hellinger como criterio de división en árboles de decisión, que ha demostrado ser insensible al sesgo, capturando la divergencia en las distribuciones sin ser dominada por las probabilidades previas de clase.
3. **Aprendizaje sensible al costo**: Esta solución puede incorporar elementos a nivel de datos, a nivel de algoritmos, o ambos a la vez, asignando costos más altos a la clasificación errónea de ejemplos de la clase positiva (minoritaria). Así, en el diagnóstico médico, resulta más importante reconocer la presencia de una enfermedad rara que su ausencia, por ello el costo de un falso negativo es mayor que el de un falso positivo.

El desbalance de clases también supone un desafío para la selección de características. Ello es así, en la medida que dichos métodos suponen la búsqueda de aquellas dimensiones que maximizan la separación de clases en su conjunto, sin ponderar sus diferencias para capturar información para cada clase en particular. Esto ocurre porque, en escenarios con distribución fuertemente desequilibrada, las técnicas de evaluación tienden a priorizar la identificación de la clase mayoritaria, reduciendo la relevancia de las características que serían valiosas para discriminar a las minoritarias. Como resultado, se descarta información relevante para la predicción de los casos poco frecuentes, y se seleccionan características que no son representativas para todas las clases por igual.

Siguiendo a Bolón-Canedo y ot., junto a los problemas mencionados sobre alta dimensionalidad y desbalance de clases, la complejidad y el ruido de los datos también representan desafíos importantes en el tratamiento de la información que debemos considerar en el presente estudio.

1.2.3. Complejidad

Respecto a la complejidad, se refiere a la dificultad de identificar las fronteras de decisión entre clases, que pueden manifestarse en tres aspectos fundamentales (todos ellos pueden coexistir en un mismo problema):

1. **Ambigüedad de clases**: Surge cuando los casos no pueden distinguirse utilizando las características disponibles, ya sea porque los conceptos de clase están mal definidos y por lo tanto son intrínsecamente indistinguibles, o porque las características seleccionadas son insuficientes para discriminar las clases.
2. **Complejidad de fronteras**: Se refiere a situaciones donde los límites entre clases requieren una descripción extensa de la frontera de decisión, llegando a

demandar, en el caso extremos de complejidad, la enumeración exhaustiva de todos los puntos con sus etiquetas de clase. Este aspecto de dificultad se debe a la naturaleza del problema y no a la muestra o selección de características, indicando que una completa separación entre clases es un problema difícil de resolver.

3. **Dispersión de muestras**: La combinación de pocas muestras y alta dimensionalidad genera una dispersión que dificulta la generación de fronteras de decisión.

Anque los casos 1 y 2 son problemas ante los cuales la selección de características no puede ofrecer una solución efectiva, el caso 3, asociado con alta dimensionalidad y escasez muestral, podría ser mitigado por estrategias de aumentación de datos.

1.2.4. Ruido

Los datos del mundo real siempre están expuestos a imperfecciones, ruido, proveniente de entornos dinámicos donde las dimensiones de interés de un fenómeno coexisten en espacios de interacciones permanentes. Así, aún considerando un escenario artificial, completamente libre de interferencia, la imperfección puede provenir de diversas fuentes como dispositivos de medición defectuosos o limitados, errores de transcripción o irregularidades en la transmisión de información.

Ante tales circunstancias, existen cuatro enfoques principales para abordar estas imperfecciones en los conjuntos de datos:

1. **Conservación del ruido**: En este enfoque, se mantiene el conjunto de datos tal como está, con sus instancias ruidosas. Los algoritmos que utilizan los datos se diseñan para ser robustos, es decir, capaces de tolerar cierta cantidad de ruido. Una estrategia común es desarrollar algoritmos que eviten el sobreajuste al modelo (como sucede en el caso de los árboles de decisión) mediante técnicas de poda.

2. **Limpieza de datos**: Este método implica descartar las instancias que se consideran ruidosas según ciertos criterios de evaluación. El clasificador se construye utilizando únicamente las instancias retenidas en un conjunto de datos más pequeño pero más limpio. Sin embargo, este enfoque presenta dos debilidades significativas:

   - Al eliminar instancias completas, se puede descartar información potencialmente útil, como valores de características no corrompidos.
   - Cuando existe una gran cantidad de ruido, la información restante en el conjunto de datos limpio puede resultar insuficiente para construir un clasificador eficaz.

3. **Transformación de datos**: Este enfoque busca corregir las instancias ruidosas en lugar de eliminarlas. Las instancias identificadas como ruidosas se reparan reemplazando los valores corrompidos por otros más apropiados, y luego se reintroducen en el conjunto de datos.

4. **Reducción de datos**: Esta estrategia implica reducir la cantidad de datos mediante la agregación de valores o la eliminación y agrupación de atributos redundantes. La reducción de dimensionalidad -y consecuentemente la selección de características- es una de las técnicas más populares para eliminar características ruidosas (es decir, irrelevantes) y redundantes.

Vistos los problemas que enfrentamos con los datos, pasemos a describir, brevemente, los distintos enfoques que se han propuesto para sortearlos empleando estrategias de selección de características.

1.3. Distintos enfoques para la selección de características

Podemos agrupar los métodos de selección de características en tres grandes grupos:

- Filtros,
- Wrappers o métodos envolventes, y
- Embedded o métodos embebidos

Los métodos de **Filtro** se basan en las características generales de los datos de entrenamiento y realizan la selección de características como un paso de preprocesamiento independiente del algoritmo de aprendizaje. El análisis de relevancia de una característica se realiza considerando sus propiedades intrínsecas, sin determinar sus relaciones posibles con otras. El resultado de dicho análisis es una lista de características ordenadas por relevancia (ranking), donde el subconjunto final de características se selecciona según ese orden. Este enfoque es ventajoso por su bajo costo computacional, rapidez y escalabilidad, pero deja sin resolver el problema apuntado por Fisher sobre la interacción entre variables. Los metodos de filtro se pueden sub-clasificar en univariados y multivariados (Solorio-Fernández, Carrasco-Ochoa, and Martínez-Trinidad 2020). Los primeros analizan cada característica de manera independiente con el fin de obtener una lista ordenada. Este tipo de métodos puede identificar y eliminar eficazmente características irrelevantes, pero no son capaces de eliminar las redundantes, ya que no consideran posibles dependencias entre las características. Ejemplos de estos métodos son: la evaluación utilizando la distribución *chi-cuadrado* , la *ganancia de información*, entre muchos otros (Bolón-Canedo, Sánchez-Maroño, and Alonso-Betanzos 2015). Los métodos filtro multivariados evalúan la relevancia de las características de forma conjunta en lugar de hacerlo individualmente. Por ejemplo, el método de selección hacia adelante (forward selection) y hacia atrás (backward selection) son métodos de filtro multivariados que sucesivamente agregan y eliminan características para obtener un subconjunto óptimo. Los métodos multivariados pueden manejar características redundantes e irrelevantes; por lo tanto, en muchos casos, la precisión alcanzada por los modelos empleando subconjuntos seleccionados con estos métodos puede ser mayor. Otros métodos filtro multivariados puden ser el *método basado en el análisis de correlación* y *algoritmo ReliefF*, entre otros (Bolón-Canedo, Sánchez-Maroño, and Alonso-Betanzos 2015).

Los métodos **Wrappers o Envolventes** involucran un algoritmo de aprendizaje como caja negra y consisten en usar su resultado para evaluar la utilidad relativa de subconjuntos de características. Aquí, el algoritmo de selección utiliza el método de aprendizaje como una subrutina, para encontrar los subconjuntos de características más relevantes. Esta estrategia es capaz de reconocer variables relevantes y capturar dependencias entre ellas, pero a un costo computacional mayor que los otros métodos. En ese sentido enfrenta desafíos importante cuando el conjunto de datos es de alta dimensionalidad, ya que evaluar 2^n subconjuntos de características puede resultar en un problema intratable. Por esa razón, se recurre comunmente a heurísticas de búsqueda capaces de encontrar soluciones adecuadas sin explorar todo el espacio del problema (R. Zhang et al. 2019).

Los métodos **Embedded o Embebidos** realizan la selección de características durante el proceso de entrenamiento de un modelo de aprendizaje, y suelen ser específicos para determinados algoritmos (e.g. árboles de decisión y métodos derivados de ellos, eliminación recursiva de características mediante SVM, entre otros). A diferencia de los métodos de filtro y wrappers, los métodos embebidos no separan la selección de características del proceso de entrenamiento, sino que la realizan de manera simultánea. En este caso, optimizan una función de pérdida regularizada con respecto a dos conjuntos de parámetros: los parámetros del algoritmo de aprendizaje y los parámetros que indican las características seleccionadas.(Bolón-Canedo, Sánchez-Maroño, and Alonso-Betanzos 2015). Este enfoque es capaz de capturar dependencias a un costo computacional menor que los wrappers.

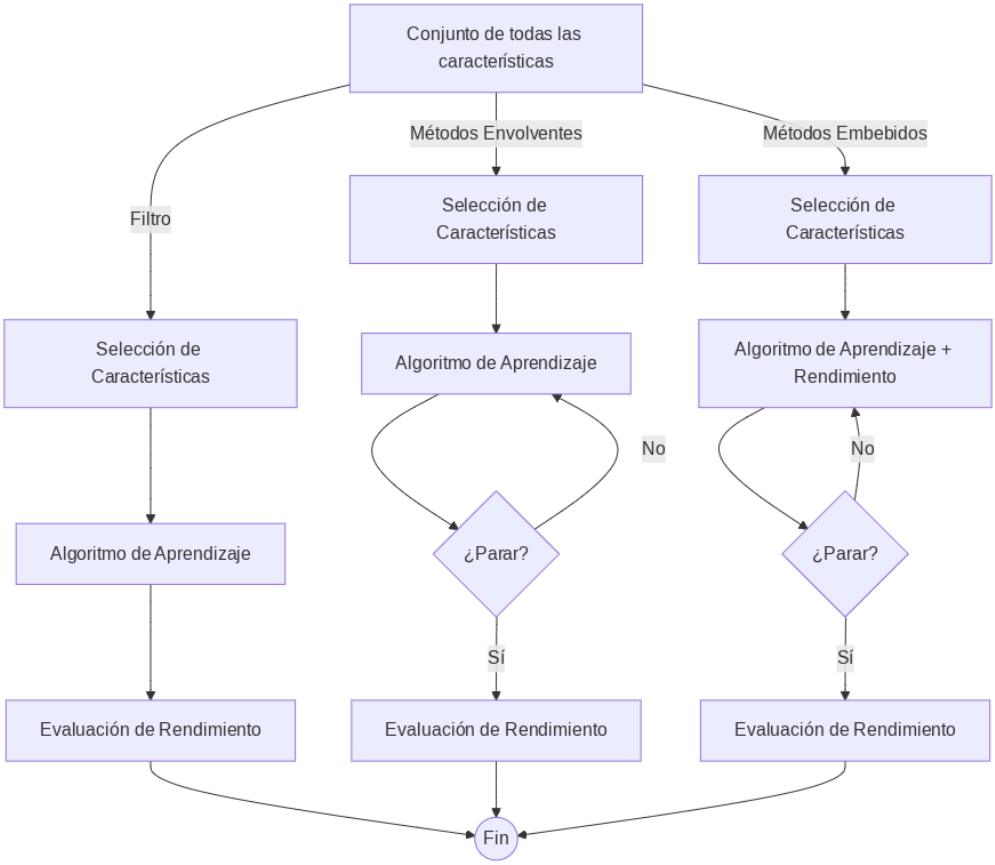

FIGURE 1.1: Diagrama de enfoques para la selección de características

Bolón-Canedo y ot. nos invitan a pensar que no existe un método que sea *superior a los otros* de manera general, sino que cada uno tiene sus ventajas y desventajas, y se ajustan distinto a diferentes tipos de contexto. De los tres enfoques, los métodos de filtro son los únicos que son independientes del algoritmo de aprendizaje, lo que les permite ser rápidos y eficientes computacionalmente. Sin embargo, los métodos de filtro no consideran la correlación entre características, lo que puede llevar a la selección de características irrelevantes o redundantes.

Los métodos de envolventes y embebidos, por su parte, consideran la correlación

entre características y etiquetas de clase, lo que les permite identificar patrones relevantes correctamente durante la fase de aprendizaje. Sin embargo, estos métodos son más complejos computacionalmente y requieren evaluación iterativa del subconjunto seleccionado de características.

En el caso de los métodos envolventes, a medida que el número de características aumenta, el costo computacional se incrementa exponencialmente, lo que limita su aplicación en problemas de alta dimensionalidad (Bolón-Canedo, Sánchez-Maroño, and Alonso-Betanzos 2015). En línea con esta dificultad, se han propuesto distintas formas de superar el problema, como el uso de heurísticas de búsqueda para evitar exploraciones exhaustivas. Entre ellas, se han propuesto a los métodos evolutivos como herramientas eficaces para la búsqueda de soluciones óptimas (Vignolo and Gerard 2017). El propósito de este enfoque es explorar el espacio de soluciones de manera eficiente, permitiendo la identificación de subconjuntos de características que sean óptimos en términos de rendimiento.

1.4. Selección, *algoritmos genéticos* y datos sintéticos

Hasta aquí hemos visto que problemas tales como la dimensionalidad, el desbalance de clases y el ruido plantean verdaderos desafíos para los procesos de aprendizaje automático, y de qué manera la selección de características puede ser una herramienta para mitigarlos. Asimismo, hemos descrito los distintos enfoques para la selección de características, resaltando el hecho de que los métodos envolventes utilizan *algoritmos genéticos* para realizar búsquedas eficientes en espacios de gran dimensionalidad. Aquí surgió el problema de la escasez de datos y su impacto negativo sobre la evaluación de los subconjuntos de características; una importante limitación para los *algoritmos genéticos*, y por ende, para los métodos envolventes de selección. En este punto final del capítulo 1, veremos de qué manera la generación sintética de datos puede resolver el problema de la escasez muestral, contribuir al funcionamiento de los *algoritmos genéticos*, y de esa forma mejorar los procesos de selección de características basados en métodos envolventes. Precisamente, este eje recorre el aporte central que realizaremos en la presente tesis.

Como dijimos, en el contexto de los métodos envolventes, los *algoritmos genéticos* representan una solución el problema de la evaluación iterativa de subconjuntos de características en datos de alta dimensionalidad. Tal evaluación debe replicarse por cada subconjunto a lo largo del espacio de búsqueda, lo que resulta en un costo computacional elevado cuando el conjunto de variables a evaluar es grande (R. Zhang et al. 2019).

Inspirados en los principios de la evolución natural, los *algoritmos genéticos* son capaces de realizar esa exploración eficientemente, iterando sobre múltiples generaciones de individuos. Exploraremos esto en detalle en el capítulo 4, sin embargo, cabe señalar desde ahora que dicha capacidad está determinada en gran medida por el conjunto de operadores evolutivos con los que trabajan los *algoritmos genéticos*: selección, cruce y mutación. La **selección**, paso crítico del proceso de búsqueda, otorga mayor probabilidad de reproducción a los individuos mejor evaluados. En este caso, los subconjuntos de características con mayor capacidad de discriminación son seleccionados en virtud de su desempeño (evaluado por una función objetivo representada comúnmente por un modelo de aprendizaje automático). El **cruce**, que combina partes de la solución de dos individuos para generar descendientes potenciamente mejores. Y la **mutación**,

que introduce variaciones aleatorias y evita la convergencia prematura hacia subóptimos locales. A través de iteraciones sucesivas, estos operadores permiten explorar de manera equilibrada tanto regiones prometedoras del espacio de búsqueda como soluciones novedosas, optimizando la búsqueda de subconjuntos de características incluso en escenarios con alta dimensionalidad y complejidad.

Dicho lo anterior, es preciso advertir que la eficacia de los *algoritmos genéticos* depende en gran medida de la disponibilidad de datos suficientes para evaluar el valor de los individuos a lo largo de las generaciones. En efecto, la falta de datos para estimar de forma confiable ese desempeño, agrega incertidumbre al momento de la **selección**, que es el paso crítico del proceso de búsqueda. Esta estimación es realizada por un modelo de aprendizaje automático con el que se evalúa la performance (**fitness**) de cada individuo (por ejemplo, un clasificador basado en árboles de decisión, máquinas de soporte vectorial, o un perceptrón multicapa). Por eso, con respecto a este componente en particular, el algoritmo genético se encuentra en la misma situación que cualquier otro método de optimización, en el sentido de que la escasez de datos afecta directamente la calidad de las predicciones utilizadas para guiar la búsqueda. Cuando los datos son insuficientes o no representan adecuadamente la variabilidad del problema, aumenta la incertidumbre en la fase de evaluación de los individuos, comprometiendo la confiabilidad del proceso de selección y la eficacia general del algoritmo genético.

Es en este escenario, ante tan importante restricción, es donde entra nuestra contribución. Entendemos que la generación sintética de datos puede ser una herramienta eficaz para resolver el problema de la escasez muestral, desbalance de clases y ruido, y por ende, mejorar la capacidad de los *algoritmos genéticos* para explorar y discriminar soluciones óptimas en espacios de gran dimensionalidad. A lo largo de la presente tesis, mostraremos cómo dicha estrategia puede ser implementada en el contexto de la selección de características basada en *algoritmos genéticos*.

Ahora bien, existen varias técnicas de aumentación de datos, desde las más tradicionales, como el sobremuestreo (SMOTE (Blagus and Lusa 2013)), hasta las más modernas, basadas en modelos generativos (por ejemplo, usando redes GANS (Fajardo et al. 2021)). Entre tales opciones los *autocodificadores variacionales* (AVs) (Kingma and Welling 2019) han adquirido popularidad gracias a su buen desempeño, incluso superando a otros métodos de aumentación. Estos modelos, con arquitectura encoder-decoder, han demostrado ser especialmente adecuados para escenarios donde se busca preservar la estructura estadística subyacente de los datos y, a la vez, generar muestras lo suficientemente diversas para captar la variabilidad del problema. Abordaremos esto en detalle en el capítulo 3, sin embargo, cabe adelantar aquí que estos modelos se basan en la estimación y reconstrucción de distribuciones latentes continuas, lo que les permite reconocer patrones relevantes y producir muestras sintéticas similares a los datos originales. Por esas propiedades deseables que presenta su funcionamiento, los AVs se han usado para aumentación de datos en el campo del tratamiento de imágenes (Fajardo et al. 2021; Ai et al. 2023; Khmaissia and Frigui 2023; Kwarciak and Wodzinski 2023), texto (Y. Zhang et al. 2019) , habla (Blaauw and Bonada 2016; Latif et al. 2020) y música (Roberts et al. 2019), y distintos formatos de datos: tabulares (Leelarathna et al. 2023), longitudinales (Ramchandran et al. 2022) y grafos (Liu et al. 2018).

La versatilidad y capacidad de los AVS, los convierte en una herramienta potencialmente transformadora para los procesos de selección de características basada en AGs,

abriendo la posibilidad de aplicaciones en escenarios muy diversos, y problemas que involucran distintas modalidades de datos.

Volviendo a los problemas señalados, la generación de muestras sintéticas puede realizar un importante aporte al tratamiento del desbalance de clases. En escenarios críticos con problemas multiclase y distribuciones asimétricas, la generación sintética de datos puede ofrecer la posibilidad de reequilibrar las proporciones de cada categoría. En efecto, creando suficientes muestras representativas de las clases minoritarias, la generación sintética de datos puede mejorar la discriminación entre categorías poco representadas. Con esta estrategia, se reduciría el sesgo introducido por la disparidad de muestras y, por ende, se mejoraría el proceso de selección de características.

Por último, es importante destacar que la generación sintética de datos mediante AVs también puede contribuir a mitigar el efecto del ruido en los modelos de aprendizaje. Durante el proceso de codificación-decodificación implementado por estos modelos, el proceso de reconstrucción identifica y proyecta las características esenciales de los datos, favoreciendo la reducción de información irrelevante. Este mecanismo, analizado en mayor detalle en el capítulo 3, abonaría la importancia de los AVs en la creación de muestras sintéticas no solo para reducir el impacto negativo del ruido sobre el proceso de entrenamiento, sino también para mejorar el desempeño de los métodos evolutivos en la búsqueda de características óptimas.

Todas estas razones respaldan la importancia de la generación sintética de datos para la selección de características con AGs. En la presente tesis se propone una estrategia integral que vincula la selección de características y la generación sintética de datos, enfocándose en solucionar problemas habituales en aprendizaje automático, tales como la alta dimensionalidad,la escasez de muestras, la complejidad y el ruido.

Siguiendo esta línea, la tesis se organiza de la siguiente manera:

- En el capítulo 2, se realiza un análisis preliminar de los datos con los que trabajaremos, se describen los modelos de aprendizaje automático evaluados para servir de función objetivo en el Algoritmo Genético que desarrollaremos, y se presentan las métricas de evaluación aplicadas a lo largo del estudio. Además, se discuten potenciales dificultades asociadas a la naturaleza de los datos.
- En el capítulo 3, se profundiza en la teoría de los *autocodificadores variacionales* y en la motivación para usarlos como técnica de aumentación de datos. Se detallan los experimentos de generación sintética, los resultados obtenidos y la forma en que estos modelos capturan la estructura subyacente de la distribución original, contribuyendo así otorgar calidad a los datos generados.
- En el capítulo 4, se presenta el Algoritmo Genético desarrollado, describiéndose la integración de los AVs como etapa de preprocesamiento para enriquecer la selección de características. Asimismo, se exponen los experimentos realizados tanto con datos originales como con datos aumentados, demostrando el impacto de la generación sintética en el rendimiento de la selección de características.
- Finalmente, en el capítulo 5, se exponen las conclusiones de la investigación, resaltando la relevancia de la generación sintética de datos para la selección de características en escenarios complejos, y se ofrecen perspectivas de trabajo futuro donde las técnicas propuestas podrían ampliarse o refinarse.

De este modo, se establece un marco metodológico completo que explora, integra y evalúa la sinergia entre *algoritmos genéticos*, *autocodificadores variacionales* y selección de características, sirviendo como aporte innovador para atender las dificultades

impuestas por la dimensionalidad, la escasez de datos, el desbalance de clases y el ruido en el ámbito del aprendizaje automático.

Capítulo 2

Modelos clásicos aplicados al espacio completo de características

En este capítulo revisaremos el desempeño de *algoritmos clásicos* en la solución de los problemas elegidos para nuestra investigación, tomando como campo de búsqueda el espacio completo de características de los datasets. El objetivo de esta exploración es doble: por un lado contar con métricas de base para comparar el desempeño de nuestras soluciones, y por otro, identificar los modelos más apropiados para emplear como función de aptitud en nuestro algoritmo genético. Al mismo tiempo, revisaremos las características de los datasets elegidos para nuestro estudio, procurando identificar aquellos rasgos que puedan influir en el desempeño de los modelos. Particularmente, nos centraremos en la alta dimensionalidad y la escasez muestral, el desbalance de clases y el ruido.

2.1. Datos elegidos en nuestro estudio

El conjunto de datos elegidos en este trabajo incluye cinco datasets: *Madelon*, *Gisette*, *Leukemia*, *GCM* y *All Leukemia*. El último de ellos, *All Leukemia*, es un dataset introducido en este trabajo iniciada la etapa de experimentación de integración entre *algoritmos genéticos y autocodificadores variacionales*. Por esa razón, no está incluido en la revisión de desempeño de los algoritmos clásicos que presentaremos en este capítulo. No obstante ello, se incluye en el detalle de los datasets para que el lector tenga una idea de la variedad de problemas que se tuvieron en cuenta para validar los resultados de nuestra propuesta.

Como veremos a continuación, cada dataset plantea desafíos distintos en términos de aprendizaje, y posee distintos niveles de complejidad en su composición. El dataset *Madelon* es un conjunto artificial de datos con 2000 observaciones y 500 características (2000x500), donde el objetivo es resolver un problema XOR multidimensional con 5 características relevantes y 15 características corresponden a combinaciones lineales de aquellas (i.e. 15 características redundantes). Las otras 480 características fueron generadas aleatoriamente (no tienen poder predictivo). Madelon es un problema de clasificación de dos clases con variables de entrada binarias dispersas. Las dos clases están equilibradas, y los datos se dividen en conjuntos de entrenamiento y prueba. Fue creado para el desafío de Selección de Características NIPS_2003, y está disponible en el Repositorio UCI. Los datos están divididos en un conjunto de entrenamiento y un conjunto de testeo.

Como es fácil de advertir, este es un problema donde la información relevante está presente junto a información redundante y otra sin valor predictivo. Es decir, existe un alto nivel de ruido en los datos, planteando importantes desafíos para los algoritmos de aprendizaje, y uno particularmente interesante para el problema de selección de características. Respecto de las dimensiones del problema, no se estaría en una situación crítica de alta dimensionalidad y escasez muestral, ya que el número de observaciones es mayor que el de características. Sin perjuicio de ello, aún queda por determinar si la cantidad de patrones disponibles es suficiente para que los algoritmos de aprendizaje puedan encontrar una solución en un contexto tan ruidoso.

El dataset *Gisette* es un dataset creado para trabajar el problema de reconocimiento de dígitos escritos a mano (Isabelle Guyon 2004). Este conjunto de datos forma parte de los cinco conjuntos utilizados en el desafío de selección de características NIPS 2003. Tiene 13500 observaciones y 5000 atributos (13500x5000). El desafío radica en diferenciar los dígitos '4' y '9', que suelen ser fácilmente confundibles entre sí. Los dígitos han sido normalizados en tamaño y centrados en una imagen fija de 28x28 píxeles. Además, se crearon nuevas características como combinación de las existentes para construir un espacio de mayor dimensión. También se añadieron características distractoras denominadas "sondas", que no tienen poder predictivo. El orden de las características y patrones fue aleatorizado. Los datos están divididos en un conjunto de entrenamiento y un conjunto de testeo.

En este caso, nos encontramos en un escenario similar al de *Madelon*, con un dataset ruidoso e información redundante. La particularidad de *Gisette* es que, pese a mantener una relación positiva entre obervaciones y características (las primeras son más que las segundas), posee un espacio de búsqueda sensiblemente más grande y, eventualmente, más complejo que el de *Madelon*. Por esa razón, esperamos que este dataset sea computacionalmente más exigente que el anterior.

El dataset *Leukemia* es un análisis de datos de expresión genética obtenidos de microarreglos de ADN, se estudia en Golub (1999) para la clasificación de tipos de cáncer. Se construyó un conjunto de datos con 72 observaciones y 7129 mediciones (72x7129) de las clases ALL (leucemia linfocítica aguda) y AML (leucemia mielogénica aguda). El problema es distinguir entre estas dos variantes de leucemia (ALL y AML). Los datos se dividen originalmente en dos subconjuntos: un conjunto de entrenamiento de 38 observaciones y un conjunto de testeo de 34 observaciones.

Con este dataset nos encontramos, precisamente, en el escenario de alta dimensionalidad y escasez muestral. El número de observaciones es menor que el de características, y las dimensiones del problema son significativamente más altas que en los casos anteriores. Además, el dataset está desbalanceado, con 27 observaciones de la clase ALL y 11 de la clase AML en la partición de entrenamiento, lo que plantea un desafío adicional para los algoritmos de aprendizaje.

El dataset *All Leukemia* es un estudio de pacientes pediátricos con leucemia linfoblástica aguda (LLA). Incluye 327 muestras con información de 12600 genes (327x12600). Está compuesto por 14 clases desequilibradas. Fue compilado por Yeoh et al., en link.

Este caso es, sin duda, uno de los más complejos de todos, y plantea un desafío computacional significativo. El número de observaciones es significativamente menor que el de características, y las dimensiones del problema son significativamente más altas que en los casos anteriores. Además, el dataset está desbalanceado, con multiples clases desequilibradas, planteando no solo el desafío de las diversas fronteras de decisión que se deben encontrar, sino también el problema de posibles ambigüedades entre

clases. Como vimos en el capítulo anterior, el desafío de procesar información genética es significativo considerando la alta correlación entre genes, y la gran cantidad de información redundante.

Finalmente, el dataset *GCM* fue compilado en Ramaswamy (2001) y contiene los perfiles de expresión de muestras de tumores que representan 14 clases comunes de cáncer humano. El dataset está compuesto por 190 muestras y 16063 atributos (biomarcadores), distribuidos en clases desequilibradas. Los datos están divididos en un conjunto de entrenamiento y un conjunto de testeo.

En GCM, como en el caso de *All Leukemia*, nos encontramos en un escenario de alta dimensionalidad y escasez muestral. El número de observaciones es significativamente menor que el de características, y las dimensiones del problema son significativamente más altas que en todos los casos anteriores. Además, el dataset está desbalanceado, eventualmente contiene abundante información ruidosa y redundante.

Entendemos que esta variedad de datasets cubre un amplio espectro de problemas de aprendizaje, y que los resultados obtenidos en cada uno de ellos nos permitirán evaluar el desempeño de nuestros algoritmos en contextos distintos.

2.2. El desempeño de algoritmos clásicos

Pasando a la evaluación de los modelos clásicos, hemos seleccionado una serie de modelos ampliamente usados en el campo del aprendizaje automático para tomar su desempeño como indicador. Entre ellos, encontramos: *modelos lineales*, *modelos basados en árboles*, *modelos de Naive Bayes*, *modelos de vecinos más cercanos*, *modelos de redes neuronales* y *modelos de Máquinas de Soporte Vectorial*. Cuando fue posible dada la naturaleza y características de los datasets, hemos evaluado todos los modelos. En caso contrario, hemos seleccionado aquellos modelos más apropiados para el contexto, dejando de lado los que no resultaban adecuados para el problema (como es el caso de GCM y Gisette, dada la dimensionalidad de ambos, y la naturaleza multiclase del primero). Finalmente, a fin de estandarizar la implementación de estos algoritmos, hemos empleado la librería `scikit-learn` que provee abstracciones convenientes para nuestro entorno de experimentación.

Los modelos lineales se basan en la premisa de que la variable objetivo puede expresarse como una función lineal de los predictores o características. Normalmente asumen que, para cada observación, la suma ponderada de las características (potencialmente con un término independiente o "bias") determina la respuesta. Entre los modelos lineales que incluimos en nuestro estudio se encuentran: el Análisis Discriminante Lineal (LDA), el Análisis Discriminante Cuadrático (QDA), la Regresión de Cresta (Ridge), y el Descenso de Gradiente Estocástico (SGD) (Hastie, Tibshirani, and Friedman 2009). El Análisis Discriminante Lineal (LDA) asume que cada clase proviene de una distribución normal multivariada con igual matriz de covarianzas y distinta media, y busca el hiperplano que maximiza la separabilidad entre clases proyectando los datos a un subespacio. El Análisis Discriminante Cuadrático (QDA) relaja el supuesto de matriz de covarianzas compartida, permitiendo que cada clase tenga su propia matriz y mejorando así la capacidad de modelar fronteras de decisión más complejas, aunque con un mayor riesgo de sobreajuste cuando se cuenta con poca muestra. La Regresión de Cresta (Ridge) introduce una penalización L2 sobre los coeficientes para controlar la varianza de la solución y mitigar la multicolinealidad, lo cual es especialmente útil si existe una gran correlación entre características. El

Descenso de Gradiente Estocástico (SGD), por su parte, consiste en un procedimiento incremental de optimización que actualiza los parámetros de un modelo lineal luego de cada observación (o mini-batch), resultando muy eficiente en problemas de alta dimensión o grandes volúmenes de datos.

En el caso de los modelos basados en árboles, todos comparten la idea de ir dividiendo recursivamente el espacio de características en regiones homogéneas. Este proceso se materializa en un árbol de decisión que, en cada nodo, escoge un umbral o criterio de partición para una característica. Los modelos incluidos en nuestro estudio son: Arbol de decisión clásico (DTC), AdaBoost, Bagging, Extra Trees Ensemble, Gradient Boosting, Random Forest, ETC, y Arboles Extremadamente Aleatorizados (ETC) (Hastie, Tibshirani, and Friedman 2009). El Árbol de Decisión Clásico (Decision Tree Classifier, DTC) utiliza criterios como la ganancia de información o la reducción de la impureza para decidir la partición óptima en cada nivel, siendo fácilmente interpretable aunque con tendencia al sobreajuste si no se regula su profundidad. Algunas variantes se basan en la combinación de múltiples árboles. Bagging, por ejemplo, entrena árboles independientes a partir de muestras "bootstrap" y agrega las predicciones para reducir la varianza del modelo. Random Forest amplía esta idea, incorporando además la selección aleatoria de características en cada división, con lo cual reduce la correlación entre árboles y mejora la capacidad generalizadora. Extra Trees Ensemble adopta una estrategia aún más aleatoria, ya que define umbrales de corte aleatorios, lo que tiende a una mayor diversidad entre árboles y puede favorecer la reducción de la varianza. AdaBoost, en contraposición, entrena secuencialmente modelos débiles (a menudo árboles de baja profundidad) poniendo más peso en las observaciones mal clasificadas en iteraciones previas, de modo que cada nuevo modelo aprenda de los errores acumulados. Gradient Boosting también combina modelos débiles, pero en su caso cada etapa del entrenamiento se orienta a predecir el error residual del ensamble previo, optimizando una función de costo de forma aditiva y generalmente logrando modelos muy potentes. Cuando se habla de ETC (Extremely Randomized Trees Classifier), se hace referencia a un enfoque similar a Random Forest, pero que enfatiza la aleatorización tanto en la selección de subconjuntos de características como en los umbrales de partición, mejorando la diversidad de los árboles y mitigando así la varianza global.

Los modelos de Naive Bayes se basan en el teorema de Bayes para predecir la probabilidad de pertenencia a cada clase, asumiendo independencia condicional de las características (Hastie, Tibshirani, and Friedman 2009). Aun cuando esta suposición rara vez se cumple por completo en problemas reales, la simplicidad computacional y la eficacia empírica suelen convertirlos en una elección sólida, especialmente en problemas de alta dimensión. En su versión Bernoulli (BNB), se asume que las variables predictoras son binarias, lo que se ajusta bien a datos dispersos donde cada característica indica la presencia o ausencia de cierta propiedad. En la versión Gaussiana (GNB), se asume que cada característica sigue una distribución normal, estimando media y varianza por clase para luego combinar esas estimaciones en la regla de decisión bayesiana.

En cuanto a los métodos basados en vecinos más cercanos, el clasificador K-Vecinos Más Cercanos (KNN) ejemplifica la estrategia de aprendizaje por vecindad, ya que no construye un modelo explícito durante la etapa de entrenamiento. En su lugar, para clasificar una nueva observación, identifica los K vecinos más cercanos en el espacio de características y asigna la clase mayoritaria de ese entorno local (Hastie, Tibshirani, and Friedman 2009). Este enfoque es intuitivo y puede capturar relaciones complejas

en los datos, aunque su desempeño se degrada en alta dimensión y requiere un costo computacional alto en predicción, pues debe calcular distancias a todos los puntos de entrenamiento.

Entre los modelos de redes neuronales, el Perceptrón Multicapa (MLP) es una arquitectura de red con múltiples capas densamente conectadas y funciones de activación no lineales (Hastie, Tibshirani, and Friedman 2009). Su capacidad de aproximar funciones complejas lo convierte en un modelo flexible, pero también más exigente en términos de datos y calibración de hiperparámetros. Aunque en su versión más simple puede considerarse un "clásico", el MLP con técnicas de regularización y optimización robustas forma parte fundamental de las estrategias de aprendizaje profundo.

Por último, las Máquinas de Soporte Vectorial (SVM) parten del principio de encontrar un hiperplano (en el caso lineal) u "frontera" (en el caso con núcleos no lineales) que maximice el margen de separación entre clases (Hastie, Tibshirani, and Friedman 2009). En contextos de alta dimensión y con un número moderado de muestras, las SVM suelen mostrar un desempeño notable por su capacidad para controlar el sobreajuste mediante el parámetro de regularización y el uso de núcleos apropiados. Su versión lineal (LSVC) se centra en resolver una optimización con un límite que separa las clases en un espacio original de altas dimensiones sin necesidad de mapeos adicionales, resultando eficiente en muchos casos de datos dispersos. La variante NuSVC introduce un parámetro *nu* que controla tanto el número de vectores de soporte como la proporción máxima de errores permitidos, proporcionando una forma alternativa de regularización y definición de la frontera de decisión. Estas particularidades hacen que las SVM sean especialmente populares en problemas donde la dimensionalidad de las características es grande con respecto al número de muestras disponibles.

2.2.1. Configuración de los Modelos

Para evaluar el desempeño de los modelos clásicos y determinar la configuración más adecuada de sus hiperparámetros, hemos implementado un proceso de búsqueda sistemática en grilla (*Grid Search*). Este procedimiento consiste en seleccionar un conjunto de valores relevantes para cada hiperparámetro e iterar sobre todas las combinaciones posibles, entrenando y validando el modelo en cada caso.

En particular, establecimos un rango de parámetros numéricos que incluye de 3 a 20 valores, dependiendo de la sensibilidad del modelo a cada hiperparámetro y de los límites sugeridos en la literatura. Por ejemplo, en modelos lineales como Ridge o SGD, la regularización (*penalty*) y la tasa de aprendizaje (*learning rate*) se probaron en al menos tres niveles para capturar comportamientos distintos. En cambio, en algoritmos basados en árboles (como Random Forest o Gradient Boosting), ampliamos el rango hasta 20 valores en parámetros críticos (número de árboles, profundidad máxima, etc.) para reflejar la diversidad de configuraciones posibles.

Para los parámetros no numéricos (como funciones de activación en MLP, criterios de partición en árboles, tipo de kernel en SVM, entre otros) se utilizaron configuraciones estándar y reconocidas por la comunidad, con el fin de reducir la complejidad combinatoria. Aun así, para cada modelo se revisó la documentación de `scikit-learn` y la literatura relacionada, asegurando una cobertura apropiada de las variantes más importantes.

La métrica de evaluación que hemos seleccionado es **AUC** para los dataset de clasificación binaria (Madelon, Gisette y Leukemia), y **F1 score** para los dataset de clasificación multiclase (GCM).

La métrica AUC mide el área bajo la curva ROC, donde la curva ROC traza la Tasa de Verdaderos Positivos (TPR o Sensibilidad) frente a la Tasa de Falsos Positivos (FPR) a distintos umbrales de decisión. Matemáticamente, la AUC puede interpretarse (bajo ciertas condiciones) como la probabilidad de que el clasificador asigne una puntuación más alta a una muestra positiva que a una negativa. Un valor de AUC igual a 1 indica un modelo perfecto, mientras que un valor cercano a 0.5 sugiere un desempeño cercano al de un clasificador aleatorio.

En *clasificación binaria*, la AUC ofrece un panorama amplio del desempeño del modelo al no depender de un umbral específico y tiene en cuenta la relación entre verdaderos positivos y falsos positivos. Esto es especialmente relevante en datasets como Madelon, Gisette o Leukemia, donde el desbalance y la alta dimensionalidad pueden sesgar otras métricas.

La métrica F1 se define como la media armónica entre la Precisión (exactitud) y el Recall (sensibilidad):
F1 = 2 × (Precisión × Recall) / (Precisión + Recall). La Precisión indica qué proporción de las predicciones positivas son realmente positivas, mientras que el Recall mide la proporción de positivos correctamente identificados respecto de todos los positivos reales. La F1 combina ambas, penalizando los modelos que desequilibran excesivamente la Precisión y el Recall. En problemas multiclase (por ejemplo, en GCM con 14 clases), la F1 puede calcularse por clase y luego promediarse (macro-F1) para evaluar la capacidad de cada modelo de reconocer de forma equilibrada todas las clases, incluso cuando existe desbalance.

En *clasificación multiclase*, la F1 proporciona una forma de resumir la Precisión y el Recall cuando hay varias clases involucradas y potencialmente desbalanceadas. La F1 balancea correctamente los falsos positivos y los falsos negativos para cada clase antes de promediar, lo que resulta más informativo para problemas con muchas clases desiguales (como GCM).

Las particiones originales de los datasets fueron concatenadas en un solo conjunto de datos, y luego se dividió en conjuntos de entrenamiento y testeo en proporción 80/20.

2.3. Resultados Obtenidos

En la siguiente tabla resumimos los resultados en el dataset de testeo para cada modelo y dataset, con las métricas AUC y F1 score.

Modelo	Leukemia Test	Madelon Test	Gisette Test	GCM Test
LDA	0.85	0.60	0.96	-
QDA	0.50	0.66	0.70	-
Ridge	0.99	0.60	0.97	-
SGD	0.98	0.64	0.99	0.71
AdaBoost	0.91	0.84	0.99	-
Bagging	**1.00**	**0.91**	-	-
DTC	0.72	0.64	0.92	0.53
ETC	0.54	0.57	0.94	0.48
Ext.Trees.Ens.	**1.00**	0.71	0.99	0.57
Gradient Boost.	0.99	0.82	**1.00**	0.58

Modelo	Leukemia Test	Madelon Test	Gisette Test	GCM Test
Random Forest	**1.00**	0.78	0.99	0.62
LSVC	0.99	0.62	0.99	0.62
NuSVC	**1.00**	0.61	0.99	0.58
SVC	**1.00**	0.61	0.99	0.58
BNB	0.89	0.63	0.94	-
GNB	0.91	0.65	0.85	-
KNN	0.86	0.65	0.99	-
MLP	0.96	0.58	0.99	**0.68**

El gráfico completo de resultados en las particiones de entrenamiento y testeo puede verse en la siguiente figura:

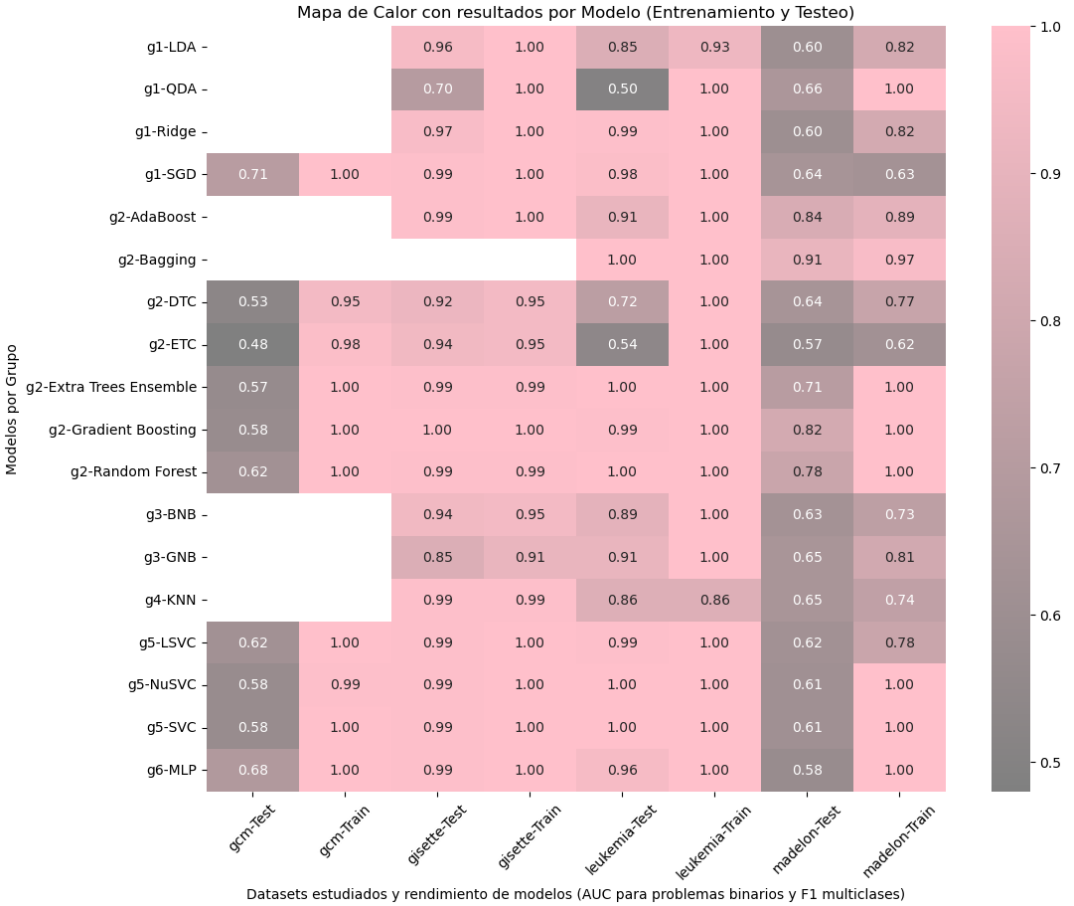

FIGURE 2.1: algoritmosclasicos

Las métricas presentadas evidencian que los resultados de los modelos clásicos varían ampliamente según las características de cada dataset. En particular, los conjuntos de datos **Leukemia** y **Gisette** muestran, en términos generales, un desempeño sólido para la mayoría de los algoritmos analizados. La **AUC** alcanzada por varios modelos en *Leukemia* (en algunos casos 1.00) ilustra la capacidad de separar correctamente

los casos, a pesar de tratarse de un problema con alta dimensionalidad y desbalance. Asimismo, *Gisette* se beneficia del hecho de contar con más observaciones que *features*, lo que facilita la labor de los clasificadores (algunos, como Gradient Boosting, alcanzan AUC = 1.00).

En contraste, **Madelon** y, muy especialmente, **GCM**, exhiben una dificultad sustancialmente mayor para casi todas las familias de modelos clásicos. En *Madelon*, la presencia de ruido y características irrelevantes afecta la capacidad de generalización, quedando reflejado en valores de AUC relativamente bajos (en general por debajo de 0.70). Esta situación concuerda con la naturaleza artificial de *Madelon*, donde únicamente un subconjunto pequeño de atributos tiene poder predictivo y los modelos se enfrentan a espacios de búsqueda con muchos distractores. Este resultado subraya la relevancia de algoritmos que incorporen estrategias que mitiguen el impacto del ruido en el aprendizaje o mecanismos de selección de características que permitan descartar las irrelevantes.

El caso de **GCM** es aún más crítico. Se trata de un problema multiclase severamente desbalanceado y con una dimensionalidad desproporcionadamente alta en comparación con el número de muestras disponibles. Dichas condiciones propician que todos los modelos clásicos evidencien dificultades para capturar las fronteras de decisión y generalizar correctamente. El hecho de que el **MLP** sea, en este caso, el método de mayor desempeño (si bien con un F1 todavía moderado) puede atribuirse a la flexibilidad de las redes neuronales y su capacidad de aproximar funciones complejas, a pesar del reducido número de muestras de entrenamiento. No obstante, la mayoría de clasificadores enfrenta limitaciones para generalizar en este escenario de desequilibrio tan marcado, reforzando la hipótesis central de la tesis sobre la importancia de enfoques generativos para rebalancear la asimetría entre clases y mejorar la clasificación.

Los resultados que el MLP obtiene por clase se muestran en la siguiente tabla:

Class	**Precision**	**Recall**	**F1-Score**	**Support**
Bladder	0.75	1.00	0.857	3
Breast	0.00	0.00	0.000	3
CNS	0.80	1.00	0.889	4
Colorectal	0.75	1.00	0.857	3
Leukemia	1.00	0.83	0.909	6
Lung	1.00	0.33	0.500	3
Lymphoma	0.83	0.83	0.833	6
Melanoma	1.00	0.50	0.667	2
Mesothelioma	1.00	1.00	1.000	3
Ovary	0.50	0.67	0.571	3
Pancreas	0.50	0.67	0.571	3
Prostate	0.00	0.00	0.000	2
Renal	1.00	0.33	0.500	3
Uterus___Adeno	0.40	1.00	0.571	2
macro avg	0.68	0.65	0.62	46
weighted avg	0.73	0.70	0.68	46
accuracy			0.70	46

Obsérvese que ciertas clases con muy pocas muestras (por ejemplo, *Breast* o *Prostate*) presentan métricas nulas, mientras que otras con más muestras, como *Leukemia* o

Lymphoma, muestran resultados más consistentes. La presencia de clases con tan baja representatividad hace que el modelo no cuente con suficiente evidencia estadística para aprender patrones característicos, conduciendo a predicciones erróneas (o directamente inexistentes) para dichas clases. Como anticipamos en el capítulo anterior, en un escenario multiclase desbalanceado, puede ser común que las clases minoritarias obtengan poco o ningún peso en la función de pérdida, lo que lleva al modelo a ignorarlas o predecir muy pocas (o ninguna) instancias de dichas clases.

En consecuencia, en problemas donde la alta dimensionalidad y bajo número de muestras se combina con múltiples clases desequilibradas, aun los modelos con capacidad de generalización elevada (por ejemplo, redes neuronales) se ven limitados. La disparidad de resultados por clase que hemos reportado (desde métricas perfectas hasta valores nulos) ilustra la necesidad de una estrategia que aborde el desbalance de manera explícita, ya sea mediante incremento sintético de datos, la implementación de funciones de pérdida adaptativas (que penalicen más los errores en las clases minoritarias), o ambos.

En síntesis, la combinación de alta dimensionalidad, escasez muestral y desbalance de clases constituye un serio obstáculo al entrenamiento estable y robusto de modelos clásicos, tal como se constata en *GCM*. Por otro lado, incluso en problemas más sencillos como *Madelon*, el ruido y la alta proporción de atributos irrelevantes pueden perjudicar significativamente la performance de estos algoritmos. Estos hallazgos abonan la necesidad e importancia de la estrategia que exploraremos en este trabajo, que se centra en la generación de datos sintéticos para incrementar la muestra de entrenamiento y mejorar la capacidad de generalización de los modelos.

Capítulo 3

Autocodificadores Variacionales y datos sintéticos

En este capítulo presentamos la arquitectura del Autocodificador Variacional (AV) que emplearemos para la generación de datos sintéticos: exponemos brevemente sus fundamentos teóricos, los pasos que hemos seguidos en su implementación y las variaciones introducidas para su apropiada aplicación a los problemas abordados. También presentaremos las métricas que empleamos para evaluar los modelos y configuraciones más relevantes para la generación de datos sintéticos.Finalmente, terminaremos el capítulo comentando los hallazgos más destacados de todo este desarrollo.

3.1. Modelos generativos

Los modelos generativos (MG) son un amplio conjunto de algoritmos de aprendizaje automático que buscan modelar la distribución de probabilidad de datos observados $p_\theta(x)$. A diferencia de los modelos discriminantes (MD), cuyo objetivo es aprender un predictor a partir de los datos, en los modelos generativos el objetivo es *resolver un problema más general vinculado con el aprendizaje de la distribución de probabilidad conjunta de todas las variables.* Así, siguiendo a Kingma, podemos decir que *un modelo generativo simula la forma en que los datos son generados en el mundo real* (Kingma and Welling 2019). Dada estas propiedades, estos modelos permiten crear nuevos datos que se asemejan a los originales, y se aplican en tareas de generación de datos sintéticos, imputación de datos faltantes, reducción de dimensionalidad y selección de características, entre otros.

Los modelos generativos pueden tener como *inputs* diferentes tipos de dato, como imágenes, texto, audio, entre otros. Por ejemplo, las imágenes son un tipo de dato para los cuales los MG han demostrado gran efectividad. En este caso, cada dato de entrada x es una imagen que puede estar representada por un vector miles de elementos que corresponden a los valores de píxeles. El objetivo de un modelo generativo es aprender las dependencias (Doersch 2021) entre los píxeles (e.g. pixeles vecinos tienden a tener valores similares) y poder generar nuevas imágenes que se asemejen a las imágenes originales.

Podemos formalizar esta idea asumiendo que tenemos ejemplos de datos x, distribuidos según una distribución de probabilidad conjunta no conocida que queremos modelo $p_\theta(x)$ para que sea capaz de generar datos similares a los originales.

3.2. Autocodificadores

Los autocodificadores son un tipo de MG especializado en la representación de un espacio de características dado en un espacio de menor dimensión (de la Torre 2023). El objetivo de esta transformación es obtener una representación de baja dimensionalidad y la mayor fidelidad posible del espacio original. Para ello el modelo aprende a preservar la mayor cantidad de información relevante en un vector denso de menos dimensiones que las originales, y descarta -al mismo tiempo- lo irrelevante. Luego, a partir de esa información codificada, se busca reconstruir los datos observados según el espacio original.

Los autocodificadores se componen de dos partes: un *codificador* y un *decodificador*. El *codificador* es una función no lineal que opera sobre una observación o patrón x_i y la transforma en un vector de menor dimensión z, mientras que el *decodificador* opera a partir del vector z y lo transforma en una observación x'_i (o patrón reconstruido), buscando que se asemeja a la observación original. Este vector de menor dimensión z es conocido como *espacio latente*.

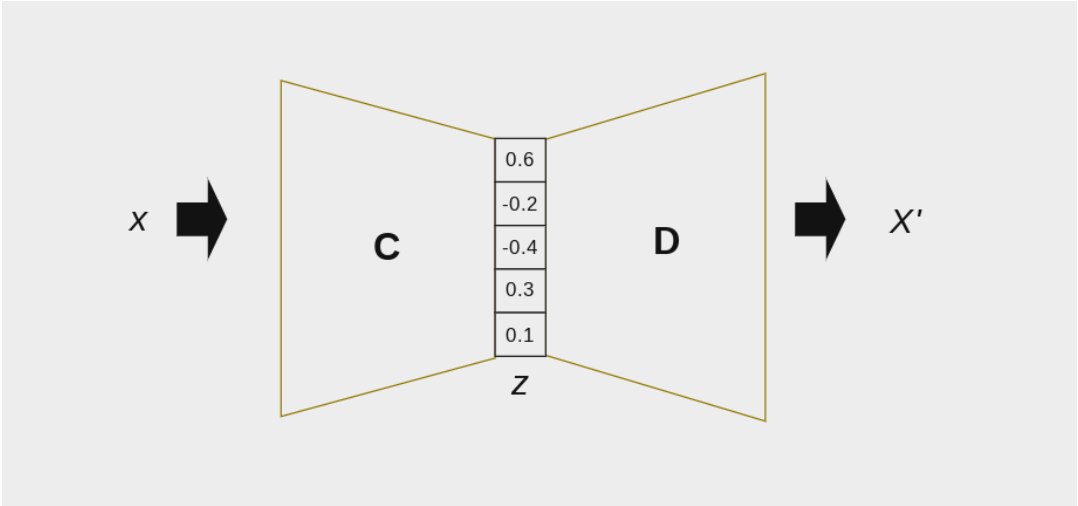

FIGURE 3.1: Ejemplo esquemático de un autocodificador

El esquema anterior muestra un autocodificador que transforma un patrón x en un vector denso z, con dimensiones menores a las originales. Este vector es usado luego por el decodificador para reconstruir el patrón original, dando lugar a un patrón reconstruido x'.

En el proceso de aprendizaje de un autocodificador, la red modela la distribución de probabilidad de los datos de entrada x y aprende a mapearlos a un espacio latente z. Para ello, se busca minimizar la diferencia entre la observación original x_i y la reconstrucción x'_i, diferencia que se denomina *error de reconstrucción*. Esta optimización se realiza a través de una *función de pérdida* que se define como la diferencia entre x_i y x'_i, que permite la optimización simultánea del codificador y decodificador.

Formalmente, podemos establecer estas definiciones (de la Torre 2023):

- Sea x el espacio de características de los datos de entrada y z el espacio latente, ambos espacios son euclidianos, $x = \mathbb{R}^m$ y $z = \mathbb{R}^n$, donde $m > n$.
- Sea las siguientes funciones paramétricas $C_\theta : x \to z$ y $D_\phi : z \to x'$ que representan el codificador y decodificador respectivamente.

- Entonces para cada observación $x_i \in x$, el autocodificador busca minimizar la función de pérdida $L(x_i, D_\phi(E_\theta(x_i)))$. Ambas funciones E_θ y D_ϕ son redes neuronales profundas que se entrenan simultáneamente.

Para optimizar un autocodificador se requiere una función que permita medir la diferencia entre la observación original y la reconstrucción. Esta diferencia usualmente se basa en la *distancia euclidia* entre x_i y x_i', es decir, $||x_i - x_i'||^2$. La función de pérdida se define como la suma de todas las distancias a lo largo del conjunto de datos de entrenamiento. Tenemos entonces que:

$$L(\theta, \phi) = argmin_{\theta,\phi} \sum_{i=1}^{N} ||x_i - D_\phi(C_\theta(x_i))||^2$$

Donde $L(\theta, \phi)$ representa la función de pérdida que queremos minimizar: θ son los parámetros del codificador C y ϕ son los parámetros del decodificados D.

3.3. Autocodificadores y el problema de la generación de datos

En el proceso de aprendizaje antes descripto, la optimización no está sujeta a otra restricción mas que minimizar la diferencia entre la observación original y la reconstrucción, dando lugar a espacios latentes generalmente discontinuos. Esto sucede porque la red puede aprender a representar los datos de entrada de manera eficiente sin necesidad de aprender una representación continua. En la arquitectura del autocodificador no hay determinantes para que dos puntos cercanos en el espacio de características se mapeen a puntos cercanos en el espacio latente.

Esta discontinuidad en el espacio latente hace posible que ciertas regiones de este espacio no tengan relación significativa con el espacio de características. Durante el entrenamiento el modelo simplemente no ha tenido que reconstruir datos cuyas distribuciones coincidan con estas regiones. Esto es un problema en la generación de datos, ya que la red podrá generar representaciones alejadas de los datos originales. Regularmente lo que se busca en los MG, no es simplemente una generación de datos completemante igual o totalmente distintos a los orginales, sino cierta situación intermedia donde los nuevos datos introducen variaciones en características específicas.

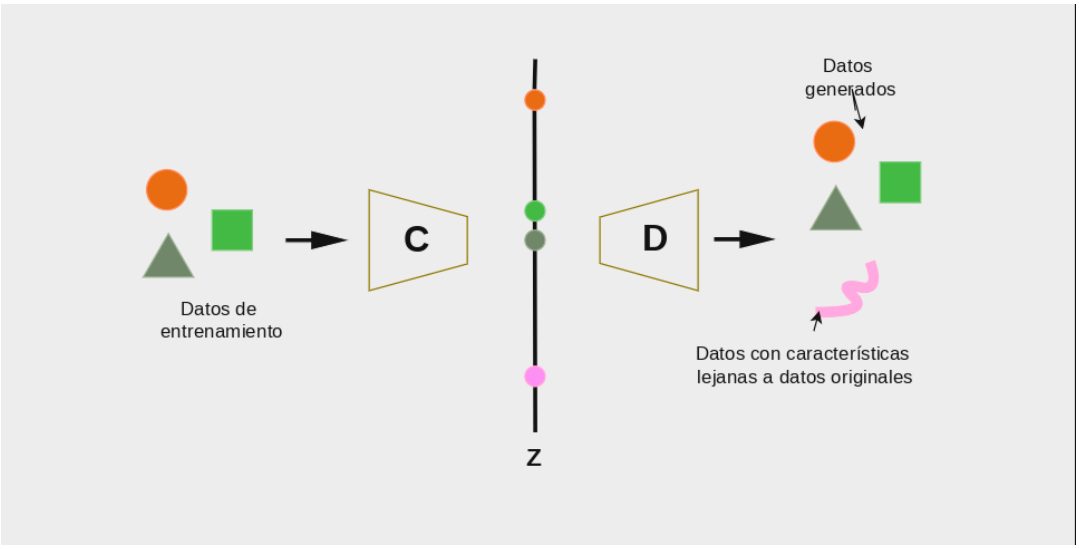

FIGURE 3.2: Discontinuidad del espacio latente

3.4. Autocodificadores Variacionales

Los Autocodificadores Variacionales (AVs) buscan resolver los problemas de discontinuidad y falta de regularidad en el espacio latente de los Autocodificadores. Comparten con éstos la arquitectura *codificador-decodificador*, pero introducen importantes modificaciones en su diseño para crear un espacio latente continuo.

Estos modelos, a diferencia de los autocodificadores que realizan transformaciones determinísticas de los datos de entrada (codificándolos como vectores n-dimensionales), buscan modelar la distribución de probabilidad de dichos datos aproximando la distribución *a posteriori* de las variables latentes $p_\theta(z|x)$. Para ello, la codificación se produce mediante la generación de dos vectores (μ y σ) que conforman el espacio latente, a partir del cual se toman las muestras para la generación.

La red codificadora, también llamada *red de reconocimiento*, mapea los datos de entrada x a los vectores μ de medias y σ de desvíos estándar, que parametrizan una distribución de probabilidad en el espacio latente. Generalmente, esta distribución es una distribución simple, como la distribución normal multivariada. La red decodificadora, también llamada *red generativa*, toma muestras de esta distribución para generar un vector, y lo transforma según la distribución de probabilidad preexistente del espacio de características. De esta manera, se generan nuevas instancias que respetan la distribución de probabilidad de los datos originales. Estas transformaciones implican que, incluso para el mismo dato observado (donde los parámetros de z son iguales), el dato de salida podrá ser diferente debido al proceso estocástico de reconstrucción.

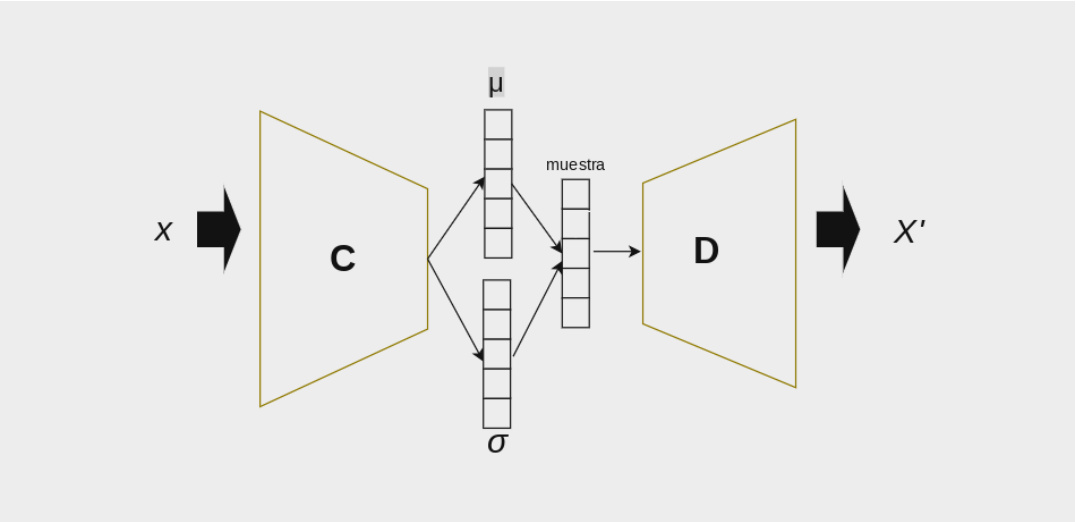

FIGURE 3.3: Autocoficadores Variacionales

Una forma de entender esta arquitectura sería relacionar los vectores que componen z como 'referecias', donde el vector de medias controla el *centro* en torno al cual se distribuirán los valores codificados de los datos de entrada, mientras que el vector de los desvíos traza el *área* que pueden asumir dichos valores en torno al *centro*.

Para indagar en estas intuiciones, veamos la solución que proponen los AV detenidamente, utilizando un enfoque formal. Así, dado un conjunto de datos de entrada $x = \{x_1, x_2, ..., x_N\}$, donde $x_i \in \mathbb{R}^m$, se asume que cada muestra es generada por un mismo proceso o sistema subyacente cuya distribución de probabilidad se desconoce. El modelo buscado procura aprender $p_\theta(x)$, donde θ son los parámetros de la función.

Por las ventajas que ofrece el logaritmo[1] para el cálculo de la misma tendremos la siguiente expresión: $\log p_\theta(x) = \sum_{x_i \in x} \log p_\theta(x)$[2].

La forma más común de calcular el parámetro θ es a través del estimador de *máxima verosimilitud*, cuya función de optimización es: $\theta^* = \arg\max_\theta \log p_\theta(x)$, es decir, buscamos los parámetros θ que maximizan la log-verosimilitud asignada a los datos por el modelo.

En el contexto de los AVs, el objetivo es modelar la distribución de probabilidad de los datos observados x a través de una distribución de probabilidad conjunta de variables observadas y latentes: $p_\theta(x, z)$. Aplicando la regla de la cadena de probabilidad podemos factorizar la distribución conjunta de la siguiente manera: $p_\theta(x, z) = p_\theta(x|z)p_\theta(z)$. Aquí $p_\theta(x|z)$ es la probabilidad condicional de los datos observados dados los latentes, y $p_\theta(z)$ es la probabilidad *a priori*[3] de los latentes.

Para determinar la distribución marginal respecto de los datos observados, es preciso integrar sobre todos los elementos de z, dando como resultado la siguiente función: $> p_\theta(x) = \int p_\theta(x, z)dz$ [4].

Esta distribución marginal puede ser extremadamente compleja, y contener un número indeterminable de dependencias (Kingma and Welling 2019), volviendo el calculo de la verosimilitud de los datos observados intratable. Esta intratabilidad de $p_\theta(x)$ está determinada por la intratabilidad de la distribución *a posteriori* $p_\theta(z|x)$, cuya dimensionalidad y multi-modalidad pueden hacer difícil cualquier solución analítica o numérica eficiente. Dicho obstáculo impide la diferenciación y por lo tanto la optimización de los parámetros del modelo.

Para abordar este problema, se acude a la inferencia variacional que introduce una aproximación $q_\phi(z|x)$ a la verdadera distribución *a posteriori* $p_\theta(z|x)$. Generalmente se emplea la distribución normal multivariada para aproximar la distribución *a posteriori*, con media y varianza parametrizadas por la red neuronal[5]. Sin embargo, la elección de la distribución no necesariamente tiene que pasar por una distribución normal, el único requerimiento es que sea una distribución que permita la diferenciación y el cálculo de la divergencia entre ambas distribuciones (por ejemplo si X es binaria la distribución $p_\theta(x|z)$ puede ser una distribución Bernoulli).

Así, en lugar de maximizar directamente el logaritmo de la verosimilitud (*log-verosimilitud*), se maximiza una cota inferior conocida como *límite inferior de evidencia* (*ELBO* por sus siglas en ingles). La derivación procede de la siguiente manera:

1. log-verosimilitud marginal (intratable):

[1] El logaritmo convierte la probabilidad conjunta (que se calcula como el producto de las probabilidades condicionales) en una suma de logaritmos, facilitando el cálculo y evitando problemas de precisión numérica: $\log(ab) = \log(a) + \log(b)$.

[2] Esta función se lee como la log-verosimilitud de los datos observados x bajo el modelo $p_\theta(x)$ y es igual a la suma de la log-verosimilitud de cada dato de entrada x_i.

[3] La expresión *a priori* alude a que no está condicionada por ningun dato observado.

[4] Aquí dz es el diferencial de z, por lo que la expresión indica la integración sobre todas las posibles configuraciones de la variable latente.

[5] En AVs, se suele asumir que z sigue una distribución normal multivariada: $p_\theta(z) = \mathcal{N}(z; 0, I)$, con media cero y matriz de covarianza identidad. La matriz de covarianza identidad es una matriz diagonal con unos en la diagonal y ceros en los demás lugares, y su empleo simplifica la implementación del modelo, permite que las variables latentes sean independientes (covarianza = 0) y varianza unitaria, evitando así cualquier complejidad vinculada a las dependencias entre dimensiones de z.

$\log p_\theta(x) = \log\left(\int p_\theta(x,z)\,dz\right)$,

2. aplicando inferencia variacional:

$\log p_\theta(x) = \log\left(\int q_\phi(z|x)\frac{p_\theta(x,z)}{q_\phi(z|x)}\,dz\right)$,

3. aplicando la desigualdad de Jensen[6]:

$\log p_\theta(x) \geq \mathbb{E}_{q_\phi(z|x)}\left[\log\left(\frac{p_\theta(x,z)}{q_\phi(z|x)}\right)\right]$,

4. descomponiendo la fracción dentro del logaritmo:

$\log p_\theta(x) \geq \mathbb{E}_{q_\phi(z|x)}\left[\log p_\theta(x|z) + \log p_\theta(z) - \log q_\phi(z|x)\right]$,

5. el resultando es el límite inferior de evidencia:

$\log p_\theta(x) \geq \mathbb{E}_{z\sim q_\phi(z|x)}[\log p_\theta(x|z)] - D_{\mathrm{KL}}(q_\phi(z|x)\|p_\theta(z))$,

donde:

- $\mathbb{E}_{z\sim q_\phi(z|x)}[\log p_\theta(x|z)]$ es el valor esperado (*esperanza*[7]) de la log-verosimilitud bajo la aproximación variacional, y determina la precision de la reconstrucción de los datos de entrada (un valor alto de esta esperanza indica que el modelo es capaz de reconstruir los datos de entrada con alta precisión a partir de los parámetros generados por $q_\phi(z|x)$).
- $D_{\mathrm{KL}}(q_\phi(z|x)\|p_\theta(z))$ es la divergencia de Kullback-Leibler entre la distribución $q_\phi(z|x)$ y la distribución *a priori* de las variables latentes $p_\theta(z)$, y determina la regularización del espacio latente.

Maximizando esta cota inferior (*ELBO*), se optimizan simultáneamente los parámetros θ del modelo y los parámetros ϕ de la distribución empleada en la aproximación, permitiendo una inferencia eficiente y escalable en modelos con z de alta dimensionalidad [8].

El objetivo de aprendizaje del AV se da entonces por:

$$\mathcal{L}_{\theta,\phi}(x) = \max(\phi,\theta)\left(E_{z\sim q_\phi(z|x)}[\log p_\theta(x|z)] - D_{\mathrm{KL}}(q_\phi(z|x)\|p_\theta(z))\right).$$

[6]Nótese que ese límite es siempre menor o igual y esto se deriva de una de las propiedades de las funciones convexas. Esta propiedad, denominada *desigualdad de Jensen*, establece que el valor esperado de una función convexa es siempre mayor o igual a la función del valor esperado. Es decir, $\mathbb{E}[f(x)] \geq f(\mathbb{E}[x])$. En el caso de funciones cóncavas, la desigualdad se invierte: $\mathbb{E}[f(x)] \leq f(\mathbb{E}[x])$. En este caso, la función logaritmo es cóncava, por lo que la desigualdad se expresa como: $\log(\mathbb{E}[x]) \geq \mathbb{E}[\log(x)]$.

[7]La esperanza es un promedio ponderado de todos los posibles valores que puede tomar una variable aleatoria, donde los pesos son las probabilidades de esos valores. En un AV, donde consideramos una distribución aproximada $q_\phi(z|x)$ para el espacio latente, la expresión citada es la esperanza de la log-verosimilitud bajo esta distribución.Aunque teóricamente esto implica un promedio sobre todas las posibles muestras z de la distribución $q_\phi(z|x)$, en la práctica, esta esperanza se estima utilizando una única muestra durante el entrenamiento por razones de eficiencia computacional. Esta única muestra permite calcular directamente $\log p_\theta(x_i|z_i)$, proporcionando una aproximación a la esperanza teórica y determinando la precisión de la reconstrucción de los datos de entrada.

[8]En la teoría, cuando derivamos el objetivo de un AV, estamos maximizando la cota inferior variacional (ELBO), para que la aproximación sea lo más cercana posible a la verdadera distribución de los datos. LLevado el problema a una implementación práctica generalmente se emplean optimizadores (SGD, Adam, etc.) que minimizan una función de pérdida. Para convertir el problema de maximización del ELBO en un problema de minimización, simplemente negamos el ELBO, resultando que los términos de la ecuación se reescriben como suma de cantidades positivas. El error o pérdida de reconstrucción se mide, según los casos, mediante MSE o entropía cruzada.

Como puede apreciearse en la ecuación anterior la función de pérdida del AV se compone de dos términos: el primero es la esperanza de la log-verosimilitud bajo la aproximación variacional y el segundo es la divergencia de Kullback-Leibler relacionada a la reconstrucción de los datos y la regularización del espacio latente. Existe entre ambos términos una relación de compromiso que permite al AV aprender una representación adecuada de los datos de entrada y, al mismo tiempo, un espacio latente continuo y regularizado.

En términos prácticos, la divergencia de Kullback-Leibler puede interpretarse como un factor que equilibra creatividad y precisión en el modelo. Por un lado, cuando esta divergencia es grande (KL » 1), el espacio latente tiende a estar más regularizado y la distribución de probabilidad de los datos generados se vuelve más "suave", pero el modelo puede terminar sobreajustándose a las muestras originales y perder capacidad de generalización. Por otro lado, si la divergencia es muy pequeña (KL « 1), el espacio latente queda menos regularizado y el modelo intenta reproducir con mayor fidelidad la distribución real de los datos de entrada, aunque a costa de generar muestras más ruidosas y posiblemente descuidar detalles esenciales. De esta manera, la elección del valor de KL refleja el compromiso entre capturar información detallada de los datos y mantener una representación latente estable y bien estructurada para la generación.

3.5. Presentación de nuestros modelos de AV y AVC

El desarrollo del modelo de AV empleado en esta investigación se organizó en dos pasos. El primero giró en torno al diseño y validación de la arquitectura del modelo, mientras que el segundo estuvo enfocado en la optimización de dicha arquitectura para la generación de datos sintéticos en cada uno de los conjuntos de datos utilizados. A continuación describiremos brevemente este proceso y la configuración final de los modelos elegidos para los experimentos de aumentación.

3.5.1. Primera versión de modelos

En la primera etapa, se centró el esfuerzo en el diseño de la arquitectura del AV. Este proceso comenzó con la creación de una versión exploratoria del modelo, cuya finalidad era establecer una base sobre la cual iterar en mejoras sucesivas. Se optó por una red con 2 capas en el codificador y en el decodificador, siguiendo la estructura básica descripta precedentemente.

El codificador incluyó dos capas lineales, cada una seguida de una activación ReLU, un diseño que sigue la lógica de la transformación no lineal en un espacio de menor dimensión para luego reconstruir la observación original a partir del espacio latente. Como se discutió en la primera parte, el codificador genera dos vectores, uno para la media y otro para la varianza logarítmica de la distribución latente, componentes críticos para el proceso de reparametrización que permite al modelo generar nuevas muestras en el espacio latente. El decodificador, encargado de reconstruir los datos originales a partir del espacio latente, fue diseñado con una estructura simétrica a la del codificador, utilizando nuevamente activaciones ReLU.

La función de pérdida del modelo combinó la divergencia Kullback-Leibler (KLD) y error cuadrático medio (MSE). La KLD se utilizó para medir la diferencia entre la distribución aprendida por el modelo y una distribución normal estándar, mientras que el error cuadrático medio se empleó para evaluar el error de reconstrucción, es

decir, qué tan bien el modelo era capaz de replicar los datos de entrada a partir del espacio latente.

Este modelo se probó en la generación de datos sintéticos en un dataset de clases binarias: Madelon, y un dataset multiclases: GCM. Para evaluar los datos generados se realizaron experimentos de clasificación utilizando un Perceptron Multicapa (MLP), comparando los resultados obtenidos en el dataset original y en el dataset con muestras sintéticas. Esta evaluación se realizó sobre los datos de testeo.

Inicialmente, la arquitectura empleada resultó insuficiente para capturar la complejidad de los datos, fundamentalmente en los datasets multiclases, lo que llevó a un modelo incapaz de representar con precisión las características latentes, produciendo datos sintéticos de baja calidad.

[grafico de resultados iniciales]

3.5.2. Segunda versión de modelos

En respuesta a los problemas identificados previamente, se exploraron diferentes configuraciones de arquitectura para el AV y el AVC (nos referimos a esto en la sección de experimentos). Se encontró que una arquitectura de tres capas lineales en el codificador y el decodificador, cada una con activaciones ReLU seguidas de normalización por lotes, brindó los mejores resultados.

Esta técnica de normalización fue seleccionada debido a su capacidad para estabilizar y acelerar el proceso de entrenamiento, promoviendo la rápida convergencia y mejorando la precisión de la reconstrucción. Al mitigar el problema de desplazamiento de covariables (*covariate shift*) durante el entrenamiento, la normalización por lotes estabiliza las activaciones intermedias de la red y permite que la información relevante sea conservada a lo largo de las capas.

Dado que el modelo fue ajustado para capturar la estructura de los datos a través de capas lineales y normalización por lotes, mantuvimos la elección de ReLU como función de activación dada su eficiencia computacional.

El modelo resultante fue entrenado con un optimizador Adam y una tasa de aprendizaje en el rango de [1e-5, 1e-3]. Se experimentó con diferentes tamaños del espacio latente, evaluando el equilibrio entre la calidad de reconstrucción y la capacidad de generalización del modelo. Se empleó un término de paciencia para detener el entrenamiento si no se observaba mejora en los datos de test durante 10 épocas consecutivas.

Estos experimentos fueron clave para ajustar el AV a las necesidades específicas de los conjuntos de datos utilizados en la investigación, permitiendo una generación de datos sintéticos que no solo replicara los patrones de los datos originales, sino que también capturara la variabilidad inherente a estos.

Arquitectura del Autocodificador Variacional

La arquitectura del AV está compuesta por tres capas lineales, cada una seguida de una normalización por lotes (Batch Normalization) y una activación ReLU. El proceso de codificación se realiza de la siguiente manera:

$h_1 = \text{ReLU}(\text{BatchNorm}(\mathbf{W_1}x + b_1))$
$h_2 = \text{ReLU}(\text{BatchNorm}(\mathbf{W_2}h_1 + b_2))$
$h_3 = \text{ReLU}(\text{BatchNorm}(\mathbf{W_3}h_2 + b_3))$

Donde:

- $\mathbf{W_1}$ es una matriz de pesos que transforma el vector de entrada x al espacio de características de dimensión $|h_1|$.
- $\mathbf{W_2}$ transforma h_1 a un espacio de características de dimensión $|h_2|$.
- $\mathbf{W_3}$ mantiene la dimensión $|h_2|$ mientras transforma h_3.
- b_1, b_2, y b_3 son los sesgos correspondientes a cada capa.

Después de las tres capas, se generan los vectores latentes μ y $\log(\sigma^2)$ mediante dos capas lineales independientes que también aplican normalización por lotes.

Reparametrización

El vector latente z se obtiene mediante la técnica de reparametrización, donde se introduce ruido gaussiano para permitir la retropropagación del gradiente:

$z = \mu + \sigma \times \epsilon,$

donde ϵ es una variable aleatoria con distribución normal estándar, y σ se calcula a partir de $\log(\sigma^2)$.

Decodificador

El decodificador reconstruye el vector de entrada a partir del vector latente z utilizando una arquitectura de tres capas lineales, cada una seguida por una normalización por lotes y una activación ReLU:

$h_4 = \text{ReLU}(\text{BatchNorm}(\mathbf{W_4}z + b_4)),$
$h_5 = \text{ReLU}(\text{BatchNorm}(\mathbf{W_5}h_4 + b_5)),$
$\hat{x} = \text{BatchNorm}(\mathbf{W_6}h_5 + b_6),$

donde:

- $\mathbf{W_4}$ transforma el vector latente z al espacio de características de dimensión $|h_4|$.
- $\mathbf{W_5}$ transforma h_4 al espacio de características de dimensión $|h_5|$.
- $\mathbf{W_6}$ transforma h_5 de regreso al espacio de la dimensión original de la entrada $|D_{in}|$.
- b_4, b_5, y b_6 son los sesgos correspondientes a cada capa.

Finalmente, la salida $\hat{x}$ es una aproximación reconstruida de la entrada original x.

3.5.3. Modelo AVC para datos multiclase

Para abordar el dataset GCM, que contiene 14 clases con distribuciones desiguales, se creó un AV Condicional (AVC) que combina la capacidad de generación de un AV tradicional con el condicionamiento explícito en las etiquetas de clase. El AVC propuesto permitió una modelización más precisa de los datos, al incorporar información de clase en el proceso de codificación y decodificación.

En escenarios donde los datasets están desbalanceados, los modelos generativos pueden tender a favorecer las clases mayoritarias, ignorando las minoritarias. Para abordar el desbalance de clases que presenta GCM, se implementó una estrategia de

ponderación de clases dentro de la función de pérdida, penalizando de manera diferenciada los errores de reconstrucción en función de la clase, buscando mejorar así la capacidad del modelo para representar adecuadamente las clases minoritarias.

Arquitectura del Autocodificador Variacional Condicional

La arquitectura del AVC se basa en una modificación del AV tradicional para incorporar información adicional en forma de etiquetas. Esta información se concatena tanto en la fase de codificación como en la de decodificación, permitiendo que el modelo aprenda distribuciones condicionales.

Codificador

El codificador del AVC combina la entrada original con las etiquetas antes de ser procesadas por una secuencia de capas lineales (3 capas), cada una seguida por una normalización por lotes (Batch Normalization) y una activación ReLU. El proceso de codificación se realiza de la siguiente manera:

$h_1 = \text{ReLU}(\text{BatchNorm}(\mathbf{W_1}[x, y] + b_1))$,
$h_2 = \text{ReLU}(\text{BatchNorm}(\mathbf{W_2}h_1 + b_2))$,
$h_3 = \text{ReLU}(\text{BatchNorm}(\mathbf{W_3}h_2 + b_3))$,

donde:

- $[x, y]$ es la concatenación del vector de entrada x con las etiquetas y.
- $\mathbf{W_1}$ es una matriz de pesos que transforma el vector combinado $[x, y]$ al espacio de características de dimensión $|h_1|$.
- $\mathbf{W_2}$ transforma h_1 a un espacio de características de dimensión $|h_2|$.
- $\mathbf{W_3}$ mantiene la dimensión $|h_2|$ mientras transforma h_3.

- b_1, b_2, y b_3 son los sesgos correspondientes a cada capa.

Al igual que en el AVC, se generan los vectores latentes μ y $\log(\sigma^2)$ mediante dos capas lineales independientes.

Reparametrización

El vector latente z se obtiene mediante la técnica de reparametrización, similar al AV tradicional:

$z = \mu + \sigma \times \epsilon$

donde ϵ es una variable aleatoria con distribución normal estándar, y σ se calcula a partir de $\log(\sigma^2)$.

Decodificador

El decodificador del AVC reconstruye el vector de entrada a partir del vector latente z y las etiquetas y, utilizando una arquitectura de tres capas lineales, cada una seguida por una normalización por lotes y una activación ReLU:

$h_4 = \text{ReLU}(\text{BatchNorm}(\mathbf{W_4}[z, y] + b_4))$
$h_5 = \text{ReLU}(\text{BatchNorm}(\mathbf{W_5}h_4 + b_5))$
$\hat{x} = \text{BatchNorm}(\mathbf{W_6}h_5 + b_6)$

Donde:

- $[z, y]$ es la concatenación del vector latente z con las etiquetas y.

- $\mathbf{W_4}$ transforma el vector combinado $[z, y]$ al espacio de características de dimensión $|h_4|$ + labels_length.
- $\mathbf{W_5}$ transforma h_4 al espacio de características de dimensión $|h_5|$.
- $\mathbf{W_6}$ transforma h_5 de regreso al espacio de la dimensión original de la entrada $|D_{in}|$.

- b_4, b_5, y b_6 son los sesgos correspondientes a cada capa.

Finalmente, la salida $\hat{x}$ es una aproximación reconstruida de la entrada original x, condicionada por las etiquetas y.

Elementos distintivos respecto a la arquitectura anterior:

- Incorporación de etiquetas: Tanto en el codificador como en el decodificador, se concatenan las etiquetas y con las entradas y el vector latente, respectivamente.
- Dimensiones ajustadas: Se han ajustado las dimensiones de las capas para acomodar las etiquetas, reflejadas en las matrices de pesos y las normalizaciones por lotes.
- Capas adicionales en la fase de decodificación: Se añaden capas y ajustes para manejar las etiquetas adicionales en el proceso de decodificación.

3.6. Configuración de modelos

La búsqueda y ajuste de hiperparámetros para los modelos de AV y AVC ha sido un proceso crucial para optimizar la generación de datos sintéticos. Asumíamos que este proceso era determinante para favorecer la estrategia de selección de características mediante el *algoritmo genético*, por esa razón analizamos distintas opciones para la configuración de los modelos.

Iniciamos esta tarea realizando una búsqueda de hiperparámetros, ajustando aspectos clave como el tamaño de las dimensiones latentes, la tasa de aprendizaje, y el número de neuronas en las diferentes capas. Se utilizaron tanto Grid Search como Optimización Bayesiana (BO). Cada una de estas técnicas tiene sus fortalezas, y la elección entre ellas depende en gran medida del objetivo de la búsqueda. Grid Search, por ejemplo, permite un control total sobre el espacio de búsqueda, lo que es útil para responder preguntas específicas, como la configuración óptima de la dimensión latente. Sin embargo, la BO demostró ser particularmente eficiente en la exploración de un espacio de hiperparámetros más amplio y menos definido, logrando un equilibrio entre la exploración y la explotación que resultó especialmente útil en nuestra investigación dado el tamaño del espacio de búsqueda.

También exploramos técnicas para evitar el sobreajuste, como un mecanismo de paciencia para detener el entrenamiento si no se observaba mejora durante 10 épocas consecutivas en datos de validación. Además, se experimentó con el uso de dropout como técnica de regularización; se probaron tasas de dropout en un rango de 0.05 a 0.5 en distintas configuraciones de AVC, tanto con arquitecturas pequeñas (100-500 neuronas por capa) como grandes (1000-7000 neuronas por capa).

Testeamos distintas métricas de pérdida, como L1 en lugar de MSE. Se realizaron pruebas para evaluar si la L1_loss podría ofrecer mejoras.

Para abordar el problema del desbalance en conjuntos de datos multiclases, se implementaron ajustes específicos en la configuración del modelo y en las estrategias de entrenamiento. El primero consistió en la asignación de pesos diferenciados a las

clases dentro de la función de pérdida, con el objetivo de aumentar la penalización por errores de clasificación en las clases minoritarias. Como vimos en el primer capítulo, el uso de pesos de clase es una técnica comúnmente utilizada para abordar este problema. Estos pesos se calcularon de manera inversamente proporcional a la frecuencia de las clases en el conjunto de datos, lo que significa que las clases con menos instancias recibieron pesos más altos. Al incorporar estos pesos de clase en la función de pérdida, nos aseguramos que los errores en las clases con menor representación tuvieran un impacto mayor durante el proceso de optimización, logrando un aprendizaje más equilibrado.

Además de ajustar la función de pérdida para mitigar el desbalance, se implementó una estrategia de muestreo ponderado en el proceso de entrenamiento. A través de un muestreador aleatorio ponderado, nos aseguramos que cada mini-lote de datos durante el entrenamiento tuviera una representación equilibrada de cada clase, asignando mayor probabilidad de ser seleccionadas a las instancias de clases minoritarias.

Para garantizar que la combinación de ambas estrategias no tuviera un efecto indeseado en el desempeño del modelo, se llevaron a cabo experimentos con el fin de validar la aplicación simultánea de ambas técnicas. Un efecto a evitar era, precisamente, que el modelo se volviera excesivamente sensible a las clases minoritarias, en detrimento de su capacidad para generalizar correctamente en clases mayoritarias. Los resultados de estos experimentos mostraron que, implementadas en conjunto, ambas estrategias lograban los mejores resultados.

3.7. Resultados obtenidos en los experimentos

Los resultados obtenidos en nuestros experimentos revelaron que un Perceptron Multicapa (MLP) entrenado con datos sintéticos generados por un AV/AVC puede igualar o incluso superar en ciertos casos el rendimiento de un MLP entrenado con datos reales.

Para dicha evaluacion, siguiendo a Fajardo et al. (2021), *comparamos las métricas alcanzadas por un MLP entrenado con datos sintéticos y evaluado con datos reales de testeo, con las métricas obtenidas por un MLP entrenado con datos reales y también evaluado en los mismos datos de testeo.* En todos los casos se emplearon las particiones originales de los datos, a fin de propiciar la comparación de nuestros resultados con los obtenidos en la literatura. El indicador particular empleado para la comparación fue el de **exactitud**, que mide la proporción de predicciones correctas sobre el total de predicciones realizadas.

Además, se emplearon otras métricas para completar nuestro análisis, a saber: la *precisión*, el *recall* y el *F1 Score*. Las métricas empleadas para dicha comparación fueron las siguientes: la *precisión* mide la proporción de predicciones positivas correctas sobre el total de predicciones positivas realizadas, es decir, qué tan preciso es el modelo al identificar casos positivos. El *recall* (o sensibilidad) indica la proporción de verdaderos positivos detectados sobre el total de positivos reales, reflejando la capacidad del modelo para capturar todos los casos positivos. El *F1 Score* es la media armónica de la precisión y el recall, proporcionando un equilibrio entre ambos y siendo especialmente útil cuando existe un desbalance en las clases. Otras métricas que utilizamos fueron el *Promedio Macro* y el *Promedio Ponderado*. Los indicadores de *Promedio Macro* calculan la media aritmética de las métricas individuales de cada clase sin considerar el número de muestras por clase, otorgando igual peso a todas las clases. Esto

es útil para evaluar el rendimiento general en presencia de clases desbalanceadas. Por otro lado, el *Promedio Ponderado* tiene en cuenta el soporte (número de muestras) de cada clase al calcular la media, lo que significa que las clases con más muestras influyen más en el promedio general. Esto proporciona una visión más realista del rendimiento cuando hay clases dominantes.

Entonces en la siguiente figura se muestran los resultados obtenidos en los experimentos realizados para los cuatro conjuntos de datos estudiados, donde comparamos datos sintéticos con datos reales:

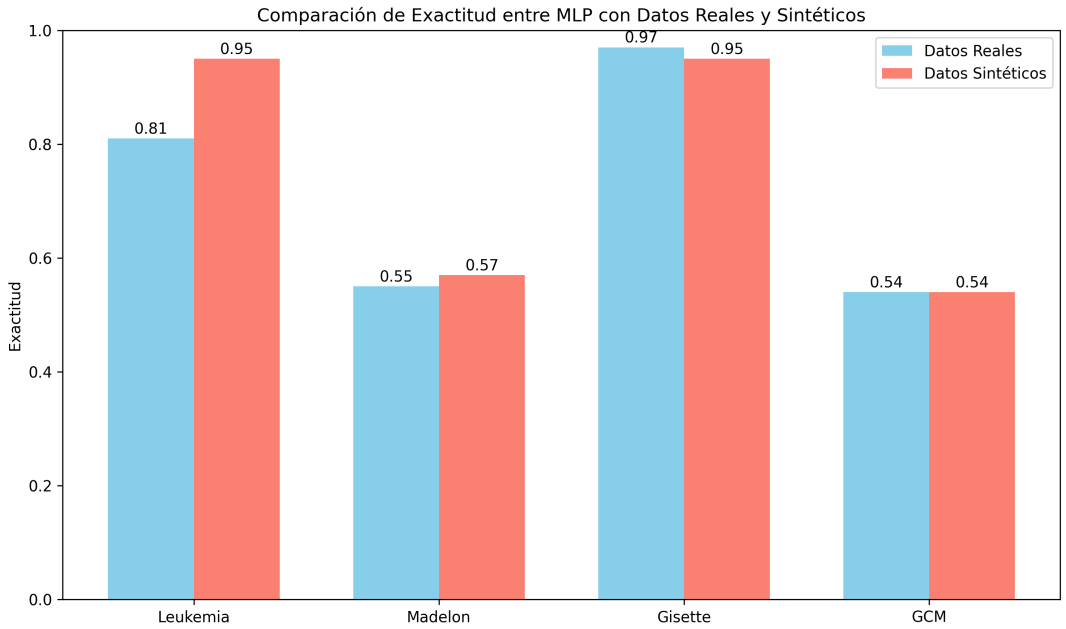

FIGURE 3.4: MLP con datos sintéticos vs MLP con datos reales

Estos resultados son particularmente relevantes ya que sugieren que, bajo ciertas condiciones, los datos sintéticos pueden ser tan útiles como los datos reales para el entrenamiento de modelos predictivos. Como se aprecia en el gráfico, este fenómeno se observó de manera consistente en tres de los cuatro conjuntos de datos de nuestro estudio: Leukemia, Madelon y GCM, donde la precisión y la exactitud del modelo entrenado con datos sintéticos alcanzaron o superaron las métricas obtenidas con los datos originales. En el caso del dataset Gisette, el modelo entrenado con datos sintéticos estuvo muy cerca de igualar el rendimiento del modelo entrenado con datos reales.

Veamos a continuación los resultados particulares de cada conjunto de datos.

3.7.0.1. Leukemia

Modelo	Clase	Precisión	Recall	F1 Score	Soporte
MLP con datos reales	0	0.76	1.00	0.87	13
	1	1.00	0.56	0.71	9
Exactitud				0.81	22
Promedio Macro		0.88	0.78	0.79	22
Promedio Ponderado		0.86	0.82	0.80	22

Modelo	Clase	Precisión	Recall	F1 Score	Soporte
MLP con datos sintéticos	0	0.93	1.00	0.96	13
	1	1.00	0.89	0.94	9
Exactitud				**0.95**	22
Promedio Macro		0.96	0.94	0.95	22
Promedio Ponderado		0.96	0.95	0.95	22

El cuadro anterior muestra el rendimiento de un MLP entrenado con datos reales frente a uno entrenado con datos sintéticos en el conjunto de datos Leukemia. Se observa que el modelo entrenado con datos sintéticos logra una mayor exactitud (**95 %**) en comparación con el modelo entrenado con datos reales (**81 %**). Las métricas de *precisión*, *recall* y *F1 Score* son superiores en casi todas las clases y promedios cuando se utiliza el modelo sobre datos sintéticos.

Estos resultados indican que el modelo entrenado con datos sintéticos no solo es más preciso, sino que también tiene un rendimiento más equilibrado entre las clases, mejorando la capacidad de generalización y discriminación entre las categorías. La mejora en las métricas sugiere que el modelo es más eficaz en la identificación correcta de ambas clases, reduciendo tanto los falsos positivos como los falsos negativos.

3.7.0.2. Madelon

Modelo	Clase	Precisión	Recall	F1 Score	Soporte
MLP con datos reales	0	0.56	0.55	0.55	396
	1	0.54	0.55	0.55	384
Exactitud				0.55	780
Promedio Macro		0.55	0.55	0.55	780
Promedio Ponderado		0.55	0.55	0.55	780
MLP con datos sintéticos	0	0.56	0.72	0.63	396
	1	0.59	0.42	0.49	384
Exactitud				**0.57**	780
Promedio Macro		0.58	0.57	0.56	780
Promedio Ponderado		0.58	0.57	0.56	780

El conjunto de datos Madelon presenta un caso donde el MLP entrenado con datos sintéticos logra una ligera mejora en la exactitud (**57 %**) en comparación con el modelo entrenado con datos reales (**55 %**). Los promedios macro y ponderado también muestran ligeras mejoras al utilizar datos sintéticos.

3.7.0.3. Gisette

Modelo	Clase	Precisión	Recall	F1 Score	Soporte
MLP con datos reales	0	0.98	0.98	0.98	904
	1	0.98	0.98	0.98	896
Exactitud				**0.97**	1800
Promedio Macro		0.98	0.98	0.98	1800
Promedio Ponderado		0.98	0.98	0.98	1800

Modelo	Clase	Precisión	Recall	F1 Score	Soporte
MLP con datos sintéticos	0	0.95	0.97	0.96	904
	1	0.97	0.95	0.96	896
Exactitud				0.95	1800
Promedio Macro		0.96	0.96	0.96	1800
Promedio Ponderado		0.96	0.96	0.96	1800

En el conjunto de datos Gisette , se observa que el modelo entrenado con datos reales supera al modelo entrenado con datos sintéticos. El MLP con datos reales alcanza una exactitud del **97 %**, mientras que el modelo con datos sintéticos logra un **95 %**. Sin perjuicio de ello, el modelo entrenado con datos sintéticos logró una buena performance en la mayoría de las métricas evaluadas.

3.7.0.4. GCM

Modelo	Clase	Precisión	Recall	F1 Score	Soporte
MLP con datos reales	0	0.14	0.25	0.18	4
	1	0.00	0.00	0.00	1
	2	1.00	1.00	1.00	3
	3	1.00	0.33	0.50	6
	4	0.89	1.00	0.94	8
	5	0.40	0.67	0.50	3
	6	0.80	0.80	0.80	5
	7	0.50	0.75	0.60	4
	8	0.33	0.25	0.29	4
	9	0.25	0.67	0.36	3
	10	1.00	0.25	0.40	4
	11	1.00	0.67	0.80	3
	12	1.00	0.25	0.40	4
	13	1.00	0.20	0.33	5
Exactitud				**0.54**	57
Promedio Macro		0.67	0.51	0.51	57
Promedio Ponderado		0.74	0.54	0.56	57
MLP con datos sintéticos	0	1.00	0.25	0.40	4
	1	0.00	0.00	0.00	1
	2	1.00	0.67	0.80	3
	3	1.00	0.17	0.29	6
	4	1.00	1.00	1.00	8
	5	0.43	1.00	0.60	3
	6	1.00	0.80	0.89	5
	7	0.10	0.25	0.14	4
	8	0.67	0.50	0.57	4
	9	0.50	0.33	0.40	3
	10	0.00	0.00	0.00	4
	11	1.00	0.67	0.80	3
	12	1.00	0.75	0.86	4
	13	0.21	0.60	0.32	5

Modelo	Clase	Precisión	Recall	F1 Score	Soporte
Exactitud				**0.54**	57
Promedio Macro		0.64	0.50	0.50	57
Promedio Ponderado		0.70	0.54	0.55	57

En el conjunto de datos GCM ambos modelos alcanzan la misma exactitud (**54 %**). Sin embargo, el análisis detallado por clase revela diferencias en el rendimiento. El modelo entrenado con datos sintéticos muestra una mayor precisión en varias clases (0, 2, 3, 4, 6, 8, 11, 12), pero también presenta casos donde falla completamente (clases 1 y 10). Ambos modelos tienen dificultades con la clase 1, que tiene el menor soporte (1 muestra). El *Promedio Macro* y *Promedio Ponderado* son similares en ambos casos, lo que sugiere que, a pesar de las diferencias en el rendimiento por clase, los modelos son comparables en su capacidad general de clasificación. Estos resultados reflejan la complejidad inherente al trabajar con datos altamente desbalanceados y multiclase, donde la generación de datos sintéticos puede mejorar el rendimiento en algunas clases pero también puede introducir sesgos en otras.

3.8. Otras consideraciones emergentes de los experimentos

Respecto de los hallazgos realizados a lo largo de todo el proceso de experimentación encontramos algunos aspectos dignos de mención.

Las pruebas con diferentes arquitecturas proporcionaron información valiosa. Como mencionamos antes, se exploraron modelos AV de tres y cuatro capas, así como AVC con múltiples capas, pero no se observaron mejoras significativas al aumentar la complejidad. En particular, se encontró que las configuraciones más simples (i.e. 3 capas), ofrecían resultados tan buenos o incluso mejores que sus contrapartes más complejas. Esta observación refuerza la idea de que, en algunos casos, la simplicidad puede ser preferible y que el sobredimensionamiento de la arquitectura no necesariamente se traduce en mejores resultados.

Un aspecto valioso se refiere a las dimensiones latentes de los modelos generativos estudiados. A medida que se ampliaba la búsqueda de hiperparámetros, se descubrió que las mejores configuraciones para la variable latente no eran necesariamente las más grandes. De hecho, en muchos casos, valores para la dimensión latente entre 3 y 100 ofrecieron los mejores resultados. Este tamaño en la dimensión latente es consistente con la utilizada en Ai et al. (2023). Esto, que inicialmente puede ser contraintuitivo, ya que se podría suponer que un mayor espacio latente permitiría capturar más complejidad en los datos; sugiere que un espacio latente excesivamente grande puede introducir ruido y hacer que el modelo pierda la capacidad de generalizar correctamente.

Un resultado interesante fue que el incremento en el número de muestras sintéticas no estuvo siempre acompañado por un incremento en el desempeño de los clasificadores. Es decir, encontramos que los resultados de clasificación no necesariamente tienen una relación positiva y directa con el número de muestras sintéticas empleadas en el entrenamiento.

Este comportamiento se constató en todos los conjuntos de datos estudiados, pero se evidenció con mayor claridad en el dataset GCM. En este problema, se lograron

buenos rendimientos incrementando las observaciones del conjunto de datos de entrenamiento de 190 a 3000 muestras balanceadas, con 214 observaciones por clase. Esta estrategia de aumentación, donde se igualan las cantidades de observaciones de todas las clases, es usada en Fajardo et al. (2021) con buenos resultados. En nuestros experimentos el aumento resultó en una mejora significativa en la performance del modelo, logrando igualar los resultados obtenidos con el clasificador MLP entrenado con datos reales. Sin embargo, al continuar incrementando la cantidad de datos sintéticos a 6000 muestras, se observó una degradación en el rendimiento.

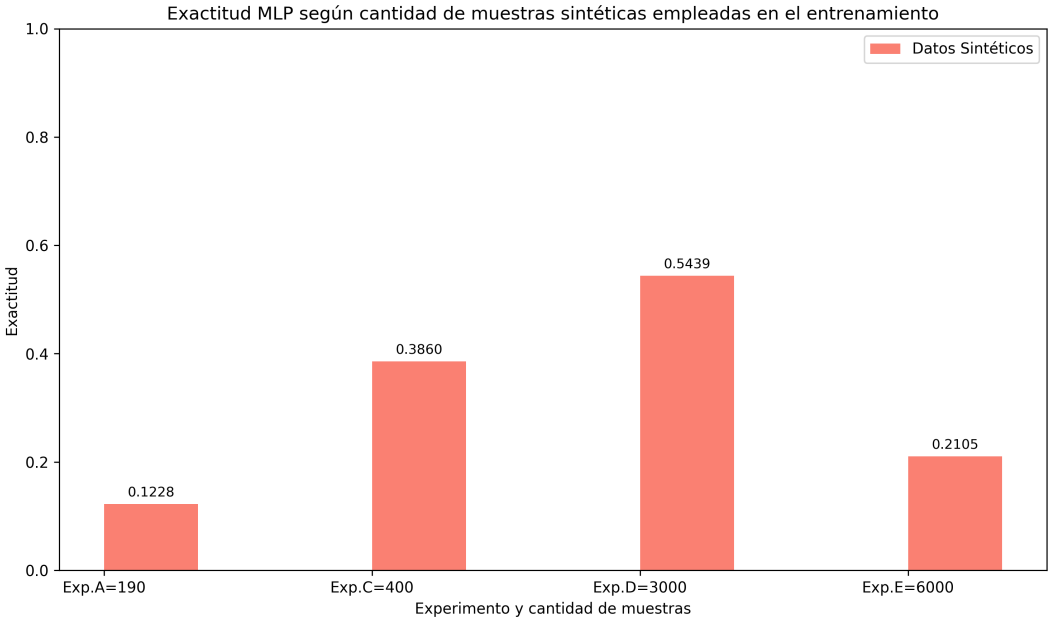

FIGURE 3.5: Exactitud según cantidad de muestras sintéticas

Esto sugiere la existencia de un umbral en la cantidad de datos sintéticos que, una vez superado, introduce ruido en el modelo en lugar de aportar valor. Este ruido puede estar relacionado con el solapamiento de las fronteras de decisión en las muestras generadas, lo que aumenta el error y disminuye la precisión del modelo.

Finalmente, otros emergentes que podemos destacar son los siguientes: - La implementación de un término de paciencia tuvo un impacto positivo en la calidad de los modelos, particularmente en el caso del AVC, - Para el caso del AVC, el uso de L1_loss no proporciona un beneficio claro sobre el MSE para la tarea de generación de datos sintéticos, y - El dropout no aporta beneficios en configuraciones ya optimizadas del AVC para este tipo de tareas.

Estos resultados llevaron a una reflexión sobre la falta de impacto positivo de ciertos ajustes, como la introducción de L1_loss y dropout. Es probable que la estabilidad y el buen rendimiento de las configuraciones ya validadas de AV y AVC, alcanzados a través de numerosos experimentos, limiten el potencial de mejora adicional mediante estos métodos.

3.9. Conclusiones

En primer lugar pudimos determinar que los datos sintéticos generados por un AV/AVC pueden replicar la distribución de los datos reales en los conjuntos de datos

estudiados, y por lo tanto, pueden ser tan útiles como aquellos para el entrenamiento de modelos predictivos.

Por otro lado, la búsqueda de hiperparámetros y el ajuste de la arquitectura del AV y AVC revelaron la importancia de un enfoque balanceado que evite tanto la simplicidad excesiva como la complejidad innecesaria. Los resultados obtenidos muestran que, bajo ciertas configuraciones, los datos sintéticos pueden igualar o superar la utilidad de los datos reales en la formación de modelos predictivos, aunque la eficiencia y la calidad de estos resultados dependen en gran medida de la cuidadosa calibración de los hiperparámetros y de la adecuada elección de la arquitectura del modelo.

Asimismo, los experimentos parecen reflejar que los beneficios de la aumentación de datos tienen un límite. Superado este umbral, la generación adicional de datos no solo deja de ser útil, sino que puede ser perjudicial, como se evidenció en nuestros experimentos. Este fenómeno destaca la importancia de una cuidadosa calibración en la cantidad de datos sintéticos generados, especialmente en conjuntos de datos con características complejas y de grandes dimensiones.

Capítulo 4

Algoritmos Genéticos

En este capítulo presentamos una descripción general de los *algoritmos genéticos (AGs)* como estrategia de selección de características, exponemos brevemente sus componentes y describimos la implementación realizada en nuestro trabajo. Seguidamente plantearemos la integración del AG con la generación sintética de datos mediante AV como etapa de preprocesamiento. Según el planteo hecho desde el inicio, la integración de ambas tecnologías permitirá resolver problemas de escasez muestral, ruido y alta dimensionalidad. Para terminar, expondremos los experimentos realizados con esta integración de AV-AG, y los resultados obtenidos en los problemas elegidos en este trabajo.

4.1. Elementos básicos de los Algoritmos Genéticos

Los Algoritmos Genéticos (AGs) son una clase de algoritmos de optimización inspirados en la evolución biológica y en la teoría de la selección natural. Los AGs se basan en el concepto de evolución de una población de soluciones potenciales a lo largo de múltiples generaciones, utilizando operadores genéticos como la selección, el cruce y la mutación para generar nuevas soluciones y mejorar la calidad de las mismas. Remitimos al lector al **Anexo A** para una descripción más detallada de los componentes los AGs.

A pesar de su extraordinaria simpleza -o quizas gracias a ella-, los AG constituyen algoritmos robustos, capaces de encontrar soluciones efectivas en una amplia variedad de problemas de optimización. Dicha robustez está determinada, como bien sostiene Goldberg (1989), por una serie de características distintivas, que fortalecen su configuración de búsqueda, a saber: a) operan sobre un espacio *codificado* del problema y no sobre el espacio en su representación original; b) realizan la exploración evaluando una *población de soluciones* y no soluciones individuales; c) tienen como guía una *función objetivo* (también llamada *función de aptitud*) que no requiere derivación u otras funciones de cálculo; y d) suponen *métodos probabilísticos de transición* (operadores estocásticos) y no reglas determinísticas. Estas características permiten a los AG superar restricciones que tienen otros métodos de optimización, condicionados -por ejemplo- a espacios de búsqueda continuos, diferenciables o unimodales. Por ello, su aplicación se ha difundido notablemente, trascendiendo los problemas clásicos de optimización, aplicándose en distintas tareas (Vie, Kleinnijenhuis, and Farmer 2021) y a lo largo de diversas industrias (Jiao et al. 2023).

La importancia de los AGs como herramientas de optimización, adquiere especial preeminencia en el problema de *selección de características* (Jiao et al. 2023). En el marco de este algoritmo, cada individuo (muestra) representa una solución candidata,

con un perfil genético particular determinado por un subconjunto de características. La búsqueda de las mejores soluciones comienza con la selección de una población inicial de individuos y un subconjunto de características generados aleatoriamente. Este subconjunto se evalúa utilizando una función de aptitud, y los individuos con mejor rendimiento (puntaje) son seleccionados para la reproducción. Este proceso continúa durante un cierto número de generaciones hasta que se cumple una condición de terminación (Goldberg, David E. 1989).

Este mecanismo simple constituye un eficaz método de selección gracias a la capacidad de explorar el problema dividiéndolo en subespacios de características y, al mismo tiempo, explotar las regiones de mayor valor en cada subespacio (Goldberg, David E. 1989).[1]

Dicho lo anterior, no es menos cierto que la capacidad de selección de los AGs depende de la evaluación de aptitud que orienta la búsqueda de las mejores soluciones, y tal evaluación descansa -finalmente- en la disponibilidad de datos. En efecto, la existencia y número de individuos condiciona la función objetivo y por esa vía también al proceso de selección de características de los AGs. La disponibilidad de datos resulta así un factor clave para la selección. Este requerimiento, vinculado particularmente a la función objetivo, se presenta no solo cuando se utiliza como evaluador a modelos complejo de aprendizaje automático (que demandan una cantidad creciente de muestras de entrenamiento)[2], sino también cuando se trabaja sobre datos cuyas clases se encuentran desbalanceadas (Fajardo et al. 2021; Blagus and Lusa 2013). En ambos escenarios, la falta de información suficiente degrada la capacidad informativa de la función objetivo (Hastie, Tibshirani, and Friedman 2009), afectando gravemente el proceso de selección de características.

En esa línea, el problema de la disponibilidad de datos en los proyecto de selección de características -sea dentro o fuera del campo de los AGs-, ha encontrado en las estrategias de aumentación una posible solución (Gm et al. 2020). Entre esas estrategias, los Autoencoders Variacionales (en adelante AV) han adquirido popularidad, superando a métodos tradicionales (ej. sobremuestreo (Blagus and Lusa 2013)) y -en ciertos casos- también a otro modelos generativos basados de redes neuronales profundas (Fajardo et al. 2021).

4.2. Implementación de un Algoritmo Genético para la selección de características

En esta sección se describe la implementación que hemos realizado de AG, usando la librería `DEAP` de Python, para la selección de características en problemas de alta dimensionalidad. La implementación se centra en la optimización simultánea de la precisión de un modelo de clasificación y la reducción del número de características

[1] Ambas funciones -exploración y explotación- permiten al algoritmo reconfigurar el espacio de búsqueda y poner a prueba sus complejas dependencias. Como vimos, el procedimiento es orientado por una función de aptitud que evalúa las distintas posibilidades combinatorias encontradas por el algoritmo y retroalimenta el proceso exploratorio. La dinámica completa tiene como resultado un procedimiento experimental de búsqueda y selección capaz de reconocer soluciones próximas al óptimo.

[2]

1.

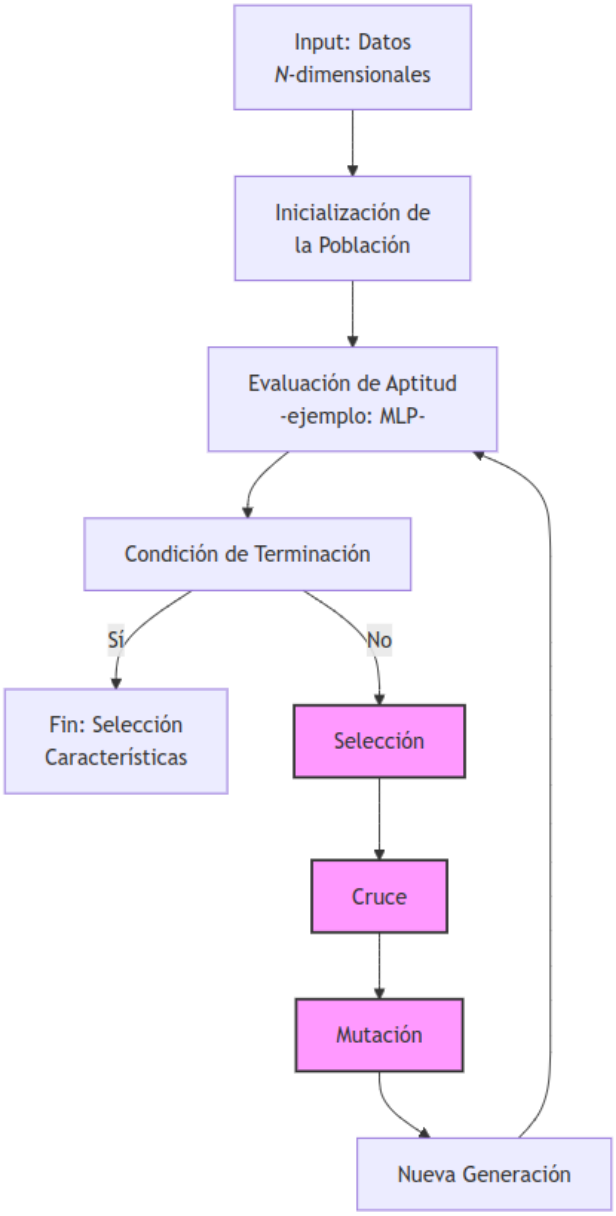

FIGURE 4.1: Diagrama de un Algoritmo Genético

seleccionadas, utilizando operadores genéticos clásicos como el cruce y la mutación, junto con técnicas avanzadas de evaluación y selección.

La configuración inicial del AG a lo largo de los distintos experimentos que formaron parte de esta investigación (y que revisaremos en detalle en el próximo capítulo) incluyó:

- **Población inicial**: individuos generados aleatoriamente, cada uno representado como una lista de bits de longitud igual al número de características.
- **Función de aptitud**: Maximización de la precisión del modelo de clasificación y minimización del número de características activas.
- **Operadores genéticos**: Selección por torneo, cruce de dos puntos y mutación por inversión de bits.
- **Parámetros del AG**: Probabilidad de mutación (e.g.`PROB_MUT = 0.1`), probabilidad de cruce (e.g. `PX = 0.75`), número máximo de generaciones (e.g.`GMAX = 15`).
- **Evaluación de características**: Análisis de la frecuencia de activación de las características a lo largo de las generaciones.
- **Criterio de terminación**: Convergencia o número máximo de generaciones alcanzado.
- **Análisis de resultados**: Selección de las características más relevantes basadas en su frecuencia de activación.

En cada experimento, la implementación del AG comienza con la definición de los componentes básicos del algoritmo. Se define una función de aptitud (`fitness`) orientada a maximizar, que evalúa cada individuo en función de dos criterios: la precisión (`accuracy`) del modelo de clasificación entrenado con las características seleccionadas, y la fracción de características activas. Esta función de aptitud está diseñada para balancear la necesidad de un modelo predictivo preciso con la simplicidad y la eficiencia del modelo, evitando el sobreajuste y facilitando la interpretación del modelo final.

Los individuos, representados como listas de bits, se construyen utilizando una función de construcción de genes que genera un bit aleatorio basado en una probabilidad definida (`p_indpb`). Estos individuos se agrupan en una población inicial, que luego se somete a un proceso evolutivo. Durante la evolución, los individuos se seleccionan mediante la técnica de torneo, donde aquellos con mejor aptitud tienen una mayor probabilidad de ser elegidos para reproducción. Los individuos seleccionados se cruzan utilizando un operador de cruce de dos puntos (`cxTwoPoint`), que intercambia segmentos de los cromosomas de los padres para generar descendientes con combinaciones genéticas novedosas. Posteriormente, se aplica un operador de mutación que invierte los bits en el cromosoma según la probabilidad de mutación definida, asegurando que el AG mantenga la capacidad de explorar nuevas regiones del espacio de búsqueda.

A lo largo de las generaciones, el AG monitoriza y registra diversas estadísticas de la población, como la aptitud promedio, la precisión y el número de genes activos. Estas métricas permiten evaluar el progreso del algoritmo y la convergencia hacia soluciones óptimas. Al final del proceso evolutivo, se realiza un análisis de las características seleccionadas, calculando la frecuencia de activación de cada característica a lo largo de las generaciones y seleccionando las más recurrentes como las más relevantes. Este enfoque permite identificar un subconjunto óptimo de características que no solo maximiza la precisión del modelo, sino que también minimiza su complejidad.

Con fines ilustrativos, en el **Apéndice A** se presenta un script genérico de un Algoritmo Genético implementado con la librería `DEAP` de Python, que puede ser adaptado para la selección de características en problemas de alta dimensionalidad. Este script incluye la definición de los componentes básicos del AG que vimos antes, como la función de aptitud, los operadores genéticos y los parámetros del algoritmo, así como la configuración de la población inicial y la ejecución del proceso evolutivo a lo largo de 15 generaciones.

Capítulo 5

Próximos Pasos: complejización del modelo y exploración de nuevas arquitecturas

A partir de los resultados obtenidos en los experimentos, se torna evidente que la aumentación de datos mediante Autocodificadores Variacionales y su combinación con Algoritmos Genéticos es una estrategia prometedora en la selección de características. Para maximizar el potencial de este enfoque, es posible avanzar en varias direcciones, tanto en términos de complejización de los modelos AV y AVC, como en la integración y encadenamiento de modelos más complejos. En este capítulo, abordaremos los próximos pasos necesarios para continuar mejorando la eficiencia y precisión de los modelos aplicados en escenarios de alta dimensionalidad y ruido, explorando nuevas arquitecturas y estrategias de integración.

5.1. 1. Complejización del modelo AV

Entendemos que uno de los primeros pasos para mejorar la calidad de los datos sintéticos generados es la complejización de los modelos AV y AVC. Aunque las versiones creadas en el marco de la presente investigación han mostrado ser efectivas en los contextos de experimenatión planteados, como en *Madelon* y en *GCM*, es necesario optimizar su capacidad para capturar mejor las estructuras subyacentes de los datos reales. Para lograr esto, se proponen dos líneas de trabajo:

5.1.1. Mejora en la función de pérdida

La función de pérdida del AV/AVC desempeña un papel fundamental en la calidad de las muestras generadas. Como vimos en los experimentos de GCM, la divergencia KL puede dominar el proceso de regularización del espacio latente, lo que lleva a una reconstrucción insuficiente de los datos originales. Para mitigar este problema, propondríamos -tal cual lo adelantado en el Capítulo 4- una modificación de la función de pérdida, donde se ajusten los pesos entre la pérdida de reconstrucción y la divergencia KL, dando mayor importancia a la precisión en la reconstrucción. Adicionalmente, se pueden explorar técnicas como la Inferencia Variacional Recocida (*Annealed Variational Inference*, Huang et al. (2018)), que ajusta gradualmente el peso de la divergencia KL durante el entrenamiento, permitiendo una transición más suave y efectiva en la regularización del espacio latente.

5.1.2. Uso de AVs jerárquicos

Otra línea de trabajo posible es la utilización de Autocodificadores Variacionales Jerárquicos (Vahdat and Kautz (2020)). Estos modelos permiten la representación de características en múltiples niveles de abstracción, lo que podría ser particularmente útil para capturar relaciones complejas en conjuntos de datos como *GCM*. Al incorporar un nivel adicional de complejidad, los HVAEs podrían generar datos sintéticos que no solo preserven mejor la estructura de los datos originales, sino que también mejoren la capacidad del AG para identificar características relevantes en escenarios de alta dimensionalidad.

5.2. 2. Integración AV-AG optimizada

Aunque los resultados iniciales con la integración AV-AG son alentadores, es posible explorar maneras más eficientes de combinar ambos modelos. El flujo de trabajo encadenado que involucra *selección-generación-selección* mostró ser efectivo en los experimentos con *GCM*, pero podría beneficiarse sustancialmente con ciertos ajustes adicionales en su implementación.

5.2.1. Selección dinámica de características

Así, una opción es, en lugar de aplicar el AG sobre la totalidad de las características de manera uniforme, podríamos implementar un proceso dinámico de selección de características en múltiples etapas. En este enfoque, el AG se aplicaría inicialmente sobre subconjuntos reducidos de características, optimizando en función de la estabilidad y relevancia de los atributos seleccionados. Finalmente, se combinarían los subconjuntos con mejor desempeño, para formar un dataset de baja dimensiones y alto contenido informativo. Posteriormente, los AVs generarían datos sintéticos solo tomando como imputs este último dataset, reduciendo áun más el ruido y permitiendo una exploración más eficiente del espacio de búsqueda. Todo esto, alineado con la técnica implementada en este trabajo, pero radicalizada en su intención y objetivo.

5.2.2. Optimización conjunta de AV y AG

En los experimentos actuales, el AV y el AG se entrenan de manera secuencial y separada, lo que puede limitar la sinergia entre ambos modelos. Un enfoque alternativo sería la optimización conjunta, donde los parámetros de ambos modelos se ajusten simultáneamente durante el entrenamiento. De esta forma, el AV/AVC podría generar muestras sintéticas que maximicen directamente la eficacia del AG en la selección de características, permitiendo una retroalimentación continua entre ambos procesos y una mejora en la calidad del conjunto de datos aumentado.

Reconocemos que, según la experiencia ganada en esta investigación, ajustar los parámetros del AVC y AG, aún de forma independiente, no es un proceso trivial. Particulamente la cantidad de muestras sintéticas en la etapa generativa y el tamaño del cromosoma activo en la etapa de selección resultan parámetros de un impacto crítico en los resultados de la arquitectura.

Pero advertimos también que, en el diseño donde el AV genera muestras sintéticas que luego son utilizadas por el AG para la selección, los modelos no comparten información durante sus respectivas fases de entrenamiento. La **optimización conjunta** permitiría entrenar ambos modelos de forma simultánea, posibilitando una retroalimentación directa y continua entre los procesos de generación de datos sintéticos (por

el AV) y la selección de características (por el AG). Es decir, se ajustaría el proceso de generación de datos en función de la capacidad del AG para seleccionar características relevantes, logrando que el AV genere datos específicamente diseñados para maximizar el rendimiento del AG.

Aunque esta propuesta entaña desafíos inocultables (diseño de una función de pérdida, coordinación entre los procesos de optimización, capacidad computacional, por mencionar algunas), también ofrecería beneficios de gran valor. En particular, podríamos disponer de una mayor sinergia entre generación y selección. Esto se produciría a través de una retroalimentación directa entre los modelos, haciendo que el AV se adapte mejor a las necesidades del AG, generando datos que aborden mejor los desafíos específicos de selección de características en cada problema. Al entrenar el AV para generar datos que optimicen el desempeño del AG, el proceso de búsqueda de características relevantes podría volverse más eficiente. El AG podría explorar de manera más efectiva el espacio de características, identificando subconjuntos más precisos y reduciendo la dimensionalidad sin perder información relevante.

5.3. 3. Exploración de encadenamientos y stacks de modelos

Los resultados obtenidos sugieren que una única iteración del proceso AV-AG puede no ser suficiente para capturar completamente las relaciones entre características en problemas altamente complejos. En este contexto, la construcción de arquitecturas más complejas mediante el encadenamiento de varios modelos (stacks) o la integración de ensambles, similar a los Bosques Aleatorios, se presenta como una vía interesante de investigación.

Una posibilidad es el diseño de un *stack* de AVs y AGs, donde varias instancias de ambos modelos se encadenen de manera secuencial o paralela. En este esquema, por ejemplo, una primer secuencia AG-AV generaría un conjunto de datos sintéticos, que luego alimentaría a una segunda secuencia de AG-AG con configuraciones más específicas. Este proceso de encadenamiento podría permitir una mayor concentración de la generación y selección, especialmente en conjuntos de datos donde las relaciones entre las características son extremadamente complejas.

Al igual que los Bosques Aleatorios combinan múltiples árboles de decisión para mejorar la precisión y la robustez del modelo, se puede explorar la creación de un ensamble de AVs y AGs. Este enfoque involucraría el entrenamiento de múltiples AVs y AGs con diferentes configuraciones y subconjuntos de datos, cuyas salidas se combinarían para producir una solución más robusta. Los ensambles suelen ser efectivos para reducir la varianza de los modelos individuales, lo que podría resultar en una selección de características más estable y en un rendimiento más consistente.

5.4. 4. Exploración de arquitecturas híbridas y meta-aprendizaje

Por último, se abre la posibilidad de explorar arquitecturas híbridas que combinen AVs y AGs con otros enfoques de aprendizaje automático, como los algoritmos de meta-aprendizaje. Estos modelos podrían ser entrenados para aprender a seleccionar automáticamente los mejores hiperparámetros y configuraciones para cada conjunto de datos, adaptándose dinámicamente a las características específicas del problema.

En lugar de fijar los parámetros del AG a priori, el meta-aprendizaje permitiría que el AG aprenda automáticamente cuáles son los mejores parámetros en función de la estructura de los datos. Este enfoque podría incluir la selección adaptativa del tamaño del cromosoma activo, las tasas de mutación y cruce, y el número de generaciones, optimizando el rendimiento del AG en cada iteración.

5.5. Conclusión

La investigación hasta el momento ha demostrado que la combinación de Autocodificadores Variacionales y Algoritmos Genéticos es una estrategia prometedora para la selección de características en escenarios de alta dimensionalidad y ruido. Sin embargo, los resultados también sugieren que existe un espacio amplio para la creatividad y mejora mediante la complejización de los modelos AV, la optimización de la integración AV-AG, y la exploración de arquitecturas más avanzadas basadas en encadenamientos y ensambles de modelos. Considerando la complejidad de los dataset reales, quizas los próximos pasos deberían priorizar la construcción de soluciones más robustas y adaptativas, capaces de abordar la complejidad de los problemas con la menor cantidad de presupuestos posibles.

References

Ai, Qingzhong, Pengyun Wang, Lirong He, Liangjian Wen, Lujia Pan, and Zenglin Xu. 2023. "Generative Oversampling for Imbalanced Data via Majority-Guided VAE." February 14, 2023. http://arxiv.org/abs/2302.10910.

Almugren, Nada, and Hala Alshamlan. 2019. "A Survey on Hybrid Feature Selection Methods in Microarray Gene Expression Data for Cancer Classification." *IEEE Access* 7: 78533–48. https://doi.org/10.1109/ACCESS.2019.2922987.

Blaauw, Merlijn, and Jordi Bonada. 2016. "Modeling and Transforming Speech Using Variational Autoencoders." In *Interspeech 2016*, 1770–74. ISCA. https://doi.org/10.21437/Interspeech.2016-1183.

Blagus, Rok, and Lara Lusa. 2013. "SMOTE for High-Dimensional Class-Imbalanced Data." *BMC Bioinformatics* 14 (1): 106. https://doi.org/10.1186/1471-2105-14-106.

Bolón-Canedo, Verónica, Noelia Sánchez-Maroño, and Amparo Alonso-Betanzos. 2015. *Feature Selection for High-Dimensional Data*. Artificial Intelligence: Foundations, Theory, and Algorithms. Cham: Springer International Publishing. https://doi.org/10.1007/978-3-319-21858-8.

Doersch, Carl. 2021. "Tutorial on Variational Autoencoders." January 3, 2021. http://arxiv.org/abs/1606.05908.

El-Hasnony, Ibrahim M., Sherif I. Barakat, Mohamed Elhoseny, and Reham R. Mostafa. 2020. "Improved Feature Selection Model for Big Data Analytics." *IEEE Access* 8: 66989–7004. https://doi.org/10.1109/ACCESS.2020.2986232.

Fajardo, Val Andrei, David Findlay, Charu Jaiswal, Xinshang Yin, Roshanak Houmanfar, Honglei Xie, Jiaxi Liang, Xichen She, and D. B. Emerson. 2021. "On Oversampling Imbalanced Data with Deep Conditional Generative Models." *Expert Systems with Applications* 169 (May): 114463. https://doi.org/10.1016/j.eswa.2020.114463.

Fisher, Ronald. 1935. *The Design of Experiments*. London: Oliver and Boyd.

Gm, Harshvardhan, Mahendra Kumar Gourisaria, Manjusha Pandey, and Siddharth Swarup Rautaray. 2020. "A Comprehensive Survey and Analysis of Generative Models in Machine Learning." *Computer Science Review* 38 (November): 100285. https://doi.org/10.1016/j.cosrev.2020.100285.

Goldberg, David E. 1989. *Genetic Algorithms in Search, Optimization, and Machine Learning*. New York, NY, USA: Addison-Wesley.

Golub, T. R., D. K. Slonim, P. Tamayo, C. Huard, M. Gaasenbeek, J. P. Mesirov, H. Coller, et al. 1999. "Molecular Classification of Cancer: Class Discovery and Class Prediction by Gene Expression Monitoring." *Science* 286 (5439): 531–37. https://doi.org/10.1126/science.286.5439.531.

Hastie, T, R Tibshirani, and J Friedman. 2009. *The Element of Statistical Learning: Data Mining, Inference, and Prediction*. Second Edition. Springer.

Huang, Chin-Wei, Shawn Tan, Alexandre Lacoste, and Aaron Courville. 2018. "Improving Explorability in Variational Inference with Annealed Variational Objectives." arXiv.org. September 6, 2018. https://arxiv.org/abs/1809.01818v3.

Isabelle Guyon, Steve Gunn. 2004. "Gisette." UCI Machine Learning Repository. https://doi.org/10.24432/C5HP5B.

Jiao, Ruwang, Bach Hoai Nguyen, Bing Xue, and Mengjie Zhang. 2023. "A Survey on Evolutionary Multiobjective Feature Selection in Classification: Approaches, Applications, and Challenges." *IEEE Transactions on Evolutionary Computation*, 1–1. https://doi.org/10.1109/TEVC.2023.3292527.

Khmaissia, Fadoua, and Hichem Frigui. 2023. "Confidence-Guided Data Augmentation for Improved Semi-Supervised Training." February 21, 2023. http://arxiv.org/abs/2209.08174.

Kingma, Diederik P., and Max Welling. 2019. "An Introduction to Variational Autoencoders." *Foundations and Trends® in Machine Learning* 12 (4): 307–92. https://doi.org/10.1561/2200000056.

Kwarciak, Kamil, and Marek Wodzinski. 2023. "Deep Generative Networks for Heterogeneous Augmentation of Cranial Defects." August 9, 2023. http://arxiv.org/abs/2308.04883.

Latif, Siddique, Rajib Rana, Junaid Qadir, and Julien Epps. 2020. "Variational Autoencoders for Learning Latent Representations of Speech Emotion: A Preliminary Study." July 27, 2020. https://doi.org/10.48550/arXiv.1712.08708.

Leelarathna, Navindu, Andrei Margeloiu, Mateja Jamnik, and Nikola Simidjievski. 2023. "Enhancing Representation Learning on High-Dimensional, Small-Size Tabular Data: A Divide and Conquer Method with Ensembled VAEs." June 27, 2023. http://arxiv.org/abs/2306.15661.

Li, Chenghao, and Chaoning Zhang. 2023. "When ChatGPT for Computer Vision Will Come? From 2d to 3d." 2023. https://doi.org/10.48550/ARXIV.2305.06133.

Liu, Qi, Miltiadis Allamanis, Marc Brockschmidt, and Alexander Gaunt. 2018. "Constrained Graph Variational Autoencoders for Molecule Design." In *Advances in Neural Information Processing Systems*. Vol. 31. Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2018/hash/b8a03c5c15fcfa8dae0b03351eb1742f-Abstract.html.

Ramaswamy, Sridhar, Pablo Tamayo, Ryan Rifkin, Sayan Mukherjee, Chen-Hsiang Yeang, Michael Angelo, Christine Ladd, et al. 2001. "Multiclass Cancer Diagnosis Using Tumor Gene Expression Signatures." *Proceedings of the National Academy of Sciences* 98 (26): 15149–54. https://doi.org/10.1073/pnas.211566398.

Ramchandran, Siddharth, Gleb Tikhonov, Otto Lönnroth, Pekka Tiikkainen, and Harri Lähdesmäki. 2022. "Learning Conditional Variational Autoencoders with Missing Covariates." March 2, 2022. http://arxiv.org/abs/2203.01218.

Roberts, Adam, Jesse Engel, Colin Raffel, Curtis Hawthorne, and Douglas Eck. 2019. "A Hierarchical Latent Vector Model for Learning Long-Term Structure in Music." November 11, 2019. http://arxiv.org/abs/1803.05428.

Solorio-Fernández, Saúl, J. Ariel Carrasco-Ochoa, and José Fco. Martínez-Trinidad. 2020. "A Review of Unsupervised Feature Selection Methods." *Artificial Intelligence Review* 53 (2): 907–48. https://doi.org/10.1007/s10462-019-09682-y.

Torre, Jordi de la. 2023. "Autocodificadores Variacionales (VAE) Fundamentos Te\'oricos y Aplicaciones." February 18, 2023. https://doi.org/10.48550/arXiv.2302.09363.

Vahdat, Arash, and Jan Kautz. 2020. "NVAE: A Deep Hierarchical Variational Autoencoder." In *Advances in Neural Information Processing Systems*, 33:19667–79. Curran Associates, Inc. https://proceedings.neurips.cc/paper/2020/hash/e3b21256183cf7c2c7a66be163579d37-Abstract.html.

Vie, Aymeric, Alissa M. Kleinnijenhuis, and Doyne J. Farmer. 2021. "Qualities, Challenges and Future of Genetic Algorithms: A Literature Review." September 13,

2021. http://arxiv.org/abs/2011.05277.

Vignolo, Leandro D., and Matias F. Gerard. 2017. "Evolutionary Local Improvement on Genetic Algorithms for Feature Selection." In *2017 XLIII Latin American Computer Conference (CLEI)*, 1–8. Cordoba: IEEE. https://doi.org/10.1109/CLEI.2017.8226467.

Zhang, Rui, Feiping Nie, Xuelong Li, and Xian Wei. 2019. "Feature Selection with Multi-View Data: A Survey." *Information Fusion* 50 (October): 158–67. https://doi.org/10.1016/j.inffus.2018.11.019.

Zhang, Yuchi, Yongliang Wang, Liping Zhang, Zhiqiang Zhang, and Kun Gai. 2019. "Improve Diverse Text Generation by Self Labeling Conditional Variational Auto Encoder." March 26, 2019. https://doi.org/10.48550/arXiv.1903.10842.

Apéndice A

Algoritmo Genético, teoría y implementación

En este apéndice presentamos una descripción general de los *algoritmos genéticos (AGs)*, exponiendo brevemente sus componentes y describiendo la implementación realizada en nuestro trabajo.

Como mencionamos en el capítulo 4, los AGs son algoritmos de búsqueda y optimización que se basan en la evolución biológica para encontrar soluciones óptimas o cercanas a ellas. Su enfoque en la población de soluciones y la aplicación de operadores genéticos permite una exploración eficiente del espacio de búsqueda, lo que los convierte en una herramienta poderosa para resolver problemas complejos.

Para entender cómo funcionan los AGs, es necesario conocer sus componentes clave. Estos incluyen la codificación del espacio de búsqueda, la búsqueda por población de soluciones, la función de aptitud y los operadores estocásticos. A continuación, presentamos cada uno de estos componentes en detalle.

A.1. a) Codificación del espacio de búsqueda

Como señalamos, los AGs se distinguen de otros algoritmos por su capacidad para operar en un espacio codificado del problema, en lugar de operar directamente el espacio en su representación original. Esto sucede gracias a la transformación de las soluciones potenciales en cadenas de datos, comúnmente conocidas como **cromosomas**, que luego son objeto de transformación mediante operadores genéticos como la mutación y el cruce. La capacidad de los AGs para operar con estas representaciones codificadas determina su adaptabilidad y eficacia en una amplia gama de problemas de optimización.

La codificación adecuada del problema es un paso inicial clave para el correcto desempeño del algoritmo. La elección de la codificación depende de la naturaleza del problema y de las características de las soluciones que se buscan optimizar. Por ejemplo, en problemas de optimización combinatoria, dado que las soluciones pueden representarse como permutaciones, una opción intuitiva en términos de codificación es la secuencias de números enteros. Por otro lado, en problemas de optimización continua, como la optimización de funciones matemáticas, las soluciones pueden representarse como vectores de números reales, lo que sugiere una codificación real-valuada. Así, la elección de la codificación adecuada en principio no tiene una respuesta única, antes bien admite múltiples alternativas confiriendo flexibilidad al diseño del AG.

Dada la importancia que tiene la codificación, es fácil advertir que así como una elección adecuada de la estrategia de codificación puede facilitar la convergencia del AG hacia soluciones óptimas, una elección inadecuada puede tener consecuencias negativas en su desempeño. En efecto, una codificación inapropiada puede llevar a una exploración ineficaz del espacio de soluciones o incluso a la generación de soluciones inviables. Así, una codificación que no preserve la viabilidad de las soluciones durante la evolución, puede resultar en la convergencia prematura del AG hacia soluciones subóptimas.

La traducción entre la representación interna codificada (genotipo) y la solución en el contexto del problema (fenotipo) es un componente importante del diseño de los AGs [1]. Este mapeo no solo permite interpretar las soluciones generadas por el algoritmo, sino que también influye en la eficacia de los operadores genéticos. Ello así, por cuanto los operadores genéticos actúan directamente sobre la representación codificada, lo que puede afectar la exploración del espacio de soluciones y la convergencia del AG.

Una de las principales ventajas de operar en un espacio codificado del problema radica en la posibilidad de aplicar operadores genéticos de manera eficiente, lo que permite una exploración exhaustiva del espacio de soluciones. En efecto, los operadores genéticos -que veremos en breve- son diseñados específicamente para actuar directamente sobre la representación codificada, generando nuevas soluciones de manera efectiva.

Un proceso típico de codificación y decodificación en un AG incluye los siguientes pasos:

1. **Espacio Original**: Representación directa del problema, por ejemplo, valores continuos o categóricos.
2. **Codificación**: Traducción del espacio original a una forma binaria o simbólica.
3. **Operadores Genéticos**: Aplicación de mutación, cruce y selección en la representación codificada.
4. **Decodificación**: Traducción inversa de la solución codificada al espacio original para evaluación.

En el caso de nuestra investigación, dada la alta dimensionalidad de los datos y la complejidad de los modelos, la codificación adecuada de las soluciones fue un proceso

[1] El **genotipo** se refiere a la representación interna de una solución en el AG. Es la "cromosoma" o la estructura de datos que codifica la información genética de un individuo. En la mayoría de los casos, el genotipo se representa como una cadena de bits (0 y 1), pero también puede ser una cadena de números, caracteres, o cualquier otra estructura adecuada dependiendo del problema. Por otro lado, el **fenotipo** es la manifestación externa o la interpretación del genotipo en el contexto del problema. Es la forma en que se evalúa la solución codificada por el genotipo. El fenotipo corresponde a la solución real en el espacio de búsqueda y es lo que se evalúa mediante la función de aptitud para determinar la calidad de un individuo. Por ejemplo, en un problema de selección de características donde cada característica puede ser incluida o excluida. El genotipo podría ser una cadena binaria donde cada bit representa si una característica está seleccionada (1) o no (0). Genotipo: `1100101`. Aquí, el genotipo representa la selección de ciertas características en un conjunto de datos. Este genotipo incluye las características 1, 2, 4, y 7, mientras que excluye las características 3, 5, y 6. El **fenotipo** es la manifestación externa o la interpretación del genotipo en el contexto del problema. Es la forma en que se evalúa la solución codificada por el genotipo. El fenotipo corresponde a la solución real en el espacio de búsqueda y es lo que se evalúa mediante la función de aptitud para determinar la calidad de un individuo. Siguiendo el ejemplo del genotipo `1100101`, el fenotipo sería la selección efectiva de características en un conjunto de datos. Si tenemos `7` características disponibles, el fenotipo se traduciría en el subconjunto de características seleccionadas por el genotipo. Por ejemplo: Características Disponibles: `[X1, X2, X3, X4, X5, X6, X7]`, Fenotipo: `[X1, X2, X4, X7]` En este caso, el fenotipo es el subconjunto de datos que incluye solo las características seleccionadas (X1, X2, X4, X7). Este subconjunto se utilizará para entrenar un modelo, y su desempeño será evaluado para determinar la aptitud del individuo.

fundamental para garantizar que los AGs puedan encontrar soluciones óptimas o cercanas al óptimo en tiempo razonable.

A.2. b) Búsqueda por población de soluciones

Una de las características distintivas de los AGs es su enfoque en la evaluación de una **población** de soluciones en cada iteración, en lugar de centrarse en una única solución. Esta población de soluciones, también conocida como población de **individuos**, permite a los AGs explorar simultáneamente múltiples regiones del espacio de búsqueda, aumentando así la probabilidad de encontrar soluciones óptimas o cercanas al óptimo.

Como vimos en el ejemplo de más arriba, la población inicial regularmente se genera de manera aleatoria, y cada individuo dentro de esta población representa una solución potencial al problema. A lo largo de las generaciones, los AGs aplican operadores genéticos como selección, cruce y mutación para producir nuevas generaciones de individuos, mejorando iterativamente la calidad de las soluciones.

Un esquema general del proceso de búsqueda por población en un AG se presenta a continuación:

Esquema del proceso de búsqueda por población

1. **Población Inicial**: Generación aleatoria de un conjunto de individuos que representan soluciones potenciales.
2. **Evaluación de Población**: Cada individuo es evaluado según una función de aptitud para determinar su calidad.
3. **Operadores Genéticos**:
   - **Selección**: Elegir individuos más aptos para reproducirse.
   - **Cruce (Crossover)**: Combinar partes de dos individuos para crear uno nuevo.
   - **Mutación**: Alterar aleatoriamente un individuo para introducir variabilidad.
4. **Nueva Generación**: Creación de una nueva población basada en los individuos más aptos.
5. **Iteración**: Repetición del proceso a través de múltiples generaciones hasta alcanzar un criterio de terminación.

La diversidad genética dentro de la población es fundamental para la eficacia de los AGs, ya que permite a los algoritmos explorar de manera más exhaustiva el espacio de características y evitar la convergencia prematura hacia soluciones subóptimas. En efecto, consideremos una población homogénea donde todos los individuos son idénticos. En este caso, la capacidad del AG para explorar nuevas regiones del espacio de búsqueda se ve severamente limitada, lo que puede resultar en una convergencia temprana hacia soluciones subóptimas. Por el contrario, una población diversa, donde cada individuo representa una solución única, permite al AG explorar una variedad de soluciones y adaptarse a las condiciones cambiantes del problema.

A modo de ejemplo consideremos estas dos población de 5 individuos codificados como sigue:

Población A, con 5 individuos de longitud 5, de alta diversidad:

- Individuo 1: `11001`

- Individuo 2: `10110`
- Individuo 3: `01101`
- Individuo 4: `11100`
- Individuo 5: `00011`

Población B, con 5 individuos de longitud 5, de baja diversidad:

- Individuo 1: `11111`
- Individuo 2: `11111`
- Individuo 3: `11011`
- Individuo 4: `11010`
- Individuo 5: `11010`

Como podemos advertir en este ejemplo, cada individuo representa una solución potencial al problema, donde cada bit en la cadena codificada corresponde a una característica que puede ser seleccionada o excluida. En estas poblaciónes los individuos de A son distintos entre sí, lo que permite al AG explorar una variedad de soluciones y adaptarse a las condiciones cambiantes del problema, mientras que los individuos de B son idénticos, lo que limita la capacidad del AG para explorar nuevas regiones del espacio de búsqueda.

A.3. c) Función de aptitud y evaluación de soluciones

La función de aptitud es el núcleo que dirige el proceso evolutivo en los AGs, determinando qué soluciones sobreviven y se propagan a la siguiente generación. Su diseño y correcta implementación son esenciales para asegurar que el AG no solo converja hacia soluciones de alta calidad, sino que también lo haga de manera eficiente y efectiva, especialmente en problemas donde las evaluaciones de aptitud son costosas o complejas.

En el proceso evolutivo de los AGs, La función de aptitud se aplica al **fenotipo** de cada solución, es decir, a su manifestación en el contexto del problema a resolver, después de que el **genotipo** (la representación codificada de la solución) ha sido transformado. Esta evaluación cuantifica qué tan bien una solución potencial cumple con los objetivos del problema, asignándole un valor numérico que refleja su desempeño relativo en comparación con otras soluciones dentro de la población.

El diseño de la función de aptitud es un aspecto crítico del proceso de modelado en los AGs, ya que guía la dirección de la búsqueda evolutiva. Específicamente, la función de aptitud debe estar alineada con los objetivos del problema, reflejando correctamente las restricciones necesarias a satisfacer. En situaciones de optimización multiobjetivo, donde varios criterios deben ser optimizados simultáneamente, es común que funciones de aptitud individuales se combinen en una única métrica a través de técnicas como la suma ponderada de los valores de aptitud individuales. En el contexto de nuestra investigación, orientada a la selección de características, la función de aptitud combina la aptitud de un individuo en términos de precisión y el tamaño del conjunto de características seleccionadas (veremos un ejemplo en breve).

En línea con lo anterior, la evaluación precisa de las soluciones mediante la función de aptitud puede constituir un proceso sujeto a multiples restricciones. Aunque la asignación de valores de aptitud más bajos a soluciones subóptimas y más altos a soluciones superiores pueda parecer un criterio ineludible, en la práctica, este proceso requiere comunmente consideraciones adicionales. Por ejemplo, en problemas con

restricciones, una solución que se acerque significativamente al óptimo global pero que infrinja requerimientos esenciales del problema debería recibir una calificación de aptitud inferior a una solución factible aunque menos cercana al óptimo con el fin de orientar la búsqueda hacia soluciones viables. Con esa lógica, en la optimización multiobjetivo es necesario establecer criterios para ponderar la proximidad al óptimo, especialmente cuando los distintos objetivos compiten entre sí.

Otro aspecto importante en la función de aptitud es la minimización del número de evaluaciones necesarias para alcanzar el óptimo o una solución lo suficientemente cercana a este. En muchos casos, cada evaluación de aptitud puede ser costosa en términos de tiempo y recursos computacionales, especialmente cuando la evaluación implica la simulación de modelos complejos o el entrenamiento de algoritmos de aprendizaje automático. Por ello, reducir el número de evaluaciones de aptitud es fundamental para mejorar la eficiencia del AG, sin sacrificar la calidad de las soluciones generadas. Este aspecto fue particularmente relevante en nuestra investigación, donde la evaluación de la función de aptitud implicaba el entrenamiento y validación de modelos de clasificación en conjuntos de datos de alta dimensionalidad. La técnica de paralelización y la evaluación diferencial de las soluciones fueron estrategias clave para reducir el tiempo de ejecución, aunque demandó una infraestructura computacional adecuada (más sobre esto en el próximo capítulo).

A.4. d) Operadores estocásticos y *esquemas* genéticos

Como hemos señalado los AGs emplean **métodos probabilísticos de transición** conformados por **operadores estocásticos**, que introducen aleatoriedad en el proceso evolutivo. Esto determina que las transformaciones dentro de un AG no siguen un camino determinista hacia la solución óptima; en su lugar, cada generación de soluciones es producto de un proceso estocástico controlado.

Los **operadores genéticos** fundamentales en este proceso son la **selección**, el **cruce (crossover)** y la **mutación**. Los mismos son responsables de la generación de nuevas soluciones, e inciden directamente en la evolución de los patrones genéticos que los AG tienden a preservar y reproducir. Patrones que se conocen como **esquemas** (1989).

Según explica Goldberg, los **esquemas** son estructuras genéticas que se repiten en la población y que influyen en la evolución de los individuos. Estos esquemas pueden ser de **orden bajo** (pocos genes) o de **orden alto** (más genes), y de **longitud de definición baja** (pocos bits) o de **longitud de definición alta** (más bits). En su operatoria, los AGs tienden a favorecer los esquemas de orden bajo y longitud de definición baja que muestran un rendimiento mejor que la media. Este fenómeno, conocido como **Teorema del Esquema**, proporciona una base para entender cómo la selección y los operadores estocásticos actúan en conjunto para guiar la evolución hacia soluciones óptimas. Veamos esto en detalle.

El operador de **selección** opera identificando y preservando los esquemas con aptitudes superiores a la media de la población. En términos probabilísticos, los esquemas con mejor aptitud tienen una mayor probabilidad de ser seleccionados y reproducidos en la siguiente generación. Esta selección basada en aptitud es clave para mantener y amplificar características beneficiosas dentro de la población. Dicho esto, la selección por sí sola no es suficiente para garantizar la exploración global del espacio de búsqueda, de ahí la importancia del cruce y la mutación.

El operador de **cruce** permite la recombinación de material genético entre dos o más soluciones. En un AG, la función principal del cruce es preservar y mejorar las características exitosas encontradas en los padres, mientras introduce suficiente variación para explorar nuevas áreas del espacio de búsqueda. Por ejemplo, en la representación binaria, un cruce de un punto dividirá dos soluciones en una posición elegida aleatoriamente y combinará segmentos de ambas para crear nuevos individuos. Este proceso asegura la transmisión de esquemas de orden bajo y longitud de definición baja, mientras introduce nuevas combinaciones genéticas que pueden llevar a soluciones más adaptativas.

Un ejemplo de cruce de un punto entre dos soluciones binarias sería:

- Padre 1: `110010`
- Padre 2: `101101`
- Punto de Cruce: `3`
- Hijo 1: `110101`
- Hijo 2: `101010`

En este caso, el cruce de un punto en la posición 3 divide los padres en dos segmentos y combina los segmentos para generar dos nuevos individuos. Este proceso de cruce permite la recombinación de material genético entre los padres, preservando y mejorando las características exitosas encontradas en ellos.

El operador de **mutación**, por su parte, introduce cambios aleatorios en las soluciones existentes, actuando como un mecanismo de perturbación que permite al AG escapar de óptimos locales y explorar más exhaustivamente el espacio de soluciones. La mutación puede variar desde simples alteraciones de bits en cadenas binarias hasta ajustes en representaciones continuas mediante la adición de ruido gaussiano. En términos del Teorema del Esquema, la mutación afecta la probabilidad de preservación de esquemas, especialmente aquellos de mayor orden, pero es crucial para asegurar que el AG mantenga la capacidad de descubrir nuevas áreas del espacio de búsqueda.

Un ejemplo de mutación en una solución binaria sería:

- Solución Original: `110010`
- Posición de Mutación: `4`
- Solución Mutada: `110110`

A esta altura ha de ser evidente que la preservación de ciertos patrones genéticos de aptitud superior es fundamental para la evolución de la población en un AG. La teoría de los esquemas, que se basa en el concepto de esquemas genéticos, proporciona un marco formal para entender cómo los operadores genéticos actúan en conjunto para guiar la evolución hacia soluciones óptimas.

Goldberg nos presenta, en relación a este punto, lo que se conoce como la **Ecuación del Esquema**, que es una herramienta teórica que permite predecir la evolución de los esquemas en una población a lo largo de múltiples generaciones. Esta ecuación tiene en cuenta factores como la aptitud de los esquemas, la probabilidad de cruce y mutación, la longitud de definición y el orden de los esquemas, y proporciona una guía para entender cómo los esquemas se propagan y se mantienen en la población.

La Ecuación del Esquema predice el número esperado de copias de un esquema H en la próxima generación $t+1$, dado su número de copias en la generación actual t. Se expresa de la siguiente manera:

$$m(H, t+1) \geq m(H, t) \cdot \tfrac{f(H)}{\bar{f}} \cdot \left[1 - p_c \tfrac{\delta(H)}{l-1}\right] \cdot (1 - p_m)^{o(H)}$$

Donde: - $m(H,t)$ es el número de copias del esquema H en la generación t. - $f(H)$ es la aptitud promedio de los individuos que pertenecen al esquema H. - $\bar{f}$ es la aptitud promedio de la población total. - p_c es la probabilidad de cruce. - $\delta(H)$ es la longitud de definición del esquema H, que es la distancia entre el primer y el último gen fijo en el esquema. - l es la longitud del cromosoma. - p_m es la tasa de mutación. - $o(H)$ es el orden del esquema, es decir, el número de posiciones fijas en el esquema.

Consideremos un ejemplo con los siguientes parámetros:

- Longitud del cromosoma: $l = 6$
- Esquema $H = 1 * 0 * 1*$ (donde $*$ puede ser 0 o 1)
- Población actual tiene 100 individuos.
- $m(H,t) = 20$ (es decir, 20 individuos coinciden con el esquema H).
- Aptitud promedio de la población $\bar{f} = 15$.
- Aptitud promedio de los individuos que coinciden con el esquema H, $f(H) = 18$.
- Probabilidad de cruce $p_c = 0.7$.
- Tasa de mutación $p_m = 0.01$.
- Longitud de definición del esquema $\delta(H) = 4$ (dado que las posiciones fijas son 1, 3 y 5, la distancia entre las posiciones es 4).
- Orden del esquema $o(H) = 3$ (el número de posiciones fijas es 3).

Aplicando estos valores a la Ecuación del Esquema:

1. **Factor de Selección**:

   $\frac{f(H)}{\bar{f}} = \frac{18}{15} = 1.2$

   Esto indica que los individuos que coinciden con el esquema H tienen una aptitud superior a la media y, por lo tanto, es más probable que sean seleccionados.

2. **Probabilidad de Conservación ante el Cruce**:

   $1 - p_c \frac{\delta(H)}{l-1} = 1 - 0.7 \cdot \frac{4}{6-1} = 1 - 0.7 \cdot 0.8 = 1 - 0.56 = 0.44$

   Hay un 44 % de probabilidad de que el esquema H se conserve tras el cruce.

3. **Probabilidad de Conservación ante la Mutación**:

   $(1 - p_m)^{o(H)} = (1 - 0.01)^3 = 0.99^3 \approx 0.9703$

   El esquema H tiene aproximadamente un 97 % de probabilidad de no ser destruido por la mutación.

4. **Cálculo Final**:

   $m(H, t+1) \geq 20 \cdot 1.2 \cdot 0.44 \cdot 0.9703 \approx 20 \cdot 0.5127 = 10.254$

   Por lo tanto, en la próxima generación, se espera que haya al menos 10 copias del esquema H en la población.

Este cálculo muestra cómo el esquema H, que tiene una aptitud superior a la media y ciertas características de proximidad posicional (es decir, una longitud de definición baja), es favorecido en la reproducción y es probable que se mantenga en la población.

Con los elementos vistos hasta aquí podemos pasar, a continuación, a la implementación de un AG en Python.

A.5. Implementación de un Algoritmo Genético

En el siguiente fragmento de código presentamos las operaciones elementales de un AG, que como dice Goldberg son *extraordinariamente sencillas*. La secuencia de operaciones se inicializa con un conjunto de soluciones, denominado población. El ciclo iterativo principal del Algoritmo Genético genera nuevas soluciones candidatas descendientes mediante cruce y mutación hasta que la población esté completa. En cada iteración, los individuos son evaluados mediante una función de aptitud que mide su calidad en relación al problema a resolver (aquí, la función de aptitud es simplemente la suma de los valores de los genes, en contextos reales, esta función se ajusta a las necesidades del problema pudiendo ser una función de costo, una métrica de desempeño, entre otras). Los individuos más aptos son seleccionados para reproducirse, lo que implica la aplicación de operadores genéticos para generar nuevos individuos. Este proceso se repite a lo largo de múltiples generaciones, permitiendo que la población evolucione y se adapte a las condiciones del problema.

```
import random

# Parámetros del Algoritmo Genético
num_individuals = 5
chromosome_length = 10
num_generations = 10
mutation_rate = 0.1

# Inicializar la población con individuos aleatorios
individuals = [random.randint(0, 1) for _ in range(chromosome_length)]
population = [individuals for _ in range(num_individuals)]

# Ejecutar el Algoritmo Genético
for generation in range(num_generations):
    # Calcular aptitud
    fitness_values = [sum(ind) for ind in population]

    # Crear nueva población
    new_population = []
    while len(new_population) < num_individuals:
        # Selección de dos padres
        parent1 = random.choices(population, weights=fitness_values)[0]
        parent2 = random.choices(population, weights=fitness_values)[0]

        # Cruce de un punto
        point = random.randint(1, chromosome_length - 1)
        child1 = parent1[:point] + parent2[point:]
        child2 = parent2[:point] + parent1[point:]

        # Mutación
        for i in range(chromosome_length):
            if random.random() < mutation_rate:
                child1[i] = 1 - child1[i]
            if random.random() < mutation_rate:
```

```
                child2[i] = 1 - child2[i]

        new_population.append(child1)
        if len(new_population) < num_individuals:
            new_population.append(child2)

    # Reemplazar la población antigua con la nueva
    population = new_population

    # Mostrar la población actual y sus aptitudes
    print(f"Generación {generation + 1}:")
    for ind in population:
        print(f"Individuo: {ind}, Aptitud: {sum(ind)}")
    print()
```

En el caso de nuestro trabajo, se implementó un AG con la librería `DEAP` de Python, que puede ser adaptado para la selección de características en diferentes contextos.

A.6. Algoritmo Genético con la librería DEAP

```
import random
import numpy as np
from deap import base, creator, tools
from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import accuracy_score

# Parámetros del Algoritmo Genético
PROB_MUT = 0.1          # Probabilidad de mutación
PX = 0.75               # Probabilidad de cruce
GMAX = 15               # Número máximo de generaciones
POP_SIZE = 20           # Tamaño de la población

# Carga del conjunto de datos de ejemplo
data = load_breast_cancer()
X = data.data
y = data.target

# División del conjunto de datos en entrenamiento y prueba
Xtrain, Xtest, y_train, y_test = train_test_split(
    X,
    y,
    test_size=0.3,
    random_state=42
)

# Tamaños derivados
DAT_SIZE = Xtrain.shape[0]
```

```
IND_SIZE = Xtrain.shape[1]
PM = 1 / IND_SIZE    # Probabilidad de mutación por gen

# Definición de la función de fitness
def fitness(individual, Xtrain, Xtest, y_train, y_test):
    """Función de aptitud para evaluar la calidad de un individuo."""
    if not any(individual):
        return 0,  # Evita seleccionar individuos sin genes activos

    X_train = Xtrain[:, individual]
    X_test = Xtest[:, individual]

    model = MLPClassifier(hidden_layer_sizes=(5, 3),
        max_iter=1000,
        random_state=42
    )
    model.fit(X_train, y_train)

    predictions = model.predict(X_test)
    accuracy = accuracy_score(y_test, predictions)

    # Minimización del número de características seleccionadas
    n_genes = np.sum(individual)
    alpha = 0.5  # Ponderación entre precisión y número de genes

    return alpha * accuracy + (1 - alpha) * (1 - n_genes / IND_SIZE),

# Configuración del entorno evolutivo
creator.create("FitnessMax", base.Fitness, weights=(1.0,))
creator.create("Individual", list, fitness=creator.FitnessMax)

toolbox = base.Toolbox()
toolbox.register("attr_bool", lambda: random.random() < PM)
toolbox.register("individual", tools.initRepeat,
                    creator.Individual, toolbox.attr_bool, n=IND_SIZE)
toolbox.register("population", tools.initRepeat, list, toolbox.individual)

toolbox.register("mate", tools.cxTwoPoint)
toolbox.register("mutate", tools.mutFlipBit, indpb=PROB_MUT)
toolbox.register("select", tools.selTournament, tournsize=3)
toolbox.register("evaluate", fitness,
                    Xtrain=Xtrain,
                    Xtest=Xtest,
                    y_train=y_train,
                    y_test=y_test)

# Función principal del Algoritmo Genético
def main():
    # Inicialización de la población
```

```
    population = toolbox.population(n=POP_SIZE)

    # Evaluación inicial
    fitnesses = list(map(toolbox.evaluate, population))
    for ind, fit in zip(population, fitnesses):
        ind.fitness.values = fit

    # Bucle evolutivo
    for gen in range(GMAX):
        # Selección y reproducción
        offspring = toolbox.select(population, len(population))
        offspring = list(map(toolbox.clone, offspring))

        # Aplicación del cruce y mutación
        for child1, child2 in zip(offspring[::2], offspring[1::2]):
            if random.random() < PX:
                toolbox.mate(child1, child2)
                del child1.fitness.values
                del child2.fitness.values

        for mutant in offspring:
            if random.random() < PROB_MUT:
                toolbox.mutate(mutant)
                del mutant.fitness.values

        # Evaluación de los nuevos individuos
        invalid_ind = [ind for ind in offspring if not ind.fitness.valid]
        fitnesses = map(toolbox.evaluate, invalid_ind)
        for ind, fit in zip(invalid_ind, fitnesses):
            ind.fitness.values = fit

        # Reemplazo de la población
        population[:] = offspring

        # Recopilación de estadísticas
        fits = [ind.fitness.values[0] for ind in population]
        print(f"Gen:{gen + 1}-Mejor_fit:{max(fits):.4f}-Promedio:{np.mean(fits):.4f}")

    # Mejor individuo al finalizar
    best_ind = tools.selBest(population, 1)[0]
    print("\nMejor individuo encontrado: ", best_ind)
    print(f"Fitness: {best_ind.fitness.values[0]:.4f}")
    print(f"Número de características seleccionadas: {np.sum(best_ind)}")

if __name__ == "__main__":
    main()
```