Universidad Autónoma de Chiapas

Facultad de Ciencias en Física y Matemáticas

Implementación de IA y técnicas de simulación en física aplicada

T E S I S

QUE PARA OBTENER EL GRADO DE:

MAESTRO EN CIENCIAS FÍSICAS

PRESENTA:

NORBERTO RUBÉN TORRES CASTILLEJOS X170040

DIRECTOR DE TESIS:
DR. FILIBERTO HUEYOTL ZAHUANTITLA

CO- DIRECTOR DE TESIS:
DR. MARIO ALBERTO AGUIRRE LÓPEZ

TUXTLA GUTIÉRREZ, CHIAPAS; ABRIL DE 2025.

Tuxtla Gutiérrez, Chiapas
25 de Marzo de 2025
Oficio No. FCFM/0188/25

Dr. Filiberto Hueyotl Zahuantitla
Director de tesis.
Investigador por México SECIHTI.
Facultad de Ciencias en Física y Matemáticas Unach.
p r e s e n t e

Estimado Dr. Filiberto

Por este medio me permito informarle que, una vez efectuada la revisión de la tesis de nominada:

"***Implementación de IA y técnicas de simulación en física aplicada***"

Ha sido aceptada para sustentar el examen de grado de la Maestría en Ciencias Físicas del **Lic. Norberto Rubén Torres Castillejos** con matrícula escolar X170040

Por lo tanto, se Autoriza la Impresión de la tesis, en virtud de haber cumplido con los requisitos correspondientes.

Sin más por el momento quedo de Usted, enviándole un cordial saludo.

Atentam
"Por la conciencia de la **vir"**

DIRECCIÓN
FCFM

Dr. Orlando Díaz Hernández
Director

C. c. p. Dra. María del Rosario Soler Zapata, Secretaria Académica de la FCFM
Mtro. René Solís López.-, Encargado del Control Escolar Posgrado de la FCFM
Archivo.

Oficio FCFM/CIP/002/2025

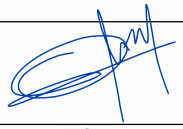
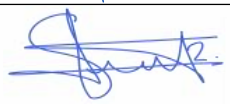
A quién corresponda

El que suscribe, Comité de Investigación y Posgrado de la Facultad de Ciencias en Física y Matemáticas de la universidad autónoma de Chiapas, hace constar que mediante minuta FCFM/MCF/002/2024 de los integrantes del NA de la Maestrla en Ciencias Físicas, que el Dr. Mario Alberto Aguirre López, investigador posdoctoral CONAHCYT en la FCFM UNACH, miembro del SNII nivel 1 fue autorizado para ser Co-director del estudiante Lic. Norberto Rubén Torres Castillejos

Sin más por el momento, reciba un afectuoso saludo.

ATENTAMENTE

"Por la conciencia de la necesidad de servir"

Grado académico y nombre completo	Cargo en el CIP	Firma
Dr. Orlando Díaz Hernández	Presidente	
Dr. Ariel Flores Rosas	Secretario	
Dr. Roberto Arceo Reyes	Representante de los Cuerpos Académicos	
Dr. Víctor Iván Ruiz Pérez	Vocal titular del Consejo Consultivo de Investigación y Posgrado	
Dr. Armando Mendoza Pérez	Representante de los Núcleos Académicos de los Programas de Posgrado	
Dr. Javier Sánchez Martínez	Representante de las profesoras y los profesores con reconocimiento vigente en el ámbito de la investigación científica, tecnológica, humanística y de innovación.	
Lic. Jesús Erick Pérez Pérez	Representación estudiantil de la Maestrla en Ciencias Físicas	

Código: FO-113-05-05
Revisión: 0

CARTA DE AUTORIZACIÓN PARA LA PUBLICACIÓN ELECTRÓNICA DE LA TESIS DE TÍTULO Y/O GRADO.

El alumno Norberto Rubén Torres Castillejos, autor de la tesis bajo el título de Implementación de IA y técnicas de simulación en física aplicada presentada y aprobada en el año 2025 como requisito para obtener el título o grado de Maestro en Ciencias Físicas, autorizo licencia a la Dirección del Sistema de Bibliotecas Universidad Autónoma de Chiapas (SIBI-UNACH), para que realice la difusión de la creación intelectual mencionada, con fines académicos para su consulta, reproducción parcial y/o total, citando la fuente, que contribuya a la divulgación del conocimiento humanístico, científico, tecnológico y de innovación que se produce en la Universidad, mediante la visibilidad de su contenido de la siguiente manera:

- Consulta del trabajo de título o de grado a través de la Biblioteca Digital de Tesis (BIDITE) del Sistema de Bibliotecas de la Universidad Autónoma de Chiapas (SIBI-UNACH) que incluye tesis de pregrado de todos los programas educativos de la Universidad, así como de los posgrados no registrados ni reconocidos en el Programa Nacional de Posgrados de Calidad del CONACYT.

- En el caso de tratarse de tesis de maestría y/o doctorado de programas educativos que sí se encuentren registrados y reconocidos en el Programa Nacional de Posgrados de Calidad (PNPC) del Consejo Nacional de Humanidades, Ciencias y Tecnologías (CONAHCYT), podrán consultarse en el Repositorio Institucional de la Universidad Autónoma de Chiapas (RIUNACH).

Tuxtla Gutiérrez, Chiapas; a los 09 días del mes de abril del año 2025.

Norberto Rubén Torres Castillejos

Resultados

Los resultados que se presentan en este trabajo de tesis derivaron en la creación de un repositorio de código libre, disponible en [github.com/fisicoerrante].

Introducción

La física es una disciplina fundamental para comprender el mundo que nos rodea, desde la dinámica de partículas subatómicas hasta la evolución de galaxias en el Universo. Muchos de estos procesos se han podido comprender y describir con un conjunto de herra-mientas analíticas y teorías bien fundamentadas. Sin embargo, muchos de los problemas físicos que enfrentamos son complicados y desafiantes de resolver de manera teórica, por lo que tradicionalmente se han abordado con métodos de resolución numérica.

En la última década, el desarrollo de la Inteligencia Artificial (AI), como el Machine Learning y el Deep Learning, han demostrado ser una herramienta poderosa para modelar y resolver problemas complejos, a menudo superando las limitaciones de enfoques convencionales. Particularmente, el método de Physics-Informed Neural Networks (PINNs) [1], brinda un área de oportunidad prometedora que aprovecha la física subyacente de un problema para entrenar una red neuronal, lo que permite obtener soluciones precisas y eficientes incluso en escenarios complejos. Así podemos obtener deducciones y soluciones a problemas que no solamente se basan en datos, si no que también respetan principios físicos fundamentales.

Las redes neuronales informadas por la física (PINN) se sitúan en la convergencia de las inteligencias artificiales basadas en datos y los modelos físicos tradicionales. Las PINNs emplean redes neuronales supervisadas que se basan en datos para adquirir conocimiento del modelo, al mismo tiempo que introducen ecuaciones que buscan mantener la consistencia con los principios físicos conocidos del sistema. Esta aproximación tiene la ventaja de utilizar datos para aprender el modelo y, al mismo tiempo, garantiza la coherencia con los fundamentos de la física, permitiendo además una extrapolación precisa más allá de los datos disponibles. En consecuencia, las PINNs pueden generar modelos más sólidos con una cantidad menor de datos.

Esta característica nos permite explorar problemas con pocos datos observacionales, o con pocas descripciones matemáticas. Por ejemplo, podemos explorar problemas de cosmo-

logía relacionadas a las sombras de agujeros negros supermasivos (SMBH Shadows), de las cuales se poseen aún pocas observaciones registradas por el telescopio horizonte de eventos (EHT) [2, 3]; o aproximar soluciones a las ecuaciones de Navier-Stokes [4]. Además, se pueden explorar problemas complejos, con soluciones teóricas bien conocidas, como la solución de las ecuaciones de campo de Einstein mediante PINN's como se propone en [5], construyendo y ampliando dicha red neuronal. Por su versatilidad, las PINNs pueden implementarse en diferentes áreas de la física, como en la física médica, para predecir las curvas de infección de COVID-19 y mejorar la toma de desiciones, y entender la dinámica de propagación de gotas en el estornudo de personas contagiadas [6].

A lo largo de la tesis, examinaremos ejemplos concretos de problemas físicos resueltos utilizando métodos de Machine Learning basados en datos y PINNs, demostrando su versatilidad en campos como la mecánica de fluidos, la relatividad general y la cosmología de agujeros negros. Además, evaluaremos el rendimiento de esta técnica en comparación con métodos tradicionales y exploraremos cómo la AI (y en particular el Machine Learning) pueden mejorar nuestra capacidad para comprender y abordar problemas físicos en la actualidad.

El contenido de esta tesis se organiza como sigue:

- En el Capítulo 1, mostraremos diversas técnicas de visualización y análisis explora-torio de datos, como la clusterización y la reducción de dimensionalidad, a través de ejemplos de agujeros negros y dinámica de fluidos.
- En el Capítulo 2, exploraremos los elementos esenciales del Machine Learning basado en datos, los tipos de aprendizaje, clasificadores y redes neuronales, para aplicarlos a ejemplos de datos astrofísicos.
- En el Capítulo 3, introducimos las PINNs, exploramos sus características y con diversas implementaciones exploramos sus capacidades para resolver problemas clásicos de la física.
- En el Capítulo 4, construimos una PINN enfocada en resolver las ecuaciones de campo de Einstein, desarrollamos el proceso de optimización y elección del modelo y discutimos los resultados.
- Finalmente, en el Capítulo 5, se presentan las conclusiones de la tesis y posibles líneas de investigación a futuro.

Índice general

Capítulo 1

Visualización de datos como herramienta en la simulación de fenómenos físicos

El análisis y la visualización de datos son herramientas fundamentales en la física moderna, ya que nos permiten interpretar y comprender fenómenos que involucran grandes cantidades de información. Este capítulo ofrece un repaso general de las metodologías y las técnicas más relevantes para el análisis y visualización de datos en el contexto de la física, mostrando cómo estas herramientas nos permiten transformar datos crudos en conocimiento físico valioso.

1.1. Análisis y visualización de datos

Aprender física a partir de los datos no es algo nuevo, recordando que, por ejemplo, J. Kepler dedujo sus tres leyes de movimiento planetario a partir de datos de las observaciones de T. Brahe. Sin embargo, la realidad actual es que la cantidad de datos de observaciones y experimentos de todo tipo, ha crecido y sigue creciendo de manera muy acelerada, por lo que se ha sido necesario el desarrollo de herramientas automáticas avanzadas para procesar esos datos, y así extraer información. En este capítulo se presentan algunas técnicas de análisis exploratorio y visualización de datos, y se muestra su aplica-ción en conjunto de datos de algunos fenómenos físicos.

Estandarización

En general, podemos visualizar un conjunto de datos a través de sus distribuciones y medidas estadísticas usuales, como la moda, mediana y desviación estándar. Las dis-tribuciones nos darán algunas pistas de cómo se comportan los datos, sin embargo, es necesario prepararlos o pre-procesarlos correctamente. En física es común, por ejemplo, que diferentes variables en un experimento tengan diferentes unidades, y más aún, que los valores de dichas variables tengan variaciones muy altas en sus órdenes de magnitud. Por ejemplo, en el caso de un agujero negro, donde la masa puede expresarse en el orden de millones de kilogramos, y el radio en apenas unos cuantos milímetros. Estas discrepancias pueden generar patrones o correlaciones donde en realidad no los hay. Una primera solu-ción a dicho problema es la estandarización de los datos. Aunque hay múltiples formas de estandarizar los datos, la más común es aplicar una transformación de tal manera que la distribución tenga media 0, y desviación estándar 1.

Correlaciones

Uno de los máximos objetivos de cualquier problema es poder describir un fenómeno a través de una ecuación (o conjunto de ecuaciones). Esta ecuación es una relación entre dos o más variables, por lo que en primera instancia, al analizar un conjunto de datos físicos, es buena idea comenzar por buscar correlaciones entre las variables.
La correlación, desde su definición estadística, indica la fuerza y la dirección de una re-lación lineal, así como la proporcionalidad entre dos variables estadísticas. Averiguar qué variables pudieran estar relacionadas en un fenómeno es esencial, pues frecuentemente nos enfrentaremos a problemas de dimensionalidad en los datos.

Las matrices de correlación, por ejemplo, nos permiten examinar las relaciones linea-les entre múltiples variables en un conjunto de datos. La idea detrás de las matrices de correlación es calcular los coeficientes de correlación entre pares de variables y representar estos coeficientes en forma de una matriz, forma numérica o con un mapa de colores. Los coeficientes de correlación, como el coeficiente de correlación de Pearson, proporcionan una medida cuantitativa de la fuerza y la dirección de la relación lineal entre dos varia-bles. Un valor cercano a 1 indica una correlación positiva fuerte, mientras que un valor cercano a -1 indica una correlación negativa fuerte. Un valor cercano a 0 indica una falta de correlación lineal, aunque debemos ser cautos pues la mayor parte de las veces, las variables se correlacionan pero no linealmente.

Pair Plots

Los pair plots, o gráficos de pares, son una de las primeras técnicas de visualización que podemos usar en el análisis exploratorio de datos. Este método es especialmente útil para analizar conjuntos de datos multidimensionales, lo que es usual en problemas físicos. Con este tipo de visualización buscamos comprender las relaciones entre múltiples variables de forma simultánea.

La idea detrás de los pair plots es sencilla: para cada par posible de variables en el conjunto de datos, se grafica un diagrama de dispersión de esas dos variables y se representa la distribución de cada variable en los márgenes o la diagonal de la gráfica. Esto nos permite identificar patrones visuales y posibles relaciones entre nuestras variables de interés.

En el contexto de la física, podemos usarlos para explorar la relación entre diferentes magnitudes físicas o para identificar posibles dependencias entre variables en experimentos o simulaciones. Además de eso, estas visualizaciones pueden ser útiles para identificar posibles anomalías o valores atípicos en los datos, lo cual es crucial en el análisis de datos físicos donde la precisión y la consistencia son fundamentales.

Heat Maps

Los mapas de calor, o heat maps, son otra herramienta visual muy útil para representar la distribución de valores en dos dimensiones mediante colores.

La idea detrás de los mapas de calor es mostrar la intensidad de una variable, como la temperatura, la densidad, o cualquier otra medida, en diferentes regiones de un espacio bidimensional. Esto se logra asignando un color a cada punto de datos según su valor en una escala de colores predefinida. Cuanto mayor sea el valor de la variable en un punto específico, más intenso será el color asignado a ese punto en el mapa de calor.

En general, son un buen complemento para otro tipo de gráficas, pues nos permiten incluir una variable adicional a la visualización sin tener que incrementar el número de ejes. Además de su utilidad para visualizar la distribución espacial de variables físicas, los mapas de calor también pueden ser empleados para identificar patrones o anomalías en los datos.

1.2. Clusterización

La clusterización es una técnica fundamental en la visualización de datos; consiste en agrupar conjuntos de datos similares en grupos o clusters, donde la similitud se define en función de alguna métrica o medida de distancia. Esta técnica es especialmente útil

cuando se tienen conjuntos de datos complejos y extensos, ya que nos permite identificar patrones subyacentes y estructuras ocultas en los mismos. La clusterización puede usarse por ejemplo, para la clasificación de partículas subatómicas, tipos de planetas o agujeros negros.

Clusterización jerárquica

Esta técnica construye una jerarquía de clusters mediante la fusión o división iterativa de clusters basada en la distancia entre ellos. La representación dendrográfica es una herramienta común para visualizar los resultados del agrupamiento jerárquico, donde la altura de las fusiones en el dendrograma representa la distancia entre los clusters fusionados, ver Fig.1.1.
El proceso puede ser aglomerativo, donde cada punto comienza en su propio cluster y se fusionan en clusters más grandes, o divisivo, donde todos los puntos comienzan en un cluster y se dividen en clusters más pequeños. Los pasos básicos del clustering aglomerativo son los siguientes:

- Usando una métrica de proximidad en particular, se construye una matriz de disimilitud y todos los puntos de datos se representan visualmente en la parte inferior del dendrograma.
- En cada iteración, se identifican los dos clusters más similares y se fusionan en uno solo; luego, se eliminan sus filas y columnas en la matriz de distancias, se añade una nueva fila y columna para el cluster resultante, y se recalculan las distancias.
- Se itera este proceso hasta que se obtiene el clúster máximo final que contiene todos los objetos de datos en un solo grupo. Esto representaría el ápice de nuestro dendrograma y marcaría la finalización del proceso de clustering.

También se pueden usar otros métodos de vinculación para calcular la distancia entre clusters, como:

- Ward: Este método se basa en minimizar la varianza dentro de cada cluster. En cada paso, se eligen los dos clusters cuya combinación produce el menor incremento de la suma de cuadrados intra-cluster, con lo cual se generan grupos más compactos y homogéneos.
- Complete Linkage: Calcula la distancia entre dos clusters como la distancia máxima entre cualquier par de puntos (uno de cada cluster).

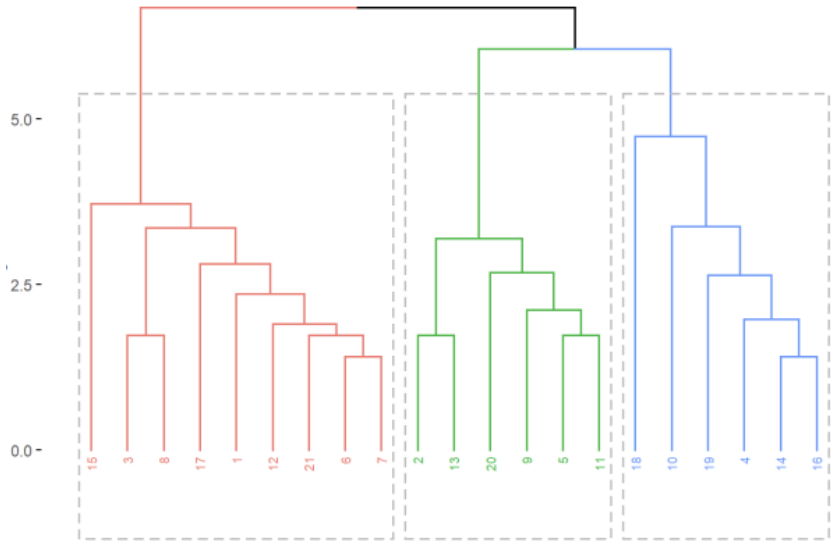


Figura 1.1: Ejemplo de un dendograma por clustering jerárquico, agrupando 21 observaciones en función de su similitud. Los colores (rojo, verde y azul) destacan tres clústeres principales, que se forman al cortar el dendrograma a una altura específica. La disposición jerárquica permite visualizar las relaciones entre las observaciones y cómo se fusionan progresivamente en grupos más grandes. Imagen tomada de bookdown.org/jsalinas/

- Average Linkage: Utiliza la distancia promedio entre todos los pares de puntos (uno de cada cluster).
- Single Linkage: Se basa en la distancia mínima entre cualquier par de puntos (uno de cada cluster).

La elección del método de vinculación afecta la forma y la cohesión de los clusters resul-tantes. Además, se deben tomar decisiones sobre el criterio de fusión, que puede basarse en la distancia entre clusters, el número máximo o mínimo de clusters, o un corte de altura en el dendrograma que visualiza el proceso. Finalmente, la evaluación de la estructura de los clusters resultantes se realiza utilizando métricas como el silhouette score, la disper-sión intra-cluster y la separación inter-cluster, que ayudan a determinar la coherencia y la calidad de la agrupación [7].

K-Means

El método K-Means busca particionar un conjunto de datos en K clusters donde cada observación pertenece al cluster con el centroide más cercano. Este algoritmo se basa en un proceso iterativo que comienza con la inicialización de K centroides, y luego, en

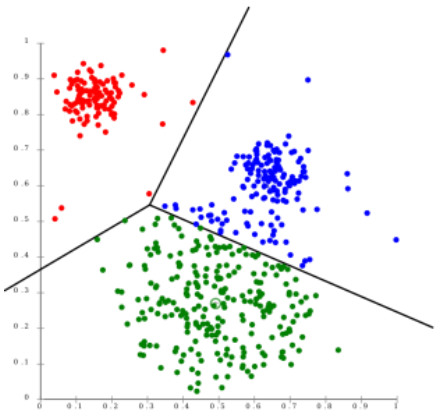


Figura 1.2: Ejemplo de conjunto bidimensional clusterizado por K-Means. Los puntos están agrupados en tres clusters principales, representados por colores rojo, azul y verde, donde cada cluster refleja la proximidad de los puntos a sus respectivos centroides. Las líneas rectas que separan los clusters representan los límites de decisión generados por el algoritmo, los cuales dividen el espacio en regiones equidistantes entre los centroides.

cada iteración, cada punto de datos se asigna al cluster cuyo centroide es el más cercano, seguido de la actualización de los centroides basada en la media de todos los puntos asignados a cada cluster. Este proceso continúa hasta que no hay cambios significativos en la asignación de puntos a clusters o se alcanza un criterio de convergencia predefinido. Una vez completado el proceso, tendremos una partición de los datos en clusters, y podríamos identificar patrones y relaciones en las observaciones.

Kernel K-Means

El método de Kernel K-Means es una variante del algoritmo de K-Means que utiliza el truco del kernel para operar en un espacio de características no lineales. A diferencia de K-Means estándar, que opera en un espacio euclidiano, en Kernel K-Means proyectamos los datos en un espacio de características de mayor dimensión mediante una función de kernel. Un kernel k es una función que calcula el producto punto de dos vectores transformados mediante cierta función . Si $x, z \in X$, siendo X el espacio de entrada, es decir, el que se forma con las variables de observación, entonces el kernel se define como:

$$k(x, z) = \langle \phi(x), \phi(z) \rangle \tag{1.1}$$

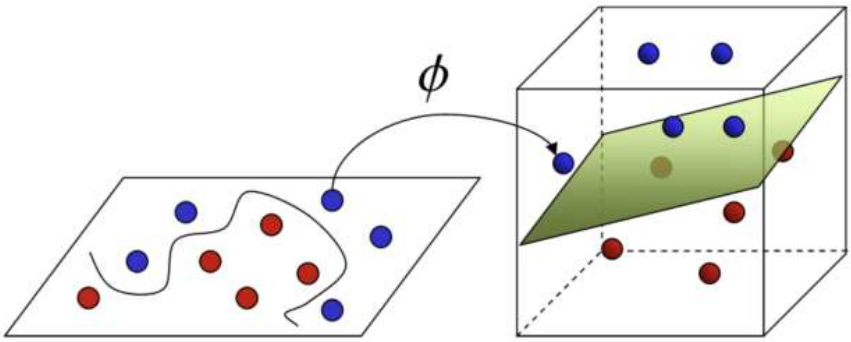


Figura 1.3: Ilustración de la tranformación de un conjunto de datos por Kernel K-Means. Datos que no son linealmente separables en un espacio bidimensional pueden transfor-marse a un espacio de mayor dimensionalidad mediante una función de proyección . En el espacio transformado, los datos ahora son linealmente separables, lo que permite la aplicación de un plano de decisión para distinguir los clústeres. Imagen de Santosch Sachin.

donde es un mapeo de X a un espacio de productos punto H , al que llamaremos espacio de características:

$$: x \in X \to (x) \in H \tag{1.2}$$

En Kernel K-Means, el proceso iterativo de asignación de puntos a clusters y actualización de centroides se realiza en el espacio de características transformado por el kernel. Esto significa que las operaciones de distancia se calculan implícitamente en el espacio de características de alta dimensión, lo que permite capturar estructuras no lineales de los datos. Otra diferencia importante es que el Kernel K-Means puede manejar datos que no son linealmente separables en el espacio original, ya que la transformación no lineal inducida por el kernel puede ayudar a separar los clusters en un espacio de características de mayor dimensión donde los datos son linealmente separables.

1.3. Reduccion de dimensionalidad

Gestionar la alta dimensionalidad es un desafío crucial cuando intentamos analizar datos. A menudo, las observaciones experimentales de un fenómeno tendrán más variables de las que podemos representar en un gráfico, incluso con algunas herramientas como las bubble plots o los heat maps. El overfitting o sobreajuste también se vuelve más probable, ya que los modelos pueden adaptarse demasiado a los datos de entrenamiento debido a la abundancia de características, lo que resulta en una generalización ineficiente para nuevos datos.

Uno de los problemas principales reside en la densidad del espacio de características, donde las observaciones tienden a estar más dispersas y distribuidas uniformemente a medida que aumenta la dimensionalidad, es decir que cuando un conjunto de datos crece en dimensión, se requiere de un mayor "volumen" de este espacio, para contener la misma cantidad de información (datos). Esto implica que preservar las estructuras de un fenómeno de alta dimensión es más complicado, y computacionalmente costoso. Este fenómeno, conocido como la maldición de la dimensionalidad [8], plantea numerosos obstáculos en la comprensión y extracción de información de conjuntos de datos con un gran número de variables.

Para abordar estos desafíos, se emplean métodos de reducción de dimensionalidad, los cua-les buscan transformar los datos a un espacio de menor dimensión mientras se conserva la información relevante. Estos métodos son fundamentales para simplificar la representación de datos complejos, reducir la complejidad computacional y facilitar la interpretación de los resultados obtenidos. En las siguientes secciones, exploraremos diversos métodos de reducción de dimensionalidad y su aplicación en el análisis y la comprensión de datos en diversas áreas, incluida la física.

PCA

El Análisis de Componentes Principales (comúnmente PCA, por sus siglas en inglés) es una técnica donde se busca identificar las combinaciones lineales de las variables originales que explican la mayor cantidad de variabilidad en los datos [9]. La clave de este método es maximizar la varianza, pues esta medida estadística es particularmente buena para preservar las estructuras en los datos. Esencialmente se trata de encontrar las direcciones de máxima varianza en un conjunto de datos y proyectar los datos originales en un nuevo espacio de menor dimensión definido por estas direcciones. El algoritmo puede describirse como sigue:

- Calculamos la matriz de covarianza de las variables originales para cuantificar las relaciones lineales entre ellas

- Con la matriz de covarianza, obtenemos las componentes principales, es decir, los eigenvectores y sus eigenvalores asociados.

- Seleccionamos los primeros k componentes principales tales que expliquen la mayor parte de la varianza en los datos. Normalmente, se establece un criterio de varianza acumulada para decidir el número k, como la varianza acumulada explicada o la regla del codo, en la que se grafica la cantidad de varianza explicada en función del número

de componentes, y se selecciona el punto donde la tasa de incremento comienza a disminuir significativamente, formando una especie de "codo" en la curva.

- Finalmente, los datos originales son proyectados en el espacio definido por las k componentes principales seleccionadas, lo que resulta en una representación de menor dimensión que conserva la mayor cantidad de información lineal posible.

Debemos tener en cuenta que este método busca encontrar patrones y estructuras en los datos, y que el espacio generado por las componentes principales no siempre tendrá una interpretación simple (o en nuestro caso, un significado físico intrínseco), pero nos permitirá identificar relaciones en los datos.

Kernel PCA

De manera similar al caso de clusterización por K-Means y Kernel K-Means, el Análisis de Componentes Principales Kernelizado (Kernel PCA) sobrepasa las limitaciones del PCA tradicional al permitir la reducción de dimensionalidad en conjuntos de datos no linealmente separables [10]. A diferencia del PCA estándar, en Kernel PCA se utiliza una función kernel para mapear los datos a un espacio de características de mayor dimensión, donde las relaciones entre los puntos pueden ser más fácilmente separables de manera lineal. Esto nos permite capturar de manera eficaz la estructura de los datos en un espacio no lineal. La mayor virtud de este método es que Kernel PCA puede ayudarnos a identificar estructuras no lineales intrínsecas en los datos que podrían pasarse por alto con PCA estándar.

t-SNE

La t-Distributed Stochastic Neighbor Embedding (t-SNE) es otra técnica de reducción de dimensionalidad no lineal. En comparación con PCA y Kernel PCA, que se centran en preservar la estructura lineal de los datos, t-SNE se enfoca en preservar las relaciones locales entre puntos en el espacio de alta dimensión, lo que lo hace especialmente efectivo para datos que exhiben estructuras no lineales o agrupamientos no convencionales [11].
El método t-SNE modela las similitudes entre pares de puntos en el espacio de alta dimensión utilizando distribuciones de probabilidad t-student.

1.4. Aplicación de los métodos de exploración

Aquí mostraremos algunos ejemplos de aplicación de los métodos de exploración a conjuntos de datos físicos. Es importante tener en cuenta que estos métodos exploratorios

no buscan resolver problemas de predicción, regresión o estimación por sí mismos, si no encontrar estructuras o patrones en los datos.

1.4.1. Sombras de agujeros negros

El radio de la sombra de un agujero negro, es descrito mediante la ecuación [12]

$$\theta_{sh}(z) = 3\sqrt{3}m\frac{(1+z)}{I(z)}\frac{H_0}{c}, \tag{1.3}$$

donde c es la velocidad de la luz, z es el redshift, H_0 la constante de Hubble, m la masa del agujero negro e $I(z)$ corresponde a la integral

$$I(z) = \int_0^z \left(\Omega_m(1+\tilde{z})^3 + \Omega_\Lambda\right)^{1/2} d\tilde{z}, \tag{1.4}$$

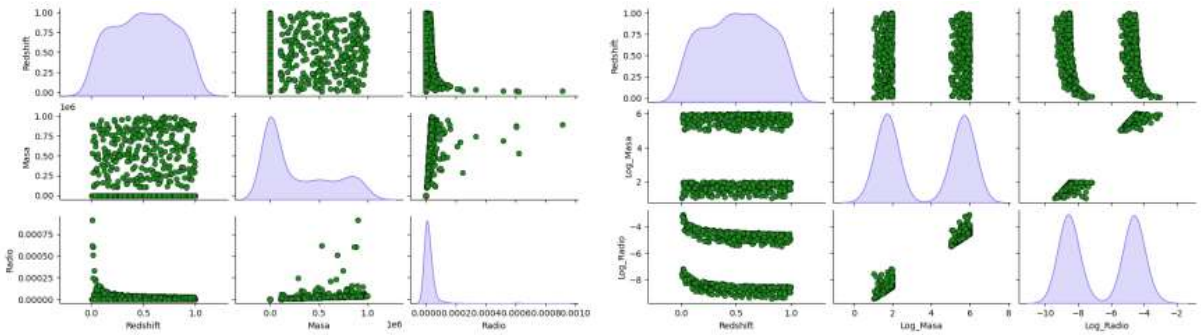
siendo Ω_m y Ω_Λ la cantidad de materia y energía oscura presentes en el Universo, respec-tivamente. En el régimen de bajo redshift se puede aproximar como:

$$\theta_{sh} \propto \frac{m}{z}. \tag{1.5}$$

Supongamos que tenemos observaciones de agujeros negros de distintas magnitudes de masa pero no tenemos un conocimiento explícito de las ecuaciones que describen el fenómeno y solamente tenemos datos observacionales de un interferómetro donde se observó la masa, el radio y el redshift de múltiples agujeros negros. Si sabemos que dentro de este catálogo, hay agujeros de masa estelar (BH) y agujeros negros de masa intermedia (IMBH), pero no han sido clasificados de antemano, ¿es posible separarlos, o identificarlos con los métodos de exploración?

Comencemos explorando los datos. Veamos por ejemplo, que la pairplot de la Fig. 1.4a no muestra claramente que exista algún tipo de relación entre las variables. Sin embargo, como es usual, podemos aplicar algunas transformaciones conocidas al conjunto completo para revelar algo más de información. Por ejemplo, en la Fig. 1.4b se muestra el pairplot correspondiente al logaritmo de los datos, y vemos que existe una separación clara en dos grupos de agujeros negros.

Además de la visualización con pairplots, podemos usar también mapas de color, o bubbleplots, para visualizar las tres variables en un espacio bidimensional. En la Fig. 1.5 se muestran los mapas de color de las tres variables y, en contraparte, la visualización en 3D del conjunto de datos. En la visualización 3D (ver Fig. 1.5a) se puede distinguir claramente que existen dos grupos dentro del conjunto de datos, sin embargo no es evidente cual es la característica o variable que más influye en la separación de ambos grupos. En comparación, tenemos las representaciones bidimensionales con mapas de color, donde se

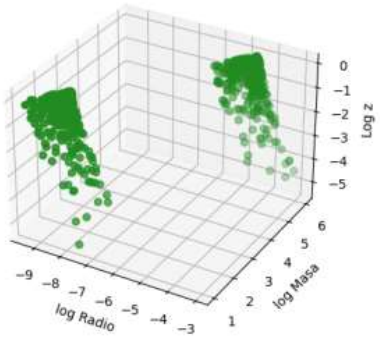


(a) (b)

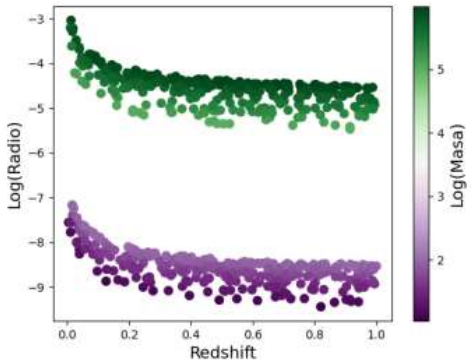
Figura 1.4: Pairplots de los datos de sombras de agujeros negros, (a) Variables originales, donde no se observan estructuras evidentes. (b) Variables bajo la transformación logarítmica donde se observan dos grupos de datos bien diferenciados.

eligen dos variables como ejes y la variable restante como la indicadora de color. En la Fig. 1.5c, por ejemplo, no aparece la separación de los conjuntos que teníamos, debido a que se sobreponen en una misma línea, y por lo tanto no es una representación útil. En las otras combinaciones si podemos identificar claramente dos grupos, y mientras que los colores de la Fig. 1.5d no nos proporcionan información muy relevante, ya que están distribuidas de manera muy similar en ambos grupos, en la Fig. 1.5b podemos identificar que, en cada grupo hay también un grupo de color dominante, es decir, en este caso la masa parece ser la variable que más información proporciona respecto a la diferencia entre un grupo y otro. Esto, de hecho, es muy interesante por que en el fondo, el conjunto de datos que tenemos, contiene información de dos tipos distintos de agujeros negros: los IMBH y los BH.

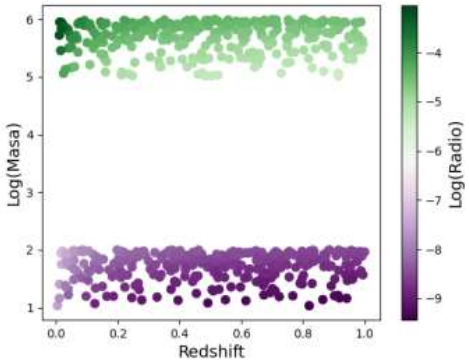
Aunque no conozcamos la Ecuación (1.3), ahora sabemos que existe una relación entre las variables. Esto se ve reforzado con los mapas de correlación (o heatmaps) que se ven en la Fig. 1.6, donde se mapea la correlación lineal entre las 3 variables. Ahí valores más cercanos al 1 (verdes) indican una correlación más fuerte. En la Fig. 1.6a, se usan los datos originales, y se puede observar que el radio y la masa, están más correlacionados entre sí que con el redshift. Sin embargo, dado que ya observamos antes que los datos se separan bajo una transformación logarítmica, entonces es interesante ver las correlaciones bajo esa transformación. En la Fig. 1.6b, se puede ver que la correlación entre el logaritmo de la masa y el logaritmo del radio es prácticamente 1 y que la correlación del redshift con las otras dos variables es menor. De este modo, podemos plantear (sin conocer completamente



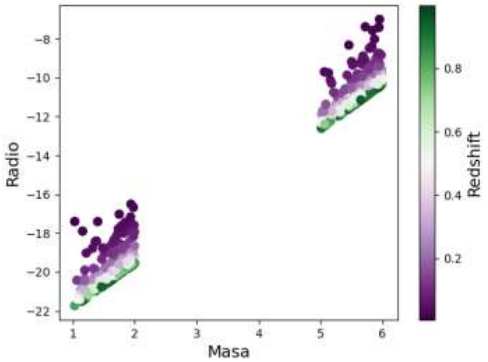
(a) Mapeo 3D de las variables



(b) Masa como variable de color

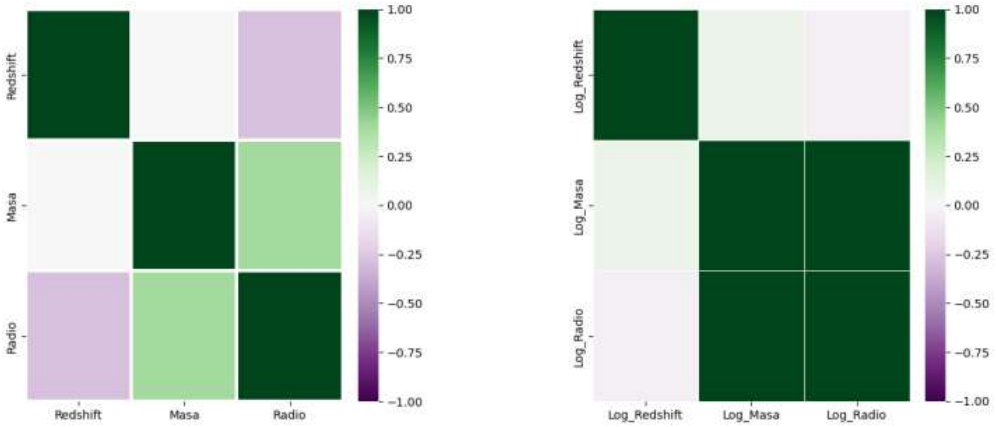


(c) Radio como variable de color



(d) Redshift como variable de color

Figura 1.5: Comparación de la visualización de las tres variables de los agujeros negros. En (a) se tiene una visualización en 3D. En (b), (c) y (d) se visualizan dos variables en los ejes y la tercer variable se usa para establecer el color de los datos. En esta visualización, el cuadro (d) tiene la peor interpretabilidad, es decir, el redshift no es un buen indicador para agrupar los datos, en cambio, la masa y el radio si se separan en dos grupos.



(a) (b)

Figura 1.6: Mapas de correlación de los datos de sombras de agujeros negros. (a) Variables originales, donde se observa correlación parcial entre la masa y el radio. (b) Variables bajo la transformación logarítmica, donde se observa una fuerte correlación lineal entre el logaritmo de la masa y el logaritmo del radio.

la física de fondo) que el radio puede escribirse como la ecuación logarítmica

$$\text{Log } \leftarrow = c_m \text{ Log } m + c_z \text{ Log } z + \text{Log } K \quad , \tag{1.6}$$

donde c_m y c_z son coeficientes desconocidos y K es alguna constante que contiene información sobre la configuración cosmológica del Universo. Usando esta ecuación podemos, por ejemplo, aplicar una regresión lineal simple, y de ella se obtiene que:

$$c_m = 0.99984313 \qquad \text{y} \qquad c_z = 0.97448343 \ .$$

Luego

$$\text{Log } \leftarrow = 0.99984313 \text{ Log } m \quad 0.9744843 \text{ Log } z + \text{Log } K \quad , \tag{1.7}$$

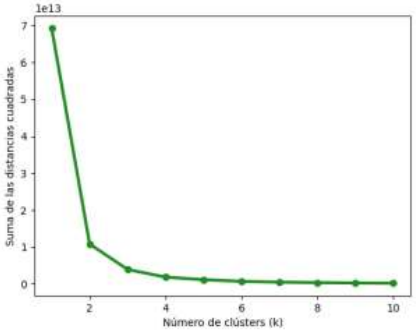
o de manera equivalente, usando propiedades de los logaritmos:

$$\text{Log } \leftarrow = \text{Log } \frac{\sqrt{0.99984313}}{m \ \sqrt{0.9744843}} K \frac{m}{z} \text{Log } K \frac{\downarrow}{z} \quad , \tag{1.8}$$

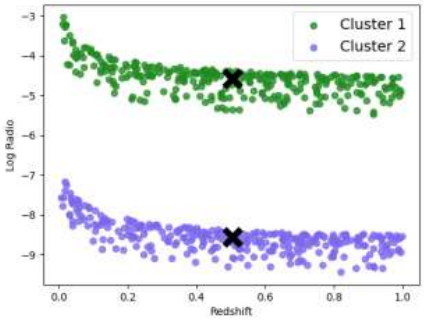
lo cual resulta interesante, pues si ahora tomamos el exponencial en ambos lados, obtenemos la Ec.(1.5):

$$\leftarrow = K \frac{m}{z} \ . \tag{1.9}$$

Finalmente, ahora que sabemos que tenemos dos grupos de agujeros negros, podemos aplicar, por ejemplo, los métodos de clusterización mencionados. En la Fig. 1.7 se muestra

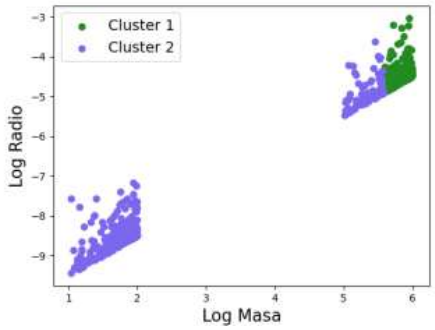


(a) Método del codo para elegir el número de clusters

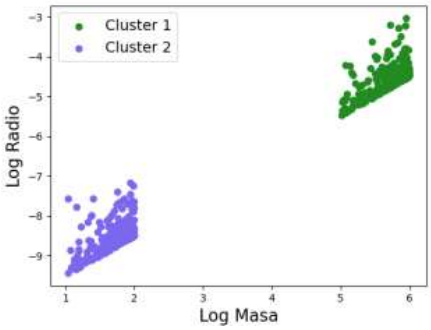


(b) Clusterización de los datos logarítmicos

Figura 1.7: Implementación del algoritmo K-Means para datos de agujeros negros transformados logarítmicamente.



(a) K-Means



(b) Kernel K-Means

Figura 1.8: Implementación de dos algoritmos de clusterización para los datos de agujeros negros sin transformar y posteriormente se visualiza con una transformación logarítmica. (a) Se aplica el método K-Means a los datos originales. Se observa que los clusters no están separados correctamente. (b) Se aplica el método de Kernel K-Means. Al aplicar el logaritmo para visualizar, se observa que los dos clusters están correctamente separados.

el resultado de implementar el método de K-Means para clusterizar los datos a los que previamente se les aplicó la transformación logarítmica. Se usó el método del codo para determinar el número adecuado de clusters minimizando la suma de las distancias cua-dradas en cada cluster hacia un centroide (Fig. 1.7a). Por convención, se elige el número donde la gráfica se dobla, es decir, a partir del cual esta suma no cambia significativa-mente. En este caso, el número adecuado es K=2, y se separan los grupos sin mayor complicación, como se ve en la Fig. 1.7b. Ahora, para ilustrar la diferencia entre K-Means y Kernel K-Means, aplicamos ambos algoritmos a los datos originales, es decir, sin aplicar

la transformación logarítmica. En esta configuración, la separación de los datos es menos evidente. Una vez aplicados los algoritmos, entonces para una visualización más cómoda, calculamos el logaritmo. Así podemos notar, de la Fig. 1.8a, que el método de K-Means confunde algunos de los puntos pertenecientes al cluster 1 y los asigna al cluster 2. Por otro lado, en la Fig. 1.8b, se tiene que el método de Kernel K-Means, que transforma los datos a un espacio de mayor dimensión y clusteriza en dicho espacio, sí puede separar correctamente los dos grupos de agujeros negros. Esto es importante, por que es el primer acercamiento a algoritmos más funcionales, como clasificadores o regresiones basadas en machine learning.

1.4.2. Dinámica de fluidos: Caracterización de gotas en un estornudo

Uno de los problemas más relevantes que hemos enfrentado como sociedad en épocas recientes, es la pandemia causada por el COVID-19. Desde hace años, se entiende que en este tipo de infecciones, el rango de contagio de un estornudo está relacionado directamente con el tamaño de las gotas, pero aun no existe un consenso sobre la distribución del tamaño de las mismas [13]. El objetivo de esta implementación es calcular la distribución del tamaño de las gotas de un estornudo humano, desde la perspectiva física y usando las herramientas de análisis de datos adecuadas en aquellos aspectos donde las capacidades del cómputo tradicional se vean rebasadas.
La mayoría de las mediciones del tamaño de gotas se centran en experimentos con líquidos tintados, o fotografía de alta velocidad. Sin embargo, se entiende que los estornudos son un fenómeno de flujos multifase y turbulentos, en el que ocurre un proceso de fragmentación previo a la aparición de gotas [13] por lo que resulta interesante estudiarlos desde la mecánica de fluidos. Partiendo de un módulo disponible en el código de Basilisk[1], se realiza una solución aproximada de las ecuaciones de Navier-Stokes de dos fases con tensión superficial, aplicadas para un chorro líquido que se inyecta en un ambiente de menor densidad a través de una boquilla cilíndrica, y posteriormente se realiza un conteo de gotas, de las que obtenemos el volumen y posición. Adecuar e implementar el código es un trabajo extenso y en el que usamos diversas técnicas de visualización como se detalla en el Apéndice A. Para este ejemplo, prestaremos atención a dos problemáticas particulares:

- No existe una caracterización completa de la velocidad en que se emite un estornudo humano.
- La formación de gotas de un estornudo ocurre tras un complejo proceso de fragmen-

[1] Disponible en http://basilisk.fr

tación, que aún es difícil de simular con precisión con los softwares actuales.

El primer punto abre la puerta a proponer nuevos modelos, mientras el segundo nos anticipa una posible escasez de gotas debido a la falta de fragmentación. Aquí simulamos tres modelos de inyección del fluido, analizando el estornudo a t = 100ms y considerando la función paramétrica:

$$\text{jet}(t; p) = u_0 + \frac{1}{N} \exp\left[\frac{\text{sen}(p\uparrow(t \quad p^2))}{p\uparrow(t \quad p^2)} + \frac{\text{sen}(p\uparrow(t \quad p^2))}{p\uparrow(t \quad p^2)}\right], \qquad (1.10)$$

en esta función, u_0 corresponde a la velocidad inicial del flujo, N es una constante de nor-

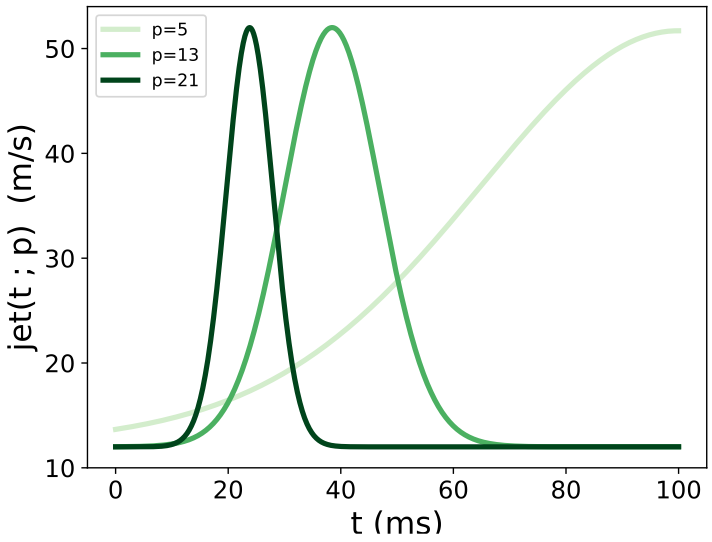


Figura 1.9: Familia de funciones de entrada de un estornudo a partir de la Ecuación 1.10.

malización que nos permite ajustar la velocidad máxima del estornudo y p es un parámetro numérico que controla la amplitud del pulso y su desplazamiento. Con ella podemos generar una familia de funciones, con propiedades que corresponden con la naturaleza del estornudo [14, 15], además de ser moldeables y tener parámetros interpretables. En la Fig. 1.9 se observa la familia elegida para las simulaciones, con tres valores distintos del parámetro p.

Luego de obtener los datos, hacemos un proceso de limpieza eliminando aquellas gotas que se sedimentan o se evaporan en el intervalo de estudio, según la descripción física dada en [16]. Ver Apéndice A para más detalles. En la Fig.1.10a se ve una proyección en dos dimensiones de las gotas obtenidas en la simulación, en ella, el tamaño de los puntos está en correspondencia con el diámetro. Exploramos con el método de clusterización de kernel K-Means, en búsqueda de estructuras en los datos. Para elegir el número de clusters

usamos el método de silhouette score, que mide qué tan similares son los puntos dentro de un cluster en comparación con otros clusters. El valor de este score oscila entre -1 y 1, y valores más altos indican mejores agrupamientos. Los resultados de la Fig.1.10b, nos permiten elegir k = 3 clusters para este ejemplo, y los resultados de la clusterización se ven en la Fig.1.10c.

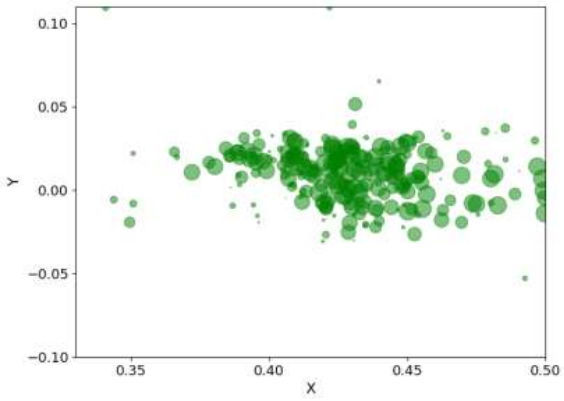
Sin embargo, la visualización de kernel K-Means tiene poca interpretabilidad, dado que la separación de los clusteres se hizo en un espacio de dimensión superior. Para tener una visualización más útil, aplicamos dos métodos de reducción de dimensiones, PCA y t-SNE. Los resultados de ambos métodos, presentados en la Fig.1.11, permiten entender mejor las diferencias entre estos clusteres. En particular, el cluster 3 parece ser un grupo de gotas de mucho mayor volumen, envuelto por capas de gotas más ligeras, siendo los clusteres 1 y 2. De estos dos métodos, PCA tiene una mayor interpretabilidad, pues pode-mos extraer los pesos que toman las variables originales para conformar las componentes principales, como se reporta en la Tabla 1.1.

	X	Y	Z	Varianza Explicada
PC1	0.045412	-0.47697	-0.738912	29.92 %
PC2	-0.680032	0.402422	0.083934	28.28 %
PC3	-0.632559	-0.684368	0.223253	23.42 %
PC4	-0.367919	0.373615	-0.630176	18.38 %

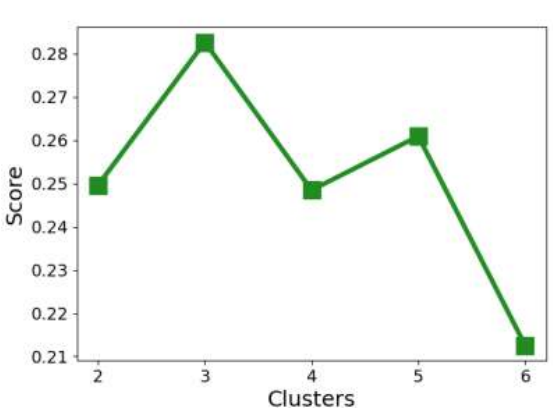
Cuadro 1.1: Cargas de las componentes principales y varianza explicada.

Después usamos el método de estimación de la densidad del kernel (KDE, por sus siglas en inglés), que es una técnica de visualización y análisis de datos no paramétrica que nos permite estimar la función de densidad de probabilidad (PDF) de una variable aleatoria. Esto resulta especialmente útil cuando se dispone de datos con posibles ausen-cias o con distribuciones desconocidas, como es el caso de los tamaños de gotas emitidas durante un estornudo humano. Este método no requiere asumir una forma específica para la distribución, lo cual es crucial en el caso de procesos tan complejos y poco caracteriza-dos como la formación de gotas en un estornudo.

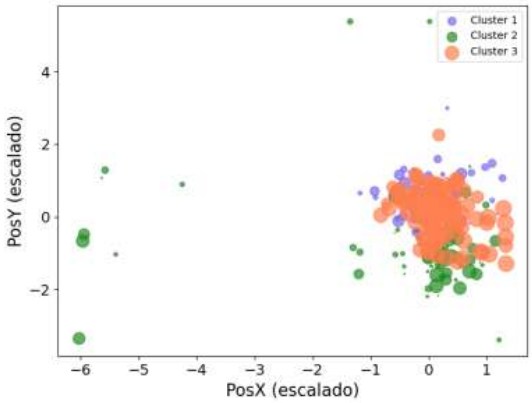
Una vez aplicado el método de KDE, podemos comparar los resultados de la simulación con resultados experimentales, tal como se ve en la Fig. 1.12. A la izquierda podemos observar las distribuciones obtenidas experimentalmente en [13], donde se observan ten-dencias unimodales y bimodales, con modas alrededor de los 100 y los 500 micrómetros y prácticamente ninguna gota por debajo de los 50 micrómetros. A la derecha, los resul-



(a)

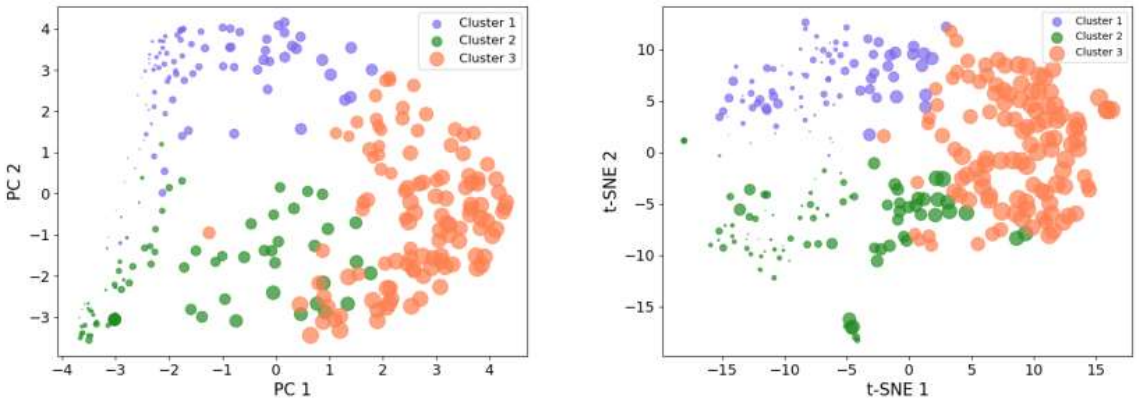


(b)



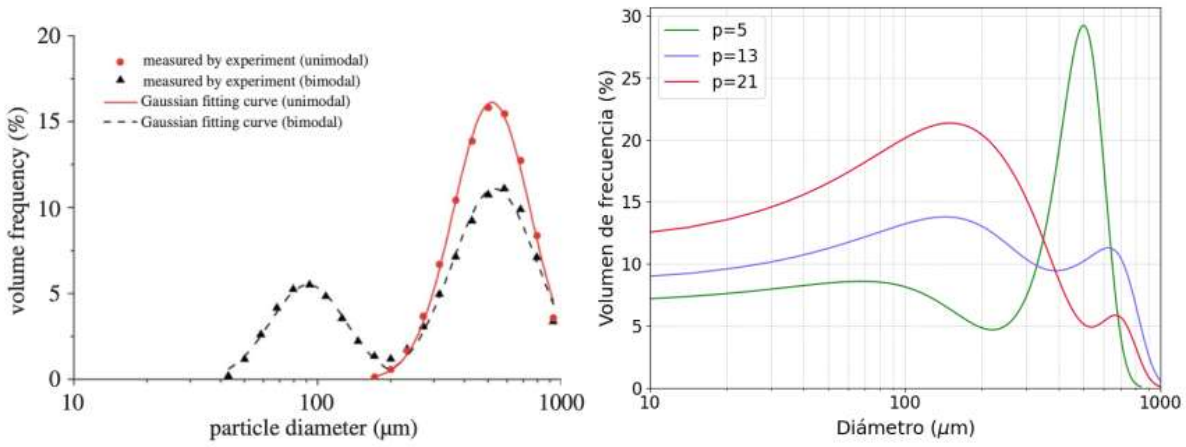
(c)

Figura 1.10: Visualización de las gotas del estornudo simulado para el caso p = 21. (a) Proyección en dos dimensiones del estornudo, el tamaño de los puntos está en función del diámetro de las gotas. (b) Método de silhouette score para elegir el número de clusters. El número más adecuado es 3, por tener un score más alto. (c) Resultados de la clusterización por kernel K-Means. Los ejes están escalados pues esto permite al método tener mejores resultados al reducir la varianza.



(a) (b)

Figura 1.11: Comparación de la distribución del tamaño de las gotas; (a) resultados experimentales y (b) resultado del KDE aplicado a la simulación.



(a) (b)

Figura 1.12: Comparación de la distribución del tamaño de las gotas; (a) resultados experimentales de [17] y (b) resultado del KDE aplicado a las simulaciones.

tados de las simulaciones muestran también una moda cerca de los 500 micrómetros; sin embargo, podemos ver una mayor cantidad de gotas pequeñas, incluso por debajo de los 10 micrómetros. Esto no necesariamente es incorrecto, pues físicamente en este intervalo es viable la existencia de gotas en el orden de nanómetros [16]. También es importante resaltar que la elección de la función de entrada impacta drásticamente en la evolución del fenómeno, pues es una de las características principales y menos entendidas de es-te fenómeno. Sin embargo, las simulaciones con las tres funciones de entrada predicen un número significativo de gotas con diámetros menores a 50µm, que no son detecta-das experimentalmente posiblemente a limitaciones técnicas, ya que no se conoce ningún mecanismo que impida su formación.

Capítulo 2

Machine Learning basado en Datos

La Inteligencia Artificial (AI por sus siglas en inglés) es un campo interdisciplinario que busca desarrollar sistemas capaces de realizar tareas que normalmente requieren inteligencia humana, como pueden ser el reconocimiento de patrones y estructuras, la toma de decisiones y el análisis de grandes volúmenes de datos. En la física, la IA ha mostrado un gran potencial en la modelación, simulación, análisis de datos experimentales y descubri-miento de patrones en fenómenos físicos complejos [18]. En este capítulo abordaremos el uso de la AI basada en datos, comenzando con una descripción de los tipos de aprendizaje y los principios fundamentales del Machine Learning (ML) y las Redes Neuronales (NN por sus siglas en inglés), y mostrando algunas implementaciones en problemas físicos.

2.1. Aprendizaje Supervisado y No Supervisado

Podemos pensar que el ML tiene dos tipos fundamentales de aprendizaje: el supervisado y el no supervisado. El aprendizaje supervisado es una técnica en la que un modelo se entrena utilizando ejemplos etiquetados, los cuales ya tienen respuestas conocidas que sirven para comparar nuestros resultados y ajustar el modelo en consecuencia [19]. En la física, este tipo de aprendizaje puede ser particularmente útil para predecir y modelar fenómenos donde tengamos observaciones previas. Podríamos por ejemplo, predecir trayectorias de partículas, o el tipo de ellas, clasificar estrellas u otros objetos astronómicos en imágenes de telescopios, o completar datos faltantes de datos experimentales.
Por otro lado, el aprendizaje no supervisado no requiere etiquetas previas en los datos. En este caso, el objetivo es encontrar estructuras y patrones inherentes en ellos, como agrupamientos o relaciones [20]. Podemos usar el aprendizaje no supervisado para cla-

sificar fases de la materia en sistemas complejos o para identificar patrones en datos de simulaciones, especialmente cuando no se tienen hipótesis claras sobre la estructura de los datos.

2.2. Redes Neuronales

En la programación tradicional, indicamos a la computadora exactamente qué, cómo y cuándo hacer algo, descomponiendo los problemas complejos en múltiples tareas pequeñas y precisas que puede ejecutar con facilidad. En cambio, en una red neuronal, no instruimos a la computadora sobre cómo resolver el problema de manera explícita; en su lugar, dejamos que ella aprenda a partir de datos observacionales, encontrando su propia solución al problema. Este proceso de aprendizaje se basa en el ajuste iterativo de parámetros de la red hasta que el modelo sea capaz de realizar predicciones precisas [21]. A grandes rasgos, podemos pensar en las redes neuronales como estructuras de transformaciones interconectadas [22], conocidas como neuronas, cuyo propósito es aproximar una función

$$f : R^n \to R^k , \tag{2.1}$$

donde n y k son el número de características o variables de entrada y salida de la red, respectivamente.

2.2.1. El Perceptrón y las Funciones de Activación

La unidad fundamental en una red neuronal es el perceptrón, también conocido como neurona, ver Fig 2.1. Una neurona toma un vector de entrada $\vec{x} \in R^n$ y lo transforma en un valor $a(z) \in R$, siendo $a(z)$ el resultado de una función de activación donde

$$z = \sum_{i=0}^{n} w_i x_i + b \tag{2.2}$$

representa una combinación lineal de pesos w_i y sesgos b aplicados a las entradas del vector $\vec{x}$. El objetivo de las funciones de activación es introducir no linealidad en el modelo, lo cual es esencial para que las redes neuronales puedan aproximar relaciones complejas en los datos. Aunque la primera en ser utilizada fue la función Heaviside, hoy existe una amplia variedad de ellas que facilitan la convergencia y mejoran el rendimiento de los modelos en distintos contextos, Ver Tabla 2.1.

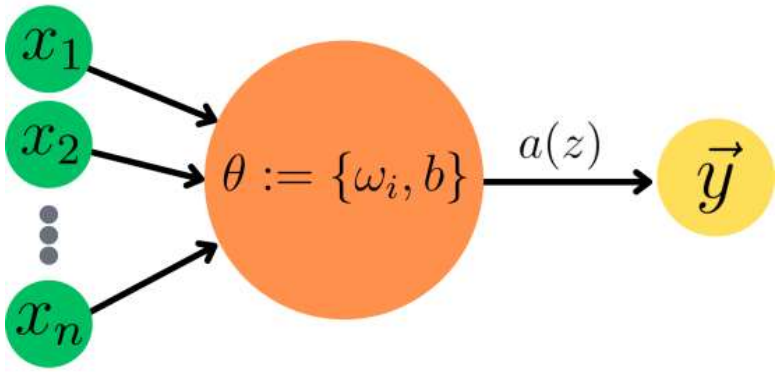


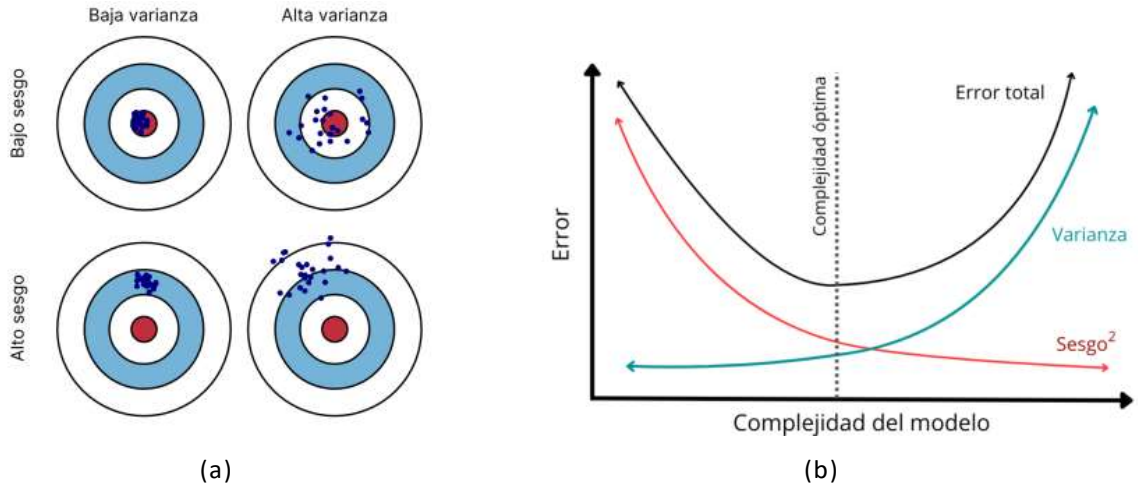
Figura 2.1: Diagrama del perceptrón. El perceptrón toma variables x_i y las transforma asignando pesos w_i y un sesgo b. Usa una función de activación a(z) para quitar la linealidad donde z está definida como en la Ec.(2.2), y con ello hace una predicción y para la variable de respuesta.

Función de Activación	a(z)
Sigmoide	$\frac{1}{1+e^{-z}}$
Log-Sigmoid	$\ln \frac{1}{1+e^{-z}}$
Tanh (Tangente hiperbólica)	$\frac{e^{z}-e^{-z}}{e^{z}+e^{-z}}$
ReLU (Rectified Linear Unit)	máx(0, z)
Softplus	$\ln(1+e^{z})$
Softmax	$\frac{e^{z_i}}{\sum_j e^{z_j}}$
Swish	$\frac{z}{1+e^{-z}}$

Cuadro 2.1: Funciones de activación comunes en redes neuronales [19].

2.2.2. Arquitectura y entrenamiento de las Redes Neuronales

A partir de la estructura y el funcionamiento del perceptrón, se puede construir la arquitectura general de las redes neuronales. Una red neuronal consiste en una capa de entrada, una o más capas ocultas y una capa de salida. Cada capa oculta aplica transformaciones lineales y no lineales a las entradas, permitiendo al modelo aprender patrones y estructuras complejas en los datos. El número de neuronas y capas ocultas se elige cuidadosamente para lograr un balance entre la precisión del modelo (bajo sesgo) y su



(a) (b)

Figura 2.2: Tradeo↵ del sesgo y la varianza: (a) interpretación visual del sesgo y la varianza, los puntos azules representan predicciones y el área roja del centro representa el valor verdadero. El caso ideal es cuando ambos, sesgo y varianza son bajos. (b) Evolución del MSE respecto a la complejidad de un modelo. El sesgo y la varianza se comportan de maneras opuestas, por lo que es necesario buscar una complejidad óptima.

capacidad de generalización (baja varianza). Este problema es a menudo conocido como el Bias-Variance Tradeo↵, ver Fig. 2.2[1].

Uno de los resultados teóricos más relevantes en el diseño de arquitecturas de redes neuronales es el llamado Teorema de Aproximación Universal. Este teorema, demostrado inicialmente para la función de activación sigmoide, afirma que una red neuronal de una sola capa oculta (con una cantidad suficiente de neuronas) es capaz de aproximar cualquier función continua en un espacio compacto, con un error arbitrariamente pequeño [23]. Esto significa que, en principio, redes simples pueden modelar una amplia variedad de funciones complejas, aunque en la práctica, agregar más capas suele ser necesario para mejorar la eficiencia y reducir la cantidad de neuronas necesarias. El Teorema de Aproximación Universal también establece una base teórica para el uso de redes neuronales en problemas de modelado y predicción en física, donde el comportamiento de los siste-mas frecuentemente es complejo y no lineal. Sin embargo, lograr una buena aproximación depende de la optimización efectiva de los parámetros de la red, lo que no es una tarea trivial.

Aunque existe una gran variedad de arquitecturas para las redes neuronales, cada una con sus ventajas en diferentes contextos, la arquitectura típica es el perceptrón multi-

[1] Imágenes de S. Fortmann Roe, traducidas al español. http://scott.fortmann-roe.com/docs/BiasVariance.html.

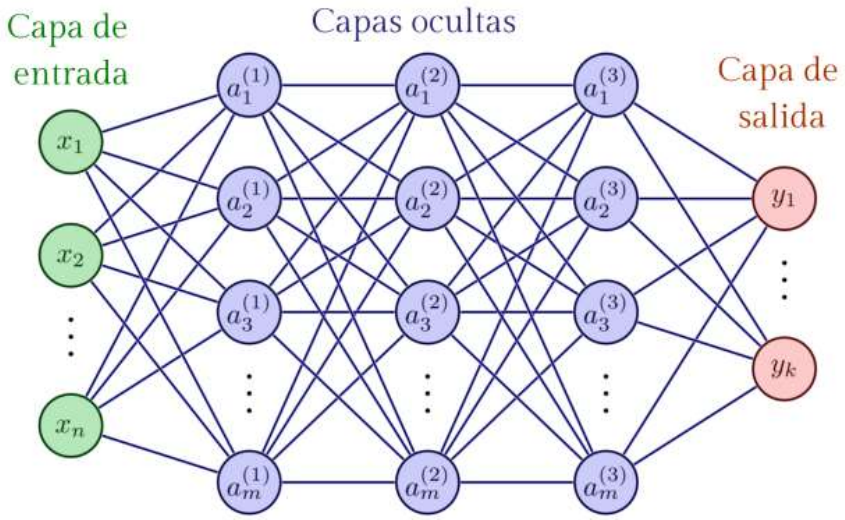


Figura 2.3: Arquitectura general de una red neuronal feed forward. En ellas, se toman variables de entrada x_i y la información fluye siempre hacia adelante a través de las neuronas $a_i^{(h)}$, hasta obtener un valor de predicción para las variables de salida y_j en la capa final.

capa. Dichas arquitecturas también son conocidas como Redes Neuronales Feed Forward (FFNN) y tienen una capa de entrada que toma n variables y las transforman a través de las neuronas en las capas ocultas, para llevarlas finalmente a la capa de salida con k datos que conforman un vector $\hat{y}_{nn} \in \mathbb{R}^k$, donde el sufijo nn se refiere a que es la salida de la red neuronal (neural network), como se ilustra en la Fig. 2.3.

Una vez que se obtienen los resultados, se busca que la "distancia" entre las predicciones y los valores reales alcance valores cercanos a cero. Una red neuronal aprende a minimizar la diferencia entre las predicciones y los valores observados, cuantificada mediante una función de costo o pérdida L. Esta minimización puede interpretarse como la búsqueda de un ajuste óptimo en un espacio de alta dimensionalidad. La función de costo puede elegirse según las necesidades de un modelo o problema, pero lo más común es el elegir el error cuadrático medio (MSE por sus siglas en inglés):

$$L_{MSE} = \frac{1}{N} \sum_{i=1}^{N} ||\hat{y}_i^{true} - \hat{y}_i^{nn}||^2 , \tag{2.3}$$

donde $\hat{y}_{true}$ y $\hat{y}_{nn}$ son los valores reales y predichos, respectivamente, que se evalúan en cada subíndice i y N es la cantidad de datos disponibles. En lo posterior, se omitirá escribir explícitamente el subíndice i para facilitar la lectura. Se busca entonces minimizar L_{MSE} a través de un método de optimización como el descenso por gradiente (GD), aunque en la actualidad es posible encontrar implementaciones con Descenso por Gradiente Esto-

castico (SGD[2]), Adaptive moment estimation (Adam[3]) y Root mean square propagation (RMSprop[4]) para actualizar los parámetros de manera que las predicciones se parezcan más a los datos reales.

Durante el entrenamiento, este proceso de actualización de parámetros se repite a lo largo de múltiples épocas, donde cada época implica un ciclo completo en el que la red neuronal procesa todo el conjunto de datos de entrenamiento. A medida que se avanza a través de las épocas, la red ajusta gradualmente sus parámetros ✓ en cada iteración:

$$✓_{i+1} = ✓_i \; ↲ r L \;, \tag{2.4}$$

donde ↲ es un hiperparámetro llamado learning rate que determina qué tan grande será el cambio en los valores de los parámetros y, por tanto, está fuertemente ligado a la convergencia del aprendizaje de una red. La elección adecuada del número de épocas es crucial, ya que un entrenamiento insuficiente puede llevar a un subajuste, donde el modelo no logra capturar los patrones fundamentales de los datos, resultando en un desempeño pobre tanto en el entrenamiento como en la prueba. Por otro lado, un exceso de épocas puede resultar en un sobreajuste, donde el modelo aprende demasiado bien los detalles, incluidos el ruido y los valores atípicos de los datos de entrenamiento, perdiendo su capacidad de generalizar y ofreciendo un desempeño deficiente en datos nuevos. Encontrar el balance adecuado permite construir modelos que sean robustos y efectivos en diferentes escenarios.

2.3. Clasificadores

La clasificación es una tarea fundamental en el aprendizaje automático, en la cual un modelo o algoritmo asigna etiquetas a nuevas observaciones en función de características previamente observadas. Los clasificadores son herramientas de modelado que buscan categorizar datos en grupos o clases distintas y son ampliamente utilizados en física para problemas que requieren la identificación o caracterización de elementos o eventos, tales como la clasificación de partículas subatómicas, de gotas, de estrellas o el análisis de datos experimentales.

En el Capítulo 1, se abordaron técnicas de clusterización, como el método jerárquico, el K-Means y el Kernel K-Means, los cuales son métodos de exploración de datos que permiten agrupar observaciones sin conocimiento previo de las etiquetas de clase. Mientras que la clusterización es un tipo de aprendizaje no supervisado que revela patrones en los datos sin clasificaciones explícitas, la clasificación es un tipo de aprendizaje supervisado

[2] https://www.tensorflow.org/api_docs/python/tf/keras/optimizers/SGD
[3] https://pytorch.org/docs/stable/generated/torch.optim.Adam.html
[4] https://keras.io/api/optimizers/rmsprop/

que asigna observaciones a clases predefinidas. Estos enfoques son complementarios en el aprendizaje automático, donde la clusterización puede servir como una etapa preliminar para la clasificación, proporcionando un análisis exploratorio de los datos que permite entender su estructura y características antes de etiquetarlos.

2.3.1. Tipos de clasificadores

Existen distintos tipos de clasificadores, que se diferencian principalmente por la cantidad de categorías en las que pueden dividir los datos. Los clasificadores binarios son los más simples, dividiendo los datos en dos posibles clases. Por otro lado, los clasificadores multiclase se emplean en problemas donde es necesario asignar cada observación a una de varias categorías posibles. Matemáticamente, el objetivo es asignar una observación representada por un vector de características $x \in R^n$ a una clase y, con $y \in \{0, 1\}$ para problemas binarios, o $y \in \{1, \ldots, K\}$ en problemas con K clases.

Clasificadores Binarios

En un clasificador binario, la función de decisión separa los datos en dos clases. Para los clasificadores lineales la función de decisión puede expresarse como

$$f(x) = \text{sign}(w^T x + b), \tag{2.5}$$

donde $w \in R^n$ es el vector de pesos, y $b \in R$ es el sesgo. En este caso, si $f(x) > 0$, la predicción es clase 1, y si $f(x) \leq 0$, la predicción es clase 0. La frontera de decisión se define por el hiperplano $w^T x + b = 0$.

Para clasificaciones probabilísticas, como en la regresión logística, la probabilidad de pertenecer a la clase 1 puede modelarse mediante la función sigmoide:

$$P(y = 1|x) = \quad (w^T x + b) = \frac{1}{1 + e^{\ (w^T x + b)}}, \tag{2.6}$$

donde es la función sigmoide y el umbral suele fijarse en 0.5 para tomar la decisión.

Clasificadores Multiclase

Para problemas multiclase, el objetivo es asignar una observación a 1 de K clases. Una opción común es la clasificación "One-vs-Rest", en la que se entrena un clasificador binario para cada clase $k \in \{1, \ldots, K\}$, donde cada clasificador $f_k(x)$ estima la probabilidad o puntuación de pertenecer a la clase k frente a todas las demás. La clase predicha es aquella con la puntuación más alta

$$\hat{y} = \arg\max_k f_k(x). \tag{2.7}$$

Otra estrategia es el uso de la función softmax, donde la probabilidad de cada clase k se calcula como:

$$P(y = k|x) = \frac{e^{w_k^T x}}{\sum_{j=1}^{K} e^{w_j^T x}}, \tag{2.8}$$

donde w_k es el vector de pesos correspondiente a la clase k. La clase con la probabilidad más alta es la predicción final:

$$\hat{y} = \arg\max_k P(y = k|x). \tag{2.9}$$

Finalmente, la efectividad de un clasificador no depende únicamente de su capacidad para separar las clases, sino también de cómo se mide su precisión a través de matrices de confusión o en función de métricas como la exactitud, precisión, recall, y el F1-score [24]. La selección de estas métricas varía según el problema y la importancia relativa de los falsos positivos o falsos negativos.

2.4. Aplicación de AI en datos físicos

Una vez que hemos abordado las generalidades del aprendizaje automático, es momento de aplicarlo a conjuntos de datos físicos. En esta sección se muestran algunos problemas representativos resueltos mediante estas técnicas.

2.4.1. Perceptrón para Supernovas

El objetivo de esta implementación es mostrar el papel del tradeo↵ en la interpretabilidad de un modelo. Usamos el catálogo Pantheon [25], que contiene 1048 observaciones de Super Novas Tipo Ia (SNIa), y particularmente nos concentramos en dos variables: el corrimiento al rojo (redshift) z y el módulo de distancia µ. Existe una relación cosmológica entre ambas variables, dada por:

$$\mu = m(z) \quad M, \tag{2.10}$$

donde m(z) y M son es la luminosidad relativa y absoluta de la SNIa, respectivamente. La luminosidad absoluta M, es una cantidad de interés en la cosmología, y frecuentemente se estudian maneras de mejorar su precisión. En nuestro caso, construimos un perceptrón, que toma como característica de entrada el redshift, lo transforma y hace una predicción

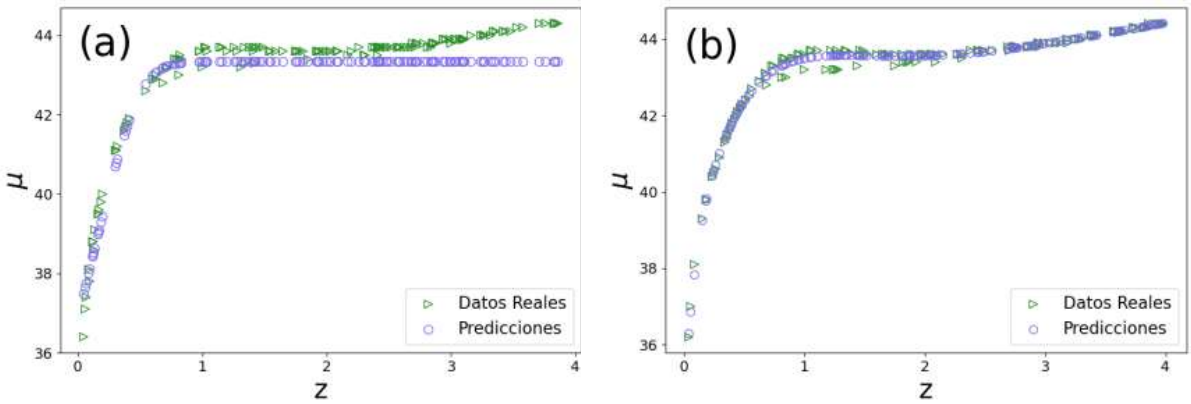


Figura 2.4: Comparación de las predicciones del Perceptrón y la FFNN

para μ usando un learning rate = 0.01 unidades[5] . Al ser un modelo tan simple, entonces podemos interpretar fácilmente sus parámetros. En este caso, de la Ec. (2.10) se puede interpretar que, el sesgo de la única neurona, juega el papel de la magnitud absoluta como b = M. En Python, es posible extraer los pesos y sesgos de las neuronas de una red. De ahí, obtenemos que para nuestra implementación después de 500 épocas de entrenamiento M = 16.2 mientras que una de las mejores estimaciones de este parámetro es M = 19.3 [25]. Teniendo en cuenta que se tiene un error porcentual cercano al 16 % que aunque es una diferencia considerable, resulta interesante dada la simplicidad del modelo.

Por supuesto que el resultado es mejorable si escogemos un modelo distinto, por ejemplo, construimos también una FFNN con 5 capas ocultas y 64 neuronas por capa, y un learning rate dinámico. Los resultados de esta red y del perceptrón pueden verse en la Fig. 2.4, donde es evidente que la FFNN comprende mejor el comportamiento de los datos, pues mientras el perceptrón predice una curva μ bastante aplanada y evidentemente alejada de los datos reales en z > 2, la FNNN se adapta de manera muy suave a la curva verdadera. En la Fig. 2.5 se compara la evolución de la función de pérdida de ambas arquitecturas en el mismo número de épocas de entrenamiento. La línea de referencia en $1.5 \nrightarrow 10^{-1}$ indica el límite asintótico para el perceptrón, mientras que la FFNN reduce la función de costo debajo de este valor en las primeras 50 épocas. Debido a la mayor complejidad del modelo, en etapas tempranas del entrenamiento, la FFNN muestra un proceso de aprendizaje inestable, pues la función de costo tiene crecimientos esporádicos y vemos una línea ruidosa o saturada, que se estabiliza después de 100 épocas debido a la capacidad de auto-ajustar el learning rate. En general, esto nos indica que la FFNN puede

[5] El learning rate es un hipérparámetro adimensional, por lo que en adelante se expresará como un valor sin unidades.

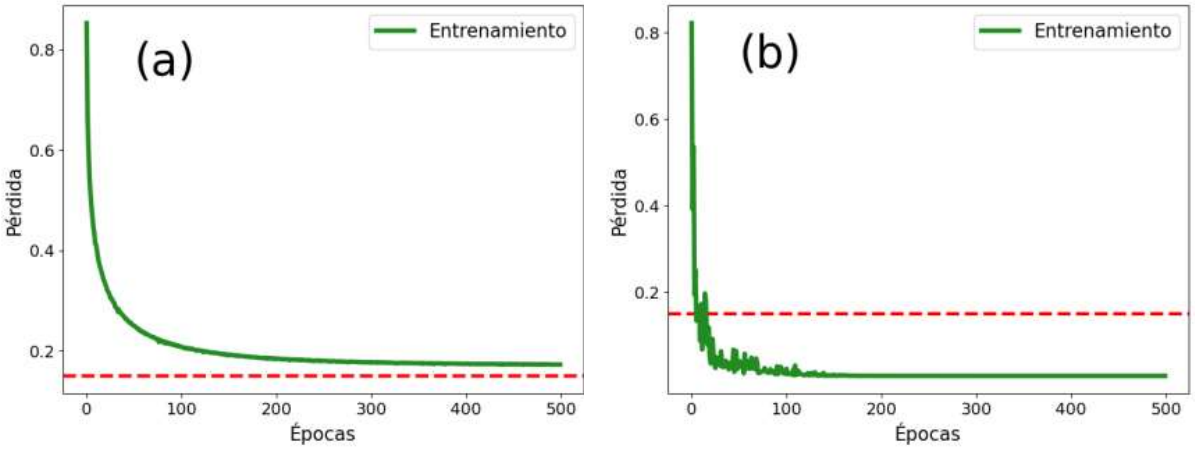


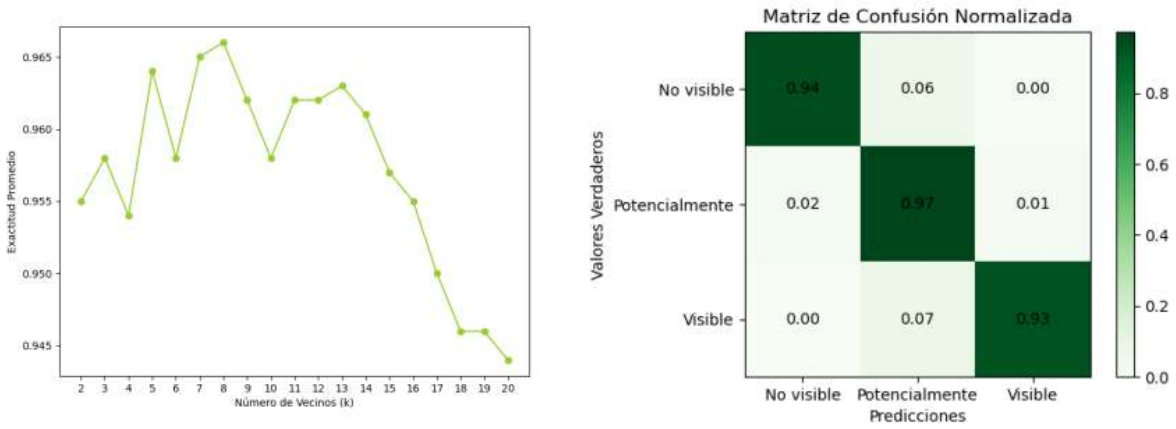
Figura 2.5: Comparación de la evolución de la función de costo del perceptrón y la FFNN. La línea roja en ambas figuras indica el valor 0.15, que para el perceptrón es un umbral asíntotico, pero que la FFNN supera en épocas tempranas del entrenamiento.

aprender mejor y más rápido que el perceptrón. Sin embargo, aunque también podemos extraer los valores de sus parámetros, ya no podemos interpretarlos como en el caso del perceptrón, pues se tienen $64 \nrightarrow 5 = 320$ pesos y sesgos, lo que muestra el compromiso que se tiene al ganar precisión pero perder interpretabilidad de un modelo.

2.4.2. Clasificador de sombras de agujero negro

Utilizando los datos de agujeros negros presentados en el Capítulo 1, construimos un clasificador multiclase basado en el método de kNN. El objetivo es alimentar al clasificador con información de masa y redshift, y que clasifique el radio de la sombra de un agujero negro con esa masa y redshift en las clases Visible, Potencialmente Visible o No visible. Lo interesante es que no necesitamos tener los valores explícitos del radio de la sombra, si no solamente sus etiquetas de clase. Podríamos pensar por ejemplo, en el caso de observaciones de donde se estime la masa en una región del espacio, y este clasificador podría ayudarnos a decidir si es viable dedicar recursos más especializados para observar dicha región.

El método de los k vecinos más cercanos (kNN) es un algoritmo de aprendizaje supervisado utilizado principalmente para clasificación. Su funcionamiento se basa en la proximidad: cuando se presenta un nuevo dato, el algoritmo busca los k puntos de entrenamiento más cercanos en el espacio de características y asigna al nuevo punto la clase más común entre esos vecinos. La elección de k es clave y puede influir en la precisión y robustez



(a) Validación cruzada (b) Matriz de Confusión

Figura 2.6: Elección y evaluación del método kNN: a) Validación cruzada que permite elegir el k óptimo, es decir, aquel que obtiene una exactitud promedio mayor al resto. b) Matriz de confusión que permite evaluar visualmente los resultados, siendo la diagonal la proporción de elementos bien clasificados y fuera de ella, los mal clasificados.

del modelo. Aunque kNN es un método supervisado, tiene una relación conceptual con la clusterización K-Means presentado en el Capítulo 1, que es no supervisada. Sin embargo, mientras K-Means busca particionar el espacio en clusters con centroides definidos, kNN usa esas particiones implícitamente, basándose en vecinos preetiquetados para asignar clases a datos desconocidos.

Para elegir el k en nuestro modelo, usamos el método de validación cruzada, que divide el conjunto de entrenamiento en varias particiones (o "folds") y evalúa el modelo en cada partición, usando el resto de los datos para entrenar; esto permite obtener una medida de exactitud promedio para cada valor de k en el clasificador kNN, optimizando así la elección. En nuestro caso, como se muestra en la Fig. 2.6a el valor óptimo es k = 8, y con ella se obtiene también la matriz de confusión de la Fig. 2.6b, donde los valores diagonales indican aciertos y los fuera de la diagonal, errores de clasificación. Con este modelo se obtuvo una precisión del 95.5 % en la clasificación.

En la Fig. 2.7, se muestran los datos etiquetados y las fronteras de clasificación obtenidas con este método, donde podemos ver que no son lineales. Además, de esta gráfica y de la matriz de confusión, podemos entender que hay diferentes tipos de error, por ejemplo si un potencialmente visible se clasifica como no visible, entonces nos estamos perdiendo de una observación, lo que también puede pasar si un visible se clasifica como potencialmente visible. Sin embargo, un error más grave es clasificar un No visible como Visible, pues entonces se destinarían recursos para observar una región donde no es posible observar

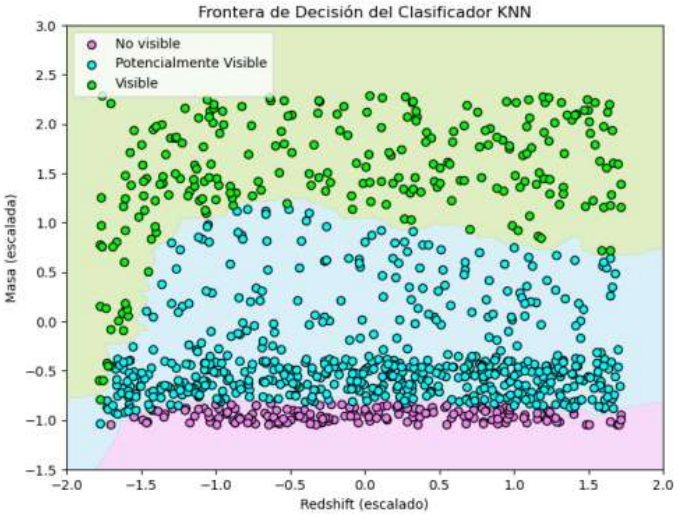


Figura 2.7: Resultados del método kNN. Se muestran los datos etiquetados y las fronteras de clasificación obtenidas por el modelo.

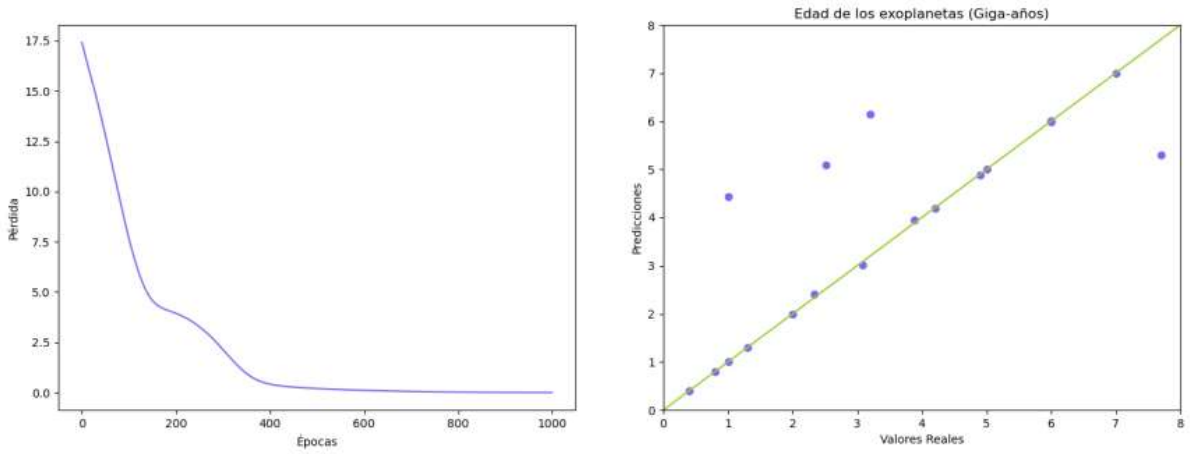
nada. Un modelo más robusto no solo es el que tiene mayor precisión, sino aquel que minimiza los errores más graves.

2.4.3. Datos faltantes de exoplanetas

Con el objetivo de aprovechar la capacidad de predicción de las redes neuronales, exploramos la base de datos de exoplanetas de catálogo de Habitable Worlds Observatory[6]. En ella se tiene información de 3 variables categóricas: Tipo, Método de detección y el Indice de Similitud de la Tierra (ESI por sus siglas en inglés), además de 6 variables numéricas: Masa, Radio, Flujo Magnético, Temperatura Superficial, Edad, Periodo y Dis-tancia.

Resulta interesante a primera vista, que la variable Edad, tiene muchos datos faltantes debido a la dificultad de estimar o deducirla de manera observacional. La estimación de la edad de un exoplaneta puede hacerse mediante diversos métodos, entre los cuales destacan la observación de la actividad estelar y el análisis de la composición de la estrella anfitriona. Estos métodos incluyen la determinación de la edad estelar a través de la ro-tación estelar y la medición de la metalicidad de la estrella, que se correlaciona con su edad aproximada [26, 27]. Para superar las limitaciones de los métodos tradicionales y completar los datos faltantes en nuestro conjunto de datos, emplearemos una red neuronal para estimar la edad de los exoplanetas en función de sus características observadas.

[6] https://phl.upr.edu/hwc



(a) Evolución de la función de costo (b) Predicciones de las edades

Figura 2.8: (a) La evolución de la función de costo en las épocas de entrenamiento, se observa un aprendizaje suave, con una precisión moderada. (b) Comparación de las edades predichas contra las reales. En el caso ideal, las predicciones caerían en la línea verde, pero debido a la cantidad de datos, es de esperar tener algunas predicciones alejadas.

Construimos una FFNN sencilla: en la capa de entrada tomamos las 5 variables numéricas restantes, asignamos dos capas ocultas con 64 neuronas, la primera con la función de activación Softmax y la segunda con la función ReLU. Finalmente, en la capa de salida, tenemos la predicción para la edad. Se entrenó la red por 100 épocas, como se puede ver en la Fig. 2.8a. Debido a la escases de datos, no es conveniente entrenar mucho más pues podríamos sobreajustar la red, y tener predicciones inadecuadas. Un set de datos más amplio, podría ayudar a mejorar ese aspecto. Por otro lado, los resultados se muestran en la Fig. 2.8b, donde al no haber una relación directa conocida con las otras variables, la visualización más conveniente de las predicciones es contra sus valores reales, y se espe-raría en el caso ideal, que se tenga una relación 1 a 1. La línea verde representa la función $f(x) = x$ y sirve como ayuda visual para entender la precisión de este modelo. El hecho de que la mayor parte de los datos se aproximen muy bien a sus valores reales, nos indica que con un set de datos más robusto, la estimación de las edades faltantes puede ser aún más precisa.

2.5. Ventajas y desventajas de usar AI

La inteligencia artificial ha transformado la manera en que se abordan problemas complejos en física, permitiendo manejar grandes volúmenes de datos y encontrar patrones

complejos [21]. En particular, las FFNN tienen la capacidad de modelar sistemas altamente no lineales, algo que es común en física, ofreciendo aproximaciones a fenómenos complejos en los que las relaciones exactas no son obvias [23]. Esta capacidad de la AI es de especial relevancia en áreas como la cosmología, donde los datos observacionales son vastos y abarcan múltiples escalas y variables [18].

Además la AI permite automatizar y optimizar procesos, lo cual es beneficioso para simulaciones complejas. Esto facilita la creación de modelos ajustados y de utilidad en diversos contextos físicos. La AI se ha implementado en tareas como la clasificación de objetos astronómicos y la predicción de propiedades de materiales [28].

Sin embargo, el uso de AI en física presenta limitaciones importantes. La falta de interpretabilidad es una desventaja fundamental de modelos complejos como las FFNN. Muchas veces la gran cantidad de parámetros y la complejidad de sus estructuras internas convierten a estos modelos en una "caja negra", lo cual dificulta la interpretación de sus resultados en términos de principios físicos. Esto es problemático, ya que el conocimiento o sentido físico puede desaprovecharse si el modelo se basa solamente en datos [20]. Al depender exclusivamente de técnicas de AI, corremos el riesgo de construir modelos precisos pero carentes de interpretación física.

Además, como vimos anteriormente, la precisión de un modelo de aprendizaje au-tomático está directamente ligada a la calidad de los datos. En problemas físicos donde los datos son escasos, los modelos pueden presentar sesgos y tener una baja capacidad de generalización [29].

Finalmente, aunque la AI permite avances significativos en física, es esencial complementarla con conocimiento físico explícito para evitar que la dependencia de los datos limite su potencial predictivo. En el siguiente capítulo, se abordará cómo las redes neu-ronales informadas en física (PINNs) integran de manera explícita las leyes físicas en el proceso de aprendizaje, buscando superar algunas de estas limitaciones.

Capítulo 3

PINNs: Redes Neuronales Informadas en Física

Hasta ahora hemos implementado métodos de exploración de datos y de AI con con-juntos de datos físicos, y hemos visto también algunas de las limitaciones de este enfoque. En este capítulo, presentamos e implementamos una herramienta que busca combinar lo mejor de las redes neuronales y mejorarlas, al combinarlas con restricciones y conocimiento físico: las Redes Neuronales Informadas en Física.

3.1. Física Computacional y el Surgimiento de las PINNs

En el campo de la física, muchos problemas han sido resueltos en primera instancia a través de deducciones basadas en observaciones. Por ejemplo, el caso de las leyes de Kepler, aunque para muchos otros, nos enfrentamos a un escenario donde la solución de un problema proviene de métodos numéricos y computacionales, y donde la Inteligencia Artificial y las Redes Neuronales han traído un cambio considerable en la manera en que resolvemos problemas computacionales. En este contexto, encontrar soluciones a ecuaciones diferenciales (DE) ha sido uno de los problemas fundamentales en las matemáticas, lo que toma relevancia al notar que la mayor parte de los fenómenos físicos pueden ser descritos a través de este tipo de ecuaciones.

Tradicionalmente, métodos numéricos como el de diferencias finitas y el método de ele-mentos finitos han sido ampliamente utilizados para resolver ecuaciones diferenciales or-dinarias y parciales (ODEs y PDEs) en diversos dominios físicos, desde la dinámica de

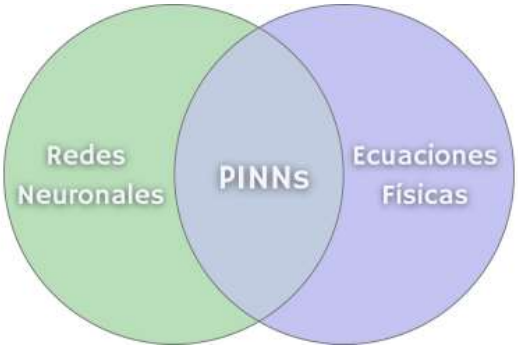


Figura 3.1: Las PINNs combinan la eficiencia de las redes neuronales y el conocimiento físico previo de un fenómeno a través de ecuaciones diferenciales.

fluidos hasta la mecánica estructural [30]. Sin embargo, estos enfoques presentan limitaciones significativas, como la necesidad de discretizar el espacio de simulación en mallas finas, como lo que presentamos en el ejemplo del estornudo humano, lo cual demanda gran cantidad de recursos computacionales y puede introducir errores de truncamiento.
Los intentos de usar AI para resolver ecuaciones diferenciales se remontan a la década de 1990 [1] cuando se propuso un algoritmo que partía de una función de prueba (anzats) que cumpliera estrictamente las condiciones iniciales, lo que requería de un conocimiento previo o habilidad para proveer dicha función de prueba. Luego de eso han surgido una gran variedad de métodos.
Recientemente, las redes neuronales informadas en física han emergido como una metodología innovadora para resolver ecuaciones diferenciales incorporando directamente las leyes físicas en la estructura de la red neuronal (ver Fig. 3.1). Las PINNs permiten resolver problemas complejos de frontera y condiciones iniciales sin necesidad de mallas espaciales y pueden aplicarse en dominios irregulares [29]. Esta capacidad ha ampliado el uso de las PINNs en diversas áreas, incluidas la dinámica de fluidos, la transferencia de calor y el electromagnetismo, marcando un avance significativo en la forma en que resolvemos problemas computacionales en física [31].

3.2. PINN's: intersección de dos mundos

Como ya se mencionó, en física muchos problemas son descritos a través de ecuaciones diferenciales, sean parciales u ordinarias. En general, una ecuación diferencial parcial (PDE) puede expresarse en la forma:

$$D\left(x, y(x), \frac{\partial y}{\partial x_1}, \ldots, \frac{\partial y}{\partial x_d}, \frac{\partial^2 y}{\partial x_1 \partial x_1}, \ldots, \frac{\partial^2 y}{\partial x_1 \partial x_d}, \ldots\right) = 0, \quad (3.1)$$

donde $\vec{x} = (x_1, \ldots, x_d)$ es un vector d-dimensional definido en una región $\mathbb{X} \dashrightarrow R^d$, y $y(\vec{x})$ es la solución, que satisface también alguna condición de frontera:

$$B(\vec{x}, y(\vec{x})) = 0 \quad \text{para} \quad \vec{x} \in \partial\mathbb{X}, \tag{3.2}$$

donde $\partial\mathbb{X}$ es la frontera de $\mathbb{X}$.

Sin embargo, es un hecho que muchas veces no es posible encontrar soluciones analíticas a las ecuaciones que describen un fenómeno. No obstante, podemos hacer algunas observaciones o experimentos sobre éste. La intención de las PINNs es buscar combinar estas ecuaciones con los datos disponibles para obtener soluciones que mantengan el sentido y las leyes físicas fundamentales de un problema, y esto se logra modificando la función de costo de una PINN como

$$L(\checkmark, \ ) = \omega_O L_O(\checkmark, \ _O) + \omega_D L_D(\checkmark, \ _D) + \omega_B L_B(\checkmark, \ _B), \tag{3.3}$$

donde $\checkmark$ son los parámetros de la red neuronal y $\ $ es el conjunto de datos de entrada, con los índices O, D y B, correspondiendo a Observaciones, Dominio y Frontera (Boundary), respectivamente. Los parámetros ω son pesos[1] que podemos asignar a cada parte de la función de costo según su relevancia para nuestro problema. Aunque no hay una forma específica de elegir valores para estos pesos, generalmente se asignan pesos iguales para cada parte de la función de costo, aunque esto puede resultar en un sobreajuste en los casos en que el conjunto $_O$ es muy pequeño en comparación al total de datos. Para evitar ese problema, una opción es escoger cada peso en proporción a la cantidad de datos de entrenamiento en cada parte de la función de costo. Por ejemplo, si en un problema hay datos distribuidos con 60 % en $_D$, 30 % en $_B$ y 10 % en $_O$, podemos proponer los pesos como $\omega_D = 0.6$, $\omega_B = 0.3$ y $\omega_O = 0.1$.

Por observaciones nos referimos a datos medidos en experimentos, si están disponibles, y generalmente usaremos la función de costo MSE:

$$L_O(\checkmark, \ _O) = \frac{1}{N_O} \sum_{\vec{y} \in \ _O} |\vec{y}_{nn} - \vec{y}|^2 \tag{3.4}$$

Por otra parte, para el dominio, generaremos datos de entrada sintéticos, simplemente eligiendo datos distribuidos generalmente de manera uniforme, que servirán para evaluar la función de costo relacionada a las constricciones de las ecuaciones diferenciales, que se escribirá en general como

[1] No se deben confundir con los pesos de las neuronas. Estos pesos actúan unicamente en la parte final del aprendizaje, en la función de costo.

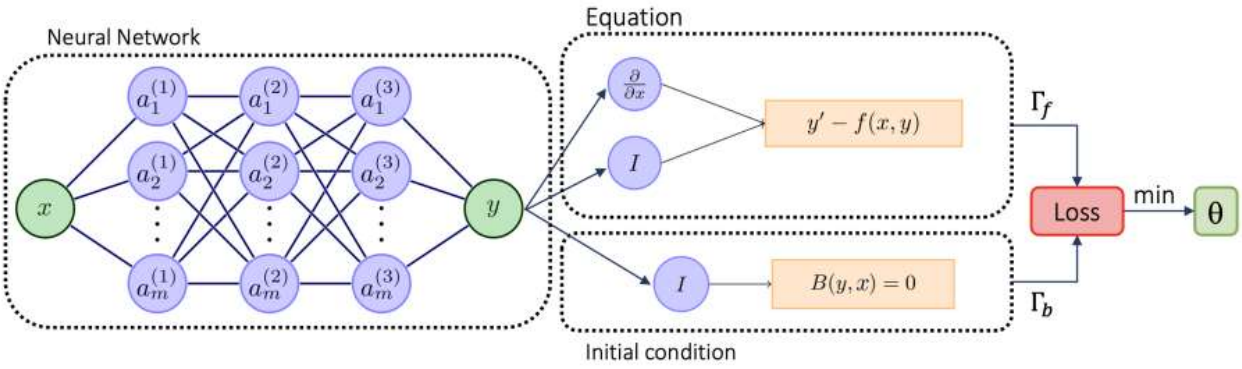


Figura 3.2: Arquitectura general de una PINN: a la izquierda se observan la capa de entrada, las conexiones neuronales de la red y la capa de salida; a la derecha se tiene la parte informada de la red, a través de las ecuaciones diferenciales que se evaluarán con los datos $_D$ y las condiciones iniciales y de frontera que se evaluarán con los datos $_B$ formando así la función de costo total que minimizará los parámetros ✓.

$$L_D(\checkmark, \ _D) = \frac{1}{N_D} \sum_{x2 \ _D} \left| D\left(x, y_{nn}(x), \frac{@y_{nn}}{@ \ _1}, \ldots, \frac{@^2 y_{nn}}{@ \ _1 @ \ _1}, \ldots\right) \right|^2 \tag{3.5}$$

, las condiciones iniciales y de frontera serán incluidas en la función de costo

$$L_B(\checkmark, \ _B) = \frac{1}{N_B} \sum_{x2 \ _B} |B(y_{nn}, x)|^2 \tag{3.6}$$

Una vez que se tienen definidas las funciones de costo, puede definirse la estructura de la PINN, que en general es como en la Fig. 3.2. Aquí es importante notar que para calcular las funciones $L_D(\checkmark, \ _D)$ y $L_B(\checkmark, \ _B)$, es necesario calcular las derivadas de la capa de salida respecto a las variables de entrada, esto es posible usando algunas utilidades de auto-derivación disponibles en paqueterías de Python como Pythorch[2] y Tensorflow[3], lo que permite aplicar algoritmos de minimización conocidos para optimizar los resultados de la PINN.

3.3. Implementaciones

En esta sección se presentan algunas implementaciones de PINNs y sus particularidades. Exploramos diversos problemas físicos, tanto en el enfoque de PINNs puras como PINNs combinadas con datos, abordamos problemas y situaciones comunes en el uso de

[2] https://pytorch.org/tutorials/beginner/blitz/autograd_tutorial.html
[3] https://www.tensorflow.org/guide/advanced_autodiff

las PINNs, su entrenamiento y la elección de un modelo adecuado al problema que se este resolviendo.

3.3.1. Decaimiento radioactivo

El decaimiento de partículas se puede describir utilizando la ley de desintegración radiactiva, que establece que la tasa de cambio de la cantidad de material radiactivo en el tiempo es proporcional a la cantidad presente. Esta relación se expresa matemáticamente como la ecuación diferencial

$$\frac{dR}{dt} = \quad R \ , \qquad 3.7 \ (\ \ )$$

donde R es la cantidad de material radiactivo presente en un momento dado y es la constante de desintegración, que representa la probabilidad de desintegración por unidad de tiempo.

Como primer ejemplo, tomaremos el caso más simple donde = 1, así implementamos una PINN sin datos, es decir, que solo tomará en consideración las funciones de pérdida del dominio, que según la Ec.(3.5) se reducen a:

$$L_D = \frac{1}{N_D} \sum_{i=1}^{N_D} \left(\frac{dR}{dt} + R \right)^2 , \qquad (3.8)$$

y considerado la condición inicial $R(0) = 1$, a partir de la Ec.(3.6) tenemos la siguiente parte de la función de costo:

$$L_B = \frac{1}{N_B} \sum \ (R(0) \quad 1)^2 . \qquad (3.9)$$

En este caso, como no estamos usando datos observacionales, los pesos de la función de costo total, indicados en la Ec.(3.3) son $!_O = 0$ y $!_D = !_B = 1$. La PINN tiene una neurona de entrada que toma valores de $0 < t < 5$ segundos y una neurona de salida que predice R, tiene 3 capas ocultas con 50 neuronas cada una. Usamos un learning rate = $8 \rightarrow 10^3$ y la función de activación LeakyReLU para entrenar con $N_D = N_B = 500$ datos. Los resultados después de entrenar la PINN por 5000 épocas se muestran en la Fig. 3.3, y como podemos ver resuelve muy bien en el dominio; aunque fuera de este la predicción se aleja gradualmente de la solución analítica. Este es un comportamiento esperado en las PINNs, y no es necesariamente un defecto, pues todos los parámetros de la red se ajustaron para minimizar la pérdida específicamente en el dominio de nuestro interés.

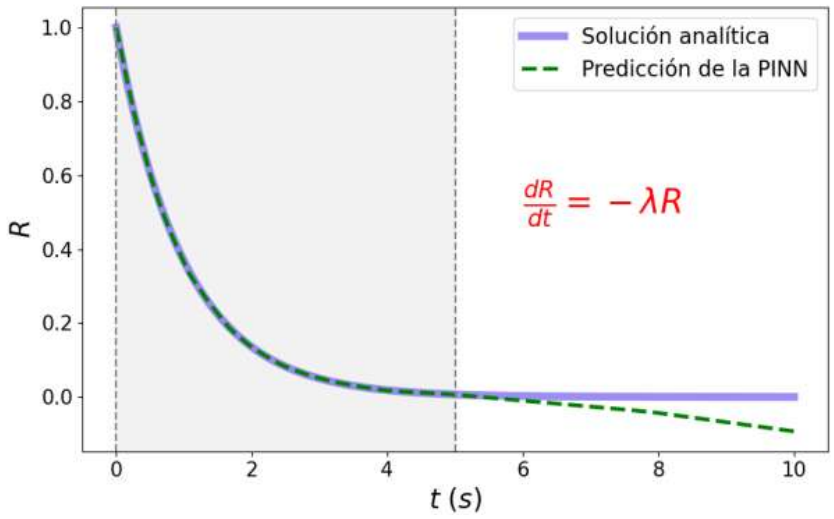


Figura 3.3: Predicción de la PINN para el problema de decaimiento radioactivo. La región sombreada indica el dominio de entrenamiento de la red.

3.3.2. Caída libre

En esta implementación veremos una de las formas en que podemos decidir cuando detener el entrenamiento de una PINN. Para esto se analiza el problema de caída libre. La caída libre es uno de los primeros fenómenos en estudiarse al introducirnos a la física. En el caso donde no se considera el rozamiento del aire, la ecuación diferencial que describe este fenómeno es:

$$\frac{d^2y}{d^2t} = g \tag{3.10}$$

con y la altura, y $g = 9.81\ m/s^2$ la aceleración debida a la gravedad. Considerando esta ecuación, construimos una PINN que tome valores de t para predecir y. La arquitectura de la PINN consiste en 4 capas ocultas de 64 neuronas cada una, así como una neurona de entrada y una de salida. En cada capa oculta elegimos usar la función de activación Tanh porque su suavidad facilita la convergencia del modelo, siendo una opción estándar y ampliamente probada en redes neuronales; además usamos un learning rate = $1 \rightarrow 10^3$. La función de costo en el dominio será:

$$L_D = \frac{1}{N_D} \sum_{i=1}^{N_D} \left(\frac{d^2y}{d^2t_i} + g \right)^2, \tag{3.11}$$

que evaluará $N_D = 1000$ puntos en el intervalo $0 < t < 1$. Por otra parte, estamos suponiendo que se parte del reposo a 10 metros sobre el origen del marco de referencia.

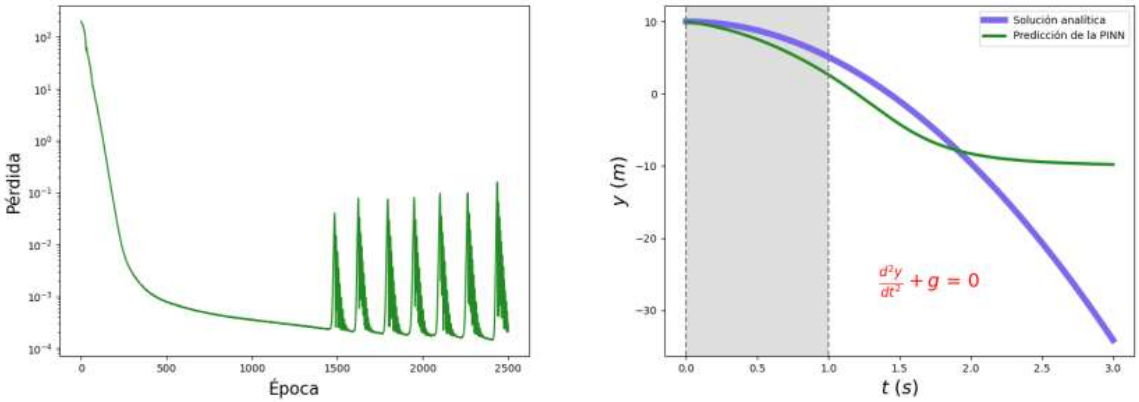
Luego con esta condición de frontera se establece la función de costo:

$$L_B = \frac{1}{N_B} \sum (y(0) \quad 10)^2 . \tag{3.12}$$

Ya que esta PINN no tiene datos observacionales, de la misma manera que en la implementación del decaimiento radioactivo, se asignan los pesos $!_O = 0$ y $!_D = !_B = 1$. En una primera etapa, se entrenó la PINN por 2500 épocas, obteniendo los resultados de la Fig. 3.4. Vemos que las predicciones se ajustan bien cerca de la condición de fron-tera, pero se alejan de la solución analítica rápidamente, incluso dentro del dominio de entrenamiento. Además, en la evolución de la función de pérdida de la Fig. 3.4a vemos una saturación a partir de 1500 épocas, es decir, el valor de pérdida aumenta y decrece de manera inestable. Esto puede deberse a diferentes razones, aunque la más común es que la PINN está cerca de un mínimo local en el espacio de parámetros y entonces el learning rate es muy grande. Aunque hay diversas formas de solucionar esta dificultad, que abordaremos en las siguientes implementaciones, en este problema podemos optar observar qué sucede al detener el entrenamiento en épocas más tempranas. Esta decisión se puede tomar considerando que el tiempo real de entrenamiento de esta red es bajo, además de que antes de las 1500 épocas la PINN alcanza un umbral de error de $1 \rightarrow 10^{-3}$, que según nuestros criterios y necesidades de precisión puede ser suficiente. Además, la función de pérdida decrece suavemente antes de saturarse, por lo que elegimos entrenar hasta 1400 épocas. Los resultados de ese entrenamiento pueden verse en la Fig. 3.5, don-de se obtiene un ajuste mucho mejor en el dominio de entrenamiento, mostrando que en este caso no es necesario entrenar más épocas, y que en general un entrenamiento más largo no necesariamente dará mejores resultados. Esta solución puede parecer empírica, sin embargo, es una buena opción en modelos que requieren bajo costo computacional y donde se haya alcanzado un umbral de precisión preestablecido.

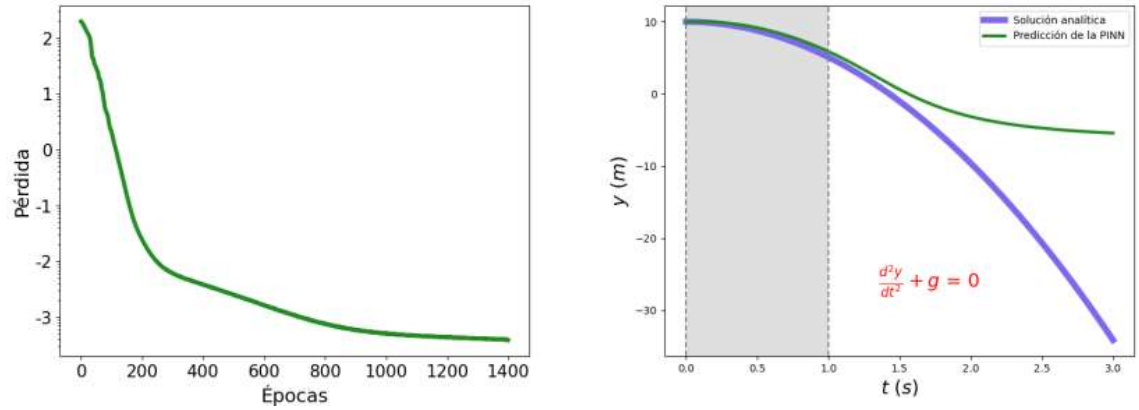
3.3.3. Trayectoria de un proyectil

En esta implementación veremos como combinar las constricciones físicas con observa-ciones y las ventajas que tiene una PINN respecto a una FFNN basada en datos, además de una nueva estrategia para mejorar el aprendizaje continuo de la PINN.
Consideremos una base de datos con 3 columnas de 100 elementos (t, x, y) que representan la información sobre el movimiento de un proyectil en el intervalo de 0 a 6 segundos. Entrenamos una FFNN estándar con una neurona de entrada y dos de salida para predecir el desplazamiento del proyectil. Dicha red, tiene 3 capas ocultas con 256, 128 y 64 neuronas cada una, y como está basada únicamente en datos, utiliza la función de costo L_{MSE}. Se utilizó el optimizador Adam, y un learning rate = $1 \rightarrow 10^{-3}$. Con esta configuración, la red



(a) Evolución de la función de costo (b) Resultados de la PINN

Figura 3.4: Entrenamiento de la PINN a 2500 épocas: (a) se observa que el valor de la función de costo se satura después de 1500 épocas, y (b) las predicciones de la PINN son buenas cerca de la condición de frontera pero se alejan rápidamente de la solución analítica. El área sombreada indica el dominio de entrenamiento.



(a) Evolución de la función de costo (b) Resultados de la PINN

Figura 3.5: Entrenamiento de la PINN a 1400 épocas: (a) Evolución de la función de costo, se observa que decrece suavemente en esta etapa, y (b) las predicciones de la PINN mejoran considerablemente en la región de entrenamiento (área sombreada).

es capaz de aprender muy bien el desplazamiento con unas cuantas épocas de entrenamiento, como se ve en la Fig. 3.6. De hecho, con esta cantidad y calidad de datos, puede resultar excesivo implementar redes neuronales, pues el fenómeno es evidente y ningún método de regresión tradicional debería tener dificultades en resolverlo.

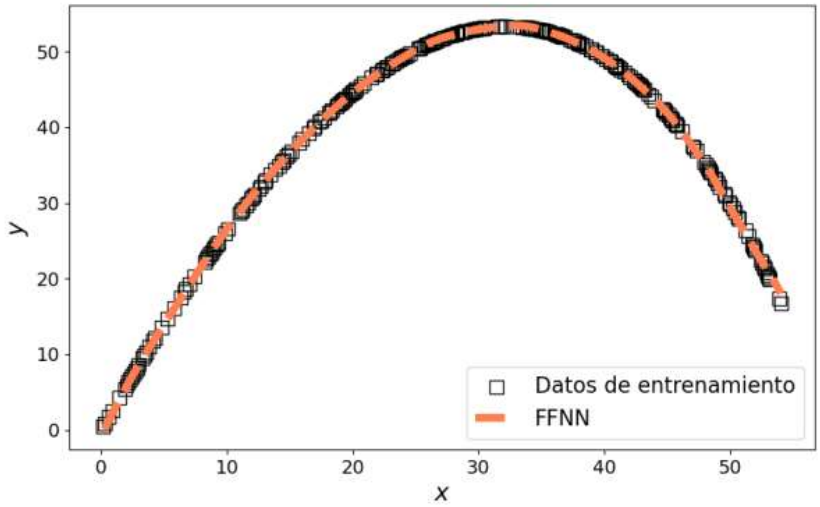


Figura 3.6: Resultados de la FFNN con 100 datos limpios de movimiento proyectil.

Sin embargo, comúnmente no tenemos tantos datos, ni tan limpios. En un escenario más realista, consideremos solo 10 datos con ruido, e implementamos una red con la misma arquitectura. En ese caso, como se ve en la Fig. 3.7, la red no es capaz de predecir el fenómeno correctamente más allá del intervalo de las observaciones, lo que es un comportamiento esperado de las FFNN y muestra la carencia de sentido físico en las predicciones que solo se basan en datos. Por lo tanto, no podemos confiar por completo en los resultados más allá de regiones donde tengamos observaciones para redes con pocos datos. Aquí es donde podemos aprovechar las virtudes de las PINNs.

En este problema, es posible deducir a partir de ecuaciones fundamentales, la ecuación diferencial para el desplazamiento de proyectil como:

$$\mu \frac{ds}{dt}\frac{ds}{dt} - g\frac{d^2s}{dt^2} = 0\,, \tag{3.13}$$

donde s es el desplazamiento, μ el coeficiente de rozamiento del aire y g la gravedad. Así, se propone la función de costo

$$L = \frac{1}{N_D}\sum_{D}\left(\mu\frac{ds}{dt}\frac{ds}{dt} - g\frac{d^2s}{dt^2}\right)^2, \tag{3.14}$$

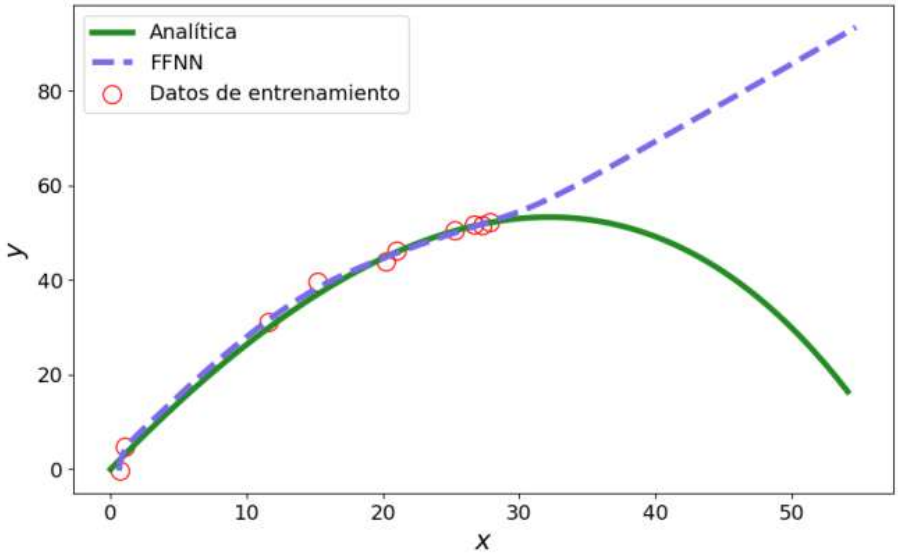


Figura 3.7: Resultados de la red neuronal (FFNN) para la trayectoria de un proyectil utilizando 10 datos con ruido solo de un pequeño intervalo del dominio. Fuera del dominio de los datos, la predicción es mala.

Como usualmente sucede en los experimentos, conocemos algunas condiciones iniciales, a decir:

$$s(0) = (0,0), \quad v_{0x} = 36, \quad v_{0y} = 12, \tag{3.15}$$

que serán consideradas para la función de costo de frontera

$$L_B = \frac{1}{N_{B_1}} \sum \left(s(0) \quad (0,0)\right)^2 + \frac{1}{N_{B_2}} \sum \checkmark \frac{ds}{dt}\Big|_{t=0} \quad (36,12) \quad \blacklozenge_2 \quad . \tag{3.16}$$

Usando una arquitectura idéntica a la de las dos redes anteriores, pero usando estas nuevas funciones de costo con pesos homogéneos $!_O = !_D = !_B = 1$, podemos solucionar el problema más allá del dominio de las observaciones, como se ve en la Fig. 3.8. En una primera evaluación se observó una saturación de la función de costo, es decir, dejó de aprender en las primeras épocas. A diferencia de la implementación de caída libre, donde se detuvo el entrenamiento de la PINN en épocas tempranas, aquí usamos un enfoque distinto pues no se logra alcanzar una precisión aceptable, en cambio, para mejorar el rendimiento de la red, utilizamos un learning rate dinámico. Partiendo de un lr=0.01, se fue reduciendo a la mitad cada 2000 épocas. Los efectos de esto se reflejan en la Fig. 3.9, donde podemos ver algo de saturación en la función de pérdida física, y en las marcas de 2000 épocas notamos una mejora en el aprendizaje. Este enfoque puede usarse tanto en épocas definidas como en este caso cambiándolo cada determinado número de épocas,

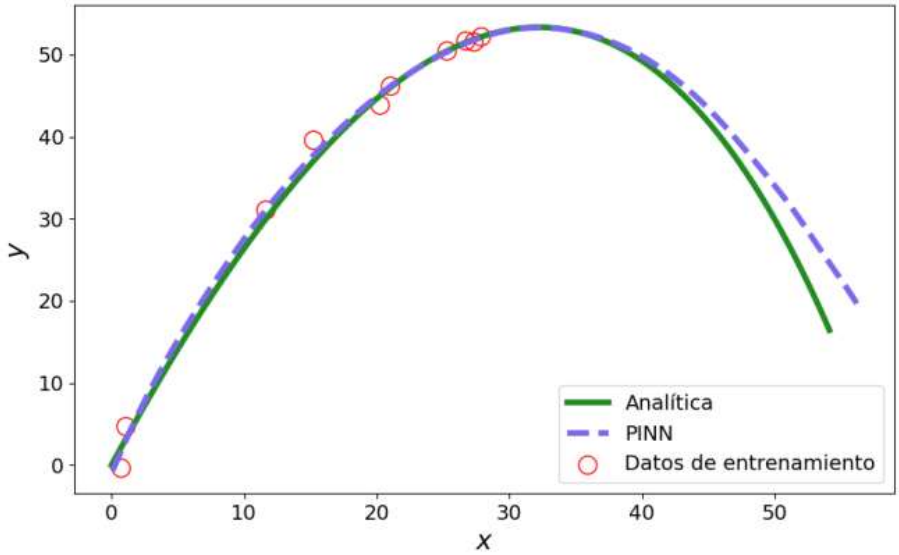


Figura 3.8: Resultados de la red neuronal informada en física (PINN) para la trayectoria de un proyectil utilizando 10 datos con ruido solo de un pequeño intervalo del dominio y la función de costo física. Fuera del dominio de los datos, la predicción mejora considera-blemente.

pero también se pueden definir ventanas de espera, es decir, si la función de costo no tiene una mejora significativa en un número determinado de épocas, entonces ahí se hace el cambio de learning rate.
De la Fig.(3.9), podemos notar que la red aprende más rápido con los datos (línea verde), pero alcanza una asíntota rápidamente. En el lado opuesto, la función de pérdida física aprende más lento al inicio del entrenamiento (línea morada), pero tiene un mejor ritmo de aprendizaje en épocas más avanzadas. Es decir, las PINNs tienen un gran potencial de aprendizaje continuo.

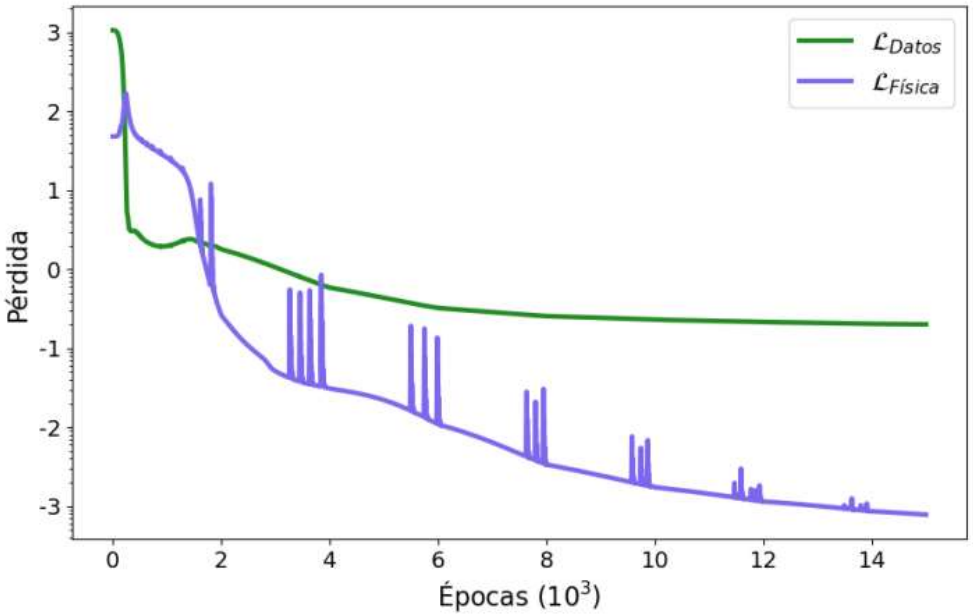


Figura 3.9: Evolución de la función de pérdida para la PINN del proyectil.

3.3.4. El péndulo simple

En esta implementación solucionamos una ecuación de orden superior y abordamos una metodología para elegir hiperparámetros representado por la ecuación del péndulo para ángulos pequeños:

$$\frac{d^2\theta}{dt^2} = -\frac{g}{L}\theta, \tag{3.17}$$

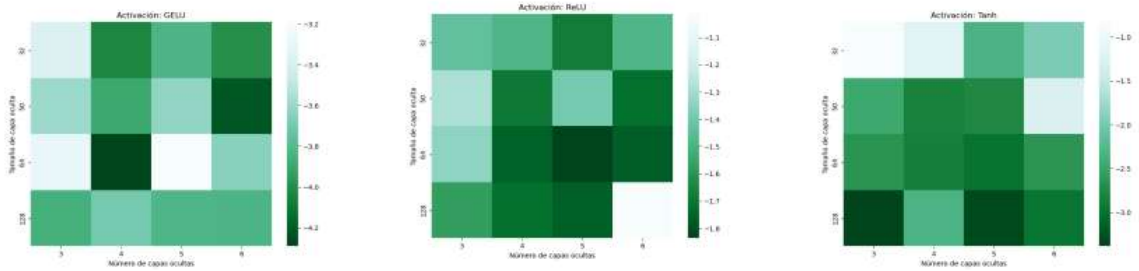
donde θ es el ángulo que forma el péndulo respecto al eje z, g es la gravedad y L la longitud de un péndulo simple. A partir de ella construimos la función de costo:

$$L_D = \frac{1}{N_B}\sum_{i=1}^{N_B}\left(\frac{d^2\theta}{dt_i^2} + \frac{g}{L}\theta\right)^2. \tag{3.18}$$

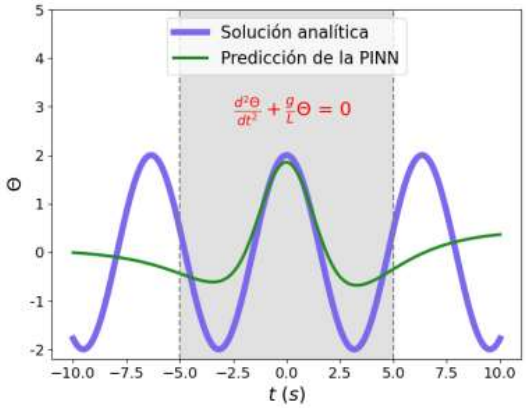
Supondremos que el valor de la longitud es conocido, $L = 10$, que además tenemos la condición inicial $\theta_0 = 2$, lo que nos permite escribir la siguiente parte de la función de costo como

$$L_B = \frac{1}{N_B}\sum(\theta(0) - 2)^2. \tag{3.19}$$

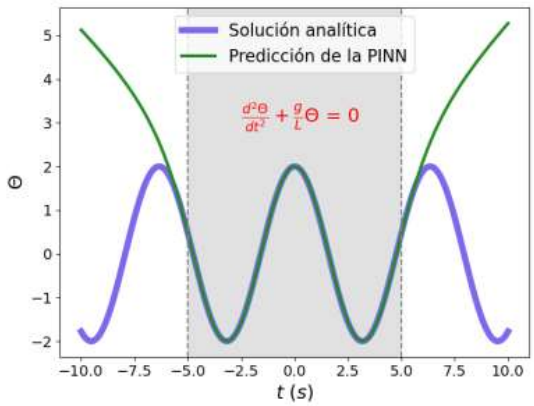
Esta PINN tiene una neurona de entrada que toma valores de $-5 < t < 5$ segundos y una neurona de salida que predice θ, tiene 3 capas ocultas con 50 neuronas cada una. Usamos un learning rate = $1\theta10^{-3}$ y la función de activación LeakyReLU para entrenar con $N_D = 500$ datos, con lo que se obtienen los resultados de la Fig. 3.10b.



(a) Grid Search CV



(b) PINN inicial



(c) PINN del Gridsearch

Figura 3.10: (a) Grid Search de la función de costo y la arquitectura de la PINN, divididos por función de activación. El modelo mejor evaluado es aquel con la puntuación más baja (color más oscuro). Entrenamiento de la PINN para el péndulo simple: (b) resultados de la arquitectura inicial, y (c) resultados con hiperparámetros elegidos por Grid Search CV.

Es evidente que los resultados no son muy buenos, y aunque hasta ahora nuestra mayor herramienta ha sido modificar el learning rate, las redes neuronales nos brindan flexibilidad en la modificación de sus hiperparámetros y su arquitectura. Por ello realizamos un Grid Search CV sobre la función de activación y diferentes arquitecturas. Probamos tres de las funciones de activación de uso frecuente: Tanh, ReLu y GeLu, y arquitecturas con $C = \{3, 4, 5, 6\}$ capas ocultas y número de neuronas $\#S = \{32, 50, 64, 128\}$. Este método probará todos las posibles combinaciones sobre los hiperparámetros escogidos, usando un entrenamiento por bloques y realizando una validación cruzada, para finalmente asignar una puntuación a cada combinación. Los resultados del Grid Search se muestran en la Fig. 3.10a; es importante notar que en esta validación se busca la puntuación más baja, y consecuentemente la celda más oscura de la visualización. Como se visualiza cada función de activación por separado, hay que poner especial atención en la escala. Este análisis nos

indica que la combinación que mejor funciona en este caso es GeLu con una arquitectura de 64 neuronas y 3 capas ocultas. Los resultados de esa modificación se ven en la Fig. 3.10c. Aunque el learning rate es un hiperparámetro que también podría ajustarse con este método, hemos visto antes las ventajas de usar el enfoque de learning rates dinámicos, por lo que no consideramos necesario ajustarlo a un valor fijo. Con estos casos hemos explorado diferentes herramientas que nos permitirán obtener resultados precisos a problemas físicos más complejos.

Capítulo 4

Una PINN para las ecuaciones de campo de Einstein

Las PINNs han mostrado ser capaces de dar soluciones a problemas de alta complejidad, por lo que ahora buscamos resolver las ecuaciones de campo de Einstein con este enfoque. En este capítulo, encontraremos la solución de Schwarzschild para estas ecuacio-nes, describiendo el proceso de construcción de la PINN; además usaremos métodos de validación para la elección del modelo más adecuado y sus hiperparámetros.

4.1. Ecuaciones de campo y la solución de Schwarzschild

Comencemos por recordar que las ecuaciones de campo de la relatividad general describen cómo la curvatura del espacio-tiempo está influenciada por la presencia de materia y energía. En 1915, Albert Einstein publicó las ecuaciones de campo de la relatividad general, que se escriben como:

$$R_{\mu\nu} - \frac{1}{2}g_{\mu\nu}R + \Lambda g_{\mu\nu} = \frac{8\pi G}{c^4}T_{\mu\nu}, \tag{4.1}$$

donde c es la velocidad de la luz, G es la constante de gravitación universal y Λ la llamada constante cosmológica. Aquí, $R_{\mu\nu}$ es el tensor de Ricci, que proporciona información de la curvatura, R es su traza, que se obtiene al contraer el tensor con la métrica $g_{\mu\nu}$, y $T_{\mu\nu}$ es el tensor de energía-momento que contiene información sobre la distribución de materia y energía en el dominio de estudio.

La solución de Schwarzschild fue la primera solución exacta a las ecuaciones de campo y describe el campo gravitatorio producido por una masa esférica inmersa en el vacío. Para obtenerla se consideran ciertas restricciones:

- La métrica debe tener simetría esférica.
- El espacio-tiempo debe ser estático.
- Lejos de la fuente de gravedad se recupera la métrica de un Universo plano.

Además, se considera que la masa está aislada en el vacío, es decir que $T_{\mu\nu} = 0$. Con estas consideraciones, resolver las ecuaciones de campo técnicamente se reduce a resolver:

$$R_{\mu\nu} = 0 \, . \tag{4.2}$$

Luego, utilizando las hipótesis físicas mencionadas, y mediante una serie de cálculos tensoriales [32], se obtiene como solución la métrica:

$$g_{\mu\nu} = \begin{pmatrix} 1 - \frac{2M}{r} & 0 & 0 & 0 \\ 0 & \left(1 - \frac{2M}{r}\right)^{-1} & 0 & 0 \\ 0 & 0 & r^2 & 0 \\ 0 & 0 & 0 & r^2 \sin^2\theta \end{pmatrix} \tag{4.3}$$

donde se asume la convención $c = G = 1$. Esta métrica describe el campo gravitatorio producido por una masa esférica inmersa en el vacío, como lo pueden ser las estrellas que giran lentamente, o los agujeros negros.

4.2. Generalidades de la PINN

En esta sección, describimos los elementos esenciales para construir una PINN que resuelva las ecuaciones de campo. El objetivo es que con valores dados de las coordenadas espacio-temporales x_μ (con $\mu = t, r, \theta, \varphi$), se predigan las componentes de la métrica $g_{\mu\nu}$ y, particularmente, las de la solución de Schwarzschild en la Ec. (4.3). Como se ha dicho, hallar la solución de Schwarzschild es equivalente a resolver la Ec. (4.2), por lo que la función de costo de esta PINN será

$$L_{\text{Schwarzschild}} = \frac{1}{N} \sum_{i=0}^{N} (R_{\mu\nu})^2 \, . \tag{4.4}$$

Esta es una función de costo válida porque contiene la información física relevante para este problema particular, y aunque podría no ser evidente, dichas constricciones físicas

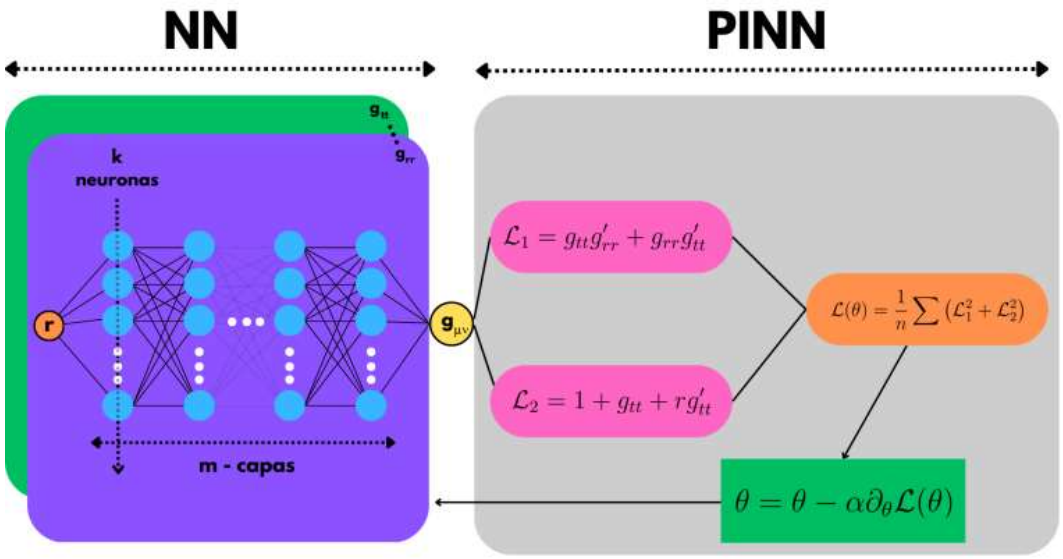


Figura 4.1: Estructura de una PINN para encontrar la solución de Schwarzschild a las ecuaciones de campo de Einstein.

están dadas por ecuaciones diferenciales. Profundizando en la Ec. (4.2) veamos que, ya que el escalar de Ricci R es una contracción del tensor de Ricci, es decir que

$$R = R^{\mu}_{\mu} . \tag{4.5}$$

y que a su vez, el tensor de Ricci es una contracción del tensor de Riemann

$$R_{\mu\nu} = R^{\lambda}_{\mu\lambda\nu} , \tag{4.6}$$

donde el tensor de Riemann por su parte está definido como

$$R^{\rho} = \partial \Gamma^{\rho} - \partial \Gamma^{\rho} + \Gamma^{\rho} + \Gamma^{\rho} , \tag{4.7}$$

podemos ver que es aquí donde comienzan a ser evidentes las ecuaciones diferenciales, pues tenemos derivadas parciales de los símbolos de Christoffel. Más aún, dichos símbolos son definidos en términos de derivadas de la métrica $g_{\mu\nu}$

$$\Gamma^{\rho}_{\mu\nu} = \frac{1}{2} g^{\rho\sigma} [\partial_\mu g_{\nu\sigma} + \partial_\nu g_{\mu\sigma} - \partial_\sigma g_{\mu\nu}] , \tag{4.8}$$

por lo que los símbolos de Christo↵el son una pieza clave en la construcción de la función de costo PINN. El cálculo completo de estos símbolos puede consultarse en el Apéndice B. Por ahora sirve conocer que los términos no nulos son los siguientes:

i) $\Gamma^{t}_{tr} = \frac{1}{2}\frac{1}{A}\partial_r A,$

ii) $\Gamma^{r}_{tt} = \frac{1}{2}\frac{1}{B}\partial_r A,$

iii) $\Gamma^{r}_{rr} = \frac{1}{2}\frac{1}{B}\partial_r B,$

iv) $\Gamma^{r}_{\phi\phi} = -\frac{r}{B},$

v) $\Gamma^{r}_{\theta\theta} = -\frac{r}{B}\operatorname{sen}^2\phi,$

vi) $\Gamma^{\phi}_{\theta\theta} = -\operatorname{sen}\phi\cos\phi,$

vii) $\Gamma^{\phi}_{r\phi} = \frac{1}{r},$

viii) $\Gamma^{\theta}_{r\theta} = \frac{1}{r},$

ix) $\Gamma^{\theta}_{\phi\theta} = \cot\phi,$

donde para facilitar la lectura se han renombrado las variables $A = -g_{tt}$ y $B = g_{rr}$. Estos coeficientes son suficientes para calcular los componentes R_{tt}, R_{rr} y $R_{\phi\phi}$ del tensor de Ricci, y obtener un conjunto de ecuaciones que tendrán como variables a las funciones g_{tt} y g_{rr}. Es decir, de la Ec. (4.2), se tiene que

$$R_{tt} = R_{rr} = R_{\phi\phi} = 0\,. \tag{4.9}$$

Sustituyendo los valores de los símbolos de Christo↵el en la definición del tensor de Ricci, y organizando algebraicamente los términos, obtenemos dos ecuaciones diferenciales

$$g_{tt}\partial_r g_{rr} + g_{rr}\partial_r g_{tt} = 0\,, \tag{4.10}$$

$$1 + g_{tt} + r\partial_r g_{tt} = 0\,. \tag{4.11}$$

Estas dos ecuaciones son las que constituyen la función de costo que se propone en la Ec. (4.4) como

$$L_D = \frac{1}{N_D}\sum (g_{tt}g^{0}_{rr} + g_{rr}g^{0}_{tt})^2 + \frac{1}{N_D}\sum (1 + g_{tt} + rg^{(}_{tt})^2\,, \tag{4.12}$$

en donde la apóstrofe (^{0}) indica una derivada parcial respecto a la coordenada radial r. Para esta implementación, asumiremos que es conocido el parámetro de masa y lo fijaremos en $M = 2$. Tomaremos las condiciones iniciales y de frontera, a partir de las hipótesis de la solución de Schwarzschild, por lo que consideraremos que cuando $r \to \infty$:

$$g_{tt} \to 1, \tag{4.13}$$

$$g_{rr} \to 1. \tag{4.14}$$

Usaremos también algunos datos sintéticos como observaciones, es decir, calcularemos un conjunto de datos a partir de la solución analítica de la Ec. (4.3), y los usaremos para comparar con las predicciones como

$$L_O = \frac{1}{N_O} \sum \left(g_{tt}^{nn} - g_{tt}^{true} \right)^2 + \frac{1}{N_O} \sum \left(g_{rr}^{nn} - g_{rr}^{true} \right)^2, \tag{4.15}$$

Con lo anterior, la PINN tendrá la siguiente estructura (ver Fig. 4.1):

- Tomar valores de la coordenada radial en un dominio definido.
- Hacer una predicción para g_{tt} y g_{rr}.
- Usar las predicciones para calcular la pérdida física de la Ec. (4.12) y la pérdida de las observaciones de la Ec. (4.15).
- Ajustar los parámetros de la red usando el algoritmo SGD.

Los ajustes de hiperparámetros y la arquitectura final se desarrollan a continuación.

4.3. Optimización y resultados de la PINN

En esta sección, implementaremos algunas de las metodologías discutidas en el Capítulo 3 para elegir una configuración adecuada de la red. Comenzaremos haciendo una distinción hasta ahora no explorada, y es que la PINN no está formada por una sola red neuronal con dos neuronas de salida, si no por dos redes independientes con una sola neurona de salida: una red para g_{tt} y una para g_{rr}. Este enfoque puede ser útil cuando el número variables a predecir es mucho mayor que el número de variables de entrada. Además esto brinda mayor flexibilidad, pues las predicciones pueden estar correlacionadas y tener redes separadas permite ajustar mejor los parámetros internos de cada una.
Ahora, buscaremos la arquitectura y los valores óptimos para los hiperparámetros de la PINN, es decir, el número de neuronas por capa, el número de capas, la función de activación y el learning rate. Como vimos en algunas implementaciones anteriores, usar un valor fijo para el learning rate nos puede llevar a situaciones donde la red estanque su aprendizaje, o donde consuma demasiados recursos en la etapa inicial, por ello, implementaremos la metodología del ajuste dinámico del learning rate. Utilizaremos un valor de learning rate inicial de 1×10^{-3}, que es relativamente grande, esto permitirá a la PINN

aprender rápidamente en la etapa temprana del entrenamiento; posteriormente este valor se reducirá a la mitad usando un scheduler que evalúa el comportamiento de la función de costo en las 50 épocas más recientes de entrenamiento, y reduce el learning rate al detectar un incremento en ella.
Para elegir la arquitectura y la función de activación usaremos el método de Grid search CV. Analizaremos cuatro diferentes arquitecturas:

- Perceptrón: 1 capa oculta con 1 neurona.
- Homogénea: 5 capas ocultas con 64 neuronas cada una.
- Descendente: 5 capas ocultas con 256, 128, 64, 32 y 16 neuronas respectivamente.
- Profunda: 50 capas ocultas de 64 neuronas cada una.

Para cada una de estas arquitecturas probaremos cuatro funciones de activación distintas: Tanh, ReLU, LogSigmoid y Softmax. En total, probaremos 16 distintos modelos para esta PINN. Para todos los modelos entrenaremos con la misma cantidad de datos de dominio $N_D = 30$, distribuidos de manera uniforme en el intervalo de r 2 $(10, 30)$[1]. En todos los casos se evaluarán 150 épocas de entrenamiento, y con los resultados se elegirá el modelo más adecuado basado en el puntaje de la validación cruzada y el tiempo de cómputo requerido; dicho modelo podría ser entrenado durante más épocas si se considera necesario.
El objetivo de aplicar el algoritmo de validación durante una cantidad limitada de épocas, es observar la estabilidad y eficiencia de la mayor cantidad de modelos posibles con la menor cantidad de recursos computacionales, pues se espera que el método de ajuste dinámico en el learning rate ayude a que el modelo elegido siga mejorando en sus etapas posteriores.
Generalmente, después de algún número de épocas habrá diferencias entre los modelos, pues las redes neuronales son sensibles a los hiperparámetros. Si en estas primeras épocas de validación no se observan diferencias significativas en el rendimiento de los modelos, se opta por entrenarlos un poco más.
En una primera búsqueda se obtienen los resultados de la Fig. 4.2, de la cual es posible descartar todas las variaciones con el modelo del perceptrón y la función de activación softmax, pues claramente obtienen los peores puntajes (valores de pérdida más grandes). Por la simplicidad del perceptrón, no es una sorpresa que sea la arquitectura con menor rendimiento (como se vio en la implementación de supernovas del Capítulo 2). Ya que el

[1] En esta implementación todas las magnitudes físicas se asumen normalizadas, por lo que la unidades no se expresarán de manera explícita.

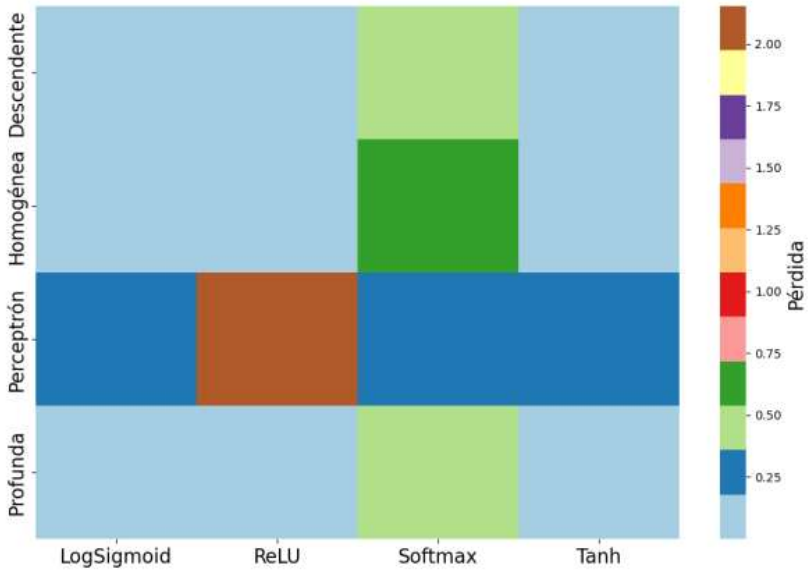


Figura 4.2: Heatmap de los resultados iniciales del Grid Search CV para la PINN. Se observa que el perceptrón tiene los peores resultados en general, y que la función Softmax es la que da peores resultados en todas las arquitecturas. Esta visualización nos permite descartar rápidamente dichas arquitectura y función de activación.

perceptrón y la función softmax presentan resultados tan distintos al resto de los modelos (particularmente el modelo Perceptrón + ReLU, que tiene valor de pérdida al menos 10 veces más alto), actúan como outliers o datos atípicos, y no permiten una correcta visualización y elección de modelo. Por ello, se realiza un nuevo heatmap sin estas variables, que da como resultado la Fig. 4.3. De esta visualización podemos descartar cuatro modelos, que tienen los valores de perdida más altos. Los modelos con red profunda muestran valores de pérdida al menos 8 veces mayores a los cinco mejores modelos. La red homogénea tiene un buen desempeño en todas las funciones de pérdida, lo que muestra que no es tan sensible a este hiperparámetro. La red descendente también tiene buenos resultados, a excepción del modelo Descendente+Tanh. Con estos resultados, una opción es elegir directamente el modelo que presente una menor pérdida promedio, sin embargo, estamos interesados en optimizar también el tiempo de computo, por lo que debemos tener esa variable en cuenta.

En la Fig. 4.4 se visualizan los tiempos de cómputo y las pérdidas con gráficas de barras. Ambas cantidades están normalizadas, es decir, el modelo con el tiempo de cómputo más alto tiene un valor de 1, y de manera similar para el modelo con la pérdida más alta. En la

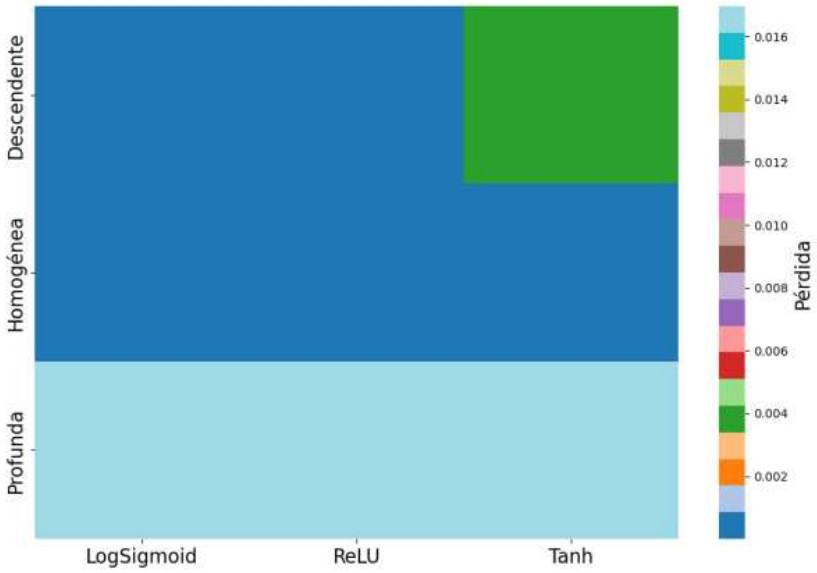
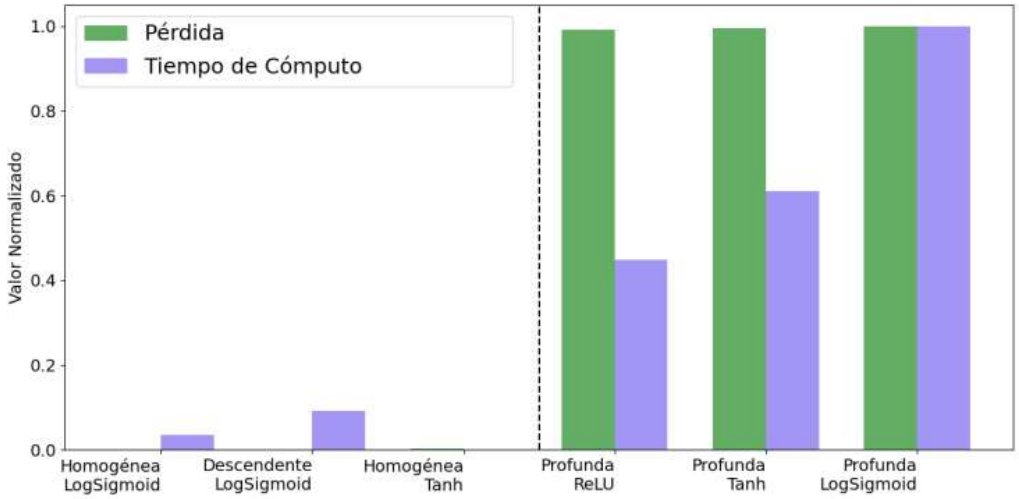
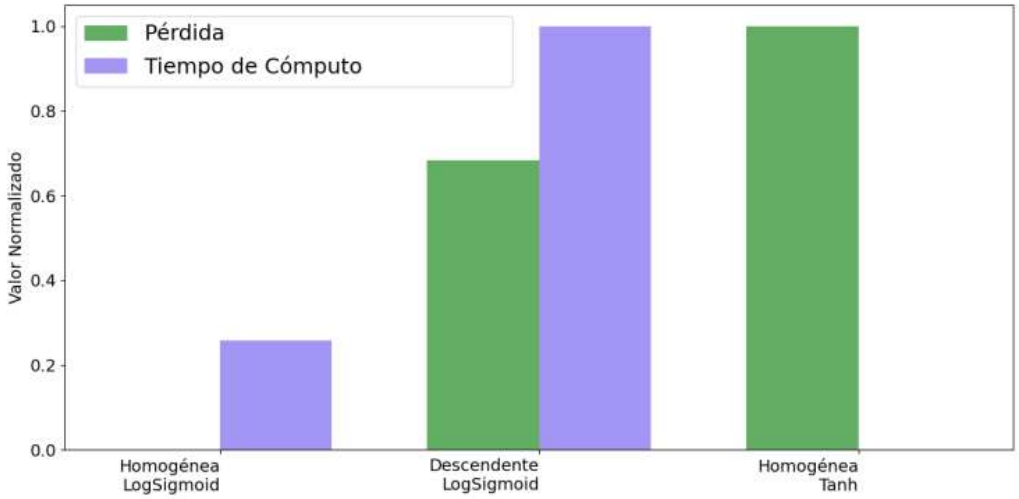


Figura 4.3: Heatmap de los resultados del Grid Search CV para la PINN con 150 épocas de entrenamiento, al descartar el perceptrón y la función softmax. Se observa que la arquitectura Profunda y la configuración Descendente+Tanh tienen los peores resultados. Los cinco modelos restantes son prometedores.

Fig. 4.4a se presentan los tres mejores (a la izquierda de la linea punteada) y los tres peores modelos (a la derecha de la línea punteada) según el valor promedio de la pérdida. De esta figura, se puede observar que la arquitectura Profunda es la que tiene el peor desempeño, no solamente en cuanto a resolver el problema de minimizar la función de pérdida, sino que también es la arquitectura que requiere más tiempo de computo, es decir, más no siempre es mejor. La diferencia de desempeño de esta arquitectura en comparación al resto de modelos, es tan grande que las barras para los mejores modelos son demasiado pequeñas en la visualización y poco prácticas para tomar una decisión. A esta escala podríamos elegir cualquiera de los tres modelos a la izquierda de la línea punteada, pues es difícil describir las diferencias entre ellos, para poder entenderlo mejor, en la Fig. 4.4b se visualizan únicamente los tres modelos con menor pérdida. De aquí, podemos ver que el modelo Homogénea+Tanh es el más veloz en ejecución, pero el que tiene la mayor pérdida. El modelo Descendente+LogSigmoid es el más lento en ejecución, pero mejora alrededor de 30 % el valor de la pérdida en relación al primero. Finalmente, es claro que el mejor modelo es Homogénea+LogSigmoid, pues tiene tanto la menor cantidad de pérdida, como el menor tiempo de ejecución, esto nos permite elegirlo con mayor seguridad de aprovechar



(a)



(b)

Figura 4.4: Tiempo de cómputo y pérdida normalizados, después de 150 épocas de entre-namiento: (a) comparación de los 3 mejores y los 3 peores modelos (izquierda y derecha de la línea punteada respectivamente) y (b) acercamiento de los 3 mejores modelos.

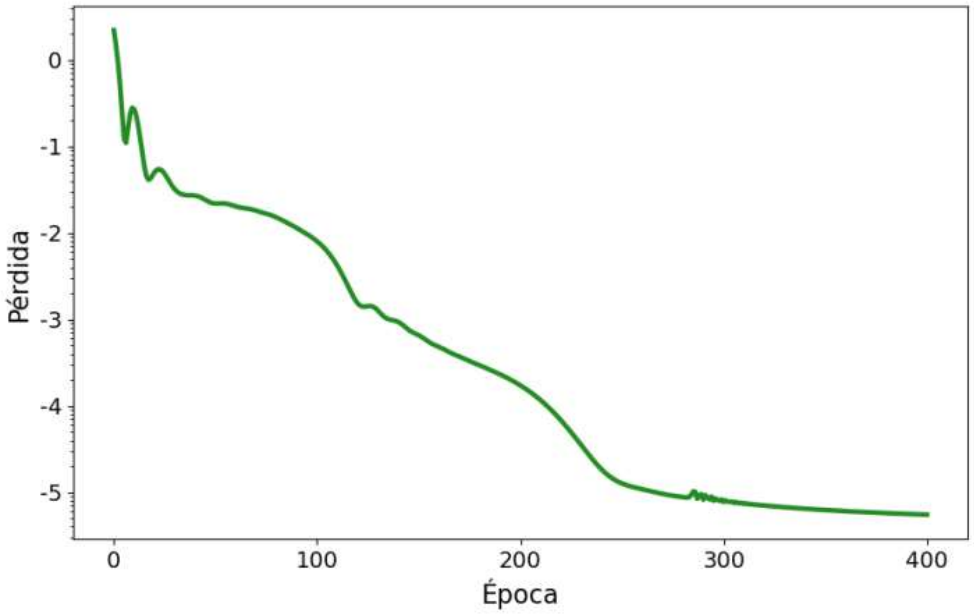


Figura 4.5: Evolución de la función de perdida para la PINN que resuelve las ecuaciones de campo.

de la mejor manera los recursos computacionales y obtener resultados físicos consistentes.

Una vez elegido el modelo Homogénea+LogSigmoid, se ejecuta un entrenamiento más largo, en total 400 épocas. En la Fig. 4.5 se observa la evolución de la función de pérdida, que en general, decrece de manera suave. Se observan pequeños periodos de estancamiento y de saturación, que son superados rápidamente gracias a la implementación dinámica del learning rate. Particularmente, aprende lentamente alrededor de la época 50, y entre las épocas 300 y 400 parece no haber un cambio significativo en el valor de la pérdida. Por ello, en la Fig. 4.6 se visualiza la solución exacta y las predicciones en 50, 150 y 400 épocas. En la Fig. 4.6a se tienen las predicciones para g_{tt}, y se puede ver que a 50 y 150 la predicción tiene un comportamiento lineal. En comparación, las predicciones para g_{rr} de la Fig. 4.6b tienen un comportamiento lineal tras 50 épocas, pero comienza a tener buenas predicciones en 150 épocas. La PINN obtiene muy buenos resultados para ambas componentes de la métrica después de 400 épocas. Finalmente, hemos construido una PINN que reproduce la solución de Schwarzschild para las ecuaciones de campo de relatividad general.

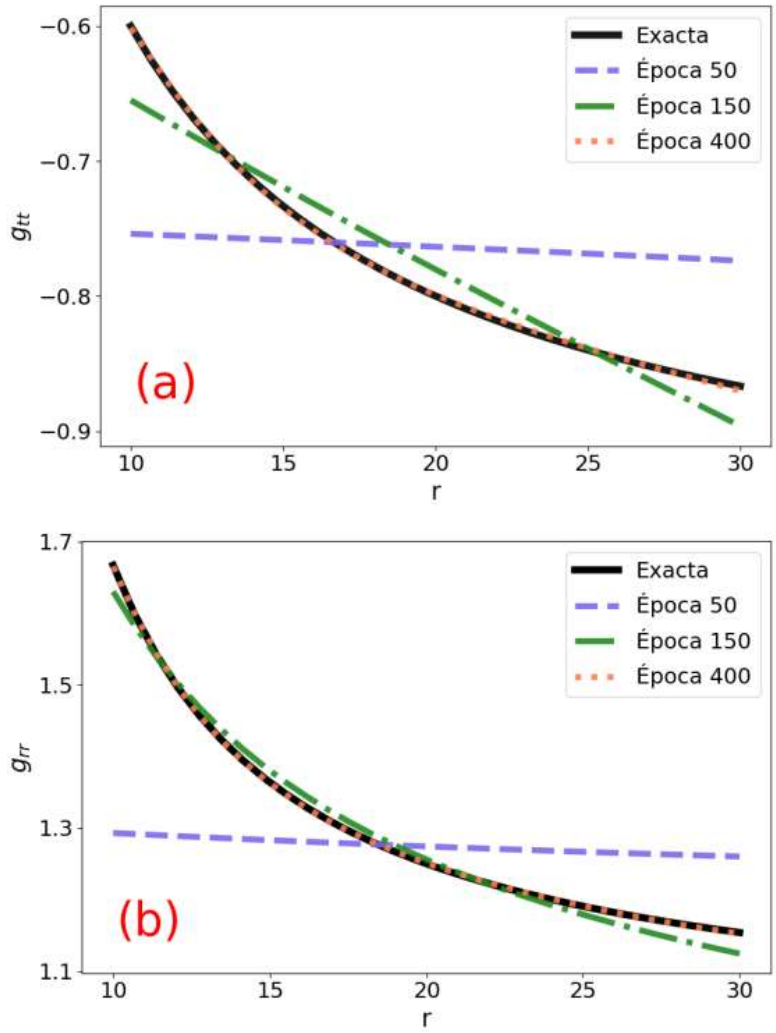


Figura 4.6: Resultados de la PINN a diferentes épocas: (a) resultados para la componente g_{tt}, y (b) resultados para la componente g_{rr}.

4.4. Trabajo futuro

La PINN que hemos construido en este capítulo ha encontrado la solución de Sch-warzschild de manera exitosa, sin embargo, la intención de estas redes no es simplemente reproducir soluciones de problemas conocidos, por el contrario, pretenden ser una herra-mienta útil para enfrentar sistemas físicos con soluciones analíticas desconocidas.

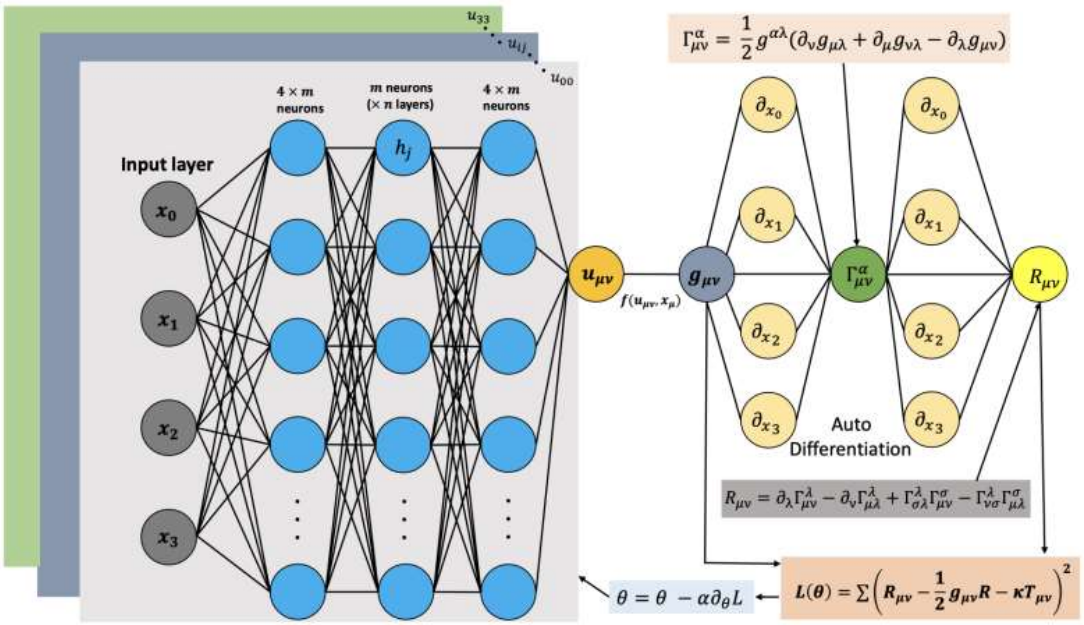


Figura 4.7: Estructura de una PINN para solucionar de manera general las ecuaciones de campo de Einstein [5]. Se conforma de 16 redes neuronales, cada una asociada a la predicción de una componente de la métrica. Luego, se usan las predicciones de cada red para construir los símbolos de Christo↵el $\Gamma^{\alpha}_{\mu\nu}$ y el tensor de Ricci $R_{\mu\nu}$ para construir las ecuaciones de campo como función de pérdida.

Particularmente, se puede extender esta última PINN a un modelo más robusto, que busque soluciones generales de las ecuaciones de campo en las que las suposiciones de las condiciones físicas sean mínimas. Se puede pensar entonces en construir una PINN como en la Fig. 4.7, donde el objetivo es que la red reproduzca las componentes de la métrica, teniendo como características de entrada las coordenadas espacio-temporales (x_α) con $\alpha = 0, 1, 2, 3$ y el valor del tensor de energía-momento $T_{\mu\nu}$. Luego las ecuaciones de campo en su totalidad formarían la función de costo física:

$$L_{Einstein} = \frac{1}{N} \sum \left(R_{\mu\nu} - \frac{1}{2} g_{\mu\nu} R + \Lambda g_{\mu\nu} - \frac{8\pi G}{c^4} T_{\mu\nu} \right)^2, \tag{4.16}$$

Esta arquitectura propuesta en [5], es capaz de encontrar la solución de Reissner-Nordstrom, que corresponde a un agujero negro con carga eléctrica. La complejidad de implementar esta red radica en construir cada una de las combinaciones de derivadas requeridas para escribir las ecuaciones en su forma tensorial, pues se pueden encontrar problemas como el desvanecimiento de gradiente [33] o una mayor dificultad en el ajuste de hiperparámetros.

Capítulo 5

Conclusiones

"No estoy discutiendo, sólo estoy explicando por qué tengo razón".
— Sheldon Cooper, The Big Bang Theory

Esta tesis se enfocó en la implementación de métodos de Machine Learning a la física, y en la comparación de estas metodologías con técnicas computacionales tradicionales. Particularmente, se implementaron Redes Neuronales Informadas en Física (PINNs) para abordar problemas complejos, como las ecuaciones de campo de la relatividad general. Aplicamos distintos métodos de visualización y clusterización de datos. Se resalta el es-tudio de la distribución de tamaños de gotas de un estornudo humano, donde el método de Kernel Density Estimation (KDE) aplicado a las simulaciones nos permite entender mejor la naturaleza de estas distribuciones.
Además, implementamos PINNs capaces de resolver problemas usuales de la física como la caída libre, el péndulo simple y el movimiento de proyectil. También desarrollamos una PINN capaz de resolver las ecuaciones de campo de Einstein, y exploramos diversas com-binaciones de parámetros, obteniendo que una red de 5 capas con 64 neuronas en cada una, y la función de activación LogSigmoid se reduce el tiempo de computo y mejora la calidad de las soluciones.
Las múltiples implementaciones y análisis realizados a lo largo de este trabajo, muestran que la AI, y herramientas específicas como las PINNs representan una manara innovadora y eficiente para abordar problemas complejos en física aplicada, y que por su capacidad para integrar datos y leyes físicas se posicionan como una alternativa prometedora a los métodos de computación tradicionales. Específicamente, dichos resultados enmarcan cier-tas líneas de investigación a futuro, tales como: i) el estudio de las gotas de un estornudo con la metodología de las PINNs, con el objetivo de mejorar significativamente los tiem-

pos de cómputo, ii) desarrollar PINNs más complejas, dedicadas a buscar soluciones más generales de las ecuaciones de campo o iii) usar PINNs para completar datos faltantes en catálogos experimentales u observacionales.

Apéndice A

Caracterización de un estornudo en Basilisk

En este Apéndice se describen los detalles de las simulaciones presentadas en el Capítulo 1 de la atomización de un líquido, utilizando los parámetros adecuados que se asemejen a un estornudo humano. Para este estudio se utilizó el código hidrodinámico Basilisk, que es un software libre para la solución de ecuaciones diferenciales parciales en mallas adaptativas[1].

Uno de los problemas más relevantes que hemos enfrentado como sociedad en épocas recientes, es la pandemia causada por el COVID-19. Se entiende que en este tipo de infecciones, el rango de contagio de un estornudo está relacionado directamente con el tamaño de las gotas, pero aun no existe un consenso sobre la distribución del tamaño de las mismas. La mayoría de las mediciones del tamaño de gotas se centran en experimentos con líquidos tintados, o fotografía de alta velocidad. Sin embargo, se entiende que los estornudos son un fenómeno de flujos multifase y turbulentos, en el que ocurre un proceso de fragmentación previo a la aparición de gotas [13] por lo que resulta interesante estudiarlos desde la mecánica de fluidos.

Parámetros físicos de un estornudo

En un estornudo, los fluidos están compuestos principalmente de agua (97 %) y la cantidad restante se divide en sales, proteínas, ácidos grasos y otros agentes como los ~~virus [34], de modo que en términ~~nos prácticos podemos usar las propiedades físicas del

[1] Disponible en http://basilisk.fr

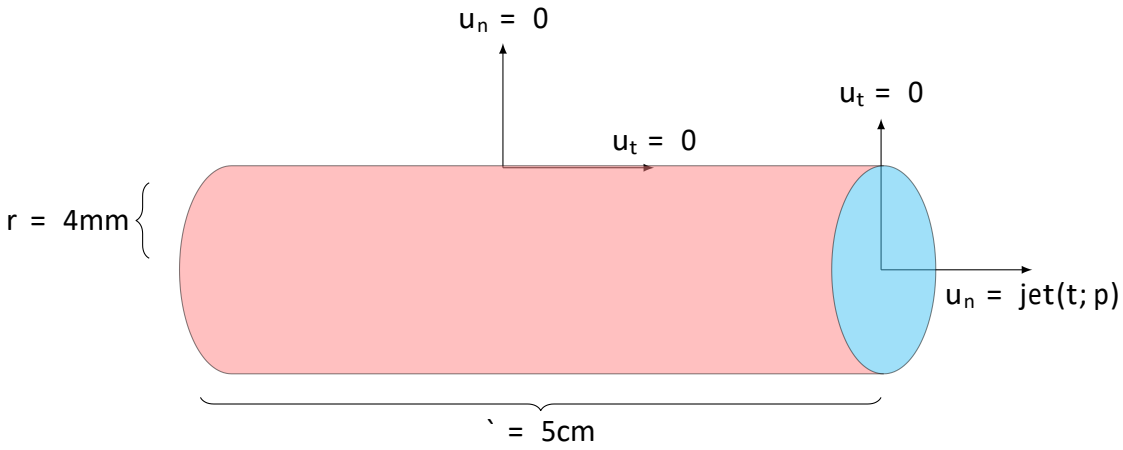


Figura A.1: Representación de la naríz como un cilindro y sus condiciones en la frontera, donde u_t y u_n se refieren a las componentes tangenciales y normales la velocidad del flujo, respectivamente.

agua. Supondremos que, el estornudo inyecta fluido en un fase más ligera a través de una boquilla cilíndrica, ver Fig.A.1, y particularmente, fijamos las siguientes características [13]:

- Radio del cilindro: r = 0.004 m
- Longitud del cilindro: l = 0.05 m
- Número de Reynolds: Re = $4 \rightarrow 10^5$
- Coeficiente de tensión superficial: = 0.06 N/m
- Velocidad inicial del flujo: u_0 = 12 m/s
- Velocidad máxima del flujo: u_{max} = 50 m/s
- Densidad del chorro: $\dashrightarrow_{agua}$ = 1000 kg/m^3
- Densidad del ambiente: $\dashrightarrow_{aire}$ = 1 kg/m^3

La función de entrada

Una vez que hemos entendido los parámetros físicos que se adecúan a nuestro caso de estudio, debemos imponer las condiciones de frontera apropiadas. A través de la función `dirichlet()`, se definen los valores que debe tomar la velocidad en la superficie del

cilindro. Como en un estornudo el fluido solo tiene una vía de escape (el orificio nasal), consideraremos que el cilindro tenga velocidad nula en las componentes normales y tangenciales de toda su superficie excepto en la tapa, donde se introduce una función jet(t) que controle la velocidad con que se expulsa el chorro, como se indica en la Fig. A.1. A ésta, la llamaremos la función de entrada.

La función de entrada es una característica determinante en la evolución del fluido. Para un estornudo, esta función debe satisfacer características particulares. En [17] se reportan distribuciones del tamaño de las gotas de estornudos a t=100ms, por lo que escogemos esta marca de tiempo para nuestras simulaciones. Además en [14], se reporta que la ve-locidad de los estornudos es variable, y que, en ocasiones excepcionales, se han podido registrar hasta en 50 m/s, alcanzando su velocidad máxima entre los 10 y los 100ms.

A partir de estas características, la función que proponemos es:

$$\text{jet}(t;p) = u_0 + \frac{1}{N}\exp\left[\frac{\sin(p\hat{\imath}(t \quad p^2))}{p\hat{\imath}(t \quad p^2)} + \frac{\sin(p\hat{\imath}(t \quad p^2))}{p\hat{\imath}(t \quad p^2)}\right], \qquad (A.1)$$

en la cual, u_0 corresponde a la velocidad inicial del flujo, N es una constante de normalización que nos permite ajustar la velocidad máxima del estornudo y p es un parámetro numérico que controla la amplitud del pulso, y su desplazamiento. Para promover las inestabilidades del flujo, incluiremos una pequeña oscilación en la función de entrada, sumando la función 2sin(↵t), y en este caso eligiendo ↵ = 2000, obteniendo la Fig. A.2, donde se observa la familia elegida para la simulación, con tres valores distintos del parámetro p.

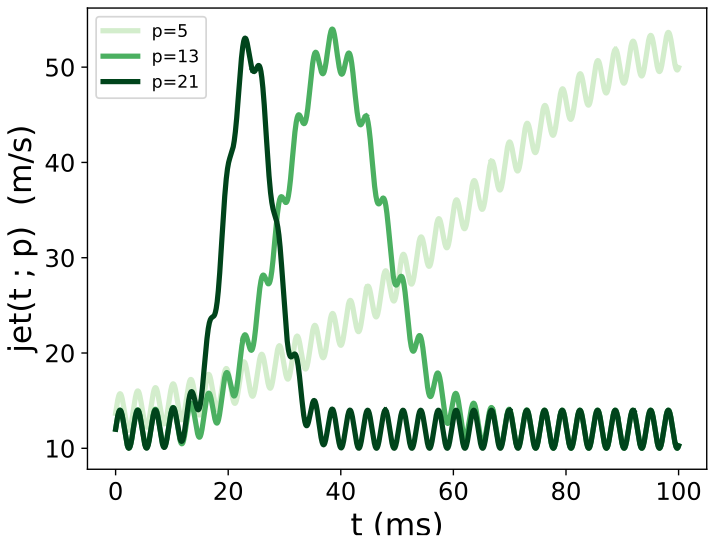


Figura A.2: Familia de funciones de entrada de un estornudo con perturbación de la Ecuación (A.1) .

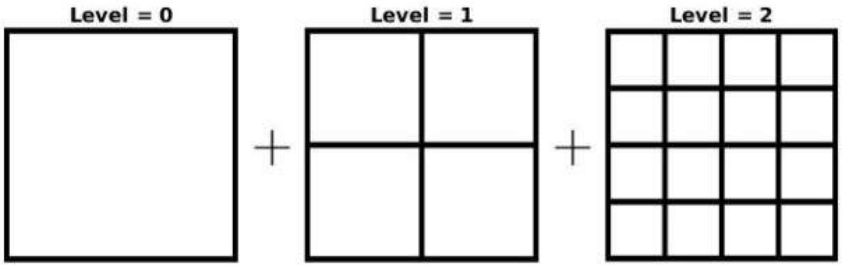


Figura A.3: Niveles del malleo adaptativo

Multigrid y análisis espacial

Basilisk utiliza un enfoque de mallas adaptativas (grids), que permite ganar resolución en las regiones del dominio donde necesitemos más detalle (en nuestro caso, cerca del chorro), al mismo tiempo que se ahorra tiempo de cómputo en las regiones donde no hay actividad relevante.

El método de la discretización del mallado es el siguiente: si tenemos una celda inicial c_0 de tamaño l_0, se considera en el nivel 0. Una vez que en c_0 se cumplen las condiciones de refinamiento, se dividirá en 2^n celdas simétricas c_1 (donde n es el número de dimensiones del problema) consideradas celdas de nivel 1. Secuencialmente, si en alguna celda c_1 se cumplen las condiciones de refinamiento, se dividirá en celdas c_2 de nivel 2, y así en lo sucesivo hasta alcanzar el nivel máximo establecido en el parámetro maxlevel. Para una malla de dos dimensiones, esto puede ilustrarse mejor con la Fig. A.3.

A su vez, el algoritmo de conteo de gotas es el siguiente: para cada celda se calcula la fracción f del fluido en ella. Si esta fracción es superior a un límite establecido, entonces se considera una gota. Además, el numero inicial de celdas es importante, ya que el malleo adaptativo se aplica en una región y no en todo el dominio, por lo que este parámetro puede afectar el resultado final, como se ve en la Fig. A.4, donde se comparan 3 distintas mallas iniciales.

Para elegir los parámetros adecuados, conviene saber qué tipo de gotas esperamos ver en nuestra simulación. Se sabe de [16] que el tiempo de evaporación para gotas de agua es:

$$\boxtimes_{ev} = \frac{R_0^2}{\checkmark(1\ R_H)}\ , \qquad (A.2)$$

donde R_0 es el radio inicial de una gota, $\checkmark = 4.2 \rightarrow 10^2\ \mu m^2/s$ es una constante con unidades de difusión y R_H es la humedad relativa del ambiente.

Por otro lado, se tiene que el tiempo de caída o sedimentación está dado por la ecuación:

$$\boxtimes_{sed} = \frac{z_0}{R^2}\ , \qquad (A.3)$$

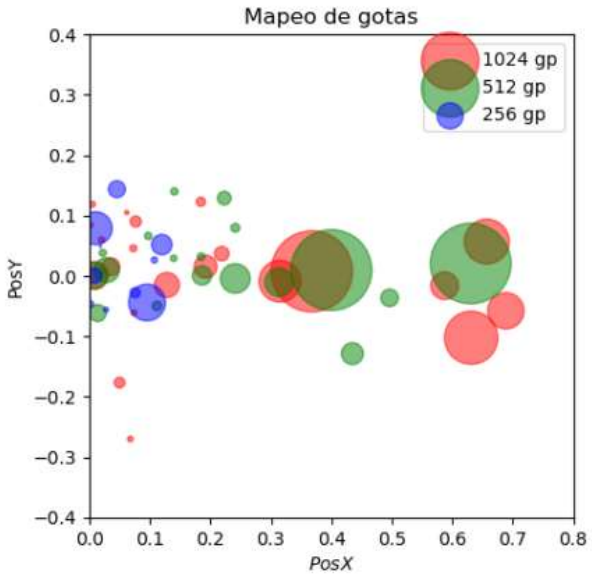


Figura A.4: Comparación de usar diferentes valores de gridpoints como malla inicial para el ejemplo original de Basilisk, el tamaño de los puntos esta en relación al volumen de las gotas. Debido a la falta de resolución en la malla de 256 gridpoints, se observan pocas gotas y de gran volumen, en comparación a la malla de 1024 gridpoints donde aumenta la cantidad y disminuye el volumen general de las gotas.

donde = 0.85 ⇥ 10 ^{2} µm · s es una constante relacionada con la gravedad, viscosidad y densidad del agua, z_0 = 1.5 m es la altura promedio en la que se sitúa la boca humana respecto al suelo y R es el radio de la gota.
De estas dos ecuaciones pueden calcularse cotas para los posibles radios observados en un instante dado. En particular, para nuestro tiempo de interés, t = 100ms, se tiene que los posibles radios están en el intervalo:

$$5\mu m < R_{posible} < 400\mu m \ . \qquad (A.4)$$

Esto permite elegir una combinación de parámetros que se ajuste a las capacidades de cómputo y que garantizan observar las gotas con suficiente refinamiento. En nuestro caso, para un dominio de 0.5 ⇥ 0.5 ⇥ 0.5 metros cúbicos, se eligió una malla inicial de 256 gridpoints, 10 niveles de refinamiento y una fracción de flujo f = 10 ^{9}, lo que finalmente permitió obtener los resultados del Capítulo 1, tras 6 ⇥ 10^4 minutos de cómputo.

Apéndice B

Símbolos de Christoffel para la métrica de Schwarzschild

En este Apéndice, vamos a calcular todos los casos de los símbolos de Christoffel para la forma general de la métrica de Schwarzschild. Los términos no nulos se usan en la determinación de la forma explícita de dicha métrica.
Será útil recordar que en este punto:

$$g_{tt} = A(r), \tag{B.1}$$

$$g_{rr} = B(r), \tag{B.2}$$

$$g_{\theta\theta} = r^2, \tag{B.3}$$

$$g_{\phi\phi} = r^2 \mathrm{sen}^2\theta. \tag{B.4}$$

Otro dato a tener en cuenta es que la métrica no cambia en el tiempo $\partial_t g_{\mu\nu} = 0$ y que está diagonalizada $g_{\mu\nu} = 0$ para $\mu = \nu$ (y análogamente para $g^{\mu\nu}$). De modo que hay que eliminar siempre las derivadas parciales en t y los términos cruzados en la métrica, así como renombrar los índices adecuados para diagonalizar la métrica, como se muestra a continuación:

$$\Gamma^t_{tt} = \Gamma^t_{tt} = \frac{1}{2}g^{t\mu}(\partial_t g_{\mu t} + \partial_t g_{t\mu} - \partial_\mu g_{tt}) = \frac{1}{2}g^{tt}(0 + \partial_t g_{tt} - 0) = 0 \tag{B.5}$$

$$\Gamma^r_{rt} = \Gamma^r_{tr} = \frac{1}{2}g^{r\mu}(\partial_t g_{\mu r} + \partial_r g_{\mu t} - \partial_\mu g_{rt}) = \frac{1}{2}g^{rt}(0 + \partial_r g_{tt} - 0) = 0 \tag{B.6}$$

De manera análoga, para el resto obtenemos:

$$\Gamma^{r}_{rr} = \frac{1}{2}g^{r\mu}(\partial_r g_{\mu r} + \partial_r g_{\mu r} - \partial_\mu g_{rr}) = \frac{1}{2}g^{rr}\partial_r g_{rr} = \frac{1}{2B}\partial_r B \tag{B.7}$$

$$\Gamma^{r}_{r\varphi} = \Gamma^{r}_{r} = \frac{1}{2}g^{r\mu}(\partial_\varphi g_{\mu r} + \partial_r g_{\mu\varphi} - \partial_\mu g_{r\varphi}) = \frac{1}{2}g^{rr}\partial_\varphi g_{rr} = 0 \tag{B.8}$$

$$\Gamma^{\varphi}_{r\varphi} = \Gamma^{r}_{r} = \frac{1}{2}g^{r\mu}(\partial_\varphi g_{\mu r} + \partial_r g_{\mu\varphi} - \partial_\mu g_{r\varphi}) = \frac{1}{2}g^{rr}\partial_\varphi g_{rr} = 0 \tag{B.9}$$

$$\Gamma^{t}_{rr} = \frac{1}{2}g^{t\mu}(\partial_r g_{\mu r} + \partial_r g_{\mu r} - \partial_\mu g_{rr}) = \frac{1}{2}g^{tt}\partial_t g_{rr} = 0 \tag{B.10}$$

$$\Gamma^{t}_{r\varphi} = \Gamma^{t}_{r} = \frac{1}{2}g^{t\mu}(\partial_\varphi g_{\mu r} + \partial_r g_{\mu\varphi} - \partial_\mu g_{r\varphi}) = \frac{1}{2}g^{tt}(\partial_\varphi g_{tr} + \partial_r g_{t\varphi}) = 0 \tag{B.11}$$

$$\Gamma^{t}_{tt} = \frac{1}{2}g^{t\mu}(\partial_t g_{t\mu} + \partial_t g_{t\mu} - \partial_\mu g_{tt}) = \frac{1}{2}g^{tt}(-\partial_t g_{tt}) = 0 \tag{B.12}$$

$$\Gamma^{t}_{tr} = \Gamma^{t}_{rt} = \frac{1}{2}g^{t\mu}(\partial_t g_{r\mu} + \partial_r g_{t\mu} - \partial_\mu g_{tr}) = \frac{1}{2}g^{tt}(\partial_r g_{tt}) = \frac{1}{2}\frac{1}{A}\partial_r A \tag{B.13}$$

$$\Gamma^{t}_{t\varphi} = \Gamma^{t}_{t} = \frac{1}{2}g^{t\mu}(\partial_t g_{\varphi\mu} + \partial_\varphi g_{t\mu} - \partial_\mu g_{t\varphi}) = \frac{1}{2}g^{tt}\partial_\varphi g_{tt} = 0 \tag{B.14}$$

$$\Gamma^{t}_{r} = \Gamma^{t}_{r} = \frac{1}{2}g^{tt}(\partial_r g_{t} + \partial_r g_{rt} - \partial_t g_{r}) = 0 \tag{B.15}$$

$$\Gamma^{t}_{\varphi\varphi} = \frac{1}{2}g^{tt}(\partial_\varphi g_{\varphi t} + \partial_\varphi g_{\varphi t} - \partial_t g_{\varphi\varphi}) = 0 \tag{B.16}$$

$$\Gamma^{t}_{\varphi} = \Gamma^{t}_{\varphi} = \frac{1}{2}g^{tt}(\partial_\varphi g_{t} + \partial_\varphi g_{\varphi t} - \partial_t g_{\varphi}) = 0 \tag{B.17}$$

$$\Gamma^{t}_{t} = \frac{1}{2}g^{tt}(\partial_t g_{t} + \partial_t g_{t} - \partial_t g_{t}) = 0 \tag{B.18}$$

$$\Gamma^{r}_{tt} = \frac{1}{2}g^{rr}(\partial_t g_{tr} + \partial_t g_{tr} - \partial_r g_{tt}) = \frac{1}{2}g^{rr}(-\partial_r g_{tt}) = \frac{1}{2}\frac{1}{B}\partial_r A \tag{B.19}$$

$$\Gamma^{r}_{t\varphi} = \Gamma^{r}_{\varphi t} = \frac{1}{2}g^{rr}(\partial_t g_{\varphi r} + \partial_\varphi g_{tr} - \partial_r g_{t\varphi}) = 0 \tag{B.20}$$

$$\Gamma^{r}_{t} = \Gamma^{r}_{t} = \frac{1}{2}g^{rr}(\partial_t g_{r} + \partial_t g_{tr} - \partial_r g_{t}) = 0 \tag{B.21}$$

$$\Gamma^{r}_{\varphi\varphi} = \frac{1}{2}g^{rr}(\partial_\varphi g_{\varphi r} + \partial_\varphi g_{\varphi r} - \partial_r g_{\varphi\varphi}) = \frac{1}{2}g^{rr}\partial_r g_{\varphi\varphi} = \frac{r}{B} \tag{B.22}$$

$$\Gamma^{r}_{\varphi} = \Gamma^{r}_{\varphi} = \frac{1}{2}g^{rr}(\partial_\varphi g_{r} + \partial_\varphi g_{\varphi r} - \partial_r g_{\varphi}) = 0 \tag{B.23}$$

$$\Gamma^{r}_{} = \frac{1}{2}g^{rr}(\partial_ g_{r} + \partial_ g_{r} - \partial_r g_{}) = \frac{1}{2}g^{rr}\partial_r g_{} = \frac{r}{B}\mathrm{sen}^2\varphi \tag{B.24}$$

$$\Gamma^{\checkmark}_{tt} = \frac{1}{2} g^{\checkmark\checkmark} (\partial_t g_{t\checkmark} + \partial_t g_{t\checkmark} - \partial_{\checkmark} g_{tt}) = 0 \qquad \text{(B.25)}$$

$$\Gamma^{\checkmark}_{tr} = \Gamma^{\checkmark}_{rt} = \frac{1}{2} g^{\checkmark\checkmark} (\partial_r g_{r\checkmark} + \partial_r g_{t\checkmark} - \partial_{\checkmark} g_{tr}) = 0 \qquad \text{(B.26)}$$

$$\Gamma^{\checkmark}_{t\checkmark} = \Gamma^{\checkmark}_{\checkmark t} = \frac{1}{2} g^{\checkmark\checkmark} (\partial_t g_{\checkmark\checkmark} + \partial_{\checkmark} g^{\checkmark}{}_t - \partial_{\checkmark} g_{\checkmark t})$$

$$= 0 \qquad \text{(B.27)}$$

$$\Gamma^{\checkmark}_{t} = \Gamma^{\checkmark}_{t} = \frac{1}{2} g^{\checkmark\checkmark} (\partial_t g_{\checkmark} + \partial g_{t\checkmark} - \partial_{\checkmark} g_{t})$$

$$= 0 \qquad \text{(B.28)}$$

$$\Gamma^{\checkmark}_{r\,r} = \frac{1}{2} g^{\checkmark\checkmark} (\partial_r g_{r\checkmark} + \partial_r g_{r\checkmark} - \partial_{\checkmark} g_{rr}) = 0 \qquad \text{(B.29)}$$

$$\Gamma^{\checkmark}_{r\checkmark} = \Gamma^{\checkmark}_{\checkmark r} = \frac{1}{2} g^{\checkmark\checkmark} \partial_r g_{\checkmark\checkmark} = \frac{1}{2} \frac{1}{r^2} (2r) = \frac{1}{r} \qquad \text{(B.30)}$$

$$\Gamma^{\checkmark}_{\checkmark\checkmark} = \frac{1}{2} g^{\checkmark\checkmark} \partial_{\checkmark} g_{\checkmark\checkmark} = 0 \qquad \text{(B.31)}$$

$$\Gamma^{\checkmark}_{\checkmark} = \Gamma^{\checkmark}_{\checkmark} = \frac{1}{2} g^{\checkmark\checkmark} \partial g_{\checkmark\checkmark} = 0 \qquad \text{(B.32)}$$

$$\Gamma^{\checkmark}_{\checkmark r} = \Gamma^{\checkmark}_{r} = \frac{1}{2} g^{\checkmark\checkmark} (\partial g_{\checkmark r} + \partial_r g_{\checkmark} - \partial_{\checkmark} g_{r}) = 0 \qquad \text{(B.33)}$$

$$\Gamma^{\checkmark}_{\checkmark} = \frac{1}{2} g^{\checkmark\checkmark} \partial_{\checkmark} g = \frac{1}{2} \frac{1}{r^2} \partial_{\checkmark} r^2 \operatorname{sen}^2 \checkmark = \operatorname{sen} \checkmark \cos \checkmark \qquad \text{(B.34)}$$

$$\Gamma_{tt} = \frac{1}{2} g (\partial_t g_{t} + \partial_t g_{t} - \partial g_{tt}) = 0 \qquad \text{(B.35)}$$

$$\Gamma_{tr} = \Gamma_{rt} = \frac{1}{2} g (\partial_r g_{t} + \partial_t g_{r} - \partial g_{tr}) = 0 \qquad \text{(B.36)}$$

$$\Gamma_{t\checkmark} = \Gamma_{\checkmark t} = \frac{1}{2} g (\partial_{\checkmark} g_{t} + \partial_t g_{\checkmark} - \partial g_{t\checkmark}) = 0 \qquad \text{(B.37)}$$

$$\Gamma_{t} = \Gamma_{t} = \frac{1}{2} g (\partial_t g) = 0 \qquad \text{(B.38)}$$

$$\Gamma_{rr} = \frac{1}{2} g (\partial g_{rr}) = 0 \qquad \text{(B.39)}$$

$$\Gamma_{r\checkmark} = \Gamma_{\checkmark r} = \frac{1}{2} g (\partial_r g_{\checkmark} + \partial_{\checkmark} g_{r} - \partial g_{r\checkmark}) = 0 \qquad \text{(B.40)}$$

$$\Gamma_{r} = \Gamma_{r} = \frac{1}{2} g (\partial_r g) = \frac{1}{2} \frac{1}{r^2 \operatorname{sen}^2 \checkmark} \partial_r r^2 \operatorname{sen}^2 \checkmark = \frac{1}{r} \qquad \text{(B.41)}$$

$$\Gamma_{\checkmark\checkmark} = \frac{1}{2} g (\partial_{\checkmark} g_{\checkmark\checkmark} + \partial_{\checkmark} g_{\checkmark\checkmark} - \partial g_{\checkmark\checkmark})$$

$$= \frac{1}{2} \frac{1}{r^2} (\partial_r r^2) = r \qquad \text{(B.42)}$$

$$\Gamma^{\checkmark}_{\checkmark\checkmark} = \frac{1}{2}g^{\checkmark\checkmark}\left(\partial_{\checkmark}g_{\checkmark\checkmark} + \partial_{\checkmark}g_{\checkmark\checkmark} - \partial_{\checkmark}g_{\checkmark\checkmark}\right) = -\frac{1}{2}\frac{1}{r^2\,\mathrm{sen}^2\checkmark}\,2r^2\,\mathrm{sen}\checkmark\cos\checkmark = \cot\checkmark \qquad \text{(B.43)}$$

$$\boxed{\Gamma^{\checkmark}_{\checkmark\checkmark} = \frac{1}{2}g^{\checkmark\checkmark}\left(\partial_{\checkmark}g_{\checkmark\checkmark} + \partial_{\checkmark}g_{\checkmark\checkmark} - \partial_{\checkmark}g_{\checkmark\checkmark}\right) = 0} \qquad \text{(B.44)}$$

Bibliografía

[1] Luis Medrano et.al. Solving di↵erential equations with Deep Learning: a beginner's guide. arXiv e-prints, 2023.

[2] Kazunori Akiyama et al. First M87 Event Horizon Telescope Results. I. The Shadow of the Supermassive Black Hole. Astrophys. J. Lett., 875:L1, 2019.

[3] Kazunori Akiyama et al. First Sagittarius A* Event Horizon Telescope Results. I. The Shadow of the Supermassive Black Hole in the Center of the Milky Way. Astrophys. J. Lett., 930(2):L12, 2022.

[4] M. Naderibeni, M. J. T. Reinders, L. Wu, and D. M. J. Tax. Learning solutions of parametric navier-stokes with physics-informed neural networks, 2024.

[5] Zhi-Han Li, Chen-Qi Li, and Long-Gang Pang. Solving Einstein equations using deep learning. arXiv e-prints, 9 2023.

[6] Shuai Han, Lukas Stelz, Horst Stoecker, Lingxiao Wang, and Kai Zhou. Approaching epidemiological dynamics of COVID-19 with physics-informed neural networks. arXiv e-prints, page arXiv:2302.08796, February 2023.

[7] Chandan K.Reddy Charu C. Aggarwal. Data Clustering: Algorithms and Applications. CRC Press, 2014.

[8] A. M. Pires and J. A. Branco. High dimensionality: The latest challenge to data analysis. arXiv preprint arXiv:1902.04679, 2019.

[9] Ian T. Jolli↵e. Principal Component Analysis. Springer, 2nd edition, 2002.

[10] Bernhard Schölkopf, Alexander Smola, and Klaus-Robert Müller. Nonlinear component analysis as a kernel eigenvalue problem. Neural Computation, 10(5):1299–1319, 1998.

[11] Laurens van der Maaten and Geo↵rey Hinton. Visualizing data using t-sne. Journal of Machine Learning Research, 9:2579–2605, 2008.

[12] Oleg Yu. Tsupko, Zuhui Fan, and Gennady S. Bisnovatyi-Kogan. Black hole shadow as a standard ruler in cosmology. Class. Quant. Grav., 37(6):065016, 2020.

[13] B.E. Scharfman, A.H. Techet, J.W.M. Bush, and et al. Visualization of sneeze ejecta: steps of fluid fragmentation leading to respiratory droplets. Exp Fluids, 57, 24, 2016.

[14] S. Kumar, D. Klassen, M. amd Klassen, R. Hardin, and M. King. Dispersion of sneeze droplets in a meat facility indoor environment - Without partitions. Environmental research, 236, 2023.

[15] Farzad Pourfattah, Lian-Ping Wang, Weiwei Deng, Yongfeng Ma, Liangquan Hu, and Bo. Yang. Challenges in simulating and modeling the airborne virus transmission: A state-of-the-art review. Physics of Fluids., 33, 2021.

[16] Roland R. Netz and William A. Eaton. Physics of virus transmission by speaking droplets. Proc. Natl. Acad. Sci. U.S.A., 117:25209–25211, 2020.

[17] Q. Y. Huang Zhuyang Han, Wenguo Weng. Characterizations of particle size distribution of the droplets exhaled by sneeze. Journal of the Royal Society Interface, 10, 2013.

[18] J. Carrasquilla and R. Melko. Machine learning phases of matter. Nature Phys, 13:431–434, 2017.

[19] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. Deep Learning. MIT Press, 2016. http://www.deeplearningbook.org.

[20] Kevin P. Murphy. Machine learning : a probabilistic perspective. MIT Press, 2012.

[21] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. Nature, 521:436–444, 2015.

[22] César Menacho. Lineal regression models with neural networks. Anales Científicos, 75 (2): 253-260 (2014), 2014.

[23] G. Cybenko. Approximation by superpositions of a sigmoidal function. Mathematics of Control, Signals and Systems, 2(4):303 –314, 1989.

[24] David M. W. Powers. Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation. arXiv e-prints, page arXiv:2010.16061, October 2020.

[25] Scolnic et.al. The complete light-curve sample of spectroscopically confirmed sne ia from pan-starrs1 and cosmological constraints from the combined pantheon sample. The Astrophysical Journal, 859(2):101, May 2018.

[26] David R. Soderblom. The ages of stars. Annual Review of Astronomy and Astrophysics, 48:581–629, 2010.

[27] Eric E. Mamajek and Lynne A. Hillenbrand. Improved age estimation for solar-type dwarfs using activity-rotation diagnostics. The Astrophysical Journal, 687(2):1264–1293, 2008.

[28] John C Snyder, Matthias Rupp, Katja Hansen, Klaus-Robert Müller, and Kieron Burke. Machine learning for molecular and materials science. Nature materials, 12(3):194–201, 2012.

[29] Maziar Raissi, Paris Perdikaris, and George E Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial di↵erential equations. Journal of Computational Physics, 378:686–707, 2019.

[30] John D. Anderson. Computational Fluid Dynamics: The Basics with Applications. McGraw-Hill, New York, 1995.

[31] George E. Karniadakis, Ioannis G. Kevrekidis, Lu Lu, Paris Perdikaris, Sifan Wang, and Liu Yang. Physics-informed machine learning. Nature Reviews Physics, 3(6):422–440, 2021.

[32] Roberto Casadio, Alexander Kamenshchik, and Jorge Ovalle. Cosmology from schwarzschild black hole revisited. arXiv preprint arXiv:2407.14130, 2024.

[33] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. Neural computation, 9(8):1735–1780, 1997.

[34] Turner BS Bansil R. Mucin structure, aggregation, physiological functions and biomedical applications. Curr Opin Colloid Interface, 11:164–170, 2006.