eman ta zabal zazu
Universidad del País Vasco
Euskal Herriko Unibertsitatea

MOLECULAR DESCRIPTORS AND BIG DATA ANALYSIS TO EXTRAPOLATE MATERIALS PROPERTIES

Carlos Manuel de Armas Morejón

A thesis submitted for the degree of Doctor of Philosophy

Supervisor:
Prof. Dr. Ángel Rubio
Dr. Joaquim Jornet-Somoza

Marzo 2023
DEPARTAMENTO DE POLÍMEROS Y MATERIALES AVANZADOS: FISICA, QUÍMICA Y TECNOLOGIA
UPV/EHU

Agradecimientos.

Quisiera agradecer a todo el grupo de ETSF, a los que ya encontraron un nuevo rumbo y a los nuevos, que tengan la misma suerte que tuve yo. A los miembros del LQCT de la Facultad de Química de la Universidad de la Habana que me acogieron y me dieron esta oportunidad.

Quisiera agradecer a Cecilia, que modestamente pone *Técnico de Gestión de Investigación* en sus correos, pero en realidad posee un Phd en el oscuro arte de la burocracia.

A los profesores Ángel y Luis, por su invaluable experiencia, que pusieron a mi disposición sin dudarlo.

Quisiera agradecer también a Ask, por su paciencia y a Quim, gracias por por el increíble esfuerzo.

A mis dos niñas, que vieron crecer este trabajo.

Abreviaturas extraídas del inglés.

GS	Estado fundamental
ML	Aprendizaje Automático
DFT	Teoría del Funcional de la Densidad
TD-DFT	Teoría del Funcional de la Densidad Dependiente del Tiempo
KS	Kohn - Sham
HF	Hartree - Fock
eq.	Ecuación
ej.	Ejemplo

Resumen.

En este trabajo se explora una selección de *descriptores* moleculares que incluyan información tanto espacial como electrónica. Haciendo uso de estos descriptores se propone simplificar operaciones complejas como la *Optimización de Geometrías* y la obtención del *Espectro de Absorción*.

Se ha estudiado la validez de los nuevos descriptores utilizando distintos modelos de aprendizaje automático como el *Regresión Ridge con Kernel* y las *Redes Neuronales*.

Se demuestra que el uso del descriptor e-State es suficiente para representar el entorno atómico. Utilizando una unidad de representación para el entorno de un átomo denominada *bloque* se predicen las coordenadas de los átomos mas cercanos. Pero se necesita más información para la correcta propagación de los enlaces moleculares.

Los resultados que se presentan avalan la utilización de propiedades obtenidas de cálculos del Estado Fundamental para la predicción de propiedades de Estados Excitados. Las Convolutional Neural Network son capaces de emular el salto complejo entre funcionales de correlación e intercambio.

Abstract.

In this work a selection of *descriptors* is explored molecular data that include both spatial and electronic information. Using these descriptors it is proposed to simplify operations complex such as *Geometry Optimization* and obtaining the *Absorption Spectrum*.

The validity of the new descriptors has been studied using different machine learning models such as *Regression Ridge with Kernel* and *Neural Networks*.

It is shown that the use of the e-State descriptor is sufficient to represent the atomic environment. Using a representation unit for the environment of an atom named *block* the coordinates of the closest atoms. But more information is needed for the correct propagation of molecular bonds.

The results presented support the use of properties obtained of calculations

of the Ground State for the prediction of properties of Excited States. The Convolutional Neural Networks are capable of emulating the complex jump between correlation and exchange functionalities.

Contents

Capítulo 1

Introducción.

El uso de métodos computacionales para realizar predicciones ha cobrado mucha fuerza en años recientes. La *Teoría del Funcional de la Densidad*[1] (del inglés *Density Functional Theory* DFT) y *Teoría del Funcional de la Densidad Dependiente del Tiempo* (del inglés *Time Dependent Density Functional Theory* TD-DFT) se han convertido en las herramienta por excelencia para realizar cálculos computacionales tanto en el campo de la Física como de la Química. La siempre creciente necesidad de obtener resultados precisos de sistemas reales demanda de grandes super-ordenadores.

Un abaratamiento en general de los componentes de los ordenadores unido a un incremento sustancial en la capacidad de procesamiento, ha propiciado un entorno muy favorable para el desarrollo de los cálculos computacionales. Por muchos años (décadas inclusive) se vienen acumulando cálculos realizados para muchos proyectos e investigaciones. Este volumen de datos empieza a ser considerable y muy heterogéneo, por ejemplo existen cálculos redundantes sobre las mismas moléculas o compuestos realizados con diferentes códigos y por distintos grupos de investigación, existen cálculos sobre muchos compuestos con el mismo *funcional*, etc. A todo esto se le suma también la publicación de grandes bases de datos con millones de compuestos[2] junto al cálculo de algunas de sus propiedades electrónicas.

Dentro de todo este océano de información resulta difícil manejarse. De aquí que nacieran iniciativas para organizar bajo un solo repositorio y de una manera homogénea todos estos datos. Este es el caso de *NOMAD CoE*[1] [3] (*NO*vel *MA*terial *D*iscovery *C*entre *o*f *E*xcellence), proyecto patrocinado por la Unión Europea.

NOMAD CoE busca crear un repositorio común donde se compartan de forma

[1] El lector encontrará una introducción al tema en la Capítulo 2.

[2] El lector puede visitar http://www.quantum-machine.org/datasets/, donde aparecen algunos ejemplos de estas bases de datos de moléculas (última visita 8 de agosto del 2020).

[3] El lector puede visitar https://nomad-lab.eu/ (última visita 8 de agosto del 2020).

gratuita todos los cálculos realizados disponibles dentro de su base de datos. Además, los cálculos son guardados, conservando la mayor cantidad de información posible de como fueron realizados. El volumen de datos almacenados sigue siendo considerable y obtener propiedades electrónicas, para los sistemas que no la poseen o simplemente son incompletas, usando DFT o TDDFT resulta en una tarea titánica.

Recientemente los algoritmos de *Aprendizaje Automático*[4] (del inglés Machine Lerarning, ML) han reaparecido con fuerza. Apoyándose en el uso de superordenadores han encontrado un nicho de aplicaciones en el análisis del enorme volumen de datos que genera la sociedad moderna.

La aplicación de ML sobre cálculos de DFT y TDDFT resulta en una poderosa herramienta. Busca acortar los tiempos de respuesta en los cálculos DFT y TDDFT, encontrar correlaciones entre propiedades, identificar compuestos con posibles aplicaciones en otros campos y en general extraer información relevante dentro de millones de valores. Algo que no resulta trivial y que enriquece los cálculos de DFT y TDDFT.

El aprendizaje automático se vienen aplicando sistemáticamente en toda rama de la Química y Física. De la literatura consultada se pueden extraer cuatro componentes/enfoques fundamentales:

- Tipo de método utilizado para el aprendizaje: destacan, (1) basados en procesos Gausianos y Máquinas de Vectores de Soporte (SVM por sus siglas en Inglés) [2, 3, 4, 5, 6, 7, **?**] y (2) Redes Neuronales [8, 9, 10, 11].

- Tipo de base de datos de aprendizaje: Se pueden apreciar dos tipos, (1) los que utilizan múltiples (pueden llegar a miles) conformaciones del mismo compuesto [12, 13, 14, 15], (2) los que utilizan múltiples compuestos diferentes [16, 4, 9, 17].

- Tipo de representación molecular (*Descriptor*): destaca el uso de información extraída de la *Tabla Periódica de Elementos Químicos*, tal como número atómico, electrones en orbitales s y/o p, además de cálculos de DFT, entre otros muchos. Esta resulta ser un área muy variada pero como factor común se encuentra, la búsqueda de simplicidad [18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29].

- Tipo de propiedades objetivas: principalmente los enfoques van a toda propiedad atómica que necesite un cálculo DFT o TD-DFT para ser obtenida. Existe un marcado esfuerzo en llevar cálculos de bajo coste computacional a otros de mejor precisión sin la tediosa necesidad de realizarlos [4, 30, 31, 32]. También

[4]El lector encontrará una introducción a al tema en el Capítulo 3.

se han identificado esfuerzos en la búsqueda de estructuras óptimas mediante exploración de las *superficies de energía potencial* [33, 34].

Aunque cada trabajo referido enfoca sus resultados en uno de estos cuatro temas, en general todos los trabajo contienen las cuatros áreas.

Estos avances resaltan el uso de ML sobre datos obtenidos con DFT y TDDFT es un tema con múltiples aplicaciones y ramas abiertas.

La simplicidad guió esta investigación en todo momento. Como método de aprendizaje automático se explora las aplicaciones de SVM y y Redes Neuronales. Para ambos métodos se busca la configuración que, consumiendo la menor cantidad de recursos de computo posible, ofrezca los mejores resultados. Para la entrenamiento se optó por una base de datos compuesta por pequeñas moléculas orgánicas. La selección de propiedades descriptoras se realizó considerando la bibliografía consultada, teniendo en cuenta la mínima información necesaria que permitiera representar una molécula y la posible extrapolación a otros sistemas similares.

Para las propiedades objetivos se seleccionaron características físicas de compuestos químicos, dos temas particulares resaltan la atención del autor: Optimización de Geometrías y Espectro de Absorción.

Para obtener buenos resultados en cualquier cálculo utilizar *Geometrías Óptimas* es importante. La geometría óptima representa donde la energía del compuesto es mínima y las fuerzas entre los átomos están en equilibrio, por tanto su estructura más estable. La obtención de geometrías precisas es un proceso costoso en tiempo y recursos de computo, si se quiere obtener la estructura donde la energía es mínima. Implica realizar cálculos *ab-initio* en múltiples ocasiones hasta obtener el mínimo (se ofrecerá una introducción al tema de optimizaciones de geometrías en el Capítulo 2). En este trabajo se propone un algoritmo para mejorar este proceso usando ML. Partiendo de una Base de Aprendizaje basada en múltiples compuestos, se utiliza un descriptor basado en la conectividad atómica y un método de Regresión para la reconstrucción de los compuestos.

En la segunda parte de este trabajo persigue el uso de métodos de ML para obtener propiedades espectroscópicas con precisión DFT/TDDFT. Específicamente en el espectro de absorción el cual juega un papel central en el análisis de cómo los compuestos reaccionan ante la luz o un láser. Existen varios trabajos que han abordado este tema con anterioridad. Cuando se presenten los resultados (ver Capítulo 6.), se hará un análisis de esta bibliografía ofreciendo al lector perspectiva de comparación. Utilizando una base de datos de aprendizaje basada en múltiples compuestos, se explora la utilización de propiedades electrónicas de cálculos poco costosos y mejorarlos mediante los métodos de Regresión y Redes Neuronales.

Este trabajo está compuesto por siete Capítulos que se distribuyen en: (i) Esta Introducción para ubicar en contexto al lector; (ii) le siguen tres Capítulos

dedicados al marco teórico y técnico relacionado con este trabajo conceptos de Mecánica cuántica y en particular la *Teoría del Funcional de la Densidad* y *Teoría del Funcional de la Densidad Dependiente del Tiempo* (ver Capítulo 2), Métodos de Aprendizaje Automático principalmente las *Máquinas de Vectores de Soporte* y las *Redes Neuronales* (ver Capítulo 3), y Aspectos Técnicos de la Implementación de Software utilizada (ver Capítulo 4); (iii) a continuación dos Capítulos cuyo contenido está dedicado a la descripción de los resultados obtenidos en *optimización de geometrías* (Ver Capítulo 5) y *propiedades espectroscópicas* (ver Capítulo 6); (4) por último la Conclusiones generales del trabajo y Trabajo futuro (ver Capítulo 7).

Capítulo 2

Elementos de Mecánica Cuántica.

... I've walked the surface of the sun
I've seen events so tiny and so fast they
hardly can be said to occurred at all ...
Dr. Manhattan, Watchmen V-12.

Contents

En este capítulo se realizará una visita a los conceptos relevantes para este trabajo sobre Mecánica Cuántica y su aplicación en Química Cuántica. Aunque el Autor considera que el Lector posee conocimientos de Química y/o Física Cúantica, se introducirán primeramente un grupo de conceptos importantes, que ayudan en la comprensión de los temas a tratar :

- $\boldsymbol{x}$: Coordenada de posición y giro (del inglés Spin) (ej. $(r_{x,y,z}, s)$).
- Función de Onda: Expresión matemática que depende de las coordenadas $\boldsymbol{x}$ y representa al sistema de N partículas y contiene toda la información sobre ese sistema en un estado dado. Por tanto todas las propiedades del sistema pueden ser extraídas evaluando la Función de Ondas con un operador apropiado (ej. *Hamiltoniano*). Será representada con la letra griega Ψ.
- Estado Fundamental: Basándose en el energía del sistema, en este trabajo se considerará el *Estado Fundamental* (Del Inglés Ground State, a partir de ahora GS) como el estado donde la energía del sistema es mínima.

- Estados Excitados: Estado en equilibrio en el que se encuentra el sistema donde la Energía total es mayor a la del GS.

- Densidad Electrónica $\rho(\boldsymbol{x})$: Medida de la probabilidad de encontrar un electrón en un punto determinado del espacio. Para un estado del sistema Ψ la densidad electrónica para un electrón se obtiene a través de: $\rho(\boldsymbol{x}) = N \int \Psi(x_1, x_2, ..., x_N)^2 dx_2^3 ... dx_N^3$

- Funcional: Se define *funcional* $F[f(x)]$ a aquella función donde la variable es otra función $f(x)$.

- Hamiltoniano $\hat{H}$: Operador aplicado sobre la función de ondas para medir la energía total del sistema[35]:

$$\hat{H} = \hat{T} + \hat{V} \tag{2.1}$$

  donde la energía cinética para electrones y el núcleo $\hat{T}$ se define como:

$$\hat{T} = -\sum_i \frac{\hbar^2}{2m_e} \nabla_i^2 - \sum_K \frac{\hbar^2}{2m_K} \nabla_K^2 \tag{2.2}$$

  donde los sub-índices e y K representan los electrones y el núcleo respectivamente, m representa la masa, $\hbar$ es la constante reducida de *Plank* y el operador ∇_i^2 se conoce como el *Operador Laplaciano*

  El término $\hat{V}$ representa la energía potencial de interacción entre electrones, electrones-núcleo y núcleo-núcleo. Está definida como:

$$\hat{V} = \sum_i \sum_K \frac{e^2 Z_K}{r_{iK}} + \sum_{i<j} \frac{e^2}{r_{ij}} + \sum_{k<l} \frac{e^2 Z_K Z_l}{r_{Kl}} \tag{2.3}$$

  donde e es la carga del electrón, Z es el número atómico y $r_{a,b}$ es la distancia entre a y b.

Después de expuestos estos conceptos, el Capítulo se organiza de la siguiente forma, primero se verá una introducción a la Teoría del Funcional de la Densidad (*DFT*) y a la Teoría del Funcional de la Densidad Dependiente del Tiempo (*TD-DFT*). Finalmente se tocará el tema de de la Optimización de Geometrías moleculares.

2.1 Teoría del Funcional de la Densidad.

Entender y predecir los procesos de la Naturaleza posee un atractivo que ha cautivado a la humanidad por milenios. La Física enfocada en como la materia se mueve en el espacio y el tiempo, ha llegado a entender procesos muy rápidos a escalas muy pequeñas. Un ejemplo de esto puede ser la obtención de propiedades electrónicas de átomos y/o sólidos puede ser calculada resolviendo la ecuación de Shrödinger [35]:

$$\hat{H}\Psi = E\Psi \tag{2.4}$$

donde Ψ representa la función de onda del sistema multi-electrónico $\hat{H}$ es el Hamiltoniano, E son lo valores propios del Hamiltoniano. A pesar de que la eq. 2.4 arrojaría la solución exacta, resulta una tarea complicada de resolver mas allá de un número muy reducido de partículas. Damos al lector un ejemplo de cuantos recursos computacionales se requieren para almacenar la función de onda un solo átomo de *Carbono*, solo el resolver el GS sin tener nada más en cuenta. El *Carbono* contiene 6 electrones , 3 coordenadas por electrón (Cartesianas x, y, z) sobre una maya muy modesta de sólo 10 puntos de resolución por coordenadas. Esto arroja 10^{18} números, y utilizando 1 byte para cada número requiere 10^{18} bytes o ≈ 1 Exabyte (10^N con $N = n_{coordenadas} * n_{átomos}$). En términos de discos de almacenamiento disponibles comercialmente con una capacidad de 4 Terabytes, serán necesarios 262144 discos solo para los electrones. Esto es muy costoso. Inclusive si fuera factible, la eq. 2.4 presenta también el problema de la escalabilidad. Como cada partícula del sistema de múltiples componentes interactúa con el resto (ver eq. 2.3), el sistema escala exponencialmente lo que resulta extremadamente difícil para los ordenadores modernos[1] [36]. Los ordenadores cuánticos podrían enfrentar ambos problemas con mayor eficiencia, pero los retos técnicos que implican la construcción de este tipo de ordenadores lo hace en este punto una opción inviable [37, 38] [2].

Para lograr utilizar la eq. 2.4 de forma práctica importantes teoremas fueron desarrollados en un lapso de varias décadas. Seguidamente se dará mención a aquellos relevantes para este trabajo.

Teoremas de Hohenberg-Kohn.

[1] La mayoría de los ordenadores comerciales disponibles están basados en la arquitectura de *Von Neumann*. Las implicaciones técnicas y teóricas de las implementaciones para tratar algoritmos de crecimiento exponencial, se van del ámbito de este trabajo

[2] Se persigue con los ordenadores cuánticos alcanzar lo que se ha denominado *Supremacía Cuántica [37, 38].*, la cual marca el momento donde el ordenador cuántico realice una tarea imposible para un ordenador tradicional.

El primer teorema de Hohenberg-Kohn propone que todas las propiedades definidas de un sistema de $N-electrones$ en su estado fundamental (GS) puede ser obtenidas a partir de la densidad electrónica $\rho_0(\Psi_{GS})$. Por ejemplo la energía total del sistema E en el GS puede ser calculada en función de la densidad electrónica utilizando:

$$E[\rho] = T[\rho] + V_{Ne}[\rho] + V_{ee}[\rho] \tag{2.5}$$

Los términos $T[\rho]$ representan la energía cinética. La energía potencial $V_{ne}[\rho]$ y $V_{ee}[\rho]$ representan las interacciones Núcleo-Electrones y las interacciones Electrones-Electrones, respectivamente. En el caso de $T[\rho]$ y $V_{ee}[\rho]$ se engloban en una sola expresión (Funcional de Hohenberg-Kohn):

$$F_{HK}[\rho] = T[\rho] + V_{ee}[\rho] \tag{2.6}$$

donde $F_{HK}[\rho]$ solo depende de la densidad electrónica ρ y es independiente de la cantidad de átomos en el sistema. Determinar si una densidad electrónica $\tilde{\rho}$ es la del GS es el tema principal del segundo teorema de Hohenberg-Kohn (*Principio Variacional*). Donde se establece que si una energía $\tilde{E}$ obtenida de una densidad $\tilde{\rho}$ es mínima para el sistema de $N-electrones$, entonces la densidad $\tilde{\rho}$ corresponde al GS, ej. $E_{GS} \leq \tilde{E}[\tilde{\rho}]$. De esta forma se establece que cualquier suposición de la densidad electrónica determina la energía de GS o una energía superior [39, 35, 40].

Formulación de Levy-Lieb.

Como prueba concluyente del primer teorema de HK, el trabajo de Levy-Lieb propone una solución a la expresión $F_{HK}[\rho]$ en la forma de un problema de minimización definido como:

$$F_{HK}[\rho] = Min_{\rho}\{Min_{\Psi\to\rho}[\langle\Psi|\hat{T} + \hat{V}_{ee}|\Psi\rangle + \int V_{ee}(\rho)d\boldsymbol{x}]\} \tag{2.7}$$

buscando entre todas las posibles funciones de ondas de una densidad dada aquella que minimizara $\langle\hat{T} + \hat{V}_{ee}\rangle$. A pesar de la enorme contribución al teorema de HK, esta formulación pierde atractivo en cuanto hay que resolver la función de ondas para el sistema multi-partículas pero representa un importante paso [35, 40].

Methodo de Kohn - Sham (*KS*) y los Campos Auto-Consistentes(*SCF*).

El problema de la interacción entre todos los cuerpos del sistema fue brillantemente resuelta en formulación realizada por Kohn - Sham (KS). Probaron que es posible obtener la densidad del GS ρ_{GS} del sistema interactuante (Sistema Real)

a través de un sistema ficticio no interactuante (Sistema de KS). En su propuesta establecen el uso de función de onda monoelectrónica y obtener la función de onda del sistema no interactuante (ej. un Determinante de Slater $\Phi_0 = |\varphi_0, \varphi_i, ..., \varphi_N|$). El sistema de KS se define como:

$$(-\frac{\hbar^2}{2m}\nabla^2 + v_{eff}(\boldsymbol{x}))\varphi_i(\boldsymbol{x}) = \varepsilon_i\varphi_i(\boldsymbol{x}) \tag{2.8}$$

La densidad electrónica del sistema no interactuante puede reformularse en función de los orbitales de KS como:

$$\rho(\boldsymbol{x}) = \sum_i^N |\varphi_i(\boldsymbol{x})|^2 \tag{2.9}$$

el término $v_{eff}(\boldsymbol{x})$ de la eq. 2.8 se define como:

$$v_{eff}(\boldsymbol{x}) = v(\boldsymbol{x}) + v_H(\boldsymbol{x}) + v_{xc}(\boldsymbol{x}) \tag{2.10}$$

donde $v(\boldsymbol{x})$ es el potencial externo, y que normalmente representa las interacciones electrostáticas entre el núcleo y los electrones, $v_H(\boldsymbol{x})$ es el llamado potencial de *Hartree* que describe las interacciones de Coulomb [35], el último término v_{xc} es conocido como el potencial de intercambio y correlación y se define como:

$$v_{xc}(\boldsymbol{x}) = \frac{\delta E_{xc}[\rho]}{\delta\rho(\boldsymbol{x})} \tag{2.11}$$

y $E_{xc}[\rho]$ es conocido como el funcional de la energía de intercambio y correlación. Más adelante (Sección 2.1.1) se indagará sobre la naturaleza de esta energía.

Con todos los términos definidos en función de la densidad del sistema no interactuante ρ, la energía total se puede calcular utilizando:

$$E = \sum_i^N \varepsilon_i - \frac{1}{2}\int\int \frac{\rho(\boldsymbol{x})\rho(\boldsymbol{x}')}{|\boldsymbol{x} - \boldsymbol{x}'|}d\boldsymbol{x}d\boldsymbol{x}' + E_{xc}[\rho] - \int v_{xc}(\boldsymbol{x})\rho(\boldsymbol{x})d\boldsymbol{x} \tag{2.12}$$

y $\sum_i^N \varepsilon_i$ son los vectores propios que se obtiene utilizando la eq. 2.8.

Estas ecuaciones tienen que ser resueltas de una manera auto-consistente. Para la primera estimación de los orbitales φ se utiliza la *Combinación Lineal de Orbitales Moleculares* (LCAO por sus siglas en Inglés), a partir del cual se obtiene una primera aproximación de la densidad ρ, usando eq. 2.9. El resultado es utilizado para resolver la eq. 2.11, seguido por la solución de eq. 2.10 y por último eq. 2.8 para obtener nuevos orbitales φ. Este procedimiento en el cual se empieza con una aproximación de una propiedad y se llega a un valor según criterio de

convergencia se denomina Campos Auto-Consistentes (*Self-Consistent Field* SCF) y resulta fundamental en la solución de las ecuaciones de Kohn - Sham [40, 35]. La Figura 2.1 muestra un diagrama simplificado del algoritmo SCF en la solución del las ecuaciones de Kohn - Sham.

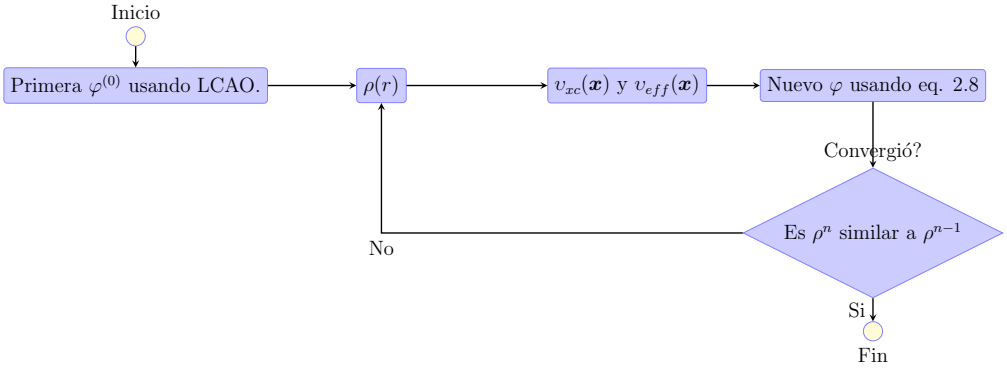

Figura 2.1: Diagrama simplificado del algoritmo auto-consistente para la solución de las ecuaciones de KS.

El esquema de KS toma un camino indirecto para la solución de la eq. 2.4 la solución que da es la exacta y no una aproximación. Sustituir el sistema de partículas interactuante por uno similar no interactuante redujo dramáticamente la complejidad computacional de exponencial a lineal, marcando la DFT como una herramienta sólida para realizar cálculos cuánticos consistentes. El único problema radica en el término que separa el sistema interactuante del no interactuante E_{xc}, para el cual no se conoce expresión universal [40, 35, 41]. En la próxima sección se abordarán distintas aproximaciones a la energía de intercambio y correlación E_{xc}.

2.1.1 La energía de Intercambio y Correlación

.

El término E_{xc} introducido en la eq. 2.11 se conoce como la energía de Intercambio y Correlación (*Exchange and Correlation Energy*). Representa la diferencia entre el sistema real interactuante y el sistema de KS de partículas no interactuante. Utilizando la eq. 2.6 como referencia, E_{xc} se define como:

$$E_{xc}[\rho] = (T[\rho] - T_{KS}[\rho]) + (V_{ee}[\rho] - J[\rho]) \tag{2.13}$$

donde $T_{KS}[\rho]$ es la energía cinética del sistema no interactuante y $J[\rho]$ representa interacciones de Coulomb que pueden calcularse de forma clásica.

La energía $E_{xc}[\rho]$ engloba todo lo que es desconocido y es el único término dentro del esquema de KS que no tiene una expresión universal para calcular su valor. Por eso se ha establecido distintos mecanismos para obtener este valor[40].

En la literatura aparecen disimiles maneras de aproximar E_{xc}. Alguna de las cuales agregan nuevos niveles de complejidad y/o coste computacional. En la práctica E_{xc} es dividido en dos contribuciones en la forma de una adición simple como: $E_{xc} = \varepsilon_x + \varepsilon_c$, donde ε_x y ε_c son las energías de intercambio y correlación [42].

Hartree - Fock expresión para la energía de intercambio.

En el trabajo de Hartree - Fock (HF en adelante) se propone un método para el cálculo de exacto de la componete de intercambio ε_x de esta energía. Dos elementos escenciales hay que destacar, el potencial de repulsión electrón - electrón de Coulomb, definido por [43, 44]:

$$(aa|bb) = \int d\mathbf{r}_1 d\mathbf{r}_2 |\psi_a(\mathbf{r}_1)|^2 \mathbf{r}_{12}^{-1} |\psi_b(\mathbf{r}_2)|^2 \tag{2.14}$$

y la componente de intercambio, definido por:

$$(ab|ba) = \int d\mathbf{r}_1 d\mathbf{r}_2 \psi_a^*(\mathbf{r}_1)\psi_b(\mathbf{r}_1)\mathbf{r}_{12}^{-1}\psi_b^*(\mathbf{r}_2)\psi_a(\mathbf{r}_2) \tag{2.15}$$

donde a y b representan a 2 electrones del sistema. De las eq. 2.14 y 2.15, esta última resula la mas complicada computacionalmente. Requiere de la permutación en pares de todos los electrones del sistema, algo que resulta costoso al aumentar el tamño del sistema [45, 43, 41].

Aproximación de la Densidad Local (*LDA*).

La idea detrás de LDA está basada en el comportamiento de de la densidad de un gas homogéneo. Dentro del volumen que la contiene, la densidad se considera constante. Entonces E_{xc} se puede expresar como:

$$E_{xc}[\rho(\boldsymbol{x})] = \int \rho(\boldsymbol{x}) \upsilon_{xc}[\rho(\boldsymbol{x})] dr \tag{2.16}$$

donde υ_{xc} es considerado como las contribuciones de intercambio y correlación independientes de cada partícula. Este término está únicamente determinado por la densidad ρ en la posición $\boldsymbol{x}$. Separando E_{xc} en sus contribuciones ε_x y ε_c , la contribución del intercambio ε_x se define por:

$$\varepsilon_x[\rho(\boldsymbol{x})] = -\frac{2}{3}\left(\frac{3}{\pi}\right)^{\frac{1}{3}} \int \rho(\boldsymbol{x})^{\frac{4}{3}} d\boldsymbol{x} \tag{2.17}$$

La componente de correlación ε_c resulta difícil de definir. en algunas ocasiones se le da valor 0, en otros casos se usan valores calculados previamente

mediante cálculos de *Monte Carlo* [40]. LDA representa una buena aproximación para sólidos conductores donde los electrones se comportan de manera similar a un gas homogéneo. Esta solución mantiene las ecuaciones de KS dentro de una complejidad computacional lineal [41, 40].

Aproximación del Gradiente Generalizado (*GGA*).

La densidad no siempre se comportar de manera uniforme, por tanto la solución de LDA tendrá limitaciones. Para manejar cambios en una distribución no uniforme se agregó un gradiente a la definición [46]. De forma general E_{xc} se expresa por:

$$E_{xc}^{GGA}[\rho(\boldsymbol{x})] = \int f^{GGA}(\rho(\boldsymbol{x}), \nabla\rho(\boldsymbol{x}))d\boldsymbol{x} \tag{2.18}$$

Considerando que $f^{GGA}(s)$ es una corrección para LDA, y la flexibilidad para definir s, hay toda una familia de funcionales basados en GGA [47, 40, 35, 48].

Aproximaciones Híbridas.

Las aproximaciones híbridas agregan un nuevo nivel de complejidad al funcional E_{xc}. Aparecen en el trabajo propuesto por *Becke* [49], dada la dificultad de determinar E_{xc}, sobre todo la componente ε_c. Intenta corregir de esta forma los problemas que encuentra LDA y GGA en algunas aplicaciones p.ej. en el cálculos de propiedades en estado gaseoso de moléculas y sólidos. De forma general se define como:

$$E_{xc} = \alpha E_x^{HF} + (1-\alpha)E_{xc}^{GGA} \tag{2.19}$$

donde E_x^{HF} es la energía de intercambio de *Hatree-Fock* definido en la eq. 2.14 y α es un parámetro de ajuste obtenido de datos experimentales.

Esta nueva aproximación tiene muchas expresiones dependiendo de cuanto de cada parte es agregada la mezcla. También mantiene una fuerte dependencia con los sistemas de donde α fue ajustada, dando mejores resultados con sistemas similares.

Tanto los funcionales GGA como los Híbridos presentan una multitud de formas producto de la flexibilidad en sus expresiones originales. Las virtudes y deficiencias de cada una están muy unidas a la definiciones de estas expresiones [42, 48].

Un ejemplo de funcional basado en GGA es *Perdew-Burke-Ernzerhof* (PBE por las iniciales de los autores), donde f^{GGA} de la eq. 2.18 se define como[47]:

$$f^{GGA} = 1 + k - \frac{k}{(1 + \frac{\mu s^2}{k})} \tag{2.20}$$

donde k y μ son restricciones física y no parámetros de ajuste. Este funcional es tan popular que tiene su propio sub-grupo dentro de GGA como *revPBE*, *RPBE* y *PBEsol* [50]. También posee su versión Híbrida conocida como *PBE0*. Descrita por Carlo Adamo y Vincenzo Barone como [51]:

$$E_{xc}^{PBE0} = E_x^{GGA} + a(E_x^{HF} + E_x^{PBE}) \tag{2.21}$$

PBE es muy utilizado en sólidos aunque presenta dificultades determinando la energía de enlace de Hidrógeno [47]. PBE0 corrige el problema de la energía de los enlaces de hidrógeno mientras mantiene las virtudes de PBE [51].

Alguna conclusiones sobre las aproximaciones a la energía de Intercambio y Correlación.

John P. Perdew y Karla Schmidt definieron la llamada *Escalera de Jacob*, (*Jacob's Ladder*) que no es mas que una clasificación de para las aproximaciones hechas a la Energía de Intercambio y Correlación, basado en precisión y coste computacional. La Figura 2.2 muestra la "Escalera" donde cada peldaño incrementa el coste computacional [52].

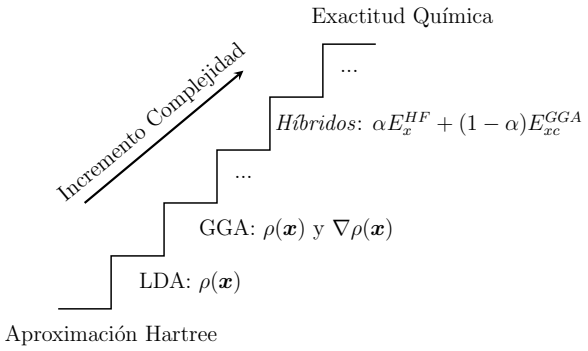

Figura 2.2: Posición de los métodos para aproximar la E_{xc} mencionados en la *Escalera de Jacob*.

Con N siendo la cantidad de electrones, LDA escala N^3 ya que depende solo de ρ, GGA también escala N^3 pero computacionalmente resulta mas costoso ya que depende de ρ y su gradiente ∇. Las aproximaciones Híbridas escalan N^4 siendo el cálculo de E_x^{HF} uno de los elementos más costosos[50].

Resulta obvio que alcanzar la *Exactitud Química* al menor costo computacional posible es deseable. Esta premisa ha sido la inspiración para la construcción de una impresionante cantidad de aproximaciones a E_{xc}, dado que la expresión universal para E_{xc} sigue incógnita. Narbe Mardirossian y Martin Head-Gordon mencionan

un enorme grupo de ellos para cada uno de los peldaños de la *Escalera de Jacob* mostrada en el Figura 2.2 [42].

2.2 Teoría del Funcional de la Densidad Dependiente del Tiempo (*TD-DFT*).

Los métodos revisados hasta ahora, DFT y el esquema de KS, describen bien las propiedades electrónicas de sistemas invariantes en el tiempo [43, 35]. La evolución en el tiempo de un sistema bajo la influencia de una perturbación externa se engloba los formalismos de la *Respuesta Lineal* (del inglés Linear Response LR) [48].

Partiendo de un Hamiltoniano que representa un perturbación dependiente del tiempo con la forma $\hat{H}(t) = F(t)\hat{\beta}$, donde $F(t)$ representa el campo externo y $\hat{\beta}$ es la propiedad que activa el campo en el instante de tiempo t_{GS}. El valor esperado de una propiedad α queda definida dependiente del tiempo, después de aplicado el Hamiltoniano sobre la función de onda como:

$$\alpha(t) = \langle \Psi(t)|\hat{\alpha}|\Psi(t)\rangle \tag{2.22}$$

La diferencia $\alpha(t) - \alpha_{GS}$ es lo que se conoce como *respuesta* del sistema a la perturbación del campo externo $F(t)$, y la primera potencia del campo $F(t)$ como *lineal*. La *respuesta lineal* que relaciona las propiedades $\hat{\alpha}$ y $\hat{\beta}$ se define como:

$$\alpha_1(t) = -i \int_{t_{GS}}^{t} dt' F(t') \langle \Psi_{GS}(t)|[\hat{\alpha}(t), \hat{\beta}(t')]|\Psi_{GS}(t)\rangle \tag{2.23}$$

La respuesta lineal se puede expresar en función de la densidad en los que se conoce como la *Representación de Lehmann* (del inglés Lehmann Representation) donde partiendo desde el GS la componente que describe las excitaciones se define como:

$$\chi(\hat{\alpha}, \hat{\beta}, \omega) = \sum_I \left[\frac{\langle \Psi_0|\hat{\rho}(\hat{\alpha})|\Psi_I\rangle \langle \Psi_I|\hat{\rho}(\hat{\beta})|\Psi_0\rangle}{\omega - \Omega_I + i0^+} \right] \tag{2.24}$$

donde Ψ_I los estados excitados de energía $E_I = E_{GS} + \Omega_I$. La eq. 2.24 expresada en plano de las frecuencias ω, asegura la obtención de la descomposición espectral del sistema interactuante [48, 35].

La polarización dinámica es una propiedad que puede ser extraída utilizado la respuesta lineal. Resulta de especial interés para la espectroscopia molecular por relacionar un potencial eléctrico externo v con los cambios en los dipolos

($\mu = -q\boldsymbol{x}$). Expresando los términos usando la representación de Lehmann se puede definir como:

$$\alpha_{\mu\lambda}(\omega) = \sum_{n=1}^{\infty} \left\{ \frac{2\Omega_n \langle \Psi_0 | \hat{\boldsymbol{x}_\nu} | \Psi_n \rangle \langle \Psi_0 | \hat{\boldsymbol{x}_\lambda} | \Psi_n \rangle}{\Omega_n^2 - \omega^2} \right\} \tag{2.25}$$

donde $\hat{\boldsymbol{x}_\nu}$ representa el operador del dipolo y $\hat{\boldsymbol{x}_\lambda}$ al campo eléctrico externo, las energías de excitación se obtienen de $\Omega_n = E_n - E_0$ y las fuerzas de oscilador $f_n = \frac{2\Omega_n}{3} \sum_{\mu=1}^{3} |\langle \Psi_0 | \hat{\boldsymbol{x}_\mu} | \Psi_n \rangle|^2$.

Resolver la eq. 2.25 resulta complicado, requiere tener conocimiento de la función de onda de varios estados de un sistema interactuante. Este problema es resuelto utilizando el esquema de Kohn - Sham dependiente del tiempo (TDDFT), donde la respuesta lineal del sistema interactuante es obtenida a través del sistema no interactuante definido como [48, 35]:

$$\chi_{KS}(\boldsymbol{x}, \boldsymbol{x}', \omega) = \lim_{\eta \to o^+} \sum_{k,j} (f_k - f_j) \frac{\varphi_j^{(0)*}(\boldsymbol{x}) \varphi_k^{(0)}(\boldsymbol{x}) \varphi_j^{(0)*}(\boldsymbol{x}') \varphi_k^{(0)}(\boldsymbol{x}')}{\omega - (\varepsilon_j - \varepsilon_k) + i\eta} \tag{2.26}$$

donde $\varphi_k^{(0)}$ son los orbitales de Kohn -Sham y $(f_k - f_j)$ son los números de ocupación.

La forma matricial de la respuesta lineal del sistema no interactuante fue propuesta por Mark E. Casida en la llamada *Ecuación de Casida* (*Casida's equation*) y se define como [53]:

$$\mathbf{R}\mathbf{F}_I = \boldsymbol{\Omega}^{\mathbf{2}}{}_I \mathbf{F}_I \tag{2.27}$$

Las energías de excitación Ω pueden ser obtenidas como los valores propios y las fuerzas del oscilador como los vectores propios $\mathbf{F}_I$ usando:

$$f_I = \frac{2}{3}(|x\mathbb{S}^{-1/2}F_I|^2 + |y\mathbb{S}^{-1/2}F_I|^2 + |z\mathbb{S}^{-1/2}F_I|^2) \backslash |F_I|^2 \tag{2.28}$$

y R se define como:

$$R_{qq'} = \omega_q^2 \delta_{qq'} + 4\sqrt{\omega_q \omega_{q'}} \int d^3r \int d^3r' \Phi_q(r) f_{Hxc}(r, r', \Omega_I) \Phi_{q'}(r') \tag{2.29}$$

donde f_{Hxc} se conoce como el *kernel* de intercambio y correlación, necesario para poder resolver el sistema de Kohn - Sham y $\Phi_q = \varphi_a^{(0)*}(\boldsymbol{x}) \varphi_i^{(0)}(\boldsymbol{x})$. La eq. 2.27 está implementada en la mayoría de los códigos de Física y Química Computacional, porque es fácil de codificar y su solución aporta directamente las energía de excitación y las fuerzas del oscilador. Ambas propiedades están directamente relacionadas con el espectro de absorción del sistema [48, 35].

2.3 Optimización de Geometrías.

Existen diversas formas de representación espacial de compuestos químicos, coordenadas cartesianas, polares, etc. En este trabajo se van a considerar sólo las llamadas *coordenadas internas* que consisten en distancias, ángulos y ángulos de torsión. Usando esta representación los compuestos tienen $3N-6$ grados de libertad eliminando rotación y traslación [54, 55]. La Figura 2.3 se puede apreciar un ejemplo de átomos representados usando este tipo de coordenadas.

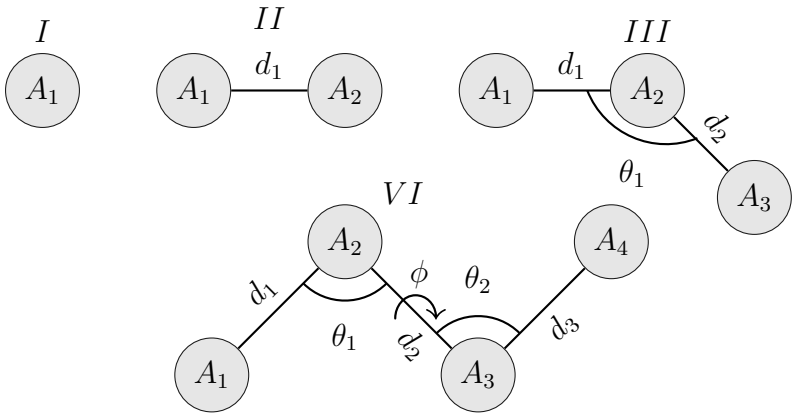

Figura 2.3: Ejemplo de coordenadas internas para un grupo de conformaciones atómicas. *I* solo un átomo, define el origen de coordenadas. *II* con dos átomos se fija A_1 y la posición de A_2 la determina la distancia d_1. *III* con tres átomos se fija A_1, A_2 y A_3 se posicionan a partir de dos distancias (d_1 y d_2) y un ángulo(θ_1). *VI* con cuatro átomos seis coordenadas son necesarias, tres distancias (d_1, d_2 y d_3), dos ángulos (θ_1 y θ_2) y un ángulo de torsión (ϕ).

El procesos de encontrar la estructura molecular que minimice la energía potencial se conoce como *Optimización de Geometrías* (del inglés *Geometry optimization*), también aparece como *relajación* ya que se minimizan las fuerzas interatómicas. Es el primer paso antes de cualquier cálculo que garantiza minimizar las fuerzas internas entre los elementos del compuesto. Se utiliza la energía potencial por ser invariante a rotación y/o traslación en la ausencia de campos externos, una característica necesaria si se trabaja con distintas conformaciones de átomos [41, 56].

El espacio de posibles geometrías y sus correspondientes energías potenciales, define lo que se conoce como *Superficies de Energía Potencial* [57] (del inglés *Potential Energy Surfaces* PES) [55].

Las superficies PES proporcionan el medio en donde realizar la búsqueda de la estructura óptima. Dentro de las PES están contenidas todas las posibles conformaciones para una agregación particular de átomos. La Figura 2.4 muestra una

superficie PES para un grupo hipotético de mas de tres átomos, solo se presentan 2 dimensiones y el resto se han mantenido como constante.

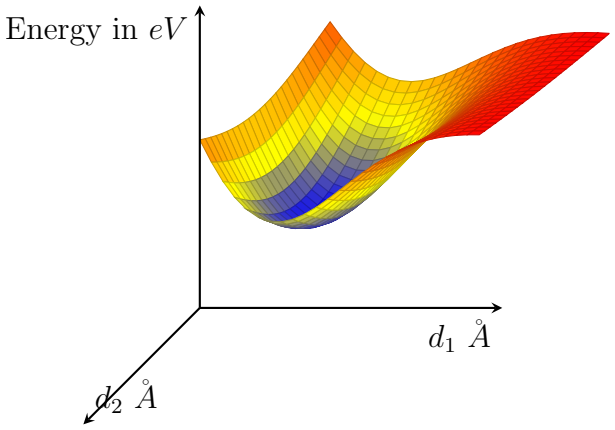

Figura 2.4: Ejemplo de superficie PES para un grupo hipotético de átomos y coordenadas arbitrarias. La región azul marca un mínimo, donde la geometría se encuentra optimizada.

Los métodos para explorar las superficies PES y encontrar el mínimo varían [56]. Constan de 2 pasos fundamentales: (1) calcular la energía potencial y (2) método de búsqueda del mínimo en la PES. El cálculo de la energía potencial puede realizarse utilizando varias técnicas como cálculos semi-empíricos donde se utilizan datos ya tabulados obtenidos mediante procesos experimentales unido a simplificaciones y parametrizaciones que permiten calcular grandes agrupaciones de átomos [58, 59, 60]. Otro método es utilizar métodos *ab-initio* como DFT o TDDFT, muy costosa si el número de átomos es grande, ya que para logar una buena aproximación de la geometría el uso de un funcional híbrido para la energía de intercambio y correlación es recomendable [55, 57, 61, 62].

La búsqueda del mínimo resulta en un problema de optimización La mayoría de los algoritmos utilizan ∇E, donde E es la energía potencial, por la garantía que ofrece de encontrar el mínimo y las expresiones de la energía potencial son conocidas y por tanto posible implementar derivadas de orden n. Otros métodos no basados en gradientes son utilizados cuando resulta imposible obtener las derivadas correspondientes, pero solo se recurre a ellos en casos muy excepcionales ya que no garantizan convergencia [63, 64, 65].

Capítulo 3

Elementos de Aprendizaje Automático.

The best computer is a man, and it's
the only one that can be mass-produced
by unskilled labor.
Wernher von Braun.

Contents

El término *Aprendizaje Automático* (Machine Learning *ML*) se considera como una sub-rama de la *Inteligencia Artificial*. Recientemente ha adquirido prominencia por su habilidad de encontrar comportamientos ocultos en grandes volúmenes de datos. También se ha aprovechado del incremento sustancial en la capacidad de computo y la disminución del precio comercial. Algo que ha permitido implementar algoritmos que asimilen mayor volumen de datos y por consiguiente

mas exigentes computacionalmente [66]. En este trabajo se abordarán métodos clasificados como *Aprendizaje Supervisado*[1], donde se conocen las *entradas* y las *salidas* y el objetivo del aprendizaje es construir la función de transformación, que convierta las entradas en las salidas [67, 68]. Antes de describir la estructura del capítulo, se introducirán a continuación algunos acuerdos en cuanto a terminología y representación de términos, así como también algunos conceptos básicos relacionados con ML.

Representación de matrices, tensores y valores:

- Representación de Tensores: Todos los tensores se representaran de acuerdo a su orden. Si el orden es mayor o igual a 2 se representará como $\mathbf{M}$; si el orden es menor de 2 como $\bar{\mathbf{m}}$; los valores escalares se representarán con uno, dos o mas subíndices si perteneces a tensores de orden correspondientes o sin ninguno si son valores constantes (ej. $m_{i,j}$, $\mathbf{m}_i$ y n).

Conceptos relacionados con ML:

- *Propiedad*: Las propiedades serán considerados como todo aquellos fenómenos sobre los que se pueden hacer mediciones.

- *Descriptor*: Grupo de valores usados para etiquetar una medición realizada sobre una Propiedad. Un Descriptor $\bar{\mathbf{D}}$ puede ser un tensor de orden 1 o mayor; ej. $\bar{\mathbf{D}} = \{d_1, d_2, \ldots, d_N\}$ donde N es un número arbitrario finito.

- *Referencia*: Mediciones hechas sobre una Propiedad para un descriptor $\bar{\mathbf{D}}$. Las mediciones de Referencia pueden ser un tensor orden 1 o mayor; ej. $\bar{\mathbf{R}} = \{r_1, r_2, \ldots, r_N\}$ donde N es un número arbitrario finito.

- *Base de Entrenamiento*: Constituye el total de todos los datos disponibles para realizar el entrenamiento $\mathbb{B}$. Puede ampliarse en volumen de datos si nueva información es incorporada.

- *Conjunto de Entrenamiento*: Conjunto de pares $\bar{\mathbf{D}}, \bar{\mathbf{R}}$ utilizados para "enseñar" (*Procesos de Aprendizaje*) al método de ML. El Conjunto de Entrenamiento es $\mathbb{T} \subseteq \mathbb{B}$ y se puede definir como:
  $\mathbb{T} = \{[\bar{\mathbf{D}}_1, \bar{\mathbf{R}}_1], [\bar{\mathbf{D}}_2, \bar{\mathbf{R}}_2], \ldots, [\bar{\mathbf{D}}_N, \bar{\mathbf{R}}_N]\}$ donde N es un número arbitrario finito.

- *Conjunto de Control*: Conjunto de elementos utilizados para evaluar el método de ML después de que el Proceso de Aprendizaje concluya. El Conjunto de Control se define usualmente como $\mathbb{C} \subseteq \mathbb{B}$, por tanto:

[1] El lector puede profundizar en este y otros métodos en [67] y [66]

$\mathbb{C} = \{[\bar{\mathbf{D}}_1, \bar{\mathbf{R}}_1], [\bar{\mathbf{D}}_2, \bar{\mathbf{R}}_2], \ldots, [\bar{\mathbf{D}}_N, \bar{\mathbf{R}}_N]\}$ donde N es un número arbitrario finito.

- *Procesos de Aprendizaje*: En general se puede ver como el grupo de paso tomados para construir la función que satisface $f(\bar{\mathbf{D}}_{\mathbb{T}}, \alpha) = \bar{\mathbf{R}}_{\mathbb{T}}$, donde α es un conjunto de parámetros (*hiperparámetros*) [2].

- *Predicción*: Si se considera el *Modelo Entrenado* como F entonces la predicciones pueden definirse como $F(\bar{\mathbf{D}}_{\mathbb{C}}, \alpha) = \bar{\mathbf{P}}$ donde: $\bar{\mathbf{P}} = \{p_1, p_2, \ldots, p_M\}$ donde M es un número arbitrario finito.

- *Pérdida del Modelo*: Métrica de cuanto la Estimación $\bar{\mathbf{P}}$ se desvía de la referencia del Conjunto de Entrenamiento $\bar{\mathbf{R}}_{\mathbb{T}}$. Se define como: $\mathbb{L}(\bar{\mathbf{R}}_{\mathbb{T}}, f(\bar{\mathbf{D}}_{\mathbb{T}}, \alpha))$.

La Figura 3.1 muestra la relación de los conceptos de ML en un diagrama simplificado. Realizadas estas aclaraciones se puede introducir al lector al contenido del

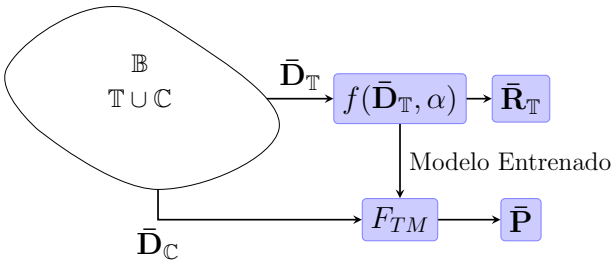

Figura 3.1: Relación de los conceptos involucrados en el *Proceso de Aprendizaje*. Para evaluar la precisión del *Modelo Entrenado* F_{TM} simplemente calcular $\mathbb{L}$

capítulo. Primeramente se va presentar una introducción general a las Máquinas de Vectores de Soportes y mas detalladamente al método de Regresión Ridge con Kernel. Seguido se detallarán una breve descripción la Redes Neuronales.

3.1 Breve introducción a las Máquinas de Vectores de Soporte.

Dentro de los métodos de *Aprendizaje Supervisado* se encuentra los conocidos como *Maquinas de Vectores de Soporte* (del inglés *Support Vector Machine*, en adelante se usará SVM)[67, 66, 68, 69]. Estos métodos usualmente utilizados para clasificación y regresión, intentan construir un hiper-plano ($f(\bar{\mathbf{D}})$) que permita separar o estimar valores respectivamente.

[2] Los hiper-parámetros aparecerán a lo largo de este capítulo representados con otros símbolos.

Se puede utilizar un ejemplo para formular el problema. Si se considera un conjunto de entrenamiento $\mathbb{T}$, donde las referencias $\bar{\mathbf{R}}_{\mathbb{T}}$ tienen solo 2 posibles valores $\{1, -1\}$ como etiqueta de clasificación para cada uno de los descriptores $\bar{\mathbf{D}}_{\mathbb{T}} = \{d_1, d_2, \ldots, d_N\}$ entonces:

$$r_i * [(\bar{\mathbf{w}} \cdot d_i) - b] \geq 1, \quad i = 1, \ldots, N \tag{3.1}$$

donde $\bar{\mathbf{w}}$ es el hiper-plano, b es una constante (*bias* o *umbral*, en adelante se usará umbral)[3] y los llamados *vectores de soporte* son todos aquellos d_i donde la desigualdad eq. 3.1 se cumple. La eq. 3.1 presenta la dificulta de la alta dimensionalidad del tensor $\bar{\mathbf{D}}$, para el que construir un hiper-plano que clasifique correctamente resulta complicado desde el punto de vista técnico.

La desigualdad planteada en la eq. 3.1 puede replantearse usando la *Dualidad de Lagrange*, donde para obtener $\bar{\mathbf{w}}$ se utiliza:

$$\bar{\mathbf{w}} = \sum_{i=0}^{N} r_i \alpha_i d_i, \quad \alpha_i \geq 0, \quad i = 1, \ldots, N \tag{3.2}$$

donde α_i son los coeficiente de *Lagrange* del hiper-plano que pueden encontrarse resolviendo la maximización:

$$max_{\alpha}\{\sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j r_i r_j d_i \cdot d_j\} \tag{3.3}$$

$$sujeta \quad a \quad \sum_{i=1}^{N} \alpha_i r_i = 0 \quad y \quad \alpha_i \geq 0 \tag{3.4}$$

Con la eq. 3.3 se obtienen los coeficientes óptimos α y con la eq. 3.2 se obtiene el hiper-plano óptimo $\mathbf{w}$. Resulta interesante destacar que los tensores de entrenamiento ($\bar{\mathbf{D}}_{\mathbb{T}}$) están representados en el espacio de los productos escalares, desagregando el costo de resolver las ecuaciones, de la dimensionalidad que pueda presentar el descriptor. Para lograr esta desagregación se realiza una operación conocida como *sustitución de kernel* (del inglés *Kernel Trick*) donde $K(d_i, d_j) = (d_i \cdot d_j)$. La función de estimación f que representa al hiper-plano tiene la forma:

$$f(\bar{\mathbf{D}}_{\mathbb{T}}, \bar{\boldsymbol{\alpha}}, \bar{\mathbf{w}}) = \sum_{i=1}^{N} r_i \alpha_i K(\bar{\mathbf{D}}, w_i) \tag{3.5}$$

$$sujeta \quad a \quad \alpha_i \geq 0 \tag{3.6}$$

[3]Este valor le otorga libertad de traslación al hiper-plano

donde $\bar{\mathbf{w}}$ es el hiper-plano, $\bar{\boldsymbol{\alpha}}$ son los coeficientes Lagrange del hiper-plano, $\bar{\mathbf{D}}_{\mathbb{T}}$ son los tensores que representan a los descriptores y r son los tensores que representan las referencias $\bar{\mathbf{R}}_{\mathbb{T}}$. Esta función es utilizada para evaluar nuevos puntos.

Se puede ver el kernel K como una función de interpolación intercambiable [4] que sustituye la necesidad de trabajar en el espacio de alta dimensionalidad que pudieran presentar los descriptores $\bar{\mathbf{D}}_{\mathbb{T}}$ [66, 69].

3.1.1 Regresión Ridge con Kernel (KRR).

La Regresión Ridge (RR) es un método de *Regresión* dentro de SVM. En particular RR se enfoca en problemas de predicción de valores. El objetivo es buscar una función $f(\mathbb{T}_{\bar{\mathbf{D}}}, \alpha)$ que estime el valor de $\mathbb{T}_{\bar{\mathbf{R}}}$, cuando este último no es una variable discreta fácilmente separable [70, 67].

La Regresión se puede definir como:

$$f(\bar{\mathbf{D}}_{\mathbb{T}}, \bar{\mathbf{w}}) = b + \mathbf{w}^T \bar{\mathbf{D}}_{\mathbb{T}} \tag{3.7}$$

donde $\bar{\mathbf{w}}$ es el hiper-plano, b es el umbral. Nótese la similitud con eq. 3.1. La eq. 3.7 se puede expresar utilizando la *Suma de los Errores al Cuadrado*(del inglés *Sum Squared Errors*) como función de pérdida $\mathbb{L}$ del modelo:

$$\mathbb{L}(\bar{\mathbf{R}}_{\mathbb{T}}, f(\bar{\mathbf{D}}_{\mathbb{T}}, \bar{\mathbf{w}})) = \frac{1}{2} \sum_{n=1}^{N} \{\bar{\mathbf{R}}_{\mathbb{T}} - (b + \mathbf{w}^T \bar{\mathbf{D}}_{\mathbb{T}})\}^2 \tag{3.8}$$

La eq. 3.7 presenta el problema de la alta dimensionalidad del tensor de entrenamiento $\bar{\mathbf{D}}_{\mathbb{T}}$. Además como se pretende construir una curva de estimación aparece un nuevo problema relacionado con el ajuste de la curva, que ocurre cuando $f(\bar{\mathbf{D}}_{\mathbb{T}}, \bar{\mathbf{w}})$ se ajusta demasiado o muy poco a los descriptores $\bar{\mathbf{D}}_{\mathbb{T}}$ produciendo resultado indeseados. La Figura 3.2 se muestra estos fenómeno:

Para el problema de la dimensionalidad se utiliza la sustitución de kernel, y para atenuar los problemas de ajuste se introduce un valor de regularización [70]. Re-escribiendo la eq. 3.8:

$$\frac{1}{2} \sum_{n=1}^{N} \{\bar{\mathbf{R}}_{\mathbb{T}} - (b + \mathbf{w}^T \bar{\mathbf{D}}_{\mathbb{T}})\}^2 + \frac{1}{\lambda} \mathbf{w}^T \mathbf{w} \tag{3.9}$$

[4] El tipo de Kernel utilizado tiene que ser representable en el espacio de productos escalares, definido positivamente y cumplir con las condiciones de *Mecer*[66, 67]

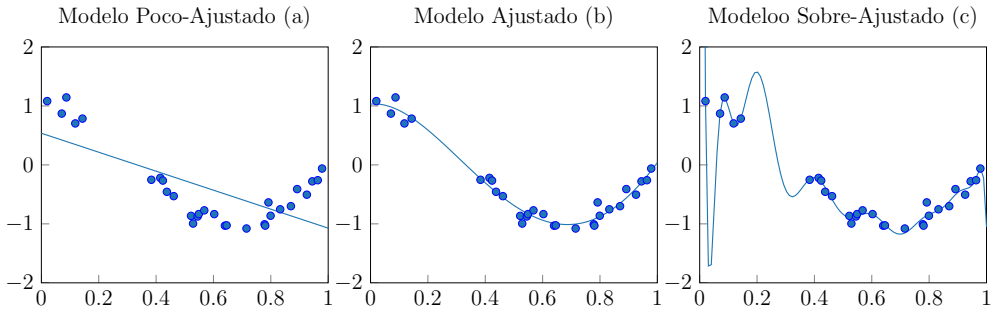

Figura 3.2: Ejemplo de modelos: Poco-Ajustado (a), Ajustado (b) y Sobre-Ajustado (c). Siendo • las *Observaciones* y – el Modelo obtenido. Nótese que para las Observaciones en el modelo (b) se pueden obtener valores de pérdidas $\mathbb{L}$ muy bajos e induce a un análisis erroneo acerca de su efectividad, ya que la evaluación de un punto del conjunto $\mathbb{C}$ provoca una desviación enorme.

donde $\frac{1}{\lambda}\mathbf{w}^T\mathbf{w}$ es el término de regularización y λ el coeficiente de rigidez (*ridge*). Agregando la sustitución de kernel, se llega a la expresión[5]:

$$\bar{\mathbf{P}} = k(\bar{\mathbf{D}}_{\mathbb{T}})^T(\mathbf{K} + \lambda\mathbf{I})^{-1}\bar{\mathbf{R}}_{\mathbb{T}} \tag{3.10}$$

donde I es la matriz de identidad, K es una matriz simétrica que se obtienen usando $K = k(d_i, d_j)$ y $k(*)$ es la función del kernel. Todo la solución está expresada en función del kernel. La selección del kernel le da flexibilidad al modelo a pasar de lineal a no lineal[6], un ejemplo es: $k(d_i) = \exp\{-\frac{(d_i-\mu_i)^2}{2*\sigma^2}\}$ denominada como *Gaussian* donde μ_i determina la localización de la función en el espacio de entrada y σ es una variable que determina el alcance espacial del kernel [70].

La Figura 3.3 muestra un ejemplo utilizando eq. 3.10 y un kernel Gaussian. Se puede apreciar como los valores de σ y λ interfieren en el proceso de aprendizaje provocando los efectos de sobre-ajuste y poco-ajuste del modelo. La primera fila desde arriba en la Figura 3.3 se modifica σ, siendo una variable propia del kernel tiene un influencia notable en el ajuste del método. La segunda fila se modifica λ, este hiper-parámetro tiene una influencia menos dramática sobre el ajuste del modelo. Sin embargo valores incorrectos llevan al completo desbalance inclusive si el valor de σ es el adecuado.

Este ejemplo demuestra el papel crucial que juega la optimización de los hiper-parámetros (Ver Sección 3.3)en los modelos de aprendizaje automático.

[5]La completa derivación de esta expresión no es tema de este trabajo. Remitimos amablemente al lector a [70] y [66]

[6]Existen otras variantes de funciones como *lineal*, *polinómica* o *sigmoid*, etc. Cada una cambia la naturaleza de la Regresión para adaptarlo a las particularidades del problema de estimación. El lector puede indagar sobre estas y otras funciones de kernel en [70], [67] y [66].

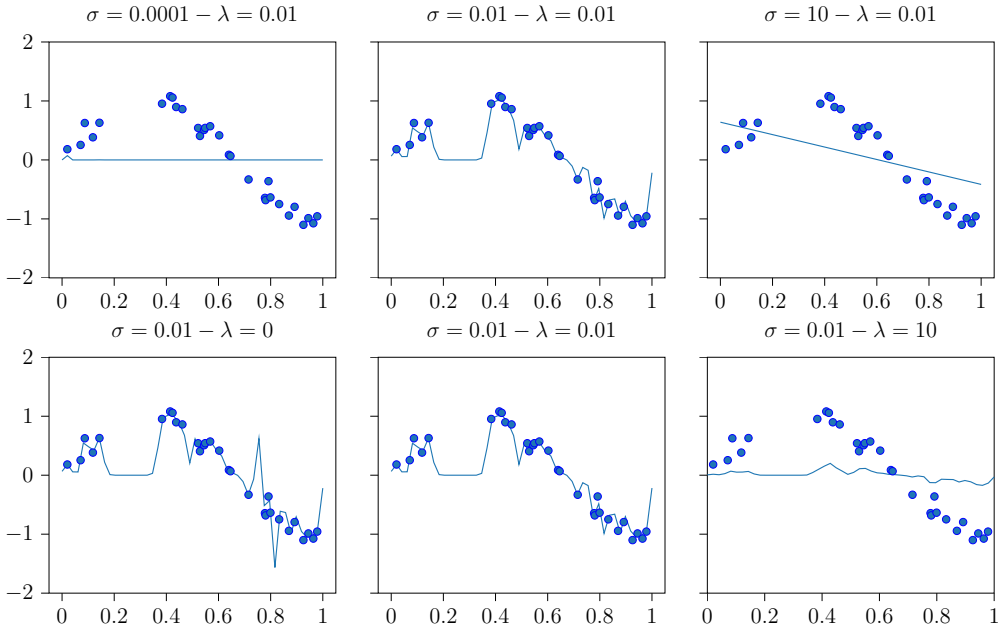

Figura 3.3: Ejemplo de valores para σ y λ. Siendo • las *Observaciones* y – el Modelo obtenido. Primera fila desde arriba se modifica σ mientras λ permanece constante. En la segunda fila se modifica λ mientras σ permanece constante.

3.1.2 Algoritmo del *Vecino Cercano.*

Para mejorar el problema de regresión cuando no se consigue estimar correctamente el conjunto de control $\mathbb{C}$, se puede asumir que puntos cercanos deben ser similares. Por tanto se realiza un cribado del conjunto de entrenamieto $\mathbb{T}$. La solución puede formularse en un corto algoritmo de la forma:

Algorithm 3.1: Búsqueda del Vecino Cercano

Require: $\mathbb{T}$ y $\bar{\mathbf{D}}_{\mathbb{C}}$

1: **for all** $d_i^{\mathbb{C}}$ en $\bar{\mathbf{D}}_{\mathbb{C}}$ **do**

2: $\quad d' = \{||d_i^{\mathbb{C}} - \bar{\mathbf{D}}_{\mathbb{T}}||^l \leq r\}$

3: $\quad \bar{\mathbf{P}} = k(d')^T(\mathbf{K'} + \lambda\mathbf{I})^{-1}\bar{\mathbf{R}}'_{\mathbb{T}}$

4: **end for**

donde $||*||^l$ es una norma y r es un umbral de tolerancia de inclusión. En otras palabras cada nuevo punto se estima utilizando aquellos tensores de entrenamiento que le son cercanos según la norma $||*||^l$ y cumpliendo la condición r. El conjunto tensores resultantes cumple $\bar{\mathbf{D}}'_{\mathbb{T}} \leq \bar{\mathbf{D}}_{\mathbb{T}}$ que trae como beneficio que el algoritmo se puede entrenar con menos tensores (en el peor de los casos todos los tensores

del conjunto $\bar{\mathbf{D}}_{\mathbb{T}}$ participarían en el entrenamiento) y los escogido presentan gran similitud con el nuevo punto lo que garantiza en gran medida una buena estimación. Es importante destacar como problema que este algoritmo agrega una sobrecarga extra, porque para estimar cada nuevo punto hay que buscar aquellos tensores donde la norma es mínima. Puede ser costoso computacionalmente si el conjunto $\bar{\mathbf{D}}_{\mathbb{T}}$ es extenso [71, 72, 73].

3.2 Introducción a las Redes Neuronales.

El término *Redes Neuronales* (del ingés *Neural Networks* NN), aparece como un intento de encontrar un representación matemática a los procesos de los sistemas biológicos. En muchas aplicaciones, las NN resultan compactas y eficientes que su contraparte SVM. Aunque nada viene sin un coste, y en el caso de las NN puede verse incrementado considerablemente en el tiempo de entrenamiento [70].

El modelo mas simple de red neuronal es el conocido como *Perceptrón* de 1 capa[7]. La definición de Preceptrón involucra un *Grafo Completo* sin ciclos compuesto por: (1) nodos de entrada, (2) capas ocultas y (3) nodos de salida. Este modelo también es conocido como *Perceptrón multi-capas* (del inglés *Multi-Layer Perceptron*, en adelante MLP). La Figura 3.4 muestra esta estructura.

Siendo $\bar{\mathbf{D}}_{\mathbb{T}}$ los tensores de entranmiento, se puede definir una NN como:

$$f(\bar{\mathbf{D}}_{\mathbb{T}}, \mathbf{w}) = g(\sum_{i=1}^{M} w_i \phi_i(\bar{\mathbf{D}}_{\mathbb{T}}) + b) \tag{3.11}$$

donde $g(*)$ se conoce como la función de *Activación* (ver Sección 3.2.3), $\phi(*)$ es una función no lineal, w_i son los *pesos* (del inglés *weights*) determinana la influencia de cada enlace sobre la siguiente capa y b representa el umbral[8]. Para poder extender este modelo a una red se busca hacer que $\phi(*)$ dependa de parámetros y que estos parámetros sean ajustables junto con $\mathbf{w}$ durante el proceso de entrenamiento [70, 74].

3.2.1 Propagación hacia adelante (*Feed-Forward*).

Se puede considerar un conjunto de entrenamiento $\mathbb{T} = \{[\bar{\mathbf{D}}_1, \bar{\mathbf{R}}_1], \ldots, [\bar{\mathbf{D}}_N, \bar{\mathbf{R}}_N]\}$ con N número de muestras y se busca extrapolar la eq. 3.11 de una neurona a

[7] Al usar esta notación se hace referencia al número de capas ocultas, debido a que la capa de entrada y salida son necesarias para el correcto funcionamiento de este algoritmo. La literatura en ocasiones contempla al Perceptrón de 1 capa con solo las entradas y salidas [66, 67]. En este trabajo siempre se hará referencia al número de capas ocultas y se dará por implícito que también existen las entradas y salidas

[8] Este valor permite un desplazamiento a los pesos, proporcionando flexibilidad.

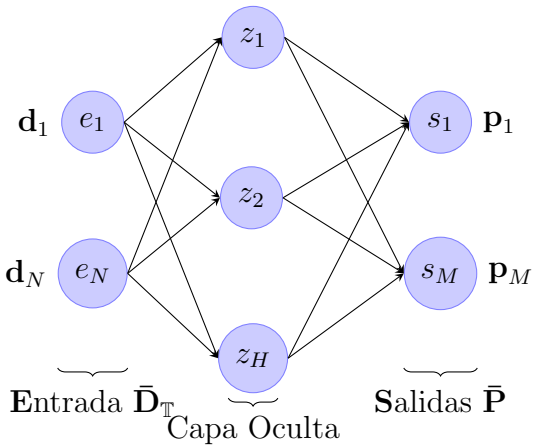

Figura 3.4: Representación de un MLP donde se aprecia su estructura de grafo completo compuesto por (1) Entradas e_N con N igual al número de tensores en $\bar{\mathbf{D}}_{\mathbb{T}}$, (2) Capas ocultas pueden existir un número arbitrario de capas ocultas y neuronas por capas y (3) Salidas S_M donde se obtienen las predicciones $\bar{\mathbf{P}}$.

varias capas de neuronas entrelazadas para conformar un MLP [70].

Usando la eq. 3.11 se puede escribir la primera capa como:

$$c_h = \sum_{n=1}^{N} w_{n,h}^0 d_n + b_h^0 \tag{3.12}$$

con $h = 1, 2, \ldots, H$ siendo H el número de enlaces con la siguiente capa, c_h es el valor de *activación*, $w_{n,h}$ son los *pesos* del enlace y b_h^0 es el umbral. La salida de la neurona a la siguiente capa se obtiene al aplicar sobre el valor de activación c_h la función de activación:

$$z_h = g(c_h) \tag{3.13}$$

Este proceso de "alimentar" la red hacia adelante, utilizando los tensores de entrada, los pesos y los valores de umbral, se conoce como *Propagación hacia adelante* (del inglés *Feed-Forward*, en adelante FF). La característica más importante de esta propagación es mantener los pesos y el umbral fijos, calculando para cada capa las salidas $\bar{\mathbf{z}}$, las cuales constituyen la entrada de la siguiente capa. La Figura 3.5 como ocurre este proceso para un MLP de una sola capa oculta. El pase completo (FF) mostrado en la Figura 3.5 ocurre en 4 etapas:

- Se evalúa la eq. 3.12 utilizando los pesos $w_{n,h}, \ldots, w_{N,H}$ con N número de tensores de entrada en $\bar{\mathbf{D}}_{\mathbb{T}}$ y H número de neuronas en la primera capa oculta.

- Se evalúa la eq. 3.13 para todo $c_h, \ldots, c_H$.
- Se evalúa la eq. 3.12 utilizando los pesos $w_{h,m}, \ldots, w_{H,M}$ con M número de tensores de salida en $\bar{\mathbf{P}}$.
- Por último se evalúa la eq. 3.13 para todo $c_1, \ldots, c_M$ en las salida. Usualmente b se representa como una neurona sin conexiones, para este caso particular se considerará como un valor intrínseco de cada capa.

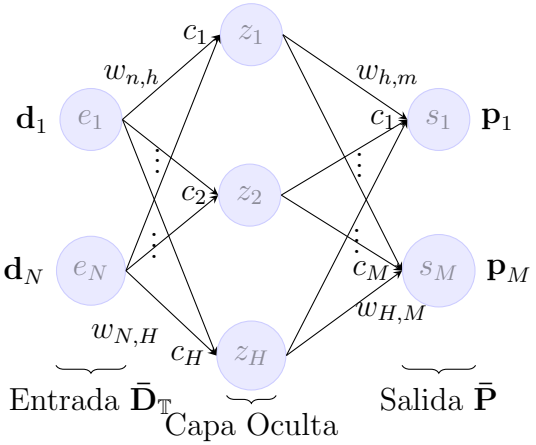

Figura 3.5: Representación de un MLP donde se aprecian el proceso de propagación FF, desde los tensores de entrada hasta las salidas.

Al finalizar el proceso de FF se obtienen las predicciones $\bar{\mathbf{P}}_M$, los cuales pueden se comparados contra las referencias $\bar{\mathbf{R}}_{\mathbb{T}}$ utilizando la función de pérdida $\mathbb{L}(\bar{\mathbf{R}}_{\mathbb{T}}, \bar{\mathbf{P}}_M)$[9] [70, 74].

3.2.2 Propagación hacia atrás (*Back-Propagation*).

Considerando que el resultado del proceso FF son las predicciones $\bar{\mathbf{P}}$ se busca que de manera automática este valor se acerque al ideal $\bar{\mathbf{R}}_{\mathbb{T}} - \bar{\mathbf{P}} = 0$ [74, 67, 69]. Resulta evidente a través de la eq. 3.12 que existe una dependencia de $\bar{\mathbf{R}}'$ con $\bar{\mathbf{w}}$ y siendo un problema de minimización, entonces al final de cada FF se pueden mejorar los pesos de la última capa usando un gradiente:

$$\bar{\mathbf{w}}^{nuevo} = \bar{\mathbf{w}}^{viejo} + \nabla \bar{\mathbf{w}} \tag{3.14}$$

[9] Las métricas para errores varían, dependiendo de la aplicación. En el Capítulo 6 se definirá esta función.

donde $\nabla\bar{\mathbf{w}}$ se puede expresar como:

$$\nabla\bar{\mathbf{w}} = -\mu\frac{\partial\mathbb{L}}{\partial\bar{\mathbf{w}}} \tag{3.15}$$

donde μ es la constante de aprendizaje que regula la evaluación de la derivada en la eq. 3.15. El término que puede causar problemas es la evaluación de la propia derivada en la eq. 3.15 que se calcula como:

$$\frac{\partial\mathbb{L}}{\partial\bar{\mathbf{w}}} = \frac{\partial\mathbb{L}}{\partial c_m}z_m \tag{3.16}$$

donde $\frac{\partial\mathbb{L}}{\partial c_m}$ por simplicidad se denotará como δ_m y M es la cantidad de tensores de salida [67, 74]. Cada capa tiene su propia salida definida por eq. 3.13. Si se generaliza para todas las capas en función de δ entonces:

$$\delta_h = g'(c_h)\sum_m w_{hm}\delta_m \tag{3.17}$$

donde $g'(*)$ es la primera derivada de la función de activación, . La eq. 3.17 indica que el valor δ_h depende directamente del valor de δ_m de la capa que le antecede y de los pesos w_{hm} de los enlaces que unen esas capas. Evaluar δ_m para la capa de salida se reduce a $(\bar{\mathbf{R}} - g(\sum_{m=1}^{M} w_{h,m}z_h + b_m))$ por tanto es el primer error con el que se puede empezar la *propagación hacia atrás* (del inglés *Back-Propagation*, en adelnate BP). Consta de 4 etapas:

1. Calcular δ_m simplemente usando $(\bar{\mathbf{R}} - g(\sum_{m=1}^{M} w_{h,m}z_h + b_m))$.
2. Calcular δ para cada neurona en las capas ocultas usando eq. 3.17.
3. Obtener los $\nabla\bar{\mathbf{w}}$ usando eq. 3.15 y los valores de δ.
4. Actualizar todos los valores de $\bar{\mathbf{w}}$ usando eq. 3.14 y $\nabla\bar{\mathbf{w}}$.

En la Figura 3.6 se muestra el proceso de BP comenzando por la capa de salida hasta las entradas buscando mejorar los pesos $\bar{\mathbf{w}}$:

El MLP puede mejorarse a si mismo usando FF y BP. Primero seleccionando aleatoriamente unos valores iniciales de $\bar{\mathbf{w}}$ y b, aplicando FF para propagar los tensores de entrada, fijando y calculando las salidas de cada neurona $\bar{\mathbf{z}}$. Al finalizar se ejecuta el BP y manteniendo constantes las $\bar{\mathbf{z}}$ se modifican los pesos $\bar{\mathbf{w}}$. Para el caso de los valores de umbral se utiliza el mismo sistema de actualización considerando que los pesos de los "enlaces"(señalado de esta forma pues suelen ser neuronas sin enlaces) es una constante con valor -1[70].

El criterio de convergencia se puede asumir de distintas maneras dependiendo del problema a representado. Por ejemplo se puede imponer una cota mínima a

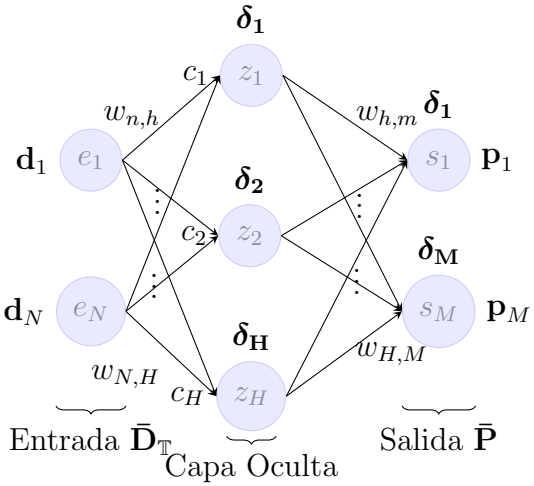

Figura 3.6: Representación de un MLP donde muestran las variables que intervienen en el proceso de BP, donde se actualizan los pesos a partir de los errores de salida.

alcanzar por la función de pérdida $\mathbb{L}$ en la capa de salida o se pueden usar un número fijo de iteraciones (si se considera una iteración como la ejecución de FF y BP una vez cada uno) [70, 74].

3.2.3 Funciones de Activación

Las funciones de activación indican si la neurona va a producir una salida o no. En su forma mas simple adquiere una definición binaria de la forma:

$$g(c) = \begin{cases} 1 & \text{if } c \geq 0, \\ 0 & \text{if } c < 0 \end{cases} \tag{3.18}$$

Existe un grupo importante de funciones de activación dependiendo de la aplicación. Cumplen con esta característica de retornar valores entre $[0, 1]$ o entre $[-1, 1]$ de ahí que sean *no lineales*. Además tienen que ser continuas y derivable, para que el proceso de BP se pueda ejecutar. Algunas relevantes para este trabajo se detallan brevemente a continuación [66, 67, 70].

Sigmoide (*Sigmoid*)

Es probablemente la mas utilizada para problemas de clasificación, junto con otras de similar comportamiento pertenecientes a la familia de las *tangentes hiperbólicas*. Este tipo de función a menudo se le denomina *aplanadora* (del inglés *squashing*), por limitar la salida a un grupo finito de valores en un rango determinado. Para

este caso. las salida oscilan entre $[0, 1]$. Está definida como:

$$g(x) = \frac{1}{1 + e^{-\beta x}} \tag{3.19}$$

donde β es un parámetro para determinar la pendiente de la función. Uno de los problemas de esta función es que valores muy grandes son aplanados a 1 y los muy pequeños a 0, esto provoca problemas en el proceso de BP, ya que una vez que se aplanan, ajustar correctamente los pesos resulta muy difícil.

Unidad Lineal Rectificada (ReLU).

Su principal ventaja es la simplicidad de su evaluación una simple comparación. Esto la ha convertido en la función de activación mas utilizada para NN de alta dimensionalidad. Propuesta en el trabajo de [75, 76] se define como:

$$g(c) = max\{0, c\} \tag{3.20}$$

y su rango va $[0, \infty)$. Esta función no tiene parte negativa lo que desplaza su valor medio creando en ocasiones problemas en el proceso de aprendizaje [77].

Unidad Lineal Exponencial (ELU)

Surge como una alternativa o versión a la función ReLU[77]. Su enfoque fundamental es corregir los problemas que presenta ReLU sobre la no negatividad de la activación mientra mantiene las fortalezas. Está definida como:

$$g(\alpha, c) = \begin{cases} \alpha(e^c - 1) & \text{if } c < 0, \\ c & \text{if } c \geq 0 \end{cases} \tag{3.21}$$

donde α es un parámetro que controla cuanto negativa puede ser la función. Por tanto su rango de valores oscila entre (α, ∞). eq. 3.21 mantiene la misma forma que eq. 3.20 para valores positivos y realiza una corrección parametrizada para los negativos.

3.2.4 Redes Neuronales Convolucionadas

La Redes Neuronales Convolucionadas cambian la estructura de las capas ocultas a una matriz de dimensiones MxN. En una sección de la matriz se representa una pequeña parte de los tensores de entrada, de esta forma una misma capa oculta contiene información de varios tensores de entrada. Los cuales pueden solaparse y compartir pesos correlacionanado los valores mas cercanos entre ellos que con los mas alejados.

La Figura 3.7 se puede visualizar como dos subsecciones, A y B, se solapan en la capa oculta h. Los pesos $w_{n,m}$ son compartidos por ambas subsecciones. La convolución de la capa se aplica antes de que se evalue la función de activación $g(*)$ y propagar a la siguiente capa.

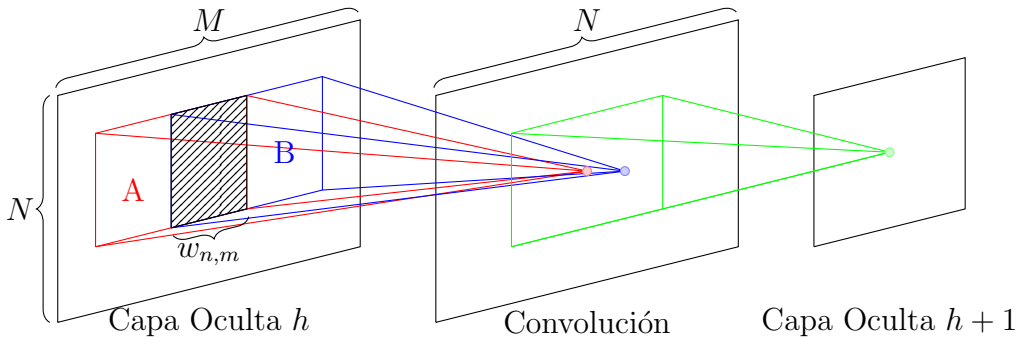

Figura 3.7: Representación de una CNN. Se puede apreciar el solapamiento en la capa donde se aplica la convolución.

Los procesos de FF y BP se ejecutan de una manera similar al MLP, pero pueden producir sobrecargas al procesamiento si el número de neuronas por capa aumenta considerablemente [70, 74, 78, 79].

3.3 Optimización de Hiper-parámetros.

A lo largo de este capítulo se han descrito métodos dependientes de parámetros independientes de las entradas, que buscan generalidad en las soluciones que proponen. La búsqueda de las mejores configuraciones está presente, siempre que exista la posibilidad de elección de estos parámetros.

Si se define $f(\alpha; \bar{\mathbf{D}}_{\mathbb{T}})$, representando una función de aprendizaje, se busca el valor de α que minimice la función de pérdida $\mathbb{L}(f(\alpha; \bar{\mathbf{D}}_{\mathbb{T}}); \bar{\mathbf{R}}_{\mathbb{T}})$. Aunque pueda parecer similar al proceso de aprendizaje, el objetivo es diferente. Se puede considerar la evaluación de $\mathbb{L}$ como una *caja negra* en función de α de la forma:

$$J(\alpha) = \mathbb{L}(f(\alpha; \bar{\mathbf{D}}_{\mathbb{T}}); \bar{\mathbf{R}}_{\mathbb{T}}) \tag{3.22}$$

solo interesa el resultado de evaluar $J(\alpha)$ sin tener en cuenta como se llegó a ese resultado.

Lo mas simple para resolver esta optimización hacer una *búsqueda exhaustiva*[10], estableciendo un rango de valores para α, de la forma $[\alpha_0, \ldots, \alpha_M]$ y considerando

[10]La literatura reconoce este proceso con el nombre de *Búsqueda en una Malla* (del inglés *Grid Search*), donde se genera una malla con los posibles valores de las variables y se evalua el modelo en cada uno de los puntos.

que existe un mínimo global en ese rango. Esto implica ejecutar $J(*)$ M veces, pero cada iteración no garaniza que el modelo mejore. Al finalizara se pueden buscar los mínimos en el rango $j_0, \ldots, j_M$ de resultados.

Es deseable encontrar el mínimo, evaluando $J(*)$ la menor cantidad de veces en el intervalo $[\alpha_0, \ldots, \alpha_M]$ o que con cada evaluación el modelo mejore. Las *Búsquedas Bayesianas* son un método bien establecido que a diferencia de los métodos basados en gradientes ∇ no requiere cálculo de derivadas de la función $J(*)$ [80, 81, 63].

Las Búsquedas Bayesianas se basan en dos conceptos importantes: (1) *Modelo Estadístico Bayesiano* y (2) *Función de Adquisición*.

El Modelo Estadístico Bayesiano se utiliza para construir una curva de distribución de probabilidades, que permita conocer aquellas α dentro del rango de búsqueda donde es mas probable que la función $J(*)$ mejore. Es habitual que se utilice una técnica conocida como *Modelos Sustitutos* (del inglés *Surrogate Models*, en adelante SM) donde se cambia la función objetivo ($J(*)$) por una menos costosa de evaluar que siga la misma distribución [63]. Cada nuevo punto se evalua en el SM (menos costoso) primero antes de probar en la función $J(*)$ (más costosa) solo si la primero evaluación resulta favorable.

La Función de Adquisición se encarga de buscar el siguiente punto basándose en SM y donde no se ha buscado antes. Se pueden expresar estos conceptos en un breve algoritmo:

Algorithm 3.2: Búsqueda Bayesiana

Require: Buscar en rango de $[\alpha_{min}, \alpha_{max}]$ en M iteraciones.
1: Evaluar $J(\alpha_0)$ con α_0 valor inicial.
2: Inicializar el SM usando las evaluaciones $J(\alpha_0)$.
3: **while** $m \leq M$ **do**
4: Generar α_m usando Función de Adquisición y SM.
5: Evaluar $J(\alpha_m)$.
6: Actualizar SM.
7: Incrementar $m++$.
8: **end while**

El resultado esperado será el punto α_m donde $J(\alpha_m)$ es mínimo en el intervalo dado. Con la ventaja de no evaluar todos los puntos en el intervalo y asegurar que cada nuevo punto evaluado en $J(*)$ va a mejorar el modelo.

3.4 Preprocesamiento de Datos.

En varias ocasiones las mediciones realizadas sobre Propiedades pueden presentar múltiples problemas. El preprocesamiento ayuda a homogenizar las mediciones de manera que el modelo se construya correctamente [67, 74].

Una técnica recomendada es la normalización de los datos, para cuando las mediciones presentan cambios abruptos en los rangos de valores o no tienen una distribución uniforme entorno a la media. Este problema puede afectar directamente los algoritmos de aprendizaje, ya que pueden decantarse por aquellas mediciones de mayor valor absoluto (no necesariamente las mas representadas) depreciando las de pequeño valor.

Una normalización muy simple es eliminar la media y la varianza de las mediciones realizada. Los datos resultantes tendrían media 0 y varianza unitaria. Otro tipo de normalización es el escalado de los datos ya sea $[0, 1]$ o $[-1, 1]$ ambos extremos inclusive (ver Capítulo 4.2, para una descripción teécnica del método utilizado).

Un problema complicado es el de *datos incompletos*. Como la mediciones en ocasiones dependen de fenómenos cambiantes no siempre se capturan toda la información. Existen varias técnicas para corregir esto la mas simple sería descartar los valores *defectuosos*[11]. Pero cuando existen pocas mediciones descartar algo puede convertirse en un lujo.

Otra técnica es el de sustitución de los valores faltantes, que puede ser con 0 (del ingés *zero-padding*), utilizando valores de la media calculados a partir de los datos disponibles o algún mecanismo mas elaborado donde se analiza la distribución de los datos y se busca el valor mas adecuado que responda a esa distribución [67].

[11] Se pueden considerar como valores defectuosos, aquellos que se desvían por un amplio margen de la distribución de los datos. La literatura los reconoce también como *out-liers*.

Capítulo 4

Revisión de la solución técnica.

Nature achieves much more with less,
Some guy, in a Youtube video about Robotics .

Contents

Este Capítulo abordará los aspectos técnicos de esta investigación. En las siguientes Secciones se describirán el código utilizado para realizar los cálculos de TD-DFT, las bibliotecas externas, el lenguaje de programación seleccionado y por último una breve descripción del código implementado para producir los resultados que justifican este trabajo. Antes de entrar en detalle se introducirán algunas siglas y terminología necesaria:

- Programa (del inglés *Software*): Secuencia de código que puede ser ejecutada por un ordenador.
- Bibliotecas (del inglés *Frameworks*): Programa o Conjunto de Programas re-utilizables que ofrecen un grupo de funcionalidades en un área de la Programación. Ayudan a no repetir código, agilizan mucho el desarrollo de software y evitan los errores ya que han sido probados y optimizados.
- UML: Lenguaje Unificado de Modelaje, conjunto de símbolos y diagramas para representar los componentes de un Programa.

Importante destacar todos los Programas externos mencionados en este trabajo se distribuyen bajo la licencia GNU.

4.1 Código para cálculos TDDFT. Octopus.

Para los cálculos TDDFT se utilizó el programa Octopus [82]. Un programa que se distribuye bajo la licencia GPL, por lo que es gratuito para descargar[1] y libre de modificar. En el mismo sitio de descarga se puede encontrar un manual para usuarios que se inician, así como varios tutoriales que ayuden a entender como representar las variables físicas para realizar los primeros cálculos.

Fue creado en el grupo del Profesor Ángel Rubio a principios del siglo XXI. Este programa está escrito en *Fortran 95* y algunos módulos en *C*. Hasta la fecha ha evolucionado en una herramienta bien completa y eficiente. A través de varias API, Octopus es capaz de ejecutarse en varios ordenadores simultáneamente, alcanzando un buen grado de paralelización, lo que agiliza las tareas de cálculo notablemente.

Una de las principales características de Octopus es su implementación de las ecuaciones de Kohn-Sham dependientes del tiempo. Estas ecuaciones se resuelven sobre una discretización del sistema en una maya homogénea en el espacio real y toda la dinámica se realiza en tiempo real.

Después de establecer condiciones iniciales y posiciones de las partículas involucradas, Octopus calcula el Estado Fundamental (GS) y a este estado se le pueden aplicar distintos tipos de perturbaciones (mecánicas, electromagnéticos, etc.). Este mecanismo es el utilizado para calcular estados excitados y el espectro de absorción entre otras propiedades electrónicas.

El uso de una maya homogénea en el espacio real para la representación de la magnitudes y partículas involucradas en el sitema físico, le permite a Octopus tener un grado arbitrario de precisión en el cálculo. Simplemente disminuyendo la distancia entre puntos de la maya se puede mejorar la calidad de los resultados a expensas de un mayor coste computacional.

Para algunas tareas Octopus delega el trabajo en bibliotecas externas, tal es el caso de LibXC[2] [83]. Esta es una biblioteca solamente enfocada a la implementación de los funcionales de energía de intercambio y correlación E_{xc} (ver Sección 2.1.1).

Actualmente se encuentra en su versíon 5 y la lista de aproximaciones de E_{xc} cubre la mayoría de las descritas en la literatura en los últimos 50 años.

[1] http://tddft.org/programs/octopus/

[2] Accesible desde https://www.tddft.org/programs/Libxc/.

4.2 Descripción de la solución implementada.

El código desarrollado involucra los conceptos expuestos en los Capítulos 2 y 3. En la Figura 4.1 se muestran los principales componentes del código, el cual se puede dividir en 3 grupos diferenciando funcionalidades:

- Manejo de Datos: Compuesto por 3 clases, encargadas de suministrar información sobre los compuesto con los que se trabaja, (1) *Oct-Data* maneja los resultados de los cálculos *ab-intio* realizados con Octopus, (2) *Mol* mantiene información referida a una molécula y (3) *MolData* maneja una lista de muchas moléculas.

- Aprendizaje Automático: Contiene 2 módulos, (1) Redes Neuronales y (2) Regresión Ridge con Kernel.

- Bibliotecas Externas: Varias bibliotecas que facilitaron las optimizaciones de los métodos de ML y la manipulación y visualización de datos.

La Figura 4.1 muestra un diagrama *UML* con alguna de las clases mas relevantes contenidas en el código desarrollado. La relación entre ellas marca la dependencia de funcionalidades.

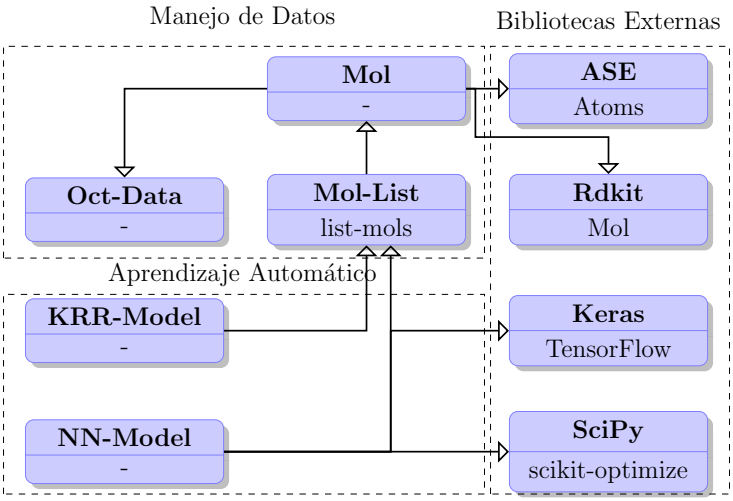

Figura 4.1: Modelo UML de las clases y Bibliotecas externas utilizados en este trabajo.

En las siguientes secciones se ampliarán los detalles de cada uno de estos componentes.

4.2.1 Manejo de Datos.

Resulta importante en trabajo con *Compuestos Químicos* contar con Programas y/o Bibliotecas que permitan un representación estándar de dichos compuestos. Se busca un grupo de características que ayuden a visualizar lo mas cercano a la realidad posible. Algunas funcionalidades deseables son:

- Capacidad de manejar varios compuestos diferentes simultáneamente y su visualización[3].
- Capacidad de manejar compuestos con gran cantidad de elementos Químicos.
- Capacidad de ofrecer algún mecanismo de edición de compuestos, agregar o eliminar elementos, modificar enlaces, ángulos y ángulos de torsión.

Entorno de Simulación de Atomos (ASE)[84].

Completamente desarrollado en el lenguaje de programación Python y distribuido bajo la licencia GNU, ASE está diseñado con una estructura modular de Biblioteca, que permite trabajar con fragmentos del programa sin la necesidad de utilizarlo todo. Posee también una interfaz visual, sobre la cual se puede interactuar con los compuestos, visualizar datos físicos y modificarlos. Además de una herramienta para trabajar a partir de comandos que pueden encadenarse con otros programas. Todas las tareas que ASE realiza son accesibles a través de un programa en Python, desde donde resulta muy cómodo[4] interactuar con todos sus componentes.

En ASE la clase sobre la que descansa es *Atoms*. Esta clase contiene en si mismo un grupo importante de funcionalidades, incluidas las encargadas de modificar todos los elementos de los compuestos con los que se está trabajando.

RDKit[85].

RDKit es un programa completamente desarrollado en *C++*, sin embargo posee una interfaz en Python desde la cual se puede acceder a una gran gama de funcionalidades. A diferencia de ASE no cuenta con interfaz visual ni con una línea de comandos. RDKit tiene que ser utilizado desde un programa, ya sea en C++ o en Python.

[3]inclusive si es poco elaborada, la visualización en 3D de compuestos ayuda a comprender la estructura interna.

[4]Es necesario conocer el lenguaje de programación Python.

Una de sus características es la posibilidad de construir compuesto a través de cadenas de caracteres conocidas como *SMILES*[5]. Esta es una representación bien simplificada de un compuesto donde solamente aparecen los elementos químicos y los enlaces entre ellos. A partir de esto RDKit puede reconstruir el compuesto en 3D mediante un método semi-empírico usando datos tabulados y estadísticos [86].

RDkit descansa en la clase *Mol* (en adelante *RDKit-Mol*) sobre el cual realiza todas las operaciones. Cada objeto RDKit-Mol contiene información de los átomos que lo componen así como de los enlaces entre estos.

Ambos programas tanto RDKit como ASE, se solapan y complementan. Ambos realizan optimizaciones de estructuras. RDKit asume un enfoque Químico manejando el concepto de enlace entre átomos para lo cual tiene implementada una estructura capaz de manejar enlaces simples, dobles, triples, etc. ASE asume un enfoque Físico donde el concepto de enlace se mantiene implícito basándose en la distancia entre elementos para la cual tiene un umbral.

La clase **Mol** implementada contiene dentro de si un objeto Atoms de a ASE y uno RDKit-Mol perteneciente a RDkit. Ambos representan el mismo compuesto y enriquecen la información. La clase **Mol** también contiene un objeto de la clase **Oct-Data** y a través de esta accede a los cálculos DFT/TDDFT realizados para ese compuesto.

La clase **Mol-List** mantiene una lista de **Mol** e implementa facilidades para construir estructuras de datos con la información de de cada **Mol** en la lista.

4.2.2 Aprendizaje Automático.

Este paquete está compuesto por dos módulos: (1) *KRR-Model* y (2) *NN-Model*. Ambos representan a los modelos evaluados en este trabajo.

Implementación de KRR-Model.

La implementación de la Regresión Ridge con Kernel (del inglés *Kernel Ridge Regresion*, ver Sección 3.1.1), es propia y está basada en modelos utilizados en la bibliografía consultada [4, **?**]. Consta de dos pasos. Primero la construcción del kernel K a partir del conjunto de entrenamiento $\bar{\mathbf{D}}^{\mathbb{T}}$ utilizando la expresión:

$$k_{ij} = f(|d_i^{\mathbb{T}} - d_j^{\mathbb{T}}|), \qquad i, j = 1, ..., N \tag{4.1}$$

donde $|*|$ es la norma y $f(*)$ es la función de construcción del kernel y la que determina la naturaleza de la distribución (ver Sección 3.1). El segundo paso es la

[5]En https://www.daylight.com/dayhtml/doc/theory/theory.smiles.html está descrita extensamente la notación SMILES.

obtención de los coeficientes **c** que serán utilizados en las estimaciones posteriores se realiza resolviendo la ecuación:

$$\bar{\mathbf{R}}^{\mathbb{T}} = (\mathbf{K} + \lambda \mathbf{I})\mathbf{c} \tag{4.2}$$

donde $\bar{\mathbf{R}}^{\mathbb{T}}$ representan las referencias del conjunto de entrenamiento $\mathbb{T}$, **K** es el kernel construido utilizando eq. 4.1, λ es el hiper-parámetro de regularización y **I** es la matriz de identidad. Por último, después de construir el kernel y obtener los coeficientes, la estimación p para el elemento i se obtienen evaluando:

$$p_i = \sum_{t=1}^{N} c_t * \exp(-\frac{|d_i^{\mathbb{C}} - d_t^{\mathbb{T}}|}{\sigma}) \tag{4.3}$$

donde N es la cantidad de elementos en el tensor $\bar{\mathbf{D}}^{\mathbb{C}}$. La eq. 4.3 tiene una versión utilizada cuando múltiples descriptores son evaluados. Esto requiere la combinación de ambos kernel distintas σ, una para cada descriptor utilizado (ver Sección 6.3.2).

$$p_i = \sum_{t=1}^{N} c_t * \sum_{j=1}^{J} f(|d_{i,j}^{\mathbb{C}} - d_{t,j}^{\mathbb{T}}|) \tag{4.4}$$

donde J representa la cantidad de descriptores. El kernel utilizado para la obtención de los coeficientes **c** es una combinación lineal de cada uno de los kernel obtenidos para cada descriptor [87].

Implementación de NN-Model.

Los algoritmos de NN no son complicados de programar. Sin embargo contar con una implementación eficiente y escalable si resulta más difícil de alcanzar. Requiere de muchas pruebas para evitar errores comunes y tener certeza de la eficiencia. Resulta mas viable incorporar bibliotecas ya probadas, con *casos de uso* implementados. Se busca en estas bibliotecas la flexibilidad para representar problemas variados, y la potencia de ejecutarse en varias configuraciones de ordenadores, incluidos *Super-Ordenadores* (del inglés *High Performance Computer*).

TensorFlow[88].

TensorFlow es una Biblioteca para el trabajo con redes neuronales y donde es posible representar otras máquinas de Aprendizaje Automático. Desarrollado y mantenido por Google, posee varias interfaces para otros lenguajes como Python, C++, Java y Java Script.

Su esquema de trabajo se basa en construir un flujo de datos basado en un grafo conectado, sobre el cual se realicen las transformaciones pertinentes (ver Sección 3.2). Su característica mas relevante es la capacidad de paralelización que posee, permitiendo su uso en super-ordenadores de alta capacidad de cálculo

TensorFlow ofrece un grupo de herramienta empaquetadas dentro de un modulo de nombre *TensorBoard* enfocado solamente en el análisis de las topologías construidas con TensorFlow. Permite hacer *profiling* para determinar donde hay problemas y los ritmos de aprendizaje de la red construida.

Keras[89].

Keras es lo que se conoce como *Front-End* (Fachada). Está desarrollado en Python y solo puede utilizarse con este lenguaje. Por si sola no implementa ningún método, se enfoca en hacer el trabajo con el *Back-End* mucho más simple (Keras soporta a varios tipos de Back-End entre ellos TensorFlow). La homogeneidad con la que se construyen los modelos hace muy simple su entendimiento independiente del Back-End utilizado.

El conjunto de estas dos Bibliotecas agilizan el trabajo con Redes Neuronales, alejando al usuario de los posible entramados y complicaciones de una implementación propia y enfocando los esfuerzos a la obtención de resultados. Las Figura 4.2 se pueden apreciar las tres topologías construidas y evaluadas sobre estas bibliotecas. Las nombradas MLP (Del Inglés Multi Layer Perceptron) representan variantes del mismo modelo, donde se modificó la manera en la que los datos son alimentados a la red combinando las entradas en un sólo tensor. La nombrada CNN (Del Inglés Convolutional Neural Network) utiliza también la combinación de las propiedades de entrada.

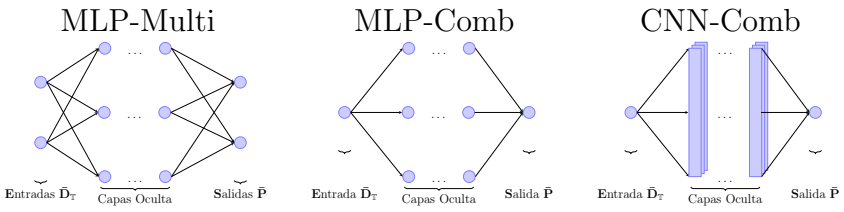

Figura 4.2: Topologías constridas sobre Keras y Tensorflow.

4.2.3 Bibliotecas Externas.

El código desarrollado se apoya en módulos externos accesibles desde el gestor de programas de Python. Específicamente *SciPy*, un Framework muy completo para

el trabajo científico. Resulta complicado y extenso describir todo lo que contiene SciPy pero se pueden mencionar varios módulos importantes:

- NumPy: Biblioteca para manejar vectores, matrices y tensores. Contiene un conjunto de funcionalidades básicas como multiplicación de matrices, solución de vectores propios.
- SciKit: Conjunto de bibliotecas con múltiples funciones que van desde el Aprendizaje Automático hasta el trabajo con Ondas de Radio, Estadística, métodos de integración, derivación e interpolación, etc.[6].
- Matplotlib: Biblioteca para realizar representaciones gráficas de datos y funciones.

Dentro de SciKit se encuentra el paquete de optimizaciones *scikit-optimize* que implementa las Búsquedas Bayesianas (ver Sección 3.3). Utiliza como Modelo Sustituto uno Gaussiano de ahí que se denomine *Gaussian Process*[81, 90].

SciKit también posee un conjunto de facilidades para el preprocesamiento de datos (ver Sección 3.4). Dentro del sub-módulo *sklearn.preprocessing* se incluye una clase para el escalado de lo datos que implementa la expresión:

$$x' = \frac{x - min(x)}{max(x) - min(x)} * (max - min) + min \tag{4.5}$$

siendo x el tensor de entrada, max y min los extremos del rango a donde el tensor x se pretende normalizar (usualmente $[0, 1]$)[7].

[6]Para una lista de los todos los módulos http://scikits.appspot.com/scikits

[7]Al ser público el código de SciPy el lector puede acceder a él e inspeccionarlo. Esto garantiza que que la literatura coincida con la implementación

Capítulo 5

Estimación de la Geometría Molecular.

Contents

En este Capítulo se abordará la propuesta de un modelo de Aprendizaje Automático para la optimización de geometrías publicado en [91]. Previamente en la Sección2.3 se hizo una introducción a la Optimización de Geometrías como primer paso antes de efectuar cálculos TDF/TDDFT. Este proceso suele ser costoso en

tiempo ya que requiere de múltiples iteraciones y en cada iteración se necesita la solución completa de las Ecuaciones de Kohn - Sham (Ver Sección 2.1). Además el uso del funcional de la energía de intercambio y correlación E_{xc}, necesario para la solución de las Ecuaciones de Kohn - Sham determina la calidad y el coste de la optimización. Por tanto todo lo que ayude a minimizar la cantidad de iteraciones y resolver la eq. 2.1 se aplica activamente.

La optimización de geometrías se compone de dos procesos: (1) cálculo de la energ'ia potencial para construir la PES, literatura reconoce varios método mas o menos efectivos dependiendo siempre de la precisión que se desee desde *semi-empíricos* donde se sustituye el cálculo de la Energía potencial por una solución menos precisa[61], y DFT con la utilización del funcional de la energía de intercambio y correlación, mucho mas complejo y preciso[55, 60, 62]. El segundo proceso es el algoritmo para recorrer la PES en la búsqueda del mínimo donde varios algoritmos de optimización son utilizados[63, 56, 92].

RDKit [85] al igual que ASE [84] implementan varios tipos de algoritmos de optimización.

RDKit implementa las optimizaciones sobre una matriz de distancias construida imponiendo restricciones en los ángulos de torsión a partir de análisis de datos en cristales(*Experimental-Torsion Knowledge Distance Geometry* ETKDG) [86, 93]. La matriz de distancias (del inglés Distance Geometry) se inicializa de forma aleatoria y las coordenadas son obtenidas a través de los valores propios después de una arbitraria cantidad de iteraciones del método [94]. Las restricciones de ETKDG fueron obtenidas de datos experimentales almacenados en varias bases de datos cristalográfico de pequeñas moléculas [86]. La matriz de distancia es refinada utilizando *UFF* (del inglés Universal Force Field) [59] o *MMFF* (del inglés Merck Molecular Force Field) [58].

ASE por otra parte implementa *Broyden–Fletcher–Goldfarb–Shanno* (BFGS) [63] y una versión de este mismo algoritmo más ligera computacionalmente, el L-BFGS [63]. Ambos basados en el uso de gradiente.

Existen también aproximaciones usando Aprendizaje Automático. Tal es el caso del uso de una red neuronal entrenada sobre un grupo de molŕeculas diferentes. La red propuesta es capaz predecir la geometría de posibles conformaciones estimando la energía de cada conformación, la cual se corrige utilizando un método semi-empírico [11].

En este Capítulo se propone un método de optimización de geometrías basado en un modelo de aprendizaje autoḿatico soportado sobre una unidad básica de construcción al que se le dominará *bloque*. Partiendo de estos *bloques* se pretende reconstruir las moléculas basandose sólo en información de conectividad como por ejemplo los *SMILES* (del inglés *Simplified molecular-input line-entry system*). Se modificó el programa RDKit para que asimilara ángulos de torsión provenientes

de una base de datos *ab-initio*. Para la descripción de los métodos se utilizarán las definiciones introducidas en el Capítulo 3.

Salvo que se especifique otra magnitud, todas las energías están en *electron-Volts* (eV) y las distancias en Angstrom ($\mathring{A}$).

5.1 Selección del Descriptor Molecular.

Existe una gran variedad de *Descriptores Moleculares*. La importancia de seleccionar correctamente el que mejor se ajuste al problema influye directamente en los resultados. Existe un grupo de características deseables en un descriptor tales como, (1) identificar unívocamente el objeto, (2) la variabilidad del descriptor tiene que reflejarse en la propiedad a predecir[1], (3) calcular el descriptor no debe ser más complicado computacionalemente que la propiedad a la que se pretende estimar, (4) mantener la dimensionalidad lo más baja posible [95, 29, 96, 25].

La biblioteca RDKit (ver Sección 4.2.1) contiene la implementación de varias *Huellas* (del inglés Fingerprints) y descriptores[2].

Dentro de los descriptores implementados en RDKit se encuentra el *e-state* (contracción del nombre en inglés Electro-topological States)[97, 98]. Este descriptor cumple con las características mencionadas con énfasis particular en la simpleza. El e-state S está definido como la combinación de un valor *intrínsico* I de cada átomo y una contribución ΔI de los vecinos de dicho átomo de la forma:

$$S = I + \Delta I \tag{5.1}$$

donde el valor intrínsico I puede ser obtenido por:

$$I = \frac{\delta^v + 1}{\delta} \tag{5.2}$$

donde δ^v es la cantidad de electrones de valencia y δ es la cantidad de electrones con enlaces sigmas. La perturbación ΔI entre un átomo y su vecino puede obtenerse usando:

$$\Delta I = \sum_{j=1}^{N} \frac{I_i + I_j}{r_{i,j}^2} \tag{5.3}$$

donde $1/r_{i,j}^2$ mide la posible interacción entre 2 átomos, y $r_{i,j}$ es el número de átomos que separa el elemento i del j. Usando esta interacción se puede aumentar

[1] Resulta en lo que se denomina problema *ill-posed*, cuando pequeñas variaciones en la distribución de una propiedad afectan al modelo de manera drástica, haciendo muy difícil el aprendizaje [66]

[2] ver https://rdkit.org/docs/GettingStartedInPython.html

la precisión del e-state expandiendo la distancia $r_{i,j}$ a segundos y terceros vecinos, si la cantidad de átomos de la molécula lo permite, lo cual resulta pertinente si los valores de I resultan muy similares.

Esta propuesta de descriptor resulta muy sencilla de calcular. El valor de S puede estar asociado a un átomo, haciendo 0 el valor de ΔI, o de una conformación de átomos calculando ΔI. Otra característica positiva es que no tiene dependencias de la ubicación espacial de los átomos, solo de sus conexiones. La lista de valores I están pre-establecidos, dejando solamente el cálculo de las contribuciones de los vecinos ΔI [97, 98].

5.2 Análisis de la Base de Datos: Distribución de Propiedades.

La base de datos para las pruebas está formada por 21k pequeñas moléculas orgánicas de hasta 8 átomos distribuidos en Carbono (C), Oxígeno (O), Nitrógeno (N), Flúor (F) y los correspondientes átomos de Hidrógeno (H). En adelante se le denominará *8CONF* [4, 99, 100, 11]. En la Figura 5.1 se muestran la distribución por número de átomos (Figura 5.1 A) donde la mayor cantidad de moléculas tienen entre 10 y 20 átomos incluidos los hidrógenos, por tipo de átomo dentro de la 8CONF (Figura 5.1 B) donde se han omitidos los átomos de Hidrógeno y moléculas lineales y con anillos de hasta 8 átomos (Figura 5.1 C), el conteo tan alto se debe a que existen moléculas, con más de un tipo de anillo.

Esta base de datos ha sido optimizada previamente, utilizando métodos DFT, con el funcional híbrido B3LYP[101] para la aproximación de E_{xc} con funciones de base 6-31G(2df, p)[4].

Al contar con coordenadas cartesianas optimizadas, se puede explorar la composición estructural de la 8CONF en términos de anillos, tipos de enlaces y distancia de enlaces, ángulos y ángulos de torsión (*coordenadas internas*, ver Sección 2.3). Se persigue conocer en profundidad las estructuras presentes. En la Figura 5.2 se puede observar las distribuciones de algunos valores de coordenadas internas presentes en la 8CONF.

Las distancias entre átomos están bien determinada por el tipo de enlace, puede ser el resultado de las optimizaciones realizadas usando DFT/B3LYP. Sin embargo los ángulos entre elementos y de torsión se ven afectados por los anillos presentes. Se puede apreciar en la Figura 5.2 *B* y *C* que una misma agrupación de átomos presenta una dispersión amplia de posibles valores. Un ejemplo de esto es *H-C-C-H* donde el enlace *C-C*, dependiendo si es simple, doble o triple (todos presentes en la 8CONF), tiene mas o menos grados de libertad en la rotación, lo que da como resultado la dispersión de valores tan grande mostrada en la Figura 5.2 *B*.

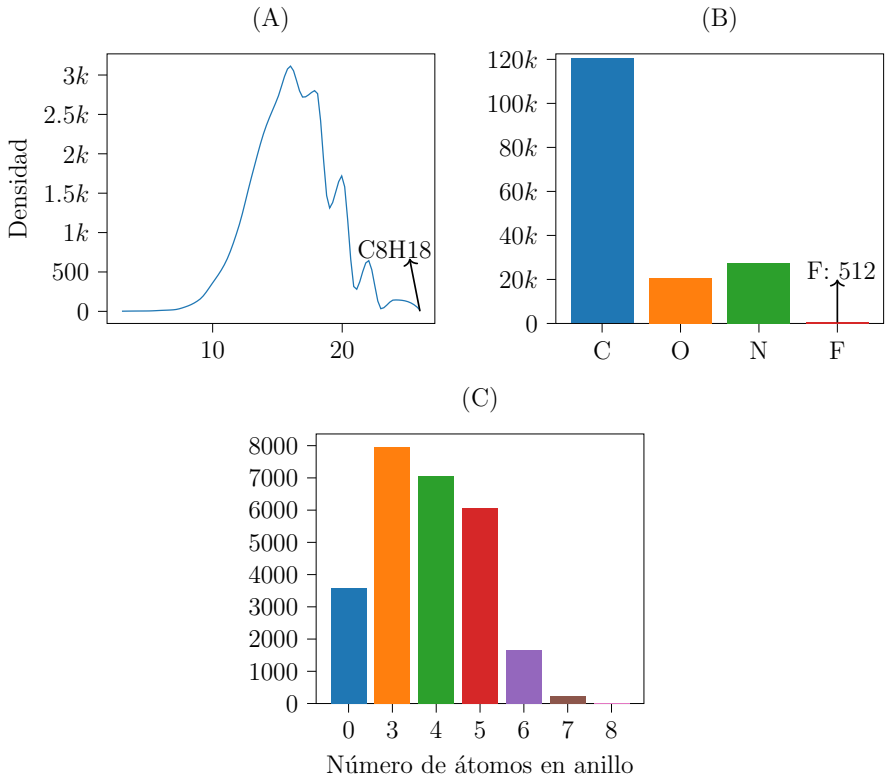

Figura 5.1: Distribución en la 8CONF: (A) número de átomos, (B) histograma por tipo de átomos donde se han omitidos los átomos de Hidrógeno y (C) moléculas lineales y con anillos de hasta 8 átomos.

En la Figura 5.3 se muestra la distribución de los valores del e-state por átomos dentro de la 8CONF. Comparar la correlación entre el descriptor *e-state* y las propiedades geométricas de las moléculas confirmaría.

5.2.1 Introducción del Concepto de Bloque.

Se considera como *bloque*: unidad con la que reconstruir las moléculas y contenga información necesaria de enlaces y ángulos atómico. Cada molécula dentro de la 8CONF puede aportar una determinada cantidad de *bloques*. Cada bloque aporta una cantidad de información dependiendo de su entorno en la forma de *coordenadas internas*[54]:

- Sujeto al tipo de átomo (C, O, N o F) se pueden obtener, tipo de enlaces y distancias de enlaces.
- Dependiendo de la cantidad de enlaces de cada átomo la disposición espacial

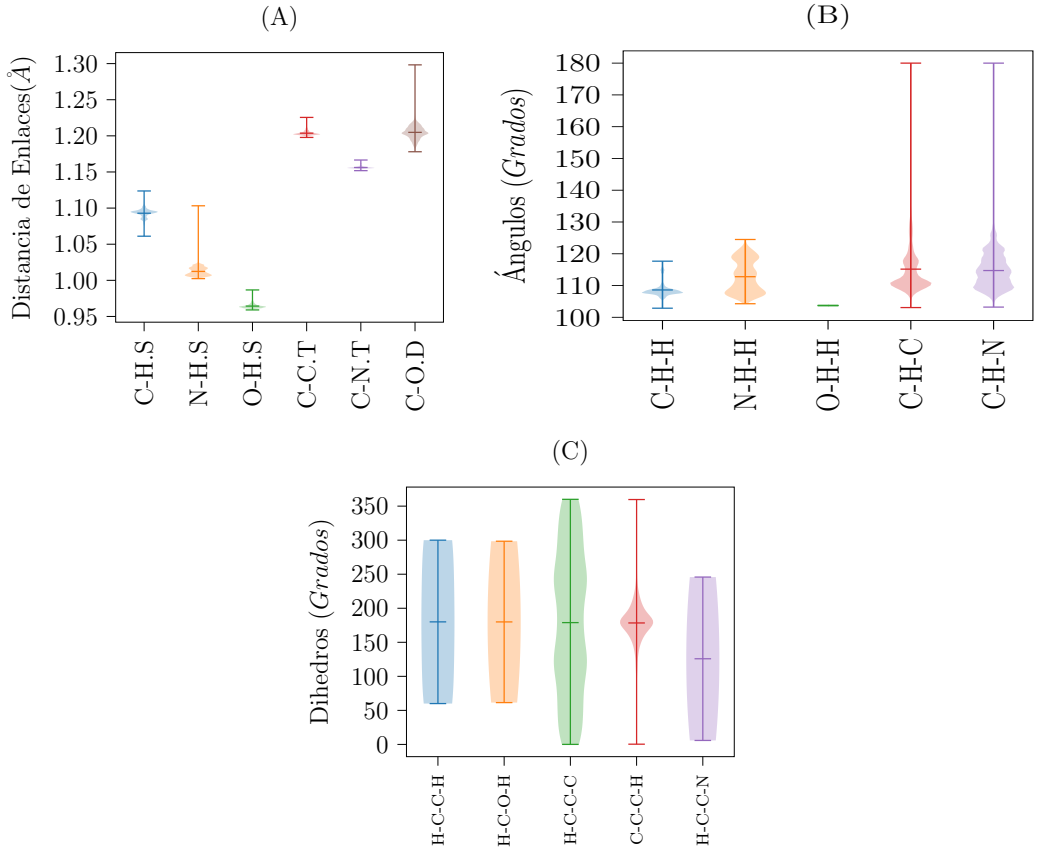

Figura 5.2: Distribución de alguna de las estructuras presentes en la 8CONF: (A) distancia de enlaces entre las especies, donde *S, D, T* son enlaces Simples, Dobles y Triples respectivamente, (B) ángulos entre alguna de las distintas especies presentes en las moléculas y (C) ángulos de torsión.

de los átomos al final del los enlaces y ángulos entre enlaces.

Cada bloque se une a otro bloque a través de dos *átomos redundantes*, presente en dos centros de cada bloque. Si se toma como ejemplo la molécula C_3H_8 (Figura 5.4), se puede observar como ésta aporta 3 bloques, 2 de los cuales se consideran del mismo grupo por contener los mismos elementos. Para clasificar los bloques y evitar que bajo el mismo grupo se junten bloques diferentes se creó una etiqueta que se construye teniendo en cuenta:

1. El átomo central es el primer átomo de la etiqueta.

2. Los vecinos inmediatos fueron organizados por *Número Atómico* de mayor a menor.

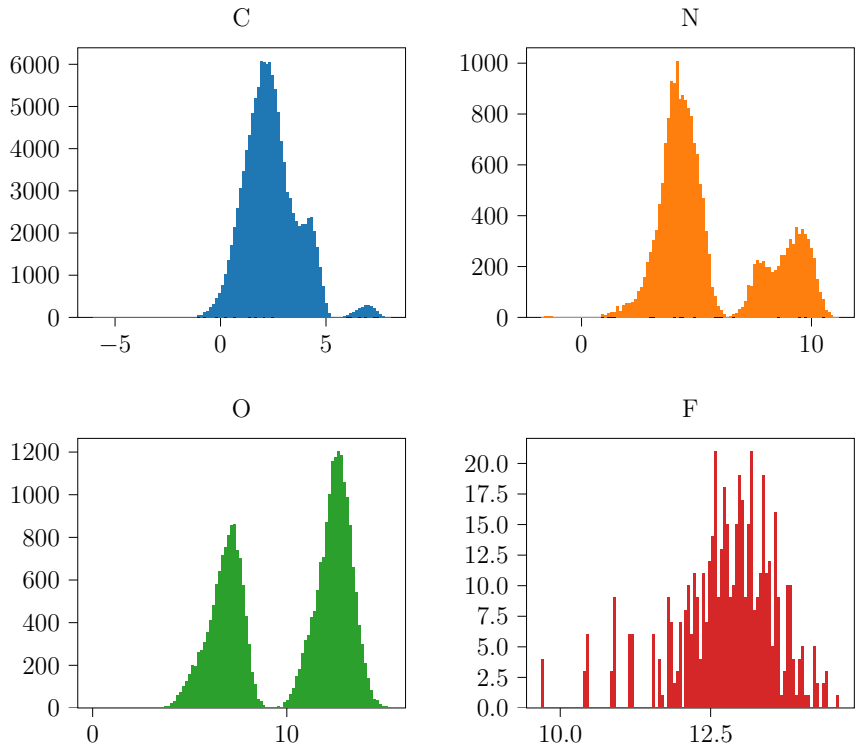

Figura 5.3: Histograma de la distribución de los valores del e-state para cada uno de los átomos dentro de la 8CONF. Se omite el Hidrógeno puesto que su valor tabulado es 0.

3. Dependiendo de la cantidad de enlaces del átomo se clasificó en: *Lineal* si es un solo enlace sin importar el tipo de enlace, *Triangular* si tiene dos enlaces diferentes y *Tetrahedro* si tienes tres enlaces diferente.

Para cada bloque se mantuvo la numeración heredada de la molécula desde la cual se extrajo, y la información de los enlaces.

Aplicando este concepto a la 8CONF se encuentran 68 tipos bloques diferente. Resulta evidente que no existirá una distribución homogénea por lo que algunos bloques serán más abundantes que otros. Los bloque que resultan en extremos de moléculas como *C-C-H-H-H* y los que forman encadenamientos como *C-C-C-H-H* resultan los más abundantes con 14511 y 20274 respectivamente. Por otra parte *C-N-H* y *C-F-F-F-F* solo tienen 1 bloque cada uno (ver Anexos A.2 para la lista completa de tipos de bloques).

Independientemente de esta distribución, los bloques y el descriptor introducido en la Sección 5.1, constituyen el centro de la tesis que se propone en las siguientes secciones.

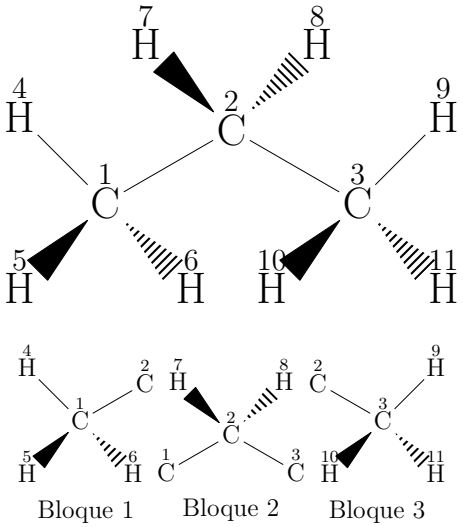

Figura 5.4: Descomposición en bloques de la molécula C_3H_8 existente dentro de la $8CONF$. Es molécula aporta 3 bloques con átomos redundantes en cada uno de ellos. De estos bloques 2 son tratados como iguales por contener los mismos elementos.

5.3 Método y Optimizaciones.

Se propone buscar un algoritmo basado en aprendizaje automático que permita reconstruir el entorno (átomos alrededor de uno central) de un grupo de atómos. El algoritmo tiene que ser simple de calcular, tiene que ofrecer mejor resultado que un método semi-empírico y acercarse lo más posible a la estructura que se obtendría en una optimización costosa usando un método *ab-initio*.

El método de aprendizaje automático, tiene que ser capaz de adaptarse con la menor variación posible a distintos tipos de entornos.

5.3.1 Métodos.

El método de aprendizaje automático más directo de evaluar sería una *SVM* (ver Sección 3.1). La simplicidad se ajusta al problema. Específicamente la Regresión Ridge con Kernel (KRR) definida en la eq. 4.3 (ver Sección 4.2.2). Se empleó una función laplaciana para el kernel definida por:

$$k(d_i, d_j) = e^{-\frac{|d_i - d_j|}{\sigma}} \tag{5.4}$$

De eq. 4.3 y eq. 4.2 se obtienen dos hiperparámetros a optimizar σ que representa el radio que alcanza el la función del kernel y λ como variable de regularización del KRR.

5.3.2 Predicción de las coordenadas de los bloques.

Una primera iteración demostró que la utilización de KRR y e-state para predecir coordenadas cartesianas directamente, no resultaron fructíferas. La precisión arbitraria que se puede establecer utilizando coordenadas cartesianas crea un problema de ajuste para el algoritmo de aprendizaje automático.

Por otra parte existe una correlación entre e-state y la distancia de enlaces (el valor I del e-state está unido a la hibridación del elemento). Producto de la disposición espacial y la variedad de bloques, los ángulos internos resultaron un reto. Los ángulos de torsión no ofrecieron los resultados esperados como para seguir explorando esta linea, principalmente porque la combinación de *algoritmo-descriptor* no era capaz de representar las simetrías presentes. Un ejemplo de esto era C_2H_6 donde el KRR no era capaz de diferenciar los distintos ángulos de torsión del H, produciendo simplemente una media de los posibles ángulos.

Para tratar de reducir algunos de los problemas, primeramente se eliminó la traslación convirtiendo los Bloques con vectores internos centrados en el átomo. Se optó por realizar la transformación de las coordenadas cartesianas R pertenecientes a los bloque abstractos al plano de productos escalares $B = R \cdot R^T$. Con esta transformación se elimina la rotación y todas las representación queda en productos escalares.

Para un bloque, como el mostrado en la Figura 5.4, compuesto por *C-C-H-H-H*. Los vectores internos serían *a, b, c, d* a partir de átomo central *C*, su correspondiente bloque de productos escalares estaría representado por la matriz:

$$\mathbf{B} = \mathbf{R} \cdot \mathbf{R}^T = \begin{bmatrix} a^2 & \mathbf{a} \cdot \mathbf{b} & \mathbf{a} \cdot \mathbf{c} & \mathbf{a} \cdot \mathbf{d} \\ \mathbf{b} \cdot \mathbf{a} & b^2 & \mathbf{b} \cdot \mathbf{c} & \mathbf{b} \cdot \mathbf{d} \\ \mathbf{c} \cdot \mathbf{a} & \mathbf{c} \cdot \mathbf{b} & c^2 & \mathbf{c} \cdot \mathbf{d} \\ \mathbf{d} \cdot \mathbf{a} & \mathbf{d} \cdot \mathbf{b} & \mathbf{d} \cdot \mathbf{c} & d^2 \end{bmatrix} \tag{5.5}$$

Directamente se obtienen las distancias de enlaces de la diagonal y el resto de los productos escalares definen los ángulos internos. El bloque queda sobredefinido debido a la existencia de coordenadas redundantes. Esta matriz tiene la propiedad de ser *definida positiva*, diagonalizable y en la mayoría de casos los valores propios son todos positivos.

La aplicación el método de aprendizaje KRR sobre esta matriz se puede hacer de dos formas: (1) construir un tensor con la matriz diagonal superior y entrenar el método para que prediga este tensor, (2) construir un modelo de KRR para cada elemento de la matriz diagonal superior y predecir sólo un número en vez de un tensor. Después de breves experimentos, los modelos de KRR eran mas precisos prediciendo un sólo valor en vez de un tensor. La explicación más probables se deba a que el método KRR trata de balancear los valores del tensor de salida,

creando un ruido que afecta el resultado numérico. Por esta razón se decidió por la opción de un modelo por cada producto escalar. De esta forma cada nueva predicción de bloque genera M modelos donde M es la cantidad de productos escalares contenidos en matriz diagonal superior.

Los descriptores que participan en cada modelo son solo aquellos que se ven involucrados en la construcción de los vectores. De esta forma otras especies dentro del mismo bloque no influyen de forma directa en la predicción. Pero sus contribuciones están presente como parte del valor ΔI del e-state.

Para obtener nuevamente las coordenadas cartesianas se utilizó la descomposición en valores propios (*Eigen-decomposition* o *Schur Decomposition*) a través de la expresión:

$$\mathbf{B} = \mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^T = (\mathbf{Q}\mathbf{\Lambda}^{1/2})((\mathbf{\Lambda}^{1/2})^T\mathbf{Q}^T) = (\mathbf{Q}\mathbf{\Lambda}^{1/2})(\mathbf{Q}\mathbf{\Lambda}^{1/2})^T = \bar{\mathbf{R}}'(\bar{\mathbf{R}}')^T. \quad (5.6)$$

donde $(*)^T$ representa la operación de transpuesta y $\bar{\mathbf{R}}'$ son las coordenadas re-construidas. Esta operación puede dar como resultado valores propios negativos lo cual implica que varias especies dentro del bloque cambiaron de posición. Este fenómeno es muy esporádico dentro de los bloques analizados como para ser relevante.

5.3.3 Pre-Procesamiento de Datos

Del análisis de la distribución de los bloques salen a relucir distintos valores para las magnitudes distancia de enlaces, ángulos y ángulos de torsión. Esta variación de valores está sujeta a dos factores fundamentales: (1) tipo de enlace entre las especies, afecta la distancia entre los dobles y triples enlaces que son más cortos que los simples, (2) ubicación dentro de la molécula, anillos afectan los ángulos entre especies y los ángulos de torsión.

Este tipo de variaciones tiene que ser tomada en cuenta en el proceso de entrenamiento, ya que bloques compuestos por las mismas especies, puede contener información considerada como *ruido* por provenir de moléculas con distintas conformaciones. Para filtrar este ruido en el grupo de entrenamiento se tiene en cuenta varios factores: (1) si la molécula a predecir contiene anillos, (2) cuantos átomos dentro de ese anillo, (3) especies químicas presentes. Después de separada en bloques se tiene en cuenta:(1) especies pertenecientes al bloque y (2) hibridación del átomo central. Estas 2 últimas restricciones garantiza que se utilicen bloques similares en el proceso de aprendizaje sin propiciar un sobre-ajuste del método. Todos estos datos son fácilmente obtenibles partiendo de una cadena SMILES y utilizando RDKit.

Este proceso es similar al descrito en la Sección 3.1.2, con la variación de la no aplicación de una norma y el umbral de inclusión es que ambos bloques contengan

las mismas especies con la misma hibridación. De aquí pueden extraerse 81 grupos de bloques muchos más que los 67 existentes anteriormente.

5.3.4 Algoritmo de Aprendizaje y predicción de bloques.

En las secciones anteriores se fueron exponiendo cada uno de los elementos que componen la *Predicción de bloques*. La relación entre ellos se puede expresar en el siguiente algoritmo:

Algorithm 5.1: Algoritmo de Aprendizaje y Predicción de Bloques. PE son productos escalares, R coordenadas cartesianas, $(*)^T$ es la matriz transpuesta

Require: $\mathbb{T}_{Mol}$ y $\mathbb{C}_{Mol}$
1: **for all** mol en $\mathbb{C}_{Mol}$ **do**
2: **for all** Bloque en mol **do**
3: Obtener todos los Bloques de entrenamiento de $\mathbb{T}$
4: Calcular productos scalares $\mathbb{T}_{PE} = \mathbb{T}_{BloquesR} \cdot \mathbb{T}^T_{BloquesR}$
5: **for all** pe en $\mathbb{T}_{PE}$ **do**
6: $PE' = k(d')^T(\mathbf{K'} + \lambda\mathbf{I})^{-1}{\mathbb{T}_{refe}}'$
7: **end for**
8: Reconstruir R' usando PE' y 5.6
9: **end for**
10: **end for**

donde $\mathbb{T}$ representa las moléculas desde las cuales se extraerán el grupo de bloques para el aprendizaje y $\mathbb{C}$ son las moléculas a predecir. El algoritmo analiza la molécula a predecir y selecciona de la mase de datos aquellas que le son mas afines en cuanto a cantidad de anillos, átomos contenidos en esos anillos y especies en la molécula

La implementación de este algoritmo está contenida dentro del código descrito en la Sección 4.2.

5.3.5 Optimización de Hiper-Parametros

Al ser el método de aprendizaje una KRR, aplicar una optimización Bayesiana carece de sentido (ver Sección 3.3). El modelo sustituto a utilizar sería uno gaussiano por lo que la evaluación del modelo sustituto y el modelo a optimizar serían computacionalmente similares.

Se optó por una búsqueda en una maya de posible valores determinados empíricamente. Estos valores se fueron acotando a través de prueba y error y buscar una mínimo

global en un rango prudente. Esto evitó la necesidad de realizar muchas iteraciones del método.

Para seleccionar las propiedades objetivo habían dos opciones. La primera eran las coordenadas internas (distancias de enlaces, ángulos y ángulos de torsión) por cada grupo de bloques. Cada una presenta sus propias distribuciones independientes y el modelo tiene que adaptarse específicamente a cada distribución. En otras palabras se requería optimizar un modelo para cada propiedad por grupo de bloque. Esto requirió mucho esfuerzo y las primeras pruebas mostraros que el descriptor seleccionado no representaba correctamente todas las propiedades, sobre todo los ángulos internos y de torsión.

Con la transformación a productos escalares de las coordenadas cartesianas, se eliminaron las coordenadas internas, por tanto solo se requería la optimización solo por grupo de bloque. Para realizar la optimización se seleccionaron aquellos bloques para los que existían suficientes datos para el aprendizaje (al menos 10 bloques). Los valores de los extremos de la maya de búsqueda fueron ajustados (ver Tabla 5.7) hasta encontrar un rango donde era posible la existencia de mínimos para todos los bloques.

$$\begin{aligned} \sigma &= [10^{-3}, 0.1, 1, 10, 15, 100, 1000, 3000] \\ \lambda &= [10^{-3}, 0.1, 0.5, 0.9, 1] \end{aligned} \tag{5.7}$$

En la Figura 5.5 se muestra un resumen de las optimizaciones realizadas. Puede notarse como partiendo de la combinación $\sigma = 1$ y $\lambda = 0.9$ empieza ha existir un decrecimiento en el RMSD para todos los bloques.

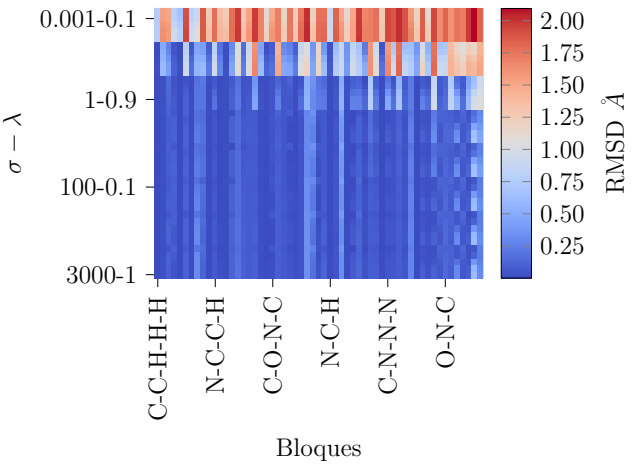

Figura 5.5: Resumen de las optimizaciones de σ y λ, hiperparámetros del la Regresión Ridge con Kernel. La escala de colores representa el RMSD. Se puede notar como todos los bloques mejoran progresivamente en los errores. En algunos casos se pueden notar oscilaciones en la coloración.

Cada bloque va a tener su mínimo en combinaciones diferente de los hiperparámetros pero puede empezar a notarse una tendencia general. En algunos casos se pudieron notar oscilaciones en los resultados del RMSD, esto se debe a que los valores de hiperparámetros se compensan en sus variaciones y existe mas de una combinación que ofrece buenos resultados.

Los valores encontrados pueden utilizarse de dos formas diferentes, (1) particularizando cada modelo de KRR con los valores obtenidos, conllevaría a N modelos diferentes para N grupos de bloques diferentes y/o (2) realizando un promedio de los hiperparámetros y promover un modelo único.

Este trabajo perseguirá la segunda linea de de investigación. Inclusive si la utilización de parámetros específicos para cada bloque puede mejorar las estimaciones, la perspectiva de presentar un modelo único, representa una solución más elegante sin afectar la precisión notablemente. Los valores seleccionados fueron $\sigma = 10$ y $\lambda = 0.9$. Los experimentos descritos en las siguientes secciones utilizan estos valores.

5.3.6 Curvas de Aprendizaje.

Podría suponerse que sustituir simplemente las coordenadas de cada nuevo bloque por una media entre todos los bloques similares llevaría a una solución de similar precisión. Las curvas de aprendizaje demuestran la capacidad del método de mejorar con el aumento de datos de aprendizaje e ir mas allá de solo la media.

Para las pruebas realizadas se tomaron 1k moléculas de forma aleatoria, las cuales fueron separadas en bloques. Este conjunto se utilizó como grupo de control $\mathbb{C}$. Para cada iteración del experimento se muestra un promedio del RMSD obtenido después de reconstruida las coordenadas a partir de la predicción de productos escalares. Para el conjunto de aprendizaje $\mathbb{T}$ se seleccionaron 10k la cuales fueron separadas en bloques.

Debido a la desigual distribución de bloques las pruebas fueron realizadas utilizando porcentajes de la base de aprendizaje disponible. La selección de que subconjunto del $\mathbb{T}$ participará en el aprendizaje fue aleatoria. En la Figura 5.6 se muestra un decremento en el RMSD en la medida que la cantidad de bloques en el conjunto de aprendizaje aumenta. Este comportamiento denota la capacidad del método de mejorar progresivamente.

Para muchos bloques existe un punto de *no mejora*. En esencia agregar más datos no cambia significativamente los valores de RMSD. Esto implica como factor positivo que se puede lograr una buena estimación con un mínimo indispensable, acortando en gran medida operaciones costosas como la diagonalización del kernel $k(*)$. Por otro lado pone una cota a cuanto puede aportar en términos de precisión el método, el cual no mejora indefinidamente. Se ha apreciado este comportamiento en bloque muy bien definido estructuralmente como $C-C-H-H-H$

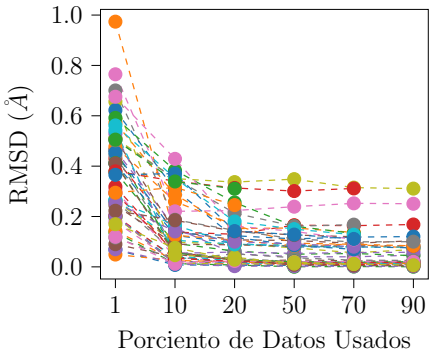

Figura 5.6: Curvas de aprendizaje para algunos de los grupos de bloques existentes dentro de la 8CONF. Nótese como con cada aumento de la base de aprendizaje mejora la precisión.

con todos los enlaces simples. La representación de productos escalares presenta una desviación estándar muy pequeña, esto unido al descriptor que capta muy bien las relaciones entre las especies son necesarios muy pocos bloque para que el RMSD llegue a 0.001812 Å.

5.4 Resultados.

Las prueba realizadas se hicieron sobre la base de datos de grupos de bloques extraídas de la 8CONF. Sin embargo siendo el objetivo la reconstrucción de conformaciones moleculares se seleccionaron 450 moléculas para conformar el grupo de control $\mathbb{C}$. Las restantes fueron utilizadas para el grupo de entrenamiento $\mathbb{T}$. Ambos fueron seleccionadas de forma aleatoria. Los hiperpámetros utilizados fueron los encontrados en las optimizaciones realizadas.

Parte fundamental de los resultados resulta la utilización del método ETKDG. Debido a esto los primeros experimentos se basaron en comprobar como se comportaba la generación de conformaciones basado en este método.

5.4.1 Comparación ETKDG de base *ab-initio* contra ETKDG base experimental.

La modificación de del código fuente de RDKit se basó en el ajuste las curvas a los histogramas de ocurrencias de ángulos de torsión [86, 93]. La expresión a parametrizar está definida como:

$$V = \sum_{i=1}^{6} V_i * (1 + s_i * \cos i * x) \tag{5.8}$$

donde x son los valores de los ángulos en el histograma (en radianes) y s_i, V_i los parámetros de la curva a ajustar.

Los parámetros s_i y V_i obtenidos del ajuste son los utilizados por el método DG (del inglés Distance Geometry) para establecer restricciones sobre la matriz de distancia creada de forma aleatoria. El método DG obtiene las coordenadas cartesianas utilizado los valores y vectores propios de esta matriz [94, 102].

Usando como base de referencia las geometrías optimizadas con DFT/B3LYP , se realizó una comparación entre el método ETKDG de base experimental[3] y el modificado (en lo adelante *TTKDG* Theoretical-Torsion Knowledge Distance Geometry) [86, 93]. La métrica utilizada para evaluar los errores fue el RMSD (*Root Mean Squared Distance*) de las matrices de distancias de cada molécula definida por:

$$RMSD = \sqrt{\frac{1}{N}\sum_{i=0}^{N}(r_i - r_i^{ref})^2} \tag{5.9}$$

donde N es la cantidad de átomos y r_i^ref representa la distancia de referencia, obtenida en este caso de la geometría optimizada con DFT/B3LYP de la molécula correspondiente en la 8CONF.

La moléculas de la 8CONF fueron transformadas a cadenas *SMILES* y reconstruidas utilizando las mismas condiciones y las funcionalidades de RDKit. Los resultados se pueden observar en la Figura 5.7 donde se muestra un histograma del RMSD entre TTKDG y el ETKDG.

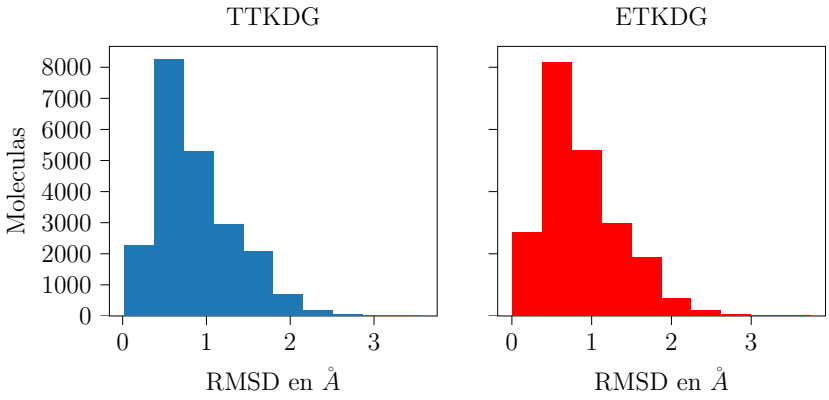

Figura 5.7: Comparación de las geometrías obtenidas a partir del método TTKDG (A) y el ETKDG (B) utilizando como referencias las coordenadas optimizadas con DFT/B3LYP. Las diferencias son imperceptibles.

[3]viene en el código original descargado de la fuente https://github.com/rdkit/rdkit

Las diferencias son pocas. Las implicaciones son alentadoras, los ángulos de torsión utilizados por el método ETKDG produce conformaciones similares a los que se producirían si se utilizaran ángulos de torsión extraídos de la base de datos 8CONF.

5.4.2 Predicción de las coordenadas cartesianas por bloques.

Utilizando los hiperparámetros encontrados en los experimentos de optimizaciones (ver Sección 5.3.5), se realizó un experimento re-construyendo las coordenadas bloques a partir de los productos escalares utilizando el Algoritmo 5.1. Como métrica se utilizó el RMSD definida en eq. 5.9 a partir de las matrices de distancias. Como referencias se utilizaron las coordenadas producidas por DFT/B3LYP.

La 450 moléculas que conforman el $\mathbb{C}$ se dividieron en bloques. Sobre cada bloque se realizaron se realizaron 10 iteraciones de predicción. En cada iteración se seleccionó de forma aleatoria un subconjunto constituidos por aproximadamente el 80% del conjunto de bloques de aprendizaje extraídos de las moléculas contenidas en el grupo de entrenamiento $\mathbb{T}$. El grupo de bloques de control $\mathbb{C}$ permaneció invariable[103].

Los resultados mostrado en la Figura 5.8 son una media del RMSD de los bloques en el grupo de control (detalles en Anexo A.3). En el histograma mostrado en la Figura 5.8 también puede notarse que existe un numeroso grupo de bloques cuyo RMSD medio no supera 0.05 Å, además de un grupo de bloques con un error por encima de 0.3 Å. Algo que comparado con el primer grupo cuyo RMSD medio no supera 0.05 Å está en otra escala diferente. Estos *bloques problemáticos* pertenecen al grupo de los poco abundantes o que están presentes en anillos muy tensionados los cuales hace variar las distancias de enlaces y los ángulos internos.

Diseccionando el histograma de la Figura 5.8 se pueden encontrar más detalles acerca de la distribución de los errores. La Figura 5.9 se muestra el RMSD en función del grupo de bloque. En algunos casos se puede observar que los valores de la desviación estándar del error (representada con una linea vertical sobre los puntos) dentro de las 10 iteraciones es prácticamente 0. Esto puede interpretarse como que la desviación del error en las predicciones para algunos bloques es muy baja, y los resultados obtenidos son consistentes para distintas base de aprendizaje.

5.4.3 Reconstrucción de moléculas.

La simpleza del método resalta la necesidad de seguir al siguiente paso, la reconstrucción de compuestos. Para esto se siguieron dos caminos: (1) emplear el método TTKDG unido a la predicción de bloques y (2) utilizar un método propio de ensamblado.

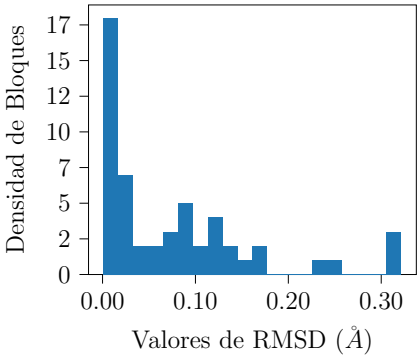

Figura 5.8: Histograma de la cantidad de bloques por valores de RMSD. La alta densidad existente por debajo de 0.05 Å, resulta alentador. Sin embargo se aprecian bloques con hasta 0.3 Å en el RMSD.

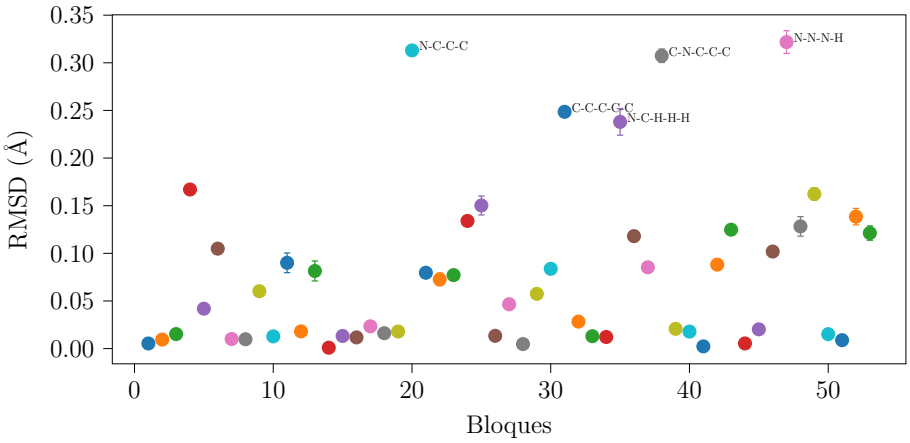

Figura 5.9: Detalles del RMSD por grupo de bloques. Pueden apreciarse los bloques problemáticos.

Debido a los buenos resultados obtenidos con el método TTKDG modificado del original de RDKit en la reconstrucción de conformaciones se decidió utilizar para encontrar los ángulos de torsión requeridos para ensamblar bloques.

Los pasos seguidos para su incorporación dentro del algoritmo están descrito en el Algoritmo 5.2:

Indicándole al método TTKDG que matriz de distancias usar se evita que esta se genera aleatoriamente. La matriz de pesos W_D se asegura que solo las distancias que implican ángulos de torsión sean afectadas.

La utilización de un método propio de ensamblado conlleva varias dificultades:

1. La unión entre dos bloques. Como unir bloques contiguos sin afectar las coordenadas estimadas de ninguno de los dos bloques participantes.

Algorithm 5.2: Algoritmo de Aprendizaje, Predicción de Bloques y Ensamblado de Moléculas.

Require: $\mathbb{T}_{Mol}$ y $\mathbb{C}_{Mol}$

1: Llamar al Algoritmo 5.1, Entrada $\mathbb{T}_{Mol}$ y $\mathbb{C}_{Mol}$
2: **for all** mol en $\mathbb{C}_{Mol}$ **do**
3: Inicializar la matriz de distancia M_D de la molécula con los bloques obtenidos usando el Algoritmo 5.1
4: Construir la matriz de *pesos* W_D para fijar las distancia obtenidas usando el Algoritmo 5.1
5: Llamar TTKDG usando las matrices de distancia M_D y la de pesos W_D
6: **end for**

2. La quiralidad y los estereo-centros.

3. La selección/estimación del ángulo de torsión entre bloques.

La unión entre dos bloques contiguos se puede resolver utilizando los índices[4] asignados a la cadena de átomos. Manteniendo estos índices constantes durante todo el proceso de predicción se pueden identificar los átomos redundante presente en ambos bloques.

Los átomos redundantes son el punto de unión entre los bloques. Se puede construir un vector de traslación para desplazar el átomo central a la posición de su átomo redundante en bloque anterior siguiendo la numeración asignada. Existe, sin embargo, una pequeña discrepancia en el ángulo de unión ϕ que se forma entre los átomos redundantes, debido a la reconstrucción de las coordenadas a partir de los valores propios. Este problema se resuelve aplicando una rotación que reduzca a 0 esta desviación. Los pasos descritos se puede observar en la Figura 5.10.

Tratar de encontrar quiralidad y/o estereo-centros durante el proceso de *ensamblaje* resulta complicado. Se optó por considerar cada unión entre bloques como un un posible estero-centro, se aplicó una transformación antes de ser unido al bloque que le antecede. Esto produce una gran variedad de posibles conformaciones, 2^N, donde N es el número de bloques que posee la molécula.

La selección del ángulo de torsión entre bloques resulta el problema más complicado de resolver. Similar a los estero-centros es necesario conocimiento previo de la estructura molecular. Este método es el utilizado por RDKit para obtener las posibles conformaciones sumado a las reglas de preferencias de ángulos de torsión [86, 93].

[4]La *IUPAC* (International Union of Pure and Applied Chemistry) propone un conjunto de reglas estándar para indexar cadenas de carbono.

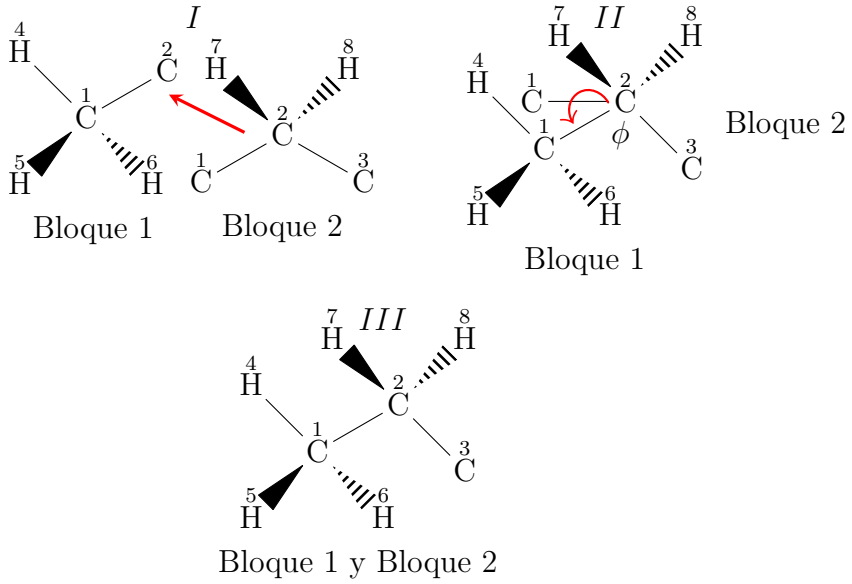

Figura 5.10: Ejemplo de unión entre 2 bloques por etapas. (1) Traslación, (2) rotación y la eliminación de elementos redundantes. Al final de este proceso se requiere ajustar los ángulos de torsión entre bloques.

Pare enfrentar estos problemas fueron diseñadas dos estrategias: (1) implica utilizar el ángulo de torsión correcto a partir del conocimiento que se posee de la molécula, opción poco deseable pero al menos garantizaría conocer si se puede reconstruir la molécula original a partir de la estimación de bloques (en adelante *Tors-Ori*); (2) utilizar un ángulo de torsión similar obtenido de la base de entrenamiento $\mathbb{T}$, seleccionando uno que contenga las mismas especies (en adelante *Tors-Apred*), en caso de existir mas de uno se seleccionó aquel que representara la cadena de carbono mas grande.

Estos mecanismos unido al uso del método TTKDG (en adelante TTKDG-ML) constituyen los métodos seleccionados para el ensamblado de moléculas.

Para los experimentos se utilizaron 450 moléculas como $\mathbb{C}$ seleccionadas aleatoriamente y 15k moléculas como $\mathbb{T}$ para obtener los datos estructurales necesarios, bloques y ángulos de torsión. Las métricas utilizadas fueron el RMSD para errores entre bloques y conformaciones y se utilizó *Effective Medium Theory*[104] (EMT implementado en ASE) para realizar cálculos de energía potencial sobre las conformaciones. EMT está descrito para sólidos no para moléculas, por tanto los valores de energía serán solo una referencia. Se tomó esta decisión porque se necesitaba tener una idea de la energía potencial sin necesidad de realizar ningún cálculo que requiriera mucho tiempo como los *ab-initio*. Sin embargo las decisión de seleccionar la mejor conformación descansó en el RMSD.

Se puede ver en la Figura 5.11 un histograma de los errores en las mejores conformaciones para moléculas con 0 átomos en anillos y con 7 átomos en un anillos (los resultados completos de las 450 moléculas se puede encontrar en Anexos A.4). Los métodos *TTKDG-ML* y la utilización de los ángulos de torsión de la molécula original *Tors-Ori* mostraron los mejores resultados donde las mayores densidades de errores se encuentran en 0.5Å y $>$ 0.5Å. Resulta comprensible el bajo error que el método *Tors-Ori* presenta, conocer el ángulo de torsión original ayuda en el ensamblaje de bloques. El método *TTKDG-ML* produce buenos resultados, manteniendo constantes las coordenadas obtenidas con la predicción de bloques. Por otra parte *Tors-Apred* produce errores de hasta 2 Å y valores de energía por encima de los 50 eV.

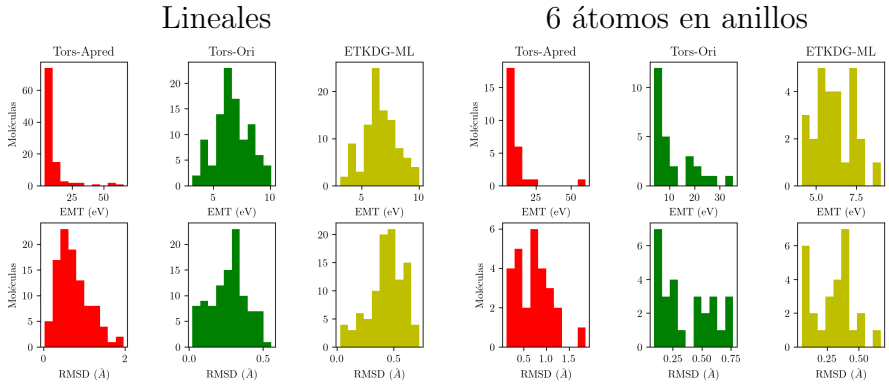

Figura 5.11: Algunos ejemplos de valores de EMT y RMSD para las moléculas reconstruidas. Se muestran aquellas consideradas *lineales* con 0 átomos en anillos y las que presentan 6 átomos en anillos.

La comparación visual permite tener una mejor perspectiva de la generación de conformaciones. En la Figura 5.12 muestra alguna de las reconstrucciones logradas con cada uno de los métodos.

El método de ensamblado *Tors-Ori* y *Tors-Apred* aportó algunos datos adicionales de interés en cuanto a la generación de conformaciones. Poniendo el RMSD calculado en función de los posibles conformaciones se encontraron múltiples valles donde existen posibles mínimos. En la Figura 5.13 se muestran este comportamiento para la molécula $C_6O_2H_{12}$. En el caso de *TTKDG-ML* esto no sucedió, aunque es posible pedirle a RDKit que genere distintas conformaciones, las restricciones impuestas por el método descrito solo permiten una sola conformación.

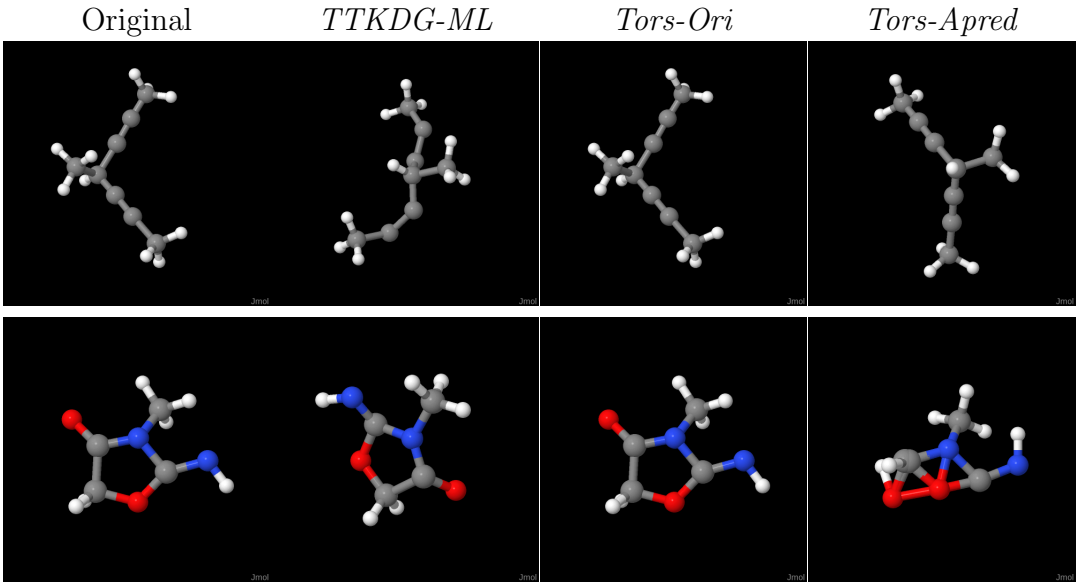

Figura 5.12: Algunos ejemplos de valores de conformaciones producidas con los métodos empleados.

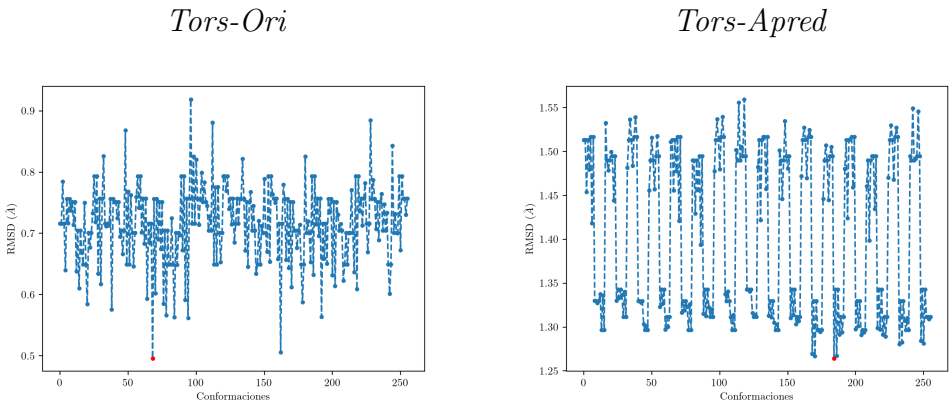

Figura 5.13: RMSD contra las conformaciones moleculares obtenidas utilizando los métodos *Tors-Ori* y *Tors-Apred*. El mínimo seleccionando está indicado en rojo.

5.5 Conclusiones.

En este capítulo se propuso un método de reconstrucción de moléculas basados en el uso del modelo de aprendizaje automático KRR. El algoritmo de predicción basados en uso de bloques demostró ser preciso si existen suficientes datos para sustentarlos.

Las conclusiones principales se listan a continuación:

- El descriptor propuesto *e-state*, es capaz de predecir con alto grado de precisión la estructura de un bloque. No sólo las distancias de enlaces si no también los ángulos internos.
- El método utilizado para el aprendizaje KRR es simple de implementar, fácil de entrenar y el modelo ya entrenado puede ser reutilizado continuamente para producir nuevos bloques. Además la producción de bloque utilizando KRR mejora los resultados sobre simplemente aplicar la media
- La propuesta de predicción de bloques en su conjunto puede ser enriquecida agregando mas moléculas que diversifiquen la base de datos de bloques.
- Los problemas de precisión numérica en las predicciones se deben principalmente a la cantidad de bloques por grupos de bloques presentes en la 8CONF.
- El método TTKDG-ML propuesto abarca completamente el proceso de obtención de conformaciones moleculares con precisión *ab-initio*.

Se identificaron un grupo de problemas relacionados principalmente con la capacidad representativa del descriptor seleccionado y el método para la predicción/reconstrucción de los ángulos de torsión. En este aspecto se llegó a la conclusión que se necesita mas información para poder tener una buena correlación con los ángulos de torsión, el punto más débil del algoritmo. La propuesta de utilizar el mecanismo TTKDG-ML, modificado del original en RDKit (ver Sección 5.4.1), junto a los bloques predicho ofreció los mejores resultados (ver Figura 5.11).

Capítulo 6

Predicción del espectro de absorción de moléculas.

Contents

En este Capítulo se abordará el uso de aprendizaje automático para la estimación de propiedades electrónicas de compuestos químicos publicado en [105].

El enfoque principal será sobre el espectro de absorción, por representar la interacción de los materiales con la luz. Una propiedad de relevante importancia para la industria de sensores, fotoeléctrica, etc..

La respuesta de un material a la interacción con la radiación electromagnética (luz) se puede estudiar de manera teórica utilizando los métodos de la mecánica cuántica como TD-DFT (Ver Capítulo 2). Dependiendo del funcional de la energía de intercambio y correlación E_{xc} utilizado, el cálculo es mas o menos costoso computacionalemente. La llamada *Escalera de Jacob* (Ver Sección 2.1.1) resulta en un obstáculo a franquear, saltar peldaños al menor precio computacional posible sigue siendo una pregunta abierta.

En este Capítulo se detalla primeramente los *Estudios Previos*, centrándonos los descriptores moleculares utilizados en este y otros trabajos. Seguidamente se analizará la distribución de los descriptores seleccionados en la base de aprendizaje. Como métodos de aprendizaje automático se seleccionó el KRR y las Redes Neuronales. En ambos caso se describirán las optimizaciones realizadas así como los resultados alcanzados. Se usará un k para denotar *mil* (ej. el número 30000 aparecerá como 30k). Salvo que se especifique otra magnitud, todas las energías están dadas en *electron-Volts* (eV) y las distancias en Angtrom (Å).

6.1 Estudios Previos.

Abordar el tema de la estimación del espectro de absorción desde el aprendizaje automático no carece de precedentes [4, 13, 9, 106]. Las Redes Neuronales han sido utilizadas con éxito para predecir propiedades electrónicas como la energías de excitación, momentos dipolos y cargas parciales con precisión parecida a TD-DFT empleando como descriptores múltiples conformaciones de un mismo compuesto [13]. También se han realizado análisis comparativos de la eficiencia con varias topologías diferentes de Redes Neuronales para producir la densidad de estados, empleando como descriptor la matriz de Coulomb (en la Sección 6.2.2 se indagará en esta representación) [9]. Por último la utilización de modelos de Regresión Ridge con Kernel, usualmente empleando un kernel Gaussiano, ha tenido su aplicación para obtener propiedades electrónicas relacionadas con el espectro de absorción como la energía del primer estado excitado [4].

Estos tres trabajos resumen los que para el Autor son las líneas de investigación en este campo. La utilización de descriptores estructurales que incluye información atómica (Coulomb Matrix) [9, 4]. Dos métodos de aprendizaje Regresión Ridge con Kernel y las Redes Neuronales, esta última con topologías complejas en lo que se conoce como *Aprendizaje Profundo*[1][9]. Dentro de las propiedades a predecir

[1]del inglés Deep Learning, Redes Neuronales donde se involucran MLP y CNN en una sola red con un número alto de capas ocultas. Resultan complicadas de implementar y entrenar.

están aquellas obtenidas de los cálculos TD-DFT, energías de los estados excitados y fuerzas del oscilador. También existe un esfuerzo en mejorar la precisión de de los cálculos a través de uso de modelos de ML, como el denominado ΔML para estimar las diferencias de energías entre los peldaños de la *Escalera Jacob* [4].

La combinación de simulaciones computacionales de bajo costo con otras más pesadas, utilizando un proceso de regresión gaussiano con el objetivo de acelerar cálculos mas costosos, fue propuesto por primera vez por M. C. Kennedy y A. H. Hagan donde se pretende mejorar un cálculo de bajo costo de simulación de producción en un pozo petrolero utilizando un método de aprendizaje automático [107] .

6.2 Análisis de la Base de Datos utilizada.

La base de moléculas utilizada está constituida por 21k compuesto orgánicos de hasta 8 átomos de Carbono (C), Oxígeno (O), Nitrógeno (N) y Flúor (F) (8CONF en lo adelante). La 8CONF tiene sus geometrías optimizadas. utilizando DFT y el funcional B3LYP con funciones de base 6-31(2df, p) [4, 99, 100].

Sobre la 8CONF se realizaron 3 grupos de cálculos TD-DFT utilizando *Octopus* (ver Sección 4.1). En el primer grupo simplemente se obtuvieron los orbitales sin optimizar que crea Octopus usando *LCAO* (del inglés Lineal Combination of Atomic Orbitals), esto es considerado un cálculo de muy bajo costo y muy impreciso (en adelante *OCT-LCAO*). En el segundo grupo se obtuvo el *estado fundamental* utilizando el funcional *LDA* como aproximación a la energía de intercambio y correlación (en adelante GS-LDA).

En el tercer y último grupo se obtuvieron el *estado fundamental* y estados excitados Casida (ver Sección 2.2) utilizando *PBE0* como aproximación híbrida a la energía de intercambio y correlación, considerado un cálculo costoso mucho mas preciso (en adelante *CASIDA-PBE0*) [2].

A cada cálculo realizado se le agregó una métrica para determinar el carácter de *transferencia de carga* de la transición. En los cálculos de GS se utilizaron las mono-excitaciones para obtener esta métrica. Esta métrica denotada por Δr se define como [23]:

$$\Delta r = \frac{\sum_{i,a} K_{ia}^2 R_{ia}}{\sum_{i,a} K_{ia}^2} \tag{6.1}$$

donde $R_{ai} = |\langle \phi_a(r)|r|\phi_a(r)\rangle - \langle \phi_i(r)|r|\phi_i(r)\rangle|$ donde ϕ_a y ϕ_i son las funciones

[2]Remitimos al lector a Anexo A.5 para mas detalles sobre los parámetros utilizados en *Octopus.*

de onda monoelectrónicas virtuales y ocupados respectivamente, y K son lo coeficientes de las transiciones obtenidos al resolver la ecuación de Casida.

En el trabajo de Ciro A. Guido y otros [103], se propone la siguiente clasificación: para los rangos $\Delta r \leqslant 1.5$ Å se considera una transferencia de carga de rango corto, si $\Delta r \geqslant 2.0$ Å se considera de largo rango.

La distribución de estas propiedades en la 8CONF se analizarán en las siguientes secciones.

6.2.1 Propiedades a Predecir.

La obtención de propiedades espectroscópicas haciendo uso de ML ha sido objetivo de varios trabajos (ver Sección 6.1). La construcción del espectro de absorción implica la solución de la *ecuación de Casida* (ver eq. 2.27, en la Sección 2.2) para la obtención de los *energías de excitación* y las *fuerzas del oscilador*, siendo uno de los cálculos más costosos en TD-DFT. La precisión en los valores de estas propiedades está unida al funcional de la energía de intercambio y correlación que se use.

Estas propiedades fueron obtenidas a partir de los cálculos realizados con *Octopus*. Los valores de las energías de excitación obtenidos fueron truncados alrededor de los 10 eV resultando en aproximadamente los 10 primeros estados excitados. Por encima de este valor de energía aparece la energía de ionización en la base de datos 8CONF por lo que carece de sentido explorar el espectro UV-Visible. El número de fuerzas del oscilador fueron limitadas igualmente por la cantidad de estados excitados.

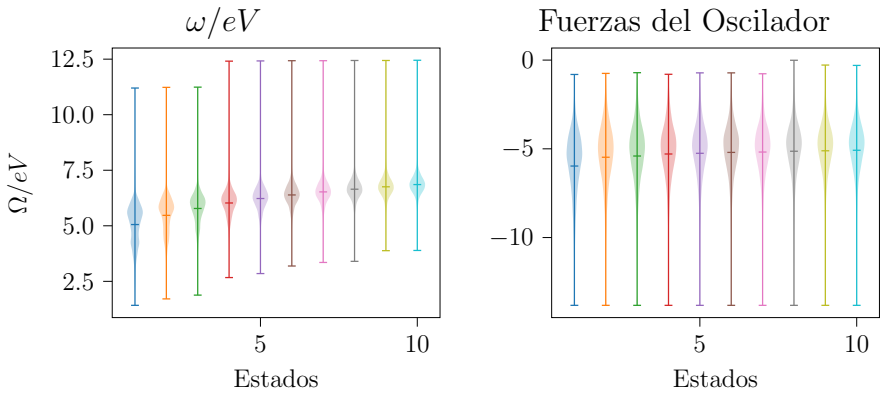

Figura 6.1: Distribución de las propiedades obtenidas a través de la solución de la ecuación de Casida utilizando el funcional PBE0. Se muestran tanto los valores extremos como la media y en que rango existe la mayor densidad de valores. Las Fuerzas del Oscilador se le aplicó la función *log* para mejorar la visibilidad en la distribución.

La Figura 6.1 muestra 15k moléculas de la las 21k para las que se realizaron los cálculos utilizando PBE0 como funcional. Puede notarse que las fuerzas del oscilador poseen un distribución muy compacta y los picos de donde se concentran la mayor cantidad de valores están alrededor del 0 con muy poca variación. Se puede notar valores extremos en otro orden de magnitud completamente diferente a la media.

Los momentos de transición μ representan un paso intermedio antes de obtener las fuerzas del oscilador en la solución de las ecuación de Casida. La relación entre las fuerzas del oscilador y los momentos de transición está definida por $f_i = \frac{2}{3}\Omega_i/\hbar^2 \sum {\mu_i}^2$. Existe precedente en la literatura consultada donde se pretende predecir esta propiedad en vez de las fuerzas del oscilador [10].

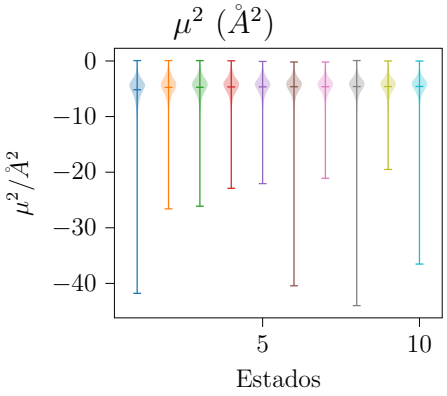

Figura 6.2: Distribución de los momentos de transición (μ^2) para el funcional PBE0. Se muestran tanto los valores extremos como la media y en que rango existe la mayor densidad de valores. A los momentos de transición se les aplicó la función *log* para mejorar la visibilidad en la distribución.

La Figura 6.2 muestra los momentos de transición de 15k moléculas correspondientes al cálculo CASIDA-PBE0. Se puede ver que la mayor densidad está se concentra en un rango similar para todos los estados, con algunos valores atípicos. Los valores medios cerca de 0 indican que la dificultad de encontrar una buena correlación todavía persiste.

El índice de transferencia de carga descrito por la eq. 6.1, mantiene dentro de la 8CONF una distribución bastante homogénea. Los valores de media se ubican sobre 1 Å. La Figura 6.3 muestra la distribución para los primeros 10 estados excitados. Se puede observar como existen valores mayores a 6 Å, considerados atípicos, algo que pude generar problemas en el entrenamiento.

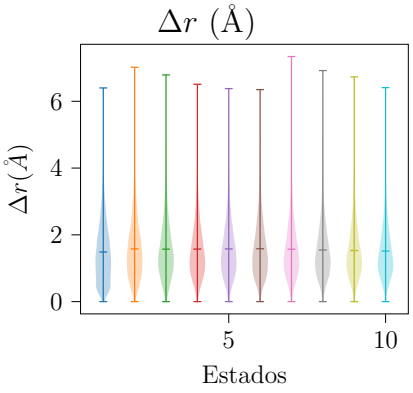

Figura 6.3: Distribución del índice de transferencia de cargas para el funcional PBE0. Se muestran los máximos, mínimos y la media de la distribución.

6.2.2 Propiedades Descriptoras.

La bibliografía consultada apunta hacia el uso de la llamada *Coulomb Matrix* (CM en adelante) como la primera opción a considerar [2]. Definida como :

$$\mathbf{CM} = \begin{cases} 0.5 * Z_i^{2.4} & \text{if } i = j \\ \frac{Z_i Z_j}{R_{ij}} & \text{if } i \neq j \end{cases} \tag{6.2}$$

donde R_{ij} es la distancia inter-atómica entre los átomos i y j, y Z es el número atómico. La simplicidad de esta representación la ha hecho ganar relevancia entre los trabajo donde se aplica el aprendizaje automático en Química/Física Cuántica [4, 2, 9].

Recientemente varias versiones de esta formulación se han desarrollado debido a que la formulación original no es invariante a la permutación de los indices atómicos [29, 108]. Resulta conocido su efectividad al estimar con precisión algunas propiedades electrónicas incluidas las energía de excitación del primer estado. Sin embargo no carece de limitaciones, pues no presenta correlación con propiedades relevantes para el espectro de absorción, como el caso de las fuerzas del oscilador [4].

Para solventar esta dificultad resulta pertinente utilizar información que permita correlacionar mejor las propiedades electrónicas. Se podría considerar la utilización de cálculos DFT de baja precisión. Un ejemplo sería las diferencias de energías entre orbitales ocupados y no ocupados de Kohn - Sham $\Delta E_{a,i} = \epsilon_a - \epsilon_i$, siendo la primera de estas energías la diferencia *HOMO-LUMO* (del inglés High Occupied Molecular Orbital donde $i = (0, {}^{N}/_{2}]$ y Lower Unoccupied Molecular Orbital, donde $a = [{}^{N}/_{2} + 1, \text{inf})$ y siendo N la cantidad de electrones). Estas constituyen la primera aproximación a las energías de los estados excitados. De

igual forma los momentos de dipolo $\langle\phi_i|\boldsymbol{x}|\phi_a\rangle$ de cada uno de esas diferencias, serían un candidato para las fuerzas del oscilador. La decisión se sustenta en la representación de Lehmann de la respuesta lineal (ver Sección 2.2):

$$\chi_{KS}(\boldsymbol{x},\boldsymbol{x}',\omega) = \lim_{\eta\to o^+}\sum_{k,j}(f_k - f_j)\frac{\varphi_j^{(0)*}(\boldsymbol{x})\varphi_k^{(0)}(\boldsymbol{x})\varphi_j^{(0)*}(\boldsymbol{x}')\varphi_k^{(0)}(\boldsymbol{x}')}{\omega - (\varepsilon_j - \varepsilon_k) + i\eta} \tag{6.3}$$

donde existe una relación entre los polos de χ_{KS}, las energías de excitación y las fuerzas del oscilador con los numeradores de la eq. 2.26. Las diferencias de energías y los momentos dipolares entre estados de KS fueron obtenidas de los cálculos OCT-LCAO y GS-LDA. Teniendo en cuenta que el uso de un funcional híbrido como PBE0 en las propiedades a predecir utiliza la energía de intercambio de HT E_x^{HT} exacta los estado finales pueden intercambiarse, se consideró el índice de transferencia de cargas Δr para simular este comportamiento. Se escogieron los 20 primeros estados, teniendo en cuenta que las Ecuaciones de Casida ofrecen mejores resultados mientras mas estados de KS se agreguen. La Figura 6.4 muestra las distribuciones de las propiedades las diferencias de energías $\Delta E_{a,i}$, momentos dipolares $\langle\phi_i|\boldsymbol{x}|\phi_a\rangle$ y indice de transferencia de cargas Δr para las 21k moléculas de la 8CONF para los cálculos realizados.

Ambos cálculos ofrecen características bien diferentes: OCT-LCAO solo se realizó una aproximación básica a los orbitales de Kohn - Sham. No se realizó ninguna iteración del algoritmo *SCF* (ver Sección 2.1). Por otra parte GS-LDA es un cálculo completamente convergido. En otras palabras los orbitales obtenidos se optimizaron hasta alcanzar el mínimo de energías para el nivel de precisión. Puede notarse en la Figura 6.4 las similaridad en la distribución entre la primera fila OCT-LCAO y la segunda GS-LDA. Las diferencias entre los cálculos están marcadas por lo valores extremos en las distribuciones, los cuales pueden incidir en las predicciones de manera negativa, porque obliga al modelo a interpolar valores que se alejan de la media y que pueden ser considerados como ruido.

6.3 Métodos y Optimizaciones.

La pruebas realizadas utilizan los métodos de Regresión Ridge con Kernel (KRR) y Redes Neuronales (NN) (ver Sección 4.2) [4, 9, 13].

La optimización de hiper-parámetros para KRR es simple. Las variables a optimizar son sigma σ, que determina cómo cada molécula contribuye a la predicción final y lambda λ, parámetro de regularización del kernel (ver Sección 3.1.1). El rango de valores para ambas variables fue elegido de forma empírica y teniendo en cuenta experiencias anteriores (ver Anexo A.6).

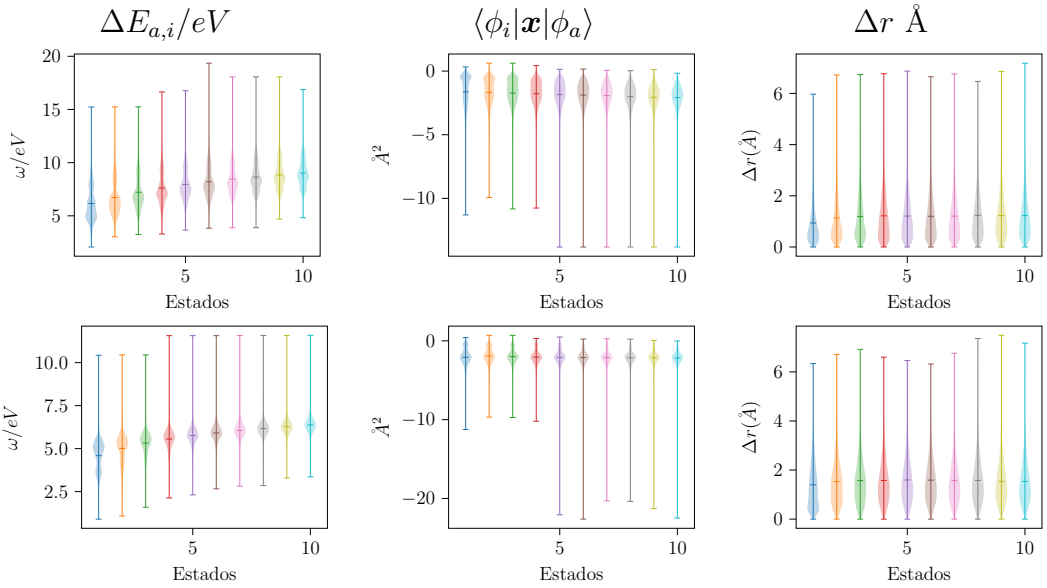

Figura 6.4: Distribución de las propiedades obtenidas a través de cálculos de bajo costo computacional. Primera fila OCT-LCAO y segunda fila GS-LDA. Por simplicidad se muestran solo los 10 primeros estados. A los momentos dipolares se les aplicó la función *log* para mejorar la visibilidad en la distribución

El caso de las NN resulta más complicado. Existen numerosos parámetros[3] los cuales podrían afectar la estimación. Dentro de las *funciones de activación* se pueden descartar aquellas que la literatura aconseja su uso en aplicaciones de clasificación, ej. *sigmoid* (ver Sección 3.2.3).

Entre los más demandantes desde el punto de vista computacional, están el *número de capas ocultas* (Hidden layers) y *cantidad de iteraciones de aprendizaje* (Epochs). El número de capas ocultas determina la cantidad capas de la red y por tanto el tamaño final de la topología, influyendo directamente en la cantidad de operaciones a realizar por los algoritmos de *Propagación hacia adelante* y *Propagación hacia atrás* (ver Sección 3.2.1 y 3.2.2). El número de iteraciones de aprendizaje va a determinar cuantas veces los algoritmos de *Propagación hacia adelantes* y *Propagación hacia atrás* se van a ejecutar sobre la topología diseñada. Para las optimizaciones de NN se utilizó la *búsqueda bayesiana* (ver Sección 3.3) la cual resulta una opción mas eficiente que una búsqueda en una maya de valores.

[3]La lista completa de los parámetros que son posible optimizar usando *Keras* y *Tensorflow* se puede ver en Anexo A.10

6.3.1 Preprocesamiento de datos

La aplicación de KRR se limitó a la predicción de las energías de los estados excitados, utilizando CM como descriptos (eq. 6.2), en una de sus versiones *organizada* (del inglés Sorted Coulomb Matrix, SCM en adelante) [108]. La SCM presenta una problemática con la dimensionalidad del sistema molecular. Para poder comparar varias moléculas de distinta dimensionalidad es necesario introducir un *zero-padding* que compense por la cantidad átomos faltantes.

Para NN se realizó un preprocesamiento de datos, particularmente un escalado. Los descriptores escogidos presentaban dispersiones muy diferentes y los valores están separados en varios ordenes de magnitud. Para que todos tengan la misma influencia sobre la predicción se realizó un escalado entre $0-1$ de todos los datos, descriptores y propiedades objetivas, utilizando los métodos implementados en el código [109] (ver Sección 4.2.3). Para algunos experimentos se aplicó la función $f(x) = log(x)$ (se utilizará *log* para denotar cuales modelos utilizan esta función) únicamente sobre los momentos de transición y las monoexitaciones tratando de mejorar la correlación entre ambos. Este proceso ocurre antes de el entrenamiento y es revertido una vez realizadas las predicciones.

6.3.2 Regresión Ridge con Kernel (KRR)

La optimización de los parámetros de los modelos de KRR se realizó tomando como referencia los valores obtenidos por Ramakrishnan y otros [4]. Entorno de ellos se construyó la maya para realizar la búsqueda. Se realizó una acotación previa de los extremos de los rangos de búsqueda para reducir lo más posible la cantidad de evaluaciones requeridas.

Se utilizaron 10k moléculas para entrenar ($\mathbb{T}$) y 1k moléculas en el grupo de control ($\mathbb{C}$). Ambos grupos fuero seleccionados aleatoriamente de las 15k moléculas a las que se realizaron cálculos PBE0 (CASIDA-PBE0). Se designaron las energías ω de los primeros 5 estados excitados como propiedad objetivo de la optimización. Se empleó una función laplaciana para el kernel definida por:

$$k(d_i, d_j) = e^{-\frac{|d_i - d_j|}{\sigma}} \tag{6.4}$$

donde $|*|$ es la norma l_1 de las diferencias entre los descriptores y σ es uno de los hiperparámetros a optimizar.

Cada energía fue predicha de forma independiente, por lo que se construyeron 5 modelos de KRR. La métrica del error utilizada fue *RMSE* (del inglés Root Mean Squared Error) definida como:

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=0}^{N}(p_i - r_i^{ref})^2} \tag{6.5}$$

donde N es el número de moléculas, p_i las predicciones realizadas por el algoritmo y r_i^{ref} los valores de referencia, en este caso las energías de excitación obtenidas del cálculo CASIDA-PBE0.

Además se emplearon dos descriptores: SCM y la diferencia de energía entre orbitales ocupados y no ocupados $\Delta E_{a,i}$ siendo el HOMO - LUMO la primera de estas diferencias. Esta energías fueron obtenida de los cálculos GS-LDA realizados. La diferencia de energía $\Delta E_{a,i}$ se encuentra en un orden de magnitud muy diferente al de la SCM y requería un ajuste de hiper-parámetros individual. Los rangos de búsqueda evaluados fueron:

$$\begin{aligned} \sigma_{SCM} &= [100, 1000, 2000, 3000, 5000] \\ \sigma_{\Delta E_{a,i}} &= [0.2, 0.3, 0.5, 1, 10, 20] \\ \lambda &= [10^{-6}, 10^{-3}, 0.1, 0.5, 1, 10] \end{aligned} \tag{6.6}$$

La maya de búsqueda se construyó de la combinación de estos valores.

La Figura 6.5 muestra la superficie generada por los hiper-parámetros y el RMSE calculados para el primer estado excitado CASIDA-PBE0 (Las restantes superficies pueden encontrarse en los Anexo A.7).

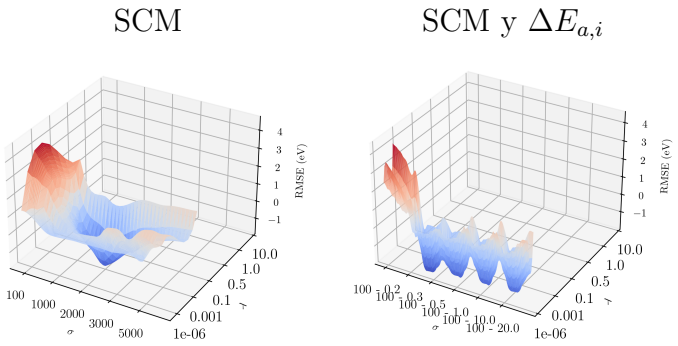

Figura 6.5: Resultados de la optimización de los hiper-parámetros sigma σ y lambda λ del KRR. Los mínimos puede apreciarse en color azul intenso.

Los valores óptimos de sigma σ y lambda λ para la SCM fueron encontrado en 1000 y 10^{-6} respectivamente. Por otra parte los valores óptimos de sigma σ para SCM y las diferencias de energías $\Delta E_{a,i}$ fueron 1000 y 0.2 respectivamente, y lambda λ 10^{-6}. Llama la atención como el valor de λ se mantiene bien pequeña en ambos casos, pero no es 0, indicando que mientras menos sea su influencia el algoritmo mejora su precisión (ver Sección 3.1.1).

6.3.3 Redes Neuronales (NN).

Para las NN, la búsqueda bayesiana requiere de límites y de un número máximo de iteraciones para encontrar un mínimo dentro de la maya construidas usando los límites. Estas optimizaciones resultan muy costosas tanto en tiempo como en capacidad de computo, debido a la necesaria evaluación de la NN para corregir el modelo sustituto (ver Sección 3.3).

Se exploraron 3 topologías de redes, dos pertenecientes al modelo *MLP* (del inglés Multi Layer Perceptron) y una relacionada con el modelo *CNN* (del inglés Convolutional Neural Network) (ver Sección 4.2). Se seleccionaron los hiperparámetros *cantidad de iteraciones de aprendizaje* (Nombrado *epochs* dentro de *Keras*. Se utilizará epochs a partir de ahora) y *número de capas ocultas* (del inglés Hidden Layers).

Estos fueron escogido a partir de varios experimentos empíricos y apoyados por la literatura relevante [9, 13]. En los experimentos realizados se alteraron progresivamente algunos de los parámetros accesibles a través de *Keras*, como por ejemplo la *función de activación* (ver Sección 3.2) donde inmediatamente se obtuvieron resultados desalentadores al utilizar funciones de la familia de *tanh* y *sigmoid*. Otros parámetros fueros utilizados en sus valores por defecto[4].

Evaluar las NN resulta costoso así que es recomendable realizar unos ajustes previos que ayuden convergencia de la búsqueda. No hay algoritmo definido de como hacer este proceso. La literatura consultada no arroja otro método que no sea prueba y error para ajustar los límites. Para *epochs* los límites de la maya fueron $[1, 1500]$, para el número de capas ocultas los límites fueron $[2, 30]$ [9, 32, 13, 10]. Se le permitió al algoritmo hacer 15 iteraciones sobre esta maya.

Como propiedades descriptoras se utilizaron los 20 primeras diferencias de energía $\Delta E_{a,i}$ con sus respectivos momentos dipolares y el índice del carácter de la transferencia de carga de la transición. Como propiedades a predecir se utilizaron los 10 primeros estados excitados junto a sus momentos de transición y el carácter de la transferencia de cargas de la transición.

Se realizaron dos variantes de dos experimentos sobre los modelos construidos, para un total de cuatro optimizaciones (ver sección 4.2.2). En la primera optimización los descriptores fueron extraídos de los cálculos GS-LDA. En la segunda, los descriptores se obtuvieron de OCT-LCAO. En una variante se aplicó la función *log* sobre los momentos de transición y las mono excitaciones, antes del aplicar el escalado de las propiedades, proceso que fue revertido después de las predicciones. En la otra variante los datos se mantuvieron sin cambio, solo se aplicó el escalado.

En todas las optimizaciones las propiedades a predecir se extrajeron de los cálculos CASIDA-PBE0. Se utilizaron 5k moléculas para entrenar ($\mathbb{T}$) y 1k como

[4]El Lector puede ver la lista completa de parámetros, los valores seleccionados y como se seleccionaron en Anexo A.10

grupo de control (ℂ). El error fue calculado sobre la métrica *Accuracy* definida en *Keras*, la cual mide la frecuencia en la que las predicciones coinciden con las referencias [89].

La Figura 6.6 muestra las superficies construidas y los mínimos encontrados en las mejores combinaciones de los hiper-parámetros seleccionados. La zonas de azul intenso marca los mínimos en cada caso.

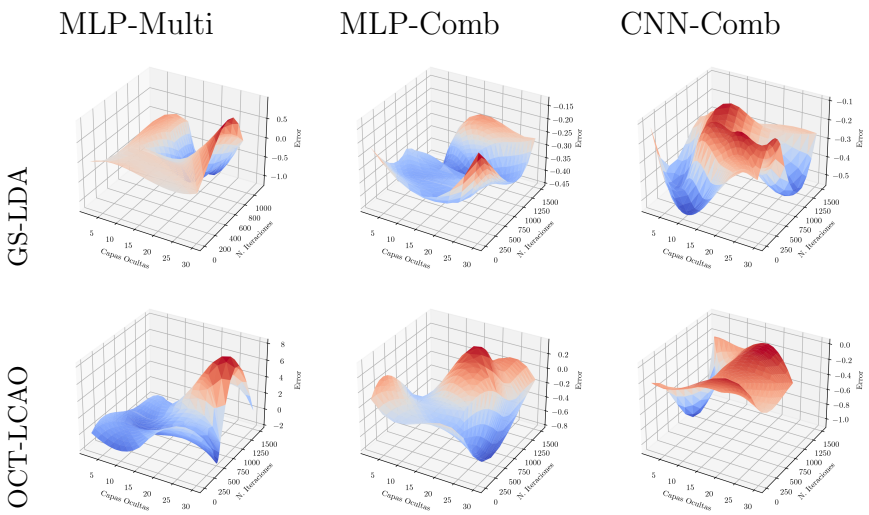

Figura 6.6: Superficies del espacio de hiper-parámetros, para las 3 topologías de redes evaluadas obtenidas durante las optimizaciones bayesianas. Primera fila descriptores obtenidos de GS-LDA, segunda fila descriptores obtenidos de OCT-LCAO. Solo se muestra la variante en la que no se utilizó la función *log* en el preprocesamiento.

Estas optimizaciones arrojaron las configuraciones que se muestran en las Tablas 6.1. Se puede apreciar como es necesario un aumento en la cantidad de *capas ocultas* cuando las propiedades descriptoras son extraídas de OCT-LCAO.

Este comportamiento puede ser consecuencia del *distanciamiento* entre los cálculos OCT-LCAO y los de CASIDA-PBE0. El algoritmo NN necesita compensar por las múltiples ejecuciones *SCF* que resultaría en la optimización de los orbitales de Kohn - Sham y la densidad ρ (ver Sección 2.1).

Otros parámetros utilizados relevantes para las topologías se muestran en la Tabla 6.2. Los valores por defectos fueron cambiados y fijados de forma empírica. Sobre estas configuraciones se construyeron los modelos con los que se realizaron los experimentos descritos en las siguientes secciones.

Tabla 6.1: Mínimos para las optimizaciones realizadas utilizado propiedades de cálculos GS-LDA y OCT-LCAO descriptoras y cálculos CASIDA-PBE0 como propiedades objetivas.

Modelo	Descriptor E_{xc}	Aplicó *log*	Iteraciones	Capas Ocultas
MLP-Multi	LCAO	No	576	30
MLP-Multi	LCAO	Si	620	22
MLP-Multi	LDA	No	796	15
MLP-Multi	LDA	Si	1756	17
MLP-Comb	LCAO	No	616	20
MLP-Comb	LCAO	Si	1320	28
MLP-Comb	LDA	No	516	8
MLP-Comb	LDA	Si	1419	4
CNN-Comb	LCAO	No	65	6
CNN-Comb	LCAO	Si	664	10
CNN-Comb	LDA	No	1500	2
CNN-Comb	LDA	Si	1500	2

Tabla 6.2: Otros parámetros pertenecientes a los modelos creados.

Modelo/Parámetro	MLP-Multi	MLP-Comb	CNN-Comb
Función de Activación	eLU/ReLU(negative slope = 0.01) [77, 75]		eLU/ReLU [77, 75]
Neuronas por Capa Oculta	60	25	60 (Filter)[5]
Optimisador	Adams [110]	Adams [110]	Adams [110]
Función de Pérdida	MSE	MSE	MSE
Dimensión del Kernel	-	-	20

6.4 Curvas de Aprendizaje

Resulta recurrente en la literatura el uso de varias pruebas para demostrar la capacidad del método para aprender [4, 9, 13, 19, 20, 21]. Esto se aprecia a través de un decrecimiento en los valores de la métrica utilizada para evaluar el error a medida que aumenta la cantidad de datos contenida en el conjunto de entrenamiento $\mathbb{T}$.

Este tipo de pruebas se realiza con los hiper-parámetros optimizados. Permite además identificar problemas como *sobre-ajuste* o *poco-ajuste* (ver Sección 3.1.1) del modelo, debido a que el comportamiento esperado de decrecimiento del error no se cumple.

6.4.1 Regresión Ridge con Kernel (KRR).

Para obtener la curvas de aprendizaje utilizando KRR se realizaron 5 experimentos. Con cada iteración de aumentó progresivamente la cantidad de moléculas el grupo de aprendizaje ($\mathbb{T}$). Los hiper-parámetros utilizados fueron los obtenidos en la Sección 6.3.2. Se compararon los 2 descriptores seleccionados SCM y SCM - $\Delta E_{a,i}$. Como propiedad objetiva se emplearon las energías ω de los 5 primeros estados excitados obtenidas del cálculo CASIDA-PBE0.

Las tendencia de ambas curvas mostrada en la Figura 6.7, denota la capacidad del algoritmo de incorporar la nueva información. La comparación de los dos descriptores se hace evidente, la dispersión mostrada para cada estado se reduce notablemente con la adición de $\Delta E_{a,i}$.

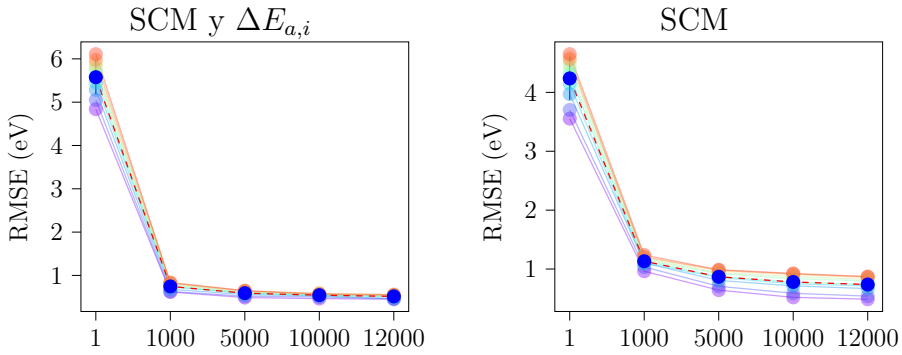

Figura 6.7: Curvas de Aprendizaje, donde — es la media del RMSE y • representan las energías ω de cada estado. Se puede notar como la utilización de dos descriptores, SCM y $\Delta E_{a,i}$, estabiliza notablemente las predicciones de todos los estados.

6.4.2 Redes Neuronales (NN).

Para construir las curvas de aprendizaje utilizando NN se tuvo en cuenta la predicción de las tres propiedades objetivas, *estados excitados*, *momentos de transición* y *índice de transferencia de cargas*. Cada una de ellas puede tener un comportamiento diferente en el aprendizaje. Sin embargo la tendencia de mejora tiene que mantenerse.

Se construyeron dos grupos de curvas de aprendizaje con con dos variantes si se utilizó o no la función *log*: (1) utilizando los cálculos GS-LDA como propiedades descriptoras y (2) utilizando los cálculos OCT-LCAO como propiedades descriptoras. Para ambos casos se realizaron 5 experimentos aumentando el tamaño del grupo de aprendizaje ($\mathbb{T}$) y utilizando los cálculos de CASIDA-PBE0 como propiedad a predecir. El error fue calculado sobre la métrica *Accuracy* definida en *Keras*, que tiene un rango entre $[0, 1]$ siendo los valores cercanos a 1 los deseables.

En la Figura 6.8 se muestran las de curvas de aprendizaje, utilizando como descriptores los cálculos OCT-LCAO y GS-LDA.

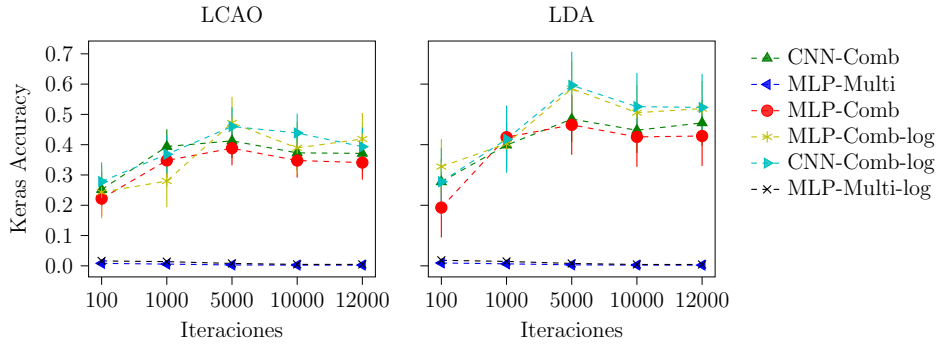

Figura 6.8: Curvas de aprendizaje para los modelos MLP-Multi, MLP-Comb y CNN-Comb. Propiedades descriptoras extraídas del cálculo OCT-LCAO y GS-LDA.

Se puede notar en las Figura 6.8 que existe un pico donde los modelos tienen su mejor desempeño justo cuando se utilizan 5k moléculas de aprendizaje. Esto puede ser el resultado de la utilización de este número de moléculas en la optimización bayesiana, y los modelos responden con un máximo de ajuste a este valor. Puede observarse como cae la precisión para valores mayores de 5k pero sin llegar a ser notable. Hay que destacar como la aplicación de la función *log* en el preprocesamiento de datos mejora el rendimiento de algunos modelos como el basado en CNN.

6.5 Resultados.

Los resultados obtenidos se basan en la estadística realizada sobre múltiples experimentos. De las 15k moléculas disponibles en la base de datos 8CONF con cálculos realizados CASIDA-PBE0, se seleccionaron 12k para el entrenamiento y 1k como grupo de control. Ambos grupos fueron seleccionados aleatoriamente.

Se realizaron dos grupos de 10 experimentos. Ambos grupos están enfocados en comprobar la robustez del los modelos en la base de datos 8CONF. En el primer grupo experimentos para cada iteración se realizó un re-muestreo del grupo de entrenamiento seleccionando 10k moléculas aleatoriamente para ser utilizadas en el aprendizaje ($\mathbb{T}$). El grupo de control ($\mathbb{C}$) no sufrió cambios. Para el segundo grupo de experimentos se seleccionaron los modelos entrenados con 5k y 10k. Se realizó un re-muestreo del grupo de control ($\mathbb{C}$) seleccionando 1k de las 15k moléculas con cálculos CASIDA-PBE0 [103].

Los valores discretos de la *Densidad de Estados* y el *Espectro de Absorción* se representaron con una gaussiana de amplitud 0.15eV. Para el espectro de absorción el área de las gaussianas es proporcional a las fuerzas del oscilador. Para la densidad de estados el área de las gaussianas representa el número de estados en un rango de energía.

En cada predicción realizada sobre las propiedades fue evaluada individualmente se utilizaron varias métricas:

- Para el espectro discreto: RMSE definido en eq. 6.5 y r-factor [111] definido por $1 - \frac{\sum(r_i - r_i^{ref})^2}{\sum(r_i^{ref} - media(r))^2}$.
- Para el espectro continuo: Se utilizó la *correlación cruzada* [109] de la reconstrucción normalizada, la diferencia de área bajo la curva entre referencia y predicción y el desplazamiento entre las curvas.

Los resultados obtenidos se presentan como una media de la desviación entre todas las iteraciones realizadas.

6.5.1 Densidad de Estados.

Para la construcción de la densidad de estados se emplearon los primeros 5 estados excitados para el algoritmo KRR, rango que se extendió a 10 para los modelos de NN. Este número fue decidido arbitrariamente y los métodos propuestos son capaces de aumentar la cantidad de estados a estimar. Sin embargo a medida que crece la energía la probabilidad de encontrar estados excitados en ese rango aumenta y tampoco resultaba atractivo para el espectro UV-Visible explorar estados por encima de la energía de ionización (-ϵ_{HOMO}).

Regresión Ridge con Kernel (KRR).

Utilizando KRR como método de aprendizaje se emplearon dos descriptores diferentes la *Sorted Coulomb Matrix* (en adelante SCM) la cual es una variante de *Coulomb Matrix*. El segundo descriptor es una combinación de la SCM y las diferencias de energías $\Delta E_{a,i}$ obtenidas en el cálculo GS-LDA. La propiedad objetivo fue en ambos casos los 5 primeros estados excitados de CASIDA-PBE0.

Se utilizaron los valores de hiper-parámetros $\sigma_{SCM} = 1000$, $\sigma_{\Delta E_{a,i}} = 0.2$ y $\lambda = 10^{-6}$ encontrados con la búsqueda en la maya (*grid search*).

La Figura 6.9 muestra los errores de los 5 primeros estados a través de los 10 experimentos realizados. La consistencia del algoritmo y los hiper-parámetros encontrados se muestra en pequeñas variaciones en el RMSE a través de las 10 iteraciones.

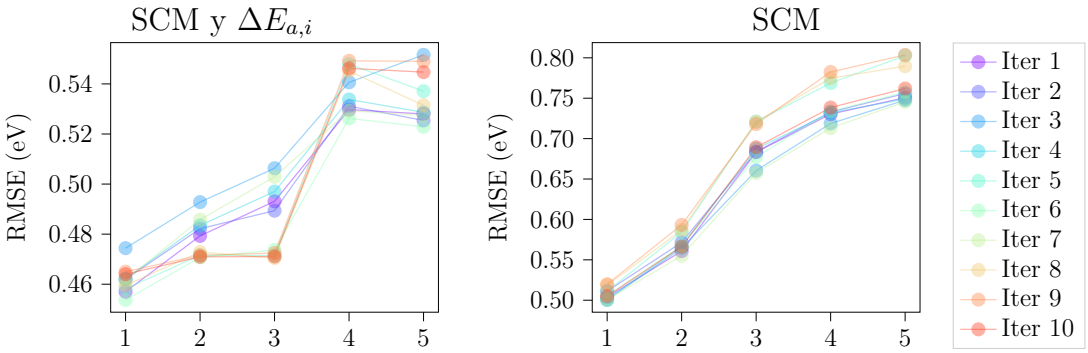

Figura 6.9: Curva de RMSE por cada estados para cada una de las 10 iteraciones realizadas. Puede notarse como la inclusión del descriptor $\Delta E_{a,i}$ mejora notablemente los resultados del RMSE.

La Tabla 6.3 muestra un resumen de estos experimentos. Para cada estado se calculó la media del RMSE obtenido de las 10 iteraciones. En rojo se muestra que las energías del primer estado son las de menor error, el descriptor SCM ya demostró su éxito prediciendo el primer estado excitado [4]. Es notable también como la introducción de la diferencia de energías $\Delta E_{a,i}$ mejora los resultados en todos los estados.

Del resultado de este experimento se puede observar que el RMSE utilizando el modelo KRR es alto, 0.46 eV para el primer estado usando SCM y $\Delta E_{a,i}$ y 0.50 eV usando solamente SCM. Sin embargo hay que considerar el salto cualitativo entre los descriptores, una combinación entre la geometría de la molécula y la diferencia de energía entre orbitales de Kohn-Sham, y las propiedades objetivos, estados excitados utilizando un funcional híbrido.

El método KRR por su simpleza es muy sencillo de implementar, pero presenta limitaciones. Su mejor desempeño se alcanza cuando los descriptores y las

Tabla 6.3: Media del RMSE de las energías de excitación para los primeros 5 estados. Obtenidas mediante el uso modelo de KRR.

Estados	Media RMSE SCM y $\Delta E_{a,i}$	Media RMSE SCM
1	0.461767	0.507682
2	0.478116	0.571691
3	0.484771	0.689880
4	0.537917	0.742515
5	0.534626	0.766405

propiedades objetivos presentan una correlación explícita. Esta es la razón por la cual las diferencias de energías $\Delta E_{a,i}$ agregan precisión a la predicción. Al provenir de un cálculo convergido y orbitales de Kohn-Sham optimizados, el método debe compensar por el cambio de funcional de LDA a PBE0.

Las correlaciones implícitas son más complicadas. Tal es el caso de los momentos de transición μ y por consiguiente las fuerzas del oscilador. La información codificada dentro de la configuración atómica en la forma de distancias entre átomos, tipos de átomos, cantidad de electrones, etc., no es fácilmente alcanzable utilizando KRR. Además después de agregar las mono-excitaciones correspondientes a cada una de las diferencias de energías $\Delta E_{a,i}$ como corrección, no se obtuvieron resultados que alentaran seguir persiguiendo esta línea de investigación.

6.5.2 Espectro de absorción.

Utilizando NN se emplearon solamente descriptores electrónicos extraídos del los cálculos realizados GS-LDA y OCT-LCAO. Se utilizaron las primeras 20 diferencias de energías ($\Delta E_{a,i}$), las correspondientes momentos dipolares ($\langle\phi_i|\boldsymbol{x}|\phi_a\rangle$) y los índice ($\Delta r$) de transferencia de cargas relacionados con las 20 diferencias.

Las propiedades objetivo energías de excitación (ω), momentos de transición (μ^2) y el índice de transferencia de carga (Δr) fueron extraídas del cálculo CASIDA-PBE0. La cantidad de estados a predecir se aumentó a 10.

Cada una las topologías de NN investigadas (ver Sección 4.2.2) combina los descriptores de manera diferente:

- MLP-Multi: Considera los descriptores como tensores independientes. Las propiedades objetivo son predichas en tensores independientes. Esta fue la primera consideración dentro de la línea de investigación por la simplicidad.

- MLP-Comb: Considera los descriptores como una combinación de las 3 propiedades. Con las propiedades a predecir se realizó una combinación similar.

- CNN-Comb: Los descriptores y las propiedades objetivo son combinadas de igual forma que MLP-Comb. Esta topología aporta una mayor densidad de *neuronas* en las capas ocultas que las topologías pertenecientes a los modelos MLP.

Solamente se utilizaron NN para la reconstrucción del espectro de absorción. La incapacidad de KRR para producir momentos de transición y por consiguiente fuerzas de oscilación [4], no alentaba a su utilización.

Reconstrucción del espectro de absorción.

Se realizaron dos grupos de experimentos cada uno con 10 iteraciones sobre las tres topologías construidas utilizando los hiper-parámetros obtenidos en las optimizaciones (ver sección 6.3.3).

El primer grupo de experimentos (*Estabilidad de Aprendizaje*) de las 15k moléculas para las que se disponían los cálculos CASIDA-PBE0 se utilizaron 12k como grupo de entrenamiento. Sobre este grupo se realizó un re-muestreo de 10k moléculas utilizadas para entrenar los modelos en cada experimento ($\mathbb{T}$). Se utilizaron 1k como grupo de control ($\mathbb{C}$) que permaneció constante durante todas las iteraciones.

En el segundo grupo de experimentos (*Estabilidad de Predicción*), se utilizaron modelos previamente entrenados con 5k y 10k moléculas. De las 15k moléculas para las que se disponían los cálculos CASIDA-PBE0, se realizaron 10 iteraciones seleccionando 1k de forma aleatoria y se realizaron sus predicciones utilizando los modelos entrenados

Los resultados mostrados representan una media de los experimentos realizados y pertinentemente se tomarán algunas iteraciones como caso de estudio.

La Figura 6.10 se muestran la 10 iteraciones realizadas, para el primer estudio *Estabilidad de Aprendizaje*. Con este experimento se busca comprobar que los modelos ofrecen similares resultados al aprender de distintas moléculas. Se puede notar una homogeneidad en las predicciones independientemente del cambio en la base de entrenamiento. Puede apreciarse también que el modelo MLP-Multi por su simplicidad es incapaz de captar las correlaciones y mantiene un pobre desempeño.

En el siguiente grupo de experimentos se seleccionaron modelos entrenados con 5k y 10k moléculas. Sobre estos modelos se realizaron 10 iteraciones prediciendo 1k moléculas seleccionadas aleatoriamente de las 15k que tenían cálculos CASIDA-PBE0. La Figura 6.11 muestra como los modelos se comportan de manera estable al realizar predicciones de moléculas fuera de la base de entrenamiento.

De las Figuras 6.10 y 6.11 se pueden extraer varias conclusiones que permitirán avanzar en la propuesta de los mejores modelos. Los modelos basados en

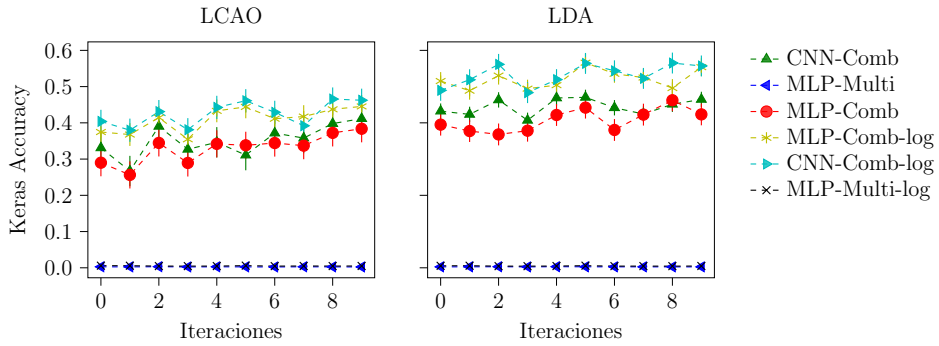

Figura 6.10: Comparación de los modelos a través de los experimentos de *Estabilidad de Aprendizaje* realizados. Los resultados obtenidos muestran que todos los modelos se muestran muy estables cuando la base de aprendizaje varia.

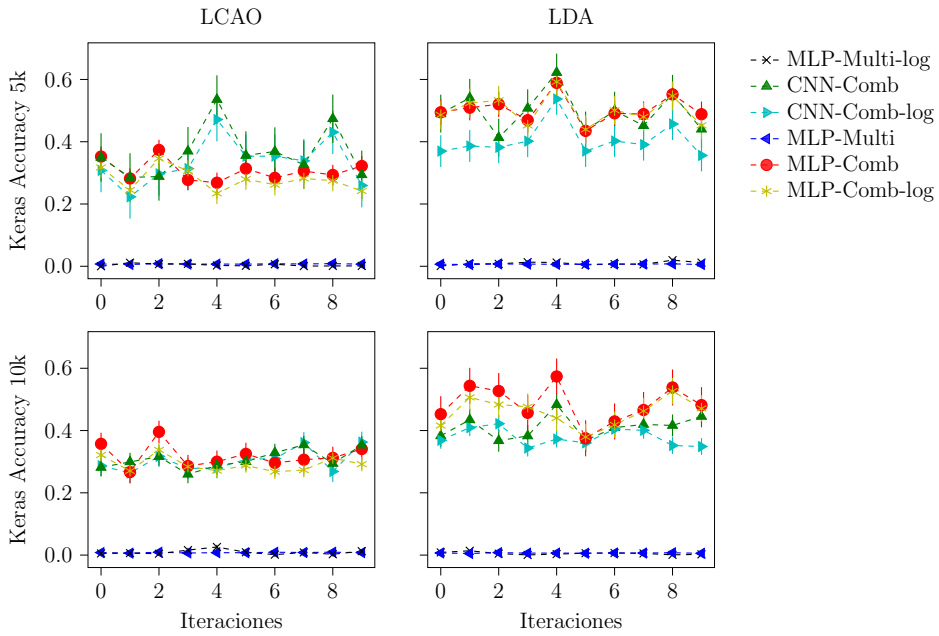

Figura 6.11: Comparación de los modelos a través de los experimentos de *Estabilidad de Predicción* realizados. La primera fila desde arriba muestra los resultados para los modelos entrenados con 5k moléculas, la segunda fila muestra los resultados para los modelos entrenados con 10k moléculas.

múltiples entradas y salidas (MLP-Multi) tienen un pobre desempeño, la manera en que se tratan los datos, como entidades independientes, puede ser la causa de este problema. Los modelos en donde las entradas y las salidas resultan de una combinación de los datos, se comportan de mejor manera. La introducción de la función *log* como preprocesamiento tanto de los momentos dipolares como de

Tabla 6.4: Comparativa de los modelos de NN usando la media de las 10 iteraciones de los RMSE obtenidos para cada propiedad por estados. Los descriptores utilizados fueron obtenidos de los cálculos GS-LDA y las propiedades objetivos de CASIDA-PBE0.

Modelo	MLP-Comb-log			CNN-Comb-log		
Estado	μ^2 ($\AA^2$)	ω (eV)	Δr ($\AA$)	μ^2 ($\AA^2$)	ω (eV)	Δr ($\AA$)
1	0.045394	0.166597	0.469226	0.047093	0.143380	0.374006
2	0.072149	0.170443	0.569405	0.137294	0.143029	0.501320
3	0.080705	0.160787	0.560417	0.125611	0.152692	0.543971
4	0.461331	0.149469	0.587609	0.104215	0.157123	0.572674
5	0.069411	0.128695	0.627864	0.068970	0.138006	0.634554
6	0.069182	0.117786	0.619804	0.053781	0.134733	0.644160
7	0.055845	0.112092	0.638577	0.056301	0.122426	0.660371
8	0.078380	0.114583	0.624572	0.065833	0.119953	0.673686
9	0.046128	0.100828	0.586439	0.046243	0.107928	0.640463
10	0.089106	0.093832	0.607456	0.052175	0.111786	0.651670

transición, ayuda en la correlación de estas dos propiedades, porque ayuda a espaciar la alta concentración de valores cerca del 0. La diferencia entre los resultado obtenidos con modelos entrenados con 5k moléculas y 10k moléculas (ver Figura 6.11) resulta más notable cuando se utiliza GS-LDA como propiedad descriptora que cuando se utiliza OCT-LCAO. Dos factores intervienen: (1) la calidad de los datos donde el salto cualitativo entre LDA y PBE0 resulta mas simple y (2) la cantidad de datos donde la utilización de 5k moléculas de un cálculo convergido (GS-LDA) parece aportar más información al modelo que similar cantidad de un cálculo sin converger (OCT-LCAO). Además se puede notar en la Figura 6.11 que utilizar más datos estabiliza el comportamiento de los modelos.

El siguiente paso sería el análisis del error por propiedades[6]. Para esto se seleccionaron aquellos modelos que mejor desempeño demostraron en los experimentos mostrados en las Figuras 6.10 y 6.11: CNN-Comb y MLP-Comb. En ambos utilizando la función *log* en el preprocesamiento de datos. Los datos mostrados en la Tabla 6.4 son una media de los RMSE por propiedades extraída de las 10 iteraciones realizadas para el experimento de *Estabilidad en el Aprendizaje* utilizando GS-LDA como propiedad descriptora, por tanto cada modelo fue entrenado con 10k moléculas y se utilizó 1k como grupo de control.

Por propiedades se puede apreciar que de los dos modelos CNN-Comb-log obtiene mejores resultado. Las energías de excitación presentan un error 0.14 eV para el primer estado, cerca de la precisión química $0.05 - 0.1$eV. Este comportamiento también se puede apreciar con el indice de transferencia de carga Δr donde el error es de 0.6 Å para ambos modelos.

[6]Ver la Anexos A.9 donde se ofrecen otras imágenes que permiten visualizar los resultados.

Tabla 6.5: Comparativa de los modelos de NN usando la media de las 10 iteraciones de los RMSE obtenidos para cada propiedad por estados. Los descriptores utilizados fueron obtenidos de los cálculos OCT-LCAO y las propiedades objetivos de CASIDA-PBE0.

Modelo	MLP-Comb-log			CNN-Comb-log		
Estado	μ^2 ($\AA^2$)	ω (eV)	Δr ($\AA$)	μ^2 ($\AA^2$)	ω (eV)	Δr ($\AA$)
1	0.049191	0.433375	0.905870	0.049098	0.425230	0.775626
2	0.076018	0.389037	0.835537	0.075770	0.397094	0.796773
3	0.082436	0.366787	0.793986	0.082309	0.375579	0.765611
4	0.063565	0.332853	0.806719	0.063460	0.348139	0.771380
5	0.070782	0.310724	0.801527	0.070711	0.326954	0.796297
6	0.054307	0.321082	0.814103	0.054131	0.331419	0.787129
7	0.056875	0.314362	0.839193	0.056983	0.327621	0.804060
8	0.064708	0.323515	0.765498	0.064410	0.332830	0.747880
9	0.046543	0.318134	0.761843	0.046476	0.324391	0.731851
10	0.050415	0.326052	0.785419	0.050431	0.331603	0.757923

La diferencia más marcada entre los modelos está en los momentos de transición μ^2. Esta propiedad presenta una distribución donde la mayor concentración de valores estaba cerca del 0 (ver Figura 6.2). Después de aplicada la función *log* los valores de RMSE de $0.1\AA^2$ son considerados grandes.

El mismo análisis se puede hacer en un *salto* en la Escalera de Jacob más atractivo. La Tabla 6.5 muestra un media del RMSE para las 10 *iteraciones* realizadas para el experimento de Estabilidad en el Aprendizaje. Para este caso los descriptores utilizado provienen del cálculo OCT-LCAO. Puede notarse un aumento significativo de la media del RMSE. El error en las energías de excitación se duplica si se compara con la Tabla 6.4. Lo mismo sucede con la propiedad Δr que ve un aumento en el error de 0.3 y 0.2 Å para MLP-Comb y CNN-Comb respectivamente.

Resulta llamativo los valores del RMSE asociados a los momentos de transición μ^2. La variabilidad del error en esta propiedad es muy pequeña y siempre se producen magnitudes similares. Más adelante se retomará este problema.

Siendo el salto desde LCAO a PBE0 más atractivo, se puede seleccionar una de la iteraciones mostrada en la Figura 6.10 del experimento para probar la Estabilidad en el Aprendizaje. Sobre esta iteración hacer un análisis más profundo del error de estas propiedades e identificar fortalezas y debilidades.

Utilizando los datos de la primera iteración, se puede reconstruir la *densidad de estados* como una suma de gaussianas con un ensanchamiento de 0.15 eV. Donde las intensidades son proporcionales al número de estados en ese rango de energía.

La Figura 6.12 muestra la reconstrucción de la densidad de estados para 3 moléculas de la 1k que componen el grupo de control $\mathbb{C}$. Estas moléculas represen-

tan la *Peor*, el *Promedio* y la *Mejor* predicción en cada modelo. Criterio basado en la correlación cruzada normalizada (ver Sección 4.2.3) entre la reconstrucción usando las energías de las predicciones y las obtenidas de CASIDA-PBE0.

El modelo CNN-Comb muestra la mejor predicción promedio con una correlación cruzada de 0.942 de un máximo de 1 y un desplazamiento de 0.080. La fortaleza de los dos modelos viendo su peor desempeño donde los dos modelos mantienen una correlación cruzada por encima de 0.6 y un máximo de 0.750 alcanzado por CNN-Comb. Hay que destacar que tanto CNN-Comb como MLP-Comb presentan problemas en la misma molécula $C_3HFN_2O_2$, de los dos modelos CNN-Comb mejora la estimación de las energías. También resulta interesante que las moléculas mejor predichas presentan las 10 primeras energías en un rango pequeño dentro de la maya, un importante detalle que solo es visible en este tipo de representación aumentando el ensanchamiento de la gaussiana.

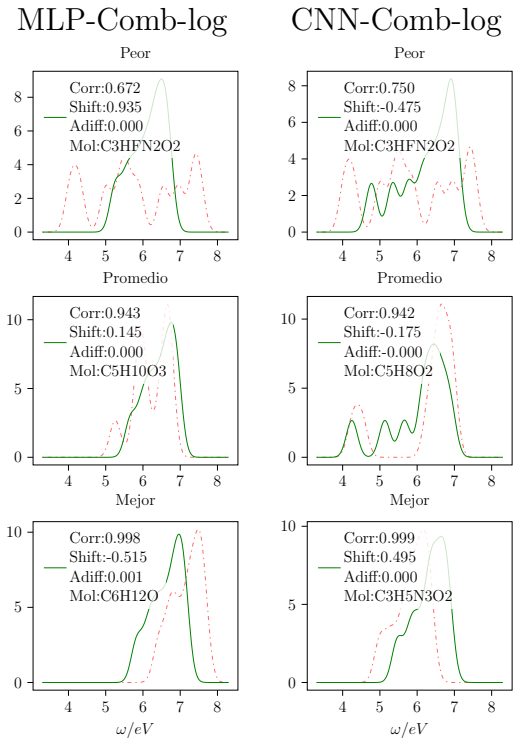

Figura 6.12: Comparativa de la reconstrucción de la densidad de estados para los modelos de NN utilizados, usando datos del cálculo OCT-LCAO como descriptor. De arriba a abajo *Peor*, *Promedio* y *Mejor* predicción. Se muestran la predicciones como - y la referencia como –.

En general ambos modelos se comportan de manera estable. En la Figura 6.13 se puede observar tres histogramas con los resultado de las 1k moléculas

utilizadas como grupo de control $\mathbb{C}$. La correlación cruzada muestra que una densidad cercana al 1. Los la media de RMSE mostrados en la Tabla 6.5 en la energía de excitación de 0.3 eV pueden notarse en la métrica *desplazamiento* (segunda columna en la Figura 6.13), donde la mayor densidad de valores se ubican al rededor del 0. La métrica utilizada para evaluar el área bajo la curva (tercera columna en la Figura 6.13), muestra una distribución donde todos los valores están muy cerca del 0, las pequeñas discrepancias que se muestran son producto del ensanchado usado en las gaussianas y de que existen moléculas con estados en los límites del rango de energía seleccionado.

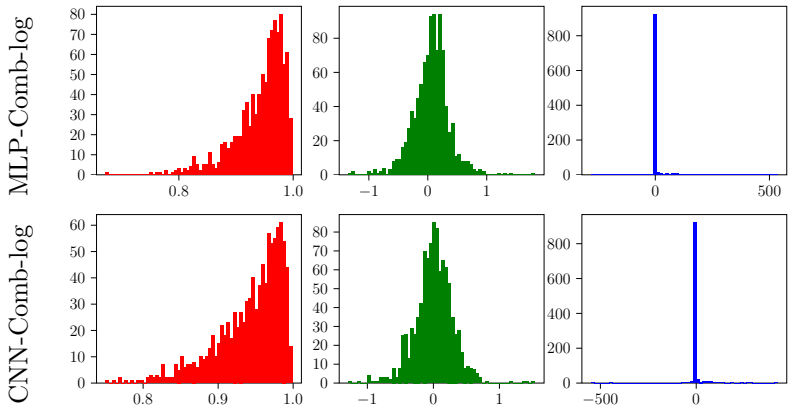

Figura 6.13: Histograma de las métricas utilizadas para evaluar el error en la reconstrucción de la densidad de estados usando datos del cálculo OCT-LCAO como descriptor: Correlación cruzada (rojo), desplazamiento (verde) y diferencia del área bajo la curva (azul).

Pasando a los resultados de la propiedad momentos de transición μ^2, la Figura 6.14 muestra la reconstrucción del espectro de absorción como la sumatoria de gaussianas sobre una malla con un ensanchamiento ded 0.15 eV y las intensidades son proporcionales a las fuerzas del oscilador obtenidas a partir de las energías ω y los momentos de transición μ^2 usando $f_i = \frac{2}{3}\frac{\omega_i}{\hbar^2}\sum {\mu_i}^2$ donde $\hbar$ es la constante de Plank reducida. Se pueden observar en la Figura 6.14 como los modelos escogidos encuentran problemas al predecir los momentos de transición, algo que se potencia considerando la acumulación de los errores producidos por las predicciones de las energías. Los peores resultados de los modelos MLP-Comb y CNN-Comb (Figura 6.14, primera fila desde arriba, segunda y tercera imagen desde la izquierda) se ubican por encima de 0.5.

La Figura 6.15 muestra una reconstrucción discreta de los espectros de absorción, la escala de colores representa el índice de transferencia de carga Δr. Se puede apreciar como los modelos son capaces de predecir el cambio de funcional al mantenerse la misma escala de colores o con pequeñas variaciones.

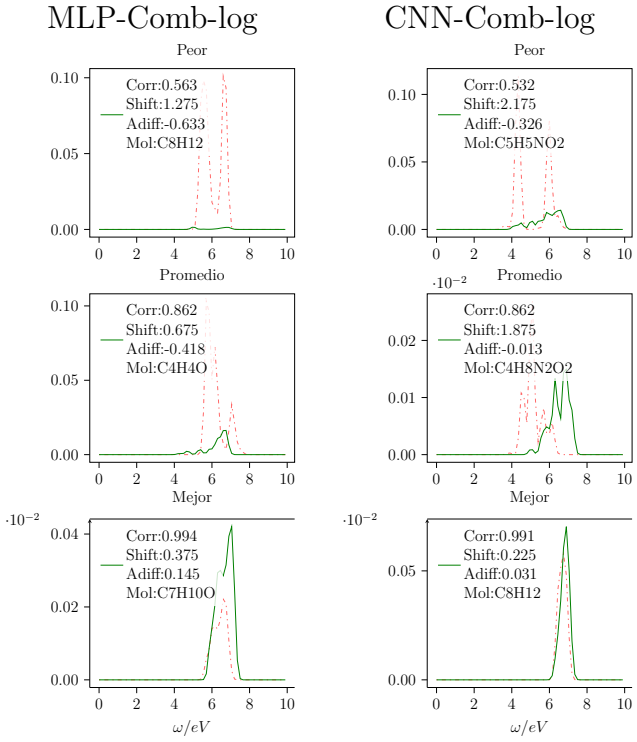

Figura 6.14: Comparativa de la reconstrucción de los espectros de absorción para los modelos de NN utilizados, usando datos del cálculo OCT-LCAO como descriptor. De arriba a abajo el *Peor*, el *Promedio* y la *Mejor* predicción. Se muestran la predicciones como - y los espectros de referencia como –.

En la Figura 6.16 se muestran las métricas escogida para para el análisis de las curvas de los espectros. En la correlación cruzada (primera columna en la Figura 6.16) muestra que las curvas reconstruidas encuentran en la mayor concentración entre $0.8-1$ lo cual es alentador. Pero los tres histogramas en su conjunto muestran que hay espacio para la mejora.

El análisis de estos errores se puede hacer desde el punto de vista Físico/ Químico y numérico. Considerando que las propiedades descriptoras provienen de un cálculo completamente convergido, los momentos dipolos obtenidos tienen relación con los momentos de transición aunque no de manera directa (ver eq 6.3). Los modelos de NN deberían compensar por el término energía de intercambio E_x exacto usado en funcionales híbridos (ver Sección 2.1.1) que no está presente en LDA.

El análisis numérico de los errores puede arrojar información de donde los modelos encuentran problemas como *poco-ajustado* o *sobre-ajustado*. Utilizando la métrica *r-factor* (ver Sección 6.5) se puede evaluar la correlación entre las

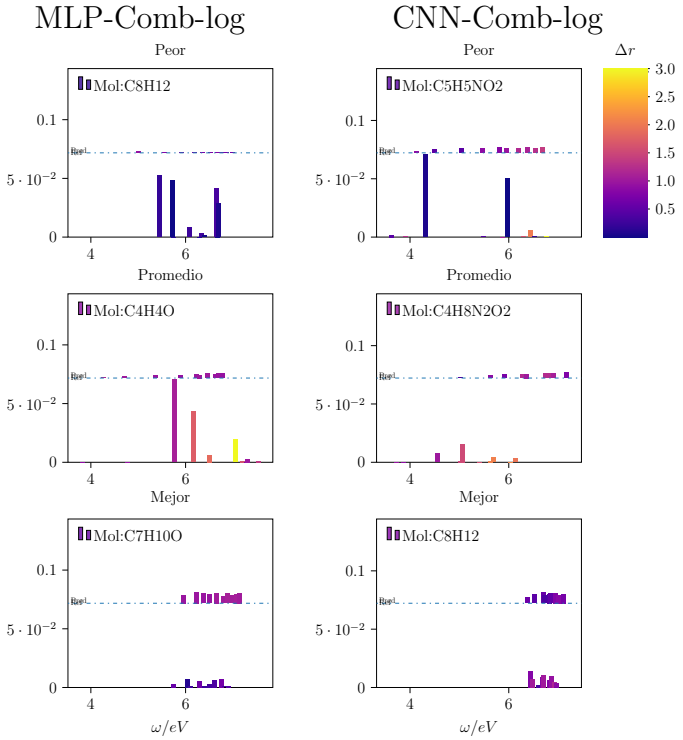

Figura 6.15: Comparativa de la reconstrucción discreta de los espectros de absorción para los modelos de NN utilizados. De arriba a abajo *Peor*, *Promedio* y *Mejor* predicción. La escala de colores representa el indice del carácter de transferencia de carga de la transición.

propiedades predichas y las referencias en el grupo de control $\bar{\mathbf{R}}^{\mathbb{C}}$.

Los valores de *r-factor* oscilan entre $[-1, 1]$, donde -1 implica correlación inversa, 1 representa correlación perfecta y los valores cercanos a 0 implican ninguna correlación. La Figura 6.17 muestra esta métrica por estados para las 10 iteraciones realizadas sobre los dos mejores modelos en el experimento de *Estabilidad en el Aprendizaje*. Del comportamiento de esta métrica se puede apreciar la calidad de las las predicciones por propiedades.

La propiedad μ^2 (Figura 6.17 primera fila desde arriba) presenta una pobre correlación. Los modelos encuentran dificultades en determinar el comportamiento de esta propiedad. Los valores negativos del *r-factor* implican la existencia de una correlación inversa. Pero estos valores resultan muy cercanos al 0 como para clasificarlo como una fuerte correlación inversa. Ambos modelos mantienen el mismo comportamiento.

La energías de los estados excitados ω (Figura 6.17 segunda fila desde arriba) tienen muy buena correlación la cual va disminuyendo a mediad que se asciende la

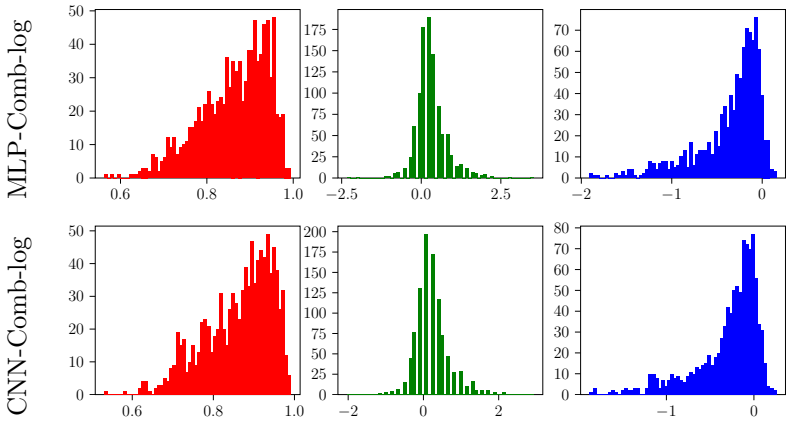

Figura 6.16: Histograma de las métricas utilizadas para evaluar el error en la reconstrucción del espectro de absorción, usando datos del cálculo OCT-LCAO como descriptor: Correlación cruzada (rojo), desplazamiento (verde) y diferencia del area bajo la curva (azul) .

escala de energía y empiezan a aparece estados con energías similares. Se pueden notar como en todos los experimentos los modelos se mantuvieron tan estables que casi hay un solapamiento de las curvas.

El índice de transferencia de carga Δr (Figura 6.17 tercera fila desde arriba) mantiene una correlación directa pero no tan fuerte como puede verse en los resultados de las energías de excitación. El comportamiento de ambos modelos es similar al caso de las energías de excitación aunque la precisión de la predicción sufre más con el cambio de base de entrenamiento. Esto se aprecia en la dispersión de resultados por estados.

6.5.3 Evaluando los modelos sobre otra Base de datos.

Como prueba final se buscó evaluar el modelo de NN ya entrenado sobre moléculas completamente fuera de la 8CONF. Se contaba con 15000 moléculas extraídas de una base de datos (en adelante ZINC-DB) de compuesto orgánicos [112]. Sobre estas moléculas no se había realizado ningún cálculo DFT ni optimización de geometrías. Se seleccionaron 20 de forma aleatoria, procurando siempre que se mantuvieran fuera de la 8CONF, teniendo en cuenta cantidad de átomos y la inclusión de elementos nuevos.

Estas 20 moléculas están compuestas por Carbono (C), Oxígeno (O), Nitrógeno (N), Flúor (F), Cloro (Cl) y Azufre (S).

Sobre estas moléculas se realizaron los cálculos de TD-DFT pertinentes usando *Octopus* (ver Sección 4.1), para obtener las propiedades electrónicas necesarias con los niveles de precisión utilizados, LCAO, LDA y PBE0. En las Figuras 6.18 y 6.19

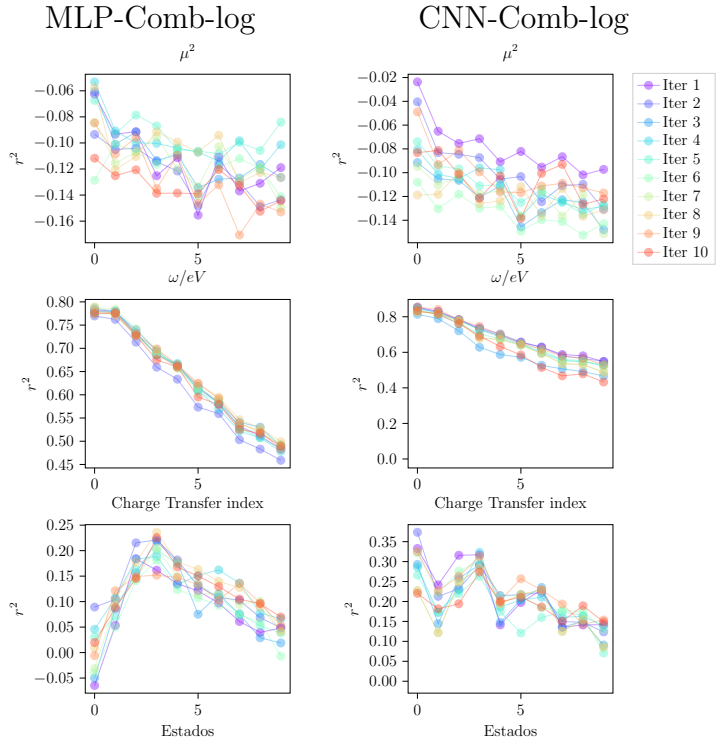

Figura 6.17: Comparación de los modelos a través de los experimentos realizados, usando datos del cálculo OCT-LCAO como descriptor.

se pueden observar las distribución de los descriptores y las propiedades objetivo respectivamente (ver Secciones 6.2.2 y 6.2.1).

Se utilizaron los modelos que mejores resultados arrojaron, MLP-Comb-log y CNN-Comb-log entrenados con 10k moléculas de la 8CONF. Ambos modelos fueron empleados en la prueba *Estabilidad en el aprendizaje* (ver Sección 6.5.2). Los hiperparámetros utilizados fueron los obtenidos por las optimizaciones Bayesianas (ver Sección 6.3.3). La Figura 6.20 muestra la reconstrucción del espectro de absorción discreto y continuo utilizando los mismos parámetros para las gausianas que los mostrados en las Seección 6.5.2. En todos los casos los momentos de transición μ^2 resultaron sobre estimados. Aunque las correlaciones cruzadas muestran valores alentadores, se pueden observar valores de desplazamiento y de diferencias de áreas bajo la curva demasiado grandes. Un indicativo que las predicciones realizadas fueron pobres.

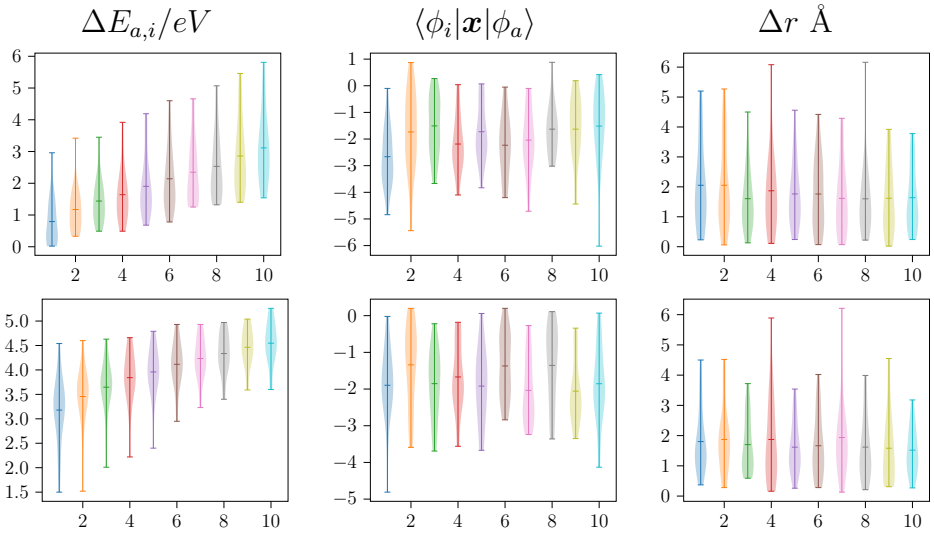

Figura 6.18: Distribución de las propiedades obtenidas a través de cálculos de bajo costo computacional. Primera fila OCT-LCAO y segunda fila GS-LDA. Por simplicidad se muestran solo los 10 primeros estados. A los momentos dipolares se les aplicó la función *log* para mejorar la visibilidad en la distribución

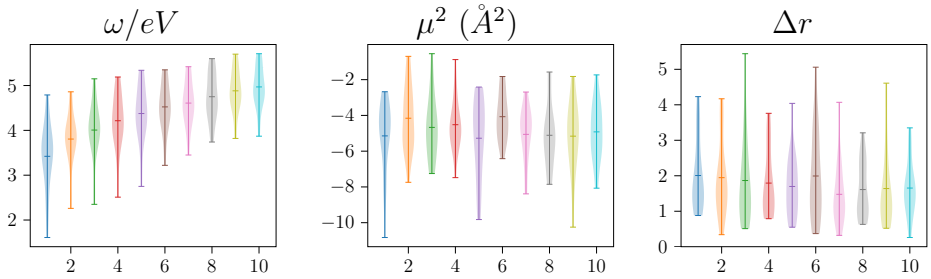

Figura 6.19: Distribución de las propiedades obtenidas a través de la solución de la ecuación de Casida utilizando el funcional PBE0. Se muestran tanto los valores extremos como la media y en que rango existe la mayor densidad de valores. Los Momentos de transición se le aplicó la función *log* para mejorar la visibilidad en la distribución.

6.6 Conclusiones del Capítulo.

Al finalizar este capítulo se pueden resaltar los avances y deficiencias encontradas durante la realización de las pruebas. Las pruebas realizadas utilizando KRR demostraron que al agregar datos electrónicos a un descriptor estructural mejora los resultados en la predicción de propiedades espectroscópicas.

Las Redes Neuronales mejoraron los resultados obtenidos por KRR usando únicamente descriptores electrónicos. Los modelos en donde las propiedades son

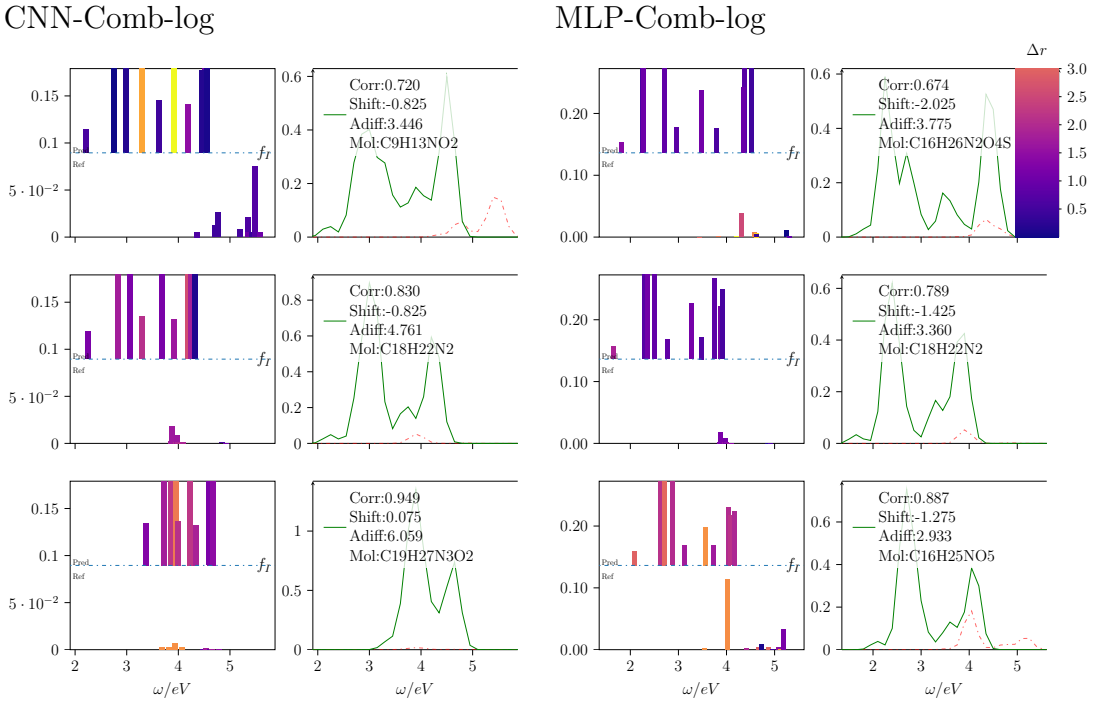

Figura 6.20: Reconstrucción del espectro de absorción para algunas de las moléculas pertenecientes a la ZINC-DB. Desde arriba el Peor, el Promedio y el Mejor resultado. Se muestran un reconstrucción el espectro discreto primera y tercera columna y el continuo segunda y cuarta columna

tratadas como pares, funciona mejor que los modelos donde se tratan de manera independiente.

Pero algunas propiedades siguen siendo elusivas debido su compleja naturaleza. La distribución de los momentos dipolares de las monoexcitaciones y momentos de transición μ^2, incluso después de realizado el escalamiento y transformaciones como preprocesamiento de datos, arrojaba una curva clara donde la mayor densidad de valores se concentraba cerca del 0. Esto unido a la poca variabilidad en los valores dificulta obtener un modelo eficiente.

El uso de datos de calidad resulta crítico. Esa puede ser una de las razones por la que el salto GS-LDA a CASIDA-PBE0 ofrece mejores resultados. Por otra parte los obtenidos de OCT-LCAO siendo una aproximación inicial, contienen ruido el cual es depurado por SCF de DFT (ver Sección 2.1) después de varias iteraciones. Las Optimizaciones Bayesianas (ver Sección 6.3.3) realizadas demostraron lo complicado del salto OCT-LCAO a CASIDA-PBE0 al encontrar que son necesarias casi el doble de capas ocultas para obtener resultados comparables.

El descriptor Δr utilizado para evaluar el carácter de transferencia de carga

de la transición ayudó a identificar correctamente los estados intercambiados por el uso del funcional híbrido PBE0. Hay que destacar que para moléculas centrosimétricas el descriptor es 0 en este caso no aporta información a la Red Neuronal. La 8CONF contiene 85 moléculas centrosimétricas que representan el 0.4% de la base de aprendizaje.

La prueba realizada sobre la base de datos ZINC-DB mostró qué tan dependiente de los datos entrenamientos son estos modelos y la carencia de una representación molecular (descriptor) genérica.

Capítulo 7

Conclusiones.

Contents

7.1 Conclusiones principales.

Como conclusión se puede hacer un resumen de los mecanismos propuestos y los principales resultados alcanzados en la investigación. Al inicio de este trabajo se propuso evaluar un grupo de propiedades moleculares extraídas de cálculos TDDFT computacionalmente baratos.

TDDFT es computacionalmente más costos en la medida que el funcional de la energía de intercambio y correlación asciende la *Escalera de Jacob* hacia la precisión química'. La carencia de una expresión universal para este funcional abre una linea de investigación con múltiples soluciones descritas en la literatura.

La propuesta descrita en este trabajo defiende la utilización de ML como mecanismo para predecir los cálculos realizados utilizando TDDFT. Los resultados descritos en los Capítulos 5 y 6 avalan el uso de ML y el cumplimiento del objetivo propuesto de evaluar múltiples propiedades moleculares como posibles descriptores moleculares. Sin embargo las aplicaciones de ML en cálculos *ab-initio* son un tema reciente y esta investigación encontró dificultades que abren nuevas posibilidades de continuidad. Las conclusiones particulares se irán proponiendo en el mismo orden que se presentaron los resultados en esta tesis.

7.1.1 Predicción de Geometrías.

En la predicción de geometrías se buscaba encontrar un descriptor que tuviera como finalidad simplificar los mecanismos existentes: (1) Construcción de la superficie de energía potencial, mediante cálculos semi-empíricos o ab-initio y (2) la navegación de la superficie de energía potencial.

Teniendo esto como premisa, la selección de un descriptor que pudiera representar la geometría interna de una molécula, que no requiriera de coordenadas de ningún tipo (cartesianas, internas, polares, etc.) y que fuera simple de calcular, resultaba importante. Este descriptor se encontró en el *e-state*, simple de calcular, solo dependiente de átomos, tipos de enlaces y contribuciones vecinas. Sobre este descriptor se construyó el concepto de *bloque*, el cual representa un átomo y su entorno inmediato.

Como método de aprendizaje se seleccionó la Regresión Ridge con Kernel(KRR), por su simplicidad de implementación. El cual fue optimizado para obtener los hiperparámetros adecuados con los que el método adquirió su máxima eficiencia para este problema. KRR demostró ser la decisión correcta avalado con los resultados de RMSD (> 0.05 Å) obtenidos en la predicción de bloques.

El modelo de bloques adoptados resulta fácilmente ampliable con la incorporación de nuevas estructuras a la base de datos.

Admite el manejo de distintas bases de datos de moléculas por separado, obtenidas bajo diferentes métodos de optimización.

Como cada nueva propuesta no está exenta de dificultades y el ensamblado de bloques resultó particularmente difícil. Dos factores influyeron: (1) el descriptor utilizado contiene información sobre bloques vecinos pero no es suficiente como para definir el ángulo de torsión entre bloques, (2) el método KRR no es capaz de producir ángulos de torsión correctos debido al fallo del descriptor en este punto. Sin embargo el método TTKDG basado en el uso de ángulos de torsión obtenidos de cálculos *ab-initio* demostró obtener conformaciones similares a las logradas utilizando el ángulo de torsión original.

7.1.2 Predicción de propiedades espectroscópicas.

En la predicción de propiedades espectroscópicas se buscaba utilizar propiedades electrónicas provenientes de cálculos computacionalmente poco costosos.

Se utilizaron dos métodos de aprendizaje automático KRR y NN. Para ambos se demostró que el uso de propiedades electrónicas mejora los resultados en la predicción de propiedades espectroscópicas. A partir de los experimentos realizados se llegó a la conclusión que los modelos basados en KRR funcionan mejor si existe una correlación explicita entre *descriptor - propiedad objetivo*. Por otra parte NN se benefician de esta correlación, pero también es capaz de encontrar

correlación en propiedades no tan cercanas. Aunque hay que destacar que la información conceptual implícita dentro de los descriptores (ej. densidad electrónica a partir de la estructura molecular, número atómico, etc.) no siempre es alcanzable. El KRR fue muy deficiente en este sentido y las NN obtuvieron mejores resultados pero requirieron topologías más complicadas.

Las complejas transformaciones realizadas por las ecuaciones de Kohn -Sham, incluidas la dependiente del tiempo donde se aplica entre otras la *Transformada de Fourier*, opacan el camino entre *estructura molecular - propiedad electrónica*. Este punto resulta vital cuando se decide construir un modelo de aprendizaje automático que sustituya el esquema de cálculos *ab-initio*. Razón por la cual agregar datos ayuda a mejorar las predicciones (ej. propiedades electrónicas).

Un tipo de ayuda sería conocer que propiedades requieren más datos que otras. En los estudios realizados se encontró que las energías de excitación no requerían de tantos puntos de aprendizaje como los momentos de transición. Puede deberse a que el descriptor escogido ($\Delta E_{a,i} = \epsilon_a - \epsilon_i$) para obtener las energías de excitación tiene mejor correlación que el escogido ($\langle \phi_i | \boldsymbol{x} | \phi_a \rangle$) para los momentos de transición. Lo mismo ocurre con el índice de transferencia de carga Δr. Conocer este tipo de problemas ayuda a optimizar el modelo y en nuestro caso evidenció que aún hay espacio para realizar mejoras.

Las NN son mucho más flexibles por consiguiente la cantidad de hiperparámetros hace complicada su utilización y optimizarlas se convierte en una tarea difícil. En este punto acotar lo más posible los grados de libertad de los hiperparámetros a través de pruebas sencillas ayuda a conocer cuales de estos parámetros tienen mayor influencia en el problema que se intenta representar. Durante las pruebas realizadas se identificaron el *número de capaz ocultas* y *iteraciones de aprendizaje (epoch)* como los más influyentes. Para estos hiperparámetros se realizaron optimizaciones Bayesianas con las que se obtuvieron las topologías más eficientes para enfrentar las predicción de propiedades espectroscópicas.

7.2 Trabajo Futuro.

Se identificaron varios temas que abren nueva preguntas e investigaciones.

Geometrías.

- El algoritmo de ensamblaje de bloques requiere algo más de trabajo, principalmente para encontrar un buen descriptor para los ángulos de torsión que sea capaz de captar simetrías.

- El uso de NN para la estimación de geometrías fue probado solo superficialmente. Faltaría un estudio mas profundo de sus posibilidades.

Espectro de absorción.

- Establecer un gradiente de peso para cada uno de los descriptores. Le daría prioridad a unos sobre otro, p.ej. restarle peso a las energías de excitación porque requieren menos puntos de aprendizaje para producir buenas predicciones.

- El potencial de las SVM y NN, hace pensar que serían un buen candidato para construir un funcional para energía de intercambio y correlación E_{xc}. Varias familias de funcionales están construidos sobre parámetros de ajustes realizados sobre resultados experimentales. SVM y NN son los métodos de ajustes por excelencia capaces de reajustarse mediante aprendizaje [113, 114].

Índice de Figuras

Índice de Tablas

Appendix A

Anexos

A.1 Breve descripción de los componentes de la solución de software.

En las siguientes secciones se dará una pequeña descripción de los *scripts* de *Python3* implementados. Cada uno contribuye a la solución propuesta desde los modelos de KRR y NN, manejo de datos, calculo de errores y reconstrucción de valores.

Estos *scripts* requieren para su correcto funcionamiento que los siguientes paquetes estén debidamente instalados en la versión de Python3 a utilizar:

- *ase* versión mayor o igual a 3.15.0.
- *Keras* versión mayor o igual a 2.2.2.
- *Keras-Applications* versión mayor o igual a 1.0.4.
- *Keras-Preprocessing* versión mayor o igual a 1.0.2.
- *numpy* versión mayor o igual a 1.13.3.
- *matplotlib* version mayor o igual a 3.0.0.
- *sklearn* versión mayor o igual a 0.0.
- *scipy* versión mayor o igual a 0.19.1.
- *scikit-learn* versión mayor o igual a 0.18.0.
- *rdkit* versión mayor o igual a 2018.09.1.
- *tensorboard* versión mayor o igual a 1.10.0.

- *tensorflow* versión mayor o igual a 1.10.1.

A.1.1 mlchem/mlmol

Este paquete contiene un conjunto de clases y funciones que se consideran básicas. No tienen dependencias entre ellas y ofrecen funcionalidades a las capas superiores. La reutilización de estos componentes es bien alta dado que muchos solo realizan funciones puntuales.

construct_data.py

Este módulo contiene la implementación de la clase *Oct_Data*, encargada de manejar los datas extraídos de los cálculo realizados. Lo datos fueron divididos por propiedades en archivos compactados mediante el algoritmo *GZ* (ej. oct-occ.lda.json.gz).

Este mecanismo permite cargar solo que es necesario y ahorra uso de RAM. Dentro de cada archivo *.gz* hay un solo archivo con formato JSON que contiene los datos de esa propiedad.

En este módulo también están las implementaciones de la CM y SCM. Así como también otras funciones que construyen otros datos a partir de las propiedades electrónicas obtenidas con Octopus (ej. la densidad de estados, el *HOMO-LUMO gap*).

get_dihedral.py

Este módulo es una modificación de algunas funciones incluidas en ASE para trabajar con los ángulos de torsión.

get_estate.py

Este archivo contiene la implementación propia del descriptor *e-state*. Debido a algunos problemas encontrados en la incluida en RdKit, como por ejemplo la imposibilidad de separar este valor en todas las contribuciones individuales. Se decidió realizar una implementación propia.

help_data_functions.py

Contiene funcionalidades variadas que ayudan a crear o procesar datos.

krr.py

Implementa las clases *krr_solver*, *SPBlockPredKrr* y *SPEnvPredKrr* para la solución del KRR. La primera *krr_solver* es utilizadas para la predicción de las energías y

por las otras dos clases son utilizadas en la predicción de productos escalares y bloques.

Este módulo depende de *norms.py* donde están implementadas las funciones para el kernel y las normas para determinar las similitudes/diferencias entre descriptores.

nn_model.py

Contiene las implementaciones de los modelos de Redes Neuronales utilizados en las pruebas. Si se desea modificar alguna topología o incluir una nueva, este es el módulo en donde se puede incluir.

norms.py

Implementaciones muy puntuales de métricas. Incluye la normas l_1 utilizada para el cálculo del kernel del método KRR y varias métricas relacionadas con el error, RMSE, MAE, RMSR normalizado y el coeficiente r^2 de correlación.

A.1.2 mlchem

Este paquete es una pequeña abstracción de lo que sería la lógica implementadas. Los elementos en este paquete dependen de los contenidos en *mlchem/mlmol* y tienen dependencia entre ellos. La reutilización de varios de los componentes en este paquete se reduce mucho porque están muy unidos a la solución de los problemas que se enfrentaron.

arg_parser.py

Este script contiene las implementaciones del paquete de *Python argparse* encargado de interpretar los parámetros con los cuales se lanzan las pruebas, desde la linea de comandos. La mayor parte de las opciones ya tienen valores por defecto, configurados a partir de las pruebas realizadas. Los parámetros están divididos en tres grupos:

1. *Comunes*:
    - -nl: Número de moléculas a utilizar en el entrenamiento. Son escogidas aleatoriamente.
    - -nt: Número de moléculas a utilizar como grupo de control. Son escogidas aleatoriamente.
    - -mol_dir: Directorio donde se alojan los archivos *.mol* que describen la geometría de las moléculas.

- -out_dir: Directorio donde se copiarán los resultados de las pruebas a realizar.
- -desc_xc: Funcional a utilizar por las propiedades descriptoras. La opciones son *lda* y *lcao*.
- -targ_xc: Funcional a utilizar por las propiedades descriptoras. La opciones son *lda* y *pbe0*.
- -oct_data_dir: Directorio donde se encuentran los datos obtenidos de los cálculos TDDFT realizados con Octopus.

2. *Regresión con Kernel*:

   - -l_value: Valor del hiperparámetro de regularización. Se espera un número que va se considerado como de punto flotante.
   - -sigma: Valor del hiperparámetro de la función de kernel. Se espera un valoro varios dependiendo de la cantidad de descriptores a utilizar. Si la función de kernel escogida no lo requiere es ignorado.
   - -state_cal: Cantidad de estados a predecir. Ya que los modelos de KRR se construyen por estados este valor indica la cantidad de modelos a construir. Se espera un entero entre $[0, 10]$.
   - -kernel: Función para construir el kernel. Por el momento solo está extensamente probada 'dist_k_exp', aunque hay otras disponibles.
   - -test_function: Especifica la prueba a realizar. La lista de posibles funciones se muestra si se lanza el script *test_krr_spectra.py* sin ningún parámetro.

3. *NN*:

   - -test_function: Especifica la prueba a realizar. La lista de posibles funciones se muestra si se lanza el script *test_nn_spectra.py* sin ningún parámetro.

mol.py

Es este archivo se encuentran las implementaciones de las clases:

- *Mol*: Encargada del manejo de toda la información correspondiente a una molécula. Contiene métodos para mostrar las propiedades electrónicas obtenidas de los cálculos TDDFT, así como también el manejo de los bloques, construcción de descriptores, ángulos dihedros y otros datos que se pueden extraer de una molécula.

- *MolData*: Maneja listas de la clase *Mol*. Construye bloques de datos de las moléculas contenidas en la lista.

- *BlockAssembler*: Clase de apoyo para ensamblar bloques después de realizada la predicción de productos escalares.

ml_nn.py

Este archivo contiene las funciones que construyen los métodos de NN. Las funciones implementadas mantienen una homogeneidad en los parámetros que necesitan, dos objetos de tipo MolData uno utilizado para aprender y otro utilizado como grupo de control y evaluar el método ('learn' y 'test'), funcional de la energía E_{xc} de donde obtener los descriptores ('desc_xc', pueden ser 'lda' o 'lcao'), funcional de la energía E_{xc} de donde obtener las propiedades objetivos ('targ_xc', pueden ser 'lda' o 'pbe0'). Permite, con limitantes, modificar la topología de la red admitiendo dos hiperparámetros, número de capas ocultas ('n_hl') y número de iteraciones ('epochs'), cantidad de estados a predecir ('sta') actualmente fijado en 10 y el directorio donde escribir los resultados ('tbn_outdir').

Cada función retorna un objeto diccionario de python a través del cual se puede acceder a los datos y las predicciones para realizar los cálculos de error y análisis estadístico de los resultados. El modelo entrenado se encuentra bajo la llave 'model' en el diccionario. La llave 'model' es un objeto 'tupla' de python donde la posición 0 es el modelo de keras/tensorflow entrenado y la posición 1 es el objeto 'history' de keras[1]. Bajo la llave 'pred' se encuentran los resultados, se mantuvo en todo momento el mismo orden en la que se construyen los descriptores $\mu^2, \Delta e_{a,i}, \Delta r$. Alguna de las funciones implementadas no usan[2] el descriptor Δr por tanto la dimensión del objeto debajo de la llave 'pred' varía en consecuencia.

En este archivo también se encuentra la función *preparen_scalen_data* que se encarga de escalar todos los datos, descriptores y propiedades objetivos antes de entrenar el modelo y retorna el objeto *sklearn.preprocessing.MinMaxScaler* para revertir esto después de realizada la predicción. Como cada propiedad tiene su propia distribución esta función retorna una tupla con el siguiente orden:

- objeto de numpy con los descriptores correspondientes los momentos de transición.

- objeto de numpy con los descriptores correspondientes las energías.

- objeto de numpy con los descriptores correspondientes los Δr.

[1]En la web https://keras.io/ se puede encontrar una extensa documentación de como funciona y que contiene este objeto.

[2]Las que si lo usan contienen 'ct' o 'class' en su nombre.

- objeto de numpy con las referencias correspondientes los momentos de transición.
- objeto de numpy con las referencias correspondientes las energías.
- objeto de numpy con las referencias correspondientes los Δr.
- objeto de sklearn.preprocessing.MinMaxScaler de los descriptores correspondientes a los momentos de transición.
- objeto de sklearn.preprocessing.MinMaxScaler de los descriptores correspondientes a las energías
- objeto de sklearn.preprocessing.MinMaxScaler de los descriptores correspondientes a los Δr.
- objeto de sklearn.preprocessing.MinMaxScaler de las referencias correspondientes a los momentos de transición.
- objeto de sklearn.preprocessing.MinMaxScaler de las referencias correspondientes a las energías
- objeto de sklearn.preprocessing.MinMaxScaler de los referencias correspondientes a los Δr.

A.1.3 Scripts de pruebas realizadas

Los siguientes módulos hacen un uso extensivo de los descritos en los Anexos A.1.2 y A.1.1.

test_krr_env.py

Contiene la implementaciones de todas las pruebas realizadas utilizando KRR para la optimización de las geometrías moleculares. Este script admite parámetros para poder modificar la manera en la que se conduce la prueba.

test_krr_spectra.py

Contiene la implementaciones de todas las pruebas realizadas utilizando KRR para la predicción de propiedades espectroscópicas. Este script admite parámetros para poder modificar la manera en la que se conduce la prueba.

test_nn_spectra.py

Contiene la implementaciones de todas las pruebas realizadas utilizando KRR para la optimización de las geometrías moleculares. Este script admite parámetros para poder modificar la manera en la que se conduce la prueba.

test_zinc_db.py

Contiene la implementaciones de todas las pruebas realizadas utilizando KRR para la optimización de las geometrías moleculares. Este script admite parámetros para poder modificar la manera en la que se conduce la prueba.

test_db_stats.py

Contiene la implementaciones de todas las pruebas realizadas utilizando KRR para la optimización de las geometrías moleculares. Este script admite parámetros para poder modificar la manera en la que se conduce la prueba.

A.2 Tabla cantidad de tipos de bloques dentro de la 8CONF

Tabla A.1: Cantidad de bloques por tipo.

Tipo de Bloque	Conteo
C-C-H-H-H	1652
C-C-C-H-H	1389
O-C-C	700
O-C-H	686
C-O-C-H-H	606
C-O-C-C-H	503
C-C-C-C-H	427
C-C-C	399
N-C-C-H	374
C-C-C-H	326
C-O-C-H	325
C-O-N-C	319
C-N-C-H	302
C-N-C-H-H	291
N-C-H-H	272
C-N-C	271
C-O-C-C	249
N-C-C	234
C-O-C-C-C	234
C-C-H	229
C-N-H-H-H	169
C-N-C-C-H	159
C-O-N-H	145
C-C-C-C-C	144
C-O-H-H-H	142
C-N-C-C	138
N-C-C-C	132
N-C-H	123
C-N-N-H	100
C-N-N-C	93
C-O-O-C	88
N-N-C	74
C-O-N-N	71
N-O-C	69
C-N-C-C-C	62
C-C-C-C	49
C-O-O-N	44
O-N-H	43
N-N-N	27
N-N-C-H	27
C-O-O-H	27
O-N-C	25
N-O-N	15
C-N-N-N	11
C-O-O-O	9
O-N-N	8
N-N-N-H	7
C-O-O-H-H	7
C-F-F-F-C	6
C-O-O-C-H	4
N-N-C-C	4
N-N-N-C	3
N-C-H-H-H	2
C-N-H	1
C-H-H-H-H	1
C-F-F-F-F	1
C-O-H-H	1
C-O-O-C-C	1
O-H-H	1
N-H-H-H	1

A.3 Tabla resultados de los RMSD por bloque

Tabla A.2: Media del RMSD por bloques y desviación Estándar del RMSD.

Tipo de Bloque	Media RMSD	Desviación Estándar	Número de Bloques de Aprendizaje
C-C-H-H-H.Td	0.005 38	0.000 13	2013
C-C-C-H-H.Td	0.009 49	0.000 71	2136
C-O-H-H-H.Td	0.015 24	0.000 82	128
O-C-C.Td	0.167 02	0.004 00	238
C-C-C-C-H.Td	0.041 91	0.004 03	301
C-N-C-C.Tr	0.104 93	0.003 44	220
N-O-C.Tr	0.010 05	0.000 51	208
O-N-H.Tr	0.009 83	0.000 45	101
C-N-C-C-H.Td	0.060 25	0.005 11	595
C-N-C-H-H.Td	0.012 84	0.000 38	746
N-C-C-C.Td	0.090 07	0.010 33	76
C-N-H-H-H.Td	0.018 06	0.000 41	380
N-C-C-H.Td	0.081 48	0.010 50	252
C-N-C.L	0.000 95	0.000 27	540
C-O-C-H.Tr	0.013 27	0.000 73	357
O-C-H.Td	0.011 75	0.000 56	823
C-O-C-H-H.Td	0.023 40	0.002 11	1363
C-O-C-C-H.Td	0.016 17	0.000 73	914
C-O-N-H.Tr	0.017 95	0.000 69	334
N-C-C-C.Tr	0.313 00	0.004 84	223
C-C-C-H.Tr	0.079 63	0.001 22	478
O-N-C.Tr	0.072 64	0.006 62	11
N-N-C.Tr	0.077 16	0.001 20	207
N-N-N.Tr	0.134 01	0.003 70	15
N-O-N.Tr	0.150 25	0.009 94	8
N-C-H-H.Tr	0.013 42	0.000 55	684
C-N-N-C.Tr	0.046 54	0.001 97	425
O-C-C.Tr	0.004 75	0.001 37	1056
N-C-C.Tr	0.057 49	0.000 63	156
C-N-C-H.Tr	0.083 79	0.000 65	400
C-C-C-C-C.Td	0.248 54	0.005 34	156
N-C-C-H.Tr	0.028 36	0.001 88	897
N-C-H.Tr	0.013 01	0.000 34	262
C-O-N-N.Tr	0.012 21	0.000 83	244
N-C-H-H-H.Td	0.238 00	0.013 99	3
C-O-O-C.Tr	0.118 04	0.002 09	168
C-O-C-C-C.Td	0.085 35	0.001 90	110
C-N-C-C-C.Td	0.307 40	0.007 06	48
C-O-C-C.Tr	0.020 74	0.002 95	319
C-C-C.L	0.017 87	0.001 10	510
C-C-H.L	0.002 34	$9.543\,62 \cdot 10^{-05}$	275
C-O-O-H.Tr	0.088 15	0.005 58	48
C-N-N-H.Tr	0.124 81	0.003 01	42
O-C-H.Tr	0.005 44	0.000 38	108
C-O-N-C.Tr	0.020 23	0.001 38	928
N-N-C-H.Tr	0.101 88	0.002 63	90
N-N-N-H.Tr	0.321 75	0.011 95	2
C-F-F-F-C.Td	0.128 27	0.010 24	9
C-C-C-C.Tr	0.162 35	0.006 59	16
N-C-H-H.Td	0.015 15	0.002 32	33
C-O-O-N.Tr	0.008 85	0.000 66	128
C-O-O-H-H.Td	0.138 50	0.008 67	9
C-N-N-N.Tr	0.121 27	0.007 57	26

A.4 Resultados de los métodos de emsamblado.

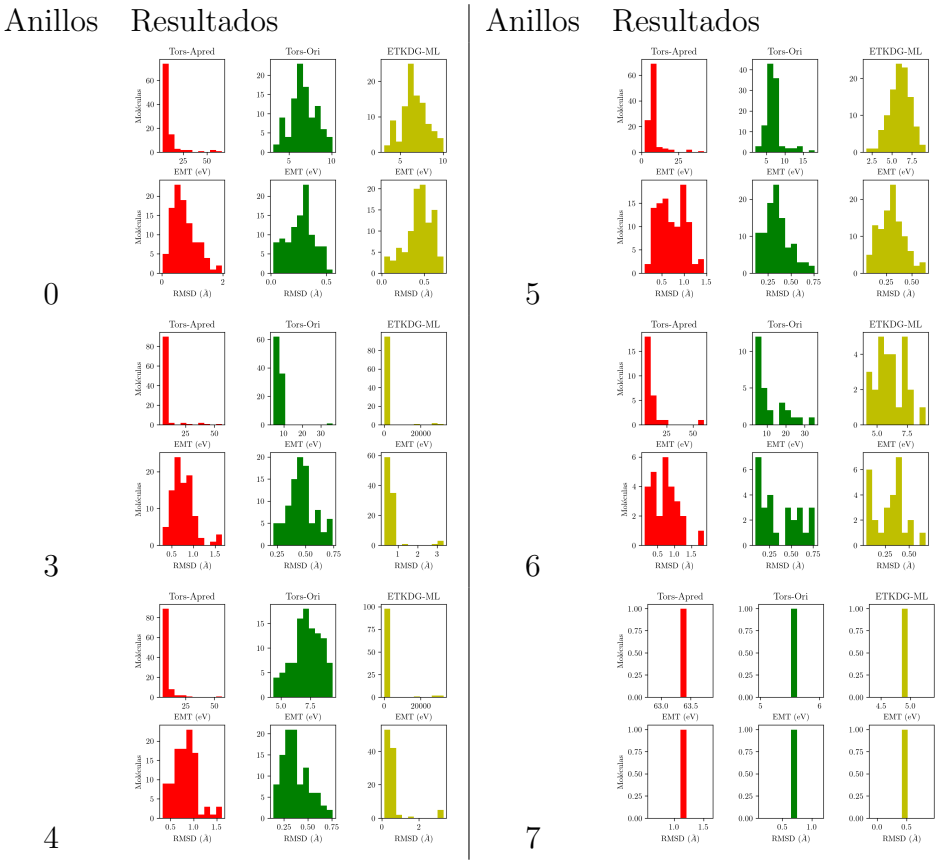

Figura A.1: Histograma de resultados de valores de EMT y RMSD para las moléculas reconstruidas. Por átomos en anillos.

A.5 Parámetros utilizados para los cálculos en Octopus

Listing A.1: Archivo de entrada para el cálculo LCAO en Octopus.

```
CalculationMode = gs
%RestartOptions
 restart_gs | 'restart' | restart_mix + restart_rho
%

XYZCoordinates = 'geom.xyz'

BoxShape = minimum
Spacing = 0.20*angstrom
Radius = 6.*angstrom

SymmetriesCompute = false

XCFunctional = lda_x + lda_c_pz_mod
%Species
 "C"  | SPECIES_PSEUDO | file | "C-gga.psf"  | lmax | 1 | lloc | 0
 "O"  | SPECIES_PSEUDO | file | "O-gga.psf"  | lmax | 1 | lloc | 0
 "N"  | SPECIES_PSEUDO | file | "N-gga.psf"  | lmax | 1 | lloc | 0
 "F"  | SPECIES_PSEUDO | file | "F-gga.psf"  | lmax | 1 | lloc | 0
 "H"  | SPECIES_PSEUDO | file | "H-gga.psf"  | lmax | 0 | lloc | 0
 "P"  | SPECIES_PSEUDO | file | "P-gga.psf"  | lmax | 1 | lloc | 0
 "S"  | SPECIES_PSEUDO | file | "S-gga.psf"  | lmax | 1 | lloc | 0
 "Si" | SPECIES_PSEUDO | file | "Si-gga.psf" | lmax | 1 | lloc | 0
 "Cl" | SPECIES_PSEUDO | file | "Cl-gga.psf" | lmax | 1 | lloc | 0
 "I"  | SPECIES_PSEUDO | file | "I-gga.psf"  | lmax | 1 | lloc | 0
%

Eigensolver = rmmdiis
EigenSolverTolerance = 1e-7
ConvEigenError = yes

ExperimentalFeatures=true

UnitsOutput = ev_angstrom

CasidaPrintExcitations = '1-50'
CasidaWeightThreshold = 0.01

Output = matrix_elements
OutputMatrixElements = dipole
OutputMEMultipoles = 1
```

Listing A.2: Archivo de entrada para el cálculo LDA en Octopus.

```
CalculationMode = gs
%RestartOptions
 restart_gs | 'restart' | restart_mix + restart_rho
%

XYZCoordinates = 'geom.xyz'

BoxShape = minimum
Spacing = 0.20*angstrom
Radius = 6.*angstrom

SymmetriesCompute = false

XCFunctional = lda_x + lda_c_pz_mod
%Species
 "C"  | SPECIES_PSEUDO | file | "C-gga.psf"  | lmax | 1 | lloc | 0
 "O"  | SPECIES_PSEUDO | file | "O-gga.psf"  | lmax | 1 | lloc | 0
 "N"  | SPECIES_PSEUDO | file | "N-gga.psf"  | lmax | 1 | lloc | 0
 "F"  | SPECIES_PSEUDO | file | "F-gga.psf"  | lmax | 1 | lloc | 0
 "H"  | SPECIES_PSEUDO | file | "H-gga.psf"  | lmax | 0 | lloc | 0
 "P"  | SPECIES_PSEUDO | file | "P-gga.psf"  | lmax | 1 | lloc | 0
 "S"  | SPECIES_PSEUDO | file | "S-gga.psf"  | lmax | 1 | lloc | 0
 "Si" | SPECIES_PSEUDO | file | "Si-gga.psf" | lmax | 1 | lloc | 0
 "Cl" | SPECIES_PSEUDO | file | "Cl-gga.psf" | lmax | 1 | lloc | 0
 "I"  | SPECIES_PSEUDO | file | "I-gga.psf"  | lmax | 1 | lloc | 0
%

Eigensolver = rmmdiis
EigenSolverTolerance = 1e-7
ConvEigenError = yes

ExperimentalFeatures=true

UnitsOutput = ev_angstrom

CasidaPrintExcitations = '1-50'
CasidaWeightThreshold = 0.01

Output = matrix_elements
OutputMatrixElements = dipole
OutputMEMultipoles = 1
```

Listing A.3: Archivo de entrada para el cálculo PBE0 en Octopus.

```
CalculationMode = gs
%RestartOptions
 restart_gs | 'restart' | restart_mix + restart_rho
%

XYZCoordinates = 'geom.xyz'

BoxShape = minimum
Spacing = 0.20*angstrom
Radius = 6.*angstrom

SymmetriesCompute = false

XCFunctional = hyb_gga_xc_pbeh
TheoryLevel = dft

%Species
 "C"  | SPECIES_PSEUDO | file | "C-gga.psf"  | lmax | 1 | lloc | 0
 "O"  | SPECIES_PSEUDO | file | "O-gga.psf"  | lmax | 1 | lloc | 0
 "N"  | SPECIES_PSEUDO | file | "N-gga.psf"  | lmax | 1 | lloc | 0
 "F"  | SPECIES_PSEUDO | file | "F-gga.psf"  | lmax | 1 | lloc | 0
 "H"  | SPECIES_PSEUDO | file | "H-gga.psf"  | lmax | 0 | lloc | 0
 "P"  | SPECIES_PSEUDO | file | "P-gga.psf"  | lmax | 1 | lloc | 0
 "S"  | SPECIES_PSEUDO | file | "S-gga.psf"  | lmax | 1 | lloc | 0
 "Si" | SPECIES_PSEUDO | file | "Si-gga.psf" | lmax | 1 | lloc | 0
 "Cl" | SPECIES_PSEUDO | file | "Cl-gga.psf" | lmax | 1 | lloc | 0
 "I"  | SPECIES_PSEUDO | file | "I-gga.psf"  | lmax | 1 | lloc | 0
%

Eigensolver = rmmdiis
EigenSolverTolerance = 1e-7
ConvEigenError = yes

ExperimentalFeatures=true

UnitsOutput = ev_angstrom

CasidaPrintExcitations = '1-50'
CasidaWeightThreshold = 0.01
CasidaKSEnergyWindow = 11eV

Output = matrix_elements
OutputMatrixElements = dipole
OutputMEMultipoles = 1
```

A.6 Notebook generado para NoMad

Como parte del trabajo realizado para *NoMad*, en la sección de *Annalitic Tools* (*Herramientas Análiticas*), enfocadas en procesar el volumen de datos almacenados en la base de datos de NoMad, se desarrollo un *Notebook* de *Jupiter*[3]. Esta tarea tuvo como objetivo la comparación entre el uso de un descriptor geométrico como la CM y agregar datos electrónicos para mejorar la predicción.

El método de aprendizaje automático utilizada fue la Regresió Ridge con Kernel descrito en este trabajo. La base de datos utilizada fue la descrita en este trabajo, aunque solo se emplearon los datos de GS-LDA y como propiedad objetivo fueron las energías de excitación del cálculo CASIDA-LDA.

Desde el punto de vista Químico/Físico, no se lograba mucho haciendo estas predicciones. Tampoco representaba un ahorro sustancial en capacidad de computo. Pero ayudó a sentar las bases de este trabajo, comprender el funcionamiento de los algoritmos de aprendizaje automático, explorar el comportamiento de los hiperparámetros y su influencia sobre el algoritmo. Además se evaluó, como prueba de concepto perdecir un arbitrario número de estados y la utilización de descriptores electrónicos.

[3]El lector puede visitar https://jupyter.org/try como introducción a estas tecnologías.

A.7 Superficies de optimizaciones de hiper-parámetros. Regresión Ridge con Kernel

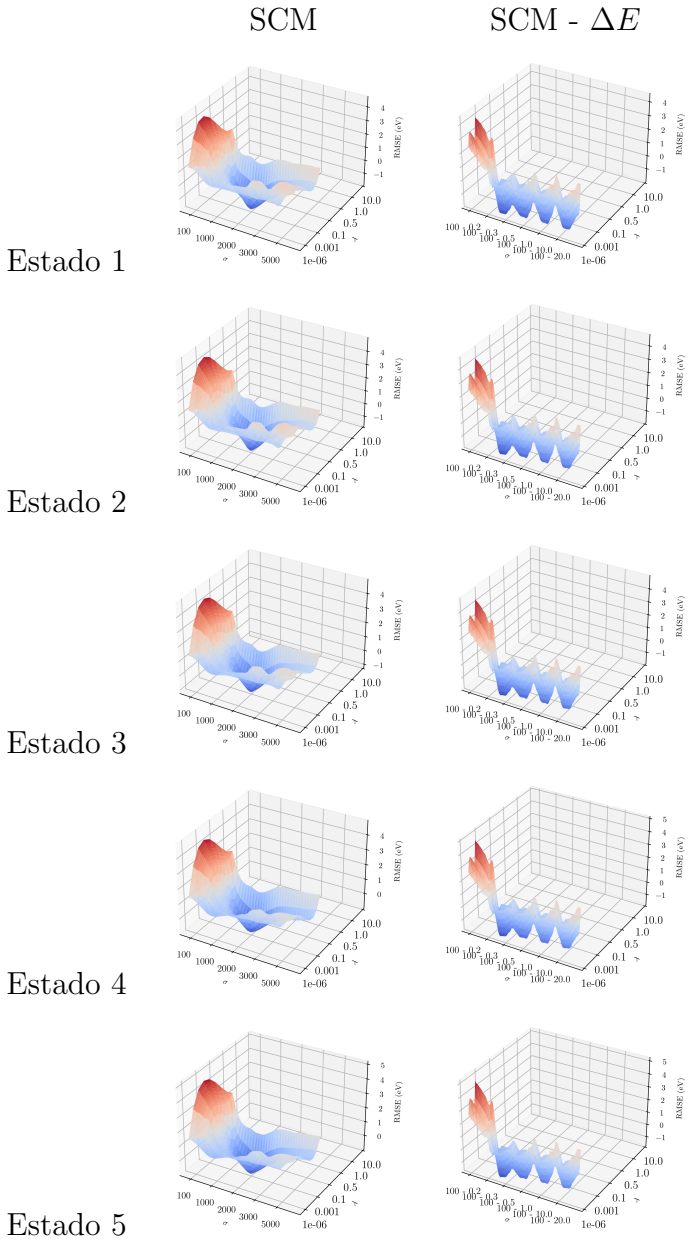

Figura A.2: Superficies obtenidas de la optimización de los hiperparámetros σ y λ pertenecientes al método KRR. Las superficies presentan variaciones muy pequeñas entre estados.

A.8 Superficies optimización de hiperparámetros para las redes neuronales

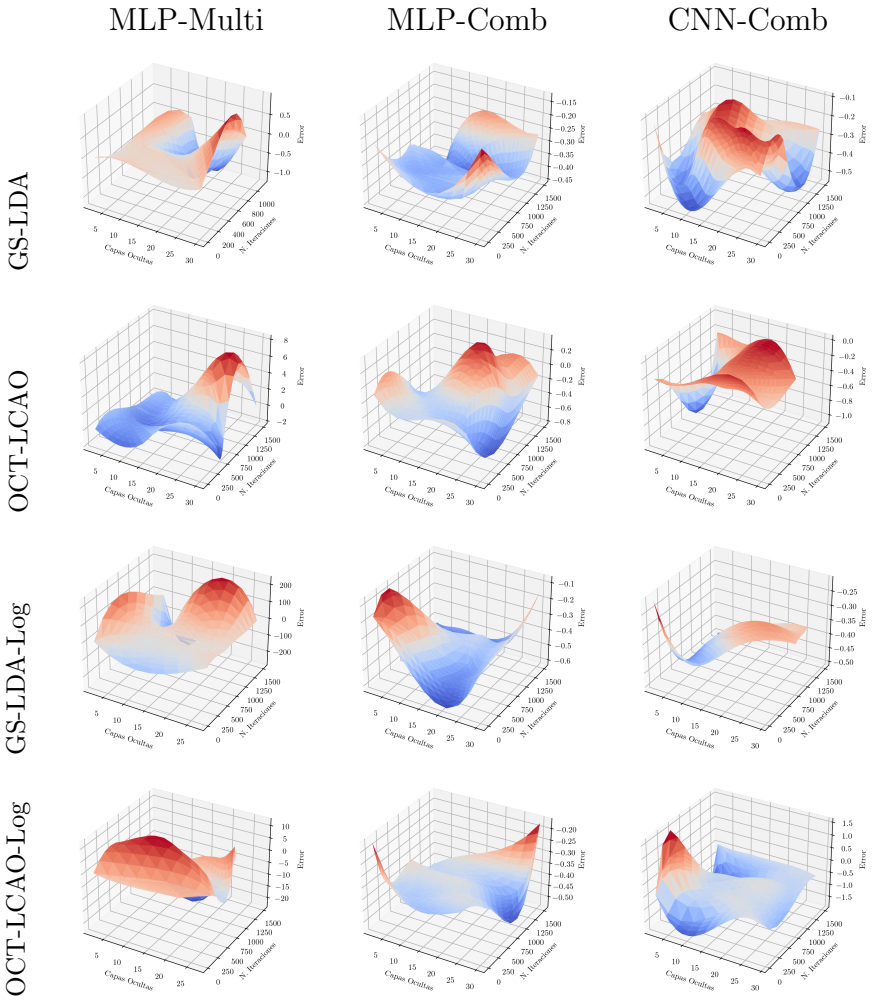

Figura A.3: Superficies del espacio de hiper-parámetros, para las 3 topologías de redes evaluadas obtenidas durante las optimizaciones bayesianas.

A.9 Imágenes relacionadas con la predicción del Espectro de Absorción.

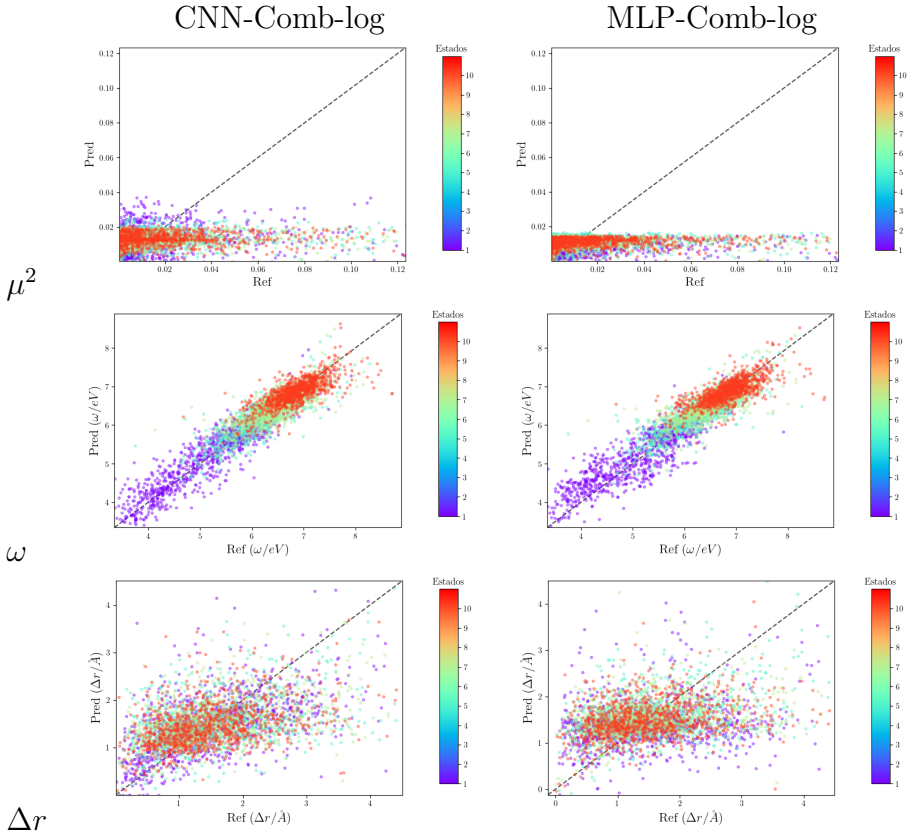

Figura A.4: Comparación entre las referencias y las predicicones por estados para las prediciones realizadas utilizando los modelos que ofrecienron los mejores resultados en el salto desde OCT-LCAO a CASIDA-PBE0.

A.10 Tabla de parámetros disponibles en Kera y Tensorflow.

Tabla A.3: Lista de hiperparámetros accesibles desde *Keras*.

Hiperparámetro	**Descripción**
Función de Activación	Defines if a neuron will have an output. We alternate an activation layer first Elu [75] and then ReLU [77].
Neuronas por Capa	Cantidad de neuronas en una capa. Este parámetro es utilizado en Keras como dimensión de salida de la capa. En el MLP este parámetro se denomina *Units*, para las CNN se denomina *Filter* representa la salida después de aplicada la convolución.
Número de Capas Ocultas	Número de capas entre las Entradas y las Salidas. Keras es muy flexible en la forma que nuevas capas son agregadas.
Optimizador	Utilizado por el algoritmo para evaluar el ritmo de aprendizaje. Keras provee varios optimizadores que para distintas aplicaciones *Adams*[110].
Cantidad de iteraciones (Epochs)	Represents a complete iteration forward and backward over the learning data. This parameter control the fitting of the model to our data, a small value can be as harmful as a big one.

Bibliography

[1] C. Draxl and M. Scheffler, "Nomad: The fair concept for big-data-driven materials science," *arXiv:1805.05039v1*, 2018.

[2] M. Rupp, A. Tkatchenko, K.-R. Müller, and O. A. von Lilienfeld, "Fast and accurate modeling of molecular atomization energies with machine learning," *Phys. Rev. Lett.*, vol. 108, p. 058301, Jan 2012.

[3] F. Brockherde, L. Vogt, L. Li, M. E. Tuckerman, K. Burke, and K.-R. Müller, "Bypassing the kohn-sham equations with machine learning," *Nature communications*, Oct. 2017.

[4] R. Ramakrishnan, M. Hartmann, E. Tapavicza, and O. A. von Lilienfeld, "Electronic spectra from tddft and machine learning in chemical space," *J. Chem. Phys*, vol. 143, p. 084111, July 2015.

[5] K. Gubaev, E. V. Podryabinkin, and A. V. Shapeev, "Machine learning of molecular properties: Locality and active learning," 2018.

[6] A. Grisafi, D. M. Wilkins, G. Csányi, and M. Ceriotti, "Symmetry-adapted machine learning for tensorial properties of atomistic systems," *Physical Review Letters*, vol. 120, 2018.

[7] M. Rupp, R. Ramakrishnan, and O. A. von Lilienfeld, "Machine learning for quantum mechanical properties of atoms in molecules," *The Journal of Physical Chemistry Letters*, vol. 6, no. 16, pp. 3309–3313, 2015.

[8] K. Schütt, A. Tkatchenko, M. Gastegger, and K.-R. Müller, "Unifying machine learning and quantum chemistrywith a deep neural network for molecular wavefunctions," *Nature Communications*, vol. 10, 2019.

[9] K. Ghosh, A. Stuke, M. Todorović, P. Jörgensen, M. Schmidt, A. Vehtari, and P. Rinke, "Deep learning spectroscopy: Neural networks for molecular excitation spectra," *Advanced Science*, vol. 6, p. 1801367, 01 2019.

[10] O. T. Unke and M. Meuwly, "Physnet: A neural network for predicting energies, forces, dipole moments, and partial charges," *Journal of Chemical Theory and Computation*, vol. 15, no. 6, pp. 3678–3693, 2019.

[11] E. Mansimov, S. Kang, O. Mahmood, and K. Cho, "Molecular geometry prediction using a deep generative graph neural network," *Sci Rep*, vol. 9, p. 20381, 2019.

[12] B. R. Goldsmith, M. Boley, J. Vreeken, L. M. Ghiringhelli, and M. Scheffler, "Uncovering structure-property relationships of materials by subgroup discovery," *New Journal of Physics*, vol. 19, 2017.

[13] J. Westermayr and P. Marquetand, "Deep learning for uv absorption spectra with schnarc: First steps towards transferability in chemical compound space," *J. Chem. Phys*, vol. 153, 2020.

[14] F. Ren, L. Ward, T. Williams, K. J. Laws, C. Wolverton, and J. Hattrick-Simpers, "Accelerated discovery of metallic glasses through iteration of machine learning and high-throughput experiments," *Science Advanced*, vol. 4, 2018.

[15] A. Grisafi, B. Meyer, C. Corminboeuf, A. Fabrizio, D. M. Wilkins, and M. Ceriotti, "Transferable machine-learning model of the electron density," *ACS Central Sci*, vol. 5, no. 1, pp. 57–64, 2019.

[16] G. Pilania, J. Gubernatis, T. Lookman, *et al.*, "Multi-fidelity machine learning models for accurate bandgap predictions of solids," *Comput. Mater. Sci.*, vol. 129, pp. 156–163, Dec. 2016.

[17] B. Meredig, A. Agrawal, S. Kirklin, J. W. Doak, K. Zhang, A. Choudhary, J. E. Saal, and A. Thompson, "Combinatorial screening for new materials in unconstrained composition space with machine learning," *Physical Review*, vol. 89, 2014.

[18] R. Ouyang, S. Curtarolo, E. Ahmetcik, M. Scheffler, and L. Ghiringhelli, "Sisso: a compressed-sensing method for systematically identifying efficient physical models of materials properties," *Phys. Rev. Materials*, vol. 2, p. 083802, Aug 2018.

[19] S. De, A. P. Bartók, G. Csányic, and M. Ceriotti, "Comparing molecules and solids across structural and alchemical space," *Phisycal Chemistry Chemical Physics*, vol. 18, pp. 13754–13769, may 2016.

[20] A. P. Bartók, R. Kondor, and G. Csányi, "On representing chemical environments," *Physical Review B*, vol. 87, p. 184115, 2013.

[21] A. P. Barto and G. Csányi, "Gaussian approximation potentials: A brief tutorial introduction," *International Journal of Quantum Chemistry*, no. 115, pp. 1051–1057, 2015.

[22] R. M. Mar[í]n, N. F. Aguirre, and E. E. Daza, "Graph theoretical similarity approach to compare molecular electrostatic potentials," *Journal of Chemical Information and Modeling*, vol. 48, no. 1, pp. 109–118, 2008.

[23] C. A. Guido, P. Cortona, B. Mennucci, and C. Adamo, "On the metric of charge transfer molecular excitations: A simple chemical descriptor," *Journal of Chemical Theory and Computation*, vol. 9, no. 7, pp. 3118–3126, 2013. PMID: 26583991.

[24] A. Sadeghi, S. A. Ghasemi, B. Schaefer, S. Mohr, M. A. Lill, and S. Goedecker, "Metrics for measuring distances in configuration spaces," *J. Chem. Phys*, vol. 139, p. 184118, 2013.

[25] J. Dong, C. DS, M. HY, S. Liu, D. BC, Y. YH, W. NN, L. AP, Z. WB, and C. AF, "Chemdes: An integrated web-based platform for molecular descriptor and fingerprint computation.," *J Cheminform*, vol. 7, p. 60, 2015.

[26] B. Huang and O. A. von Lilienfeld, "Communication: Understanding molecular representations in machine learning: The role of uniqueness and target similarity," *J. Chem. Phys*, vol. 145, p. 161102, 2016.

[27] C. R. Collins, G. J. Gordon, O. A. von Lilienfeld, and D. J. Yaron, "Constant size descriptors for accurate machine learning models of molecular properties," *J. Chem. Phys*, vol. 148, p. 241718, 2018.

[28] A. E. Sifain, N. Lubbers, B. T. Nebgen, J. S. Smith, A. Y. Lokhov, O. Isayev, A. E. Roitberg, K. Barros, and S. Tretiak, "Discovering a transferable charge assignment model using machine learning," *The Journal of Physical Chemistry Letters*, vol. 9, no. 16, pp. 4495–4501, 2018.

[29] L. Himanen, M. O. J. Jäger, Eiaki V. Morooka, F. F. Canova, Y. S. Ranawat, D. Z. Gao, P. Rinke, and A. S. Foster, "Dscribe: Library of descriptors for machine learning in materials science," *Computer Physics Communications*, vol. 247, p. 106949, 2020.

[30] T. Gao, S.-L. Sun, L.-L. Shi, H. Li, H.-Z. Li, Z.-M. Su, and Y.-H. Lu, "An accurate density functional theory calculation for electronic excitation energies: The least-squares support vector machine," *J. Chem. Phys*, vol. 130, no. 18, p. 184104, 2009.

[31] F. Häse, S. Valleau, E. Pyzer-Knapp, and A. Aspuru-Guzik, "Machine learning exciton dynamics," *Chemical Science*, vol. 7, pp. 5139–5147, 2016.

[32] F. A. Faber, L. Hutchison, B. Huang, J. Gilmer, S. S. Schoenholz, O. Vinyals, G. E. Dahl, S. Kearnes, P. F. Riley, and O. A. von Lilienfeld, "Prediction errors of molecular machine learning models lower than hybrid dft error," *Journal of Chemical Theory and Computation*, vol. 13, no. 11, pp. 5255–5264, 2017.

[33] K. T. Schütt, F. Arbabzadah, S. Chmiela, K. R. Müller, and A. Tkatchenko, "Quantum-chemical insights from deep tensor neural networks," *Nature communications*, no. 8, p. 13890.

[34] Y. Cytter, D. Neuhauser, and R. Baer, "Development of generalized potential-energy surfaces using many-body expansions, neural networks, and moiety energy approximations," vol. 130, 2009.

[35] M. A. L. Marques, N. T. Maitra, F. M. S. Nogueira, E. K. U. Gross, and A. Rubio, *Fundamentals of Time-Dependent Density Functional Theory*. Physics Editorial Department I, 2012.

[36] J. D. Whitfield, M.-H. Yung, D. G. Tempel, S. Boixo, and A. Aspuru-Guzik, "Computational complexity of time-dependent density functional theory," *New Journal of Physics*, vol. 16, 2014.

[37] S. Boixo, S. V. Isakov, V. N. Smelyanskiy, R. Babbush, N. Ding, Z. Jiang, M. J. Bremner, J. M. Martinis, and H. Neven, "Characterizing quantum supremacy in near-term devices," *arXiv:1608.00263v3*, 2017.

[38] L. Gyongyosi and S. Imre, "A survey on quantum computing technology," *Computer Science Review*, vol. 31, pp. 51–71, 2019.

[39] P. Hohenberg and W. Kohn, "Inhomogeneous electron gas," *Physical Review*, vol. 136, no. 3B, p. B864, 1964. Copyright (C) 2008 The American Physical Society Please report any problems to prola@aps.org PR.

[40] R. G. Parr and Y. Weitao, *Density-functional theory of atoms and molecules*. Oxford University Press, Inc, 1989.

[41] C. J. Cramer, *Essentials of Computational Chemistry: Theories and Models*. Wiley, 2004.

[42] N. Mardirossian and M. Head-Gordon, "Thirty years of density functional theory in computational chemistry: an overview and extensive assessment of 200 density functionals," *Molecular Physics An International Journa*, vol. 115, no. 19, 2017.

[43] A. Szabo and N. S. Ostlund, *Modern Quantum Chemistry Introduction to Advanced Electronic Structure Theory*. Dover Publications, INC, 1996.

[44] Y. Cytter, D. Neuhauser, and R. Baer, "Metropolis evaluation of the hartree fock exchange energy," vol. 10, no. 10, pp. 4317–4323, 2014.

[45] D. R. HARTREE, *The Calculation of Atomic Structures*. New York, JOHN WILEY and SONS. Inc.London CHAPMAN and HALL Ltd., 1955.

[46] S. Boixo, S. V. Isakov, V. N. Smelyanskiy, R. Babbush, N. Ding, Z. Jiang, M. J. Bremner, J. M. Martinis, and H. Neven, "Accurate density functional for the energy: Real-space cutoff of the gradient expansion for the exchange hole," *Physical Review Letters*, vol. 55, 1985.

[47] J. P. Perdew, K. Burke, and M. Ernzerhof, "Rationale for mixing exact exchange with density functional approximations," *J. Chem. Phys*, vol. 105, Dec. 1996.

[48] C. Ullrich, *Time-Dependent Density-Functional Theory: Concepts and Applications*. Oxford University Press, 2012.

[49] A. D. Becke, "A new mixing of hartree-fock and local density-functional theories," *jcp*, vol. 98, pp. 1372–1377, Jan. 1993.

[50] J. Leszczynski, ed., *Comput. Mater. Sci.*, vol. 15. Elsevier, 2004.

[51] C. Adamo and V. Barone, "Toward reliable density functional methods without adjustable parameters: The pbe0 model," *J. Chem. Phys*, vol. 110, Apr 1999.

[52] J. P. Perdew and K. Schmidt, "Jacob's ladder of density functional approximations for the exchange-correlation energy," *AIP Conf. Proc*, vol. 557, no. 1, 2001.

[53] M. E. Casida, "Time-dependent density functional response theory for molecules," *ReseachGate*, nov 1995.

[54] P. Pulay and G. Fogarasi, "Geometry optimization in redundant internal coordinates," *J. Chern. Phys.*, vol. 96, feb 1992.

[55] F. Eckert, P. Pulay, and H.-J. Werner, "Ab initio geometry optimization for large molecules," *Journal Comput Chem*, vol. 18, pp. 1473–1483, 1997.

[56] T. U. Helgaker, P. Jorgensen, and J. Simons, *Geometrical Derivatives of Energy Surfaces and Molecular Properties*. Reidel Publishing Company., 1986.

[57] S. Chatzieleftheriou, M. R. Adendorff, and N. D. Lagaros, "Generalized potential energy finite elements for modeling molecular nanostructures," *J. Chem. Inf. Model.*, vol. 56, no. 10, pp. 1963–1978, 2016.

[58] T. A. Halgren, "Merck molecular force field. i. basis, form, scope, parameterization, and performance of mmff94," *Journal of Computational Chemistry*, vol. 17, pp. 490–519, Apr 1996.

[59] A. K. Rappe, C. J. Casewit, K. S. Colwell, W. A. Goddard, and W. M. Skiff, "UFF, a full periodic table force field for molecular mechanics and molecular dynamics simulations," *J . Am. Chem. SOC*, vol. 114, no. 25, pp. 10024–10035, 1992.

[60] S. Chatzieleftheriou, M. R. Adendorff, and N. D. Lagaros, "Generalized potential energy finite elements for modeling molecular nanostructures," *Journal Chemical Information and Modeling*, vol. 56, no. 10, pp. 1963–1978, 1997.

[61] J. McIVER, Jr. and A. KOMORNICKI, "Rapid geometry optimization for semi-empirical molecular orbital methods," *CHEMICAL PHYSICS LETTERS*, vol. 10, no. 3, pp. 303–306, 1971.

[62] D. Marx and J. Hutter, *Ab initio molecular dynamics basic theory and advanced methods*. CAMBRIDGE UNIVERSITY PRESS, 2009.

[63] J. A. Snyman and D. N. Wilke, *Practical Mathematical Optimization Basic Optimization Theory and Gradient-Based Algorithms*. Springer, 2nd ed., 2018.

[64] H. B. Schlegel, *Some Practical Suggestions for Optimizing Geometries and Locating Transition States*. New Theoretical Concepts for Understanding Organic Reactions, Kluwer Academic Publishers, 1989.

[65] C. Q. Sun, "Relaxation of the chemical bond skin chemisorption size matter ztp mechanics h2o myths," 2014.

[66] V. N. Vapnik, *The Nature of Statistical Learning Theory*. Statistics for Engineering and Information Science Series Editors M. Jordan, S.L. Lauritzen, J.E Lawless, V. Nair Springer Science Business Media, 2nd ed., 2000.

[67] S. Theodoridis and K. Koutroumbas, *Pattern Recognition*. Elsevier, 4th edition ed., 2008.

[68] O. Simeone, "A very brief introduction to machine learningwith applications to communication systems," *IEEE*, vol. 4, no. 4, pp. 648–664, 2018.

[69] V. N. Vapnik, "An overview of statistical learning theory," *IEEE Transactions on Neural Networks*, vol. 10, no. 5, pp. 988–999, 1999.

[70] C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer Science, 2006.

[71] M. P. Kumar, P. Torr, and A. Zisserman, "An invariant large margin nearest neighbour classifier," *2007 IEEE 11th International Conference on Computer Vision*, pp. 1–8, 2007.

[72] K. Q. Weinberger and L. K. Saul, "Distance metric learning for large margin nearest neighbor classification," *Journal of Machine Learning Research*, vol. 10, pp. 207–244, 2009.

[73] J. Goldberger, S. Roweis, G. Hinton, and R. Salakhutdinov, "Neighbourhood components analysis," *MIT Press*, 2005.

[74] S. Theodoridis, *Machine Learning A Bayesianand Optimization Perspective*, vol. 1. Elsevier, 1 ed., 2015.

[75] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," *Proceedings of the 27 International Conference on Machine Learning*, 2010.

[76] R. H. R. Hahnloser, H. S. Seung, and J.-J. Slotine, "Permitted and forbidden set s in symmetric threshold-linear networks," *Neural Computation*, vol. 15, pp. 621–638, 2003.

[77] D.-A. Clevert, T. Unterthiner, and S. Hochreiter, "Fast and accurate deep network learning by exponential linear units (elus)," *ICLR*, p. arXiv:1511.07289v5, 2016.

[78] K. Fukushima, "A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position," *Biol. Cybernetic*, vol. 45, 1980.

[79] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation applied to handwritten zip code recognition," *Neural Computation*, vol. 1, p. 541–551, 1989.

[80] J. Mockus, *Bayesian Approach to Global Optimization Theory and Applications.* Kluwer Academic Publishers, 1 ed., 1989.

[81] P. I. Frazier, "A tutorial on bayesian optimization," *arXiv:1807.02811 [stat.ML]*, 2018.

[82] X. Andrade, J. Alberdi-Rodriguez, D. A. Strubbe, M. J. T. Oliveira, F. Nogueira, A. Castro, J. Muguerza, A. Arruabarrena, S. G. Louie, A. Aspuru-Guzik, A. Rubio, and M. A. L. Marques, "Time-dependent density-functional theory in massively parallel computer architectures: the octopus project," *Journal of Physics: Condensed Matter*, vol. 24, p. 233202, may 2012.

[83] S. Lehtola, C. Steigemann, M. J. Oliveira, and M. A. Marques, "Recent developments in libxc – a comprehensive library of functionals for density functional theory," *SoftwareX*, vol. 7, pp. 1–5, 2018.

[84] A. H. Larsen, J. J. Mortensen, J. Blomqvist, I. E. Castelli, R. Christensen, M. Dułak, J. Friis, M. N. Groves, B. Hammer, C. Hargus, E. D. Hermes, P. C. Jennings, P. B. Jensen, J. Kermode, J. R. Kitchin, E. L. Kolsbjerg, J. Kubal, K. Kaasbjerg, S. Lysgaard, J. B. Maronsson, T. Maxson, T. Olsen, L. Pastewka, A. Peterson, C. Rostgaard, J. Schiøtz, O. Schütt, M. Strange, K. S. Thygesen, T. Vegge, L. Vilhelmsen, M. Walter, Z. Zeng, and K. W. Jacobsen, "The atomic simulation environment–a python library for working with atoms," *Journal of Physics: Condensed Matter*, vol. 29, no. 27, p. 273002, 2017.

[85] G. Landrum *et al.*, "Rdkit: Open-source cheminformatics," 2006.

[86] C. Scharfer, T. Schulz-Gasch, *et al.*, "Torsion angle preferences in druglike chemical space: A comprehensive guide," *Journal of Medicinal Chemistry*, vol. 56, no. 5, pp. 2016–2028, 2013.

[87] C. Cortes, M. Mohri, and A. Rostamizadeh, "Learning non-linear combinations of kernels.," vol. 22, pp. 396–404, 2009.

[88] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster,

J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "Tensorflow: Large-scale machine learning on heterogeneous systems." https://www.tensorflow.org, 2015. Software available from tensorflow.org.

[89] F. Chollet *et al.*, "Keras." https://keras.io, 2015.

[90] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning*. The MIT Press, 2006.

[91] C. M. de Armas-Morejón, A. H. Larsen, L. A. Montero-Cabrera, A. Rubio, and J. Jornet-Somoza, "A basic electro-topological descriptor for the prediction of organic molecule geometries by simple machine learning," *https://arxiv.org/abs/2210.10700*, 2022.

[92] G. Raggi, I. F. Galván, C. L. Ritterhoff, M. Vacher, and R. Lindh, "Restricted-variance molecular geometry optimization based on gradient-enhanced kriging," *Journal of Chemical Theory and Computation*, vol. 16, no. 6, pp. 3989–4001, 2020. PMID: 32374164.

[93] S. Riniker and G. A. Landrum, "Better informed distance geometry: Using what we know toimprove conformation generation," *Chem. Inf. Model.*, vol. 55, no. 12, pp. 2562–2574, 2015.

[94] G. Crippen and T. Havel, *Distance Geometry and Molecular Conformation*. 1988.

[95] L. Ghiringhelli, J. Vybiral, S. V. Levchenko, C. Draxl, and M. Scheffler, "Big data of materials science - critical role of the descriptor," *Physical Review Letters*, vol. 114, p. 105503, Mar 2015.

[96] I. Tanaka, ed., *Nanoinformatics*. Springer, ene 2018.

[97] L. H. Hall and L. B. Kier, "Electrotopological state indices for atom types: A novel combination of electronic, topological, and valence state information.," *Journal of Chemical Information and Computer Sciences*, vol. 35, no. 6, pp. 1039–1045, 1995.

[98] L. H. Hall, B. Mohney, and L. B. Kier, "The electrotopological state: Structure information at the atomic level for molecular graphs.," *Journal of Chemical Information and Computer Sciences*, vol. 31, no. 1, pp. 76–82, 1991.

[99] L. Ruddigkeit, R. van Deursen, L. C. Blum, and J.-L. Reymond, "Enumeration of 166 billion organic small molecules in the chemical universe database gdb-17," *Journal of Chemical Information and Modeling*, vol. 52, no. 11, p. 2864–2875, 2012.

[100] R. Ramakrishnan, P. O. Dral, M. Rupp, and O. A. von Lilienfeld, "Quantum chemistry structures and properties of 134 kilo molecules," *Scientific Data*, vol. 1, p. 140022, 2014.

[101] P. J. Stephens, F. J. Devlin, C. F. Chabalowski, and M. J. Frisch, "Ab initio calculation of vibrational absorption and circular dichroism spectra using density functional force fields," *ACS Publications*, vol. 98, pp. 11623–11627, Nov. 1994.

[102] J. M. Blaney and J. S. Dixon, *Distance Geometry in Molecular Modeling*. 1994.

[103] F. Musil, M. J. Willatt, M. A. Langovoy, and M. Ceriotti, "Fast and accurate uncertainty estimation in chemical machine learning," *Journal of Chemical Theory and Computation*, vol. 15, no. 2, pp. 906–915, 2019.

[104] K. Jacobsen, P. Stoltze, and J. Norskov, "A semi-empirical effective medium theory for metals and alloys," *ELSEVIER Surface Science*, vol. 366, no. 2, pp. 394–402, 1996.

[105] C. M. de Armas-Morejón, L. A. Montero-Cabrera, A. Rubio, and J. Jornet-Somoza, "Electronic descriptors for supervised spectroscopic predictions," *https://arxiv.org/abs/2205.12074*, 2022.

[106] H. Kawai and Y. O. Nakagawa, "Predicting excited states from ground state wavefunction by supervised quantum machine learning," vol. 1, 2020.

[107] M. C. Kennedy and A. H. Hagan, "Predicting the output from a complex computer code when fast approximations are avalible," *Biometrika*, vol. 87, p. 13, Mar. 2000.

[108] G. Montavon, K. Hansen, S. Fazli, M. Rupp, F. Biegler, A. Ziehe, A. Tkatchenko, O. A. von Lilienfeld, and K.-R. Müller, "Learning invariant representations of molecules for atomization energy prediction," *Computer Physics Communications*, 2020.

[109] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. Jarrod Millman, N. Mayorov, A. R. J. Nelson,

E. Jones, R. Kern, E. Larson, C. Carey, İ. Polat, Y. Feng, E. W. Moore, J. Vand erPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, and S. . . Contributors, "Scipy 1.0-fundamental algorithms for scientific computing in python," *arXiv e-prints*, p. arXiv:1907.10121, Jul 2019.

[110] D. P. Kingma and L. immy Ba, "Adam: A method for stochastic optimization," *3rd International Conference for Learning Representations*, p. arXiv:1412.6980v8, 2015.

[111] P. Szymański and T. Kajdanowicz, "scikit-multilearn: A python library for multi-label classification," *Journal of Machine Learning Research*, vol. 20, no. 6, pp. 1–22, 2019.

[112] J. J. Irwin and B. K. Shoichet, "Zinc – a free database of commercially available compounds for virtual screening," *J Chem Inf Model*, vol. 45, no. 1, pp. 177–182, 2005.

[113] J. Schmidt, M. R. G. Marques, S. Botti, and M. A. L. Marques, "Recent advances and applications of machine learning in solid-state materials science," *Nature Patner Journal Computational Materials*, vol. 5, 2019.

[114] S. Dick and M. Fernandez-Serra, "Machine learning accurate exchange and correlation functionals of the electronic density," *Nature Communications*, vol. 11, 2020.