



UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE CIENCIAS

ANN'S COMO PROPAGADORES EN DINÁMICA CUÁNTICA

T E S I S

Que para obtener el título de:

Física

Presenta:

ITZEL JESSICA MARTÍNEZ MARCELO

Tutor:

Dr. Huziel Enoc Saucedá Félix

2023 – classicthesis v4.6

Itzel Jessica Martínez Marcelo : *ANN's como Propagadores en Dinámica Cuántica*,  
Facultad de Ciencias, © 2023

TUTOR:

Dr. Huziel Enoc Saucedá Félix LOCATION:

Ciudad de México, México



## ÍNDICE GENERAL

---



---

## ÍNDICE DE FIGURAS

---



---

## ÍNDICE DE CUADROS

---



---

## LISTINGS

---



---

## ACRÓNIMOS

TDSE	Ecuación de Schrödinger Dependiente del Tiempo (Time Dependent Schrödinger Equation)
DVR	Representación de Variable Discreta (Discrete Variable Representation)
ANN	Red Neuronal Artificial (Artificial Neural Network)
RNN	Red Neuronal Recurrente (Recurrent Neural Network)
LSTM	(Long Short-Term Memory)

## Parte I

### PRÓLOGO

# 1



## INTRODUCCIÓN

La mecánica cuántica es un área de la física que describe sistemas a escalas atómicas. Dado que en la Física, como en las ciencias en general, gran importancia de una teoría radica en su poder de predecir, el entendimiento de la mecánica cuántica fue cuestionado por las personas de la época en sus inicios (1920's), pues sus fundamentos eran ideas revolucionarias contra intuitivas con la descripción de la naturaleza que ofrecía la mecánica que hasta entonces se conocía: la mecánica clásica, pues en esta última, dado un sistema y sus condiciones iniciales, preguntas como: ¿Dónde estará la partícula al tiempo  $t$ ?, tenían una respuesta concreta que podía ser comprobada mediante experimentos. La mecánica cuántica por otro lado, plantea una respuesta en términos de una densidad de probabilidad, en donde, hasta hacer una medición, podremos saber con exactitud la posición de una partícula. La interpretación de la mecánica cuántica sigue siendo un tema que genera discusión, sin embargo, la validez y fortaleza de la teoría no son puestas en duda, pues es capaz de explicar el comportamiento de las partículas a pequeña escala, así como sus interacciones, de manera exitosa y de acuerdo a las pruebas experimentales.

La evolución temporal de un sistema cuántico es de gran interés para diversas áreas de la ciencia, pues tiene diversas aplicaciones para procesos químicos y biológicos. En tales procesos es importante tener una descripción cuántica del sistema, ya que trata de interacciones a niveles atómicos o moleculares. La Ecuación de Schrödinger Dependiente del Tiempo, **TDSE**, por sus siglas en inglés, es la ecuación que describe la evolución temporal de un sistema cuántico; por otro lado, el Hamiltoniano de un sistema, en el formalismo de la mecánica cuántica, es un operador que contiene la información de la energía total del sistema.

La solución a la ecuación de Schrödinger depende del Hamiltoniano del sistema, y cuando el Hamiltoniano tiene una dependencia temporal, como es el caso de muchos sistemas de procesos químicos, adquiere una forma más compleja dependiendo del sistema. Para resolver la **TDSE** y encontrar una descripción de la evolución del sistema, se pueden emplear diversos métodos analíticos o numéricos; sin embargo, dependiendo de la naturaleza del sistema, los cálculos y su tiempo de cómputo pueden llegar a representar un problema para obtener resultados de manera eficiente.

En los últimos años, las Redes Neuronales Artificiales Red Neuronal Artificial (Artificial Neural Network)s (ANNs) por sus siglas en inglés, han sido modelos de Aprendizaje Automático que han cobrado popularidad debido a los avances en computación, al uso de Unidades de Procesamiento Gráfico (GPU's), al mejoramiento en algoritmos y modelos; y al incremento de datos. Actualmente, en diversos problemas, pueden ofrecer resultados que consumen menos tiempo y/o memoria de cómputo, con una precisión exitosa. De la manera más general, las ANNs pueden verse como gráficas computacionales de operaciones matemáticas, cuya forma de aprender<sup>1</sup> se basa en procesar ejemplos de datos referentes al problema o sistema en cuestión.

El objetivo de este trabajo es la implementación de una ANN que funcione como propagador para cierto tipo de sistemas cuánticos, es decir, que mapee una función de onda en un tiempo inicial  $t$ :  $\psi(r, t)$ , a la función de onda después de un intervalo de tiempo  $\Delta t$ :  $\psi(r, t + \Delta t)$ , bajo condiciones iniciales que dependen del sistema, obteniendo así, un modelo alternativo a los métodos analíticos y numéricos para la resolución de la TDSE para este tipo de sistemas.

## 1.1 ORGANIZACIÓN

En la segunda parte de la tesis (Parte ii) se abordan conceptos básicos de Dinámica Cuántica y la Ecuación de Schrödinger Dependiente del Tiempo (TDSE) para la descripción de la evolución temporal de sistemas cuánticos. En el Capítulo 3 se desarrolla un ejemplo de método numérico para la resolución de la TDSE aplicada al tipo de sistemas cuánticos de interés para este trabajo.

La tercera parte de la tesis (Parte iii) describe una breve introducción sobre Aprendizaje Automático y aprendizaje supervisado, posteriormente se describen los fundamentos y características principales de las redes neuronales artificiales (ANNs).

La última parte de la tesis (Parte iv), detalla la implementación de un modelo particular de ANN como propagador aplicado a los sistemas cuánticos de interés (§ 3.3.1); la obtención y manejo de datos, arquitectura de la red, los parámetros e hiper-parámetros establecidos; así como la precisión obtenida y las predicciones del modelo entrenado. Finalmente se presenta un análisis de los resultados y sugerencias para implementar en trabajos futuros relacionados.

---

<sup>1</sup>Históricamente las ANNs fueron inspiradas por las estructuras neuronales biológicas; un concepto como el *entrenamiento* de una ANN, puede interpretarse como el proceso análogo de *aprendizaje* que tienen los organismos biológicos.

## Parte II

# ECUACIÓN DE SCHRÖDINGER DEPENDIENTE DEL TIEMPO



# 2



## CONCEPTOS FUNDAMENTALES DE MECÁNICA CUÁNTICA

El formalismo matemático general para trabajar en mecánica cuántica es el álgebra lineal. En las siguientes secciones se desarrollan algunos de los conceptos fundamentales en mecánica cuántica utilizando la notación de *bra-ket*, introducida por primera vez por Paul Dirac.

### 2.1 ESTADOS Y OPERADORES

La información física de un sistema en mecánica cuántica está contenida en su **estado o ket**, que en general es un elemento del espacio de Hilbert<sup>1</sup>.

$$|\psi\rangle$$

La dimensión del espacio depende de las diferentes posibilidades o alternativas que el sistema puede tomar, por ejemplo, si el sistema es un átomo y la medición que se está realizando es de su spin, el átomo, que se representa por su estado, pertenece a un espacio vectorial de dos dimensiones, dado que existen dos posibilidades de spin. En ocasiones, las alternativas que puede tomar un estado son infinitas y no numerables, por ejemplo, cuando se requiere saber la posición o momento de una partícula.

Un **operador**  $\hat{a}$ , en mecánica cuántica, es también un elemento del espacio de Hilbert, que opera sobre los estados, y cuyo resultado es, en general un estado distinto, que pertenece al mismo espacio.

$$\hat{a} \cdot (|\psi\rangle) = \hat{a} |\psi\rangle$$

Para cada operador  $\hat{a}$  existen kets “especiales” llamados eigenkets o **eigenestados** de  $\hat{a}$ :  $|a\rangle$ , que al aplicarles el operador resulta el mismo estado  $|a\rangle$  multiplicado por  $a$ , es decir:

---

<sup>1</sup>Los espacios de Hilbert son espacios vectoriales reales o complejos de dimensión infinita que cuentan con un producto punto.

$$\hat{a}|a\rangle = a|a\rangle$$

en donde  $a$  puede ser un número real o complejo, y es llamado **eigenvalor** de  $\hat{a}$ . Un **observable** en mecánica cuántica puede ser la posición, el momento o el spin de una partícula, y se puede representar mediante un operador. Una propiedad importante de los observables, es que son operadores Hermitianos<sup>2</sup>, por lo que sus eigenvalores son valores reales, y además sus eigenestados son ortonormales, y, por construcción, forman una base completa del espacio de Hilbert.

## 2.2 OPERADOR DE PROPAGACIÓN: EVOLUCIÓN TEMPORAL DE UN SISTEMA

Dado un sistema cuántico a un tiempo inicial  $t_0$ , representado por el estado:

$$|\psi_{t_0}\rangle$$

la evolución temporal del estado a un tiempo  $t > t_0$  se puede escribir como:

$$|\psi_{t_0}(t)\rangle$$

haciendo referencia de que el estado evolucionado en el tiempo depende del estado inicial al tiempo  $t_0$ .

Bajo el formalismo de la mecánica cuántica, se puede visualizar a este nuevo estado  $|\psi_{t_0}(t)\rangle$ , como el resultado de aplicar un operador al estado  $|\psi_{t_0}\rangle$ :

$$|\psi_{t_0}(t)\rangle = \mathcal{U}(t_0, t) |\psi_{t_0}\rangle$$

donde el operador  $\mathcal{U}$  es el **operador de propagación**. Para que este operador sea físicamente consistente, debe satisfacer las siguientes propiedades:

- $\mathcal{U}$  debe ser un operador unitario, es decir, se debe cumplir que:

$$\mathcal{U}^\dagger(t_0, t) \mathcal{U}(t_0, t) = 1$$

- Propiedad de composición, para  $t_0 < t_1 < t_2$ :

$$\mathcal{U}(t_0, t_2) = \mathcal{U}(t_1, t_2) \mathcal{U}(t_0, t_1)$$

A partir de estas propiedades, de la relación de energía Planck-Einstein:  $E = \hbar\omega$ , y de considerar al Hamiltoniano como el generador de la evolución temporal

<sup>2</sup>El espacio de Hilbert Dual de los kets es el espacio de los bras, cada estado  $|\psi\rangle$  tiene su dual correspondiente:  $\langle\psi|$ , así como cada operador  $X$  tiene su dual correspondiente  $X^\dagger$ , llamado Hermitiano adjunto, cuando resulta que  $X = X^\dagger$ , se dice que  $X$  es un **operador Hermitiano**.

de un sistema físico[13], se puede escribir la siguiente ecuación diferencial para el operador de propagación:

$$i\hbar \frac{\partial}{\partial t} \mathcal{U}(t_0, t) = H \mathcal{U}(t_0, t) \quad (2.1)$$

Se sigue que, para un estado  $|\psi_{t_0}\rangle$ , la evolución temporal está dada por la **Ecuación de Schrödinger Dependiente del Tiempo** para un estado:

$$i\hbar \frac{\partial}{\partial t} \mathcal{U}(t_0, t) |\psi_{t_0}\rangle = H \mathcal{U}(t_0, t) |\psi_{t_0}\rangle \quad (2.2)$$

**AFIRMACIÓN:** Cuando el hamiltoniano  $H$  no tiene dependencia temporal, el operador de propagación toma la siguiente forma:

$$\mathcal{U}(t_0, t) = \exp\left\{\frac{-iH(t - t_0)}{\hbar}\right\} \quad (2.3)$$

Para probar la afirmación anterior, se escribe la función exponencial como serie de potencias:

$$\exp\left\{\frac{-iH(t - t_0)}{\hbar}\right\} = 1 + \frac{-iH(t - t_0)}{\hbar} + \left(\frac{-i^2}{2}\right) \left(\frac{H(t - t_0)}{\hbar}\right)^2 + \dots$$

al derivar respecto al tiempo se obtiene:

$$\frac{\partial}{\partial t} \exp\left\{\frac{-iH(t - t_0)}{\hbar}\right\} = \frac{-iH \cdot 1}{\hbar} + \left(\frac{-i^2}{2}\right) \cdot 2 \left(\frac{H}{\hbar}\right)^2 (t - t_0) + \dots$$

si se multiplica esta última ecuación se observa que se cumple la **Ecuación 2.1**.

Cuando el Hamiltoniano tiene dependencia temporal, pero conmuta en diferentes tiempos, es decir:

$$[H(t_1), H(t_2)] = H(t_1)H(t_2) - H(t_2)H(t_1) = 0$$

para  $t_1 \neq t_2$ , se puede probar que el operador de propagación está dado por:

$$\mathcal{U}(t_0, t) = \exp\left\{-\left(\frac{i}{\hbar}\right) \int_{t_1}^{t_2} H(t') dt'\right\}$$

Encontrar de manera analítica el operador de propagación se vuelve más complejo y complicado cuando el Hamiltoniano tiene dependencia temporal y no conmuta en diferentes tiempos, para estos casos se pueden desarrollar métodos de aproximación a la **Ecuación 2.2** como: la Expansión de Magnus, Teoría de Perturbaciones con dependencia temporal, integral de caminos, entre otros.

## 2.3 FUNCIÓN DE ONDA EN EL ESPACIO DE POSICIONES

Un observable comúnmente estudiado en sistemas cuánticos es la posición:  $\hat{x}$ . Como se mencionó en la sección § 2.1, un observable da lugar a un conjunto de eigenvectores:  $\{|x'\rangle, |x''\rangle \dots\}$ , con las siguientes propiedades:

$$\begin{aligned}\hat{x}|x'\rangle &= x'|x'\rangle \\ \langle x''|x'\rangle &= \delta(x'' - x')\end{aligned}$$

Cualquier estado el tiempo  $t$ :  $|\psi(t)\rangle$ , puede ser escrito en función de los eigenestados de la siguiente manera:

$$|\psi(t)\rangle = \int dx |x\rangle \langle x|\psi(t)\rangle$$

En el formalismo de Dirac, el producto interno:

$$\langle x|\psi(t)\rangle = \psi(x, t)$$

es conocido como la **función de onda** para el estado  $|\psi(t)\rangle$ . Las funciones de onda en física cuántica deben cumplir con la condición de normalización:

$$\int_{-\infty}^{\infty} |\psi(x, t)|^2 dx < \infty$$

este conjunto de funciones de onda forma un espacio de Hilbert.

### 2.3.1 Densidad de probabilidad

La **densidad de probabilidad** está definida por la siguiente función:

$$\rho(x) = |\psi(x)|^2 \quad (2.4)$$

Esta relación describe para cada  $x$  en el dominio la probabilidad de encontrar a la partícula en esa posición. La probabilidad de encontrar una partícula en un intervalo  $dx$  alrededor de  $x$  está dado por:

$$|\psi(x)|^2 dx$$

El elemento  $dx$  puede ser un intervalo de longitud, cuando se considera una sola dimensión, o un elemento de volumen  $d^3\vec{x}$ , cuando se tratan las tres dimensiones espaciales. Experimentalmente, está relacionado con la capacidad que tiene un detector (de posición) de asegurar que la partícula está en la vecindad  $dx$  alrededor de  $x$ .

# 3



## REPRESENTACIÓN DE VARIABLE DISCRETA

Existen diferentes métodos numéricos para resolver la ecuación de Schrödinger Dependiente del Tiempo ([Ecuación 2.2](#)). En esta capítulo se desarrollará un método particular conocido como el método de **Representación de Variable Discreta**, **DVR** por sus siglas en inglés.

Formalmente, el espacio de Hilbert de las funciones de onda es infinito, sin embargo, para resolver y tratar problemas de numéricamente, es necesario hacer un truncamiento de la dimensión del espacio a un número finito  $N$ . Este espacio reducido de Hilbert, puede verse como un espacio generado por un operador de proyección, además, el espacio reducido de Hilbert tiene el mismo formalismo de la mecánica cuántica, pues, si se recuerda que la dimensión del espacio de Hilbert está dado por las distintas alternativas que puede tomar el sistema cuántico en cuestión, este espacio de dimensión  $N$ , puede ser un espacio completo para otro problema en mecánica cuántica.

### 3.1 PROYECCIÓN ESPECTRAL

Una función de onda y sus operadores se pueden representar mediante una base de funciones ortogonales, a esta base se le conoce como **base espectral**:

$$\{\phi_i(x)\}_{i=1}^N$$

que, por ser funciones ortogonales cumplen que:

$$\langle \phi_i(x) | \phi_j(x) \rangle = \delta_{ij}$$

Como se mencionó anteriormente, la dimensión del espacio de Hilbert se debe reducir a un número finito  $N$ , esta reducción de dimensión se puede expresar mediante un **operador de proyección**:

$$P_N = \sum_{n=1}^N |\phi_n\rangle \langle \phi_n| \quad (3.1)$$

Mediante el operador  $P_N$  se puede mapear la dinámica del espacio de Hilbert al espacio reducido de Hilbert, en particular, es de interés para resolver la **Ecuación 2.2** conocer cómo se representa el operador Hamiltoniano:

$$H = \frac{\hat{p}^2}{2m} + V(\hat{x}) \quad (3.2)$$

Utilizando la base espectral, el Hamiltoniano en el espacio reducido de Hilbert se puede representar como:

$$H_N = P_N H P_N \quad (3.3)$$

Definiendo  $Q_N = 1 - P_N$ , la Ecuación de Schrödinger Dependiente del Tiempo se puede escribir en términos de  $P_N$  y  $Q_N$  como un conjunto de dos ecuaciones diferenciales acopladas:

$$i\hbar \frac{\partial \psi_N}{\partial t} = P_N H P_N \psi_N + P_N H Q_N \psi_{\perp} \quad (3.4)$$

$$i\hbar \frac{\partial \psi_{\perp}}{\partial t} = Q_N H P_N \psi_N + Q_N H Q_N \psi_{\perp} \quad (3.5)$$

en donde:  $\psi_N = P_N \psi$  y  $\psi_{\perp} = Q_N \psi$ .

Para este trabajo se usará la aproximación de Galerkin [7], en donde se desprecia la contribución de  $\psi_{\perp}$ , de esta forma, la Ecuación de Schrödinger Dependiente del Tiempo a resolver es:

$$i\hbar \frac{\partial \psi_N}{\partial t} = P_N H P_N \psi_N \quad (3.6)$$

### 3.1.1 Representación de la Función de Onda

Una función de onda  $\psi(x)$  se puede escribir como una suma infinita en términos de funciones ortonormales  $\phi_i$ :

$$\psi(x) = \sum_{n=1}^{\infty} a_n \phi_n(x) \quad (3.7)$$

con:

$$\int \phi_m^*(x) \phi_n(x) dx = \delta_{mn}$$

$$a_n = \int \phi_n^*(x) \psi(x) dx$$

para  $m, n = 1, 2, 3, \dots, \infty$ .

Utilizando la definición del operador de proyección de la [Ecuación 3.1](#), se puede probar que:

$$\psi_N(x) = P_N \psi(x) = \sum_{n=1}^N a_n \phi_n(x) \quad (3.8)$$

### 3.1.2 Colocación del Operador de Proyección

Dado un conjunto de puntos en el espacio de posiciones:  $\{x_i\}$  con  $i = 1, 2, 3, \dots, N$ , la siguiente relación:

$$\psi_N(x_i) = P_N \psi(x_i) = \sum_{n=1}^N b_n \phi_n(x_i) = \psi(x_i) \quad (3.9)$$

está asociada a la **colocación**<sup>1</sup> del operador de proyección, y los puntos  $\{x_i\}$  son llamados **puntos de colocación**. Los coeficientes  $b_n$  están determinados por la condición de que:  $\psi_N(x) = \psi(x)$  en el conjunto de puntos de colocación.

## 3.2 BASE PSEUDO-ESPECTRAL

Una **base pseudo-espectral**:  $\{\theta_j\}_{j=1}^N$  se define como la base de las funciones localizadas espacialmente. La base de funciones ortogonales  $\{\phi_n\}_{n=1}^N$ , la colocación en la [Ecuación 3.9](#), los puntos de colocación  $\{x_i\}_{i=1}^N$ , y los factores de peso definidos como  $\Delta_j$  con  $j = 1, \dots, N$ , determinan completamente la forma de la base pseudo-espectral:

$$\theta_j(x) \equiv \sum_{n=1}^N \phi_n(x) \Phi_n^*(x_j) \quad (3.10)$$

donde:

$$\begin{aligned} \Phi_n(x_j) &\equiv \sqrt{\Delta_j} \phi_n(x_j) \\ \theta_j(x_i) &= \Delta_j^{-1/2} \delta_{ij} \end{aligned}$$

En conjunto, las  $N$  funciones de la base pseudo-espectral generan el mismo espacio de Hilbert reducido que las  $N$  funciones ortogonales de la base espectral, y ambas bases están relacionadas mediante una transformación unitaria:  $\Phi_n^*$ . Como consecuencia, se tiene que el operador de proyección es idéntico en ambas bases, es decir:

$$P_N = \sum_{n=1}^N |\phi_n\rangle \langle \phi_n| = \sum_{j=1}^N |\theta_j\rangle \langle \theta_j| \quad (3.11)$$

<sup>1</sup>Los **métodos de colocación** son soluciones numéricas de un conjunto de ecuaciones, cuya solución resulta ser exacta en un conjunto discreto de puntos llamados puntos de colocación.[15]

Se puede probar [15], que tanto la base espectral, como la base pseudo-espectral tienen propiedades de completos y ortogonalidad completamente análogas. Esto es de particular importancia, pues, dado que ambas bases generan el mismo espacio reducido de Hilbert, los operadores se pueden escribir tanto en términos de una base, como en la otra, y, si es necesario hacer cálculos con estos operadores (como sumarlos), se puede utilizar la transformación unitaria para escribirlos en términos de la misma base.

### 3.3 ALGORITMO DVR APLICADO A UN PROCESO FÍSICO-QUÍMICO

En esta sección se aplicarán los conceptos revisados a lo largo del capítulo para resolver un problema físico-químico que involucra potenciales dependientes del tiempo utilizando el método DVR.

La implementación numérica se realizó en Python 3.9.7 y se encuentra disponible en el repositorio: [Transferencia de Protones](#)

#### 3.3.1 Sistema de Transferencia de Protones

Los sistemas de **transferencia de protones** ocurren en un complejo de enlaces de hidrógeno:  $A-H \cdots A'$ . El modelo simplificado se muestra en la siguiente figura:

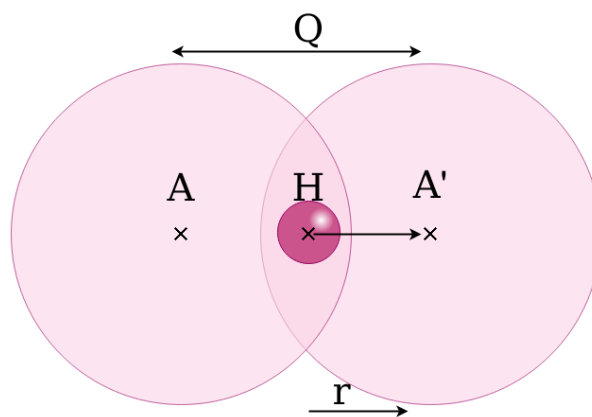


Figura 3.1: Descripción del modelo de transferencia de protones  $H^+$ .

En donde la coordenada  $Q$  hace referencia a la separación entre los átomos  $A$  y  $A'$ , mientras que la coordenada  $r$  es la distancia del protón al centro de los enlaces. [3].

En las descripciones teóricas de la transferencia de protones, a menudo el hidrógeno es representado moviéndose a través de un pozo de doble potencial unidimensional. [9]



A continuación se presenta un modelo particular de potencial para el sistema de transferencia de protones, en donde la descripción está dada por la coordenada del protón  $r \in [-1.5 \text{ \AA}, 1.5 \text{ \AA}]$ . A cada tiempo  $t$  el valor del potencial  $V(r, t)$  está dado por el eigenvalor más bajo de la siguiente matriz:

$$\begin{pmatrix} U_1(r, R(t)) & V \\ V & U_2(r, R(t)) + X(t) \end{pmatrix} \quad (3.12)$$

En donde  $U_1(r, R(t))$  y  $U_2(r, R(t))$  son potenciales de oscilador armónico, y  $V$  es una constante.

$$U_1(r, R(t)) = \frac{1}{2}m\omega_1^2 \left( r + \frac{R(t)}{2} \right) \quad (3.13)$$

$$U_2(r, R(t)) = \frac{1}{2}m\omega_2^2 \left( r - \frac{R(t)}{2} \right) \quad (3.14)$$

En las ecuaciones de potenciales de oscilador armónico,  $m$  se refiere a la masa del protón,  $\omega$  es la frecuencia del pozo de protones. Los términos  $U_1(r, R(t))$  y  $U_2(r, R(t))$  están desplazados mediante un término de energía dependiente del tiempo:  $X(t)$ , y un termino de distancia dependiente del tiempo:  $R(t)$ . La dinámica de  $X(t)$  corresponde a las fluctuaciones del entorno, mientras que  $R(t)$  representa las vibraciones de los sitios donantes y aceptores de protones. [14]

$$X(t) = \lambda \cos(\omega_x t + \theta_x) + X_{eq} \quad (3.15)$$

$$R(t) = (R_0 - R_{eq}) \cos(\omega_R t + \theta_R) + R_{eq} \quad (3.16)$$

La **Figura 3.2** muestra un ejemplo de potencial  $V(r, t)$  generado con la implementación de las ecuaciones anteriores. Los parámetros de potencial utilizados se muestran en la ??

### 3.3.2 Elección de una base Ortonormal y Grid

Para proceder con la solución a la **TDSE Ecuación 2.2**, elegimos una base de funciones ortogonales: [2]

$$\phi_n(r) = \sqrt{\frac{2}{b-a}} \sin \left( \frac{n\pi(r-a)}{b-a} \right) \quad n = 1, \dots, N \quad (3.17)$$

y un grid en el espacio de posiciones:

$$r_i = a + \frac{(b-a)i}{N-1} \quad i = 0, \dots, N-1 \quad (3.18)$$

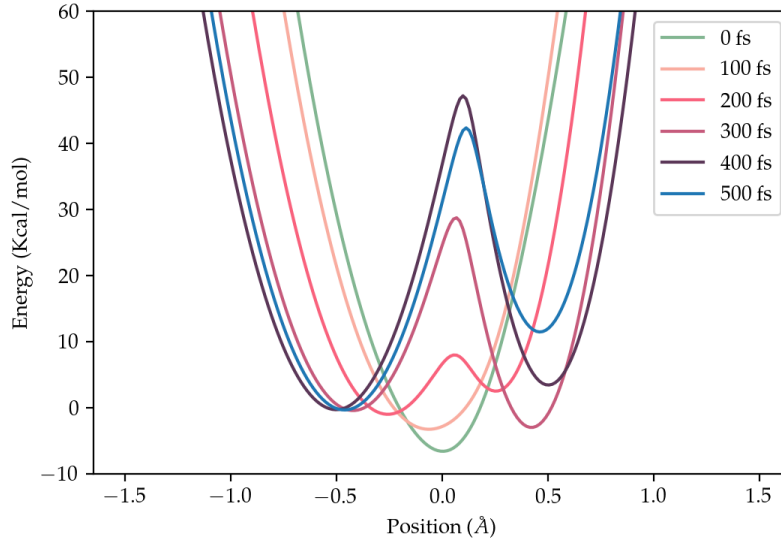


Figura 3.2: Potencial  $V(r, t)$  a diferentes tiempos.

donde:  $a = -1.5$ ,  $b = 1.5$  y  $N = 200$  para la implementación numérica, es decir, que el espacio reducido de Hilbert del sistema tiene una dimensión de  $N = 200$ . La [Figura 3.3](#) muestra las primeras cinco funciones ortogonales de la base en el grid del espacio de posiciones. Las funciones  $\phi_n$  están construidas para que se cumpla:  $\phi_n(r = r_0 = a) = \phi_n(r = r_{N-1} = b) = 0$ .

### 3.3.2.1 Representación de la matriz del Hamiltoniano en la Base Espectral

Los elementos de matriz de la Energía Cinética en la base pseudo-espectral están dados por:

$$T_{ij}^\theta = \langle \theta_i | T | \theta_j \rangle = \sum_{n=1}^N \langle \theta_i | \phi_n \rangle \langle \phi_n | T | \theta_j \rangle \quad (3.19)$$

realizando la aproximación:

$$\langle \theta_j | T | \phi_n \rangle \approx \langle r_j | T | \phi_n \rangle$$

y considerando que el operador  $T$  en el espacio de posiciones  $r$  está dado por:

$$T = -\frac{\hbar^2}{2m} \frac{d^2}{dr^2} \quad (3.20)$$

se puede obtener que: [\[15\]](#)[\[2\]](#)

$$T_{ij}^\theta \approx -\frac{\hbar^2}{2m} \Delta r \sum_{n=1}^N \phi_n(r_i) \frac{\partial^2 \phi_n}{\partial r^2} \Big|_{r_j} \quad (3.21)$$

donde:  $\Delta r = (b - a) / N - 1$ . Obteniendo la segunda derivada de la [Ecuación 3.17](#), y sustituyendo en la [Ecuación 3.21](#) se tiene:

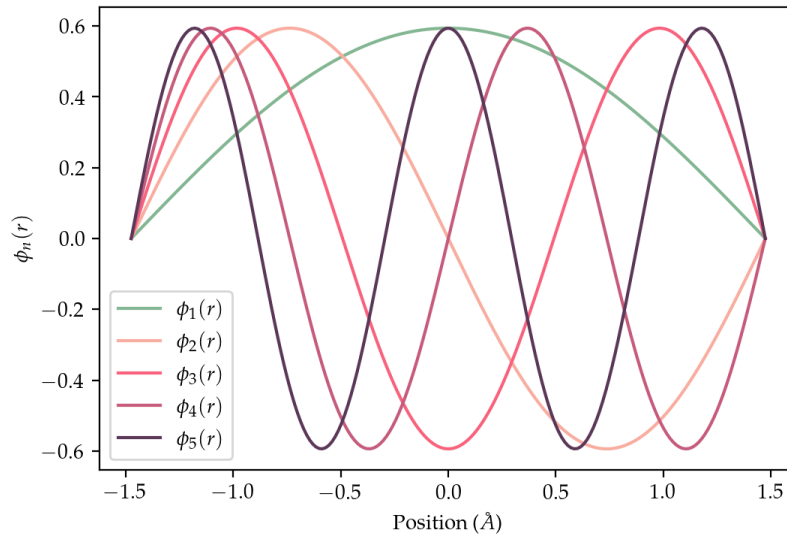


Figura 3.3: Funciones ortogonales  $\phi_n(r)$  en el grid  $\{r_i\}_{i=0}^{N-1}$  para  $n = 1, \dots, 5$ .

$$T_{ij}^\theta = \frac{\hbar^2}{2m} \left( \frac{\pi}{b-a} \right)^2 \frac{2}{N-1} \sum_{n=1}^N n^2 \sin \left( \frac{n\pi i}{N-1} \right) \sin \left( \frac{n\pi j}{N-1} \right) \quad (3.22)$$

que son los elementos de matriz de la Energía Cinética en la base pseudo-espectral,  $T^\theta \in \mathcal{M}_{200 \times 200}(\mathbb{R})$ .

La representación de la matriz de Energía Potencial en la base pseudo-espectral es más simple que la de la Energía Cinética, pues las funciones  $\{\theta_i\}$  son localizadas en el espacio de posiciones  $r$  y cumplen la condición de ortogonalidad:  $\langle \theta_i | \theta_j \rangle = \delta_{ij}$ , así:

$$V_{ij}^\theta = \langle \theta_i | V(\hat{r}) | \theta_j \rangle = V(r_i) \delta_{ij} \quad (3.23)$$

$V^\theta \in \mathcal{M}_{200 \times 200}(\mathbb{R})$  es una matriz diagonal.

Dado un tiempo  $t$ , los elementos de la diagonal  $V_{ii}^\theta$  están dados por  $V(r_i, t)$  (Ecuación 3.12), con  $i = 0, \dots, 31$ .

A partir de la Ecuación 3.22 y Ecuación 3.23, se construye la matriz del Hamiltoniano en la base pseudo-espectral al tiempo  $t$ :

$$H(t)^\theta = V(t)^\theta + T^\theta \quad (3.24)$$

en donde el Hamiltoniano tiene dependencia temporal debido a que el Potencial del sistema es dependiente del tiempo.

### 3.3.3 Propagación de un Paquete de Onda

Sea un paquete de onda al tiempo  $t = 0$  (Figura 3.4):

$$\psi(r, 0) = \sum_{i=1}^{k=5} C_i \cdot \phi_i(r) \quad (3.25)$$

con  $C_i$  números complejos aleatorios de una distribución uniforme, y elegidos de tal forma que:  $\langle \psi(r, 0) | \psi(r, 0) \rangle = 1$

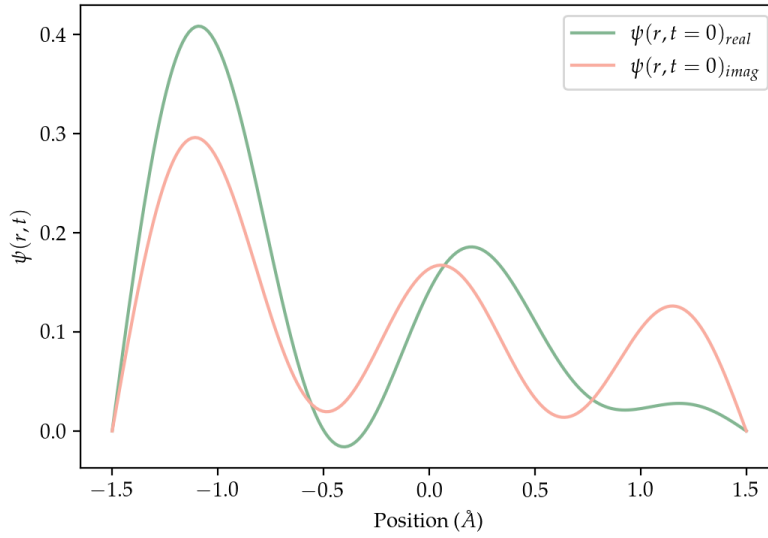


Figura 3.4: Paquete inicial de onda al tiempo  $t = 0$ . Parte real y compleja.

Si se toma un intervalo de tiempo  $\Delta t$  lo suficientemente pequeño como para que el potencial se pueda considerar constante en ese intervalo de tiempo ( $\approx 1$  fs para el sistema en cuestión), la evolución temporal del paquete de onda está dado por (Ecuación 2.3):

$$\psi(r, t) = \exp\{-iH^\theta t/\hbar\} \psi(r, 0) \quad (3.26)$$

Si  $H^\theta = UDU^{-1}$ , con  $D$  una matriz diagonal:

$$\psi(r, t) = \exp\{-iUDU^{-1}t/\hbar\} \psi(r, 0)$$

así,

$$\psi(r, t) = U \exp\left\{\frac{-it}{\hbar} D\right\} U^{-1} \psi(r, 0) \quad (3.27)$$

En donde la matriz  $U$  está formada por los  $N$  eigenvectores de  $H^\theta$  como vectores columna, y:

$$D = U^{-1}H^\theta U$$

La Figura 3.5 muestra la evolución de la parte real del paquete de onda en intervalos de  $1fs$  a lo largo de  $5fs$ , obtenida con la Ecuación 3.27. En la Figura 3.6 se muestra la evolución temporal de la densidad del protón.

► Evolución Temporal: Potencial y Densidad

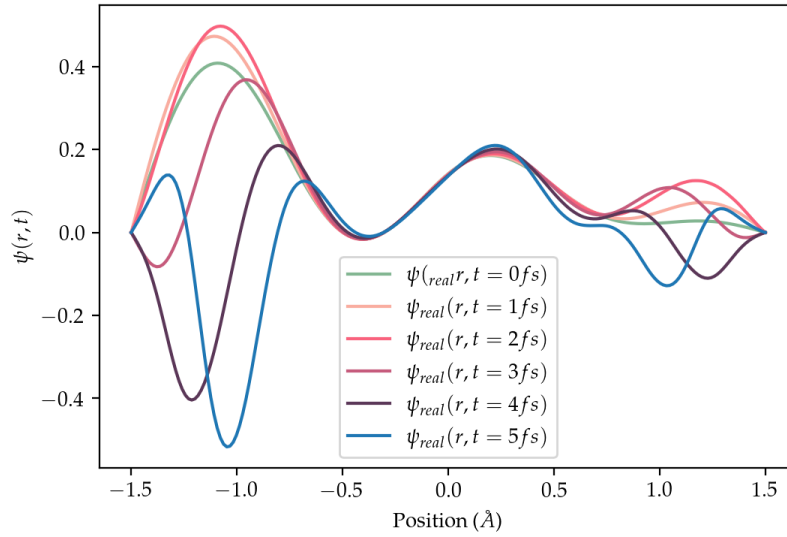


Figura 3.5: Propagación del paquete de onda  $\psi(r, t)$  parte real.

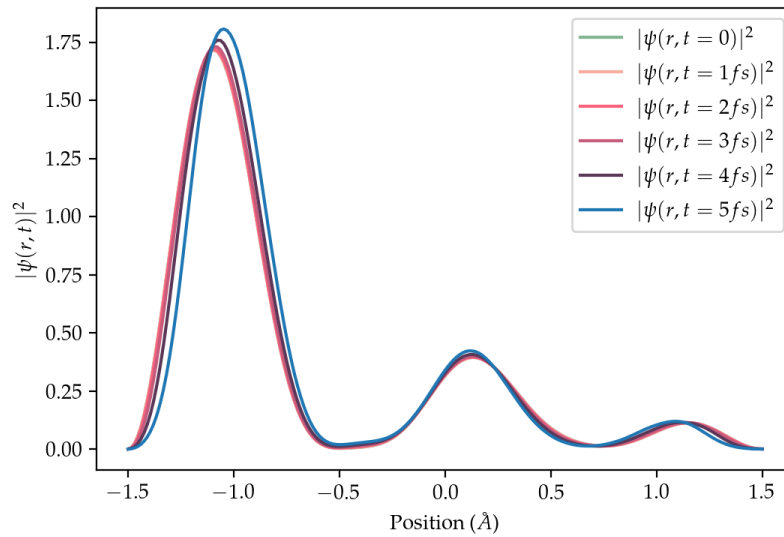


Figura 3.6: Evolución temporal de la densidad  $|\psi(r, t)|^2$ .

## Parte III

# REDES NEURONALES ARTIFICIALES (ANN'S)

# 4



## APRENDIZAJE AUTOMÁTICO

La **Inteligencia Artificial**, o AI por sus siglas en inglés, se puede definir como un sistema capaz de interactuar con su entorno. Algunos ejemplos de Inteligencias Artificiales son: *Siri* de Apple y *Alexa* de Amazon. Para poder generar una respuesta al entorno, estos sistemas contienen sensores que permiten la entrada de información, en estos ejemplos la información es obtenida mediante la voz o las palabras escritas de los usuarios, esta información es procesada a través de métodos de Aprendizaje Automático o ML por sus siglas en inglés.

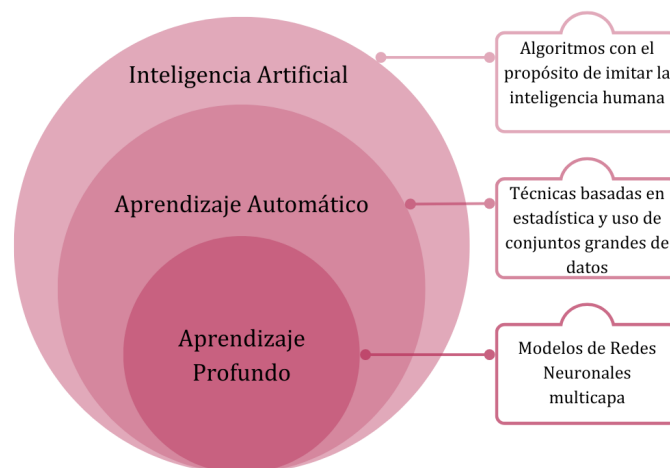


Figura 4.1: Diagrama de Venn que muestra los conceptos de Inteligencia Artificial, Aprendizaje Automático y Aprendizaje profundo.

Los métodos de **Aprendizaje Automático** se pueden definir como un conjunto de métodos que pueden detectar automáticamente patrones en datos y aplicarlos para predecir nuevos datos. A grandes rasgos, los métodos de aprendizaje automático pueden dividirse en dos categorías: aprendizaje supervisado y aprendizaje no supervisado. [10]

## 4.1 APRENDIZAJE SUPERVISADO

En el **aprendizaje supervisado** el objetivo es aprender, a partir de un conjunto de  $M$  datos de entrenamiento definidos como:

$$\{\vec{x}_i, y_i\}_{i=1}^M \quad (4.1)$$

una forma de mapear  $\vec{x}_i$  a  $y_i$ .

Cada vector:

$$\vec{x}_i = (x_1, x_2, \dots, x_d)_i \quad (4.2)$$

corresponde a un dato, en donde cada componente es una **característica** o **atributo** del dato en cuestión, y el número de componentes depende del problema. Algunos ejemplos de  $\vec{x}_i$  son:

- Imágenes
- Enunciados de texto
- Audios de voz
- Series de tiempo

Por otro lado,  $y_i$  corresponde a la **etiqueta** de  $\vec{x}_i$ . Cuando  $y_i$  puede tomar un valor categórico, es decir, que de un conjunto finito:

$$y_i \in \{1, \dots, c\}, \text{ por ejemplo: } 1=\text{perro}, 2=\text{gato}, \text{ etc.}$$

se dice que se trata de un problema de **clasificación**. Por otro lado, si  $y_i$ <sup>1</sup> es un valor real, se dice que el problema es de **regresión**.

Algunos ejemplos de métodos de aprendizaje supervisado son:

- Árboles de decisión: Para problemas de clasificación
- Regresión lineal: Para problemas de regresión
- Redes Neuronales: Para problemas de regresión y de clasificación

---

<sup>1</sup> $y_i$  puede ser también un vector:  $\vec{y}_i$ , en donde su dimensión está determinada por el problema a resolver, y no necesariamente es igual a la de  $\vec{x}_i$ .



## 4.2 APRENDIZAJE NO SUPERVISADO

En el **Aprendizaje No Supervisado** el conjunto de entrenamiento de  $M$  datos se reduce a:

$$\{\vec{x}_i\}_{i=1}^M$$

en donde no se cuenta con una etiqueta  $y_i$ . En este tipo de problemas, los métodos están enfocados en buscar patrones importantes a partir de únicamente los datos de entrada.

Algunos ejemplos de métodos de aprendizaje no supervisado son:

- Clustering: Agrupar datos similares entre sí
- Análisis de Componentes Principales: Buscar la relación entre las características de los datos y reducir su dimensionalidad

El **aprendizaje por refuerzo** es comúnmente considerado una tercer categoría de aprendizaje automático, de manera general, en estas técnicas el sistema inteligente interactúa con su entorno con el objetivo de obtener recompensas, bajo el contexto en el que se está implementando, a menudo se utiliza este tipo de aprendizaje para enseñar a las máquinas cómo jugar video juegos. [16]

## 4.3 APRENDIZAJE PROFUNDO

El **Aprendizaje Profundo**, o **Deep Learning** en inglés, como se muestra en la **Figura 4.1**, es un sub-campo del aprendizaje automático, que puede definirse como aquellos métodos que procesan la información en múltiples capas, en donde el nivel de abstracción y complejidad de la información incrementa conforme avanza de una capa a la siguiente.

En la práctica, los modelos de aprendizaje profundo son redes neuronales artificiales multicapa (**Capítulo 5**). Algunos ejemplos comúnmente utilizados son: [16]

- Perceptrones Multicapa § 5.2.4
- Redes Neuronales Convolucionales
- Redes Neuronales Recurrentes § 6.2

En el siguiente capítulo se abordarán conceptos básicos acerca de las redes neuronales artificiales, cómo están construidas, es decir, su arquitectura; y cómo, de manera general, este tipo de algoritmos aprenden con base al procesamiento de múltiples datos o muestras de ejemplo.

# 5



## REDES NEURONALES ARTIFICIALES

### 5.1 INTRODUCCIÓN

En el capítulo anterior se abordaron conceptos básicos del aprendizaje automático, así como dos de sus principales enfoques: aprendizaje supervisado y no supervisado. Las Redes Neuronales Artificiales o **ANNs** por sus siglas en inglés, son comúnmente un método de aprendizaje supervisado<sup>1</sup>, que han tenido gran impulso en los últimos años debido al incremento de datos y a su fácil acceso, así como al crecimiento en poder computacional.

Las **Redes Neuronales Artificiales** son algoritmos de aprendizaje automático que simulan<sup>2</sup> el mecanismo de aprendizaje de los organismos biológicos, en donde las neuronas, es decir, las células del sistema nervioso, se conectan unas con otras mediante las dendritas y los axones en la región espacial nombrada sinapsis **Figura 5.1**, en donde las conexiones sinápticas a menudo cambian en respuesta a estímulos externos del organismo, proceso que a grandes rasgos es como aprenden los seres vivos. [11]

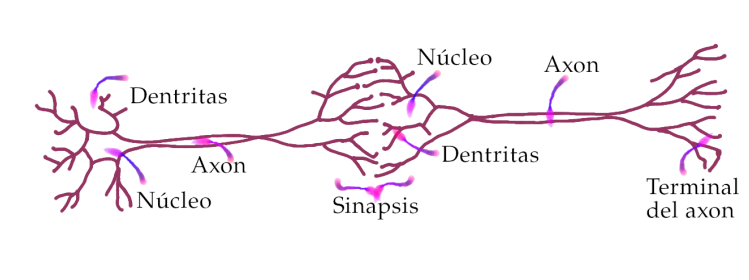


Figura 5.1: Neuronas biológicas transmitiendo información [5]

<sup>1</sup>Las Redes Neuronales Artificiales son algoritmos que también se pueden aplicar en métodos de aprendizaje no supervisado.

<sup>2</sup>Las Redes Neuronales Artificiales a menudo son consideradas más como una caricatura de las biológicas, pues su complejidad rebasa el entendimiento e interpretación que se le puede dar a través de los modelos de **ANNs**.

## 5.2 ARQUITECTURA BÁSICA

### 5.2.1 El Perceptrón

La **Figura 5.2** muestra un diagrama de un modelo de Perceptrón, que es el modelo más simple de las **ANNs**. En este ejemplo la **capa de entrada** tiene dos componentes:

$$(x_1, x_2)$$

Es decir,  $d = 2$  en la **Ecuación 4.2**. La constante 1 es añadida para asignarle el **sesgo**:  $b \in \mathbb{R}$ , un parámetro que a menudo se emplea por razones de estadística. La **neurona** es la unidad computacional que calcula:

$$f \left( \sum_{i=1}^d w_i \cdot x_i + b \right) \quad (5.1)$$

donde los  $w_i \in \mathbb{R}$  son conocidos como los **pesos** de  $x_i$ , y  $f$  es una función § 5.2.2.

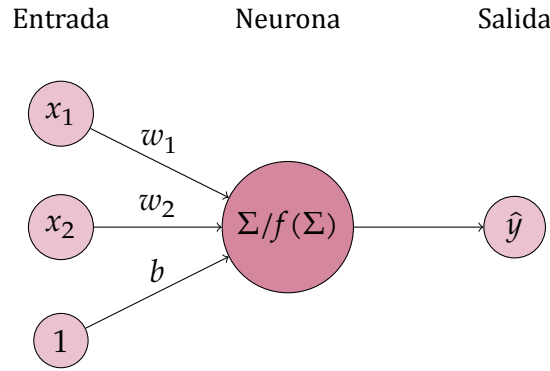


Figura 5.2: Diagrama de un modelo simple de Perceptrón

El objetivo del algoritmo, es que mediante el entrenamiento de la red § 5.3 con un conjunto grande de  $M$  datos:  $\{\vec{x}_j, y_j\}_{j=1}^M$  se obtengan los  $w_i$  óptimos para que se cumpla:

$$y_l = \hat{y}_l = f \left( \sum_{i=1}^d w_i \cdot x_l \right)$$

con  $y_l$  la etiqueta, o valor esperado, correspondiente al vector  $\vec{x}_l$ , en donde  $(\vec{x}_l, y_l)$  es un dato que no necesariamente pertenece al conjunto de entrenamiento que utilizó la red. Esta última propiedad es llamada **generalización del modelo**, y es lo que permite hacer predicciones sobre nuevos datos.

*Para simplificar la lectura, en las siguientes secciones y capítulos se utilizará la palabra “red” para hacer referencia a la red neuronal artificial o ANN.*

### 5.2.2 Función de Activación

En la [Ecuación 5.1](#) la función  $f$  es conocida como una **función de activación**. Algunos ejemplos de funciones de activación comúnmente utilizadas se muestran en la [Figura 5.3](#).

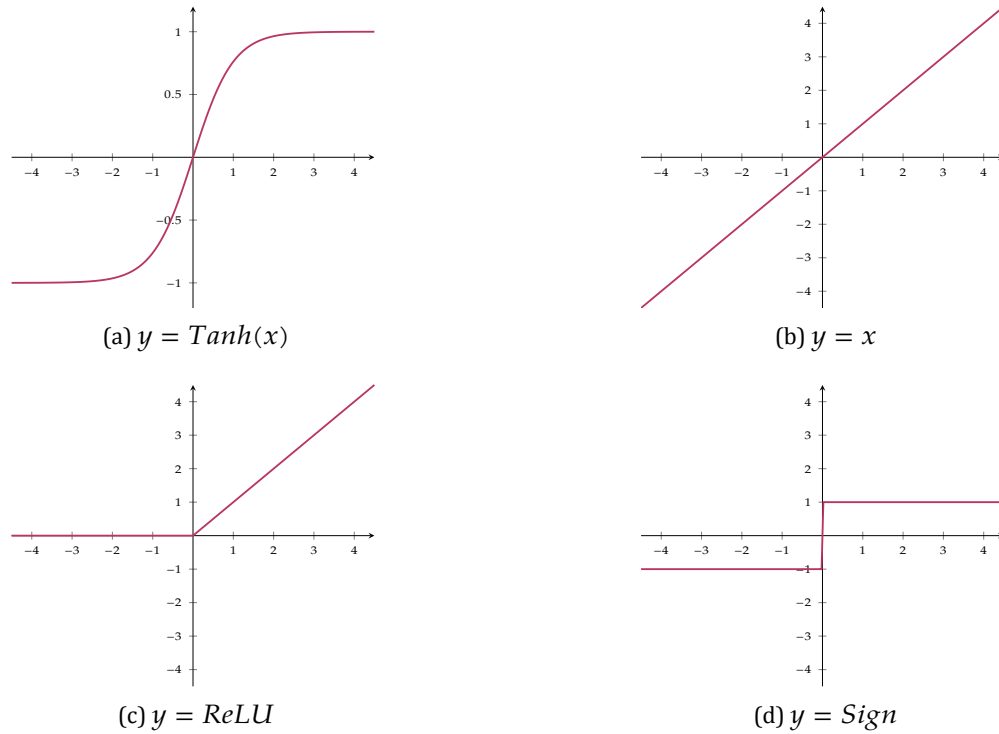


Figura 5.3: Ejemplos de funciones de activación

La importancia en la elección de una función de activación u otra es más notable en los siguientes casos:

1. En la capa de salida de cualquier modelo de red
2. En los modelos multicapa [§ 5.2.4](#)

En el primer caso, la función de activación está motivada por el tipo de valores que puede tomar la etiqueta  $y$ . En problemas de clasificación binaria, las funciones de activación comúnmente utilizadas son *Tanh* o *Sigmoid*, en problemas de clasificación múltiple la función *Softmax*; cuando  $y$  toma valores reales la función de activación utilizada es la *Identidad* [Figura 5.3b](#).

En el segundo caso, la importancia de elegir una función de activación no lineal entre capas para modelos de clasificación, por ejemplo, permite que el modelo pueda transformar un problema que inicialmente no es linealmente separable, a uno que sí lo es.

### 5.2.3 Función de Pérdida

La **función de pérdida**:  $L(\hat{y}, y)$  es la función que compara el resultado o salida de la red:  $\hat{y}$  respecto a la etiqueta original  $y$  correspondiente a la entrada  $\vec{x}$ , en otras palabras, indica qué tan bueno es el modelo implementado. El modelo es mejor cuanto menos sea la diferencia entre el valor predicho  $\hat{y}$  respecto al valor esperado  $y$ .

En el entrenamiento de una red, el objetivo es minimizar esta función a través de la actualización de los pesos  $w_i$ . La elección de la función de pérdida, depende del problema a resolver, como en el caso de la función de activación de la última capa. Para modelos en donde los valores de la salida  $\hat{y}$  son reales, se puede utilizar una *función de pérdida cuadrática*:

$$L = (\hat{y} - y)^2 \quad (5.2)$$

Para problema de clasificación binaria se puede utilizar:

$$L = \log(1 + \exp(-y \cdot \hat{y})) \quad (5.3)$$

mientras que para problemas de múltiples categorías, donde  $\hat{y} = (\hat{y}_1, \dots, \hat{y}_k)$  son las probabilidades<sup>3</sup> de las  $k$  clases<sup>4</sup> una función de pérdida que se puede aplicar es:

$$L = -\log(\hat{y}_r) \quad (5.4)$$

con  $r$  la clase a la que corresponde la etiqueta  $y$ .

### 5.2.4 Modelos Multicapa

En la sección § 5.2.1 se revisó el Perceptrón como el modelo de red más simple. La Figura 5.4 muestra un diagrama de un modelo multicapa, estos modelos tienen capas ocultas entre las capas de entrada y salida, cada capa contiene un número determinado de neuronas o nodos. El número de capas ocultas y los nodos en cada una, son **hiper-parámetros** de la red, y en general<sup>5</sup>, no se tiene una manera determinista para conocer los valores óptimos en cada modelo.

La arquitectura específica de las redes multicapa se conoce como redes *feed-forward*, pues las capas se alimentan una tras otra desde la capa de entrada hasta la capa de

<sup>3</sup>Las probabilidades de cada clase se pueden obtener aplicando la función de activación *softmax*.

<sup>4</sup>Las **clases** se refieren a las diferentes opciones de salida que puede tener una red de un problema de clasificación múltiple, por ejemplo si la red reconoce tres tipos de fruta, cada fruta es una clase distinta.

<sup>5</sup>*Hyperparameter Grid Search* es un método que se puede utilizar para conocer los hiper-parámetros óptimos de un modelo en particular, sin embargo, este proceso generalmente implica costos computacionales mayores.

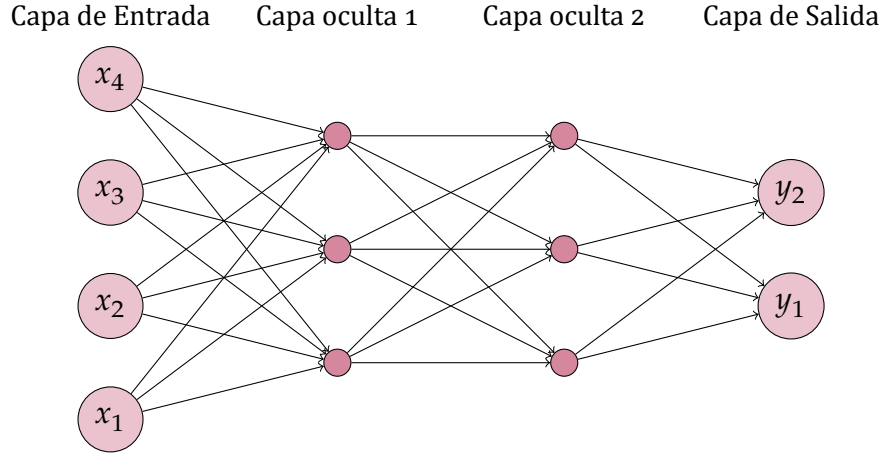


Figura 5.4: Diagrama de un modelo de ANN Multicapa con dos capas ocultas.

salida. La arquitectura por defecto de las redes feed-forward supone que todos los nodos de una capa están conectados a los de la capa siguiente. [11]

La operación que realiza una red multicapa de la capa de entrada a la primer capa oculta con  $p_1$  nodos está definida como:

$$\vec{h}_1 = f(W_1^T \cdot \vec{x})$$

onde  $W_1^T$  es la transpuesta de  $W_1$ , de dimensión  $p_1 \times d$ , y  $\vec{x}$  el vector de entrada de dimensión  $d$ . La operación que se realiza de la capa oculta  $p$  y la  $p + 1 \forall p \in \{1, \dots, k - 1\}$  es:

$$\vec{h}_{p+1} = f(W_p^T \cdot \vec{h}_p)$$

donde  $W_p^T$  es de dimensión  $p_{r+1} \times p_r$  y  $\vec{h}_p$  es de dimensión  $p$ . Finalmente, de la última capa oculta a la capa de salida con dimensión  $o$  la operación realizada es:

$$\vec{o} = f(W_{k+1}^T \cdot \vec{h}_k)$$

con  $W_{k+1}^T$  de dimensión  $o \times p_k$ , y  $\vec{h}_k$  de dimensión  $p_k$ . En todas las expresiones  $f$  es una función de activación que se aplica a cada elemento del vector.

### 5.3 ENTRENAMIENTO DE UNA RED NEURONAL

Una vez construida la arquitectura de una red, el objetivo es obtener los parámetros  $w$  y  $b$  de cada nodo y en cada capa, que minimicen la función de pérdida  $L$ . El algoritmo que se utiliza para minimizar la función de pérdida es conocido como **Descenso del Gradiente Estocástico**<sup>6</sup> o SGD por sus siglas en inglés. La Figura 5.5 muestra un sencillo ejemplo del funcionamiento del algoritmo, en donde la función

<sup>6</sup>poner algo del Descenso de Gradiente

de pérdida únicamente depende de un parámetro  $w$ , el peso inicial es un valor aleatorio  $w_1$ , luego se calcula el valor del gradiente (en este caso unidimensional, la derivada) y se actualiza el peso como:

$$w_2 = w_1 - \alpha \frac{dL}{dw} \Big|_{w_1}$$

en donde  $\alpha$  es un hiper-parámetro de la red llamado **tasa de aprendizaje** o **learning rate** en inglés. El proceso continúa actualizando  $w$  hasta llegar a  $w_m$ , el mínimo de la función de pérdida, pues el algoritmo actualiza los pesos en dirección opuesta al gradiente, que apunta a la dirección de máximo cambio.

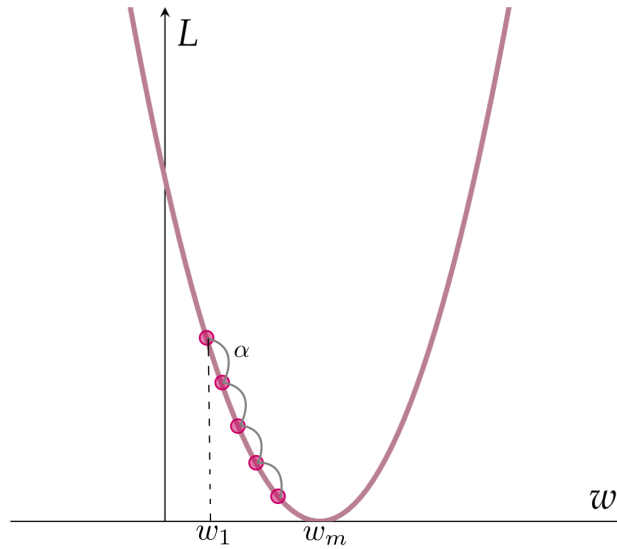


Figura 5.5: Algoritmo de Descenso del Gradiente para una función de pérdida  $L(w)$

Para modelos de una capa el proceso del Descenso del Gradiente es directo, pues la función de pérdida depende de los pesos de la única capa, sin embargo, para modelos multicapa la función de pérdida es el resultado de una composición de múltiples funciones de los pesos de las capas anteriores, el algoritmo utilizado para calcular el gradiente en función de la composición se llama **Retropropagación** o **Backpropagation** en inglés, que se compone de dos fases principales:

1. Fase de avance: La red recibe las entradas con pesos establecidos de manera aleatoria, se calcula la función de pérdida y las derivadas respecto a la última capa, es decir, la capa de salida.
2. Fase hacia atrás: En esta fase el objetivo es conocer el gradiente de la función de pérdida respecto a los pesos de las capas anteriores utilizando la regla de la cadena.

La Figura 5.6 muestra un ejemplo de la composición de funciones que se realizan en una red neuronal. En este ejemplo existen se muestran dos capas antes de la salida  $o$ .

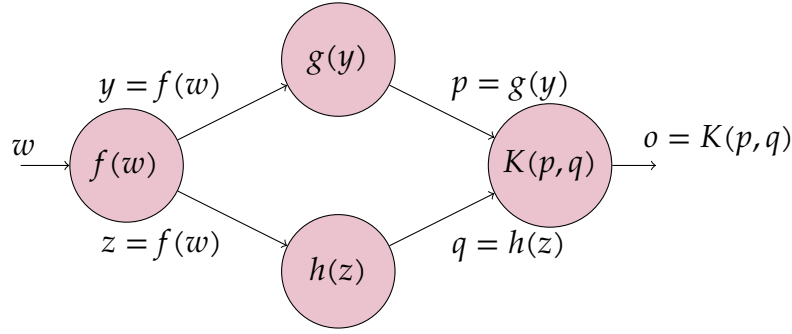


Figura 5.6: Gráfica computacional con dos posibles caminos [11]

Utilizando la regla de la cadena para funciones de varias variables se puede determinar la derivada de  $o$  respecto al peso  $w$ :

$$\frac{\partial o}{\partial w} = \frac{\partial o}{\partial p} \cdot \frac{\partial p}{\partial w} + \frac{\partial o}{\partial q} \cdot \frac{\partial q}{\partial w} \quad (5.5)$$

$$= \frac{\partial o}{\partial p} \cdot \frac{\partial p}{\partial y} \cdot \frac{\partial y}{\partial w} + \frac{\partial o}{\partial q} \cdot \frac{\partial q}{\partial z} \cdot \frac{\partial z}{\partial w} \quad (5.6)$$

$$= \frac{\partial K(p, q)}{\partial p} \cdot g'(y) \cdot f'(w) + \frac{\partial K(p, q)}{\partial q} \cdot h'(z) \cdot f'(w) \quad (5.7)$$

En este ejemplo existen únicamente dos caminos posibles de  $w$  a  $o$ . En la práctica, las redes neuronales multicapa contienen una gran cantidad de caminos posibles entre los pesos de cada capa hasta la salida. Si se considera una secuencia de capas ocultas:  $h_1, \dots, h_k$  seguida de la capa de salida  $o$  respecto a la cual se calcula la función de pérdida  $L$ . La parcial de la función de pérdida respecto al peso  $w_{h_r, h_{r+1}}$  de la conexión de la capa  $h_r$  a la capa  $h_{r+1}$  es: [11]

$$\frac{\partial L}{\partial w_{h_r, h_{r+1}}} = \frac{\partial L}{\partial o} \cdot \left( \sum_{(h_r, h_{r+1}, \dots, h_k, o) \in \mathcal{D}} \frac{\partial o}{\partial h_k} \prod_{i=r}^{k-1} \frac{\partial h_{i+1}}{\partial h_i} \right) \frac{\partial h_r}{\partial w_{h_{r-1}, h_r}} \quad (5.8)$$

una vez conocidos los gradientes respecto a los pesos en las múltiples capas, se actualizan los valores  $w$  con el objetivo de minimizar la función de pérdida, y así mejorar el modelo.

El número de datos de entrenamiento, es decir de pares:  $(X_l, y_l)$ , que pasan por la red antes de que se actualicen los pesos, se llama **tamaño de lote** o **batch size**. Una vez que todos los datos de entrenamiento han pasado por la red, se dice que ha pasado una **época** o **epoch**. Tanto el número de épocas, como el tamaño de lote son hiper-parámetros de la red.



## Parte IV

# IMPLEMENTACIÓN DE UN MODELO DE ANN COMO PROPAGADOR EN DINÁMICA CUÁNTICA

# 6



## LSTM COMO PROPAGADOR EN DINÁMICA CUÁNTICA

### 6.1 INTRODUCCIÓN

En el capítulo anterior se desarrollaron conceptos básicos generales de las redes neuronales, actualmente existen diversos tipos de redes que se aplican a problemas comunes y que se distinguen por su arquitectura.

En este capítulo se abordará un modelo particular de red llamado *Long Short-Term Memory* (LSTM, por sus siglas en inglés), que pertenece a un tipo de redes llamadas *Redes Neuronales Recurrentes* (Red Neuronal Recurrente (Recurrent Neural Network)s (RNNs), por sus siglas en inglés). Este tipo de redes es comúnmente utilizada cuando los atributos en el conjunto de datos están relacionados entre sí, o guardan una dependencia.

En la sección § 6.3 se desarrollará la arquitectura de las redes LSTM, así como sus aplicaciones más comunes, haciendo énfasis a las series de tiempo, que son de particular interés en el presente trabajo, pues la evolución temporal de un paquete de onda puede tratarse como un problema de series de tiempo, encontrando así una manera de abordar el problema de propagar una onda en el tiempo utilizando una red LSTM.

En la sección § 6.4 se implementará el modelo de LSTM que servirá como propagador  $\mathcal{U}$  en la ecuación de Schrödinger Dependiente del Tiempo (Ecuación 2.2) para los potenciales  $V(r, t)$  revisados en la sección § 3.3.1.

A lo largo de la sección se presentará el conjunto de datos de entrenamiento de la red, así como los parámetros del modelo del potencial que se utilizaron para su obtención; también se presentarán los hiper-parámetros empleados en la red. Finalmente, en la sección ?? se presentarán los resultados y predicciones obtenidas con del modelo.

## 6.2 REDES NEURONALES RECURRENTES

Como se mencionó anteriormente, las Redes Neuronales Recurrentes se emplean cuando entre los atributos del conjunto de datos existen relaciones. Un dato de entrada para este tipo de redes puede verse de la siguiente forma:

$$\vec{X} = (\vec{x}_0, \vec{x}_1, \dots, \vec{x}_n) \quad (6.1)$$

en donde cada  $\vec{x}_t$  es un vector de dimensión  $d$  recibido al tiempo  $t$ . Para cada tiempo  $t$  se tiene un valor de salida:  $\vec{y}_t$  con la misma dimensión.

Algunos ejemplos de aplicaciones se muestran a continuación:

- *Secuencias de texto*: En donde la importancia o información de una palabra depende del contexto en el que se encuentra dentro de una oración. Cada palabra<sup>1</sup>:  $\vec{x}_t$ , puede verse como un atributo relacionado a la palabra anterior en una oración:  $\vec{x}_{t-1}$ , o a la palabra siguiente:  $\vec{x}_{t+1}$ .
- *Series de tiempo*:  $\vec{x}_t$  en este caso puede almacenar  $d$  valores, que pueden ser, por ejemplo, los correspondientes al valor de una función  $f(x, t)$  en una maya de  $d$  puntos en el espacio  $x$  al tiempo  $t$ . En este tipo de configuración la salida  $\vec{y}_t$  podría ser la predicción pronosticada de  $\vec{x}_{t+1}$ .

Un diagrama sencillo de una RNN se muestra en la Figura 6.1, en donde un auto-bucle actualizará el **estado oculto**  $\vec{h}_t$  a cada tiempo  $t$  después de la entrada  $\vec{x}_t$ :

$$\vec{h}_t = \text{Tanh}(W_{xh}\vec{x}_t + W_{hh}\vec{h}_{t-1}) \quad (6.2)$$

donde  $\text{Tanh}$  es la función de activación, que está siendo aplicada a cada elemento del vector resultante de la multiplicación y suma de las matrices de peso  $W$  por los vectores  $\vec{x}_t$  y  $\vec{h}_{t-1}$  para cada tiempo  $t$ . La salida, por su parte, está dada por:

$$\vec{y}_t = W_{hy}\vec{h}_t \quad (6.3)$$

En la Figura 6.1, a la derecha se muestra la representación de la RNN en capas para cada tiempo, en donde es importante notar que las matrices de peso  $W$  son las mismas en cada tiempo  $t$ , de manera que ecuación de recurrencia Ecuación 6.2 aplica la misma función.

Uno de los principales problemas de las RNNs en la práctica es en el entrenamiento, pues el gradiente tiende a desvanecerse o incrementar excesivamente, debido a la inestabilidad que produce la multiplicación de forma recursiva de las matrices

<sup>1</sup>En este caso, la dimensión  $d$  hace referencia al número de palabras que tiene el léxico utilizado.

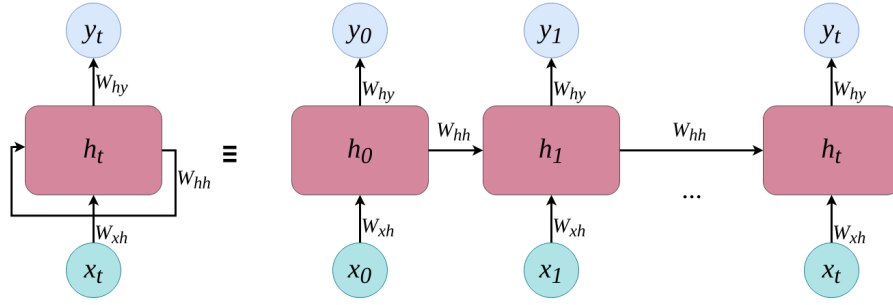


Figura 6.1: Diagrama de una RNN y su representación en capas de tiempo.

de peso  $W$ , problema que puede incrementarse cuanto más grande sea la secuencia de tiempo que procesa la red y cuantas más capas ocultas tenga.

Una alternativa que previene este problema es el cambio de la relación de recurrencia para el estado oculto  $\vec{h}_t$  (Ecuación 6.2) mediante el uso de la *memoria a largo plazo* de una red LSTM, que se abordará en la siguiente sección. [11]

### 6.3 REDES LONG-SHORT TERM MEMORY

Las redes Long-Short Term Memory son un tipo de RNN diseñadas para mitigar los problemas con el gradiente que estas tienen en el entrenamiento. La manera en la que realizan esto es cambiando las condiciones de recurrencia de cómo el estado oculto  $\vec{h}_t$  es propagado en el tiempo, para ello, se introduce un nuevo vector llamado **celda de estado**:  $\vec{c}_t$  de la misma dimensión de  $\vec{h}_t$ , que puede interpretarse como una *memoria a largo plazo* que retiene parte de la información de las celdas de estados de tiempos pasados. [11]

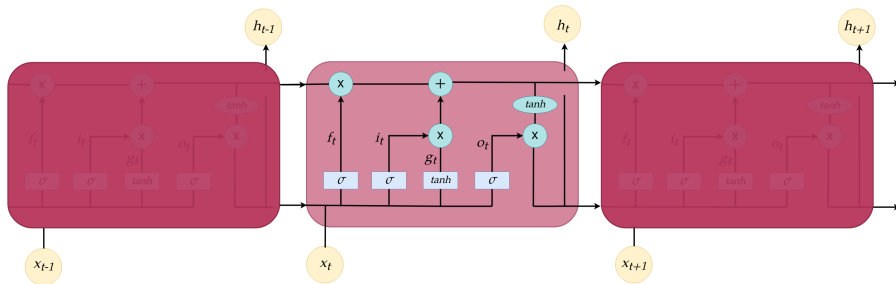


Figura 6.2: Diagrama de una LSTM representada en capas de tiempo.

El diagrama de la Figura 6.2 muestra la estructura de una LSTM desplegada en capas de tiempo, su forma en cadena es igual a la de las RNNs mencionadas en la sección anterior; sin embargo, cada bloque de memoria (recuadros rosas) tiene una arquitectura particular más compleja que la de una RNN común.

### 6.3.1 Arquitectura

La Figura 6.3 muestra un bloque de memoria de una **LSTM**, en donde se observa cómo interactúan los vectores de estado oculto y de celda de estado pasados:  $\vec{h}_{t-1}$  y  $\vec{c}_{t-1}$  con funciones llamadas compuertas:  $\vec{f}_t$ ,  $\vec{i}_t$ ,  $\vec{o}_t$  y operaciones básicas como suma y multiplicación, para producir nuevos vectores  $\vec{h}_t$  y  $\vec{c}_t$ . Al tiempo  $t = 0$ , los vectores de estado oculto y de celda de estado comienzan inicializados con un valor aleatorio.

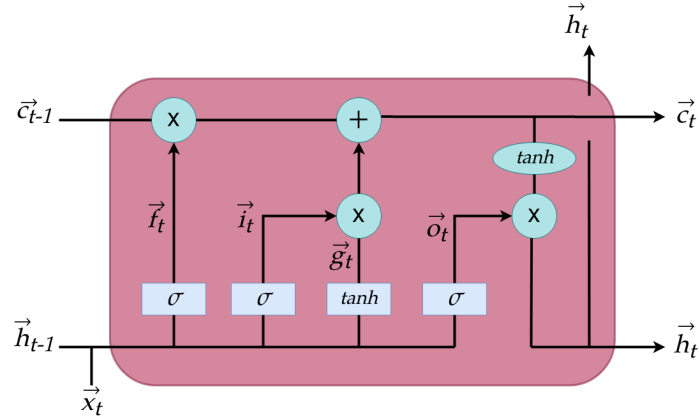


Figura 6.3: Diagrama de un Bloque de Memoria de una **LSTM**.

Una compuerta llamada **forget**  $\vec{f}_t$ , decide qué información proveniente del tiempo anterior:  $t - 1$  se mantiene y cual no, tomando en consideración también la entrada  $\vec{x}_t$ , aplicando la función  $\sigma$  (es decir,  $\vec{f}_t$  es una capa *Sigmoid*):

$$\vec{f}_t = \sigma(W_{xf}\vec{x}_t + W_{hf}\vec{h}_{t-1} + b_f) \quad (6.4)$$

En donde  $W$  y  $b$  con sus correspondientes subíndices representan las matrices de peso y vectores de sesgo correspondientes a cada capa oculta. Posteriormente, para decidir qué nueva información será incluida a la celda de estado se calculan:

$$\vec{i}_t = \sigma(W_{xi}\vec{x}_t + W_{hi}\vec{h}_{t-1} + b_i) \quad (6.5)$$

$$\vec{g}_t = \text{Tanh}(W_{xg}\vec{x}_t + W_{hg}\vec{h}_{t-1} + b_g) \quad (6.6)$$

en donde  $\vec{i}$  es una capa *Sigmoid*, llamada compuerta **input**, y  $\vec{g}$  es una capa *Tanh* que genera un vector con valores nuevos que son candidatos para ser agregados a la nueva celda de estado  $\vec{c}_t$ .

La actualización de la celda de estado se da como:

$$\vec{c}_t = \vec{f}_t * \vec{c}_{t-1} + \vec{i}_t * \vec{g}_t \quad (6.7)$$

en donde  $*$  representa el *producto Hadamard*, en donde la multiplicación de los vectores se realiza elemento por elemento. Finalmente, la salida esta dada por:

$$\vec{o}_t = \sigma(W_{xo}\vec{x}_t + W_{ho}\vec{h}_{t-1} + b_o) \quad (6.8)$$

$$\vec{h}_t = \vec{o}_t * \text{Tanh}(\vec{c}_t) \quad (6.9)$$

en donde  $\vec{o}$  es una capa *Sigmoid* llamada compuerta **output**, y el vector  $\vec{h}_t$  es el estado oculto al tiempo  $t$ , que servirá como entrada, junto con el vector de celda de estado  $\vec{c}_t$  para el bloque de memoria del tiempo  $t + 1$ , como se muestra en la Figura 6.2.

## 6.4 PROPAGACIÓN TEMPORAL DE FUNCIONES DE ONDA CON LSTM

En esta sección se implementa una red **LSTM** para predecir la evolución temporal de un paquete de onda a un tiempo  $t$   $\psi(r, t)$  bajo un potencial dependiente del tiempo  $V(r, t)$  después de un intervalo de tiempo  $\Delta t$ :  $\psi(r, t + \Delta t)$ . Tomando como referencia un trabajo previo [14] en donde se implementó un modelo de perceptrón multicapa.

### 6.4.1 Obtención de Datos

El conjunto de datos de entrenamiento utilizado se obtuvo generando tres mil trayectorias de  $200 fs$ , cada una con un  $\Delta t = 1 fs$ . El diagrama de la Figura 6.4 muestra una trayectoria, en donde  $\psi(r, t_l)r$  con  $l = 0, \dots, 200$  representa la parte real del paquete de onda,  $\psi(r, t_l)i$  la parte imaginaria y  $V(r, t_l)$  el potencial.

Para cada trayectoria, los paquetes de onda iniciales, es decir al tiempo  $t = 0$ , se definieron como ondas Gaussianas: (Figura 6.5)

$$\psi(r, 0) = C_i \cdot \frac{1}{\sigma\sqrt{2\pi}} \exp\left\{-\frac{(r - \mu)^2}{2\sigma^2}\right\} \quad (6.10)$$

en donde  $\mu$  y  $\sigma$  son valores aleatorios elegidos de una distribución uniforme de  $(-0.5, 0.5)\text{\AA}$  y  $(0.1, 0.3)\text{\AA}$  respectivamente.  $C_i$  es un número complejo aleatorio elegido de tal manera que la onda esté normalizada, es decir:

$$\langle \psi(r, 0) | \psi(r, 0) \rangle = 1$$

Para propagar la onda se utilizó el método **DVR** revisado en la sección § 3.3 con un grid de  $N = 32$  puntos, con  $a = r_0 = -1.5\text{\AA}$  y  $b = r_{N-1} = 1.5\text{\AA}$  (Ecuación 3.18).

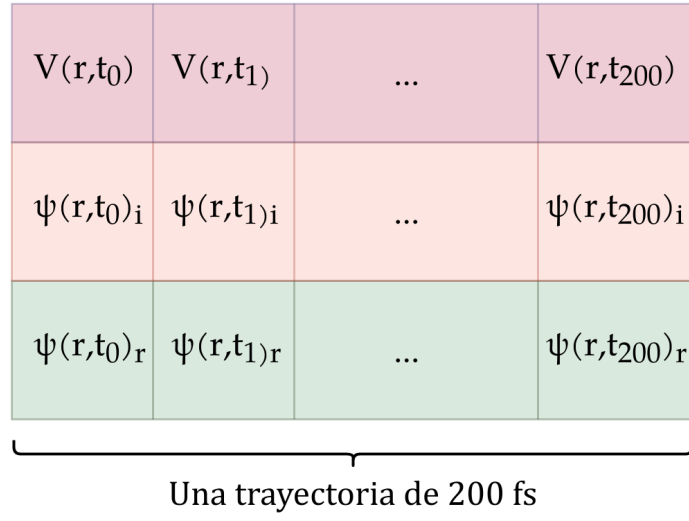


Figura 6.4: Diagrama de una trayectoria generada.

Para cada tiempo  $t_i$  la onda  $\psi(r, t_i)$  contiene el valor de la onda en los  $N = 32$  puntos en el grid del espacio de posiciones en  $r$ .

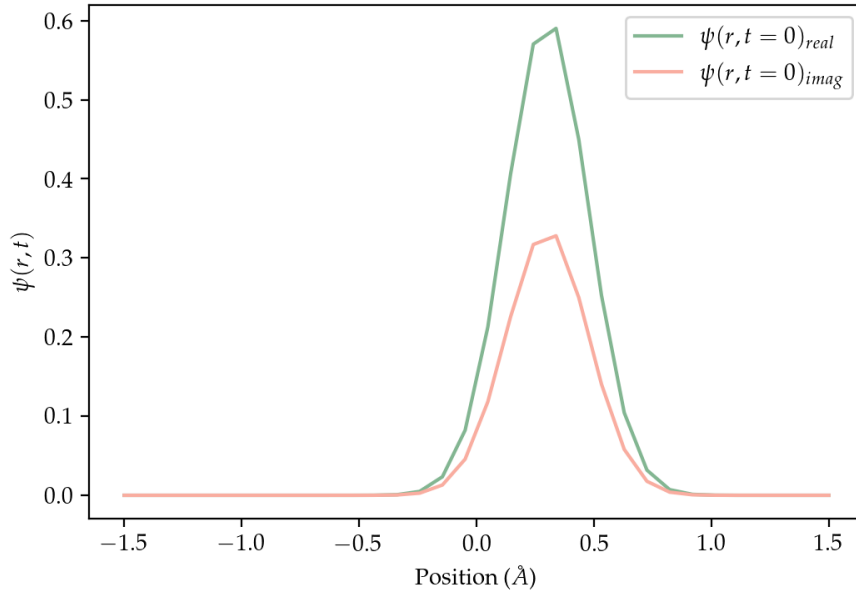


Figura 6.5: Parte real y compleja de un paquete de onda inicial  $\psi(r, t = 0)$ .

El modelo de potencial  $V(r, t)$  utilizado fue el mismo que se usó en la sección § 3.3.1. Para cada trayectoria se eligieron valores de parámetros para el modelo de potencial aleatorios entre rangos especificados en la ???. En la Figura 6.6 se muestra un ejemplo de un potencial generado.

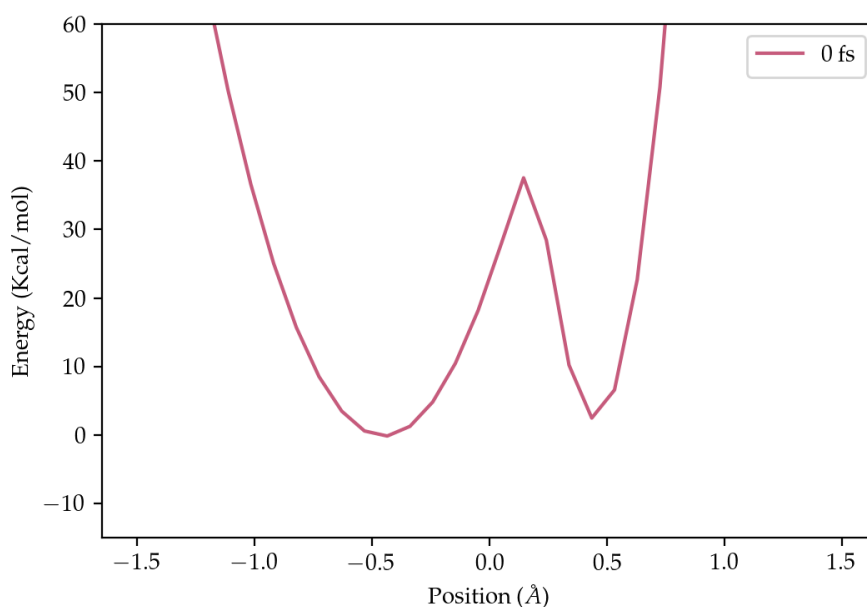


Figura 6.6: Potencial inicial al tiempo  $t = 0$

En el repositorio [Trayectorias](#) se encuentran los datos generados. Cada carpeta contiene los directorios y archivos siguientes:


#### Trayectorias

```

└─ data0
    └─ Potential
        ├── 0-Potential.npy
        ├── ⋮
        └── 200-Potential.npy
    └─ Wavepacket
        ├── 0-Wavepacket.npy
        ├── ⋮
        └── 200-Wavepacket.npy
  
```



En donde `data0` corresponde a la trayectoria 0. El archivo *o-Potential.npy* es un arreglo de  $N = 32$  entradas que corresponde al valor del potencial al tiempo  $t = 0fs$ , el archivo *o-Wavepacket.npy* es un arreglo de  $N = 32$  entradas complejas que corresponden al paquete de onda al tiempo  $t = 0fs$ ; así respectivamente para cada tiempo hasta  $t = 200fs$ . Además de los datos, se encuentra un archivo de texto: *ValuesPotential.txt*, en donde se registran los valores de los parámetros para generar el potencial, es decir, los valores exactos elegidos de la ??.

El código para generar las trayectorias se encuentra en el repositorio:  **Generación de datos**, que contiene dos clases: *Potential\_System*, que contiene las ecuaciones del modelo del potencial y la clase *ProtonTransfer*, con la que se generan las trayectorias. A continuación se muestra un ejemplo para generar una trayectoria, con la información que muestra el diagrama de la **Figura 6.4**:

```
trayectoria0 = ProtonTransfer(n=32, a=-1.5, b=1.5, time=True,
    var_random=True, save_dir='data0')
trayectoria0.vector_potential(t=200, step=1)
trayectoria0.evolution_wp(t=200, step=1, gaussiana=True)
```

En donde se utilizan las siguientes variables:

- `n`: Número de puntos en el grid
- `a`: Punto inicial del grid [ $\text{\AA}$ ]
- `b`: Punto final del grid [ $\text{\AA}$ ]
- `time`: True o False. Determina si se utiliza un potencial dependiente del tiempo: True, o independiente del tiempo: False
- `var_random`: True o False. True inicia las variables de manera aleatoria para el potencial del sistema. False solicita al usuario cada variable. ??
- `save_dir`: Nombre del directorio donde se guardarán los datos del potencial y la evolución de onda.
- `t`: Tiempo total de la trayectoria [ $fs$ ]
- `step`:  $\Delta t$  para la evolución temporal de la onda [ $fs$ ]
- `gaussiana`: True o False. True genera una onda inicial Gaussiana (**Ecuación 6.10**), False solicita  $k$  para generar la onda inicial como una suma de eigenfunciones (**Ecuación 3.25**)

#### 6.4.2 Procesamiento de Datos: Visualización y Forma

Una vez generadas las trayectorias, para entrenar el modelo de red

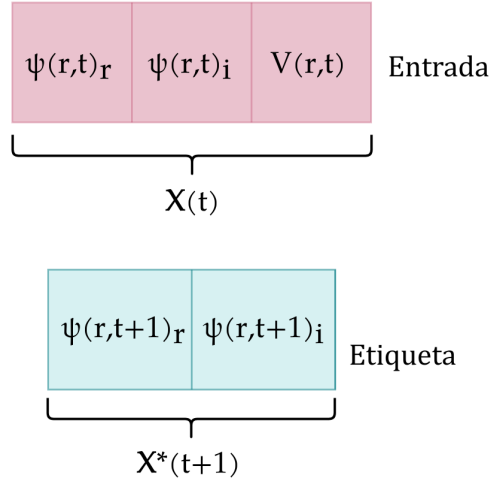


Figura 6.7: Entrada  $X$  y etiqueta  $y$  a un tiempo  $t$ .

#### 6.4.3 Modelo LSTM

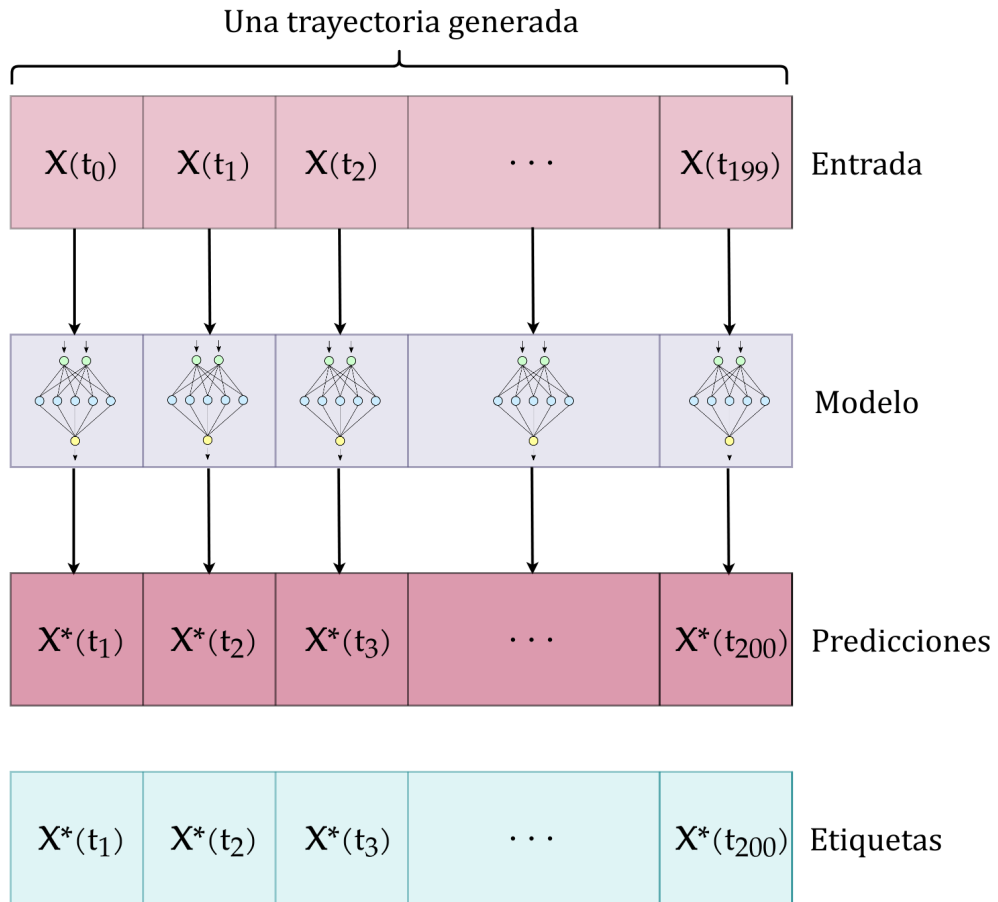
#### 6.4.4 Función de Precisión

$$S = \langle \psi_{pred} | \psi_{real} \rangle = |S| \exp\{i\theta\} \quad (6.11)$$

#### 6.4.5 Resultados: Precisión y Predicciones

#### 6.4.6 Análisis de Resultados

#### 6.4.7 Conclusión

Figura 6.8: Vector  $X$  a un tiempo  $t$ .

## Parte V

### APÉNDICE

# A



## APÉNDICE

### A.1 VARIABLES DE PARÁMETROS PARA EL POTENCIAL

VARIABLE	VALOR
$V$	0.015936 [ $a.u$ ]
$\omega_1$	0.00811569 [ $a.u$ ]
$\omega_2$	0.00978836 [ $a.u$ ]
$\lambda$	0.00915455 [ $a.u$ ]
$X_{eq}$	0.00628338 [ $a.u$ ]
$\omega_x$	0.000357994 [ $Jiffy^{-1}$ ]
$\theta_X$	0.106646 [ $rad$ ]
$R_{eq}$	1.39712 [ $a_0$ ]
$R_0$	0.42091 [ $a_0$ ]
$\omega_R$	0.00135403 [ $au$ ]
$\theta_R$	4.52653 [ $rad$ ]
$m$	1836 [ $m_e$ ]

Tabla A.1: Valores de parámetros del potencial utilizados para generar la gráfica de la [Figura 3.2](#)

### A.2 UNIDADES ATÓMICAS

VARIABLE	VALOR
$V$	$0.015936 [a.u.]$
$\omega_1$	$0.006834 - 0.018224 [a.u.]$
$\omega_2$	$0.006834 - 0.018224 [a.u.]$
$\lambda$	$0 - 0.015936 [a.u.]$
$X_{eq}$	$-0.015936 - 0.015936 [a.u.]$
$\omega_x$	$0.000357994 [Jiffy^{-1}]$
$\theta_X$	$0 - 2\pi [rad]$
$R_{eq}$	$0.377945 - 1.889725 [a_0]$
$R_0$	$0 - 2R_{eq} [a_0]$
$\omega_R$	$0.0004556 - 0.0013668 [a.u.]$
$\theta_R$	$0 - 2\pi [rad]$
$m$	$1836 [m_e]$

Tabla A.2: Rango de valores de parámetros del potencial utilizados para generar trayectorias.

UNIDADES ATÓMICAS			VALOR SI	NOMBRE (SÍMBOLO)
Masa $m_e$			9.10939(-31)	Masa del electrón
Carga $e$			1.602188(-19)	Carga del electrón
Momento angular $\hbar$			1.05457(-34)	Constante de Planck/ $2\pi$
Energía ( $m_e e^4 / \hbar^2$ )			4.35975(-18)	Hartree ( $H$ )
Longitud ( $\hbar^2 / m_e e^2$ )			5.29177(-11)	Radio de Bohr ( $a_0$ )
Tiempo ( $\hbar^3 / m_e e^4$ )			2.41888(-17)	Jiffy
Momento	Dipolar	Eléctrico	8.47836(-30)	2.541765 Debye ( $D$ ) units
( $\hbar^2 / m_e e$ )				
Momento	Dipolar	Magnético	9.27402(-24)	Magneton de Bohr ( $\mu_B$ )
( $e\hbar / 2m_e$ )				

Tabla A.3: Conversión de Unidades Atómicas a Unidades SI.

Unidad	$a.u.$	$kcal/mol$	$eV$	$cm^{-1}$	$Hz$
$a.u.$	1	6.27510(2)	2.72114(1)	2.19475(5)	6.57968(15)
$kcal/mol$	1.59360(-3)	1	4.33641(-2)	3.49755(2)	1.04854(13)
$eV$	3.67493(-2)	2.30605(1)	1	8.06554(3)	2.41799(14)
$cm^{-1}$	4.55634(-6)	2.85914(-3)	1.23984(-4)	1	2.99792(10)
$Hz$	1.51983(-16)	9.5371(-14)	4.13567(-15)	3.33564(-11)	1

Tabla A.4: Conversión de Energía (diversas unidades).



## BIBLIOGRAFÍA

- [1] François Chollet. *Deep Learning with Python*. Manning, **november** 2017. ISBN: 9781617294433.
- [2] Daniel T. Colbert **and** William H. Miller. ?A novel discrete variable representation for quantum mechanical reactive scattering via the  $S$ -matrix Kohn method? **in**: *The Journal of Chemical Physics* 96.3 (**february** 1992), **pages** 1982–1991. DOI: [10.1063/1.462100](https://doi.org/10.1063/1.462100). URL: <https://doi.org/10.1063/1.462100>.
- [3] James T. Hynes Daniel Borgis. ?Dynamical theory of proton tunneling transfer rates in solution: general formulation? **in**: *Chemical Physics* 170 (1993), **pages** 315–346. DOI: [10.1016/0301-0104\(93\)85117-Q](https://doi.org/10.1016/0301-0104(93)85117-Q). URL: [https://doi.org/10.1016/0301-0104\(93\)85117-Q](https://doi.org/10.1016/0301-0104(93)85117-Q).
- [4] Richard P. Feynman. *The Feynman Lectures On Physics Volume III: Quantum Mechanics* 3. 140522731 **edition**. **volume** III. The Feynman Lectures On Physics 03. Addison-Wesley Pub, 1977. ISBN: 0201020106.
- [5] Cutter Mary Ann Gardell. *The brain: Understanding neurobiology through the study of addiction*. Center for Curriculum Development, 2010.
- [6] I. Goodfellow, Y. Bengio **and** A. Courville. *Deep Learning*. Adaptive computation and machine learning. MIT Press, 2016. ISBN: 9780262035613. URL: <https://books.google.co.in/books?id=Np9SDQAAQBAJ>.
- [7] David Gottlieb **and** Steven A. Orszag. *Numerical Analysis of Spectral Methods*. Society for Industrial **and** Applied Mathematics, 1977. DOI: [10.1137/1.9781611970425](https://epubs.siam.org/doi/pdf/10.9781611970425). eprint: <https://epubs.siam.org/doi/pdf/10.1137/1.9781611970425>. URL: <https://epubs.siam.org/doi/abs/10.1137/1.9781611970425>.
- [8] Andrej Karpathy, Justin Johnson **and** Li Fei-Fei. *Visualizing and Understanding Recurrent Networks*. 2015. DOI: [10.48550/ARXIV.1506.02078](https://arxiv.org/abs/1506.02078). URL: <https://arxiv.org/abs/1506.02078>.



- [9] Joshua P. Layfield **and** Sharon Hammes-Schiffer. ?Hydrogen Tunneling in Enzymes and Biomimetic Models? **in:** *Chemical Reviews* 114.7 (2014). PMID: 24359189, **pages** 3466–3494. DOI: [10.1021/cr400400p](https://doi.org/10.1021/cr400400p). eprint: <https://doi.org/10.1021/cr400400p>. URL: <https://doi.org/10.1021/cr400400p>.
- [10] Kevin P. Murphy. *Machine learning : a probabilistic perspective*. Cambridge, Mass. [u.a.]: MIT Press, 2013. ISBN: 9780262018029 0262018020.
- [11] Michael A. Nielsen. *Neural Networks and Deep Learning*. misc. 2018. URL: <http://neuralnetworksanddeeplearning.com/>.
- [12] Haşim Sak, Andrew Senior **and** Françoise Beaufays. *Long Short-Term Memory Based Recurrent Neural Network Architectures for Large Vocabulary Speech Recognition*. 2014. DOI: [10.48550/ARXIV.1402.1128](https://arxiv.org/abs/1402.1128). URL: <https://arxiv.org/abs/1402.1128>.
- [13] J. J. Sakurai. *Modern quantum mechanics*. Rev. ed. Addison-Wesley, 1994. ISBN: 9780201539295.
- [14] Maxim Secor, Alexander V. Soudackov **and** Sharon Hammes-Schiffer. ?Artificial Neural Networks as Propagators in Quantum Dynamics? **in:** *The Journal of Physical Chemistry Letters* 12.43 (2021). PMID: 34704767, **pages** 10654–10662. DOI: [10.1021/acs.jpcllett.1c03117](https://doi.org/10.1021/acs.jpcllett.1c03117). eprint: <https://doi.org/10.1021/acs.jpcllett.1c03117>. URL: <https://doi.org/10.1021/acs.jpcllett.1c03117>.
- [15] David J. Tannor. *Introduction to Quantum Mechanics: A Time-Dependent Perspective*. University Science Books, 2006. ISBN: 1891389238.
- [16] Ivan Vasilev, Daniel Slater, Gianmario Spacagna, Peter Roelants **and** Valentino Zocca. *Python deep learning: Exploring deep learning techniques and neural network architectures with pytorch, Keras, and tensorflow*. Packt Publishing, 2019.