UNIVERSITY NAME

DOCTORAL THESIS

Thesis Title

Author:
John SMITH

Supervisor:
Dr. James SMITH

A thesis submitted in fulfillment of the requirements
for the degree of Doctor of Philosophy

in the

Research Group Name
Department or School Name

22 de junio de 2021

«*Thanks to my solid academic training, today I can write hundreds of words on virtually any topic without possessing a shred of information, which is how I got a good job in journalism.*»

Dave Barry

UNIVERSITY NAME

Resumen

Faculty Name
Department or School Name

Doctor of Philosophy

Thesis Title

by John SMITH

The Thesis Abstract is written here (and usually kept to just this page). The page is kept centered vertically so can expand into the blank space above the title too…

Agradecimientos

The acknowledgments and the people to thank go here, don't forget to include your project advisor...

Índice general

Índice de figuras

Índice de cuadros

For/Dedicated to/To my…

Capítulo 1

Solución a la ecuación de Ornstein-Zernike usando redes neuronales

Las *redes neuronales* fungen como aproximadores universales [HSW89; Hor91; Cyb89], y como tal pueden ser utilizadas para aproximar cualquier función continua para un determinado tipo de arquitectura. En particular, se espera que una red neuronal pueda servir como paramatrización de la función puente en la condición de cerradura de la ecuación de OZ, y así evitar escoger una aproximación en especial, y dejar que la red neuronal tome la forma necesaria para resolver la ecuación integral.

En este capítulo se detalla la metodología creada y los fundamentos matemáticos bajo los cuales se puede hacer uso de redes neuronales para resolver la ecuación de OZ. Se comparan estos resultados con los obtenidos con simulación por computadora. En el apéndice se describe la solución general a la ecuación de OZ, mientras que en este capítulo se detalla su modificación para emplear redes neuronales.

1.1. Parametrización de la función puente

La ecuación de Ornstein-Zernike está dada por

$$c(\mathbf{r}) = h(\mathbf{r}) + n \int_V c(\mathbf{r}')h(|\mathbf{r} - \mathbf{r}'|)d\mathbf{r}'$$
$$c(\mathbf{r}) = \exp\left[-\beta u(\mathbf{r}) + \gamma(\mathbf{r}) + B(\mathbf{r})\right] - \gamma(\mathbf{r}) - 1$$

con la notación ya conocida para esta ecuación.

Sea $N_\theta(\mathbf{r})$ una red neuronal con pesos θ. Se propone que $N_\theta(\mathbf{r})$ reemplace a la función puente $B(\mathbf{r})$ en la ecuación anterior tal que ahora se tiene la siguiente expresión para la cerradura

$$c(\mathbf{r}) = \exp\left[-\beta u(\mathbf{r}) + \gamma(\mathbf{r}) + N_\theta(\mathbf{r})\right] - \gamma(\mathbf{r}) - 1, \tag{1.2}$$

y la ecuación de OZ queda de la misma forma. Se pretende trabajar a partir de esta hipótesis donde la función puente pueda tomar cualquier valor en particular que permita la solución de la ecuación de OZ.

1.2. Esquema de entrenamiento

Ahora que se tiene la parametrización, se debe desarrollar una forma para ajustar los pesos de la red neuronal $N_\theta(\mathbf{r})$, y que al mismo tiempo permita la solución de la ecuación de OZ.

1.2.1. Función de costo

Para crear un esquema de entrenamiento se requiere primero de una **función de costo** que sea utilizada para generar información del ajuste necesario de los pesos θ. Tomando en cuenta que el esquema iterativo de Piccard crea una sucesión de funciones estimadas $\{\gamma_1(\mathbf{r}), \gamma_2(\mathbf{r}), \ldots, \gamma_n(\mathbf{r})\}$, se propone que la función de costo sea

$$J(\theta) = [\gamma_n(\mathbf{r};\theta) - \gamma_{n-1}(\mathbf{r};\theta)]^2 \tag{1.3}$$

donde $\gamma_n(\mathbf{r};\theta)$ es la enésima función estimada de $\gamma(\mathbf{r})$ mediante el método iterativo. La notación $\gamma(\mathbf{r};\theta)$ indica que depende implícitamente de los pesos de la red neuronal, como se puede ver en el ecuación (1.2). Esto significa que si los pesos de $N_\theta(\mathbf{r})$ cambian, entonces la función γ debe cambiar. Sin embargo, esto no significa que γ depende *directamente* de los pesos.

1.2.2. Problema de optimización

Usando la función de costo (1.3) ahora se puede plantear un problema de optimización que permita el ajuste de los pesos de $N_\theta(\mathbf{r})$.

El problema de optimización en este caso es *sin restricciones* y está definido de la siguiente forma

$$\min_\theta \quad J(\theta) \tag{1.4}$$

Esto significa que buscamos los parámetros óptimos θ^* que minimicen la diferencia entre las iteraciones calculadas de las funciones $\gamma(\mathbf{r};\theta)$. Este problema de optimización se puede resolver de forma iterativa, al mismo tiempo que se resuelve la ecuación de OZ.

1.2.3. Actualización de pesos

El método iterativo empleado está basado en la técnica de *descenso de gradiente* para actualizar los pesos de la red neuronal. Esto significa que en general, una regla de actualización está dada por una expresión semejante a la siguiente

$$\theta_{n+1} = \theta_n - \eta \nabla_\theta J(\theta). \tag{1.5}$$

donde η se conoce como la *razón de aprendizaje*, y es un parámetro que controla el tamaño de paso del gradiente hacia la dirección de descenso.

Independientemente de la expresión de la actualización de los pesos, este cálculo requiere de la información del gradiente de la función de costo respecto a los pesos, $\nabla_\theta J(\theta)$. Para este caso en particular, el cálculo detallado del gradiente se describe en el apéndice. Una vez que se tiene esta información, lo que resta es construir un algoritmo que use este esquema de entrenamiento y que resuelva la ecuación de OZ.

1.2.4. Solución a la ecuación de Ornstein-Zernike con redes neuronales

Con todos los elementos necesarios, se puede entonces definir el esquema general para la solución a la ecuación de Ornstein-Zernike usando redes neuronales.

Para resolver la ecuación de OZ con la parametrización (1.2) se tienen lo siguientes pasos:

1. Dado el potencial de interación $u(\mathbf{r})$, se emplea la ecuación (1.2) para obtener el valor de $c(\mathbf{r};\theta)$, la cual ahora depende implícitamente de los pesos de $N_\theta(\mathbf{r})$. En este paso también se requiere de un valor inicial para $\gamma_n(\mathbf{r})$, el cual se inicializa de acuerdo a la metodología de Ng de cinco puntos, descrita en el apéndice.

2. La función $c(\mathbf{r};\theta)$ se transforma mediante la transformada de Fourier, para obtener $\hat{c}(\mathbf{k};\theta)$.

3. Se transforma mediante la transformada de Fourier toda la ecuación de OZ, y se deja en términos de la función $\hat{\gamma}(\mathbf{k})$. Usando la información encontrada sobre la función $\hat{c}(\mathbf{k};\theta)$, se encuentra una nueva función $\hat{\gamma}_{n+1}(\mathbf{k};\theta)$.

4. Luego, se antitransforma la función $\hat{\gamma}_{n+1}(\mathbf{k};\theta)$ para obtener la nueva estimación, $\gamma_{n+1}(\mathbf{r};\theta)$.

5. Usando ambas estimaciones γ_n y γ_{n+1}, se evalúa la función de costo (1.3) y se realiza el cálculo del gradiente, $\nabla_\theta J(\theta)$.

6. Se actualizan los pesos usando la expresión (1.5), y se reinicia el proceso. En la siguiente iteración, la función inicial será ahora γ_{n+1}, para entonces obtener una nueva estimación γ_{n+2}, y así sucesivamente.

1.2.5. Convergencia del entrenamiento

El procedimiento descrito anteriormente se repite de forma indefinida hasta lograr la convergencia, dada por la siguiente expresión

$$|\gamma_{n+1} - \gamma_n|^2 \leq \epsilon \tag{1.6}$$

donde $|\cdot|$ indica el valor absoluto, y $\epsilon \in [0, 1]$. En particular, la tolerancia en todos los experimentos realizados fue de $\epsilon = 1 \times 10^{-5}$. En otras palabras, se exige a la red neuronal ajustar sus pesos hasta que las estimaciones sucesivas de las funciones γ sean muy semejantes entre sí.

1.3. Implementación

En esta sección se detallan los aspectos importantes de la implementación del método descrito en la sección anterior, como lo son la arquitectura de la red neuronal, el método de optimización, y la elección de las funciones de activación.

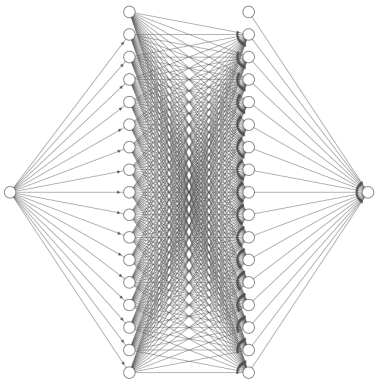

FIGURA 1.1: Esquema genérico de una red neuronal multicapa completamente conectada. En especial, es importante notar la forma de la red, donde se tienen dos capas escondidas. Cada una de estas *unidades* o *nodos* está siendo evaluada por una función de activación. La arquitectura usada en los resultados de este capítulo es más grande, pero tiene la misma forma.

1.3.1. Elección de optimizador

La regla general de actualización de pesos (1.5) fue implementada para resolver el problema de optimización, sin embargo, problemas numéricos impedían la convergencia del entrenamiento.

Para resolver este problema se optó por utilizar el método *Adam* [KB17], el cual es un método de optimización adecuado para entrenamiento de redes neuronales, sobre todo cuando el gradiente puede ser *disperso*, i.e. que muchas de sus entradas sean cero. El método *Adam* emplea diversas reglas para ajustar la dirección de descenso del gradiente, así como los hiperparámetros asociados al método. En particular, este método utiliza dos hiperparámetros, β_1, que controla el promedio móvil del gradiente calculado; y β_2, que controla el cuadrado del gradiente. Ambas cantidades son necesarias para la convergencia óptima del algoritmo.

En los resultados presentados en esta sección, se emplearon los hiperparámetros con sus valores estándar, $\beta_1 = 0{,}9$ y $\beta_2 = 0{,}999$. Es importante notar que este método de optimización utiliza sus propios mecanismos para controlar los gradiente, así como estos hiperparámetros. El valor de la *razón de aprendizaje*, η en la ecuación (1.5), utilizado en todos los experimentos fue de $\eta = 1 \times 10^{-4}$.

1.3.2. Arquitectura de la red neuronal

La arquitectura de la red neuronal empleada en todos los experimentos es aquella que se muestra en el esquema de la figura 1.1, excepto por el número de unidades en las capas ocultas. Es decir, la arquitectura es de *cuatro capas*, una capa de *entrada* con una unidad, dos capas ocultas con 5000 unidades cada una, y por último una capa de *salida* con una sola unidad.

La función de activación que se empleó fue la función *ReLU* [GBB11], la cual está definida con la siguiente expresión

$$\mathrm{ReLU}(x) = \max(0, x).$$

Esta función de activación se aplica a todas las capas a excepción de la capa de entrada. Se escogió esta función de activación debido a que todas las más comunes (tanh, softmax, entre otras), provocaban inconsistencias numéricas.

1.4. Resultados

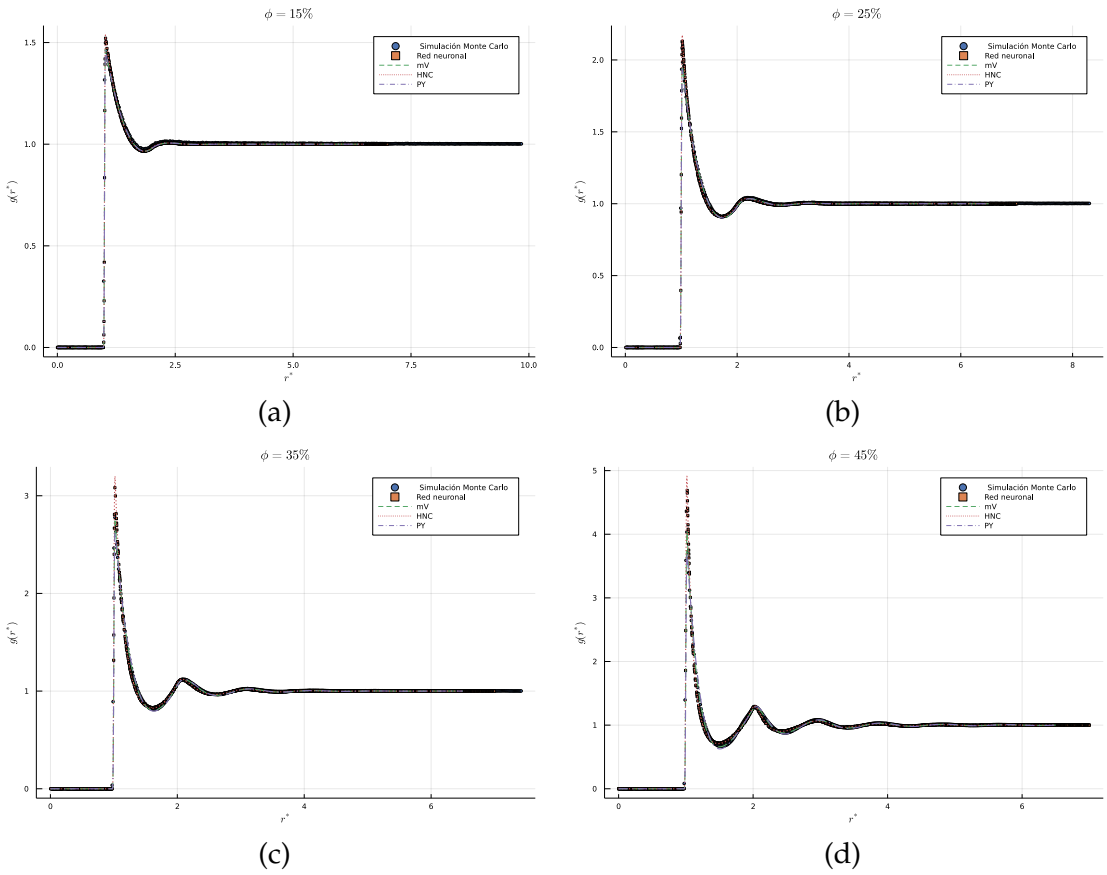

FIGURA 1.2: Funciones de distribución radial obtenidas con simulación por computadora y mediante el uso de la metodología descrita en este capítulo, usando redes neuronales. Se presentan cuatro densidades: (a) $\phi = 15\,\%$, (b) $\phi = 25\,\%$, (c) $\phi = 35\,\%$, (d) $\phi = 45\,\%$. En cada una, se muestran las comparaciones con tres diferentes aproximaciones para la función puente: *mV*, Verlet modificada; *PY*, Percus-Yevick; y *HNC*, Hypernetted Chain.

Bibliografía

[Cyb89] G. Cybenko. «Approximation by Superpositions of a Sigmoidal Function». en. En: *Mathematics of Control, Signals and Systems* 2.4 (dic. de 1989), págs. 303-314. ISSN: 1435-568X. DOI: 10.1007/BF02551274.

[GBB11] Xavier Glorot, Antoine Bordes y Yoshua Bengio. «Deep Sparse Rectifier Neural Networks». en. En: *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*. JMLR Workshop and Conference Proceedings, jun. de 2011, págs. 315-323.

[Hor91] Kurt Hornik. «Approximation Capabilities of Multilayer Feedforward Networks». en. En: *Neural Networks* 4.2 (ene. de 1991), págs. 251-257. ISSN: 0893-6080. DOI: 10.1016/0893-6080(91)90009-T.

[HSW89] Kurt Hornik, Maxwell Stinchcombe y Halbert White. «Multilayer Feedforward Networks Are Universal Approximators». en. En: *Neural Networks* 2.5 (ene. de 1989), págs. 359-366. ISSN: 0893-6080. DOI: 10.1016/0893-6080(89)90020-8.

[KB17] Diederik P. Kingma y Jimmy Ba. «Adam: A Method for Stochastic Optimization». En: *arXiv:1412.6980 [cs]* (ene. de 2017). arXiv: 1412.6980 [cs].