TRABAJO FIN DE GRADO

Doble Grado en Ingeniería Informática y Matemáticas

Título en castellano.

Autor
Daniel Jiménez López

Directores
Antonio Miguel Lallena Rojo
Juan Antonio López Villanueva

Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación

—

Granada, 19 de febrero de 2019

Título en castellano.

Daniel Jiménez López

Palabras clave: Teorema No Free Lunch, Deep Learning, Backpropagation, Gradiente Descendente, redes neuronales, redes neuronales convolucionales, algoritmos evolutivos, Evolución Diferencial

Resumen

En los últimos seis años, las redes neuronales profundas, comúnmente llamadas **Deep Learning**, han demostrado muy buenos resultados en diversos problemas de clasificación de imágenes, en algunos casos superando a personas expertas en sus respectivos campos. Una de las claves del éxito de estos modelos es el uso del **Gradiente Descendente Estocástico** o una de sus variantes, como es **RMSProp**, para optimizar el aprendizaje de estas redes.

Por otra parte, el conjunto de algoritmos evolutivos basados en **Evolución Diferencial**, han demostrado ser muy útiles para problemas de optimización continua. Sin embargo, todavía no está demostrado su potencial para el aprendizaje en redes neuronales. En este proyecto se analiza la capacidad de los algoritmos de **Evolución Diferencial auto-adaptativa (SaDE)**, así como sus variantes **L-SHADE** y **SHADE-ILS**, para el entrenamiento de las redes neuronales. Como caso de estudio se considera el problema popular de reconocimiento de dígitos manuscritos en imágenes (MNIST). Nuestros resultados muestran que la **Evolución Diferencial** puede ser muy competitva con el **Gradiente Descendente**, a pesar de los muy buenos resultados obtenidos por el mismo, como optimizador de redes neuronales.

Título en inglés

Daniel Jiménez López

Keywords: No Free Lunch Theorem, Stochastic Gradient Descent, Backpropagation , Deep Learning, evolutionary algorithms, Differential Evolution, Neural Networks, Convolutional Neural Networks

Abstract

Deep Neural Networks, also known as **Deep Learning** constitute a subset of algorithms in the field of **Machine Learning**. Their functioning is inspired from the way human brain works. In the last six years, these networks have shown great results in computer vision task such as image classification. A lot of people have studied the classification problem known as **handwritten digits**, which uses a database called **'MNIST'**. Up to date, the most used algorithm to train Neural Networks is **RMSProp**, a type of **Stochastic Gradient Descent**.

In this project we are going to deepen the study of the optimization of Neural Networks. The optimizer will be changed and it will be shown how results become competitive and moreover, if we fit correctly all the parameters and adjust the new optimizers to our problem, results will be close to results using RMSProp. Algorithms chosen as optimizers in our project are the **evolutionary algorithms**, based on the natural evolution and the power of natural selection between generations in populations. So, we create individuals who represent solutions to the handwritten problem. We represent them as a set of weights which we can reshape and assign to the different connections between layers, obtaining, after evaluation, a set of labels which should be the most similar possible to labels of the set we are training. Therefore, the idea is to create a population of individuals and use the correct operators of crossover, mutation and selection to obtain, after some generations, a population with quality individuals. These algorithms are atractive because of different characteristics: simplicity, ease to combine with other search algorithms (like Local Search), ease to parallelize, applicability over non-differenciable problems, etc. These characteristics, are a motivation to use this algorithms as the new optimizers in Neural Networks.

Within evolutionary algorithms, we will use **Differential Evolution**. These algorithms works well in real-valued problems and for continuous fitness functions. They do not need the fitness function to be differentiable. They

have the same foundations that all evolutionary algorithms: creating a population of candidate solutions we try to evolve them and obtein a final population with quality solutions. Then, we obtain the best in this population and we consider it as the solution of our problem. They use operators such as all evolutionary algorithms: crossover, selection and mutation. They need to control three very important parameters: *number of individuals in population*, F, which control the width of change in new individuals, and CR which is a cross rate. It is important to choose correctly this parameters; if we try to do it by hand, it could cost a lot of time and a lot of computing. In this context, **Self-adaptative Differential Evolution (SaDE)** appears. They adapt themselves throughout the execution of the algorithm. In this project we will use two algorithms with great results in some competitions. They are **SHADE** and **L-SHADE**.

The final objective of this draft is show how, adapting well different SaDE algorithms to our problem and to their use in Neural Networks, results are competitive. So it involves that other type of algorithms, such evolutionary algorithms could be employed as optimizers in Neural Networks and also, it implies to leave the idea of **Backpropagation**. Trying to reach this objective, is important to learn the mathematical foundations within optimization of Gradient Descent and how Backpropagation propagate the error. Therefore, we will explain this mathematical aspects joined to an important theorem which it is useful as motivation for trying this change of algorithms used as optimizers: **No Free Lunch Theorem**, which says no-one algorithm is the best comparing it with the rest of algorithms applied on the total set of all the existing problems. After that, we deepen in computing aspects linked with Neural Networks. Specially, Convolutional Neural Networks, which results are great in images recognition.Also, we will study evolutionary algorithms, ending the theorical showing some works where Neural Networks are mixed with evolutionary algorithms. finisthing therical aspects, algorithms using in experimentation will be explained with more details and we will show results of applying them. We will extract some conclusion of this results where the most important is that, indeed, SaDE (especially combined with Local Search), is competitive used as optimizer of Neural Networks. There is a hopeful future trying to improve this algorithms for using them as optimizers, that perhaps it would allow to obtain results equal or better than those obtained with RMSProp.

Índice general

1 Introducción

El juego de vida es un tipo de autómata celular propuesto por Conway en 1970 y popularizado por Martin Gardner en el mismo año [6]. Éste consiste en la evolución de una configuración inicial de células con dos estados mutuamente excluyentes, vida (1) o muerte (0), en una malla rectangular infinita dos dimensional. Dicha evolución viene dada por un conjunto de reglas que se aplican simultáneamente a todas las células y el vecindario de éstas, identificado por las 8 células adyacentes que la rodean horizontal, vertical y diagonalmente (vecindario tipo Moore). El conjunto de reglas de evolución se identifica por B3/S23, donde las cifras que siguen a la letra B (Born) indican el número de vecinos necesario para que se dé un nacimiento y las cifras que siguen a la letra S expresan el número de vecinos necesarios para la supervivencia de una célula , en otro caso la célula muere. Así pues, en el juego de vida, dada una célula viva, ésta continua viviendo si en su vecindario hay 2 o 3 células, en otro caso muere y dada una célula muerta, nace si tiene 3 células en su vecindario.

La elección de las reglas de evolución parecería a priori aleatoria, sin embargo, Conway perseguía con ellas obtener el siguiente comportamiento [8]:

- No debe de haber una configuración inicial de reglas para las cuales haya una prueba simple de que la población pueda crecer sin límite.
- Debe de haber configuraciones iniciales simples que crezcan y cambien durante un periodo considerable, llegando a tres posibles finales: desaparecer complemante ya sea debido a sobrepoblación o dispersión, estabilizarse en una configuración que se mantenga constante o entrar en un ciclo sin fin de oscilación de periodo igual o mayor a dos.
- Uno de los motivos por los que atrajo la atención de científicos de diferentes campos es la capacidad de observar como patrones complejos surgen de la aplicación de un conjunto muy simple y reducido de reglas.

Uno de los motivos por los que atrajo la atención de científicos de diferen-

tes campos es la capacidad de observar como patrones complejos surgen de la aplicación de un conjunto muy simple y reducido de reglas. De esta manera comenzaron a observarse configuraciones iniciales que daban lugar a comportamientos interesantes. Tales como las de 'naves espaciales' que se desplazan sobre la malla rectángular, los 'osciladores' que retornan a su configuración inicial después de un número finito de generaciones o las 'vidas inmóviles', osciladores de periodo la unidad.

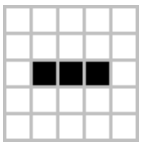

Figura 1.1: Oscilador de periodo dos nombrado 'Blinker'

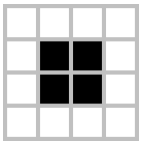

Figura 1.2: Vida inmóvil nombrada 'Block'

Debido al creciente interés, Conway propuso la búsqueda de una configuración inicial que podría crecer sin límite, la cual William Gosper encontró con la construcción de una configuración inicial de células que genera infinitos deslizadores durante su evolución.

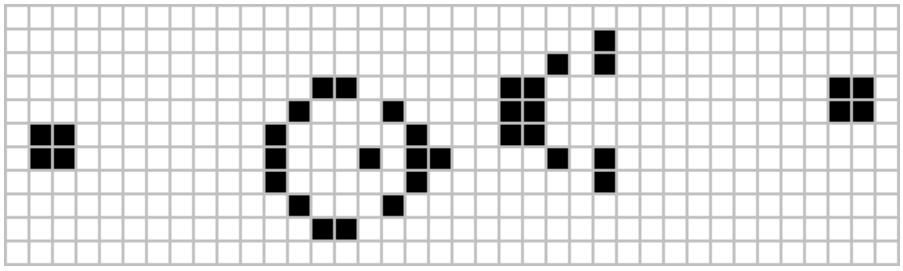

Figura 1.3: 'Pistola de desplizadores' o 'Gosper glider gun' propuesta por William Gosper.

Dada la popularidad del juego de vida, surge la necesidad de realizar simulaciones en los incipientes ordenadores y se enfrentan al problema de representar una malla infinita en un ordenador con memoria finita. Para ello

se propone alterar las características topológicas de la malla, imitando las de una botella de Klein, una esfera, o un toro. En particular, esta última resulta atraer gran interés, pues se obtiene evidencia de que reduce los efectos asociados a la finitud de la malla [5, 7]. Cabe descatar el estudio de la alteración de las cualidades geométricas de la malla, tales como el uso de figuras geométricas regulares diferentes al cuadrado (triángulo y hexágono) [4], teselaciones de Penrose [10] o el empleo del espacio geométrico hiperbólico [11]. Finalmente, cabe notar que existen implementaciones en las cuales no se aplican condiciones sobre los bordes del dominio [2].

El juego de vida también muestra interesantes características en el campo de teoría de la computación. Pertenece a la clase IV de Wolfram [9, 14] y se ha demostrado que es una máquina de Turing universal [12]. Por tanto existe una configuración inicial que simula una máquina de Turing, la cual fue extendida a una máquina universal de Turing [3]. Más recientemente en un esfuerzo colectivo, se ha implementado un ordenador con su propio lenguaje ensamblado, compilador a lenguaje de alto nivel, y sobre este último se ha implementado el conocido juego Tetris [1, 13].

En el juego de vida las células se actualizan simultáneamente en unidades de tiempo discretas, lo que implica que es un autómata celular síncrono. Si relajamos la condición de sincronicidad, obtenemos

En este trabajo emplearemos el método de Monte Carlo a configuraciones en situación de α-asincronicidad caracterizando su comportamiento a través de diferentes variables; el número de clusteres de células diferentes por unidad discreta de tiempo, su velocidad y densidad, entre otras, las cuales hasta donde podemos saber no han sido de interés en anteriores publicaciones. De esta manera podremos cuantificar el efecto de la α-asincronicidad en configuraciones iniciales bien conocidas, tales como las comentadas anteriormente.

Bibliografía

[1] Build a working game of Tetris in Conway's Game of Life - StackExchange. https://web.archive.org/web/20190120082452/https://codegolf.stackexchange.com/questions/11880/build-a-working-game-of-tetris-in-conways-game-of-life. [Online; accessed 14-February-2019].

[2] Conway's Game of Life - Boardless approach. https://web.archive.org/web/20190212184430/https://rosettacode.org/wiki/Conway's_Game_of_Life#Boardless_approach. [Online; accessed 14-February-2019].

[3] A Turing Machine in Conway's Game of Life, extendable to a Universal Turing Machine. https://web.archive.org/web/20190118023640/http://rendell-attic.org/gol/tm.htm. [Online; accessed 14-February-2019].

[4] Bays, C. Cellular Automata in the Triangular Tessellation. *Complex Systems 8* (1994), 127–150.

[5] Block, H. J., and Bergersen, B. Effect of boundary conditions on scaling in the 'Game of Life'. *Physical Review E 55*, 5 (1997), 6249–6252.

[6] Gardner, M. Mathematical Games - The fantastic combinations of John Conway's new solitaire game "life". *Scientific American 223*, 4 (1970), 120–123.

[7] Hemmingsson, J. Consistent results on 'Life'. *Physica D 80*, 1-2 (1995), 151–153.

[8] Klarner, D. Wheels, Life and Other Mathematical Amusements, by Martin Gardner. *The American Mathematical Monthly 93*, 4 (1986), 321–323.

[9] McIntosh, H. V. Wolfram's class IV automata and a good life. *Physica D 45* (1990), 105–121.

[10] Owens, N., and Stepney, S. Investigations of the Game of Life cellular automata rules on Penrose tilings: lifetime, ash and oscillator statistics. *J. Cell. Autom. 5*, 3 (2010), 207–255.

[11] Reiter, C. The game of life on a hyperbolic domain. *Computers and Graphics 21*, 5 (1997), 673–683.

[12] Rendell, P. *Turing universality of the game of life. In: Adamatzky A. (Dds.) Collision-Based Computing*. Springer, 2002.

[13] Rennard, J. P. *Implementation of Logical Functions in the Game of Life. In: Adamatzky A. (eds) Collision-Based Computing*. Springer, London, 2002.

[14] Wolfram, S. *Celular Automata and Complexity*. Addison-Wesley, 1994.