Optimización de RNA que reconoce dígitos

Procesos Estocásticos Aplicados a la Física (Redes Neuronales)

Jorge Velázquez Castro

Rafael Hernández Muñoz

21 de septiembre de 2023

Resumen

Se presentan los resultados, problemas y circunstancias por las que se pasó mientras se optimizaba una red neuronal artificial básica que reconoce dígitos. Se propuso probar varios de los optimizadores tales como SGD+Inercia, RMSprop y Adam. Además de optimizar la función de costo con un Cross-Entropy y aplicando una Soft-Max.

1. Corrección de Errores de Arranque

Durante un buen rato de la semana estuve enfrentandome a un problema con Spyder, pues me arrojaba constantemente un problema cuando corría el "test.py", al intentar de mil maneras como solucionarlo la única manera que funcionó fue cambiarme de IDE. Probé correr los códigos en la consola de Python y en la de Windows pero me seguía arrojando errores. Así que mudé toda la bilioteca a Jupyter Notebooks, esta fue la única opción con la que logré hacer que volviera a funcionar, pero me atrasó mucho en cuestión de tiempos para la entrega.

2. Optimizadores Alternativos

2.1. SGD+Inercia

Una vez en el entorno de Jupyter logré mejorar el código optimizandolo con el método de SGD con inercia. El problema fue solucionar un intento de grafica con Matplotlib, pues en Jupyter tenía una versión mientras que la que estaba usando en Spyder era otra distinta y me arrojaba errores que no eran precizos por lo que hallarlos me fue muy difícil.

A lo que se me fue la hora de entrega y mejor eliminé las gráficas donde solo obtenía errores, por lo que ahora mejor subo el SGD+Inercia funcionando correctamente.

Al momento de buscar tener el tiempo que obtenía con el SGD normal borré un par de indentados y volví a tener problemas con el código original, por lo que ahora solo tengo un valor "Probable"que era de 3:30s mientras que con el SGD+Inercia tendría un tiempo de 2:50s.

Estos valores fueron medidos con el cronómetro de mi celular mientras corría los programas, al agregar los cornómetros al código dejaron de funcionar.

De igual forma queda decir que los intentos se hicieron con 50 epocas y un learning rate de 0.07. Estos valores fueron enontrados buscando manualmente mientras jugaba con los parámetros.

Ya no logré cambiar y probar con valores menores pues tuve un percance de salud y hasta el momento (mañana de miércoles) voy llegando a mi casa de nuevo. Voy a seguir intentando mejorar el código y también ir completanto el reporte, espero en lo que sigue de la noche pueda terminarlo.

2.2. SGD+Inercia. Vol. 1.2

Después de varias horas de corregir problemas de indentación logré que corriera todo el scritp.

Al momento de correrlos, pude hacer una gráfica de los resultados y fuí comparando cada línea de aprendizaje. Decidí entrenar las redes con los siguientes parámetros:

- Epochs = 15
- Minibatch size = 10
- Learning Rate = 0.07
- $\mu = 0.5$ y 0.9

En el primer entrenamiento no recordé que debía ponerle un valor para μ de entre 0.5 y 0.9, entonces no le puse ningún valor, y me arrojó una línea muy buena, con un tiempo bastante logrado. En los entrenamientos siguientes ya usé los valores antes mencionados.

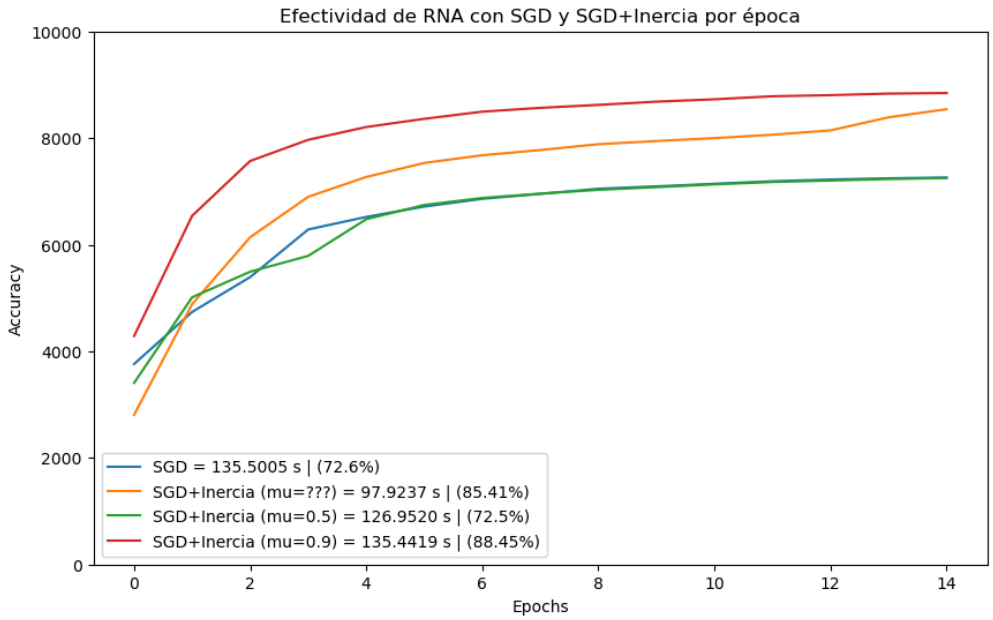

Figura 1:

En la Fig. 1 podemos ver las diferencias de las líneas de aprendizaje que mostraban las distintas modificaciones de los parámetros. Podemos ver que encontré mejor precisión con un valor de $\mu = 0{,}9$ pero tomó más tiempo que cuando no le puse ningún valor al factor de momento, supongo que encontró un valor óptimo por su cuenta.

2.3. RMSprop

Mientras intentaba hacer el ajuste al código para que pudiera correr el RMSprop, me arrojaba errores como si las matrices que estaba usando no fueran del mismo tamaño que las deseadas.

Después de ir imprimiendo cada una de las matrices me dí cuenta que las tuplas de vectores estaban bien. Para después ver que mi error estaba en no intercalar las funciones de la *velocidad de los bias y los weights*, entonces lo realicé pero ahora los resultados estaban en un solo valor, no avanzaba el aprendizaje como se puede ver en la tabla 1. También me arroja un error del tipo red.ipynb:3: RuntimeWarning: overflow encountered in exp".

Epoch	**Resultados**
0	1032 / 10000
1	1032 / 10000
2	1032 / 10000
3	1032 / 10000
4	1032 / 10000
5	1032 / 10000
6	1032 / 10000
7	1032 / 10000
8	1032 / 10000
9	1032 / 10000
10	1032 / 10000
11	1032 / 10000
12	1032 / 10000
13	1032 / 10000
14	1032 / 10000

Cuadro 1: Entrenamiento de la red neuronal con RMSprop

Se supone que el error me esta diciendo que hay numeros muy grandes en la función de exp, pero ya bajé el valor de $\eta = 0{,}001$ y nada. Después de varios intentos logré que aprendiera pues modifiqué los parámetros de la siguiente manera:

- Epochs = 20
- Minibatch size = 15
- Learning Rate = 0.0001
- $\mu = 0.9$

Con lo que logré obtener un porcentaje de acierto de 94.55 %. Después intenté con diferentes tamaños de épocas y también modifiqué el tamaño del minibatch hasta que obtuve los resultados

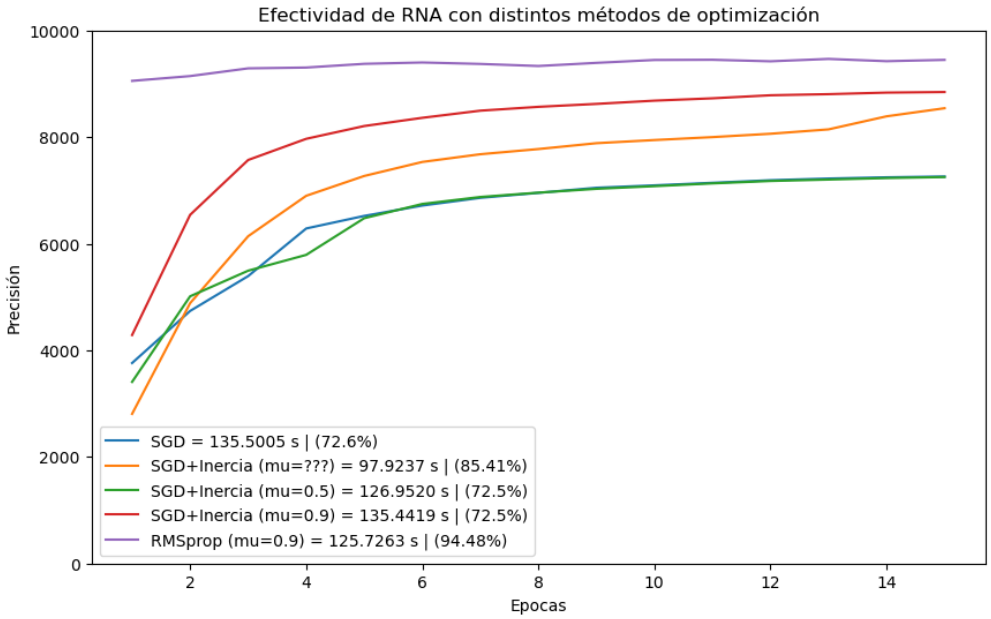

Figura 2:

que se muestran en la Fig. 2. Lo que pude notar es que en ninguna prueba superó el 94 % de efectividad.

Lo mejor es que comienza con mejor cantidad de aciertos, tiene una acertividad nata.

2.4. Problemas de Actualización en GitHub

Como pasé toda mi área de trabajo a Jupyter, para pushear mi progreso descargaba los scripts donde trabajaba con mi código, los convertía a archivo python y los guardaba en la carpeta clonada. Pero al verlos desde mi repositorio en GitHub, noté que se veía como si el único avance fuera el del momento en el que los subía. Todo los pequeños progresos no quedaban guardados cronológicamente. Por lo que decidí hacer un nuevo repositorio donde tuviera clonada la carpeta donde se guardaban los progresos de Jupyter y dejarlos en formato ipynb, de esa forma ahora se puede ver desde el apartado blame, todo las modificaciones que le voy haciendo al código original. Entonces en el repositorio "Tarea1_BasicNetwork" se encuentran la tarea que correspondía a documentar el código, correrlo con el algoritmo SGD, optimizarlo con el $SGD+Inercia$ y con el $RMSprop$. Mientras que en el repositorio "RNAs" están: la optimización con el $Adam$, $Cross-Entropy$ y las siguientes tareas (T3, T4,...).

2.5. Adam

Cuando estaba escribiendo las lineas de codigo que actualizaban los minibatches, tuve varios errores con la indentación pero los solucioné antes de pushear, además de tener un error con que no encontraba la red, pero reiniciando el kernel y borrando las variables que estaba usando logré que funcionara. Me quedé con dudas con el uso de dos B y ya no usar el μ que se usaba en el SGD+Inercia, pues en realidad sería el B_2.

Mientras tanto la Fig. 3 contiene las gráficas del desempeño de los optimizadores y vemos

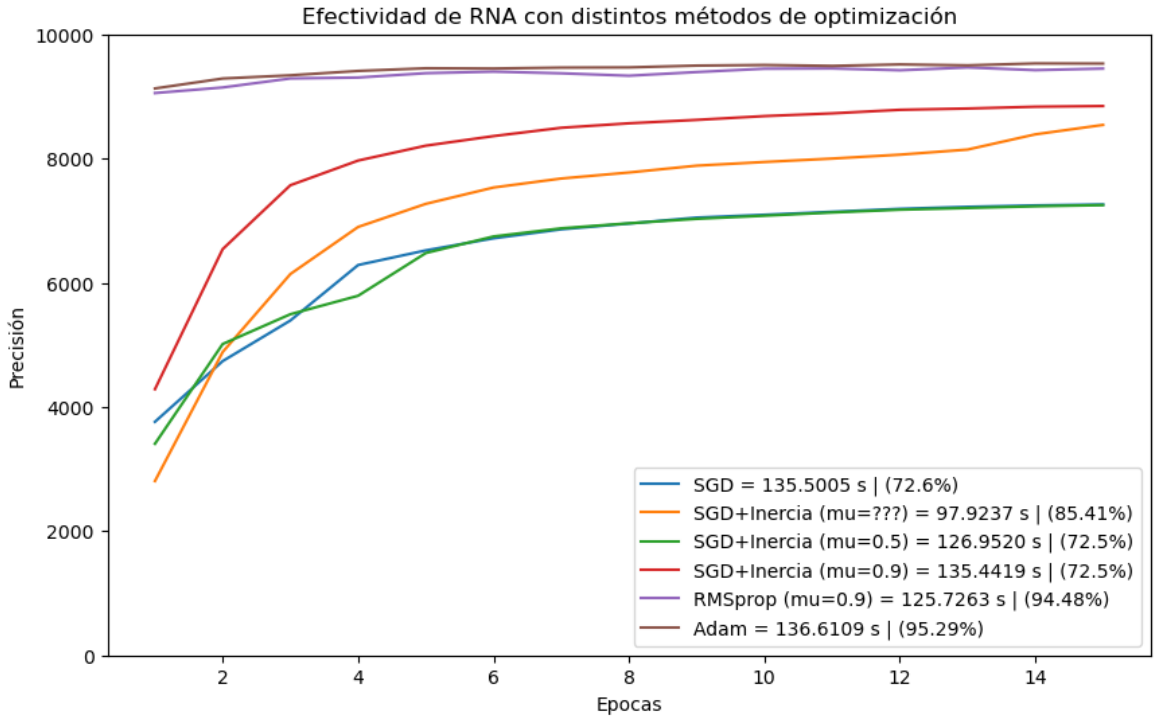

Figura 3:

que el RMSprop y el *Adam* tienen un desempeño muy similar, y solo le gana en poco menos de 1 % de efectividad al terminar el entrenamiento de 15 épocas, con un tamaño del minibatch de 10 y un LearningRate de $\eta = 0{,}01$.