Redes Neuronales Recurrentes (RNN)

Yana Alanoca Cesar Florencio
Universidad Nacional de Ingenieria
Lima, Perú
cesar.yana.a@uni.pe

Rivera Granados Franklin Félix
Universidad Nacional de Ingeniería
Lima, Perú
friverag@uni.pe

***Resumen*—En este proyecto, nos sumergiremos en el mundo de la Inteligencia Artificial, donde sabemos que existen muchos tipos de arquitecturas, que nos ayudan procesar una IA. Para esta investigación, nos centraremos en la arquitectura de Redes Neuronales Recurrentes, donde conoceremos a fondo, las ventajas y desventajas del uso de este nuevo tipo de RN, también conoceremos sus diferentes tipos de RNN, luego implementaremos una RNN, para posteriormente entrenarlo y por último realizaremos la demostracion del funcionamiento de este tipo de RN.**

Índice de Términos— RNN.

I. Introducción

Desde la aparición de la inteligencia artificial, se ha comprendido que esta tecnología es capaz de aproximarse o parecerse muchísimo a la forma de pensar de un ser humano, esto ocurre, porque esta compuesto de algoritmos muy complejos, pero que a su vez, permiten obtener un resultado confiable. Una de las arquitecturas mas eficaces, es el uso de las redes neuronales, sin embargo existen varios tipos de redes neuronales, pero que se diferencian en la cantidad de datos que se procesen en ese mismo instante.

Las redes neuronales tradicionales, se les tiene que dar un dato y esta RN, lo procesara y arrojara un resultado, pero el inconveniente es que no pueden procesar una secuencia de datos, a comparación las redes neuronales recurrentes, que si pueden procesar muchos datos en secuencia, como una secuencia de voz(música), secuencia de imágenes(video). La creación de las redes neuronales (RNN), han contribuido mucho en la actualidad, como la creación de traductores de texto, reconocimiento de texto, reconocimiento de imágenes, etc.

Para la creación de esta red neuronal recurrente, se usará el lenguaje Python, también deberemos utilizar las librerías propias de pyhton.

I-A. Objetivos

- Mencionar y mostrar los diferentes tipos de RNN
- Crear una red neuronal recurrente en Python
- Entrenar nuestra red neuronal recurrente
- Mostrar el funcionamiento de la red neuronal recurrente

II. Redes Neuronales Recurrentes

En primer lugar definamos a qué nos referimos con **instante de tiempo.** Para una secuencia, el instante de tiempo es simplemente un número entero que define la posición de cada elemento dentro de la secuencia.

Así por ejemplo, en la palabra "d-i-p-l-o-s-a-u-r-i-o", el instante de tiempo 1 correspondiente al primer carácter de la secuencia(la letra "d"), el instante de tiempo 2 al segundo carácter (la letra "i") y así sucesivamente.

Nos referimos a x_t como la entrada a la red recurrente en el instante de tiempo t, y a y_t como la salida en el instante de tiempo t.

II-A. La Red Recurrente: entradas y salidas

Pero, ¿Cómo logra la Red Recurrente predecir correctamente el siguiente caracter en la secuencia? Es decir, ¿dónde está la memoria que mencionamos anteriormente?

La respuesta precisamente en un elemento importante que observamos en la figura anterior: las flechas horizontales de color rojo. Se oberva que en cada instante de tiempo la red tiene realmente dos entrada y dos salidad.

Las entradas son el dato actual (x_t) y la activación anterior (a_{t-1}) mientras que las salidas son la predicción actual (y_t) y la activación actual (a_t). Esta activación también recibe el nombre de "hidden state.º estado oculto.

Son estas las activaciones las que corresponden precisamente a la memoria de la red, pues permiten **preservar y comprtir la información entre un instante de tiempo y otro**.

Veamos cómo se genera la predicción y la activación, y cómo estas se relacionan con la memoria de la red.

II-B. La Red Recurrente: funcionamineto detallado

Para calcular la salida y la activación de la Red Recurrente, a partir de sus dos entradas, se usa la misma lógica de una Neurona Artificial convencional.

En decir una Red Neuronal Recurrente tiene una entrada (x) y genera una salida (y), y que la salida es el resultado de aplicar dos operaciones al dato de entrada: una transformación y una función de activación no-lineal:

$$X \Rightarrow neurona\_artificial \Rightarrow y = f(WX + b)$$

En el caso de las Redes Recurrentes, la activación se calcula de manera similar, y es el resultado primero de

transformar los datos de entrada (es decir la activación anterior y la entrada actual) y luego llevarlos a una función de activación no-lineal:

$$a_{t-1} \Rightarrow RNN \Rightarrow a_t = f(W_{aa}a_{t-1} + W_{aX}X_t + b_a)$$

¿Y cómo se calculan los valores más adecuados de los coeficientes W y b? Con el mismo procedimiento usado en las Redes Neuronales, es decir a través del entrenamiento:

$$a_{t-1} \Rightarrow RNN \Rightarrow a_t = f(W_{aa}a_{t-1} + W_{aX}X_t + b_a)$$

De igual forma, para obtener la salida, se usa la activación del instante previo y se realizan las mismas operaciones (transformación y función de activación) aplicadas anteriormente:

$$y_t = g(W_{ya}a_t + b_y)$$

donde g es la función de activación

$$a_{t-1} \Rightarrow RNN \Rightarrow a_t = f(W_{aa}a_{t-1} + W_{aX}X_t + b_a)$$

Al igual que en el caso anterior, los coeficientes W y b para el cálculo de esta salida se obtienen durante el proceso de entrenamiento.

Si observamos detalladamente estas dos ecuaciones, veremos el concepto de recurrencia y la memoria asociada a las redes recurrentes:

La salida y_t depende de la activación actual (a_t) pero a su vez, dicha activación depende no solo de la entrada actual (x_t) sino igualmente del valor previo de la activación (a_{t-1}). ¡Esta es precisamente la memoria de la red! Y la forma como este concepto permite preservar y compartir la información entre uno y otro instante de tiempo.

III. Estado Del Arte

III-A. Redes Neuronales Recurrentes Para El Análisis De Secuencias

En este artículo con el mismo nombre Redes Neuronales Recurrentes Para El Análisis De Secuencias [1] se estudia de la importancia de las redes neuronales recurrentes y de su naturaleza en análisis de secuencias o señales donde es muy importante tener en cuenta el pasado o el futuro.

Se muestra la fortaleza de estos métodos para analizar secuencias de tamao variable, por su despliegue en el tiempo en función del tamaño de la entrada. Se construyen específicamente redes neuronales recurrentes bidireccionales, como una especificación de redes recurrentes, mostrando la potencialidad de las mismas en sistemas no causales, donde las entradas pueden depender de entradas de tiempos pasados y futuros. Además se desarrolla Una plataforma para implementar redes recurrentes dinámicas, con algoritmo de aprendizaje Backpropagation Trough Time; que permiten desarrollar redes para cualquier problema donde las entradas son secuencias analizadas en el tiempo y la salida son otras secuencias o simplemente descriptores de funciones o propiedades de las mismas.

III-B. Modelos De Redes Neuronales Recurrentes En Clasificación De Patentes

En este artículo del mismo nombre Modelos De Redes Neuronales Recurrentes En Clasificación De Patentes [2] se centra en la clasificación automática de patentes usando redes LSTM (Long-Short Term Memory). Para empezar se describen distintos métodos de tratamiento y extracción de características de textos y de clasificación por aprendizaje automático.

Se establece una metodología de desarrollo y pruebas de modelos para poder evaluar con facilidad los distintos experimentos. Dentro de esta metodología se incluye un framework de flujo de datos y entrenamiento de modelos así como pequeñas utilidades para poder replicar todos los experimentos en entornos virtualizados con docker pero con acceso a GPUs. Además, este framework realiza un guardado automático de los resultados intermedios de los distintos módulos de procesado de texto (que también ha habido que implementar) para así conseguir que operaciones costosas previas al entrenamiento se ejecuten solo cuando sea necesario y no más de una vez.

Se realizaron unos experimentos con distintas variaciones de modelos con LSTM, algunos con un mapeo precalculado y otros entrenando un mapeo de cero. La gran mayoría fueron poco conclusivos y no tuvieron resultados muy buenos, pero dejan abierta la puerta a sencillas mejoras e indican los pasos a seguir para mejorar la precisión.

III-C. Redes Neuronales Convolucionales y Redes Neuronales Recurrentes

En este artículo de nombre Redes neuronales convolucionales y redes neuronales recurrentes en la transcripción automática [3] trata el problema de que las transcripciones varian de persona en persona, dependiendo de su experiencia y habilidad, siendo una más acertada uno que otros. La forma en que se ha buscado solucion a este problema es con el uso de redes neuronales, específicamente redes neuronales convolucionales y redes neuronales recurrentes. Con el uso de ambos tipos de redes neuronales se propone en este artículo una metodología mixta que permita automatizar el proceso de transcripcion de una lengua.

En esta investigación se encontró que la forma en que se pueden combinar estas dos arquitecturas es dejando a las redes neuronales convolucionales extraer características acusticas de alto nivel, mientras que a las redes neuronales recurrentes se les asigna la tarea de clasificar secuencias. Así se reducira la posibilidad de error en

las transcripciones eliminando el factor subjetivo que subyace al trabajo hecho por humanos.

IV. Metodología

En este trabajo sobre Redes Neuronales Recurrentes(RNN), describiremos todas las herramientas que usaremos, para lograr construir, entrenar, a una RNN, y por ultimo, mostrar su funcionamiento.

- Numpy
  Es una libreria de python [4], que permite la creacion de cálculos, usada frecuentemente en operaciones con matrices.
  Para nuestro trabajo de RNN, esta libreria la usaremos para poder ajustar nuestro generador aleatorio de nuestra semilla o neurona, y asi poder garantizar que nuestra neurona tenga la capacidad de reproducirse por si sola.
- Keras Es un framework de aprendizaje [5], diseñado en Python, compatible con otros framework, como, TensorFlow,CNTK. Creado con el fin de facilitar los procesos de experimentacion rapida, como lo que es RN.
  En nuestro trabajo, importaremos Keras para poder usar las funciones, Input y Dense, con el fin de crear una celda recurrente.
  Tambien nos ayduara a importa nuestra SGD, para el entrenamiento de nuestra RNN
  Por ultimo, tambien nos ayudara a importar to_categorical y el backend, logrando representar nuestra entrada y salida, durante el entrenamiento y la prediccion.
- Gradiente Descendiente(SGD) Es un enfoque, conocido por ser simple [6] pero a su vez, tambien es muy eficaz, en cuanto a aprendizaje discriminatorio en clasificadores lineales, como funciones de perdida convexa y regresion logistica.
  En nuestro RNN, nos ayudara a poder creae y poder entrenar, esta ultima.

Metodologia del trabajo

- Importaremos las librerias numpy, para garantizar la reproducidad en el entrenamiento.
- Despues importamos keras, para usar las funciones input, dense, y simpleRnn
- Luego importamos la funcion model y gradiente descendiente, ayudando a crear y representar el modelo.
- Y por ultimo, importamos el bakend y ”to_categorical ”, para que exista una entrada y una salida.

V. Experimentación y resultados

VI. Discusión De Resultados

VII. Conclusiones

VIII. Trabajos Futuros

Referencias

[1] Cruz, I. B., Martínez, S. S., Abed, A. R., Ábalo, R. G., Lorenzo, M. M. G. (2007). Redes neuronales recurrentes para el análisis de secuencias. Revista Cubana de Ciencias Informáticas, 1(4), 48-57.

[2] Guridi Mateos, Guillermo. Modelos de redes neuronales recurrentes en clasificación de patentes. BS thesis. 2017.

[3] Prieto, Juan Sebastian Gelvez. Redes neuronales convolucionales y redes neuronales recurrentes en la transcripción automática.”

[4] Numpy: https://aprendeconalf.es/docencia/python/manual/numpy/

[5] Keras: https://enmilocalfunciona.io/deep-learning-basico-con-keras-parte-1/

[6] Gradiente Descendiente Estocastico: https://unipython.com/descenso-gradientes-estocastico-sgd/