Programa académico: Ingeniería en Inteligencia Artificial
Unidad de aprendizaje : Aplicaciones del Lenguaje Natural
Nombre del Alumno:_____
Fecha de entrega:_____
Calificación:_____

Practica de laboratorio 4: Redes neuronales recurrentes

Objetivo: Analizar el funcionamiento de las redes neuronales recurrentes mediante la implementación de 2 códigos basados en una serie temporal y un texto. El texto denota la serie de palabras consecutivas para encontrar relaciones a priori y generar datos a posteriori con una base (semilla) de texto, frase o palabras.

A) Marco teórico

Las redes neuronales recurrentes (RNN) son una familia de redes adecuadas para el aprendizaje de representaciones de datos secuenciales como el texto en el Procesamiento del Lenguaje Natural (PLN).

La idea que subyace a las RNN es aprovechar la información secuencial.

En una red neuronal tradicional suponemos que todas las entradas (y salidas) son independientes entre sí.

Pero para muchas tareas esa es una muy mala idea. Si queremos predecir la siguiente palabra de una frase, es mejor que sepamos qué palabras la preceden.

Las RNN se llaman recurrentes porque realizan la misma tarea para cada elemento de una secuencia, y la salida depende de los cálculos anteriores.

Otra forma de pensar en las RNN es que tienen una "memoria" que captura información sobre lo que se ha calculado hasta el momento.

En teoría, las RNN pueden utilizar información de secuencias arbitrariamente largas, pero en la práctica se limitan a mirar hacia atrás sólo unos pocos pasos.

Las RNN han sido ampliamente utilizadas por la comunidad del procesamiento del lenguaje natural (PLN) para diversas aplicaciones. Una de ellas es la construcción de modelos lingüísticos. Un modelo de lenguaje nos permite predecir la probabilidad de una palabra en un texto dadas las palabras anteriores. Los modelos de lenguaje son importantes para diversas tareas de nivel superior, como la traducción automática, la corrección ortográfica, etc.

Datos usados

Datos aleatorios generados y libro Alicia en el país de las Maravillas formato txt obtenido en Proyecto Gutenberg.

B) Procedimiento:

Implementar el código 1.

La estructura de la RNR consiste en diferentes pesos y sesgos asociados a las distintas capas, realizan los cálculos para obtener la salida cuando se da una entrada. En el código 1 se presenta una aplicación completa para la predicción de series temporales.

El modelo consiste en dos unidades ocultas creadas mediante la capa SimpleRNN y una unidad densa creada mediante la capa Dense. La forma_de_entrada se establece en 3×1, y se utiliza una función de activación lineal en ambas capas por simplicidad. Sólo para recordar, la función de activación lineal no hace ningún cambio en la entrada.

Dado la configuración de las capas SimpleRNN y Dense, se procede a ejecutar una RNR completa en un conjunto de datos de series temporales simples, se tiene que seguir los siguientes pasos

- ☐ *Leer el conjunto de datos desde una URL dada*
- ☐ *Dividir los datos en conjuntos de entrenamiento y prueba*
- ☐ *Preparar la entrada en el formato Keras requerido*
- ☐ *Revisar modelo RNN y entrenarlo*
- ☐ *Realizar las predicciones en los conjuntos de entrenamiento y prueba e imprimir el error cuadrático medio en ambos conjuntos*
- ☐ *Ver los resultados*

Implementar el código 2.

En este ejemplo, se entrena un modelo de lenguaje basado en caracteres sobre el texto de Alicia en el País de las Maravillas para predecir el siguiente carácter a partir de 10 caracteres anteriores. Se ha optado por construir un modelo basado en caracteres porque tiene un vocabulario más reducido y se entrena más rápido.

La idea es la misma que la de utilizar un modelo de lenguaje basado en palabras, pero utilizando caracteres en lugar de palabras.

Ahora estamos construyendo una RNN a nivel de carácter. En este caso, el vocabulario es el conjunto de caracteres que aparecen en el texto.

Se realiza el proceso mediante los índices de estos caracteres en lugar de los caracteres en sí (tablas de búsqueda) , posteriormente se crean los textos de entrada y etiquetas para poder vectorizar y construir el modelo neuronal.

La RNR está conectada a una capa densa (totalmente conectada), la capa densa tiene unidades (nb_char), que emiten puntuaciones para cada uno de los caracteres del vocabulario.

La activación en la capa densa es un softmax, que normaliza las puntuaciones a probabilidades, el carácter con la probabilidad más alta se elige como predicción.

Se compila el modelo con la función de pérdida de entropía cruzada categórica, una buena función de pérdida para resultados categóricos, y el optimizador RMS prop.

El modelo consiste en generar un carácter a partir del modelo dado una entrada aleatoria, luego eliminar el primer carácter de la entrada y añadir el carácter predicho de nuestra ejecución anterior, y generar otro carácter a partir del modelo.

Se continua con esto 100 veces (NUM_PREDS_PER_EPOCH=100) y se genera e imprime la cadena resultante, la cadena nos da una indicación de la calidad del modelo (si la cadena tiene sentido o no).

Se observa que el modelo se basa en caracteres y no conoce las palabras, pero aprende a deletrear palabras que parecen sacadas del texto original.

Dado el contexto de la implementación del código seguir los siguientes pasos:

- ☐ *Leer el conjunto de datos*
- ☐ *Preparar la entrada*
- ☐ *Revisar modelo RNN y entrenarlo*
- ☐ *Realizar las predicciones en los conjuntos de entrenamiento*
- ☐ *Ver los resultados*

Reportar todas las observaciones realizadas que sean relevantes de los códigos implementados, en caso de ser necesario implementar ajustes en la red neuronal para incrementar el rendimiento del algoritmo.

C) Resultados

D) Análisis de Resultados

E) Conclusiones

F) Referencias bibliográficas