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

Practica de laboratorio 2: Aplicaciones de redes neuronales con Texto

A) Marco teórico

El programa con el que va trabajar se adjunto en classroom.

En este ejemplo se clasifica las críticas de películas como positivas o negativas utilizando el texto de la crítica. Se trata de un ejemplo de clasificación binaria o de dos clases, un tipo de problema de aprendizaje automático importante y ampliamente aplicable.

Se utilizará el conjunto de datos IMDB, que contiene el texto de 50.000 críticas de películas de Internet Movie Database. Se dividen en 25.000 críticas para el entrenamiento y 25.000 críticas para las pruebas. Los conjuntos de entrenamiento y prueba están equilibrados, lo que significa que contienen el mismo número de críticas positivas y negativas.

El conjunto de datos IMDB viene empaquetado con TensorFlow, ya ha sido preprocesado de forma que las reseñas (secuencias de palabras) **se han convertido en secuencias** de enteros, donde cada entero representa una palabra específica en un diccionario.

Exploración de los datos

El conjunto de datos viene preprocesado: cada ejemplo es una matriz de números enteros que representan las palabras de la crítica de la película. Cada etiqueta es un valor entero de 0 o 1, donde 0 es una crítica negativa y 1 es una crítica positiva.

Además el texto de las reseñas se ha convertido en números enteros, donde cada número entero representa una palabra específica en un diccionario. Las reseñas de las películas pueden tener distintas longitudes, dado que las entradas de una red neuronal deben tener la misma longitud, se tiene que resolver este inconveniente.

Preparar los datos

Las reseñas (matrices de enteros) deben convertirse en tensores antes de introducirlas en la red neuronal. Esta conversión puede hacerse de dos maneras:

Convertir las matrices en vectores de 0s y 1s que indiquen la aparición de palabras, de forma similar a una codificación de una sola palabra. Por ejemplo, la secuencia [3, 5] se

convertiría en un vector de 10.000 dimensiones que es todo ceros excepto los índices 3 y 5, que son unos. Esta tarea se realiza en la primera capa de la red -una capa Densa- que pueda manejar datos vectoriales en coma flotante. Sin embargo, este enfoque consume mucha memoria, ya que requiere una matriz de tamaño num_words * num_reviews.

Como alternativa, se puede rellenar las matrices para que todas tengan la misma longitud y, a continuación, crear un tensor entero de forma max_length * num_reviews. Podemos utilizar una capa de incrustación capaz de manejar esta forma como la primera capa de la red.

Dado que las críticas de películas deben tener la misma longitud, utilizar la función pad_sequences para estandarizar las longitudes.

Las contribuciones de mayor magnitud tienen un mayor impacto en la predicción del modelo. Las contribuciones negativas indican que el valor de la característica para este ejemplo dado redujo la predicción del modelo, mientras que los valores positivos contribuyen a un aumento de la predicción.

B) Procedimiento:

Construir el modelo

Las capas se apilan secuencialmente para construir el clasificador:

La primera capa es una capa de incrustación (embedding), esta capa toma el vocabulario codificado con números enteros y busca el vector de incrustación para cada índice de palabra. Estos vectores se aprenden a medida que el modelo se entrena. Los vectores añaden una dimensión a la matriz de salida. Las dimensiones resultantes son: (lote, secuencia, incrustación).

A continuación, una capa ***GlobalAveragePooling1D*** devuelve un vector de salida de longitud fija para cada ejemplo promediando sobre la dimensión de secuencia. Esto permite al modelo manejar entradas de longitud variable de la forma más sencilla posible.

Este vector de salida de longitud fija se canaliza a través de una capa totalmente conectada (densa) con 16 unidades ocultas.

La última capa está densamente conectada con un único nodo de salida. Utilizando la función de activación sigmoide, este valor es un flotador entre 0 y 1, que representa una probabilidad o nivel de confianza.

Capas ocultas

El modelo tiene dos capas intermedias u "ocultas", entre la entrada y la salida, el número de salidas (unidades, nodos o neuronas) es la dimensión del espacio de representación de la capa. En otras palabras, el grado de libertad que se permite la red a la hora de aprender una representación interna.

Si un modelo tiene más unidades ocultas (un espacio de representación de mayor dimensión), y/o más capas, entonces la red puede aprender representaciones más

complejas. Sin embargo, esto hace que la red sea más costosa desde el punto de vista computacional y puede conducir al aprendizaje de patrones no deseados, es decir, patrones que mejoran el rendimiento en los datos de entrenamiento pero no en los datos de prueba, esto se denomina sobreajuste.

Función de pérdida y optimizador

Un modelo necesita una función de pérdida y un optimizador para el entrenamiento, como se trata de un problema de clasificación binaria y el modelo produce una probabilidad (una capa de una sola unidad con una activación sigmoide), utilizar la función de pérdida binary_crossentropy.

Esta no es la única opción para una función de pérdida, usted podría, por ejemplo, elegir mean_squared_error. Pero, en general, binary_crossentropy es mejor para tratar con probabilidades: mide la "distancia" entre las distribuciones de probabilidad o, en nuestro caso, entre la distribución real y las predicciones.

Crear un conjunto de validación

Al entrenar, queremos comprobar la precisión del modelo en datos que no ha visto antes. Cree un conjunto de validación separando 10.000 ejemplos de los datos de entrenamiento originales.

¿Por qué no utilizar ahora el conjunto de prueba?

El objetivo es desarrollar y ajustar el modelo utilizando sólo los datos de entrenamiento y, a continuación, utilizar los datos de prueba una sola vez para evaluar la precisión.

Entrenar el modelo

Entrené el modelo durante 40 épocas en minilotes de 512 muestras, esto supone 40 iteraciones sobre todas las muestras de los tensores x_train e y_train. Durante el entrenamiento, controle la pérdida y la precisión del modelo en las 10.000 muestras del conjunto de validación.

Modifique el número de muestras y el número de épocas, reporte cambios en los parámetros de salida en el apartado de resultados.

Evaluar el modelo

Observar cómo se comporta el modelo, se devolverán dos valores de pérdida (un número que representa el error, los valores más bajos son mejores), y precisión, reportar en la sección de resultados. Que representan estos resultados obtenidos (indicar en la sección de resultados). Proponer cambios o mejoras para reducir los valores de pérdida.

C) Resultados

D) Análisis de Resultados

E) Conclusiones

F) Referencias bibliográficas