**30-06-2021**

Convolución en una dimensión: una capa, filas y columnas

Convolución 2D: imágenes, sonido, series temporales

Cada capa realiza una función en función de la anterior. Para imágenes es mejor empezar con más neuronas (se centran en detalles más difusos, menos definidos) y terminar con menos neuronas (se fijan en detalles más completos)

Buena práctica hacer el reshape con la estructura de la base reshape(nº filas,nº columnas)

Con el Dense le decimos el número de neuronas que queremos que tenga en cada capa.

Red neuronal recursiva

No todo va hacia adelante. Hay veces que la neurona se retroalimenta a sí misma, la neurona posterior le da información a la anterior o le da información a otra neurona que está en su misma capa. Hace que una red neuronal mantenga información no solo para el presente sino también para el pasado.

Lstm: Long short term members

Los datos de entrenamiento son menores que los de test, que harán referencia al futuro. Cuando el pasado es parecido al futuro (ej. temperaturas en agosto en diferentes años)

Tp.- puntos de entrenamiento

Se le da el número de veces que queremos que se repita el último valor. Steps = 4. Cuanto menos steps menos datos va a tener para predecir el futuro. Cuanto más steps tendrán más datos para predecir pero tb más probabilidad de equivocarse al hacer la predicción.

Permite que pueda entrenar con el pasado más reciente que puede ser el que mejor prediga lo que vaya a ocurrir después

Habrá tantos batches como filas haya en el conjunto de entrenamiento. Cada batch estará constituido Spor el número de steps que le hayamos dado

Se pasa a matriz el conjunto de datos

Cuando los datos pueden cambiar a lo largo del tiempo debido a eventos imprevistos (ej. la bolsa), es mejor utilizar un modelo en el que el tamaño de la muestra de entrenamiento respecto a la de test va aumentando progresivamente

Summary. Red neuronal recursiva en series temporales

1. Analizamos los datos y vemos si se repiten comportamientos a lo largo del tiempo
2. Decidimos el tipo de entrenamiento que vamos a utilizar:
   1. Si los datos se repiten a lo largo del tiempo entrenamos el modelo dándole como dato de entrenamiento una parte muy representativa de todo el conjunto de datos
   2. Si no se repiten los datos a lo largo del tiempo, entrenamos dándole poco a poco todo el conjunto de datos
3. Repetimos los steps (últimos valores del conjunto de entrenamiento y test)
4. Transformamos o redimensionamos los datos de entrenamiento y test para que tengan la forma de (Nº batches, 1, step). Step = embedding
5. Entrenamos el modelo utilizando RNN, simpleRNN, LSTM
6. Dependiendo del resultado probamos diferentes arquitecturas en tamaño de ventana (step), bath-size, epoch,
7. Seleccionamos el mejor modelo

Sparse: una neurona una salida. Da la probabilidad

Sparse categorical: varias neuronas, varias salidas.- Nos da el índice, la clase de cada neurona. Softmax se aplica a sparse categorical

Github copy pidiendo un código verbalmente te lo crea

Para detectar anomalías en una serie temporal se calcula la media y la desviación típica. Lo habitual es la media + la desviación típica. Lo que supera en un 10% la desviación típica se considera una anomalía. El umbral se establece por el investigador

**01-07-2021**

**Procesamiento de lenguaje natural**

Hay que codificar las palabras. De esta forma se puede entrenar el modelo con array

padding permite que todas las frases tengan la misma longitud. En las de menor longitud se completará con 0 las columnas que no corresponden con ninguna palabra

Con embedding se agrupan las palabras por cercanía. Dependiendo de las distancias en el texto

Al tokenizar pasamos de un string\_frase a un string\_palabras. Posteriormente se pasa a número cada palabra de la frase. Después con padding se crean los arrays con el mismo número de elementos. Con embedding se asigna un número (peso) a cada array en función de las distancias entre las palabras que contiene.

Al final lo que hace es clusterizar palabras para que estén en el clúster adecuado

Algoritmos que asignan pesos a las palabras de forma diferente: Cbow (le llegan las palabras) y skip-gram (le llegan las frases)

Herramientas de lenguaje natural: NLTK. Tokeniza e identifica si cada palabra en la frase es un verbo, complemento directo, etc

sent\_tokenize.- tokeniza frases, separa frases. Le asignará pesos a cada frase a lo largo de todo el proceso de entrenamiento

Word\_tokenize.- tokeniza palabras, separa palabras. Va a asignar pesos a cada palabra a lo largo del todo el proceso de entrenamiento

Una vez que se tokenizan las palabras o frases, con gemsin hacemos un array del tamaño que establezcamos (size = 100)

min\_count.- mínimo número de veces para que una palabra aparezca en el entrenamiento

Gemsim.-

Intenta ajustar los pesos y en función de eso

TensorFlow\_Keras

Eliminar el fichero NLP\_sentiments de la carpeta week 12

Take = 1 va a coger el batch\_size que le damos (ej. 1024; el take =1, tomará 1024 frases)

Se pueden reducir el conjunto de palabras del texto a x dimensiones

Para seleccionar una parte de un conjunto de texto no se puede hacer como con Python (x[o]) sino con una función especifica de tensorflow en el que se le da el ‘take’

En vez de maxpool se utiliza globalAveragepooling

----------------------------------------------------------------

Si se trabaja con imágenes, todas ellas han de tener la misma resolución

**02-07-2021**

Como leer un archivo de imágenes (read\_image). Igualar la resolución de todas ellas a (211,211)

ImageDataGenerator.- Permite modificar las imágenes rotándolas, …

Utilizar redes preentrenadas.

Transfer learning.- coger una red entrenada y utilizarla diciéndole que se entrenen por ejemplo las últimas capas. Al entrenarlas partes de que ya conocen lo que vas a entrenar. Se van a concatenar dos redes, una entrenada y otra no entrenada.

Estas imágenes han sido entrenadas con miles de imágenes diferentes (multitud de temas, áreas). La red se entrenó con imágenes a color con muy alta resolución. Reescalará tus imágenes para que tengan la escala de la dimensión original de las redes ya entrenadas.

Importamos ResNet50V2

Al trabajar con ella le decimos que nos descargue todos los pesos que se le han asignado

Coger las imágenes de dos carpetas diferentes. Una de ellas será X y otra y

Transfer learning.- le decimos que capas no queremos que sean entrenables

Concatenamos la base entrenada con la nuestra

Autoencoders.- nos permite disponer de los pesos que se utilizan para modificar la imagen y ponerla con menor información