Modelos de Secuencia. Todo lo que necesitas es atención

Francisco Sanz Machine Learning

Los modelos de secuencias (en inglés sequence models) son las técnicas utilizadas cuando el orden y la secuencia de los datos aportan mucho valor predictivo. (Aunque están algo relacionados, no lo confundáis con las series temporales).

Índice

¿A qué nos referimos con secuencia?

> *Una secuencia es una serie de elementos que se suceden unos a otros y guardan relación entre sí.*

Definiciones de Oxford Languages

Voy a transcribirlo a un ejemplo fácil de asimilar. Cuando alguien nos pregunta el abecedario sabemos responder rápidamente sin dudarlo. Sin embargo, cuando tenemos que empezar el abecedario por una letra aleatoria nos cuesta un poco más, y si tenemos que decirlo de atrás para delante ya ni te cuento. Esto se debe a que tenemos metido en la cabeza el abecedario como una secuencia de letras y lo tenemos memorizado as?. Esto es porque tanto **las letras que preceden como las posteriores nos aportan mucha información** para saber cuál toca.

¿Para qué tipo de problemas se utilizan los modelos de secuencia?

Después del ejemplo del abecedario igual os habéis dado cuenta ya. **Esta tipología está presente en casi cualquier problema de la vida** real. Los ejemplos que se me vienen ahora a la cabeza son:

- Reconocimiento de actividad en vídeo: saber cuándo una persona está corriendo, saltando, hablando etc.
- Análisis de la secuencia de ADN
- Generación de música, de diálogo etc.

- **Reconocimiento de voz** (Speech recognition). Ahora muy de moda con todos los asistentes por voz como Alexa, Siri, Google Assistant…
- Análisis de sentimiento: saber cuando algo tiene un sentimiento positivo, negativo, neutro.
- **Traducción de texto**: el mítico Google Translate.
- Un gran largo etc.

Quiero destacar, que un campo del aprendizaje automático en el que **los modelos secuenciales son claves es el de NLP** (Natural Language Processing – Procesado Natural de Lenguaje). Esto se debe a que en el lenguaje natural la secuencia de palabras es muy importante (es decir, el contexto tiene un valor predictivo tremendamente grande).

¿Por qué no podemos utilizar modelos tradicionales o redes neuronales normales?

Las redes neuronales estándar (NN a partir de ahora) presentan una serie de debilidades a la hora de afrontar este tipo de problemas:

1. Las NN están diseñadas para tener siempre el mismo tamaño de input y de output siempre. Sin embargo, **en los problemas secuenciales**, **los inputs y los outputs pueden tener tamaños muy diferentes dependiendo de la observación**. Por ejemplo a la hora de traducir, cuando queremos trabajar tres palabras tenemos un input y output mucho más pequeños que cuando queremos traducir un párrafo.
2. Las NN no comparten características entre las diferentes posiciones. Es decir, **las NN asumen que cada input (y output) es independiente uno del otro**. Cuando esto no es verdad cuando estamos trabajando con secuencias, como hemos visto un poco más arriba.

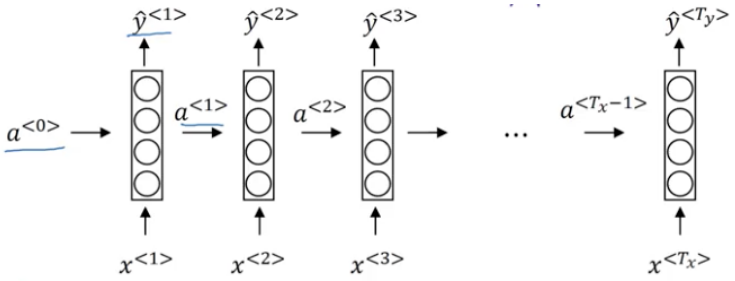

Imagen sacada de la especializaci?n de DeepLearning.ai

Arquitecturas neuronales utilizadas en los Secuence Models

Hay un montón de redes diferentes que se utilizan para trabajar con secuencias y cada vez hay más. Mi intención con este post es explicar lo básico para que sea más fácil asimilar conceptos más complicados más adelante.

Redes Neuronales Recurrentes

Las Recurrent Neural Nets (RNN) son una familia de redes en las que se comparten características entre nodos. Es decir, estas neuronas **permiten utilizar salidas anteriores como entradas de otros nodos** a la vez que se mantienen capas ocultas en la arquitectura. El hecho de compartir estas características nos proporciona una serie de ventajas y desventajas:

Ventajas

Las ventajas clave que ofrecen las RNN son, entre otras:

- Ofrecen la posibilidad de procesar entradas (y salidas) de cualquier longitud, sin que estas longitudes afecten al tamaño del modelo
- Los pesos se comparten a lo largo del tiempo

Desventajas

Las RNN presentan una computación es lenta, no se pueden considerar entradas futuras para el estado actual y es **difícil acceder información de estados muy antiguos**.

Pero este tipo de redes sufre de otro problema importante. La **desaparición/explosión del gradiente**. Estos fenómenos son frecuentes cuando utilizamos RNN. Se debe a que es complicado capturar las dependencias a largo plazo debido a la disminución/aumento exponencial del gradiente a medida que creamos más capas.

Tipos y aplicaciones

Como dije, las RNN son una familia de redes. Es decir, hay un montón de tipos diferentes (dependiendo de las longitudes de entrada y de salida) y cada tipo tiene una aplicación diferente:

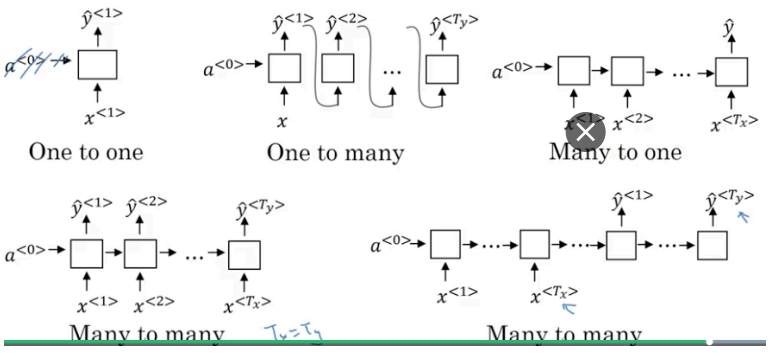

Imagen sacada de la especializaci?n de DeepLearning.ai

- **One-to-One**: la NN clásica.
- **One-to-Many**: este tipo es en el que teniendo un único input la red devuelve múltiples salidas. Una aplicación típica es la generación de música.
- **Many-to-One**: tenemos muchas entradas a la red pero sólo un output. Por ejemplo en análisis de sentimiento, en el que tenemos muchos inputs (cada palabra cuenta como uno) y una única salida.
- **Many-to-Many** (mismo número de entradas que de salidas): se utilizan para el reconocimiento de nombres.
- **Many-to-Many** (diferente número de entradas y de salidas): para traducciones, en el que el texto en idioma original no tiene por qué tener el mismo número de palabras que el traducido.

Cómo resolver los problemas de gradiente: GRU

Las **Gated-Recurrent-Units (GRU)** fueron introducidas por primera vez en 2014 por Cho *et al*. en este paper publicado en 2014 y por Chung *et al.* en este otro paper. Este tipo de NN se introdujeron con el **objetivo de solventar** el problema de **desaparición de gradiente**. Esto lo consiguen agregando dos puertas (puertas de actualización y restablecimiento, o en inglés **Update** y **Reset**) que mantienen un registro de la información más relevante para la predicción. Básicamente, son dos vectores que deciden qué información se pasa a la salida.

La **puerta de Update** determina **cuánta información** previa se debe transmitir **al siguiente paso** de tiempo. La **puerta de Reset** se utiliza para determinar **cuánta información** es irrelevante y **debe olvidarse**.

Cómo resolver la memoria a largo plazo: LSTM

Uno de los problemas principales de las RNN que he destacado antes, es que es **difícil acceder información de estados muy**

antiguos. En los casos en los que necesitamos un gran contexto para poder predecir, las RNN no son muy eficientes.

> *Cuando intentamos predecir la siguiente palabra, es muy posible que necesitemos el contexto entero. Por ejemplo: Yo nací en Francia [...] y hablo muy bien francés. Aquí, con poco contexto, es saber que la palabra a predecir es un idioma. Pero necesitamos el contexto entero para poder saber qué idioma es.*

Tal y como nos lo explica Christopher Olah en su maravilloso art?culo

Cuando tenemos un **hueco muy grande entre la predicción y el contexto**, las RNN no son capaces de transmitir información tan antigua.

Aquí es donde entran en juego las **LSTM** (Long-Short Term Memory), unas RNN capaces de mantener la información durante largas etapas.

Las RNN son una cadena de módulos repetidos, en los que cada módulo puede ser tan simple como una tanh de función de activación.

Las LSTM tienen esta estructura de **cadena de módulos**, pero esta vez el módulo es un poco más complejo. **Cada módulo**, en vez de tener una simple función de activación, existen **4 neuronas que interactúan entre ellas**.

GRU vs LSTM

Ambas arquitecturas de red se utilizan para resolver los problemas de las RNN de gradiente y de memoria a corto plazo.

En teoría, las celdas **LSTM** tienen una puerta adicional y, por lo tanto, son **más complejas y tardan más en entrenarse**. Esta complejidad añadida debería facilitar recordar secuencias más largas. Sin embargo, no hay evidencia empírica clara de que una de un tipo de redes supere a al otro en todos los casos. Andrew Ng recomienda comenzar con GRU, ya que son más simples y escalables que las celdas LSTM.

Para tener una explicación ilustrativa de cómo funcionan ambas arquitecturas, os dejo el siguiente video.

Lo último: Transformers

Debido a que estas arquitecturas son bastante complejas, en este post me limitar? a mencionarlos simplemente.

> *Transformer: la tecnología que domina el mundo*

Un Transformer se trata del **estado del arte de los modelos de secuencias**. Vaswani *et al.* describen a la perfección qué son los transformers y sus aplicaciones en su paper «**Attention is all you need**«. Los transformers se tratan modelos de Secuencia a secuencia (Sequence to Sequence, o comúnmente Seq2Seq). Son modelos que teniendo una secuencia como input devuelven otra secuencia (la arquitectura many-to-many que hemos descrito antes).

Un **Transformer es una arquitectura que convierte una secuencia en otra con la ayuda de dos partes (codificador y decodificador)**, pero es diferente de los modelos secuenciales que hemos hablado

antes ya que no utiliza RNN. En la imagen siguiente se pueden ver el codificador (izquierda) y decodificador (derecha) básicos de un Transformer.

Estos modelos han avanzado mucho en los últimos años gracias a las grandes tecnológicas. Uno de los Transformers que más impacto han tenido en el segmento se trata de BERT, el algoritmo de Google que presentaron en este paper: BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding.

Sin embargo, el modelo más avanzado a día de hoy se trata de GPT-3, introducido por OpenAI, una mejora del famoso GPT-2.

Conclusión

Los modelos de secuencia se trata de **soluciones que atacan los problemas en los que el contexto** (información previa y/o posterior) **es muy importante a la hora de predecir**. La **estructura básica** de estos modelos son las **RNN**. Las cuales presentan **problemas** de gradiente y de memoria que **se resolvieron con** la introducción de **GRU y LSTM**.

El estado del arte de los modelos secuenciales son los Transformers, como BERT, el algoritmo de Google, o los GPT de OpenAI.

Si te ha gustado este post no dudes en comentarnos las dudas o lo próximo que quieras leer. Te dejo otros posts para que les eches un vistazo!

Francisco Sanz

Hola! Soy Francisco Sanz Estévez. Soy ingeniero de telecomunicaciones de ICAI, con máster por la misma universidad. Soy consultor de Data Science para proyectos tanto en España como en Estados Unidos. Me gusta mucho trastear con los datos que tengo, indagar las posibles relaciones entre las variables y ser capaz de representarlas visualmente. Como bien dijo Oppenheimer, la mejor manera de aprender es enseñando algo, así que aquí estoy para tratar de enseñaros poco a poco lo que sí y también para aprender de vosotros.

ANTERIOR
Docker para principiantes! Apre...

SIGUIENTE
Transformer: la tecnología que ...

Deja una respuesta

Comentario *

Nombre * **Correo electrónico ***

Web

Guarda mi nombre, correo electrónico y web en este navegador para la próxima vez que comente.

Publicar el comentario

Matplotlib para visualizar datos

Cloud Computing – Computación en la Nube

Manejo de excepciones en Python