Inteligencia Artificial

CC-421

César Lara Avila

Universidad Nacional de Ingeniería

(actualización: 2021-02-06)

Bienvenidos

Redes neuronales recurrentes

- Introducción
- Bloques básicos de construcción de RNN
- Propiedades de la RNN
- Desaparición del problema del gradiente y regularización

Introducción

La redes neuronales recurrentes (RNN) extienden el aprendizaje profundo a las secuencias.

Las RNN acumulan información de cada paso de tiempo en un **estado oculto**. Esto permite que la información secuencial sea **resumida** y se puedan hacer predicciones basadas en el historial completo de la secuencia.

RNN es un tipo de red neuronal, que puede procesar datos secuenciales con longitud variable. Los ejemplos de dichos datos incluyen las palabras de una oración o el precio de una acción en varios momentos del tiempo. Al utilizar la palabra secuencial, implicamos que los elementos de la secuencia están relacionados entre sí y que su orden es importante.

La principal característica de una RNN con respecto a otras redes neuronales artificiales es el ciclo de conexiones que cada unidad tiene (poseen conexiones internas que apuntan a sí mismas). Esto proporciona una funcionalidad similar a la **memoria** que hace de esta red neuronal un modelo para el procesamiento del lenguaje.

A diferencia de un modelo MLP que mapea entradas en salidas directamente, las RNN, pueden hacer uso de todas las entradas previas para cada salida, proporcionado un estado interno, además de los parámetros de la red.

En este tipo de red el aprendizaje está modelado por neuronas con conexiones recurrentes las cuales procesan cada nuevo patrón para obtener una salida que permite modificar la entrada de cada neurona cambiando el estado de toda la red.

Esta naturaleza le permite a la red comprimir toda la memoria en un espacio de baja dimensión lo que conlleva a desarrollar una característica importante de las redes recurrentes: formar **memoria a corto plazo**.

Bloques básicos de construcción de RNN

Una RNN es una red neuronal de alimentación directa estándar aplicada a entradas vectoriales en una secuencia.

Para incorporar el contexto secuencial en la predicción del siguiente paso de tiempo, se debe conservar una **memoria** de los pasos de tiempo anteriores en la secuencia.

Recurrencia y memoria

Definamos una secuencia de entrada de longitud T como X, donde $X = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_T\}$, tal que $\mathbf{x}_t \in \mathbf{R}^N$ es una entrada vectorial en el tiempo t.

Luego definimos nuestra memoria o historia hasta e incluyendo el tiempo t como $\mathbf{h}_t$. Por lo tanto, podemos definir nuestra salida de $\mathbf{o}_t$ como:

$$\mathbf{o}_t = f(\mathbf{x}_t, \mathbf{h}_{t-1})$$

donde la función f mapea memoria y entrada a una salida. La memoria del paso de tiempo anterior es $\mathbf{h}_{t-1}$ y la entrada es $\mathbf{x}_t$. Para el caso inicial $\mathbf{x}_1$, $\mathbf{h}_0$ es el vector cero $\mathbf{0}$.

De manera abstracta, se considera que la salida $\mathbf{o}_t$ ha resumido la información de la entrada actual $\mathbf{x}_t$ y el historial anterior de $\mathbf{h}_{t-1}$. Por lo tanto, $\mathbf{o}_t$ puede considerarse el vector histórico para toda la secuencia hasta el tiempo t incluido. Esto produce la ecuación:

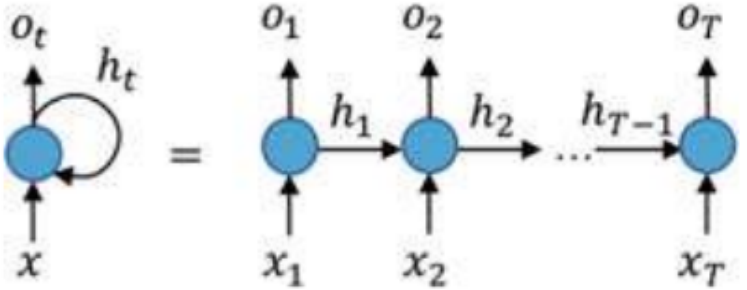
$$\mathbf{h}_t = \mathbf{o}_t = f(\mathbf{x}_t, \mathbf{h}_{t-1})$$

Más formalmente, podemos extender este concepto a las redes neuronales redefiniendo la función de transformación f de la siguiente manera:

$$\mathbf{h}_t = f(\mathbf{U}\mathbf{x}_t + \mathbf{W}\mathbf{h}_{t-1})$$

donde $\mathbf{W}$ y $\mathbf{U}$ son matrices de peso $\mathbf{W}, \mathbf{U} \in \mathbf{R}^{(N \times N)}$ y f es una función de activación, como ReLU.

Propiedades de las RNN



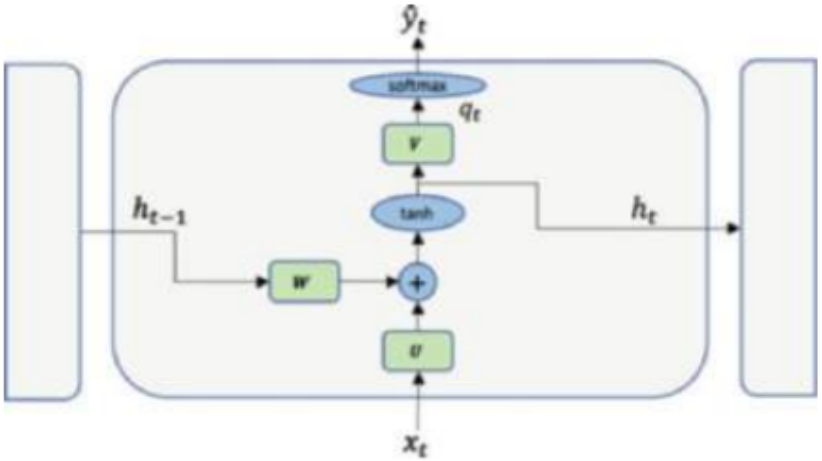
Propagación hacia adelante y hacia atrás en RNN

Los RNN se entrenan a través del BPTT y el descenso de gradiente de forma similar a las redes neuronales usuales. Las ecuaciones de propagación hacía adelante para h_t y la predicción de salida $\hat{y}_t$ son:

$$\mathbf{h}_t = \tanh(\mathbf{U}\mathbf{x}_t + \mathbf{W}\mathbf{h}_{t-1})$$
$$\hat{\mathbf{y}}_t = \text{softmax}(\mathbf{V}\mathbf{h}_t)$$

donde los parámetros que se pueden aprender son $\mathbf{U}$, $\mathbf{W}$ y $\mathbf{V}$. $\mathbf{U}$ incorpora la información de $\mathbf{x}_t$, $\mathbf{W}$ incorpora el estado recurrente y $\mathbf{V}$ aprende una transformación al tamaño y la clasificación de salida.

Propagación hacía adelante de un RNN



Calculamos el error, usando la función de pérdida de entropía cruzada en cada paso de tiempo t, donde $\mathbf{y}_t$ es el objetivo.

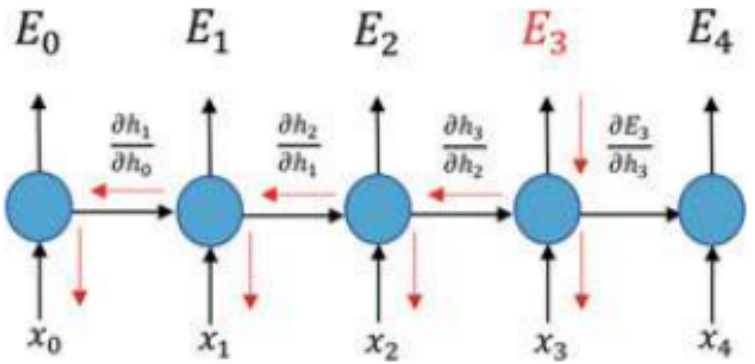
$$E_t = -\mathbf{y}_t \log \hat{\mathbf{y}}_t$$

Esto nos da la pérdida general con lo siguiente ecuación:

$$L(\mathbf{y}, \hat{\mathbf{y}}) = -\frac{1}{N} \sum_t \mathbf{y}_t \log \hat{\mathbf{y}}_t.$$

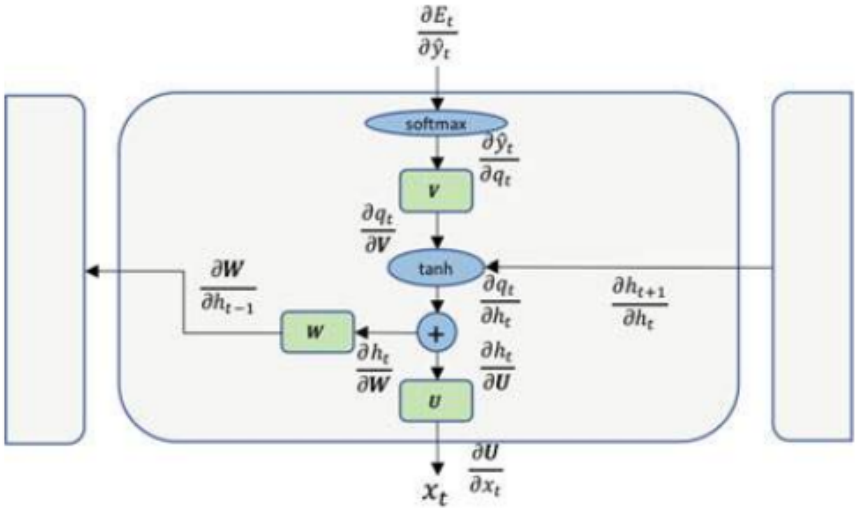
Los gradientes se calculan evaluando cada ruta que contribuye a la predicción $\hat{\mathbf{y}}_t$.

Este proceso se denomina **retropropagación a través del tiempo (BPTT)**.



El error E_3 se compone de la entrada de cada paso de tiempo anterior y las entradas de esos pasos de tiempo. Se excluye la propagación hacia atrás con respecto a E_1, E_2 y E_4.

Los parámetros de la RNN son $\mathbf{U}$, $\mathbf{V}$ y $\mathbf{W}$, por lo que debemos calcular el gradiente de la función de pérdida con respecto a estas matrices.



Retropropagación a través de un solo paso de tiempo de una RNN simple.

Pesos de salida (V)

la matriz de peso V controla la dimensionalidad de salida de $\hat{y}$ y no contribuye a la conexión recurrente. Por lo tanto, calcular el gradiente es lo mismo que una capa lineal. Por conveniencia,

$$\mathbf{q}_t = \mathbf{V}\mathbf{h}_t$$

Entonces,

$$\frac{E_t}{\partial V_{i,j}} = \frac{\partial E_t}{\partial \hat{y}_{t_k}} \frac{\partial \hat{y}_{t_k}}{\partial q_{t_l}} \frac{\partial q_{t_l}}{\partial V_{i,j}}.$$

Desde la definición de E_t , tenemos:

$$\frac{\partial E_t}{\partial \hat{y}_{t_k}} = -\frac{y_{t_k}}{\hat{y}_{t_k}}.$$

La retropropagación a través de la función softmax se puede calcular como:

$$\frac{\partial \hat{y}_{t_k}}{\partial q_{t_l}} = \begin{cases} -\hat{y}_{t_k}\hat{y}_{t_l}, & k \neq l \\ \hat{y}_{t_k}\left(1-\hat{y}_{t_k}\right), & k = l \end{cases}.$$

Si combinamos las ecuaciones anteriores, obtenemos la suma para todos los valores k para obtener $\frac{\partial E_t}{\partial q_{t_l}}$:

$$\begin{aligned} -\frac{y_{t_l}}{\hat{y}_{t_l}}\hat{y}_{t_l}(1-\hat{y}_{t_l}) + \sum_{k\neq l}\left(-\frac{y_{t_k}}{\hat{y}_{t_k}}\right)(-\hat{y}_{t_k}\hat{y}_{t_l}) &= -y_{t_l} + y_{t_l}\hat{y}_{t_l} + \sum_{k\neq l} y_{t_k}\hat{y}_{t_l} \\ &= -y_{t_l} + \hat{y}_{t_l}\sum_{k} y_{t_k}. \end{aligned}$$

Como todos los y_t son vectores one-hot, todos los valores en el vector son cero, excepto uno que indica la clase. Por lo tanto, la suma es 1, entonces:

$$\frac{E_t}{\partial q_{t_l}} = \hat{y}_{t_l} - y_{t_l}$$

Por último $\mathbf{q}_t = \mathbf{V}\mathbf{h}_t$, así $q_{t_l} = V_{l,m} h_{t_m}$.

Por tanto:

$$\frac{\partial q_{t_l}}{\partial V_{i,j}} = \delta_{il} h_{tj}.$$

Combinamos estas ecuaciones para obtener la ecuación que es un producto externo:

$$\frac{E_t}{\partial V_{i,j}} = \hat{y}_{t_i} - y_{t_i} h_{t_j},$$

Por lo tanto:

$$\frac{\partial E_t}{\partial \mathbf{V}} = (\hat{\mathbf{y}}_t - \mathbf{y}_t) \otimes h_t$$

donde $\otimes$ representa un producto externo.

Pesos recurrentes (W)

El parámetro $\mathbf{W}$ aparece en el argumento $\mathbf{h}_t$, por lo que se debe verificar el gradiente en $\mathbf{h}_t$ y $\hat{\mathbf{y}}_t$. También $\hat{\mathbf{y}}_t$ depende de $\mathbf{W}$ tanto directa como indirectamente (a través de $\mathbf{h}_{t-1}$).

Sea $\mathbf{z}_t = \mathbf{U}\mathbf{x}_t + \mathbf{W}\mathbf{h}_{t-1}$. Entonces $\mathbf{h}_t = \tanh(\mathbf{z}_t)$. Por la regla de la cadena se tiene:

$$\frac{\partial E_t}{\partial W_{i,j}} = \frac{\partial E_t}{\partial \hat{y}_{t_k}} \frac{\partial \hat{y}_{t_k}}{\partial q_{t_l}} \frac{\partial q_{t_l}}{\partial h_{t_m}} \frac{\partial h_{t_m}}{\partial W_{i,j}}$$

Calculemos los dos términos finales (los dos primeros términos ya fueron calculados)

$$\begin{aligned} \frac{\partial q_{t_l}}{\partial h_{t_m}} &= \frac{\partial}{\partial h_{t_m}} (V_{l,b} h_{t_b}) \\ &= V_{l,b} \delta_{b,m} \\ &= V_{l,m} \end{aligned}$$

El término final necesita una dependencia implícita de h_t en $W_{i,j}$, a h_{t-1}, así como una dependencia directa. Por lo tanto, tenemos:

$$\frac{\partial h_{t_m}}{\partial W_{i,j}} \rightarrow \frac{\partial h_{t_m}}{\partial W_{i,j}} + \frac{\partial h_{t_m}}{\partial h_{t-1_n}} \frac{\partial h_{t-1_n}}{\partial W_{i,j}}.$$

Pero podemos aplicar esto nuevamente para obtener:

$$\frac{\partial h_{t_m}}{\partial W_{i,j}} \rightarrow \frac{\partial h_{t_m}}{\partial W_{i,j}} + \frac{\partial h_{t_m}}{\partial h_{t-1_n}} \frac{\partial h_{t-1_n}}{\partial W_{i,j}} + \frac{\partial h_{t_m}}{\partial h_{t-1_n}} \frac{\partial h_{t-1_n}}{\partial h_{t-2_p}} \frac{\partial h_{t-2_p}}{\partial W_{i,j}}.$$

Este proceso continúa hasta llegar a $h_{(-1)}$, que se inicializó en un vector de ceros $(\mathbf{0})$. El último término en la ecuación anterior se contrae a $\frac{\partial h_{t_m}}{\partial h_{t-2_n}} \frac{\partial h_{t-2_n}}{\partial W_{i,j}}$ y podemos convertir el primer término en $\frac{\partial h_{t_m}}{\partial h_{t_n}} \frac{\partial h_{t_n}}{\partial W_{i,j}}$.

Con todo estos resultados, podemos expresar la ecuación en una forma más compacta:

$$\frac{\partial h_{t_m}}{\partial W_{i,j}} = \frac{\partial h_{t_m}}{\partial h_{r_n}} \frac{\partial h_{r_n}}{\partial W_{i,j}}$$

donde sumamos todos los valores de r menores que t además del índice dummy estándar n, es decir:

$$\frac{\partial h_{t_m}}{\partial W_{i,j}} = \sum_{r=0}^{t} \frac{\partial h_{t_m}}{\partial h_{r_n}} \frac{\partial h_{r_n}}{\partial W_{i,j}}.$$

Este término es responsable del problema de la desaparición y explosión del gradiente.

La multiplicación del término $\frac{\partial h_{tm}}{\partial h_{rn}}$ por el término $\frac{\partial h_{rn}}{\partial W_{i,j}}$ significa que el producto será más pequeño si ambos términos son menores que 1 o mayores si los términos son mayores que 1.

Combinando todos estos rendimientos obtenemos la siguiente ecuación:

$$\frac{\partial E_t}{\partial W_{i,j}} = (\hat{y}_{t_l} - y_{t_l}) V_{l,m} \sum_{r=0}^{t} \frac{\partial h_{t_m}}{\partial h_{r_n}} \frac{\partial h_{r_n}}{\partial W_{i,j}}.$$

Pesos de entrada (U)

Tomar el gradiente de $\mathbf{U}$ es similar a hacerlo para $\mathbf{W}$ ya que ambos requieren tomar derivadas secuenciales del vector $\mathbf{h}_t$. Tenemos entonces:

$$\frac{\partial E_t}{\partial U_{i,j}} = \frac{\partial E_t}{\partial \hat{y}_{t_k}} \frac{\partial \hat{y}_{t_k}}{\partial q_{t_l}} \frac{\partial q_{t_l}}{\partial h_{t_m}} \frac{h_{t_m}}{\partial U_{i,j}}$$

Después de los cálculos anteriores solo necesitamos calcular el último término de la ecuación. Siguiendo el mismo procedimiento para W, encontramos que:

$$\frac{h_{t_m}}{\partial U_{i,j}} = \sum_{r=0}^{t} \frac{\partial h_{t_m}}{\partial h_{r_n}} \frac{\partial h_{r_n}}{\partial U_{i,j}},$$

y así tenemos:

$$\frac{\partial E_t}{\partial U_{i,j}} = (\hat{y}_{t_l} - y_{t_l}) V_{l,m} \sum_{r=0}^{t} \frac{\partial h_{t_m}}{\partial h_{r_n}} \frac{\partial h_{r_n}}{\partial U_{i,j}}.$$

La diferencia entre $\mathbf{U}$ y $\mathbf{W}$ aparece en la implementación real ya que los valores de $\frac{\partial h_{rn}}{\partial U_{i,j}}$ y $\frac{\partial h_{rn}}{\partial W_{i,j}}$ difieren.

Gradiente agregado

El error para todos los pasos de tiempo es la suma de E_t según la función de pérdida dada.

Por lo tanto, podemos sumar los gradientes para cada uno de los pesos en la red $\mathbf{U}$, $\mathbf{V}$ y $\mathbf{W}$ y actualizar luego con los gradientes acumulados.

Desaparición del problema del gradiente y regularización

En el BPTT, Los gradientes se multiplican por la contribución del peso al error en cada paso de tiempo.

El impacto de esta multiplicación en cada paso de tiempo reduce o aumenta drásticamente el gradiente propagado al paso de tiempo anterior, que a su vez se multiplicará nuevamente.

La multiplicación recurrente de este paso en el BPTT provoca un efecto exponencial para cualquier irregularidad.

- Si los pesos son pequeños, los gradientes se reducirán exponencialmente.
- Si los pesos son grandes, los gradientes crecerán exponencialmente.

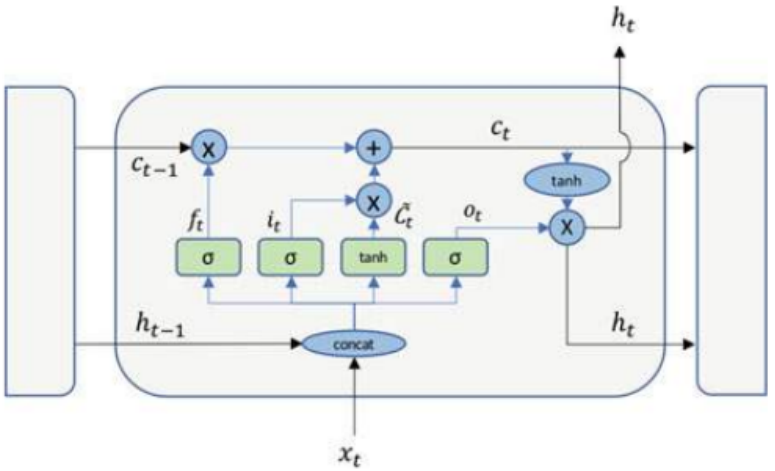
Hay muchos métodos utilizados para combatir el problema de la desaparición del gradiente, muchos de ellos se centran en una inicialización cuidadosa o en controlar el tamaño del gradiente que se propaga.

El método más utilizado para tratar este problema es la **adición de puertas a las RNN**.

LSTM

Este tipo de redes se desvían de las arquitecturas tradicionales, ya que presentan el concepto de **celda memoria** que les permite aprender a preservar o actualizar su estado en cada iteración de recurrencia.

Una red LSTM está compuesta por celdas de memoria y puertas, para calcular los estados internos.



Las 3 puertas LSTM controlan el comportamiento de las celdas de memoria: la puerta de entrada i, la puerta del olvido f y la puerta de salida o.

Estas puertas determinan con precisión que parte de un vector dado se debe tener en cuenta en un paso de tiempo de tiempo específico t.

Una celda de memoria contiene ponderaciones, que controlan cada puerta, el algoritmo BPTT optimiza esas ponderaciones utilizando el error de salida de la red resultante. Estos mecanismos de activación son en sí mismo capas de redes neuronales.

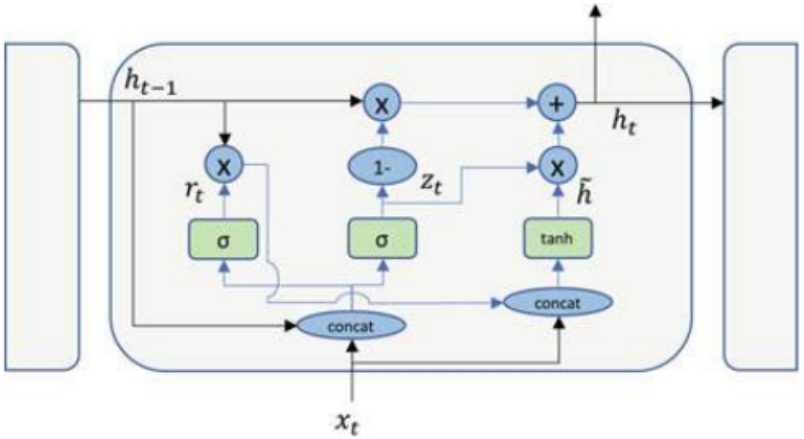
Esto permite que la red conozca las condiciones para olvidar, ignorar o guardar información en la celda de memoria.

Una celda LSTM se define formalmente como:

$$
\begin{aligned}
\mathbf{i}_t &= \sigma(\mathbf{W}_i \mathbf{x}_t + \mathbf{U}_i \mathbf{h}_{t-1} + \mathbf{b}_i) \\
\mathbf{f}_t &= \sigma(\mathbf{W}_f \mathbf{x}_t + \mathbf{U}_f \mathbf{h}_{t-1} + \mathbf{b}_f) \\
\mathbf{o}_t &= \sigma(\mathbf{W}_o \mathbf{x}_t + \mathbf{U}_o \mathbf{h}_{t-1} + \mathbf{b}_o) \\
\tilde{\mathbf{c}_t} &= \tanh(\mathbf{W}_c \mathbf{x}_t + \mathbf{U}_c \mathbf{h}_{t-1}) \\
\mathbf{c}_t &= \mathbf{f}_t \circ \mathbf{c}_{t-1} + \mathbf{i}_t \circ \tilde{\mathbf{c}_t} \\
\mathbf{h}_t &= \mathbf{o}_t \circ \tanh(\mathbf{c}_t)
\end{aligned}
$$

GRU

En 2014, Kyunghyun Cho presentó una alternativa a las redes LSTM, llamadas **unidad recurrente cerrada (GRU)** que tiene un similar o mejor desempeño que las LSTM, pero con menos paramétros y operaciones.



Una GRU combina las puertas en el LSTM para crear una regla de actualización más simple con una capa menos aprendida, lo que reduce la complejidad y aumenta la eficiencia.

La elección entre usar LSTM o GRU se decide en gran medida empíricamente.

Las ecuaciones para las reglas de actualización se muestran a continuación:

$$
\begin{aligned}
\mathbf{z}_t &= \sigma(\mathbf{W}_z \mathbf{x}_t + \mathbf{U}_z \mathbf{h}_{t-1}) \\
\mathbf{r}_t &= \sigma(\mathbf{W}_r \mathbf{x}_t + \mathbf{U}_r \mathbf{h}_{t-1}) \\
\tilde{\mathbf{h}_t} &= \tanh(\mathbf{W}_h \mathbf{x}_t + \mathbf{U}_h \mathbf{h}_{t-1} \circ \mathbf{r}_t) \\
\mathbf{h}_t &= (1 - \mathbf{z}_t) \circ \tilde{\mathbf{h}_t} + \mathbf{z}_t * \mathbf{h}_{t-1}
\end{aligned}
$$

En una GRU, el nuevo estado candidato, $\tilde{\mathbf{h}_t}$, se combina con el estado anterior, con $\mathbf{z}_t$ determinando cuánto de la historia se lleva adelante o cuánto reemplaza el nuevo candidato.

De manera similar a establecer el sesgo de la puerta de olvido de una LSTM para mejorar la memoria en las primeras etapas, los sesgos de puerta de reinicio de GRU se pueden establecer en -1.

RNN bidireccionales

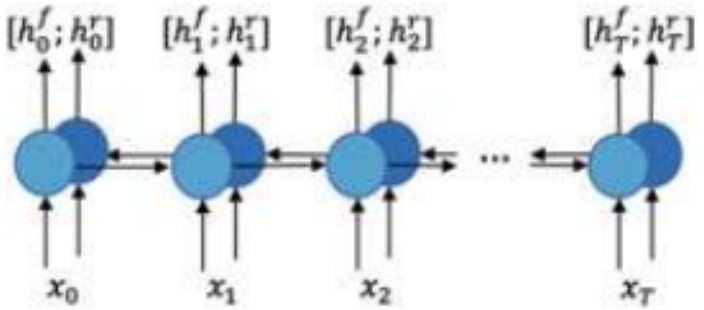
En muchas situaciones es deseable saber qué se encontrará en los pasos de tiempo futuros para informar la predicción en el tiempo t.

Las RNN bidireccionales permiten incorporar tanto el contexto hacia adelante como el contexto *hacia atrás* en una predicción. Esto se logra ejecutando dos RNN sobre una secuencia, uno en la dirección hacia adelante y otro en la dirección hacia atrás.

Para una secuencia de entrada $X = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_T\}$, el contexto hacía adelante RNN recibe las entradas en orden forward $t = \{1, 2, \ldots, T\}$ y el contexto hacia atrás RNN recibe las entradas en orden inverso $t = \{T, T-1, \ldots, 1\}$.

Estos dos RNN juntos constituyen una sola capa bidireccional.

La salida de los dos RNN, $\mathbf{h}^f$ y $\mathbf{h}^r$, a menudo se une para formar un solo vector de salida, ya sea sumando los dos vectores, concatenando, promediando u otro método.



Una limitación de los RNN bidireccionales es que la secuencia de entrada completa debe conocerse antes de la predicción, porque el RNN inverso requiere $\mathbf{x}_T$ para el primer cálculo.

Por lo tanto, las RNN bidireccionales no se pueden utilizar para aplicaciones en tiempo real. Sin embargo, dependiendo de los requisitos de la aplicación, tener un búfer fijo para la entrada puede aliviar esta restricción.

Recorte del gradiente

Por lo general, es una buena práctica rastrear la norma del gradiente para comprender sus características y luego reducir el gradiente cuando se excede el rango operativo normal

Las dos formas más comunes de recortar gradientes son:

Recorte de la norma L_2 con un umbral t.

$$\nabla_{\text{nuevo}} = \nabla_{\text{actual}} \circ \frac{t}{L_2(\nabla)}$$

Rango fijo

$$\nabla_{\text{nuevo}} = t_{\text{min}} \ \text{ si } \nabla < t_{min}$$

$$\nabla_{\text{nuevo}} = t_{\text{max}} \ \text{ si } \nabla > t_{max}$$

Con un máximo umbral t_{max} y un mínimo umbral t_{min}.

Longitud de secuencia BPTT

Una forma de arreglar o limitar la cantidad de cómputo en el proceso de entrenamiento es establecer una longitud máxima de secuencia para el procedimiento de entrenamiento .

Las formas comunes de establecer la longitud de la secuencia son:

- Completar los datos de entrenamiento con la mayor longitud.
- Truncar el número de pasos que se retropropagan durante el entrenamiento.

Establecer una longitud máxima de secuencia puede ser útil en una variedad de situaciones. En particular, cuando:

- El grafo computacional estático requiere una entrada de tamaño fijo
- El modelo tiene limitaciones de memoria
- Los gradientes son muy grandes al comienzo del entrenamiento.

Dropout recurrente

El dropout al ser una técnica de regularización común, es una opción intuitiva para aplicar también a los RNN, sin embargo, la forma original debe modificarse.

Si se aplica la forma original en cada paso, la combinación de máscaras puede provocar que se pase poca señal a secuencias más largas.

En cambio, podemos reutilizar la misma máscara en cada paso para evitar la pérdida de información entre los pasos de tiempo.

Fin!

Estas notas están basados en el texto de Deep Learning for NLP and Speech Recognition de Kamath, Liu y Whitaker.