Redes Neuronales Recurrentes

Análisis de series de tiempo

Verónica E. Arriola-Rios

Facultad de Ciencias, UNAM

14 de abril de 2023

Intro

Temas

1. Intro
   - Antecedentes
   - Retropropagación en el tiempo
   - Características
   - Sistemas dinámicos

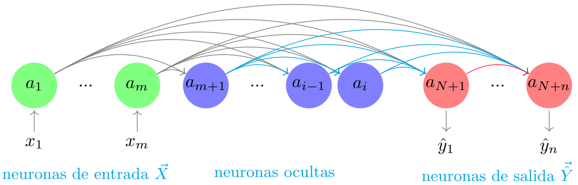
Redes neuronales ordenadas

- Son las redes con más conexiones que se podrían diseñar, pero aún utilizan alimentación hacia adelante simple. Werbos 1990

$$a_i = x_i \qquad 1 \leqslant i \leqslant m$$

$$a_i = \sigma(z_i) \qquad z_i = \sum_{j=1}^{i-1} w_{ij} a_i \qquad m < i \leqslant N + n, N \geqslant m$$

$$\hat{y}_i = a_{N+i} \qquad 1 \leqslant i \leqslant n$$



Ejemplo de funciones ordenadas

Función ordenada:

$$z_2 = 4z_1$$
$$z_3 = -3z_1 + 5z_2$$
$$z_4 = 4z_3 - 2z_2 + 8z_1$$

Regla de la cadena para derivadas ordenadas

Esta regla de la cadena fue propuesta por Werbos 1990 y sólo es válida para *sistemas ordenados* donde los valores pueden ir siendo calculados uno por uno (si es necesario) en el orden $x_1, x_2, ..., x_n, E$ y la función es una suma de términos:

$$E = c_1 x_1 + c_2 x_2 + ... + c_i x_i$$

con las c_i constantes.

Sean las ∂^+ derivadas ordenadas y ∂ derivadas parciales ordinarias:

$$\frac{\partial^+ E}{\partial x_i} = \frac{\partial E}{\partial x_i} + \sum_{j>i} \frac{\partial^+ E}{\partial x_j} * \frac{\partial x_j}{\partial x_i}$$

Simplificando la notacion usando F_x para las derivadas ordenadas:

$$F_{x_i} = \frac{\partial E}{\partial x_i} + \sum_{j>i} F_{x_j} * \frac{\partial x_j}{\partial x_i}$$

Ejemplo de derivadas ordenadas

Función ordenada:

$$z_2 = 4z_1$$
$$z_3 = -3z_1 + 5z_2$$
$$z_4 = 4z_3 - 2z_2 + 8z_1$$

Derivada ordenada:

$$\begin{aligned}\frac{\partial^+ z_3}{\partial z_1} &= \frac{\partial z_3}{\partial z_1} + \frac{\partial^+ z_3}{\partial z_2}\frac{\partial z_2}{\partial z_1} \\ &= -3 + 5(4) = 17\end{aligned}$$

Temas

1. Intro
   - Antecedentes
   - Retropropagación en el tiempo
   - Características
   - Sistemas dinámicos

Red Neuronal Recurrente

El valor de activación de una neurona puede depender de los valores de activación de las otras neuronas en tiempos pasados.

$$z_i(t) = \sum_{j=1}^{i-1} W_{ij} a_j(t) + \sum_{j=1}^{N+n} W'_{ij} a_j(t-1) + \sum_{j=1}^{N+n} W''_{ij} a_j(t-2) \tag{1}$$

Todas las neuronas activas (excepto las de entrada) pueden tener como entradas a las salidas de cualquier otra neurona siempre y cuando haya un *retrazo temporal* en la conexión. Aquí W' y W'' son los pesos hacia esas conexiones retrazadas por uno y dos periodos, respectivamente.

Retropropagación en el tiempo

Sean $\hat{Y}$ los valores a la salida de la red:

$$F_{a_i}(t) = F_{\hat{Y}_{i-N}}(t) + \sum_{j=i+1}^{N+n} W_{ij} F_{net_j}(t) + \tag{2}$$

$$\sum_{j=m+1}^{N+n} W'_{ij} * F_{net_j}(t+1) + \sum_{j=m+1}^{N+n} W''_{ij} * F_{net_j}(t+2)$$

$$F_{W'_{ij}} = \sum_{t=1}^{T} F_{net_i}(t+1) * a_j(t) \tag{3}$$

$$F_{W''_{ij}} = \sum_{t=1}^{T} F_{net_i}(t+2) * a_j(t) \tag{4}$$

Temas

1. Intro
   - Antecedentes
   - Retropropagación en el tiempo
   - Características
   - Sistemas dinámicos

Redes Neuronales Recurrentes

- Las *redes neuronales recurrentes* se especializan en trabajar con secuencias de valores.

$$\{\vec{x}^{(1)}, \ldots, \vec{x}^{(\tau)}\}$$

  donde τ es el máximo número de pasos en el tiempo.

- Así como las redes convolucionales reducen el número de parámetros que se debe ajustar para analizar imágenes, las redes recurrentes reducen el número de parámetros requeridos para modelar secuencias temporales como señales de audio o cadenas de texto.

Temas

1. Intro
   - Antecedentes
   - Retropropagación en el tiempo
   - Características
   - Sistemas dinámicos

Sistema dinámico y su grafo

Definición recurrente:

$$\vec{s}^{(t)} = f(\vec{s}^{(t-1)}; \vec{\theta}) \tag{5}$$

Si $\tau = 3$:

$$\vec{s}^{(3)} = f(\vec{s}^{(2)}; \vec{\theta}) \tag{6}$$

$$= f(f(\vec{s}^{(1)}; \vec{\theta}); \vec{\theta}) \tag{7}$$

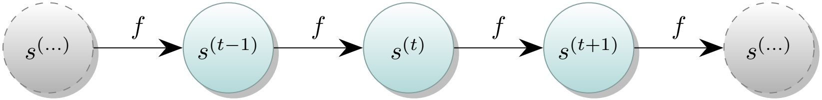


Figura: Despliegue del grafo computacional.

Sistema dinámico dirigido por una señal externa

$$\vec{s}^{(t)} = f(\vec{s}^{(t-1)}, \vec{x}^{(t)}; \vec{\theta}) \tag{8}$$

Para redes neuronales el estado corresponde a las capas ocultas:

$$\vec{h}^{(t)} = f(\vec{h}^{(t-1)}, \vec{x}^{(t)}; \vec{\theta}) \tag{9}$$

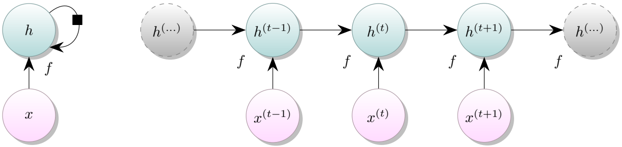


Figura: Red recurrente sin salidas. $\vec{x}$ son las entradas, $\vec{h}$ los estados. *Izquierda* Diagrama del circuito, el cuadro negro marca un retraso de un tiempo. *Derecha* Grafo desplegado.

- El tamaño del grafo desplegado depende de la longitud de la secuencia.
- Para el tiempo t el grafo desplegado se puede representar con la función $g^{(t)}$

$$\begin{aligned}\vec{h}^{(t)} &= g^{(t)}(\vec{x}^{(t)}, \vec{x}^{(t-1)}, \vec{x}^{(t-2)}, ..., \vec{x}^{(2)}, \vec{x}^{(1)}) \qquad (10)\\ &= f(\vec{h}^{(t-1)}, \vec{x}^{(t)}; \vec{\theta}) \qquad (11)\end{aligned}$$

Ventajas del proceso de despligue

1. El tamaño de la entrada para la red no depende de la longitud de la secuencia.
2. La *misma* función de transición f se usa con los mismos parámetros en cada paso temporal. *En analogía a la convolución usando los mismos pasos sobre diferentes regiones de una imagen.*

Arquitecturas comunes para clasificación

Red 1: una salida en cada t, recurrencia entre ocultas

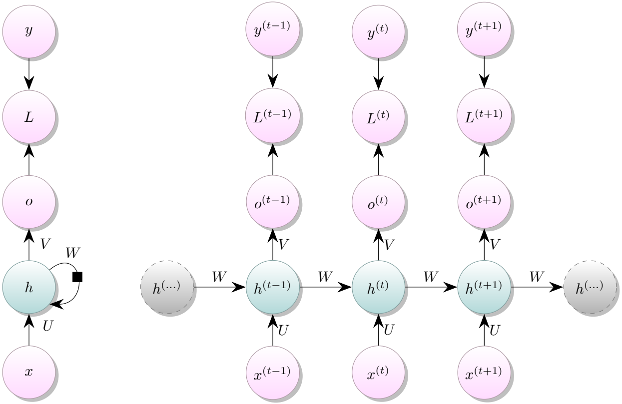


Figura: L es la función de error o pérdida, $\vec{y}$ los resultados esperados.

Ejemplo: *Entra un cuadro de señal acústica y sale la nota que está siendo reproducida.*

Universalidad

- Esta red es *Universal* en el sentido de que puede calcular cualquier función que pueda calcular una máquina de Turing.
- *Propagación hacia adelante:*
  1. Require especificar el estado inicial $\vec{h}^{(0)}$.
  2. Para $t \in [1, \tau]$.

$$\vec{a}^{(t)} = \vec{b} + \mathbf{W}\vec{h}^{(t-1)} + \mathbf{U}\vec{x}^{(t)}, \tag{12}$$

$$\vec{h}^{(t)} = \tanh(\vec{a}^{(t)}), \tag{13}$$

$$\vec{o}^{(t)} = \vec{c} + \mathbf{V}\vec{h}^{(t)}, \tag{14}$$

$$\vec{\hat{y}}^{(t)} = \mathrm{softmax}(o^{(t)}) \tag{15}$$

Función de error

- Se utiliza un planteamiento probabilista: ¿cuál es la probabilidad de obtener cada señal de salida $y^{(t)}$ dadas las entradas $x^{(i)}$ hasta el momento en que fue calculada?

$$
\begin{aligned}
L\left(\{\vec{x}^{(1)}, ..., \vec{x}^{(\tau)}\},\ \{\vec{y}^{(1)}, ..., \vec{y}^{(\tau)}\}\right) &= \sum_t L^{(t)} \\
&= -\sum_t \log \underbrace{p_{modelo}\left(y^{(t)} | \{\vec{x}^{(1)}, ..., \vec{x}^{(t)}\}\right)}_{\hat{\vec{y}}^{(t)}[y]}
\end{aligned}
\tag{16}
$$

Esta función es la log-verosimilitud o, en inglés, *log-likelihood*.

Red 2: una salida en cada t, recurrencia de salida a ocultas

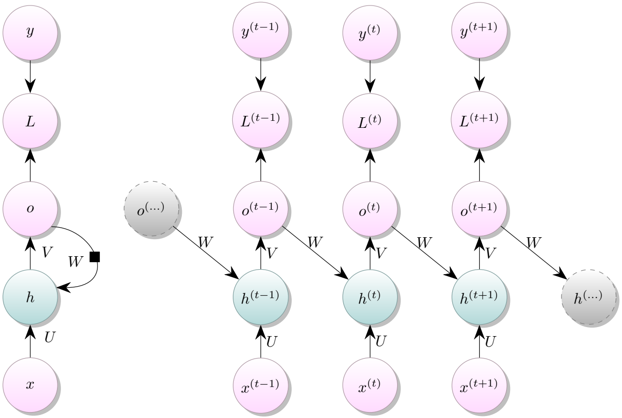


Figura: L es la función de error o pérdida, $\vec{y}$ los resultados esperados.

Forzamiento del profesor

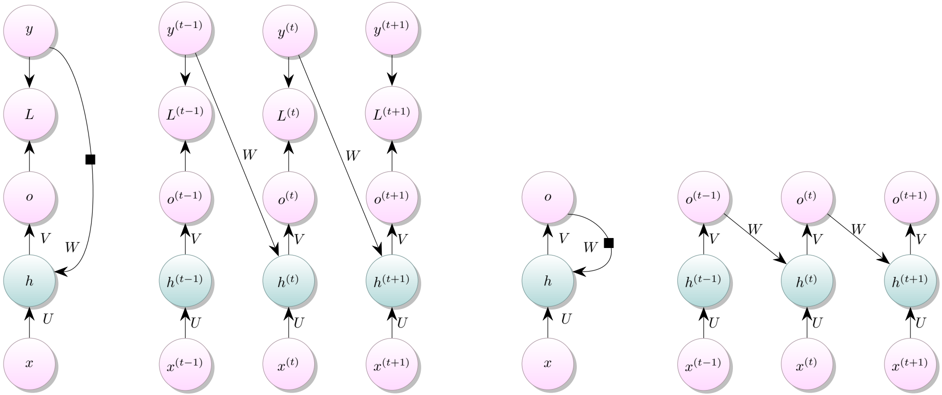


Figura: Conocer la respuesta correcta para el paso anterior permite paralelizar el entrenamiento. **Izquierda:** Durante el entrenamiento. **Derecha:** Durante la prueba.

Función de error

$$L\left(\{\vec{x}^{(1)}, \ldots, \vec{x}^{(t)}\}, \{\vec{y}^{(1)}, \ldots, \vec{y}^{(t)}\}\right) = \sum_t L^{(t)} \tag{17}$$

$$= -\sum_t \log p_{modelo}\left(\vec{y}^{(t)} | \{\vec{x}^{(1)}, \ldots, \vec{x}^{(t)}, \vec{y}^{(1)}, \ldots, \vec{y}^{(t-1)}\}\right) \tag{18}$$

Red 3: una salida al final de la secuencia, recurrencia entre ocultas

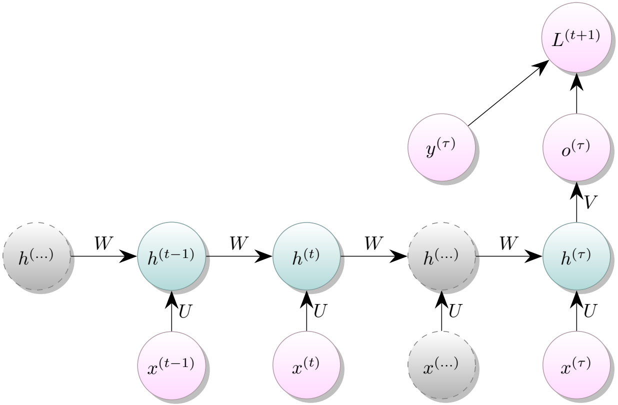


Figura: L es la función de error o pérdida, $\vec{y}$ los resultados esperados.

Ejemplo: *Entra un fragmento de una canción y sale el género al que pertenece.*

Predicción de secuecias

Modelando una secuencia sin entrada $\vec{x}$

$$P(\mathbb{Y}) = P(y^{(1)}, ..., y^{(\tau)}) = \prod_{t=1}^{\tau} P(y^{(t)}|y^{(t-1)}, y^{(t-2)}, ..., y^{(1)}) \tag{19}$$

En este modelo, el negativo de la log-verosimilitud es:

$$L = \sum_t L^{(t)}, \tag{20}$$

con

$$L^{(t)} = -\log P(y^{(t)} = y^{(t)}|y^{(t-1)}, y^{(t-2)}, ..., y^{(1)}) \tag{21}$$

Modelo gráfico completamente conectado

a)

b)

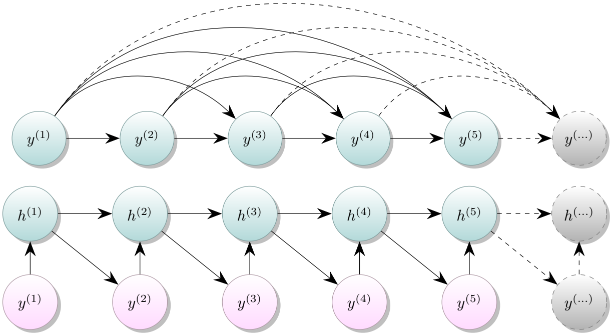


Figura: a) Modelo probabilista (con $\vec{\mathrm{h}}$ marginalizadas), parametrizar (9) sería ineficiente. b) Implementación usando el estado oculto de la RNN.

Fin de la secuencia

Formas de determinar el fin de la secuencia generada:

- Si la salida son símbolos de un vocabulario, agregar un símbolo especial para indicar *fin de secuencia*.
- Agregar una salida dada por la distribución de Bernoulli que represente la decisión de *continuar* o *parar*.
  - Unidad sigmoide
  - Función a optimizar: log-verosimilitud sobre la predicción correcta de si la secuencia termina o continua.
- Predecir τ directamente y generar ese número de pasos.
  - Agregar τ o $\tau - t$ como entrada recurrente, para que el generador tome en cuenta cuántos pasos le quedan.

$$P(\vec{x}^{(1)}, ..., \vec{x}^{(\tau)}) = P(\vec{x}^{(1)}, ..., \vec{x}^{(\tau)}|\tau)P(\tau) \tag{22}$$

Temas

3. Predicción de secuecias
   - Secuencia dependiente del contexto
   - Dependencia completa

Secuencia dependiente del contexto

Dependen de una sola señal de entrada $\vec{x}$ de tamaño fijo.
$\vec{x}$ es provista como:

1. Entrada extra en cada paso temporal, o
2. estado inicial $\vec{h}^{(0)}$, o
3. ambos

$\vec{x}$ como entrada extra

- Ej: Generar texto que describa una imagen. La imagen da el contexto.

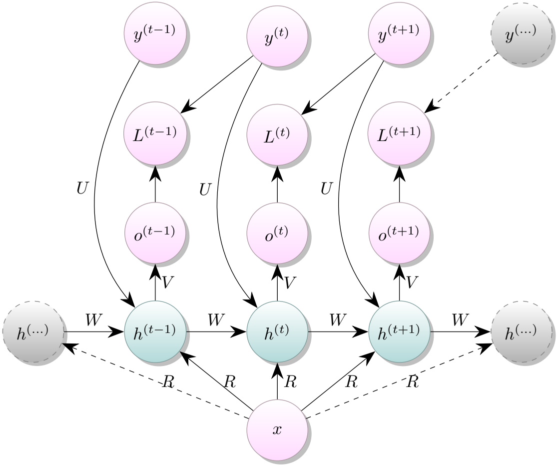


Figura: Red $\vec{x} \rightarrow P(\vec{Y})$. Cada elemento de la secuencia $\vec{y}^{(t)}$ sirve como entrada en t. Durante el entrenamiento es el objetivo del paso anterior. Se puede interpretar que los sesgos $\vec{x}^{T}\mathbf{R}$ son los que varían según la entrada.

Temas

3. Predicción de secuecias
   - Secuencia dependiente del contexto
   - Dependencia completa

RNN Condicional

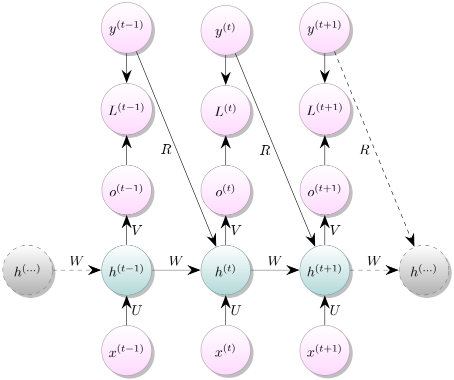
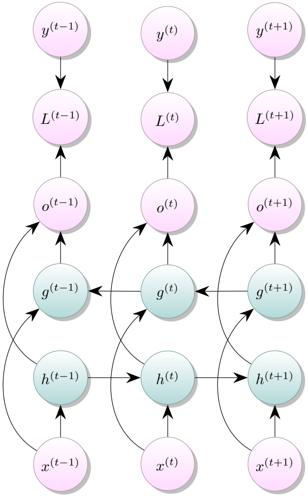


Figura: En $P(\vec{y}^{(1)}, ..., \vec{y}^{(\tau)} | \vec{x}^{(1)}, ..., \vec{x}^{(\tau)})$, las $\vec{y}^{(i)}$ ya no son independientes entre sí, pero ambas secuencias $\vec{x}$ y $\vec{y}$ deben ser de la misma longitud.

RNN Bidireccional

Figura: Compuesta por dos RNN.



RNR Codificadora-Decodificadora

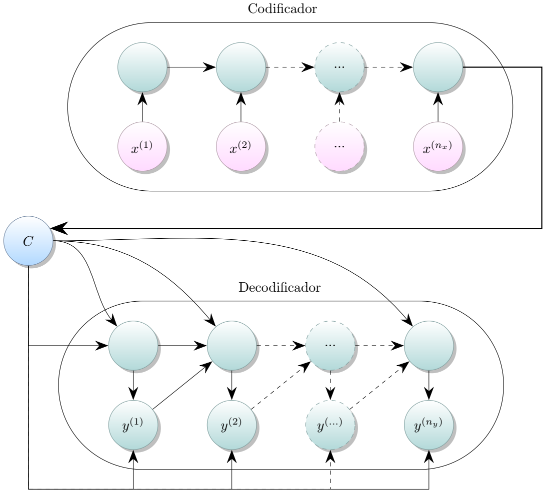


Figura: Aprende a generar $(\vec{y}^{(1)}, ..., \vec{y}^{(n_y)})$ a partir de $(\vec{x}^{(1)}, ..., \vec{x}^{(n_x)})$. Ej: para traductores de idiomas.

RNR Profundas

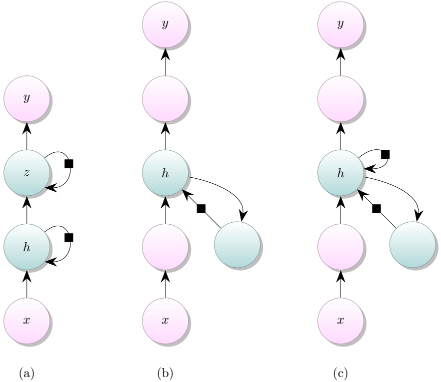


Figura: (a) Jerarquía en capas ocultas recurrentes. (b) Más capas antes/después de las recurrentes. (c) Conexiones tipo salto.

Redes Neuronales Recursivas

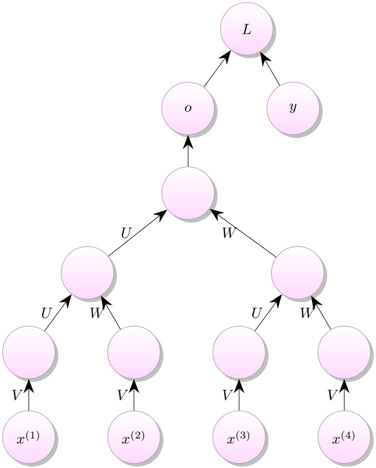


Figura: Su gráfica computacional es un árbol. $(\vec{x}^{(1)}), \ldots, \vec{x}^{(t)}) \to \vec{o}$. Sus parámetros son únicamente $\mathbf{U}, \mathbf{V}, \mathbf{W}$.

El problema del gradiente

Temas

4. El problema del gradiente
   - Explosión y desvanecimiento
   - Cómputo de yacimientos
   - Multiescalas temporales
   - Memoria de corto y largo plazo
   - Unidades Recurrentes con Compuertas

RNR Profundas

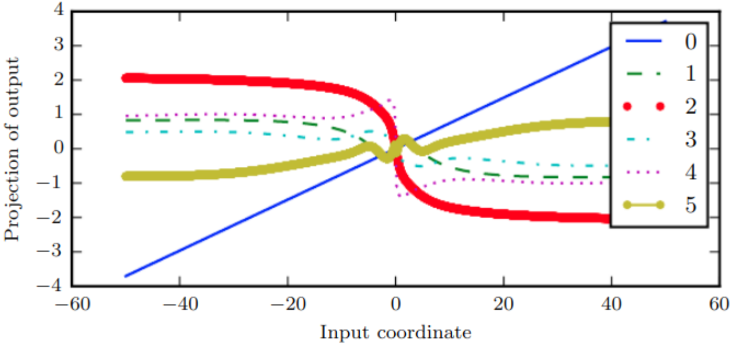


Figura: Corte de una salida Y en 100 dimensiones, cuando tanh es usada como función de activación de forma recursiva. Nótese la variación en la magnitud del gradiente conforme se incrementa la no-linealidad. **Fuente:** Goodfellow, Bengio y Courville 2016

Explición y desvanecimiento

- Ignorando de momento la función de activación, supongamos:

$$\vec{h}^{(t)} = \mathbf{Q}^\top \mathbf{\Lambda}^t \mathbf{Q} \vec{h}^{(0)} \tag{23}$$

  Nótese que la potencia de un valor propio tiende a cero o a explotar.
- Si la red no es recurrente, no se trata del mismo peso, por lo que se puede controlar la varianza v del producto de los pesos $\prod_t w^{(t)}$. Basta elegir $v = \sqrt[n]{v^*}$

Recortando el gradiente

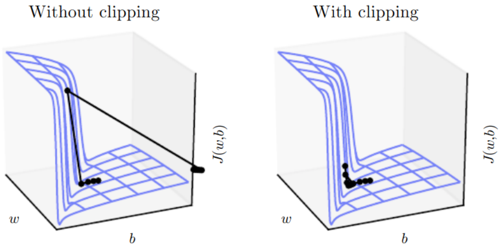


Figura: (a) Jerarquía en capas ocultas recurrentes. (b) Más capas antes/después de las recurrentes. (c) Conexiones tipo salto. **Fuente:** Goodfellow, Bengio y Courville 2016

$$\text{if } ||\vec{g}|| > v \qquad \vec{g} \leftarrow \frac{\vec{g}v}{||\vec{g}||} \tag{24}$$

Temas

4. El problema del gradiente
   - Explosión y desvanecimiento
   - Cómputo de yacimientos
   - Multiescalas temporales
   - Memoria de corto y largo plazo
   - Unidades Recurrentes con Compuertas

Cómputo de yacimientos

- *Red de estados con echo*. Su objetivo es fijar los pesos recurrentes de tal manera que las unidades ocultas recurrentes capturen correctamente la historia de las entradas pasadas y *sólo se entrenan los pesos hacia la salida*.
  - Para ello asignar pesos cuyo radio espectral sea 3, o 1.2 si serán entrenados.
- *Máquinas de estado líquido*. Idea similar, pero utilizan neuronas de impulsos (*spiking neurons*), con salidas binarias.

Temas

4. El problema del gradiente
   - Explosión y desvanecimiento
   - Cómputo de yacimientos
   - Multiescalas temporales
   - Memoria de corto y largo plazo
   - Unidades Recurrentes con Compuertas

Multiescalas temporales

Pretenden modelar secuencias distinguiendo entre detalles finos y detalles gruesos.

- *Conexiones con salto*. Agregar conexiones entre t y $t + d$.
- *Unidades con fugas*. Unidades con conexiones lineales a sí mismas, modelan un promedio acumulativo del tipo $\mu^{(t)} \leftarrow \alpha\mu^{(t-1)} + (1 - \alpha)\nu^{(t)}$ donde el valor de α indica qué tan rápido se olvida el pasado.
- *Remover conexiones* de un tiempo y sustituirlas por conexiones más al pasado.

Temas

4. El problema del gradiente
   - Explosión y desvanecimiento
   - Cómputo de yacimientos
   - Multiescalas temporales
   - Memoria de corto y largo plazo
   - Unidades Recurrentes con Compuertas

LSTM

Las memorias de corto y largo plazo, popularmente conocidas por su nombre en inglés *Long-Short Term Memory* (*LSTM*) son útiles para:

- Reconocimiento de escritura.
- Reconocimiento del habla.
- Generación de escritura.
- Traducción.
- Rotulado de imágenes.
- Análisis sintáctico.

LSTM

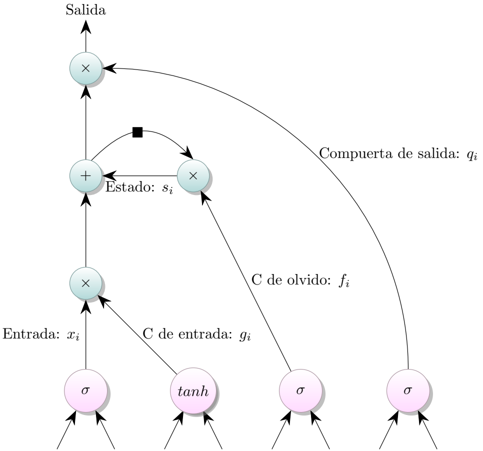


Figura: Célula/*cell* (en lugar de neurona oculta).
Entrada: $x_i^{(t)}$,
compuerta de entrada: $g_i^{(t)}$,
compuerta de olvido: $f_i^{(t)}$,
compuerta de salida: $q_i^{(t)}$,
salida: $h_i^{(t)}$.
Cada compuerta se puede alimentar con las entradas $x_i^{(t)}$ y las salidas del tiempo anterior $h_i^{(t-1)}$.

Valores de activación

Compuertas:

$$g_i^{(t)} = \sigma\left(b_i^g + \sum_j U_{i,j}^g x_j^{(t)} + \sum_j W_{i,j}^g h_j^{(t-1)}\right) \quad \text{entrada} \tag{25}$$

$$f_i^{(t)} = \sigma\left(b_i^f + \sum_j U_{i,j}^f x_j^{(t)} + \sum_j W_{i,j}^f h_j^{(t-1)}\right) \quad \text{olvido} \tag{26}$$

$$q_t^{(t)} = \sigma\left(b_i^o + \sum_j U_{i,j}^o x_j^{(t)} + \sum_j W_{i,j}^o h_j^{(t-1)}\right) \quad \text{salida} \tag{27}$$

Entrada $x_i^{(t)}$:

$$a_i^{(t)} = \sigma\left(b_i + \sum_j U_{i,j} x_j^{(t)} + \sum_j W_{i,j} h_j^{(t-1)}\right) \tag{28}$$

Estado $s_i^{(t)}$:

$$s_i^{(t)} = f_i^{(t)} s_i^{(t-1)} + g_i^{(t)} a_i^{(t)} \qquad \text{autoconexión} \tag{29}$$

Salida $h_i^{(t)}$:

$$h_i^{(t)} = \tanh(s_i^{(t)}) q_i^{(t)} \qquad \text{salida/valor oculto} \tag{30}$$

Versión de Hochreiter resumida

En azul se muestra cómo agregar los sesgos.

$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} \left[W \begin{pmatrix} \vec{h}^{(t-1)} \\ \vec{x}^{(t)} \end{pmatrix} + \vec{b} \right] \tag{31}$$

$$\vec{s}^{(t)} = \vec{f} \odot \vec{s}^{(t-1)} + \vec{i} \odot \vec{g} \tag{32}$$

$$\vec{h}^{(t)} = \vec{o} \odot \tanh(s^{(t)}) \tag{33}$$

- Donde $\odot$ representa la multiplicación término a término.
- i es *input* y es la entrada que se calcula,
- y las tres compuertas son f *forget*, o *output* y g *gate*.

Adaptada de Hochreiter and Schmidhuber, "Long short term memory". Neural Computation.

- Esta versión se encuentra comúnmente en internet.
- En este caso la compuerta de salida se llama o en lugar de q y la función de activación para g es tanh, en lugar de σ.
- Obsérvese que en este modelo tanto la entrada como las compuertas son alimentadas con los mismos valores: la concatenación de la entrada actual y la salida de la celda en el paso anterior.
- Sin embargo la matriz W tiene pesos distintos para cada uno de estos cálculos.

$$\mathbf{W} = \begin{pmatrix} \mathbf{W} & \mathbf{U} \\ \mathbf{W}^{\mathrm{f}} & \mathbf{U}^{\mathrm{f}} \\ \mathbf{W}^{\mathrm{o}} & \mathbf{U}^{\mathrm{o}} \\ \mathbf{W}^{\mathrm{g}} & \mathbf{U}^{\mathrm{g}} \end{pmatrix} \tag{34}$$

Temas

4. El problema del gradiente
   - Explosión y desvanecimiento
   - Cómputo de yacimientos
   - Multiescalas temporales
   - Memoria de corto y largo plazo
   - Unidades Recurrentes con Compuertas

GRU

Las *unidades recurrentes con compuertas*, llamas en inglés *Gated Recurrent Units* son células más sencillas que las LSTM, por lo que se entrenan más rápido, pero que persiguen el mismo fin.

No tienen un *estado s de la célula* y sólo tienen **dos** compuertas:

Actualización (*update*)

$$u_i^{(t)} = \sigma\left(b_i^u + \sum_j U_{i,j}^u x_j^{(t)} + \sum_j W_{i,j}^u h_j^{(t)}\right) \quad \text{actualización} \tag{35}$$

Reinicio (*reset*)

$$r_i^{(t)} = \sigma\left(b_i^r + \sum_j U_{i,j}^r x_j^{(t)} + \sum_j W_{i,j}^r h_j^{(t)}\right) \quad \text{reinicio} \tag{36}$$

Entradas para las compuertas

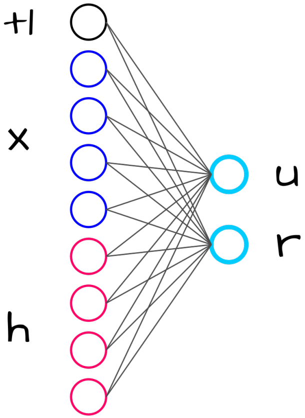


Figura: El valor de las compuertas se calcula como el de neuronas en una capa siguiente.

Para actualizar el valor almacenado:[1]

$$a_i^{(t)} = \tanh\left(b_i + \sum_j U_{i,j} x_j^{(t)} + \sum_j W_{i,j} r_j^{(t-1)} h_j^{(t-1)}\right) \tag{37}$$

Salida:

$$h_i^{(t)} = (1 - u_i^{(t-1)}) h_i^{(t-1)} + u_i^{(t-1)} a_i^{(t)} \tag{38}$$

Nota: Este conjunto de ecuaciones varía ligeramente entre fuentes, aunque la composicón escencial es la misma, por lo que no todos los diagramas que representan a las GRU son consistentes entre sí. Shi y col. 2021 Multiplican la proporción para actualizar u_i por el valor actualizado y la complementaria por el valor anterior, como se muestra en esta versión de la ecuación.

[1] Goodfellow, Bengio y Courville 2016 usan σ en lugar de tanh, pero esta última es más popular.

Esquema del cálculo de una GRU

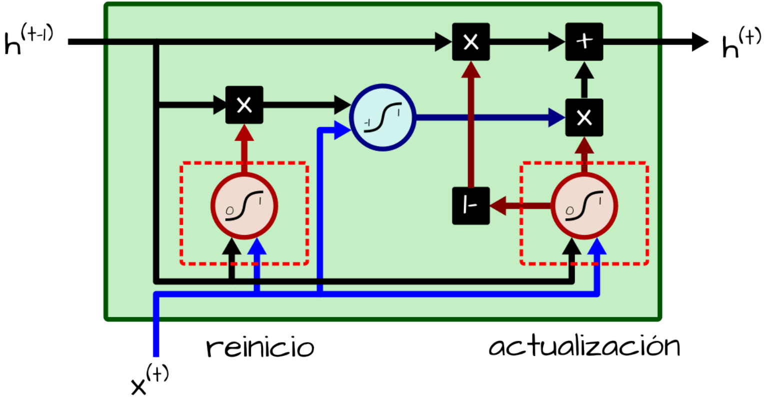


Figura: Diagrama de la GRU según las ecuaciones presentadas anteriormente. [2]

[2] Rediseñado con ayuda de Shi y col. 2021

GRU con matrices

Siguiendo el concepto de Shi y col. 2021 podemos resumir las ecuaciones anteriores en versión matricial:

$$
\begin{aligned}
u^{(t)} &= \sigma(W_u[x^{(t)}, h^{(t-1)}]) \\
r^{(t)} &= \sigma(W_r[x^{(t)}, h^{(t-1)}]) \\
a^{(t)} &= \varphi(W_a[x^{(t)}, r^{(t)} \odot h^{(t-1)}]) \\
h^{(t)} &= (I - u^{(t-1)}) \odot h^{(t-1)} + u^{(t-1)} \odot a^{(t)}
\end{aligned}
$$

con φ representando a la tangente hiperbólica y $\odot$ siendo la multiplicación componente a componente.
Ellos también agregan una salida:

$$
y^{(t)} = \sigma(W_o h^{(t)})
$$

Referencias I

- Goodfellow, Ian, Yoshua Bengio y Aaron Courville (2016). «Deep Learning». En: MIT Press. Cap. Sequence Modeling: Recurrentand Recursive Nets, págs. 367-415. URL: https://www.deeplearningbook.org/contents/rnn.html.
- Shi, Hanhong y col. (2021). «Short-Term Load Forecasting Based on Adabelief Optimized Temporal Convolutional Network and Gated Recurrent Unit Hybrid Neural Network». En: *IEEE Access* 9, págs. 66965-66981. DOI: 10.1109/ACCESS.2021.3076313.
- Werbos, Paul J. (oct. de 1990). «Backpropagation Through Time: What It Does and How to Do It». En: *Proceedings of the IEEE*. Vol. 78. 10.

Licencia