Aprendizaje profundo

Redes recurrentes

Gibran Fuentes-Pineda

Noviembre 2020

Tareas uno a uno

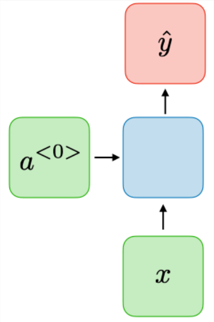


Imagen tomada de Amidi. Recurrent Neural Networks cheatsheet

Tareas uno a muchos

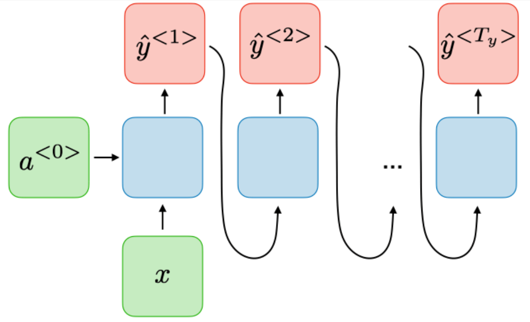


Imagen tomada de Amidi. Recurrent Neural Networks cheatsheet

Tareas muchos a uno

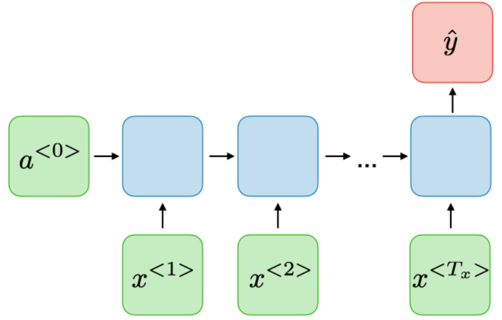


Imagen tomada de Amidi. Recurrent Neural Networks cheatsheet

Tareas muchos a muchos

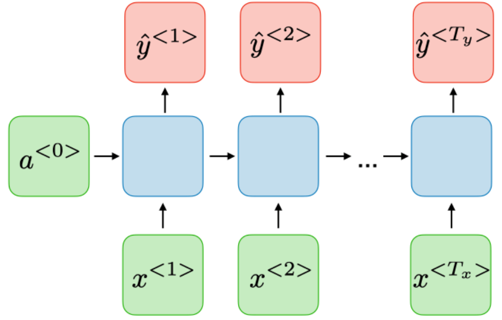


Imagen tomada de Amidi. Recurrent Neural Networks cheatsheet

Tareas muchos a muchos con tiempos distintos

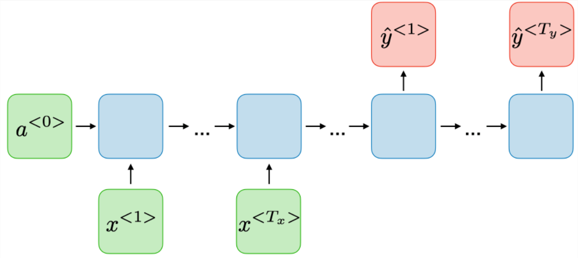


Imagen tomada de Amidi. Recurrent Neural Networks cheatsheet

Redes recurrentes

- Redes con retro-alimentación en sus conexiones

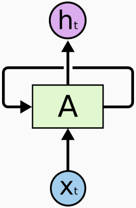


Imagen tomada de Colah 2015 (http://colah.github.io/posts/2015-08-Understanding-LSTMs/)

Elementos básicos

1. Entradas en cada instante de tiempo t ($\mathbf{x}^{[t+1]}$)
2. Estados en cada instante de tiempo t ($\mathbf{h}^{[t+1]}$)
3. Salidas en cada instante de tiempo t ($\mathbf{y}^{[t+1]}$)

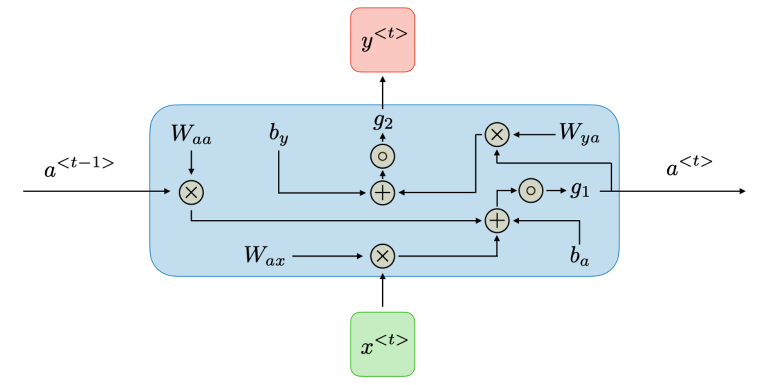


Imagen tomada de Amidi. Recurrent Neural Networks cheatsheet

Tipos de retroalimentación

1. Salida anterior ($\mathbf{y}^{[t]}$)
2. Estado anterior ($\mathbf{h}^{[t]}$)
3. Salida y estado anterior ($\mathbf{h}^{[t]}$ y $\mathbf{y}^{[t]}$)

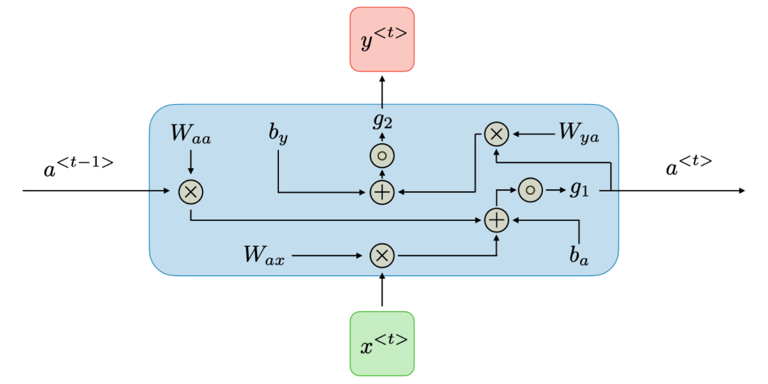


Imagen tomada de Amidi. Recurrent Neural Networks cheatsheet

Celda recurrente básica

- Capa que procesa el estado anterior y la entrada actual para generar un nuevo estado y la salida

$$\mathsf{h}^{[t+1]} = \phi(\mathsf{W}_{hh} \cdot \mathsf{h}^{[t]} + \mathsf{W}_{hx} \cdot \mathsf{x}^{[t+1]} + \mathsf{b}_h)$$

$$\hat{\mathsf{y}}^{[t+1]} = \phi(\mathsf{W}_{yh} \cdot \mathsf{h}^{[t+1]} + \mathsf{b}_y)$$

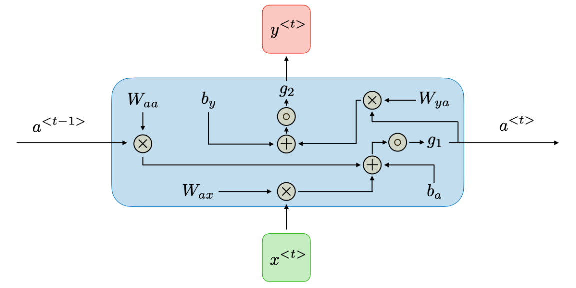
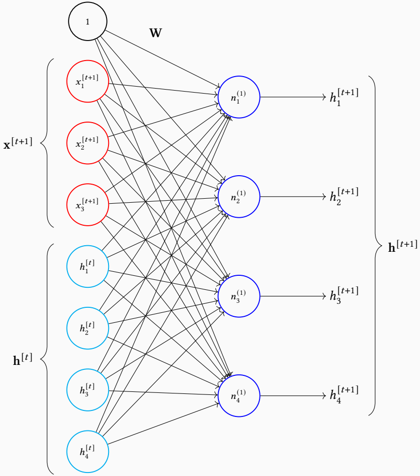


Imagen tomada de Amidi. Recurrent Neural Networks cheatsheet

Celda recurrente básica: otra perspectiva

$$\mathrm{h}^{[t+1]} = \sigma\left(\mathrm{W}_h \cdot \underbrace{\left[\mathrm{h}^{[t]}, \mathrm{x}^{[t+1]}\right]}_{\text{Concatenación}} + \mathrm{b}_h\right)$$



Despliegue de celdas

- Una celda recurrente para una secuencia de T valores, se puede desplegar en T capas con parámetros idénticos

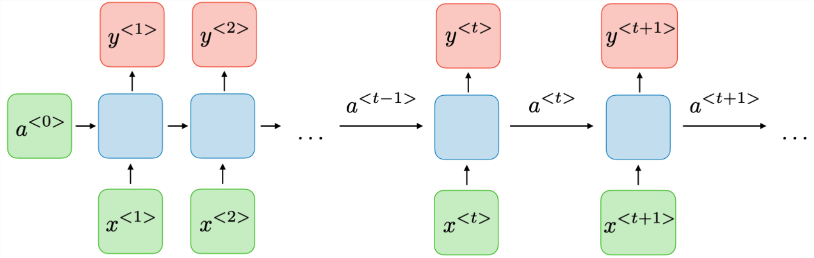


Imagen tomada de Amidi. Recurrent Neural Networks cheatsheet

Red neuronal con celdas recurrentes

- Típicamente contiene celdas recurrentes en conjunto con otras capas
- La salida de una celda puede alimentar otras capas u otras celdas
- Un clasificador simple

$$\mathsf{h}^{[t+1]} = \sigma\left(\mathsf{W}_h \cdot \left[\mathsf{h}^{[t]}, \mathsf{x}^{[t+1]}\right] + \mathsf{b}_h\right)$$
$$\hat{\mathsf{y}}^{[t+1]} = softmax\left(\mathsf{W}_{yh} \cdot \mathsf{h}^{[t+1]} + \mathsf{b}_y\right)$$

Ejemplo: modelo de lenguaje a nivel símbolo

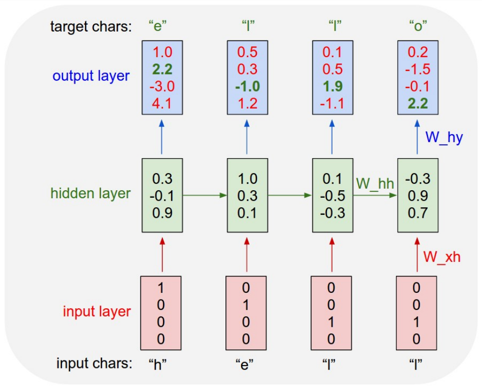


Imagen tomada de Karpathy 2015 (`http://karpathy.github.io/2015/05/21/rnn-effectiveness/`)

Modelando dependencias a corto plazo

- En teoría una red recurrente básica puede modelar dependencias a corto y largo plazo
  - Siegelmann y Sontag mostraron que las redes recurrentes son Turing completas[1]

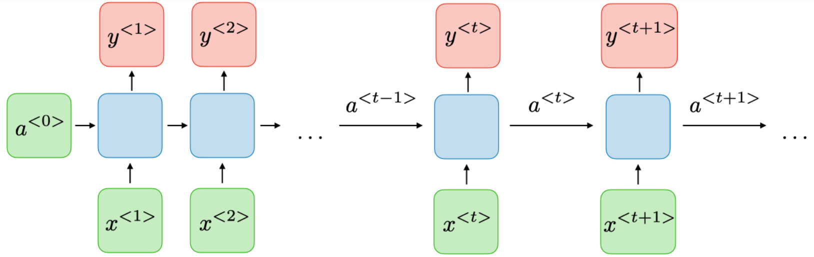


Imagen tomada de Amidi. Recurrent Neural Networks cheatsheet

[1] Siegelmann and Sontag. On The Computational Power Of Neural Nets, 1995.

El problema de la memoria a largo plazo

- En práctica es muy difícil entrenarlas para tareas con dependencias a largo plazo

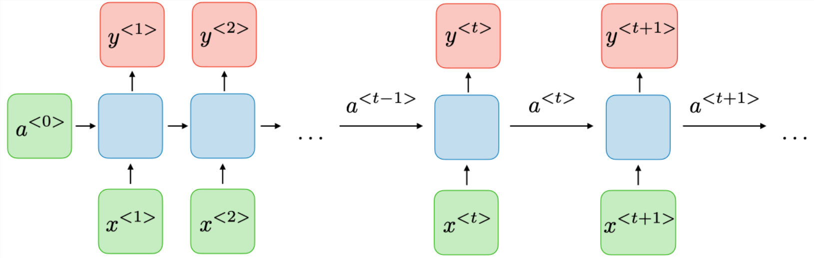


Imagen tomada de Amidi. Recurrent Neural Networks cheatsheet

Memorias a corto y largo plazo

- Agregan elementos internos a la celda básica que permiten capturar dependencias a corto y largo plazo

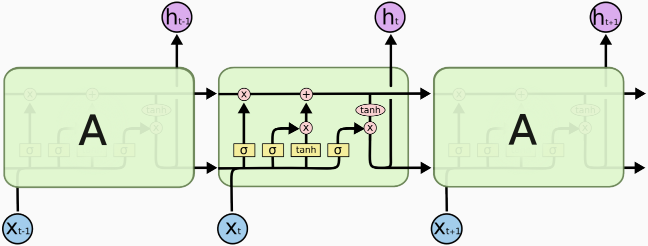


Imagen tomada de Colah 2015 (`http://colah.github.io/posts/2015-08-Understanding-LSTMs/`)

LSTM: salida de la capa anterior

- Agrega o elimina elementos del estado anterior de la celda $\mathbf{C}^{[t]}$ basado en transformación de la entrada actual $\mathbf{x}^{[t+1]}$ y el estado oculto anterior $\mathbf{h}^{[t]}$

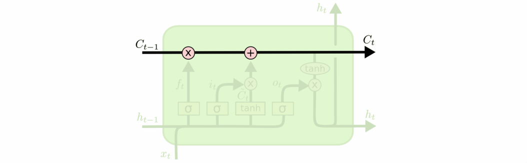


Imagen tomada de Colah 2015 (`http://colah.github.io/posts/2015-08-Understanding-LSTMs/`)

LSTM: compuerta de olvido

- Determina qué olvidar del estado de la celda $\mathbf{C}^{[t]}$ y en qué proporción a partir de la entrada actual $\mathbf{x}^{[t+1]}$ y estado oculto anterior $\mathbf{h}^{[t]}$

$$\mathbf{f}^{[t+1]} = \sigma\left(\mathbf{W}_f \cdot \left[\mathbf{h}^{[t]}, \mathbf{x}^{[t+1]}\right] + \mathbf{b}_f\right)$$

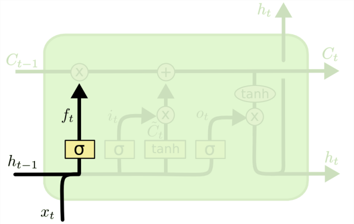


Imagen tomada de Colah 2015 (`http://colah.github.io/posts/2015-08-Understanding-LSTMs/`)

LSTM: computerta de entrada

- Determina qué agregar al estado de la celda $\mathbf{C}^{[t]}$ y en qué proporción a partir de la entrada actual $\mathbf{x}^{[t+1]}$ y estado oculto anterior $\mathbf{h}^{[t]}$

$$\mathbf{i}^{[t+1]} = \sigma\left(\mathbf{W}_i \cdot \left[\mathbf{h}^{[t]}, \mathbf{x}^{[t+1]}\right] + \mathbf{b}_i\right)$$

$$\hat{\mathbf{C}}^{[t+1]} = \tanh\left(\mathbf{W}_C \cdot \left[\mathbf{h}^{[t]}, \mathbf{x}^{[t+1]}\right] + \mathbf{b}_C\right)$$

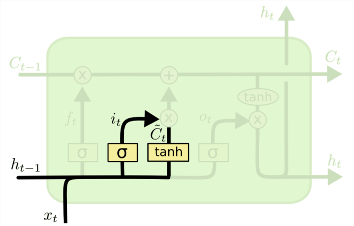


Imagen tomada de Colah 2015 (http://colah.github.io/posts/2015-08-Understanding-LSTMs/)

LSTM: nuevo estado

- El nuevo estado de la celda se obtiene como una combinación de la salida de la compuerta de olvido $\mathbf{f}^{(t)}$ y las salidas $\mathbf{i}^{[t+1]}$ y $\tilde{\mathbf{c}}^{[t+1]}$ de la compuerta de entrada

$$\mathbf{c}^{[t+1]} = \mathbf{f}^{[t+1]} \odot \mathbf{c}^{[t]} + \mathbf{i}^{[t+1]} \odot \tilde{\mathbf{c}}^{[t+1]}$$

donde $\odot$ denota el producto de Hadamard

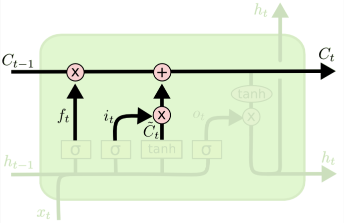


Imagen tomada de Colah 2015 (`http://colah.github.io/posts/2015-08-Understanding-LSTMs/`)

LSTM: computerta de salida

- El siguiente estado oculto $\mathbf{h}^{[t+1]}$ se obtiene como una combinación de la entrada actual $\mathbf{x}^{[t+1]}$, el estado oculto anterior $\mathbf{h}^{[t]}$ y el nuevo estado de la celda $\mathbf{C}^{[t+1]}$

$$\mathbf{o}^{[t+1]} = \sigma\left(\mathbf{W}_o \cdot \left[\mathbf{h}^{[t]}, \mathbf{x}^{[t+1]}\right] + \mathbf{b}_o\right)$$

$$\mathbf{h}^{[t+1]} = \mathbf{o}^{[t+1]} \odot \tanh\left(\mathbf{C}^{[t+1]}\right)$$

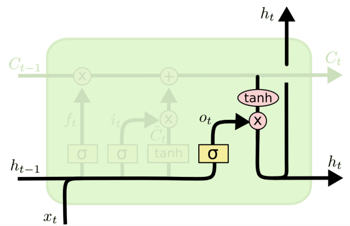


Imagen tomada de Colah 2015 (`http://colah.github.io/posts/2015-08-Understanding-LSTMs/`)

Gated Recurrent Unit

- Combina compuertas de olvido y entrada en una sóla

$$\mathrm{z}^{[t+1]} = \sigma\left(\mathrm{W}_z \cdot \left[\mathrm{h}^{[t]}, \mathrm{x}^{[t+1]}\right] + \mathrm{b}_z\right)$$
$$\mathrm{r}^{[t+1]} = \sigma\left(\mathrm{W}_r \cdot \left[\mathrm{h}^{[t]}, \mathrm{x}^{[t+1]}\right] + \mathrm{b}_r\right)$$
$$\tilde{\mathrm{h}}^{[t+1])} = \tanh\left(\mathrm{W}_h \cdot \left[\mathrm{r}^{[t+1]} \odot \mathrm{h}^{[t]}, \mathrm{x}^{[t+1]}\right] + \mathrm{b}_h\right)$$
$$\mathrm{h}^{[t+1]} = \left(1 - \mathrm{z}^{[t+1]}\right) \odot \mathrm{h}^{[t]} + \mathrm{z}^{[t+1]} \odot \tilde{\mathrm{h}}^{[t+1]}$$

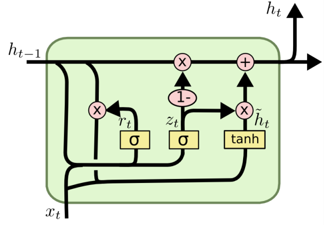


Imagen tomada de Colah 2015 (`http://colah.github.io/posts/2015-08-Understanding-LSTMs/`)

Retropropagación en el tiempo

- Pérdida en el tiempo

$$\mathcal{L}\left(\hat{\mathrm{y}}, \mathrm{y}) = \sum_{t=1}^{T} \mathcal{L}(\hat{\mathrm{y}}^{[t]}, \mathrm{y}^{[t]}\right)$$

- Retropropagación

$$\frac{\partial \mathcal{L}^{[T]}}{\partial \boldsymbol{\theta}} = \sum_{t=1}^{T} \frac{\partial \mathcal{L}^{[t]}}{\partial \boldsymbol{\theta}}$$

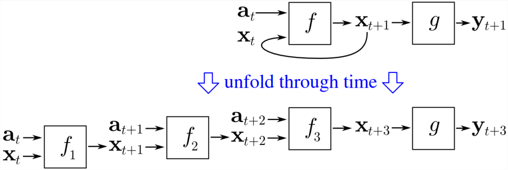


Imagen tomada de Wikipedia (`https://en.wikipedia.org/wiki/Backpropagation_through_time`)

Redes recurrentes apiladas

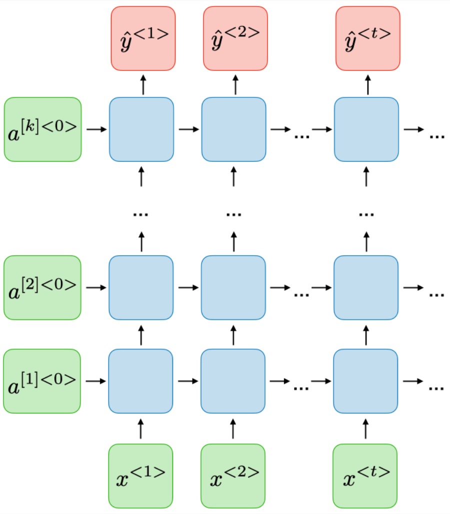


Imagen tomada de Amidi. Recurrent Neural Networks cheatsheet

RNR Bidireccional

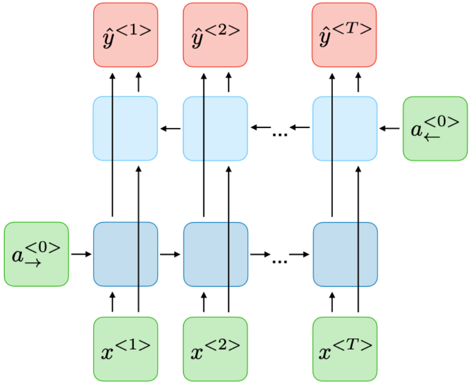


Imagen tomada de Amidi. Recurrent Neural Networks cheatsheet

Modelos secuencia a secuencia

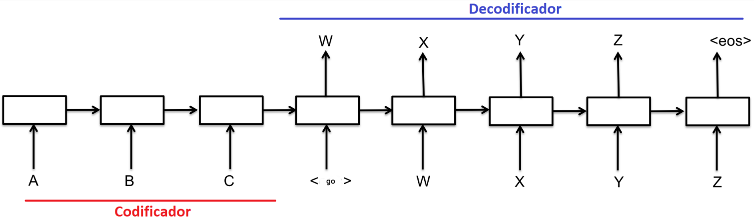


Imagen derivada de `https://www.tensorflow.org/tutorials/seq2seq`