

# Tutorial: GRU4Rec

Patricio Cerda Mardini


IIC3633  
Sistemas Recomendadores



# Contenidos tutorial

- 1 Definición del problema
- 2 Técnicas
- 3 Notebook
- 4 Extensiones
- 5 Referencias



- **Repositorio:**  /pcerdam/KerasGRU4Rec/tree/DemoIIC3633
- **Notebook:** link en el aviso siding de ayer
- **Ambiente:** Python3 + GPU
- Ejecutar todo ahora, demora un poco!



Recomendación basada en sesiones para *e-commerce*.

¿Qué es una sesión?

- Usuario anónimo
- *Secuencia de clicks*: visitas orgánicas a distintos productos



# Dataset: RecSys Challenge 2015

Usaremos el set de datos publicado para el desafío RecSys del 2015.

- Sesiones anónimas en *e-commerce* europeo grande (7 millones)
- Proveedor: empresa YooChoose - *outsourcing* de RS

En este caso, solo hay feedback implícito: si el usuario/sesión visitó al ítem, y cuándo:

Session	Item	Timestamp
1	25	5
1	34	10
1	17	15
2	98	18
...	...	...

RSC15 ofrece 31 millones de estos eventos.



# ¿Problema a resolver?

## Next-item prediction

Dada una secuencia de items previamente visitados, predecir el siguiente item que el usuario verá.

En la práctica (*test time*), se recomienda la lista top-N predicha:

$(25, 34) \rightarrow ?$

$? = [5, \mathbf{17}, 68, 99, 7]$



# Contenidos tutorial

- 1 Definición del problema
- 2 Técnicas**
- 3 Notebook
- 4 Extensiones
- 5 Referencias



# Trabajo previo

## Item k-Nearest Neighbors

Matriz de similaridad: co-ocurrencia de ítems en base a clicks  
Usa solamente el último click como prior

## Markov Decision Process

$\langle \text{estados}, \text{acciones (recomendaciones)}, \text{recompensas}, \text{transiciones} \rangle$

Inmanejable si se incluye todas las posibles secuencias de acciones del usuario

## General Factorization Framework

Usa info. pasada, pero no considera el orden de la secuencia





Influyente trabajo publicado en ICLR 2016.

## SESSION-BASED RECOMMENDATIONS WITH RECURRENT NEURAL NETWORKS

**Balázs Hidasi** \*  
Gravity R&D Inc.  
Budapest, Hungary  
balazs.hidasi@gravityrd.com

**Alexandros Karatzoglou**  
Telefonica Research  
Barcelona, Spain  
alexk@tid.es

**Linus Baltrunas** †  
Netflix  
Los Gatos, CA, USA  
lbaltrunas@netflix.com

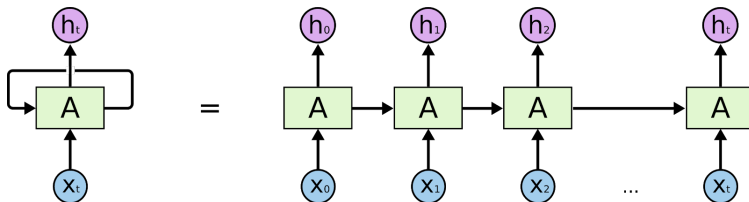
**Domonkos Tikk**  
Gravity R&D Inc.  
Budapest, Hungary  
domonkos.tikk@gravityrd.com

Primera aproximación al problema mediante redes neuronales, con buenos resultados.



# Recurrent Neural Networks

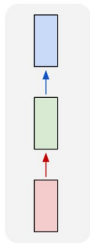
Tratan datos secuenciales



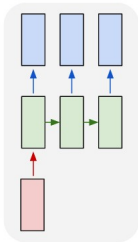
# Recurrent Neural Networks

Arquitectura personalizable según tipo de entrada y salida

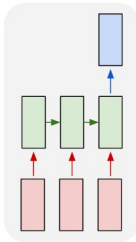
one to one



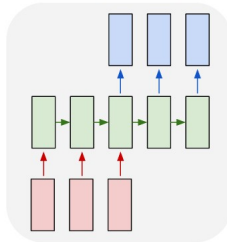
one to many



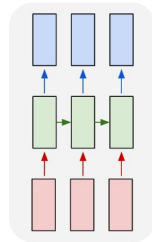
many to one



many to many

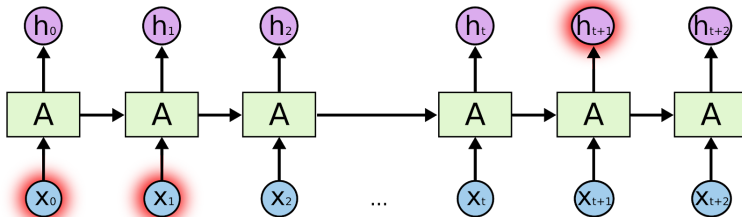


many to many



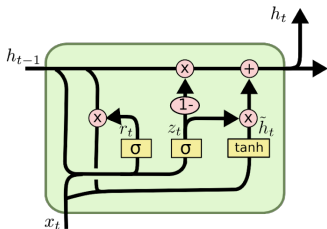
# RNN: vanishing gradient

Aún cuando en teoría se puede, la arquitectura típica no es capaz de capturar dependencias de largo plazo.



# Gated Recurrent Units

Resuelven el problema del gradiente. Propuestas por Bengio et al. en 2014.



$$z_t = \sigma(W_z \cdot [h_{t-1}, x_t])$$

$$r_t = \sigma(W_r \cdot [h_{t-1}, x_t])$$

$$\tilde{h}_t = \tanh(W \cdot [r_t * h_{t-1}, x_t])$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

Permite recordar y olvidar, selectivamente, por intervalos indeterminados de tiempo. Gradiente no desaparece ni explota.

Entrena más rápido que una LSTM pues tiene menos parámetros.



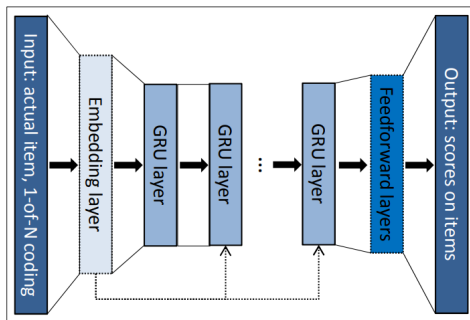
Idea: aprovechar dependencias de largo plazo en la sesión para mejorar las recomendaciones.

¿Qué pasa si un item visto hace muchos clicks es clave para sugerir algo que el cliente termine comprando? La GRU surge como opción.

Esto se extiende naturalmente si ahora tenemos cookies que nos permiten seguir a un usuario entre visitas distintas al portal de e-commerce.



Modelo neuronal muy simple. Procesa secuencialmente sesiones para predecir el siguiente item.



100 hidden units, dropout 0.5



En Keras:

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	(512, 1, 8725)	0
cu_dnngru_1 (CuDNNGRU)	[(512, 100), (512, 100)]	2648100
dropout_1 (Dropout)	(512, 100)	0
dense_1 (Dense)	(512, 8725)	881225
Total params: 3,529,325		
Trainable params: 3,529,325		
Non-trainable params: 0		

Optimizador: ADAM, parámetros por defecto.

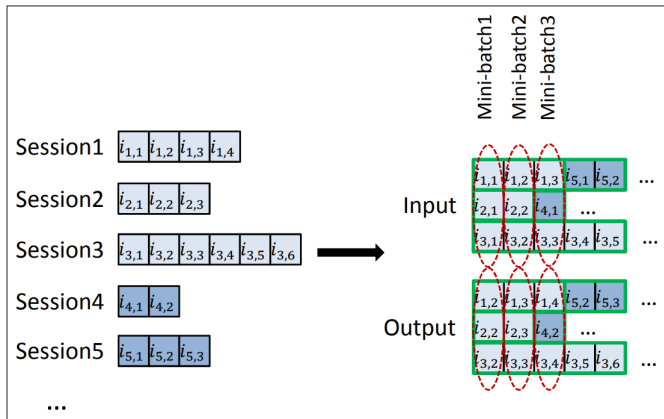
Función de pérdida: *cross-entropy*.





# Modificación

Para entrenar eficientemente, se procesan sesiones en paralelo:



Notar que se analiza cada sesión por completo.



# Algunas consideraciones

- 1 Se ordenan las sesiones
- 2 Para cada sesión, si el evento actual es  $t$ , el *target* es el evento siguiente  $t + 1$
- 3 Al reemplazar una sesión finalizada por la siguiente, se resetea el estado oculto de la unidad GRU



Se utilizan dos métricas para evaluar el desempeño del sistema:

- 1 Recall @ 20
- 2 Mean Reciprocal Rank @ 20

$$\text{Recall} = \frac{\text{relevant items}}{\text{total items}}$$

$$\text{MRR} = \frac{1}{|Q|} \sum_{i=1}^{|Q|} \frac{1}{\text{rank}_i}$$



Los sistemas recomendadores contra los que los autores se comparan son:

- 1 Most Popular (POP)
- 2 Session Most Popular (S-POP)
- 3 BPR-MF



Los resultados reportados en el paper son:

Method	Recall@20	MRR@20
POP	0.0050	0.0012
S-POP	0.2672	0.1775
BPR-MF	0.2574	0.0618
GRU4Rec	<b>0.5781</b>	<b>0.2375</b>



# Contenidos tutorial

- 1 Definición del problema
- 2 Técnicas
- 3 Notebook**
- 4 Extensiones
- 5 Referencias



# Contenidos tutorial

- 1 Definición del problema
- 2 Técnicas
- 3 Notebook
- 4 Extensiones**
- 5 Referencias



# Loss functions

Aparte de *cross-entropy*, autores experimentan con otras familias de funciones de pérdida.

Las tipo *pairwise* comparan el ranking del *target* versus otros items. Las utilizadas en el paper original son BPR y Top1:

$$L_{\text{bpr}} = -\frac{1}{N_S} \sum_{j=1}^{N_S} \log \sigma(r_i - r_j) \quad L_{\text{top1}} = \frac{1}{N_S} \sum_{j=1}^{N_S} \sigma(r_j - r_i) + \sigma(r_j^2)$$

Problema: *vanishing gradient*!





Solución: familia de funciones *pairwise max-ranking*. De la forma:

$$L_{\text{pairwise-max}} \left( r_i, \{r_j\}_{j=1}^{N_S} \right) = L_{\text{pairwise}}(r_i, \max_j r_j)$$

Modificaciones dan lugar a  $BPR_{\max}$  y  $Top1_{\max}$ :

$$L_{\text{bpr-max}} = -\log \sum_{j=1}^{N_S} s_j \sigma(r_i - r_j) \quad L_{\text{top1-max}} = \sum_{j=1}^{N_S} s_j \left( \sigma(r_j - r_i) + \sigma(r_j^2) \right)$$



# Resultados

Loss function	Recall@20	MRR@20
Cross Entropy	0.5781	0.2375
Top1	0.6117	0.2367
BPR	0.6322	0.2467
Top1Max	0.7086	0.3045
BPRMax	<b>0.7211</b>	<b>0.3170</b>

Se obtienen mejoras de un 23.2 % en R@20, y un 37.5 % en MRR@20.

Esto, con  $2^{11}$  negative samples adicionales respecto al trabajo original.



Podemos utilizar las *timestamp* en los datos para determinar el *Dwell Time*: cuánto tiempo permanece el usuario visitando cada item.

## Supuesto

A mayor *Dwell Time*, más afinidad item-usuario hay

V. Bogina y T. Kuflik siguen esta idea en “*Incorporating Dwell Time in Session-Based Recommendations with Recurrent Neural Networks*”.



# Item Boosting

Sea una sesión:

$$s_i = \{x_{i1}, x_{i2}, \dots, x_{in}\}$$

Cada item posee un *dwell time*  $dt_{ij}$

Dado threshold<sup>1</sup>  $t$ , la sesión inflada queda:

$$s'_i = \{x_{i1} \cdot (1 + \frac{dt_{i1}}{t}), x_{i2} \cdot (1 + \frac{dt_{i2}}{t}), \dots, x_{in} \cdot (1 + \frac{dt_{in}}{t})\}$$

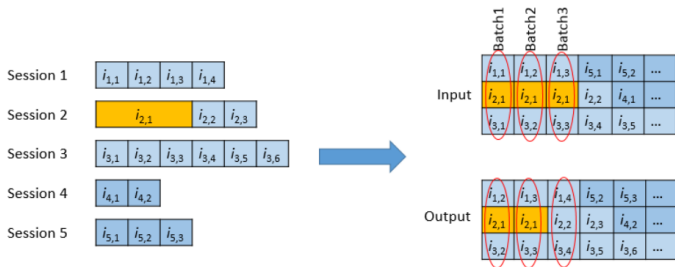
---

<sup>1</sup> $t$  hiperparámetro. Autores usan  $t = 75$ .



# Item Boosting

- Preprocesamiento sobre el *training set*



# Resultados para RSC15

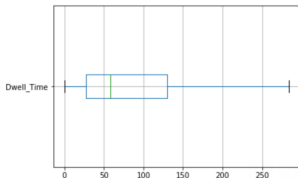


Figure 3. Boxplot with statistics on the data set's dwell time

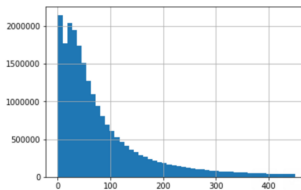


Figure 4. Dwell time distribution that is limited to items with dwell time less than 450 seconds.

Table 1. Comparison between different results

Method	recall@20	MRR@20
<i>GRU4Rec [5]</i>	<i>0.5853</i>	<i>0.2305</i>
GRU4Rec with dwell time threshold 75	0.7885	0.5834
GRU4Rec with dwell time threshold 100	0.7754	0.548
<i>GRU4Rec with sampling [6]</i>	<i>0.7117</i>	<i>0.308</i>
GRU4Rec with sampling and dwell time threshold 100	0.84	0.61
GRU4Rec with sampling and dwell time threshold 75	<b>0.853</b>	<b>0.636</b>



- Las RNN son una gran herramienta para hacer recomendaciones
- Agnóstico al dominio: cualquier dataset de sesiones puede usarse
- Framework Keras permite rápida experimentación y prototipado



# Contenidos tutorial

- 1 Definición del problema
- 2 Técnicas
- 3 Notebook
- 4 Extensiones
- 5 Referencias**





- [1] Hochreiter, S. & Schmidhuber, J. Long short-term memory. Neural Comput. 9,1735–1780 (1997).
- [2] Cho, K. et al (2014). Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation.
- [3] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, Domonkos Tikk: Session-based recommendations with recurrent neural networks. ICLR, 2016
- [4] Hidasi, B., & Karatzoglou, A. (2018). Recurrent Neural Networks with Top-k Gains for Session-based Recommendations. CIKM.
- [5] Bogina, Veronika and Tsvi Kuflik. “Incorporating Dwell Time in Session-Based Recommendations with Recurrent Neural Networks.” RecTemp@RecSys (2017).

