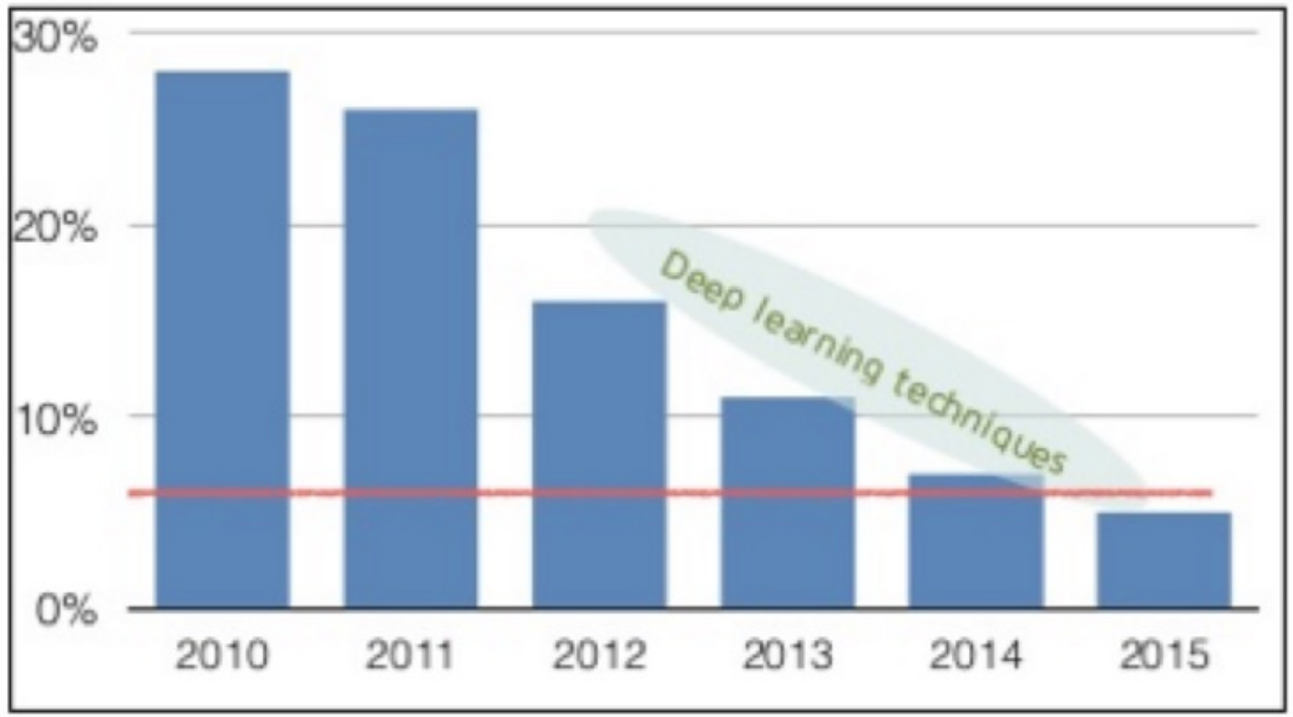
Deep Learning en Sistemas Recomendadores y Filtrado Colaborativo

Denis Parra, PhD

En esta clase

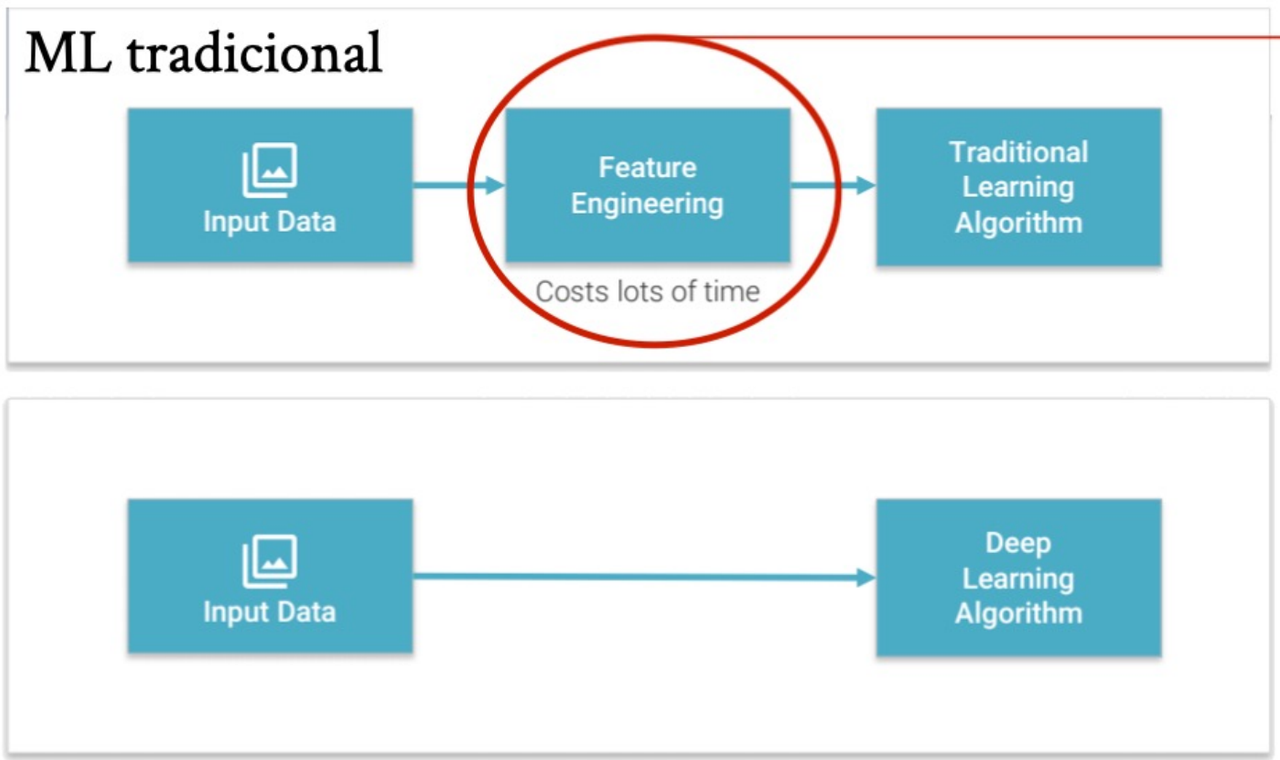
1. Introducción a redes neuronales
2. Recomendación con redes neuronales
3. Tutorial de *VisRank + Pytorch*

¿Por qué Deep Learning?



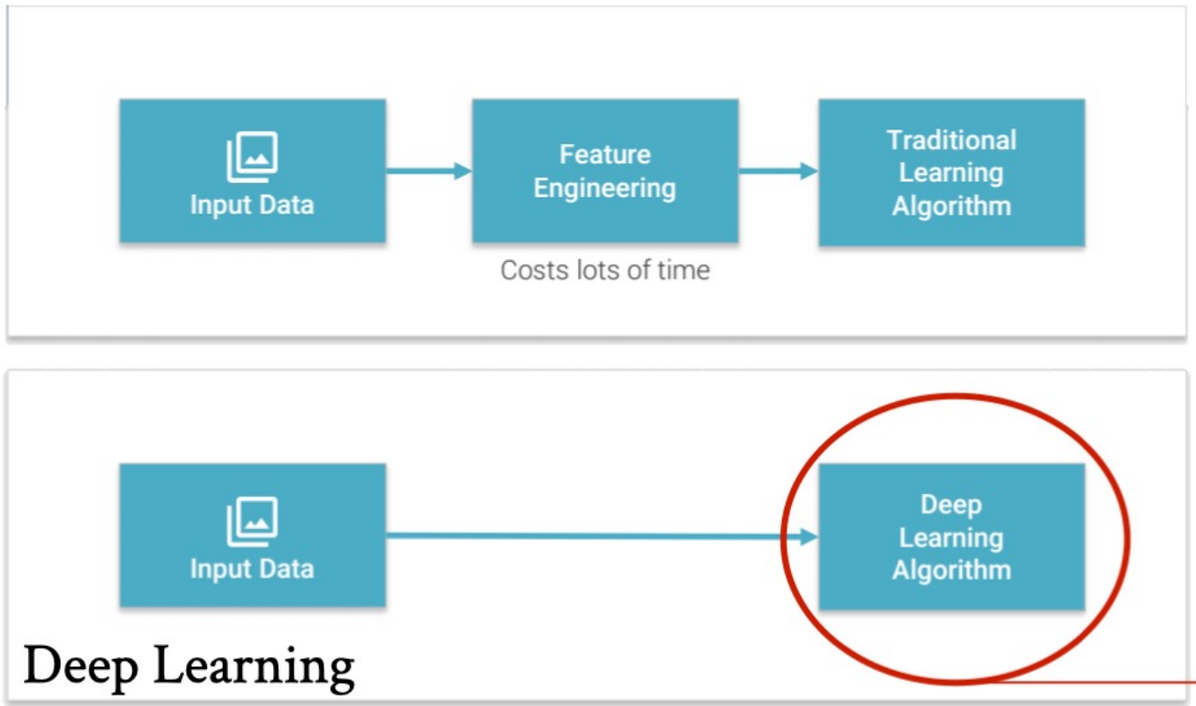
ImageNet challenge error rates (red line = human performance)

¿Por qué Deep Learning?



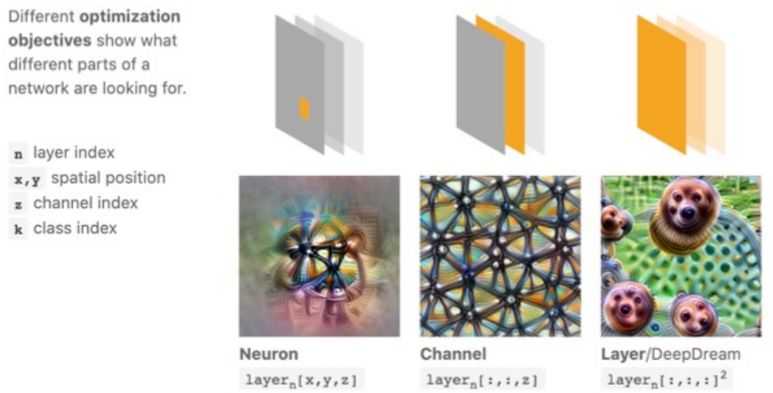
Brightness (brillo)
Saturation (saturación)
Sharpness (nitidez)
RMS-contrast (contraste RMS)
Colorfulness (colorido)
Naturalness (naturalidad)
Etc.

¿Por qué Deep Learning?

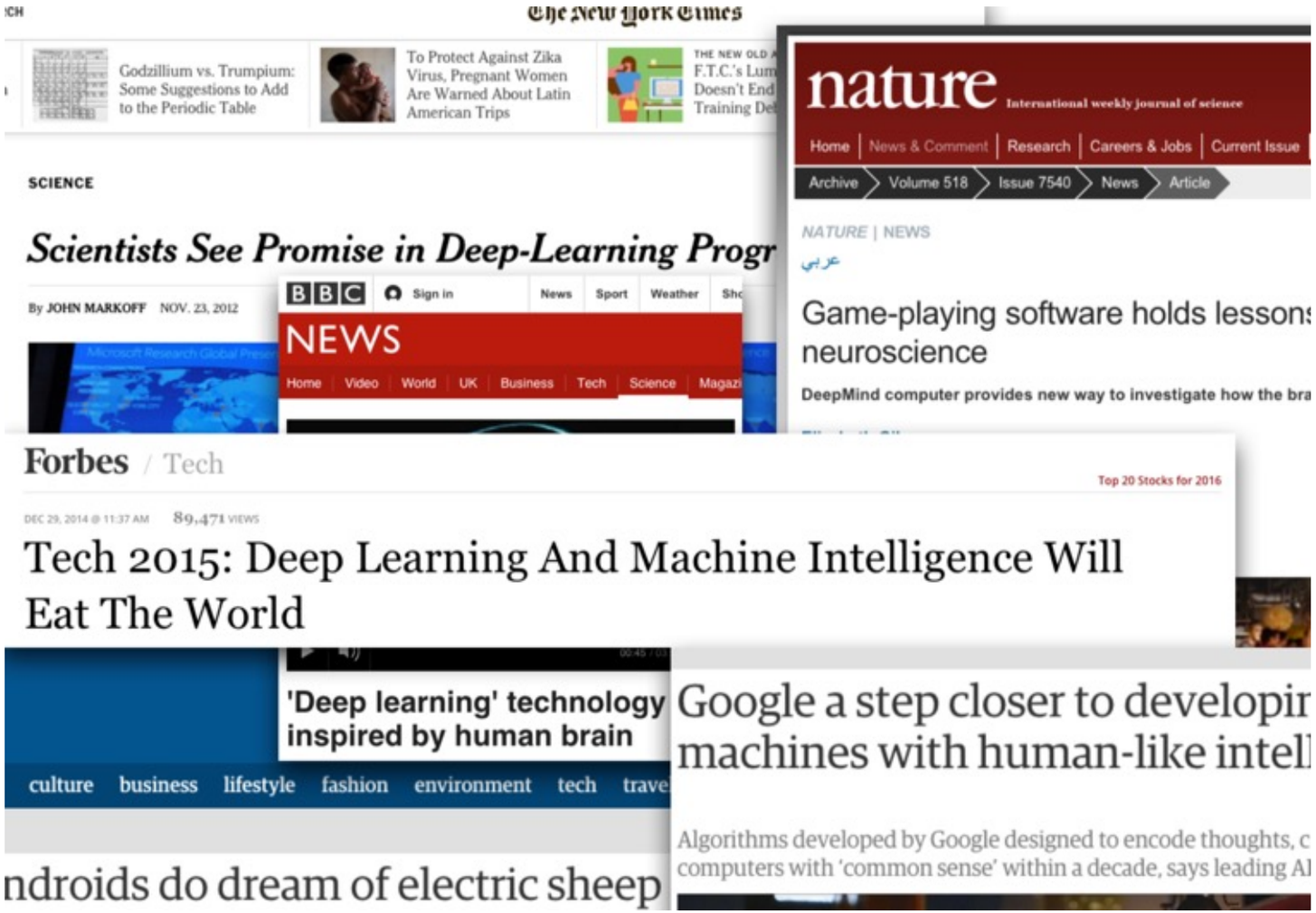


https://distill.pub/2017/feature-visualization/

Olah, et al., "Feature Visualization", Distill, 2017.

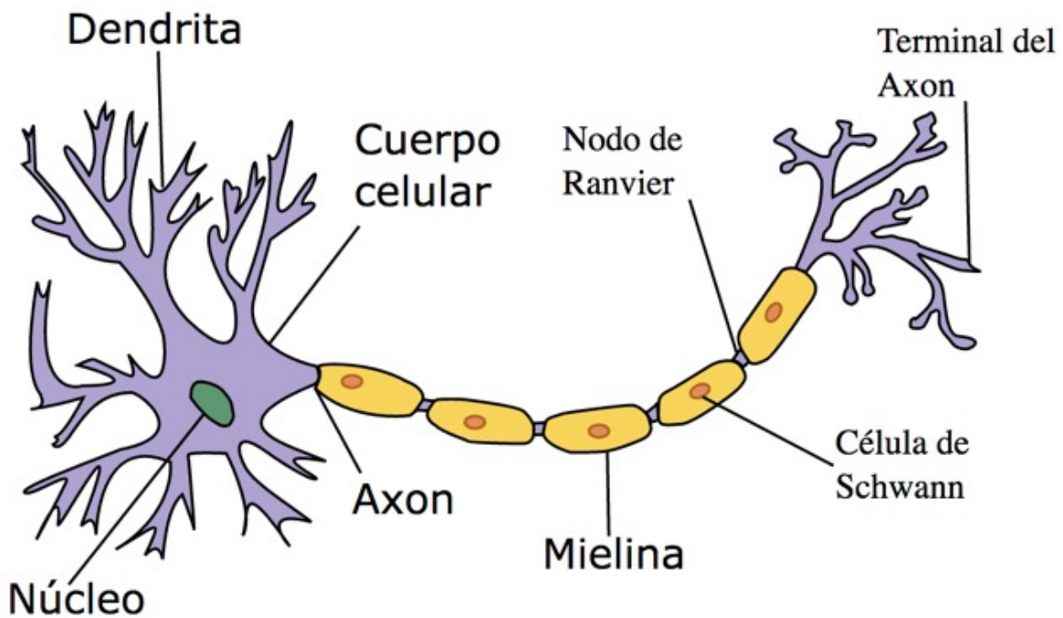


¿Por qué Deep Learning?

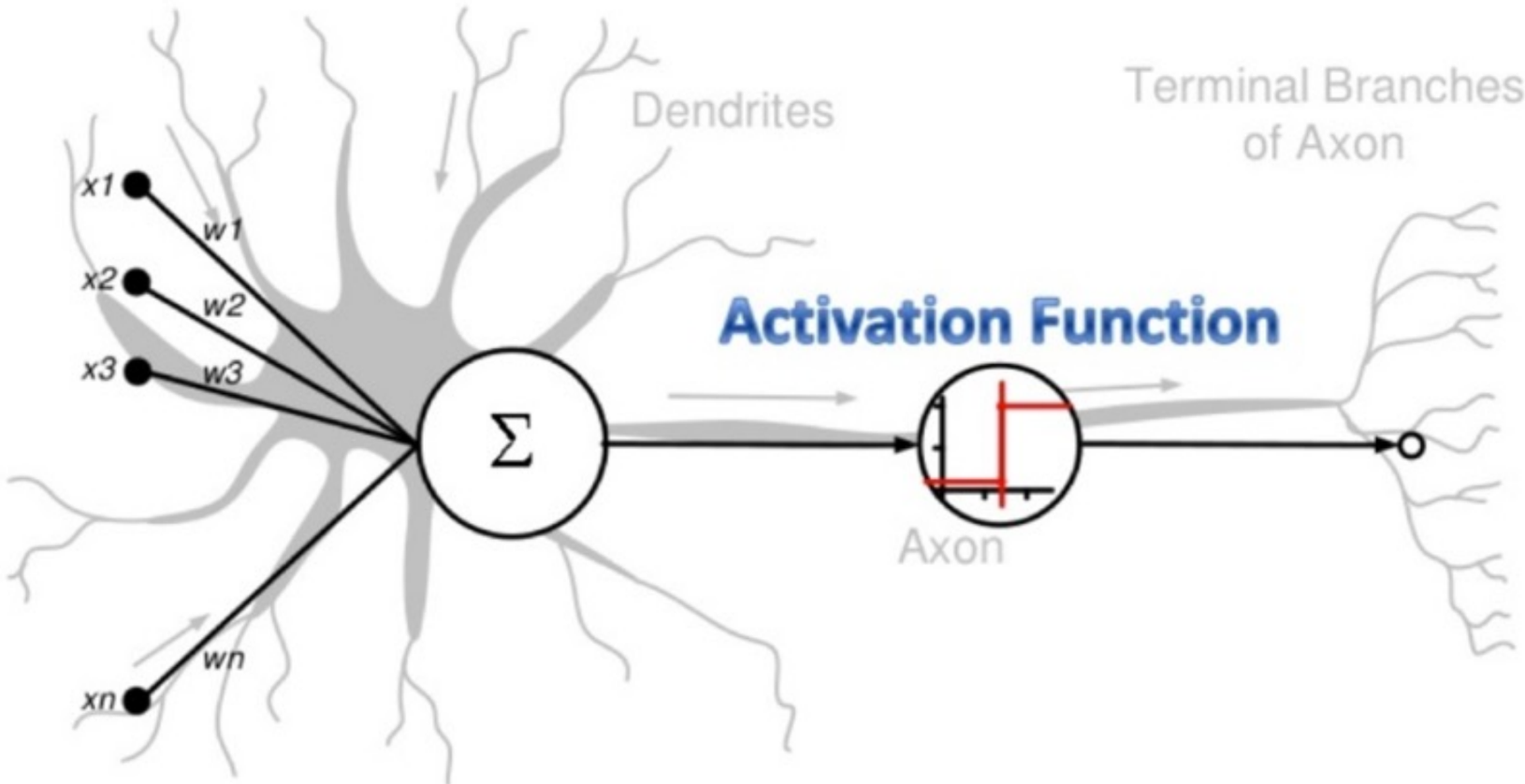


Base Biológica: Neurona

- Tipo de células del sistema nervioso cuya principal característica es la excitabilidad eléctrica de su membrana plasmática

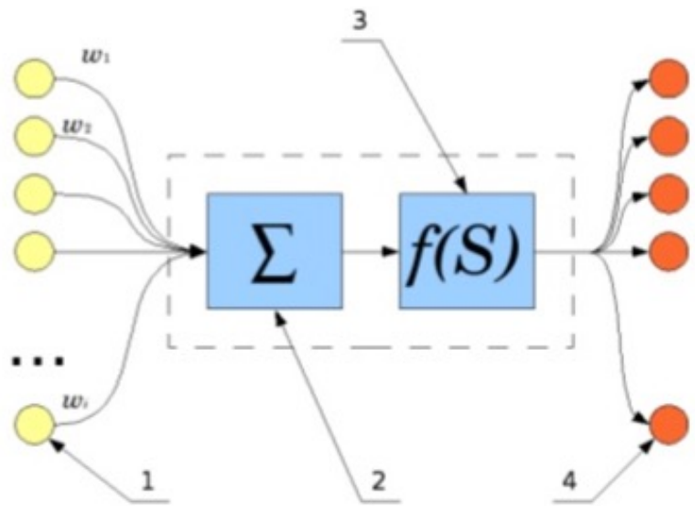


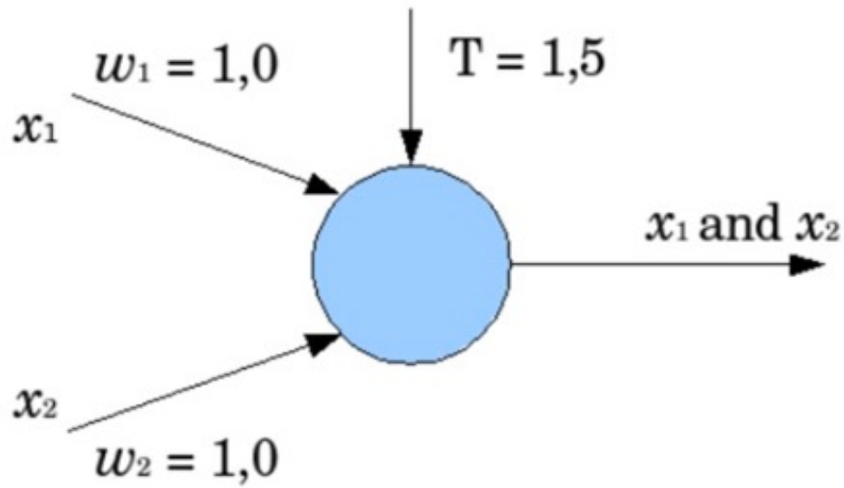
Redes Neuronales Artificiales



Redes Neuronales Artificiales

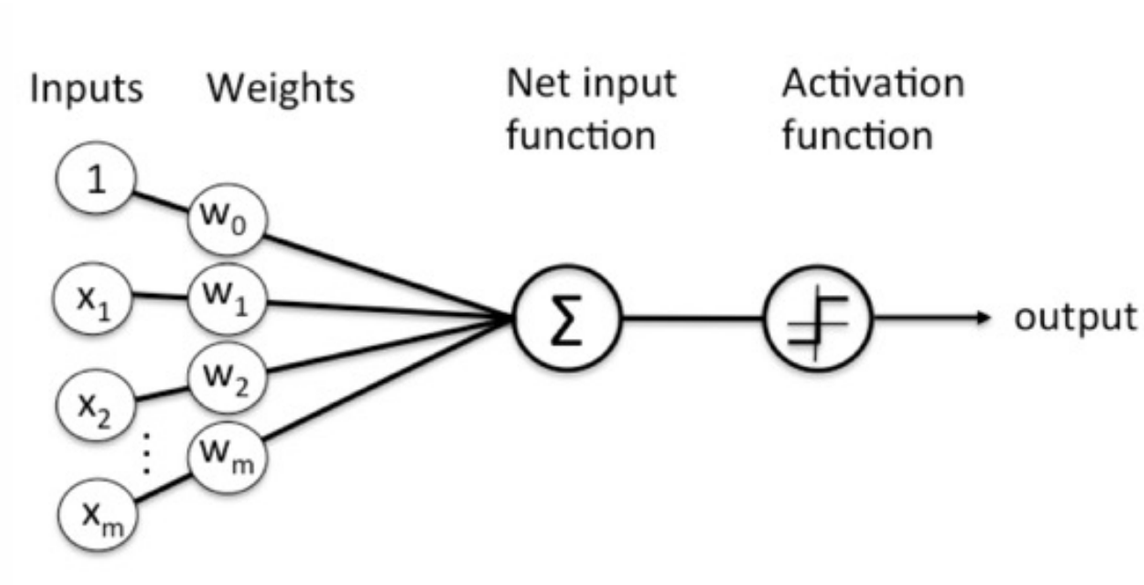
- 1943: McCulloh y Pitts “A Logical Calculus of the Ideas Immanent in Nervous Activity”





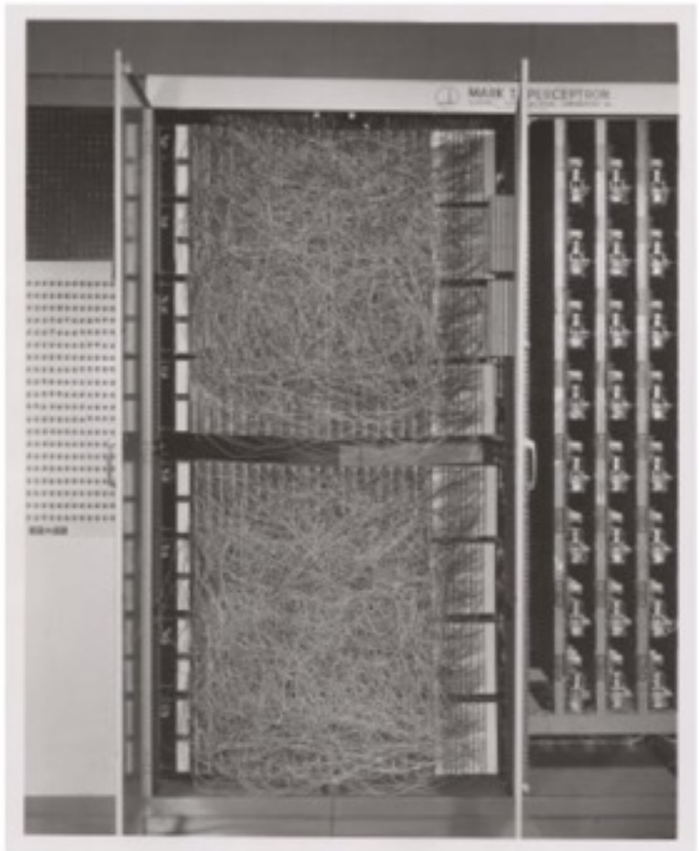
Perceptrón

- 1957: Frank Rosenblatt



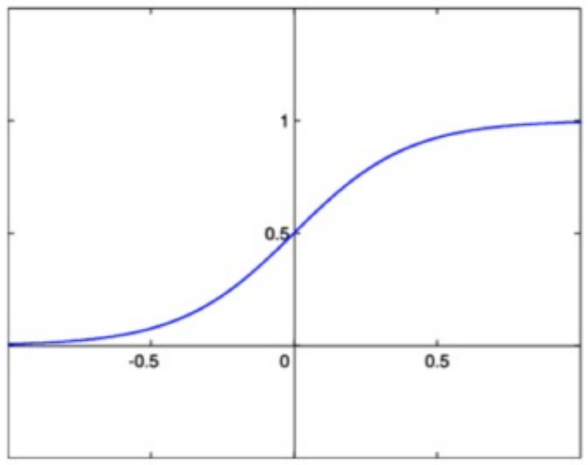
"[The Perceptron is] the embryo of an electronic computer that [the Navy] expects will be able to walk, talk, see, write, reproduce itself and be conscious of its existence."

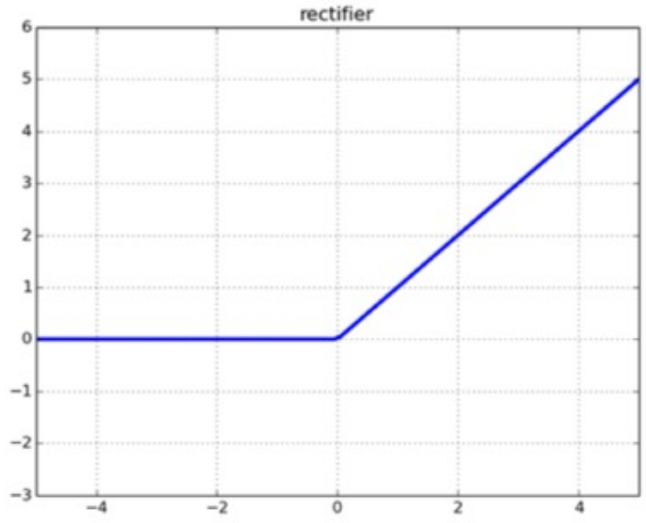
THE NEW YORK TIMES



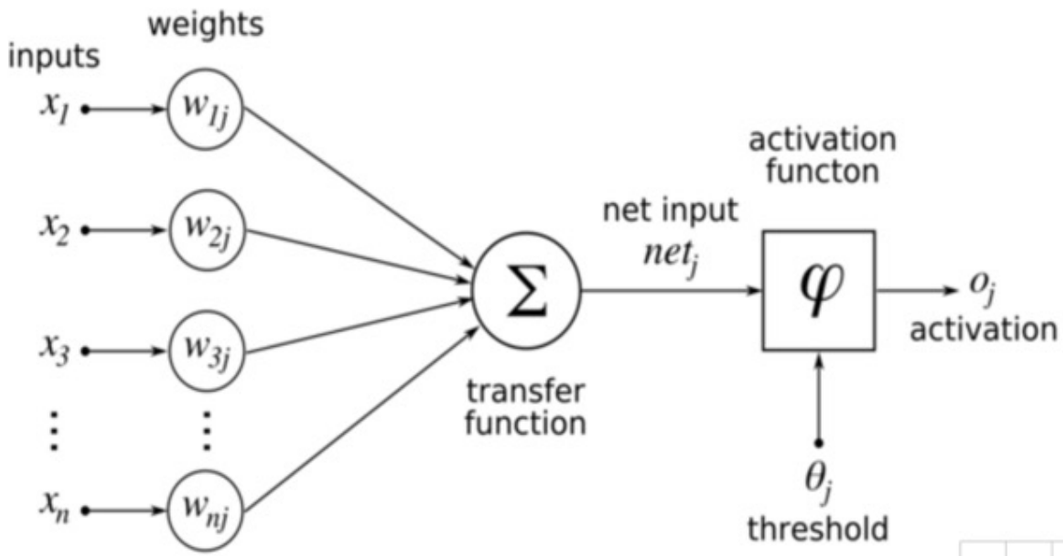
Función de Activación

- Step, tanh, sigmoid, ReLU



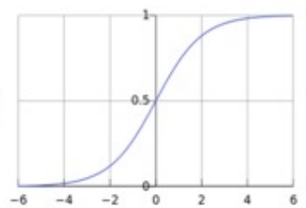


Perceptrón



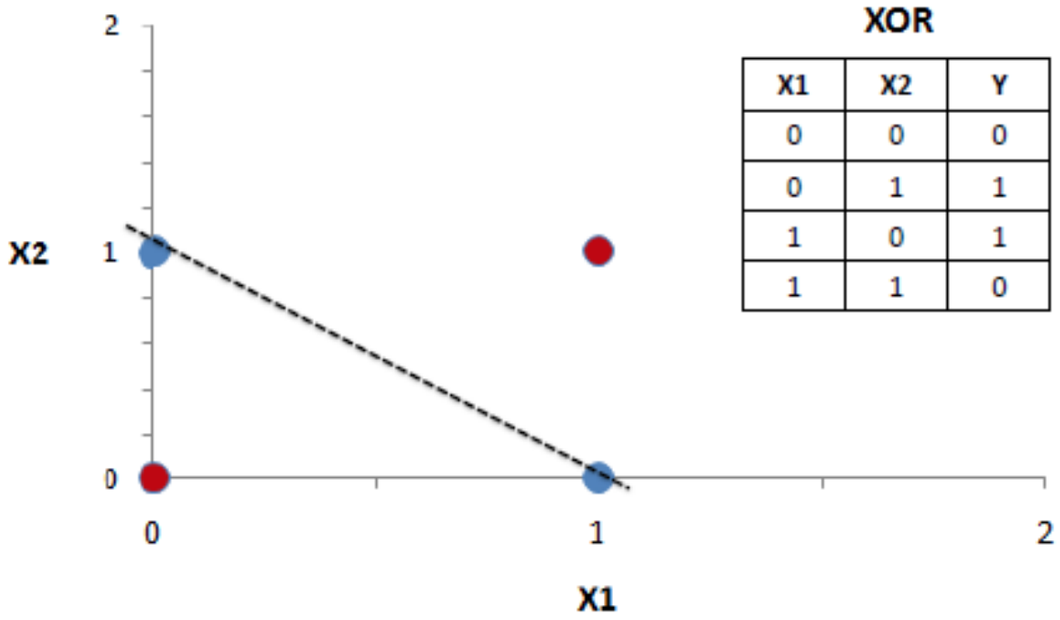
$$P(o_j = 1|x) = \phi\left(\sum_{i=1}^{n} w_{ij}x_i + \theta_j\right)$$

$$\phi = \frac{1}{1 + e^{-\Sigma}}$$



Limitaciones del Perceptrón

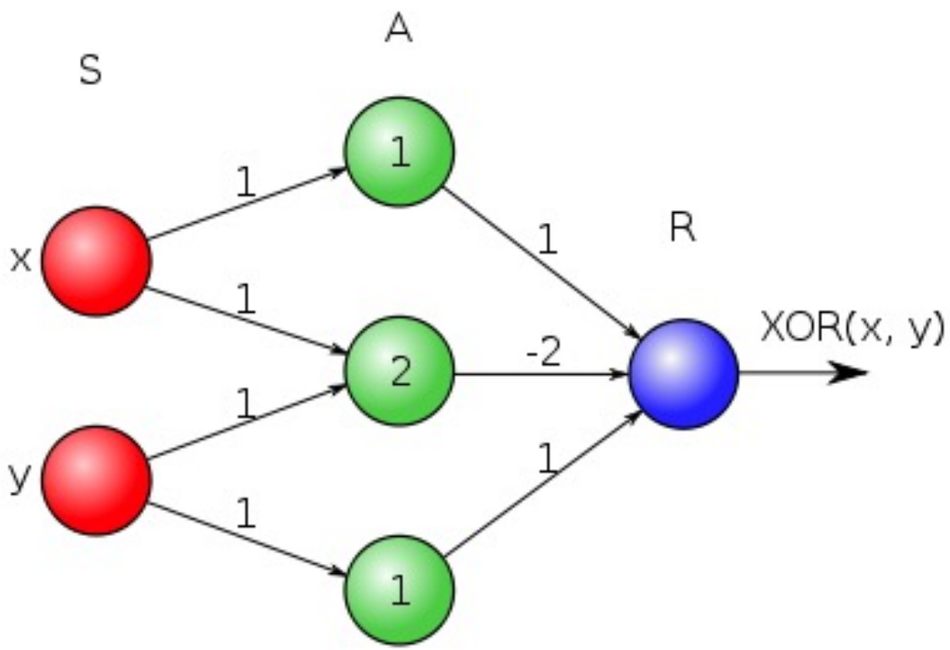
- El perceptron de una capa es un clasificador lineal
- No puede separar alguna funciones como el XOR



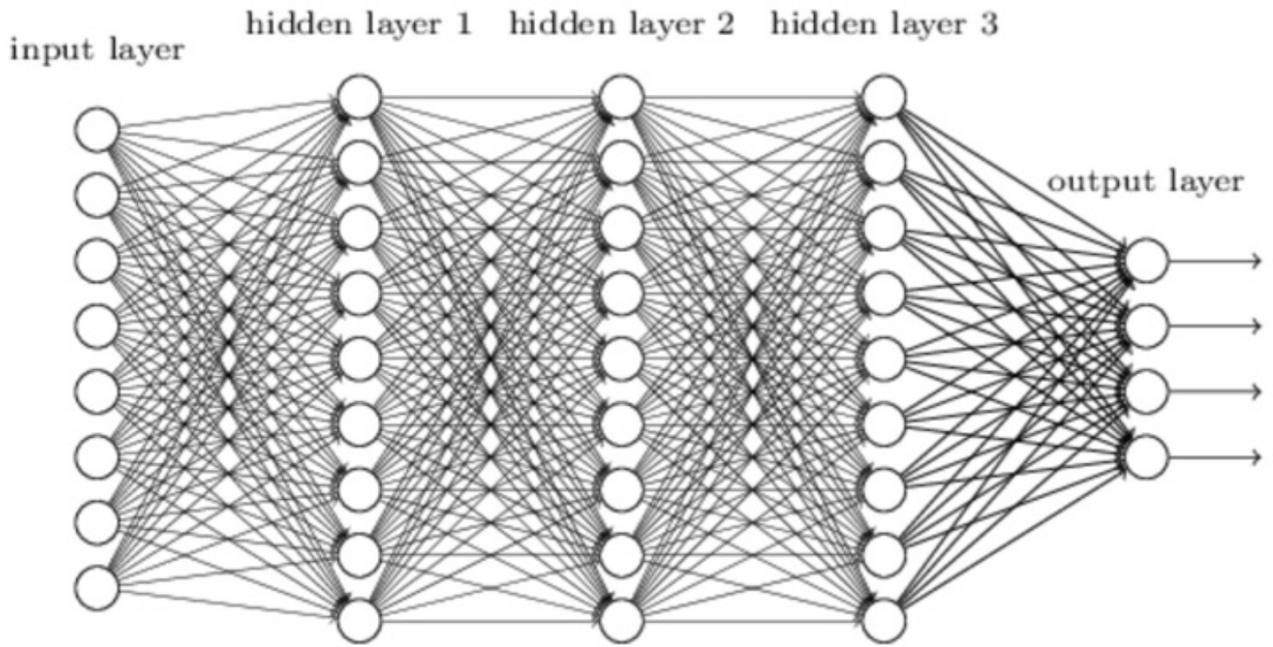
XOR

X1	X2	Y
0	0	0
0	1	1
1	0	1
1	1	0

Perceptron con capas ocultas



Redes Feedforward Multilayer



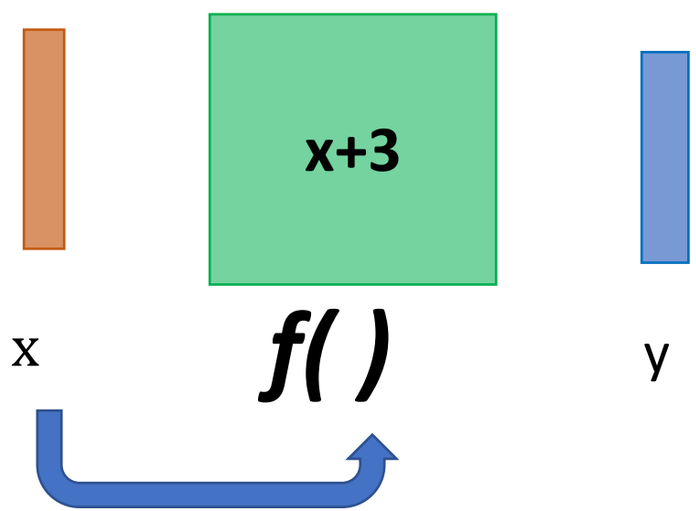
$$F(\mathbf{x}) := \sigma(\ldots \mathbf{W}^2 \sigma(\mathbf{W}^1 \mathbf{x}))$$

¿Qué es una red neuronal artificial III?

- Podemos pensar en la red neuronal como una función ***f( )***

Por ejemplo, si nos dan una función ***f(x) = x +3***

Se puede, luego, hacer una tabla de los pares de valores

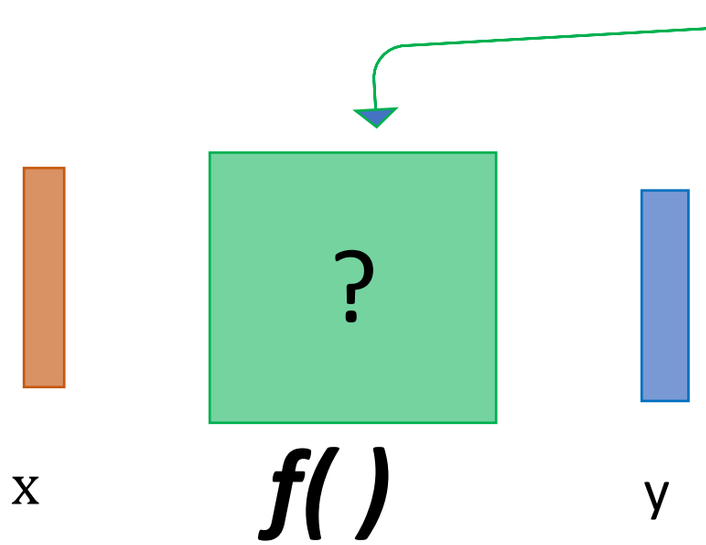


Entrada: x	y = f(x)
0	3
1	4
2	5
3	6
4	7
5	8
…	….

¿Qué es una red neuronal artificial III?

- Podemos pensar en la red neuronal como una función ***f( )***

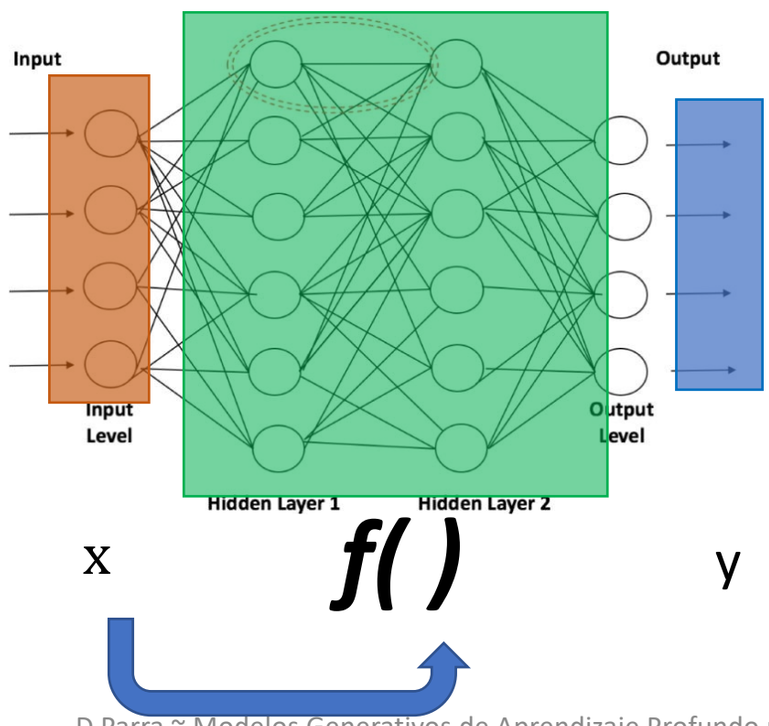
¿ y si nos pidieran ADIVINAR o APRENDER la función
f(x) = …
a partir de la tabla de pares de números?



Entrada: x	y = f(x)
0	3
1	4
2	5
3	6
4	7
5	8
…	….

¿Qué es una red neuronal artificial III?

- Podemos pensar en la red neuronal como una función ***f( )***

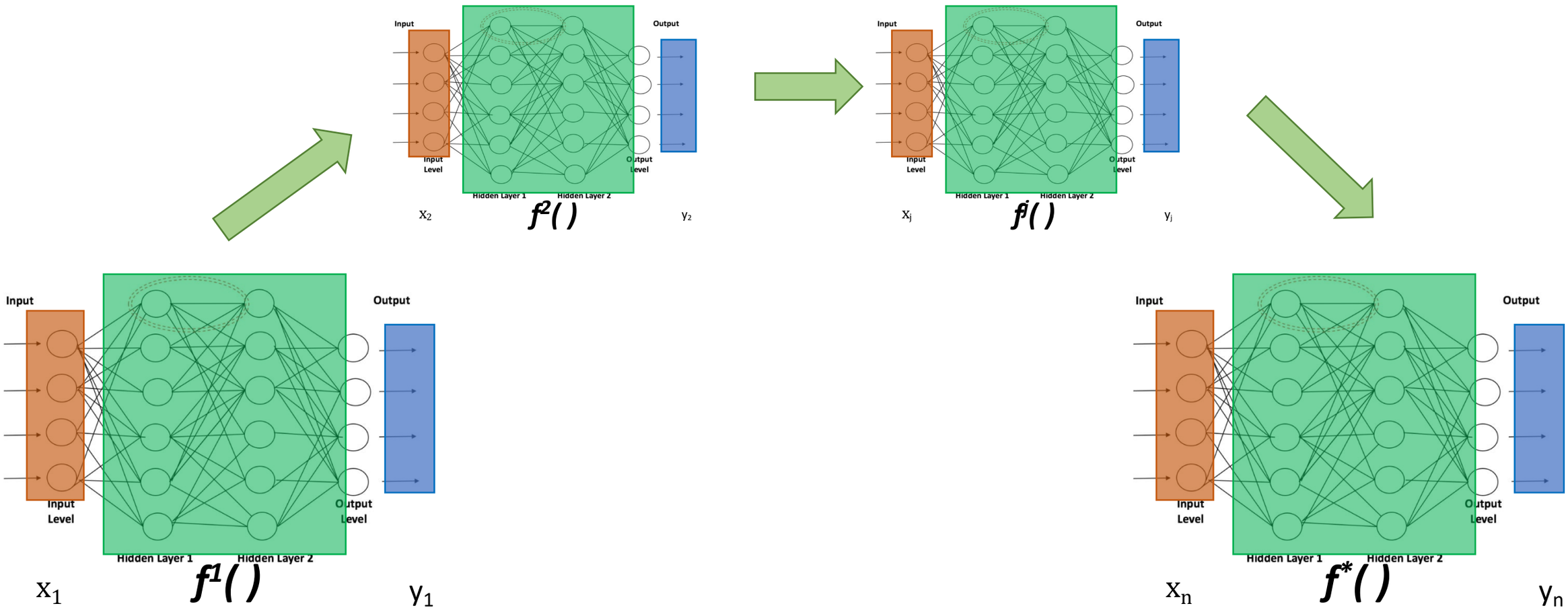


Podemos ver la red neural como una función que procesa la entrada ***x***,

f(x)

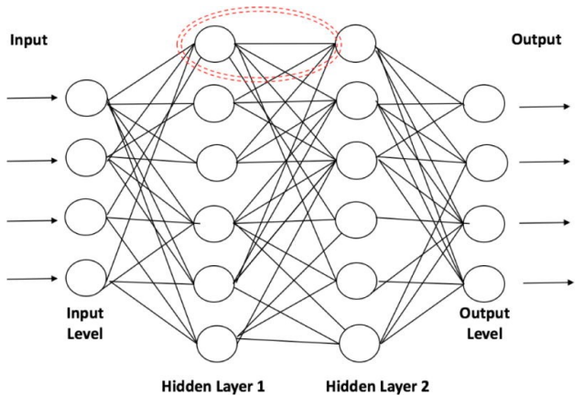
al final arroja una salida ***y***

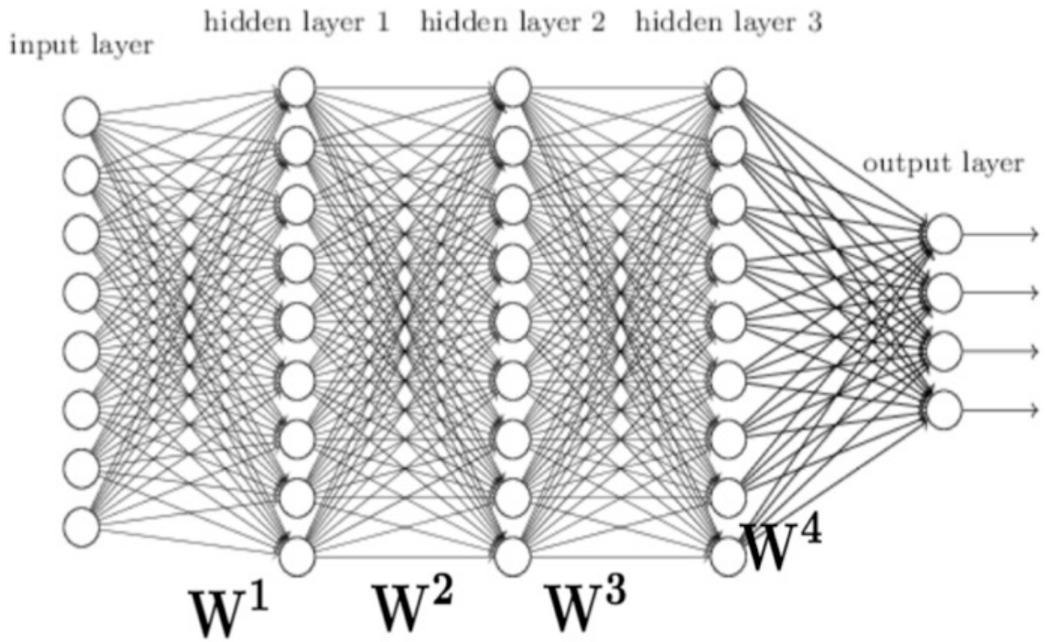
¿Qué es una red neuronal artificial III?



¿Qué es una red neuronal artificial IV?

- Una red neuronal artificial de muchas capas se le conoce **como deep neural network o red neuronal profunda**



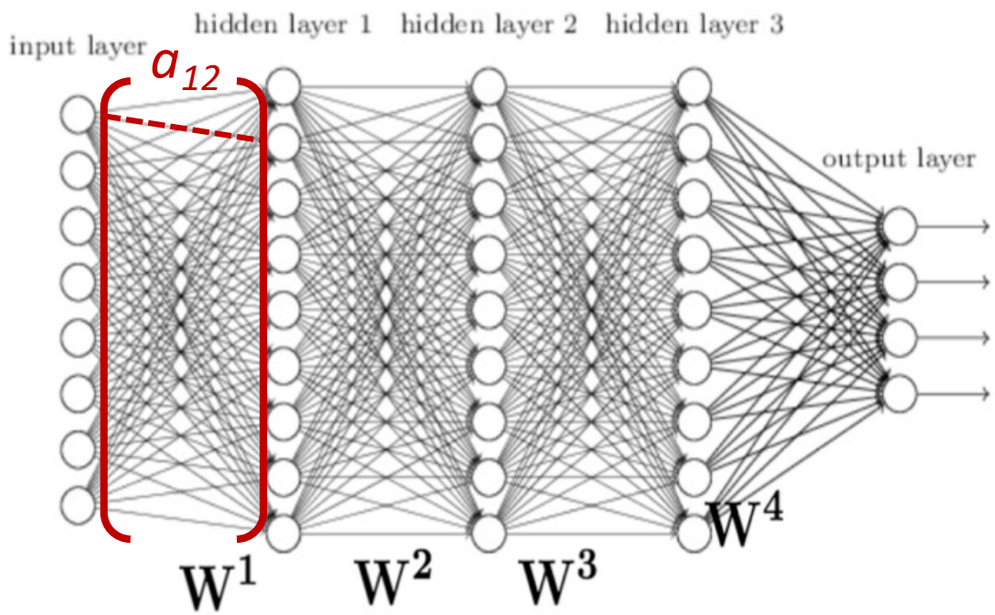


Ann Surg. 2018 Jul; 268(1): 70–76. doi: 10.1097/SLA.0000000000002693

¿Qué es una red neuronal artificial V?

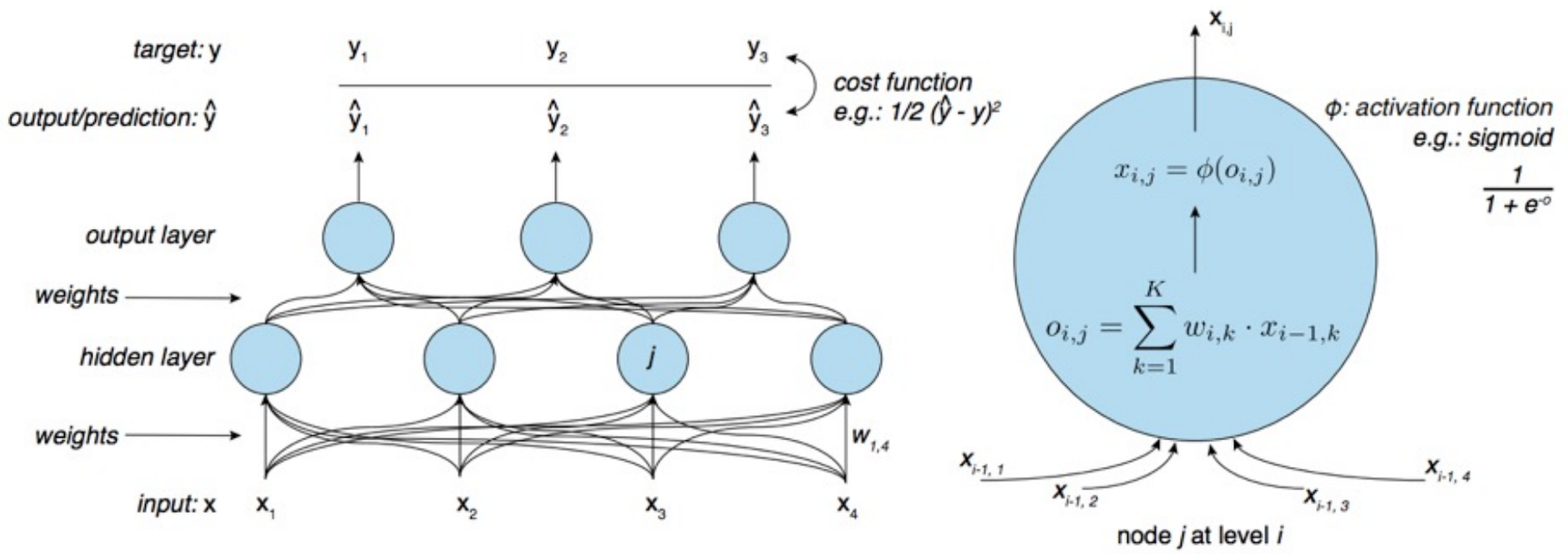
- Para aprender ***f*( )***, **nos interesa aprender los pesos (w_i)** asociados a las **conexiones entre neuronas**
- Las conexiones entre las capas de neuronas se pueden representar con matrices W^i

$$W^1 = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix}$$

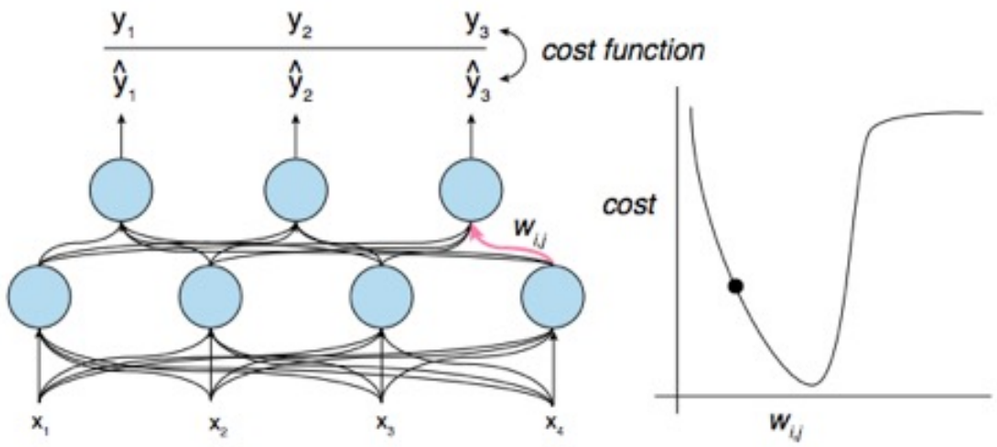


Backpropagation: Ejemplo

x_{ij} : output del nodo *j* de la capa *i*



Backpropagation: Ejemplo



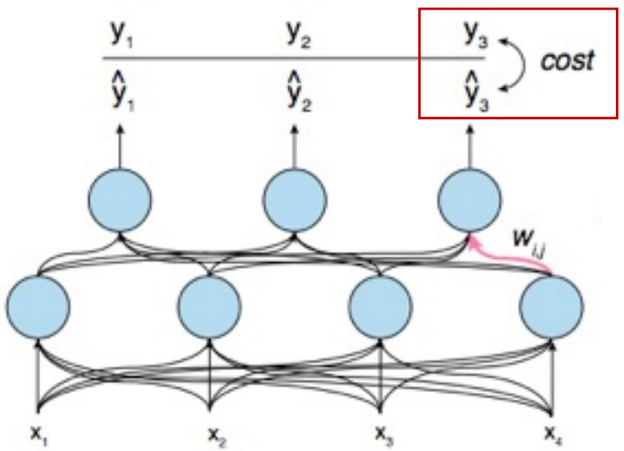
until convergence:
- do a forward pass
- compute the cost/error
- adjust weights ← how??

Adjust every weight $w_{i,j}$ by:

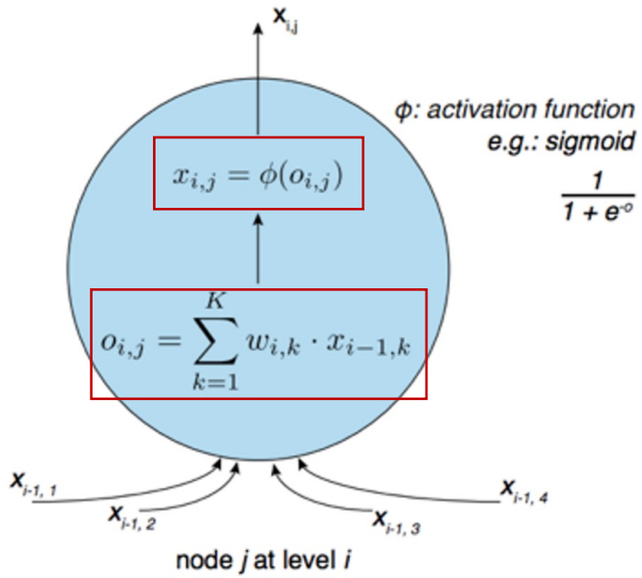
$$\Delta w_{i,j} = -\alpha \frac{\partial cost}{\partial w_{i,j}}$$

α is the learning rate.

Backpropagation: Ejemplo



x_{ij} : output del nodo j de la capa i



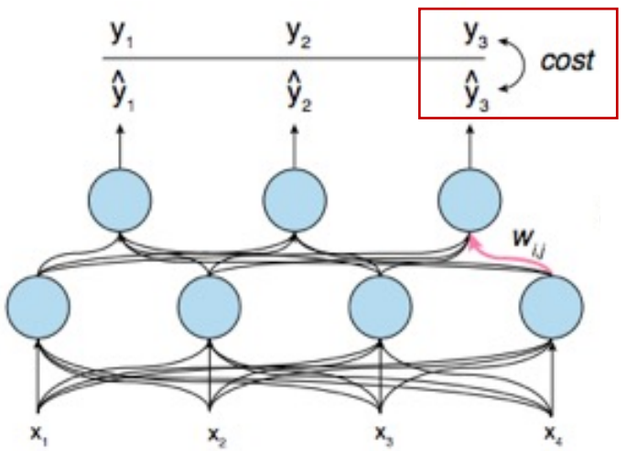
$$\Delta w_{i,j} = -\alpha \frac{\partial cost}{\partial w_{i,j}}$$

$$= -\alpha \frac{\partial cost}{\partial x_{i,j}} \frac{\partial x_{i,j}}{\partial w_{i,j}} \leftarrow \text{chain rule}$$

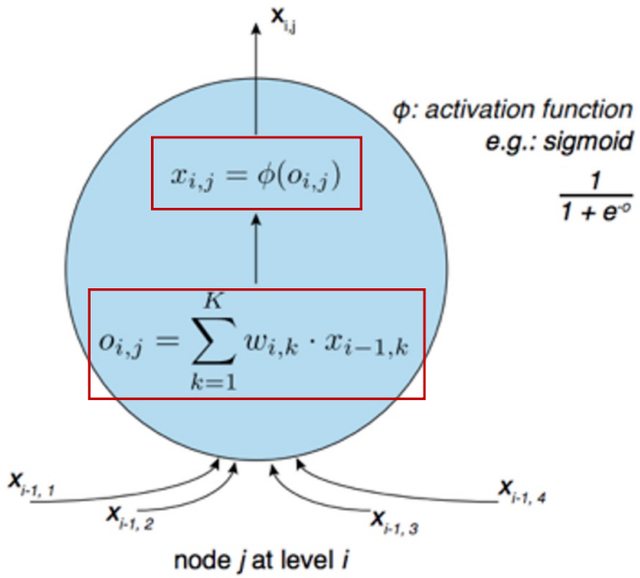
$$cost(\hat{y}, y) = \frac{1}{2}(y - \hat{y})^2$$

$$\hat{y}_j = x_{i,j} = \phi(o_{i,j})$$

Backpropagation: Ejemplo



x_{ij} : output del nodo j de la capa i



$$\Delta w_{i,j} = -\alpha \frac{\partial cost}{\partial w_{i,j}}$$

$$= -\alpha \frac{\partial cost}{\partial x_{i,j}} \frac{\partial x_{i,j}}{\partial w_{i,j}}$$

$$= -\alpha \frac{\partial cost}{\partial x_{i,j}} \frac{\partial x_{i,j}}{\partial o_{i,j}} \frac{\partial o_{i,j}}{\partial w_{i,j}}$$ ← chain rule

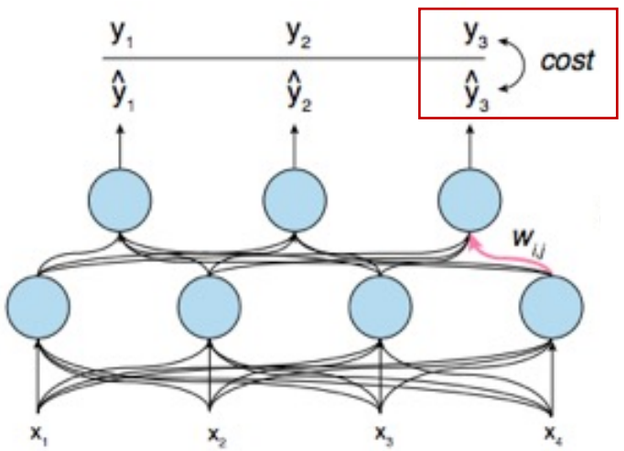
$$cost(\hat{y}, y) = \frac{1}{2}(y - \hat{y})^2$$

$$\hat{y}_j = x_{i,j} = \phi(o_{i,j}), \text{e.g. } \sigma(o_{i,j})$$

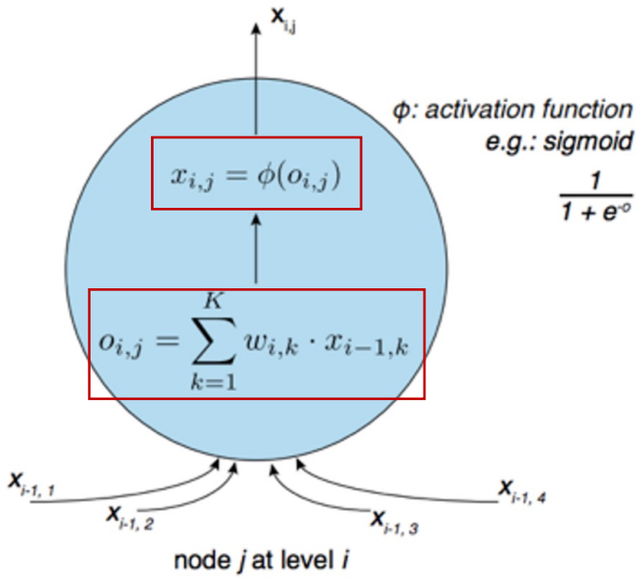
$$x_{i,j} = \sigma(o) = \frac{1}{1 + e^{-o}}$$

$$o_{i,j} = \sum_{k=1}^{K} w_{i,k} \cdot x_{i-1,k}$$

Backpropagation: Ejemplo



x_{ij} : output del nodo j de la capa i



$$\Delta w_{i,j} = -\alpha \frac{\partial cost}{\partial w_{i,j}}$$

$$= -\alpha \frac{\partial cost}{\partial x_{i,j}} \frac{\partial x_{i,j}}{\partial w_{i,j}}$$

$$= -\alpha \frac{\partial cost}{\partial x_{i,j}} \frac{\partial x_{i,j}}{\partial o_{i,j}} \frac{\partial o_{i,j}}{\partial w_{i,j}}$$

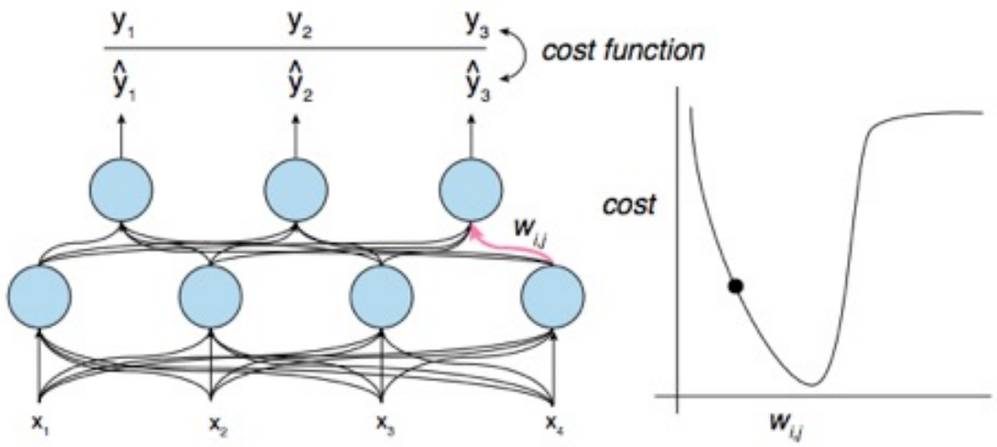
$$cost(\hat{y}, y) = \frac{1}{2}(y - \hat{y})^2$$

$$\hat{y}_j = x_{i,j} = \phi(o_{i,j}), \text{e.g. } \sigma(o_{i,j})$$

$$x_{i,j} = \sigma(o) = \frac{1}{1 + e^{-o}}$$

$$o_{i,j} = \sum_{k=1}^{K} w_{i,k} \cdot x_{i-1,k}$$

Backpropagation: Ejemplo



$$\Delta w_{i,j} = -\alpha \frac{\partial cost}{\partial w_{i,j}}$$

$$= -\alpha \frac{\partial cost}{\partial x_{i,j}} \frac{\partial x_{i,j}}{\partial w_{i,j}}$$

$$= -\alpha \frac{\partial cost}{\partial x_{i,j}} \frac{\partial x_{i,j}}{\partial o_{i,j}} \frac{\partial o_{i,j}}{\partial w_{i,j}}$$

$$= -\alpha \frac{\partial cost}{\partial x_{i,j}} \frac{\partial x_{i,j}}{\partial o_{i,j}} x_{i-1,j}$$

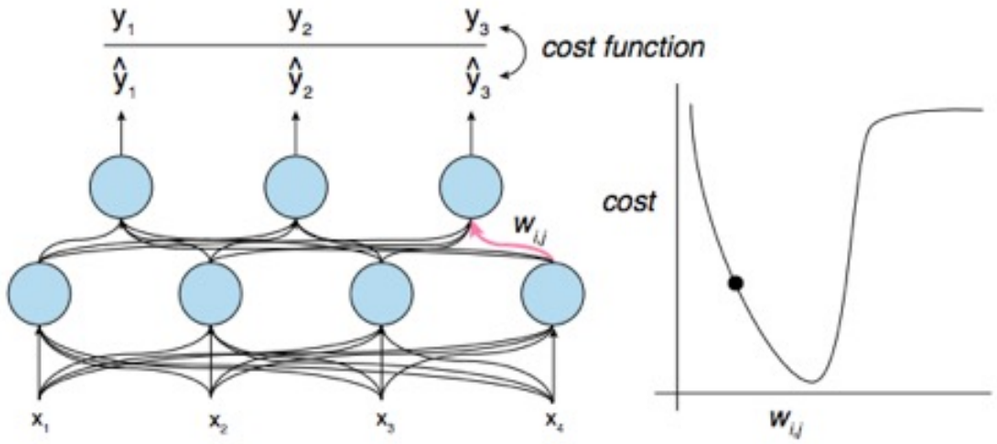
$$cost(\hat{y}, y) = \frac{1}{2}(y - \hat{y})^2$$

$$\hat{y}_j = x_{i,j} = \phi(o_{i,j}), \text{e.g. } \sigma(o_{i,j})$$

$$x_{i,j} = \sigma(o) = \frac{1}{1 + e^{-o}}$$

$$o_{i,j} = \sum_{k=1}^{K} w_{i,k} \cdot x_{i-1,k}$$

Backpropagation: Ejemplo



$$\Delta w_{i,j} = -\alpha \frac{\partial cost}{\partial w_{i,j}}$$
$$= -\alpha \frac{\partial cost}{\partial x_{i,j}} \frac{\partial x_{i,j}}{\partial w_{i,j}}$$
$$= -\alpha \frac{\partial cost}{\partial x_{i,j}} \frac{\partial x_{i,j}}{\partial o_{i,j}} \frac{\partial o_{i,j}}{\partial w_{i,j}}$$
$$= -\alpha \frac{\partial cost}{\partial x_{i,j}} x_{i,j}(1 - x_{i,j})\, x_{i-1,j}$$

$$cost(\hat{y}, y) = \frac{1}{2}(y - \hat{y})^2$$

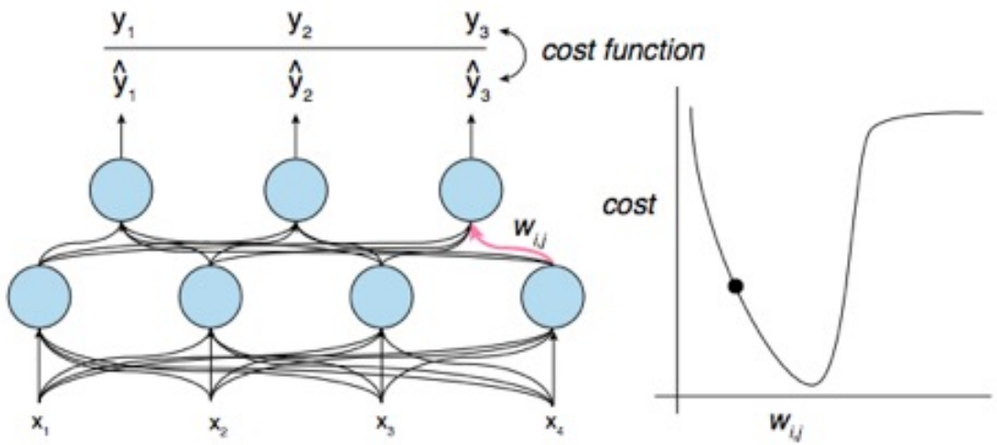
$$\hat{y}_j = x_{i,j} = \phi(o_{i,j}), \text{e.g. } \sigma(o_{i,j})$$

$$x_{i,j} = \sigma(o) = \frac{1}{1 + e^{-o}}$$

$$o_{i,j} = \sum_{k=1}^{K} w_{i,k} \cdot x_{i-1,k}$$

$$\begin{aligned} g'_{\text{logistic}}(z) &= \frac{\partial}{\partial z}\left(\frac{1}{1+e^{-z}}\right) \\ &= \frac{e^{-z}}{(1+e^{-z})^2} (\text{chain rule}) \\ &= \frac{1+e^{-z}-1}{(1+e^{-z})^2} \\ &= \frac{1+e^{-z}}{(1+e^{-z})^2} - \left(\frac{1}{1+e^{-z}}\right)^2 \\ &= \frac{1}{(1+e^{-z})} - \left(\frac{1}{1+e^{-z}}\right)^2 \\ &= g_{\text{logistic}}(z) - g_{\text{logistic}}(z)^2 \\ &= g_{\text{logistic}}(z)(1 - g_{\text{logistic}}(z)) \end{aligned}$$

Backpropagation: Ejemplo



$$\Delta w_{i,j} = -\alpha \frac{\partial cost}{\partial w_{i,j}}$$

$$= -\alpha \frac{\partial cost}{\partial x_{i,j}} \frac{\partial x_{i,j}}{\partial w_{i,j}}$$

$$= -\alpha \frac{\partial cost}{\partial x_{i,j}} \frac{\partial x_{i,j}}{\partial o_{i,j}} \frac{\partial o_{i,j}}{\partial w_{i,j}}$$

$$= -\alpha \boxed{y_j - x_{i,j}}\ x_{i,j}(1 - x_{i,j})\ x_{i-1,j}$$

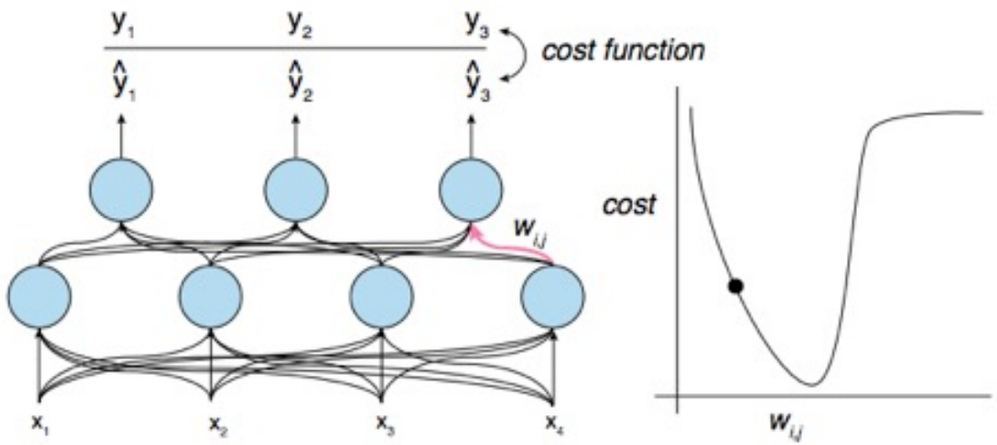
$$cost(\hat{y}, y) = \frac{1}{2}(y - \hat{y})^2$$

$$\hat{y}_j = x_{i,j} = \phi(o_{i,j}), \text{e.g. } \sigma(o_{i,j})$$

$$x_{i,j} = \sigma(o) = \frac{1}{1 + e^{-o}}$$

$$o_{i,j} = \sum_{k=1}^{K} w_{i,k} \cdot x_{i-1,k}$$

Backpropagation: Ejemplo



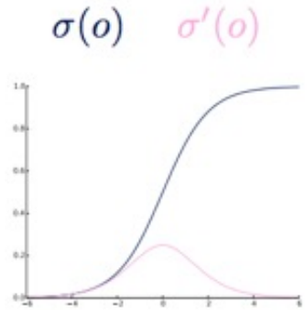
$$cost(\hat{y}, y) = \frac{1}{2}(y - \hat{y})^2$$

$$\hat{y}_j = x_{i,j} = \phi(o_{i,j}), \text{e.g. } \sigma(o_{i,j})$$

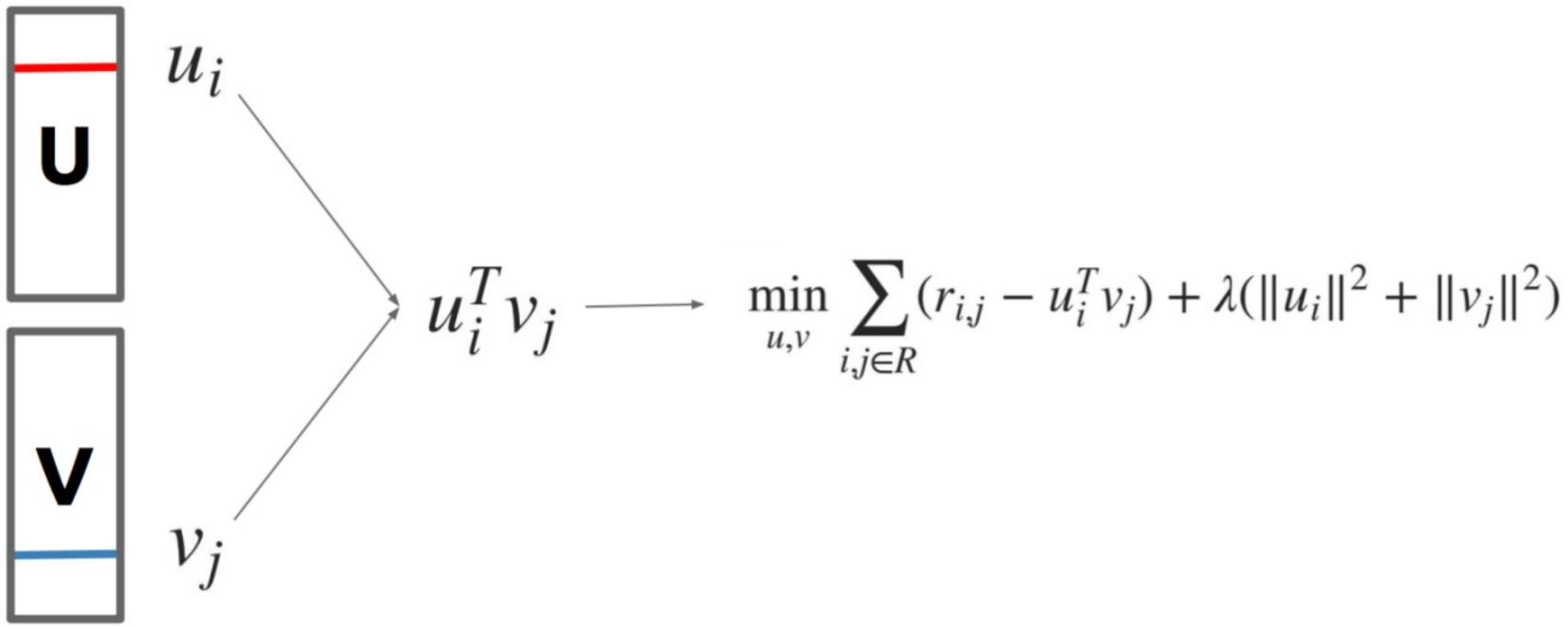
$$x_{i,j} = \sigma(o) = \frac{1}{1 + e^{-o}}$$

$$o_{i,j} = \sum_{k=1}^{K} w_{i,k} \cdot x_{i-1,k}$$

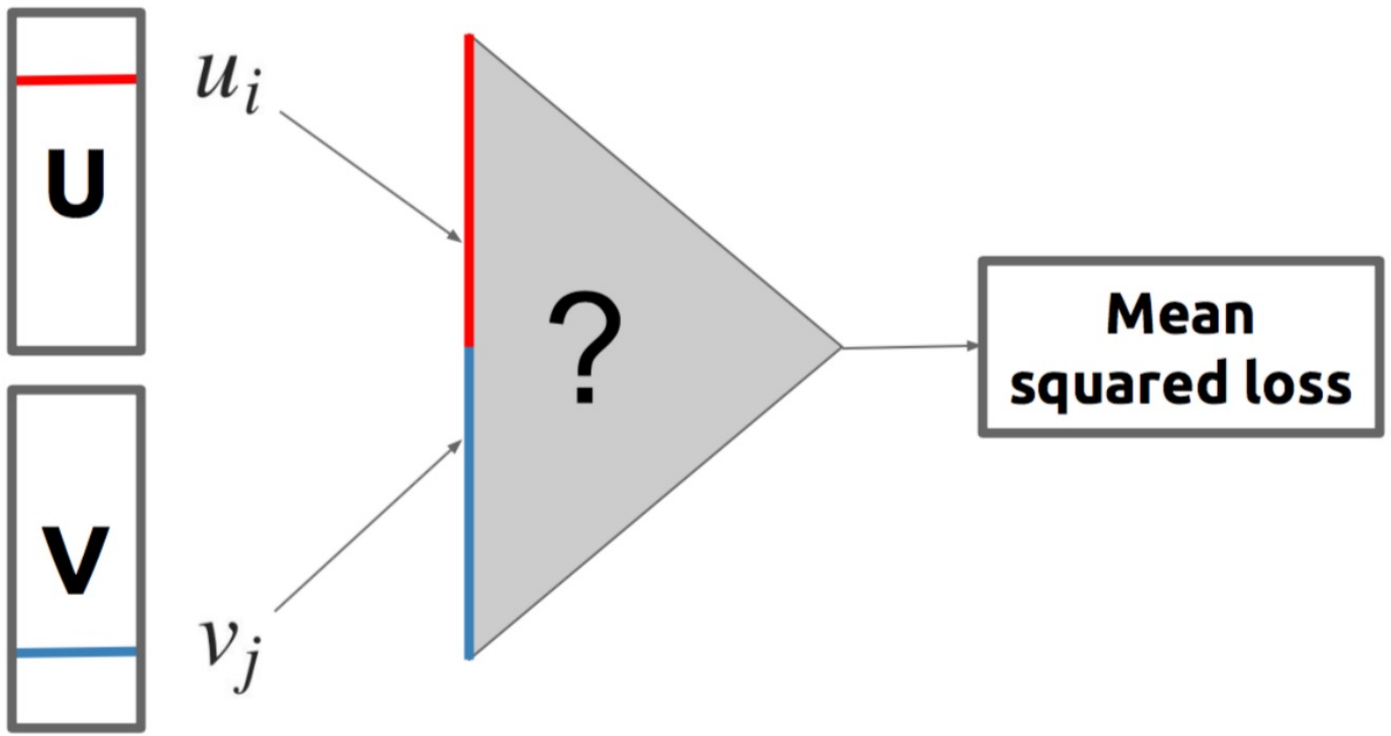
$$\begin{aligned}
\Delta w_{i,j} &= -\alpha \frac{\partial cost}{\partial w_{i,j}} \\
&= -\alpha \frac{\partial cost}{\partial x_{i,j}} \frac{\partial x_{i,j}}{\partial w_{i,j}} \\
&= -\alpha \frac{\partial cost}{\partial x_{i,j}} \frac{\partial x_{i,j}}{\partial o_{i,j}} \frac{\partial o_{i,j}}{\partial w_{i,j}} \\
&= -\alpha \; y_j - x_{i,j} \; x_{i,j}(1 - x_{i,j}) \; x_{i-1,j} \\
&= \text{l.rate} \; cost \; activation \; input
\end{aligned}$$



Factorización Matricial con Redes Neuronales



Factorización Matricial con Redes Neuronales



Factorización Matricial con Redes Neuronales

- Redes neuronales y factorización matricial son similares
  - Uso de embeddings
  - Pérdida mínimos cuadrados
  - Óptimo con descenso de gradiente
- Una red neuronal puede aprender más combinaciones que el producto punto
- Una red neuronal profunda requiere muchos datos para aprender patrones e interacciones

SLIM: Sparse Linear Models for CF

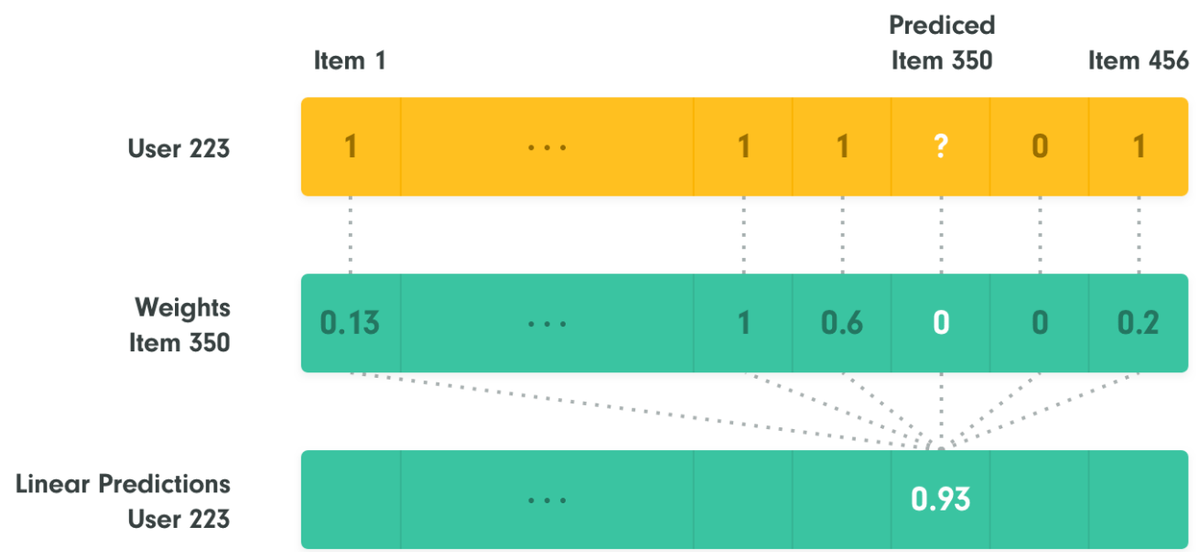
$$\begin{aligned} \min_{W} \quad & \frac{1}{2}\|A - AW\|_F^2 + \frac{\beta}{2}\|W\|_F^2 + \lambda\|W\|_1 \\ \text{subject to} \quad & W \geq 0,\ \text{diag}(W) = 0 \end{aligned}$$

Ning, X., & Karypis, G. (2011, December). Slim: Sparse linear methods for top-n recommender systems. In 2011 IEEE 11th international conference on data mining (pp. 497-506). IEEE.

https://github.com/KarypisLab/SLIM

SLIM: Sparse Linear Models for CF

$$\min_{W} \quad \frac{1}{2}\|A - AW\|_F^2 + \frac{\beta}{2}\|W\|_F^2 + \lambda\|W\|_1$$
$$\text{subject to} \quad W \geq 0,\ \text{diag}(W) = 0$$



Here is an example of how the other items that have been rated by a particular user (223) are summed up weighted by their similarities (or co-occurrence) with predicted item (350).

EASE: Embarrasingly Shallow Autoencoder

3 MODEL TRAINING

We use the following convex objective for learning the weights B:

$$\min_{B} \quad ||X - XB||_F^2 + \lambda \cdot ||B||_F^2 \tag{2}$$

$$\text{s.t.} \quad \text{diag}(B) = 0 \tag{3}$$

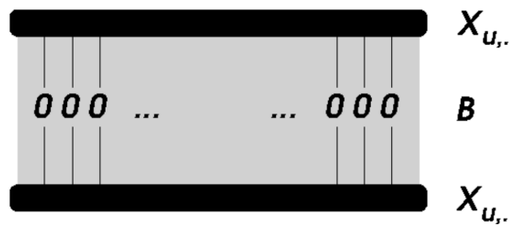


Figure 1: The self-similarity of each item is constrained to zero between the input and output layers.

Steck, H. (2019, May). Embarrassingly shallow autoencoders for sparse data. In *The World Wide Web Conference* (pp. 3251-3257).

EASE: Embarrasingly Shallow Autoencoder

Algorithm 1: Training in Python 2 using numpy

Input: data Gram-matrix $G := X^\top X \in \mathbb{R}^{|\mathcal{I}|\times|\mathcal{I}|}$,
L2-norm regularization-parameter $\lambda \in \mathbb{R}^+$.
Output: weight-matrix B with zero diagonal (see Eq. 8).

```
diagIndices = numpy.diag_indices(G.shape[0])
G[diagIndices] += λ
P = numpy.linalg.inv(G)
B = P / (-numpy.diag(P))
B[diagIndices] = 0
```

- Notar que al eliminar la restricción de no-negatividad, el modelo obtiene mejores resultados

https://github.com/Darel13712/ease_rec

(a) *ML-20M*	Recall@20	Recall@50	NDCG@100
popularity	0.162	0.235	0.191
EASE^R	0.391	0.521	0.420
EASE$^R\geq 0$	0.373	0.499	0.402
results reproduced from [13]:			
SLIM	0.370	0.495	0.401
WMF	0.360	0.498	0.386
CDAE	0.391	0.523	0.418
MULT-VAE^{PR}	0.395	0.537	0.426
MULT-DAE	0.387	0.524	0.419

(c) *MSD*			
popularity	0.043	0.068	0.058
EASE^R	0.333	0.428	0.389
EASE$^R\geq 0$	0.324	0.418	0.379
results reproduced from [13]:			
SLIM	— did not finish in [13] —		
WMF	0.211	0.312	0.257
CDAE	0.188	0.283	0.237
MULT-VAE^{PR}	0.266	0.364	0.316
MULT-DAE	0.266	0.363	0.313

SANSA: EASE on million item datasets

Official implementation of scalable collaborative filtering model **SANSA**.

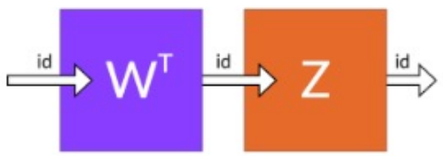


Fig. 1. SANSA is a sparse nonsymmetric encoder-decoder model. To disallow recommending input items, we mask the prediction vector, or add an input-output residual connection.

input user-item interaction matrix X, L2 regularization λ
1: compute sparse $LDL^T \approx P(X^TX + \lambda I)P^T$ (for some permutation P)
2: compute sparse $K \approx L^{-1}$
3: $W \leftarrow KP$
4: $Z_0 \leftarrow D^{-1}W$
5: $\vec{r} \leftarrow \text{diag}(W^TZ_0)$
6: $Z \leftarrow$ scale the columns of Z_0 by $-\vec{r}$
return W^T, Z

Algorithm 1. The training procedure of SANSA is based on factorized sparse approximate inversion. The final scaling is applied to the decoder only.

> **Scalable Approximate NonSymmetric Autoencoder for Collaborative Filtering**
> Spišák M., Bartyzal R., Hoskovec A., Peška L., Tůma M.
> Paper: 10.1145/3604915.3608827
>
> *Best Short Paper Runner-Up*, 17th ACM Conference on Recommender Systems (ACM RecSys 2023)

SANSA is a scalable modification of EASE, a shallow autoencoder for collaborative filtering, **specifically designed to handle item sets with millions of items.**

- End-to-end sparse training procedure: instead of strenuously inverting the Gramian X^TX of user-item interaction matrix X, SANSA efficiently finds a *sparse approximate inverse* of X^TX.
- Training memory requirements are proportional to the number of non-zero elements in X^TX (and this can be improved further).
- The model's density is prescribed via a hyperparameter.
- As a sparse neural network, SANSA offers *very fast inference* times.

https://github.com/glami/sansa

NCF: Neural Collaborative Filtering

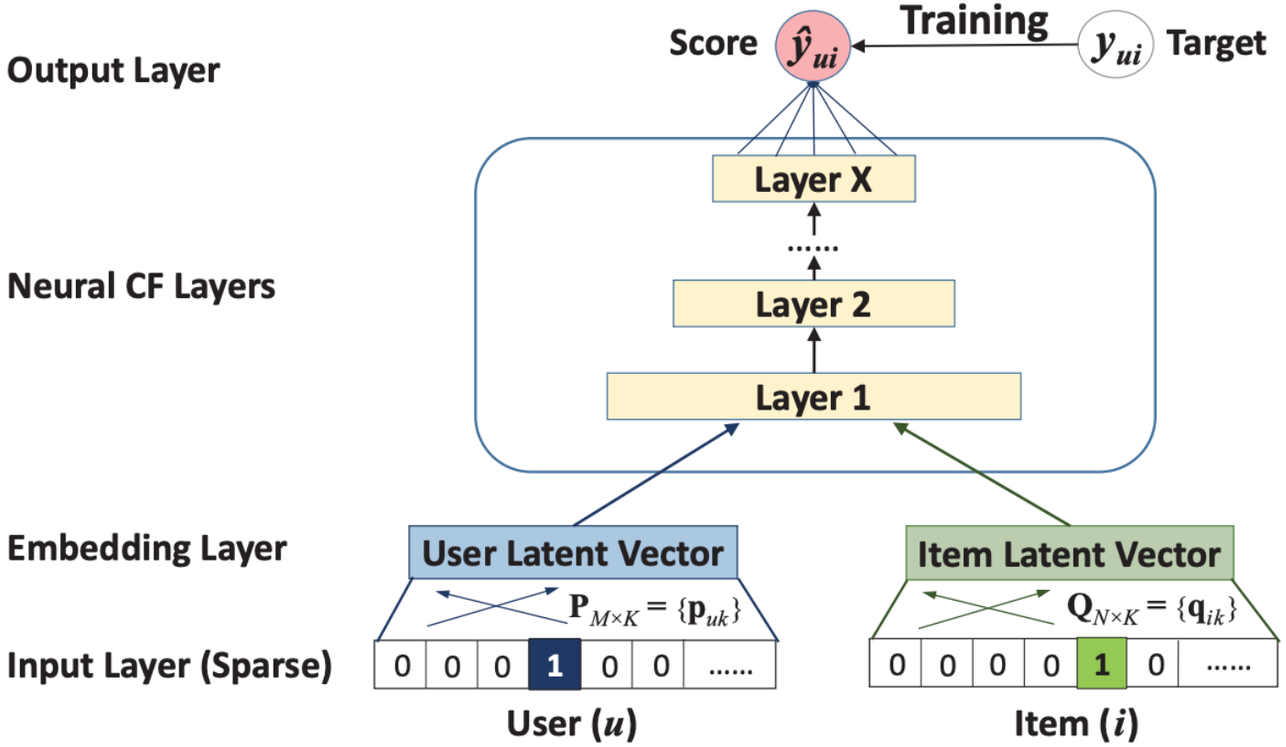


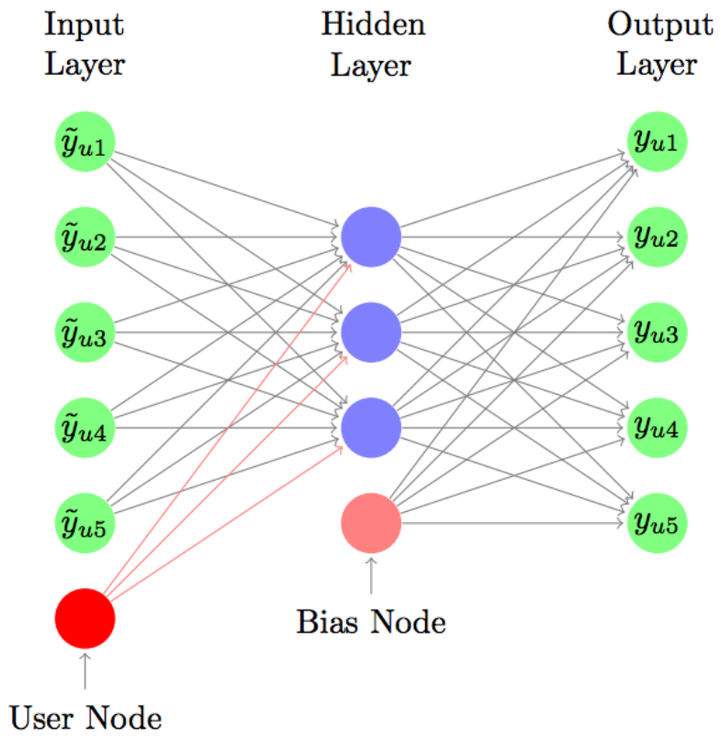
Figure 2: Neural collaborative filtering framework

He, X., Liao, L., Zhang, H., Nie, L., Hu, X., & Chua, T. S. (2017, April). Neural collaborative filtering. In Proceedings of the 26th international conference on world wide web (pp. 173-182)

Recomendación Basada en Autoencoders

Collaborative Denoising Auto-Encoders for Top-N Recommender Systems [Wu et al., 2016]

- Idea: reconstruir un vector del usuario objetivo
- Pesos de color rojo son propios del usuario
- El resto de los nodos permanecen constante



Recomendación Basada en Autoencoders

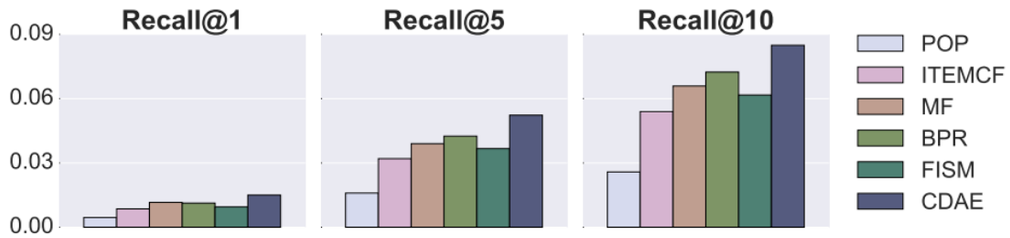


Figure 9: The Recall scores with different N on the Yelp data set.

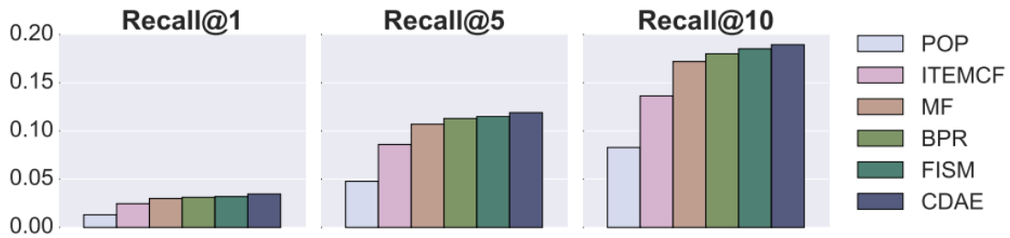


Figure 10: The Recall scores with different N on the MovieLens data set.

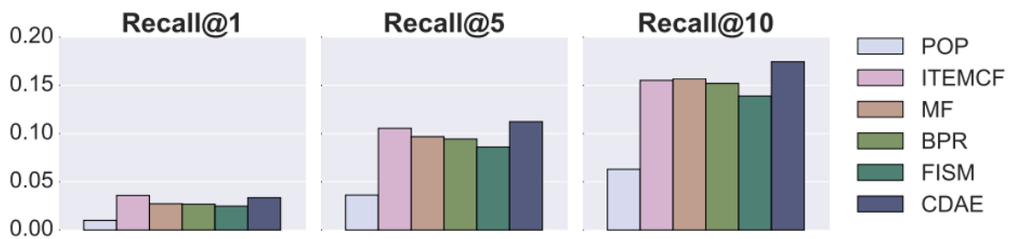


Figure 11: The Recall scores with different N on the Netflix data set.

MultVAE

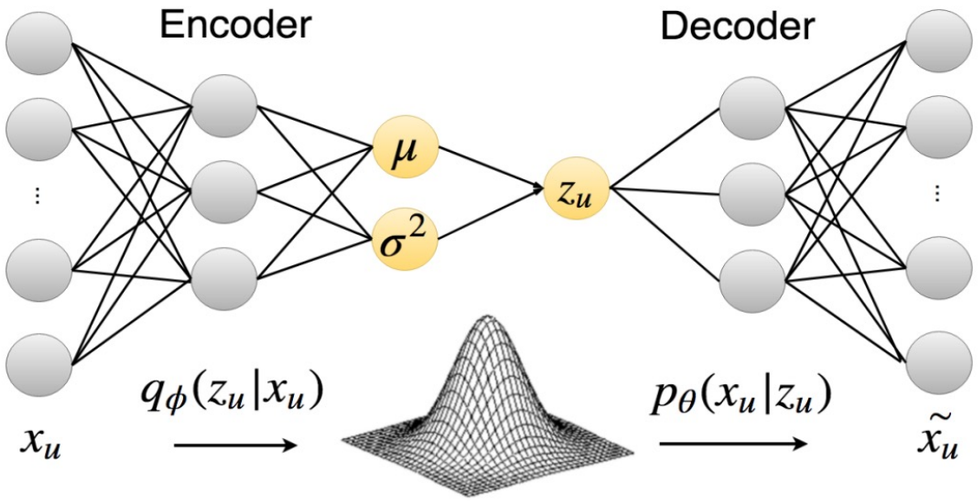


Figure 1: Variational Autoencoders for collaborative filtering. For each user: (1) a bag-of-items vector x_u is provided as input to the encoder; (2) a latent user vector z_u is sampled from a Gaussian distribution with parameters specified by the encoder; (3) a new bag-of-items vector $\tilde{x_u}$ is reconstructed via the decoder.

Liang, D., Krishnan, R. G., Hoffman, M. D., & Jebara, T. (2018). Variational autoencoders for collaborative filtering. In Proceedings of the 2018 world wide web conference (pp. 689-698).

Combinación de dos Modelos

Wide & Deep Learning for Recommender Systems [Cheng et al., 2016]

- Modelo Profundo
  - Utiliza embeddings
  - Se encarga de generalizar el modelo
- Modelo Lineal
  - Utiliza embeddings
  - Utiliza características de los productos
  - “Men

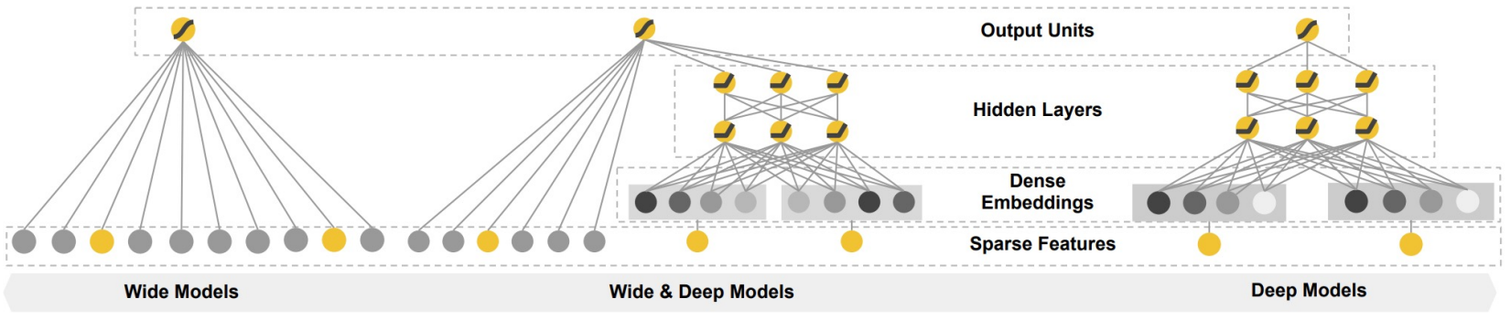


Figure 1: The spectrum of Wide & Deep models.

DeepFM

DeepFM: A Factorization-Machine based Neural Network for CTR Prediction

Huifeng Guo[*1] **, Ruiming Tang**[2]**, Yunming Ye**[†1]**, Zhenguo Li**[2]**, Xiuqiang He**[2]
[1]Shenzhen Graduate School, Harbin Institute of Technology, China
[2]Noah's Ark Research Lab, Huawei, China
[1]huifengguo@yeah.net, yeyunming@hit.edu.cn
[2]{tangruiming, li.zhenguo, hexiuqiang}@huawei.com

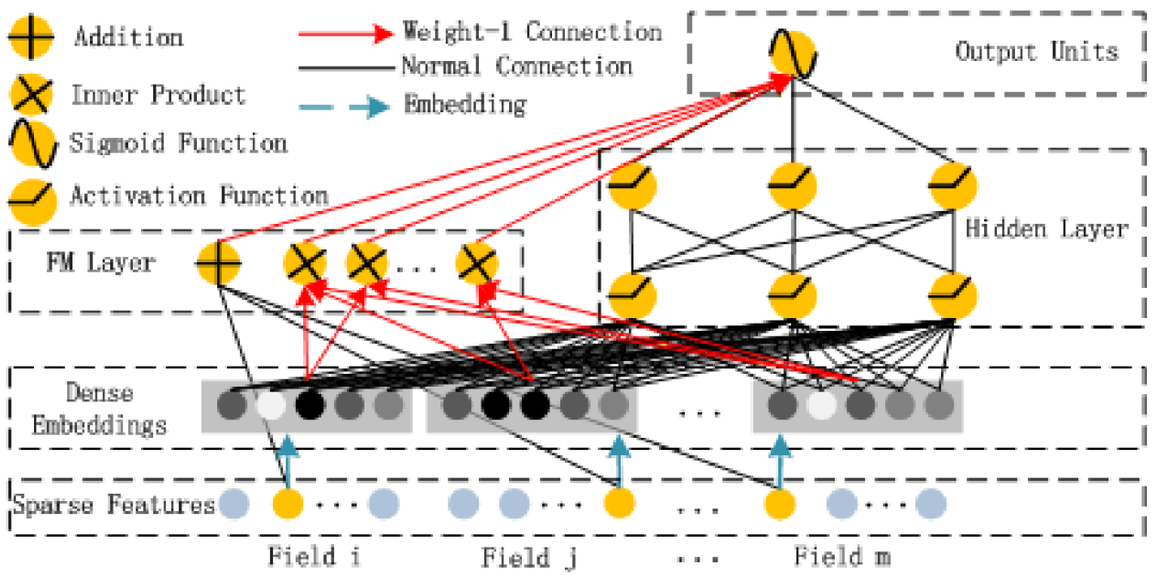


Figure 1: Wide & deep architecture of DeepFM. The wide and deep component share the same input raw feature vector, which enables DeepFM to learn low- and high-order feature interactions simultaneously from the input raw features.

Combinación de dos Modelos

- Arquitectura del modelo implementado en Google Play Store

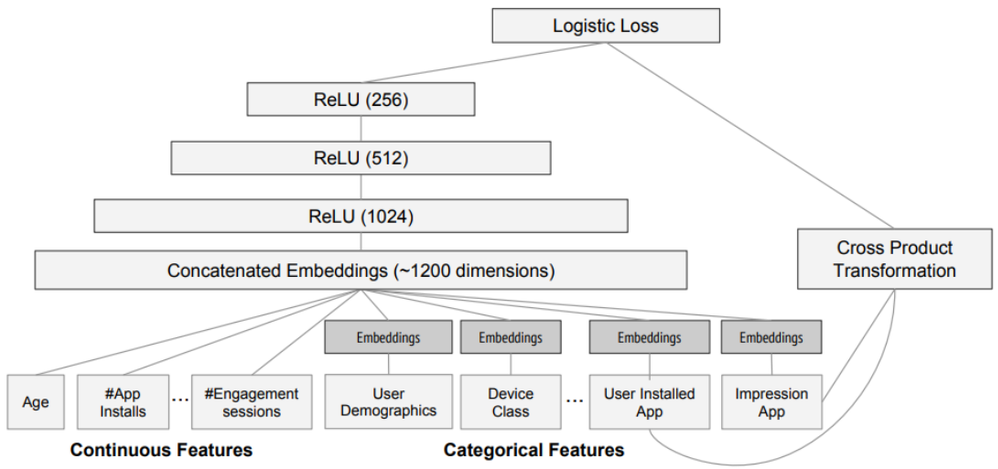


Figure 4: Wide & Deep model structure for apps recommendation.

Table 1: Offline & online metrics of different models. Online Acquisition Gain is relative to the control.

Model	Offline AUC	Online Acquisition Gain
Wide (control)	0.726	0%
Deep	0.722	+2.9%
Wide & Deep	0.728	+3.9%

Redes Neuronales para Factorización Matricial

- https://arxiv.org/pdf/1907.06902.pdf

Are We Really Making Much Progress? A Worrying Analysis of Recent Neural Recommendation Approaches

Maurizio Ferrari Dacrema
Politecnico di Milano, Italy
maurizio.ferrari@polimi.it

Paolo Cremonesi
Politecnico di Milano, Italy
paolo.cremonesi@polimi.it

Dietmar Jannach
University of Klagenfurt, Austria
dietmar.jannach@aau.at

- Si solo usamos datos de interacciones, puede que las redes neuronales no nos ofrezcan tanto beneficio.

Redes Neuronales para Factorización Matricial

- https://arxiv.org/pdf/1907.06902.pdf

Are We Really Making Much Progress? A Worrying Analysis of Recent Neural Recommendation Approaches

Maurizio Ferrari Dacrema
Politecnico di Milano, Italy
maurizio.ferrari@polimi.it

Paolo Cremonesi
Politecnico di Milano, Italy
paolo.cremonesi@polimi.it

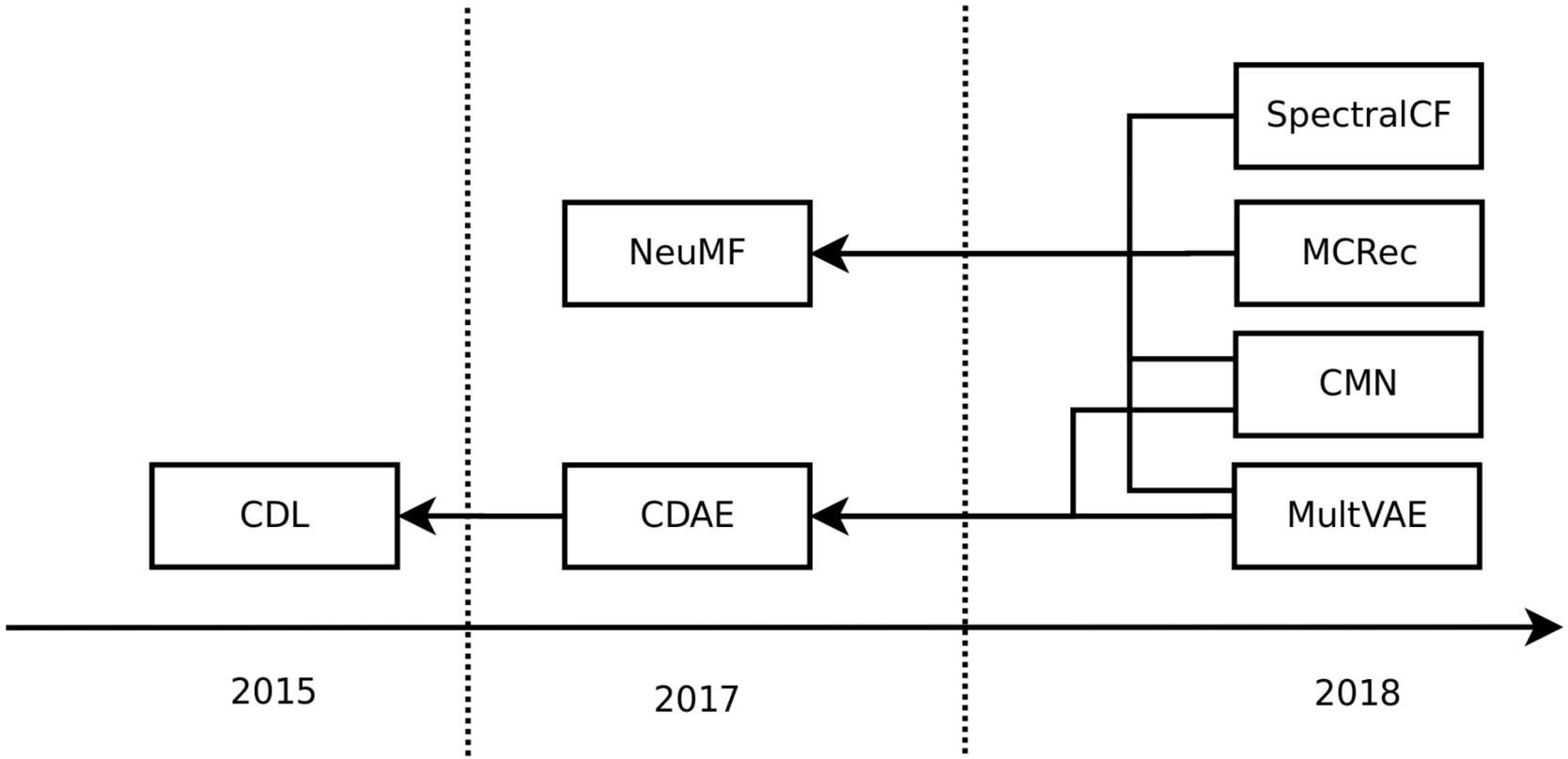
Dietmar Jannach
University of Klagenfurt, Austria
dietmar.jannach@aau.at

- Si solo usamos datos de interacciones, puede que las redes neuronales no nos ofrezcan tanto beneficio.

Redes Neuronales para Factorización Matricial

SIGIR ‘18	CMN	Collaborative memory networks
KDD ’18	MCRec	Metapath based context for rec.
KDD ’17	CVAE	Collaborative variational autoencoder
KDD ’15	CDL	Collaborative deep learning
WWW ’17	NeuMF	Neural collaborative filtering
WWW ’18	Mult-VAE	Variational autoencoder for CF
RecSys’18	SpectralCF	Spectral collaborative filtering

Redes Neuronales para Factorización Matricial



Redes Neuronales para Factorización Matricial

Algorithm	CF + CBF	CF + CBF + NP	CF + CBF + NP + SLIM
MCRec	-	-	-
SpectralCF	-	-	-
CMN	4/12 - 30%	-	-
NeuMF	6/12 - 50%	6/12 - 50%	-
CDL	9/24 - 37%	9/24 - 37%	9/24 - 37%
CVAE	9/24 - 37%	9/24 - 37%	9/24 - 37%
Mult-VAE	12/12 - 100%	12/12 - 100%	10/12 - 83%

Factorización Matricial con Redes Neuronales

- Redes neuronales y factorización matricial son similares
  - Uso de embeddings
  - Pérdida mínimos cuadrados
  - Óptimo con descenso de gradiente
- Una red neuronal puede aprender más combinaciones que el producto punto
- Una red neuronal requiere muchos datos para aprender patrones e interacciones
- **El uso de redes neuronales no es siempre mejor que una buena factorización matricial… por ahora…**

Las redes neuronales permiten capturar más información que una factorización matricial como información de texto, imágenes, música, etc.

Ventajas de las Redes Neuronales

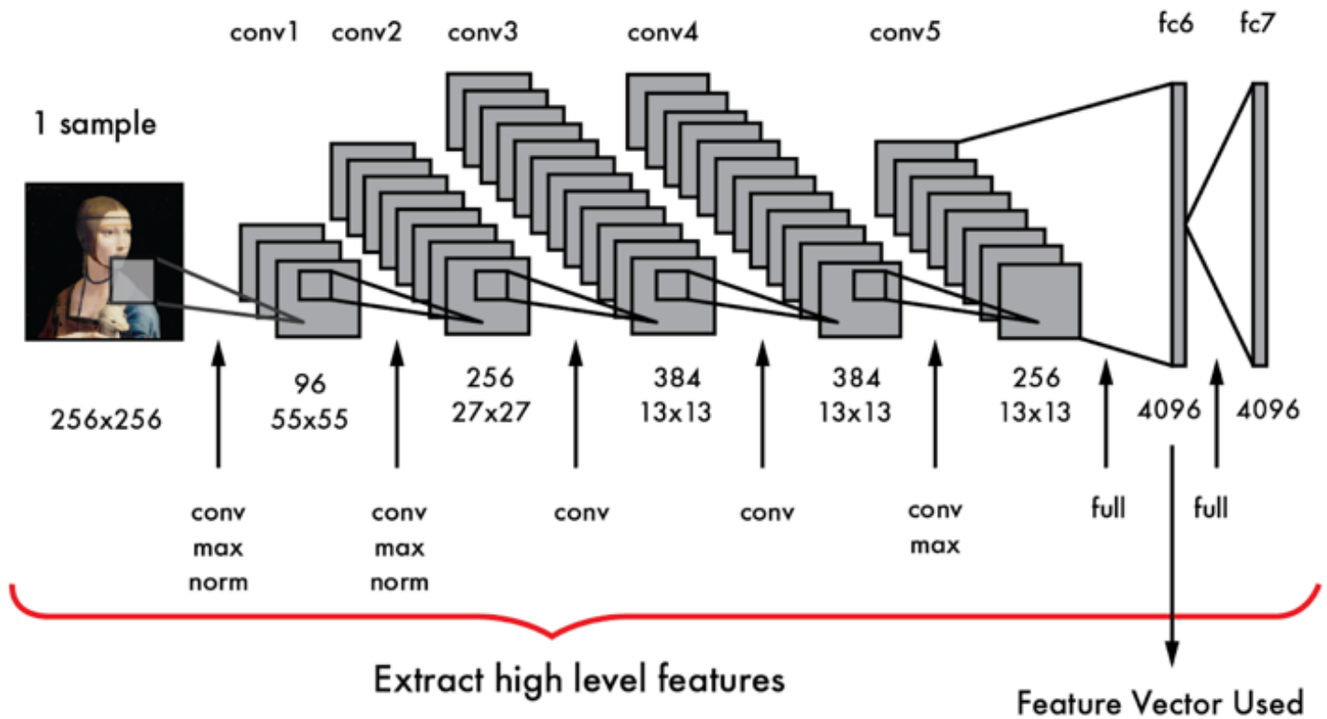
- Información de texto
  - Ejemplo: descripción de productos, comentarios de usuarios
  - Extracción: RNN, Transformer
  - Aplicaciones: noticias, libros, e-commerce
- Imágenes
  - Ejemplo: foto de productos, thumbnail de videos
  - Extracción: CNN, Visual Transformer
  - Aplicaciones: moda, video
- Música / audio
  - Ejemplo: spotify
  - Extracción: CNN, RNN y Transformer
  - Aplicación: música

Recomendación visual basada en contenido

- En los modelos tradicionales la extracción de características a partir de imágenes se realiza por diferentes técnicas. Algunas de ellas:
  - Local Binary patterns (LBP): Método “manual” usado tradicionalmente como referencia de comparación en tareas de Visión por Computador. Obtiene un histograma de 59 patrones encontrados en una imagen.
  - Attractiveness : serie de 7 métricas

Features manuales versus Deep Learning

- Con DL podemos usar features aprendidas automáticamente con una red neuronal pre-entrenada para otra tarea: clasificación de objetos del dataset Imagenet.



Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information pro*

VisRank team

Vicente Domínguez

Pablo Messina

Ivania Donoso-Guzmán

Christoph Trattner

Alvaro Soto

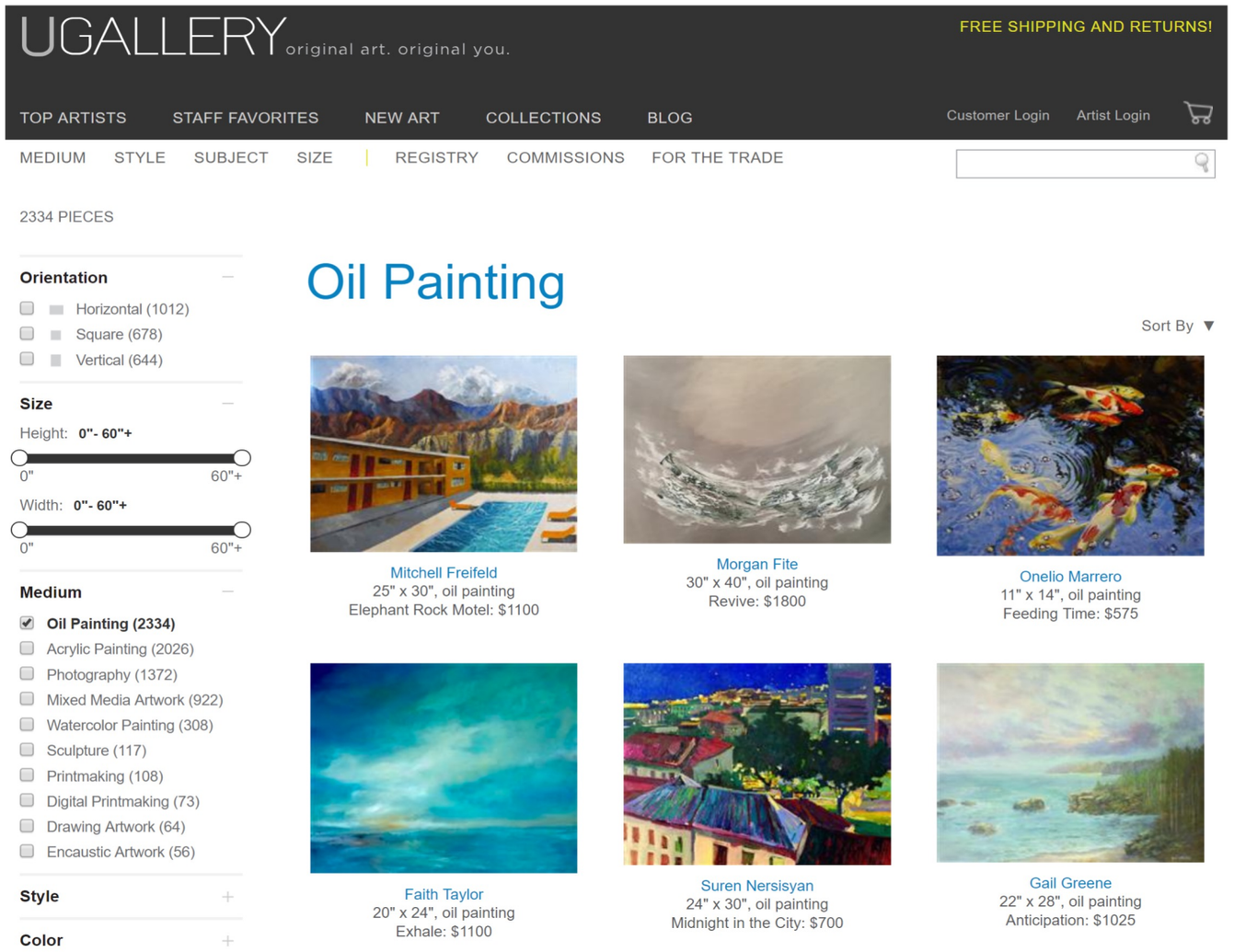
Manuel Cartagena

Denis Parra

Domingo Mery

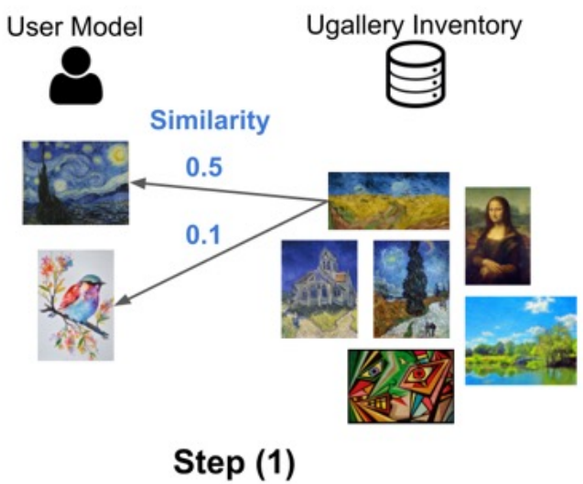
Context: Ugallery

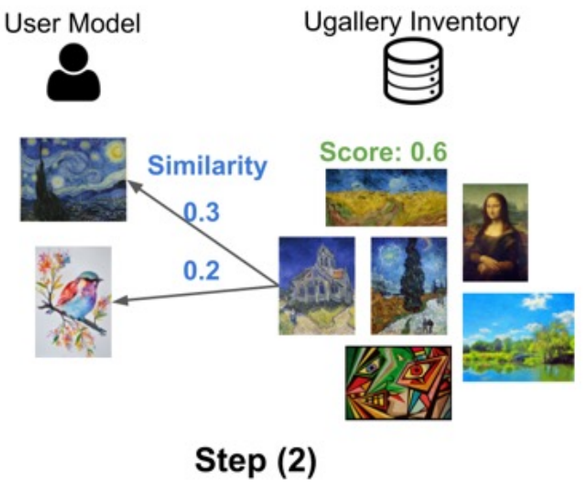
- Online art store
- Focuses specially in emergent artists: to help them sell their paintings
- Our partners in this project

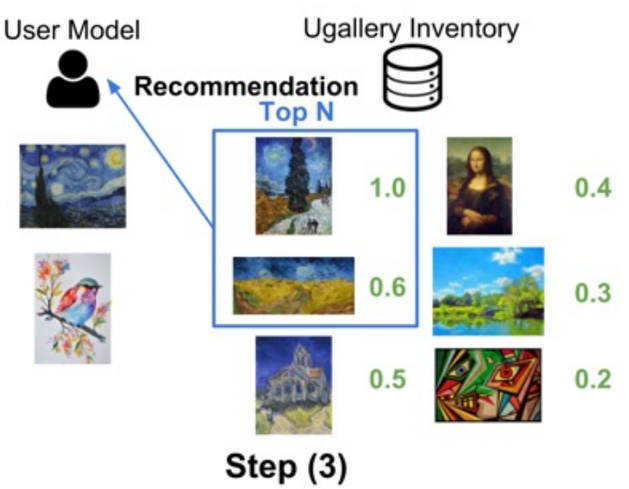


Content-based Recommendation

- User Transactions on mostly one-of-a-kind paintings, difficult to do collaborative filtering







Transfer Learning and fine-tuning

- del Rio, F., Messina, P., Dominguez, V., & Parra, D. (2018). Do Better ImageNet Models Transfer Better... for Image Recommendation?. *arXiv preprint arXiv:1807.09870*.
- ResNet was the best for transfer learning (with and w/o finetuning)

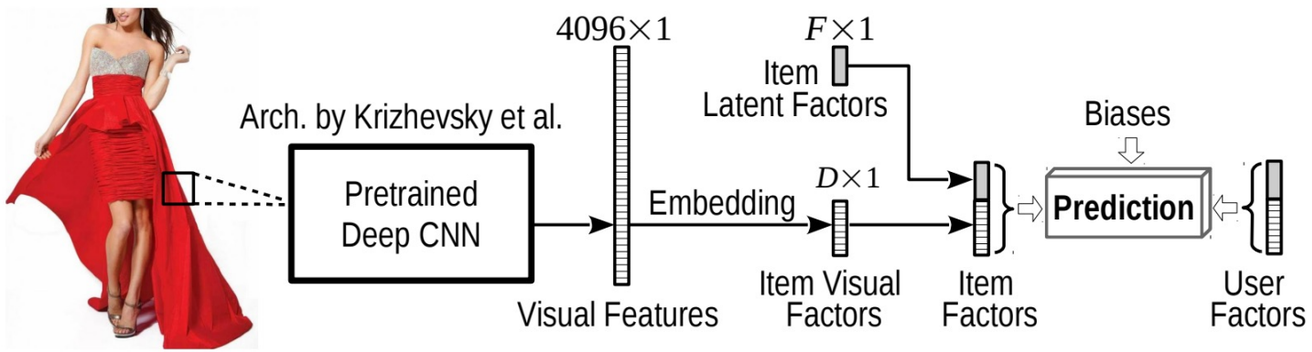
CNN	Artwork Image Recommendation				ILSVRC-2012-CLS	
	R@20	**P@20**	**MRR@20**	**nDCG@20**	**Top-1 Acc. (%)**	**Top-5 Acc. (%)**
ResNet50	**.1632**	**.0141**	**.0979**	**.1253**	75.2	92.2
VGG19	**.1398**	**.0124**	**.0750**	**.1008**	71.1	89.8
NASNet Large	.1379	.0120	.0743	.0998	**82.7**	**96.2**
InceptionV3	.1332	.0125	.0744	.1007	78.0	93.9
InceptionResNetV2	.1302	.0117	.0692	.0936	**80.4**	**95.3**
Random	.0172	.0013	.0051	.0093	-	-

Table 1: Results of different pre-trained embeddings at the artwork image recommendation task to the left (R:Recall, P:Precision), and their performance at the ILSVRC Challenge trained on ImageNet dataset (Acc: Accuracy). The top methods in both tasks do not correlate.

Imágenes en recomendaciones

- VBPR: Visual Bayesian Personalized Ranking from Implicit Feedback [He and McAuley, 2016]
- Utiliza CNN para obtener características visuals
- Utilizan embeddings de usuarios e ítems
- Pérdiad BPR

$$x_{u,i} = \beta_i + \gamma_u^T \gamma_i + \theta_u^T (E \cdot f_i)$$

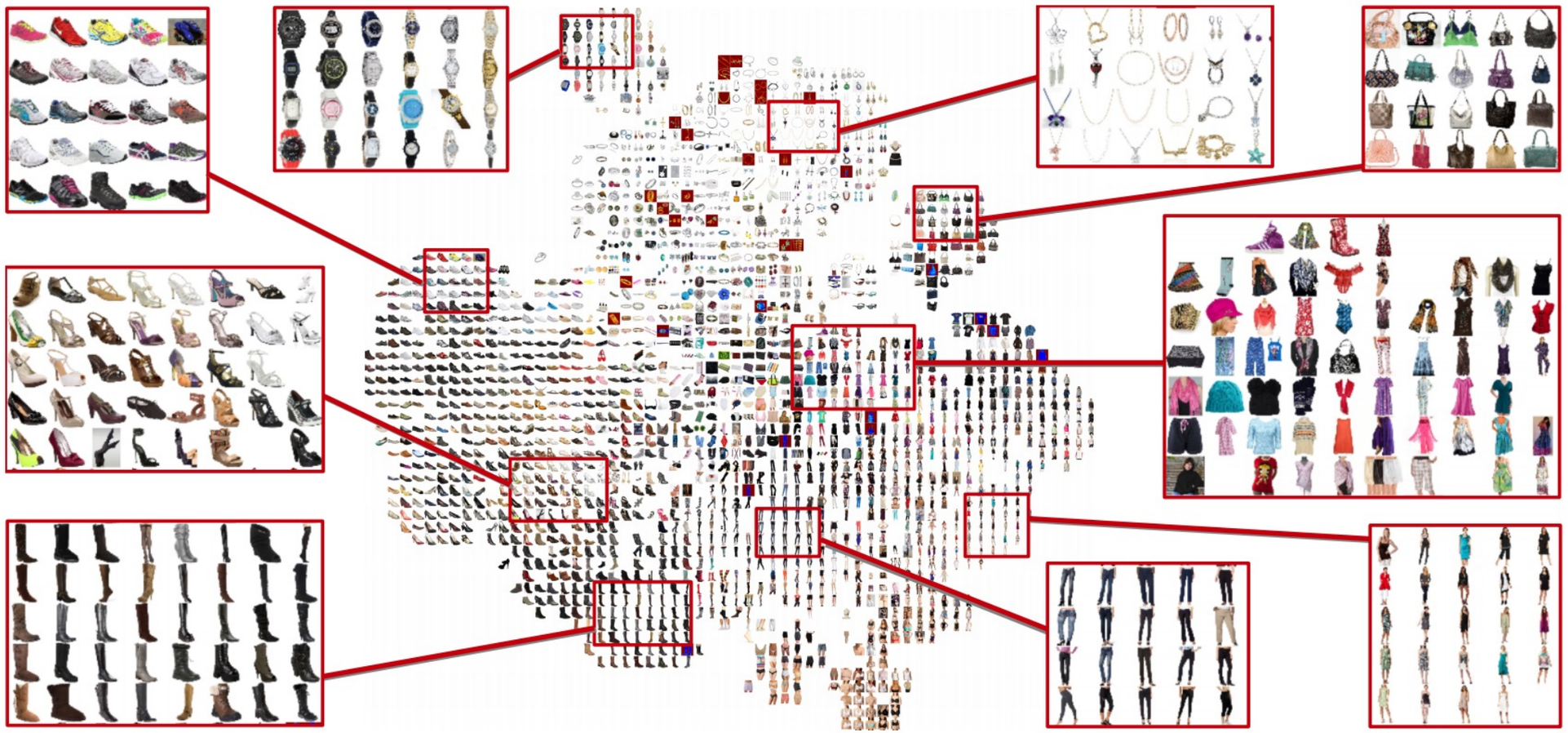


Imágenes en Recomendaciones

Table 3: AUC on the test set $\mathcal{T}$ (#factors = 20). The best performing method on each dataset is boldfaced.

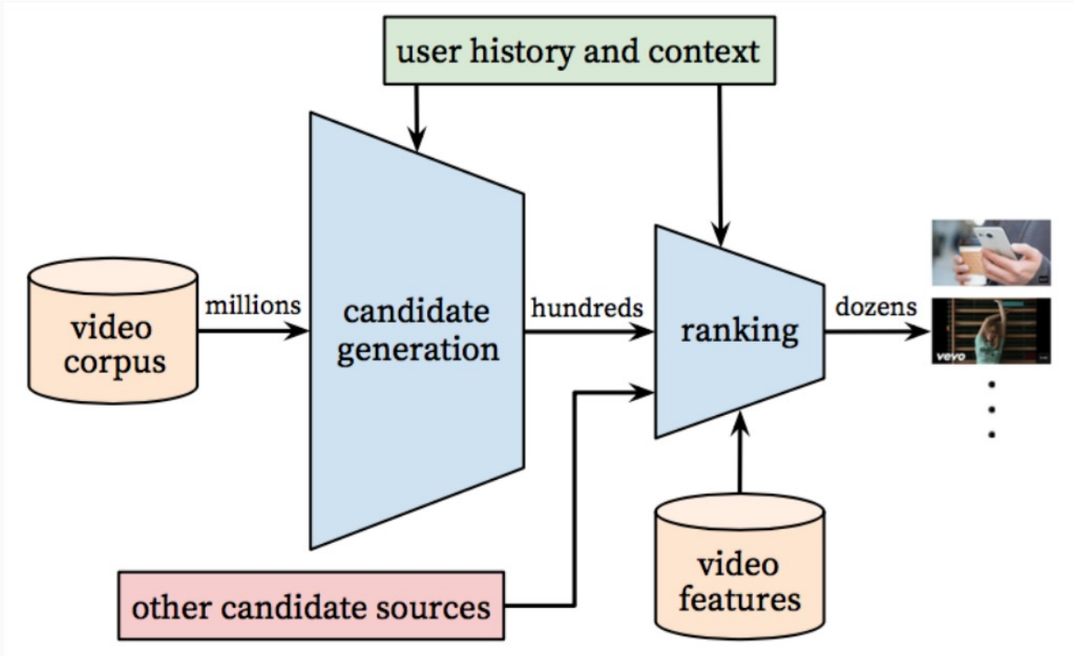
Dataset	Setting	(a) RAND	(b) MP	(c) IBR	(d) MM-MF	(e) BPR-MF	(f) VBPR	improvement f vs. best	improvement f vs. e
Amazon Women	All Items	0.4997	0.5772	0.7163	0.7127	0.7020	**0.7834**	9.4%	11.6%
	Cold Start	0.5031	0.3159	0.6673	0.5489	0.5281	**0.6813**	2.1%	29.0%
Amazon Men	All Items	0.4992	0.5726	0.7185	0.7179	0.7100	**0.7841**	9.1%	10.4%
	Cold Start	0.4986	0.3214	0.6787	0.5666	0.5512	**0.6898**	1.6%	25.1%
Amazon Phones	All Items	0.5063	0.7163	0.7397	0.7956	0.7918	**0.8052**	1.2%	1.7%
	Cold Start	0.5014	0.3393	**0.6319**	0.5570	0.5346	0.6056	-4.2%	13.3%
Tradesy.com	All Items	0.5003	0.5085	N/A	0.6097	0.6198	**0.7829**	26.3%	26.3%
	Cold Start	0.4972	0.3721	N/A	0.5172	0.5241	**0.7594**	44.9%	44.9%

Imágenes en Recomendaciones



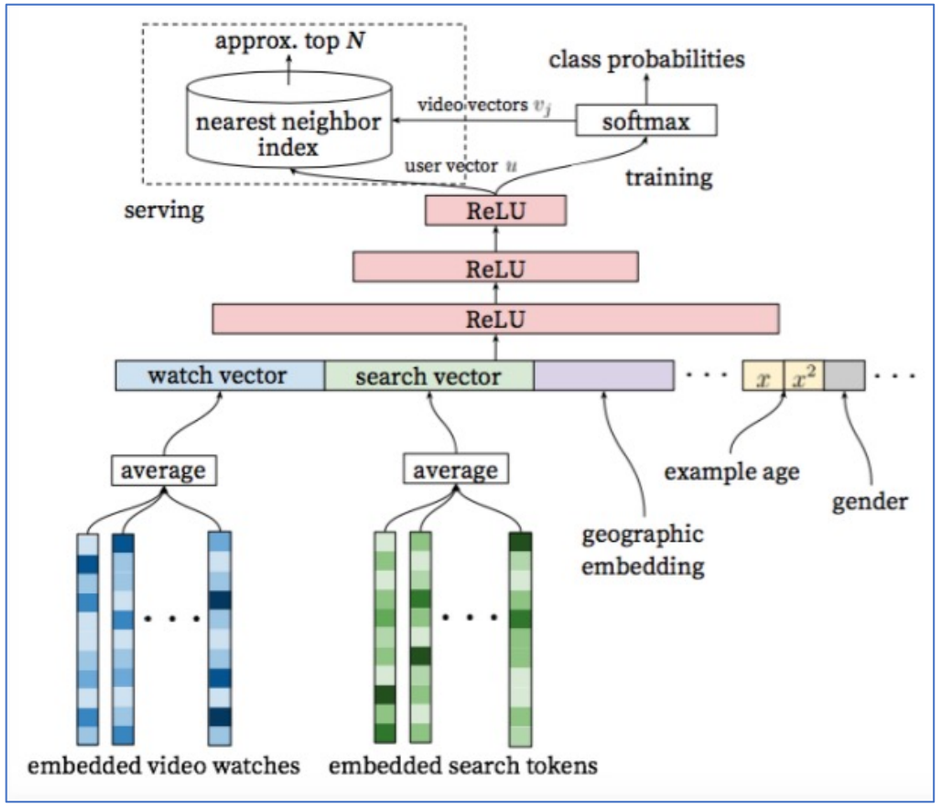
(Un) Sistema Recomendador de YouTube

- El Sistema Recomendador de YouTube (2016)

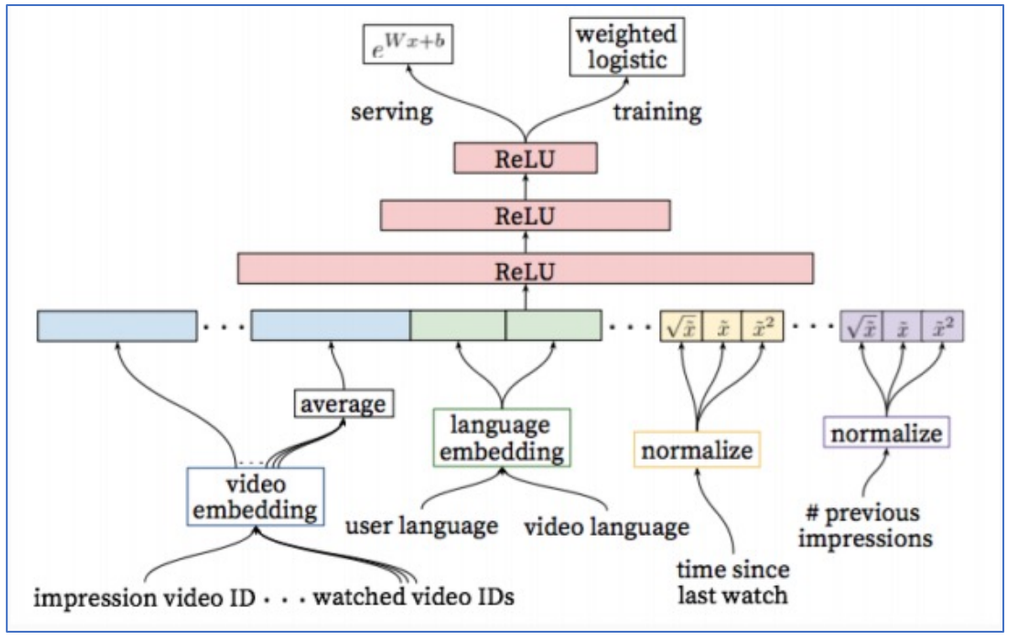


Covington, P., Adams, J., & Sargin, E. (2016). Deep neural networks for YouTube recommendations. In *Proceedings of the 10th ACM conference on recommender systems* (pp. 191-198). ACM.

Redes Neuronales en YouTube (2016-19)



Generación de Candidatos



Ranking

About the paper

- Accepted at the International Conference on Data Mining (ICDM), 2017

- Authors: Adobe Research & Prof. McAuley's Lab @ UCSD

- Proposes fashion 1) recommendation and 2) design (through GANs)

- We focus on 1)

Methodology

Source: Kang et al. 2017

Context

- Fashion domain is complex: long tails, cold starts, evolving dynamics

- Content-aware recommender systems are well-suited to it

Source: Kang et al. 2017

DVBPR Key Insights

- Opt for “domain-aware” visual embeddings instead of “off-the-shelf” as in VBPR

- Joint training of visual embeddings and recommender system

- Generate new items consistent with each user’s preference

Approach

- BPR framework: optimize rank of purchased vs non-purchased items

- Siamese trainable CNNs contrast positive and negative pairs
  - Images are retrieved and rescaled in the DataLoader

- Original datasets: Amazon fashion + Tradesy

- In this tutorial: Wikimedia Commons dataset

Model

- Users $u \in \mathcal{U}$
- Items $i \in \mathcal{I}$
- Positive items $\mathcal{I}_u^+$
- Item image X_i

VBPR:

$$x_{u,i} = \beta_i + \gamma_u^T \gamma_i + \theta_u^T (E \cdot f_i)$$

DVBPR:

$$x_{u,i} = \beta_i + \gamma_u^T \gamma_i + \theta_u^T \phi(X_i)$$

Model

VBPR:

Item bias

Latent CF factors

Visual factors

$$x_{u,i} = \beta_i + \gamma_u^T \gamma_i + \theta_u^T (E \cdot f_i)$$

Dimensionality reduction

Pre-trained embedding

DVBPR:

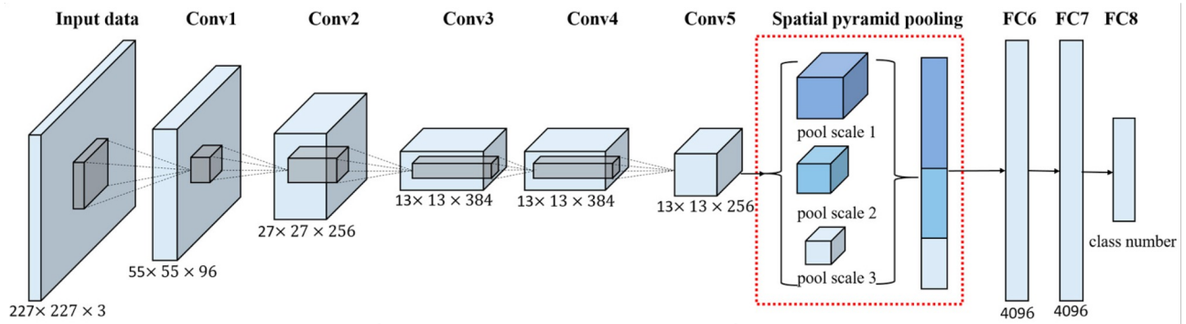
$$x_{u,i} = \beta_i + \gamma_u^T \gamma_i + \theta_u^T \phi(X_i)$$

We keep this!

Trainable CNN

Convolutional Neural Networks

- We use AlexNet with K=100



AlexNet. Source: Han et al. 2017

- Paper uses CNN-F with K=50

conv1	conv2	conv3	conv4	conv5	full6	full7	full8
64x11x11	256x5x5	256x3x3	256x3x3	256x3x3	4096	4096	K
st. 4, pad 0	st. 1, pad 2	st. 1, pad 1	st. 1, pad 1	st. 1, pad 1	drop-	drop-	-
x2 pool	x2 pool	-	-	x2 pool	out	out	-

CNN-F. Source: Kang et al. 2017

BPR Optimization

$$u \in \mathcal{U} \qquad i \in \mathcal{I}_u^+ \qquad j \in \mathcal{I} \setminus \mathcal{I}_u^+ \qquad (u,i,j) \in \mathcal{D}$$

$$\mathcal{D} = \{(u,i,j) | u \in \mathcal{U} \wedge i \in \mathcal{I}_u^+ \wedge j \in \mathcal{I} \setminus \mathcal{I}_u^+\}$$

$$\max \sum_{(u,i,j) \in \mathcal{D}} \ln \sigma(x_{u,i,j}) - \lambda_\Theta \|\Theta\|^2$$

$$x_{u,i,j} = x_{u,i} - x_{u,j}$$

Retrieval / Recommendation

$$\delta(u, c) = \underset{i \in X_c}{\operatorname{argmax}}\ x_{u,i} = \underset{i \in X_c}{\operatorname{argmax}}\ \beta_i + \gamma_u^T \gamma_i + \theta_u^T \phi(X_i)$$

Observations

- Model converges after 5 epochs (~12 hours on an 8-core CPU + GTX 1080 Ti)

- In our experience, latent CF factors are crucial for the model to learn

Datasets

Dataset	# Users	# Items	# Interactions	# Categories
Amazon Fashion	64583	234892	513367	6
Amazon Women	97678	347591	827678	53
Amazon Men	34244	110636	254870	50
Tradesy.com	33864	326393	655409	N / A
Wikimedia	1078	32959	96991	N / A

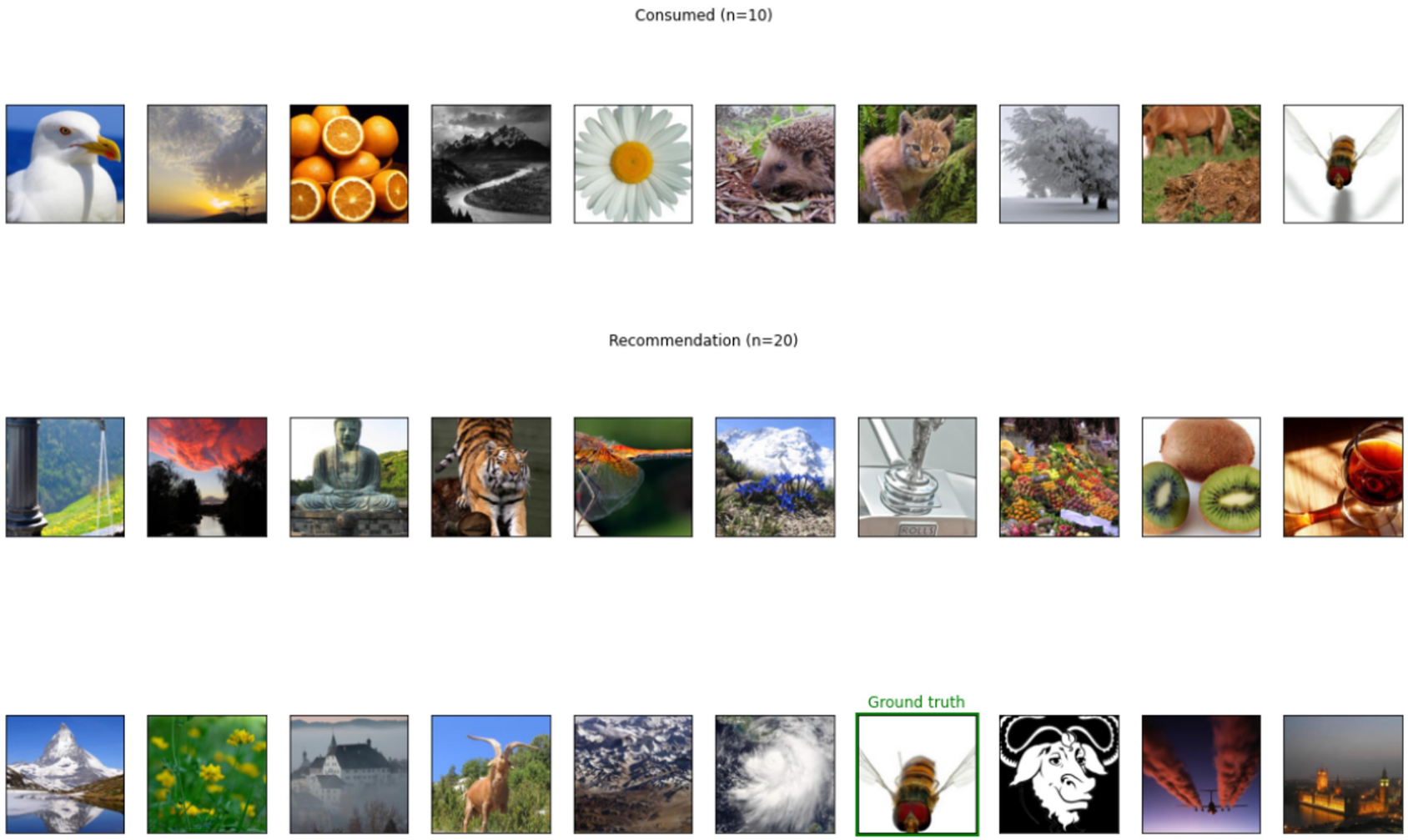
Main Results

AUC	RR	R@20	P@20	nDCG@20	R@100	P@100	nDCG@100
0.83169	0.04507	0.12152	0.00608	0.05814	0.25696	0.00257	0.08245

- Better AUC than in Tradesy and Amazon

- Best Wikimedia performer out of 4 architectures presented in this tutorial

Example recommendations



Consumed (n=10)

Recommendation (n=20)

Ground Truth (n=1)

Conclusions

- Wikimedia dataset is different from Amazon and Tradesy
  - Image quality is primary concern
  - Content < Collaborative Filtering

- This might explain why latent non-visual factors are needed

- DVBPR approach is simple yet effective for visual recommendation in challenging domains

- Newer CNN architectures might be interesting to explore
  - EfficientNet
  - Lambda Networks

VisRank & CuratorNet

- Publications:
  - Messina, P., Cartagena, M., Cerda, P., del Rio, F., & Parra, D. (2020). CuratorNet: Visually-aware Recommendation of Art Images. arXiv preprint arXiv:2009.04426. *Proceedings of ComplexRec-ImpactRS workshop, co-located at RecSys 2020*

  - Messina, P., Dominguez, V., Parra, D., Trattner, C., & Soto, A. (2019). Content-based artwork recommendation: integrating painting metadata with neural and manually-engineered visual features. *User Modeling and User-Adapted Interaction*, *29*(2), 251-290.

CuratorNet team

Pablo Messina

Manuel Cartagena

Felipe del Río

Patricio Cerda

Denis Parra

CuratorNet

- Inspired by both VBPR (2015) and YouTube Recsys (2016)

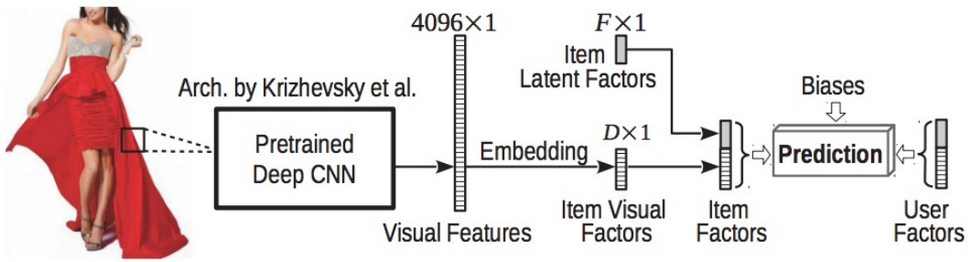
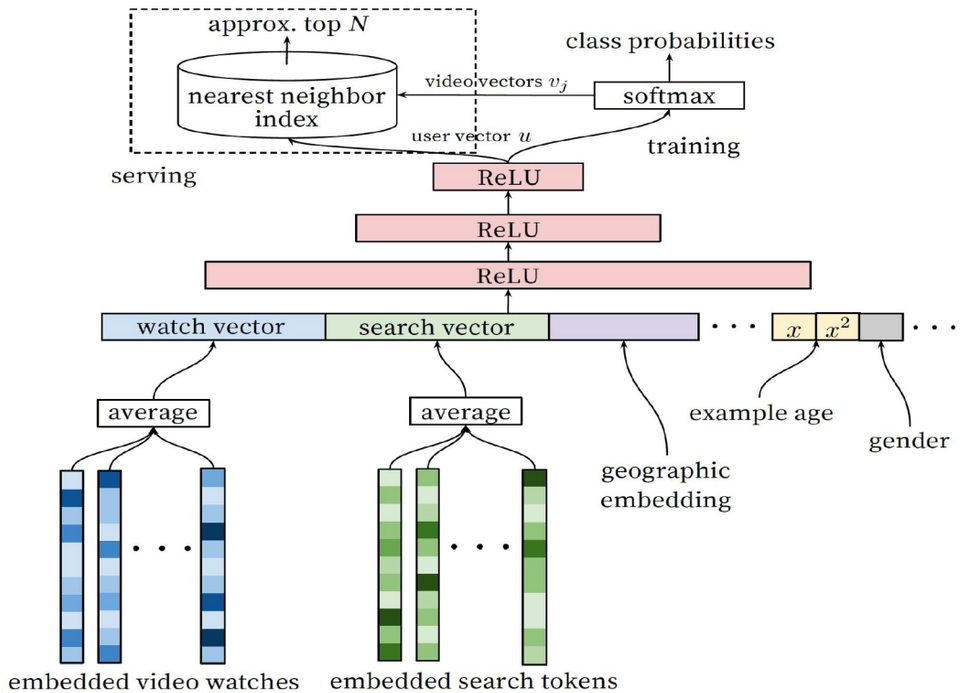


Figure 1: Diagram of our preference predictor. Rating dimensions consist of visual factors and latent (non-visual) factors. Inner products between users and item factors model the compatibility between users and items.

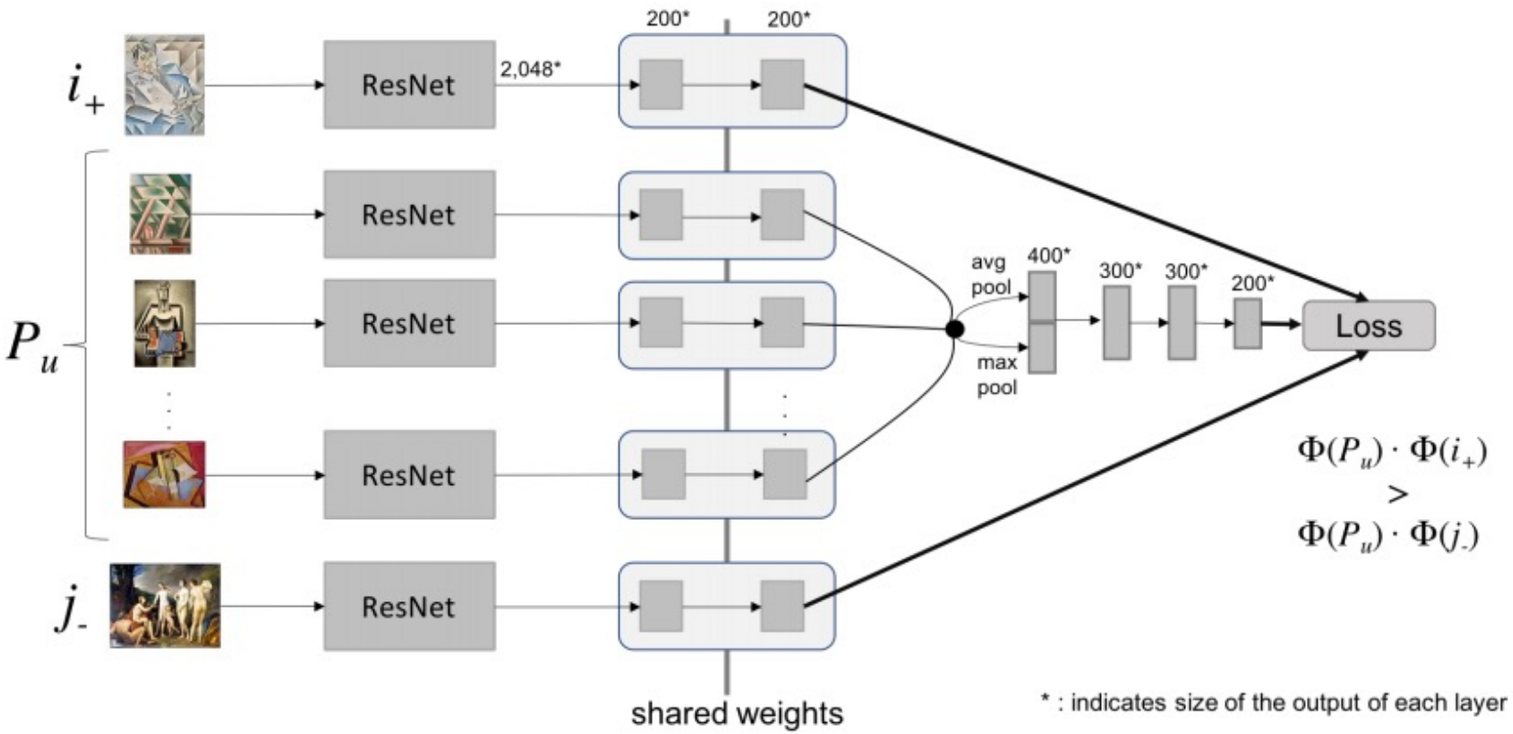
He, R., & McAuley, J. (2016). VBPR: visual Bayesian Personalized Ranking from implicit feedback. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence* (pp. 144-150).



Covington, P., Adams, J., & Sargin, E. (2016, September). Deep neural networks for youtube recommendations. In Proceedings of the 10th ACM conference on recommender systems (pp. 191-198).

CuratorNet

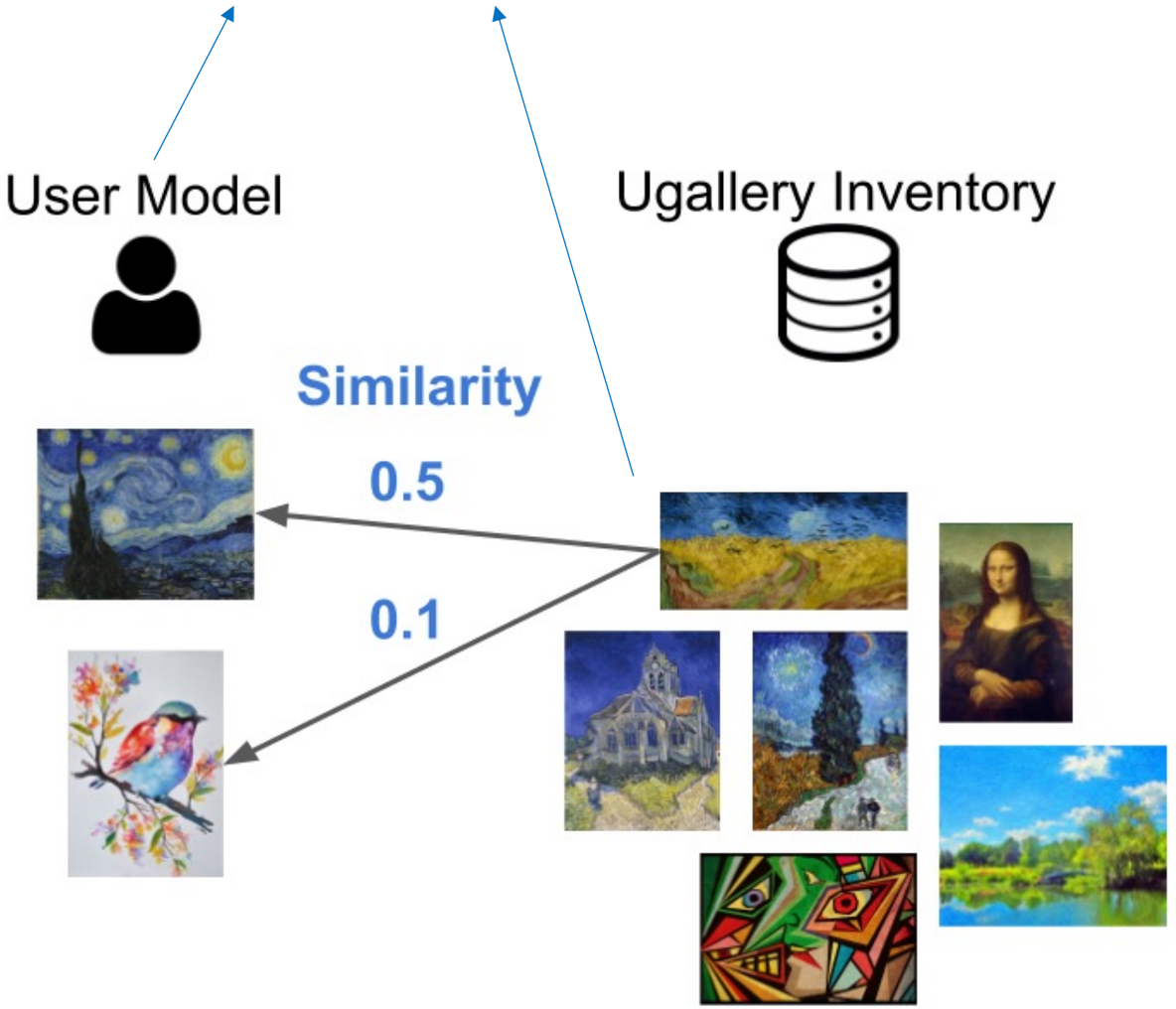
- Using BPR framework, we learn from triples (P_u, $i+$, $j-$)



Messina, P., Cartagena, M., Cerda, P., del Rio, F., & Parra, D. (2020). CuratorNet: Visually-aware Recommendation of Art Images. Proceedings of ComplexRec-ImpactRS workshop, co-located at RecSys 2020

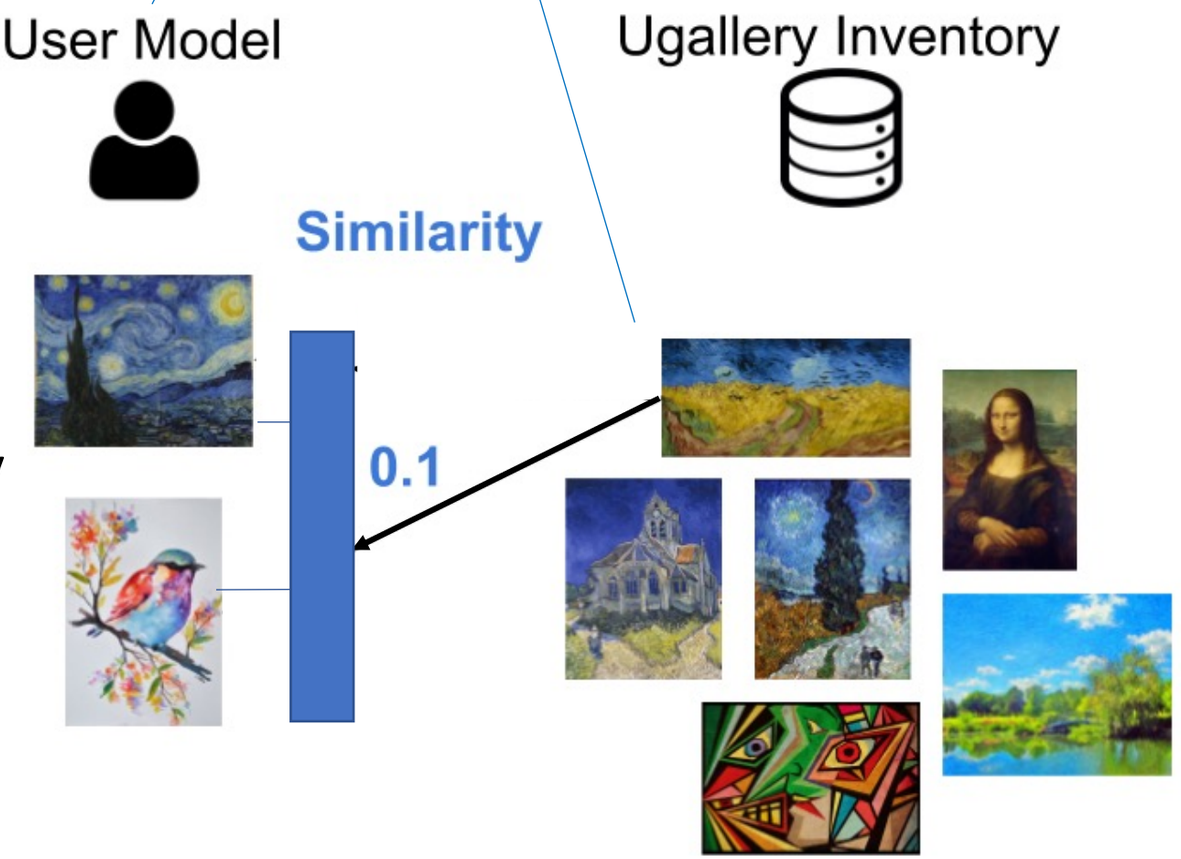
VisRank (baseline)

$$s(u,i) = score(\ ContentBasedProfile(u),\ Content(i)\ )$$



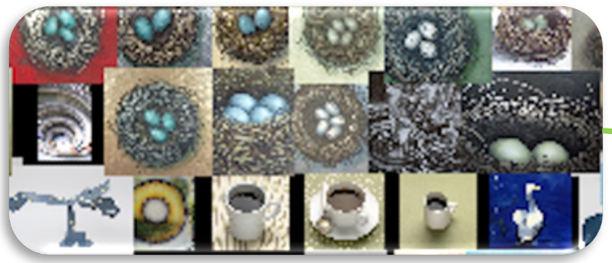
CuratorNet

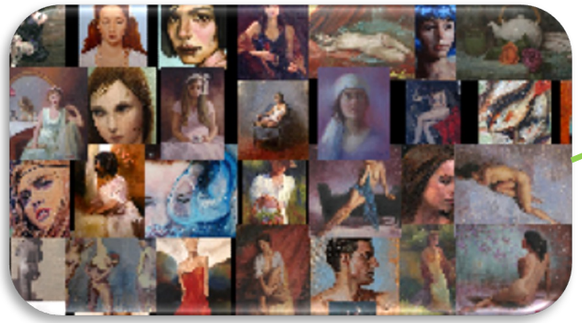
$$s(u,i) = score(\ ContentBasedProfile(u),\ Content(i)\ )$$

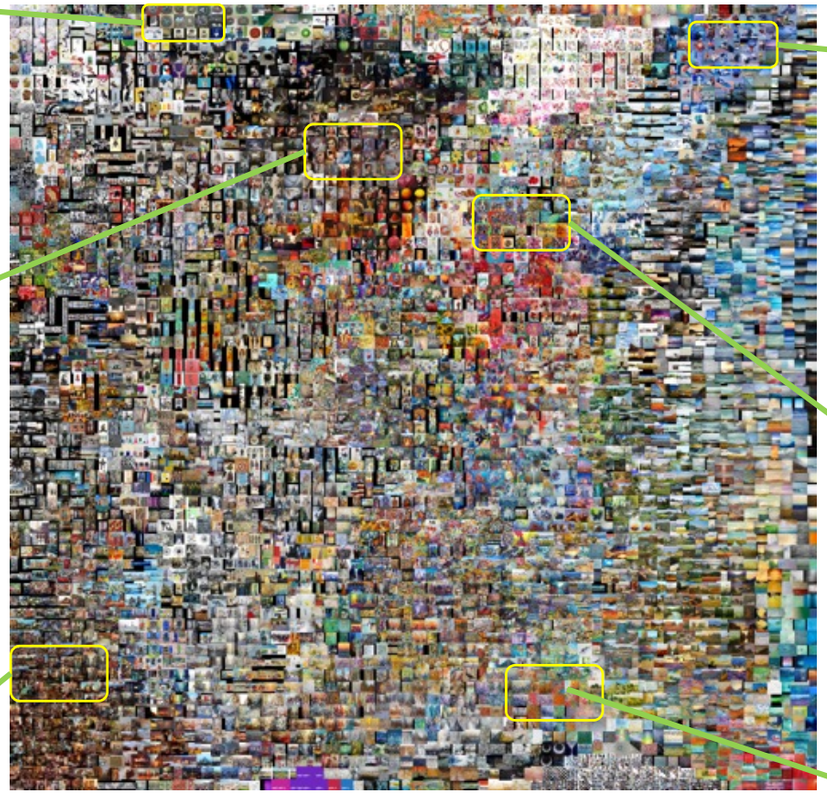


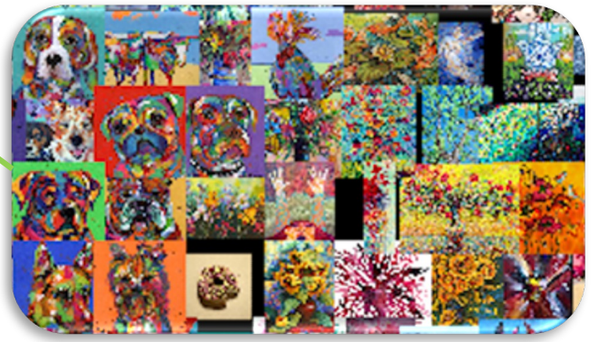
User Model representation by CuratorNet

Guidelines for negative sampling









Results: VisRank, VBPR, CuratorNet

Method	λ (L2 Reg.)	AUC	R@20	P@20	nDCG@20	R@100	P@100	nDCG@100
Oracle	–	**1.0000**	**1.0000**	**.0655**	**1.0000**	**1.0000**	**.0131**	**1.0000**
CuratorNet	.0001	**.7204**	**.1683**	**.0106**	**.0966**	**.3200**	**.0040**	**.1246**
CuratorNet	.001	**.7177**	**.1566**	**.0094**	**.0895**	**.2937**	**.0037**	**.1160**
VisRank	–	**.7151**	**.1521**	**.0093**	**.0956**	**.2765**	**.0034**	**.1195**
CuratorNet	0	**.7131**	**.1689**	**.0100**	**.0977**	**.3048**	**.0038**	**.1239**
CuratorNet	.01	.7125	.1235	.0075	.0635	.2548	.0032	.0904
VBPR	.0001	.6641	.1368	.0081	.0728	.2399	.0030	.0923
VBPR	0	.6543	.1287	.0078	.0670	.2077	.0026	.0829
VBPR	.001	.6410	.0830	.0047	.0387	.1948	.0024	.0620
VBPR	.01	.5489	.0101	.0005	.0039	.0506	.0006	.0118
Random	–	.4973	.0103	.0006	.0041	.0322	.0005	.0098

Guidelines for negative sampling

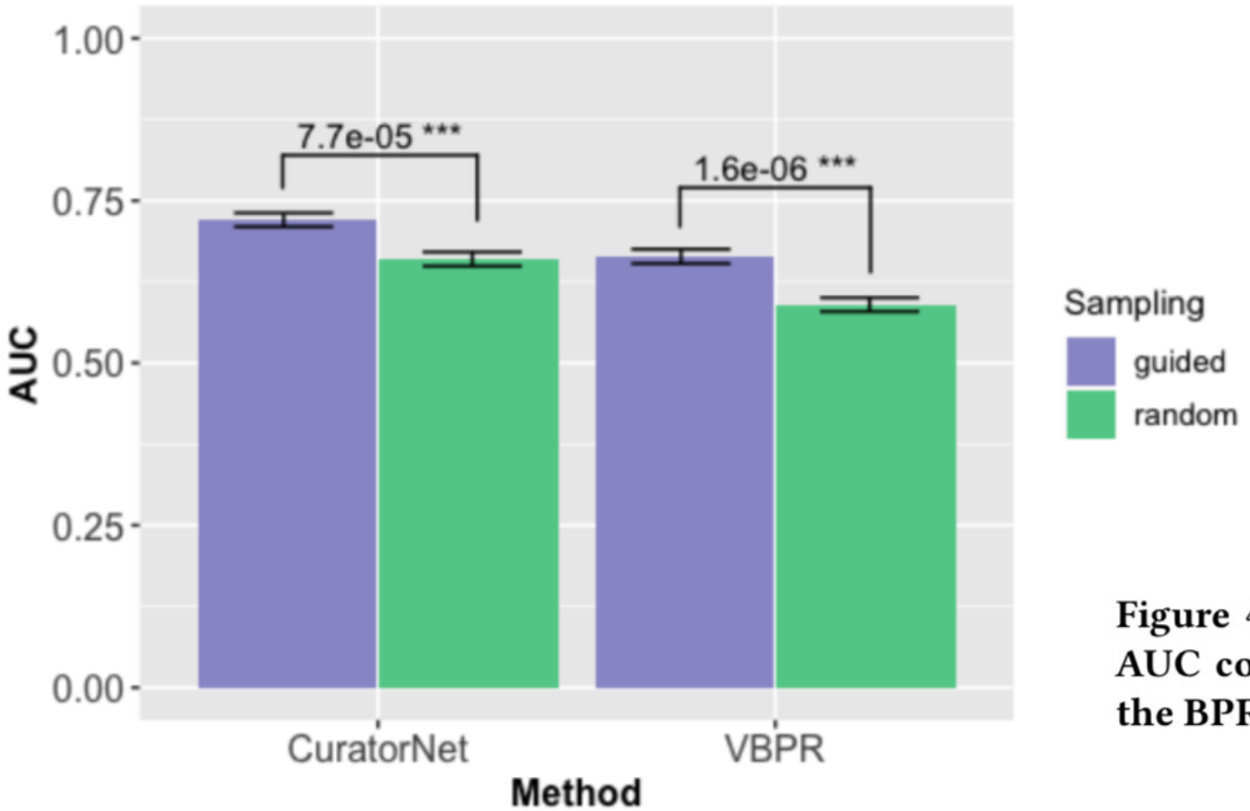


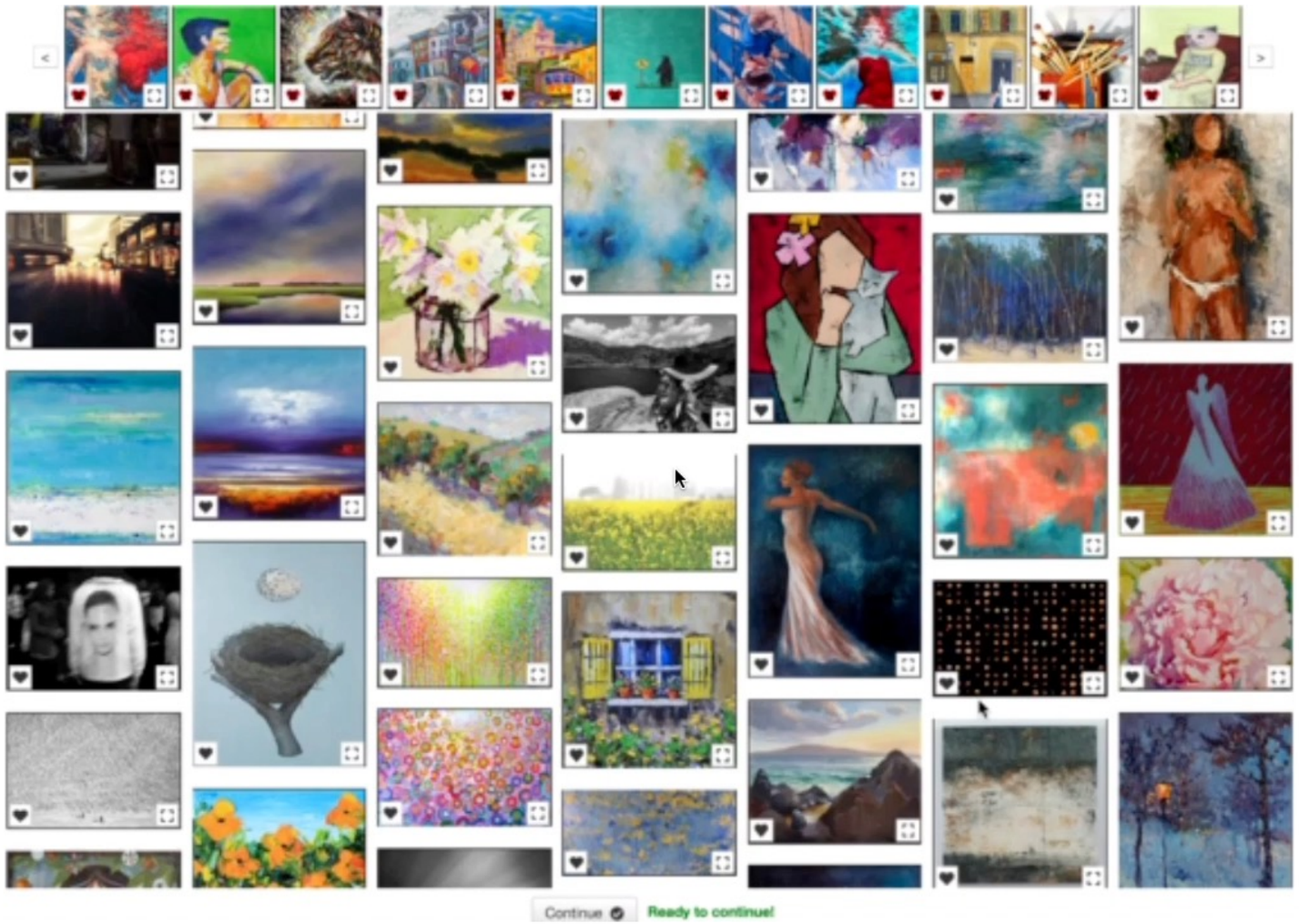
Figure 4: The sampling guidelines had a positive effect on AUC compared to random negative sampling for building the BPR training set.

Conclusion

- CuratorNet improves upon VBPR and VisRank for the case of one-of-a-kind recommendation
- Unlike VBPR, CuratorNet does not need re-training to recommend to new users since it does not explicitly train user factors
- The proposed sampling guidelines benefit both CuratorNet and VBPR

Code and Experimental Data https://github.com/ialab-puc/CuratorNet

CuratorNet Demo Ugallery



Ecosistema de DeepLearning

- PyTorch
- Keras
- Tensorflow
- Mxnet
- …