# Deep Learning en Sistemas de Recomendación

Denis Parra

IIC1005 Sistemas Recomendadores

PUC Chile

# En esta clase

- Introducción a las redes neuronales artificiales
  - Bases Biológicas
  - Perceptron
  - FeedForward y BackPropagation
- Algunos tipos de arquitecturas de redes neuronales
- Aplicaciones en Sistemas Recomendadores
  - YouTube Recommendations
  - Video Recommendations
  - Recomendación de Productos

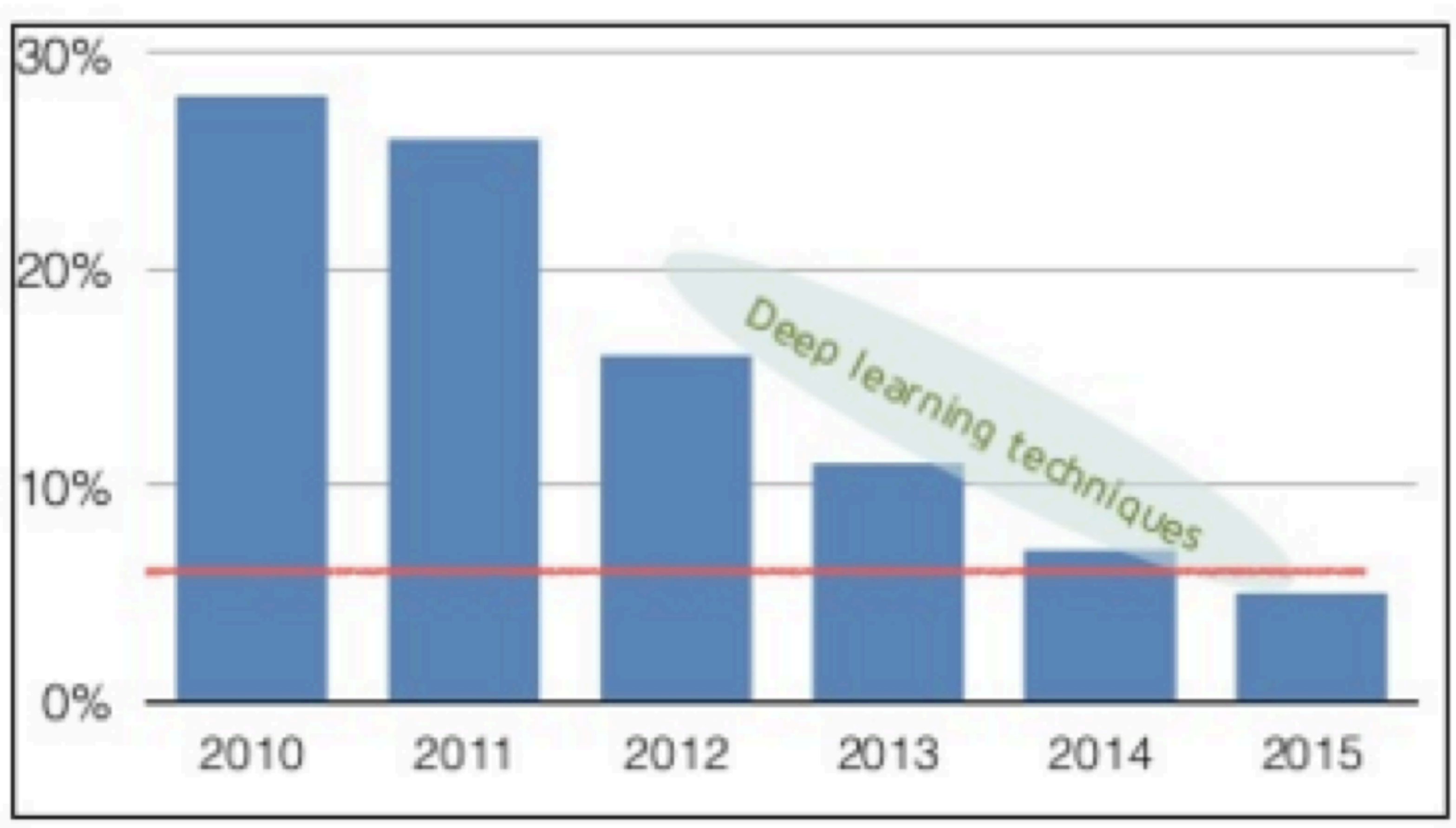
# Referencias

- Estas slides fueron posibles gracias al material que otros investigadores han compartido en línea:
  - Arno Candel http://www.slideshare.net/0xdata/deep-learning-through-examples
  - Alexandros Karatzoglou
  - https://www.slideshare.net/kerveros99/deep-learning-for-recommender-systems-recsys2017-tutorial
  - Balazs Hidasi http://www.slideshare.net/balazshidasi/deep-learning-to-the-rescue-solving-long-standing-problems-of-recommender-systems
  - Tom Kenter et al. http://nn4ir.com/slides/02_Preliminaries.pdf

# Presentaciones el Jueves

- L. Cubillos
  Sungyong Seo, Jing Huang, Hao Yang, and Yan Liu. Interpretable Convolutional Neural Networks with Dual Local and Global Attention for Review Rating Prediction. (2017)
- A. Espinosa
  Massimo Quadrana, Alexandros Karatzoglou, Balázs Hidasi, and Paolo Cremonesi. 2017. Personalizing Session-based Recommendations with Hierarchical Recurrent Neural Networks. In Proceedings of the Eleventh ACM Conference on Recommender Systems (RecSys 2017)
- G. Contreras
  Trapit Bansal, David Belanger, and Andrew McCallum. Ask the GRU: Multi-task Learning for Deep Text Recommendations. In Proceedings of the 10th ACM Conference on Recommender Systems (RecSys 2016).
- A. Ossa
  Pyykkö, J., & Głowacka, D.. Interactive content-based image retrieval with deep neural networks. In International Workshop on Symbiotic Interaction (pp. 77-88). Springer, Cham. (2016)

# ¿Por qué Deep Learning?



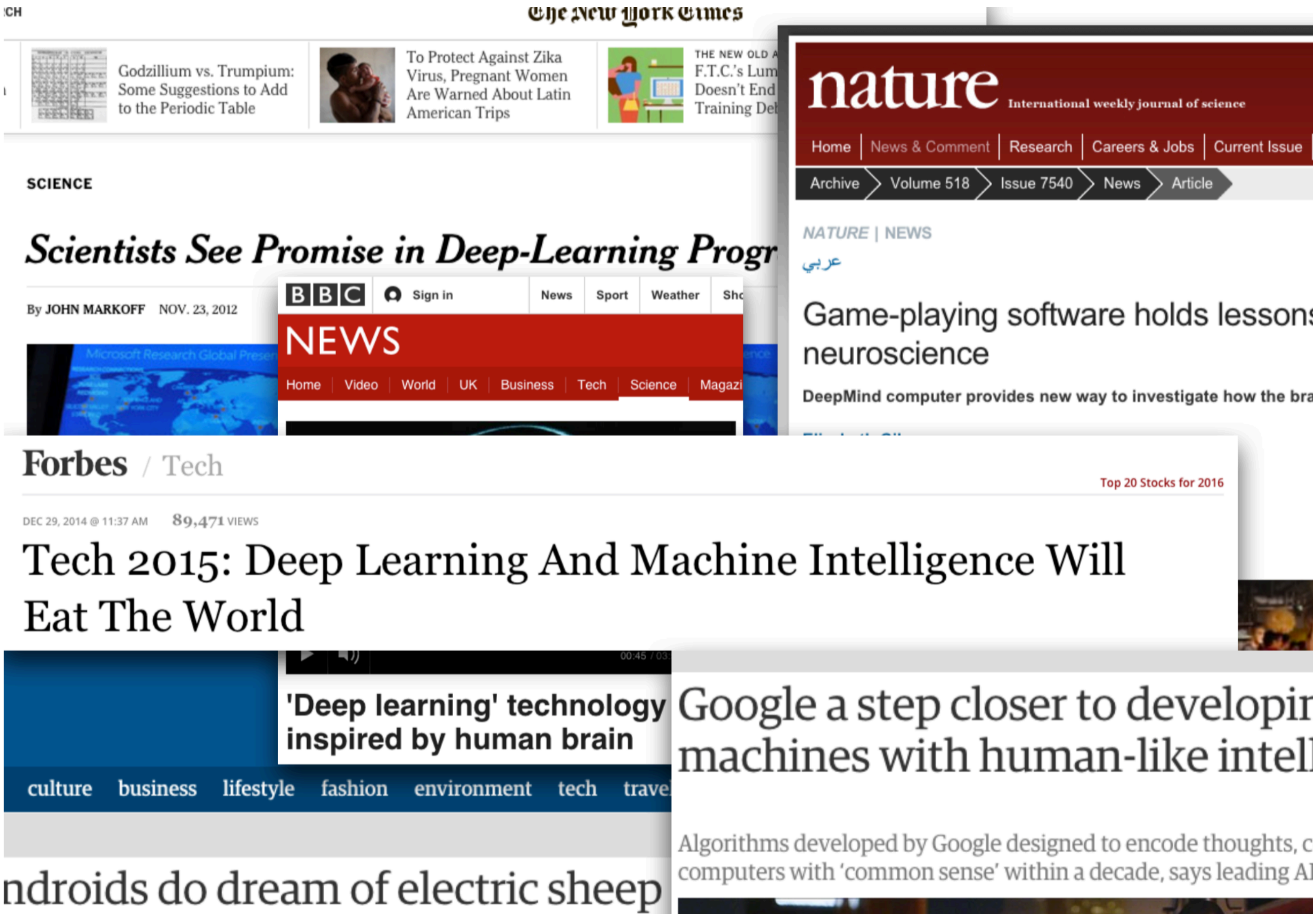

ImageNet challenge error rates (red line = human performance)

# ¿Por qué Deep Learning?

hand-crafted Feature Extractor → "Simple" Trainable Classifier →

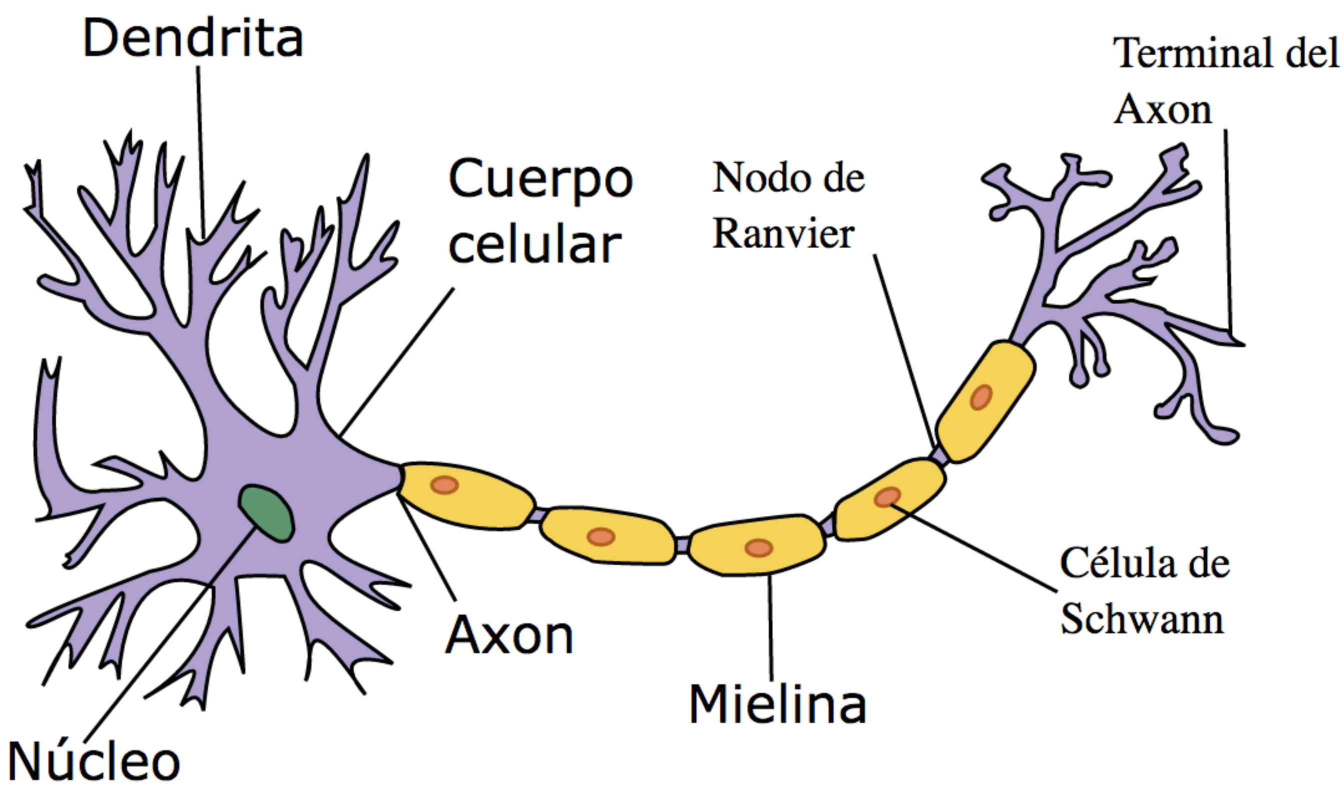
Trainable Feature Extractor → Trainable Classifier →

# ¿Por qué Deep Learning?

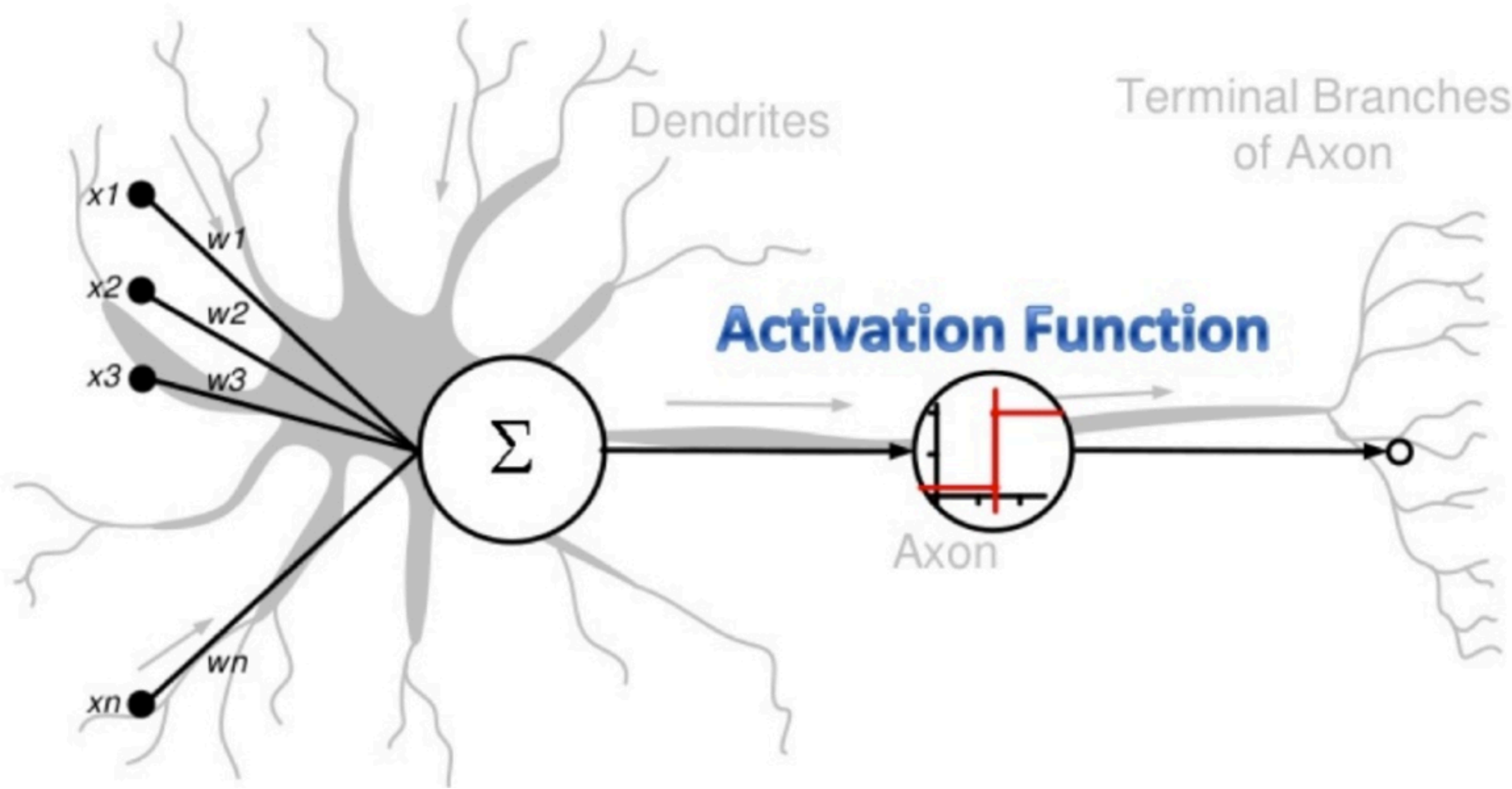


# Bases Biológicas: Neurona

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

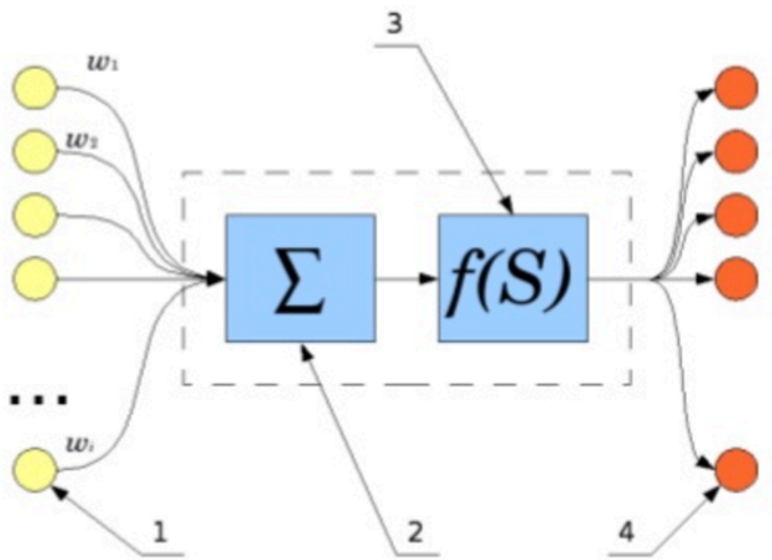


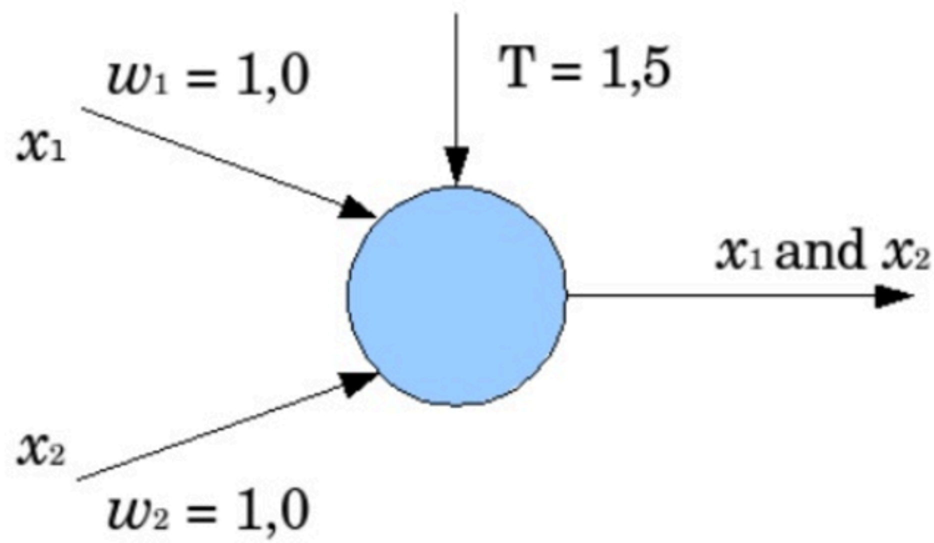
# Redes Neuronales Artificiales



# Redes Neuronales Artificiales

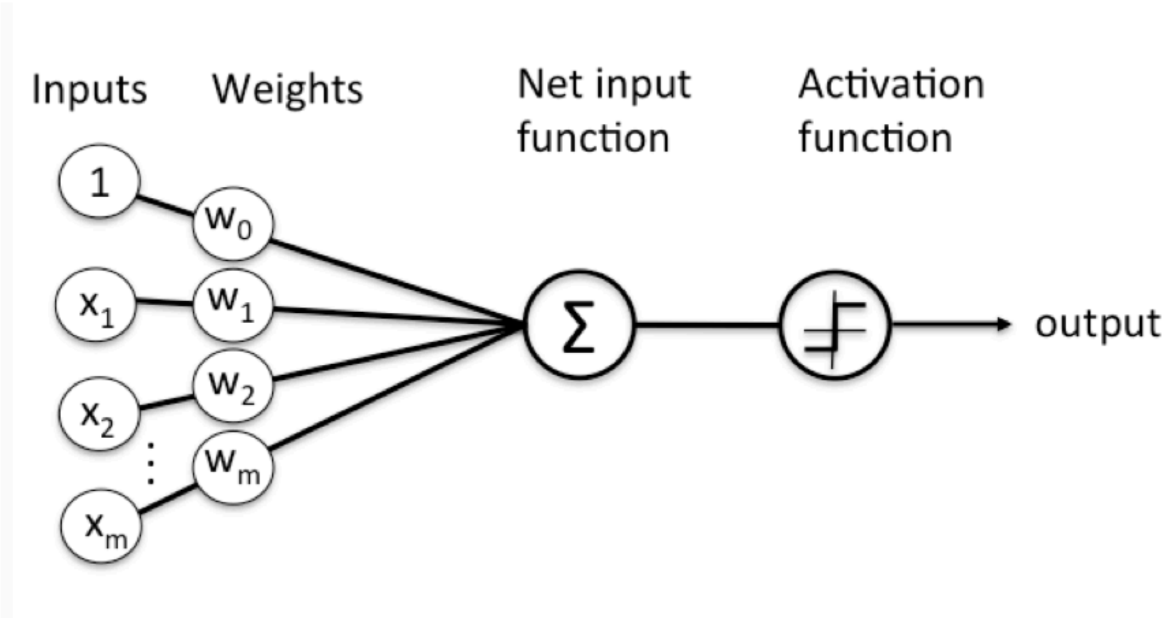
- 1943: McCulloh y Pitts “A Logical Calculus of the Ideas Immanent in Nervous Activity”



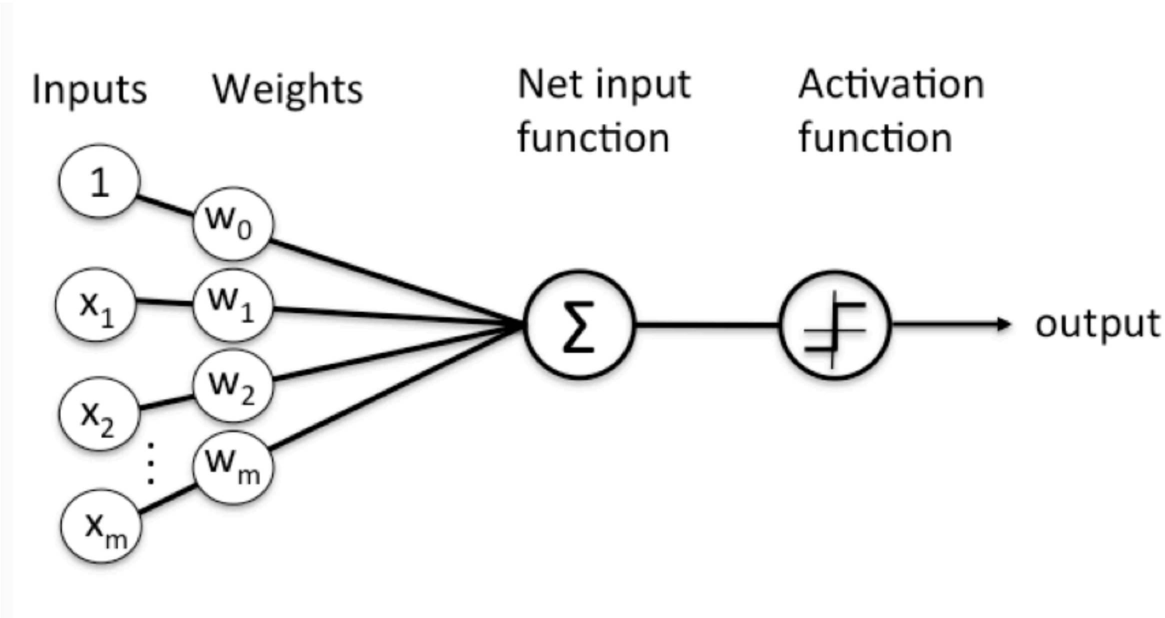
# Perceptron

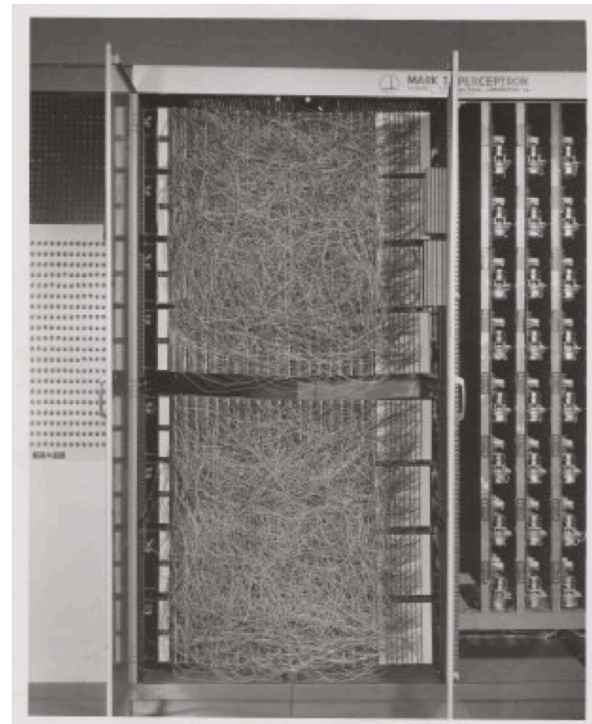
- 1957: Frank Rosenblatt



# Perceptron

- 1957: Frank Rosenblatt



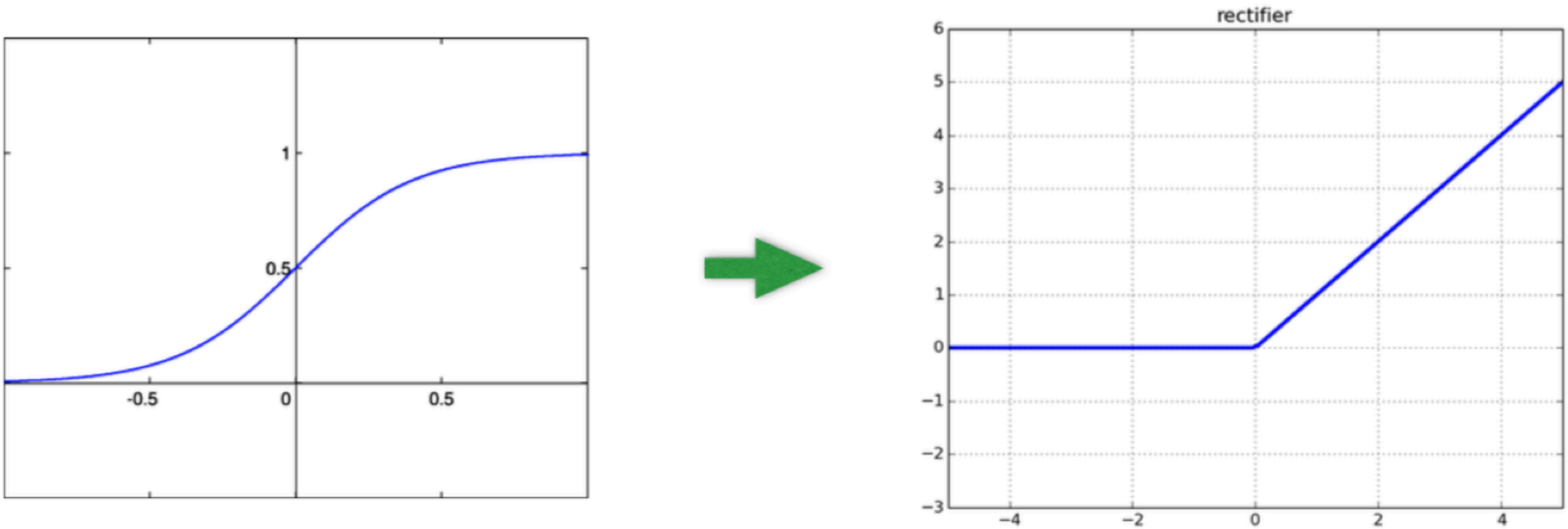



*"[The Perceptron is] the embryo of an electronic computer that [the Navy] expects will be able to walk, talk, see, write, reproduce itself and be conscious of its existence."*

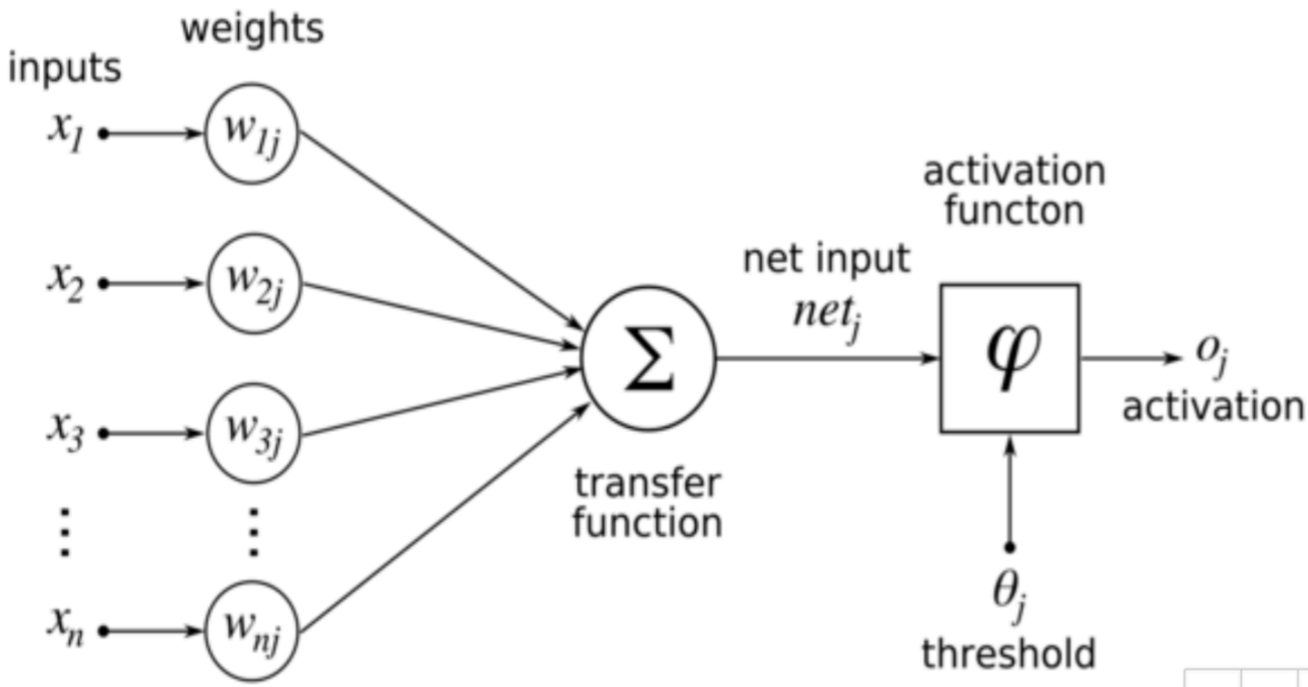
THE NEW YORK TIMES

# Funciones de Activación

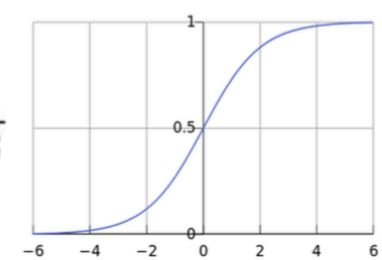
- Step, tanh, sigmoid, ReLU (más reciente)



# Perceptron



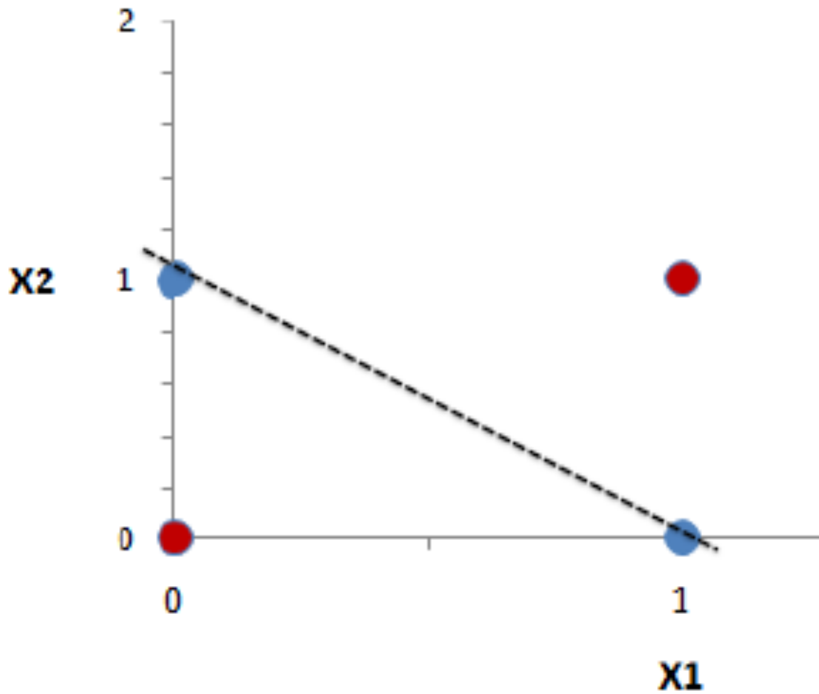

$$\phi = \frac{1}{1 + e^{-\Sigma}}$$




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

# Limitaciones del Perceptron

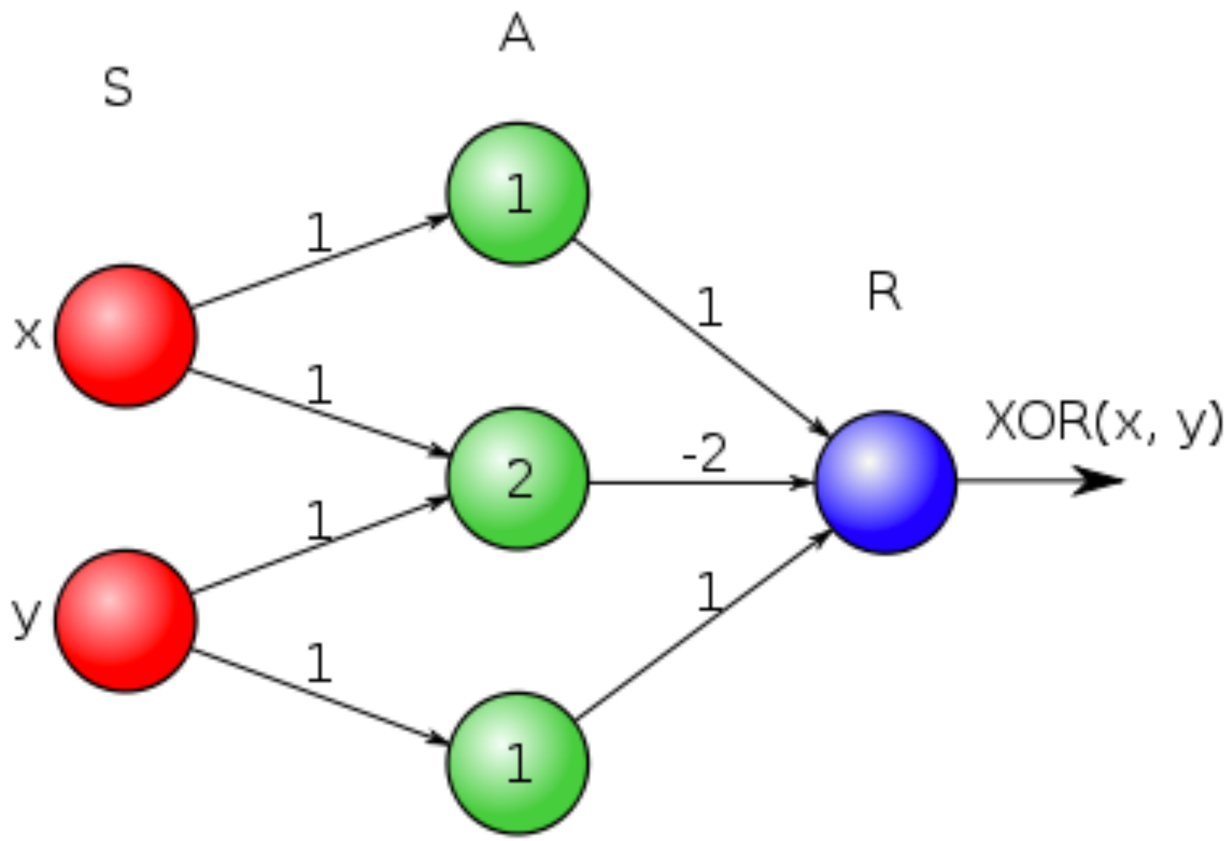
- El perceptron de una capa es un clasificador lineal
- No puede separar alguna funciones como el XOR



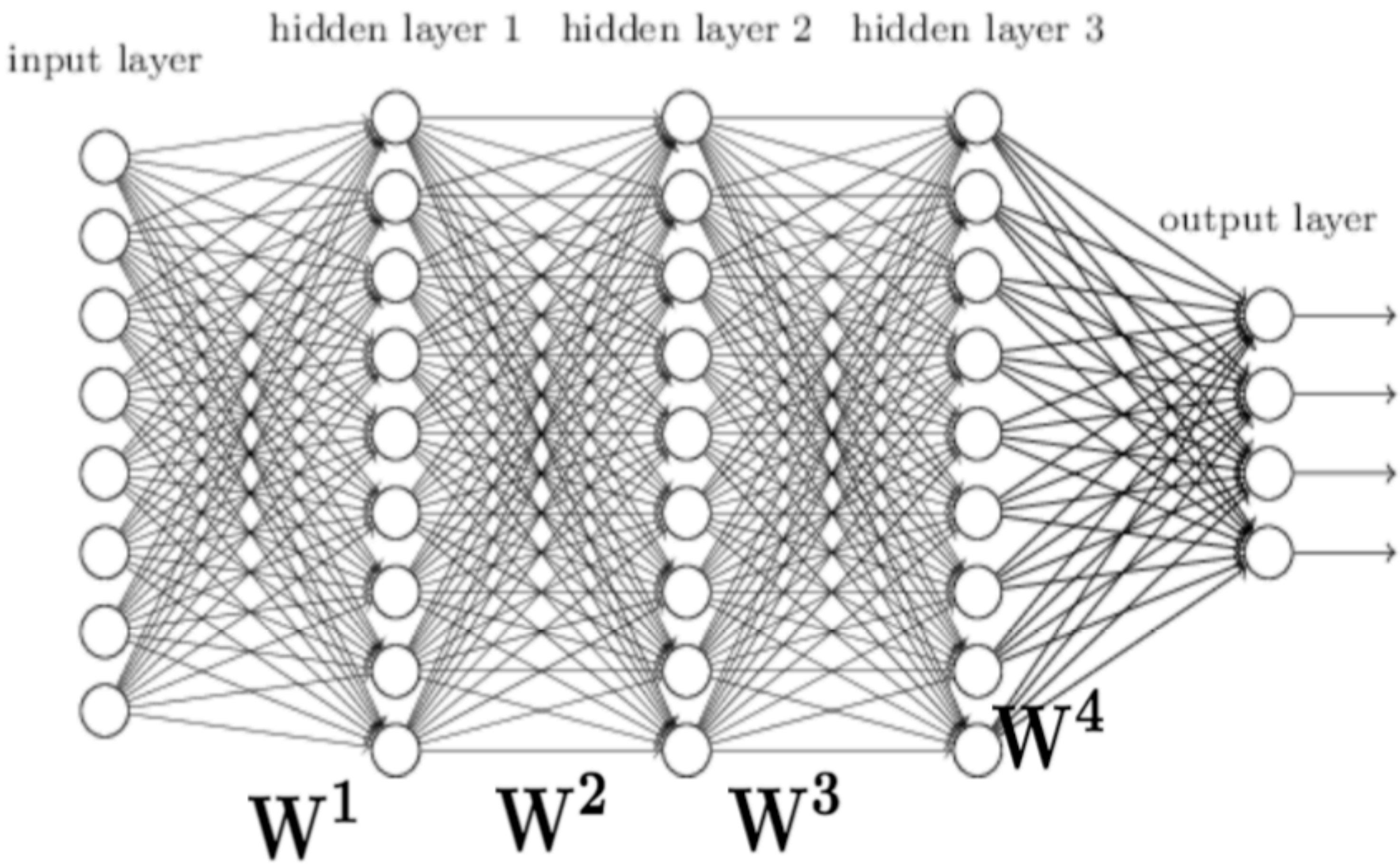

**XOR**

| X1 | X2 | Y |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

# Perceptron con Hidden layers

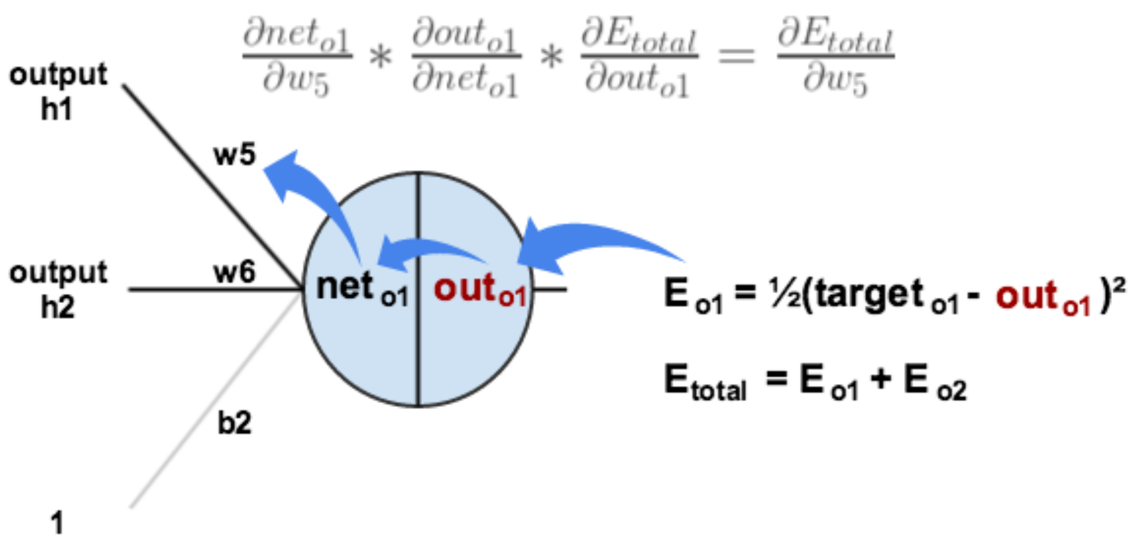


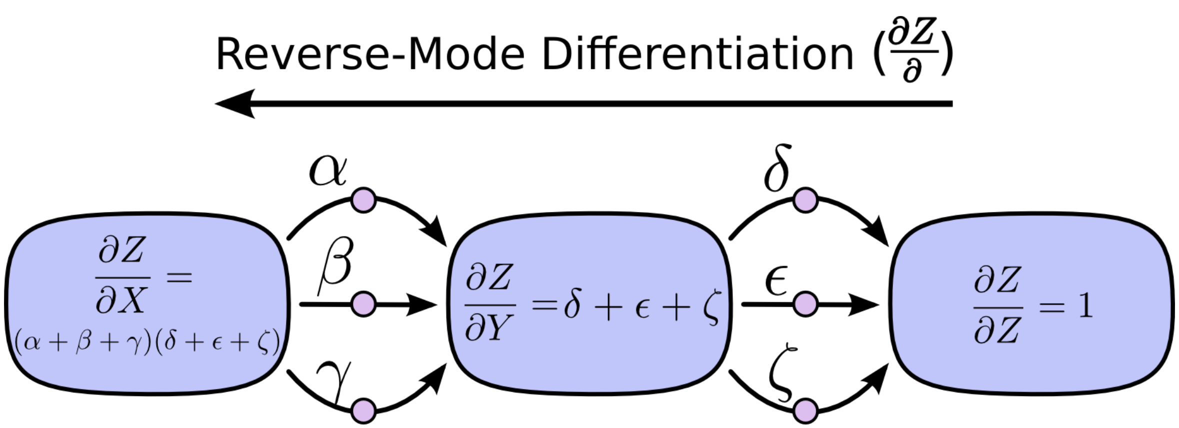
# Redes Feedforward Multilayer




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

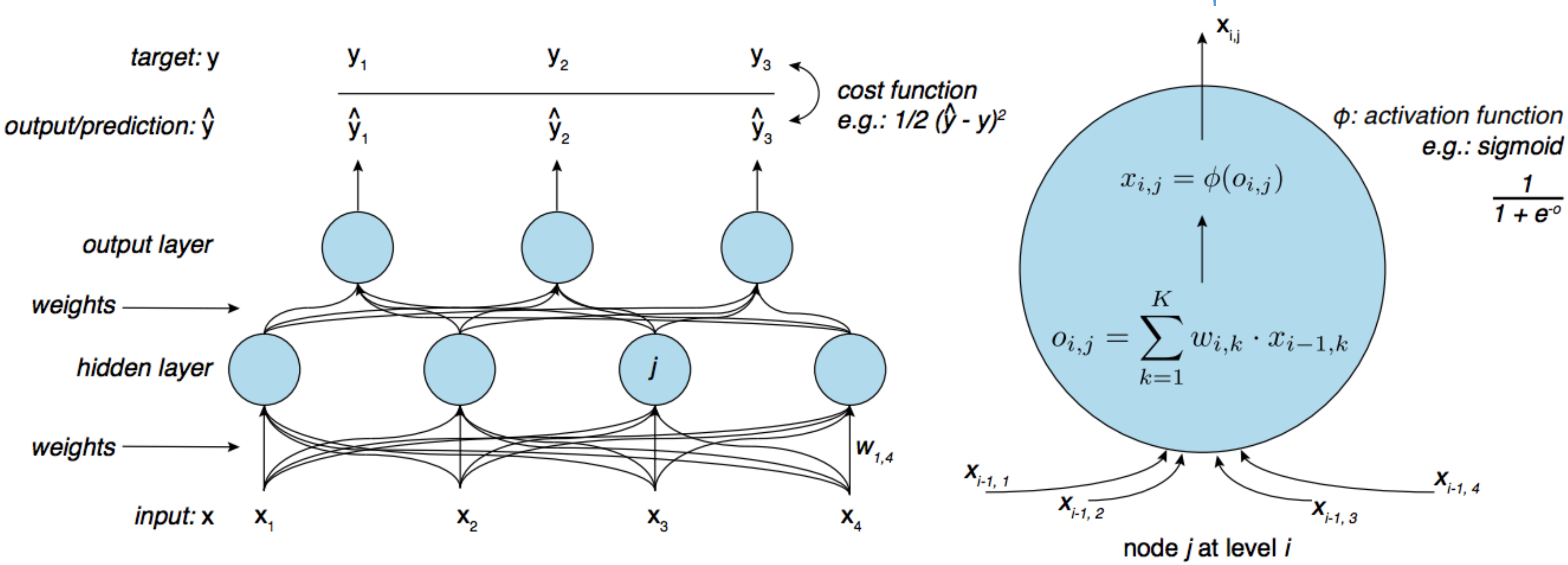
# Backpropagation: Aprender los $W_i$



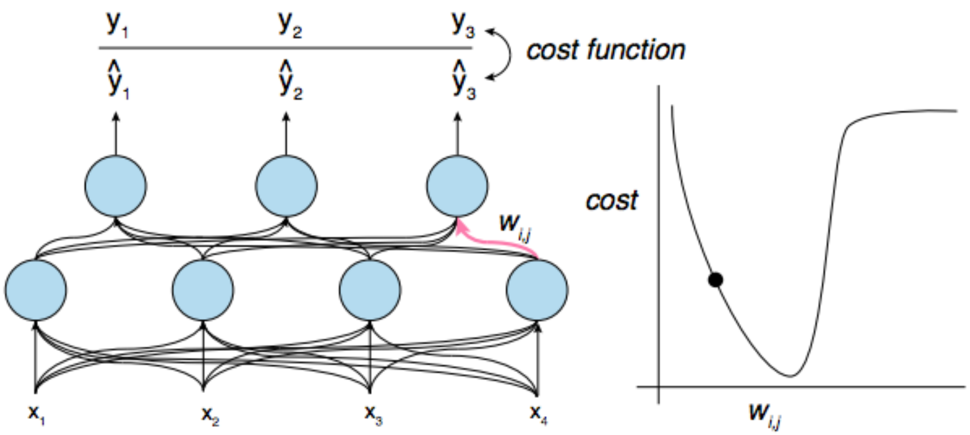
# Backpropagation: Ejemplo

$x_{ij}$ : output of node *j* at layer *i*




http://nn4ir.com/slides/02_Preliminaries.pdf

# Backpropagation: Ejemplo




until convergence:

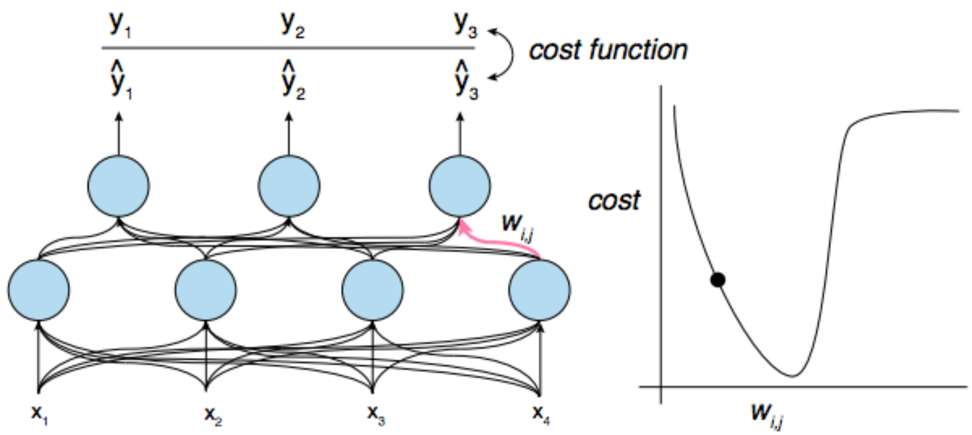
- do a forward pass
- compute the cost/error
- adjust weights $\leftarrow$ how??

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

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

$\alpha$ is the learning rate.

# Backpropagation: Ejemplo



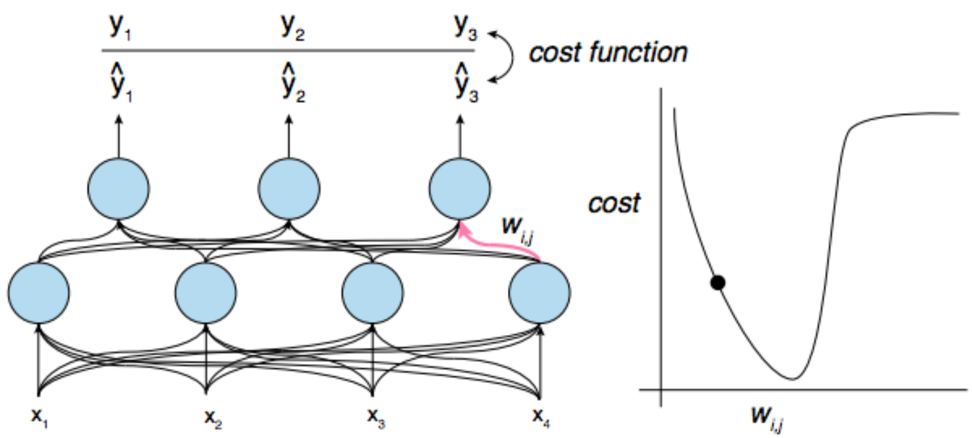

$$\Delta w_{i,j} = -\alpha \ \frac{\partial cost}{\partial w_{i,j}}$$

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

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

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

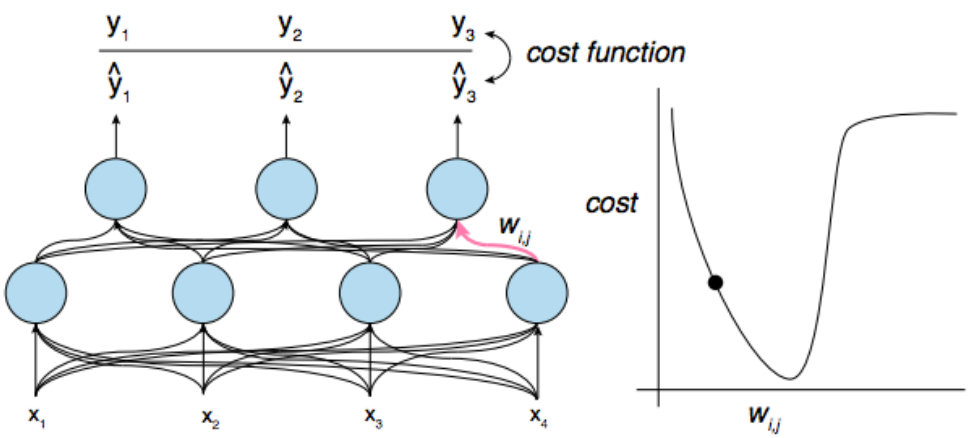
# Backpropagation: Ejemplo




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

$$cost(\hat{y}, y) = \frac{1}{2}(y - \hat{y})^2$$
$$\hat{y}_j = x_{i,j} = \phi(o_{i,j}), \text{e.g. } \sigma(o_{i,j})$$
$$x_{i,j} = \sigma(o) = \frac{1}{1 + e^{-o}}$$
$$o_{i,j} = \sum_{k=1}^{K} w_{i,k} \cdot x_{i-1,k}$$

# Backpropagation: Ejemplo




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

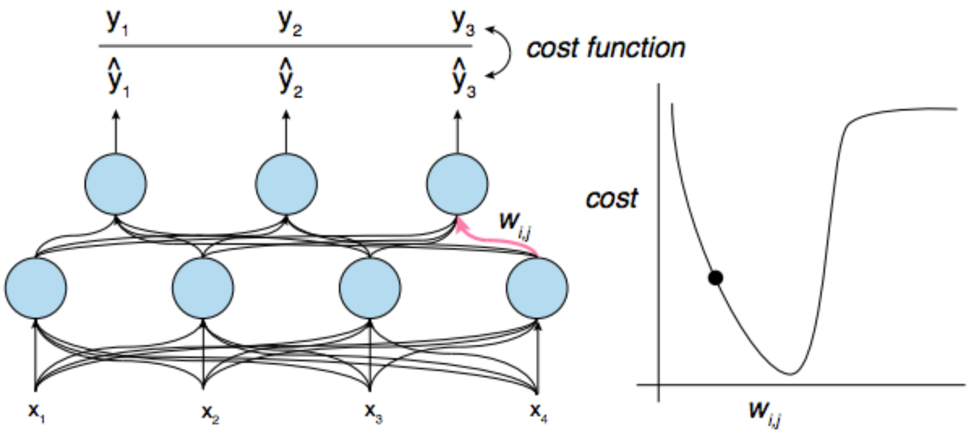
$$cost(\hat{y}, y) = \frac{1}{2}(y - \hat{y})^2$$

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

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

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

# Backpropagation: Ejemplo




$$
\begin{aligned}
\Delta w_{i,j} &= -\alpha \frac{\partial cost}{\partial w_{i,j}} \\
&= -\alpha \frac{\partial cost}{\partial x_{i,j}} \frac{\partial x_{i,j}}{\partial w_{i,j}} \\
&= -\alpha \frac{\partial cost}{\partial x_{i,j}} \frac{\partial x_{i,j}}{\partial o_{i,j}} \frac{\partial o_{i,j}}{\partial w_{i,j}} \\
&= -\alpha \frac{\partial cost}{\partial x_{i,j}} \frac{\partial x_{i,j}}{\partial o_{i,j}} \boxed{x_{i-1,j}}
\end{aligned}
$$

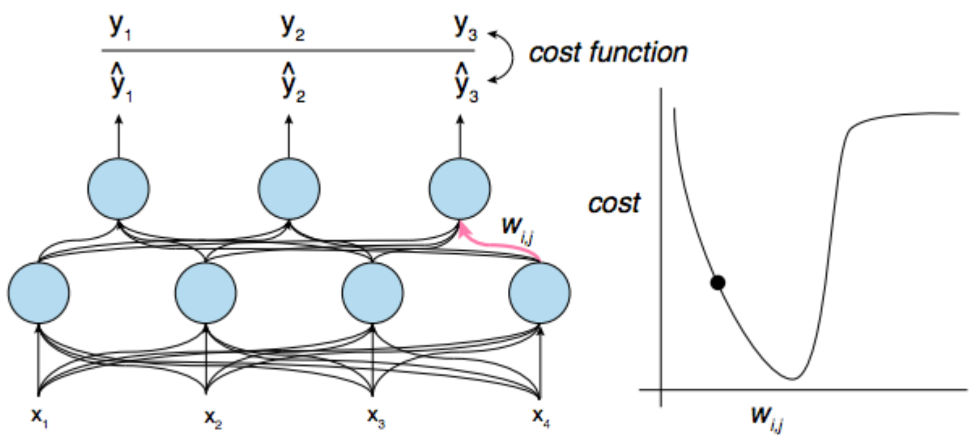
$$cost(\hat{y}, y) = \frac{1}{2}(y - \hat{y})^2$$

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

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

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

# Backpropagation: Ejemplo




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

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

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

$$= -\alpha \frac{\partial cost}{\partial x_{i,j}} \boxed{x_{i,j}(1 - x_{i,j})} \; x_{i-1,j}$$

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

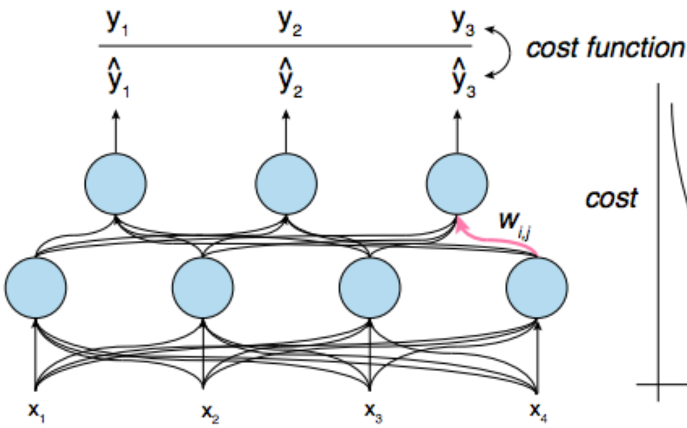
$$\hat{y}_j = x_{i,j} = \phi(o_{i,j}), \text{e.g. } \sigma(o_{i,j})$$

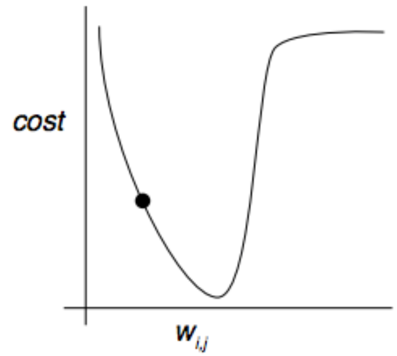
$$x_{i,j} = \sigma(o) = \frac{1}{1 + e^{-o}}$$

$$\boxed{\sigma'(o) = \sigma(o)(1 - \sigma(o))}$$

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

# Backpropagation: Ejemplo




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

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

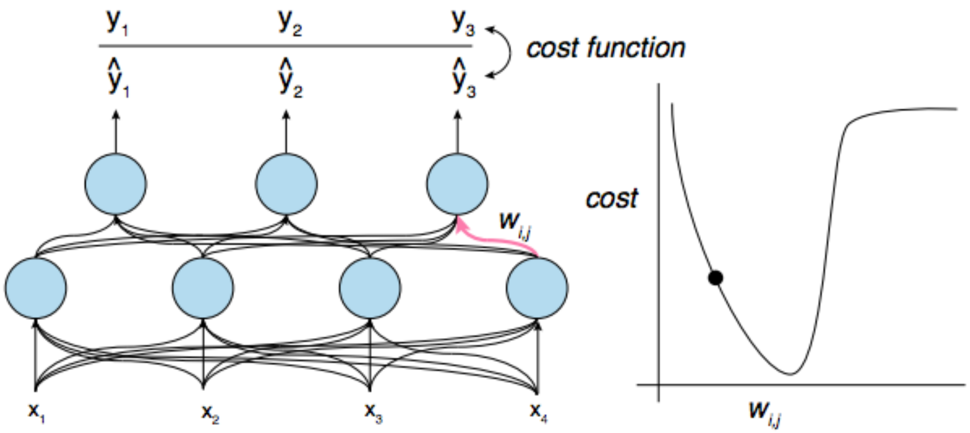
$$\hat{y}_j = x_{i,j} = \phi(o_{i,j}), \text{e.g. } \sigma(o_{i,j})$$

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

$$\boxed{\sigma'(o) = \sigma(o)(1 - \sigma(o))}$$

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

# Backpropagation: Ejemplo




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

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

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

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

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

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

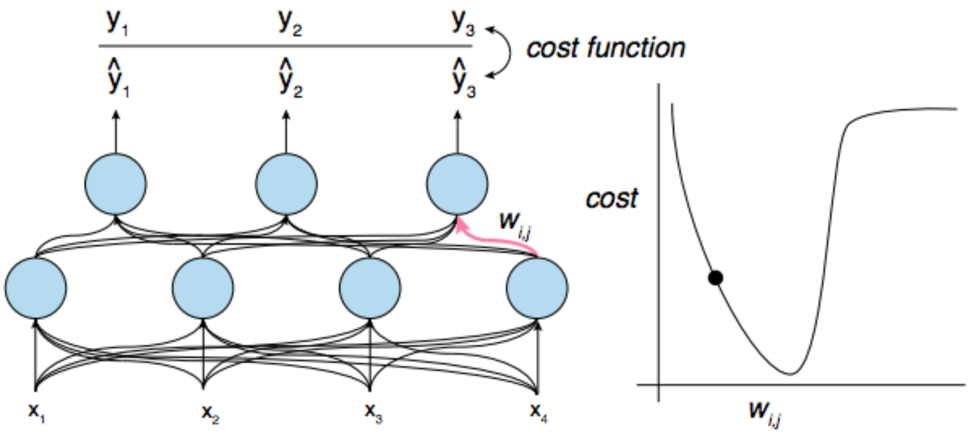
$$x_{i,j} = \sigma(o) = \frac{1}{1 + e^{-o}}$$

$$\boxed{\sigma'(o) = \sigma(o)(1 - \sigma(o))}$$

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

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

# Backpropagation: Ejemplo




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

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

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

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

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

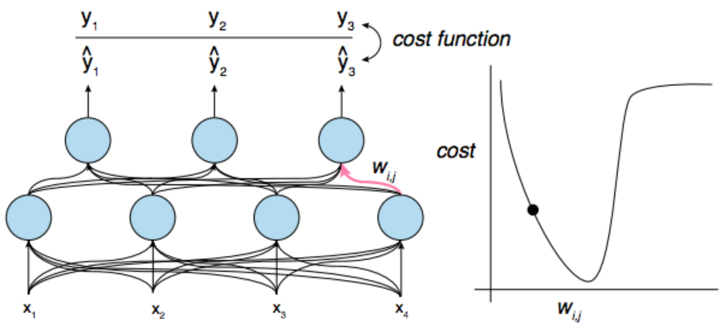
$$\hat{y}_j = x_{i,j} = \phi(o_{i,j}), \text{e.g. } \sigma(o_{i,j})$$

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

$$\sigma'(o) = \sigma(o)(1 - \sigma(o))$$

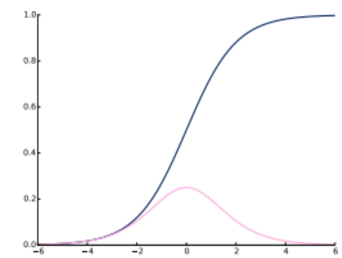
$$o_{i,j} = \sum_{k=1}^{K} w_{i,k} \cdot x_{i-1,k}$$

# Backpropagation: Ejemplo




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

$\sigma(o)$ $\sigma'(o)$



$$cost(\hat{y}, y) = \frac{1}{2}(y - \hat{y})^2$$
$$\hat{y}_j = x_{i,j} = \phi(o_{i,j}), \text{e.g. } \sigma(o_{i,j})$$
$$x_{i,j} = \sigma(o) = \frac{1}{1 + e^{-o}}$$
$$\sigma'(o) = \sigma(o)(1 - \sigma(o))$$
$$o_{i,j} = \sum_{k=1}^{K} w_{i,k} \cdot x_{i-1,k}$$

# Backpropagation: Ejemplo

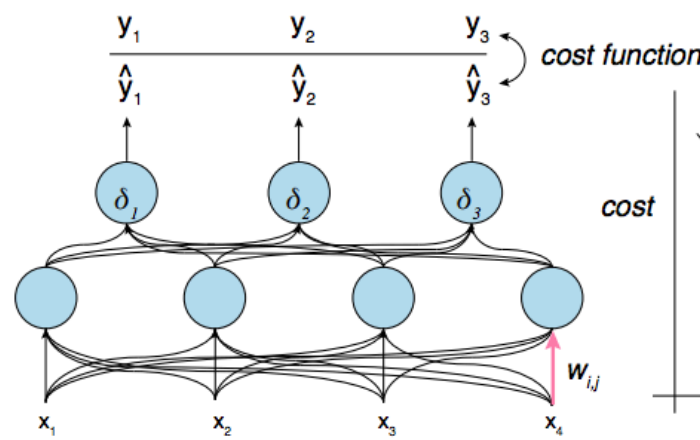
$$\Delta w_{i,j} = -\alpha \ \frac{\partial cost}{\partial w_{i,j}}$$

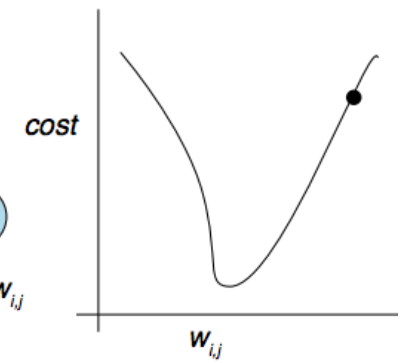
$$= -\alpha \ \frac{\partial cost}{\partial x_{i,j}} \ \frac{\partial x_{i,j}}{\partial o_{i,j}} \ \frac{\partial o_{i,j}}{\partial w_{i,j}}$$

$$= \text{l.rate} \ \ cost \ activation \ input$$

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

$$= -\alpha \ \delta \ \ x_{i-1,j}$$

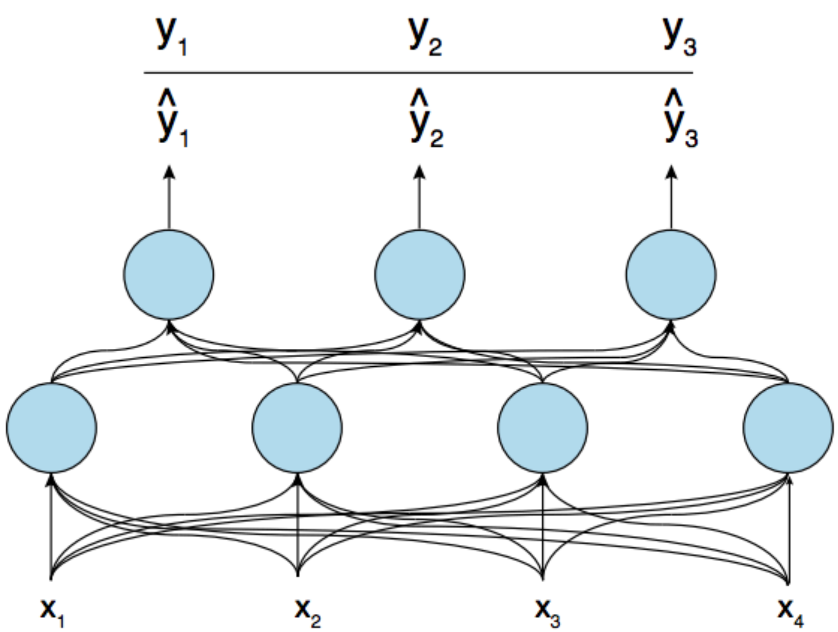


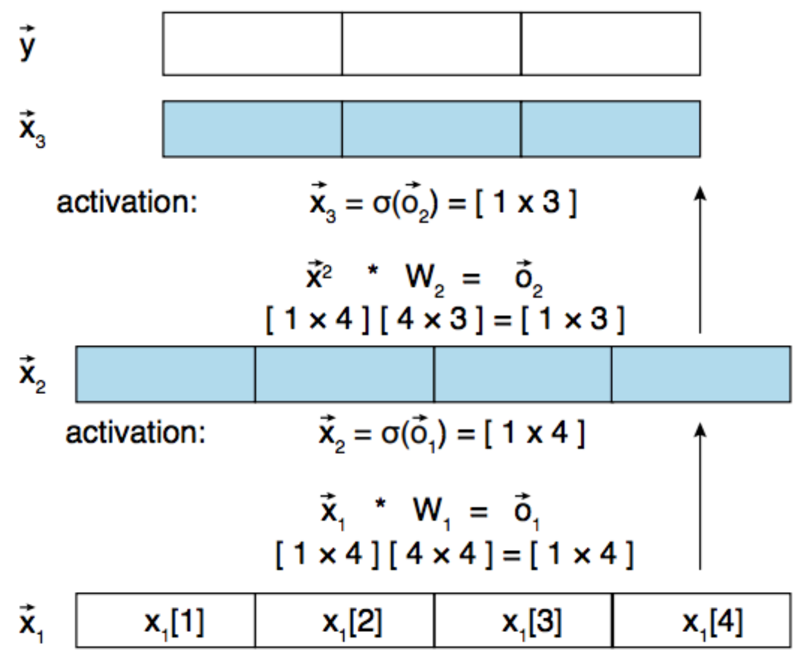

$$\delta_{output} = (y_j - x_{i,j}) \ \ x_{i,j}(1 - x_{i,j}) \qquad \leftarrow \text{previous slide}$$

$$\delta_{hidden} = \left( \sum_{n \in nodes} \delta_n w_{n,j} \right) \ \ x_{i,j}(1 - x_{i,j})$$

# Feedforward




↔



# Ahora les puedo contar este chiste…

**Basic Math**

$$\begin{pmatrix} A_{11} & A_{12} & \cdots & A_{1m} \\ A_{21} & A_{22} & \cdots & A_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ A_{n1} & A_{n2} & \cdots & A_{nm} \end{pmatrix}$$

**Dangerous Artificial Intelligence**

$$\begin{pmatrix} A_{11} & A_{12} & \cdots & A_{1m} \\ A_{21} & A_{22} & \cdots & A_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ A_{n1} & A_{n2} & \cdots & A_{nm} \end{pmatrix} * \begin{pmatrix} A_{11} & A_{12} & \cdots & A_{1m} \\ A_{21} & A_{22} & \cdots & A_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ A_{n1} & A_{n2} & \cdots & A_{nm} \end{pmatrix} * \begin{pmatrix} A_{11} & A_{12} & \cdots & A_{1m} \\ A_{21} & A_{22} & \cdots & A_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ A_{n1} & A_{n2} & \cdots & A_{nm} \end{pmatrix} * \begin{pmatrix} A_{11} & A_{12} & \cdots & A_{1m} \\ A_{21} & A_{22} & \cdots & A_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ A_{n1} & A_{n2} & \cdots & A_{nm} \end{pmatrix}$$

# Ahora les puedo contar este chiste…

**Basic Math**

$$\begin{pmatrix} A_{11} & A_{12} & \cdots & A_{1m} \\ A_{21} & A_{22} & \cdots & A_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ A_{n1} & A_{n2} & \cdots & A_{nm} \end{pmatrix}$$

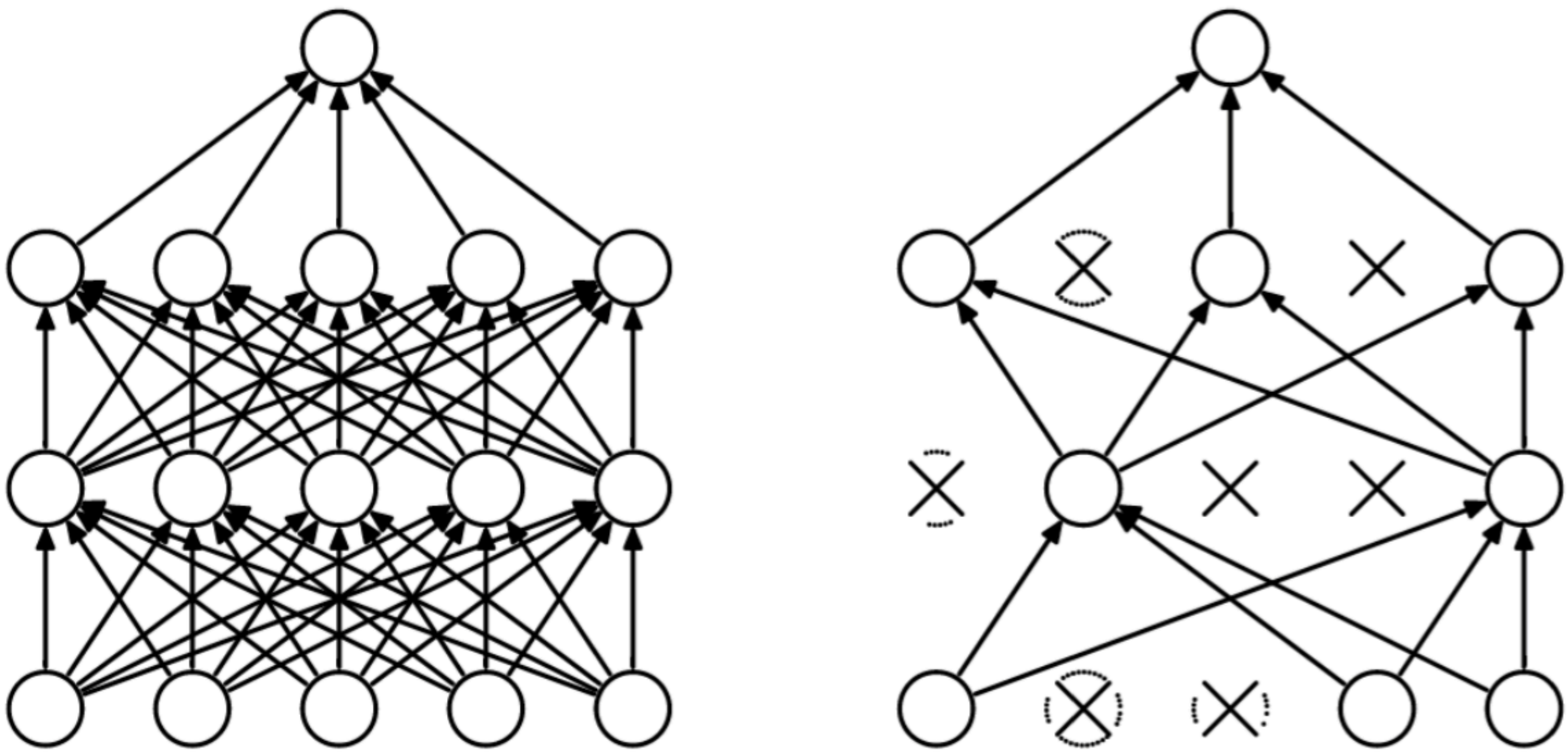
**Dangerous Artificial Intelligence**

$$\begin{pmatrix} A_{11} & A_{12} & \cdots & A_{1m} \\ A_{21} & A_{22} & \cdots & A_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ A_{n1} & A_{n2} & \cdots & A_{nm} \end{pmatrix} * \begin{pmatrix} A_{11} & A_{12} & \cdots & A_{1m} \\ A_{21} & A_{22} & \cdots & A_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ A_{n1} & A_{n2} & \cdots & A_{nm} \end{pmatrix} * \begin{pmatrix} A_{11} & A_{12} & \cdots & A_{1m} \\ A_{21} & A_{22} & \cdots & A_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ A_{n1} & A_{n2} & \cdots & A_{nm} \end{pmatrix} * \begin{pmatrix} A_{11} & A_{12} & \cdots & A_{1m} \\ A_{21} & A_{22} & \cdots & A_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ A_{n1} & A_{n2} & \cdots & A_{nm} \end{pmatrix}$$

# Otras Arquitecturas y F(x)s

- Dropout
- RLUs
- Autoencoders
- CNN
- RNN

# Dropout

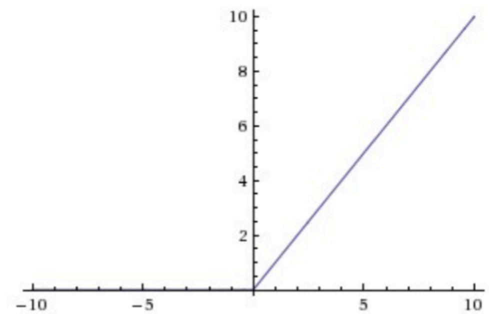


(a) Standard Neural Net

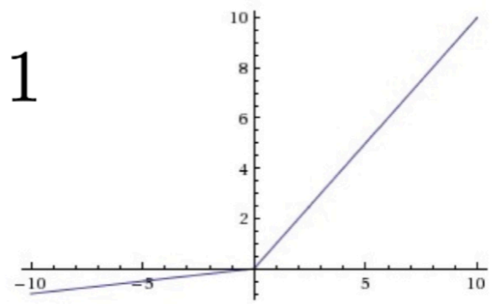
(b) After applying dropout.

# Nuevas Rectified Linear Units

$$\sigma(x) = max(0, x)$$

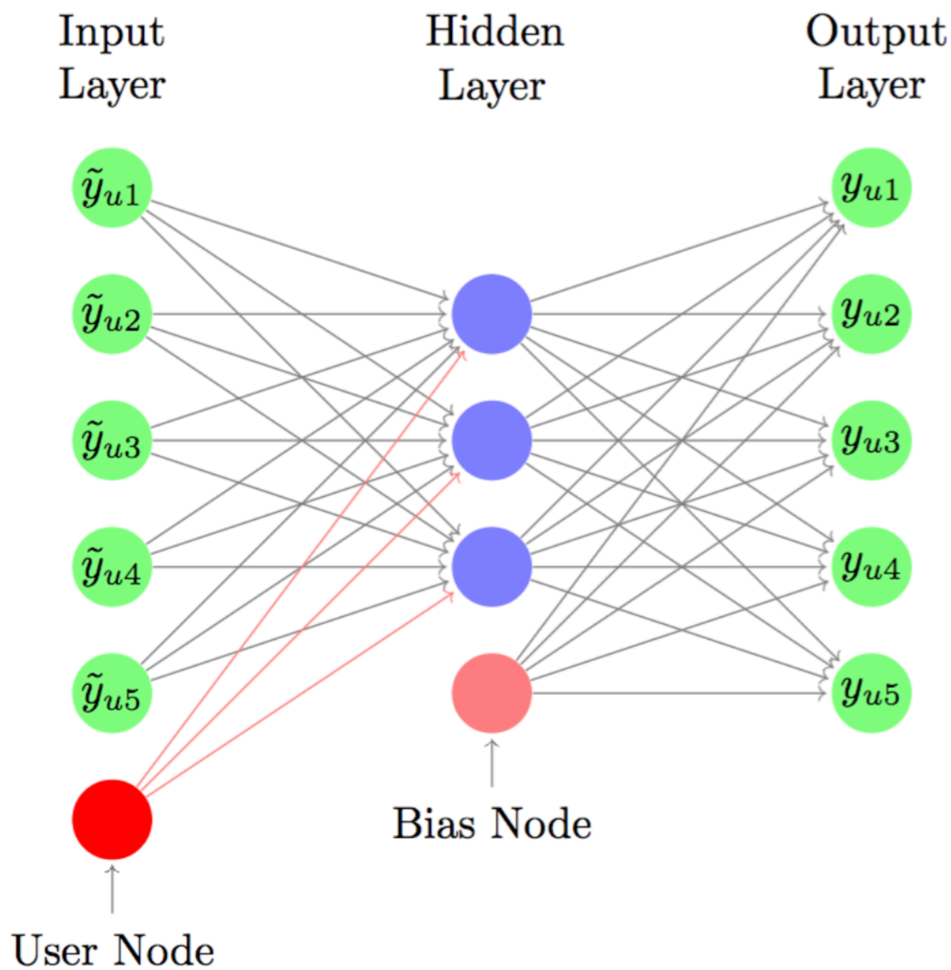



$$\sigma(x) = max(\alpha x, x) \quad \alpha < 1$$



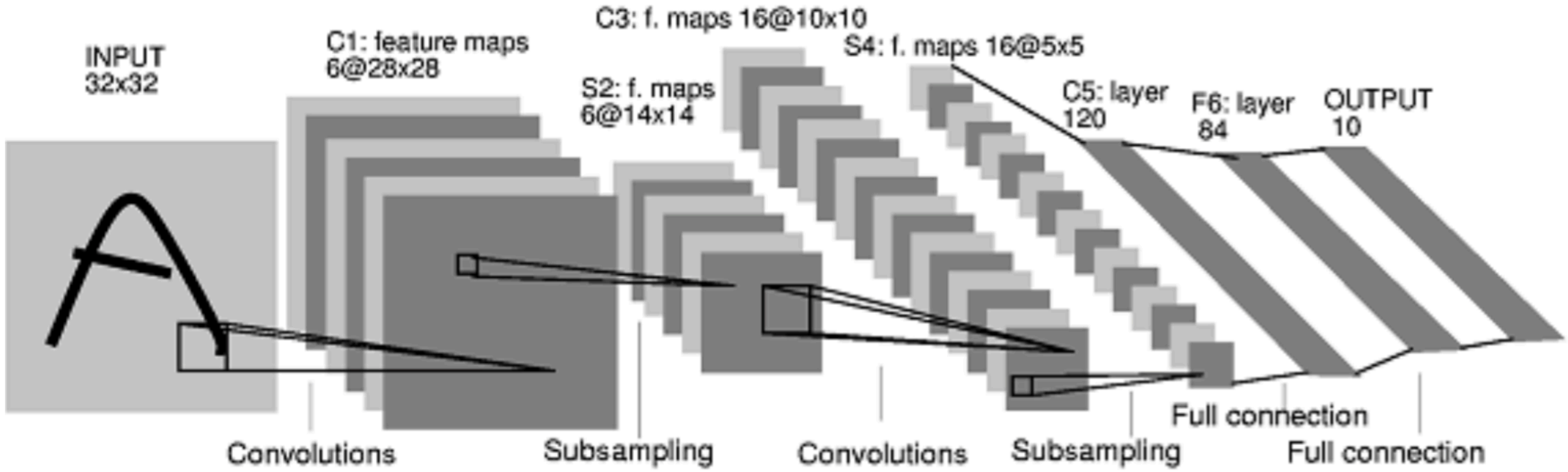

| **Neuron** | **MNIST** | **CIFAR10** | **NISTP** | **NORB** |
|---|---|---|---|---|
| ***Without* unsupervised pre-training** | | | | |
| Rectifier | **1.43%** | **50.86%** | **32.64%** | **16.40%** |
| Tanh | 1.57% | 52.62% | 36.46% | 19.29% |
| Softplus | 1.77% | 53.20% | 35.48% | 17.68% |

# Autoencoders



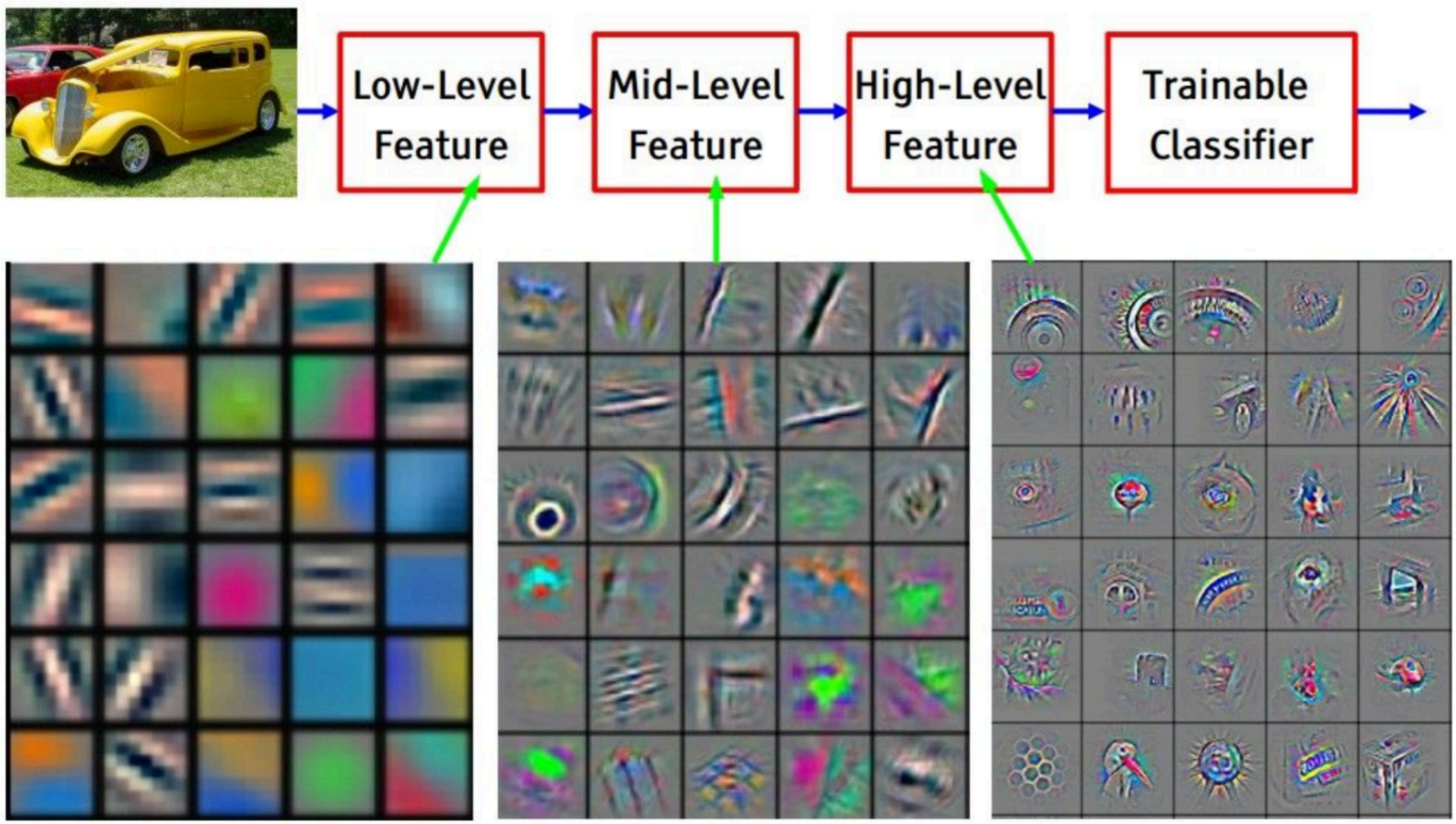

Collaborative Denoising Auto-Encoders for Top-N Recommender Systems Wu et.al. WSDM 2016

# CNN



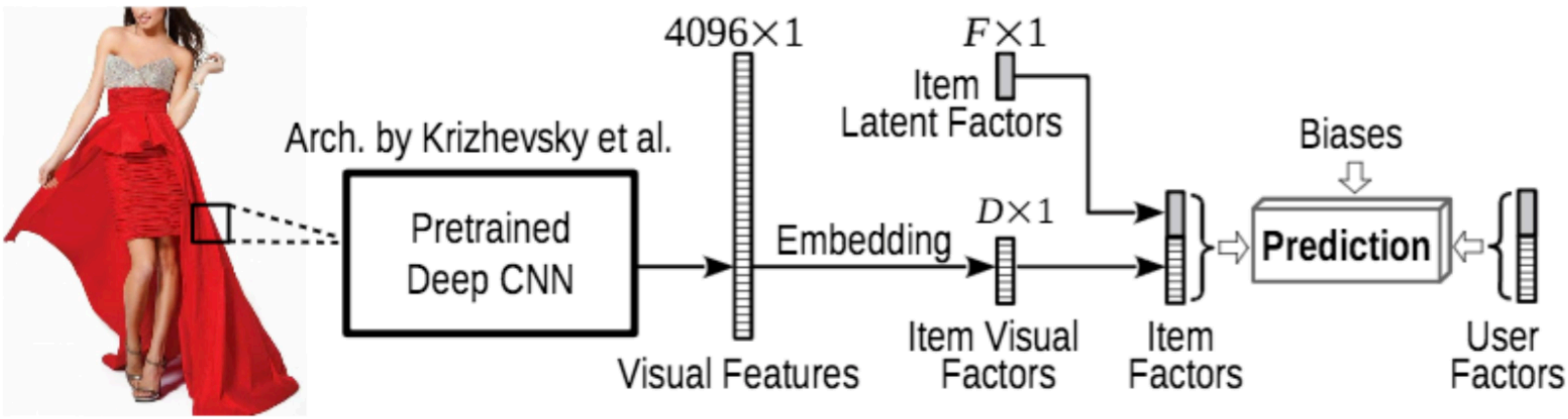
INPUT
32x32
C1: feature maps
6@28x28
S2: f. maps
6@14x14
C3: f. maps 16@10x10
S4: f. maps 16@5x5
C5: layer
120
F6: layer
84
OUTPUT
10
Convolutions
Subsampling
Convolutions
Subsampling
Full connection
Full connection

# CNN



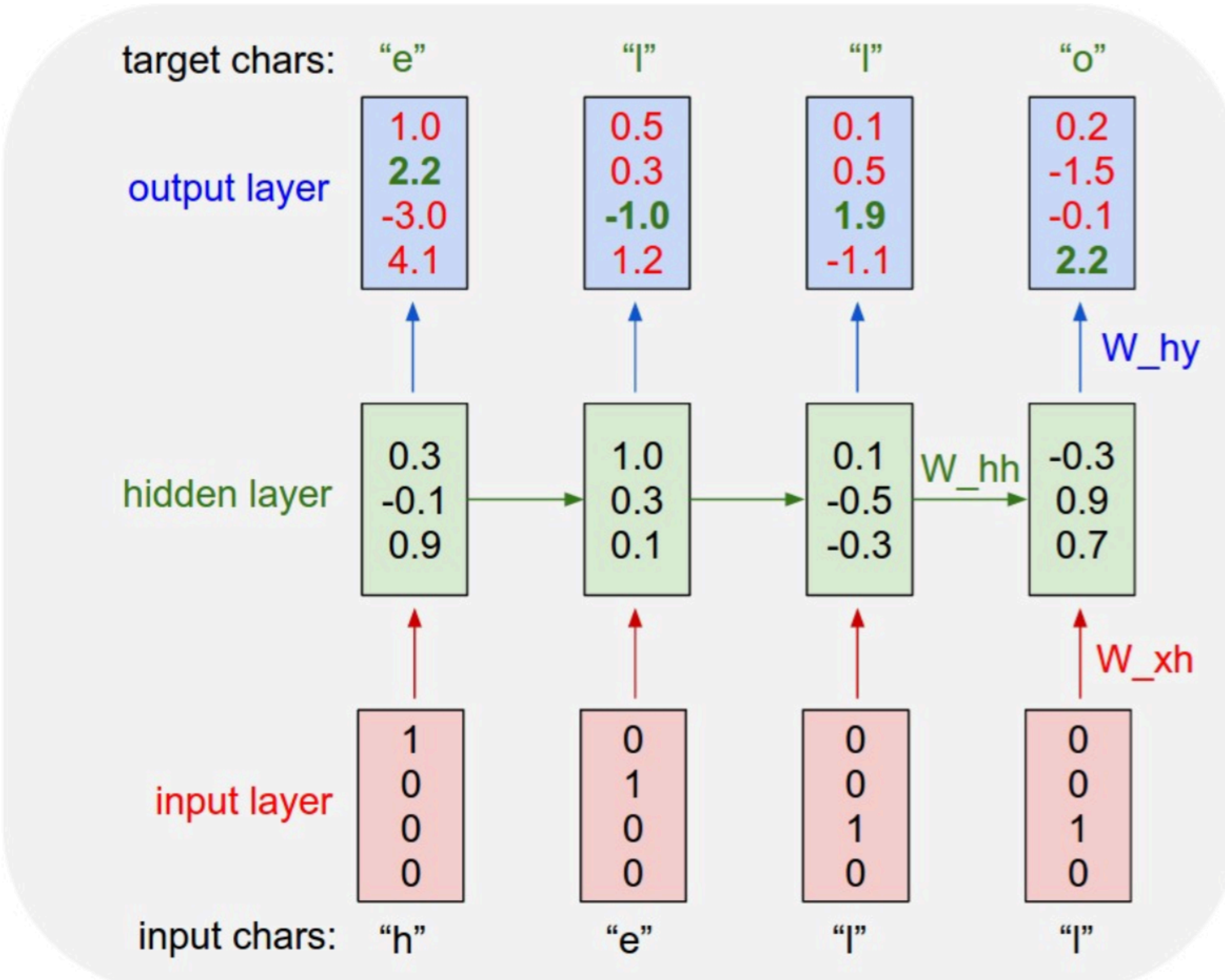

Feature visualization of convolutional net trained on ImageNet from [Zeiler & Fergus 2013]

# CNN para Filtrado Colaborativo




VBPR: Visual Bayesian Personalized Ranking from Implicit Feedback He, etl AAAI 2015

# RNN


target chars:
"e"
"l"
"l"
"o"
output layer
1.0
2.2
-3.0
4.1
0.5
0.3
-1.0
1.2
0.1
0.5
1.9
-1.1
0.2
-1.5
-0.1
2.2
W_hy
hidden layer
0.3
-0.1
0.9
1.0
0.3
0.1
0.1
-0.5
-0.3
W_hh
-0.3
0.9
0.7
W_xh
input layer
1
0
0
0
0
1
0
0
0
0
1
0
0
0
1
0
input chars:
"h"
"e"
"l"
"l"

# Question Answering via RNN

Figure 3: An example story with questions correctly answered by a MemNN. The MemNN was trained on the simulation described in Section 5.2 and had never seen many of these words before, e.g., Bilbo, Frodo and Gollum.

Bilbo travelled to the cave. Gollum dropped the ring there. Bilbo took the ring.
Bilbo went back to the Shire. Bilbo left the ring there. Frodo got the ring.
Frodo journeyed to Mount-Doom. Frodo dropped the ring there. Sauron died.
Frodo went back to the Shire. Bilbo travelled to the Grey-havens. The End.
Where is the ring? A: Mount-Doom
Where is Bilbo now? A: Grey-havens
Where is Frodo now? A: Shire

# Herramientas

- TensorFlow: Python Library (sponsored by Google)
- Theano: Python Library (descontinued)
- Pytorch: (sponsored by Facebook)
- Keras: High Level Python Library (Theano &TF as low-level library)
- MXNET: R, Python, Julia

# ImageNet:
# Crowdsourcing a Large Dataset of Image Labels
# (and how this helps our research)

# Datasets in Computer Vision

- 1996: faces and cars 14,000 images of 10,000 people
- 1998: MNIST 70,000 images of handwritten digits
- 2004: Caltech 101, 9,146 images of 101 categories
- 2005: PASCAL VOC 20,000 images with 20 classes

# Datasets in Computer Vision

- Imagenet: Presented in 2009 at CVPR

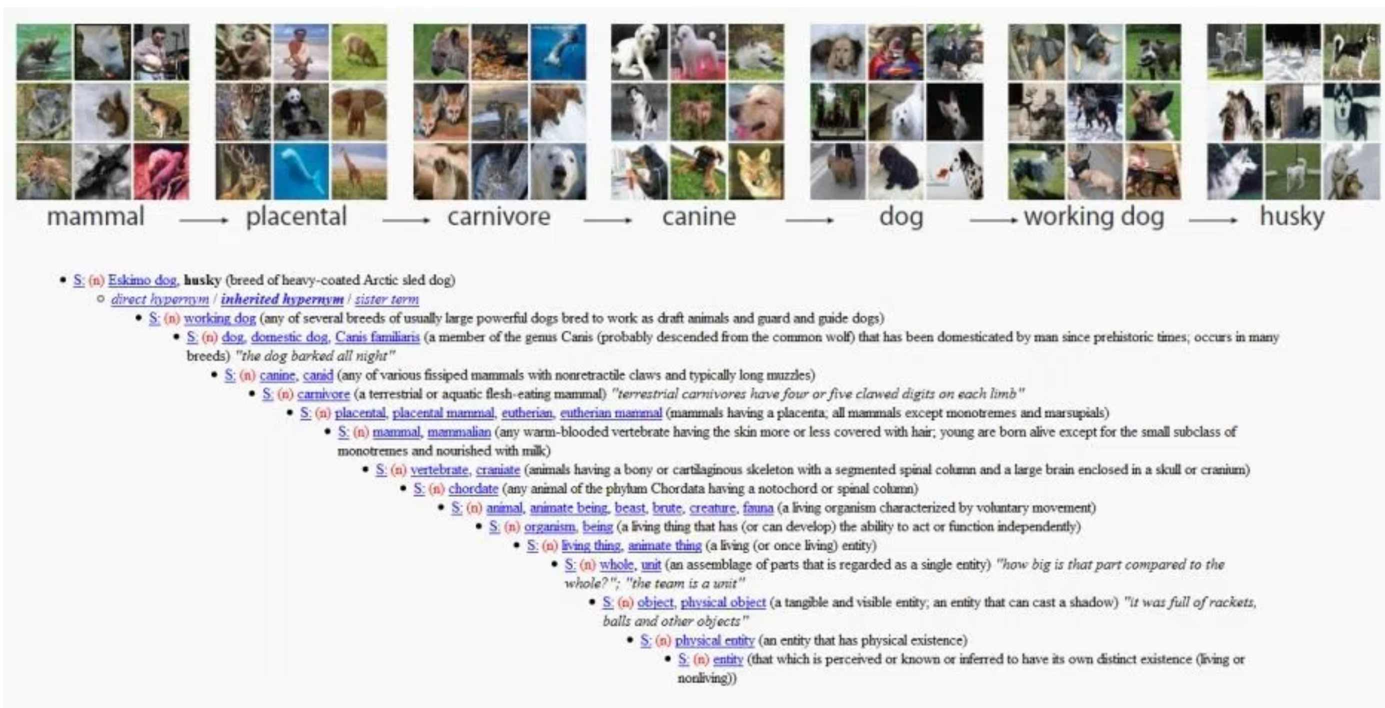
- Crowdsourced

- 14,197,122 images
- 21,841 categories (non-empty synsets)

- Categories based on WordNet taxonomy

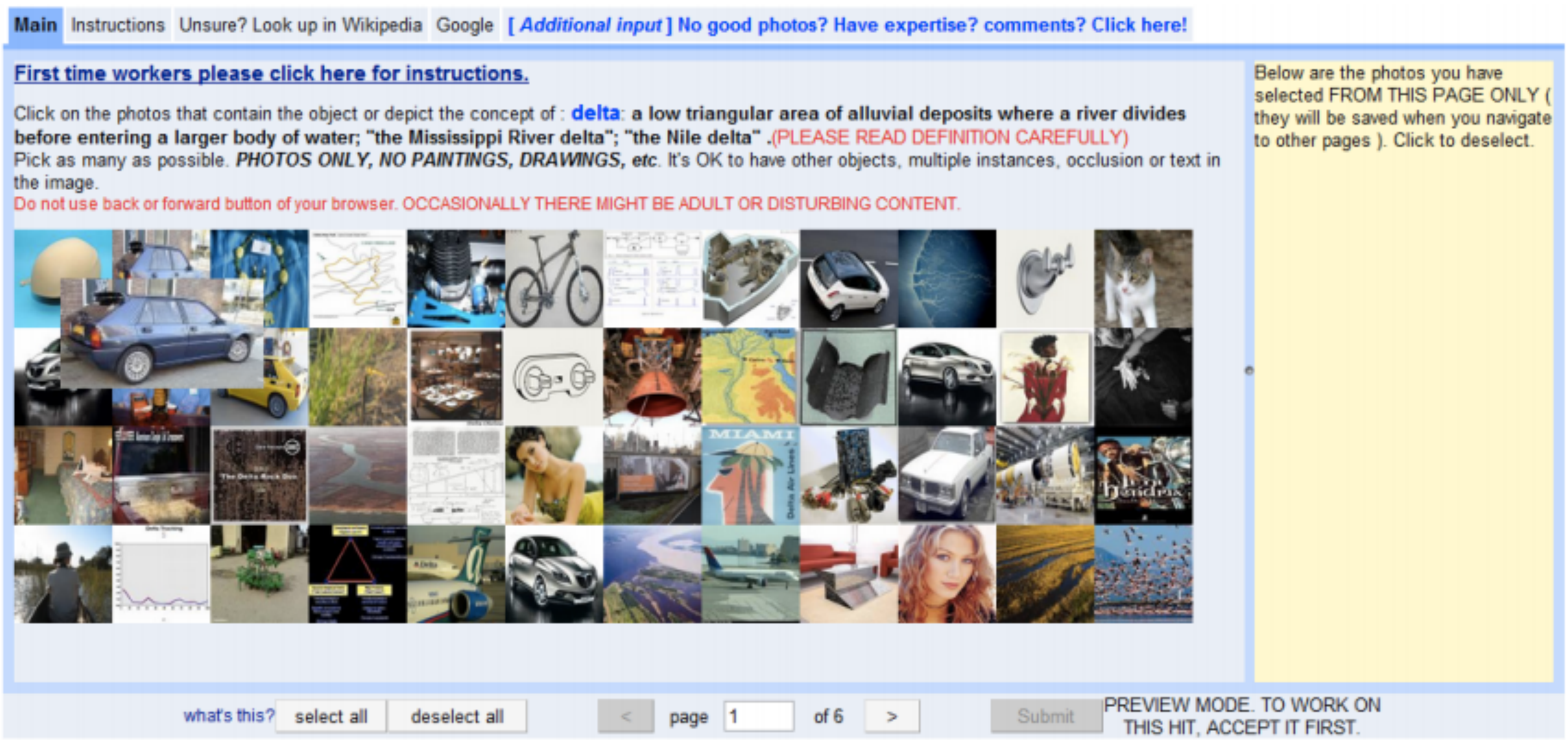
# WordNet

- Wordnet: Miller's project started in 1980 at Princeton, a hierarchy for the English language
- Prof. Fei-Fei Li (then at UIUC and then Princeton), worked on filling WordNet with many images.



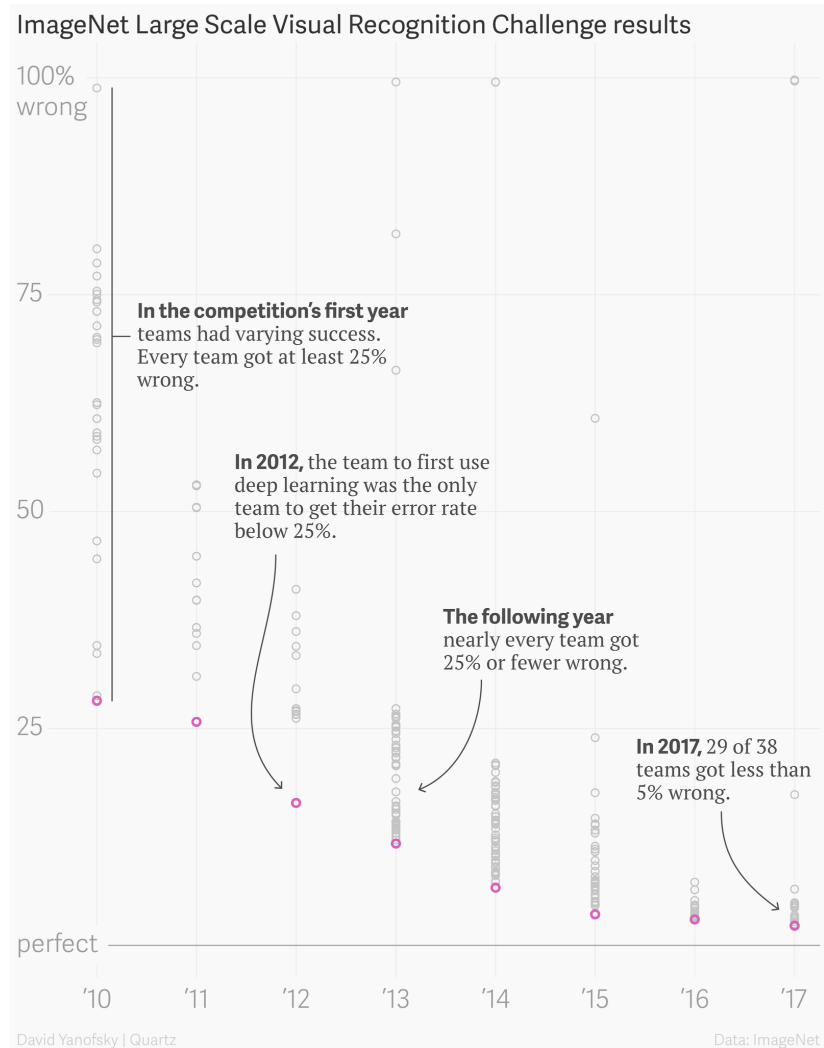
# Crowdsourced

- Amazon Mechanical Turk
- It took 2.5 years to complete. Originally 3.2 million images in 5,247 categories (mammal, vehicle, etc.)



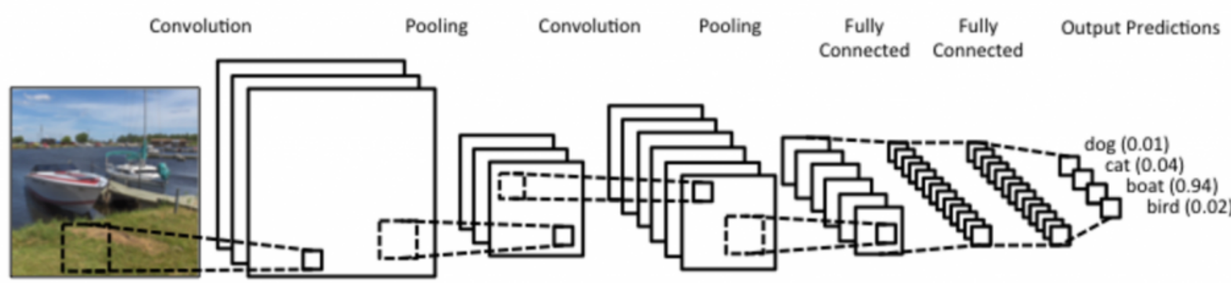
# ImageNet Challenge

- The dataset was used to set a competition for image classification: from 2010 on.
- In 2012 a team used deep learning, got error rate below 25% (Hinton et al.), 10.8 point margin, 41% better than next best.



# Transfer Learning

- 2012 model was called AlexNet: a Convolutional Neural Network



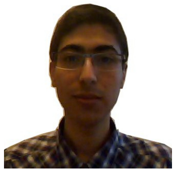

- The features learned (fc6, fc7) have been used in other tasks succesfully, **allowing to transfer the learning to other tasks**.

# Transfer Big Data Learning to Smaller Artwork Recommender System

Pablo Messina & Vicente Dominguez, Master Students
Denis Parra, Assistant Professor
CS Department
School of Engineering
Pontificia Universidad Católica de Chile

# All people involved (apart from me)

- Pablo Messina & Vicente Dominguez




- Christoph Trattner, Modul U., Austria

- Alvaro Soto & Domingo Mery, PUC Chile


# Artwork Recommendation

- Online artwork market: Growing since 2008, despite global crises!
  - In 2011, art received $11.57 billion in total global annual revenue, over $2 billion versus 2010 (*forbes)

- Previous recommendation projects date for as long as 2007, such as the CHIP project to recommend paintings from Rijksmuseum.

- Little use of recent advances in Deep Neural Networks for Computer Vision.

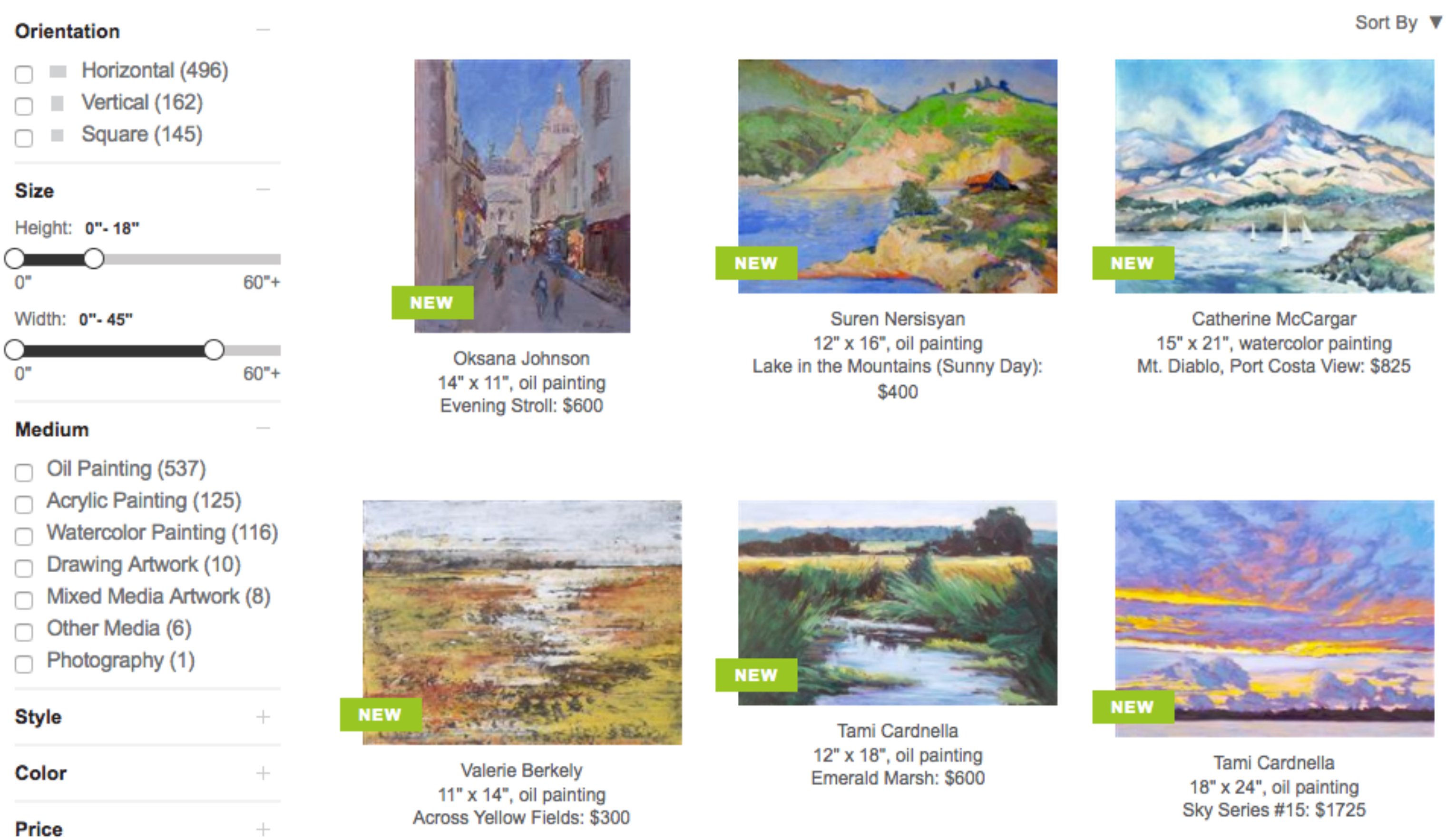
[forbes] The World's Strongest Economy? the Global Art Market. https://www.forbes.com/sites/abigailesman/2012/02/29/the- worlds- strongest- economy- the- global- art- market/ (2012)

# Our research

- Art catalogues rely heavily con manual curation.
- Do manually-curated features work better than automatically obtained features (visual) ?
- Among the visual approaches, do DNN features perform better than EVF?
  - DNN : Deep Neural Networks
  - EVF : Explicit Visual Features

# Data: UGallery

- Online Artwork Store, based on CA, USA.
- Mostly sales one-of-a-kind physical artwork.



# UGallery Data: Metadata

| Attribute | Type | Attribute Values |
|---|---|---|
| Color | Nominal | B&W, Beige, Black, Blue, Brown, Dark Blue, Dark Green, Dark Red, Green, Grey, Orange, Pink, Purple, Red, Turquoise, Violet, White, Yellow |
| Subject | Nominal | Animals, Architecture, Cuisine, Fantasy, Fashion, Flora, Landscape, Nature, Nudes, People, Religion, Seascape, Sports, Still Life, Travel, Western |
| Style | Nominal | Abstract, Classical, Expressionism, Impressionism, Minimalism, Modern , Non-representational, Pop, Primitive, Realism, Representational, Street Art, Street Photography, Surrealism, Vintage |
| Medium | Nominal | Acrylic Painting, Ceramic Artwork, Chalk Drawing, Charcoal Drawing, Colored Pencil, Digital Printmaking, Drawing Artwork, Encaustic Artwork, Gouache Painting, Ink Artwork, Marker Artwork, Mixed Media Artwork, Oil Painting, Other Media, Pastel Artwork, Pencil Drawing, Photography, Printmaking, Sculpture, Watercolor |
| Energy | Ordinal | Calm, Neutral, Energetic |
| Seriousness | Ordinal | Playful, Neutral, Serious |
| Warmness | Ordinal | Warm, Neutral, Cool |
| Purpose | Ordinal | Decorative, Neutral, Thought-Provoking |
| Complexity | Ordinal | Simple, Neutral, Complex |
| Formality | Ordinal | Formal, Neutral, Informal |
| Age Perception | Ordinal | Young, Neutral, Old |

# Data II

- 1,371 users / 3,940 items / 2,846 transactions

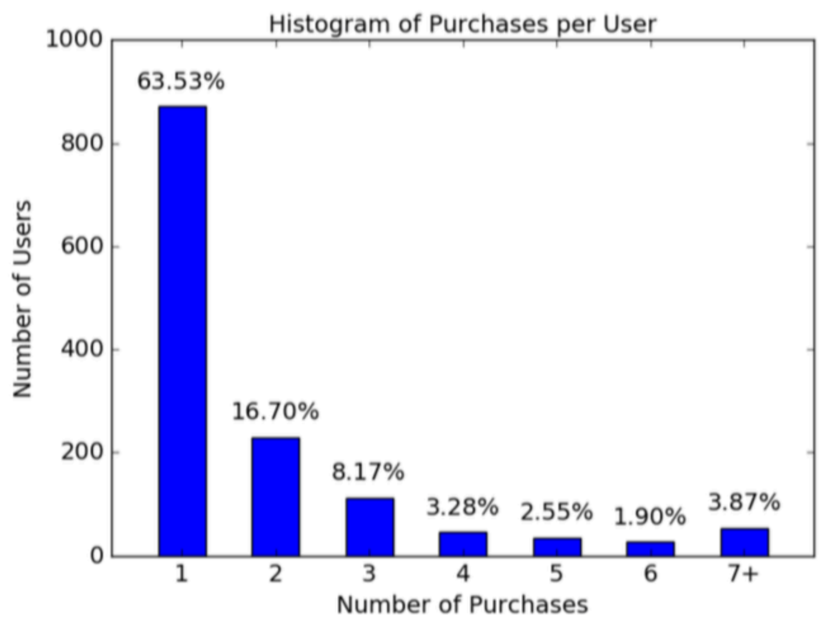


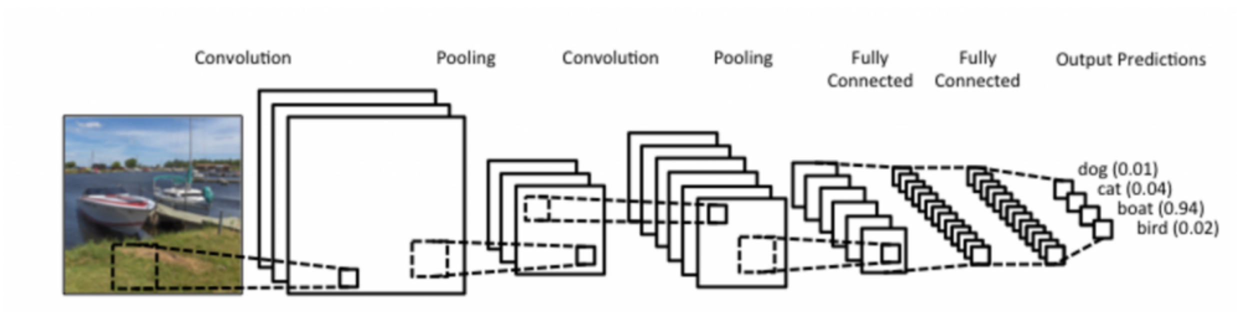

**Figure 2: Distribution of purchases per user. It resembles the typical skewed behavior in online websites.**

**Table 1: Statistics of attributes present and absent for each artwork in *UGallery* dataset.**

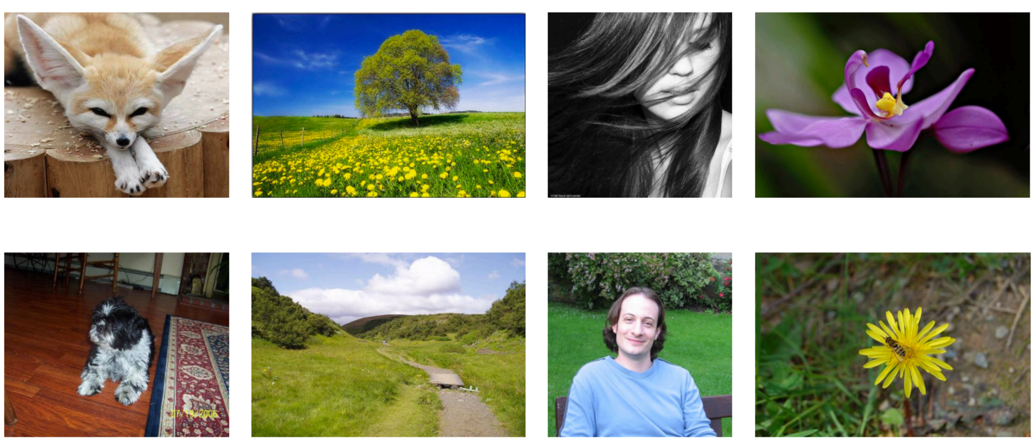
| | color | style | subject | mood | medium |
|---|---|---|---|---|---|
| present | 3,391 | 646 | 578 | 1,550 | 3,490 |
| absent | 99 | 2,844 | 2,912 | 1,940 | 0 |

# Visual Features

- (DNN) Deep Neural Networks

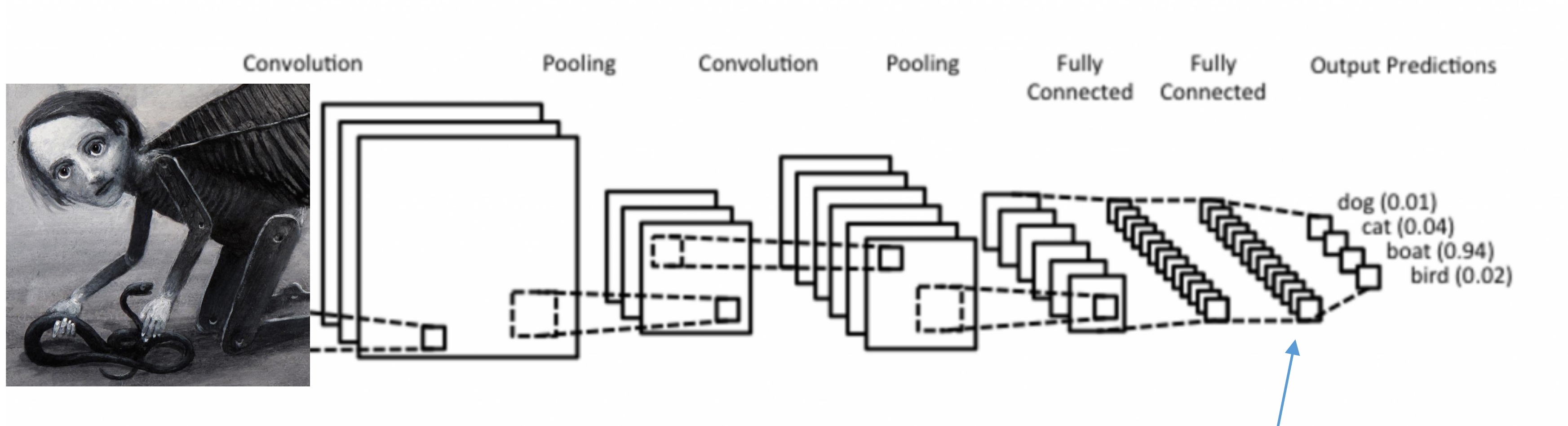



- (EVF) Attractiveness-based



# UGallery Data: DNN Visual Features

- AlexNet network, pre-trained with 1,000 classes (ImageNet dataset)




Ugallery Picture

Image feature vector with 4,096 dimensions

# UGallery Data: Visual Features

- Average brightness,
- Saturation,
- Sharpness,
- Entropy,
- RGB-contrast,
- Colorfulness,
- Naturalness

*Jose San Pedro and Stefan Siersdorfer. 2009.*

*Ranking and Classifying Attractiveness of Photos in Folksonomies.*

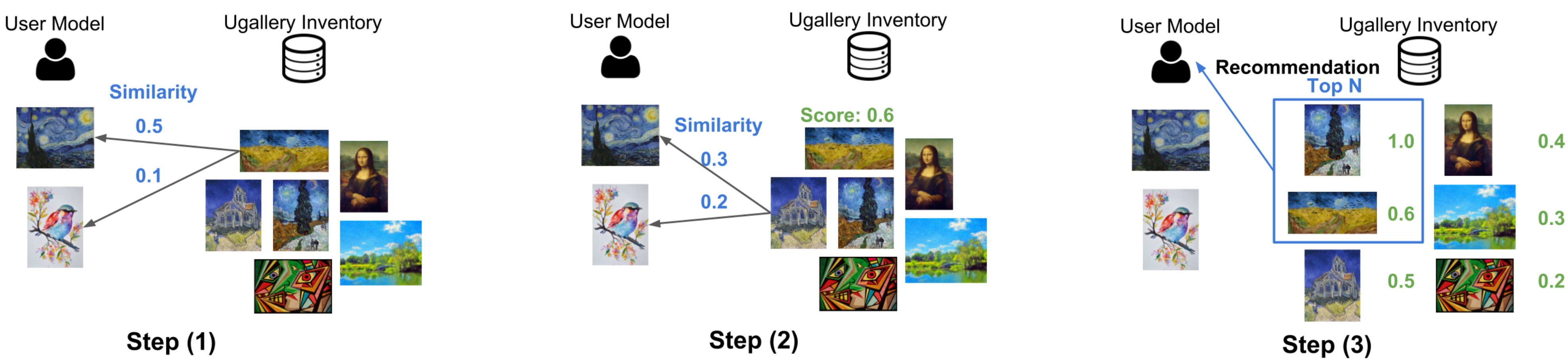
*In Proceedings of the 18th International Conference on World Wide Web (WWW '09).*

# Ugallery: All Features

- Most Popular Attribute Value (MPAV): A Non-personalized metadata-based method,
- Personalized MPAV (PMPAV): Personalized version of MPAV.
- Favorite Artist (FA): Analog to PMPAV, but recommends artworks based on the user's favorite artist.
- DNN.
- EVF.
- Price Based (Price).
- Hybrid Recommendation (Hybrid): A hybrid recommender based on a combination of metadata attributes and visual features.

# Ugallery: Making Recommendations

- Scoring items based on cosine similarity between user model and item model:




$$sim(V_i, V_j) = cos(V_i, V_j) = \frac{V_i \cdot V_j}{||V_i||\,||V_j||}$$

# Evaluation: Based on Transactions

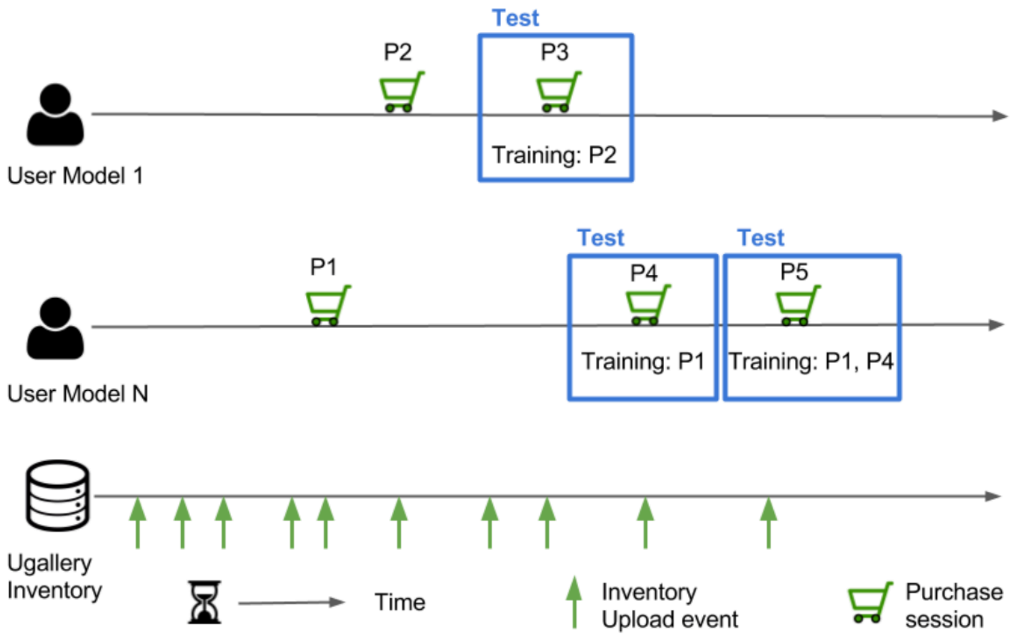



**Figure 3: Evaluation procedure. Each surrounding box represents a test, where we predict the items of the purchase session. In the figure, we predict which artworks bought User 1 in purchase P3. 'Training:P2' means we used items from purchase session P2 to train the model.**

# Results I: Metadata Personalization

**Table 3: Recall@k, Diversity, and Coverage for MPAV and PMPAV by attribute. The best results are highlighted in bold.**

| Attribute | r@5 | r@10 | diversity | user cov. | sess. cov. |
|---|---|---|---|---|---|
| MPAV - Most Pop. Attr. Value | | | | | |
| Color | **.0108** | **.0223** | .6218 | **.9993** | **.9995** |
| Medium | .0095 | .0199 | .6706 | **.9993** | **.9995** |
| Style | .0083 | .0164 | **.7055** | **.9993** | **.9995** |
| Subject | .0060 | .0139 | .6881 | **.9993** | **.9995** |
| Mood | .0089 | .0177 | .5724 | **.9993** | **.9995** |
| All Combined | .0099 | .0198 | .6683 | **.9993** | **.9995** |
| PMPAV - Personalized Most Pop. Attr. Value | | | | | |
| Color | .0166 | .0351 | .6574 | .2619 | .3570 |
| Medium | .0184 | .0336 | .6774 | **.2640** | **.3593** |
| Style | **.0280** | **.0494** | .7127 | .0766 | .1168 |
| Subject | .0125 | .0236 | .6897 | .0890 | .1407 |
| Mood | .0159 | .0270 | .6967 | .1327 | .1822 |
| All Combined | .0148 | .0285 | .6863 | **.2640** | **.3593** |

# Results II : EVF

- Contrast & Entropy (+)
- Colorfulness (-)

| name | ndcg@5 | ndcg@10 | rec@5 | rec@10 | prec@5 | prec@10 |
|---|---|---|---|---|---|---|
| EVF (all features) | .0344 | .0459 | .0547 | .0885 | .0127 | .0111 |
| EVF (all, except LBP) | .0370 | .0453 | .0585 | .0826 | .0152 | .0109 |
| EVF (LBP) | .0292 | .0388 | .0431 | .0715 | .0103 | .0085 |
| EVF (brightness) | .0048 | .0083 | .0080 | .0186 | .0035 | .0031 |
| EVF (colorfulness) | .0020 | .0052 | .0033 | .0126 | .0008 | .0017 |
| EVF (contrast) | .0079 | .0102 | .0149 | .0220 | .0033 | .0025 |
| EVF (entropy) | .0088 | .0098 | .0110 | .0137 | .0026 | .0019 |
| EVF (naturalness) | .0062 | .0110 | .0108 | .0248 | .0026 | .0032 |
| EVF (saturation) | .0055 | .0084 | .0096 | .0187 | .0021 | .0020 |
| EVF (sharpness) | .0063 | .0085 | .0117 | .0178 | .0024 | .0021 |

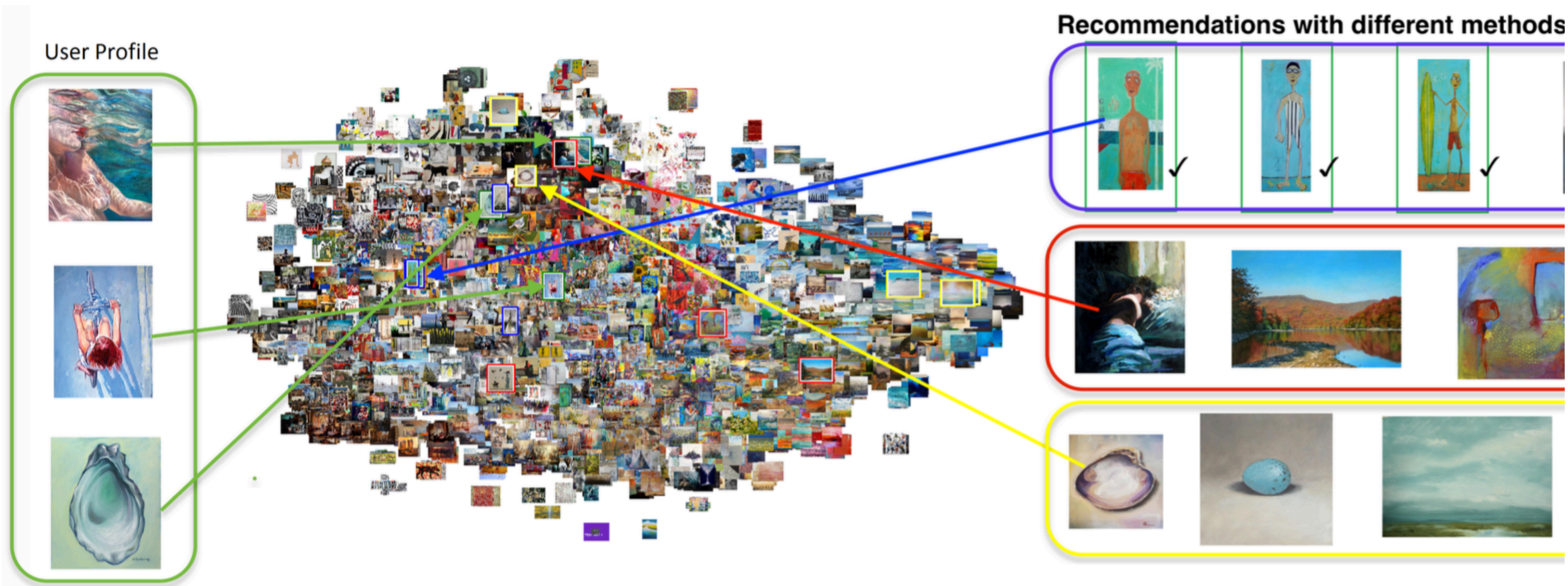
# Results III : Altogether

| Group | Method | nD@5 | nD@10 | r@5 | r@10 | p@5 | p@10 | f1@5 | f1@10 |
|---|---|---|---|---|---|---|---|---|---|
| G4★ | $Hyb_1$ (FA+DNN+EVF+Price) | **.1367**▼◆ | **.1569**▼◆■ | **.1768**▼◆ | **.2368**▼◆ | **.0425**▼◆ | **.0287**▼◆ | **.0663**▼◆ | **.0501**▼◆ |
| G4★ | $Hyb_2$ (FA+DNN) | **.1342**▼◆ | **.1526**▼◆■ | .1716▼◆ | **.2251**▼◆ | **.0420**▼◆ | **.0283**▼◆ | .0651▼◆ | **.0490**▼◆ |
| G4★ | $Hyb_3$ (FA+EVF) | .1282▼◆ | .1439▼◆ | **.1780**▼◆ | **.2252**▼◆ | **.0420**▼◆ | **.0272**▼◆ | **.0659**▼◆ | **.0474**▼◆ |
| G4★ | $Hyb_4$ (FA+Price) | **.1322**▼◆ | **.1441**▼◆ | **.1780**▼◆ | .2139▼◆ | **.0435**▼◆ | .0263▼ | **.0672**▼◆ | .0456▼ |
| G3■ | FA | .1132▼□ | .1270▼□ | .1591▼◆□ | .2039▼□ | .0385▼□ | .0252▼□ | .0596▼□ | .0436▼□ |
| G3■ | Price | .0239 | .0293 | .0366 | .0527 | .0083 | .0065 | .0131 | .0112 |
| G2◆ | DNN | .0810▼◇ | .0968▼◇ | .1052▼ | .1525▼◇ | .0269▼ | .0195▼◇ | .0415▼ | .0338▼◇ |
| G2◆ | EVF | .0370▼ | .0453▼ | .0585▼ | .0826▼ | .0152▼ | .0109▼ | .0233▼ | .0188▼ |
| G1▼ | PMPAV (Color) | .0097▽ | .0167▽ | .0166▽ | .0351▽ | .0039▽ | .0042▽ | .0061▽ | .0073▽ |
| G1▼ | MPAV (Color) | .0067 | .0106 | .0108 | .0223 | .0029 | .0029 | .0043 | .0049 |

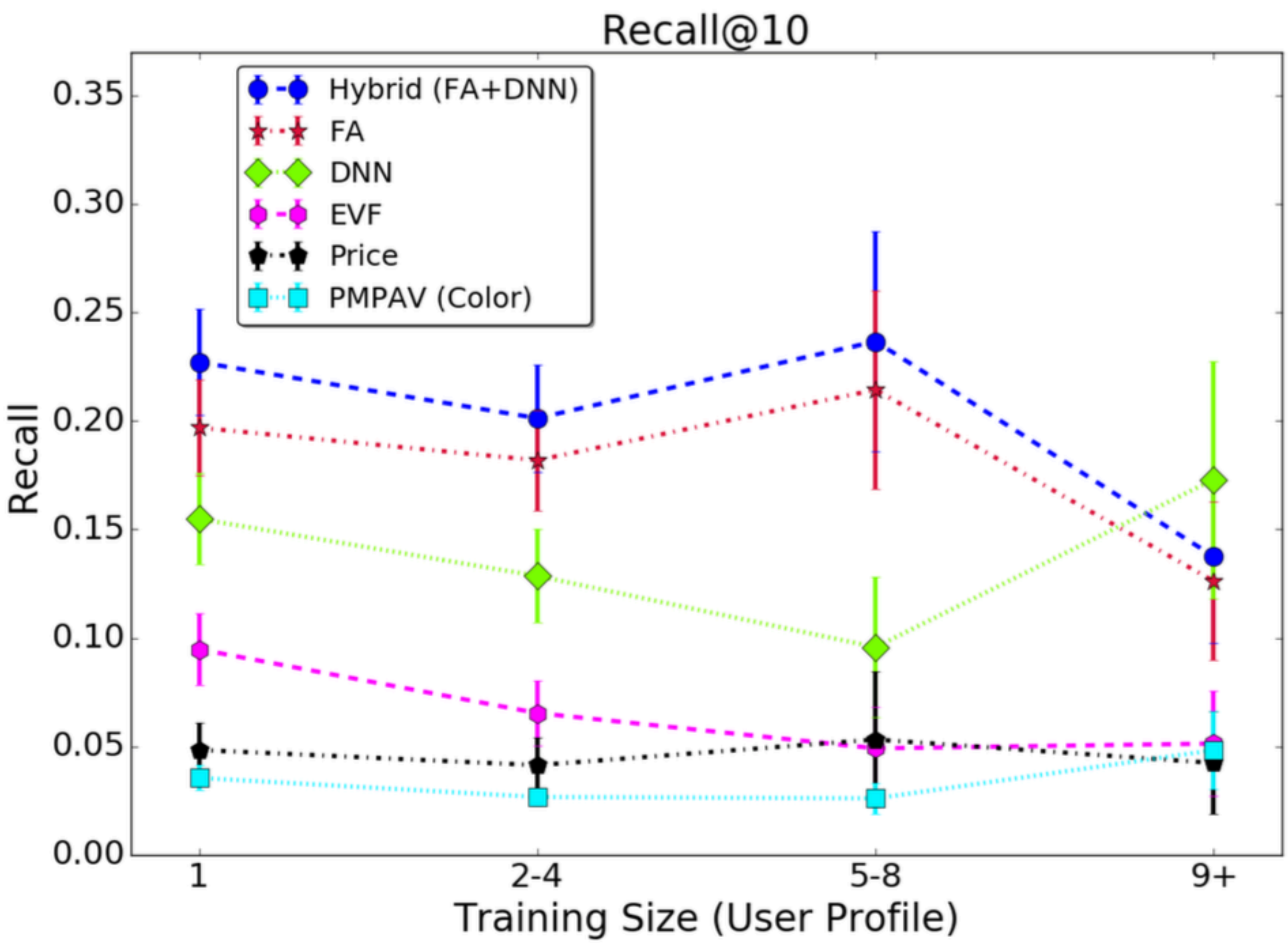
Statistical significance using multiple pairwise t-tests with Bonferroni correction, $\alpha_{bonf} = \alpha/n = 0.05/45 = .001$

# DNN Embedding

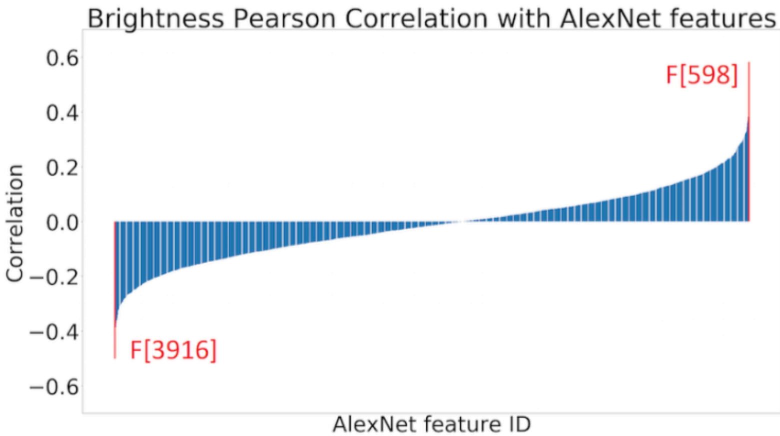
# DNN Embedding and other methods



# Discussion: Profile Size



# Explanaibility: DNN vs. EVF?




F[598] : 44.17

F[3916] : 0.0

Brightness : 231.00

F[598] : 0.0

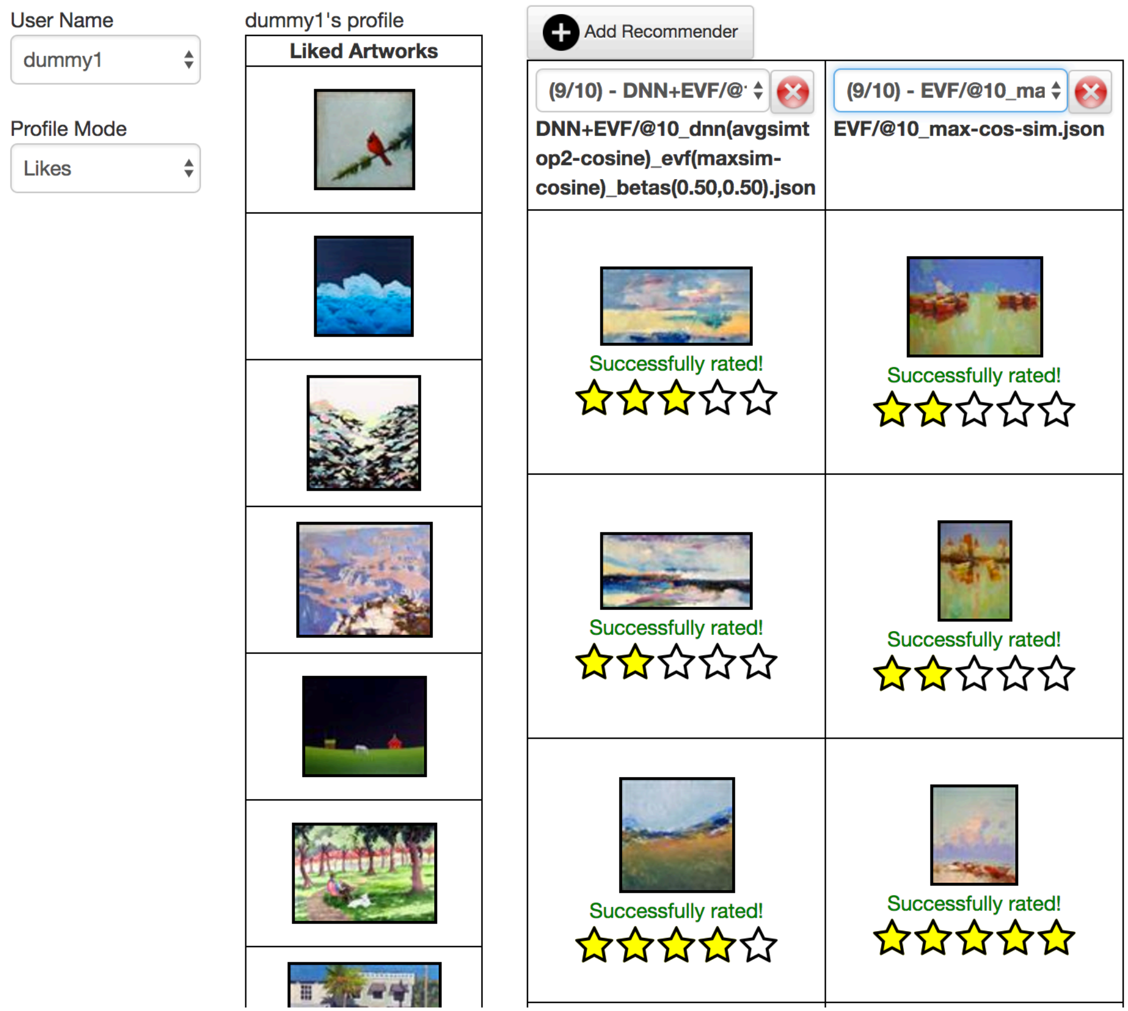
F[3916] : 26.02

Brightness : 20.68

| Visual feature | $max(corr.)$ | $index(F_{max})$ | $min(corr.)$ | $index(F_{min})$ |
|---|---|---|---|---|
| brightness | **0.5815** | 598 | **-0.5020** | 3916 |
| sharpness | 0.4682 | 4095 | -0.3668 | 1025 |
| saturation | 0.3645 | 445 | -0.5019 | 2370 |
| colorfulness | 0.3034 | 1014 | -0.3383 | 2410 |
| entropy | 0.3410 | 286 | -0.4369 | 3499 |
| contrast | 0.3941 | 469 | -0.2799 | 3761 |
| naturalness | 0.0852 | 2626 | -0.0256 | 2499 |

# Evaluation with 7 experts

- Results are consistent with our off-line analysis.
- Further research is conducted now in the intersection of explainibility and recommendations.



# Future Work

- Implementing a more sophisticated way to make the recommendations (BPR, FM + dimensionality reduction)

- Conduct a user study (with Ugallery customers).

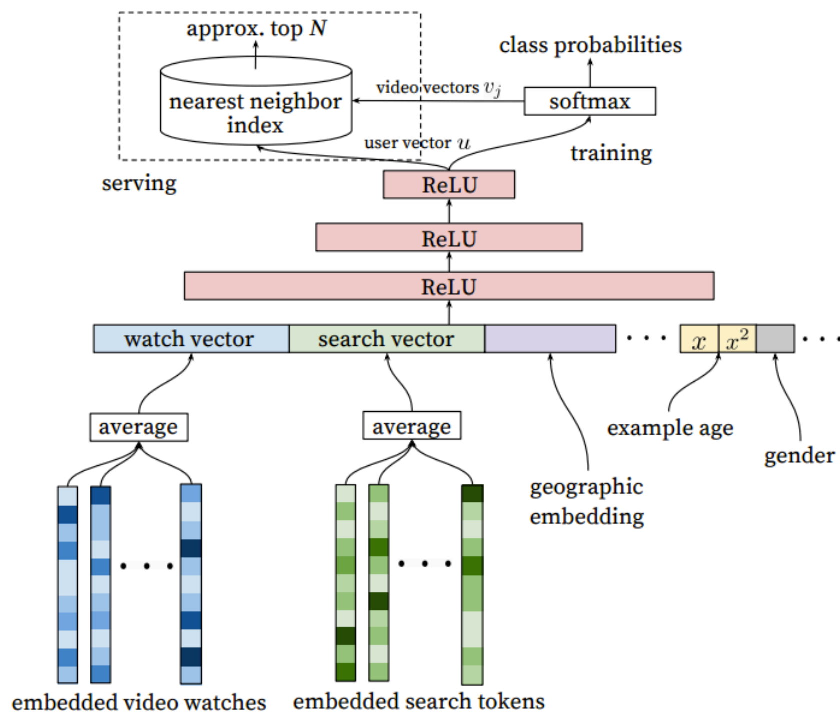
- Implementing an interface to produce the visualizations.

# Otros Papers en RecSys

- ***Ask the GRU*: Multi-task Learning for Deep Text Recommendations**. Trapit Bansal, David Belanger, and Andrew McCallum. 2016
- Este paper usa RNN (redes neuronales recurrentes) con GRUs (gated recurrent units)
- Presenta Germán Contreras este jueves

# Paper 2

- **Deep Neural Networks for YouTube Recommendations.** Paul Covington, Jay Adams, and Emre Sargin. 2016



# Paper 3

- **Meta-Prod2Vec: Product Embeddings Using Side-Information for Recommendation**. Flavian Vasile, Elena Smirnova, and Alexis Conneau. 2016.

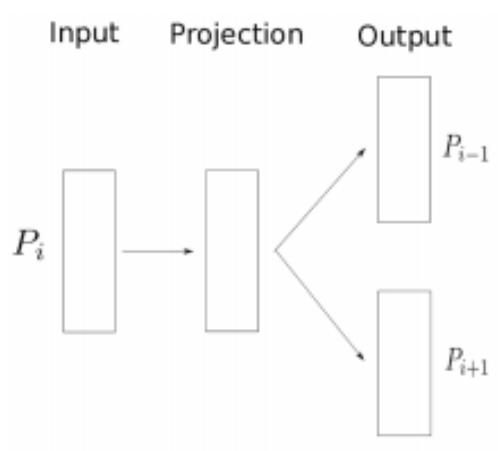



**Figure 1: Prod2Vec Neural Net Architecture.**

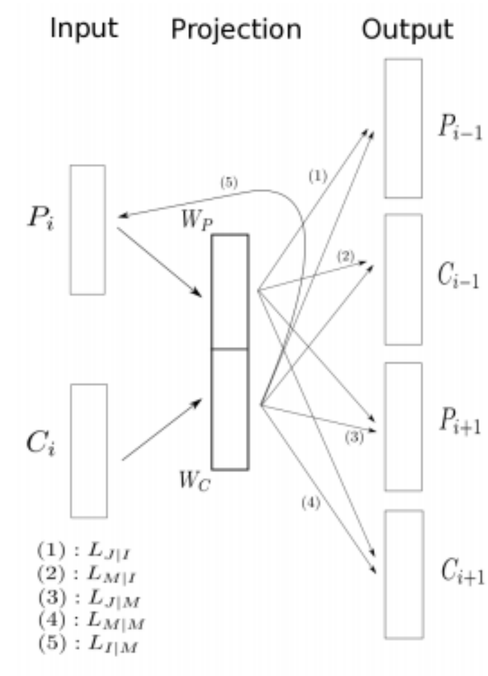



**Figure 2: Meta-Prod2Vec Neural Net Architecture.**

# Meta-Prod2Vec

- Basado en Word2Vec, donde la función objetivo del embedding se acerca a Shifted Positive PMI (SPMI)

$$PMI_{ij} = \log\left(\frac{X_{ij} \cdot |D|}{X_i X_j}\right)$$

$$SPMI_{ij} = PMI(i,j) - \log k$$

$$\begin{aligned} L_{\mathrm{P2V}} &= L_{J|I}(\theta) \\ &= \sum_{ij}(-X_{ij}^{\mathrm{POS}} \log q_{j|i}(\theta) - (X_i - X_{ij}^{\mathrm{POS}}) \log(1 - q_{j|i}(\theta)) \\ &= \sum_{ij} X_i(-p_{j|i} \log q_{j|i}(\theta) - p_{\neg j|i} \log q_{\neg j|i}(\theta)) \\ &= \sum_{i} X_i H(p_{\cdot|i}, q_{\cdot|i}(\theta)). \end{aligned}$$

- Función de pérdida Meta-Prod2Vec

$$L_{MP2V} = L_{J|I} + \lambda \times (L_{M|I} + L_{J|M} + L_{M|M} + L_{I|M})$$

# Meta-Prod2Vec

- Resultados

| Method | HR@10 | NDCG@10 | HR@20 | NDCG@20 |
|---|---|---|---|---|
| BestOf | 0.0003 (0.0002;0.0003) | 0.001 (0.001;0.001) | 0.0003 (0.0002;0.0003) | 0.002 (0.002;0.002) |
| CoCounts | 0.0248 (0.0245;0.0251) | 0.122 (0.121;0.123) | 0.0160 (0.0158;0.0161) | 0.141 (0.139;0.142) |
| Prod2Vec | 0.0170 (0.0168;0.0171) | 0.105 (0.103;0.106) | 0.0101 (0.0100;0.0102) | 0.113 (0.112;0.115) |
| Meta-Prod2Vec | 0.0191 (0.0189;0.0194) | 0.110 (0.108;0.113) | 0.0124 (0.0123;0.0126) | 0.125 (0.123;0.126) |
| Mix(Prod2Vec,CoCounts) | 0.0273 (0.027;0.0276) | 0.140 (0.139;0.141) | 0.0158 (0.0157;0.0160) | 0.152 (0.151;0.153) |
| Mix(Meta-Prod2Vec,CoCounts) | **0.0292** (0.0288;0.0297) | **0.144** (0.142;0.145) | **0.0180** (0.0178;0.0182) | **0.161** (0.160;0.162) |

**Table 1: Comparison of recommendation performance of Meta-Prod2Vec and competing models in terms of HitRate and NDCG.**

| Method | Pair freq=0 | Pair freq<3 |
|---|---|---|
| BestOf | 0.0002 | 0.0002 |
| CoCounts | 0.0000 | 0.0197 |
| Prod2Vec | 0.0003 | 0.0078 |
| Meta-Prod2Vec | **0.0013** | 0.0198 |
| Mix(Prod2Vec,CoCounts) | 0.0002 | 0.0200 |
| Mix(Meta-Prod2Vec,CoCounts) | 0.0007 | **0.0291** |

**Table 2: Recommendation accuracy (HR@20) in cold-start regime as a function of training frequency of the pair (*query item*, *next item*).**

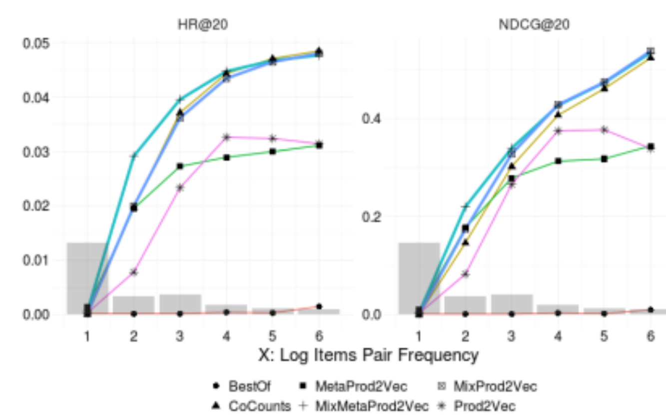



**Figure 4: Cold-start improvements on the query and next item pairs.**

# Thanks!

dparra@ing.puc.cl

# Ask the GRU…

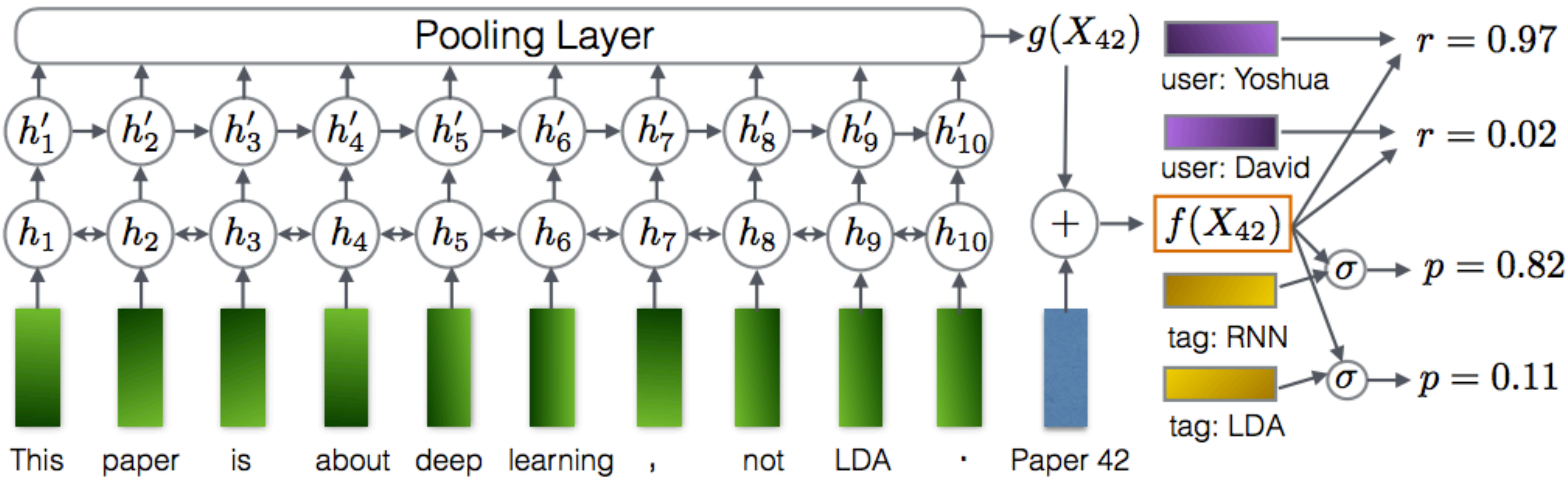



**Figure 1: Proposed architecture for text item recommendation. Rectangular boxes represent embeddings. Two layers of RNN with GRU are used, where the first layer is a bi-directional RNN. The output of all the hidden units at the second layer is pooled to produce a text encoding which is combined with an item-specific embedding to produce the final representation $f(X)$. Users and tags are also represented by embeddings, which are combined with the item representation to do tag predection and recommendation.**

# Ask the GRU …

- Citeulike-a consists of 5551 users, 16980 papers and 3629 tags with a total of 204,987 user-item likes.
- Citeulike-t [5] consists of 5219 users, 25975 papers and 4222 tags with a total of 134,860 user-item likes.
- Note Citeulike-t is much more sparse (99.90%) than Citeulike-a (99.78%).

**Table 1: % Recall@50 for all the methods (higher is better).**

| | Citeulike-a | | | Citeulike-t | | |
|---|---|---|---|---|---|---|
| | Warm Start | Cold Start | Tag Prediction | Warm Start | Cold Start | Tag Prediction |
| GRU-MTL | **38.33** | **49.76** | **60.52** | 45.60 | **51.22** | **62.32** |
| GRU | 36.87 | 46.16 | — | 42.59 | 47.59 | — |
| CTR-MTL | 35.51 | 39.87 | 48.95 | **46.82** | 34.98 | 46.66 |
| CTR | 31.10 | 39.00 | — | 40.44 | 33.74 | — |
| Embed-MTL | 36.64 | 41.71 | 60.36 | 43.02 | 38.16 | 62.29 |
| Embed | 33.95 | 38.53 | — | 37.98 | 35.85 | — |