Deep Learning en Sistemas de Recomendación

Denis Parra

IIC1005 Sistemas Recomendadores

PUC Chile

2016

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

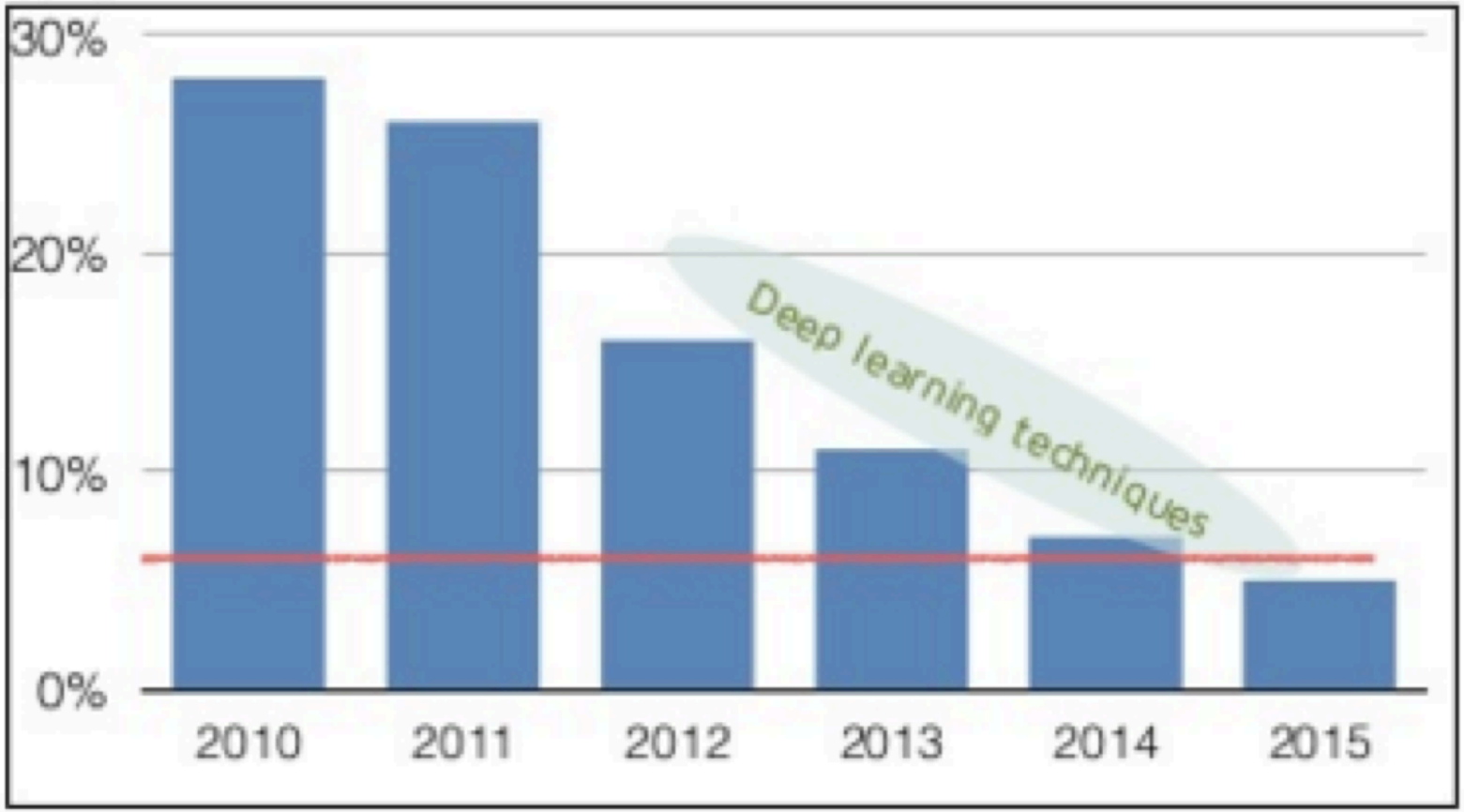
Referencias

- 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 http://www.slideshare.net/kerveros99/deep-learning-for-recommender-systems-budapest-recsys-meetup
  - Balazs Hidasi http://www.slideshare.net/balazshidasi/deep-learning-to-the-rescue-solving-long-standing-problems-of-recommender-systems

Presentaciones el Jueves

- Deep Neural Networks for YouTube Recommendations (Felipe del Río)
- Convolutional Matrix Factorization for Document Context-Aware Recommendation (Gabriel Sepulveda)
- Paper 3 (Lucas Pose)

¿Por qué Deep Learning?



ImageNet challenge error rates (red line = human performance)

¿Por qué Deep Learning?

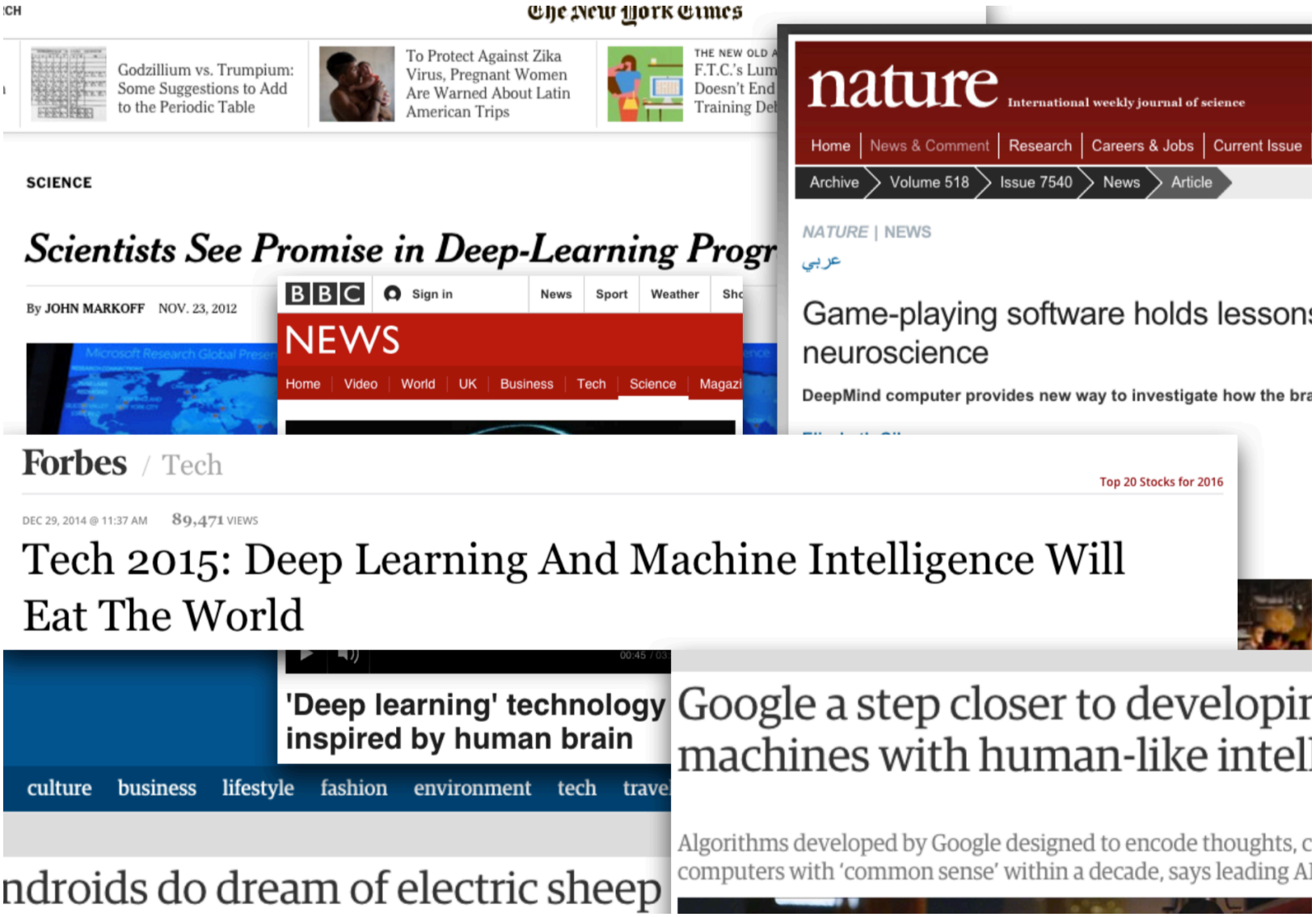
hand-crafted
Feature Extractor

"Simple" Trainable
Classifier

Trainable
Feature Extractor

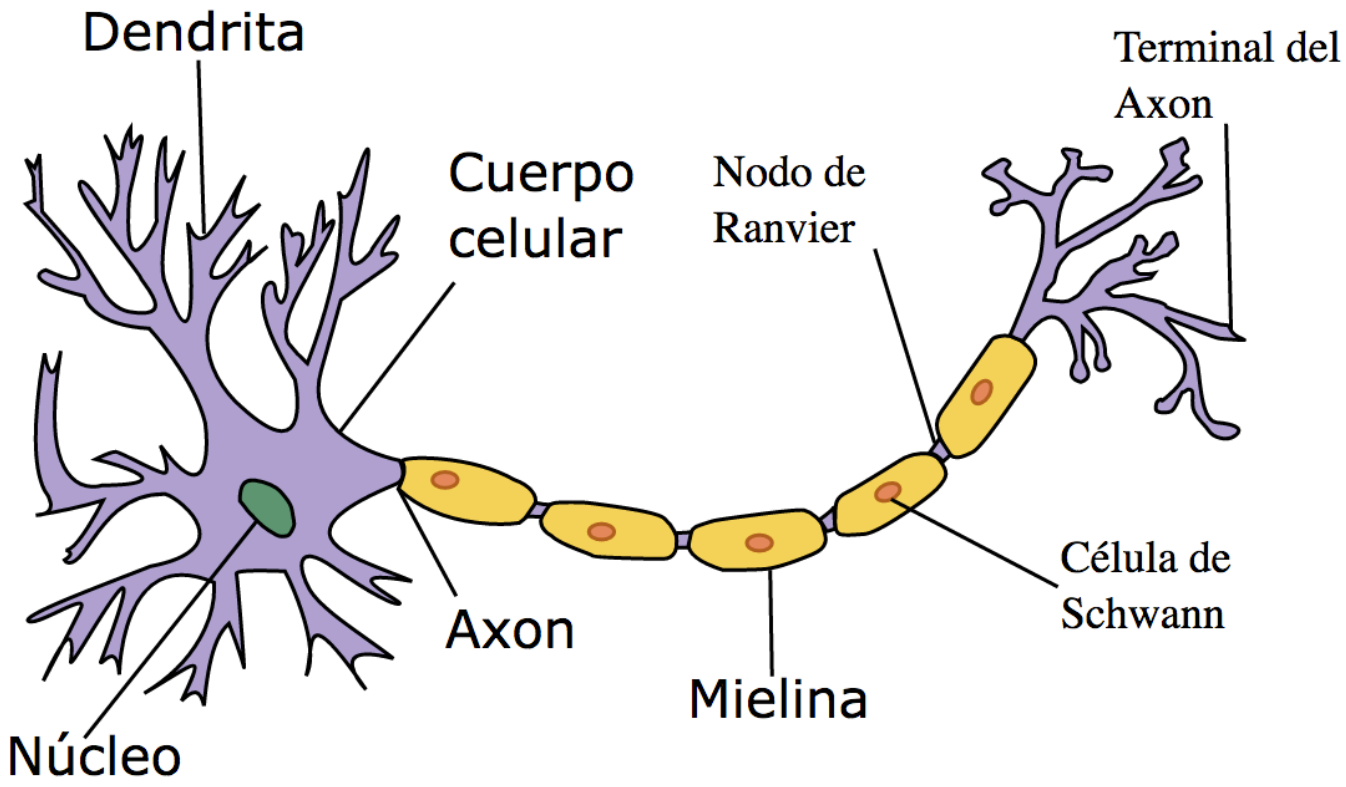
Trainable
Classifier

¿Por qué Deep Learning?

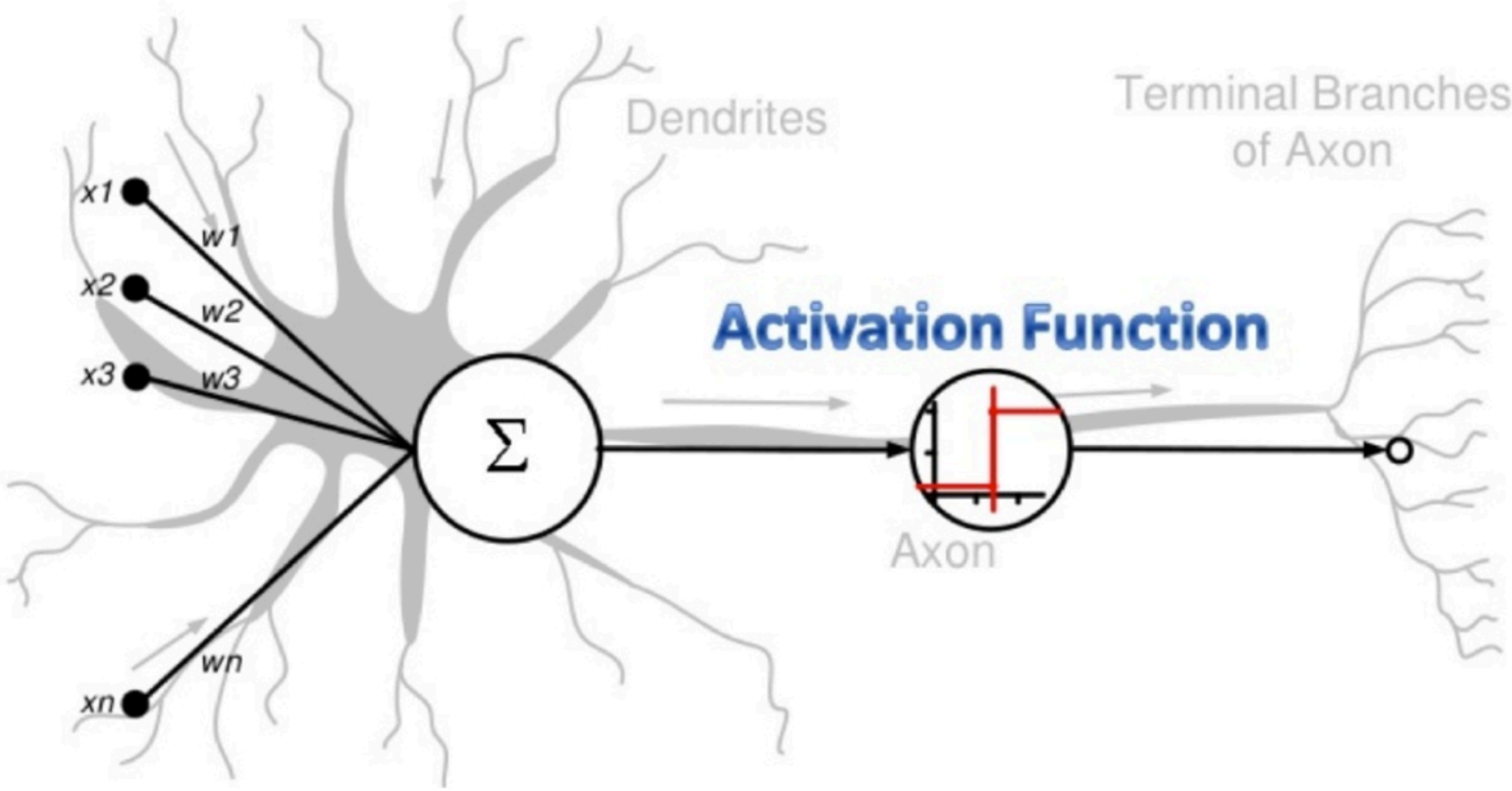


Bases Biológicas: Neurona

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

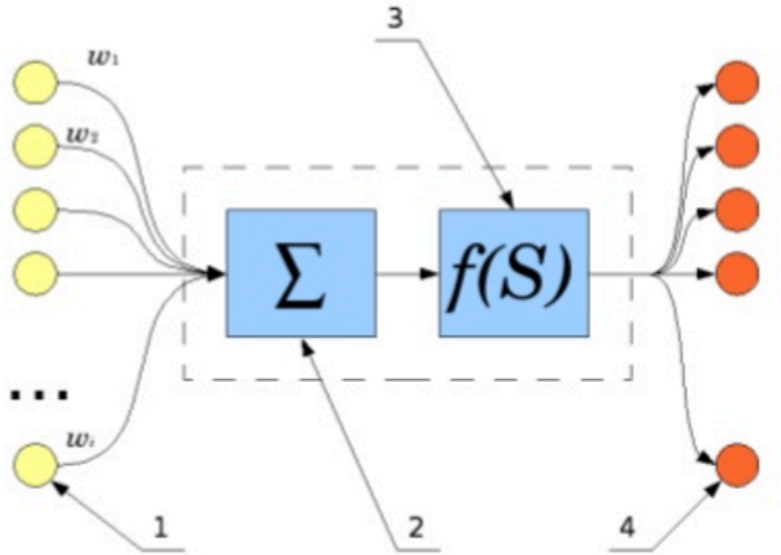


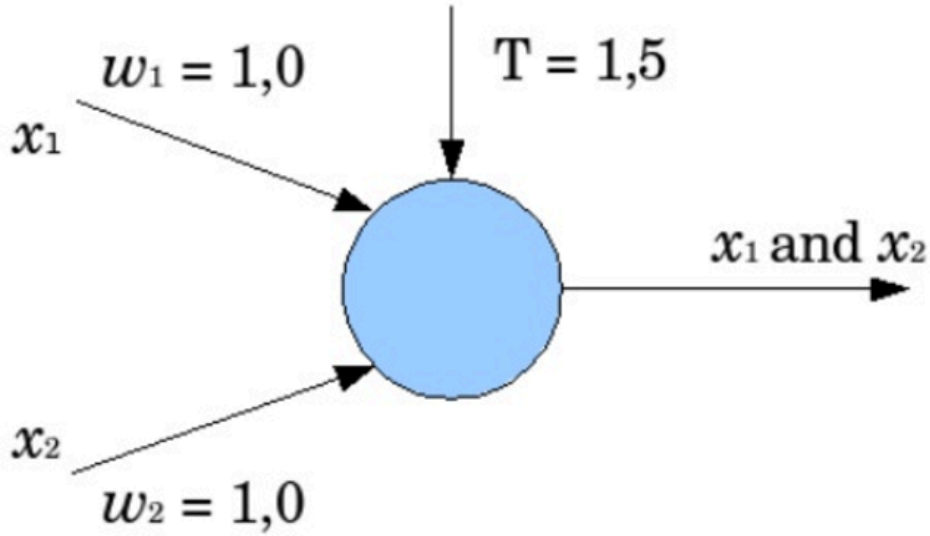
Redes Neuronales Artificiales



Redes Neuronales Artificiales

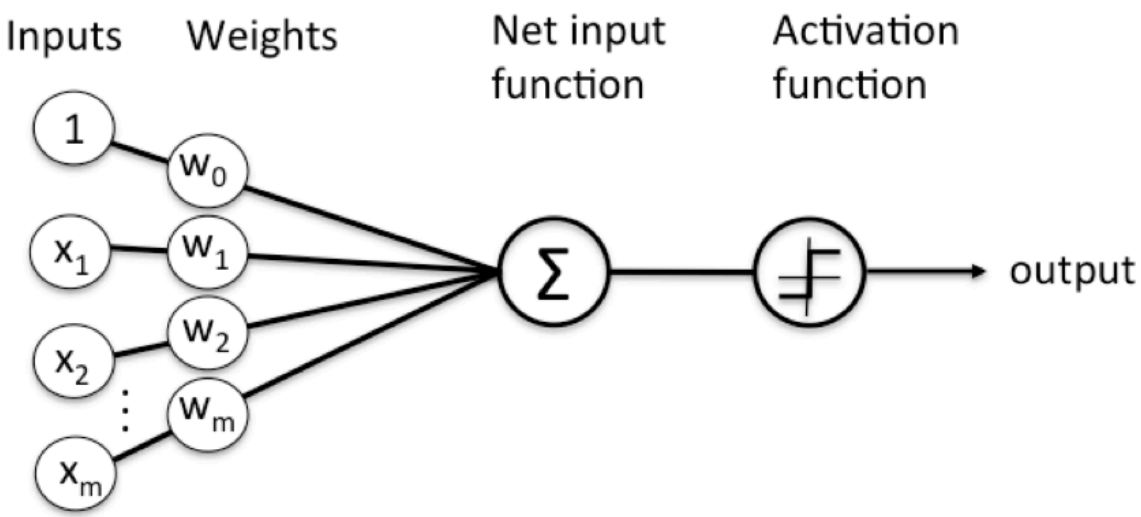
- 1943: McCulloh y Pitts “A Logical Calculus of the Ideas Immanent in Nervous Activity”





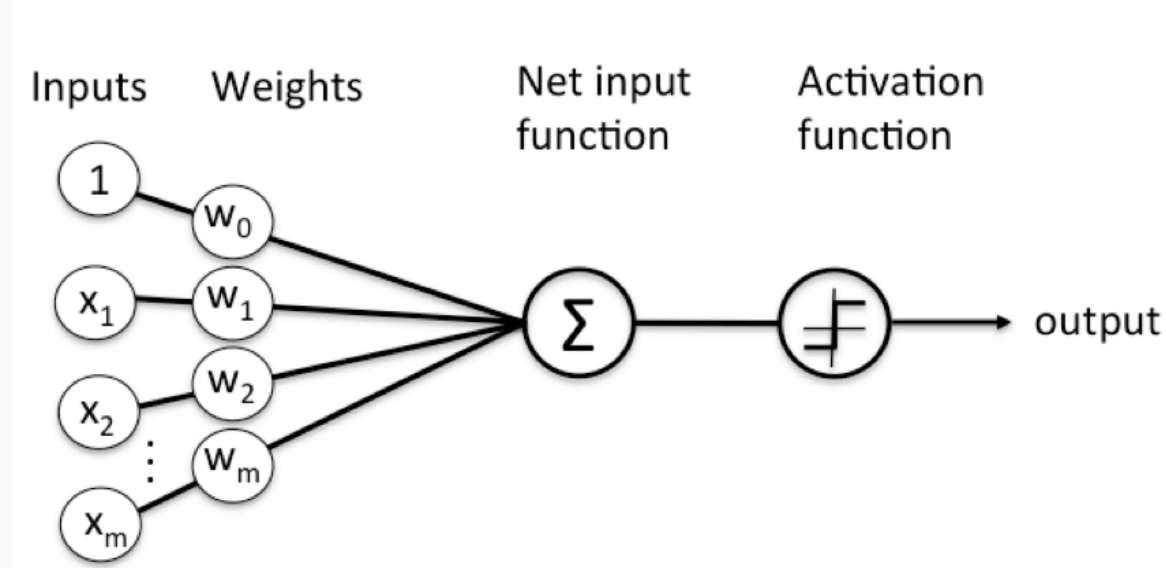
Perceptron

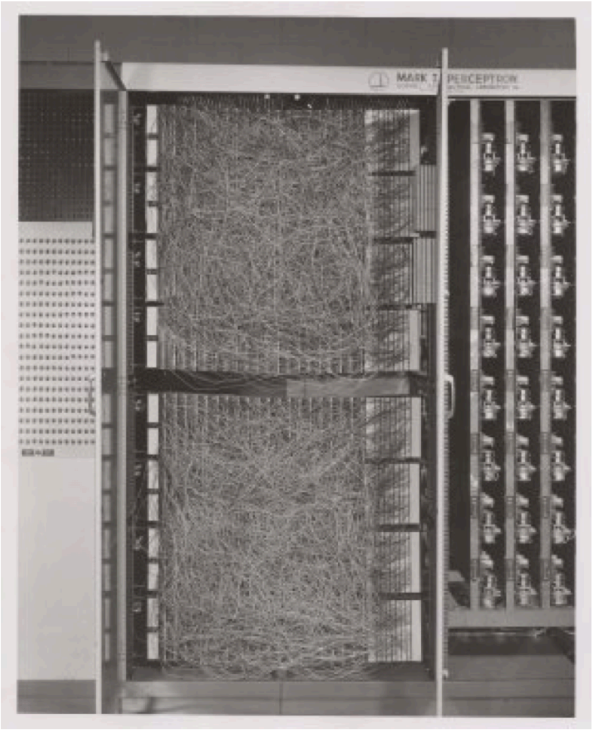
- 1957: Frank Rosenblatt



Perceptron

- 1957: Frank Rosenblatt



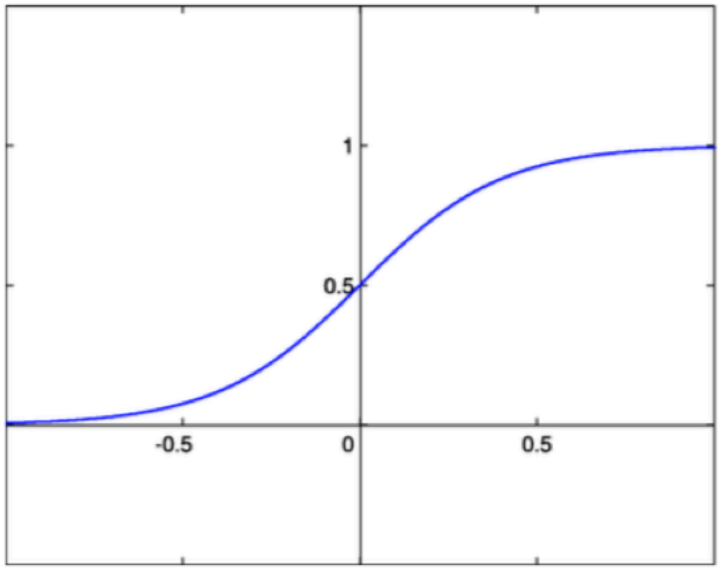


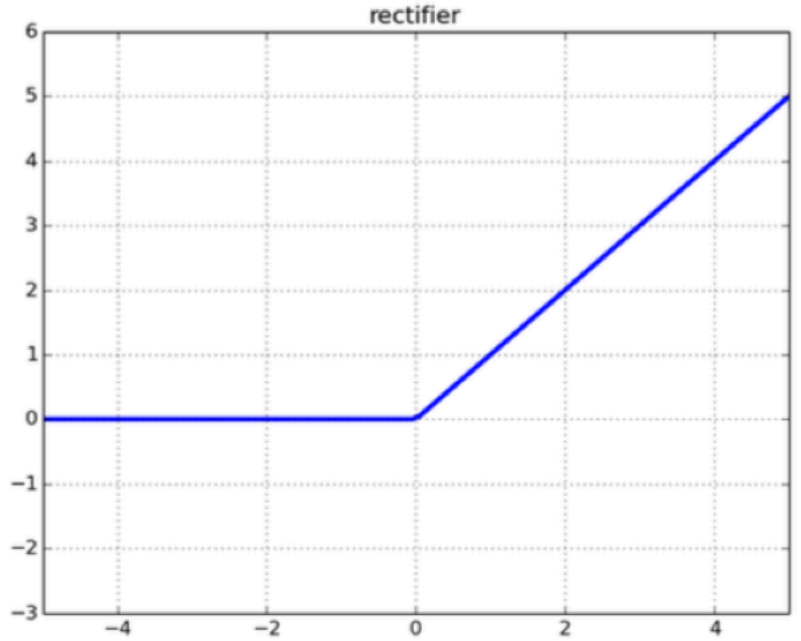
"[The Perceptron is] the embryo of an electronic computer that [the Navy] expects will be able to walk, talk, see, write, reproduce itself and be conscious of its existence."

THE NEW YORK TIMES

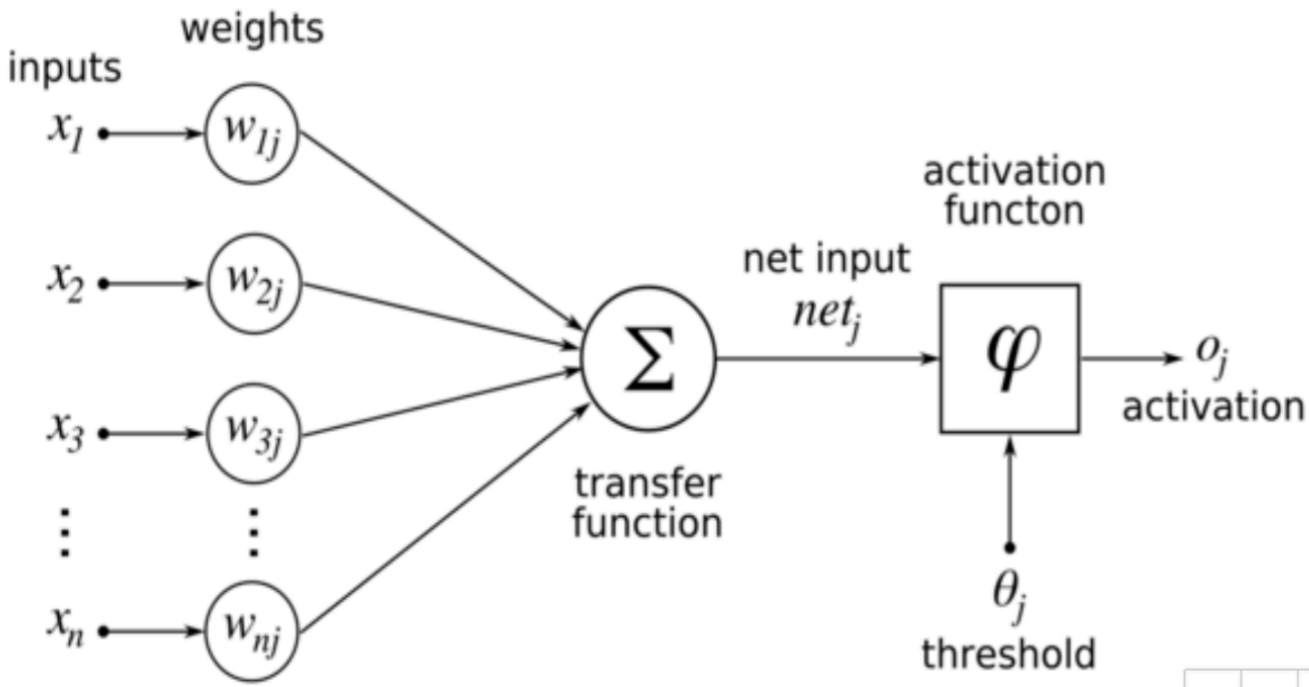
Funciones de Activación

- Step, tanh, sigmoid, ReLU



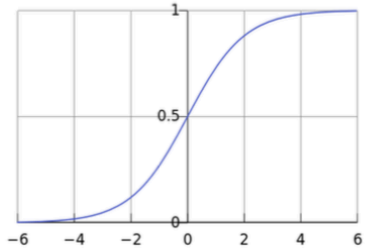


Perceptron



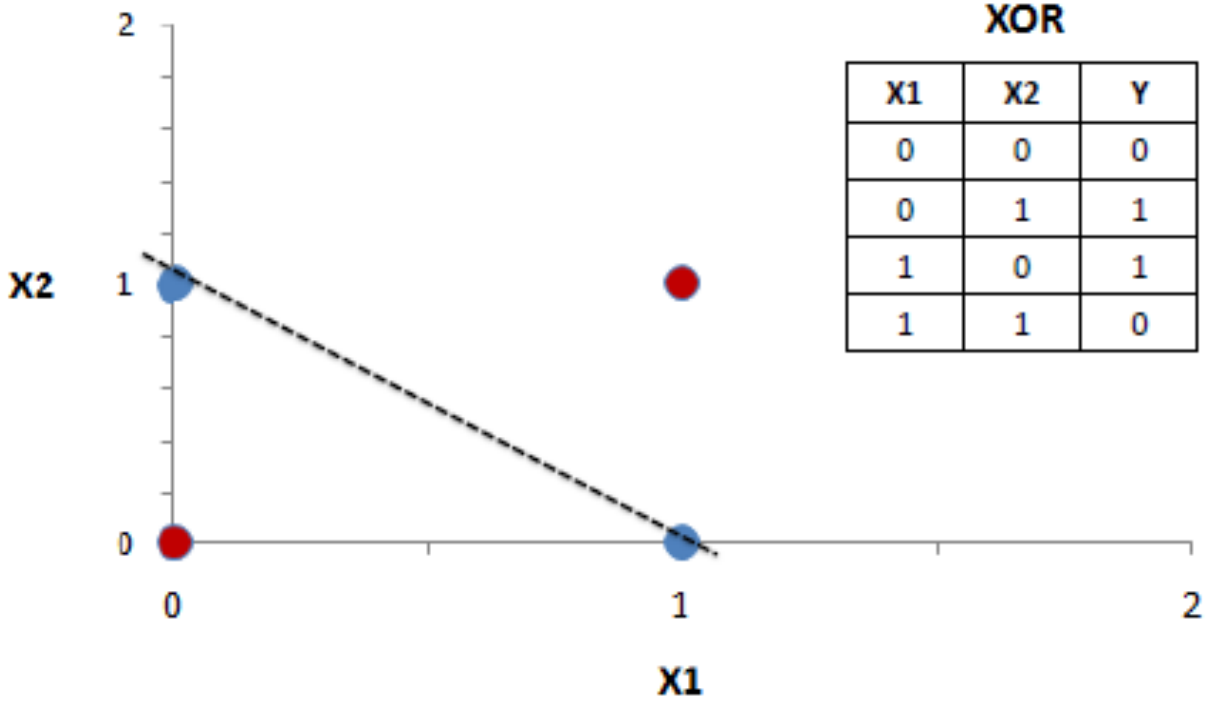
$$P(o_j = 1|x) = \phi\left(\sum_{i=1}^{n} w_{ij}x_i + \theta_j\right)$$

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



Limitaciones del Perceptron

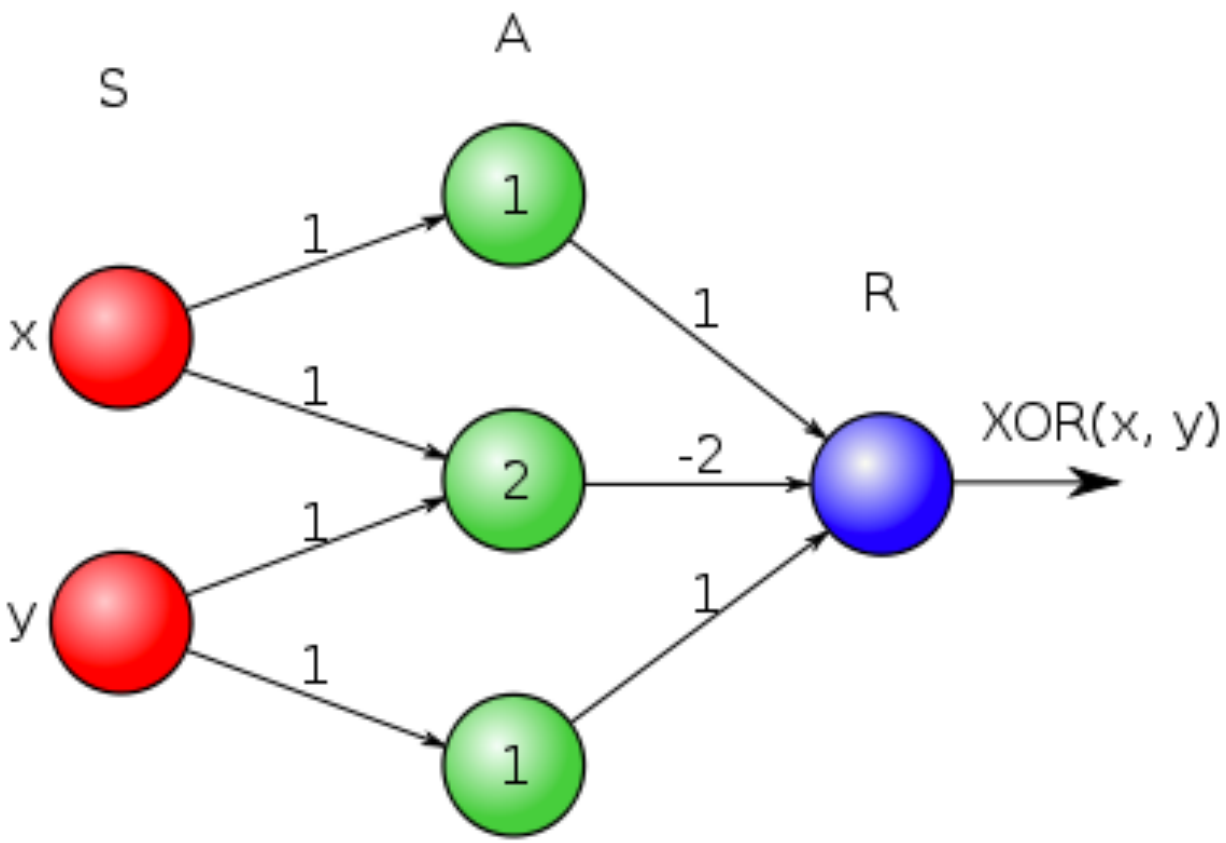
- El perceptron de una capa es un clasificador lineal
- No puede separar alguna funciones como el XOR



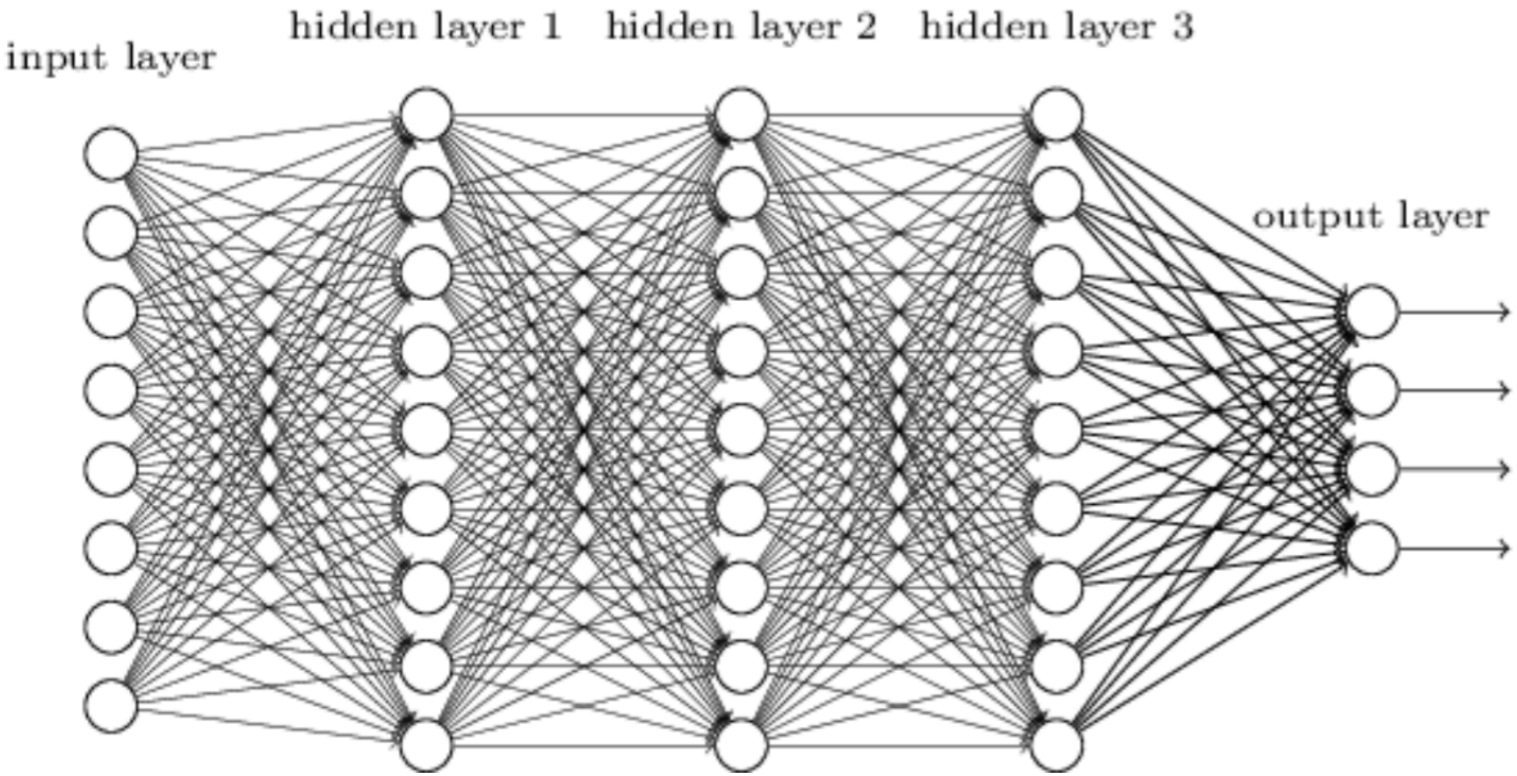
XOR

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

Perceptron con Hidden layers



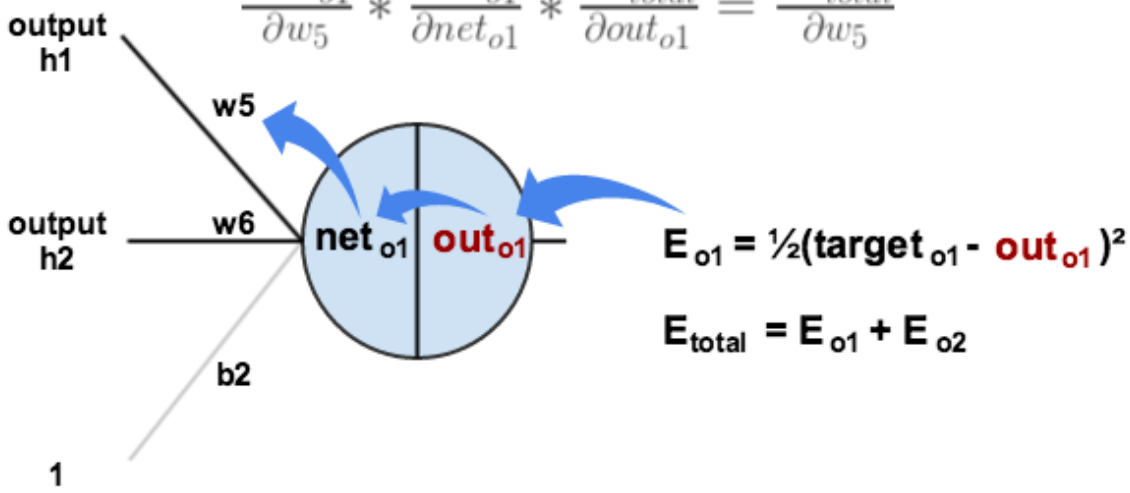
Redes Feedforward Multilayer

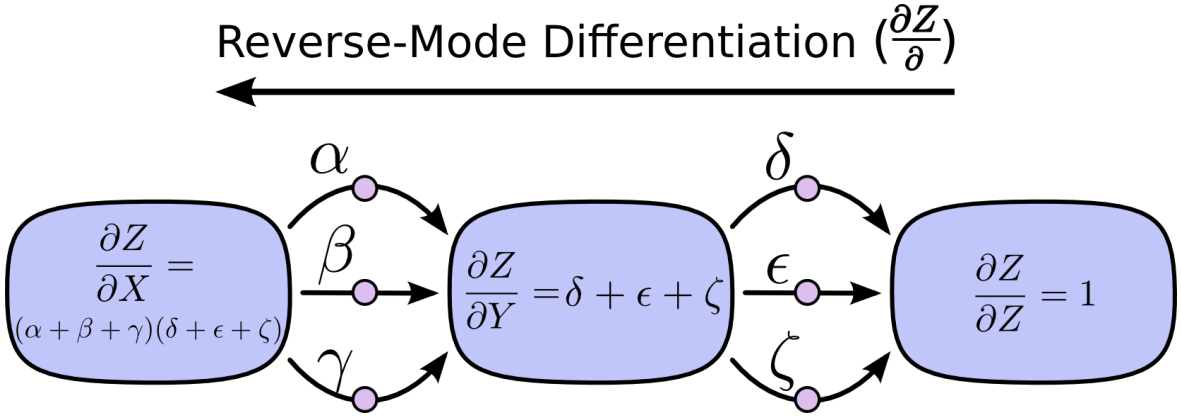


$$F(\mathbf{x}) := \sigma(\ldots \mathbf{W}^2 \sigma(\mathbf{W}^1 \mathbf{x}))$$

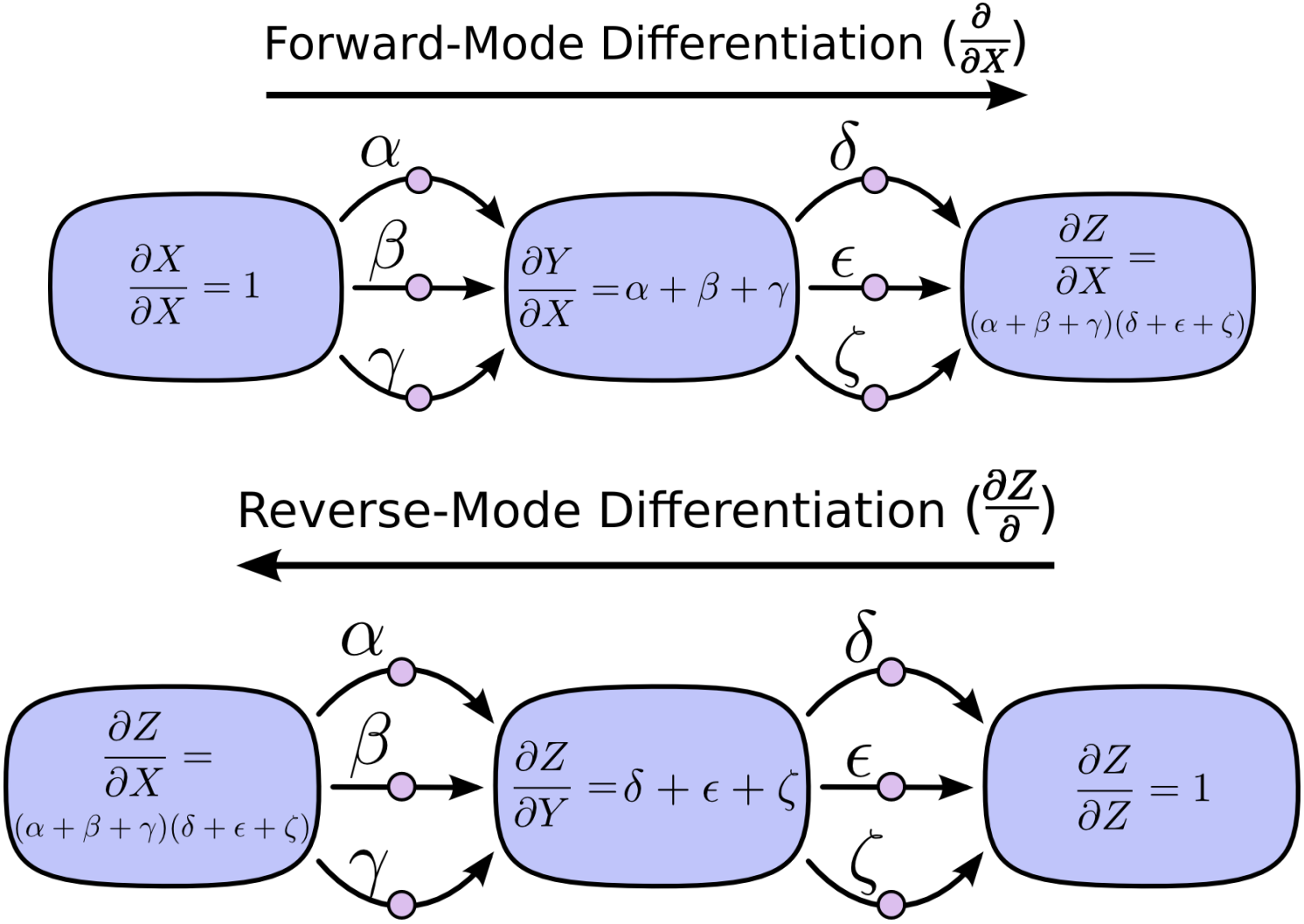
Backpropagation

$$\frac{\partial net_{o1}}{\partial w_5} * \frac{\partial out_{o1}}{\partial net_{o1}} * \frac{\partial E_{total}}{\partial out_{o1}} = \frac{\partial E_{total}}{\partial w_5}$$





Forward vs. Back Propagation



¿Cómo actualizar los pesos W?

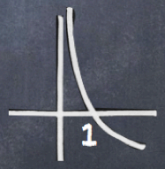
For each training row, we make a prediction and compare with the actual label (supervised learning):

predicted	actual	
0.8	1	married
0.2	0	single

Objective: minimize prediction error (MSE or cross-entropy)

Mean Square Error = $(0.2^2 + 0.2^2)/2$ "penalize differences per-class"

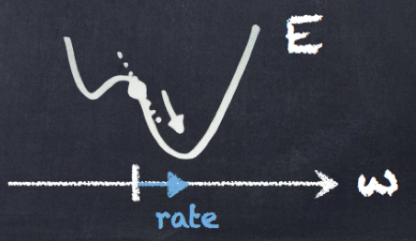
Cross-entropy = -log(0.8)



"strongly penalize non-1-ness"

Stochastic Gradient Descent: Update weights and biases via gradient of the error (via back-propagation):

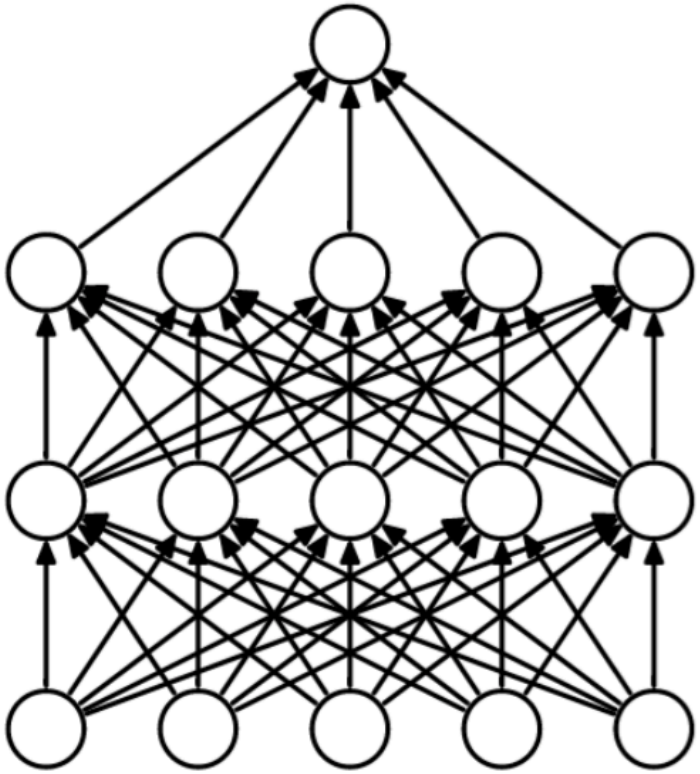
$$w \leftarrow w - rate * \partial E/\partial w$$



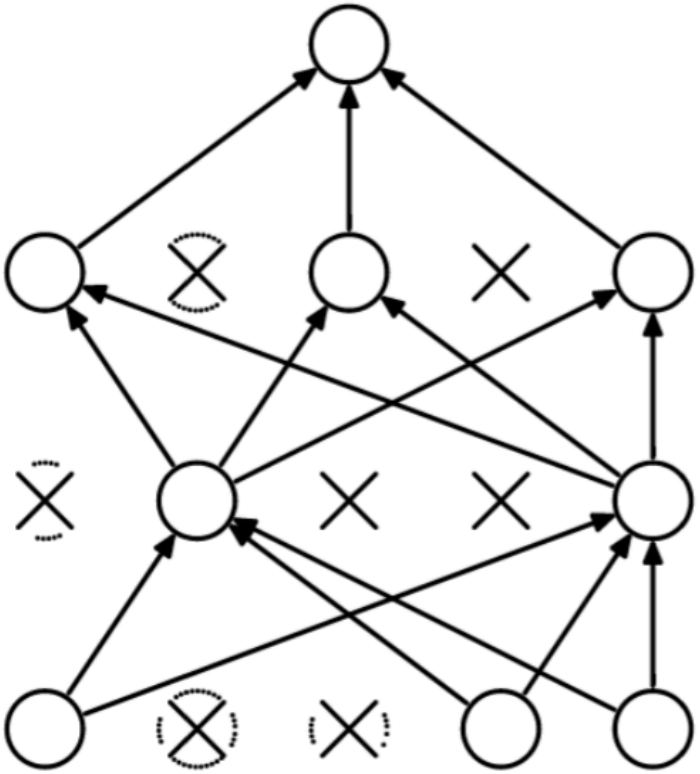
Otras Arquitecturas y F(x)s

- Dropout
- RLUs
- Autoencoders
- CNN
- RNN

Dropout



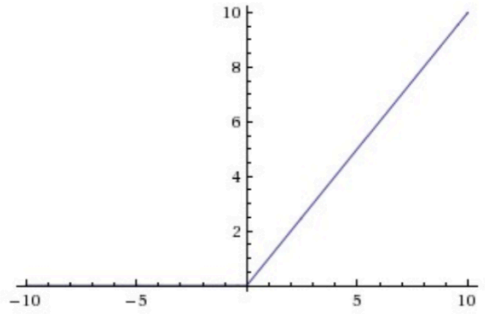
(a) Standard Neural Net



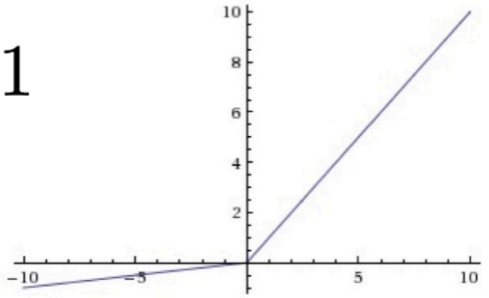
(b) After applying dropout.

Nuevas Rectified Linear Units

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

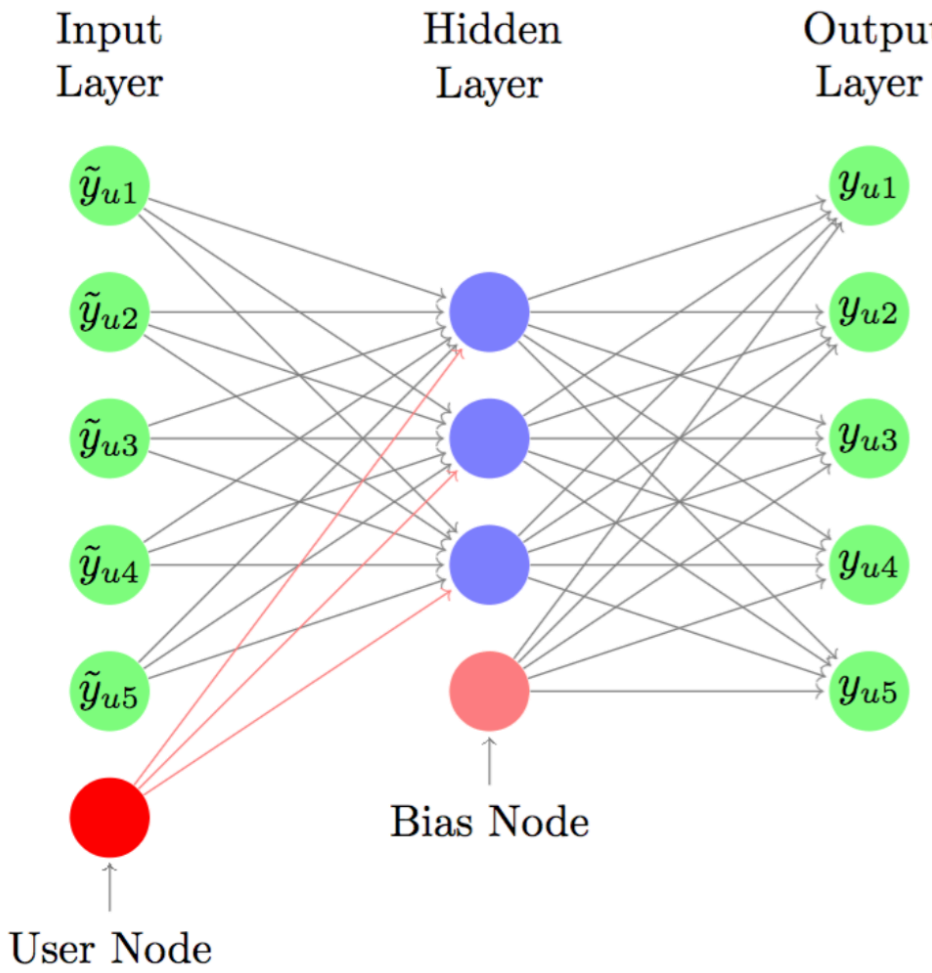


$$\sigma(x) = max(\alpha x, x) \quad \alpha < 1$$



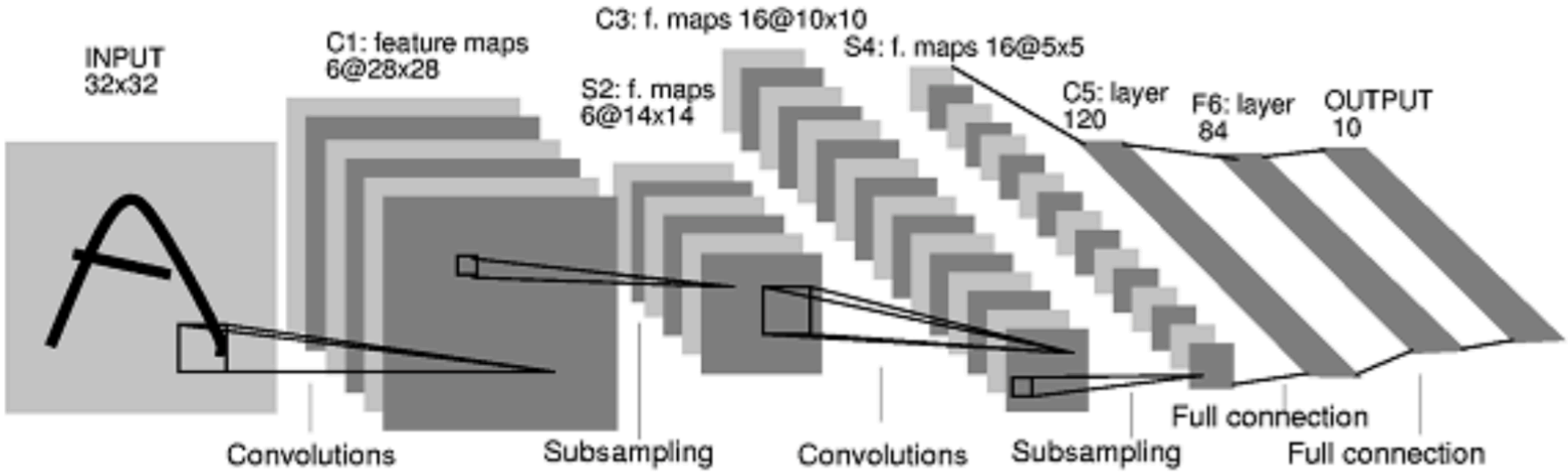
Neuron	MNIST	CIFAR10	NISTP	NORB
Without **unsupervised pre-training**				
Rectifier	**1.43%**	**50.86%**	**32.64%**	**16.40%**
Tanh	1.57%	52.62%	36.46%	19.29%
Softplus	1.77%	53.20%	35.48%	17.68%

Autoencoders

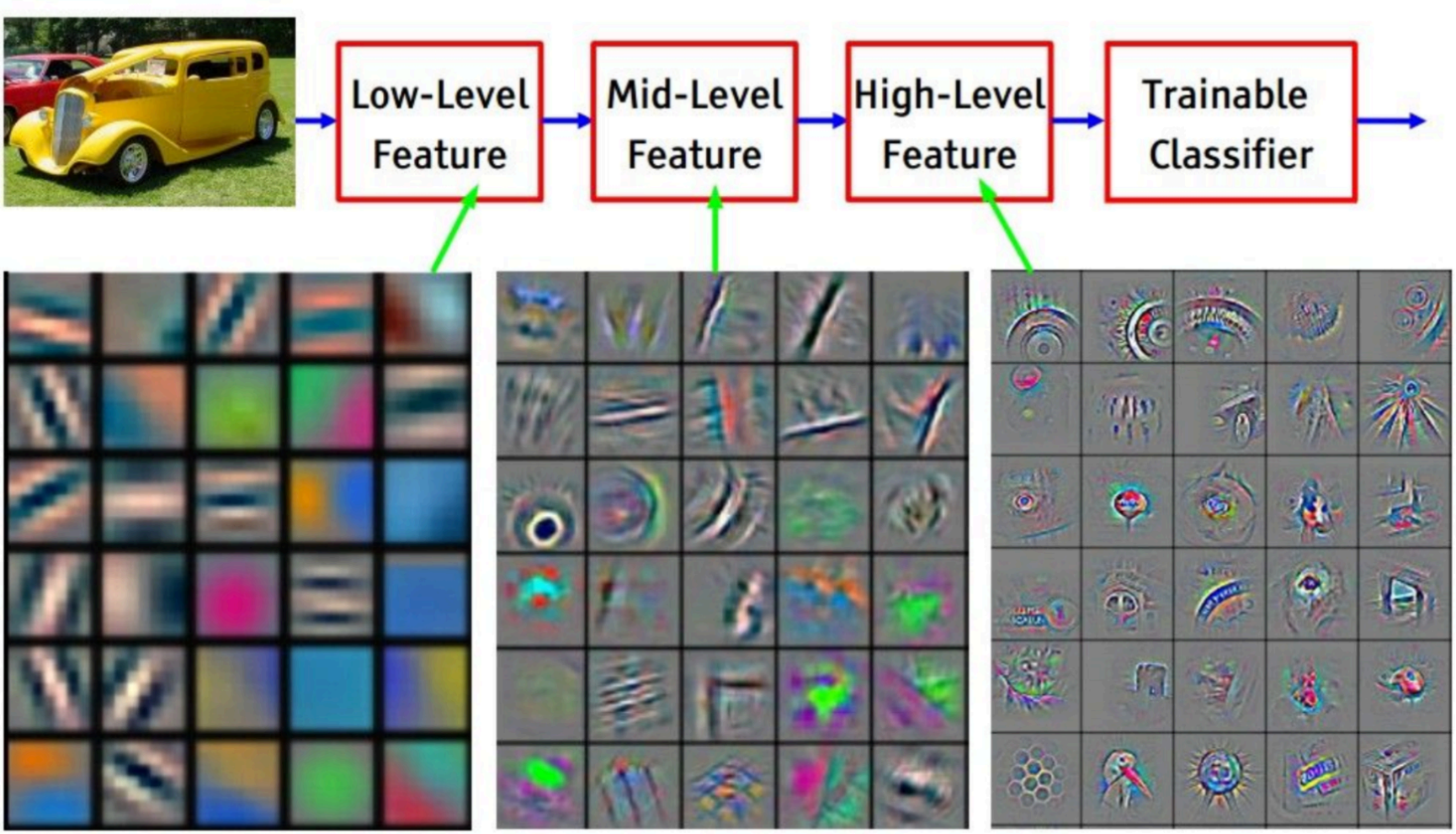


Collaborative Denoising Auto-Encoders for Top-N Recommender Systems Wu et.al. WSDM 2016

CNN

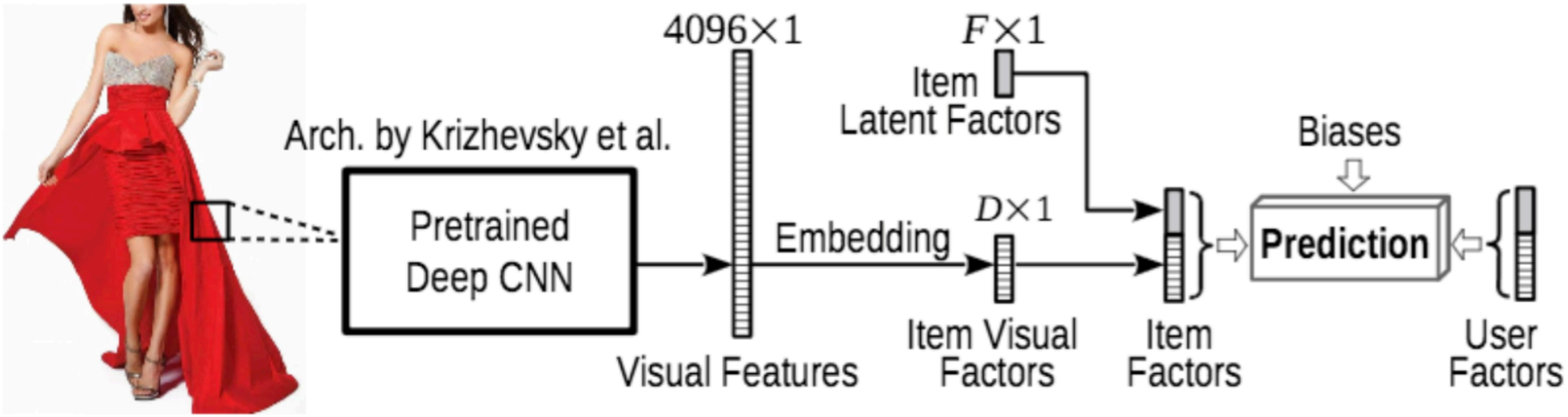

INPUT
32x32
C1: feature maps
6@28x28
S2: f. maps
6@14x14
C3: f. maps 16@10x10
S4: f. maps 16@5x5
C5: layer
120
F6: layer
84
OUTPUT
10
Convolutions
Subsampling
Convolutions
Subsampling
Full connection
Full connection

CNN



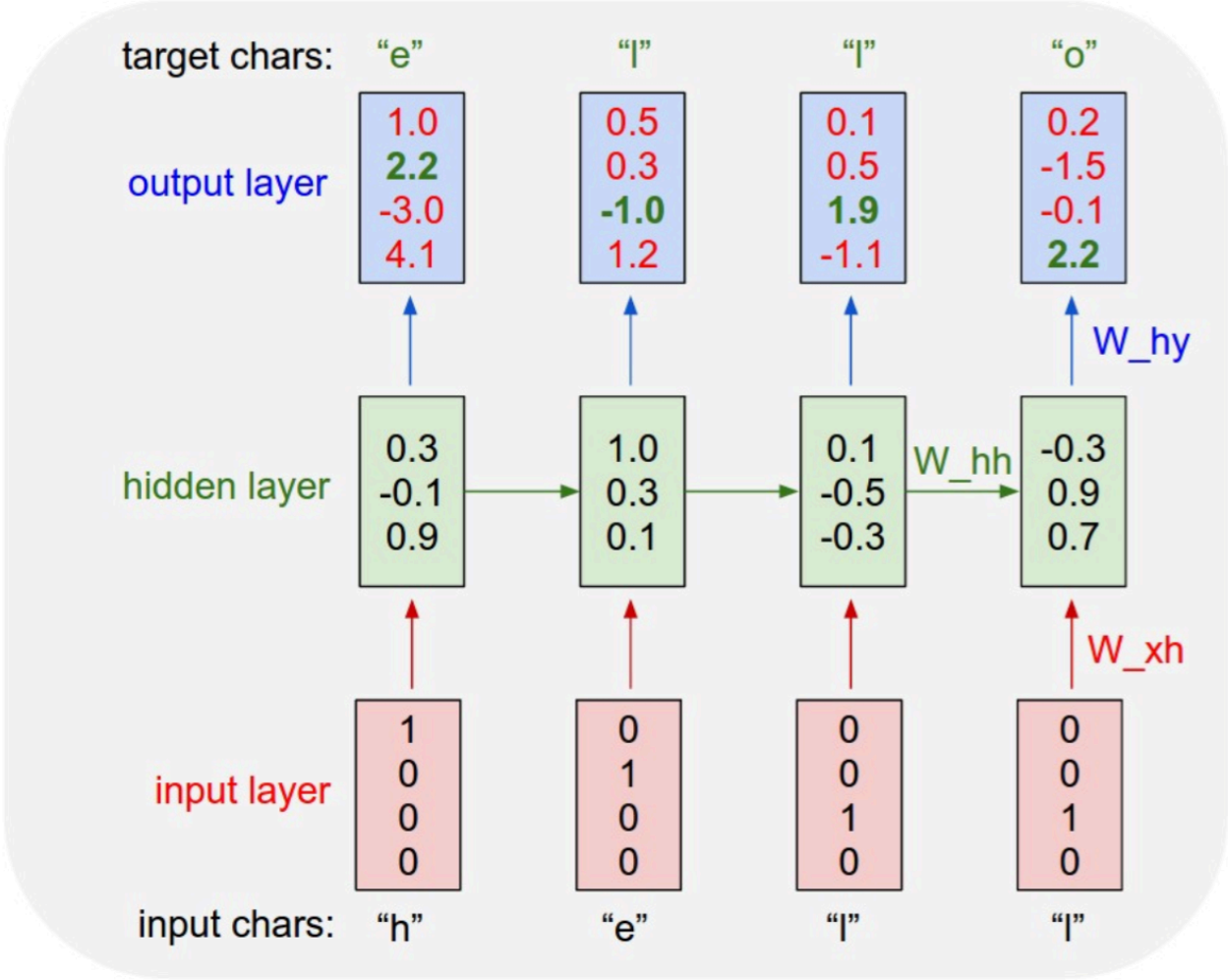
Feature visualization of convolutional net trained on ImageNet from [Zeiler & Fergus 2013]

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

- Theano: Python Library
- TensorFlow: Python Library
- Keras: High Level Python Library (Theano &TF)
- MXNET: R, Python, Julia

Papers en RecSys

- ***Ask the GRU*: Multi-task Learning for Deep Text Recommendations**. Trapit Bansal, David Belanger, and Andrew McCallum. 2016
- Modelos de factores latentes para recomendadores: trabajos previos han usado topic models o promedios de los word embeddings
- Este paper usa RNN (redes neuronales recurrentes) con GRUs (gated recurrent units)

Ask the GRU…

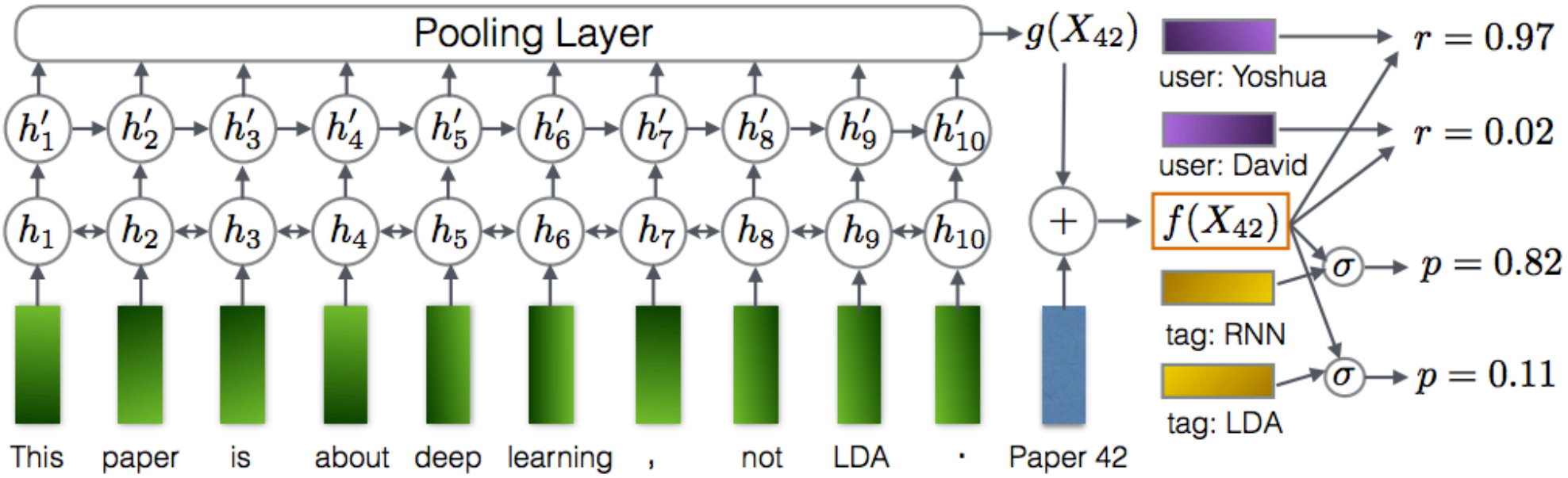


Figure 1: Proposed architecture for text item recommendation. Rectangular boxes represent embeddings. Two layers of RNN with GRU are used, where the first layer is a bi-directional RNN. The output of all the hidden units at the second layer is pooled to produce a text encoding which is combined with 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).

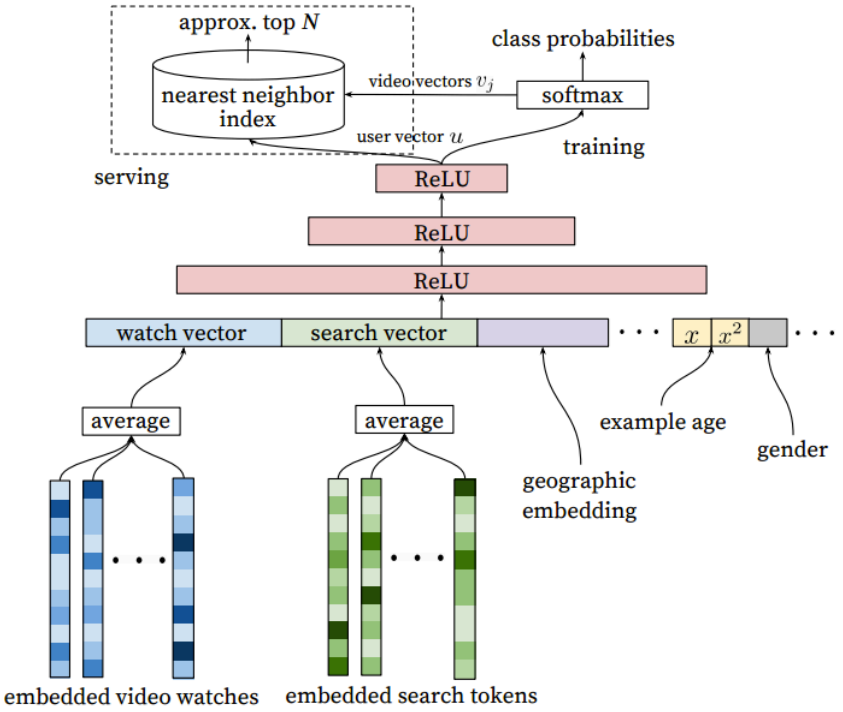
	Citeulike-a			Citeulike-t		
	Warm Start	Cold Start	Tag Prediction	Warm Start	Cold Start	Tag Prediction
GRU-MTL	**38.33**	**49.76**	**60.52**	45.60	**51.22**	**62.32**
GRU	36.87	46.16	—	42.59	47.59	—
CTR-MTL	35.51	39.87	48.95	**46.82**	34.98	46.66
CTR	31.10	39.00	—	40.44	33.74	—
Embed-MTL	36.64	41.71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

Paper 2

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

- Presenta Felipe del Río

este jueves



Paper 3

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

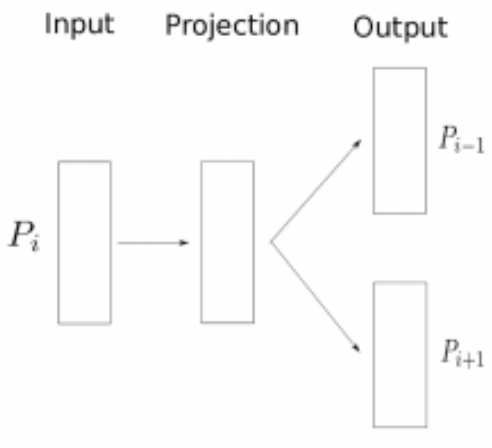


Figure 1: Prod2Vec Neural Net Architecture.

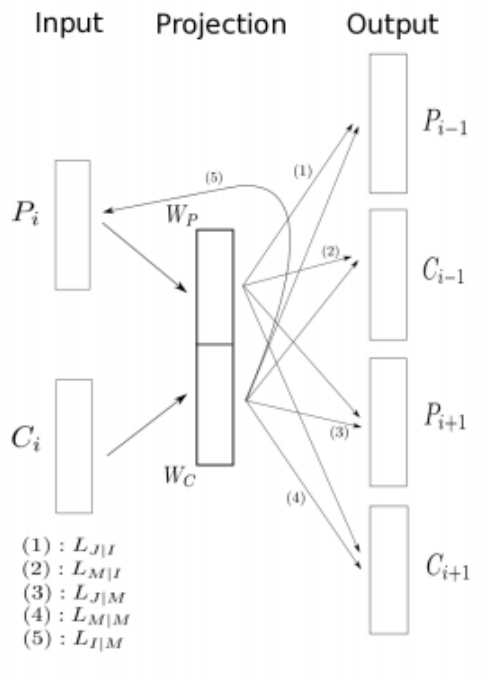


Figure 2: Meta-Prod2Vec Neural Net Architecture.

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*).

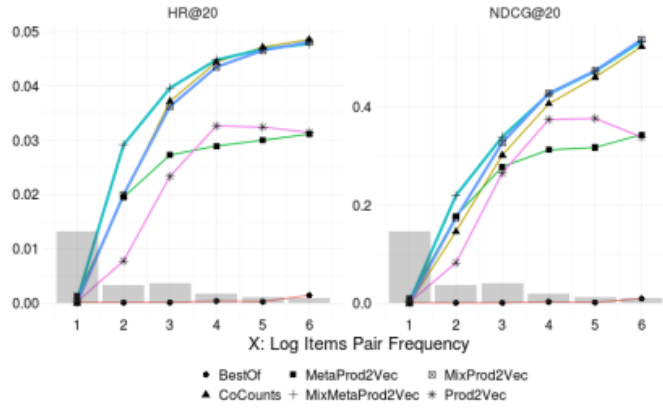


Figure 4: Cold-start improvements on the query and next item pairs.

Gracias!