Procesamiento del Lenguaje Natural mediante Redes Neuronales

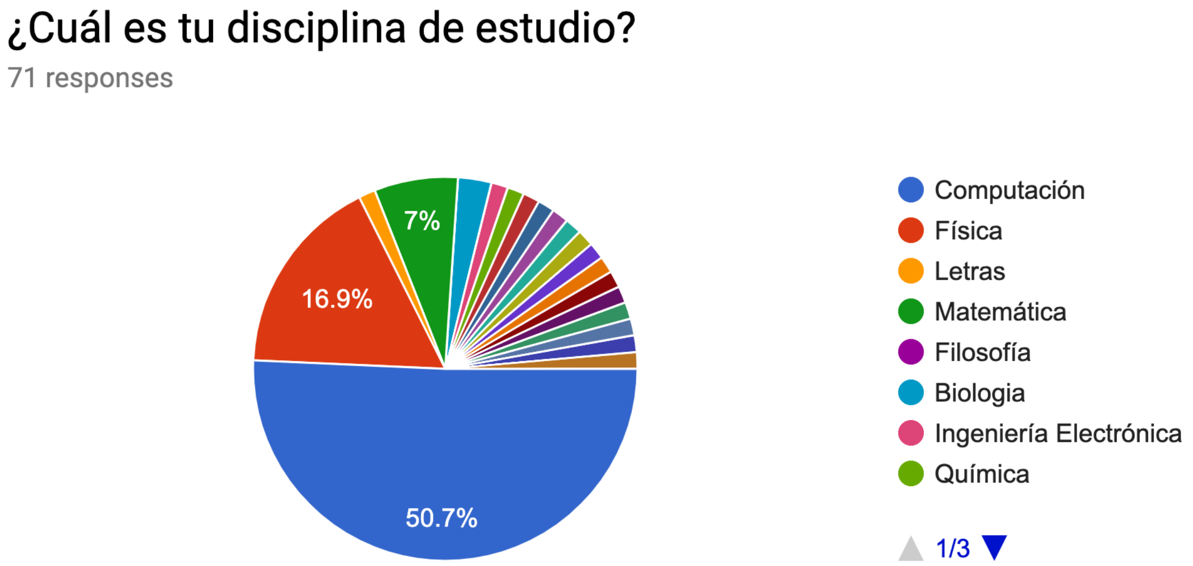
ECI 2019

Día 2: Word embeddings y redes neuronales multi-capa

Germán Kruszewski

Facebook AI Research

Qué somos? (Tiburones!)



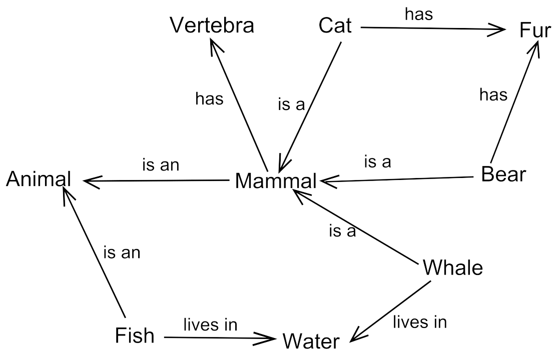
Día 2: Word embeddings y redes neuronales multi-capa

1. El significado de las palabras

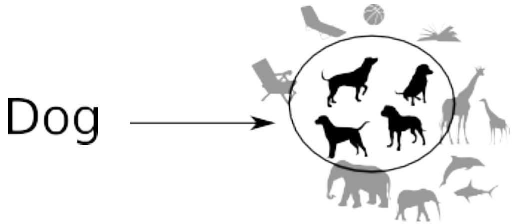
2. Redes neuronales multi-capa y backpropagation

3. Elementos prácticos

El significado de una palabra



Redes semánticas
(Wordnet; Miller, 1995)



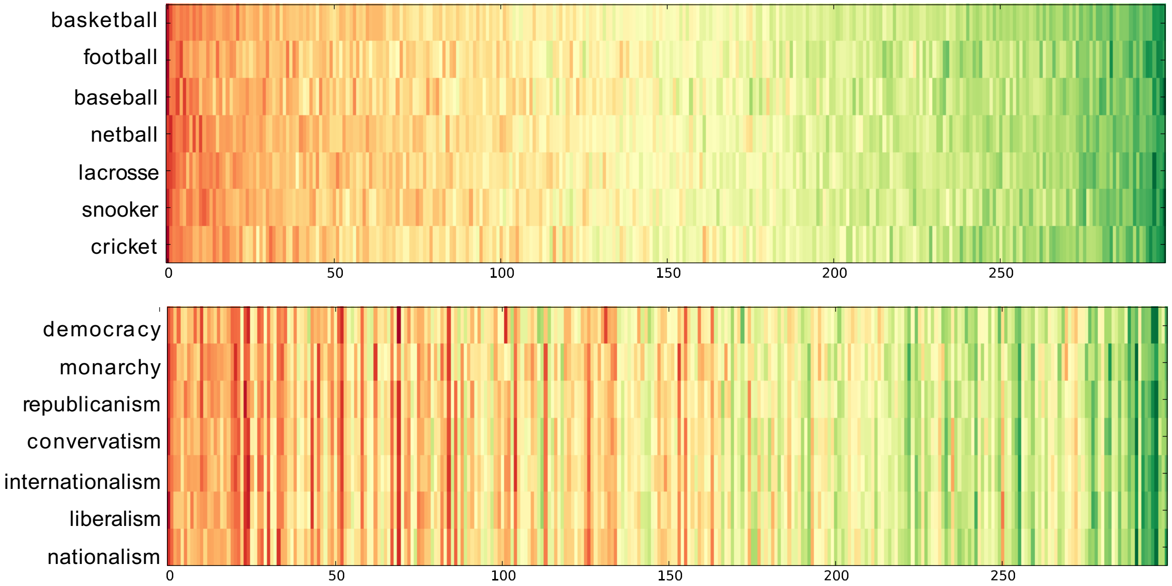
Semántica denotacional

La hipótesis distribucional
(Harris, 1954)

"*You shall know a word by the company it keeps*"

– Firth (1957)

Modelos distribucionales del significado



(dimensiones ordenadas en un orden arbitrario)

El contexto es el significado

st one-armed **professional** <**baseball**> **player** . Hector Castro (

is is no one way to run a <**baseball**> **team** or a ballet company

" invariably refers to a <**baseball**> **field** . Baseball has oft

ad out of the park with a <**baseball**> **bat** . Itchy and Scratchy

all " , to the sayings of <**baseball**> **star** Yogi Berra : " You

fortune in the 1980s as a <**basketball**> **player** , but it is his s

our favorite **NBA** and **NCAA** <**basketball**> **team** here . .. . Betting

e politicians is a former <**basketball**> **star** and another a forme

up was shown a video of a <**basketball**> **game** they were asked to

with livestock or even a <**basketball**> **court** in your garage . A

tatorship to the **Athenian** <**democracy**> of Summerhill . Institut

ace in royal mail . It is <**democracy**> versus **authoritarianism**

who murdered hundreds of <**democracy**> **activists** when they pour

fully-fledged **capitalist** <**democracy**> with its own role to pla

tted to the rule of **law** , <**democracy**> and human **rights** that it

El contexto es el significado

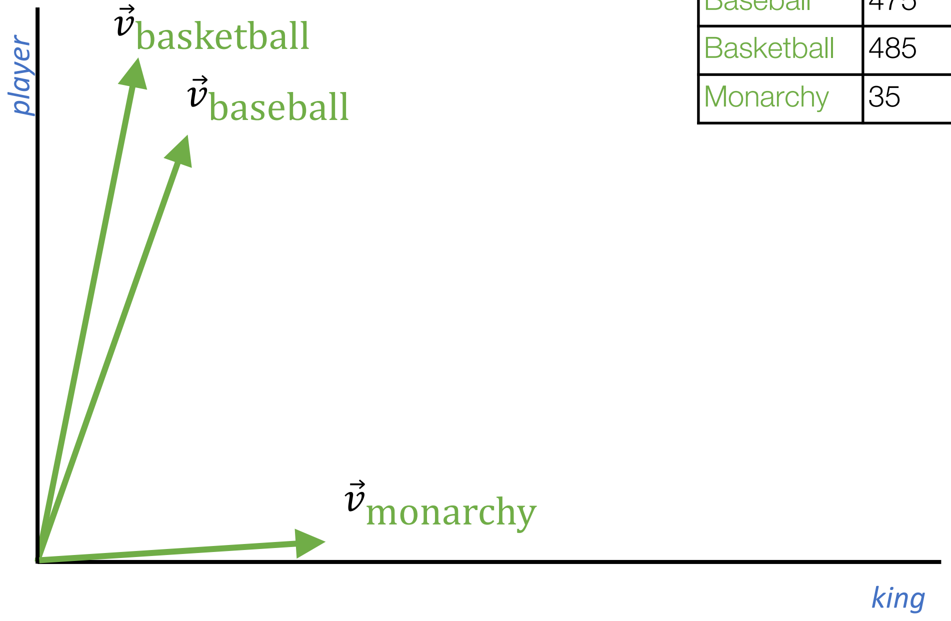
Matriz de co-ocurrencia

	player	*field*	*court*	*Athenian*	*king*	*the*
Baseball	475	350	5	1	115	975
Basketball	485	10	410	1	45	330
Democracy	1	5	2	350	10	375
Monarchy	35	1	4	7	276	1053

Matriz de co-ocurrencia

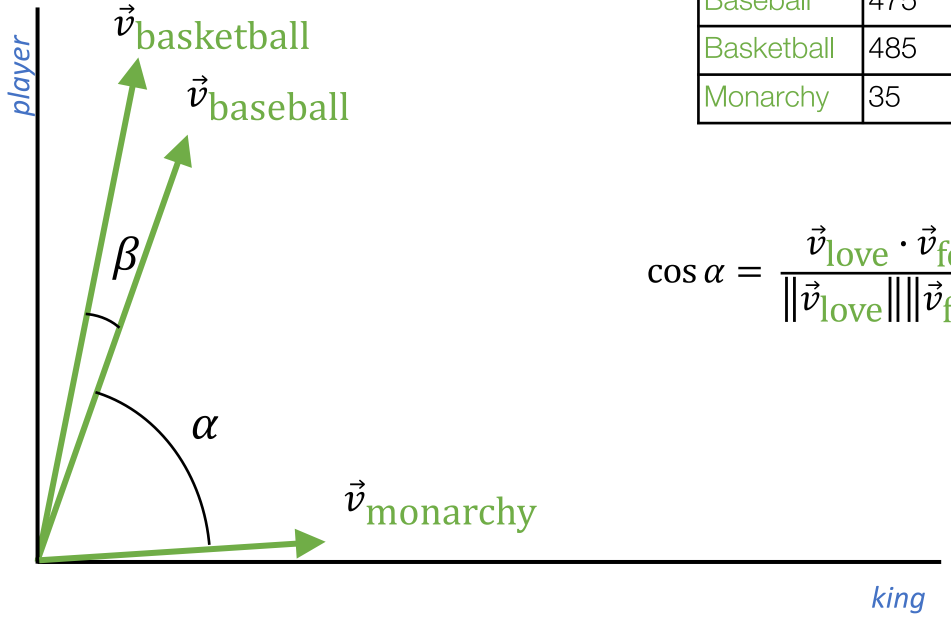
	player	*field*	*court*	*Athenian*	*king*	*the*
Baseball	475	350	5	1	115	975
Basketball	485	10	410	1	45	330
Democracy	1	5	2	350	10	375
Monarchy	35	1	4	7	276	1053

Interpretación geométrica



	player	*king*
Baseball	475	115
Basketball	485	45
Monarchy	35	276

Interpretación geométrica



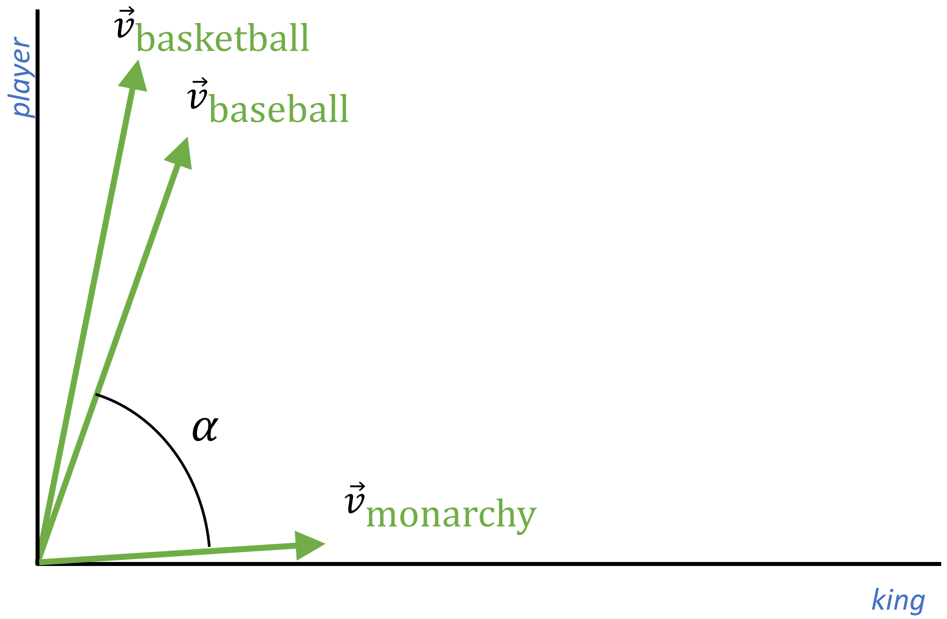
	player	*king*
Baseball	475	115
Basketball	485	45
Monarchy	35	276

$$\cos \alpha = \frac{\vec{v}_{\text{love}} \cdot \vec{v}_{\text{football}}}{\left\|\vec{v}_{\text{love}}\right\| \left\|\vec{v}_{\text{football}}\right\|}$$

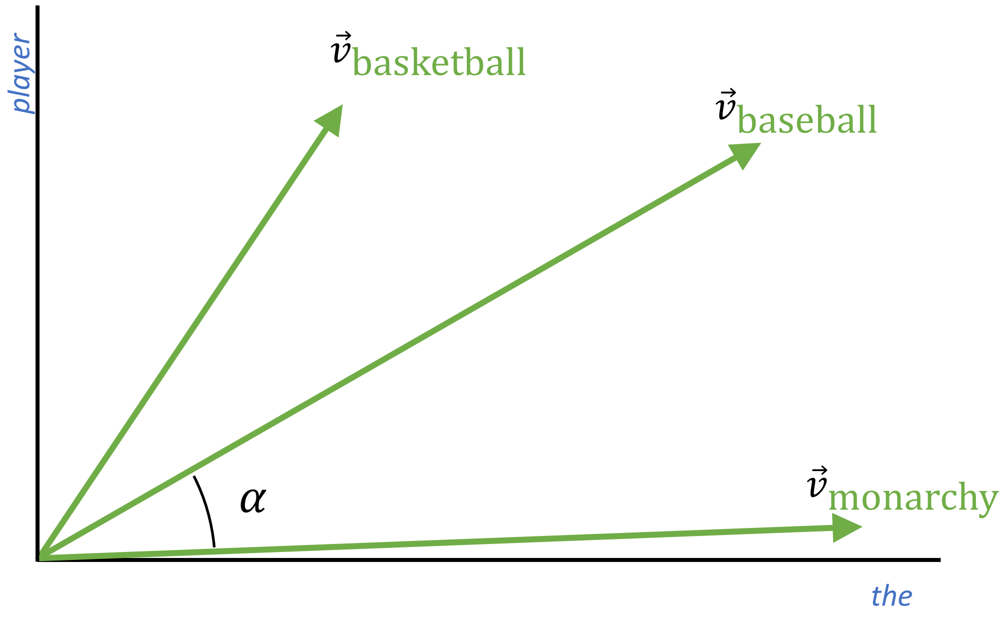
Efecto de las palabras muy frecuentes

	player	*field*	*court*	*Athenian*	*king*	*the*
Baseball	475	350	5	1	115	975
Basketball	485	10	410	1	45	330
Democracy	1	5	2	350	10	375
Monarchy	35	1	4	7	276	1053

Efecto de las palabras muy frecuentes



Efecto de las palabras muy frecuentes



Midiendo el nivel de asociación

	player	*field*	*court*	*Athenian*	*king*	*the*
Baseball	475	350	5	1	115	975
Basketball	485	10	410	1	45	330
Democracy	1	5	2	350	10	375
Monarchy	35	1	4	7	276	1053

$$PPMI(w,c) = \max\left(\log\frac{P(w,c)}{P(w,*)P(*,c)},0\right)$$

Positive Punctual Mutual Information

Midiendo el nivel de asociación

	player	*field*	*court*	*Athenian*	*king*	*the*
Baseball	475	350	5	1	115	975
Basketball	485	10	410	1	45	330
Democracy	1	5	2	350	10	375
Monarchy	35	1	4	7	276	1053

$$\frac{P(w, c)}{P(w,*)P(*, c)}$$

Midiendo el nivel de asociación

	player	*field*	*court*	*Athenian*	*king*	*the*
Baseball	475	350	5	1	115	975
Basketball	485	10	410	1	45	330
Democracy	1	5	2	350	10	375
Monarchy	35	1	4	7	276	1053

$$\frac{P(\text{baseball}, \text{player})}{P(\text{baseball}, *)P(*, \text{player})}$$

Midiendo el nivel de asociación

	player	*field*	*court*	*Athenian*	*king*	*the*
Baseball	475	350	5	1	115	975
Basketball	485	10	410	1	45	330
Democracy	1	5	2	350	10	375
Monarchy	35	1	4	7	276	1053

$$\log \frac{P(\text{baseball}, \text{player})}{P(\text{baseball}, *)P(*, \text{player})}$$

Midiendo el nivel de asociación

	player	*field*	*court*	*Athenian*	*king*	*the*
Baseball	475	350	5	1	115	975
Basketball	485	10	410	1	45	330
Democracy	1	5	2	350	10	375
Monarchy	35	1	4	7	276	1053

$$PPMI(w, c) = \max\left(\log \frac{P(\text{baseball}, \text{player})}{P(\text{baseball}, *)P(*, \text{player})}, 0\right)$$

Midiendo el nivel de asociación

	player	*field*	*court*	*Athenian*	*king*	*the*
Baseball	475	350	5	1	115	975
Basketball	485	10	410	1	45	330
Democracy	1	5	2	350	10	375
Monarchy	35	1	4	7	276	1053

$$\frac{P(\text{baseball}, \text{player})}{\blacksquare\blacksquare}$$

Midiendo el nivel de asociación

	player	*field*	*court*	*Athenian*	*king*	*the*
Baseball	475	350	5	1	115	975
Basketball	485	10	410	1	45	330
Democracy	1	5	2	350	10	375
Monarchy	35	1	4	7	276	1053

$$\frac{\blacksquare}{P(\text{baseball}, *)\blacksquare}$$

Midiendo el nivel de asociación

	player	*field*	*court*	*Athenian*	*king*	*the*
Baseball	475	350	5	1	115	975
Basketball	485	10	410	1	45	330
Democracy	1	5	2	350	10	375
Monarchy	35	1	4	7	276	1053

$$\frac{\blacksquare}{\blacksquare P(*, \text{player})}$$

Midiendo el nivel de asociación

	player	*field*	*court*	*Athenian*	*king*	*the*
Baseball	0.38	0.97	0	0	0	0
Basketball	0.21	0	0.94	0	0	0.03
Democracy	0	0	0	1.93	0	0
Monarchy	0	0	0	0	1.86	0.01

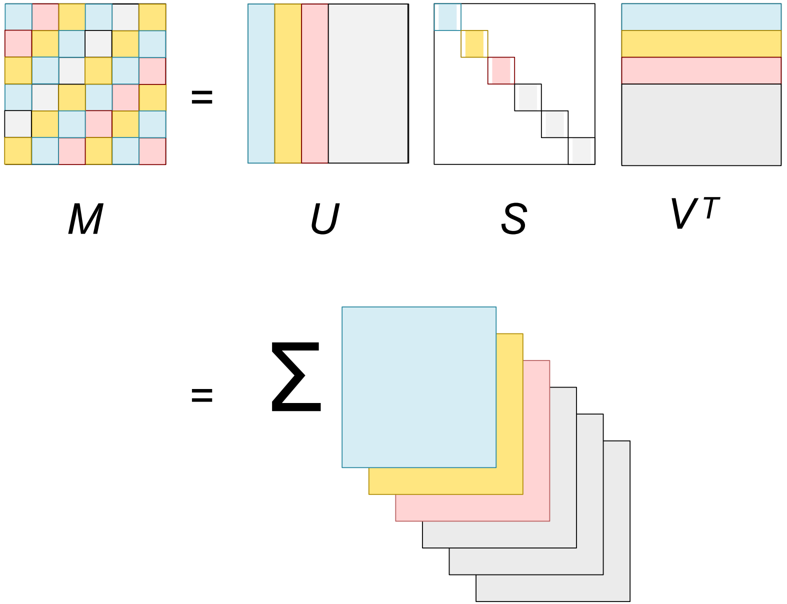
$$PPMI(w, c) = \max\left(\log \frac{P(\text{baseball}, \text{player})}{P(\text{baseball}, *)P(*, \text{player})}, 0\right)$$

Reducción de la dimensionalidad

	player	*field*	*court*	*Athenian*	*king*	*the*	…	…	…
Baseball	0.38	0.97	0	0	0	0			
Basketball	0.21	0	0.94	0	0	0.03			
Democracy	0	0	0	1.93	0	0			
Monarchy	0	0	0	0	1.86	0.01			

¡Un montón! (10k – 100k)

Descomposición en valores singulares

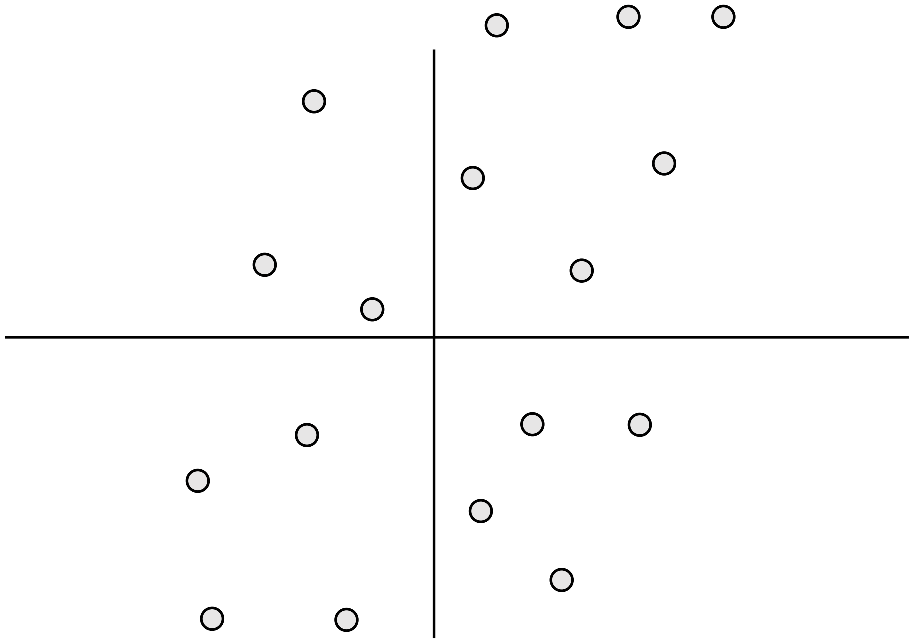


Descomposición en valores singulares (SVD).

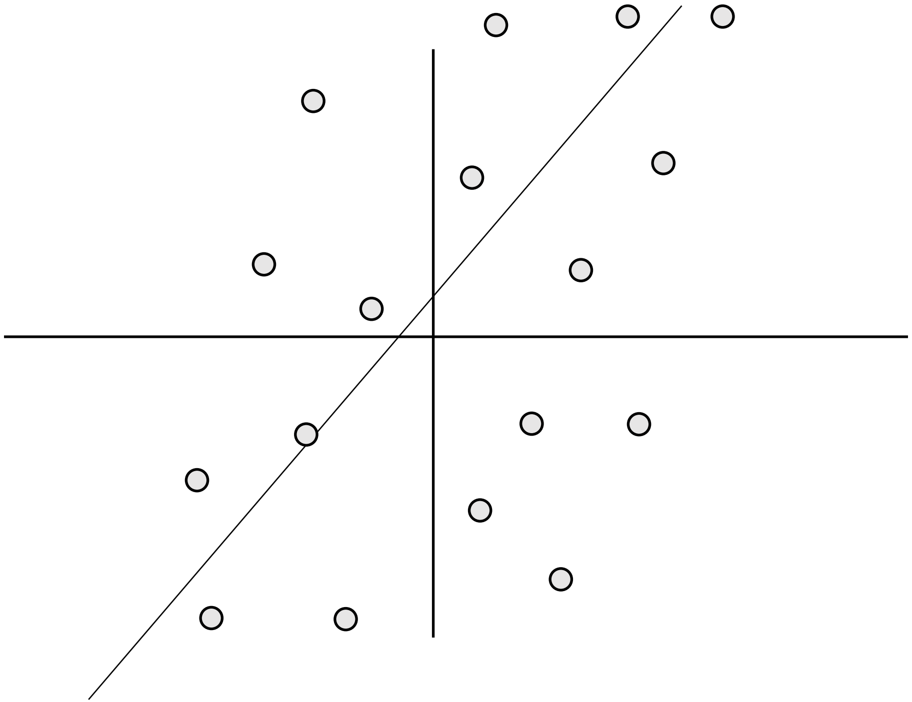
Vectores de palabras y vectores de contextos

$$\underset{V \times V}{M} \approx \underset{V \times d}{W} \times \underset{d \times V}{C}$$

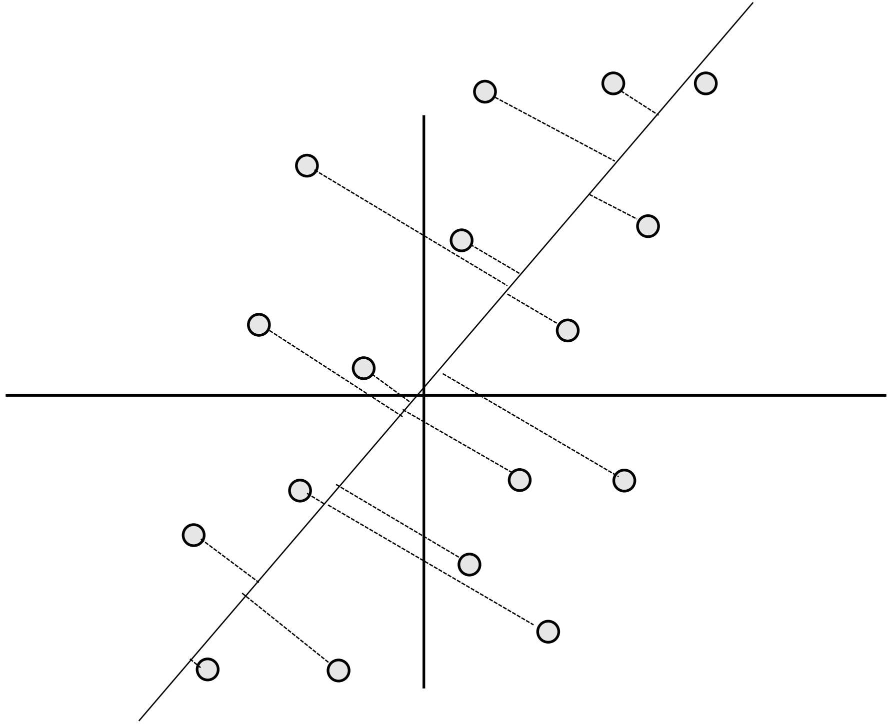
Reducción de la dimensionalidad



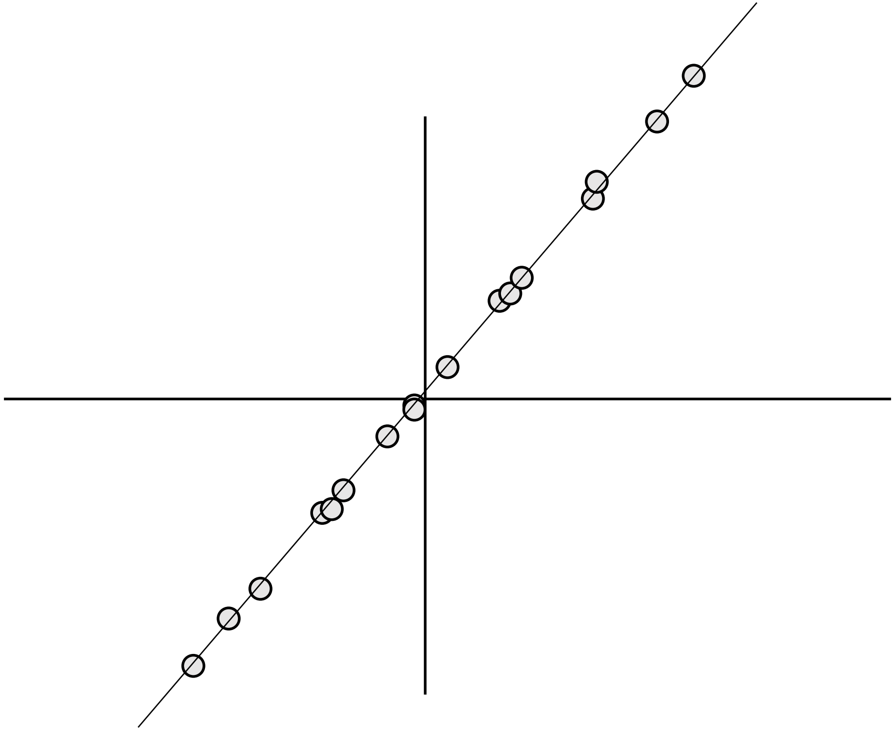
Reducción de la dimensionalidad



Reducción de la dimensionalidad



Reducción de la dimensionalidad



Representaciones semánticas distribuídas basadas en conteo

	0	*1*	*2*	*3*	*4*	*5*
Baseball	0.2	-1	-0.1	0	-0.4	0.2
Basketball	0.1	0.1	-0.5	0.2	-0.1	0.1
Democracy	-0.5	-0.4	0	0.2	0.1	-0.5
Monarchy	0.3	-0.1	0.2	-0.2	0	0.3

Dimensiones abstractas representando algún tipo de contexto (p. ej. “jugador”, “deportista”, “árbitro”, …)

Representaciones distribuídas basadas en conteo

- Ventajas:
  - ¡Rápido!
- Desventajas:
  - No está claro qué estamos optimizando.
  - Necesidad de mantener explícitamente en memoria la matriz de co-ocurrencia.

Representaciones distribuídas basadas en predicción

- Asumimos un corpus de texto $T = [w_1, w_2, \dots, w_{|T|}]$
- Observamos cada palabra w_i
- Por cada palabra w_i observamos las palabras que aparecen a no más de c (tamaño de ventana) posiciones de distancia.

… life, the **universe** and everything …

c=2

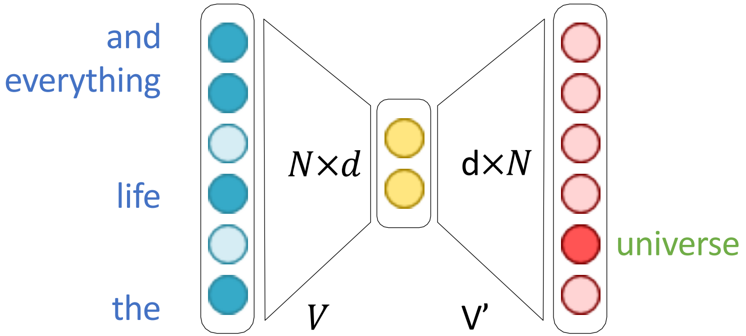
w_i (= **universe**)
$C_i = [w_{i-c}, \dots, w_{i-1}, w_{i+1}, \dots, w_{i+c}]$
(= [life, the, and, everything])

- Usamos la palabra central y el contexto en un problema de clasificación.

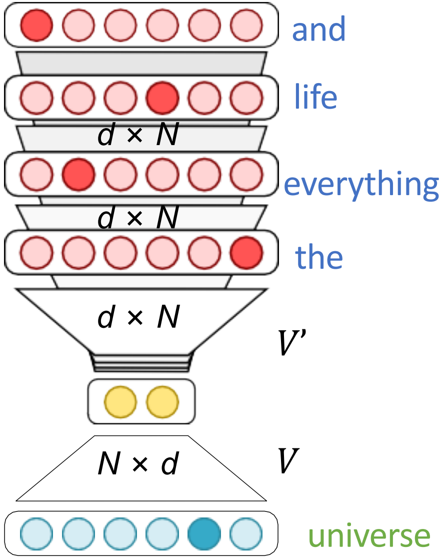
Representaciones distribuídas basadas en predicción

... life, the **universe** and everything

...

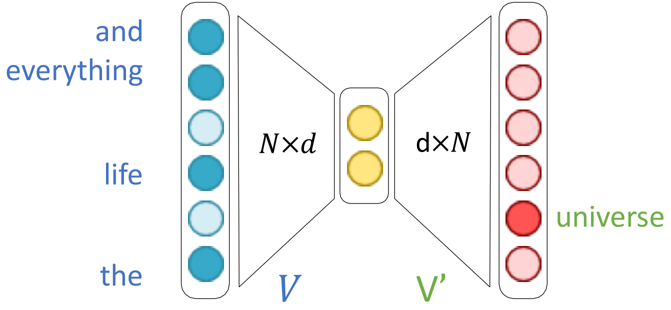


CBOW



Skip-Gram

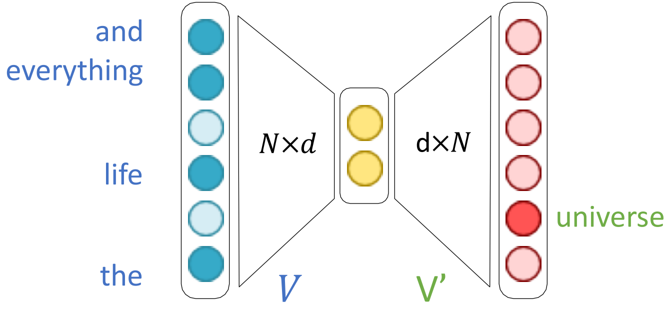
CBOW



- El objetivo de CBOW es predecir w_i a partir de C_i, o equivalentemente maximizar $P(w_i \,|C_i)$.

$$L = -\sum_{i=1}^{|T|} \log P(w_i|C_i)$$

CBOW



- Representamos el contexto como un bag of word embeddings:

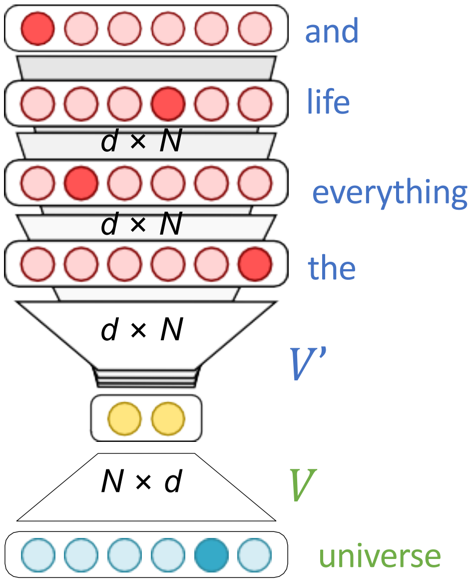
$$\boldsymbol{h} = \frac{1}{|\mathrm{C_i}|} \sum_{w \in C_i} \boldsymbol{V_w}$$

$$P(w_i|C_i) = \frac{\exp\left(\boldsymbol{V'_{w_i}} \cdot \boldsymbol{h}\right)}{\sum_{w=1}^{N} \exp(\boldsymbol{V'_w} \cdot \boldsymbol{h})}$$

Skip-Gram

- El objetivo de Skip-gram es predecir cada $w_j \in C_i$ a partir de w_i, o equivalentemente maximizar $P(w_j | w_i)$.

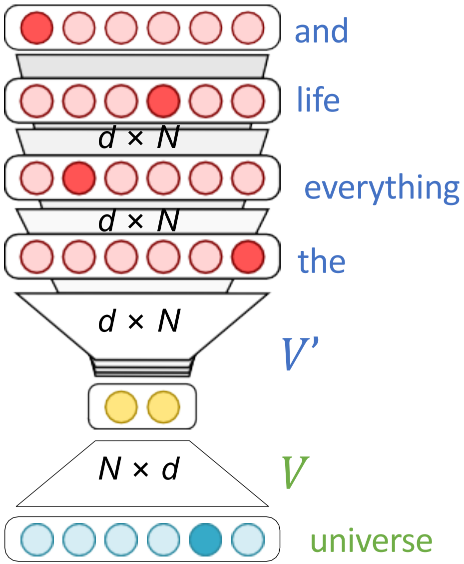
$$L = -\sum_{i=1}^{|T|} \sum_{\substack{j=i-c \\ j \neq i}}^{i+c} \log P(w_j | w_i)$$



Skip-Gram

$$P(w_j|w_i) = \frac{\exp\left(V_{w_i} \cdot V'_{w_j}\right)}{\boxed{\sum_{w=1}^{N} \exp\left(V_{w_i} \cdot V'_{w}\right)}}$$

¡Costoso de calcular!



Skip-Gram con Negative Sampling

- Aproximamos $P(w_j|w_i)$ usando N clasificadores binarios (uno por cada contexto):

$$f_j(w_i) = \sigma\left(V_{w_i} \cdot V'_{w_j}\right)$$

- De modo que podemos escribir:

$$P(w_j|w_i) \approx f_j(w_i) \prod_{k \neq j} (1 - f_k(w_i))$$

- Luego, el conjunto de contextos "negativos" puede ser aproximado por un subconjunto de K palabras vocabulario tomado al azar $C_n = \{w_1, w_2, \dots, w_K : w_i \sim P_c\}, w_j \notin C_n$:

$$P(w_j|w_i) \approx f_j(w_i) \prod_{k \in C_n} (1 - f_k(w_i))$$

Skip-Gram con Negative Sampling

- Reemplazando la aproximación de la probabilidad en

$$L = -\sum_{i=1}^{|T|} \sum_{\substack{j=i-c \\ j \neq i}}^{i+c} \log P\left(w_j \middle| w_i\right)$$

- Nos queda:

$$L = -\sum_{i=1}^{|T|} \sum_{\substack{j=i-c \\ j \neq i}}^{i+c} \left(\log \sigma\left(V_{w_i} \cdot V'_{w_j}\right) - \sum_{k \in C_n} \log \sigma\left(V_{w_i} \cdot V'_k\right) \right)$$

Trucos bajo el capot

- La probabilidad de elegir una palabra para el conjunto de contextos negativos depende de una distribución de unigramas suavizada $\left(\alpha = \frac{3}{4}\right)$:

$$P_c(w) = \frac{\#(w)^{\alpha}}{\sum_{w'} \#(w')^{\alpha}}$$

- Las palabras del corpus son descartadas con probabilidad (parámetro $t = 10^{-5}$):

$$P_{\text{disc}} = 1 - \sqrt{\frac{t}{\#(w)}}$$

Entrenamiento de SGNS

- $L_{ij} = -\log\sigma\left(V_{w_i} \cdot V'_{w_j}\right) + \sum_{k \in C_n} \log\sigma(V_{w_i} \cdot V'_k)$

- Entrenamiento por SGD:

$$V_t = V_{t-1} - \eta \frac{\partial L_{ij}}{\partial V}$$

$$V'_t = V'_{t-1} - \eta \frac{\partial L_{ij}}{\partial V'}$$

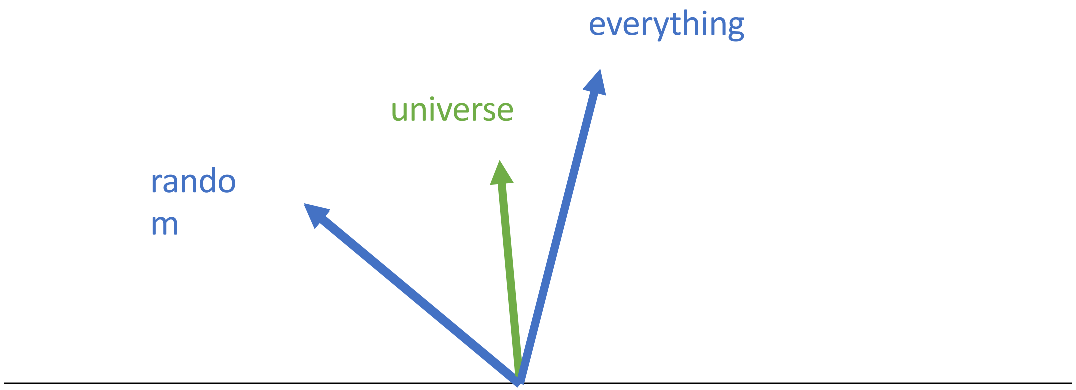
Entrenamiento de SGNS

$$\frac{\partial L_{ij}}{\partial V_{w_i}} = \left(\sigma\left(V'_{w_j} \cdot V_{w_i}\right) - 1\right) V'_{w_j} + \sum_{k \in C_n} \sigma(V'_{w_k} \cdot V_{w_i}) V'_{w_k}$$

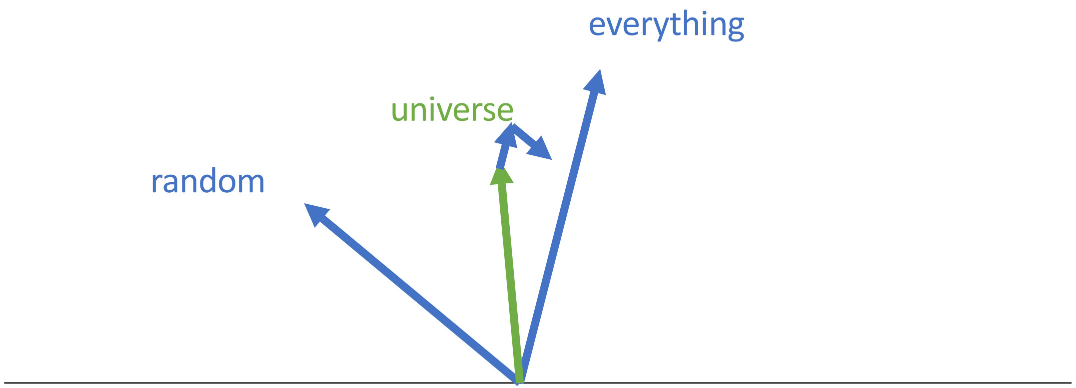
$$\frac{\partial L_{ij}}{\partial V'_{w_j}} = \left(\sigma\left(V'_{w_j} \cdot V_{w_i}\right) - 1\right) V_{w_i}$$

$$\frac{\partial L_{ij}}{\partial V'_{w_{k \in N_c}}} = \sigma\left(V'_{w_j} \cdot V_{w_i}\right) V_{w_i}$$

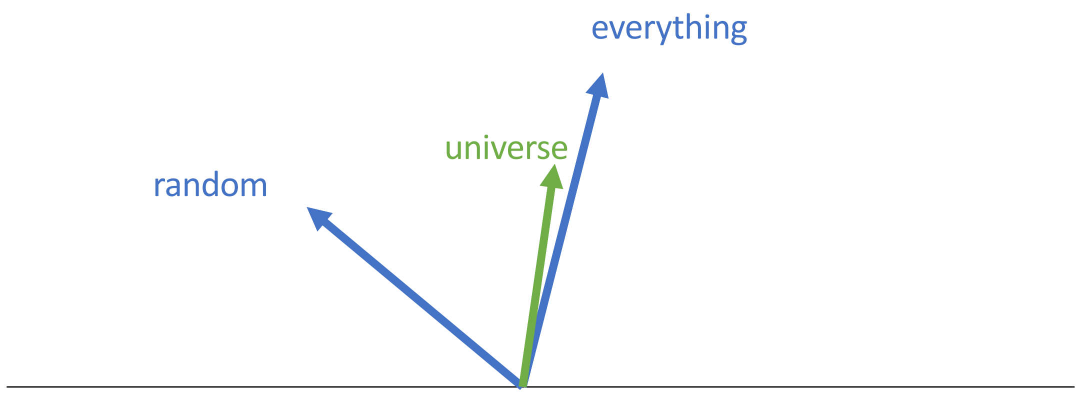
Interpretación geométrica



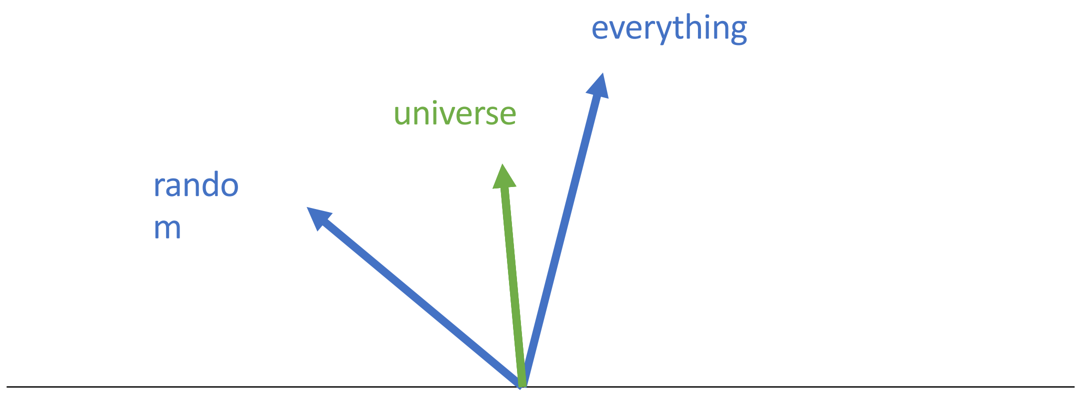
Interpretación geométrica



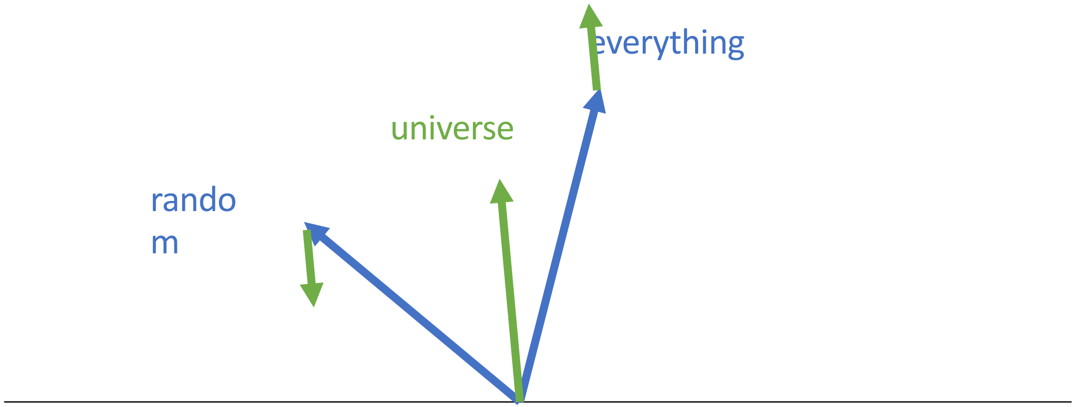
Interpretación geométrica



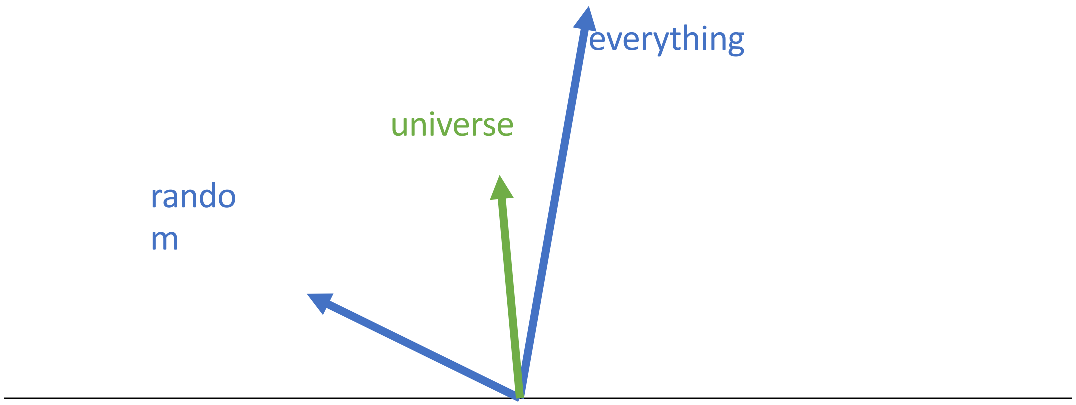
Interpretación geométrica



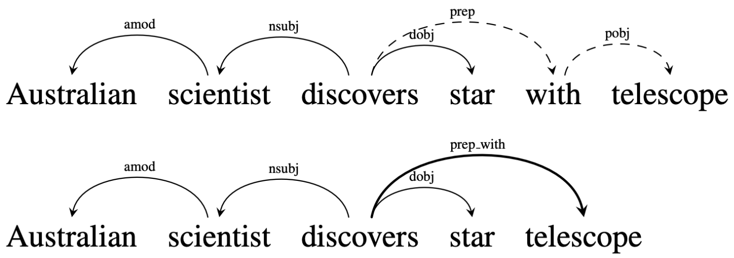
Interpretación geométrica



Interpretación geométrica



Dependency-based word-embeddings



WORD	CONTEXTS
australian	scientist/amod^{-1}
scientist	australian/amod, discovers/nsubj^{-1}
discovers	scientist/nsubj, star/dobj, telescope/prep_with
star	discovers/dobj^{-1}
telescope	discovers/prep_with^{-1}

Levy and Goldberg (2014)

Evaluación
Juicios de similitud

- Comparando los juicios de similitud entre palabras provistos por humanos con aquellos de los modelos (coseno).

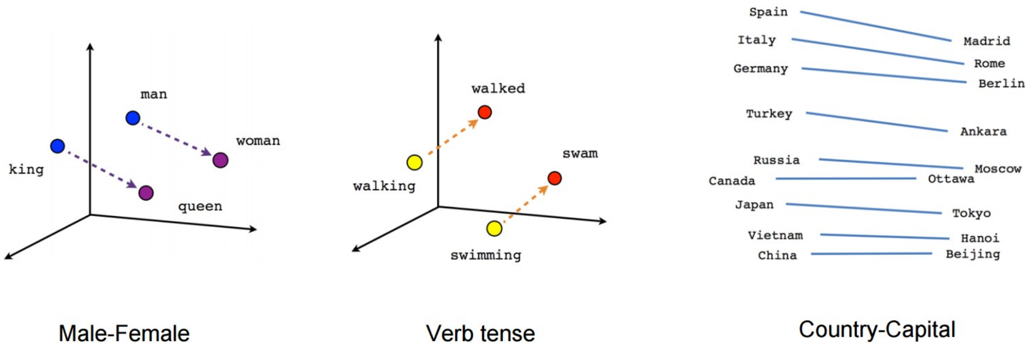
- Datasets:
  - WordSim353
  - MEN
  - Rare Words
  - SimLex

money	cash	9.15
Maradona	football	8.62
love	sex	6.77
smart	stupid	5.81
governor	interview	3.25
king	cabbage	0.23

- Figura de mérito: spearman r.

Evaluación
Analogías

- Semánticas: `hombre/mujer` como `rey/?`
- Sintácticas: `caminando/caminó` como `nadando/?`
- Figura de mérito: accuracy



(Crédito de la imagen: tensorflow)

Evaluación
Resolución de Analogías

- Versión aditiva :

$$\text{argmax}_{\text{w}\in W-\{\text{hombre,mujer,rey}\}} \cos\left(v_w, v_{\text{mujer}} - v_{\text{hombre}} + v_{\text{rey}}\right)$$

- Versión multiplicativa:

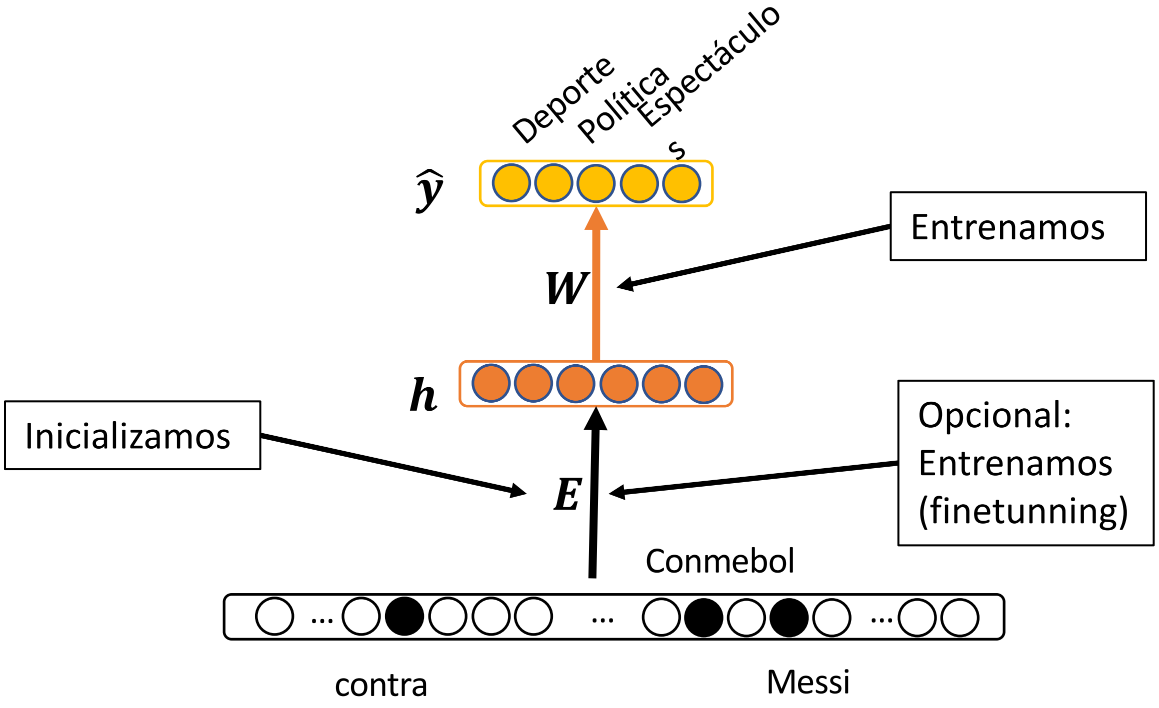
$$\text{argmax}_{\text{w}\in W-\{\text{hombre,mujer,rey}\}} \frac{\cos\left(v_w, v_{\text{mujer}}\right) + \cos(v_w, v_{\text{rey}})}{\cos(v_w, v_{\text{hombre}})}$$

pig:oink :: raven:nevermore

- pig : oink :: raven : nevermore
- pig : oink :: roadrunner : beep beep
- pig : oink :: bro : wassup
- pig : oink :: Homer Simpson : D’oh
- pig : oink :: Donald Trump : YOU’RE FIRED

https://graceavery.com/word2vec-fish-music-bass/

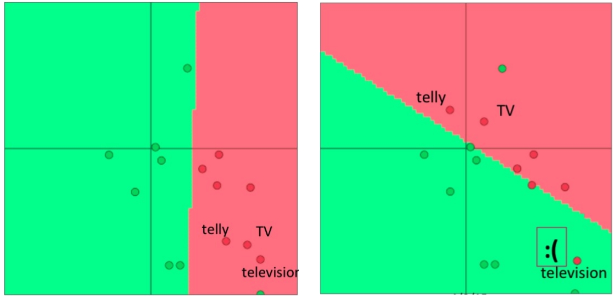
Aplicación de las representaciones semánticas distribuídas



Finetunning

- Cuando puedo entrenar los embeddings en mi tarea:
  - Tengo muchos datos
  - No hay palabras en el vocabulario de evaluación que no estén en el vocabulario de entrenamiento

- Si alguna de esas condiciones no se cumplen, mejor dejarlos donde están.



(crédito: stanford)

Día 2: Word embeddings y redes neuronales multi-capa

1. El significado de las palabras

2. Redes neuronales multi-capa y backpropagation

3. Elementos prácticos

Problemas en el PLN: Natural Language Inference (NLI)

S_a: Una tortuga marina está cazando peces

$\Rightarrow$

S_b: Una tortuga está buscando comida

S_a: Una tortuga marina está cazando peces

↯

S_b: Unos peces están cazando una tortuga.

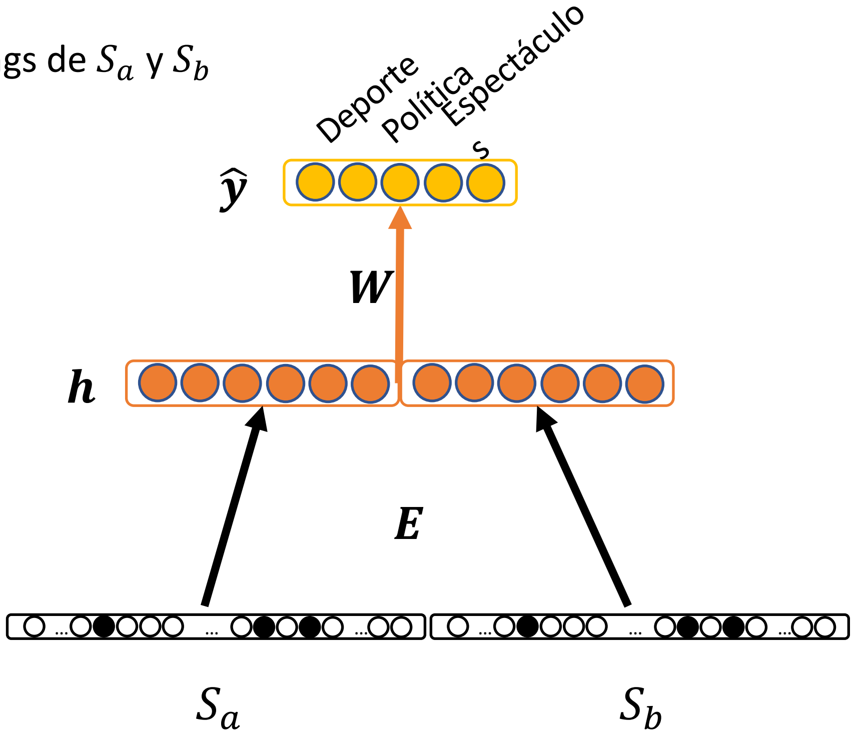
Objetivo: entrenar un clasificador

$$\hat{y}(S_a, S_b) = \begin{cases} 1\,, & si\ S_a \Rightarrow S_b \\ 0, & si\ S_a \nRightarrow S_b \end{cases}$$

NLI con clasificador lineal

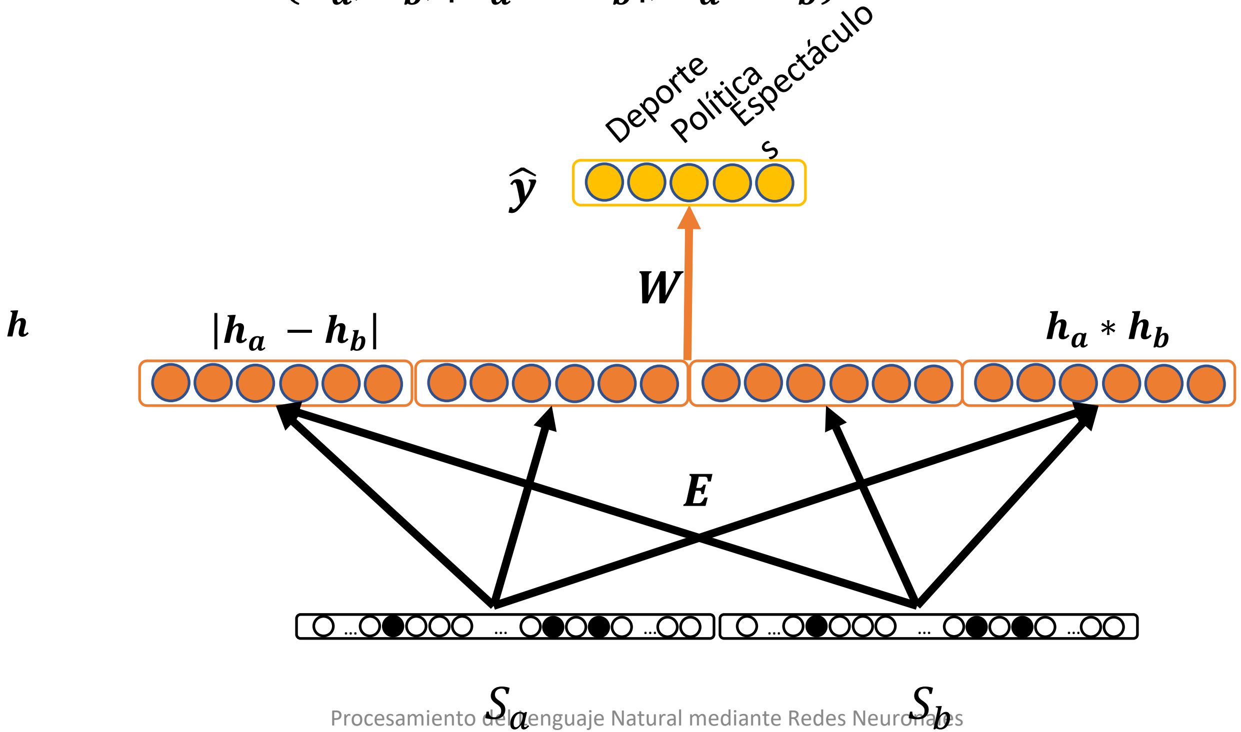
$\boldsymbol{h_a}, \boldsymbol{h_b}$: bag of embeddings de S_a y S_b

$\boldsymbol{h} = \mathrm{concat}(\boldsymbol{h_a}, \boldsymbol{h_b})$



NLI con clasificador lineal

$$\boldsymbol{h} = \mathrm{concat}(\boldsymbol{h_a}, \boldsymbol{h_b}, |\boldsymbol{h_a} - \boldsymbol{h_b}|, \boldsymbol{h_a} * \boldsymbol{h_b})$$

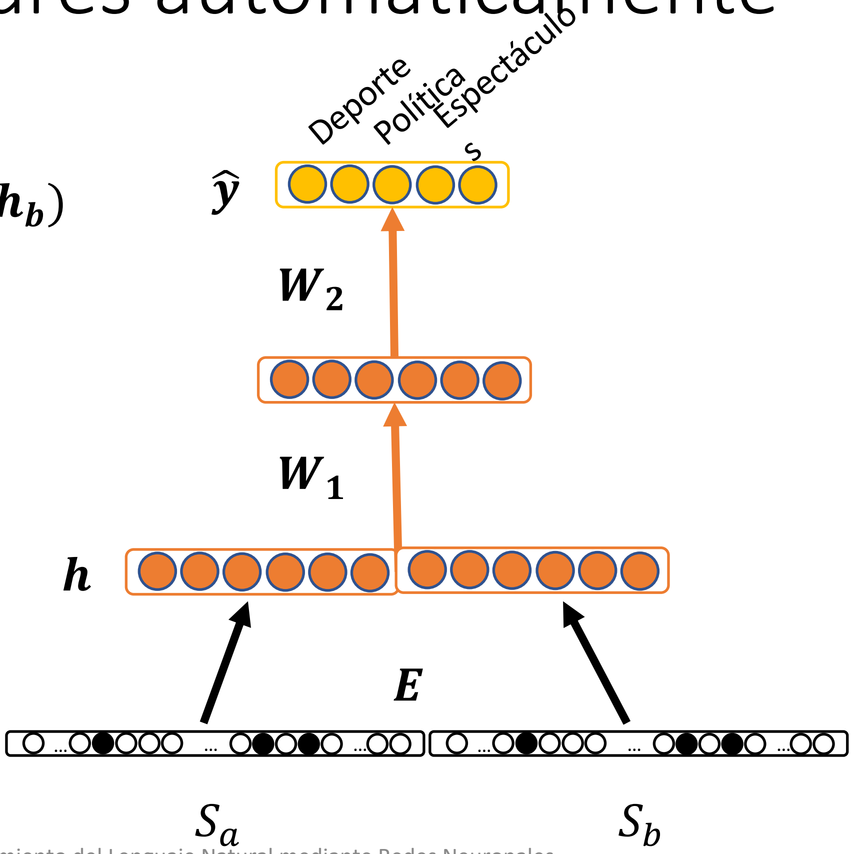


Extrayendo features automáticamente

$$\boldsymbol{h_1} = \text{concat}(\boldsymbol{h_a}, \boldsymbol{h_b})$$

$$\boldsymbol{h_2} = \boldsymbol{W_1 h_1}$$

$$\boldsymbol{\hat{y}} = \boldsymbol{\sigma}(\boldsymbol{W_2 h_2})$$



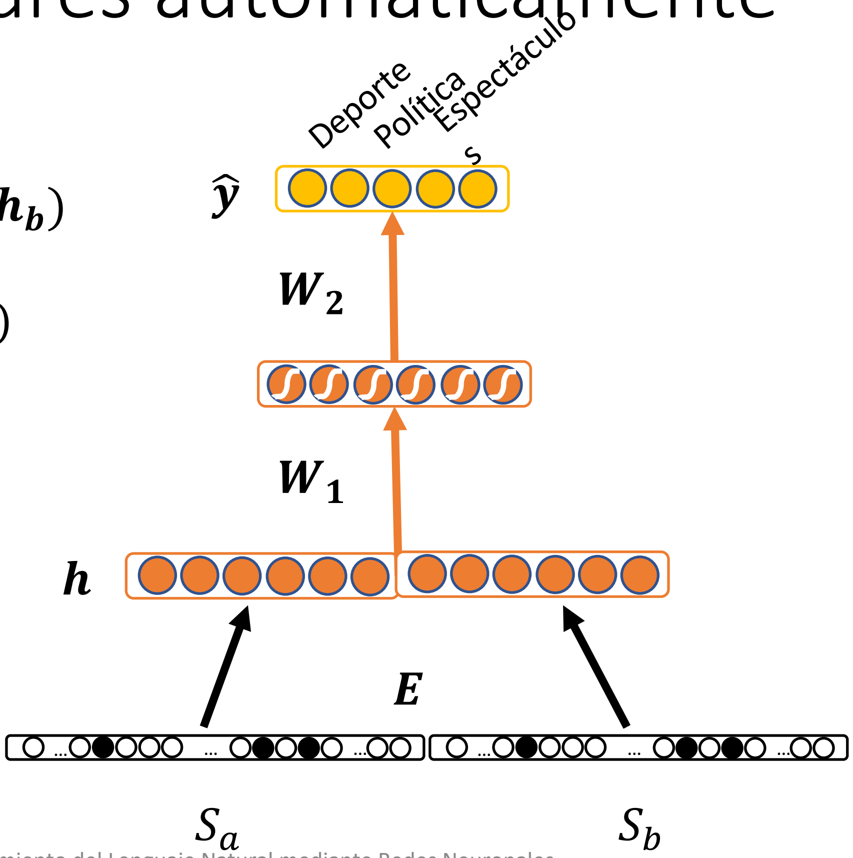
Extrayendo features automáticamente

$$\boldsymbol{h_1} = \text{concat}(\boldsymbol{h_a}, \boldsymbol{h_b})$$

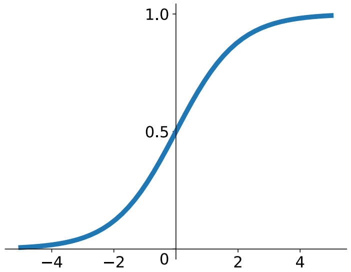
$$\boldsymbol{h_2} = f(\boldsymbol{W_1 h_1})$$

$$\boldsymbol{\hat{y}} = \boldsymbol{\sigma}(\boldsymbol{W_2 h_2})$$

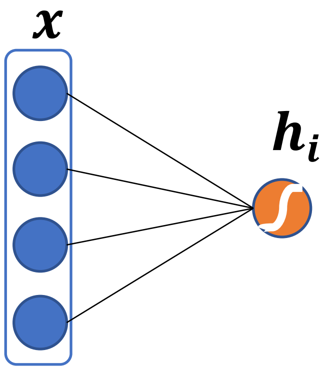
Función no-lineal (función de activación)



Función de activación sigmoid



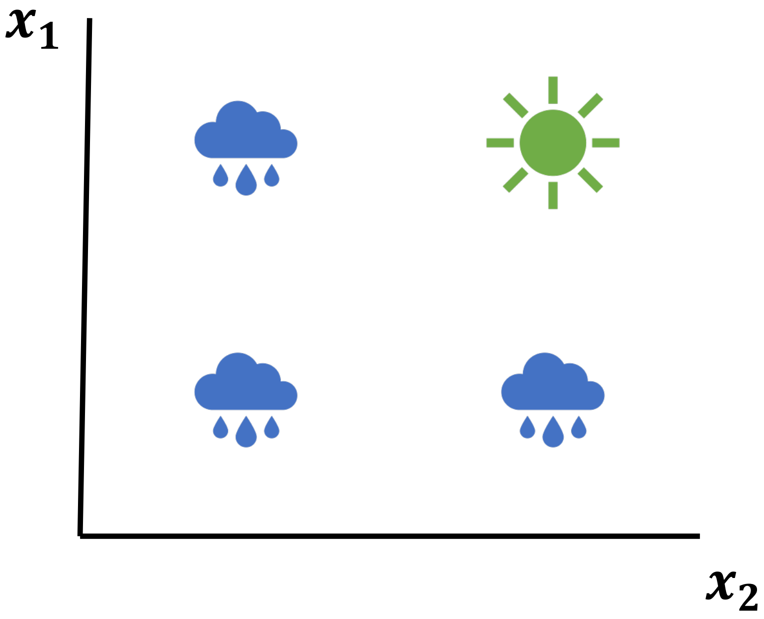
$$\sigma(x) = \frac{1}{1 + e^{-x}}$$



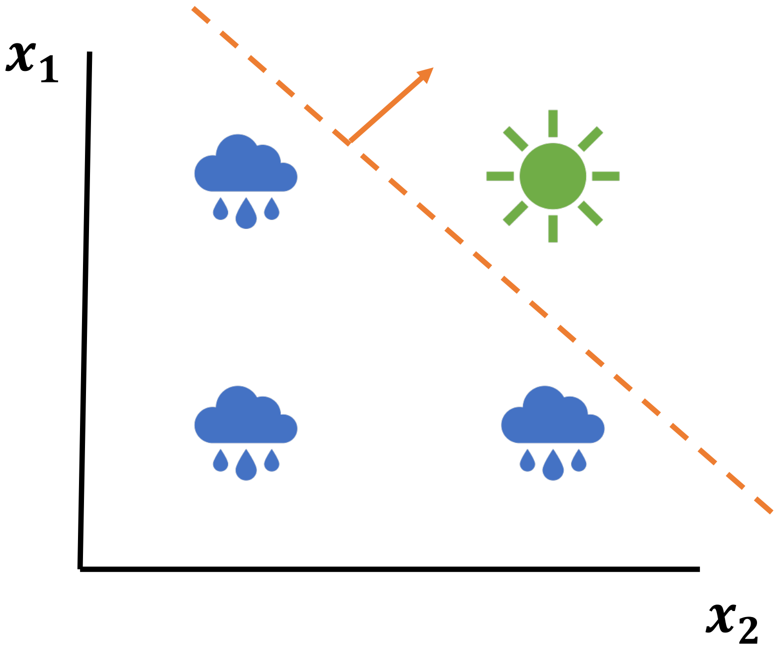
Por ejemplo:

$$\boldsymbol{h} = \sigma(\boldsymbol{W_1 x})$$

Representación de predicados lógicos

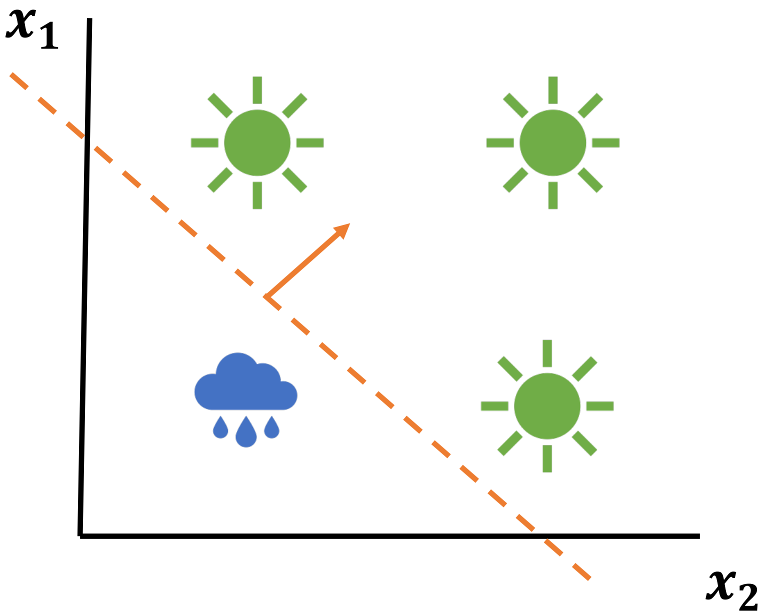


Representación de predicados lógicos



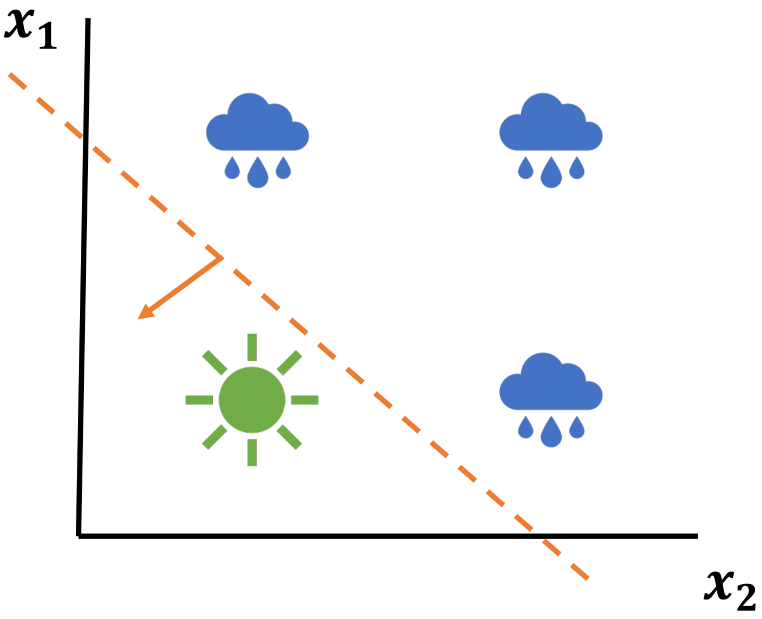
x_1 **y** x_2 con valores grandes predicen sol

Representación de predicados lógicos



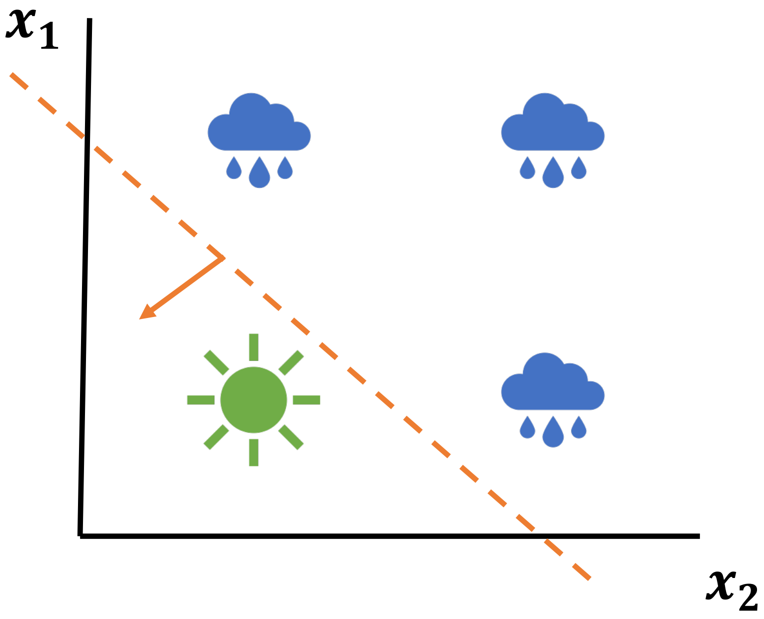
x_1 **ó** x_2 con valores grandes predice sol

Representación de predicados lógicos



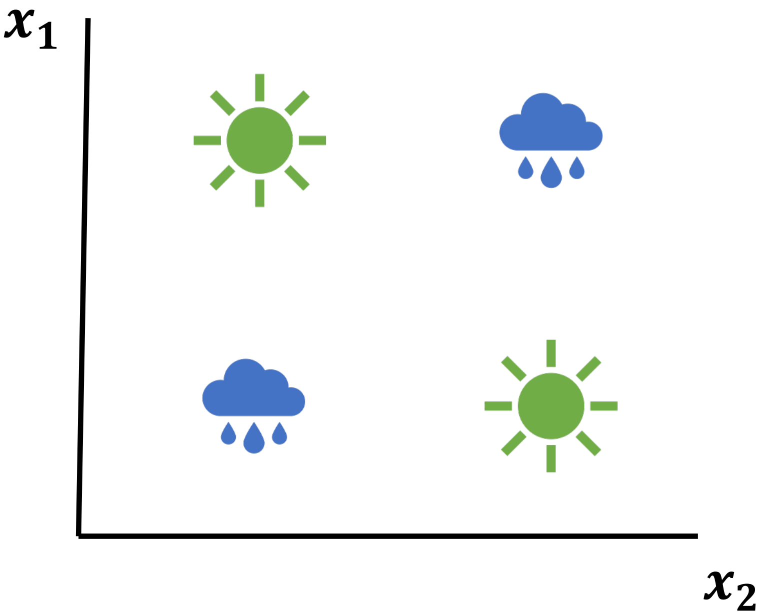
x_1 y x_2 **no** tienen valores grandes predice sol

Representación de predicados lógicos



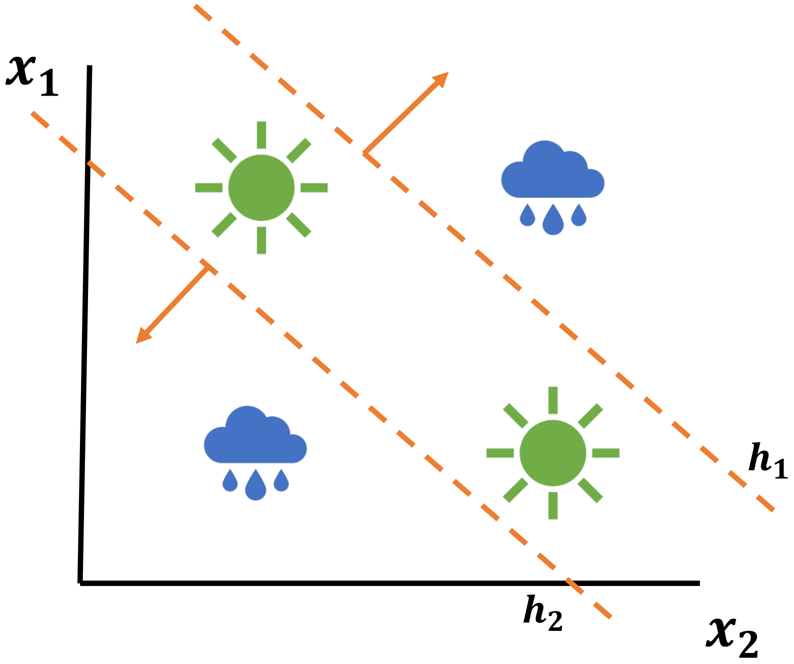
x_1 y x_2 **no** tienen valores grandes predice sol

Representación de predicados lógicos



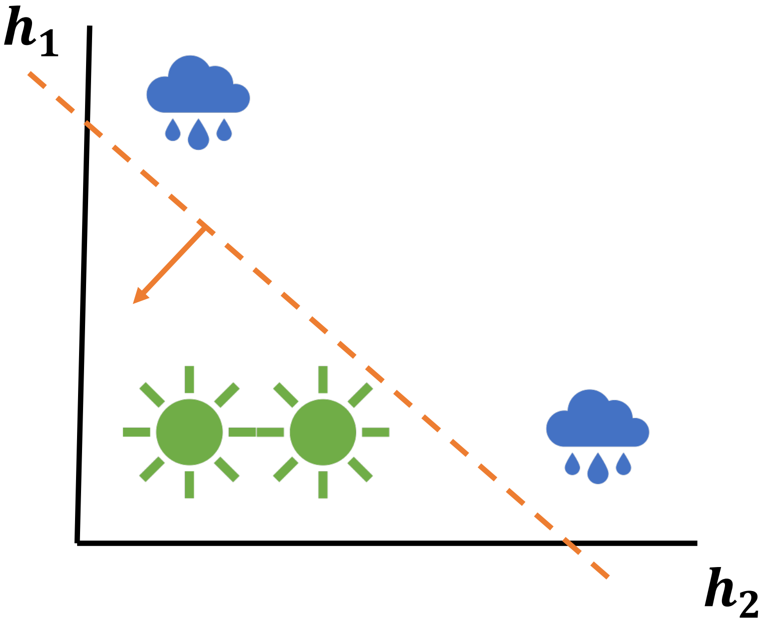
x_1 **ó** x_2 con valores grandes, pero no ambos predice sol

Representación de predicados lógicos



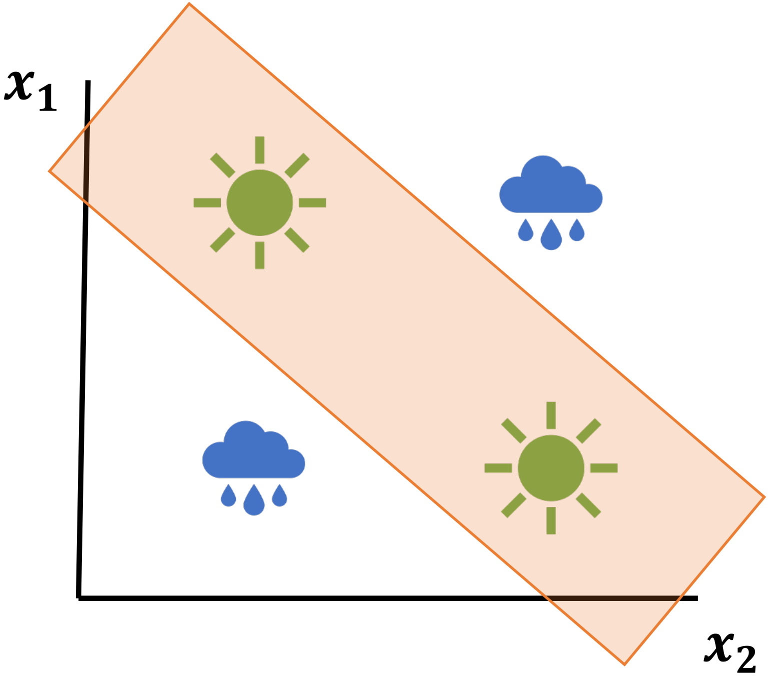
h_1: x_1 y x_2 con valores grandes predicen lluvia
h_2: x_1 y x_2 con valores pequeños predicen lluvia

Representación de predicados lógicos



h_1 y h_2 con valores pequeños predicen sol

Clasificador no lineal

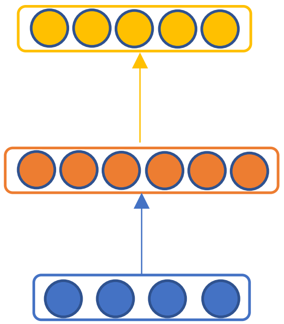


Ni x_1 y x_2 tienen ambos valores grandes (h_1)
Ni x_1 y x_2 tienen ambos valores pequeños (h_2)

Red neuronal de una capa

$$\boldsymbol{h} = \sigma(\boldsymbol{W_1}\boldsymbol{x})$$

$$\hat{\boldsymbol{y}} = \text{softmax}(\boldsymbol{W_2}\boldsymbol{h})$$



- Puede representar cualquier función[1].
- Pero puede requerir un número exponencialmente grande de unidades en $\boldsymbol{h}$.

[1] http://neuralnetworksanddeeplearning.com/chap4.html

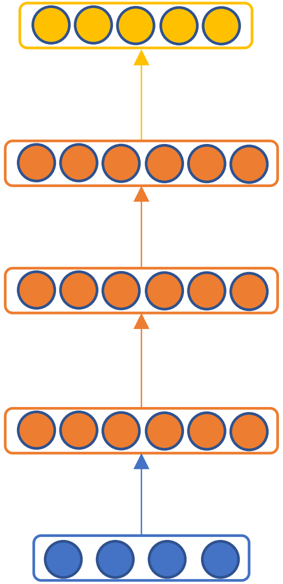
Extendiendo a varias capas

$$\boldsymbol{h_1} = \sigma(\boldsymbol{W_1 x})$$

$$\boldsymbol{h_2} = \sigma(\boldsymbol{W_2 h_1})$$

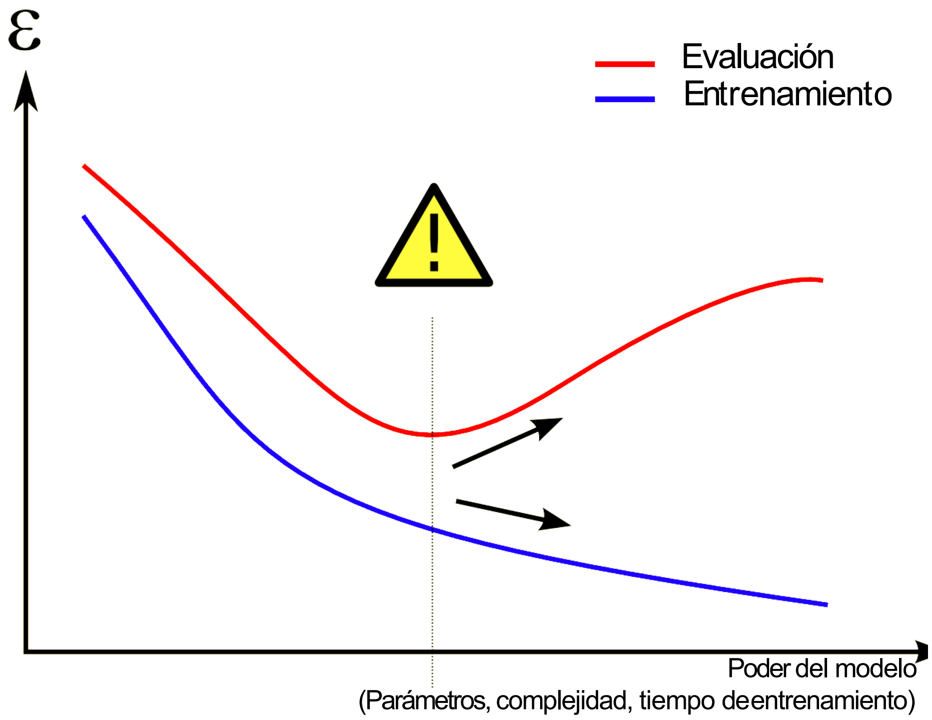
$$\boldsymbol{h_3} = \sigma(\boldsymbol{W_3 h_2})$$

$$\hat{\boldsymbol{y}} = \text{softmax}(\boldsymbol{W_4 h_3})$$

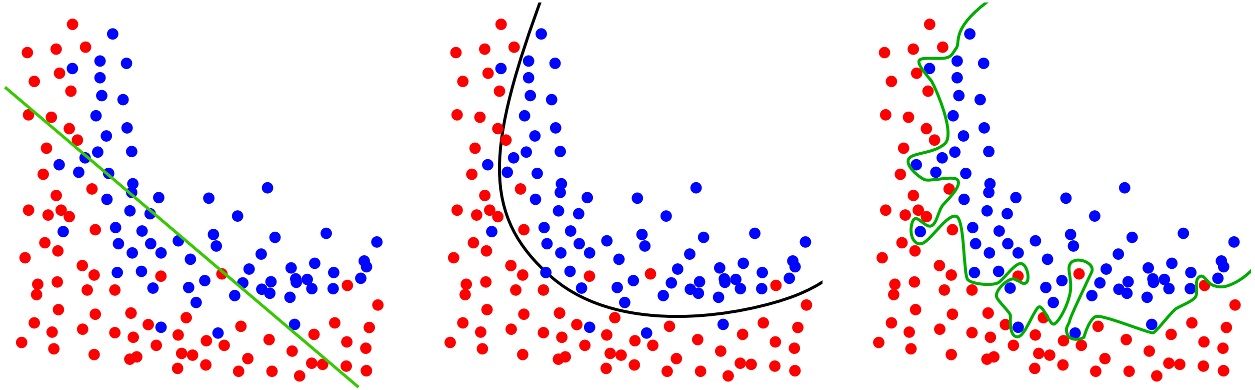


- Puede encontrar patrones más complejos.
- Más difícil de entrenar: no necesariamente va a encontrarlos.

Overfitting



Balance entre overfitting y underfitting



(a) Underfitting (b) Buen modelo (c) Overfitting

Entrenamiento por GD

- Los parámetros de la red se ajustan para minimizar el error

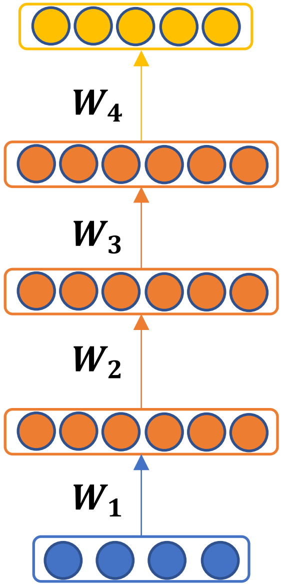
$$\boldsymbol{W}_1 \leftarrow \boldsymbol{W}_1 - \eta \boxed{\frac{\partial L(y,\hat{y})}{\partial \boldsymbol{W}_1}}$$

$$\boldsymbol{W}_2 \leftarrow \boldsymbol{W}_2 - \eta \boxed{\frac{\partial L(y,\hat{y})}{\partial \boldsymbol{W}_2}}$$

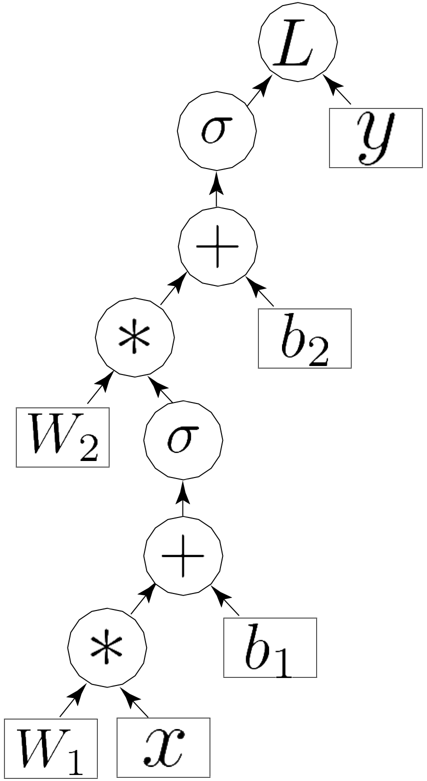
$$\boldsymbol{W}_3 \leftarrow \boldsymbol{W}_3 - \eta \boxed{\frac{\partial L(y,\hat{y})}{\partial \boldsymbol{W}_3}}$$

$$\boldsymbol{W}_4 \leftarrow \boldsymbol{W}_4 - \eta \boxed{\frac{\partial L(y,\hat{y})}{\partial \boldsymbol{W}_4}}$$

Tenemos que calcular

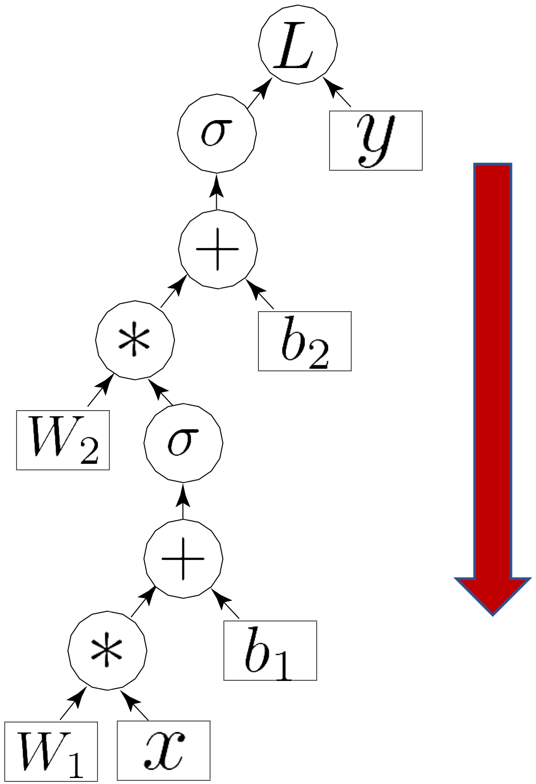


Grafo de cómputo

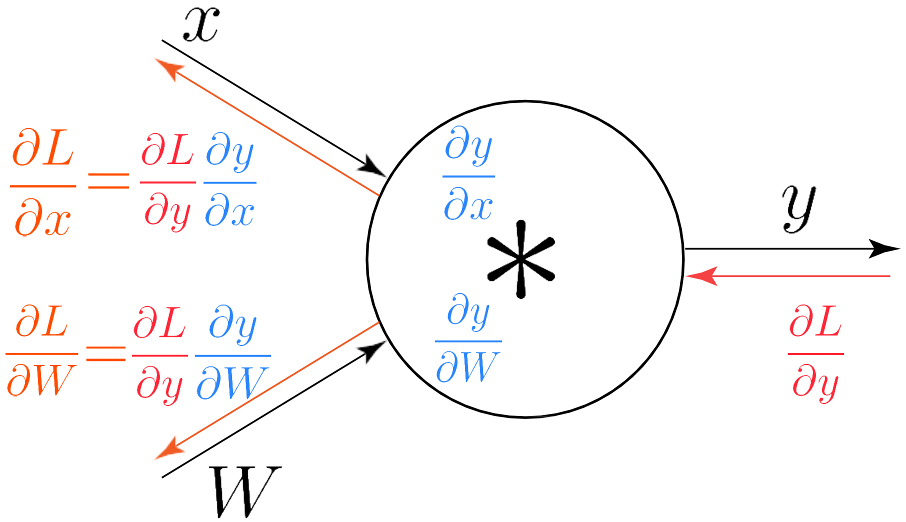


Backpropagation

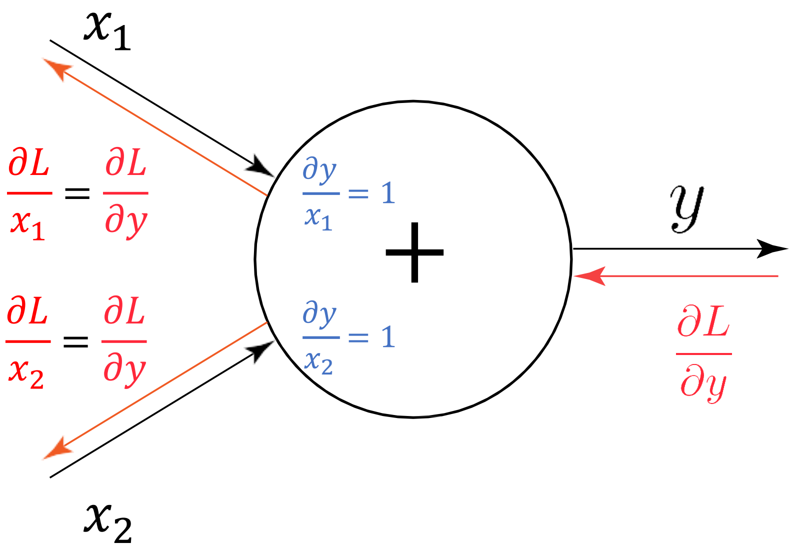
- Recorre el grafo de cómputo en dirección inversa.
- Calcula el gradiente del error con respecto a cada cálculo intermedio y cada parámetro.



Backpropagation en el grafo de cómputo

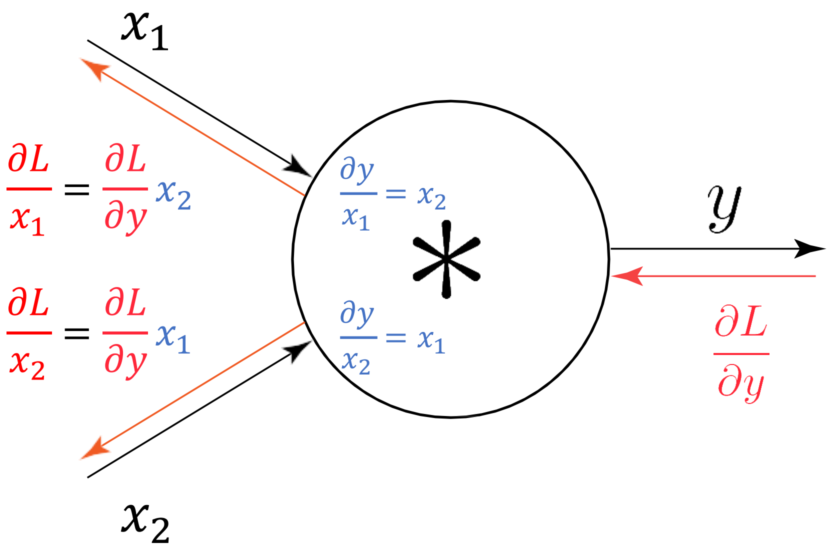


Backpropagation en la suma



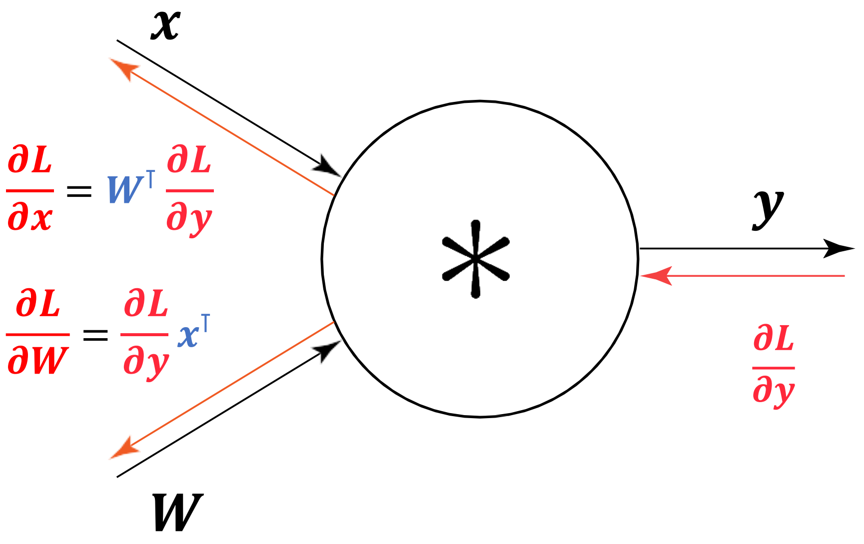
$$y = x_1 + x_2$$

Backpropagation en la multiplicación



$$y = x_1 * x_2$$

Backpropagation en la transformación lineal



$$\boldsymbol{y} = \boldsymbol{W}\boldsymbol{x}$$

Backpropagation en la transformación lineal

$$\frac{\partial L}{\partial W_{ij}} = \frac{\partial L}{\partial y_i} \underbrace{\frac{\partial y_i}{\partial W_{ij}}}_{x_j}$$

$$\frac{\partial \boldsymbol{L}}{\partial \boldsymbol{W}} = \frac{\partial \boldsymbol{L}}{\partial \boldsymbol{y}} \boldsymbol{x}^{\intercal}$$

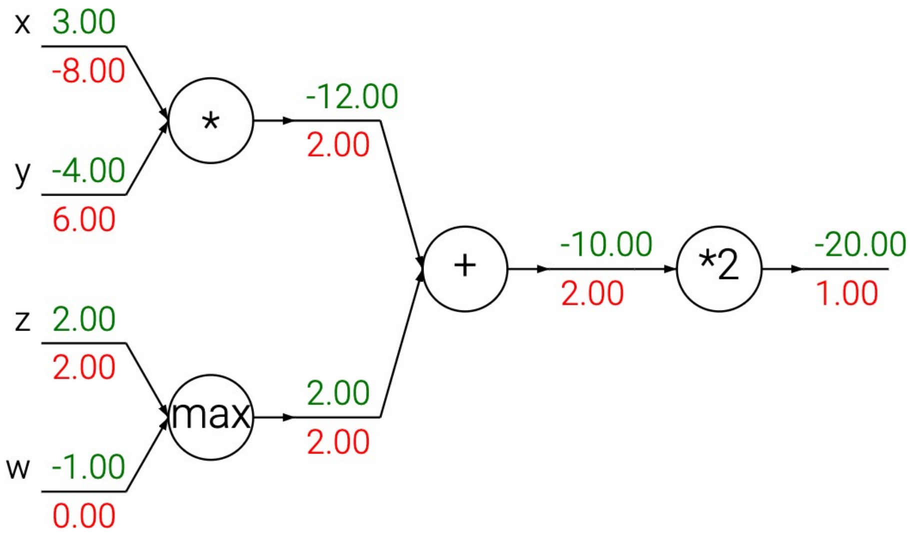
$$\frac{\partial L}{\partial x_j} = \sum_i \frac{\partial L}{\partial y_i} \underbrace{\frac{\partial y_i}{\partial x_j}}_{W_{ij}}$$

$$\frac{\partial L}{\partial x_j} = \boldsymbol{W}^{\intercal}_{\cdot j} \frac{\partial \boldsymbol{L}}{\partial \boldsymbol{y}}$$

$$\frac{\partial \boldsymbol{L}}{\partial \boldsymbol{x}} = \boldsymbol{W}^{\intercal} \frac{\partial \boldsymbol{L}}{\partial \boldsymbol{y}}$$

$$\boldsymbol{y} = \boldsymbol{W}\boldsymbol{x}$$

Ejemplo



credito: stanford

Frameworks con grafos de cómputo dinámicos

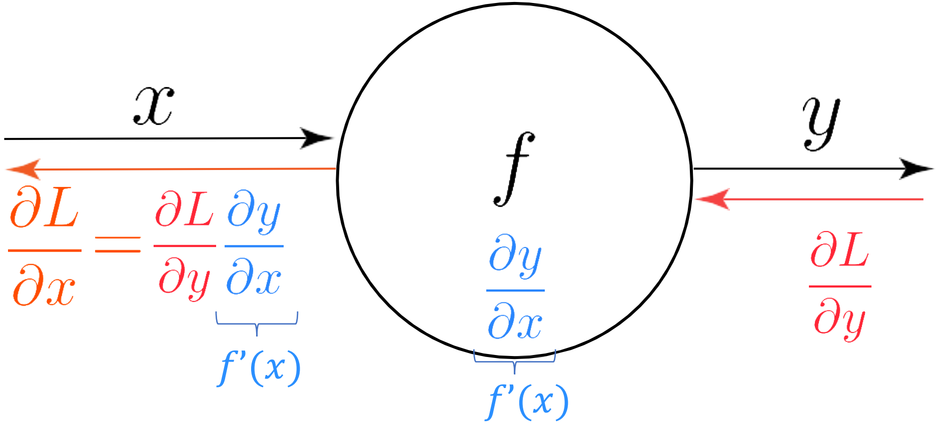
PyTorch, Chainer, Theano, DyNet son algunos ejemplos.

```python
import torch

x = torch.tensor(3.0, requires_grad=True)
y = torch.tensor(-4.0, requires_grad=True)
z = torch.tensor(2.0, requires_grad=True)
w = torch.tensor(0.0, requires_grad=True)

r = ((x * y) + (torch.max(z, w)))* 2

r.backward()  # backpropagation

print("∂r/∂x", x.grad)  # ∂r/∂x tensor(-8.)
print("∂r/∂y", y.grad)  # ∂r/∂y tensor(6.)
print("∂r/∂z", z.grad)  # ∂r/∂z tensor(2.)
print("∂r/∂w", w.grad)  # ∂r/∂w tensor(0.)
```

Las operaciones de antes implementadas en PyTorch

Funciones de activación

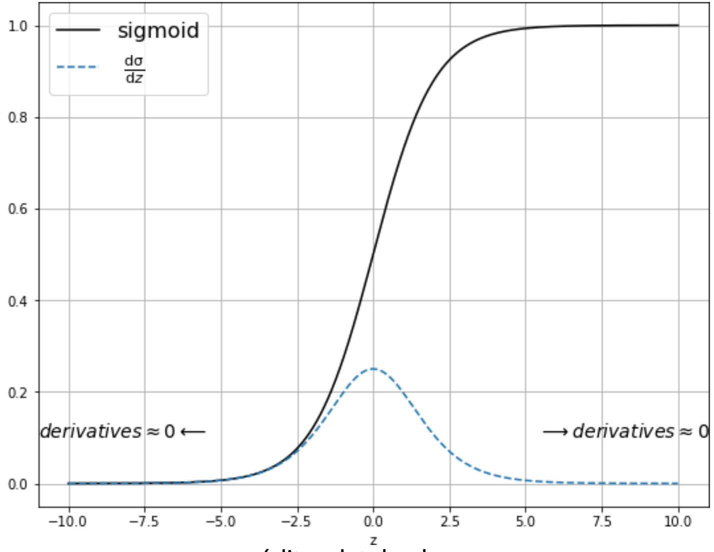
- La función de activación f se aplicada componente a componente.
- Para obtener el gradiente, se multiplica la derivada f' componente a componente.



Funciones de activación
Sigmoid

$$\sigma(x) = \frac{1}{1 + \exp(-x)}$$

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



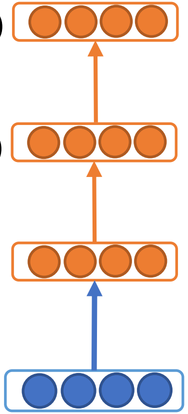
crédito: datahacker.rs

Vanishing gradient

$h_3 = \sigma(W_3 h_2)$

$h_2 = \sigma(W_2 h_1)$

$h_1 = \sigma(W_1 x)$



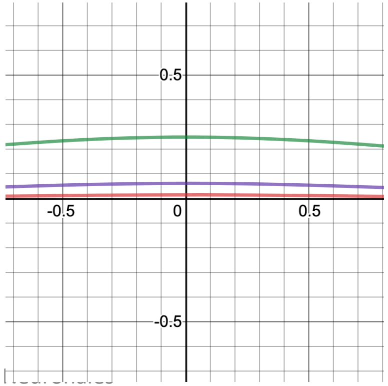
$$\frac{\partial L}{\partial h_3} \propto \sigma'(W_3 h_2)$$

$$\frac{\partial L}{\partial h_2} \propto \sigma'(W_3 h_2) W_3^{\intercal} \sigma'(W_2 h_1)$$

$$\frac{\partial L}{\partial h_1} \propto \sigma'(W_3 h_2) W_3^{\intercal} \sigma'(W_2 h_1) W_2^{\intercal} \sigma'(W_1 x)$$

$\sigma'(x)$ $\sigma'(x)^2$ $\sigma'(x)^3$

Potencias de la derivada de sigmoid



http://neuralnetworksanddeeplearning.com/chap5.html

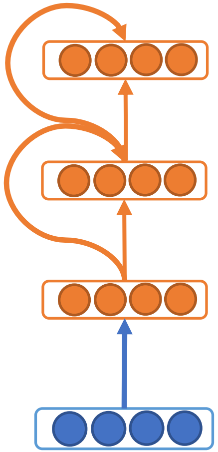
Conexiones residuales

$$h_3 = \boxed{\sigma(W_3 h_2)} \boxed{+ h_2}$$

residuo

$$h_2 = \boxed{\sigma(W_2 h_1)} \boxed{+ h_1}$$

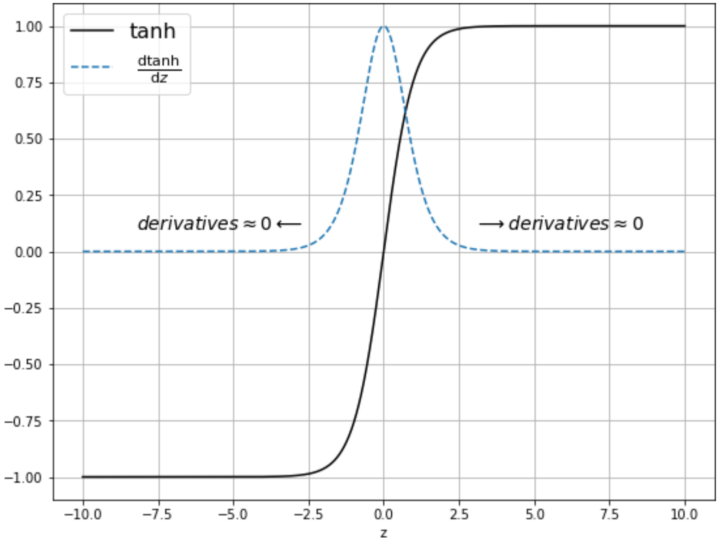
$$h_1 = \sigma(W_1 x)$$



Funciones de activación
Sigmoid

$$\tanh(x) = \frac{\exp(x) - \exp(-x)}{\exp(x) + \exp(-x)} = 2\sigma(x) - 1$$

$$\tanh'(x) = 1 - \tanh(x)^2$$

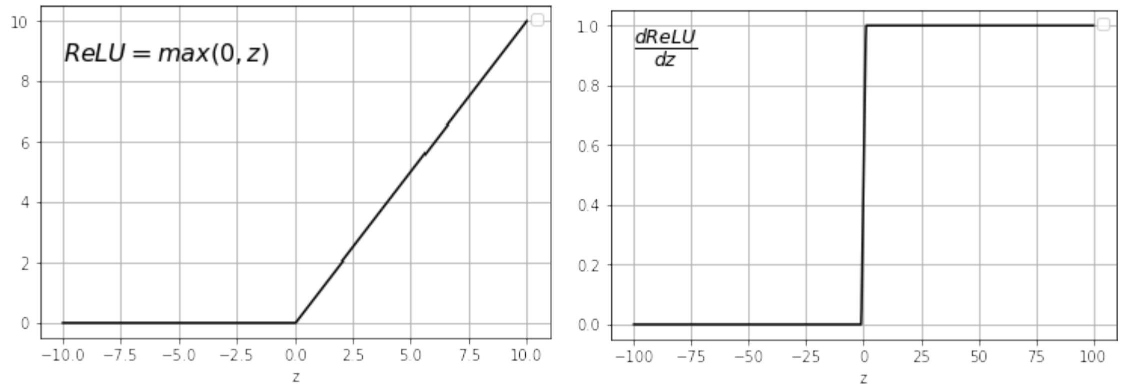


crédito: datahacker.rs

Funciones de activación
Sigmoid

$$\mathrm{ReLU}(x) = \max(x, 0)$$

$$\mathrm{ReLU'}(x) = \begin{cases} 1, & \text{si } x > 0 \\ 0, & \text{si } x < 0 \end{cases}$$

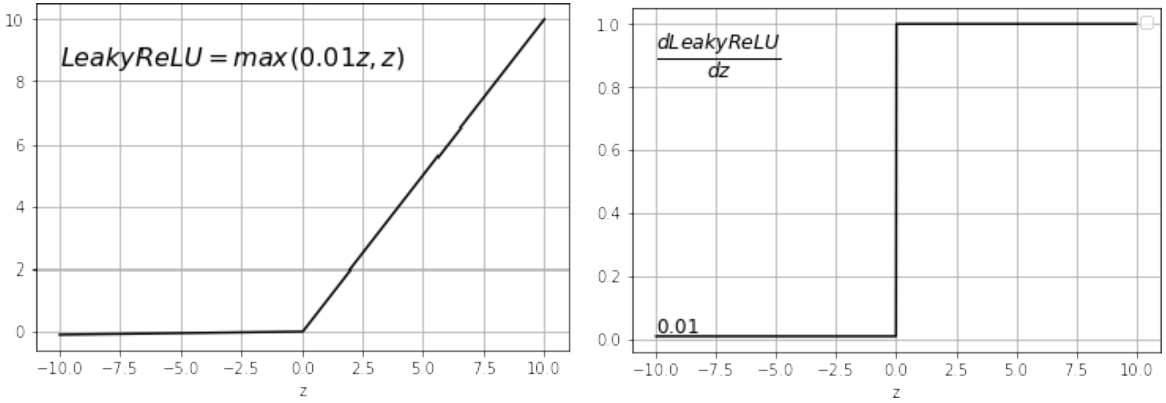


crédito: datahacker.rs

Funciones de activación
Sigmoid

$$\text{LeakyReLU}(x) = \max(x, 0.01)$$

$$\text{LeakyReLU'}(x) = \begin{cases} 1, & \text{si } x > 0 \\ 0.01, & \text{si } x < 0 \end{cases}$$



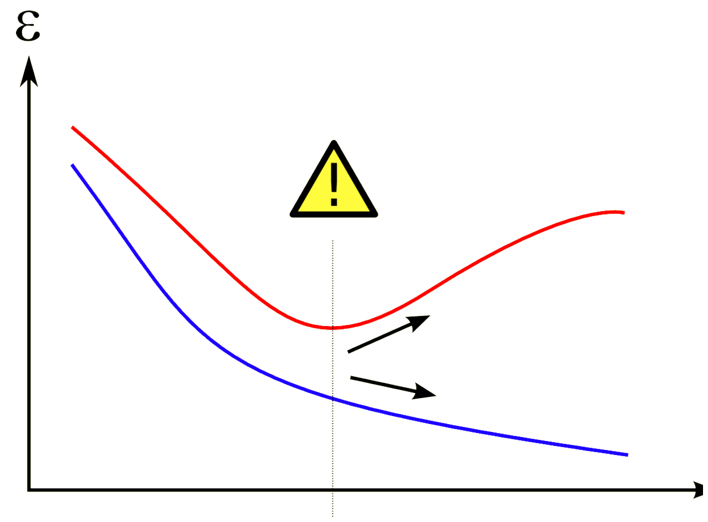
crédito: datahacker.rs

Día 2: Word embeddings y redes neuronales multi-capa

1. El significado de las palabras
2. Redes neuronales multi-capa y backpropagation
3. Elementos prácticos

Tecnicas de regularización
Early stopping

- Medimos la función de error en el set de validación después cada epoch. Si comienza a crecer, paramos.



Técnicas de regularización L1/L2

- Pedimos que el valor de los parámetros de la red sea pequeño, agregando un término de penalización a la función objetivo.

$$J(\Theta) = L(\hat{y}_\Theta, y) + \lambda\,\Omega(\Theta)$$

- L2: $\Omega(\Theta) = \|\Theta\|_2$
- L1: $\Omega(\Theta) = \|\Theta\|_1$
- Elastic Net: $\Omega(\Theta) = \lambda_1\|\Theta\|_1 + \lambda_2\|\Theta\|_2$

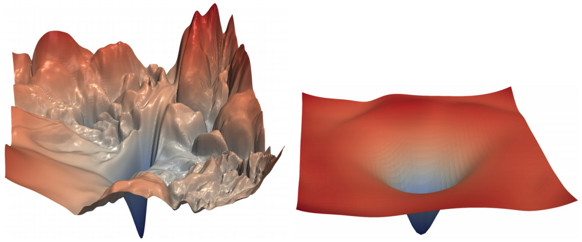
Tecnicas de regularización
Dropout

- Se aplica sobre una matriz de pesos $\boldsymbol{W}$.

- En cada aplicación de la red, con probabilidad p, se descarta temporalmente el peso W_{ij}.

Inicialización de los parámetros

- Es importante inicializar los parámetros en un buen punto.



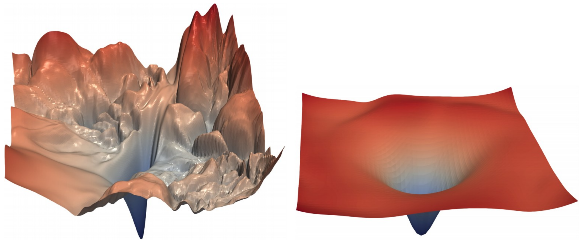
Li et al. (2018)

- Xavier initialization (Golorot and Bengio, 2010)

$$\Theta_i \sim U\left[-\frac{\sqrt{6}}{\sqrt{d_{in}+d_{out}}}, \frac{\sqrt{6}}{\sqrt{d_{in}+d_{out}}}\right]$$

Inicialización de los parámetros

- Es importante inicializar los parámetros en un buen punto.



Li et al. (2018)

- Keiming initialization (He et al., 2015):

$$\Theta_i \sim \mathcal{N}\left(0, \frac{2}{\sqrt{d_{in}}}\right)$$

Restarts y ensembles

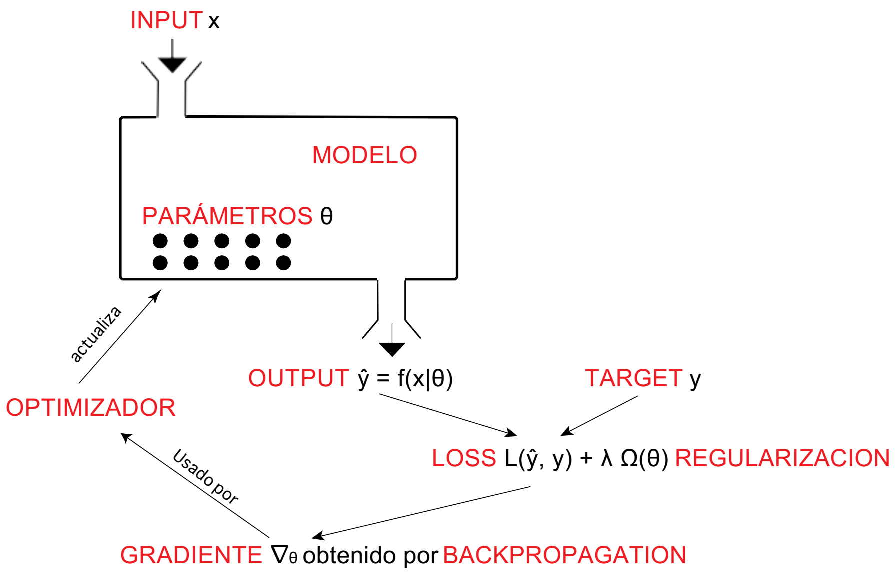
- Distintas inicializaciones llevan a soluciones distintas

- Podemos entrenar varios modelos empezando de puntos distintos (random seed) y:

  - Quedarnos con el que funciona mejor en validación (Útil en aplicaciones prácticas. Con ojo en research)

  - Usar la predicción promedio (ensemble).

Entrenamiento de una red neuronal en una precaria imagen



Resumen

- Podemos aprovechar grandes cantidades de texto para producir **representaciones semánticas de las palabras**.
- Aplicamos estas representaciones en una red neuronal aplicada a una tarea específica, ya sea como **representaciones fijas o como punto de partida**.
- Una red neuronal se construye como **composición de funciones** (generalmente involucrando transformaciones lineales y funciones de activación)
- **Backpropagation** calcula los gradientes. **SGD** actualiza los parámetros.
- Feedback Día 2: https://forms.gle/cPG2mAgQrcCtVwa98

Referencias

- Word2vec explained: https://arxiv.org/pdf/1411.2738.pdf
- Derivation of Backpropagation: https://www.cs.swarthmore.edu/~meeden/cs81/s10/BackPropDeriv.pdf.
- Bottou et al. (2012) Stochastic gradient descent tricks: https://cilvr.cs.nyu.edu/diglib/lsml/bottou-sgd-tricks-2012.pdf
- LeCun et al. (1998) Efficient Backprop http://yann.lecun.com/exdb/publis/pdf/lecun-98b.pdf
- Mikolov et al. (2013) Efficient estimation of word representations in vector space: https://arxiv.org/pdf/1301.3781