Detección de Discurso de Odio (Laboratorio)

IntroPLN – Grupo de PLN - InCo

Agenda

- Laboratorio: Detección de Discurso de Odio
- Intro. Redes Neuronales
- Repaso. Word Embeddings
- Intro. Python, numpy, sklearn, keras

Laboratorio: Detección de Odio

Detección de Discurso de Odio

Hacer un algoritmo que detecte si un texto tiene discurso de odio.

Ejemplos

1. Significa que estás muy sonsa y necesitas ayuda

2. Callate la jeta guebon, o mejor muerete chavista de mierda!!!

3. Que pueden esperar de un negro Así, igual que el hermano, lo echaron de Boca por chorro.

Detección de Discurso de Odio

Hacer un algoritmo que detecte si un texto tiene discurso de odio.

Ejemplos

1. Significa que estás muy sonsa y necesitas ayuda

2. Callate la jeta guebon, o mejor muerete chavista de mierda!!!

3. Que pueden esperar de un negro Así, igual que el hermano, lo echaron de Boca por chorro.

Detección de Discurso de Odio

Vamos a usar un conjunto de datos anotado manualmente (**corpus**).

El corpus esta partido en **train** (entrenamiento), **val** (validación) y **test** (evaluación).

Evaluación con **Precision**, **Recall** y **F**

Modalidad del Laboratorio

Construir un programa teniendo en cuenta lo siguiente:

1- Respetar la interfaz definida en la letra del lab
> python3 es_odio.py **<data_path> test_file1.csv** … **test_fileN.csv**

2- usar los recursos impartidos
- **cropus**: train, val, test
- **word embeddings**

El programa tiene que
1- Entrenar usando los corpus train y val en **<data_path>**
2- Usar opcionalmente los word embeddings almacenados en **<data_path>**
3- Aplicar el modelo entrenado en cada uno de los **test_fileX.csv X=1…N**
4- Generar por cada archivo de test un archivo **test_fileX.out** con las salidas obtenidas

Laboratorio: Corrección

Entrega:
- código (es_odio.py y .py adicionales)
- readme.txt describiendo que hicieron
- grupo.txt con número de grupo y c.i. de integrantes
- test.out con la salida de la ejecución de lo entregado

Corrección:
- se ejecuta el código
- se evalua en un conjunto de test no conocido

Los 3 mejores grupos tienen 5 puntos extra!

Laboratorio: ¿Qué pueden usar?

sklearn
clasificadores como SVM, Naive Bayes, MLP
Facilidades como: pipeline, búsqueda de hiperparámetros

Keras o pytorch
Redes neuronales densas (fully connected)
Recurrentes, LSTM, GRU, Attention
Convolucionales, etc.

Corpus y los word embeddings pre-entrenados

Redes Neuronales Artificiales

2. Classification setup and notation

- Generally we have a training dataset consisting of samples

$$\{x_i, y_i\}^N_{i=1}$$

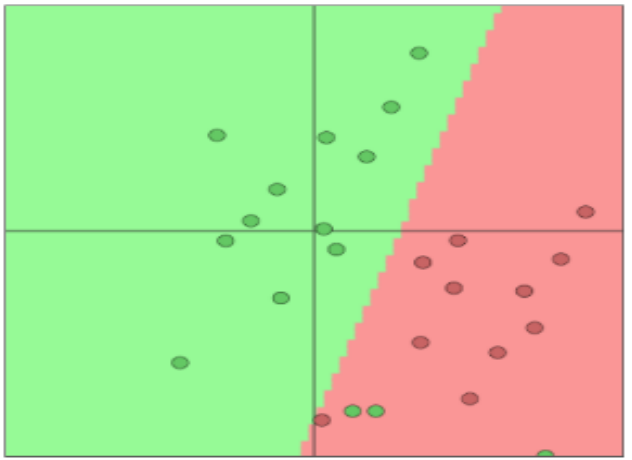
- x_i are inputs, e.g. words (indices or vectors!), sentences, documents, etc.
  - Dimension d

- y_i are labels (one of C classes) we try to predict, for example:
  - classes: sentiment, named entities, buy/sell decision
  - other words
  - later: multi-word sequences

Extraído del curso de Stanford http://web.stanford.edu/class/cs224n/slides/cs224n-2019-lecture03-neuralnets.pdf

Classification intuition

- Training data: $\{x_i, y_i\}^N_{i=1}$
- Simple illustration case:
  - Fixed 2D word vectors to classify
  - Using softmax/logistic regression
  - Linear decision boundary



Visualizations with ConvNetJS by Karpathy!
http://cs.stanford.edu/people/karpathy/convnetjs/demo/classify2d.html

- **Traditional ML/Stats approach:** assume x_i are fixed, train (i.e., set) softmax/logistic regression weights $W \in \mathbb{R}^{C \times d}$ to determine a decision boundary (hyperplane) as in the picture
- **Method**: For each x, predict:

$$p(y|x) = \frac{\exp(W_y.x)}{\sum_{c=1}^{C} \exp(W_c.x)}$$

Extraído del curso de Stanford http://web.stanford.edu/class/cs224n/slides/cs224n-2019-lecture03-neuralnets.pdf

Training with softmax and cross-entropy loss

- For each training example (x,y), our objective is to maximize the probability of the correct class *y*

- Or we can minimize the negative log probability of that class:

$$-\log p(y|x) = -\log\left(\frac{\exp(f_y)}{\sum_{c=1}^{C}\exp(f_c)}\right)$$

Extraído del curso de Stanford http://web.stanford.edu/class/cs224n/slides/cs224n-2019-lecture03-neuralnets.pdf

Background: What is “cross entropy” loss/error?

- Concept of “cross entropy” is from information theory
- Let the true probability distribution be p
- Let our computed model probability be q
- The cross entropy is:

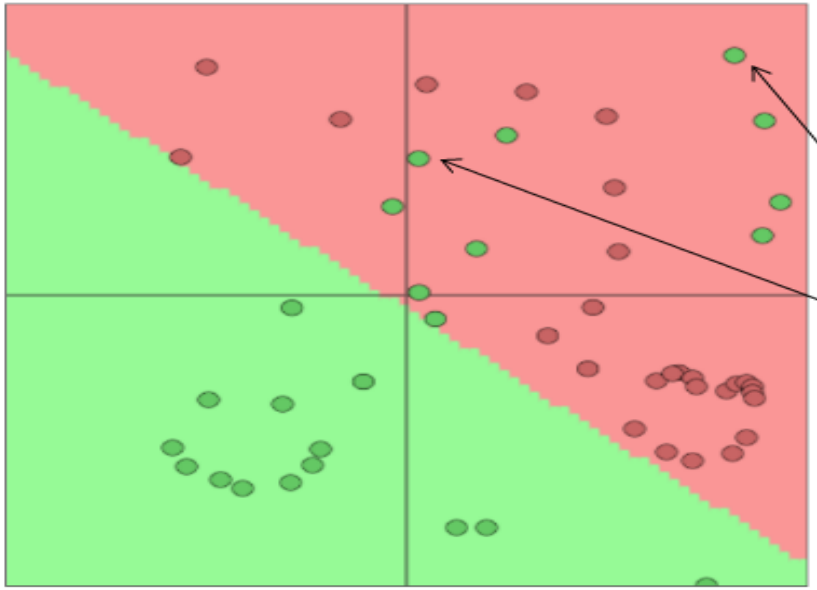
$$H(p,q) = -\sum_{c=1}^{C} p(c) \log q(c)$$

- Assuming a ground truth (or true or gold or target) probability distribution that is 1 at the right class and 0 everywhere else: p = [0,...,0,1,0,...0] then:
- **Because of one-hot p, the only term left is the negative log probability of the true class**

Extraído del curso de Stanford http://web.stanford.edu/class/cs224n/slides/cs224n-2019-lecture03-neuralnets.pdf

3. Neural Network Classifiers

- Softmax (≈ logistic regression) alone not very powerful
- Softmax gives only linear decision boundaries



This can be quite limiting

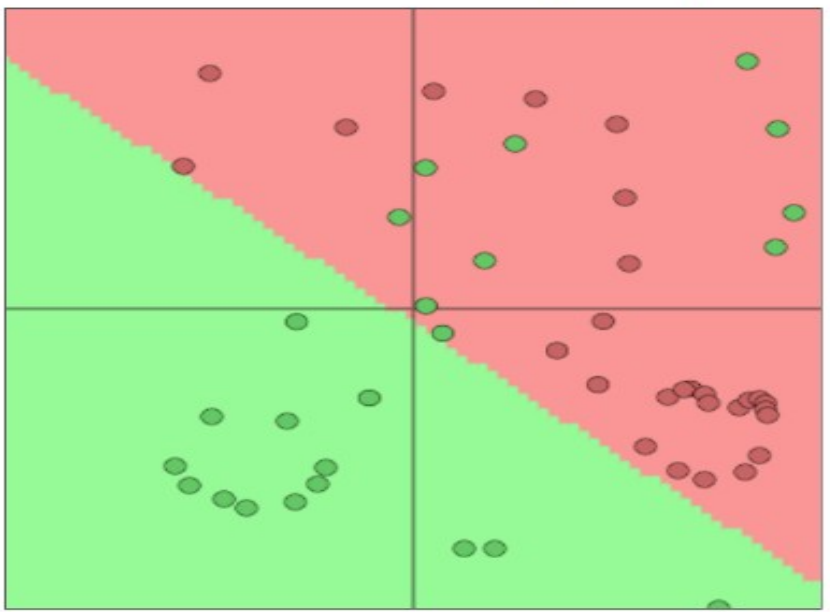
→ Unhelpful when a problem is complex

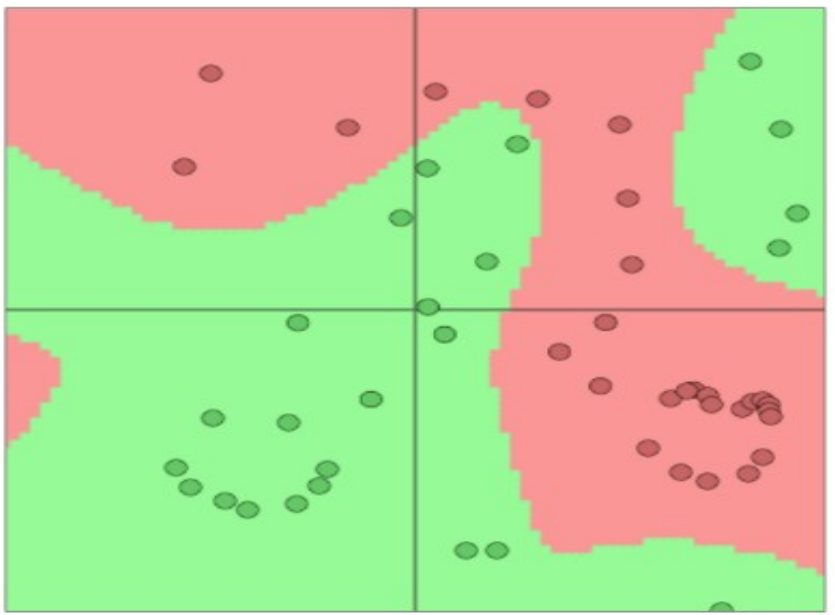
Wouldn't it be cool to get these correct?

Extraído del curso de Stanford http://web.stanford.edu/class/cs224n/slides/cs224n-2019-lecture03-neuralnets.pdf

Neural Nets for the Win!

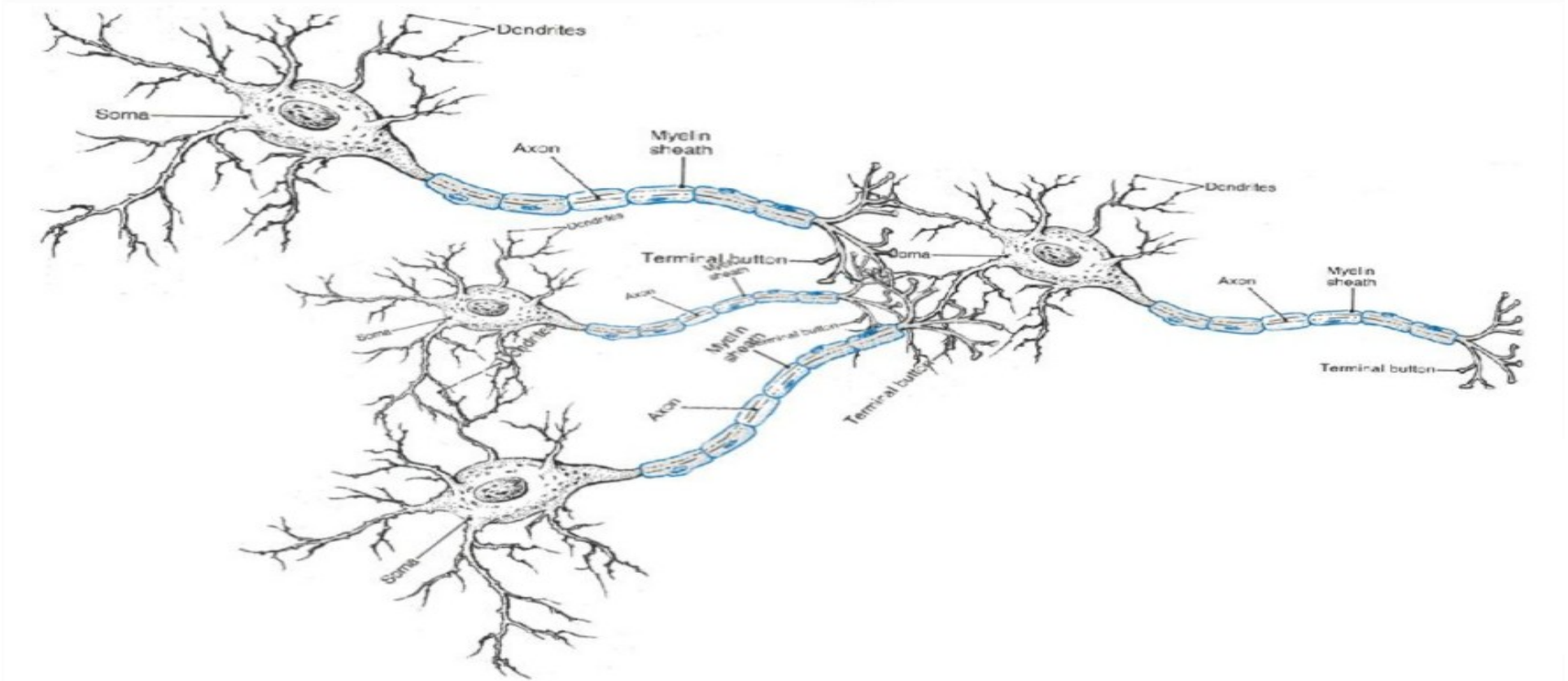
- Neural networks can learn much more complex functions and nonlinear decision boundaries!
  - In original space





Extraído del curso de Stanford http://web.stanford.edu/class/cs224n/slides/cs224n-2019-lecture03-neuralnets.pdf

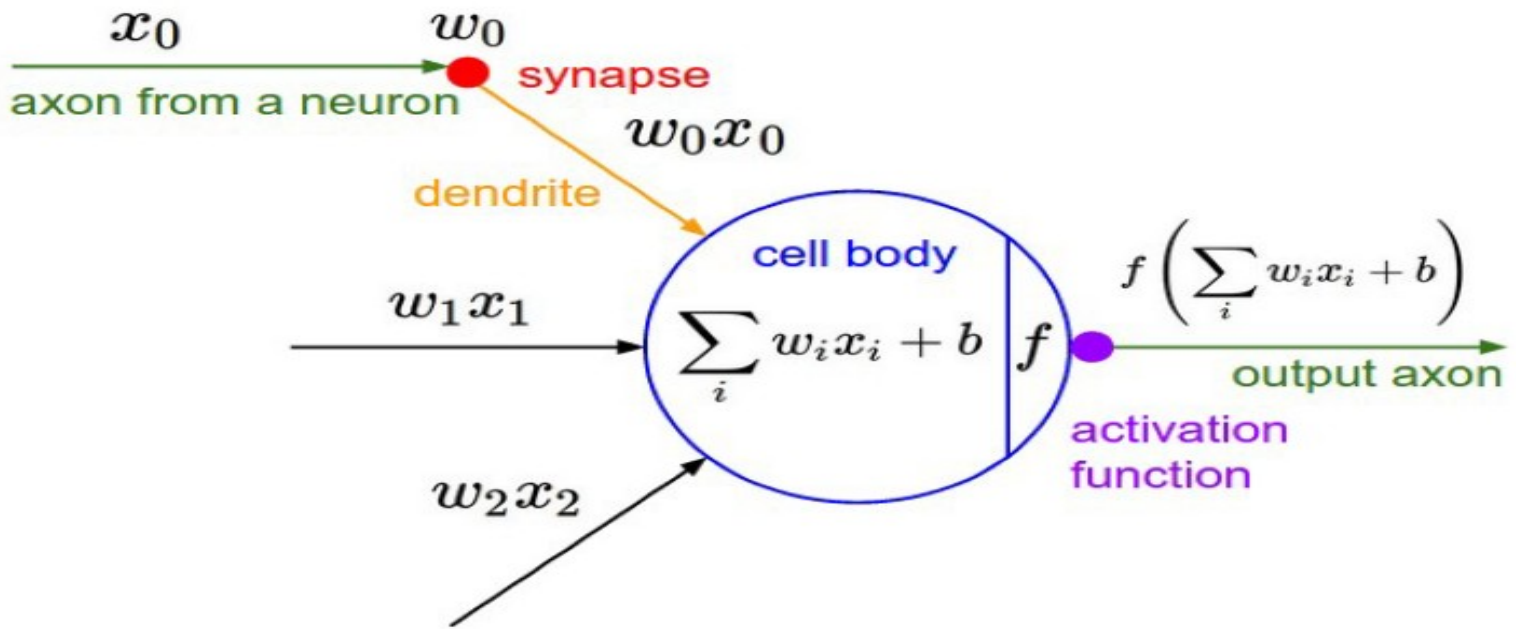
Neural computation



Extraído del curso de Stanford http://web.stanford.edu/class/cs224n/slides/cs224n-2019-lecture03-neuralnets.pdf

An artificial neuron

- Neural networks come with their own terminological baggage
- But if you understand how softmax models work, then you can easily understand the operation of a neuron!

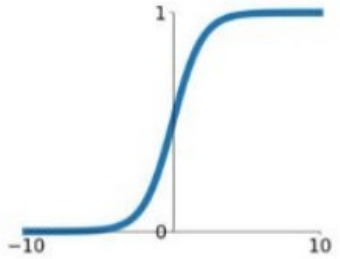


Extraído del curso de Stanford http://web.stanford.edu/class/cs224n/slides/cs224n-2019-lecture03-neuralnets.pdf

Funciones de Activación

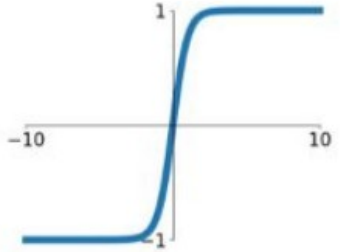
Sigmoid

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



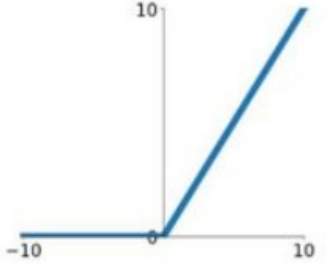
tanh

$$\tanh(x)$$



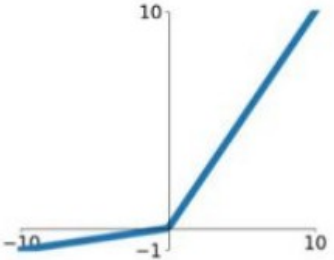
ReLU

$$\max(0, x)$$



Leaky ReLU

$$\max(0.1x, x)$$

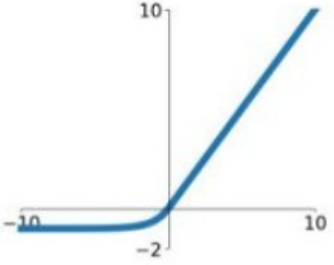


Maxout

$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

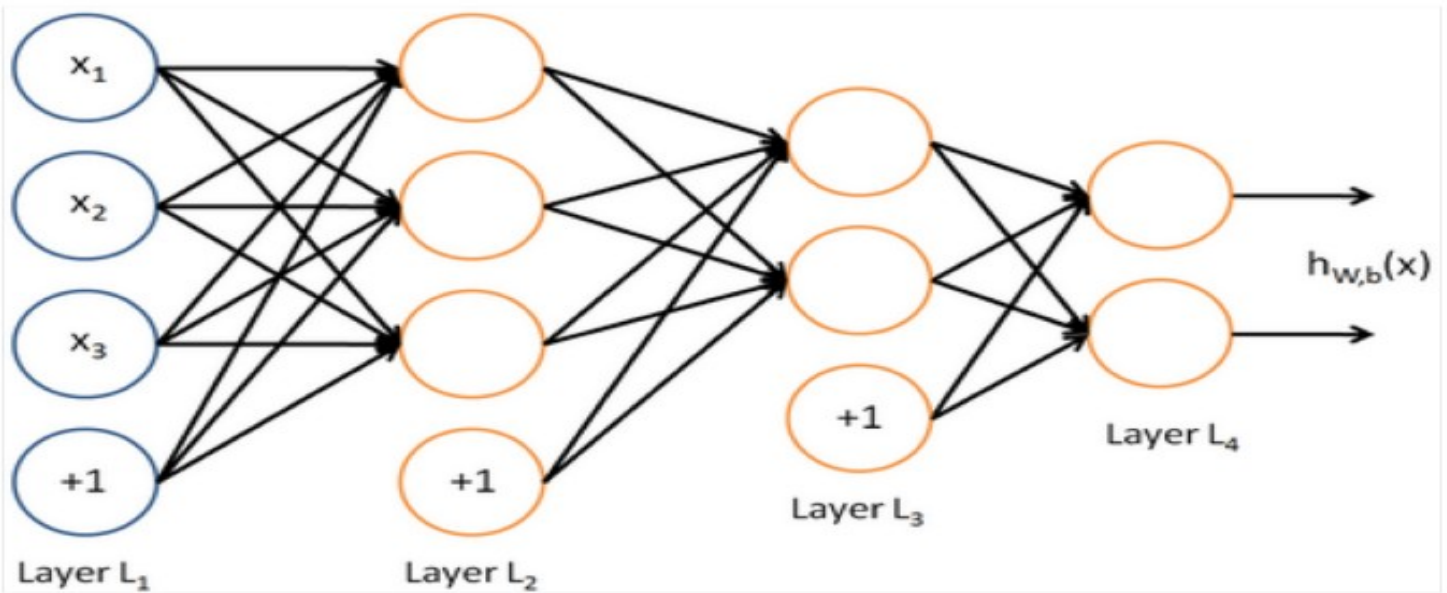
ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



A neural network
= running several logistic regressions at the same time

Before we know it, we have a multilayer neural network....



Extraído del curso de Stanford http://web.stanford.edu/class/cs224n/slides/cs224n-2019-lecture03-neuralnets.pdf

Matrix notation for a layer

We have

$$a_1 = f(W_{11}x_1 + W_{12}x_2 + W_{13}x_3 + b_1)$$

$$a_2 = f(W_{21}x_1 + W_{22}x_2 + W_{23}x_3 + b_2)$$

etc.

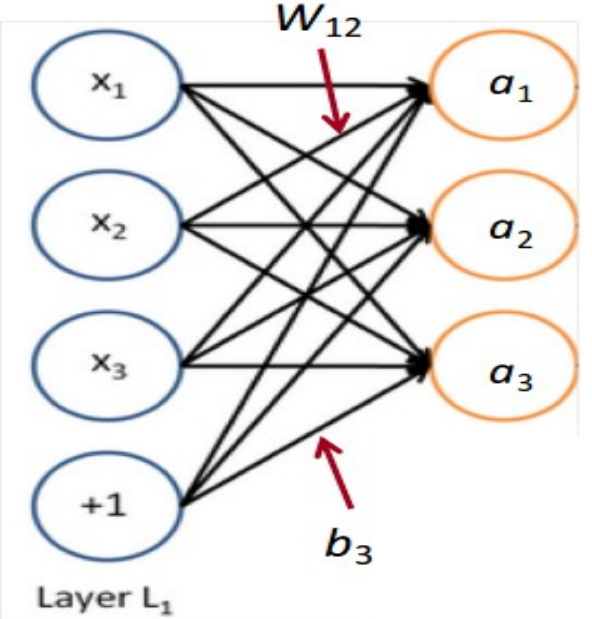
In matrix notation

$$z = Wx + b$$

$$a = f(z)$$

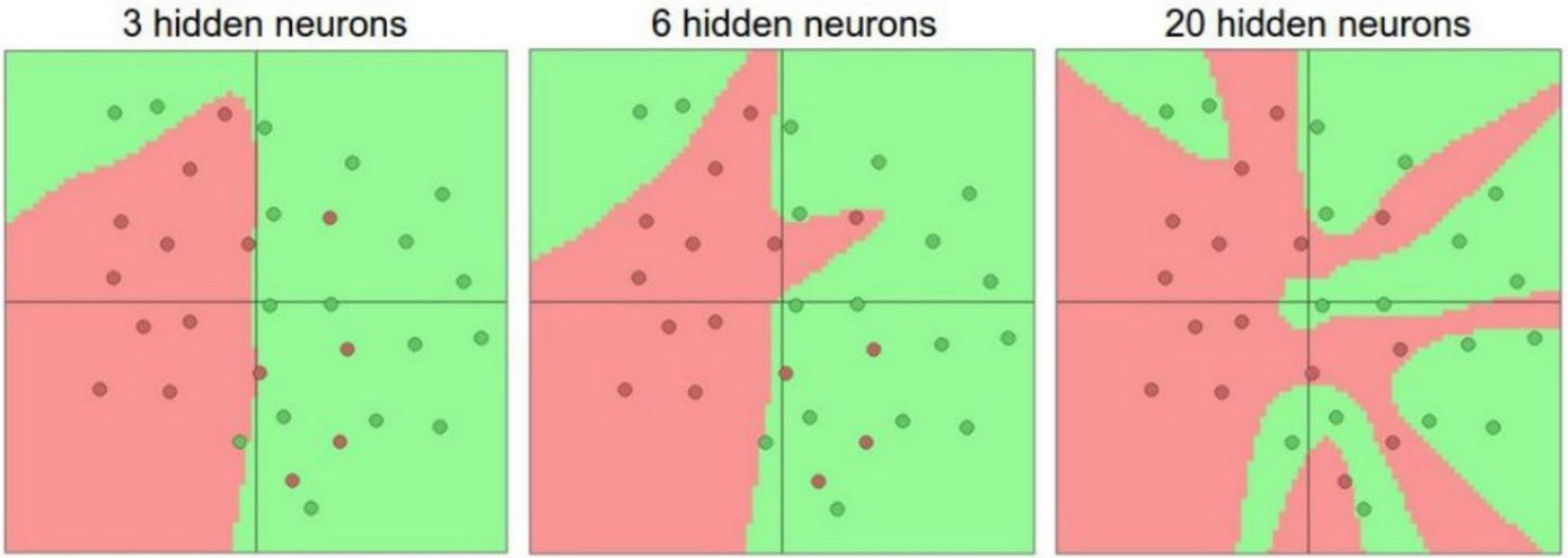
Activation f is applied element-wise:

$$f([z_1, z_2, z_3]) = [f(z_1), f(z_2), f(z_3)]$$



Extraído del curso de Stanford http://web.stanford.edu/class/cs224n/slides/cs224n-2019-lecture03-neuralnets.pdf

Capacidad de la capa oculta



Cuanto más neuronas, más capacidad de ajuste tiene el modelo

Capacidad: relacionada con cantidad de neuronas

1. Derivative wrt a weight matrix

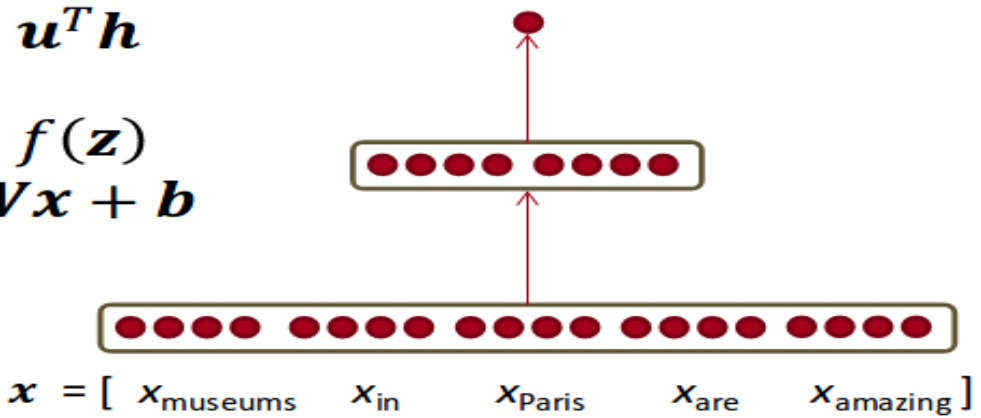
- Let's look carefully at computing $\frac{\partial s}{\partial \boldsymbol{W}}$
  - Using the chain rule again:

$$\frac{\partial s}{\partial \boldsymbol{W}} = \frac{\partial s}{\partial \boldsymbol{h}} \frac{\partial \boldsymbol{h}}{\partial \boldsymbol{z}} \frac{\partial \boldsymbol{z}}{\partial \boldsymbol{W}}$$

$$s = \boldsymbol{u}^T \boldsymbol{h}$$

$$\boldsymbol{h} = f(\boldsymbol{z})$$

$$\boldsymbol{z} = \boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}$$



Extraído del curso de Stanford http://web.stanford.edu/class/cs224n/slides/cs224n-2019-lecture04-backprop.pdf

Deriving gradients for backprop

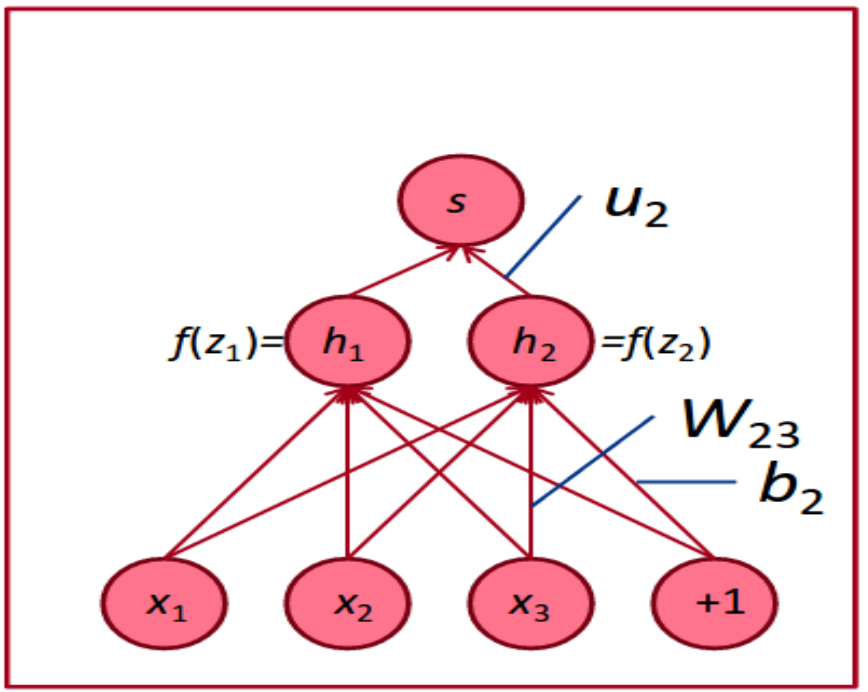
- For this function (following on from last time):

$$\frac{\partial s}{\partial \boldsymbol{W}} = \boldsymbol{\delta}\frac{\partial \boldsymbol{z}}{\partial \boldsymbol{W}} = \boldsymbol{\delta}\frac{\partial}{\partial \boldsymbol{W}}\boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}$$

- Let's consider the derivative of a single weight W_{ij}
- W_{ij} only contributes to z_i
  - For example: W_{23} is only used to compute z_2 not z_1

$$\frac{\partial z_i}{\partial W_{ij}} = \frac{\partial}{\partial W_{ij}}\boldsymbol{W}_{i.}\boldsymbol{x} + b_i$$

$$= \frac{\partial}{\partial w_{ij}}\sum_{k=1}^{d} W_{ik}x_k = x_j$$



Extraído del curso de Stanford http://web.stanford.edu/class/cs224n/slides/cs224n-2019-lecture04-backprop.pdf

Deriving gradients for backprop

- So for derivative of single W_{ij} :

$$\frac{\partial s}{\partial W_{ij}} = \delta_i x_j$$

Error signal from above — Local gradient signal

- We want gradient for full $\boldsymbol{W}$ – but each case is the same
- Overall answer: Outer product:

$$\frac{\partial s}{\partial \boldsymbol{W}} = \boldsymbol{\delta}^T \quad \boldsymbol{x}^T$$

$$[n \times m] \quad [n \times 1][1 \times m]$$

Extraído del curso de Stanford http://web.stanford.edu/class/cs224n/slides/cs224n-2019-lecture04-backprop.pdf

Entrenamiento

Para entrenar la red se usa una función de pérdida
    Ej. entropía cruzada

Se realiza descenso por gradiente (mini-batches) o una técnica basadas usando derivadas en pasos previos.
    Ej. SGD+momentum, Adagrad, Adam

Los pesos de las capas se calculan por backpropagation (regla de la cadena, derivada de composición de funciones).

Una pasada completa por el conjunto de entrenamiento (fraccionado en batches) se denomina una época (**1 epoch**)

Regularizaciones

Obtener mejores resultados evitando que la red se sobre-ajuste (overfitting) a los datos de entrenamiento.

Ruido – pequeñas perturbaciones a los dato de entrada
L1 – Norma L1 a los pesos (dispersión)
L2 – Norma L2 a los pesos (evitar picos)
Dropout – Anular aleatoriamente algunas neuronas
Early Stopping – Parar el entrenamiento cuando en el conjunto de validación no se observan mejoras
...

Redes Neuronales Recurrentes

La capa recibe además de la entrada, la propia salida en la ejecución anterior.

Útiles para modelar información secuencial, por ejemplo, texto.

Un tipo de red recurrente que da buenos resultados es la LSTM.

No vamos a profundizar en esta clase pero pueden hacerlo para el laboratorio!

Redes Neuronales Convolucionales

Originalmente inspiradas en la visión artificial en convoluciones 2D (ej. convolución para detectar bordes)

La variante de convolución 1D se utiliza para modelar entradas secuenciales, por ejemplo, texto.

Se aplica múltiples operaciones (con pesos que se entrenan) a lo largo de la secuencia desplazando una ventana (filtro)

No vamos a profundizar en esta clase pero pueden hacerlo para el laboratorio!

Capa de embeddings (look-up table)

Representación one-hot

[0..010..0] (la posición del 1 indica el elemento)

Capa de embeddings

Matriz X de dimensión |V|xD
- V es el vocabulario (suponiendo que se usa para word embeddings)
- D es la dimensión de la representación distribuida (hiperparámetro)

1_i * X selecciona la fila i (corresp. al i-ésimo elemento).
La capa siguiente recibe $X_{i,:}$ (o una secuencia si la entrada es secuencial)

Los valores de X (pesos) pueden ajustarse durante el entrenamiento o haber sido pre-entrenados, o ambos.

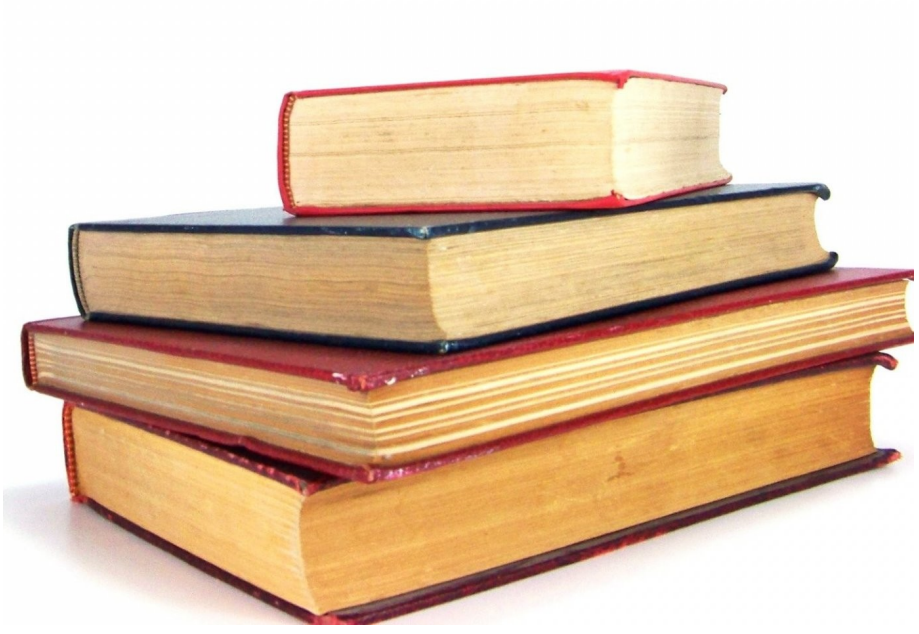
Word Embeddings
(Repaso para el lab)

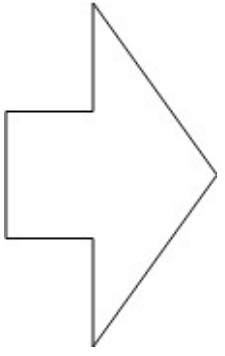
Word Embeddings

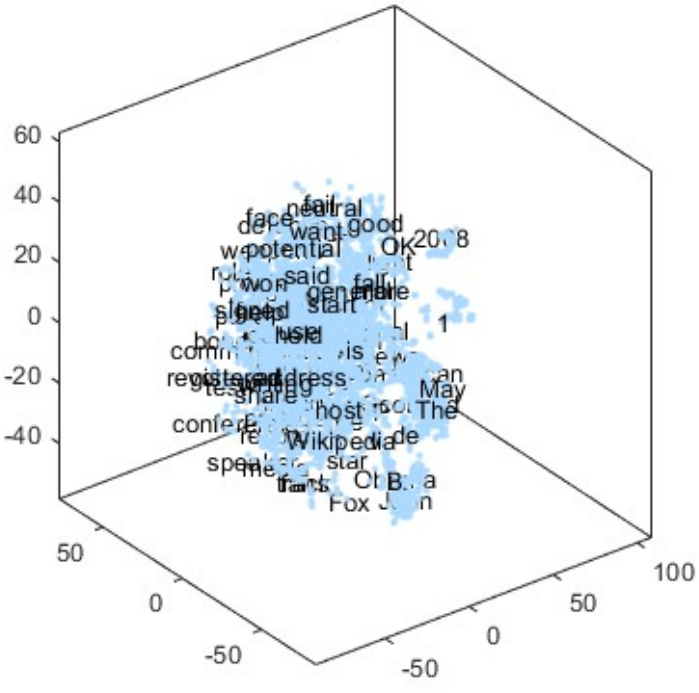
Hablamos de ellos hace algunas clases. ¿Qué son?

Word Embeddings

Representar a las **palabras** como **vectores** a partir de **texto**







Word Embeddings

- Representar el **significado** de las palabras
- Usando el **contexto** donde ocurren (texto)
- Con **vectores**

Las palabras que ocurren en contextos similares tienden a tener significados similares (Harris, 1954)

“You shall know a word by the company it keeps”(J. R. Firth 1957: 11)

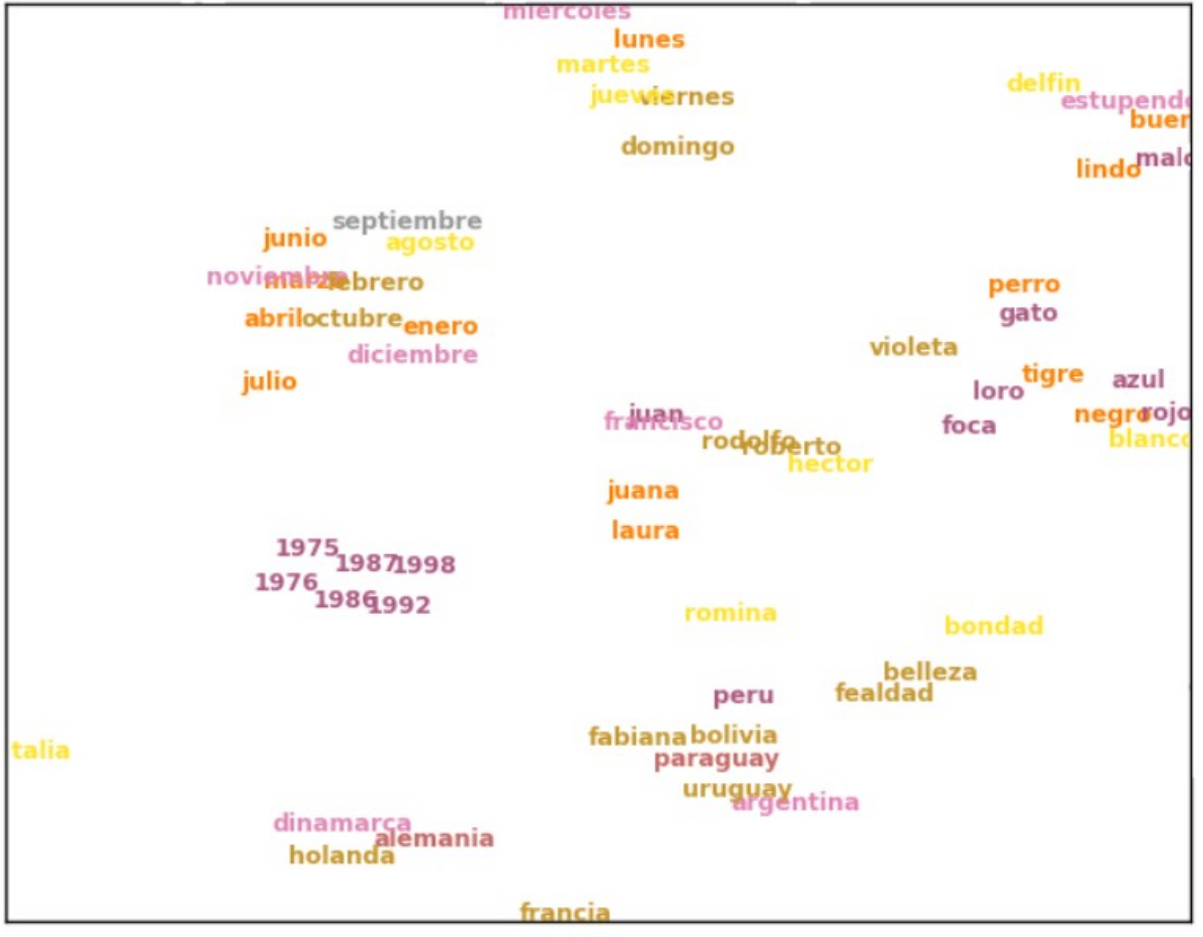
One of the most successful ideas of modern statistical NLP!
(Christopher Manning, CS224N/Lect1)

Aplicando la Hipótesis Distribucional

Significados Similares
(semántica)
= (hip. dist)
Contextos Similares
(texto)
= (func. contextos)
Representaciones Similares
(vectores)

Aplicando la Hipótesis Distribucional

Vectores entrenados con **GloVe** en la **Wikipedia** en español, dimensión 300. Visualización con reducción de la dimensión a 2 con **t-SNE**.



Ejemplo

...dig a [hole. The	**car**	drove away] leaving behind ...
...to directly [drive the	**car**	wheel angle] 3. Force ...
...celebrity status, [drove fast	**cars**	and partied] with some ...
...but there [are police	**cars**	that chase] you. Each ...
...world of [money, fast	**cars**	and excitement] and, under ...
...to pet [the family's	**cat**	and dog,] who tended ...
...and then [wanted a	**cat**	to eat] the many ...
...murmur is [detectable. The	**cat**	often eats] and drinks ...
...behaviour of [a domestic	**cat**	playing with] a caught ...
...have never [seen a	**cat**	eat so] little and ...
...bank, children [playing with	**dogs**	and a] man leading. ...
...sure you [encourage your	**dog**	to play] appropriate chase ...
...Truth, Lord: [yet the	**dogs**	eat of] the crumbs ...
...vegetable material [and enzymes.	**Dogs**	also eat] fruit, berries ...
...hubby once [ate the	**dog**	food and] asked for ...
...were back [at the	**van**	and drove] down to ...
...go down [as the	**van**	drove off.] As he ...
...heavy objects, [driving transit	**vans**	, wiring plugs] and talking ...
...of the [fast food	**van**	being located] outside their ...
...each of [the six	**van**	wheels , and] also under ...

(Ejemplo tomado de la charla de Alessandro Lenci en la Global WordNet Conference (GWC 2014).)

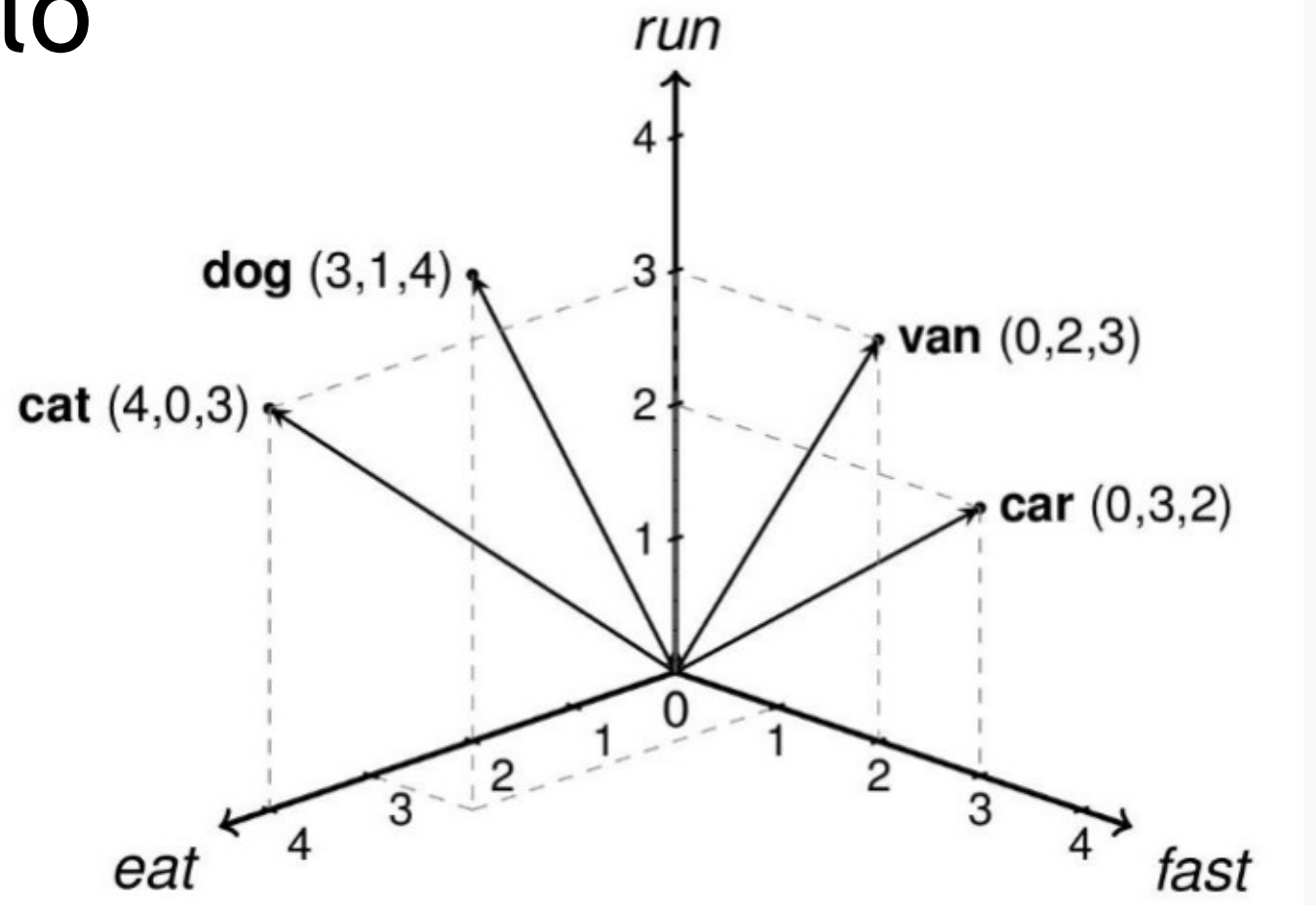
Ejemplo (co-occurrence matrix)

	dog	*drive*	*eat*	*fast*	*play*	...	*the*	*wheel*
car	0	3	0	2	0	⋮	2	1
cat	1	0	3	0	1	⋮	2	0
dog	0	0	3	0	2	⋮	2	0
van	0	3	0	1	0	⋮	3	1

co-occurrence matrix

(Ejemplo tomado de la charla de Alessandro Lenci en la Global WordNet Conference (GWC 2014).)

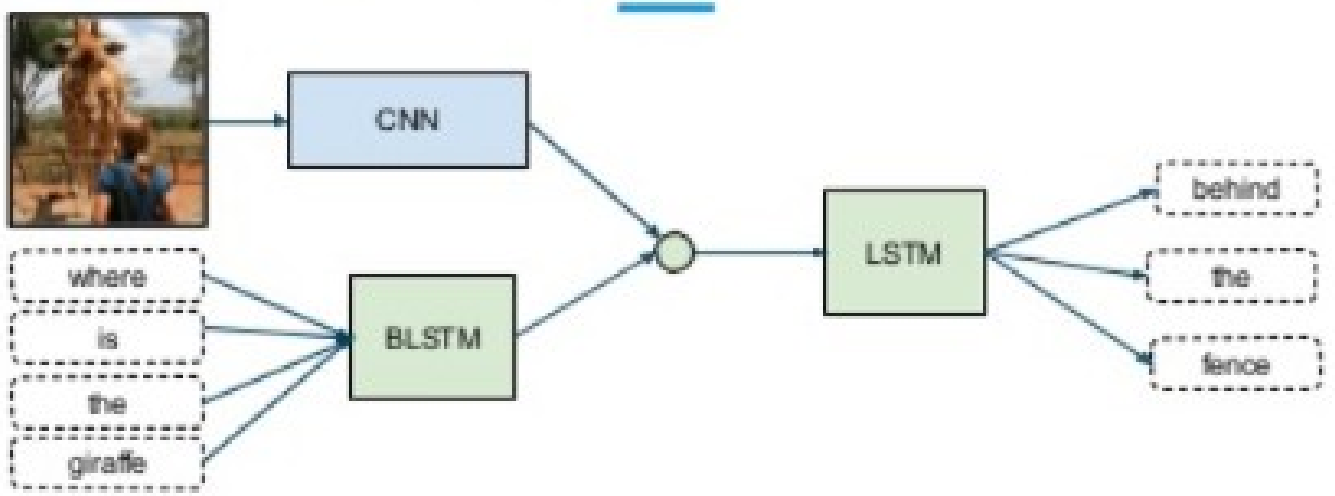
Ejemplo



(Ejemplo tomado de la charla de Alessandro Lenci en la Global WordNet Conference (GWC 2014).)

Multimodalidad

Podemos combinarlos con otros tipos de información. Ej. audio, imágenes, videos, ...



Imágen de https://towardsdatascience.com/multimodal-deep-learning-ce7d1d994f4

Word Embeddings en este punto del curso

Existen muchos métodos para construirlos
LSA, word2:vec: (skip-gram, cbow), fastText, ELMo BERT, ...

Hay una activa investigación usando, construyendo y estudiando word embeddings

Para el laboratorio vamos a utilizar un conjunto de vectores pre-entrenados con fastText.

Word Embeddings en este punto del curso

Asumimos un conjunto de vectores pre-entrenado para un vocabulario.

- ¿Que podemos hacer con ellos?
    - **operaciones** (Ej. promedio)
    - **entrada de redes neuronales** (Ej. Embedding Layer)
    - otros clasificadores, SVM,
    - **retrofitting** (Ej. con WordNet (Faruqui et al., 2015))
    - **specialising** (a un dominio particular)

Pythton, numpy, sklearn, keras, ...

Python

Simple. Usado en la comunidad científica. Tiene librerías como numpy, sklearn, keras, …

Layout 2D (2 o 4 espacios, o tabs, consistente en archivo)

Fuertemente tipado — `7 + 'uno'` **Error!**

Tipado dinámico

```
a = 1
a = 'uno'
```

Puede ser extendido a otros lenguajes (Ej. C++ bindings)

Language Basics

Does anyone want to guess what this function[1] (or any line of code) does?

```
def someGreatFunction(arr):
    if len(arr) <= 1:
        return arr
    pivot = arr[len(arr) // 2]
    left = [x for x in arr if x < pivot]
    middle = [x for x in arr if x == pivot]
    right = [x for x in arr if x > pivot]
    return someGreatFunction(left) + middle + someGreatFunction(right)

print(someGreatFunction([3,6,8,10,1,2,1]))
```

[1] Example code from Andrej Karpathy's tutorial: http://cs231n.github.io/python-numpy-tutorial/

Extraído del repaso de Python de curso de Stanford http://web.stanford.edu/class/cs224n/readings/python-review.pdf

Language Basics

Does anyone want to guess what this function[1] (or any line of code) does?

```
def QuickSort(arr):
    if len(arr) <= 1:
        return arr
    pivot = arr[len(arr) // 2]
    left = [x for x in arr if x < pivot]
    middle = [x for x in arr if x == pivot]
    right = [x for x in arr if x > pivot]
    return QuickSort(left) + middle + QuickSort(right)

print(someGreatFunction([3,6,8,10,1,2,1]))
```

[1] Example code from Andrej Karpathy's tutorial: http://cs231n.github.io/python-numpy-tutorial/

Extraído del repaso de Python de curso de Stanford http://web.stanford.edu/class/cs224n/readings/python-review.pdf

Collections: List

Lists are **mutable arrays** (think **`std::vector`**)

```
names = ['Zach', 'Jay']
names[0] == 'Zach'
names.append('Richard')
len(names) == 3
print names  >> ['Zach', 'Jay', 'Richard']
names.extend(['Abi', 'Kevin'])
print names  >> ['Zach', 'Jay', 'Richard', 'Abi', 'Kevin']
names = []     # Creates an empty list
names = list() # Also creates an empty list
stuff = [1, ['hi','bye'], -0.12, None]  # Can mix types
```

Extraído del repaso de Python de curso de Stanford http://web.stanford.edu/class/cs224n/readings/python-review.pdf

List Slicing

List elements can be accessed in convenient ways.

Basic format: **`some_list[start_index:end_index]`**

```
numbers = [0, 1, 2, 3, 4, 5, 6]
numbers[0:3] == numbers[:3] == [0, 1, 2]
numbers[5:] == numbers[5:7] == [5, 6]
numbers[:] == numbers = [0, 1, 2, 3, 4, 5, 6]

numbers[-1] == 6                # Negative index wraps around
numbers[-3:] == [4, 5, 6]
numbers[3:-2] == [3, 4]         # Can mix and match
```

Extraído del repaso de Python de curso de Stanford http://web.stanford.edu/class/cs224n/readings/python-review.pdf

Collections: Tuple

Tuples are **immutable arrays**

```
names = ('Zach', 'Jay')  # Note the parentheses
names[0] == 'Zach'
len(names) == 2
print names >> ('Zach', 'Jay')
names[0] = 'Richard'
>> TypeError: 'tuple' object does not support item assignment

empty = tuple()    # Empty tuple
single = (10,)     # Single-element tuple. Comma matters!
```

Extraído del repaso de Python de curso de Stanford http://web.stanford.edu/class/cs224n/readings/python-review.pdf

Collections: Dictionary

Dictionaries are **hash maps**

```
phonebook = dict()                    # Empty dictionary
phonebook = {'Zach': '12-37'}         # Dictionary with one item
phonebook['Jay'] = '34-23'            # Add another item
print('Zach' in phonebook)            >> True
print('Kevin' in phonebook)           >> False
print(phonebook['Jay'])               >> `34-23`# phonebook.get('Ray', '-')
del phonebook['Zach']                 # Delete an item
print(phonebook)                      >> {'Jay' : '34-23'}
for name, number in phonebook.iteritems():
  print name, number                  >> Jay 34-23
```

Extraído del repaso de Python de curso de Stanford http://web.stanford.edu/class/cs224n/readings/python-review.pdf

Numpy

Manejo de vectores y matrices (tensores).

Tiene implementado el “cómputo pesado” en C/C++

Las librerías como sklearn y keras lo usan.

Por ejemplo, para entrenar un clasificador:
- los datos son una matriz **X** de numpy (np.array)
- los resultados un vector **y** (matriz con shape (N,))

np.ndarray

```
x = np.array([1,2,3])
```

```
>> [1  2  3]
```

```
y = np.array([[3,4,5]])
```

```
>> [[3  4  5]]
```

```
z = np.array([[6,7],[8,9]])
```

```
>> [[6  7]
    [8  9]]
```

```
print x,y,z
```

```
print x.shape
```

```
>> (3,)
```

A list of scalars!

```
print y.shape
```

```
>> (1,3)
```

A (row) vector!

```
print z.shape
```

```
>> (2,2)
```

A matrix!

Extraído del repaso de Python de curso de Stanford http://web.stanford.edu/class/cs224n/readings/python-review.pdf

np.ndarray Operations

Reductions: **np.max, np.min, np.amax, np.sum, np.mean, …**

Always reduces along an axis! (Or will reduce along all axes if not specified.)

(You can think of this as “collapsing” this axis into the function’s output.)

```
x = np.array([[1,2],[3,4]])

print(np.max(x, axis = 1))                      >> [2  4]

print(np.max(x, axis = 1, keepdims = True))     >> [[2]
                                                    [4]]
```

Extraído del repaso de Python de curso de Stanford http://web.stanford.edu/class/cs224n/readings/python-review.pdf

np.ndarray Operations

Matrix Operations: `np.dot, np.linalg.norm, .T, +, -, *, ...`

Infix operators (i.e. `+, -, *, **, /`) are **element-wise.**

Matrix multiplication is done with `np.dot(x, W)` or `x.dot(W)`

Transpose with `x.T`

Note: `Shapes (N,) != (N, 1)`

```
print(np.array([1,2,3]).T)              >> [1  2  3]

np.sum(np.array([1,2,3]), axis = 1)     >> Error!
```

Note: Scipy and np.linalg have many, many other advanced functions that are very useful!

Extraído del repaso de Python de curso de Stanford http://web.stanford.edu/class/cs224n/readings/python-review.pdf

Indexing

```
x = np.random.random((3, 4))  # Random (3,4) matrix


x[:]                          # Selects everything in x

x[np.array([0, 2]), :]        # Selects the 0th and 2nd rows

x[1, 1:3]                     # Selects 1st row as 1-D vector

                              # and 1st through 2nd elements

x[x > 0.5]                    # Boolean indexing
```

Note: Selecting with an ndarray or range will preserve the dimensions of the selection.

Extraído del repaso de Python de curso de Stanford http://web.stanford.edu/class/cs224n/readings/python-review.pdf

Broadcasting

```
x = np.random.random((3, 4))  # Random (3, 4) matrix

y = np.random.random((3, 1))  # Random (3, 1) matrix

z = np.random.random((1, 4))      # Random (3,) vector

x + y      # Adds y to each column of x

x * z      # Multiplies z element-wise with each row of x

print((y + y.T).shape) # Can give unexpected results!
```

Note: If you're getting an error, print the shapes of the matrices and investigate from there.

Extraído del repaso de Python de curso de Stanford http://web.stanford.edu/class/cs224n/readings/python-review.pdf

Efficient Numpy Code

Avoid explicit for-loops over indices/axes at all costs.

For-loops will *dramatically* slow down your code *(~10-100x)*.

```
for i in range(x.shape[0]):
    for j in range(x.shape[1]):
        x[i,j] **= 2
```

```
x **= 2
```

```
for i in range(100, 1000):
    for j in range(x.shape[1]):
        x[i, j] += 5
```

```
x[np.arange(100,1000), :] += 5
```

Extraído del repaso de Python de curso de Stanford http://web.stanford.edu/class/cs224n/readings/python-review.pdf

Sklearn (scikit-learn)

Kit de heramientas para ciencia de datos.

Utiliza NumPy, SciPy, y matplotlib

Métodos de clasificación, regresión y clustering

Funcionalidades como: pipeline, random search, evaluación, ...

Web: https://scikit-learn.org/

Keras

Framework para definición, entrenamiento y ejecución de modelos de redes neuronales (deep learning).

Puede correr sobre TensorFlow, Theano o CNTK
(librerías para diferenciación automática y ejecución en GPU)

Se puede definir la estructura del modelo, la función de pérdida, la técnica de entrenamiento, …

Web: https://keras.io/

En resumen..

Usando el corpus y los word embeddings

- vectorizar los textos
- entrenar un clasificador (ej. red neuronal con capa lstm, ...)
- o más de uno
- ajustar hiperparámetros
- evaluar en test

Entregar

- código, test.out, notebooks, readme.txt, grupo.txt