Lingüística Computacional

Parte 05: word2vec

Marcelo Finger and Felipe Salvatore

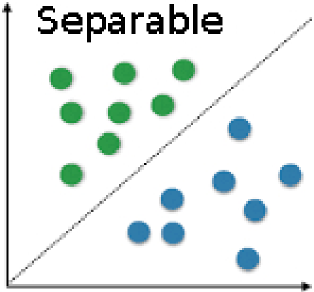
Departamento de Ciência da Computação
Instituto de Matemática e Estatística
Universidade de São Paulo

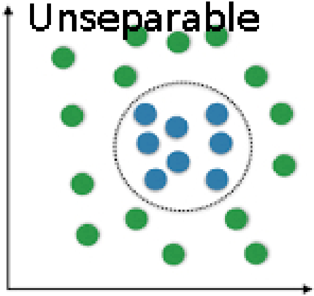
2º Semestre 2019

Introduction

Neural Networks

It all started with the **linear separation** problem





Perceptron

- Created by Rosenblatt [1957]
- Binary separation of data $X = \{x_1, \dots x_n\}$, $dim(x_i) = k$
- Find w and b such that, for $x \in X$

$$perceptron(x) = \begin{cases} 1, & w \cdot x + b > 0 \\ 0, & w \cdot x + b < 0 \end{cases}$$

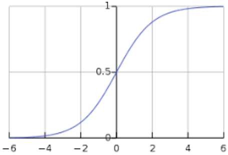
- Zero (unseparable) or infinitely many hyperplanes (w, b)
- Minsky & Papert [1969]: Xor-functions cannot be learned by single layer perceptrons

Dealing with non-separable cases

- SVM
  - Employs Quadratic Programming to obtain single answer
  - Expanding number of dimensions leads to separability, extra dimensions are a function of given ones
  - Uses "kernels" to deal with large number of dimensions

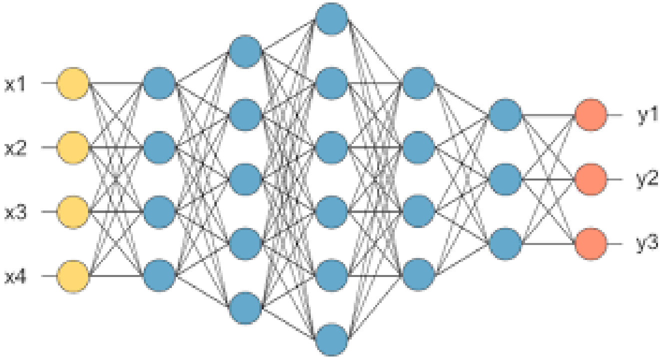
Dealing with non-separable cases

- SVM
  - Employs Quadratic Programming to obtain single answer
  - Expanding number of dimensions leads to separability, extra dimensions are a function of given ones
  - Uses "kernels" to deal with large number of dimensions
- Neural Networks
  - Multilayer perceptrons
  - 0-1 functions a problem for learning (not differentiable)
  - Use some sigmoid (σ) function instead



  - Neuron: $\sigma(w \cdot x + b)$

Neural Net (Old View)



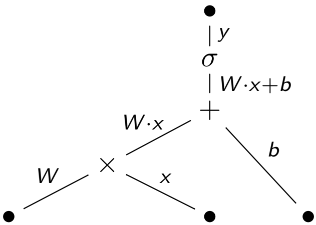
Does not scale. e.g. $|X| \geq 15,000$

Neural Net (New View)

- A whole layer may be represented as :

$$layer(x) = \sigma(Wx + b)$$

- $x, b, layer(x)$: vectors; W: matrix



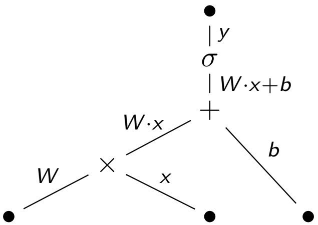
Nodes are operations
arcs are *tensors*

Neural Net (New View)

- A whole layer may be represented as :

$$layer(x) = \sigma(Wx + b)$$

- $x, b, layer(x)$: vectors; W: matrix



Nodes are operations
arcs are *tensors*

- Training occurs over such graph using **backpropagation**

The Backpropagation Algorithm

- Proposed by Kelley [1960], Bryson [1961] and Dreyfus [1962]
- Popularized by Rumelhart, Hinton & Williams [1986]
- Learning by **gradient descent**.
- Supervised learning
- Applies to a tensor graph (not only NN!)

The Backpropagation Algorithm (Guts)

- Initializes weights to be learned randomly
- Minimizes a **loss function**. E.g. $L(x) = \sum(y_i - \hat{y}_i(x))^2$
  - Phase 1: Propagation. Computes the output $\hat{y}$ and loss L
  - Phase 2: Weight update. α is the **learning rate**

$$W^{t+1} = W^t + \alpha \frac{\partial L}{\partial W}$$

$$b^{t+1} = b^t + \alpha \frac{\partial L}{\partial b}$$

- A cycle propagation-update is an **epoch**
- Local optimization, may get stuck at **local minima**

Word2vec: Basic Model

Word2vec

Word2vec is a name that covers two models

- Skip-gram
- **Continuous Bag-of-Words (CBOW)**

- Aim: learn efficiently a vectorial representation of words

$$banana \longrightarrow \langle v_1, \ldots, v_N \rangle, v_i \in \mathbb{Q}$$

- Unsupervised learning from a large unstructured corpus
- Based on word co-ocurrence statistics. The idea is not new:

"You shall know a word by the company it keeps" (J.R Firth, 1957)

A Simplified CBOW model

Given a corpus, choose:

- A vocabulary V.
- A vector size N to represent words

Use matrices $W \in \mathbb{Q}^{|V|,N}$ and $W' \in \mathbb{Q}^{N,|V|}$ to create **two** vetor representations of each word $\mathbb{w}$:

- **input vector**: $v_{\mathbb{w}}$ (line of W).
- **output vector**: $v'_{\mathbb{w}}$ (column of W').

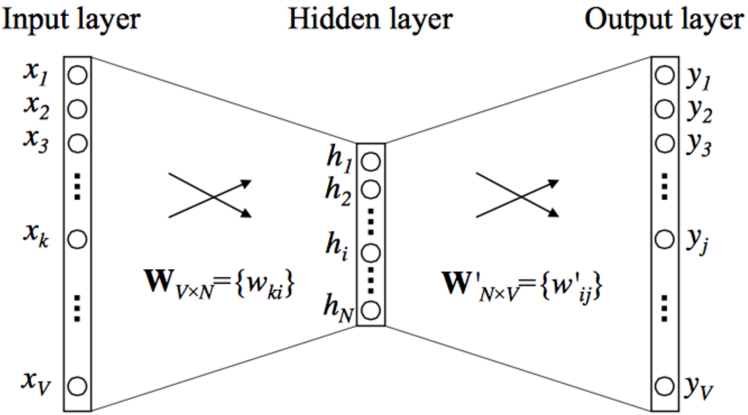
A Simplified CBOW model

The model's task is to predict a focus word given a context of words:

O primeiro rei de Portugal nasceu em ...

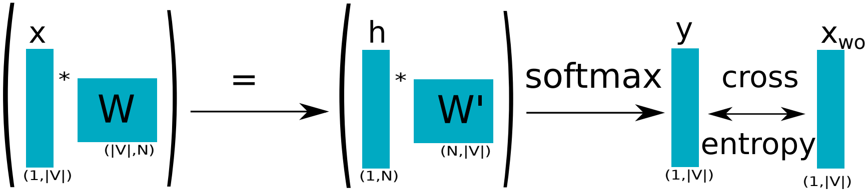
$$
\begin{aligned}
\text{Observation} &\Rightarrow (\text{rei, primeiro}) \\
&\Rightarrow (\text{input word, output word}) \\
&\Rightarrow (\mathbb{w}_I, \mathbb{w}_O)
\end{aligned}
$$

A Simplified CBOW model



Deep learning, with depth 2!!!

A Simplified CBOW model



Some Definitions

A **one-hot** is a vector of bits with a single 1-bit; all other bits 0. Given N elements we associate them to N 1-hot vectors of size N:

$$\langle 0 \cdots 010 \cdots 0 \rangle$$

The **softmax** function is a probability distribution over the elements of a vector:

$$P(z_j) = \frac{e^{z_j}}{\sum_{i=1}^{N} e^{z_i}}$$

The **cross-entropy** of distributions p and q

$$CE(p,q) = -\sum_i p_i \log q_i$$

A Simplified CBOW model

Given $(x_{\mathbb{w}_I}, x_{\mathbb{w}_O})$ one-hot of $(\mathbb{w}_I, \mathbb{w}_O)$ and $x = x_{\mathbb{w}_I}$, the model is:

$$h_i = \sum_{s=1}^{|V|} w_{si} x_s \quad \text{com } i = 1, \ldots, N \tag{1}$$

$$u_j = \sum_{s=1}^{N} w'_{sj} h_s \quad \text{com } j = 1, \ldots, |V| \tag{2}$$

$$y_j = P(\mathbb{w}_j | \mathbb{w}_I) = \frac{\exp(u_j)}{\sum_{j'=1}^{|V|} \exp(u_{j'})} \quad \text{com } j = 1, \ldots, |V| \tag{3}$$

$$E = CE(x_{\mathbb{w}_O}, y) = -\sum_{s=1}^{|V|} x_{\mathbb{w}_O s} \log(y_s) \tag{4}$$

A Simplified CBOW model

Due to 1-hot format of $x_{\mathbb{w}_I}$ e $x_{\mathbb{w}_O}$ we simplify (1), (2), (3) e (4)

$$h = v_{\mathbb{w}_I} \tag{5}$$

$$u_j = {v'_{\mathbb{w}_j}}^T v_{\mathbb{w}_I} \tag{6}$$

$$y_j = \frac{\exp(u_j)}{\sum_{j'=1}^{|V|} \exp(u_{j'})} \tag{7}$$

$$E = -u_{j^*} + \log(\sum_{j'=1}^{|V|} \exp(u_{j'})) \tag{8}$$

where j^* is the index of $\mathbb{w}_O$.

A Simplified CBOW model: update

Applying backpropagation we have the weight update of the last layer is

$$w'^{(new)}_{ij} = w'^{(old)}_{ij} - \alpha\, e_j\, h_i \tag{9}$$

in vector notation

$$v'^{(new)}_{\mathbb{W}_j} = v'^{(old)}_{\mathbb{W}_j} - \alpha\, e_j\, v_{\mathbb{W}_I} \tag{10}$$

where $e = y - x_{\mathbb{W}_O}$

A Simplified CBOW model: update

- $\mathbb{w}_j \neq \mathbb{w}_O \Rightarrow -\alpha\, e_j < 0 \Rightarrow$ subtract from $v'_{\mathbb{w}_j}$ a fraction of $v_{\mathbb{w}_I} \Rightarrow$ increase the cosine distance between $v_{\mathbb{w}_I}$ and $v'_{\mathbb{w}_j}$.
- $\mathbb{w}_j = \mathbb{w}_O \Rightarrow -\alpha\, e_j > 0 \Rightarrow$ add a fraction of $v_{\mathbb{w}_I}$ to $v'_{\mathbb{w}_j} \Rightarrow$ decrease the cosine distance between $v_{\mathbb{w}_I}$ and $v'_{\mathbb{w}_j}$.

A Simplified CBOW model: update

Proceed with backpropagation:

$$W^{(new)} = W^{(old)} - \alpha\, xEH^{T} \tag{11}$$

$$v_{\mathbb{w}_I}{}^{(new)} = v_{\mathbb{w}_I}{}^{(old)} - \alpha\, xEH^{T}_{(k_I,.)} \tag{12}$$

where $EH = e(W')^{T}$ and k_I is the index of $\mathbb{w}_I$.

A Simplified CBOW model

Repeat this process with examples from the corpus, the effect accumulates and as a result words with similar contexts will get close to each other.

The model captures the co-ocurrence statistics using cosine distance

Full CBOW Model

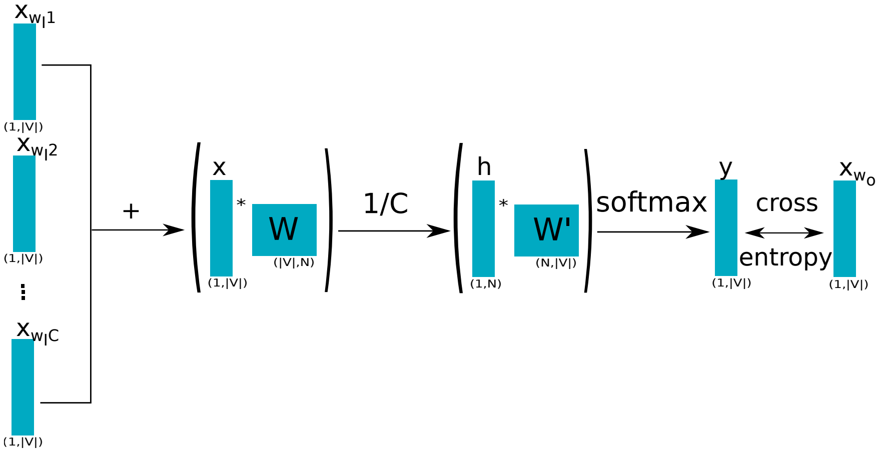
CBOW

Now, starting from an arbitrary window of size C, we construct observations such as $([\mathbb{w}_{I_1}, \dots, \mathbb{w}_{I_C}], \mathbb{w}_O)$.
E.g., for $C = 4$:

Nunca me acostumei com o cantor dessa banda, e nem ...

$$([com, o, dessa, banda], cantor)$$

CBOW: modelo (I)



CBOW: model

$$x = x_{\mathrm{w}_{I_1}} + \cdots + x_{\mathrm{w}_{I_C}} \tag{13}$$

$$h = \frac{1}{C}(v_{\mathrm{w}_{I_1}} + \cdots + v_{\mathrm{w}_{I_C}}) \tag{14}$$

$$u_j = \sum_{s=1}^{N} w'_{sj} h_s \tag{15}$$

$$y_j = p(\mathrm{w}_j | \mathrm{w}_{I_1}, \ldots, \mathrm{w}_{I_C}) = \frac{\exp(v'_{\mathrm{w}_j}.^T h)}{\sum_{j'=1}^{|V|} \exp(v'_{\mathrm{w}_{j'}}.^T h)} \tag{16}$$

$$E = -u_{j^*} + \log(\sum_{j'=1}^{|V|} \exp(u_{j'})) \tag{17}$$

CBOW: update

$$
{v'_{\mathbb{w}_j}}^{(new)} = {v'_{\mathbb{w}_j}}^{(old)} - \alpha \, e_j \, h \tag{18}
$$

$$
{v_{\mathbb{w}_{I_c}}}^{(new)} = {v_{\mathbb{w}_{I_c}}}^{(old)} - \frac{1}{C} \, \alpha \, x E H^{T}_{(k_{I_c},.)} \tag{19}
$$

for $c = 1, \ldots, C$. Where $k_{I_1}, \ldots, k_{I_C}$ are the indexes of $\mathbb{w}_{I_1}, \ldots, \mathbb{w}_{I_C}$ respectively.

Optimization

$$y_j = \frac{\exp(u_j)}{\sum_{j'=1}^{|V|} \exp(u_{j'})}$$

Too costly to compute for each input training instance

- Negative Sampling
- Hierarchical Softmax

Negative Sampling

We keep x, W, W', h as before. To compute the error function, we employ a distribution $P_n(\mathbb{w})$ over all words in the corpus. E.g.:

$$P_n(\mathbb{w}) = \frac{U(\mathbb{w})^{\frac{3}{4}}}{Z}$$

Using $P_n(\mathbb{w})$ we sample $\mathbb{w}_{i_1}, \ldots, \mathbb{w}_{i_K}$; avoid $\mathbb{w}_O$ amog them

Negative Sampling

$$(\mathbb{w}_I, \mathbb{w}_O)$$

Positive example

$$(\mathbb{w}_{i_1}, \mathbb{w}_O), \ldots, (\mathbb{w}_{i_K}, \mathbb{w}_O)$$

Negative examples

Negative Sampling: the model

$$p(D = 1 \mid \mathbb{w}, \mathbb{w}_O) = \sigma(v'_{\mathbb{w}}\ .^T\ h)$$

probability of $(\mathbb{w}, \mathbb{w}_O)$ to co-occur in the corpus (in a C-window)

$$p(D = 0 \mid \mathbb{w}, \mathbb{w}_O)$$

probability of $(\mathbb{w}, \mathbb{w}_O)$ not to co-occur in the corpus

The goal of training now is to maximize the probabilities

$p(D = 1 \mid \mathbb{w}_I, \mathbb{w}_O),\ p(D = 0 \mid \mathbb{w}_{i_1}, \mathbb{w}_O), \ldots,\ p(D = 0 \mid \mathbb{w}_{i_K}, \mathbb{w}_O)$

Negative Sampling: the model

Minimize the following error function:

$$
\begin{aligned}
E &= -\log(p(D=1 \mid \mathrm{w}_I, \mathrm{w}_O) \,.\, \prod_{s=1}^{K} p(D=0 \mid \mathrm{w}_{i_s}, \mathrm{w}_O)) \\
&= -(\log p(D=1 \mid \mathrm{w}_I, \mathrm{w}_O) + \log(\prod_{s=1}^{K} p(D=0 \mid \mathrm{w}_{i_s}, \mathrm{w}_O))) \\
&= -(\log p(D=1 \mid \mathrm{w}_I, \mathrm{w}_O) + \sum_{s=1}^{K} \log(p(D=0 \mid \mathrm{w}_{i_s}, \mathrm{w}_O))) \\
&= -\log \sigma(v'_{\mathrm{w}_O}\,.^T\, h) - \sum_{s=1}^{K} \log(\sigma(-v'_{\mathrm{w}_{i_s}}\,.^T\, h))
\end{aligned}
$$

Negative Sampling: update

$$v'_{\mathbb{w}_O}{}^{(new)} = v'_{\mathbb{w}_O}{}^{(old)} - \alpha\,(\sigma(v'_{\mathbb{w}_O}\,.^T\ h) - 1)\,h \tag{20}$$

$$v'_{\mathbb{w}_{i_s}}{}^{(new)} = v'_{\mathbb{w}_{i_s}}{}^{(old)} - \alpha\,\#(i_s)\,\sigma(v'_{\mathbb{w}_{i_s}}\,.^T\ h)\,h \tag{21}$$

$$W^{(new)} = W^{(old)} - \alpha\,EH^T \tag{22}$$

where

$$EH = (\sigma(v'_{\mathbb{w}_O}\,.^T\ h) - 1)\,v'_{\mathbb{w}_O} + \sum_{s=1}^{K} \sigma(v'_{\mathbb{w}_{i_s}}\,.^T\ h) v'_{\mathbb{w}_{i_s}}$$

Deep learning with 1.5 layers !!!

Evaluation

Intrinsic Evaluation of the Method

$\mathbb{w}_1$ is to $\mathbb{w}_2$ as $\mathbb{w}_3$ is to x

- $\mathbb{w}_1 =$ France, $\mathbb{w}_2 =$ Paris, $\mathbb{w}_3 =$ Japan; $x =$ **Tokyo**
- $\mathbb{w}_1 =$ man, $\mathbb{w}_2 =$ king, $\mathbb{w}_3 =$ woman; $x =$ **queen**

Intrinsic Evaluation of the Method

$\mathbb{w}_1$ is to $\mathbb{w}_2$ as $\mathbb{w}_3$ is to x

- $\mathbb{w}_1$ = France, $\mathbb{w}_2$ = Paris, $\mathbb{w}_3$ = Japan; x = **Tokyo**
- $\mathbb{w}_1$ = man, $\mathbb{w}_2$ = king, $\mathbb{w}_3$ = woman; x = **queen**

Problematic cases:

- $\mathbb{w}_1$ = man, $\mathbb{w}_2$ = manager, $\mathbb{w}_3$ = woman; x = **secretary**
- $\mathbb{w}_1$ = white, $\mathbb{w}_2$ = beauty, $\mathbb{w}_3$ = black; x = **ugly**

Intrinsic Evaluation of the Method

$\mathbb{w}_1$ is to $\mathbb{w}_2$ as $\mathbb{w}_3$ is to x

- $\mathbb{w}_1 =$ France, $\mathbb{w}_2 =$ Paris, $\mathbb{w}_3 =$ Japan; $x =$ **Tokyo**
- $\mathbb{w}_1 =$ man, $\mathbb{w}_2 =$ king, $\mathbb{w}_3 =$ woman; $x =$ **queen**

Problematic cases:

- $\mathbb{w}_1 =$ man, $\mathbb{w}_2 =$ manager, $\mathbb{w}_3 =$ woman; $x =$ **secretary**
- $\mathbb{w}_1 =$ white, $\mathbb{w}_2 =$ beauty, $\mathbb{w}_3 =$ black; $x =$ **ugly**

The method unveils sexism and racism buried in the data!
Book: Weapons of Math Destruction

Applications (extrinsic evaluation)

Many NLP applications employ word2vec

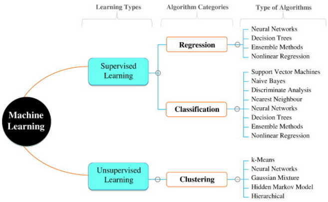
We have implemented word2vec in portuguese, using Google's TensorFlow

`https://github.com/felipessalvatore/Word2vec-pt`

And used it for Named Entity Recognition (NER)

Conclusion

Thank you!



Visit our group's Machine Learning tutorials
https://github.com/MLIME/Frameworks

- Pytorch
- Tensorflow
- Keras

Bibliography

- Halevy, A. Norvig, P. and Pereira, F. (2009). The unreasonable effectiveness of data in *Intelligent Systems*, IEEE.
- Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013). Efficient estimation of word representation in vector space. *arXiv preprint arXiv:1301.3781*.
- Mikolov, T., Sutskever, I., Chen, K., Corrado, G., and Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, pages 3111-3119.
- Morin, F., Bengio, Y. (2005). Hierarchical probabilistic neural network language model. In *AISTATS*, pages 246-252.
- Rong, X. (2016). Word2vec Parameter Learning Explained. *arXiv preprint arXiv:1411.2738*.