Neural Nets for text classification

Alexandre Allauzen

Fall 2023

MILES
Machine Intelligence and Learning Systems

Dauphine | PSL
UNIVERSITÉ PARIS

ESPCI PARIS | PSL

Roadmap

Call for Participation

Introduction

Word embeddings

Convolution for text

Outline

Who wants to do a presentation on:

- ELECTRA
- Mega
- State-Space model
- RWKV
- Pegasus
- LORA
- Adapter, ...

Schedules

- 10 minutes presentation
- a draft version of the slides to have feedback: 14/11
- presentation: 21/11

Outline

Call for Participation

Introduction

Word embeddings

Convolution for text

Text classification/rating

my wonderful friend took
me to see this movie
for our anniversary.
it was terrible.

→ ☹ : $0 \cdots 1$: ☺

How to represent the input text ?

- Bag of features (words, …)
- Really represent the sentence as a whole

Bag of words (BOW)

this movie is just great , with a great music , while a bit long

vocabulary	binary bag	count bag	tf.idf bag	...
awesome	0	0	0	...
great	1	2	1.9	...
long	1	1	2.5	...
the	0	0	0	...
this	1	1	0.1	...

A basic vectorial representation of text

$$\mathbf{x} = \begin{pmatrix} 0 \\ 2 \\ 1 \\ 0 \\ 1 \end{pmatrix} \in \mathbb{R}^D \qquad \left.\begin{array}{r} \textit{awesome} \\ \textit{great} \\ \textit{long} \\ \textit{the} \\ \textit{this} \end{array}\right\} D$$

A simple problem

Assumptions

- Let define a finite set of known words: the vocabulary $\mathcal{V}$
- A text is a vector $\mathbf{x}$ of dimension $D = |\mathcal{V}|$
- Each component encodes the presence of a word

Then machine learning

- Naive Bayes
- SVM, Random Forrest, ...
- Logistic Regression

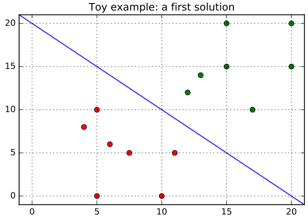
Logistic regression

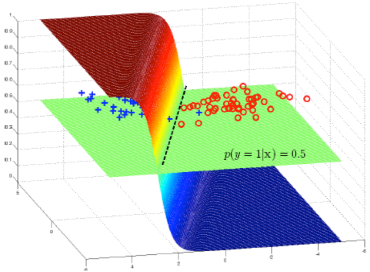
The class c is the outcome of the binary random variable C

The sigmoid/logistic function

$$a = w_0 + \mathbf{w}^t\mathbf{x} \in \mathbb{R}$$

$$\sigma(a) = \frac{e^a}{1+e^a} = \frac{1}{1+e^{-a}} \text{ and } y = P(C=1|\mathbf{x}) = \sigma(w_0 + \mathbf{w}^t\mathbf{x})$$





Training a Logistic regression model

- The parameters are $\boldsymbol{\theta} = (w_0, \mathbf{w})$,
- The i.i.d dataset: $\mathcal{D} = (\mathbf{x}_{(i)}, c_{(i)})_{i=1}^{n}$

Loss function minimization

$$\begin{aligned}
\mathcal{L}(\boldsymbol{\theta}; \mathcal{D}) &= -\sum_{i=1}^{n} log(P(C = c_{(i)} | \mathbf{x}; \boldsymbol{\theta})) \\
&= -\sum_{i=1}^{n} \left(c_{(i)} log(y_{(i)}) + (1 - c_{(i)}) log(1 - y_{(i)}) \right) \\
y_{(i)} &= \sigma(w_0 + \mathbf{w}^t \mathbf{x}_{(i)})
\end{aligned}$$

Optimization method

Stochastic Gradient Descent, or improved version (ADAM, L-BFGS, ...)

Outline

Call for Participation

Introduction

Word embeddings

Convolution for text

Back to logistic regression

$$\mathbf{x} = \begin{pmatrix} 0 \\ 2 \\ 1 \\ 0 \\ 1 \end{pmatrix} \in \mathbb{R}^D \qquad \left.\begin{array}{r} \textit{awesome} \\ \textit{great} \\ \textit{long} \\ \textit{the} \\ \textit{this} \end{array}\right\} D$$

For one input text:

$$w_0 + \mathbf{w}^t\mathbf{x} = w_0 + 2 \times w_2 + w_3 + w_5$$

The class is positive ($y = 1$) if

$$w_0 + 2 \times w_2 + w_3 + w_5 > 0$$

$$2 \times w_{great} + w_{long} + w_{this} + > -w_0$$

A limited representation of words

With the logistic regression model on a bag of words:

$w_{\text{bad}} < 0$ $\qquad$ $w_{\text{good}} > 0$

☹ $\longrightarrow$ ☺

0

Consider the two following examples:

the end is **really** **bad**	☹ $\Rightarrow w_{\text{bad}} \searrow$
the **bad** guy is *awesome*	☺ $\Rightarrow w_{\text{bad}} \searrow, w_{\text{awesome}} \nearrow$

Multiple dimensions could help to:

- represent different usage
- consider the context
- leverage more from sparse, sometime ambigous observations.

A simple model for document classification - part 1

Idea

- The word representation could be shared among classes
- While their interpretation depends on the class

Input representation and composition

$$\mathbf{R}\times\mathbf{x} = \begin{pmatrix} \vdots & \vdots & \vdots & \vdots & \vdots \\ \mathbf{v}_1 & \mathbf{v}_2 & \mathbf{v}_3 & \mathbf{v}_4 & \mathbf{v}_5 \\ \vdots & \vdots & \vdots & \vdots & \vdots \end{pmatrix} \times \begin{pmatrix} 0 \\ \mathbf{2} \\ \mathbf{1} \\ 0 \\ \mathbf{1} \end{pmatrix} = 2\times\mathbf{v}_2+\mathbf{v}_3+\mathbf{v}_5 = \mathbf{d}$$

A simple model for document classification - part 2

Classification

$$P(y|\mathbf{x}) = \text{softmax}(\mathbf{W^o d}) = \text{softmax}(\mathbf{W^o} \times \mathbf{Rx}), \text{ or}$$
$$= \text{softmax}(\mathbf{W^o} \times f(\mathbf{Rx})),$$

with f a non-linear activation function.

Parameters

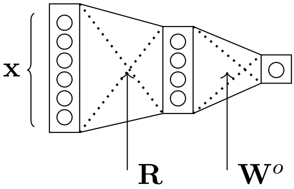
$$\boldsymbol{\theta} = (\mathbf{R}, \mathbf{W^o}) \rightarrow \textbf{to learn !!}$$

Reminder

If $\mathbf{y} = \text{softmax}(\mathbf{a})$, $\mathbf{y}$ is a vector and $\mathbf{a}$ is called the logit vector

$$y_i = \frac{e^{a_i}}{\sum_j e^{a_j}}$$

A first neural network



$$P(y = 1|\mathbf{x};\boldsymbol{\theta}) = \sigma(\mathbf{W}^o \times \mathbf{d})$$

- $\mathbf{x}$: $(|\mathcal{V}|, 1)$
- $\mathbf{R}$: $(K, |\mathcal{V}|)$
- $\mathbf{d}$: $(K, 1)$ $\qquad \mathbf{d} = \mathbf{R} \times \mathbf{x}$
- $\mathbf{W^o}$: $(1, K)$
- y : $(1, 1)$ $\qquad y = \sigma(\mathbf{W^o} \times \mathbf{d})$

Word embeddings

Definitions:

- To each word, a continous vector is associated: its embedding.
- The matrix $\mathbf{R}$ is called the look-up table and store the word embeddings.

Note:

- The term *look-up* comes from the real operation $\mathbf{R} \times \mathbf{x}$ is only theoritical !
- No computational cost, only storage and trainability challenge (enough observations for each word, Zipf, ...)
- **Pre-training** and **fine-tuning**

Unsupervised Pre-training of Word Embeddings

The question

- How to efficiently learn word representation
- based on the observation of raw texts ?

Distributional representations

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

and

> *Words are similar if they appear in similar contexts (Harris 1954).*

In practice

Word2Vec [6]

Context Bag of Words (CBOW)

The game

southern trees **[???]** strange fruits

Guess the word in the middle !

Context Bag of Words (CBOW)

The game

southern trees **[???]** strange fruits

Guess the word in the middle !

$\mathbf{x} =$ (southern trees strange fruits)

$\downarrow \quad \downarrow \quad \downarrow \quad \downarrow$

look-up

$\mathbf{v}_{-2} \quad \mathbf{v}_{-1} \quad \mathbf{v}_{+1} \quad \mathbf{v}_{+2}$

$\downarrow \quad \downarrow \quad \downarrow \quad \downarrow$

composition $\mathbf{v}_{\mathbf{southern}} + \mathbf{v}_{\mathbf{trees}} + \mathbf{v}_{\mathbf{strange}} + \mathbf{v}_{\mathbf{fruits}} \rightarrow \boldsymbol{h}$

Prediction softmax$(\boldsymbol{W}_o \times \boldsymbol{h}) \rightarrow$ bear ?

CBOW: details

Fast pre-training of word embeddings

- Introduced in [6] as a simplification of [1] (neural language model)
- Trained with negative sampling (Closed to Noise Contrastive Estimation [2])
- An efficient and tractable approximation of the count based method [5]

Other flavor

- Skip-gram [6]
- Glove [7]
- Fastext [3]

CBOW: Maximum Likelihood Estimate

In $P(w|\mathbf{x};\boldsymbol{\theta})$:

- predict the word w in the middle,
- given $\mathbf{x}$ the context.

MLE

$$\mathcal{L}(\boldsymbol{\theta};\mathcal{D}) = -\sum_{i=1}^{n} log(P(C = w|\mathbf{x};\boldsymbol{\theta})),$$

- The probability distribution over $\mathcal{V}$ is given by a softmax
- The set of possible outcomes is $\mathcal{V}$.

Cost of the softmax

$$\mathcal{L}(\boldsymbol{\theta};\mathcal{D}) = -\sum_{(\mathbf{x},\hat{w})\in\mathcal{D}} \log P_{\boldsymbol{\theta}}(\hat{w}|\mathbf{x})$$

$$P_{\boldsymbol{\theta}}(\hat{w}|\mathbf{x}) = \frac{e^{s_{\boldsymbol{\theta}}(\hat{w}|\mathbf{x})}}{\sum_{w'\in\mathcal{V}} e^{s_{\boldsymbol{\theta}}(w'|\mathbf{x})}}$$

$$\log P_{\boldsymbol{\theta}}(\hat{w}|\mathbf{x}) = s_{\boldsymbol{\theta}}(\hat{w}|\mathbf{x}) - \log\Big(\sum_{w'\in\mathcal{V}} e^{s_{\boldsymbol{\theta}}(w'|\mathbf{x})}\Big)$$

$$\frac{\partial \log P_{\boldsymbol{\theta}}(\hat{w}|\mathbf{x})}{\partial\boldsymbol{\theta}} = \frac{\partial s_{\boldsymbol{\theta}}(\hat{w}|\mathbf{x})}{\partial\boldsymbol{\theta}} - \underbrace{\sum_{w'\in\mathcal{V}} P_{\boldsymbol{\theta}}\ (w'|\mathbf{x})\frac{\partial s_{\boldsymbol{\theta}}(w',\mathbf{x})}{\partial\boldsymbol{\theta}}}_{\textit{costly!}}$$

Negative sampling

Recast the problem as a binary classification task:

- Positive examples: $(\mathbf{x}, w) \in \mathcal{D}$
- Negative examples: $(\mathbf{x}, \tilde{w})$, with $\tilde{w} \sim \mathcal{V}$

Use a binary classifier !

In practice:

- for one positive example $\sim \mathcal{D}$
- sample K negative and random samples from $\mathcal{V}$
- K is small (compared to the size of $\mathcal{V}$)
- the noise distribution does matter

Outline

Local contexts

the	end	is	very	bad	but	what	a	great	music

Local contexts

the	end	is	very	bad	but	what	a	great	music
			$\underbrace{\text{very} \quad \text{bad}}_{very \rightarrow bad++}$						

Local contexts

$$\text{the} \quad \text{end} \quad \text{is} \quad \underbrace{\underbrace{\text{very} \quad \text{bad}}_{very \rightarrow bad{+}{+}} \quad \text{but}}_{but \text{ will change } bad} \quad \text{what} \quad \text{a} \quad \text{great} \quad \text{music}$$

Local contexts

the	end	is	very	bad	but	what	a	great	music
			very→*bad*++						
			but will change *bad*						
		bad is for *end* not *music*						*great* is for *music* not fo *end*	

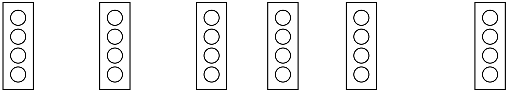
Motivations

- Local contextualisation
- Global view of the sentence

Another view of a sentence

$\mathbf{x} =$ [this, movie, was, a, great, experience]

Look-up



$\mathbf{v}_{this}$ $\mathbf{v}_{movie}$ $\mathbf{v}_{was}$ $\mathbf{v}_{a}$ $\mathbf{v}_{great}$ $\mathbf{v}_{experience}$

$$\downarrow$$

$$\begin{bmatrix} 1.7 & -0.3 & -0.5 & -2.7 & -0.0 & -0.3 \\ -0.5 & 0.3 & 0.4 & -1.1 & -0.9 & -0.5 \\ 0.7 & 0.6 & -1.3 & -1.1 & 0.7 & 1.6 \\ -0.0 & -0.7 & 1.1 & -0.3 & 0.7 & 1.5 \end{bmatrix}$$

Convolutional Neural Networks for Sentence Classification

- A short paper of 2014 [4]
- A simple and SOTA paper on text classification

Convolution in 1D

Extract a frame, or a window, and apply a “filter”

The filter

Kernel size of $ks = 2$

$w_{1,1}$	$w_{1,2}$
$w_{2,1}$	$w_{2,2}$
$w_{3,1}$	$w_{3,2}$
$w_{4,1}$	$w_{4,2}$

The input sequence

$L = 6$ vectors in $\mathbb{R}^D$, $D = 4$

$x_{1,1}$	$x_{1,2}$	$x_{1,3}$	$x_{1,4}$	$x_{1,5}$	$x_{1,6}$
$x_{2,1}$	$x_{2,2}$	$x_{2,3}$	$x_{2,4}$	$x_{2,5}$	$x_{2,6}$
$x_{3,1}$	$x_{3,2}$	$x_{3,3}$	$x_{3,4}$	$x_{3,5}$	$x_{3,6}$
$x_{4,1}$	$x_{4,2}$	$x_{4,3}$	$x_{4,4}$	$x_{4,5}$	$x_{4,6}$

The output value (output channel)

$$\text{At time } t = 1,\ h_1 = \sum_{i,j} w_{i,j} \times x_{i,j}$$

Convolution in 1D : time $t = 2$

Stride $= 1$

The filter

Kernel size of $ks = 2$

$w_{1,1}$	$w_{1,2}$
$w_{2,1}$	$w_{2,2}$
$w_{3,1}$	$w_{3,2}$
$w_{4,1}$	$w_{4,2}$

The input sequence

$L = 6$ vectors in $\mathbb{R}^D$, $D = 4$

$x_{1,1}$	$x_{1,2}$	$x_{1,3}$	$x_{1,4}$	$x_{1,5}$	$x_{1,6}$
$x_{2,1}$	$x_{2,2}$	$x_{2,3}$	$x_{2,4}$	$x_{2,5}$	$x_{2,6}$
$x_{3,1}$	$x_{3,2}$	$x_{3,3}$	$x_{3,4}$	$x_{3,5}$	$x_{3,6}$
$x_{4,1}$	$x_{4,2}$	$x_{4,3}$	$x_{4,4}$	$x_{4,5}$	$x_{4,6}$

The output value (output channel)

$$\text{At time } t = 2,\ h_2 = \sum_{i,j} w_{i,j} \times x_{i+1,j}$$

Convolution in 1D : time $t = 5$

Stride $= 1$

The filter

Kernel size of $ks = 2$

$w_{1,1}$	$w_{1,2}$
$w_{2,1}$	$w_{2,2}$
$w_{3,1}$	$w_{3,2}$
$w_{4,1}$	$w_{4,2}$

The input sequence

$L = 6$ vectors in $\mathbb{R}^D$, $D = 4$

$x_{1,1}$	$x_{1,2}$	$x_{1,3}$	$x_{1,4}$	$x_{1,5}$	$x_{1,6}$
$x_{2,1}$	$x_{2,2}$	$x_{2,3}$	$x_{2,4}$	$x_{2,5}$	$x_{2,6}$
$x_{3,1}$	$x_{3,2}$	$x_{3,3}$	$x_{3,4}$	$x_{3,5}$	$x_{3,6}$
$x_{4,1}$	$x_{4,2}$	$x_{4,3}$	$x_{4,4}$	$x_{4,5}$	$x_{4,6}$

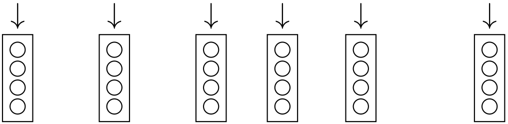
The output value (output channel)

$$\text{At time } t = 5,\ h_5 = \sum_{i,j} w_{i,j} \times x_{i+5,j}$$

Feature extraction for subsequences

Embeddings

$$\mathbf{x} = [\ \text{this}, \ \text{movie}, \ \text{was}, \ \text{a}, \ \text{great}, \ \text{experience}\ ]$$



$$\mathbf{v}_{this} \quad \mathbf{v}_{movie} \quad \mathbf{v}_{was} \quad \mathbf{v}_{a} \quad \mathbf{v}_{great} \quad \mathbf{v}_{experience}$$

After the convolution

$$\mathbf{h} = \Big(\underbrace{h_1}_{\text{(this,movie)}}, \underbrace{h_2}_{\text{(movie,was)}}, \underbrace{h_3}_{\text{(was,a)}}, \underbrace{h_4}_{\text{(a,great)}}, \underbrace{h_5}_{\text{(great,experience)}} \Big)$$

Convolution 1D on a embedding sequence

Local features extraction

- Each Kernel / Filter application gives one local feature.
- The feature relates to a n-gram.
- The kernel size defines the scope of each feature.
- The stride controls the amount of slides.

A sequence of vector $\rightarrow$ a sequence of features

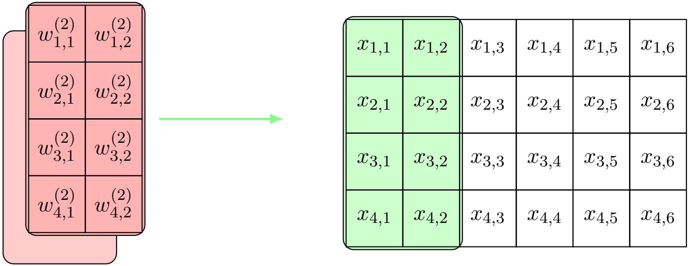
More features ?

- More Filters !
- More "output channels"

Convolution with two output channels

Output channels = 2

Input channels = 1

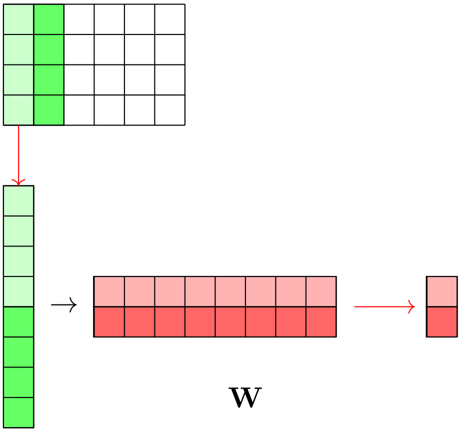


The output value (output channel)

$$h_{1,1} = \sum_{i,j} w^{(1)}_{i,j} \times x_{i,j}$$

$$h_{2,1} = \sum_{i,j} w^{(2)}_{i,j} \times x_{i,j}$$

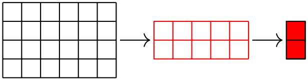
Another view for two output channels



- Two filters applied to the same frame (or window)
- Two projections
- $\mathbf{W}$: the parameters of the filters
- $\mathbf{W}$ is learnt

Pool !

Compress the "local" information along one dimension (e.g time)



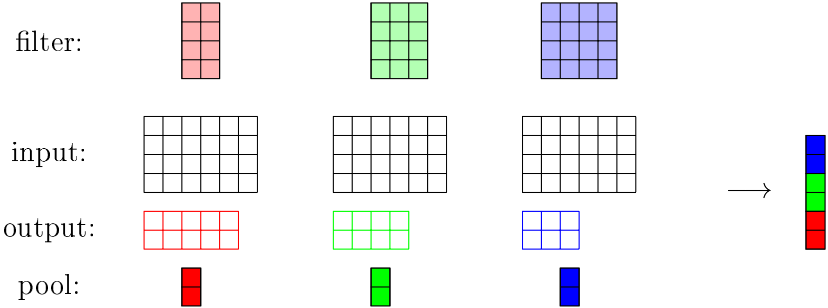
- Mean pooling
- Max pooling, and k-max pooling

For example with:

$$\begin{bmatrix} 1.7 & -0.3 & -0.5 & -2.7 & -0.0 & -0.3 \\ -0.5 & 0.3 & 0.4 & -1.1 & -0.9 & -0.5 \end{bmatrix} \rightarrow \begin{bmatrix} -0.3 \\ -0.4 \end{bmatrix} \text{ or } \begin{bmatrix} 1.7 \\ 0.4 \end{bmatrix} \text{ or } ...$$

Pooling can also apply to sliding windows

More Convolutions and pooling



And they can be combined (concatenation).

Convolutional Neural Networks for Sentence Classification

A summary of [4]

- Window (kernel) sizes : 3, 4, 5 with 100 feature maps for each
- Static/non-static/random/multi-channel word embeddings
- Auxiliary data for word embeddings:~w2v trained on 100 billion words from Google News ($dim = 300$)
- dropout on the penultimate layer (after the max-pooling)
- Relu and early stopping

CNN applications for NLP

Word level

- The unit = a word (the vocabulary ?)
- Compose word representation to derive a sentence representation
- Extract n-gram patterns (phrasal)

Char level

- The unit = the char (closed vocab)
- Compose chars to infer a word representation
- Extract morphological features (mostly concatenative)

  unbelievable, untractable, believer, writter, forever, ...