Neural Nets for text classification

Alexandre Allauzen

Fall 2024

MILES
Machine Intelligence and Learning Systems

Dauphine UNIVERSITÉ PARIS | PSL

ESPCI PARIS | PSL

Roadmap

Outline

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 ?
- How to make classification ?

Bag of words (BOW)

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

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

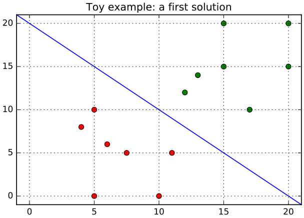
Logistic regression

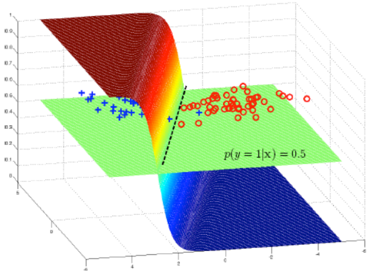
The class c is the outcome of the binary random variable C

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

Back to logistic regression

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

For one input text:

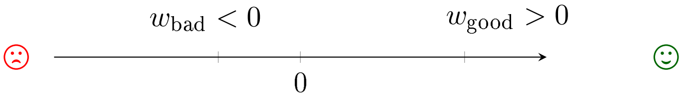
$$w_0 + \mathbf{w}^t\mathbf{x} = w_0 + 2 \times w_2 + w_3 + w_5$$

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:



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

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

with f a non-linear activation function.

Parameters

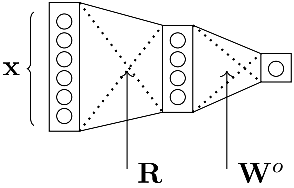
$$\boldsymbol{\theta} = (\mathbf{R}, \mathbf{W^o}) \rightarrow \textbf{to learn !!}$$

Reminder

If $\mathbf{y} = \mathrm{softmax}(\mathbf{a})$, $\mathbf{y}$ 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 [5]

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)

look-up

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

composition

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

Prediction

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

CBOW: details

Fast pre-training of word embeddings

- Introduced in [5] 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 [4]

Other flavor

- Skip-gram [5]
- Glove [6]
- 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

[1] Yoshua Bengio and Réjean Ducharme. "A Neural Probabilistic Language Model". In: *Advances in Neural Information Processing Systems (NIPS).* Vol. 13. Morgan Kaufmann, 2001.

[2] M. Gutmann and A. Hyvärinen. "Noise-contrastive estimation: A new estimation principle for unnormalized statistical models". In: *Proceedings of th International Conference on Artificial Intelligence and Statistics (AISTATS).* Ed. by Y.W. Teh and M. Titterington. Vol. 9. 2010, pp. 297–304.

[3] Armand Joulin et al. "Bag of Tricks for Efficient Text Classification". In: *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers.* Valencia, Spain: Association for Computational Linguistics, Apr. 2017, pp. 427–431. URL: `https://www.aclweb.org/anthology/E17-2068`.

[4] Oren Melamud, Ido Dagan, and Jacob Goldberger. "PMI Matrix Approximations with Applications to Neural Language Modeling". In: *CoRR* abs/1609.01235 (2016). arXiv: `1609.01235`. URL: `http://arxiv.org/abs/1609.01235`.

[5] Tomas Mikolov et al. "Distributed Representations of Words and Phrases and their Compositionality". In: *Advances in Neural Information Processing Systems 26.* Ed. by C.J.C. Burges et al. Curran Associates, Inc., 2013, pp. 3111–3119. URL: `http://papers.nips.cc/paper/5021-distributed-representations-of-words-and-phrases-and-their-compositionality.pdf`.

[6] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. "GloVe: Global Vectors for Word Representation". In: *Empirical Methods in Natural Language Processing (EMNLP)*. 2014, pp. 1532–1543. URL: http://www.aclweb.org/anthology/D14-1162.