Introduction to Deep Learning

Sequence classification and modelling

Alexandre Allauzen

ESPCI PARIS | PSL

Dauphine | PSL
UNIVERSITÉ PARIS

MILES
Machine Intelligence and Learning Systems

10/02/20

Outline

1. Introduction
2. Neural Network language model - part 1
3. Case study: ADN sequence classification
4. Sequence classification with convolution NNet
5. Embedding pre-training

Outline

1 Introduction

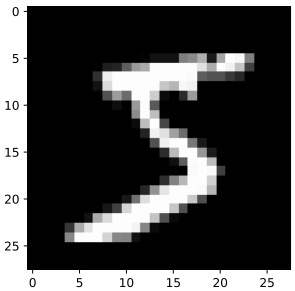
2 Neural Network language model - part 1

3 Case study: ADN sequence classification

4 Sequence classification with convolution NNet

5 Embedding pre-training

Image classification



$$: \quad \boldsymbol{x} \in \mathbb{R}^{784} \longrightarrow c \in \{0, 1, 2, \ldots, 9\}$$

- $\mathcal{D} = (\boldsymbol{x}_{(i)}, c_{(i)})_{i=1}^{N}$
- **Supervised** learning of a **classification** task

Review classification

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

$: \boldsymbol{x} \in \mathbb{R}^D \longrightarrow c \in \{0, 1\}$

- $\mathcal{D} = (\boldsymbol{x}_{(i)}, c_{(i)})_{i=1}^N$
- The input is a sequence $\rightarrow$ how to build $\boldsymbol{x}$?
- A sequence of words (discrete symbols)
- Words interact with each other, the neighborhood is important

Enhancer Identification in DNA Sequences

From (Min et al.2017; Cohn et al.2018)

Enhancer sequences regulate the expression of genes from afar by providing a binding platform for transcription factors, (...). Despite their importance in health and disease, our understanding of these DNA sequences, and their regulatory grammar, is limited. This impairs our ability to identify new enhancers along the genome, or to understand the effect of enhancer mutations and their role in genetic diseases.

Sequence classification task / property identification

A T C G A T C G
... G T A A T C G

$: \boldsymbol{x} \in \mathbb{R}^D \longrightarrow c \in \{0, 1\}$

- Long sequence (arbitrary length)
- Only 4 discrete symbols, but some subsequences are more informative (segments ?)
- Long range interactions between segments

Predicting the sequence specificities of DNA- and RNA-binding proteins

from (Alipanahi et al.2015)

Knowing the sequence specificities of DNA- and RNA-binding proteins is essential for developing models of the regulatory processes in biological systems and for identifying causal disease variants.

Sequence classification / regression task

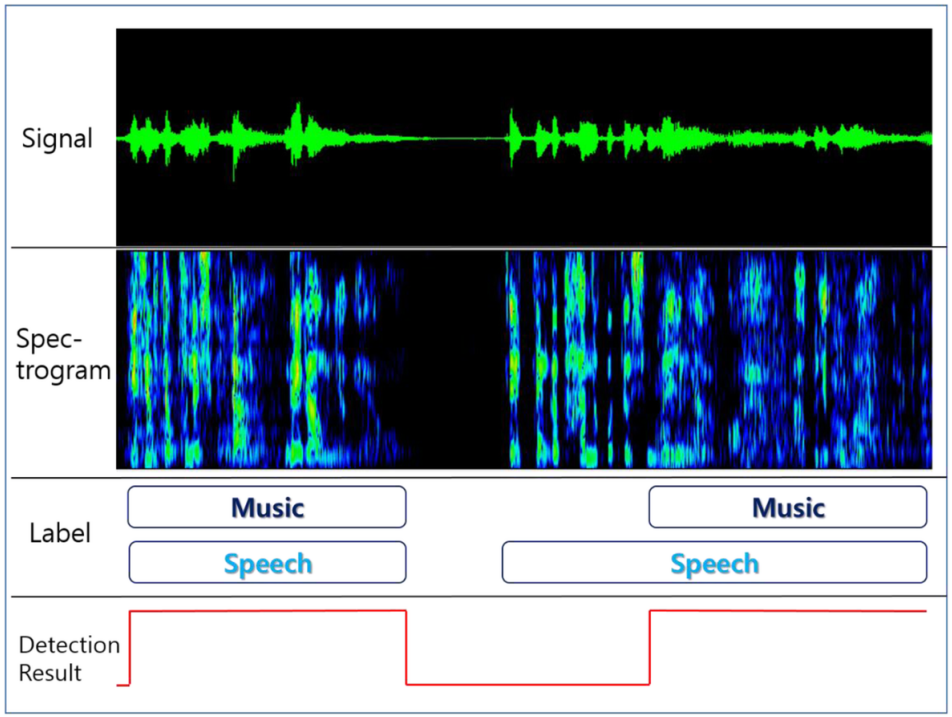
A T C G A T C G
... G T A A T C G

: $\boldsymbol{x} \in \mathbb{R}^D \longrightarrow c \in \{0,1\}$ *or* $y \in \mathbb{R}$

- The data come in qualitatively different forms:
  - Protein binding microarrays (PBMs) and RNAcompete assays provide a specificity coefficient for each probe sequence
  - chromatin mmunoprecipitation (ChIP)-seq10 provides a ranked list of putatively bound sequences of varying length;
  - HT-SELEX11 generates a set of very high affinity sequences.
- Each acquisition technology has its own artifacts, biases and limitations

Audio classification / segmentation

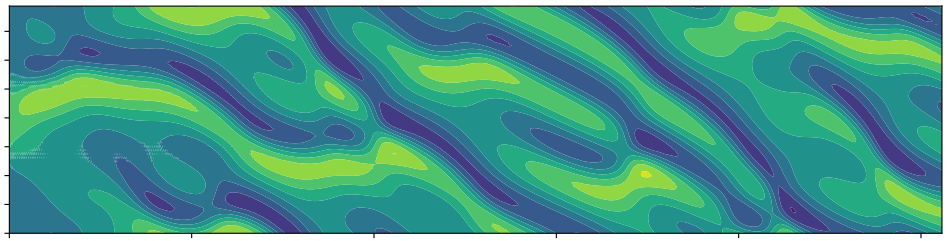
(Jang et al.2019)



- Classification at each time step
- But the context is crucial (for the input and the output) !

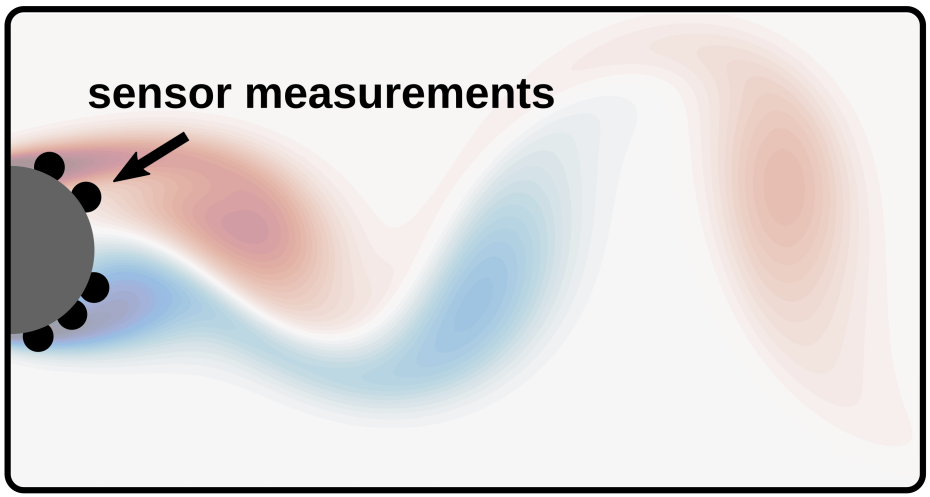
Sequence generation

Predicting the evolution of a dynamical system



$$P(\boldsymbol{y}_{1:T}) = \prod_{i=1}^{T} P(\boldsymbol{y}_i|\boldsymbol{y}_{1:i-1})$$

Sequence generation for sparse observation



$$P(\boldsymbol{y}_{1:T}|\boldsymbol{x}) = \prod_{i=1}^{T} P(\boldsymbol{y}_i|\boldsymbol{y}_{1:i-1}, \boldsymbol{x})$$

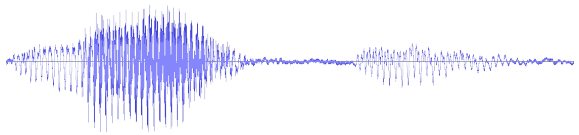
Sequence tagging

$$\mathbf{w} = w_1^L = w_1, w_2, ..., w_L$$
$$\mathbf{t} = t_1^L = t_1, t_2, ..., t_L$$

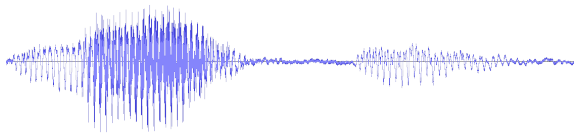
Example : Part-of-Speech (POS) tagging

Sentence	POS-tags
Er	PPER-case=nom|@gender=masc|number=sg|person=3
fürchtet	VVFIN-mood=ind|number=sg|person=3|tense=pres
noch	ADV
Schlimmeres	NN-case=acc|gender=neut|number=sg
.	$.

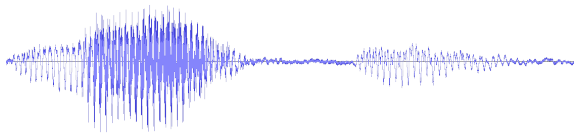
Speech recognition / Machine Translation

Input	*Output*
	[Martine, boude]

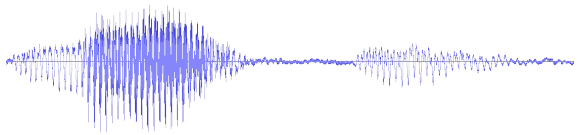
Speech recognition / Machine Translation

Input	*Output*
	[Martine, boude]
[il, est, temps]	[es , ist , zeit]

Speech recognition / Machine Translation

Input	*Output*
	[Martine, boude]
[il, est, temps]	[es , ist , zeit]
$\mathbf{x}$	$\mathbf{y} = (y_1, y_2, ..., y_I)$

Speech recognition / Machine Translation

Input	*Output*
	[Martine, boude]
[il, est, temps]	[es , ist , zeit]
$\mathbf{x}$	$\mathbf{y} = (y_1, y_2, ..., y_I)$

$$P(\mathbf{y}|\mathbf{x}) = \prod_i^I P(y_i|\mathbf{y}_{<\mathbf{i}}, \mathbf{x})$$

- Generate $\mathbf{y}$ from $\mathbf{x}$
- Evaluate $\mathbf{y}$ in the context $\mathbf{x}$

Sequence in machine learning

Sequence classification / regression

- A sequence as input (of arbitrary length)
- Made of discrete symbols of real vectors

→ Predict a value, a class as output

→ Extract features from the sequence (summarize/compress)

→ Represent the sequence as a whole to make a prediction

Generative sequence model

$$P(\mathbf{y}|\mathbf{x}) = \prod_{i}^{I} P(y_i|\mathbf{y}_{<\mathbf{i}}, \mathbf{x})$$

Unsupervised pre-training and supervised fine-tuning

Anotated datasets are usually scarce and sparse, while unannotated data are readily available

Unsupervised pre-training

Train a generative model of the input sequence:

$$P_{\boldsymbol{\theta}}(x_i|\mathbf{x}_{<\mathbf{i}})$$

on unannotated data

Supervised fine-tuning

Train a classifier $P_{\boldsymbol{\phi}}(c_{(i)}|\boldsymbol{x}_{(i)})$ on annotated dataset, but:

- Initialize some parameters (part of $\boldsymbol{\phi}$) with some parts of $\boldsymbol{\theta}$
- Optionally fine tune them.

DNA Sequence *vs* Sentence classification

Text representation

- a text is a structured sequence made of words;
- a word is a discrete symbol;
- belonging to a *finite* set: the vocabulary $\mathcal{V}$.

Compositionality

- The meaning of a complex expression is determined by the meanings of its constituent,
- and the rules used to combine them.

DNA sequence

- Potentially very long sequence made of 4 discrete symbols
- Subsequences can be compared to words, but the segmentation is missing
- Complexe interaction with long range dependancies

Outline

1 Introduction

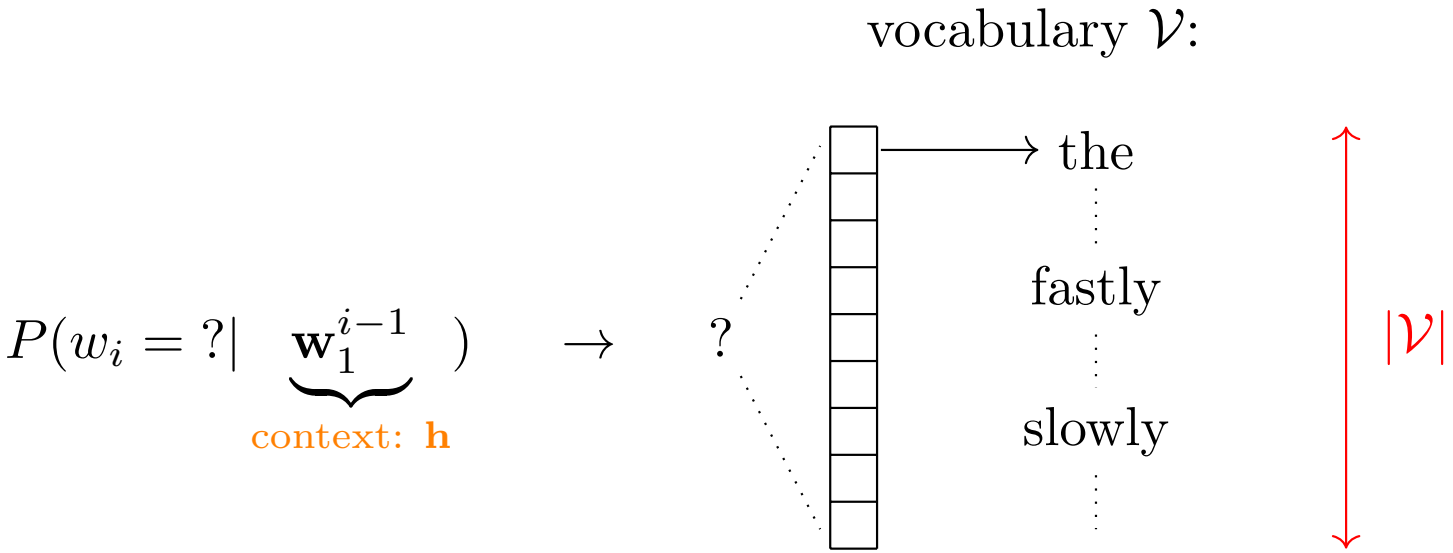
2 Neural Network language model - part 1

3 Case study: ADN sequence classification

4 Sequence classification with convolution NNet

5 Embedding pre-training

A word prediction "game"



- **h** = [time, goes, by, so]
- A probability distribution over $\mathcal{V}$, $\forall$ **h**
- $\mathcal{V}$ is a finite set of words

Language model / Generative sequence model

Applications

Automatic Speech Recognition, Machine Translation, OCR, ...

The goal

Estimate the **non-zero** probability of a word sequence given a vocabulary

$$P(w_1^L) = P(w_1, w_2, ...w_L) = \prod_{i=1}^{L} P(w_i|w_1^{i-1}), \quad \forall i, w_i \in \mathcal{V}$$

with the n**-gram assumption**:

$$P(w_1^L) = \prod_{i=1}^{L} P(w_i|w_{i-n+1}^{i-1}), \quad \forall i, w_i \in \mathcal{V},$$

in the **recurrent** way

$$P(w_i|w_1^{i-1})$$

Challenges

- Share knowledge among similar words

you go to London	you go to Montelimar
you went to London	you went to Montelimar

- Keep/skip what is meaningful/meaningless

Dr. Janet who visisted us last year Smith talks about ...

- Long distance dependancies

to greatly play **chess** he wants to have a nice **board**
to greatly play **music** he wants to buy a new **keyboard**

Count-based language model

n-gram LM

$$P(w_1^L) = \prod_{i=1}^{L} P(w_i | w_{i-n+1}^{i-1}), \quad \forall i, w_i \in \mathcal{V},$$

$$P_{ML}(w_i | w_{i-n+1}^{i-1}) = \frac{c(w_{i-n+1}^{i})}{c(w_{i-n+1}^{i-1})}$$

- Very inefficient !
- How to deal with zero counts in the denominator ?
- In the numerator ?
- Zero counts are the most frequent case.

$\rightarrow$ smoothing (Kneser and Ney1995; Chen and Goodman1998)

Estimate n-gram probabilities in a continuous space

Introduced in (Bengio and Ducharme2001; Bengio et al.2003)

In a nutshell

1. associate each word with a continuous feature vector
2. express the probability function of a word sequence in terms of the feature vectors of these words
3. learn simultaneously the feature vectors and the parameters of that probability function.

Why should it work ?

- "similar" words are expected to have a similar feature vectors
- the probability function is a smooth function of these feature values
  - a small change in the features will induce a small change in the probability
  - Remember: London and Montelimar

Word (discrete symbols) embeddings

One-hot encoding

$$\text{Assume the vocabulary:} \quad \left.\begin{array}{l} \textit{the} \\ \textbf{this} \\ \textit{awesome} \\ \ldots \\ \textit{great} \end{array}\right\} \rightarrow \boldsymbol{x} = \begin{pmatrix} 0 \\ \mathbf{1} \\ 0 \\ \ldots \\ 0 \end{pmatrix}$$

Look-up table

$$\underbrace{\mathbf{R}}_{\text{Look-up table}} \times \boldsymbol{x} = \begin{pmatrix} \vdots & \vdots & \vdots & \vdots & \vdots \\ \mathbf{v}_{the} & \mathbf{v}_{this} & \mathbf{v}_{awesome} & \cdots & \mathbf{v}_{great} \\ \vdots & \vdots & \vdots & \vdots & \vdots \end{pmatrix} \times \begin{pmatrix} 0 \\ \mathbf{1} \\ 0 \\ 0 \\ 0 \end{pmatrix} = \mathbf{v}_{this}$$

Word embeddings

Definitions

- A continous vector associated to each word (discrete symbol): its **embedding**.
- The matrix $\mathbf{R}$ is called the **look-up table** and store the word embeddings.

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

Neural language model basics

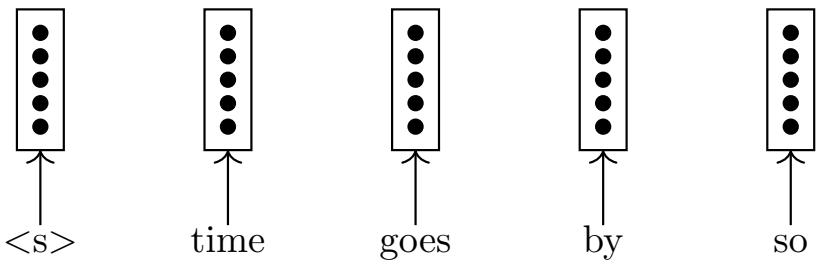
To estimate $P(w_i|w_1^{i-1})$:

<s> time goes by so

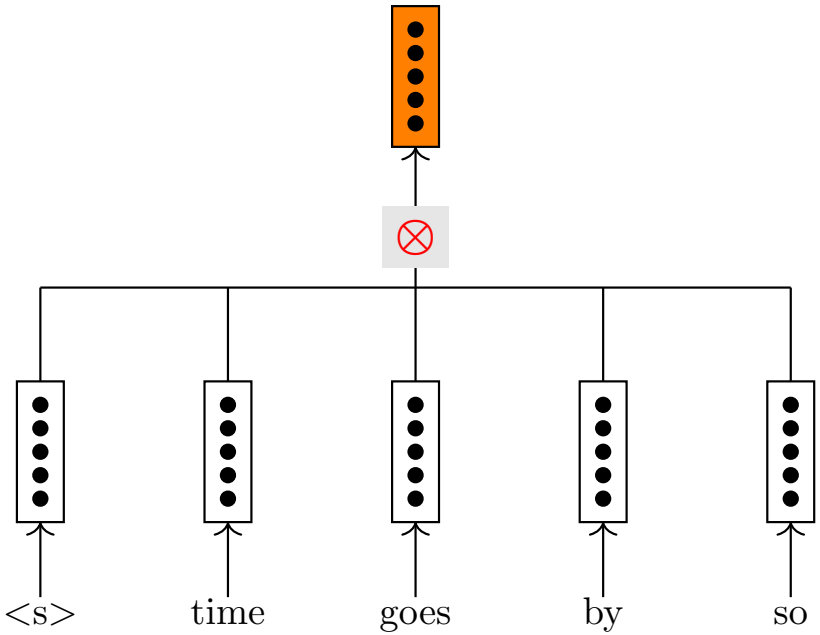
Neural language model basics

To estimate $P(w_i|w_1^{i-1})$:

- each $w_i \in w_1^{i-1} \rightarrow$ embedded



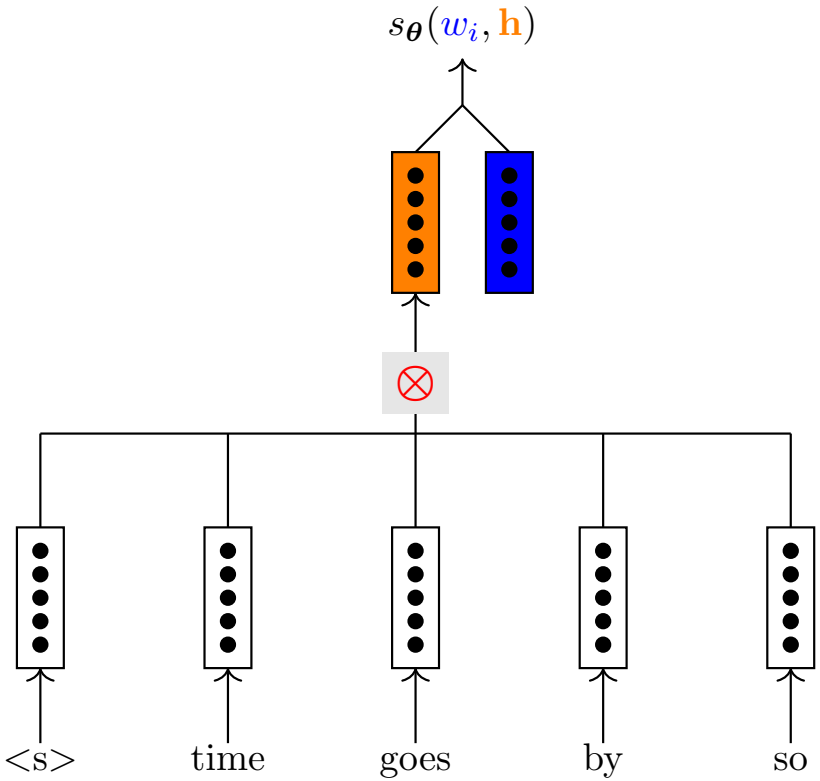
Neural language model basics



To estimate $P(w_i|w_1^{i-1})$:

- each $w_i \in w_1^{i-1} \rightarrow$ embedded
- build a context vector **h** for w_1^{i-1}

Neural language model basics

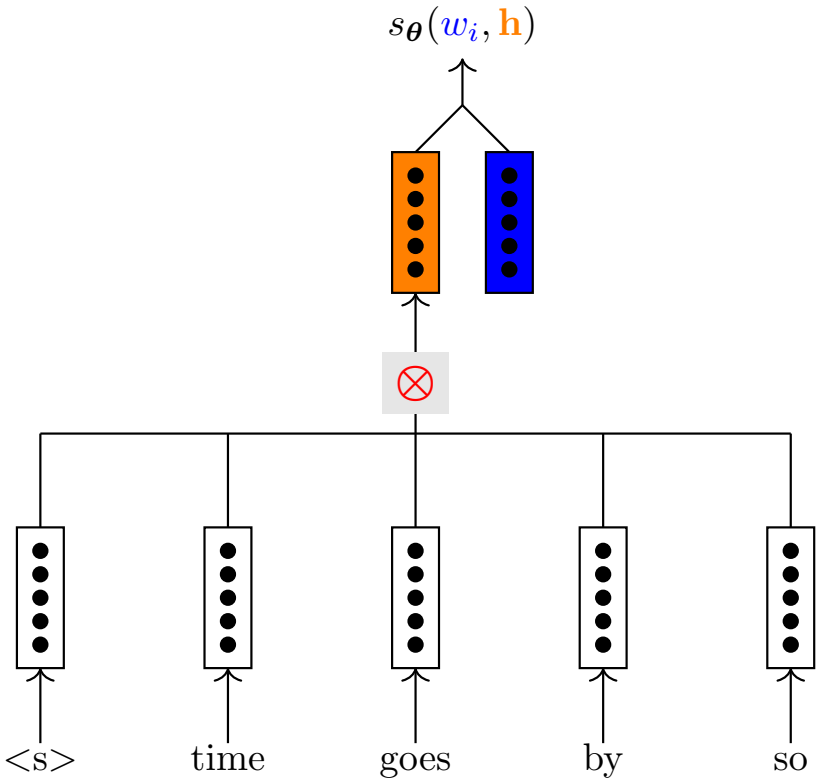


To estimate $P(w_i|w_1^{i-1})$:

- each $w_i \in w_1^{i-1} \rightarrow$ embedded
- build a context vector $\mathbf{h}$ for w_1^{i-1}
- the score of a target word w_i:

$$s_{\boldsymbol{\theta}}(w_i, \mathbf{h})$$

Neural language model basics



To estimate $P(w_i|w_1^{i-1})$:

- each $w_i \in w_1^{i-1} \rightarrow$ embedded
- build a context vector $\mathbf{h}$ for w_1^{i-1}
- the score of a target word w_i:

$$s_{\boldsymbol{\theta}}(w_i, \mathbf{h})$$

Details:

- $\boldsymbol{\theta}$: all the parameters
- $s_{\boldsymbol{\theta}}(w_i, \mathbf{h}) = \langle w_i, \mathbf{h} \rangle$
- The probability distribution:

$$P(w_i|w_1^{i-1}) = \frac{\exp(s_{\boldsymbol{\theta}}(w_i, \mathbf{h}))}{Z(\mathbf{h})}$$

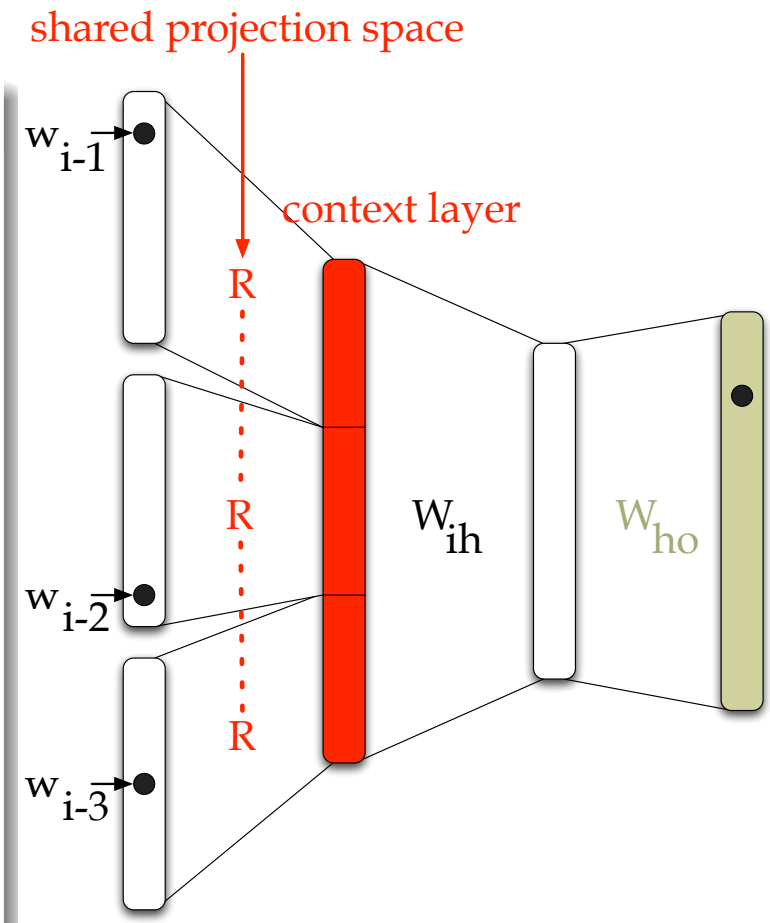
$$Z(\mathbf{h}) = \sum_{w' \in \mathcal{V}} \exp(s_{\boldsymbol{\theta}}(w, \mathbf{h}))$$

- What is $\otimes$?

Estimate the n-gram probability

The program

- Given the history expressed as a feature vector : $\mathbf{v}$ (only the $n-1$ last words)

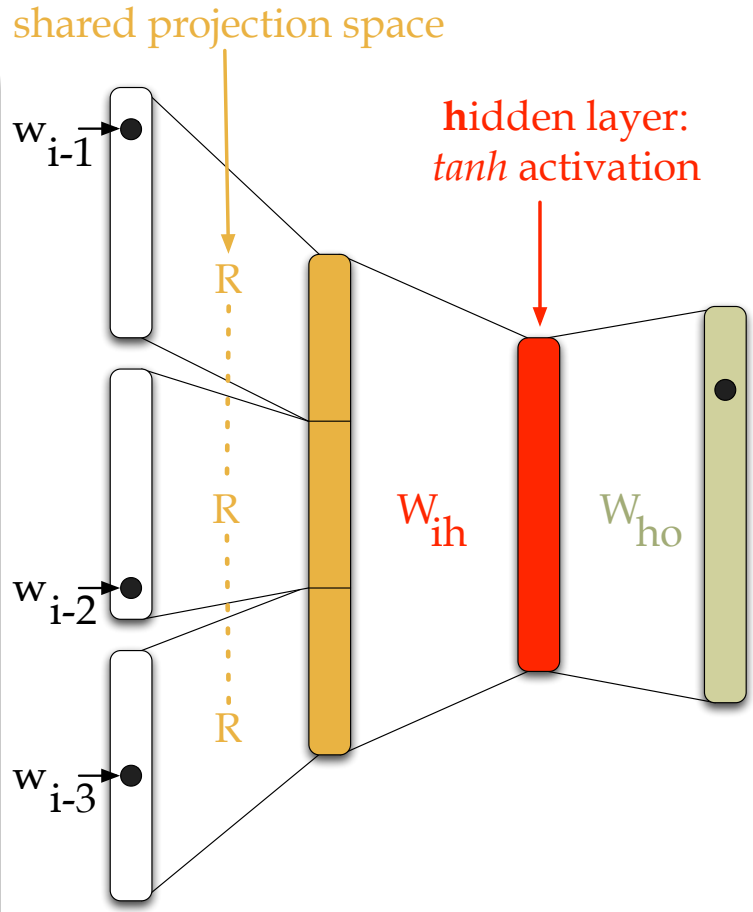


Estimate the n-gram probability

The program

- Given the history expressed as a feature vector : $\mathbf{v}$ (only the $n-1$ last words)
- Create a feature vector for the next word:

$$\mathbf{h} = f(\mathbf{W}_{vh}\mathbf{v})$$



Estimate the n-gram probability

The program

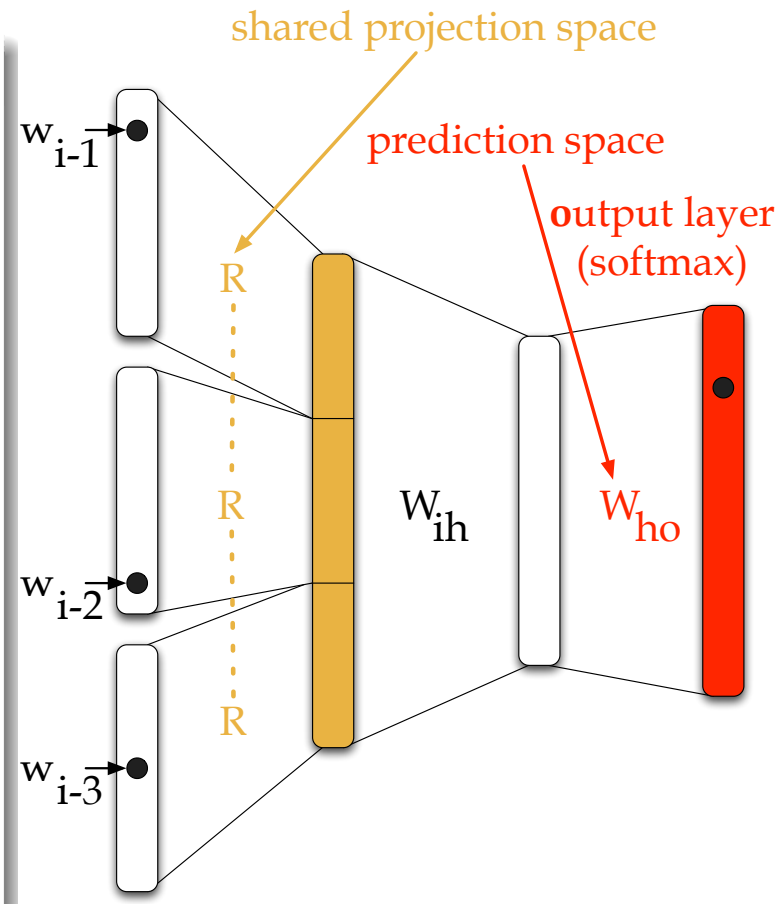
- Given the history expressed as a feature vector : $\mathbf{v}$ (only the $n-1$ last words)
- Create a feature vector for the next word:

$$\mathbf{h} = f(\mathbf{W}_{vh}\mathbf{v})$$

- For all words given the history:

$$\mathbf{o} = f(\mathbf{W}_{ho}\mathbf{h})$$

$$P(w_i|w_{i-n+1}^{i-1}) = \frac{\exp(o_{w_i})}{\sum_{w\in\mathcal{V}}\exp(o_{w_i})}$$



Learning and inference issues

Training objective and gradients

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

$$\frac{\partial \mathcal{L}(\boldsymbol{\theta}; \mathcal{D})}{\partial \boldsymbol{\theta}} = \frac{\partial s_{\boldsymbol{\theta}}(w_i, \mathbf{h})}{\partial \boldsymbol{\theta}} - \sum_{w'\in\mathcal{V}} P_{\boldsymbol{\theta}}(w'|\mathbf{h}) \frac{\partial s_{\boldsymbol{\theta}}(w', \mathbf{h})}{\partial \boldsymbol{\theta}}$$

- Large $|\mathcal{V}|$ is prohibitive
- Prediction in a high dimensional space

Solutions

- Structured output (Le et al.2010; Le et al.2011; Le et al.2013)
- New training criterion (Mnih and Teh2012; Labeau and Allauzen2017)

Outline

1 Introduction

2 Neural Network language model - part 1

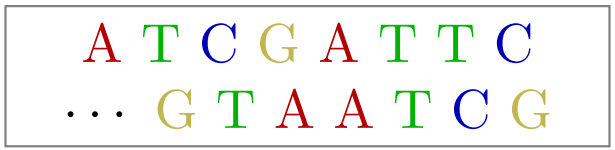
3 Case study: ADN sequence classification

4 Sequence classification with convolution NNet

5 Embedding pre-training

Problem setting

Sequence classification task / property identification



$$: \boldsymbol{x} \in \mathbb{R}^D \longrightarrow c \in \{0, 1\}$$

$$\mathcal{D} = (\boldsymbol{s}_{(i)}, c_{(i)})_{i=1}^{N}$$

Classification assumption

- The input $\boldsymbol{s}_{(i)}$ can be "summarized" in a vector $\boldsymbol{x}_{(i)}$
- The classification is performed by a linear classification (layer)
- The loss can be the binary cross entropy
- How can we process and transform $\boldsymbol{s}_{(i)}$ to build $\boldsymbol{x}_{(i)}$?

Feature engineering

Count based summarization / bag of features

A T C G A T C G G T A A T C G

$\mathcal{V}$	count
A	4
C	3
G	4
T	4

Feature engineering

Count based summarization / bag of features

A T C G A T C G G T A A T C G

$\mathcal{V}$	count
A	4
C	3
G	4
T	4

$+$

$...\mathcal{V}$	count
AA	1
AC	0
AG	0
AT	3
CA	0
CC	0
CG	1
CT	0

$+$

$...\mathcal{V}$	count
GA	1
GC	0
GG	1
GT	0
TA	1
TC	3
TG	0
TT	0

$\Rightarrow \boldsymbol{x}$ of dimension $m + m^2$, with m the number of symbols

k-mers representation (*aka* n-grams)

From (Lee et al.2011)

a k-mers are oligomers of length k

Transform $\boldsymbol{s}$ to $\boldsymbol{x}$ with k-mers encoding

- Count the set of k-mers of varying length (3–10 bp) in $\boldsymbol{s}$
- Generate a vector of d dimensions

$$D = 4^4 + 4^5 + 4^6 + 4^7 + 4^8 + 4^9 + 1(\text{for padding}) = 349441$$

Linear (or not) classifier (SVM)

The model is parametrized with $\boldsymbol{w}$:

$$\boldsymbol{w}^t\boldsymbol{x} = \sum_{j=0}^{D} \underbrace{w_j}_{\text{representation of the oligomer}} \times \underbrace{x_j}_{\text{its count}}$$

ADN sequence classifier

The goal is to design a way to represent the sequence as a vector:

- The oligomer j is represented by w_j: its contribution to the decision
- each oligomer of length k is considered as a discrete symbol
- As k increases, observations are sparse
- No interaction between oligomers (except the value of w_j):

$$\begin{array}{llll} \text{C} & \text{A} & \text{TTGT} & \rightarrow 1.45 \\ & \text{A} & \text{TTGT} \quad \text{C} & \rightarrow 1.19 \\ \text{C} & \text{T} & \text{TTGT} & \rightarrow 1.06 \end{array}$$

- The count summarizes its contribution.

Roadmap

- Symbol embeddings
- Convolution
- Pooling

Outline

1 Introduction

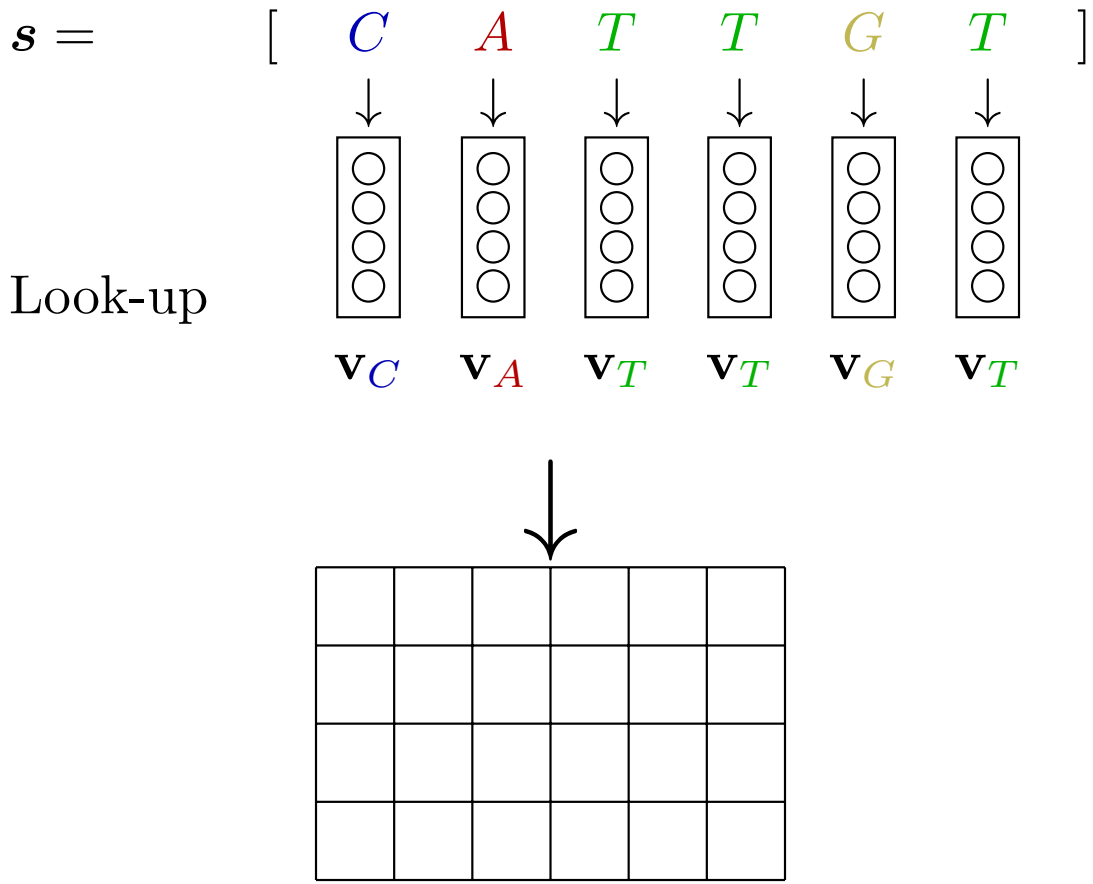
2 Neural Network language model - part 1

3 Case study: ADN sequence classification

4 Sequence classification with convolution NNet

5 Embedding pre-training

Another view of a sequence

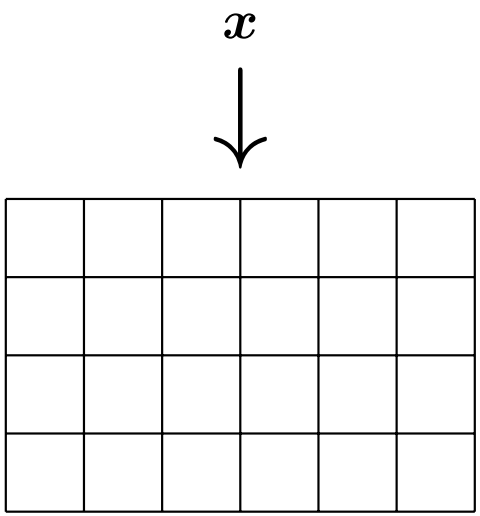


Inspiration

Convolutional Neural Networks for Sentence Classification

- A short paper of 2014 (Kim2014)
- A simple and SOTA paper on text classification

Highlights

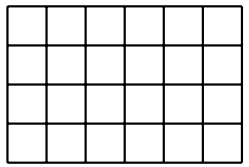


- This input matrix can be seen as an image
- We can extract features with convolution filters
- The filters and the embeddings are parameters

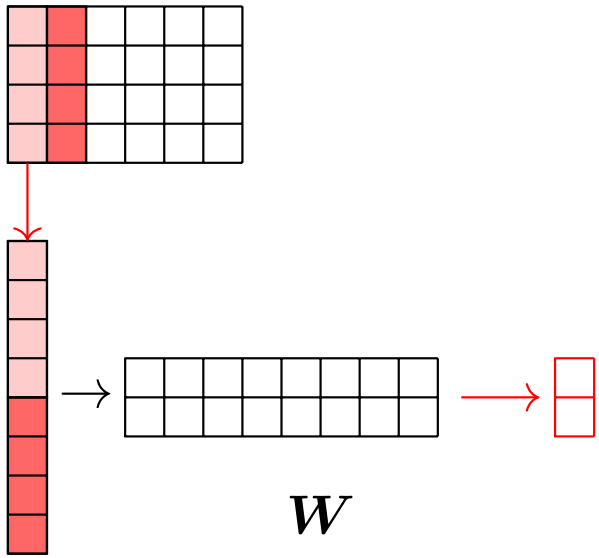
Application to DNA classification in (Min et al.2017; Cohn et al.2018)

Convolution 1D - the first step

The input text is now a matrix $(L = 6, d = 4)$ after the embedding step:



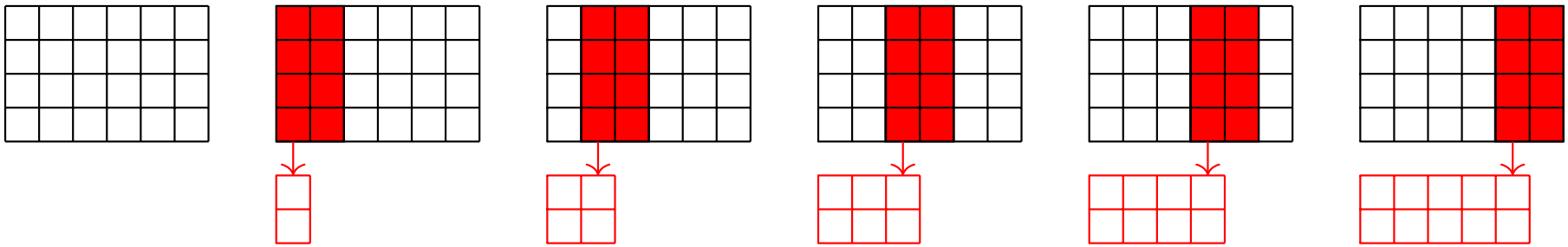
Consider a window size of 2 words:



- 1D : 1 dimension to slide, here it is the time
- $\boldsymbol{W}$ is a matrix to learn

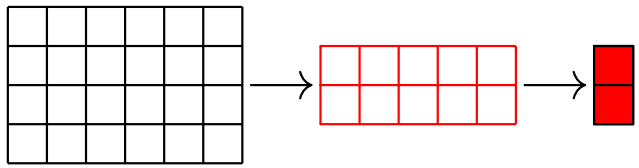
Convolution 1D - sliding

The input text is a matrix $(L = 6, d = 4, od = 2)$ after the embedding step:



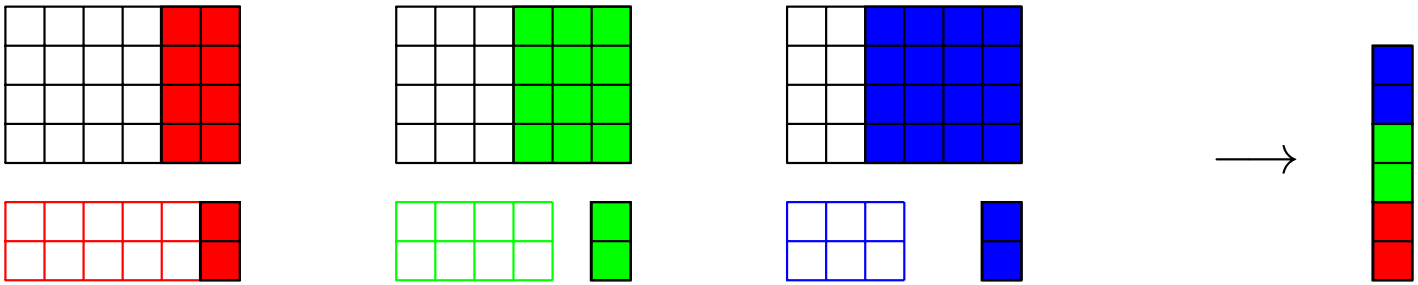
- The transformation: a matrix of size $(od, ws \times d) = (2, 8)$
- ws: window size
- od is sometime called the number of featuremap.

Max-pooling over-time



More Convolutions

The window size can vary:



And they can be combined (concatenation).

Convolutional Neural Networks for Sentence Classification

- 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

Pooling

How to compress information along one dimension (e.g time) ?
→ Pooling

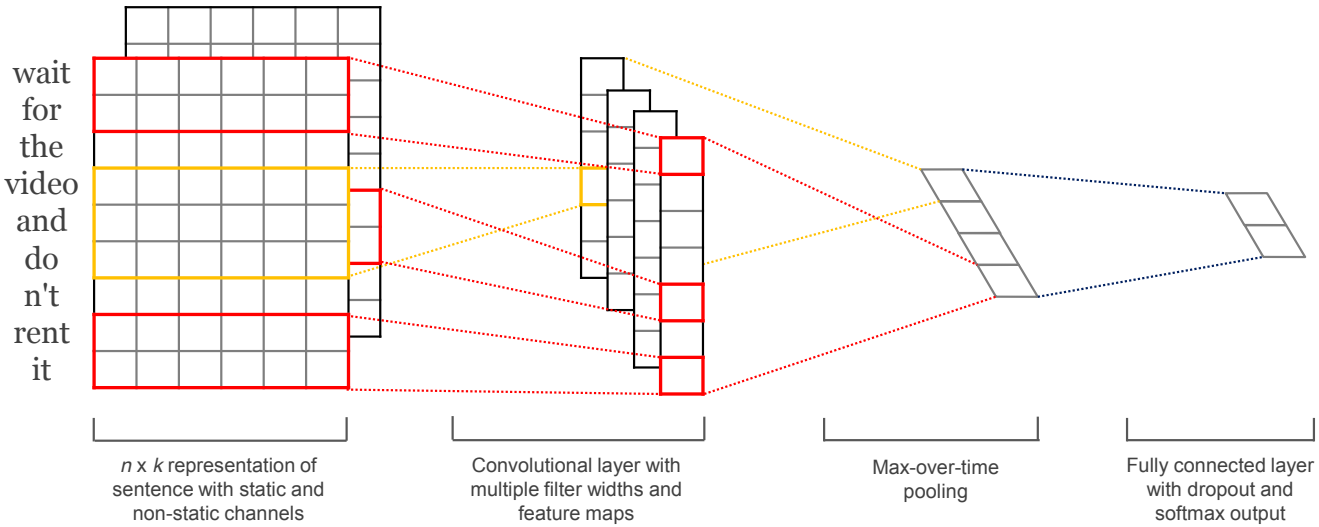
- Max pooling: most common
- Average pooling: like the sum
- k-Max pooling

An overview of sequence convolution

- The CNN layer's responsibility is to extract meaningful sub-structures that are useful for the overall prediction task at hand.
- A convolutional neural network is designed to identify indicative local predictors in a large structure,
- and to combine them to produce a fixed size vector representation of the structure,
- capturing the local aspects that are most informative for the prediction task at hand.
- In the NLP case the convolutional architecture will identify n-grams that are predictive for the task at hand, without the need to pre-specify an embedding vector for each possible n-gram.

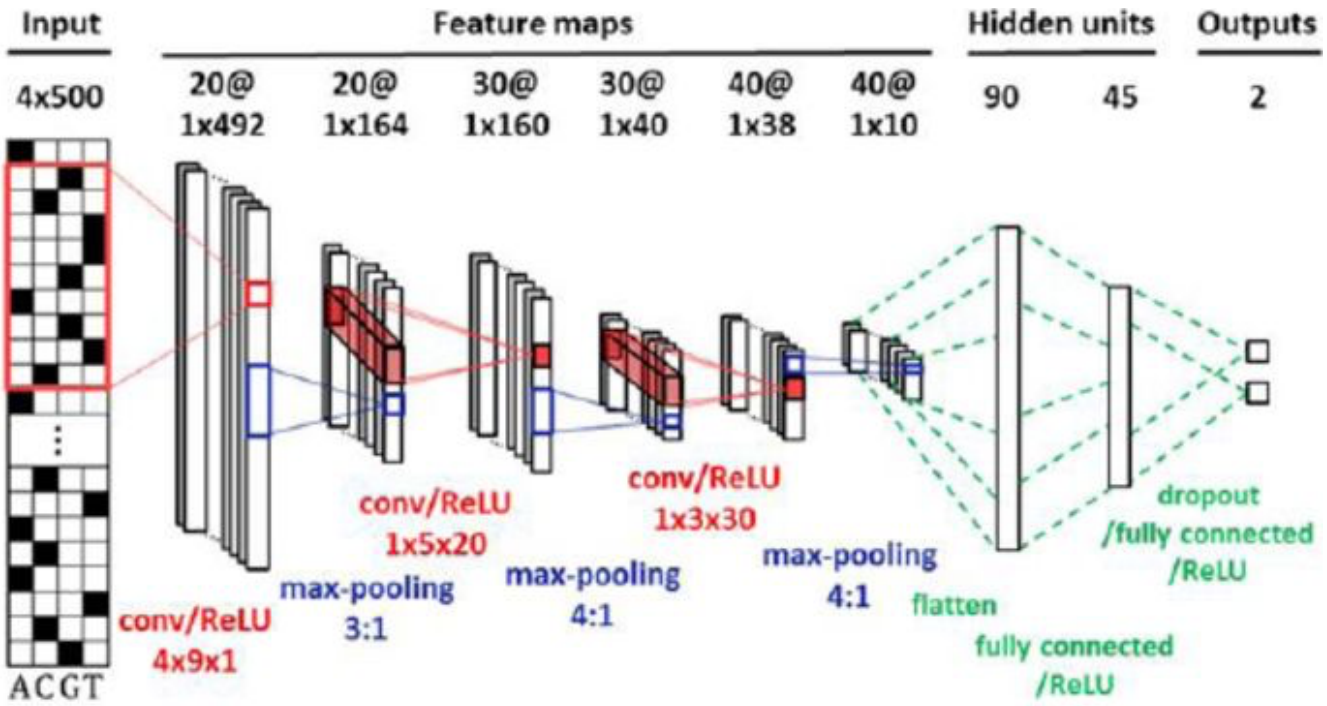
Y. Goldberg in (Goldberg2015)

Sentence classification



From (Kim2014)

Enhancer detection in DNA sequence



From (Min et al.2017)

Outline

1 Introduction

2 Neural Network language model - part 1

3 Case study: ADN sequence classification

4 Sequence classification with convolution NNet

5 Embedding pre-training

Unsupervised Pre-training of Word Embeddings

How to 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)*

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

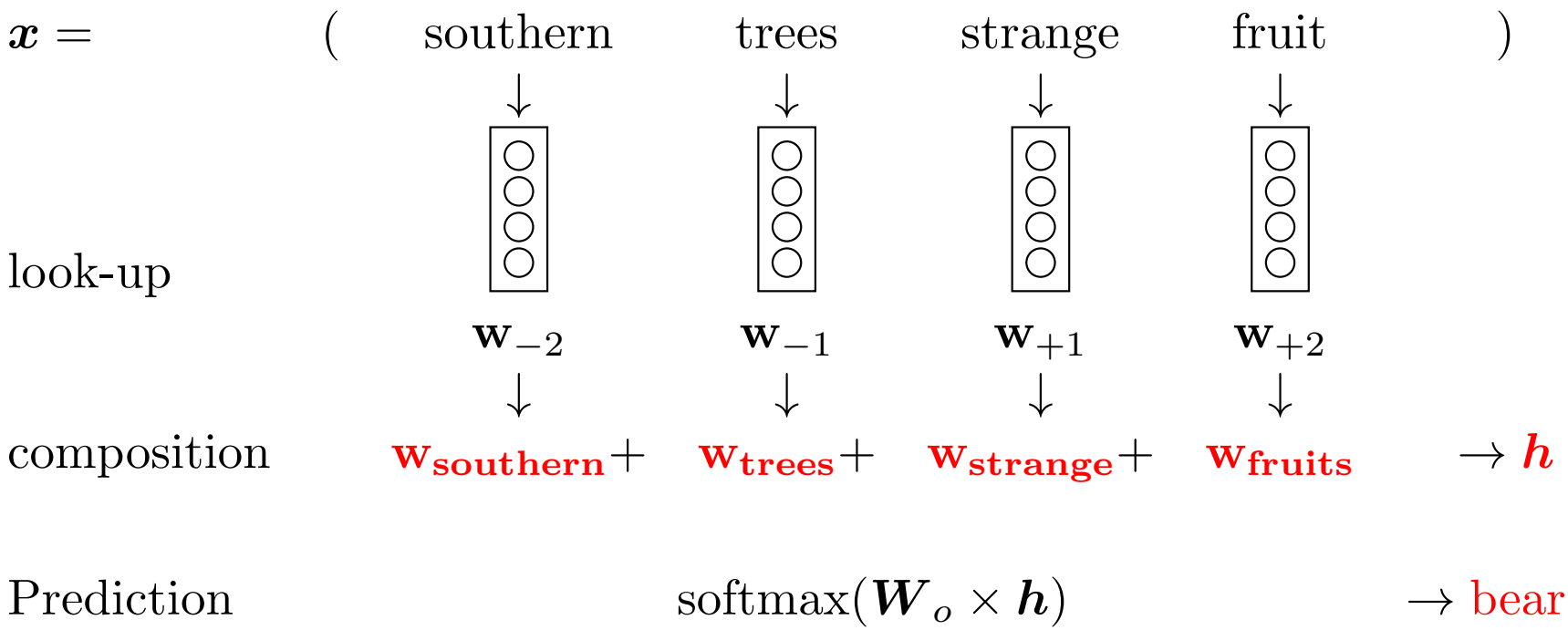
Count-based Methods

- Create a word-context count matrix
- Count the number of co-occurrences of word/context, with rows as word, columns as contexts
- Maybe weight with pointwise mutual information
- Reduce dimensions using SVD
- Measure their closeness using cosine similarity (or others)

Scalability is the bottleneck !

Context Bag of Words (CBOW)

Word2vec - first flavor



CBOW: details

Fast pre-training of word embeddings

- Introduced in (Mikolov et al.2013) as a simplification of (Bengio and Ducharme2001) (neural language model)
- Trained with negative sampling (Closed to Noise Contrastive Estimation (Gutmann and Hyvärinen2010))
- An efficient and tractable approximation of the count based method (Melamud et al.2016)

Other flavor

- Skip-gram (Mikolov et al.2013)
- Glove (Pennington et al.2014)
- Fastext (Joulin et al.2017)

Application to protein embeddings

From (Asgari and Mofrad2015)

n-gram modeling of protein informatics

- Generation of 3 lists of shifted non-overlapping "words"
- The procedure is applied on all 546,790 sequences in Swiss-Prot: a corpus with $546,790 \times 3 = 1,640,370$ sequences of 3-grams
- A 3-gram is a "biological" word consisting of 3 amino acids.

Original Sequence

$$^{(1)}\overrightarrow{M}{}^{(2)}\overrightarrow{A}{}^{(3)}\overrightarrow{F}SAEDVLKEYDRRRRRMEAL..$$

Splittings

$$\begin{cases} 1) & \text{MAF, SAE, DVL, KEY, DRR, RRM, ..} \\ 2) & \text{AFS, AED, VLK, EYD, RRR, RME, ..} \\ 3) & \text{FSA ,EDV, LKE, YDR, RRR, MEA, ..} \end{cases}$$


Babak Alipanahi, Andrew Delong, Matthew T Weirauch, and Brendan J Frey.
2015.
Predicting the sequence specificities of dna- and rna-binding proteins by deep learning.
Nature Biotechnology, 33(8):831–838.


Ehsaneddin Asgari and Mohammad R. K. Mofrad.
2015.
Continuous distributed representation of biological sequences for deep proteomics and genomics.
PLOS ONE, 10(11):1–15, 11.


Yoshua Bengio and Réjean Ducharme.
2001.
A neural probabilistic language model.
In *Advances in Neural Information Processing Systems (NIPS)*, volume 13. Morgan Kaufmann.


Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin.
2003.
A neural probabilistic language model.
Journal of Machine Learning Research, 3:1137–1155.


Stanley F. Chen and Joshua T. Goodman.
1998.

An empirical study of smoothing techniques for language modeling.

Technical Report TR-10-98, Computer Science Group, Harvard University.

Dikla Cohn, Or Zuk, and Tommy Kaplan.

2018.

Enhancer identification using transfer and adversarial deep learning of dna sequences.

bioRxiv.

Yoav Goldberg.

2015.

A primer on neural network models for natural language processing.

M. Gutmann and A. Hyvärinen.

2010.

Noise-contrastive estimation: A new estimation principle for unnormalized statistical models.

In Y.W. Teh and M. Titterington, editors, *Proceedings of th International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 9, pages 297–304.

Byeong-Yong Jang, Woon-Haeng Heo, Jung-Hyun Kim, and Oh-Wook Kwon.

2019.

Music detection from broadcast contents using convolutional neural networks with a mel-scale kernel.

EURASIP Journal on Audio, Speech, and Music Processing, 2019(1):11, Jun.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov.

2017.

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*, pages 427–431, Valencia, Spain, April. Association for Computational Linguistics.

Yoon Kim.

2014.

Convolutional neural networks for sentence classification.

In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751, Doha, Qatar, October. Association for Computational Linguistics.

Reinhard Kneser and Hermann Ney.

1995.

Improved backing-off for m-gram lan- guage modeling.

In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, volume 1, pages 181–184.

Matthieu Labeau and Alexandre Allauzen.

2017.

An experimental analysis of noise-contrastive estimation: the noise distribution matters.

In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 15–20, Valencia, Spain, April. Association for Computational Linguistics.

Hai Son Le, Alexandre Allauzen, Guillaume Wisniewski, and François Yvon.

2010.

Training continuous space language models: Some practical issues.

In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 778–788, Cambridge, MA, October. Association for Computational Linguistics.

Hai-Son Le, Ilya Oparin, Alexandre Allauzen, Jean-Luc Gauvain, and François Yvon.

2011.

Structured output layer neural network language model.

In *Proceedings of the IEEE international conference on Acoustics, speech, and signal processing (ICASSP)*, pages 5524–5527.

Hai Son Le, Ilya Oparin, Alexandre Allauzen, Jean-Luc Gauvain, and François Yvon.

2013.

Structured output layer neural network language models for speech recognition.

IEEE Transactions on Acoustic, Speech and Signal Processing, 21(1):197 – 206, January.

Dongwon Lee, Rachel Karchin, and Michael A. Beer.

2011.

Discriminative prediction of mammalian enhancers from dna sequence.
Genome Research, 21(12):2167–2180.

Oren Melamud, Ido Dagan, and Jacob Goldberger.
2016.
PMI matrix approximations with applications to neural language modeling.
CoRR, abs/1609.01235.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean.
2013.
Distributed representations of words and phrases and their compositionality.
In C.J.C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc.

Xu Min, Wanwen Zeng, Shengquan Chen, Ning Chen, Ting Chen, and Rui Jiang.
2017.
Predicting enhancers with deep convolutional neural networks.
BMC Bioinformatics, 18, 11.

A. Mnih and Y. W. Teh.
2012.
A fast and simple algorithm for training neural probabilistic language models.
In *Proceedings of the International Conference on Machine Learning*.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning.
2014.
Glove: Global vectors for word representation.
In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.