

Deep Learning for Natural Language Processing

Xipeng Qiu
xpqiu@fudan.edu.cn
<http://nlp.fudan.edu.cn>

Weibo:@邱锡鹏

Fudan University

Corpus and Empirical Linguistics Workshop
City University of Hong Kong

June 23, 2016

Table of Contents

1 Basic Concepts

- Artificial Intelligence
- Machine Learning
- Deep Learning
- Implementation

2 Neural Models for Representation Learning

- General Architecture
- Convolutional Neural Network
- Recurrent Neural Network
- Recursive Neural Network
- Attention Model

3 Applications

- Text Generation
- Question Answering
- Machine Translation
- Text Matching

4 Challenges & Open Problems

Table of Contents

1 Basic Concepts

- Artificial Intelligence
- Machine Learning
- Deep Learning
- Implementation

2 Neural Models for Representation Learning

- General Architecture
- Convolutional Neural Network
- Recurrent Neural Network
- Recursive Neural Network
- Attention Model

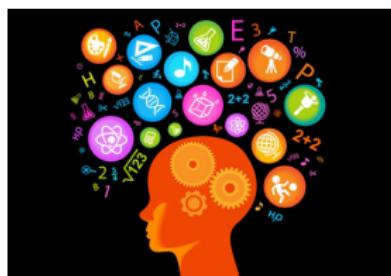
3 Applications

- Text Generation
- Question Answering
- Machine Translation
- Text Matching

4 Challenges & Open Problems

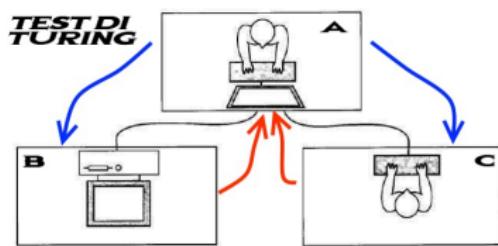
Begin with “AI”

Human: Memory, Computation



Computer: Learning, Thinking, Creativity

Turing Test



“Computing Machinery and Intelligence”, Alan Turing, 1950.

Artificial Intelligence

Definition from Wikipedia

Artificial intelligence (AI) is the intelligence exhibited by machines. Colloquially, the term “artificial intelligence” is likely to be applied when a machine uses cutting-edge techniques to competently perform or mimic “cognitive” functions that we intuitively associate with human minds, such as “learning” and “problem solving”.

Research Topics

- Knowledge Representation
- Machine Learning
- Natural Language Processing
- Computer Vision
- ...

Natural Language Processing (NLP)

What is NLP?

Natural language processing turns to engineering to solve problems that need to analyze (or generate) natural language text.

- Data Driven
- Accuracy Driven
- More Engineered

V.S. Computational Linguistics

- Computational methods to answer the scientific questions of linguistics.
- More theoretic

Challenge: Semantic Gap

Text in Human

床前明月光，
疑是地上霜。
举头望明月，
低头思故乡。

Challenge: Semantic Gap

Text in Computer

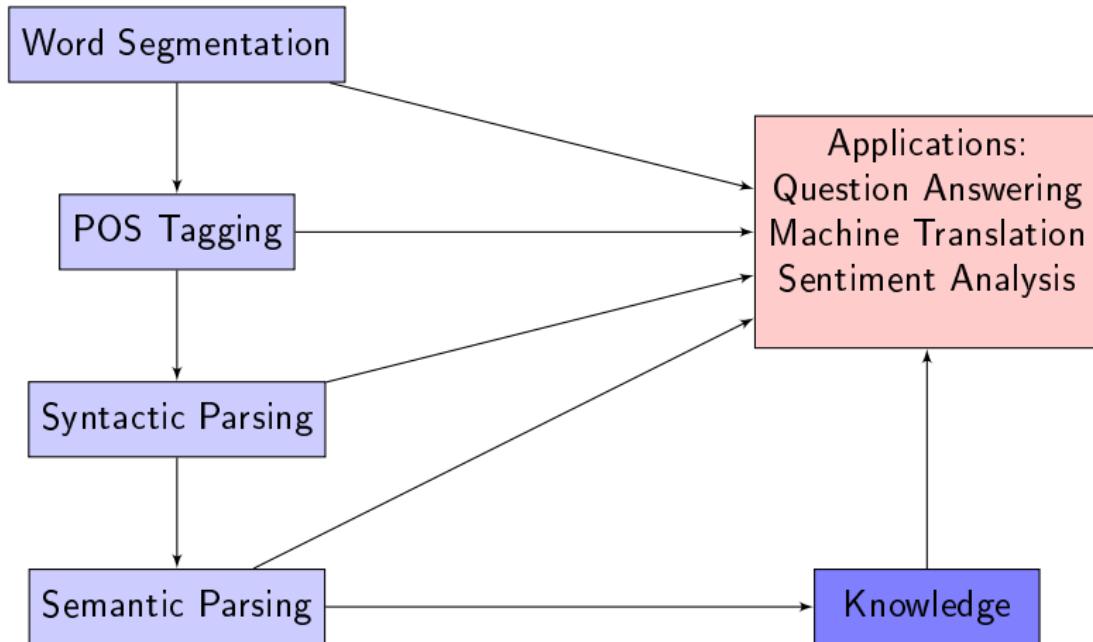
```
1110010110111010100010101110010110001001100110111001101001100010001  
110111001101001110010001000111001011000010110001001111011110010001100  
11100111100101101001000111100110100110001010111111001011001110010110  
0001110010010111000100010101110100110011100100111000111000000010000010  
00100000001000000001000000000101011100100101110001011110111001011001  
0010110100111001101001110010011011110011010011000100011101110011010011100  
100010001110111101111001000110000001010111001001011110110001110111001  
0110100100101101001110011010000000100111011110011010010101100001011110010010  
11100110100001111000111000000010000010
```

Challenge: Sematic Gap

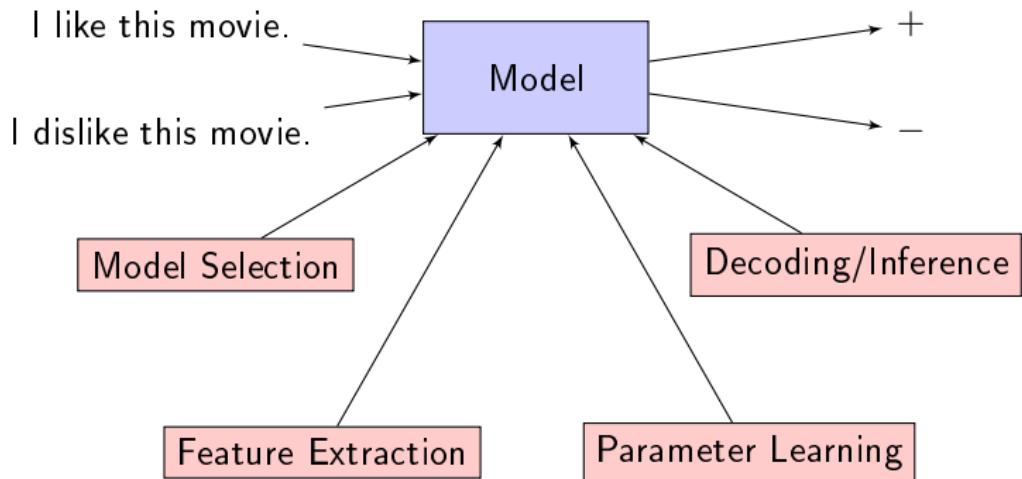


Figure: Guernica (Picasso)

The idea pipeline of NLP

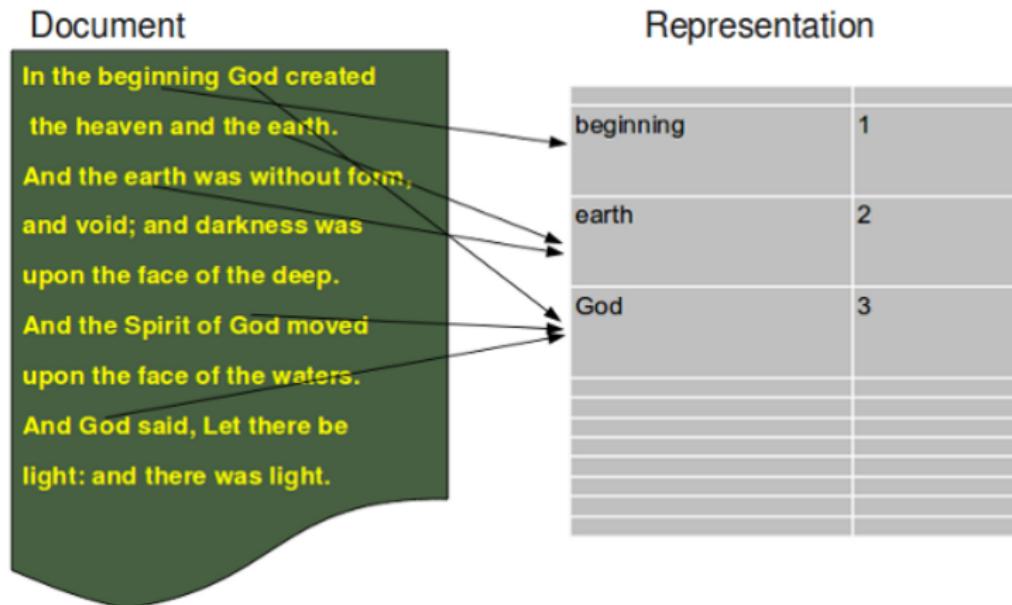


But in practice: End-to-End



Feature Extraction

Bag-of-Word



Example: Sentiment Classification

根据文本内容来判断文本的相应类别

D_1 : “我喜欢读书”

D_2 : “我讨厌读书”

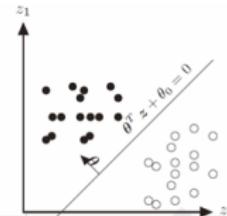


	我	喜欢	讨厌	读书
D_1	1	1	0	1
D_2	1	0	1	1

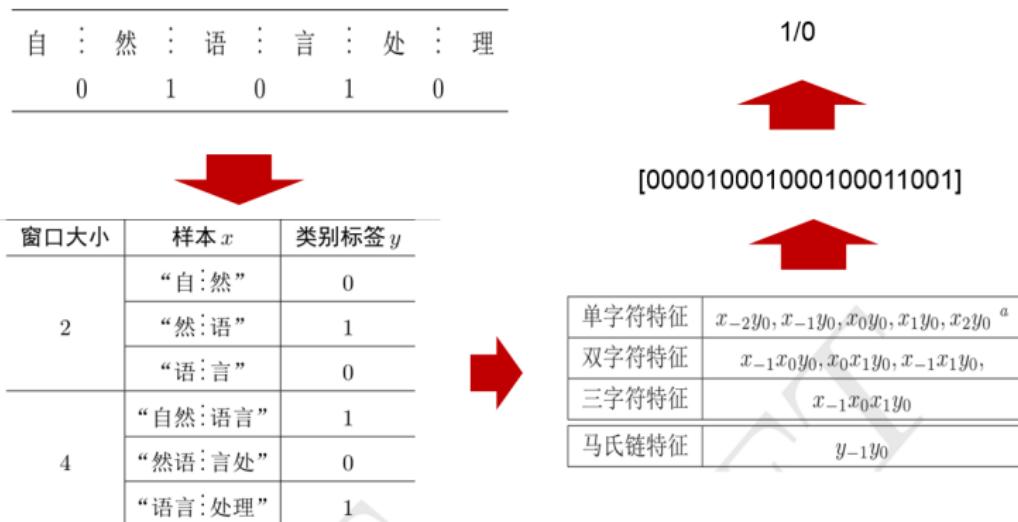


+

-



Example: Chinese Word Segmentation



Example: Chinese Word Segmentation

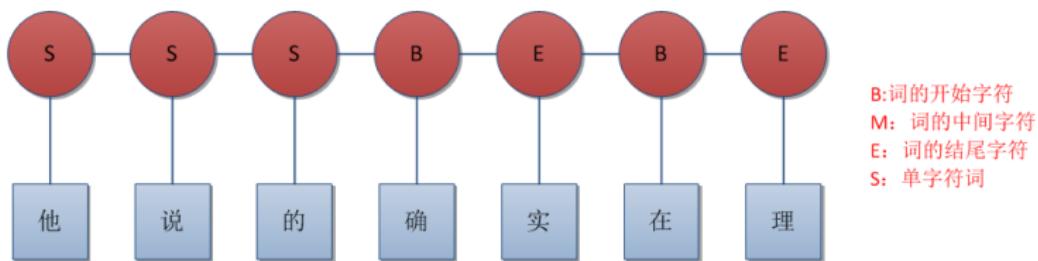


Table of Contents

1 Basic Concepts

- Artificial Intelligence
- Machine Learning
- Deep Learning
- Implementation

2 Neural Models for Representation Learning

- General Architecture
- Convolutional Neural Network
- Recurrent Neural Network
- Recursive Neural Network
- Attention Model

3 Applications

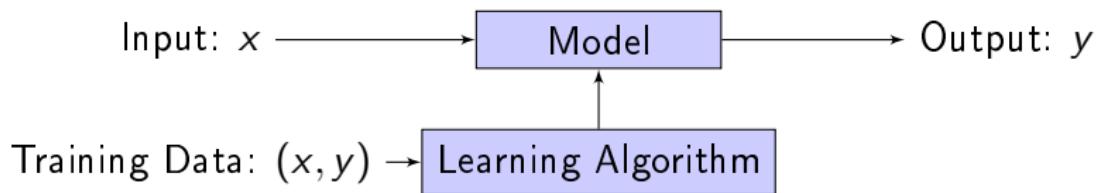
- Text Generation
- Question Answering
- Machine Translation
- Text Matching

4 Challenges & Open Problems

Machine Learning

Definition from Wikipedia

Machine learning explores the study and construction of algorithms that can learn from and make predictions on data.



Formal Specification of Machine Learning

- Input Data: $(x_i, y_i), 1 \leq i \leq m$
- Model:
 - Linear Model: $y = f(x) = w^T x + b$
 - Generalized Linear Model: $y = f(x) = w^T \phi(x) + b$
 - Non-linear Model: Neural Network
- Criterion:
 - Loss Function:
 $L(y, f(x)) \rightarrow \text{Optimization}$
 - Empirical Risk:
 $Q(\theta) = \frac{1}{m} \cdot \sum_{i=1}^m L(y_i, f(x_i, \theta)) \rightarrow \text{Minimization}$
 - Regularization: $\|\theta\|^2$
- Objective Function: $Q(\theta) + \lambda \|\theta\|^2$

Loss Function

Given a test sample (x, y) , the predicted label is $f(x, \theta)$

0-1 Loss

$$L(y, f(x, \theta)) = \begin{cases} 0 & \text{if } y = f(x, \theta) \\ 1 & \text{if } y \neq f(x, \theta) \end{cases} \quad (1)$$

$$= I(y \neq f(x, \theta)), \quad (2)$$

here I is indicator function.

Quadratic Loss

$$L(y, \hat{y}) = (y - f(x, \theta))^2 \quad (3)$$

Loss Function

Cross-entropy Loss We regard $f_i(x, \theta)$ as the conditional probability of class i .

$$f_i(x, \theta) \in [0, 1], \quad \sum_{i=1}^C f_i(x, \theta) = 1 \quad (4)$$

$f_y(x, \theta)$ is the likelihood function of y . Negative Log Likelihood function is

$$L(y, f(x, \theta)) = -\log f_y(x, \theta). \quad (5)$$

Loss Function

We use one-hot vector y to represent class c in which $y_c = 1$ and other elements are 0.

Negative Log Likelihood function can be rewritten as

$$L(y, f(x, \theta)) = - \sum_{i=1}^C y_i \log f_i(x, \theta). \quad (6)$$

y_i is distribution of gold labels. Thus, Eq 6 is Cross Entropy Loss function.

Loss Function

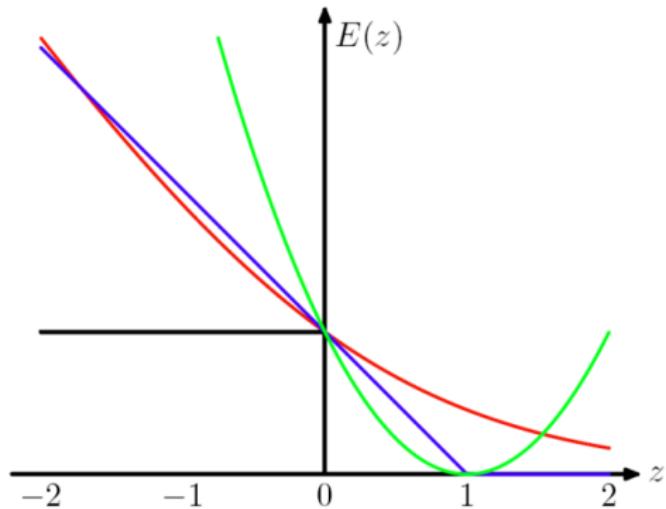
Hinge Loss For binary classification, y and $f(x, \theta)$ are in $\{-1, +1\}$. Hinge Loss is

$$L(y, f(x, \theta)) = \max(0, 1 - yf(x, \theta)) \quad (7)$$

$$= |1 - yf(x, \theta)|_+. \quad (8)$$

Loss Function

For binary classification, y and $f(x, \theta)$ are in $\{-1, +1\}$.
 $z = yf(x, \theta)$.



<http://www.cs.cmu.edu/~yandongl/loss.html>

Parameter Learning

In ML, our objective is to learn the parameter θ to minimize the loss function.

$$\theta^* = \arg \min_{\theta} \mathcal{R}(\theta_t) \quad (9)$$

$$= \arg \min_{\theta} \frac{1}{N} \sum_{i=1}^N \mathcal{L}(y^{(i)}, f(x^{(i)}, \theta)). \quad (10)$$

How to learn θ ?

Parameter Learning

In ML, our objective is to learn the parameter θ to minimize the loss function.

$$\theta^* = \arg \min_{\theta} \mathcal{R}(\theta_t) \quad (9)$$

$$= \arg \min_{\theta} \frac{1}{N} \sum_{i=1}^N \mathcal{L}(y^{(i)}, f(x^{(i)}, \theta)). \quad (10)$$

How to learn θ ?

Gradient Descent!

Gradient Descent

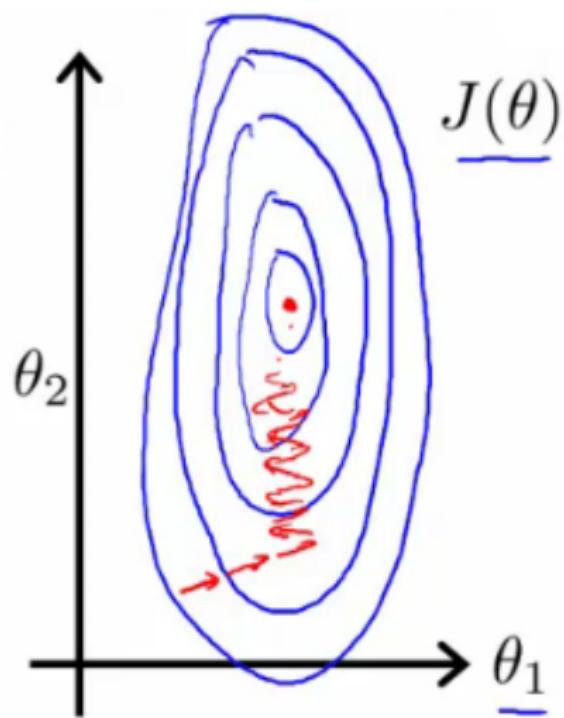
Gradient Descent:

$$\mathbf{a}_{t+1} = \mathbf{a}_t - \lambda \frac{\partial \mathcal{R}(\theta)}{\partial \theta_t} \quad (11)$$

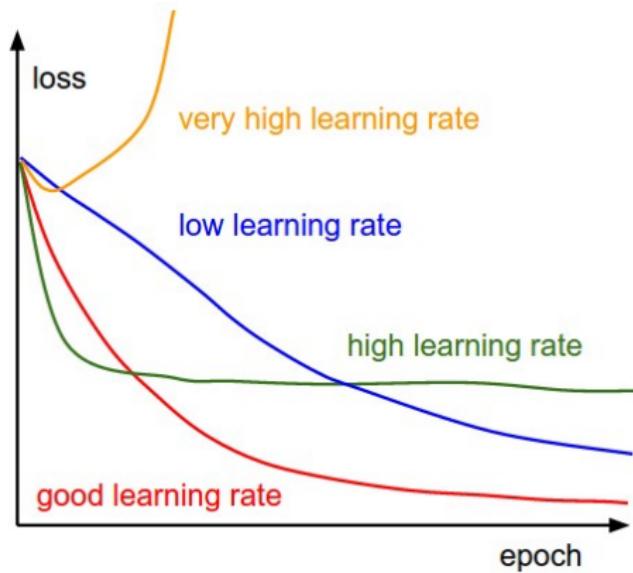
$$= \mathbf{a}_t - \lambda \sum_{i=1}^N \frac{\partial \mathcal{R}(\theta_t; x^{(i)}, y^{(i)})}{\partial \theta}, \quad (12)$$

λ is also called **Learning Rate** in ML.

Gradient Descent



Gradient Descent



Stochastic Gradient Descent (SGD)

It is difficult to handle large scale training data with (batch) Gradient Descent.

Stochastic Gradient Descent (SGD)

It is difficult to handle large scale training data with (batch) Gradient Descent.

We can use **one** (randomly picked) example in each iteration, called Stochastic Gradient Descent.

$$\mathbf{a}_{t+1} = \mathbf{a}_t - \lambda \frac{\partial \mathcal{R}(\theta_t; x^{(t)}, y^{(t)})}{\partial \theta}, \quad (13)$$

Mini-batch Stochastic Gradient Descent

A tradeoff between batch and stochastic gradient descent: Mini-batch SGD.

Mini-batch Stochastic Gradient Descent

A tradeoff between batch and stochastic gradient descent: Mini-batch SGD.

We can use m ($1 \leq m \leq N$) examples in each iteration.

$$\mathbf{a}_{t+1} = \mathbf{a}_t - \lambda \frac{1}{m} \sum_{i \in I_t} \frac{\partial \mathcal{R}(\theta_t; x^{(i)}, y^{(i)})}{\partial \theta}, \quad (14)$$

Two Tricks of SGD

- Early-Stop
- Shuffle

Linear Classification

For binary classification $y \in \{0, 1\}$, the classifier is

$$\hat{y} = \begin{cases} 1 & \text{if } \mathbf{w}^T \mathbf{x} > 0 \\ 0 & \text{if } \mathbf{w}^T \mathbf{x} \leq 0 \end{cases} = I(\mathbf{w}^T \mathbf{x} > 0), \quad (15)$$

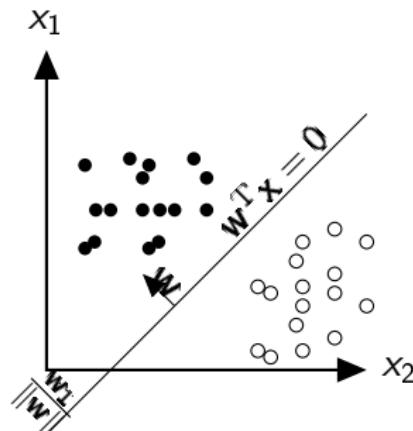


Figure: Binary Linear Classification

Logistic Regression

How to learn the parameter \mathbf{w} : Perceptron, Logistic Regression, etc.
The posterior probability of $y = 1$ is

$$P(y = 1|\mathbf{x}) = \sigma(\mathbf{w}^T \mathbf{x}) = \frac{1}{1 + \exp(-\mathbf{w}^T \mathbf{x})}, \quad (16)$$

where, $\sigma(\cdot)$ is logistic function.

The posterior probability of $y = 0$ is $P(y = 0|\mathbf{x}) = 1 - P(y = 1|\mathbf{x})$.

Logistic Regression

Given n samples $(x^{(i)}, y^{(i)})$, $1 \leq i \leq N$, we use the cross-entropy loss function.

$$\mathcal{J}(\mathbf{w}) = - \sum_{i=1}^N \left(y^{(i)} \log \left(\sigma(\mathbf{w}^T \mathbf{x}^{(i)}) \right) + (1 - y^{(i)}) \log \left(1 - \sigma(\mathbf{w}^T \mathbf{x}^{(i)}) \right) \right) \quad (17)$$

The gradient of $\mathcal{J}(\mathbf{w})$ is

$$\frac{\partial \mathcal{J}(\mathbf{w})}{\partial \mathbf{w}} = \sum_{i=1}^N \left(\mathbf{x}^{(i)} \cdot \left(\sigma(\mathbf{w}^T \mathbf{x}^{(i)}) - y^{(i)} \right) \right) \quad (18)$$

Initialize $\mathbf{w}_0 = 0$, and update

$$\mathbf{w}_{t+1} = \mathbf{w}_t + \lambda \frac{\partial \mathcal{J}(\mathbf{w})}{\partial \mathbf{w}}, \quad (19)$$

Multiclass Classification

Generally, $y = \{1, \dots, C\}$, we define C discriminant functions

$$f_c(\mathbf{x}) = \mathbf{w}_c^T \mathbf{x}, \quad c = 1, \dots, C, \quad (20)$$

where \mathbf{w}_c is weight vector of class c .

Thus,

$$\hat{y} = \arg \max_{c=1}^C \mathbf{w}_c^T \mathbf{x} \quad (21)$$

Softmax Regression

Softmax regression is a generalization of logistic regression to multi-class classification problems.

With **softmax**, the posterior probability of $y = c$ is

$$P(y = c|x) = \text{softmax}(\mathbf{w}_c^\top \mathbf{x}) = \frac{\exp(\mathbf{w}_c^\top \mathbf{x})}{\sum_{i=1}^C \exp(\mathbf{w}_i^\top \mathbf{x})}. \quad (22)$$

To represent class c by one-hot vector

$$\mathbf{y} = [I(1 = c), I(2 = c), \dots, I(C = c)]^\top, \quad (23)$$

where $I()$ is indicator function.

Softmax Regression

Redefine Eq 22,

$$\begin{aligned}\hat{y} &= \text{softmax}(W^T x) \\ &= \frac{\exp(W^T x)}{\underline{1}^T \exp((W^T x))} \\ &= \frac{\exp(z)}{\underline{1}^T \exp(z)},\end{aligned}\tag{24}$$

where,

$$W = [w_1, \dots, w_C],$$

\hat{y} is predicted posterior probability,

$\hat{z} = W^T x$ is input of **softmax** function.

Softmax Regression

Given training set $(\mathbf{x}^{(i)}, \mathbf{y}^{(i)})$, $1 \leq i \leq N$, the cross-entropy loss is

$$\mathcal{J}(W) = - \sum_{i=1}^N \sum_{c=1}^C \mathbf{y}_c^{(i)} \log \hat{\mathbf{y}}_c^{(i)} = - \sum_{i=1}^N (\mathbf{y}^{(i)})^T \log \hat{\mathbf{y}}^{(i)}$$

The gradient of $\mathcal{J}(W)$ is

$$\frac{\partial \mathcal{J}(W)}{\partial \mathbf{w}_c} = - \sum_{i=1}^N \mathbf{x}^{(i)} \left(\mathbf{y}^{(i)} - \hat{\mathbf{y}}^{(i)} \right)_c \quad (25)$$

Table of Contents

1 Basic Concepts

- Artificial Intelligence
- Machine Learning
- Deep Learning
- Implementation

2 Neural Models for Representation Learning

- General Architecture
- Convolutional Neural Network
- Recurrent Neural Network
- Recursive Neural Network
- Attention Model

3 Applications

- Text Generation
- Question Answering
- Machine Translation
- Text Matching

4 Challenges & Open Problems

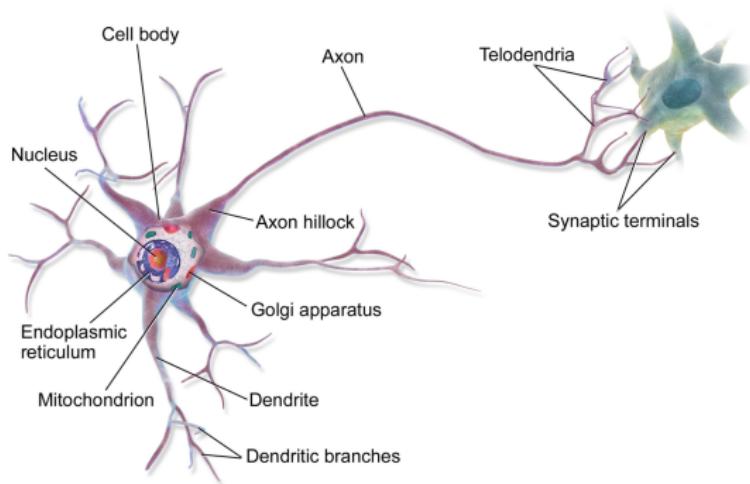
Artificial Neural Network

Artificial neural networks¹ (ANNs) are a family of models inspired by biological neural networks (the central nervous systems of animals, in particular the brain).

Artificial neural networks are generally presented as systems of interconnected “neurons” which exchange messages between each other.

¹https://en.wikipedia.org/wiki/Artificial_neural_network

Biological Neuron



Artificial Neuron

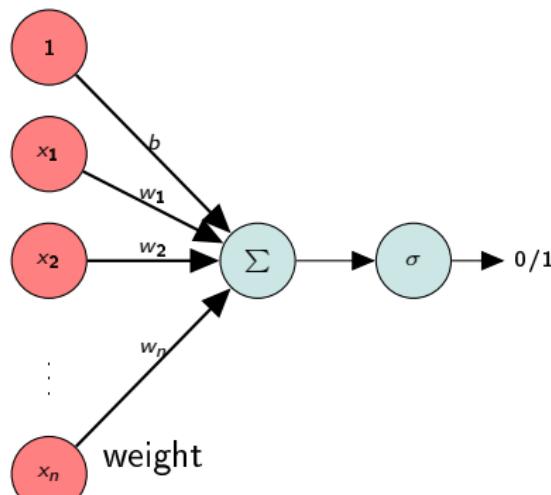
An artificial neuron is a mathematical function to simulate biological neurons. Input: $\mathbf{x} = (x_1, x_2, \dots, x_n)$

State: z

Output: a

$$z = \mathbf{w}^\top \mathbf{x} + b \quad (26)$$

$$a = f(z) \quad (27)$$



Activation Functions

Sigmoid Function:

$$\sigma(x) = \frac{1}{1 + e^{-x}} \quad (28)$$

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \quad (29)$$

$$\tanh(x) = 2\sigma(2x) - 1$$

Activation Functions

rectifier function², also called rectified linear unit (ReLU)³.

$$\text{rectifier}(x) = \max(0, x) \quad (30)$$

softplus function⁴:

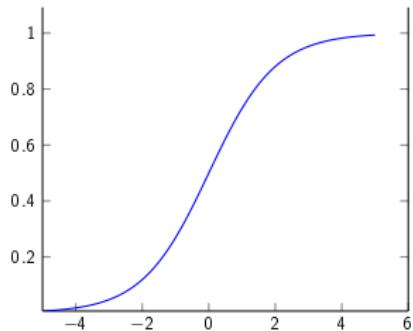
$$\text{softplus}(x) = \log(1 + e^x) \quad (31)$$

² X. Glorot, A. Bordes, and Y. Bengio. “Deep sparse rectifier neural networks”. In: International Conference on Artificial Intelligence and Statistics. 2011, pp. 315–323.

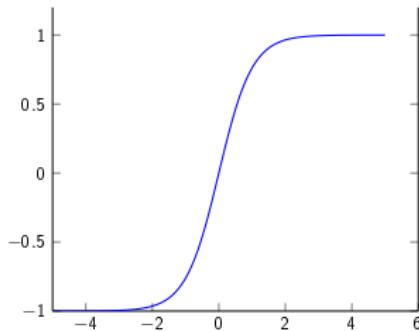
³ V. Nair and G. E. Hinton. “Rectified linear units improve restricted boltzmann machines”. In: Proceedings of the 27th International Conference on Machine Learning (ICML-10). 2010, pp. 807–814.

⁴ C. Dugas et al. “Incorporating second-order functional knowledge for better option pricing”. In: Advances in Neural Information Processing Systems (2001).

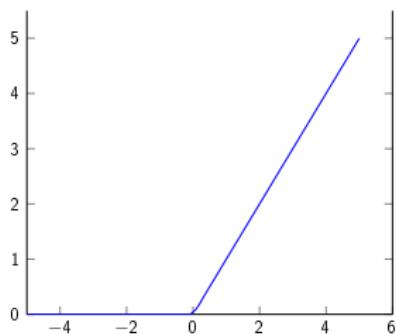
Activation Functions



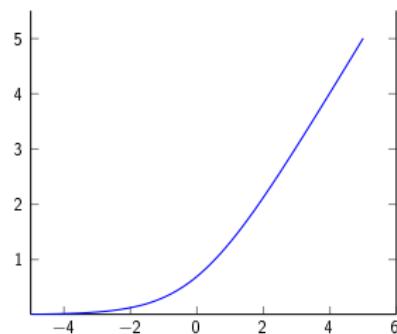
(a) logistic



(b) tanh



(c) rectifier



(d) softplus

Types of Artificial Neural Network⁵

- Feedforward neural network, also called Multilayer Perceptron (MLP).
- Recurrent neural network.

⁵https://en.wikipedia.org/wiki/Types_of_artificial_neural_networks

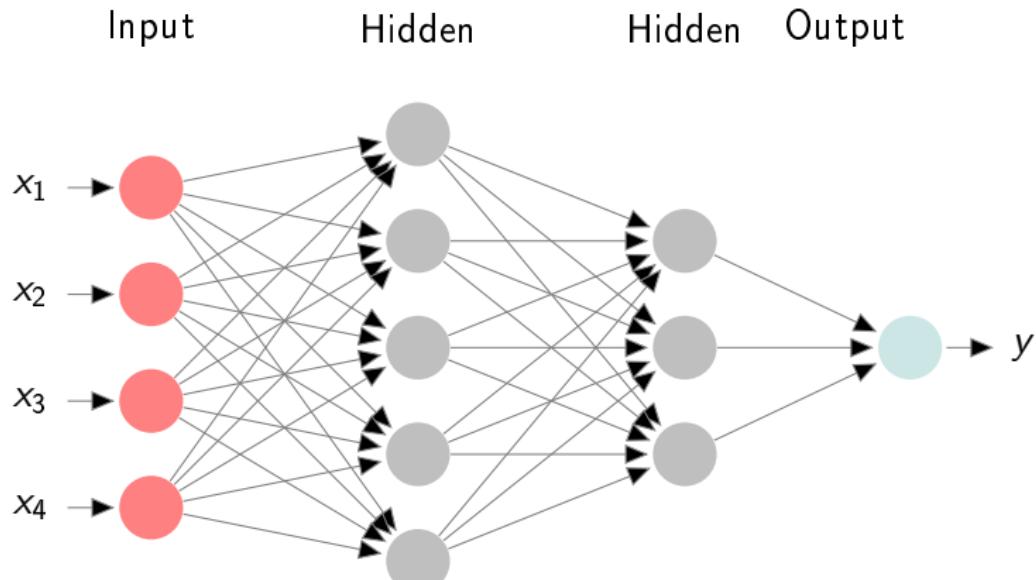
Basic Concepts of Deep Learning

- Model: Artificial neural networks that consist of multiple hidden non-linear layers.
- Function: Non-linear function $y = \sigma(\sum_i w_i x_i + b)$.

Feedforward Neural Network

In feedforward neural network, the information moves in only one direction forward: From the input nodes data goes through the hidden nodes (if any) and to the output nodes.

There are no cycles or loops in the network.



Feedforward Computing

Definitions:

- L Number of Layers;
- n^l Number of neurons in l -th layer;
- $f_l(\cdot)$ Activation function in l -th layer;
- $W^{(l)} \in \mathbb{R}^{n^l \times n^{l-1}}$ weight matrix between $l - 1$ -th layer and l -th layer;
- $\mathbf{b}^{(l)} \in \mathbb{R}^{n^l}$ bias vector between $l - 1$ -th layer and l -th layer;
- $\mathbf{z}^{(l)} \in \mathbb{R}^{n^l}$ state vector of neurons in l -th layer;
- $\mathbf{a}^{(l)} \in \mathbb{R}^{n^l}$ activation vector of neurons in l -th layer.

$$\mathbf{z}^{(l)} = W^{(l)} \cdot \mathbf{a}^{(l-1)} + \mathbf{b}^{(l)} \quad (32)$$

$$\mathbf{a}^{(l)} = f_l(\mathbf{z}^{(l)}) \quad (33)$$

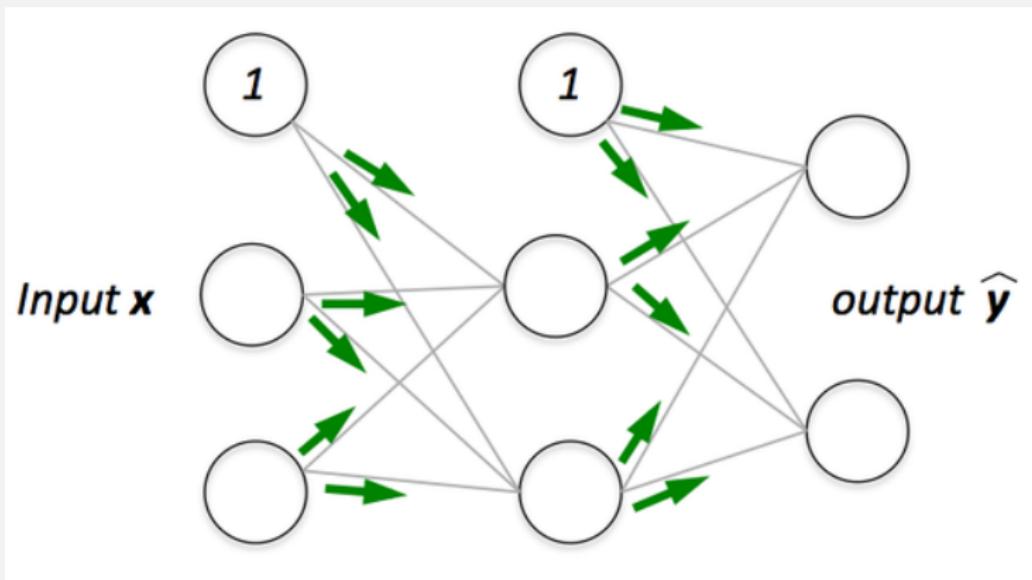
Feedforward Computing

$$\mathbf{z}^{(l)} = W^{(l)} \cdot f_l(\mathbf{z}^{(l-1)}) + \mathbf{b}^{(l)} \quad (34)$$

Thus,

$$\mathbf{x} = \mathbf{a}^{(0)} \rightarrow \mathbf{z}^{(1)} \rightarrow \mathbf{a}^{(1)} \rightarrow \mathbf{z}^{(2)} \rightarrow \cdots \rightarrow \mathbf{a}^{(L-1)} \rightarrow \mathbf{z}^{(L)} \rightarrow \mathbf{a}^{(L)} = y \quad (35)$$

Feedforward Computing



⁵ Image Source: <https://www.quora.com/What-is-the-best-visual-explanation-for-the-back-propagation-algorithm-for-neural-networks>

Combining feedforward network and Machine Learning

Given training samples $(\mathbf{x}^{(i)}, y^{(i)})$, $1 \leq i \leq N$, and feedforward network $f(\mathbf{x}|\mathbf{w}, \mathbf{b})$, the objective function is

$$J(W, \mathbf{b}) = \sum_{i=1}^N L(y^{(i)}, f(\mathbf{x}^{(i)}|W, \mathbf{b})) + \frac{1}{2}\lambda\|W\|_F^2, \quad (36)$$

$$= \sum_{i=1}^N J(W, \mathbf{b}; \mathbf{x}^{(i)}, y^{(i)}) + \frac{1}{2}\lambda\|W\|_F^2, \quad (37)$$

where $\|W\|_F^2 = \sum_{l=1}^L \sum_{j=1}^{n^{l+1}} \sum_{j=}^{n^l} W_{ij}^{(l)}$.

Learning by GD

$$W^{(l)} = W^{(l)} - \alpha \frac{\partial J(W, \mathbf{b})}{\partial W^{(l)}}, \quad (38)$$

$$= W^{(l)} - \alpha \sum_{i=1}^N \left(\frac{\partial J(W, \mathbf{b}; \mathbf{x}^{(i)}, y^{(i)})}{\partial W^{(l)}} \right) - \lambda W, \quad (39)$$

$$\mathbf{b}^{(l)} = \mathbf{b}^{(l)} - \alpha \frac{\partial J(W, \mathbf{b}; \mathbf{x}^{(i)}, y^{(i)})}{\partial \mathbf{b}^{(l)}}, \quad (40)$$

$$= \mathbf{b}^{(l)} - \alpha \sum_{i=1}^N \left(\frac{\partial J(W, \mathbf{b}; \mathbf{x}^{(i)}, y^{(i)})}{\partial \mathbf{b}^{(l)}} \right), \quad (41)$$

(42)

Backpropagation

How to compute $\frac{\partial J(W, \mathbf{b}; \mathbf{x}, y)}{\partial W^{(l)}}$?

$$\frac{\partial J(W, \mathbf{b}; \mathbf{x}, y)}{\partial W_{ij}^{(l)}} = \left(\frac{\partial J(W, \mathbf{b}; \mathbf{x}, y)}{\partial \mathbf{z}^{(l)}} \right)^T \frac{\partial \mathbf{z}^{(l)}}{\partial W_{ij}^{(l)}}. \quad (43)$$

We define $\delta^{(l)} = \frac{\partial J(W, \mathbf{b}; \mathbf{x}, y)}{\partial \mathbf{z}^{(l)}} \in \mathbb{R}^{n^{(l)}}$.

Backpropagation

Because $\mathbf{z}^{(l)} = W^{(l)} \cdot \mathbf{a}^{(l-1)} + \mathbf{b}^{(l)}$,

$$\frac{\partial \mathbf{z}^{(l)}}{\partial W_{ij}^{(l)}} = \frac{\partial (W^{(l)} \cdot \mathbf{a}^{(l-1)} + \mathbf{b}^{(l)})}{\partial W_{ij}^{(l)}} = \begin{bmatrix} 0 \\ \vdots \\ a_j^{(l-1)} \\ \vdots \\ 0 \end{bmatrix} \leftarrow i\text{-th row (44)}$$

Therefore,

$$\frac{\partial J(W, \mathbf{b}; \mathbf{x}, y)}{\partial W_{ij}^{(l)}} = \delta_i^{(l)} a_j^{(l-1)} \quad (45)$$

(46)

Backpropagation

$$\frac{\partial J(W, \mathbf{b}; \mathbf{x}, y)}{\partial W^{(l)}} = \delta^{(l)} (\mathbf{a}^{(l-1)})^\top. \quad (47)$$

In the same way,

$$\frac{\partial J(W, \mathbf{b}; \mathbf{x}, y)}{\partial \mathbf{b}^{(l)}} = \delta^{(l)}. \quad (48)$$

How to compute $\delta^{(l)}$?

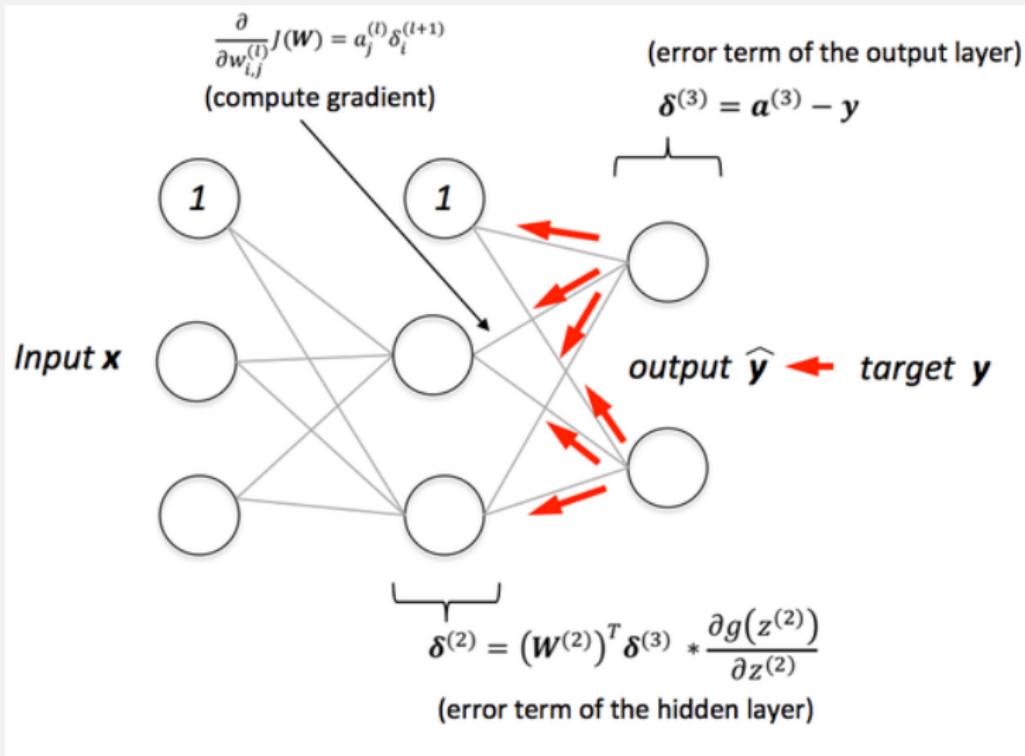
$$\delta^{(l)} \triangleq \frac{\partial J(W, \mathbf{b}; \mathbf{x}, y)}{\partial \mathbf{z}^{(l)}} \quad (49)$$

$$= \frac{\partial \mathbf{a}^{(l)}}{\partial \mathbf{z}^{(l)}} \cdot \frac{\partial \mathbf{z}^{(l+1)}}{\partial \mathbf{a}^{(l)}} \cdot \frac{\partial J(W, \mathbf{b}; \mathbf{x}, y)}{\partial \mathbf{z}^{(l+1)}} \quad (50)$$

$$= \text{diag}(f'_l(\mathbf{z}^{(l)})) \cdot (W^{(l+1)})^\top \cdot \delta^{(l+1)} \quad (51)$$

$$= f'_l(\mathbf{z}^{(l)}) \odot ((W^{(l+1)})^\top \delta^{(l+1)}), \quad (52)$$

Backpropogation Error



Backpropagation Algorithm

Input : Training Set: $(\mathbf{x}^{(i)}, \mathbf{y}^{(i)}), i = 1, \dots, N$, Iteration: T

Output: \mathbf{W}, \mathbf{b}

```

1 Initialize  $\mathbf{W}, \mathbf{b}$  ;
2 for  $t = 1 \dots T$  do
3   for  $i = 1 \dots N$  do
4     (1) Feedforward Computing;
5     (2) Compute  $\delta^{(l)}$  by Eq. 52;
6     (3) Compute gradient of parameters by Eq. 47 and 48;
7        $\frac{\partial J(\mathbf{W}, \mathbf{b}; \mathbf{x}^{(i)}, \mathbf{y}^{(i)})}{\partial \mathbf{W}^{(l)}} = \delta^{(l)} (\mathbf{a}^{(l-1)})^\top$ ;
8        $\frac{\partial J(\mathbf{W}, \mathbf{b}; \mathbf{x}^{(i)}, \mathbf{y}^{(i)})}{\partial \mathbf{b}^{(l)}} = \delta^{(l)}$ ;
9     (4) Update Parameters;
10     $\mathbf{W}^{(l)} = \mathbf{W}^{(l)} - \alpha \sum_{i=1}^N \left( \frac{\partial J(\mathbf{W}, \mathbf{b}; \mathbf{x}^{(i)}, \mathbf{y}^{(i)})}{\partial \mathbf{W}^{(l)}} \right) - \lambda \mathbf{W}$ ;
11     $\mathbf{b}^{(l)} = \mathbf{b}^{(l)} - \alpha \sum_{i=1}^N \left( \frac{\partial J(\mathbf{W}, \mathbf{b}; \mathbf{x}^{(i)}, \mathbf{y}^{(i)})}{\partial \mathbf{b}^{(l)}} \right)$ ;
12  end
13 end

```

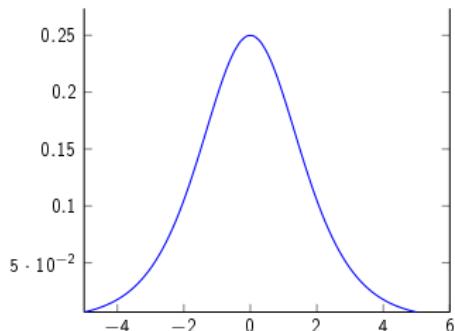
Gradient Vanishing

$$\delta^{(l)} = f'_l(z^{(l)}) \odot ((W^{(l+1)})^\top \delta^{(l+1)}), \quad (53)$$

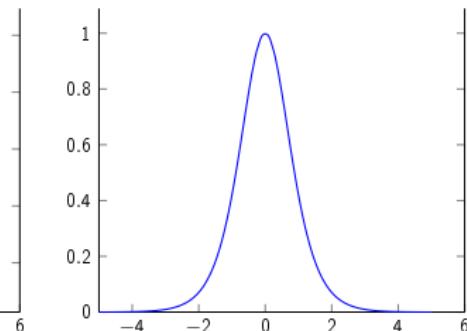
When we use sigmoid function, such as logistic $\sigma(x)$ and \tanh ,

$$\sigma'(x) = \sigma(x)(1 - \sigma(x)) \in [0, 0.25] \quad (54)$$

$$\tanh'(x) = 1 - (\tanh(x))^2 \in [0, 1]. \quad (55)$$



(e) logistic



(f) tanh

Table of Contents

1 Basic Concepts

- Artificial Intelligence
- Machine Learning
- Deep Learning
- Implementation

2 Neural Models for Representation Learning

- General Architecture
- Convolutional Neural Network
- Recurrent Neural Network
- Recursive Neural Network
- Attention Model

3 Applications

- Text Generation
- Question Answering
- Machine Translation
- Text Matching

4 Challenges & Open Problems

How to Implement DL?

- ① Design the topological structure;
- ② Compute the gradient by hand;
- ③ Write Code from scratch;
- ④ Deal with data.

How to Implement DL?

- ① Design the topological structure;
- ② Compute the gradient by hand;
- ③ Write Code from scratch;
- ④ Deal with data.

It's too inefficient!

Using tools with automatic differentiation. Gradient based machine learning algorithms will benefit from the automatic differentiation capabilities.

What I actually do.

Deep Learning



What society thinks I do



What my friends think I do



What other computer scientists think I do



What mathematicians think I do



What I think I do

```
from theano import *
```

What I actually do

⁵ Image Source: <http://www.marekrei.com/blog/theano-tutorial/>

Popular Deep Learning Libraries for NLP

Theano

Theano¹ is a Python library that allows you to define, optimize, and evaluate mathematical expressions involving multi-dimensional arrays efficiently. It can use GPUs and perform efficient symbolic differentiation.

TensorFlow

TensorFlow² is another flexible architecture originally developed by Google Brain Team. TensorFlow allows you to deploy computation to one or more CPUs or GPUs in a desktop, server, or mobile device with a single API.

¹<http://wwwdeeplearningnetsoftwaretheano/>

²<https://www.tensorflow.org/>

Popular Deep Learning Libraries for NLP

Keras (Recommended for non-programmers)

Keras¹ is a minimalist, highly modular neural networks library, written in Python and capable of running on top of either TensorFlow or Theano.

- allows for easy and fast prototyping (through total modularity, minimalism, and extensibility).
- supports both convolutional networks and recurrent networks, as well as combinations of the two.
- supports arbitrary connectivity schemes (including multi-input and multi-output training).
- runs seamlessly on CPU and GPU.

¹<http://keras.io/>

Getting started: 30 seconds to Keras

```
from keras.models import Sequential
from keras.layers import Dense, Activation
from keras.optimizers import SGD

model = Sequential()
model.add(Dense(output_dim=64, input_dim=100))
model.add(Activation("relu"))
model.add(Dense(output_dim=10))
model.add(Activation("softmax"))

model.compile(loss='categorical_crossentropy',
              optimizer='sgd', metrics=['accuracy'])

model.fit(X_train, Y_train, nb_epoch=5, batch_size=32)

loss = model.evaluate(X_test, Y_test, batch_size=32)
```

Difficulties

- Huge Parameters
- Non-convex Optimization
- Gradient Vanishing
- Poor Interpretability

Requirements

- High Computation
- Big Data
- Good Algorithms

Tricks and Skills^{6 7}

- Use ReLU non-linearities
- Use cross-entropy loss for classification
- SGD + mini-batch
 - Shuffle the training samples (← very important)
 - Early-Stop
- Normalize the input variables (zero mean, unit variance)
- Schedule to decrease the learning rate
- Use a bit of L1 or L2 regularization on the weights (or a combination)
- Use “dropout” for regularization
- Data Argument

⁶ G. B. Orr and K.-R. Müller. Neural networks: tricks of the trade. Springer, 2003.

⁷ Geoff Hinton, Yoshua Bengio & Yann LeCun, Deep Learning, NIPS 2015 Tutorial.

Table of Contents

1 Basic Concepts

- Artificial Intelligence
- Machine Learning
- Deep Learning
- Implementation

2 Neural Models for Representation Learning

- General Architecture
- Convolutional Neural Network
- Recurrent Neural Network
- Recursive Neural Network
- Attention Model

3 Applications

- Text Generation
- Question Answering
- Machine Translation
- Text Matching

4 Challenges & Open Problems

How to use neural network for the NLP tasks?

How to encode or learn the representations of word, phrase, sentence, paragraph, or even document?

How to use neural network for the NLP tasks?

How to encode or learn the representations of word, phrase, sentence, paragraph, or even document?

Distributed Representation

Representation of Text

- Local Representation
 - One-hot Representation (Word)
 - N-gram (Phrase)
 - BOW (Document)
- Non-local Representation
 - Distributional Representation (Co-occurrence matrix, Latent Semantic Analysis)
 - Latent Dirichlet Allocation
 - Distributed Representation

Representation of Text

Local Representation

A = 1 0 0 0

B = 0 1 0 0

C = 0 0 1 0

D = 0 0 0 1

Distributed Representation

A = 0.5 0.3 0.0 0.5

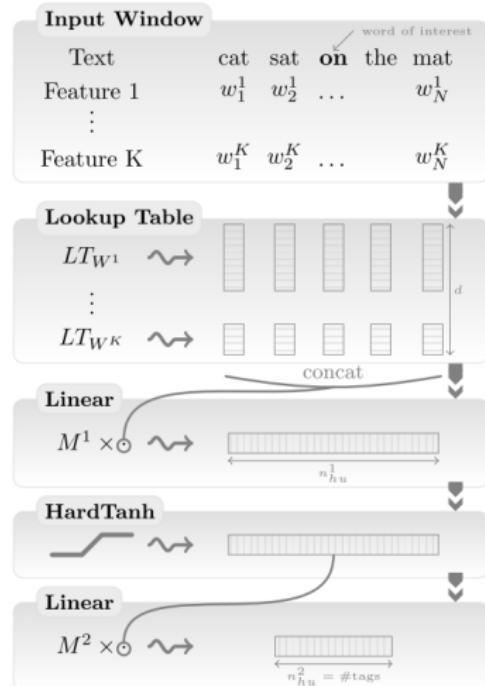
B = -0.1 -0.2 0.0 0.1

C = -0.1 0.5 0.0 -0.5

D = 0.0 0.0 0.5 0.1

Advantages: computable, condensed

General Neural Architectures for NLP⁸



- ① represent the words/features with dense vectors (embeddings) by lookup table;
- ② concatenate the vectors;
- ③ multi-layer neural networks.
 - classification
 - matching
 - ranking

⁸ R. Collobert et al. "Natural language processing (almost) from scratch". In: [The Journal of Machine Learning Research \(2011\)](#).

Difference with the traditional methods

	Traditional methods	Neural methods
Features	Discrete Vector (One-hot Representation)	Dense Vector (Distributed Representation)
	High-dimension	Low-dimension
Classifier	Linear	Non-Linear

Representation Learning for NLP

- **Word Level**

- NNLM
- C&W
- CBOW & Skip-Gram

- **Sentence Level**

- NBOW
- **Sequence Models:** Recurrent NN (LSTM/GRU), Paragraph Vector
- **Topological Models:** Recursive NN,
- **Convolutional Models:** Convolutional NN

- **Document Level**

- NBOW
- **Hierachical Models** two-level CNN
- **Sequence Models** LSTM, Paragraph Vector

Let's start with Language Model

A **statistical language model** is a probability distribution over sequences of words.

A sequence W consists of L words.

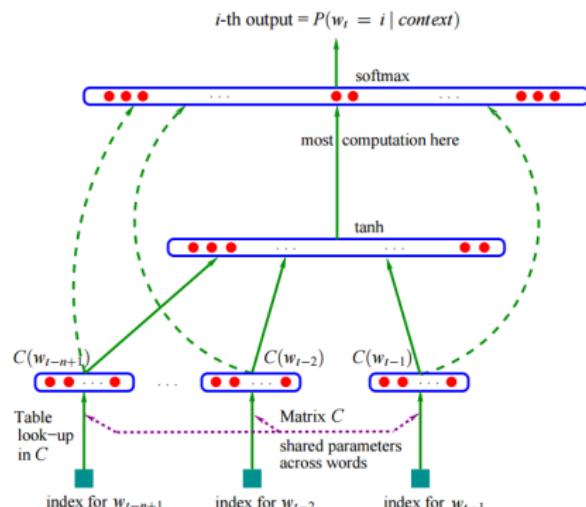
$$\begin{aligned} P(W) &= P(w_{1:L}) = P(w_1, \dots, w_L) \\ &= P(w_1)P(w_2|w_1)P(w_3|w_1 w_2) \cdots P(w_L|w_{1:(L-1)}) \\ &= \prod_{i=1}^L P(w_i|w_{1:(i-1)}). \end{aligned} \tag{56}$$

n-gram model:

$$P(W) = \prod_{i=1}^L P(w_i|w_{(i-n+1):(i-1)}). \tag{57}$$

Neural Probabilistic Language Model⁹

- turn unsupervised learning into supervised learning;
- avoid the data sparsity of n-gram model;
- project each word into a low dimensional space.



⁹ Y. Bengio, R. Ducharme, and P. Vincent. "A Neural probabilistic language model". In: [Journal of Machine Learning Research \(2003\)](#).

Problem of Very Large Vocabulary

Softmax output:

$$P_\theta^h = \frac{\exp(s_\theta(w, h))}{\sum_{w'} \exp(s_\theta(w', h))}, \quad (58)$$

Unfortunately both evaluating P_θ^h and computing the corresponding likelihood gradient requires normalizing over the entire vocabulary

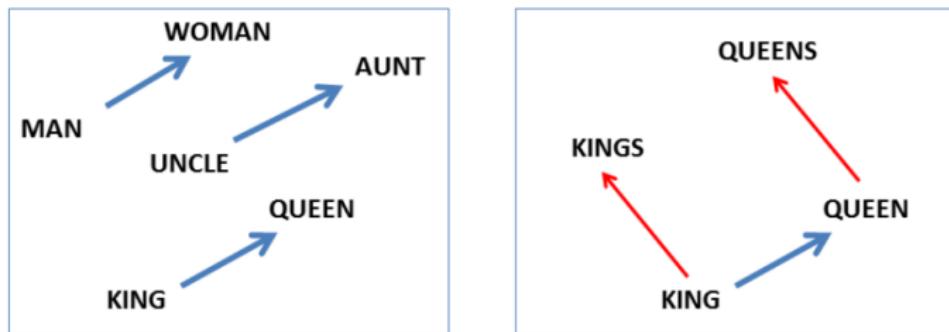
- Hierarchical Softmax: a tree-structured vocabulary¹⁰
- Negative Sampling¹¹, noise-contrastive estimation (NCE)¹²

¹⁰ A. Mnih and G. Hinton. "A scalable hierarchical distributed language model". In: *Advances in neural information processing systems* (2009); F. Morin and Y. Bengio. "Hierarchical Probabilistic Neural Network Language Model." In: *Aistats*. Vol. 5. Citeseer. 2005, pp. 246–252.

¹¹ T. Mikolov et al. "Efficient estimation of word representations in vector space". In: *arXiv preprint arXiv:1301.3781* (2013).

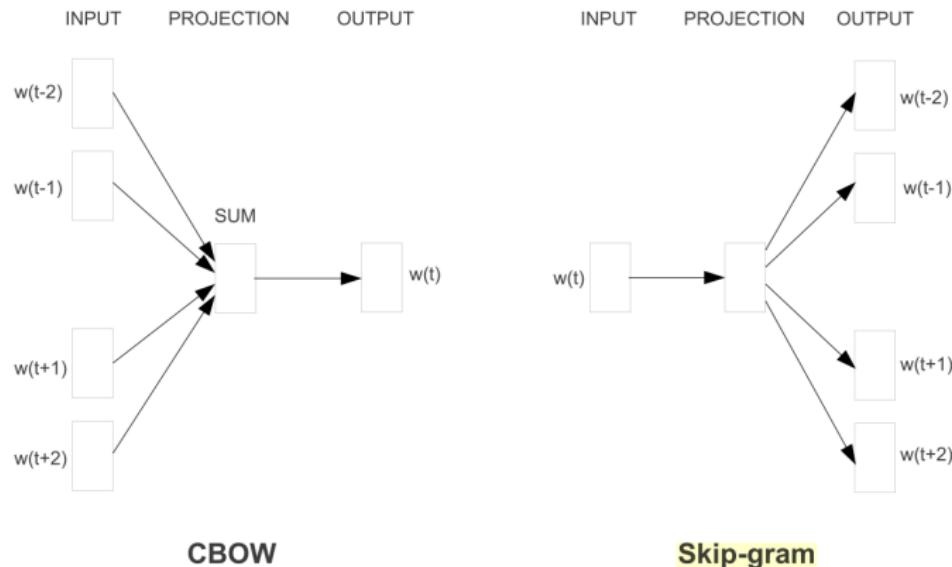
¹² A. Mnih and K. Kavukcuoglu. "Learning word embeddings efficiently with noise-contrastive estimation". In: *Advances in Neural Information Processing Systems*. 2013, pp. 2265–2273.

Linguistic Regularities of Word Embeddings¹³



¹³ T. Mikolov, W.-t. Yih, and G. Zweig. "Linguistic Regularities in Continuous Space Word Representations." In: HLT-NAACL, 2013, pp. 746–751.

Skip-Gram Model¹⁴



¹⁴ Mikolov et al., "Efficient estimation of word representations in vector space".

Skip-Gram Model

Given a pair of words (w, c) , the probability that the word c is observed in the context of the target word w is given by

$$Pr(D = 1|w, c) = \frac{1}{1 + \exp(-\mathbf{w}^T \mathbf{c})},$$

where \mathbf{w} and \mathbf{c} are embedding vectors of w and c respectively.

The probability of not observing word c in the context of w is given by,

$$Pr(D = 0|w, c) = 1 - \frac{1}{1 + \exp(-\mathbf{w}^T \mathbf{c})}.$$

Skip-Gram Model with Negative Sampling

Given a training set \mathcal{D} , the word embeddings are learned by maximizing the following objective function:

$$J(\theta) = \sum_{w,c \in \mathcal{D}} Pr(D = 1|w, c) + \sum_{w,c \in \mathcal{D}'} Pr(D = 0|w, c),$$

where the set \mathcal{D}' is randomly sampled negative examples, assuming they are all incorrect.

Efficiency

<i>Model</i>	<i>Vector Dimensionality</i>	<i>Training Words</i>	<i>Training Time</i>	<i>Accuracy [%]</i>
Collobert NNLM	50	660M	2 months	11
Turian NNLM	200	37M	few weeks	2
Mnih NNLM	100	37M	7 days	9
Mikolov RNNLM	640	320M	weeks	25
Huang NNLM	50	990M	weeks	13
Our NNLM	100	6B	2.5 days	51
Skip-gram (hier.s.)	1000	6B	hours	66
CBOW (negative)	300	1.5B	minutes	72

Table of Contents

1 Basic Concepts

- Artificial Intelligence
- Machine Learning
- Deep Learning
- Implementation

2 Neural Models for Representation Learning

- General Architecture
- **Convolutional Neural Network**
- Recurrent Neural Network
- Recursive Neural Network
- Attention Model

3 Applications

- Text Generation
- Question Answering
- Machine Translation
- Text Matching

4 Challenges & Open Problems

Convolutional Neural Network

Convolutional neural network (CNN, or ConvNet) is a type of feed-forward artificial neural network.

Distinguishing features¹⁵:

- ① Local connectivity
- ② Shared weights
- ③ Subsampling

¹⁵ Y. LeCun et al. "Gradient-based learning applied to document recognition". In: [Proceedings of the IEEE 11 \(1998\)](#).

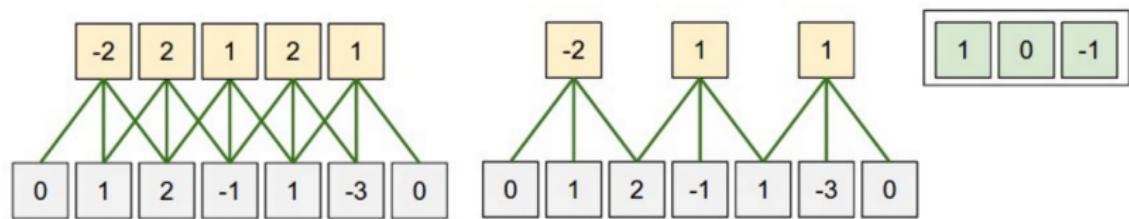
Convolution

Convolution is an integral that expresses the amount of overlap of one function as it is shifted over another function.

Given an input sequence x_t , $t = 1, \dots, n$ and a **filter** f_t , $t = 1, \dots, m$, the convolution is

$$y_t = \sum_{k=1}^n f_k \cdot x_{t-k+1}. \quad (59)$$

One-dimensional convolution



¹⁵Figure from: <http://cs231n.github.io/convolutional-networks/>

Two-dimensional convolution

Given an image x_{ij} , $1 \leq i \leq M$, $1 \leq j \leq N$ and filter f_{ij} , $1 \leq i \leq m$, $1 \leq j \leq n$, the convolution is

$$y_{ij} = \sum_{u=1}^m \sum_{v=1}^n f_{uv} \cdot x_{i-u+1, j-v+1}. \quad (60)$$

Mean filter: $f_{uv} = \frac{1}{mn}$

Convolutional Layer

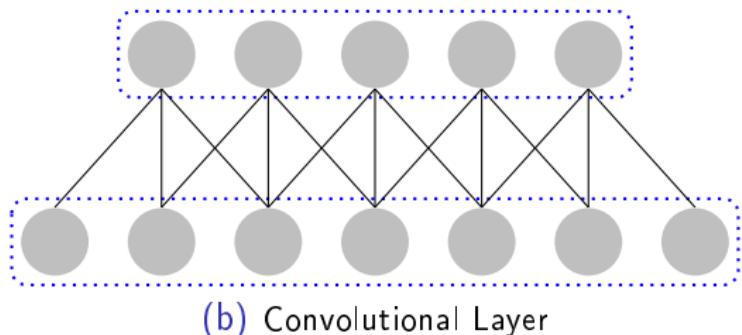
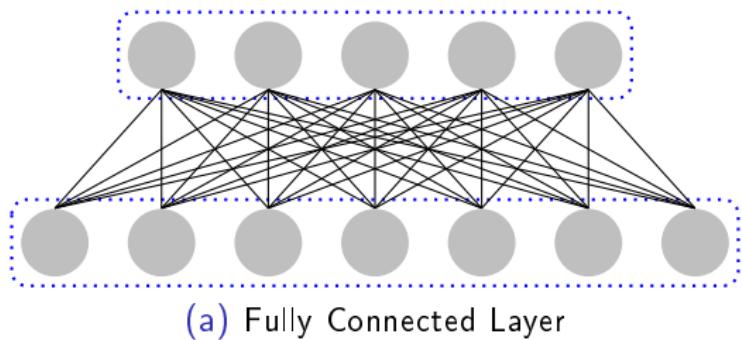
$$\mathbf{a}^{(l)} = f(\mathbf{w}^{(l)} \otimes \mathbf{a}^{(l-1)} + \mathbf{b}^{(l)}), \quad (61)$$

\otimes is convolutional operation.

$\mathbf{w}^{(l)}$ is **shared** by all the neurons of l -th layer.

Just need $m + 1$ parameters, and $n^{(l+1)} = n^{(l)} - m + 1$.

Fully Connected Layer V.S. Convolutional Layer



Pooling Layer

It is common to periodically insert a pooling layer in-between successive convolutional layers.

- progressively reduce the spatial size of the representation
- reduce the amount of parameters and computation in the network
- avoid overfitting

For a feature map $X^{(l)}$, we divide it into several (non-)overlapped regions $R_k, k = 1, \dots, K$. A pooling function **down**(\dots) is

$$X_k^{(l+1)} = f(Z_k^{(l+1)}), \quad (62)$$

$$= f\left(w^{(l+1)} \cdot \text{down}(R_k) + b^{(l+1)}\right). \quad (63)$$

Pooling Layer

$$X^{(l+1)} = f(Z^{(l+1)}), \quad (64)$$

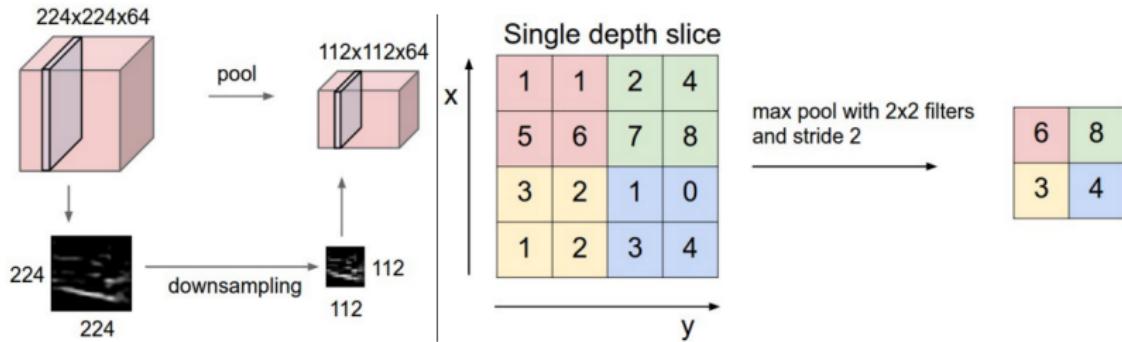
$$= f \left(w^{(l+1)} \cdot \text{down}(X^l) + b^{(l+1)} \right), \quad (65)$$

Two choices of **down**(\cdot): Maximum Pooling and Average Pooling.

$$pool_{max}(R_k) = \max_{i \in R_k} a_i \quad (66)$$

$$pool_{avg}(R_k) = \frac{1}{|R_k|} \sum_{i \in R_k} a_i. \quad (67)$$

Pooling Layer



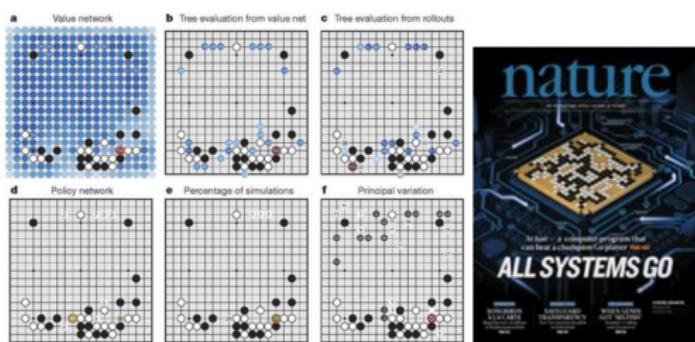
¹⁵Figure from: <http://cs231n.github.io/convolutional-networks/>

Large Scale Visual Recognition Challenge

2010-2015

2012 Teams	%error	2013 Teams	%error	2014 Teams	%error
Supervision (Toronto)	15.3	Clarifai (NYU spinoff)	11.7	GoogLeNet	6.6
ISI (Tokyo)	26.1	NUS (singapore)	12.9	VGG (Oxford)	7.3
VGG (Oxford)	26.9	Zeiler-Fergus (NYU)	13.5	MSRA	8.0
XRCE/INRIA	27.0	A. Howard	13.5	A. Howard	8.1
UvA (Amsterdam)	29.6	OverFeat (NYU)	14.1	DeeperVision	9.5
INRIA/LEAR	33.4	UvA (Amsterdam)	14.2	NUS-BST	9.7
		Adobe	15.2	TTIC-ECP	10.2
		VGG (Oxford)	15.2	XYZ	11.2
		VGG (Oxford)	23.0	UvA	12.1

DeepMind's AlphaGo



¹⁵<http://cs231n.stanford.edu/>

DeepMind's AlphaGo

The input to the policy network is a $19 \times 19 \times 48$ image stack consisting of 48 feature planes. The first hidden layer zero pads the input into a 23×23 image, then convolves k filters of kernel size 5×5 with stride 1 with the input image and applies a rectifier nonlinearity. Each of the subsequent hidden layers 2 to 12 zero pads the respective previous hidden layer into a 21×21 image, then convolves k filters of kernel size 3×3 with stride 1, again followed by a rectifier nonlinearity. The final layer convolves 1 filter of kernel size 1×1 with stride 1, with a different bias for each position, and applies a softmax function. The match version of AlphaGo used $k = 192$ filters; Fig. 2b and Extended Data Table 3 additionally show the results of training with $k = 128, 256$ and 384 filters.

policy network:

[$19 \times 19 \times 48$] Input

CONV1: 192 5×5 filters , stride 1, pad 2 => [$19 \times 19 \times 192$]

CONV2..12: 192 3×3 filters, stride 1, pad 1 => [$19 \times 19 \times 192$]

CONV: 1 1×1 filter, stride 1, pad 0 => [19×19] (*probability map of promising moves*)

¹⁵<http://cs231n.stanford.edu/>

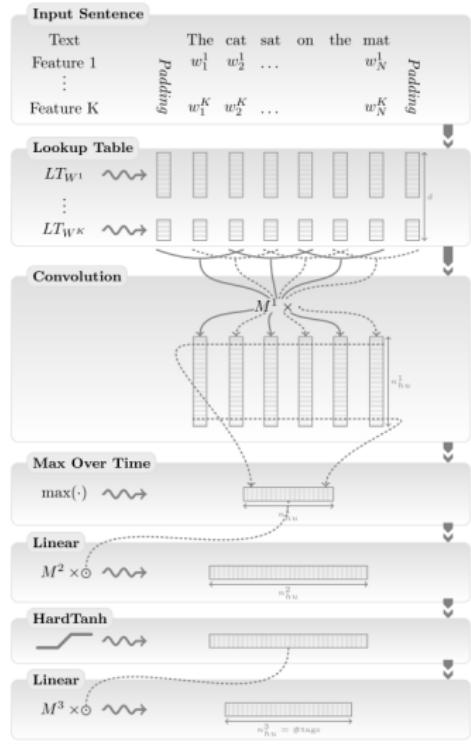
CNN for Sentence Modeling

Input: A sentence of length n ,

After Lookup layer, $X = [x_1, x_2, \dots, x_n] \in \mathcal{R}^{d \times n}$

- variable-length input
- convolution
- pooling

CNN for Sentence Modeling¹⁶

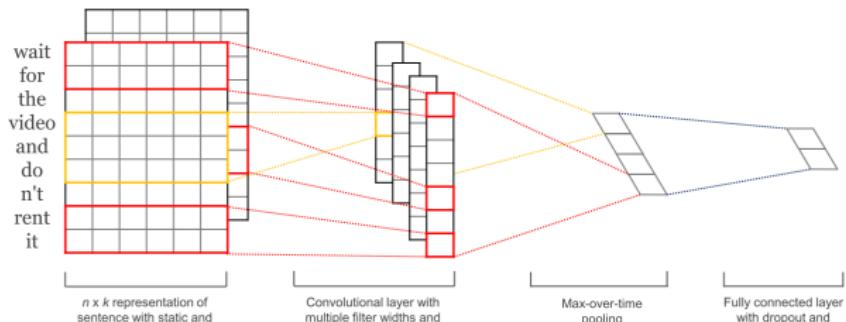


Key steps

- convolution
 $z_{t:t+m-1} = x_t \oplus x_{t+1} \oplus \dots \oplus x_{t+m-1} \in \mathcal{R}^{dm}$
- matrix-vector operation
 $x_t^I = f(W^I z_{t:t+m-1} + b^I)$
- Pooling (max over time)
 $x_i^I = \max_t x_{i,t}^{I-1}$

¹⁶ Collobert et al., "Natural language processing (almost) from scratch".

CNN for Sentence Modeling¹⁷



Key steps

- convolution

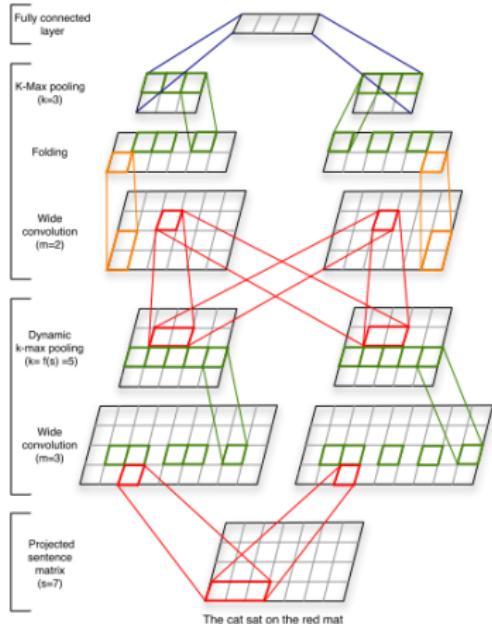
$$\mathbf{z}_{t:t+m-1} = \mathbf{x}_t \oplus \mathbf{x}_{t+1} \oplus \dots \oplus \mathbf{x}_{t+m-1} \in \mathcal{R}^{dm}$$
- vector-vector operation

$$\mathbf{x}_t^I = f(\mathbf{w}^I \mathbf{z}_{t:t+m-1} + b^I)$$
- multiple filters / multiple channels
- pooling (max over time)

$$\mathbf{x}_i^I = \max_t \mathbf{x}_{i,t}^{I-1}$$

¹⁷ Y. Kim. "Convolutional neural networks for sentence classification". In: [arXiv preprint arXiv:1408.5882](https://arxiv.org/abs/1408.5882) (2014).

Dynamic CNN for Sentence Modeling¹⁸



Key steps

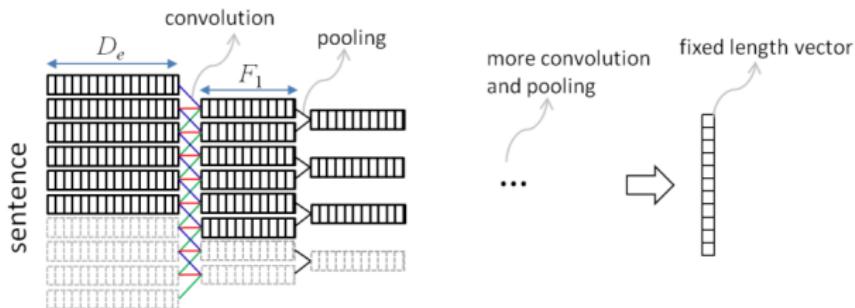
- one-dimensional convolution

$$\mathbf{x}_{i,t}^l = f(\mathbf{w}_i^l \mathbf{x}_{i,t:t+m-1} + b_i^l)$$
- k -max pooling (max over time)

$$k^l = \max(k^{top}, \frac{|L|-l}{L} n)$$
- (optional) folding: sums every two rows in a feature map component-wise.
- multiple filters / multiple channels

¹⁸ N. Kalchbrenner, E. Grefenstette, and P. Blunsom. "A Convolutional Neural Network for Modelling Sentences". In: Proceedings of ACL. 2014.

CNN for Sentence Modeling¹⁹



Key steps

- convolution
- $$\mathbf{z}_{t:t+m-1} = \mathbf{x}_t \oplus \mathbf{x}_{t+1} \oplus \dots \oplus \mathbf{x}_{t+m-1} \in \mathcal{R}^{dm}$$
- matrix-vector operation
- $$\mathbf{x}_t^I = f(\mathbf{W}^I \mathbf{z}_{t:t+m-1}^{I-1} + b^I)$$
- binary max pooling (over time)
- $$\mathbf{x}_{i,t}^I = \max(\mathbf{x}_{i,2t-1}^{I-1}, \mathbf{x}_{i,2t}^{I-1})$$

¹⁹ B. Hu et al. "Convolutional neural network architectures for matching natural language sentences". In: Advances in Neural Information Processing Systems. 2014.

Table of Contents

1 Basic Concepts

- Artificial Intelligence
- Machine Learning
- Deep Learning
- Implementation

2 Neural Models for Representation Learning

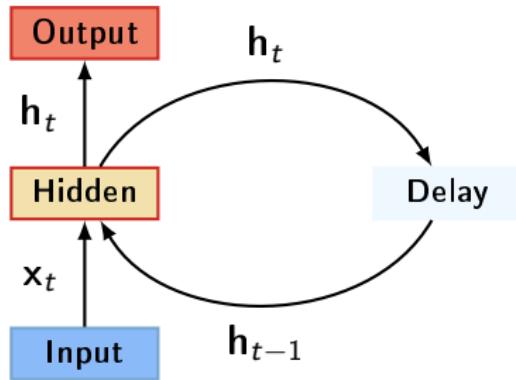
- General Architecture
- Convolutional Neural Network
- Recurrent Neural Network**
- Recursive Neural Network
- Attention Model

3 Applications

- Text Generation
- Question Answering
- Machine Translation
- Text Matching

4 Challenges & Open Problems

Recurrent Neural Network (RNN)



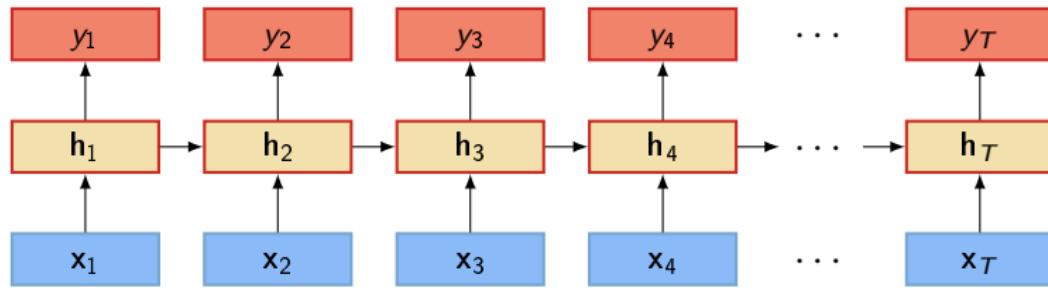
The RNN has recurrent hidden states whose output at each time is dependent on that of the previous time. More formally, given a sequence $\mathbf{x}^{(1:n)} = (\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(t)}, \dots, \mathbf{x}^{(n)})$, the RNN updates its recurrent hidden state $\mathbf{h}^{(t)}$ by

$$\mathbf{h}_t = \begin{cases} 0 & t = 0 \\ f(\mathbf{h}_{t-1}, \mathbf{x}_t) & \text{otherwise} \end{cases} \quad (68)$$

Simple Recurrent Network²⁰

$$\mathbf{h}_t = f(\mathbf{U}\mathbf{h}_{t-1} + \mathbf{W}\mathbf{x}_t + \mathbf{b}), \quad (69)$$

where f is non-linear function.



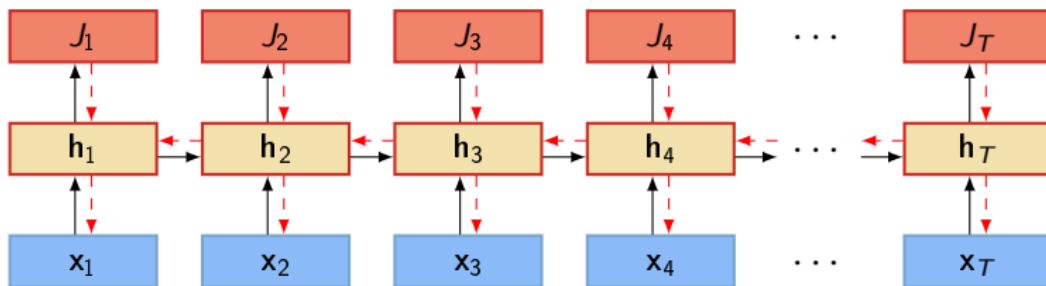
²⁰ J. L. Elman, "Finding structure in time". In: Cognitive science 2 (1990).

Backpropagation Through Time, BPTT

Loss at time t is \mathcal{J}_t , the whole loss is $\mathcal{J} = \sum_{t=1}^T \mathcal{J}_t$.
 The gradient of \mathcal{J} is

$$\frac{\partial \mathcal{J}}{\partial U} = \sum_{t=1}^T \frac{\partial \mathcal{J}_t}{\partial U} \quad (70)$$

$$= \sum_{t=1}^T \frac{\partial \mathbf{h}_t}{\partial U} \frac{\partial \mathcal{J}_t}{\partial \mathbf{h}_t}, \quad (71)$$



Gradient of RNN

$$\frac{\partial \mathcal{J}}{\partial U} = \sum_{t=1}^T \sum_{k=1}^t \frac{\partial \mathbf{h}_k}{\partial U} \frac{\partial \mathbf{h}_t}{\partial \mathbf{h}_k} \frac{\partial \mathbf{y}_t}{\partial \mathbf{h}_t} \frac{\partial \mathcal{J}_t}{\partial \mathbf{y}_t}, \quad (72)$$

$$\frac{\partial \mathbf{h}_t}{\partial \mathbf{h}_k} = \prod_{i=k+1}^t \frac{\partial \mathbf{h}_i}{\partial \mathbf{h}_{i-1}} \quad (73)$$

$$= \prod_{i=k+1}^t U^T \mathbf{diag}[f'(h_{i-1})]. \quad (74)$$

$$\frac{\partial \mathcal{J}}{\partial U} = \sum_{t=1}^T \sum_{k=1}^t \frac{\partial \mathbf{h}_k}{\partial U} \left(\prod_{i=k+1}^t U^T \mathbf{diag}(f'(h_{i-1})) \right) \frac{\partial \mathbf{y}_t}{\partial \mathbf{h}_t} \frac{\partial \mathcal{J}_t}{\partial \mathbf{y}_t}. \quad (75)$$

Long-Term Dependencies

Define $\gamma = \|U^T \text{diag}(f'(h_{i-1}))\|$,

- Exploding Gradient Problem: When $\gamma > 1$ and $t - k \rightarrow \infty$,
 $\gamma^{t-k} \rightarrow \infty$.
- Vanishing Gradient Problem: When $\gamma < 1$ and $t - k \rightarrow \infty$, $\gamma^{t-k} \rightarrow 0$.

There are various ways to solve **Long-Term Dependency** problem.

Long Short Term Memory Neural Network (LSTM)²¹

The core of the LSTM model is a memory cell \mathbf{c} encoding memory at every time step of what inputs have been observed up to this step.

The behavior of the cell is controlled by three “gates”:

- input gate i
- output gate o
- forget gate f

$$i_t = \sigma(W_i \mathbf{x}_t + U_i \mathbf{h}_{t-1} + V_i \mathbf{c}_{t-1}), \quad (76)$$

$$f_t = \sigma(W_f \mathbf{x}_t + U_f \mathbf{h}_{t-1} + V_f \mathbf{c}_{t-1}), \quad (77)$$

$$o_t = \sigma(W_o \mathbf{x}_t + U_o \mathbf{h}_{t-1} + V_o \mathbf{c}_t), \quad (78)$$

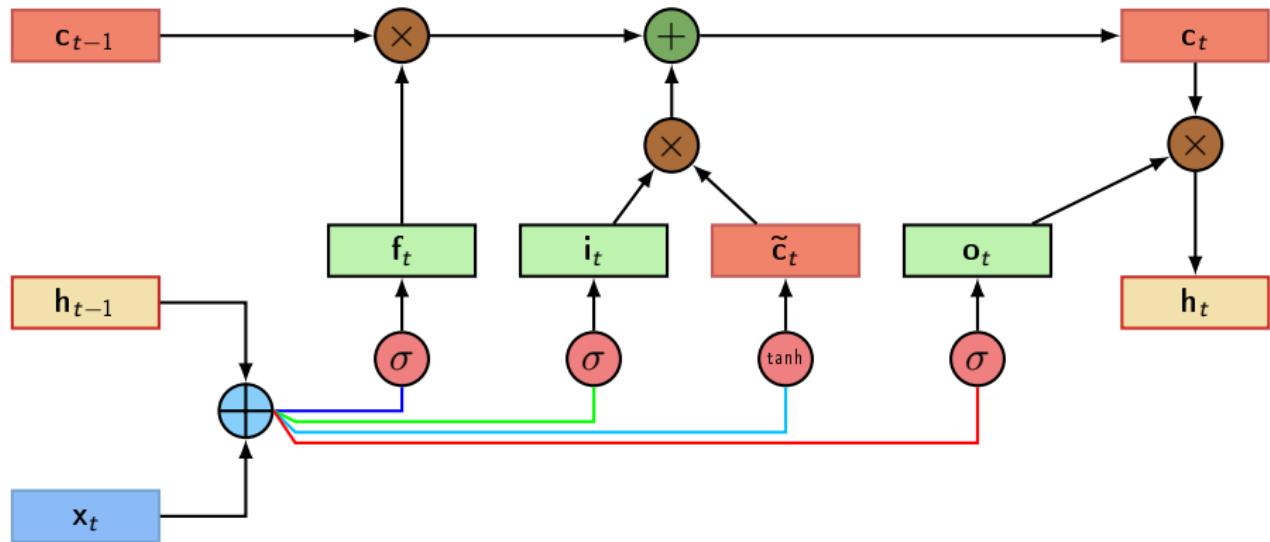
$$\tilde{\mathbf{c}}_t = \tanh(W_c \mathbf{x}_t + U_c \mathbf{h}_{t-1}), \quad (79)$$

$$\mathbf{c}_t = f_t \odot \mathbf{c}_{t-1} + i_t \odot \tilde{\mathbf{c}}_t, \quad (80)$$

$$\mathbf{h}_t = o_t \odot \tanh(\mathbf{c}_t), \quad (81)$$

²¹ S. Hochreiter and J. Schmidhuber. “Long short-term memory”. In: *Neural computation* 8 (1997).

LSTM Architecture



Gated Recurrent Unit, GRU²²

Two gates: update gate z and reset gate r .

$$r_t = \sigma(\mathbf{W}_r \mathbf{x}_t + \mathbf{U}_r \mathbf{h}_{t-1}) \quad (82)$$

$$z_t = \sigma(\mathbf{W}_z \mathbf{x}_t + \mathbf{U}_z \mathbf{h}_{t-1}) \quad (83)$$

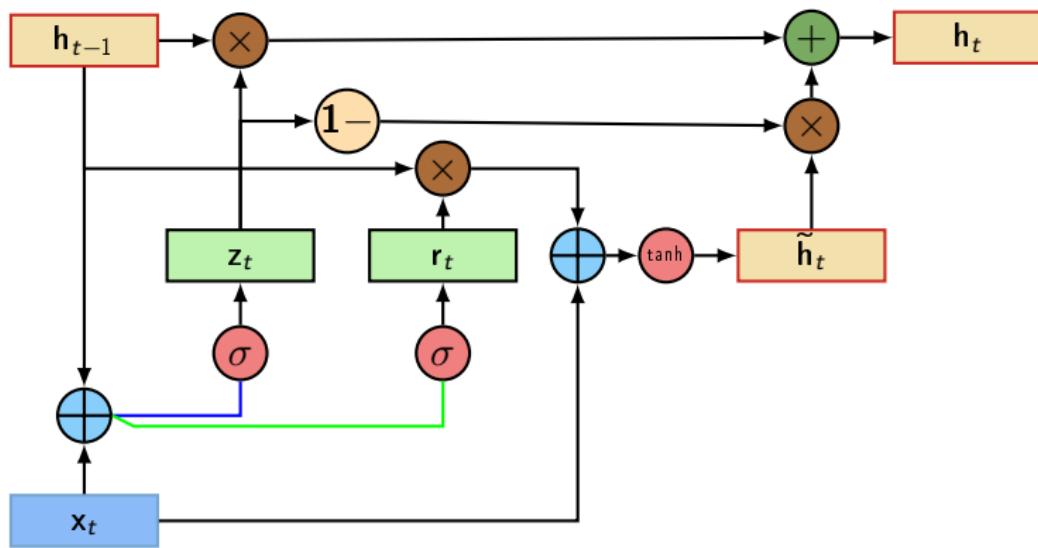
$$\tilde{\mathbf{h}}_t = \tanh(\mathbf{W}_c \mathbf{x}_t + \mathbf{U}(r_t \odot \mathbf{h}_{t-1})), \quad (84)$$

$$\mathbf{h}_t = z_t \odot \mathbf{h}_{t-1} + (1 - z_t) \odot \tilde{\mathbf{h}}_t, \quad (85)$$

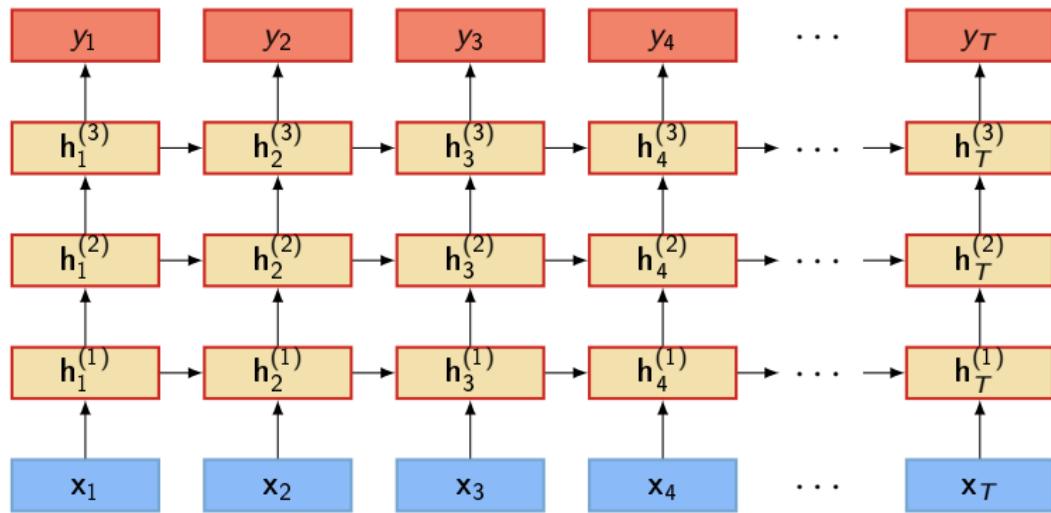
$$(86)$$

²² K. Cho et al. "Learning phrase representations using rnn encoder-decoder for statistical machine translation". In: [arXiv preprint arXiv:1406.1078](#) (2014); J. Chung et al. "Empirical evaluation of gated recurrent neural networks on sequence modeling". In: [arXiv preprint arXiv:1412.3555](#) (2014).

GRU Architecture



Stacked RNN

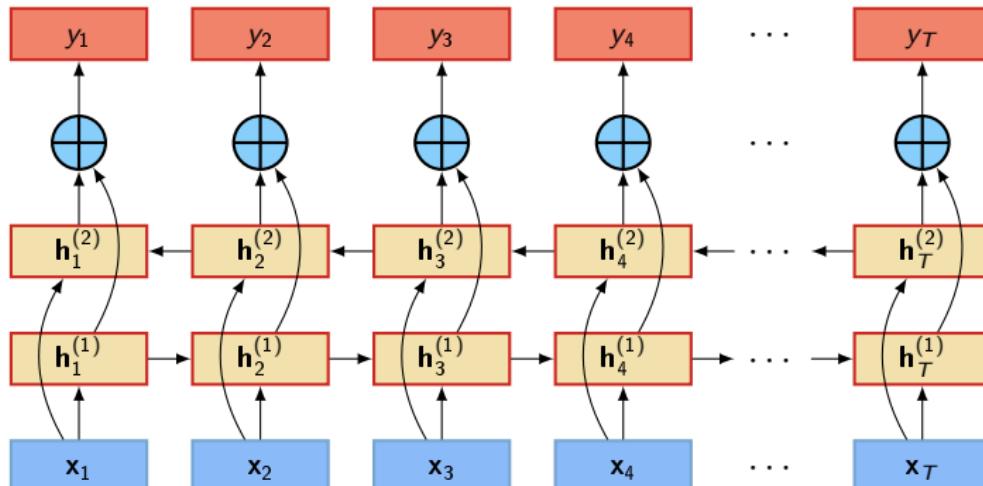


Bidirectional RNN

$$\mathbf{h}_t^{(1)} = f(\mathbf{U}^{(1)} \mathbf{h}_{t-1}^{(1)} + \mathbf{W}^{(1)} \mathbf{x}_t + \mathbf{b}^{(1)}), \quad (87)$$

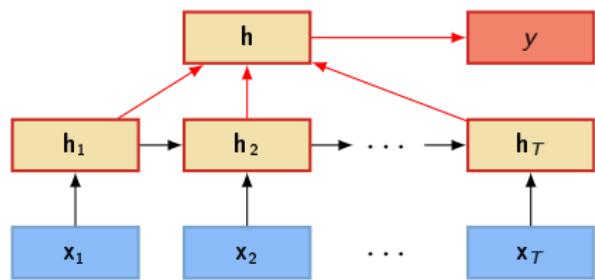
$$\mathbf{h}_t^{(2)} = f(\mathbf{U}^{(2)} \mathbf{h}_{t+1}^{(2)} + \mathbf{W}^{(2)} \mathbf{x}_t + \mathbf{b}^{(2)}), \quad (88)$$

$$\mathbf{h}_t = \mathbf{h}_t^{(1)} \oplus \mathbf{h}_t^{(2)}. \quad (89)$$

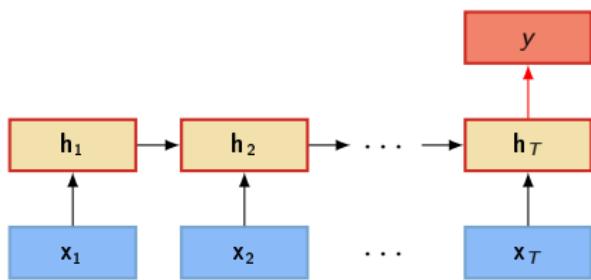


Application of RNN: Sequence to Category

- Text Classification
- Sentiment Classification



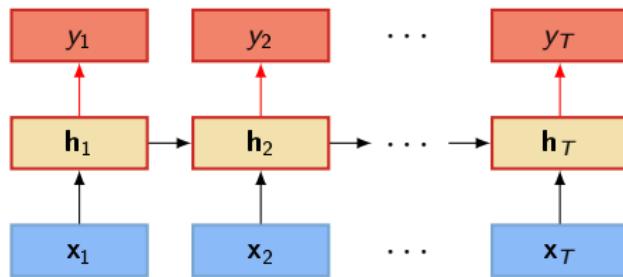
(c) Mean



(d) Last

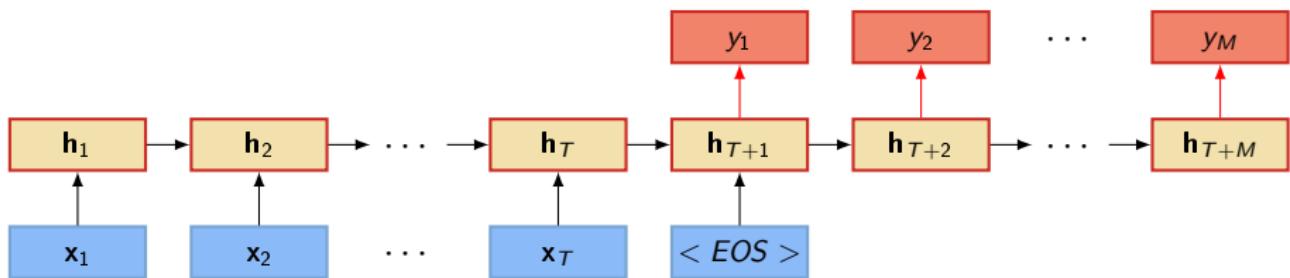
Application of RNN: Synchronous Sequence to Sequence

- Sequence Labeling, such as Chinese word segmentation, POS tagging

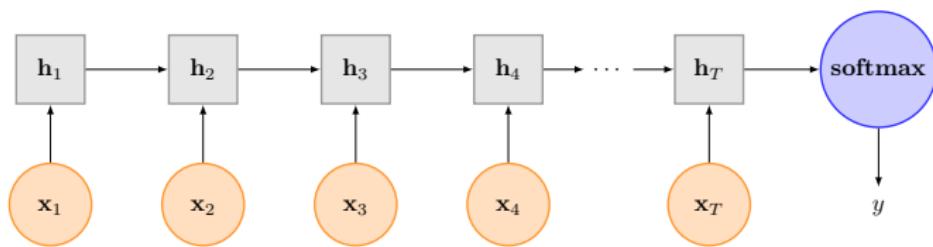


Application of RNN: Asynchronous Sequence to Sequence

- Machine Translation

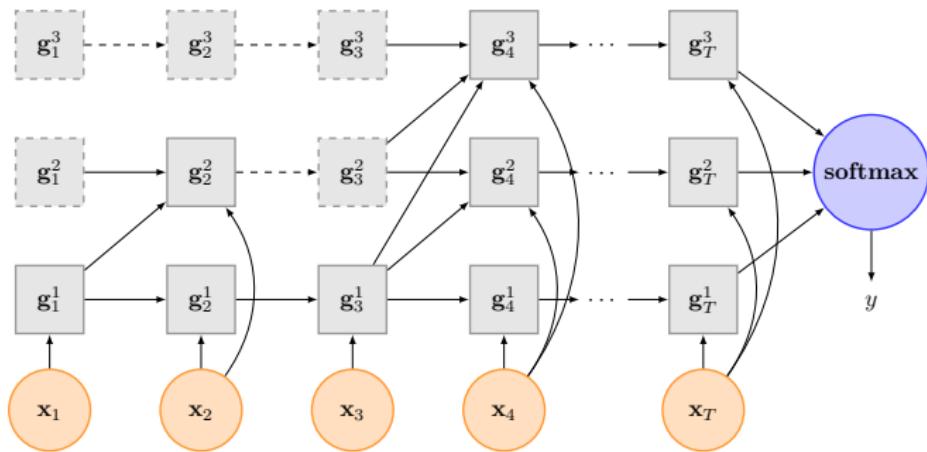


Unfolded LSTM for Text Classification



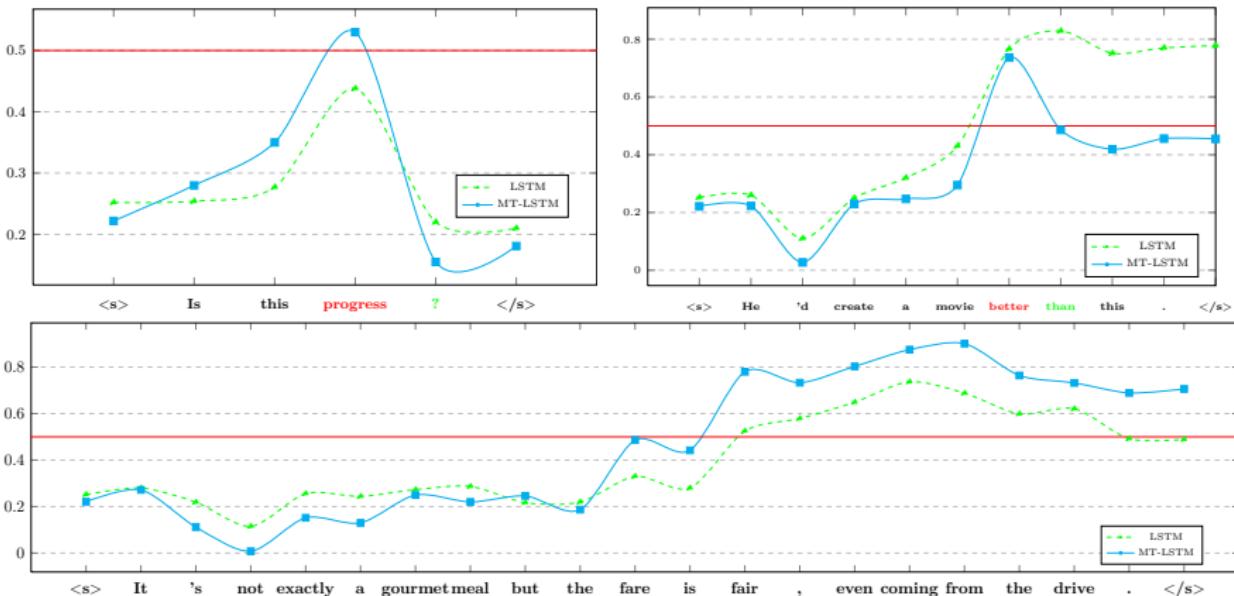
Drawback: long-term dependencies need to be transmitted one-by-one along the sequence.

Unfolded Multi-Timescale LSTM²³

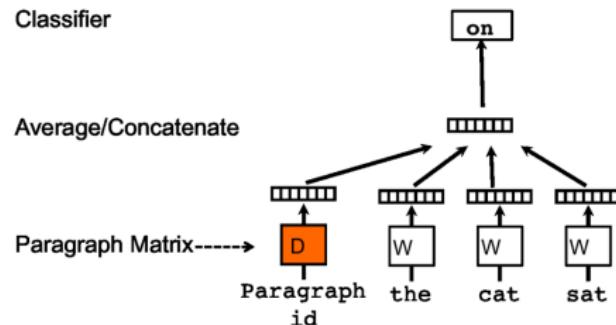


²³ P. Liu et al. "Multi-Timescale Long Short-Term Memory Neural Network for Modelling Sentences and Documents". In: *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. 2015.

LSTM for Sentiment Analysis



Paragraph Vector²⁴



²⁴ Q. V. Le and T. Mikolov. "Distributed representations of sentences and documents". In: arXiv preprint arXiv:1405.4053 (2014).

Memory Mechanism

What differences among the various models from memory view?

	Short-term	Long short-term	Global	External
SRN	Yes	No	No	No
LSTM/GRU	Yes	Yes	Maybe	No
PV	Yes	Yes	Yes	No
NTM/DMN	Yes	Yes	Maybe	Yes

Table of Contents

1 Basic Concepts

- Artificial Intelligence
- Machine Learning
- Deep Learning
- Implementation

2 Neural Models for Representation Learning

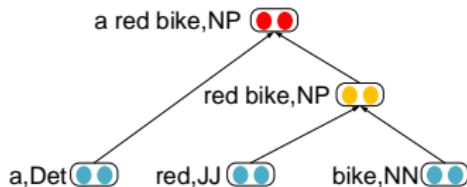
- General Architecture
- Convolutional Neural Network
- Recurrent Neural Network
- Recursive Neural Network**
- Attention Model

3 Applications

- Text Generation
- Question Answering
- Machine Translation
- Text Matching

4 Challenges & Open Problems

Recursive Neural Network (RecNN)²⁵



Topological models compose the sentence representation following a given topological structure over the words.

Given a labeled binary parse tree, $((p_2 \rightarrow ap_1), (p_1 \rightarrow bc))$, the node representations are computed by

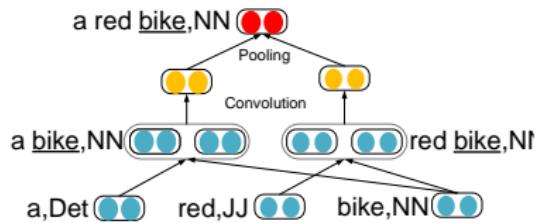
$$\mathbf{p}_1 = f(\mathbf{W} \begin{bmatrix} \mathbf{b} \\ \mathbf{c} \end{bmatrix}),$$

$$\mathbf{p}_2 = f(\mathbf{W} \begin{bmatrix} \mathbf{a} \\ \mathbf{p}_1 \end{bmatrix}).$$

²⁵ R. Socher et al. "Parsing with compositional vector grammars". In: Proceedings of ACL. 2013.

Recursive Convolutional Neural Network²⁶

Recursive neural network can only process the binary combination and is not suitable for dependency parsing.

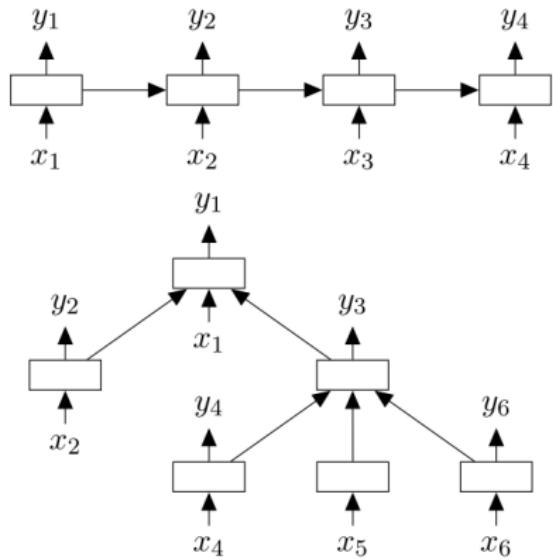


- introducing the convolution and pooling layers;
- modeling the complicated interactions of the head word and its children.

²⁶ C. Zhu et al. "A Re-Ranking Model For Dependency Parser With Recursive Convolutional Neural Network". In: Proceedings of Annual Meeting of the Association for Computational Linguistics. 2015.

Tree-Structured LSTMs²⁷

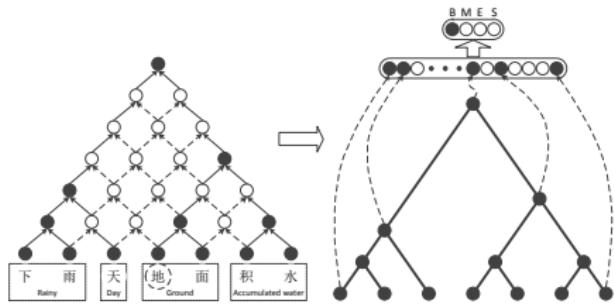
Natural language exhibits syntactic properties that would naturally combine words to phrases.



- Child-Sum Tree-LSTMs
- N -ary Tree-LSTMs

²⁷ K. S. Tai, R. Socher, and C. D. Manning. "Improved semantic representations from tree-structured long short-term memory networks". In: [arXiv preprint arXiv:1503.00075](https://arxiv.org/abs/1503.00075) (2015).

Gated Recursive Neural Network²⁸



- DAG based Recursive Neural Network
- Gating mechanism

An relative complicated solution
 GRNN models the complicated combinations of the features, which selects and preserves the useful combinations via reset and update gates.

²⁸ X. Chen et al. "Gated Recursive Neural Network For Chinese Word Segmentation". In: *Proceedings of Annual Meeting of the Association for Computational Linguistics*. 2015; X. Chen et al. "Sentence Modeling with Gated Recurrent Neural Network". In: *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. 2015.

Table of Contents

1 Basic Concepts

- Artificial Intelligence
- Machine Learning
- Deep Learning
- Implementation

2 Neural Models for Representation Learning

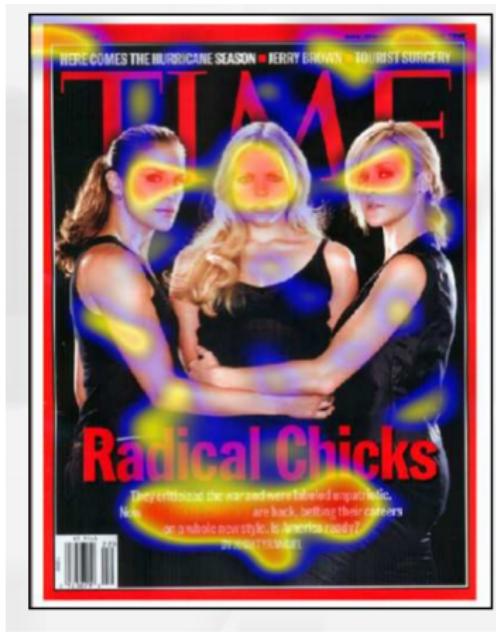
- General Architecture
- Convolutional Neural Network
- Recurrent Neural Network
- Recursive Neural Network
- **Attention Model**

3 Applications

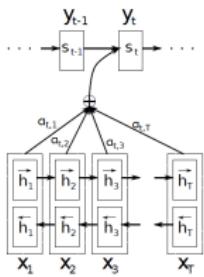
- Text Generation
- Question Answering
- Machine Translation
- Text Matching

4 Challenges & Open Problems

Attention



Attention Model²⁹



- ① The context vector c_i is computed as a weighted sum of h_i :

$$c_i = \sum_{j=1}^T \alpha_{ij} h_j$$

- ② The weight α_{ij} is computed by

$$\alpha_{ij} = \text{softmax}(\mathbf{v}^T \tanh(\mathbf{W}s_{i-1} + \mathbf{U}h_j))$$

²⁹ D. Bahdanau, K. Cho, and Y. Bengio. "Neural Machine Translation by Jointly Learning to Align and Translate". In: ArXiv e-prints (Sept. 2014). arXiv: 1409.0473 [cs.CL].

Table of Contents

1 Basic Concepts

- Artificial Intelligence
- Machine Learning
- Deep Learning
- Implementation

2 Neural Models for Representation Learning

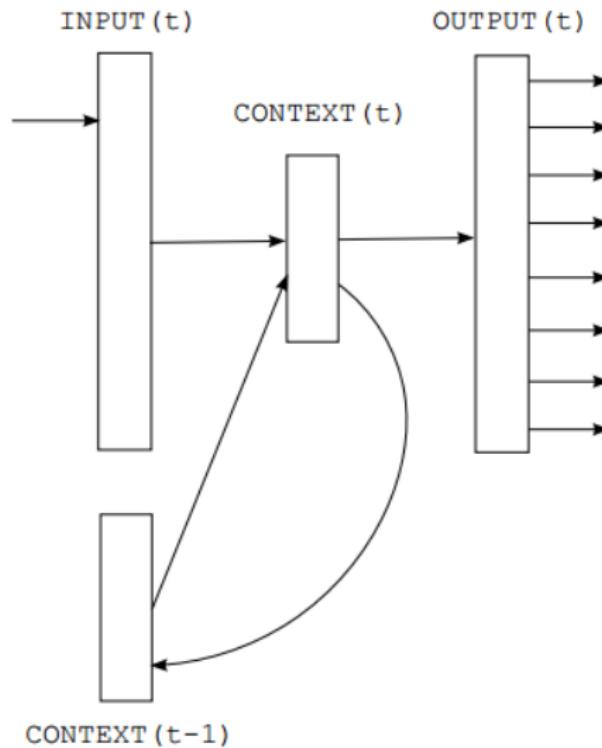
- General Architecture
- Convolutional Neural Network
- Recurrent Neural Network
- Recursive Neural Network
- Attention Model

3 Applications

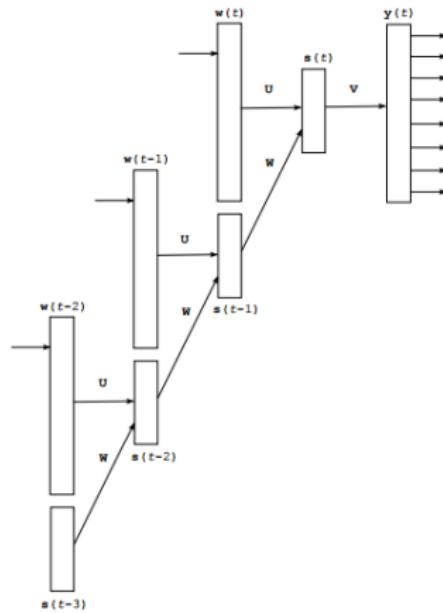
- Text Generation
- Question Answering
- Machine Translation
- Text Matching

4 Challenges & Open Problems

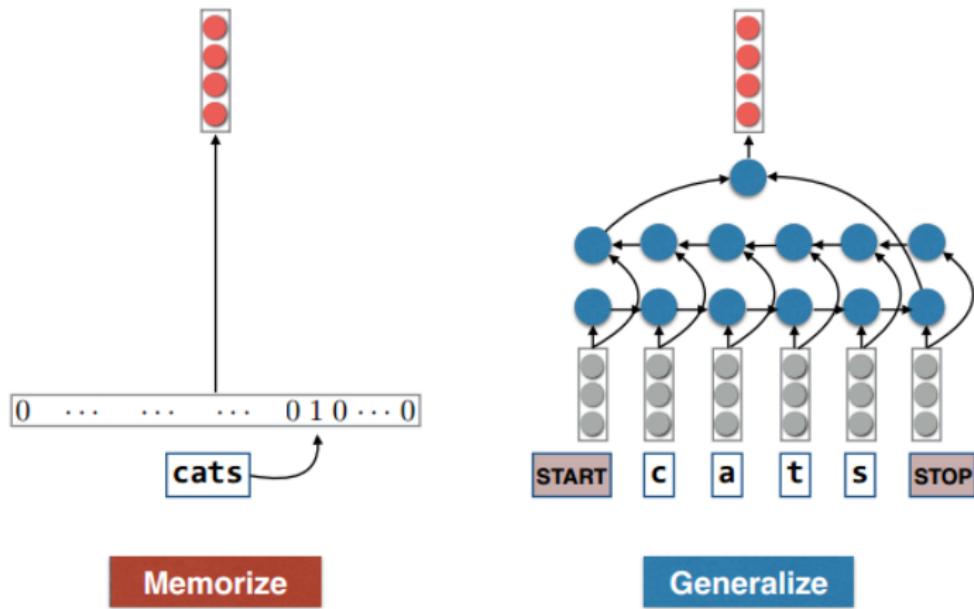
RNN-LM



Unfolded RNN-LM



Word VS. Character Modeling



²⁹Chris Dyer, Should Model Architecture Reflect Linguistic Structure?

Composition of Lyric

汪峰作词机³⁰

我在这里中的夜里
就像一场是一种生命的意
就像我的生活变得在我一样
可我们这是一个知道
我只是天你会怎吗
可我们这是我们的是不要为你
我们想这有一种生活的时候

³⁰<https://github.com/phunterlau/wangfeng-rnn>

Poetry Generation³¹

<p>白鹭窥鱼立， Egrets stood, peeping fishes. 青山照水开。 Water was still, reflecting mountains. 夜来风不动， The wind went down by nightfall, 明月见楼台。 as the moon came up by the tower.</p>	<p>满怀风月一枝春， Budding branches are full of romance. 未见梅花亦可人。 Plum blossoms are invisible but adorable. 不为东风无此客， With the east wind comes Spring. 世间何处是前身。 Where on earth do I come from?</p>
--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

³¹ X. Zhang and M. Lapata. "Chinese Poetry Generation with Recurrent Neural Networks." In: EMNLP, 2014, pp. 670–680.

Table of Contents

1 Basic Concepts

- Artificial Intelligence
- Machine Learning
- Deep Learning
- Implementation

2 Neural Models for Representation Learning

- General Architecture
- Convolutional Neural Network
- Recurrent Neural Network
- Recursive Neural Network
- Attention Model

3 Applications

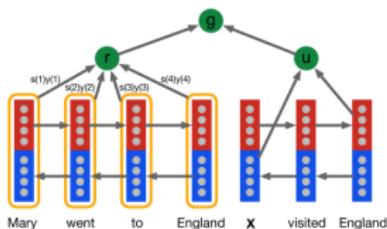
- Text Generation
- Question Answering**
- Machine Translation
- Text Matching

4 Challenges & Open Problems

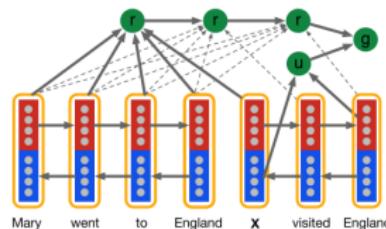
Question Answering

- | | |
|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <p>I: Jane went to the hallway.</p> <p>I: Mary walked to the bathroom.</p> <p>I: Sandra went to the garden.</p> <p>I: Daniel went back to the garden.</p> <p>I: Sandra took the milk there.</p> <p>Q: Where is the milk?</p> <p>A: garden</p> <p>I: Everybody is happy.</p> <p>Q: What's the sentiment?</p> <p>A: positive</p> <p>Q: What are the POS tags?</p> <p>A: NN VBZ JJ .</p> | <p>I: Peter's sister is called Isabelle.</p> <p>Q: What are the mentions?</p> <p>A: [[Peter] 's sister] is called [Isabelle].</p> <p>I: Peter's sister is called [Isabelle].</p> <p>Q: Is it coreferent with: [Peter's sister] is called Isabelle?</p> <p>A: yes</p> <p>Q: Is it coreferent with: [Peter] 's sister is called Isabelle?</p> <p>A: no</p> <p>I: The answer is far from obvious.</p> <p>Q: In French?</p> <p>A: La réponse est loin d'être évidente.</p> |
|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

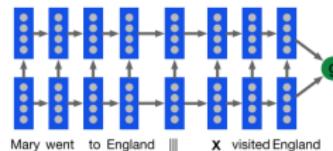
LSTM³²



(a) Attentive Reader.



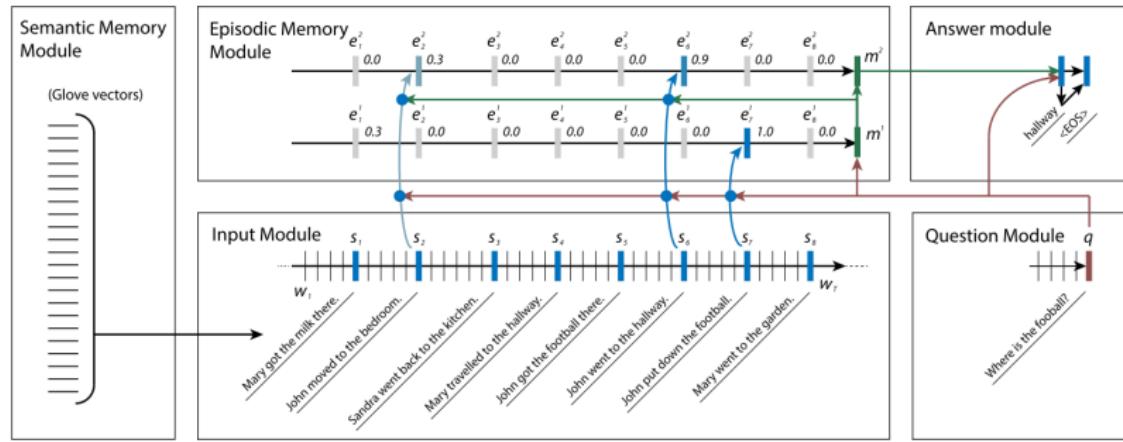
(b) Impatient Reader.



(c) A two layer Deep LSTM Reader with the question encoded following the document.

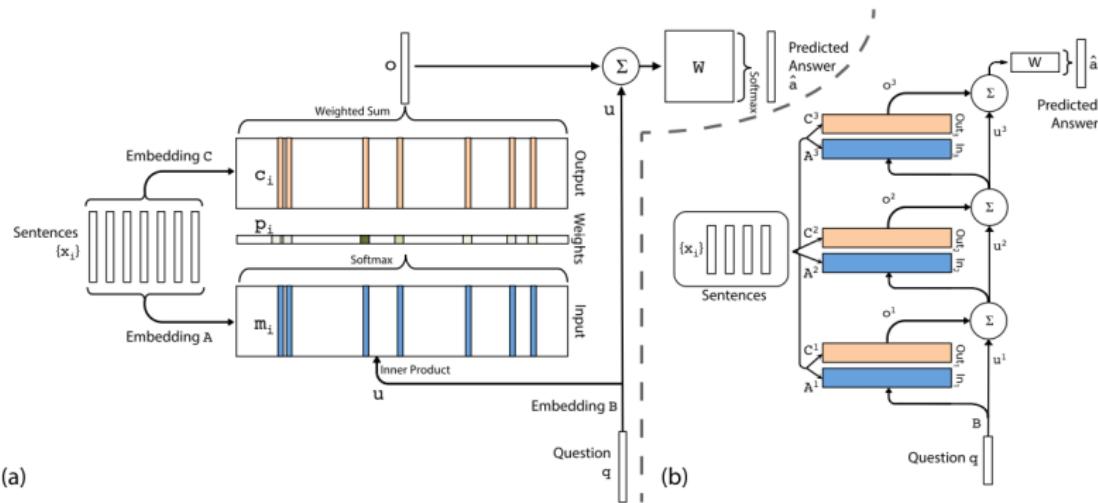
³² K. M. Hermann et al. "Teaching machines to read and comprehend". In: Advances in Neural Information Processing Systems. 2015, pp. 1684–1692.

Dynamic Memory Networks³³



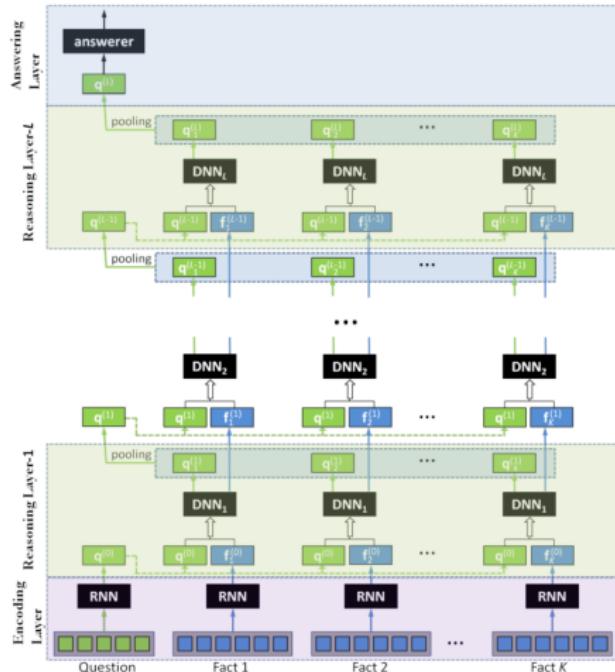
³³ A. Kumar et al. "Ask me anything: Dynamic memory networks for natural language processing". In: arXiv preprint arXiv:1506.07285 (2015).

Memory Networks³⁴



³⁴ S. Sukhbaatar, J. Weston, R. Fergus, et al. “End-to-end memory networks”. In: Advances in Neural Information Processing Systems. 2015, pp. 2431–2439.

Neural Reasoner³⁵



³⁵ B. Peng et al. "Towards Neural Network-based Reasoning". In: [arXiv preprint arXiv:1508.05508](https://arxiv.org/abs/1508.05508) (2015).

Table of Contents

1 Basic Concepts

- Artificial Intelligence
- Machine Learning
- Deep Learning
- Implementation

2 Neural Models for Representation Learning

- General Architecture
- Convolutional Neural Network
- Recurrent Neural Network
- Recursive Neural Network
- Attention Model

3 Applications

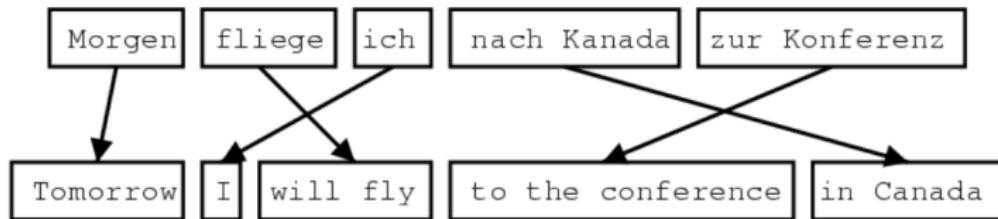
- Text Generation
- Question Answering
- Machine Translation**
- Text Matching

4 Challenges & Open Problems

Current Statistical Machine Translation

Neural machine translation is a recently proposed framework for machine translation based purely on neural networks.

- Source Language: f
- Target Language: e
- Model: $\hat{e} = \arg \max_e p(e|f) = \arg \max_e p(f|e)p(e)$
 - $p(f|e)$: translation model
 - $p(e)$: language model

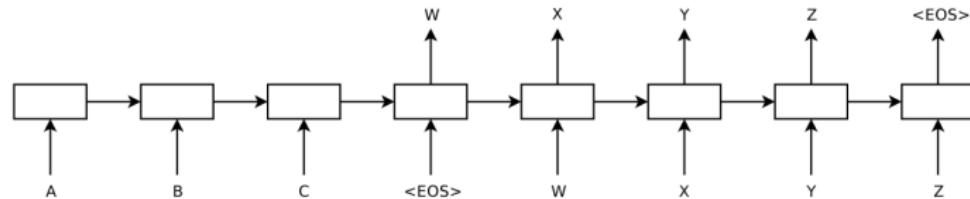


³⁵ Image Source: <http://www.statmt.org/wpt05/mt-shared-task/>

Sequence to Sequence Machine Translation³⁶

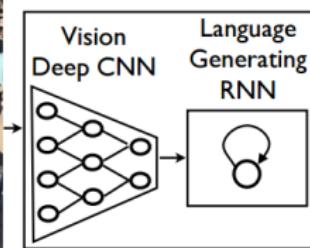
Neural machine translation is a recently proposed framework for machine translation based purely on neural networks.

- One RNN for encoding;
- Another RNN for decoding.



³⁶ I. Sutskever, O. Vinyals, and Q. V. Le. "Sequence to sequence learning with neural networks". In: Advances in Neural Information Processing Systems. 2014, pp. 3104–3112.

Image Caption³⁷³⁸³⁹



A group of people shopping at an outdoor market.

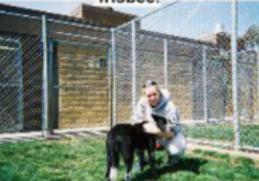
There are many vegetables at the fruit stand.

³⁷ A. Karpathy and L. Fei-Fei. "Deep visual-semantic alignments for generating image descriptions". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2015, pp. 3128–3137.

³⁸ O. Vinyals et al. "Show and tell: A neural image caption generator". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2015, pp. 3156–3164.

³⁹ K. Xu et al. "Show, attend and tell: Neural image caption generation with visual attention". In: *arXiv preprint arXiv:1502.03044* (2015).

Image Caption

A person riding a motorcycle on a dirt road.	Two dogs play in the grass.	A skateboarder does a trick on a ramp.	A dog is jumping to catch a frisbee.
			
A group of young people playing a game of frisbee.	Two hockey players are fighting over the puck.	A little girl in a pink hat is blowing bubbles.	A refrigerator filled with lots of food and drinks.
			
A herd of elephants walking across a dry grass field.	A close up of a cat laying on a couch.	A red motorcycle parked on the side of the road.	A yellow school bus parked in a parking lot.
			

Describes without errors

Describes with minor errors

Somewhat related to the image

Unrelated to the image

Figure 5. A selection of evaluation results, grouped by human rating.

Table of Contents

1 Basic Concepts

- Artificial Intelligence
- Machine Learning
- Deep Learning
- Implementation

2 Neural Models for Representation Learning

- General Architecture
- Convolutional Neural Network
- Recurrent Neural Network
- Recursive Neural Network
- Attention Model

3 Applications

- Text Generation
- Question Answering
- Machine Translation
- Text Matching**

4 Challenges & Open Problems

Text Matching

Among many natural language processing (NLP) tasks, such as text classification, question answering and machine translation, a common problem is modelling the relevance/similarity of a pair of texts, which is also called **text semantic matching**.

Three types of interaction models:

- Weak interaction Models
- Semi-interaction Models
- Strong Interaction Models

Weak interaction Models

Some early works focus on sentence level interactions, such as ARC-I⁴⁰, CNTN⁴¹ and so on. These models first encode two sequences into continuous dense vectors by separated neural models, and then compute the matching score based on sentence encoding.

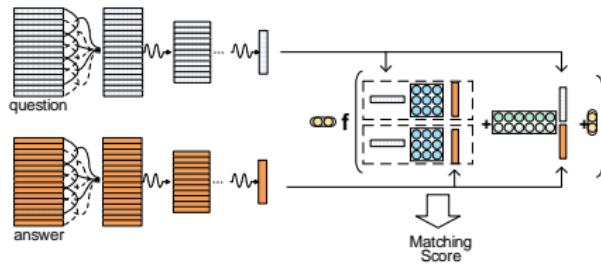


Figure: Convolutional Neural Tensor Network

⁴⁰ Hu et al., "Convolutional neural network architectures for matching natural language sentences".

⁴¹ X. Qiu and X. Huang, "Convolutional Neural Tensor Network Architecture for Community-based Question Answering". In: Proceedings of International Joint Conference on Artificial Intelligence. 2015.

Semi-interaction Models

Another kind of models use soft attention mechanism to obtain the representation of one sentence by depending on representation of another sentence, such as ABCNN⁴², Attention LSTM⁴³. These models can alleviate the weak interaction problem to some extent.

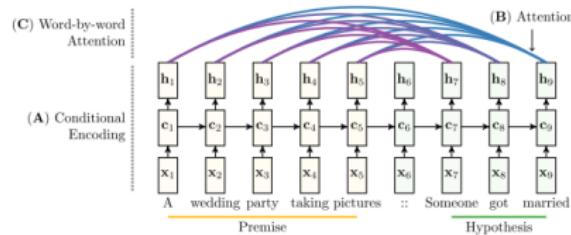


Figure: Attention LSTM⁴⁴

⁴² W. Yin et al. "ABCNN: Attention-Based Convolutional Neural Network for Modeling Sentence Pairs". In: [arXiv preprint arXiv:1512.05193 \(2015\)](https://arxiv.org/abs/1512.05193).

⁴³ Hermann et al., "Teaching machines to read and comprehend".

⁴⁴ T. Rocktäschel et al. "Reasoning about Entailment with Neural Attention". In: [arXiv preprint arXiv:1509.06664 \(2015\)](https://arxiv.org/abs/1509.06664).

Strong Interaction Models

The models build the interaction at different granularity (word, phrase and sentence level), such as ARC-II⁴⁵, MV-LSTM⁴⁶, coupled-LSTMs⁴⁷. The final matching score depends on these different levels of interactions.

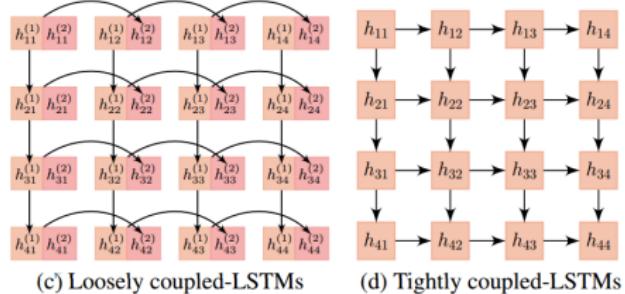


Figure: coupled-LSTMs

⁴⁵ Hu et al., "Convolutional neural network architectures for matching natural language sentences".

⁴⁶ S. Wan et al. "A Deep Architecture for Semantic Matching with Multiple Positional Sentence Representations". In: AAAI, 2016.

⁴⁷ P. Liu, X. Qiu, and X. Huang. "Modelling Interaction of Sentence Pair with coupled-LSTMs". In: arXiv preprint arXiv:1605.05573 (2016).

Conclusion of DL4NLP (just kidding)

- Long long ago: you must know **intrinsic rules** of data.
- Past ten years: you just know **effective features** of data.
- Nowadays: you just need to have **big** data.

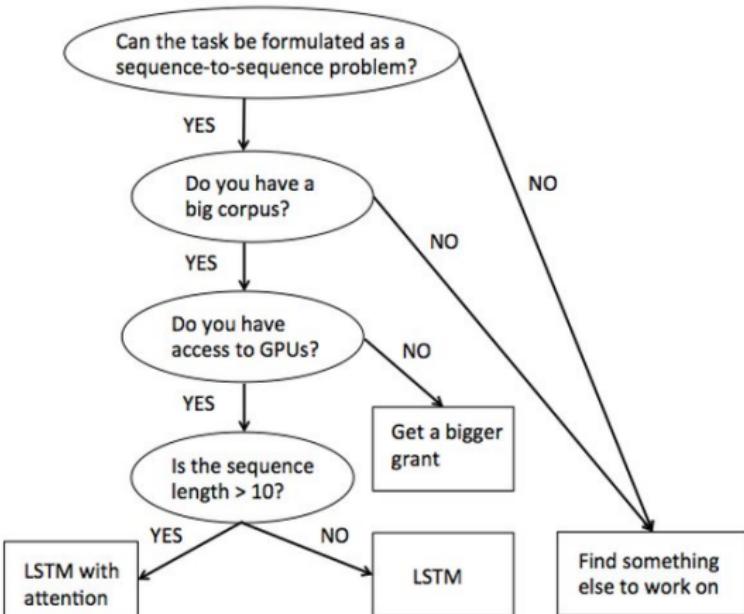
Challenges & Open Problems

- Depth of network
- Rare words
- Fundamental data structure
- Long-term dependencies
- Natural language understanding & reasoning
- Biology inspired model
- Unsupervised learning

DL4NLP from scratch

- Select a real problem
- Translate your problem to (supervised) learning problem
- Prepare your data and hardware (GPU)
- Select a library: Theano, Keras, TensorFlow
- Find an open source implementation
- Incrementally writing your own code
- Run it.

Seq2Seq⁴⁸



⁴⁸<https://twitter.com/Augenstein/status/710837374473920512>

More Information

<http://nlp.fudan.edu.cn/dl-book/>

http://vdisk.weibo.com/s/A_pmE4ilotGf

Recommended Courses

CS224d: Deep Learning for Natural Language Processing

- <http://cs224d.stanford.edu/>
- 斯坦福大学Richard Socher
- 主要讲解自然语言处理领域的各种深度学习模型

CS231n: Convolutional Neural Networks for Visual Recognition

- <http://cs231n.stanford.edu/>
- 斯坦福大学Fei-Fei Li Andrej Karpathy
- 主要讲解CNN、RNN在图像领域的应用