Representation Learning and Information Retrieval

Kuan-Yu Chen (陳冠宇)

2020/12/11 @ TR-313, NTUST

HW5

#	Team Name	Notebook	Team Members	Score
1	80847002S_羅天宏			0.59666
2	好想要不勞而獲喔			0.57917
3	.....			0.56956
4	M10915027_石成峰			0.56753
5	前面太高嘞這樣我會0分...			0.56593
6	너는나만큼(👈這首歌好聽)			0.56472
7	B10615034_黃柏翰			0.55821
8	不快樂ㄟ甘蔗man😭			0.55781
9	80847001s_顏必成			0.55121
10	Eric_S			0.53861
11	jjj_liao			0.53853
12	Enilesab			0.53813
13	可以不要出作業6嗎....			0.53807
14	QQQ			0.53794
15	M10915006_廖勗宏			0.53568
16	For Fun			0.53489
17	hongyun			0.53436
18	M10915019_顏佑庭			0.53388
19	SummerSHIT			0.53301
20	60947036S_李俊廷			0.53296

#	△pub	Team Name	Notebook	Team Members	Score
1	—	80847002S_羅天宏			0.56026
2	—	好想要不勞而獲喔			0.55615
3	▲3	너는나만큼(👈這首歌好聽)			0.55404
4	▲3	B10615034_黃柏翰			0.54435
5	—	前面太高嘞這樣我會0分...			0.53125
6	▼2	M10915027_石成峰			0.53016
7	▲9	For Fun			0.53006
8	▲1	80847001s_顏必成			0.52816
9	▲16	香蕉你個拔辣			0.52021
10	▲2	Enilesab			0.51904
11	▲4	M10915006_廖勗宏			0.51316
12	▲10	FatLee			0.51069
13	▼5	不快樂ㄟ甘蔗man😭			0.50861
14	▲7	過了baseline還是0分?			0.50770
15	▼5	Eric_S			0.50746
16	▼5	jjj_liao			0.50701
17	▼4	可以不要出作業6嗎....			0.50669
18	▲8	M10915100_郭智威			0.50643
19	▼5	QQQ			0.50593
20	▲4	壞ㄌ			0.50404

Final Project.

- The presentation order is announced

1/8	Presentation-1 M10915100 郭O威、D10907005 陳O宏、M10915034 黃O泓、M10915066 盧O恆 M10915103 邱O儒、M10915006 廖O宏、M10915046 陳O穎、M10915092 林O哲 M10915095 薛O翔、M10915050 林O瑜 、M10909112 石O安、M10909120 樊O驊 M10909211 李O妍、M10909118 蔡O真 、M10909109 陳O炫、M10909114 李O凱 80847002S 羅O宏、80847001S 顏O成、60947058S 曹O升、60947012S 王O偉 M10815111 謝O耀、M10815112 鄭O哲、M10915017 林O箴 M10915201 陳O凡、M10915097 朱O亞、M10815064 侯O林 M10915012 黃O愷、M10915036 王O崴、M10915082 張O哲 M10915045 施O宏、M10915031 鄭O謙、M10915080 羅O程 M10915019 顏O庭、M10915013 王O翔、M10815103 陳O揚 M10915028 陳O勳、M10815036 王O德、M10815048 張O銘
1/15	Presentation-2 B10615013 李O鎧、B10615024 李O宗、B10615026 溫O勳、B10615043 何O峻 B10615022 姜O昀、B10615034 黃O翰、B10615036 黃O銘、B10615056 黃O翔 B10615047 陳O緯、B10615017 林O叡、B10615023 楊O安、B10615039 高O雲 B10632026 吳O瑄、M10907505 游O臨、M10915010 盧O函 B10615046 柯O豪、B10615045 陳O富、B10601002 廖O捷 M10802131 李O宇、M10802130 陳O瑋 M10915060 林O崴、B10430302 許O森、M10815090 曾O筑、M10915002 許O樂 M10815013 陳O妮、M10815074 張O綸 M10915027 石O峰、B10615033 王O禎 B10630024 劉O奇、B10630040 吳O宏

Final Project..

- 關於期末報告 請每一組同學自選一個研究主題
  - 可以是與你或你們實驗室研究相關的題目 或是你想嘗試的題目
  - 如果沒什麼想法 也可以做單純的資訊檢索 所以題目沒有什麼限制 可以自由發揮
  - 主要是希望同學可以活用這一個學期所學的IR相關方法

- 每一組同學請報告10~15分鐘
  - (a) 研究主題是什麼
  - (b) 研究背景介紹
  - (c) 用課堂上教的東西做了什麼改進?
  - (d) 實驗結果與成果
  - 最後需要繳交課堂報告簡報檔&source codes

- 評分方式
  - 團體分數(各組互評) 13%
  - 個人分數(問與答) 2%

Review

- Pseudo-Relevance Feedback Methods
  - The Rocchio's Algorithm
  - Relevance Model
    - Topic-based RM
    - Word-based RM
  - Simple Mixture Model
  - Tri-Mixture Model

- Methods for Selecting Pseudo Relevant Documents
  - Gapped Top-*K*
  - Cluster Centroid
  - Active-RDD
  - Resampling Method

KL-Divergence.

- In mathematical statistics, the Kullback–Leibler divergence is a measure of how one **probability distribution** is different from a second **reference probability distribution**
  - It also called relative entropy
  - For discrete case

Reference Distribution

$$KL(q||d) = \sum_{x \in X} P_q(x) \log \frac{P_q(x)}{P_d(x)}$$

Approximation Distribution

  - For continuous case

$$KL(q||d) = \int_{-\infty}^{\infty} P_q(x) \log \frac{P_q(x)}{P_d(x)} dx$$

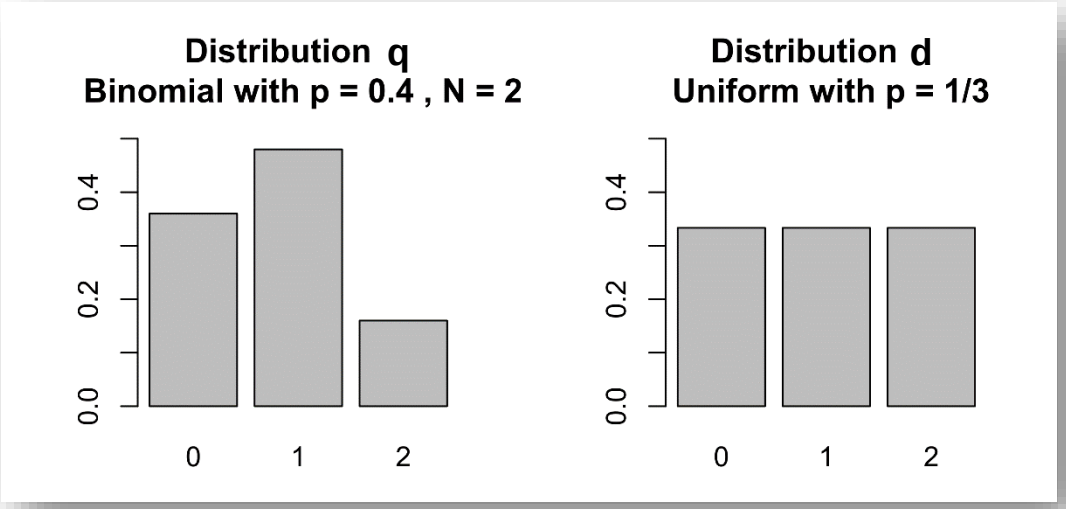
  - It is the expectation of the **logarithmic difference** between the two distributions
  - Relative entropy is always non-negative

KL-Divergence..

- Since the Kullback–Leibler divergence is a measure of how a **distribution** is different from a **reference distribution**, it is **not symmetric**

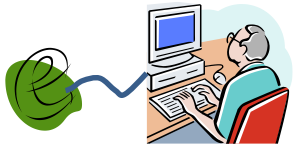
$$KL(q||d) = \sum_{x \in X} P_q(x) \log \frac{P_q(x)}{P_d(x)} = \frac{9}{25} ln\left(\frac{9/25}{1/3}\right) + \frac{12}{25} ln\left(\frac{12/25}{1/3}\right) + \frac{4}{25} ln\left(\frac{4/25}{1/3}\right) = 0.0852996$$

$$KL(d||q) = \sum_{x \in X} P_d(x) \log \frac{P_d(x)}{P_q(x)} = \frac{1}{3} ln\left(\frac{1/3}{9/25}\right) + \frac{1}{3} ln\left(\frac{1/3}{12/25}\right) + \frac{1}{3} ln\left(\frac{1/3}{4/25}\right) = 0.097455$$



x	0	1	2
Distribution $P_q(x)$	$9/25$	$12/25$	$4/25$
Distribution $P_d(x)$	$1/3$	$1/3$	$1/3$

KL-Divergence…



- Kullback-Leibler (KL)-Divergence measure is a LM-based IR method

Reference Distribution

$$KL(q||d_j) = \sum_{w \in V} P(w|q) \log \frac{P(w|q)}{P(w|d_j)}$$ ← Approximation Distribution

$$= \sum_{w \in V} P(w|q) \log P(w|q) - \sum_{w \in V} P(w|q) \log P(w|d_j)$$

$$\propto - \sum_{w \in V} P(w|q) \log P(w|d_j)$$

  - Query and document are both in the form of **probability distributions**

KL-Divergence….

- Zero probability problem
  - Using background language
    - Similar to the role of IDF!

$$KL(q||d_j) \propto -\sum_{w \in V} P(w|q) \log P(w|d_j)$$

$$P(w|q) = \alpha \times P_{ULM}(w|q) + \beta \times P_{RM}(w|q) + (1 - \alpha - \beta) \times P_{BG}(w)$$

$$P(w|d_j) = \gamma \times P_{ULM}(w|d_j) + (1 - \gamma) \times P_{BG}(w)$$

$$P_{ULM}(w|q) = \frac{c(w,q)}{|q|}$$

$$P_{ULM}(w|d_j) = \frac{c(w,d_j)}{|d_j|}$$

$$P_{BG}(w) = \frac{\sum_{d_k \in D} c(w,d_k)}{\sum_{d_i \in D} |d_i|}$$

- How to speedup?

$$KL(q||d_j) \propto -\sum_{w \in V} P(w|q) \log P(w|d_j)$$

Various Query Modeling

- Vector Space Model
  - Rocchio's Algorithm
- Language Models
  - Relevance Model
  - Simple Mixture Model
  - Tri-mixture Model
  - PLSA?
- Probabilistic Model
  - BM25?

The Evolution

David M. Blei
Columbia University, USA

Thomas Hofmann
ETH Zurich, Switzerland

2003 Latent Dirichlet Allocation

2001 Relevance-based LM & Simple Mixture Model

1999 Probabilistic Latent Semantic Analysis

1998 Language Modeling Approaches

1994 Best Match Models (Okapi Systems)

1988 Latent Semantic Analysis

1976 Probabilistic Model

1975 Vector Space Model

1973 Boolean Model

1972 Inverse Document Frequency

1965 Rocchio Algorithm

1957 Term Frequency

Scott Deerwester

Stephen Robertson, 1946-present

Karen Spärck Jones, 1935-2007

Hans Peter Luhn, 1896-1964

V. Lavrenko
Edinburgh

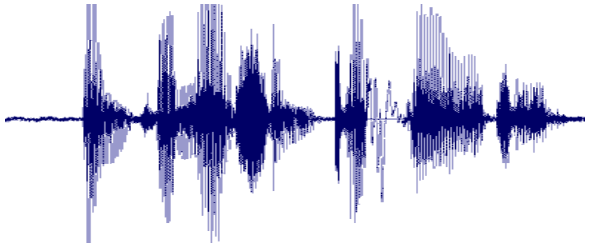
C.X. Zhai
Illinois University

Gerard Salton, 1927-1995

Frederick Wilfrid Lancaster, 1933-2013

What are we looking for?

- A **Wonderful** Function!
  - Speech Recognition

$f($  $) =$ It is a nice day today

  - Handwritten Recognition

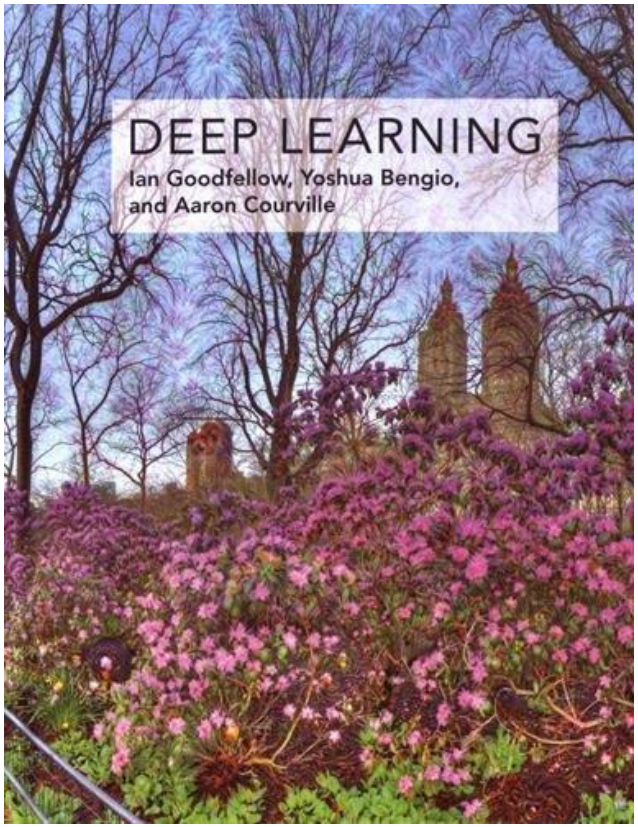
$f($  $) =$ It is a nice day today

  - Machine Translation

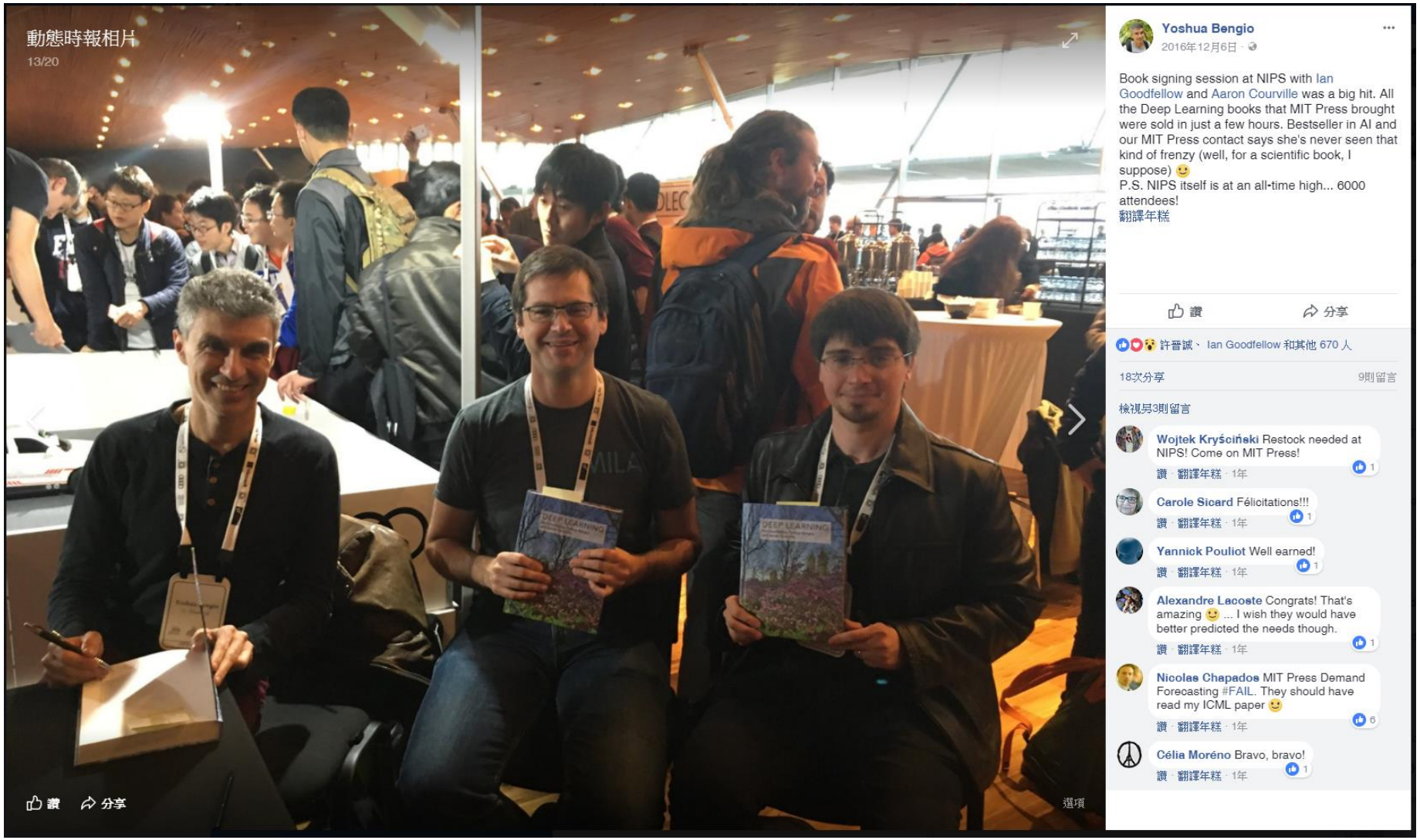
$f(\text{"今天天氣很好"}) =$ It is a nice day today

Bible

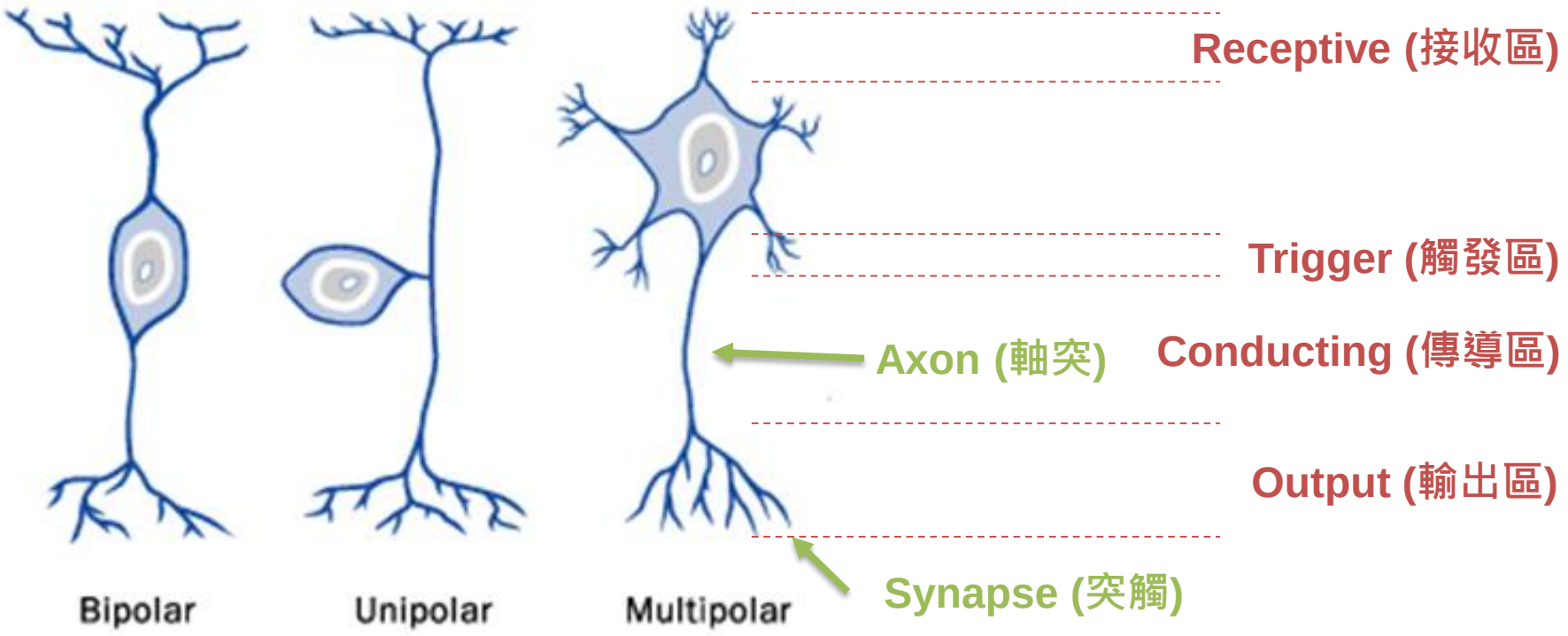
- Ian Goodfellow and Yoshua Bengio and Aaron Courville, Deep Learning, MIT Press, 2016



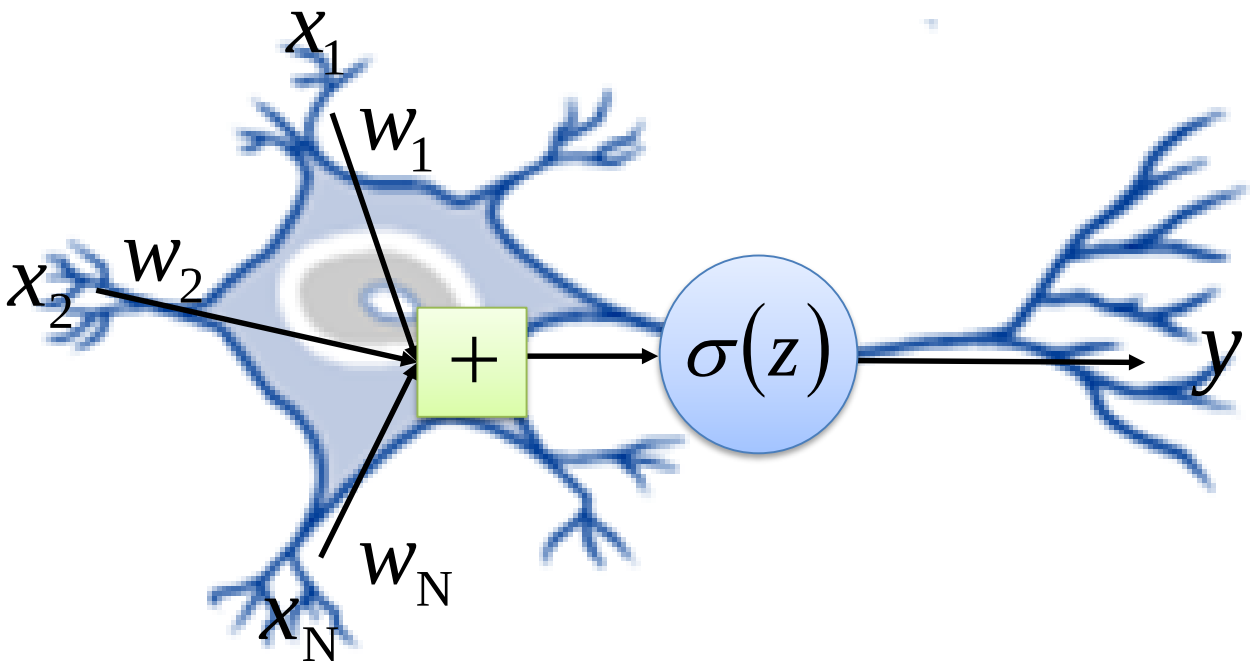
Book Signing at NIPS 2016



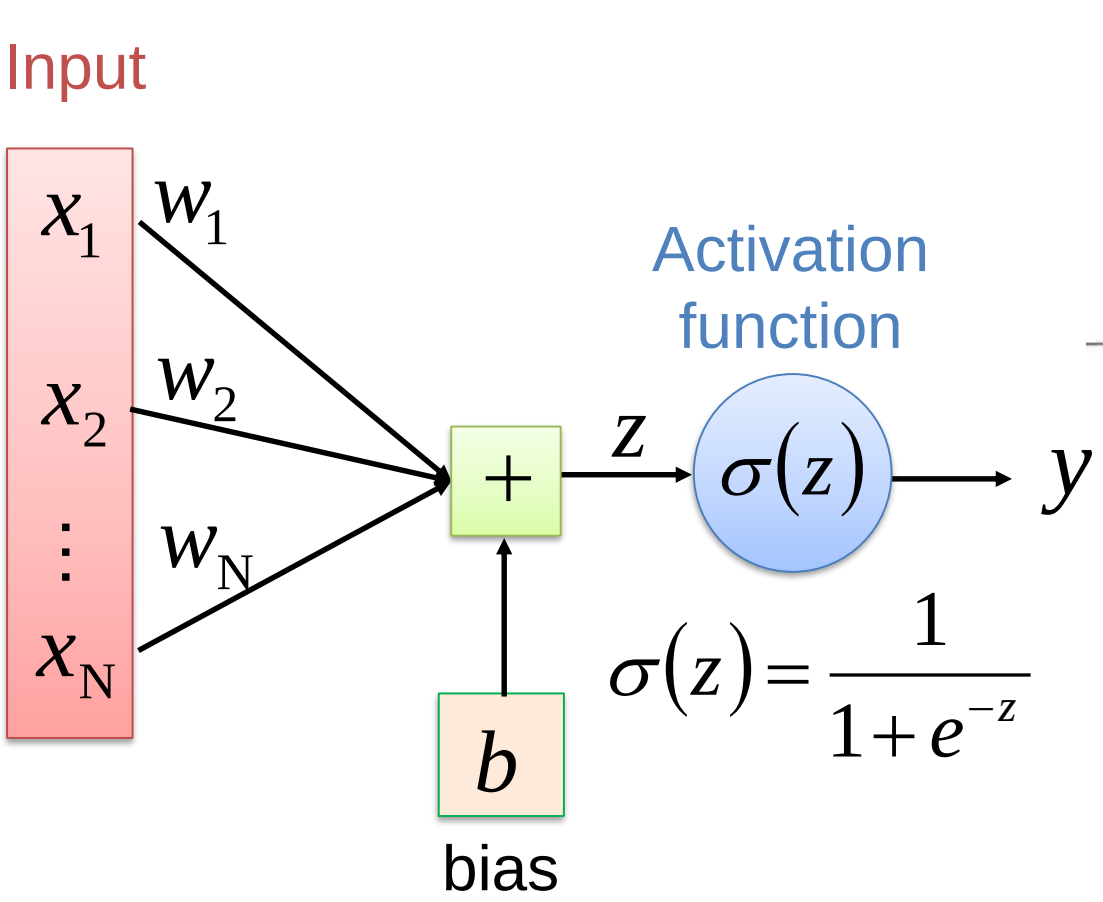
Neuron

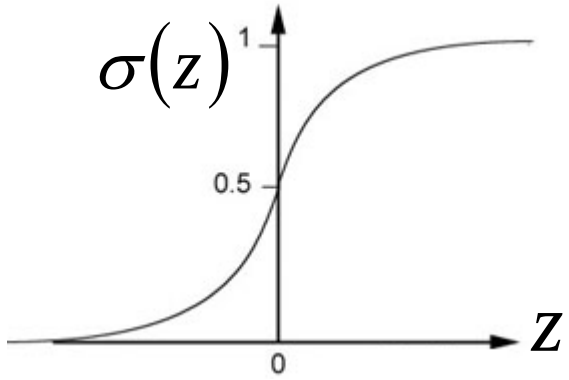

Dendrites (樹突)
Receptive (接收區)
Trigger (觸發區)
Axon (軸突)
Conducting (傳導區)
Output (輸出區)
Synapse (突觸)
Bipolar
Unipolar
Multipolar

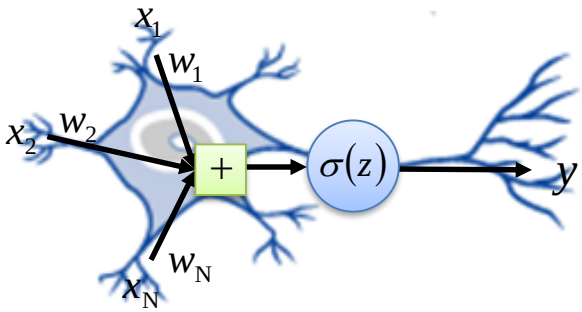
Handcrafted Neuron



Single Neuron

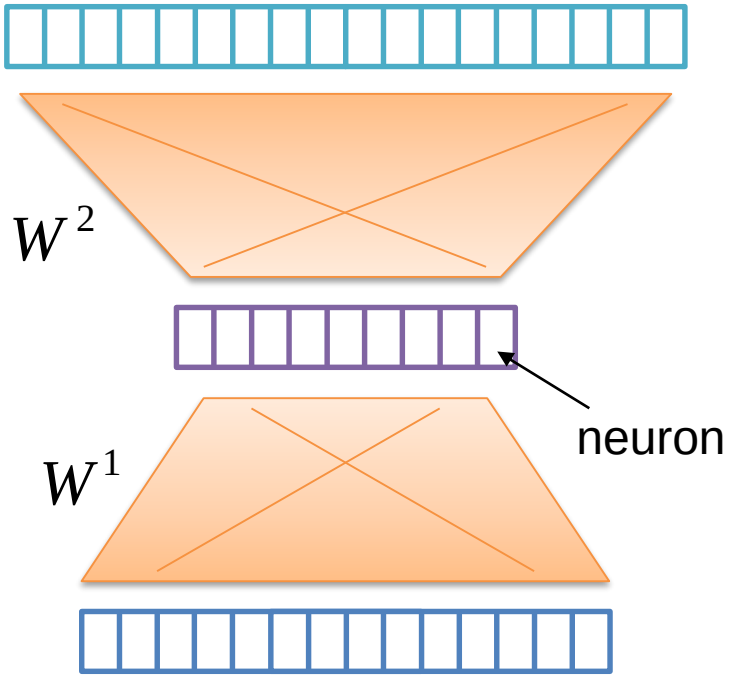


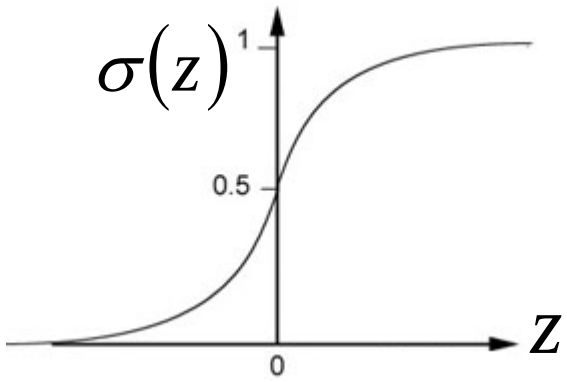




Neural Networks – a hidden layer

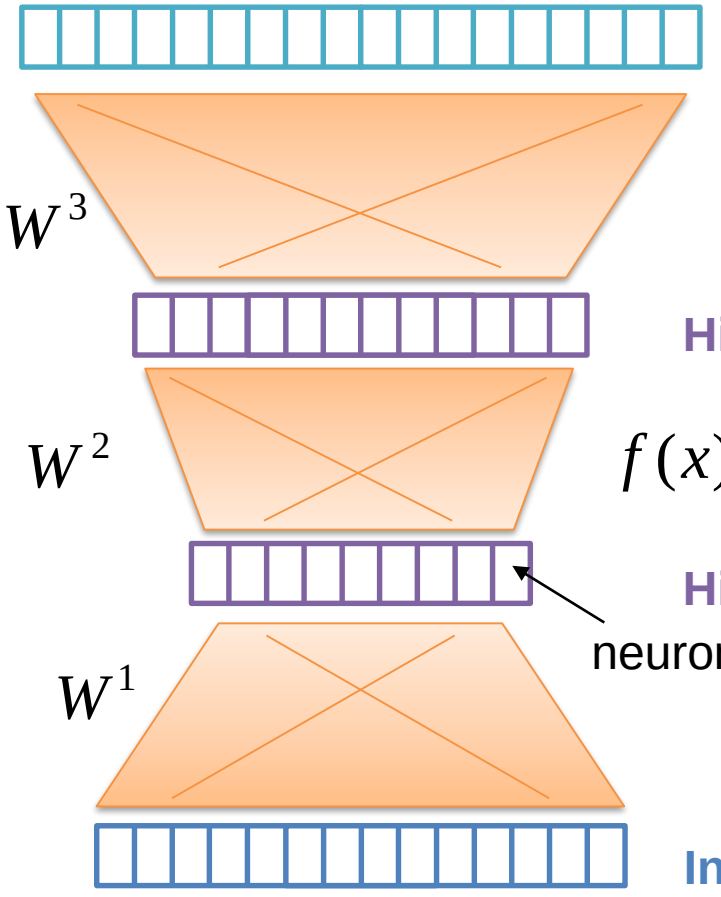
$$f(x) = \sigma\left(W^2 \cdot \sigma\left(W^1 \cdot x + b^1\right) + b^2\right) = y$$





Neural Networks – two hidden layers

- It is the well-known **fully-connected feed-forward** neural network



$$f(x) = \sigma\left(W^3 \cdot \sigma\left(W^2 \cdot \sigma\left(W^1 \cdot x + b^1\right) + b^2\right) + b^3\right)$$

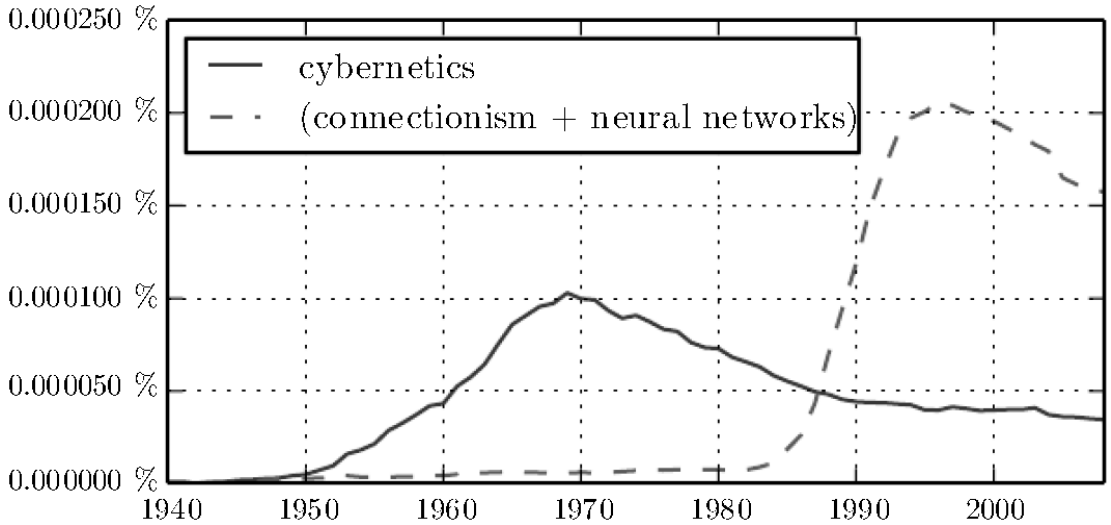
Easy to implement: tools
Highly computational Cost: GPU
Hyper-parameters: ?

Deep Neural Networks

- Deep Neural Networks
  - G. Hinton (UTORONTO & Google) @*IEEE Signal Processing Magazine 2012*
    - **more than one layer of hidden units**
  - D. Yu (Microsoft Research) @Automatic Speech Recognition 2015
    - The term deep neural network was originally introduced to mean **multilayer perceptron with many hidden layers**, but was later extended to mean any neural network with a deep structure
  - Rich Caruana (Microsoft Research) @ASRU2015
    - **three hidden layers**

Deep Learning & Deep Neural Networks

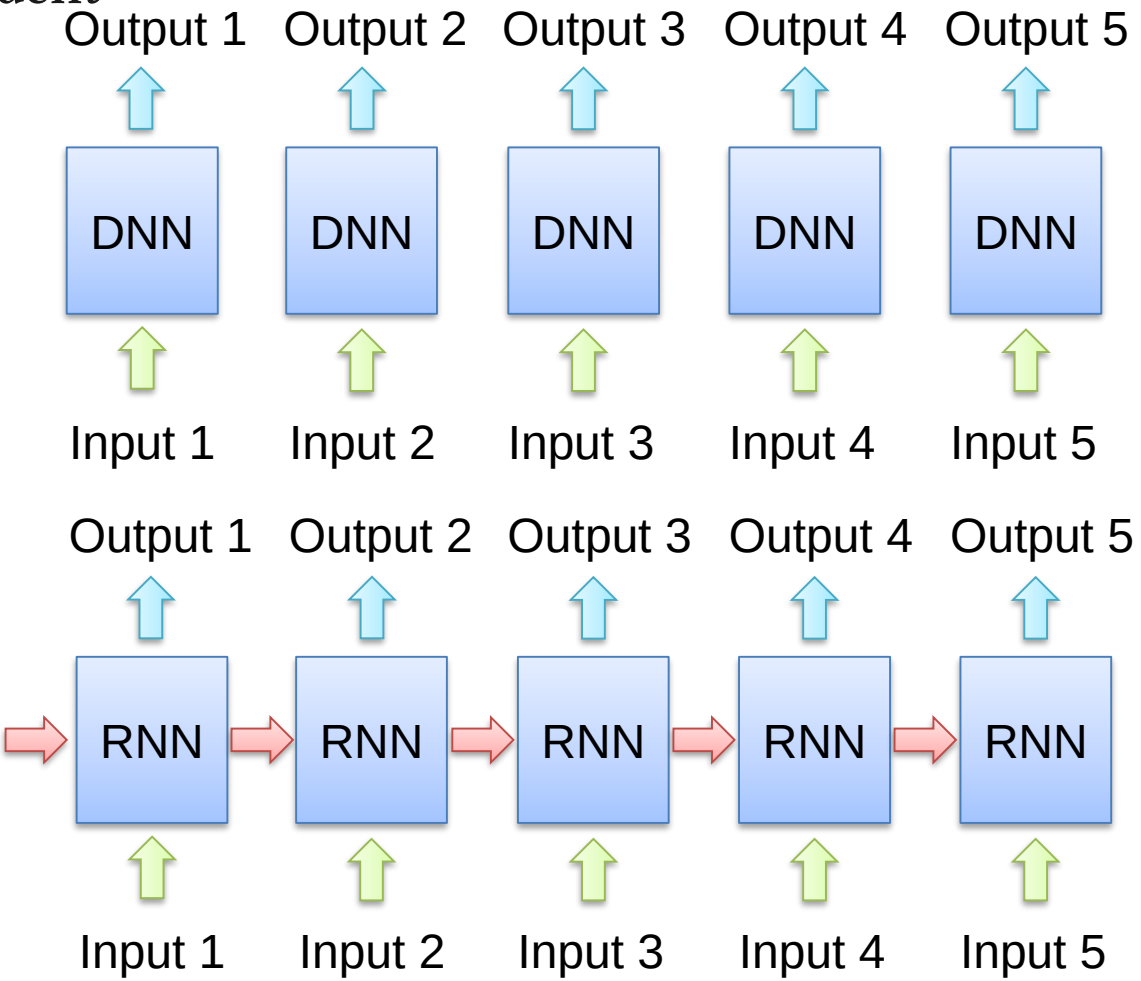
- Yoshua Bengio (UMONTRAL) *@Deep Learning, 2015*
  - Deep learning has a long and rich history. It only appears to be new, because it was relatively unpopular for several years preceding its current popularity, and because it has gone through many different names.
    - 1940s~1960s: cybernetics (McCulloch-Pitts Neuron, 1943; Perceptron, 1958)
    - 1980s~1990s: connectionism (Neocognitron, 1980)
    - 2006~: deep learning



Neural Networks with Memory.

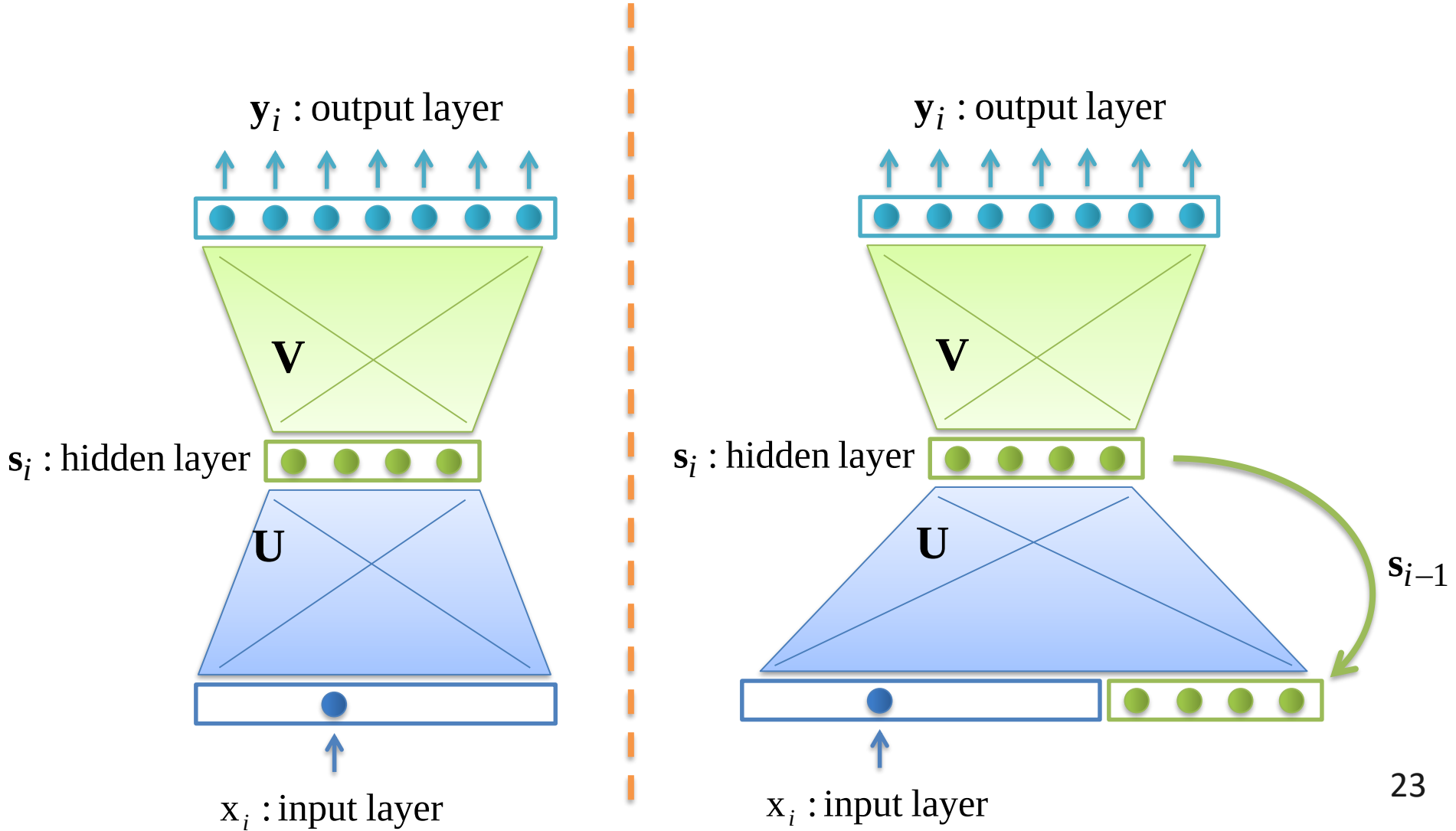
- Conventional neural network can only capture local information
  - Samples are independent
  - Time series data
    - Word sequence
    - Picture sequence

      Movie
    - Speech Signal



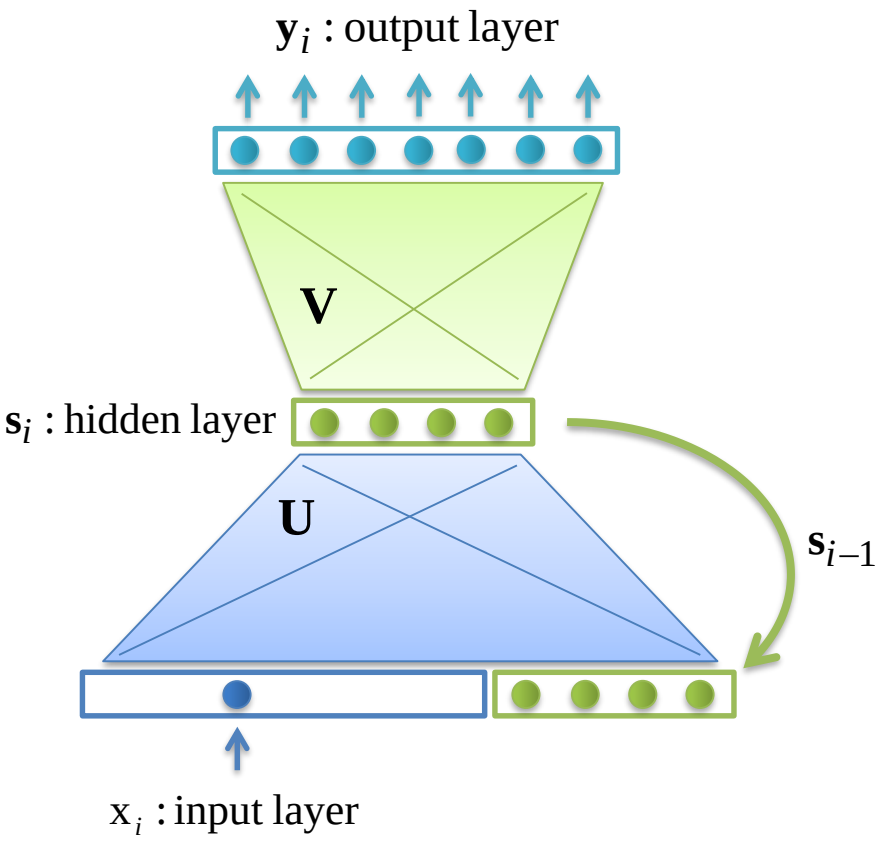
Neural Networks with Memory..

- Hidden layer can be thought as a memory!!

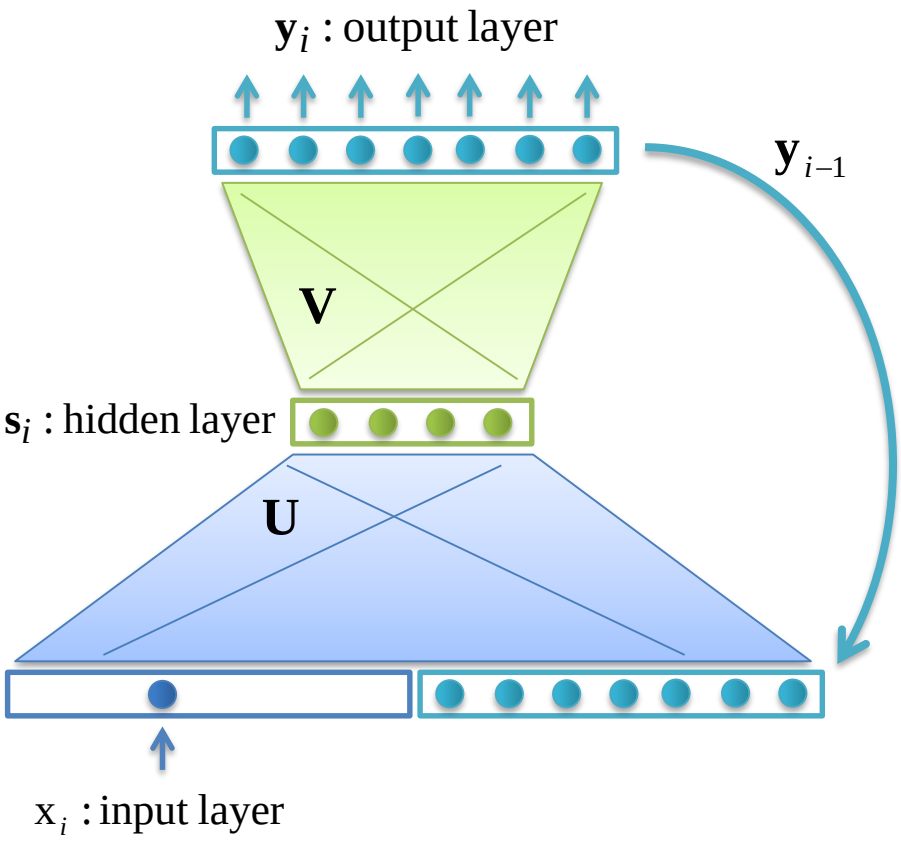


Neural Networks with Memory…

Elman-type RNN

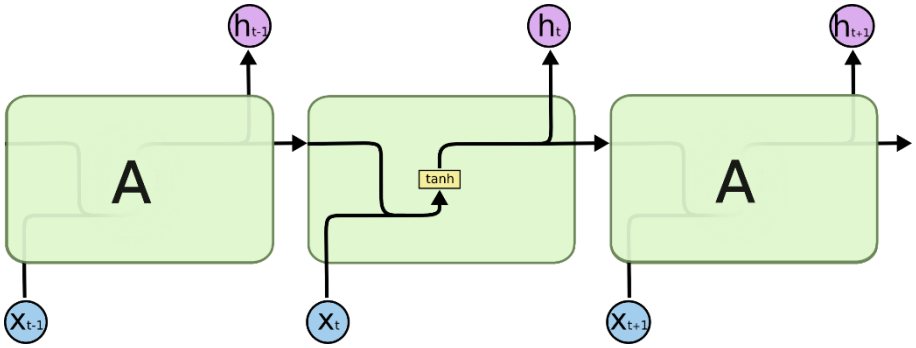


Jordan-type RNN

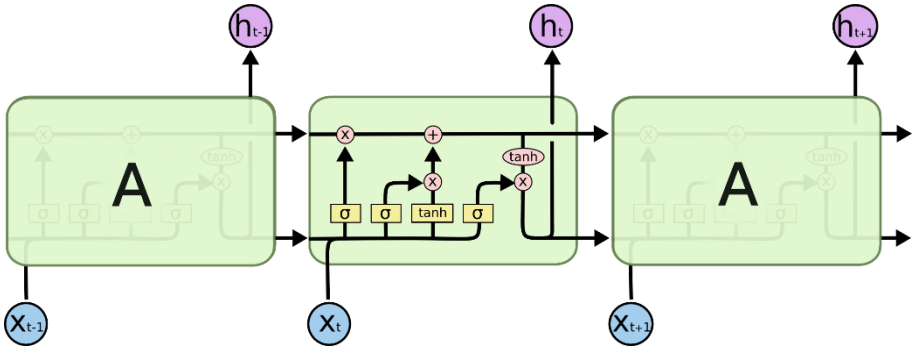


RNN, LSTM & GRU

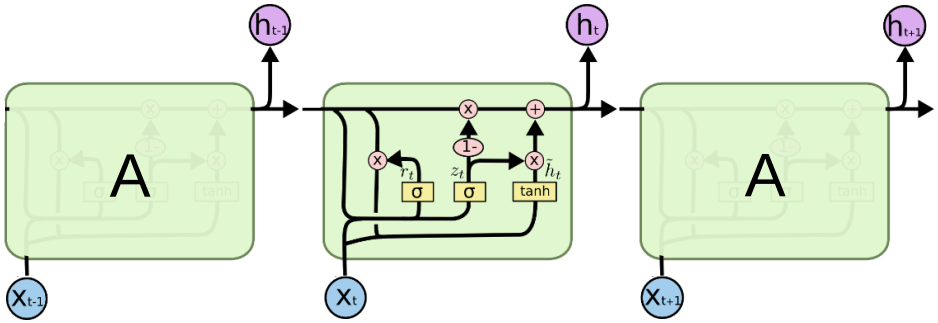
- RNN
  - The classic model



- LSTM
  - Learning to forget
  - Capture longer information
  - Very slow in practice



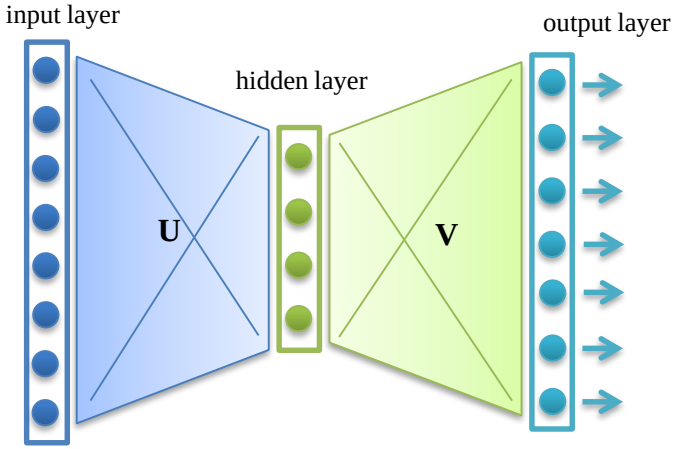
- GRU
  - A balanced choice



Parsimonious Neural Networks

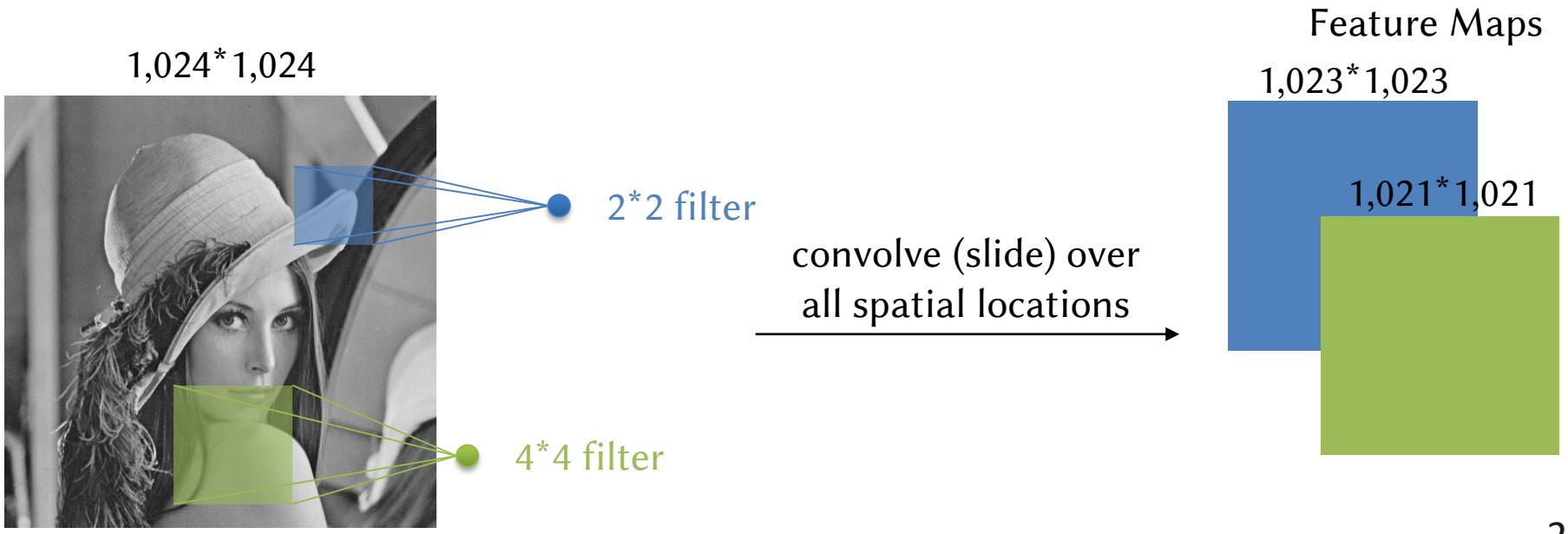
- For image processing, the conventional neural networks have to estimate too many parameters
  - For a 1024*1024 image, the size of the input layer is up to 1,048,576
  - If the size of the first hidden layer is 100, the number of model parameter is over 104,857,600

1,024*1,024

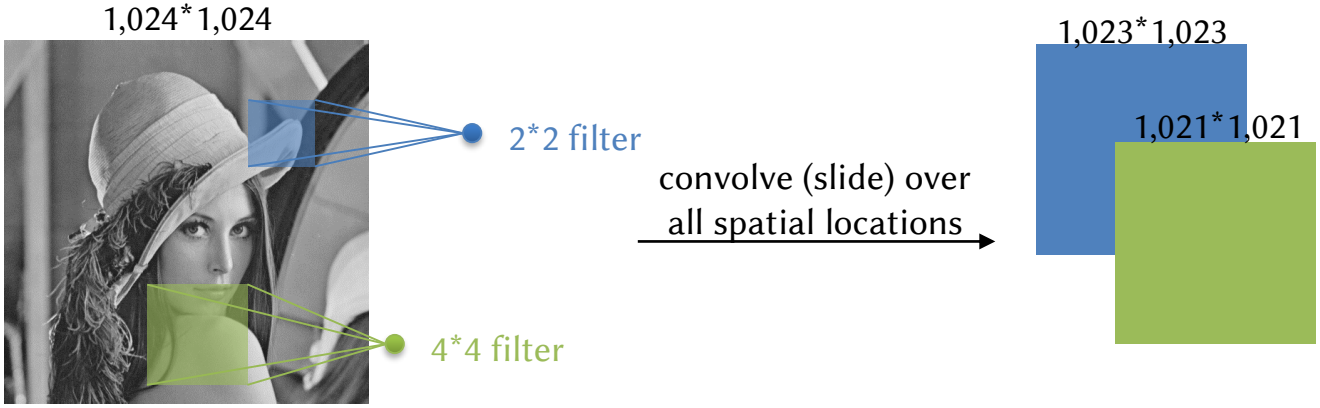


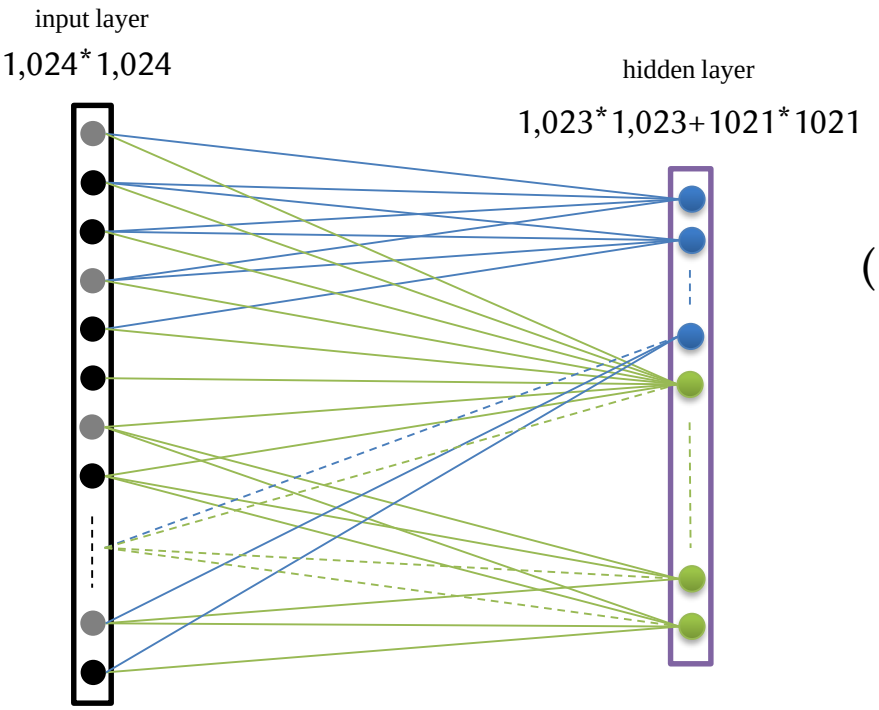
Convolution Neural Networks

- Inspired from the visual cortex, each neuron can only perceive a sub-region (perceptive field) at a time
  - **Convolve** the filter with the image
  - If we have two filters (2*2 and 4*4), the total parameters are 4+16=20
    - Parameter sharing!



Parameter Sharing





$$(1024*1024)*(1023*1023+1021*1021)$$

$$2*2+4*4$$

Joint the Trend!

- You can build your own architectures easily and flexibly

- Do not need to take care about the mathematics
  - Optimization is only a function!

  - Open Neural Network Exchange

One Framework to Rule Them All

She is a chief scientist at Google Cloud from 2016 to 2018

Google:
TensorFlow

"One framework to rule them all"

Facebook:
PyTorch +Caffe2

Research

Production

Language Modeling

- A goal of statistical language modeling is to learn the joint probability function of sequences of words in a language

$$P(w_1, w_2, \dots, w_T)$$

- A statistical model of language can be represented by the conditional probability of the next word given all the previous ones (**chain rule**)

$$P(w_1, w_2, \dots, w_T) = \prod_{t=1}^{T} P(w_t | w_1, w_2, \dots, w_{t-1})$$

$$\approx \prod_{t=1}^{T} P(w_t | w_{t-n+1}, \dots, w_{t-1})$$

  - Such statistical language models have already been found useful in many technological applications involving natural language

NN-based Language Models



Language Representations (2013~)

Neural Network Language Models (2001~)

Continuous Language Models

Continuous Language Models (2007~2009)

Topic Models (1997~2003)


Topic Models

Query Language Models (2001~2006)


Word-Regularity Models

Discriminative Language Models (2000~2011)




Word-Regularity Models (~1997)

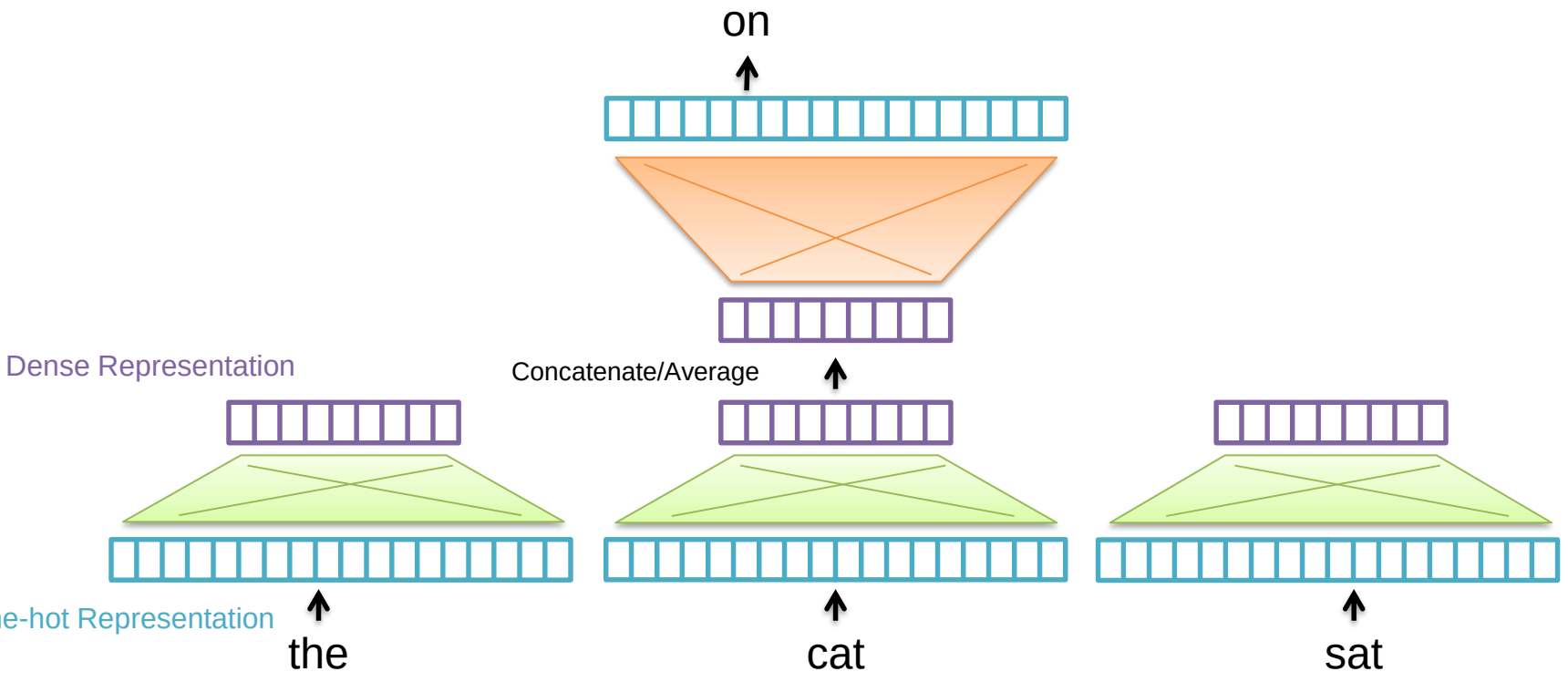
2000 2002 2004 2006 2008 2010 2012 2014 2016

Y. Bengio, *et al.*, "A Neural Probabilistic Language Model," in *Proc. of NIPS*, 2001.

Neural Network Language Modeling

- The Neural Network Language Mode (NNLM) estimated a statistical (n-gram) language model for **predicting future words**

$$P(w_1, w_2, \ldots, w_T) \approx \prod_{t=1}^{T} P(w_t | w_{t-n+1}, \ldots, w_{t-1})$$



From Modeling to Vectorization

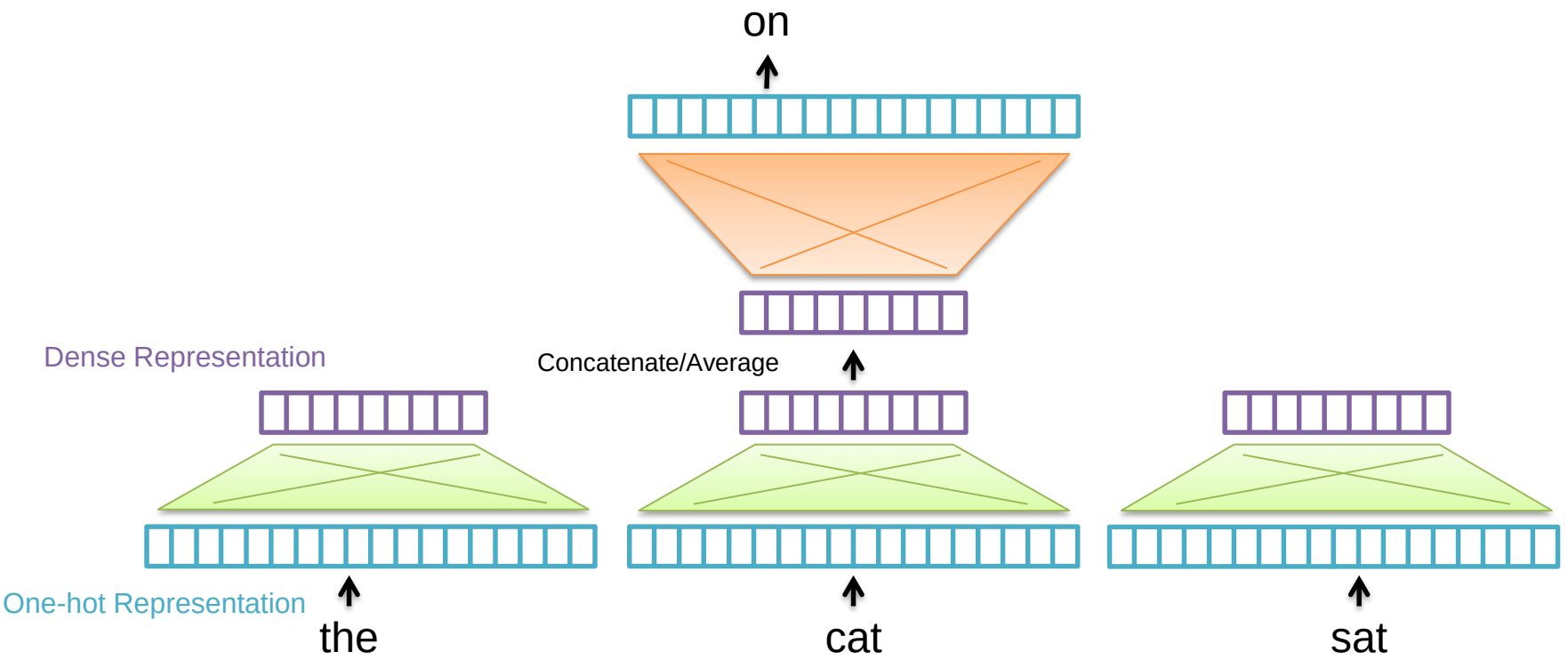
- Recent methods for learning vector space representations of words have succeeded in capturing **fine-grained semantic** and **syntactic** regularities
  - One-hot representation vs. Distributed representation

- The two main model families for learning word vectors
  1. Global matrix factorization methods
     - Global Vector
  2. Local context window methods
     - Continuous Bag-of-Words model, and Skip-gram model

Y. Bengio, *et al.*, "A Neural Probabilistic Language Model," in *Proc. of NIPS*, 2001.

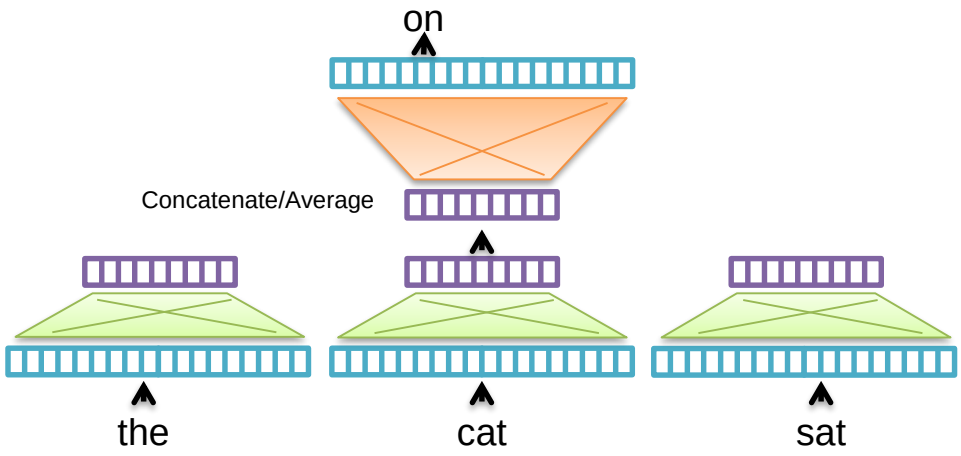
NNLM

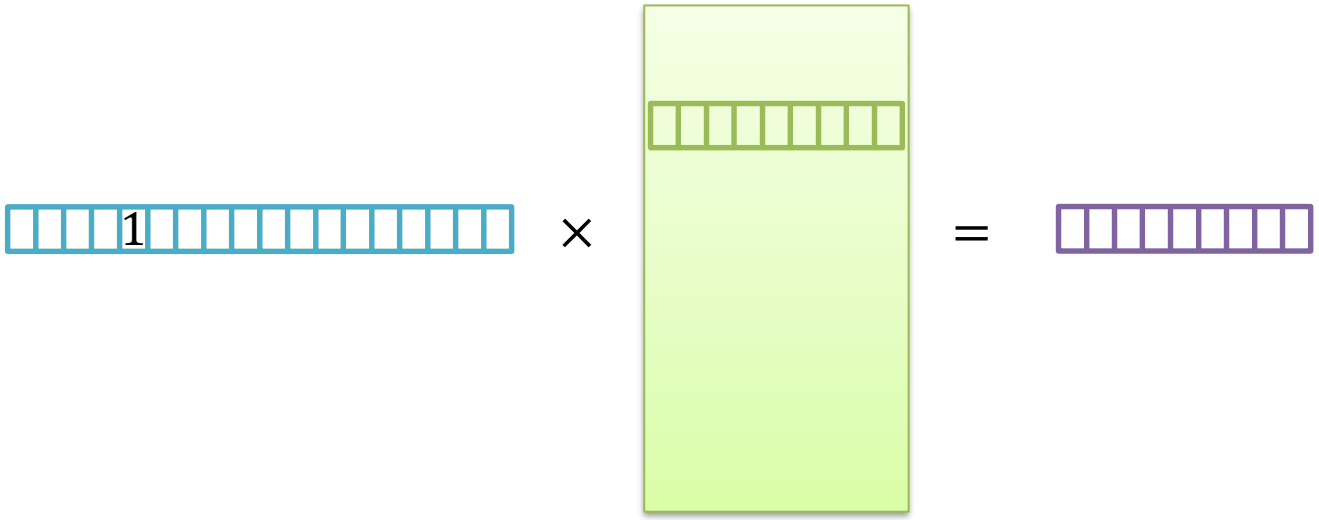
- Perhaps one of the most-known seminal studies on developing word embedding methods was rooted in the Neural Network Language Modeling (NNLM)



  - It estimated a statistical (n-gram) language model for **predicting future words** in context while inducing word embeddings as a by-product

One-hot to Dense

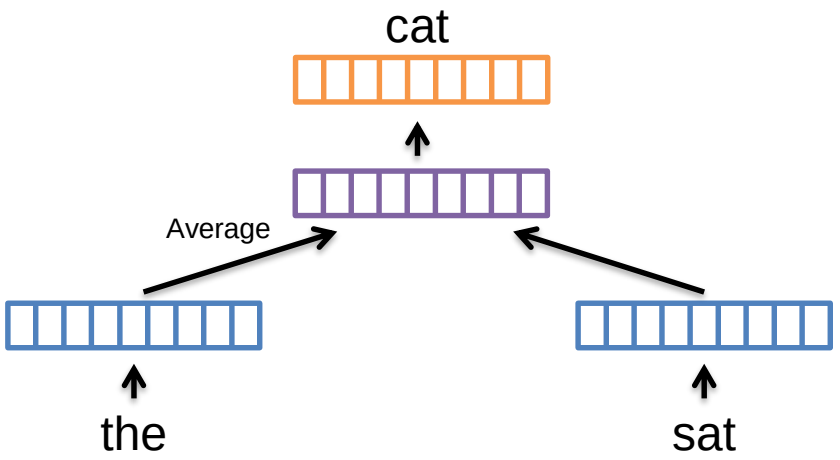




Continuous Bag-of-Words Modeling – 1

- Rather than seeking to learn a statistical language model, the CBOW model manages to obtain a dense vector representation (embedding) of each word directly

$$\prod_{t=1}^{T} P(w_t | w_{t-c}, \ldots, w_{t-1}, w_{t+1}, \ldots, w_{t+c}) = \prod_{t=1}^{T} \frac{\exp(v_{\bar{w}_t} \cdot v_{w_t})}{\sum_{w \in V} \exp(v_{\bar{w}_t} \cdot v_w)}$$

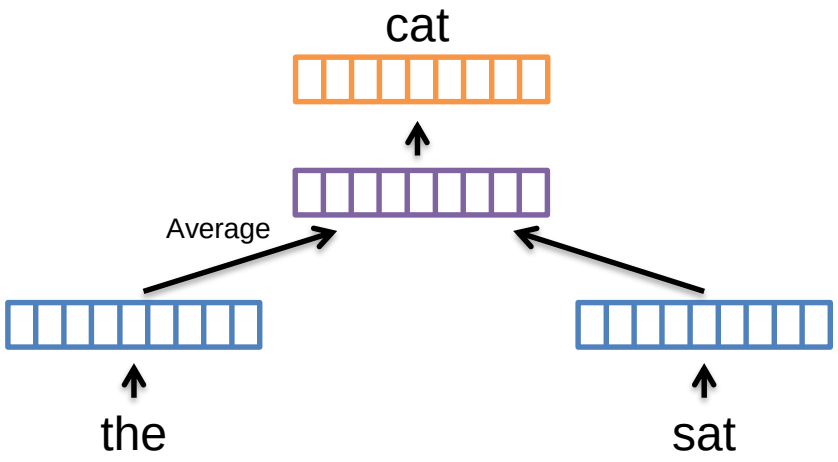


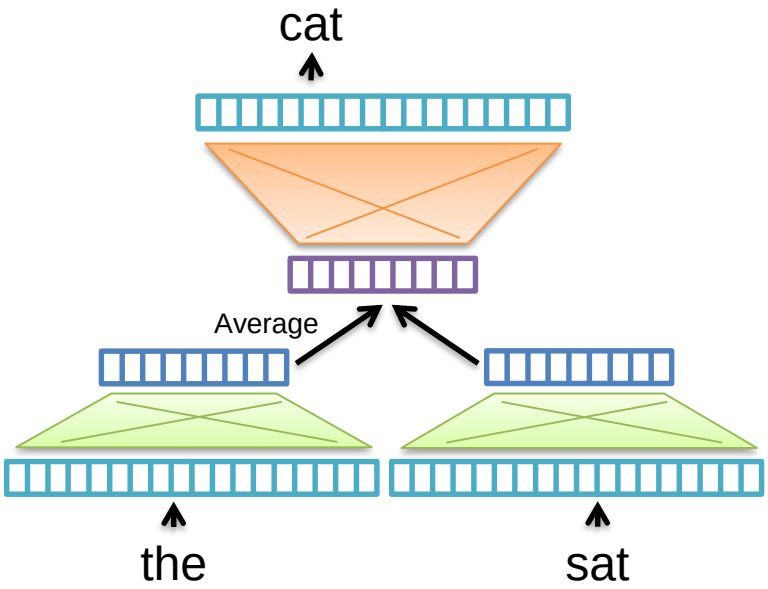
$$v_{\bar{w}_t} = \frac{1}{2c} \sum_{j=-c \&\& j \neq 0}^{c} v_{w_{t+j}}$$

T. Mikolov, *et al.*, "Efficient Estimation of Word Representations in Vector Space," in *Proc. of ICLR*, 2013.

Continuous Bag-of-Words Modeling – 2

$$\prod_{t=1}^{T} P(w_t | w_{t-c}, \ldots, w_{t-1}, w_{t+1}, \ldots, w_{t+c}) = \prod_{t=1}^{T} \frac{\exp(v_{\bar{w}_t} \cdot v_{w_t})}{\sum_{w \in V} \exp(v_{\bar{w}_t} \cdot v_w)}$$





T. Mikolov, *et al.*, "Efficient Estimation of Word Representations in Vector Space," in *Proc. of ICLR*, 2013.

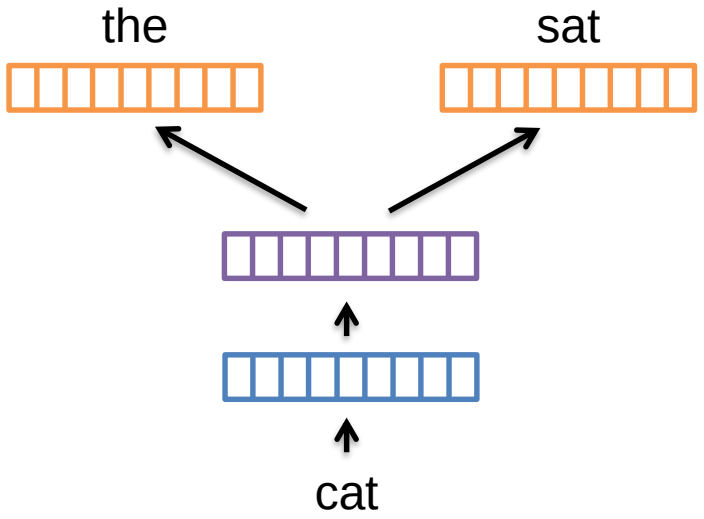
Skip-Gram Modeling

- In contrast to the CBOW model, the SG model employs an inverse training objective for learning word representations

$$\prod_{t=1}^{T} P(w_{t-c}, \ldots, w_{t-1}, w_{t+1}, \ldots, w_{t+c} | w_t)$$

$$= \prod_{t=1}^{T} \prod_{j=-c \&\& j \neq 0}^{c} P(w_{t+j} | w_t)$$

$$= \prod_{t=1}^{T} \prod_{j=-c \&\& j \neq 0}^{c} \frac{\exp(v_{w_{t+j}} \cdot v_{w_t})}{\sum_{w \in V} \exp(v_w \cdot v_{w_t})}$$



  - In the implementations of CBOW and SG, the **hierarchical soft-max algorithm** and the **negative sampling algorithm** can make the training process more efficient and effective

The Training Process

- Negative Sampling
  - Noise contrastive estimation (NCE) posits that a good model should be able to differentiate data from noise
    - Take skip-gram for example

$$\prod_{t=1}^{T} P(w_{t-c}, \dots, w_{t-1}, w_{t+1}, \dots, w_{t+c} | w_t)$$

$$= \prod_{t=1}^{T} \prod_{j=-c \&\& j \neq 0}^{c} P(w_{t+j} | w_t)$$

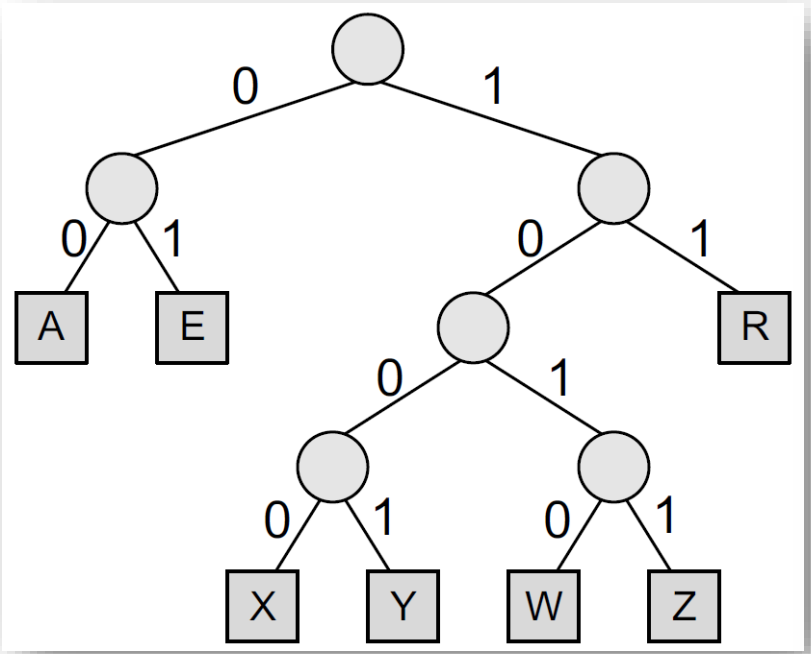
$$= \prod_{t=1}^{T} \prod_{j=-c \&\& j \neq 0}^{c} \frac{\exp(v_{w_{t+j}} \cdot v_{w_t})}{\sum_{w \in V} \exp(v_w \cdot v_{w_t})} \simeq \prod_{t=1}^{T} \prod_{j=-c \&\& j \neq 0}^{c} \frac{\exp(v_{w_{t+j}} \cdot v_{w_t})}{\sum_{w'} \exp(v_{w'} \cdot v_{w_t})}$$

- Hierarchical Softmax
  - The main advantage is that it is needed to evaluate only about $log_2(V)$ nodes

F. Morin and Y. Bengio, "Hierarchical Probabilistic Neural Network Language Model," in *Proc. of AISTATS*, 2005.

Hierarchical Softmax.

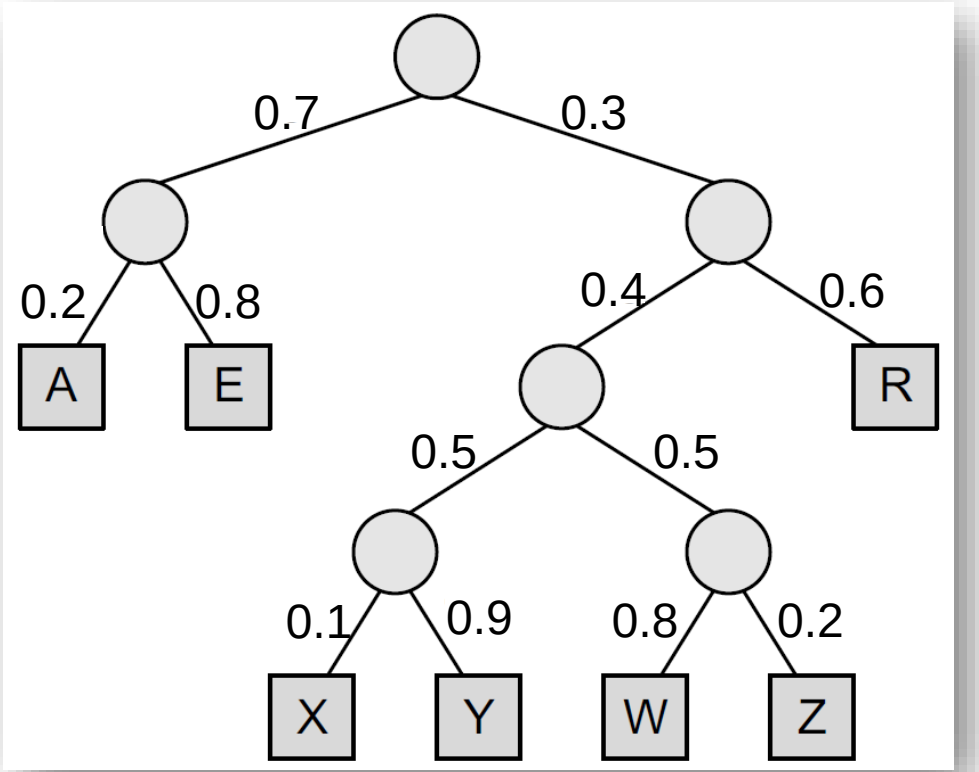
- For a Huffman tree, every left branch is coded with 0 and every right branch is coded with 1
  - For a character sequence: AAERZ
    - By Huffman Coding Scheme: 000001111011
    - By Original Coding Scheme: 000000001010110



Character	Code	Original Coding
A	00	000
E	01	001
R	11	010
W	1010	011
X	1000	100
Y	1001	101
Z	1011	110

Hierarchical Softmax..

- For a Huffman tree, every left branch is coded with 0 and every right branch is coded with 1
  - For a character sequence: AAERZ
    - By Huffman Coding Scheme: 000001111011
    - By Original Coding Scheme: 000000001010110



$$P(A) = 0.7 \times 0.2$$

$$P(E) = 0.7 \times 0.8$$

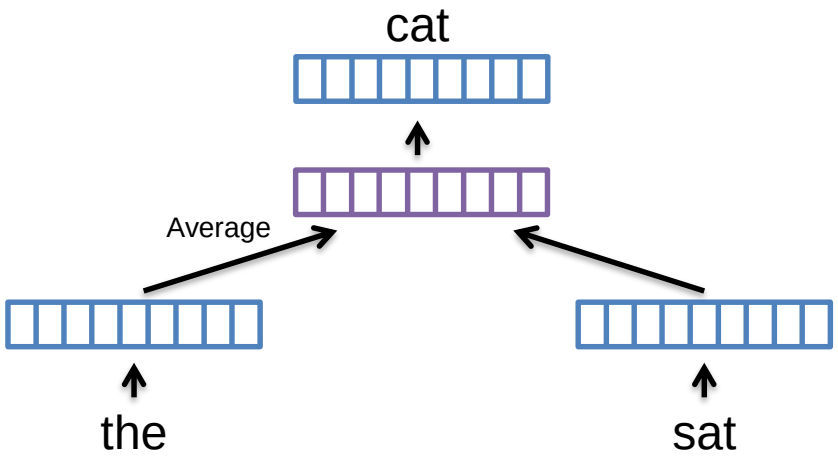
$$P(X) = 0.3 \times 0.4 \times 0.5 \times 0.1$$

$$P(Y) = 0.3 \times 0.4 \times 0.5 \times 0.9$$

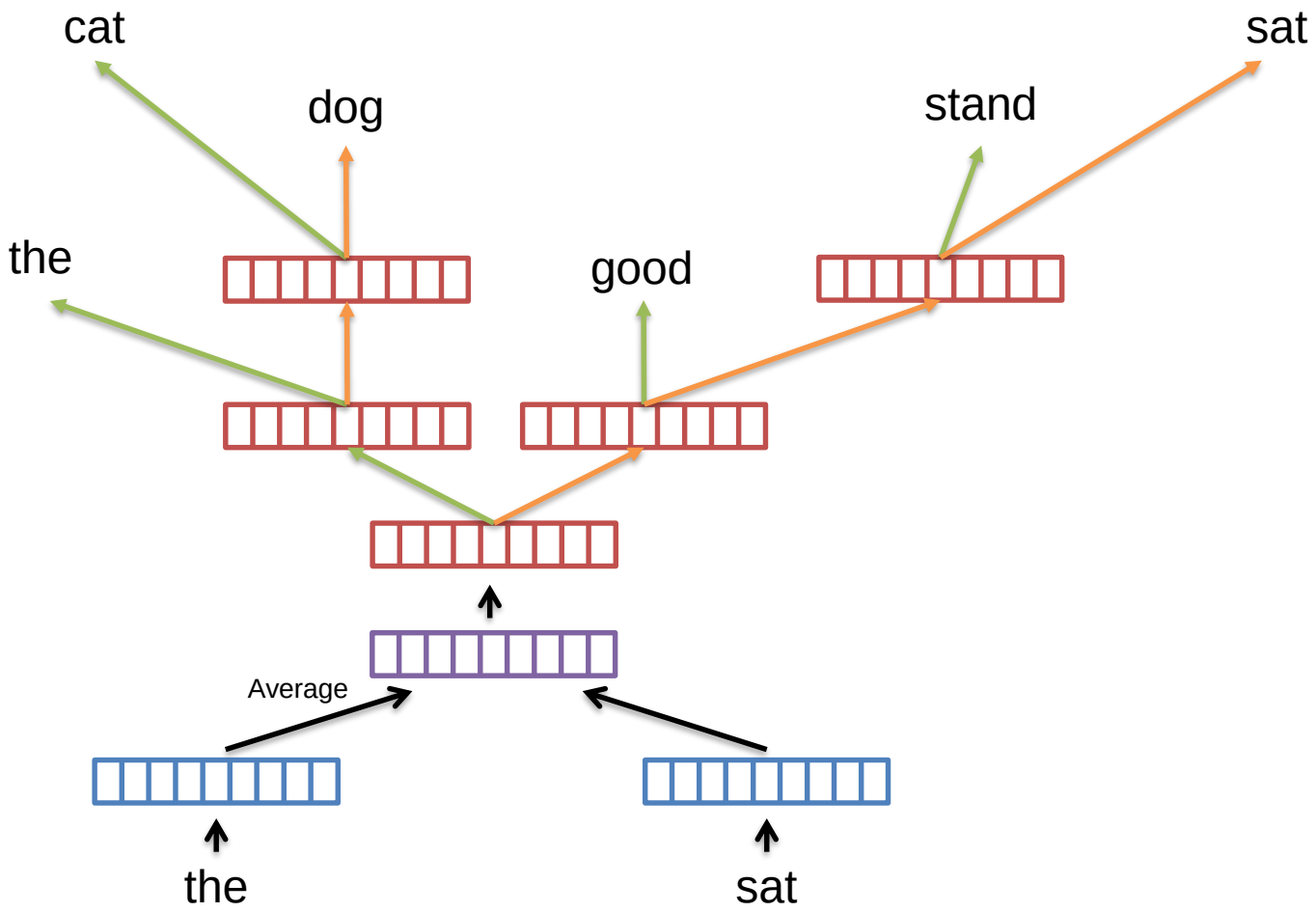
$$\vdots$$

$$P(R) = 0.3 \times 0.6$$

Hierarchical Softmax…



Hierarchical Softmax….



J. Pennington, *et al.*, "GloVe: Global Vectors for Word Representation," in *Proc. of EMNLP*, 2014.

Global Vector Model (GloVe)

- The idea is:
  - For word solid related to ice but not steam, we expect the ratio:
    - $P_{ice-solid}/P_{steam-solid}$ will be large
    - $P_{ice-gas}/P_{steam-gas}$ will be small
    - $P_{ice-fashion}/P_{steam-fashion}$ should be close to one

- The **starting point** for word vector learning should be with **ratios of co-occurrence probabilities** rather than the probabilities themselves
  - A weighted least squares regression model can be introduced to addresses these problems

$$\sum_{i=1}^{|V|}\sum_{j=1}^{|V|} f(X_{ij})\left(v_{w_i}\cdot v_{w_j} + b_i + b_j - log(X_{ij})\right)^2$$

Singular Value Decomposition

- The **SG** and the **GloVe** have an implicit/explicit relation with the classic weighted matrix factorization approach

- Motivated by the relationship between word embedding methods with matrix factorization, we also leverage the singular value decomposition (SVD) to derive the word embeddings

$$\mathrm{A}_{|V|\times|V|} \approx \mathrm{U}_{|V|\times K}\Sigma_{K\times K}\mathrm{V}^{\mathrm{T}}_{K\times|V|} = \mathrm{A}'_{|V|\times|V|}$$

$$\|\mathrm{A}-\mathrm{A}'\|_F^2 = \sum_{i=1}^{|V|}\sum_{j=1}^{|V|}\left(\log(X_{ij}) - \mathrm{U}_i^{\mathrm{T}}\cdot\Sigma\cdot\mathrm{U}_j\right)^2$$

  - Each row vector of matrix U (or the column vector of matrix V^{T}) is the word embeddings corresponding to each distinct word in the vocabulary

$$\sum_{i=1}^{|V|}\sum_{j=1}^{|V|} f(X_{ij})\left(v_{w_i}\cdot v_{w_j} + b_i + b_j - log(X_{ij})\right)^2$$

Classic Word Embeddings

- Various word embeddings have been proposed and applied to several NLP-related tasks
  - Prediction-based Methods
    - CBOW and Skip-gram
      - Local context
  - Count-based Methods
    - GloVe and SVD
      - Global matrix

Word Embeddings for IR – 1

- A straightforward way to leverage the word embedding methods for IR is to represent a document (or query) by averaging the vector representations of words occurring in the document (query)

$$\vec{d} = \sum_{w \in d} \frac{c(w,d)}{|d|} v_w \qquad \vec{q} = \sum_{w \in q} \frac{c(w,q)}{|q|} v_w$$

- Accordingly, the cosine similarity measure can be used to quantify the relevance degree between a document and a query

$$sim(d,q) = \cos(\vec{d}, \vec{q}) = \frac{\vec{d} \cdot \vec{q}}{\left|\vec{d}\right| \cdot |\vec{q}|}$$

Word Embeddings for IR – 2

- In addition to the vector space model, we can construct a new word-based language model for predicting the occurrence probability of any arbitrary word by using the word embeddings

$$P(w_j|w_i) = \frac{\exp(v_{w_j} \cdot v_{w_i})}{\sum_{w \in V} \exp(v_w \cdot v_{w_i})}$$

- Consequently, the document model can be obtained by linearly combining the associated word-based language models of the words occurring in the document

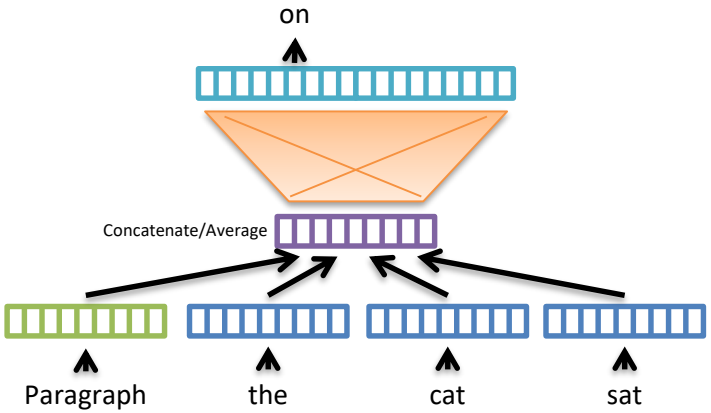
$$P(q|d) = \prod_{j=1}^{|q|} P(w_j|d) = \prod_{j=1}^{|q|} \left(\sum_{i=1}^{|d|} P(w_j|w_i) P(w_i|d) \right)$$

Translation Model

Distributed Memory (DM) Model

- Learning of paragraph representations is more reasonable and suitable for some tasks
  - The distributed memory model, the distributed bag-of-words model, and the thought vector model

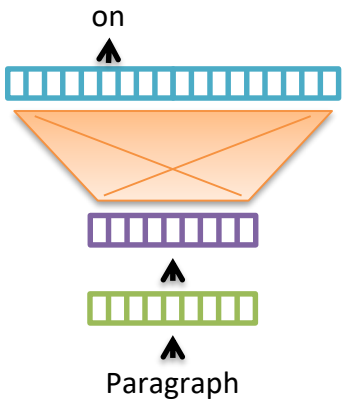
- The DM model is inspired from the CBOW model



  - The idea is that a given paragraph also contributes to the prediction of a next word

Distributed Bag-of-words (DBOW) Model

- Opposite to the DM model, a simplified version is to only leverage the paragraph representation to predict all of the words occurring in the paragraph

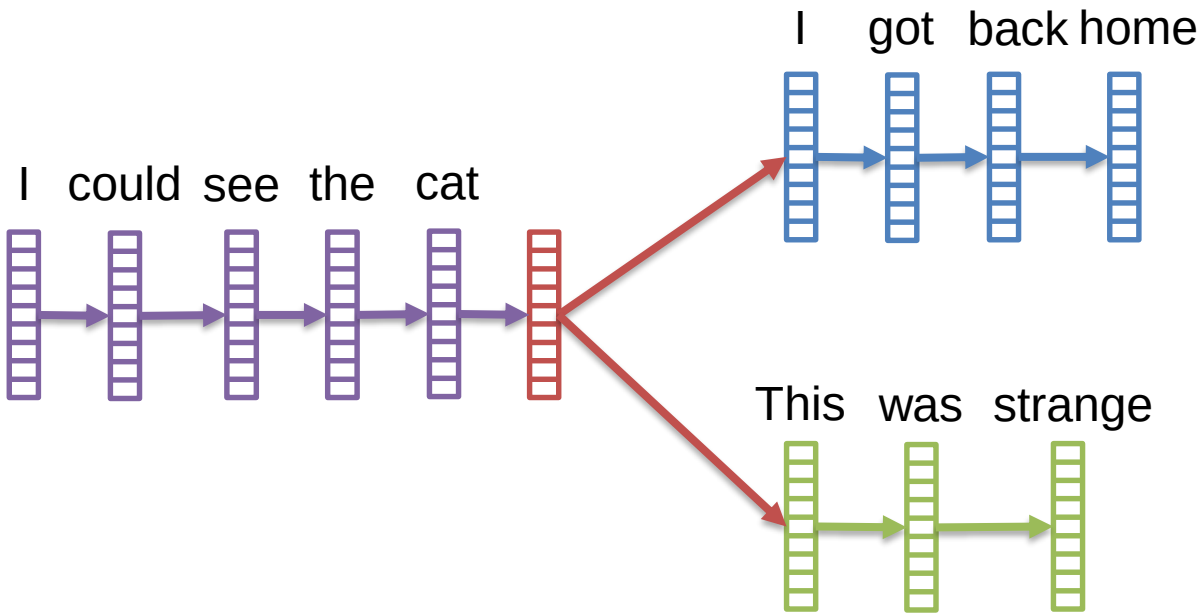


- Since the model ignores the contextual words at the input layer, it is named the distributed bag-of-words (DBOW) model

R. Kiros, *et al*., "Skip-thought vectors," arXiv:1506.06726, 2015.

Skip-Thought Vector Model

- The skip-thought vector model presents an objective function that abstracts the **skip-gram** model to the sentence level
  - Instead of using a word to predict its surrounding context, thought vector encodes a sentence to predict the sentences around it



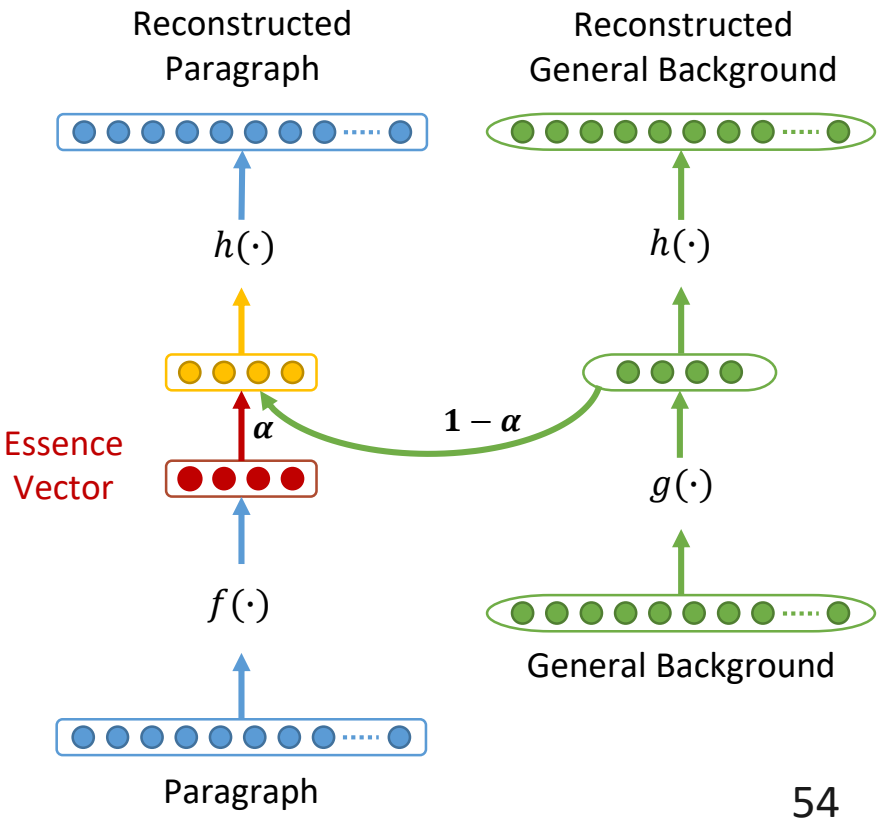
Classic Paragraph Embedding Methods

- Classic paragraph embedding methods infer the representation of a given paragraph by **considering all of the words** occurring in the paragraph
  - Such as the Distributed Memory model, the Distributed Bag-of-words model, and the Skip-Thought Vector model

- The **stop** or **function words** that occur frequently may mislead the embedding learning process
  - The learned representation for the paragraph might be undesired
  - The performance is limited
  - Our goal is to
    - Distill the most representative information from a given paragraph
    - Get rid of the general background information

Learning to Distill

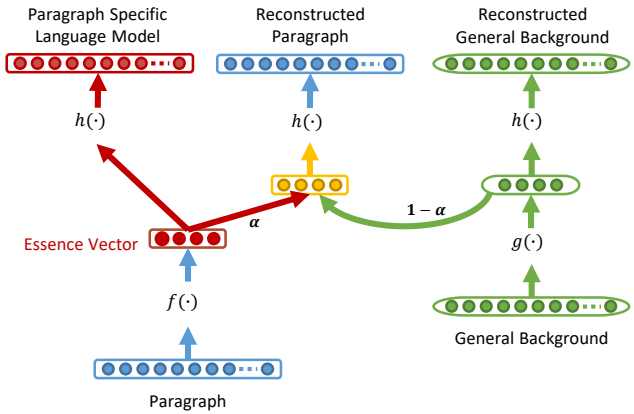
- We assume that each paragraph can be assembled by the **paragraph specific information** and the **general background information**
  - This assumption also holds in the low-dimensional representation space
  - Three modules
    - Paragraph encoder $f(\cdot)$
    - Background encoder $g(\cdot)$
    - Decoder $h(\cdot)$



Essence Vector-based Language Model

- A brilliant property inherits in the EV model is that it can be readily inferred a "paragraph" specific language model

$$\hat{P}(w) \equiv h\left(f\left(P_{D_t}\right)\right)$$



Paragraph Specific Language Model

Original Paragraph Language Model

Background Language Model

Questions?

kychen@mail.ntust.edu.tw