# Bridge Deep Learning to Recommender Systems

Presenter: Liang Hu

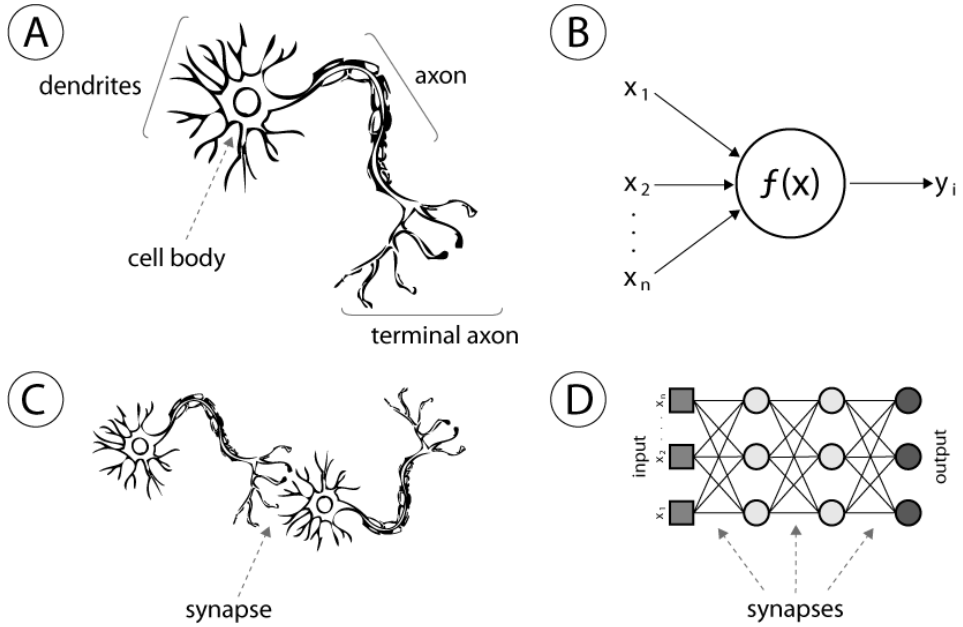
# Outline

- Preliminary of Artificial Neural Networks
- A Brief Review of Deep Learning
- Deep Neural Models Based Recommender Systems
- Attention Mechanism

- Preliminary of Artificial Neural Networks
- A Brief Review of Deep Learning
- Deep Neural Models Based Recommender Systems
- Attention Mechanism

# Artificial neural networks

- Artificial neural networks (ANN) are computing systems inspired by the biological neural networks that constitute animal brains.
- Warren McCulloch and Walter Pitts (1943) opened the subject by creating a computational model for neural networks.

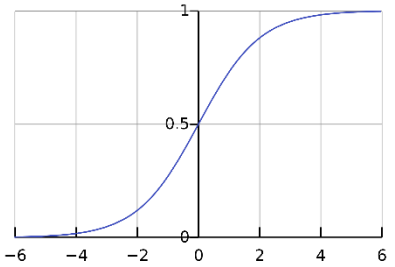


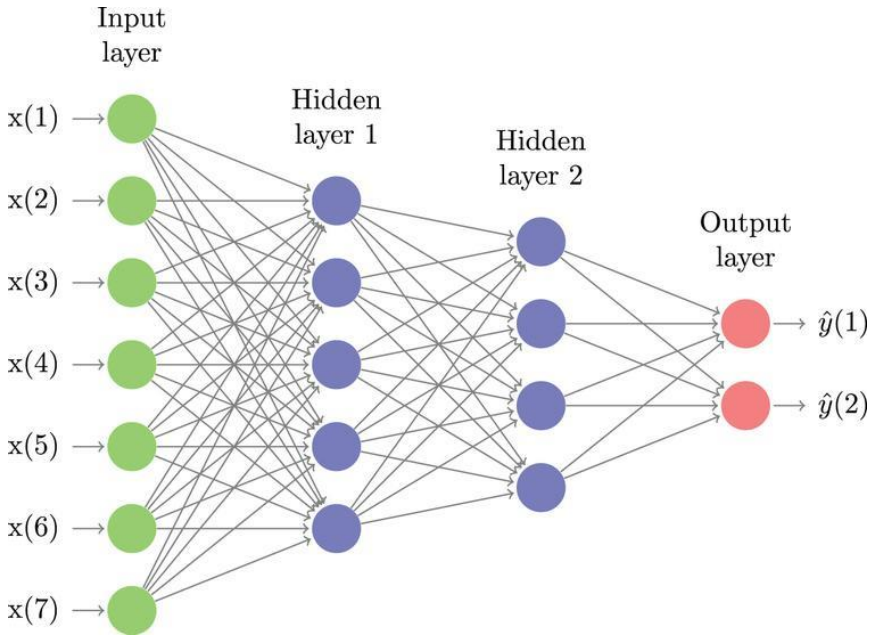
# Neural network model

$$\mathbf{h}_2 = \text{sigmoid}(\mathbf{W}_1\mathbf{h}_1 + \mathbf{b}_1)$$

- Input layer: $\mathbf{x}$
- Hidden layers:
  - $\mathbf{h}_1 = f_{\mathbf{W}_1}(\mathbf{x}) = f_{\mathbf{W}_1}(\mathbf{h}_0)$, where $f(\cdot)$ is activation function
  - $\mathbf{h}_2 = f_{\mathbf{W}_2}(\mathbf{h}_1)$
  - $\mathbf{h}_{n+1} = f_{\mathbf{W}_{n+1}}(\mathbf{h}_n)$

    .
    .
    .
- Output layer:
  - $\hat{\boldsymbol{y}} = f_{\mathbf{W}_N}(\mathbf{h}_{N-1}) = f_{\mathbf{W}_N}\left(\cdots\left(f_{\mathbf{W}_2}\left(f_{\mathbf{W}_1}(\boldsymbol{x})\right)\right)\right)$
- Error function:
  - $L(\boldsymbol{y}, \hat{\boldsymbol{y}}) = L\left(\boldsymbol{y}, f_{\mathbf{W}_N}\left(\cdots\left(f_{\mathbf{W}_2}\left(f_{\mathbf{W}_1}(\boldsymbol{x})\right)\right)\right)\right)$



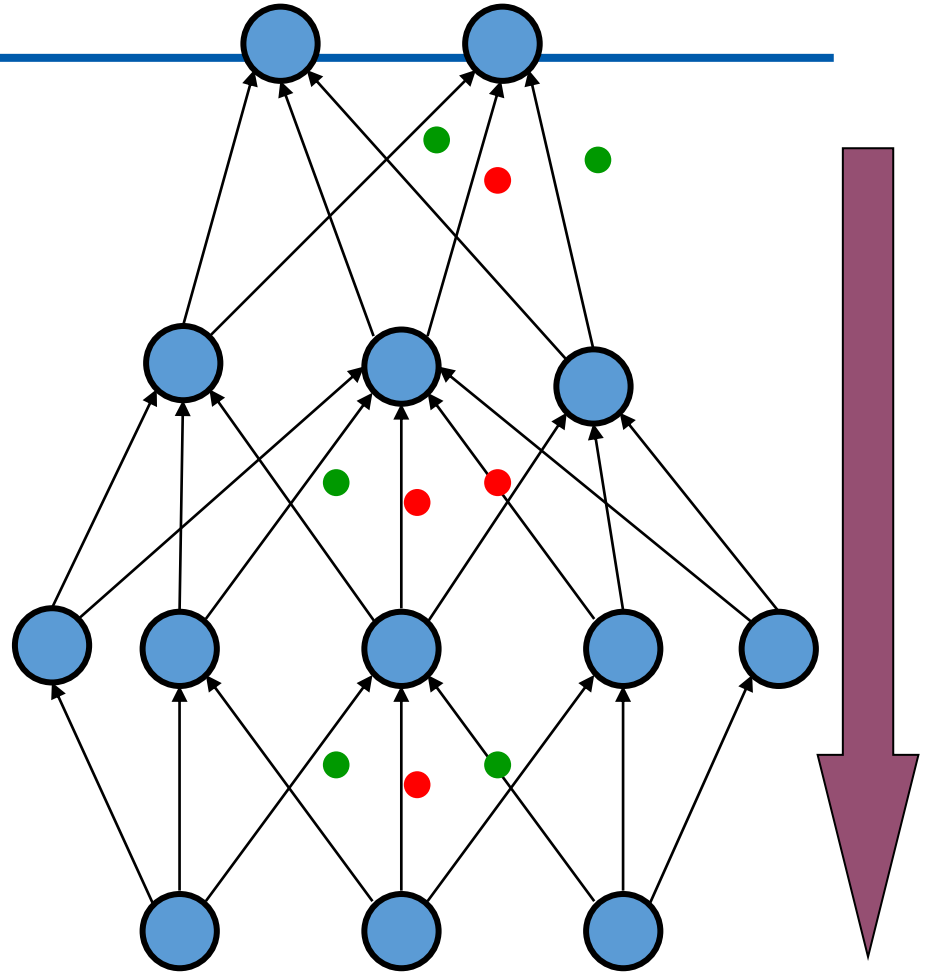
# Gradient for each layer

- $L(\boldsymbol{y}, \hat{\boldsymbol{y}}) = L\left(\boldsymbol{y}, f_{\mathbf{W}_N}\left(\cdots\left(f_{\mathbf{W}_2}\left(f_{\mathbf{W}_1}(\boldsymbol{x})\right)\right)\right)\right)$
- $\frac{\partial L}{\partial \boldsymbol{W_N}} = \underbrace{\frac{\partial L}{\partial f_{\boldsymbol{W_N}}(\mathbf{h}_{N-1})}}_{\delta_N} \frac{\partial f_{\boldsymbol{W_N}}(\mathbf{h}_{N-1})}{\partial \boldsymbol{W_N}}$
- $\frac{\partial L}{\partial \boldsymbol{W_{N-1}}} = \underbrace{\frac{\partial L}{\partial f_{\boldsymbol{W_N}}(\mathbf{h}_{N-1})} \frac{\partial f_{\boldsymbol{W_N}}\left(f_{\boldsymbol{W_{N-1}}}(\mathbf{h}_{N-2})\right)}{\partial f_{\boldsymbol{W_{N-1}}}(\mathbf{h}_{N-2})}}_{\delta_{N-1}} \frac{\partial f_{\boldsymbol{W_{N-1}}}(\mathbf{h}_{N-2})}{\partial \boldsymbol{W_{N-1}}} = \delta_N \frac{\partial f_{\boldsymbol{W_N}}\left(f_{\boldsymbol{W_{N-1}}}(\mathbf{h}_{N-2})\right)}{\partial f_{\boldsymbol{W_{N-1}}}(\mathbf{h}_{N-2})} \frac{\partial f_{\boldsymbol{W_{N-1}}}(\mathbf{h}_{N-2})}{\partial \boldsymbol{W_{N-1}}}$
- $\frac{\partial L}{\partial \boldsymbol{W_{N-2}}} = \underbrace{\frac{\partial L}{\partial f_{\boldsymbol{W_N}}(\mathbf{h}_{N-1})} \frac{\partial f_{\boldsymbol{W_N}}\left(f_{\boldsymbol{W_{N-1}}}(\mathbf{h}_{N-2})\right)}{\partial f_{\boldsymbol{W_{N-1}}}(\mathbf{h}_{N-2})} \frac{\partial f_{\boldsymbol{W_{N-1}}}\left(f_{\boldsymbol{W_{N-2}}}(\mathbf{h}_{N-3})\right)}{\partial f_{\boldsymbol{W_{N-2}}}(\mathbf{h}_{N-3})}}_{\delta_{N-2}} \frac{\partial f_{\boldsymbol{W_{N-2}}}(\mathbf{h}_{N-3})}{\partial \boldsymbol{W_{N-2}}} = \delta_{N-1} \frac{\partial f_{\boldsymbol{W_{N-1}}}\left(f_{\boldsymbol{W_{N-2}}}(\mathbf{h}_{N-3})\right)}{\partial f_{\boldsymbol{W_{N-2}}}(\mathbf{h}_{N-3})} \frac{\partial f_{\boldsymbol{W_{N-2}}}(\mathbf{h}_{N-3})}{\partial \boldsymbol{W_{N-2}}}$
- $\frac{\partial L}{\partial \boldsymbol{W_n}} = \underbrace{\frac{\partial L}{\partial f_{\boldsymbol{W_N}}(\mathbf{h}_{N-1})} \frac{\partial f_{\boldsymbol{W_N}}\left(f_{\boldsymbol{W_{N-1}}}(\mathbf{h}_{N-2})\right)}{\partial f_{\boldsymbol{W_{N-1}}}(\mathbf{h}_{N-2})} \frac{\partial f_{\boldsymbol{W_{N-1}}}\left(f_{\boldsymbol{W_{N-2}}}(\mathbf{h}_{N-3})\right)}{\partial f_{\boldsymbol{W_{N-2}}}(\mathbf{h}_{N-3})} \cdots \frac{\partial f_{\boldsymbol{W_{n+1}}}\left(f_{\boldsymbol{W_n}}(\mathbf{h}_{n-1})\right)}{\partial f_{\boldsymbol{W_n}}(\mathbf{h}_{n-1})}}_{\delta_n} \frac{\partial f_{\boldsymbol{W_n}}(\mathbf{h}_{n-1})}{\partial \boldsymbol{W_n}}$

# Back-propagation

- The term backpropagation and its general use in neural networks was announced in Rumelhart, Hinton & Williams

- In the 1990's, many researchers abandoned neural networks with multiple adaptive hidden layers.



Back-propagate error signal to get derivatives for learning

Rumelhart, David E.; Hinton, Geoffrey E.; Williams, Ronald J. (8 October 1986). "Learning representations by back-propagating errors". Nature. 323 (6088)

# What's the issue with back-propagation (1)?

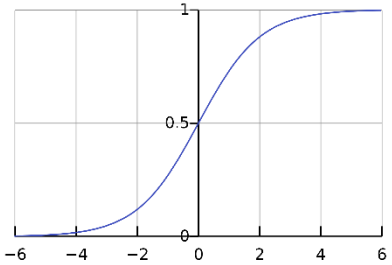
- The learning time does not scale well. It is very slow in networks with multiple hidden layers.
  - For $N$ fully connected layers, the number of parameters is
$$\sum_{n=1}^{N} |\mathbf{h}_n| \times |\mathbf{h}_{n-1}|$$
  - For 100 layers and 200 hidden units for each layer, we have 400,000,000 parameters

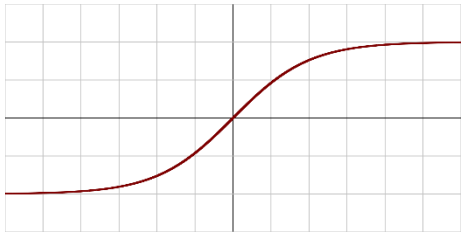
# What's the issue with back-propagation (2)?

- It can get stuck in poor local optima because of gradient vanishing.

$$\frac{\partial L}{\partial \boldsymbol{W_n}} = \frac{\partial L}{\partial f_{\boldsymbol{W_N}}(\mathbf{h}_{N-1})} \frac{\partial f_{\boldsymbol{W_N}}\left(f_{\boldsymbol{W_{N-1}}}(\mathbf{h}_{N-2})\right)}{\partial f_{\boldsymbol{W_{N-1}}}(\mathbf{h}_{N-2})} \cdots \frac{\partial f_{\boldsymbol{W_{n+1}}}\left(f_{\boldsymbol{W_n}}(\mathbf{h}_{n-1})\right)}{\partial f_{\boldsymbol{W_n}}(\mathbf{h}_{n-1})} \frac{\partial f_{\boldsymbol{W_n}}(\mathbf{h}_{n-1})}{\partial \boldsymbol{W_n}}$$

Normally, sigmoid or tanh is often used as the activation $f_{\boldsymbol{W}}$



- This has the effect of multiplying n of these small numbers to compute gradients of the "front" layers in an n-layer network, meaning that the gradient (error signal) decreases exponentially.

# Outline

- Preliminary of Artificial Neural Networks
- A Brief Review of Deep Learning
- Deep Neural Models Based Recommender Systems
- Attention Mechanism

# Neural Networks VS. Support Vector Machines

**Geoffrey Hinton**

- 1957: The first neural networks (perceptron) algorithm
- 1986: The invention of Backpropagation
- 1992~2005: The cold winter of ANN
- 2006: The birth of Deep Learning

**Bernhard Schölkopf; Isabelle Guyon, Vladimir Vapnik**

- 1963 : The original SVM algorithm was invented
- 1993: SVM employs kernel tricks and maximal margin to perform better in non-linear tasks
- Schölkopf: Vapnik wanted to kill ANN

https://medium.com/analytics-vidhya/the-scuffle-between-two-algorithms-neural-network-vs-support-vector-machine-16abe0eb4181

# Major Breakthrough in 2006

- Ability to train deep architectures by using layer-wise unsupervised learning, whereas previous purely supervised attempts had failed.
- Hinton: I want to call SVM shallow learning

Geoffrey Hinton

Joshua Bengio

- Unsupervised feature learners:
  - Restricted Boltzmann Machines
  - Auto-encoder variants

Hinton, G. E., & Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. *Science*, *313*(5786), 504-507.

# Deep architecture in the human brain

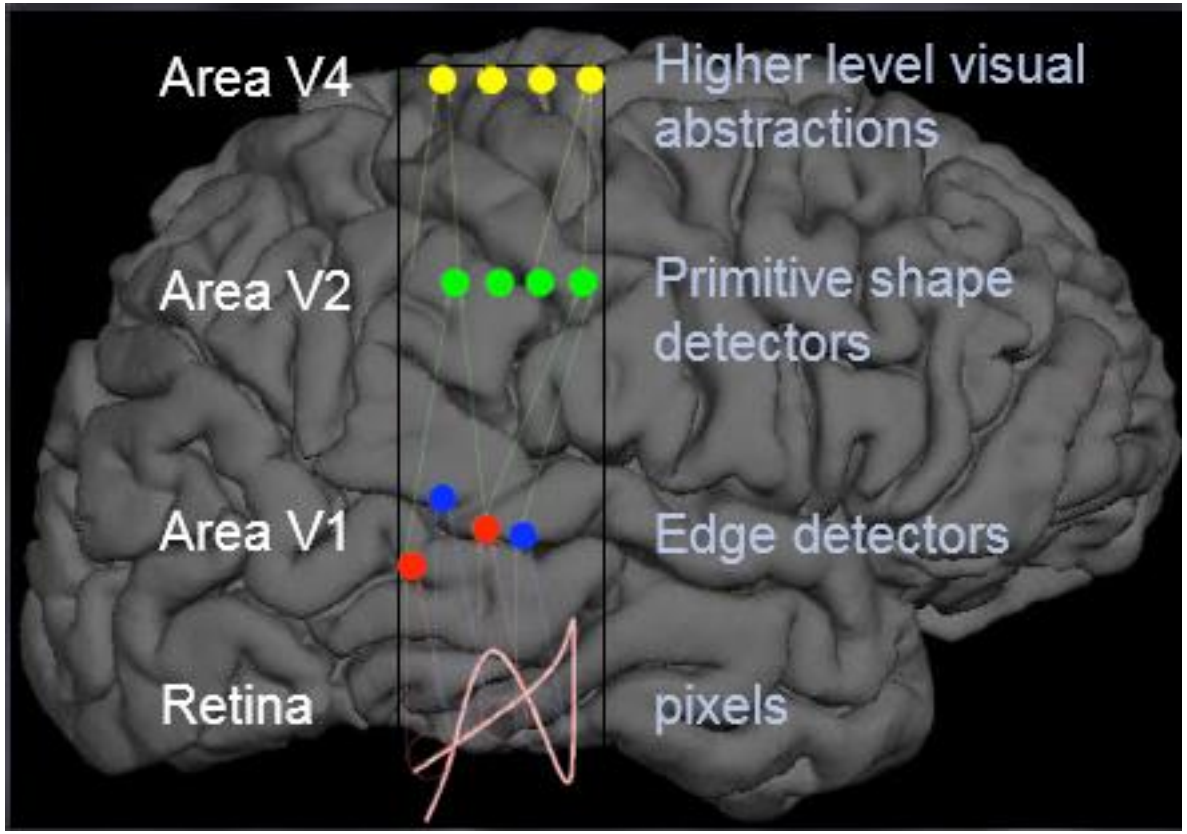
- Visual System

  Input:
  - Data: Pixels

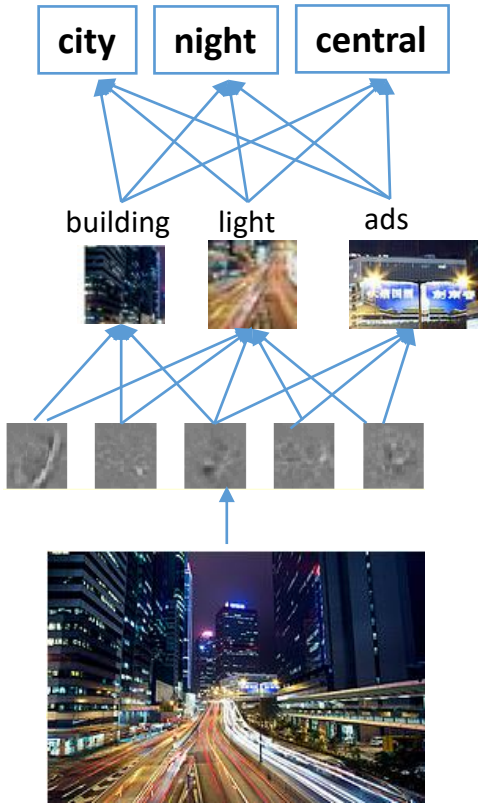
  1st Layer:
  - Features: Edges

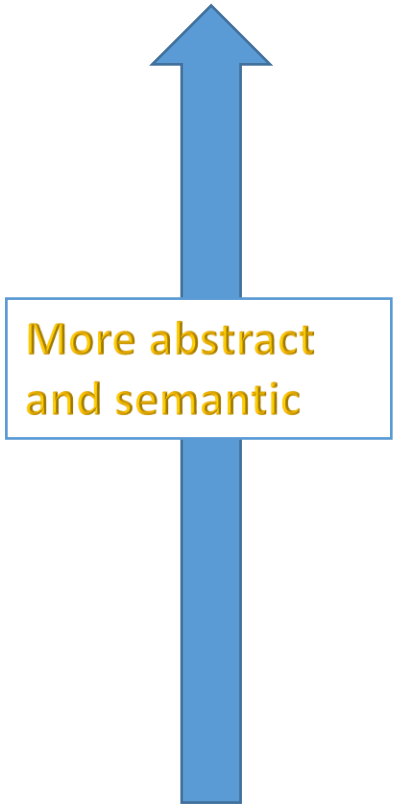
  2nd Layer:
  - Features: Shapes

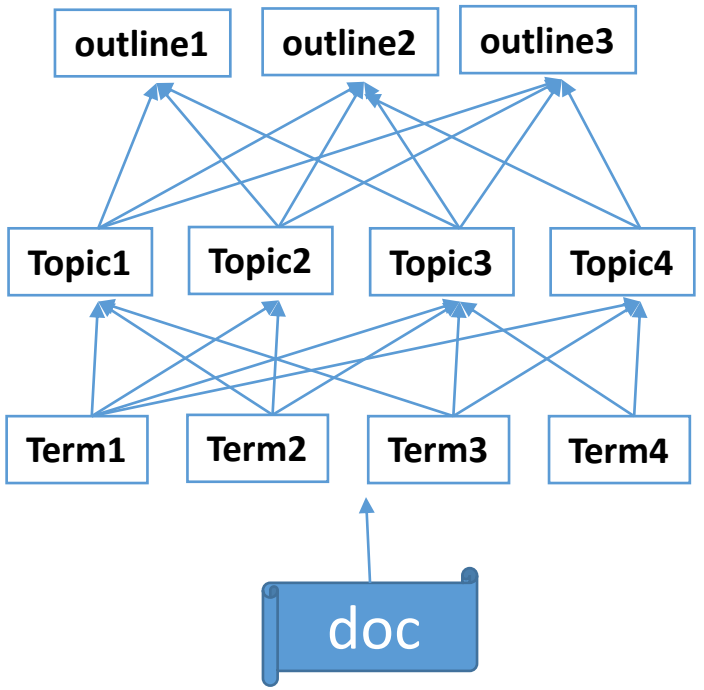
  3rd Layer:
  - Features: Visual Abstraction



# Learning high-level features



# Abstraction and Invariance

- Deep architectures can lead to abstract representations because ***more abstract concepts*** can often be constructed in terms of ***less abstract ones***.

- More abstract concepts are generally ***invariant*** to most local changes of the input. That makes the representations that capture these concepts generally highly non-linear functions of the raw input.

“研表究明，汉字序顺并不定一影阅响读。比如当你看完这句话后，才发这现里的字全是都乱的。”

## How does our brain perform tasks?

1. Our brain learns high-level knowledge (representation) $\boldsymbol{h}$ given any input $\boldsymbol{x}$

2. When we conduct any task, we first search the knowledge $\boldsymbol{h}$ from our memory that is similar to the input $\boldsymbol{x}_{new}$

3. We adapt our knowledge $\boldsymbol{h}$ to fit the new task instead of learning $\boldsymbol{x}_{new}$ from scratch

# Shallow learning VS deep learning

- Shallow learning
  - Manual features, and a direct map between features and a specific task:

$$P(task|features) \quad P(features|data)$$

  - E.g. Linear regression and support vector machines (SVM).

- Deep learning
  - Learning features (representation) from data:
    - A multi-layer model to learn higher-level features from low-level features

$$P(\mathbf{h}_4, \mathbf{h}_3, \mathbf{h}_2, \mathbf{h}_1|data) = P(\mathbf{h}_4|\mathbf{h}_3)\ P(\mathbf{h}_3|\mathbf{h}_2)\ P(\mathbf{h}_2|\mathbf{h}_1)\ P(\mathbf{h}_1|data)$$

  - Learning tasks based on the high-level features output by the representation model

$$P(task1|\mathbf{h}_4)P(\mathbf{h}_4|\mathbf{h}_3)P(\mathbf{h}_3|\mathbf{h}_2)P(\mathbf{h}_2|\mathbf{h}_1)P(\mathbf{h}_1|data)$$
$$P(task2|\mathbf{h}_4)P(\mathbf{h}_4|\mathbf{h}_3)P(\mathbf{h}_3|\mathbf{h}_2)P(\mathbf{h}_2|\mathbf{h}_1)P(\mathbf{h}_1|data)$$

# Layer-wise unsupervised pre-training

$$P(task|\mathbf{h}_4)P(\mathbf{h}_4|\mathbf{h}_3)P(\mathbf{h}_3|\mathbf{h}_2)P(\mathbf{h}_2|\mathbf{h}_1)P(\mathbf{h}_1|data)$$

- To better model $P(task|\mathbf{h}_4)$, we first need to better learn features from data

$$P(\mathbf{h}_4|\mathbf{h}_3)P(\mathbf{h}_3|\mathbf{h}_2)P(\mathbf{h}_2|\mathbf{h}_1)P(\mathbf{h}_1|data)$$

- That is, we can learn an optimized representation of data using a multi-layer model.
- Simulating the layer-wise learning of brains, we can train deep architectures by using layer-wise unsupervised learning.
  - Restricted Boltzmann machines (RBM)
  - Auto Encoder (AE)

# Restricted Boltzmann machines

- Energy function:

$$E(\mathbf{v}, \mathbf{h}) = -\left(\mathbf{v}^{\mathrm{T}}\mathbf{W}\mathbf{h} + \mathbf{a}^{\mathrm{T}}\mathbf{v} + \mathbf{b}^{\mathrm{T}}\mathbf{h}\right)$$

- Probability distribution:

$$P(\mathbf{v}) = \sum_{\mathbf{h}} P(\mathbf{v}, \mathbf{h}) = \frac{e^{-\sum_{\mathbf{h}} E(\mathbf{v},\mathbf{h})}}{Z}, \qquad \mathrm{Z} = e^{-\sum_{\mathbf{v},\mathbf{h}} E(\mathbf{v},\mathbf{h})}$$

- Objective:

$$\arg\max_{\mathbf{W},\mathbf{a},\mathbf{b}} \log P(\mathbf{v})$$

$$\frac{\partial \log P(\mathbf{v})}{\partial \boldsymbol{W}} = \langle \mathbf{v}, \mathbf{h} \rangle_{\mathrm{data}} + \langle \mathbf{v}', \mathbf{h}' \rangle_{\mathrm{model}}$$

- Gibbs sampling:
  - Sampling $h_j$ given $\mathbf{v}$ with the probability $P\left(h_j|\mathbf{v}\right)$

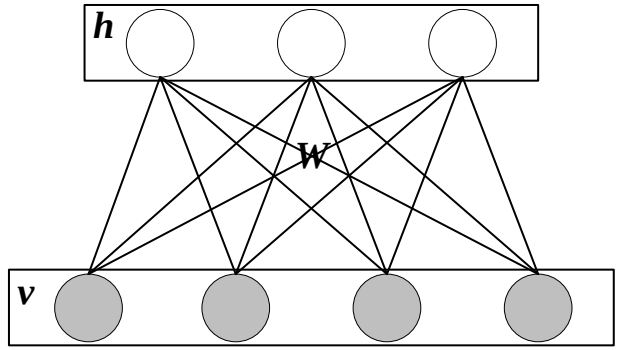
$$P\left(h_j = 1|\mathbf{v}\right) = \mathrm{sigmoid}\left(\mathbf{v}^{\mathrm{T}}\mathbf{W}_{:,j} + b_j\right)$$

  - Sampling $v_i'$ given $\mathbf{h}$ with the probability $P(v_i|\mathbf{h})$

$$P(v_i' = 1|\mathbf{h}) = \mathrm{sigmoid}\left(\mathbf{W}_{i,:}\,\mathbf{h} + a_i\right)$$

  - Sampling $h_j'$ given $\mathbf{v}'$ with the probability $P\left(h_j'|\mathbf{v}'\right)$

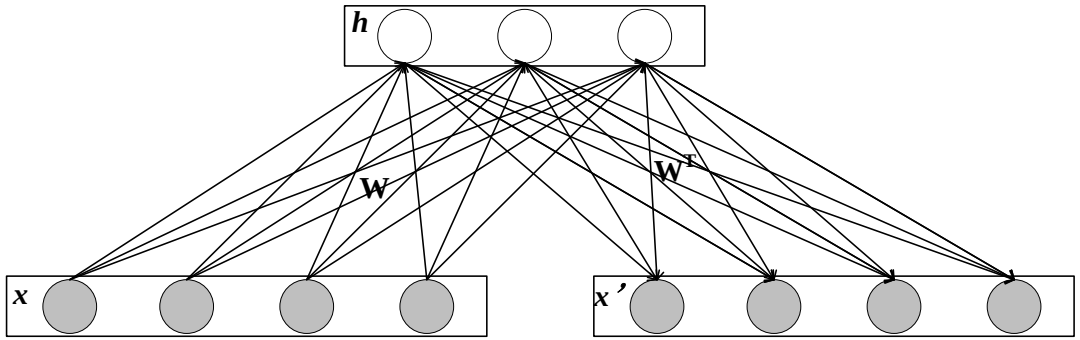
$$P\left(h_j' = 1|\mathbf{v}'\right) = \mathrm{sigmoid}\left(\mathbf{v}'^{\mathrm{T}}\mathbf{W}_{:,j} + b_j\right)$$



# Autoencoder

- Autoencoders are trained to minimize reconstruction errors (such as squared errors):

- $L(\boldsymbol{x}, \boldsymbol{x}') = \|\boldsymbol{x} - \boldsymbol{x}'\|^2$
  $= \left\|\boldsymbol{x} - \left(\mathbf{W}^{\mathrm{T}}\mathbf{h} + \mathbf{b}'\right)\right\|^2$
  1. Encoding: $\mathbf{h} = \mathrm{sigmoid}(\mathbf{W}\boldsymbol{x} + \mathbf{b})$
  2. Decoding: $\boldsymbol{x}' = \left(\mathbf{W}^{\mathrm{T}}\mathbf{h} + \mathbf{b}'\right)$



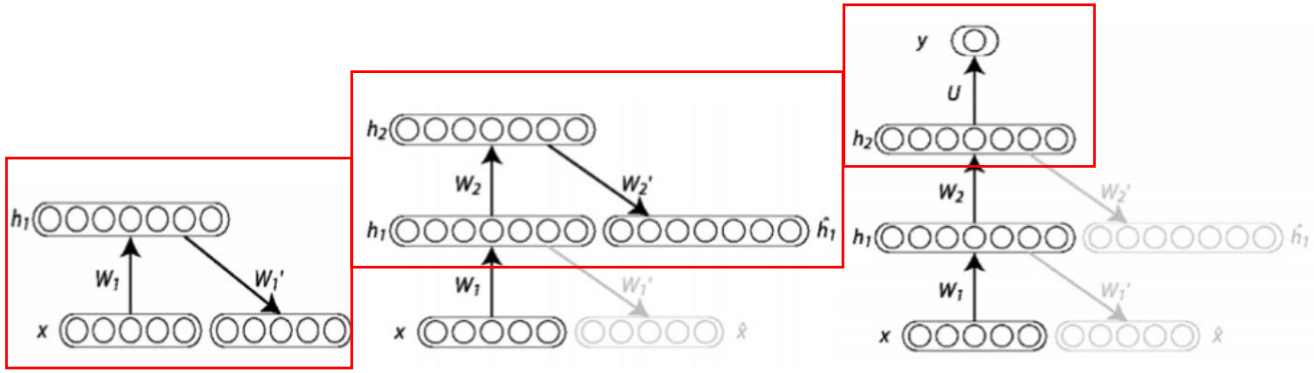
# Example: unsupervised pre-training + supervised fine tuning

1. Greedy layer-wise learning using autoencoders.

2. Initialized a deep neural network with the learning weights.

3. Build a supervised model on the top of deep neural network.

4. Using the label data for fine tuning through backpropagation.




Bengio, Y., Lamblin, P., Popovici, D., & Larochelle, H. (2007). Greedy layer-wise training of deep networks. *NIPS*, *19*, 153.

# Summary: the success of deep learning

1. Unsupervised representation learning (*layer-wise learning*).

$$P_{\mathbf{W_4}}(\mathbf{h}_4|\mathbf{h}_3)\ P_{\mathbf{W_3}}(\mathbf{h}_3|\mathbf{h}_2)\ P_{\mathbf{W_2}}\ (\mathbf{h}_2|\mathbf{h}_1)\ P_{\mathbf{W_1}}(\mathbf{h}_1|data)$$

2. Task-specific learning (*supervised fine-tuning*).

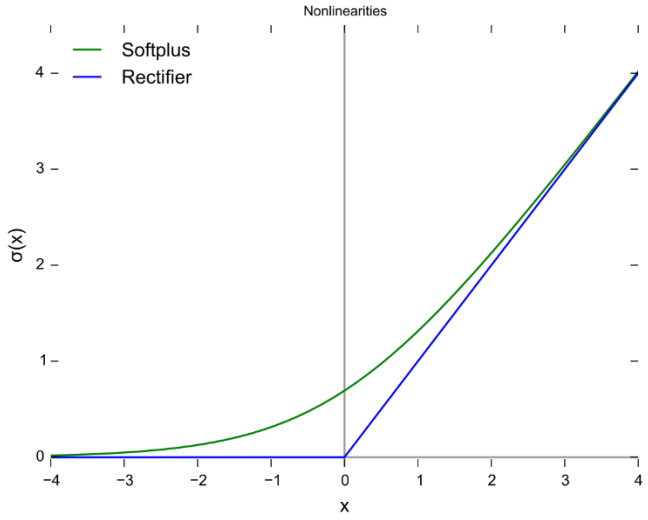
$$P_{\mathbf{W}_t}(task|\mathbf{h}_4)P_{\mathbf{W_4}}(\mathbf{h}_4|\mathbf{h}_3)P_{\mathbf{W_3}}(\mathbf{h}_3|\mathbf{h}_2)P_{\mathbf{W_2}}(\mathbf{h}_2|\mathbf{h}_1)P_{\mathbf{W_1}}(\mathbf{h}_1|data)$$

# Supervised learning without pre-training (1)

- Rectified Linear Unit (ReLU)

$$f(x) = x^{+} = \max(0, x)$$

- The activation ReLU suffers less from the vanishing gradient problem, because they only saturate in one direction.



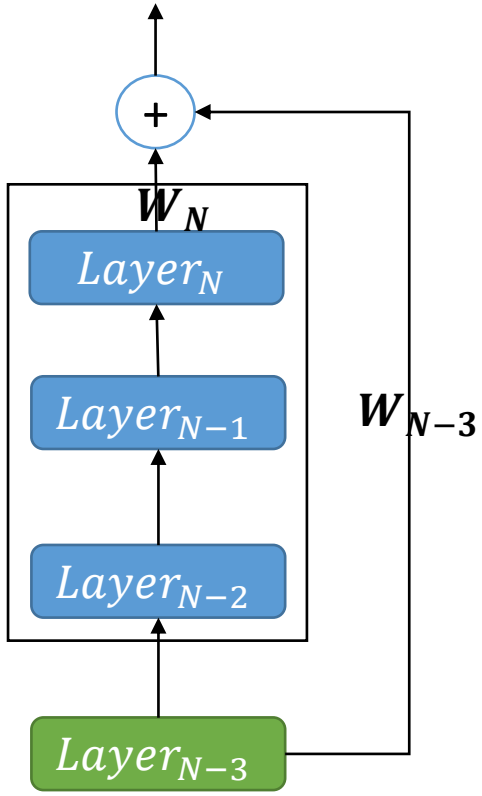
# Supervised learning without pre-training (2)

- Residual Networks (ResNets)

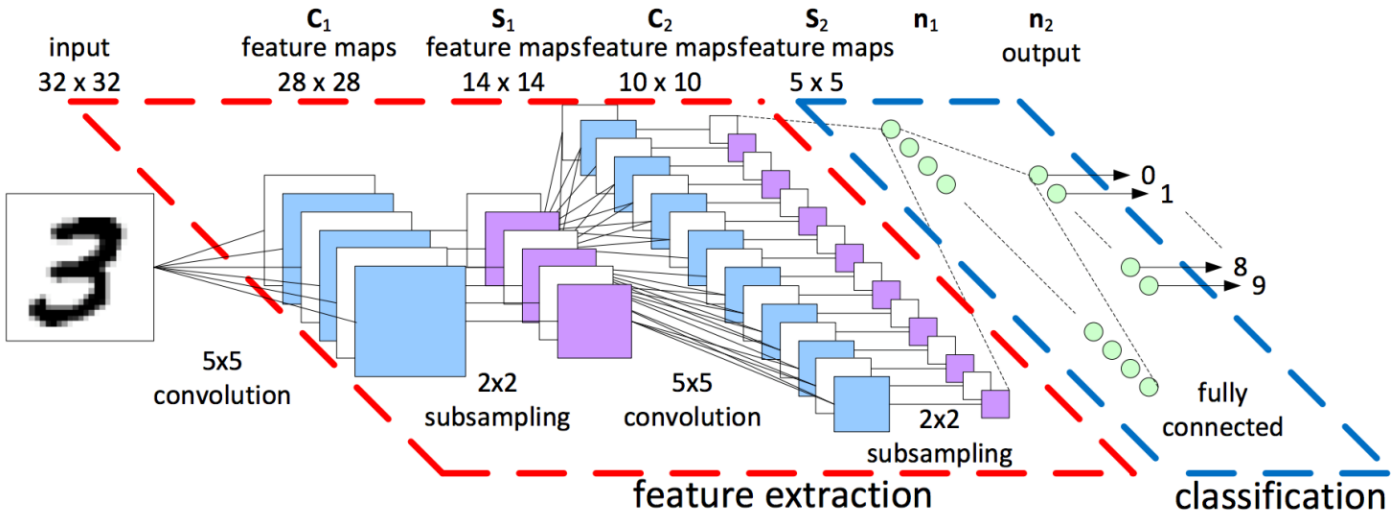
$$\mathbf{h}_{N+1} = f_{\mathbf{W}_N}(\mathbf{h}_N) + f_{\mathbf{W}_{N-3}}(\mathbf{h}_{N-3})$$

$$\frac{\partial L}{\partial \boldsymbol{W}_{\boldsymbol{N-3}}} = \delta_{N+1} \frac{\partial f_{\boldsymbol{W}_{\boldsymbol{N-3}}}(\mathbf{h}_{N-3})}{\partial \boldsymbol{W}_{\boldsymbol{N-3}}}$$

- ResNets do not resolve the vanishing gradient problem by preserving gradient flow throughout the entire depth of the network
- ResNets avoid the problem simply by constructing ensembles of many short networks together.



# Convolutional neural networks




Yann LeCun (杨立昆)

LeCun, Y.; Boser, B.; Denker, J. S.; Henderson, D.; Howard, R. E.; Hubbard, W. & Jackel, L. D. (1989). Backpropagation applied to handwritten zip code recognition. Neural Computation, 1(4).

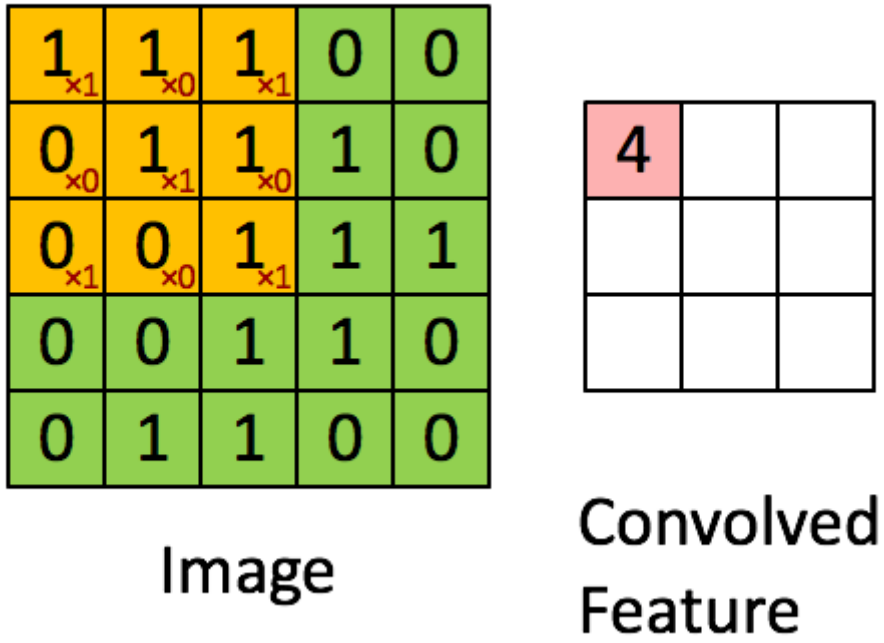
# Machine bias



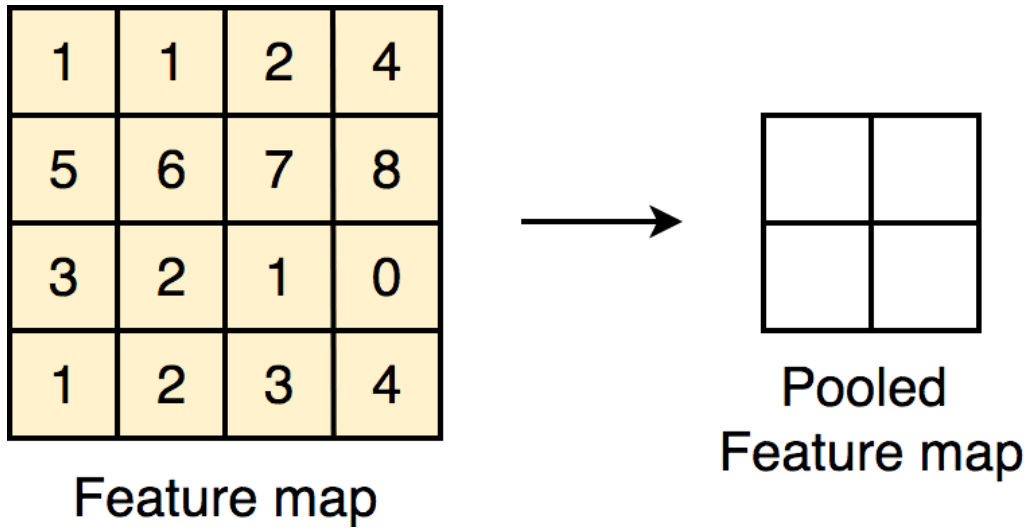

https://36kr.com/p/772591048947202

# Core components of CNN

- Convolution

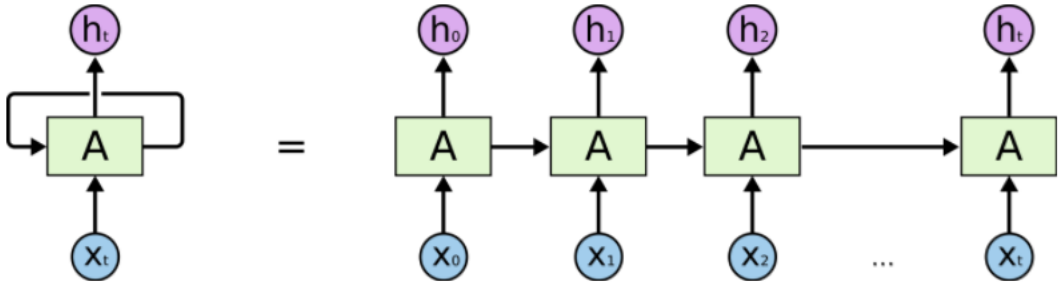



- Subsampling (Pooling)



# Recurrent neural networks (RNN)

$$\begin{aligned}\mathbf{h}_t &= f_{\mathbf{W}}(\mathbf{x}_t, \mathbf{h}_{t-1}) \\ &= f_{\mathbf{W}}\left(\mathbf{x}_t, f_{\mathbf{W}}(\mathbf{x}_{t-1}, \mathbf{h}_{t-2})\right) \\ &= f_{\mathbf{W}}\left(\mathbf{x}_t, f_{\mathbf{W}}\left(\mathbf{x}_{t-1}, f_{\mathbf{W}}(\mathbf{x}_{t-2}, \mathbf{h}_{t-3})\right)\right) \\ &= f_{\mathbf{W}}\left(\mathbf{x}_t, f_{\mathbf{W}}\left(\mathbf{x}_{t-1}, \cdots, f_{\mathbf{W}}(\mathbf{x}_0)\right)\right)\end{aligned}$$

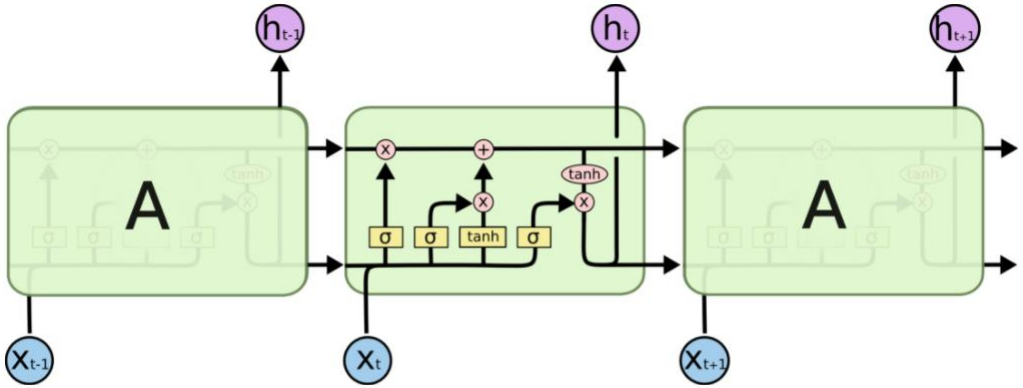



An unrolled recurrent neural network.

- Gradient vanishing and exploding problems occur when learning this naïve RNN

# Long short-term memory (LSTM) networks

- LSTM models the long and short-term dependencies, where the accumulation of information is controlled by gate modules



The repeating module in an LSTM contains four interacting layers.

- LSTM units allow gradients to also flow *unchanged*. However, LSTM networks can still suffer from the exploding gradient problem.


Jürgen Schmidhuber

Sepp Hochreiter; Jürgen Schmidhuber (1997). "Long short-term memory". Neural Computation. 9 (8): 1735–1780.

# Views of deep learning

- Schmidhuber complained in a "scathing 2015 article" that fellow deep learning researchers Geoffrey Hinton, Yann LeCun and Yoshua Bengio "heavily cite each other," but "fail to credit the pioneers of the field," allegedly understating the contributions of Schmidhuber and other early machine learning pioneers including Alexey Grigorevich Ivakhnenko who published the first deep learning networks already in 1965.

- LeCun denies the charge, stating instead that Schmidhuber "keeps claiming credit he doesn't deserve".


**Deep Learning: Our Miraculous Year 1990-1991**

http://people.idsia.ch/~juergen/deep-learning-miraculous-year-1990-1991.html

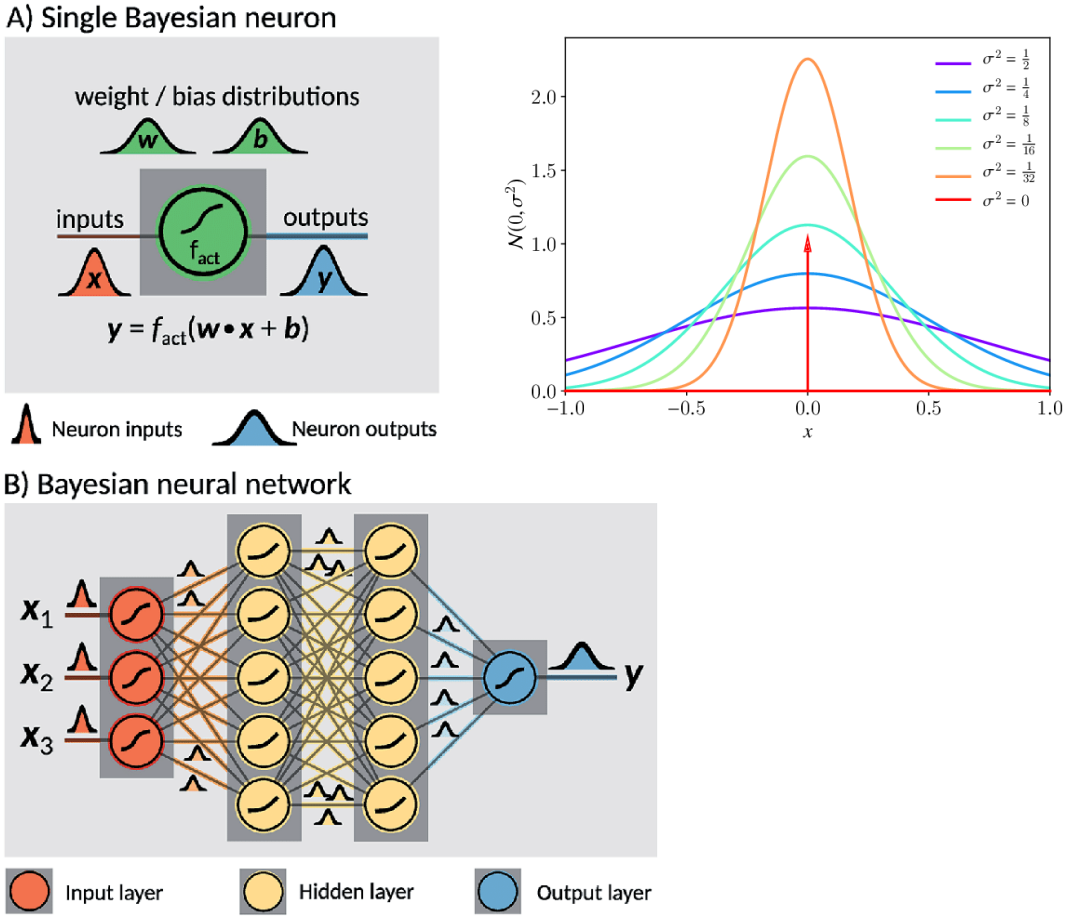
# Who are the heroes behind the renaissance of ANN?

- Faster hardware
  - The computer power (especially as delivered by GPUs) has increased around a million-fold from 1991 to 2015, the original models were trained in CPUs (e.g. Xeon processor), not GPUs.

- Big data
  - The rapid growing quantities of data makes it possible to train a deep neural networks with millions of parameters
  - The number of data types is rapidly growing, including text, audio, video from millions of possible sources, which enables to train various types of deep models, including DNN, CNN, RNN.

# Other direction: probabilistic deep learning

- Probabilistic approach to deep learning becomes increasingly popular
- The core benefit of using a Bayesian framework is being able to propagate *uncertainty* to improve prediction and model reasoning.
- Recent advances for Bayesian deep learning provide improvements in accuracy and calibration while retaining scalability.



# Deep learning is tailored for recommender systems

- DL is the SOTA technique to build AI systems


- RSs are the most widely used AI systems in our daily life

- DL models are build on large quantity of data for prediction


- RSs aim to predict demands based on large quantity of data

- DL can consume multiple types of data, using DNN, CNN, RNN, GNN, etc.

- RSs need to deal with multiple types of data, such as attributes, text, images, relationships

# Outline

- Preliminary of Artificial Neural Networks
- A Brief Review of Deep Learning
- Deep Neural Models Based Recommender Systems
- Attention Mechanism

# Embedding

- Why learn embedding?
  - For example, NLP systems traditionally treat words as discrete symbols
    - E.g. ‘cats‘: id 22, 'dogs‘: id 23, while they are both animals, four-legged, etc.
  - Using vector representations can overcome some of these obstacles

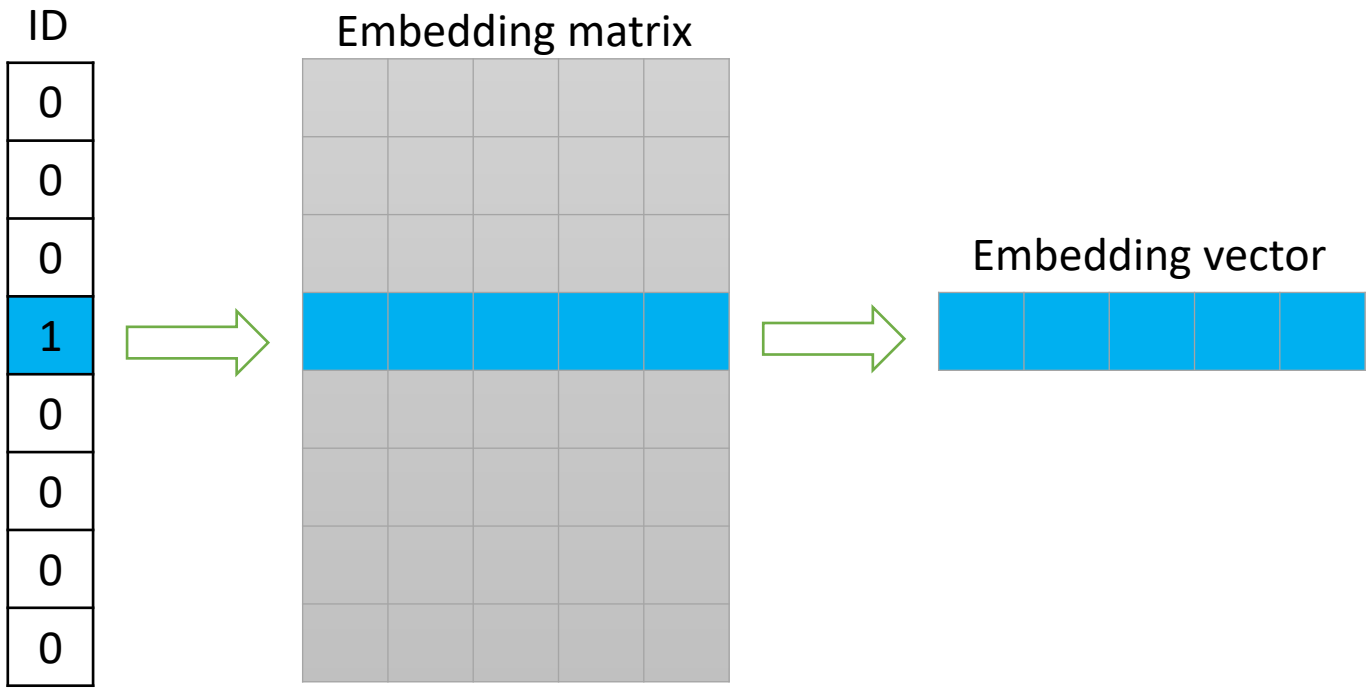
- An **embedding** is a relatively low-dimensional space into which you can translate high-dimensional vectors.

- An embedding captures some of the semantics of the input by placing semantically similar inputs close together in the embedding space.

# Retrieve embedding vector

- Given $N$ objects, we map each object ID to an embedding vector:

$$ID \rightarrow \mathbb{R}^K$$



# Movie and user features

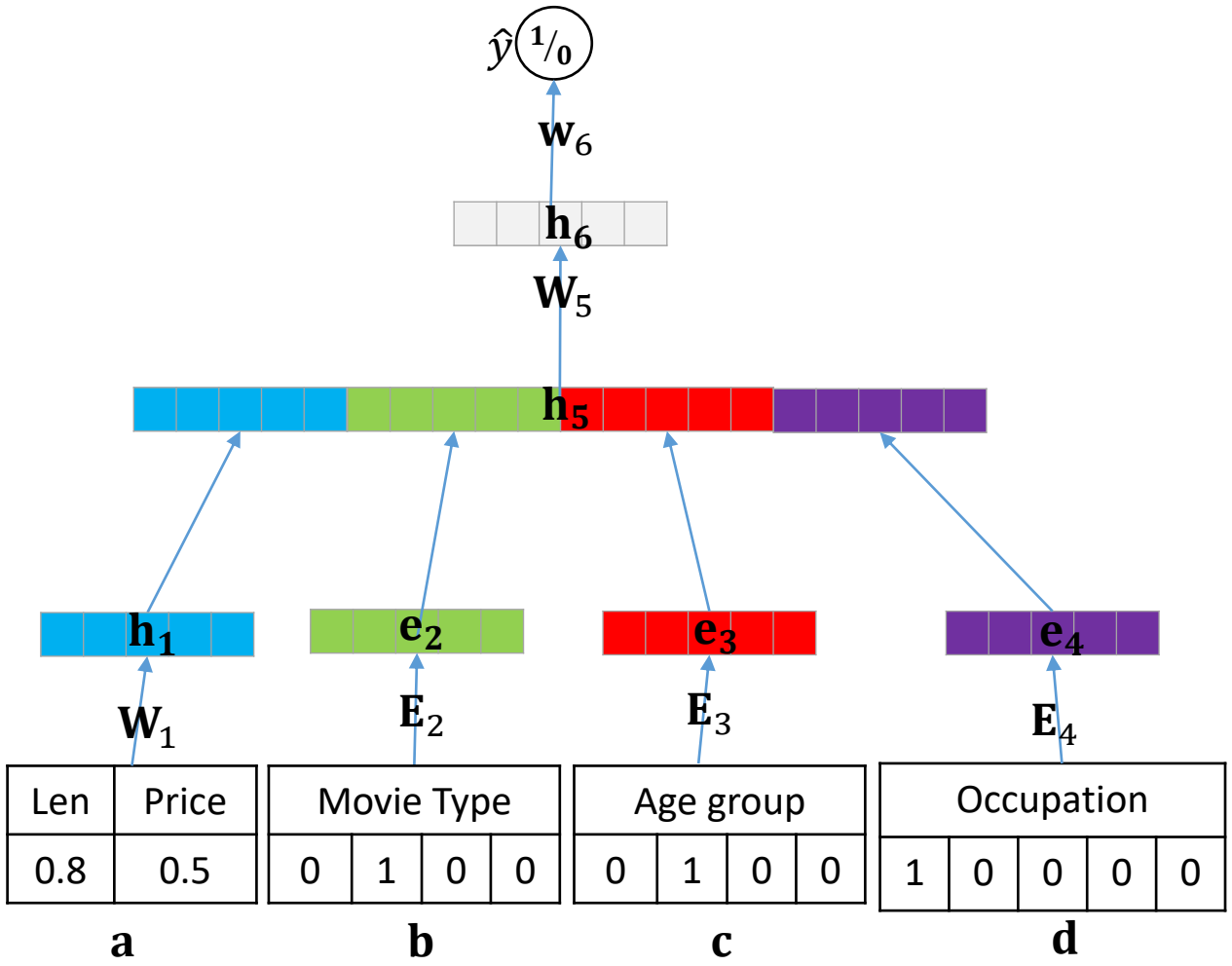
- Movie features:
  - Length: 120 Min, Ticket price: $20, Type: Action

| | Len | Price | Type | | | |
|---|---|---|---|---|---|---|
| ***a*** | 0.8 | 0.5 | 0 | 1 | 0 | 0 |

- User features:
  - Age group: 20-30, Occupation: IT Engineer

| | Age group | | | | Occupation | | | | |
|---|---|---|---|---|---|---|---|---|---|
| ***b*** | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |

# Deep regression models for recommendation




- Input layer:
  $\mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{d}$
- Layer 2:
  $\mathbf{h_1} = \mathrm{Relu}(\mathbf{W_1 a} + \mathbf{b_1})$
  $\mathbf{e_2} = \mathbf{E_2 b}$
  $\mathbf{e_3} = \mathbf{E_3 c}$
  $\mathbf{e_4} = \mathbf{E_4 d}$
- Layer 3:
  $\mathbf{h_5} = \mathrm{concat}([\mathbf{h_1}, \mathbf{e_2}, \mathbf{e_3}, \mathbf{e_4}])$
- Layer 4:
  $\mathbf{h_6} = \tanh(\mathbf{W_5 h_5} + \mathbf{b_5})$
- Output layer:
  $\hat{y} = \mathrm{sigmoid}(\mathbf{w}_6^{\mathrm{T}} \mathbf{h_6} + \mathrm{b}_6)$
- Objective:
  $L = \mathrm{binary\_crossentropy}(y, \hat{y})$

File Edit View Navigate Code Refactor Run Tools VCS Window Help DeepBlueTutorial - eval_neural_regression.py

DeepBlueTutorial > L5 > eval_neural_regression.py

eval_neural_regression Git:

Project

DeepBlueTutorial E:\AcadmicCode\DeepBlueTutorial
- L1
  - main.py
  - ua.base
  - ua.test
- L2
  - rankingmetric.py
- L3
  - preprocessMovieLens.py
- L4
  - mfdemo.py
  - ua.base
  - ua.test
- L5
  - eval_neural_mf.py
  - eval_neural_regression.py
  - neural_mf.py
  - neural_regression.py
- .gitignore
- README.md
- External Libraries
- Scratches and Consoles

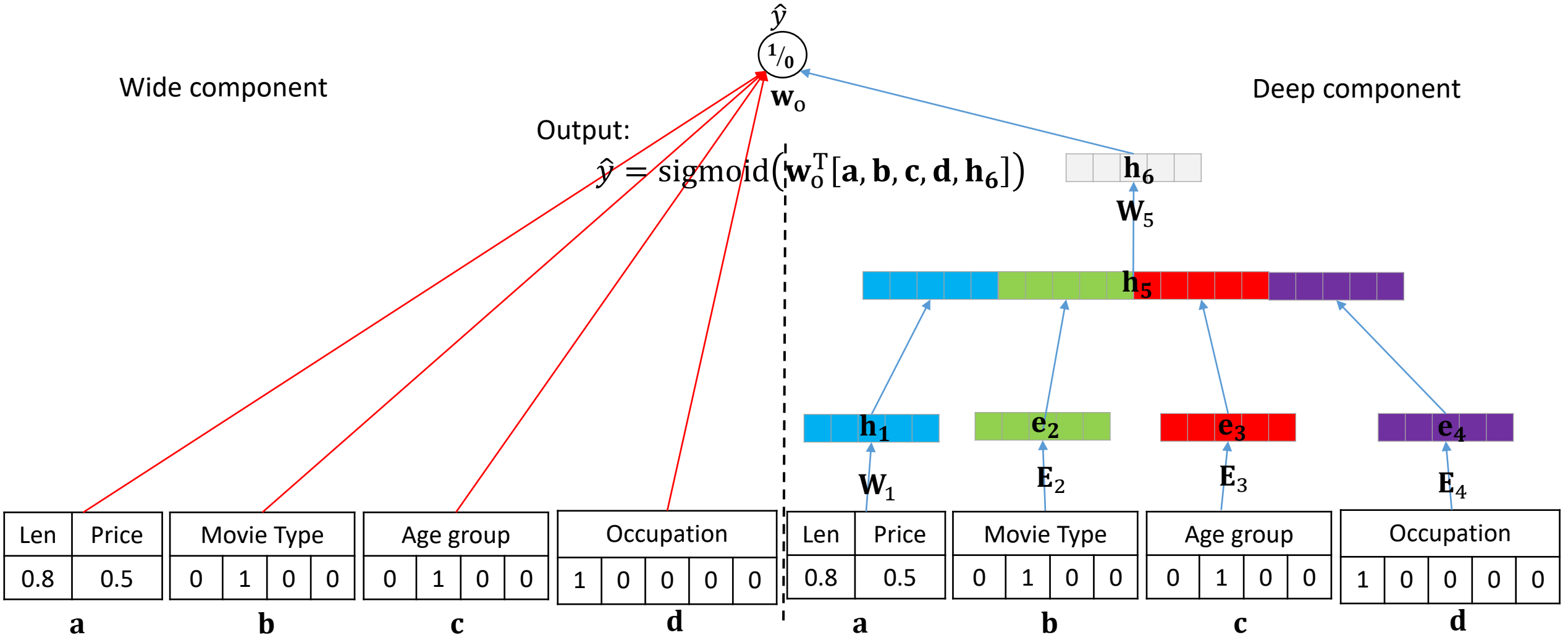
mfdemo.py | eval_neural_mf.py | neural_regression.py | eval_neural_regression.py | neural_mf.py | preprocessMovieLens.py | rankingmetric.py | main.py

```
1  from L5.neural_regression import NeuRegression
2
3
4  neu_reg = NeuRegression(nb_feature=2, nb_mv_type=5, nb_age_grp=5, nb_occupation=10)
5  neu_reg.model.compile(optimizer='adam', loss='binary_crossentropy')
6  neu_reg.model.summary()
7
```

6: Problems 9: Git Terminal Python Console TODO Event Log

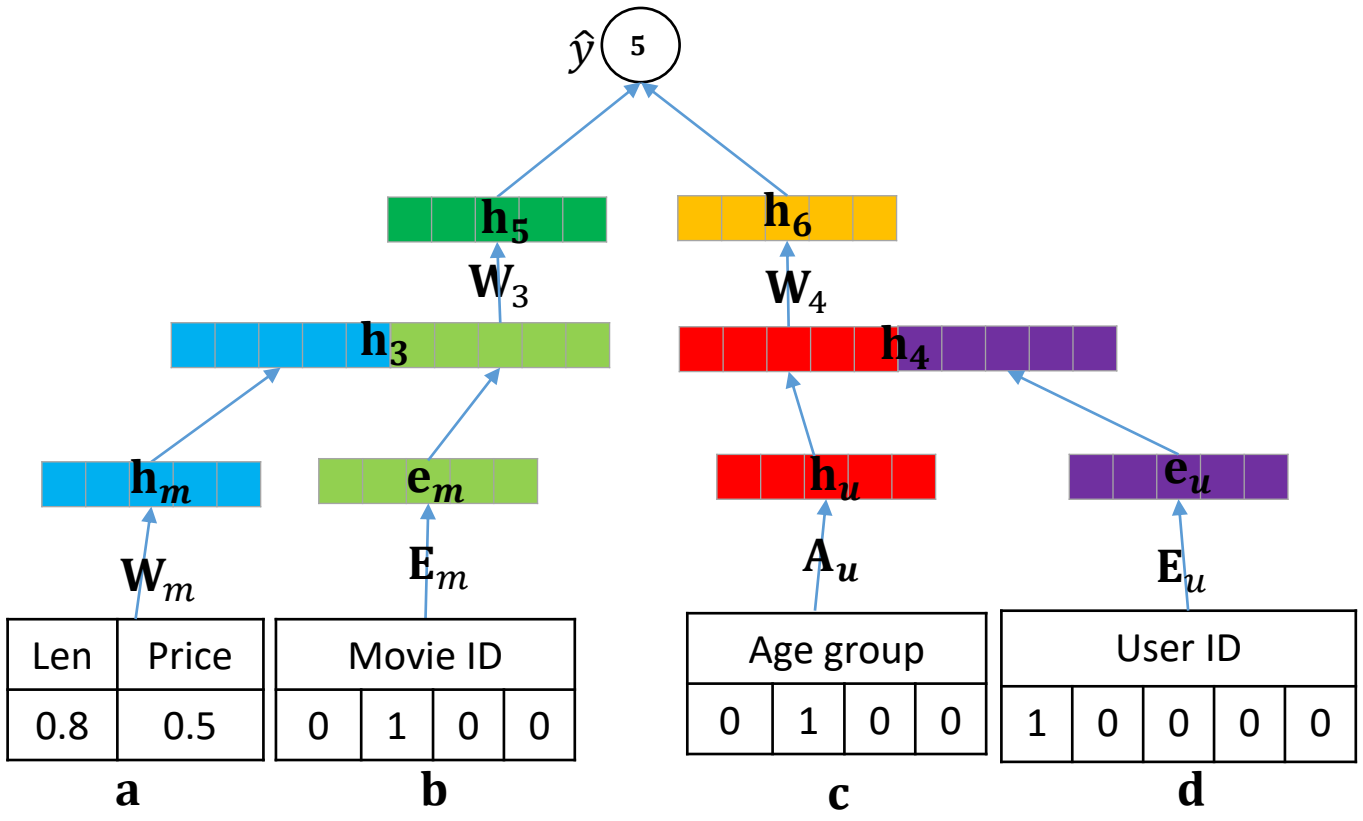
6:1 CRLF UTF-8 4 spaces Python 3.7 (tfgpu) master

# Wide and Deep Learning RS



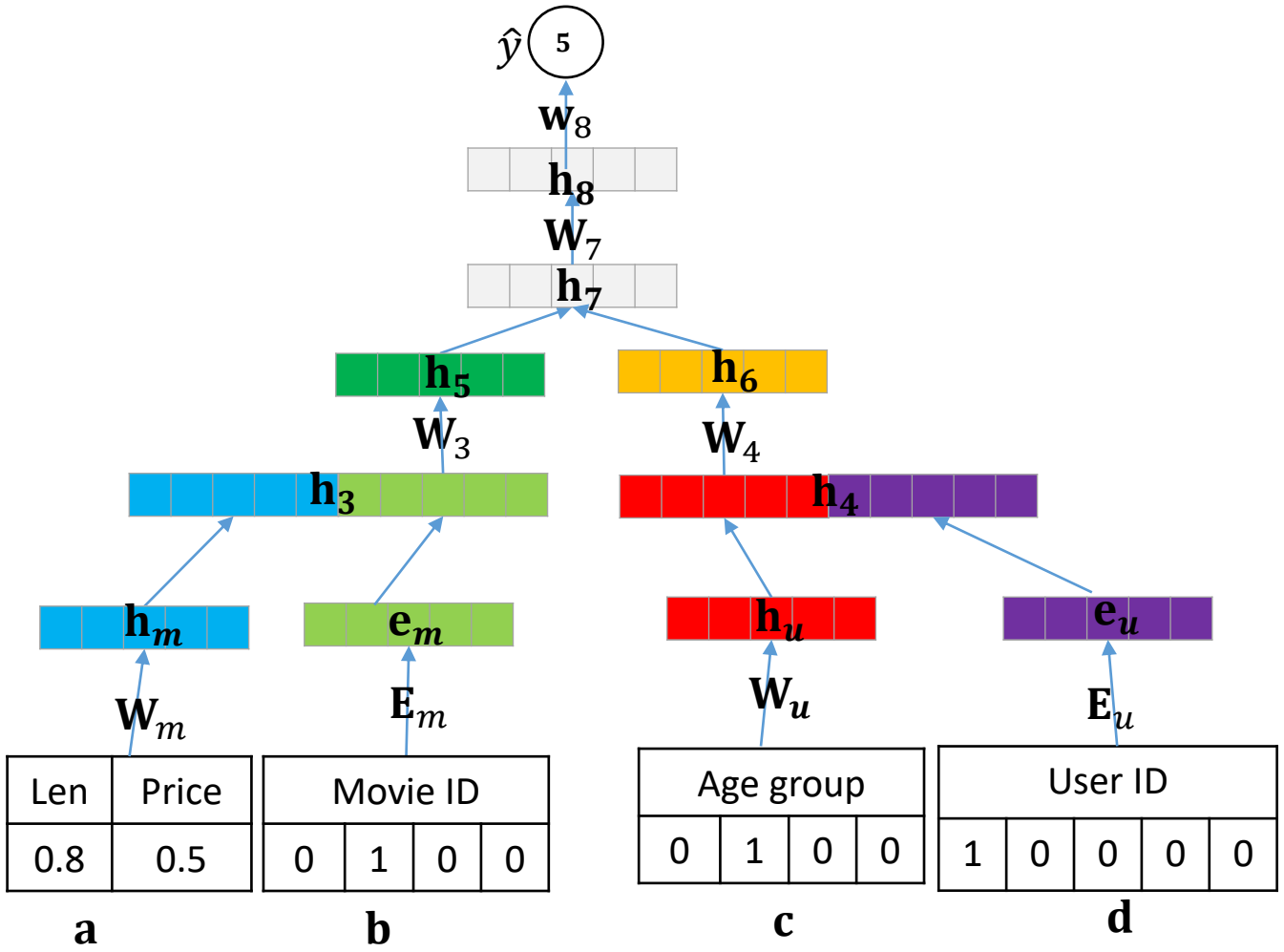

https://ai.googleblog.com/2016/06/wide-deep-learning-better-together-with.html

# Neural MF for rating prediction (V1)




- Input layer:
  $\mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{d}$
- Layer 2:
  $\mathbf{h}_m = \mathrm{Relu}(\mathbf{W}_\mathrm{m}\mathbf{a} + \mathbf{b}_m)$, $\mathbf{e}_m = \mathbf{E}_\mathrm{m}\mathbf{b}$
  $\mathbf{h}_u = \mathbf{A}_\mathrm{u}\mathbf{c}$, $\mathbf{e}_u = \mathbf{E}_\mathrm{u}\mathbf{d}$
- Layer 3:
  $\mathbf{h_3} = \mathrm{concat}([\mathbf{h}_m, \mathbf{e}_m]), \mathbf{h_4} = \mathrm{concat}([\mathbf{h}_u, \mathbf{e}_u])$
- Layer 4:
  $\mathbf{h_5} = \mathrm{ReLU}(\mathbf{W_3}\mathbf{h_3} + \mathbf{b}_3), \mathbf{h_6} = \mathrm{ReLU}(\mathbf{W_4}\mathbf{h_4} + \mathbf{b}_4)$
- Output layer:
  $\hat{y} = \langle \mathbf{h_5}, \mathbf{h_6} \rangle$
- Objective:
  $L = \mathrm{MSE}(y, \hat{y})$

# Neural MF for rating prediction (V2)




- Input layer:

  $\mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{d}$

- Layer 2:

  $\mathbf{h}_m = \mathrm{Relu}(\mathbf{W}_\mathrm{m}\mathbf{a} + \mathbf{b}_m)$, $\mathbf{e}_m = \mathbf{E}_\mathrm{m}\mathbf{b}$

  $\mathbf{h}_u = \mathbf{A}_\mathrm{u}\mathbf{c}$, $\mathbf{e}_u = \mathbf{E}_\mathrm{u}\mathbf{d}$

- Layer 3:

  $\mathbf{h_3} = \mathrm{concat}([\mathbf{h}_m, \mathbf{e}_m]), \mathbf{h_4} = \mathrm{concat}([\mathbf{h}_u, \mathbf{e}_u])$

- Layer 4:

  $\mathbf{h_5} = \mathrm{ReLU}(\mathbf{W}_3\mathbf{h_3} + b_3), \mathbf{h_6} = \mathrm{ReLU}(\mathbf{W}_4\mathbf{h_4} + \mathbf{b}_4)$

- Layer 5:

  $\mathbf{h_7} = \mathbf{h_5} * \mathbf{h_6}$

- Layer 6:

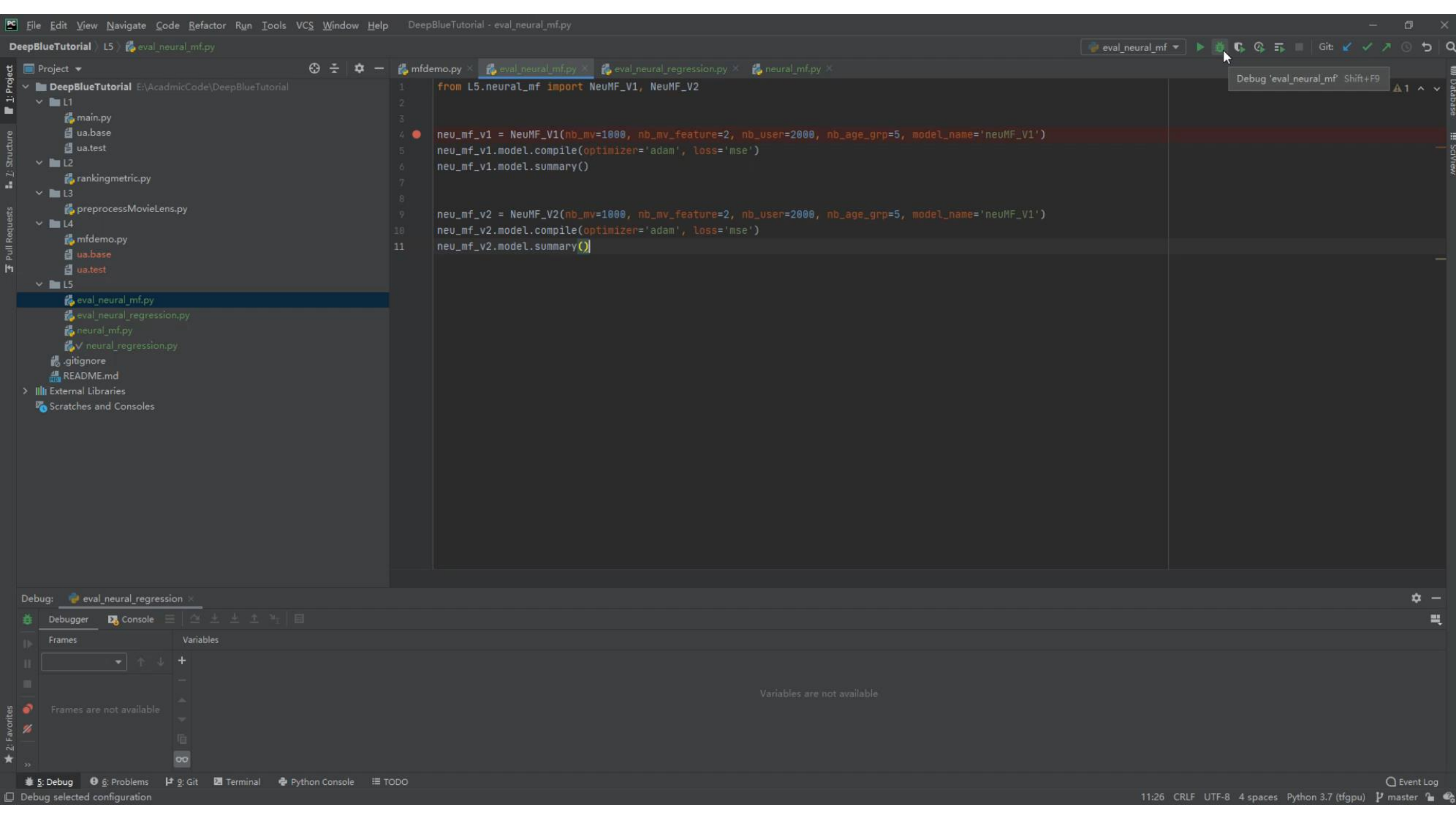
  $\mathbf{h_8} = \mathrm{ReLU}(\mathbf{W}_7\mathbf{h_7} + \mathbf{b}_7)$

- Output layer:

  $\hat{y} = \mathbf{w}_8^\mathrm{T}\mathbf{h_8} + b_8$

- Objective:

  $L = \mathrm{MSE}(y, \hat{y})$


File Edit View Navigate Code Refactor Run Tools VCS Window Help
DeepBlueTutorial - eval_neural_mf.py
DeepBlueTutorial L5 eval_neural_mf.py
eval_neural_mf
Git:
Debug 'eval_neural_mf' Shift+F9
Project
DeepBlueTutorial E:\AcadmicCode\DeepBlueTutorial
L1
main.py
ua.base
ua.test
L2
rankingmetric.py
L3
preprocessMovieLens.py
L4
mfdemo.py
ua.base
ua.test
L5
eval_neural_mf.py
eval_neural_regression.py
neural_mf.py
neural_regression.py
.gitignore
README.md
External Libraries
Scratches and Consoles
mfdemo.py
eval_neural_mf.py
eval_neural_regression.py
neural_mf.py
from L5.neural_mf import NeuMF_V1, NeuMF_V2
neu_mf_v1 = NeuMF_V1(nb_mv=1000, nb_mv_feature=2, nb_user=2000, nb_age_grp=5, model_name='neuMF_V1')
neu_mf_v1.model.compile(optimizer='adam', loss='mse')
neu_mf_v1.model.summary()
neu_mf_v2 = NeuMF_V2(nb_mv=1000, nb_mv_feature=2, nb_user=2000, nb_age_grp=5, model_name='neuMF_V1')
neu_mf_v2.model.compile(optimizer='adam', loss='mse')
neu_mf_v2.model.summary()
Debug: eval_neural_regression
Debugger
Console
Frames
Variables
Frames are not available
Variables are not available
5: Debug
6: Problems
9: Git
Terminal
Python Console
TODO
Event Log
Debug selected configuration
11:26 CRLF UTF-8 4 spaces Python 3.7 (tfgpu) master

# Review: factorization machines

$$\hat{y} = \boldsymbol{\mathcal{W}} \times \boldsymbol{x}^{\mathrm{T}} \times \boldsymbol{x}^{\mathrm{T}} \times \boldsymbol{x}^{\mathrm{T}} + \boldsymbol{x}^{\mathrm{T}}\mathbf{W}\boldsymbol{x} + \boldsymbol{x}^{\mathrm{T}}\mathbf{w}$$

- Factorize parameter tensors

$$\boldsymbol{\mathcal{W}} = [\![\mathbf{U}, \mathbf{U}, \mathbf{U}]\!], \qquad \mathbf{W} = \mathbf{V}^{\mathrm{T}}\mathbf{V}$$

- Replace parameter tensors $\boldsymbol{\mathcal{W}}$ and $\mathbf{W}$ with the factorization form

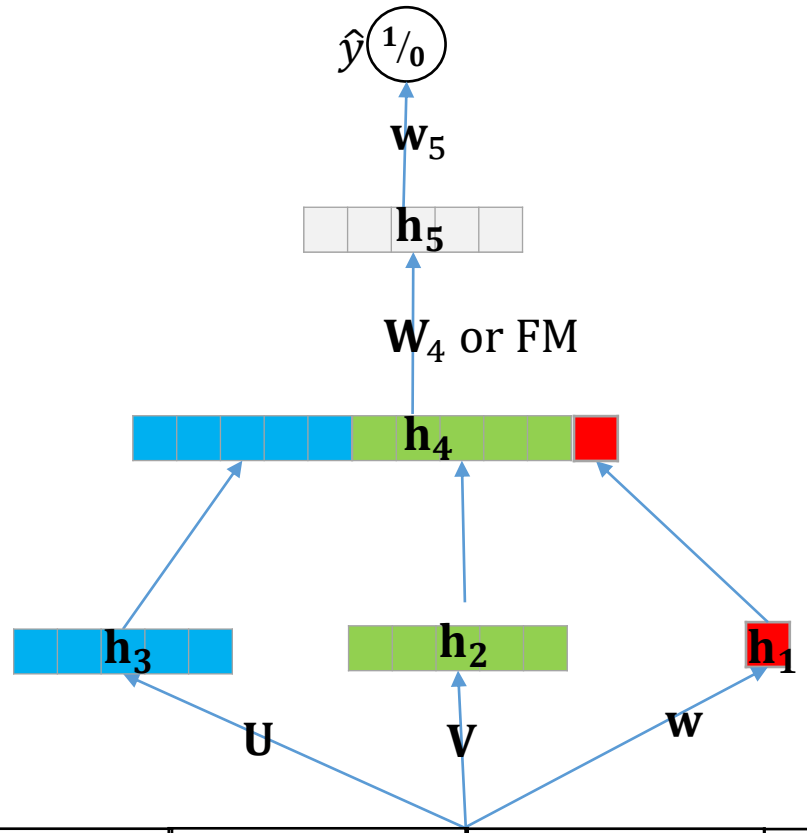
$$\hat{y} = \sum_{x_i, x_j, x_k \in \boldsymbol{x}} \langle \mathbf{u}_i, \mathbf{u}_j, \mathbf{u}_k \rangle x_i x_j x_k + \sum_{x_i, x_j \in \boldsymbol{x}} \langle \mathbf{v}_i, \mathbf{v}_j \rangle x_i x_j + \sum_{x_i \in \boldsymbol{x}} w_i x_i$$

$$\Rightarrow \quad \hat{y} = \langle \boldsymbol{x}^{\mathrm{T}}\mathbf{U}, \boldsymbol{x}^{\mathrm{T}}\mathbf{U}, \boldsymbol{x}^{\mathrm{T}}\mathbf{U} \rangle + \langle \boldsymbol{x}^{\mathrm{T}}\mathbf{V}, \boldsymbol{x}^{\mathrm{T}}\mathbf{V} \rangle + \boldsymbol{x}^{\mathrm{T}}\mathbf{w}$$

$$= \sum \mathbf{h}_3 + \sum \mathbf{h}_2 + \mathbf{h}_1$$

Where $\mathbf{h}_3 = \boldsymbol{x}^{\mathrm{T}}\mathbf{U} * \boldsymbol{x}^{\mathrm{T}}\mathbf{U} * \boldsymbol{x}^{\mathrm{T}}\mathbf{U}, \mathbf{h}_2 = \boldsymbol{x}^{\mathrm{T}}\mathbf{V} * \boldsymbol{x}^{\mathrm{T}}\mathbf{V}, \mathbf{h}_1 = \boldsymbol{x}^{\mathrm{T}}\mathbf{w}$

# Neural factorization machines




| User ID | | | | | Movie ID | | | | Age group | | | | Len | Price |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0.8 | 0.5 |

$\mathbf{x}$

- Input layer:

  $\mathbf{x}$

- Layer 2 (FM):

  $\mathbf{h_3} = \boldsymbol{x}^{\mathrm{T}}\mathbf{U} * \boldsymbol{x}^{\mathrm{T}}\mathbf{U} * \boldsymbol{x}^{\mathrm{T}}\mathbf{U}$

  $\mathbf{h_2} = \boldsymbol{x}^{\mathrm{T}}\mathbf{V} * \boldsymbol{x}^{\mathrm{T}}\mathbf{V}$

  $\mathbf{h_1} = \boldsymbol{x}^{\mathrm{T}}\mathbf{w}$

- Layer 3:

  $\mathbf{h_4} = \mathrm{relu}\big(\mathrm{concat}([\mathbf{h_1}, \mathbf{h_2}, \mathbf{h_3}])\big)$

- Layer 4 (you can also use FM again):

  $\mathbf{h_5} = \tanh(\mathbf{W}_4\mathbf{h_4} + b_4)$

- Output layer:

  $\hat{y} = \mathrm{sigmoid}\big(\mathbf{w}_5^{\mathrm{T}}\mathbf{h_5} + b_5\big)$

- Objective:

  $L = \mathrm{binary\_crossentropy}(y, \hat{y}\,)$

## All in one

- All heterogeneous models can be easily converted to neural versions:
  - Regression→Neural Regression Model
  - MF→Neural MF Model
  - TF→Neural TF Model
  - FM→ Neural FM Model
  - …
- Different neural models can be easily integrated.

# Outline

- Preliminary of Artificial Neural Networks
- A Brief Review of Deep Learning
- Deep Neural Models Based Recommender Systems
- Attention Mechanism

## Inspiration from human attention

- Attention Mechanisms in Neural Networks are loosely based on the visual attention mechanism found in humans.

- Human visual attention is well-studied and while there exist different models, all of them essentially come down to being able to **focus on a certain region** of an image with “high resolution” while perceiving the surrounding image in “low resolution”, and then adjusting the focal point over time.

# Attention in machine world

- Attention in ANNs has a long history, particularly in image recognition.
  - Learning to combine foveal glimpses with a third-order Boltzmann machine (2010)
  - Learning where to Attend with Deep Architectures for Image Tracking (2012)

# Neural Image Caption Generation with Visual Attention

- What is the magic behind?

*Figure 2.* Attention over time. As the model generates each word, its attention changes to reflect the relevant parts of the image. "soft" (top row) vs "hard" (bottom row) attention. (Note that both models generated the same captions in this example.)

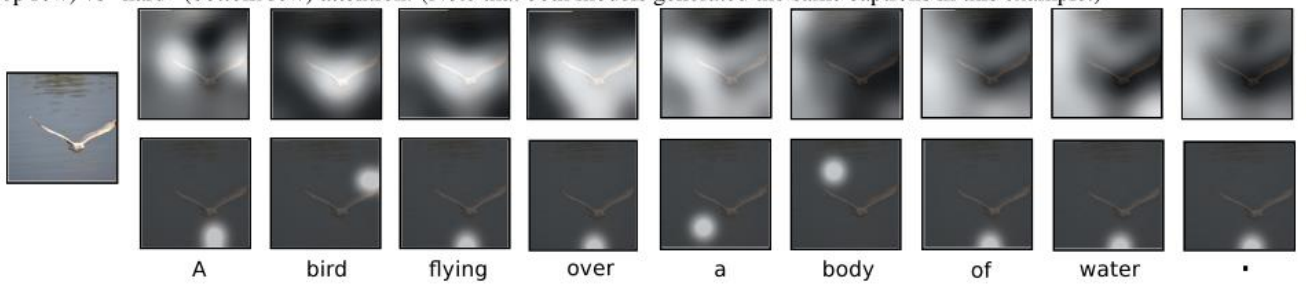
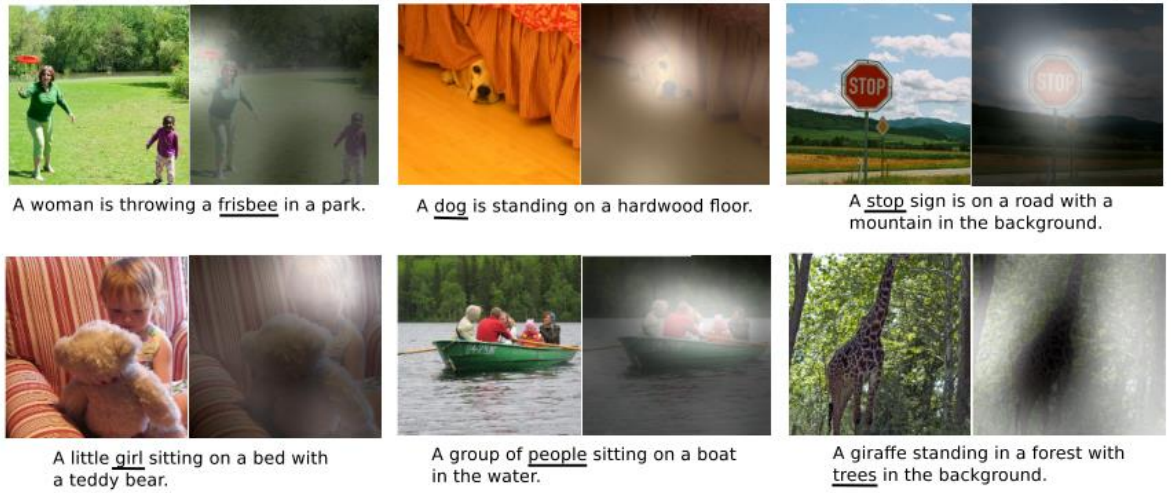
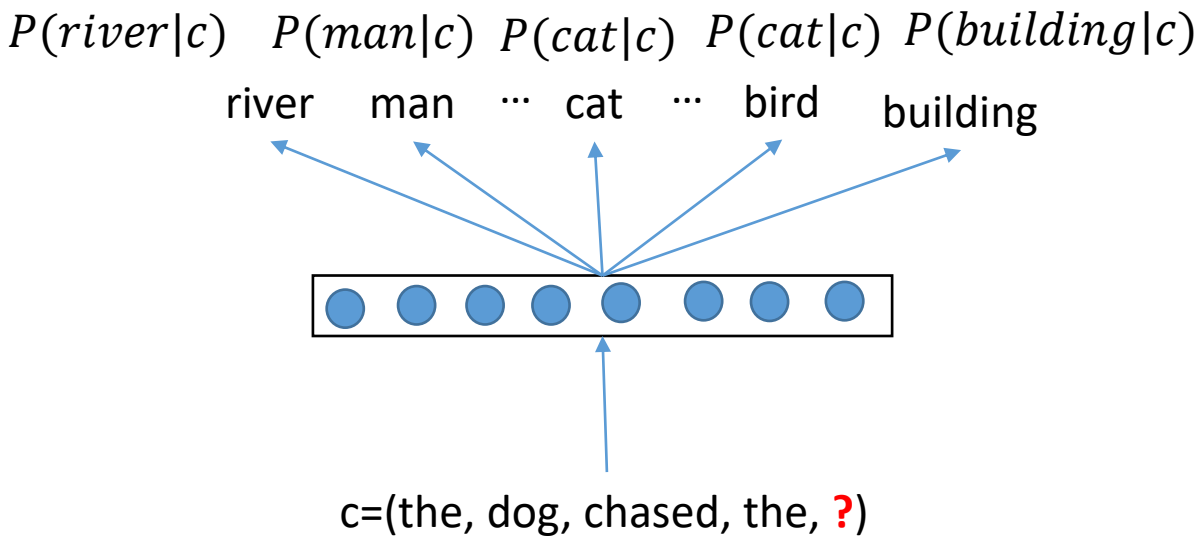



*Figure 3.* Examples of attending to the correct object (*white* indicates the attended regions, *underlines* indicated the corresponding word)



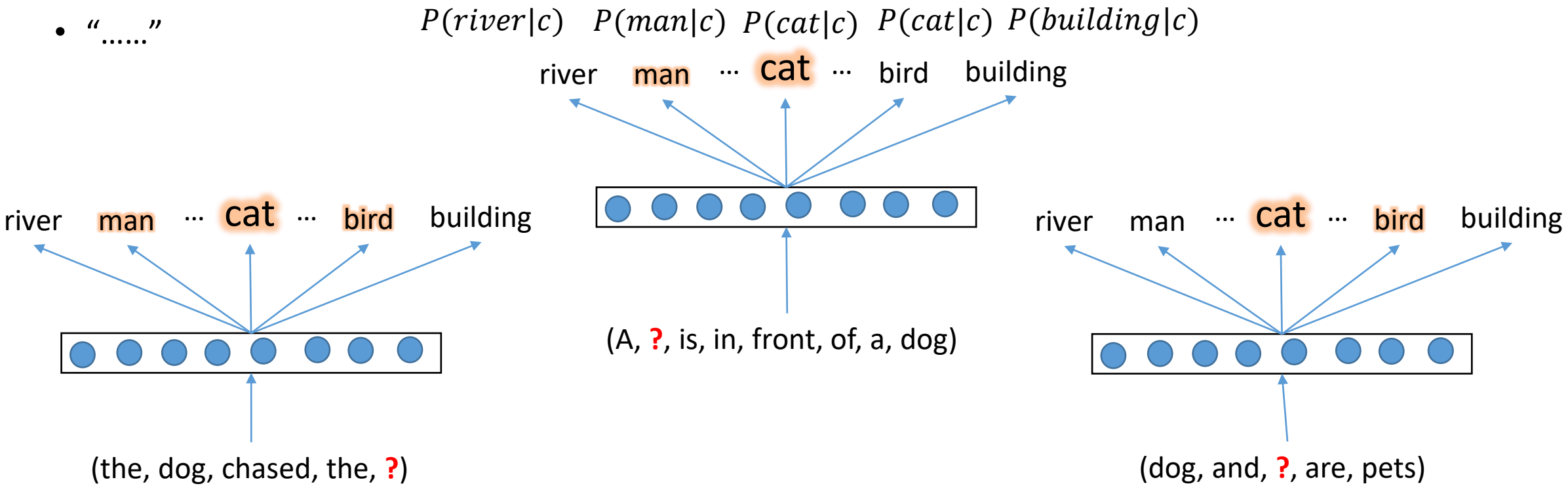

Xu, Kelvin, et al. "Show, attend and tell: Neural image caption generation with visual attention." *International conference on machine learning*. 2015.

# Machines have no bias without training data



# Statistical context biases

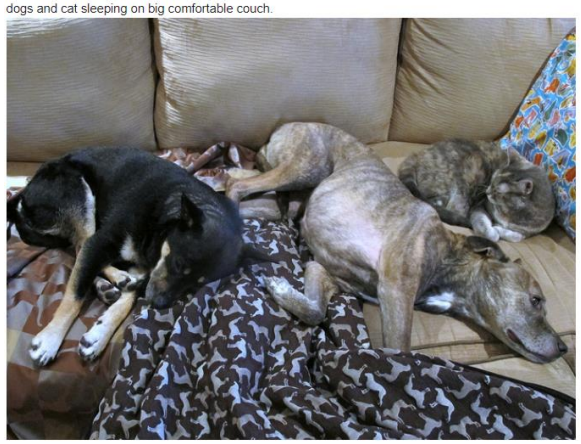
- “the dog chased the cat”
- “a cat is in front of a dog”
- “dog and cat are pets”
- “……”



# Machines do not have physical attention
# Only statistical scores

a photo of a woman sitting on the beach with two cows.
a woman on a beach with four dogs and two large animals.
person sitting on a beach with dogs and cattle.
a girl on beach with several farm animals
a person sitting on a beach with some animals.

two dogs and a cat take a nap on a couch.
two dogs sleep on the couch near the grey cat.
a couple of dogs are laying on a couch
two dogs and a cat laying down on a couch
dogs and cat sleeping on big comfortable couch.

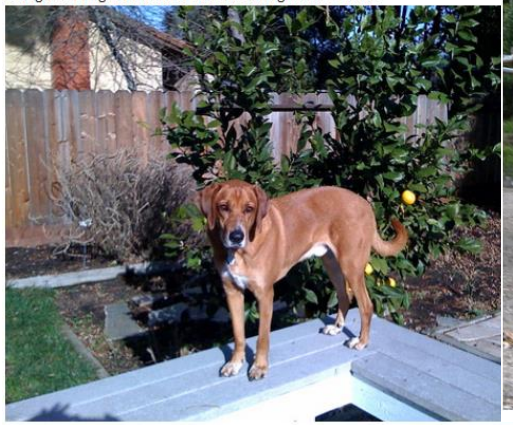


a dog that is jumping up to catch a frizbee
dog outside standing on his hind legs catching a frisbee.
a black dog standing on top of a grass covered forest.
a dog catches a toy in a wooded area.
a black and white border collie catching a red frisbee,

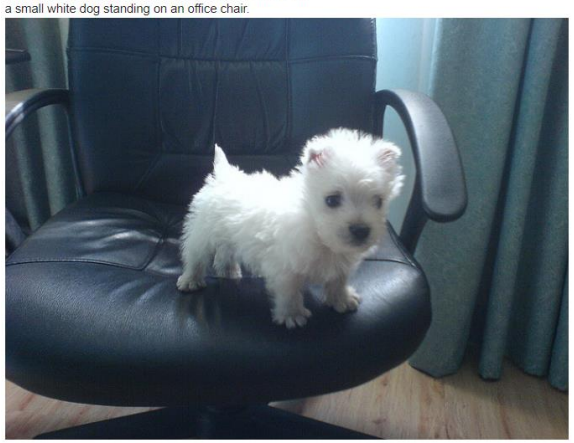
a dog in a rear view of a car mirror through a car window.
a dog that is in a car looking in a mirror
a dog being reflected on a side view mirror on a car.
the image of a dog can be seen in the rearview mirror.
dog as seen in side mirror of vehicle in black and white photo.

a view of a dump truck outside of a window
a dog looking out a tracker window watching rocks fall.
a dog watching a back hoe dumping rocks through a window.
a large window with a large brown dog looking out of it.
a truck is dumping objects in a trailer

a brown and white dog a fence and a tree
a dog standing on a ledge in front of a lemon tree.
a brown dog standing on a wooden bench near a lemon tree.
a large brown dog standing on the side of a white wood fence.
a dog standing on a bench with an orange tree in back.



a puppy sits on top of a computer chair
a small white dog is standing on a desk chair
the small dog is sitting on the office chair.
white puppy standing on the edge of a black office chair
a small white dog standing on an office chair.



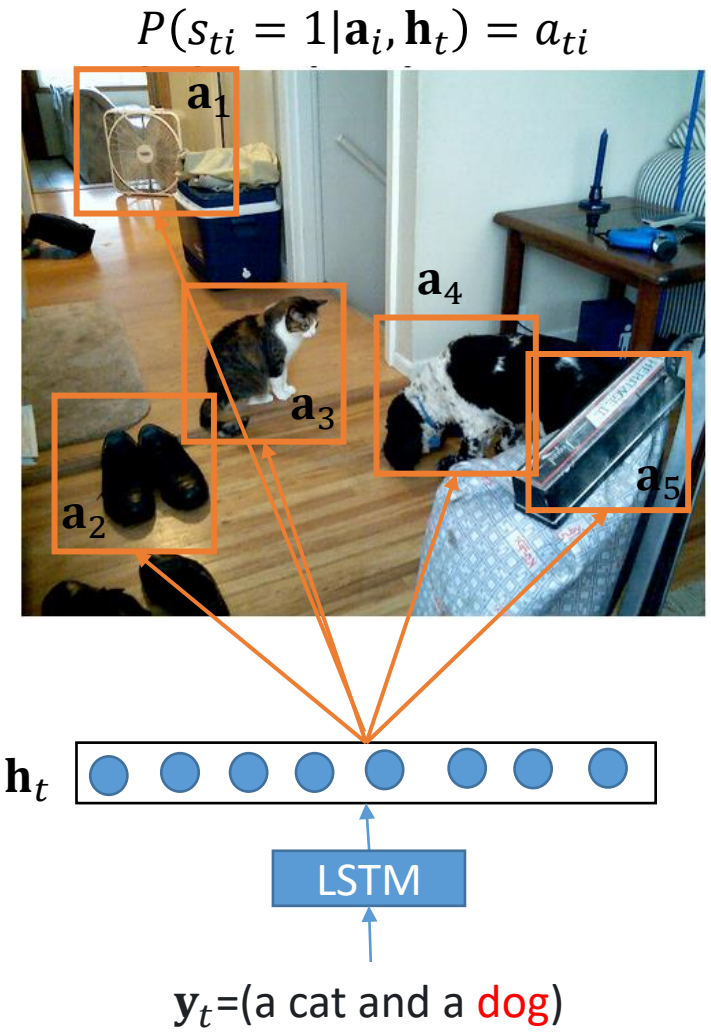
a group of horses are galloping near a dog.
a dog is running toward a group of running horses.
many horses running in a group on a path.
a herd of horses running down a dirt road.
a herd of horses running across a gravel road.

# Example: Statistical machine attention

"a cat and a dog on wooden floor next to a stairwell."

$$e_{ti} = f_{\text{att}}(\mathbf{a}_i, \mathbf{h}_t)$$
$$a_{ti} = \frac{\exp(e_{ti})}{\sum_k \exp(e_{tk})}$$

$\mathbf{y}_t$: the input word context
$\mathbf{a}_i$: the representation of a region of image
$\mathbf{h}_t$: context representation of a sequence of words
$e_{ti}$: the attention score
$a_{ti}$: the attention probability of an object
$f_{\text{att}}$: the attention model, e.g., multilayer neural network



# Summary of attention models

### Popular attention models and corresponding alignment score functions

| Name | Alignment score function | Citation |
|---|---|---|
| Content-base attention | $\text{score}(\boldsymbol{s}_t, \boldsymbol{h}_i) = \text{cosine}[\boldsymbol{s}_t, \boldsymbol{h}_i]$ | Graves2014 |
| Additive(*) | $\text{score}(\boldsymbol{s}_t, \boldsymbol{h}_i) = \mathbf{v}_a^\top \tanh(\mathbf{W}_a[\boldsymbol{s}_t; \boldsymbol{h}_i])$ | Bahdanau2015 |
| Location-Base | $\alpha_{t,i} = \text{softmax}(\mathbf{W}_a \boldsymbol{s}_t)$<br>Note: This simplifies the softmax alignment to only depend on the target position. | Luong2015 |
| General | $\text{score}(\boldsymbol{s}_t, \boldsymbol{h}_i) = \boldsymbol{s}_t^\top \mathbf{W}_a \boldsymbol{h}_i$<br>where $\mathbf{W}_a$ is a trainable weight matrix in the attention layer. | Luong2015 |
| Dot-Product | $\text{score}(\boldsymbol{s}_t, \boldsymbol{h}_i) = \boldsymbol{s}_t^\top \boldsymbol{h}_i$ | Luong2015 |
| Scaled Dot-Product(^) | $\text{score}(\boldsymbol{s}_t, \boldsymbol{h}_i) = \frac{\boldsymbol{s}_t^\top \boldsymbol{h}_i}{\sqrt{n}}$<br>Note: very similar to the dot-product attention except for a scaling factor; where n is the dimension of the source hidden state. | Vaswani2017 |

### Categories of attention models

| Name | Definition | Citation |
|---|---|---|
| Self-Attention(&) | Relating different positions of the same input sequence. Theoretically the self-attention can adopt any score functions above, but just replace the target sequence with the same input sequence. | Cheng2016 |
| Global/Soft | Attending to the entire input state space. | Xu2015 |
| Local/Hard | Attending to the part of input state space; i.e. a patch of the input image. | Xu2015; Luong2015 |

https://lilianweng.github.io/lil-log/2018/06/24/attention-attention.html

# Applying attention in recommender systems

- Session-based recommender systems (temporal attention)
- Graph-based recommender systems (topological attention)
- Explainable recommender systems (sentimental attention)

# Summary

- Artificial neural networks (ANN) are computing systems inspired by the biological neural networks; Backpropagation is the major learning algorithm for ANN.

- Deep learning is the iconic product which results in the renaissance of AI of this generation. However, the core components of deep learning have not been essentially changed since three decades ago.

- Deep learning is tailored for recommender systems. It is easy to convert traditional recommendation models into deep neural versions.

- Attention mechanisms in ANN are just statistical interaction scores.

# Assignment I

MovieLens 1M dataset contains 1,000,209 anonymous ratings of approximately 3,900 movies made by 6,040 users.

Experimental dataset: https://grouplens.org/datasets/movielens/1m/

**Task 1:** Using Wide and Deep model to predict movie ratings in terms of user and item attributes.

- Evaluation metrics: MAE, RMSE

**Task 2:** Using Neural MF to predict movie ratings.

- Evaluation metrics: MAE, RMSE

# Assignment II

**Restaurant & consumer data for context-aware recommendation.**

The tasks were to generate a top-n list of venue according to the consumer preferences at the given time.

Experimental dataset: https://www.kaggle.com/chetanism/foursquare-nyc-and-tokyo-checkin-dataset

**Task:** Using neural FM to model context-aware user-item interactions:

$$\langle User\ ID, Venue\ ID, Venue\ Category, Checkin\ Time \rangle$$

- Evaluation metrics: Recall, MAP@5, MAP@10, nDCG@5, nDCG@20