中国科学院大学

University of Chinese Academy of Sciences

Deep Learning

Representative Deep Learning Methods

Xinfeng Zhang（张新峰）

School of Computer Science and Technology

University of Chinese Academy of Sciences

Email: xfzhang@ucas.ac.cn

计算机科学与技术学院

SCHOOL OF COMPUTER SCIENCE AND TECHNOLOGY

提纲

- 生成对抗网络
- 胶囊网络
- 注意力机制
- 记忆网络
- 深度强化学习
- 深度森林
- 中英文术语对照

3 注意力机制

人类注意力机制

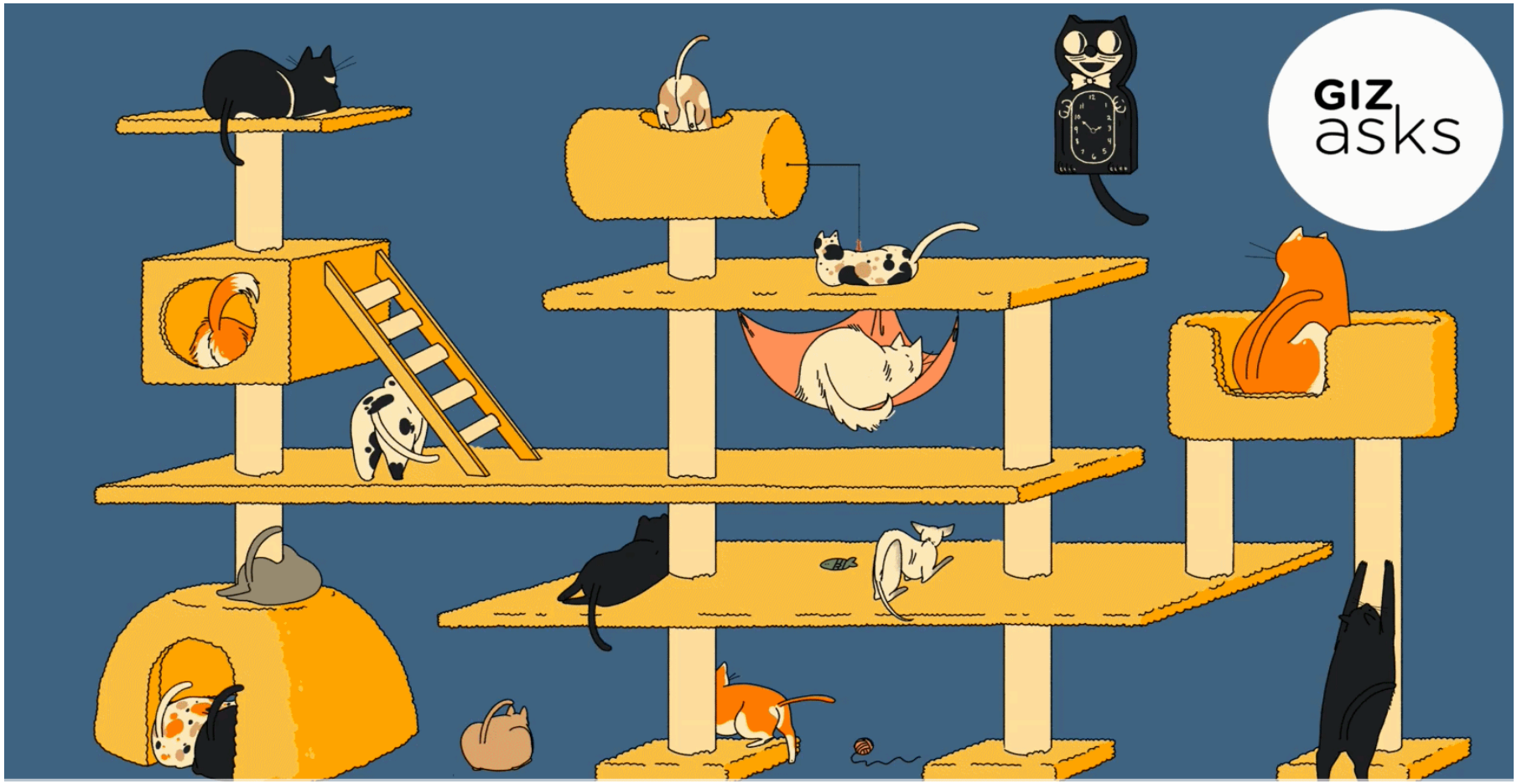
A woman is throwing a frisbee in a park.

A dog is standing on a hardwood floor.

A little girl sitting on a bed with a teddy bear.

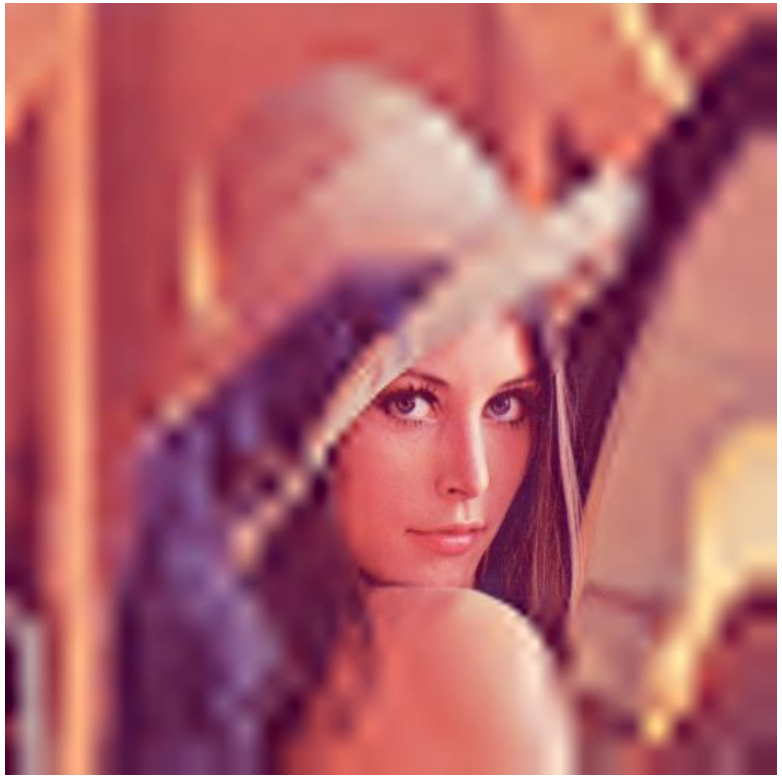
A group of people sitting on a boat in the water.

人类注意力机制



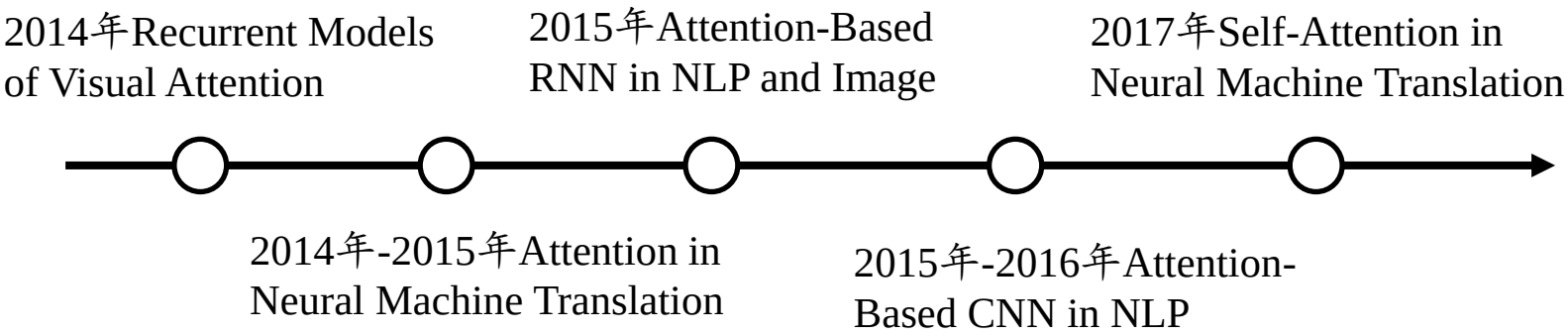
注意力机制的研究和应用

- http://saliency.mit.edu/datasets.html
- 数据库构建
  - 眼动仪
- 传统图像和视频处理中的应用
  - 视频/图像质量评价
  - 视频/图像的视觉编码优化
  - ….



在深度学习中的注意力机制

- **Recurrent Models of Visual Attention. NIPS 2014: 2204-2212**
- **Neural machine translation by jointly learning to align and translate, ICLR 2015**
- **Show, Attend and Tell: Neural Image Caption Generation with Visual Attention，ICML 2015**
- **Attention is all you need，NIPS 2017**

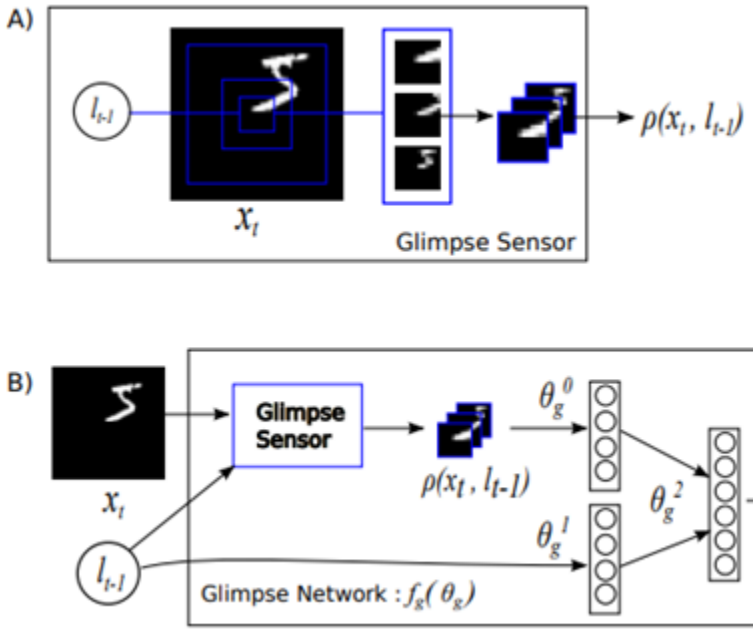


注意力机制

❑ Recurrent Models of Visual Attention

- RNN模型处理图像分类任务，采用强化学习的方法训练
- 按照时间顺序处理输入，一次在一张图像中处理不同的位置，逐渐的将这些部分的信息结合起来，来建立一个该场景或者环境的动态间隔表示
- Glimpse Sensor:
  - 提取图像输入

    通过位置信息l_{t-1}，以其为中心提取长宽为w的倍数提取图像区域，并经他们归一化为$w * w$, 拼接得到图像层次信息
- Glimpse Network
  - $f(\theta_g^0), f(\theta_g^1), f(\theta_g^2),$



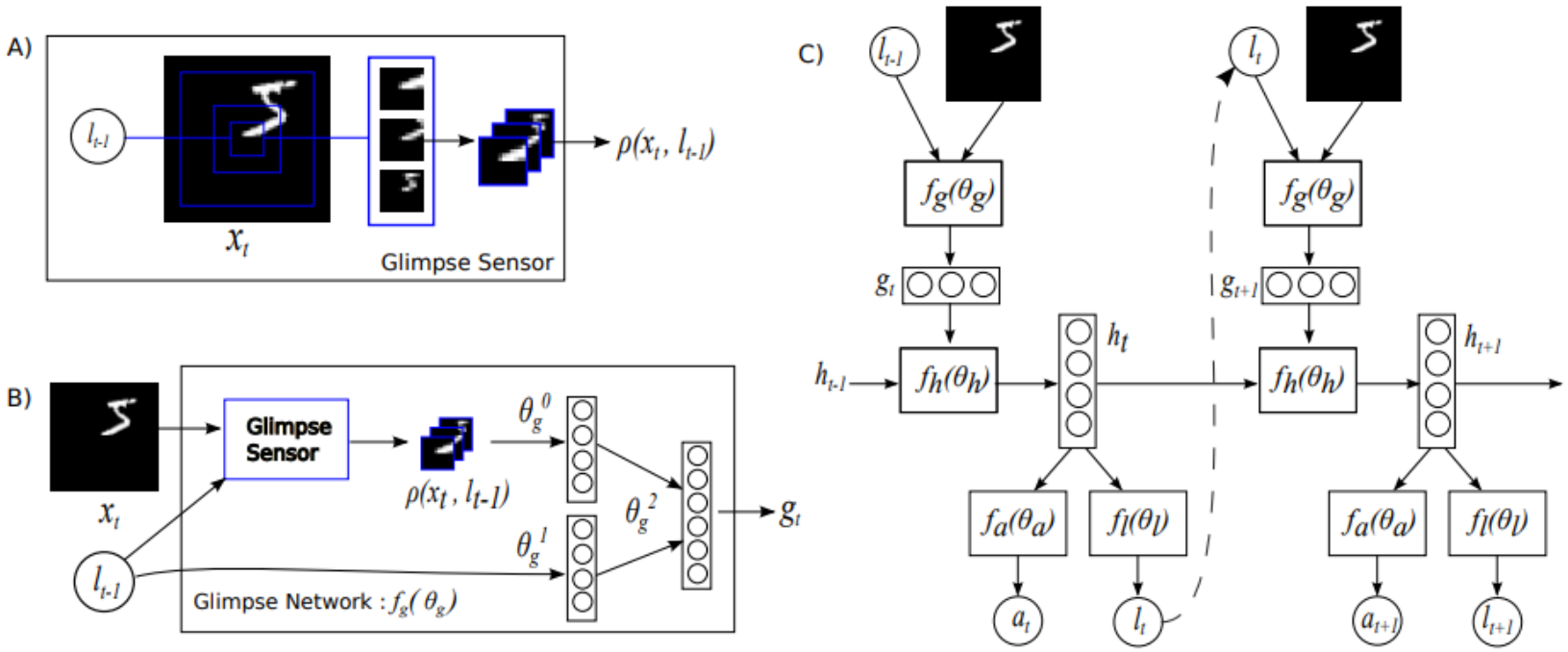
Volodymyr Mnih, Nicolas Heess, Alex Graves, Koray Kavukcuoglu. Recurrent Models of Visual Attention. NIPS 2014: 2204-2212

注意力机制

❑ Recurrent Models of Visual Attention

– 完整的结构

• Action和Location提取

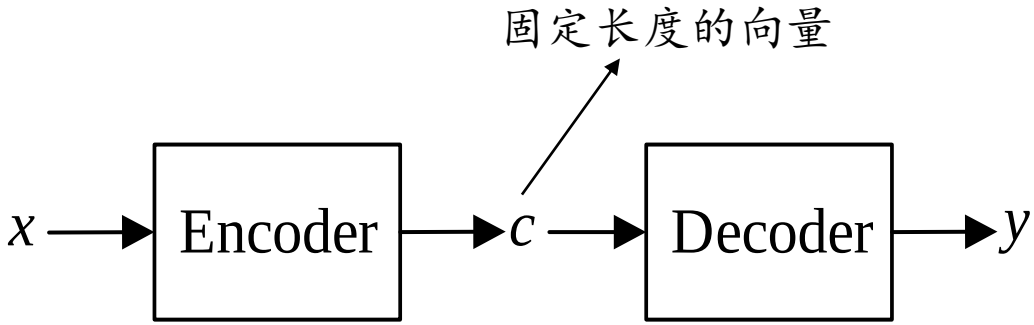


Volodymyr Mnih, Nicolas Heess, Alex Graves, Koray Kavukcuoglu. Recurrent Models of Visual Attention. NIPS 2014: 2204-2212

注意力机制

❑ 注意力机制在在神经机器翻译领域的应用

– 神经机器翻译主要以Encoder-Decoder模型为基础结构



源语言输入序列：$x=(x_1,x_2,...,x_n)$

目标语言输出序列：$y=(y_1,y_2,...,y_m)$

Dzmitry Bahdanau, et al., Neural machine translation by jointly learning to align and translate, 2015

注意力机制

❑ 注意力机制在神经机器翻译领域的应用

– 在神经机器翻译中，Encoder一般采用RNN或者LSTM实现

• 从统计角度，翻译相当于寻找译句**y**，使得给定原句$\boldsymbol{x}$时条件概率最大

$$arg\max_{\boldsymbol{y}} p(\boldsymbol{y}|\boldsymbol{x})$$

– 得到上下文向量c的方法有很多，可以直接将最后一个隐状态作为上下文变量，也可对最后的隐状态进行一个非线性变换σ(·)，或对所有的隐状态进行非线性变换σ(·)

$$h_t = f(x_t, h_{t-1})$$

$$c = h_T$$

$$c = \sigma(h_T)$$

$$c = \sigma(h_1, h_2, \cdots, h_T)$$

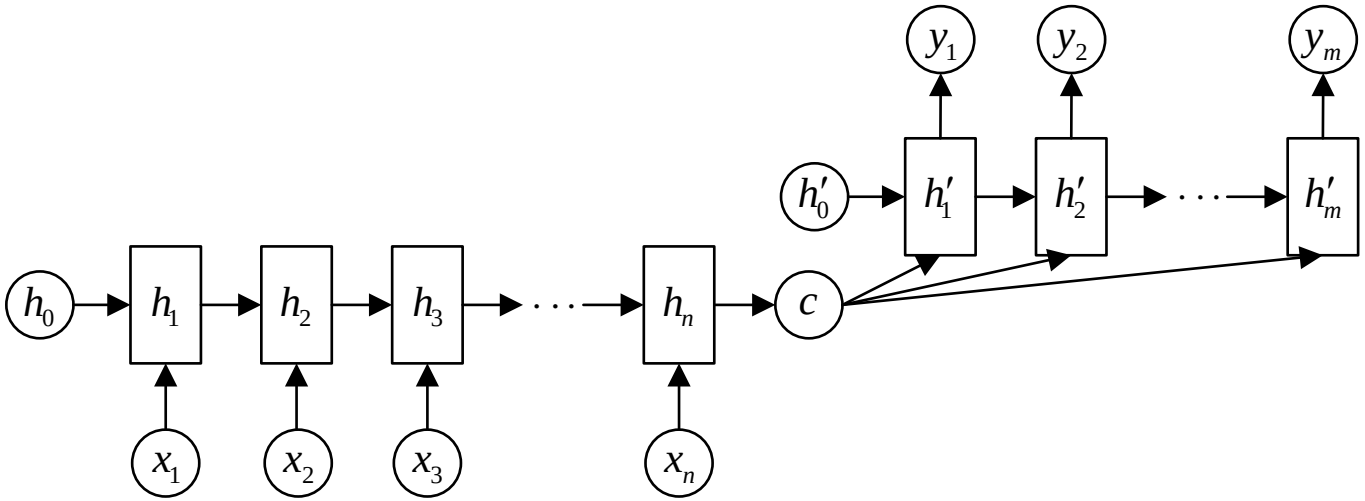
Dzmitry Bahdanau, et al., Neural machine translation by jointly learning to align and translate, 2015

注意力机制

❑ 解码器

– 用给定的上下文向量$\boldsymbol{c}$和之前已经预测的词$\{y_1,\ldots,y_{t-1}\}$

$$y_1 = G(\boldsymbol{c}) \qquad y_2 = G(\boldsymbol{c}, y_1) \qquad y_3 = G(\boldsymbol{c}, y_1, y_2)$$



注意力机制

❑ 现存问题

- 输入序列不论长短都会被编码成一个固定长度的向量表示，而解码则受限于该固定长度的向量表示
- 这个问题限制了模型的性能，尤其是当输入序列比较长时，模型的性能会变得很差

注意力机制

❑ 神经网络模型注意力机制

– 在这个新结构中，定义每个输出的条件概率为：

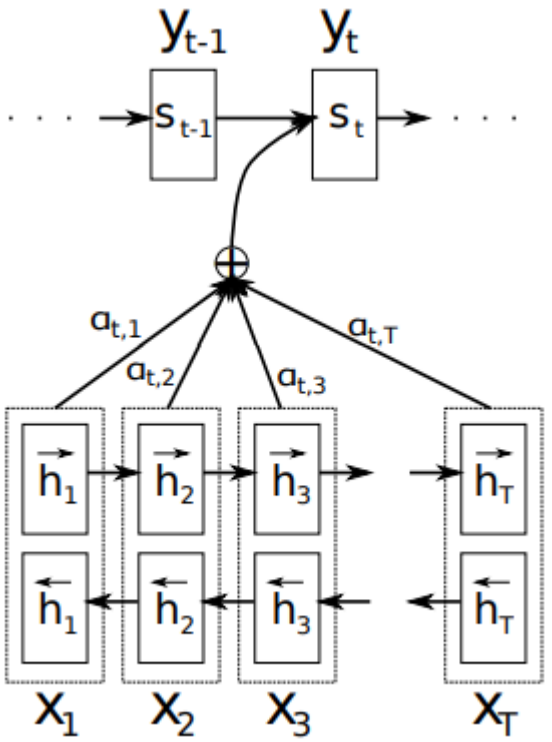
$$p(y_i \mid y_1,\cdots,y_{i-1}, \boldsymbol{x})=g(y_{i-1},x_i,c_i)$$

– 其中s_i为解码器RNN中的隐层状态：

$$s_i=f(s_{i-1},y_{i-1},c_i)$$

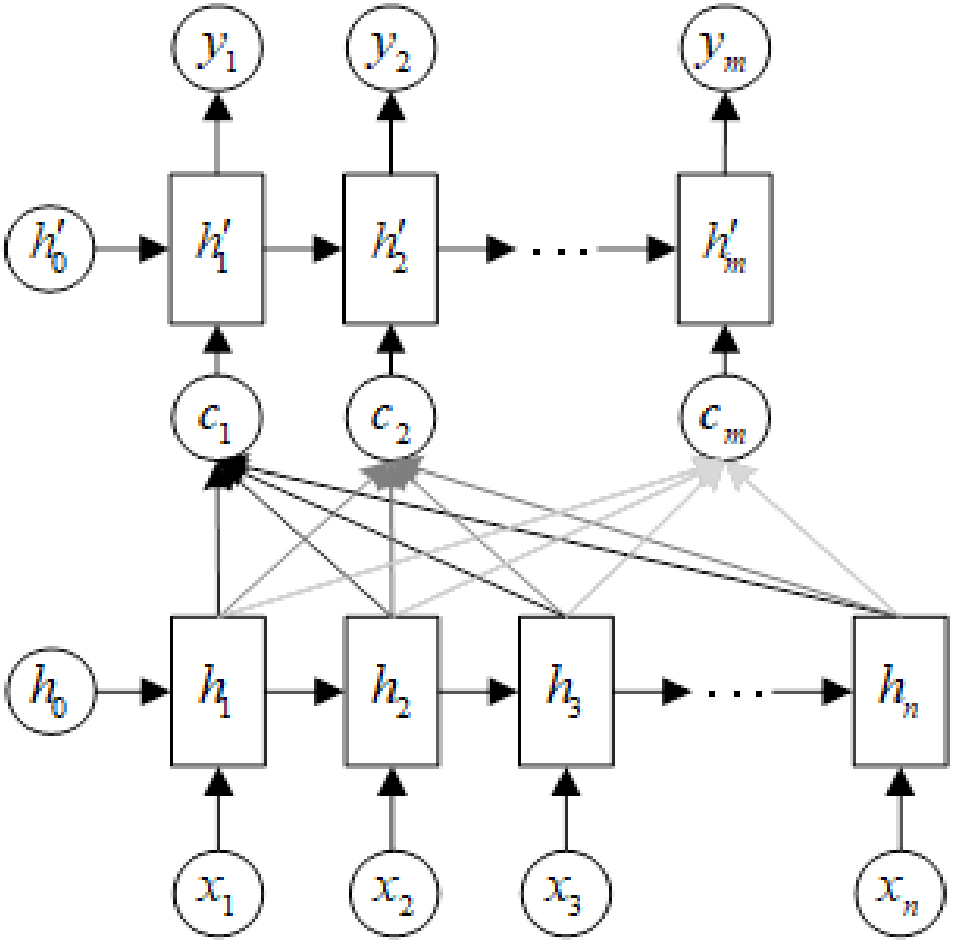
– 这里的上下文向量c_i取决于解码器状态序列，通过使用注意力系数α_{ij}对h_j加权求得

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



注意力机制

❑ 神经网络模型注意力机制



注意力机制

❑ 注意力系数计算

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{T_x} \exp(e_{ik})} \qquad e_{ij} = a(s_{i-1}, h_j)$$

↓

alignment mode

反映i位置的输入和j位置输出的匹配程度

❑ 计算注意力系数的相似函数(alignment model)有以下几种:

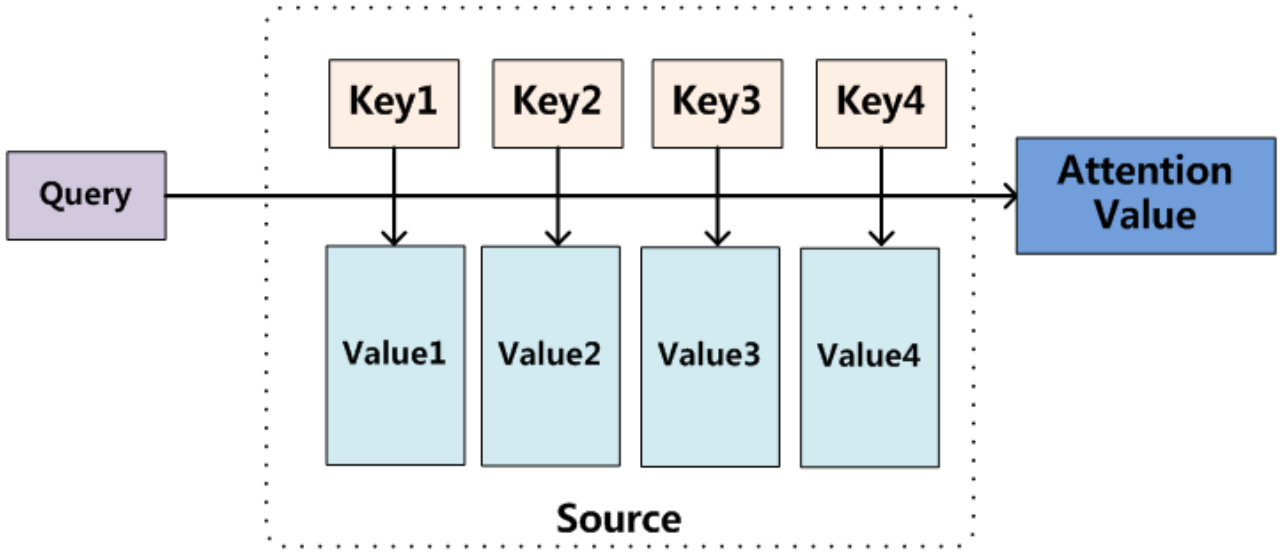
$$a(s_{i-1}, h_j) = \begin{cases} h_j^T \cdot s_{i-1} \\ \dfrac{h_j^T \cdot W_\alpha \cdot s_{i-1}}{W_\alpha \cdot \left[h_j^T, s_{i-1}^T\right]^T} \\ v_\alpha \tanh(U_\alpha h_j + W_\alpha s_{i-1}) \end{cases}$$

几种主流的注意力机制

Name	Alignment score function	Citation
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	$\alpha_{t,i} = \text{softmax}(\mathbf{W}_a \boldsymbol{s}_t)$ Note: This simplifies the softmax alignment max to only depend on the target position.	Luong2015
General	$\text{score}(\boldsymbol{s}_t, \boldsymbol{h}_i) = \boldsymbol{s}_t^\top \mathbf{W}_a \boldsymbol{h}_i$ 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(^)	$\text{score}(\boldsymbol{s}_t, \boldsymbol{h}_i) = \frac{\boldsymbol{s}_t^\top \boldsymbol{h}_i}{\sqrt{n}}$ Note: very similar to the dot-product attention except for a scaling factor; where n is the dimension of the source hidden state.	Vaswani2017
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

注意力机制的抽象理解

- **Attention函数的本质可以被描述为一个查询（query）到一系列（键key-值value）对的映射**



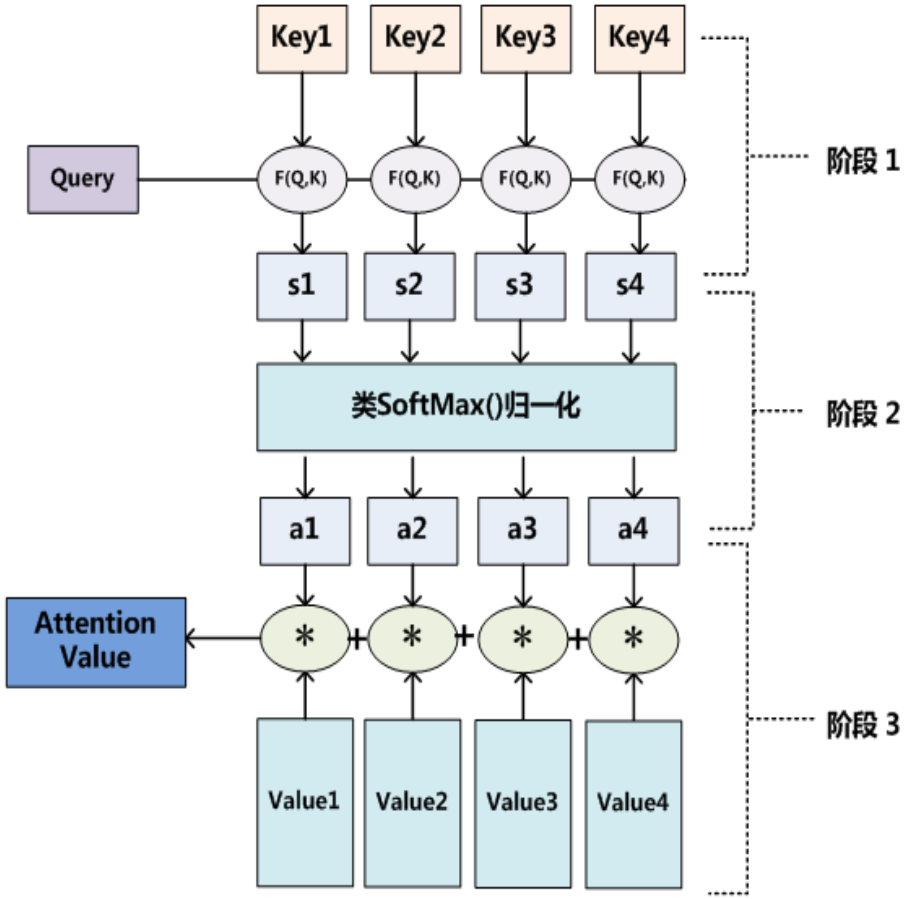
$$\text{Attention}(Query, Source) = \sum_{i=1}^{L_x} similarity(Query, Key_i) * Value_i$$

Vaswani A, Shazeer N, Parmar N, et al. Attention is all you need[C]. Advances in Neural Information Processing Systems. 2017: 6000-6010.

注意力机制的抽象理解

❑ 注意力系数计算

- 阶段1：根据Query和Key计算两者的相似性或者相关性
- 阶段2：对第一阶段的原始分值进行归一化处理
- 阶段3：根据权重系数对Value进行加权求和，得到Attention Value



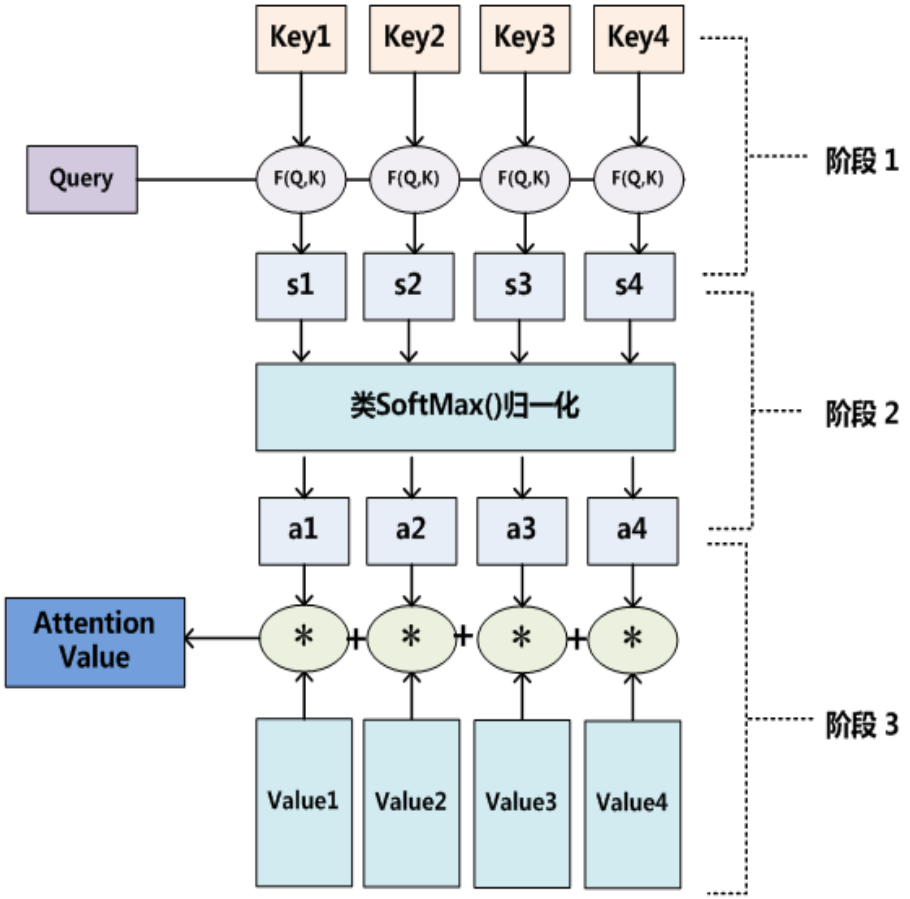
注意力机制的抽象理解

❑ 注意力系数计算

$$F(Q,K_i)=\begin{cases} Q^T\cdot K_i \\ Q^T\cdot W_\alpha\cdot K_i \\ W_\alpha\cdot[Q;K_i]^T \\ v_\alpha\tanh(U_\alpha K_i+W_\alpha Q) \end{cases}$$

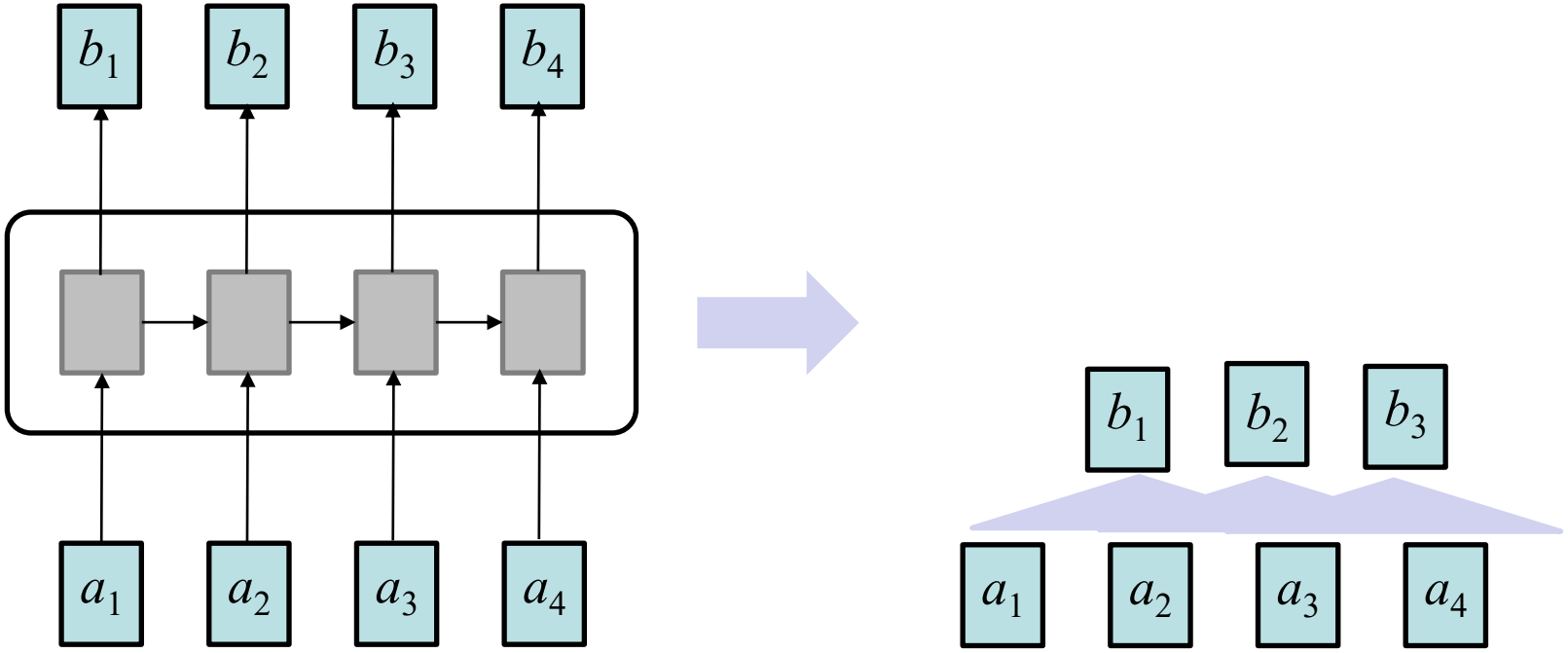
$$a_i=\frac{\exp(f(Q,K_i))}{\sum_j exp(f(Q,K_j))}$$

$$\text{Attention}(Q,K,V)=\sum_j a_j V_j=\text{c}$$



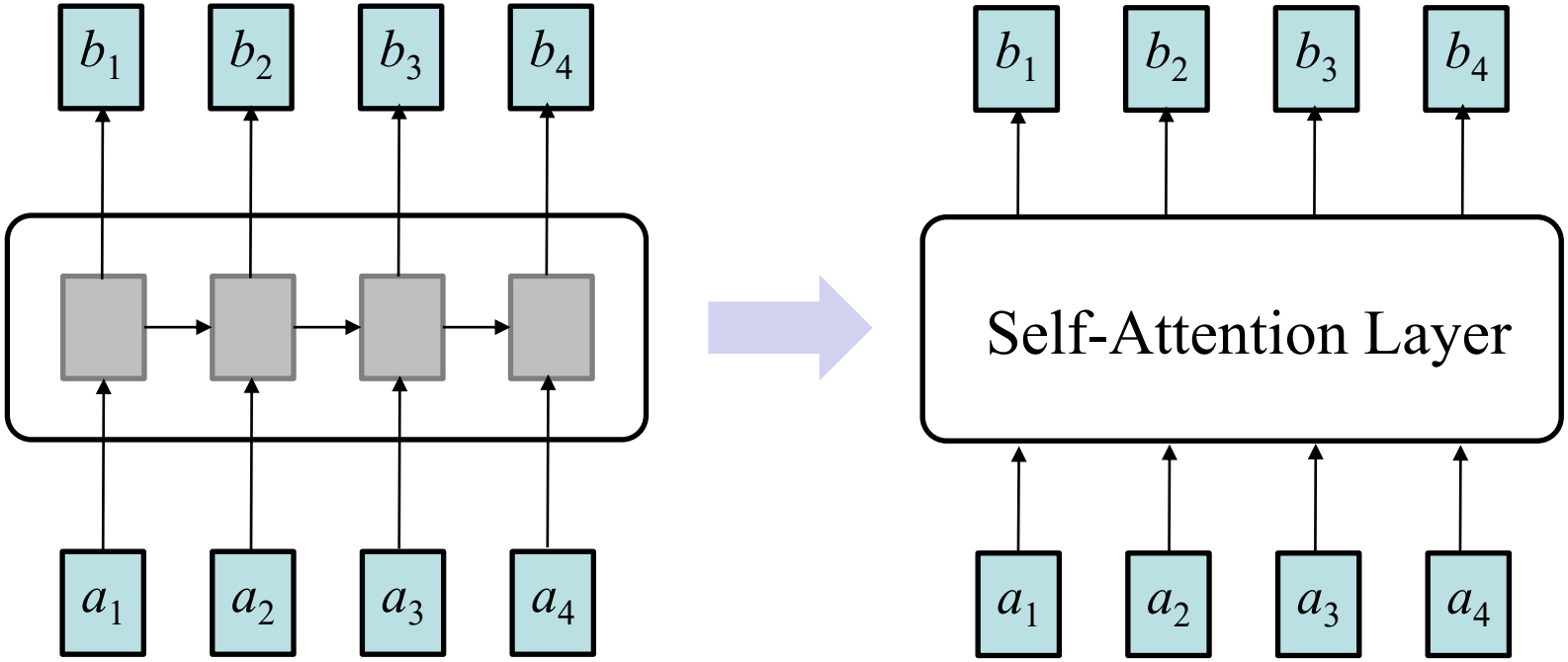
注意力机制的抽象理解

- **Self-attention layer in "Attention is all you need"**



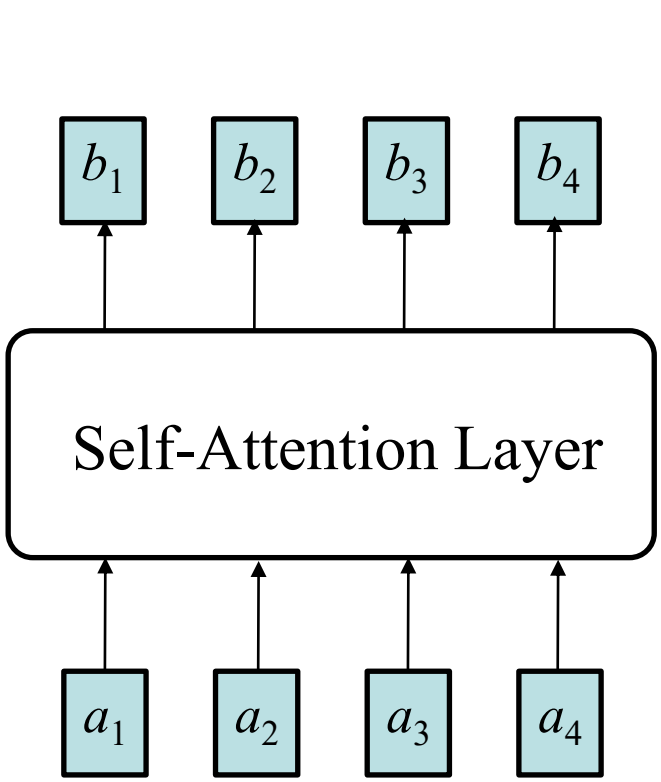
注意力机制的抽象理解

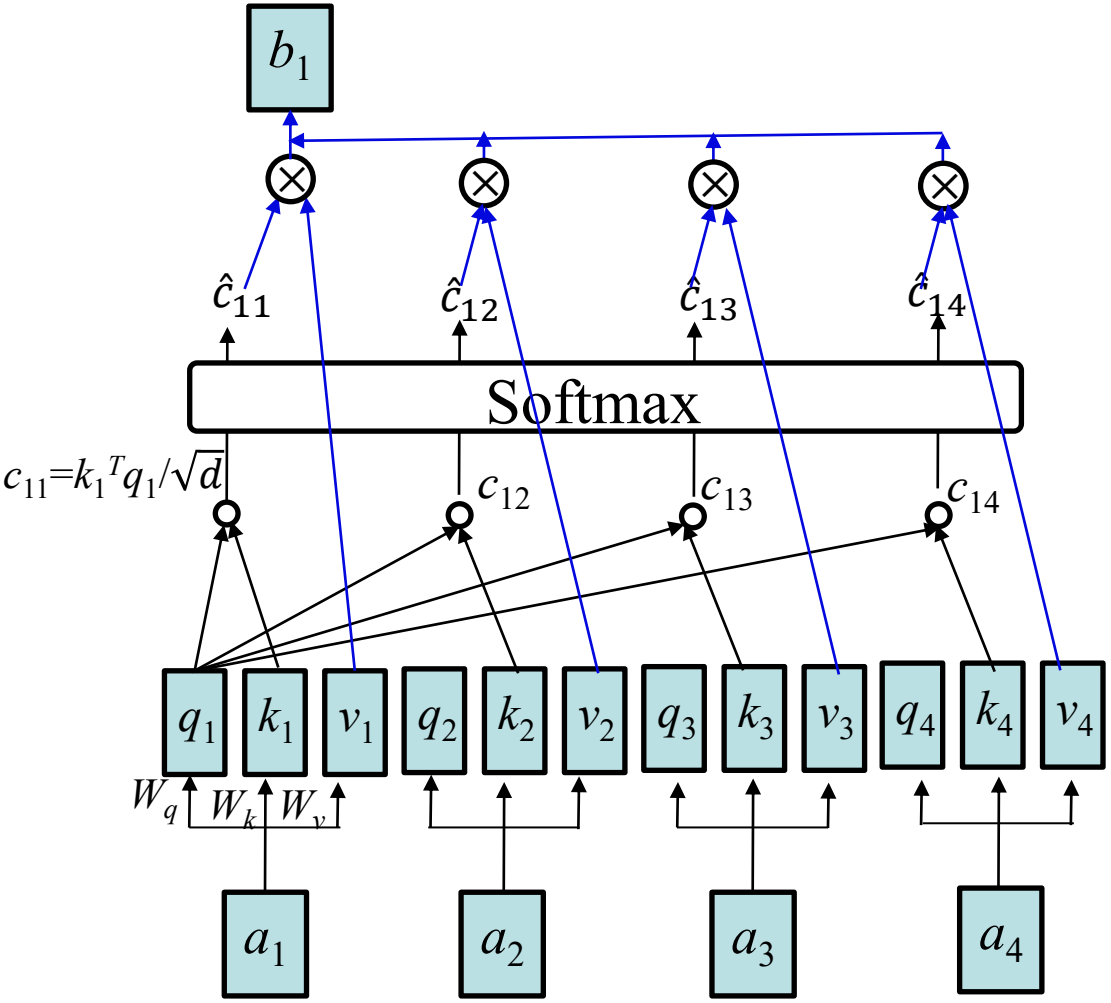
- **Self-attention layer in "Attention is all you need"**



注意力机制的抽象理解

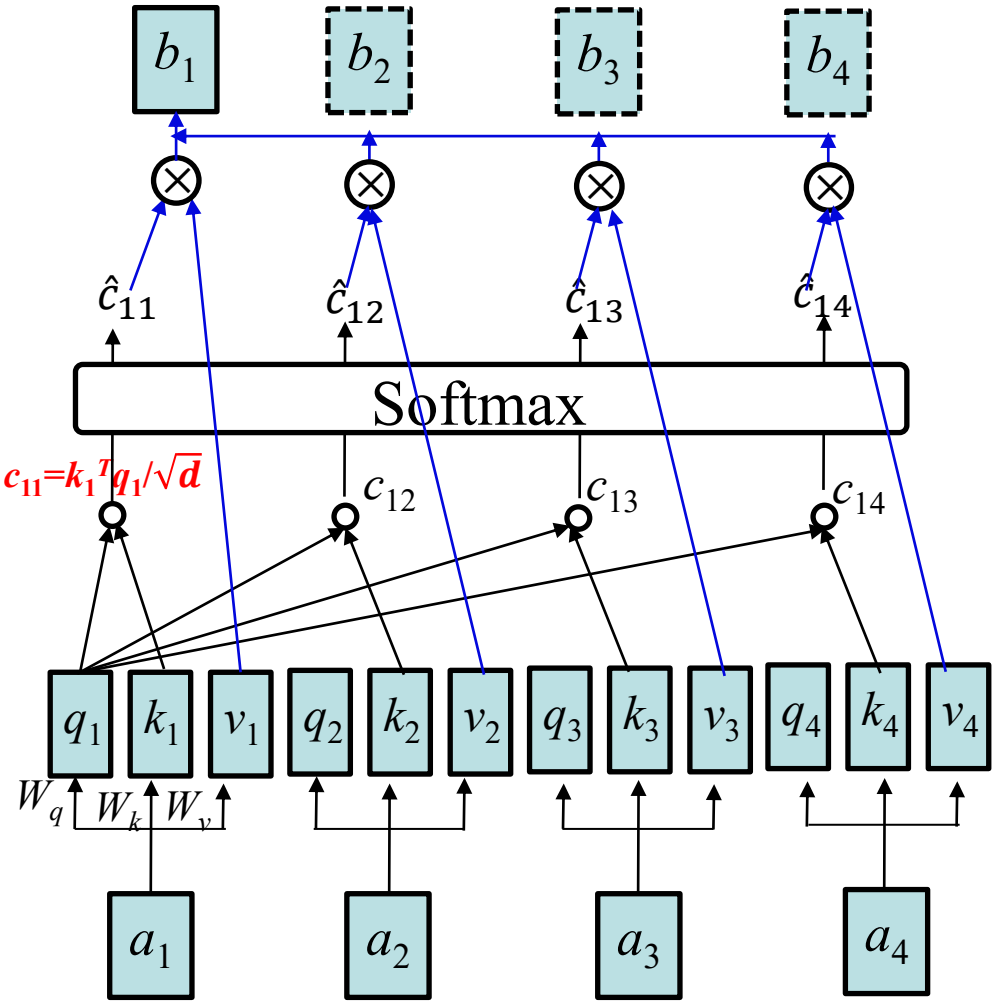
❑ **Self-attention layer in "Attention is all you need"**

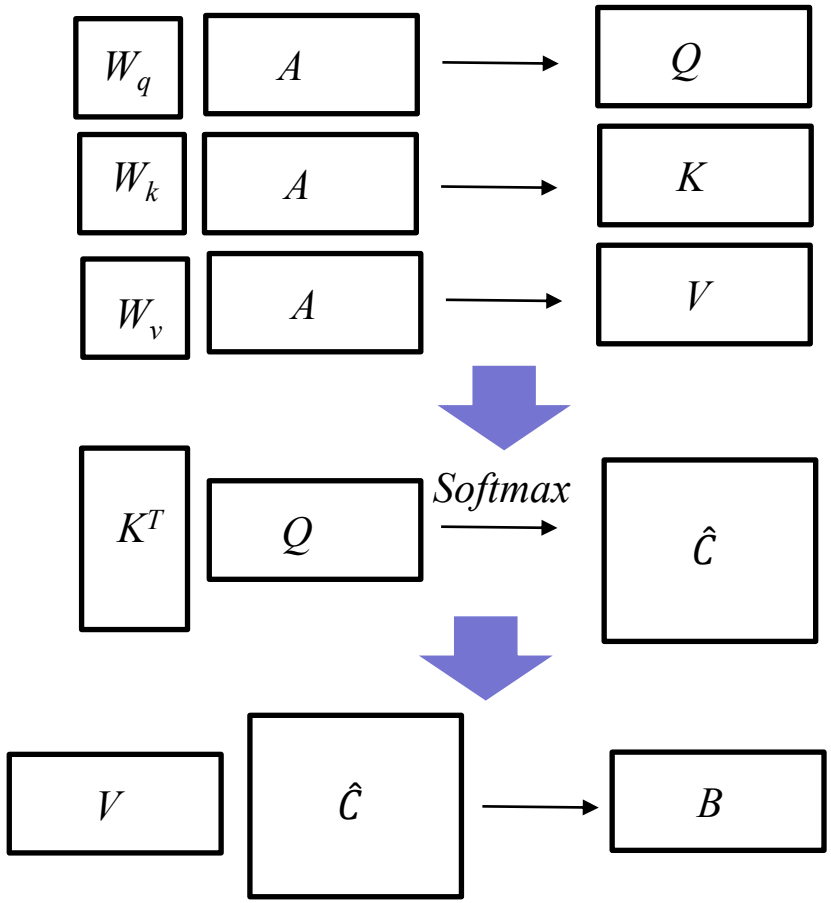




注意力机制的抽象理解

❑ **Self-attention layer in "Attention is all you need"**





注意力机制

❑ Transformer

Encoder

- 输入 $z \in R^{n \times d_{model}}$
- 输出大小不变
- Positional Encoding
- 6个Block
  - Multi-Head Self-Attention
  - Position-wise Feed Forward
  - Residual connection
    - LayerNorm(x + Sublayer(x))
    - 引入了残差，尽可能保留原始输入x的信息
- d_{model}=512

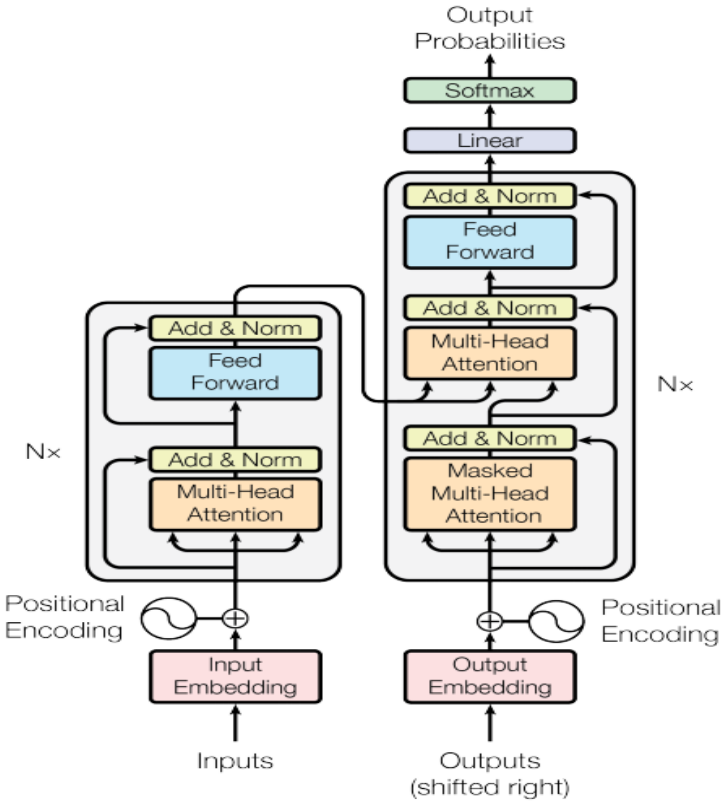


Figure 1: The Transformer - model architecture.

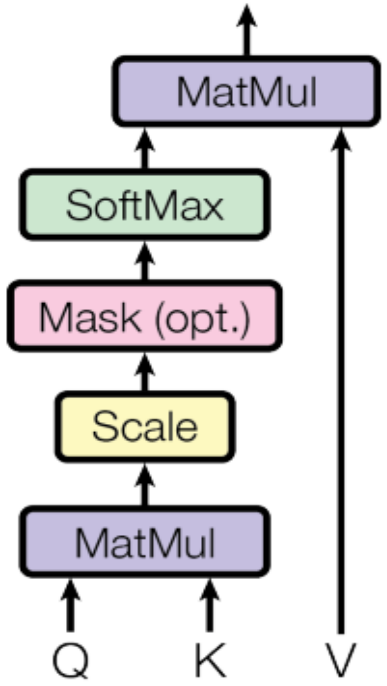
Decoder

- Positional Encoding
- 6个Block
  - Multi-Head Self Attention (with mask)
    ——采用 0-1mask 消除右侧 单词对当前单词 attention 的影响
  - Multi-Head Self Attention (with encoder)
    使用Encoder的输出作为一部分输入
  - Position-wise Feed Forward
  - Residual connection

Vaswani A, Shazeer N, Parmar N, et al. Attention is all you need[C]. Advances in Neural Information Processing Systems. 2017: 6000-6010.

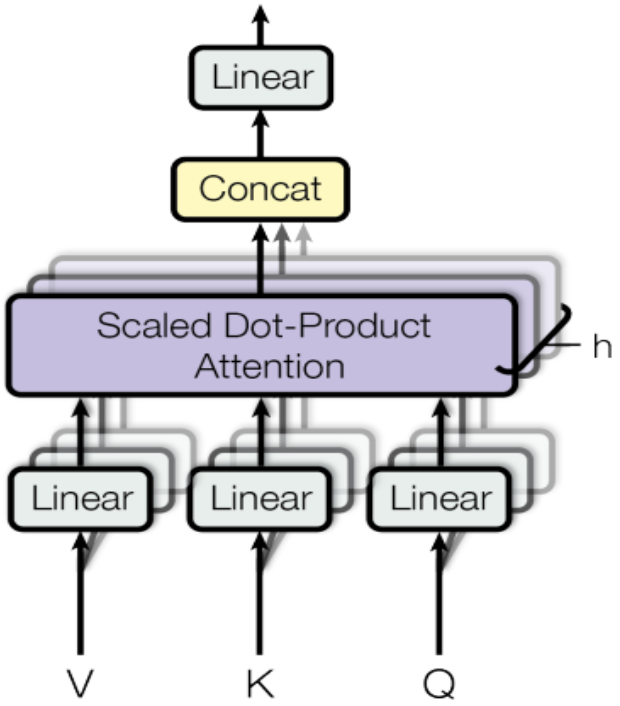
注意力机制

Scaled Dot-Product Attention



$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$

Multi-Head Attention

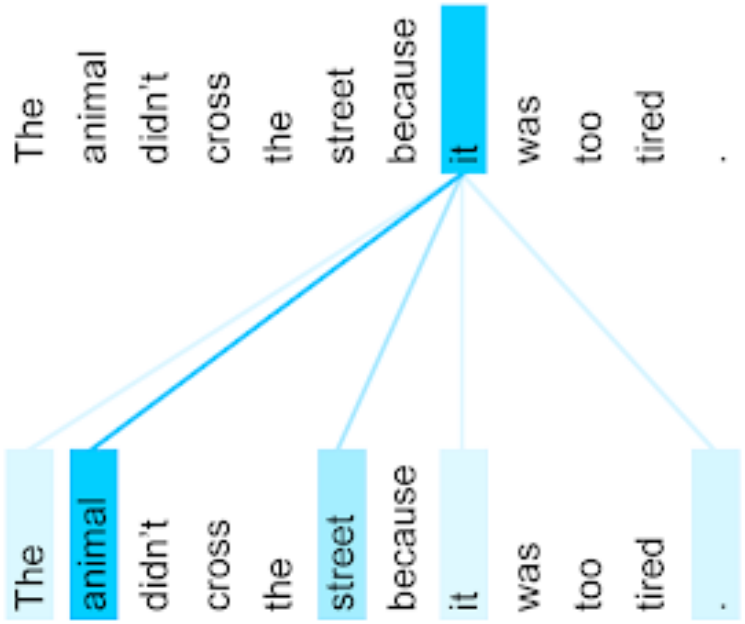


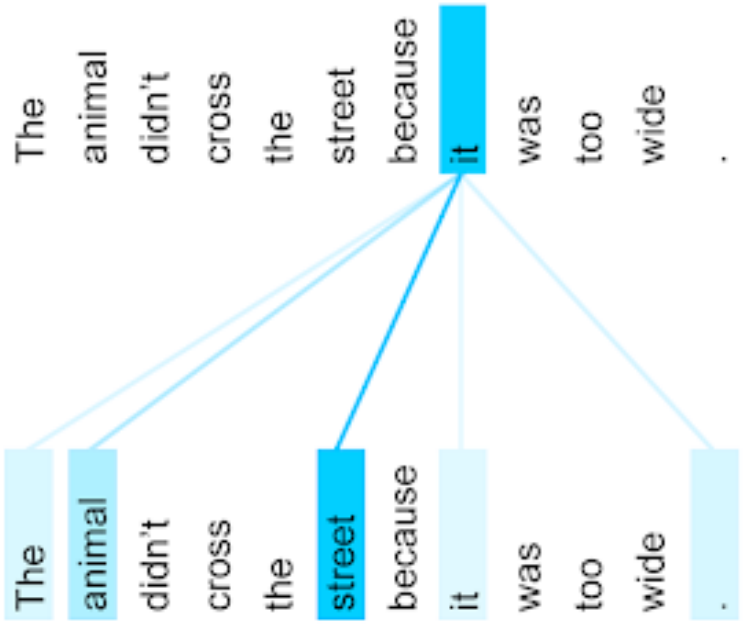
$$\text{head}_\text{i} = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, ..., \text{head}_\text{h})W^O$$

注意力机制

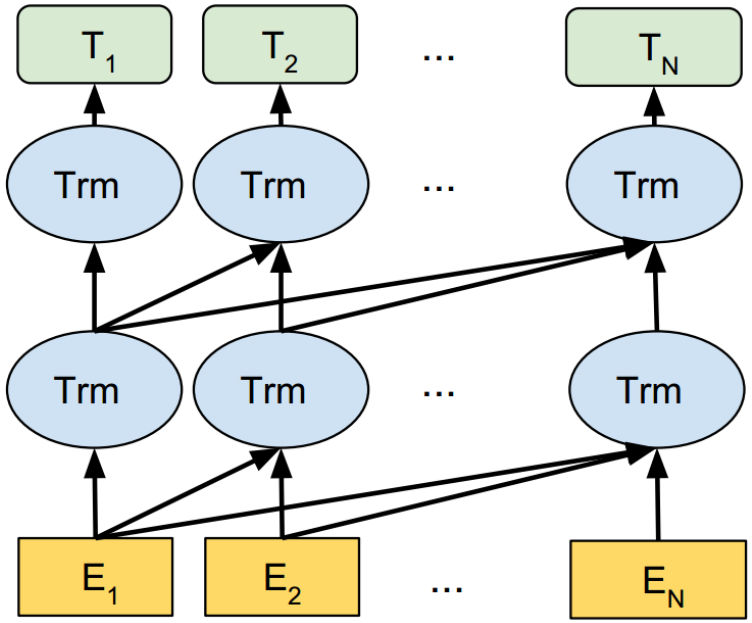
- 注意力机制可视化



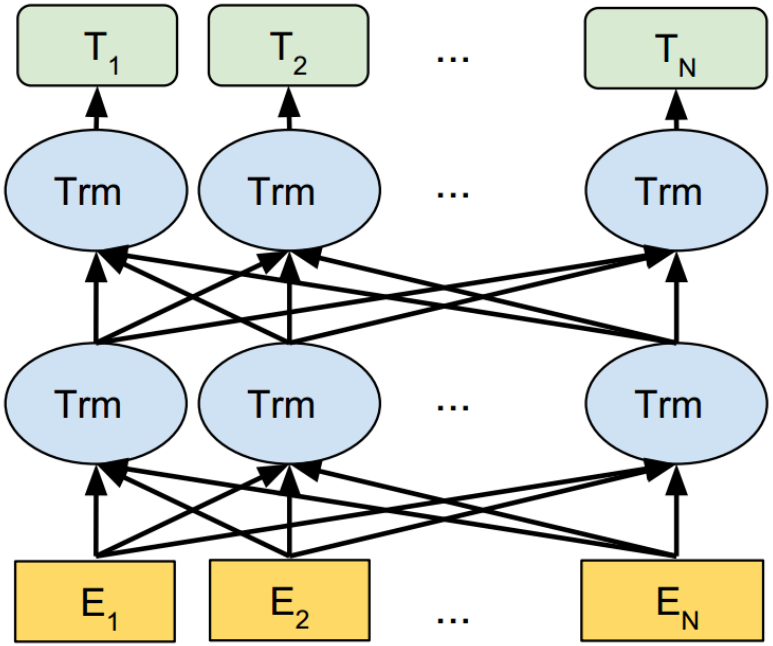


注意力机制的应用

❑ **GPT/GPT-2**



❑ **BERT**



中国科学院大学
University of Chinese Academy of Sciences

4 记忆网络

记忆网络产生背景

- 现代计算机的一个巨大优势就是可以对信息进行存储。但是，大多数机器学习模型缺少这种可以读取、写入的长期记忆的内存结构
- RNN、LSTM这样的神经网络原则上可以实现记忆存储，但是，它们由隐藏状态和权重编码包含的记忆太小了，不能记忆足够信息
- 基于此，Facebook AI Research提出了一种用于问答任务的记忆网络，实现了记忆的存储
- 广义上讲，循环神经网络也是记忆网络的一种

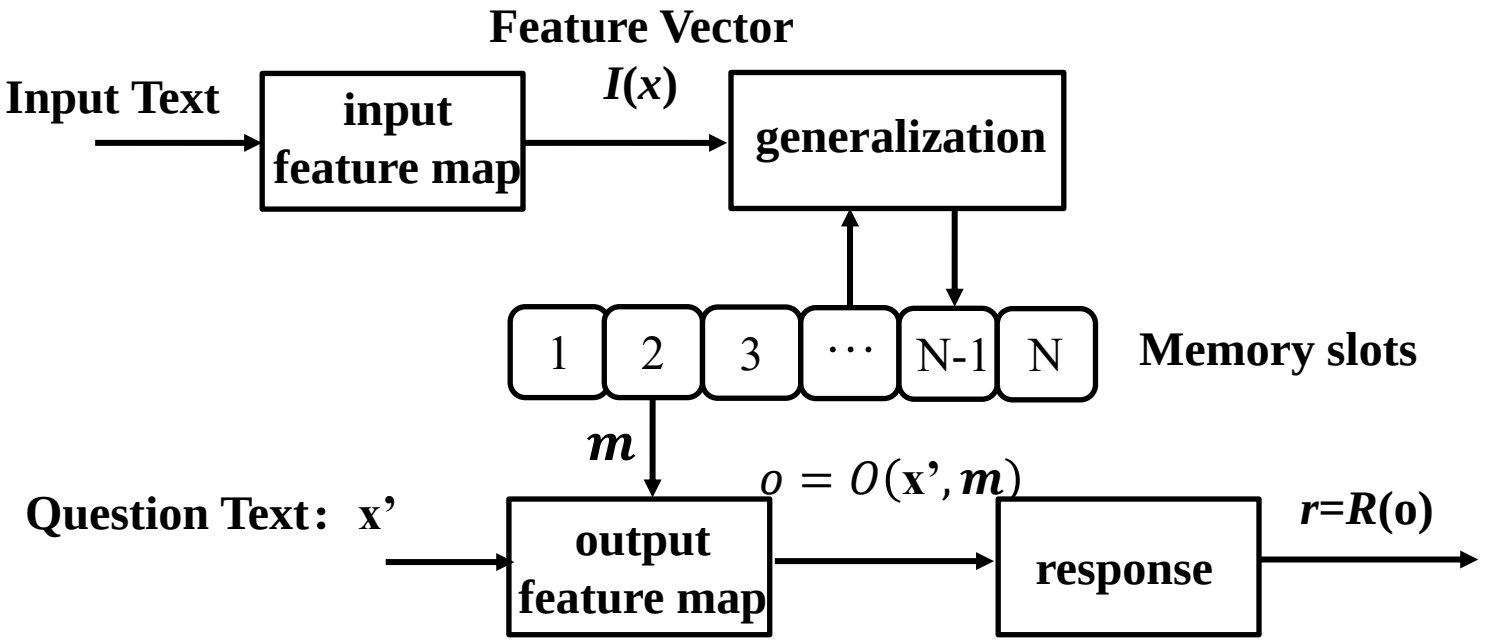
记忆网络代表工作

- Memory Networks. ICLR 2015
- End-To-End Memory Networks. NIPS 2015: 2440-2448
- Key-Value Memory Networks for Directly Reading Documents. EMNLP 2016: 1400-1409
- Tracking the World State with Recurrent Entity Networks. ICLR 2017

记忆网络结构

❑ 结构示意图

– I, G, O, R四个模块



Weston, Jason, Sumit Chopra, and Antoine Bordes. "Memory networks." arXiv preprint arXiv:1410.3916 (2014).

记忆网络结构

- **Input vector**
  - 将输入$\boldsymbol{x}$（字符、单词、句子等不同的粒度）转成内部特征向量的表示$I(x)$
- **Generalization**
  - 根据新的输入更新记忆单元中的memory slot $\boldsymbol{m_i}$

$$\boldsymbol{m_i} = G(\boldsymbol{m_i}, \boldsymbol{I(x)}, \boldsymbol{m}), \forall i$$

$$e.g., \ \boldsymbol{m}_{H(x)} = I(x),$$

记忆网络结构

❑ Output feature map

- 根据记忆单元和新的输入，输出特征 $\mathbf{o} = \mathrm{O}(\mathbf{x}', \boldsymbol{m})$
- 例如：也可以选出top k个

$$o_1 = Q_1(\mathbf{x}', \boldsymbol{m}) = \max_{i=1,\ldots,N} s_O(\mathbf{x}', \boldsymbol{m})$$

❑ Response

- 最后，解码输出特征o，并给出对应的响应 $r = R(\mathbf{o})$
- R可以是一个RNN网络生成回答的句子
- 更简单的话可以计算相关分数, 比如W是一个单词集合（dictionary）

$$r = \max_{w \in W} s_R(\mathbf{x}', o_1, o_2, w)$$

记忆网络结构

- 例子：
  - 训练：输入句子和标记的response

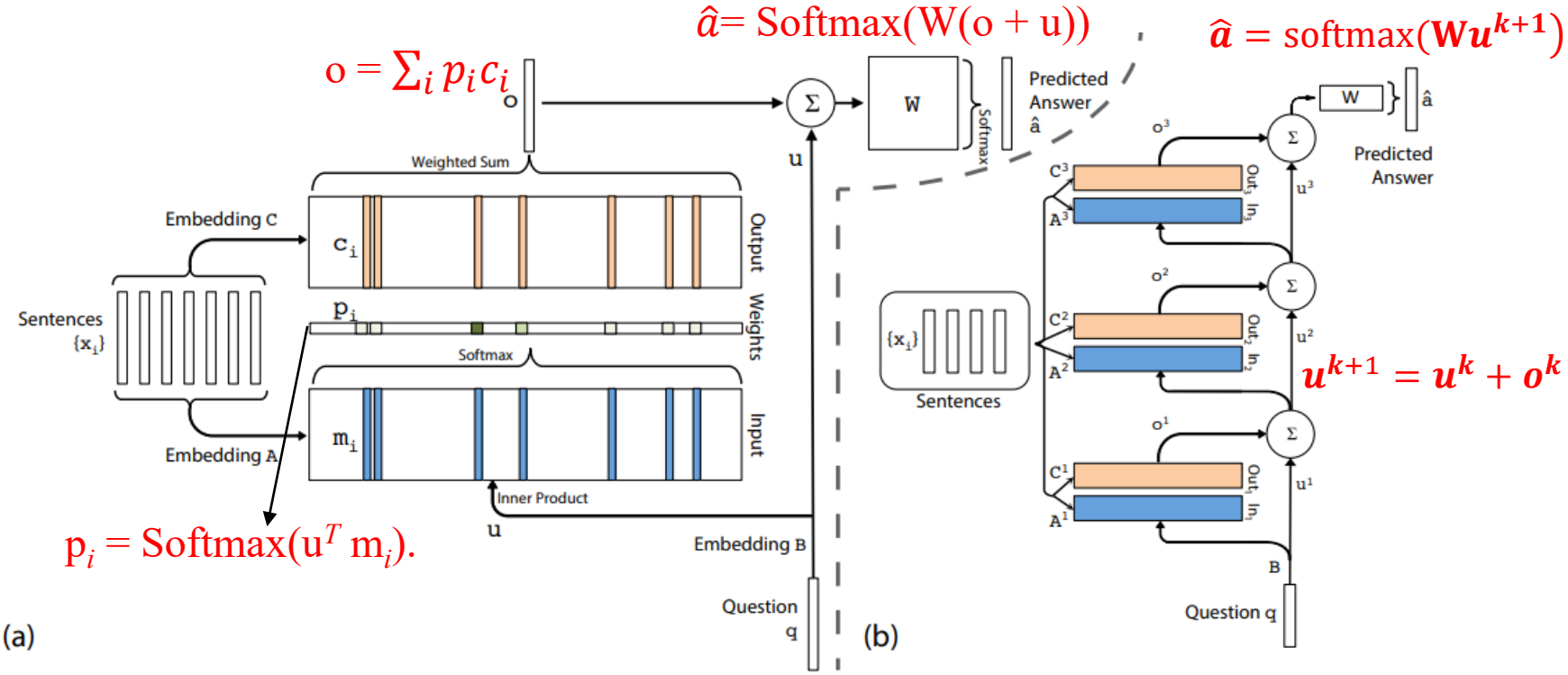
Figure 1: Example "story" statements, questions and answers generated by a simple simulation. Answering the question about the location of the milk requires comprehension of the actions "picked up" and "left". The questions also require comprehension of the time elements of the story, e.g., to answer "where was Joe before the office?".

Joe went to the kitchen. Fred went to the kitchen. Joe picked up the milk.
Joe travelled to the office. Joe left the milk. Joe went to the bathroom.
Where is the milk now? A: office
Where is Joe? A: bathroom
Where was Joe before the office? A: kitchen

端到端的记忆网络

❑ 单层和三层网络

- 输入$x_1,x_2,\ldots,x_n$存储到memory中， input memory$\{m_i\}$, output memory $\{c_i\}$; 问题q, 回答 a
- 用连续的权重代替了hard max



Sukhbaatar, Sainbayar, Arthur Szlam, Jason Weston, and Rob Fergus. "End-to-end memory networks." arXiv preprint arXiv:1503.08895 (2015).

端到端的记忆网络

- **训练参数**
  - A: input embedding matrix
  - C: output embedding matrix
  - W: answer prediction matrix
  - B: question embedding matrix
- **损失函数：交叉熵**
- **约束限制**
  - 为了减少训练参数量
    - Adjacent：$A^{k+1} = C^k$
    - Layer-wise：$A^1 = A^2 = \ldots = A^K$, $C^1 = C^2 = \ldots = C^K$

Key-Value记忆网络

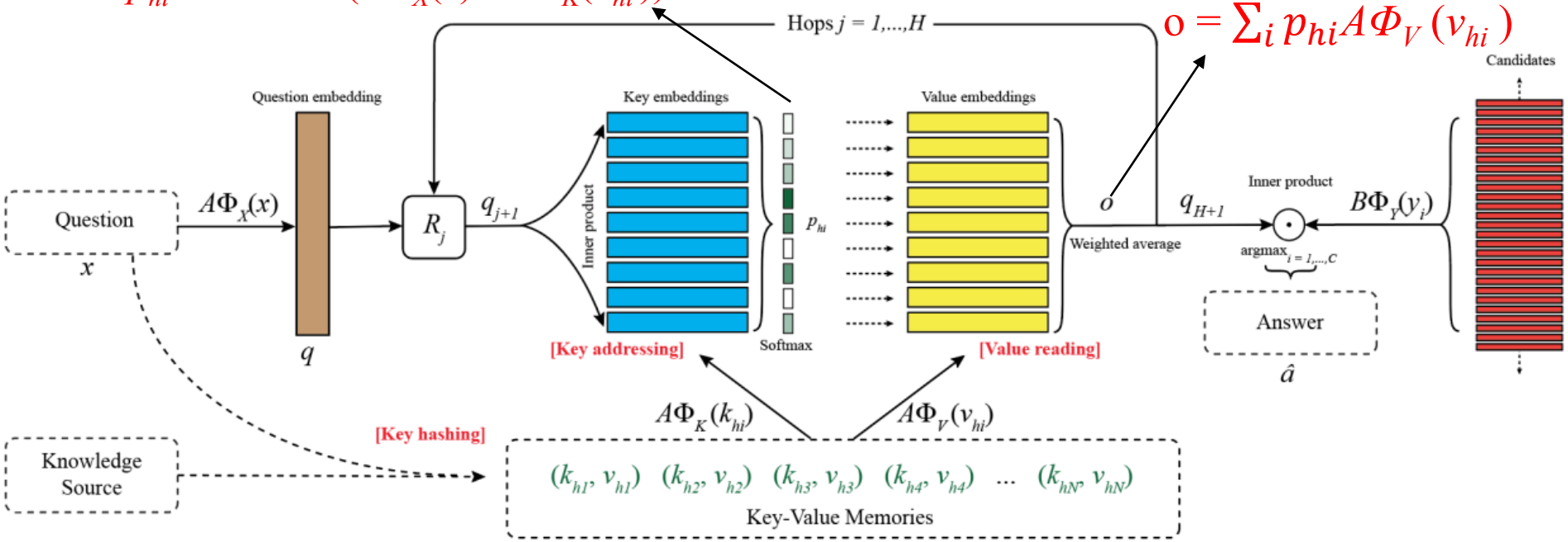
❑ 面向QA的Key-Value网络结构

– 高效的知识存储和检索

- key负责寻址lookup，也就是对memory与Question的相关程度进行评分，
- value则负责reading，也就是对记忆的值进行加权求和得到输出

$p_{hi} = \text{Softmax}(A\Phi_X(x) \cdot A\Phi_K(k_{hi}))$

$o = \sum_i p_{hi} A\Phi_V(v_{hi})$



其它记忆网络

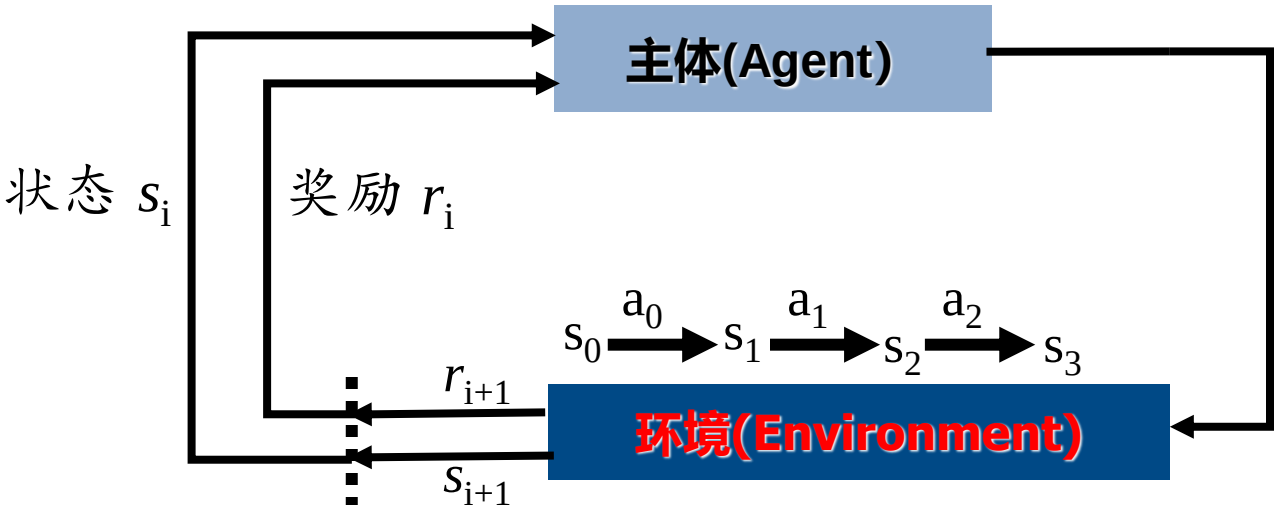
- 2017年，Facebook AI Research提出了一种新的基于记忆网络的循环实体网络，其使用固定长度的记忆单元来存储世界上的实体，主要存储该实体相关的属性，且该记忆会随着输入内容实时更新
- Google DeepMind也提出了多种记忆网络，如：Neural Turing Machines、Neural Random Access Machines以及使用像栈或（双端）队列结构的连续版本等
- 利用外部存储形式的机器学习方式已经成为机器学习领域中一个热点方向

中国科学院大学
University of Chinese Academy of Sciences

5 深度强化学习

强化学习(Reinforcement learning)

❑ 网络结构



r: reward

s: state

a: action

强化学习

❑ 马尔科夫决策过程包含五个元素：

- 状态集S：比如：物体位置坐标
- 动作集A：a 表示某个特定动作，比如：执行上下左右等一系列可行的动作集。
- 状态转移概率P_a：P_a表示在状态$s \in S$下执行动作$a \in A$后，转移到下一个状态的概率分布

$$P_a(s, s') = p(s_{t+1} = s' | s_t = s, a_t = a)$$

- 奖励函数R (reward), $R(s,a)$表示当前状态下执行某个动作能得到的奖励（回报）值
- 回报(Return) $U(s_0,s_1,\ldots)$:表示执行一组action后所有状态累积的reward之和

$$U(s_0, s_1 \cdots) = \sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) \leq \frac{R_{max}}{1-\gamma}$$

  - 阻尼系数γ，或者称为折扣因子(Discount factor), $\gamma \in [0,1]$

强化学习

- **agent的决策过程：**

$$s_0 \xrightarrow[a_0]{} s_1 \xrightarrow[a_1]{} s_2 \xrightarrow[a_2]{} s_3 \xrightarrow[a_4]{} \cdots$$

- **策略函数**
  - 输入是当前时刻的环境信息，输出是要执行的动作 $a = \pi(s)$
  - 某种状态下执行某个动作的概率 $\pi(a|s) = P(a|s)$
- **选择一个最佳的策略使得回报最大：**

$$\max_{\pi} E[R(s_0, a_0) + \gamma R(s_1, a_1) + \gamma^2 R(s_2, a_2) + \cdots]$$

  - 总的回报值U

$$\mathrm{U} = R(s_0, a_0) + \gamma R(s_1, a_1) + \gamma^2 R(s_2, a_2) + \cdots$$

强化学习

- **定义状态价值函数V**
  - V为在当前状态s下执行策略π能得到的期望的累积折扣回报：

$$V_\pi(s) = E(U_t|S_t = s) = \sum_{a\in A}\pi(a|s)\left(R_s^a + \gamma\sum_{s'\in S}P_{ss'}^a V_\pi(s')\right)$$

当前回报和后续回报

$$U_t = R_{t+1} + \gamma R_{t+1} + \cdots + \gamma^{T-t-1}R_T$$

$$P_{ss'}^a = P(S_{t+1} = s'|S_t = s, A_t = a)$$

- **假设有一个最优策略π，使得我们的得到一个最优的值函数(Bellman方程）：**

$$V_*(s) = \max_a\left(R_s^a + \gamma\sum_{s'\in S}P_{ss'}^a V_*(s')\right)$$

强化学习算法

❑ 时序差分学习 (Temporal-Difference Learning, TD)算法

- 在执行一个动作之后进行状态价值函数更新，算法无需依赖状态转移概率，直接通过生成随机的样本来计算，用Bellman方程估计价值函数的值，然后构造更新项
- Sutton's TD（0）算法：考虑当前回报和下一状态的估计值
  - 更新公式如下：

$$V(S_t) := V(S_t) + \alpha[R_{t+1} + \gamma V(S_{t+1}) - V(S_t)]$$

其中公式右侧的$V(S_t)$是$V(S_t)$的旧值（值迭代的过程），α是学习效率

强化学习算法

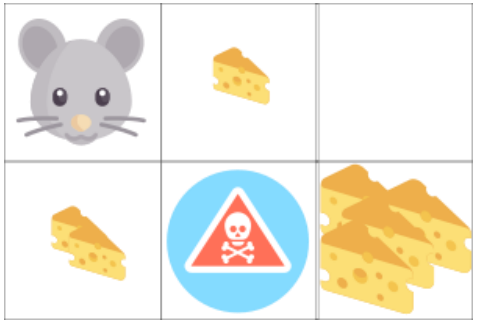
❑ Q-learing算法

– 它和Sutton's TD(0)算法类似，定义动作价值函数Q(*s*,*a*)，Q-learning的更新规则：

$$Q(S_t,a) := Q(S_t,a) + \alpha\left[R_{t+1} + \gamma \max_{a'} Q(S_{t+1},a') - Q(S_t,a)\right]$$

– 动作价值函数$Q(S_t,a)$表示当前状态采取动作a时得到未来的reward

– 一个简单示例：



	left	right	up	down
start	0	0.1	0	0
1 cheese	0	0	0	0
Nothing	0	0	0	0
2 cheese	0	0	0	0
Death	0	0	0	0
Many cheese	0	0	0	0

$$NewQ(start,right) = Q(start,right) + \alpha[\Delta Q(start,right)]$$

$$\Delta Q(start,right) = R(start,right) + \gamma\ max\ Q'(1cheese,a')) - Q(start,right)$$

$$\Delta Q(start,right) = 1+0.9*max(Q'(1cheese,left),Q'(1cheese,right),Q'(1cheese,down)) - Q(start,right)$$

$$\Delta Q(start,right) = 1+0.9*0-0 = 1$$

$$NewQ(start,right) = 0+0.1*1 = 0.1$$

强化学习算法

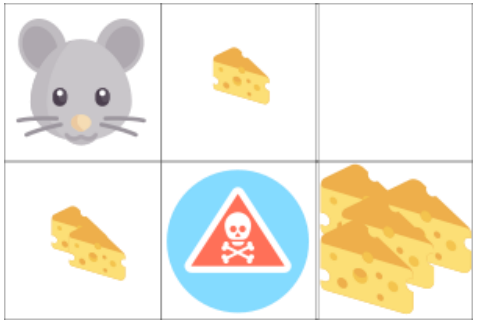
❑ Q-learing算法

– 它和Sutton's TD(0)算法类似，定义动作价值函数Q(*s*,*a*)，Q-learning的更新规则：

$$Q(s,a) := Q(s,a) + \alpha\left[R(s,a) + \gamma \max_{a'} Q(S_{t+1}, a') - Q(s,a)\right]$$

– 动作价值函数$Q(S_t, a)$表示当前状态采取动作a时得到未来的reward

– 一个简单示例：



初始化 Q = {};
while Q 未收敛：
　初始化老鼠的位置s，
　开始新一轮游戏 while s != 死亡状态：
　使用策略π，获得动作a=π(s)
　使用动作a进行游戏，获得老鼠的新位置s',与奖励R(s,a)
　Q[s,a] ← Q[s,a] + α*(R(s,a) + γ* max Q[s',a']) //
　更新Q s ← s'
endwhile

– 其他方法：SARSA、Monte-Carlo、Actor-Critic算法…

深度强化学习

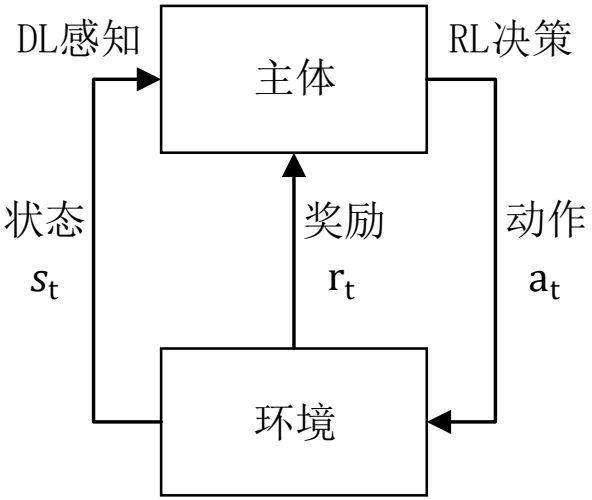
❑ 深度强化学习是深度学习和强化学习的结合

- 深度学习具有较强的感知能力，但是缺乏一定的决策能力
- 强化学习具有决策能力，但对感知问题却束手无策
- 深度强化学习（Deep Reinforcement Learning，DRL）将深度学习的感知能力和强化学习的决策能力相结合，可以直接根据输入的状态进行控制，是一种更接近人类思维方式的人工智能方法
- 深度强化学习目前侧重在强化学习上，解决的仍然是决策问题，只不过是借助神经网络强大的表征能力去拟合Q表或直接拟合策略以解决状态-动作空间过大或连续状态-动作空间问题

深度强化学习原理框架

❑ 深度强化学习基本过程

- 利用深度学习方法来感知状态，以得到具体的状态特征表示
- 基于预期回报来评价各动作的价值函数
- 通过某种策略做出相应的动作，环境对此动作做出反应，并得到下一个状态与回报
- 通过不断循环以上过程，最终可以得到实现目标的最优策略



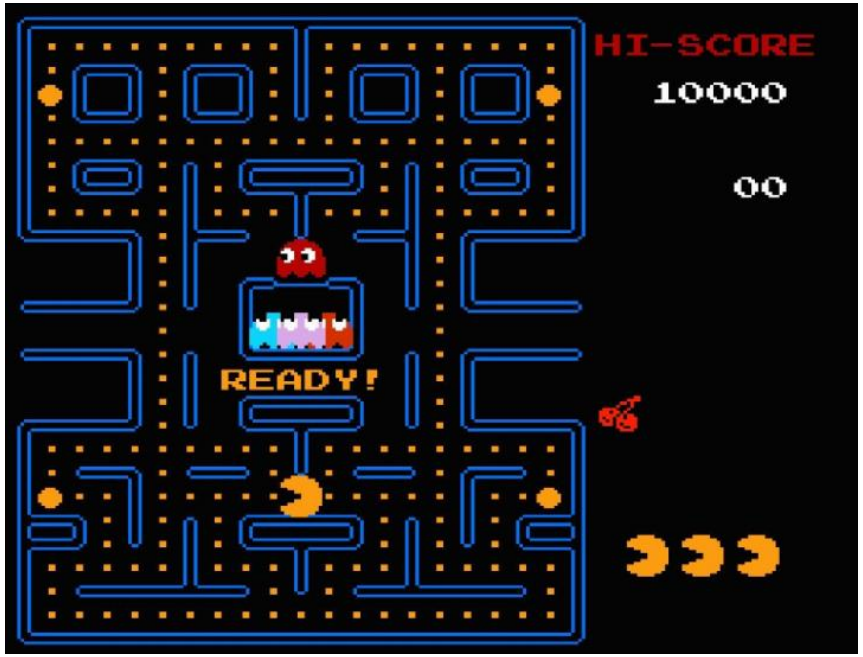
深度Q网络（DQN）

❑ 深度Q网络（Deep Q Network，DQN）

- 一个较为典型的DRL算法
- DQN融合了神经网络和Q learning的方法
- DQN把Q learning中的价值函数用深度神经网络近似。除此之外，DQN算法还做了经验回放（Experience Replay），即将系统探索环境得到的数据储存起来，然后随机采样样本更新深度神经网络的参数。这是一种离线学习法，它能学习当前经历着的，也能学习过去经历过的，甚至是学习别人的经历。随机采样这种做法打乱了经历之间的相关性，也使得神经网络更新更有效率
- Human-level control through deep reinforcement learning, nature 2015

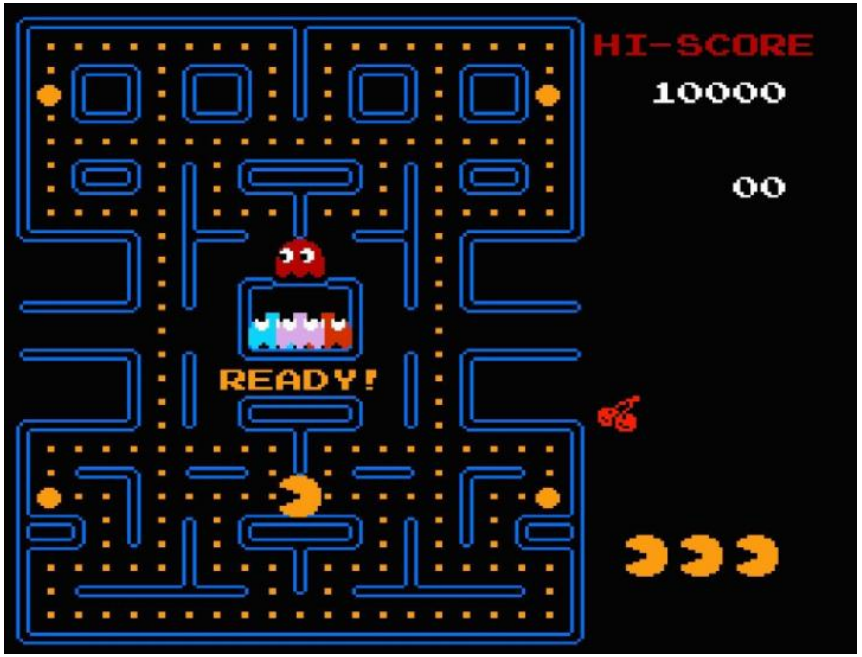
深度Q网络（DQN）

- **算法可以通过观察Atari 2600的游戏画面和得分信息，自主的学会玩游戏**
  - 因为输入是RGB，像素也是高维度，因此，对图像进行初步的图像处理，变成灰度矩形84*84的图像作为输入，有利于卷积

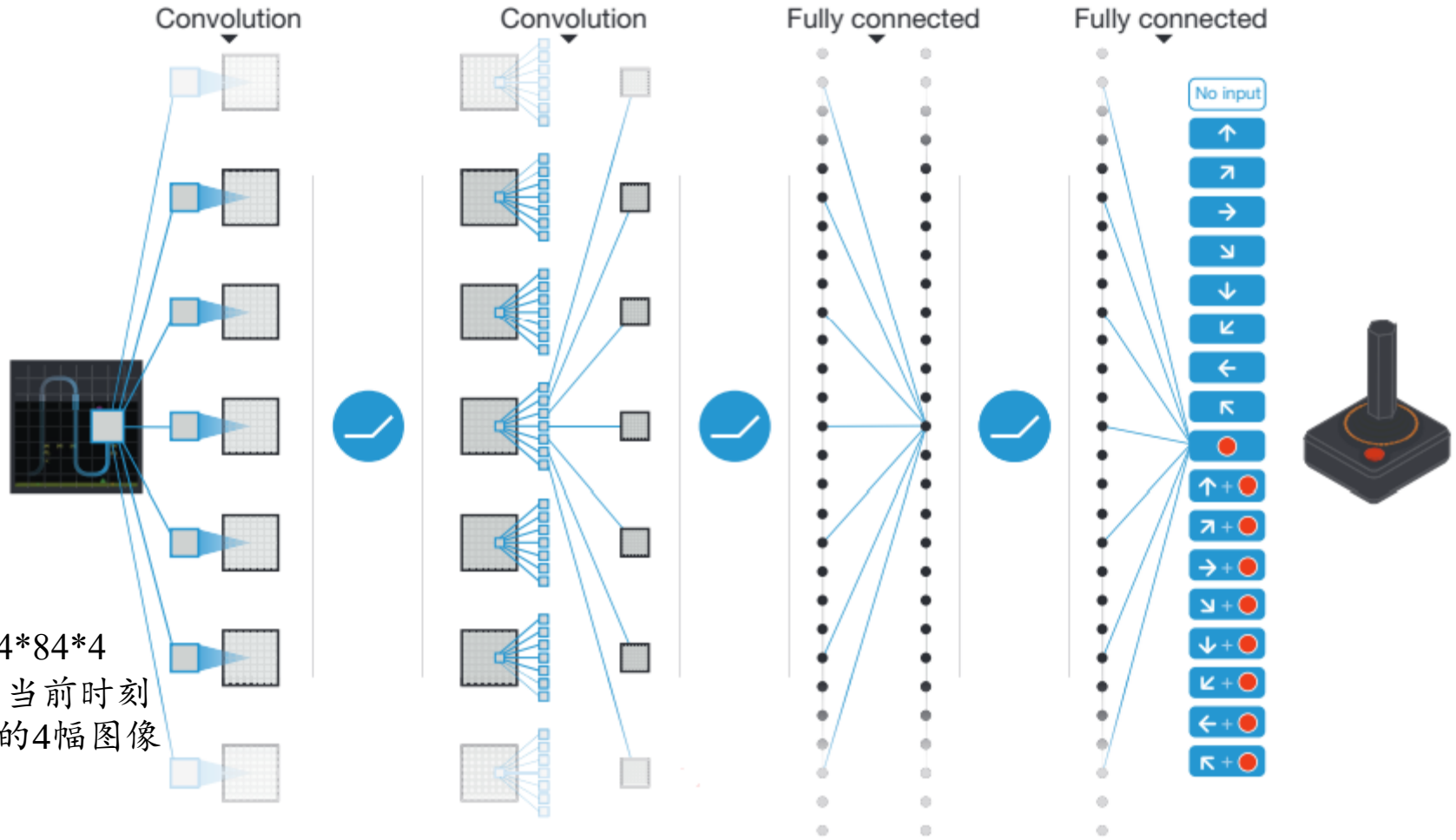


深度Q网络（DQN）

- **算法可以通过观察Atari 2600的游戏画面和得分信息，自主的学会玩游戏**
  - 接下来就是模型的构建问题，毕竟Q(s,a)包含s和a。一种方法就是输入s和a，输出q值，这样并不方便，每个a都需要forward一遍网络
  - Deepmind的做法是神经网络只输入s，输出则是每个a对应的q。这种做法的优点就是只要输入s，forward前向传播一遍就可以获取所有a的q值，毕竟a的数量有限



深度Q网络（DQN）


Convolution
Convolution
Fully connected
Fully connected
No input
84*84*4
距离当前时刻
最近的4幅图像

深度Q网络（DQN）

❑ 双网络结构

- Main DQN（主网络/当前值网络）：通过其最大Q值选择Action，而这个被选定的Action的Q值则由target DQN生成
- Target DQN（目标网络）：辅助计算目标Q值，这样做的目的是避免让网络训练陷入目标Q值与预测Q值的反馈循环中

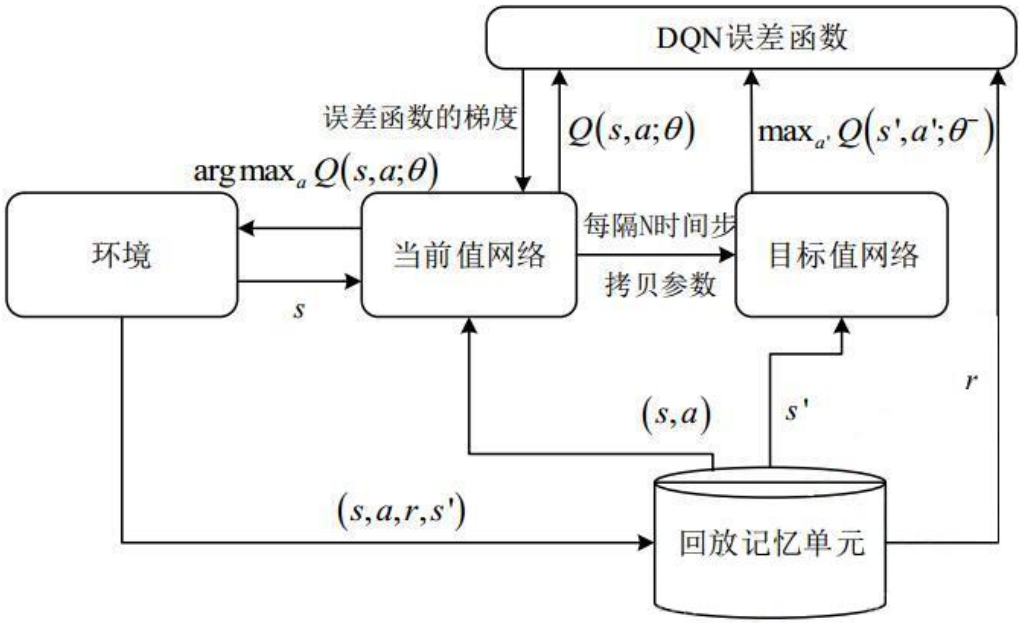
深度Q网络（DQN）

❑ 双网络结构

- Main DQN（主网络/当前值网络）：通过其最大Q值选择Action，而这个被选定的Action的Q值则由target DQN生成
- Target DQN（目标网络）：辅助计算目标Q值，这样做的目的是避免让网络训练陷入目标Q值与预测Q值的反馈循环中

$$Y_i = r + \gamma \max_{a'} Q(s', a' | \theta_i^-)$$

$$L(\theta_i) = \mathrm{E}_{s,a,r,s'}\left[(Y_i - Q(s, a|\theta_i))^2\right]$$



其它深度强化学习方法

- **对于时间序列信息，深度Q网络的处理方法是加入经验回放机制。但是经验回放的记忆能力有限，每个决策点需要获取整个输入画面进行感知记忆**
  - 将长短时记忆网络与深度Q网络结合，提出深度递归Q网络(Deep Recurrent Q Network，DRQN)，在部分可观测马尔科夫决策过程中表现出了更好的鲁棒性，同时在缺失若干帧画面的情况下也能获得很好的实验结果。
  - 受此启发的深度注意力递归Q网络(Deep Attention Recurrent Q network，DARQN)。它能够选择性地重点关注相关信息区域，减少深度神经网络的参数数量和计算开销

中国科学院大学
University of Chinese Academy of Sciences

6 深度森林

决策树

❑ 决策树(Decision Tree)

- 是一种基于树结构进行决策的机器学习方法，这恰是人类面临决策时一种很自然的处理机制

❑ 决策树生成过程：

- 寻找最适合分割的特征
- 根据纯度判断方法，寻找最优的分割点，基于这一特征把数据分割成纯度更高的两部分数据
- 判断是否达到要求，若未达到，重复步骤一继续分割，直到达到要求停止为止
- 剪枝，防止过拟合

随机森林

❑ 随机森林（Random Forest）

- 用随机的方式建立起一棵棵决策树，然后由这些决策树组成一个森林，其中每棵决策树之间没有关联，当有一个新的样本输入时，就让每棵树独立的做出判断，按照多数原则决定该样本的分类结果。这是一种典型的集成学习思想
- 集成学习(Ensemble learning)
  - 组合多个弱监督模型以期得到一个更好更全面的强监督模型，集成学习潜在的思想是即便某一个弱分类器得到了错误的预测，其他的弱分类器也可以将错误纠正回来

随机森林

❑ 随机森林的生成方法

- 从样本集中通过重采样的方式产生n个样本
- 假设样本特征数目为a，对n个样本选择a中的k个特征，用建立决策树的方式获得最佳分割点(例如ID3算法)
- 重复m次，产生m棵决策树

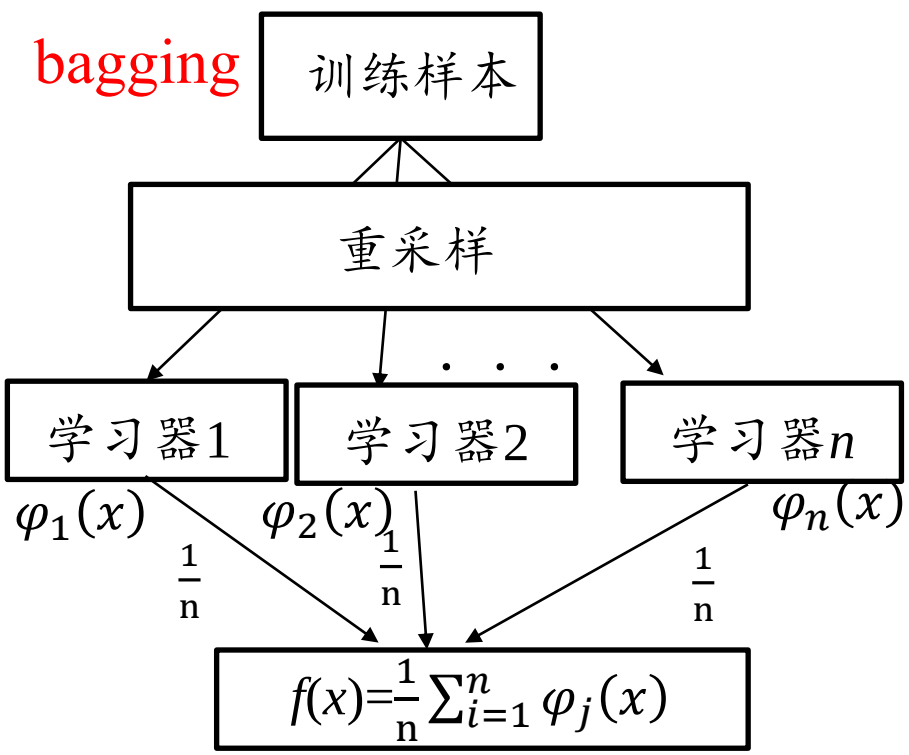
❑ 预测步骤：多数投票机制来进行预测

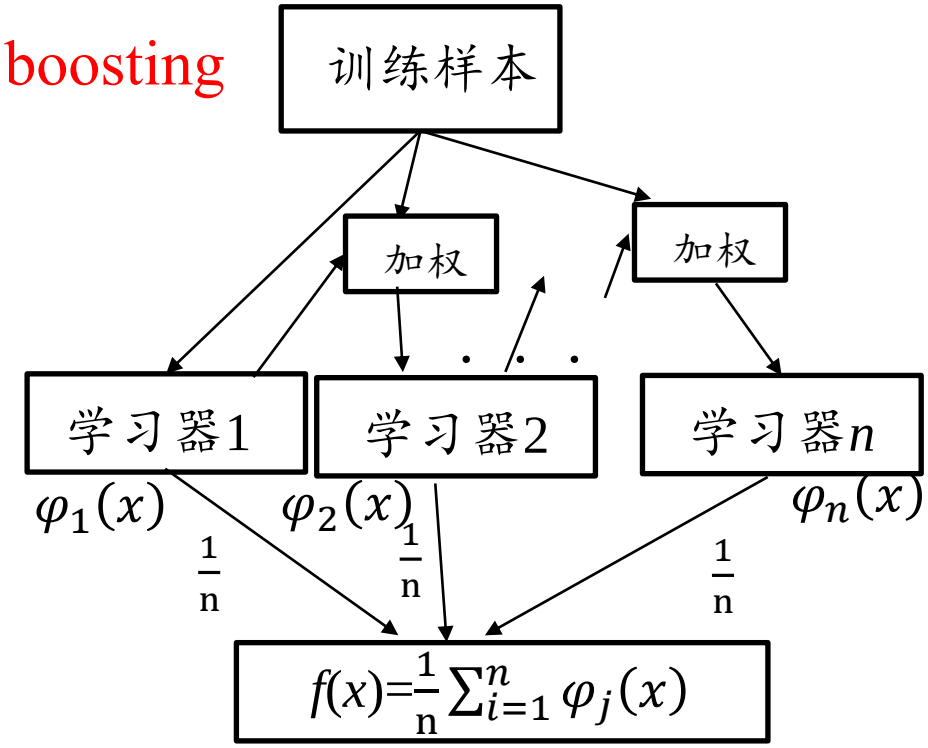
- 向建立好的随机森林中输入一个新样本
- 随机森林中的每棵决策树都独立的做出判断
- 将得到票数最多的分类结果作为该样本最终的类别

梯度提升决策树

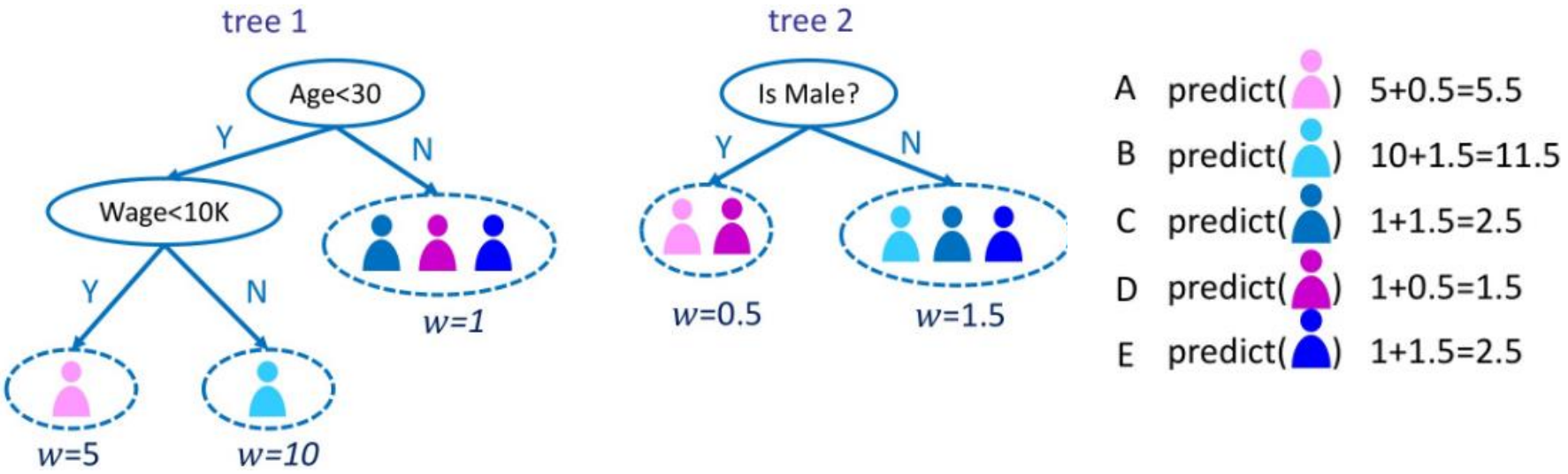
❑ 梯度提升决策树 (Gradient Boosting Decision Tree，GBDT)

- 是一种集成使用多个弱分类器（决策树）来提升分类效果的机器学习算法，所有树的结论累加起来作为最终结果
- 在很多分类和回归的场景中，表现不错且泛化能力较强





GBDT：消费者的消费力预测



1. 在第一棵树中，根节点选取的特征是年龄，年龄小于30的被分为左子节点，年龄大于30的被分为右叶子节点，右叶子节点的预测值为1；
2. 第一棵树的左一节点继续分裂，分裂特征是月薪，小于10K划分为左叶子节点，预测值为5；工资大于10k的划分右叶子节点，预测值为10
3. 建立完第一棵树之后，C、D和E的预测值被更新为1，A为5，B为10
4. 根据新的预测值，开始建立第二棵树，第二棵树的根节点的性别，女性预测值为0.5，男性预测值为1.5
5. 建立完第二棵树之后，将第二棵树的预测值加到每个消费者已有的预测值上，比如A的预测值为两棵树的预测值之和：5+0.5=5.5

深度森林

- 神经网络可以堆叠为深度神经网络，并且取得了显著的效果。那我们可以考虑，是不是可以将其他的学习模型堆叠起来，以获取更好的表示性能，深度森林模型就是基于这种想法提出来的一种深度结构

- **2017**年，深度森林由周志华老师提出，基于树模型的方法，主要使用集成学习思想方法的深度学习框架，也可作为一种在某些任务下替代深度神经网络的方法
  - Deep Forest: Towards an Alternative to Deep Neural Networks, 2017

深度森林

❑ 深度神经网络的问题

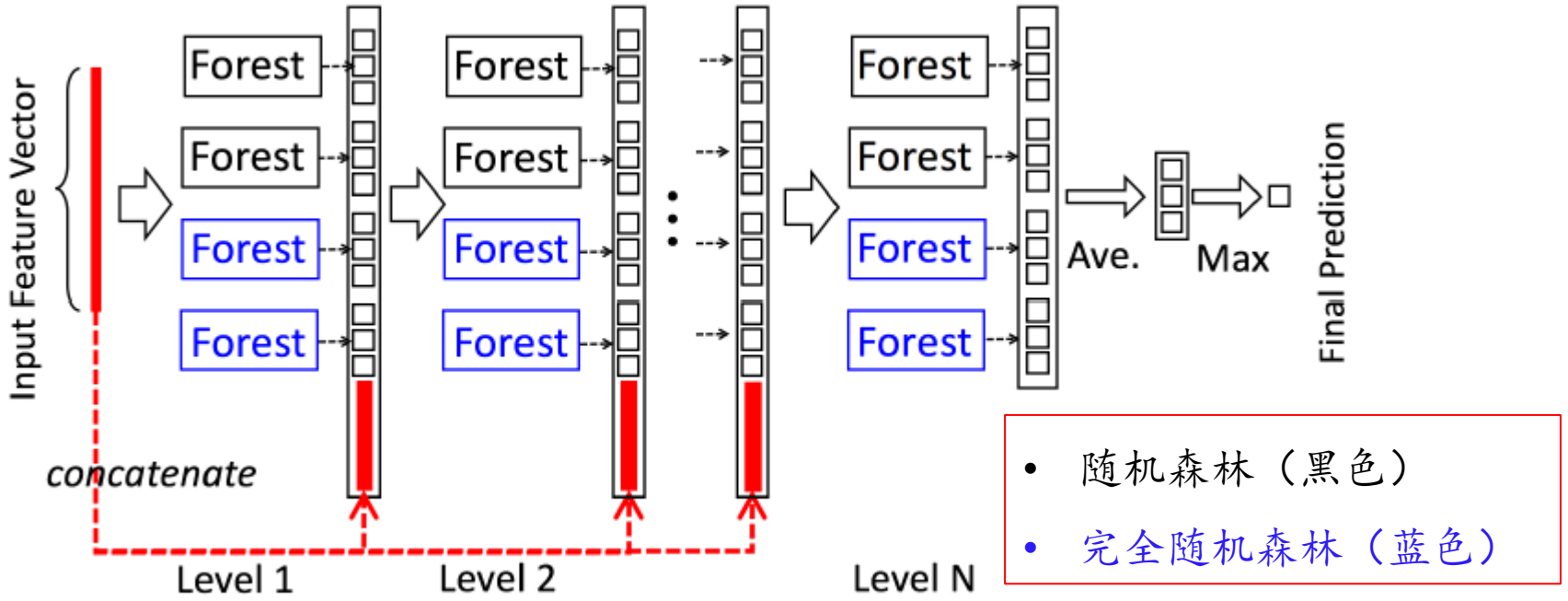
- 需要使用大量的数据进行训练
- 计算资源消耗大
- 可解释性差
- 超参数多，训练依赖于经验

❑ 深度森林的优势

- 与DNN相比需要的参数更少
- 训练速度快
- 不仅适合大规模数据也适合小规模数据
- 基于树模型解释性比较好

多粒度级联森林

- **GCForest（muti-Grained Cascade Forest，多粒度级联森林），它是基于树的集成方法，通过对树组成的森林来集成并前后串联起来达到表征学习的效果**



上图每层包含两个随机森林，随机森林和完全随机森林，处理一个三分类的问题

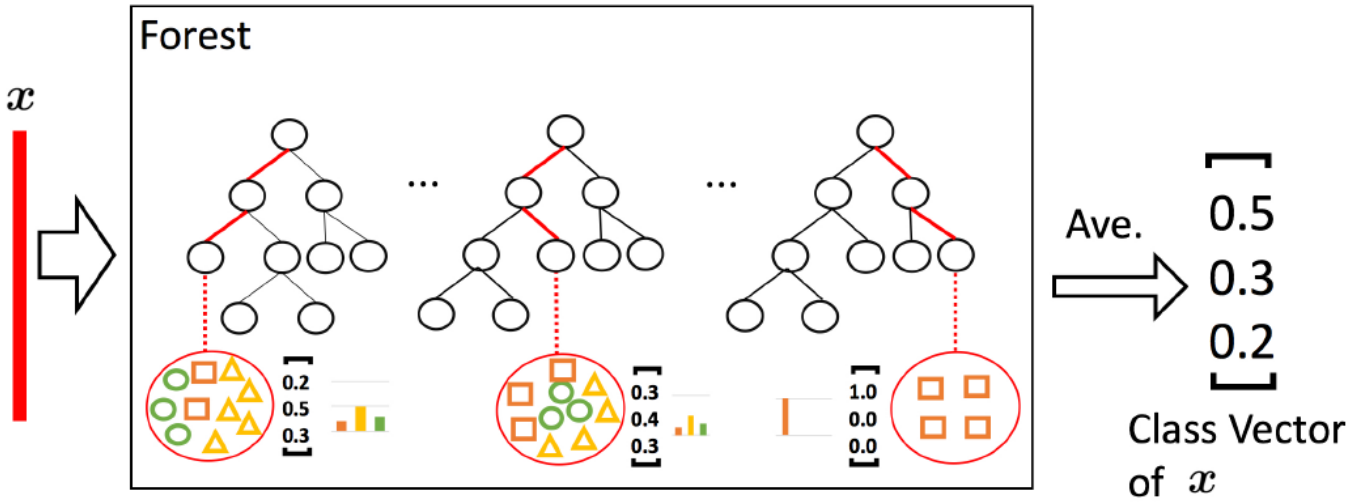
多粒度级联森林

- 每个完全随机的森林包含完全随机树，通过随机选择一个特征在树的每个节点进行分割实现生长，直到叶子节点只包含相同类的实例。
- 类似的，每个随机森林包含的树，通过随机选择sqrt(d)数量的特征作为候选（d是输入特征的数量)，然后选择具有最佳gini值的特征作为分割
- 每个森林中的树的数值是一个超参数

多粒度级联森林

❑ GCForest每个森林的类分布向量生成流程：

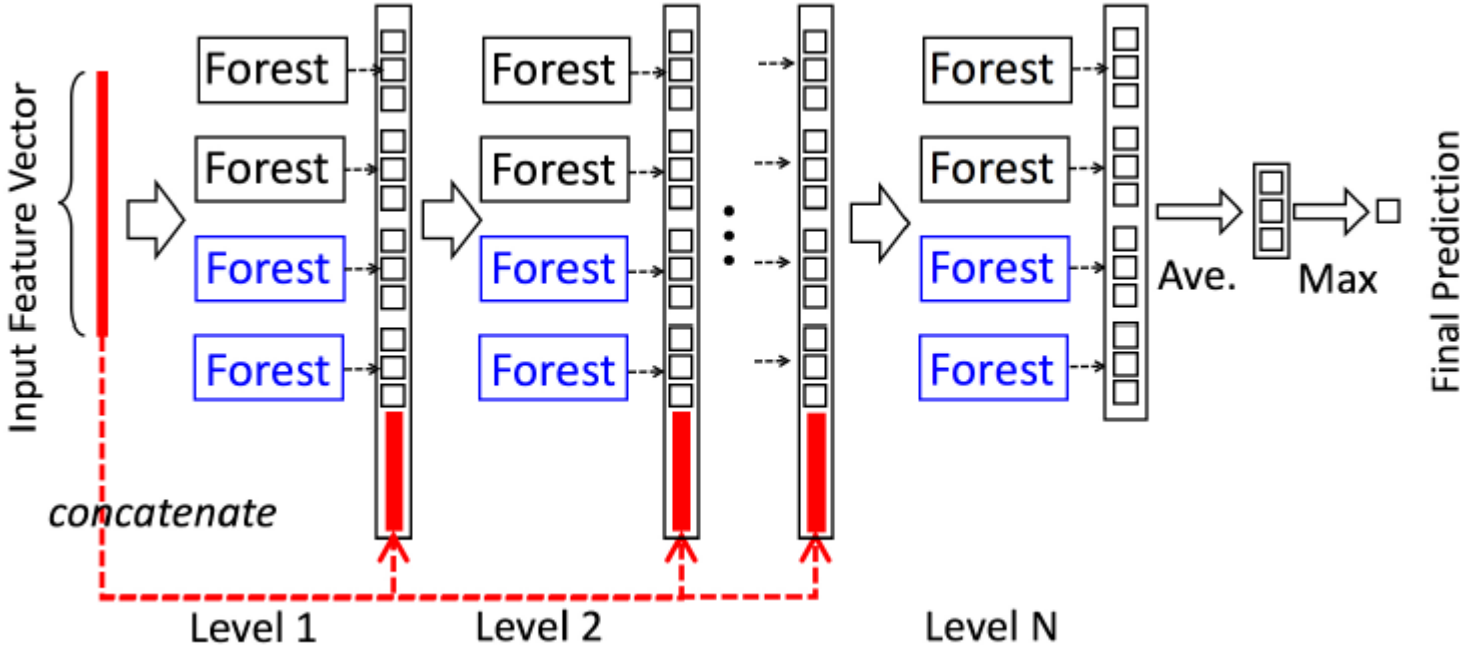
- 对于测试样本，每棵树会根据样本所在的子空间中训练样本的类别占比生成一个类别的概率分布，然后对森林内所有树的各类比例取平均，输出整个森林对各类的比例
- 每个森林都会生成长度为C的概率向量，如果一层有N个森林，那么每个森林生成的C个元素会拼接在一起，组成C*N个元素向量，这就是一层的输出



多粒度级联森林

❑ GCForest每个森林的类分布向量生成流程：

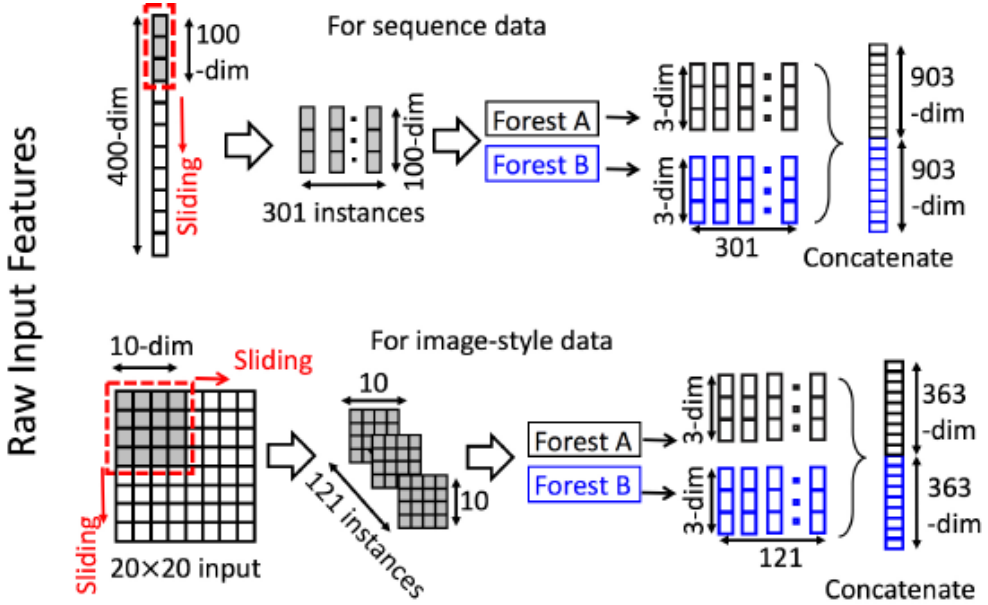
- 每个森林都会生成长度为C的概率向量，如果一层有N个森林，那么每个森林生成的C个元素会拼接在一起，组成C*N个元素向量，这就是一层的输出
- 这基础上把源输入特征向量拼接上去组成了下一层的输入



多粒度级联森林

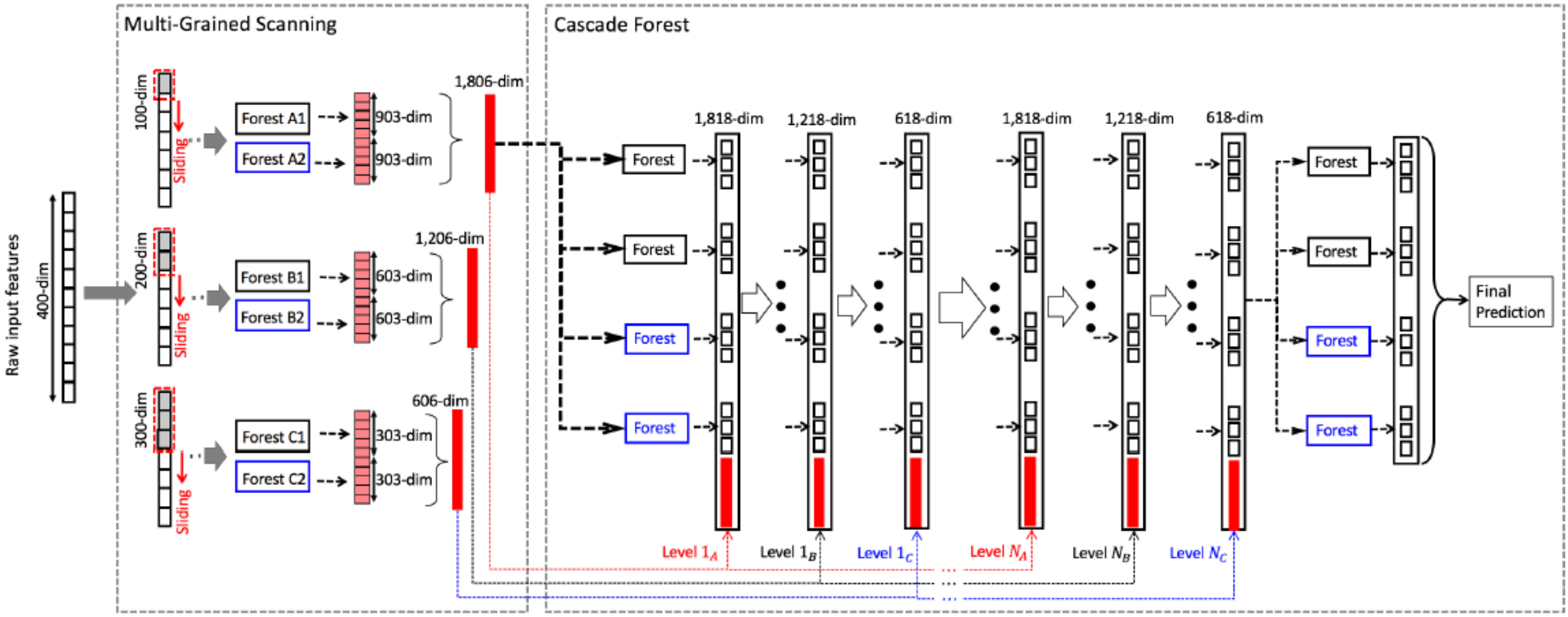
❑ GCForest的多粒度扫描：

- 采用滑动窗口的方法，首先生成若干个实例，其次通过实例生成两个森林，一个完全随机森林，一个随机森林，接着再把生成的两个森林生成对应的相同维度的“类向量”，最后把这两大类向量连接在一起。
- 一般来说会采用多个不同大小的窗口做扫描



多粒度级联森林

❑ 整体结构



多粒度级联森林

❑ 超参数对比

Deep neural networks (e.g., convolutional neural networks)	gcForest
Type of activation functions: Sigmoid, ReLU, tanh, linear, etc. Architecture configurations: **No. Hidden layers**: ? **No. Nodes in hidden layer**: ? **No. Feature maps**: ? **Kernel size**: ? Optimization configurations: **Learning rate**: ? Dropout: {0.25/0.50} **Momentum**: ? **L1/L2 weight regularization penalty**: ? Weight initialization: Uniform, glorot_normal, glorot_uni, etc. Batch size: {32/64/128}	Type of forests: Completely-random tree forest, random forest, etc. Forest in multi-grained scanning: **No. Forests**: {2} **No. Trees in each forest**: {500} Tree growth: till pure leaf, or reach depth 100 **Sliding window size**: $\{\lfloor d/16 \rfloor, \lfloor d/8 \rfloor, \lfloor d/4 \rfloor\}$ Forest in cascade: **No. Forests**: {8} **No. Trees in each forest**: {500} Tree growth: till pure leaf

多粒度级联森林

❑ 相关性能

– 手写数字识别(MNIST)

gcForest	**99.26%**
LeNet-5	99.05%
Deep Belief Net	98.75% [Hinton *et al.*, 2006]
SVM (rbf kernel)	98.60%
Random Forest	96.80%

– 人脸识别(ORL)

	5 image	7 images	9 images
gcForest	**91.00%**	**96.67%**	**97.50%**
Random Forest	91.00%	93.33%	95.00%
CNN	86.50%	91.67%	95.00%
SVM (rbf kernel)	80.50%	82.50%	85.00%
kNN	76.00%	83.33%	92.50%

– 音乐分类(GTZAN)

gcForest	**65.67%**
CNN	59.20%
MLP	58.00%
Random Forest	50.33%
Logistic Regression	50.00%
SVM (rbf kernel)	18.33%

– 情感分类(IMDB)

gcForest	**89.16%**
CNN	89.02% [Kim, 2014]
MLP	88.04%
Logistic Regression	88.62%
SVM (linear kernel)	87.56%
Random Forest	85.32%

基于森林的自编码器

❑ 自编码器（Auto-Encoder）

– 是神经网络的一种，是一种重要的表示学习模型，是深度学习的关键要素之一。自编码器的基本结构是由一个编码器（encoder）和一个解码器(decoder)组成，其中encoder将输入映射到隐空间，decoder将隐空间的表示重构为原表示

❑ eForest（Encoder-Forest）

– 基于森林的自编码器，能够利用决策树的决策路径所定义的等效类来进行后向重建。利用决策树集成算法进行向前编码和向后解码的操作

– AutoEncoder by Forest, AAAI 2017

基于森林的自编码器

❑ 前向编码

- 在一个有N个已训练决策树的森林中，前向编码过程接受输入数据并将其发送到集成方法中每棵树的根结点
- 一旦数据遍历（traverse）到所有树的叶结点，该过程将返回 T 维向量，T中的每个元素 t 是树 t 中的叶结点的整数索引

基于森林的自编码器

- 前向编码

Algorithm 1: Forward Encoding

Input: A trained forest F with T trees,
An input data x

Output: x_{enc}

$x_{enc} \leftarrow zeros[T,1]$ % initialize x_{enc}

for i *in range*(T) **do**

  $x_{enc}[i] \leftarrow F.tree[i].get\_leaf\_index(x)$

end

return x_{enc}

基于森林的自编码器

❑ 后向解码

– 利用决策树进行决策时需要将决策的路径记录下来，而每个决策路径对应了一个规则（rule）

识别出来的路径用红色突出显示，每个路径对应一个符号规则。

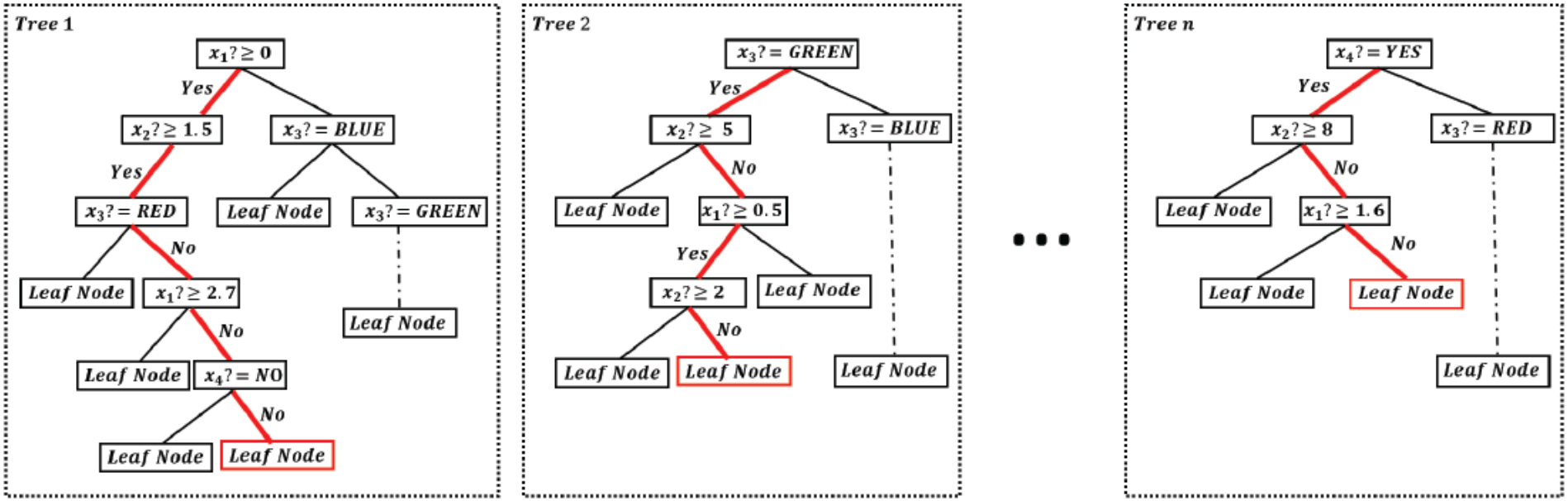


Figure 1: Traversing backward along decision paths

基于森林的自编码器

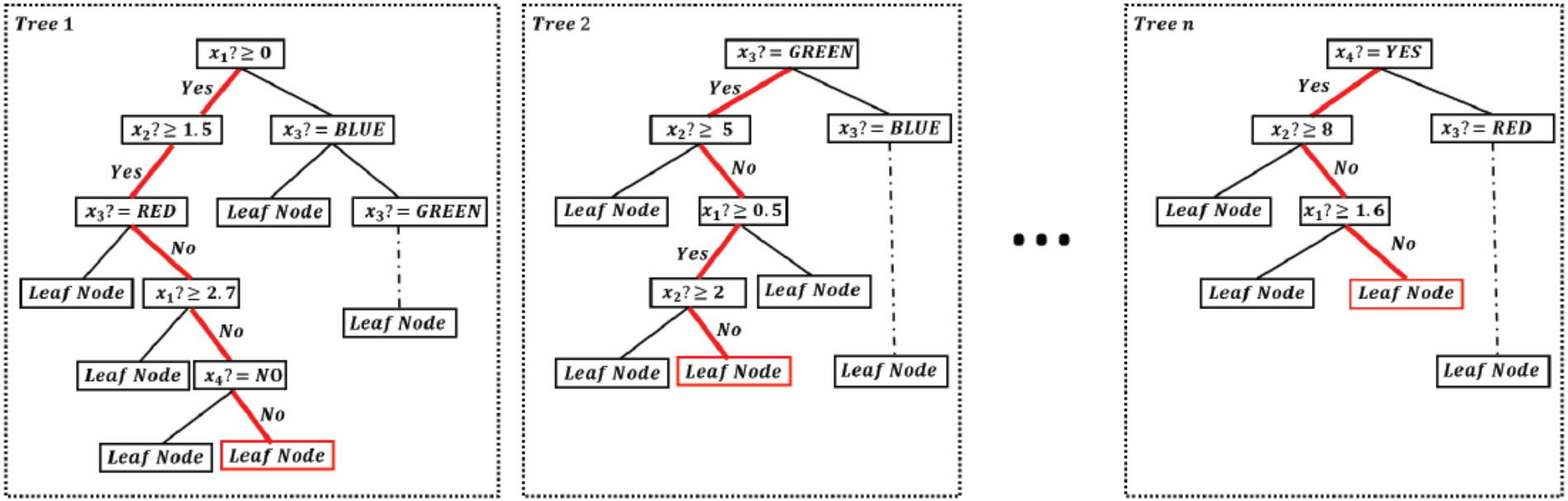
❑ 后向解码

– 利用决策树进行决策时需要将决策的路径记录下来，而每个决策路径对应了一个规则（rule）

$RULE_1$: $(x_1 \geq 0) \wedge (x_2 \geq 1.5) \wedge \neg(x_3 == RED) \wedge \neg(x_1 \geq 2.7) \wedge \neg(x_4 == NO)$

$RULE_2$: $(x_3 == GREEN) \wedge \neg(x_2 \geq 5) \wedge (x_1 \geq 0.5) \wedge \neg(x_2 \geq 2)$

$RULE_n : (x_4 == YES) \wedge \neg(x_2 \geq 8) \wedge \neg(x_1 \geq 1.6)$



基于森林的自编码器

❑ 后向解码

– 利用决策树进行决策时需要将决策的路径记录下来，而每个决策路径对应了一个规则（rule）

$$RULE_1\text{: } (x_1 \geq 0) \wedge (x_2 \geq 1.5) \wedge \neg(x_3 == RED) \wedge \neg(x_1 \geq 2.7) \wedge \neg(x_4 == NO)$$

$$RULE_2\text{: } (x_3 == GREEN) \wedge \neg(x_2 \geq 5) \wedge (x_1 \geq 0.5) \wedge \neg(x_2 \geq 2)$$

...

$$RULE_n : (x_4 == YES) \wedge \neg(x_2 \geq 8) \wedge \neg(x_1 \geq 1.6)$$

$$RULE'_1 : (2.7 > x_1 \geq 0) \wedge (x_2 \geq 1.5) \wedge \neg(x_3 == RED) \wedge (x_4 == YES)$$

$$RULE'_2 : (x_1 \geq 0.5) \wedge \neg(x_2 \geq 2) \wedge (x_3 == GREEN)$$

...

$$RULE'_n : \neg(x_1 \geq 1.6) \wedge \neg(x_2 \geq 8) \wedge (x_4 == YES)$$

基于森林的自编码器

❑ 后向解码

- 利用决策树进行决策时需要将决策的路径记录下来，而每个决策路径对应了一个规则（rule）
- 在进行解码重构的过程中，利用森林中N个规则，运用最大相容规则Maximum Compatible Rule（MCR），获取N维编码所对应的更精确的规则，以此规则进行重构

$$RULE'_1 : (2.7 > x_1 \geq 0) \wedge (x_2 \geq 1.5) \wedge \neg(x_3 == RED) \wedge (x_4 == YES)$$

$$RULE'_2 : (x_1 \geq 0.5) \wedge \neg(x_2 \geq 2) \wedge (x_3 == GREEN)$$

...

$$RULE'_n : \neg(x_1 \geq 1.6) \wedge \neg(x_2 \geq 8) \wedge (x_4 == YES)$$

$$(1.6 > x_1 \geq 0.5) \wedge (2 > x_2 \geq 1.5) \wedge (x_3 == GREEN) \wedge (x_4 == YES)$$

基于森林的自编码器

- 后向解码—计算出最大相容规则MCR

Algorithm 2: Calculate MCR

Input: A path rule list $Rule\_List$ consists of T rules defined by a forest with T trees

Output: MCR

$MCR \leftarrow initialize\_list()$

for i *in* $range(T)$ **do**

  $path\_rule \leftarrow rule\_list[i]$

  for *node_rule in path_rule.node_rule_list* **do**

    $j \leftarrow node\_rule.get\_attribute()$

    $bound \leftarrow node\_rule.get\_bound()$

    $MCR[j] \leftarrow intersect(MCR[j], bound)$

  end

end

return MCR

基于森林的自编码器

❑ 后向解码

- 对于类别属性，例如x_3和x_4原始样本就取MCR中的值
- 对于数值属性，例如x_2，可以去区间内的代表值
- 那么重构的样本x=[0.5,1.5,GREEN, YES]

Algorithm 3: Backward Decoding

Input: x_{enc}, trained eForest F with T trees
Output: x_{dec}
rule_list $\leftarrow$ *initialize_list*()
for i *in range*(T) **do**
    $path \leftarrow$ F.tree[i].*get_path*($x_{enc}[i]$)
    $path\_rule \leftarrow calculate\_rules(path)$
    $path\_rule \leftarrow simplify(path\_rule)$
    $rule\_list.append(path\_rule)$
end
$MCR \leftarrow calculate\_MCR(rule\_list)$
$x_{dec} \leftarrow sample(MCR$,method='minimum')
return x_{dec}

基于森林的自编码器

❑ 优势

- 训练速度较快：如在MNIST和 CIFAR10 上的训练速度比基于神经网络的模型快数倍以上
- 重构误差低：基于规则，而不是计算
- 容损性：编码规则具有强相关性，因此在部分损坏的情况下也能很好地工作
- 可复用性：在一个数据集上训练好的模型能够直接应用于同领域另外一个数据集

❑ 不足

- 与神经网络相比编码表达力不足
- 编码、解码速度相对神经网络较慢

深度森林的适用条件

- 具备逐层处理的任务
- 参数不可微
- 一定的模型复杂度

7

中英文术语对照

中英文术语对照

- 生成式对抗网络：Generative Adversarial Networks
- 生成器：Generator
- 判别器：Discriminator
- 注意力： Attention
- 深度强化学习: Deep Reinforcement Learning
- 状态：State
- 激励：Reward
- 动作：Action
- 深度森林：Deep Forest
- 决策树：Decision Tree
- 随机森林：Random Forest

中英文术语对照

- 梯度提升决策树：Gradient Boosting Decision Trees
- 多粒度级联森林：Multi-Grained Cascade Forest
- 自编码器：Auto Encoder
- 多层梯度提升决策树：Multi-Layered Gradient Boosting Decision Trees

谢谢！