深度学习基础

常宝宝
北京大学计算语言学研究所
chbb@pku.edu.cn

概要

- 导引
- 前馈神经网络
- 卷积神经网络
- 循环神经网络

神经网络模型概要

- 人工神经网络早期研究可以追溯到上世纪40年代。
- 1950年代，Frank Rosenblatt提出感知机算法。
- 1960年代末，Marvin Minsky指出感知机关键缺陷：
  - 无法对XOR问题建模(限于线性描述能力)
  - (当时)计算能力无法满足需求
- 1970年代，神经网络研究低潮
- 1980年代，神经网络研究复苏(BP算法被关注和应用)
- 1990年代，SVM等取代神经网络模型获得更多关注
- 2006年以后，神经网络方法重新崛起(Deep Learning)

特点

- 非线性学习
- 端到端建模
- 分布式特征表示
- 多层表示学习

非线性学习

- 神经网络模型

$$\boldsymbol{y} = \mathrm{NN}(\boldsymbol{x})$$

  其中，$\boldsymbol{x}$是输入向量(特征向量)，$\boldsymbol{y}$是输出向量(任务相关)

- 与最大熵模型、支持向量机的差异
  - 最大熵模型、支持向量机是(对数)线性机器学习
  - 神经网络模型是非线性机器学习技术

- 神经网络模型有更强的表达能力 (capacity)
  - 数据(特征)和输出之间不是(对数)线性关系

非线性学习

- 神经网络模型表达能力很强

- **universal approximation theorem** (Hornik et al., 1989)

  Neural networks with a single hidden layer (*with non-linear activation*) *can be used to approximate any continuous function to any desired precision..*

- 表达能力强 ≠ 成功
  - 存在未必一定找得到

    逼近的近似程度与hidden neuron的数量有关
  - 不能保证学到最优的参数(局部最优)

端到端建模

- 经典机器学习中，需要特征工程
  - 人工设计和提取特征，将任务输入转换为特征向量
  - 模型实现从特征向量到任务输出的映射
- 深度学习中，可以无需特征工程
  - 模型实现从任务输入到任务输出的映射
- 端到端建模
  - 避免了特征工程

分布式特征表示

- 经典机器学习　One-hot 向量表示
  - 输入数据→特征表示$(f_1, f_2, \dots, f_n)$

    $w_{-1} = the, w_{+1} = ship, p_{-1} = Det, \dots$
  - 特征向量的维数=特征的数量
  - 实质是对语言对象或范畴的One-hot 向量表示

    $$f_i = (0,0, \dots, 1, \dots, 0)$$
  - 特征表示的特点：高维、稀疏
  - 特征和特征之间相互独立

    $w_{+1} = ship$和$w_{+1} = boat$没有关系

分布式特征表示

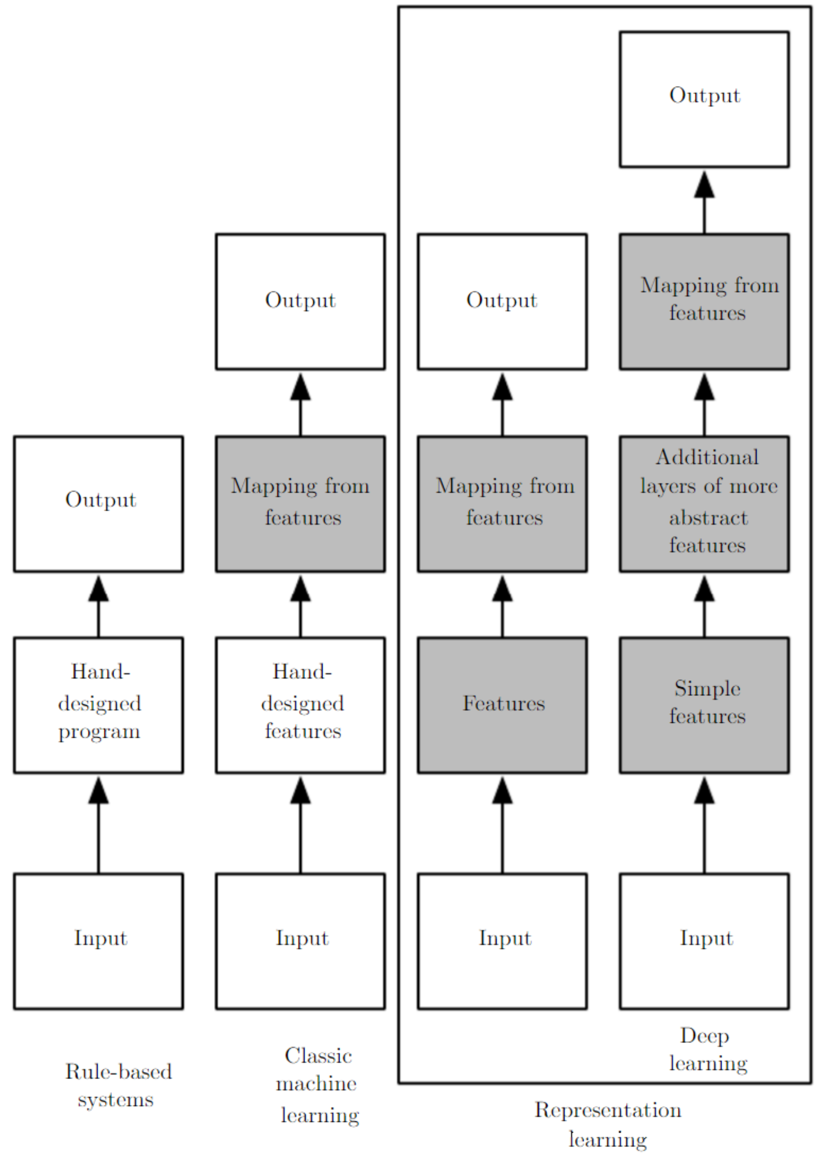
- 深度学习，分布式向量表示
  - 特征表示为低维、稠密向量
  - 特征被嵌入d维向量空间($d \ll$ 特征数量)
    $f_i = (0.16, \quad 0.03, \quad -0.17, \quad -0.13)$
  - 分量不只是0和1，是一个任意实数
  - 类似的特征拥有类似的向量表示
    - $w_{+1} = ship$和$w_{+1} = boat$拥有相近的特征表示
    - 稠密表示拥有更好的推广能力
  - 特征表示被称作 word embedding 或者 feature embedding
  - 特征表示被视作模型的一个组成部分，自动学习

多层表示学习

- 深度学习模型中，对输入数据的特征表示是分层的
  每层对应输入***x***的一种表示
- 高层特征表示经由底层特征表示自动习得
- 底层特征对应局部、具体特征
  高层特征对应全局、抽象特征
- 表示学习 自动学习数据多层表示的学习机制
- 大量使用预训练技术(自指导学习)
- 深度学习 体现为特征表示的层次性

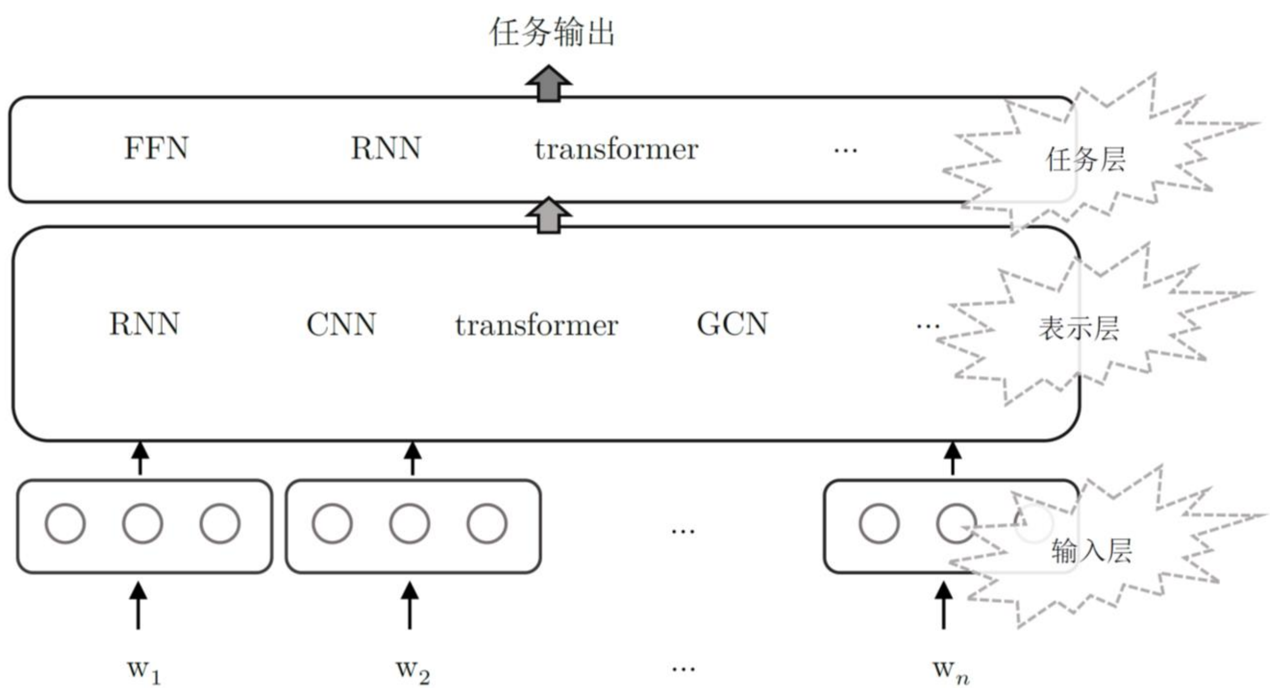
多层表示学习

- 基于规则的模型
- 经典学习模型
- 浅层神经网络模型
- 深度学习模型



基于深度学习的自然语言处理

- 输入层　将输入中的词例转换为向量表示
- 表示层　实现输入文本的多层表示(编码)
- 任务层　实现文本表示到任务输出的映射(解码)

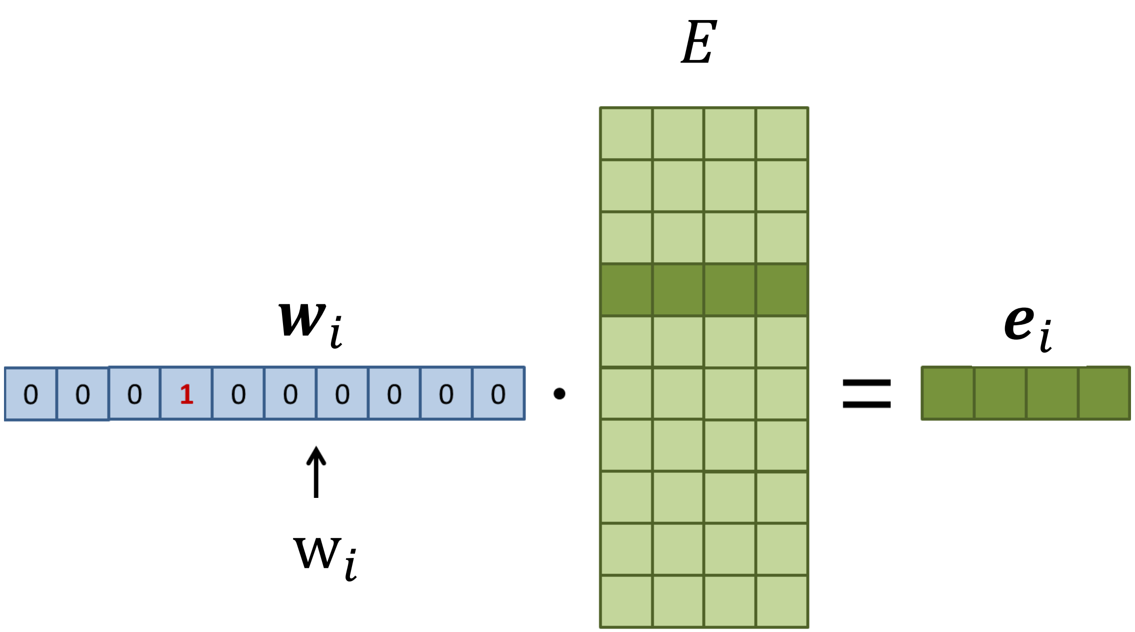


词的向量表示

- 词向量查找表$E \in \mathbb{R}^{d \times |V|}$
- 是给定w_i查找其对应的向量表示$\boldsymbol{e}_i \in \mathbb{R}^d$
- $\boldsymbol{w}_i$是w_i的one-hot向量表示

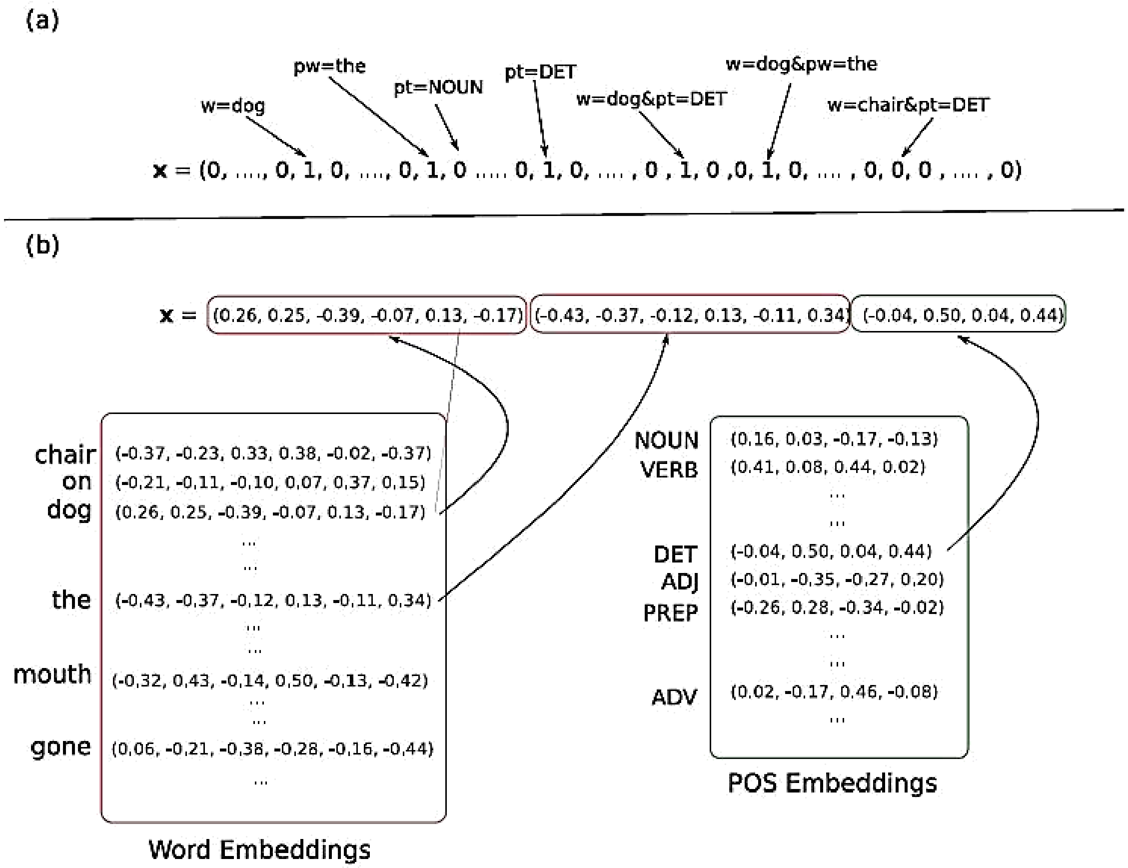
$$\boldsymbol{e}_i = E\boldsymbol{w}_i$$

- E是模型参数



词的向量表示

- 词类、位置等其他特征均可以同样的方式向量化
- 特征综合
  - 拼接
  - 相加

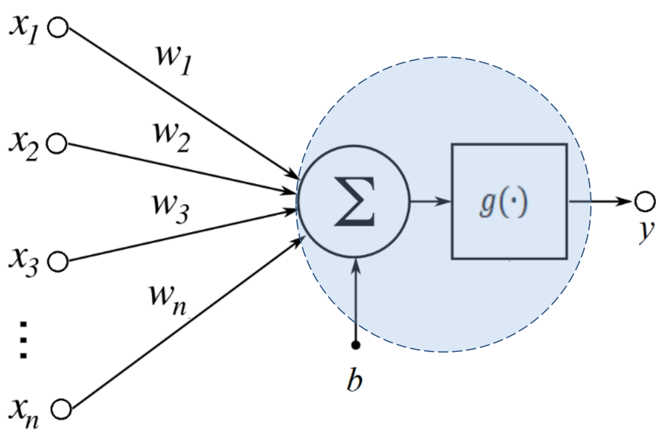


概要

- 深度语言处理模型概述
- **前馈神经网络**
- 卷积神经网络
- 循环神经网络

人工神经元

- 神经网络的基本计算单位：神经元(neuron)



$$
\begin{aligned}
y &= g\left(\boldsymbol{w}^{\top}\boldsymbol{x} + b\right) \\
&= g(w_1 x_1 + w_2 x_2 + \cdots + w_n x_n + b) \\
&= g\left(\sum_{i=1}^{n} w_i x_i + b\right)
\end{aligned}
$$

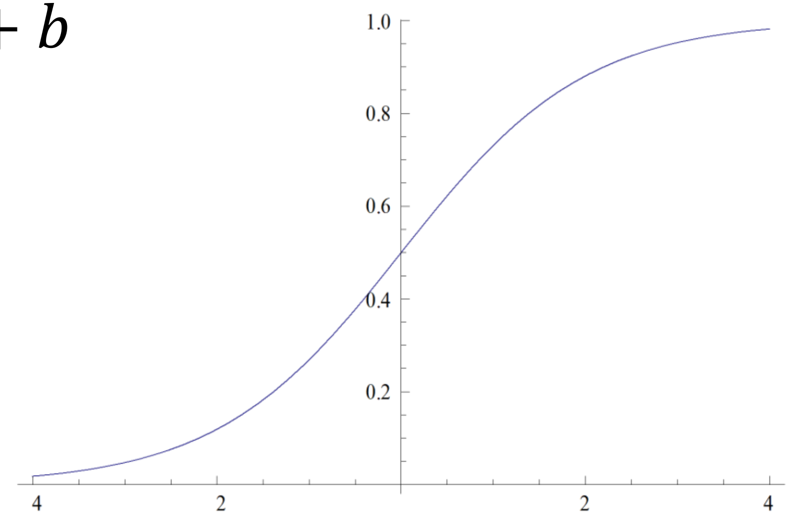
其中： $\boldsymbol{x} = (x_1, x_2, \dots, x_n)^{\top}$ (特征向量)

$\boldsymbol{w} = (w_1, w_2, \dots, w_n)^{\top}$ (特征权重)

$g(\cdot)$ 激活函数 (可选)

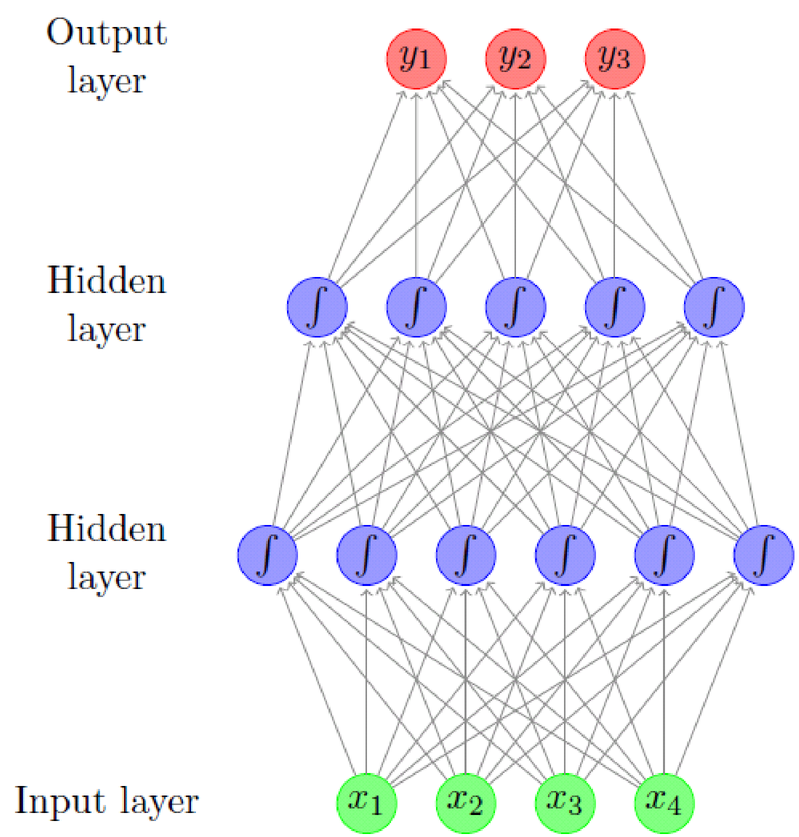
人工神经元

- 无激活函数，线性模型
  $$y = w_1 x_1 + w_2 x_2 + \cdots + w_n x_n + b$$
- sigmoid激活函数
  $$\sigma(x) = \frac{1}{1 + e^{-x}}$$
- 非线性模型
  $$y = \frac{1}{1 + e^{-(w_1 x_1 + w_2 x_2 + \cdots + w_n x_n + b\ )}}$$



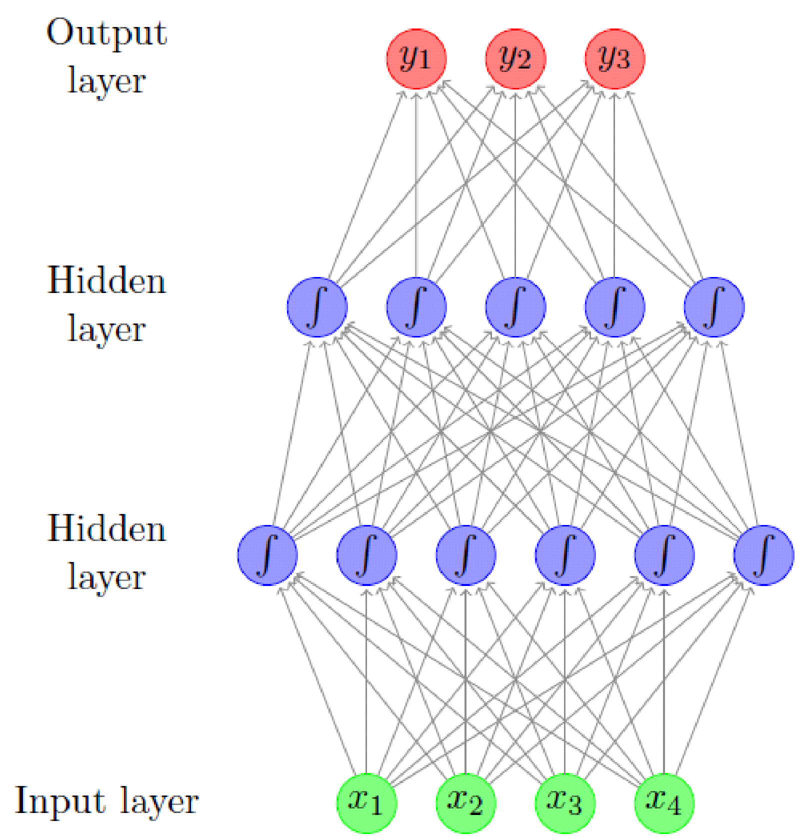
前馈神经网络(FFN)

- 前馈神经网络
  - 多个神经元逐层互联
- 全连接
  - 每个神经元都与下一层的所有神经元有连接
- 网络结构
  输入层
  隐藏层(n层, $n \geq 0$)
  输出层
- 又称多层感知机



前馈神经网络

- 数学描述(假设含2个隐层)
  $NN_{MLP2}(\boldsymbol{x}) = \boldsymbol{y}$
  $$\boldsymbol{h}^1 = g(\boldsymbol{W}^1\boldsymbol{x} + \boldsymbol{b}^1)$$
  $$\boldsymbol{h}^2 = g(\boldsymbol{W}^2\boldsymbol{h}^1 + \boldsymbol{b}^2)$$
  $$\boldsymbol{y} = \boldsymbol{W}^3\boldsymbol{h}^2$$
- 网络参数
  $\boldsymbol{\Theta} = (\boldsymbol{W}^1, \boldsymbol{b}^1, \boldsymbol{W}^2, \boldsymbol{b}^2, \boldsymbol{W}^3)$
- 学习输入对象的多层表示
  $\boldsymbol{x} \rightarrow \boldsymbol{h}^1 \rightarrow \boldsymbol{h}^2 \rightarrow \boldsymbol{y}$



前馈神经网络

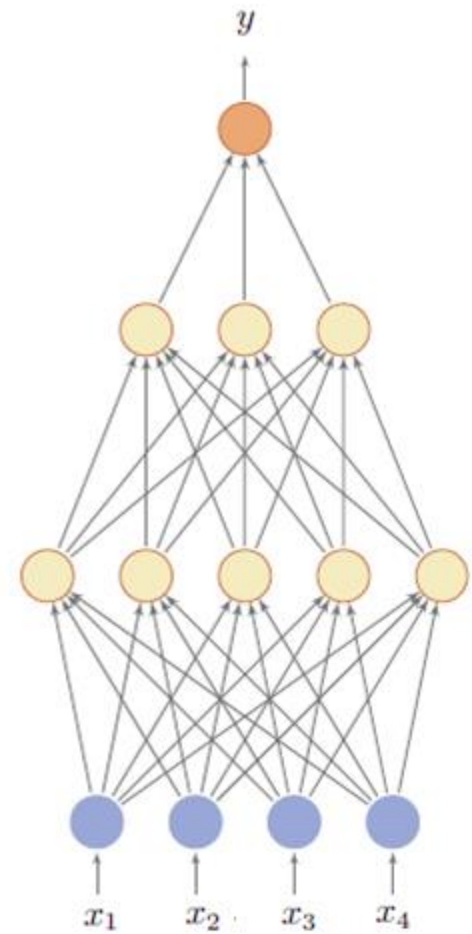
- $\boldsymbol{y}$可以是标量，二分类模型或回归模型
- 非概率化模型

  若$y \geq 0$　　$\boldsymbol{x}$ 是正类(+)

  若$y < 0$　　$\boldsymbol{x}$ 是负类(-)
- 概率化模型(logistic回归)

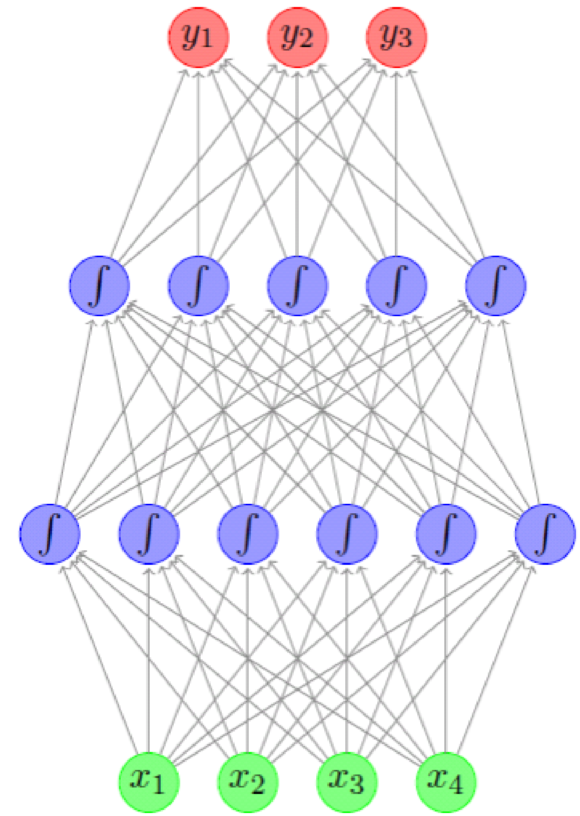
  $$p(+|\boldsymbol{x}) = \mathrm{sigmoid}(y) = \frac{1}{1+e^{-y}}$$

  $$p(-|\boldsymbol{x}) = 1 - p(+|\boldsymbol{x})$$
- 定义损失函数$Loss(\hat{y}, y)$



前馈神经网络

- $\mathbf{y}$是k维向量，对应k分类模型
- 非概率化模型
  $\hat{c} = \underset{i}{\operatorname{argmax}}\, y_i$
- 概率化模型(softmax)
  $$p(c = i|\boldsymbol{x}) = \operatorname{softmax}(y_i) = \frac{e^{y_i}}{\sum_{j=1}^{k} e^{y_j}}$$
  $\hat{c} = \underset{c}{\operatorname{argmax}}\, p(c|\boldsymbol{x})$
- 定义损失函数$Loss(\hat{y}, y)$



激活函数

- 激活函数为模型引入非线性(描述能力)
- 激活函数有多种选择
  - sigmoid函数

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

  - tanh函数

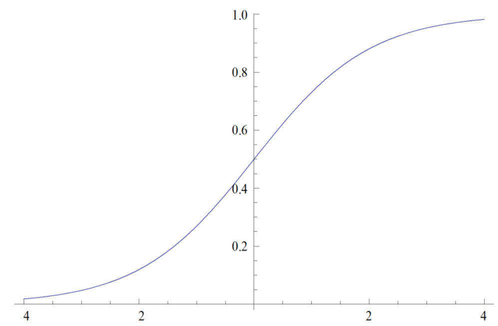
    $$\tanh(x) = \frac{e^{2x} - 1}{e^{2x} + 1}$$

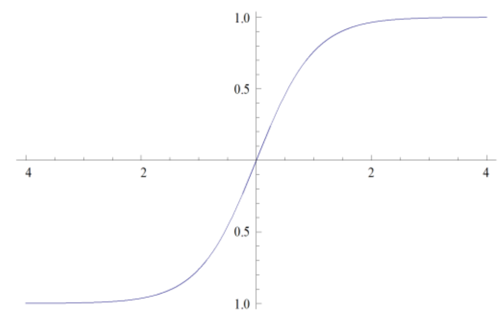
  - ReLu函数

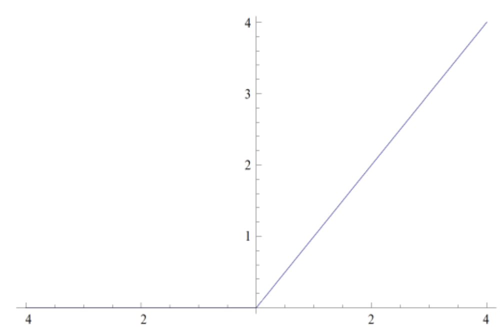
    $$\mathrm{ReLU}(x) = \max(0, x) = \begin{cases} 0, & x < 0 \\ x, & x \geq 0 \end{cases}$$

- 激活函数选定

ReLU≻tanh≻sigmoid





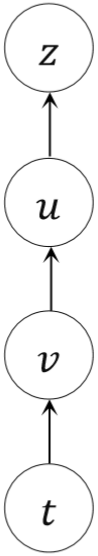


BP算法

- 神经网络训练—梯度下降
- 神经网络是复合函数，如何计算梯度？
- 复合函数求导的链式法则

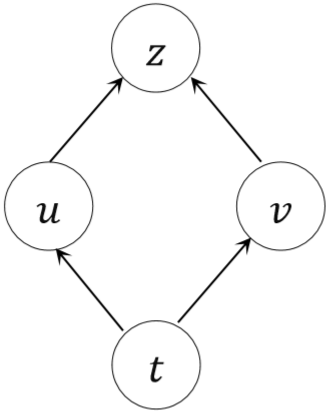
令$z = f(u), u = \varphi(v), v = \phi(t)$

$$\frac{dz}{dt} = \frac{dz}{du} \cdot \frac{du}{dv} \cdot \frac{dv}{dt}$$



令$z = f(u, v), u = \varphi(t), v = \phi(t)$

$$\frac{dz}{dt} = \frac{\partial z}{\partial u} \cdot \frac{du}{dt} + \frac{\partial z}{\partial v} \cdot \frac{dv}{dt}$$



BP算法

- BP算法：计算梯度$\nabla_{\boldsymbol{w}} Loss$和$\nabla_{\boldsymbol{b}} Loss$
- 令$\boldsymbol{z}^l \equiv \boldsymbol{W}^l \boldsymbol{h}^{l-1} + \boldsymbol{b}^l$，则
  $\boldsymbol{h}^l = g\left(\boldsymbol{W}^l \boldsymbol{h}^{l-1} + \boldsymbol{b}^l\right) \;\Rightarrow\; \boldsymbol{h}^l = g\left(\boldsymbol{z}^l\right)$
  $\boldsymbol{y} = g\left(\boldsymbol{W}^L \boldsymbol{h}^{L-1} + \boldsymbol{b}^L\right) \;\Rightarrow\; \boldsymbol{y} = g(\boldsymbol{z}^L)$
- 定义：$\delta_j^l \equiv \frac{\partial Loss}{\partial z_j^l}$ (损失函数关于z_j^l的微分)
- 则对输出层(L层)而言
  $$\boldsymbol{BP}1:\ \delta_i^L = \frac{\partial Loss}{\partial y_i} \cdot g'(z_i^L)$$
  损失$Loss$是网络输出$\boldsymbol{y}$的函数

图示

BP算法

- 对于l层而言

$$\boldsymbol{BP}2:\ \ \delta_i^l = \sum_j \left(\delta_j^{l+1} w_{ji}^{l+1}\right) \cdot g'\left(z_i^l\right)$$

- 损失函数针对$\boldsymbol{b}^l$的微分

$$\boldsymbol{BP}3:\ \ \frac{\partial Loss}{\partial b_i^l} = \delta_i^l$$

- 损失函数针对$\boldsymbol{W}^l$的微分

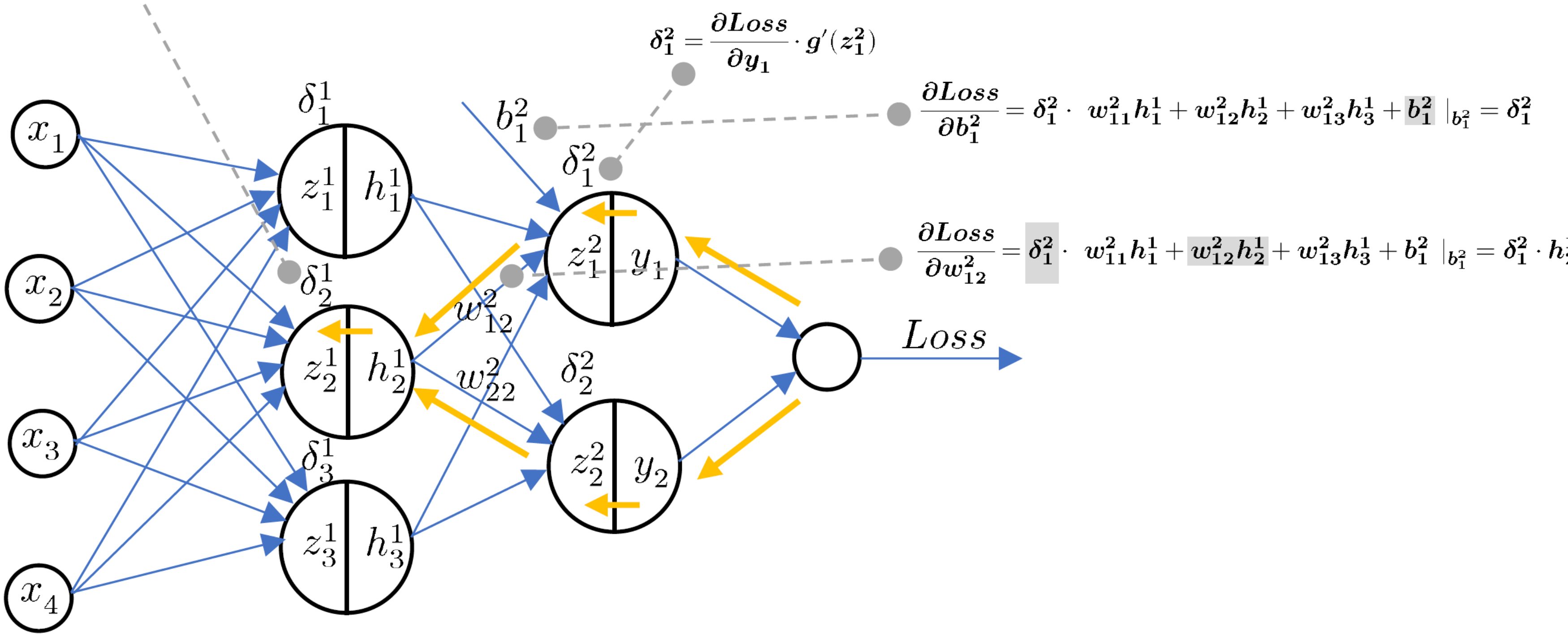
$$\boldsymbol{BP}4:\ \ \frac{\partial Loss}{\partial w_{ij}^l} = \delta_i^l \cdot h_j^{l-1}$$

图示

$$\delta_2^1 = \frac{\partial Loss}{\partial z_2^1}$$

$$= [\boldsymbol{\delta_1^2} \cdot \ \boldsymbol{w_{11}^2 h_1^1 + w_{12}^2 h_2^1 + w_{13}^2 h_3^1 + b_1^2} \ |_{h_2^1} + \boldsymbol{\delta_2^2} \cdot \ \boldsymbol{w_{21}^2 h_1^1 + w_{22}^2 h_2^1 + w_{23}^2 h_3^1 + b_2^2} \ |_{h_2^1}] \cdot \boldsymbol{g' \ z_2^1}$$

$$= [\boldsymbol{\delta_1^2 \cdot w_{12}^2 + \delta_2^2 \cdot w_{22}^2}] \cdot \boldsymbol{g'(z_2^1)}$$



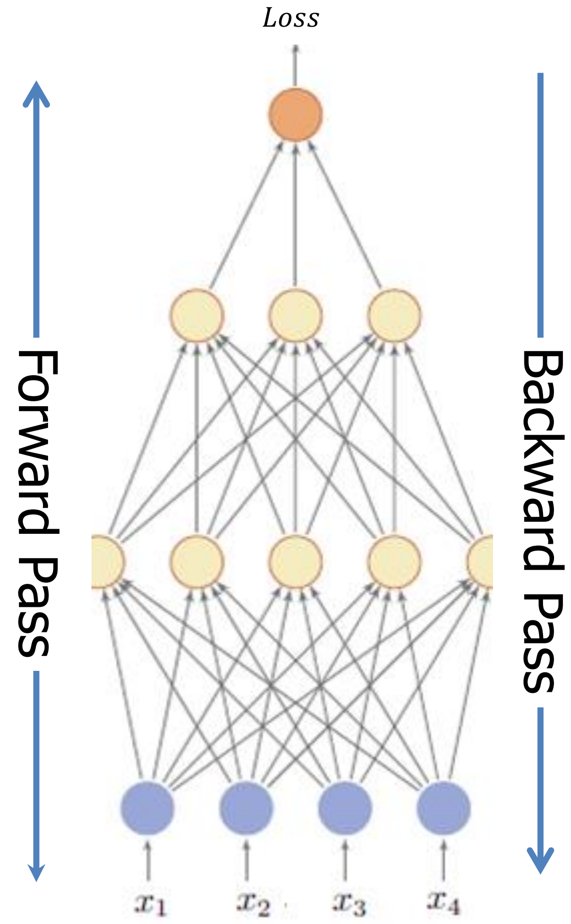
BP算法

- 梯度计算
  - Forward Pass

    从输入层开始，向前逐层计算隐层结点并最终算出Loss
  - Backward Pass

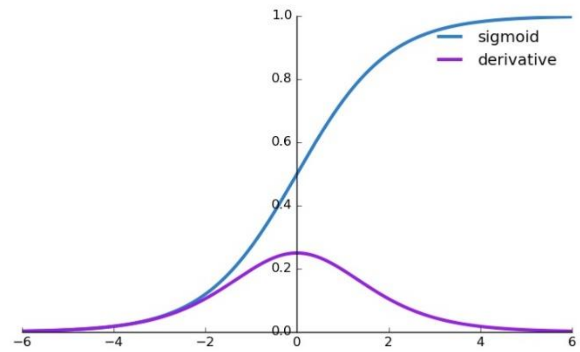
    从Loss出发，向后逐层计算梯度(***BP***1-***BP***4)

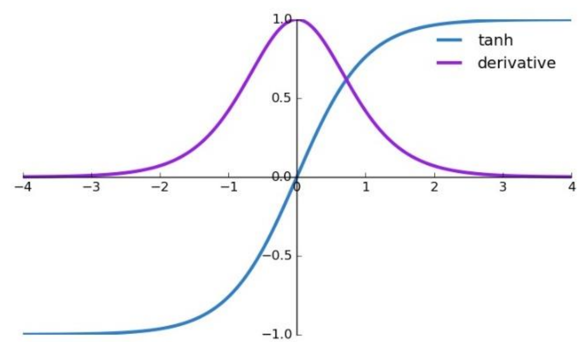
- 基于梯度更新神经网络参数
- BP算法解决了神经网络梯度计算问题

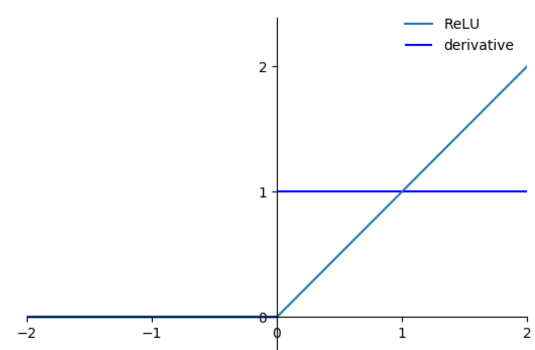


激活函数微分

- sigmoid函数微分
  $$g'_{\text{sigmoid}}(x) = \sigma(x)\big(1 - \sigma(x)\big)$$
- tanh函数微分
  $$g'_{\tanh}(x) = 1 - \tanh^2(x)$$
- ReLu函数微分
  $$g'_{\text{ReLU}}(x) = \begin{cases} 0, & x < 0 \\ 1, & x \geq 0 \end{cases}$$
- 激活函数和梯度消失

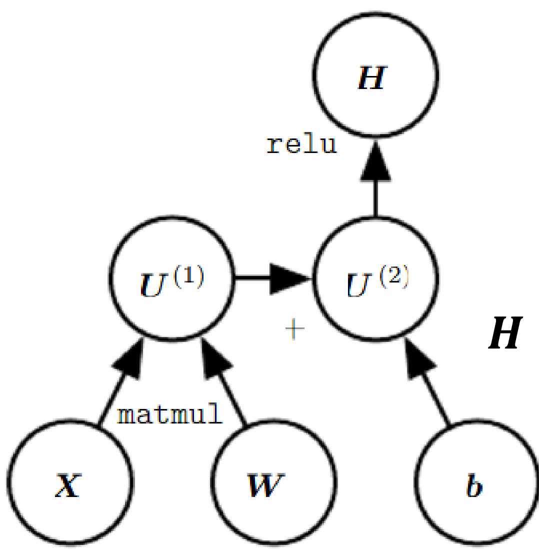




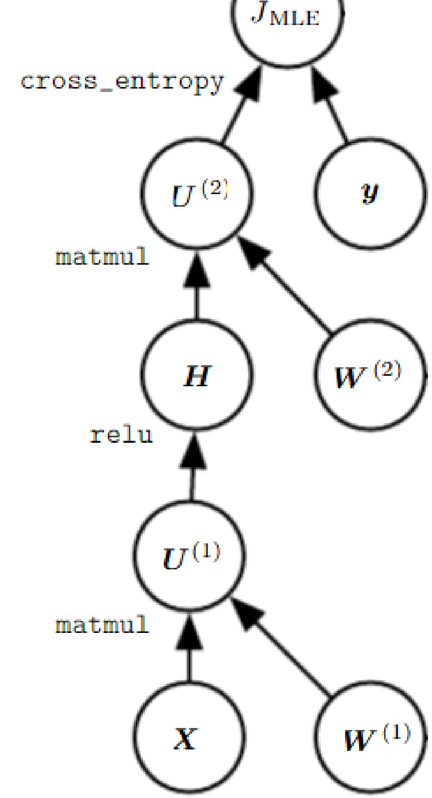


自动梯度计算

- 根据网络结构，手工推导梯度计算公式并编码
- 基于Computation Graph的自动梯度计算
- Computation Graph是计算过程的一种图形表示。
  - 叶节点代表输入
  - 分支结点代表运算及运算结果



$$\boldsymbol{H} = \max\{0, \boldsymbol{X}\boldsymbol{W} + \boldsymbol{b}\}$$



基于计算图的BP算法

- Forward Pass

  for $i = n_i + 1, \ldots, n$ **do**

  $\mathbb{A}^{(i)} \leftarrow \{u^{(j)} | u^{(j)} \in Parent(u^{(i)})\}$

  $u^{(i)} \leftarrow f^{(i)}(\mathbb{A}^{(i)})$

  return $u^{(n)}$

- Backward Pass

  $D[u^{(n)}] \leftarrow 1$

  for $j = n - 1$ down to 1 **do**

  $D[u^{(j)}] \leftarrow \sum_{i \in Child(u^{(j)})} D[u^{(i)}] \cdot \frac{\partial u^{(i)}}{\partial u^{(j)}}$

  return $\{ D[u^{(i)} | i = 1,2, \ldots, n_i\}$

$u^{(n)} = NN(u^{(1)}, u^{(2)}, \ldots, u^{(n_i)})$

$u^{(n)}$ -- loss node

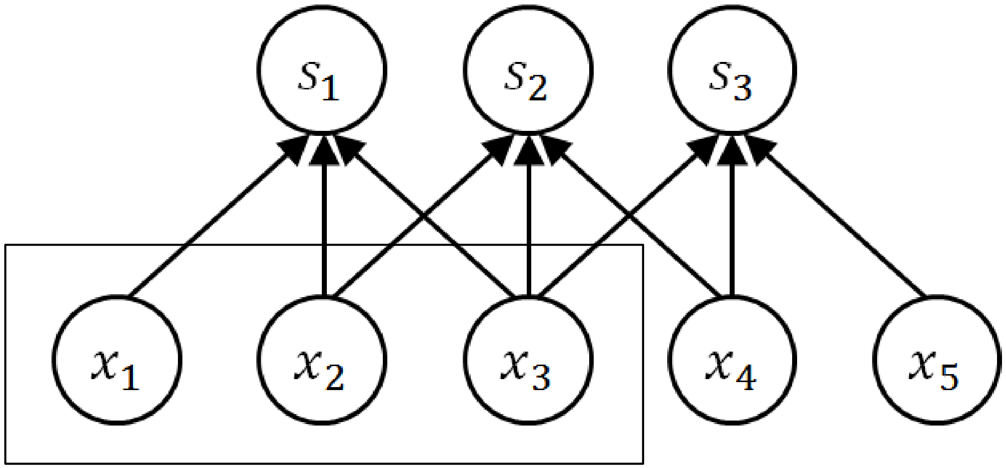
概要

- 深度语言处理模型概述
- 前馈神经网络
- **卷积神经网络**
- 循环神经网络

卷积神经网络

- 稀疏连接
  - 输出单元只和有限个输入单元连接
  - 卷积窗口k
- 参数共享
  - $\boldsymbol{w} = (w_1, w_2, \dots, w_k)^\top$
  - $\boldsymbol{w}$ 称作filter (kernel)
- feature map

  $\boldsymbol{s} = (s_1, \cdots, s_m)$



$$s_i = g(\boldsymbol{w}^\top \cdot \boldsymbol{x}_{i:i+k-1} + b)$$

卷积运算

卷积神经网络

- 窄卷积和宽卷积
  - 在输入层两侧填充$k-1$个0, 形成宽卷积
  - 输入层两侧不做填充，形成窄卷积
- 卷积层可以堆叠
- 参数共享 不同窗口的同质特征
- 多filter卷积
  - W的行对应不同的filter

$$\boldsymbol{s}_i = g(W\boldsymbol{x}_i + \boldsymbol{b})$$

- 多种filter size (kernel size)

卷积神经网络

- 给定句子$T = w_1 w_2 \dots w_n$
- 对应的词向量序列

$$\boldsymbol{e}_1, \boldsymbol{e}_2, \cdots, \boldsymbol{e}_n$$

- 设窗口宽度为k,，令

$$\boldsymbol{w}_i = [\boldsymbol{e}_i; \boldsymbol{e}_{i+1}; \dots; \boldsymbol{e}_{i+k-1}]$$

- 进行卷积操作

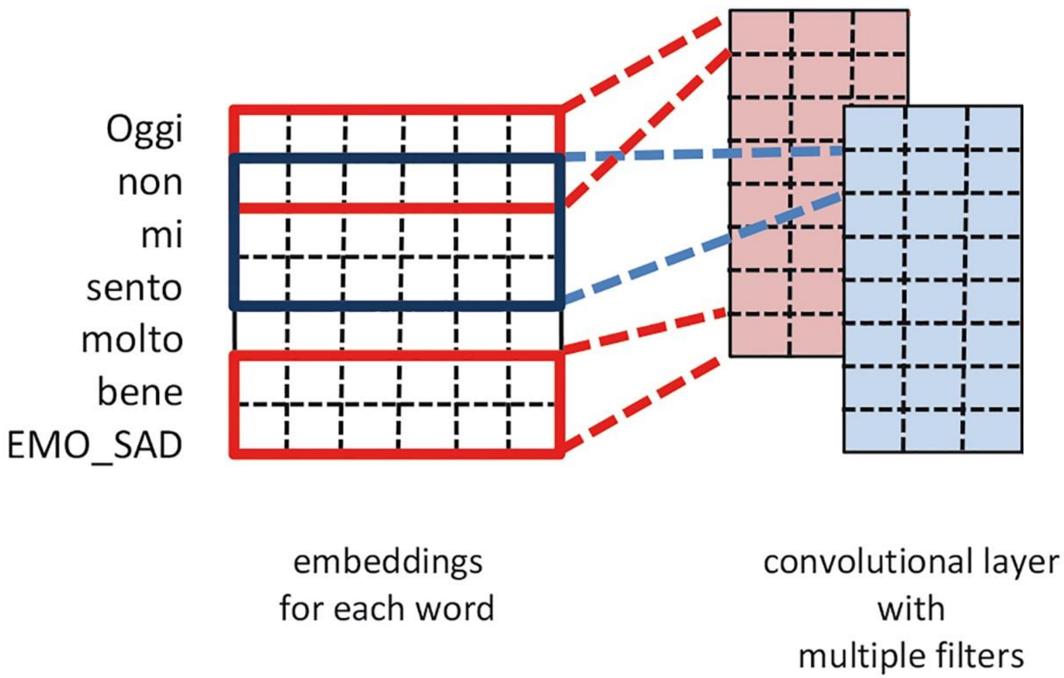
$$\boldsymbol{p}_i = g(W\boldsymbol{w}_i + \boldsymbol{b})$$

其中，$W \in \ \mathbb{R}^{d_c \times k \cdot d_e}$为filter(共享参数)

- 经过卷积操作，句子被转换为m个向量$\boldsymbol{p}_1, \boldsymbol{p}_2, \dots, \boldsymbol{p}_m$,
其中$m = n - k + 1$(narrow)或者 $m = n + k - 1$(wide)

卷积神经网络

- 两种kernel size
  - k=2
  - k=3
- 2*3种filter



卷积神经网络

- 语言中，序列(句子)长度是变化的，卷积结果依然
- 池化操作可将变长序列转换为固定维度的向量

- 池化操作：在卷积结果上进行缩减采样
  - 对卷积结果进行汇聚摘要
  - 缩减卷积结果的维度

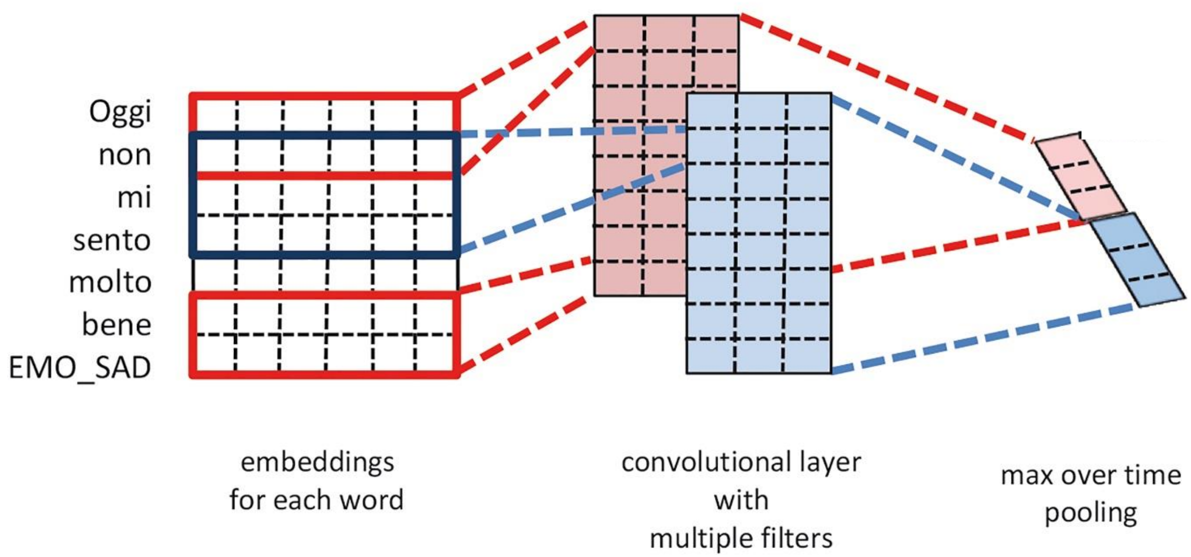
- NLP中常用 最大池化(Max over time pooling)

卷积神经网络

- $\boldsymbol{p}_i$是窗口$\boldsymbol{w}_i$编码表示
- Max over time pooling (对m个向量的每一维求最大值)

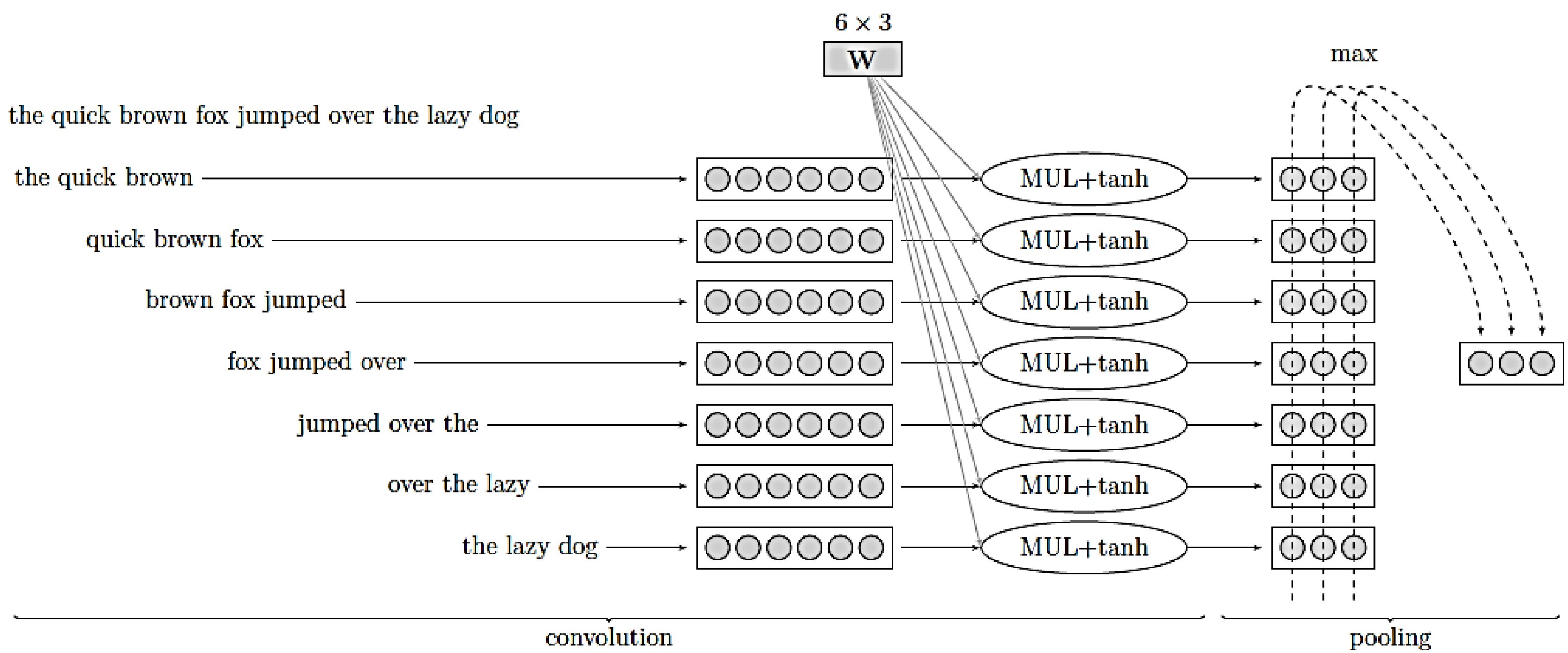
$$\boldsymbol{p}_1, \boldsymbol{p}_2, \dots, \boldsymbol{p}_m \rightarrow \boldsymbol{c}$$
$$c_j = \max_{1 \le i \le m} \boldsymbol{p}_i[j]$$

- 选择最显著特征作为最终特征



卷积神经网络

- 卷积与最大池化示例



卷积神经网络

- NLP中，卷积层和池化层经常组合使用，用作表示层实现技术，再结合适当任务层设计完成相应的任务建模
- 句子情感极性分析
  - 判定给定句子的情感极性
  - 三分类： positive, negative, neutral
  - 基于卷积神经网络实现句子的表示
  - 利用前馈神经网络实现情感极性分类
- 卷积操作有利于发现与位置无关的局部特征
  - It was *not good*, it was quite bad.

卷积神经网络

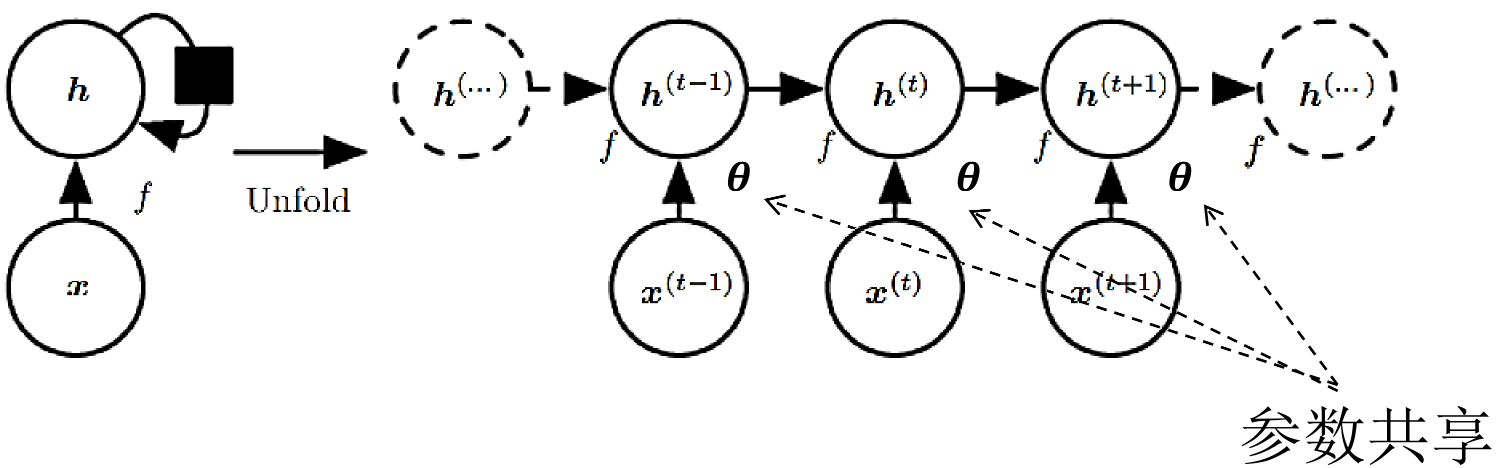
- 池化操作将任意长度的序列转换为固定长度的向量表示。
- 卷积操作可以堆叠(深层卷积)
- 其他池化操作
  - average pooling, k-max pooling,…
- 池化操作损失了序列结构信息

概要

- 深度语言处理模型概述
- 前馈神经网络
- 卷积神经网络
- **循环神经网络**

循环神经网络

- 循环神经网络是面向序列结构的建模工具
$$\boldsymbol{x}^{(1)}\boldsymbol{x}^{(2)},\dots,\boldsymbol{x}^{(t)},\dots,\boldsymbol{x}^{(n)} \Rightarrow \boldsymbol{h}^{(1)}\boldsymbol{h}^{(2)},\dots,\boldsymbol{h}^{(t)},\dots,\boldsymbol{h}^{(n)}$$
- 循环神经网络的数学表示
$$\boldsymbol{h}^{(t)} = f(\boldsymbol{h}^{(t-1)},\boldsymbol{x}^{(t)},\boldsymbol{\theta})$$



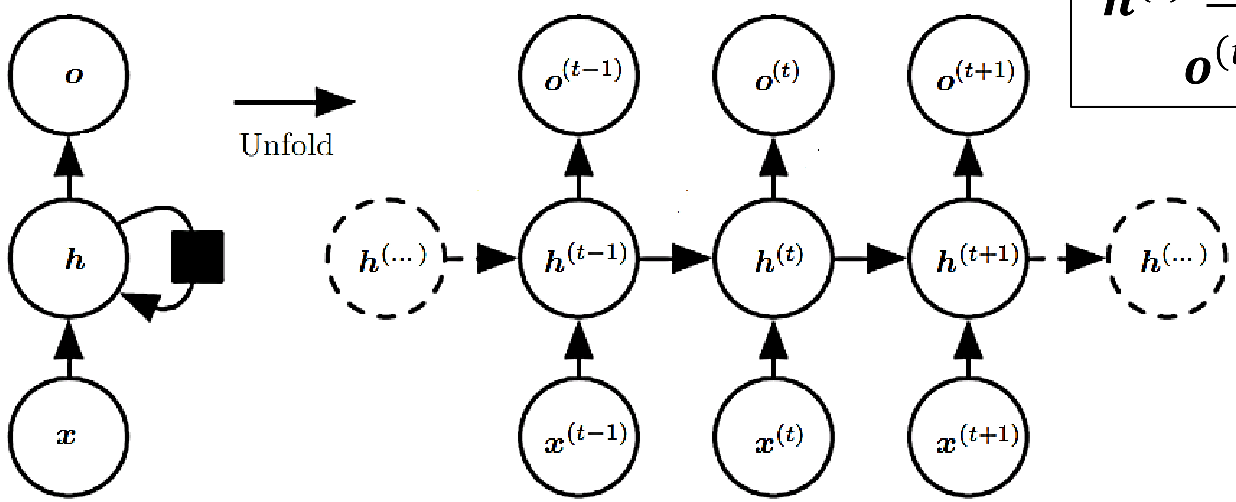
循环神经网络

- $\boldsymbol{h}^{(t)}$是对$\boldsymbol{x}^{(1)}\boldsymbol{x}^{(2)},\ldots,\boldsymbol{x}^{(t)}$中信息的压缩表示

$$\begin{aligned}\boldsymbol{h}^{(t)} &= f\left(\boldsymbol{h}^{(t-1)},\boldsymbol{x}^{(t)},\boldsymbol{\theta}\right)\\ &= f\left(f\left(\boldsymbol{h}^{(t-2)},\boldsymbol{x}^{(t-1)},\boldsymbol{\theta}\right),\boldsymbol{x}^{(t)},\boldsymbol{\theta}\right)\\ &= f\left(f\left(\ldots f\left(h^{(0)},x^{(1)},\theta\right)\ldots\right),\boldsymbol{x}^{(t-1)},\boldsymbol{\theta}\right),\boldsymbol{x}^{(t)},\boldsymbol{\theta})\end{aligned}$$

- 基于$\boldsymbol{h}^{(t)}$, 每个时刻可以有一个输出$\boldsymbol{o}^{(t)}$

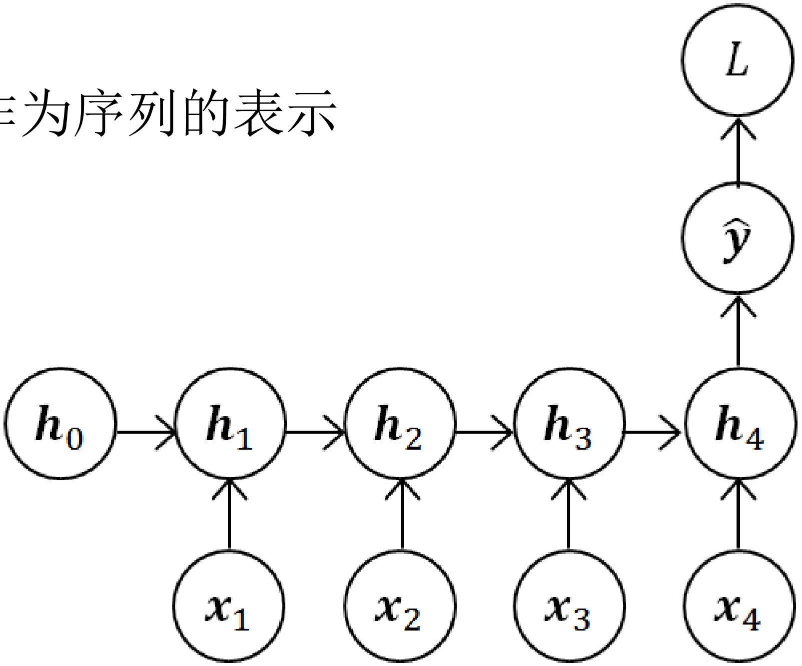
$$\boxed{\begin{gathered}\boldsymbol{h}^{(t)} = f_h(\boldsymbol{h}^{(t-1)},\boldsymbol{x}^{(t)},\boldsymbol{\theta}_h)\\ \boldsymbol{o}^{(t)} = f_o(\boldsymbol{h}^{(t)},\boldsymbol{\theta}_o)\end{gathered}}$$



循环神经网络

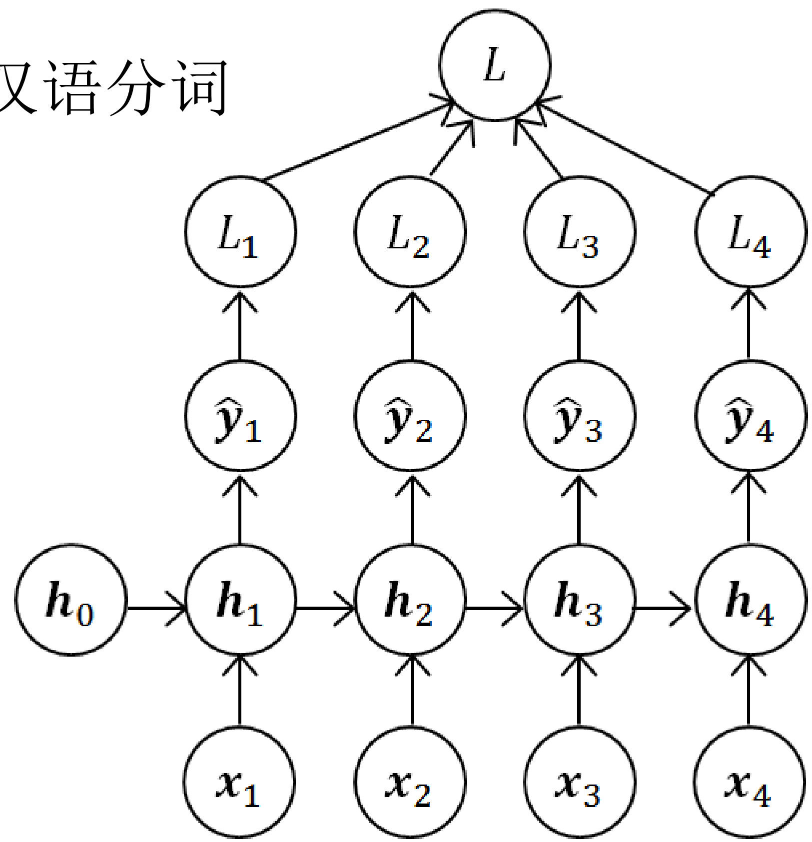
- 训练方法BPTT(backpropagation through time)
- 与其他类型神经网络训练本质相同

- 作为序列表示模型
  - 用最后一个时刻的隐层状态作为序列的表示
  - 例如：情感分析



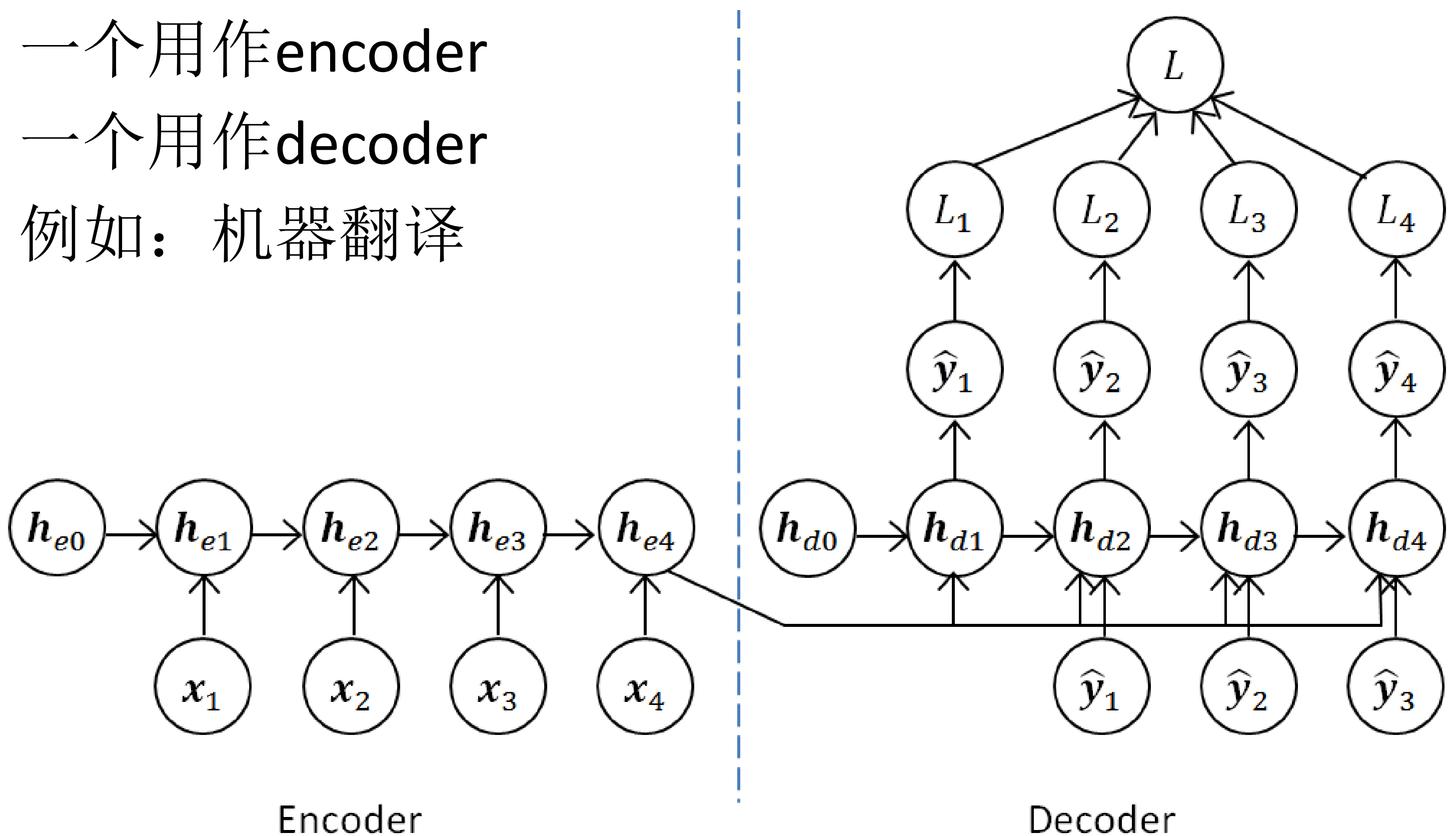
循环神经网络

- 作为序列标注模型
  - 每个时刻均有输出
  - 例如：词类标注、汉语分词



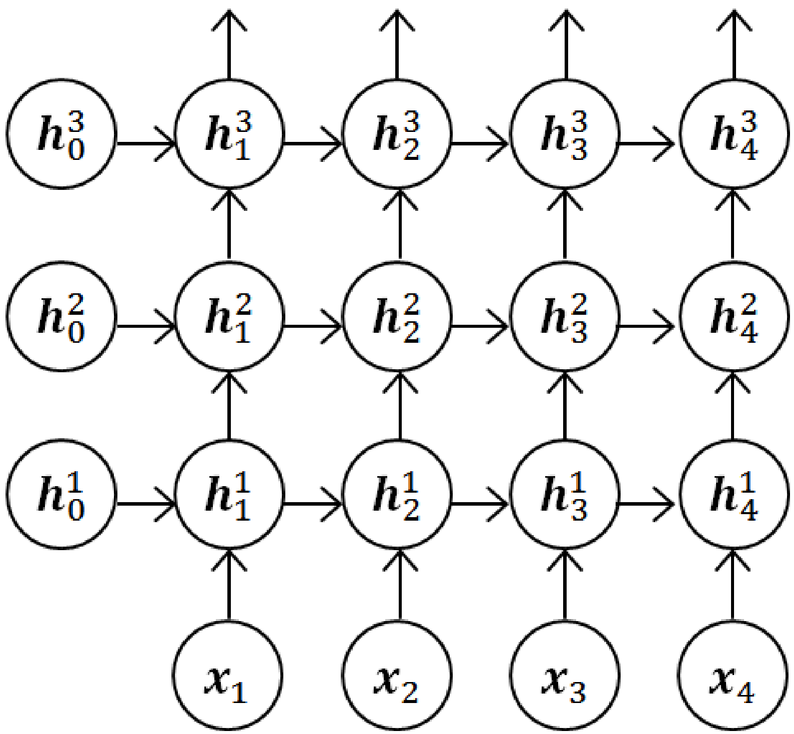
循环神经网络

- 作为翻译模型(encoder-decoder)
  - 两个RNN串接
  - 一个用作encoder
  - 一个用作decoder
  - 例如：机器翻译

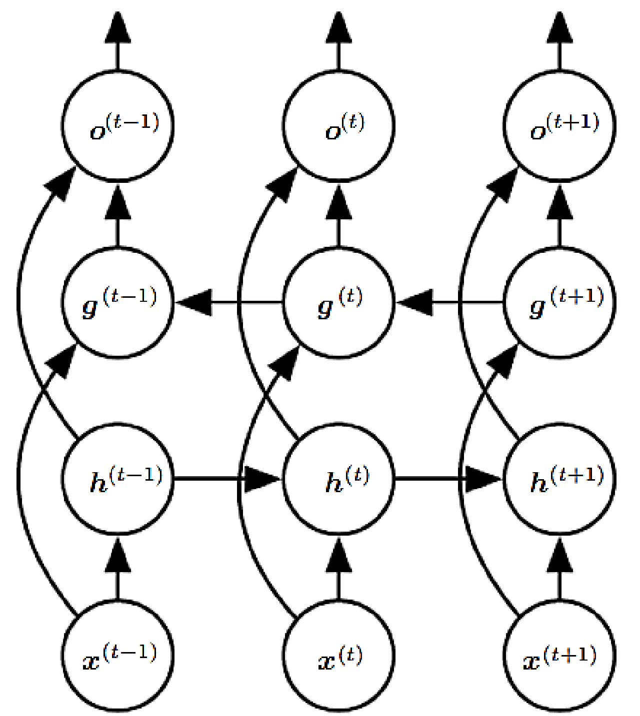


循环神经网络

- 多层循环神经网络



- 双向循环神经网络



循环神经网络

- 循环神经网络的实现 — 朴素实现

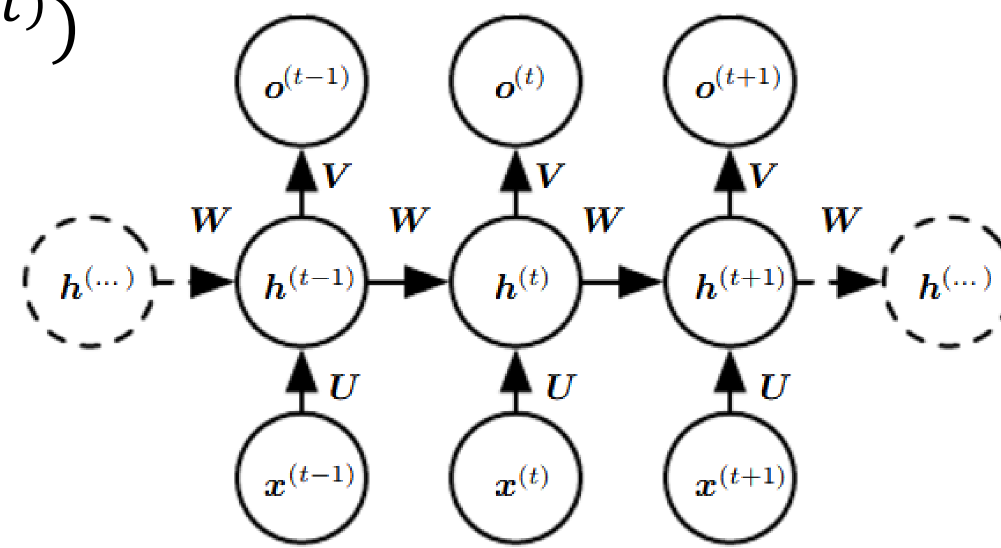
$$\boldsymbol{h}^{(t)} = f_h(\boldsymbol{h}^{(t-1)}, \boldsymbol{x}^{(t)}, \boldsymbol{\theta}_h)$$

$$\boldsymbol{o}^{(t)} = f_o(\boldsymbol{h}^{(t)}, \boldsymbol{\theta}_o)$$

$$\boldsymbol{h}^{(t)} = g_h\left(\boldsymbol{b} + \boldsymbol{W}\boldsymbol{h}^{(t-1)} + \boldsymbol{U}\boldsymbol{x}^{(t)}\right), t = 1,2,\dots,n$$

$$\boldsymbol{o}^{(t)} = g_o(\boldsymbol{c} + \boldsymbol{V}\boldsymbol{h}^{(t)})$$

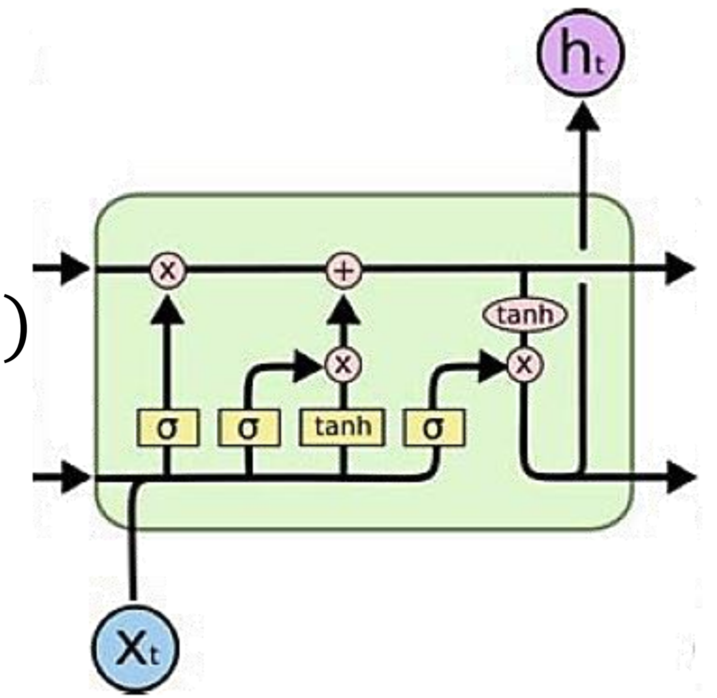
- 缺陷
  - 梯度消失/爆炸

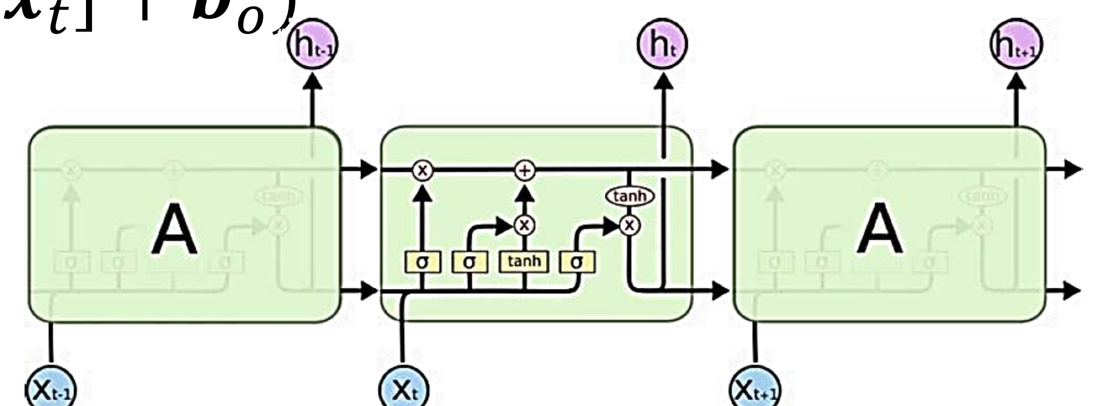


循环神经网络

- 循环神经网络的实现 — LSTM

$$\boldsymbol{h}_t = \boldsymbol{o}_t \odot \tanh(\boldsymbol{C}_t)$$
$$\widetilde{\boldsymbol{C}}_t = \tanh(\boldsymbol{W}_C \cdot [\boldsymbol{h}_{t-1}, \boldsymbol{x}_t] + \boldsymbol{b}_C)$$
$$\boldsymbol{C}_t = \boldsymbol{f}_t \odot \boldsymbol{C}_{t-1} + \boldsymbol{i}_t \odot \widetilde{\boldsymbol{C}}_t$$
$$\boldsymbol{f}_t = \sigma(\boldsymbol{W}_f \cdot [\boldsymbol{h}_{t-1}, \boldsymbol{x}_t] + \boldsymbol{b}_f)$$
$$\boldsymbol{i}_t = \sigma(\boldsymbol{W}_i \cdot [\boldsymbol{h}_{t-1}, \boldsymbol{x}_t] + \boldsymbol{b}_i)$$
$$\boldsymbol{o}_t = \sigma(\boldsymbol{W}_o \cdot [\boldsymbol{h}_{t-1}, \boldsymbol{x}_t] + \boldsymbol{b}_o)$$

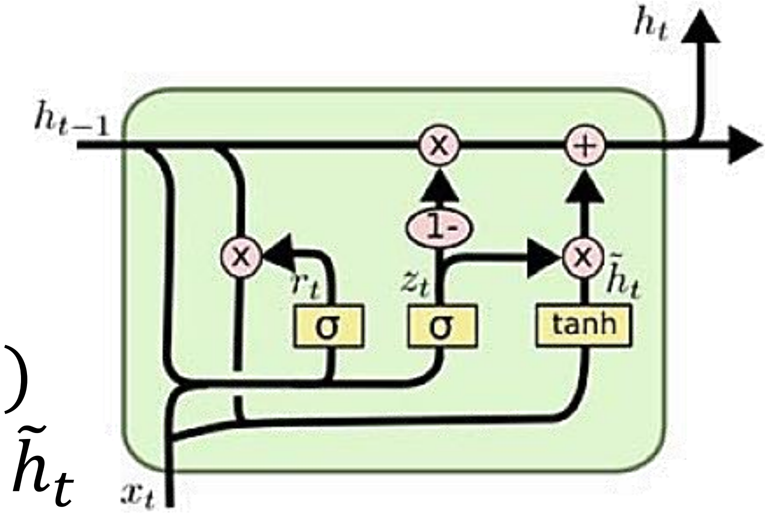




循环神经网络

- 循环神经网络的实现 — GRU

$$z_t = \sigma(W_z \cdot [h_{t-1}, x_t])$$
$$r_t = \sigma(W_r \cdot [h_{t-1}, x_t])$$
$$\tilde{h}_t = \tanh(W \cdot [r_t \odot h_{t-1}, x_t])$$
$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h}_t$$



- LSTM、GRU因为减缓了梯度消失问题，是比较常用的循环神经网络模型