人工智能

人工智能概述

主讲: 赵国亮

内蒙古大学电子信息工程学院

May 22, 2020

目录 I

图卷积网络由 University of Amsterdam 的 Thomas N. Kipf 提出，2020 年博士毕业，SEMI-SUPERVISED CLASSIFICATION WITH GRAPH CONVOLUTIONAL NETWORKS,

阿里如何把图卷积网络用于闲鱼的垃圾评论过滤. 目前该系统已经部署到了闲鱼使用环境中, 它每天能处理百万级的闲鱼评论, 并在其他强有力的深度学习模型基础上, 额外筛选出一千多条非常隐秘的垃圾评论.

基于图卷积的垃圾信息筛选是一种非常通用的思想, 它的应用范围远不止垃圾评论过滤, 淘宝信息的知识产权保护、淘宝商品管控和用户恶意评价等方面都可以采用.

GAS 的整体流程是如图 1, 模型会从左侧图抽取出表示商品、用户和评论的信息, 从右侧抽取出类似评论表示的意义. 最后结合这些信息进行分类, 模型就能很好地识别垃圾信息了.

图 1: GCN-based Anti-Spam System (GAS) 的垃圾评论过滤系统

图卷积的核心思想是希望利用近邻节点的信息进行聚合而生成当前节点的新表征, 这样的节点表示可以进一步用于下游任务. 如果我们直接从核心表达式出发, 跳过推导过程, 其实能更容易地理解. 如下所示为两层图卷积网络之间的传播方法, 它看起来只不过比常规的神经网络多了 $\tilde{\mathrm{D}}$ 与 $\tilde{\mathrm{A}}$ 这几项.

$$\mathrm{H}^{(\mathrm{l}+1)} = \sigma\left(\tilde{\mathrm{D}}^{-\frac{1}{2}}\tilde{\mathrm{A}}\tilde{\mathrm{D}}^{-\frac{1}{2}}\mathrm{H}^{(\mathrm{l})}\mathrm{W}^{(\mathrm{l})}\right). \tag{1}$$

如果我们的图有 n 个节点, 那么节点与节点之间的关系可以用 $n \times n$ 的邻接矩阵表示, 它再加上由节点特征向量组成的矩阵 H 就是图卷积的输入. 在上式中, $\tilde{A}$ 以及 $\tilde{D}$ 就是由邻接矩阵算出来的东西, 它对于同一张图是不变的, 因此可以预先计算好. 现在, 剩下的 $H \times W$ 就是输入 Embedding H 经过一层全连接层了, 以这样方式进行层级传播的卷积网络就是图卷积, 我们可以将传播理解为每个节点拿到邻居节点信息, 并聚合到自身嵌入向量上. 如图 2 所示, 图卷积网络的输入是表示节点及边的特征向量, 经过一系列隐藏层的变换, 可以计算出每个节点的深度表征. 这样的 Z 再来做预测或生成就会非常有效. 直观而言, 图卷积将图片的 RGB 像素值替换成节点特征, 并且通过边的关系引入了邻居的概念, 完成卷积运算.

图 2: 图卷积网络的输入输出结构

- 异构图上的图卷积

阿里 GAS 一共有两种输入图, 它们分别用来表示局部信息与全局信息. 首先我们看看异构图, 一般只要边的种类加上节点的种类大于 2, 我们就可以称之为异构图. 如 3 所示闲鱼图为一个标准的异构图, 目前图卷积网络大部分都关注更简单的同构图, 闲鱼图这种异构图很难处理. 从图 3 我们可以看到, 闲鱼 Graph 有商品 I 和用户 U 这两种节点, 它们的边为评论 E. 如上, e_2,e_4 和 e_5 都是垃圾评论, 它们都来自于同一用户. 利用图来判断垃圾评论, 能利用更多的额外信息, 准确率也会比纯文本好得多. 现在回到图卷积, 一般图卷积的层级可以分为聚合 (aggregation) 与结合 (combination) 两大操作. 其中 AGG 会聚合邻近节点的嵌入向量, 例如最大池化或基于注意力权重的加权和等. COMBINE 操作会结合自身的嵌入向量与前面聚合的嵌入向量, 很多 GCN 方法将 COMBINE 操作放到了 AGG 里面.

在阿里的 GAS 中, 研究者使用拼接的方式将信息聚合到边上. 比如说如果 GAS 需要将信息聚合到不同的边 (即评论) 上, 那么比较核心的表达式可以写为:

$$\mathrm{h}_{\mathrm{e}}^{\mathrm{l}}=\sigma\left(\mathrm{W}_{\mathrm{E}}^{\mathrm{l}} \cdot \mathrm{AGG}_{\mathrm{E}}^{\mathrm{l}}\left(\mathrm{h}_{\mathrm{e}}^{\mathrm{l}-1}, \mathrm{h}_{\mathrm{U}(\mathrm{e})}^{\mathrm{l}-1}, \mathrm{h}_{\mathrm{I}(\mathrm{e})}^{\mathrm{l}-1}\right)\right), \tag{2}$$

其中

$$\mathrm{AGG}_{\mathrm{E}}^{\mathrm{l}}\left(\mathrm{h}_{\mathrm{e}}^{\mathrm{l}-1}, \mathrm{h}_{\mathrm{U}(\mathrm{e})}^{\mathrm{l}-1}, \mathrm{h}_{\mathrm{I}(\mathrm{e})}^{\mathrm{l}-1}\right)=\mathrm{concat}\left(\mathrm{h}_{\mathrm{e}}^{\mathrm{l}-1}, \mathrm{h}_{\mathrm{U}(\mathrm{e})}^{\mathrm{l}-1}, \mathrm{h}_{\mathrm{I}(\mathrm{e})}^{\mathrm{l}-1}\right),$$

其中 h^l 表示第 l 层边的隐藏向量, 它需要聚合 l-1 层自身的特征向量以及与它相连的两个节点向量, 聚合的方法是拼接三个向量. W^l 表示该神经网络层所需要学习的权重, σ 表示激活函数. 看上去它其实和一般的卷积网络并没有什么差别, 只不过输入都是图的各种信息, 这样也能基于局部上下文判断该评论是否是垃圾评论.

图 3: 图卷积异构网络——闲鱼 Graph

- 同构图上的图卷积

对于闲鱼 Graph 这种大型图, 我们能处理邻近节点这些局部信息, 但与此同时还应该能处理全局信息, 这样才能有效地减轻用户的对抗行为. 为此, 模型应该站在所有评论的角度, 看看与当前相似的评论都是什么样, 它们是不是垃圾评论. 阿里的研究者基于闲鱼 Graph 构建了一种新的 Comment Graph, 它是一种同构图, 每一个节点为评论内容, 节点之间的边为两条评论之间的相似性. 因为相似的评论距离非常近, 因此模型可以考虑与当前评论相近的评论, 从而更好地判断当前评论是不是垃圾评论.

图 4 给出了一小部分 Comment Graph, 如果说局部模型无法根据「add v」判断出意思是加微信, 那么放在 Comment Graph 中就非常明确了, 它与类似的说法都应该被判断为垃圾评论.

图 4: 基于闲鱼 Graph 构建的同构 Comment Graph

简单而言, Comment Graph 的构建主要分为四个步骤: 移除所有重复的评论; 通过词嵌入模型为评论生成嵌入向量; 利用 KNN Graph 算法获得相似的评论对;

移除同一用户提出的评论对, 或者同一卖家提出的评论, 因为之前的闲鱼 Graph 已经考虑了这些信息.

构建了 Comment Graph, 再用图卷积就能抽取节点信息了, 因为每一个节点输出向量都聚合了周围节点的信息, 它就能代表全局上这一些相似评论的意义. 最后, 结合异构图卷积与同构图卷积的结果, 再来做个简单的分类就很合理了.

针对对象

序列数据. 例如文本是字母和词汇的序列; 语音是音节的序列; 视频是图像的序列; 气象观测数据, 股票交易数据等也都是序列数据. RNN 专门解决时间序列问题, 用来提取时间序列信息, 一般放在 CNN 特征提取层之后.

核心思想

样本间存在顺序关系, 每个样本和它之前的样本存在关联. 通过神经网络在时序上的展开, 我们能够找到样本之间的序列相关性.

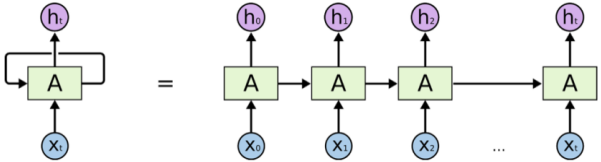

图 5: RNN 网络结构按时间展开

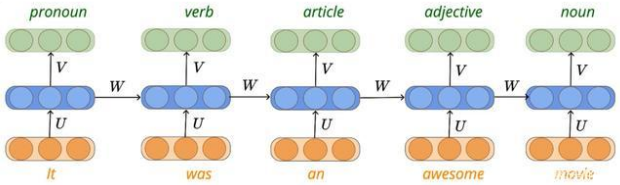
pronoun
verb
article
adjective
noun
V
W
U
It
was
an
awesome
movie

图 6: RNN 网络结构

RNN 主要用于序列分类

- 情感分类和视频分类序列标记.
- 语音标记和命名实体识别序列生成.
- 机器翻译和音译.

在循环神经网络中的每个时间步处, 旧信息都会随着当前输入而发生变化. 对于较长的句子, 我们可以想象, 在 t 时间步之后, 存储在 $t - k$ 时间步 ($k \ll t$) 的信息会经历一个逐渐转变的过程. 在反向传播过程中, 信息必须流经较长的时间步才能更新网络参数, 以最大程度地减少网络的损失.

考虑一个场景, 在该场景中, 我们需要计算时间步 L 处的网络损失. 假定损失是由于在时间步 S_1 对隐藏表示的错误计算而产生的. S 处的错误是由于向量 W 的参数不正确造成的. 此信息必须反向传播到 W, 以便向量可以校正其参数.

$$a_j = Ux_j + Ws_{j-1} + b, s_j = \sigma\left(a_j\right). \tag{3}$$

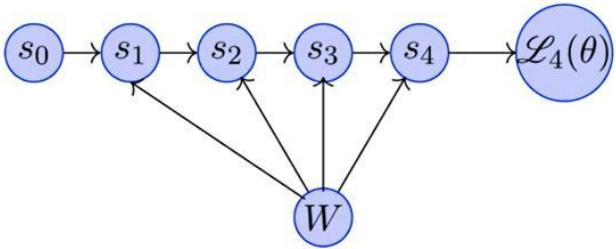

图 7: RNN 权重作用示意图

为了将信息传播回向量 W, 我们需要用到链式法则的概念. 简而言之, 链式法则归结为特定时间步的所有隐藏表示的偏导数的乘积.

$$\frac{\partial L_t(\theta)}{\partial W} = \frac{\partial L_t(\theta)}{\partial s_t} \sum_{k=1}^{t} \frac{\partial s_t}{\partial s_k} \frac{\partial s_k}{\partial W} \tag{4}$$

$$\frac{\partial s_t}{\partial s_k} = \frac{\partial s_t}{\partial s_{t-1}} \frac{\partial s_{t-1}}{\partial s_{t-2}} \frac{\partial s_{t-2}}{\partial s_{t-3}} \cdots \frac{\partial s_{k+1}}{\partial s_k} \Rightarrow \frac{\partial s_t}{\partial s_k} = \prod_{j=k+1}^{t} \frac{\partial s_j}{\partial s_{j-1}}. \tag{5}$$

如果我们有超过 100 个或者更长的序列的隐藏表示, 那么我们必须计算这些表示的乘积来进行反向传播. 假设其中一个偏导数值很大, 那么整个梯度值就会非常大, 从而导致梯度爆炸问题. 如果其中一个偏导数是一个很小的值, 那么整个梯度就会变得更小或消失, 使得网络难以训练, 这就是梯度消失问题. 由于 RNN 具有有限的状态大小, 而不是从所有的时间步长中提取信息并计算隐藏状态表示. 在从不同的时间步长中提取信息时, 我们需要遵循选择性读 (read)、写 (write) 和遗忘 (forget) 策略.

例 3.1

RNN 示例
让我们以使用 RNN 进行情感分析为例, 看看选择性 read, write 和 forget 策略的工作原理.

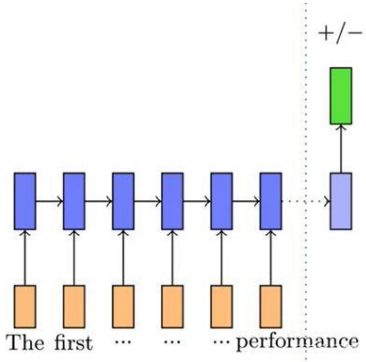

图 8: RNN 权重作用示意图

评论始于负面情绪, 后来变成正面.

过程如下:

过程说明: 我们首先遗忘由 stop words (a, the, is 等) 添加的信息. 有选择地阅读由情感表达的单词所添加的信息 (amazing, awesome 等). 有选择地将隐藏状态表示信息从当前单词写入新的隐藏状态. 使用选择性的读、写和遗忘策略, 我们可以控制信息流, 这样网络就不会出现短期记忆的问题, 也可以确保有限大小的状态向量得到有效的使用.

长期短期记忆—LSTM: 引入 LSTM 是为了克服 vanilla RNN 存在的短期记忆和梯度消失等问题. 在 LSTM 中, 通过使用门来调节信息流, 我们可以有选择地读写和遗忘信息.

• RNN 或 LSTM 中的写.

在 vanilla RNN 版本中, 隐藏表示 (s_t) 为: 是根据前一个时间步长的输出计算得出的隐藏表示 (s) 和当前输入 (x) 以及偏差 (b).

$$s_t = \sigma\left(Ux_t + Ws_{t-1} + b\right), \tag{6}$$

其中 s_{t-1} 为前一个时间步的隐藏表示, x_t 当前输入, b 为偏差.

在这里, 我们获取 s 的所有值并计算当前时间 (s_t) 的隐藏状态表示.

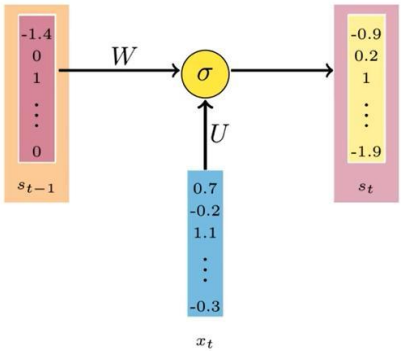

图 9: RNN 节点计算

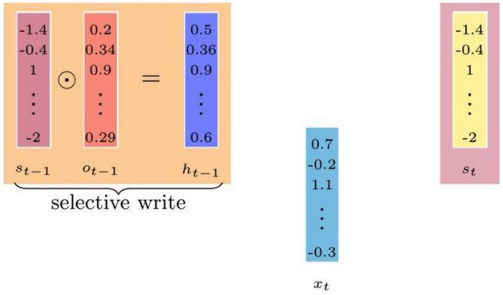

图 10: 平面 RNN 节点计算

- RNN 或 LSTM 中的选择性写方式.

在 “选择性写” 中, 不是将所有信息写入 s_{t-1} 中以计算隐藏表示 (s_t). 我们只将有关 s_{t-1} 的一些信息传递给下一个状态来计算 s_t. 一种方法是在 0-1 之间分配一个值, 该值确定将当前状态信息的哪一部分传递给下一个隐藏状态.

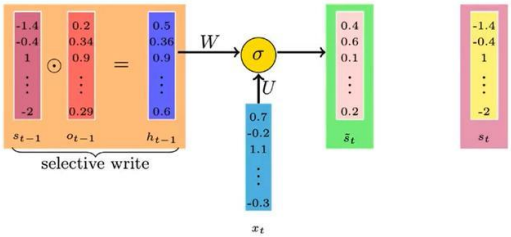
-1.4
-0.4
1
-2
0.2
0.34
0.9
0.29
0.5
0.36
0.9
0.6
W
σ
U
0.4
0.6
0.1
0.2
0.7
-0.2
1.1
-0.3
s_{t-1}
o_{t-1}
h_{t-1}
selective write
$\tilde{s}_t$
x_t
-1.4
-0.4
1
-2
s_t

图 11: RNN 节点计算

我们进行 “选择性写” 的方式是, 将 s_{t-1} 的每个元素乘以一个介于 0 到 1 之间的值, 以计算出一个新的向量 h_{t-1}. 我们将使用这个新向量来计算隐藏表示 s_t. 就像我们使用基于梯度下降优化的参数学习来学习其他参数 (例如 U 和 W) 一样, 我们将从数据中学习 o_{t-1} 的数学方程式如下:

$$o_{t-1} = \sigma\left(U_o x_{t-1} + W_o h_{t-2} + b_o\right), \tag{7}$$

$$h_{t-1} = s_{t-1} \odot o_{t-1}. \tag{8}$$

一旦我们从数据中获知 o_{t-1}, 就将其与 s_{t-1} 相乘以获得新的向量 h_{t-1}. 由于 o_{t-1} 正在控制哪些信息将进入下一个隐藏状态, 因此称为输出门.

- RNN 或 LSTM 中的选择性读.

在计算了新向量 h_{t-1} 之后, 我们将计算一个中间隐藏状态向量 t (标记为绿色). 在本节中, 我们将讨论如何实现选择性读以获得最终的隐藏状态 s_t. 数学公式 $\tilde{s}_t$ 给出如下:

$$\tilde{s}_t = \sigma\left(Ux_t + Wh_{t-1} + b\right). \tag{9}$$

t 从先前状态 h_{t-1} 和当前输入 x_t 捕获所有信息. 但是, 我们可能不希望使用所有新信息, 而只是在构造新的单元结构之前有选择地从中读取信息, 即我们只想从 t 中读取一些信息来计算 s_t.

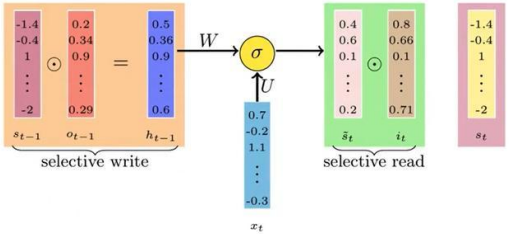
-1.4
-0.4
1
-2
0.2
0.34
0.9
0.29
0.5
0.36
0.9
0.6
W
σ
U
0.7
-0.2
1.1
-0.3
0.4
0.6
0.1
0.2
0.8
0.66
0.1
0.71
-1.4
-0.4
1
-2
s_{t-1}
o_{t-1}
h_{t-1}
selective write
$\tilde{s}_t$
i_t
selective read
s_t
x_t

图 12: RNN 节点计算

就像我们的输出门一样, 在这里, 我们将 t 的每个元素乘以一个新的向量 i_t, 该向量的值在 0-1 之间. 由于向量 i_t 正在控制从当前输入中流入的信息, 因此称为输入门.

i_t 的数学公式如下:

Input gate: $i_t = \sigma\left(W_i h_{t-1} + U_i x_t + b_i\right)$.

Selectively Read: $i_t \odot \tilde{s}_t$.

在输入门中, 我们将前一时间步的隐藏状态信息 h_{t-1} 和当前输入 x_t 连同偏差传递给 sigmoid 函数. 计算的输出将在 0-1 之间, 它将决定从当前输入和上一时间步隐藏状态流入哪些信息. 0 表示不重要, 1 表示重要.

回顾到目前为止所学的知识, 我们拥有先前的隐藏状态 s_{t-1}, 我们的目标是使用选择性取、写和遗忘策略来计算当前状态 s_t:

- Previous state: s_{t-1}.
- Output gate: $o_{t-1} = \sigma(W_o h_{t-2} + U_o x_{t-1} + b_o)$.
- Selectively Write: $h_{t-1} = o_{t-1} \odot \sigma(s_{t-1})$.
- Current (temporary) state: $\tilde{s_t} = \sigma(W h_{t-1} + U x_t + b)$.
- Input gate: $i_t = \sigma(W_i h_{t-1} + U_i x_t + b_i)$.
- Selectively Read: $i_t \odot \tilde{s_t}$.

- RNN 或 LSTM 中的选择性遗忘.

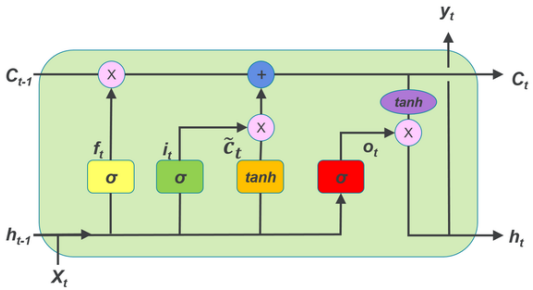

图 13: LSTM 结构

遗忘门决定从隐藏向量中保留或丢弃的信息部分. 本小节讨论如何结合 s_{t-1} 和 来计算当前状态向量 s_t.

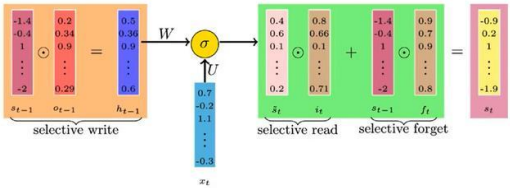

图 14: RNN 节点计算

遗忘门 f(t) 的数学方程式如下:

$$f_t = \sigma\left(U_f x_t + W_f h_{t-1} + b_f\right). \tag{10}$$

在 “遗忘门” 中, 我们将前一时间步的隐藏状态信息 h_{t-1} 和当前输入 x_t 连同偏差传递给 sigmoid 函数. 计算结果将在 0-1 之间, 它将决定要保留或丢弃的信息. 如果该值接近 0, 则表示丢弃; 如果该值接近 1, 则表示保留.

通过组合遗忘门和输入门, 我们可以计算当前的隐藏状态信息.

$$s_t = \tilde{s}_t \odot i_t + s_{t-1} \odot f_t. \tag{11}$$

最终如下所示:

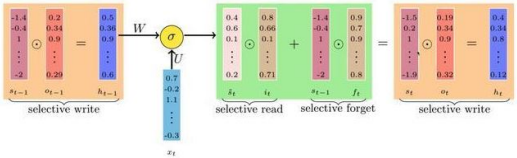

图 15: RNN 节点计算

完整的方程如下所示:

$$
\begin{aligned}
&\text{门} && && \text{状态} \\
&o_t = \sigma\left(W_o h_{t-1} + U_o x_t + b_o\right) && \tilde{s}_t = \sigma\left(W h_{t-1} + U x_t + b\right), \\
&i_t = \sigma\left(W_i h_{t-1} + U_i x_t + b_i\right) && s_t = f_t \odot s_{t-1} + i_t \odot \tilde{s}_t, \\
&f_t = \sigma\left(W_f h_{t-1} + U_f x_t + b_f\right) && h_t = o_t \odot \sigma\left(s_t\right).
\end{aligned}
\tag{12}
$$

注意: 某些版本的 LSTM 架构不会具有 “遗忘门” 功能, 而是仅具有输出门和输入门来控制信息流. 它将仅实现选择性读和选择性写策略.

• 门控循环单元—GRU

门控循环单元是 LSTM 的一个变体. GRU 使用的门较少. 在像 LSTM 一样的门控循环单元中, 我们有一个输出门, 可以控制哪些信息进入下一个隐藏状态. 同样, 我们还有一个输入门, 它控制从当前输入中流入的信息. LSTM 和 GRU 之间的主要区别在于它们将中间隐藏状态向量和先前的隐藏状态表示向量 s 组合的方式. 在 LSTM 中, forget 确定要从 s 中保留多少信息. 在 GRU 中, 我们根据输入门向量 (1-it) 的剩余来决定保留或丢弃多少过去的信息, 而不是遗忘门.

$$f_t = 1 - x_t(1 - i_t). \tag{13}$$

GRU 的完整方程式如下:

$$
\begin{array}{ll}
\text{门} & \text{状态} \\
o_t = \sigma\left(W_o s_{t-1} + U_o x_t + b_o\right) & \tilde{s}_t = \sigma\left(W\left(o_t \odot s_{t-1}\right) + U x_t + b\right) \\
i_t = \sigma\left(W_i s_{t-1} + U_i x_t + b_i\right) & s_t = \left(1 - i_t\right) \odot s_{t-1} + i_t \odot \tilde{s}_t
\end{array}
\tag{14}
$$

从方程式中, 我们可以注意到只有两个门 (输入和输出), 并且我们没有显式计算隐藏状态向量 h. 因此, 我们没有在 GRU 中维护额外的状态向量, 即拥有比 LSTM 更少的计算和更快的训练.

循环神经网络在处理较长句子时的不足之处. RNN 受短期记忆的困扰, 即在信息变形之前, 它只能存储有限数量的状态. 之后, 我们详细讨论了通过使用门机制控制信息流的选择性读, 写和遗忘策略在 LSTM 中的工作方式. 最后, 我们研究了 LSTM 的变体 (称为门控循环单元), 其门数和计算量均比 LSTM 模型少.

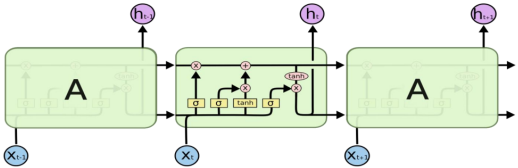

图 16: 长短期记忆 (Long Short-Term Memory, LSTM)

LSTM 改变了循环体 A 的网络结构, 引入 “门” 的概念, 让信息有选择地影响循环神经网络中每个时刻的状态. 长短期记忆 LSTM (Long Short-Term Memory, LSTM) 可以解决梯度消失问题和长期依赖问题. 应用在诸如语言建模和文本生成、机器翻译、语音识别、生成图像描述和视频标记.

图 17: 机器翻译

"man in black shirt is playing guitar."

"construction worker in orange safety vest is working on road."

"two young girls are playing with lego toy."

图 18: 生成图像描述

图 19: 语音识别

2019 年 12 月 8 日-14 日在加拿大温哥华举行, 据官方消息, NeurIPS 今年共收到投稿 6743 篇, 再次打破了历年来的接收记录.

- 接收论文 1429 篇, 其中, Oral 论文 36 篇, 占比 0.5%; Spotlight 论文接收量为 164 篇, 占比 2.4%.
- 强化学习 61 篇, 占比: 4.2%
- 理论大约 (21) 篇, 强化学习技巧 (3) 篇, 框架大约 (3) 篇, 探索和利用 (1) 篇, 元强化学习 (4) 篇,
- 分层强化学习 (2) 篇, 逆强化学习 (2) 篇, 多智能体 (6) 篇, 奖励函数 (2) 篇, 应用 (6) 篇, 其他 (4) 篇

Google 对自动驾驶出租车实现的预测, 已经改变了原来的乐观态度, 变得充满克制. Facebook 的 AI 副总裁 Jerome Pesenti 最近表示, 他的公司和其他公司不应该期待仅通过开发具有更多计算能力和数据的更大的深度学习系统来继续在 AI 方面取得进步.

Arcas 展示了一项模拟细菌的试验. 这些细菌通过人工进化的方式进行觅食和交流. 而 Yoshua Bengio 认为深度学习这个方法行得通, 他正在往工具箱里增加更多的东西. 他在会议上做了主题为从深度学习系统 1 到深度学习系统 2 的演讲, 提出软注意力和深度强化学习方式能够促进解决推理、计划、捕获因果关系等问题. 他的新方法受到了自然智能的启发. 根据意识的先验性进行相关假设, 许多高级依赖关系可以通过稀疏因子图近似地捕获. 软注意力机制构成了一个关键因素, 它可以一次将计算集中于几个概念 (“意识思维”).

Problem

假设 $\mathbf{w}_1(0) = 0.2, \mathbf{w}_2(0) = 0.4, \theta(0) = 0.3, \eta = 0.4$, 请用单层感知器完成逻辑或运算的学习过程.

Problem

学习无监督的方式快速训练生成型循环神经网络 (Recurrent World Models Facilitate Policy Evolution, David Ha 和 Jürgen Schmidhuber)

Problem

运行如下位置的演示代码: Convolutional Neural Network