Neural Episodic Control

Alexander Pritzel APRITZEL@GOOGLE.COM
Benigno Uria BURIA@GOOGLE.COM
Sriram Srinivasan SRSRINIVASAN@GOOGLE.COM
Adrià Puigdomènech ADRIAP@GOOGLE.COM
Oriol Vinyals VINYALS@GOOGLE.COM
Demis Hassabis DEMISHASSABIS@GOOGLE.COM
Daan Wierstra WIERSTRA@GOOGLE.COM
Charles Blundell CBLUNDELL@GOOGLE.COM
DeepMind, London UK

【强化学习 76】NEC

张楚珩
清华大学 交叉信息院博士在读

Neural episodic control 简称 NEC。

原文传送门

Pritzel, Alexander, et al. "Neural episodic control." Proceedings of the 34th International Conference on Machine Learning-Volume 70. JMLR. org, 2017.

特色

专栏前面讲的一篇综述（【强化学习 64】Fast and Slow）里面讲的一种 non-parametric 学习算法，能够大大提高采样效率。由于存在如下疑问，因此来读一下原文。

- 这种方法维护了一个 memory，memory 的 key 是状态的表示，而 value 是 Q 函数。在学习的过程中表示也是逐渐学习的，那么 memory 中以前的状态表示就会和现在产生不匹配。该问题如何解决？
- 文章使用 memory 来表示 Q 函数，方法是对于过去状态类似的 Q 函数进行加权平均。而过去的 Q 函数是通过 bootstrap 方法得到，当 memory（也就是 Q 函数）变化的时候，value 中的数值（原来的 Q 函数值）不再和现在的 Q 函数匹配了。文章有解决这件事么？
- 使用 append-only memory 会导致 memory 不断变大，计算效率会不会很低？内存开销会不会很大？

总体说来，读完之后感觉本文对这些问题都有比较好的处理。

过程

1. Forward pass of NEC

我们先讲 NEC 的结构以及当它训练好之后是如何使用的（即，forward pass），然后再讲它应该如何训练。

首先，NEC 是一个 value-based 方法，它的核心是估计一个 Q 函数。和其他方法不同的是，它是一个 non-parametric 的方法，即它维护一个 memory，每次要估计 (s,a) 的价值函数时，会在

memory 中找到相似的若干个状态，并把这些状态对应的 Q 函数加权平均得到要查询的估计值。

它包括两个部分：第一个部分是一个表示网络，用于把一个状态（可能是高维的，比如图像）编码为低维向量；另一个部分是一个 memory，含有 $|A|$ 个模块，每个模块储存多个 (key, value) 对，key 为相应状态的地位编码，value 为该 (s,a) 对应的 Q 函数值。后者也称作 differentiable neural dictionary（DND）。

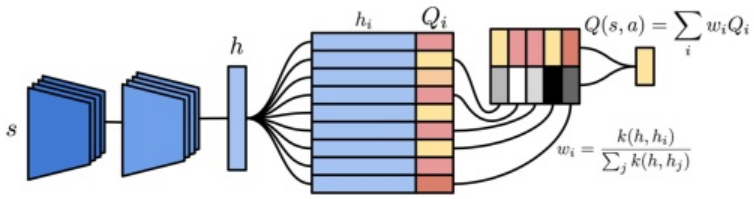

Figure 2. Architecture of episodic memory module for a single action a. Pixels representing the current state enter [illegible] neural network on the bottom left and an estimate of $Q(s,a)$ exits top right. Gradients flow through the entire architecture.

假设我们需要使用上述模型估计 (s,a) 的 Q 函数值。状态 s 首先需要经过编码网络产生 h，该部分就是一个普通的神经网络。对于图像来说，是一个 CNN + MLP 的神经网络。

得到的编码 h 作为相应的 query 去在 memory 中做查询，使用 kd-tree 来查找 memory 中最相近的 $p=50$ 个 entries，距离度量定义为

$$k(h,h_i) = \frac{1}{\|h-h_i\|_2^2 + \delta}. \tag{5}$$

查询的结果为

$$o = \sum_i w_i v_i, \tag{1}$$

其中，求和对于前面得到的 $p=50$ 个近邻来做，v_i 就是储存的 value 值（即图中的 Q_i），w_i 是权重

$$w_i = k(h,h_i) / \sum_j k(h,h_j), \tag{2}$$

2. Backward pass of NEC

先贴出整体算法

Algorithm 1 Neural Episodic Control

$\mathcal{D}$: replay memory.
M_a: a DND for each action a.
N: horizon for N-step Q estimate.
for each episode **do**
  for $t = 1, 2, \ldots, T$ **do**
    Receive observation s_t from environment with embedding h.
    Estimate $Q(s_t, a)$ for each action a via (1) from M_a
    $a_t \leftarrow \epsilon$-greedy policy based on $Q(s_t, a)$
    Take action a_t, receive reward r_{t+1}
    Append $(h, Q^{(N)}(s_t, a_t))$ to M_{a_t}.
    Append $(s_t, a_t, Q^{(N)}(s_t, a_t))$ to $\mathcal{D}$.
    Train on a random minibatch from $\mathcal{D}$.
  end for
end for

下面先讲如何更新 DND，再讲如何训练 representation network。

通过采样能够得到新的 $(h(s_t), Q^{(N)}(s_t, a_t))$ 对，其中目标 Q 函数是 N-step bootstrapped target

$$Q^{(N)}(s_t, a) = \sum_{j=0}^{N-1} \gamma^j r_{t+j} + \gamma^N \max_{a'} Q(s_{t+N}, a') \,. \quad (3)$$

更新的方式如下：如果相应的 h 在 memory 中没有出现过，那么就把它附加（append）在 memory 的尾部；如果出现过就按照如下公式更新该 key 对应的 value（tabular Q-learning update）

$$Q_i \leftarrow Q_i + \alpha(Q^{(N)}(s, a) - Q_i) \,. \quad (4)$$

训练的方式比较直接。注意到整个都是可以进行求导的，因此能够直接对于 representation network 做 SGD，损失函数就是得到的 Q 值和 target Q 值的 L2 损失函数。同时，该求导过程还把 memory 中的 key、value 看做 parameter，也做相应的求导。因此，memory 在添加进去之后也会发生缓慢的变化。主要注意到，整个的训练是比较缓慢的（相比于直接 append 新的 Q 值以及上式中的 α）。

3. 前述疑问

前面关心的是随着 representation network 的训练，memory 中的 key-value 如果不变就会和当前的结构不匹配，但是这里是同时训练 representation network 和 memory，因此个人感觉是说得通的。

另外一个问题是 memory 的效率问题，查询部分使用了 kd-tree 提高了查询效率。同时，对于相同的 embedding 也会去更新而不是附加（虽然不太懂一个 embedding 是一个实向量，为什么会有“相同”，可能是距离低于某个阈值吧，文中没说）。另外 memory 的总数目也限定在 10^6，超出限制就会抹掉最早的 entries。

发布于 2019-06-22

强化学习 (Reinforcement Learning)