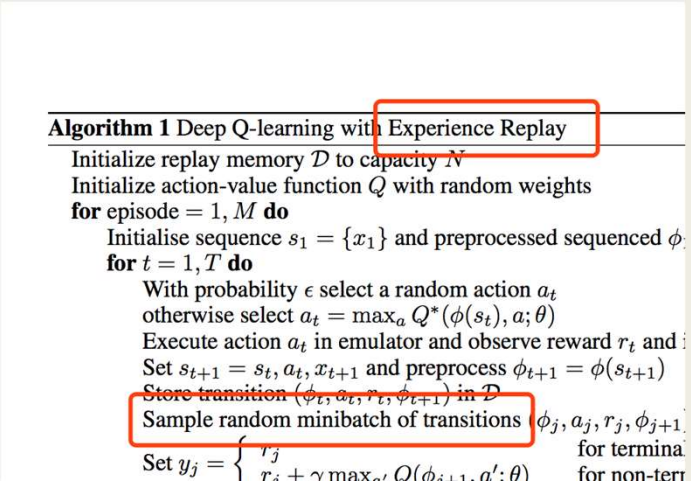
克服神经网络的灾难性遗忘问题

- 迁移学习 以及 继续学习 （continual learning）
  - *工业应用不常见一个模型解决多个问题*
  - *一个随时间改变了特征空间的分布的数据样本重新训练模型后变成了一个新的模型*
  - *即便是一个数据集，如果不小心选择batch num也会造成无法学习的情况*
  - ***但是真正的人工智能不是这样的，继续学习的问题必须解决***
    - the sequence of tasks may not be explicitly labelled,
    - tasks may switch unpredictably,
    - and any individual task may not recur for long time intervals

目前的workaround

- 一次训练全部的数据集（训练大量的数据集）
  - *Current approaches have typically ensured that data from all tasks are simultaneously available during training*
- 如果是时间序列的问题
  - *使用强化学习玩Atari游戏的，一种局面可能长时间不出现*
  - *解决办法是添加memory 和replay*



Algorithm 1 Deep Q-learning with Experience Replay

Initialize replay memory $\mathcal{D}$ to capacity N
Initialize action-value function Q with random weights
for episode $= 1, M$ **do**
  Initialise sequence $s_1 = \{x_1\}$ and preprocessed sequenced ϕ
  for $t = 1, T$ **do**
    With probability ϵ select a random action a_t
    otherwise select $a_t = \max_a Q^*(\phi(s_t), a; \theta)$
    Execute action a_t in emulator and observe reward r_t and
    Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$
    Store transition $(\phi_t, a_t, r_t, \phi_{t+1})$ in $\mathcal{D}$
    Sample random minibatch of transitions $(\phi_j, a_j, r_j, \phi_{j+1}$
    Set $y_j = \begin{cases} r_j & \text{for termina} \\ r_j + \gamma \max_{a'} Q(\phi_{j+1}, a'; \theta) & \text{for non-terr} \end{cases}$

仿生学带给我们希望 ---- 插一条广告

一切没有仿生学依据的设计都是伪科学

- 人的神经网络里没有深度多层
- 人的神经网络里面没有BP
- 人学习一张动物图片不需要连续看上几十遍
- **人在连续学习的时候不需要同时将之前的任务完整的记忆下来**

哺乳动物如何学习

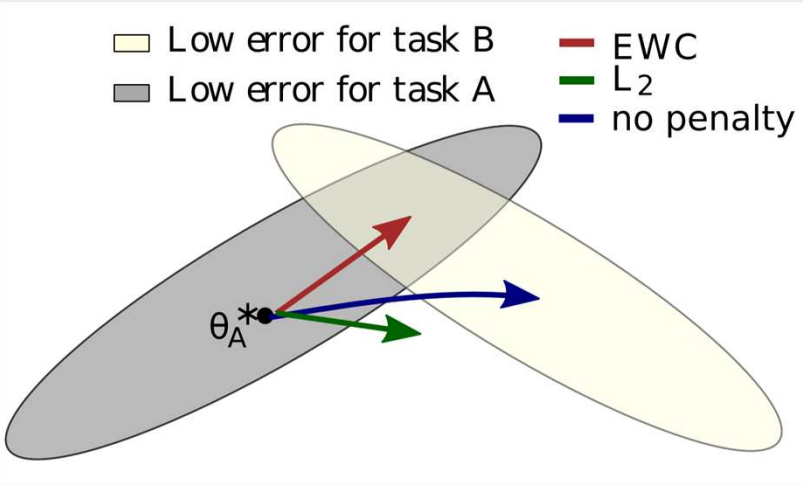
- Learning takes place through the formation and elimination of postsynaptic dendritic spines over time
  - *学习是通过形成或者是消除**突触后突状棘**发生的行为*
  - *一个跟之前学习到的任务相关的突出会受到某种保护，在学习新的任务的时候收到的更改相对较少*
    - ---- Sleep promotes branch-specific formation of dendritic spines after learning

Can catastrophic forgetting be overcome by using a similar approach of *varying individual neuron's plasticity* depending on each *neuron's importance*?

能通过基于权重的重要性来改变权重的大小吗

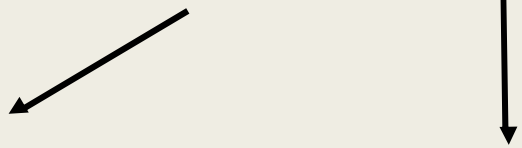
一些理论基础

- 一些研究表明：对于一个任务，很多不同的权重组（参数组）（大型网络）（过参数化：设置多于需要的参数个数，神经元个数）能达到相似的效果
  - https://pdfs.semanticscholar.org/4d3f/050801bd76ef10855ce115c31b301a83b405.pdf
- 那么过参数化就很有可能在满足一个任务的最优参数组附近找到满足另一个任务的最优参数组（无论如何不要过分的改变一个已经为了A任务优化好的参数）
  - *绿色很可能找到的是一个中庸的参数组*
  - *蓝色又会找到只对B来说最优的参数组*
  - *红色EWC才能找到最优的交集*



EWC算法：如果确定对A任务重要的参数

- $\log p(\theta|\mathrm{D}) = \log p(D|\theta) + \log p(\theta) - \log p(D)$
- 假若D是由两个相互独立事件组成的，那么有：
  - $\log p(\theta|D) = \log p(D_B|\theta) + \log p(\theta|D_A) - \log p(D_B)$



D_B的对数似然概率
（交叉熵，损失函数？）

D_A下的θ，已知

- $\log p(\theta|D) = \log p(D_B|\theta) + \log p(\theta|D_A) - \log p(D_B)$

D_B的对数似然概率
（交叉熵，损失函数？）

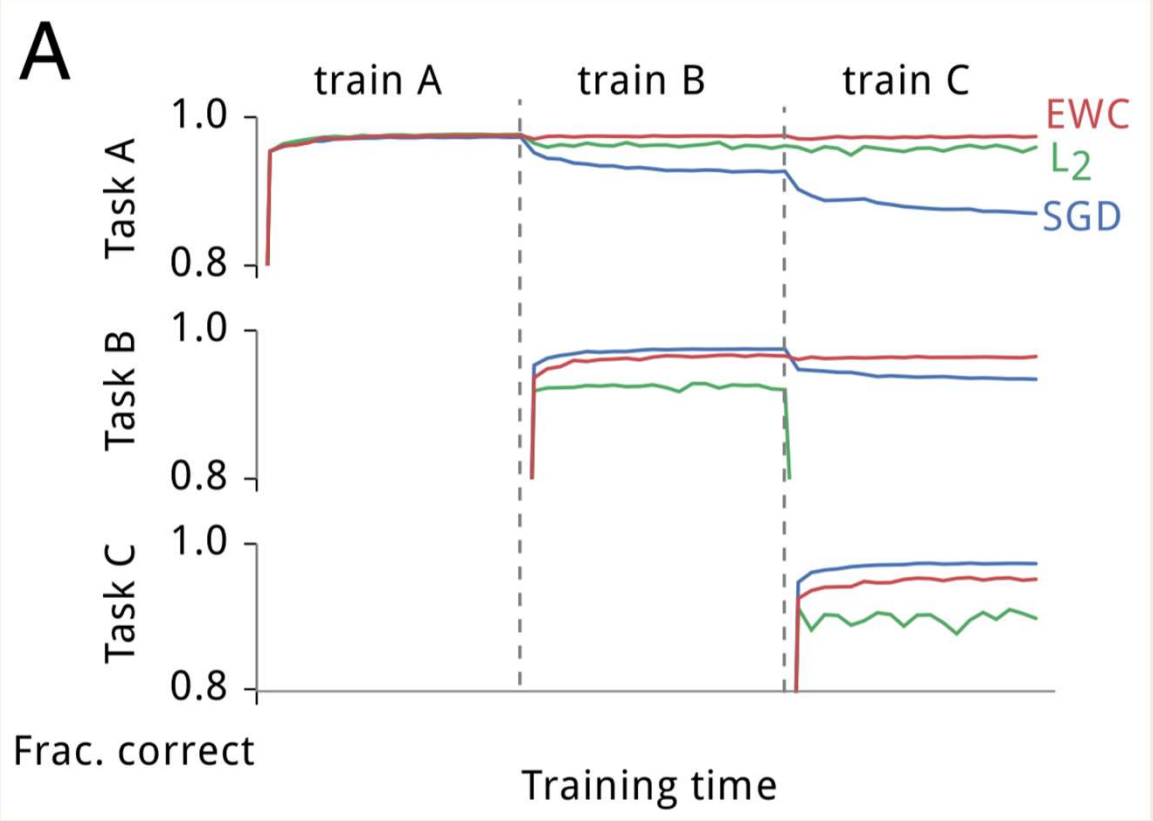
D_A下的θ，已知

- 其实这个公式告诉我们，在训练B任务的时候，全局最优参数组只跟B任务的损失有关，而且此时你要保护D_A下的θ（保护D_A下重要的θ）

如何训练D_B

- 训练D_B时的损失函数 $\mathcal{L}(\theta) = \mathcal{L}_B(\theta) + \sum \frac{\lambda}{2} F_i \left(\theta_i - \theta^*_{A,i}\right)^2$
- 其中F为费雪信息矩阵函数
  - *（用来计算参数组中的参数所含有的信息量）*
  - *含有信息量较大的对该参数的惩罚越大（quadratic）*

效果 – 多份MNIST样本



- https://rylanschaeffer.github.io/content/research/overcoming_catastrophic_forgetting/main.html
- https://github.com/ariseff/overcoming-catastrophic
- http://www.inference.vc/comment-on-overcoming-catastrophic-forgetting-in-nns-are-multiple-penalties-needed-2/
- https://arxiv.org/pdf/1612.00796.pdf