第七讲：函数逼近与深度神经网络

主讲人 陈达贵

清华大学自动化系

在读硕士

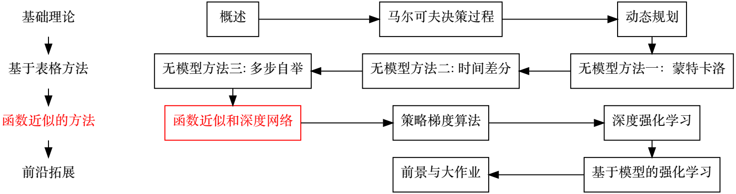
深蓝学院

shenlanxueyuan.com

强化学习理论与实践

目录

章节目录


基础理论
基于表格方法
函数近似的方法
前沿拓展
概述
马尔可夫决策过程
动态规划
无模型方法一：蒙特卡洛
无模型方法二: 时间差分
无模型方法三: 多步自举
函数近似和深度网络
策略梯度算法
深度强化学习
基于模型的强化学习
前景与大作业

本章目录

值函数近似

- 之前所提到方法均为基于值函数的方法
  - 通过求解最优值函数来求解最优策略
- 而且基于表格的方法

	a_1	a_2	a_3
s_1	2.0	1.0	0.0
s_2	1.7	2.0	1.1
s_3	2.3	1.1	4.3

backup
$Q(s_2, a_2)$

	a_1	a_2	a_3
s_1	2.0	1.0	0.0
s_2	1.7	3.0	1.1
s_3	2.3	1.1	4.3

- 表格的大小会随着状态数量和动作数量快速膨胀
- 对于表格中某一项的更新不会影响到其他项的更新

大规模强化学习问题

强化学习能够用来解决大规模的问题

- 西洋双陆棋：10^{20} 种不同的状态
- 围棋：10^{170} 种不同的状态
- 机器人控制以及无人机控制：连续状态
- 图像状态：$256^{\text{像素点数}}$

我们如何将无模型的方法应用到如此大规模的强化学习问题？

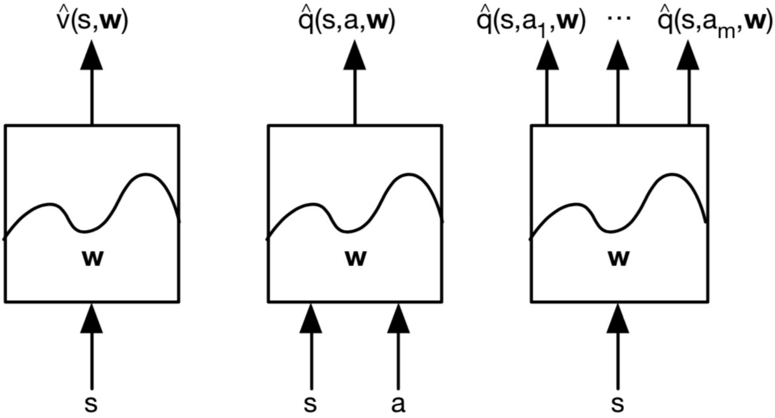
值函数近似

- 过去我们都是使用表格来表示值函数
  - 每个状态 s 都有一个 V 函数值 $V(s)$——V 函数向量
  - 每个状态动作对 $\langle s, a\rangle$ 都有一个 Q 函数值——Q 函数矩阵
- 在大规模 MDPs 中会存在如下的问题:
  - 需要在内存空间中存储大量的状态或者动作
  - 学习太缓慢
- 解决大规模 MDPs 的方法
  - 使用函数近似的方法

$$\hat{v}(s, \mathbf{w}) \approx v_\pi(s)$$
$$\hat{q}(s, a, \mathbf{w}) \approx q_\pi(s, a)$$

  - 从已经经历过的状态推广到未见的状态
  - 可以使用 MC 或者 TD 更新参数 $\mathbf{w}$

值函数近似的类型



函数近似器

我们考虑可微的函数近似器，比如

- 线性模型
- 神经网络
- 决策树
- 最近邻法
- 傅立叶基
- 小波变换
- ...

目录

目录

梯度下降

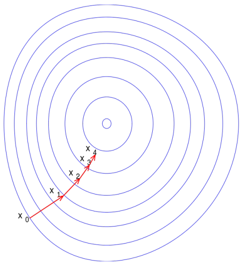
- 如果 $J(\mathbf{w})$ 是参数向量 $\mathbf{w}$ 的可微函数
- 那么 $J(\mathbf{w})$ 的梯度定义为

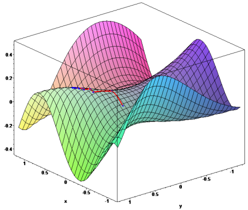
$$\nabla_{\mathbf{w}} J(\mathbf{w}) = \begin{bmatrix} \frac{\partial J(\mathbf{w})}{\partial \mathbf{w}_1} \\ \vdots \\ \frac{\partial J(\mathbf{w})}{\partial \mathbf{w}_n} \end{bmatrix}$$

- 为了能找到 $J(\mathbf{w})$ 的局部最优值
- 沿负梯度方向更新参数向量 $\mathbf{w}$

$$\Delta \mathbf{w} = -\frac{1}{2}\alpha \nabla_{\mathbf{w}} J(\mathbf{w})$$

这里 α 代表步长





值函数近似和随机梯度下降

- 目标：寻找参数向量 $\mathbf{w}$，以最小化近似值函数 $\hat{v}(s, \mathbf{w})$ 和真实的值函数 $v_\pi(s)$ 之间的均方误差 (mean-squared error, MSE)

$$J(\mathbf{w}) = \mathbb{E}_\pi \left[\left(v_\pi(S) - \hat{v}(S, \mathbf{w}) \right)^2 \right]$$

- 梯度下降算法会寻找局部最小值

$$\Delta \mathbf{w} = -\frac{1}{2} \alpha \nabla_{\mathbf{w}} J(\mathbf{w}) = \alpha \mathbb{E}_\pi \left[\left(v_\pi(S) - \hat{v}(S, \mathbf{w}) \right) \nabla_{\mathbf{w}} \hat{v}(S, \mathbf{w}) \right]$$

- 随机梯度下降算法会对梯度进行采样

$$\Delta \mathbf{w} = \alpha \left(v_\pi(S) - \hat{v}(S, \mathbf{w}) \right) \nabla_{\mathbf{w}} \hat{v}(S, \mathbf{w})$$

- 其他常见的优化算法：牛顿法，批量梯度下降，动量梯度下降，RMSprop，Nesterov，Adagrad，Adam...

目录

特征向量

- 通过一个特征向量表达状态

$$\mathbf{x}(S) = \begin{bmatrix} \mathbf{x}_1(S) \\ \vdots \\ \mathbf{x}_n(S) \end{bmatrix}$$

- 常用于特征之间关系不明显的数据

线性值函数近似

- 通过特征的线性组合表达值函数

$$\hat{v}(S, \mathbf{w}) = \mathbf{x}(S)^T \mathbf{w} = \sum_{j=1}^{n} \mathbf{x}_j(S) \mathbf{w}_j$$

- 目标函数是 $\mathbf{w}$ 的二次形式

$$J(\mathbf{w}) = \mathbb{E}_\pi \left[\left(v_\pi(S) - \mathbf{x}(S)^T \mathbf{w} \right)^2 \right]$$

- 随机梯度下降会收敛到全局最优值
- 在线性值函数近似的情况下，梯度的计算变得非常简单

$$\nabla_{\mathbf{w}} \hat{v}(S, \mathbf{w}) = \mathbf{x}(S)$$
$$\Delta \mathbf{w} = \alpha \left(v_\pi(S) - \hat{v}(S, \mathbf{w}) \right) \mathbf{x}(S)$$

表格检索特征

- 表格检索是线性值函数近似的一种特殊形式
- 使用表格检索特征

$$x^{table}(S) = \begin{bmatrix} \mathbf{1}(S = s_1) \\ \vdots \\ \mathbf{1}(S = s_n) \end{bmatrix}$$

- 参数向量 $\mathbf{w}$ 本质上相当于给了每个状态对应的值函数

$$\hat{v}(S, \mathbf{w}) = \begin{bmatrix} \mathbf{1}(S = s_1) \\ \vdots \\ \mathbf{1}(S = s_n) \end{bmatrix}^T \begin{bmatrix} \mathbf{w}_1 \\ \vdots \\ \mathbf{w}_n \end{bmatrix}$$

目录

增量式评价算法

- 之前是假设了给定了真实的值函数 $v_\pi(s)$
- 但是在 RL 环境中，并不知道真实的值函数，只有奖励值
- 直观地，我们用目标值替代 $v_\pi(s)$
  - 对于 MC，目标值是回报值 G_t

$$\Delta \mathbf{w} = \alpha(G_t - \hat{v}(S_t, \mathbf{w}))\nabla_{\mathbf{w}} \hat{v}(S_t, \mathbf{w})$$

  - 对于 TD(0)，目标值是 TD 目标值 $R_{t+1} + \gamma \hat{v}(S_{t+1}, \mathbf{w})$

$$\Delta \mathbf{w} = \alpha \left(R_{t+1} + \gamma \hat{v}(S_{t+1}, \mathbf{w}) - \hat{v}(S_t, \mathbf{w})\right) \nabla_{\mathbf{w}} \hat{v}(S_t, \mathbf{w})$$

  - 对于 TD(λ)，目标值是 λ 回报值 G_t^λ

$$\Delta \mathbf{w} = \alpha \left(G_t^\lambda - \hat{v}(S_t, \mathbf{w})\right) \nabla_{\mathbf{w}} \hat{v}(S_t, \mathbf{w})$$

值函数近似下的 MC

- 回报值 G_t 是真实值函数 $v_\pi(S_t)$ 的无偏估计
- 构建监督学习的“训练数据”

$$\langle S_1, G_1\rangle, \langle S_2, G_2\rangle, \cdots, \langle S_T, G_T\rangle$$

- 更新参数

$$\begin{aligned}\Delta \mathbf{w} &= \alpha({\color{red}G_t} - \hat{v}(S_t, \mathbf{w}))\nabla_{\mathbf{w}}\hat{v}(S_t, \mathbf{w}) \\ &= \alpha(G_t - \hat{v}(S_t \mathbf{w}))\mathbf{x}(S_t) \quad \text{线性情况下}\end{aligned}$$

值函数近似下的 TD

- TD 目标值 $R_{t+1} + \gamma \hat{v}(S_{t+1}, \mathbf{w})$ 是真实值函数 $v_\pi(S_t)$ 的有偏估计
- 仍然可以构建监督学习的 “训练数据”

$$\langle S_1, R_2 + \gamma \hat{v}(S_2, \mathbf{w})\rangle, \langle S_2, R_3 + \gamma \hat{v}(S_3, \mathbf{w})\rangle, \cdots, \langle S_{T-1}, R_T\rangle$$

- 更新参数

$$\begin{aligned}\Delta \mathbf{w} &= \alpha \left(R_{t+1} + \gamma \hat{v}(S_{t+1}, \mathbf{w}) - \hat{v}(S_t, \mathbf{w}) \right) \nabla_{\mathbf{w}} \hat{v}(S_t, \mathbf{w}) \\ &= \alpha \delta \mathbf{x}(S) \quad \text{线性情况下}\end{aligned}$$

值函数近似下的 TD(λ)

- λ 回报值 G_t^λ 也是真实值函数 $v_\pi(s)$ 的有偏估计
- 仍然可以构建监督学习的“训练数据”

$$\langle S_1, G_1^\lambda\rangle, \langle S_2, G_2^\lambda\rangle, \cdots, \langle S_{T-1}, G_{T-1}^\lambda\rangle$$

- 前向视角的 TD(λ)

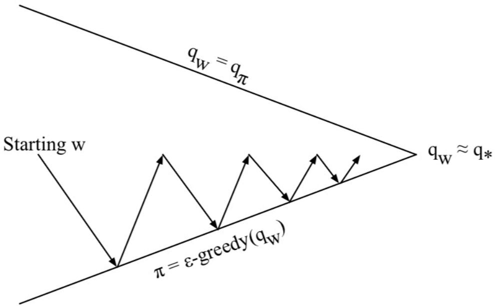
$$\begin{aligned}\Delta\mathbf{w} &= \alpha\left(G_t^\lambda - \hat{v}(S_t,\mathbf{w})\right)\nabla_{\mathbf{w}}\hat{v}(S_t,\mathbf{w}) \\ &= \alpha(G_t^\lambda - \hat{v}(S_t,\mathbf{w}))\mathbf{x}(S_t) \quad \text{线性情况下}\end{aligned}$$

- 后向视角的 TD(λ)

$$\begin{aligned}\delta_t &= R_{t+1} + \gamma\hat{v}(S_{t+1},\mathbf{w}) - \hat{v}(S_t,\mathbf{w}) \\ E_t &= \gamma\lambda E_{t-1} + \nabla_{\mathbf{w}}\hat{v}(S_t,\mathbf{w}) = \gamma\lambda E_{t-1} + \mathbf{x}(S_t) \quad \text{线性情况下} \\ \Delta\mathbf{w} &= \alpha\delta_t E_t\end{aligned}$$

目录

通用策略迭代



- **策略评价**：近似化策略评价，$\hat{q}(\cdot, \cdot, \mathbf{w}) \approx q_\pi$
- **策略提升**：ε 贪婪策略提升

对 Q 函数的近似

- 近似 Q 函数

$$\hat{q}(S, A, \mathbf{w}) \approx q_\pi(S, A)$$

- 最小化近似值和真实值的均方误差

$$J(\mathbf{w}) = \mathbb{E}_\pi\left[\left(q_\pi(S, A) - \hat{q}(S, A, \mathbf{w})\right)^2\right]$$

- 使用随机梯度下降来找到局部最小值

$$\begin{aligned} -\frac{1}{2}\nabla_{\mathbf{w}} J(\mathbf{w}) &= \left(q_\pi(S, A) - \hat{q}(S, A, \mathbf{w})\right) \nabla_{\mathbf{w}} \hat{q}(S, A, \mathbf{w}) \\ \Delta\mathbf{w} &= \alpha\left(q_\pi(S, A) - \hat{q}(S, A, \mathbf{w})\right) \nabla_{\mathbf{w}} \hat{q}(S, A, \mathbf{w}) \end{aligned}$$

线性 Q 函数近似

- 用一个特征向量表示某一个具体的 S, A

$$\mathbf{x}(S,A) = \begin{bmatrix} \mathbf{x}_1(S,A) \\ \vdots \\ \mathbf{x}_n(S,A) \end{bmatrix}$$

- 通过特征的线性组合表达 Q 函数

$$\hat{q}(S,A,\mathbf{w}) = \mathbf{x}(S,A)^T\mathbf{w} = \sum_{j=1}^{n} x_j(S,A)\mathbf{w}_j$$

- 随机梯度下降

$$\begin{aligned} \nabla_{\mathbf{w}}\hat{q}(S,A,\mathbf{w}) &= \mathbf{x}(S,A) \\ \Delta\mathbf{w} &= \alpha\left(q_\pi(S,A) - \hat{q}(S,A,\mathbf{w})\right)\mathbf{x}(S,A) \end{aligned}$$

增量式策略优化算法

同样地，我们用目标值替换真实的 $q_\pi(S, A)$

- 对于 MC，目标值即回报值 G_t

$$\Delta \mathbf{w} = \alpha \left(G_t - \hat{q}\left(S_t, A_t, \mathbf{w}\right)\right) \nabla_{\mathbf{w}} \hat{q}\left(S_t, A_t, \mathbf{w}\right)$$

- 对于 TD(0)，目标值是 TD 目标值 $R_{t+1} + \gamma Q(S_{t+1}, A_{t+1})$

$$\Delta \mathbf{w} = \alpha \left(R_{t+1} + \gamma \hat{q}\left(S_{t+1}, A_{t+1}, \mathbf{w}\right) - \hat{q}\left(S_t, A_t, \mathbf{w}\right)\right) \nabla_{\mathbf{w}} \hat{q}\left(S_t, A_t, \mathbf{w}\right)$$

- 对于前向视角的 TD(λ)，目标值是针对 Q 的 λ 回报值

$$\Delta \mathbf{w} = \alpha \left(q_t^\lambda - \hat{q}\left(S_t, A_t, \mathbf{w}\right)\right) \nabla_{\mathbf{w}} \hat{q}\left(S_t, A_t, \mathbf{w}\right)$$

- 对于后向视角的 TD(λ)，

$$\begin{aligned}
\delta_t &= R_{t+1} + \gamma \hat{q}\left(S_{t+1}, A_{t+1}, \mathbf{w}\right) - \hat{q}\left(S_t, A_t, \mathbf{w}\right) \\
E_t &= \gamma \lambda E_{t-1} + \nabla_{\mathbf{w}} \hat{q}\left(S_t, A_t, \mathbf{w}\right) \\
\Delta \mathbf{w} &= \alpha \delta_t E_t
\end{aligned}$$

目录

策略评价时的收敛问题

在/离-策略	算法	表格检索	线性	非线性
在策略	MC	✓	✓	✓
	TD(0)	✓	✓	×
	TD(λ)	✓	✓	×
离策略	MC	✓	✓	✓
	TD(0)	✓	×	×
	TD(λ)	✓	×	×

- 表格检索的收敛性最好
- 离策略的收敛性比在策略要差
- 非线性近似会影响收敛性
- TD 算法的收敛性不如 MC

策略优化算法的收敛性

算法	表格检索	线性	非线性
MC 优化算法	$\checkmark$	$(\checkmark)$	$\times$
Sarsa	$\checkmark$	$(\checkmark)$	$\times$
Q 学习	$\checkmark$	$\times$	$\times$

其中 $(\checkmark)$ 表示接近最优值函数

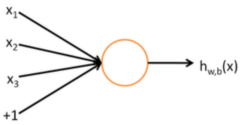
目录

参考资料

由于神经网络和卷积神经网络并不是我们这门课的重点，因此这里主要是简述，具体大家可以参考如下的资料

- UFLDL 教程，有关神经网络和反向传播算法
  - http://ufldl.stanford.edu/wiki/index.php/UFLDL_Tutorial (可以选择中文)
- CS231
  - http://cs231n.github.io/

神经网络单元



本质上是一个线性特征组合 + 一个非线性激活函数。在这里

$$h_{W,b}(x) = f(W^T x + b) = f(\sum_{i=1}^{3} W_i x_i + b)$$

- 其中函数 $f : \mathcal{R} \mapsto \mathcal{R}$ 称为 “激活函数”
- W, b 表示这个神经元的参数
- 通过 W, b 对输入的特征进行了特征组合，然后经过非线性映射，将输出投射到新的空间

激活函数

为什么要激活函数？

- 特征组合运算都是线性运算。
- 分隔不同的神经网络层

常用的激活函数

- sigmoid.

$$f(z) = \frac{1}{1 + e^{-z}}$$

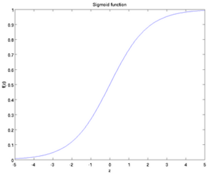
- tanh.

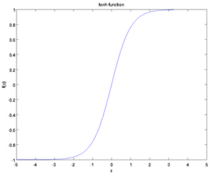
$$f(z) = tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$$

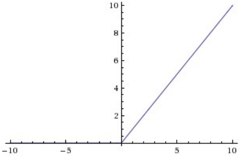
- ReLU(Rectified Linear Unit)

$$f(z) = ReLU(z) = \max(0, z)$$

激活函数

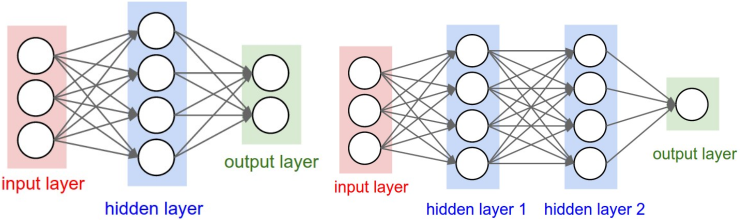






- sigmoid 和 tanh 形式形同，取值范围有所差别。([0,1], [-1, 1])
- 激活函数除了要求非线性外，还要求容易求导数
  - sigmoid： $f'(z) = f(z)(1 - f(z))$
  - tanh: $f'(z) = 1 - (f(z))^2$
- sigmoid 和 tanh 具有饱和效应，不适用于太深的神经网络。

多层神经网络



- 输入层，输出层，隐藏层
- 每一层的输出本质上是将输入特征进行了变换
- 如果用向量的形式表示 $f([z_1, z_2, z_3]) = [f(z_1), f(z_2), f(z_3)]$，且用 $W^{(l)}, b^{(l)}$ 表示第 l 层的权重和偏置

$$a^{(2)} = f(W^{(1)}x + b^{(1)})$$

$$a^{(3)} = f(W^{(2)}a^{(2)} + b^{(2)})$$

- 上述过程叫做前向传播——给定第 l 层的输出，可以计算 $l+1$ 层的输出

反向传播

神经网络中最重要的概念之一就是反向传播。

- 前向传播指的是从前往后依次计算输出的过程
- 反向传播指的是从后往前依次计算梯度的过程
- 损失函数

$$J = \frac{1}{2}\left\|h_{W,b}(x) - y\right\|^2$$

- 如果要更新参数 $W^{(1)}, b^{(1)}, W^{(2)}, b^{(2)}$，那么就需要计算相应的梯度

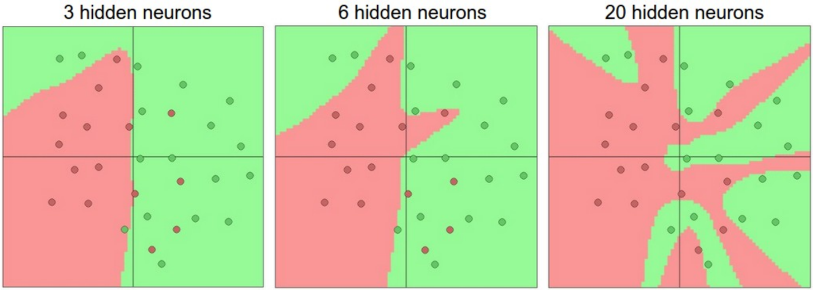
$$\nabla_{W^{(1)}} J, \nabla_{b^{(1)}} J, \nabla_{W^{(2)}} J, \nabla_{b^{(2)}} J \ldots$$

- 反向传播本质上就是利用了求导的链式法则:

$$\frac{\partial y}{\partial x} = \frac{\partial y}{\partial z}\frac{\partial z}{\partial x} \quad \frac{\partial J}{\partial W^{(2)}} = \frac{\partial J}{\partial a^{(2)}}\frac{\partial a^{(2)}}{\partial W^{(2)}} \quad \frac{\partial J}{\partial W^{(1)}} = \frac{\partial J}{\partial a^{(2)}}\frac{\partial a^{(2)}}{\partial a^{(1)}}\frac{\partial a^{(1)}}{\partial W^{(1)}}$$

网络的大小和深度

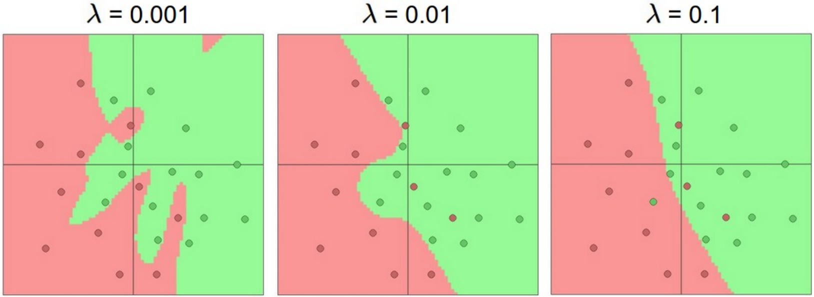
- 多层神经网络几乎可以拟合任意连续函数
- 网络越大越深，模型复杂度越高，对函数的拟合能力越强
- 网路越大越深，参数也会越多，容易导致过拟合



正则化

为了防止过拟合，除了修改网络大小外，还可以在损失函数里加入正则化项

$$J_{\text{带正则化}} = J + \frac{\lambda}{2}\sum_{l}\sum_{i}\sum_{j}\left(W_{ji}^{(l)}\right)^2$$



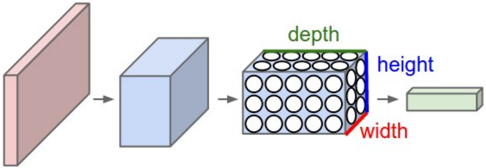
课程要求

- 对于本门课而言
  - 神经网络是工具的一种
  - 要求知道如何使用
  - 深度学习框架可以自动求导
- 不过要真正地深入了解深度强化学习，深度学习，神经网络的知识必不可少

目录

卷积神经网络的整体结构

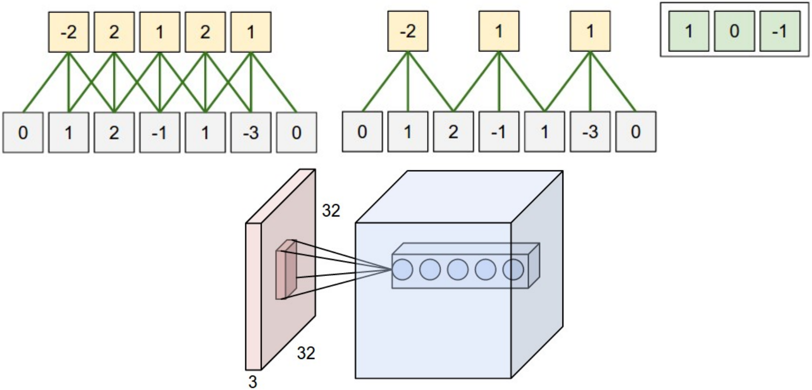
- 常规的神经网络每一层都是一维向量
- 卷积神经网络的每一层是三维数据：高，宽，深



- 由宽和高组成的平面称为特征图 (feature map)
- 深度表示特征图的个数，也叫做通道 (channal) 数
- 特别地，当输入是一个图像时，它表示深度为 3(RGB) 的特征图

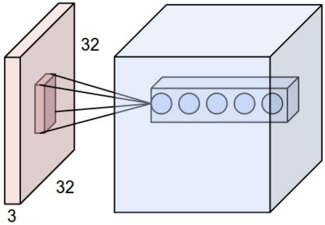
卷积

- 局部连接
- 参数共享
- 关键概念
  - 卷积核、步长（stride）、pad



卷积层

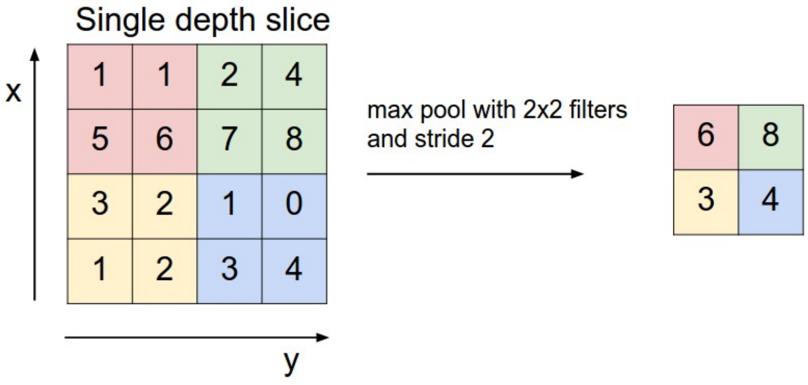
- 对于每一个卷积核
  - 使用一个滑动窗口在输入层滑动
  - 每滑动一次，就会得到一个输出值
  - 滑动完整个输入层就得到一个输出的特征图
- 每有一个卷积核就提取一种特征，得到一个特征图
- 使用多个卷积核就能得到多个特征图



http://cs231n.github.io/convolutional-networks/

池化层（Pooling）

- 池化层本质上是一种降采样
- 降低特征图大小



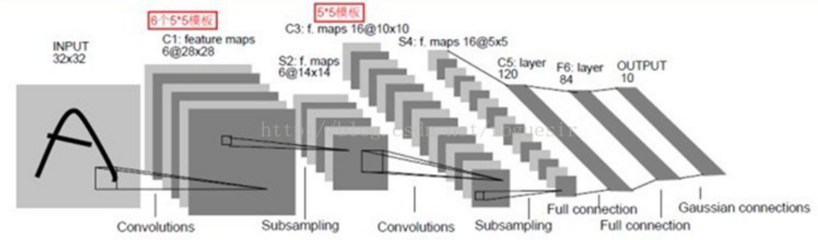
全连接层

- 全连接层就是传统神经网络中的连接方式
- 每一层是一个一维向量
- 也可以用从三维的角度去看待全连接层
  - 即宽和高均为 1 的层
- 一般卷积神经网络的最后几层是全连接层
  - 输出不同类别的概率
  - 输出不同动作所对应的 Q 值 (RL 问题中)

目录

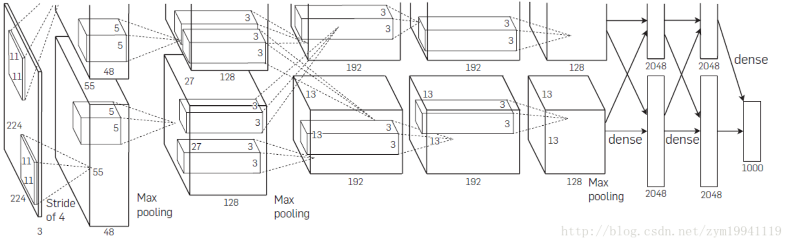
LeNet5

1994 年 Yann LeCun



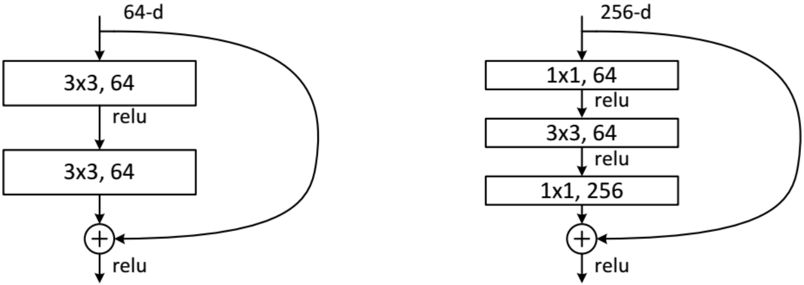
- Conv, Ave-pooling, sigmoid

AlexNet



- 2012 年引发深度学习热潮
- 使用了多个 GPU 分布式训练
- 使用了 ReLU，数据增广，dropout

ResNet



- 提出残差结构，使得超深的网络训练成为现实
- 2015 年 152 层的结构，导致识别率超过人类
- 2016 CVPR best paper

深度学习的其他拓展

- 循环神经网络 RNN, LSTM
- Attention 结构
- 可微分存储器
- ...