

# 强化学习

## 第八讲：神经网络与逆强化学习

教师：赵冬斌 朱圆恒 张启超

中国科学院大学  
中国科学院自动化研究所



May 28, 2021

## ■ 1.1 神经网络

神经网络的定义

神经网络的训练

## ■ 1.2 卷积神经网络

卷积层

池化层

## ■ 1.3 逆强化学习

模仿学习

逆向强化学习

# 神经网络的定义

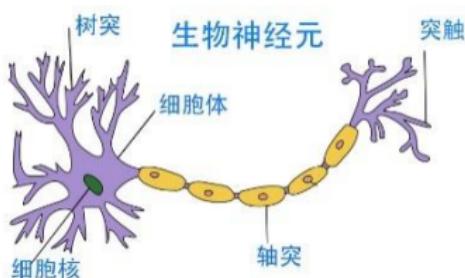


人工神经网络(这里简称为神经网络)的定义:

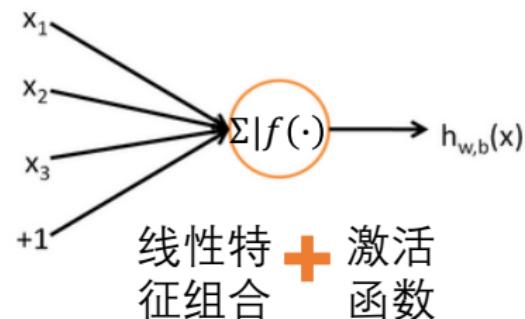
神经网络是具有适应性的简单单元组成的并行互联的网络系统，能够模拟生物神经系统对真实世界物体所作出的交互反应。主要用于对函数进行估计或近似。

## 神经网络单元 (神经元)

生物神经元

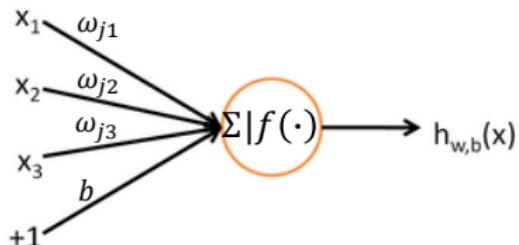


人工神经元



# 神经网络的定义

## 神经网络单元（神经元）



$$h_{W,b}(x) = f(W^T x + b)$$

线性特征组合  $\textcolor{red}{+}$  激活函数



$W$  为神经网络权重  
 $b$  为偏置项

$f(\cdot)$  为激活函数

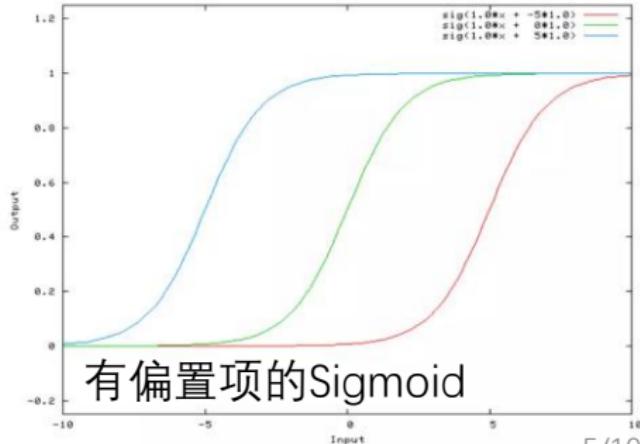
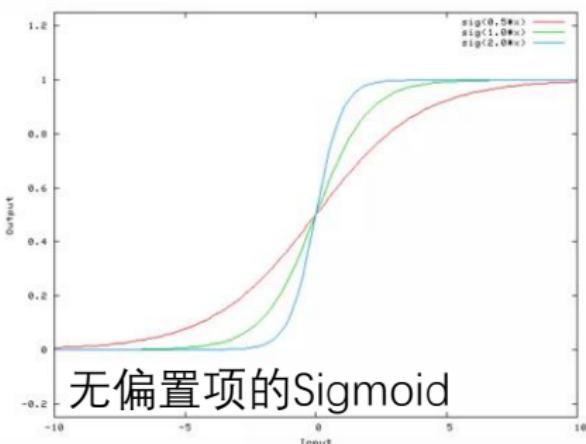
# 神经网络的定义



## 神经网络单元（神经元）

$$h_{W,b}(x) = f(W^T x + b)$$

通过  $W$  和  $b$  对输入进行线性特征组合，经过激活函数，将输出投射到新的空间，输出范围由激活函数  $f(\cdot)$  决定。



# 神经网络的定义



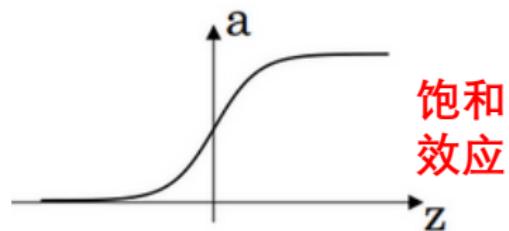
## 典型的非线性激活函数

### 1. Sigmoid函数

常被用于分类任务输出层

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

梯度弥散



优点：

输出有界(0,1);  
单调连续递增;  
求导容易;

缺点：

左右两侧梯度接近于0,  
学习效率会变很慢;  
非零均值;  
对指数操作计算量大

# 神经网络的定义

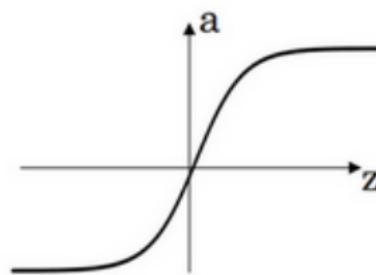


## 典型的非线性激活函数

### 2. Tanh. 函数

$$\begin{aligned} g(z) &= \tanh(z) \\ &= \frac{e^z - e^{-z}}{e^z + e^{-z}} \end{aligned}$$

**梯度弥散**



优点：

tanh 的收敛速度比 Sigmoid 快；  
零均值

缺点：

左右两侧梯度接近于0，  
学习效率会变很慢；  
对指数操作计算量大

# 神经网络的定义



## 典型的非线性激活函数

### 3. ReLU(Rectified Linear Unit)

线性整流函数

$$g(z) = \max(0, z)$$

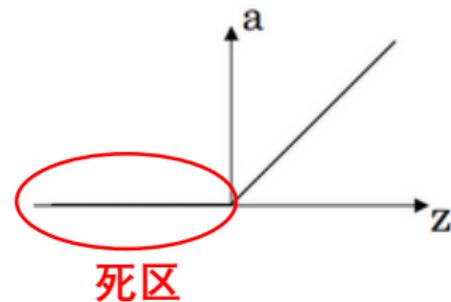
输出值非负，没有上界

稀疏激活，缓解过拟合

单调递增，非零均值

分段线性函数，计算速度快

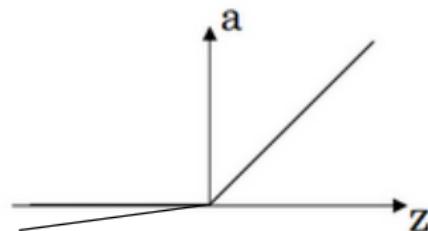
常被用于CNN网络



### 4. Leaky ReLU

$$g(z) = \max(0.01z, z)$$

改进了负数区域的死区问题



## 神经网络的训练

## 单隐层神经网络

## 1. 前向传播 从前往后依次计算输出的过程

输入层

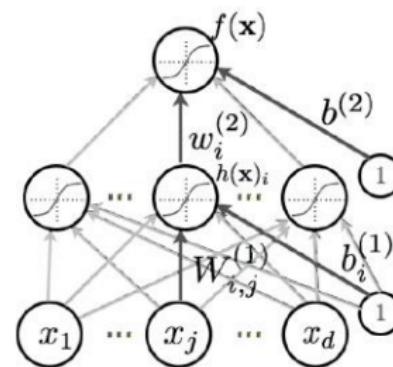
$$x = [x_1, \dots, x_d]^T$$

隐藏层

$$h(x) = h(W^{(1)T}x + b^{(1)})$$

输出层

$$\sigma(x) = f(W^{(2)T}h(x) + b^{(2)})$$



万能近似定理(Hornik, 1991):

前馈神经网络，只需具备单层隐含层和有限个神经单元，就能以任意精度拟合任意复杂度的函数。

# 神经网络的训练

## 多隐层神经网络

### 1. 前向传播

输入层

$$x = [x_1, \dots, x_d]^T$$

隐藏层 (从1至m)

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

$$h^{(1)}(x) = h^{(1)}(a^{(1)}(x))$$

$$a^{(m)}(x) = W^{(m)T}h_{m-1}(x) + b^{(m)}$$

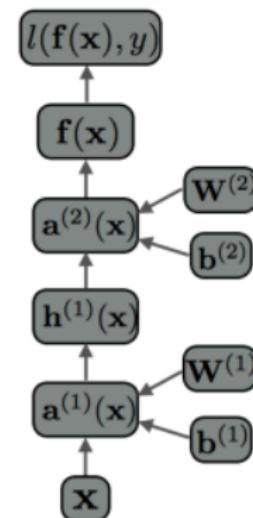
$$h^{(m)}(x) = h^{(m)}(a^{(m)}(x))$$

每一层的输出本质上是将输入特征进行了**特征变换**



输出层

$$o(x) = f(W^{(m+1)T}h^{(m)}(x) + b^{(m+1)})$$



## 2. 反向传播

从后往前依次计算和反传梯度的过程

损失函数：

训练过程等价于一个优化过程

$$l = \frac{1}{2} \sum \| \text{target} - \text{output} \|^2 \rightarrow l(f, y) = \frac{1}{2} \sum \| y - f_{W,b}(x) \|^2$$

$$f_{W,b}(x) \rightarrow f(x; \theta)$$

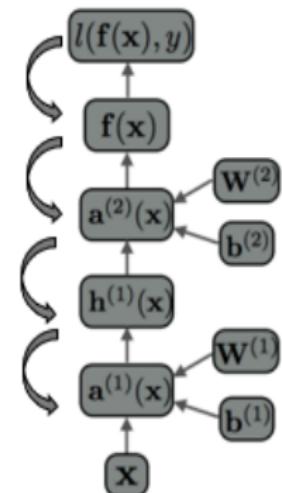
更新参数：

求梯度

$$\nabla_{W^{(1)}} l, \nabla_{b^{(1)}} l, \nabla_{W^{(2)}} l, \nabla_{b^{(2)}} l, \dots$$

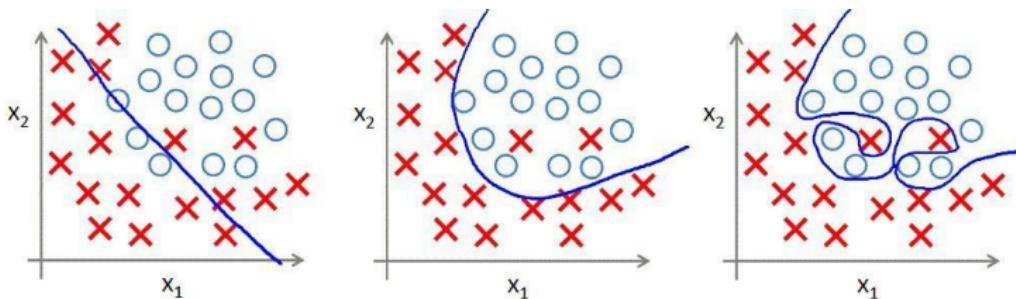
链式法则反传：

$$\frac{\partial l}{\partial W^{(2)}} = \frac{\partial l}{\partial o} \frac{\partial o}{\partial W^{(2)}}, \quad \frac{\partial l}{\partial W^{(1)}} = \frac{\partial l}{\partial o} \frac{\partial o}{\partial h} \frac{\partial h}{\partial W^{(1)}}$$



## 2. 反向传播

网络越大越深，模型复杂度越高，对函数的拟合能力越强  
 网络越大越深，参数也会越多，容易导致过拟合



正则化：减小权重的大小，防止过拟合

损失函数

$$l = \frac{1}{2} \sum \|y - f_{W,b}(x)\|^2 + \frac{\lambda}{2} \sum_m \sum_i \sum_j (W_{ji}^{(m)})^2$$

正则化项Ω

# 神经网络的训练

## 随机梯度下降

步骤1：网络参数初始化  $\theta = \{W^{(1)}, b^{(1)}, \dots, W^{(m+1)}, b^{(m+1)}\}$

步骤2：对  $t=1:T$  迭代

对每个训练样本  $(x^t, y^t)$

计算梯度  $\Delta = -\nabla_{\theta} l(f(x^t; \theta), y^t) - \lambda \nabla_{\theta} \Omega(\theta)$

$\theta \leftarrow \theta + \alpha \Delta$ ,  $\alpha$  为步长

训练一个神经网络，我们需要

- 计算损失函数  $l(f(x^t; \theta), y^t)$
- 计算损失函数  $l$  的梯度  $\nabla_{\theta} l(f(x^t; \theta), y^t)$
- 正则化项及其梯度  $\Omega(\theta), \nabla_{\theta} \Omega(\theta)$

# 神经网络的训练



## 小批量-Mini-batch

更新是基于一组**小批量的样本**  $\{(x^{(i:i+b)}, y^{(i:i+b)})\}$ 不再是单一样本)

- 梯度对应于正则化损失在小批量样本上的平均

$$l_{i:i+b}(\theta) = \frac{1}{b} \sum (f(x^{i:i+b}; \theta), y^{i:i+b}) + \lambda \Omega(\theta)$$

- 可以得到对梯度更加精确的估计
- 可以使用矩阵运算, 计算效率更高

## 批处理

更新是基于**整批样本**  $\{(x^{(i:T)}, y^{(i:T)})\}$ 不再是小批量样本)

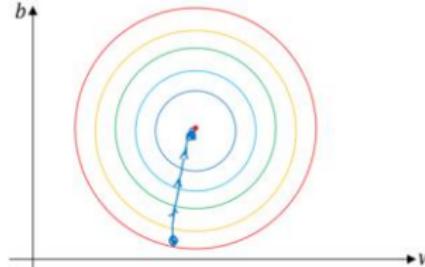
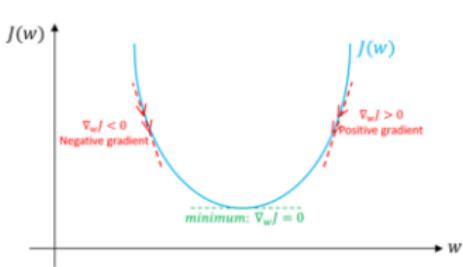
**loop**

- 梯度对应于正则化损失在整批样本上的平均

$$l_{i:i+b}(\theta) = \frac{1}{T} \sum (f(x^{i:T}; \theta), y^{i:T}) + \lambda \Omega(\theta)$$

- 计算 $\Delta\theta$ 并对 $\theta$ 做调整

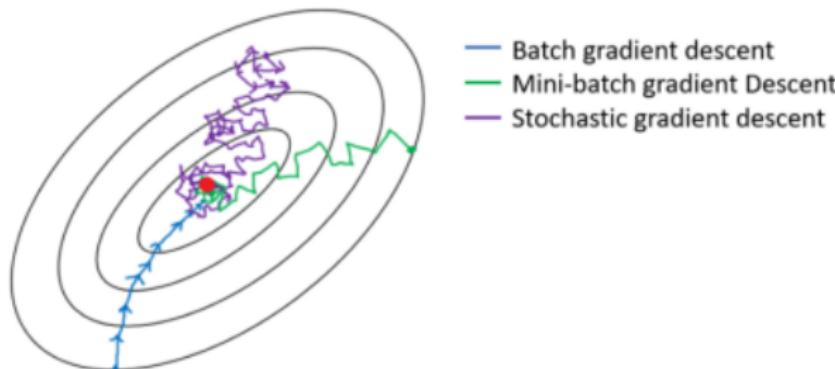
**end loop**



# 神经网络的训练



样本的选择：随机 vs 小批量处理vs批处理



如果数据集  $\{(x^t, y^t)\}$  很大 (e.g. ImageNet over 14 million), 每次批处理的损失函数和梯度计算量太大，常常采用小批量 Mini-batch 的方式训练。

# 神经网络的训练



## 模型的选择

### ■ 训练技巧

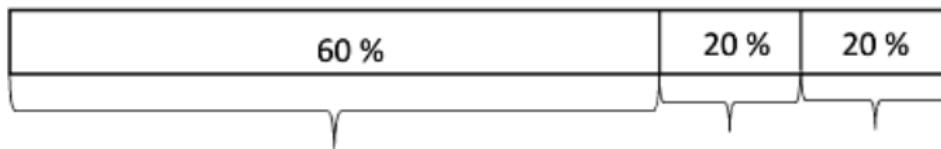
在训练集  $\mathcal{D}^{\text{train}}$  上训练你的模型

在验证集  $\mathcal{D}^{\text{valid}}$  上选择模型

包括选择超参; 隐含层尺寸; 学习率; 迭代/训练次数; 等等

在测试集  $\mathcal{D}^{\text{test}}$  上评估泛化能力

### ■ 泛化的含义是模型在未见过的样本上的表现



万级别数据集  $\mathcal{D}^{\text{train}}$

$\mathcal{D}^{\text{valid}}$   $\mathcal{D}^{\text{test}}$

# 神经网络的训练



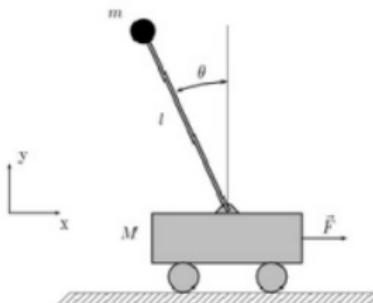
## 早停法-Early Stopping

当要确定训练次数时,可以在验证集误差开始增加时停止训练(超前判断)



## 神经网络的训练

举例：小车倒立摆问题



$$\frac{d^2\theta}{dt^2} = \frac{g \sin \theta + \cos \theta [-F - ml\dot{\theta}^2 \sin \theta + \mu_c \text{sgn}(\dot{x})] - \frac{\mu_p \dot{\theta}}{ml}}{l \left( \frac{4}{3} - \frac{m \cos^2 \theta}{m_c + m} \right)}$$

$$\frac{d^2x}{dt^2} = \frac{F + ml[\dot{\theta}^2 \sin \theta - \ddot{\theta} \cos \theta] - \mu_c \text{sgn}(\dot{x})}{m_c + m}$$

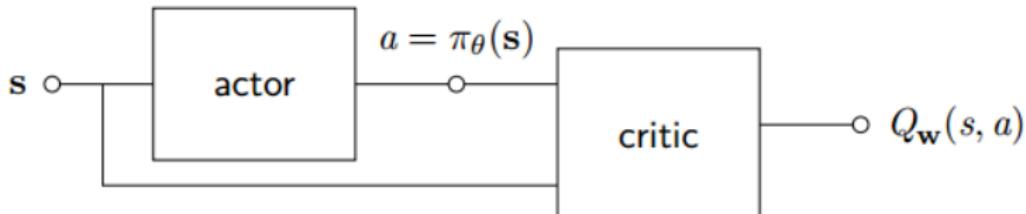
J. Si and Y.-T. Wang, On-Line Learning Control by Association and Reinforcement, IEEE TNNLS, 2001

# 神经网络的训练

举例：小车倒立摆问题

- 状态  $s = [x, \theta, \dot{x}, \dot{\theta}]^T$
- 动作  $u = \{-10, 10\}$
- 奖赏  $R(s) = \begin{cases} 0, & \text{if } |x| \leq 2.4 \text{ and } |\theta| \leq 12^\circ \\ -1, & \text{otherwise} \end{cases}$  (目标: 顶点稳定)
- 折扣因子  $\gamma = 0.95$

对 Critic NN 可以使用例如 TD 学习算法训练网络权重  
 对 Actor NN 希望能够输出最优动作使得 Q 函数最大化



## 神经网络的训练



举例：小车倒立摆问题

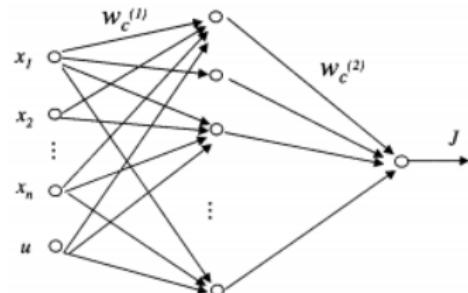
■ Critic NN 结构  $\delta_t = r_t + \gamma Q_{\mathbf{w}}(s_{t+1}, \pi_{\theta}(s_{t+1})) - Q_{\mathbf{w}}(s_t, a_t)$

三层网络: 输入层, 隐含层, 输出层

输入: 状态, 动作

隐含层结点: 6 个

输出层:  $Q(s; a)$



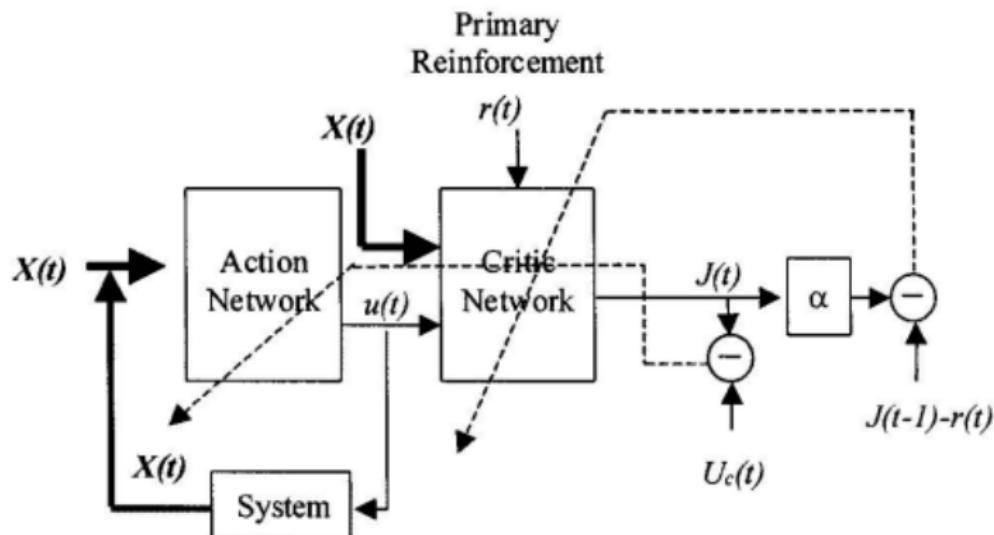
■ Actor NN 结构  $\nabla_{\theta} Q_{\mathbf{w}}(s, \pi_{\theta}(s)) = \nabla_a Q_{\mathbf{w}}(s, a) \nabla_{\theta} \pi_{\theta}(s)$

与 Critic 类似, 只不过输入层只  
包含状态

输出层:  $a=\pi(s)$ , 确定性策略

## 神经网络的训练

举例：小车倒立摆问题

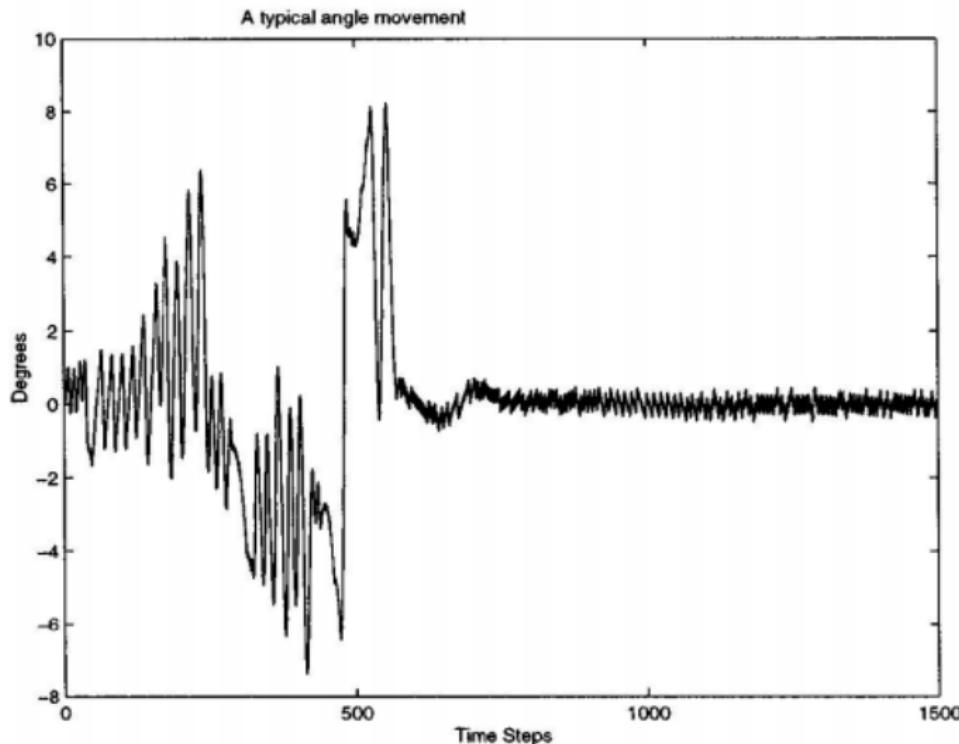


如果 Actor network 输出大于零,  $u = +10$ ; 否则  $u = -10$

## 神经网络的训练



## 实验结果



## ■ 1.1 神经网络

神经网络的定义

神经网络的训练

## ■ 1.2 卷积神经网络

卷积层

池化层

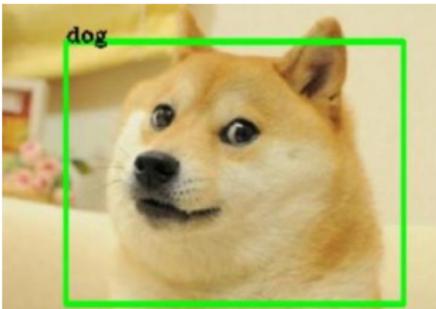
## ■ 1.3 逆强化学习

模仿学习

逆向强化学习

# 深度学习

深度学习近年来广受关注，凭借深度模型的优异性能在各领域应用广泛



图像分类任务



风格迁移



自然语言处理



图像生成



深度模型最关键的三点特性(by 周志华教授)

- 逐层传递 (Layer-by-Layer processing) 深度神经网络；
- 特征变化 (Feature transformation) 深度随机森林；  
...
- 足够的模型复杂度 (Sufficient model complexity)

深度神经网络(DNN)取得成功的一些技术

梯度弥散问题：激活函数Relu、Batch Normalization、分布式表征

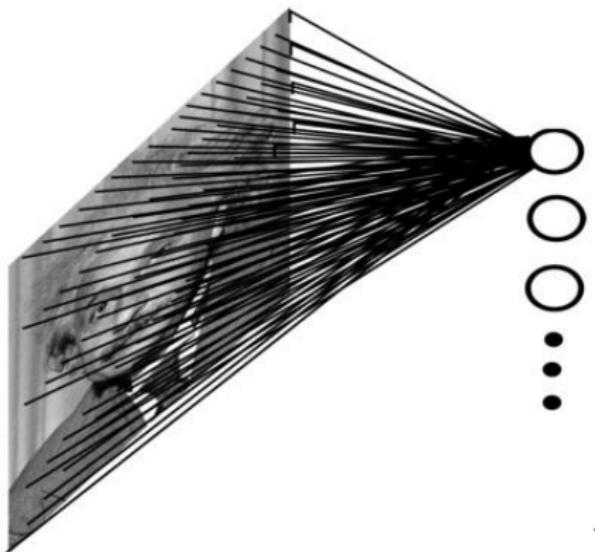
过拟合问题：正则化、预训练+fine-tuning、dropout、数据增广

# 高维输入



考虑直接以图像的像素作为输入, 构造深度神经网络

举例:  $1000 \times 1000$  像素的灰白图片



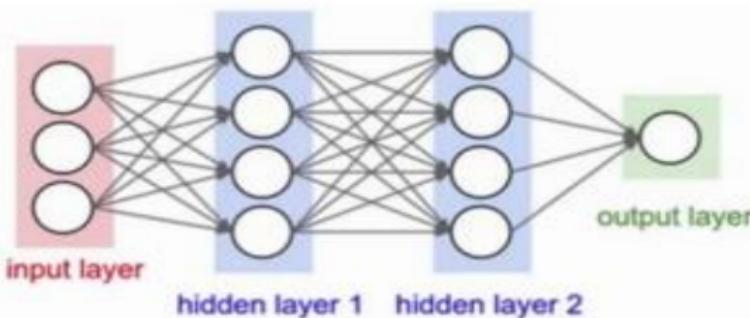
$10^6$  个输入  
一个神经元和输入层所有点连接一共有多少权重?  
 $10^6$ (输入) + 1(偏置)

隐含层有 1M 个结点的话  
一共有  
 $\approx 10^{12}$  个参数

如果网络变深, 参数爆炸增长

# 高维输入

- 传统的神经网络将输入看作是整体的向量, 然后逐层输入(全连接)到隐含层
  - 如果输入维度增加or网络变深, 参数爆炸增长
  - 计算时间/空间的复杂度
  - 忽略了图片局部结构信息



每一层都是一维向量

# 图像特征



## ■ 图片是有结构的

局部结构和相关性  
在空间和频域上有辨别性的特征

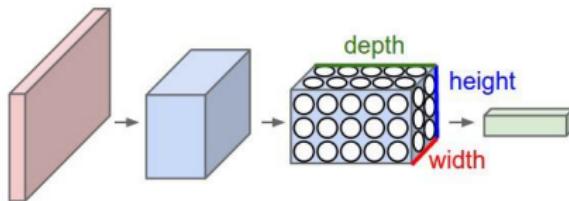
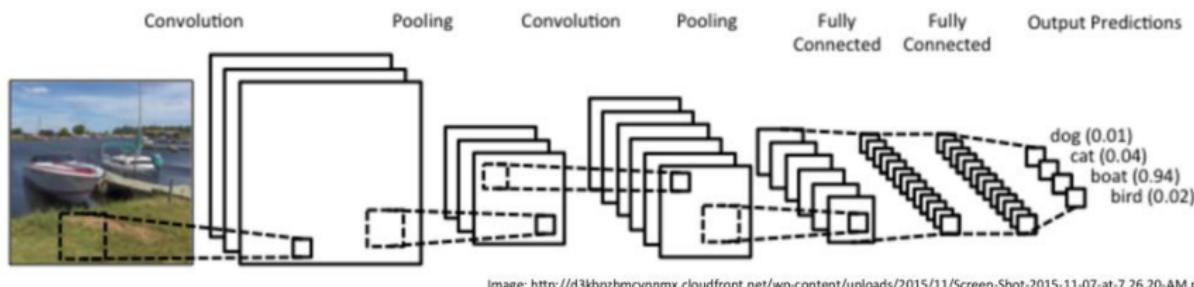
## ■ 图像特征

唯一性  
不变性  
几何不变性: 平移, 旋转, 缩放  
光度不变性: 明暗, 曝光  
在不同图片上都有明确的匹配  
找出图片中关键点

## ■ 但是找出满足这些条件的特征不容易

# 卷积神经网络

## Convolutional Neural Network, CNN



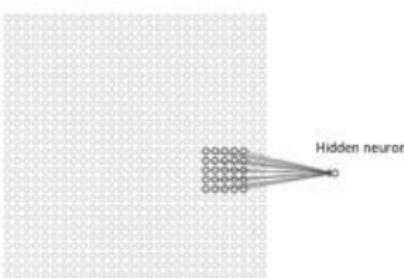
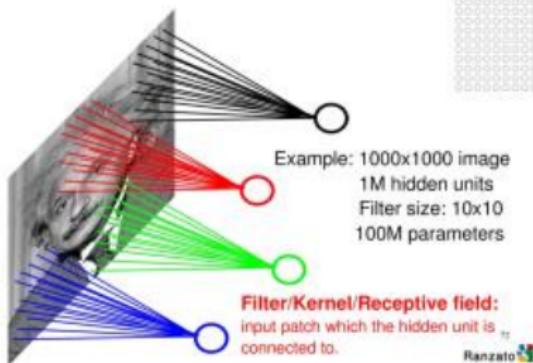
卷积神经网络的每一层是三维数据:  
高, 宽, 深

- 由宽和高组成的平面称为特征图谱(feature map)
- 深度表示特征图谱的个数, 也叫做通道(channel) 数
- 当输入是一个RGB图像时, 表示深度为3的输入

# 卷积层

- 局部连接
- 参数共享

只与部分图像块局部连接，  
进行分布式特征表示

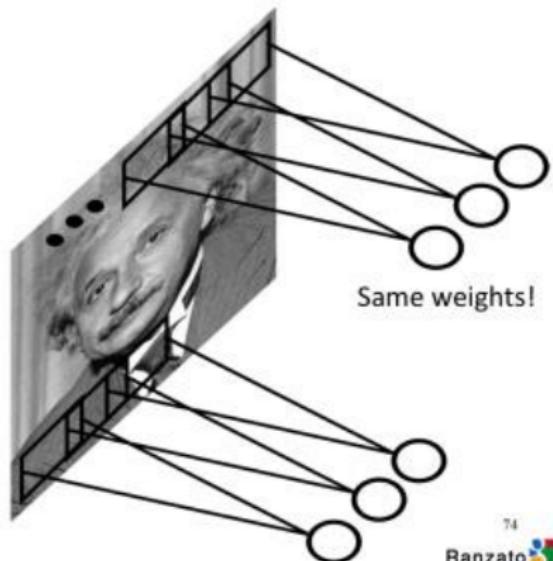


- 卷积核/感受野  
隐含层神经元对输入层的感受范围
- 如：输入  $1000 \times 1000$  的图像像素
- 隐含层是 1M 个结点
- 每个隐含层结点都使用  $10 \times 10$  感受域的卷积核
- 那么参数一共只有 100M 个

# 卷积层

- 局部连接
- 参数共享

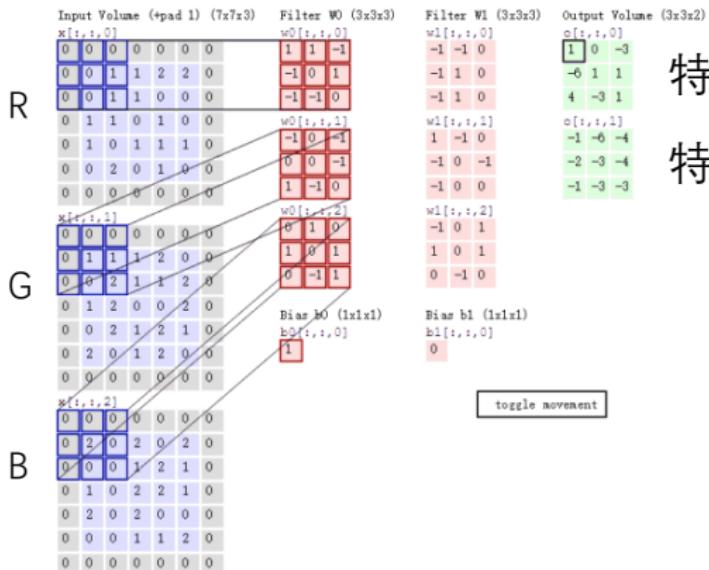
卷积核参数全局共享



- 隐含层的一些结点可以全局共享卷积核的参数
  - ◆ 同一特征图谱的隐层结点都是在检测相同的特征, 不过可以是对图像的不同区域做检测

# 卷积层

关键参数：卷积核(filter)、步长(stride)、填充(padding)



特征图谱  
特征图谱

卷积核：3\*3\*3  
个数为2

Stride=2

防止边缘特征稀疏

Padding：在输入外围填充0

## 特征图谱

### ■ 特征图谱：

- ◆ 每一组卷积核提取一种特征，通过共享同一组卷积核参数，得到一个特征图
- ◆ 多个卷积核可以得到多个特征图谱，特征图深度就是卷积核个数

### ■ 背后的原理：

- ◆ 如果一组权重和偏置（经过学习）能够使隐层神经元在某个感受野认出一个竖直的边缘
- ◆ 我们希望这种能力是可以在图片其它位置同样能发挥作用的
- ◆ 这样可以将相同的特征提取能力用在整个图片上，具有平移不变性
- ◆ 受现实人类视觉能力的启发

## 特征图谱



输出特征图尺寸计算：

输入RGB图像：32\*32\*3

卷积核：5\*5\***10**

步长：stride=1

填充：padding=2

输出特征图：32\*32\***10**

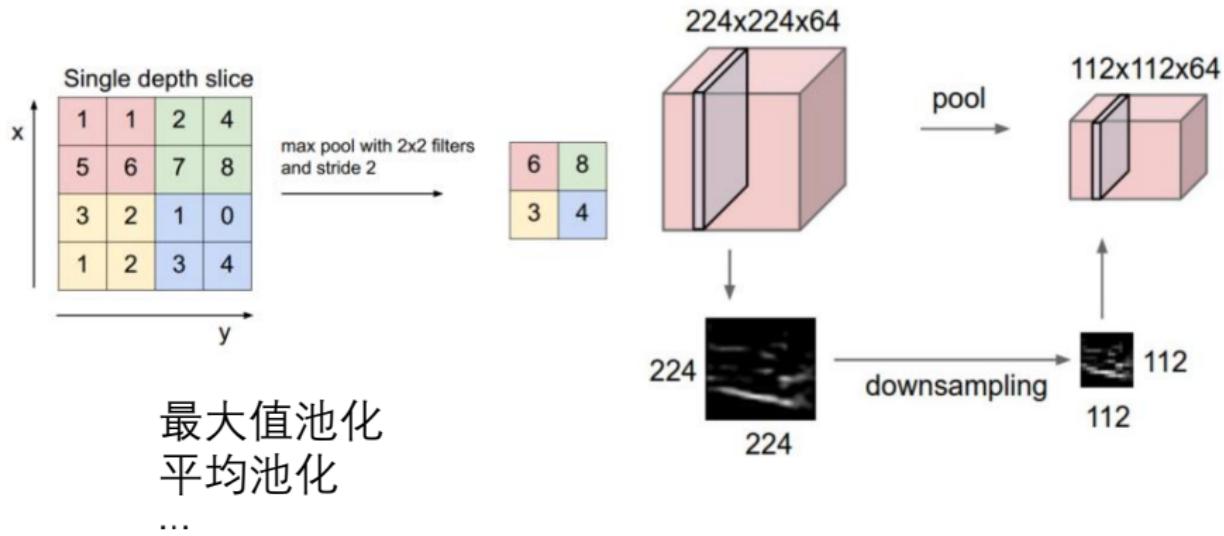
$$w_{out} = \frac{w_{in} - Filter + 2Pad}{stride} + 1$$

$$h_{out} = \frac{h_{in} - Filter + 2Pad}{stride} + 1$$

# 池化层

## Pooling layer

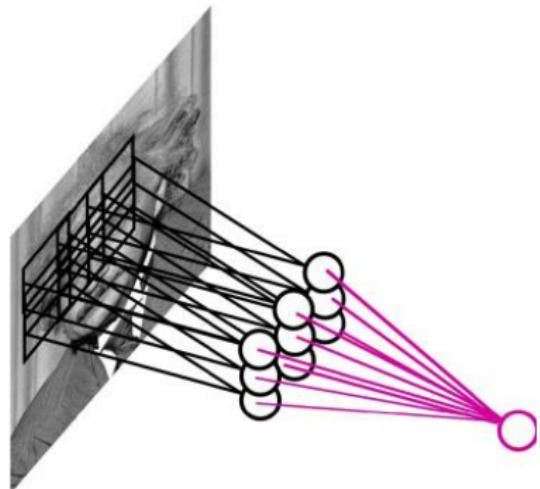
- 池化层本质上是一种降采样
- 接在卷积层后面，压缩生成的特征图大小



最大值池化  
平均池化

# 池化层

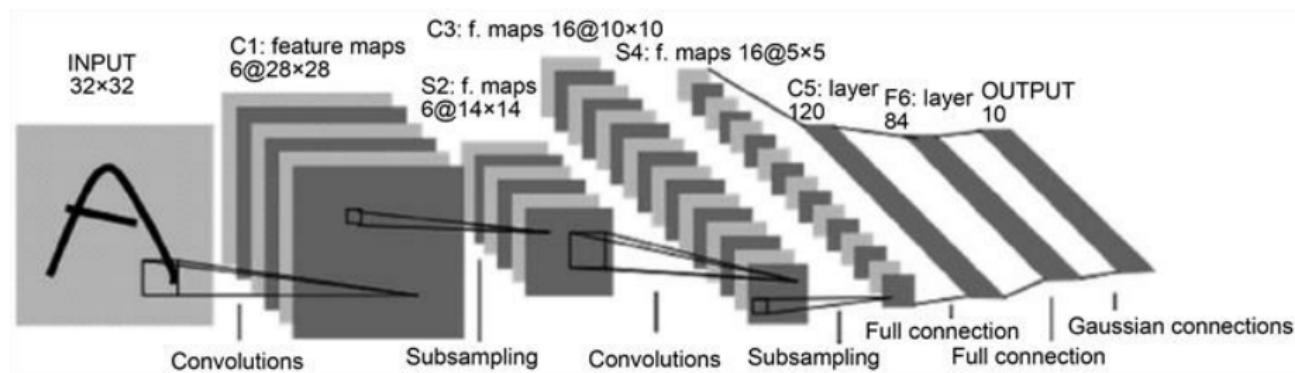
## ■ 池化的意义：



- ◆ 能够让网络知道某种特征是否在图片的某一区域内出现,而不用考虑特征具体出现在该区域的哪个位置
- ◆ 减少了隐藏层中单元数量,显著减少网络参数
- ◆ 能够提高特征检测能力特征对空间位置的鲁棒性

# 举例

## Lenet5 (Lecun 1994)



一般卷积神经网络的由卷积层和池化层交替连接，

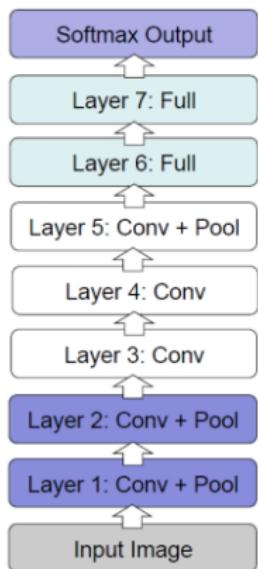
最后几层是全连接层

- 输出不同类别的概率(分类任务)
- 输出不同动作所对应的Q值(RL问题)

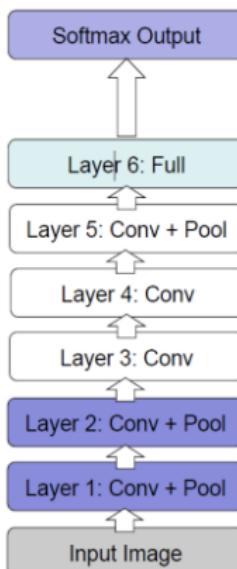
# 举例

AlexNet(Deng et al. CVPR'09)

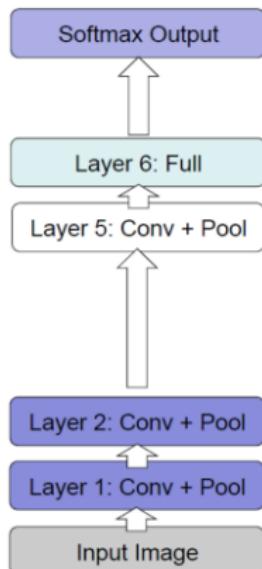
卷积层操作及网络的深度对性能影响至关重要



ImageNet top-5 error: 18.2%



19.3%



51.7%

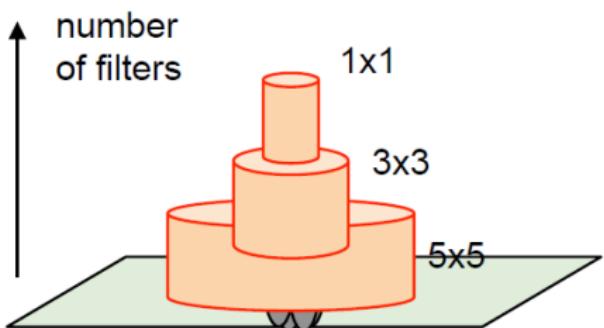
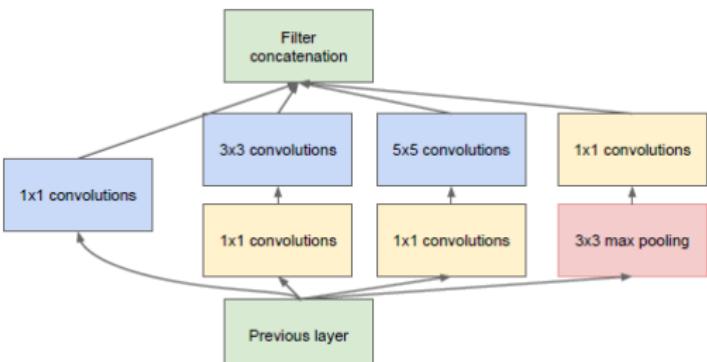
# 举例

GoogLeNet(Szegedy et al., Going Deep with Convolutions, 2014)

inception module:

每层多个卷积核尺寸

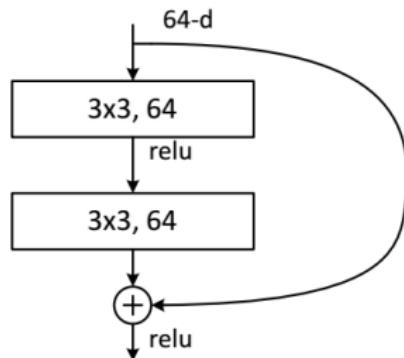
特征变化+模型复杂度



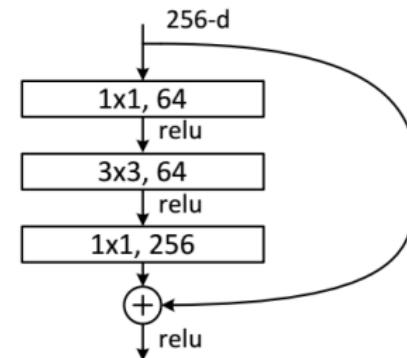
ImageNet top-5 error: 6.7%

## 举例

ResNet(He et al., CVPR 2016) 2016 CVPR best paper



ImageNet top-5 error: 4.49%



在每一层引入“pass through”

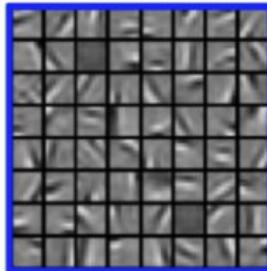
- 提出残差结构，使得超深的网络训练成为现实
- 152 层的网络结构，使得识别率超过人类

### 举例

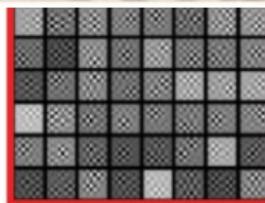
如何选择网络结构与参数?

- 任务依赖
- 交叉验证
- 多组[卷积+  
越多数据:  
可视化结果
- 1. 特征最好
- 2. 特征可展

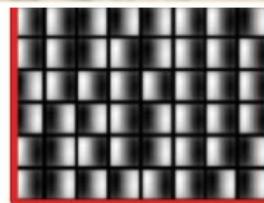
GOOD



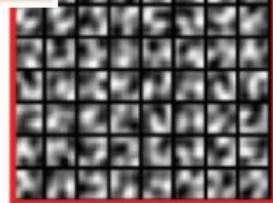
BAD



too noisy



too correlated



lack structure

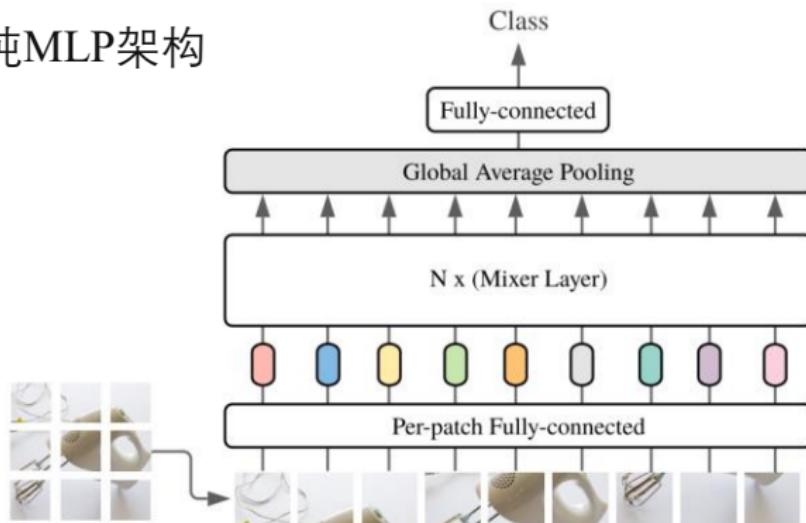
## 卷积网络的拓展

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

...

## 神奇的MLP: MLP-Mixer

纯MLP架构



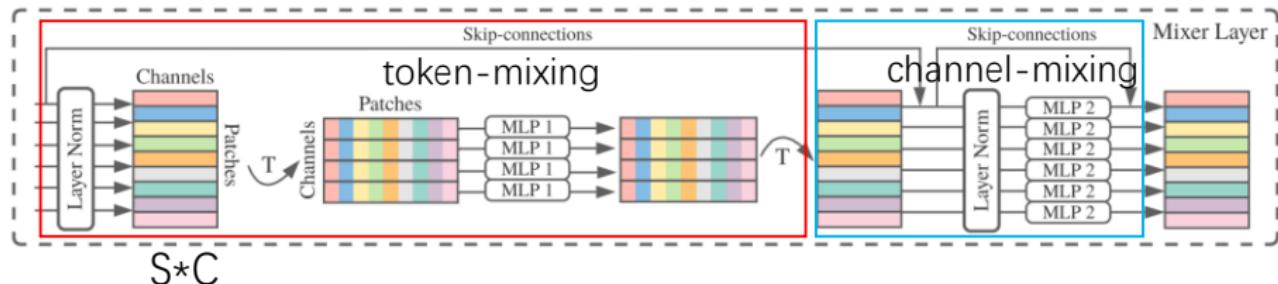
输入尺寸(H, W)

Patch尺寸(P, P)

$$\text{Patch的数量 } S = HW/P^2$$

1. 逐层传递
2. 特征变化
3. 足够的模型复杂度

## Mixer层



隐层维度 C

$$\mathbf{U}_{*,i} = \mathbf{X}_{*,i} + \mathbf{W}_2 \sigma(\mathbf{W}_1 \text{LayerNorm}(\mathbf{X})_{*,i}), \quad \text{for } i = 1 \dots C,$$

$$\mathbf{Y}_{j,*} = \mathbf{U}_{j,*} + \mathbf{W}_4 \sigma(\mathbf{W}_3 \text{LayerNorm}(\mathbf{U})_{j,*}), \quad \text{for } j = 1 \dots S.$$

- token-mixing: 对列数据进行处理, 不同空间位置的特征
- channel-mixing: 对行数据进行处理, 不同通道的特征

## 神奇的MLP: MLP-Mixer

## 实验结果

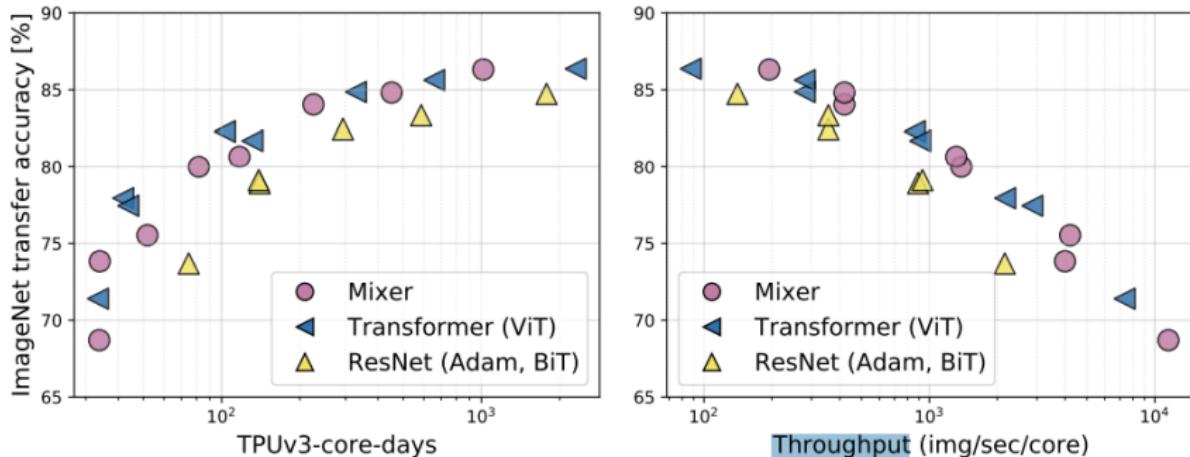


Figure 3: The role of the model scale. ImageNet validation top-1 accuracy vs. total pre-training compute (**left**) and throughput (**right**) of ViT, BiT, and Mixer models at various scales. All models are pre-trained on JFT-300M and fine-tuned at resolution 224, which is lower than in Figure 2 (left).

MLP-Mixer和ViT的训练计算量、推理速度水平基本相当，好于ResNet

MLP → CNN → Transformer → MLP

- 1.1 神经网络
  - 神经网络的定义
  - 神经网络的训练
- 1.2 卷积神经网络
  - 卷积层
  - 池化层
- 1.3 逆强化学习
  - 模仿学习
  - 逆向强化学习

强化学习：基于**奖赏信号reward**(通常是稀疏奖赏)学习策略

- 优势：简单、廉价的半监督方法
- 劣势：样本利用率低

目前强化学习的成功应用在：

- 数据易获取且并行容易的仿真环境 （游戏、仿真环境）
- 而非在
  - 动作执行较慢
  - 试错失败成本昂贵，或者无法容忍失败的情况
  - 安全性要求较高
  - 如机器人控制系统、无人车/无人机控制等

## 奖赏设计

非稀疏奖赏可以很好的指导智能体的学习

那么该如何获得这些奖赏？

- 手工设计[1]: RL中最常见的方式，显式设计、需要经验、且比较脆弱
- 不依赖奖赏: 模仿学习，利用监督学习训练策略
- 从演示中学习奖赏: 逆强化学习 (学习隐式奖赏)

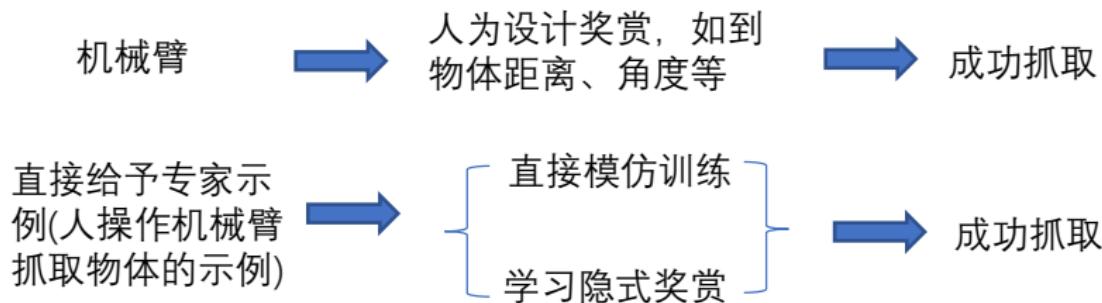


Learning from demonstration  
for autonomous navigation in  
complex unstructured terrain,  
Silver et al. 2010

[1] <https://www.bons.ai/blog/reward-functions-reinforcement-learning-video> 19/105

# 举例

## 机械臂抓取



很多时候，尤其是真实环境中在线学习非常昂贵，而采集真实样本数据却非常容易。

## 举例



### 高速场景无人驾驶

Abbeel and Ng, ICML 2004

Syed and Schapire, NIPS 2007

Majumdar et al., RSS 2017



### 停车场导航

Abbeel, Dolgov, Ng, and Thrun, IROS 2008



### 四足机器人

Ratliff, Bradley, Bagnell, and Chestnutt,  
NIPS 2007

Kolter, Abbeel, and Ng, NIPS 2008



## 适用情况

1. 专家提供一系列**演示轨迹**: 状态-动作序列
2. 当专家可以很容易地演示, 同时演示轨迹很容易收集的时候, 逆强化学习将会非常适用

1. 当得到期望策略的奖赏很容易设计或很容易得到
2. 期望策略很容易直接得到

目标：智能体需要找到**期望策略**，使得该策略下状态-动作轨迹分布尽可能匹配专家演示样本的轨迹分布

输入：

- 状态空间、动作空间
- 状态转移概率模型  $P(s'|s, a)$
- 没有奖赏函数  $R$
- 专家演示样本集  $(s_0, a_0, s_1, a_1, \dots)$

**直接**：利用监督学习获得期望策略（状态到专家动作的映射）  
输入为演示样本集中的状态，标签为演示样本集中的动作

## 模仿学习



- 行为克隆 (Behavior Cloning)
- 样本增广 (Demonstration augmentation)
- 数据聚合 (DAGGER, Dataset Aggregation)

模仿学习的核心均是监督学习，不同在于对专家演示样本集的增广和提升

## 标准的监督学习任务

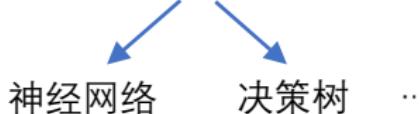
基于专家演示样本集直接学习状态到动作的映射

假设专家轨迹中的**样本是i.i.d.**，与绝大多数深度学习方法一致

专家演示样本集  $\{(s_t, a_t) | t \in \mathcal{T}\}$

在专家演示样本集训练得到**期望策略** $\hat{\pi}$ 最小化误差

$$\sum_{t \in \mathcal{T}'} (\hat{\pi}(s_t) - a_t)^2$$



$s_t$



$a_t$



样本集



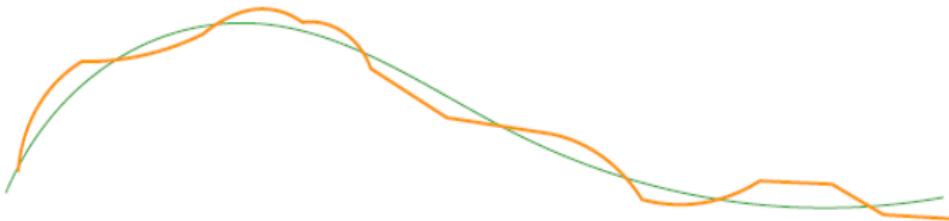
行为  
克隆



$\hat{\pi}(a_t | s_t)$

行为克隆的问题：复合误差 (compounding errors)

假设样本是i.i.d



Error 时刻 $t$ 下的误差概率  $\epsilon$

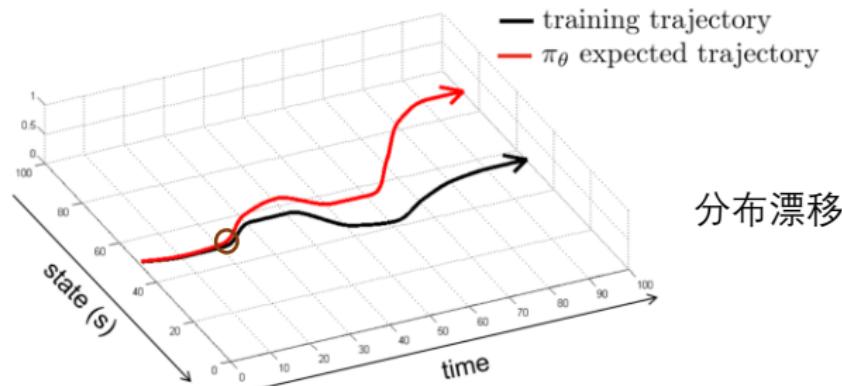
$\mathbb{E}[\text{总误差}] \leq \epsilon T$

## 行为克隆

行为克隆的问题：复合误差 (compounding errors)

MDPs: 状态分布依赖于执行的动作

下一时刻的状态依赖于当前时刻的动作



Error 时刻 $t$ 下的误差概率  $\epsilon$

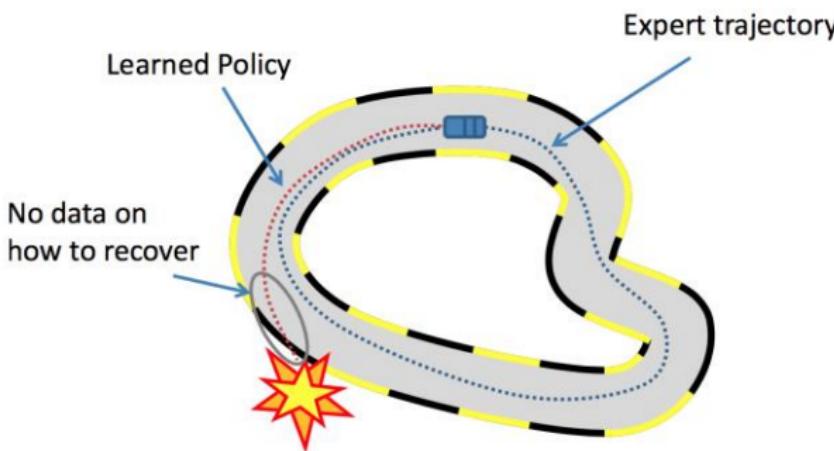
$$\mathbb{E}[\text{总误差}] \leq \epsilon(T + (T - 1) + (T - 2) \dots + 1) \propto \epsilon T^2$$

## 行为克隆的问题：复合误差

监督学习成功的基本假设：训练集和测试集的分布一致

MDPs：训练集/专家样本演示集  $s_t \sim D_E$

测试集  $s_t \sim D_{\pi_\theta}$



- 同一驾驶场景下专家动作可能不同
- 遇到未见过的测试样本性能变差

两个成功例子：

1. Dean Pomerleau's ALVINN system, NIPS 1989

利用人类驾驶数据学习无人驾驶小车的横向控制行为，首次成功实现基于视觉和雷达输入的神经网络控制器下的长距离无人驾驶任务

2. Claude Sammut et al., Learning to fly, ICML 1992

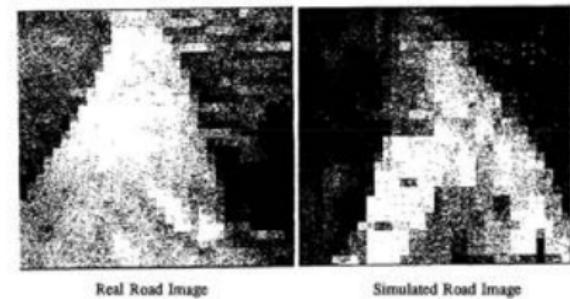
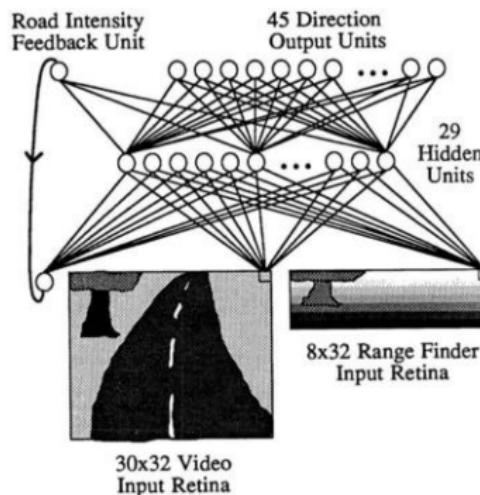
利用飞行仿真器下的人类飞控数据学习到行为决策树，并在该飞行仿真器中成功运行

## 样本增广

## 累积误差的解决办法：数据增广

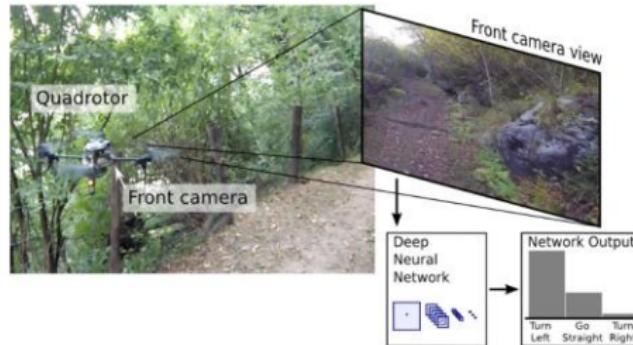
增加训练集中的样本轨迹，尽可能覆盖足够广的状态空间

ALVINN, 1989



利用仿真器产生虚拟的道路图像样本  
在线自适应生成人类驾驶员转角

## 累积误差的解决办法：数据增广



Giusti et al. A Machine Learning Approach to Visual Perception of Forest Trails for Mobile Robots, 2015

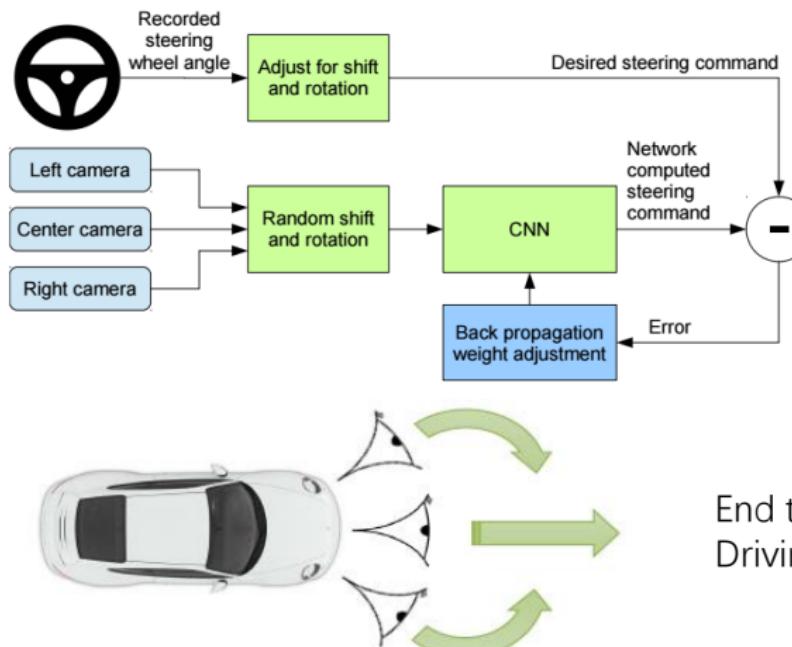


Fig. 4: Left: stylized top view of the acquisition setup; Right: our hiker during an acquisition, equipped with the three head-mounted cameras.

## 样本增广

## 累计误差的解决办法：数据增广

NVIDIA 2016



输入端增加了左侧和右侧的图像输入来实现数据增广，缓解误差的积累，使得策略可以从错误中纠正

End to End Learning for Self-Driving Cars , Bojarski et al. 2016

NVIDIA 2016

BB8 demo

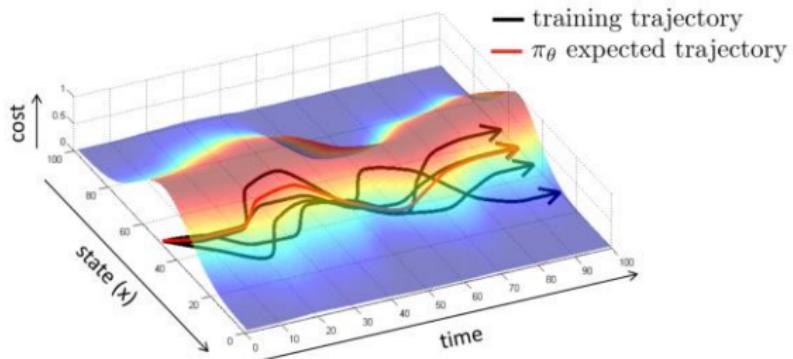


## 样本增广

## 累计误差的解决办法：数据增广

NVIDIA 2016

由专家轨迹到轨迹分布



stability

人为引入噪声情况下的样本轨迹，教会网络从错误状态中纠正自己的误差

# 数据聚合DAGGER

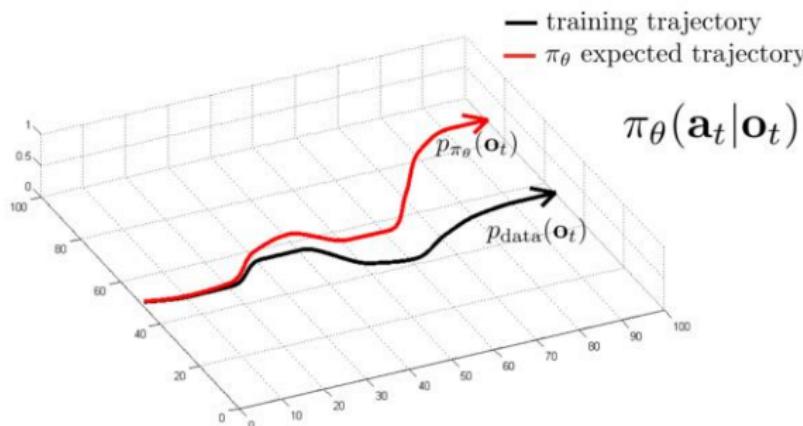
$$p_{data}(s_t) \neq p_{\pi_\theta}(s_t)?$$



专家演示训练集

智能体策略 $\pi_\theta$ 测试集

完全可观环境  $s_t = o_t$



是否可以让专家轨迹分布与智能体轨迹分布尽可能接近?

## 数据聚合DAGGER

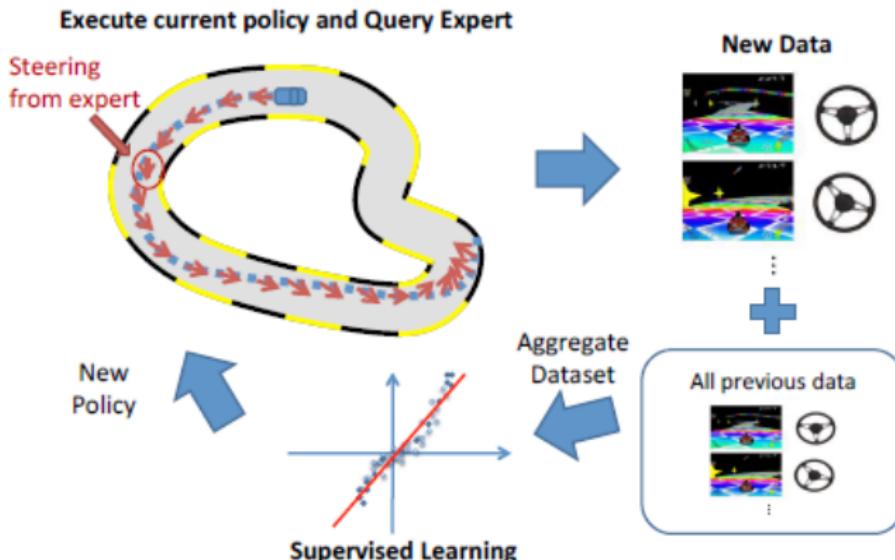


如何确保  $p_{data}(s_t) \approx p_{\pi_\theta}(s_t)$ ?      既然无法直接改变  $\pi_\theta$

通过重新标记当前策略下的新样本标签，添加至  $p_{data}$   
确保专家轨迹分布与智能体轨迹分布尽可能接近

1. 训练  $\pi_\theta(a_t|s_t)$  基于专家演示数据集  $\{(s_t, a_t) | t \in \mathcal{T}\}$
2. 执行  $\pi_\theta(a_t|s_t)$  得到新轨迹  $\mathcal{T}_\pi = \{s_1, s_2, \dots, s_M\}$
3. 专家再次标记新轨迹  $\mathcal{T}_\pi$  的标签  $\{a_1, a_2, \dots, a_M\}$
4. 数据聚合:  $\mathcal{T} \leftarrow \mathcal{T} \cup \mathcal{T}_\pi$
5. 执行步骤1, 循环

# 数据聚合DAGGER



问题：

1. 一直在执行不安全/部分训练的策略
2. 需要大量专家资源重新标记样本，同时与任务相关

# 数据聚合DAGGER

举例：基于RGB图像预测无人机转向角



Learning monocular reactive UAV control in cluttered natural environments,  
Ross et al. 2013

# 模仿学习的瓶颈

模仿学习经常面临：

- 数据分配不匹配
- 专家非马尔可夫行为
- 专家多模态行为

一些提升的办法：

1. 增加左右侧输入图像数据
2. 从一个稳定的轨迹分布中采样（用更稳定的控制策略采集样本）
3. 增加on-policy的数据（Dagger）
4. 改进模型以提升精度（RNN/LSTM等）

# 模仿学习的瓶颈

如果有足够多的数据集，模型学习是否足够好用？

但是多少数据算足够多呢？Waymo表明3千万样本模型性能仍然不够

## Abstract

Our goal is to train a policy for autonomous driving via imitation learning that is robust enough to drive a real vehicle. We find that standard behavior cloning is insufficient for handling complex driving scenarios, even when we leverage a perception system for preprocessing the input and a controller for executing the output on the car: 30 million examples are still not enough. We propose exposing the learner to synthesized data in the form of perturbations to the expert's driving, which creates interesting situations such as collisions and/or going off the road. Rather than purely imitating all data, we augment the imitation loss with additional losses that penalize undesirable events and encourage progress – the perturbations then provide an important signal for these losses and lead to robustness of the learned model. We show that the *ChauffeurNet* model can handle complex situations in simulation, and present ablation experiments that emphasize the importance of each of our proposed changes and show that the model is responding to the appropriate causal factors. Finally, we demonstrate the model driving a car in the real world.

**Keywords:** Deep Learning, Mid-to-mid Driving, Learning to Drive, Trajectory Prediction.

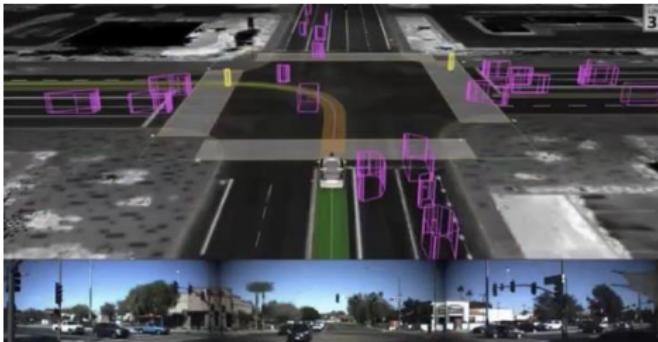
ChauffeurNet: Learning to Drive by Imitating the Best and Synthesizing the Worst, 2018, Waymo

## 模仿学习的瓶颈

美国兰德公司研究报告：

需要驾驶多少里程才能展示自动驾驶车辆的可靠性？

自动驾驶汽车需要行驶**数百亿英里**才能验证其可靠性



收集**100亿英里**的数据：

- 2000辆测试车辆（平均每辆100万元）
- 每天以平均40英里每小时的速度行驶8个小时，
- **需时40多年**

Waymo至今只累计了2000万里程

## 逆强化学习

间接：从专家演示样本中学习奖赏函数

- 状态空间、动作空间
- 状态转移概率模型  $P(s'|s, a)$
- 没有奖赏函数  $R$
- 专家演示样本集  $(s_0, a_0, s_1, a_1, \dots)$

IRL：假设专家策略  $\pi_E$  是最优的，从专家演示样本中恢复出奖赏函数  $R^*$

$$\mathbb{E}[\sum_{t=0}^{\infty} \gamma^t R^*(s_t) | \pi_E] \geq \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t R^*(s_t) | \pi] \quad \forall \pi \quad \text{或者} \quad V^{\pi_E} \geq V^{\pi}, \forall \pi$$

1. 专家策略往往不是精确最优
2. 如何评估学到奖赏的好坏？

## 强化学习 (RL)

给定：

- 状态空间、动作空间
- 状态转移概率模型  $P(s'|s, a)$
- 奖赏函数  $R$

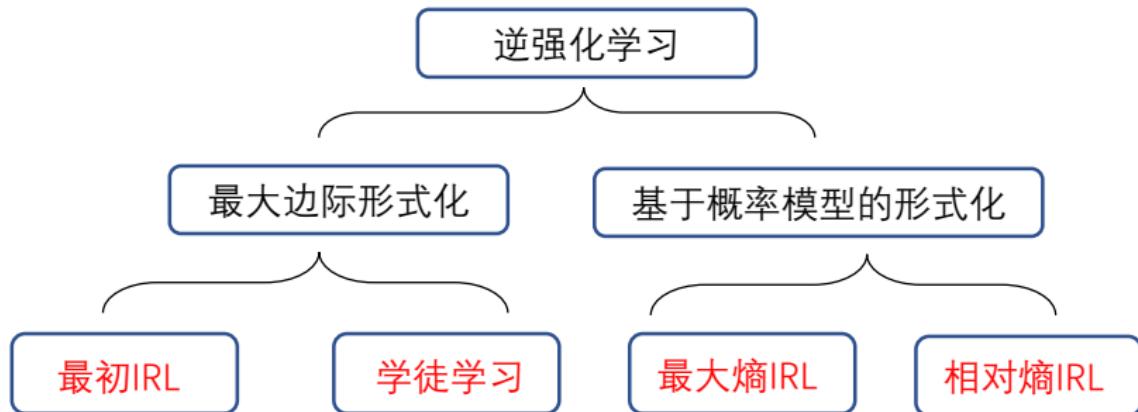
学习最优策略  $\pi^*(a|s)$ 

## 逆强化学习 (IRL)

给定：

- 状态空间、动作空间
- 状态转移概率模型  $P(s'|s, a)$
- 专家策略  $\pi_E$  的样本集  $(s_0, a_0, s_1, a_1, \dots)$

学习奖赏函数  $R^*$ 



- 对抗模仿学习 (Adversarial Imitation learning)

<https://github.com/MatthewJA/Inverse-Reinforcement-Learning>

# 最初的IRL

- 状态转移概率模型  $P(s'|s, a)$ 已知
- 状态-动作空间为离散
- 假设专家策略 $\pi_E$ 是最优的，从专家演示样本中恢复出奖赏函数 $R^*$

*Bellman*方程

$$V^\pi(s) = R(s) + \gamma \sum_{s'} P_{s\pi(s)}(s') V^\pi(s')$$

$$Q^\pi(s, a) = R(s) + \gamma \sum_{s'} P_{sa}(s') V^\pi(s')$$

*Bellman*最优性

$$\pi(s) \in \arg \max_{a \in A} Q^\pi(s, a)$$

# 最初的IRL

对于  $\pi(s) \equiv a_1$ ，基于Bellman方程：

$$\mathbf{V}^\pi = \mathbf{R} + \gamma \mathbf{P}_{a_1} \mathbf{V}^\pi \quad \rightarrow \quad \mathbf{V}^\pi = (\mathbf{I} - \gamma \mathbf{P}_{a_1})^{-1} \mathbf{R}$$

根据Bellman最优性，可得

$$a_1 \equiv \pi(s) \in \arg \max_{a \in A} \sum_{s'} P_{sa}(s') V^\pi(s') \quad \forall s \in S$$

$$\Leftrightarrow \sum_{s'} P_{sa_1}(s') V^\pi(s') \geq \sum_{s'} P_{sa}(s') V^\pi(s') \quad \forall s \in S, a \in A$$

$$\Leftrightarrow \mathbf{P}_{a_1} \mathbf{V}^\pi \succeq \mathbf{P}_a \mathbf{V}^\pi \quad \forall a \in A \setminus a_1$$

$$\Leftrightarrow \mathbf{P}_{a_1} (\mathbf{I} - \gamma \mathbf{P}_{a_1})^{-1} \mathbf{R} \succeq \mathbf{P}_a (\mathbf{I} - \gamma \mathbf{P}_{a_1})^{-1} \mathbf{R} \quad \forall a \in A \setminus a_1$$

## 有限维离散状态空间

**Theorem 3** Let a finite state space  $S$ , a set of actions  $A = \{a_1, \dots, a_k\}$ , transition probability matrices  $\{\mathbf{P}_a\}$ , and a discount factor  $\gamma \in (0, 1)$  be given. Then the policy  $\pi$  given by  $\pi(s) \equiv a_1$  is optimal if and only if, for all  $a = a_2, \dots, a_k$ , the reward  $\mathbf{R}$  satisfies

$$(\mathbf{P}_{a_1} - \mathbf{P}_a)(\mathbf{I} - \gamma \mathbf{P}_{a_1})^{-1} \mathbf{R} \succeq 0$$

$a_1$ 取自被认为是最优的专家策略 $\pi$ .

问题：

1. 如果 $\mathbf{R}=0$  或常数，那么方程自动满足；
2. 如果求解出来多个 $\mathbf{R}$ 的形式，如何判断哪个更好？

引用另一个约束条件：

对任何状态 $s$ , 让最优动作(专家动作)  $a_1$ 对应的 $Q(s, a_1)$ 都应该比  $Q(s, a)$  尽可能大, 即最大化

$$\text{maximize} \quad \sum_{s \in S} \left( Q^\pi(s, a_1) - \max_{a \in A \setminus a_1} Q^\pi(s, a) \right)$$

让专家策略  
相比与次好  
策略的优势  
尽可能大

1. 需要对 $R$ 进行限制, 即规定一个  $R_{\max}$ , 否则只需要把所有状态的 $R$ 等倍放大就可以增大专家的和非专家的策略区分度;
2. 认为 $R$ 越小越简单, 因此加入了L1正则项约束

# 最初的IRL

## 有限维离散状态空间IRL对应的优化问题

$$\begin{aligned}
 & \text{maximize} \quad \sum_{i=1}^N \min_{a \in \{a_2, \dots, a_k\}} \{ (\mathbf{P}_{a_1}(i) - \mathbf{P}_a(i)) \\
 & \quad (\mathbf{I} - \gamma \mathbf{P}_{a_1})^{-1} \mathbf{R} \} - \lambda \|\mathbf{R}\|_1 \\
 \text{s.t.} \quad & (\mathbf{P}_{a_1} - \mathbf{P}_a) (\mathbf{I} - \gamma \mathbf{P}_{a_1})^{-1} \mathbf{R} \succeq 0 \\
 & \forall a \in A \setminus a_1 \\
 & |\mathbf{R}_i| \leq R_{\max}, \quad i = 1, \dots, N
 \end{aligned}$$

让专家策略  
相比与次好  
策略的优势  
尽可能大

对于有限维离散状态-动作空间情况，可以利用线性规划求解 $R$

对于连续状态空间的情况，难以直接求解，需要逼近求解 $R$

## 线性特征组合

- 奖赏函数  $R$  通过状态特征的线性函数逼近

$$R(s) = w^T \phi(s) \quad w \in \mathbb{R}^n, \phi: S \rightarrow \mathbb{R}^n$$

$\phi(s)$  为定义的基函数，可以为多项式基底/傅立叶基底等

- 目标：基于专家演示样本训练权重  $w$

从专家演示样本中恢复出奖赏函数  $R^*$ ，使得专家策略  $\pi_E$  为最优策略  $\pi^*$

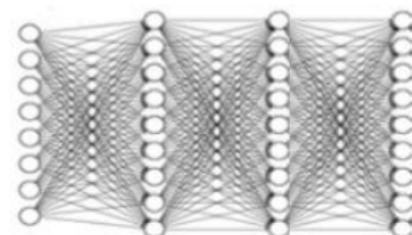
$$\mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R^*(s_t) | \pi_E\right] \geq \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R^*(s_t) | \pi\right] \quad \forall \pi$$

值函数是若干特征  
期望的线性组合

$$\begin{aligned}\mathbb{E} \left[ \sum_{t=0}^{\infty} \gamma^t R(s_t) | \pi \right] &= \mathbb{E} \left[ \sum_{t=0}^{\infty} \gamma^t w^T \phi(s_t) | \pi \right] \\ &= w^T \mathbb{E} \left[ \sum_{t=0}^{\infty} \gamma^t \phi(s_t) | \pi \right] \\ &= w^T \mu(\pi)\end{aligned}$$

策略 $\pi$ 下的特征期望向量

$s_t, a_t \rightarrow$  参数 $w \rightarrow R_w$



令  $R(s) = w^T \phi(s)$  其中  $w \in \mathbb{R}^n$ ,  $\phi: S \rightarrow \mathbb{R}^n$

得到权重  $w^*$  确保  $w^{*T} \mu(\pi^*) \geq w^{*T} \mu(\pi) \quad \forall \pi$

找到奖赏函数的参数使得专家策略的值函数优于其他策略

注意:  $R(s_t)$  是指状态  $s_t$  下对应的  $a_t$  的奖赏, 也可以理解为上  $R(s_t, a_t)$

当给定m条专家轨迹后，可以估计专家策略的特征期望为：

$$\mu(\pi^*) = \frac{1}{m} \sum_{i=1}^m \sum_{t=0}^{\infty} \gamma^t \phi(s_t^{(i)})$$

找到一个策略，使其表现与专家策略相近，其实就是找到一个策略的特征期望与专家策略的特征期望相近，即：

$$\|\mu(\pi^*) - \mu(\pi)\| \leq \epsilon$$

当该不等式成立时，对于任意的权重  $\|w\|_2 \leq 1$ ，值函数满足如下不等式

$$|w^*{}^T \mu(\pi^*) - w^*{}^T \mu(\pi)| \leq \epsilon$$

$$\begin{aligned} \|\mu(\pi^*) - \mu(\pi)\| &\leq \epsilon \\ \|w\|_2 &\leq 1 \end{aligned}$$



$$|w^{*T} \mu(\pi^*) - w^{*T} \mu(\pi)| \leq \epsilon$$

$$\begin{aligned} & |E \left[ \sum_{t=0}^{\infty} \gamma^t R(s_t) | \pi_E \right] - E \left[ \sum_{t=0}^{\infty} \gamma^t R(s_t) | \tilde{\pi} \right]| \\ &= |w^T \mu(\tilde{\pi}) - w^T \mu_E| \\ &\leq \|w\|_2 \|\mu(\tilde{\pi}) - \mu_E\|_2 \\ &\leq 1 \cdot \epsilon = \epsilon \end{aligned}$$

## 优化目标

$$t^{(i)} = \max_{w: \|w\|_2 \leq 1} \min_{j \in \{0 \cdots (i-1)\}} w^T (\mu_E - \mu^{(j)})$$

写成标准的优化形式为

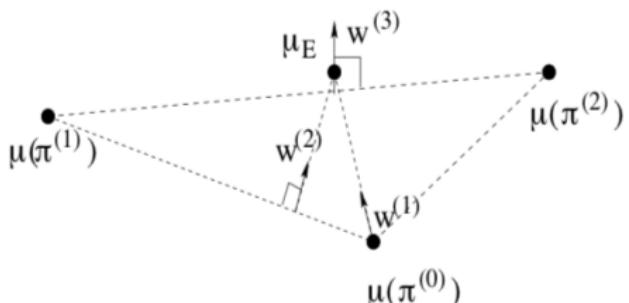
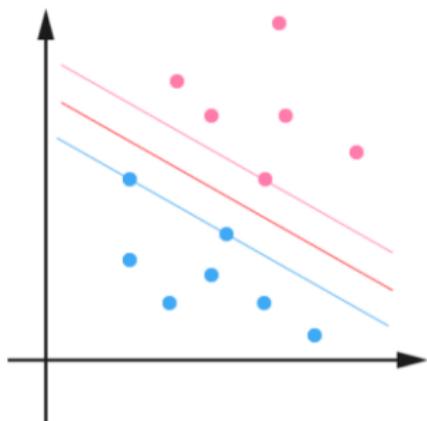
$$\begin{aligned} & \max_{t,w} t \\ s.t. \quad & w^T \mu_E \geq w^T \mu^{(j)} + t, \quad j = 0, \dots, i-1 \\ & \|w\|_2 \leq 1 \end{aligned}$$

让专家策略  
相比与其他  
策略的优势  
尽可能大

注意，在进行第二行求解时， $\mu^{(j)}$  中的  $j = 0, \dots, i-1$  是前  $i-1$  次迭代得到的最优策略。

第 $i$ 次求解参数 $w$ 时， $i-1$ 次迭代的策略是已知的。这时候的最优函数值 $t$ 相当于专家策略 $\pi_E$ 与 $i-1$ 个迭代策略之间的最大边际。

理解参考SVM中的最大边际分类器解释。



逆强化学习求奖  
赏函数参数w

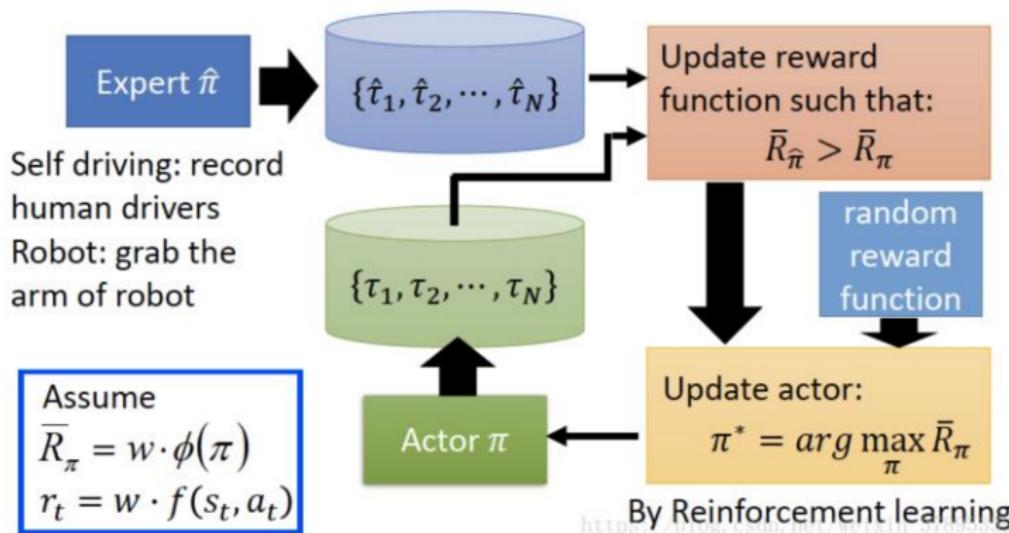
基于奖赏函数求  
最优策略

1. Randomly pick some policy  $\pi^{(0)}$ , compute (or approximate via Monte Carlo)  $\mu^{(0)} = \mu(\pi^{(0)})$ , and set  $i = 1$ .
2. Compute  $t^{(i)} = \max_{w: \|w\|_2 \leq 1} \min_{j \in \{0..(i-1)\}} w^T (\mu_E - \mu^{(j)})$ , and let  $w^{(i)}$  be the value of  $w$  that attains this maximum.
3. If  $t^{(i)} \leq \epsilon$ , then terminate.
4. Using the RL algorithm, compute the optimal policy  $\pi^{(i)}$  for the MDP using rewards  $R = (w^{(i)})^T \phi$ .
5. Compute (or estimate)  $\mu^{(i)} = \mu(\pi^{(i)})$ .
6. Set  $i = i + 1$ , and go back to step 2.

Abbeel P, Ng A Y. Apprenticeship learning via inverse reinforcement learning,  
ICML. ACM, 2004.

# Framework of IRL

$$\mathbb{E}[\sum_{t=0}^{\infty} \gamma^t R^*(s_t) | \hat{\pi}] \geq \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t R^*(s_t) | \pi] \quad \forall \pi$$



1. 在已经迭代得到的最优策略中，利用最大边际方法求出当前的回报函数的参数值；
2. 将求出的回报函数作为当前系统的回报函数，并利用强化学习方法求出此时的最优策略。
3. 可以基于不同的专家演示数据学习到不同的风格

### 最大边际方法中的“歧义”问题

1. 一个最优策略可能对应无穷个奖赏函数
2. 有无穷多个随机策略可以匹配特征数
3. 那么应该选择哪一个呢？存在随机的偏好

从概率模型的角度分析：

在概率论中，熵是不确定性的度量。不确定性越大，熵越大。

在学习概率模型时，在所有满足约束的概率模型（分布）中，熵最大的模型是最好的模型。

最大熵方法避免了歧义问题，熵最大意味着概率分布越近似均匀分布

Ziebart B D, Mass A, Bagnell J A, et al. Maximum entropy inverse reinforcement learning. AAAI Press, 2008.

$$\mathbb{E} \left[ \sum_{t=0}^{\infty} \gamma^t R(s_t) | \pi \right] = w^T \mu(\pi) \quad \mu(\pi^*) = \frac{1}{m} \sum_{i=1}^m \sum_{t=0}^{\infty} \gamma^t \phi(s_t^{(i)})$$

假设存在一个概率分布  $P(\tau)$ , 在该概率分布下, 产生了专家轨迹。

$$\sum_{\tau} P(\tau) \mu_{\tau} = \mu_{\text{dem}}$$

$$\sum_{\tau} P(\tau) = 1$$

$$\tau = \{s_1, a_1, \dots, s_T\}$$

$\mu_{\tau}$  为轨迹  $\tau$  上的特征期望

$$D_E: \{\tau_i\} \sim \pi^*$$

$\mu_{\text{dem}}$  为演示数据集的专家特征期望

在满足上述约束条件的所有概率分布中, 熵最大的概率分布是除此约束外对其他未知情况不做任何主观假设的分布, 即

$$\max_P - \sum_{\tau} P(\tau) \log P(\tau)$$

## 最大熵优化问题

优化目标

$$\max_P - \sum_{\tau} P(\tau) \log P(\tau)$$

示例数据特征  
约束条件

$$\sum_{\tau} P(\tau) \mu_{\tau} = \mu_{\text{dem}}$$

$$\sum_{\tau} P(\tau) = 1$$

针对线性奖赏函数的情况，该优化问题等价于指定权重  $w$  使得策略在匹配专家特征期望的同时，**确保熵值最大化，消除歧义问题**

## 最大熵原理

在满足示例数据特征约束条件下，使示例数据的分布熵最大化，意味着**最大化示例数据关于回报函数参数的对数似然值**

$$P(\tau_i|w) = \frac{1}{Z(w)} e^{w^T \mu_{\tau_i}} = \frac{1}{Z(w)} e^{(R_w(\tau_i))} \quad \text{拉格朗日乘子法+求微分}$$

$$Z(w) = \sum_{\tau_i} e^{w^T \mu_{\tau_i}} \quad / \quad Z(w) = \int e^{(R_w(\tau))} d\tau \quad \text{配分函数项}$$

策略轨迹的回报越大，表示该轨迹被从专家数据集中采样的机率越大

轨迹选择实际上取决于未知的奖赏信号

# 最大熵IRL

为了得到奖赏函数，选择参数 $w$ 最大化示例数据的对数似然值

$$w^* = \arg \max_w L(w) = \arg \max_w \sum_{\tau \in D_E} \log P(\tau|w)$$

$$L(w) = \sum_{\tau \in D_E} \log \frac{1}{Z} e^{(R_w(\tau))} = \sum_{\tau \in D_E} R_w(\tau) - M \log Z$$

$$= \sum_{\tau \in D_E} R_w(\tau) - M \log \sum_{\tau} e^{(R_w(\tau))}$$

$$\nabla L(w) = \sum_{\tau \in D_E} \frac{dR_w(\tau)}{dw} - M \frac{1}{\sum_{\tau} e^{(R_w(\tau))}} \sum_{\tau} e^{(R_w(\tau))} \frac{dR_w(\tau)}{dw}$$

$$\sum_{\tau} P(\tau|w)$$

## 最大熵IRL

$$\nabla L(w) = \sum_{\tau \in D_E} \frac{dR_w(\tau)}{dw} - M \frac{1}{\sum_{\tau} e^{(R_w(\tau))}} \sum_{\tau} e^{(R_w(\tau))} \frac{dR_w(\tau)}{dw}$$

$$= \sum_{\tau \in D_E} \mu_{\tau} - \sum_{\tau \in D_E} P(\tau|w) \mu_{\tau}$$

$$\mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R(s_t)|\pi\right] = w^T \mu(\pi)$$

$$= \mu_{\text{dem}} - \sum_{\tau \in D_E} P(\tau|w) \mu_{\tau}$$

$$= \mu_{\text{dem}} - \sum_s P(s|w) \mu_s$$

状态访问频率

令  $\delta_t(s)$  为在时间  $t$  下访问状态  $s$  的概率

$$\delta_1(s) = p(s_1 = s)$$

$$\delta_{t+1}(s') = \sum_a \sum_s \delta_t(s) \pi(a|s) P(s'|s, a) \rightarrow P(s|w) = \frac{1}{T} \sum_t \delta_t(s)$$

## 最大熵逆强化学习

1. 初始化参数  $w$ , 收集专家样本集  $D_E$
2. 基于奖赏函数  $R_w$  求最优策略  $\pi(a|s)$
3. 求解状态访问频率  $P(s|w)$
4. 计算梯度  $\nabla L(w) = -\frac{1}{|D|} \sum_{\tau \in D_E} \frac{dR_w(\tau)}{dw} - \sum_s P(s|w) \frac{dR_w(s)}{dw}$
5. 基于梯度  $\nabla L(w)$  更新参数  $w$

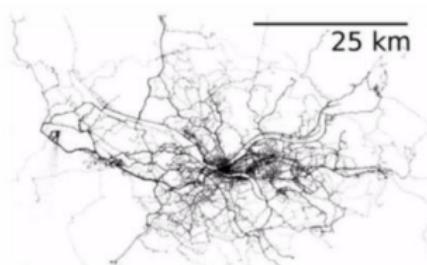
<https://www.youtube.com/watch?v=d9DIQSJQAOI>

# Maximum Entropy Inverse Reinforcement Learning

Brian D. Ziebart, Andrew Maas, J. Andrew Bagnell, and Anind K. Dey

School of Computer Science  
Carnegie Mellon University  
Pittsburgh, PA 15213

bziebart@cs.cmu.edu, amaas@andrew.cmu.edu, dbagnell@ri.cmu.edu, anind@cs.cmu.edu



Feature	Value
Hard left turn	3
Soft left turn	5
Soft right turn	0
Hard right turn	25
No turn	25
U-turn	0

对于**状态转移概率模型未知**的情况

## 利用相对熵

$Q$  为利用均匀分布策略产生的轨迹分布

$$\min_P \sum_{\tau \in \mathcal{T}} P(\tau) \ln \frac{P(\tau)}{Q(\tau)}$$

$$\sum_{\tau} P(\tau) \mu_{\tau} = \mu_{\text{dem}}$$

利用重要性采样的方法放松了求次梯度时对模型的依赖

$$\sum_{\tau} P(\tau) = 1$$

Boularias A, Kober J, Peters J. Relative entropy inverse reinforcement learning. 2011.

最大熵逆强化学习提供了在许多可能的奖赏函数中选择最优的途径，消除了最大边际IRL中的歧义问题，影响重大

最大熵逆强化学习的不足：

1. 假设奖赏函数可以表示为特征的线性组合
  - Guided Cost Learning, Finn et al. 2016 (重要性采样)
2. 要求转移概率模型已知  
sample based approximations for the partition function  $Z$ :  
Kalakrishnan et al. 2013, Finn et al. 2016

# Guided Cost Learning: Deep Inverse Optimal Control via Policy Optimization

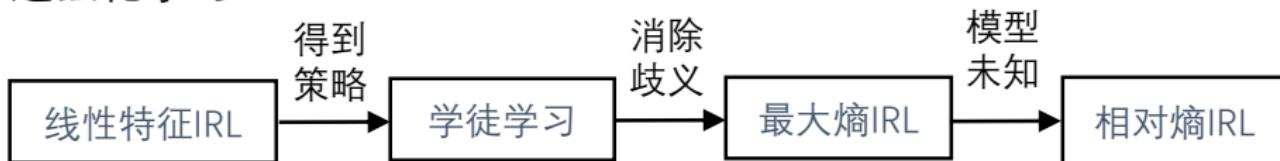
Chelsea Finn, Sergey Levine, Pieter Abbeel  
UC Berkeley



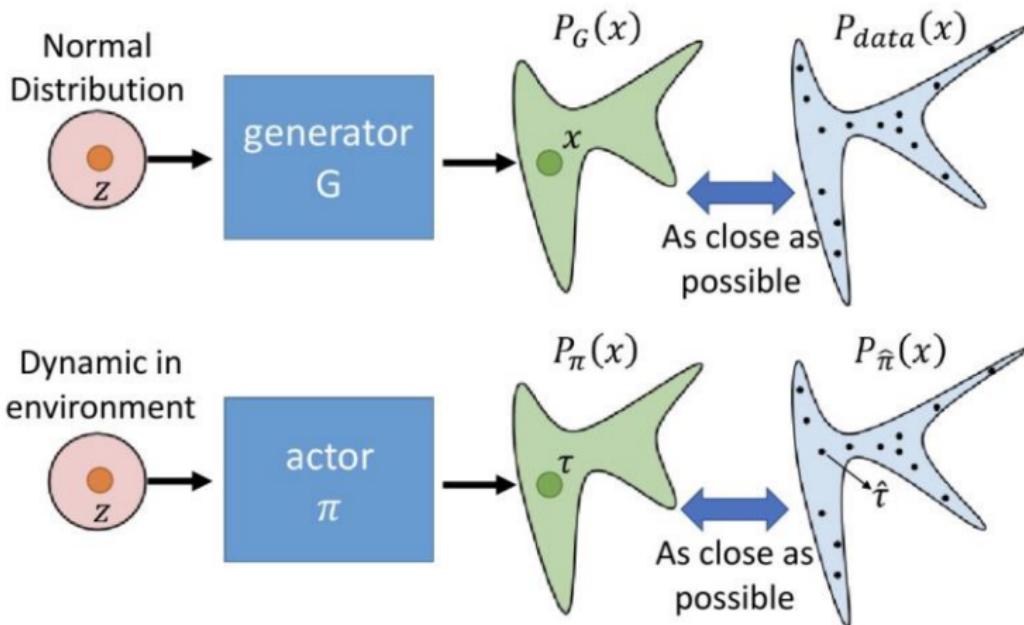
## 模仿学习



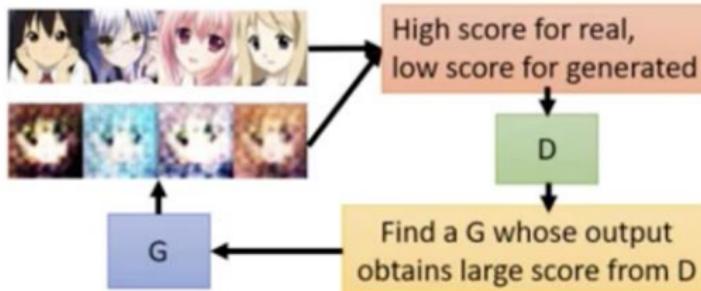
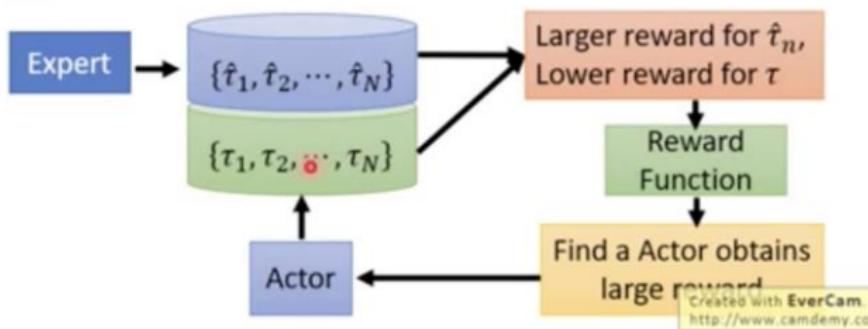
## 逆强化学习



专家轨迹是属于某一分布，能否构建一个模型也预测一个分布，并且使这两个分布尽可能的接近，这样我们就可以用该模型生成大量虚拟样本



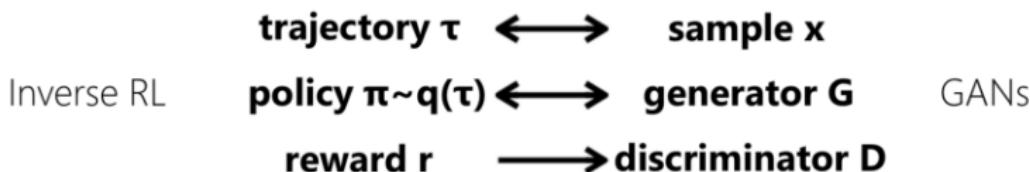
# 生成对抗模仿学习(GAIL)

GANIRL

唯一不同只是样本形式

Created with EverCam.  
<http://www.camdemmy.com>

# 生成对抗模仿学习(GAIL)



**GAN的损失函数：**

$$\min_G \max_D \mathbb{E}_{x \sim P_{data}(x)} [\log D(x)] + \mathbb{E}_{z \sim P_z(z)} [\log(1 - D(G(z)))]$$

鉴别器D的任务是区分生成器G生成的数据与真实数据。  
通过训练，找到稳定后的鉴别器D和生成器G的网络参数。

Goodfellow I J, Pouget-Abadie J, Mirza M, et al. Generative Adversarial Networks. NIPS, 2014.

## GAIL的损失函数：

$$\min_{\pi} \max_D \mathbb{E}_{\pi_E} [\log(D(s, a))] + \mathbb{E}_{\pi} [\log(1 - D(s, a))] - \lambda H(\pi)$$

- $\pi_\theta$ 是一个参数化的policy,  $\theta$ 为权重。 $D_\omega$ 是一个参数化的鉴别器, 权重为 $\omega$ 。
- 对 $\omega$ 使用Adam梯度算法, 从而使上式上升。
- 对 $\theta$ 使用TRPO算法, 从而使上式下降。
- TRPO能够保证 $\pi_{\theta_{i+1}}$ 不远离 $\pi_{\theta_i}$

# 生成对抗模仿学习(GAIL)

---

## Algorithm 1 Generative adversarial imitation learning

- 1: **Input:** Expert trajectories  $\tau_E \sim \pi_E$ , initial policy and discriminator parameters  $\theta_0, w_0$
- 2: **for**  $i = 0, 1, 2, \dots$  **do**
- 3:   Sample trajectories  $\tau_i \sim \pi_{\theta_i}$
- 4:   Update the discriminator parameters from  $w_i$  to  $w_{i+1}$  with the gradient

$$\hat{\mathbb{E}}_{\tau_i}[\nabla_w \log(D_w(s, a))] + \hat{\mathbb{E}}_{\tau_E}[\nabla_w \log(1 - D_w(s, a))] \quad (17)$$

- 5:   Take a policy step from  $\theta_i$  to  $\theta_{i+1}$ , using the TRPO rule with cost function  $\log(D_{w_{i+1}}(s, a))$ . Specifically, take a KL-constrained natural gradient step with

$$\begin{aligned} & \hat{\mathbb{E}}_{\tau_i} [\nabla_\theta \log \pi_\theta(a|s) Q(s, a)] - \lambda \nabla_\theta H(\pi_\theta), \\ & \text{where } Q(\bar{s}, \bar{a}) = \hat{\mathbb{E}}_{\tau_i} [\log(D_{w_{i+1}}(s, a)) \mid s_0 = \bar{s}, a_0 = \bar{a}] \end{aligned} \quad (18)$$

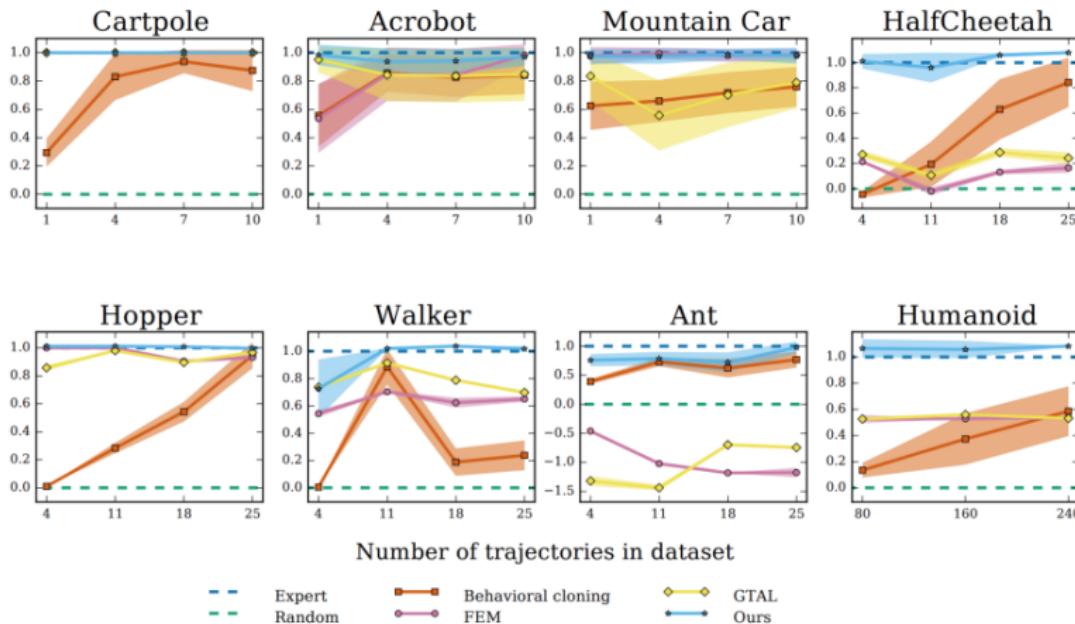
- 6: **end for**
- 

传统IRL

"专家样本-> reward-> RL policy"

GAIL: 严格的数学证明  
 "专家样本-> RL policy"  
 不需要显示的reward

## 生成对抗模仿学习(GAIL)



## 延伸阅读

- Generative Adversarial Imitation Learning, GAIL(J Ho, 2016 NIPS)
  - Imitating Driver Behavior with GAN ( Kuefler, 2017)
- InfoGAIL(Li, 2017 NIPS)
  - Imitation Driving from visual demo in TORCS
  - Latent variable c , interpretable, latent mode in expert trajectory
  - InfoGAN(Chen, 2016 NIPS)
- One-shot Imitation Learning(openAI, 2017 NIPS)
  - few demonstrations, instantly generalize in same task
- Generative Adversarial Self-Imitation Learning, GASIL(2019, ICLR)

## 本节课内容

- 1.1 神经网络
- 1.2 卷积神经网络
- 1.3 逆强化学习

模仿学习：行为克隆，样本增广，数据聚合

逆向强化学习：离散空间IRL，学徒学习，最大熵/相对熵IRL，GAIL

下节课内容

深度强化学习