四川大學
SICHUAN UNIVERSITY

机器学习-第十五章 强化学习简介

教师：胡俊杰 副教授

邮箱：hujunjie@scu.edu.cn

1.凸优化概述

■ 优化问题的一般形式

$$minimize\ f_0(x)$$

待优化的目标函数

$$subject\ to\ f_i(x) \boxed{\leq} 0, i = 1,2, \ldots, m$$

$$(s.t.)$$

m个不等式约束

$$h_i(x) = 0, i = 1,2, \ldots, p$$

p个等式约束

1.凸优化概述

$$minimize\ f_0(x)$$

$$s.t.\ f_i(x) \le 0, i = 1,2,\dots,m$$

$$h_i(x) = 0, i = 1,2,\dots,p$$

■ 拉格朗日函数

- 为每个约束指定一个拉格朗日乘子，以乘子为加权系数将约束增加到目标函数中

$$L(x,\lambda,v) = f_0(x) + \sum_{i=1}^{m} \lambda_i f_i(x) + \sum_{i=1}^{p} v_i h_i(x)$$

$$L(x,\lambda,v): R^n \times R_+^m \times R^p \to R$$

$$x \in R^n \qquad \lambda \in R_+^m, v \in R^p$$

1.凸优化概述

- 拉格朗日函数

$$L(x,\lambda,v) = f_0(x) + \sum_{i=1}^{m} \lambda_i f_i(x) + \sum_{i=1}^{p} v_i h_i(x)$$

$$\begin{aligned} & minimize\ f_0(x) \\ & s.t.\ f_i(x) \le 0, i = 1,2,\dots,m \\ & \qquad h_i(x) = 0, i = 1,2,\dots,p \end{aligned}$$

- 拉格朗日对偶函数
  - 对拉格朗日函数$L(x,\lambda,v)$中的x取下确界可定义拉格朗日对偶函数

$$g(\lambda,v) = \inf_{x\in R^n} L(x,\lambda,v) = \inf_{x\in R^n}\left(f_0(x) + \sum_{i=1}^{m} \lambda_i f_i(x) + \sum_{i=1}^{p} v_i h_i(x)\right)$$

- 拉格朗日对偶函数 $g(\lambda,v)$: $R_+^m \times R^p \rightarrow R$

inf (infimum): 下确界，数学分析中的概念，小于等于集合中的所有成员的最大实数

$$\inf\{x \in R: 0 < x < 1\} = 0$$

sup (supremum) : 上确界，大于等于集合中所有成员的最小实数

$$\sup\{x \in R: 0 < x < 1\} = 1$$

1.凸优化概述

■ 拉格朗日对偶函数

$$g(\lambda, v) = \inf_{x \in R^n} L(x, \lambda, v) = \inf_{x \in R^n} \left(f_0(x) + \sum_{i=1}^{m} \lambda_i f_i(x) + \sum_{i=1}^{p} v_i h_i(x) \right)$$

- 拉格朗日对偶函数是凹函数，无论原问题是否为凸问题
- 拉格朗日对偶函数给出了原问题最优值的下界：$g(\lambda, v) \leq p^*$， p^*：原问题（primal problem）的最优值（optimal value）

1.凸优化概述

■ 拉格朗日对偶函数

$$g(\lambda, v) = \inf_{x \in R^n} L(x, \lambda, v) = \inf_{x \in R^n} \left(f_0(x) + \sum_{i=1}^{m} \lambda_i f_i(x) + \sum_{i=1}^{p} v_i h_i(x) \right)$$

从拉格朗日对偶函数获得的下界中，哪个是最优的?

当$g(\lambda, v) = -\infty$，则其提供的下界无实际意义

■ 拉格朗日对偶问题

$$\max_{\lambda \geq 0, v} g(\lambda, v) = \max_{\lambda \geq 0, v} \inf_{x \in R^n} L(x, \lambda, v)$$

- 假设拉格朗日对偶问题（dual problem）的最优值为d^*

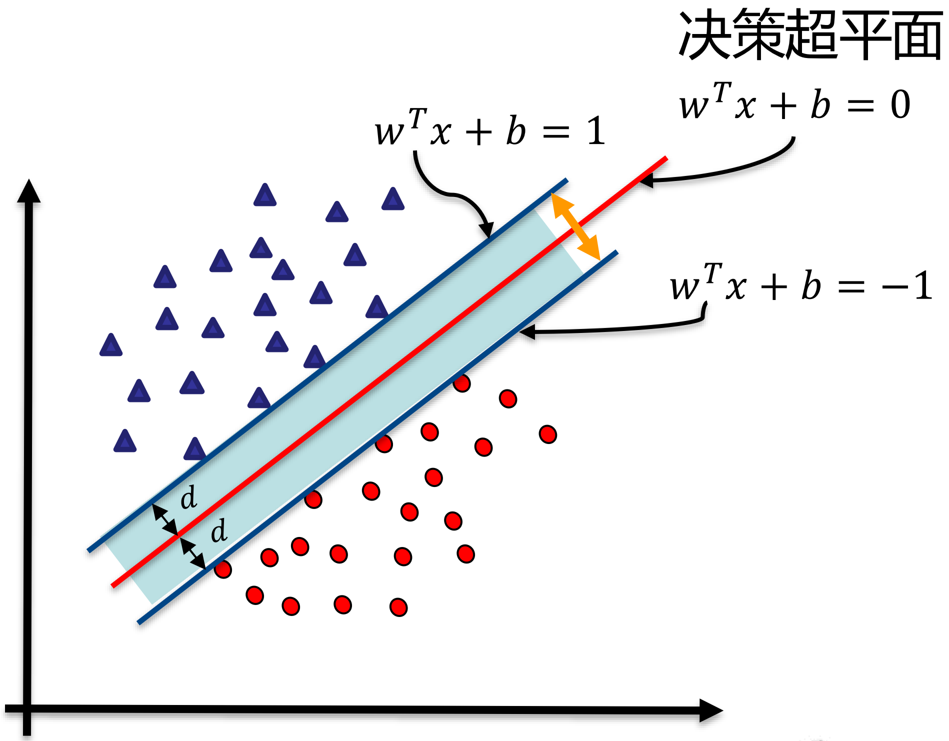
2.支持向量机概述

- 给定训练数据集$D=\{(x_1,y_1),(x_2,y_2),\ldots,(x_N,y_N)\}$，其中

  $x_i\in R^n, y_i\in\{+1,-1\}, i=1,2,\ldots,N$

假设超平面(w,b)将线性可分数据集正确分类，即对于$(x_i,y_i)\in D$，若$y_i=+1$，则$w^Tx_\mathrm{i}+b>0$；若$y_i=-1$，则$w^Tx_\mathrm{i}+b<0$，令

$$\begin{cases} w^Tx_\mathrm{i}+b\geq +1, y_i=+1 \\ w^Tx_i+b\leq -1, y_i=-1 \end{cases}$$



- 将以上两个方程合并，可得：$y_i(w^Tx_i+b)\geq 1$

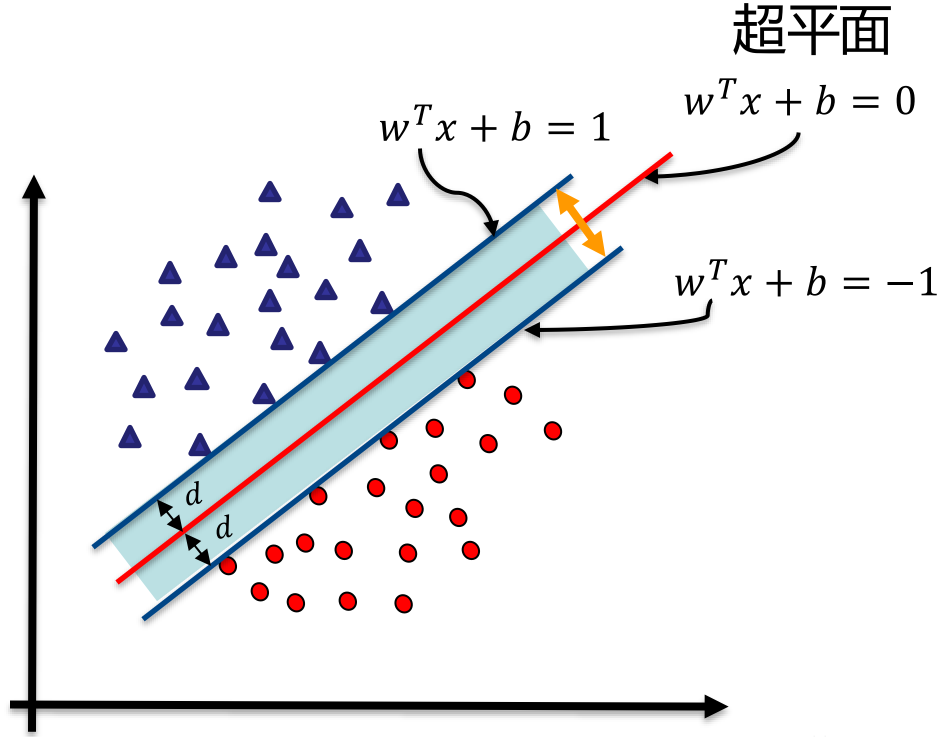
- $y_i(w^Tx_i+b)\geq 1$为约束条件，即要求决策超平面(w,b)将所有样本分类正确

2.支持向量机概述

- 支持向量到超平面的距离可以写为：$d=\frac{|w^Tx+b|}{||w||}=\frac{1}{||w||}$
- 两类（正类和负类）支持向量到超平面的距离之和为$\gamma=\frac{2}{||w||}$，也被称为“间隔”，即优化目标

$$\max_{w,b}\frac{2}{||w||}$$

$$s.t.\, y_i(w^Tx_i+b)\geq 1, i=1,2,\ldots,N$$



2.支持向量机概述

$$\max_{w,b} \frac{2}{||w||}$$

$$s.t.\ y_i(w^T x_i + b) \geq 1, i = 1,2,\ldots,N$$

$$\Updownarrow$$

$$\min_{w,b} \frac{1}{2} ||w||^{\boxed{2}}$$ 方便后续求导

$$s.t.\ y_i(w^T x_i + b) \geq 1, i = 1,2,\ldots,N$$

- 即支持向量机（SVM）的基本形式

3.支持向量机求解

原优化问题：
$$\min_{w,b} \frac{1}{2}\|w\|^2$$
$$s.t.\, y_i(w^T x_i + b) \geq 1, i = 1,2,\ldots,N$$

拉格朗日函数：$L(w,b,\alpha) = \frac{1}{2}\|w\|^2 - \sum_{i=1}^{N} \alpha_i y_i(w^T x_i + b) + \sum_{i=1}^{N} \alpha_i$

- 线性可分SVM满足强对偶条件，即$d^* = p^*$，原问题最优值等于对偶问题最优值
- 通过解对偶问题，得到最优解$\alpha^* = (\alpha_1^*, \alpha_2^*, \ldots, \alpha_N^*)$，得到原问题的最优解$(w^*, b^*)$

3.支持向量机求解

线性可分支持向量机学习算法

第1步：根据原始优化问题，写出拉格朗日函数

$$L(w,b,\alpha)=\frac{1}{2}\|w\|^2-\sum_{i=1}^{N}\alpha_i y_i(w^T x_i+b)+\sum_{i=1}^{N}\alpha_i$$

第2步：求$\min\limits_{w,b} L(w,b,\alpha)$，并代入$L(w,b,\alpha)$

$$\min_{w,b} L(w,b,\alpha)=-\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j\left(x_i\cdot x_j\right)+\sum_{i=1}^{N}\alpha_i$$

第3步：求解拉格朗日对偶问题，即

$$\max_{\alpha}\min_{w,b} L(w,b,\alpha)=\max_{\alpha}\ -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j\left(x_i\cdot x_j\right)+\sum_{i=1}^{N}\alpha_i$$

$$s.t.\sum_{i=1}^{N}\alpha_i\, y_i=0$$

$$\alpha_i\geq 0, i=1,2,\ldots,N$$

求得最优解$\alpha^*=(\alpha_1^*,\alpha_2^*,\ldots,\alpha_N^*)$

第4步：根据KKT条件可得原优化问题最优解w^*和b^*

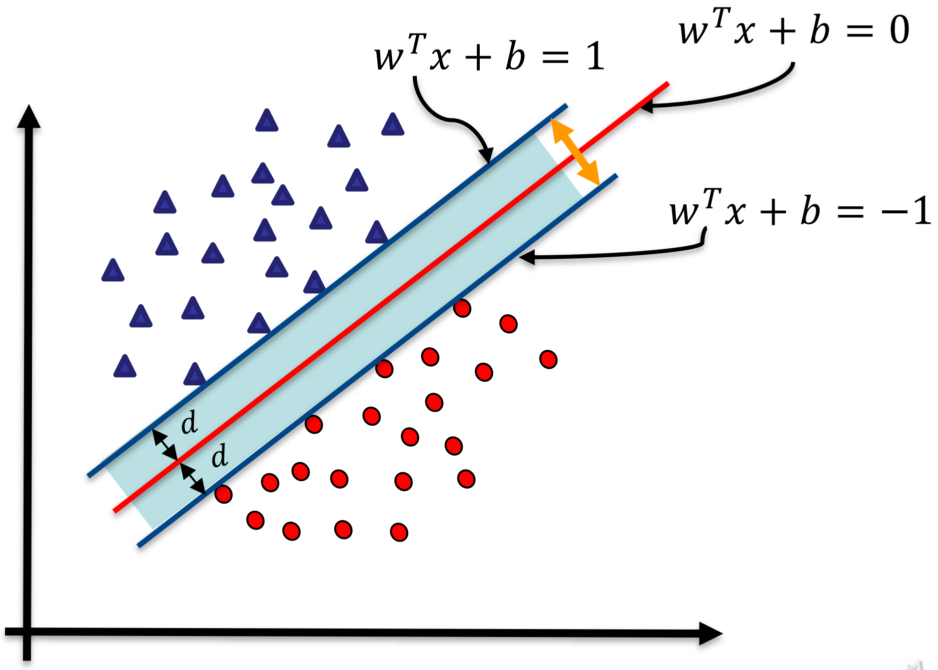
$$w^*=\sum_{i=1}^{N}\alpha_i\, y_i x_i \qquad b^*=y_j-\sum_{i=1}^{N}\alpha_i^* y_i(x_i\cdot x_j)$$

第5步：构建决策超平面以及分类决策函数

决策超平面：$w^{*T}x+b^*=0$

决策函数：$f(x)=sign(w^{*T}x+b^*)$

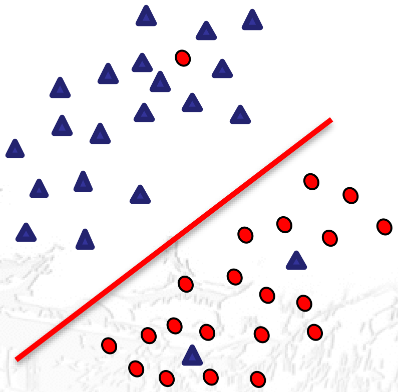
1.线性支持向量机



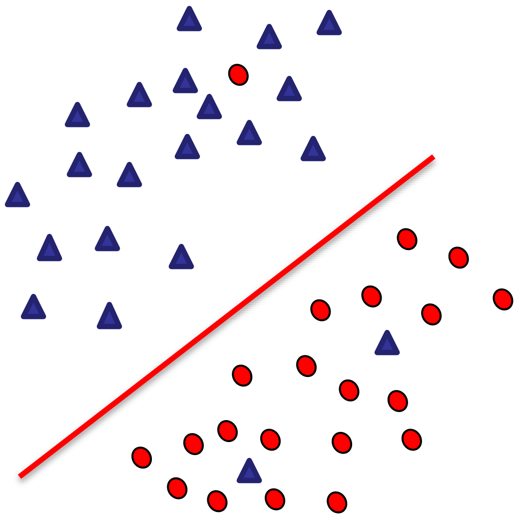
$$\min_{w,b} \frac{1}{2}\|w\|^2$$

$$s.t.\ y_i(w^T x_i + b) \geq 1, i = 1,2, \dots, N$$

- 以上假设训练样本是线性可分的，即存在一个超平面将两类样本完全分开
- 然而现实情况很复杂（如标签错误），应允许支持向量机对部分样本出错



1.线性支持向量机



- 允许一些样本不满足约束 $y_i(w^T x_i + b) \geq 1$

- 在最大化间隔的同时，不满足约束的样本应尽可能少，目标如下

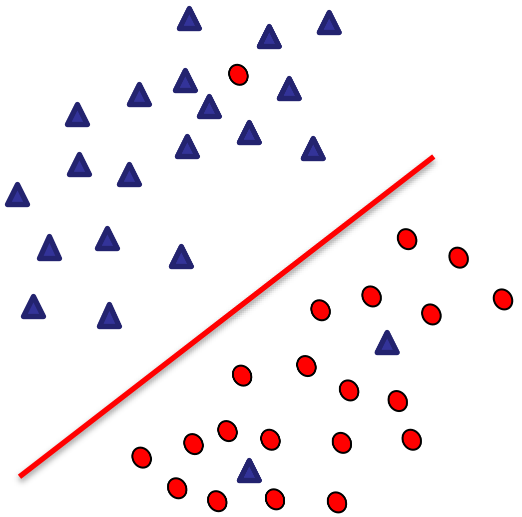
$$\min_{w,b} \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{N} l_{0/1}(y_i(w^T x_i + b) - 1)$$

  - $C > 0$为人为指定的惩罚参数，$l_{0/1}$代表0/1损失函数

$$l_{0/1} = \begin{cases} 1, if\ z < 0 \\ 0, otherwise \end{cases}$$

  - 常量C允许部分样本不满足约束 $y_i(w^T x_i + b) \geq 1$

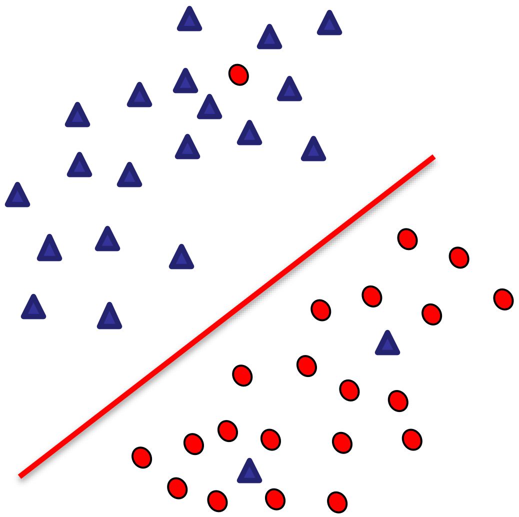
1.线性支持向量机



$$\min_{w,b} \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{N} l_{0/1}(y_i(w^T x_i + b) - 1)$$

- $l_{0/1}$非凸、不连续，以上目标函数不易求解。用数学性质较好的“代理损失函数”（surrogate loss function），替代$l_{0/1}$
- 常用代理损失函数:
  - hinge损失函数（hinge loss）：$l_{hinge}(z) = \max(0,1-z)$
  - 指数损失函数（exponential loss）：$l_{exp}(z) = e^{-z}$
  - 对率损失函数（logistic loss）：$l_{log}(z) = \log(1+e^{-z})$

1.线性支持向量机



$$\min_{w,b} \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{N} l_{0/1}(y_i(w^T x_i + b) - 1)$$

$l_{hinge}(z) = \max(0,1 - z)$ 代替$l_{0/1}$

$$\min_{w,b} \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{N} \max\left(0,1 - y_i(w^T x_i + b)\right)$$

$\xi_i = \max(0,1 - y_i(w^{\mathrm{T}} x_i + b))$

(/ksaɪ/)

$$\min_{w,b,\xi_i} \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{N} \xi_i$$

$$s.t.\, y_i(w^T x_i + b) \geq 1 - \xi_i$$

$$\xi_i \geq 0, i = 1,2, \ldots, N$$

优化目标及约束

1.线性支持向量机

线性支持向量机的拉格朗日对偶问题

$$\min_{\alpha} \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j (x_i \cdot x_j) - \sum_{i=1}^{N}\alpha_i$$

$$s.t. \sum_{i=1}^{N}\alpha_i y_i = 0$$

$$0 \le \alpha_i \le C, i = 1,2,\dots,N$$

C为为人为指定的
惩罚参数

线性可分支持向量机的拉格朗日对偶问题

$$\min_{\alpha} \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j (x_i \cdot x_j) - \sum_{i=1}^{N}\alpha_i$$

$$s.t. \sum_{i=1}^{N}\alpha_i\, y_i = 0$$

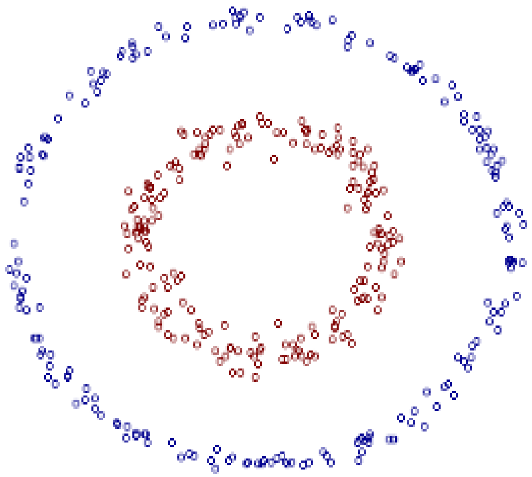
$$0 \le \alpha_i, i = 1,2,\dots,N$$

- 设$\alpha^* = (\alpha_1^*, \alpha_2^*, \dots, \alpha_N^*)$为拉格朗日对偶问题最优解，则原问题最优解$w^*$和$b^*$如下

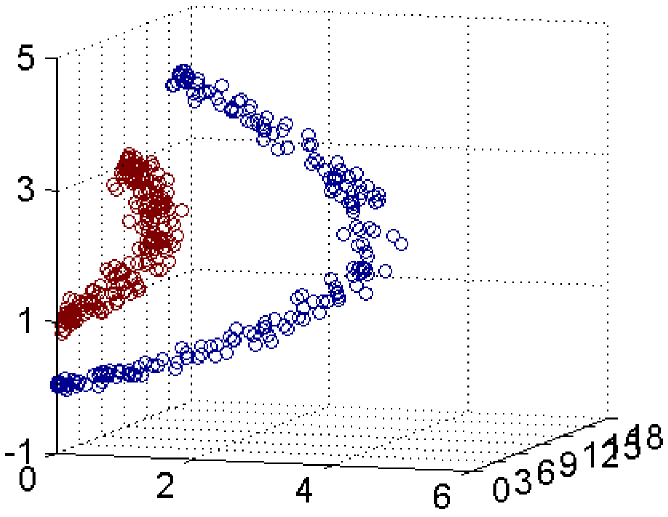
$$w^* = \sum_{i=1}^{N}\alpha_i^*\, y_i x_i \qquad b^* = y_j - \sum_{i=1}^{N}\alpha_i^* y_i (x_i \cdot x_j)$$

由KKT中的互补松弛条件可得

3.非线性支持向量机



线性不可分



高维下可分

- 无法找到一个超平面将两类样本分开
- 解决思路：对输入x作用非线性变换ϕ，将x 从原始空间映射至高维空间，使得$\phi(x)$可分

3.非线性支持向量机

线性可分支持向量机

$$\min_{w,b} \frac{1}{2}\|w\|^2$$

$$s.t.\, y_i(w^T x_i + b) \geq 1, i = 1,2,\dots,N$$

拉格朗日对偶问题如下:

$$\max_{\alpha} -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j (x_i \cdot x_j) + \sum_{i=1}^{N}\alpha_i$$

$$s.t. \sum_{i=1}^{N}\alpha_i\, y_i = 0$$

$$\alpha_i \geq 0, i = 1,2,\dots,N$$

非线性支持向量机，优化目标及约束如下:

$$\min_{w,b} \frac{1}{2}\|w\|^2$$

$$s.t.\, y_i(w^T \phi(x_i) + b) \geq 1, i = 1,2,\dots,N$$

拉格朗日对偶问题如下:

$$\max_{\alpha} -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j \big(\phi(x_i) \cdot \phi(x_j)\big) + \sum_{i=1}^{N}\alpha_i$$

$$s.t. \sum_{i=1}^{N}\alpha_i\, y_i = 0$$

$$0 \leq \alpha_i \leq C, i = 1,2,\dots,N$$

3.非线性支持向量机

非线性支持向量机，优化目标及约束如下:

$$\min_{w,b}\frac{1}{2}\|w\|^2$$

$$s.t.\, y_i(w^T\phi(x_i)+b)\geq 1, i=1,2,\dots,N$$

拉格朗日对偶问题如下：

$$\max_{\alpha}\ -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j\left(\phi(x_i)\cdot\phi(x_j)\right)+\sum_{i=1}^{N}\alpha_i$$

$$s.t.\sum_{i=1}^{N}\alpha_i\, y_i=0$$

$$0\leq\alpha_i\leq C, i=1,2,\dots,N$$

- 直接定义非线性映射函数ϕ较困难
- 引入核函数$K(x_i,x_j)=\phi(x_i)\cdot\phi(x_j)$，$x_i$与$x_j$在特征空间中的内积等于在原始输入空间通过函数$K(x_i,x_j)$计算的结果，从而避开定义$\phi$

3.非线性支持向量机

线性可分支持向量机

$$w^* = \sum_{i=1}^{N} \alpha_i^* y_i x_i \qquad b^* = y_j - \sum_{i=1}^{N} \alpha_i^* y_i (x_i \cdot x_j)$$

超平面：$$\sum_{i=1}^{N} \alpha_i^* y_i (x \cdot x_i) + b^* = 0$$

决策函数：$$f(x) = sign\left(\sum_{i=1}^{N} \alpha_i^* y_i (x \cdot x_i) + b^*\right)$$

非线性支持向量机

$$w^* = \sum_{i=1}^{N} \alpha_i^* y_i \phi(x_i) \qquad b^* = y_j - \sum_{i=1}^{N} \alpha_i^* y_i \underbrace{\phi(x_i) \cdot \phi(x_j)}_{K(x_i, x_j)}$$

超平面：$$\sum_{i=1}^{N} \alpha_i^* y_i \underbrace{\phi(x) \cdot \phi(x_i)}_{K(x, x_i)} + b^* = 0$$

决策函数：$$f(x) = sign\left(\sum_{i=1}^{N} \alpha_i^* y_i \underbrace{\phi(x) \cdot \phi(x_i)}_{K(x, x_i)} + b^*\right)$$

3.序列最小优化算法

拉格朗日对偶问题:

$$\max_{\alpha} -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j K(x_i,x_j) + \sum_{i=1}^{N}\alpha_i$$

$$s.t. \sum_{i=1}^{N}\alpha_i\, y_i = 0$$

$$0 \le \alpha_i \le C, i = 1,2,\dots,N$$

- 如何求解α^*?

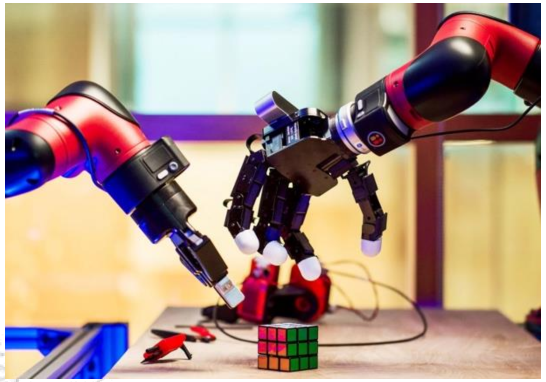
- 序列最小优化算法（Sequential Minimal Optimization）：每次选择α_i和α_j，并固定其他参数，求α_i和α_j的极值，直至收敛。求解思路如下:
  - 选择一对需要更新的变量α_i和α_j
  - 固定α_i和α_j之外的参数，求∇_{α_i}和∇_{α_j}

- SMO算法的两个主要部分:
  - 求解两个变量α_i和α_j的解析方法
  - 选择待优化变量α_i和α_j的方法

本章目录

两种人工智能类型

- 预测型任务
  - 根据数据预测所需输出（有监督学习）
  - 数据降维或得到表达（无监督学习）
- 决策型任务
  - 在动态环境中采取行动（强化学习）
    - 转变到新的状态
    - 获得即时奖励
    - 随着时间的推移最大化累计奖励



两种人工智能类型

- 决策下达到环境中，直接改变环境
  - 未来发展随之改变
- 预测仅产生信号，不考虑环境的改变
  - 不需要考虑预测的信号是否用、怎么用

智能医学

- 决策
  - 医生或者人工智能模型直接给病人下达治疗方案
- 预测
  - 人工智能模型告诉医生关于病人可能的得病预测，医生综合各方面判断给病人下达治疗方案

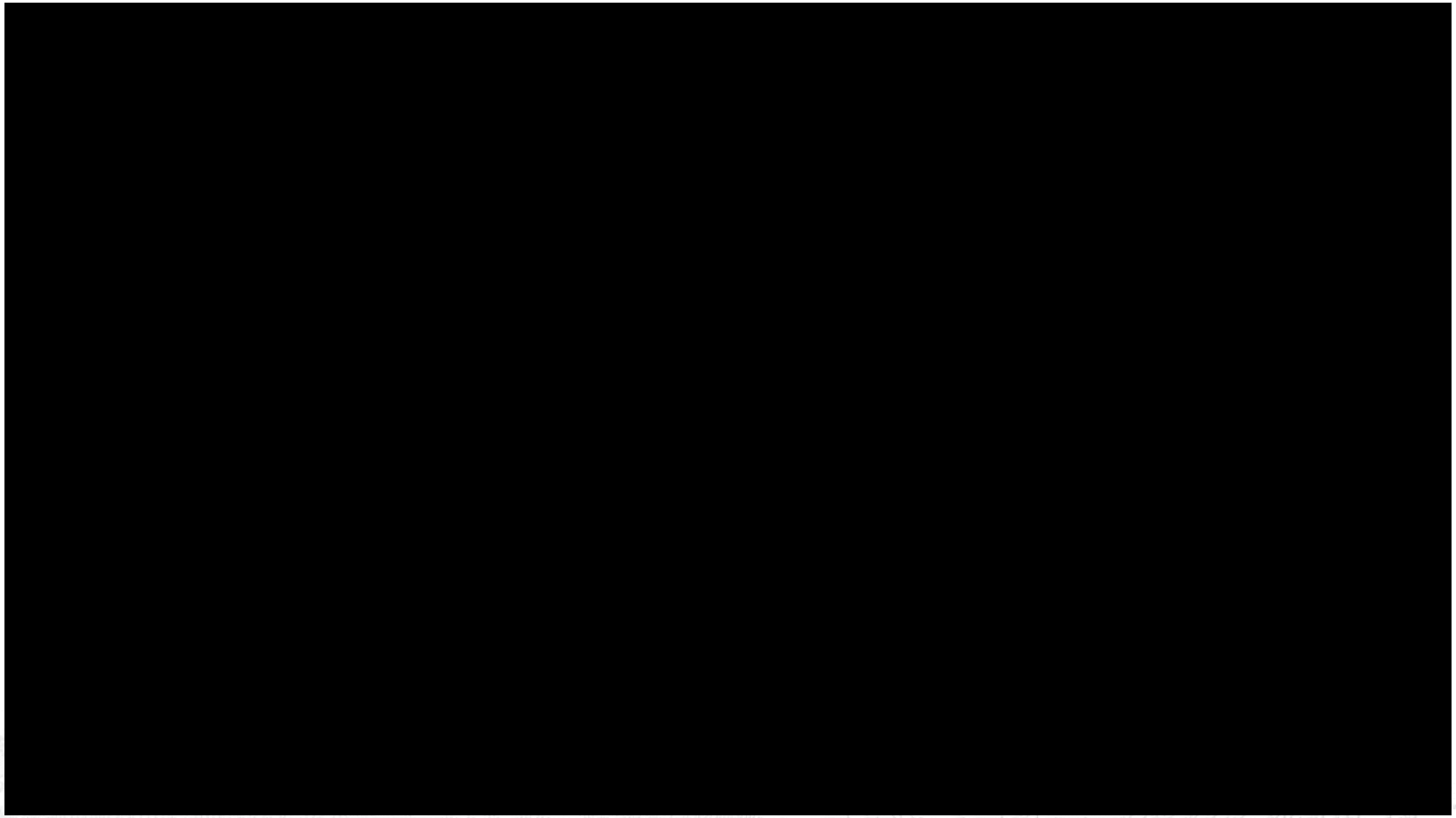
两种人工智能类型

- 序贯决策
  - 决策者序贯地做出一个个决策，并接续看到新的观测，直到最终任务结束。

绝大多数序贯决策问题，可以用强化学习来建模

两种人工智能类型

- 序贯决策
  - 决策者序贯地做出一个个决策，并接续看到新的观测，直到最终任务结束。



本章目录

强化学习定义

有监督、无监督学习

Model

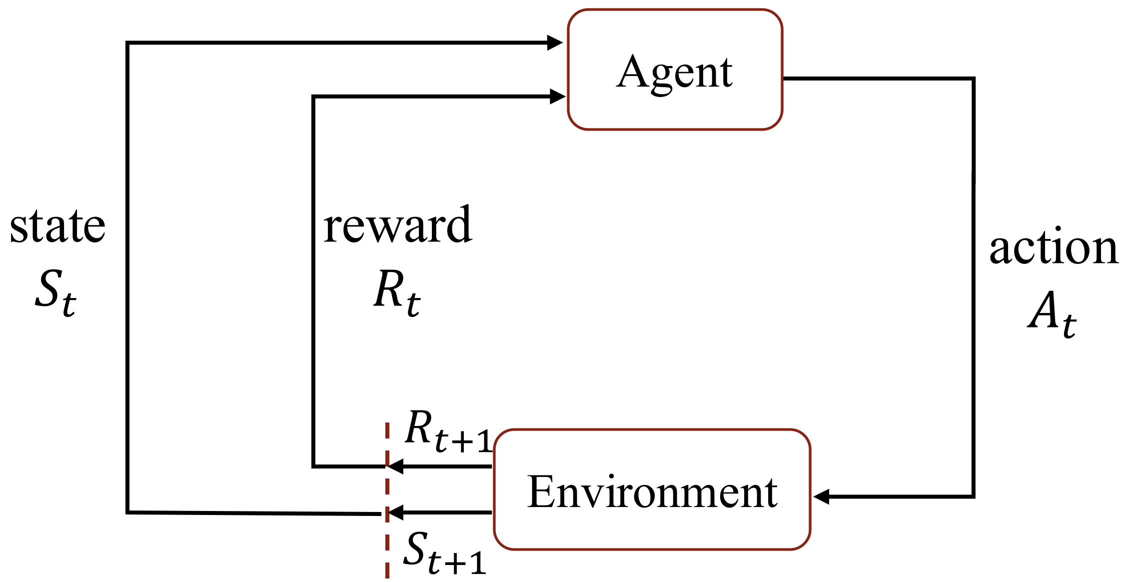
Fixed Data

强化学习

Agent
(Decision maker)

Dynamic Environment

Basic Concepts of RL



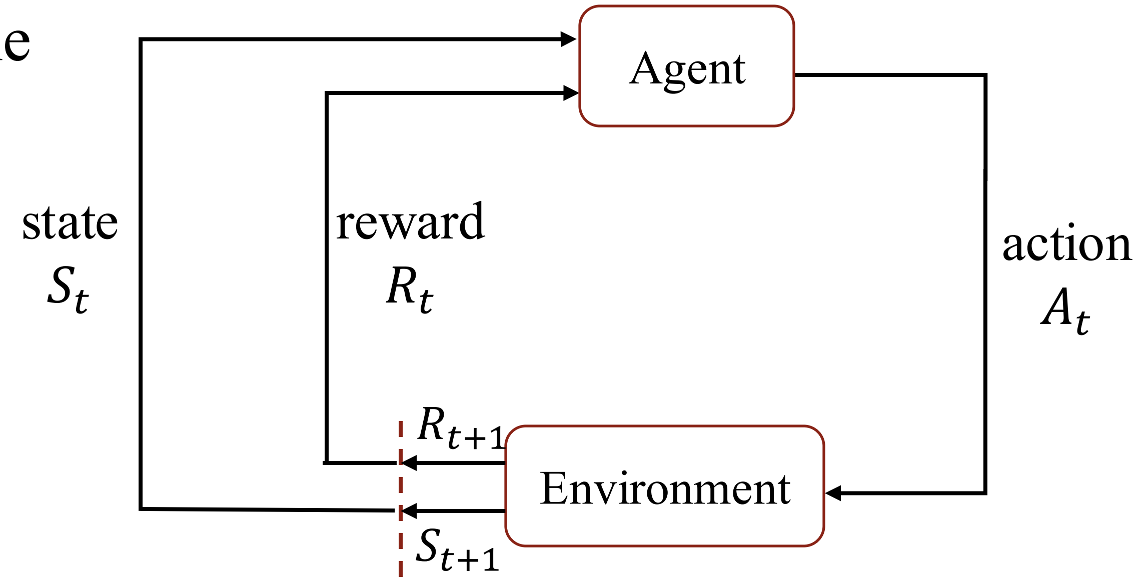
S_t, A_t, R_t are random variables.

Basic Concepts of RL : State

State: A state is a representation of the environment at a specific time.

State Space: $S = \{s_1, \cdots, s_n\}$

State $S_t = s$

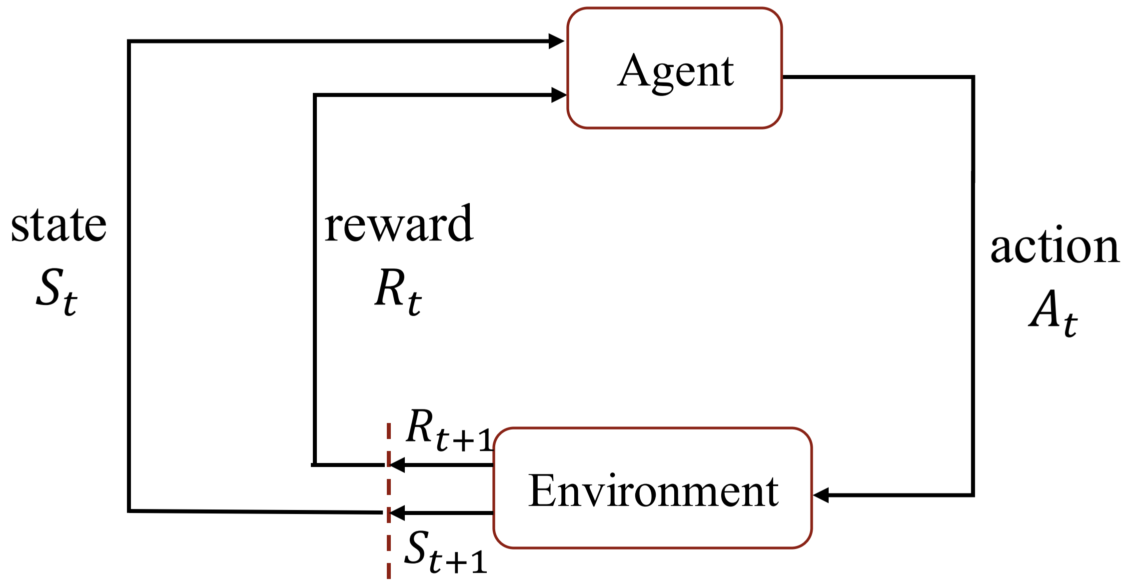


Basic Concepts of RL : Action

Action: An action is a decision or move made by the agent that affects the environment.

Action Space: $A = \{a_1, \cdots, a_n\}$

Action $A_t = a$



Basic Concepts of RL : State Transition

State transition: A state transition describes how the environment moves from one state s to another state s' after the agent takes an action a.

$$s \xrightarrow[p(s'|s,a)]{a} s'$$

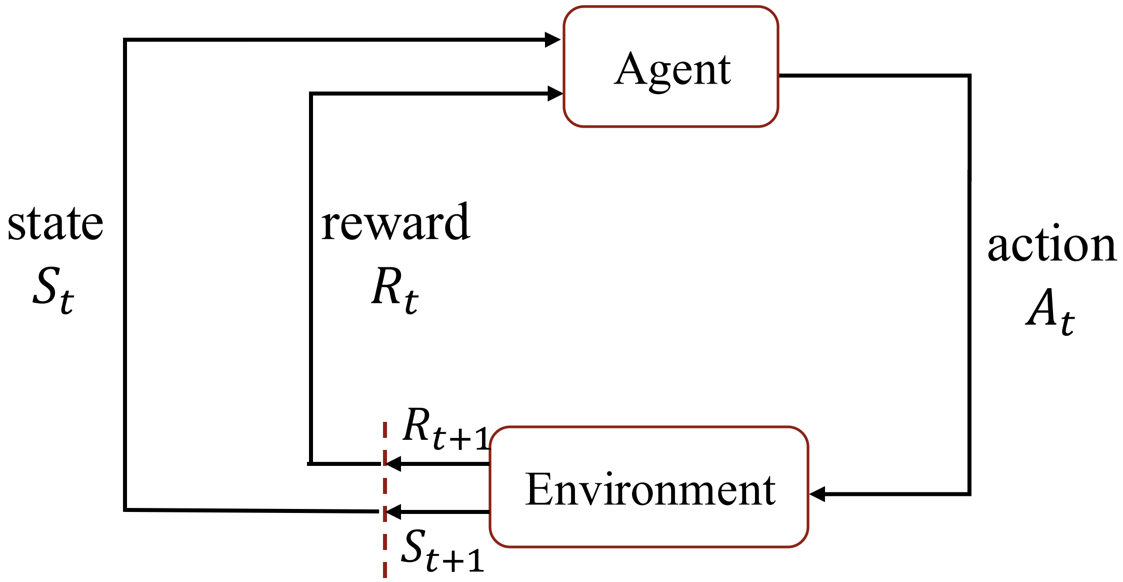
Basic Concepts of RL : Reward

Reward: a scalar signal that tells the agent how good or bad its action is in a given state.
The reward is typically defined as a function of the state or the state-action pair.

$r(s, a)$ with probability $p(r|s, a)$

Reward $R_t = r(s, a) = r$

Reward Space: $R = \{r_1, \cdots, r_n\}$

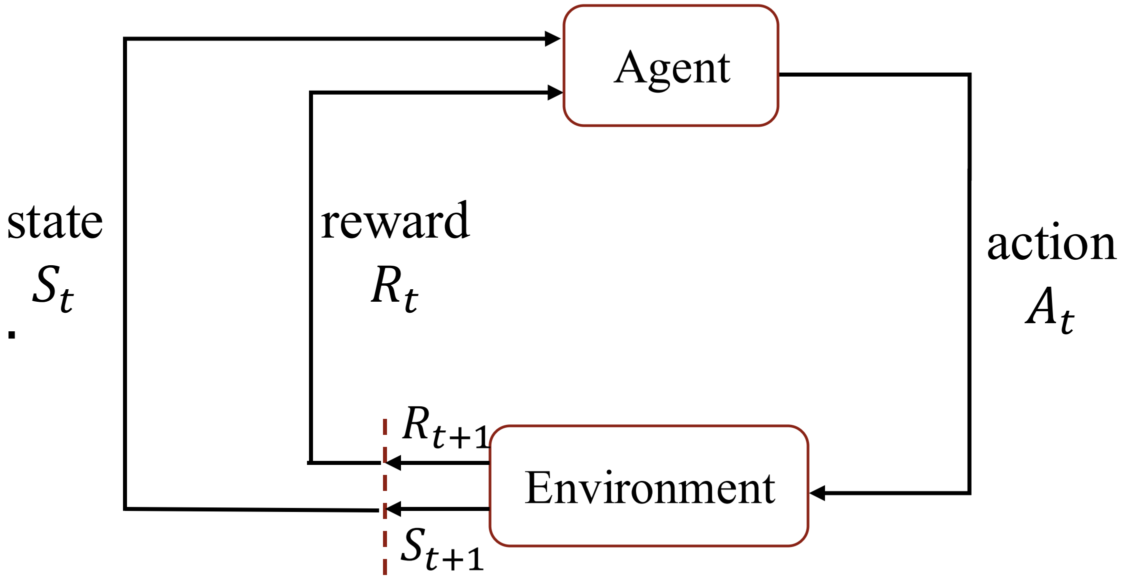


$$s \xrightarrow{a} s' \quad r(s, a)$$

$$p(s'|s, a) \qquad p(r|s, a)$$

Basic Concepts of RL : Return

$$S_t \xrightarrow{A_t} S_{t+1}, R_{t+1} \xrightarrow{A_{t+1}} S_{t+2}, R_{t+2} \xrightarrow{A_{t+2}} S_{t+3}, R_{t+3} \dots \dots$$



Relationship between Return and Reward

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots$$

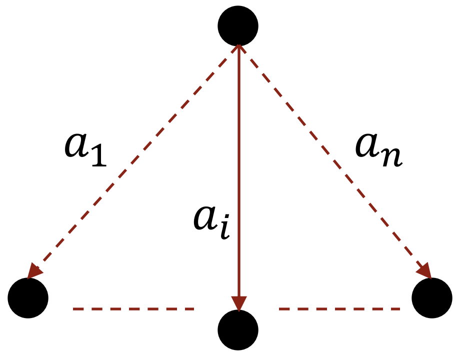
$$G_t = R_{t+1} + \gamma G_{t+1}$$

$\gamma \in [0,1]$: A hyperparameter that controls the trade-off between short-term and long-term rewards

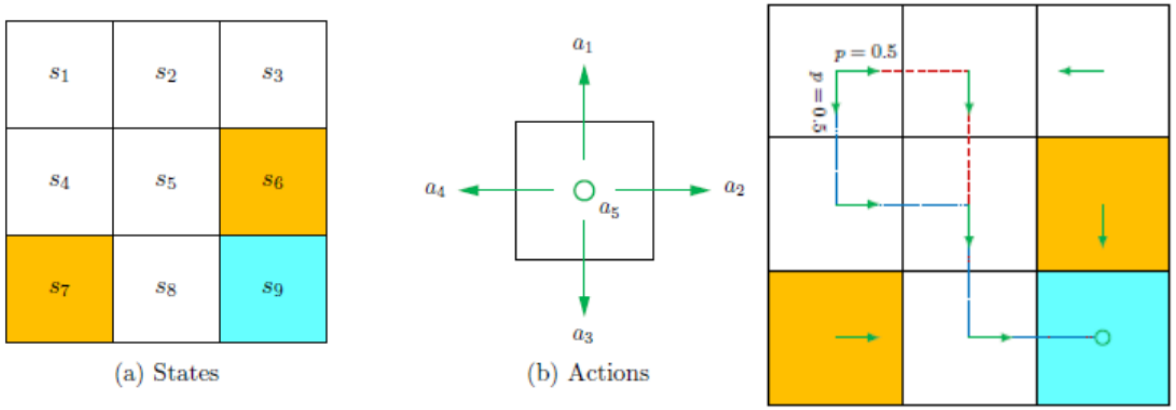
Basic Concepts of RL : Policy

Policy: A policy tells the agent which actions to take at every state.

$$\pi(a|s) = p(a|s)$$



Action space: $A = \{a_1, \cdots, a_n\}$

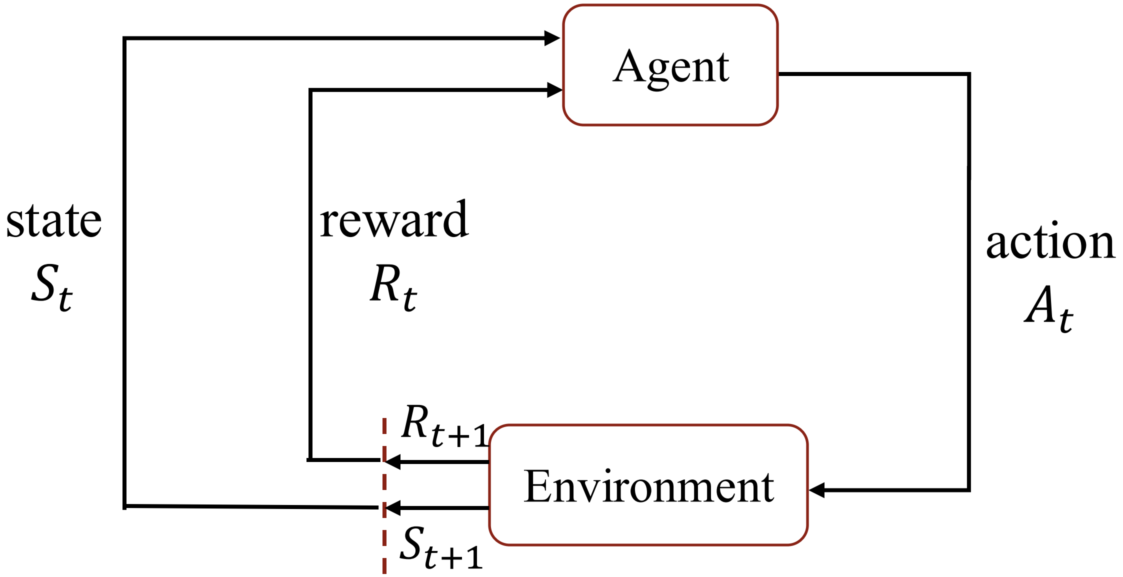


(a) States (b) Actions

	a_1 (upward)	a_2 (rightward)	a_3 (downward)	a_4 (leftward)	a_5 (still)
s_1	0	0.5	0.5	0	0
s_2	0	0	1	0	0
s_3	0	0	0	1	0
s_4	0	1	0	0	0
s_5	0	0	1	0	0
s_6	0	0	1	0	0
s_7	0	1	0	0	0
s_8	0	1	0	0	0
s_9	0	0	0	0	1

Ultimate goal of RL: Find the optimal policy

Basic Concepts of RL : Markov Decision Processes (MDP)



Markov property: The next state s_{t+1} and reward r_{t+1} depend only on the current state s_t and action a_t, not on the full history.

$$S_t \xrightarrow{A_t} S_{t+1}, R_{t+1} \xrightarrow{A_{t+1}} S_{t+2}, R_{t+2} \xrightarrow{A_{t+2}} S_{t+3}, R_{t+3} \dots \dots$$

$$p(s_{t+1}|s_t, a_t, s_{t-1}, a_{t-1}, \cdots, s_0, a_o) = p(s_{t+1}|s_t, a_t)$$

$$p(r_{t+1}|s_t, a_t, s_{t-1}, a_{t-1}, \cdots, s_0, a_o) = p(r_{t+1}|s_t, a_t)$$

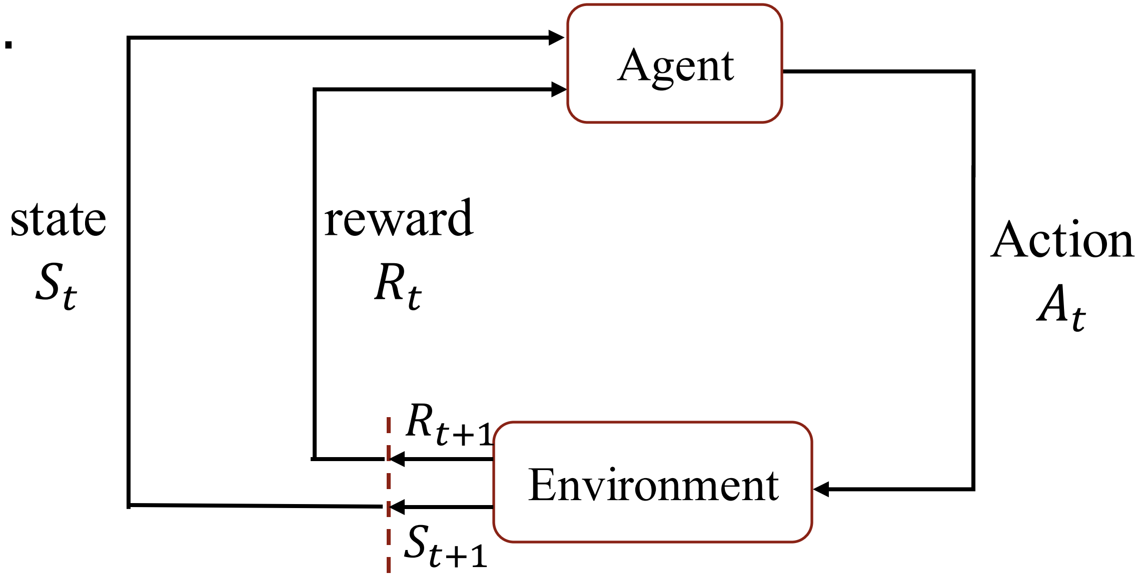
Basic Concepts of RL : State Value

$$S_t \xrightarrow{A_t} S_{t+1}, R_{t+1} \xrightarrow{A_{t+1}} S_{t+2}, R_{t+2} \xrightarrow{A_{t+2}} S_{t+3}, R_{t+3} \dots \dots$$

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots$$

State value $v_\pi(s)$: expected return (cumulative future reward) an agent can get by starting in state s and following policy π

State value measures how good it is for the agent in a state follows a certain policy



$$v_\pi(s) = E[G_t | S_t = s]$$

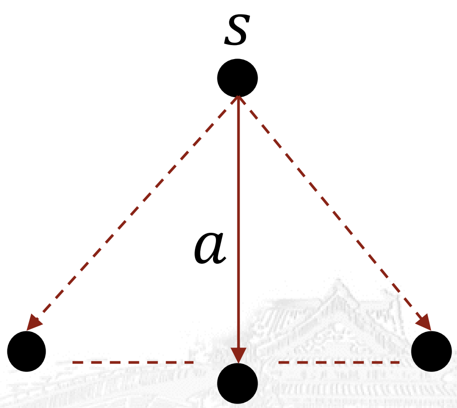
How to measure the effectiveness of action in state value?

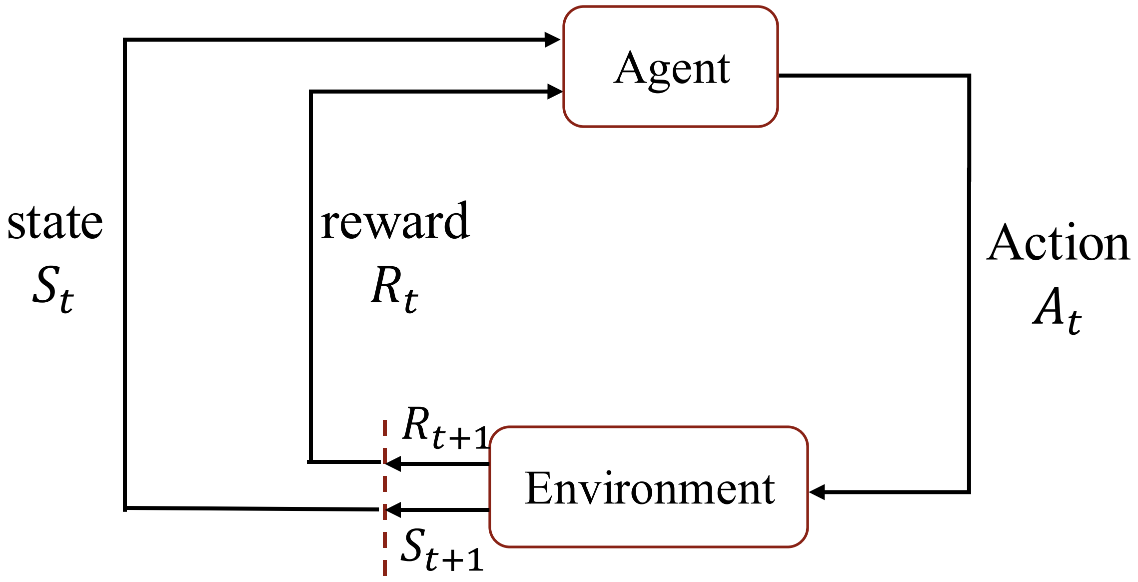
Basic Concepts of RL : Action Value

Action value $q_\pi(s, a)$: expected return starting from state s, taking action a, and then following policy π

It tells how good it is to take action a in state s under policy π

$$q_\pi(s, a) = E[G_t | S_t = s, A_t = a]$$





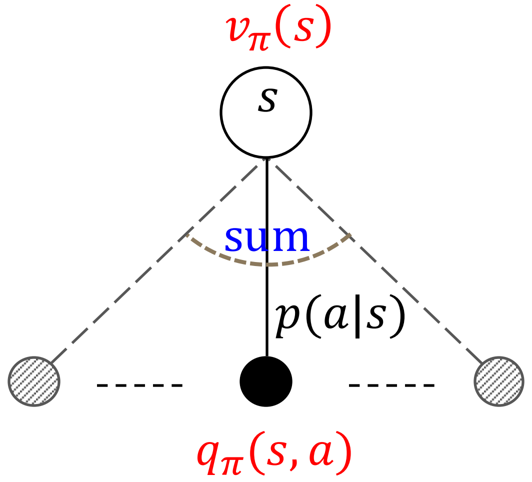
Basic Concepts of RL : Relationship between State Value and Action Value

$$v_\pi(s) = E[G_t|S_t = s]$$

$$q_\pi(s,a) = E[G_t|S_t = s, A_t = a]$$

Law of total expectation
全期望公式

$$\mathbb{E}[X] = \sum_a \mathbb{E}[X|A=a]p(a)$$



$$v_\pi(s) = E[G_t|S_t = s] = \sum_{a\in A} E[G_t|S_t = s, A_t = a]\,\pi(a|s) = \sum_{a\in A} \pi(a|s)\,q_\pi(s,a)$$

本章目录

Bellman Equation

State value: $v_\pi(s) = E[G_t|S_t = s]$

Action value: $q_\pi(s,a) = E[G_t|S_t = s, A_t = a]$

Relationship between state value and action value:

$$v_\pi(s) = \sum_{a\in A} E[G_t|S_t = s, A_t = a]\,\pi(a|s) = \sum_{a\in A} \pi(a|s)\,q_\pi(s,a)$$

How to solve the above equation to obtain state value and action value?

Bellman equation!

Bellman Equation

$$v_\pi(s) = E[G_t|S_t = s]$$
$$= E[R_{t+1} + \gamma G_{t+1}|S_t = s]$$
$$= E[R_{t+1}|S_t = s] + \gamma E[G_{t+1}|S_t = s]$$

mean of immediate rewards   mean of future rewards

$$v_\pi(s) = E[G_t|S_t = s]$$

$$G_t = R_{t+1} + \gamma G_{t+1}$$

$$E[R_{t+1}|S_t = s] = \sum_{a \in A} E[R_{t+1}|S_t = s, A_t = a]\, p(a|s) = \sum_{a \in A} \pi(a|s) \sum_{r \in R} r(s,a) p(r|s,a) = \sum_{a \in A} \pi(a|s) \sum_{r \in R} r p(r|s,a)$$

$$E[G_{t+1}|S_t = s] = \sum_{s' \in S} E[G_{t+1}|S_t = s, S_{t+1} = s']\, p(s'|s) = \sum_{s' \in S} E[G_{t+1}|S_{t+1} = s']\, p(s'|s) = \sum_{s' \in S} p(s'|s)\, v_\pi(s')$$

Markov property

$$\boxed{v_\pi(s) = \sum_{a \in A} \pi(a|s) \sum_{r \in R} r p(r|s,a) + \gamma \sum_{s' \in S} p(s'|s)\, v_\pi(s')}$$

Bellman Equation

$$v_\pi(s) = \sum_{a \in A} \pi(a|s) \sum_{r \in R} rp(r|s,a) + \gamma \sum_{s' \in S} p(s'|s)\, v_\pi(s')$$

$$p(s'|s) = \sum_{a \in A} p(s'|s,a)\pi(a|s)$$

$$v_\pi(s) = \sum_{a \in A} \pi(a|s) \left[\sum_{r \in R} rp(r|s,a) + \gamma \sum_{s' \in S} p(s'|s,a) v_\pi(s') \right]$$

Bellman Equation

From action value to state value: $v_\pi(s) = \sum_{a \in A} E[G_t | S_t = s, A_t = a]\, \pi(a|s) = \sum_{a \in A} \pi(a|s)\, q_\pi(s,a)$

From state value to action value: $v_\pi(s) = \sum_{a \in A} \pi(a|s) \boxed{\sum_{r \in R} r p(r|s,a) + \gamma \sum_{s' \in S} p(s'|s)\, v_\pi(s')}$

$$q_\pi(s,a) = \sum_{r \in R} r p(r|s,a) + \gamma \sum_{s' \in S} p(s'|s)\, v_\pi(s')$$

Bellman Equation

$$v_\pi(s) = \underbrace{\sum_{a\in A} \pi(a|s) \sum_{r\in R} rp(r|s,a)}_{r_\pi(s)} + \gamma \sum_{s'\in S} p(s'|s)\, v_\pi(s')$$

$$v_\pi(s) = r_\pi(s) + \gamma \sum_{s'\in S} p(s'|s) v_\pi(s')$$

Vector form

$$v_\pi = r_\pi + \gamma P_\pi v_\pi$$

$$v_\pi = \begin{bmatrix} v_\pi(s_1) \\ v_\pi(s_2) \\ \vdots \\ v_\pi(s_n) \end{bmatrix} \qquad r_\pi = \begin{bmatrix} r_\pi(s_1) \\ r_\pi(s_2) \\ \vdots \\ r_\pi(s_n) \end{bmatrix}$$

$$P_\pi = \begin{bmatrix} p(s_1|s_1) & \cdots & p(s_n|s_1) \\ \vdots & \ddots & \vdots \\ p(s_1|s_n) & \cdots & p(s_n|s_n) \end{bmatrix}$$

$$\sum_{j=1}^{n} p(s_j|s_i) = 1$$

Solution of Bellman Equation

Vector form

$$v_\pi = r_\pi + \gamma P_\pi v_\pi$$

$$\Downarrow$$

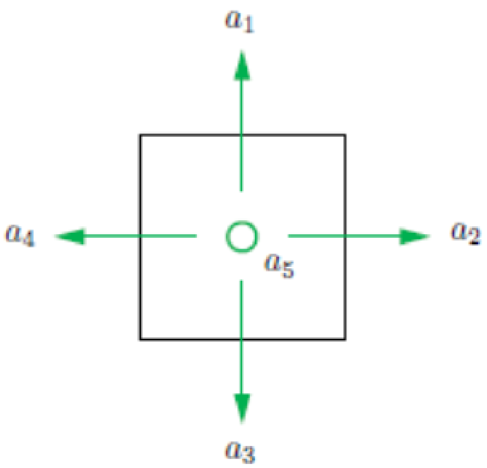
$$v_\pi = [I - \gamma P_\pi]^{-1} r_\pi$$

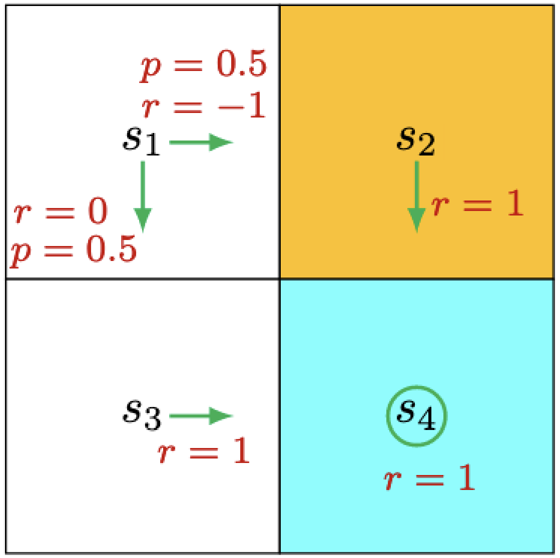
$$v_\pi = \begin{bmatrix} v_\pi(s_1) \\ v_\pi(s_2) \\ \vdots \\ v_\pi(s_n) \end{bmatrix} \quad r_\pi = \begin{bmatrix} r_\pi(s_1) \\ r_\pi(s_2) \\ \vdots \\ r_\pi(s_n) \end{bmatrix}$$

$$P_\pi = \begin{bmatrix} p(s_1|s_1) & \cdots & p(s_n|s_1) \\ \vdots & \ddots & \vdots \\ p(s_1|s_n) & \cdots & p(s_n|s_n) \end{bmatrix}$$

$$\sum_{j=1}^{n} p(s_j|s_i) = 1$$

Example of Computing Action Value and State Value





- ◇ If the agent attempts to exit the boundary, let $r_{\text{boundary}} = -1$.
- ◇ If the agent attempts to enter a forbidden cell, let $r_{\text{forbidden}} = -1$.
- ◇ If the agent reaches the target state, let $r_{\text{target}} = +1$.
- ◇ Otherwise, the agent obtains a reward of $r_{\text{other}} = 0$.

$$q_\pi(s,a) = \sum_{r \in R} r p(r|s,a) + \gamma \sum_{s' \in S} p(s'|s)\, v_\pi$$

$$v_\pi(s) = \sum_{a \in A} \pi(a|s)\, q_\pi(s,a)$$

Considering the a_2 and a_3 in s_1

$$q_\pi(s_1,a_2) = -1 + \gamma v_\pi(s_2)$$

$$q_\pi(s_1,a_3) = 0 + \gamma v_\pi(s_3)$$

$$v_\pi(s_1) = 0.5 q_\pi(s_1,a_2) + 0.5 q_\pi(s_1,a_3)$$

$$= 0.5[-1 + \gamma v_\pi(s_2)] + 0.5[-1 + \gamma v_\pi(s_2)]$$

Value Iteration to find the optimal policy

model-based method

Initialization: The probability models $p(r|s,a)$ and $p(s'|s,a)$ for all (s,a) are known. Initial guess v_0.

Goal: Search for the optimal state value and an optimal policy for solving the Bellman optimality equation.

While v_k has not converged in the sense that $\|v_k - v_{k-1}\|$ is greater than a predefined small threshold, for the kth iteration, do

For every state $s \in \mathcal{S}$, do

For every action $a \in \mathcal{A}(s)$, do

q-value: $q_k(s,a) = \sum_r p(r|s,a)r + \gamma \sum_{s'} p(s'|s,a)v_k(s')$

Maximum action value: $a_k^*(s) = \arg\max_a q_k(s,a)$

Policy update: $\pi_{k+1}(a|s) = 1$ if $a = a_k^*$, and $\pi_{k+1}(a|s) = 0$ otherwise

Value update: $v_{k+1}(s) = \max_a q_k(s,a)$

Q-table

$q_\pi(s,a)$

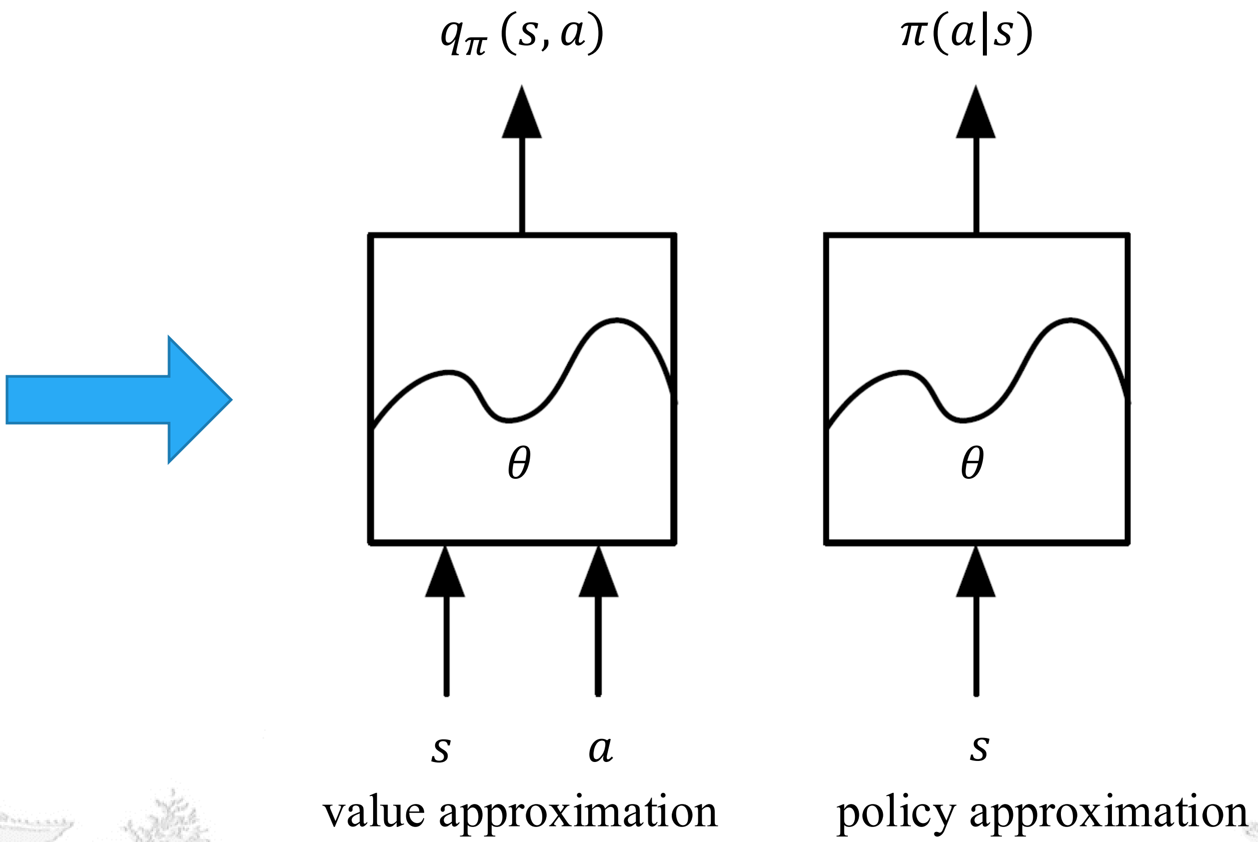
	a_1	a_2
s_1	3.2	4.1
s_2	0.8	1.2
s_3	2.3	-0.3

Q-table

$\pi(a|s)$

	a_1	a_2
s_1	0.2	0.8
s_2	0.4	0.6
s_3	0.9	0.1

Policy table



Deep reinforcement learning: use deep neural networks to approximate the $q_\pi(s,a)$ and $\pi(a|s)$

Deep Reinforcement Learning

- 2012, AlexNet is proposed
- 2013, the first deep reinforcement learning paper was presented at the NIPS 2013

Playing Atari with Deep Reinforcement Learning

Volodymyr Mnih **Koray Kavukcuoglu** **David Silver** **Alex Graves** **Ioannis Antonoglou**

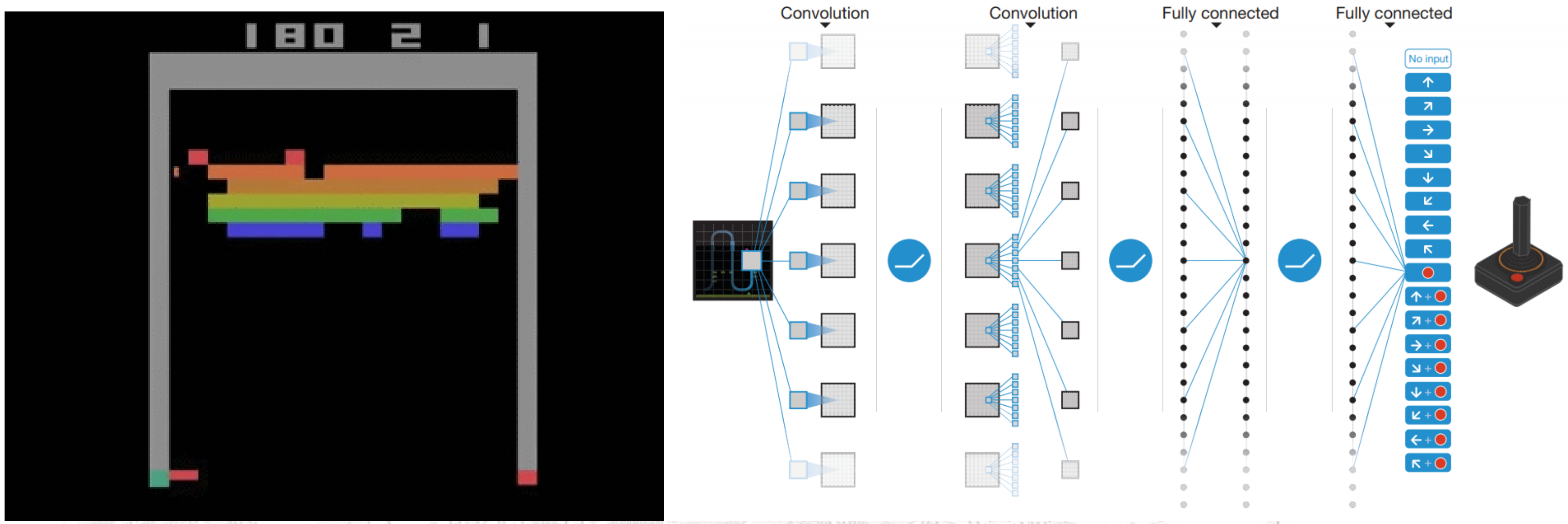
Daan Wierstra **Martin Riedmiller**

DeepMind Technologies

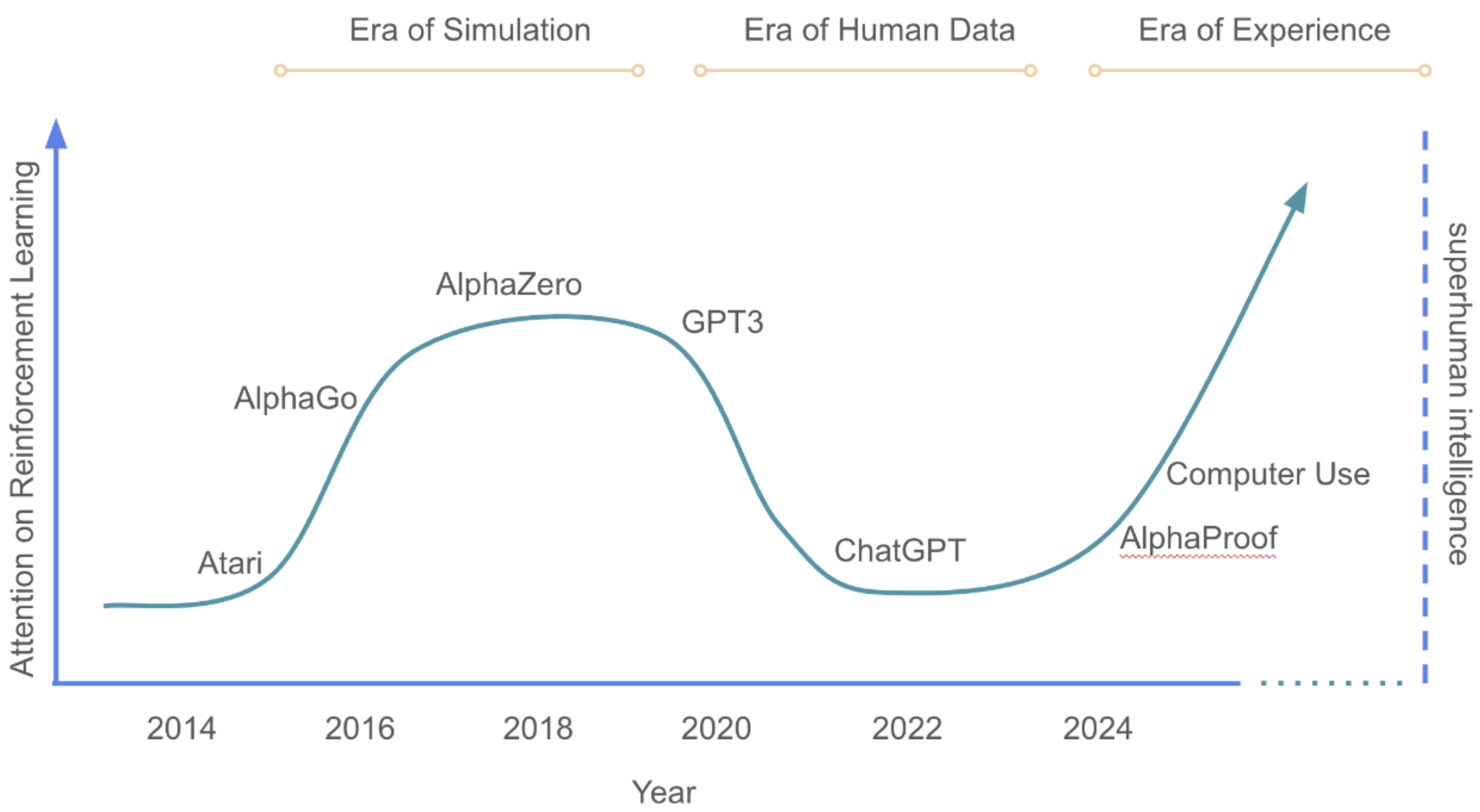
{vlad,koray,david,alex.graves,ioannis,daan,martin.riedmiller} @ deepmind.com

Deep Reinforcement Learning

- Deep reinforcement learning
  - approximate the value or policy using deep neural networks
  - solving complex decision problem end-to-end



Deep Reinforcement Learning



谢 谢！