人工智能笔记

第一讲　人工智能简介与课程概要

一、人工智能流派

① 符号主义：从功能模拟入手，使用符号处理（离散数学）

一组符号集 + 一组操作符号的规则集 ⇒ 专家系统（人写规则）

② 联结主义：从结构设计入手，智能是大量神经元复杂连接后涌现

由数据驱动

感知器 → Hopfiled网络 → 深度学习

③ 行为主义：智能取决于感知与行动，从环境感知得到进化，（具身智能）

从经验中进行能力的持续学习，用问题引导

第二讲　机器学习(一) 线性回归

一、基本概念

机器学习：从数据中学习知识 $\xrightarrow{\text{得到}}$ $f(x)=y$

① · 按问题分类：
- 回归问题：函数输出标量（连续）→ 一堆输入 $\xrightarrow{f}$ 输出
- 分类问题：函数输出一个类别（离散）

② · 按数据标注情况分类：
- 监督学习：所有数据都有标签，得到对应标签
- 半监督学习：部分数据有标签
- 无监督学习：数据无标签，寻找数据规律
- 强化学习：稀疏标签（奖励），序列决策

③ 监督学习：

主要要素：
- 训练数据：样本数据 + 标注信息
- 学习模型：映射函数 $f(x_i)$
- 损失函数：真值与预测值之间差值，记为 Loss.
- 模型优化：损失之和最小，$\min \sum_{i=1}^{n} Loss(f(x_i), y_i)$.

数据集：训练集 - 测试集 - 验证集

· 映射函数 f：$\hat{y_i} = f(x_i) = \hat{w}x_i + \hat{b}\ (1 \le i \le n)$，$\hat{w}$、$\hat{b}$ → 未知.

· 损失函数 Loss → 让 $f(x_i)$ 尽量等于 y

0-1损失函数、平方损失函数、绝对损失函数、对数损失函数 $(-\log P(y_i|x_i))$

二、线性回归：分析不同变量之间存在的关系，模型为线性的 ⇒ 用于预测

① 一元线性回归

训练数据：$D = \{(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)\}$

学习模型：$f(x_i) = wx_i + b$

损失函数：$L(w, b) = \sum_{i=1}^{n} loss(x_i, y_i) = \sum_{i=1}^{n} (y_i - (wx_i + b))^2$ ⇐ 使其最小

(1) 优化方法一：最小二乘法　适合小特征、数据量不大

对 $L(w, b)$ 参数 w、b 分别求偏导，使其导数值为零 $\xrightarrow{\text{解方程}}$ w 与 b

$$① \frac{\partial L(w,b)}{\partial b} = \sum_{i=1}^{n} 2(y_i - wx_i - b)\cdot(-1) = 0 \Rightarrow b = \bar{y} - w\bar{x}$$

$$② \frac{\partial L(w,b)}{\partial w} = \sum_{i=1}^{n} 2(y_i - wx_i - b)(-x_i) = 0 \quad \because b = \bar{y} - w\bar{x}$$

$$\therefore \Rightarrow \sum_{i=1}^{n}(y_i x_i - wx_i x_i - \bar{y}x_i + w\bar{x}x_i) = 0 \Rightarrow w = \frac{\sum_{i=1}^{n} x_i y_i - n\bar{x}\bar{y}}{\sum_{i=1}^{n} x_i x_i - n\bar{x}^2}$$

(2) 优化方法二：梯度下降　适合大数据、高维特征

随机初始化 w^0 $\longrightarrow$ 计算 $\frac{\partial L}{\partial w}\Big|_{w=w^0}$（斜率）$\Rightarrow$ 梯度为负 → 增大 w；梯度为正 → 减小 w（看图像）

变化多少？ $\longrightarrow$ $w^1 \leftarrow w^0 - \eta \frac{\partial L}{\partial w}\Big|_{w=w^0}$，$\eta$：参数：学习率（一次变多大幅度）

$$\begin{cases} w^1 \leftarrow w^0 - \eta \frac{\partial L}{\partial w}\Big|_{w=w^0,\, b=b^0} \\ b^1 \leftarrow b^0 - \eta \frac{\partial L}{\partial b}\Big|_{w=w^0,\, b=b^0} \end{cases}$$ ⇒ 迭代更新

② 多元线性回归

$$\begin{cases} \text{数据向量 } x_i = [x_{i,1}, x_{i,2}, \dots, x_{i,D}] \in R^D \\ f(x_i) = w^T x_i + w_0 \quad (D+1 \text{ 个参数}) \\ L(w) = \sum_{i=1}^{n} (y_i - (w^T x_i + w_0))^2 \end{cases}$$

~~不~~ 一般会扩展数据 x_i 的维度，多一个 1，对应 w_0。

所有训练数据矩阵 $X = \begin{bmatrix} x_1 & \cdots & x_n \\ 1 & \cdots & 1 \end{bmatrix}^T$，$\bar{w} = \begin{bmatrix} w \\ w_0 \end{bmatrix}$

均方差函数变为 $L(w) = (y - Xw)^T(y - Xw)$

向量转置乘自己 → 每个元素平方和 ↗

$L(w)$求导得：$\nabla L(w) = -2X^T(y - Xw)$ 令=0

$\therefore \quad w = (X^TX)^{-1}X^Ty$

第三讲 机器学习(二) 线性分类

一、二分类问题

1. 定义
   - 数据：类别标签为y_i，$y_i \in \{0, 1\}$
   - 模型：线性分类器 $\hat{y}(x) = g(f(x)) = g(w^Tx + b)$ → g根据线性结果分类

2. 线性分类：

$X \to w^Tx + b \to sigmod \to$ 概率 → 分类

① 一般形式：$\hat{y}(x) = g(w^Tx + b)$，$g(z) = \begin{cases} 1 & z > 0 \\ 0.5 & z = 0 \\ 0 & z < 0 \end{cases}$

其中：g为激活函数，$g(w^Tx + b) = 0.5$为决策边界

② 广义线性模型 $\hat{y} = w^Tx + b \Rightarrow g(\hat{y}) = w^Tx + b \Rightarrow \hat{y} = g^{-1}(w^Tx + b)$

意思：对数几率回归是广义线性回归的一种，回归后再分类

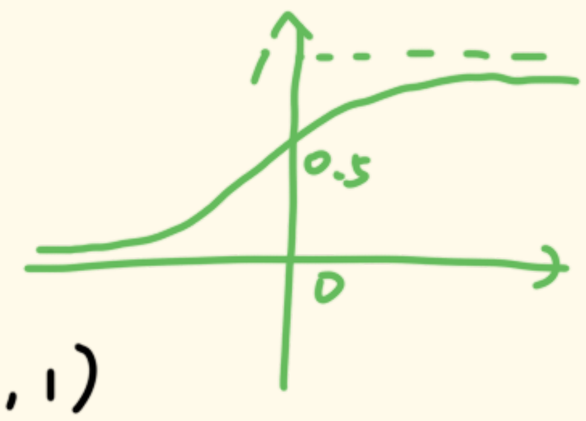
原函数：$\ln\left(\frac{\hat{y}}{1 - \hat{y}}\right) = w^Tx + b$

③ 对数几率回归 (logistic)

引入sigmod函数，令 $z = f(x) = w^Tx + b$

$\hat{y} = g(z) = \frac{1}{1 + e^{-z}}$ ← sigmod函数，$\hat{y} \in (0, 1)$

$= \frac{1}{1 + e^{-(w^Tx + b)}}$



- Sigmod函数特点：$\sigma(z) = \frac{1}{1 + e^{-z}}$ $(-\infty, +\infty) \to (0, 1)$

Odds，事情发生/不发生↑

(0, 0.5)中心对称，反函数 $z = \ln\left(\frac{\sigma}{1 - \sigma}\right)$ 为对数几率.

- 模型输出：后验概率 $f_\theta(x) = p(y = 1 | x) = \frac{1}{1 + e^{-(w^Tx + b)}}$

↳ 知道x后属于$\hat{y}$的概率

∴证明模型含义：$logit(p) = \ln\frac{p}{1 - p} = w^Tx + b$ (线性)

$\hat{y} = p = p(y = 1 | x) > 0.5$ → 证明属于"1"类概率大

- 参数 w、b 怎么学出来？

在训练集上使用极大似然估计 ⇒ 使 $P(y=y_{真}|x)$ 最大

∴ 第 i 个样本上二分类概率：$P(y_i|x_i) = f_\theta(x_i)^{y_i} \cdot (1-f_\theta(x_i))^{1-y_i}$

⇒ 极大似然：所有样本概率相乘，再取 log（好算），加负号

= 分类交叉熵损失：要尽可能小

预测值

损失函数：$L(\theta) = -\sum_{i=1}^{n}[y_i \log f_\theta(x_i) + (1-y_i)\log(1-f_\theta(x_i))]$

真实值

- 优化方法：梯度下降.

对 $L(\theta)$ 的 θ 求偏导：$\frac{\partial L(\theta)}{\partial \theta_j}$，更新 $\theta_j = \theta_j - \eta \frac{\partial L(\theta)}{\partial \theta_j}$（$\eta$：学习率）

$$\Rightarrow \theta_j = \theta_j - \eta \sum_{i=1}^{n}(y_i - f_\theta(x_i))x_i$$

二. 多分类问题

1. Logistic 回归只能处理二分类问题

⇒ 多分类使用 softmax 函数回归

2. 步骤：

① 把真值表示为独热向量 $y = \begin{bmatrix}1\\0\\0\end{bmatrix}$ = 类别 1

② 每一类进行线性打分：z_1, z_2, z_3

⇒ 用 softmax 变为概率：$p(y=k|x) = \frac{e^{z_k}}{\sum_j e^{z_j}}$ ⇒ 选 p 最大结果.

③ 损失函数：$L(\theta) = -\sum_{i=1}^{n}\sum_{j=1}^{K} y_{ij} \log(p(y_{ij}|x_i))$（交叉熵损失）

④ 优化方法：梯度下降

三. 监督学习

1. 两种风险：经验风险：训练集中数据产生损失

期望风险：当测试集中存在无穷多数据产生的损失

（大数定律）一般用经验风险估计期望风险

结构风险：经验风险 + 正则化项 ← 防止过拟合

2. 过拟合(测试集差) ⇓ 模型太复杂 、 欠拟合(都差) ⇓ 模型太简单

3. 正则化：结构风险最小 ⇐ 惩罚模型复杂度(加Loss里)

① L1 正则化 $R(f) = \|w\|_1 = |w_1| + |w_2| + \dots + |w_d|$

权重绝对值之和 → 让没用权重变0 → 降低复杂度

② L2 正则化 $R(f) = \|w\|_2^2 = w_1^2 + w_2^2 + \dots + w_d^2$

让权重整体变小，不一定变0 → 让参数不要过大

第4讲 机器学习(三) 线性判别分析与支持向量机

一、线性判别分析 (Fisher 线性判别)

• 核心思想：把高维数据样本线性投影到一个低维空间，使得满足"类内方差小，类间间隔大"

1. 二分类问题：通过线性函数 $g(x) = w^T x$ 投影到一维空间

两类各自均值：$\begin{cases} \vec{m_1} = \frac{1}{N_1} \sum_{x \in C_1} x \\ \vec{m_2} = \frac{1}{N_2} \sum_{x \in C_2} x \end{cases}$ 投影后 ⟹ $\begin{cases} m_1 = w^T \vec{m_1} \\ m_2 = w^T \vec{m_2} \end{cases}$

2. 描述"分开"程度 ⇓ 最大化 $\|m_2 - m_1\|_2^2$ ⇓ 类间间隔大　最小化 $S_1^2 + S_2^2$ ↓ 类内集中

$$J(w) = \frac{\|w_2 - w_1\|_2^2}{S_1^2 + S_2^2} = \frac{\|w^T(m_2 - m_1)\|_2^2}{w^T \Sigma_1 w + w^T \Sigma_2 w} = \frac{w^T (m_2 - m_1)(m_2 - m_1)^T w}{w^T (\Sigma_1 + \Sigma_2) w} = \frac{w^T S_b w}{w^T S_w w}$$

• S_b：类间散度矩阵：两个类别均值点间分离程度

$$S_b = (m_2 - m_1)(m_2 - m_1)^T$$

S_w：类内散度矩阵：衡量每个类别中数据点的分离程度

$$S_w = \Sigma_1 + \Sigma_2$$

3. 损失函数：令 $w^T S_w w = 1$，$L(w) = w^T S_b w - \lambda(w^T S_w w - 1)$

对w偏导并等于零，得 $S_b w = \lambda S_w w$ 或 $S_w^{-1} S_b w = \lambda w$

- 优化方法：广义特征值分解

二、支持向量机 - 线性分类

1. 背景：画一条直线/超平面 ⇒ 将两类数据区分开

2. 感知器模型：$\hat{y}(x) = g(w^T x) = sign(w^T x)$，$g(z) = \begin{cases} +1, & z \geq 0 \\ -1, & z < 0 \end{cases}$

损失函数：$L(w) = \begin{cases} 0, & y \cdot f(x_i) \geq 0 \\ 1, & y \cdot f(x_i) < 0 \end{cases}$ (直接算错误次数)

优化方法：迭代优化

问题：① 阶跃函数 sign 难以优化

② 有无限多完全正确分类解，不一定最优

3. 支持向量机 (SVM)

① 目的：寻找超平面 $w^T x + b = 0$ 最优，其中 $\begin{cases} w = (w_1 \dots w_n) \text{ 为超平面法向量} \\ b \text{ 偏置项为与原点的距离} \end{cases}$

② 定义：函数间隔：对数据点 (x_i, y_i)，定义为 $\hat{\gamma}_i = y_i (w^T x_i + b)$ (标签{1,-1})

$\begin{cases} \text{分类正确}, & y_i (w^T x_i + b) > 0, \text{ 间隔为正} \\ \text{错误}, & y_i (w^T x_i + b) < 0, \text{ 间隔为负} \end{cases}$

且间隔越大，表示分类的置信度越高 ⇒ 可做指标

几何间隔：对数据点 (x_i, y_i)，$\gamma_i = \dfrac{y_i (w^T x_i + b)}{\|w\|_2}$

($\|w\|_2 = \sqrt{w^T w}$) 真正几何意义的距离

分类间隔边界：中间空最大的两个边界

↓

支持向量：上下间隔边界经过的样本数据点

(决定边界位置)

w, b 同时除以 c ⇒ 分类边界与上下间隔边界 $\begin{cases} w^T x + b = 0 \\ w^T x + b = 1 \\ w^T x + b = -1 \end{cases}$

($w^T = \dfrac{w^T}{c}$, $b = \dfrac{b}{c}$)

上下间隔边界几何距离：

$$\gamma = \frac{|1 - b - (-1 - b)|}{\|w\|_2} = \frac{2}{\|w\|_2}$$

分类间隔：$\gamma = \frac{2}{\|w\|_2}$（上下边界距离）

③ 对超平面约束：$y_i(w^T x_i + b) \geq 1$（比上面范围大，因为有上下边界）

满足等号成立称为支持向量（就是边界上点）

基本形式：找 w 与 b 使得 $\gamma = \frac{2}{\|w\|_2}$ 最大（最小化 $\frac{\|w\|_2}{2}$）

解法（优化方法）：拉格朗日乘子法（带约束优化）

（都得先分对）

$$\min_{w,b} \frac{\|w\|^2}{2} = \min_{w,b} \frac{1}{2} w^T w \quad s.t. \quad y_i(w^T x_i + b) \geq 1, \ i = 1, 2, \ldots, n$$

引入拉格朗日乘子：$\min_{w,b} L(w, b, a) = \frac{1}{2}\|w\|^2 - \sum_{i=1}^{n} a_i (y_i(w^T x_i + b) - 1)$

↓ 求解 a.

④ 线性不可分 $\xrightarrow{(1)}$ 核函数.

把线性不可分样本映射到一个更高维特征空间，变得可分.

↓ 核函数

常见核函数　　暂略

↓

(2) 松弛变量 _ 软间隔与 hinge 损失函数

软间隔：允许错分部分训练样本（异常点等）

松弛变量 ξ_i：$y_i(w^T x_i + b) \geq 1 - \xi_i \quad (\xi_i \geq 0)$

Hinge 损失函数：$\min_{w,b} \frac{1}{2} w^T w + C \sum_i \xi_i = \min_{w,b} \frac{1}{2} w^T w + C \sum_i \max(0, 1 - y_i(w^T x_i + b))$

↓

替换为任意其他损失函数：$\min_f \Omega(f) + C \sum_{i=1}^{n} l(f(x_i), y_i)$

$\Omega(f)$：结构风险（正则项），描述复杂度

$\sum_{i=1}^{n} l(f(x_i), y_i)$：经验风险，描述 f 与训练数据契合度

线性模型	激活函数	损失/目标函数	损失/目标函数定义	优化方法
线性回归	-	平方损失	$(y_i - \boldsymbol{w}^T \boldsymbol{x_i})^2$	最小二乘、梯度下降
对数几率回归	$sigmoid(\boldsymbol{w}^T \boldsymbol{x})$	二值交叉熵损失	$-y_i \log \sigma(\boldsymbol{w}^T \boldsymbol{x_i}) + (1 - y_i)\log(1 - \sigma(\boldsymbol{w}^T \boldsymbol{x_i}))$	梯度下降
Softmax分类	$softmax(\boldsymbol{W}^T \boldsymbol{x})$	交叉熵损失	$-y_i \log softmax(\boldsymbol{w}^T \boldsymbol{x_i})$	梯度下降
线性判别分析	-	Fisher准则	$\frac{\|m_2 - m_1\|_2^2}{s_1^2 + s_2^2}$	广义特征值分解
感知器	$sign(\boldsymbol{w}^T \boldsymbol{x})$	0-1损失	$\begin{cases} 0, y \cdot f(\boldsymbol{x_i}) \geq 0 \\ 1, y \cdot f(\boldsymbol{x_i}) < 0 \end{cases}$	迭代优化
支持向量机	$sign(\boldsymbol{w}^T \boldsymbol{x})$	Hinge损失	$\max(0, 1 - y_i \boldsymbol{w}^T \boldsymbol{x_i})$	二次规划、SMO等

模型	相对于线性回归，为分类做的关键优化
对数几率回归	加 Sigmoid 输出概率；用交叉熵损失；稳健可解释
感知器	阶跃函数硬分类；只优化分错样本
支持向量机	最大化分类间隔；关注支持向量；软间隔更鲁棒
线性判别分析	最优投影方向；类间大 / 类内小；天然多分类 + 降维

第五讲 机器学习(四) 决策树和集成学习

一、决策树

1. 符号定义

- 训练集 $D=\{(x_1,y_1),\dots (x_m,y_m)\}$ x为d维属性向量，y为标签
- 属性集合 $A=\{a_1,a_2,\dots,a_d\}$ 共d个属性 ⟨离散属性 / 连续属性
- 标签空间 $Y=\{c_1,c_2,\dots,c_k\}$

2. 决策树定义：树型结构，节点+有向边组成

- 节点：⟨内部节点：属性/特征；叶节点：一种类别
- 有向边/分支：代表一个判断

几何特征：把决策空间划分成不相交的单元或区域.

3. 分类训练

目的：提高纯度.

① 选一个属性作为根节点

② 如果一个叶节点都是一类 → 标记为叶节点

③ 否则，选择一个没出现到的属性来分叉.

属性 ⟨离散属性：按类别分类；连续属性：取最优分类点（目标函数最大分割点）

which one?

4. 评价指标：（越低越好）

信息熵：离散随机变量Y概率分布为 $P(Y=C_k)$，信息熵为：

$$H(Y)=-\sum_{k=1}^{K}P(Y=C_k)\log_2 P(Y=C_k)$$

信息熵↑不确定性↑

（概率 × 信息量(-log)）

P越低，信息量越大.

经验信息熵：信息熵中用频率表示概率

条件熵：$H(Y|a)$ 表示在已知变量a的条件下，Y的不确定性

$$H(Y|a)=-\sum_{v=1}^{V}P(a=a^v)\sum_{k=1}^{K}P(Y=C_k|a=a^v)\log_2 P(Y=C_k|a=a^v)$$

（v为属性选项）

① 信息增益：划分后熵的变化. $IG(D,a)=H(Y)-H(Y|a)$

↳越大越好. H(Y)：划分前；H(Y|a)：划分后

(ID3) → 缺陷：变量多的属性更易被选择.

⇓ so.

② 信息增益率 (C4.5)

$$Gain\_ratio(D,a)=\frac{H(Y)-H(Y|a)}{IV(a)}$$

IV(a)为属性a固有值 $IV(a)=-\sum_{v=1}^{V}P(a=a^v)\log_2 P(a=a^v)$

(属性a可取值越多，IV(a)值会越大)

③基尼系数(CART)　基尼值：从数据集D中抽取两个样本，分类不一致概率

$$Gini(D) = \sum_{k=1}^{K}\sum_{k'\neq k} p_k p_{k'} = 1 - \sum_{k=1}^{K} p_k^2$$

$$Gini\_index(D,a) = \sum_{v=1}^{V} P(a=a^v)\, Gini(a=a^v)$$

5. 剪枝：防止"过拟合"与噪音，提升泛化(上面难)

↳ 主动去掉一些分支．"子树变叶"　↳ 预剪枝 / 后剪枝

二、集成学习．

1. 定义：复杂的分类任务，分解为若干子任务，组合分类器

目标：集成个体应该"好而不同"

2. 方法：自适应提升法 Adaboost　↳ 样本权重 / 分类器权重　⇒ 两点核心

① 初始化训练样本权值分布，给相同权重

② 训练弱分类器，样本错则+样本权重(反之…)

③ 所有弱分类器组合为强分类器，看误差分分类器权重

第六讲　深度学习(一)　前馈神经网络与模型训练策略

一、前馈神经网络 ⟶ 也建立在线性模型基础之上

1. 核心思想：先做线性变换，再加非线性激活函数 ⇒ 组合多个单元

$$神经元输出 = \sigma(w^T x + b)$$

↳ σ：激活函数　(不能为线性，因为线性叠加还是线性)

2. 非线性回归

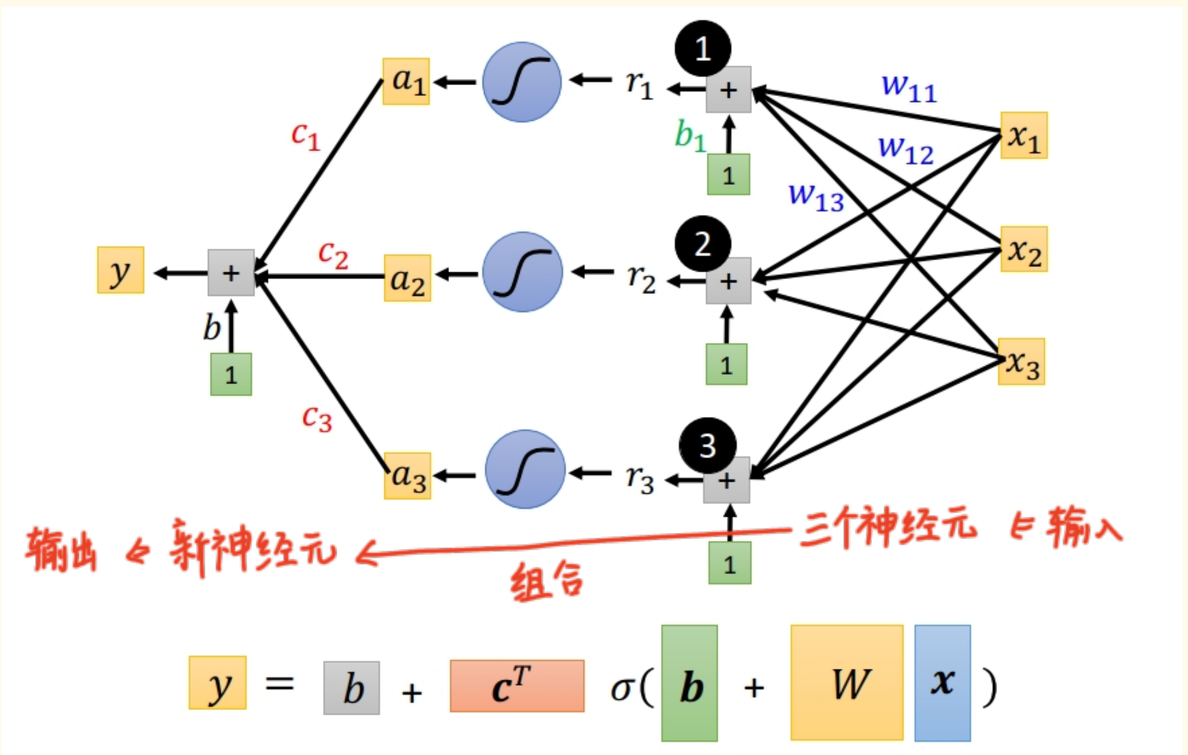
① 多元线性回归：$y = b + \sum_j w_j x_j$

⇓

组合多个神经元：$y = b + \sum_j c_i \cdot sigmod(b_i + \sum_j w_j x_j)$ ← 套3-层新的

⇓

② 矩阵形式：$y = \underbrace{b}_{参数} + c^T \cdot \overbrace{\sigma}^{激活函数}(\vec{b} + W\vec{x})$　(学习模型)



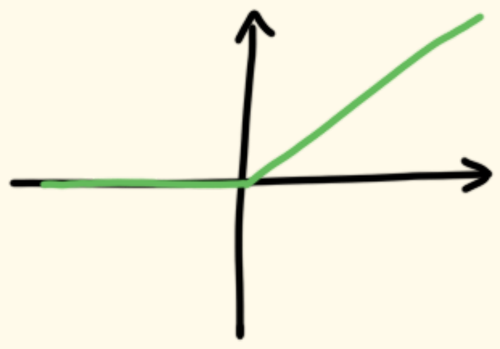
② 损失函数：$L(\theta) = \sum_{i=1}^{n} (y_i - f(x_i))^2$

优化方法：梯度下降法　　$b, c_i, b_i, w_{ij} \leftarrow$ 变量 $- \eta \frac{\partial L}{\partial \text{变量}}$

3. 如何改进？→ (1) 改变激活函数 σ

① Sigmoid：$\sigma(x) = \frac{1}{1+e^{-x}}$

② ReLU：$ReLU(x) = \max(0, x)$



神经元数量↑，模型表达能力↑，误差↓

(2) 增加模型层数

$$f(x) = b + c^T \sigma(b + Wx) \Rightarrow f(x) = b + c^T \sigma(b' + W' \sigma(b + Wx))$$

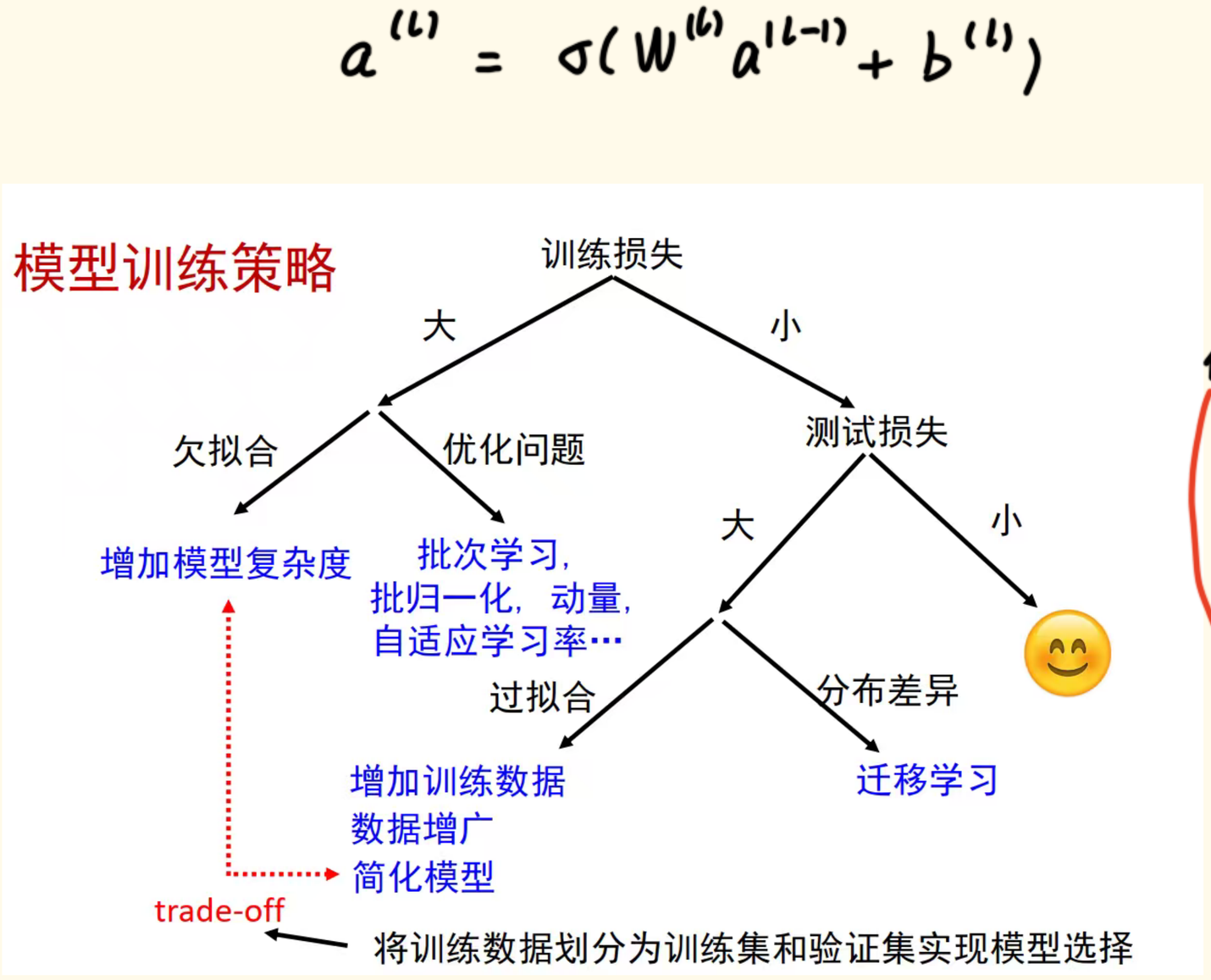
4. 定义：由输入层、输出层和至少一层隐藏层构成，前一层传给下一层

$x \longrightarrow$ 隐藏层1 $\rightarrow$ 隐藏层2 $\rightarrow \dots \rightarrow y$

$$a^{(l)} = \sigma(W^{(l)} a^{(l-1)} + b^{(l)})$$

也叫：全连接网络 / 多层感知机

二、



欠拟合：模型太简单，表达力不够

优化问题：模型够复杂，但训练没训好

区分：增加神经元、更多层 → 没用

过拟合：泛化性弱

eg. "局部最优" ≠ 全局最优

第七讲 深度学习(二) 卷积神经网络和自注意力机制

一、卷积神经网络 CNN

1. 引入： 图片直接全连接 → 参数量过大 → ∴无需覆盖整图"提取特征"

2. 两个核心思想： ①局部感受野： 让神经元只能看局部区域
②参数共享： 训练一个filter，让它在整张图上滑动，参数不变，用于判断某个特征

3. 基础概念：①通道 channel： 图像的层数 / 不同维度上信息表示
eg. RGB三通道 、中间层：特征通道

②感受野： 神经元看到图像的实际一部分（覆盖所有通道）
↳ 避免整张图片计算，使参数过大

③卷积核 kernel / 滤波器 filter： 一组可以学习的权重
· 把感受野与filter相乘、相加，得到卷积结果
↳ 参数共享：整个图用同一个filter.

④特征图 feature map（输出通道）：一个filter在整张图所有输出"相似程度"

⑤步长 stride： 卷积核每次移动几格.

⑥填充 padding： 图片边缘补一圈0，让边缘也能计算，控制输出
eg. 输入 6×6 —3×3→ 输出只有 4×4

⑦重叠 overlap： 两次相邻卷积看到区域有重合（stride小）

4. 参数量：
$$k_h \times k_w \times C_{in} \times C_{out} + C_{out}$$
(k_h：卷积核高度，k_w：卷积核宽度，C_{in}：输入通道数，C_{out}：输出通道数，C_{out}：(bias))

5. 多层卷积： 使得小卷积核也能看大范围 ⟶ 有效感受野扩大，捕大pattern

6. 池化 Pooling： 对像素降采样（缩小尺寸）不影响识别语义

CNN结构： 卷积 → 池化 → 卷积 → 池化 → Flattern → 全连接 → Softmax
（矩阵拉为向量）（神经网络）（计算概率）

7. CNN应用： 图像分类，语义分割，物体检测，AlphaGo
不足： 普通卷积操作不满足尺寸与旋转不变性（数据增广）

二、序列数据模型

1. 定义：数据变为3-组向量集　　一句话、一个图像、一个图结构、一段语音

输出形式：　一对一(n-n)　　多对一(n-1)　　多对多(n-m)

2. 上下文信息：每个位置一个FC(fully-connected)，考虑邻域信息

局部上下文：窗口机制(左右n个词)

全局上下文：↘

三、自注意力机制 Self-attention

1. 核心思路：让序列中每个位置都从整个序列找与自己相关的信息，然后生成带上下文新表示

步骤：①计算相关性分数　②把分数变成attention weight

③用attention weight对信息加权求和

2. Q、K、V：每个输入向量 a_i 变成三个向量：

$$\begin{cases} q_i = W_q a_i \\ k_i = W_k a_i \\ v_i = W_v a_i \end{cases}$$

$\begin{cases} q_i: \text{query}，\text{用于主动查找相关信息} \\ k_i: \text{key}，\text{用于被别人匹配} \\ v_i: \text{value}，\text{真正被加权求和的信息} \end{cases}$

计算：①用query与key算相关性分数　$\alpha_{ij} = q_i \cdot k_j$

②softmax计算概率：　$\alpha'_{i,j} = \dfrac{\exp(\alpha_{ij})}{\sum_j \exp(\alpha_{ij})}$

③与v加权求和：　$b_i = \sum_j \alpha'_{ij} v_j$

3. Self-attention矩阵形式

整个输入向量(放一起)记为矩阵 I $\Rightarrow$ $\begin{cases} Q = W_q \cdot I \\ K = W_k \cdot I \\ V = W_v \cdot I \end{cases}$

$\Rightarrow$ 相关性矩阵 $A = k^T Q$ $\Rightarrow$ 做softmax得 A' $\Rightarrow$ $O = A'V$

(W_q、W_k、W_v：参数，训练得来)

4. 多头注意力机制：学多种相关性

同一个输入，用多套Q、K、V来计算，不同head关注不同相关性

$\Rightarrow$ 最后合并多个head结果，经过一个 W_o 得到输出

5. 位置编码：每个位置赋一个唯一位置编码向量

6. 比较： RNN：顺序处理，难并行，长距离依赖难
CNN：适合数据少的任务

7. 训练： 随机初始化 → 反向传播 → 梯度下降 → 更新参数

第八讲 深度学习(三) Transformer 与 Bert

一、Transformer

1. 核心步骤： Tokenization → Input Layer → Attention → Feed Forward → Output Layer
文本转 Token　文本变向量　理解上下文　整合信息　得到输出

① Tokenization 文字切为Token → 对应数字id （统计，非训练）

② Token Embedding：让Token有语义，每个Token分配向量
·意思相近token → 相近embedding ； 普通embedding不考虑上下文

③ Positional Embedding：位置向量，训练得到
输入表示：Token Em… + Positional Em…

2. ④ Encoder：编码器，读入输入队列，把token变为带上下文的表示
Transformer Encoder 用 self-attention
主要包含 ① self-attention ② Feed Forward：每个位置进一步非线性变换
③ Residual Connection 残差连接： 原输入加回输出 → 深层网络稳定

原本 $a \xrightarrow{\text{selfatt}} b$ ⇒ $a \longrightarrow a+b$

④ Layer Norm 归一化： 向量标准化 $x_i' = \frac{x_i - m}{\sigma}$

3. ⑤ Decoder：解码器，负责生成输出
(1) 自回归AT：一次生成一个Token，依赖之前生成的
使用 Masked（遮住未来位置） Self-Attention
(2) 非自回归NAT：一次并行生成多个Token.
确定输出长度：长度预测器 / 多生成点再截

4. ⑥ Stop Token：单独一个token eg. END.

5. Cross-Attention： Decoder 生成时看 Encoder
Decoder 拿自己 query，去 Encoder 查 key. value.

6. 训练： 输出概率分布，与真实 Token 算交叉熵损失
·Decoder 每次都拿真实前缀做输入，不用自己算错的. → 错误传播

二、BERT：双向编码器，擅长理解类任务 ←（看左右两边）

1. 自监督学习：不需要人工标签
2. 预训练任务一：掩码语言模型(MLM)，随机遮住一些Token让预测
预训练任务二：下一句预测NSP，给两个句子：AB，让模型判断B是不是A下一句

3. 微调 Fine-tune. : 少量标注数据适配下游任务.

eg. { 输入一个序列 → 输出一个类别
输入一个序列 → 输出同样长度标签
输入两个序列 → 输出一个类别

三、 GPT : 自回归生成式语言模型 , 适合生成

GPT用的是 Decode 结构 Masked Self-Attention + Feed Forward

第九章 深度学习(四) GPT与智能体

一、 GPT

1. In-context Learning: 上下文学习

不更新模型参数, 只在prompt 中给示例 ⇒ GPT找规律、模仿的能力

分:小样本学习、单样本学习、零样本学习

2. ChatGPT ① 自监督预训练 : 基础语言能力
② 有监督微调 : 人工构造高质量回答
③ 强化学习微调 : 对回答评价 → 生成更有价值

二、 AI Agent

1. 组成: LLM + 工具 + 记忆 + 任务管理 + 规划 + 执行环境
↓
仍不变, 可能多输出工具调用指令

2. Context Window 、 System Prompt 、多轮对话

3. 工具调用 : System Prompt 中给出工具与用途 → 模型输出工具调用
↓ → 本地执行(得到响应) → 模型根据响应回答/决策→...
风险防御: ① 提示词限制 ② 框架中对指令检查

4. SKILL : 是工具标准操作程序 SOP → 索引在 SKILL.md.
多步骤调用流程

5. 记忆系统 : 外部文件/数据库 , 分长期记忆/原始记忆
模型自行判断读/写, 以及哪种 ↗

6. RAG 记忆召回：对记忆做 RAG（检索增强生成）

↳ 每次只找相似度最高几个

7. 上下文压缩 Compact

第十章 生成式人工智能(一) 自编码器、变分自编码器

一、生成式人工智能

概念：让机器产生复杂有结构的对象（深度学习）

文章、图像、语音都能生成

二、自编码器 Auto-Encoder (AE)

1. PCA 主成分分析：特征降维方法 → 高维投影到低维

投影方向：方差最大的方向 → 数据蕴含信息多

① 推导：原本高维：$z = Wx$ ⟶ 降维：$z_1 = (w^1)^T x$

映射到 w_1 方向，得新特征 z_1 ⇒ z_1 方差要尽量大

∴ $Var(z_1) = \frac{1}{N}\sum_{z_1}(z_1 - \bar{z}_1)^2$ 同时规定 $\|w^1\|_2 = 1$ (若多维，不同方向需正交：$(w^1)^T w^2 = 0$)

② 另一个角度：压缩后还能重构. eg. 一张手写图片 → 拉平成一个很长的像素向量 x

而 x 可用多个成分线性组合：$x \approx c_1 w^1 + c_2 w^2 + \dots + c_k w^k + \bar{x}$

⇒ 用最少主成分尽可能重构原始数据 ∴ 最小化重构误差 ≈ 最大化投影方差

$x - \bar{x} \approx c_1 w^1 + c_2 w^2 + \dots + c_k w^k = \hat{x}$ ⇒ 最小化 $L(w) = \sum_{i=1}^{N} \left\| (x_i - \bar{x}) - \sum_{k=1}^{K} c_k w^k \right\|_2^2$

$x - \bar{x}$：原始图片 - 平均；$\hat{x}$：重构图；$L(w)$：重构误差

③ PCA可看作单隐层神经网络：先编码：高维压缩 $x \to c$ 解码：还原原图 $c \to \hat{x}$

↳ 但全是线性的（≠自编码器）

2. 自编码器：无标注图像 → Encoder（压缩） → Embedding → Decoder（还原） → 重建图像

- 目标：输出接近输入 $\hat{x} \approx x$
- 为什么有效：模型压缩/还原时必须学会哪些信息最有效

3. 去噪自编码器：输入中加噪，目标：输出还原出原图 → 鲁棒性更强

4. 若用AE生成图片：　若随机向量 → Decoder → 生成图像

问题：这个向量很可能不在隐空间中（没训练过）→ 生成结果不好.

三、变分自编码器 VAE

1. 生成模型：　条件生成　z + condition → G → image

z：随机细节　condition：控制生成内容

核心：应学习一个分布，在给定条件下所有可能图像概率分布（多样性）

↳ 隐空间

2. VAE：　x → Encoder → 一个分布 → 从分布采样code → Decoder → $\hat{x}$

Encoder 输出 $\vec{\mu}$：均值，σ：方差

↳ $c_i = \exp(\sigma_i) \times e_i + \mu_i$

c_i：最终隐变量　e_i：随机噪声

① 采样：把随机性拆出来 → 利于反向传播 → 可以训练

② 训练目标：　(1) 重构损失：　$\hat{x} \approx x$

(2) KL散度：　希望隐变量分布 ~ 标准正态分布 $N(0, I)$

⇒ code空间规整 + 随机采样不易空洞

③ 难点：　权衡两个目标权重

第十一讲　生成式人工智能(二)　变分自编码器、扩散模型、生成对抗网络

一、扩散模型 Diffusion Model

1. 主要思想：　训练：真实图片反复加噪，直到变成纯噪声

生成：从纯噪声开始，一步步去噪，最终得到图像

2. DDPM去噪扩散概率模型：　分为前向过程与反向生成

① 前向过程：加噪　　原干净图片 x_0，随机采样噪声 $\epsilon \sim N(0, I)$，

选择时间步 t（噪声程度），得 $x_t = \sqrt{\bar{\alpha}_t} x_0 + \sqrt{1-\bar{\alpha}_t} \cdot \epsilon$

② 训练Noise Predictor

- 训练目标：让模型预测噪声　　x_t、t、条件文本 y ⟶ 预测噪声 ϵ_θ

训练损失 $\|\epsilon - \epsilon_\theta(x_t, t, y)\|^2$　　ϵ：训练时填实噪声

③ 推理过程：(1) 标准正态分布采样一个纯噪声图　(2) 输入Noise Predictor.

(3) 预测当前噪声，减掉得 x_{t-1}　(4) 重复 t 次 → 干净图像 x_0

3. 统一框架：　文本 → 文本编码器 → 中间产物 → 生成式模型

① Stable Diffusion　中间用压缩(latent)随机噪声 → 去噪 → Decoder

② DALL-E　自回归 + 扩散

③ Imagen　强文本编码器

二、生成对抗网络 GAN

1. 两部分组成：生成器 + 判别器

① 训练判别器：分辨真实/生成图像　⤵迭代

② 训练生成器：输入随机噪声，生成假图像，目标：骗过判别器

2. 常加在别的模型旁边辅助

三、生成式模型评价

1. FID：把真实、生成图像放进一个 CNN → 提取特征 → 计算分布距离

问题：需要大量样本

2. CLIP Score：看图像与 prompt 符合程度

第十二讲　强化学习(一) 基于策略的强化学习

一、强化学习定义：通过从交互中学习来实现目标

- 让模型在环境中不断试错，通过奖励信号学习策略，使累积奖励最大

动作 = f(状态) → 动作作用于环境 → 返回奖励

1. 三个方面：感知、行动、目标

2. 概念：
   - 状态：智能体当前看到的信息
   - 动作：智能体能干什么
   - 奖励：环境给智能体的反馈
   - 环境：智能体以外的一切

3. 系统要素：① 策略 Policy：状态到动作的映射
   - 确定性策略：$a = \pi(s)$
   - 随机策略：$\pi(a|s) = P(A_t = a \mid S_t = s)$

② 奖励 reward：一步反馈 $R(s, a)$

回报 return：从某一时刻开始累计奖励
- 无折扣回报：$G_t = R_{t+1} + R_{t+2} + \dots$
- 折扣回报：$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots$　$(0 \le \gamma \le 1$，折扣因子$)$

③ 价值函数：未来累积奖励的期望：$V^\pi(s) = E_\pi[G_t \mid S_t = s]$

递归形式：$V^\pi(s) = E_\pi[R_{t+1} + \gamma V^\pi(s') \mid S_t = s]$

④ 环境模型 Model：
- 状态转移概率：$P_{ss'}^a = P(S_{t+1} = s' \mid S_t = s, A_t = a)$
- 奖励函数：$R_s^a = E[R_{t+1} \mid S_t = s, A_t = a]$

4. 马尔科夫　① 性质：$P(S_{t+1}|S_t) = P(S_{t+1}|S_1,\dots,S_t)$

② 马尔科夫决策过程　$MDP = \{S, A, P_{sa}, \gamma, R\}$

5. 分类：
- 基于策略 $\pi_\theta(a|s)$　学习状态下该做什么
- 基于价值 $V(s)$, $Q(s,a)$　间接
- Actor-Critic：同时学习

二、基于策略的强化学习

1. 步骤：① step 1 定义策略网络 π_θ　状态 → 输出每个动作分数或概率.

② step 2 定义轨迹和回报
- 轨迹：初始至结束　$T = (s_0, a_0, r_1, s_1, a_1, r_2 \dots)$
- 回报：累计奖励　$G(T) = \sum_t r_t$

目标：最大化期望回报　$E_{T\sim P_\theta(T)}[G(T)]$

2. 如何学习 Actor？

交叉熵损失　$e = -\log \pi_\theta(\hat{a}|s)$　← 要执行，减小 e；不执行，最大化 e

$$L = -\sum_t A_t \cdot \log \pi_\theta(a_t|s_t)$$

A_t：状态下动作好坏程度 (+-)　↑ 权重

① Version 0　即时奖励做权重　$A_t = r_{t+1}$　← 短视

② Version 1　未来累积奖励　$A_t = G_t = r_{t+1} + r_{t+2} + \dots$　没区分贡献

③ Version 2　折扣回报　$A_t = G_t^\gamma = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \dots$　$(\gamma<1)$

④ Version 3　加入 baseline　$A_t = G_t^\gamma - b$

最终目标函数　$L(\theta) = -E_{T\sim p_\theta(T)}\left[\sum_{t=0}^{T} A_t \log \pi_\theta(a_t|s_t)\right]$

3. 算法流程：初始化策略网络 π_θ ⇒ 迭代训练：① 用当前策略 π_θ 与环境交互　② 获得训练数据 s0, a0…　③ 计算 A0, A1…　④ 计算损失 L → 梯度下降

分类
- On-policy：用当前策略采样（每次重新采样）
- Off-policy：用旧策略采样
  - ⇓ PPO 限制策略更新幅度，让旧数据可靠

第十三讲　强化学习（二）基于价值的强化学习.

一、基于价值强化学习.

1. Critic 评判器

状态价值函数 $V^\pi(s)$：评价状态 s 的好坏，$V^\pi(s) = E_\pi[G_t|S_t=s]$

动作价值函数 $Q^\pi(s,a)$：评价状态下做动作好坏，$Q^\pi(s,a) = E_\pi[G_t|S_t=s, A_t=a]$

V 与 Q 关系：$V^\pi(s) = \sum_{a\in A} \pi(a|s) Q^\pi(s,a)$

2. 贝尔曼方程： 递推关系

状态价值： $V^{\pi}(s) = E_{\pi}[r_{t+1} + \gamma V^{\pi}(s') | S_t = s]$

动作价值： $Q^{\pi}(s,a) = E_{\pi}[r_{t+1} + \gamma Q^{\pi}(s', a') | S_t = s, A_t = a]$

3. 基于价值的强化学习 ： 策略评估 + 策略优化

4. 策略评估 $V^{\pi}(s)$ 或 $Q^{\pi}(s,a)$

① 动态规划 DP ： 需知道环境模型 { 状态转移概率 $P(s'|s,a)$; 奖励函数 $r(s,a,s')$ } ⇒ 迭代

$$Q^{\pi}(s,a) = \sum_{s'} P(s'|s,a)[r(s,a,s') + \gamma V^{\pi}(s')]$$ 需环境

② 蒙特卡洛 MC ： 策略 π 在环境中跑很多轨迹，统计回报 G_i ： $V^{\pi}(s) \leftarrow \frac{1}{N}\sum_{i=1}^{N} G_i$ 慢 可能偏离

③ 时序差分 TD: 每走一步更新一次（贝尔曼方程）

$$V^{\pi}(s) \leftarrow V^{\pi}(s) + \alpha[r + \gamma V^{\pi}(s') - V^{\pi}(s)]$$

$[V^{\pi}(s) = r + \gamma V^{\pi}(s')]$ 一着应接近，不接近则修正 $V(s)$

5. 策略优化 已有 $Q^{\pi}(s,a)$ ⟶ $\pi'(s) = \arg\max_a Q^{\pi}(s,a)$ 选最高的（策略优化定理）

6. Q-Learning ↓ off-policy 目标：学一个动作价值函数 $Q(s,a)$，用其选动作 $a = \arg\max_a Q(s,a)$

$$Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma \max Q(s',a') - Q(s,a)]$$ （相似TD）

当前奖励 下个状态最好未来价值

① 技巧1： Target Network ∵ Q进化中两边是同一个网络，当前Q负责更新，目标Q固定

② 技巧2： Exploration 探索： 一开始Q不准确 → 选Q最大动作并不一定好 → 要探索

常见方法： ε-greedy ： 大多数时候选Q最大动作，少数时候随机选

③ 技巧3： Replay Buffer 经验回收 ： 把经验都存起来，训练时随机抽一批

二、Actor-Critic 策略 + 价值

1. 策略里： $A_t = G_t^{\gamma} - b$ ⟶ baseline 固定 × 不合理 → 应与状态有关

Version 3.5 $A_t = G_t^{\gamma} - V^{\pi}(S_t)$ → 平均回报水平

2. { Actor ： 负责选择动作 ; Critic ： 评价状态、动作好坏

3. Version 4 上一个里面 G_t^{γ} 要等结束才知道 ⇒ G_t^{γ} 近似为 $r_{t+1} + \gamma V^{\pi}(S_{t+1})$

$$\therefore \quad A_t = r_{t+1} + \gamma V^{\pi}(S_{t+1}) - V^{\pi}(S_t)$$

三、其他强化学习算法

1. Reward Shaping 奖励塑形　　设置中间步骤奖励

2. 模仿学习：专家示范　$s \to a$　⇒ 会复制无关动作

3. 逆强化学习：从专家学习中训练奖励函数