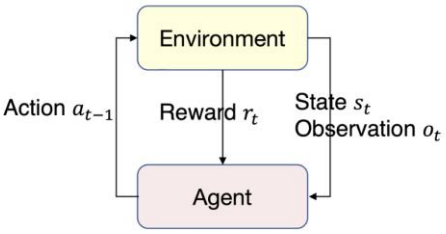
机器学习算法模型概述

- **监督学习（supervised learning）**：在学习过程中有监督，给一个输入数据，告诉了应该正确的是什么输出（有标签）；学习数据到标签的一个映射
  - 分类问题（classification）
    - 决策树（线性边界）【总是在沿着特征做切分】
    - 随机森林【随机选取不同的 feature 和 training sample，生成大量的决策树，然后综合这些决策树的结果来进行最终的分类】
    - 朴素贝叶斯分类（多分类，只能处理离散数据，连续数据需要离散化）【根据条件概率计算待判断点的类型】
    - KNN（K 近邻算法）（可非线性分类）【对于待判断的点，找到离它最近的几个数据点，根据它们的类型决定待判断点的类型】
    - 感知机（线性分类器）
    - 逻辑回归（线性分类器，在感知机分类器基础上增加了 sigmoid，二分类）【衍生出的 softmax 可以用于多分类】
    - SVM 支持向量机（线性分类器）【找到不同类别之间的分界面，使得两类样本尽量落在面的两边，而且离分界面尽量远】【对于线性不可分的情况，通过使用非线性映射算法将低维输入空间线性不可分的样本转化为高维特征空间使其线性可分】
    - MLP 多层感知机（非线性分类器）【多层感知机只需要一层隐含层（足够多的隐含单元），就可以拟合任意的非线性函数】
  - 回归问题（regression）
    - 线性回归（损失函数均方误差 MSE）
  - 生成（generation）
- **非监督学习（unsupervised learning）**：在学习过程中只需要原始数据（无标签），可以从数据当中发现其本身蕴含的模式和规律（特征分析）
  - 聚类（clustering）
    - K-means
  - 概率估计（probability estimation）：异常检测，数据生成
  - 数据生成（data generation）
  - 降维与特征选择（dimension reduction and feature selection）
    - 主成分分析法 PCA
    - 自编码器 Auto-encoder（非线性）
  - 主题模型（Topic Modeling）
  - 异常检测（Outlier Detection）
- **强化学习（reinforcement learning）**：没有老师，每次做完只会给一个分数，不会告诉标准答案；从与环境的交互当中学习，目标是通过不断优化看到 state 之后反馈 action 的策略，最大化环境给的 reward【适用于状态很多无法枚举遍历】
  - 五大部件：agent（学习体，要学习的对象）；action（选答案的行为）；environment（打分的）；reward（答对之后的奖励）；state observation（环境给学习体的 state）

- Learning from **interacting** with an **environment.**



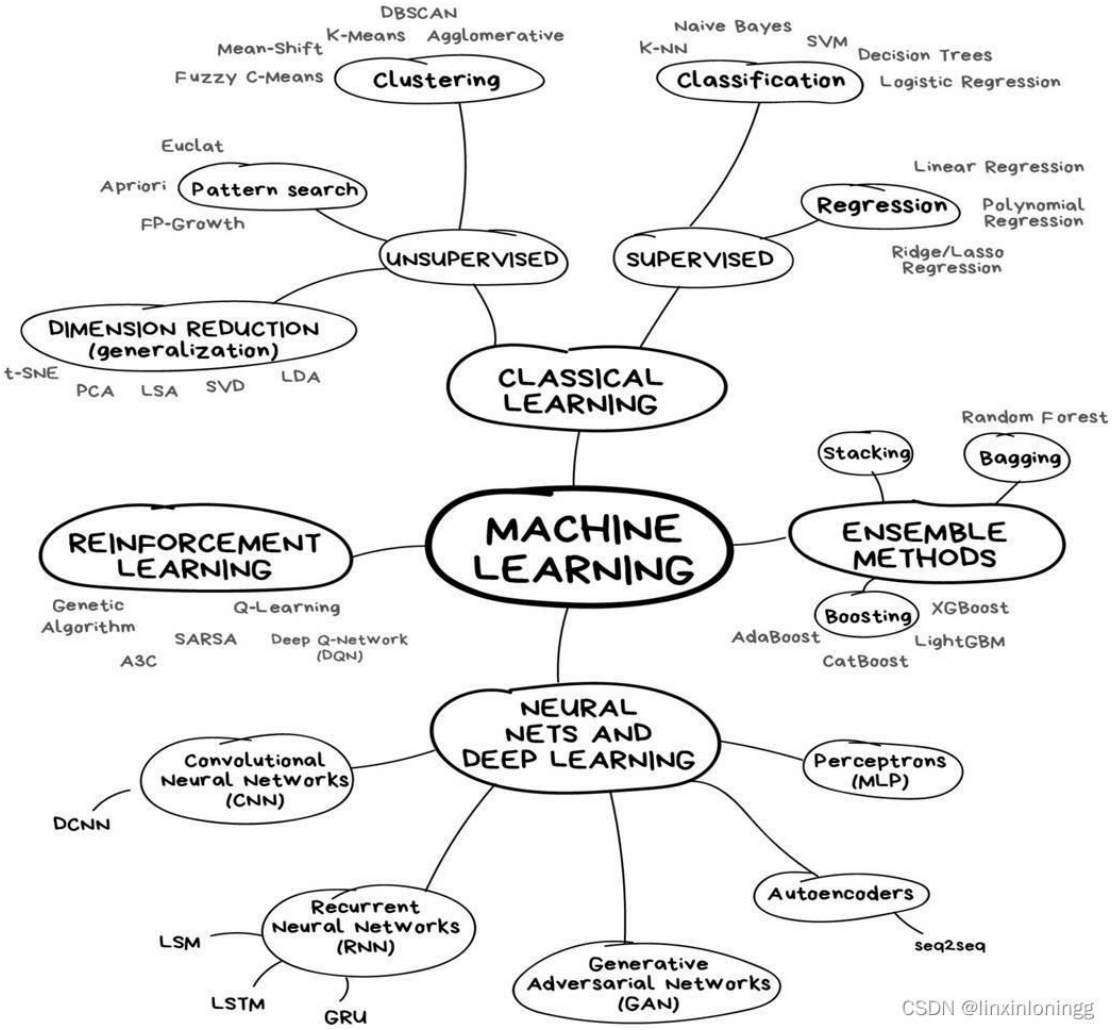
- ❑ a set of environment states S;
- ❑ a set of actions A;
- ❑ rules of transitioning between states;
- ❑ rules that determine the scalar immediate reward of a transition; and
- ❑ rules that describe what the agent observes.

Learning a mapping from states to actions to maximize long-term reward.

Goal: maximize the expected long-term payoff

Example: graded examinations with only overall scores but no correct answers

例子如 AlphaGo，机器人对控制逻辑的学习



Deep Learning

- Word embedding
- DNN（Deep Neural Networks）
- RNN（Recurrent neural networks 循环神经网络）

- LSTM（Long Short Term Memory）
- Transformer
  - Attention
- CNN （Convolutional neural networks 卷积神经网络）
- CV
  - 图像分割（Image segmentation）
  - 目标检测（Object detection）
  - 图像注解（Image Captioning）
- GAN 生成对抗网络

系统设计题

1. 定义问题
   明确你想要解决的问题是什么。这可能包括定义你的目标函数（即你想最小化或最大化的量），确定你的输入和输出变量，以及设定其他限制条件。
2. 数据收集与处理
   a) 数据收集
      i. 方法：用户与现有系统的交互；人工贴标
      ii. 注意
         - 使用什么样的数据，并争论是否有足够的通用性
         - 确保正负样本平衡，以避免过度拟合到一类
         - 数据收集过程中不应该有任何偏见

   b) 预处理（过滤和去除冗余参数）
      i. 数据清洗（例如去除重复数据或缺失值）
      ii. 归一化数据（使所有变量具有相似的尺度）
      iii. 研究特征（均值、中值、直方图、矩等）
      iv. 探索特征之间的关系（协方差、相关性等）
      v. 应用降维（如 PCA）去除冗余参数（探索哪些特征是重要的，并去掉多余的特征；不必要的特征往往会在模型训练中产生问题，通常称为**维度灾难**）

   c) 将数据分成训练集、验证集和测试集
3. 模型
   a) 模型类型（取决于可用数据和性能指标）
   b) 如选择 DNN，请讨论结构、层数、层类型等
   c) 网络的内存和计算使用情况

模型的选择取决于可用数据和性能指标。确保讨论不同的 KPI 以及它们之间的比较。此类 KPI 包括但不限于

- **分类问题：**准确度、精确度、召回率、F1 分数、ROC 下面积 (AUROC)
- **回归：** MSE、MAE、R-squared/Adjusted R-squared
- **目标检测/定位：**联合交集 (IoU)、平均精度 (AP)
- **强化学习：**累积奖励、回报、Q 值、成功率。
- **系统/硬件：**延迟、能量、功率
- **业务相关 KPI：**用户留存、每日/每月活跃用户（DAU、MAU）、新用户

4. 训练
   - **损失函数选择：** CrossEntropy、MSE、MAE、Huber loss、Hinge loss

损失函数	释义	公式	其他
0-1 损失函数	模型预测的错误率	$\mathcal{L}(y, f(\mathbf{x};\theta)) = \begin{cases} 0 & \text{if } y = f(\mathbf{x};\theta) \\ 1 & \text{if } y \neq f(\mathbf{x};\theta) \end{cases} = I(y \neq f(\mathbf{x};\theta))$	虽然 0-1 损失能够客观的评价模型的好坏，但缺点是数学性质不好：不连续且导数为 0，难以优化。
平方损失函数	经常用在预测标签y为实数值的任务	$\mathcal{L}(y, f(\mathbf{x};\theta)) = \frac{1}{2}(y - f(\mathbf{x};\theta))^2$	一般不适用于分类问题
交叉熵损失函数	一般用于分类问题	$\mathcal{L}(\mathbf{y}, f(\mathbf{x};\theta)) = -\sum_{c=1}^{C} y_c \log f_c(\mathbf{x};\theta)$	正确答案概率和预测值概率之间的距离
Hinge 损失函数	主要用于 SVM 算法	$\mathcal{L}(y, f(x;\theta)) = \max(0, 1 - yf(x;\theta)) \triangleq [1 - yf(x;\theta)]_+$	最大间隔分类器，最显著的用于支持向量机。

   - **正则化：** L1、L2、熵正则化、K-fold CV、dropout
   - **反向传播：** SGD、ADAGrad、动量、RMSProp
   - **梯度消失**以及如何解决它
   - **激活函数：** Linear、ELU、RELU、Tanh、Sigmoid
   - **其他问题：**数据不平衡、过拟合、归一化等
5. 预测
   使用测试数据来评估模型的性能。这可能包括计算模型的预测准确率、精确度和召回率，或者使用其他度量标准来衡量模型的效果。
6. 模型评估

   训练模型的最终目标是在手头问题的真实场景中表现良好。它在看不见的数据上表现如何？角落案例是否被覆盖？要分析这一点，需要同时进行离线和在线评估。

   - **离线评估：**模型在数据集的保留样本上的性能。在数据集收集期间，数据被分为训练、测试和验证子集。这个想法是分析模型对看不见的数据集的概括程度。您还可以进行 K 折交叉验证，以找出不同数据子集下的性能。选择对所选 KPI 表现良好的模型进行实施和部署。
   - **在线评价：**在现实场景中部署训练好的模型（在离线评估之后）的第一步是进行 A/B 测试。训练后的模型不会很快被用于面对现实世界的大量数据。这风险太大了。相反，该模型部署在一小部分场景上。例如，假设设计的模型是为了匹配优步司机和骑手。在 A/B 测试中，该模型会说只部署在较小的地理区域而不是整个全球。然后，该模型的 beta 版本将在更长的时间内与现有模型进行比较，如果它导致业务相关 KPI 的性能提高（例如 Uber 应用程序的 DAU/MAU 更多，用户留存率，并最终提高 uber 收入），那么它将在更大范围内实施。

- Word embedding
- CBOW（根据上下文预测单词）
- Skip-gram（根据单词预测上下文）
- word2vec【无监督】（词汇聚类，词语类推）
- Language Models（考虑词汇之间的出现顺序）
  根据一串词连续在某语言中出现的概率判断
  - RNN（序列数据）【不能并行化】
    - 多对一（最后一个时间步出结果）：文本分类（整一句话看其属于哪一类）；文本生成（给前面的 t-1 个 token，预测下一个）

        - 多对多（每一个时间步的结果都用）
        - RNN encoder-decoder（序列数据→RNN→vector→RNN→序列数据）
            - encoder 或者 decoder 也可以改（比如处理图像 CNN encoder）
            - 很长序列：用 Attention
    - Transformer（seq2seq model with self-attention，也是 encoder+decoder）【可以并行化】
        - Self-attention network 可以替代任何 RNN 能处理的
        - 表示单词间顺序：Position encoding
    - Pretrained language models
        - 2 stages：Pre-training（大规模语料库上）+Fine-tuning（小规模特定数据集上）
        - BERT（基于 transformer encoder）（fine-tuning 时输入为 sequence，输出为 vector）
        - GPT（基于 transformer decoder）（句子序列生成）
        - BART（encoder-decoder）（encoder 加噪声，decoder denoising；序列到序列）
- LSTM
- MLP 与 CNN 区别
    - 局部连接：MLP 隐含层神经元连接所有输入，CNN 隐含层神经元只连接局部输入
    - 参数共享：卷积核用同样一组参数（一个卷积核负责识别一个特征）
    - …
    - MLP 等价于卷积核大小与每层输入大小相同的 CNN（如输入图片为 100*100，卷积核大小为 100*100），所以 MLP 是 CNN 的一个特例
- CNN（常用于计算机视觉）
    - LeNet-5（CONV1-POOL1-CONV2-POOL2-FC1-FC2）
    - AlexNet，VGGNet，ResNet（P111-113）
- CV
    - 图像分割（Image segmentation）
    - 目标检测（Object detection）
        - R-CNN（划分区块），Fast R-CNN（把图片先卷积成小的 feature map 再做），Faster R-CNN
    - 图像注解（Image Captioning）
        - Encoder（CNN，图像到 feature map）-Decoder（RNN，Transformer 等）
        - 对 CNN 得到的 feature map 应用 self-attention

- Vision Transformer：不要 CNN encoder，直接图片切小块并当成序列直接输入（加 position embedding），直接用 transformer encoder+decoder
- BEIT（pretrained）：基于 Vision Transformer 的预训练模型

- GAN 生成对抗网络
  - Conditional GANs（监督学习）：需要 pair 数据
  - CycleGAN：图片领域（风格）转换，不需要源和目标的 pair 数据

多分类器：MLP（fully-connected，输入数量为向量的维度，输出数量为类别数量/希望学到的参数的数量）+softmax（输出为各类别的概率）
Loss function：交叉熵（Cross Entropy）

序列建模

- 问题定义：该问题是序列建模问题，假设我们要做的是 xx 的 next-item prediction 即预测下一个可能的 xx。
- 数据处理：训练集、验证集和测试集的划分。在这个问题中，假设我们将所有 xx 序列中的最后一个 xx 作为测试集，序列中倒数第二个 xx 作为验证集，其余的作为训练集。
- 特征嵌入：不同类型的特征如何嵌入。比如 numerical feature、categorical feature、图片、文本等各类特征。
- 模型选择：考虑需要使用的预测模型，比如 RNN、Transformer、encoder-decoder、LSTM 等适合序列建模的模型（对模型的结构进行必要的说明和解释。可以包含具体的模块设计，重要参数的设置等）。把前面特征嵌入后所有的数据打包作为一个 block, 然后一个一个按照时间顺序输入，这样就得到了一个序列。我们要在 encoder 里面加 position embedding 位置变量，【简要描述 transformer 的几层，这里只要 encoder】， 再过一个 MLP 得到 vector 之后用 softmax 算概率，挑出最大概率的即可。如果是 next k, 可以再把预测出的数据作为新的 block 送给原来结构的模型再算一次
- 模型训练：选用什么样的 Loss 和优化目标，比如这里我们会选用最小化 cross entropy loss；选用什么样的优化器和优化方法；是否做模型检验等。
- 模型评估：比如在这里我们选取 recall 和 precision 作为评价指标，对 top 5 的预测输出结果在测试集上进行评估，希望取得尽可能高的 recall 和 precision。

图像分类

- 问题定义： 输入各种带有标签的图像，使得训练出来的模型可以预测没有见过的图像类别
- 数据处理： 数据样本都来自 xxx，每个都是 num×num 的**灰度（彩色）**图像，其中总共有

N 类标签，每张图像都有各自的标签。可以对图像进行处理，比如旋转，翻转，亮度调整等增加样本的数量。划分为训练集 80%，验证集 10%，测试集 10%

- 特征嵌入： 图片变成 N*N*3 的向量
- 模型选择： LeNet-5（CONV1-POOL1-CONV2-POOL2-FC1-FC2）
  - 卷积层：定义卷积核大小，padding，步长，进行卷积运算
  - 非线性激活层：使用非线性激活函数 Relu（f(x)=max(0,x)）
  - 池化层：使用 MaxPooling
  - 全连接层：输出 N 维（类别数量为 N）向量。完全连接层观察上一层的输出（其表示了更高级特征的激活映射）并确定这些特征与哪一分类最为吻合。
  - 最后接 softmax（处理后 N 维向量中的每一数字都代表某一特定类别的概率）
- 模型训练：
  - 使用反向传播来进行训练，为了防止过拟合，采用 drop out 或者 batch normalization 的策略。
  - Dropout 层将丢弃该层中一个随机的激活参数集，即在前向通过（forward pass）中将这些激活参数集设置为 0。Dropout 层只能在训练中使用，而不能用于测试过程。
  - 损失函数：使用 MSE（最小均方误差）（$E_{total}=\Sigma ½（target - output）^2$）

  前向传导、损失函数、后向传导、以及参数更新被称为一个学习周期。对每一训练图片，程序将重复固定数目的周期过程。一旦完成了最后训练样本上的参数更新，网络有望得到足够好的训练，以便层级中的权重得到正确调整。
- 模型评估： 准备不同的另一组图片与标记集（不能在训练和测试中使用相同的）并让它们通过这个 CNN。我们将输出与实际情况（ground truth）相比较，看看网络是否有效并，输出正确率，recall rate。如果使用 dropout, 在权重还要乘（1-dropout rate）,这个 dropout rate 是超参数， 要在训练的时候人为调整。

强化学习

- 问题定义：

  输入：用向量或矩阵表示的模型对于环境的观察（比如图像）
  输出：每个动作对应输出层的一个神经元
  Action= π (observation)
  寻找一个最优的策略或者值函数。要学内容给定一个 observation，输出 action，学习的内容就是 π 函数，其实就是学习这个神经网络，使得累积 reward 最高
- 数据处理：整个画面是 observation，初始画面是初始 state（或量化成坐标位置），出招

表是 action，reward 是（整个累积的，而不是每一步）

- 特征嵌入：获取当前 state 的图像。使用 CNN 神经网络得到一个 vector，再把这个 vector 丢给 MLP 后 softmax，得到出招的概率分布。之后根据这个概率分布进行一个 episode 的游戏，在此期间我们获取的是一个序列，即<s1,a1,r1><s2,a2,r2>,... <sn,an,rn>, 对这个序列我们先对每个数值进行归一化之后输入到 transformer 的 encoder 里面，输出一个 vector, 这个就是模型的参数 θ 。
- 模型选择：model-free appraoch 和 model-based approach.前者有 policy-based 和 value based，这里选用前者的 policy-based。

  States：
  Actions：
  Rewards：immediate & long-term
  Policy：Which action should be taken under each state?
  Goal：Find a policy that maximizes the expected total payoff

- 模型训练：model 观察环境 state 得到 observation, 然后根据之前的 π 函数得到 action, 这个 action 对环境 state 产生新的影响，于是又得到新的 reward，model 就根据这个 reward 调整 π 函数，生成新的 action。我们需要累积的 reward，把所有的 reward 加起来，让 total reward 最大。但是由于有随机性，用期望来表示 total reward，但是由于不可能遍历所有的可能性，我们就采样 N 个来完成。

  Q-learning（value learning）（或者 DQN）：

```
初始化 Q = {};
while Q 未收敛:
    初始化小鸟的位置S，开始新一轮游戏
    while S != 死亡状态:
        使用策略π，获得动作a=π(S)
        使用动作a进行游戏，获得小鸟的新位置S',与奖励R(S,a)
        Q[S,A] ← (1-α)*Q[S,A] + α*(R(S,a) + γ* max Q[S',a]) // 更新Q
        S ← S'
```

1. 使用策略π，获得动作a=π(S)

最直观易懂的策略π(S)是根据**Q**表格来选择效用最大的动作（若两个动作效用值一样，如初始时某位置处效用值都为0，那就选第一个动作）。

但这样的选择可能会使**Q**陷入局部最优：在位置 S_0 处，在第一次选择了动作1（飞）并获取了 $r_1 > 0$ 的奖赏后，算法将永远无法对动作2（不飞）进行更新，即使动作2最终会给出 $r_2 > r_1$ 的奖赏。

改进的策略为**ε-greedy**方法：每个状态以**ε的概率进行探索**，此时将随机选取飞或不飞，而**剩下的1-ε的概率则进行开发**，即按上述方法，选取当前状态下效用值较大的动作。

2.更新Q表格

Q表格将根据以下公式进行更新：

$$Q(S,A) \leftarrow (1-\alpha)Q(S,A) + \alpha[R(S,a) + \gamma \max_a Q(S',a)]$$

其中α为**学习速率**（learning rate），γ为**折扣因子**（discount factor）。根据公式可以看出，学习速率α越大，保留之前训练的效果就越少。折扣因子γ越大，$max_a Q(S',a)$ 所起到的作用就越大。但$max_a Q(S',a)$ 指什么呢?

考虑小鸟在对状态进行更新时，会关心到**眼前利益**（ R ），和**记忆中的利益**（$\max_a Q(S',a)$ ）。

$max_a Q(S',a)$ 是**记忆中的利益**。它是小鸟记忆里，新位置 S' 能给出的最大效用值。如果小鸟在过去的游戏中于位置 S' 的某个动作上吃过甜头（例如选择了某个动作之后获得了50的奖赏），这个公式就可以让它提早地得知这个消息，以便使下回再通过位置 S 时选择正确的动作继续进入这个吃甜头的位置 S' 。

可以看出，γ越大，小鸟就会越重视以往经验，越小，小鸟只重视眼前利益（R）。

根据上面的伪代码，便可以写出Q-learning的代码了。

- 模型评估：直接用 reward 大小

特征嵌入

要说出哪些数据是什么信息，用什么方法来做特征嵌入

- 特征需要筛选：比如需要在同一个城市等
- 数值型：归一化处理
- 类别型：onehot 编码之后 embedding：case a: 001, caseB:010,然后乘以一个 dense 的矩阵，得到一个 dense vector,需要这么做的原因是，如果类别很多 10000，那只要 10000 的 vector 变成一个 64*64 的矩阵（embedding 就是压缩 vector 把高维的数据降低纬度到低维的表示）
- 文字：简单的处理方式：word2Vec, 把所有的文字的每个单词都用 CBOW 或者 skip-gram 生成一个 64 维 vector, 然后这些 vector 做平均，就表示这一段话的 vector；或者使用预训练的模型如 BERT，gpt 得到 64 维的向量。
- 图片：输入 CNN，若干层之后有 feature 的输出层，得到向量。（可以适当描述一下 CNN）
- 最后所有特征都要嵌入到一个 item 里面，无论多少所有的特征都排好，只要过一遍 MLP，总是能够达到 64 维（这个人为规定）。（编码常见的做法）