西安交通大学
XI'AN JIAOTONG UNIVERSITY

人工智能技术导论

电气工程学院

西安交通大學
XI'AN JIAOTONG UNIVERSITY

内容概览

探索AI核心技术：从理解到试用

- 搜索与群体智能：AI的起点
- 机器学习：从数据中学习规律
- 计算机视觉：让机器“看懂”世界
- 自然语言处理：让机器“理解”语言
- 语音处理：让机器“听懂”声音

算法简单直观，适合初学者理解AI核心思想

人工智能算法的三大研究和应用领域（3~5年前）

可能的方向

- 强化学习与决策智能（Reinforcement Learning and Decision Intelligence）
- 联邦学习与隐私计算（Federated Learning）
- 多模态学习（Multimodal Learning）
- 生成模型与内容生成（Generative Models and Content Creation）
- 可解释AI与可信AI（Explainable AI and Trustworthy AI）
- 大模型与通用人工智能（Large Models and AGI）
- …… （还有很多）

西安交通大學
XI'AN JIAOTONG UNIVERSITY

从搜索开始

搜索与群体智能：AI的起点

问题1： “如果你在一个迷宫里，怎样才能最快找到出口？”

问题2： “如果你玩拼图游戏，怎样才能最快拼完？”

解空间搜索的概念：在所有可能的解中，找到最优解。

“解空间”（solution space）是一个在搜索、优化和机器学习领域中非常重要的概念，对于一个给定的问题，所有可能的解决方案的集合。是一个设计算法的重要思路提示。 *(选择题比填空题得分概率大)*

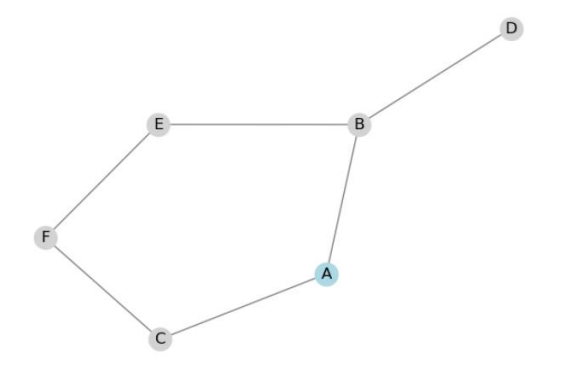
在解决具体问题时，比如路径规划、参数优化或游戏中的策略选择等，算法需要在解空间中寻找最优或满意的解决方案。解空间的大小和复杂性直接影响到解决问题的难易程度以及所采用算法的效率。例如，在遗传算法中，通过模拟自然选择过程来探索解空间；在深度学习中，训练过程可以被看作是在权重空间（一种解空间）中寻找一组最优权重值的过程。

搜索与群体智能：AI的起点

1. 理论基础

1.1 搜索算法

- 核心思想：在解空间中寻找最优解或可行解。
- 常见算法：
  - 直接搜索：深度优先搜索（DFS）、广度优先搜索（BFS）。
  - 启发式搜索：A*算法、IDA*算法。
- 应用场景：路径规划、拼图问题、机器人导航。

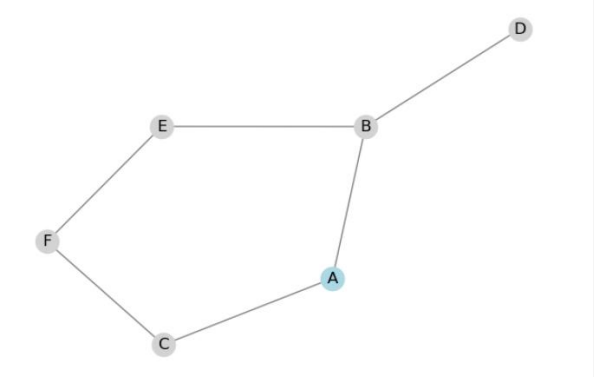



```
graph = {
    'A': ['B', 'C'],
    'B': ['D', 'E'],
    'C': ['F'],
    'D': [],
    'E': ['F'],
    'F': ['C']
}
```

图的简单定义形式：

- 定义A~F个节点；
- 定义每个节点连接的邻居；

深度优先和广度优先所搜



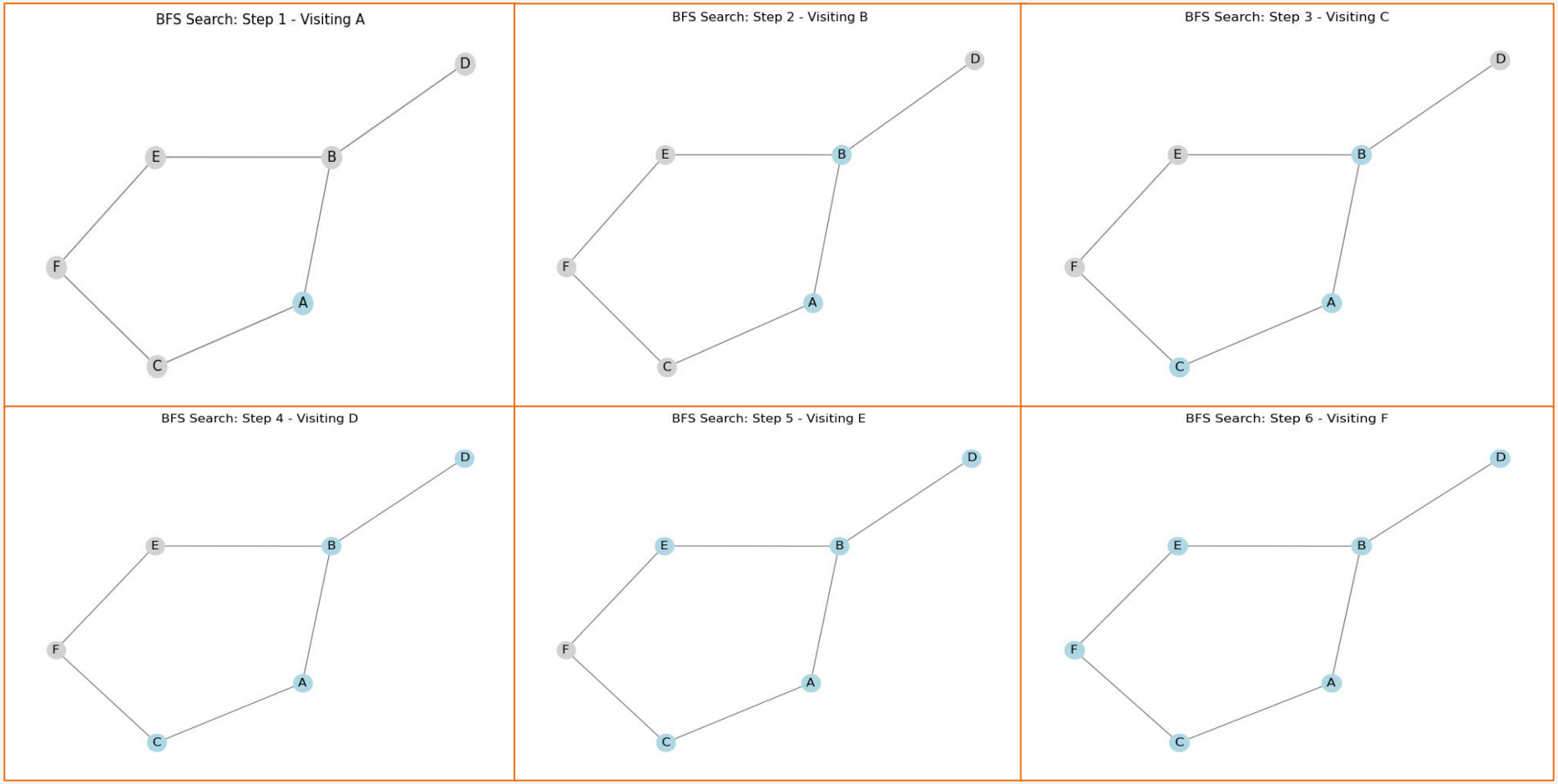
如何遍历这个图上的所有点?

- **BFS（广度优先搜索）**：先访问当前节点的所有相邻节点，再继续向外走
- 从A开始BFS访问顺序：
  1. 访问A，待访问：B，C
  2. 访问B，待访问：C、D、E
  3. 访问C，待访问：D、E、F
  4. 访问D，待访问：E、F
  5. 访问E，
  6. 访问F

- **DFS（深度优先搜索）**：深入一条路径，直到支路全完成，再回溯。
- 从A开始DFS访问顺序：
  1. 访问A，待访问：B，C
  2. 访问B，待访问：D、E、C
  3. 访问D，待访问：E、C
  4. 访问E，待访问：F，C
  5. 访问F，
  6. 访问C

不去纠结是BC还是CB，这里是示意，而且这里使用了个人的入栈习惯。

广度优先搜索示意图



- 试一下：maze.ipynb，尝试定义不同的图，看看深度和广度搜索的效果

```
[9]: # 创建一个示例图
     G = nx.Graph()
     edges = [('A', 'B'), ('A', 'C'), ('B', 'D'), ('B', 'E'), ('C', 'F'), ('E', 'F')]
     G.add_edges_from(edges)
```

有了两种基本方法之后呢?

算法	数据结构	应用场景	优点	缺点
BFS	队列/先进先出	寻找最短路径、测试连通性	理解起来简单，Dijskstra算法（加权图）	空间复杂度高
DFS	栈/先进后厨+递归	全遍历，适于树形结构，迷宫	空间复杂度低	递归回退效率低

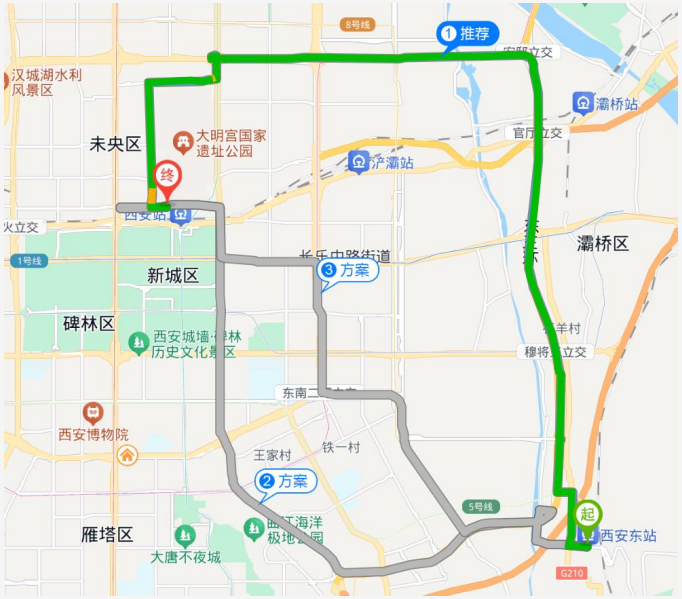
实际中，肯定不会单独使用BFS，因为实际上图的边带有权重，有权重就可以考虑代价。你的办法越来越多，你面临的问题页越来越具体。（学术无止境）

A*算法

综合考虑了**BFS**和**启发式搜索**的优点，通过公式评估每个节点的可能成本：

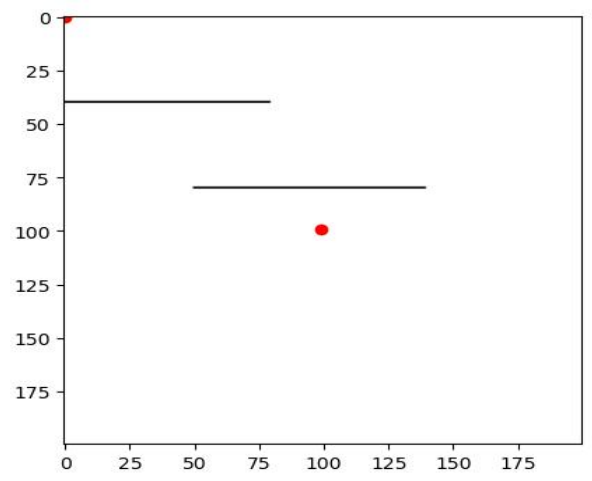
$$f(n) = g(n) + h(n)$$

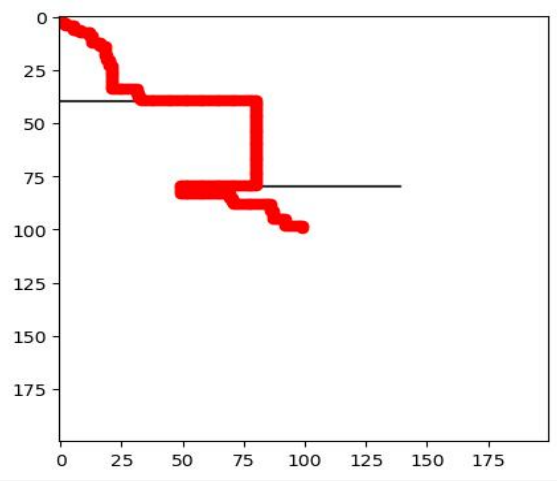
g(n)：起点到点n的成本；h(n): 点n到终点的估计成本；f(n)总估计成本。这里的h(n)就是一个启发函数。对h估计的越准，对你越有利。



A*：pros and cons，试一试

- **有效性**：如果启发式函数够用，A*算法能够找到最短路径。
- **高效性**：通过启发式函数引导搜索，能够显著减少搜索空间，提高效率。
- **灵活性**：适用于各种图结构（如网格图、加权图）。
- **广泛应用**：在路径规划、游戏AI、机器人导航等领域有广泛应用。

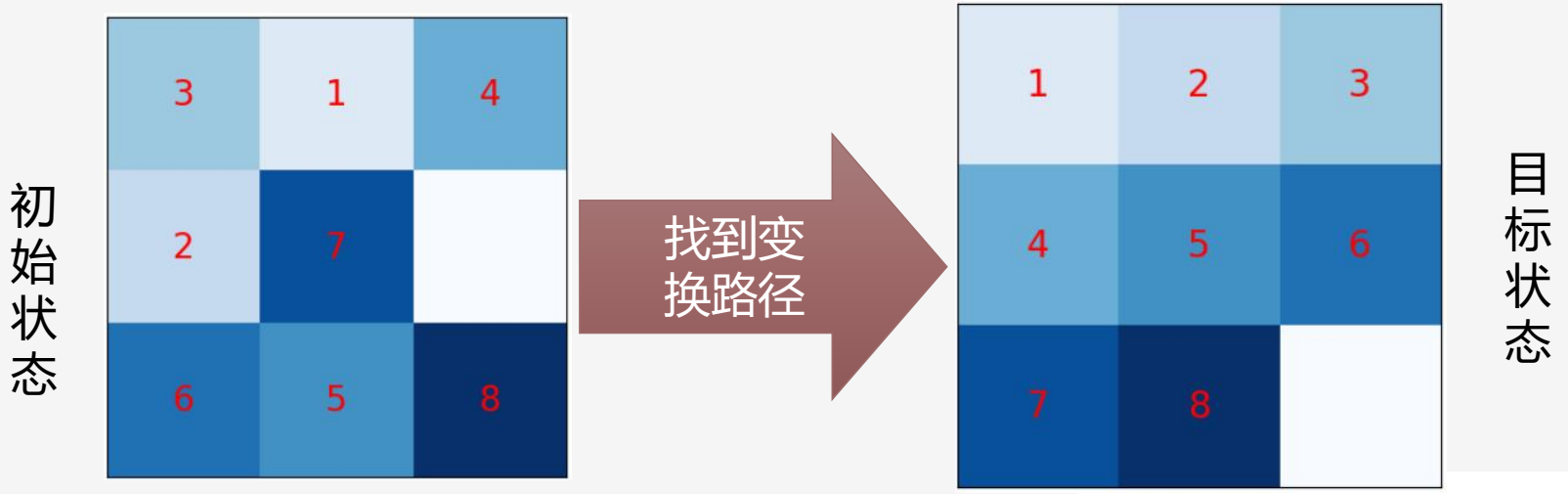




- A*算法需要存储所有已探索的节点，内存占用较高。（IDA*）
- 性能高度依赖于启发式函数的质量。如h不靠谱，效率会显著下。
- A*算法假设环境是静态的。如果环境发生变化（如新增障碍物），需要重新规划路径。

课堂试一试：定义200*200的maze,自由设置障碍物。通过A*算法寻找最优路径。

A*：八数码问题，试一试



效率比较：
BFS搜索: 117429步，0.698秒
DFS搜索: 11880步，1.880秒
A* 搜索： 1055步，0.005秒

A* 搜索快的原因： 启发式搜索定义的状态到目标的曼哈顿距离，找到看一看。

八数码的搜索

```
[8]: import matplotlib.pyplot as plt
     import numpy as np
     from collections import deque
     import time
     import random
     import heapq

     # 目标状态
     goal_state = (1, 2, 3, 4, 5, 6, 7, 8, 0)

     # 可能的移动方向
     moves = {
         'U': -3,  # 上移
         'D': 3,   # 下移
         'L': -1,  # 左移
         'R': 1    # 右移
```

```
# A* 搜索
def heuristic(state):
    return sum(abs((val-1) % 3 - i % 3) + abs((val-1) // 3 - i // 3) for i, val in enumerate(state) if val != 0)
```

```
    if move == 'D' and blank > 5:
        return False
    if move == 'L' and blank % 3 == 0:
        return False
    if move == 'R' and blank % 3 == 2:
        return False
    return True
```

提示：这个数字应该的坐标和真实坐标的差。差距越小，说明这个数字离目标位置越近。

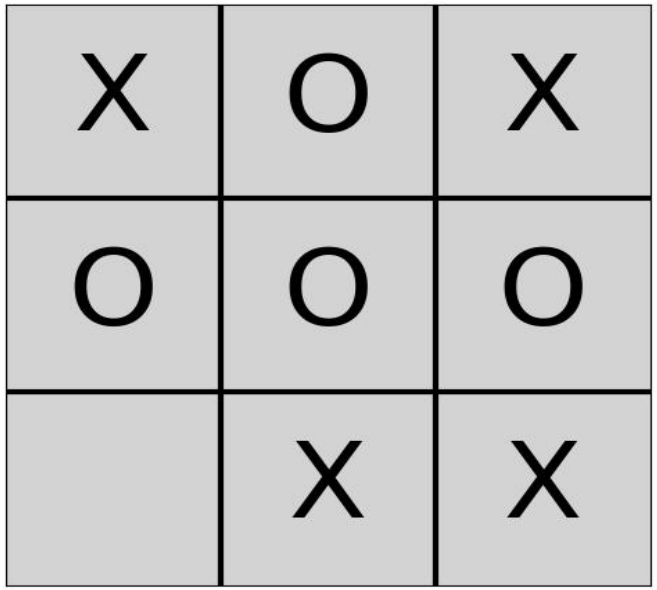
试验时，可以找到这个函数，plot_puzzle(state, step, "XXX")，可以实时输出当前9宫格状态。

井字棋（Tic-Tac-Toe），试一试

在井字棋中，我们可以把如何落子当作一个“解空间搜索问题”：

- 状态空间：所有可能的棋盘布局。
- 搜索树：每个节点代表一个棋盘状态，子节点代表可能的下一步。
- 目标：选择能最大化自己获胜机会的行动，同时最小化对手的获胜机会。
- 你们肯定会搜到minimax算法，可以看一下下发文件的代码。
- 你还会搜到MCTS算法，尝试和AI交互一下，尝试看看实现效果。

其核心是通过递归模拟双方所有可能的移动，并假设对手会采取最优策略来最小化己方收益，从而选择对己方最有利的移动。



使用**minimax**算法时，井字棋最多有 9 层递归（每次落子一层，最多 9 步），虽然不如一些其他复杂游戏，那么高，但它的时间复杂度仍然是 O(9!)，即 362880 次递归，计算量较大。他不是启发式的。详细想想，棋盘一共多少种状态？19683。

Alpha-Beta 剪枝 是 Minimax 算法的优化版，其原理是通过记录当前已知的最优解（alpha 和 beta）来避免不必要的分支计算，从而减少搜索树的大小。

- Alpha：玩家1能够获得的最大值。
- Beta：玩家2能够获得的最小值。
- 剪枝：如果某个分支的结果不可能影响最终的最优解，就可以提前剪枝，避免继续计算。

群智能：对自然的模拟

1.2 群智能（Swarm Intelligence）

- 核心思想：模拟自然界中的群体行为（如蚁群）来解决复杂问题。
- 常见算法：
  - 遗传算法（GA）：模拟生物进化，通过选择、交叉、变异优化解。
  - 粒子群优化（PSO）：模拟鸟群觅食，通过个体和群体经验更新解。
  - 蚁群算法（ACO）：模拟蚂蚁觅食，通过信息素更新路径。
- 应用场景：旅行商问题（TSP）、函数优化、调度问题。
- 这部分我没有实现的程序，有兴趣的同学可以试试。

搜索与群体智能是AI解决问题的基础，为后续复杂算法提供理论支持，应用广泛。这些算法简单直观，适合初学者理解AI核心思想，可以为深入学习奠定基础。

理解算法的思想，尝试当作自己的方法论。

机器学习

机器学习：从数据中学习规律

“**机器学习**是指计算机程序在任务T和性能度量P上表现的改善，如果**针对任务T的性能P随着经验E的增长**，则称为机器学习。” -- “机器学习之父”**汤姆•米切尔（Tom Mitchell）**，美国卡内基梅隆大学计算机学院。

“最基本的机器学习是使用算法解析数据，从中学习，然后对世界上某事做出决定或预测的做法。”

“机器学习是让计算机在没有明确编程的情况下采取行动的科学。”

机器学习的四大类：**监督学习、无监督学习、半监督学习、强化学习。**

机器学习的应用场景：图像识别、语音识别、推荐系统等。

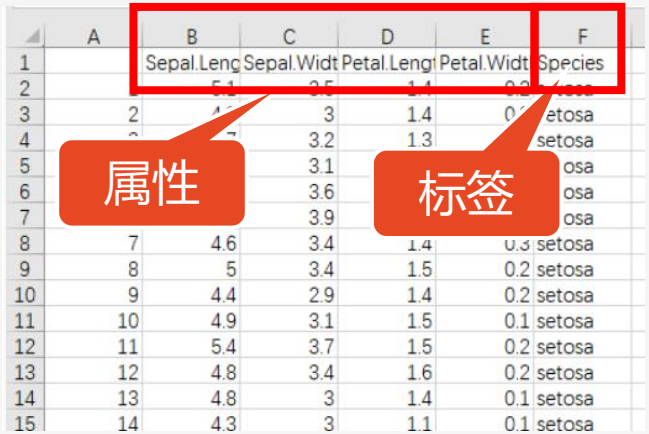
机器学习流程：数据 → 模型 → 预测。

案例：

- 监督学习：垃圾邮件分类。
- 无监督学习：客户分群。
- 半监督学习：医疗影像分类。
- 强化学习：游戏AI，AlphaGo。

监督学习：试一试

监督学习：从标注数据中学习，数据包含基本特征和目标特征（标签）。
目标是通过训练得到一个模型，使其能够对未知数据做出准确预测。常见应用包括分类和回归任务，即根据 输入特征 → 得到输出标签。
分类（Classification）：输出标签是离散值，例如男女，对错。
回归（Regression）：输出标签是连续值，例如温度。



	A	B	C	D	E	F
1		Sepal.Leng	Sepal.Widt	Petal.Leng	Petal.Widt	Species
2		5.1	3.5	1.4	0.2	setosa
3	2	4.7	3	1.4	0.2	setosa
4		4.6	3.2	1.3	0.2	setosa
5		5	3.1			osa
6		5.4	3.6			osa
7		4.6	3.9			osa
8	7	4.6	3.4	1.4	0.3	setosa
9	8	5	3.4	1.5	0.2	setosa
10	9	4.4	2.9	1.4	0.2	setosa
11	10	4.9	3.1	1.5	0.1	setosa
12	11	5.4	3.7	1.5	0.2	setosa
13	12	4.8	3.4	1.6	0.2	setosa
14	13	4.8	3	1.4	0.1	setosa
15	14	4.3	3	1.1	0.1	setosa

试一试经典算法（**问问deepseek，Scikit-learn**）：

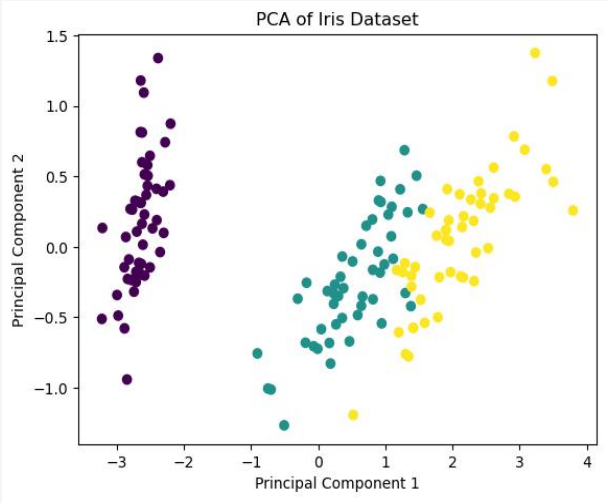
- 线性回归：连续值预测，糖尿病数据集。
- 逻辑回归：二分类问题，鸢尾花数据集。
- 决策树：树形结构的分类与回归，葡萄酒数据集。
- 支持向量机（SVM）：寻找最优分类边界，手写字识别。

无监督学习：试一试

无监督学习使用的训练数据没有标签，主要目的是从数据中发现隐藏的结构或模式，如聚类分析、关联规则学习等

常见任务：聚类（Clustering）和降维（Dimensionality Reduction）。

经典算法（继续deepseek，问问怎么用scikit-learn）：



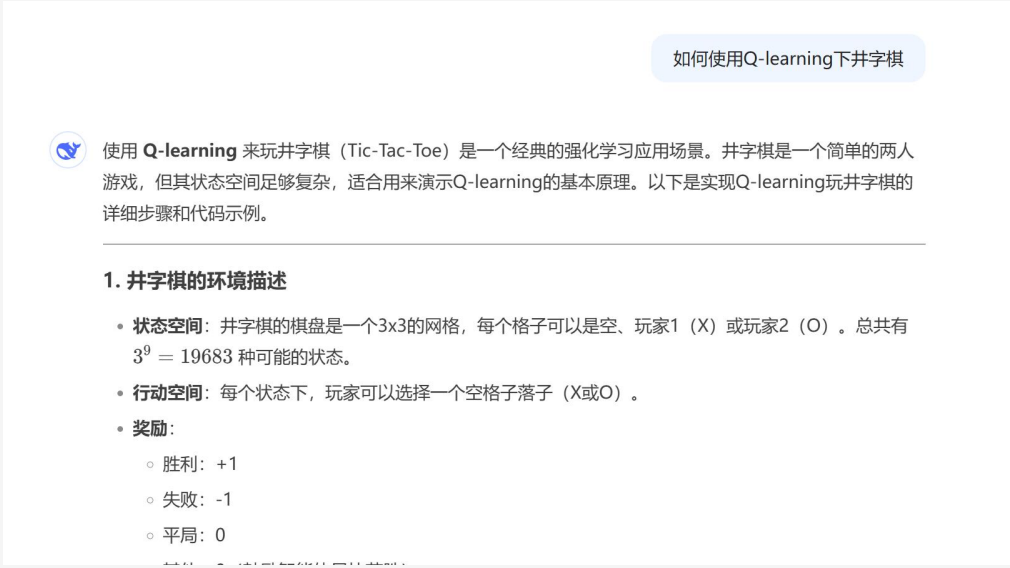
- K均值聚类（K-Means）：将数据分为K个簇。
- 层次聚类：树状结构的聚类。
- 主成分分析（PCA）：降维与特征提取。
- 高斯混合模型（Gaussian Mixture Model, GMM）聚类。
- 自组织映射（SOM）：这个要借助minisom包。

强化学习：看一看

强化学习的核心思想：智能体与环境交互，通过试错，获得奖励进行学习，找到最优策略

Q-learning：基于值函数的强化学习。能体通过与环境的交互，学习一个策略，使得在给定状态下采取的行动能够最大化累积奖励。算法中维护一个Q表，是一个二维表格，记录了在每一个状态（State）下采取每一个行动（Action）的预期累积奖励（Q值）。

核心：搜索所有的状态，确定这个状态+各个动作的Q值。



我试了，deepseek的代码有点弱。课上练习优化这个Q-learning算法。（围棋可能用得上）

世界的所有秘密都被呈现在围棋中，而只要稍微开拓一下视野，**那无穷无尽的变化就会滚滚涌现。在那如同迷宫般交错的横线和竖线上思索，然后每次解决困局找到新的棋路的欣喜，就如同发现天下至宝。**

-- 李昌镐

神经网络&生成模型

神经网络和深度学习，卷积神经网络（CNN）、循环神经网络（RNN）、长短期记忆网络（LSTM）、Transformer等，通过神经元和权重模拟复杂函数。

生成模型的核心思想：学习数据分布，生成新样本。

经典案例：

生成对抗网络（GAN）：生成器与判别器的对抗。

变分自编码器（VAE）：基于概率的生成模型。

机器学习流程：

数据收集 → 数据预处理 → 模型训练 → 模型评估 → 部署。

常用工具：Python、Scikit-learn、TensorFlow、PyTorch、mindspore

当前挑战：数据质量、模型解释性、计算资源。

未来方向：自动化机器学习（AutoML）、联邦学习、可解释AI

人工智能三大应用领域

计算机视觉：让机器“看懂”世界

计算机视觉：让机器能够“看懂”图像和视频，模拟人类的视觉系统。

主要任务：

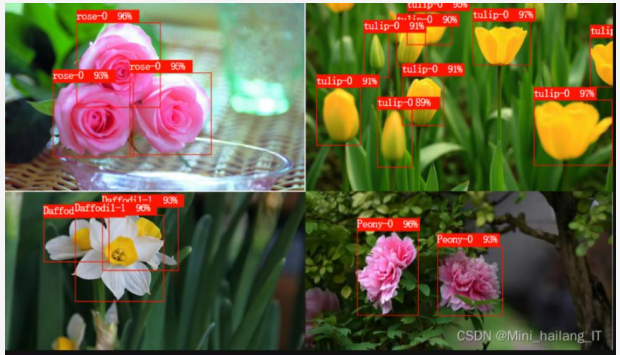
- 图像分类：识别图像中的物体或场景（如猫、狗、汽车等）。
- 目标检测：在图像中定位并识别多个物体（如人脸检测、车辆检测）。
- 图像分割：将图像划分为多个区域，每个区域对应一个物体或部分（如医学图像中的器官分割）。
- 人脸识别：识别或验证图像中的人脸。
- 视频分析：分析视频中的动作、行为或事件。

关键技术：

- 卷积神经网络（CNN）
- 目标检测算法（如YOLO、Faster R-CNN）
- 生成对抗网络（GAN）

应用场景：

- 自动驾驶：识别道路、车辆、行人等。
- 医疗影像分析：辅助医生诊断疾病。
- 安防监控：检测异常行为或入侵者。
- 增强现实（AR）：将虚拟物体叠加到现实世界中。



自然语言处理：让机器“理解”语言

自然语言处理旨在让机器能够理解、生成和处理人类文字符号体系。

核心任务：

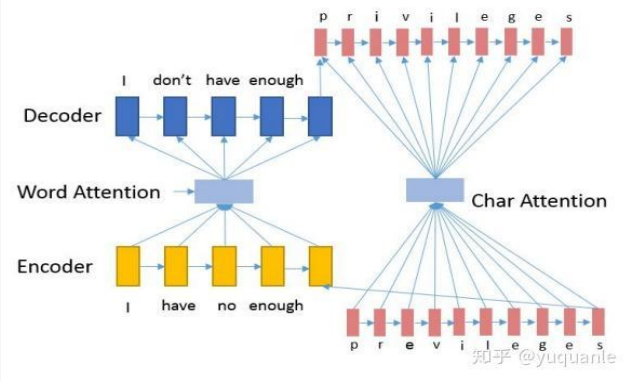
- 文本分类：将文本归类到预定义的类别（如垃圾邮件分类、情感分析）。
- 机器翻译：将一种语言翻译成另一种语言（如中英翻译）。
- 问答系统：根据问题生成答案（如智能客服、ChatGPT）。
- 文本生成：生成自然语言文本（如新闻写作、故事生成）。
- 情感分析：分析文本的情感倾向（如正面、负面、中性）。

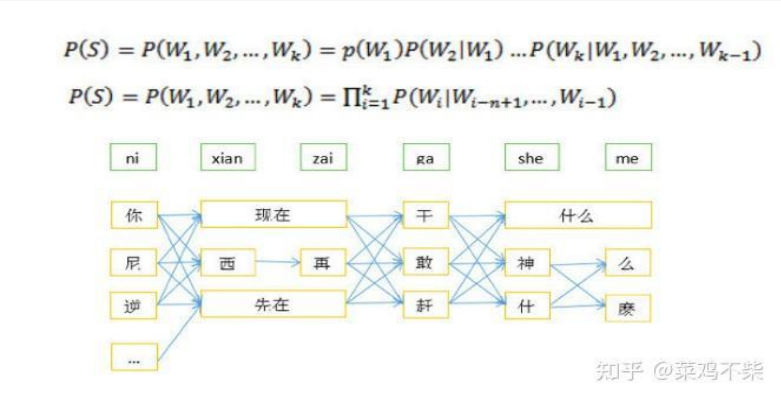
关键技术：

- 循环神经网络（RNN）
- Transformer模型（如BERT、**GPT**）
- 注意力机制（Attention Mechanism）

应用场景：

- 智能客服：自动回答用户问题。
- 搜索引擎：理解用户查询并返回相关结果。
- 文本摘要：从长文本中提取关键信息。





语音处理：让机器“听懂”声音

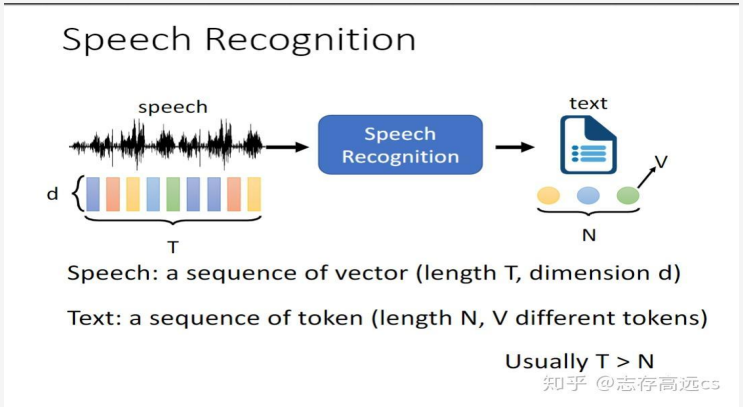
语音识别旨在让机器能够“听懂”人类语音，并将其转换为文本或命令。

核心任务：

- 语音转文本（ASR）：将语音信号转换为文本。
- 语音合成（TTS）：将文本转换为语音。
- 语音情感分析：分析语音中的情感（如愤怒、高兴、悲伤）。
- 说话人识别：识别说话人的身份。

关键技术：

- 隐马尔可夫模型（HMM）
- 深度神经网络（DNN）
- 端到端模型（如CTC、Transformer）



应用场景：

- 语音助手：如Siri、小爱同学、亚马逊Alexa、Google Assistant。
- 语音输入法：将语音转换为文字输入。
- 语音翻译：实时翻译不同语言的语音。
- 语音控制：通过语音控制智能家居设备。

三大应用场景的关系

交叉融合：

这三大应用场景并非孤立，而是常常交叉融合。例如：语音助手需要结合语音识别和自然语言处理。

多模态学习：近年来，多模态学习（如结合视觉和语言）成为研究热点，例如图像描述生成（Image Captioning）。

推荐系统：根据用户行为推荐内容（如电影、商品、新闻）。

机器人技术：让机器人能够感知环境并执行任务。

游戏AI：在游戏中模拟人类玩家的行为（如AlphaGo）。

金融科技：股票预测、风险评估、欺诈检测。

医疗AI：疾病预测、药物研发、个性化治疗。

华为昇腾: AI架构与芯片

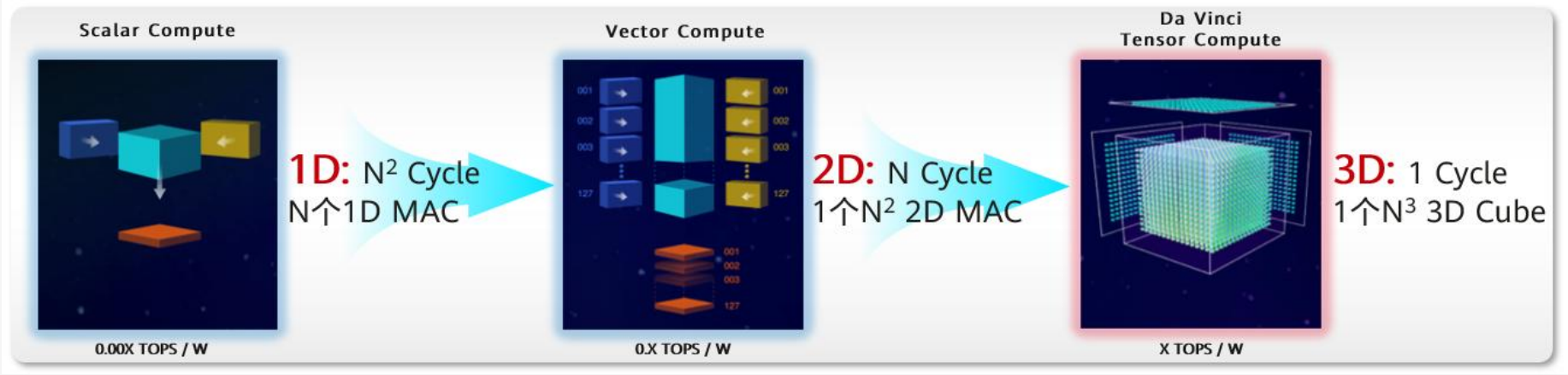
华为人工智能体系结构


昇腾AI基础软硬件平台，构筑智能世界的基石
制造
能源
交通
教育
...
深度学习框架
ModelA
HiAi Service
第三方平台&服务
MindSDK AI应用软件开发套件 >
应用使能 >
MindClust 群使能 >
MindEdge 端边使能 >
MindSpeed 加速库 >
MindIE 推理引擎 >
MindStudio
全流程工具链 >
CCAE
集群自智引擎 >
AI框架
MindSpore 昇思 >
PyTorch >
PP飞桨 >
TensorFlo
想象成cuda
硬件使能
CANN 异构计算架构 >
昇腾系列硬件
集群 >
服务器 >
标卡 >
小站 >
模组 >
开发者套件 >
伙伴硬件
...
想象成集成NVIDA显卡的硬件服务器

华为算力基础：芯片

基于Da Vinci AI技术架构

- 华为自研 AI 芯片，设计的专用神经网络计算架构。
- 昇腾芯片（HUAWEI Ascend）是华为公司发布的两款人工智能处理器，包括昇腾910和昇腾310处理器，采用自家的达芬奇架构。
- 通过专用的硬件加速单元（ NPU，神经网络处理单元）和优化的算法，核心是 3D Cube 矩阵计算引擎，专门针对深度学习中的矩阵运算进行优化。它能够高效处理卷积神经网络（CNN）等模型中的张量计算，显著提升计算效率。



华为算力基础：硬件系列

AI模块

在端侧实现目标识别、图像分类等，广泛用于智能摄像机、机器人、无人机等端侧AI场景。

Atlas 200I DK开发者套件 >

Atlas 200I A2 加速模块 >

AI加速卡

提供AI推理、视频分析等功能，支持检索聚类、OCR识别、语音分析、视频分析等场景。

Atlas 300I Pro 推理卡 >

Atlas 300I Duo 推理卡 >

Atlas 300V 视频解析卡 >

Atlas 300V Pro 视频解析卡 >

智能边缘

具有超强计算性能、体积小、环境适应性强、易于维护等特点，可以在边缘场景广泛部署。

Atlas 500 A2 智能小站 >

Atlas 500 Pro 智能边缘服务器(型号：3000) >

AI服务器

具有超强计算性能，可广泛应用于中心侧AI推理、深度学习模型开发和训练场景。

Atlas 800 推理服务器 (型号：3000) >

Atlas 800T A2训练服务器 >

Atlas 800I A2推理服务器 >

AI集群

具有超强AI算力、更优AI能效、极佳AI拓展等特点，可广泛应用于深度学习模型开发和训练。

Atlas 900 A2 PoD >

Atlas 900 AI 集群 >

Atlas 200I DK A2 开发者套件是一款高性能的AI开发者套件，可提供8TOPS INT8的计算能力，可以实现图像、视频等多种数据分析与推理计算，可广泛用于教育、机器人、无人机等场景。

Atlas 200I DK A2 开发板结构

华为算力基础：CANN

CANN（Compute Architecture for Neural Networks）

是昇腾针对AI场景推出的异构计算架构，对上支持多种AI框架，对下服务AI处理器与编程，发挥承上启下的关键作用，是提升昇腾AI处理器计算效率的关键平台。

特性	CANN（华为）	CUDA（NVIDIA）
定位	AI 计算专用架构	通用并行计算平台
硬件支持	华为 Ascend 系列 AI 处理器	NVIDIA GPU
软件生态	与 MindSpore 深度集成	支持多种框架（TensorFlow、PyTorch）
性能优化	针对 AI 计算任务优化	通用计算任务优化
应用场景	华为 AI 硬件生态	多种计算密集型任务

华为MindSpore框架

Mindspore 昇思

MindSpore 是华为推出的一款开源深度学习框架，旨在提供一站式的深度学习开发解决方案，涵盖了模型开发、训练、部署等方面。它支持在不同硬件平台（如CPU、GPU和Ascend芯片）上高效运行。

一个用来示意的例子：

[20]:
```
import mindspore
from mindspore import nn
from mindspore import context
from mindspore.dataset import GeneratorDataset
import numpy as np

# 设置设备为CPU（如果使用GPU，则为GPU）
context.set_context(mode=context.GRAPH_MODE, device_target="CPU")
```

还是python代码，引入mindspore。

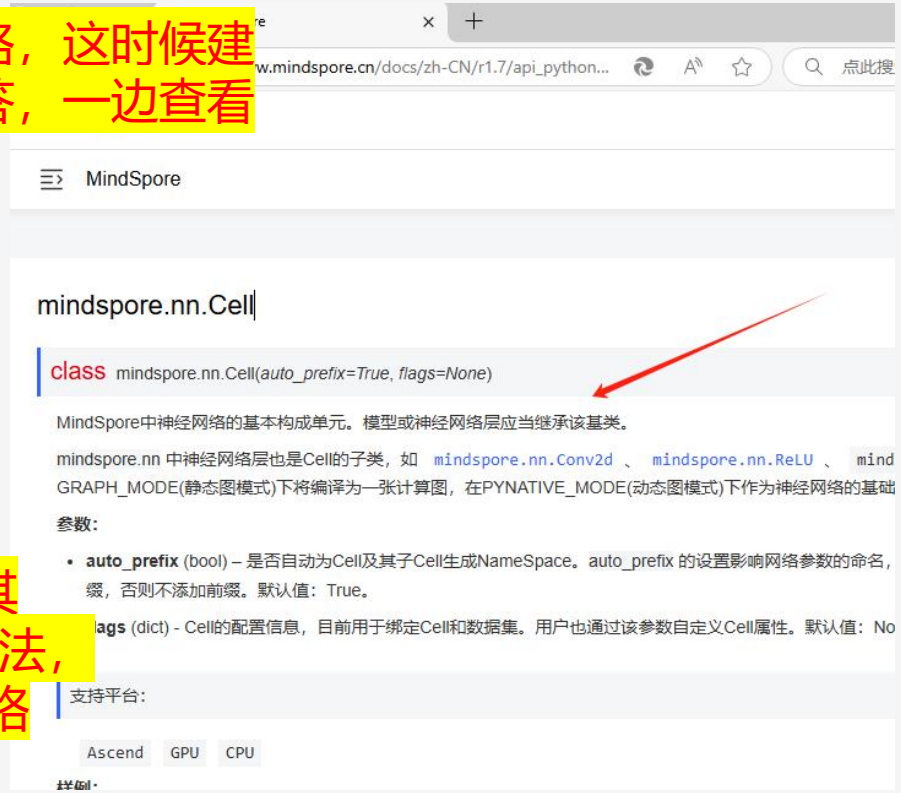
设置上下文（运行环境），可以理解为初始化

Mindspore 昇思

定义最简网络，这时候建议一边AI问答，一边查看官方手册。

```python
# 创建简单的模型
class SimpleNN(nn.Cell):
    def __init__(self):
        super(SimpleNN, self).__init__()
        self.dense1 = nn.Dense(784, 128)
        self.dense2 = nn.Dense(128, 10)

    def construct(self, x):
        x = self.dense1(x)
        x = self.dense2(x)
        return x
```



construct 方法类似于其他框架中的 forward 方法，指定了数据如何通过网络层前向传播。

self.dense1 = nn.Dense(784, 128)：第一个全连接层。此层有 784 个输入特征和 128 个输出特征。这通常用于处理扁平化后的28x28像素的手写数字图像（例如MNIST数据集）。

self.dense2 = nn.Dense(128, 10)：第二个全连接层，该层有 128 个输入特征和 10 个输出特征。这些输出对应于分类任务中的10个类别（手写数字识别的目标，即数字0到9）

Mindspore 昇思


```
# 数据集封装
dataset = GeneratorDataset(source=generate_data, column_names=["data", "label"])

# 创建网络模型
model = SimpleNN()

# 定义损失函数和优化器
loss_fn = nn.SoftmaxCrossEntropyWithLogits(sparse=True, reduction="mean")
optimizer = nn.Adam(model.trainable_params(), learning_rate=0.001)
```

导入数据。

创建神经网络。

损失函数来了

```
# 定义训练过程
train_network = nn.TrainOneStepCell(model, optimizer, loss_fn)
```

```
# 模型训练
for data, label in dataset
    data = mindspore.Tens                    .float32)
    label = mindspore.Tensor(label, mindspore.int32)
    loss = train_network(data, label)
    print(f"Loss: {loss}")
```

这里开始真正训练了，一般输出loss观察训练的效果。Loss就是训练结果和真实值的偏差。越小越好。

Mindspore 的特点

- **自动微分**：MindSpore 支持自动微分功能，可以自动计算梯度，简化了反向传播过程中的复杂操作。
- **动态图与静态图统一**：提供了 “Graph Mode”和“Pynative Mode”的模式来分别支持静态图（预编译优化）和动态图（调试友好）。
- **分布式训练**：MindSpore 内置了对大规模分布式训练的支持，允许用户配置并行训练任务，以加速模型训练过程。
- **硬件支持**：MindSpore 能够很好地适配不同的硬件平台，包括但不限于 CPU、GPU 以及华为自家的 Ascend 系列 AI 处理器。

西安交通大学
XI'AN JIAOTONG UNIVERSITY

谢谢大家

标题