# 概述

参考：<https://zhuanlan.zhihu.com/p/674095744>

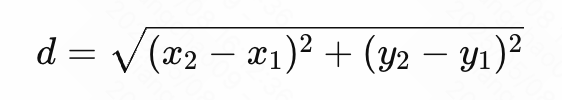
# 路径

参考：<https://www.cnblogs.com/AlvinSui/p/8931074.html>

## 欧氏距离(Euclidean Distance)

1、公式

欧几里得距离是最常见的直线距离计算方法，适用于平面上的两点之间。其计算公式为：



2、适用场景：

城市之间的平面距离估算。

无需考虑地球曲率的近似计算。

3、优缺点：

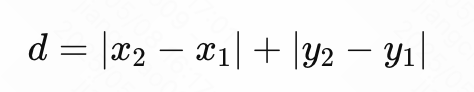
优点：计算简单，速度快。

缺点：不适用于大范围地理计算，忽略地球曲率。

## 曼哈顿距离(Manhattan Distance)

1. 公式

曼哈顿距离计算两点之间沿坐标轴的总距离，适用于网格状城市布局。其计算公式为：



2、适用场景：

城市道路呈网格状布局，如纽约等城市。

机器人路径规划中的网格环境。

3、优缺点：

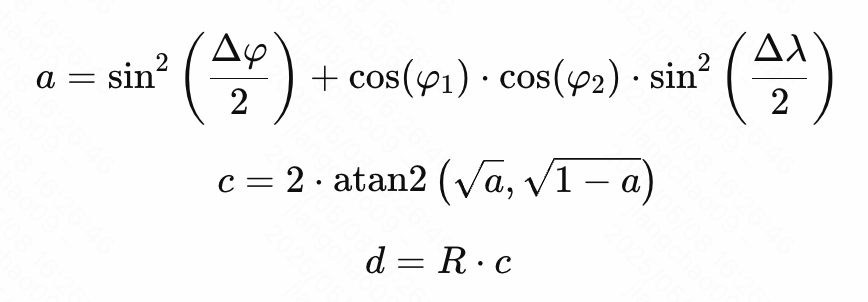
优点：适用于网格状环境，计算简单。

缺点：不适用于自然地形复杂的区域。

## 哈弗辛距离(Haversine Distance)

1、公式

哈弗辛公式用于计算球面上两点之间的最短距离，考虑地球曲率。其计算公式为：



其中，𝑅为地球半径，𝜑1,𝜑2为两点的纬度，𝜆1,𝜆2为经度。

2、适用场景：

城市间的地理距离计算。

导航系统中的路径规划。

3、优缺点：

优点：准确计算球面距离，适用于大范围地理计算。

缺点：计算复杂度较高。

## 大圆距离(Great Circle Distance)

1、原理：

大圆距离是球面上两点之间的最短路径，考虑地球曲率。其计算方法与哈弗辛距离类似，但使用不同的公式。

2、适用场景：

飞行航线规划。

航空公司调度系统。

3、优缺点：

优点：准确计算球面最短路径。

缺点：计算复杂度较高。

## 加权距离(Weighted Distance)

1、原理：

在计算距离时，考虑不同区域的权重，如交通状况、地形等因素。其计算公式为：

𝑑weighted = 𝑑 ⋅ 𝑤

其中，𝑤为权重因子。

2、适用场景：

城市交通网络中的路径规划。

考虑地形、交通等因素的路径优化。

3、优缺点：

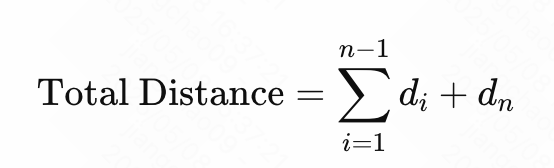
优点：更贴近实际情况，考虑更多因素。

缺点：需要额外的数据支持，计算复杂度增加。

## 旅行商问题（TSP）距离

1、原理：

旅行商问题是寻找一条最短路径，经过每个城市一次且仅一次，最终返回起点。其计算方式为：



其中，𝑑𝑖为城市间的距离，𝑛为城市数量。

2、适用场景：

快递配送路径优化。

销售路线规划。

3、优缺点：

优点：提供最优路径。

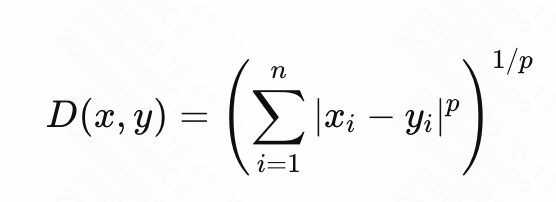
缺点：计算复杂度高，适用于城市数量较少的情况。

## 切比雪夫距离(Chebyshev Distance)

闵可夫斯基距离（Minkowski Distance）是一种度量两个点之间距离的通用方法，是欧几里得距离和曼哈顿距离的推广形式。它在机器学习、聚类、推荐系统等领域中广泛使用。

1、原理

闵可夫斯基距离的公式如下：



其中：

x = (x1, x2, ..., xn)、y = (y1, y2, ..., yn)：表示两个n维空间的点；

p：参数，决定了具体的距离类型。

常见的特例：

当 p = 1：曼哈顿距离（Manhattan Distance）

当 p = 2：欧几里得距离（Euclidean Distance）

当 p -> ∞：切比雪夫距离（Chebyshev Distance）

2、适用场景

1）聚类分析（如K-Means）：

可根据不同的 p 值选择合适的距离度量。

2）推荐系统：

用于度量用户/物品之间的相似度。

3）分类算法（如K近邻KNN）：

KNN可以灵活选择不同的 p 值来优化分类效果。

4）图像处理：

不同距离度量影响图像相似度判断。

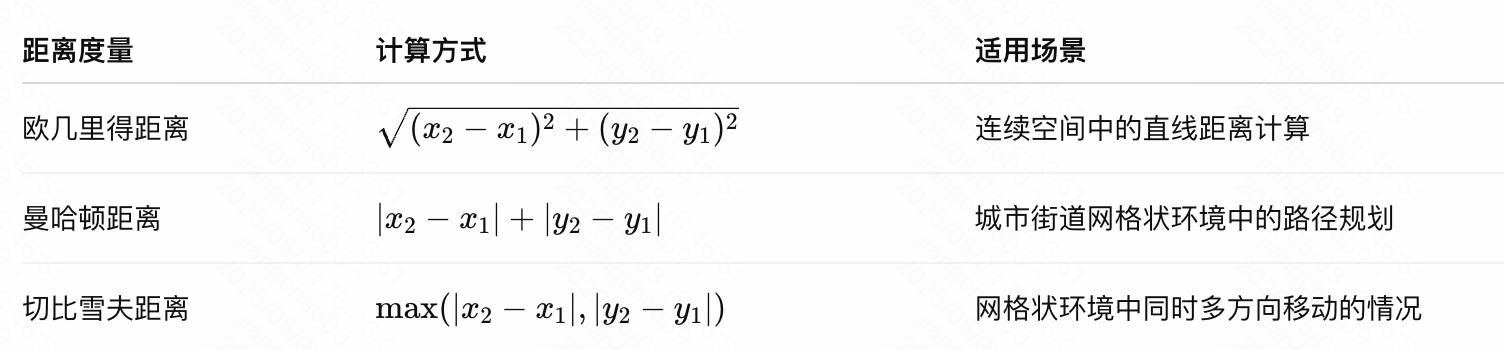
3、优点



4、缺点



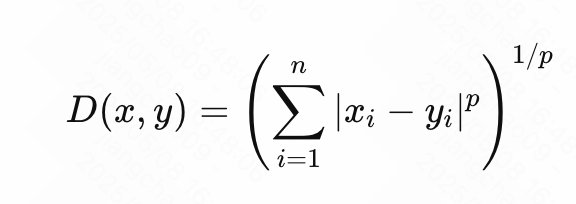
5、对比：



## 闵可夫斯基距离(Minkowski Distance)

闵可夫斯基距离是一种 度量两个向量之间差异的指标，常用于机器学习和数据分析领域，尤其适合在向量空间中衡量相似度或距离。

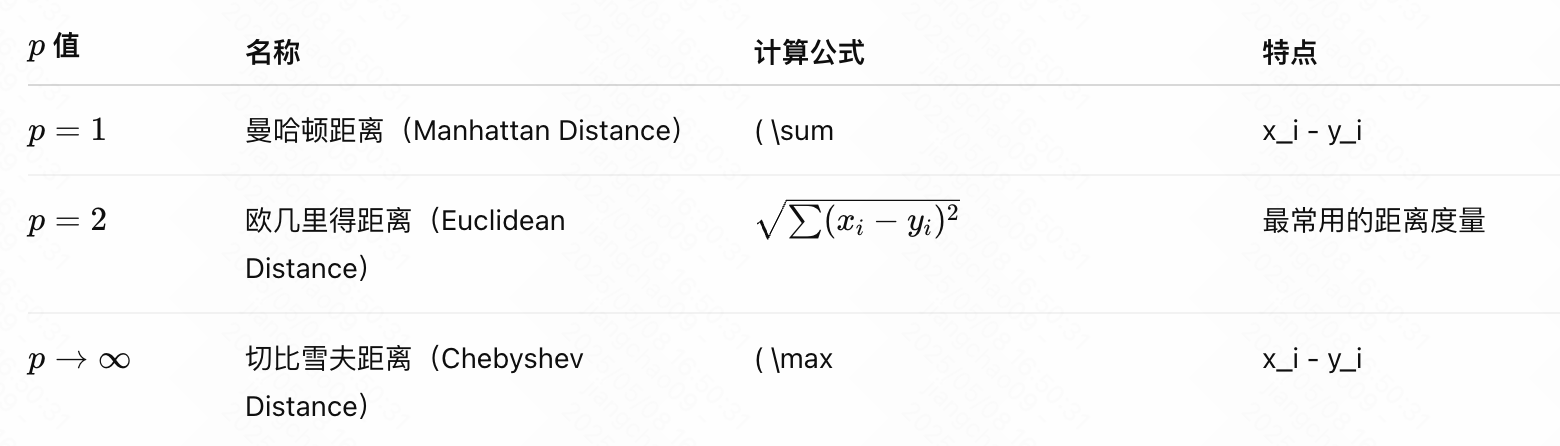
1、公式：



其中：𝑥=(𝑥1,𝑥2,...,𝑥𝑛)、𝑦=(𝑦1,𝑦2,...,𝑦𝑛)：两个点或向量；

𝑝：是一个可变的参数，用来控制距离的“形状”。

2、特殊情况（常见的 𝑝值）：



3、适用场景

K近邻算法（KNN）：自定义 𝑝值可以调节相似度判断方式。

聚类分析（如 K-Means、DBSCAN）：根据问题选取合适的距离度量。

推荐系统：衡量用户/物品的向量相似性。

异常检测：判定数据点与中心的距离。

图像识别：衡量像素或特征向量间的距离。

4、优点

通用性强：可以统一各种距离计算方式。

可调节性：通过 𝑝调整度量“灵敏度”。

适配多种算法：灵活集成于各种算法框架中。

5、缺点

对数据尺度敏感：数据未标准化时，某些特征影响过大。

维度灾难问题：在高维空间中效果可能下降。

参数调节难：选择合适的 𝑝需要经验或验证。

## 标准化欧氏距离(Standardized Euclidean Distance)

## 马氏距离(Mahalanobis Distance)

## 余弦距离(Cosine Distance)

## 汉明距离(Hamming Distance)

## 杰卡德距离(Jaccard Distance)

## 相关距离(Correlation distance)

## 信息熵(Information Entropy)

# 分类

## 经典图搜索算法

### Dijkstra算法

原理：广度优先搜索 + 贪心策略，逐步扩展最短路径树

优点：

- 保证找到全局最优解

- 适用于带权有向/无向图

缺点：

- 时间复杂度高（O(V²)）

- 无法处理负权边

适用场景：

- 城市道路导航（无负权）

- 网络路由规划

### A\*算法

原理：Dijkstra + 启发式函数（如曼哈顿距离）

优点：

- 比Dijkstra更快（利用启发式剪枝）

- 仍保证最优解（启发函数可采纳时）

缺点：

- 启发函数设计影响性能

- 高维空间可能内存爆炸

适用场景：

- 游戏AI寻路

- 机器人栅格地图导航

### D\*算法

原理：A\*的动态版本，反向搜索并重用路径信息，适用于动态环境。

优点：动态环境下效率高，路径重规划速度快。

缺点：实现复杂，极端动态场景性能可能下降。

适用场景：动态障碍物环境（如自动驾驶、无人机避障）。

## 局部优化算法

### 人工势场法

原理：模拟目标引力与障碍物斥力，通过合力控制运动方向。

优点：计算简单，实时性好，路径平滑。

缺点：易陷入局部最优，对复杂障碍物敏感。

适用场景：实时避障（如机器人局部导航）。

### 动态窗口法（DWA）

原理：在速度空间采样并模拟轨迹，通过评价函数选择最优速度。

优点：动态避障能力强，实时性高。

缺点：依赖评价函数设计，可能陷入局部最优。

适用场景：动态环境下的局部路径规划（如移动机器人避障）。

## 启发式算法

### 遗传算法（GA）

原理：模拟生物进化（选择/交叉/变异）

优点：

- 适合高维非线性问题

- 可处理多目标优化

缺点：

- 参数敏感（种群大小、变异率等）

- 早熟收敛风险

适用场景：

- 物流配送路径优化

- 超大规模TSP问题

### 蚁群算法（ACO）

原理：模拟蚂蚁信息素追踪

优点：

- 自适应动态环境

- 分布式计算友好

缺点：

- 收敛速度慢

- 需要调参（信息素挥发系数）

适用场景：

- 动态交通路径规划

- 电力网络布线

## 基于采样的算法

### RRT（快速随机探索树）

原理：随机采样构建空间搜索树

优点：

- 高维空间高效

- 避障能力强

缺点：

- 路径不一定最优

- 可能产生锯齿路径

适用场景：

- 无人机三维避障

- 机械臂运动规划

### PRM（概率路线图）

原理：先建图后搜索

优点：

- 可预处理静态环境

- 查询阶段高效

缺点：

- 建图质量影响结果

- 动态环境需重建

适用场景：

- 仓库AGV调度

- 静态环境机器人导航

## 深度学习算法

### 深度强化学习（DRL）

原理：Q-learning + 神经网络

优点：

- 可学习复杂策略

- 适应未知环境

缺点：

- 需要大量训练数据

- 黑箱决策难解释

适用场景：

- 自动驾驶决策

- 游戏NPC智能路径

### GNN路径规划

原理：图神经网络处理拓扑关系

优点：

- 直接处理图结构数据

- 端到端训练

缺点：

- 泛化能力受限

- 计算资源消耗大

适用场景：

- 社交网络信息传播优化

- 电路板布线

## 对比



# 联系

- 递进关系：Dijkstra → A\* → D\*（增量式A\*）→ 强化学习

- 互补使用：常用组合如：

- PRM + A\*（先建图后搜索）

- RRT\* + 轨迹优化（粗规划+精细调整）

- 蚁群算法 + 局部Dijkstra（全局+局部优化）

实际应用中常采用混合策略，例如京东物流使用遗传算法规划大区域配送路线，结合A\*算法处理最后一公里路径。

# 选择建议

1. 精度优先：选A\*或Dijkstra（低维静态环境）
2. 动态环境：选D或RRT变种（如RRT），需实时性时用DWA。

2、实时性要求：RRT系列（机器人避障）

3、超大规模问题：遗传/蚁群算法（物流配送）

4、未知环境：DRL（自动驾驶）