标题居中

目录

@[toc] 此位置之后 pdf 换页

KD-Tree

构建

KD-Tree 的构建算法如下：

1. 首先，计算数据集 $Data$ 各个维度的方差，选择方差最大的坐标轴作为枢轴 $pivot$
2. 然后，计算数据集在枢轴上的中位数 med，作为数据集的划分标准
3. 所有枢轴坐标不大于 med 的样本收集到子集合 L 里，所有枢轴坐标大于 med 的样本收集到子集合 R 里
4. 递归构建左右子树，直到子集合大小不超过某个阈值 T

```
\begin{algorithm}
\caption{构建 KD-Tree}
\begin{algorithmic}
\STATE \textbf{输入}: 集合 $Data = \{x_1,x_2,\cdots,x_n\}$，叶子阈值 $T$
\STATE \textbf{输出}: 树根 $root$
\PROCEDURE{KDTree}{$Data,T$}
        \IF{$n \le T$}
                \STATE $root.data := Data$
                \STATE $root.isleaf := 1$
                \RETURN $root$
        \ENDIF
        \STATE // 选择方差最大的坐标轴作为枢轴，划分数据集
        \STATE $root.pivot := \argmax_{1 \le j \le D} variance(Data,j)$
        \STATE $root.med := medain(Data,r)$
        \STATE $L,R := \empty$
        \FOR{$i:=1$ \TO $n$}
                \IF{$x_i[root.pivot] \le root.med$}
                        \STATE $L := L \cup \{x_i\}$
                \ELSE
                        \STATE $R := R \cup \{x_i\}$
                \ENDIF
        \ENDFOR
        \STATE // 递归构建左右子树
        \STATE $root.left :=$ \CALL{KDTree}{$L,T$}
        \STATE $root.right :=$ \CALL{KDTree}{$R,T$}
        \STATE $root.isleaf := 0$
        \RETURN $root$
\ENDPROCEDURE
\end{algorithmic}
\end{algorithm}
```

最近邻

在 KD-Tree 上查找给定数据的最近邻，算法如下：

1. 从根节点开始，数据与枢轴上的中值比较，进入 L, R 子集合。递归，直到进入某个叶子节点

2. 计算数据与节点上数据的最小距离点，计算距离 d_1
3. 然后回溯到父节点，计算与枢轴中值的距离 d_2
4. 如果 $d_1<d_2$，那么已经找到了最近邻；否则还要继续进入兄弟节点，以查找可能存在的更近点，然后继续回溯，直到满足 $d_1<d_2$

```
\begin{algorithm}
\caption{在 KD-Tree 上查找最近邻}
\begin{algorithmic}
\STATE \textbf{输入}: 数据 $x$，树根 $root$
\STATE \textbf{输出}: 最近邻 $y$
\PROCEDURE{FindNearest}{$x,root$}
        \IF{$root.isleaf = 1$}
                \RETURN $y := \argmin_{i \in root.data} dist(x,i)$
        \ENDIF
        \STATE // 递归查找最近邻，找到可能值之后回溯
        \IF{$x[root.pivot] \le root.med$}
                \STATE $tag := 0$
                \STATE $y :=$ \CALL{FindNearest}{$x,root.left$}
        \ELSE
                \STATE $tag := 1$
                \STATE $y :=$ \CALL{FindNearest}{$x,root.right$}
        \ENDIF
        \STATE $d_1 := dist(x,y)$
        \STATE $d_2 := |x[root.pivot]-root.med|$
        \STATE // 判断是否已经获得最近邻
        \IF{$d_1 > d_2$}
                \IF{$tag = 0$}
                        \STATE $z :=$  \CALL{FindNearest}{$x,root.right$}
                \ELSE
                        \STATE $z :=$  \CALL{FindNearest}{$x,root.left$}
                \ENDIF
                \STATE $y := \argmin_{i=y,z} dist(x,i)$
        \ENDIF
        \RETURN $y$
\ENDPROCEDURE
\end{algorithmic}
\end{algorithm}
```

添加数据

在已有的数据集上构建好 KD-Tree 之后，我们可能还有加入新样本的需求。新样本的加入规则很简单，只需找出这个样本所属于的区域（某个叶子节点），然后把新样本添加到这个区域内即可。

KD-Tree 的数据添加算法如下：

1. 从根节点开始，数据与枢轴上的中值比较，进入 L, R 子集合。递归，直到进入某个叶子节点
2. 如果添加新数据后，叶子节点中包含的集合大小超过阈值 T，那么就把叶子集合按照 KD-Tree 构建算法，分割为多个节点

```
\begin{algorithm}
\caption{在 KD-Tree 上添加新数据}
\begin{algorithmic}
\STATE \textbf{输入}: 新数据 $x$, 树根 $root$, 叶子阈值 $T$
\STATE \textbf{输出}: 树根 $root$
\PROCEDURE{AddData}{$x,root,T$}
        \IF{$root.isleaf = 1$}
                \STATE // 判断是否需要分裂
                \IF{$|root.data| \ge T$}
                        \STATE $root :=$ \CALL{KDTree}{$root.data \cup \
{x\}$}
                \ELSE
                        \STATE $root.data := root.data \cup \{x\}$
                \ENDIF
                \RETURN $root$
        \ENDIF
        \STATE // 递归进入左右子树
        \IF{$x[root.pivot] \le root.med$}
                \STATE \CALL{AddData}{$x,root.left,T$}
        \ELSE
                \STATE \CALL{AddData}{$x,root.right,T$}
        \ENDIF
        \RETURN $root$
\ENDPROCEDURE
\end{algorithmic}
\end{algorithm}
```