k 近邻算法(knn, k nearest neighbor)代码

前两天受朋友之托，帮忙与两个 k 近邻算法，k 近邻的非正式描述，就是给定一个样本集 exset，样本数为 M，每个样本点是 N 维向量，对于给定目标点 d，d 也为 N 维向量，要从 exset 中找出与 d 距离最近的 k 个点(k<=N)，当 k=1 时，knn 问题就变成了最近邻问题。最 naive 的方法就是求出 exset 中所有样本与 d 的距离，进行按出小到大排序，取前 k 个即为所求，但这样的复杂度为 O(N)，当样本数大时，效率非常低下. 我实现了层次 knn(HKNN)和 kdtree knn，它们都是通过对树进行剪枝达到提高搜索效率的目的，hknn 的剪枝原理是(以最近邻问题为例)，如果目标点 d 与当前最近邻点 x 的距离，小于 d 与某结点 Kp 中心的距离加上 Kp 的半径，那么结点 Kp 中的任何一点到目标点的距离都会大于 d 与当前最近邻点的距离，从而它们不可能是最近邻点(K 近邻问题类似于它)，这个结点可以被排除掉。 kdtree 对样本集所在超平面进行划分成子超平面，剪枝原理是， 如果某个子超平面与目标点的最近距离大于 d 与当前最近点 x 的距离，则该超平面上的点到 d 的距离都大于当前最近邻点，从而被剪掉。 两个算法均用 matlab 实现(应要求)，把代码帖在下面，以备将来查用或者需要的朋友可以参考.

VecDist.m

```
function y = VecDist(a, b)
%%返回两向量距离的平方
assert(length(a) == length(b));
y = sum((a-b).^2);
end
```

下面是 HKNN 的代码

Node.m

```
classdef Node < handle
    %UNTITLED2 Summary of this class goes here
    %     Detailed explanation goes here
    % Node  层次树中的一个结点，对应一个样本子集 Kp
    properties
        Np; %Kp 的样本数
        Mp; %Kp 的样本均值，即中心
        Rp; %Kp 中样本到 Mp 的最大距离
        Leafs; %生成的子节点的叶子,C * k 矩阵,C 为中心数量，k 是样本维数。如果不是叶结点，则为空
        SubNode; %子节点, 行向量
    end
    methods
        function obj = Node(samples, maxLeaf)
            global SAMPLES
            %samples 是个列向量，它里面的元素是 SAMPLES 的行的下标,而不是 SAMPLES
```

行向量，使用全局变量是出于效率上的考虑

```
                obj.Np = length(samples);
                if (obj.Np <= maxLeaf)
                    obj.Leafs = samples;
                else
%                       opts = statset('MaxIter',100);
%                                           [IDX] = kmeans(SAMPLES(samples, :), maxLeaf,
'EmptyAction','singleton','Options',opts);
                    [IDX] = kmeans(SAMPLES(samples, :), maxLeaf, 'EmptyAction','singleton');
                    for k = 1:maxLeaf
                        idxs = (IDX == k);
                        samp = samples(idxs);
                        newObj = Node(samp, maxLeaf);
                        obj.SubNode = [obj.SubNode newObj];%SubNode 为空说明当层的
Centers 是叶结点
                    end
                end
                obj.Mp = mean(SAMPLES(samples, :), 1);
                dist = zeros(1, obj.Np);
                for t = 1:obj.Np
                    dist(t) = VecDist(SAMPLES(samples(t), :), obj.Mp);
                end
                obj.Rp = max(dist);
            end
        end
end
```

SearchKNN.m

```
function SearchKnn(Node)
global KNNVec KNNDist B DEST SAMPLES
m = length(Node.Leafs);
if m ~= 0
    %叶结点
    %是叶结点
    for k = 1:m
        D_X_Xi = VecDist(DEST, SAMPLES(Node.Leafs(k), :));
        if (D_X_Xi < B)
            [Dmax, I] = max(KNNDist);
            KNNDist(I) = D_X_Xi;
            KNNVec(I) = Node.Leafs(k);
            B = max(KNNDist);
        end
    end
```

```
else
    %非叶结点
    tab = Node.SubNode;
    D = zeros(size(tab));
    delMark = zeros(size(tab));
    for k = 1:length(tab)
        D(k) = VecDist(DEST, tab(k).Mp);
        if (D(k) > B + tab(k).Rp)
            delMark(k) = 1;
        end
    end
    tab(delMark == 1) = [];
    for k = 1:length(tab)
        SearchKnn(tab(k));
    end
end
```

下面是 kdtree 的代码

KDTree.m

```
classdef KDTree < handle
    %UNTITLED2 Summary of this class goes here
    %     Detailed explanation goes here

    properties
        dom_elt; %A point from Kd_d space, point associated with the current node
        split_pos;%分割位置，比如对于 K 维向量，这个位置可以是从 1 到 k
        left;%左子树
        right;%右子树
        bNULL;%标识这个结点是否是 NULL
    end

    methods (Static)
        function [sample, index, split] = ChoosePivot1(samples)
            global SAMPLES
            dimVar = var(SAMPLES(samples, :));
            [maxVar, split] = max(dimVar);%分界点的维，即从第多少维处分
            [sorted, IDX] = sort(SAMPLES(samples, split));
            n = length(IDX);
            index = IDX(round(n/2));
            sample = samples(index);
        end
        function [sample, index, split] = ChoosePivot2(samples)
```

```
            %第二种 pivot 选择策略,选择范围最长的那维作为 pivot
            %注意：这个选择策略是以树的不平衡性换取剪枝时的效果，对于有些数据分布，性能可能反而下降
            global SAMPLES
            [upper, I] = max(SAMPLES(samples, :), [], 1);%按列取最大值
            [bottom, I] = min(SAMPLES(samples, :), [], 1);%
            range = upper-bottom;%行向量
            [maxRange, split] = max(range);%分界点的维，即从第多少维处分
            [sorted, IDX] = sort(SAMPLES(samples, split));
            n = length(IDX);
            index = IDX(round(n/2));
            sample = samples(index);
        end
        function [exleft, exright] = SplitExset(exset, ex, pivot)
            global SAMPLES
            vec = SAMPLES(exset, pivot);%列向量
            flag = (vec <= SAMPLES(ex, pivot));
            exleft = exset(flag);
            flag = ~flag;
            exright = exset(flag);
        end
    end

    methods
        function obj = KDTree(exset)
            %输入向量集,SAMPLES 的下标
            if isempty(exset)
                obj.bNULL = true;
            else
                obj.bNULL = false;
                [ex, index, split] = KDTree.ChoosePivot1(exset);
                %[ex, index, split] = KDTree.ChoosePivot2(exset);
                obj.dom_elt = ex;
                obj.split_pos = split;

                exset_ = exset;%exset 除去先作分割点的那个点
                exset_(exset == ex) = [];

                %将 exset_分成左右两个样本集
                [exsetLeft, exsetRight] = KDTree.SplitExset(exset_, ex, split);
                %递归构造左右子树
                obj.left = KDTree(exsetLeft);
                obj.right = KDTree(exsetRight);
            end
```

```
            end
        end
end
```

SearchKnn.m

```
function SearchKNN(kd, hr)

%SearchKNN Summary of this function goes here
%     Detailed explanation goes here
% kd 是 kdtree
% hr 是输入超平面图,它是由两个点决定，类比平面和二维点，所有二维点都在平面上，
% 而平面上的一个矩形区域，可以由平面上的两个点决定
% 首次迭代，输入超平面为一个能覆盖所有点的超平面。对于二维，可以想像 p1 = (-infinite, -infinite)
% 到 p2 = （infinite, infinite)的平面可以覆盖二维平面所有点。可以推测一个可以覆盖 K 维空间所有点的的超平面图
% 应该是(-inf, -inf....-inf),k 维,到正的相应无穷点
    global SAMPLES DEST MAX_DIST_SQD %global in
    %DIST_SQD, SQD 是指距离的平方
    global KNNVec KNNDist %global out
    if kd.bNULL
        %kd 是空的
        return;
    end
    %kd 不为空
    pivot = kd.dom_elt;%下标
    s = kd.split_pos;
    %分割输入超平面
    %分割面是经过 pivot 并且 cui 直于第 s 维
    %还原是以二维情况联想，可以得到分割后的两个超平面图
    left_hr_right_point = hr(2,:);
    left_hr_right_point(s) = SAMPLES(pivot,s);
    left_hr = [hr(1,:);left_hr_right_point];%得到分割后的 left 超平面
    right_hr_left_point = hr(1,:);
    right_hr_left_point(s) = SAMPLES(pivot, s);
    right_hr = [right_hr_left_point;hr(2,:)];%得到 right 超平面

    % 判断目标点在哪个超平面上
    % 始终以二维情况来理解，不然比较抽象
    bTarget_in_left = (DEST(s) <= SAMPLES(pivot, s));
    nearer_kd = [];
    nearer_hr = [];
    further_kd = [];
```

```
    further_hr = [];
    if bTarget_in_left
        %如果在左边超平面上
        %那么最近点在 kd 的左孩子上
        nearer_kd = kd.left;
        nearer_hr = left_hr;
        further_kd = kd.right;
        further_hr = right_hr;
    else
        %在右孩子上
        nearer_kd = kd.right;
        nearer_hr = right_hr;
        further_kd = kd.left;
        further_hr = left_hr;
    end
    SearchKNN(nearer_kd, nearer_hr);
    % A nearer point could only lie in further_kd if there were some
    % part of further_hr within distance sqrt(MAX_DIST_SQD) of target
    sqrt_Maxdist = sqrt(MAX_DIST_SQD);
%       剪枝就在这里
    bIntersect = CheckInterSect(further_hr, sqrt_Maxdist, DEST);
    if ~bIntersect
        %如果不相交,没有必要继续搜索了
        return;
    end
    %如果超平面与超球有相交部分
    d = VecDist(SAMPLES(pivot, :), DEST);
    if d < MAX_DIST_SQD
        [Dmax, I] = max(KNNDist);
        KNNVec(I) = pivot;
        KNNDist(I) = d;
        MAX_DIST_SQD = max(KNNDist);
    end
    SearchKNN(further_kd, further_hr);
end

function bIntersect = CheckInterSect(hr, radius, t)
%检查以点 t 为中心，radius 为半径的圆，与超平面 hr 是否相交,为方便
%在超平面上找到一个距 t 最近的点，如果这个距离小于等于 radius，则相交
%如何确定超平面上到 t 最近的点 p：
%假设超平面 hr 在第 i 维的上限和下限分别是 hri_max, hri_min,则有
%           hri_min, if ti <= hri_min
% pi = ti, if hri_min < ti < hri_max
%           hri_max, if ti >= hri_max
```

```
    p = zeros(size(t));%超平面上与 t 最近的点,待求
    minHr = hr(1,:);maxHr = hr(2,:);
%       for k = 1:length(t)
%           if (t(k) <= minHr(k))
%               p(k) = minHr(k);
%           elseif (t(k) >= maxHr(k))
%               p(k) = maxHr(k);
%           else
%               p(k) = t(k);
%           end
%       end
    flag1 = (t <= minHr);p(flag1) = minHr(flag1);
    flag2 = (t >= maxHr);p(flag2) = maxHr(flag2);
    flag3 = ~(flag1 | flag2);p(flag3) = t(flag3);

    if (VecDist(p, t) >radius^2)
        bIntersect = false;
    else
        bIntersect = true;
    end
end
```

OK,就这么多了，需要的朋友可以随便下载使用~