

Pattern Recognition and Similarity Search

1. For this question, (a) and (b) worth one point, (c) and (d) worth two points and (e) gives you two bonus points. I am going to show the general case that the VC dimension of linear half spaces in n -dimensional space is $n + 1$.

There are two parts in the proof: first, you need to show that arbitrary configuration of $n + 1$ points can be shattered by a half plane. Second, you need to provide a counter-example of $n + 2$ points that cannot be shattered by a half plane.

For the first part, I am following Wei Zhang's idea: the idea is to first show that one can always shatter a particular $n + 1$ point configuration, namely the origin plus the vertices of a simplex. Then, for arbitrary general configuration, one can always map them to the simple case while preserving which side of the half plane they are on. First, consider this particular configuration $e_0 = (0, 0, \dots, 0)^T$, $e_1 = (1, 0, \dots, 0)^T$, $e_2 = (0, 1, 0, \dots, 0)^T$, \dots , $e_n = (0, \dots, 0, 1)^T$. For arbitrary labeling $y_i \in \{1, -1\}$ for $i = 0, \dots, n$, it is easy to check that the plane with normal $w = (y_1 - y_0, y_2 - y_0, \dots, y_n - y_0)$ and intercept $b = y_0$ will do the job, i.e. $\text{sgn}(w^T e_i + b) = y_i$ for all i .

Now, for general s_0, \dots, s_n , if they are not coplanar, then the matrix $S = (s_1 - s_0, s_2 - s_0, \dots, s_n - s_0)$ is full-rank and its inverse exists. For arbitrary labeling $y_i \in \{1, -1\}$, consider the normal $\hat{w} = w^T S^{-1}$ and intercept $\hat{b} = b - w^T S^{-1} s_0$, with w and b as defined before. For all i , we have $\text{sgn}(\hat{w}^T s_i + \hat{b}) = \text{sgn}(w^T S^{-1} s_i - w^T S^{-1} s_0 + b) = \text{sgn}(w^T e_i + b) = y_i$.

For the second part, it is intuitive to see that in general, the convex hull of $n + 1$ points s_0, s_1, \dots, s_n in the n -dimensional has non-zero "volume" in the n -dimensional space – two points form a line segment with non-zero length in 1-D, three points form a triangle in 2-D, and four points form a tetrahedron in 3-D, etc. Label all these $n + 1$ points as class 1. Since the convex hull has non-zero volume, it is possible to find one additional point s_{n+1} inside the convex hull such that there exists an ϵ -ball $B_{s_{n+1}}(\epsilon)$ centered at s_{n+1} with radius ϵ entirely contained inside the convex hull. Label s_{n+1} as class -1.

If a half plane can shatter this configuration, the entire convex hull must be on one side of the half plane. This is easy to see because an arbitrary point p inside the convex hull can be written as $p = \sum_{i=0}^n \lambda_i s_i$, with all $\lambda_i \geq 0$ and $\sum_{i=0}^n \lambda_i = 1$. Let w be the normal of the half plane. Since s_0, \dots, s_n are all on the same side of the plane, we have, for all i , $s_i^T w \geq b$ for some positive b . As a result, $p^T w = \sum_{i=0}^n \lambda_i w^T s_i \geq b \sum_{i=0}^n \lambda_i = b$. So p must also be on the same side as s_0, \dots, s_n . Since $B_{s_{n+1}}(\epsilon)$ is inside the convex hull, p must be entirely on the same side with all the other points, not touching any part of the half plane.

2. For the second problem, 3 points for (a) and 1 point for each data set in (b). Most of you did very well on this question.
3. For the third problem, 1 point for the fast search and 1 point for sequential search. Everyone uses KD-tree for the fast search and the timing performance for 80 queries ranges from 1 second to 300 seconds, with the mean search time around 20 seconds.

For sequential search, you will get zero point if your search method spends more than 5 minutes on either dataset. Even if your CPU is slow and the memory is not enough, you can do the computation

while reading from the database. This is the fundamental idea of pipelining and all of you should know this. Besides, it is an extremely simple task and you should try to write a C program instead of using matlab.

The speed performance for highD1 should be better than highD2. The first dataset comes from real data while the second one is randomly generated. Real dataset typically exhibits a clustered structure which means that the index structure most likely does not have to look far to search for the closest neighbor.