K-Local Hyperplane and Convex Distance Nearest Neighbor Algorithms

Pascal vincent and Yeshua Bengio

January 2021

1 Analysis

The main aim of the authors in writing this paper is to improve the existing KNN(K-nearest neighbor) algorithm by making some changes, which lead to the ideation of the new algorithms HKNN(K-local hyperplane distance nearest neighbor) and CKNN(K-local Convex Distance Nearest Neighbor Algorithm), both looks very promising and can produce better results compared to the existing KNN. They explained why we are comparing KNN with SVM, in SVM we are considering a hyperplane which exactly at equal distance from all the class points, for non-linear kernel trick is used to get the solution, where 'local margin' is defined and explained as the euclidean distance from a point on the decision surface to the test point in the input space, so we are making the 'local linear decision surfaces'. On the other hand, the KNN algorithm works with the principle of finding the distance of the data points to the test point, and assigning the smallest distance value. So both work the same with local linearly classified data. In this paper, they also tried to explain why KNN performance is poor when compared to SVM, When there are only finite samples in input data and some of the samples are missing. Then in order to compensate for that, they said that we have to fantasize(fill or imagine) the missing data and approximate each manifold locally,(we can think that each class is a low dimensional space in high dimensional input space then we can consider this data as locally linear) introduced some new terms like 'local hyperplane', the test point is associated to a class based on the local linear manifold (as making these will reduce the effects of the missing data and improves the result accuracies) and this algorithm as HKNN, this is same as SVM which makes decision surface in high dimension, but in HKNN hyperplane is made in input space itself. but this algorithm also has a constraint when the value of K (the number of nearest neighbors) increases, it is difficult to form the hyperplanes, as looking for collinearities is difficult. so defined some solutions in which one of them takes a convex hull around the points rather than taking local hyperplane, called a convex-hull solution and another one adds-up a weight(penalizing) term for the existing HKNN algorithm which makes it to penalize for high values of α restricting it from going far from the centroid, called as a generalization of the HKNN.

2 Advantages and Disadvantages of the paper

The authors have given enough theory part to understand the algorithm, their ideations and intuitions had been clearly put on the paper and expressed neatly. The literature used is simple and understandable. Explained some concepts from the roots and provided the basic principle operations of the algorithms SVM and KNN. provided solutions to many questions like why SVM outperforms KNN, what are the drawbacks of the KNN. at the same time, some parts are questionable as not enough explanation is given. One disadvantage of the paper is that some parts of the equations are not understandable and only through theory we can't understand the whole algorithm. Some doubts are unsolved, like how we are getting the constraint on the value alpha, how to fantasize(fill or imagine) missing points in a dataset. These can be understood if they gave any examples or references. some parts of the equation are to be modified and better explained. Just given only the experimental results for datasets like MNIST, but didn't give information like how we will get the value of alpha and how to tune between two hyperparameters K and lambda. Not given any ideas on how to implement it in basic programming, as implementing a new algorithm is not easy, so we can take this as a future work(research) and can implement the algorithm with python scratch, as python libraries also don't contain this algorithm HKNN. We can also work on creating local hyperplanes for different types of Data and have to implement them with different hyperparameters.

3 Pros and Cons of proposed algorithms

The proposed algorithms are giving better results than KNN. The HKNN algorithm is simple to implement from the top of the KNN algorithm as there are only a few changes or few steps to follow. Just we have to solve a linear plane equation to get the algorithm, solve for the alpha, and have to tune the parameter K. the results of this algorithm have reached the level of SVM and even better!. one of the major problem for HKNN is it gives best results only with euclidean distance and not with others, other problem occurs with large values of K, for which the data may not be forming collinear surfaces so the decision surface will take the wrong paths. This is a disadvantage of this algorithm. Two solutions are proposed to compensate for this drawback. The results of the CKNN algorithm may look satisfying, but at the same time it's not easy to calculate the distance of the test point from the convex-hull surface by just using a linear system, instead, we have to use some quadratic calculations as the plane produced is not linear. So it is complex to implement in terms of the calculations.

The second solution adding a weight decay term is the same as HKNN but a term is added to compensate/penalize for large values of alpha, but here we have to tune two parameters, which is somewhat a time taking process. In general, this is the algorithm we will use for any task. These algorithms will share the advantages of the KNN algorithm like easy to implement, no training

required as KNN is a lazy learning algorithm and it will make remember whole input and classify, ideal for fast adaptation, handling multi-class problems is simpler and some disadvantages like requiring large memory for the data, slow classification(testing).

4 Suggestions for some methods

As discussed above, the research paper selects K neighbors from the same class and will produce a local hyperplane that passes through the K points, for this, we are searching for the exact collinearities in the K points, sometimes we may not found it if the dimension of the class points is large, so in that case, we can generally produce a plane including all the collinear points and approximate it to the points which are not in the plane.

Say we have K points, First we will find the points which are collinear and which are not l points are collinear, K-l points are nearer to them and not collinear. we can first find the plane passing through the l collinear points and now we will calculate the distance of all remaining K-l points from the plane, we will assign a positive value to the distance of the points which are above/behind the plane and negative values below/front of the plane. Now we will add all the distances from the K-l points to the plane and have to check if it is zero or not. If that sum is zero then that's the required plane, if not then, we will change the plane, trying different points to get a plane that has the least sum of distance values from it. It is a complex process, but the plane we got will be a good one which is at an average distance from all the K-l non-linear points and it will be the optimum plane in my view. We can work on this update. On paper, it seems easy, but finding colinear points than making a plane and finding distances, changing the plane for optimum distance is a big process.

A suggestion regarding CKNN: as in CKNN it is difficult to calculate the distance of the point from the convex hull as it is not linear and includes complex processes and more equations. One way to simplify this problem is, we can project this point onto the plane, we want this distance, we will build a right angle triangle taking three lines as this projected line, a line from projected point to any line, line from the original point to the same line. Now we can find the distance of the point from the line and we can also calculate the distance of the projected point to the line, and from Pythagora's theorem, we can find the distance. This can be implemented with python, but with complex measurements. So it's a good solution to get the distance and can satisfy the disadvantage of the CKNN.

In further years this method has been extended to non-linear spaces and a new algorithm is derived from the existing one. Which is the non-linearization of the HKNN algorithm. as this algorithm is proposed in 2001, there are many other research papers and findings based on this paper one is,protein fold recognition with HKNN, link provided here : https://www.researchgate.net/publication/225214132_K-Local_hyperplane_distance_nearest_neighbor_algorithm_and_protein_fold_recognition Up to my knowledge, there are

no technical errors related to any components of ML or regarding the equations used in this paper.

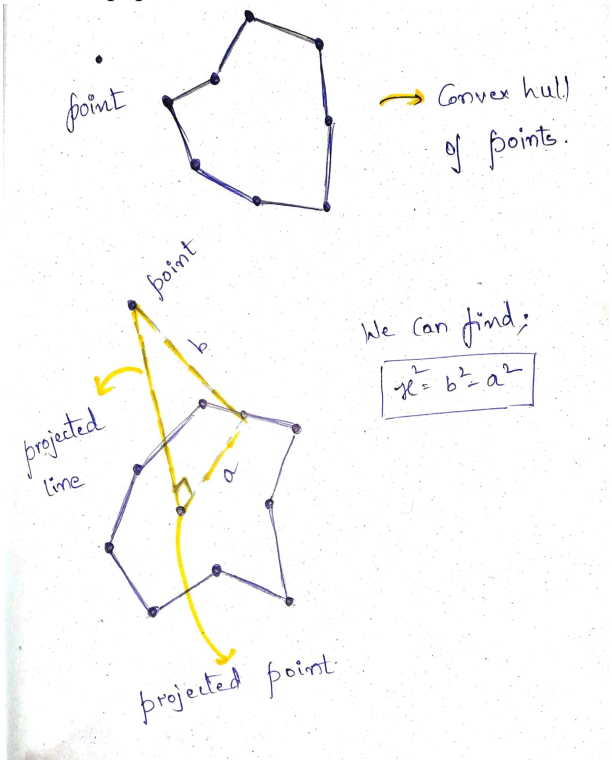

image showing how we can calculate the distance from the convex-hull to a point.

5 Review

Overall, this paper is good for beginners who want to do research in the field of machine learning, as this primarily focused on giving the basics of how we can improve a general algorithm. we will take the disadvantage of the existing algorithm and we will work in the way to minimize the error or to find a solution to the disadvantages. here we learn a new algorithm HKNN from the existing algorithm KNN, we compared two algorithms KNN and SVM, the disadvantages of the KNN over the SVM which makes the performance poorer, and learned how to make successive steps towards a new algorithm. but this newly proposes algorithm also has some disadvantages, so another algorithm is proposed CKNN which finds a solution to the disadvantages. so this is an iterative process of finding the optimum solution. at last, this paper increased my thoughts and knowledge on KNN and SVM, and HKNN.