K-Local Hyperplane and Convex Distance Nearest Neighbor Algorithms

Pascal Vincent and Yoshua Bengio

January 31, 2021

1 Technical update related HKNN

HKNN is itself a technical update for the algorithm KNN, but it is also not optimum, we can also make updates in HKNN algorithm. The proposed HKNN algorithm works by making hyperplanes with K points linearly. one major limitation of the HKNN algorithm is it is olny best if we consider euclidean formula as a distance metrics, one technical update of this algorithm is making it non-linear by using kernalization process, in which we will define subspace and all. We will discuss that here. for doing kernel trick we will project the given input data into a high dimensional space.but there is a problem with HKNN as in kernel PCA we will use mapped samples or we can say null spaces of covariance matrices of mapped samples, and we can't do it with HKNN. we know, local scatter matrix;

S_i=Σ $(x - x_m)$ $(x - x_m)^T$: x_m is the mean for the K nearest points.

and also local total scatter matrix, of all the neighbors nearest to the test sample. and we also define, new local subspace as intersection of the null space of that local class scatter matrix and range space, i.e $N(S_c)$ η $R(S_c)$. we will find projection matrices, projection matrix of the intersection is found by P_{int} = P_N + P_R. (here these matices N and R should be commute.) in order to get the projection matrix, we will project the nearest samples onto the Range space and find the null spaces.

non-linearization; this method will be like we will map the nearest nighbors into a higher dimensional space using a non-linear tranforming functions. we will find dot product of the spaces.

if there are C classes then there will be M=CK points in the neighborhood of test point. let the columns in mapped space represented as:

$\phi = (\phi^1, \phi^2.....)$

and for each i-th class we represent this as;

$\phi_i = (\phi_i^1, \phi_i^2.....)$

so our mapped local scatter matrix becomes:

$S^\phi = \Sigma(\phi(x) - \mu^\phi)(\phi(x) - \mu^\phi)^T$

$= (\phi - \phi I_k)(\phi - \phi I_k)^T$

where μ^{ϕ} is the mean of the mapped nearest neighbors. and local scatter matrix for each class becomes:

$S_i^{\phi} = \Sigma(\phi(x) - \mu_i^{\phi})(\phi(x) - \mu_i^{\phi})^T$

$= (\phi_i - \phi I_m)(\phi_i - \phi I_m)^T$

μ_i^{ϕ} is the mean of the mapped nearest neighbors in the i-th class. and here I_m and I_k are the matrices for which all elements are 1/M and 1/K.

so the kernel matrix for the data is given as : $G = \phi\phi^T = (G^{ij})_{i=1.C}$ and each submatrix is defined as; $G^{ij} = (k^{ij}_{mn})$

we have to find the basis vectors of the intersection matrix. for that we will follow the below process,

step-1 : we have to find K nearest points to the test point in each class.

step-2 : we will tranform all the nearest samples to the Range space or high dimensional space using the kernel PCA method.

step-3 : we will find the local scatter matrices of every class in the high dimensional spaces.

step-4 : we have to find the basis of the null space of the each scatter matrix. we can do it by eigen decomposition. (we will decompose the matrix int eogen values and eigen vectors). this is called matrix of basis vectors.

step-5 : we will define a feature vector which gives the dot product of the basis vector and the projected vectors, we will compare the distance metrix(euclidean, manhattan) between feature vector of a test point and the class vectors, and assign it to the minimum valued space.

This algorithm is called as NHKNN, non-linear k-local hyperplane and nearest neighbor algorithm.

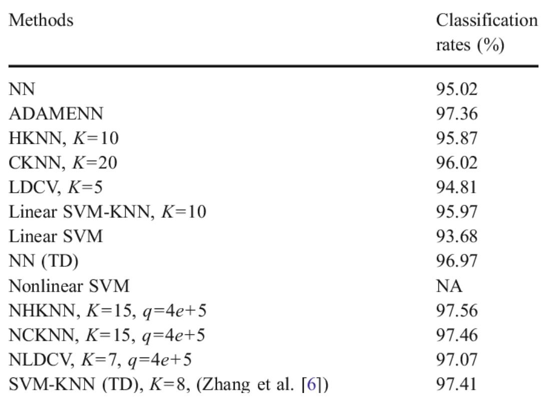
In the same way we can also make the CKNN function as a non-linear one, we will define convex-hull in the higher space by kernel PCA called as NCKNN.

with this update there are significant changes in the results of the experiments and these results are better than SVM.

we can also say this by considering that HKNN doesn't really works great with other distance metrices rather than euclidean distance, and in the proposed technique we are transforming to the high space using dot product technique and we are non-liearizing it. we can't say that everytime it is possible to make linear planes, so making the algorithm non-linear will help us to make that every classifiable data can be fit into this algorithm. By using this kernel trick and finding the matrices, we are finding te optimum plane for a HKNN in this process.

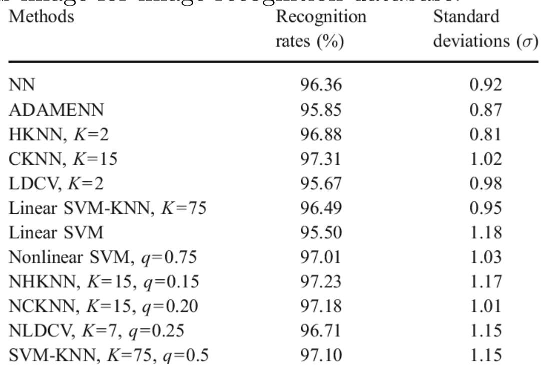
These are some images showing the accuracy rates of using different algorithms to a dataset.

NHKNN is the defined algorithm. image epicting accuracies for USPS database.



Methods	Classification rates (%)
NN	95.02
ADAMENN	97.36
HKNN, K=10	95.87
CKNN, K=20	96.02
LDCV, K=5	94.81
Linear SVM-KNN, K=10	95.97
Linear SVM	93.68
NN (TD)	96.97
Nonlinear SVM	NA
NHKNN, K=15, q=4e+5	97.56
NCKNN, K=15, q=4e+5	97.46
NLDCV, K=7, q=4e+5	97.07
SVM-KNN (TD), K=8, (Zhang et al. [6])	97.41

this image for image recognition database.



Methods	Recognition rates (%)	Standard deviations (σ)
NN	96.36	0.92
ADAMENN	95.85	0.87
HKNN, K=2	96.88	0.81
CKNN, K=15	97.31	1.02
LDCV, K=2	95.67	0.98
Linear SVM-KNN, K=75	96.49	0.95
Linear SVM	95.50	1.18
Nonlinear SVM, q=0.75	97.01	1.03
NHKNN, K=15, q=0.15	97.23	1.17
NCKNN, K=15, q=0.20	97.18	1.01
NLDCV, K=7, q=0.25	96.71	1.15
SVM-KNN, K=75, q=0.5	97.10	1.15

conclusion:this is a update in the implementation of the algorithm HKNN, actually upto this point HKNN as of my view HKNN is not implemented with scratch, this is not also in sklearn library or in any github code. In future i will try to implement it in the process of woc as a project.