# A Proposal for Local k Values for k-Nearest Neighbor Rule

Nicolás García-Pedrajas, Juan A. Romero del Castillo, and Gonzalo Cerruela-García

# Summary of the paper

**Muhammad Qasim**
Roll no: k19-1612
Subject: Machine Learning
Assignment 1

CONTENTS



## SUMMARY OF THE PAPER

In this paper, they have used three different datasets and three different measure of success to compare or prove how K-Nearest neighbor algorithm is performing well to others algorithms. Really K-NN working well with simplicity of this algorithm. There is a big problem to find the best or optimal value of k. although we can find best value with Cross Validation (CV) technique. In our approach we will try to find best k value directly connect with whole training dataset. Our approach is based on two hypotheses (one, assign value of k to each prototype its ideal number of neighbors to be used in its neighborhood, second obtain value associated with each prototype consider local performance value) form (x, y) where $x_i$ is prototype and $y_i$ is category. Our approach allowing the selection of local K value with a very simple and fast procedure in training time.

Although a few methods proposed some types of non-global K value we can set global K value with CV. We will used this algorithms:

**Algorithm 1** Outline of the Proposed Algorithm

**Data** : A training set $T = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$, $\mathbf{x}_i \in \mathbb{R}^D$, a minimum value of $k$, $k_{min}$, and a maximum value of $k$, $k_{max}$.

**Result** : The vector of local $k$ values **k**.

1 Obtain vector of 10-fold cross-validation accuracy for global values of $k$ from $k_{min}$ to $k_{max}$.

**for** *every* $x \in T$ **do**

  **for** $k = k_{min}$ *to* $k = k_{max}$ **do**

2  Obtain $eval(k)$ using eq. 1

  **end**

3  Obtain optimal local value of $k$

4  Assign optimal value of $k$ to $k_i$

**end**

5 Return **k**

Allots of method have introduced but K-NN still well performing to others old algorithms.

## OPTIMALLY LOCAL K VALUE FOR K-NN RULE

In our approach we have training data where all samples associated with this class. We will compare each testing sample with all training samples then we will assigned class to each testing sample which one has less loss value with training sample. If we found some testing sample those have equal loss value with more than two class with training data so we can select randomly one class or we can assign category based on 3 or 5 bottom loss values with training samples.

## EXPERIMENTAL SETUP

We will try to apply K-NN different rules like standard K-NN (Rule1), adaptive K-NN (Rule2) and symmetrical K-NN (Rule3). There are the major difference among the adaptive, standard and symmetrical k-NN rules is the distance measures. The standard way of obtaining the optimal value of K are labeled as Standard Training Method (STM) and our local approach proposed Training Method (PTM)

We will used 80 problem for problems for standard data sets 64 problems for class-imbalanced data sets. We used tenfold CV and getting good results.

We also used some other measure of success like:

1) Accuracy
2) Cohen's K measure
3) G-Mean = $\sqrt{sp.Sn}$
4) ROC/auROC

## EXPERIMENTAL RESULTS

We have used 3 KNN different rules and compare their results:

TABLE I

COMPARISON OF OUR APPROACH FOR THE THREE VERSIONS OF $k$-NN RULE IN TERMS OF ACCURACY AND $\kappa$ MEASURE

| Accuracy | | | | | | |
|---|---|---|---|---|---|---|
| | Rule 1 | | Rule 2 | | Rule 3 | |
| | STM | PTM | STM | PTM | STM | PTM |
| Average | 0.8281 | 0.8320 | 0.8021 | 0.8042 | 0.8293 | 0.8335 |
| Win/loss | | 49/22 | | 44/21 | | 50/23 |
| Wilcoxon | | 0.0034 | | 0.0031 | | 0.0017 |
| $R^+/R^-$ | | 2231.5/ 1008.5 | | 2237.0/ 1003.0 | | 2273.5/ 966.5 |

| $\kappa$ measure | | | | | | |
|---|---|---|---|---|---|---|
| | $k$-NN | | Rule 2 | | Rule 3 | |
| | STM | PTM | STM | PTM | STM | PTM |
| Average | 0.6333 | 0.6472 | 0.6015 | 0.6060 | 0.6352 | 0.6474 |
| Win/loss | | 54/20 | | 42/26 | | 51/27 |
| Wilcoxon | | 0.0000 | | 0.0192 | | 0.0006 |
| $R^+/R^-$ | | 2507.5/ 732.5 | | 2108.0/ 1132.0 | | 2337.5/ 902.5 |

TABLE II

COMPARISON OF OUR APPROACH FOR THE THREE VERSIONS OF $k$-NN RULES IN TERMS OF $G$-MEAN AND auROC

| $G$-mean | | | | | | |
|---|---|---|---|---|---|---|
| | Rule 1 | | Rule 2 | | Rule 3 | |
| | STM | PTM | STM | PTM | STM | PTM |
| Average | 0.7680 | 0.8162 | 0.7811 | 0.8023 | 0.7755 | 0.8287 |
| Win/loss | | 50/15 | | 37/23 | | 51/14 |
| Wilcoxon | | 0.0000 | | 0.0214 | | 0.0000 |
| $R^+ / R^-$ | | 1909.0/ 236.0 | | 1424.5/ 720.5 | | 1915.0/ 230.0 |

| auROC | | | | | | |
|---|---|---|---|---|---|---|
| | Rule 1 | | Rule 2 | | Rule 3 | |
| | STM | PTM | STM | PTM | STM | PTM |
| Average | 0.8539 | 0.8743 | 0.8376 | 0.8417 | 0.8649 | 0.8810 |
| Win/loss | | 39/26 | | 43/18 | | 37/28 |
| Wilcoxon | | 0.0084 | | 0.0198 | | 0.0166 |
| $R^+ / R^-$ | | 1476.0/ 669.0 | | 1429.0/ 716.0 | | 1439.0/ 706.0 |

TABLE III

MINIMUM, AVERAGE, AND MAXIMUM TRAINING TIMES IN SECONDS FOR PTM FOR STANDARD AND CLASS-IMBALANCED PROBLEMS

| | Standard datasets | | | Class-imbalanced datasets | | |
|---|---|---|---|---|---|---|
| | Minimum | Average | Maximum | Minimum | Average | Maximum |
| Rule 1 | 0.1 | 114.4 | 1814.2 | 0.0 | 33.9 | 819.8 |
| Rule 2 | 0.2 | 50.7 | 726.0 | 0.0 | 29.7 | 770.6 |
| Rule 3 | 0.6 | 28316.6 | 522552.0 | 0.0 | 5677.8 | 146943.0 |

## CONCLUSION AND FUTURE WORK

We can see that local value of k for KNN is performing better than the standard k-NN for both balanced and imbalanced data sets. Local value of k with every samples and using that k value for the nearest neighbor of the sample. We have compared standard approach of obtaining the best value of k using tenfold CV for the different versions of k-NN rule. We have also studied the average value for k obtained by our approach and compared then with the standard CV method. The value of k for each instance or sample whether an instance is removed would be simultaneously considered. Furthermore, the some philosophy presented here can be applied to the other forms of k-NNs, such as fuzzy k-NN and distance-weighted k-NN algorithms.