Introduction to Machine Learning
kNN Project Report

Chaabane Ouammou [*1] **Sahra Zeghoudi** [*1]

Abstract

This project aims to study the k-Nearest Neighbors (kNN) algorithm applied to waveform classification and to evaluate how data reduction and distance optimization techniques affect its performance. We first implemented a basic version of kNN to establish a baseline, which achieved an accuracy close to 86 %. Then, we applied two data reduction methods to simplify the training set. These methods significantly decreased execution time (up to 14 times faster) while maintaining similar accuracy levels. Finally, we compared our implementation with scikit-learn's Support Vector Classifier (SVC) (Cortes & Vapnik, 1995) to evaluate performance differences. Our results show that data reduction can effectively speed up computation without substantially sacrificing accuracy, while distance optimization methods showed limited impact.

1. Introduction

The k-Nearest Neighbors (kNN) algorithm is a simple yet powerful method for classification and regression tasks. It classifies a new instance by comparing it to the closest samples in the training data according to a chosen distance metric. Despite its conceptual simplicity, kNN can become computationally expensive when applied to large datasets, as every prediction requires distance computations with all training samples.

In this project, we will apply kNN to the waveform dataset to explore how its performance can be improved through data reduction and distance computation optimizations. Our objectives will be threefold: (1) implement a basic kNN classifier, (2) apply and evaluate data reduction strategies to reduce complexity and execution time, and (3) explore distance optimization techniques such as the triangle inequality and kd-tree structures. Finally, we will compare our implementation to scikit-learn's Support Vector Classifier (SVC).

[*]Equal contribution [1]Jean Monnet University, Saint-Etienne, France. Correspondence to: Chaabane Ouammou <chaabane.ouammou@etu.univ-st-etienne.fr>, Sahra Zeghoudi <sahra.zeghoudi@etu.univ-st-etienne.fr>.

Proceedings of the 42^{nd} International Conference on Machine Learning, Vancouver, Canada. PMLR 267, 2025.

2. Experimental set-up

2.1. Dataset

The dataset, `waveform.data`, consists of 5,000 samples, each described by 21 noisy features and divided into three waveform classes that overlap. We separate the dataset into two parts: 4000 data for training and 1000 data for testing. We also implement a PCA (Hotelling, 1933) algorithm to display the training data after reduction.

We shuffle the dataset, then separate it: the first 4000 for training and the other 1000 for testing.

2.2. kNN

kNN (k-Nearest Neighbors) (Cover & Hart, 1967) is used to classify or predict a value based on the similarity between data. When we want to determine the category or value of a new element, the algorithm searches the training set for the k elements closest to this new element. In the case of classification (our case), we do a majority vote on these k nearest neighbors and the prediction will be the most numerous label. Proximity is defined by a distance measure, which in our case is the Euclidean norm (we only consider the points in a hypersphere V with our element as center).

2.3. Data reduction

We applied two data reduction strategies: removal of the bias zone and elimination of irrelevant samples.

The first one consists in separating the dataset into two subsets S1 and S2. We classify S1 with respect to S2 using a 1-NN classifier, and we remove the misclassified examples from S1 and do the same thing with S2, using the new S1. We continue until stabilization.

The 2nd algorithm consist into taking a random point in the set and putting into a storage. We then classify the dataset with the storage. Every wrong prediction is added to the

storage and we repeat the last process until stabilization. We end up with the cleaned set.

These reductions are applied only to the training data. We test data reductions on 50 seeds (from 100 to 149 inclusive) and execution time because data reductions reduce complexity. For this we will use a 1-NN.

2.4. Optimizations of distance calculation

We implement the triangle inequality (Moreno-Seco et al., 2002) and kd-tree (Friedman et al., 1977) algorithms to reduce the algorithmic complexity.

The triangle inequality for 3 points A, B and C: $d(A, C) \leq d(A, B) + d(B, C)$. This property allows to reduce the number of distance calculations because it allows to eliminate points that are too far away without calculating their distance.

The kd-tree (k-dimensional tree) allows you to quickly find the points closest to a given point. The principle is to divide the space in two subspace at each step along a dimension. Each node in the tree represents a point and separates the other points into two groups: those with a smaller value and those with a larger value on that dimension. When searching for a neighbor, you traverse the tree to visit only the useful areas instead of comparing the point to all the others.

2.5. Compare the kNN classifier with another machine learning algorithm

We use the Support Vector Classifier (SVC) (Cortes & Vapnik, 1995) from Scikit-Learn to compare it to our model (k-NN triangle Inequality).

3. Analysis of the results

3.1. Our kNN

We first test whether our knn works well. With Figure 1, we can see that for approximately $k \geq 25$, accuracy stabilizes and therefore that our model converges.

We repeat the operation 50 times (with seeds ranging from 100 to 149) and we obtain Table 1. We notice that we have almost reached the maximum accuracy of 86 % (within 0.02 % for the accuracy at test time). But we are not overfitting because the accuracies at training time and test time are very close with a difference of 0.12 %. So, our kNN works well.

Table 1. Results for our kNN

BEST K	BEST ACCURACY (TRAINING)
70.5	85.82 %
FINAL ACCURACY (TEST)	AVERAGE TIME
85.94 %	152.7 S

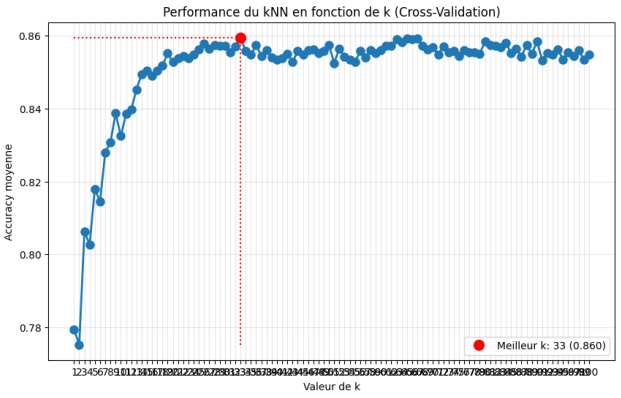


Figure 1. Example of kNN execution (accuracy at training time = 85.95%, accuracy at test time = 84.50%, seed = 42, k ranging from 1 to 100)

3.2. Data reduction

3.2.1. REDUCTION

In this part, we shuffle the dataset with seed = 42. Before the reductions, there are 4000 examples (Figure 2). After the first reduction, there are only 3201. And finally, after the second reduction: 707 examples (Figure 3). We perform a PCA (Hotelling, 1933) for display.

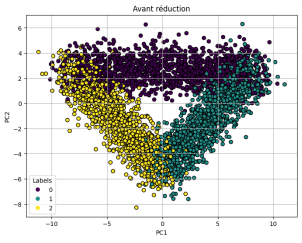


Figure 2. *
(a) Display of the 4000 training data

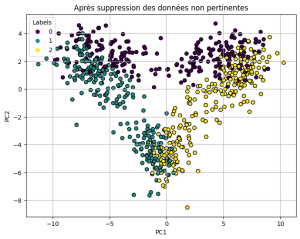


Figure 3. *
(b) Display of the 707 training data remaining after the two reductions

Figure 4. Comparison of training data (seed = 42) before and after reduction (PCA).

3.2.2. ACCURACY (1-NN)

We can see in Table 2, that in test time the accuracy is similar with or without reduction within 2 %. After the first reduction the execution is 1.6 times faster and after the second 14.6 times faster, there is a huge gain in execution time. So the reductions reduce the complexity and are therefore faster without losing too much accuracy.

Table 2. Average 1-NN results for before, after a first reduction and after the second

	WITHOUT RED.	1ST RED.	2ND RED.
BEST ACC. (TRAIN)	78 %	95 %	42 %
FINAL ACC. (TEST)	78 %	80 %	77 %
TIME	3.36 S	2.13 S	0.23 S

3.3. Optimizations of distance calculation

We tested all three methods with k ranging from 100 to 109. We have similar test time accuracies. But the time

differences are due to the use of `cdist` from the `scipy` library, which is very fast.

Table 3. (1) Best k (average), (2) Best accuracy (train), (3) Final accuracy (test), (4) Time

	KNN (1ST VERSION)	TRIANGLE INEQ.	KD-TREE
(1)	39.75	40.75	38.28
(2)	81.07 %	79.66 %	84.82 %
(3)	85.86 %	85.48 %	85.60 %
(4)	3.27 s	11.96 s	360.44 s

3.4. Compare the kNN classifier with another machine learning algorithm

We compare our kNN (triangular inequality) with Scikit-Learn's SVC. In Figures 5 and 6, we see that these SVC models are much faster than our kNN for similar accuracies: 1 % difference between the best and the worst. This difference is mainly explained by our implementation, such as the use of python loops.

Moreover, for the kNN the training is instantaneous and a few seconds for the SVC (for about 4000 data), but what really makes the difference is at the time of the test: for the kNN the complexity is $O(4000 \times 21) \approx 84000$ against $O(n\_sv \times 21)$ such that $n\_sv = 1513 << 4000$ (in our example).

Figure 5. Histograms of average execution time (and standard deviation) and average accuracies

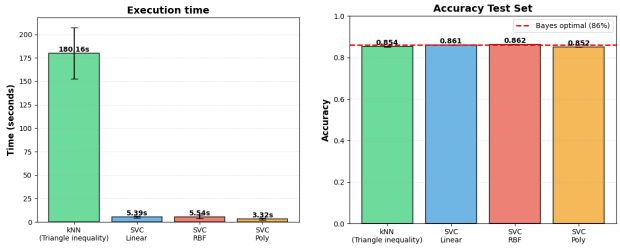
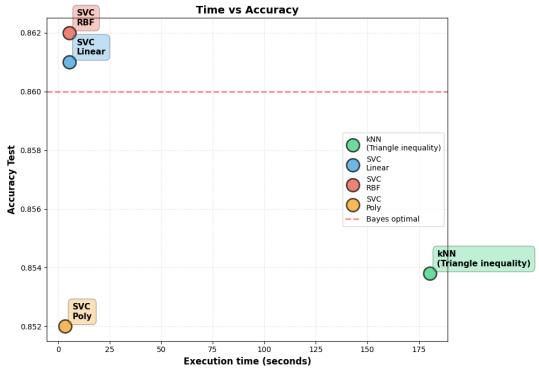


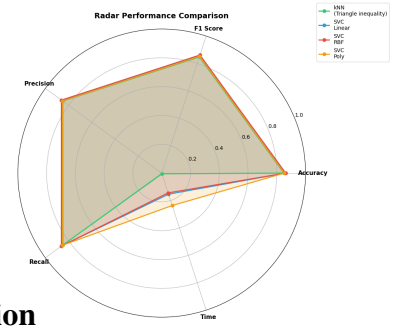
Figure 6. Our kNN vs SVCs



3.4.1. SCORES

We can see that on the Radar Chart, all scores are more or less similar for the kNN and the SVCs, except for the time, where a significant difference is observed. The best overall is the Polynomial SVC.

Figure 7. Scores comparison kNN vs SVCs



4. Conclusion

In this project, we compared the *k*-Nearest Neighbors algorithm and the Support Vector Classifier on the waveform dataset. Our experiments showed that although kNN provides good classification performance, it remains limited by its high computational cost at test time and its sensitivity to data dimensionality. Data reduction and distance optimization techniques helped improve efficiency (up to 14× faster) without significant loss of accuracy. Nevertheless, SVCs proved to be more efficient overall, mainly due to their smaller number of support vectors and better scalability in high-dimensional spaces.

Future work: Several improvements could further enhance our study. Additional dimensionality reduction techniques (e.g., LDA, t-SNE, ...) could be explored to mitigate the curse of dimensionality. Finally, testing other algorithms such as Random Forests, AdaBoost, or Neural Networks would allow a broader comparison and deeper understanding of the trade-offs between accuracy, complexity, and interpretability.

References

Cortes, C. and Vapnik, V. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995. doi: 10.1007/BF00994018.

Cover, T. M. and Hart, P. E. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13 (1):21–27, 1967.

Friedman, J. H., Bentley, J. L., and Finkel, R. A. An algorithm for finding best matches in logarithmic expected time. *ACM Transactions on Mathematical Software*, 3(3): 209–226, 1977.

Hotelling, H. Analysis of a complex of statistical variables into principal components. *Journal of Educational Psychology*, 24(6):417–441, 1933.

Moreno-Seco, F., Oncina, J., and Micó, L. Extending laesa fast nearest neighbour algorithm to find the k nearest neighbours. In Caelli, T., Singh, M., and Crawford, M. M. (eds.), *Proceedings of SSPR/SPR 2002 (LNCS 2396)*, pp. 718–724, Berlin Heidelberg, 2002. Springer-Verlag.