Waveform Classification using k-Nearest Neighbors and Data Reduction Techniques

Alexis Zawada [1] **Livio Singarin-Solé** [1]

Abstract

This report presents a KNN study on the `waveform.data` dataset. First, a k-Nearest Neighbors (kNN, (Cover & Hart, 1967)) classifier is implemented. Its hyperparameter k is tuned by cross validation. Then, to reduce computational cost, custom data reduction methods are applied such as the Condensed Nearest Neighbor algorithm. In addition, in order to shrink the feature set, Principal Component Analysis (PCA) is performed. Furthermore, comparison between other classifiers such as Logistic Regression and K-means is executed. The kNN, with an appropriate choice of k, demonstrates strong performance: Accuracy $\approx (1 - \varepsilon_b) \pm 0.02$, where ε_B denotes the Bayesian error.

1. Introduction

Machine learning provides robust tools for handling noisy classification problems. In this project, an exploration of methods such as kNN is conducted on the waveform dataset and it is demonstrated that their efficiency can be improved through data reduction. This work also tries to show in a simple way how these techniques can make classifiers faster and still keep sufficient accuracy.

2. Contributions

- Tuning of k in the kNN algorithm using cross-validation.
- Implementation of data reduction algorithms to reduce the amount of samples and features.
- Comparison between kNN, logistic regression and Kmeans.

3. An Introduction to k-nearest-neighbor (kNN)

Even though K-nearest neighbors is one of the simplest Machine learning algorithms, it can be very effective to handle both classification and regression problems. The K-nearest neighbors is a non-parametric learning method, which means it does not make the assumption that the data are drawn from a special distribution. The specificity of the kNN is to be considered a "lazy" algorithm. Indeed, it does not learn anything: at test time, the distances between the unseen data and the training dataset points are computed to find the k-nearest neighbor.Then ,the label assigned to the unseen data corresponds to the majority class of its neighbors. Despite the simplicity of kNN, on noisy datasets such as the Waveform dataset, it is possible to obtain great accuracy when its k-hyperparameter is fine-tuned carefully.

4. Tuning of k

4.1. Convergence of kNN towards the optimal Bayes classifier

A very strong result is kNN algorithm is asymptotically converging towards the optimal Bayes Classifier if these two following conditions are satisfied:

$$K \to +\infty \quad \text{and} \quad \frac{K}{n} \to 0$$

However, in practice , this property cannot be observed because the dataset size is finite. Therefore, to guarantee a good generalization at test time, a carefully tuned k is mandatory in order to achieve a good trade-off between underfitting and overfitting.

4.2. K fold Cross Validation method

The hyperparameter k is tuned by cross validation method. Two k-fold parameters commonly used in research papers (5 and 10) were considered, and finally a 10-fold cross-validation was selected, because this choice provides a good balance between computational cost and performance. The cross-validation process was done for a range of k-values around $\sqrt{n}$. Even if , in most cases, it is not the optimal value for k , it usually provides a good starting point before applying cross-validation to tune k. For each value of k in the range, the output is the mean accuracy over all the validation dataset during CV.

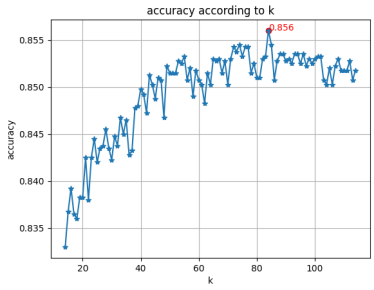

Figure 1. Validation accuracy according to the number of neighbors k. Accuracy increases with k up to around $k = 80$, where it reaches a maximum of 85.6% during cross validation. Beyond this point, the performance stabilizes or slightly decreases, indicating that too large k values may over-smooth the decision boundaries.(risk of underfitting)

5. PCA reduction

With a view to reducing the number of features and improving the comprehensiveness of the report, a dimensionality reduction method such as Principal Component Analysis (PCA, (Jolliffe, 2002),, is applied. PCA linearly projects the feature space $\mathbb{R}^n$ towards a lower-dimensional subspace $\mathbb{R}^t$, where n is the number of original features and $t < n$. The projection is realized with the Singular Value Decomposition (SVD), which identifies the principal directions capturing the largest variance.

Before applying PCA, the data are normalized using the z-score standardization, defined as $z = \frac{x-\mu}{\sigma}$, where μ and σ denote the mean and standard deviation of each feature. Reducing to a 2-dimensional feature vector is sufficient to preserve most of the variance and ensure effective visualization.

Figure 2. Illustration of the linear projection performed by Principal Component Analysis (PCA) with t=2.

6. Samples reduction

Despite the efficiency of kNN, its space and time complexity stay high compared to other classifiers. Indeed, it must compute all distances between a new point and the training points. Therefore, the objective is to shrink, at maximum, the amount of training samples used during inference.

6.1. Handling outliers and Bayesian Area

Outliers correspond to the data of the training set that are biased. The Bayesian Area represents the region of the space where it is not possible to always correctly classify the data, even with an optimal classifier. In an effort to remove those parts, the Repeated Edited Nearest Neighbor (RENN) algorithm is performed. This algorithm repeatedly removes samples that are classified incorrectly by their neighbors. The dataset is divided into two subsets, S_1 and S_2, which are used to cross-validate each other, allowing to delete outliers, and points close to the Bayes boundary.

6.2. Reducing Complexity via Condensed nearest neighbor rule

In most cases, not all training samples contribute equally to the prediction in a kNN classifier. Following this observation, the Condensed Nearest Neighbor (CNN, (Hart, 1968)) rule is applied. The idea is simple yet effective as it results in keeping only the samples misclassified by their neighbors. This procedure is typically performed after removing outliers and samples near to the Bayesian boundary.

6.3. Data cleaning

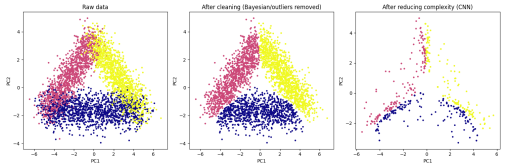

Figure 3. Evolution of the Training points after applying RENN (Repeated Edited Nearest Neighbor) and CNN (Condensed nearest neighbor)

7. Experimental Set-up

- Dataset: 5000 samples, 3 classes, 21 features (some noisy).
- Split: 4000 for training, 1000 for testing.
- $\varepsilon_B = 14\%$
- Validation: cross-validation on training data to select k.
- Metric: accuracy

8. Analysis of waveform.data

Before computing classification algorithms on the data, it is important to analyse its properties.

8.1. Distribution

The 3 classes are equally distributed (33 percent). Therefore, algorithms won't deal with imbalanced dataset issues and even if the precision or recall metric are displayed in the code, comparing only accuracy should be sufficient.

8.2. Features Correlation

Observing feature correlation helps remove redundancy. Features that are dimensionally near are slightly correlated but not not strong enough to justify removal.

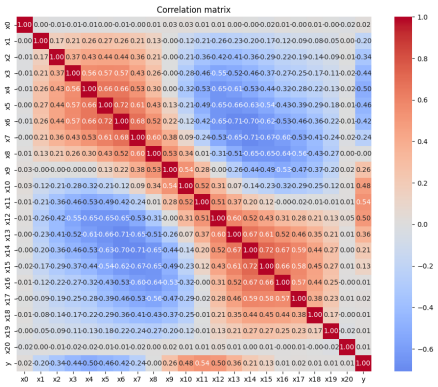

Figure 4. Correlation Matrix for between all features of waveform.data.

9. Analysis of the results of 1-NN

Table 1. Performance comparison of 1-NN ($k = 1$) with and without PCA.

Config.	Acc.	Time (s)	Data shape
Without PCA			
1-NN	0.783	0.480	(4000, 21) / (1000, 21)
1-NN with CNN	0.763	0.129	(610, 21) / (1000, 21)
With PCA			
1-NN	0.831	0.283	(4000, 2) / (1000, 2)
1-NN with CNN	0.870	0.076	(177, 2) / (1000, 2)

As shown in Table 1, applying PCA and the Condensed Nearest Neighbor (CNN) method significantly improves computing efficiency. The CNN method reduces the training set size, leading to much faster inference while preserving similar accuracy.

10. Comparison with others machine learning algorithms

In this part, an implementation of Logistic Regression (Hosmer et al., 2013), and Kmeans is conducted. All the results are plotted for comparison.

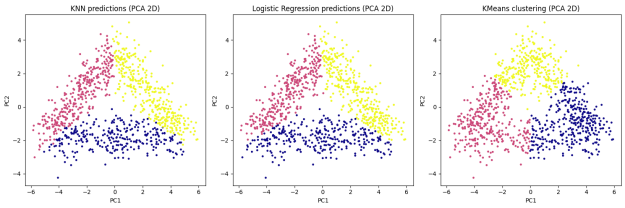

Figure 5. Comparison of kNN (k=67), Logistic Rgression and Kmeans.

Table 2. Comparison of performance metrics between kNN ($k = 84$) and Logistic Regression.

Classifier	Accuracy	Precision	Recall
kNN ($k = 84$)	0.879	0.880	0.878
Logistic Regression	0.872	0.872	0.871

At first view, the logistic regression performs similary to kNN. However, KMeans fails to properly separate the three classes.

In addition, it is not possible to directly compute an accuracy score, as it is an unsupervised algorithm, so in this case, only the plots are compared.

11. Conclusion

This report summarizes the study of kNN efficiency and complexity after introducing data reduction techniques for a balanced classes dataset (`waveform.data`). By reducing amount of samples using CNN (Condensed Nearest Neighbor) and RENN (Repeated Edited Nearest Neighbor) methods, kNN inference is considerably simplified, offering huge gain in time and memory. Furthemore, PCA (principal component analysis) allows to shrink the features vector to a 2D vector without losing any performance. In the last part, a comparison with other classifiers demonstrates the robustness of kNN compared to other algorithms, such as K-Means. Future work should focus on the implementation of KD-Tree algorithm in order to speed-up calculation of kNN.

References

Cover, T. M. and Hart, P. E. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 1967.

Hart, P. E. The condensed nearest neighbor rule (corresp.). *IEEE Transactions on Information Theory*, 1968.

Hosmer, D. W., Lemeshow, S., and Sturdivant, R. X. *Applied Logistic Regression*. Wiley, 2013.

Jolliffe, I. T. *Principal Component Analysis*. Springer, 2002.