Tuning a classifier to predict waveform categories

Ugo LABBÉ [*1] **Alejandro CARVAJAL MONTEALEGRE** [*1]

Abstract

This paper aims at identifying the most optimal classifier for categorizing wave-forms into three distinct classes. Our approach involves using different methods, tuning different hyper-parameter and observing the behavior of modified data-sets and how these can modify the output. The goal is to create a classifier with an error as close to the optimal Bayes classification rate of 86% as possible.

1. The different classifiers

The different hyper-parameters to tune the most optimal KNN Classifier were tested. However, other classification methods different to KNN were explored.

1.1. K-Nearest Neighbor

The KNN classifier works by finding the k-training examples in its data-set that are closest to a given input and then classifies the input based on the majority class of those k neighbors. The ”closeness” is typically measured using a distance metric. The choice of k and the distance metric are important hyper-parameters that impact the performance of the kNN classifier.

1.2. Logistic Regression

Logistic Regression is a binary classification algorithm that uses the logistic function to transform a linear combination of input features into a value between 0 and 1, representing the probability of the instance belonging to the positive class. It can be extended to handle more than two classes through techniques like ”One-vs-Rest” or ”One-vs-One”.

[1]Master MLDM, University Jean Monnet, Saint-Etienne, France, Country. Correspondence to: Ugo LABBÉ <ugo.labbe@etu.univ-st-etienne.fr>, Alejandro CARVAJAL MONTEALEGRE <alejandro.carvajal.montealegre@etu.univ-st-etienne.fr>.

Proceedings of the 39^{th} International Conference on Machine Learning, Baltimore, Maryland, USA, PMLR 162, 2022.

1.3. Support Vector Machine

Support Vector Machine finds the hyperplane that best separates different classes in the feature space. The ”support vectors” are the closest data-points to the hyperplane and helps at defining its position. The algorithm aims to maximize the distance between the hyperplane and the nearest data points of each class. SVM also uses a kernel trick to map the data into a higher-dimensional space, when a linear separation is not possible, making non-linear separations feasible.

1.4. Naive Bayes Classifier

The Naive Bayes classifier assumes that features are conditionally independent given the class label. The classifier calculates the probability of a particular class for a given set of feature values by multiplying the probabilities of individual features given that class. The class with the highest probability is assigned to the input.

1.5. Random Forest

Random Forest constructs a multitude of decision trees during training, and outputs the mode prediction of the individual trees. Each tree is built using a random subset of the training data and a random subset of features at each split. This randomness helps to decorrelate the trees, reduce overfitting and improve generalization. During prediction, the input passes through all the trees, and the final output is determined by aggregating the individual predictions.

1.6. Decision tree

A Decision Tree recursively splits the dataset based on the most significant feature at each node, creating decision nodes and leaf nodes. The splitting process continues until a stopping criterion is met, such as a predefined tree depth or a minimum number of samples in a leaf. Each leaf node represents a class label.

1.7. Classifiers results

To optimize the performance of the classifiers, we employed a 10-fold cross-validation process and tuned different hyper-parameters. The classifiers were trained on 80% of the data-set, and the resulting scores are presented below

	Accuracy	Hyper-parameters
K-Nearest Neighbor	85.82%	neighbors=100, p=1, Metric=euclidean
Logisitc Regression	87.15%	C=0.01, Max iteration=10, Solver=newton-cg
Support Vector Machines	86.95%	C=2, kernel=linear, Gamma=scale
Naive Bayesian Classifier	81.45%	Alpha=0
Random Forest	84.92%	Decision trees=200
Decision Tree	76.52%	Max depth=None, Max features=Nonce, Min Samples leaf=8, Min Samples leaf=10

Table 1. Accuracy of different classifiers explored and its hyper-parameters

2. Altered data sets and Experiments

2.1. Reducing Complexity

2.1.1. Condensed Nearest Neighbor

A way to reduce complexity is to use a data reduction algorithm such as the Condensed Nearest Neighbor. It selects a representative subset of data points and removes the others. Using this process, the training set was reduced from 3999 to 387 data points.

The best accuracy using the CNN was found to be at 20 neighbors, with an accuracy of 85.70% on test set. The best score found for the non-reduced training set was at 40 neighbors, with an accuracy of 86.60%.

2.1.2. Principle Component Analysis (PCA)

Another way to reduce complexity is by decreasing the number of features using Principal Component Analysis (PCA), which helps in capturing the most significant variability in the data while reducing its dimensionality. Applying PCA to the training set, reduced the number of features from 21 to 15, explaining 91% of the total variance with these components. The highest accuracy was achieved with 40 and 50 neighbors, scoring 86% accuracy on the test data.

2.2. Speeding up the calculation

Another experiment done is speed up the calculation of a 1 nearest neighbor algorithm. For that, we can think of computing a KD-Tree. To achieve this, we can consider employing a KD-Tree, a data structure that organizes points in a multidimensional space, enabling efficient queries and proving particularly valuable in high-dimensional datasets. Using a KD-Tree to find the nearest neighbor on our data set takes 0.105 seconds while a 1-NN takes 0.160 seconds. The KD-Tree decreased the computation time.

2.3. Artificial imbalance

The waveform data-set is balanced as it contains almost the same amount of observations per class. In this section, we introduced artificial imbalance to see the performance of the classifiers with this new configuration. We decided to imbalance the first category using the following ratios to scale it : 0.5%, 1%, 5%, 10%, 50% and 100%. For this section, it was used the F-Measure to catch a score, as it combines precision and recall into a single value.

We tuned hyper-parameters of a k-nearest neighbor and a support vector machine classifiers using a grid-search and a cross validation of 5 fold .

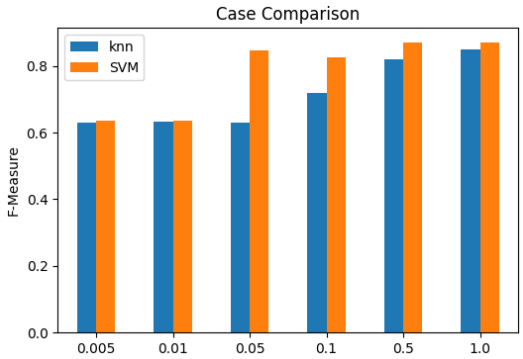

Figure 1. F-Measure on K-NN and SVM on Imbalanced data-set

3. Conclusion

After tuning different classifiers, 3 seems to stand out, the K-NN, Logistic Regression and SVM. The last two, had an accuracy above 86%. We could then consider these classifiers to be the best to retrieve the class of a specific waveform as they are very close to the optimal Bayes classification rate. Reducing complexity of the data either using a PCA or a CNN algorithm give promising results, demonstrating performance closely aligned to an unmodified data-set. It was also found out that computing a KD-Tree is effective in producing results faster than a One Nearest Neighbor algorithm. Also, while creating artificial imbalance, we could easily see that a Support Vector Machine algorithm perform better than a KNN classifier in any case.

References

[1] Lars Buitinck, Gilles Louppe, Mathieu Blondel, Fabian Pedregosa, Andreas Mueller, Olivier Grisel, Vlad Nic-

ulae, Peter Prettenhofer, Alexandre Gramfort, Jacques Grobler, Robert Layton, Jake VanderPlas, Arnaud Joly, Brian Holt, and Gaël Varoquaux. API design for machine learning software: experiences from the scikit-learn project. In *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, pages 108–122, 2013.

[2] Adrien Guille. Lecture notes in data mining, 2021.

[3] Marc Sebban. Lecture notes in introduction to machine learning, 2023.