Tuning a classifier to predict waveform categories

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

Abstract

This paper is aiming at finding a best classifier to categories waveform into 3 classes. Using different methods, tuning different hyper-parameter and observing the behaviour of modified data sets helped us finding solutions. The goal is to be the closest to the known Bayes error of 86%.

1. The different classifier

1.1. K-nearest neighbour

The k-Nearest Neighbors classifier works by finding the k training examples in its dataset 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 like Euclidean distance. 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 models the probability of an instance belonging to a particular class. It 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".

1.3. Support Vector Machine

Support Vector Machine works by finding the hyperplane that best separates different classes in the feature space. The "support vectors" are the data points closest to the hyperplane and play a crucial role in defining its position. The algorithm aims to maximize the margin, which is the distance between the hyperplane and the nearest data points of each class. In cases where a linear separation is not possible, SVM can use a kernel trick to map the data into a higher-dimensional space, making non-linear separations feasible.

1.4. Naive Bayesian Classifier

The Naive Bayes classifier assumes that features are conditionally independent given the class label, which is a "naive" assumption. 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 operates by constructing 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, reducing overfitting and improving 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. The decision-making process involves traversing the tree from the root to a leaf based on feature values, ultimately assigning the corresponding class label.

1.7. Classifiers results

Tuning the different classifiers was done by a cross validation process (10 fold) over the different hyper-parameters and training them on 80% of the dataset, we get the following accuracies:

[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.

	Accuracy	Hyper-parameters
K-Nearest Neighbour	0.8582	neighbours=100, p=1, Metric=euclidean
Logisitc Regression	0.8715	C=0.01, Max iteration=10, Solver=newton-cg
Support Vector Machines	0.8695	C=2, kernel=linear, Gamma=scale
Naive Bayesian Classifier	0.8145	Alpha=0
Random Forest	0.8492	Decision trees=200
Decision Tree	0.7652	Max depth=None, Max features=Nonce, Min Samples leaf=8, Min Samples leaf=10

2. Altered data sets and Experiments

2.1. Reducing Complexity

2.1.1. Principle Component Analysis

To reduce the complexity we decided to reduce the numbers of features, using Principal Component Analysis, which helps in capturing the most significant variability in the data while reducing its dimensionality, making it useful for visualization, noise reduction, and improving the efficiency of machine learning algorithms

Applying a PCA on the training set we reduced the number of features from 21 to 15, getting 91% of the variance explained by these components. The best obtained accuracy is at 40 and 50 neighbors, with a score of 0.86 on the test data.

2.1.2. Condensed Nearest Neighbor

Another 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, we managed to reduce the training set from 3999 to 387 data points.

The best accuracy using the CNN was found to be at 20 neighbor, with a score of 0.857 on test set. The best score found for the unreduced training set was at 40 neighbor, with a score of 0.866.

2.2. Speeding up the calculation

Another interesting experiment is to speed up the calculation of a 1 nearest neighbor algorithm. For that, we can think of computing a KD-Tree. A data structure that organizes points in a multidimensional space, making possible effective queries, being useful in high dimensional data set.

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

Our data set being quite balanced between the categories (between 1647 for the smallest category to 1695 for the biggest), it could be interesting to generate imbalancy and observe the behaviour of classifiers.

Instead of accuracy, we will use the F-Measure to catch a score, as it combines precision and recall into a single value.

We decided to create an imbalancy on the first category using different ratio to scale it : 0.005, 0.01, 0.05, 0.1, 0.5 and 1.

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 .

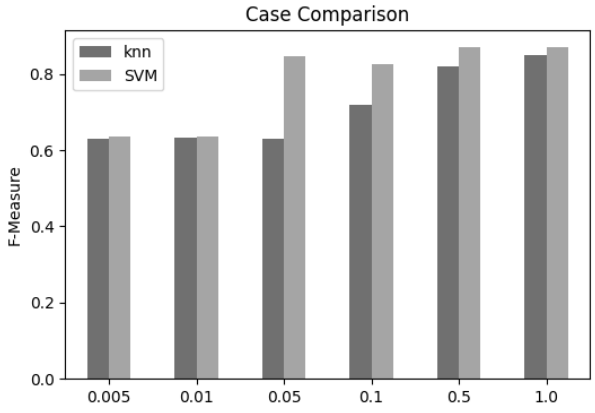

Figure 1. F-Measure on K-NN and SVM on Imbalanced data

3. Conclusion

After having tuned different classifiers 3 seems to stand out, the K-NN, Logistic Regression and SVM, the two last having an accuracy above the 86% Bayes error of the data set. We could then consider these classifiers to be the best to retrieve the class of a specific waveform. Reducing complexity of the data either using a PCA or a CNN algorithm give promising results as they give results really close to an unmodified data set. We also found that computing a KD-Tree is effective to compute a result faster than a One Nearest Neighbor algorithm. Also, while creating artifical imbalance, we could easily see that a Support Vector Machine algorithm perform better than a KNN classifier in any case.