Unlabeled Consensus Modeler: Leveraging Voting Ensemble's Consensus on Unlabeled Data

Dong Cao
Computer Science Department at Earlham College
Richmond, Indiana, USA
dcaohuu18@earlham.edu

ABSTRACT

Voting is an important Ensemble Learning technique. However, there has not been much discussion about leveraging the base classifiers' consensus on unlabeled data to better inform the final prediction. My proposed method identifies the data points where the ensemble reaches consensus and where conflict arises in the unlabeled space. A meta weighted KNN model is trained upon this half-labeled set with the labels of the consensus and the conflict points marked as "Unknown," which is treated as a new, additional class. The predictions of the meta model are expected to better inform the decision of the ensemble in the case of conflict. This research project aims to implement my proposed method and evaluate it on a range of benchmark datasets.

KEYWORDS

Machine Learning, Voting Ensemble, Semi-supervised Learning

1 INTRODUCTION

Ensemble learning has received great attention from the Machine Learning community, as the aggregated output of multiple learners is often better than that of any single one of them. For classification problems specifically, several methods have been proposed to combine multiple classifiers' predictions: Voting, Bagging, Boosting, Stacking, etc. Among these methods, Voting is very important not only because it is a simple, intuitive, and effective method in and of itself, but also because it plays the role of determining the collective prediction in other ensemble frameworks, like Bagging and Boosting. However, there has not been much discussion about leveraging the consensus of the base classifiers on unlabeled data in order to better inform the final prediction. My proposed method identifies the data points where the ensemble reaches consensus and where conflict arises in the unlabeled space. A meta weighted KNN model is trained upon this half-labeled set with the labels of the consensus and the conflict points marked as "Unknown," which is treated as a new, additional class. The predictions of the meta model are expected to better inform the decision of the ensemble in the case of conflict. I named this method Unlabeled Consensus Modeler (UCM). The motivation is that the points agreed upon by the base classifiers represent patterns that we can confidently extract from the training set. However, the training set can also contain noise that leads the base classifiers astray and makes them disagree with each other. In the unlabeled space, the agreed patterns may be clearer around a point of dispute and we can use this information to strengthen the prediction for that point. This research project is an attempt to implement UCM and compare it with Simple Majority Voting in terms of their performance on a variety of benchmark datasets under some conditions.

In the next section, I will review the related and background knowledge in two domains: Voting Ensemble and Semi-supervised Learning, as well as the differences between UCM and the work introduced. The third section delves deeper into the theoretical motivation and the specific problem that my method targets. It also presents a formal design of UCM and how its components are implemented. Finally, the experimental setup and evaluation framework are discussed in the fourth section.

CS488 Senior Capstone, Fall 2021, Earlham College

2 RELATED WORK

2.1 Voting Methods

Majority Voting: Majority Voting or Plural Voting is probably the most well-known voting system due to its straightforwardness. A class is assigned to a sample if it receives a majority of votes from the base classifiers. However, there are different definitions of "majority." Depending on the situation, it can mean unanimity, simple majority (i.e. more than 50% of the base classifiers agree on a label). Yet, the most common approach has been that whichever class receiving the most votes "wins" and becomes the ultimate prediction. Simple Majority Voting makes a lot of assumptions about the relative accuracy of the classifiers and each classifiers' performance with respect to a particular class [10]. In reality, most of these assumptions are not accurate. However, the simplicity and efficiency of this method still makes it one of the favorite options of Machine Learning practitioners.

Weighted Voting: Another popular voting scheme is Weighted Voting. Weighted Voting drops one of the assumptions made by Majority Voting [10]. Instead of considering the predictions of the base classifiers equally likely to be accurate, Weighted Voting attaches different weights to the classifiers' predictions based on their performance in the training set. One of the most well-known examples of this approach is the voting system of AdaBoost, which trains a number of weak learners, weights them differently based on their error rates, and aggregates their predictions by taking into account these weights [6]. Other techniques make use of fuzzy sets [3], particle swarm optimization [8], genetic algorithm [12], or instance-wise weight assignment based on a classifier's relative performance with respect to others' [5].

Support Function: With a support function, instead of being confined to a single output class, a base classifier can provide their predictions in terms of the likelihood of a sample belonging to each of the available classes. The term "Support Function" is used by Woźniak et al. in their survey of classifiers combination [15]. One of the widely used types of likelihood is the *a posteriori* probability. Kittler et al. have proposed a variety of rules by which a class's *a posteriori* probabilities from different base classifiers can be combined [9]. Later work has focused on comparing the effectiveness of these rules under various conditions [1]. The value of the support function can also be the rankings of the classes. In this case, the method is known as Borda Count, which outperforms Majority Voting in some experiments [11].

While the aforementioned methods tackle the problem differently, they all agree that it is critical for the ensemble to be diverse. This concept of diversity can be interpreted and measured in a variety of ways. Usually, it is important that the classifiers don't make the same mistake together. Some diversification strategies are using different underlying algorithms for the base classifiers or different feature sets [9], and bootstrapping.

2.2 Semi-supervised Learning

Because UCM utilizes information in both the labeled and unlabeled spaces, it can be linked to semi-supervised learning. However, since semi-supervised learning is a broad field, I will only focus on the areas that are relevant to making use of the ensemble's consensus on unlabeled data. Before I go into the details of each area, let us quickly touch upon the rudiments of semi-supervised learning. The big problem that semi-supervised learning tries to solve is that labeled data for training is often insufficient and difficult to acquire while unlabeled data is abundant. Semi-supervised learning aims to fully exploit the few labeled samples available to extract patterns from the pool of ample unlabeled data [14].

Inductive versus Transductive: The landscape of semi-supervised learning methods comprises of two major approaches: Inductive and Transductive. The goal of an inductive framework is to build a mechanism that can independently predict unlabeled samples one by one. This goal is shared with most of the supervised algorithms but the training process of an inductive algorithm takes in both labeled and unlabeled data. On the other hand, a transductive method seeks to optimize the predictions for each space of data. This space contains samples that are either labeled or unlabeled and a transductive algorithm attempts to use the distribution of the entire space to provide a set of predictions for all data points. In other words, the input for a transductive algorithm is the whole data space, not a single data point [14]. In this aspect, UCM is similar to the transductive approach when the meta model needs to be fed the complete unlabeled set. However, while most transductive algorithms use graph theory to model the similarity among the data points [14], UCM looks to draw a connection between the patterns in the training set and those in the unlabeled set via inspecting the consensus of the base classifiers.

Tri-Training: Tri-Training is an inductive method that uses three classifiers, all of which are trained upon the same complete dataset. When it comes to leveraging unlabeled data for "refinement," a classifier is given a sample to train with the label agreed upon by the other two [16]. A variation of Tri-Training is Multi-Train when more than three classifiers are used and a sample is accepted for the refinement of one classifier if a majority of the rest of the classifiers return the same label [7]. It is not difficult to point out the similarities between my idea and that of Tri-Training, when my meta KNN model learns from the data labeled based on the base classifiers' consensus. However, there are fundamental differences between UCM and Tri-Training:

- As mentioned, Tri-Training falls under the inductive approach while UCM is generally transductive.
- My meta KNN model does not learn from the labeled data in the training set.
- There is no co-training. In other words, my meta KNN model does not affect the base classifiers in any way. Hence, there is no refinement of the base classifiers using unlabeled data.
- My meta KNN model also takes into account the uncertainty of the nearby conflict points.

3 UNLABELED CONSENSUS MODELER

3.1 Theoretical Motivation

UCM is expected to take advantage of the collective decisions of a voting system and summarize how strong these collective patterns are in the unlabeled set to help with the classification of the conflict points. In Figure 1, the graph on the left presents the training set, which has two classes "A" and "B." On the right is the graph of the unlabeled set, which is pseudo-labeled based on the majority decisions of two classifiers cl_1 and cl_2. The circled question mark indicates a conflict point where cl_1 and cl_2 disagree. It is also shown in the training set for the sake of convenient comparison.

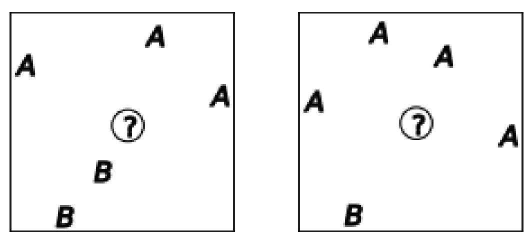

Figure 1: Training (left) and unlabeled (right) sets

In the unlabeled set, it is apparent that the conflict point is more likely to have the label "A." However, the additional B in the training set causes confusion and dissent between cl_1 and cl_2. UCM settles this dispute by adding another voice based on the consensus in the unlabeled space. The feasibility of UCM rest on two assumptions. First, the pseudo-labels inferred from consensus are reliable. This assumption can be satisfied with a diverse ensemble. If base classifiers with different learning "lenses" all agree on the label of a point then this label is credible. The second assumption of UCM is that the distribution of the unlabeled set is more trustworthy and contains less noise than the training set, especially around the point of dispute.

It is also possible that in the unlabeled space, there are other conflict points around the point in question. Figure 2 signifies the other conflict points with unringed question marks. These points are assigned the class "Unknown."

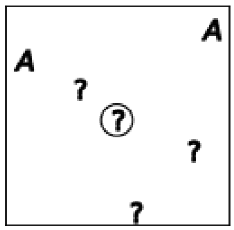

Figure 2: Accounting for uncertainty in the unlabeled set

In Figure 2, although there are two A's near the point in question, UCM also takes notice of the three unknown samples around it and is less positive that the point in question also has the label "A." The analysis of the theoretical motivation of UCM makes it clearer that the distinctions between UCM and Tri-Training reflect the different objectives that the two methods are pursuing. While Tri-Training, a representative of semi-supervised learning, tackles the lack of labeled data, UCM aims at detecting fake patterns that exist in the training set but not in the unlabeled set, thereby reducing the chance of overfitting. Even though the approach of UCM is semi-supervised, it is intended to serve supervised frameworks. There is no need for refinement using unlabeled data since the training data should be sufficient for the base classifiers to perform decently on their own and UCM will only play the role of assisting them with making the final prediction where discord occurs. Another reason for no retraining of the base classifiers is that many semi-supervised techniques suffer from degradation due to their biased conjecture about the unlabeled data [17]. By keeping the opinions of the base classifiers intact and only putting another voice on top of their opinions when needed, UCM is anticipated to be less prone to the issue of degradation.

3.2 Design and Implementation

Figure 3 is the architectural diagram of UCM. I will now dissect each of its components, most of which are implemented with the Scikit-learn open source library [13].

Basic preprocessing: Missing values are imputed using the median for numerical variables and the mode for categorical features. After that, one-hot encoding is applied to satisfy many algorithms' requirement that categorical data be represented as numeric values. Although it may make sense for some categorical features to be transformed according to an ordinal scale, the vast number of evaluation datasets and the fact that they spread across a range of specialized domains make it difficult to determine the ordinality of each categorical feature. More importantly, the objective of this research is not to achieve the best performance on the benchmark datasets. Rather, it is to carry out a comparative experiment of two frameworks and the choice of the preprocessing method does not interfere with this objective significantly. In addition to the categorical encoding, all numerical features are standardized and clamped to the same scale.

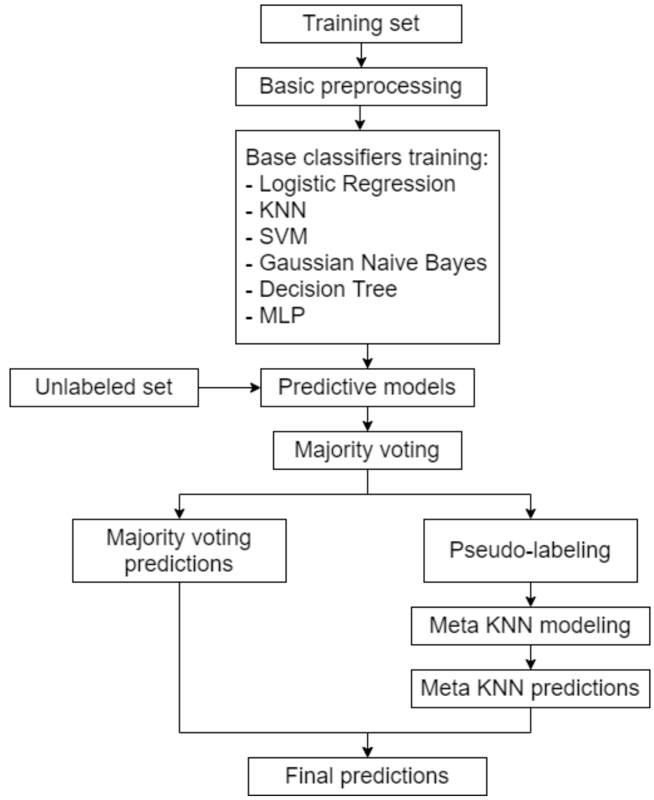

Figure 3: UCM framework

Base classifiers training: I follow the strategy of using different underlying algorithms to diversify the ensemble. There are six base classifiers, each of which corresponds to one algorithm in the diagram. The classifiers are constructed from Scikit-learn's standard implementations of the listed algorithms with the default hyperparameter values. Only the `n_jobs` argument, if available, is set to `-1` to enable parallelization. For the Multilayer Perceptron (MLP) algorithm, there is only one hidden layer and the number of hidden nodes is $\frac{n+1}{2}$ where n is the number of features, i.e. the number of input nodes. This is based on the suggestion that the number of hidden neurons should be "somewhere between the input layer size and the output layer size." [2] All other hyperparameters are set to the package's default values, including the ReLU activation function.

Majority voting: After the unlabeled points are predicted by the base classifiers, their predictions go through majority voting. The predictions of the majority voting system are in terms of probabilities. For instance, if a sample is classified as "A" by five out of six learners and as "B" by only one learner then the prediction will be $\frac{5}{6}$ "A" and $\frac{1}{6}$ "B." UCM can work with any voting schemes whose output can be interpreted probabilistically and if there is a way to determine consensus. Thus, it is well-suited with a support function. Nevertheless, a more complicated voting technique is unnecessary since UCM is not a voting system on its own but is built upon an existing voting framework. Simple majority voting, therefore, is good enough for assessing UCM and its contribution, if any, to the improvement of the voting ensemble. Another advantage of using majority voting is, potentially, plenty of ties that will be helpful for evaluating UCM against the baseline of random guessing.

Pseudo-labeling: A conflict threshold needs to be set. For example, if a data point is agreed upon by at least five out of six (or approximately 83%) classifiers then it is labeled as the majority's decision. Otherwise, it is indicated as "Unknown."

Meta KNN modeling: A distance-weighted KNN is then applied to the pseudo-labeled set to predict each of the unknown points. For each of these points, the meta model also considers the other unknown samples around it. The model's output is the probabilities of the point belonging to one of the original classes or the class "Unknown", which consolidates the amount of uncertainty into the predictions and serves as a regulating factor. This is why it is crucial for the predictions to be probabilistic.
KNN is chosen to be the algorithm of the meta model because it is an intuitive way of thinking about the dissimilarity in distribution between the training set and the unlabeled set. Other algorithms that make a strong use of the data distribution and that can produce probabilistic predictions like SVM may also be good candidates. However, for each unknown point to be classified, it needs to be removed from the pseudo-labeled set before the learner is fit. KNN, as a lazy learning algorithm, nicely meets this "leave-one-out" requirement, although the relaxation of this requirement may be acceptable for some eager learning algorithms.

Producing the final predictions: For each sample and class, the two probabilities from majority voting and meta KNN modeling are added and whichever class receives the highest probability score becomes the ultimate label for that sample. I may experiment with other algebraic operations but addition is selected at this point because of its simplicity.

4 EXPERIMENT AND EVALUATION

4.1 Datasets

I plan to reuse the 73 public benchmark datasets from the UCI repository that Kuncheva et al. employ in their study of major voting systems [10].

4.2 Evaluation

The performance of UCM is compared with that of mere majority voting to see if the additional technique of modeling the consensus brings any benefit. I intend to particularly examine the effectiveness of UCM in breaking ties, compared with random guessing. The accuracy rate and F_1 score will be the metrics due to their popularity and applicability and will be estimated using cross-validation to reduce bias, especially with small datasets. The results will be evaluated with one of the statistical tests recommended by Demšar for comparing two classifiers over multiple datasets [4] and that have been widely adopted.
Another interesting experiment would involve using only five base classifiers with KNN versus without KNN to see the effect of the meta model being different from any of the base classifiers and check if the meta model biasedly favors the base classifier of the same learning algorithm, i.e. whether it agrees with this base classifier most of the time, especially when the base classifier is wrong. The results will also provide a sense of the influence of the meta model in 3-2 situations and whether it can overturn the decision of majority voting.

5 CONCLUSION

...

ACKNOWLEDGMENTS

I would like to thank Dr. David Barbella for his detailed feedback during the development of this proposal.

REFERENCES

[1] Luís A. Alexandre, Aurélio C. Campilho, and Mohamed Kamel. 2001. On combining classifiers using sum and product rules. *Pattern Recognition Letters* 22, 12 (2001), 1283–1289.
[2] Adam Blum. 1992. *Neural networks in C++ an object-oriented framework for building connectionist systems.* John Wiley & Sons, Inc.
[3] Robert Burduk. 2012. Recognition task with feature selection and weighted majority voting based on interval-valued fuzzy sets. In *International Conference on Computational Collective Intelligence*. Springer, 204–209.
[4] Janez Demšar. 2006. Statistical comparisons of classifiers over multiple data sets. *The Journal of Machine Learning Research* 7 (2006), 1–30.
[5] Alican Dogan and Derya Birant. 2019. A weighted majority voting ensemble approach for classification. In *2019 4th International Conference on Computer Science and Engineering (UBMK).* IEEE, 1–6.
[6] Yoav Freund and Robert E. Schapire. 1997. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences* 55, 1 (1997), 119–139.
[7] Shenkai Gu and Yaochu Jin. 2017. Multi-train: A semi-supervised heterogeneous ensemble classifier. *Neurocomputing* 249 (2017), 202–211.
[8] Asma Kausar, M. Ishtiaq, M. Arfan Jaffar, and Anwar M. Mirza. 2010. Optimization of ensemble based decision using PSO. In *Proceedings of the World Congress on Engineering*, Vol. 1. IAENG, 1–6.
[9] Josef Kittler, Mohamad Hatef, Robert P.W. Duin, and Jiri Matas. 1998. On combining classifiers. *IEEE transactions on pattern analysis and machine intelligence* 20, 3 (1998), 226–239.
[10] Ludmila I. Kuncheva and Juan J. Rodríguez. 2014. A weighted voting framework for classifiers ensembles. *Knowledge and Information Systems* 38, 2 (2014), 259–275.
[11] Florin Leon, Sabina-Adriana Floria, and Costin Bădică. 2017. Evaluating the effect of voting methods on ensemble-based classification. In *2017 IEEE International Conference on INnovations in Intelligent SysTems and Applications (INISTA).* IEEE, 1–6.
[12] Yasir Mehmood, Muhammad Ishtiaq, Muhammad Tariq, and M. Arfan Jaffar. 2010. Classifier ensemble optimization for gender classification using genetic algorithm. In *2010 International Conference on Information and Emerging Technologies.* IEEE, 1–5.
[13] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12 (2011), 2825–2830.
[14] Jesper E. Van Engelen and Holger H. Hoos. 2020. A survey on semi-supervised learning. *Machine Learning* 109, 2 (2020), 373–440.
[15] Michał Woźniak, Manuel Grana, and Emilio Corchado. 2014. A survey of multiple classifier systems as hybrid systems. *Information Fusion* 16 (2014), 3–17.
[16] Zhi-Hua Zhou and Ming Li. 2005. Tri-training: Exploiting unlabeled data using three classifiers. *IEEE Transactions on knowledge and Data Engineering* 17, 11 (2005), 1529–1541.
[17] Xiaojin Jerry Zhu. 2005. Semi-supervised learning literature survey. (2005).