Ömer Faruk Tuna 2018DCS8074

Class Based Ensemble Construction

Introduction:

There are many different learning algorithms in literature. Though these are generally successful, no single algorithm is always the most accurate. In this study, we are going to discuss different models composed of multiple learners that complement each other so that by combining them, we attain higher accuracy.And we propose a new approach for constructing an ensemble via diversity.

Background Information:

The No Free Lunch Theorem states that there is no single learning algorithm that in any domain always induces the most accurate learner. The usual approach is to try many and choose the one that performs the best on a separate validation set.

Each learning algorithm dictates a certain model that comes with a set of assumptions. This inductive bias leads to error if the assumptions donot hold for the data. Learning is an ill-posed problem and with finite data, each algorithm converges to a different solution and fails under different circumstances. The performance of a learner may be fine-tuned to get the highest possible accuracy on a validation set, but this fine tuning is a complex task and still there are instances on which even the best learner is not accurate enough. The idea is that there may be another learner that is accurate on these. By suitably combining multiple base learners then, accuracy can be improved. Recently with computation and memory getting cheaper, such systems composed of multiple learners have become popular.

There are basically two questions here:

1. How do we generate base-learners that complement each other?

2. How do we combine the outputs of base-learners for maximum accuracy?

It has been understood that model combination is not a trick that always increases accuracy; model combination does always increase time and space complexity of training and testing, and unless base-learners are trained carefully and their decisions combined smartly, we will only pay for this extra complexity without any significant gain in accuracy.

Generating diverse learners:

Since there is no point in combining learners that always make similar decisions, the aim is to be able to find a set of learners who differ in their decisions so that they complement each other. At the same time, there cannot be a gain in overall success unless the learners are accurate, at least in their domain of expertise. We therefore have this double task of maximizing individual accuracies and the diversity between learners.

We can use different learning algorithms to train different base-learners. Different algorithms make different assumptions about the data and lead to different classifiers. For example, one base-learner may be parametric and another may be nonparametric. When we decide on a single algorithm, we give emphasis to a single method and ignore all others. Combining multiple learners based on multiple algorithms, we free ourselves from taking a decision and we no longer put all our eggs in one basket.

Subset Selection Methods in classifier combination:

Ensemble construction can be viewed as an optimization problem and methods proposed in the literature correspond to different search strategies in optimization: there are greedy ‘‘forward” algorithms that are incremental, and add one model at a time if the addition improves the criterion that is to be optimized. There are ‘‘backward” search methods that prune from a large set if the removal is not harmful. There are also ‘‘floating” methods that do both, as well as ones that use genetic algorithms whose operators allow both addition and deletion.

Model Combination:

There are different ways the multiple base-learners are combined to generate the final output.

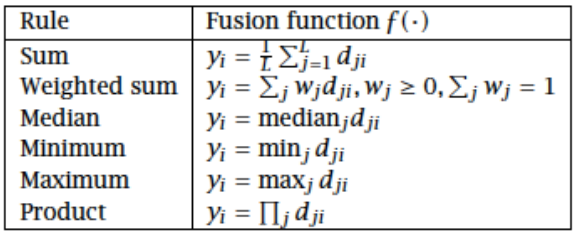
Multiexpert combination methods have base-learners that work in parallel. These methods can in turn be divided into two:

- In the global approach, given an input, all base-learners generate an output and all these outputs are used. Examples are voting and stacking.

- In the local approach, or learner selection, for example, in mixture of experts, there is a gating model, which looks at the input and chooses one (or very few) of the learners as responsible for generating the output.

Multistage combination uses a serial approach where the next base-learner is trained with or tested on only the instances where the previous base-learners are not accurate enough. The idea is that the base-learners (or the different representations they use) are sorted in increasing complexity so that a complex base-learner is not used (or its complex representation is not extracted) unless the preceding simpler base-learners are not confident. An example is cascading.

There are several rules to combine multiple learners, an example of these rules are shown in blow table:



Rule	Fusion function $f(\cdot)$
Sum	$y_i = \frac{1}{L}\sum_{j=1}^{L} d_{ji}$
Weighted sum	$y_i = \sum_j w_j d_{ji}, w_j \geq 0, \sum_j w_j = 1$
Median	$y_i = \text{median}_j d_{ji}$
Minimum	$y_i = \min_j d_{ji}$
Maximum	$y_i = \max_j d_{ji}$
Product	$y_i = \prod_j d_{ji}$

Proposed Method:

Based on the above information, my proposal can be classifed as Multiexpert Combination – global approach.

For subset selection:

Instead of searching for the candidate algorithms to be used in the ensemble, I have followed a novel strategy and pick the most suitable algorithms based on some smart logic. Using training set, I have calculated performance metrics of each base classifier for each class. The ones performing best scores for each class has been selected and used in the ensembling phase. I have used two different performance metric for classifier evaluations: f1-score and precision.

For model combination:

I have used posterior probability outputs of each selected base classifier to calculate sum, median, max, product, weighted sum and weighted median values. I have chosed the class with maximum value.

Experimental details:

Datasets: I have used 3 datasets in total. These are Optdigits, Pendigits and Pageblocks. All of them are from UCI.

Base classifiers: I have used 16 base classifiers which has been chosen to span the wide spectrum of possible machine learning algorithms as much as possible:

- knn: k-nearest neighbor with k . 1; 3; 5.
- mlp: Multilayer perceptron was with D inputs and K classes, the number of hidden units is taken as D(ml1), K(ml2),(D+K)/2(ml3), (D + K)(ml4), 2.(D + K)(ml5)
- lnp: Linear perceptron
- svm: Support vector machines (SVM) with a linear kernel (sv1), polynomial kernel of degree 2 (sv2), and a radial (Gaussian) kernel (svr).
- CART: Decision tree algorithm
- Naive Bayes algorithm
- LDA (Linear Discriminant Analysis) algorithm
- QDA (Quadratic Discriminant Analysis) algorithm

Division of training, validation, and test sets: Dataset is first divided into two parts, with 1/3 as the test set, and 2/3 as the training set, train-all, (with stratification). The training set, train-all, is then resampled using 5 fold cross-validation (with stratification). I have calculated accuracy of each algorithm using 5 fold cross validaton on training set and noted these accuracy values. In 5-fold CV, I have obtained 5 different trained models for each base learner, than picked the one that has the best score and used that model on the test data for ensembling.

Experimental Results:

Below are the accuracy values of 16 base classifiers in training set of Optdigit dataset:

K-nearest neighbor k=1	97,7683%
K-nearest neighbor k=3	97,7418%
K-nearest neighbor k=5	97,4230%
Linear Perceptron	94,4740%
MLP with 64 hidden layer D(ml1)	97,1838%
MLP with 10 hidden layer K(ml2)	96,0414%
MLP with 37 hidden layer (D+K)/2(ml3)	97,2901%
MLP with 74 hidden layer D+K(ml4)	97,6621%
MLP with 148 hidden layer 2(D+K)(ml5)	97,3433%
SVM linear kernel (sv1)	97,5824%
SVM polynomial kernel of degree 2 (sv2)	98,2731%
SVM Gaussian radial kernel (sv3)	98,4060%
Decision Tree (CART algorithm)	88,3103%
Naive Bayes	77,3114%
LDA (Linear Discriminant Analysis)	95,1116%
QDA (Quadratic Discriminant Analysis)	72,1307%

When we check the performance of each classifier for each class (based on f1 score), we saw that k-nearest neighbor algorithm (with k = 1) , sv2 and sv3 are the best performed classifiers on class level. And when we used these 3 algorithms to construct an enseble, we have observed below values:

Accuracy values for ensemble	
98,49%	median
98,65%	avg
98,49%	max
98,65%	multiply
98,49%	weighted median
98,65%	weighted mean

When we check the performance of each classifier for each class (based on precision), we saw that k-nearest neighbor algorithm (with k = 1) , MLP with 2(D+K), sv2, sv3 and naive bayes are the best performed classifiers on class level. And when we used these 5 algorithms to construct an enseble, we have observed below values:

Accuracy values for ensemble	
98,60%	median
98,44%	avg
98,60%	max
98,44%	multiply
98,60%	weighted median
98,44%	weighted mean

Below are the accuracy values of 16 base classifiers in training set of Pendigit dataset:

K-nearest neighbor k=1	99,2395%
K-nearest neighbor k=3	99,1445%
K-nearest neighbor k=5	99,0087%
Linear Perceptron	89,8832%
MLP with 64 hidden layer D(ml1)	98,5198%
MLP with 10 hidden layer K(ml2)	97,4199%
MLP with 37 hidden layer (D+K)/2(ml3)	98,5606%
MLP with 74 hidden layer D+K(ml4)	98,6013%
MLP with 148 hidden layer 2(D+K)(ml5)	98,6692%
SVM linear kernel (sv1)	98,1668%
SVM polynomial kernel of degree 2 (sv2)	98,9815%
SVM Gaussian radial kernel (sv3)	99,3074%
Decision Tree (CART algorithm)	95,2607%
Naive Bayes	84,9674%
LDA (Linear Discriminant Analysis)	87,1537%
QDA (Quadratic Discriminant Analysis)	96,6730%

When we check the performance of each classifier for each class (based on precision), we saw that k-nearest neighbor algorithm (with k = 1) , k-nearest neighbor algorithm (with k = 5), sv1, sv3 and QDA are the best performed classifiers on class level. And when we used these 5 algorithms to construct an enseble, we have observed below values:

Accuracy values for ensemble	
99,53%	median
99,56%	avg
99,53%	max
99,56%	multiply
99,53%	weighted median
99,56%	weighted mean

When we check the performance of each classifier for each class (based on f1-score), we saw that k-nearest neighbor algorithm (with k = 1) , k-nearest neighbor algorithm (with k = 5) and sv3 are the best performed classifiers on class level. And when we used these 3 algorithms to construct an enseble, we have observed below values:

Accuracy values for ensemble	
99,48%	median
99,45%	avg
99,48%	max
99,45%	multiply
99,48%	weighted median
99,45%	weighted mean

Conclusion:

We see that the resulting ensemble outperforms each of the individual base classifiers. So, using the algorithms which has highest score for each class only can be a good way to construct an ensemble.

Compared to incrementally constructing an ensemble which has polynomial $O(N^2)$ time comlexity, we can construct an ensembe with above mentioned approach with O(N) time complexity. This way, we have both increased resulting accuracy and decreased time complexity.

References:

[1] Aydın Ulas, Murat Semerci, Olcay Taner Yıldız, Ethem Alpaydın Incremental construction of classifier and discriminant ensembles, in: Information Sciences 179 (2009) 1298–1318

[2] A. Asuncion, D.J. Newman, UCI machine learning repository, 2007. <http://www.ics.uci.edu/~mlearn/MLRepository.html>.