Dynamic Ensemble Construction via Heuristic Optimization

Şenay Yaşar Sağlam and W. Nick Street
Department of Management Sciences
The University of Iowa

Abstract

Classifier ensembles, in which multiple predictive models are combined to produce predictions for new cases, generally perform better than a single classifier. Most existing methods construct static ensembles, in which one collection is used for all test cases. Recently, some researchers have proposed dynamic ensemble construction algorithms, which choose an ensemble specifically for each point from a large pool of classifiers. Ensemble performance is generally seen as having two factors: the accuracy of the individual classifiers and the diversity of the ensemble. In this study we employ heuristic optimization to examine the role of a third factor: the confidence of each classifier's prediction on the specific data point. We experiment with genetic algorithms and various hill climbing algorithms, in both single- and multi-objective scenarios, to choose locally-optimal sets of 25 classifiers from a large pool to classify each new example. We focus on dynamic ensemble construction by analyzing how diversity, accuracy and confidence interact with each other and how they affect the performance of the ensemble on new examples.

1 Introduction

An ensemble of classifiers consists of a set of trained classifiers whose individual decisions are combined to classify new examples. Ensembles have received great attention over the past decade because averaging errors of multiple predictors increases overall accuracy. Well-known algorithms for creating ensembles include Bagging [1], Boosting [3], and Random Subspace [5].

Several factors contribute to the generalization ability of an ensemble. The first is the accuracy of the base classifiers. It is clear that constructing an ensemble with the most accurate classifiers will result in good generalization error. However, since the classifiers need to make different errors in order for the ensemble to be effective, diversity among the members of an ensemble is also important. Measuring diversity is not straightforward because there is no accepted formal definition. The confidence of the prediction - that is, the closeness of one of the posterior class probability estimates to one - has also been considered. For example, Dos Santos et al. [10] used confidence measures based on the ensembles' vote margin to improve the quality of their solutions.

Most existing methods construct static ensembles, in which one collection is used for all test cases. Recently, there have been studies in which each test point is treated individually. These studies used an overproduce-and-choose strategy, in which we produce a large initial pool of candidate classifiers, and select a classifier or subset of classifiers for each new example based on some criteria. In dynamic *classifier* selection, a single predictor is chosen based on its likelihood to correctly predict the test pattern [2]. For example, Giacinto and Roli [4] proposed dynamic classifier selection approaches based on estimating local competence in selection regions defined by k-nearest neighbors (k-NN).

Dynamic ensemble selection is a relatively new area. Dynamic overproduce and choose strategy (DOCS) uses different ensembles for different test instances, using confidence as an evaluation method [10]. They defined confidence as a measure of the ensemble, based on vote margin, and used it as a second-level objective after optimizing on accuracy and diversity. Woods et al. [12] used feature subset selection to create a set of diverse classifiers and dynamically chose an ensemble for each test point based on the estimated accuracy of the predictors in the local region of the test point. Another study which used class prediction confidence as a criteria to create the dynamic ensemble is [6]. However, this method only applies to the test instances with low confidence values.

In this study, we employ heuristic optimization to examine the role of the confidence of each classifier's prediction on the specific data point. Unlike [10], we considered all test instances instead of only with low confidence value. It is important to have a fast searching method which produces high quality solutions for dynamic ensemble selection since running time will increase with the size of test set. So, we experimented with several heuristic search methods to see which of them will perform better for this purpose.

2 Dynamic Ensemble Construction

Given a set of classifiers $C = \{C_a | a = 1, ..M\}$ which have been already trained in the overproduction phase, search algorithms are used to search for the best subset of classifiers in the selection phase. When dealing with such an optimization problem, two important aspects should be analyzed: the search criterion and the search algorithm. The combined error rate, diversity measures and ensemble size are search criteria traditionally employed [9].

Rather than combining objective functions, we approach classifier selection as a Pareto or multi-objective optimization problem (MOOP), an optimization problem for which the objective function is not a singleton. It is uncommon that a single solution is is optimal for all objectives of a MOOP. Therefore, our aim is to find the tradeoffs among multiple, usually conflicting objective functions. A solution $s(1)$ is said to dominate solution $s(2)$, if $s(1)$ is no worse than $s(2)$ on all the objective functions, and $s(1)$ is better than $s(2)$ in at least one objective function. The non-dominated solutions that represent different tradeoffs between the multiple objective functions are referred to as the Pareto front.

The Ensemble construction problem can be formulates as follows: given an existing pool of component classifiers $C = \{C_a | a = 1, ..M\}$ find the best subset of classifiers to create an ensemble of classifiers E with size n. We can formulate this problem as follows:

$$\begin{aligned} \max_{E} \quad & \{Acc(E), Conf(E), Div(E)\} \\ \text{s.t.} \quad & \|E\| = n \end{aligned} \tag{1}$$

In this formulation we are attempting to simultaneously optimize three separate objectives. The accuracy $Acc(E)$ calculates the average of individual accuracies of the classifiers in the ensemble. Each C_a's performance is evaluated on a validation set. $Acc(C_a)$ is basically the percentage how many times C_a correctly classified data points in the validation set. This simple objective may sometimes work well by itself. However, theoretically it is weak because an ensemble of three identical classifiers with 85% accuracy may be worse than an ensemble of three classifiers with three 65% accuracy.

Confidence $Conf(E)$ calculates the average of individual confidences of the classifiers in the ensemble based on each test point. In our experiments, base classifiers are RBF kernel support vector machines (SVMs) [11] which are generated using LIBSVM. LIBSVM uses Platt's formula [8] to calculate the class conditional probabilities. The confidence of a classifier on a point is defined as the distance from 1.0 of the probability of the most likely class.

$Div(E)$ calculates the diversity of the ensemble on the validation data set using the method proposed in [13]. The pairwise diversity between classifier a and b is defined to be the average probabilities of classifier a misclassifying a point given that classifier b misclassifies it, and vice versa. Pairwise diversity between C_a and C_b is defined as

$$d_{ab} = 1 - \frac{P\{O_{ia} = 1 | O_{ib} = 1\} + P\{O_{ib} = 1 | O_{ia} = 1\}}{2},$$

$$\text{where } O_{ia} = \begin{cases} 0 & \text{if } a^{th} \text{ classifier is correct on data point } i, \\ 1 & \text{otherwise.} \end{cases}$$

We experimented with all combinations of the three objectives described above. The values for the objective functions were linearly scaled to the interval $[0, 1]$, by converting the minimum and the maximum values to zero and one, respectively. To find the minimum and maximum values of $Div(E)$, we used the results of the single-objective heuristic search.

3 Heuristic Search Methods

It is important to have a searching method that provides high quality solutions at a reasonable time because we try to select an ensemble for each test point. Running time of the algorithms depends on how many data points that we have in our test set. So, the choice of the search method is an important decision. There are many heuristic search methods. However, in this paper we will only consider three of them: first improvement, best improvement and multi-objective GA.

For the local search algorithms, it is important how the neighborhood is defined because the search space is too large and the aim of searching algorithms is to find high quality solutions at a reasonable time. The neighborhood is as follows: Let $\boldsymbol{S}$ be the solution space. For each solution $s \in \boldsymbol{S}$, the set $N(s) = \{j : \|s/j\| = 1\}$ is defined as the neighborhood of s. In other words, s and j are two sets which have $n - 1$ common classifiers. The size of $N(s)$ is $n(M - n)$ where M is the total number of classifiers in the pool and n is the size of the ensemble. In this study, we specified M to be 1024 and n to be 25.

In this study, we considered linear combinations of weighted objectives. Since we don't know the appropriate trade-offs between the various objectives, it is important to cover as much of the objective space as possible. Therefore, for each search method, we re-ran the algorithm 100 times with randomly-selected weights on the objectives.

The first improvement algorithm begins by randomly generating a feasible solution s. At each iteration it randomly chooses both a classifier $a \in s$ to remove and a classifier $b \notin s$ from the pool to replace, creating the new ensemble, s_{new}. It terminates when an iteration count N_{stop} has been met or no improvement is found after exhausting all possible swaps. For our experiments, N_{stop} is set to 1000.

The best improvement algorithm also begins with a random ensemble s. At each step it then iterates over all possible swaps, picking the best one if it is an improvement.

For this paper, we implemented a modified version of MOGA. This method is similar to single-objective GA. However, the selection procedure and the elitist strategy they used are different [7]. Their selection procedure takes a weighted sum of the individuals' objectives and chooses individuals for crossover based on this. The other difference is that a certain number of individuals are selected from Pareto optimal solutions and moved to the next generation as elite individuals. In the original MOGA, the weights of the various objectives were assigning randomly at each generation, changing the direction of the search as it proceeded. In order to increase coverage of the objective space and make a fair comparison to the other methods, we kept the weights constant during all generations of a single run.

4 Experimental Results

The computer used for experimentation was an Dell Studio XPS laptop with 4 GB of DDR2 RAM. The program was written in Matlab and used LIBSVM to construct classifiers. We chose the parameters such that classifiers overfit the data. In order to create a highly diverse initial pool of classifiers, they were constructed using a combination of bootstrap instance sampling (as in bagging) and random subspace selection. Five separate runs of five-fold cross validation are performed. 60% of the dataset is used to train classifiers, 20% is used for tuning and 20% for testing.

We first performed an experiment to decide which search algorithm to use. In our experiments, even though MOGA and best improvement resulted in somewhat better solutions, they were dramatically slower than first improvement.

Table 1: Comparison of Heuristic Search Methods

	MOGA			BEST IMP			FIRST IMP		
	Speed	ObjValue	TestAcc	Speed	ObjValue	TestAcc	Speed	ObjValue	TestAcc
Div	60sec	0.625	0.7392	300 sec	0.612	0.7411	13 sec	0.5457	0.7299
Acc +Conf	91sec/pt	0.8661	0.7391	400 sec/pt	0.8243	0.7459	15sec/pt	0.5835	0.7328
Acc+Div	102sec	0.9234	0.7304	480sec	0.8452	0.7558	32sec	0.7544	0.746
Conf+Div	121sec/pt	0.8975	0.7296	513sec/pt	0.6433	0.7352	27sec/pt	0.5732	0.7516
Acc+Conf+Div	137sec/pt	0.6351	0.7522	600sec/pt	0.6042	0.7574	43 sec/pt	0.621	0.7449

Due to computation time that these experiments require, we used first improvement as our searching algorithm for the comparative tests.

Because of our multi-objective optimization approach, we are left with the problem of choosing a single solution (ensemble) to classify each test point. Originally we examined only those solutions on the Pareto front (that is, non-dominated solutions), and chose based on Pareto ranking, the number of other local solutions that each one dominated. Better results were obtained by evaluating local solutions based on a tuning error to estimate generalization performance. Still, in some cases the size of the Pareto front was quite small, as low as three ensembles. In the experiments that follow, therefore, we chose the final ensemble from among *all* the local solutions, based on tuning performance. Tables 2-5 show the computational results on four datasets retrieved from the LIBSVM site: Liver Disorder, Australian, Heart Disease and Diabetes. The tables show the objective values obtained, along with the generalization errors, for each combination of objective functions.

Table 2: Liver Disorder

	f	Acc(E)	Conf(E)	Div(E)	Test Acc	Test Acc SE
All Classifiers	0.0000	0.5535	0.6568	0.4877	0.6365	0.0064
Acc	0.7442	0.7442	0.7298	0.3158	0.7154	0.0150
Conf	0.9595	0.5535	0.9595	0.4268	0.6330	0.0176
Div	0.5558	0.5668	0.6566	0.5558	0.6916	0.0122
Acc +Conf	0.4078	0.5802	0.7064	0.5136	0.6962	0.0129
Acc+Div	0.7306	0.5759	0.6602	0.5402	0.6986	0.0085
Conf+Div	0.6805	0.5641	0.6947	0.5369	0.6957	0.0134
Acc+Conf+Div	0.6133	0.5767	0.6900	0.5318	0.6968	0.0106

Table 3: Australian

	f	Acc(E)	Conf(E)	Div(E)	Test Acc	Test Acc SE
All Classifiers	0.0000	0.5526	0.6231	0.4458	0.5634	0.0187
Acc	0.8339	0.8339	0.8406	0.2985	0.8249	0.0069
Conf	0.9524	0.5526	0.9524	0.4019	0.7956	0.0085
Div	0.5764	0.5908	0.6387	0.5764	0.8307	0.0084
Acc +Conf	-0.1872	0.5792	0.6738	0.5512	0.8095	0.0109
Acc+Div	0.6397	0.5841	0.6481	0.5662	0.8237	0.0159
Conf+Div	-0.0108	0.5731	0.6681	0.5610	0.8254	0.0115
Acc+Conf+Div	0.1088	0.5797	0.6688	0.5606	0.8205	0.0132

Table 4: Heart

	f	Acc(E)	Conf(E)	Div(E)	Test Acc	Test Acc SE
All Classifiers	0.0000	0.5380	0.6029	0.5068	0.6342	0.0475
Acc	0.7787	0.7787	0.7292	0.4110	0.7613	0.0190
Conf	0.8926	0.5380	0.8926	0.4155	0.7219	0.0173
Div	0.5746	0.5795	0.6150	0.5746	0.7659	0.0161
Acc +Conf	0.4579	0.5684	0.6393	0.5554	0.7553	0.0206
Acc+Div	0.6902	0.5677	0.6161	0.5621	0.7672	0.0134
Conf+Div	0.6004	0.5571	0.6360	0.5583	0.7516	0.0181
Acc+Conf+Div	0.5643	0.5647	0.6278	0.5607	0.7449	0.0067

Table 5: Diabetes

	f	Acc(E)	Conf(E)	Div(E)	Test Acc	Test Acc SE
Acc	0.7613	0.7613	0.7735	0.3538	0.7461	0.0101
Conf	0.9706	0.5784	0.9706	0.4355	0.7352	0.0087
Div	0.5583	0.5621	0.6745	0.5607	0.7299	0.012
Acc+Conf	0.5835	0.5995	0.693	0.5354	0.7328	0.0048
Acc+Div	0.7123	0.6008	0.6954	0.5257	0.746	0.017
Conf+Div	0.5412	0.5571	0.636	0.5583	0.7516	0.0045
Acc+Conf+Div	0.645	0.5647	0.6278	0.5466	0.7449	0.0141

Contrary to our expectations, confidence appears to be a poor complement to the two traditional ensemble objectives of accuracy and diversity. Choosing classifiers based solely on their confidence gives worse performance than choosing based on either accuracy or diversity alone. Adding confidence to any other objective or set of objectives has a small but detrimental effect. Since confidence is the only objective specific to an individual test point, this unfortunately means that we find no advantage to creating dynamic ensembles in this fashion. Note however that our definition of confidence (the average confidence of the base predictors) is different from that used in previous studies (the vote margin of the ensemble). In order to understand the effects of the weights on the results we also examined the weights of the solutions (ensembles) chosen to classify the test points. The weights of the solutions appear to be approximately uniformly distributed, with a small preference for weights that are close to each other. So, curiously, our method generally chooses ensembles (based on tuning error) that were optimized giving confidence (which doesn't appear to help with generalization) an equal weight. Why the selection process chooses these weights, as opposed to reducing the importance of confidence, is under investigation.

5 Conclusions

This study examines one form of dynamic ensemble construction and the role of classifier confidence as a criterion in a multi-objective optimization setting. The primary problem to overcome in such systems is performance; re-optimizing for each test point consumes significant time and reduces the size and complexity of the data sets we can reasonably experiment on. Our results, based on a thorough examination of four small datasets, indicates that confidence (and therefore, dynamic construction) are not helpful. One challenge with these small datasets was to maintain reasonable accuracy and diversity in the initial classifier pool; this can be partly solved by experimenting on larger datasets. In future work, we will develop a heuristic search method that gives good solutions and runs faster. Then we can use more datasets with more data points.

In addition, we believe that the results can be improved if we use different ways to combine objective values instead of taking linear combination of them. As a future work, we want to create cluster of data points based on the classifiers' confidence and accuracy values. This will help us to decrease the incorrectly classified instances due to high confidence value.

References

[1] L. Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.

[2] L. Didaci, G. Giacinto, F. Roli, and G.L. Marcialis. A study on the performances of dynamic classifier selection based on local accuracy estimation. *Pattern Recognition*, 38(11):2188–2191, 2005.

[3] Y. Freund and R. Schapire. Experiments with a new boosting algorithm. In *13th International Conference on Machine Learning*, pages 148–156, 1996.

[4] G. Giacinto and F. Roli. Dynamic classifier selection based on multiple classifier behavior. *Pattern Recognition*, 34:1879–1881, 2001.

[5] T. K. Ho. The random subspace method for constructing decision forests. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(8):832–844, 1998.

[6] B. Li, K. Goh, and E. Y. Chang. Confidence-based dynamic ensemble for image annotation and semantics discovery. In *Proc. 11th ACM Int. Conf. on Multimedia*, pages 195–206, 203.

[7] T. Murata and H. Ishibuchi. Moga: Multi-objective genetic algorithms. In *IEEE International Conference on Evolutionary Computation*, volume 1, pages 289–294, 1995.

[8] J. Platt. Probabilistic outputs for support vector machines and comparison to r egularized likelihood methods. In *Advances in Large Margin Classifiers*, pages 61–74. MIT Press, 1999.

[9] E. M. Dos Santos, R. Sabourin, and P. Maupin. Single and multi-objective genetic algorithms for the selection of ensemble of classifiers. In *International Joint Conference on Neural Networks*, pages 5377–5384, Vancouver, B.C., 2006.

[10] E.M. Dos Santos, R. Sabourin, and P. Maupin. A dynamic overproduce-and-choose strategy for the selection of classifier ensembles. *Pattern Recognition*, 41:2993–3009, 2008.

[11] V. N. Vapnik. *The Nature of Statistical Learning Theory*. Springer-Verlag, New York, NY, 1995.

[12] K. Woods, W.P. Kegelmeyer, and K. Bowyer. Combination of multiple classifiers using local accuracy estimates. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(4):405–410, 1997.

[13] Y. Zhang, S. Burer, and W. N. Street. Ensemble pruning via semi-definite programming. *Journal of Machine Learning Research*, 7:1315–1338, 2006.