Cooperative Coevolutionary Ensemble Learning

Daniel Kanevskiy and Konstantin Vorontsov

Computing Center of the Russian Academy of Sciences.
Vavilov st. 40, 119991, Moscow GSP-1, Russia
kanevskiy@forecsys.ru, voron@ccas.ru

Abstract. A new optimization technique is proposed for classifiers fusion — Cooperative Coevolutionary Ensemble Learning (CCEL). It is based on a specific multipopulational evolutionary algorithm — cooperative coevolution. It can be used as a wrapper over any kind of weak algorithms, learning procedures and fusion functions, for both classification and regression tasks. Experiments on the real-world problems from the UCI repository show that CCEL has a fairly high generalization performance and generates ensembles of much smaller size than boosting, bagging and random subspace method.

1 Introduction

Combining classifiers is one of the most prominent techniques currently used to augment the accuracy of learning machines. A large number of combination schemes have been proposed in the literature [1].

Boosting is probably the most popular combination technique [2]. Base classifiers are trained in sequence so that each focuses its attention on the "hardest" examples poorly classified by the previous ones. Outputs of the base classifiers are aggregated by the weighted voting. Boosting is simple and powerful, yet it suffers from certain disadvantages. First, the greedy sequential strategy takes into account only the previous classifiers but not the next ones, thus each base classifier turns out to be suboptimal in the resulting composition. Second, outliers become "hardest" examples with a high probability, thus concentration on them may weaken base classifiers. To compensate these drawbacks boosting generates exhaustively large number of base classifiers. Generalization error of boosting may reach its minimum at thousands of base classifiers [3].

Bagging trains classifiers independently on different parts of training set, thus making them sufficiently diverse [4]. Training subsets are created by drawing objects randomly with replacements from the initial training set. Base classifiers trained on these subsets are aggregated using simple or weighted voting. Bagging is rather effective on small data sets and when base learning algorithm is instable, that is small changes in the training set lead to significantly diverse classifiers [5]. Otherwise bagging does not improve the performance of a single classifier much [6]. Also, though the bootstrapping procedure helps to maintain the diversity, no optimization is made to select training subsets. Then the resulting composition may be rather far from optimal. Like boosting, these drawbacks are compensated by taking exhaustive number of classifiers.

Random Subspace Method (RSM) trains base classifiers independently on the same training set using different random subsets of features [7]. Outputs of the obtained classifiers are aggregated by simple majority voting. RSM helps to struggle with the curse of dimensionality and is useful when the number of training objects is small compared to the number of features [8]. The disadvantage is that a little optimization is made to select features subsets carefully, which again leads to exhaustively large number of base classifiers.

Evolutionary Algorithms are frequently used in pattern recognition for feature selection and classifiers fusion. For example, [9] exploits a genetic optimization technique to choose different subsets of features for the constituents of the ensemble, and the type of each base classifier as well. The fitness function used in this approach is claimed to be advantageous, because it evaluates the performance of the combination, not of the single classifiers. But the encoding of the chromosome strongly depends on the number of classifiers in the ensemble, so it must be prespecified. This often leads to another branch of resource-intensive research aimed to choose the best ensemble size. A canonical genetic algorithm is used in [10] to independently choose a separate feature subset for each base classifier. After the subsets are chosen, one of the known fusion techniques (boosting, bagging) is applied without further ensemble optimization. This approach implicitly feeds different base classifiers with different subsets of objects and features, but no global optimization is applied to the ensemble as a whole.

Some contributions use genetic optimization to choose a subset of classifiers from a wider set of pretrained ones. They may strongly depend on the type of the learning machine such as neural networks [11], or exploit some additional information such as reliability measures [12]. Both approaches do not optimize the ensemble globally, leaving this task to the local optimization.

The latter two approaches are combined in [13], where a two-level multi-objective genetic algorithm is suggested. The first level finds the Pareto-optimal front of feature subsets, while the second chooses the best ensemble of classifiers among those trained on the Pareto-optimal subsets. The chosen classifiers are then averaged to produce the final output. This also reduces the fusion problem to a number of independent optimization tasks, and no global optimization is held to make classifiers work together. And still the number of classifiers can not be chosen automatically.

There are also a technique specific to neural networks that incorporates the power of evolutionary optimization with thoroughly selected heuristics [14]. This technique automatically determines the number of hidden neurons in NN and the number of NNs in the ensemble. But a greedy optimization technique (though with a feedback) des not allow to take advantages of classifiers cooperation.

In this paper we use a special kind of evolutionary algorithm, inspired by the symbiosis in nature, and called *cooperative coevolution* [15]. It allows to optimize all base operators and fusion function simultaneously, learns base operators to cooperate rather than to solve the problem individually, and choose the number of operators dynamically, thus obtaining an accurate small size ensemble. This technique is appropriate to any type of base classifiers and fusion functions for

both binary and multiclass classification. It can be easily propagated to regression tasks also. We embody this approach into a new ensemble learning algorithm called *Cooperative Coevolutionary Ensemble Learning* (CCEL). Section 2 introduces necessary notations. Section 3 describes the universal CCEL framework. Section 4 specifies it for the linear fusion. Section 5 presents experimental results and compares CCEL with other popular linear fusion techniques.

2 Definitions and Notation

We consider an input space X, an output space Y and a given finite dataset $D = \{x_i, y_i\}_{i=1}^{\ell}$ of pairs from $X \times Y$. Elements of X are described by n features $g_j \colon X \to V_j$, $j = 1, \ldots, n$, where V_j is a set of all permissible values of the feature g_j. The goal is to learn a function $a \colon X \to Y$ that approximates the unknown dependence of outputs from inputs. The approximation quality of a function a on a finite set $U \subseteq D$ is measured by the *empirical error*:

$$Q(a, U) = \frac{1}{|U|} \sum_{x_i \in U} L(a(x_i), y_i),$$

where $L(y, y')$ is a real-valued *loss function* that gives a deviation of the output $a(x_i)$ from the truth y_i.

For the sake of generality and following *algebraic approach to pattern recognition* [16] we introduce an intermediate space R and suppose that a function a has a form of superposition: $a(x) = C(b(x))$ for any $x \in X$, where $b \colon X \to R$ is called *base learner*, and $C \colon R \to Y$ is a fixed function. For example, in two-class classification, $Y = \{-1, +1\}$, one let $R = \mathbb{R}$ and $C(b) = \operatorname{sgn}(b)$ to deal with a real-valued classifier $b(x)$. In multiclass classification, $Y = \{1, \ldots, M\}$, the reasonable choice is $R = \mathbb{R}^M$, $C(b_1, \ldots, b_M) = \operatorname{argmax}_{y \in Y} b_y$. Regression and binary-valued classification are trivial examples with the most natural choice $R = Y$, $C(b) = b$.

An *ensemble* of base learners $b_1(x), \ldots, b_p(x)$ aggregated by a *fusion function* $F \colon R^p \to R$ is defined as superposition:

$$a(x) = C\big(F\big(b_1(x), \ldots, b_p(x)\big)\big). \tag{1}$$

Linear fusion also called *weighted voting* is a most popular example:

$$F(b_1, \ldots, b_p) = \alpha_1 b_1 + \cdots + \alpha_p b_p. \tag{2}$$

Here the usual requirement $\alpha_i \geq 0$ means that F must be a monotone function of its arguments. Less known are non-linear monotone fusion functions for both classification and regression tasks [17].

Learning algorithm as a mapping $\mu \colon (U, G) \mapsto b$ that generates base learner $b \colon X \to R$ using a finite subset of objects $U \subseteq D$ described by a finite subset of features $G \subseteq \{g_1, \ldots, g_n\}$. For example, μ may be an empirical error minimizer:

$$\mu(U, G) = \operatorname*{argmin}_{a \in A(G)} Q(a, U),$$

where $A(G)$ is a set of functions that uses only features from G.

3 Cooperative Coevolutionary Ensemble Learner

In this section we propose a generalized evolutionary algorithm for global optimization of composition (1). It uses a fixed learning algorithm μ to train base learners $b_1, \ldots, b_p$. We do not restrict neither a family of algorithms $A(G)$ nor an optimization technique that the learning algorithm μ may apply.

The evolution forms a set of $p(t)$ isolated populations $\Pi_1(t), \ldots, \Pi_{p(t)}(t)$ at each generation $t = 1, \ldots, t_{\max}$. Each population $\Pi_j(t)$ is a set of N_0 individuals. Each individual is a binary vector of the length $\ell + n$, which encodes a subset of objects $U \subseteq D$ and a subset of features $G \subseteq \{g_1, \ldots, g_n\}$. So, each individual v_j from $\Pi_j(t)$ can be considered as a pair $v_j = (U, G)$ and thus can be fed to the learning algorithm μ to obtain a base learner $b_j = \mu(v_j)$.

The evolution process starts from a single population initialized at random, see step 1 of Algorithm 1. Populations evolve independently except one but very important thing: the fitness $\varphi(v_j)$ of individual v_j is evaluated as the quality of the ensemble (1), in which j-th position is occupied by b_j, and others are the most fitted base learners b_s^*, taken by one from each population $\Pi_s(t)$, $s \neq j$:

$$\varphi(v_j) = Q\big(F(b_1^*, \ldots, b_{j-1}^*, \mu(v_j), b_{j+1}^*, \ldots, b_{p(t)}^*), D\big). \tag{3}$$

This is a main distinguishing property of the cooperative coevolution. Another ways exist to choose representatives from other populations, but the fittest ones are argued to be better if the evaluation involves only one collaboration [18].

All populations go through a common generational loop. Genetic operations (crossover and mutation) are applied to the individuals, creating offsprings, that form an intermediate populations Π_j''. The main population for the next generation consists of a number of most fitted individuals, selected from the intermediate population, and a few elite individuals, transferred from the previous main population unchanged. For each generation t the best composition $F(b_1^*, \ldots, b_{p(t)}^*)$ is selected and saved.

Populations may be added or erased during the evolution, changing the size p of the ensemble. A population is erased when its contribution into the ensemble remains too small for a number of generations. New population is created when the evolution comes into stagnation. The evolutionary process stops, when all attempts to change the size p does not cease the stagnation.

Now we give details of heuristics governing the evolutionary process. There is quite a number of ways to define them, and our choice is based on either our or other available empirical observations.

`Init`(N_0) generates N_0 random individuals. Objects are included with probability p_x and features with p_x one. We have chosen both parameters to be 0.5, but prohibited the chromosomes with less than 25% objects or features.

`Select`(Π, N) is chosen to be the deterministic truncation selection. It returns a subset of N fittest individuals from the population Π. In CCEL it is used twice: first, when N_2 elite individuals are transferred to the next generation (step 4); second, when N_0 best individuals from intermediate population are taken to form the population of the next generation (step 6).

Algorithm 1 Cooperative Coevolutionary Ensemble Learner (CCEL).

Require:
Sample $D = \{x_i, y_i\}_{i=1}^{\ell}$;
Base learning algorithm μ;
Parameters: $t_{\max}$, p_x, p_g, p_m, N_0, N_1, N_2, d_1, ε_1, d_2, ε_2, d_3, ε_3;
Ensure:
ensemble $F(b_1, \ldots, b_p)$;

initialize a single population:
  $p(1) := 1$; $\Pi_1(1) := \texttt{Init}(N_0)$;
for all generations $t := 1, \ldots, t_{\max}$ **do**
  for all populations $\Pi_j(t)$, $j := 1, \ldots, p(t)$ **do**
    create an intermediate population:
      $\Pi'_j := \texttt{Crossover}(\Pi_j(t), N_1)$;
      $\Pi''_j := \texttt{Mutation}(\Pi'_j) \cup \texttt{Select}(\Pi_j(t), N_2)$;
    fix the best individual v_j^* and corresponding b_j^*:
      $v_j^* := \arg\min_{v \in \Pi''_j} \varphi(v)$; $b_j^* := \mu(v_j^*)$; $Q_t := \varphi(v_j^*)$;
    keep top N_0 individuals:
      $\Pi_j(t+1) := \texttt{Select}(\Pi''_j, N_0)$;
    if $\texttt{Contribution}(\Pi_j)$ is small **then**
      delete population Π_j; $p(t+1) := p(t) - 1$;
  if $\texttt{Stagnation}(Q, t)$ **then**
    add population $\Pi_{p(t)+1}(t+1) := \texttt{Init}(N_0)$; $p(t+1) := p(t) + 1$;
  if $\texttt{Termination}(Q, t)$ **then**
    exit;
return ensemble $F\big(b_1^*, \ldots, b_{p(t)}^*\big)$.

$\texttt{Crossover}(\Pi, N_1)$ is the uniform crossover operator, generating N_1 new individuals (offsprings) in the following way. Two individuals (parents) are taken at random from the population Π and the offspring inherits every chromosome bit equiprobably from one of the parents.

$\texttt{Mutation}(\Pi)$ is the usual bit-flip operator that makes random changes in the bits of the individuals in Π. The canonical choice for the bit-flip probability is $p_m = \frac{1}{k}$, where k is the length of the chromosome. In our experiments we used a greater value, which seems to provide better exploration of the search space in combination with the elitist strategy [19].

$\texttt{Contribution}(\Pi_j)$ evaluates the contribution of the population Π_j to the ensemble using the *take-one-out* procedure. Two ensembles are constructed: with and without the j-th base learner, and their respective qualities Q_{jt} and $\bar{Q}_{jt}$ are estimated. The contribution is defined as their average difference for the last d_1 generations: $\frac{1}{d_1} \sum_{\tau=t-d_1+1}^{t} \big(\bar{Q}_{j\tau} - Q_{j\tau}\big)$. If the contribution is smaller than ε_1, the population is erased and the size of the ensemble decreases by one.

$\texttt{Stagnation}(Q, t)$ checks a stagnation criterion for the sequence $Q = \{Q_t\}$ at the time t. New population is created when no significant quality change happens last d_2 generations: $Q^*_{t-d_2} - Q^*_t < \varepsilon_2$, where $Q^*_t = \min\{Q_1, \ldots, Q_t\}$.

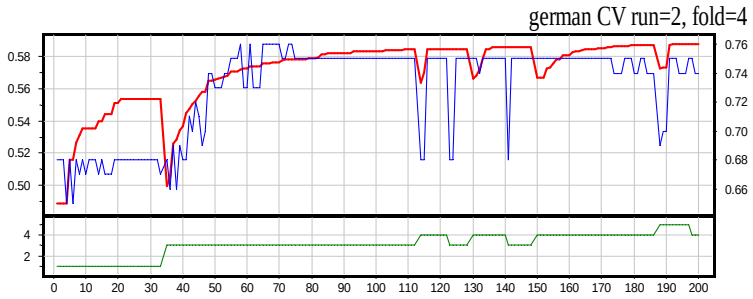

Fig. 1. Main characteristics of the evolution process as functions of generation number t. On the top chart: the thick curve is the margin functional $\widetilde{Q}$ on the training set with values along left vertical axis; the thin curve is the correct answers rate on the test set with values along right vertical axis. The bottom chart show the number of populations $p(t)$ that is equal to the size of composition.

`Termination`(Q, t) is a criterion of a lingering stagnation: $Q^*_{t-d_3} - Q^*_t < \varepsilon_3$, where $d_3 \gg d_2$ is assumed. At least several unsuccessful changes of the ensemble size p should be made before stopping the whole process.

The contribution, stagnation and termination criteria allows to control the number of populations dynamically [15]. Fig. 1 shows a typical example of how CCEL works. Here several attempts to create and erase a population was made before termination criterion has stopped a process. Note, that the structural change of a composition always leads to the significant drop of the quality on both training and test samples, but after a few generations it rehabilitates.

Now the universal CCEL framework is fully defined, except one thing: nothing was said about the optimization of the fusion function F. Although this framework allows to add another population of fusion parameters and proceed in the same manner, this approach is computationally ineffective. Instead we suggest to use a fast optimization procedure specific to the given fusion function. In the next section we demonstrate this for the linear fusion.

4 CCEL for Linear Fusion

In this section we consider the liner fusion function (2) for binary classification task assuming $Y = \{-1, +1\}$, $R = \mathbb{R}$, $C(b) = \text{sgn}(b)$.

A wide variety of weight optimization techniques is known [1]. The trivial one is the simple voting, when $\alpha_j = 1$ and no tuning is necessary. The other end of the stick is the Support Vector Machine (SVM), that allows one to obtain near-optimal weights, but seems to be too slow for evolutionary algorithms. In this work we use the naïve Bayes assumption that base classifiers are independent.

This allows to calculate weights by explicit formula:

$$\alpha_j = \ln \frac{|S| - E_j + 1}{E_j + 1},$$

where E_j is the number of errors that the classifier b_j makes on a sample $S \subseteq D$. In CCEL each base algorithm is trained on its own subsample $U \subset D$, therefore three variants of weight estimation are possible: from training $S = U$, from testing $S = D \setminus U$, or from full sample $S = D$. Our preliminary experiments showed that the second variant is the best. Additionally, if $E_j \geq \frac{1}{2}|S|$, then α_j is taken to be zero, so the algorithm b_j is excluded from the ensemble.

The quality functional $Q(a, U)$ is commonly defined as the error rate of an ensemble a on a sample U. Yet linear classifiers are known to generalize better when the direct maximization of margins is used [20]. For the composition (2) the *margin* of an object x_i is defined as

$$M(x_i) = \frac{y_i \sum_{j=1}^{p} \alpha_j b_j(x_i)}{\sum_{j=1}^{p} \alpha_j}.$$

Margin $M(x_i)$ can be thought as a distance from the object x_i to the classes boundary. If the margin is negative, $M(x_i) < 0$, then the composition makes an error, $a(x_i) \neq y_i$. Hence we define the fitness function (3) via the average margin of the training objects, and we maximize it:

$$\widetilde{Q}(a, U) = \frac{1}{|U|} \sum_{x_i \in U} M(x_i) \to \max_a .$$

A number of our preliminary experiments has shown, that $\widetilde{Q}$ really outperforms both the standard error rate functional and its combinations with the explicit diversity measures [21].

Further heuristic simplifies the `Contribution` procedure in the case of linear fusion: the contribution of Π_j can be evaluated as the average weight of the j-th base classifier for the last d_1 generations: $\frac{1}{d_1} \sum_{\tau=t-d_1+1}^{t} \alpha_j(\tau)$. When this value becomes too small, the population is erased.

Yet another linear-specific heuristic is the "1-3 rule". As a weighted voting of two classifiers makes no sense (the one with larger weight always wins), then the single-element ensemble is always increased to 3 elements.

Finally, note that the multiclass generalization of the average margin functional can easily be done, see, for example, [22].

5 Experimental Results

In this section the linear version of CCEL is compared experimentally with other linear fusion techniques: boosting, bagging, and RSM.

The base algorithm was taken to be the naïve Bayes classifier [1]. Its learning algorithm is very fast, what is very important for the resource-intensive evolutionary techniques. On the other hand, the quality of this algorithm is rather

Table 1. Bias and variance estimations through 10 runs of 10-fold cross-validation for 12 problems from the UCI repository.

Problem	Bayes		CCEL		boosting		bagging		RSM	
	bias	var	bias	var	bias	var	bias	var	bias	var
Cancer	5.24	**0.37**	**3.14**	**0.32**	**3.01**	1.74	5.32	**0.36**	4.17	0.77
Credit-a-1	14.07	1.15	**11.50**	**1.46**	14.80	1.59	14.13	1.07	13.76	2.53
Credit-a-2	15.05	**0.85**	**12.74**	1.42	13.81	4.85	14.81	1.05	14.77	1.56
Credit-g	28.23	**3.84**	**21.04**	4.74	24.46	5.02	27.57	**3.73**	27.61	4.54
DBC	11.04	**0.47**	**4.64**	0.74	9.29	15.71	10.82	0.57	10.64	0.64
Heart	16.78	**2.64**	15.44	**2.71**	**13.73**	10.38	16.45	**2.87**	16.60	**2.83**
Hepatitis	15.11	**1.82**	**14.12**	4.27	15.25	4.65	**14.40**	2.40	15.19	2.26
Liver	28.51	**7.90**	**23.77**	11.06	24.50	10.34	28.25	8.31	28.40	8.18
Diabetes	29.27	2.49	21.60	**2.06**	**18.40**	12.99	28.76	2.69	28.44	3.13
Survival	23.69	7.60	23.01	**4.67**	**21.03**	6.09	23.87	6.34	23.51	8.99
Tic-tac-toe	24.61	4.30	**18.82**	5.80	32.56	**0.73**	25.01	4.02	24.51	4.73
Voting	5.82	0.79	**4.03**	**0.57**	4.94	1.31	5.81	0.73	6.09	0.92

moderate, because the underlying assumption of features independence does not hold for most real-world problems. Note also that naïve Bayes classifier has a very low variance, which prevents standard fusion techniques from improving its quality. Indeed, it is rarely used as ensemble building block. The low variance leaves the only way of significant performance improvement: reducing a bias. Both bagging and RSM are known to fail in reducing bias, while boosting sometimes effectively trades off bias for variance and vice versa. To estimate how CCEL works in these terms we made empirical evaluation of bias and variance using a standard technique from [23]. Results are summarized in Table 1.

We compared CCEL with the base classifier, AdaBoost, bagging, and RSM. For the latter three methods the ensemble size was fixed to be $p = 250$ [4]. The experiments were made on 12 two-class problems from the UCI repository [24]. Table 2 summarizes the average test error.

The results verify the assumption that bagging and RSM fail to improve the quality of the low-variance base classifier. CCEL competes with boosting in bias reduction, while preserving substantially lower level of variance. As a result, CCEL outperforms others in 11 problems from 12, see Table 2.

6 Conclusions and Open Problems

Cooperative coevolution is a very natural approach to ensemble learning. It trains base learners to cooperate with each other rather than to solve the problem independently. Each base learner tends to specialize on its own subset of objects and subspace of features. This results in significant reduction of the ensemble size compared to standard techniques. CCEL takes 3–6 base learners whereas boosting, bagging and RSM require hundreds. CCEL is applicable to base learning algorithms and fusion functions of any type, though it is also open to any

Table 2. The test error rates with standard deviations (in percents) averaged through 10 runs of 10-fold cross-validation for 12 problems from the UCI repository. For CCEL the average size of the ensemble is written in parentheses; size deviations typically do not exceed ±1.

Problem	Bayes	CCEL	boosting	bagging	RSM
Cancer	5.55 ± 0.26	$\mathbf{3.46 \pm 0.37}$ (3.16)	4.14 ± 1.48	5.63 ± 0.24	4.97 ± 0.40
Credit-a-1	15.22 ± 0.39	$\mathbf{12.96 \pm 0.57}$ (2.48)	25.23 ± 6.65	15.20 ± 0.42	15.31 ± 0.62
Credit-a-2	15.94 ± 0.40	$\mathbf{14.16 \pm 0.53}$ (2.99)	17.72 ± 2.86	15.87 ± 0.45	16.12 ± 0.48
Credit-g	32.07 ± 0.67	$\mathbf{25.78 \pm 0.65}$ (1.74)	29.48 ± 0.93	31.30 ± 0.67	32.11 ± 0.71
DBC	11.50 ± 0.30	$\mathbf{5.38 \pm 0.44}$ (2.66)	25.00 ± 7.22	11.40 ± 0.27	11.24 ± 0.35
Heart	19.42 ± 0.92	$\mathbf{18.15 \pm 0.85}$ (3.32)	24.11 ± 6.47	19.31 ± 1.16	19.35 ± 1.19
Hepatitis	$\mathbf{16.93 \pm 1.06}$	18.38 ± 1.43 (2.87)	19.90 ± 1.80	$\mathbf{16.80 \pm 1.14}$	17.32 ± 1.46
Liver	36.42 ± 1.86	$\mathbf{34.38 \pm 0.95}$ (1.95)	$\mathbf{34.84 \pm 2.45}$	36.56 ± 1.88	36.48 ± 1.77
Diabetes	31.76 ± 0.63	$\mathbf{23.66 \pm 0.43}$ (2.30)	31.39 ± 2.05	31.45 ± 0.68	31.53 ± 0.58
Survival	31.29 ± 2.58	$\mathbf{26.21 \pm 1.02}$ (2.02)	27.12 ± 1.56	30.22 ± 2.77	32.41 ± 3.24
Tic-tac-toe	28.92 ± 1.02	$\mathbf{24.61 \pm 1.11}$ (2.59)	33.29 ± 0.40	29.03 ± 0.96	29.20 ± 1.07
Voting	6.61 ± 0.60	$\mathbf{4.60 \pm 0.46}$ (3.53)	6.25 ± 0.65	6.54 ± 0.40	7.0 ± 0.75

type-specific heuristics. CCEL yields good results even for such stable and inaccurate algorithm like naïve Bayes. Finally, analysis of the CCEL results can tell much about the structure of the problem. For example, one can determine most important features and filter out some useless objects (outliers).

The only disadvantage of CCEL is the training speed: the solution of a middle-size problem takes a few minutes on usual PC. On the other hand, CCEL is very suitable as "anytime" learning algorithm that may be interrupted at any moment to return a solution, and then continued to learn more [25, 26].

Further development of CCEL seems very promising in many directions: increasing the algorithm's speed, application to various base classifiers and fusion functions, finding optimal combinations of heuristics.

References

1. Kuncheva, L.: Combining pattern classifiers. John Wiley & Sons, Inc. (2004)
2. Schapire, R.: The boosting approach to machine learning: An overview. In: MSRI Workshop on Nonlinear Estimation and Classification, Berkeley, CA. (2001)
3. Freund, Y., Schapire, R.E.: Experiments with a new boosting algorithm. In: International Conference on Machine Learning. (1996) 148–156
4. Breiman, L.: Arcing classifiers. The Annals of Statistics **26**(3) (1998) 801–849
5. Dietterich, T.G.: An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. Machine Learning **40**(2) (2000) 139–157
6. Bauer, E., Kohavi, R.: An empirical comparison of voting classification algorithms: Bagging, boosting, and variants. Machine Learning **36**(1-2) (1999) 105–139

7. Ho, T.K.: The random subspace method for constructing decision forests. IEEE Transactions on Pattern Analysis and Machine Intelligence **20**(8) (1998) 832–844
8. Skurichina, M., Duin, R.P.W.: Limited bagging, boosting and the random subspace method for linear classifiers. Pattern Analysis & Applications (5) (2002) 121–135
9. Kuncheva, L.I., Jain, L.C.: Designing classifier fusion systems by genetic algorithms. IEEE-EC **4**(4) (2000) 327
10. Guerra-Salcedo, C., Whitley, D.: Genetic approach to feature selection for ensemble creation. In: Proceedings of the Genetic and Evolutionary Computation Conference. Volume 1., Morgan Kaufmann (1999) 236–243
11. Zhou, Z.H., Wu, J.X., Jiang, Y., Chen, S.F.: Genetic algorithm based selective neural network ensemble (2001)
12. Stefano, C.D., Cioppa, A.D., Marcelli, A.: Exploiting reliability for dynamic selection of classifiers by means of genetic algorithms. In: Int. Conf. On Document Analysis and Recognition ICDAR03, Edinburgh (UK), August 3-6. (2003) 671–675
13. Oliveira, L., Sabourin, R., Bortolozzi, F., Suen, C.: Feature selection for ensembles: A hierarchical multi-objective genetic algorithm approach. In: International Conference on Document Analysis and Recognition, Edinburgh-Scotland, IEEE Computer Society (2003)
14. Islam, M., Yao, X., Murase, K.: A constructive algorithm for training cooperative neural network ensembles. IEEE Transactions on Neural Networks **14**(4) (2003) 820–834
15. Potter, M.A., De Jong, K.A.: Cooperative coevolution: An architecture for evolving coadapted subcomponents. Evolutionary Computation **8**(1) (2000) 1–29
16. Zhuravlev, J.I.: An algebraic approach to recognition or classifications problems. Pattern Recognition and Image Analysis **8**(1) (1998) 59–100
17. Rudakov, K.V., Vorontsov, K.V.: Methods of optimization and monotone correction in the algebraic approach to the recognition problem. Doklady Mathematics **60**(1) (1999) 139
18. Wiegand, R.P., Liles, W.C., De Jong, K.A.: An empirical analysis of collaboration methods in cooperative coevolutionary algorithms. In Proceedings of the Genetic and Evolutionary Computation Conference (GECCO) (2001)
19. Andre's Pena-Reyes, C.: Coevolutionary Fuzzy Modeling. Springer-Verlag New York (2004)
20. Mason, L., Bartlett, P., Baxter, J.: Direct optimization of margins improves generalization in combined classifiers. Technical report, Deparment of Systems Engineering, Australian National University (1998)
21. Kuncheva, L., Whitaker, C.: Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy. Machine Learning **51**(2) (2003) 181–207
22. Allwein, E.L., Schapire, R.E., Singer, Y.: Reducing multiclass to binary: A unifying approach for margin classifiers. In: Proc. 17th International Conf. on Machine Learning, Morgan Kaufmann, San Francisco, CA (2000) 9–16
23. Webb, G.I.: MultiBoosting: A technique for combining Boosting and Wagging. Machine Learning **40**(2) (2000) 159–196
24. Blake, C., Merz, C.: UCI repository of machine learning databases. Technical report, Department of Information and Computer Science, University of California, Irvine, CA (1998)
25. Russel, S.J., Zilberstein, S.: Composing real-time systems. In: IJCAI. (1991) 212–217
26. Esmeir, S., Markovitch, S.: Lookahead-based algorithms for anytime induction of decision trees. In: Proceedings of the 21st International Conference on Machine Learning (ICML-2004). (2004)