A Probabilistically Optimal Ensemble Technique for Training Based Classifiers

Kosuke Imamura, Kris Smith
Department of Computer Science
Eastern Washington University
Cheney, WA 99004-2412
E-mail: {kimamura,ksmith}@ewu.edu

***Abstract*— Diversity of individual classifiers is essential for successful ensemble learning. However, without a quantifiable definition of diversity, it is unknowable whether or not the desired diversity is obtained. This paper introduces a probabilistically optimal ensemble technique for training based classifiers. The existing ensemble techniques focus on diversity promotion, hoping that a to-be-formed ensemble will be adequately diversified by such techniques. However, little attention has been paid to measuring the diversity of a formed ensemble. A probabilistically optimal ensemble technique, based on fault masking among individual classifiers, provides the metric for diversity.**

I. Introduction

The purpose of ensemble learning is to enhance accuracy and to reduce performance fluctuation through voting by the ensemble members. Diversity among the voters is essential for successful ensembles, since there is little gained by combining individuals of a similar voting behavior. Without a quantifiable diversity definition, whether or not desired diversity is obtained is unknowable. This paper provides a metric that probabilistically quantifies diversity of an ensemble, which in turn enables us to form an optimal ensemble.

The existing ensemble methods focus on diversity promotion techniques, hoping that a to-be-formed ensemble will be adequately diversified by such techniques. An isolated island model in evolutionary computation is an example of diversity promotion. Individual classifiers are trained in isolation and combined after training is completed. However, little attention has been paid to measuring the diversity of a formed ensemble. Without a proper diversity metric, we will not recognize insufficient diversity of a formed ensemble when the applied diversity pressure (such as an isolated island model and different initial weights in neural networks) did not function as intended.

A probabilistically optimal ensemble is an adaptation of the principles of fault tolerant computing. For example, a three version system can tolerate one fault by a simple majority rule (fault masking). To quantify the diversity, we use the expected error rate of combined individuals when the individual faults are assumed to be independent. If the error rate of an ensemble classifier is not in the neighborhood of the expected error rate, we conclude some of the voters are not casting independent votes. In other words, given classifiers and the individual error rates, we can compute the error rate of a probabilistically optimal fault masking ensemble. This metric is beneficial because it allows us to detect and reject ensembles of insufficient diversity. Take bagging for instance. Bagging [1] is a technique to promote diversity of ensemble members. However, whether or not desired diversity is obtained is unchecked. The proposed metric can be used as an acceptance test of an ensemble formed by bagging.

Initially, this probabilistic metric was developed to enhance accuracy and to reduce performance fluctuation of programs produced by genetic programming [2]. Genetic programming is a randomized and training based learning algorithm. Inevitably, a trained individual produces faulty outputs due to the stochastic nature of the algorithm and the quality of the training data. Also, performance of equally fit individuals over the identical training set may widely fluctuate on an unseen data set. Hence, N-version Genetic Programming (NVGP) was developed. However, the generality of NVGP technique does not limit its application only to genetic programming. Any classifier combination method will benefit from this probabilistic diversity metric.

The paper is organized as follows: Section II provides N-version programming background. Section III reviews current ensemble approaches. Section IV reviews diversity issues and introduces the probabilistic quantification of diversity and the optimal ensemble. Section V presents the experimental results on a DNA sequence classification problem. Section VI discusses the implications of our study and future research.

II. Background of N-version programming (NVP)

A fault is an undesirable behavior within a system, such as incorrect output from a program module or a dropped bit in a communication line. However, the system may behave correctly. A failure occurs when a system behaves incorrectly. For example, returning an incorrect value to the user or forwarding an erroneous packet is a failure. Fault-tolerant systems detect and process faults before they become failures. NVP is defined as the independent generation of $N \geq 2$ functionally equivalent programs (versions) from the same initial specifications [3]. A fundamental assumption of the NVP is that an independent programming effort will reduce the probability that similar errors will occur in two or more versions [3].

But this assumption was questioned. Knight and Leveson applied a probabilistic metric to measure the assumed independence of modules in NVP [4], and rejected the hypothesis of the assumed independence of faults by independently developed programs. However, this conclusion does not invalidate NVP in general. Hatton determined that multiple versions developed for NVP are sometimes more reliable and cost effective than a single good version [5], even with non-independent faults. His 3-version system increased the reliability of the composite NVP system by a factor of 45. This is far less than the theoretical improvement of a factor of 833.6. Nonetheless, it is still a significant improvement in system reliability.

III. Ensemble techniques

A. Ensemble technique development

Hashem has published a series of research papers on linearly optimal combination of artificial neural networks (ANN). Among them are [6], [7]. His research showed that the performance of an optimal linear combination of neural networks was superior to the individual combination constituent modules. In his approach, an optimal weight vector for the redundant modules (separately trained ANNs to be combined) is computed from the sample space. In 1995, Krogh and Vedelsby also proposed a weighted average technique. However, the concept of diversity emerged in their study [8]. They expressed it as "the disagreement among the networks on input x". The ensemble generalization error was decomposed into two terms: the errors of the individual networks and all correlations between the networks. The importance of disagreement without increasing the individuals' errors is recognized. To promote disagreement, they suggested subset training and a mixture of neural networks of different topologies.

In 1996, Rosen proposed decorrelated neural networks, which perform error cancellation by averaging [9]. It is a similar work to the study conducted by Krogh and Vedelsby. However, he made an important remark on the penalty imposed on correlations. If the penalty is too high, then even though the individual networks may be highly decorrelated, the ensemble network performance may be poor. This situation occurs when covariance is reduced at the expense of an increase in individual network errors.

In 1997, Zhang and Joung proposed Mixing Genetic Programs (MGP). MGP chooses a pool of individuals from a population and the master unit assigns the voting weights to these individuals using an additive weighting scheme [10].

In 1998, Jimenez and Walsh proposed a dynamically weighted ensemble which assigns weights to each output according to the individual network's confidence [11]. If an individual is highly confident with its output, the output will have a high weight. Likewise, if an individual is not confident, the output will have a low weight. This work is related to our future research on decision abstaining NVGP [12].

In 1998, Feldt adapted NVP to GP ensembles [13]. In his study, 120 ensembles are formed by exhaustively combining 3 of the 10 top performing individuals out of 80 individuals. Pairwise diversity of program behaviors is computed by averaging correlation coefficients of every pair of individuals. However, the low correlation often did not lead to a low failure rate, and the paper left this for a future study. The problem is that the pairwise measure does not define the optimal ensemble. Therefore, the ensemble that obtains the lowest correlation may be sub-optimal. We have a simple explanation as to why pairwise correlation as a measure of the ensemble diversity may not lead to effective fault masking. Suppose we have 3 programs, P1, P2, and P3. Assume that (P1, P2) are correlated but (P1,P3) and (P2,P3) are not. Even if this system had a low average correlation coefficient, the system would be essentially a 2-version system, which is capable of detecting errors but not capable of masking errors. A diversity metric that is based on pairwise dissimilarity has also been proposed by Ekárt and Németh [14]. We need to reexamine the validity of the assumption that pairwise dissimilarity is equivalent to the system-wise diversity. In 1999, Iba applied boosting and bagging to genetic programming. His experiment validated the effectiveness of these techniques and showed the potential for controlling bloat (the tendency of GP to produce larger programs over time) [15]. In the same year, Soule applied voting to GP [16].

In 2001, Land used an adaptive boosting (AdaBoost) technique to improve performance neural network architectures that employ evolutionary techniques in a breast cancer diagnostic application [17]. The AdaBoost algorithm trains a weak learner (slightly better than random guessing) by iterating training while increasing the weights of mis-classified samples and decreasing the weights of correctly classified ones [18]. The effect is that the weak learner focuses more and more on the previously mis-classified samples. Therefore, noise and outliers receive high weights at the later stages and the AdaBoost becomes susceptible to them. Rätsch et. al. adapted weight decay in order to curve the susceptibility [19].

Also, in 2001, Langdon proposed the use of GP to combine given classifiers. [20]. Instead of optimizing linear combination, GP was used to discover optimal combination. Imamura used simple averaging to track a moving object by multiple individuals generated by GP [21]. The experimental distributed GP ensemble system significantly improved the mean-time-between-failures compared with a single best version.

In 2002, Imamura proposed NVGP [22]. Unlike Feldt's N-version ensemble approach, the NVGP optimal ensemble do not use pairwise correlation as a measure of the ensemble diversity. Instead, NVGP uses the expected failure rate of an ensemble when the individual faults are assumed to be independent. NVGP's system-wise diversity metric solves the problem associated with the pairwise correlation method. NVGP is different from weight optimization for given individuals or subset training methods. NVGP defines the optimality of an ensemble based on the fundamental assumption of N-version programming, that is, independent programming efforts would reduce the probability that similar errors will occur in multiple versions [23]. NVGP uses this definition to validate the ensembles.

In the same year, Imamura proposed accuracy enhancement by decision abstention. Decision Abstaining N-Version Genetic Programming is NVGP that abstains from decision-making, when there is no decisive vote among the modules to make a decision. A special course of action may be taken for an abstained instance.

B. Boosting and Bagging

Boosting and bagging are methods that perturb the training data by resampling to induce classifier diversity. The AdaBoost algorithm trains a weak learner (slightly better than random guessing) by iterating training while increasing the weights of misclassified samples and decreasing the weights of correctly classified ones [18]. The effect is that the weak learner focuses more and more on the misclassified samples. The trained classifiers in each successive round are weighted according to their performance and cast a weighted majority vote. Bagging (Bootstrap aggregating) replicates training subsets by sampling with replacement [1]. It then trains classifiers separately on these subsets and builds an ensemble by aggregating these individual classifiers. However, both techniques have limitations. Boosting is susceptible to noise, Bagging is not any better than a simple ensemble in some cases, neither Boosting nor Bagging is appropriate for data poor cases, and bootstrap methods can have a large bias [18], [24], [25], [26], [27], [28].

IV. Definition of probabilistically optimal ensemble

A. Conditions for probabilistically optimal ensemble

Apparently, if multiple individuals acquire the same knowledge, there will be no benefit of combining them since fault masking will not occur. We must consider the following two conditions in order for fault masking to take place:

1. If individuals produce faulty outputs, then such outputs should be statistically independent.
2. Individuals must have a reasonably high fitness.

The first condition is to quantify the degree of individual learning. The second condition is a restriction on the first condition so that we do not have a situation where combination of low fit individuals becomes a high fit system. In such a system, the low fit individuals can be viewed as terms of an expression rather than vote participating individuals because the outputs of low fit individuals do not contribute to fault masking. Consider the following case for example:

1. Unseen target function is: $y = x$.
2. Evolved individuals are: $f1 = 999 * x + 1000$ and $f2 = 10 * x + 10$.
3. Linear combination is: $y = 100 * f2 - f1$.

Neither $f1$ nor $f2$ will approximate the target function in proximity. Although a linear combination yields perfect solutions, the correctness is not due to fault masking. It is more appropriate to view this ensemble system as a single program. Thus, different behaviors alone do not necessarily lead to a needed diversity for fault masking [2].

B. Definition of probabilistically optimal ensemble

Let n be the size of an ensemble, p be the probability that each of n individuals produces a faulty output, and m is the minimum number of faulty outputs for an ensemble to fail. (We assume the same fault p rate for individuals for simplicity.) Then, the expected failure rate f of this ensemble (initially derived for n-modular redundant hardware systems [29]) is

$$f = \sum_{k=m}^{n} \binom{n}{k} (1-p)^{n-k} p^k \tag{1}$$

For an N-version classifier system, such as ours, the i_{th} individual fault rate p_i is the ratio of misclassified examples to the total number of training instances. In this case, f is the expected error rate of an ideal ensemble. If the fault rate is the same for every p_i, f is an area under a binomial probability density function as shown in the above formula. The error rate of an ensemble is close to the statistically expected error rate f precisely when component failures are not dependent. For example, suppose we have three individuals of error rate 0.2. Then, the expected ensemble failure rate should be $0.2^3 + 3.2^2.8 = 0.104$. If the ensemble does not attain this error rate, the desired fault masking is not happening. Therefore, the ensemble is not optimal. Explicit quantification of the module diversity relative to the expected failure rate measures the effectiveness of a *formed* ensemble.

C. A view on weighted linear combinations

As far as voting is concerned, weighted linear combination can be viewed as an ensemble some individuals of which have the same voting behavior. For example, if the combination weight vector is $0.5, 0.25, 0.25$ for classifiers c_1, c_2, c_3. Then, it is the same as an ensemble which consists of c_1, c_1, c_2, c_3 with the equal weights. Although probabilistically optimal ensemble can be considered as a special case of linear combination, we state the following to clarify the fundamental difference:

> *Linear combination is a search problem for the best weight vector for given classifiers, while the probabilistically optimal ensemble is a search for classifiers of independent voting behavior.*

V. Experiment: Escherichia coli (E. coli) promoter recognition

This section briefly summarizes our experiment. For detail experimental setups and computational methods, see [2]. The problem is to classify whether a given DNA sequence is an E. coli promoter. A DNA sequence can be described as a character string of `(a|c|g|t)`$^+$. A promoter is a DNA sequence that regulates when and where an associated gene will be expressed. We used 2-gram encoding for input. The 2-gram encoding counts the occurrences of two consecutive input characters (nucleotides) in a sliding window. The data set is taken from USC ML repository [30]. It contains 53 E. coli

TABLE I

PROBABILISTICALLY OPTIMAL ENSEMBLES OUTPERFORM NON-OPTIMALS

	3voter	9voter	10voter	11voter	13voter	15voter	20voter	30voter	31voter	Avrg
Optimal wins %	86	72	68	74	71	70	78	100	NA	77

DNA promoter sequences and 53 non-promoter sequences, all of length 68. Our objective was to quantify the effectiveness of a fault tolerant system built with our ensemble construction method, not to produce a competitive promoter detection tool (although it is promising). This problem has been investigated with neural networks and genetic programming [31], [32], [33], [34].

We compared the performance distributions of a group of single best versions and a group of NVGP ensembles, since evaluation and comparison of one or small number of evolved individuals or ensembles would have been susceptible to stochastic errors in performance estimation. We assume the number of errors have a normal distribution, since each test instances can be viewed as a Bernoulli trial [28].

Our classifiers are built by linear genetic programming [35]. The length of an individual program is restricted to a maximum of 80 instructions. Each program used 16 read-only registers for input data, which contained counts for individual nucleotide 2-grams, and 4 read-write working registers. Our linear genome machine mimics MIPS instruction architecture. Forty individual classifiers are evolved in isolation to promote diversity among them.

The classifier clusters the positive instances and places the negative instances outside the cluster. The cluster is defined by the interval $(\mu - 3\sigma, \mu + 3\sigma)$, where μ is the mean of the classifier output values for the positive instances and σ is the standard deviation. If an output value from a given sequence falls within this interval, then it is classified as a promoter. Otherwise, it is a non-promoter. In order to generate sufficiently large statistical samples for the experience, we used the Beowulf cluster supercomputing facilities from the Initiative for Bioinformatics and Evolutionary STudies (IBEST), University of Idaho.

We collected optimal and non-optimal ensembles by the following procedures for performance statistics. The error rate is calculated by the total number of incorrect classification divided by the total number of training instances (70 instances).

```
1. Create 40 isolated islands with 100
   individuals each.
2. Evolution ends when predefined fitness
   is achieved.
3. Pool those with the predefined fitness
   from each island.
4. Randomly select N individuals from
   this pool to create an ensemble E.
5. Evaluate the performance of E.
   If E is optimal (error rate 0.014),
   place E into the optimal ensemble set.
   Else place E into the non-optimal set.
6. Goto 4.
```

Table I shows that the probabilistically optimal ensembles outperform non-optimal ensembles approximately 70-80% of the time. This particular table excludes the ensembles of error rate 2/70 to 3/70, since they are close to the optimal ensembles. It is also shown that when the optimals outperform non-optimals they do so with a greater margin than when the non-optimals outperform the optimals. See [2] for details of the experimental result.

Fig. 1 shows the error distribution of classifiers at 90% interval. The leftmost bar indicates the error distribution of single best versions. The middle and leftmost are 15-voter and 31 voter ensembles' error distributions respectively. Notice the performance fluctuations of single best versions. It indicates that a well trained individual has a chance to become practically a random classifier (error rate > 0.4) roughly 10-20% of the time on unseen data. Unfortunately, we have no way of knowing which one would become a random classifier beforehand, since they are all equally well trained. However, none of the optimal ensembles is a random classifier. This is why training based algorithms should form an ensemble.

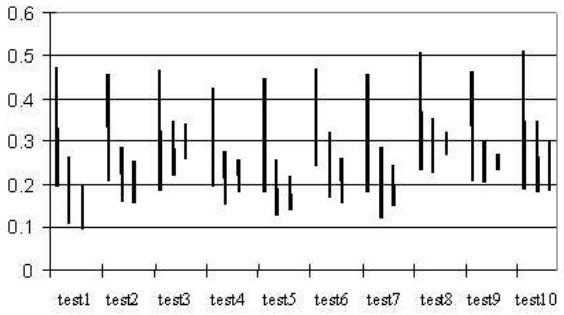

Fig. 1. Error distribution at 90% interval. Leftmost, middle, and rightmost bars are distribution of single-best version, 15-voter, and 30-voter respectively

VI. DISCUSSION AND FUTURE RESEARCH

Freund, et al., showed the error bound of averaging classifier with abstention [36]. The optimal ensemble's error rates are shown to be far below this theoretical bound even without decision abstention [12].

The performance fluctuation of an ensemble decreases, as the ensemble size becomes closer to the pool size. This is because more components overlap among the ensembles. Obviously, if we combine all of the individuals in the candidate pool (though not necessarily optimal), there is no performance fluctuation to be observed. However, we claim that a larger ensemble size is better for accuracy and small performance fluctuations as long as it is optimal. Let $\mathcal{D}$ is the set of

all distinct voting patterns by high fit individuals over a training set. If a training set is finite, then $\mathcal{D}$ is finite. Assume that we obtained a sufficiently large ensemble candidate pool of high fit individuals of the same error rate. We further assume that the fault masking technique is only as good as a simple averaging scheme. Then, by the central limit theorem, given a distribution (μ, σ^2), the sampling distribution of the mean approaches a normal distribution $(\mu', \sigma^2/N)$ as N, the sample size, increases. But, the means remain the same, $\mu = \mu'$. This is exactly what averaging can achieve. NVGP optimal ensembles form a subset of $\mathcal{D}$ with lower error rates because of ideal fault masking. The lower error rate is also shown experimentally in Fig. 1. Thus, $\mu > \mu'$. Since an optimal ensemble is combination of individuals whose faults are independent, we conclude from Equation 1 that the larger the ensemble size the lower the expected error rate, provided that the ensemble is optimal. Thus, the probabilistic optimal ensemble method offers asymptotic improvement of accuracy as the ensemble size increases. This is an important feature because it does not introduce a configuration problem such as the number of training subsets and its size in bagging or the number of rounds in boosting. However, it does need to search for individuals of independent faults. This search problem was shown to be easy for an isolated model genetic programming [2].

Is there a case where probabilistic optimal ensemble method fail? Potentially yes. For instance, if there are noisy data in the training set, individuals may produce dependent faults on these instances. Therefore, the ensembles deviate from the expected error rate. Failure to produce the optimal ensemble gives us an opportunity to examine the quality of training data.

Ensemble is particularly useful when learning is difficult. For instance, when individual classifiers will not improve beyond 70% accuracy, a five voter system achieves 84% accuracy if it is probabilistically optimal.

Currently, we are developing a training based classifier system for network intrusion detection. The intrusion detection system monitors the kernel level activities and a classifier system is used to predict whether the system is under attack. The kernel level activities include kernel routine call-frequency and call-sequence. We plan to apply this probabilistic diversity metric as an acceptance test of the combined classifiers.

ACKNOWLEDGMENT

One of the authors (Imamura) is partially supported by Congressional Appropriation in support of Eastern Washington University's Cybersecurity Initiative.

REFERENCES

[1] L. Breiman, "Bagging predictor," Department of Statistics, University of California Berkley, http://www.salford-systems.com/docs/BAGGING_PREDICTORS.PDF, Technical Report 421, 1994.

[2] K. Imamura, T. Soule, R. Heckendorn, and J. Foster, "Behavioral diversity and a probabilistically optimal gp ensemble," *Genetic Programming and Evolvable Machines*, vol. 4, no. 3, pp. 235–253, 2003.

[3] A. Avizienis and J. Kelly, "Fault tolerance by design diversity: Concepts and experiments," *IEEE Computer*, vol. 17, no. 8, pp. 67–80, 1984.

[4] J. Knight and N. Leveson, "An experimental evaluation of the assumption of independence in multiversion programming," *IEEE Transaction on Software Engineering*, vol. SE-12, no. 1, 1986.

[5] L. Hatton, "N-version vs. one good program," *IEEE Software*, vol. 14, no. 6, pp. 71–76, 1997.

[6] S. Hashem, "Optimal linear combinations of neural networks," *Neural Networks*, vol. 10, no. 4, pp. 599–614, 1997.

[7] ——, "Improving model accuracy using optimal linear combinations of trained neural networks," *IEEE Transactions on Neural Networks*, vol. 6, no. 3, pp. 792–794, 1995.

[8] A. Krogh and J. Vedelsby, "Neural network ensembles, cross validation, and active learning," in *Advances in Neural Information Processing Systems*, G. Tesauro, D. Touretzky, and T. Leen, Eds., vol. 7. citeseer.nj.nec.com/krogh95neural.html: The MIT Press, 1995, pp. 231–238.

[9] B. Rosen, "Ensemble learning using decorrelated neural networks," *Connection Science*, vol. 8, pp. 373–384, 1996. [Online]. Available: citeseer.nj.nec.com/rosen96ensemble.html

[10] B. T. Zang and J. J. G., "Enhancing robustness of genetic programming at the species level," in *Proceeding of the 2nd Annual Conference Genetic Programming 97*. Morgan Kaufmann, 1997, pp. 336–342.

[11] D. Jimenez and N. Walsh, "Dynamically weighted ensemble neural networks for classifi cation," in *The 1998 International Joint Conference on Neural Networks*, citeseer.nj.nec.com/jimenez98dynamically.html, 1998.

[12] K. Imamura, R. Heckendorn, T. Soule, and J. Foster, "Abstention reduces errors - decision abstaining n-version genetic programming," in *Proceedings of Genetic and Evolvalble Computing Conference (GECCO)*, 2002, pp. 796–803.

[13] R. Feldt, "Generating diverse software versions with genetic programming: an experimental study," *IEE Proceedings - Software, Special issue on Dependable Computing Systems*, vol. 145, no. 6, pp. 228–236, 1998.

[14] A. Ek´art and S. Z. N´emeth, "Maintaining the diversity of genetic programs," in *Proceedings of 5th European Conference EuroGP2002*. Springer, 2002, pp. 162–171.

[15] H. Iba, "Bagging, boosting, and bloating in genetic programming," in *Proceedings of the Genetic and Evolutionary Computation Conference*, vol. 2. Morgan Kaufmann, 1999, pp. 1053–1060.

[16] T. Soule, "Voting teams: A cooperative approach to non-typical problems," in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-99)*, vol. 1. Morgan Kaufmann, 1999, pp. 916–922.

[17] W. J. Land, M. T., L. J.Y., D. McKee, and F. Anderson, "New results in breast cancer classifi cation obtained from an evolutionary computation/adaptive boosting hybrid using mammogram and history data," in *Proceedings of the 2001 IEEE Mountain Workshop on Soft Computing in Industrial Applications*. IEEE, 2001, pp. 47–52.

[18] R. Schapire and Y. Freund, "A short introduction to boosting," *Journal of Japanese Society for Artifi cial Intelligence*, vol. 14, no. 5, pp. 771–80, 1999.

[19] G. Gunnar Rätsch, T. Onoda, and K. Müller, *An improvement of AdaBoost to avoid overfi tting*, citeseer.nj.nec.com/6344.html.

[20] W. Langdon and B. Buxton, "Genetic programming for combining classifi ers," in *Proceedings of Genetic and Evolvable Computing Conference (GECCO)*. Morgan Kaufmann, 2001, pp. 66–73.

[21] K. Imamura and J. Foster, "Fault tolerant computing with n-version genetic programming," in *Proceedings of Genetic and Evolvable Computing Conference (GECCO)*. Morgan Kaufmann, 2001, p. 178.

[22] K. Imamura, R. Heckendorn, T. Soule, and J. Foster, "N-version genetic programming via fault masking," in *Proceedings of 5th European Conference EuroGP2002*. Springer, 2002, pp. 172–181.

[23] V. Hilford, M. R. Lyu, B. Cukic, A. Jamoussi, and F. B. Bastani, "Diversity in the software development process," in *Proceedings of Third International Workshop on Object-Oriented Real-Time Dependable Systems*. http://www.cse.cuhk.edu.hk/~lyu/papers.html#SFT_Techniques: IEEE Comput. Soc, 1997, pp. 129–36.

[24] S. Basak, B. Gute, G. Grunwald, D. Opitz, and K. Balasubramanian, "Use of statistical and neural net methods in predicting toxicity of chemicals: A hierarchical qsar approach," in *Predictive Toxicology of Chemicals: Experiences and Impact of AI Tools - Papers from the 1999 AAAI Symposium*. AAAI Press, 1999, pp. 108–111.

[25] D. Opitz, S. Basak, and B. Gute, "Hazard assessment modeling: An evolutionary ensemble approach," in *Proceedings of the Genetic and Evolutionary Computation Conference*, vol. 2. Morgan Kaufmann, 1999, pp. 1643–1650.

[26] R. Maclin and D. Opitz, "An empirical evaluation of bagging and boosting," in *Proceedings of the Fourteenth International Conference on Artificial Intelligence*. http://citeseer.nj.nec.com/maclin97empirical.html: AAAI Press/MIT Press, 1999, pp. 546–551.

[27] E. Bauer and R. Kohavi, "An empirical comparison of voting classification algorithms: Bagging, boosting, and variants," *Machine Learning*, vol. 36, no. 1/2, pp. 105–139, 1999.

[28] R. Kohavi, "A study of cross-validation and bootstrap for accuracy estimation and model selection," in *Proceedings of the 14th International Joint Conference on Artificial Intelligence (IJCAI)*. http://citeseer.nj.nec.com/kohavi95study.html: Morgan Kaufmann, 1995, pp. 1137–1145.

[29] D. K. Pradhan and P. Banerjee, "Fault-tolerance multiprocessor and distributed systems: Principles," in *Fault-Tolerant Computer System Design*. Prentice Hall PTR, 1996, ch. 3, p. 142.

[30] *UCI Machine Learning Repository – Molecular Biology Databases*, http://www1.ics.uci.edu/~mlearn/MLSummary.html.

[31] Q. Ma and J. Wang, "Recognizing promoters in dna using bayesian neural networks," in *Proceedings of the IASTED International Conference - Artificial Intelligence and Soft Computing*, http://citeseer.nj.nec.com/174424.html, 1999, pp. 301–305.

[32] A. Pedersen and J. Engelbrecht, "Investigations of escherichia coli promoter sequences with artificial neural networks: New signals discovered upstream of the transcriptional startpoint," in *Proceedings of the Third International Conference on Intelligent Systems for Molecular Biology*, http://citeseer.nj.nec.com/25393.html, 1995, pp. 292–299.

[33] G. Towell, J. Shavlik, and M. Noordewier, "Refinement of approximate domain theories by knowledge-based neural networks," in *Proceedings of AAAI-90*, http://citeseer.nj.nec.com/towell90refinement.html, 1990, pp. 861–866.

[34] S. Handley, "Predicting whether or not a nucleic acid sequence is an e. coli promoter region using genetic programming," in *Proceedings of First International Symposium on Intelligence in Neural and Biological Systems*. IEEE Computer Society Press, 1995, pp. 122–127.

[35] W. Banzhaf, P. Nordin, R. Keller, and F. Francone, *Genetic Programming: An Introduction: On the Automatic Evolution of Computer Programs and Its Applications*. Academic Press/Morgan Kaufmann, 1998.

[36] Y. Freung, Y. Mansour, and R. E. Schapire, "Why averaging classifier can protect against overfitting," in *Proceeding of 8th International Workshop on Artificial Intellignece and Statistics 2001*, http://citeseer.nj.nec.com/freund00why.html, 2001.