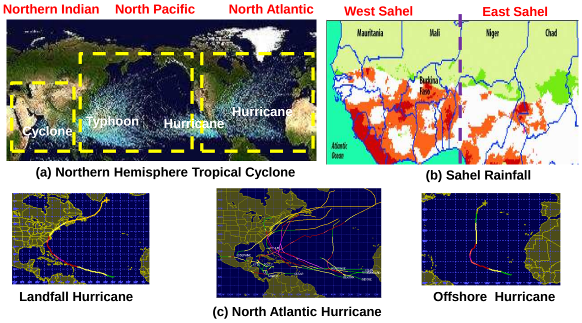
Automatic Detection and Correction of Multi-class Classification Errors Using System Whole-part Relationships

Zhengzhang Chen[*] [†] [‡] John Jenkins[*] [†] Alok Choudhary[‡] Jinfeng Rao[*] [§]
Fredrick Semazzi[*] Anatoli V. Melechko[*] Vipin Kumar[¶] Nagiza F. Samatova[*] [†] [‖]

Abstract

Real-world dynamic systems such as physical and atmosphere-ocean systems often exhibit a hierarchical system-subsystem structure. However, the paradigm of making this hierarchical/modular structure and the rich properties they encode a "first-class citizen" of machine learning algorithms is largely absent from the literature. Furthermore, traditional data mining approaches focus on designing new classifiers or ensembles of classifiers, while there is a lack of study on detecting and correcting prediction errors of existing forecasting (or classification) algorithms. In this paper, we propose DETECTOR, a hierarchical method for detecting and correcting forecast errors by employing the *whole-part* relationships between the target system and non-target systems. Experimental results show that DETECTOR can successfully detect and correct forecasting errors made by state-of-art classifier ensemble techniques and traditional single classifier methods at an average rate of 22%, corresponding to a 11% average forecasting accuracy increase, in seasonal forecasting of hurricanes and landfalling hurricanes in North Atlantic and North African rainfall.



(a) Northern Hemisphere Tropical Cyclone

(b) Sahel Rainfall

Landfall Hurricane

(c) North Atlantic Hurricane

Offshore Hurricane

Figure 1: Global extreme event forecast systems and their subsystems. (a) Northern Hemisphere TCs consist of Northern Indian cyclones, North Pacific typhoons and North Atlantic hurricanes; (b) Sahel rainfall indices differ between West Sahel and East Sahel regions; (c) Hurricanes include landfall hurricanes and offshore hurricanes.

1 Introduction

Physical and climate systems exhibit a hierarchical structure in numerous contexts. For example, global climate simulations usually have a coarse-grained structure overall with looser accuracy bounds, but also consist of *multiple, finer-grained regional structures* with higher accuracy. For example, the IPCC (Intergovernmental Panel on Climate Change) reports that global climate models (GCMs) are unable to resolve features such as clouds and topography due to the sheer size of the model and the required spatial resolution [16]. However, a number of smaller regional climate models (R-CMs) can be used, driven by boundary conditions from a GCM, to obtain higher resolutions of climate information, such as topography, large lake systems, or narrow land masses. The relationships between the GCM and the set of RCMs thus forms a hierarchical structure.

The hierarchical or modular nature of these simulations are rife with structural and semantic properties that can be utilized to mine knowledge of great interest to physical/climate scientists. Properties such as system-system feedback loops and correlations between subsystem state and global system state (or a subset thereof) are particularly prevalent, high-value targets. Furthermore, global system state and spatio-temporal trends can oftentimes be inferred from local system states, even when the whole is greater than the sum of its parts. For example, a large seasonal tropical cyclone (TC) count of Northern Hemisphere (Fig. 1 (a)) may indicate that large TC counts occur at one or all of the three subregions (North Atlantic, North Pacific, and Northern Indian) during the same season. Likewise, a large seasonal TC count of North Atlantic region implies that there is likely a large TC count in the overall Northern Hemisphere during the same season.

However, the paradigm of making this hierarchical/modular structure and the rich properties they encode a "first-class citizen" of machine learning algorithms is largely absent from the literature. For example, to predict seasonal TC, Chu *et al.* [5] constrain the prediction of extreme events to those occurring in a specific region only focused on the Taiwan region, while Kim *et al.* [14] focused on the North Atlantic region. As a result, predictions between different subsystems of a global atmospheric-ocean system, or between a subsystem and a global system, are treated independently and the *existent relationships* (see Fig. 1) between the global system and its subsystems such as the *whole-part relation* are largely ignored. Deducing these structure defi-

[*] North Carolina State University, NC 27695, USA
[†] Oak Ridge National Laboratory, TN 37831, USA
[‡] Northwestern University, IL 60208, USA
[§] Zhejiang University, Zhejiang 310000, China
[¶] University of Minnesota, MN 55455, USA
[‖] Corresponding author:samatova@csc.ncsu.edu

nitions and relationships without prior knowledge is a highly non-trivial problem, meaning that state-of-the-art machine learning can gain at most minimal access to the information these structures contain.

Rather than outright developing a machine learning algorithm for a particular simulation structure, we demonstrate the utility of exploiting hierarchical, system-subsystem relationships through a different light: through the *detection* and *correction* of classification errors of arbitrary, "black-box" classifiers. We propose an algorithm, named DETECTOR, to detect and correct potential errors in the results of an existing multi-class classification algorithm by employing the *whole-part* relationships between the global system and its subsystems. Given forecast results on a target system, DETECTOR learns non-target system class thresholds relative to the target system using linear regression, forecasts the non-target system phases based on the learned class thresholds, and uses the results and conflict rules to automatically detect and correct forecast errors on the target system. A general overview of our method is given in Fig. 2. To the best of our knowledge, no literature has addressed this problem before.

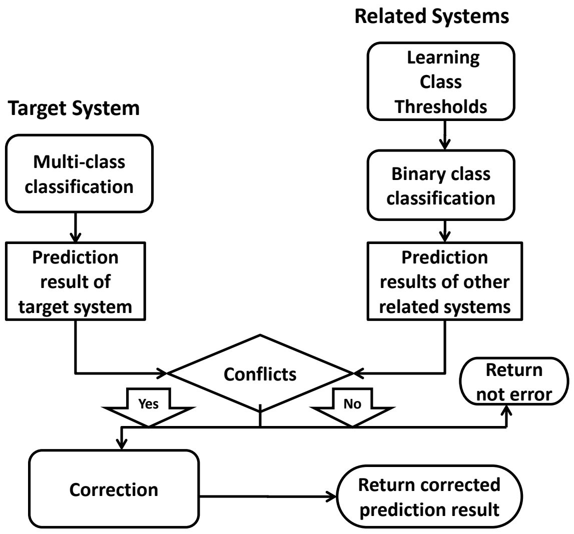

Related Systems
Target System
Learning Class Thresholds
Multi-class classification
Binary class classification
Prediction result of target system
Prediction results of other related systems
Conflicts
Return not error
Yes
No
Correction
Return corrected prediction result

Figure 2: Overview of DETECTOR methodology.

We successfully apply our methodology to seasonal extreme event forecasts of North Atlantic landfalling hurricane counts and Sahel rainfall intensity. For these two tasks, our experimental results show that the usage of DETECTOR on both state-of-the-art classifier ensemble techniques and traditional single classifier methods results in an average accuracy improvement of 12%.

2 Motivating Example

Accurate forecasting of extreme events, such as heat waves, cold waves, storms, floods, and droughts, is crucial to our society because of their catastrophic socio-economic consequences. For example, in Western Africa, periods of very low relative humidity (RH) often coincide with higher incidences of meningitis epidemics that affects more than 200,000 people throughout the African Sahel region annually [15]. The ability to forecast the occurrence of such events with greater precision could translate to taking better preventive measures to reduce the severity of the event, effectively helping decision makers to mitigate effects more adequately. Conversely, prediction errors may result in taking the wrong action, or no action at all, in response to a possible future event.

However, accurate forecasting of extreme events, for example in atmospheric-ocean systems, is a non-trivial task. Such dynamic systems are inherently complex, and often operate in multiple phases, described as having similar defining characteristics but whose feedbacks behave in a non-linear fashion [10]. The number of observational events to build the prediction models is very low, and the extreme events can occur in different locations and different times of the year. Thus, considering the fact of having only a handful of available observational events ($n \approx 100$'s) in high-dimensional spaces ($d \approx 10,000$'s), the existing machine learning methods become hardly suitable for dealing with such *underdetermined*, or *unconstrained*, problems ($n << d$). For example, applying traditional ensemble classification methods—bagging and boosting with C4.5 as their base classifier to predict North Atlantic TC over 1982–2011 time period, we manage to achieve the resulting accuracy with leave-one-out cross validation of only 33.3% for bagging and 40% for boosting. The state-of-art regression methodologies developed over the past decade also report a limited success. For example, for the Pacific region, Chu *et al.* [5] report only 58.8% accuracy over the 1970—2003 time period using the LAD regression models for this task.

Initial research has been performed to mitigate these problems, albeit in different problem domains, but much work remains to be done. For instance, the machine learning algorithms [2, 3, 17] have been specifically designed to overcome underdetermined problem spaces when performing classification. While they are largely successful in their test applications, their predictions do not take advantage of hierarchical characteristics such as system-subsystem relationships, nor are the prediction results checked against these characteristics for sanity. In this paper, we perform the latter.

3 Problem Formulation

The following three concepts provide the backdrop for our error detection and correction algorithm: the classification task for which we wish to analyze for errors, the hierarchical system-subsystem domain whose relationships we use to perform the analysis, and finally the definition of the error detection/correction problem itself. Table 1 provides the necessary symbols used in this paper.

As discussed in Section 1, the classification task is to provide a rough estimate of the number of occurrences of

Table 1: Symbol Table

Symbol	Description
Domain Information	
S_g	Global system
k	Number of subsystems of the global system
$S_1, \ldots, S_k$	Subsystems of the global system
S_t	Target system for classification
B, N, A	System activity phases/classes (B–below normal, N–normal, A–above normal)
$C \in \{B, N, A\}$	A system phase/class
Parameters ($i \neq t$)	
E_i	A series of observation event counts in S_i
$e_{j,i} \in E_i$	Event counts for jth instance in S_i
L_t^A	The lower bound of class A in system S_t
U_t^B	The upper bound of class B in system S_t
P_t	Prediction result set of test instances for S_t
$c_{j,t} \in P_t$	Predicted class for jth test instance in S_t
Derived Information ($i \neq t$)	
L_i^A	The lower bound of class A in system S_i
U_i^B	The upper bound of class B in system S_i
P_i	Prediction result set of test instances for S_i
$c_{j,i} \in P_i$	Predicted phase for jth test instance/season in S_i

some *event* of interest, discretized into classes based on the ranges of segments. That is each training *instance* contains the feature set, the number of event occurrences, and the subsequent class label. We call the ranges the *class thresholds*. For the applications in this paper, we focus on three classes: *below normal* (B), *normal* (N), and *above normal* (A), corresponding to domain-dependent expected event counts. We denote the threshold separating the classes B and N to be U^B and the threshold separating the classes N and A to be L^A.

Furthermore, the system defined by the feature set and event counts does not exist as a monolithic whole. The *global* system S_g is composed of multiple co-interacting *local subsystems* $\{S_1, S_2 \ldots, S_k\}$, where k is the number of subsystems, typically defined on spatial boundaries. For our purposes, we assume that the subsystems are disjoint and that there is no event overlap between the subsystems. In other words, the event counts follow the property $\sum_{i=1}^{k} e_{j,i} = e_{j,g}$, for all instances j. We call this the *whole-part* relationship. In the classification problem, the *target system* S_t, the system to classify, may be either the global system or one of the subsystems.

Given these two domains, the error detection and correction problem is defined as follows: Let Z be a classification algorithm and P_t be the set of class labels predicted by classification algorithm for target system S_t. The *error detection and correction problem* is to determine whether each predicted class label $c_{j,t}$ in P_t is a classification error, and if so, replace the prediction with a class label more likely to be correct.

4 Methodology

4.1 Overview In contrast to developing new classification algorithms to maximize forecast accuracy, our goal is to improve the accuracy of existing classifiers by detecting and correcting potential prediction errors. We perform this primarily through use of the *whole-part* paradigm.

Intuitively, our method is based on the key observation that the prediction results for different interacting systems (see examples in Fig. 1) are not always consistent and often have some obvious conflicts. For example, the random forest method predicts the seasonal TC of the North Atlantic region (consisting of landfall TC and offshore TC) in the year of 1950 to be above normal (class A). Based on class thresholds, the number of TCs is predicted to be larger than 7. However, when the same method is applied to the landfall and offshore TC subsystems, both predict below normal (class B), meaning the number of TCs for each subsystem is less than 3. Based on the predictions, the sum of the two subsystems is no larger than 4, which conflicts with the prediction result of the global system. This conflict tells us that either the seasonal TC of the North Atlantic predicts wrongly, or at least one of its subsystems predicts wrongly. Thus, if we could detect those conflicts among the results of global system and its subsystems, it might be possible to discover the potential prediction errors in a target system (either the global system or any of its subsystems).

However, in order to detect those conflicts, we must first derive the prediction results from each non-target system. This classification task requires class thresholds of the non-target systems. Finding good class thresholds for the non-target systems is important for effective conflict detection. In particular, we tie the class thresholds for the non-target systems to the thresholds for the target system, using a "best fit" for the purposes of error detection/correction.

Therefore, DETECTOR consists first of learning the class ranges of the non-target systems relative to the target system using linear regression, followed by applying an off-the-shelf classifier to generate the non-target system predictions, and finally deriving the conflicts and taking appropriate correction action. We base the learning and conflict detection/resolution steps on relationships between the global system and its subsystems. The key innovative steps underlying the DETECTOR methodology are summarized in Algorithm 1.

For the following sections, we assume that the target system S_t is the global system S_g, and there are only three possible system phases (classes): B, N and A in the target prediction system, for simplicity of discussion. DETECTOR also works for non-global target systems, but some minor changes are required, which we discuss in Section 4.6.

4.2 Learning Non-target System Class Thresholds To reiterate, for the target system S_g, class thresholds U_g^B and L_g^A can be determined based on the distribution of historical event counts. For example, in the case of Taiwan region TC prediction, years with fewer than three TCs are classified as below normal, and years with at least five TCs are classified

Algorithm 1: DETECTOR: Detecting and correcting prediction errors in multi-class prediction systems

```
Input:  $S_t$ : the target prediction system
        $E$ : the observation count series
        $U_t^B$: upper bound threshold of class $B$ of $S_t$
        $L_t^A$ : lower bound threshold of class $A$ of $S_t$
        $D$ : the predictor data
        $\Re$ : a linear regression algorithm
        $Z$ : a classification algorithm (e.g., decision tree)
        $P_t$ : the prediction result by $Z$ on target system
Output:
    $P_t'$: the corrected prediction result of $S_t$
for $\forall i \in \{g, 1, 2, ..., k\}$ and $i \neq t$ do
    Run regression algorithm $\Re$on training data to get compute
    $U_i^B$ and $L_i^A$ based on the class thresholds $U_t^B$ and $L_t^A$;
    Determine $c_{j,i}$ for all subsystem data using $U_i^B$ and $L_i^A$;
    Run classification algorithm $Z$ on predictor data $D$ and $c_{j,i}$
    to get all 2-class prediction results $P_i$;
end
// $|P_t|$ is the instance size of $P_t$
for $\forall j \in \{1, 2, ..., |P_t|\}$ do
    if Definitive conflicts detected in instance $j$ of $P_t$ then
        Definitive conflicts resolution;
    else if Non-definitive conflicts detected in instance $j$ of $P_t$
    then
        Non-definitive conflicts resolution;
    else if Bound conflicts detected in instance $j$ of $P_t$ then
        Bound conflicts resolution;

end
Update $P_t$ as $P_t'$;
return $P_t'$;
```

as above normal, with a distribution of 40% as normal and 30% each as below normal and above normal [5]. Using these thresholds and the event counts of all systems in the training set ($e_{j,i}$ for all training instances j and all i corresponding to systems $\{S_g, S_1, S_2, \ldots, S_k\}$), we wish to derive the "best" class thresholds for each non-target system to capture the whole-part relationship, which is crucial in correctly detecting errors (see Section 4.6.2). We model the learning process as a linear regression problem, but in order to do so, we first introduce a few definitions and constraints, based on whole-part relationships.

For our error detection process, it is necessary to align the class thresholds of the global system with those of the subsystems. Since there is a *whole-part* relationship between the global system and the subsystems, the simplest and most effective way of performing this is to partition the threshold counts of the global system among each subsystem. That is, we define variables α_i and β_i for each subsystem $S_1, S_2, \ldots, S_k$ subject to the following constraints:

$$(4.1) \qquad \begin{cases} U_i^B = \alpha_i * U_g^B \\ L_i^A = \beta_i * L_g^A \\ \sum_{i=1}^k \alpha_i = \sum_{i=1}^k \beta_i = 1, \end{cases}$$

Note that the assumption that events are disjoint between subsystems allows us to define the summation constraint on the set of α's and β's.

Based on Equation 4.1, the relationships between the thresholds of the global system and those of the subsystems are linear. Therefore, we can use linear regression to calculate each α_i and β_i.

Generally, a linear regression model with one predictor can be written as:

$$(4.2) \qquad y = b * x + \epsilon,$$

where y is the predictand vector, x is the predictor vector, and ϵ is the regression residual vector. The regression task is to find the scalar b that minimizes ϵ by some measure.

Let E_g^C be a subset of event counts in E_g, containing only event counts in training instances belonging to class C in system S_g. Let $E_{i,g}^C$ be the set of event counts in S_i corresponding to the instances in E_g^C. For example, if there are only three years (1951, 1953, 1960) where S_g is classified as B, and the event counts in those three years of S_g and some S_i are $\{2, 1, 3\}$ and $\{1, 0, 2\}$, respectively, then $E_g^B = \{2, 1, 3\}$ and $E_{i,g}^B = \{1, 0, 2\}$.

Then, we define the regression for classes B and A as follows:

$$(4.3) \qquad E_{i,g}^B = \alpha_i * E_g^B + \epsilon_B.$$

Similarly,

$$(4.4) \qquad E_{i,g}^A = \beta_i * E_g^A + \epsilon_A.$$

We find the α_i and β_i that minimizes ϵ_B and ϵ_A, in turn giving us the L_i^A and U_i^B class thresholds for each system. Different regression methods including Least Absolute Deviation regression (LAD), LSE regression and Bayesian linear regression are compared to compute the thresholds α_i and β_i. The experimental results show that there is no bearing on error detection and correction rates by choosing differen-t regression techniques. Thus, the simple LAD regression method is used in our algorithm. Because each α_i and β_i are real numbers, the constraint $\sum_{i=1}^k \alpha_i = \sum_{i=1}^k \beta_i = 1$ is not strictly achieved, but since the event subsets in each regression are disjoint, we find that $\sum_{i=1}^k \alpha_i \approx \sum_{i=1}^k \beta_i \approx 1$, which is suitable for our purposes.

4.3 Forecasting Non-target System Phases Once the class thresholds for the non-target systems are obtained, it is relatively simple to use an off-the-shelf classifier to perform the classification of the non-target systems. However, in order to get more reliable prediction results from the non-target systems, we model this step as a set of binary or two-class classification problems, rather than performing the classification as a three-class classification.

Two important observations underlying this decision are that: (1) The three-class classification problem of each non-target system has the same complexity as the original three-class classification of target system. Thus, it is hard to tell whether or not the prediction results of the non-target systems P_i are more reliable than that of target system P_g. We found in our experiments that, in many cases, the forecast accuracy of three-class classification for some P_i is even lower than the accuracy of P_g; (2) Binary classification in this context is much easier than three-class classification, and within each class (phase) a system might be able to be described in linear fashion [10], although the entire system is nonlinear.

For each non-target system S_i, we perform three binary classification tasks: B vs. $\overline{B}$, N vs. $\overline{N}$, and A vs. $\overline{A}$. The rules for generating the training set classes, given the class thresholds derived in Section 4.2, are given as follows:

$$(4.5) \qquad \begin{cases} e_{j,i} \le U_i^B \rightarrow \text{B} \\ U_i^B < e_{j,i} \rightarrow \overline{B} \\ U_i^B \le U_g^B \end{cases}$$

$$(4.6) \qquad \begin{cases} e_{j,i} < L_i^A \rightarrow \overline{A} \\ L_i^A \le e_{j,i} \rightarrow \text{A} \\ L_i^A \le L_g^A \end{cases}$$

$$(4.7) \qquad \begin{cases} U_i^B < e_{j,i} < L_i^A \rightarrow \text{N} \\ Otherwise \rightarrow \overline{N} \end{cases}$$

4.4 Target System Prediction Error Detection We develop a number of rules for error detection, based on the predicted results from the non-target systems derived in Section 4.3. The general idea behind the rules is that, if the prediction results for the non-target systems overwhelmingly conflict with the prediction results for the target system, then it is likely that the prediction result for the target system is incorrect.

Our error detection rules are applied in the order listed, for each set of binary classification results (B vs. $\overline{B}$, N vs. $\overline{N}$, A vs. $\overline{A}$). That is, our first rule is applied for all sets of binary classification results; only if the rule is not triggered for them is the next rule evaluated. As input, we examine every instance j in the set of instances to predict, the non-target prediction results $c_{j,i}$ for all i corresponding to non-target systems, and the target system prediction results $c_{j,g}$, corresponding to the global system. The following rules assume binary class labels (C vs. $\overline{C}$, where $C \in \{B, N, A\}$), but $c_{j,g}$ can easily be considered a binary class for each rule evaluation. For example, if $c_{j,g} = N$ for some j, then $c_{j,g} \neq B$ and $c_{j,g} \neq A$.

Definitive Conflict $\forall i, c_{j,i} = C$ and $c_{j,g} = \overline{C}$. In other words, all non-target systems are predicted to have class C but the target system was predicted to have a different class.

In other words, if all subsystems are predicted to have class C, the target system can not predict to have a different class. For example, if $c_{j,g} = A$ and $\forall i, c_{j,i} = B$, then $c_{j,g} = A$ is a prediction error, because it conflicts with the whole-part property: $\sum_{i=1}^{k}(e_{j,i}) = e_{j,g}$.

Non-definitive Conflict $\forall i, c_{j,i} = \overline{C}$ and $c_{j,g} = C, C \neq N$. In other words, only the target system predicts class C, where C is not the N class (the class between the upper and lower class thresholds).

For example, if the global system predicts the test instance to be A, then, for the A vs. $\overline{A}$ classification task, all subsystems predict $\overline{A}$.

Bound Conflict Let $c_{j,g} = C$ and $e_{j,i}^{min}$ and $e_{j,i}^{max}$ be the minimum and maximum number of event counts for prediction instance j in the binary classification of C vs. $\overline{C}$ of system S_i, computed using the predicted class label and class threshold bounds. Table 2 shows how the individual bounds are calculated, given the predicted class label. There are two cases to this conflict:

$$(4.8) \qquad \begin{cases} \sum_{i=1}^{k} e_{j,i}^{max} < e_{j,g}^{min} \\ \sum_{i=1}^{k} e_{j,i}^{min} > e_{j,g}^{max} \end{cases}$$

In the first case, the smallest possible prediction for the global system counts is larger than the largest possible sum of its parts. In the second case, the smallest possible sum of the subsystem counts is greater than the predicted global system counts can possibly be.

Note that this rule can be seen as a relaxation of the previous two rules. Bound conflicts will often arise when the other rules trigger, but can also trigger without the strict set of class assignments used in the other rules, since ranges rather than class labels are being used to derive the conflict.

Table 2: Minimum and Maximum Class Thresholds for Each Class of System i, Used in Computing Bound Conflicts.

Class	Min Event Count	Max Event Count
B	0	U_i^B
$\overline{B}$	$U_i^B + 1$	∞
N	$U_i^B + 1$	L_i^A
$\overline{N}$	0	∞
A	$L_i^A + 1$	∞
$\overline{A}$	0	L_i^A

4.5 Target System Prediction Error Resolution The actions we can perform given error detection vary on which rule is triggered, as well as whether the rule is triggered by one or more of the binary classification result sets.

Definitive Conflict If only one definitive conflict is discovered for instance j, simply correct the prediction result $c_{j,g}$ to the result predicted by each subsystem, since the subsystem predictions overwhelmingly disagree with the global system prediction.

It may be the case that multiple definitive conflicts arise for the same instance, under the different binary classification tasks. For example, in the B vs. $\overline{B}$ task, all instances but the target instance could be classified as B, while in the N vs. $\overline{N}$ task, all instances but the target instance could be classified as N. In this case, we can use one of two heuristics. If the classifier provides prediction probability distributions, then the second-best heuristic strategy can be employed to correct the error. For example, if $c_{j,g}$ has a probability of 51% to be B, 48% to be A, and 1% to be N, and $c_{j,g} = B$ is a prediction error detected by the rule, then we can correct the error to $c_{j,g} = A$. Without the probability distribution, a "closest-neighbor" heuristic can be used. For instance, if $c_{j,g} = B$ and the definitive conflicts imply changing the class to both N and A, then we would choose class N.

Non-definitive Conflict Without probability distributions, the error $c_{j,g} = C$ can merely be flagged, since the global system is alone in predicting a particular, "non-normal" class, absent the triggering of the definitive conflict rule.

If the classifier provides probabilistic information, then the predicted class label can be corrected based on the second-best heuristic.

Note that only one non-definitive conflict can be triggered for the set of binary classification results, due to the target system belonging to a specific class. For example, if some $c_{j,g} = B$, then the rule cannot be triggered for the A vs. $\overline{A}$ classification task.

Bound Conflict When a conflict of this nature is triggered, the natural solution is to change the predicted class label so that the bounds are no longer in conflict. However, in the case that multiple class labels would suffice (e.g., changing a predicted A to a N or B), the second-best or closest-neighbor heuristic can be applied.

4.6 Utilizing DETECTOR for Non-global System Targets When the target system is not the global system S_g but instead one of its subsystems S_m, a number of changes must be made to DETECTOR to accommodate the change in relationship between the target and non-target systems, though the changes do not warrant discussing an entirely new method. Specifically, the regression stage and error correction stage undergo minor changes, while the non-target system classification process remains unchanged.

4.6.1 Regression for Non-global System Targets Since the predetermined class thresholds are now for a subsystem rather than the global system, we first determine the bounds U_g^B and L_g^A for the global system, then use those bounds to determine the remaining subsystem bounds as before.

First, we use Equation 4.1 to define the global bound in terms of the target subsystem bound:

$$\begin{cases} U_g^B = \frac{1}{\alpha_m} * U_m^B \\ L_g^A = \frac{1}{\beta_m} * L_m^A. \end{cases} \tag{4.9}$$

Recall that E_g^C is the set of event counts in S_g with class label C, and $E_{i,g}^C$ the corresponding set of event counts for S_i. We perform regression on these to compute α_m and β_m:

$$E_{g,m}^B = \frac{1}{\alpha_m} * E_m^B + \epsilon'_B \tag{4.10}$$

Similarly,

$$E_{g,m}^A = \frac{1}{\beta_m} * E_m^A + \epsilon'_A \tag{4.11}$$

The regression steps for the remaining systems proceed as normal.

4.6.2 Error Detection/Correction Rules for Non-global System Targets The detection rules for non-global target systems are the same as those for a global target system, except now the actions arising from each rule are different.

Definitive Conflict In this case, all subsystems in instance j, including $c_{j,m}$, are predicted to have class C, but the global system class is predicted to be $\overline{C}$. This strongly suggests that the target system classification should be changed to another class, and the second-best heuristic in this case can be used, because we do not know which alternate class would be more accurate.

Non-definitive Conflict In this case, only the global system is predicted to have class C, and all other classes, including $c_{j,m}$, are predicted to have class $\overline{C}$. This strongly suggests that the target system classification should be changed to class C.

Bound Conflict There are no changes to bound conflict resolution except that the target system being changed is a subsystem rather than the global system.

4.7 Generalization to More than Three Class Labels Thus far, we have only presented our DETECTOR algorithm for three-class classification problems. However, our DETECTOR method can be generalized to detect the prediction errors in multi-class ($|C| > 3$) classification results, where $|C|$ is the number of different class labels.

Specifically, for the regression step, $|C| - 1$ class thresholds need to be derived rather than U_i^B and L_i^A, requiring minimal changes to the underlying method. For the binary classification step, $|C|$ binary classification tasks are performed instead of three binary classification tasks for each non-target system S_i, with similar class assignment rules. The detection step is similarly alike to the three-class classification error detection. The definitive and bound conflicts are the same, while the non-definitive conflict changes slightly: instead of the conflict applying for $C \neq N$, the conflict applies for only the classes representing the minimum and maximum event range, respectively. Finally, the error correction rules are the same, although future work will focus on developing a more robust and complete set of rules that can be applied to ordinal class labels representing integer ranges.

5 Experiments

5.1 Real-world Extreme Event Prediction Tasks Two real-world extreme event prediction tasks (see Fig. 1) are considered in this paper:

1. *Seasonal hurricane prediction* (SHP): The first task is to forecast the seasonal hurricane count, with emphasis on landfall hurricanes. We build up a North Atlantic (NA2) TC system consisting of the landfall hurricane system (LNA) and the offshore hurricane system (ONA). TC in this task only includes hurricanes. NA2 and LNA are used as target systems, respectively.

2. *North Africa rainfall prediction* (NARP): The second task is to forecast the seasonal rainfall in North Africa, specifically in the Sahel area (SH), which is an important problem highly related to meningitis epidemics that affects more than 200,000 people throughout the African Sahel region annually. East Sahel (ES) and West Sahel (WS) are considered as subsystems of NARP in this task. SH and WS are used as target systems, respectively.

5.2 Extreme Event Data For the SHP experiments, we use the seasonal tropical cyclone (TC) count series from 1950 to 2011 of North Atlantic. These series are obtained from Atlantic hurricane database (HURDAT) at the National Climatic Data Center, Central Weather Bureau [5], and JTWC Northern Indian Ocean best track data. The landfall hurricanes that strike land are distinguished by using the "Hit" feature of the HURDAT. Likewise, hurricanes that made landfall in Mexico are also considered as landfall hurricanes in our analysis.

For the NARP experiments, the East Sahel and West Sahel rainfall indices from 1951–2004 are obtained from HURDAT by averaging seasonal (July through September) mean Precipitation data over (10-20°N, 0–20°E) and (10–20°N, 20W–0°E), respectively. Sahel rainfall indices are calculated as the sum of the East Sahel and West Sahel rainfall indices. Monthly rainfall data is obtained from the Climate Research Unit at a $0.5^\circ \times 0.5^\circ$ latitude and longitude resolution.

For these experiments, relative humidity (RHUM) and tropospheric vertical wind shear (VWS) data in preceding June are used as predictors to forecast the SHP TC, and wind speed (WSPD) data are used to forecast NARP rainfall classes. RHUM and WSPD are available on NCEP/NCAR reanalysis data repository with a $2.5^\circ \times 2.5^\circ$ latitude and longitude resolution. VWS is calculated by computing the square root of the sum of the square of the difference in zonal wind component between 850 and 200 hPa levels and the square of the difference in meridional wind component between 850 and 200 hPa levels [6] from NCEP/NCAR reanalysis data.

5.3 Feature Selection Due to the high-dimensional feature spaces ($d \approx 10,000$'s), after the aforementioned mathematical abstraction of the original multivariate spatio-temporal data, the next step is feature selection designed to filter out redundant features or "noise." Since the feature selection technique is out of the scope of this paper, we use simple Pearson correlation-based pruning as a measure of feature significance [5]. For the feature series F and the extreme event series R the correlation δ is computed as $\delta(F,R) = \frac{\sum(f_i-\bar{f})(r_i-\bar{r})}{\sqrt{\sum(f_i-\bar{f})^2\sum(r_i-\bar{r})^2}}$, where f_i is the i^{th} value in F and $\bar{f}$ is the mean of all values in the series. Note that the correlation coefficient has a range of [-1, 1], where 1 denotes perfect agreement and -1 perfect disagreement, with values near 0 indicating no correlation. Since an inverse relationship is equally relevant in the present application, we set the correlation score to $|\delta|$, the absolute value of the correlation coefficient. Although nonlinear relationships are known to exist in climatological systems, the observed similarity of Pearson correlation still can be considered statistically significant, as concluded by Donges *et al.* [7]. Thus, we use Pearson correlation to measure the similarity between the series of each climate feature and the extreme event counts in this work. Each feature are considered statistically significant if and only if the corresponding p-value of the correlation score is less than 0.1.

Note that it may be possible for some of the test classifiers to perform better given different predictors, and the optimal set of predictors in our evaluation problems may differ from one classifier to another. However, as the goal of this paper is to detect and correct classification errors, we use the same set of predictors, as described in section 5.2, for all classifiers to provide a fair cross-classifier comparison of DETECTOR's performance.

5.4 Evaluation Methodology and Metrics To avoid bias, we do feature selection only from the randomly selected 32-year data in SHP experiments and 24-year data in NARP experiments. And the remaining 30-year data in SHP or NARP experiments are used as the evaluation data sets to perform the cross validation. Because of the small sample size of the spatio-temporal data, leave-one-out cross validation (LOOCV) is employed to evaluate DETECTOR's robustness. In LOOCV, a single instance from the original sample is used as the validation data, and the remaining instances are used as the training data. This procedure is repeated until all instances are tested once.

DETECTOR, being a method that uses existing classification results as input, requires evaluation metrics in addition to standard classifier metrics:

1. Accuracy (Ac): the ratio of the number of correctly classified data points to the total number of data points in the test set.
2. Detection rate (σ): the effectiveness of the forecast

error detection. The detection rate is calculated as the fraction of the correctly detected forecast errors:

$$\sigma = \frac{num_{d_e}}{num_e}, \tag{5.12}$$

where num_{d_e} is the number of forecast errors correctly detected, and num_e is the total number of existing forecast errors.

3. Detection error (ω): the false detection rate of DETECTOR. It measures the fraction of the wrongly detected forecast errors in all detected errors.
4. Correction rate (μ): the effectiveness of the forecast error correction. The correction rate is calculated as the fraction of the forecast errors correctly remedied:

$$\mu = \frac{num_{r_e}}{num_e}, \tag{5.13}$$

where num_{r_e} is the number of forecast errors corrected accurately. The upper bound of the correction rate is given by the detection rate.

5. Improved accuracy (ΔAc): the performance difference between accuracy of the underlying classifier and accuracy after forecast error correction by DETECTOR.

Table 3: DETECTOR's LOOCV Performance on Classifier Ensembles with Global Systems as Target Systems

Method	S	Ac	ΔAc	Ac+ΔAc	σ	ω	μ
C4.5	NA2	33.3	16.7	50.0	40.0	0	25.0
	SH	33.3	16.7	50.0	75.0	16.7	40.0
BO	NA2	33.3	30.0	63.3	50.0	9.1	50.0
	SH	36.7	6.7	43.3	47.4	10.0	15.6
MBO	NA2	33.3	23.3	56.6	40.0	0	35.0
	SH	33.3	16.7	50.0	50.0	9.1	30.0
BA	NA2	40.0	23.3	63.3	50.0	10.0	44.4
	SH	33.3	16.7	50.0	75.0	16.7	40.0
RS	NA2	36.7	33.3	70.0	68.4	7.1	57.8
	SH	40.0	6.7	46.7	50.0	25.0	27.8
END	NA2	36.7	20.0	56.7	57.8	8.3	36.8
	SH	26.7	13.3	40.0	50.0	8.3	22.7
SPICE	NA2	70.0	6.7	76.7	22.2	0	22.2
	SH	76.7	3.3	80.0	14.3	0	14.3

S: target system; BO: boosting; MBO: MultiBoosting;
BA: bagging; RS: random subspace; END: nested dichotomies.
All numbers are in percentages.

5.5 DETECTOR's Performance on Real-world Extreme Event Prediction Tasks To test DETECTOR's performance on detecting and correcting forecast errors,of we apply DETECTOR to numerous classifiers and classifier ensembles. In particular, we use C4.5 [18], boosting [9], MultiBoosting [19], bagging [1], random subspace method [11], nested dichotomies [8] and SPICE algorithm [2]. And C4.5 is used as the base learner for those ensemble methods. To eliminate bias in the choice of classifier (such as by choosing a better underlying classifier for DETECTOR to use versus the ones being tested), the underlying classifier for DETECTOR is the same as the classifier being tested. For example, DETECTOR uses C4.5 when detecting/correcting errors by the C4.5 classifier. Table 3 summarizes DETECTOR's ability to improve the LOOCV performances of classifier ensembles with global systems as the target systems, which shows that DETECTOR can correctly detect more than 40% forecast errors and improve classification accuracy by at least 6.7%.

By contrast, Table 4 shows classification and DETECTOR results for non-global target systems. In this case, DETECTOR is only able to improve the LOOCV performances of classifiers by up to 10%, though the average false detection rate (ω) of DETECTOR in subsystems results is near zero. One possible reason for difficult detecting the prediction errors in subsystems as the target systems is that the non-target system relationships are more complicated when a subsystem is chosen as the target system.

Table 4: DETECTOR's LOOCV Performance on Classifier Ensembles with Subsystems as Target Systems

Method	S	Ac	ΔAc	Ac+ΔAc	σ	ω	μ
C4.5	LNA	40.0	10.0	50.0	16.7	0	16.7
	WS	33.3	6.7	40.0	20.0	20.0	15.0
BO	LNA	36.7	6.7	43.3	10.5	0	10.5
	WS	50.0	10.0	60.0	26.7	0	20.0
MBO	LNA	36.7	10.0	46.7	21.1	0	15.8
	WS	53.3	6.7	60.0	21.4	0	14.3
BA	LNA	46.7	3.3	50.0	6.3	0	6.3
	WS	56.7	0	56.7	0	0	0
RS	LNA	30.0	3.3	33.3	14.3	0	4.8
	WS	40.0	6.7	46.7	11.1	0	11.1
END	LNA	36.7	3.3	40.0	10.5	0	5.3
	WS	50.0	6.7	56.7	26.7	3.3	26.7
SPICE	LNA	80.0	3.3	83.3	16.7	0	16.7
	WS	70.0	6.7	76.7	22.0	0	22.2

Notes: S: target system; BO: boosting; MBO:multiBoosting;
BA: bagging; RS: random subspace; END: nested dichotomies.
All numbers are in percentages.

6 Related Work

Classification has been extensively studied in various domains, including extreme event prediction [3, 4], text classification [12], and sentiment analysis [13], etc. To the best of our knowledge, this paper is the first work addressing the detection and correction of potential errors in prediction results of a multi-class classification algorithm.

Recently, classifier ensemble methods have witnessed a growing interest. In contrast to the single classifiers, the ensemble methods construct multiple "base" classifiers and combines their predictions to produce aggregate prediction. Thus, ensemble methods are also considered as one type of meta learning algorithms. Classifier ensemble methods can be categorized into instance-based re-sampling and feature-based sub-sampling.

Instance-based re-sampling generates the training sets by re-sampling from the original dataset with instance replacement. Two well-known instance-based re-sampling methods are bagging [1] and boosting [9]. Bagging generates a set of classifiers by re-sampling the training data using a bootstrap technique. The prediction results of all classifiers are combined by majority voting with equal weights. Bagging has been found to be a good method to reduce variance and help to avoid overfitting. The boosting method tries to boost the performance of a weak classifier or learner. It generates the classifiers by using a different re-sampling technique based on the sample distribution.

In contrast, feature-based sub-sampling generates the training sets by using different subsets of the original feature set. There are also two types of feature subset-based ensemble methods: random-based and reduct-based. The random-based strategy is based on the random forest concept [11] when a set of features with fixed size is randomly chosen to build a "base" classifier. One example of random-base strategy is random subspace method [11]. The random subspaces method builds a set of classifiers by random selecting different subsets of features from the original feature set. In the reduct-based strategy, several reducts are combined to create an ensemble of classifiers, where reduct is defined as the smallest subset of features that has the same predictive power as the whole feature set. An example of reduct-base strategy is the SPICE algorithm [2]. SPICE is an iterative feature subsets enumeration method, which has demonstrated its superior performance in terms of various skill and robustness metrics on solving the highly underdetermined problem.

7 Conclusion

In this paper, we have addressed an important and novel machine learning problem, namely, forecast error detection and correction in an existing target system. We have proposed DETECTOR, a hierarchal method for detecting and correcting forecast errors by employing the whole-part relationships between target system and non-target systems. After applying to two extreme event prediction use cases, DETECTOR successfully detected and corrected by up to 57% forecast errors in the results of state-of-art classifier ensemble techniques and traditional single classifier methods with an average of 11% accuracy increase. An interesting direction to further explore would be incorporating more relationships besides whole-part relationship in DETECTOR algorithm.

Acknowledgments

This work was supported in part by the U.S. Department of Energy, Office of Science, the Office of Advanced Scientific Computing Research (ASCR) and the Office of Biological and Environmental Research (BER) and the U.S. National Science Foundation(Expeditions in Computing). This work by ZC and AC was supported in part by NSF award numbers CCF-0833131, CNS-0830927, IIS- 0905205, CCF-0938000, CCF-1029166, and OCI-1144061, and in part by DOE grants DE-FG02-08ER25848, DESC0001283, DE-SC0005309, DESC0005340, and DESC0007456.

References

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

[2] Z. Chen, K. Padmanabhan, A. Rocha, Y. Shpanskaya, J. Mihelcic, K. Scott, and N. Samatova. SPICE: discovery of phenotype-determining component interplays. *BMC Systems Biology*, 6(1):40+, 2012.

[3] Z. Chen, W. Hendrix, H. Guan, I. Tetteh, A. Choudhary, F. Semazzi, and N. Samatova. Discovery of extreme events-related communities in contrasting groups of physical system networks. *Data Mining and Knowledge Discovery*, 2012.

[4] H. Sencan, Z. Chen, W. Hendrix, T. Pansombut, F. Semazzi, A. Choudhary, A. Melechko, V. Kumar, and N. Samatova. Classification of emerging extreme event tracks in multivariate spatio-temporal physical systems using dynamic network structures: application to hurricane track prediction. In *IJCAI*, pages 1478–1484, 2011.

[5] P. S. Chu, X. Zhao, C. T. Lee, and M. M. Lu. Climate prediction of tropical cyclone activity in the vicinity of Taiwan using the multivariate least absolute deviation regression method. *Terr. Atmos. Ocean. Sci.*, 18(4):805–825, October 2007.

[6] J. D. Clark and P. S. Chu. Interannual variation of tropical cyclone activity over the Central North Pacific. *JMSJ*, 80(3):403–418, 2002.

[7] J. F. Donges, Y. Zou, N. Marwan, and J. Kurths. Complex networks in climate dynamics. *The European Physical Journal - Special Topics*, 174(1):157–179, July 2009.

[8] E. Frank and S. Kramer. Ensembles of nested dichotomies for multi-class problems. In *ICML)*, pages 305–312. ACM Press, 2004.

[9] Y. Freund and R E. Schapire. Experiments with a New Boosting Algorithm. In *ICML*, pages 148–156, 1996.

[10] J. W. Gibbs. On the equilibrium of heterogeneous substances. 1876.

[11] T. K. Ho. The random subspace method for constructing decision forests. *IEEE Trans. Pattern Anal. Mach. Intell.*, 20(8):832–844, 1998.

[12] X. Hu and H. Liu. Text analytics in social media. *Mining Text Data*, pages 385–414, 2012.

[13] X. Hu, L. Tang, J. Liu, and H. Liu. Exploiting social relations for sentiment analysis in microblogging. In *WSDM*, 2013

[14] H. M. Kim and P. J. Webster. Extended-range seasonal hurricane forecasts for the North Atlantic with a hybrid dynamical-statistical model. *Geophys. Res. Lett.*, 37(21):L21705, 2010.

[15] A. M. Molesworth, L. E. Cuevas, S. J. Connor, A. P. Morse, and M. C. Thomson. Environmental risk and meningitis epidemics in Africa. *EID*, 9(10):1287–1293, 2003.

[16] Intergovernmental Panel on Climate Change. The regional impacts of climate change.

[17] T. Pansombut, W. Hendrix, Z. J. Gao, B. E. Harrison, and N. F. Samatova. Biclustering-driven ensemble of bayesian belief network classifiers for underdetermined problems. In *IJCAI*, pages 1439–1445, 2011.

[18] J. R. Quinlan. Improved use of continuous attributes in c4.5. *CoRR*, cs.AI/9603103, 1996.

[19] G. I. Webb. Multiboosting: A technique for combining boosting and wagging. *Machine Learning*, 40(2):159–196, 2000.