7AI Ensemble Classifier for Winter Storm Precipitation in Polarimetric Radar Data

Rajibul Alam
School of Computer Science
University of Oklahoma
Norman, OK
rajibul@ou.edu

Emmanuel Goossaert
School of Computer Science
University of Oklahoma
Norman, OK
emmanuel.goossaert@ou.edu

Abstract

The precipitation classification problem for this contest consists of classifying data instances into three class types: frozen, liquid and none. Based on our experiments on the data set using available open source classifiers, we observed that different algorithms have different precisions for each class type. Thus, we decided to develop two classifiers adapted to the data set: Neural Network and Decision Tree. Our implemented solutions had similar performance issues as the open source classifiers. Since individual classifiers are unable to provide high accuracy classification for all of the class types, we chose to combine the best classifiers using a multi-class linear ensemble technique. This technique allows the different classifiers to vote based on their accuracy and precision for each class type. The obtained solution performs better compared to the individual classifiers that compose it. These classifiers are: Decision Tree C4.5 and J48, Neural Network, Naive Bayes, Naive Bayes Tree, Bayes Network, Random Forest, OneR and SMO.

1. Introduction

During winter storms, the prediction of precipitation types is vital for taking early safety measures. The first attempts of such prediction have been made using data provided by polarimetric radars, but they produced extremely poor results. Due to this failure, several attempts with new techniques have been made to improve the classification accuracy, and this project contributes toward the same objective.

One of the main reasons behind the failure to achieve high accuracy is that the problem is a three class classification. Therefore, the task is harder compared to a two class problem. This issue has been observed in a similar classification experiment which consists in the forecasting of short-term rain falls based on radar data. It has been shown that accuracy drops when the number of classes increases from two to three (Ingsrisawang et al. 2008). It has also been observed that different classifiers are able to classify different precipitation types with different precisions. Therefore, only one classifier is not enough to classify all types of precipitation properly and doesn't provide high accuracy rates. Considering these limitations, we combine several classification techniques into an ensemble classifier. Our hypothesis in this regard is that the ensemble technique will give higher accuracy compared to each of the individual classifiers.

2. Precipitation Classification

The task is to classify each instance of our dataset into one of the three categories: liquid, none or frozen. The data comes from meteorological analyzes of environmental conditions close in time and space to the actual observations, along with data from a polarimetric radar. The data contains fifteen attributes, excluding the index and instance label, and includes information regarding the location, temperature, humidity, freeze level, radar information, etc. Considering these attributes, it is obvious that data preprocessing is required. As a consequence, for the training of our classifiers we removed the longitude and attribute from the data, since they are more biased than meaningful for our classification problem.

This classification of precipitation types is extremely challenging and interesting due to the fact that it just uses meteorological and radar data. In addition, since several known classifiers have already been applied to this problem with no significant high accuracy, it provides an opportunity to experiment with some new ideas. This classification problem is also important because if a highly accurate classifier can be designed, then it would be possible to take measures early for public safety. This would significantly cut off the loss that people undergo due to winter storms each year.

3. Classification Algorithms

As mentioned before, we plan to design an ensemble machine learning classifier to address this precipitation classification problem. The critical factor to achieve this goal is the selection of classifiers, since it is vital to achieve high accuracy from an ensemble technique. Thus, based on performance mentioned in different related classification experiments (Jain et al. 1996; Umehara et al. 2008; C. Venkatesan 1997; Alexiuk et al. 1999), we decided to implement a neural network and a decision tree specifically for this problem. Moreover, we use some available classifiers from the Weka open source library(Witten and Frank 2005) to populate our ensemble method. This process is illustrated in Figure 1.

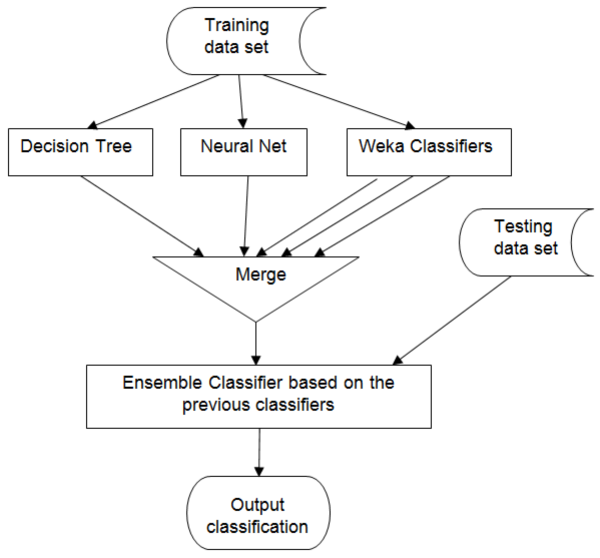


Figure 1: Ensemble method process

3a. *Decision tree*

Since the problem is a three-class problem, not all decision tree algorithms can be used. We decided to use the C4.5 algorithm, because the entropy information function allows to classify over a deterministic number of classes. In addition, all the attributes in the data set are interval attributes. Our implementation of the C4.5 algorithm takes this into account, so that each splitting at a node always creates a maximum of two sub-nodes.

We have two kind of nodes: split nodes and leaf nodes. Split nodes contain a real value used as a threshold, which allows to search into the tree. Leaf nodes just contain the identifier of the class at this location of the tree. The construction of the tree, that is to say the splitting process, stops when all nodes at the bottom of the tree are leaf nodes. The subset of the data set described by a leaf node contains instances of only one single class. This way, when a request is performed on the tree, it is directed down the tree by the split nodes until it hits a leaf node, which gives the computed classification.

While we are building the tree, we are testing it at each step. Obviously, not all of the leaf nodes have been created at those steps, thus when a request hits a split node which does not have any sub-nodes, it classifies the instances using the proportion of each class observed at this node.

3b. *Neural network*

The implemented neural network contains thirteen input neurons, one for each attribute after filtering out the location information, and obviously three output nodes to represent the three class types. Considering the dataset and the number of attributes, the network contains only one hidden layer with eight hidden neurons. The number of neurons in this layer is determined based on the performance of the network after some experiment by trying with several number of nodes (starting from three hidden nodes and going up to ten).

The network is fully connected. Each of the connections has a random weight at start within the range [-1, 1], and based on back-propagation these weights are then updated. A feed forward process is used to get the outputs from each node, and a sigmoid function is used in hidden and output layers to compute the output values. The sigmoid function is used mainly because it generates value between 0 and 1, and as this problem is a three class classification problem, sigmoid works better compared to tanh. Learning rate is another vital component of a neural network. After several trials, we decided to use a variable learning rate to help the network to achieve a higher accuracy. The learning rate starts at 0.005 and is decreased by half each time the network reaches a local minima. Eventually, the learning process is stopped using an early-stop method when the difference between two consecutive accuracy rates goes below a certain threshold.

3c. *Ensemble learning*

The well-known Bagging, Boosting and Ada Boost techniques are not applicable in our case (Polikar 2007; Ratsch 2003). They consist of using the same classifier several times, whereas we want to use different kinds of classifiers all together. Thus we choose to use a weighted voting system (Polikar 2006). That is, each of the classifiers has a specific weight on the votes, based on the precision of its predictions for each class. For instance, if one classifier has better precision in classifying the frozen instances, then when this classifier classifies an instance as ”frozen” it is given higher weights compared to another algorithm classifying the same instance as ”none”. But as the neural network and the decision tree are implemented specifically for this classification problem, these two classifiers get more weight on their votes compared to the classifiers from the Weka library.

Also, considering the precision problems on the liquid and none classes, we decided to include a biased classifier in our ensemble technique. That is, we trained a decision tree using our algorithm, but with a specific sub set of the data set, containing more instances of the liquid class compared to the frozen and none classes. The goal is to obtain a classifier that is able to classify better the liquid instances rather than a classifier trained on the whole data set. And since the data set does not present enough instances of the none class, we do not use a none biased classifier.

The classifiers from Weka are used with their default settings, and are selected based on their accuracy on a 10-fold cross validation process. Table 1 in Section 5 contains a complete list of those classifiers along with their performance metrics. The whole ensemble process is very straight forward. This process is described by the following steps:

1. Train the Neural network with the training set. Assign voting weight based on accuracy.

2. Train the decision tree. Assign voting weight based on accuracy.

3. Train weka classifiers using the training set. Assign voting of each class based on precision of the class along with accuracy of the classifiers.

4. Feed the testing set to the Weka classifiers, neural network and decision tree. Store the results for each classifier separately.

5. Using the precision and accuracy of each classifier on each class type, assign weights. Then multiply the votes of each classifier by its weights, and sum up the obtained values for all the classifiers over each class type. The final classification is made based on the highest vote.

4. Performance Metrics and Validation Process

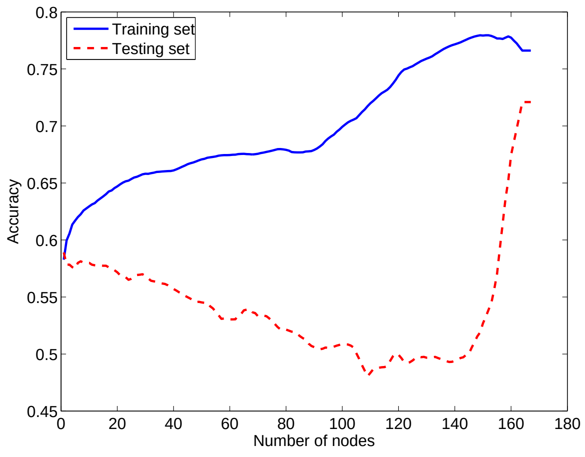
We use two different metrics to test the performance of our algorithms. The first one is the accuracy (ACC), which corresponds to $1 - errorrate$, and the second one is the Pierce Skill Score, also called True Skill Statistic (TSS). We prefer TSS over the accuracy because it is a more meaningful metric for multiclass problems, since it takes into account the classification over all the classes, whereas the accuracy simply considers the fact that an occurrence is either well classified or misclassified.

Since the performance of a classifier depends mostly on the training set which is used, we decided to adopt the cross-validation methods with 10 folds. That is, the initial dataset is decomposed into 10 equally sized folds, and the classifiers are built and tested 10 times, each time with 9 of the folds used as the training set and the remaining one used as the testing set. Then, we average the performance values obtained on each of those trials in order to obtain a more accurate estimation of the real performance.

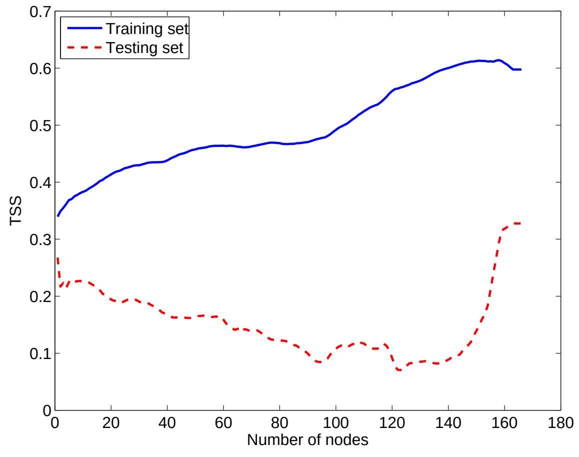
5. Results

5a. *Decision tree results*

The results we obtain with the decision tree are not common. The performance of the testing set seems to decrease over the whole learning process, which tends to be quite logical since the tree is probably overfitting very early. But there is a huge jump in the last twenty steps, and we think that some really meaningful splittings must occur in this period. Still, we consider that in our case, no pruning phase is necessary, and we will use the computed tree just as it is in its final state. We manage to reach a TSS of 0.33, which we think is a good value considering the difficulty of the task.



(a) Accuracy



(b) TSS

Figure 2: Decision tree performance with 10-fold cross-validation

5b. *Neural network results*

Figure 3 shows how the accuracy of the neural network increases along with learning steps. We present one curve for every class, and thus we can observe that the network presents a different accuracy for every class. It achieves to get an accuracy of 74% and a TSS of 0.55. We cannot compute a 10-fold cross validation test on the neural network, because the time required by the learning process is too long. This is why we cannot provide values for the testing in Table 1. Finally, the neural network provides classification probabilities for the three classes, and instances are classified into certain types based on highest probabilities.

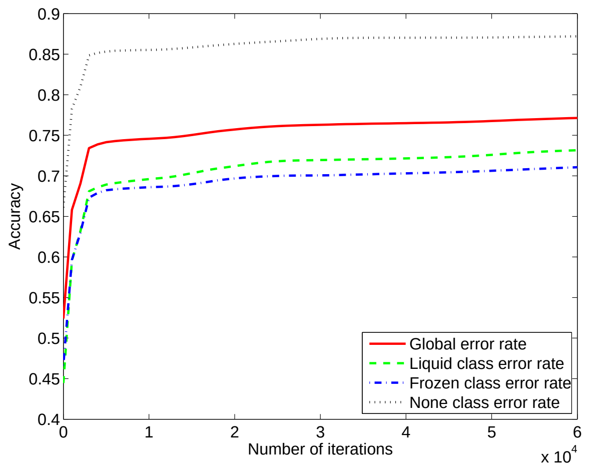


Figure 3: Neural network learning curve

5c. *Ensemble technique results and performance summarization*

With regard to the ensemble technique, we think that the lack of performance comes from the fact that all the classifiers we are using have the same precision in their classification of the three different classes. It seems that the frozen state is easily classified, but that the two other states, liquid and none, are most of the time wrongly classified. Thus we think that most of the errors observed in the final classifiers and in the ensemble classifier are due to their inability to correctly classify liquid and none instances. The fact is our data set only contains around 250 liquid instances. As a consequence, this is possible that the classifiers do not have enough information to correctly learn how to classify those instances. A better performance may be reach with more data of the liquid class.

Our ensemble technique is performing better than all our classifiers taken separately, based on the TSS metric. We obtain a value of 0.36 whereas the higher value reached so far was 0.33 by our implementation of the C4.5 algorithm for the decision tree. This represents a meaningful increase of performance. Table 1 summarizes all the results we obtain thanks to a 10-fold cross validation over all the classifiers.

6. Related Work

Radar data has already been used for classification. During the developed a multi-state precipitation es-

Algorithm	Accuracy		TSS	
	Train	Test	Train	Test
Decision tree C4.5	77%	72%	0.60	0.33
Neural network	74%	N/A	0.55	N/A
Random algorithm	33%	33%	0.0	0.0
Bayes network	66%	61%	0.41	0.27
Naive Bayes	64%	61%	0.38	0.28
Naive Bayes tree	74%	56%	0.55	0.21
Decision tree J48	83%	65%	0.71	0.31
Random Forest	98%	57%	0.97	0.16
OneR	69%	65%	0.46	0.33
SMO	67%	62%	0.43	0.32
Ensemble technique	82%	66%	0.70	0.36

Table 1: Accuracy and TSS for all the classifiers computed over a 10-fold cross validation

timation system using support vector machines and neural networks, an error rate inferior to 15% has been achieved (Umehara et al. 2008).

Also, genetic algorithms can be used for this kind of classification. It has been shown that one can find the best linear discriminant line into the data set, and classify the data set into two classes, rainy and non-rainy (Sen and Oztopal 2001). A small number of input parameters is used, but an important number can be used to refine the results. This solution provides only 13% of misclassification, which constitutes a very good result. We are looking forward to seeing if such a technique could correctly classify our precipitation data.

Finally, it is possible to use a set of neural networks. Given a set of neural networks which perform worse individually compared to a highly trained neural network, it is possible to merge those networks into an ensemble technique, which will perform better than the highly trained neural network (Hansen and Salamon 1990).

7. Future Work

The main problem of our approach is that it does not solve the problem of the liquid and none classes. Liquid and none instances are apparently very hard to predict, at least for the classifiers we trained and those we tried using the Weka library. Most of the mistakes in classification come from these labels. Consequently, there are two possible ways to solve the problem, we can either add more instances of the liquid and none instances to the training data, and see if the performance improves, or we can find a classifier that performs well on these classes, or at least on one of them, and merge its capability to the ones we already have in our ensemble technique.

8. Conclusion

Throughout this paper, we have presented the implementation of two classifiers along with an ensemble classifier, in order to settle our three-class precipitation classification problem. We have shown that the classifiers we developed achieved better performance than the general classifiers available in the Weka library. But due to a serious problems with regard to the classification of the liquid and none instances, it seems that all the classifiers obtain globally the same classification precision. Ensemble techniques are only relevant if the classifiers complement one another (Dietterich 2002), which is clearly not the case here. In order to increase significantly the performance of the classification, we need to develop a classifier able to correctly classify the liquid and none instances.

References

Alexiuk, M., N. Pizzi, and W. Pedrycz, 1999: Classification of volumetric storm cell patterns. *Conference on Electrical and Computer Engineering CCECE-99*, 1081–1085.

C. Venkatesan, R. K., S.D.Raskar, 1997: Prediction of all india summer monsoon rainfall using error-back-propagation neural network. *Proceeding of the Metrol Atomos Phy 62*, 225–240.

Dietterich, T. G., 2002: *In The Handbook of Brain Theory and Neural Networks*. The MIT Press.

Hansen, L. K. and P. Salamon, 1990: Neural network ensembles. *IEEE Trans. Pattern Anal. Mach. Intell.*, volume 12, 993–1001.

Ingsrisawang, L., S. Ingsriswang, S. Somchit, P. Aungsuratana, and W. Khantiyanan, 2008: Machine learning techniques for short-term rain forecasting system in the northeastern part of thailand. *Proceeding of the World academy of science, engineering and technology*, volume 31, 248.

Jain, A. K., J. Mao, and K. Mohiuddin, 1996: Artificial neural networks: A tutorial. *IEEE Computer*, **29**, 31–44.

Polikar, R., 2006: Ensemble based systems in decision making. *IEEE Circuits and Systems Magazine*, volume 6, 21–45.

— 2007: Bootstrap inspired techniques in computational intelligence: ensemble of classifiers, incremental learning, data fusion and missing features. *Signal Processing Magazine, IEEE*, volume 24, 59–72.

Ratsch, G., 2003: Robust multi-class boosting. *EuroSpeech IEEE*.

Sen, Z. and A. Oztopal, 2001: Genetic algorithms for the classification and prediction of precipitation occurrence. *Hydrological Sciences Journal*, volume 46, 255–267.

Umehara, S., T. Yamazaki, and Y. Sugai, 2008: A precipitation estimation system based on support vector machine and neural network. *IEICE Transactions on Information and Systems, Pt.2 (Japanese Edition)*.

Witten, I. H. and E. Frank, 2005: *Data Mining*. Morgan Kaufmann.