AdaBoost with Neural Networks for Yield and Protein Prediction in Precision Agriculture

Amy Peerlinck
Gianforte School of Computing
Montana State University
Bozeman, MT, USA
amy.peerlinck@student.montana.edu

John Sheppard
Gianforte School of Computing
Montana State University
Bozeman, MT, USA
john.sheppard@montana.edu

Jacob Senecal
Gianforte School of Computing
Montana State University
Bozeman, MT, USA
jacob.senecal@student.montana.edu

***Abstract*—Adaptive Boosting, or AdaBoost, is an algorithm aimed at improving the performance of ensembles of weak learners by weighing the data itself as well as the learners. Two versions of AdaBoost—AdaBoost-R2 and AdaBoost RΔ—are applied in this project, as well as a third novel algorithm combining ideas of these two methods, to the problem of predicting crop yield and protein content in support of precision agriculture. All three algorithms use Feedforward Neural Networks (FFNN) trained with backpropagation as the weak model. Data from four different fields were gathered as a result of on-farm experiments of different nitrogen rate applications using randomly stratified trials based on previous years' yield and protein. The three AdaBoost algorithms are compared to a simple FFNN with a single hidden layer. The results confirm previous findings in different fields, where ensemble methods outperform single models. The results are improved by 3 to 10 units for yield prediction, and by a small percentage for protein prediction.**

I. Introduction

Adaptive Boosting (AdaBoost) trains several weak models on the same data but with different sampling weights for each model and combines the predictions across all models for the final prediction [7]. AdaBoost was first proposed to solve classification problems using decision trees but has since been applied to several different classification and regression problems using different models as the weak learners. In this paper, two existing versions of AdaBoost—AdaBoost.R2 and AdaBoost.RΔ—are applied to a real-world problem in Precision Agriculture (PA), i.e. yield and protein prediction. A third novel algorithm combining ideas of these two versions is also applied. All three algorithms use single layer Feedforward Neural Networks (FFNN) trained with backpropagation [21] as the weak model. Ensembles of FFNNs are not common in PA, and to our knowledge, AdaBoost has not yet been applied to the problem of yield prediction. Furthermore, protein prediction has not been widely studied in PA in general.

PA focuses on applying new technology to farming to decrease cost, increase net return, and maintain or improve environmental impact. This is not only beneficial to the farmer, but directly impacts food supplies and water quality. There has been substantial work in yield monitoring and fertilizer optimization [13], [19], [20]; however, comprehensive decision-support tools for site-specific crop management are still lacking, especially systems that are adaptable across farms and flexible for farmer use [15]. (See [11] for an overview of decision support systems in Precision Agriculture.) Pedersen and Lind [17] discuss the progress in PA and conclude there is need for better information management and decision support systems, together with better integration of fertilizer plans into these systems. Recently, the importance and value of a well built site-specific decision support tool has become apparent. To this end, a project focusing on on-farm experimentation was initiated in Montana, with the goal of creating an economic-based decision model to increase yield, decrease cost, and improve environmental impact [10], [14], [18].

The goal of our work is to improve yield and protein prediction of winter wheat using both non-spatial and spatial data. We analyze farmer-provided data for specific fields, including previous years' yield and protein, and freely available data on the field and crops (e.g., elevation, slope, topological position index, precipitation, and the normalized difference vegetation index (NDVI)). All of this data is combined and analyzed to create site-specific experiments and to perform economic optimization. The experiments aim to create a nitrogen prescription map. The yield and protein points are discretized into bins, and different nitrogen rates are evenly prescribed across the cells in the field. An example of such a prescription map is shown in Figure 1. The farmers then use these prescription maps to apply nitrogen to their fields and provide back the resulting crop data, creating a spatio-temporal data set.

An economic model is used to optimize net return based on probability distributions of nitrogen cost and crop prices, as well as yield and protein predictions. As net return optimization is the ultimate goal, achieved by minimizing nitrogen costs and optimizing yield, accurate predictions of yield and protein are an essential aspect of the problem. A description of the yield and protein prediction methodology is given in Section V. The entire workflow can be seen in Figure 2.

II. Related Work

A. AdaBoost and Neural Networks

Using neural networks as the weak learner in AdaBoost has become more prevalent over the past years. In [22], Schwenk and Bengio apply AdaBoost to classifier ANNs for character recognition, showing the decrease in error rates for boosted NNs. In later work, Schwenk and Bengio compare three different sampling weight update methods for AdaBoost,

Fig. 1: Example prescription map for field "sec35mid." Nitrogen rates 0, 20, 40, 80 and 120 pounds per acre correspond to red, orange, blue, light green, and dark green respectively.

also using ANNs as the weak model [23]. In resampling, a new sub-sample is chosen randomly from the original data set using the calculated probability distribution of the data points. The first method samples a subset once before training the neural network while not updating the sample set through the training process; the second approach uses a different training set for each epoch; and the third approach directly weighs the cost function. The results indicate that applying AdaBoost has less of an impact on data sets needing larger networks, and all three sampling methods perform similarly.

Liu *et al.* [12] compare four neural network models with different training methods for wind-speed prediction. The four training methods are Gradient Descent and Gradient Descent with Momentum with Adaptive Learning Rate Back Propagation (GD-ALR-BP and GDM-ALR-BP respectively), Conjugate Gradient Back Propagation with Fletcher-Reeves Updates (CG-BP-FR), and the BroydenFletcherGoldfarbShanno (BFGS) algorithm. They apply AdaBoost using the four models as the weak learners and compare the single model results to the ensemble results. Applying AdaBoost improves results for all four training methods, where the CG-BP-FR training method has the best performance overall. In [27] Zhou *et al.* suggest that it could be more beneficial to combine many of the weak models as opposed to all the models. In other words, if 20 neural networks are trained, it could improve results if only 19 out of 20 models are combined for the final predictions, where the excluded model satisfies certain constraints. For this purpose, the authors propose a new ensemble method—GASEN (Genetic Algorithm based Selective ENsemble)—that selects a subset of networks and compares it to bagging and boosting methods. Their results show that ensemble methods outperform a single neural network; however, GASEN does not always perform better than the other ensemble methods.

B. Yield Prediction

Overall, the goal of PA is to improve crop management while maintaining or improving environmental impact. It does this by analyzing specific fields and crops and proposing improvements to the farming process, for example, by specifying fertilizer and pesticide application, providing an optimal irrigation strategy, or developing more efficient machinery. Optimizing fertilizer application is a widely studied problem, as decreasing the amount of nitrogen while maintaining and potentially improving crop profitability is a relatively simple way to lower costs [2]. The optimal application of a fertilizer such as nitrogen not only impacts profit, but influences the environment as well [4].

Artificial neural networks (ANNs) and many of their adaptations have been applied successfully to different PA problems. What follows is a limited overview of the application of ANNs to yield prediction of various crops. Kaul *et al.* [9] use ANNs to determine corn and soybean yield for farms in Maryland under typical climate conditions, specifically precipitation. The results using the ANN were promising compared to multiple linear regression. ANNs were used by Panda *et al.* [16] to look at the influence of including four different spectral indices—NDVI, green vegetation index (GVI), soil adjusted vegetation index (SAVI), and perpendicular vegetation index (PVI)—for corn yield prediction. The authors found that using PVI gives more favorable results than the other indices and confirm the usefulness of using ANNs for yield prediction.

Using image data for yield prediction is also fairly common practice in PA. You *et al.* [25] use remote sensing data, which is available worldwide, to predict soybean crop yield in the U.S. In their research, images are transformed into histograms of pixel counts, effectively performing dimensionality reduction, and are then used as input to Convolutional Neural Networks (CNNs) and Long-Short Term Memory networks (LSTMs) for yield prediction. In addition to these ANNs, a Gaussian process layer is applied to account for spatio-temporal dependencies. Their method outperforms decision trees and a 3-layer neural network for large-scale yield prediction. Raw image data of apple fruit and tree canopies are used in [3] for early yield prediction. The images are split into fruit, foliage, and background through image segmentation techniques and the resulting features—cross-sectional areas of fruit, small fruits, and foliage, as well as number of fruits—provide the input into the FFNN. The FFNN, in combination with the extracted features, was shown to achieve desirable results for early yield prediction of fruit.

In a preliminary paper [18], Stacked Autoencoders (SAE) [8] and FFNNs were applied to a simple spatial representation of the data and compared to a simple non-spatial data set. In general the SAE in combination with a spatial representation performed better than the simple FFNN and non-spatial data. However, the predictive results were not satisfactory for the final goal of the project, thus motivating the current study by using the FFNNs as a weak model in an ensemble method, as this has been a successful technique in other fields.

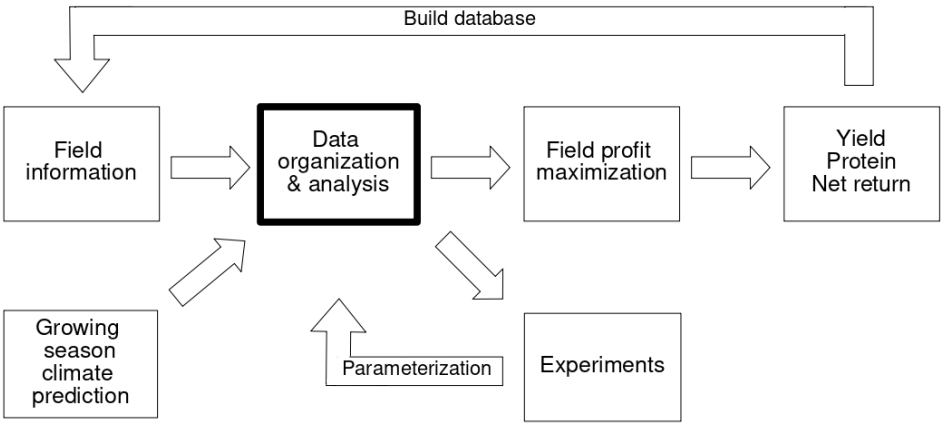


Fig. 2: Flowchart of the optimization process for field profit maximization. The "Data organization and analysis" stage is highlighted to indicate the main focus of this paper.

III. AdaBoost Background

AdaBoost was first introduced by Freund and Schapire in 1996 for binary classification [7]. The basic algorithm takes a data set with N d-dimensional samples, $\{x_i \in X^d\}_{i=1}^N$, and corresponding labels $y_i \in \{0, 1\}$ as input, as well as the number of models T to be learned. The initial weight vector or probability distribution $\mathbf{p}^0 = [\frac{1}{N}, ..., \frac{1}{N}]^\top$ is set to a uniform distribution across all points. For each weak learner, the weight vector is provided to select a subset of data based on the distribution or to assign a weight to each data point as the model is being trained. A hypothesis h_t is returned after training and is then used to calculate the model's loss ϵ_t. If $\epsilon_t > 0.5$ the iteration is aborted; otherwise, the loss is used to update the weights:

$$p_i^{t+1} = p_i^t \beta_t^{1-|h_t(x_i)-y_i|},$$

where $\beta_t = \epsilon_t/(1-\epsilon_t)$, and the distribution is normalized using a normalization factor Z. The final hypothesis is a weighted majority vote of all weak learners' hypotheses.

Basic AdaBoost was improved and adapted for regression problems by the same authors in 1997 [6]. Their first regression adaptation is called AdaBoost.R and reduces the regression problem to binary classification as follows,

$$h^c(x, y') = \begin{cases} 0, & \text{if } y' < y \\ 1, & \text{otherwise} \end{cases},$$

where y' is the predicted value from $h(x)$. The weak models are combined to determine the final hypothesis by calculating the weighted median based on the β value of each model [6].

Drucker proposed a new adaptation for regression called AdaBoost.R2 (Algorithm 1) [5]. The hypothesis is changed to be able to use any loss function as long as $L \in [0, 1]$. The loss for each training sample is calculated using three different candidate loss functions (linear, square and exponential), and the overall model loss is the average of each point's loss L_i times its probability p_i. The weight is then updated as:

$$p_i^{t+1} = p_i^t \beta_t^{1-L_i}.$$

Algorithm 1 AdaBoost.R2

Input:

- Dataset $x_i \in X$, $i = 1, 2, \ldots, N$
- Number of models T

1) Initialize the probability distribution $p_i \in D$ where $p_i = 1/N$, $i = 1, 2, \ldots, N$
2) For $t = 1$ to T:
   a) Fit a weak learner $h_t(X)$ to the training data using weight distribution D.
   b) Compute loss

$$L_t = \frac{\sum_{i=1}^N L_i}{N}.$$

   where,

$$\text{Linear: } L_i = \frac{|h_t(x_i) - y_i)|}{\sup(h_t : x \longrightarrow y)}$$

$$\text{Square: } L_i = \left(\frac{|h_t(x_i) - y_i)|}{\sup(h_t : x \longrightarrow y)}\right)^2$$

$$\text{Exponential: } L_i = 1 - \exp\left(-\frac{|h_t(x_i) - y_i)|}{\sup(h_t : x \longrightarrow y)}\right)$$

   where,

$$h_t : x \longrightarrow y = |h_t(x_i) - y_i| \text{ for } i = 1, ..., N.$$

   c) Compute $\beta_t = L_t/(1 - L_t)$.
   d) Set

$$p_i^{t+1} = \frac{p_i^t}{Z_t} \times \beta_t^{(1-L_i)}$$

   where Z_t is the normalization factor
3) Output:

$$H(x) = \inf\left[y \in Y : \sum_{t:h_t \le y} log(\frac{1}{\beta_t}) \ge \frac{1}{2}\sum_t \log(\frac{1}{\beta_t})\right]$$

The final hypothesis is still obtained by calculating the weighted median of all model hypotheses.

AdaBoost.RT was proposed in 2004 by Solomatine and Shrestha [24] and uses a threshold for prediction accuracy. If the loss or error rate goes above 0.5, the iteration is no longer aborted but continues until the set number of weak learners has been trained. Instead of calculating the loss, AdaBoost.RT calculates the error by summing probability distribution $D_t(i)$ for all points, as long as the error rate ϵ_i is larger than the provided threshold δ. Then

$$\epsilon_i = \left| \frac{h_t(x_i) - y_i}{y_i} \right|.$$

The probability distribution is then updated in the same way as the original AdaBoost for classification but where $\beta_t = \epsilon_t^2$, and a point is "classified" as correct if its error is below or equal to the threshold value. If the difference is above the threshold, the weight remains the same. The final hypothesis is the result of computing the weighted average as opposed to the weighted median.

An obstacle to using AdaBoost.RT is that it requires the threshold value to be chosen correctly for the best performance. Zhang *et al.* [26] developed a method to infer the threshold parameter for each weak learner by looking at the scaled standard deviation of the approximation errors.

A further adaptation known as AdaBoost.RΔ was proposed by Bertoni *et al.* [1]. AdaBoost RΔ also uses a threshold value but creates a binary classification of 0 or 1, multiplies this binary outcome with the corresponding probability p_i, and sums these to obtain the model error ϵ_t. That is,

$$\epsilon_t = \sum_{i=1}^{N} p_i \times HS(||h_t(x_i) - y_i|| - \delta)$$

where,

$$HS(x) = \begin{cases} 1, & \text{if } x \geq 0 \\ 0, & \text{otherwise} \end{cases},$$

and $||\cdot||$ denotes a norm. The full implementation can be found in Algorithm 2.

AdaBoost.RΔ calculates the final hypothesis as

$$H(x) = \underset{y \in [0,1]^m}{\arg\max} \sum_{y} \alpha_t HS(\delta - ||h_t(x_i) - y_i||)$$

where $\alpha_t = \log \frac{1}{\beta_t}$. But for calculating predictions on a test set, this hypothesis calculation is not possible, as the actual value y_i is not known. Therefore, we calculate the weighted average to determine the final hypothesis.

IV. Approximate AdaBoost

Here, we propose a new approach to applying AdaBoost to ANNs for regression, which we call "Approximate AdaBoost" or AdaBoost.App (Algorithm 3). The idea behind AdaBoost.App is to combine the loss function error calculation from AdaBoost.R2 with the threshold approach from AdaBoost.RΔ and AdaBoost.RT. Theoretically, when a data point falls within the threshold, the weight would be set to 0.

Algorithm 2 AdaBoost.RΔ

Input:

- Dataset $x_i \in X$, $i = 1, 2, \ldots, N$
- Number of models T
- Threshold δ

1) Initialize the probability distribution $p_i \in D$ where $p_i = 1/N$, $i = 1, 2, \ldots, N$
2) For $t = 1$ to T:
   a) Fit a weak learner $h_t(X)$ to the training data using probability distribution D.
   b) Compute
   $$\epsilon_t = \sum_{i=1}^{N} p_i \times HS(||h_t(x_i) - y_i|| - \delta)$$
   where,
   $$HS(x) = \begin{cases} 1, & \text{if } x \geq 0 \\ 0, & \text{otherwise} \end{cases}$$
   If
   $$\epsilon_t > 0.5$$
   $T = t - 1$ and abort loop.
   c) Compute $\beta_t = \epsilon_t / (1 - \epsilon_t)$.
   d) Set
   $$p_i^{t+1} = \frac{p_i^t}{Z_t} \times \beta^{1 - HS(||h_t(x_i) - y_i|| - \delta)}$$
   where Z_t is the normalization factor
3) Output:
$$H(x) = \sum_{t=1}^{T} \beta_t h_t(x)$$

The biggest difference lies in the error calculation and weight updates. The error of a single data point ϵ_i is calculated as:

$$\epsilon_i = \begin{cases} |h_t(x_i) - y_i|, & \text{if } |h_t(x_i) - y_i| > \delta \\ 0, & \text{otherwise} \end{cases}.$$

The final errors are normalized using a normalization factor Z. The model error ϵ_t is calculated in the same way as AdaBoost.R2 by summing the errors and dividing by the length, and $\beta_t = \epsilon_t / (1 - \epsilon_t)$. The probabilities p_i are updated based on ϵ_i for $i = 1, 2, \ldots, N$ as follows:

$$p_i^{t+1} = \begin{cases} p_i^t \times \beta^{-\epsilon_i}, & \text{if } \epsilon_i > 0 \\ 0, & \text{otherwise} \end{cases}.$$

While AdaBoost.RT maintains the weight of wrongly classified points and updates the weights of the correctly classified points, AdaBoost.App sets the probability of a correctly classified point to 0 and changes the weight of incorrectly classified points based on how far off they are. The intuition behind this approach is that it is more important to emphasize points that are further away from the target value. Unfortunately, the implemented neural network is unable to train on a sparsely

Algorithm 3 AdaBoost.App

Input:

- Dataset $x_i \in X$, $i = 1, 2, \ldots, N$
- Number of models T
- Threshold δ

1) Initialize the probability distribution $p_i \in D$ where $p_i = 1/N$, $i = 1, 2, \ldots, N$
2) For $t = 1$ to T:
   a) Fit a weak learner $h_t(X)$ to the training data using weight distribution D.
   b) Compute
   $$\epsilon_t = \frac{\sum_{i=1}^{N} \epsilon_i}{N}.$$
   where,
   $$\epsilon_i = \begin{cases} \frac{|h_t(x_i) - y_i|}{Z_t}, & \text{if } |h_t(x_i) - y_i| > \delta \\ 0, & \text{otherwise} \end{cases},$$
   where Z_t is the normalization factor ensuring $\epsilon_i \in [0, 1]$.
   c) Compute $\beta_t = \epsilon_t/(1 - \epsilon_t)$.
   d) Set
   $$p_i^{t+1} = \begin{cases} p_i^t \times \beta^{-\epsilon_i}, & \text{if } \epsilon_i > 0 \\ 0, & \text{otherwise} \end{cases}.$$
3) Output:
$$H(x) = \sum_{t=1}^{T} \beta_t h_t(x)$$

weighted data set, thus the weights are set to 10^{-6} when classified correctly, to minimize their influence effectively. The final model is then a weighted average over the weak models.

V. Methodology

Four different yield and protein data sets were chosen, each representing a different field from a different farmer, totaling eight data sets. An overview of the number of points for each of these data sets is given in Table I. For each yield and protein point, the data included the following features:

- Previous years' nitrogen application
- Field slope
- Field elevation
- Topographic Position Index (TPI)
- Field aspect at the specified point
- Precipitation (in inches of rainfall)
- Normalized Difference Vegetation Index (NDVI)
- Growing Degree Day (GDD) value.

The number of protein points is significantly less for the protein data sets, as the protein monitors mounted on the harvesters were unable to sample at as high of a rate as the yield monitors. Yield is measured in bushels per acre, for winter wheat this means 60 lbs of wheat per bushel. Protein is represented as a percentage of the total grain.

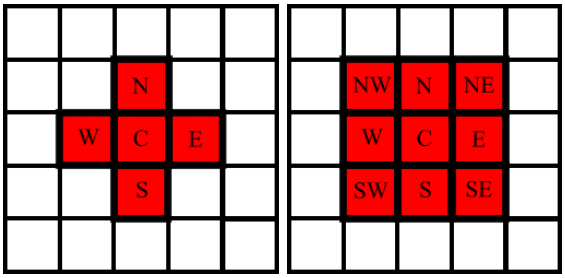


Fig. 3: Neighborhood configurations for cell "C": von Neuman is left, Moore is right for sampling.

TABLE I: Number of data points for each field.

	sec35mid	**sre1314**	**davidson**	**carlin**
yield points	17873	24646	11802	15621
protein points	1019	2998	560	655

The available field data is transformed into a grid structure for the prescription maps (Figure 1); therefore, using this grid structure for spatial sampling is a logical consequence. There are two different spatial sampling strategies that make use of a grid structure: the von Neumann neighborhood (Figure 3 left) and the Moore neighborhood (Figure 3 right). Data points are laid out throughout the field, each data point belonging to a specific grid cell. Each of the neighborhood sampling methods looks at certain neighboring cells of the center cell, to which a data point belongs. The von Neumann neighborhood only looks at the cardinal directions, whereas the Moore neighborhood includes the primary intercardinal directions. To use as much information as possible, the Moore neighborhood was chosen as the spatial sampling method for the results reported here. We chose to only use data points that have data in all eight neighboring cells.

To determine the parameters of the neural networks, a limited grid search was performed with 10 fold cross validation for each parameter combination. Each neural network uses a single hidden layer, as this is sufficient for universal approximation of non-linear functions. Three different parameters were tuned: number of epochs (200, 300, 400, and 500), gradient descent optimizer (Adam and Root Mean Square Propagation (RMSProp)), and numbers of hidden nodes (5, 10, 20, 50, 75, 100). As both Adam and RMSProp adapt the learning rate during training, extensive tuning of the learning rate was less important. We therefore chose to set the learning rate to 0.01. We applied RMSProp over 400 epochs across all data sets. The best number of hidden nodes varied between 5, 10, 50, 75, and 100 as shown in Table II.

Each AdaBoost method was run on each of the data sets with the number of models ranging from 0 to 20. Two examples of such runs are shown in Figure 4. These results indicate that increasing the number of models too far tends to diminish the predictive quality for this specific problem. Based on these results, each AdaBoost method was applied with 5,

TABLE II: Chosen number of hidden nodes (one hidden layer) for each field.

	sec35mid		sre1314		davidson		carlin	
	Yield	**Protein**	**Yield**	**Protein**	**Yield**	**Protein**	**Yield**	**Protein**
Spatial	100	10	10	5	50	5	5	100
Non-Spatial	50	50	10	50	75	5	5	5

10, and 20 weak learners across all data sets. Weight sampling was performed by assigning a weight to each data point as the neural network was being trained, directly influencing the loss function. This method was chosen because it is closest to the original AdaBoost algorithm and was shown by Schwenk and Bengio to perform well given sufficient epochs [23]. The weak learners use mean squared error (MSE) as the loss function for training. The threshold δ was set to 1.5 for AdaBoost.RΔ and AdaBoost.App. Each of the algorithms was run using 10-fold cross validation, and the results for the test predictions were evaluated using the root mean squared error (RMSE) on the unnormalized values. The RMSE values averaged over the ten folds were then compared using a paired t-test to determine whether or not observed differences were statistically significant at a confidence level of 0.05. The results from the t-tests indicated significant differences between spatial and non-spatial results, as well as between different AdaBoost methods that train a different number of weak models. There were also statistical differences between all single neural network results and the AdaBoost results.

VI. Results

Tables III and IV show the average RMSE scores for yield and protein data using the three different AdaBoost implementations for 5,10, and 20 models. Both the spatial and non-spatial data results are given. The best results are carried over into Table V, which compares the simple FFNN results to the AdaBoost results.

In general, the original hypothesis that spatial results will increase performance, as stated in the preliminary paper [18], holds for the yield data. There is one notable exception for sec35mid. As shown in Table III, the spatial AdaBoost model consistently performs worse than the AdaBoost model using the non-spatial data on this field. The spatial analysis can only consider data points that belong to cells that have eight surrounding cells, as each data point has to have the same dimensions to be used by the predictive models. Because the field in question consists of two separated parts (Figure 1), the number of data points reduces more than for other fields, as there are two parts that lose the edge cells on the field (edge cells are the cells on the borders that are not fully surrounded by other cells). This data point reduction could explain why the spatial analysis for this field is not as effective.

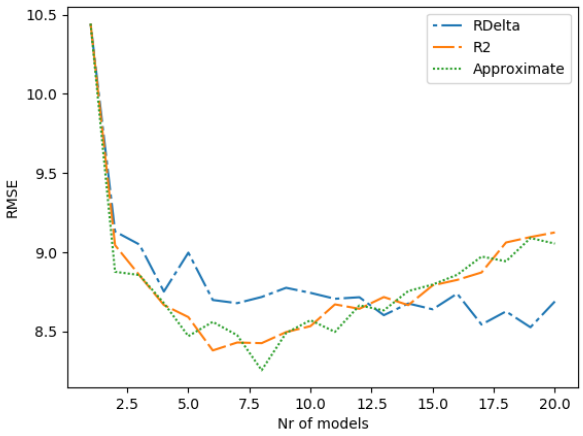
Similarly, for protein data, using spatial data does not always improve results. Table I shows that the number of data points for protein is much smaller than for yield. Certain spatial data sets could therefore suffer from the curse of dimensionality. The number of features becomes much larger when adding in the spatial information, but there may not be enough data to analyze the data properly, thereby negatively influencing predictive power.

Second, note that applying AdaBoost improves over, and in two cases maintains, results from a single FFNN across all fields and data sets (Table V). This confirms our hypothesis that the AdaBoost ensemble method can improve predictive results. For yield, even if the prediction error only drops by one unit, this means that it is 60 pounds of yield per acre closer to the actual value. The results show that there is a drop in error of 3 to 10 units, or 180 to 600 lbs of yield per acre, which is a non-negligible amount. The protein results are improved by smaller numbers, as this indicates the percentage of protein in a single grain, which is inherently a small number.

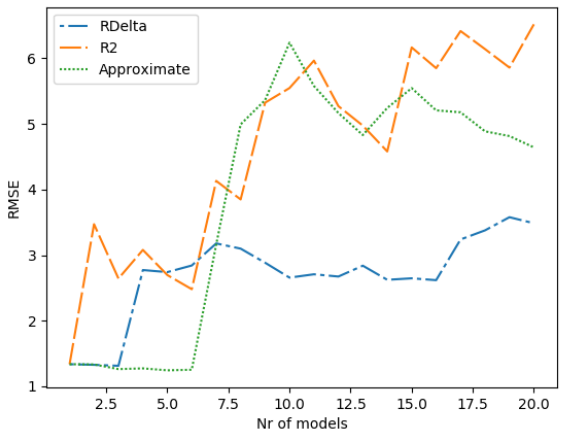
It also seems that combining a smaller number of weak models seems to perform better than increasing the number of models (this can be seen in Tables III and IV and Figure 4). This is in line with the idea that "many may be better than all" [27], where increasing the number of models may tend to increase the error, perhaps through some form of over-fitting. However, instead of training all the models and selecting from those models, simply training fewer models may have a similar effect while needing less time.

When comparing the different AdaBoost methods, no one method seems to outperform the others. It is interesting to note that AdaBoost.R2 and AdaBoost.App both seem to work much better with a smaller number of models, whereas results for AdaBoost.RΔ vary across the number of models (Figure 4). This may be due to the way the model error ϵ_t is calculated and the individual points' weights are updated for AdaBoost.RΔ. As the probability distribution of incorrectly classified data points is updated based on their previous probability, the distribution may not be changed substantially. This could have resulted in relatively small differences between models' weights. Then, even if later models have a larger influence (i.e., a heavier weight) on the end result, the difference in model weight may be small enough such that initial models, which should be correctly predicting easier data points, still play an important role in calculating the final values.

Finally, note when comparing the different implementations that the similarity in predictions between AdaBoost.R2 and AdaBoost.App, especially for yield predictions. We believe this may be due to improper tuning of the threshold value for AdaBoost.App, resulting in only a small number of data points falling below that threshold, effectively resulting in very similar predictions for both models. When looking at figure 4b, there is a clear difference in the performance of the two models. This indicates that the threshold was more appropriately set for the protein data for AdaBoost.App.



(a) Field sec35mid non-spatial yield prediction.



(b) Field carlinwest non-spatial protein prediction.

Fig. 4: RMSE values for two of the fields studied for the three different AdaBoost methods.

TABLE III: Yield prediction RMSE for the AdaBoost methods, each training 5,10, and 20 models, across all fields using spatial and non-spatial data.Results in *italics* are carried over into Table V. **Bolded** results are the lowest score for that field.

		AdaBoost.RΔ			AdaBoost.R2			AdaBoost.App		
		5	**10**	**20**	**5**	**10**	**20**	**5**	**10**	**20**
sre1314	**spatial**	9.674	9.546	***9.508***	9.828	11.095	11.923	9.920	11.018	11.938
	non-spatial	10.478	10.509	10.489	10.469	10.668	11.031	*10.462*	10.737	10.989
sec35mid	**spatial**	19.998	21.462	21.898	20.241	13.840	10.209	19.722	13.898	*10.092*
	non-spatial	8.998	8.657	8.612	8.591	8.545	9.077	8.471	***8.388***	9.026
davidson	**spatial**	8.008	7.929	7.889	***7.815***	8.140	8.480	7.821	7.990	8.467
	non-spatial	14.049	13.766	13.778	13.433	13.865	14.573	*13.420*	13.906	14.590
carlin	**spatial**	***5.425***	5.467	5.652	5.638	6.779	7.885	5.691	6.795	8.436
	non-spatial	7.893	6.850	6.877	*6.437*	6.596	7.570	6.625	6.781	7.576

TABLE IV: Protein prediction RMSE for the AdaBoost methods, each training 5,10, and 20 models, across all fields using spatial and non-spatial data. Results in *italics* are carried over into Table V. **Bolded** results are the lowest score for that field.

		AdaBoost.RΔ			AdaBoost.R2			AdaBoost.App		
		5	**10**	**20**	**5**	**10**	**20**	**5**	**10**	**20**
sre1314	**spatial**	0.981	0.977	0.994	0.956	***0.934***	0.974	0.970	0.954	0.974
	non-spatial	1.019	0.994	0.981	0.978	*0.971*	0.985	0.983	0.992	1.132
sec35mid	**spatial**	2.185	1.920	*1.850*	2.211	2.199	2.210	2.171	2.119	2.356
	non-spatial	1.636	1.602	***1.320***	1.609	1.494	1.377	1.498	1.412	1.406
davidson	**spatial**	1.706	1.720	1.849	1.709	1.737	1.721	1.746	1.701	*1.680*
	non-spatial	1.551	***1.527***	1.926	1.594	1.627	1.754	1.609	1.686	1.688
carlin	**spatial**	1.739	1.676	1.891	1.852	1.599	*1.454*	1.730	1.527	1.426
	non-spatial	2.741	2.657	3.485	2.696	5.545	6.516	***1.244***	6.236	4.641

TABLE V: Yield and protein prediction RMSE for a single FFNN, next to the best AdaBoost result from Tables III and IV.

		sre1314		sec35mid		davidson		carlin	
		Yield	**Protein**	**Yield**	**Protein**	**Yield**	**Protein**	**Yield**	**Protein**
Spatial	**FFNN**	12.546	1.184	14.606	1.964	17.735	1.641	9.712	1.307
	AdaBoost	9.508	0.934	10.092	1.850	7.815	1.680	5.425	1.454
Non-spatial	**FFNN**	12.141	1.187	14.916	2.320	22.836	6.945	10.607	7.452
	AdaBoost	10.489	0.971	8.388	1.320	13.766	1.527	6.596	1.244

VII. Conclusion and Future Work

Finding a way to improve nitrogen rate application for specific crops and fields is an important challenge in the field of Precision Agriculture. Yield and protein prediction can help this process by providing information for subsequent crop fertilization. Previous research in this field has mainly used simple models and has not looked at a boosted ensemble method. This paper compared a simple single layer FFNN to three different AdaBoost methods, each using single layer FFNNs as the weak learner, and used both spatial and non-spatial data to train these models. Our results confirm that spatial data tends to improve predictions, but that the amount

of available data can negatively influence performance. Consistent with findings from other studies, ensemble methods can also improve predictions compared to the simple model. In our case, yield predictions improved by between 3 to 10 units, meaning the error was reduced by 180 to 600 lbs of yield per acre. When comparing results from different AdaBoost runs, the three implementations performed similarly. AdaBoost.RΔ seemed to handle an increase in the number of weak learners to be combined better than the other two implementations.

In terms of the novel AdaBoost.App method, the results indicated the importance of correctly setting the threshold value when calculating the error. To this end, we would like to research ways to automatically set the threshold value. Aside from implementing the method proposed by [26], devising a method using the values of bias nodes in a neural network might be a good way to determine the magnitude of the threshold value. Creating an automatic threshold calculation avoids the addition of an extra tuning parameter and could improve overall predictive power of the model. In order to properly evaluate the performance of the model compared to other AdaBoost implementations, a study using benchmark data as well as the PA data would have to be performed.

For future work, we plan to incorporate the results obtained through the prediction process into the economic model for actual profit optimization (Figure 2), as the work here is only a small part of a larger decision model. For future research directions, we would like to look at random forests (RF), as these are widely applied for yield prediction. Implementing an RF and adapting the random forest structure to neural networks could further improve results. Furthermore, the lack of data points for some of the spatial analysis, could be addressed by performing other forms of spatial analysis such as kriging.

ACKNOWLEDGMENTS

We thank the OFPE team for providing the site-specific data and the Numerical Intelligence Systems Laboratory at MSU for help with the idiosyncrasies of running GPU-based experiments, as well as comments arising from discussions with our lab mates. This project was funded, in part, by MREDI grant 51040-MUSR12015-02 and by NSF grant 1658971.

REFERENCES

[1] Alberto Bertoni, Paola Campadelli, and M Parodi. A boosting algorithm for regression. In *International Conference on Artificial Neural Networks*, pages 343–348. Springer, 1997.

[2] R. Bongiovanni and J. Lowenberg-Deboer. Precision agriculture and sustainability. *Precision Agriculture*, 5(4):359–387, Aug 2004.

[3] Hong Cheng, Lutz Damerow, Yurui Sun, and Michael Blanke. Early yield prediction using image analysis of apple fruit and tree canopy features with neural networks. *Journal of Imaging*, 3(1):6, 2017.

[4] Zhenling Cui, Fusuo Zhang, Xinping Chen, Zhengxia Dou, and Junliang Li. In-season nitrogen management strategy for winter wheat: Maximizing yields, minimizing environmental impact in an over-fertilization context. *Field Crops Research*, 116(1):140 – 146, 2010.

[5] Harris Drucker. Improving regressors using boosting techniques. In *ICML*, volume 97, pages 107–115, 1997.

[6] Yoav Freund and Robert E Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, 55(1):119–139, 1997.

[7] Yoav Freund, Robert E Schapire, et al. Experiments with a new boosting algorithm. In *Icml*, volume 96, pages 148–156. Citeseer, 1996.

[8] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2017.

[9] Monisha Kaul, Robert L Hill, and Charles Walthall. Artificial neural networks for corn and soybean yield prediction. *Agricultural Systems*, 85(1):1–18, 2005.

[10] Patrick G Lawrence, Lisa J Rew, and Bruce D Maxwell. A probabilistic bayesian framework for progressively updating site-specific recommendations. *Precision Agriculture*, 16(3):275–296, 2015.

[11] Jessica Lindblom, Christina Lundström, Magnus Ljung, and Anders Jonsson. Promoting sustainable intensification in precision agriculture: review of decision support systems development and strategies. *Precision Agriculture*, 18(3):309–331, Jun 2017.

[12] Hui Liu, Hong-qi Tian, Yan-fei Li, and Lei Zhang. Comparison of four adaboost algorithm based artificial neural networks in wind speed predictions. *Energy Conversion and Management*, 92:67–81, 2015.

[13] EV Lukina, KW Freeman, KJ Wynn, WE Thomason, RW Mullen, ML Stone, JB Solie, AR Klatt, GV Johnson, RL Elliott, et al. Nitrogen fertilization optimization algorithm based on in-season estimates of yield and plant nitrogen uptake. *Journal of plant nutrition*, 24(6):885–898, 2001.

[14] Bruce Maxwell, Paul Hegedus, Philip Davis, Anton Bekkerman, Robert Payn, John Sheppard, Nicholas Silverman, and Clemente Izurieta. Can optimization associated with on-farm experimentation using on-farm experimentation using site-specific technologies improve producer management decisions. In *International Conference on Precision Agriculture (ICPA)*, 2018.

[15] Alex McBratney, Brett Whelan, Tihomir Ancev, and Johan Bouma. Future directions of precision agriculture. *Precision Agriculture*, 6(1):7–23, Feb 2005.

[16] Sudhanshu Sekhar Panda, Daniel P Ames, and Suranjan Panigrahi. Application of vegetation indices for agricultural crop yield prediction using neural network techniques. *Remote Sensing*, 2(3):673–696, 2010.

[17] Søren Marcus Pedersen and KM Lind. Precision agriculture–from mapping to site-specific application. In *Precision Agriculture: Technology and Economic Perspectives*, pages 1–20. Springer, 2017.

[18] Amy Peerlinck, John Sheppard, and Bruce Maxwell. Using deep learning in yield and protein prediction of winter wheat based on fertilization prescriptions in precision agriculture. In *International Conference on Precision Agriculture (ICPA)*, 2018.

[19] Anup K Prasad, Lim Chai, Ramesh P Singh, and Menas Kafatos. Crop yield estimation model for iowa using remote sensing and surface parameters. *International Journal of Applied Earth Observation and Geoinformation*, 8(1):26–33, 2006.

[20] William R Raun, John B Solie, Gordon V Johnson, Marvin L Stone, Robert W Mullen, Kyle W Freeman, Wade E Thomason, and Erna V Lukina. Improving nitrogen use efficiency in cereal grain production with optical sensing and variable rate application. *Agronomy Journal*, 94(4):815–820, 2002.

[21] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 322:533–536, October 1986.

[22] Holger Schwenk and Yoshua Bengio. Training methods for adaptive boosting of neural networks. In *Advances in neural information processing systems*, pages 647–653, 1998.

[23] Holger Schwenk and Yoshua Bengio. Boosting neural networks. *Neural computation*, 12(8):1869–1887, 2000.

[24] Dimitri P Solomatine and Durga L Shrestha. Adaboost. rt: a boosting algorithm for regression problems. *Neural Networks*, 2:1163–1168, 2004.

[25] Jiaxuan You, Xiaocheng Li, Melvin Low, David Lobell, and Stefano Ermon. Deep gaussian process for crop yield prediction based on remote sensing data. In *AAAI*, pages 4559–4566, 2017.

[26] Peng-Bo Zhang and Zhi-Xin Yang. A novel adaboost framework with robust threshold and structural optimization. *IEEE transactions on cybernetics*, 2016.

[27] Zhi-Hua Zhou, Jianxin Wu, and Wei Tang. Ensembling neural networks: many could be better than all. *Artificial intelligence*, 137(1-2):239–263, 2002.