A bandit algorithm for efficient on-farm research

Yuji Saikai and Paul D. Mitchell

Department of Agricultural and Applied Economics
University of Wisconsin-Madison

Abstract

Field experiment is the foundation for gaining agronomic insight into causal relationship between inputs and outputs and, based on them, optimizing cropping systems. When facilitated by precision agriculture, individual farmers may benefit from experimenting with new inputs and practices on their own heterogeneous fields. However, on-farm experiment is different from dedicated research activities in universities or corporations as it is often constrained by limited resources, requiring even higher efficiency, and needs to be incorporated into day-to-day operations, making a trade-off between exploration and exploitation. To address it, we develop an algorithm based on suitable machine learning techniques: active learning, multi-armed bandit, and Thompson sampling. Then, we demonstrate its advantage against the conventional practice using simulation built on the existing field trial data. Our algorithm uniformly outperforms it, and the efficiency gain could be substantial. In addition, the generality of our algorithm makes itself applicable to a wide range of similar problems.

Keywords: *Machine learning, Active learning, Multi-armed bandit, Thompson sampling, Precision agriculture, On-farm experiment*

1 Introduction

Big data has become increasingly visible in agriculture, being analyzed and utilized in many applications (Kamilaris et al., 2017). In addition, each field-level data can be pooled and aggregated to a large agricultural dataset, which has economic and policy implications in society at large (Coble et al., 2018; Coble et al., 2016). Practice collectively named precision agriculture is a main vehicle for mechanically collecting data from individual fields with modern sensor technologies and processing algorithms such as soil sampling (Huuskonen et al., 2018), computer vision (Rehman et al., 2019; Patrício et al., 2018), and thermal images (Khanal et al., 2017). In making use of this large amount of environmental data, domain knowledge certainly plays vital roles, providing intuitive understanding of the relationships between variables, interpretation of the data, and insight into fruitful research directions. However, due to its large size and high dimensionality, subtle yet crucial relationships between variables may not be straightforward and overall systems behavior can be very complex. Machine learning, also known as statistical pattern recognition (Hastie et al., 2016; Bishop, 2006), could help to discover hidden, surprising associations between many variables (supervised and unsupervised learning) and, based on them, enable systems to optimize some of the control variables (reinforcement learning).

An empirical goal is to identify the causal relationship between key inputs such as fertilizer, seed, or irrigation water and outputs such as crop yield, crop quality and/or profit. Given this production relationship, the system can be optimized, and choices implemented using precision agriculture. Traditional approaches to identifying production relationships include experiments with randomization and replication or combining observational data with structural or behavioral models (e.g. estimating supply and demand via two-stage least square or using primal or dual approaches) or panel data methods (e.g. instrumental variables and fixed effects models). Typically, production scientists rely on small plot experiments, while economists use the latter two econometric approaches. Design, implementation, and data collection for small plot experiments are relatively costly, so they only focus on one or two inputs at a time, resulting in difficulty in generalizing to other seasons and locations. On the other hand, assembling the observational data and the required econometric analysis are beyond the capacity of the vast majority of farmers, and the analytical results are usually at such an aggregate level so as to not be relevant for sub-field level management decisions.

Machine learning is commonly mentioned as the solution for automating the analysis of large data sets to quickly provide actionable recommendations to farmers (Chlingaryan et al., 2018). It is a broad term covering many types of algorithms, but at this time, their application to agricultural production contexts for decision making still remains rudimentary (Mishra et al., 2016; Shine et al., 2018; Sun et al., 2017). Athey (2017) highlighted the pervasive problem that machine learning algorithms have with identifying causal relationship from observational data. However, farms can also purposely vary inputs from currently optimal levels as an experimental approach but are typically not interested in experiments with full randomization and replication over a wide range of levels for key inputs. Wide adoption would require an automated experimentation and analysis process that did not imply high losses from substantial over use or under use of key inputs. Linking machine learning with precision agriculture offers a potential solution for lower cost and more widely used on-farm experiments to optimize crop management.

Farmers commonly experiment informally with new technologies, production techniques, crop varieties and other inputs, in essence, trialing them on small portions of their fields to see if improvements result (Marra et al., 2003; Griffin, 2010). The nature of on-farm research is that farmers must balance between learning the value of new or alternative practices that will potentially improve income in future seasons and earning income in the current season by following the standard production practices. This conceptualization of the on-farm research problem fits nicely in the classical trade-off in reinforcement learning of exploration versus exploitation. Reinforcement learning is a subfield of machine learning with the objective of learning, through trials and errors, a policy that prescribes an optimal action to each environment (Sutton et al., 2018). Hence, we look to reinforcement learning for insights on algorithms to automate and to increase the efficiency of the on-farm research process, as farmers would be more willing to adopt an easy-to-use and low-cost process that generated a positive return on investment.

In addition to the exploration-exploitation trade-off, the cost efficiency aspect of on-farm experiment is well captured by a class of machine learning problems called active learning. It concerns sampling or a line of query to ask the environment in order to reach an inferential conclusion as efficiently as possible (Settles, 2012). When machine learning tasks are regression, as opposed to classification, active learning is closely related to optimal experimental design in statistics (Montgomery, 2017). To illustrate the essence of active learning and relate it to the farmer's problem, consider the following example along with Figure 1. Suppose that a farmer wants to estimate the production relationship between seeding rate and yield, which is captured by the curve in the figure. To efficiently pin it down, we want data points to be as close to the curve as possible. Since we do not know the exact shape and location of the function, the closeness is random and technically

we use the tendency of closeness, which is captured by conditional variance of y given x. Suppose that the farmer is concerned about only five seeding rates, $x \in \{40, 60, 80, 100, 120\}$. Then, for each x, it is sufficient for good estimation to have the closest point to the curve (indicated by $\circ$). We cast this active learning problem as a multi-armed bandit and use metadata to search for parts of the fields where conditional variance of y is small. Similar applications of multi-armed bandit for active learning are found in the literature (Ganti et al., 2013; Bouneffouf et al., 2014; Gutiérrez et al., 2017).

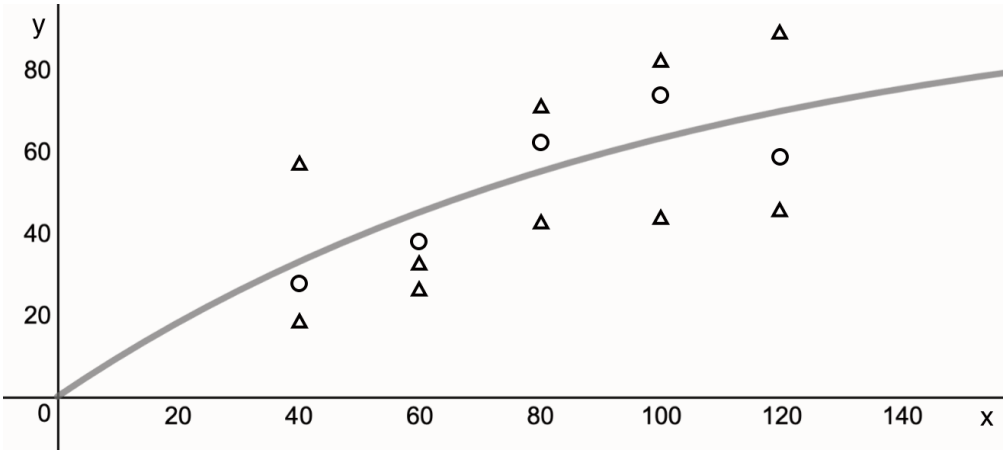

Figure 1: Illustration of active learning. Five circles are sufficient for good estimation of the curve.

In this paper, we develop a general machine learning algorithm for efficiently conducting on-farm experiment when site-specific information is available. It is for on-farm as we address the exploration-exploitation dilemma; it is for efficiency as we address the active learning aspect; and it is for generality as applicable to many other choice problems with the same structure. We conceptualize the on-farm experimentation problem in terms of the multi-armed bandit, a type of reinforcement learning, and build a solution algorithm based on Thompson sampling. Then, we test the algorithm in the context of experimental plot selection where a farmer determines which area of the fields to use for experimentation of seeding rate and which areas to use for typical crop production. Results demonstrate that plot selection for on-farm experiment guided by our algorithm can more efficiently estimate the yield response function than randomly placing plots in fields (the generally recommended method). To reiterate, what we develop in this paper is an algorithm, instead of estimation of a particular function, and therefore it can be used in many other situations with the same problem structure. Nevertheless, some technical issues still remain to be solved, and several interesting economic problems exist to address. We conclude by discussing some of the next steps needed for practical applications of our work.

2 Review

2.1 Multi-armed bandit

The multi-armed bandit is probably the simplest model to capture the exploration-exploitation dilemma (Berry et al., 1985). A K-armed bandit problem imagines a slot machine to play with K arms. The player pulls one arm at a time and receives a reward (win or lose). Over T number of plays, the player maximizes the sum total of expected rewards. As a slot machine, rewards are

random, so each arm $i \in \{1, \ldots, K\}$ defines a stochastic process, an i.i.d. sequence of Bernoulli random variables $X_{i,1}, X_{i,2}, \ldots$ with an unknown success probability μ_i. This situation requires a balance between exploring the arms to find those with higher success probabilities and exploiting the arm with the highest estimate $\hat{\mu}_i$ at any given round. A number of algorithms have been proposed and theoretically analyzed to solve this class of problems (Bubeck et al., 2012).

2.2 Thompson sampling

Since each μ_i is unknown, from a Bayesian point of view, we may treat it as a random variable. Let $\boldsymbol{\mu}$ be a vector of $\{\mu_i\}$, i.e. $\boldsymbol{\mu} = (\mu_1, \ldots, \mu_K)^T$, and $p(\boldsymbol{\mu})$ be a prior distribution of $\boldsymbol{\mu}$. Also, let D_t be the available data after t rounds. Then, by Bayes' rule, a posterior distribution of $\boldsymbol{\mu}$ is

$$p(\boldsymbol{\mu}|D_t) = \frac{p(D_t|\boldsymbol{\mu})p(\boldsymbol{\mu})}{p(D_t)}.$$

When the data D_t is generated through K-armed bandit with binary rewards and there is no prior knowledge to incorporate, a natural choice for the prior distribution $p(\boldsymbol{\mu})$ is a product of K identical beta distributions

$$p(\boldsymbol{\mu}) = \prod_{i=1}^{K} \text{beta}(\mu_i; a, b),$$

where a and b are shape parameters for the beta distribution. In particular, $a = b = 1$ corresponds the case of a product of the uniform distributions over $[0, 1]$, which may be appropriate when we have neither knowledge about any particular arm nor preference across the arms. Since beta distribution is a conjugate prior for the Bernoulli likelihood, this choice allows for efficient posterior updating. To illustrate the updating, let $n_{i,1}$ and $n_{i,0}$ denote respectively the number of successes (1s) and the number of failures (0s) from arm i over t rounds, where we must have

$$t = \sum_{i=1}^{K} (n_{i,1} + n_{i,0}).$$

Then, the posterior distribution of $\boldsymbol{\mu}$ is also a product of betas (Russo et al., 2018):

$$p(\boldsymbol{\mu}|D_t) = \prod_{i=1}^{K} \text{beta}(\mu_i; a + n_{i,1}, b + n_{i,0}).$$

As a solution method for the multi-armed bandit problem, Thompson sampling draws a sample $\hat{\boldsymbol{\mu}} = \{\hat{\mu}_1, \ldots, \hat{\mu}_K\}$ from this posterior and chooses an arm that corresponds to the maximum component of $\hat{\boldsymbol{\mu}}$ for the next round. That is, it prescribes as the next arm j such that

$$j = \arg\max_i \{\hat{\mu}_1, \ldots, \hat{\mu}_K\}.$$

Thompson sampling belongs to a class of optimization techniques called Bayesian optimization, which is known for efficient balancing between exploration and exploitation (Shahriari et al., 2016).

3 Materials and Methods

3.1 Plot selection as a multi-armed bandit problem

The following example adapts the multi-armed bandit problem as a model for a farmer choosing where to place experimental plots in the fields for on-farm research. Suppose a farmer has precision

agricultural equipment that allows control of inputs and measurement of yield, which together define a minimal management unit. For example, a farmer may have an 8-row planter with variable rate seeding and an 8-row combine that can measure yield along 40 feet of row, so that the management unit is 8 rows by 40 feet. Conceptually, the farmer manages many fields and each field is composed of many such management units and each of these units is potentially a small plot for conducting experiments for on-farm research. However, the farmer is primarily interested in earning income from growing a crop and experimentation is costly, both directly and in terms of opportunity costs from non-optimal input use or management.

Suppose a farmer wants to better estimate the soybean yield response to the seeding rate on the land he operates and for the varieties he/she plants. Thus, he/she wants to choose T management units as small plots within the fields of many such plots for experiment in which the seeding rate is varied around the recommended level and the corresponding yield is measured. Each field is not homogeneous, with soil characteristics such as soil type, organic matter, slope, depth to bedrock, and water-holding capacity varying across the fields. The farmer must decide how to use this soil characteristic information to decide where to place each of these plots in the fields.

Suppose that there are M soil characteristics or features found across the fields. Let $\boldsymbol{z}$ denote a M-tuple, $\boldsymbol{z} \in Z_1 \times Z_2 \times \cdots \times Z_M$ where Z_j is a set for jth feature for all $j \in \{1, \ldots, M\}$. Each coordinate takes a finite number of possible values denoted by $|Z_j|$. Then, there are in total $K = \sum_j |Z_j|$ feature values, each of which identifies the union of disjoint plots with that value. We conjecture that some areas characterized by particular values of soil characteristics provide better areas in which to place small plots for the purpose of estimating the yield response to the seeding rate. For example, Z_1 is a set for soil organic matter content and takes such values as less than 3%, 3% to 3.5%, 3.5% to 4%, and more than 4%. In this case, Z_1 partitions the fields into four areas. Similarly, Z_2 is a set for soil pH and takes such values as less than 6.5, 6.5 to 6.7, 6.8 to 7.0, 7.1 to 7.3 and more than 7.3, partitioning the field into five areas. In this case, with these Z_1 and Z_2 features, we have 9 (overlapping) areas. We can regard each area defined by these characteristics as an arm of the bandit. The farmer chooses an area (arm) from K possible alternatives and sets a plot in the area, conducts the experiment and observes yield (sampling), and then determines whether or not the sample helps estimation of the yield response function (win or lose). Based on this correspondence, plot selection is formulated as a K-armed bandit problem.

3.2 Design of experiment

The objective of this experiment is to compare the performance of our algorithm against that of random sampling, which is the norm in practice. We use the dataset collected by Gaspar et al. (2015). These data are from replicated and randomized field trials conducted at nine locations throughout Wisconsin in 2012 and 2013. They varied the seeding rate and measured yield, generating 1,148 observations in total. Soil pH was among the metadata collected for each location, with eleven unique values. Based on this information, we define variable $\boldsymbol{z}$ by location (Z_1) and soil pH (Z_2). Thus, the farmer sequentially chooses T experimental plots, each of which corresponds to one of 19 feature values, i.e. 9 locations and 10 soil pH values from the following sets:

$$\begin{aligned} \text{location} &= \{\text{Arlington, Chippewa Falls, Fond du Lac, Galesville, Hancock, Janesville,} \\ &\quad \text{Lancaster, Marshfield, Seymour}\} \\ \text{pH} &= \{6.0, 6.2, 6.3, 6.5, 6.6, 6.8, 6.9, 7.0, 7.2, 7.5\} \end{aligned}$$

As with Gaspar et al. (2015), we use a negative-exponential function to model the soybean yield

response (y) to the seeding rate (x):

$$y = \alpha(1 - e^{-\beta x}) + \varepsilon,$$

where α is asymptotic yield maximum, β is responsiveness, and ε is zero-mean noise.

Using the same subset of the original dataset, the performance of each algorithm is assessed by the root mean squared error (RMSE), which is a measure of how well each algorithm performed in terms of estimating the yield response function. Specifically, each algorithm sequentially suggests a sampling plot T times in total. At each round $t \leq T$, the simulation environment returns a random sample (x_t, y_t) from the suggested plot. After T rounds, each algorithm estimates parameters α and β using the nonlinear least squares with the collected sample set $\{(x_1, y_1), \ldots, (x_T, y_T)\}$, and finally reports the corresponding RMSE.

The subset, called training set, from which to take samples is generated by randomly pick 574 observations (50%) from the original dataset of 1,148 observations. Also, random 344 observations (30%) are used to compute the performance metric. Note that this is an out-of-sample test, using a separate subset from the one for estimation. To eliminate chance outcomes, we iterate the above experiment 10,000 times, and the final performance metric is the average RMSE over these iterations. Furthermore, to investigate the effect of sample size T, we repeat the process with varying T, from 5 to 75 with an increment of 2. As a result, the grand total number of experiment is $10,000 \times 36 = 360,000$. Note that we use a unique random seed for each of $10,000$ iterations so that Thompson sampling and random sampling compete on the same ground, and the same seed is used for all T within an iteration. With this treatment, although T is an exogenous parameter that defines the size of (independent) experiment, we can consistently investigate the effect of T as if experiments were seamlessly conducted in a given environment. In addition, this also guarantees reproducibility of the results. Appendix provides the full code written in Python.

3.3 Solution algorithm

While multi-armed bandit problems can be addressed in many ways such as standard UCB algorithms (Auer et al., 2002), we adopt Thompson sampling as our strategy to efficiently search for good areas to place experimental plots characterized by particular location or soil pH. Farmers almost always have some prior knowledge or belief about the relationship they are investigating e.g. information obtained from universities, neighbors, and their own experience of similar inputs. Therefore, we think that Thompson sampling as a Bayesian approach is more suitable for the problem, enabling farmers to incorporate prior knowledge from similar situations.

The algorithm takes the total number of observations T as an exogenous parameter as well as the predefined "field areas", a set of unique areas, each of which is characterized by location or soil pH (Line 1 of Algorithm 1). It makes use of the remaining 230 observations (20%) as a validation set. To initialize the process, the sample set S is emptied; the number of bandit arms K is set equal to the number of areas; the area set is represented as numbers 1 to K; the prior beta distributions are all identical with parameters $a = b = 1$; and finally two observations are randomly drawn from the training (Line 2). For each round t, each field area is assigned a probability that locating an experimental plot in that area will give a sample point that will improve the estimation of the parameters α and β (Line 6). These probabilities are based on the number of times over the previous $t-1$ rounds that placing a plot in that field area improved and worsened estimation (a_j, b_j, Lines 12-13). The field area with the largest probability of improvement is identified (Line 7), and an observation from that field area is added to the sample set (Lines 8-9). Only for the chosen area, estimates of the parameters $\hat{\alpha}_t$ and $\hat{\beta}_t$ are updated, and then the reward r_t is calculated (Lines 10-11).

Algorithm 1 Thompson sampling for selecting experimental plots

Require: T, field areas
Initialize: $S \leftarrow \{\}; K \leftarrow |\text{field areas}|; A \leftarrow \{1, \ldots, K\}; a_i \leftarrow 1, b_i \leftarrow 1 (\forall i \in A)$;
    Add two random samples to S
for $t \in \{3, 4, \ldots, T\}$ **do**
    if any field area is empty **then** remove its index from A
    for $i \in A$ **do**
        Draw $\hat{q}_i$ from Beta(a_i, b_i)
    $j \leftarrow \text{argmax}_i \ \hat{q}_i$
    Sample (x_t, y_t) from j-th field area
    Add (x_t, y_t) to S and remove it from all field areas
    Estimate $(\hat{\alpha}_t, \hat{\beta}_t)$ based on S
    Compute reward r_t based on prediction $\hat{y}_t = \hat{\alpha}_t(1 - e^{-\hat{\beta}_t x_t})$
    if $r_t = 1$ **then** $a_j \leftarrow a_j + 1$
    else $b_j \leftarrow b_j + 1$
return $(\hat{\alpha}_T, \hat{\beta}_T)$

The success of our algorithm (and reinforcement learning in general) critically depends on how we define the reward, as it signals the value of each arm and balances exploration and exploitation accordingly (Sutton et al., 2018). The current application focuses on estimating the yield response function and uses RMSE to determine its performance. Therefore, we use the following reward scheme: the binary reward for each round t is 1 if placing a plot in that field area and adding an observation to the sample set decreased RMSE (success) and -1 if it increased RMSE (failure). Note that RMSE for each round t is calculated using the validation set, not the training set used for the estimation or the testing set used for calculating RMSE after the algorithm ends.

4 Results

Figure 1 plots the average RMSE for both algorithms for estimating the soybean yield response to seeding rate as the number of observations (T) varies from 5 to 75. Not surprisingly, the relative performance of both algorithms is poor when $T = 5$, as they are both estimating two parameters with five observations. However, as the number of observations used increases, the average RMSE declines and gradually approaches the limit of about 20 bushels per acre. Our algorithm outperforms random selection uniformly over the entire range of observations used. The performance difference between the algorithms is small at low values of T and again at high values. Furthermore, our algorithm stabilizes at the average RMSE of 20 at about $T = 30$, while the standard randomization method does not stabilize until about $T = 60$ or even more. Finally, the advantage of our algorithm shrinks as the number of observations increases because with large T two sets of samples are more or less similar whether they are collected randomly or in a sophisticated fashion.

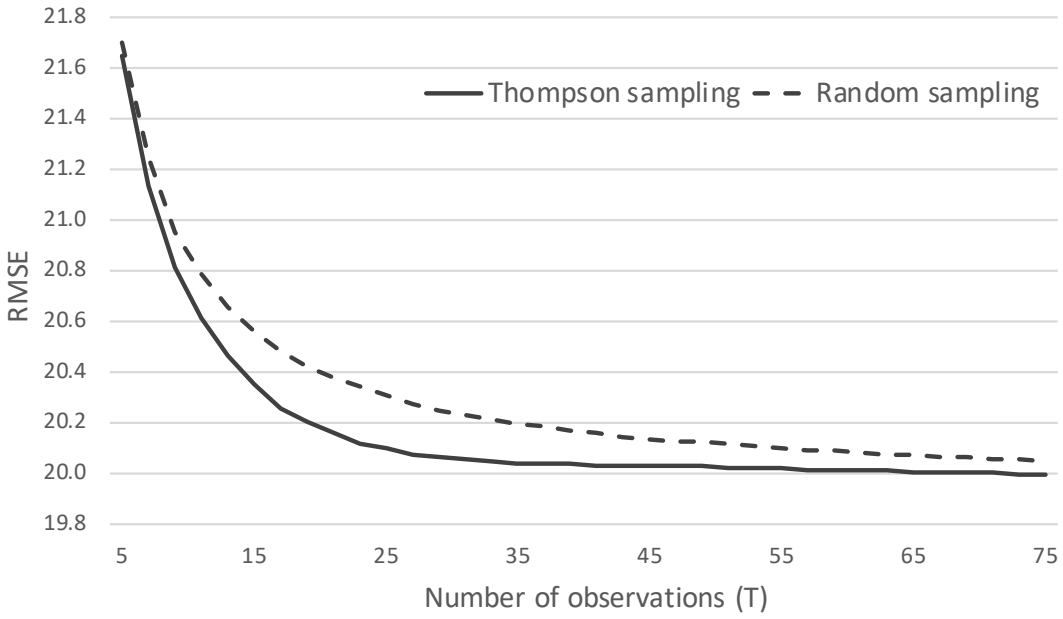

Figure 2: Average root mean squared error for estimating the soybean yield response to the seeding rate using our algorithm versus random selection field plot observations, plotted against a range of the number of observations used

The following bar charts present snapshots of how our algorithm made use of z values. It shows the evolution of learning which z is useful for the objective – efficient estimation of the yield response function. As frequency counts, the bar heights in each chart sum to T (technically, $T-2$ because of two initialization samples). Recall that, by definition, random sampling would result in on average a completely flat bar chart for all T. At $T = 21$, the algorithm tends to explore more than to exploit as indicated by its relative flatness. This makes sense due to the fact that $T = 21$ is relatively small given the total number of feature values $K = 19$ and the identical prior distribution for all features. Nonetheless, we see the same shape or pattern between two charts. At $T = 75$, in contrast, the choices are more pronounced. For example, pH=6 area is clearly preferred to pH=6.3 area, and Lancaster is the most useful location for the purpose of estimation.

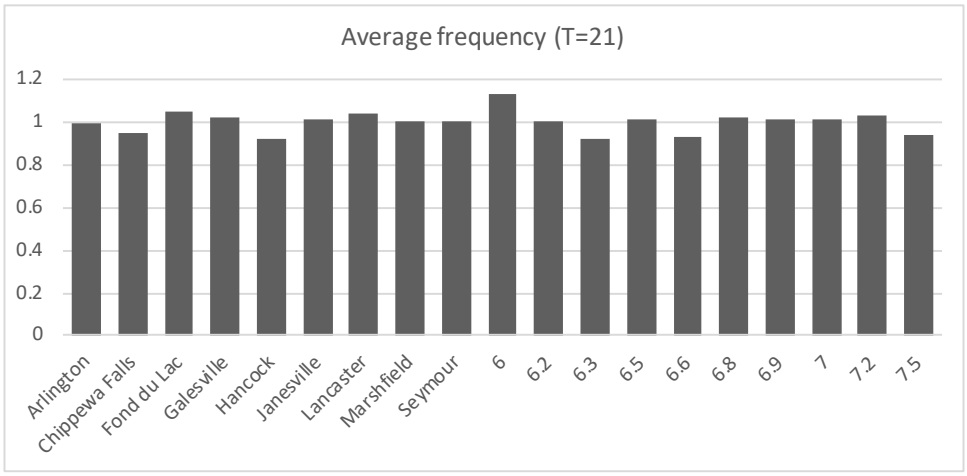

Figure 3: Average frequency of each z value selected over 21 observations

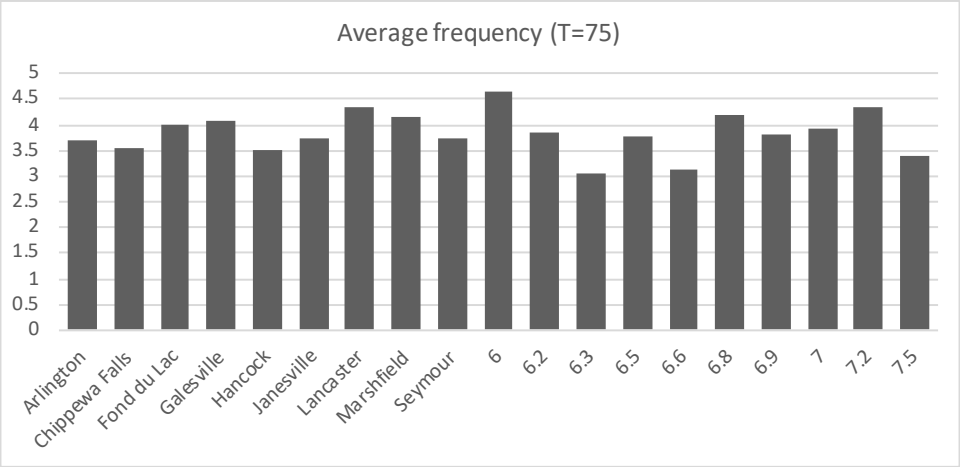

Figure 4: Average frequency of each z value selected over 75 observations

5 Discussion

First of all, it is important to emphasize our algorithmic approach. When we say "pH=6 area is clearly preferred to pH=6.3 area", we do not make any agronomic statement; it simply means that in this particular environment the area characterized by pH=6 provides better samples than that of pH=6.3 in terms of estimation error. pH provides meta-information and is not an explanatory variable for yield in our model. In different environments, the algorithm will mostly likely learn different information and suggest different sampling points. Also, keep in mind that Z_1 is a set of location and somehow partitions the entire fields, and Z_2 is a set of pH that also partitions the fields in a different way. So, clearly, any plot in any region in the fields has both z_1 value and z_2 value, and it is probably the case that some areas characterized by some z values, e.g. area of Lancaster and area of pH=6, are made up of similar parts of the fields. This is both strength and weakness of our approach. Without any information about the characteristics of locations, the algorithm can still make for a particular purpose a useful distinction between them purely based on their locational difference. In contrast, without prior knowledge and insight into the relationships between z features, the algorithm suffers from informational redundancy and are forced to explore similar alternatives. As with most machine learning applications, combination of algorithms and domain knowledge is the key to success (Koller et al., 2009).

Second, our algorithm requires calculating a reward and therefore needs a separate validation dataset. In simulations, we are free to provide any synthetic data for any purpose. In practice, however, most farmers will usually begin with little or no data as they will be experimenting on something new. In this situation, cross-validation potentially offers a way to get around and makes efficient use of data (James et al., 2013). Specifically, modify Line 9-11 of Algorithm 1 as follows. Suppose 5-fold cross-validation to use. Then, at each round t, randomly split the data already collected $S = \{(x_1, y_1), \ldots, (x_{t-1}, y_{t-1})\}$ into five subsets of equal size. Set aside one of them as a validation set and let $\bar{S}$ be the union of the remaining four. Estimate the function based on $\bar{S} \cup \{(x_t, y_t)\}$, and compute a reward using the validation set. Repeat the process for five different validation sets, and if the total of five rewards is positive, then the final reward is $r_t = 1$ and return to Line 12.

Third, our algorithm assumes only a single plot is sequentially added at each round t, as typical multi-armed bandit applications have rapid feedback streams. In agriculture, given the annual

or seasonal timeframe of many production systems, farmers will probably want to add multiple samples at each round. If N plots are sampled for each round t, we could sample all N from the recommended single field area, but it may duplicate the information and waste opportunities for exploration. More promising is either to use the ordering of field areas based on the set of success probability estimates $\{\hat{q}_i\}$ (Line 6 of Algorithm 1) or simply draw the set N times.

Fourth, our algorithm recommends where in a field to place plots to sample from, but it is silent about which seeding rate to use for the chosen plots. Given the objective of efficient estimation of the yield response curve, a natural extension is to develop an algorithm that also recommends the seeding rate to use for the new plots to sample from. Gaspar et al. (2015) data used for this demonstration contains only six different values for the seeding rate (x), ranging from 40,000 to 140,000 seeds per acre. Thus, despite having the total sample size of 1,148, we fit the smooth curve $y = \alpha(1 - e^{-\beta x})$ with little variation in x. Just as it was possible to improve the efficiency of estimation using soil metadata and our algorithm to target placement of plots within the fields, it seems likely that additional efficiency can be gained by selecting input levels more carefully. In general, the selection of experimental levels for continuous treatments has received little attention in the production sciences.

Fifth, recall that our algorithm also treats the number of rounds or observations T as exogenous. In an on-farm context, this assumption implies that a farmer has a set budget to allocate for on-farm research that defines the number T. However, an interesting economic problem is what budget to set, as on-farm experimentation is costly but also beneficial. Costs include plot implementation and differential management, data collection and analysis, the costs for collecting the metadata for soil characteristics and other features, and costs for yield losses or unneeded inputs for super- and sub-optimal input use on experimental plots. These costs must be balanced against the expected benefits of improved profitability in future years as a result of more efficient input use. This higher level economic optimization is beyond the scope of this paper, but nonetheless quite interesting.

Finally, we note that the algorithm is computationally quite light and easy to run on personal computers, smartphones and devices embedded in farm machinery. We are excited about the potential for practical, low-cost, on-farm experimentation that will increase profitability, enhance sustainability, and empower individual farmers in terms of information gathering and decision making.

6 Conclusion

In this paper, we have developed a general machine learning algorithm for efficiently conducting on-farm experiment when site-specific information is available. It is for on-farm as we address the exploration-exploitation dilemma; it is for efficiency as we address the active learning aspect; and it is for generality as applicable to many other choice problems with the same structure. We have conceptualized the on-farm experimentation problem in terms of the multi-armed bandit, a type of reinforcement learning, and built a solution algorithm based on Thompson sampling. Then, we have tested the algorithm in the context of experimental plot selection where a farmer determines which area of the fields to use for experimentation of seeding rate and which areas to use for typical crop production. The results have demonstrated that our algorithm uniformly outperforms the random selection in terms of estimation accuracy. If a farmer becomes satisfied with a certain level of estimation accuracy, our algorithm could enable him/her to achieve it with much fewer samples than random allocation. Since on-farm experiment usually incurs some costs due to the deviation from the empirically best practice, the amount of cost saving could be significant. To reiterate, what we have developed in this paper is an algorithm, instead of estimation of a particular function,

and therefore it can be used in many other situations with the same problem structure. Although these promising initial results suggest that additional research is warranted, as discussed above, some improvements are needed before practical application to real-world production systems.

References

Athey, Susan (2017). "Beyond Prediction: Using Big Data for Policy Problems". In: *Science* 355.6324, pp. 483–485.

Auer, Peter, Nicolò Cesa-Bianchi, and Paul Fischer (2002). "Finite-Time Analysis of the Multiarmed Bandit Problem". In: *Machine Learning* 47.2-3, pp. 235–256.

Berry, Donald A and Bert Fristedt (1985). "Bandit Problems: Sequential Allocation of Experiments". In: *London: Chapman and Hall* 5, pp. 71–87.

Bishop, Christopher M (2006). *Pattern Recognition and Machine Learning*. New York: Springer Science & Business Media.

Bouneffouf, Djallel, Romain Laroche, Tanguy Urvoy, Raphael Féraud, and Robin Allesiardo (2014). "Contextual Bandit for Active Learning: Active Thompson Sampling". In: *International Conference on Neural Information Processing*, pp. 405–412.

Bubeck, Sébastien and Nicolo Cesa-Bianchi (2012). "Regret Analysis of Stochastic and Nonstochastic Multi-Armed Bandit Problems". In: *Foundations and Trends in Machine Learning* 5.1, pp. 1–122.

Chlingaryan, Anna, Salah Sukkarieh, and Brett Whelan (2018). "Machine Learning Approaches for Crop Yield Prediction and Nitrogen Status Estimation in Precision Agriculture: A Review". In: *Computers and Electronics in Agriculture* 151, pp. 61–69.

Coble, Keith H, Terry Griffin, Mary Ahearn, Shannon Ferrell, Jonathan McFadden, Steve Sonka, and John Fulton (2016). *Advancing US Agricultural Competitiveness with Big Data and Agricultural Economic Market Information, Analysis, and Research*. Council on Food, Agricultural, and Resource Economics (C-FARE).

Coble, Keith H, Ashok K Mishra, Shannon Ferrell, and Terry Griffin (2018). "Big Data in Agriculture: A Challenge for the Future". In: *Applied Economic Perspectives and Policy* 40.1, pp. 79–96.

Ganti, Ravi and Alexander G Gray (2013). "Building Bridges: Viewing Active Learning from the Multi-Armed Bandit Lens". In: *arXiv preprint arXiv:1309.6830*.

Gaspar, Adam P, Paul D Mitchell, and Shawn P Conley (2015). "Economic Risk and Profitability of Soybean Fungicide and Insecticide Seed Treatments at Reduced Seeding Rates". In: *Crop Science* 55.2, pp. 924–933.

Griffin, Terry (2010). "The Spatial Analysis of Yield Data". In: *Geostatistical Applications for Precision Agriculture*. Springer, pp. 89–116.

Gutiérrez, Benjamín, Loïc Peter, Tassilo Klein, and Christian Wachinger (2017). "A Multi-Armed Bandit to Smartly Select a Training Set from Big Medical Data". In: *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pp. 38–45.

Hastie, Trevor, Robert Tibshirani, and Jerome Friedman (2016). *The Elements of Statistical Learning*. 2nd ed. New York: Springer.

Huuskonen, Janna and Timo Oksanen (2018). "Soil Sampling with Drones and Augmented Reality in Precision Agriculture". In: *Computers and Electronics in Agriculture* 154, pp. 25–35.

James, Gareth, Daniela Witten, Trevor Hastie, and Robert Tibshirani (2013). *An Introduction to Statistical Learning*. Vol. 112. Springer.

Kamilaris, Andreas, Andreas Kartakoullis, and Francesc X. Prenafeta-Boldú (2017). "A Review on the Practice of Big Data Analysis in Agriculture". In: *Computers and Electronics in Agriculture* 143, pp. 23–37.

Khanal, Sami, John Fulton, and Scott Shearer (2017). "An Overview of Current and Potential Applications of Thermal Remote Sensing in Precision Agriculture". In: *Computers and Electronics in Agriculture* 139, pp. 22–32.

Koller, Daphne, Nir Friedman, and Francis Bach (2009). *Probabilistic Graphical Models: Principles and Techniques*.

Marra, Michele, David J Pannell, and Amir Abadi Ghadim (2003). "The Economics of Risk, Uncertainty and Learning in the Adoption of New Agricultural Technologies: Where Are We on the Learning Curve?" In: *Agricultural systems* 75.2-3, pp. 215–234.

Mishra, Subhadra, Debahuti Mishra, and Gour Hari Santra (2016). "Applications of Machine Learning Techniques in Agricultural Crop Production: A Review Paper". In: *Indian Journal of Science and Technology* 9.38.

Montgomery, Douglas C (2017). *Design and Analysis of Experiments*. John Wiley & Sons.

Patrício, Diego Inácio and Rafael Rieder (2018). "Computer Vision and Artificial Intelligence in Precision Agriculture for Grain Crops: A Systematic Review". In: *Computers and Electronics in Agriculture* 153, pp. 69–81.

Rehman, Tanzeel U., Md. Sultan Mahmud, Young K. Chang, Jian Jin, and Jaemyung Shin (2019). "Current and Future Applications of Statistical Machine Learning Algorithms for Agricultural Machine Vision Systems". In: *Computers and Electronics in Agriculture* 156, pp. 585–605.

Russo, Daniel J., Benjamin Van Roy, Abbas Kazerouni, Ian Osband, and Zheng Wen (2018). "A Tutorial on Thompson Sampling". In: *Foundations and Trends in Machine Learning* 11.1, pp. 1–96.

Settles, Burr (2012). *Active Learning*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool.

Shahriari, Bobak, Kevin Swersky, Ziyu Wang, Ryan P Adams, and Nando De Freitas (2016). "Taking the Human out of the Loop: A Review of Bayesian Optimization". In: *Proceedings of the IEEE* 104.1, pp. 148–175.

Shine, P, Michael D Murphy, John Upton, and T Scully (2018). "Machine-Learning Algorithms for Predicting on-Farm Direct Water and Electricity Consumption on Pasture Based Dairy Farms". In: *Computers and electronics in agriculture* 150, pp. 74–87.

Sun, Lijia, Yanxiang Yang, Jiang Hu, Dana Porter, Thomas Marek, and Charles Hillyer (2017). "Reinforcement Learning Control for Water-Efficient Agricultural Irrigation". In: Ubiquitous Computing and Communications (ISPA/IUCC), 2017 IEEE International Symposium on Parallel and Distributed Processing with Applications and 2017 IEEE International Conference On. IEEE, pp. 1334–1341.

Sutton, Richard S and Andrew G Barto (2018). *Reinforcement Learning: An Introduction*. 2nd ed. Cambridge, MA: MIT press.

Appendix

The algorithm is implemented by the following code in Python 3.6. It takes an input file, input.csv, which contains four columns: "sr", "yield", "location", and "pH" for respectively seeding rate, soybean yield, location and soil pH. Note that we use a unique random seed for each iteration (the first line in the definition of algorithm1).

```
import os
import numpy as np
import pandas as pd
from scipy.optimize import curve_fit
import multiprocessing as mp


def func(x, a, b):
  xx = np.float64(-b*x)
  return a*(1-np.exp(xx))


def algorithm1(ii,iii):
  np.random.seed(ii) # seed = iter number
  T = TT[iii]
  arr_Z = np.array([0]*len(Z))

  rows_pool = np.array(range(num_sample))
  '''Test set'''
  rows_test = np.random.choice(rows_pool,num_test,replace=False
       )
  df_test = df_in.iloc[rows_test, :]
  df_test = df_test[["sr","yield"]]
  '''Validation set'''
  rows_pool = np.setdiff1d(rows_pool, rows_test)
  rows_valid = np.random.choice(rows_pool,num_valid,replace=
       False)
  df_valid = df_in.iloc[rows_valid, 0:2]

  '''Training set'''
  rows_train = np.setdiff1d(rows_pool, rows_valid)
  '''add two random samples to the selected set S'''
  S = pd.DataFrame(columns=df_test.columns) # empty
  samples = np.random.choice(rows_train, 2, replace=False)
  S = S.append(df_in.iloc[samples, 0:2])
  rows_train = np.setdiff1d(rows_train, samples)
  df_train = df_in.iloc[rows_train, :]
  '''for random selection'''
  S_rnd = S.copy() # with same two samples
  samples = np.random.choice(rows_train, T-2, replace=False)
  S_rnd = S_rnd.append(df_in.iloc[samples, 0:2])

  '''Segments'''
  locations = df_train.location.unique()
  pHs = df_train.pH.unique()
  A = [x for sublist in [locations, pHs] for x in sublist]
  K = len(A)
  d_df_seg = dict() # dict of dfs (one df for each segment)
  for location in locations:
    df = df_train.loc[df_train['location'] == location]
    d_df_seg[location] = df[["sr","yield"]]
  for pH in pHs:
    df = df_train.loc[df_train['pH'] == pH]
    d_df_seg[pH] = df[["sr","yield"]]

  a = dict(zip(A,[1]*K))
  b = dict(zip(A,[1]*K))
  q = dict(zip(A,[.5]*K)) # arbitrary
  SSE1 = np.inf # sum of squared error
  SSE2 = 0

  for t in range(2,T):
    for aa in A:
      q[aa] = np.random.beta(a[aa],b[aa])
    arm = max(q, key=q.get)
    arr_Z[Z.index(arm)] += 1 # for counting chosen arms
    df = d_df_seg[arm]
    indices = df.index.values
    ind = np.random.choice(indices)
    S = S.append(df.loc[ind])
    for aa in A: # remove the sample from all the segments
      df = d_df_seg[aa]
      d_df_seg[aa] = df[~df.index.isin([ind])]

    '''estimation'''
    xdata = np.array(S["sr"])
    ydata = np.array(S["yield"])
    (alpha,beta),pcov = curve_fit(func,xdata,ydata,bounds=(0,np
         .inf))
    '''reward'''
    xx = df_valid["sr"]
    yy = df_valid["yield"]
    SSE2 = sum(( yy-func(xx,alpha,beta) )**2)
    r = int(SSE2 < SSE1)
    if r == 1:
      a[arm] += 1
    else:
      b[arm] += 1
    SSE1 = SSE2 # for next loop

    '''remove empty segments'''
    AA = [_ for _ in A]
    for aa in AA:
      if d_df_seg[aa].empty:
        A.remove(aa)
        q[aa] = 0

  '''random selection'''
  xdata = np.array(S_rnd["sr"])
  ydata = np.array(S_rnd["yield"])
  popt, pcov = curve_fit(func, xdata, ydata, bounds=(0, np.inf)
       )
  alpha_rnd, beta_rnd = popt

  '''compare'''
  xx = df_test["sr"]
  yy = df_test["yield"]
  MSE_TS = ((yy-func(xx,alpha,beta))**2).mean()
  MSE_RND = ((yy-func(xx,alpha_rnd,beta_rnd))**2).mean()

  return (ii, iii, np.sqrt(MSE_TS), arr_Z, np.sqrt(MSE_RND))


'''
Main
'''
'''Parameters'''
TT = list(range(5,76,2)) # 5-75 w/ step 2
per_test = 0.3
per_valid = 0.2
iters = 10000
print("iters =", iters)
print("TT =", TT)

'''Initialization'''
os.chdir(os.path.dirname(os.path.abspath(__file__)))
df_in = pd.read_csv("input.csv")
num_sample = df_in.shape[0]
num_test = int(per_test*num_sample)
num_valid = int(per_valid*num_sample)

locations = df_in.location.unique()
pHs = df_in.pH.unique()
Z = [x for sublist in [locations, pHs] for x in sublist]
TS = np.zeros((iters,len(TT),1+len(Z))) # output for TS
RND = np.zeros((iters,len(TT))) # output for Random

'''Parallel processing'''
pool = mp.Pool(mp.cpu_count())
pairs = []
for iii in range(len(TT)):
  tmp = [(ii,iii) for ii in range(iters)]
  pairs += tmp
results = pool.starmap_async(algorithm1,pairs).get()
for r in results:
  TS[r[0],r[1],:] = np.concatenate([[r[2]],r[3]])
  RND[r[0],r[1]] = r[4]
pool.close()


'''Results'''
h = "t,RMSE,"
h = h + ','.join(map(str, Z))
tmp = np.column_stack([TT,TS.mean(axis=0)])
np.savetxt("TS.csv",tmp,fmt="%.4f",delimiter=",",header=h)
tmp = np.column_stack([TT,RND.mean(axis=0)])
np.savetxt("RND.csv",tmp,fmt="%.4f",delimiter=",",header=h)
```