Active Learning in Discrete Input Spaces

Jeff Schneider **Andrew Moore**

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Abstract

Traditional design of experiments (DOE) from the statistics literature focuses on optimizing an output parameter over a space of continuous input parameters. Here we consider DOE, or active learning, for discrete input spaces. A trivial example of this is the k-armed bandit problem, which is the case of having a single input attribute of arity k. We address the full problem of many attributes where it is impossible to test every combination of attribute-value pairs even once within the given number of experiments, but we expect to be able to generalize on the results of experiments. We further pose the problem of active learning on fixed experiment sets where we can not choose any possible setting of the input variables, but instead must choose from a fixed set of available experiments. We discuss discrete DOE and fixed experiment sets in marketing and pharmaceutical domains. We propose several active learning algorithms based on the idea of building a function approximator for the experiments taken so far and using its predictions and confidence intervals to select future experiments. The algorithms are tested using commonly available data sets. We conclude with our ideas for extending these algorithms.

1 Introduction

We consider the problem of optimizing an unknown function where the function evaluations will be expensive and/or limited, and will produce noisy results. A traditional problem of this type is the optimization of a manufacturing process. Usually, the inputs are continuous parameter settings on machines or in recipes. The outputs are the amount and/or quality of the resulting product.

The increased availability of data and popularity of data mining have brought new applications into this domain. Many of these have discrete or mixed discrete and continuous input variables. Two examples are:

- **Marketing.** You have a large database of potential customers and demographic information about them. You will run a promotion for a new product, but cannot afford to blindly apply it to all of the potential customers. Through a series of trial runs, you want to identify which demographic groups will be most responsive to your promotion so you can target them specifically.

- **Pharmaceuticals.** You have a large database of compounds and properties about their structure and behavior. There is a new disease for which you would like to develop a drug. Testing all of the compounds in your database will be impossible because of the time and cost involved. Through a series of trials, you want to identify and test the most promising compounds.

In addition to discrete input variables, there are other differences in the new domains. A traditional process control scenario uses a pre-specified region of valid inputs. Experiments can be taken anywhere within the region and the final result will be a single choice of the best input setting. In the examples above, experiments are chosen from a finite set of possiblities and each may only be available for a single experiment. The final result will not be a single additional selection from the set, but a selection of many candidates from a set of untested possibilities.

1.1 Related Work

Design of experiments is an important topic in the statistics literature. See [10] for an overview. DOE is primarily concerned with continuous valued input variables. Most of these methods have relatively straightforward extensions to discrete input variables, but do not scale to the cases of large numbers of discrete input variables and large arity discrete variables.

Active learning from the machine learning literature attacks the same problem. Often active learning methods use nonlinear function approximators in the place of more conservative models in the statistics literature. Correspondingly, the theoretical basis for these algorithms is less well developed. Some examples with continuous inputs are in [4, 7, 8].

Bandit problems [1] are one of the simplest forms of active learning. The usual problem formulation involves selection from a discrete set of independent arms. The interesting feature of bandit problems, which sets them apart from the previous examples, is that there are tractable algorithms to solve a broad class of them optimally. In the language of bandit problems, we will consider a huge number of arms with the addition that each arm has several features and arms with similar features are expected to have similar payoffs. The authors do not know of an extension to bandit algorithms for this case and suspect that it is not possible to do so while retaining both tractability and optimality. One of the algorithms presented in this paper makes use of a Gittins' indices solution to bandit problems, though the required assumptions are violated and thus optimality is lost.

Active learning is commonly known as the exploration/exploitation problem in the reinforcement learning literature [12]. As with the other problems, the issue is how the reinforcement learner should trade off taking actions that will help it learn against those that will help it gain a larger expected payoff in the short term. As an active learning problem, this is much harder than the others mentioned because of the state space of the system being learned. In the language of reinforcement learning, the examples above consist of a degenerate, single state system with a large number of actions that all produce a stochastic reward and return the system to the same single state.

The reader should not confuse the exploration/exploitation problem in reinforcement learning with the fact that Markov Decision Processes are often used to solve bandit problems. In the latter case, the state of the MDP is the knowledge gained so far from past trials.

Genetic algorithms are commonly used to optimize functions of numerous discrete and continuous input variables. They suffer from two drawbacks in this domain. First they are not explicitly concerned with careful selection of expensive experiments. Their population-based representation of good samples is based on the notion that function evaluations are cheap and noiseless. Second, they work under the classical assumption that there is a space of potential input settings and experiments may be taken anywhere in that space. Their population-based modeling does not offer a way to evaluate the expected outcome of each potential experiment in a finite set of possibilities. The extension of GAs to the applications discussed above is an interesting research area, but not one that will be addressed here.

1.2 Problem Definition

Before describing the algorithms, we will clearly define the problem we are addressing. In the training phase, the learner is given a set of n_{train} training data instances with the associated output values hidden from it. It is allowed to choose k_{train} ($k_{train} < n_{train}$) of these instances for which the output values will be revealed. The choice of the jth element of that subset can depend on the output values observed from the previous $j-1$ choices.

After the training phase, there is a test phase. A new set of n_{test} instances is made available which is drawn from the same distribution as the training instances. The learner must select k_{test} ($k_{test} < n_{test}$) of these instances using the knowledge gained from the training phase. The objective is to maximize the average value of the output on the instances selected during the test phase.

2 Algorithms

2.1 Optimization Methods

All of the optimization methods described here (except RANDOM) depend on a function approximator that returns a prediction and confidence intervals on that prediction. The BANDIT method also makes use of the effective number of data points, that prediction is based on. Prior to each experiment selection, the function approximator is trained on the results of all previous experiments. PMAX and IEMAX are described in more detail in continuous input variable domains in [8].

2.1.1 RANDOM

RANDOM is an algorithm we use for comparison. It chooses each experiment at random from the remaining candidates.

2.1.2 PMAX

PMAX (Predicted MAXimum) estimates the outcome of all candidate experiments and chooses the one with the best predicted outcome. This tends to be the default algorithm used in industrial applications when experimental design considerations are ignored. When faced with the problem of making a business decision the most common response of a manager is, "give me whichever alternative you think will perform the best." It is less common to hear, "give me the one that will help us learn the most about our business environment."

2.1.3 IEMAX

IEMAX (Interval Estimate MAXimum) is an attempt to consider both the expected result of an experiment and the information to be gained by taking the experiment. It looks at the confidence intervals on the expected response and chooses the one with the highest upper confidence interval (in the case of maximization problems). Having a high upper confidence interval indicates a combination of a high expected response and a large amount of uncertainty in the response. Therefore, an experiment there has the potential to be very good and very informative. In the experiments presented below we use 95% confidence intervals for the IEMAX tests.

2.1.4 BANDIT

BANDIT is an algorithm derived from a Gittins' indices solution to bandit problems. A full description of Gittins' indices is beyond the scope of this paper. See [5] for more details.

In order to use Gittins' indices we assume we have k independent arms. Each one, when pulled, returns a continuous valued reward from a Gaussian distribution with some mean and standard deviation. The means and standard deviations are independent for each arm (though we set common bounds and priors on them). The state of each arm is represented in a Markov Chain (MC). The state consists of the number of previous pulls of that arm, and the mean and standard deviation of the resulting rewards. Using standard statistics for estimating the mean and standard deviation of a Gaussian from samples of it we can compute a distribution for the next sample draw (a t distribution), and thus have the transition function for the MC.

In order to compute Gittins indices we construct a Markov Decision Process (MDP) from the Markov Chain described above. In each state, the action choice is to continue another step in the Markov Chain, or to receive a fixed reward and terminate. The MDP uses temporal discounting to make the future rewards available from the MC finite. In each state, the Gittins index for that state is the value of the fixed reward for which the MDP solution is indifferent toward choosing the fixed reward or continuing with the MC. These indices are computed off line and cached by discretizing the state space and considering a discretized set of termination rewards. For each level of the termination reward, a new MDP is solved with dynamic programming.

There is one additional element of complexity in our application. Gittins' indices require a temporal discount factor. Our problem description is not time discounted, but is based on having a fixed number of experiments. There is no known Gittins indices solution with a fixed time horizon. We work around this problem using a simple observation. If a system with a discount factor, γ, receives a reward, R, at every time step going on forever, then its total reward will be $R/(1-\gamma)$. Thus its total reward will be the same as if there is no temporal discounting and it terminates after $1/(1-\gamma)$ steps. This gives a heuristic for converting between the number of time steps remaining and an "effective γ." When solving and caching the Gittins' indices we also do it for a range of γs. Note that doing this breaks the optimality proofs for Gittins indices, but we have found it to behave well in practice.

Once the indices are computed and stored, the experiment selection policy is simple:

1. For each candidate experiment, query the function approximator to get the predicted mean, standard deviation, and number of prior samples.
2. Look up the Gittins index corresponding to the mean, standard deviation, number of samples, and "effective γ" for each candidate experiment.
3. Choose the experiment with the highest Gittins index.

Gittins' indices are optimal on the problem and assumptions for which they were developed. The following three things break that optimality in our implementation:

- The "effective γ" heuristic described above is not optimal. We do not expect this to cause a qualitative difference in performance.
- Gittins' indices are optimal when using on line discounted reward. In the experiment section we compare them using the score on 100 test set selections made after experimentation is over. We also do not expect this to cause a qualitative difference in performance.

- In the case of a single attribute, the assumption of a discrete set of independent arms holds. When multiple attributes and generalizing function approximators are used, this independence assumption is violated. The effects of this are likely to be significant and we regard this as the biggest opportunity for future work.

2.2 Function Approximators

The optimization algorithms discussed above require function approximators that provide confidence estimates in addition to their predictions. In this section we describe two function approximators and the case of doing no generalization in learning.

2.2.1 No Generalization

This is the case of a single input attribute that just specifies which of the finite number of possible input settings is used. In order to predict the expected outcome of a future experiment with a particular setting, we simply find all previous experiments with that setting and compute their mean. We assume the noise is Gaussian, and find confidence intervals on the mean using the appropriate t distribution. In the case of no prior experiments with that setting, we use the mean of all other experiments and confidence intervals wider than those of any other individual setting. In the case of only one prior experiment, we use the outcome of that experiment and confidence intervals as in the case of zero prior experiments. In the experimental results section, we will refer to this as the IAVG (Independent AVeraGe) function approximator.

2.2.2 Regression Trees

We implement an algorithm similar to CART [3]. A tree is greedily constructed in a top-down fashion. At each node all possible split tests are considered of the form: *attribute* i $==$ *value* j. For each test, the data points at that node are separated into those that pass the test and those that don't. Then using the same Gaussian noise assumptions as above, we compute the probability that the means of the two sets are different (this is a difference of two Gaussian populations with unknown and unequal variances, which was done numerically as the difference of two corresponding t distributions). The test that results in the largest probability of unequal means is used to split the node and the process recurses until no split can be found that satisfies a minimum leaf size and a requirement of 95% confidence in a difference between the means of the two children. In the experiments reported below we use a minimum leaf size of 2.

Predictions are made by first identifying the leaf that matches the new case. Then the mean and confidence intervals for that leaf are computed exactly as in the “no generalization” method described above. In the experiments section, we will refer to this function approximator as DTREE.

2.2.3 RADREG

RADREG [9] is a simple linear regression-based machine learning algorithm designed to learn a mapping from discrete-valued inputs to continuous-valued outputs. It operates by searching for useful conjuctive queries that form indicator functions for a linear regression. It is very similar to GMDH modeling [6], Model Trees [11] or even stepwise polynomial regression.

In a dataset with two input attributes "Gender" and "HairColor" and one output attribute "Age", a typical RADREG model might decide to use the following features: (Gender=Female AND HairColor=Grey), (HairColor=Red), (Gender=Male AND HairColor=Red), in which case a regression would be performed with four terms derived from each record. One would be the constant "1", and the other three would be the evaluations of each of the three conjunctive predicates. Assume the dataset has R records. Then, if the k'th record in the dataset were (Gender=Female,HairColor=Red,Age=55), it would be transformed into the vector of terms $(z_{k1}, z_{k2}, z_{k3}, z_{k4}) = (1, 0, 1, 0)$ prior to performing the regression $\beta = (Z^T Z)^{-1}(Z^T Y)$ where Z has R rows and 4 columns and where y_k is the age of the kth record.

Since RADREG is based on classic linear regression, confidence intervals can be obtained using the usual t distributions for the mean response of a linear fit.

3 Experimental Results

We test our active learning algorithms using the adult data set from the UCI Irvine machine learning repository [2]. We use the age as a continuous output and all other attributes as inputs. The other continuous attributes were discretized to three levels for use as inputs. A description of the dataset and its attributes is as follows:

```
Adult Dataset has 48842 records, 15 attributes.
         age real  low 17, high 90, mean 38.64
  employment symb  9 values
taxweighting disc  3 values: 150K- 150K+ 200K+
   education symb 16 values
      edunum disc  3 values: 9.5- 9.5+ 11+
     marital symb  7 values
         job symb 15 values
    relation symb  6 values
        race symb  5 values
      gender symb  2 values: Female Male
 capitalgain disc  3 values: 200- 200+ 600+
```

```
 capitalloss disc  3 values: 100- 100+ 150+
hours_worked disc  3 values: 39.5- 39.5+ 42+
     country symb 42 values
      wealth symb  2 values: poor rich
```

The goal will be to maximize the average value of age for a number of test cases selected after experimentation. The test sequence is as follows:

1. Randomly divide the data set into an experiment set and a test set, with 50% of the records going to each set.
2. Using the algorithms described in the previous section sequentially select records from the experiment set. For each selection the results of all previous experiments are available. Additionally, the input attribute values are available for each record remaining in the experiment set. "Running" an experiment consists of observing the value of age for that record and adding it to the other observed records.
3. Repeat to 2 for a pre-specified number of experiments. We report results for 10, 30, 100, 300, and 1000 experiments.
4. Using the PMAX version of each algorithm on the results of the experiments, select 100 records from the test set. The selection algorithm observes the inputs and outputs of all experiments it has run, and the inputs only of the records in the test set.
5. Compute and report the average value of age for the 100 experiments selected from the test set.
6. Repeat to step 1 for 200 independent trials.

3.1 One Input Case

First we consider the case of a single input attribute: relation. For this case we will ignore all attributes other than relation and age. These attributes have the following properties in the data set:

Attribute value	# of rows	Mean age	Std dev
Husband	19716	43.91	12.05
Not_in_family	12583	38.42	13.99
Other_relative	1506	33.42	14.13
Own_child	7581	24.79	8.11
Unmarried	5125	40.31	13.13
Wife	2331	40.08	11.28

As mentioned above, this is similar to a 6-armed bandit problem, except that we will only score the algorithms based on a final selection of 100 test records at the end of the experiment.

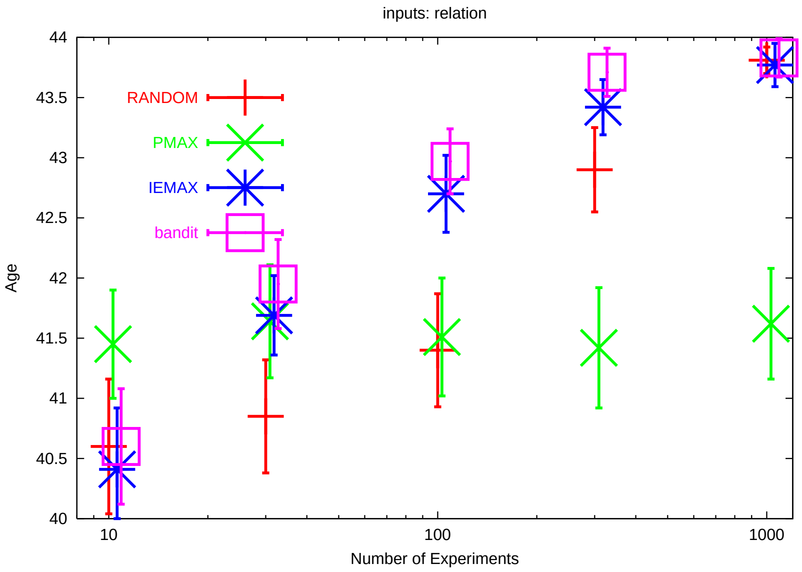

Figure 1: Results of experiments using only relation as an input attribute.

Fig. 1 shows the results of the RANDOM, PMAX, IEMAX, and BANDIT methods with the IAVG function approximator. Since there is only one input attribute, there is nothing to be gained by using the other function approximators. The figure shows the mean and 95% confidence intervals for 200 trials of each method.

The first thing to observe is that PMAX makes little sense in this case. Its resulting behavior is to choose at random until it has seen two different values of the input attribute. Then it continues choosing more cases of whichever one has the higher average. As a result of further experiments, it may change its mind about which of the two is better, but will never try any others. Its performance is nearly flat across the whole range of 10 to 1000 experiments. By looking at the table above, we can see why it does better than the others when only 10 experiments are available. The best value of relation is husband and 40% of the records have that value. Therefore it is fairly likely that husband will be one of the first two values found and PMAX will often discover its superiority over the other.

With only 10 experiments RANDOM, IEMAX and BANDIT perform poorly. They spread their experiments across all values and the result is that they are unable to get accurate estimates of the mean for any value and they often choose poorly at the end. In the range 30 to 300 BANDIT and IEMAX outperform RANDOM. This is where the policy of choosing values that are expected to perform well, but have high uncertainty pays off for IEMAX. It is unsurprising that BANDIT appears to perform best since it is based on an optimal algorithm. With 1000 experiments available any policy will lead to accurate coverage of the space, except PMAX which is still stuck on one of the two values it started with.

From this first experiment we observe three things:

1. With a small number of experiments, it is difficult for any intelligent experimentation to do better than random.
2. With a larger number of experiments, you should carefully explore the space in a way that reduces the variance in your estimates.
3. With enough experiments, you can afford to choose randomly.

In the following sections, we will see how these observations play out as the number of input attributes increases.

3.2 Two Input Case

We now consider two input attributes: relation and job. First we create an additional attribute which is the cartesian product of these two. Although there are 90 possible combinations of the values for relation and job, only 88 of them appear in the data set. We repeated the experiments from the previous section and the results are in fig. 2. The vertical scale on this plot is different from fig. 1, but all other plots to follow will keep this scale.

With 88 possibilities PMAX almost never makes a good choice in one of its first two and always performs poorly. With less than 100 experiments IEMAX behaves similarly to RANDOM because it is explicitly trying to take samples from each of the 88 different possible values. With more than 100 experiments IEMAX is able to make better choices. At some level of experiments the strategy won't matter and RANDOM is as good as IEMAX, but that point is clearly much more than 1000 experiments. Again, BANDIT equals or outperforms all the other algorithms.

With two input attributes available, it is now possible to use the other function approximators. Figs. 3 and 4 shows the results of using the RADREG and DTREE function approximators with PMAX, IEMAX, and BANDIT. The RANDOM results and the BANDIT results without function approximation are also in this figure for comparison. Adding function approximation to the BANDIT algorithm made it worse! We speculate that this is because function approximators explicitly attempt to avoid overfitting and thus base their predictions only on what they have strong support for. This works against the BANDIT algorithm though because it makes its decision explicitly with the amount of data support in mind. It prefers that weakly supported, but high valued predictions are made so it can sort out which to invest more experiments on. In other words, it wants highly speculative predictions as long as the function approximator is accurate in its assessment of its uncertainty about those predictions.

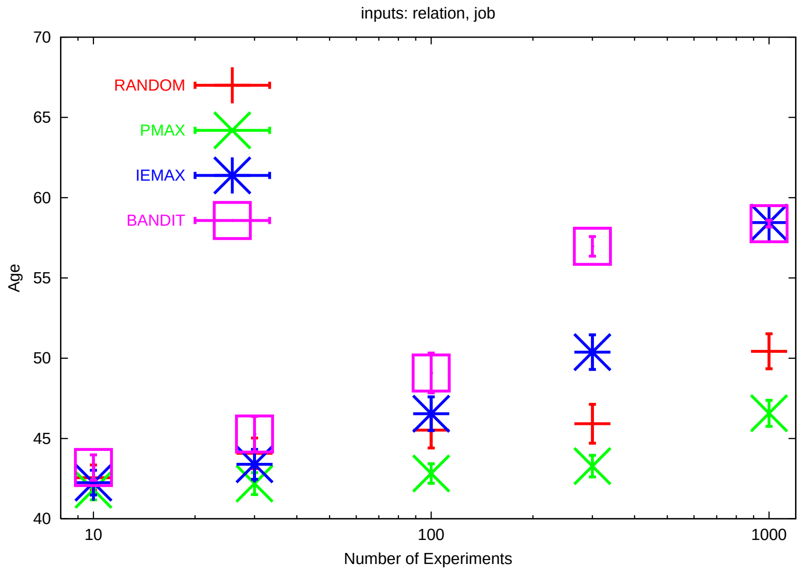

Figure 2: Results of experiments using the cartesian product of relation and job as a single input attribute.

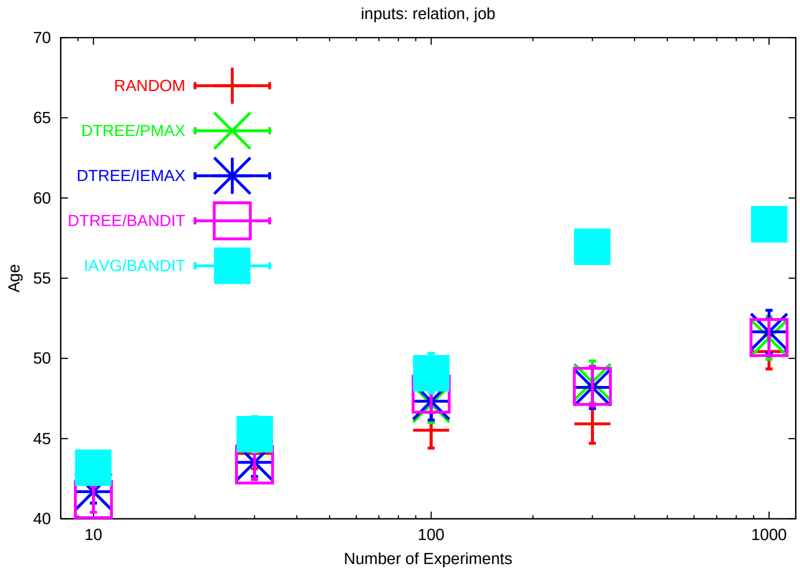

Figure 3: Results of experiments using relation and job as input attributes and the DTREE function approximator.

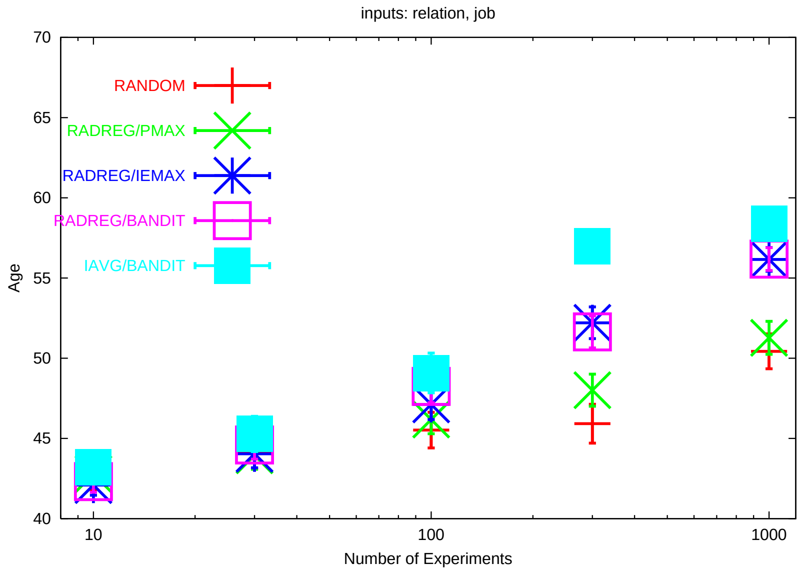

Figure 4: Results of experiments using relation and job as input attributes and the RADREG function approximator.

3.3 Four Input Case

We now consider four input attributes: relation, job, gender, and education. We create an additional attribute which is the cartesian product of those four. 1503 of the 2880 possible combinations appear in the data set. We repeated the experiments from the previous section and the results are in fig. 5. We show only the results using the RADREG function approximator since it performs better. The best earlier results, obtained with BANDIT and no function approximation, are shown as black line for comparison.

The first thing to notice is that none of these improves on the best results for two attributes! In fact, the BANDIT algorithm without function approximation actually does worse when given more attributes. The lack of function approximation and the generalization that goes with it has finally begun to hurt. 1503 possibilities are just too many to handle with 1000 or less trials. Using only two attributes is the same as using all four, but choosing a function approximator that just averages over all values of the extra two attributes, and this is better.

Fig. 5 also shows that the fortunes have changed for PMAX. Unlike before, PMAX actually performs competitively. The reason is that the function approximator selects some attribute-value pairs to base its prediction on, but others are left out and the result is that the chosen experiments have random values of those attributes. The new data points then change the attribute-value

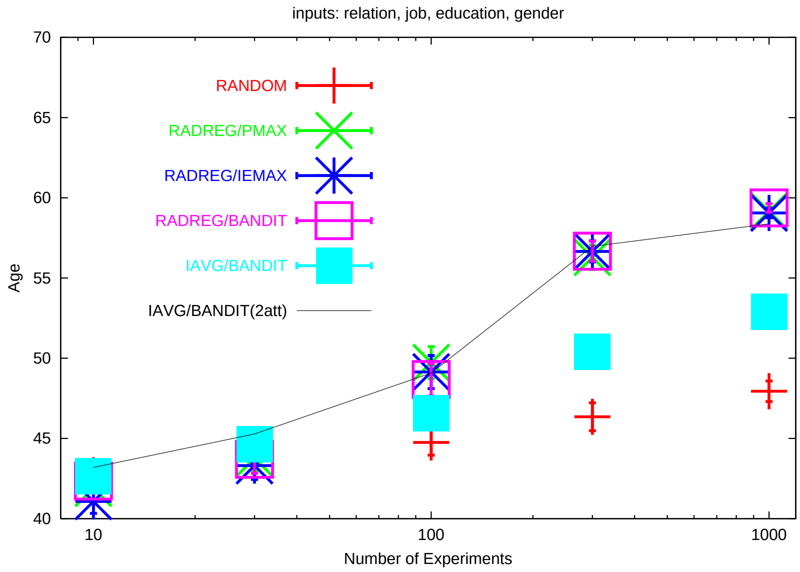

Figure 5: Results of experiments using relation, job, gender, and education as input attributes.

pairs that are chosen by the function approximator on the next iteration. The effect of this is that PMAX inadvertently explores.

3.4 All Inputs Case

Finally, we consider the whole data set. 25025 of the possible values of the cartesian product of all 14 inputs appear in the data set. The experimental results are shown in fig. **??**. The addition of all the extra attributes finally makes it possible for the function approximator to improve the performance of the experiment selection algorithms. Comparing the results in fig. **??** to the best earlier results we see that these are better in all cases, and significantly so in all except the 10 experiment case. Again, the results are shown only for RADREG since the DTREE algorithm performed poorer.

4 Discussion

In our previous work in continuous variables [7] we found PMAX and IEMAX to behave qualitatively differently, and to produce very different results. Similarly, in our experiments with the IAVG function approximator, the two algorithms were very different (see figs. 1, 2). IEMAX has a behavior of "searching ev-

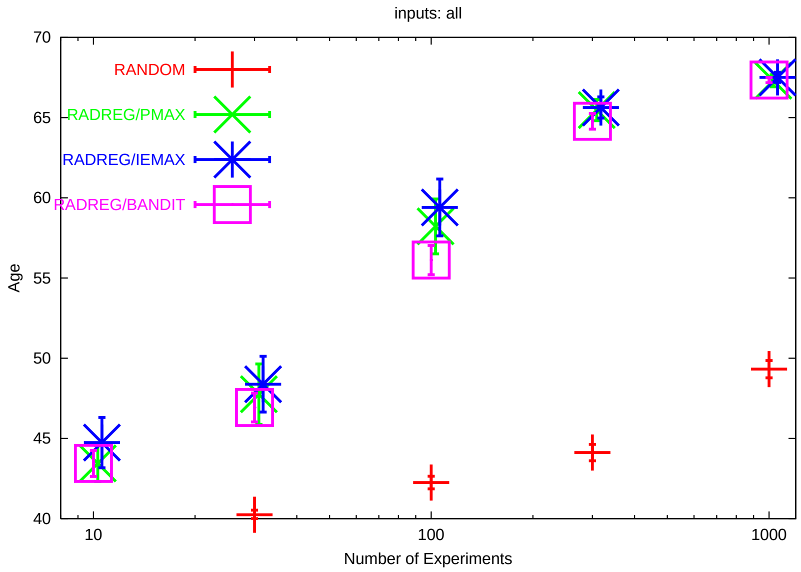

Figure 6: Results of experiments using all input attributes.

erywhere" before trying to narrow down the best of its options, while PMAX blindly goes to the first good thing it sees and stays there.

The results in figs. 5 and 6 are surprising in light of this intuition. The algorithms almost always perform about the same. Further inspection into the RADREG and DTREE algorithms provides an explanation. In our previous work, and in the case of the IAVG function approximator, the space of inputs actually used by the function approximator is fixed in advance. Therefore, it is possible to provide a query for which the function approximator has little or no data and thus has wide confidence intervals. Similarly, it is possible to provide a query for which the function approximator has a large amount of data and very narrow confidence intervals. RADREG and DTREE actively prevent this situation, because they select their features each time based on the data available (RADREG by the creation of its indicator functions, and DTREE by its choice of node splitting criteria). No attribute-value pair with limited support will ever be chosen by either algorithm because its use will be shown to be insignificant. Similarly, any large block of data is likely to be subdivided by DTREE's node splitting, and RADREG will probably find an additional indicator feature that breaks it up. This intuitive explanation is supported by an examination of the size of the confidence intervals encountered by IEMAX during its runs with RADREG and DTREE. Their widths remain fairly uniform for all queries made during experimentation.

With uniform confidence intervals, IEMAX acts like PMAX. We have already seen that PMAX is not a good experimentation algorithm in many scenarios. So the next question is why does it work well with the RADREG and DTREE function approximators? The answer comes from the same property of these algorithms: that they reselect their features each time a new data point is added. Each new data point provides new information about all the input attributes, not just the ones used by the function approximator for that particular experiment choice. This new information may cause the function approximator to choose completely different features the next time it runs. It is this property that prevents PMAX from getting "stuck" in the way it normally does with fixed input features. The best way to understand this behavior is to watch a trace of its execution:

———————————————————————————————-

Experiment 1: Record number 2019 :
age=38, employment=State_gov, taxweighting=v0:150000- , education=Some_college, edunum=v1:9.5+, marital=Divorced , job=Protective_serv, relation=Unmarried, race=Amer_Indian_Eskimo , gender=Female, capitalgain=v0:200-, capitalloss=v0:100- , hours_worked=v1:39.5+, country=United_States, wealth=rich

Experiment 2: Record number 14566 :
age=56, employment=Private, taxweighting=v0:150000- , education=Some_college, edunum=v1:9.5+, marital=Married_civ_spouse , job=Exec_managerial, relation=Wife, race=White, gender=Female , capitalgain=v2:600+, capitalloss=v0:100-, hours_worked=v2:42+ , country=United_States, wealth=rich

Experiment 3: Record number 3588 :
age=53, employment=Private, taxweighting=v0:150000-, education=HS_grad , edunum=v0:9.5- , marital=Married_civ_spouse, job=Transport_moving , relation=Husband, race=White, gender=Male, capitalgain=v0:200- , capitalloss=v0:100-, hours_worked=v1:39.5+, country=United_States , wealth=rich

coefficient	indicator_function
-16.50	gender=Female,hours_worked=v1:39.5+

Experiment 4: Record number 3316 :
age=75, employment=Self_emp_not_inc, taxweighting=v0:150000- , education=Bachelors, edunum=v2:11+, marital=Widowed, job=Prof_specialty , relation=Unmarried, race=White, gender=Male, capitalgain=v0:200- , capitalloss=v0:100-, hours_worked=v2:42+, country=United_States , wealth=rich

coefficient	indicator_function
26.00	capitalgain=v0:200-,hours_worked=v2:42+
-16.50	gender=Female,hours_worked=v1:39.5+

Experiment 5: Record number 19377 :
age=44, employment=Self_emp_not_inc, taxweighting=v1:150000+ , education=Prof_school, edunum=v2:11+, marital=Married_civ_spouse , job=Prof_specialty, relation=Husband, race=White, gender=Male , capitalgain=v0:200-, capitalloss=v0:100-, hours_worked=v2:42+ , country=United_States, wealth=poor

coefficient	indicator_function
27.25	relation=Unmarried,hours_worked=v2:42+
13.50	marital=Married_civ_spouse,wealth=rich
-4.50	capitalgain=v0:200-,wealth=rich

Experiment 6: Record number 18871 :
age=57, employment=Self_emp_not_inc, taxweighting=v1:150000+ , education=11th, edunum=v0:9.5-, marital=Divorced, job=Other_service , relation=Unmarried, race=White, gender=Female, capitalgain=v2:600+ , capitalloss=v0:100-, hours_worked=v2:42+, country=United_States , wealth=poor
——————————————————————————-

After three experiments, RADREG decides that females working more than 39.5 hours per week are on average 16.5 years younger than the rest of the population. This is clearly superfluous, but a male working over 42 hours is selected next for experiment 4. Based on the new data point, the old indicator is retained but another indicating a gain of 26 years for individuals with a capital gain under 200 and more than 42 hours worked is added. Therefore, experiment 5 is a male working over 42 hours with less than 200 in capital gains. Using this data point, RADREG changes its indicators to ignore gender and focus more on wealth, relation, and marital status. The result of that is that a female is once again selected for experiment 6. Similar "new revelations" affect the construction of regression trees and allow PMAX to repeatedly shift its emphasis in search.

5 Future Work

The extension to the case of mixed discrete and continuous variables is relatively straightforward. RADREG already is based linear regression. Continuous input variables can be included along with the indicator functions. Alternatively, or in combination, the continuous variables can also be used in the indicator functions via threshold tests. Similarly, CART and MARS based algorithms extend easily to the case of continuous inputs.

Rather than measuring performance only on a test set chosen at the end, it may be desirable to measure online performance during experimentation. This would result in a desire to be more conservative during experimentation so that performance is not sacrificed much.

The IEMAX and PMAX algorithms do not explicitly consider the number of experiments taken, the number remaining, or the number to be used for testing. Our previous discussion suggests that in the case of the IAVG function approximator, an IEMAX approach should be used earlier in the experimentation while a PMAX approach is more appropriate near the end. In fact, PMAX and IEMAX are on a continuum of algorithms. PMAX is IEMAX with 0% confidence intervals, and intermediate algorithms that focus more on the predicted response or more on the width of the intervals can be obtained by adjusting the percentile. This suggests experimentation with a "confidence interval schedule" that moves from 95% to 0%. Intuitively, it is similar to the annealing schedule in simulated annealing. The BANDIT algorithm is able to produce this effect because a new "effective γ" is computed before each experiment, and in fact we observe it experimenting very aggressively at the beginning and sticking to what it predicts will do best at the end.

The number of experiments to be used for testing at the end is also important. If that number is a significant fraction of the test set, then modeling and identifying the best input attribute settings is not enough. An accurate model of the entire space of inputs will be needed, because in the end it will be necessary to separate even the average candidates from the bad ones. In this case, an experimentation algorithm that focuses strictly on model error over the whole space is needed instead of one that aggressively optimizes the output.

We have not applied genetic or evolutionary algorithms to this problem. Our intuition is that they are relatively inefficient with the modest numbers of experiments used here, but the results of these tests will be interesting.

Possibly the largest open question is what sort of metric should be used to select a function approximator during the experimentation process. Traditional selection of function approximators (and tuning of their parameters) is done with a goal of minimizing prediction error on a test set. Our experiments have already shown that this may not be appropriate. The result may be function approximation that is too conservative – it fails to give the experimentation algorithm any potential leads to track down with new experiments. Further inspection into our experimental results (not presented in this paper) showed that it was frequently the case that a function approximator with worse mean squared test set error yielded better performance in guiding experimentation.

6 Summary

We have presented a new active learning problem – active learning with discrete inputs. We have shown that the performance of common approaches to this problem based on k-armed bandits deteriorate as the number and arity of input attributes increases. We have shown that discrete-input function approximators, which choose their feature sets at training time, can be used to obtain good results on this problem. Furthermore, we have observed that the usual intuitions about PMAX and IEMAX do not hold when the features sets of function approximators are not fixed. Similarly, an algorithm based on the Gittins' indices solution to BANDIT problems was presented. Again with that algorithm it was observed that traditional ideas about good function approximation do not necessarily result in the best experiment selection.

Our experimental results represent a set of "first attempts" on this problem and we believe significantly better performance is possible. In light of that, we offer the data set and optimization problem as a challenge to other researchers.

References

[1] D. Berry and B. Fristedt. *Bandit Problems: Sequential Allocation of Experiments.* Chapman and Hall, 1985.

[2] C.L. Blake and C.J. Merz. UCI repository of machine learning databases, 1998.

[3] L. Brieman, J. Friedman, R. Olshen, and C. Stone. *Classification and Regression Trees.* Wadsworth, 1984.

[4] D. Cohn, Z. Ghahramani, and M. Jordan. Active learning with statistical models. In *Advances in Neural Information Processing Systems 7*, 1994.

[5] J. Gittins. *Multi-Armed Bandit Allocation Indices.* Wiley, 1989.

[6] H. R. Madala and A. G. Ivakhnenko. *Inductive Learning Algorithms for Complex Systems Modeling.* CRC Press, LLC, January 1994.

[7] A. Moore and J. Schneider. Memory based stochastic optimization. In *Advances in Neural Information Processing Systems (NIPS-8)*, 1995.

[8] A. Moore, J. Schneider, J. Boyan, and M. Lee. Q2: Memory-based active learning for optimizing noisy continuous functions. In *International Conference on Machine Learning*, 1998.

[9] A. W. Moore and J. G. Schneider. Radsearch: A new approach for finding optimal rules efficiently from dense datasets. In *AAAI-2002*, 2002.

[10] R. Myers and D. Montgomery. *Response Surface Methodology: Process and Product Optimization Using Designed Experiments.* Wiley, 1995.

[11] J. R. Quinlan. Combining Instance-Based and Model-Based Learning. In *Machine Learning: Proceedings of the Tenth International Conference*, 1993.

[12] R. Sutton and A. Barto. *Reinforcement Learning: An Introduction.* MIT Press, 1998.