Cost-sensitive Selection of Variables by Ensemble of Model Sequences

Donghui Yan[†¶], Zhiwei Qin[$], Songxiang Gu[§],
Haiping Xu[‡¶], Ming Shao[‡¶]

[†]Department of Mathematics and Program in Data Science

[$]DiDi Research America, Mountain View, CA

[§]JD Digital, Mountain View, CA

[‡]Department of Computer and Information Science

[¶]University of Massachusetts Dartmouth, MA

May 14, 2020

Abstract

Many applications require the collection of data on different variables or measurements over many system performance metrics. We term those broadly as measures or variables. Often data collection along each measure incurs a cost, thus it is desirable to consider the cost of measures in modeling. This is a fairly new class of problems in the area of cost-sensitive learning. A few attempts have been made to incorporate costs in combining and selecting measures. However, existing studies either do not strictly enforce a budget constraint, or are not the 'most' cost effective. With a focus on classification problems, we propose a computationally efficient approach that could find a near optimal model under a given budget by exploring the most 'promising' part of the solution space. Instead of outputting a single model, we produce a *model schedule*—a list of models, sorted by model costs and expected predictive accuracy. This could be used to choose the model with the best predictive accuracy under a given budget, or to trade off between the budget and the predictive accuracy. Experiments on some benchmark datasets show that our approach compares favorably to competing methods.

Index terms— Variable selection, cost-sensitive, ensemble, discrete optimization, L_1-regularization, model schedule, classification

1 Introduction

Many applications require the collection of data on different variables or measurements over a number of system performance metrics. For example, some cyber systems rely on scanning various system metrics to detect or to predict potential cyber intrusions or threats. In the maintenance of airplanes or major factory machinery, measurements of different system components and their

usage statistics are collected to determine when a maintenance is required. In medical diagnosis, a patient may be asked to take various medical tests on measures such as blood pressure, cholesterol level, heart rates and so on, so that the doctor could determine if the patient has a certain disease. In the development of an e-commerce product that predicts the click or purchase of a product at an e-commerce website, many data related to a user's shopping behavior will be collected, and often extra data relevant to the product or the user's shopping behavior are purchased from a third-party vendor etc. The data collected on various measures need to be combined, and if cost is a concern, a subset of measures needs to be selected to satisfy the budget constraint.

The problem of combining measures for a target application can be formulated as follows. Assume there are p measures, then a measurement of the system will be a vector in $\mathbb{R}^p$. Let $X_i = (X_{i1}, X_{i2}, ..., X_{ip})$ be an instance of *measurement* with X_{ij} indicating the i^{th} measurement on the j^{th} measure. Each measurement is associated with a state variable, denoted by Y_i, indicating the system status. Examples of the state variable include an indicator of whether a person is healthy or otherwise in a health screening or diagnosis, whether a major repair is required in airplane or machinery maintenance, whether a cyber system is under attack, or an indicator on the click or purchase of a product item in an e-commerce application. By collecting a sample of measurements and the associated status, we can estimate their relationship

$$f : X \mapsto Y.$$

That is, a model of the system operation—the relationship between the measurement and the system status. Or, for measurement $X = x$, what would be the likelihood of a certain event, such as a disease, a cyber attack, or an immediate repair of some airplane parts or machinery. Our interest is to solve the prediction or classification problem. Formally, we seek to solve the following

$$\arg\min_{f \in \mathcal{F}} \mathbb{E} l(f(X), Y), \tag{1}$$

where $\mathcal{F}$ is the function class of interest, such as linear classifiers, decision trees etc, $l(.,.)$ is the loss function, and $\mathbb{E}$ indicates that we are taking expectation over the distribution of (X, Y) (i.e., expected risk in future prediction). The simplest loss function is the 0-1 loss, for which (1) amounts to solve for f for the best predictive accuracy. This is our focus for the present work.

In practice, the measurements along each variable may incur a cost; sometimes the cost may be substantial. Let $\boldsymbol{b} = (b_1, ..., b_p)$ denote the *cost profile* where b_i stands for the cost of the i^{th} variable. It is highly desirable, sometimes mandated, that the model could be built under a total budget, say, B. That is, the total cost for variables used by the model satisfies the following constraint

$$\sum_{\beta_i \neq 0} b_i \leq B, \tag{2}$$

where β_i is either the coefficient of the i^{th} variable in a linear model, or otherwise an indicator of whether the i^{th} variable is present in the model, $i = 1, ..., p$. We call (1), with the additional constraint (2), the problem of *cost-sensitive*

selection of measures, and this is the focus of the present work. When the cost of all variables are equal, i.e., $b_1 = b_2 = ... = b_p$, then the above reduces to the usual feature subset selection problem.

Finding a subset of variables so that they collectively achieve a good predictive accuracy is a challenging problem. The major difficulty lies in the fact that it is a discrete optimization problem (or more particularly, the *cardinality problem*)—as for those variables taking discrete values, one cannot apply gradient descent types of algorithm and *all possible combinations of possible discrete values* will have to be evaluated in order to find the optimal solution. When the number of variables increases, it quickly leads to a combinatorial explosion. Clearly the problem becomes more difficult when incorporating a budget constraint on the total cost of selected variables. As a result, often solutions resort to heuristics.

The overall strategy in our proposed approach is to find ways that could explore the solution space in an efficient way. We achieve this by aiming at those *critical points* in the solution space, in the sense that such points are either themselves 'special' (one could think of such points as vertices of the polytope of feasible solutions in linear programming [21]) or would allow us to gauge the value of many others approximately. Thus, if one is able to visit those critical points, or come close to such points, then effectively one has explored a large portion of the solution space. We implement this by following a number of 'promising' solution sequences. Each solution sequence consists of a series of solutions to the target optimization problem (i.e., a model to the cost-sensitive learning problem) that are organized by simulating the path of a gradient optimization along a certain direction. This leads to either a sequence that progressively removes the least predictive variables when starting with the full model (that is, a model that uses all the variables), or that sequentially removes the most costly variables, or that iteratively removes the least important variables by randomly sampling with a weight inversely proportional to some normalized importance measure. As the total cost of variables can be calculated for each model, thus by examining individual models in all the model sequences, we can find all solutions among the model sequences that satisfies a predefined budget constraint. Then we can learn the predictive accuracy of all such solutions using some validation data, and pick the one with the best predictive accuracy as the solution to the budget constrained learning problem.

Our contributions are as follows. We rigorously formulate the problem of learning under a budget constraint on variables. Previous work either does not strictly enforce the budget constraint (e.g., [40]), or is limited to a particular learning method (e.g., [24] specializes to L_1-logistic regression). Our approach would work for any classification method, especially those strong classifiers, such as Random Forests (RF) [3], boosting and its variants [13, 14, 7]. Our approach also makes it possible to use different classifiers to generate multiple model sequences. As an ensemble based method, one would likely see a boost in performance, also it is possible to generate individual model sequences in the ensemble in parallel by multicore computing [34]. We discover a new use of the L_1-regularization path as an efficient way of exploring the model space, rather than a necessary part of the model fitting procedure; thus one can use

L_1-logistic regression to generate the regularization path and then apply a different classifier on the set of variables selected at each step of the regularization path. We generalize the concept of a *single predefined budget* to that of a model schedule, which specifies what would be the best predictive model under each of a list of varying budget levels. This leads to a much greater flexibility in practice. Given a budget, one can choose a model from the model schedule that would deliver the best predictive accuracy, or achieve a given prediction accuracy level with the least budget, or trade-off between the budget and the expected predictive accuracy. Finally, our idea of multi-path search, along with the learning of model power from data and multicore computing, can become a general strategy for efficiently finding an approximate solution for a large class of discrete optimization problems.

The remaining of this paper is organized as follows. In Section 2, we introduce necessary background and discuss related work. This is followed by a detailed description of our proposed approach in Section 3. In Section 4, we define the optimal model schedule, and use a small scale problem to demonstrate the optimality of the model schedule produced by our algorithm. In Section 5, we present experimental results on some real datasets. Finally, we conclude in Section 6.

2 Background and related work

In this section, we discuss necessary background and work related to ours. We start by an introduction to two machine learning algorithms, RF and L_1-logistic regression, which are important ingredients of our algorithm. Then we discuss work that are related to ours.

2.1 Random Forests

RF is widely viewed as one of the most powerful tools in statistics and machine learning according to many empirical studies [3, 6, 5]. It is an ensemble of decision trees with each tree constructed on a bootstrap resample of the data. Each tree is built by recursively partitioning the data space. At each node (the root node corresponds to the bootstrap sample), RF randomly samples a number of features (or sets of features) and then select one that would lead to an 'optimal' partition of that node. This process continues recursively until a stopping criterion is met. [3] argues RF would achieve an 'optimal' bias and variance combination by fully growing individual trees (for classification). RF is easy to use (e.g., very few tuning parameters) and shows a remarkable built-in ability for feature selection.

The computational complexity of RF can be calculated as follows. The average height of a tree is given by $O(\log(n))$, and each level in the tree involves the calculation of the merit of $O(p^{1/2})$ candidate variables on all the observations and the subsequent search of best split, which costs $O(p^{1/2} \cdot n \log(n))$ in total. Assume the total number of trees in RF is T, then the average computational complexity in growing RF is given by $O(T \cdot p^{1/2} \cdot n(\log(n))^2)$. Similarly the applying RF on the test or validation set, with a sample size of n, costs

$O(T \cdot n \log(n))$.

2.2 The L_1-logistic regression

The L_1-logistic regression is the usual logistic regression with an L_1-penalty on the coefficients in the logistic regression model. That is, to solve optimization problem (3).

$$\arg \max_{\beta=(\beta_1,...\beta_p)} L(\beta) = \Pi_{i=1}^{n} \left[p(X_i)\right]^{Y_i} \left[1 - p(X_i)\right]^{1-Y_i} + \lambda \sum_{i=1}^{p} |\beta_i|, \qquad (3)$$

where λ is a regularization parameter, $p(X_i) \triangleq P(Y_i = 1|X_i)$ is the posterior probability, $\beta = (\beta_1, ..., \beta_p)$ is a vector of coefficients in the linear model (logistic regression models the log-odds ratio of the posterior probability as a linear model of independent variables) and $i = 1, ..., n$, and n is the training sample size. The optimization problem (3) is typically solved by a gradient descent type of algorithm. A variant, coordinate descent, is used by the popular package *glmnet()* [15], which does gradient descent along one variable at a time while keeping other variables fixed, and the variables are evaluated in turn until convergence (after a predefined number, E, steps). A full range of different values of λ are attempted, and each leads to a feasible solution. One then uses some model selection procedure, such as cross-validation, to select the λ value that would lead to the best predictive performance.

The L_1-regularization path is a sequence of solutions to (3) under different values of λ such that $\lambda_1 > \lambda_2 > ... > \lambda_{n_\lambda}$, where n_λ depends on the number of steps one wishes to include in the regularization path. Typically λ_1 is chosen such that the model consists of only the intercept, $\lambda_{n_\lambda} = 0$ implies no regularization, and $\lambda_i, i = 2, ..., n_\lambda - 1$, are chosen adaptively such that their choice will cause a change to current set of variables in the model. For details about model fitting in L_1-logistic regression, please refer to [26, 15]. Each solution to (3) corresponds to a model. A nice property of L_1-regularization is the sparsity of the solution, i.e., if one keeps on increasing λ, then the coefficient of some parameters will shrink towards 0. This can be seen in Figure 1. Thus a regularization path corresponds to an organized sequence of fitted models.

The computational complexity of L_1-logistic regression is calculated as follows. The computing of gradient optimization in logistic regression costs $O(p \cdot J \cdot E \cdot n)$, where E is number of epochs in gradient descents and J is the number of classes. To generate the full regularization path for L_1-logistic regression involves n_λ steps of logistic regression thus has a computational complexity $O(n_\lambda \cdot p \cdot J \cdot E \cdot n)$ [26, 15].

2.3 Related work

Work related to ours fall into two categories. One is on variable selection, also known as feature selection or model selection. The other is on work that incorporates a cost in the model, known generally as cost-sensitive learning. The literature on feature selection is enormous, we shall refer the readers to [1, 28, 20, 17, 27] and references therein for early work. More recent

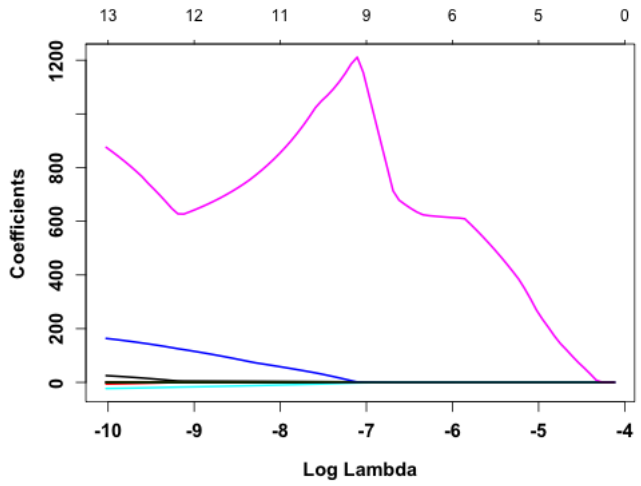

Figure 1: *The regularization path of L_1 logistic regression. As λ increases, the value of some coefficients will shrink to 0.*

developments include numerous methods based on the idea of regularization [32, 11, 41, 22, 26, 15], feature screening [33, 35], univariate statistics [10, 9, 31] etc. The development on feature selection has been explosive during the last decades, and references listed here are just a small sample of the huge body of literature.

The seminal work by [12] is an early work on cost-sensitive learning. It is an example that associates costs with data instances for a classification error. In classification, usually the same loss is incurred to each data instance for en error, but [12] distinguishes errors committed to different classes and charges at different costs. For example, there would be a different cost for errors in classifying a safe system to be under attack and those errors in mis-detecting a cyber attack. There are a number of followups [29, 25] and extension of the cost to per example based [39]. Cost-sensitive learning has also been studied in the setting of active classifier [16] and adaptive feature acquisition [18].

Incorporating a cost for features is a fairly new area. [40] considers the cost of variables in RF, where, at each node split in the construction of decision trees, variables are selected by sampling with a probability inversely proportional to their costs. While the resulting classifier may have a low-cost in variables, it does not necessarily satisfy the budget constraint. In cases when those more predictive variables also have a high variable cost, this will badly hurt the performance of the resulting classifier as one looks for *collectively not individually* inexpensive variables. The work that is most closely related to ours is [24] which achieves a cost-sensitive combination and selection of measures by an L_1*-logistic regression* formulation. In particular, it incorporates an L_1-penalty [32, 26, 15] in the model fitting of logistic regression with an additional cost constraint as

follows

$$\arg \max_{\beta=(\beta_1,\ldots\beta_p)} L(\beta) = \Pi_{i=1}^{n} \left[p(X_i)\right]^{Y_i} \left[1-p(X_i)\right]^{1-Y_i} + \lambda \sum_{i=1}^{p} |\beta_i|, \quad (4)$$

$$\sum_{\beta_i \neq 0} b_i \leq B. \quad (5)$$

Then it navigates through the L_1-regularization path for (4), and generates a sequence of models with each using the β corresponding to a different value of λ (there are totally n_λ such λ's). As L_1-regularization encourages sparse models, the solution given by [24] typically yields a model with a satisfactory predictive performance at a low cost. The budget constraint (5) is enforced by following the L_1-regularization path, and then the best predictive model (based on performance on a validation set) that is under the budget is selected as the solution to the variable selection problem under a budget. While this work selects variables that enforces the budget constraint, it is limited to L_1-logistic regression.

We take a broad view of the L_1-regularization path explored in [24], and view it as an effective search path in the solution space to the challenging optimization problem for the cost-sensitive variable selection problem. This inspires us to consider more search paths other than the L_1-regularization path, which may potentially overcome its limitations, thus more model sequences can be packed to form an ensemble; this would lead to a substantial gain in the resulting model. We also extend the scope of budget, thus more flexibility, by proposing a concept called the *model schedule* which is a table of budgets and the associated best predictive model under each budget level. Thus, one just needs to run our algorithm once and would then be able to give the best predictive model for any particular budget level, or to tell what would be the minimal budget required to deliver a model with a predefined predictive accuracy.

3 The method

Our main idea in solving the challenging optimization problem for the selection of variables under budgets is to efficiently explore the most promising part of the solution space. This is implemented by visiting multiple model sequences, with each having the potential of finding a 'near' optimal solution (each model corresponding to a point or a feasible solution to the target optimization problem). Such model sequences are typically greedy in nature, and pursued separately in the practice of variable selection and often lead to fairly decent result. Thus we expect each individual model sequence explored in our search are already 'close' to the optimal solution. Now combining such model sequences would likely lead to improvement. As the resulting model sequences are often a nested sequence of models (i.e., the set of variables in a model is a subset of that of preceding models in the sequence), they can all be constructed efficiently.

Note that, for each model in a model sequence, we can form a tuple by the parameters of this model, along with the model cost and an expected predictive accuracy produced from a validation set. We can treat tuples resulting from the same model sequence as a set. This would allow one to easily combine

model sequences as the union of sets. To make the model sequence directly useable in practice, we apply two operations. First the model sequence is *sorted* by the model cost and predictive accuracy. Then, the model sequence is *compressed* such that those members with a higher model cost but lower predictive accuracy than others will be removed from the model sequence. This produces a *model schedule* which is a list of models with increasing cost and predictive accuracy. Note that a model schedule has the monotonicity property—higher cost models in the schedule always lead to a higher expected predictive accuracy.

For the rest of this section, we will describe different ways in generating model sequences, and how to produce the output model schedule from such model sequences. The generation of such model sequences is illustrated by the Naval propulsion plants (NPP) dataset (for a brief description, please see Section 5).

3.1 Generating multiple model sequences

We consider four different ways of generating model sequences, including following the L_1-regularization path, selecting variables by their importance, selecting variables by their costs, and sampling according to a tradeoff between cost and variable importance. Other ways of generating model sequences, such as forward stagewise variable selection, can also be considered. We will discuss each of the four different model sequence generation procedures in the sequel.

3.1.1 Model sequence by importance or cost of variables

To produce a nested sequence of models by variable importance, we first rank the variables by their importance to predictive accuracy. There are many ways around in doing this, for example, by t-statistics [20]. As we use RF as the engine for selecting and combining variables, we will use a built-in tool by RF to produce a variable importance profile. There are two feature importance metrics in RF, one based on the Gini index [4, 3] and the other permutation accuracy. We consider the later here, as it is often considered superior. The idea is as follows. Randomly permute the values of a feature, say, the i^{th} feature, then its association with the response Y is broken. When this feature, along with those un-permuted features, is used for prediction, the accuracy tends to decrease. The difference in the prediction accuracy before and after permuting the i^{th} feature can then be used as a measure of its importance. Figure 2 shows the relative importance of different variables used in the NPP data.

With a profile of variables importance, a nested sequence models is produced as follows. We start with the *full model*, that is, a model with all variables present. Then, we delete the least important variable, according to its importance value; this gives a new model. We record its predictive accuracy on a validation set and compute the total cost of all variables in the new model. This procedure continues until there are two variables left (at which point we have to stop as RF does not allow less than two variables). This produces a list of models, with such information as model cost, predictive accuracy, and variables used. From this list, we can generate a model schedule specifying at which cost, what kind of predictive accuracy we can expect. It may happen that there are models in the list with higher cost but lower predictive accuracy,

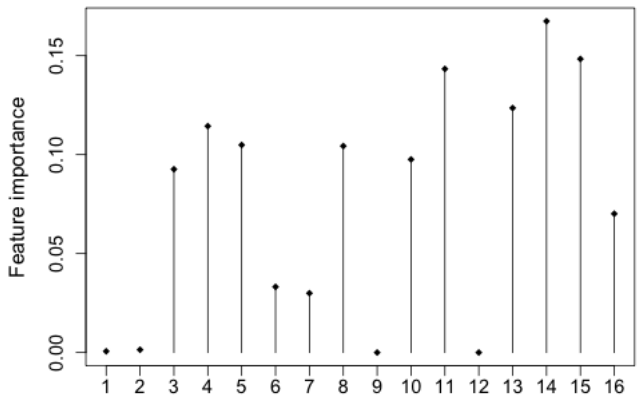

Figure 2: *Feature importance produced by RF on the Naval propulsion plants dataset. The x-axis indicates feature index.*

when this happens we simply remove such models from the list. Thus in the final model schedule, a higher-cost model would have a higher expected predictive accuracy. The following is an instance of model schedule under a certain cost profile (i.e., $\boldsymbol{B} = (b_1, b_2, ..., b_p)$ where b_i is the cost of the i^{th} variable) and the variable importance profile shown in Figure 2. Here the cost profile is generated by sampling uniformly at random from $[1, 100]$; the same applies to all figures in this section.

```
      Cost    Accuracy       Variables
[1]   417     0.9907834     4,5,8,11,13,14,15
[2]   385     0.9874319     4,5,11,13,14,15
[3]   340     0.9773775     4,11,13,14,15
[4]   248     0.9706745     11,13,14,15
[5]   171     0.9400922     11,14,15
[6]   119     0.8504399     11,14
```

For a budget level not in the list, one can look up the model schedule and interpolate the expected predictive accuracy. For example, for a budget $B \in [171, 248)$, the expected predictive accuracy would be 0.9400922 (a conservative estimation). A visualization of the model schedule is shown in Figure 3. The *staircase curve* shows the expected predictive accuracy at different budget levels.

A similar model sequence can be generated by using the cost profile of variables. We start with the full model. Then we recursively prune the most expensive variable that remains until we are left with two variables. Necessary bookkeeping allows us to construct a model schedule similarly as that by variable importance. The resulting model schedule will be visualized along with

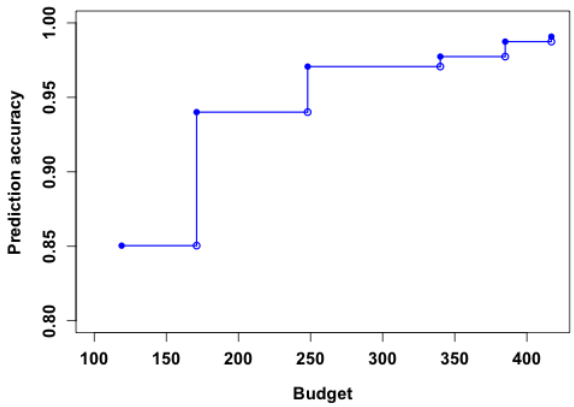

Figure 3: *A graphical representation of the model schedule produced according to the importance of variables. Open circles on the curve mean that the corresponding points are not defined according to conventions in mathematics.*

that by other model sequences in Figure 4.

The above two model sequences are generated according to a single metric, the importance or the cost of variables. However, the model schedule clearly depends on both, maybe also other factors, in a complicated way. As a simple case to start, one can assume that the dependance is only on the importance and the cost, and is captured by a function $f(b_i, I_i)$ where b_i and I_i are the cost and importance of the i^{th} variable, such that f is proportional to the variable importance and inversely proportional to its cost. Here we consider a simple case

$$f(b_i, I_i) = (I_i/b_i)^\gamma, \tag{6}$$

where γ is a parameter (set to be 0.1 in this work). f is called the *normalized importance* of a variable. Other choices of f include $f(b_i, I_i) = \alpha_1 I_i + \alpha_2(1/b_i)$, which we leave to future work. We will start from the full model, then sample variables at a probability inversely proportional to their normalized importance. Once a variable is selected, it is removed from the current model. That is, less 'important' variables are removed from the model first. This continues until only two variables are left. Related to this, a model sequence can be generated by sampling the variables uniformly at random. Note that the sampling procedure introduces randomness in the selection of variables; if the total cost for a particular model in the sequence exceeds the given budget, then it would be discarded. Again, the resulting model schedule is illustrated in Figure 4.

3.1.2 Model sequence by L_1-regularization path

The L_1-regularization path, as a way of generating a model sequence, is attractive for its computational efficiency, and efficient algorithms [26, 15] have been developed to generate the entire regularization path. In this work, we use the

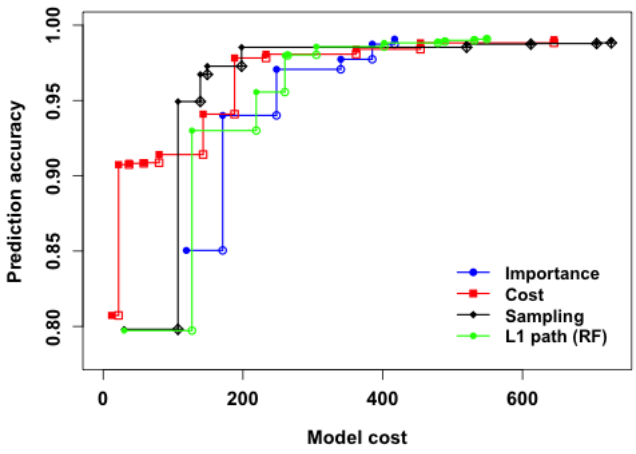

Figure 4: *Model schedules generated by different model sequences, including that by variable importance, by variable cost, sampling by normalized variable importance, and by following the L_1 regularization path. Note that here RF is used as the classifier to generate all the model schedules, and we use "L_1 path (RF)" to emphasize that the classifier is RF not L_1-logistic regression.*

glmnet() package [15] for generating the L_1-regularization path.

By following the L_1 regularization path, i.e., run RF on the set of variables corresponding to a different λ, one can keep track of the model cost (that is, *the total cost of all variables with a nonzero coefficient*) for each model along the path. The predictive accuracy can be evaluated on a validation set. From models along the regularization path, one can produce a model schedule. The user can then pick a model from the schedule with the best accuracy such that the total model cost is under a budget B, or to tradeoff between cost and accuracy.

Next we give an algorithmic description of generating a model sequence by following the L_1-regularization path. Let $(\boldsymbol{X}, Y)$ be the input data. Let vector $\boldsymbol{B} \in \mathbb{R}^p$ be the cost profile for the p variables. Assume there are J classes. Let n_λ be the number of different λ's we take along the regularization path. Let $\Theta_{J\times p\times n_\lambda}$ store the fitted coefficients for L_1-logistic regression, i.e., it consists of coefficients for each variable and each class for each step along the regularization path (totally n_λ steps). Let $\mathcal{M}_L$ be the model schedule produced by following the L_1-regularization path. The algorithm is described as Algorithm 1. Generating a model sequence by following the L_1-regularization path involves the computation of predictive accuracy by RF on the validation set for each of the n_λ set of variables, the calculation of the total variable costs and necessary bookkeeping along the regularization path. In total this costs $O(n_\lambda \cdot (T \cdot p^{1/2} \cdot n(\log(n))^2))$. Thus the computational complexity for Algo-

Algorithm 1 *modelSeqL(X, Y, $\boldsymbol{B}$)*

Invoke *glmnet*() with the *training* data;
Initialize the model schedule $\mathcal{M}_L \leftarrow \emptyset$;
for $i = 1$ to n_λ **do**
  Let α_i be predictive accuracy on the validation set;
  for $j = 1$ to J **do**
    Let V_j store the index of variables used for class j;
  end for
  Set $V_{used} \leftarrow \cup_{j=1}^{J} V_j$;
  Calculate total cost β_i of all variables based on V_{used} and $\boldsymbol{B}$;
  Add the new model to model schedule by $\mathcal{M}_L \leftarrow \mathcal{M}_L \cup \{(\beta_i, \alpha_i, V_{used})\}$;
end for
Return($\mathcal{M}_L$);

rithm 1 is

$$\begin{aligned} & O\left(n_\lambda \cdot p \cdot J \cdot E \cdot n + n_\lambda \cdot (T \cdot p^{1/2} \cdot n(\log(n))^2)\right) \\ = \; & O\left(n_\lambda \cdot max\left(p^{1/2} \cdot J \cdot E,\ T \cdot (\log(n))^2\right) \cdot p^{1/2} \cdot n\right). \end{aligned}$$

It follows that the overall computational complexity of our algorithm is give by $O\left(n_\lambda \cdot max\left(J \cdot E,\ T \cdot p^{1/2} \cdot (\log(n))^2\right) \cdot p \cdot n\right)$.

3.1.3 Example model schedules for the NPP dataset

Figure 4 shows the model schedule produced by four different ways of generating model sequences—by variable importance, variable cost, sampling with normalized variable importance, and L_1-regularization path. It can be seen that each resulting model schedule has its own merit, and no one dominates others. By 'dominate' we mean the staircase curve corresponding to one model schedule is higher than that of another at *all* different budget levels.

A curious question is, does RF improve over L_1 logistic regression, if following the same L_1-regularization path? The answer is 'Yes' for the NPP dataset. This is illustrated in Figure 5, which shows that the model schedule produced by RF dominates that by L_1 logistic regression with a large margin. This gives support for RF to be a preferred engine for the selection and combination of variables in some applications.

A more important question is, does ensemble, i.e., a model schedule produced by combining those generated by different ways, improve the model schedule? The answer is 'Yes' for the NPP dataset. Figure 6 is an illustration. The staircase curve by an ensemble of four model sequences dominates those by any individual ones. Indeed this is a consequence of the way that different model schedules are combined in our approach, and by packing more model sequences into the ensemble will results in a better model schedule.

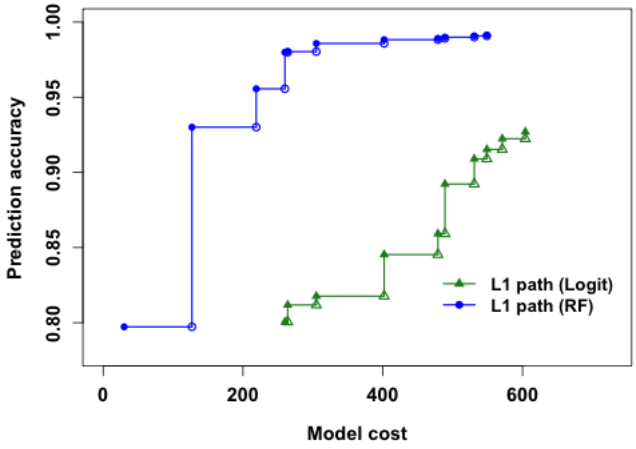

Figure 5: *RF Vs logistic regression by following the same L_1-regularization path.*

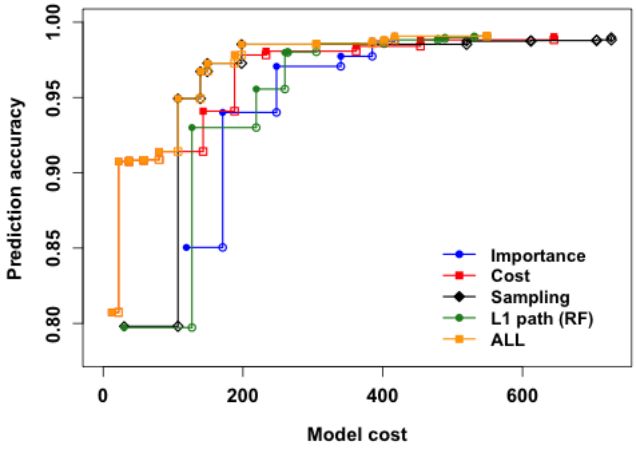

Figure 6: *The ensemble (indicated by 'All') vs individual model schedules.*

3.2 Algorithmic description

In this section, we will describe algorithms to implement our approach. Let $\mathcal{M}$ be the *final model schedule* produced by our approach. That is, by combining model schedules generated by members in the ensemble; our current implementation consists of model schedules generated by variable cost, by variable importance, by sampling, and by following L_1-regularization path. The combining of multiple model sequences is implemented by treating each model sequence as a *set* of triples *(accuracy, cost, variables)*, then we take the union of all such sets from individual model sequences. This is then compressed by removing those triples corresponding to a higher cost but lower predictive accuracy. Our approach is implemented as three algorithms

- $msB()$ for generating the final model schedule from multiple model sequences
- $modelSeq()$ for generating a model schedule for a given type of model sequence
- $modelSeqL()$ for generating a model schedule by following the regularization path of L_1 logistic regression.

The algorithm for $modelSeqL()$ is described as Algorithm 1 in Section 3.1.2, and that for $msB()$ and $modelSeq()$ are described as Algorithm 2 and Algorithm 3, respectively, in the rest of this section, where $\boldsymbol{W}_i$ and $\boldsymbol{W}_u$ denote the importance and normalized importance of variables, respectively.

Note that our approach does not exclude the use of classifiers other than RF,

Algorithm 2 *msB(X, Y, $\boldsymbol{B}$)*

1: Invoke RF with the training and validation set;
2: Generate model sequence by variable cost $\mathcal{M}_C \leftarrow modelSeq(X, Y, \boldsymbol{B}, 2)$;
3: Generate model sequence by variable importance $\mathcal{M}_I \leftarrow modelSeq(X, Y, \boldsymbol{B}, 1)$;
4: Generate model sequence by sampling $\mathcal{M}_S \leftarrow modelSeq(X, Y, \boldsymbol{B}, 3)$;
5: Generate model sequence by L_1-logit $\mathcal{M}_L \leftarrow modelSeqL(X, Y, \boldsymbol{B})$;
6: Combine model sequences by set union $\mathcal{M} \leftarrow \mathcal{M}_C \cup \mathcal{M}_I \cup \mathcal{M}_S \cup \mathcal{M}_L$;
7: Sort model list $\mathcal{M}$ by model cost associated with each entry in the list;
8: For each model cost in $\mathcal{M}$, delete $\mathcal{M}$'s entries with higher cost but lower accuracy;
9: Return($\mathcal{M}$);

neither does it have a restriction on the number of model sequences, in the ensemble.

The computational complexity of producing a model sequence by variable importance is calculated as follows. The generation of variable importance by RF costs $O(T \cdot p^{3/2} \cdot n(\log(n))^2)$ (the ranking of variable importance is abosorbed in this term), while that by normalized importance is $O(p)$. Then there is a sequential removal of variables with each round involves the selection or sampling of variables which costs $O(p)$, and the fitting of RF and then applying to the validation set which costs $O(T \cdot p^{1/2} \cdot n(\log(n))^2)$, so this step costs

Algorithm 3 *modelSeq(X, Y, $\boldsymbol{B}$, type)*

Invoke RF with the training and validation set;
Let α_0 be the predictive accuracy on the validation set;
Set variable cost for full model $\beta_0 \leftarrow \sum_{i=1}^{p} b_i$;
Set variable weights $w \leftarrow \boldsymbol{W}_i$ if $type == 1$;
Set variable weights $w \leftarrow 1/\boldsymbol{B}$ if $type == 2$;
Set variable weights $w \leftarrow \boldsymbol{W}_u$ if $type == 3$;
Initialize the list of models $\mathcal{M} \leftarrow (\beta_0, \alpha_0, V)$;
for $i = 1$ to $p - 2$ **do**
  if $type == 3$ **then**
    Sample variable $v \in V$ with weights $\boldsymbol{W}_u$ and set $V \leftarrow V - \{v\}$;
  else
    Let $v \leftarrow \arg\min_{i \in V}\{w[i] : \ i \in V\}$ and set $V \leftarrow V - \{v\}$;
  end if;
  Invoke RF on variables from the set V;
  Let α_V and β_V be the predictive accuracy and model cost;
  Append the new model by set union $\mathcal{M} \leftarrow \mathcal{M} \cup \{(\beta_V, \alpha_V, V)\}$;
end for
Return($\mathcal{M}$);

$O(T \cdot p^{3/2} \cdot n(\log(n))^2)$ in total. Thus the computational cost for the generation of each of these three model sequences, or each invocation of Algorithm 3, is $O(T \cdot p^{3/2} \cdot n(\log(n))^2)$. Algorithm 2 is an invocation of Algorithm 1 and that of Algorithm 3 for 3 times plus some post-processing, thus its computational complexity is $O\left(n_\lambda \cdot max\left(J \cdot E,\ T \cdot p^{1/2} \cdot (\log(n))^2\right) \cdot p \cdot n\right)$. Putting the computational complexity of individual algorithms together gives the overall computational complexity of our algorithm as $O\left(n_\lambda \cdot max\left(J \cdot E,\ T \cdot p^{1/2} \cdot (\log(n))^2\right) \cdot p \cdot n\right)$. It scales linearly, with some additional log-factors, with the the number of data instances n.

4 Optimality and a toy example

Just like any machine learning problem, it is always important to consider optimality—what would be the 'optimal' model schedule for a given data distribution and cost profile? Let us focus on the classification problem. In the following, we will define the optimal model schedule, and give a toy example to demonstrate that our algorithm can deliver a near optimal model schedule.

Assume the data is generated from a distribution (often unknown) in $\mathbb{R}^p \otimes \mathcal{J}$ where $\mathcal{J} = \{1, 2, ..., J\}$ is the set of labels. Assume the cost profile of variables is given by $\boldsymbol{B}$. Let $V = \{1, 2, ..., p\}$ be the set of indices of all the p variables. Then the set of all possible combinations of variables is given by $\mathcal{V} = \{S : \ S \subseteq V \text{ s.t. } |S| > 0\}$. Let $g(S)$ denote the total cost of variables in set S. For a given data, the set $\mathcal{V}$ is finite. Thus, there are only finite possible values for the total cost of variables used in the model; let $\mathcal{C}$ denote the set of possible costs. Then, for a given data distribution, the *optimal model schedule* is defined as the following collection of pairs (here for simplicity we omit such

information as variables used, coefficients etc in the model schedule)

$$\left\{(c,\alpha):\ c\in\mathcal{C}, \alpha=\max_{S\in\mathcal{V},g(S)=c}\{\text{Bayes rate on feature set S}\}\right\}.$$

The above defines the best predictive accuracy for each possible cost level. If a universally consistent classifier, such as AdaBoost with early stop [2], support vector machines [8] etc is used, then the Bayes rate can be achieved on any subset of variables as long as such a subset is visited by the algorithm. Thus, it is desirable to include a universally consistent classifier in the algorithm (RF is used in our algorithm due to its superior empirical performance though its universal consistency is still unknown), and then the remaining issue is to try to hit as many critical points in the solution space as possible. That is the idea of our approach. We will use a toy example to illustrate this.

The toy example we choose is a small scale problem for which the optimal model schedule can be computed by exhaustive search. The data is generated by a 4-component Gaussian mixture in $\mathbb{R}^8$ specified as

$$\frac{1}{4}\sum_{i=1}^{4}\mathcal{N}(\mu_i,\Sigma),$$

where the covariance matrix Σ is defined by

$$\Sigma_{i,j}=\rho^{|i-j|},\quad \text{for } \rho=0.1,0.3,0.6,$$

and the center of the four components are

$$\mu_1=(2.0,1.8,1.6...,0.6),\ \mu_4=-\mu_1,$$

with μ_2 and μ_3 the same as μ_1 except that the second half and the first half of their components are taking an opposite sign. To introduce variety into the underlying data, we let ρ vary over $\{0.1,0.3,0.6\}$. The mixture ID is used as the label for each data instance. The sample size is 50,000 with 60% used for training, 20% for the selection of models in individual model sequences (validation set), and 20% for producing the predictive accuracy by the final model schedule. The sample size is chosen to be large enough so that the predictive error rate stops decreasing when further increasing the sample size. The cost of variables are set as follows (produced by sampling from $[1,100]$ uniformly at random and then stay fixed)

$$92,81,45,23,23,33,72,5.$$

Since there are only 8 variables for this classification problem, we can try all possible (totally 255) combinations of variables. For each combination, the total cost of all involving variables is calculated, and the predictive accuracy by RF is assessed. The optimal model is found by an exhaustive search over all combinations of variables. A similar approach was taken by [23].

Figure 7 shows the model schedule found by exhaustive search and by our algorithm, under different values of ρ. In all cases, the model schedules found

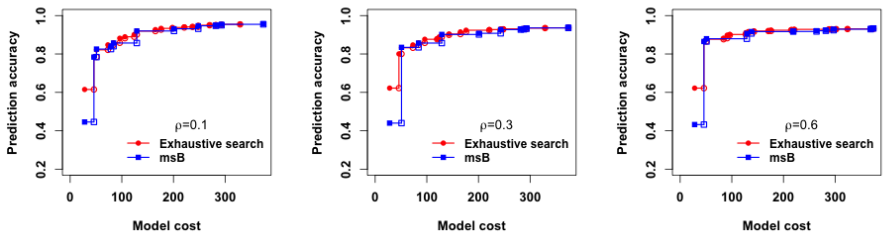

Figure 7: *Model schedule produced by exhaustive search and by msB (our approach) on the Gaussian mixture data.*

by the two are nearly identical. For this problem, the total number of variable combinations, or candidate pairs $(cost, accuracy)$, is about 250 (excluding cases with only one variables for which RF cannot run). We term the collection of all candidate pairs as the *solution space* or model space. Our algorithm only visits a small fraction, about 30/250=12%, of the solution space, but does surprisingly well in producing the model schedule. To uncover the mystery, we plot the solution space, and then mark points visited by our algorithm in Figure 8. Our algorithm is very efficient in that it visits only the most *promising* part, a small fraction, of the solution space. In particular, the two sequences of points, marked as "1-2-3-4-5-6-7" and "a-b-c-d-e-f-g" (produced by selecting variables according to variable cost and variable importance, respectively), almost always stay close to the 'optimal' part of the solution space.

The reason why our algorithm is efficient in finding 'promising' search paths can be understood as follows. Starting at the full model (top right corner in Figure 8), our algorithm successively removes the most expensive variable (by cost) or the weakest variable (by importance), this effectively does a gradient descent in some functional space, i.e., follows the direction along which the model cost decreases the most (i.e., a cost-greedy direction) or accuracy decreases the least, thus the next point visited by our algorithm in the model space will be either a point that is cheaper in model cost but with potentially similar (maybe even better) predictive accuracy (since the weakest variable is removed), or much cheaper (since the most expensive variable is removed) in cost but potentially not much reduction in accuracy. Such moves in the solution space are desirable in reaching an economic model schedule. Of course, other model sequences adopted by our algorithm may potentially correct sub-optimal moves, or throw in some better moves along the way. The overall effect is, by visiting a small part of the solution space we have already seen the 'best' part of the world.

5 Experiments

We conduct a range of experiments on several different aspects. We compare the model schedules generated by our method, denoted by *msB*, and a compet-

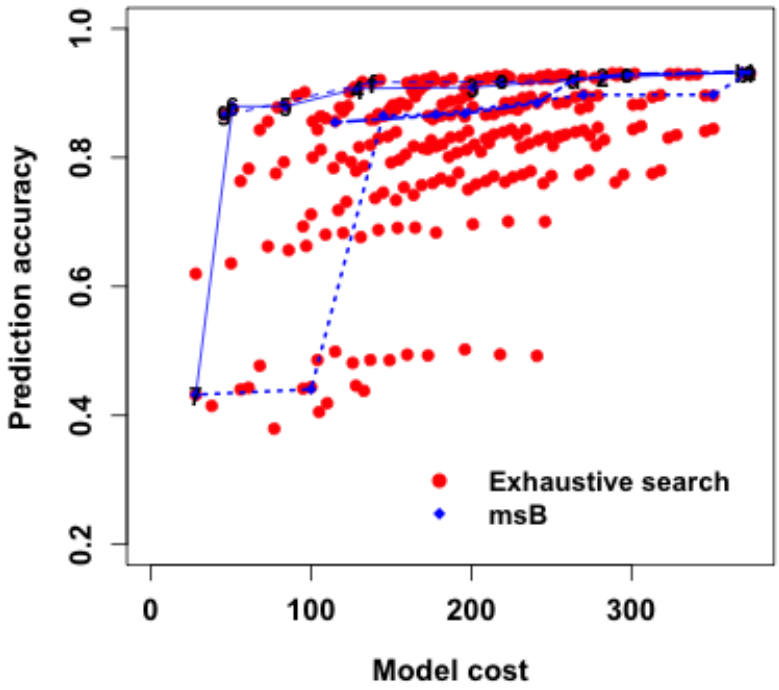

Figure 8: *Points in the solution space that are visited by exhaustive search and by msB. For points in the solution space visited by msB, there are 4 sequences, marked by numbers "1-2-3-4-5-6-7", letters "a-b-c-d-e-f-g", by dotted line, and by two-dash line, corresponding to variable selection by cost, accuracy, sampling, and* L_1*-regularization path, respectively.*

ing method [24], denoted by *logitB*, as that is the only method available that strictly enforces the budget constraint. The comparison is done both visually and numerically under a quantitative metric. We also compare our approach to the method proposed in [40], denoted by *rfB*. As *rfB* does not produce a model schedule, we obtain the total cost, ν, incurred by *rfB* first, and then compute the predictive accuracy achieved by *msB* under budget ν. The computation time is also compared among different methods. We also explore the feasibility of exhaustive search as well as the optimality of our approach by comparing our approach to exhaustive search. We report the results in Section 5.2, Section 5.3, and Section 5.4, respectively. We start by an introduction of the datasets used in our experiments.

5.1 Datasets

We conduct experiments on a number of datasets, including nine from the UC Irvine Machine Learning Repository [19] and an additional remote sensing data adopted from a recent study [36]. The UC Irvine datasets are the *Naval propulsion plants (NPP), Steel plate faults, Spam filter, Concrete compressive strength, Landsat, Thyroid disease, Vehicle silhouette, Bank marketing*, and *US census income (USCI)*. A summary of the datasets is given in Table 1.

Two of the UC Irvine datasets were originally used for regression and we convert the relevant output variable(s) so that they could be used for classification. These are the Naval propulsion plants data and the Concrete compressive strength data. For the former, we treat any record of measurements as requiring maintenance if both the *q3Compressor* and *q3Turbine* variables are above their median values. For the latter, we convert by rounding the compressive strength variable to 4 categories according to its 4 quartiles. The original USCI data has 299,285 instances with 41 features. We follow the preprocessing procedure in [37], and exclude instances with missing values, and also features #26, #27, #28 and #30, due to too many values. This leaves 285,799 instances on 37 features, with all categorical variables converted to integers. For the Bank marketing data, we follow [38] where all features are converted into numeric values and then standardize.

The remote sensing data is about a region, spanning 23°2'-23°25'N, 113°8'-113°35'E, in Guangdong Province of South China. There are 7 different land-use types (classes), including *water, residential area, natural forest, orchard, industry or commercial area, idle land*, and *bareland*. The features were derived from a Landsat Thematic Mapper (TM) image about the region acquired in January 2009. There are totally 56 features, including 6 spectral features corresponding to the 6 TM bands, 8 texture features, *mean, variance, homogeneity, contrast, dissimilarity, entropy, second moment, correlation*, for each of the 6 TM bands, and two location features, the *latitude* and *longitude* of the ground position associated with each data instance.

5.2 Comparison with *logitB*

The experiments are conducted as follows. For all datasets used, a random sample of 60% of the data are used for training, 20% for the selection of models

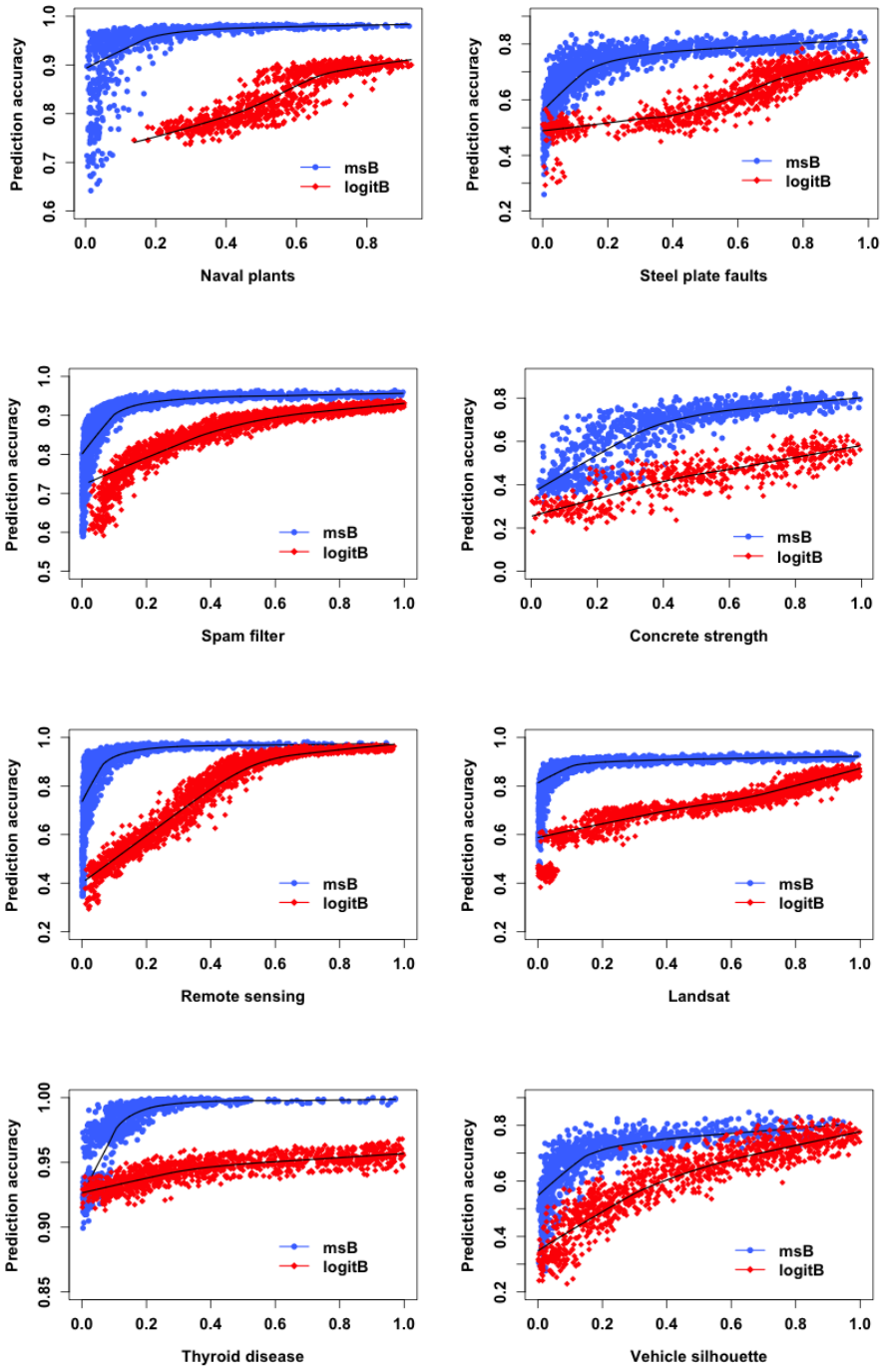

Figure 9: *The expected predictive accuracy by logitB and msB for the 8 datasets used in the our experiment. The x-axis indicates the normalized model cost, which are relative cost with respect to the cost for the full model. msB and logitB indicate our approach and L_1 logistic regression under a budget, respectively.*

Dataset	Features	Classes	Instances
Naval propulsion plants	16	2	11934
Steel plate faults	27	7	1941
Spam filter	57	2	4601
Concrete strength	8	4	1030
Remote sensing	56	7	3303
Landsat	36	6	4435
Thyroid disease	21	3	7200
Vehicle silhouette	18	4	846
Bank marketing	16	2	45211
US censors income	40	2	285779

Table 1: A summary of datasets used in experiments.

in individual model sequence, and 20% for producing the predictive accuracy by the final model schedule. As no dataset used in our experiments comes with a cost profile for its variables, we randomly generate the cost by sampling from $[1, 100]$ uniformly at random. Then a model schedule is generated by *msB* and by *logitB*. This is repeated for 100 runs. As different runs of our experiments are under a different variable cost profile, we normalize the model cost by dividing it by the cost of a full model in the same run. An *average model schedule* is produced by curve smoothing (with *lowess()* function in R) over model schedules generated over the 100 runs.

Our final output over 100 runs is visualized as follows. Instead of using the staircase curve (which would make the plot too crowded for 100 model schedules), we plot individual pairs, *(model cost, predictive accuracy)*, in a model schedule as points in a scatter plot. Then we add the *average model schedule* (indicated by solid curves) to the scatter plot. Figure 9 shows model schedules generated for the first 8 datasets used in our experiment (the other two omitted due to page size limit). It can be seen that, in almost all cases, scatter points generated by our algorithm lie substantially higher above those by *logitB*. This indicates that, for the same normalized model cost, our algorithm could produce a model with substantially higher predictive accuracy. Similarly, the average model schedules produced by our algorithm dominate that by *logitB* by a substantial margin on all datasets used in our experiment.

Beyond visualization, we also produce a single number metric for the cost-accuracy curve for a given model schedule similarly as the AUC (area under the curve) [30]. This is done by scaling the model cost as a proportion of that under the full model, which turns the model cost into a number in the range of $(0, 1]$. Thus, a model schedule curve, after cost normalization, would lie within the unit square defined by $(0, 1] \times (0, 1]$. By calculating the area above the x-axis but under the model schedule curve, we obtain a number in the range of $(0, 1]$. This would be a sensible metric to compare different model schedules, since at any cost level the higher the curve lies, the better the accuracy. We call this metric the *area under the staircase* (AUS), as the model schedule curve takes the shape of a (irregular) staircase. Note that we do not have to restrict

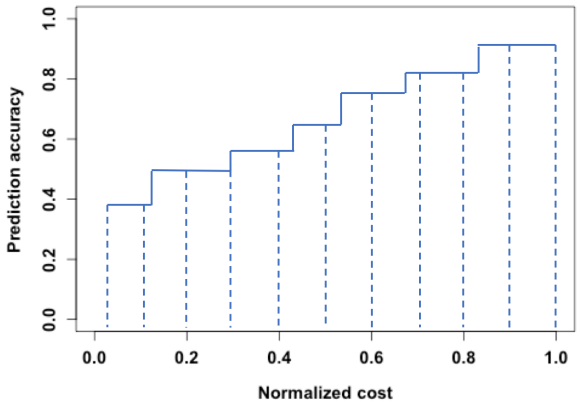

Figure 10: *Illustration of the area under the staircase (dashed area in the figure).*

to the staircase-shaped curve; the curve can be a generic one, for instance the smoothed average model schedule curve over many runs. Figure 10 illustrates the area under a model schedule curve, where we are interested in the size of the shaded area.

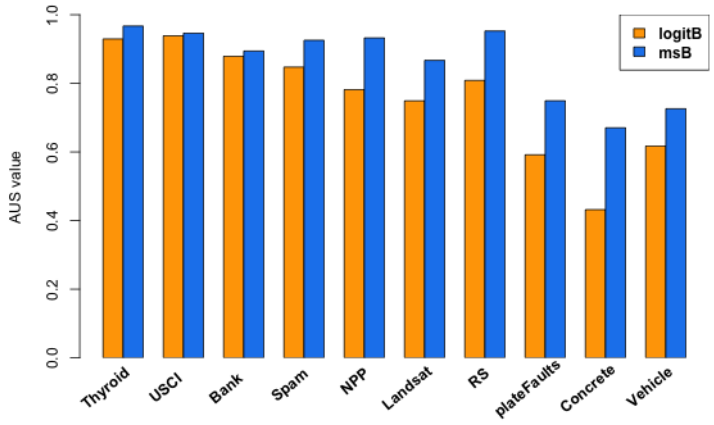

Figure 11: *A comparison of the AUS score for model schedules generated by logitB and msB.*

Figure 11 shows the AUS score computed for all the datasets under *logitB* and *msB*, respectively. We can see that *msB* outperforms *logitB* by a large margin on most of the 10 datasets. We also evaluate the running time for *logitB* and *msB*. The results are shown in Figure 12, and we see that the *logitB* algorithms runs faster than *msB* on all the datasets. This is expected as *logitB* only does part of the work of that of *msB*.

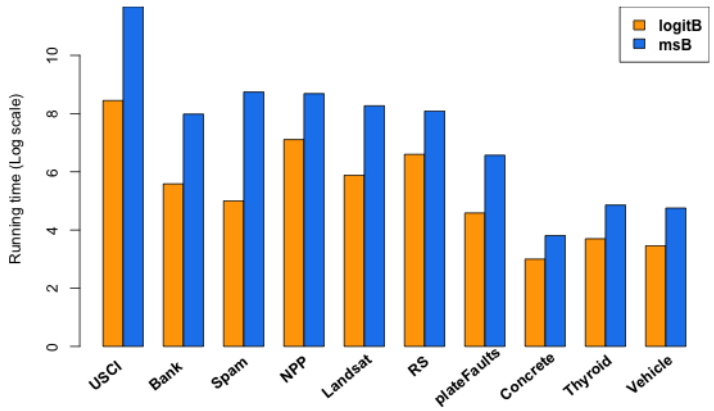

Figure 12: *A comparison of the running time (in* $\log_2$*seconds) for model schedules generation by LogitB and msB.*

5.3 Comparison with *rfB*

The comparison to another competing algorithm, *rfB*, is different. As *rfB* does not produce a model schedule, we choose to compare its predictive accuracy to that by *msB* under similar variable costs. For each dataset, we obtain the total cost, ν, incurred by *rfB* first, and then we compute the predictive accuracy achieved by *msB* under budget ν; the respective predictive accuracies are compared. The results are shown in Figure 13. It can be seen that in most of the cases, *rfB* gives inferior predictive accuracy than *msB* under a similar cost. A limitation with *rfB* is that there is neither a guarantee on the total variable cost nor that on the predictive accuracy. While the resulting model by *rfB* may have a small variable cost, but that is not necessary under the predefined budget. This is because while it aims to produce a low-cost model, but because it achieves this by sampling, the variables enter the model partially by chance. When the predefined budget is high, *rfB* cannot adapt to the high budget to deliver a better predictive accuracy; this can be seen from the often substantial performance gap towards the original RF. As *rfB* tries to give more weights to low-cost variables, rather than the most predictive ones, that hurts the predictive accuracy. Thus even at the same cost, the predictive accuracy it achieves may be substantially lower than that by *msB*. This can be seen from the noticeable gaps in predictive accuracy between *rfB* and *msB*. On the other hand, in several cases, even at a low budget level, *msB* is able to achieve the level of predictive accuracy by RF under no budget constraint.

5.4 Feasibility of exhaustive search

We also explore the feasibility of exhaustive search on three datasets with relatively small number of features, including the Naval propulsion plants (NPP) data, the Concrete compressive data and the Vehicle silhouette data. Table 2 summarizes the results. It can be seen that exhaustive search is only feasi-

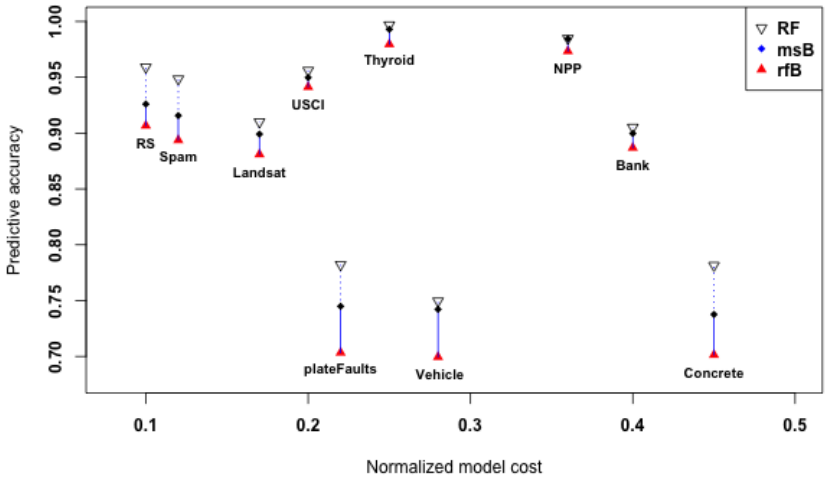

Figure 13: *Comparison of msB to rfB. The predictive accuracy by RF is included to indicate what one would expect without any budget constraint.*

Dataset	**#features**	**Exhaustive search**	**msB**
Concrete	8	28 (0.6975)	14 (0.6904)
NPP	16	105657 (0.9781)	412 (0.9723)
Vehicle	18	36143 (0.7672)	27 (0.7559)

Table 2: A comparison of exhaustive search and *msB* on running time (in seconds) and the resulting AUS score (indicated in the parenthesis).

ble for dataset with a very small number of features. Despite a much shorter running time, *msB* achieves an AUS score very close to that obtained by exhaustive search. Note that due to the excessively long running time, we repeat the exhaustive search on the NPP and Vehicle dataset only for 10 runs.

6 Conclusions

This work tackles the challenging problem of measures or variables selection under a budget (or under different budget levels). We proposed an efficient strategy in navigating the solution space by following multiple model sequences with each having the potential of leading to an 'optimal' solution. Instead of delivering a single model as output, we produce a model schedule which would allow a user to pick the model with the best predictive accuracy under a given budget, or to get the best tradeoff between model cost and predictive accuracy. Experiments on several benchmark or real datasets show that our approach compares favorably to competing methods.

Given the high cost in the collection, storage, processing, and maintenance

of large scale data, methods that incorporate variable costs will be highly desirable and widely applicable. We expect our approach could be generalized to many settings beyond classification. Also, our idea of multi-path search, along with the learning of model power from data and multicore computing has the potential of becoming a general strategy for efficiently finding an approximate solution for a large class of discrete optimization problems.

Acknowledgements

We thank the editors and the anonymous reviewers for their helpful comments and suggestions.

References

[1] H. Akaike. A new look at the statistical model identification. *IEEE Transactions on Automatic Control*, 19(6):716–723, 1974.

[2] P. L. Bartlett and M. Traskin. Adaboost is consistent. *Journal of Machine Learning Research*, 8:2347–2368, 2007.

[3] L. Breiman. Random Forests. *Machine Learning*, 45(1):5–32, 2001.

[4] L. Breiman, J. Friedman, C. J. Stone, and R. A. Olshen. *Classification and Regression Trees*. Chapman and Hall/CRC, 1984.

[5] R. Caruana, N. Karampatziakis, and A. Yessenalina. An empirical evaluation of supervised learning in high dimensions. In *Proceedings of the Twenty-Fifth International Conference on Machine Learning (ICML)*, pages 96–103, 2008.

[6] R. Caruana and A. Niculescu-Mizil. An empirical comparison of supervised learning algorithms. In *Proceedings of the 23rd International Conference on Machine Learning (ICML)*, 2006.

[7] T. Chen and C. Guestrin. XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD*, pages 785–794, 2016.

[8] C. Cortes and V. N. Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.

[9] A. Delaigle, P. Hall, and J. Jin. Robustness and accuracy of methods for high dimensional data analysis based on student's t-statistic. *Journal of Royal Statistical Society, Series B*, 73(3):283–301, 2011.

[10] D. Donoho and J. Jin. Higher criticism thresholding: Optimal feature selection when useful features are rare and weak. *Proceedings of the National Academy of Sciences, U. S. A.*, 105(39):14790–5, 2008.

[11] B. Efron, T. Hastie, I. M. Johnstone, and R. Tibshirani. Least angle regression. *Annals of Statistics*, 32:407–499, 2004.

[12] C. Elkan. The foundations of cost-sensitive learning. In *In Proceedings of the 17th International Joint Conference on Artificial Intelligence*, pages 973–978, 2001.

[13] Y. Freund and R. E. Schapire. Experiments with a new boosting algorithm. In *Proceedings of the 13rd International Conference on Machine Learning (ICML)*, 1996.

[14] J. Friedman. Stochastic gradient boosting. *Computational Statistics and Data Analysis*, 38(4):367–378, 2002.

[15] J. Friedman, T. Hastie, and R. Tibshirani. Regulzrization paths for generalized linear models via coordinate descent. *Journal of Statistical Software*, 33(1):1–22, 2010.

[16] R. Greiner, A. Grove, and D. Roth. Learning cost-sensitive active classifiers. *Artificial Intelligence*, 139(2):137–174, 2002.

[17] I. Guyon and A. Elisseeff. An introduction to variable and feature selection. *Journal of Machine Learning Research*, 3:1157–1182, 2003.

[18] S. Ji and L. Carin. Cost-sensitive feature acquisition and classification. *Pattern Recognition*, 40(5):1474–1485, 2007.

[19] M. Lichman. UC Irvine Machine Learning Repository. http://archive.ics.uci.edu/ml, 2013.

[20] H. Liu and H. Motoda. *Feature Selection for Knowledge Discovery and Data Mining*. Springer, 1998.

[21] D. G. Luenberger. *Linear and Nonlinear Programming*. Springer, 2003.

[22] N. Meinshausen and P. Buhlmann. High-dimensional graphs and variable selection with the lasso. *The Annals of Statistics*, 34(3):1436–1462, 2006.

[23] F. Min, H. He, Y. Qian, and W. Zhu. Test-cost-sensitive attribute reduction. *Information Sciences*, 181(22):4928–4942, 2011.

[24] V. Nagaraju, D. Yan, and L. Fiondella. A framework for selecting a subset of metrics considering cost. In *24th ISSAT International Conference on Reliability and Quality in Design (RQD 2018)*, 2018.

[25] D. B. O'Brien, M. R. Gupta, and R. M. Gray. Cost-sensitive multi-class classiØcation from probability estimates. In *Proceedings of the 25th International Conference on Machine Learning (ICML)*, 2008.

[26] M. Park and T. Hastie. L1-regularization path algorithm for generalized linear models. *Journal of the Royal Statistical Society (B)*, 69(4):659–677, 2007.

[27] H. Peng, F. Long, and C. Ding. Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(8):1226–1238, 2005.

[28] G. E. Schwarz. Estimating the dimension of a model. *Annals of Statistics*, 6(2):461–464, 1978.

[29] V. S. Sheng and C. X. Ling. Thresholding for making classifiers cost-sensitive. In *Proceedings of AAAI*, 2006.

[30] K. A. Spackman. Signal detection theory: Valuable tools for evaluating inductive learning. In *Proceedings of the 6th International Workshop on Machine Learning*, 1989.

[31] J. Tang, S. Alelyani, and H. Liu. Feature selection for classification: A review. In C. C. Aggarwal, editor, *Data Classification: Algorithms and Applications*. Chapman and Hall/CRC, 2014.

[32] R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of Royal Statistics Society (Series B)*, 58(1):267–288, 1996.

[33] H. Wang. Forward regression for ultra-high dimensional variable screening. *Journal of the American Statistical Association*, 104(488):1512–1524, 2009.

[34] K. A. Wang, X. Bian, P. Liu, and D. Yan. DC^2: A divide-and-conquer algorithm for large-scale kernel learning with application to clustering. *arXiv:2931216*, 2019.

[35] X. Wang and C. Leng. High dimensional ordinary least squares projection for screening variables. *Journal of Royal Statistical Society, Series B*, 78(3):589–611, 2016.

[36] D. Yan, C. Li, N. Cong, L. Yu, and P. Gong. A structured approach to the analysis of remote sensing images. *International Journal of Remote Sensing*, 40(20):7874–7897, 2019.

[37] D. Yan, Y. Wang, J. Wang, G. Wu, and H. Wang. Fast communication-efficient spectral clustering over distributed data. *arXiv:1905.01596*, 2019.

[38] D. Yan and Y. Xu. Learning over inherently distributed data. *arXiv:1907.13208*, 2019.

[39] B. Zadrozny, J. Langford, and N. Abe. Cost-sensitive learning by cost-proportionate example weighting. In *Proceedings of IEEE International Conference on Data Mining (ICDM)*, 2003.

[40] Q. Zhou, H. Zhou, and T. Li. Cost-sensitive feature selection using random forest: Selecting low-cost subsets of informative features. *Knowage-Based Systems*, 95:1–11, 2016.

[41] H. Zou and T. Hastie. Regularization and variable selection via the elastic net. *Journal of Royal Statistical Society, Series B*, 67(2):301–320, 2005.