Importance Sampled Learning Ensembles

Jerome H. Friedman
Stanford University
207 PUBLICATIONS **80,413** CITATIONS

Importance Sampled Learning Ensembles

Jerome H. Friedman[*] Bogdan E. Popescu[†]

September 9, 2003

Abstract

Learning a function of many arguments is viewed from the perspective of high–dimensional numerical quadrature. It is shown that many of the popular ensemble learning procedures can be cast in this framework. In particular randomized methods, including bagging and random forests, are seen to correspond to random Monte Carlo integration methods each based on particular importance sampling strategies. Non random boosting methods are seen to correspond to deterministic quasi Monte Carlo integration techniques. This view helps explain some of their properties and suggests modifications to them that can substantially improve their accuracy while dramatically improving computational performance.

Keywords and phrases: learning ensembles, bagging, random forests, boosting, gradient boosting, AdaBoost, MART

1 Introduction

Approximating a function of many arguments is central to supervised learning problems in data mining, machine learning, and pattern recognition. The goal is to predict (estimate) likely values $\hat{y}$ for the unknown value y of a particular attribute (variable) associated with a system under study, given the known joint values of other attributes $\mathbf{x} = (x_1, x_2, \cdots, x_n)$ also associated with that system. The prediction rule can be regarded as defining a function $\hat{y} = F(\mathbf{x})$ that maps points $\mathbf{x}$ in the space χ of all joint values, $\mathbf{x} \in \chi$, to values in the corresponding space of y–values, $(y, \hat{y}) \in Y$.

Defining a loss $L(y, \hat{y})$ that characterizes the cost of predicting a value $\hat{y}$ when the true value is y, the lack of quality of the prediction rule is often taken to be the average or expected loss ("risk") over all future predictions

$$R(F) = E_{q_t(\mathbf{z})} L(y, F(\mathbf{x})) \tag{1}$$

where $\mathbf{z} = (\mathbf{x}, y)$ and $q_t(\mathbf{z})$ is is the joint distribution of future ("test") attribute values. In this case the optimal "target" function is the one that minimizes prediction risk

$$F^*(\mathbf{x}) = \arg\min_{F(\mathbf{x})} E_{q_t(\mathbf{z})} L(y, F(\mathbf{x})). \tag{2}$$

[*]Department of Statistics and Stanford Linear Accelerator Center, Stanford University, Stanford, CA 94305 (jhf@stanford.edu)

[†]Department of Statistics, Stanford University, Stanford, CA 94305 (bogdan@stat.stanford.edu)

In the predictive learning problem one is given a collection of previously solved cases $\{\mathbf{z}_i = (\mathbf{x}_i, y_i)\}_1^N$ where for each individual case (observation) the values of all of the attributes have been observed. The prediction rule is constructed from this "learning" data

$$\hat{y} = \hat{F}(\mathbf{x}) = F\left(\mathbf{x};\, \{\mathbf{z}_i\}_1^N\right).$$

The learning observations are often regarded as being a random sample drawn from some probability distribution $q_l(\mathbf{z})$, where $q_l(\mathbf{z}) \simeq q_t(\mathbf{z})$ (1). Situations where $q_l(\mathbf{z}) \neq q_t(\mathbf{z})$ are referred to as "concept drift" in the machine learning literature.

When $q_l(\mathbf{z}) = q_t(\mathbf{z}) = q(\mathbf{z})$ the learning problem can be viewed as estimating $F^*(\mathbf{x})$ (2) given its approximate value

$$y_i = F^*(\mathbf{x}_i) + \varepsilon_i$$

at a set of randomly realized $\{\mathbf{x}_i\}_1^N$. Here ε_i is a random variable with probability distribution $\varepsilon_i \sim q(y_i - F^*(\mathbf{x}_i) \,|\, \mathbf{x}_i)$. The "distance" criterion to be minimized is

$$D(F, F^*) = E_{q(\mathbf{z})}\left[L(y, F(\mathbf{x})) - L(y, F^*(\mathbf{x}))\right]. \tag{3}$$

2 Approximating Model

There are a great many methods proposed for predictive learning. (See for example, Hastie, Tibshirani, and Friedman 2001.) Here the focus in on linear prediction models of the form

$$F(\mathbf{x}; a) = a_0 + \int_P a(\mathbf{p})\, f(\mathbf{x}; \mathbf{p})\, d\mathbf{p}, \tag{4}$$

where $f(\mathbf{x}; \mathbf{p})$ ("base learner") is a specified (often simple) function of the predicting variables $\mathbf{x}$, characterized by a set of parameters $\mathbf{p} = (p_1, p_2, \cdots p_K)$. A specific set of joint parameter values $\mathbf{p} \in P$ indexes a particular function of $\mathbf{x}$ from the set of all such functions $\{f(\mathbf{x}; \mathbf{p})\}_{\mathbf{p} \in P}$, and $a(\mathbf{p})$ is its corresponding coefficient in the linear model (4). Thus, given a particular base learner $f(\mathbf{x}; \mathbf{p})$, the coefficient function $a(\mathbf{p})$ (along with the intercept parameter a_0) specifies the predicting model $F(\mathbf{x}; a)$. From (3) the optimal coefficient function is given by

$$a^* = \arg\min_a E_{q(\mathbf{z})} L(y, F(\mathbf{x}; a)) \tag{5}$$

and the corresponding function of $\mathbf{x}$, $F(\mathbf{x}; a^*)$, is the optimal one for prediction among those in the class of functions defined by the base learner through (4). If the target function $F^*(\mathbf{x})$ (2) resides in this class then $F^*(\mathbf{x}) = F(\mathbf{x}; a^*)$; otherwise $F(\mathbf{x}; a^*)$ represents the closest function to $F^*(\mathbf{x})$ in this class with distance defined by (3).

Popular base learners include sigmoids of linear combinations

$$f(\mathbf{x}; \mathbf{p}) = \left[1 + \exp(\mathbf{p}^t \mathbf{x})\right]^{-1} \tag{6}$$

used in neural networks (Rumelhart, Hinton and Williams 1985), multivariate spline functions where the parameters are the knot locations on the respective variables (Friedman 1991), and decision trees (Breiman, Friedman, Olshen and Stone 1981, Quinlan 1991) where the parameters are the splitting variables and values that define the partition of the predictor variable space χ, and the values assigned to the terminal nodes.

3 Numerical quadrature

Computing $F(\mathbf{x}; a)$ (4) for any given $\mathbf{x}$ requires the evaluation of a multidimensional integral of the form

$$\int_P I(\mathbf{p})\, d\mathbf{p}, \tag{7}$$

where here the integrand is $I(\mathbf{p}) = a(\mathbf{p})\, f(\mathbf{x}; \mathbf{p})$. It is unlikely that this integral can be represented by an easily computable closed form. Thus numerical quadrature rules must be employed. Numerical quadrature approximates (7) by

$$\int_P I(\mathbf{p})\, d\mathbf{p} \simeq \sum_{m=1}^{M} w_m I(\mathbf{p}_m) \tag{8}$$

where $I(\mathbf{p}_m)$ is the integrand evaluated at $\mathbf{p}_m \in P$, and w_m is a corresponding weight applied to the evaluation.

Specific quadrature rules are defined by choice of evaluation points $\{\mathbf{p}_m\}_1^M$ and corresponding weights $\{w_m\}_1^M$. For (4) the quadrature rule becomes

$$F(\mathbf{x}; a) \simeq a_0 + \sum_{m=1}^{M} w_m\, a(\mathbf{p}_m)\, f(\mathbf{x}; \mathbf{p}_m) = \tilde{F}\left(\mathbf{x}; \{c_m\}_0^M\right) = c_0 + \sum_{m=1}^{M} c_m\, f(\mathbf{x}; \mathbf{p}_m) \tag{9}$$

with $c_0 = a_0$ and $c_m = w_m\, a(\mathbf{p}_m)$, since w_m and $a(\mathbf{p}_m)$ are not separately identifiable. For a given set of evaluation points $\{\mathbf{p}_m\}_1^M$ the optimal coefficient values, averaged over all $\mathbf{x}$ values, are (from (3)) given by

$$\{c_m^*\}_0^M = \arg\min_{\{c_m\}_0^M} E_{q(\mathbf{z})} L\left(y, \tilde{F}\left(\mathbf{x}; \{c_m\}_0^M\right)\right). \tag{10}$$

Thus, knowing the joint distribution $q(\mathbf{z})$, the problem becomes one of choosing a good set of evaluation points $\{\mathbf{p}_m\}_1^M$ for constructing the approximating quadrature rule (9). Of course $q(\mathbf{z})$ is not known; only a random sample $\{\mathbf{z}_i\}_1^N$ drawn from this distribution is available. This complication is addressed in Section 5. For now we only consider the idealized (computational) problem of choosing a good quadrature rule given the joint distribution $q(\mathbf{z})$.

4 Monte Carlo methods

Most useful base learners $f(\mathbf{x}; \mathbf{p})$ (4) involve several to many parameters. Thus, the integration problem (7) is high dimensional. Among the most successful high dimensional quadrature rules are those based on Monte Carlo methods. The set of evaluation points $\{\mathbf{p}_m\}_1^M$ are randomly drawn from some probability distribution $r(\mathbf{p})$ defined on the parameter space $\mathbf{p} \in P$. This distribution defines the Monte Carlo integration rule. The simplest rule chooses $r(\mathbf{p})$ to be constant so that any point $\mathbf{p} \in P$ is equally likely to be selected at each of the M draws. Other choices for $r(\mathbf{p})$ define alternative Monte Carlo rules; the best choice as judged by (3) will depend on the target function $F^*(\mathbf{x})$ (2).

4.1 Importance sampling

The goal of importance sampling Monte Carlo is to increase accuracy by using information known about a particular problem to choose a good sampling probability distribution $r(\mathbf{p})$ for that problem. The cardinality M of the selected points is generally a small fraction of all $\mathbf{p} \in P$ so that judicious choices for $\{\mathbf{p}_m\}_1^M$ can often dramatically increase accuracy over simple constant probability sampling. The challenge is to identify what information is known, or easily obtained, about the problem at hand and then using it to construct an appropriate sampling distribution $r(\mathbf{p})$. To be effective, this distribution should assign relatively high mass to points in the parameter space $\mathbf{p} \in P$ that are more relevant or important to achieving a high accuracy integration rule when used with a relatively small number of other evaluation points, and with the coefficients (weights) given by (10).

In general, the incremental value of any selected point $\mathbf{p}_m$ for reducing approximation error (3) will depend on the other points included with it in the quadrature rule. Therefore, importance values should be assigned to groups of points taken together as being proportional to the accuracy of the rule based on those points. Unfortunately, even approximately determining such group importances, for all possible groupings, is very difficult to do in practice for any given problem. Approximate sequential sampling strategies, based on estimating the importance of a single point $\mathbf{p}_m$ conditional on the previously selected points $\{\mathbf{p}_{m'}\}_{m'<m}$ can sometimes be employed. This is discussed in Section 7.

We begin by considering the simplest importance sampling strategies for which the importance or relevance of an evaluation point $\mathbf{p}_m$ is estimated without regard to the other points that might be used with it in the integration rule. As discussed in Section 6, this is the strategy used by bagging (Breiman 1996a), random forests (Ho 1995 and Breiman 2001), and Bayesian model averaging (Denison, Holmes, Mallik and Smith 2002, Chipman, George and McCullagh 1998). Although less than ideal, such strategies can often result in dramatic gains in accuracy over naive constant probability sampling.

4.2 Partial importance

Given only a single potential evaluation point $\mathbf{p} \in P$, without knowledge of the other points that will be used with it in the integration rule, one measure of its lack of relevance is

$$J(\mathbf{p}) = \min_{\alpha_0,\alpha} E_{q(\mathbf{z})} L(y,\ \alpha_0 + \alpha\, f(\mathbf{x},\mathbf{p})). \tag{11}$$

This is the prediction risk of using $\mathbf{p}$ alone in a single point ($M = 1$) rule (9). The optimal single point rule is obtained by using

$$\mathbf{p}^* = \arg\min_{\mathbf{p}\in P} J(\mathbf{p}). \tag{12}$$

Although this optimal single point rule is unlikely to be as accurate as one involving many evaluation points, it is commonly used in practice. This is the case, for example, when a single decision tree or logistic regression is employed.

This measure (11) is clearly an imperfect reflection of the (inverse) value of a point when it is used together with other points in a quadrature rule. It is possible for a particular point $\mathbf{p}$ to have a small value of $J(\mathbf{p})$ but still contribute little toward increasing accuracy in the presence of a given set of other evaluation points. Although perhaps less likely, it is also possible for $J(\mathbf{p})$ to be relatively large for evaluation points that can significantly aid in increasing accuracy when used in conjunction with particular sets of other points. However, (11) represents an easily computable importance surrogate. The assumption (hope) is that a collection of such evaluation points $\{\mathbf{p}_m\}_1^M$, each with relatively small values of $J(\mathbf{p}_m)$, will result in a integration rule with higher accuracy than with either a similar sized collection of points sampled with constant probability, or the best single point rule (12). As evidenced by the success of bagging and random forests, this is often the case in practice.

4.3 Partial importance sampling

In order to apply importance sampling Monte Carlo based on $J(\mathbf{p})$ (11), one must derive a sampling probability density $r(\mathbf{p})$ that gives higher probability to points $\mathbf{p} \in P$ with smaller values of $J(\mathbf{p})$. That is

$$r(\mathbf{p}) = g(J(\mathbf{p})) \tag{13}$$

with $g(\cdot)$ a monotonically decreasing function of its argument. This density will be centered at or near $\mathbf{p}^*$ (12) and have correspondingly decreasing values for points $\mathbf{p}$ increasingly distant from $\mathbf{p}^*$. Here the relevant distance measure is

$$d(\mathbf{p}, \mathbf{p}^*) = J(\mathbf{p}) - J(\mathbf{p}^*). \tag{14}$$

In addition to its location, it is well known that a critical parameter of any probability density used for importance sampling is its characteristic scale

$$\sigma = \int_P d(\mathbf{p}, \mathbf{p}^*)\, r(\mathbf{p})\, d\mathbf{p}. \tag{15}$$

If σ is too large then too few points of high importance will be sampled, thereby decreasing the accuracy of the resulting integration rule. In the extreme limit of very large values of σ the result will be similar to that of naive constant probability Monte Carlo. If the value of σ is too small then each sampled point provides little additional information beyond that provided by the other nearby sampled points used with it. In the extreme limit of $\sigma = 0$, $r(\mathbf{p}) = \delta(\mathbf{p} - \mathbf{p}^*)$, where $\delta(\cdot)$ is the Dirac delta function. In this case the integration rule will have the same accuracy as that of the best single point ($M = 1$) rule (12) regardless of the number of sampled points actually used. In any particular problem there is an unknown optimal value for σ. It will depend on M and the detailed nature of the problem at hand, namely the joint distribution of the attributes $q(\mathbf{z})$ and the choice of base learner $f(\mathbf{x}; \mathbf{p})$.

These concepts are illustrated in the context of simple importance sampling Monte Carlo integration in Fig. 1. The upper (black) curve represents a function $F(p)$ to be integrated over the real line $-\infty < p < \infty$, and the lower (colored) curves represent several potential sampling distributions $r(p)$. Each distribution is characterized by a respective location and scale. The almost constant (lower orange) curve is located near the maximum of $F(p)$ but has a very large scale thereby placing only a small fraction of sampled points where the integrand is large (important). The (blue) curve to the right has the proper scale but is not located near the important region of $F(p)$, thereby also producing a small fraction of important evaluation points. The narrow (red) distribution is centered near the maximum of $F(p)$ and produces all of its evaluation points where $F(p)$ is large. However, its small scale causes very few points to be sampled in other regions where $F(p)$ also assumes substantial

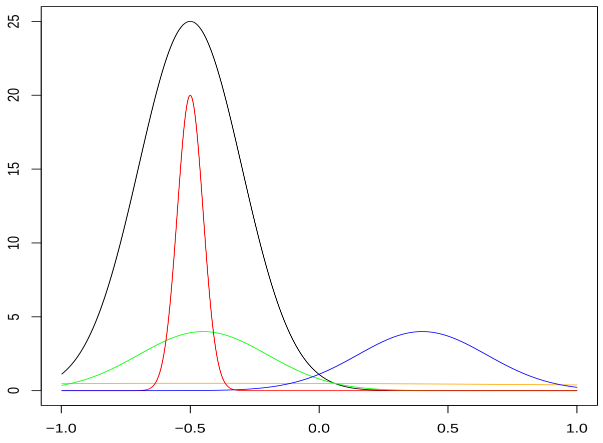


Figure 1: Various importance sampling distributions (lower colored curves) for integrating the function represented by the upper (black) curve. The (blue) curve with location and scale similar to that of the integrand represents the most efficient among these importance sampling distributions.

values, thereby leading to an inaccurate result. The broader (green) distribution has both the proper location and scale for integrating $F(p)$ by covering the entire region where $F(p)$ assumes substantial values while producing relatively few points in other regions. It thereby represents the most efficient sampling distribution for this particular problem.

Although very simple, the example shown in Fig. 1 illustrates the concepts relevant to important sampling Monte Carlo. Both the location and scale of the parameter sampling distribution $r(\mathbf{p})$ must be chosen appropriately so as to place most of its mass in all regions where the integrand realizes important values, and relatively small mass elsewhere. For partial importance sampling (11) (13) the parameter sampling distribution $r(\mathbf{p})$ is located near the optimal single point rule $\mathbf{p}^*$ (12). Its scale σ (15) is determined by the choice of $g(\cdot)$ (13) which in turn is controlled by the details of the sampling strategy as discussed below.

Choosing the width (15) of $r(\mathbf{p})$ is related to the empirically observed trade-off between the "strength" of the individual learners and the correlations among them (Breiman 2001). The strength of a base learner $f(\mathbf{x}, \mathbf{p})$ is directly related to its partial importance (inversely related to its lack of importance $J(\mathbf{p})$ (11)). An ensemble of strong base learners $\{f(\mathbf{x}, \mathbf{p}_m)\}_1^M$, all of which have $J(\mathbf{p}_m) \simeq J(\mathbf{p}^*)$ is well known to perform poorly, as does an ensemble consisting of all very weak learners $J(\mathbf{p}_m) \gg J(\mathbf{p}^*)$. Best results are obtained with ensembles of moderately strong base learners whose predictions are not very highly correlated with each other.

A narrow sampling distribution (small σ (15)), produces base learners $\{f(\mathbf{x}, \mathbf{p}_m)\}_1^M$ all of which having similar strength to the strongest one $f(\mathbf{x}, \mathbf{p}^*)$ (12), and thereby yielding similar highly correlated predictions. This is analogous to the narrow (red) curve in Fig. 1.

A very broad $r(\mathbf{p})$ (large σ) produces highly diverse base learners, most of which have larger values of $J(\mathbf{p})$ (smaller strength) and less correlated predictions. This corresponds to the almost constant lower (orange) curve in Fig. 1. Thus, finding a good strength-correlation trade-off for a learning ensemble can be understood in terms of seeking an appropriate width for an importance sampling parameter distribution (13) (15). As seen above, the best width and thus best strength-correlation trade-off point, will depend on the particular problem at hand as characterized by its true underlying target function $F(\mathbf{x}^*)$ (2).

4.4 Perturbation sampling

For any given problem finding, and then sampling from, an appropriate density $r(\mathbf{p})$ on the parameter space $\mathbf{p} \in P$ is usually quite difficult. However, there is a simple trick that can indirectly approximate this process. The idea is to repeatedly modify or perturb in a random way some aspect of the problem, and then find the optimal single point rule (12) for each of the modified problems. The collection of these single point solutions is then used for the quadrature rule. The characteristic width σ (15) of the corresponding sampling distribution $r(\mathbf{p})$ is controlled by the size of the perturbations. Aspects of the problem that can be modified are the loss criterion $L(y, \hat{y})$, the joint distribution of the variables $q(\mathbf{z})$, and the specifics of the algorithm used to solve the optimization problem (12).

For example, one could repeatedly perturb the loss criterion by

$$L_m(y, \hat{y}) = L(y, \hat{y}) + \eta \cdot l_m(y, \hat{y}) \tag{16}$$

where each $l_m(y, \hat{y})$ is a different randomly constructed function of its arguments. The sampled points are obtained by

$$\left\{ \mathbf{p}_m = \arg \min_{\alpha_0, \alpha, \mathbf{p} \in P} E_{q(\mathbf{z})} L_m(y,\ \alpha_0 + \alpha f(\mathbf{x}; \mathbf{p})) \right\}_1^M . \tag{17}$$

Alternately, one could repeatedly modify the argument to the loss function in a random way, for example taking

$$\left\{ \mathbf{p}_m = \arg \min_{\alpha_0, \alpha, \mathbf{p} \in P} E_{q(\mathbf{z})} L(y,\ \alpha_0 + \alpha f(\mathbf{x}; \mathbf{p}) + \eta g_m(\mathbf{x})) \right\}_1^M . \tag{18}$$

Here each $g_m(\mathbf{x})$ is a different randomly constructed function of $\mathbf{x}$. In both cases, the expected size of the perturbations is controlled by the value of η which thereby indirectly controls the width σ of the corresponding sampling distribution $r(\mathbf{p})$.

Another approach is to repeatedly modify the joint distribution of the variables by random reweighting

$$q_m(\mathbf{z}) = q(\mathbf{z}) \cdot [w_m(\mathbf{z})]^\eta \tag{19}$$

where each $w_m(\mathbf{z})$ is a different randomly constructed (weighting) function of $\mathbf{z}$. The collection of sampled points $\{\mathbf{p}_m\}_1^M$ are the respective solutions to

$$\left\{ \mathbf{p}_m = \arg \min_{\alpha_0, \alpha, \mathbf{p} \in P} E_{q_m(\mathbf{z})} L(y, \alpha_0 + \alpha f(\mathbf{x}; \mathbf{p})) \right\}_1^M . \tag{20}$$

Again the width σ of the corresponding sampling distribution $r(\mathbf{p})$ is controlled by the value chosen for η.

Another method for perturbation sampling involves directly modifying in a random way various aspects of the algorithm used to solve (12). The collection of outputs $\{\mathbf{p}_m\}_1^M$ from each of these randomly perturbed algorithms is then used in the quadrature rule (9) (10). Most algorithms for solving (12) iteratively perform repeated partial optimizations over subsets of the parameters as they converge to a solution. At each iteration the solution values for each of these internal partial optimizations are replaced by alternative randomly chosen values. The degree to which these alternative values are constrained to be close to their corresponding optimal solution values (indirectly) controls the width σ of the sampling distribution $r(\mathbf{p})$. For example, Dietterich 2000 replaces the optimal split at each recursive step of decision tree construction by a random selection among the k best splits. Here the value chosen for k controls the width σ. Breiman 2001 chooses the optimal split among a randomly chosen subset of the predictor variables. The size of this subset inversely controls the degree of the perturbations, thereby regulating the width of $r(\mathbf{p})$.

The implicit goal of all of these perturbation techniques is to produce a good set of parameter points $\mathbf{p} \in P$ to be used with a quadrature rule (9) (10) for approximating the optimal predictor $F(\mathbf{x}; a^*)$ (4) (5). Their guiding principle is the notion of partial importance sampling (11) (13), which produces a parameter sampling distribution $r(\mathbf{p})$ that puts higher mass on points $\mathbf{p}$ that tend to have smaller risk when used in a Monte Carlo integration rule to evaluate (9). A critical meta–parameter of such procedures is the characteristic width σ (15) of the indirectly induced sampling distribution $r(\mathbf{p})$. This aspect is discussed in detail below.

5 Finite data

The discussion so far has presumed that the joint distribution $q(\mathbf{z})$ of the variables $\mathbf{z} = (\mathbf{x}, y)$ is known, and the expected values in (10) (11) can be calculated. As noted in Section 1, this is not the case in practice. Instead one has a collection of previously solved cases $\{\mathbf{z}_i\}_1^N$ drawn from that distribution representing an empirical point mass approximation

$$\hat{q}(\mathbf{z}) = \frac{1}{N} \sum_{i=1}^{N} \delta(\mathbf{z} - \mathbf{z}_i). \tag{21}$$

This must serve as a surrogate for the unknown population distribution $q(\mathbf{z})$. Because $\hat{q}(\mathbf{z}) \neq q(\mathbf{z})$, complications emerge when the goal is to approximate the target function $F^*(\mathbf{x})$ (2) by $F(\mathbf{x}, a^*)$ (4) (5), since both are based on the population distribution $q(\mathbf{z})$.

5.1 Sampling

One such complication involves the sampling procedure. The partial importance $J(\mathbf{p})$ (11) of a parameter point $\mathbf{p} \in P$ is unknown and must be approximated by

$$\hat{J}(\mathbf{p}) = \min_{\alpha_0, \alpha} E_{\hat{q}(\mathbf{z})} L(y, \alpha_0 + \alpha f(\mathbf{x}; \mathbf{p})) \tag{22}$$

with the corresponding optimal single point rule (12) approximated by

$$\hat{\mathbf{p}} = \arg \min_{\mathbf{p} \in P} \hat{J}(\mathbf{p}). \tag{23}$$

Equivalently, perturbation sampling (Section 4.4) must be based on the empirical distribution $\hat{q}(\mathbf{z})$ (21) rather than $q(\mathbf{z})$, yielding an empirical parameter sampling distribution $\hat{r}(\mathbf{p})$.

Implementation of perturbation sampling (Section 4.4) on the data distribution (21) is straight forward. The strategies corresponding to randomizing the loss criterion or random algorithm modification are unaltered. One simply substitutes $\hat{q}(\mathbf{z})$ for $q(\mathbf{z})$ respectively in (17), (18), or the distribution to which an randomized algorithm is applied. Modification of the joint distribution (19) can be accomplished by randomly weighting the observations

$$\hat{q}_m(\mathbf{z}) = \sum_{i=1}^{N} w_{im} \cdot \delta(\mathbf{z} - \mathbf{z}_i) \tag{24}$$

where $\{w_{im}\}_{i=1}^{N}$ are randomly drawn from some probability distribution with $E(w_{im}) = 1/N$. For a given sample size N, the scale (coefficient of variation) of the weight distribution controls the characteristic width $\hat{\sigma}$ (25) of the parameter sampling distribution $\hat{r}(\mathbf{p})$.

To the extent that the parameter estimates $\hat{\mathbf{p}}$ (23) approach their corresponding population values $\mathbf{p}^*$ (12) as the sample size N increases, the characteristic width

$$\hat{\sigma} = \int_P [\hat{J}(\mathbf{p}) - \hat{J}(\hat{\mathbf{p}})]\, \hat{r}(\mathbf{p})\, \mathbf{dp} \tag{25}$$

of $\hat{r}(\mathbf{p})$ will become smaller with increasing N for a given coefficient of variation of the weights. Therefore as $N \to \infty$, $\hat{r}(\mathbf{p}) \to \delta(\mathbf{p} - \mathbf{p}^*)$, and the Monte Carlo integration rule reduces to that of the optimal single point ($M = 1$) rule. As noted in Section 4.3, this is unlikely to produce the best results except in the unlikely event that the target function $F^*(\mathbf{x})$ (2) is a member of the set of base learners; that is $F^*(\mathbf{x}) = f(\mathbf{x}; \mathbf{p}^*)$. This suggests that the coefficient of variation of the weights should increase with increasing sample size in order to maintain the width $\hat{\sigma}$ of the corresponding parameter sampling distribution $\hat{r}(p)$.

From a computational perspective, the most efficient random weighting strategy (24) is to restrict the weight values to $w_{im} \in \{0, 1/K\}$, where $K \leq N$ is the number of non–zero weight values. This can be accomplished by randomly selecting K observations, without replacement from the collection of all N observations, to have non–zero weights. The coefficient of variation of all weight values is

$$C(\{w_{im}\}_{i=1}^{N}) = (N/K - 1)^{1/2}, \tag{26}$$

so that, for a given sample of size N, reducing K increases the characteristic $\hat{\sigma}$ of $\hat{r}(\mathbf{p})$ while also reducing computation by a factor of K/N. Also, as discussed in the preceding paragraph, the value of K should grow more slowly than linearly with increasing N in order to maintain the width $\hat{\sigma}$ of the corresponding parameter sampling distribution $\hat{r}(\mathbf{p})$.

5.2 Quadrature coefficients

A second problem caused by finite data is determining the quadrature coefficients $\{c_m\}_0^M$ (9) given a set of evaluation points $\{\mathbf{p}_m\}_1^M$. The optimal coefficient values are given by (10). These are based on the unknown population distribution $q(\mathbf{z})$ and therefore must be estimated by using the empirical distribution $\hat{q}(\mathbf{z})$ (21). This is a standard linear regression problem where y is the response (outcome) variable and the "predictors" are the base learners $\{f(\mathbf{x};\mathbf{p}_m)\}_1^M$ corresponding to the selected parameter evaluation points.

There are a wide variety of linear regression procedures, many of which take the form

$$\{\hat{c}_m\}_0^M = \arg\min_{\{c_m\}_0^M} \frac{1}{N}\sum_{i=1}^N L\left(y_i,\ c_0+\sum_{m=1}^M c_m f(\mathbf{x}_i;\mathbf{p}_m)\right) + \lambda\cdot\sum_{m=1}^M h(|\, c_m - c_m^{(0)}\,|). \quad (27)$$

The first term in (27) is the empirical risk on the learning data obtained by simply substituting $\hat{q}(\mathbf{z})$ (21) for $q(\mathbf{z})$ in (10). By including the second "regularization" term, one attempts to increase accuracy through reduced variance of the estimated coefficient values by "shrinking" them towards a prespecified set of values $\{c_m^{(0)}\}_1^M$. This is accomplished by choosing the function $h(\cdot)$ to be a monotone increasing function of its argument; usually

$$h(u) = u^\gamma,\ \gamma \geq 0. \quad (28)$$

Popular choices for the value of γ are $\gamma = 2$ ("ridge regression", Hoerl and Kannard 1971), and $\gamma = 1$ ("ShureShrink", Donoho and Johnstone 1996 and "Lasso", Tibshirani 1996). The amount of shrinkage is regulated by $\lambda \geq 0$ which is a model selection "meta" parameter of the regression procedure. Its value is chosen to minimize estimated future prediction risk. The most popular choice for the shrinkage targets is $\{c_m^{(0)} = 0\}_1^M$, reflecting prior lack of knowledge concerning the signs of the population coefficient values $\{c_m^*\}_1^M$ (10).

In the present application, the shrinkage implicit in (27) has the important function of reducing *bias* of the coefficient estimates as well as variance. The importance sampling procedure selects predictors $f(\mathbf{x};\mathbf{p}_m)$ that have low *empirical* risk $\hat{J}(\mathbf{p})$ (22) and thereby preferentially high values for their empirical partial regression coefficients

$$\hat{\alpha}_m = \arg\min_{\alpha_0,\alpha} E_{\hat{q}(\mathbf{z})} L(y, \alpha_0 + \alpha f(\mathbf{x};\mathbf{p}_m)). \quad (29)$$

If the multiple regression coefficients $\{\hat{c}_m\}_1^M$ were estimated using data independent of that used to select $\{\mathbf{p}_m\}_1^M$, then using (27) with $\lambda = 0$ would produce unbiased estimates. However, using the same data for selection and estimation produces coefficient estimates that are biased towards high absolute values (see Copas 1983). The shrinkage provided by (27) for $\lambda > 0$ helps compensate for this bias.

The value chosen for the exponent γ in the penalty function (28) reflects one's prior belief concerning the relative values of $\{|\, c_m^* - c_m^{(0)}\,|\}_1^M$. Higher values for γ produce more accurate estimates to the extent these absolute differences have equal values. For $\{c_m^{(0)} = 0\}_1^M$, this implies that all of the $f(\mathbf{x};\mathbf{p}_m)$ are equally relevant predictors. When this is not the case, smaller values of γ will give rise to higher accuracy. In the context of partial importance sampling (Section 4.3), the dispersion of importances among the respective $f(\mathbf{x};\mathbf{p}_m)$ is roughly controlled by the characteristic width $\hat{\sigma}$ (25) of the empirical parameter sampling distribution $\hat{r}(\mathbf{p})$. Thus, there is a connection between choice of regularizer $h(u)$ in (27) and the sampling strategy used to select the predictors $\{f(\mathbf{x};\mathbf{p}_m)\}_1^M$.

6 ISLE

The previous sections outline in general terms the ingredients comprising an importance sampled learning ensemble (ISLE). They consist of an importance sampling strategy (Section 4.3), implemented by perturbation sampling (Section 4.4), on the empirical data distribution (Section 5.1) to select a set of parameter evaluation points $\{\mathbf{p}_m\}_1^M$ for a numerical integration quadrature rule. The corresponding coefficients (weights) $\{\hat{c}_m\}_1^M$ assigned to each of the evaluation points are calculated by a regularized linear regression (27) using the corresponding selected base learners $\{f(\mathbf{x};\mathbf{p}_m)\}_1^M$ as predictor variables. In this section we show how several popular ensemble learning methods can be cast within this framework. This in turn helps explain some of their properties and suggests improvements to them that can (often dramatically) increase both their statistical and especially their computational properties.

6.1 Bagging

Bagging (Breiman 1996a) is an elegantly simple ensemble learning method. Each member of the learning ensemble is obtained by fitting the base learner to a different booststrap sample (Efron and Tibshirani 1993) drawn from the data. Ensemble predictions are taken to be the average of those of the individually fitted base learners. In many applications bagged ensembles achieve considerably lower prediction risk than the single estimated optimal base learner (23). This is especially the case when the base learners are decision trees (Breiman *et al* 1983, Quinlan 1992).

The learning ensemble produced by bagging is clearly an ISLE. The parameter evaluation points $\{\mathbf{p}_m\}_1^M$ are obtained by (20), substituting (24) for $q_m(\mathbf{z})$ with the weights $\{w_{im}\}_{i=1}^N$ independently drawn from a multinomial distribution $w_{im} \in \{0, 1, \cdots, N\}$ with probability $1/N$. The quadrature coefficients are obtained from (27) with $\{c_m^{(0)} = 1/M\}_1^M$ and $\lambda = \infty$, thereby reducing the quadrature rule (9) to being a simple average.

When viewed form this perspective some of the properties of bagged ensembles are easily understood. As with any ISLE, one should not expect substantial gains in accuracy for base learners that are linear in the parameters

$$f(\mathbf{x}, \mathbf{p}) = p_0 + \sum p_j x_j,$$

since from (4) the ensemble spans the same space of functions as does the base learner, and thus does not enlarge that space.

It is also apparent that for any base learner $f(\mathbf{x};\mathbf{p})$, the learning capacity of a bagged ensemble does not increase with its size M. This is a consequence of using $\lambda = \infty$ in (27). The quadrature regression coefficients are set to prespecified constants rather than being fit to the data. Thus bagging does not "over fit" the data when M is arbitrarily increased. In fact, the learning capacity of a bagged ensemble is generally less than that of the single best base learner (23) since the individual ensemble members (20) are optimized on $\hat{q}_m(\mathbf{z})$ (24) rather than $\hat{q}(\mathbf{z})$ (21).

Viewing bagging as an ISLE suggests several straight forward modifications that might improve its performance. First, the use of bootstrap sampling induces a particular expected level of perturbation on the learning sample. The expected coefficient of variation of the bootstrap observation weights is

$$E\, C_B(\{w_{im}\}_{i=1}^N) = (1 - 1/N)^{1/2}. \tag{30}$$

Comparing this with (26), one sees that bagging with bootstrap samples is roughly equivalent to half–sampling without replacement. This in turn indirectly specifies a particular characteristic width $\hat{\sigma}$ (25) of the parameter sampling distribution $\hat{r}(\mathbf{p})$. As discussed in Section 5.1, this width will decrease with increasing sample size N. It is not clear that the particular width induced by bagging is optimal for any given situation. Other perturbation strategies that induce different characteristic widths might perform better. Second, the total regularization implied by $\lambda = \infty$ in (27) may not be optimal. Setting $\lambda < \infty$ and thereby allowing some level of coefficient fitting to the data may improve performance.

6.2 Random forests

Random forests (Breiman 2001) represents a hybrid approach involving random modifications to both the joint distribution $\hat{q}_m(\mathbf{z})$ (24) and the algorithm used to solve (22) (23). The base learners $f(\mathbf{x}, \mathbf{p})$ are taken to be decision trees grown to largest possible size with no pruning. As with bagging, each member of the learning ensemble is obtained by constructing such a tree on a different bootstrap sample drawn from the data. In addition, at each successive split during tree construction the optimal splitting variable is chosen from a randomly chosen subset of the predictor variables rather than from all of them. Ensemble predictions are taken to be the average of those over all of the individual trees.

Including the random splitting has the effect of increasing the characteristic width $\hat{\sigma}$ (25) of the parameter sampling distribution $\hat{r}(\mathbf{p})$ beyond that obtained by using bagging alone. Here the parameters are the identities of the splitting variables and the splitting values. The degree of increase in $\hat{\sigma}$ is inversely controlled by the size n_s of the selected variable subset at each split. Breiman 1998 suggests choosing $n_s = \lfloor \log_2(n) + 1 \rfloor$, where n is the total number of predictor variables. Subset splitting has the additional effect of reducing computation by a factor of n_s/n.

Like a bagged ensemble, a random forest is clearly an ISLE sharing the same basic properties discussed in Section 6.1. In particular, its learning capacity (less than bagged trees of the same size) does not increase with its size M (number of trees). Its distinguishing characteristic is a broader parameter sampling distribution $\hat{r}(\mathbf{p})$ controlled by the number of variables n_s used to determine each split. Since the optimal width $\hat{\sigma}$ (25) is problem dependent, the relative accuracy of the two approaches will depend on the problem at hand. Also, random forests are amenable to the same potential improvements discussed in the last paragraph of Section 6.1. In addition to adjusting n_s, one can as well control $\hat{\sigma}$ by altering the data sampling strategy. Also setting $\lambda < \infty$ in (27), allowing the tree weights to be (partially) fit to the data, may improve performance.

6.3 Bayesian model averaging

Bayesian methods produce a parameter sampling distribution $\hat{r}(\mathbf{p})$ by proposing a sampling model $\Pr(\{\mathbf{z}_i\}_1^N \mid \mathbf{p})$ for the data $\{\mathbf{z}_i\}_1^N$ given a parameter evaluation point $\mathbf{p}$, and a prior distribution $\Pr(\mathbf{p})$ for the parameters $\mathbf{p} \in P$. The parameter sampling distribution is taken to be the corresponding posterior distribution of $\mathbf{p}$ given the data

$$\hat{r}(\mathbf{p}) = \Pr(\mathbf{p} \mid \{\mathbf{z}_i\}_1^N) = \frac{\Pr(\{\mathbf{z}_i\}_1^N \mid \mathbf{p}) \Pr(\mathbf{p})}{\int_P \Pr(\{\mathbf{z}_i\}_1^N \mid \mathbf{p}) \Pr(\mathbf{p})\, d\mathbf{p}}. \tag{31}$$

The sampled points $\{\mathbf{p}_m\}_1^M$ are intended to be independent draws from this distribution. In practice this is approximated by Markov chain Monte Carlo methods which use a sequence of dependent draws that converge to a stationary distribution identical to $\hat{r}(\mathbf{p})$ (31). The realized set of parameter points produce a corresponding ensemble of base learners

$\{f(\mathbf{x};\mathbf{p}_m)\}_1^M$; ensemble predictions are taken to be the average of those of the corresponding individual base learners.

This approach is similar to that of partial importance sampling described in Section 4.3 provided that the chosen data sampling model $\Pr(\{\mathbf{z}_i\}_1^N \,|\, \mathbf{p})$ is a monotone deceasing function of $\hat{J}(\mathbf{p})$ (22). For example with squared–error loss, and motivated by Gaussian homoscedastic errors, one might choose

$$\Pr(\{\mathbf{z}_i\}_1^N \,|\, \mathbf{p}) \backsim \exp(-N\hat{J}(\mathbf{p})/2\delta^2) \tag{32}$$

where

$$\hat{J}(\mathbf{p}) = \frac{1}{N}\sum_{i=1}^{N}(y_i - f(\mathbf{x}_i;\mathbf{p}))^2 \tag{33}$$

and δ^2 is the variance of the irreducible error.

The Bayesian approach incorporates additional flexibility by allowing the specification of a prior distribution $\Pr(\mathbf{p})$ to incorporate knowledge outside the data concerning the potential value of a parameter point $\mathbf{p}$ to the accuracy of the integration rule. However, this can be equivalently accomplished in the perturbation sampling framework (Section 4.4) by appropriately modifying the original loss function $L(y, f(\mathbf{x};\mathbf{p}))$.

The characteristic width $\hat{\sigma}$ (25) of the parameter sampling distribution $\hat{r}(\mathbf{p})$ induced by Bayesian model averaging decreases with increasing sample size N. The posterior distribution (31) places increasing mass close to its maximum located at $\mathbf{p} = \mathbf{p}^*$ as N becomes larger, approaching $\hat{r}(\mathbf{p}) = \delta(\mathbf{p} - \mathbf{p}^*)$ as $N \to \infty$, as illustrated in (32) (33). This is a consequence of defining the data sampling distribution $\Pr(\{\mathbf{z}_i\}_1^N \,|\, \mathbf{p})$ in terms of a single parameter point $\mathbf{p}$. This assumes that the target function $F^*(\mathbf{x})$ (2) is one of the base learners and $\hat{r}(\mathbf{p})$ (31) is interpreted as the posterior probability that $f(\mathbf{x};\mathbf{p}) = F^*(\mathbf{x}) = f(\mathbf{x};\mathbf{p}^*)$. When this is not the case, modifying the Bayesian model averaging approach along the lines discussed above for bagging and random forests may lead to improved performance.

7 Sequential ISLE

The ensembles of base learners produced by the techniques described in Section 6 can be obtained in a *parallel* manner; each individual learner is trained without any information about the others. Fitting the final coefficients (10) (27) in the post–processing stage of the ISLE takes into account the fact that the entire ensemble is being used for prediction. However, information about the other members of the ensemble can as well be used in the first stage of generating the base learners themselves.

7.1 Sequential sampling

Pure Monte Carlo techniques involving random sampling are not the only methods used for high dimensional quadrature. So called “quasi”–Monte Carlo methods attempt to evaluate (8) by constructing sets of evaluation points $\{\mathbf{p}_m\}_1^M$ in a deterministic manner (see Stroud 1971). The goal is to *jointly* locate parameter points $\mathbf{p}_m \in P$ to produce an accurate quadrature rule. Thus, importance is assigned to groups of points taken together, rather than separately to individually sampled points. In this approach the location of each evaluation point will depend on the locations of the other points to be used with it in the integration rule.

In the context of learning ensembles, the lack of importance of a set of parameter evaluation points $\{\mathbf{p}_m\}_1^M$ is

$$J(\{\mathbf{p}_m\}_1^M) = \min_{\{\alpha_l\}_0^M} E_{q(\mathbf{z})} L\left(y, \alpha_0 + \sum_{l=1}^{M} \alpha_l\, f(\mathbf{x}, \mathbf{p}_l)\right).$$

As noted in Section 4.1, this is difficult to evaluate for all possible sets of evaluation points. A more computable approximation is sequential sampling in which the relevance of each successive individual evaluation point is judged in the context of the (fixed) previously sampled points. In the interest of further reducing computation, this can be approximated by

$$J_m(\mathbf{p} \mid \{\mathbf{p}_l\}_1^{m-1}) = \min_{\alpha_0, \alpha} E_{q(\mathbf{z})} L\left(y, \alpha_0 + \alpha f(\mathbf{x}, \mathbf{p}) + \sum_{l=1}^{m-1} \alpha_l f(\mathbf{x}, \mathbf{p}_l)\right) \tag{34}$$

where the values of the coefficients $\{\alpha_l\}_1^{m-1}$ are each fixed at their minimizing value of (34) when their corresponding point $\mathbf{p}_l$ was entered. Each sequentially selected parameter evaluation point $\mathbf{p}_m$ is taken to be the solution to

$$\mathbf{p}_m = \arg\min_{\mathbf{p} \in P} J_m(\mathbf{p} \mid \{\mathbf{p}_l\}_1^{m-1}). \tag{35}$$

This criterion (34) is similar to (18) with $\eta = 1$ and

$$g_m(\mathbf{x}) = \sum_{l=1}^{m-1} \alpha_l f(\mathbf{x}, \mathbf{p}_l). \tag{36}$$

Here however each $g_m(\mathbf{x})$ is not a random function, but rather is deterministically generated by the sequential sampling process (34) (35). The characteristic width σ of this (non–random) sampling distribution is given by (15) with $r(\mathbf{p})$ being a set of point masses at the corresponding selected evaluation points. It will tend to increase with the number M of sampled points since the successive loss criteria (18) (36) increasingly differ from (11) as the sampling proceeds. This causes the respective sampled points $\mathbf{p}_m$ to be increasingly distant from the first one $\mathbf{p}_1 = \mathbf{p}^*$ (12) where distance is given by (14). For a given M, the width σ (15) can be controlled by choosing values for η in (18) (36) other than $\eta = 1$ (34). As seen in Sections 9 and 10, smaller values $\eta << 1$ generally provide superior performance in the context of finite samples. Implementation of sequential sampling on finite data is straight forward. One simply substitutes the empirical data distribution $\hat{q}(\mathbf{z})$ (21) in place of the population distribution $q(\mathbf{z})$ in (34), or its analog with $\eta < 1$ (18), (36).

7.2 Boosting

A sequentially sampled learning ensemble can be viewed as an ISLE in which the parameter evaluation points $\{\mathbf{p}_m\}_1^M$ are chosen deterministically through (18) (36) rather than independently by random sampling (Section 4.4). There is a connection between the sequential sampling strategy described here and boosting.

AdaBoost (Freund and Schapire 1996) produces a learning ensemble $\{f(\mathbf{x}, \mathbf{p}_m)\}_1^M$ by repeatedly solving (34) (35) using a exponential loss criterion $L(y, \hat{y}) = \exp(-y \cdot \hat{y})$ for $y \in \{-1, 1\}$ (Breiman 1997, Schapire and Singer 1998, and Friedman, Hastie and Tibshirani

2000). In the finite data context, this translates into a sequential strategy where the data distribution is deterministically perturbed. At step m, the modified data distribution is given by (24) with

$$w_{im} = e^{-y_i \cdot \hat{g}_m(\mathbf{x}_i)},$$

where

$$\hat{g}_m(\mathbf{x}) = \sum_{l=1}^{m-1} \hat{\alpha}_l \cdot f(\mathbf{x}, \mathbf{p}_l), \quad m = 1, .., M.$$

Here $\hat{\alpha}_l$ is the data estimate of α_l (36) at each sequential step. Ensemble predictions are taken to be

$$F_M(\mathbf{x}) = sign\left(\hat{\alpha}_0 + \sum_{l=1}^{M} \hat{\alpha}_l \cdot f(\mathbf{x}; \mathbf{p}_l)\right).$$

Gradient boosting (MART, Friedman 2001) approximately solves (18) (36) using a small value for η ("shrinkage parameter"). Each base learner is fit to the gradient of the corresponding loss criterion by least–squares. The respective coefficients are taken to be $\tilde{\alpha}_m = \eta \cdot \hat{\alpha}_m$. Instead of modifying the data distribution by reweighting, this implementation sequentially modifies the loss function in a deterministic manner. The ensemble predictions are taken to be

$$F_M(\mathbf{x}) = \hat{\alpha}_0 + \sum_{l=1}^{M} \tilde{\alpha}_l \cdot f(\mathbf{x}; \mathbf{p}_l).$$

As discussed in Hastie, Tibshirani, and Friedman 2001, this methodology can be viewed as approximately solving (27) (28), with $\gamma = 1$ and $\{c_m^{(0)} = 0\}_1^M$, where $\{\mathbf{p}_m\}_1^M$ represent all realizable parameter values ($M \lesssim \infty$).

7.3 Hybrid strategies

From a purely computational perspective the strategy outlined in Section 7.1 is less efficient than the random sampling without replacement strategy (Section 5.1, last paragraph) since all of the data are used to compute each parameter evaluation point $\mathbf{p}_m$. In Sections 9 and 10, hybrid strategies that combine these two approaches are seen to achieve comparable accuracy to sequential sampling alone, while substantially reducing computation. In generating the base learners, a hybrid ISLE will solve at step m

$$\hat{\mathbf{p}}_m = \arg \min_{\alpha_0, \alpha, \mathbf{p} \in P} E_{\hat{q}_m(\mathbf{z})} L_m(y, \alpha_0 + \alpha \cdot f(\mathbf{x}, \mathbf{p})), \tag{37}$$

where $\hat{q}_m(\mathbf{z})$ is obtained, at each step by sampling *without replacement* ν% of the data, and L_m is defined as before, through (18) (36). For fixed number M of base learners generated, the dispersion of the ensemble increases when the fraction ν of observations sampled decreases (see (26)). Another parameter that controls the degree of dispersion is η (18), (36) which controls the rate of change of L_m throughout the sequence; the dispersion decreases as η decreases. Thus, to maintain a given dispersion, the value of η should be reduced as ν is decreased to gain computational efficiency. This is illustrated in Sections 9 and 10. Further perturbation and increase in computational performance can be obtained by randomizing the algorithm used to solve (37) as well (e.g also sampling the input variables as in Section 6.2).

8 Post-processing algorithms

The second stage of any ISLE involves estimating the final quadrature coefficients through a regularized regression, as described in Section 5.2. Various algorithms correspond to different loss functions L and penalties h in (27). The empirical evidence suggests that, in finite data context, the best ISLEs (computational performance as well as accuracy) are obtained by generating diverse ensembles with a large characteristic width $\hat{\sigma}$, and using an l_1 (lasso) penalty that shrinks the coefficients of the base learners towards zero:

$$\{\hat{c}_m(\lambda)\}_0^M = \arg\min_{\{c_m\}_0^M} \frac{1}{N}\sum_{i=1}^{N} L\left(y_i, c_0 + \sum_{m=1}^{M} c_m f(\mathbf{x}_i; \mathbf{p}_m)\right) + \lambda \cdot \sum_{m=1}^{M} |\, c_m \,|. \tag{38}$$

By generating an ensemble with large characteristic width ($\hat{\sigma}$), the individual predictors (base learners) for the post-processing stage will be highly diverse and the ensemble will thereby contain relevant as well as many non relevant predictors (base learners). As noted in Donoho et al. 1995 and Tibshirani 1996, the l_1 penalty is especially appropriate in such situations. The amount of regularization (λ) should be chosen using a separate test set (or cross validation); therefore the *entire coefficient paths* may need to be estimated (i.e. all the values of $\{\hat{c}_m(\lambda)\}_0^M$ corresponding to all $\lambda \in (0, \infty)$ in (38)). Since the post–processing stage involves solving large optimization problems (one usually deals with hundreds or thousands of predictors (base learners) and tens to hundreds of thousands observations), fast algorithms are a necessity.

For regression problems where the outcome y assumes numeric values and the loss criterion is squared error

$$L(y, \hat{y}) = (y - \hat{y})^2, \tag{39}$$

the minimization (38) is approximated in the interest of *computational efficiency* by the incremental forward-stagewise approach described in Hastie, Tibshirani, and Friedman 2001, Section 10.12.2. (see also Efron et al. 2002). There exist updating strategies that make this forward-stagewise algorithm very fast. The determination of the *optimal* coefficients in (38) for the loss (39) involves estimating the optimal value for λ. With the forward–stagewise algorithm this can be done in less time than that required to compute the coefficients in a *single* least squares fit. Details and the actual algorithm are provided in Friedman and Popescu 2003.

Sections 9 and 10 present results for ISLEs that use, besides the least–squared loss (39), other loss criteria for robust regression and classification.
The Huber-M loss (Huber 1964)

$$L(y, F) = \begin{cases} \frac{1}{2}(y - F)^2, & |y - F| \leq \delta \\ \delta(|y - F| - \delta/2), & |y - F| > \delta \end{cases} \tag{40}$$

is a compromise between the squared-error loss (39) and least absolute deviation loss $L(y, \hat{y}) = |y - \hat{y}|$. The values of the "transition" point δ in (40) differentiates the residuals that are treated as "outliers" being subject to the absolute loss, from the other residuals that are subject to the squared error loss. The parameter δ is chosen adaptively as the forward–stagewise algorithm proceeds, in the same manner as in Friedman 2001. A computationally efficient forward–stagewise algorithm for the loss (40) is described in Friedman and Popescu 2003 and Popescu 2003.

For binary classification, ISLE algorithms used in Sections 9 and 10 are based on the loss function:

$$L(y, F) = [y - R(F)]^2 \tag{41}$$

where $y \in \{-1, 1\}$ and $R(F)$ is a *ramp* function that truncates the values of F at ± 1,

$$R(F) = \max(-1, \min(1, F)).$$

The population minimizer of this squared–error *ramp loss* (41) is

$$F^*(\mathbf{x}) = 2 \cdot P(y = 1|\mathbf{x}) - 1.$$

This loss (41) is attractive both in terms of the computational properties of the derived algorithm and the robustness against output noise (label misspecification). In the absence of label misspecification it compares favorably, in terms of accuracy, with other loss functions commonly used for classification. Details are discussed in Friedman and Popescu 2003 and Popescu 2003.

9 Simulation study

As discussed in the previous sections, there are a wide variety of methods for producing importance sampled learning ensembles. Additionally, these various approaches can be combined to produce hybrid strategies in an attempt to improve accuracy and/or reduce computation. As with any learning method, there are also a variety of choices for base learners $f(\mathbf{x}; \mathbf{p})$, loss criteria $L(y, \hat{y})$, and regularization methods for estimating the quadrature coefficients (27) (28). In this section we attempt to gain some understanding into the characteristics of several such approaches through a systematic simulation study. Applications to actual data are illustrated in Section 10.

In all applications presented here the base learners $f(\mathbf{x}; \mathbf{p})$ are taken to be decision trees (Breiman *et al*, 1983). For *any* loss criterion L the *sequential* ISLEs used in Sections 9 and 10 are built with the methodology described in Section 7 by fitting at each step a *regression* tree to the gradient of this loss, evaluated at the current model:

$$\hat{\mathbf{p}}_m = \arg \min_{\alpha_0, \alpha, \mathbf{p} \in P} E_{\hat{q}_m(\mathbf{z})} (\tilde{y}_m - \alpha_0 - \alpha \cdot f(\mathbf{x}, \mathbf{p}))^2, \tag{42}$$

where

$$\tilde{y}_m = - \left[\frac{\partial L(y, F(\mathbf{x}))}{\partial F(\mathbf{x})} \right]_{F(\mathbf{x}) = \eta g_m(\mathbf{x})},$$

with $g_m(\mathbf{x})$ by (36). This procedure can be obtained from (18), (36) following a derivation in Friedman 2001.

We first focus on (regression) problems where the outcome variable y assumes numeric values $y \in R$ and the loss criterion is squared–error (39). Robust regression results are shown in Section 10.1 in a real data context. Binary classification, $y \in \{-1, 1\}$, is treated in Sections 9.6 (simulation context) and 10.2 (real data). The quadrature coefficients are estimated by approximately solving (38) using the computationally efficient forward–stagewise algorithms mentioned in Section 8 and described in Friedman and Popescu 2003.

In order to gauge the value of any estimation method it is necessary to accurately evaluate its performance over many different situations. This is most conveniently accomplished through Monte Carlo simulation where data can be generated according to a wide variety of prescriptions, and resulting performance accurately calculated.

9.1 Random function generator

One of the most important characteristics of any problem affecting performance is the true underlying target function $F^*(\mathbf{x})$ (2). Every method has particular targets for which it is most appropriate and others for which it is not. Since the nature of the target function can vary greatly over different problems, and is seldom known, we compare the merits of several learning ensemble methods on a variety of different randomly generated targets (Friedman 2001). Each target function takes the form

$$F^*(\mathbf{x}) = \sum_{l=1}^{20} a_l h_l(\mathbf{x}_l). \tag{43}$$

The coefficients $\{a_l\}_1^{20}$ are randomly generated from a uniform distribution $a_l \backsim U[-1, 1]$. Each $h_l(\mathbf{x}_l)$ is a function of a randomly selected subset, of size n_l, of the n–input variables $\mathbf{x}$. Specifically,

$$\mathbf{x}_l = \{x_{P_l(j)}\}_{j=1}^{n_l}$$

where each P_l is a separate random permutation of the integers $\{1, 2, \cdots, n\}$. The size of each subset n_l is itself taken to be random, $n_l = \lfloor 1.5 + r \rfloor$, with r being drawn from an exponential distribution with mean $\lambda = 2$. Thus, the expected number of input variables for each $h_l(\mathbf{x}_l)$ is between three and four; however, more often there will be fewer than that, and somewhat less often, more. This reflects a bias against strong very high order interaction effects. However, for any realized $F^*(\mathbf{x})$ there is a good chance that at least a few of the 20 functions $h_l(\mathbf{x}_l)$ will involve higher order interactions. In any case, $F^*(\mathbf{x})$ will be a function of all, or nearly all, of the input variables.

Each $h_l(\mathbf{x}_l)$ is an n_l–dimensional Gaussian function

$$h_l(\mathbf{x}_l) = \exp(-\frac{1}{2}((\mathbf{x}_l - \mu_l)^T \mathbf{V}_l^{-1}(\mathbf{x}_l - \mu_l)), \tag{44}$$

where each of the mean vectors $\{\mu_l\}_1^{20}$ is randomly generated from the same distribution as that of the input variables $\mathbf{x}$. The $n_l \times n_l$ covariance matrix $\mathbf{V}_l$ is also randomly generated. Specifically,

$$\mathbf{V}_l = \mathbf{U}_l \mathbf{D}_l \mathbf{U}_l^T$$

where $\mathbf{U}_l$ is a random orthonormal matrix (uniform on Haar measure) and $\mathbf{D}_l = diag\{d_{1l} \cdots d_{n_l l}\}$. The square–roots of the eigenvalues are randomly generated from a uniform distribution $\sqrt{d_{jl}} \backsim U[a, b]$, where the limits a, b depend on the distribution of the input variables $\mathbf{x}$.

For all of the studies presented here the number of variables input to $F^*(\mathbf{x})$ was ten, and their joint distribution was taken to be standard normal $\mathbf{x} \backsim N(0, \mathbf{I})$. The eigenvalue limits were $a = 0.1$ and $b = 2.0$.

In the simulation studies below, 100 target functions $F^*(\mathbf{x})$ were randomly generated according to the above prescription (43) (44). This approach allows a wide variety of quite different target functions to be generated in terms of the shapes of their contours in the ten–dimensional input space. Although lower order interactions are favored, these functions are not especially well suited to additive regression trees. Decision trees produce tensor product basis functions, and the components $h_l(\mathbf{x}_l)$ of the targets $F^*(\mathbf{x})$ are not tensor product functions.

For each of the 100 target functions, data sets $\{\mathbf{x}_i, y_i\}_1^N$ of size N were randomly generated. Each $\mathbf{x}_i$ was generated from a $n = 40$ dimensional standard normal distribution. The corresponding responses y_i were obtained from

$$y_i = F^*(\mathbf{x}'_i) + \varepsilon_i \tag{45}$$

where $\mathbf{x}' = (x_1, ..., x_{10})$ represents the first ten input variables. Thus, there are 30 "noise" variables unrelated to the response y. Each ε_i is generated from a normal distribution with variance chosen to produce a one–to–one signal–to–noise ratio, $var\{\varepsilon_i\}_1^N = var\{F^*(\mathbf{x}_i)\}_1^N$.

9.2 Regression

For each data set $\{\mathbf{x}_i, y_i\}_1^N$, the performance is evaluated in terms of the standardized root–mean–square error in approximating its respective target function $F_l^*, l = 1, ..., 100$

$$e_{RMS}(\hat{F}_{jl}) = \left(\frac{E_{\mathbf{x}}(F_l^*(\mathbf{x}) - \hat{F}_{jl}(\mathbf{x}))^2}{Var_{\mathbf{x}}(F_l^*(\mathbf{x}))} \right)^{1/2}, \tag{46}$$

where $\hat{F}_{jl}(\mathbf{x})$ is the approximating function obtained by applying a method j under consideration to the data corresponding to $F_l^*(\mathbf{x})$ (45). Datasets with $N = 10,000$ observations were used in training the ensembles and (46) was estimated with independent test sets of 10,000 observations.

Different methods can be evaluated by comparing their respective distributions of (46) obtained over the 100 data sets. The comparative root–mean–square error

$$e_{CRMS}(\hat{F}_{jl}) = e_{RMS}(\hat{F}_{jl}) / min_k e_{RMS}(\hat{F}_{kl}) \tag{47}$$

facilitates comparison between methods. It is the ratio of the error (46) of a given algorithm j to the error of the best algorithm being compared with it on a particular data set. Therefore, the best method receives a value of 1.0 and the others will have larger values. An ensemble of five hundred trees was generated for each method. For the post–processing

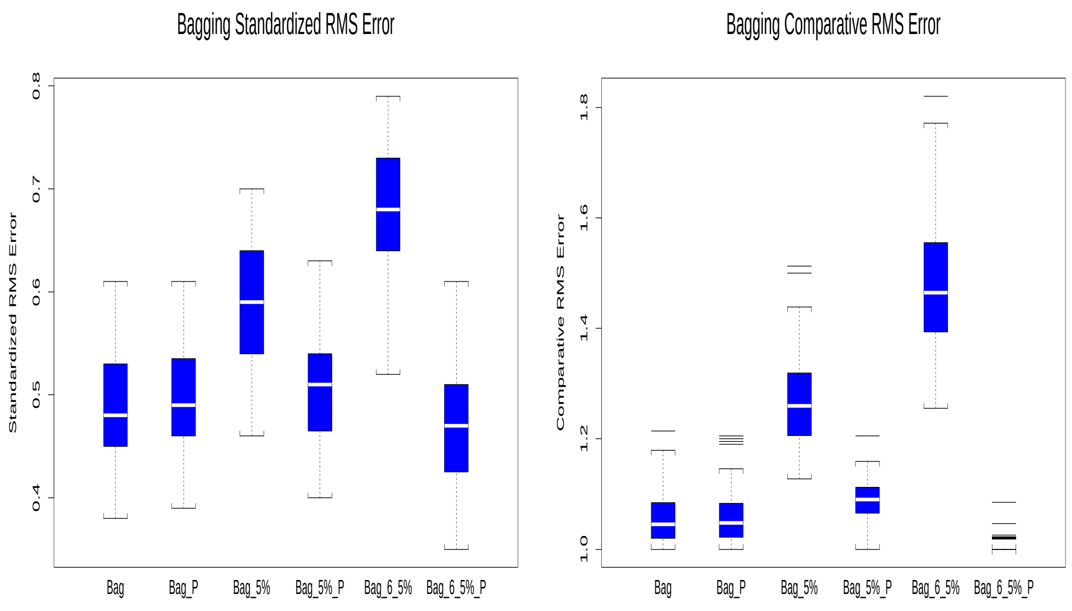


Figure 2: Distributions of standardized (left) and comparative (right) root mean squared error for several bagged and corresponding post–processed bagged ensembles (suffix "P") over the 100 data sets in the regression Monte Carlo study. Post-processing provides the greatest improvement for the most diverse ensembles based on the weakest base learners.

stage, as well as for all sequential ensembles a separate set, *from the training data*, is left aside for model selection. The resulting distributions of (46) (47) over the 100 data sets for each method are summarized by boxplots.

9.3 Bagged ISLEs

Figure 2 displays results for various bagged tree ensembles and post-processed bagged ensembles. One can see, in the plot of relative root–mean–square error (46) distributions (left panel), that the 100 targets represent a fairly wide spectrum of difficulty for all the methods. The right panel shows the comparative RMS error (47) among these methods. Bagging large trees (fully grown on the 10,000 observation bootstrap training set, labelled as *Bag*) does better than bagging trees built on small 5% samples without replacement (labelled as *Bag_5%*), and much better than bagging shallow (six terminal node) trees built on 5% samples (*Bag_6_5%*). For these methods the ensemble prediction is taken to be the average of the individual base learner (tree) predictions. The respective adjacent distributions in Fig. 2 with labels ending in suffix "*P*" represent results for the corresponding method using the estimated optimal quadrature coefficients (38) ("post–processing"). These results show that with post–processing the relative ranking of the corresponding ensembles is reversed. Using the optimal estimated quadrature weights transforms the poorest performing ensemble (*Bag_6_5%*) to being the best. Comparing (26) with (30), one sees that an ensemble of trees, each constructed on randomly drawn 5% samples, is far more diverse (larger $\hat{\sigma}$

(25)) than the trees built on bootstrap samples. It will thus contain many trees that are not highly relevant to predicting the outcome variable y. It will also likely contain some trees that are highly relevant. Simple averaging gives all trees equal weight so that the less relevant trees dilute the effect of the more relevant ones. Post–processing the ensemble with an l_1 penalty (38) tends to assign larger coefficients (weights) to the relevant predictors (trees) thereby mitigating this dilution effect.

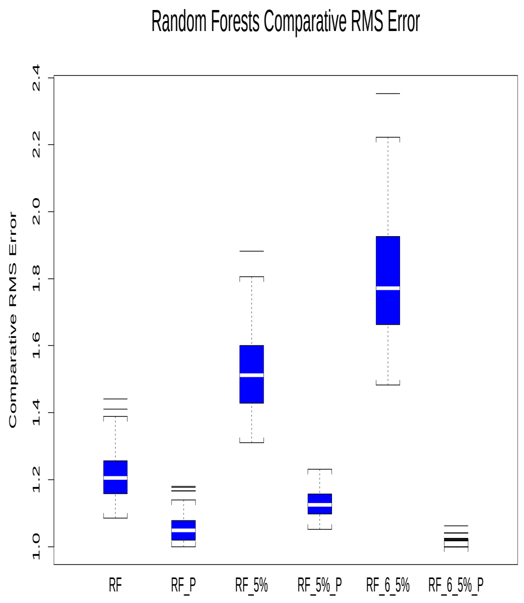
Another element that accounts for the larger dispersion is the *size of the trees*. *Bag_5%* uses the largest possible trees that can be built on a 5% sample. These are considerably smaller than the large trees that can be built on a bootstrap sample, and therefore less correlated with the response; this leads to a more diverse ensemble. *Bag_6_5%* uses six terminal nodes trees, and thus provides even greater dispersion through this mechanism.

Post-processing with the l_1 penalty provides the greatest improvement when the dispersion of the ensemble is the greatest. For the case for shallow trees each fitted on 5% samples post-processing improved the accuracy of the ensemble by about 50%, ultimately causing it to be the *most* accurate of all the competitors shown in Fig. 2. It is important to note that, besides being the most accurate, *Bag_6_5%_P* brings a huge improvement in speed as compared with the classic bagging of large trees. The generation of the ensemble is few hundreds times faster and subsequent prediction is roughly five times faster.

Post–processing bagged large trees (ensemble labelled *Bag_P*) does not produce an improvement in accuracy over straightforward bagging. The large trees are highly correlated and the dispersion of the ensemble is low. Fitting the coefficients in the post–processing regression is a bias reduction technique where the l_1 penalization reduces variance through shrinkage and feature selection. These ingredients do not seem to help much in settings where each individual base learner (tree) has low bias, in this case *comparable with the bias of the ensemble as a whole*. The averaging involved in the classic bagging is an aggressive variance reduction technique that is appropriate in the context of high variance and low bias. The very large trees being highly correlated on the training set are roughly equally relevant as predictors; the l_1 regularization is best suited when the predictors do not have equal importance. In fact the best settings for the l_1 penalty are *sparse* situations where relatively few of the predictors have non-zero coefficients (Donoho et al. (1995). While this regularization seems to be appropriate for the predictors from highly diverse ensembles like *Bag_6_5%*, it does not seem well suited for the large bagged trees.

9.4 Random Forests ISLEs

The left panel of Fig. 3 deals with random forests (*RF*) like ensembles. At each split, during tree construction $n_s = \lfloor log_2 n + 1 \rfloor$ input predictor variables are randomly selected (the default suggested by Breiman 2001) and the best split variable is chosen among them. This default is used in all the results of the simulation study and real data experiments. Post–processing (38) improves the classic random forests (*RF*) that build large (fully grown on the bootstrap samples) trees by around 10%, and the forests that build trees on 5% (without replacement) of the data (*RF_5%*) by 30%. The biggest improvement with post–processing (80%) is obtained, as with bagging, for the random forest ensemble using shallow (six terminal node) trees built on 5% of the data (*RF_5%_P*), again making it the most accurate among these competitors. The dispersion of the random forest ensembles is, overall, larger



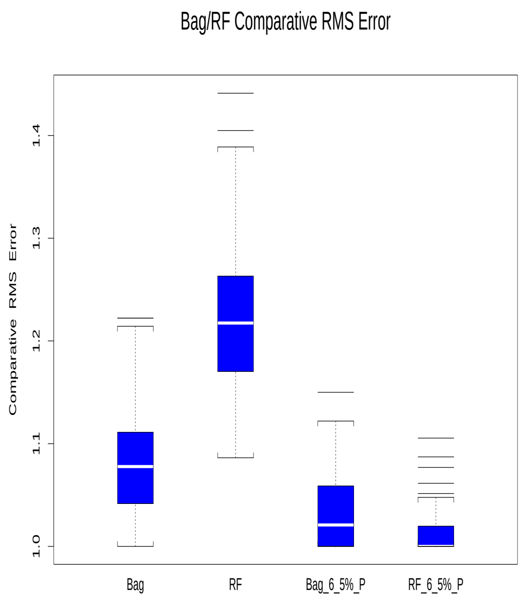


Figure 3: Distributions (left panel) of comparative root mean squared error over the 100 data sets in the regression Monte Carlo study for various random forest ensembles and corresponding post–processed ensembles (suffix "P"). Qualitative results are similar to that of bagged ensembles (Fig. 2). The right panel compares results from selected bagged and random forest ensembles.

than that with bagging (as noted in Section (6.2)) and post-processing here provides more improvement.

Bagging, random forests, and their two corresponding fast post-processed winning ensembles (*Bag_6_5%_P* and *RF_6_5%_P*) are compared on the right panel of Fig. 3. Here straight bagging is better than random forests: the presence of the spurious variables seems to limit the performance of the randomized trees that choose relatively small random subsets of features at each split. The degradation is even more pronounced when one deals with shallow trees built on small fractions of data. This is the case with *RF_6_5%* (left panel). However, post-processing (*RF_6_5%_P*) transforms this ensemble to be the overall winner. It is also the overall winner in terms of training speed: it is more than 100 times faster than the classic random forests and more than 1,000 times faster than bagging.

9.5 Sequential ISLEs

The relative performance of several sequential ISLEs (Section 7.1) is displayed in the left panel of Fig. 4. The base learners for all sequential ISLEs are taken to be the small six terminal nodes trees. The leftmost boxplot corresponds to the sequential ensemble *Seq_0.1* generated as in Section 7.2 with $\eta = 0.1$ in (18) (36). This is the default value giving rise

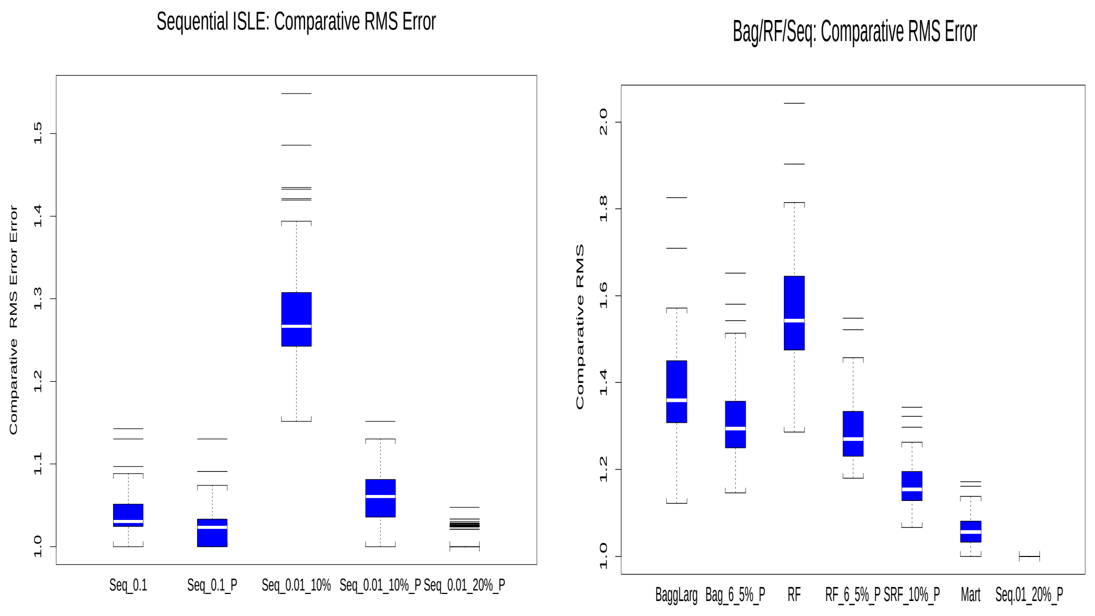


Figure 4: Distributions (left panel) of comparative root mean squared error for several sequentially induced ensembles and corresponding post–processed ensembles (suffix "P") over the 100 data sets in the regression Monte Carlo study. As with the bagged and random forest ensembles, post–processing most improves the most diverse weakest ensembles, but the improvements are less dramatic. The right panel compares several bagged, random forest, and sequential ensembles. Here the sequentially induced ensembles produce more accurate results.

to the L2–Boost algorithm (MART) described in Friedman 2001. Post–processing (38) this ensemble (*Seq_0.1_P*) produces an improvement that is relatively small compared with the improvement obtained by post–processing the sequential ensemble with a larger dispersion *Seq_0.01_10%*. This is a hybrid ensemble (Section 7.3) obtained by sequentially modifying the loss function (18) (36) and data distribution at each step as in (37). Here the parameter $\eta = 0.01$ is used in (18) (36) in order to compensate for the increased ensemble dispersion caused by 10% sampling. Post-processing this ensemble (*Seq_0.01_10%_P*) significantly improves its accuracy. The best among the displayed ISLEs is *Seq_0.01_20%_P*, an ensemble that is significantly faster to construct than *Seq_0.1*.

Here, as well as in the other simulation and real data results throughout the paper, it should be noted that the "optimal" fraction of observations sampled for the sequential ensembles is, in general, *larger* than for the parallel ones (bagging and random forests). The sequential strategy of building the ensembles (Section 7) intrinsically produces a *larger dispersion* (larger $\hat{\sigma}$ in (15)) as a consequence of modifying the loss criterion at each step.

One should not expect a large improvement by post–processing an ensemble such as *Seq_0.1*. The regularization given by the shrinkage involved in the gradient boosting (small η in (18)), approximately solves (38) in a setting where the predictors are all possible trees with fixed size (six terminal nodes in our case) that can be built on the training set (see

Hastie, Tibshirani and Friedman 2001, Chapter 10.12.2). The approximation involves a greedy search strategy in place of an "optimal" exhaustive search for finding the best tree to add to the ensemble at each step. Here post–processing can help a little by *re–adjusting* the coefficients for each tree. The hybrid strategies that sample data (and sometimes features as well) represent a further departure from the optimal exhaustive search. For these strategies post–processing can provide a dramatic improvement, often leading to higher accuracy than the pure sequential approach.

The right plot of Fig. 4 shows an overall comparison for the parallel and sequential ensembles under study. The ensemble labelled *Mart* is the default L2–Boost (Friedman 2001) MART algorithm that uses (39) as a loss function; it corresponds to the *Seq_0.1_50%* ensemble. The ensemble labelled *SRF_10%_P* (*sequential random features* ISLE) is obtained by post-processing a hybrid (Section 7.3) that perturbs all aspects of the problem: the data distribution, by sampling 10% of the data at each step, the loss function as in (18) (36), and also the algorithm used to construct each individual tree, by sampling six features at random at each split as described in Section 6.2. The overall picture shows that here the sequential ISLEs tend to perform better than the parallel ones. This is consistent with results observed in classical Monte Carlo integration where quasi-Monte carlo methods tend to outperform those based on purely random Monte Carlo sampling. In all cases post-processing the ensembles with large dispersion (large $\hat{\sigma}$ in (15)) tend to produce more accurate results.

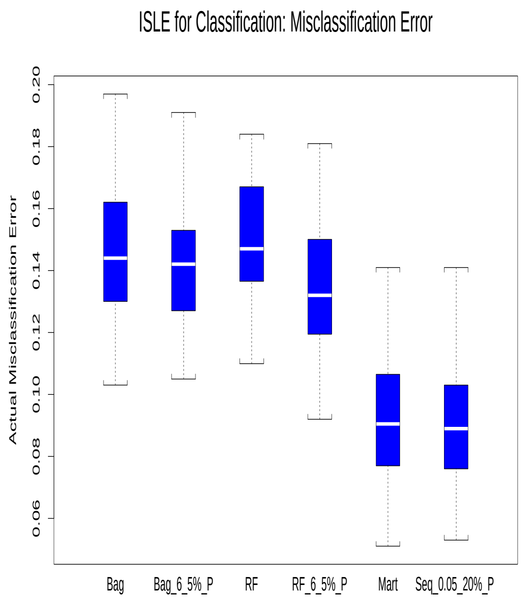
9.6 Classification

To simulate an environment for testing the various types of ISLEs in a classification framework, 100 randomly generated target functions were simulated as in Section 9.1 and the class labels ($y \in \{-1, 1\}$) were obtained by thresholding each target at its median. In this way, 100 different training data sets of size 10,000 were obtained and the misclassification error rate for each ensemble under study was estimated on separate corresponding test sets of size 10,000. Although the Bayes error rate is zero for all of these problems, the decision boundaries are fairly complex and the nature of the targets can make the classification problems difficult.

The sequential ISLEs are generated as in Section 7, using the loss criterion (41); this loss was also used for post–processing all the ensembles. This criterion, unlike others used for classification such as the logistic–likelihood loss (Friedman 2001, Friedman, Hastie and Tibshirani 2000) allows for updates that lead to a computationally fast and numerically stable post–processing algorithm (Friedman and Popescu 2003). When used in conjunction with $\eta << 1$ in (18) (36), the ramp loss (41) compares favorably to the logistic–likelihood loss in terms of accuracy when generating the sequential ensembles.

As in Section 9.2, all sequential ISLEs use 6 terminal nodes trees. As before, when features are sampled, a subset of 6 out of 40 attributes are randomly selected at each split.

In general, the relative performance of the parallel and sequential classification ISLEs is similar to that obtained in the regression setting. The results closely correspond to the ones presented in Sections 9.3–9.5 and summarized in Fig. 2–4. Figure 5 shows the overall picture of various ISLEs for classification. In the left plot of the actual misclassification



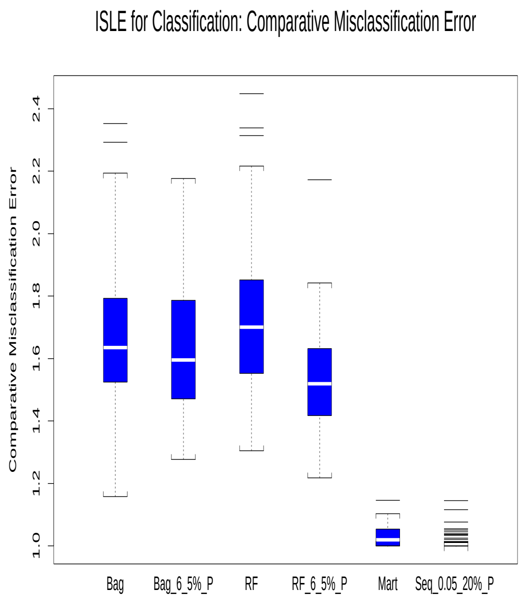


Figure 5: Distributions of misclassification (left) and comparative (right) error rates for several bagged, random forrest, and sequential ensembles over the 100 data sets in the classification Monte Carlo study. The results parallel those obtained in the regression Monte Carlo study.

error rate it is seen that the degree of difficulty for the 100 problems varies. The right plot using the analog of (47) facilitates a comparison among these methods. Random forests are not as accurate as bagging, an artifact of the spurious variables; however the fast post–processed forest–like ensemble *RF_6_5%_P* achieves the best accuracy among the parallel ISLEs. The sequential ISLEs (quasi-Monte Carlo) are here much better than the parallel ones. The ensemble labelled *Mart* corresponds to a sequential ensemble generated with the logistic–likelihood loss (the algorithm K2–Boost from Friedman 2001). Here the overall winner is *Seq_0.05_20%_P*, being slightly more accurate than MART.

10 Real Data Experiments

This section presents experiments on actual data. Five hundred trees were generated for each ensemble. Shallow tree base learners have ten terminal nodes, while the large trees used for classic bagging and random forests are fully grown. For random forests the number of random features are always taken to be $\lfloor log_2 n + 1 \rfloor$, where n is the total number of features, as suggested in Breiman 2001.

Algorithm/MAE (2· stdev)	data	target=RF	target=MART
Bagging	0.563(0.008)	0.328(0.013)	0.465(0.007)
Bagging P	*0.552(0.008)*	*0.375(0.005)*	*0.378(0.011)*
Bagging 10 nodes 5%	0.885 (0.016)	0.550(0.010)	0.756(0.013)
Bagging 10 nodes 5% P	*0.537 (0.010)*	*0.321(0.006)*	*0.274(0.007)*
Random Forests	0.560 (0.008)	0.272(0.013)	0.500(0.007)
Random Forests P	*0.502 (0.008)*	*0.312(0.013)*	*0.344(0.007)*
RF 10 nodes 5 %	0.863 (0.014)	0.592(0.010)	0.779(0.011)
RF 10 nodes 5 % P	*0.495 (0.014)*	*0.297(0.009)*	*0.222(0.006)*
MART	0.473(0.009)	0.280(0.005)	0.145(0.002)
Seq_0.01_5%	0.615(0.010)	0.429(0.009)	0.492(0.015)
Seq_0.01_5%_P	*0.487(0.020)*	*0.301(0.007)*	*0.173(0.003)*

Table 1: Scaled median absolute error of several bagged, random forest, and sequentially induced ensembles, along with their corresponding post–processed counterparts (suffix "P") on the census data (second column). Qualitatively, the results reflect those obtained in the regression Monte Carlo study. The third and fourth columns respectively represent corresponding results for the random forest and MART created target functions (see text). Most ensembles do comparatively well on the random forest generated target, whereas the sequentially induced ensembles seem to have a competitive edge on the MART target function.

10.1 Robust regression: census data

The data set used here was obtained from the IPUMS (*http://www.ipums.umn.edu/usa*) census database. There are 71 variables selected from the 2000 Supplementary Survey (CS2SS). The goal here is to predict the total personal income of a given individual from the other 70 variables. These consist of a mixture of categorical (e.g. "occupation", "industry") and of orderable (e.g. "grade level attending", "family size") features. There are many missing values. The dataset is an IPUMS sample of size 46,937 from the United States population. The observations come with individual weights that reflect the distribution of the United States population. These weights are used both in training and in reporting the results.

Unlike the datasets presented in the simulation study, severe response (income) outliers were present in the census data. Therefore, the Huber-M loss (40) was used in the post-processing as well as in generating the sequential ensembles (following the generic methodology described in Section 7). Also the test error reported in every case is the scaled median absolute error (MAE):

$$MAE = median(y - \hat{y})/median(y - median(y)). \tag{48}$$

The data set was randomly partitioned into a learning (training) set consisting of 36,000 and a test set of 10,937 observations. The quantities from Table 1 are test errors averaged over five such partitions. The numbers in parentheses are two times the standard deviations of the means that assess the significance of the differences. The error of a single ten node tree is 0.805 (0.041) and the error of the largest possible tree grown on the training data is 0.687 (0.006).

The second column in Table 1 present the results obtained by various ISLE methods (first column) on these data. The post–processed ensembles performances are shown in italics. The results are consistent with those from the simulation study. The largest improvement by post-processing (38) among the parallel (bagging and random forest) ISLEs appears for the ensembles with the highest dispersion that employ shallow (10 nodes) trees, each built on 5% of the data (improvement of 65% for bagging and 74% for random forests). The overall winner among these parallel ISLEs is a fast post-processed random forest ensemble that builds shallow trees on small fractions (5%) of the data. The sequential ISLE *Seq*_0.01_5%_*P* that generates 10 terminal node trees built each on 5% samples with the loss function (40) performs comparably in accuracy with MART (the algorithm Huber–M–Boost of Friedman 2001). As in the simulation study, the sequential (quasi Monte Carlo) ISLEs appear to have a performance edge, but the differences here are much less pronounced.

It is important to note that measure (48), unlike (46), includes the irreducible error ϵ (45), and, therefore, the fractional differences in estimating the actual underlying target function are much greater. In order to get an idea of such differences, two "target" functions were defined on these data. The first is taken to be the random forest approximation, $F_1^*(\mathbf{x}) = \hat{F}_{RF}(\mathbf{x})$, obtained on the census data as averaged over the five replications (learning–test partitions). The second target is the *Seq*_0.1_50% (MART) approximation $F_2^*(\mathbf{x}) = \hat{F}_{MART}(\mathbf{x})$, obtained in the same manner. For each of these targets five data sets $\{\tilde{y}_i, \mathbf{x}_i\}_1^N$ were constructed by forming the residuals from the corresponding target

$$\{r_{il} = y_i - F_l^*(\mathbf{x}_i)\}_{i=1}^N, \quad l = 1, 2, \tag{49}$$

randomly permuting them among the observations, and then adding them to each corresponding target value

$$\{\tilde{y}_{il} = F_l^*(\mathbf{x}_i) + r_{P(i)l}\}_{i=1}^N \quad , l = 1, 2. \tag{50}$$

Here $P(\cdot)$ is a random permutation of the integers $\{1, ..., N\}$. This process was repeated five times with five different random perturbations.

The last two columns of Table 1 show the scaled median error in approximating these two targets

$$median[F_l^*(\mathbf{x}) - \hat{F}_l(\mathbf{x})]/median[F_l^*(\mathbf{x}) - median(F_l^*(\mathbf{x}))], \quad l = 1, 2,$$

as averaged over the five replications. Here $\hat{F}_l(\mathbf{x})$ represents the approximation obtained by the respective methods as applied to the "data" (49) (50), constructed from $F_l^*(\mathbf{x}), l = 1, 2$.

As would be expected the sequential ISLEs (including MART) perform best when approximating the MART target function $F_2^*(\mathbf{x})$. The parallel ISLEs based on bagging and random forests do relatively poorly when compared to the sequential ones. Nevertheless, post–processing brings significant improvements. For example, post–processing the fast random forest ensemble that builds shallow trees on 5% samples leads to an ISLE that is more than tree times as accurate as the corresponding non post–processed ensemble, and more than twice as accurate as the classic random forest ensemble!

For the random forest target function $F_1^*(\mathbf{x})$, the parallel ISLEs perform relatively well (especially random forests itself). However the sequential ISLEs also tend to do comparatively well on the random forest target function.

MART produces approximations that emphasize low order interactions and, when appropriate, relatively few input predictors variables. By contrast random forests tend to produce higher order interaction models involving many predictors. The corresponding derived targets $F_2^*(\mathbf{x})$ and $F_1^*(\mathbf{x})$ will reflect these properties. The results shown in the last two columns of Table 1 suggest that sequential ISLEs achieve their greatest advantage for low interaction targets involving relatively few relevant predictor variables. This was the case for the simulation study presented in Section 9. In situations where the true underlying target function $F^*(\mathbf{x})$ (2) tends to involve many relevant variables and/or high order interactions, the sequential ISLE strategy does not seem to have an advantage (nor disadvantage) with respect to the parallel ISLEs based on bagging and random forests.

10.2 Classification: spam 2003 data

The spam prediction problem considered here has its goal to discriminate the spam emails from the non-spam ones. The database consists in 19,177 emails. Each email is characterized by 833 binary features among which are the presence in the email text of various strings like *"cash bonus"*, *"guarantee"*, characteristics of the email header such as *Invalid Date* or *"From" yahoo.com does not match "Received" headers*, URL features such as: *Uses a numeric IP address in URL* or *Uses non-standard port number for HTTP* etc. Details and the actual dataset are available at *http://www.data-mining-cup.com/*.

A test set of 4,000 emails was randomly chosen; the remaining emails were used for training each ensemble. This experiment was repeated five times with different random partitions. The sequential ensembles for the classification experiments in this section are generated as in Section 7 using the least-squares ramp loss function (41). The same loss function is also used in post–processing for all the ensembles.

Table 2 presents the test misclassification errors (average and two standard deviations). Although the base error (predicting majority class) is 39%, many of the ensembles considered here achieve very low misclassification errors (less than 0.01=1%). The misclassification error of a single 10 terminal nodes tree is 0.0303 ($2 \cdot stdev = 0.0008$) and the error for a 15,177 terminal node tree (the largest possible trees that can be built on the training data) is 0.0120 ($2 \cdot stdev = 0.0012$).

One can see from Table 2 that the ensemble *Bag_10_1%_P* obtained from postprocessing the widely dispersed ensemble using 10 terminal node trees built on 1% subsamples (*Bag_10_1%*) produces an improvement by a factor of three and comes close to the error rate achieved by bagging large trees while being 700 times faster to construct.

The random forests use 10 out of the total of 833 features randomly sampled at each split. Post–processing random forests helps; the improvement in accuracy is dramatic when

Algorithm	Error(2·stdev)
Bagging	0.0085(0.0011)
Bagging P	*0.0089(0.0012)*
Bagging 10 nodes 1%	0.0327(0.0013)
Bagging 10 nodes 1% P	*0.0103(0.0020)*
Random Forests	0.0147(0.0012)
Random Forests P	*0.0086(0.0011)*
RF 10 nodes 1%	0.0837(0.0048)
RF 10 nodes 1 % P	*0.0091(0.0010)*
Seq_0.1_50%	0.0083(0.0005)
Seq_0.01_5%	0.0120(0.0013)
Seq_0.01_5%_P	*0.0080(0.0010)*
SRF_0.01_5%	0.0256(0.0016)
SRF_0.01_5%_P	*0.0084(0.0006)*

Table 2: Misclassification error rates of various ensembles and their post–processed counterparts (suffix "P") on the spam 2003 data. Error rates for all methods are quite small. Bagging and all fast post–processed ensembles have comparable performance.

post–processing the ensemble *RF_10_1%*: here a factor of nine! The ensemble *RF_10_1%_P* that is competitive with the best ensembles in Table 2 builds 10 terminal nodes trees on only 1% of the data while selecting at each split a very small fraction of features. Furthermore, ensemble construction based on bagging and random forests can be parallelized, thereby allowing training to be done on databases involving a huge number of emails. Post-processing helps the sequential ensembles built on small fraction of the data (5%) (as it is the case for *Seq_0.01_5%*) and leads to a big improvement (a factor of three) for the hybrid *SRF_0.01_5%* that sequentially builds randomized trees (10 random features at each split) on 5% of the data.

Unlike the simulation examples (Section 9.6), and the census regression problem (Section 10.1) in the real data classification example presented here there is no signifficant difference between the best parallel and sequential ensembles. Here the ISLE methodology produces overall competitive ensembles in terms of accuracy while giving rise to huge computational reductions in training time. The post-processed ensembles also have a substantial advantage in time required for subsequent prediction (see below). This can be very important for servers that receive email messages at high rates.

The fractions used here for sampling the training data (1% for the parallel ISLEs and 5% for the sequential ones) are smaller than those used in Section 9.3. This is due to the larger sized training data sets involved (see Section 5.1).

11 Discussion

A two–stage strategy for creating a final model has been previously considered in the context of prediction ensembles. Breiman 1996b suggests forming linear combinations of various learners to improve accuracy. This methodology has its roots in the idea of *stacking*, introduced in Wolpert 1992. The stacking methodology is used also in Wolpert and Macready 1996. They employ various post–processing strategies applied to classic regression bagged ensembles (built on bootstrap samples) and report improvements in accuracy when compared with classic bagging.

The present paper presents classic ensemble prediction methodologies such as bagging, random forests, boosting (and bayesian model averaging) in a unifying framework based on numerical integration with importance sampling. This helps explain some of their properties. In particular, the empirically observed trade-off between the strength of individual base learners and the correlations among them (Breiman 2001) is related to the width of the corresponding parameter sampling distribution (15) used by the integration rule (9). It is seen that estimating the optimal coefficients (10) for the integration rule through (38), rather than simply averaging the integrand values, allows for the use of much weaker less correlated base learners (more diverse ensembles), often leading to substantial gains in prediction accuracy and dramatic reductions in training time. As in classical Monte Carlo integration, the sequential strategy (Section 7) corresponding to quasi Monte Carlo tends to outperform the parallel strategy (Section 6) that corresponds to purely random sampling Monte Carlo. However, the parallel implementation can result in a strong computational advantage for very large data sets.

Estimating the optimal coefficients through (38) also results in reduced computation for subsequent predictions; the regularized l_1 regression-like algorithms used for post–processing produce *sparse* models, with many zero coefficients. Therefore, the corresponding (zero coefficient) base learners need not be evaluated for prediction. Another ingredient that helps here is the use of the shallow trees with a considerably faster look-up speed than the large trees used in ensembles like bagging or random forests. Prediction speed can be very important for some real time applications (e.g. spam recognition, fraud detection, automatic target recognition etc).

The methodology presented in this paper is not restricted to decision tree ensembles. One could use the ISLE methodology to create ensembles formed by sigmoids of linear combinations of the predictor variables (6), or radial basis functions, thereby creating single layer neural or radial basis function networks. One could combine base learners from different classes (trees, linear functions of the original features, sigmoids, etc) to obtain MISLEs (Multiple Importance Sampled Learning Ensembles). This can be especially useful when the base learners from a given class have difficulty in estimating particular functional (input-output) dependencies.

Finally, it should be noted that all comparisons presented in this paper (Sections 9 and 10) are made on moderate to large data sets (number of observations) commonly encountered in data mining applications. It is in these settings that the computational advantages associated with the post–processing strategy achieve their greatest significance. The particular trade–offs suggested by these examples may or may not extend to applications involving relatively small data sets.

12 Acknowledgements

The work of Jerome Friedman was partially supported by the Department of Energy under contract DE-AC03-76SF00515 and the National Science Foundation under grant DMS-97-64431. The work of Bogdan Popescu was partially supported by the National Science Foundation under grant DMS-97-64431.

References

[1] Breiman,L. (1996a). Bagging Predictors. *Machine Learning*, **26** 123-140.

[2] Breiman,L. (1996b). Stacked Regressions. *Machine Learning*, **24** 51-64.

[3] Breiman,L. (2001). Random Forests, random features. Technical Report, University of California, Berkeley.

[4] Breiman,L., Friedman, J. H., Olshen, R., and Stone, C. (1983). *Classification and Regression Trees.* Wadsworth.

[5] Freund, Y. and Scahpire, R.E. (1996). Experiments with a new boosting algorithm. *Machine Learning: Proceedings of the Thirteenth International Conference*, Morgan Kauffman, San Francisco, 148-156.

[6] Copas, J. B. (1983). Regression, prediction and shrinkage (with discussion). *J.R. Statist. Soc. B* **45**, 311-354.

[7] Chipman H., George E., McCullagh R. (1998). Bayesian CART model search (with discussion). *Journal of Am. Stat. Assoc. 93 (6)*, 937-960.

[8] Denison, D. G. T., Holmes, C. C. , Mallik, B. K. and Smith, A. F. M. (2002). *Bayesian nonlinear methods for classification and regression.* Chichester: John Wiley.

[9] Dietterich, T. G. (2000). An experimental comparison of three methods for constructing ensembles of decision trees: bagging, boosting and randomization. *Machine Learning*, **40 (2)** 139-158.

[10] Donoho, D., Johnstone, I. (1993). Ideal spatial adapdation by wavelet shrinkage. *Biometrika* **81**, 425-455.

[11] Donoho, D., Johnstone, I., Kerkuacharian, G. and Picard, D. (1995) Wavelet shrinkage; asymptotya? (with discussion). *J. Royal. Statist. Soc* **57**: 201-337.

[12] Efron, B. and Tibshirani, R. (1993). *An introduction to the bootstrap*, Chapman and Hall, London.

[13] Efron, B., Hastie, T., Johnstone, I. and Tibshirani, R. (2002). Least Angle Regression. *Annals of Statistics.* To appear.

[14] Friedman, J. H. (2001). Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, Vol. 29, No. 5.

[15] Friedman, J. H., Hastie, T. and Tibshirani, R. (2000). Additive logistic regression: a statistical view of boosting (with discussion). *Annals of Statistics*, **28**, 337-407.

[16] Friedman, J. H., and Popescu, B. E. (2003). Gradient directed regularization for regression and classification. *Stanford University, Department of Statistics.* Preprint.

[17] Hastie, T., Tibshirani, R. and Friedman, J.H. (2001). *Elements of Statistical Learning.* Springer-Verlag.

[18] Ho, T. K. (1995). Random Decision Forests. *Proceedings of the 3rd International Conference on Document Analysis and Recognition* 278-282.

[19] Hoerl, A. E. and Kennard, R. (1970). Ridge Regression: Biased estimation for nonorthogonal problems. *Technometrics* **12**, 55-67.

[20] Huber, P. (1964). Robust estimation of a location parameter *Annals of Math. Stat*, **53**, 73-101.

[21] Popescu,B.E. (2003). Ensemble Learning. *Ph.D Thesis*. Stanford University.

[22] Quinlan, R. (1991). *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Mateo.

[23] Rumelhart, D.E.,Hinton, G. E., Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature* **323**, 533-536.

[24] Schapire, R. and Singer, Y. (1998). Improved boosting algorithms using confidence-rated predictions. *Proceedings of the Eleventh Annual Conference on Computational Learning Theory.*

[25] Stroud (1971). *Approximate calculation of multiple integrals*. Prentice Hall, Englewood Cliffs, New Jersey.

[26] Tibshirani, R. (1996). Regression shrinkage and selection via the lasso *J. Royal. Statist. Soc. B.* **58**: 267-288.

[27] Wolpert, D. H. (1992). Stacked Generalization *Neural Networks* **5**: 241-259.

[28] Wolpert, D. H. and Macready W. G. (1996). Combining stacking with bagging to improve a learning algorithm. *Technical Report SFI-TR-96-03-123, Santa Fe Institute, Santa Fe, New Mexico.*