Knowl Inf Syst
DOI 10.1007/s10115-010-0358-0

REGULAR PAPER

Multi-resolution boosting for classification and regression problems

Chandan K. Reddy · Jin-Hyeong Park

Received: 15 January 2009 / Revised: 29 July 2010 / Accepted: 23 October 2010

Abstract Various forms of additive modeling techniques have been successfully used in many data mining and machine learning–related applications. In spite of their great success, boosting algorithms still suffer from a few open-ended problems that require closer investigation. The efficiency of any additive modeling technique relies significantly on the choice of the weak learners and the form of the loss function. In this paper, we propose a novel multi-resolution approach for choosing the weak learners during additive modeling. Our method applies insights from multi-resolution analysis and chooses the optimal learners at multiple resolutions during different iterations of the boosting algorithms, which are simple yet powerful additive modeling methods. We demonstrate the advantages of this novel framework in both classification and regression problems and show results on both synthetic and real-world datasets taken from the UCI machine learning repository. Though demonstrated specifically in the context of boosting algorithms, our framework can be easily accommodated in general additive modeling techniques. Similarities and distinctions of the proposed algorithm with the popularly used methods like radial basis function networks are also discussed.

Keywords Boosting · Regression · Classification · Weak learner · Additive models · Gaussian kernel · Multi-resolution

1 Introduction

In statistical data mining, boosting methods have been proven to be very effective for not only improving the classification accuracies but also reducing the bias and variance of the estimated classifier. The boosting meta-algorithm is an efficient, simple and easy to

C. K. Reddy (✉)
Department of Computer Science, Wayne State University,
Detroit, MI 48202, USA
e-mail: reddy@cs.wayne.edu

J.-H. Park
Integrated Data Systems Department, Siemens Corporate Research,
Princeton, NJ 08540, USA

Published online: 16 November 2010

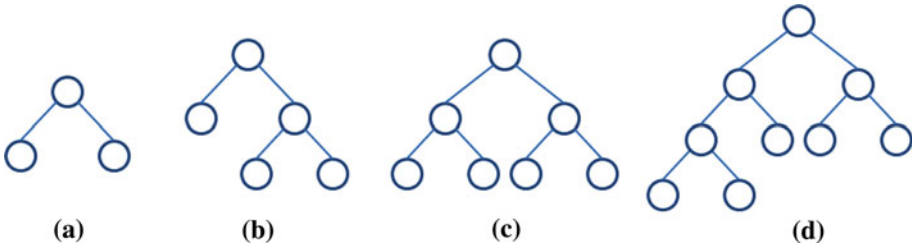

Fig. 1 Model-driven multi-resolution boosting: Every iteration of boosting considers all the datapoints. The model complexity of the model (a decision tree in this case) is increased as the algorithm progresses. **a**–**d** Tree models with different model complexity starting with a decision stump (**a**). The term 'complexity' used here is reflected by the number of splits in the tree

manipulate additive modeling technique that can use potentially any weak learner available [15]. The most popular variant of boosting, namely the AdaBoost (Adaptive Boosting) in combination with the trees, has been described as the "best off-the-shelf classifier in the world" [7]. In simple terms, boosting algorithms combine weak learning models that are slightly better than random models. Boosting algorithms are generally viewed as functional gradient descent schemes and obtain the optimal updates based on the global minimum of the error function [11]. Both classification and regression-based boosting algorithms have been successfully used in a wide variety of applications in the fields of computer vision [33,34], data retrieval [2,29], bioinformatics [16,18] etc.

In spite of their great success, boosting algorithms still suffer from a few open-ended issues such as the choice of the optimal set of parameters for the weak learner. The framework proposed in this paper, termed as "*Multi-resolution Boosting*", can model any arbitrary function using the boosting methodology at different resolutions of either the model or the data. Here, we propose a novel boosting model that can take advantage of using the weak learners at multiple resolutions. In this work, we achieve the *multi-resolution* concept in the context of boosting algorithms by one of the following two ways:

- *Model-driven multi-resolution:* This is achieved by varying the complexity of the classification boundary. This approach will provide a systematic procedure that increases the complexity of the weak learner as the boosting iterations progress. The first few iterations will use weak learners that obtain a simple classification boundary and the boundary becomes more and more complex as the iterations proceed. Figure 1 demonstrates an example of model-driven multi-resolution boosting. This framework can obtain weak learners in a systematic manner. The detailed algorithm will be discussed in Sect. 4.1 of this paper.
- *Data-driven multi-resolution:* This can be achieved by considering the data (not the model) at multiple resolutions during each iteration in the boosting algorithm. Our framework chooses the weak learners for the boosting algorithm that can best fit the current resolution; as the boosting iterations progress, the modeling resolution is increased. Our algorithm provides the flexibility of choosing the weak learner dynamically compared to static learners with certain prespecified parameters. For every learner in the boosted model, the resolution is either maintained or doubled and a weak learner is modeled to accurately fit the data during that particular iteration. Figure 2 demonstrates an example of data-driven multi-resolution boosting. This framework is discussed in Sect. 4.2 of this paper.

The main idea of the proposed framework is to *use Multi-resolution data (or model)-driven fitting in the context of additive modeling*. The rest of the paper is organized

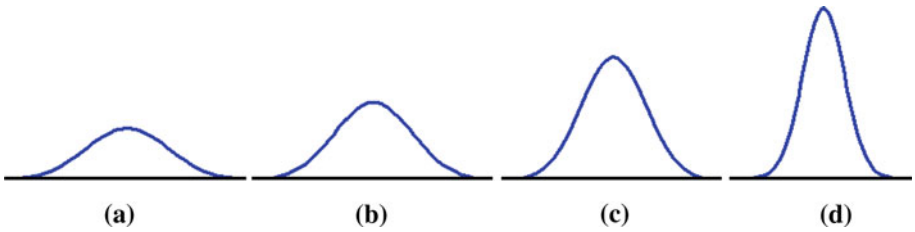

Fig. 2 Data-driven multi-resolution boosting considers the same model (e.g. Gaussian) at every iteration. However, the number of datapoints that will be modeled during each iteration will be reduced as the algorithm progresses and hence the kernel width of the model is reduced. **a–d** The width of the Gaussian weak regressor is being reduced since the number of datapoints that are modeled is reduced

as follows: Sect. 2 gives some relevant background on various boosting techniques and scale-space kernels. Section 3 shows the problem formulation in detail and discusses the concepts necessary to comprehend our algorithm. Section 4 describes both the model-driven and the data-driven multi-resolution boosting frameworks. Section 5 gives the experimental results of our algorithms on both synthetic and real-world datasets. Finally, Sect. 6 concludes our discussion with future research directions.

2 Relevant background

Ensemble learning is one of the most powerful modeling methods that is effective and successful in a wide variety of applications in recent years [25]. Several ensembling techniques have been proposed in the literature and is still a very active area of research. Boosting [30] is one of the most widely used algorithm that has caught the attention of several researchers working in the areas of data mining and machine learning. As opposed to other popular ensemble learning techniques like bagging [6], boosting methods reduce the bias and the variance simultaneously [7]. A detailed comparison of other combining schemes is thoroughly investigated in [3]. A comprehensive study on boosting algorithms and their theoretical properties are given in [12]. One main advantage of boosting algorithms is that the weak learner can be a black-box which delivers the result in terms of accuracy. This is a very desirable property of the boosting algorithms that can be applied in several applications of predictive modeling [15]. The additive model provides a reasonable flexibility in choosing the optimal weak learners for a desired task. Various extensions for the original boosting algorithms had been proposed recently [31,38]. There had also been some work on boosting for regression problems [10,37]. A detailed study on L_2 norm-based classification and regression is given in [8]. The use of various loss functions for the boosting algorithm is discussed in [15]. Several extensions and theoretical generalizations have also been studied in the literature [9,28,35]. Despite the vast literature on boosting methods, researchers have not investigated the use of multi-resolution analysis in the context of boosting.

The data-driven multi-resolution framework proposed in this paper has some similarity with other popularly used models such as Radial Basis Function (RBF) Networks [4]. The RBF networks rely on the fact that any continuous function can be approximated by a sum of appropriately chosen Gaussian functions [23]. These networks are one form of neural networks that have a static Gaussian function for the hidden layer processing elements to attain non-linearity. The RBF networks compute the input to output map using local Gaussian approximators. During the training phase, the centers and width of the Gaussians in the hidden units along with the weights of the connections are determined. In spite of their

success, the RBF networks suffer from some of the drawbacks of neural networks such as fixing the topology, including the number of hidden nodes and the active connections between the layers [13]. Our algorithm contains very minimal number of user-defined parameters and does not suffer from these problems. It provides a systematic approach to automatically model the target function as a linear combination of Gaussian kernels. Being complex models, the training of RBF networks is difficult and they are more prone to the overfitting problem compared to the multi-resolution boosting algorithm. Being powerful local approximators, RBF networks perform poorly on noisy data since the coefficients are adjusted in order to accommodate local data. These networks do not average out noise over the entire data space.

In this paper, we propose a novel multi-resolution framework for choosing optimal weak learners during the iterations in boosting. This approach allows for effective modeling of the dataset at any given resolution [24]. In terms of analyzing (or modeling) a given dataset at different resolutions, our approach resembles wavelet decomposition techniques that are effective tools in the field of multi-resolution signal analysis [14,21]. Recently, there had been extensive use of this multi-resolution ideas in the data mining community [17]. In the model-driven multi-resolution boosting framework, the models are built by increasing the complexity during the boosting process. The main advantages of using this multi-resolution framework in the context of boosting are that they:

- allow systematic hierarchical modeling of the final target model.
- provide more flexibility by allowing the user to stop at a reasonable resolution.
- require fewer predefined user parameters.
- avoid the use of good learners in the beginning stages of modeling and progressively use them toward the end.

3 Problem formulation

We will now provide some notations that are used in the rest of this paper. Table 1 gives the notations and symbols used in our paper.

Let us consider N i.i.d. training samples $\mathcal{D} = (\mathcal{X}, \mathcal{Y})$ consisting of samples $(\mathcal{X}, \mathcal{Y}) = (x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)$ where $\mathcal{X} \in \mathbb{R}^{N \times d}$ and $\mathcal{Y} \in \mathbb{R}^{N \times 1}$. y_i indicate the final

Table 1 Description of the notations used

Notation	Description
N	Number of training samples
d	Dimensionality of the data
$\mathcal{X}$	Input features
$\mathcal{Y}$	Output values
$\mathcal{D}$	Data comprising of $\{\mathcal{X}, \mathcal{Y}\}$
x_i	ith datapoint
y_i	Target output of the ith datapoint
$f(\mathcal{X})$	Weak model
$F(\mathcal{X})$	Final model
L	Loss function
t	Iteration index for additive modeling
T	Total number of iterations
r	Residual $\lvert\mathcal{Y} - F\rvert$

target value to be modeled. In the case of binary classification problems, we have $y_i \in \{-1, +1\}$, and for regression problems, y_i takes any arbitrary real value. In other words, the univariate response $\mathcal{Y}$ is continuous for regression problems and discrete for classification problems. Now, we will discuss boosting algorithms applied to general classification problems. We choose to demonstrate the power of scale-space kernels in the context of logitboost algorithm because of its popularity and the power of additive modeling. The logitBoost algorithm uses adaptive Newton steps for fitting an additive symmetric logistic model by maximum likelihood [12].

3.1 Boosting for classification

Algorithm 1 Logitboost algorithm (2 classes)

Start with weights $w_i^{(0)} = \frac{1}{N}$, Initialize $F^{(0)}(x_i) = 0$ and probability estimates $p^{(0)}(x_i) = \frac{1}{2}$, for $i = 1, 2, \ldots, N$

for $t = 1$ to T **do**

1. Compute working response and weights for all datapoints

$$z_i^{(t)} = \frac{y_i - p^{(t-1)}(x_i)}{p^{(t-1)}(x_i) \cdot \left(1 - p^{(t-1)}(x_i)\right)}$$
$$w_i^{(t)} = p^{(t-1)}(x_i) \cdot \left(1 - p^{(t-1)}(x_i)\right)$$

2. Fit a weak learner $f^{(t)}$ by a weighted least squares regression of $z_i^{(t)}$ to the training data (x_i) using weights $w_i^{(t)}$.

$$f^{(t)} = arg\min_f \sum_{i=1}^{n} w_i^{(t)} \left(z_i^{(t)} - f(x_i)\right)^2$$

3. Update the Classifier and the probabilities

$$F^{(t)}(x_i) = F^{(t)}(x_i) + \frac{1}{2} f^{(t)}(x_i)$$
$$p^{(t)}(x_i) = \left(1 + \exp(-2F^{(t)}(x_i))\right)^{-1}$$

end for

Output the classifier $C(X) = sign(F^{(t)}(x_i))$.

Algorithm 1 gives generic Logitboost algorithm to solve two-class classification problem. The basic idea of boosting is to repeatedly apply the weak learner to modified versions of the data, thereby producing a sequence of weak learners $f^{(t)}(x)$ for $t = 1, 2, \ldots, T$ where T denotes predefined number of iterations. Each boosting iteration performs the following three steps: (1) Computes the response $(z_i^{(t)})$ and weights $(w_i^{(t)})$ for every datapoint. (2) Fits a weak learner $(f(\mathcal{X}))$ to the weighted training samples and (3) Computes the error and updates the final model $(F(\mathcal{X}))$. In this way, the final model obtained by boosting algorithm is a linear combination of several weak learning models.

In the case of classification problems, the penalty function induced by the error estimation is given by:

$$L(y_i, F^{(t)}(x_i)) = I(y_i \neq F^{(t)}(x_i)) \tag{1}$$

where I(.) denotes an indicator function that returns value 0, when $y_i \neq F^{(t)}(x_i)$ and 1 otherwise. In other words, the penalty term is 1 if the ith sample is misclassified and 0 if it is

correctly classified. Whether it is a classification or a regression problem, the main challenges in the boosting framework are the following: (i) the choice of the weak learner and (ii) the complexity of the weak learner. While choosing a weak learner model can be a complicated issue, tuning the right complexity for such a weak learner might be even more challenging. The multi-resolution framework proposed in this paper addresses the second issue.

We represent the probability of $y_i = 1$ by $p(x_i)$, where

$$p(x_i) = (1 + \exp(-2F(x_i)))^{-1} \tag{2}$$

3.2 Boosting for regression

The boosting framework discussed above works for classification problems and can be easily adapted to solve regression problems. In the case of regression problems, the penalty function is given by:

$$L(y_i, F^{(t)}(x_i)) = \|y_i - F^{(t)}(x_i)\|_p \tag{3}$$

where $\|\cdot\|_p$ indicates the L_p norm. We will consider $p = 2$ in this paper.

We formulate this multi-resolution boosting using the standard logitBoost algorithm with exponential L_2 norm loss function and demonstrate empirical results on both classification and regression problems. We also derive the update equations for the regression algorithms using logitboost (Newton's iteration) and first derivatives. Let us consider the following exponential loss function

$$L(y, F) = \exp(\|y - (F + f)\|^2) \tag{4}$$

Depending on the choice of weak regressor and the rate of convergence, one can use one of the following update mechanisms :

- Equate the first derivative to zero
- obtain the Hessian and use Quasi-Newton method

Let us consider the residual $r = |\mathcal{Y} - F|$.

Lemma 1 *During each boosting iteration, the minimum of the loss function is achieved by setting* $f = r$ *and the Newton's update is chosen by setting* $f = -(I + 2rr^T)^{-1} \cdot r$.

The proof is given in appendix-A. This is the residual fitting used in our data-driven multi-resolution boosting algorithm.

3.3 Scale-space kernels

Figure 3 demonstrates the *data-driven multi-resolution boosting* framework. In the beginning of the iteration, complete input data are considered for fitting a weak model. As the iterations progress, the number of datapoints considered for fitting is reduced and a Gaussian model is fit to the data [26]. At every boosting iteration, we obtain an optimal weak model that can fit the given number of datapoints at that particular resolution of interest. In this manner, we propose a hierarchical approach for modeling the data using weak models.

The data-driven multi-resolution framework proposed in this paper also uses the concepts of the scale-space-based methods that were popularly used in the computer vision literature [19,20]. Let us consider the general regression problem which is a continuous mapping $p(x) : \mathbb{R}^d \rightarrow \mathbb{R}$. More specifically, Gaussian kernels have been extensively studied in this

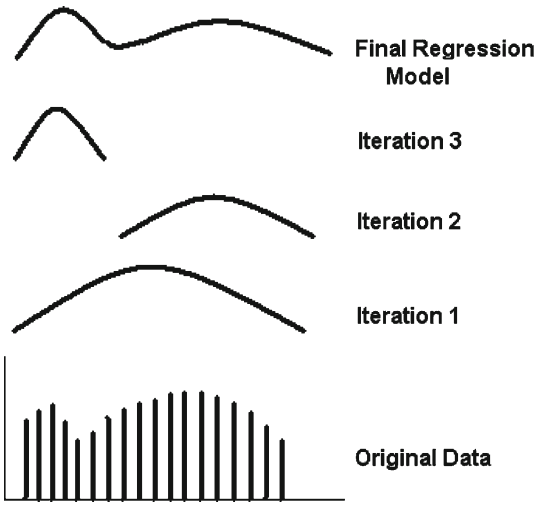

Fig. 3 Illustration of the original data and the weak models at multiple resolutions for boosting iterations. The additive model linearly combines these weak models to obtain the final 'strong' model

scale-space framework [32]. In scale-space theory, $p(x)$ is embedded into a continuous family $P(x, \sigma)$. Our method starts with an approximation of the entire dataset with Gaussian kernel of $\sigma = 0$. As the resolution (or scale) increases, the sigma value is reduced and eventually converges to zero. In our case, the highest frequency (or resolution) corresponds to fitting every datapoint with a Gaussian kernel. In simple terms, one can write the new kernel $p(x, \sigma)$ as a convolution of $p(x)$ with a Gaussian kernel $g(x, \sigma)$ given as follows:

$$P(x, \sigma) = p(x) \otimes g(x, \sigma) = \int p(x - y) \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{\|y\|^2}{2\sigma^2}} dy \tag{5}$$

The main reasons for choosing Gaussian models as weak regressors because:

- They are powerful universal approximators.
- Other popular regression models like Radial Basis Functions use Gaussian kernels.
- The scale-space theory is well formulated for Gaussian models. Hence, it allows systematic hierarchical modeling for regression problems.

As described earlier, choosing optimal σ value during every iteration of boosting becomes a challenging task. In other words, one cannot predetermine the reduction in the σ value.

4 Multi-resolution boosting framework

We will now describe both the model-driven and data-driven multi-resolution boosting algorithms. To demonstrate a reasonably wide-range applicability of the multi-resolution framework, we implement our framework using both the adaboost and logitboost algorithms [26, 27]. We show the model-driven multi-resolution algorithm using the adaboost framework for classification problems and the data-driven multi-resolution algorithm using the logitboost framework for regression problems. Though, we chose to demonstrate in this setting, the proposed framework can be generically applied to both classification and regression in the context of generic additive modeling approach. It should be noted that the data-driven multi-resolution is more general and can be applied for both classification and regression problems. However, the data-driven approach is more applicable for regression problems

because of the use of scale-space kernels (discussed in the previous section) that are more appropriate for regression tasks. The model-driven multi-resolution boosting is more applicable for classification tasks because changing the model complexity is more suitable for weak classifiers better than weak regressors.

4.1 Model-driven multi-resolution boosting

Algorithm 2 Model-driven multi-resolution boosting

Input: Data ($\mathcal{X}$), No. of samples (N), No. of iterations (T).
Output: Final model (F)
Algorithm:
Initialize the weight vector $W^{(1)}$ such that $w_i^1 = 1/N$ for $i = 1, 2, \ldots, N$
$nplits = 1$
for $t = 1$ to T **do**
  $[\hat{f}_0, err_0] = Train\_Val(\mathcal{X}, W^{(t)}, nsplits)$
  $[\hat{f}_1, err_1] = Train\_Val(\mathcal{X}, W^{(t)}, nsplits + 2)$
  if $err_0 < err_1$ **then**
    $f_t = \hat{f}_0 \quad \epsilon_t = err_0$
  else
    $f_t = \hat{f}_1 \quad \epsilon_t = err_1$
    $nsplits = nsplits + 1$
  end if
  Compute $\alpha_t = \frac{1}{2} ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$
  Modify the training weight $w_i^{(t+1)}$ as follows:

$$w_i^{(t+1)} = \frac{w_i^{(t)} \cdot \exp(-\alpha_t y_i f_t(x_i))}{z_t}$$

  where z_t is the normalization factor (chosen so that $\sum_{i=1}^{N} w_i^{(t+1)} = 1$)
end for
Output the final model $F(\mathcal{X}) = \sum_{t=1}^{T} \alpha_t f_t(\mathcal{X})$

Algorithm 2 describes our model-driven multi-resolution boosting framework using the adaboost algorithm for a *binary classification problem*. The weight vector W is initialized to $1/N$ uniformly. The main algorithm runs for a predefined number (T) of iterations. The procedure $Train\_Val$ will obtain weak learner (and the corresponding training error) using the weights $W^{(t)}$. It should be noted that this function returns the error on the validation data which is then used to decide the complexity of the tree used in that iteration. The number of splits ($nsplits$) is a parameter that determines the complexity of the model i.e. the more the number of splits in the weak learner, the higher the complexity of the model will be. It is initialized to one at the beginning. As the iterations progress, the complexity of the weak learner is either retrained or incremented depending upon the training error. The training error at the tth iteration is given by the following:

$$err(t) = \sum_{i=1}^{N} w_i^{(t)} \cdot I\{y_i \neq f_t(x_i)\}$$

For every iteration, the training error of the current model is compared with the error of a slightly complex model (with $nsplits + 2$ nodes in the tree). If this new model performs well, then the complexity of the current model is increased ($nsplits = nsplits + 1$) and the

re-weighting of the datapoints is computed using this new model. The weights are normalized (so that they sum to one) in every iteration. α_t measures the importance to be assigned to the weak model in the tth iteration (f_t). One can see that the algorithm appears to be working in a similar manner to the traditional adaboost, except for the fact that the choice of the weak learner is made more systematically from simple to complex and is not chosen arbitrarily as done in the standard boosting procedure. In this way, the algorithm increases the complexity of the weak learners chosen, and eventually, the weighted combinations of the selected weak learners are used as the final trained model. Hence, the model will have a very simple classification boundary in the initial stages, and the boundary becomes more and more complex as the iterations proceed.

4.2 Data-driven multi-resolution boosting

In this section, we will describe the data-driven approach where we maintain the same complexity of the weak learner, but change the number of datapoints to be modeled during each boosting iteration. Algorithm 3 describes our data-driven multi-resolution boosting framework for a *regression problem*. As mentioned earlier, this approach is demonstrated using the logitboost algorithm. The initial model is set to null or the mean value of the target values. The main program runs for a predefined number (T) of iterations. Initially, res (which indicates the resolution) is set to 1, which implies the simplest model by considering all the datapoints. The feature values are sorted independently by column-wise and the indices corresponding to each column are stored. As the iterations progress, the resolution considered for fitting the weak learner is retained or doubled depending on the error. In other words, depending on the error obtained at a given iteration, the resolution of the data is maintained or increased for the next iteration. For every iteration, the residual r is computed depending on the difference between the target value ($\mathcal{Y}$) and the final model (F). By equating the first derivative of the loss function ($r = L(\mathcal{Y}, F)$) to zero, we will set the residual as the data to be modeled during the next iteration using another weak regressor.[1] The best multi-variate weak model will be fitted to this data at a given resolution. The details of the procedure $bestfit$ that obtains the best weak model at a given resolution of the data are described in the next section. This procedure returns the best weak model ($\hat{f}_0$) and the error (err_0). The main reason for retaining the resolution of the next iteration is that sometimes there might be more than one significant component at that given resolution. One iteration can model only one of these components. In order to model the other component, one has to perform another iteration of obtaining the best weak model at the same resolution. Increasing the resolution for the next iteration in this case might fail to model the component accurately. Only after ensuring that there are no more significant components at a given resolution, our algorithm will increase the resolution for the next iteration. Hence, the best weak model corresponding to current resolution or next higher resolution is obtained at every iteration and the model with the lowest error added to the final model. Though demonstrated for regression problems, our algorithm can also be used for classification problems by performing regression on class indicators.

The main aspect of our algorithm, which is the multi-resolution, can be seen from the fact that the resolution of the data to be modeled is either maintained or increased as the number of iterations increases. In fact, one might interpret this approach as an improvement in the weak learner alone because the algorithm proposed here will obtain improved weak learner at every iteration and hence the overall boosting will have faster convergence. We consider that our main contribution of this paper is not just at the level of choosing a weak learner but it is at

[1] Using the quasi-Newton's method the data to be modeled in the next iteration will be set to $-(I + 2rr^T)^{-1} \cdot r$.

Algorithm 3 Data-driven multi-resolution boosting

Input: Data ($\mathcal{X}$), No. of samples (N), No. of iterations (T).
Output: Final Model (F)
Algorithm:
set $res = 1$, $F = \emptyset$
for $i = 1 : d$ **do**
  /*Get the sorted data and the corresponding indices after sorting */
  $[\hat{\mathcal{X}}, idx(:, i)]$ =sort$(\mathcal{X}(:, i))$
end for
for $t = 1 : T$ **do**
  $r = L(\mathcal{Y}, F)$
  $[\hat{f}_0, err_0] = bestfit(\hat{\mathcal{X}}, r, N, d, res, idx)$
  $[\hat{f}_1, err_1] = bestfit(\hat{\mathcal{X}}, r, N, d, res * 2, idx)$
  if $err_0 < err_1$ **then**
    $F = F + \hat{f}_0$
  else
    $F = F + \hat{f}_1$
    $res = res * 2$
  end if
end for
$return\ F$

the junction of the choice of weak learner and the iterations in the boosting algorithm. Also, our algorithm obtains the weak models and models the data in a more systematic hierarchical manner. Most importantly, the increase in the resolution is monotonically non-decreasing, i.e. the resolution either remains the same or increased.

For every iteration, the best weak model is fitted to the data based on a single feature value at a given resolution. This is performed using the $bestfit$ function in the algorithm. Changing the resolution can be done either by reducing the complexity of the weak learner or by changing the number of datapoints. Changing the complexity of the weak model can be done in a very intuitive manner depending on the choice of the weak learner. For example, if decision trees are used as a weak learner, the resolution can be changed by changing the number of levels in the decision tree that is being considered. The initial boosting iterations use trees with only one level (or decision stumps) and later on the resolution can be increased by increasing the depth of the tree. One has to note that the complexity of the modeling (or classification boundary) is significantly increased by changing the resolution.

More theoretical way of approaching this problem is to use scale-space kernel to model a subset of data and handling the data in a multi-resolution way. The procedure $bestgaussfit$ (instead of $bestfit$) performs this task for a particular value of resolution. Additive modeling with smooth and continuous kernels will result in smooth functions for classifier boundary and regression functions. Gaussian kernels are a simple and a trivial choice for scale-space kernels that are powerful universal approximators. Also, Gaussian kernels allow generative modeling of a target function which is good choice for many applications. Choosing optimal kernel width during every iteration of boosting becomes a non-trivial task. In other words, one cannot predetermine the reduction in the kernel width.

The basic idea is to slide a Gaussian window across all the sorted datapoints corresponding to each feature at a given resolution (See Fig. 4). Algorithm 4 contains a loop with a nested loop inside it. The outer loop ensures that the Gaussian fit has to be computed for each feature and the inner loop corresponds to the sliding Gaussian. In other words, depending on the given resolution (indicated by n datapoints), a Gaussian kernel containing n datapoints is moved across all the datapoints and the location where the minimal residual error is obtained.

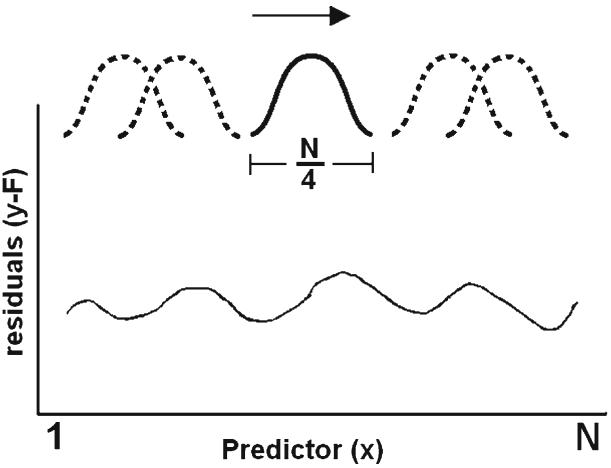

Fig. 4 Demonstration of the sliding Gaussian kernel. There are N datapoints and each has its corresponding response as a residual (y-F). A Gaussian kernel of resolution $\frac{N}{4}$ is slided across the entire range of datapoints to find the optimal fit. The optimal fit is obtained by minimizing the L_2 norm between the residual and the weighted Gaussian fit. The *dotted* Gaussians represent the sliding windows, while the *dashed* Gaussian represent the active window

Algorithm 4 *bestgaussfit*

1: **Input:** Sorted feature data ($\hat{\mathcal{X}}$), No. of samples(N), No. of samples to fit Gaussian (n), Residual vector (r), Sorting indices (idx).
2: **Output:** Best fit Regressor ($\hat{f}$), Error (Err_{min})
3: **Algorithm:**
4: Err_{min} =MAXDOUBLE
5: **for** $i = 1 : d$ **do**
6: **for** $j = 1 : N - n + 1$ **do**
7: $\hat{x} = \hat{\mathcal{X}}(:, j : j + n - 1)$
8: $\hat{r} = r(idx(j : j + n - 1, i))$
9: $wgt(1 : n) = abs(\hat{r}(1 : n))/sum(abs(r))$
10: $\mu = E_{wgt}(\hat{x}) = wgt^T * \hat{x}$
11: $\sigma = \text{sqrt}(E_{wgt}((\mu - \hat{x})^2))$
12: $f =$normpdf($\hat{\mathcal{X}}, \mu, \sigma$)
13: $\beta =$sum($\hat{r}$)/sum($f(j : j + n - 1)$)
14: $err = (r - \beta f)^T \cdot (r - \beta f)$
15: **if** $err < Err_{min}$ **then**
16: $Err_{min} = err$
17: $\hat{f} = f$
18: **end if**
19: $fp = min(\hat{f}(1 : d))$
20: **end for**
21: **end for**
22: return $\{fp, Err_{min}\}$

Within each active window, the weight of individual datapoint is computed based on the normalized value.[2]

[2] The weak learner is being fit to the sorted feature data. Hence, the corresponding value must also be sorted using the idx values. This is done in line 8 of the $bestgaussfit$ procedure.

The result f is obtained by fitting a normal distribution computed using weighted mean (μ) and standard deviation (σ) for the datapoints within this window. After obtaining the weak learner, it must be scaled (scale factor is β) according to the target values. Finally, the error is computed between the weak learner and the target values. It should be noted that the error is not computed using the given window alone. In other words, Gaussian kernel that will fit the data is computed based on the parameters computed within the window, and the final error is computed based on the deviation in the entire dataset (not just the active window). This is a non-trivial aspect in our algorithm and failing to perform this might lead to erroneous results that might fit the data locally. This is due to the fact that every iteration, the result will be a local fit to the data and each of this weak learner might not perform well globally. Finally, the best weak learner across each feature that has the minimum error is returned.

The resolution is increased at a logarithmic scale. i.e. every time the resolution is doubled or the number of datapoints considered to fit a Gaussian is halved. In fact, we can use any other heuristic to change the resolution more efficiently. Experimental results indicated that this change of resolution is optimal and also this logarithmic change of resolution has nice theoretical properties.

5 Experimental results

We will now demonstrate our empirical results on both synthetic and real-world datasets. We will also demonstrate the use of data-driven framework to solve regression problems. All of our experiments were run in MATLAB 7.0 and on a pentium IV 2.8 GHz machine. Six different real-world binary classification datasets were chosen from the UCI machine learning repository [5]. Two different sets of experiments were conducted on these datasets to illustrate the power of multi-resolution boosting. To demonstrate the model-driven framework, decision trees at multiple resolutions (different number of levels in the decision tree) are considered, and to demonstrate the data-driven framework, scale-space kernels are considered for fitting the data at multiple resolutions.

5.1 Results for model-driven multi-resolution boosting

Figure 5 shows the results of the multi-resolution boosting algorithm on a synthetic two-dimensional synthetic binary classification example. Due to the increase in the number of levels in the decision trees used as weak learners at each iteration, the complexity of the classification boundary gradually increases as the boosting iterations progress. Figure 6 shows the test error results on different datasets during the boosting iterations. Comparisons are made between the standard Adaboost and the multi-resolution boosting framework. One can see that the error obtained in the multi-resolution boosting procedure is significantly lower compared to the standard procedure. Under this framework, during the initial iterations of boosting, decision stumps (trees with only one level of child nodes) are used. As the iterations proceed, more deeper trees (with levels greater than 2) are used for modeling. This way, a hierarchical approach is used for computing the classification boundary from low resolution to high resolution. For example, using a tree with many levels in the first few iterations in the boosting procedure might obtain a very complicated decision boundary.

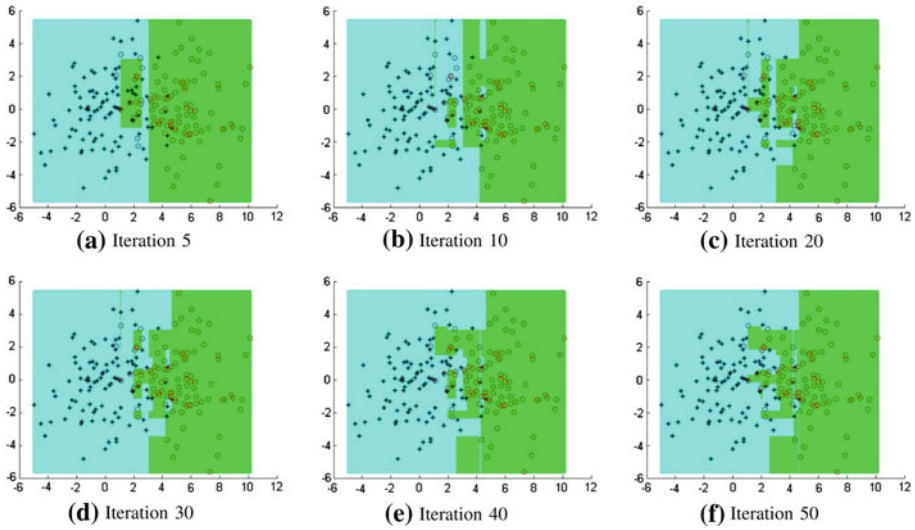

Fig. 5 Experimental results of the multi-resolution boosting framework along with the number of iterations using a synthetic two-dimensional binary dataset. '*' Indicates datapoints that belong to class 1 and 'o' indicates datapoints that belong to class 2. The classification boundaries and the regions obtained during various iterations of the data-driven multi-resolution boosting algorithms are indicated by different *shaded regions*

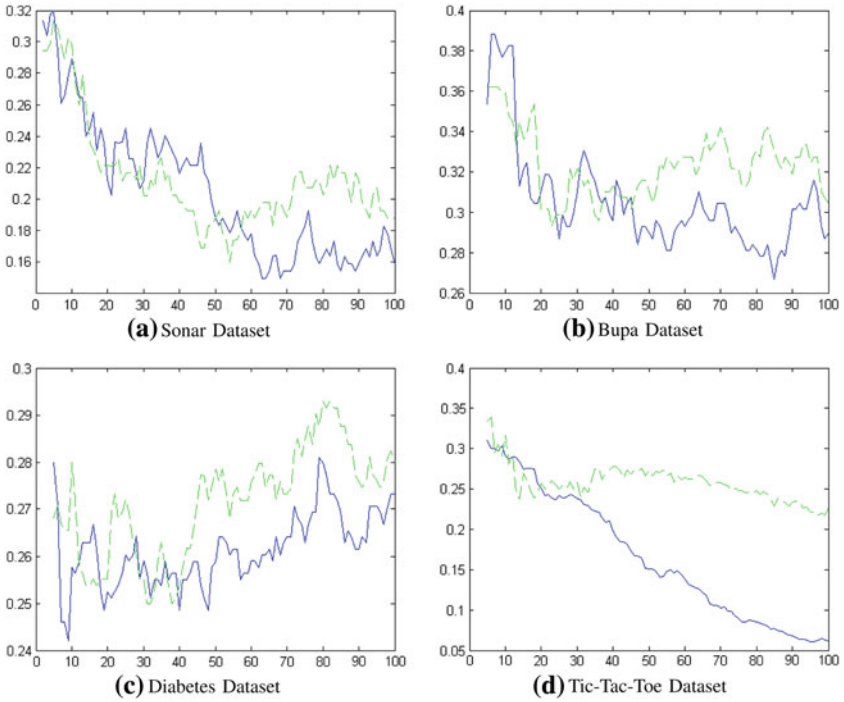

Fig. 6 Test error during the boosting iterations on various datasets. The *dashed line* indicates the error obtained using the standard Adaboost algorithm and the *solid line* indicates the error obtained using the multi-resolution boosting algorithm. The boosting iterations are on the *x*-axis and the MSE is on the *Y*-axis

5.2 Results on data-driven multi-resolution boosting

We demonstrate the power of data-driven multi-resolution approach using scale-space kernels on binary classification problems. Experiments on multi-class classification problems can also be performed using methods similar to [1]. Additive modeling with smooth and continuous kernels will result in smooth functions for classifier boundary and regression functions. Since obtaining the width of the kernel during the boosting process can be a challenging task, the use of scale-space kernels can resolve the problem using adaptive step-sizes by a '*global-to-local*' fitting process. One cannot predetermine the reduction in the kernel width.

We compare the performance of these scale-space kernels with other static and dynamic kernels. Exhaustive kernel is the most expensive one which tries to fit a kernel of various widths during each iteration of boosting. Dynamic kernel (or random kernel) fits a kernel of random width during the boosting process. Static kernels will have fixed width that does not change during the boosting process. The multi-resolution boosting approach automatically chooses the optimal width of the kernel and does not require any input from the user. This is a vital component of the proposed approach since the task of choosing the appropriate kernel width is the most computationally intensive part. Also, in the case of model-driven multi-resolution boosting, the user does not need to specify the complexity of a weak learner (e.g. the depth of a tree in tree-based learners). The algorithm will gradually increase the complexity of the weak learners starting from simple to complex learners in a systematic manner. This is one of the most critical parameters in boosting approaches and does not require any sort of user involvement in our framework. Hence, the proposed model is flexible with minimal user involvement. Our model still allows for any kind of weak learner, and hence, it is flexible with any form of weak learners.

Figure 7 shows the Convergence of boosting algorithms with different kernels. Exhaustive kernel is the fastest and the best kernel in terms of convergence of the training error. Scale-space kernel converges faster than all other kernels except the exhaustive kernel. Experiments were conducted using fivefold cross-validation method, and the test results are reported for the algorithm using various kernels.

We compare the performance of these scale-space kernels with other static and dynamic kernels. Exhaustive kernel is the most expensive one which tries to fit a kernel of various widths during each iteration of boosting. Dynamic kernel (or random kernel) fits a kernel of random width during the boosting process. Static kernels will have fixed widths that do not change during the boosting procedure. Compared to other static kernels of fixed width, the scale-space kernel does not suffer from the generalization problem as clearly illustrated by the results on the test data shown in Table 2. To compare the results using different kernels, we show the root mean square error (RMSE) between the class labels and the output of the model, though we used the classification datasets. Scale-space kernel consistently performs better than the exhaustive or dynamic kernels. For some datasets, wider static kernels perform better than the scale-space kernel and for other datasets static kernels with lesser width perform better. However, scale-space kernels are competitive with the best possible kernels and can be generically used for any dataset. Overall, the scale-space kernels are two times cheaper than the static width kernels. One can also see that the results of the scale-space kernels are fairly robust compared to other kernels. This multi-resolution framework provides a systematic hierarchical approach for obtaining the classification boundary in the context of additive modeling. *To the best of our knowledge, this is the first effort to combine the multi-resolution concepts with additive modeling*.

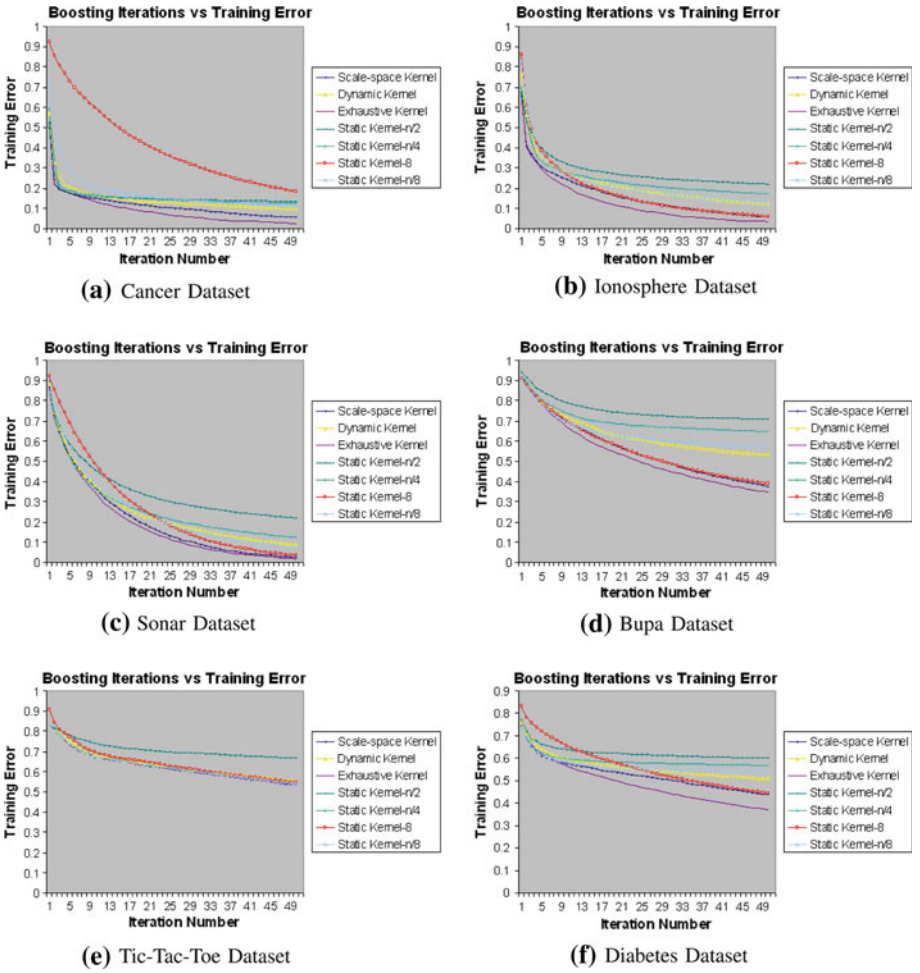

Fig. 7 Convergence of boosting algorithms with different kernels on various datasets. Exhaustive kernel is the fastest in terms of convergence and the scale-space kernel also converges relatively rapidly

Table 2 Performance of scale-space kernels with other kernels on various real-world datasets

Dataset	Cancer	Ionosphere	Sonar	Bupa	Tic-Tac-Toe	Diabetes
Number of samples	569	351	208	345	958	768
Number of features	30	34	60	6	9	8
Static kernel $-n/2$	0.44 ± 0.21	0.60 ± 0.28	0.81 ± 0.36	0.94 ± 0.28	0.83 ± 0.20	0.82 ± 0.25
Static kernel $-n/4$	0.45 ± 0.17	0.58 ± 0.28	0.82 ± 0.26	0.96 ± 0.34	0.76 ± 0.14	0.80 ± 0.24
Static kernel $-n/8$	0.49 ± 0.29	0.64 ± 0.32	0.96 ± 0.36	0.97 ± 0.25	0.75 ± 0.15	0.81 ± 0.14
Static kernel -8	0.87 ± 0.26	0.71 ± 0.24	1.07 ± 0.39	0.97 ± 0.34	0.75 ± 0.18	0.86 ± 0.23
Dynamic kernel	0.43 ± 0.18	0.60 ± 0.25	0.87 ± 0.30	0.93 ± 0.23	0.76 ± 0.19	0.82 ± 0.27
Exhaustive kernel	0.48 ± 0.25	0.65 ± 0.35	0.90 ± 0.56	0.98 ± 0.34	0.75 ± 0.20	0.81 ± 0.26
Scale-space kernel	0.43 ± 0.21	0.58 ± 0.30	0.84 ± 0.38	0.95 ± 0.36	0.75 ± 0.19	0.80 ± 0.23

Test error along with the standard deviation during fivefold cross validation is reported

5.2.1 Results on regression problems

Another set of experiments were conducted to demonstrate the effect of multi-resolution framework using the Gaussian kernels. We performed experiments on both synthetic and real-world regression datasets. Regression datasets were chosen to demonstrate the capability of our algorithm to obtain smooth functions. Using boost stumps or trees, one might be able to obtain more accurate models in terms of metrics such as mean square error and accuracy. But this will come at a cost of obtaining highly discontinuous models that are not desirable. Hence, we will now obtain smooth and continuous functions that are useful for regression problems.

We demonstrate the behavior of algorithm on two synthetic sine wave datasets: (1) noise-free sine wave and (2) the sine wave with Gaussian noise. For the first dataset, 241 samples were generated from a sine wave (with x from 0 to 12 uniformly and $y = \sin(x)$). For the second dataset, Gaussian random noise with zero mean and 0.2 standard deviation is overlaid onto all the samples of the first dataset. Figure 8 depicts the experiment results using the two synthetic sine wave datasets: (1) noise-free sine wave in the first row and (2) sine wave with Gaussian noise in the second row. In Fig. 8, the red dots represent the input data and the blue lines represent the results of the regression computed using our proposed algorithm. The green color in the second row of Fig. 8 is the noise-free sine wave which is the same as the red sine wave in the first row. We overlaid the noise-free sine wave in the second row in order to compare our results not only with the given noisy data but also with the original noise-free data.

We performed experiments using two non-linear regression datasets from NIST StRD (Statistics Reference Datasets [22]). We selected two datasets : (1) *Gauss3* from the category of average level of difficulty containing 250 samples with 1 predictor variable (x) and 1 response variable (y). (2) *Thurber* from the category of high level of difficulty containing 37 samples with 1 predictor variable (x) and 1 response variable (y). Figure 9 shows experimental results on these two datasets using the proposed data-driven multi-resolution boosting algorithm after 1, 5, 10 and 50 iterations are shown. We also ran our experiments on the following complex real-world datasets [5]:

- **Diabetes** dataset contains 43 samples with 2 predictor variables.
- **Ozone** dataset contains 330 samples with 8 predictor variables.
- **Abalone** dataset contains 4177 samples with 8 predictor variables.

The multi-resolution boosting algorithm is very effective in reducing the error quickly in the first few iterations. Significant reduction in the training error occurs within first 10 boosting iterations. By using scale-space kernels, one can achieve the optimal point (point where the over-fitting starts) within this first few iterations. Usually this point is obtained after at least 40 boosting iterations in the case of static kernels as shown in Fig. 10, which gives the convergence of the regression error during the boosting iterations. Clearly, the behavior of the convergence is similar to static kernels of very less width but the error is much lesser in the case of scale-space kernel. The main reason for using the scale-space framework is for faster convergence of the results by *dynamically choosing the weak regressors* during the boosting procedure. One can also see the comparison between the convergence behavior of a randomly chosen dynamic kernel versus the scale-space kernel. Choosing an optimal weak regressor by exploring all possibilities might yield a better result, but it will be computationally inefficient and infeasible for most of the practical problems. For such problems, scale-space kernels will give the users with a great flexibility of adaptive kernel scheme at a very low computational effort (also considering the fact of speedy convergence). The fact that the scale-space kernels converge much faster than static kernels makes them more suitable

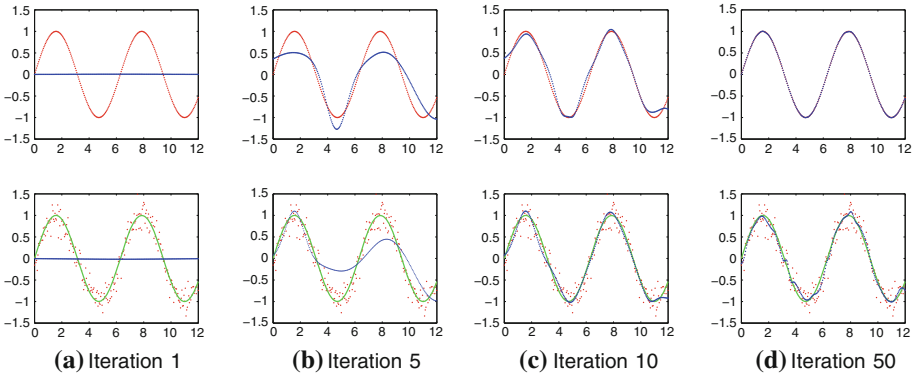

Fig. 8 Experimental results along with the number of iterations using two synthetic sinewave datasets. The first row is a sine wave without noise and the second row is a sine wave with Gaussian noise ($\mu = 0$ and $\sigma = 0.2$). As the iterations progress, our method models the original sine wave even in the presence of noise

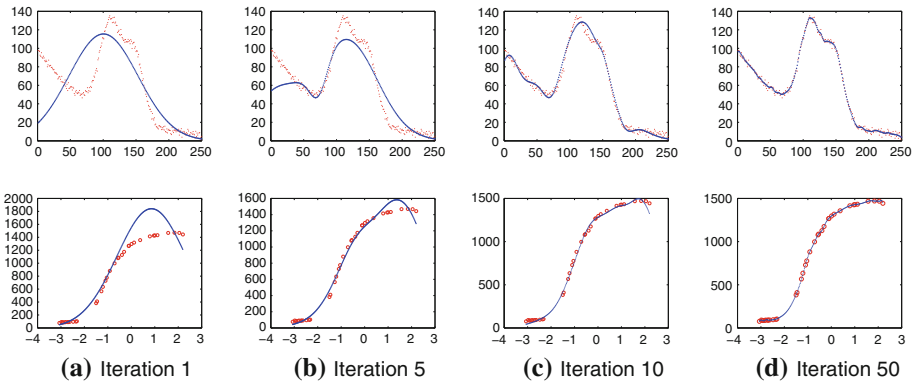

Fig. 9 Experimental results for *Gauss3* (first row) and *Thurber* (second row) datasets after 1, 5, 10 and 50 iterations

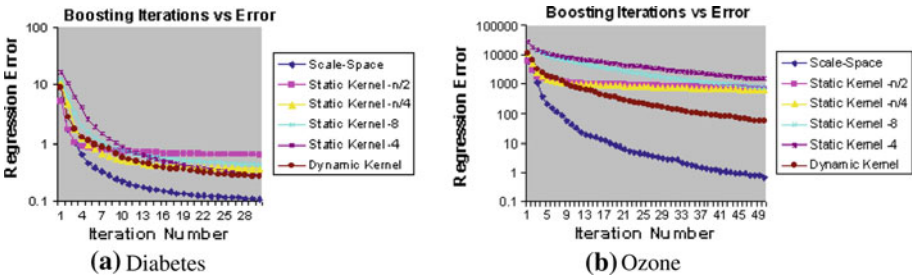

Fig. 10 Convergence of the regression error during the boosting procedure (training phase) using scale-space kernel and other static and dynamic kernels of various widths

for additive modeling algorithms. Also, one can see that the results of the scale-space kernels are fairly robust compared to other kernels.

We also demonstrate the fact that the scale-space framework does not suffer from the overfitting problem. Figure 11 shows the train and test errors during the boosting iterations along with the standard deviation using fivefold cross-validation scheme for different datasets. For

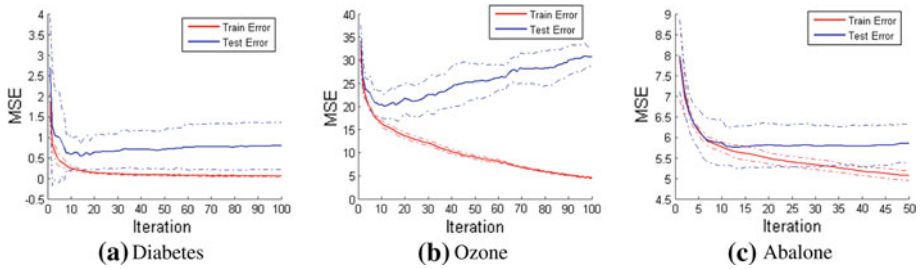

Fig. 11 Results of training and test error on different datasets using fivefold cross validation. The *solid lines* indicate the mean of the error and the *dashed lines* indicate the standard deviation in the errors

Table 3 Comparison of RMSE values for multi-Resolution boosting algorithm with other state-of-the-art regression methods on various real-world datasets. The best results are in bold

Dataset	Diabetes	Ozone	Abalone
Linear regression	**0.6277**	4.538	2.4471
Isotonic regression	0.7461	5.0	2.4604
Additive regression	0.815	4.9504	2.4439
RBF networks	0.6953	6.9133	2.8464
Multi-resolution boosting	0.6532	**4.4721**	**2.3979**

improving the computational efficiency, the sliding window kernel in the $bestgaussfit$ procedure is moved in steps of multiple datapoints rather than individual datapoints. One other advantage of using the scale-space-based boosting approach is that it obtains smooth regression functions (approximators) at different level of accuracies as shown in our results. This cannot be achieved using a decision tree or a boosting stump though they might yield lower RMSE values for prediction. Hence, our comparisons were specifically made with other smooth kernels that were used in the literature.

5.2.2 Comparison with other methods

We also compared the performance of the multi-resolution boosting algorithm with the other popular regression algorithms proposed in the literature, namely linear regression, isotonic regression, additive regression and RBF networks. All comparisons were made using the Waikato Environment for Knowledge Analysis (WEKA) software [36]. Table 3 reports the results of fivefold cross validation on different regression datasets. We can see that the multi-resolution boosting is either comparable or better than RBF networks and other methods in terms of the RMSE values. One other primary advantage of using the multi-resolution approach compared to other methods is its simplicity in modeling and learning. As shown in the graphs of Figure 11, it requires less iteration (approximately ten) for the multi-resolution algorithm to obtain the optimal model corresponding to the best test error. Hence, the final model is a combination of ten simpler models, which is easy to explain and evaluate. The advantage of boosting compared to other methods such as RBF networks and support vector machines for real-time processing is well documented in the computer vision literature [34].

6 Conclusions and future research

Recently, boosting techniques have received a great attention from several researchers working in a wide variety of applications in science and engineering. Choosing optimal weak

learners and setting their parameters during the modeling have been a crucial and challenging task. In this paper, we proposed a novel boosting algorithm that uses multi-resolution framework to obtain the optimal weak learner at every iteration. We demonstrated our results for logitboost-based regression problems on both synthetic and real datasets. We also show the derivations for the Newton's method and updates using first derivative for the regression problem in the logitboost framework with exponential L_2 norm loss function. Advantages of our method compared to other standard methods proposed in the literature are also demonstrated.

As a continuation of this work, we would like to perform the generalization of the multi-resolution approach for other additive modeling techniques. Optimal choice of the weak learner and obtaining the multi-resolution concept for any given weak learner might sometimes become a non-trivial task. Some user-friendly guidelines to obtain the amount of change in the resolution that is to be chosen adaptively during each boosting iteration must be studied more thoroughly depending on individual dataset to be modeled. Effects of different loss functions in this multi-resolution boosting framework are yet to be studied. Further investigation into more theoretical insights related to combination of additive models and multi-resolution techniques appears to be a promising research direction.

Appendix-A

Proof of Theorem 1 We will discuss the derivations for the first derivative and the second derivative and show the Newton updates in the case of the boosting for regression problems. Consider the following exponential loss function:

$$L(y, F, f) = \exp(\|y - F - f\|^2)$$

For the Newton's update equation, we need to compute the first and second derivatives with respect to $f(x)$ and evaluate them at $f(x) = 0$.

$$\begin{aligned} s(x) &= \frac{\partial L(y, F, f)}{\partial f(x)}|_{f(x)=0} \\ &= 2\exp(\|r - f\|)(r - f)|_{f=0} \\ &= 2 \cdot \exp(r^T r) \cdot r \end{aligned}$$

Taking the derivative again, we have

$$\begin{aligned} H(x) &= \frac{\partial^2 L(y, F, f)}{\partial f(x)^2}|_{f(x)=0} \\ &= 2\exp(\|r - f\|^2) \cdot I \\ &\quad + 4\exp(\|r - f\|^2) \cdot (r - f) \cdot (r - f)^T|_{f=0} \\ &= 2 \cdot \exp(r^T r) \cdot I + 4\exp(r^T r) \cdot rr^T \\ &= 2\exp(r^T r) \cdot (I + 2rr^T) \end{aligned}$$

Hence, the inverse of the Hessian becomes

$$H^{-1}(x) = \frac{(I + 2rr^T)^{-1}}{2\exp(r^T r)}$$

Finally, the Newton's update is given as follows:

$$\begin{aligned} F(x) &\leftarrow F(x) - H(x)^{-1} s(x) \\ &= F(x) - \frac{(I + 2rr^T)^{-1}}{2\exp(r^T r)} \cdot 2\exp(r^T r) \cdot r \\ &= F(x) - (I + 2rr^T)^{-1} \cdot r \end{aligned}$$

Hence, we plug-in the value $-(I + 2rr^T)^{-1} \cdot r$ as the regression value to be modeled using the weak regressor. Also, we can notice that the minimum of the loss function can also be obtained by equating the first derivative to zero.

$$2\exp(\|r - f\|)(r - f) = 0 \Rightarrow r = f$$

In other words, by modeling the residual directly using the weak regressor, the minimum of the loss function can be obtained.

References

1. Allwein E, Schapire R, Singer Y (2001) Reducing multiclass to binary: a unifying approach for margin classifiers. J Mach Learn Res 1:113–141
2. Athitsos V, Alon J, Sclaroff S, Kollios G (2008) Boostmap: an embedding method for efficient nearest neighbor retrieval. IEEE Trans Pattern Anal Mach Intell 30(1):89–104
3. Bauer E, Kohavi R (1999) An empirical comparison of voting classification algorithms: bagging, boosting, and variants. Mach Learn 36(1–2):105–139
4. Bishop CM (1995) Neural networks for pattern recognition. Oxford University Press, Oxford
5. Blake CL, Merz CJ (1998) UCI repository of machine learning databases. http://www.ics.uci.edu/mlearn/MLRepository.html, University of California, Irvine, Deptartment of Information and Computer Sciences
6. Breiman L (1996) Bagging predictors. Mach Learn 24(2):123–140
7. Breiman L (1998) Arcing classifiers. Ann Stat 26(3):801–849
8. Buhlmann P, Yu B (2003) Boosting with the l2 loss: regression and classification. J Am Stat Assoc 98(462):324–339
9. Collins M, Schapire RE, Singer Y (2002) Logistic regression, adaboost and bregman distances. Mach Learn 48(1–3):253–285
10. Duffy N, Helmbold D (2000) Leveraging for regression. In: Proceedings of 13th annual conference on computational learning theory. pp 208–219
11. Friedman JH (2001) Greedy function approximation: a gradient boosting machine. Ann Stat 29(5):1189–1232
12. Friedman JH, Hastie T, Tibshirani R (2000) Additive logistic regression: a statistical view of boosting. Ann Stat 28(2):337–407
13. Fritzke B (1994) Fast learning with incremental RBF networks. Neural Process Lett 1(1):2–5
14. Graps AL (1995) An introduction to wavelets. IEEE Comput Sci Eng 2(2):50–61
15. Hastie T, Tibshirani R, Friedman J (2001) The elements of statistical learning. Data mining, inference, and prediction, chapter boosting and additive trees. Springer, New York
16. Hong P, Liu XS, Zhou Q, Lu X, Liu JS, Wong WH (2005) A boosting approach for motif modeling using chip-chip data. Bioinformatics 21(11):2636–2643
17. Kadiyala S, Shiri N (2008) A compact multi-resolution index for variable length queries in time series databases. Knowl Inf Syst 15(2):131–147
18. Krishnaraj Y, Reddy CK (2008) Boosting methods for protein fold recognition: an empirical comparison. In: IEEE International Conference on Bioinformatics and Biomedicine. pp 393–396
19. Leung Y, Zhang J, Xu Z (2000) Clustering by scale-space filtering. IEEE Trans Pattern Anal Mach Intell 22(12):1396–1410
20. Lindeberg T (1994) Scale-space theory in computer vision. Kluwer Academic Publishers, Dordrecht
21. Mallat S (1989) A theory for multiresolution signal decomposition: the wavelet representation. IEEE Trans Pattern Anal Mach Intell 11:674–693
22. National Institute of Standards Information Technology Laboratory and Technology (NIST). Nist strd (statistics reference datasets). http://www.itl.nist.gov/div898/strd/
23. Park D (2009) Multiresolution-based bilinear recurrent neural network. Knowl Inf Syst 19(2):235–248

24. Park J-H, Reddy CK (2007) Scale-space based boosting for weak regressors. In: Proceedings of European Conference on Machine Learning, (ECML '07). Warsaw, Poland, pp 666–673
25. Preisach C, Schmidt-Thieme L (2008) Ensembles of relational classifiers. Knowl Inf Syst 14(3):249–272
26. Reddy CK, Park J-H (2008) Scale-space kernels for additive modeling. In: Joint IAPR international workshop on structural syntactic and statistical pattern recognition (SSPR & SPR). Orlando, USA, p. 714–723
27. Reddy CK, Park J-H (2009) Multi-resolution boosting for classification and regression problems. In: Proceedings of Pacific-Asia conference on knowledge discovery and data mining (PAKDD). Bangkok, Thailand, pp 196–207
28. Rudin C, Schapire RE, Daubechies I (2007) Analysis of boosting algorithms using the smooth margin function. Ann Stat 35(6):2723–2768
29. Schapire R, Singer Y, Singhal A (1998) Boosting and rocchio applied to text filtering. In: Proceedings of ACM SIGIR. pp 215–223
30. Schapire RE, Freund Y, Bartlett P, Lee WS (1998) Boosting the margin: a new explanation for the effectiveness of voting methods. Ann Stat 26(5):1651–1686
31. Schapire RE, Singer Y (1999) Improved boosting using confidence-rated predictions. Mach Learn 37(3):297–336
32. Sporring J, Nielsen M, Florack L, Johansen P (1997) Gaussian scale-space theory. Kluwer Academic Publishers, Dordrecht
33. Tieu K, Viola PA (2004) Boosting image retrieval. Int J Comput Vis 56(1–2):17–36
34. Viola PA, Jones MJ (2004) Robust real-time face detection. Int J Comput Vis 57(2):137–154
35. Webb GI (2000) Multiboosting: a technique for combining boosting and wagging. Mach Learn 40(2):159–196
36. Witten IH, Frank E (2005) Data mining: practical machine learning tools and techniques, 2nd edn. Morgan Kaufmann, Los Altos
37. Zemel RS, Pitassi T (2000) A gradient-based boosting algorithm for regression problems. In: Neural information processing systems. pp 696–702
38. Zhu J, Rosset S, Zou H, Hastie T (2005) Multi-class adaboost. Technical Report 430, Department of Statistics, University of Michigan

Author Biographies

Chandan K. Reddy is currently Assistant Professor in the Department of Computer Science at Wayne State University since 2007. He received his PhD from Cornell University and MS from Michigan State University. He is the Director of the COmputational Learning and Discovery (COLD) Laboratory and a scientific member of Karmanos Cancer Institute. His primary research interests are machine learning, data mining and computational statistics with applications to biomedical informatics and business intelligence. He has published over 30 peer-reviewed articles in leading conferences and journals including IEEE TPAMI, ACM SIGKDD, IEEE ICDM, SIAM DM and ACM CIKM. He is a member of IEEE and ACM.

Jin-Hyeong Park received the PhD degree in Computer Science and Engineering in 2004 from Pennsylvania State University. He is currently a research scientist at Siemens Corporate Research working on medical image analysis using pattern recognition and computer vision. His research interests include pattern classification, clustering, object detection and segmentation in images and video, motion segmentation, manifold learning and medical image analysis. He has co-authored scholarly publications in the leading conferences including IEEE CVPR, ICCV, ECCV and MICCAI.