New Boosting Methods of Gaussian Processes for Regression

Yangqiu Song
State Key Laboratory of
Intelligent Technology and Systems
Department of Automation
Tsinghua University
Beijing 100084, P.R.China
E-mail: songyq99@mails.tsinghua.edu.cn

Changshui Zhang
State Key Laboratory of
Intelligent Technology and Systems
Department of Automation
Tsinghua University
Beijing 100084, P.R.China
E-mail: zcs@mail.tsinghua.edu.cn

***Abstract*— Feed forward neural networks are popular tools for nonlinear regression and classification problems. Gaussian Process(GP) can be viewed as an RBF neural network which have infinite number of hidden neurons. On regression problems, they can predict both the mean value and the variance of the given sample. Boosting is one of the most important recent developments in machine learning. Classification problems have dominated research on boosting to date. On the other hand, the application of boosting of regression has received less investigation. In this paper, we develop two boosting methods of GPs for regression according to the characteristic of them. We compare the performance of our ensembles with other boosting algorithms and find that our methods are more stable and essentially have less over-fitting problems than the other methods.**

I. INTRODUCTION

Neural networks are popular tools for nonlinear regression and classification problems. Mackay[10] points that, feed forward neural networks can be viewed as defining a prior probability distribution over non-linear functions, and the neural network's learning process can be interpreted in terms of the posterior probability distribution over the unknown function. Neal[11][12] shows that the properties of an RBF neural network with one hidden layer converges to a Gaussian process as the number of hidden neurons tends to infinity when the prior of the parameters is Gaussian. Thus, we can ignore the parameters of the neural networks and work directly with the input and the output of the data. We find that Gaussian process has a good characteristic that is not possessed by many other non-parametric regressors. It can not only regress the mean value of the given point, but also can estimate the variance. In bayesian inference framework, it can use the MCMC technique or MAP estimation to solve the hyper-parameters' problem[10][12][15], compared with other kernel methods such as the Support Vector Machine(SVM)[14] which needs users give the value of the hyper-parameters. However, its training time is scaling of $O(n^3)$ and memory is scaling of $O(n^2)$, where n the number of training points, hinder their more widespread application. But recently, there have been several sparse online algorithms such as [8] and [9], which make Gaussian process more and more popular. The main work of Gaussian processes for regression is done by Williams and Rassmussen[15], Neal[11], we also follow the introduction made by MacKay[10] in this paper.

On the other hand, for a wide variety of machine learning problems, especially the classification problems, the boosting techniques have been proven to be an effective method for reducing bias and variance, and improving misclassification[1]. But the knowledge about the utility of these techniques in regression is not as much as classification. Freund and Schapire[4][5] attacks the regression problem by reducing it to a classification problem using their algorithm AdaBoost.R. Druck[2] and Ridgewary, et al[13] suggest an actual implementation and experimentation with boosting regression in which they applies an ad hoc modification of AdaBoost.R to some regression problems respectively. Friedman, et al[6] explore regression using the gradient descent approach. They explain the AdaBoost as an additive logistic regression model. They model the posterior probability as $P(x) = P(y = 1|x) = e^{F(x)}/(1+e^{F(x)})$, and use some AdaBoost algorithms to regress the additive function $F(x)$. But it is essentially a classification model. In the absence of the hypothesis that the target value is binary, the algorithm could not be explained in theory. Duffy and Helmbold examine ensemble methods by iteratively calling them on modified samples, the present several gradient descent algorithms and prove AdaBoost-like bounds on their sample errors using intuitive assumptions on the base learners. They use a decision stumps as a base learner and the main weakness of their theoretical results is the assumption that the base learner can consistently return hypotheses with a useful edge, even when the data are re-labelled by the master algorithm. Zemel and Pitassi[16] propose an analogous formulation for adaptive boosting of regression problems, utilizing a novel objective function that leads to a simple boosting algorithm. They use a back-propagation neural networks as the the base learner and prove that the method reduces the training error. But they only give a result that has only 10 iterations, in this paper, our experiment shows that, if there is one criterion to stop the algorithm, their boosting is very likely to exhibit over-fitting.

In this paper, mainly based on the work of Zemel and Pitassi[16], we present a new objective function according to the characteristic of Gaussian process which can estimate the variance of the given sample. The methods can also work with other regressors if only it can return a estimated value and a variance at the test sample. Results show that our methods are more stable than the other methods and exhibit no over-fitting problems at all. The paper is organized as follows. In section 2, we give an short introduction to Gaussian process for regression that MacKay[10] gives in their paper. Starting from section 3, we present our two methods of ensembling the regressors. Experimental results are given in section 4 and section 5 we conclude the text and show the future work.

II. A SHORT INTRODUCTION TO GAUSSIAN PROCESS

Firstly, we define some notations. Here we only introduce the basic Gaussian process for regression problems, following the methods introduced by MacKay[10]. Since Gaussian process is very similar to Bayesian neural networks, we follow the notation of them. We denote the training set of N data points as $\mathbf{X}_N \triangleq \{\mathbf{x}^{(n)}\}_{n=1}^{N}$, their target set is $\mathbf{t}_N \triangleq \{t_n\}_{n=1}^{N}$. In neural networks, we have a non-linear function $y(\mathbf{x})$, with respect to the parameter $\mathbf{w}$.

In parameter approach to regression such as RBF neural networks, we use H fixed basis functions. Let us assume that a list of N input points $\mathbf{X}_N$ has been specified and define the N$\times$H matrix $\mathbf{R}$ to be the matrix of values of the basis functions at the points: $\mathbf{R}_{nh} = \phi_h(\mathbf{x}^{(n)})$. And then define the vector $\mathbf{y}_N$ to be the vector of values of $y(\mathbf{x})$ at the N points: $y_n = \sum_h \mathbf{R}_{nh} w_h$. If the prior distribution of $\mathbf{w}$ is Gaussian with zero mean: $P(\mathbf{w}) = N(\mathbf{0}, \sigma_w^2 \mathbf{I})$, then $\mathbf{y}$, being a linear function of $\mathbf{w}$, is also Gaussion distributed with zero mean.The covariance matrix is:

$$\begin{aligned}\mathbf{Q} &= \left\langle \mathbf{y}, \mathbf{y}^T \right\rangle = \left\langle \mathbf{R}\mathbf{w}, \mathbf{w}^T \mathbf{R}^T \right\rangle \\ &= \mathbf{R} \left\langle \mathbf{w}, \mathbf{w}^T \right\rangle \mathbf{R}^T = \sigma_w^2 \mathbf{R}\mathbf{R}^T \end{aligned} \tag{1}$$

So the prior distribution of $\mathbf{y}$ is:

$$P(\mathbf{y}) = N(\mathbf{0}, \mathbf{Q}) = N(\mathbf{0}, \sigma_w^2 \mathbf{R}\mathbf{R}^T) \tag{2}$$

We call $y(\mathbf{x})$ is a Gaussian process if for any finite selection of points $\mathbf{x}^{(1)}, \mathbf{x}^{(1)}, ..., \mathbf{x}^{(N)}$, the joint density $P(y(\mathbf{x}^{(1)}), y(\mathbf{x}^{(1)}), ..., y(\mathbf{x}^{(N)}))$is Gaussian.

For regression problems, we assume the target and the corresponding estimation function differ by a Gaussian noise of variance σ_v^2: $t = y + v, P(v) = N(0, \sigma_v^2)$, then the prior distribution of target vector is also Gaussian:

$$P(\mathbf{t}) = N(\mathbf{0}, \mathbf{Q} + \sigma_v^2 \mathbf{I}) = N(\mathbf{0}, \mathbf{C}) \tag{3}$$

We denote the covariance matrix $\mathbf{C}$ by:

$$\mathbf{C} = \sigma_w^2 \mathbf{R}\mathbf{R}^T + \sigma_v^2 \mathbf{I} \tag{4}$$

When the neuron's number of the RBF neural networks increases to infinite, it is proved[10] that the entry of covariance matrix $\mathbf{C}$ has the form:

$$C(\mathbf{x}, \mathbf{x}') = \theta_1 \exp\left[-\frac{1}{2}\sum_{i=1}^{D} \frac{(x_i - x_i')^2}{r_i^2}\right] + \theta_2 \tag{5}$$

where x_i is the ith component of $\mathbf{x}$, D is the dimension of the data space, and $\mathbf{\Theta} = \{\theta_1, \theta_2, \{r_i\}\}$ is a set of hyper-parameters. We can use the MCMC technique[12][15] or a MAP estimation[10][15] to obtain them, in this paper, we adopt the latter.

We now distinguish between different sizes of covariance matrix $\mathbf{C}$ with a subscript, we define sub-matrices of $\mathbf{C}_{\mathrm{N}+1}$ as follow:

$$\mathbf{C}_{N+1} = \begin{bmatrix} \mathbf{C}_N & \mathbf{k} \\ \mathbf{k}^T & \kappa \end{bmatrix} \tag{6}$$

According to the partitioned inverse equations, we have:

$$\begin{aligned} &\mathbf{C}_{N+1}^{-1} = \begin{bmatrix} \mathbf{M} & \mathbf{m} \\ \mathbf{m}^T & \mu \end{bmatrix}, \mathbf{M} = \mathbf{C}_N^{-1} + \tfrac{1}{\mu}\mathbf{m}\mathbf{m}^T, \\ &\mu = \left(\kappa - \mathbf{k}^T \mathbf{C}_N^{-1}\mathbf{k}\right)^{-1}, \mathbf{m} = -\mu \mathbf{C}_N^{-1}\mathbf{k}. \end{aligned} \tag{7}$$

So the inference of a new target is a conditional distribution which is also Gaussian:

$$P(t_{N+1} | \mathbf{t}_N) = \frac{1}{Z}\exp\left[-\frac{(t_{N+1} - \hat{t}_{N+1})^2}{2C_{\hat{t}_{N+1}}^2}\right] \tag{8}$$

where:

$$\hat{t}_{N+1} = \mathbf{k}^T \mathbf{C}_N^{-1} \mathbf{t}_N \quad C_{\hat{t}_{N+1}}^2 = \kappa - \mathbf{k}^T \mathbf{C}_N^{-1}\mathbf{k} \tag{9}$$

Thus we get the predictive mean value and the variance which define the error bars at the new given point. Due to the characteristic of estimation of mean and variance, we develop two ensemble methods to boost the Gaussian process to get more efficient regressors. One is a AdaBoost-like algorithm, the other is a bias/variance decomposition based method.

III. ENSEMBLES OF GAUSSIAN PROCESSES

Zemel and Pitassi[16], in their paper, present a gradient based Boosting algorithm for regression problems. The method is different from AdaBoost.R which is proposed by Freund and Schapire[4][5], which is not directly used to a regression problem. And also, it is different with the methods of Duffy and Helmbold[3] which do not resample the training data through a new distribution at each iteration, but to modify the target value of the training data. They define a new objective function:

$$J_T = \frac{1}{n}\sum_{i=1}^{N}\left(\prod_{t=1}^{T}\beta_t^{-\frac{1}{2}}\right)\exp\left[\sum_{t=1}^{T}\beta_t\left(t_i - y_t(\mathbf{x}^{(i)})\right)^2\right] \tag{10}$$

This is the cost after T iterations to minimize the over all exponentiated squared error and can be viewed as minimizing the T hypothesis' error over distribution of the training data:

$$\begin{aligned} J_T &= \frac{1}{n}\sum_{i=1}^{N}\left(\left[\prod_{t=1}^{T-1}\beta_t^{-\frac{1}{2}}\right]\exp\left[\sum_{t=1}^{T-1}\beta_t\left(t_i - y_t(\mathbf{x}^{(i)})\right)^2\right]\right. \\ &\qquad \left.\beta_T^{-\frac{1}{2}}\exp\left[\beta_T\left(t_i - y_T(\mathbf{x}^{(i)})\right)^2\right]\right) \\ &= \sum_{i=1}^{N}\omega_t^{(i)}\beta_T^{-\frac{1}{2}}\exp\left[\beta_T\left(t_i - y_T(\mathbf{x}^{(i)})\right)^2\right] \end{aligned} \tag{11}$$

They compare the objective with a probabilistic expression. They point that the objective function can be viewed as a product of the reciprocal likelihood at each iteration, while the likelihood function is:

$$g(y_t|\mathbf{x}^{(i)}, M) = (2\pi\beta_t)^{-\frac{1}{2}} \exp\left[-\frac{1}{2}\beta_T\left(t_i - y_t(\mathbf{x}^{(i)})\right)^2\right] \tag{12}$$

This can be theoretically interpreted, but in practice, we find that after many iterations, their method will seriously meet a over-fitting problem if without a stop criterion. To avoid of this, we present two novel methods that can stably boost GPs and without any over-fitting problems. We first present an AdaBoost-like algorithm which modifies the objective function of Zemel and Pitassi's and develop a new method of updating the weight. Further more, we use a new method to combine regressors using the bias/variance based composition algorithm while maintain our objective function unchanged.

A. An AdaBoost-Like Ensemble

1) Algorithm: The two crucial elements of boosting algorithm are the way in which a new distribution is constructed and the way in which hypotheses are combined to produce a new output[16]. We assume that the method of ensembling the GPs follows the way which Zemel and Pitassi do:

$$\bar{y}(\mathbf{x}^{(i)}) = \frac{\sum_t \beta_t y_t(\mathbf{x}^{(i)})}{\sum_t \beta_t} \tag{13}$$

where $\mathbf{x}^{(i)}$ is the point which we want to know the target value, y_t is the hypotheses which a GP outputs at each iteration, and β_t is the combination coefficient of each GP.

Since at each iteration t the GP estimates the value of a given point is Gaussian, we can rewrite the Gaussian formulation as:

$$t_i - y_t(\mathbf{x}^{(i)}) \sim N(0, C_t(\mathbf{x}^{(i)})) \tag{14}$$

where t_i is the target value of a given point $\mathbf{x}^{(i)}$. Thus the difference between target and the hypotheses of the combined regressor is also Gaussian:

$$\begin{aligned} t_i - \bar{y}(\mathbf{x}^{(i)}) &= t - \frac{\sum_t \beta_t y_t(\mathbf{x}^{(i)})}{\sum_t \beta_t} \\ &= \frac{\sum_t \beta_t(t - y_t(\mathbf{x}^{(i)}))}{\sum_t \beta_t} \sim N(0, \frac{\sum_t \beta_t C_t(\mathbf{x}^{(i)})}{\sum_t \beta_t}) \end{aligned} \tag{15}$$

We use the reciprocal likelihood, the new objective function is:

$$J_T = \frac{1}{n}\sum_{i=1}^{N} \bar{C}_T^{\frac{1}{2}}(\mathbf{x}^{(i)}) \exp\left[\frac{1}{2}\frac{\left(t_i - \bar{y}_T(\mathbf{x}^{(i)})\right)^2}{\bar{C}_T(\mathbf{x}^{(i)})}\right] \tag{16}$$

where:

$$\bar{y}_T(\mathbf{x}^{(i)}) = \frac{\sum\limits_{t=1}^{T} \beta_t y_t(\mathbf{x}^{(i)})}{\sum_t \beta_t}, \quad \bar{C}_T(\mathbf{x}^{(i)}) = \frac{\sum\limits_{t=1}^{T} \beta_t C_t(\mathbf{x}^{(i)})}{\sum_t \beta_t} \tag{17}$$

The minimizing of this function (16) is equal to maximizing the likelihood function of the combined regressor. Since we can not decompose the function J_T to a function of $y_1 ... y_{t-1}$ and a function of y_t, we need some approximation.

Zemel and Pitassi's method has an intuitional meaning that the weight is proportional to the reciprocal likelihood: when at the former $t-1$ iterations the likelihood's product is small at a given training point, meaning that the uncertainty of the regression result is large, so the weight of the point is large which makes more points are re-sampled in the next iteration. Fig.1 shows that the variance is a function of the training data's density, the error bars increase where the data point density decreases. We simply modify the update rule of the weights to get some interesting results. At iteration t, we assume the weight is proportional to a function of the average predicting variance which is estimated using the former $t-1$ iterations:

$$\omega_t^{(i)} \propto \bar{C}_{t-1}^{\frac{1}{2}}(\mathbf{x}^{(i)}) \exp(-\bar{C}_{t-1}(\mathbf{x}^{(i)})) \tag{18}$$

When the average variance is varying from 0 to 1, the weight is a monotonously increasing function with the variable. In practice, we find that the average variance varies not so suddenly as the likelihood function, and this method will not cause the accumulation of the weight which will cause the over-fitting problem. According to the constraint of the objective function, our Boosting can get a more stable result. The flow chart of the algorithm is shown in Table I.

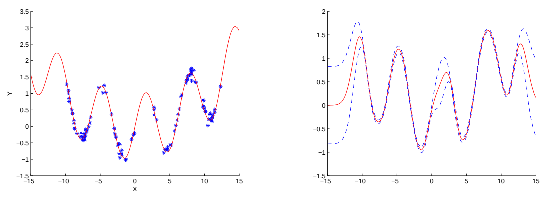

(a) training data (b) regression result

Fig. 1. A simple function regression result. Left: the training data and the true function; Right: the GP regression result(include mean and variance).

TABLE I
ADABOOST-LIKE ENSEMBLE ALGORITHM

```
1.Input:
    Training set examples $\{\mathbf{x}^{(n)}, t_n\}_{n=1}^{N}$.
    Base learner: Learning a Gaussian process produces a hypothesis.
2.Choose an initial distribution $\omega_1^{(i)} = \frac{1}{n}$.
3.Iterate:
    a) Learn a Gaussian process to regress a function
with distribution $\omega_t^{(i)}$.
    b) Set $0 \leq \beta_t \leq 1$to minimize$J_t$.
    c) Update training distribution:
            $\omega_{t+1}^{(i)} \propto \bar{C}_t^{\frac{1}{2}}(\mathbf{x}^{(i)}) \exp(-\bar{C}_t(\mathbf{x}^{(i)}))$.
4.Estimate the output : $\bar{y}_T(\mathbf{x}^{(i)}) = \frac{\sum_{t=1}^{T} \beta_t y_t(\mathbf{x}^{(i)})}{\sum_t \beta_t}$.
```

2) An EM View: According to our method, we can rewrite the objective function as:

$$J_T = \frac{1}{n}\sum_{i=1}^{N} \omega_T^{(i)} \exp\left[\frac{1}{2}\left(t_i - \bar{y}_T(\mathbf{x}^{(i)})\right)^2\right] \tag{19}$$

$\omega_T^{(i)}$ is not same the weight we mention before, but a function of predictive variance, we rewrite formula (18) as:

$$\omega_t^{(i)} \propto \hat{C}^{\frac{1}{2}}(\mathbf{x}^{(i)}) \exp(-\hat{C}(\mathbf{x}^{(i)})) \tag{20}$$

where $\hat{C}$ is the predictive variance at each iteration. Then the new objective function can be viewed as an exponential squared error over a certain distribution L. Hence, we can infer some interesting results from the squared error:

$$\begin{aligned}
&(t_i - \bar{y}_T(\mathbf{x}^{(i)}))^2 \\
&= (\frac{\sum_{t=1}^{T-1}\beta_t(t_i - \bar{y}_{T-1}(\mathbf{x}^{(i)}) + \beta_T(t - y_T(\mathbf{x}^{(i)}))}{\sum_{t=1}^{T-1}\beta_t + \beta_T})^2 \\
&\leq 2\frac{(\sum_{t=1}^{T-1}\beta_t)^2(t_i - \bar{y}_{T-1}(\mathbf{x}^{(i)}))^2 + \beta_T^2(t - y_T(\mathbf{x}^{(i)}))^2}{(\sum_{t=1}^{T-1}\beta_t + \beta_T)^2}
\end{aligned} \tag{21}$$

We then use a EM view to examine the objective function (19). At E-step, we estimate latent variables: the mean value $\bar{y}_{T-1}$ and the average predictive variance $\bar{C}_{T-1}(\mathbf{x}^{(i)})$, and then use a function $\omega_t^{(i)} \propto \hat{C}^{\frac{1}{2}}(\mathbf{x}^{(i)})\exp(-\hat{C}(\mathbf{x}^{(i)}))$ to generate a distribution to re-sample the data set. At M-step, we want to find a new regressor $\bar{y}_T$ that have minimum squared error with a optimal parameter β_T. The parameter β_T is to make the objective function (16) minimized, that is to make the likelihood maximized. According to the right side of the inequality (21), the M-step can be viewed as another objective function which is generated after re-sampling. We can regard it as to optimize the function $\left(t - y_T(\mathbf{x}^{(i)})\right)^2$ under the constraint of $\left(t_i - \bar{y}_{T1}(\mathbf{x}^{(i)})\right)^2 = 0$ where the Lagrange factor is :

$$\frac{\left(\sum_{t=1}^{T-1}\beta_t\right)^2}{\beta_T^2} \tag{22}$$

B. Bias/Variance Decomposition Based Boosting

In the previous sub-section we give an ensemble method which is similar to Zemel and Pitassi's. But we do not use a decomposition form of the weight and the new regressor with the objective function (16). Instead, we use an ad hoc technique that intuitionally uses a function of variance to update the weights. Heskes[7] in his paper proves an interesting conclusion: the mean squared error is a special case of the Kullback-Leibler divergence of his bias/variance decomposition model. If a regressor can estimate the mean value and the variance of the given point, we can use a new ensemble method to obtain the average:

$$\begin{aligned}
\bar{C}_T(\mathbf{x}^{(i)}) &= \frac{1}{\sum_t 1/C_t(\mathbf{x}^{(i)})} \\
\bar{y}_T(\mathbf{x}^{(i)}) &= \bar{C}_T(\mathbf{x}^{(i)})\sum_{t=1}^{T}\frac{y_t(\mathbf{x}^{(i)})}{C_t(\mathbf{x}^{(i)})}
\end{aligned} \tag{23}$$

Then the decomposition yields:

$$\begin{aligned}
&E\left[\frac{(y_t(\mathbf{x}^{(i)}) - t_i)^2}{C_t(\mathbf{x}^{(i)})} + \frac{1}{2}\log C_t(\mathbf{x}^{(i)})\right] \\
&= \left[\frac{(\bar{y}(\mathbf{x}^{(i)}) - t_i)^2}{\bar{C}(\mathbf{x}^{(i)})} + \frac{1}{2}\log \bar{C}(\mathbf{x}^{(i)})\right] \\
&+ E\left[\frac{(y_t(\mathbf{x}^{(i)}) - \bar{y}(\mathbf{x}^{(i)}))^2}{C_t(\mathbf{x}^{(i)})} + \frac{1}{2}\log\frac{C_t(\mathbf{x}^{(i)})}{\bar{C}(\mathbf{x}^{(i)})}\right]
\end{aligned} \tag{24}$$

The first term between brackets on the righthand side is the error of the average model, the second term measures the variance of the different estimators. And also the first term of the righthand is a logarithmic form of our objective function. We rewrite the formula (24) in the exponential form as:

$$\begin{aligned}
J_T &= \frac{1}{n}\sum_{i=1}^{N}\bar{C}_T^{\frac{1}{2}}(\mathbf{x}^{(i)})\exp\left[\frac{1}{2}\frac{\left(t_i - \bar{y}_T(\mathbf{x}^{(i)})\right)^2}{\bar{C}_T(\mathbf{x}^{(i)})}\right] \\
&= \frac{1}{n}\sum_{i=1}^{N}\exp\left\{E\left[\frac{1}{2}\frac{(t_i - y_t(\mathbf{x}^{(i)}))^2}{C_t(\mathbf{x}^{(i)})} + \frac{1}{2}\log C_t(\mathbf{x}^{(i)})\right]\right. \\
&\quad \left. - E\left[\frac{1}{2}\frac{(y_t(\mathbf{x}^{(i)}) - \bar{y}(\mathbf{x}^{(i)}))^2}{C_t(\mathbf{x}^{(i)})} + \frac{1}{2}\log\frac{C_t(\mathbf{x}^{(i)})}{\bar{C}(\mathbf{x}^{(i)})}\right]\right\} \\
&(Jensen's\ inequality) \\
&\leq \frac{1}{n}\sum_{i=1}^{N}E\exp\left\{\left[\frac{1}{2}\frac{(t_i - y_t(\mathbf{x}^{(i)}))^2}{C_t(\mathbf{x}^{(i)})} + \frac{1}{2}\log C_t(\mathbf{x}^{(i)})\right]\right. \\
&\quad \left. - \left[\frac{1}{2}\frac{(y_t(\mathbf{x}^{(i)}) - \bar{y}(\mathbf{x}^{(i)}))^2}{C_t(\mathbf{x}^{(i)})} + \frac{1}{2}\log\frac{C_t(\mathbf{x}^{(i)})}{\bar{C}(\mathbf{x}^{(i)})}\right]\right\}
\end{aligned} \tag{25}$$

We find that between the exponential brackets the first term can not be used to estimate the new regressor, therefore we simply use a new regressor to replace the former $T-1$ iterations'. This instant estimate which replaces the expectation leads to a new boosting algorithm. We develop a new objective function as:

$$\begin{aligned}
J_T^{new} &\triangleq \frac{1}{n}\sum_{i=1}^{N}E\left[\frac{\left(C_T^{\frac{1}{2}}(\mathbf{x}^{(i)})\right)\exp\left\{\frac{1}{2}\frac{\left(t_i - y_T(\mathbf{x}^{(i)})\right)^2}{C_T(\mathbf{x}^{(i)})}\right\}}{\exp\left\{\frac{1}{2}\frac{(y_t(\mathbf{x}^{(i)}) - \bar{y}(\mathbf{x}^{(i)}))^2}{C_t(\mathbf{x}^{(i)})} + \frac{1}{2}\log\frac{C_t(\mathbf{x}^{(i)})}{\bar{C}(\mathbf{x}^{(i)})}\right\}}\right] \\
&= \sum_{i=1}^{N}\omega_T^{(i)}\left(C_T^{\frac{1}{2}}(\mathbf{x}^{(i)})\right)\exp\left\{\frac{1}{2}\frac{\left(t_i - y_T(\mathbf{x}^{(i)})\right)^2}{C_T(\mathbf{x}^{(i)})}\right\}
\end{aligned} \tag{26}$$

where

$$\begin{aligned}
\omega_T^{(i)} &\propto E\frac{1}{\exp\left\{\frac{1}{2}\frac{(y_t(\mathbf{x}^{(i)}) - \bar{y}(\mathbf{x}^{(i)}))^2}{C_t(\mathbf{x}^{(i)})} + \frac{1}{2}\log\frac{C_t(\mathbf{x}^{(i)})}{\bar{C}(\mathbf{x}^{(i)})}\right\}} \\
&\approx \sum_{t=1}^{T-1}\left(\frac{C_t(\mathbf{x}^{(i)})}{\bar{C}(\mathbf{x}^{(i)})}\right)^{-\frac{1}{2}}\exp\left[-\frac{1}{2}\frac{(y_t(\mathbf{x}^{(i)}) - \bar{y}_t(\mathbf{x}^{(i)}))^2}{C_t(\mathbf{x}^{(i)})}\right]
\end{aligned} \tag{27}$$

Note that we express the objective function as a decomposition form which has a weight $\omega_T^{(i)}$ and a new reciprocal likelihood. The weight is approximate to the likelihood of each regressor to their average. This indicates that the weight is large at the stable points of the regressors. The flow chart of this algorithm is shown in Table II.

TABLE II
BIAS/VARIANCE DECOMPOSITION BASED BOOSTING

```
1.Input:
    Training set examples $\{\mathbf{x}^{(n)}, t_n\}_{n=1}^{N}$.
    Base learner: Learning a Gaussian process produces a hypothesis.
2.Choose an initial distribution $\omega_1^{(i)} = \frac{1}{n}$.
3.Iterate:
    a) Learn a Gaussian process to regress a function
  with distribution $\omega_t^{(i)}$.
    b) Update training distribution:
        $\omega_{t+1}^{(i)} \propto \sum_{j=1}^{t} \left(\frac{C_j(\mathbf{x}^{(i)})}{\bar{C}(\mathbf{x}^{(i)})}\right)^{-\frac{1}{2}} \exp\left[-\frac{1}{2}\frac{(y_j(\mathbf{x}^{(i)})-\bar{y}_j(\mathbf{x}^{(i)}))^2}{C_j(\mathbf{x}^{(i)})}\right].$
4.Estimate the output :
        $\bar{C}_T(\mathbf{x}^{(i)}) = \frac{1}{\sum_t 1/C_t(\mathbf{x}^{(i)})}$,
        $\bar{y}_T(\mathbf{x}^{(i)}) = \bar{C}_T(\mathbf{x}^{(i)}) \sum_{t=1}^{T} \frac{y_t(\mathbf{x}^{(i)})}{C_t(\mathbf{x}^{(i)})}$.
```

IV. MAIN RESULTS

In this section we show some results to see how the methods above perform. We also implement two methods which are given by Duffy and Helmbold in [3]: SquareLevR and ExpLev, and also Zemel and Pitassi' algorithm[16] naming nips-boosting. We use a MSE(mean squared error) to be a criterion function. In fig.2 the $\ln(MSE)$ is adopted to be the output of each iteration. We test our algorithm using the following functions and data sets:

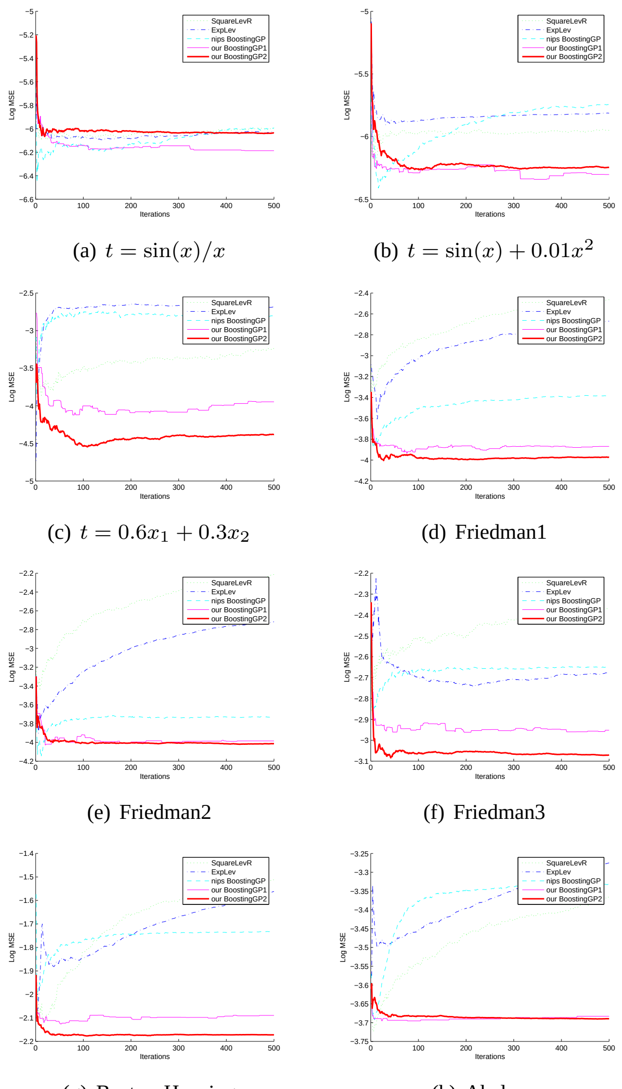
1) Sinc function $t = \sin(x)/x + \varepsilon$:
   $x \in U[-10, 10], \varepsilon \sim N(0, 0.1)$.
2) $t = sin(x) + 0.01x^2 + \varepsilon$:
   $x \in U[-10, 10], \varepsilon \sim N(0, 0.1)$.
3) A Plane $t = 0.6x_1 + 0.3x_2 + \varepsilon$:
   $x \in U[-1, 1], \varepsilon \sim N(0, 0.5)$.
4) Friedman1:
   $t = 10\sin(\pi x_1 x_2) + 20(x_3 - 0.5)^2 + 10x_4 + 5x_5 + \varepsilon$.
   $x_1 \sim x_5 \in U[0, 1]$, $\varepsilon \sim N(0, 0.2)$.
5) Friedman2:
   $$t = \left[x_1^2 + \left(x_2 x_3 - \frac{1}{x_1 x_4}\right)^2\right]^{\frac{1}{2}} + \varepsilon.$$
6) Friedman3:
   $t = \tan^{-1}\frac{x_2 x_3 - \frac{1}{x_1 x_4}}{x_1} + \varepsilon$.
   For both 5 and 6: $x_1 \in U[0, 100]$
   $x_2 \in U[40\pi, 560\pi]$
   $x_3 \in U[0, 1]$
   $x_4 \in U[1, 11]$,
   $\varepsilon \sim N(0, 0.2)$.
7) Boston Housing: This dataset contains information collected by the U.S Census Service concerning housing in the area of Boston Mass. It comprises 506 examples with 14 variables and the results are given for the task of predicting the median house value from the other 13 variables. We normalize the target value within the range from -1 to 1.
8) Abalone: This data set comes from the UCI repository of machine learning databases. The task is to predict the age (number of rings) of abalone from physical measurements. We treat the output as a continuous variable, even though it is a positive integer with a maximum value of 29. The input variables is 8 dimension vectors, the target is the normalized value of a fish's age. There are totally 4177 examples in the data set.

For function 5, 6 and data set 8, we simply normalize the input vectors to the range from -1 to 1, while for data set 7, being too many variables of 0, is performed PCA to reduce to 5 dimensions. For function 1 to 6, we select 100 random samples for training and 500 random samples for testing. For data set 7, we randomly select 100 samples for training and the rest for testing. For data set 8, we randomly select 300 samples for training and 2000 samples for testing.

TABLE III
MSE RESULT OF 8 DATASETS

	SquareLevR	ExpLev	Nips	Our 1	Our 2
$t = \sin(x)/x$	0.0025	0.0024	0.0025	**0.0021**	0.0024
$t = \sin(x) + 0.01x^2$	0.0026	0.0030	0.0032	**0.0018**	0.0019
$t = 0.6x_1 + 0.3x_2$	0.0396	0.0682	0.0609	0.0193	**0.0125**
Friedman1	0.0851	0.0692	0.0340	0.0209	**0.0188**
Friedman2	0.1094	0.0660	0.0239	0.0186	**0.0181**
Friedman3	0.0940	0.0688	0.0705	0.0522	**0.0464**
Boston Housing	0.2203	0.2096	0.1768	0.1238	**0.1139**
Abalone	0.0345	0.0378	0.0357	0.0251	**0.0250**

Table III shows the main results of 8 functions and data sets, each row has 5 algorithms implementation playing on the signed data set as column 1 shows. The best MSE value is highlighted with a bold font. Fig.2 shows the 500 iterations of each algorithm played on each data set. Note that for simple functions, SquareLevR and ExpLev will converge to some fixed value which may be not the best. For complex functions and real data sets, these two algorithms will diverge to some unexpectable value. This is an over-fitting problem because these algorithm will modify the target which is relative to the residual of the hypotheses value at each iteration, then use an additive model to represent the final result, so the learner will learn until the residual is 0. Nips-boosting rapidly descends with in 10 or 20 steps, but it will also over-fit after 20 or 30 iterations. The reason is that, after several steps of iterations, the weight will accumulate at some certain samples. The product of the reciprocal likelihood is very large at some samples and will not diminish any more.



(a) $t = \sin(x)/x$ (b) $t = \sin(x) + 0.01x^2$

(c) $t = 0.6x_1 + 0.3x_2$ (d) Friedman1

(e) Friedman2 (f) Friedman3

(g) Boston Housing (h) Abalone

Fig. 2. 500 iterations on 8 data sets of 5 algorithms

Compared with the former three, our algorithm has no over-fitting problems at all, and the descending speed is also comparable to the other three. The AdaBoost-like ensemble method is the slowest of five since we use a Genetic Algorithm(GA) to optimize the objective function (16) to get a new β_t. In practice, after several iterations, most of the parameter β_t is 0 or 1, we assume β_t only to be 0 or 1, then we can get a bagging-like expression. When t tends to infinity, we have:

$$\sum_t \beta_t y_t(\mathbf{x}^{(i)}) \Big/ \sum_t \beta_t \Big|_{t\to\infty} \to E(y) \tag{28}$$

This shows that, if at $T-1$ iteration our combined regressor is very nearly to the expectation $E(y)$, then the parameter β_t is very likely to be 0. On the other hand, the bias/variance decomposition based Boosting algorithm dose not optimize an objective function over a parameter, but to update the weight and get a new hypotheses of Gaussian process over the weight. So this algorithm will rapidly converge to a certain value. And for a majority of the results, this method is better to the AdaBoost-like ensemble method.

V. Conclusions

In this paper, we first give an AdaBoost-like algorithm which modifies the objective function of Zemel and Pitassi's and develop a new method of updating the weight. Secondly, we adopt a new method to combine regressors using the bias/variance based composition algorithm while maintain our objective function unchanged. Experimental results indicate that our methods are comparable with others on descend speed and will not exhibit any over-fitting problems. But our methods also have some problems. Firstly, we use some ad hoc techniques to update the weights of the training samples, especially in bias/variance decomposition based Boosting method, it has no interpretation in theory; Secondly, we only show the algorithms and the experimental results while do not give any proof of convergence. In the future work, we will prove our algorithm in theory.

Boosting of regressors is not received as much attention as the classification problems, we give two Boosting methods and obtain some interesting results. We expect that the boosting methods of regression will be paid more attention in the future.

References

[1] E. Bauer and R. Kohavi, "An empirical comparison of voting classification algorithms: Bagging, boosting, and variants," *Machine Learning*, vol. 36:1/2, pp. 105–39, 1999.

[2] H. Drucker, "Improving regressors using Boosting techniques," *Proceedings of the 14th International Conference on Machine Learning*, pp. 107–115, 1997.

[3] N. Duffy and D. Helmbold, "Boosting methods for regression," *Machine Learning*, vol. 47, pp. 153–200, 2002.

[4] Y. Freund and R. E. Schapire, "Experiments with a new Boosting algorithm," *In Proc. 13th International Conference on Machine Learning* pp. 148–156, San Matco, CA: Morgan Kaufmann, 1996.

[5] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *Journal of Computer and System Sciences*, vol. 55:1, pp. 119–139, 1997.

[6] J. Friedman, T. Hastie and R. Tibshirani, "Additive logistic regression: A statistical view of boosting," *The Annals of Statistics*, vol. 28:2, pp. 337–374, 2000.

[7] T. Heskes, "Bias-variance decompositions for likelihood-based estimators," Neural Computation, 10, pp. 1425–1433, 1998.

[8] Lehel Csat'o and Manfred Opper, "Sparse online Gaussian processes," *Neural Computation*, vol. 14 pp. 641–668, 2002.

[9] N. D. Lawrence , M. Seeger and R. Herbrich, "Fast sparse Gaussian process methods: the informative vector machine," *Advances in Neural Information Processing Systems 15*, MIT Press, Cambridge, MA, 2003.

[10] D. MacKay, "Introduction to Gaussian processes," *Technical Report*, Cambridge University, UK, 1997.

[11] R. M. Neal, "Bayesian Learning for Neural Networks," *in Lecture Notes in Statistics*, Springer, New York, 1996

[12] R.M. Neal, "Monte Carlo implementation of Gaussian process models for Bayesian classification and regression," *Technical Report 9702*, Department of Statistics, University of Toronto, January, 1997.

[13] G. Ridgeway, D. Madigan and T. Richardson, "Boosting methodology for regression problems," In D. Heckerman, and J. Whittaker(Eds.), *Proc. Artificial Intelligence and Statistics*, pp. 152–161, 1999

[14] V. Vapnik, The Nature of Statistical Learning Theory. Springer, 1995.

[15] C. K. I. Williams and C. E. Rasmussen, "Gaussian processed for regression," *In Advances in Neural Information Processing Systems 8*, MIT Press, 1996.

[16] R. S. Zemel and T. Pitassi, "A gradient-based boosting algorithm for regression problems," In *Advances in Neural Information Processing Systems, 13*, Cambridge, MA. MIT Press, 2001.