Image Based Regression Using Boosting Method

Shaohua Kevin Zhou, Bogdan Georgescu, Xiang Sean Zhou, and Dorin Comaniciu
Siemens Corporate Research, Integrated Data Systems Department
755 College Road East, Princeton, NJ 08540
{shaohua.zhou, bogdan.georgescu, xiang.zhou, dorin.comaniciu}@siemens.com

Abstract

We present a general algorithm of image based regression that is applicable to many vision problems. The proposed regressor that targets a multiple-output setting is learned using boosting method. We formulate a multiple-output regression problem in such a way that overfitting is decreased and an analytic solution is admitted. Because we represent the image via a set of highly redundant Haar-like features that can be evaluated very quickly and select relevant features through boosting to absorb the knowledge of the training data, during testing we require no storage of the training data and evaluate the regression function almost in no time. We also propose an efficient training algorithm that breaks the computational bottleneck in the greedy feature selection process. We validate the efficiency of the proposed regressor using three challenging tasks of age estimation, tumor detection, and endocardial wall localization and achieve the best performance with a dramatic speed, e.g., more than 1000 times faster than conventional data-driven techniques such as support vector regressor in the experiment of endocardial wall localization.

1. Introduction

Consider the following vision tasks shown in Figure 1: (A) Given an image of a human face, can the computer tell his/her age? (B) Given a computer tomography (CT) image containing a pulmonary tumor, can the computer detect the tumor's position and anisotropic spread? (C) Given an ultrasound image of human heart or echocardiogram, can the computer automatically delineate the endocardial wall of the left ventricle? The above seemingly heterogeneous tasks can be solved using a general technique of image based regression (IBR).

The problem of IBR is defined as follows: Given an image $\mathbf{x}$, we are interested in inferring an entity $\mathbf{y}(\mathbf{x})$ that is associated with the image $\mathbf{x}$. The meaning of $\mathbf{y}(\mathbf{x})$ varies a lot in different applications. For example, it could be a feature characterizing the image (e.g., the human age in the problem A), a parameter related to the image (e.g., the position and anisotropic spread of the tumor in the problem B), or other meaningful quantity (e.g., the location of the endocardial wall in the problem C).

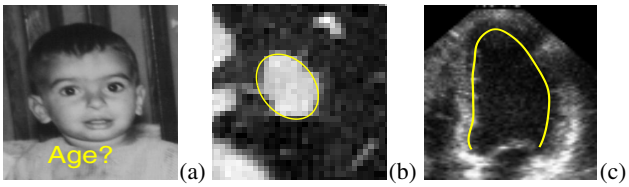

Figure 1: *Three image based regression tasks: (a) Age estimation; (b) Tumor detection; and (c) Endocardial wall delineation.*

IBR is an emerging challenge in the vision literature. In the article of Wang *et al.* [15], support vector regression was employed to infer a shape deformation parameter. In a recent work [2], Agarwal and Triggs used relevance vector regression to estimate a 3D human pose from silhouettes. However, in the above two works, the inputs to the regressors are not images themselves, rather pre-processed entities, e.g., landmark locations in [15] and shape context descriptor in [2].

Numerous algorithms [8] were proposed in the machine learning literature to attack the regression problem in general. Data-driven approaches gained prevalence. Popular data-driven regression approaches include nonparameteric kernel regression (NPR), linear methods and their nonlinear kernel variants such as kernel ridge regression (KRR), support vector regression (SVR), etc. We will briefly review them in section 2. However, it is often difficult or inefficient to directly apply them to vision applications due to the following challenges.

Curse of dimensionality. The input (i.e. image data) is of very high dimensional, which manifests the phenomenon commonly referred to as curse of dimensionality. Ideally, in order to well represent the sample space, the number of required image samples should be exponential to the cardinality of the input space. However, in practice, the number of training samples is often extremely sparse, compared with the cardinality of the input space. Overfitting is likely to happen without a careful treatment.

Varying appearance. First, there are a lot of factors that affect the appearance of the foreground object of interest. Apart from the intrinsic differences among the objects, ex-

trinsic factors include camera system, imaging geometry, lighting conditions, makeup, etc. Second, the variation arises from the presence of background whose appearance varies too. The third variation is caused by alignment. The regression technique must either tolerate the alignment error (as in the problem A) or regress out the alignment parameter (as in the problems B and C).

Multiple output. The output variable is also of high dimensional. Most regression approaches, such as SVR, can deal with the single-output regression problem very robustly. Extending them to the multiple-output setting is nontrivial as in the case of SVR. A naive practice of decoupling a multiple-output problem to several isolated single-output tasks ignores the statistical dependence among different dimensions of the output variable.

Storage and computation. The regression techniques such as NPR, KRR, and SVR are data-driven. There are two main disadvantages of the data-driven approaches: storage and computation. First, the techniques require storing a large amount of training data. In NPR and KRR, all training data are stored. In SVR, support vectors are stored[1]. Because the training data are images with high dimension, storing the training images can take a lot of memory space. Second, evaluating the data-driven regression function is slow because comparing the input image with the stored training images is time-consuming.

To overcome the above challenges, we propose an IBR algorithm using boosting method [4, 5, 7, 12]. AdaBoosting is the state-of-the-art classification method. After its theoretic connection to forward stagewise additive modeling [7] was discovered, Friedman [6] used boosting as a greedy function approximation in a regression setting [6]. Multiple additive regression tree (MART) [8] was proposed as a boosting tree solution to a single-output regression problem. Duffy and Helmbold [4] also studied boosting methods for regression for a single-output setting. However, a multiple-output regression setting is rarely studied in the literature. In this paper, we focus on this setting that takes images as inputs. We make the following contributions.

$\diamond$ We formulate the multiple-output regression problem in such a way that an analytic solution is allowed at each round of boosting. No decoupling of the output dimension is performed. Also, we decrease overfitting using an image-based regularization term that has an interpretation as prior knowledge and also allows an analytic solution.

$\diamond$ We invoke the boosting framework to perform feature selection such that only relevant local features are preserved to conquer the variations in appearance. The use of decision stumps as weak learners also makes it robust to appearance change.

$\diamond$ We use the Haar-like simple features [14] that can be rapidly computed. We do not store the training data. The knowledge of the training data is absorbed in the weighting coefficients and the selected feature set. Hence, we are able to evaluate the regression function almost in no time during testing.

$\diamond$ We propose an efficient implementation to perform boosting training, which is usually a time-consuming process if a truly greedy feature selection procedure is used. In our implementation, we select the features incrementally over the dimension of the output variable.

[1] In our experiments, we found that a large number of support vectors, often 80%-100% of the training data, are kept.

2. Review of regression techniques

In this section, we first recapitulate the regression problem and then briefly review three data-driven regression methods, namely NPR, KRR, and SVR. Details are covered in [8]. Below, we assume that $\mathtt{x} \in \mathcal{R}^d$ and $\mathtt{y}(\mathtt{x}) \in \mathcal{R}^q$.

2.1. Regression

Regression finds the solution to the following minimizing problem:

$$\hat{\mathtt{g}}(\mathtt{x}) = \arg\min_{\mathtt{g}\in\mathcal{G}} \mathcal{E}_{p(\mathtt{x},\mathtt{y})}\{L(\mathtt{y}(\mathtt{x}), \mathtt{g}(\mathtt{x}))\}, \qquad (1)$$

where $\mathcal{G}$ is the set of allowed output functions, $\mathcal{E}_{p(\mathtt{x},\mathtt{y})}$ takes the expectation under the generating distribution $p(\mathtt{x}, \mathtt{y})$, and the $L(\circ, \circ)$ function is the loss function that penalizes the deviation of the regressor output $\mathtt{g}(\mathtt{x})$ from the true output $\mathtt{y}(\mathtt{x})$.

In practice, it is impossible to compute the expectation since the distribution $p(\mathtt{x}, \mathtt{y})$ is unknown. Given a set of training examples $\{(\mathtt{x}_n, \mathtt{y}(\mathtt{x}_n))\}_{n=1}^N$, the cost function $\mathcal{E}_{p(\mathtt{x},\mathtt{y})} L(\mathtt{y}(\mathtt{x}), \mathtt{g}(\mathtt{x}))$ is approximated as the training error $J(\mathtt{g}) = \sum_{n=1}^N L(\mathtt{y}(\mathtt{x}_n), \mathtt{g}(\mathtt{x}_n))/N$.

If the number of samples N is infinitely large, the above approximation is exact by the law of the large number. Unfortunately, a practical value of N is never large enough, especially when dealing with image data and high-dimensional output parameter. A more severe problem is *overfitting*: given a limited number of training examples, it is easy to construct a function $\mathtt{g}(\mathtt{x})$ that yields a zero training error. To combat the overfitting, additional regularization constraints are often used, which results in a combined cost function (ignoring the scaling factor N^{-1})

$$J(\mathtt{g}) = \sum_{n=1}^{N} L(\mathtt{y}(\mathtt{x}_n), \mathtt{g}(\mathtt{x}_n)) + \lambda R(\mathtt{g}),$$

where $\lambda > 0$ is the *regularization coefficient* that controls the degree of regularization and $R(\mathtt{g})$ is the regularization term. Regularization often imposes certain smoothness on the output function or reflects some prior belief about the output.

2.2. Nonparametric kernel regression (NPR)

Nonparametric kernel regression (NPR) [8] is a smoothed version of k-nearest-neighbor (kNN) regression. The kNN regressor approximates the conditional mean, an optimal estimate in the L^2 sense. NPR takes the following form:

$$\mathtt{g}(\mathtt{x}) = \frac{\sum_{n=1}^{N} h_\sigma(\mathtt{x};\mathtt{x}_n)\mathtt{y}(\mathtt{x}_n)}{\sum_{n=1}^{N} h_\sigma(\mathtt{x};\mathtt{x}_n)},$$

where $h_\sigma(\circ;\mathtt{x}_n)$ is a kernel function. The most widely used kernel function is the RBF kernel

$$h_\sigma(\mathtt{x};\mathtt{x}_n) = rbf_\sigma(\mathtt{x};\mathtt{x}_n) = exp(-\frac{\|\mathtt{x}-\mathtt{x}_n\|^2}{2\sigma^2}).$$

The RBF kernel has a noncompact support. Other kernel functions with compact supports such as the Epanechnikov kernel can be used too.

In general, when confronted with the scenario of image based regression, NPR, albeit smooth, tends to overfit the data, i.e., yielding a low bias and a high variance.

2.3. Kernel ridge regression (KRR)

Kernel ridge regression (KRR) [8] assumes that the multiple-output regression function takes a linear form:

$$\mathtt{g}(\mathtt{x}) = \sum_{n=1}^{N} \alpha_n k(\mathtt{x};\mathtt{x}_n),$$

where $k(\mathtt{x};\mathtt{x}_n)$ is a reproducing kernel function and α_n is a $q \times 1$ vector that weights the kernel function. The choices for the reproducing kernel [13] include the RBF kernel, the polynomial kernel and so on. The solution to the multiple-output KRR derived from the training data is

$$\mathtt{g}(\mathtt{x}) = \mathtt{Y}(\mathtt{K}+\lambda\mathtt{I})^{-1}\kappa(\mathtt{x}),$$

where $\mathtt{Y}_{q\times N} = [\mathtt{y}(\mathtt{x}_1), \mathtt{y}(\mathtt{x}_2), ..., \mathtt{y}(\mathtt{x}_N)]$ is the training output matrix, $\mathtt{K}_{N\times N} = [k(\mathtt{x}_i;\mathtt{x}_j)]$ is the Gram matrix for the training data, and $\kappa(\mathtt{x})_{N\times 1} = [k(\mathtt{x};\mathtt{x}_1), k(\mathtt{x};\mathtt{x}_2), ..., k(\mathtt{x};\mathtt{x}_N)]^\mathsf{T}$.

In general, when a linear kernel is used, KRR tends to underfit the data, i.e., yielding a high bias and a low variance, because it uses a simple linear form. Using the nonlinear kernel function often gives enhanced performance. One computational difficulty of KRR lies in inverting the $N \times N$ matrix $\mathtt{K}+\lambda\mathtt{I}$.

2.4. Support vector regression (SVR)

Support vector regression (SVR) [13] is a robust regression method. Its current formulation works for *single output* data, i.e. $q = 1$. SVR minimizes the following cost function

$$\frac{1}{2}\|\mathtt{w}\|^2 + C\sum_{n=1}^{N} |y(\mathtt{x}_n) - g(\mathtt{x}_n)|_\epsilon,$$

where $|\circ|_\epsilon$ is an ϵ-insensitive function, $g(\mathtt{x}) = \sum_{n=1}^{N} w_n k(\mathtt{x};\mathtt{x}_n)$ with $k(\mathtt{x};\mathtt{x}_n)$ being a reproducing kernel function and w_n being its weight, and $\mathtt{w} = [w_1, w_2, \ldots, w_n]^\mathsf{T}$. Because some of the coefficients w_n, which can be found through a quadratic programming procedure, are zero-valued, the samples $\mathtt{x}_n$ associated with nonzero weights are called support vectors.

SVR strikes a good balance between bias and variance tradeoff and hence very robust. Unfortunately, directly applying SVR to the multiple-output regression problem is difficult.

3. Regression using boosting method

We now define the loss function and the regularization term that are appropriate for our purpose of developing regression algorithm using boosting method.

We focus on the L^2 loss function. To allow a general treatment and to deal with the scaling effort of different data dimensions, we use the normalized error cost:

$$\begin{aligned} L(\mathtt{y}(\mathtt{x}),\mathtt{g}(\mathtt{x})) &= [\mathtt{y}(\mathtt{x})-\mathtt{g}(\mathtt{x})]^\mathsf{T}\mathtt{A}[\mathtt{y}(\mathtt{x})-\mathtt{g}(\mathtt{x})] \\ &= \|\mathtt{y}(\mathtt{x})-\mathtt{g}(\mathtt{x})\|^2_\mathtt{A}, \end{aligned}$$

where $\mathtt{A}_{q\times q}$ is a *normalization matrix* that must be positive definite.

Regularization exists in various forms. In this paper, we focus on the following data-driven regularization term $\|\mu - \mathtt{g}(\mathtt{x})\|^2_\mathtt{B}$, where $\mathtt{B}_{q\times q}$ is a *normalization matrix* that must be positive definite. This regularization term has a subspace interpretation with μ being the mean and $\mathtt{B}^{-1}$ being the covariance matrix.

Hence, it boils down to the following cost function to be minimized.

$$\begin{aligned} J(\mathtt{g}) &= \sum_{n=1}^{N} \|\mathtt{y}(\mathtt{x}_n)-\mathtt{g}(\mathtt{x}_n)\|^2_\mathtt{A} + \lambda\sum_{n=1}^{N} \|\mu-\mathtt{g}(\mathtt{x}_n)\|^2_\mathtt{B} \\ &= \sum_{n=1}^{N} \|\mathtt{r}(\mathtt{x}_n)\|^2_\mathtt{A} + \lambda\sum_{n=1}^{N} \|\mathtt{s}(\mathtt{x}_n)\|^2_\mathtt{B} \\ &= tr\{\mathtt{ARR}^\mathsf{T}\} + \lambda tr\{\mathtt{BSS}^\mathsf{T}\} \\ &= \|\mathtt{R}\|^2_\mathtt{A} + \lambda\|\mathtt{S}\|^2_\mathtt{B}, \end{aligned}$$

where $\mathtt{r}(\mathtt{x}) = \mathtt{y}(\mathtt{x}) - \mathtt{g}(\mathtt{x})$ is the *approximation error*, $\mathtt{s}(\mathtt{x}) = u - \mathtt{g}(\mathtt{x})$ is the *deviation error*, and the matrices $\mathtt{R}_{q\times N}$ and $\mathtt{S}_{q\times N}$ are, respectively, defined as follows

$$\mathtt{R} = [\mathtt{r}(\mathtt{x}_1), \mathtt{r}(\mathtt{x}_2), \ldots, \mathtt{r}(\mathtt{x}_N)], \mathtt{S} = [\mathtt{s}(\mathtt{x}_1), \mathtt{s}(\mathtt{x}_2), \ldots, \mathtt{s}(\mathtt{x}_N)].$$

We now resort to the influential framework of boosting to derive an *analytic* solution.

3.1. Boosting

In boosting method for regression, the regression output function $\mathtt{g}(\mathtt{x}) : \mathcal{R}^d \rightarrow \mathcal{R}^q$ is assumed to take a linear form:

$$\mathtt{g}(\mathtt{x}) = \sum_{t=1}^{T} \alpha_t \mathtt{h}_t(\mathtt{x});\ \mathtt{h}_t(\mathtt{x}) \in \mathcal{H},$$

where each $\mathtt{h}_t(\mathtt{x})$ is a weak learner (or weaker function) and $\mathtt{g}(\mathtt{x})$ is a strong learner (or strong function). Further, it is assumed that a weak function $\mathtt{h}(\mathtt{x}) : \mathcal{R}^d \rightarrow \mathcal{R}^q$ lies in a *dictionary* set or weak function set $\mathcal{H}$.

Boosting iteratively approximates the target function $\mathtt{y}(\mathtt{x})$ by adding one more weak function

$$\mathtt{g}'(\mathtt{x}) = \mathtt{g}(\mathtt{x}) + \alpha \mathtt{h}(\mathtt{x}). \tag{2}$$

At each round of boosting, we select the function $\hat{\mathtt{h}}$ and its weight coefficient $\hat{\alpha}$ that mostly decreases the cost function. In other words, the following problem is attacked.

$$(\hat{\alpha}, \hat{\mathtt{h}}) = \arg \min_{\alpha, \mathtt{h} \in \mathcal{H}} J(\mathtt{g} + \alpha \mathtt{h}). \tag{3}$$

Main theorem: By adding a function $\alpha\mathtt{h}(\mathtt{x})$ to the output function $\mathtt{g}(\mathtt{x})$ as in (2), the new cost function $J(\mathtt{g}')$ maximally decreases the cost function $J(\mathtt{g})$ by a factor of $(1 - \epsilon^2(\mathtt{h}))$, with $|\epsilon(\mathtt{h})| \leq 1$.

Proof: As shown in Appendix, the cost function $J(\mathtt{g}')$ is computed as

$$\begin{aligned} J(\mathtt{g}') &= \sum_{n=1}^{N} \|\mathtt{y}(\mathtt{x}_n) - \mathtt{g}'(\mathtt{x}_n)\|_{\mathtt{A}}^2 + \lambda \sum_{i=n}^{N} \|\mu - \mathtt{g}'(x_n)\|_{\mathtt{B}}^2 \\ &= J(\mathtt{g}) - 2\alpha tr\{(\mathtt{AR} + \lambda \mathtt{BS})\mathtt{H}^{\mathsf{T}}\} + \alpha^2 \|\mathtt{H}\|_{\mathtt{A}+\lambda\mathtt{B}}^2 \\ &= J(\mathtt{g}) - 2\alpha tr\{\mathtt{DH}^{\mathsf{T}}\} + \alpha^2 \|\mathtt{H}\|_{\mathtt{C}}^2, \end{aligned}$$

where $\mathtt{C} = \mathtt{A} + \lambda\mathtt{B}$, $\mathtt{D} = \mathtt{AR} + \lambda\mathtt{BS}$, and $\mathtt{H}_{q \times N} = [\mathtt{h}(\mathtt{x}_1), \mathtt{h}(\mathtt{x}_2), \ldots, \mathtt{h}(\mathtt{x}_N)]$.

With the function $\mathtt{h}$ fixed, the cost function $J(\mathtt{g}')$ is quadratic in α so that there is a unique minimizer $\hat{\alpha}(\mathtt{h})$. By letting $\frac{\partial J(\mathtt{g}')}{\partial \alpha} = 0$, simple algebra yields that

$$\hat{\alpha}(\mathtt{h}) = \frac{tr\{\mathtt{DH}^{\mathsf{T}}\}}{\|\mathtt{H}\|_{\mathtt{C}}^2} = \frac{tr\{(\mathtt{AR} + \lambda\mathtt{BS})\mathtt{H}^{\mathsf{T}}\}}{\|\mathtt{H}\|_{\mathtt{A}+\lambda\mathtt{B}}^2}.$$

The minimum cost $J(\mathtt{g}')$ is then calculated as

$$J(\mathtt{g}') = J(\mathtt{g}) - \frac{tr^2\{\mathtt{DH}^{\mathsf{T}}\}}{\|\mathtt{H}\|_{\mathtt{C}}^2} = J(\mathtt{g})(1 - \epsilon^2(\mathtt{h})),$$

where

$$\epsilon(\mathtt{h}) = \frac{\hat{\alpha}(\mathtt{h})\sqrt{\|\mathtt{H}\|_{\mathtt{C}}^2}}{\sqrt{\|\mathtt{R}\|_{\mathtt{A}}^2 + \lambda\|\mathtt{S}\|_{\mathtt{B}}^2}} = \frac{tr\{\mathtt{DH}^{\mathsf{T}}\}}{\sqrt{\|\mathtt{H}\|_{\mathtt{C}}^2}\sqrt{\|\mathtt{R}\|_{\mathtt{A}}^2 + \lambda\|\mathtt{S}\|_{\mathtt{B}}^2}}$$

It is obvious that $|\epsilon(\mathtt{h})| \leq 1$ since the cost functions $J(\mathtt{g}')$ and $J(\mathtt{g})$ is nonnegative. $< E.O.F. >$

In practice, we can always assume $\epsilon(\mathtt{h}) \geq 0$ because, if $\epsilon(h) < 0$, we simply change the sign of the function h. Therefore, in the sequel, the absolute symbol $|.|$ is removed. Correspondingly, we have $\hat{\alpha}(\mathtt{h}) \geq 0$ because $\hat{\alpha}(\mathtt{h})$ and $\epsilon(\mathtt{h})$ have the same sign.

Therefore, each boosting round aims at finding the function $\mathtt{h}$ such that the cost function is maximally reduced. Equivalently, the value of $\epsilon(\mathtt{h})$ is maximized.

$$\hat{\mathtt{h}} = \arg \max_{\mathtt{h} \in \mathcal{H}} \epsilon(\mathtt{h}) = \arg \max_{\mathtt{h} \in \mathcal{H}} \frac{tr\{\mathtt{DH}^{\mathsf{T}}\}}{\sqrt{\|\mathtt{H}\|_{\mathtt{C}}^2}}. \tag{4}$$

Note that the term $\sqrt{\|\mathtt{R}\|_{\mathtt{A}}^2 + \lambda\|\mathtt{S}\|_{\mathtt{B}}^2}$ does not depend on $\mathtt{H}$ and hence can be ignored in the above. The corresponding value of α is $\hat{\alpha}(\hat{\mathtt{h}})$. Finally, the cost $J(\mathtt{g}')$ maximally decreases the cost $J(\mathtt{g})$ by a factor of $(1 - \epsilon(\hat{\mathtt{h}})^2)$.

3.2. Shrinkage

Shrinkage [3, 6] is another measure for reducing the effect of overfitting. The idea is very simple: at each round of boosting, simply shrink the newly selected function $\alpha\mathtt{h}(\mathtt{x})$ by a *shrinkage factor* $\eta \in [0, 1]$. The new updating rule is

$$\mathtt{g}'(\mathtt{x}) = \mathtt{g}(\mathtt{x}) + \eta\hat{\alpha}\hat{\mathtt{h}}(\mathtt{x}),$$

where $\hat{\alpha}$ and $\hat{\mathtt{h}}$ are the optimal solutions found above. In practice, we found that a modest choice of $\eta = 0.5$ gives good results.

Figure 2 summarizes the regression algorithm using boosting method. Appendix provides a glossary of notations used in the paper.

4. Image based regression

The image-related entity is the dictionary set $\mathcal{H}$, whose every element is based on the image $\mathtt{x}$. Intuitively, this function set must be sufficiently large such that it allows rendering, through a linear combination, highly complex output function $\mathtt{y}(\mathtt{x})$. Following the spirit of the Viola and Jones [14], we use one-dimensional decision stumpsas primitives to construct the weak function set $\mathcal{H}$. The advantages of using decision stumps include (i) that they are robust to appearance variation; (ii) that they are local features; (iii) that they are fast to evaluate using the so-called integral image [14]; and, most importantly, (iv) that they allows an incremental feature selection scheme that will be addressed later.

4.1. Weak function set

A one-dimensional (1D) decision stump $h(\mathtt{x})$ is associated with a Haar filter feature $f(\mathtt{x})$, a decision threshold θ, and

1. Initialization $t = 0$.
   (a) Set the fixed parameter values: μ (the mean vector), $\mathtt{A}$ and $\mathtt{B}$ (the normalization matrices), λ (the regularization coefficient), and η (the shrinkage factor).
   (b) Set the values related to the stopping criteria: T_{max} (the maximum number of iterations), J_{min} (the minimum cost function), ϵ_{min}, and α_{min}.
   (c) Set initial values for $t = 0$: $\mathtt{g}_0(\mathtt{x}) = 0$, $\mathtt{r}_0(x) = \mathtt{y}(\mathtt{x})$, and $\mathtt{s}_0(\mathtt{x}) = \mu$.
2. Iteration $t = 1, \ldots, T_{max}$
   (a) Find $\hat{\mathtt{h}}_t = \arg \max_{\mathtt{h}\in\mathcal{H}} \epsilon_t(\mathtt{h})$ and its corresponding $\hat{\alpha}_t(\hat{\mathtt{h}}_t)$ and $\epsilon_t(\hat{\mathtt{h}}_t)$.
   (b) Form the new function $\mathtt{g}_t(\mathtt{x}) = \mathtt{g}_{t-1}(\mathtt{x}) + \eta\hat{\alpha}_t\hat{\mathtt{h}}_t(\mathtt{x})$.
   (c) Evaluate the approximation error $\mathtt{r}_t(\mathtt{x}) = \mathtt{y}(\mathtt{x}) - \mathtt{g}_t(\mathtt{x})$, the deviation error $\mathtt{s}_t(\mathtt{x}) = \mu - \mathtt{g}_t(\mathtt{x})$, and the cost function $J(\mathtt{g}_t)$.
   (d) Check convergence, e.g. see if $J(\mathtt{g}_t) < J_{min}$, $\alpha_t < \alpha_{min}$, $\epsilon_t < \epsilon_{min}$, or combination of them.

Figure 2: *Regression algorithm using boosting method.*

a parity direction indicator p that takes a binary value of either $+1$ or -1.

$$h(\mathtt{x}) = \begin{cases} +1 & \text{if } pf(\mathtt{x}) \geq p\theta \\ -1 & \text{otherwise} \end{cases} \tag{5}$$

Each Haar filter $f(\mathtt{x})$ has its own attributes: type, window position, and window size. Given a moderate size of image, one can generate a huge number of Haar filters by varying the filter attributes. See [14] for details. Denote the number of Haar filters by M. By adjusting the threshold θ (say K even-spaced levels), for every Haar filter, one can further create K decision stumps. In total, we have $2KM$ 1-D decision stumps. Note that the number $2KM$ can be prohibitively large so that it can even create difficulty in storing all these decision stumps during training.

A weak function in the present paper is constructed as a q-dimensional (q-D) decision stump $\mathtt{h}(\mathtt{x})$ that simply stacks q 1D decision stumps.

$$\mathtt{h}(\mathtt{x})_{q\times 1} = [h_1(\mathtt{x}), h_2(\mathtt{x}), ..., h_q(\mathtt{x})]^{\mathsf{T}}.$$

Note that each $h_j(\mathtt{x})$ in the above may be associated with a different parameter. Hence, one can construct a sufficiently large weak function set that contains $(2KM)^q$ weak functions!

4.2. Feature selection

Boosting operates as a feature selection oracle. At each round of boosting, the features that can maximally decrease the cost function are selected. However, to transform the boosting recipe in Figure 2 into an efficient algorithm, there is a computational bottleneck, that is Step $(2a)$. This step necessitates a *greedy* feature selection scheme that is too expensive to evaluate because, in principle, it involves evaluating $(2MNK)^q$ decision stumps, a formidable computational task in practice.

One possible way is to break the q-D regression problem into q *independent* 1D regression problems, leading to an *independent* feature selection scheme. Consequently, only $2qMNK$ decision stumps are evaluated at each round of boosting. However, the independence assumption is too strong to be hold in real situations.

We propose an *incremental* feature selection scheme by breaking the q-D regression problem into q *dependent* 1D regression problems. Using the incremental vector

$$\mathtt{h}^i(\mathtt{x})_{i\times 1} = [h_1(\mathtt{x}), h_2(\mathtt{x}), ..., h_i(\mathtt{x})]^{\mathsf{T}} = [\mathtt{h}^{i-1}(\mathtt{x})^{\mathsf{T}}, h_i(\mathtt{x})]^{\mathsf{T}},$$

and the incremental matrices $\mathtt{C}^i$, $\mathtt{D}^i$, and $\mathtt{H}^i$,

$$\mathtt{C}^i = \begin{bmatrix} \mathtt{C}^{i-1} & \mathtt{c}^{i-1} \\ \mathtt{c}^{i-1\mathsf{T}} & c_i \end{bmatrix},\ \mathtt{D}^i = \begin{bmatrix} \mathtt{D}^{i-1} \\ \mathtt{d}_i^{\mathsf{T}} \end{bmatrix},\ \mathtt{H}^i = \begin{bmatrix} \mathtt{H}^{i-1} \\ \mathtt{h}_i^{\mathsf{T}} \end{bmatrix}$$

we define the incremental coefficient as

$$\epsilon^i(h) = tr\{\mathtt{D}^i\mathtt{H}^{i\mathsf{T}}\}/\sqrt{\|\mathtt{H}^i\|^2_{\mathtt{C}^i}}. \tag{6}$$

Therefore, we learn a 1D decision stump $h_i(\mathtt{x})$ at one time.

$$\hat{h}_i = \arg\max_{h\in\mathcal{H}} \epsilon^i(h).$$

In terms of computation, the incremental selection scheme requires evaluating $2qMNK$ decision stumps, the same as the independent selection scheme. Of course, compared with the independent scheme, there are overhead computations needed in the incremental scheme because we calculate matrix quantities like $tr\{\mathtt{D}^i\mathtt{H}^{i\mathsf{T}}\}$ and $\|\mathtt{H}^i\|^2_{\mathtt{C}^i}$; whereas in the independent feature selection scheme, the counterparts are vector inner products. Fortunately, there exist reusable computations. For example, it is easy to show that

$$\|\mathtt{H}^i\|^2_{\mathtt{C}^i} = \|\mathtt{H}^{i-1}\|^2_{\mathtt{C}^{i-1}} + 2\mathtt{h}_i^{\mathsf{T}}\mathtt{H}^{i-1\mathsf{T}}\mathtt{c}^{i-1} + c_i\mathtt{h}_i^{\mathsf{T}}\mathtt{h}_i.$$

$$tr\{\mathtt{D}^i\mathtt{H}^{i\mathsf{T}}\} = tr\{\mathtt{D}^{i-1}\mathtt{H}^{i-1\mathsf{T}}\} + \mathtt{d}_i^{\mathsf{T}}\mathtt{h}_i. \tag{7}$$

Although the incremental selection scheme in principle yields a suboptimal solution, it is better than the independence selection scheme because it utilizes the dependence among the output data dimensions to some extent. In fact, there exist special cases when the incremental feature selection yields the same solution as the greedy selection scheme. One such case is that when $\mathtt{C} = \beta\mathtt{I}$ (e.g. when

$\mathtt{A} = \mathtt{B} = \mathtt{I}$). This makes the denominator term $\|\mathtt{H}^i\|^2_{\mathtt{C}^i}$ in (6) a constant value of $i\beta^2$, which does not vary with different choices of h functions. Therefore, $\epsilon^i(h)$ only depends on $tr\{\mathtt{D}^i\mathtt{H}^{i\mathsf{T}}\}$. But, according to (7), maximizing $tr\{\mathtt{D}^i\mathtt{H}^{i\mathsf{T}}\}$ can be done by maximizing the term $\mathtt{d}_i^{\mathsf{T}}\mathtt{h}_i$ in each incremental step.

To improve robustness and remove bias, we randomly permutate the order of the dimensions of the output variable. Other tricks to improve computational efficiency include: (i) randomly sampling the dictionary set, i.e. replacing M by a smaller M'; and (ii) randomly sampling the training data set, i.e., replacing N by a smaller N'.

Figure 3 presents the incremental feature selection scheme.

1. Initialization.
   - Create a random permutation of $\{1, 2, \ldots, q\}$, yielding $\{< 1 >, < 2 >, \ldots, < q >\}$.
2. Iteration over the dimension of the output variable $i = 1, 2, \ldots, q$
   - (optional) Sample M' Haar filters from the dictionary set and form the reduced set of weak functions $\mathcal{H}'$.
   - (optional) Sample N' data points from the training set.
   - Loop over the filter index $m = 1, 2, \ldots, M'$ and the threshold level index $k = 1, 2, \ldots, K$ to find $h_{<i>} = \arg\ \max_{h \in \mathcal{H}'} \epsilon^{<i>}(h)$.
   - Form the new vector $\mathtt{h}^{<i>} = [\mathtt{h}^{<i-1>\mathsf{T}}, h_{<i>}]^{\mathsf{T}}$.
   - Compute reusable quantities $tr\{\mathtt{D}^{<i>}\mathtt{H}^{<i>\mathsf{T}}\}$ and $tr\{\|\mathtt{H}^{<i>}\|^2_{\mathtt{C}^{<i>}}\}$.

Figure 3: *Incremental feature selection.*

5. Experiments

We tested the proposed IBR algorithm on the three problems mentioned in beginning of the paper. For comparison, we also implemented NPR, KRR and SVR, all using the RBF kernel function. We used 5-fold cross-validation as the evaluation protocol and tuned the RBF kernel width for empirical best performance. Because SVR only works for the single-output regression problem, we decoupled the multiple-output regression problem to isolated single-output ones. For IBR, we simply set $\mathtt{A} = \mathtt{B} = \mathtt{I}$ and stopped learning after a maximum number of boosting rounds is reached.

If the output is multidimensional, we applied a whitening filter to decorrelate the output variable: $\mathtt{y} = \mathtt{D}^{-1/2}\mathtt{V}^{\mathsf{T}}\{\mathtt{y} - \mu\}$, where μ is the mean and $\mathtt{D}$ and $\mathtt{V}$ are eigenvalue and eigenvector matrices of the covariance matrix.

Table 1 shows the error statistics and computational time for evaluating regression outputs of all testing images belonging to the 5 testing subsets used in the 5-fold cross validation. We used different error measurement that is meaningful to the data of interest. We collected the error statistics for all testing images and reported their mean, 25% percentile, median, and 75% percentile. We used C++ programs on a PC with 2.4GHz dual CPUs and 3GB memory to record the computational time.

5.1. Age estimation

Aging modeling [9] is important for face analysis and recognition. In this experiment, we focused on estimating the human age.

Data statistics: We used the FGnet aging database [1]. There are 1002 facial images in the database. Five random divisions with 800 for training and 202 for testing are formed. The age ranges from 0 to 69. Normalization was done by first aligning 68 landmark points provided by [1] and then performing a zero-mean-unit-variance operation. However, we kept enough background pixels.

Input/output: The input $\mathtt{x}$ is a 60×60 image; the output $\mathtt{y}$ is his/her normalized age. We converted the actual age to $\mathtt{y} = log(\mathtt{y} + 1)$ to avoid negative regressor output.

Variation: The face images involve all possible variations including illumination, pose, expression, beards, moustaches, spectacles, etc. Figure 4 shows sample images of one person at different ages and with various appearance variations.

Performance: We computed the absolute age difference as the error measurement. The proposed IBR approach (with 500 weak functions, the regularization coefficient $\lambda = 0.1$ and the shrinkage factor $\eta = 0.5$) achieves the best performance and runs fastest. In [9], age estimation is performed on a smaller set with mostly frontal-view images. The reported mean absolute error in years is 7.48 using a pure appearance based regressor.

Figure 4: *Sample images (before and after normalization) of one person at different ages.*

5.2. Pulmonary tumor detection

In the second experiment, we applied the IBR algorithm to detect a pulmonary tumor in a CT image [10]. To be more specific, given an input CT image, we regressed out the center position (t, s) and anisotropic spread of the tumor. The

Table 1: *Comparison of different regressors for (a) age estimation, (b) tumor detection, and (c) endocardial wall detection.*

	NPR	KRR	SVR	IBR	
mean err.	8.44	13.56	6.60	5.81	
25% per. err.	2.54	3.99	1.38	1.26	(a)
median err.	5.50	10.80	4.39	3.15	
75% per. err.	10.87	17.99	9.04	7.79	
testing time(s)	3.6s	3.6s	3.3s	0.016s	

	NPR	KRR	SVR	IBR	
mean err.	0.544	0.593	0.557	0.533	
25% per. err.	0.420	0.458	0.335	0.332	(b)
median err.	0.543	0.609	0.509	0.496	
75% per. err.	0.664	0.719	0.796	0.716	
testing time(s)	0.63s	0.55s	0.51s	0.02s	

	NPR	KRR	SVR	IBR	
mean err.	2.438	3.423	2.423	2.148	
25% per. err.	1.810	1.484	1.756	1.582	(c)
median err.	2.271	2.061	2.013	1.996	
75% per. err.	2.850	3.154	2.734	2.516	
testing time(s)	668s	534s	512s	0.45s	

2D anisotropic spread is described by a 2×2 positive definite matrix $[a_{11}, a_{12}; a_{12}, a_{22}]$, with $a_{11} > 0$ and $a_{22} > 0$.

Data statistics: There are 525 CT images in total. Five random divisions with 400 for training and 125 for testing are formed. The center position is mostly within 6 pixels of the image center, but the anisotropic spread is rather arbitrary in terms of scale and orientation.

Input/output: The input $\mathbf{x}$ is a 33×33 image; the output $\mathbf{y}$ is a 5-D variable (after whitening), i.e, $q = 5$. To avoid the negative output values of a_{11} and a_{22}, we again used $log(a_{11})$ and $log(a_{22})$. So the whitening filter is applied to $[t, s, log(a_{11}), a_{12}, log(a_{22})]^{\mathsf{T}}$.

Variation: Figure 5 shows some sample images that encompass typical appearance variations: cluttered background, imaging noise, arbitrary shape, fake signal, etc.

Performance: Since each output parameter defined an ellipse in the 2D image (See Figure 5 for illustration), we use an area non-overlapping ratio to measure performance. Given two ellipses A and B, we measure the area non-overlapping ratio as

$$ratio = 1 - area(A \bigcap B)/area(A \bigcup B).$$

The smaller the ratio is, the better the two ellipses overlap. The proposed IBR approach (with 500 weak functions, $\lambda = 0.2$, and $\eta = 0.5$) outperforms the other regressors in terms of performance and computation. Figure 5 also visualizes the regressed output, with the ground truth[2] superimposed. These images are selected from the testing subset.

[2]The ground truth is actually computed using the method in [10] and may not fit the data well. For example, in the bottom leftmost image, the ground truth is off; the regressed output gives a better result.

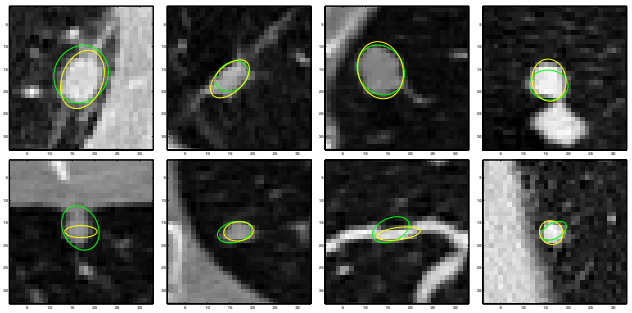

Figure 5: *Sample CT images with ground truth (yellow) and regression result (green).*

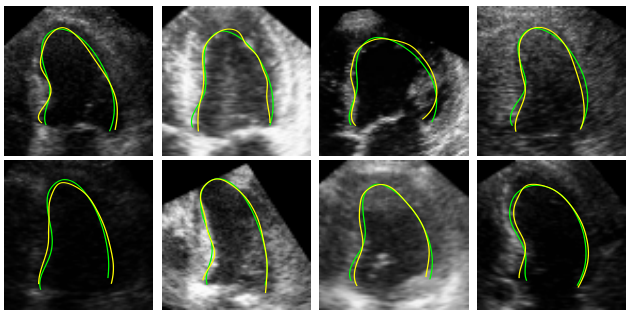

Figure 6: *Sample echocardiograhic images with ground truth (yellow) and regression result (green).*

5.3. Endocardial wall localization

Myocardial wall localization and tracking [16] is a challenging task in processing echocardiographic images [11] that is the ultrasound 2D image of the heart. In particular, accurate localization of the left ventricle is essential to clinical cardiac analysis. In this experiment, we focused on locating the endocardial wall of the left ventricle in the apical four chamber view.

Data statistics: There are 7943 images in total. Five random divisions with 6400 for training and 1543 for testing are formed.

Input/output: The input $\mathbf{x}$ is an 80×74 image; the output $\mathbf{y}$ is a 7-D variable, i.e., $q = 7$. The endocardial wall is a nonrigid open contour parameterized by 17 control points, i.e. with 34 variables. After whitening, we kept only the top 7 principal components.

Variation: Depending on the sonographer's imaging experience and the patient's anatomic structure and tissue characterization, the left ventricle appearance, which contains heart apex, septal wall, lateral wall, papillary muscle, annulus, etc., varies significantly across patients. Also signal dropout is often found in ultrasound imaging. Consequently, the endocardial border deforms a lot. Figure 6 shows sample images illustrating the appearance variations. All these images belongs to one testing subset.

Performance: We measured the average pixel error for the control points: $\sqrt{\|\mathtt{g}(\mathtt{x}) - \mathtt{y}(\mathtt{x})\|^2/34}$. The proposed IBR approach (with 1500 weak functions, $\lambda = 0.05$, and $\eta = 0.5$) surpasses the other regressors in terms of performance and computation. In particular, IBR runs more than 1000 times faster than the other regressors because its computational time does not scale up with the number of training samples. Figure 6 visualizes the ground truth contour and the regressed contour.

6. Conclusions

We presented a general IBR algorithm using boosting to select relevant features from the image. Since the regressor does not depend the training data, it is efficient in terms of storage and computation. An efficient training algorithm that performs incremental feature selection was also presented. Experimental results on real datasets demonstrated the advantage of the proposed IBR algorithm over others often used in the literature, such as NPR, KRR, and SVR.

References

[1] http://sting.cycollege.ac.cy/∼alanitis/fgnetaging/index.htm.

[2] A. Agarwal and B. Triggs. 3D human pose from silhouette by revelance vector regression. *CVPR*, 2004.

[3] H. Copas. Regression, prediction, and shrinkage. *J. R. Statist. Soc. B*, 45:311–354, 1983.

[4] N. Duffy and D. Helmbold. Boosting methods for regression. *Machine Learning*, 47:153–200, 2002.

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

[6] J. Friedman. Greedy function approxiamtion: A gradient boosting machine. *The Annals of Statistics*, 28(2), 2001.

[7] J. Friedman, T. Hastie, and R. Tibshirani. Additive logistive regression: A statistical view of boosting. *The Annals of Statistics*, 28(2):337–374, 2000.

[8] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer-Verlag, New York, 2001.

[9] A. Lanitis, C. Taylor, and T. Cootes. Toward automatic simulation of aging effects on face images. *PAMI*, 24(4):442–455, 2002.

[10] K. Okada, D. Comaniciu, and A. Krishanan. Scale selection for anisotropic scale-space: Application for volumetric tumor characterization. *CVPR*, 2004.

[11] C. Otto. *Textbook of Clinical Echocardiography*. W. B. Sauders, Philadephia, 2nd edition, 2000.

[12] R. Schapire, Y. Freund, P. Bartlett, and W. S. Lee. Boosting the margin: A new explanation for the effectiveness of voting methods. *The Annals of Statistics*, 26:1651–1686, 1998.

[13] V. N. Vapnik. *The Nature of Statistical Learning Theory*. Springer-Verlag, New York, ISBN 0-387-94559-8, 1995.

[14] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. *CVPR*, 2001.

d	input data dimension.
q	output data dimension.
$\mathtt{x}$	image. $\mathtt{x} \in \mathcal{R}^d$.
$\mathtt{y}(\mathtt{x})$	ground truth output. $\mathtt{y} \in \mathcal{R}^q$.
$\mathtt{g}(\mathtt{x})$	regression output. A strong function.
$h(\mathtt{x})$	decision stump.
$\mathtt{h}(\mathtt{x})$	weak function, stacking q decision stumps.
M	number of decision stumps.
N	number of training examples.
K	number of thresholds for one decision stump.
$\{(\mathtt{x}_n, \mathtt{y}(\mathtt{x}_n))\}_{n=1}^N$	training set.
$\mathtt{A}_{q\times q}$, $\mathtt{B}_{q\times q}$	normalization matrices.
λ	regularization coefficient.
η	shrinkage factor.
$\mathtt{C}_{q\times q}$	used-defined matrix $\mathtt{C} = \mathtt{A} + \lambda\mathtt{B}$.
$\mathtt{H}_{q\times N}$	weak function data matrix.
$\mathtt{R}_{q\times N}$	approximation error matrix.
$\mathtt{S}_{q\times N}$	deviation error matrix.
$\mathtt{D}_{q\times N}$	used-defined matrix $\mathtt{D} = \mathtt{AR} + \lambda\mathtt{BS}$.

Table 2: A glossary of notations.

[15] S. Wang, W. Zhu, and Z. Liang. Shape deformation: SVM regression and application to medical image segmentation. *ICCV*, 2001.

[16] X. Zhou, D. Comaniciu, R. Cruceanu, B. Xie, and A. Gupta. A unified framework for uncertainty propagation in automatic shape tracking. *CVPR*, 2004.

Appendix

Table 2 provides a glossary of notations used for the IBR algorithm. Below presents the detailed computation required for the main theorem.

$$
\begin{aligned}
& J(\mathtt{g}') \\
= & \sum_{n=1}^{N} \|\mathtt{y}(\mathtt{x}_n) - \mathtt{g}'(\mathtt{x}_n)\|_{\mathtt{A}}^2 + \lambda \sum_{n=1}^{N} \|\mu - \mathtt{g}'(\mathtt{x}_n)\|_{\mathtt{B}}^2 \\
= & \sum_{n=1}^{N} \|\mathtt{y}(\mathtt{x}_n) - \mathtt{g}(\mathtt{x}_n) - \alpha\mathtt{h}(\mathtt{x}_n)\|_{\mathtt{A}}^2 \\
& + \lambda \sum_{n=1}^{N} \|\mu - \mathtt{g}(\mathtt{x}_n) - \alpha\mathtt{h}(\mathtt{x}_n)\|_{\mathtt{B}}^2 \\
= & \sum_{n=1}^{N} \|\mathtt{r}(\mathtt{x}_n) - \alpha\mathtt{h}(\mathtt{x}_n)\|_{\mathtt{A}}^2 + \lambda \sum_{n=1}^{N} \|\mathtt{s}(\mathtt{x}_n) - \alpha\mathtt{h}(\mathtt{x}_n)\|_{\mathtt{B}}^2 \\
= & \; tr\{\mathtt{A}[\mathtt{R} - \alpha\mathtt{H}][\mathtt{R} - \alpha\mathtt{H}]^{\mathsf{T}}\} + \lambda tr\{\mathtt{B}[\mathtt{S} - \alpha\mathtt{H}][\mathtt{S} - \alpha\mathtt{H}]^{\mathsf{T}}\} \\
= & \; (tr\{\mathtt{ARR}^{\mathsf{T}}\} + \lambda tr\{\mathtt{BSS}^{\mathsf{T}}\}) \\
& - 2\alpha(tr\{\mathtt{ARH}^{\mathsf{T}}\} + \lambda tr\{\mathtt{BSH}^{\mathsf{T}}\}) \\
& + \alpha^2(tr\{\mathtt{AHH}^{\mathsf{T}}\} + \lambda tr\{\mathtt{BHH}^{\mathsf{T}}\}) \\
= & \; J(\mathtt{g}) - 2\alpha \, tr\{(\mathtt{AR} + \lambda\mathtt{BS})\mathtt{H}^{\mathsf{T}}\} + \alpha^2 \|\mathtt{H}\|_{\mathtt{A}+\lambda\mathtt{B}}^2 \\
= & \; J(\mathtt{g}) - 2\alpha \, tr\{\mathtt{DH}^{\mathsf{T}}\} + \alpha^2 \|\mathtt{H}\|_{\mathtt{C}}^2
\end{aligned}
$$