Probabilistic Regression Using Basis Function Models

Gregory Z. Grudic
Department of Computer Science
University of Colorado, Boulder
grudic@cs.colorado.edu

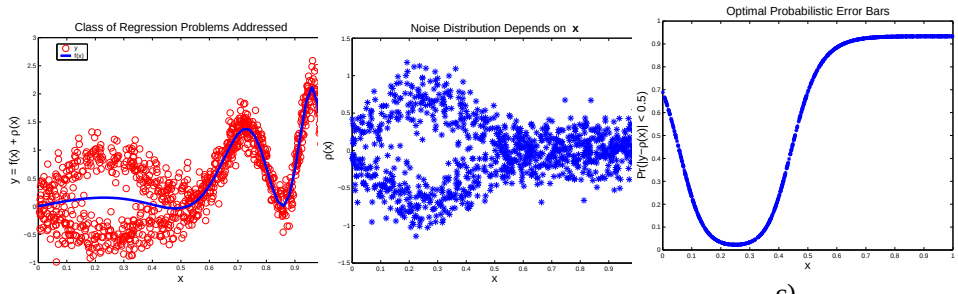
Abstract

Our goal is to accurately estimate the error in any prediction of a regression model. We propose a probabilistic regression framework for basis function regression models, which includes widely used kernel methods such as support vector machines and nonlinear ridge regression. The framework outputs a point specific estimate of the probability that the true regression surface lies between two user specified values, denoted by y_1 and y_2. More formally, given any $y_2 > y_1$, we estimate the $\Pr(y_1 \leq y \leq y_2|\mathbf{x}, \hat{f}(\mathbf{x}))$, where y is a true regression surface, $\mathbf{x}$ is the input, and $\hat{f}(\mathbf{x})$ is the basis function model. Thus the framework encompasses the less general standard error bar approach used in regression. We assume that the training data is independent and identically distributed (iid) from a stationary distribution, and make no specific distribution assumptions (e.g. no Gaussian or other specific distributions are assumed). Theory is presented showing that as the number of training points increases, estimates of $\Pr(y_1 \leq y \leq y_2|\mathbf{x}, \hat{f}(\mathbf{x}))$ approach the true value. Experimental evidence demonstrates that our framework gives reliable probability estimates, without sacrificing mean squared error regression accuracy.

1 Introduction

The statistics community has long studied regression models that predict an output $\hat{y}$, as well as an error bar estimate for the probability that the observed output y is within some ϵ of the prediction: i.e. $\Pr(|\hat{y} - y| \leq \epsilon)$ [5]. Estimates of $\Pr(|\hat{y} - y| \leq \epsilon)$ are useful in practice because they measure spread of observed regression values, allowing the user to make informed decisions about how predictions should be used. Although standard statistical techniques such as locally linear regression [3] can give very good error bar predictions for low dimensional problems, such techniques do not generally work well on complex, high dimensional, problem domains.

In contrast, the machine learning community has potentially powerful techniques for regression [4, 1], but very little attention has been given to solving the general accuracy regression accuracy estimation problem of finding $\Pr(y_1 \leq y \leq y_2|\mathbf{x}, \hat{f}(\mathbf{x}))$ given any $y_2 > y_1$. It is important to distinguish this problem from the one posed in Gaussian Process Regression [9, 8], where the goal is to obtain a error estimate on the model of the *mean* of the regression surface which is given by $\hat{f}(\mathbf{x})$. Our goal is to estimate the spread



a) Toy Data b) Noise Term c) $\Pr(-0.5 \leq \rho(x) \leq 0.5|x)$

Figure 1: An example of a the type of regression problem considered in this paper. The noise term is a function of the input x and it is not Gaussian.

of values about *any hypothesized* mean $\hat{f}(\mathbf{x})$. Applications of such models are plentiful and include such things as weather prediction (e.g. what is the likely spread of tomorrow's temperature values) and economics (e.g. with what certainty can next years GDP be predicted).

Section 2 presents our theoretical formulation. Section 3 gives a numerical implementation of this theory. Experimental results are given in Section 4 Finally, Section 5 concludes with a brief discussion of further research.

2 Theoretical Formulation

Let $\{(\mathbf{x}_1, y_1), ..., (\mathbf{x}_N, y_N)\}$ be a set of N training examples, where $\mathbf{x} \in \Gamma \subseteq \Re^d$ are independently and identically distributed (iid) from some stationary distribution D_x. The standard regression function formulation assumes that the outputs $y \in \Re$ is generated from:

$$y = f(\mathbf{x}) + \rho$$

where $f(\mathbf{x}) \in \Re$ is a single valued function defined on $\mathbf{x} \in \Re^d$, and ρ is a random variable with mean zero (i.e. $E[\rho] = 0$) and finite variance (i.e. $V[\rho] = c, 0 \leq c < \infty, c \in \Re$). This regression formulation assumes that $E[\rho]$ and $V[\rho]$ are independent of $\mathbf{x}$. Furthermore, the error analysis typically assumes that noise term ρ is Gaussian [9, 3]. Therefore, this standard framework *cannot* be used to analyze the noise in a regression problem given in Figure 1. Figure 1a shows a regression function where the noise is not Gaussian, and it changes as a function of the input x. This noise term is shown in Figure 1b. The variation, as a function of x, in the probability that the noise term lies between -0.5 and 0.5, is shown in Figure 1c. It is interesting to note that any method that does not use knowledge of noise at specific points $\mathbf{x}$, cannot in general be used on the class of problems depicted in Figure 1.

This paper assumes a more general regression formulation. We make the same distribution assumptions on $\mathbf{x}$, however, we assume that $y \in \Re$ is generated from:

$$y = f(\mathbf{x}) + \rho(\mathbf{x}) \tag{1}$$

where the random variable ρ has $E[\rho(\mathbf{x})] = 0$ and $V[\rho(\mathbf{x})] = c(\mathbf{x})$, $0 \leq c(\mathbf{x}) < \infty$, $c(\mathbf{x}) \in \Re$. Therefore the distribution of the noise term *depends* on $\mathbf{x}$. In addition, no Gaussian assumptions (or any other specific distributions assumptions) are made on the noise $\rho(\mathbf{x})$. As a result, the framework proposed in this paper can be applied to regression problems of the type in Figure 1.

We symbolize the probability function that generated ρ at a specific $\mathbf{x}$ as $h(\rho|\mathbf{x})$, and the cumulative distribution function (*cdf*) as:

$$H(\rho_1|\mathbf{x}) = \Pr(\rho \leq \rho_1|\mathbf{x}) = \int_{-\infty}^{\rho_1} h(\rho'|\mathbf{x}) d\rho' \tag{2}$$

Because $f(\mathbf{x})$ is constant at each point $\mathbf{x}$, the cumulative distribution function (*cdf*) of y at $\mathbf{x}$ is simply given by:

$$\begin{aligned}\Pr(y \leq y_1 | \mathbf{x}) &= \int_{-\infty}^{y_1} h(y' - f(\mathbf{x}) | \mathbf{x}) \, dy' \\ &= H(y_1 - f(\mathbf{x}) | \mathbf{x})\end{aligned}$$

Therefore, if we can exactly know the point specific cdf of the noise term $H(\rho_1 | \mathbf{x})$, we can solve the problem posed in this paper. Namely,

$$\begin{aligned}\Pr(y_1 \leq y \leq y_2 | \mathbf{x}) &= \Pr(y \leq y_2 | \mathbf{x}) - \Pr(y \leq y_1 | \mathbf{x}) \\ &= H(y_2 - f(\mathbf{x}) | \mathbf{x}) - H(y_1 - f(\mathbf{x}) | \mathbf{x})\end{aligned} \tag{3}$$

One contribution of this paper is a framework for estimating $H(\rho_1 | \mathbf{x})$ from training data.

2.1 Error cdfs for Basis Function Models

Intuitively, the point specific cumulative distribution function of the error term can be obtained by looking at the distribution of points that pass through that specific point. Formally, this simply consists of all outputs y that are generated at a specific input $\mathbf{x}$. However, from a practical standpoint, if $\mathbf{x}$ is high dimensional, data is in general sparse, and obtaining samples of y at any given $\mathbf{x}$ is not possible (this results from the well known curse of dimensionality problem which is especially prevalent when the regression function is non-linear [3]). However, if we restrict our class of regression models to be a superposition of basis functions, the problem becomes potentially more tractable in high dimensional domains. Specifically, let the regression models $\hat{f}(\mathbf{x})$ to be of the form:

$$\hat{f}(\mathbf{x}) = \sum_{i=1}^{M} a_i \phi_i(\mathbf{x}) + b \tag{4}$$

where for all $i \in 1, ..., M$, $\phi_i : \Re^d \mapsto \Re$, and $a_i, b \in \Re$. If we restrict ϕ_i to be a Mercer Kernel $K(\mathbf{x}_i, \mathbf{x})$, this gives the familiar Support Vector Machine Regression model [7]:

$$\hat{f}(\mathbf{x}) = \sum_{i=1}^{M} a_i K(\mathbf{x}_i, \mathbf{x}) + b \tag{5}$$

When the model is thus restricted, the problem of estimating an error *cdf* becomes linear in basis function space ϕ_i. We now must find all outputs y that are generated at a specific point in basis function space give by $(\phi_1(\mathbf{x}), ..., \phi_M(\mathbf{x}))$. Given this regression model representation, the problem constrained further to:

$$\begin{aligned}&\Pr(y_1 \leq y \leq y_2 | \Phi(\mathbf{x})) = \\ &H\left(y_2 - \hat{f}(\mathbf{x}) | \Phi(\mathbf{x})\right) - H\left(y_1 - \hat{f}(\mathbf{x}) | \Phi(\mathbf{x})\right)\end{aligned}$$

where $\Phi(\mathbf{x}) = (\phi_1(\mathbf{x}), ..., \phi_M(\mathbf{x}))$, $\hat{f}(\mathbf{x})$ is defined in (4), and the outputs y are obtained as defined in equation (1). It is interesting to note that the mean of the true error cdf $H(y - \hat{f}(\mathbf{x}) | \Phi(\mathbf{x}))$ in this space is *not* necessarily zero. The reason for this that the true regression function $f(\mathbf{x})$ in (1) is not necessarily exactly representable in a user specified basis function space, and therefore, in general, $f(\mathbf{x}) \neq \hat{f}(\mathbf{x})$. Therefore, if we can empirically estimate the local mean of $H(y - \hat{f}(\mathbf{x}) | \Phi(\mathbf{x}))$, we can potentially obtain a better approximation of $f(\mathbf{x})$ using:

$$\hat{y} = \hat{f}(\mathbf{x}) + E\left[H\left(y - \hat{f}(\mathbf{x}) | \Phi(\mathbf{x})\right)\right] \tag{6}$$

This leads us to the following theorem.

Theorem 1: *Given the above assumptions, and further assuming that $H(y - \hat{f}(\mathbf{x}) | \Phi(\mathbf{x}))$ is known exactly, let $\hat{f}(\mathbf{x}) : \Re^{\mathbf{d}} \mapsto \Re$ be any bounded function that has the form defined in equation (4). Then, for all $\mathbf{x}$ generated according to the distribution D_x, the following is holds:*

$$\left|\hat{f}(\mathbf{x}) + E\left[H\left(y_1 - \hat{f}(\mathbf{x}) | \Phi(\mathbf{x})\right)\right] - f(\mathbf{x})\right|$$

$$\leq$$

$$\left|\hat{f}\left(\mathbf{x}\right)-f\left(\mathbf{x}\right)\right|$$

Proof Sketch: If $f(\mathbf{x}) = \hat{f}(\mathbf{x})$, then the above equation becomes an equality because $E[H(y - \hat{f}(\mathbf{x})|\Phi(\mathbf{x}))] = 0$. If $f(\mathbf{x}) \neq \hat{f}(\mathbf{x})$, then $E[H(y - \hat{f}(\mathbf{x})|\Phi(\mathbf{x}))]$ measures how far $f(\mathbf{x})$ is from $\hat{f}(\mathbf{x})$. Therefore adding $E[H(y - \hat{f}(\mathbf{x})|\Phi(\mathbf{x}))]$ to $\hat{f}(\mathbf{x})$ must, by definition, bring it closer to $f(\mathbf{x})$. This completes the proof sketch.

Empirical evidence supporting this theorem is given in Section 4. In order to make this theoretical framework useful in practice, we need a numerical formulation for estimating $H(y - \hat{f}(\mathbf{x})|\Phi(\mathbf{x}))$. We refer to this estimate as $\hat{H}(y - \hat{f}(\mathbf{x})|\Phi(\mathbf{x}))$, and the next section describes how it is obtained.

3 Numerical Formulation

We assume a set of training examples $\{(\mathbf{x}_1, y_1), ..., (\mathbf{x}_N, y_N)\}$ generated according to equation (1). Given these examples, we want to estimate the probability that the true output y, at some specific $\mathbf{x}'$ generated according to the distribution D_x, falls between some user specified bounds $y_2 > y_1$. Given the theory in Section 2, we reduce this problem to:

$$\begin{aligned}&\hat{\Pr}\left(y_1 \leq y \leq y_2 \left|\Phi\left(\mathbf{x}\right)\right.\right) = \\ &\hat{H}\left(y_2 - \hat{f}\left(\mathbf{x}\right)\left|\Phi\left(\mathbf{x}\right)\right.\right) - \hat{H}\left(y_1 - \hat{f}\left(\mathbf{x}\right)\left|\Phi\left(\mathbf{x}\right)\right.\right)\end{aligned} \tag{7}$$

Similarly, we calculate the prediction at each point using (see (6)):

$$\hat{y} = \hat{f}(\mathbf{x}) + E\left[\hat{H}\left(y - \hat{f}\left(\mathbf{x}\right)\left|\Phi\left(\mathbf{x}\right)\right.\right)\right] \tag{8}$$

Our framework depends on how well we can estimate the point specific noise cdf $\hat{H}(y - \hat{f}(\mathbf{x})|\Phi(\mathbf{x}))$, which we describe next two sections. In Section 3.1 we assume that the training examples $\{(\mathbf{x}_1, y_1), ..., (\mathbf{x}_N, y_N)\}$ were *NOT* used to construct the regression model $\hat{f}(\mathbf{x})$, and show that these independent samples can be used to obtain $\hat{H}(y - \hat{f}(\mathbf{x})|\Phi(\mathbf{x}))$. In Section 3.2, we present a cross validation approach for getting this unbiased data.

3.1 Estimating $\hat{H}(y - \hat{f}(\mathbf{x})|\Phi(\mathbf{x}))$ From Unbiased Data

If we assume that $\{(\mathbf{x}_1, y_1), ..., (\mathbf{x}_N, y_N)\}$ where not used to construct the regression model $\hat{f}(\mathbf{x})$, then as $N \to \infty$, for any specific point $\mathbf{x}'$ we can obtain an infinite independent sample of points that $\{(\mathbf{x}_i, y_i), i = 1, 2, ...\}$ that satisfy $\Phi(\mathbf{x_i}) = \Phi(\mathbf{x}')$, where $\Phi(\mathbf{x_i}) = (\phi_1(\mathbf{x_i}), ..., \phi_M(\mathbf{x_i}))$ and $\Phi(\mathbf{x}') = (\phi_1(\mathbf{x}'), ..., \phi_M(\mathbf{x}'))$ are the basis functions. Given these outputs y' that correspond to the inputs in this set $\{(\mathbf{x}_i, y_i), i = 1, 2, ...\}$, we could directly estimate the *cdf* numerically [2]. The obvious problem with this approach is that, if $\mathbf{x}$ is a real valued vector, it is likely that there are *no* points in $\{(\mathbf{x}_i, y_i), i = 1, 2, ...\}$ that satisfy $\Phi(\mathbf{x_i}) = \Phi(\mathbf{x}')$. To address this problem we measure the distance, in basis function space, between $\Phi(\mathbf{x}')$ and the points $\{(\mathbf{x}_i, y_i), i = 1, 2, ...\}$. At first glance this approach may seem problematic because the number of basis functions may be large, which once more leads to the curse of dimensionality dimensionality [3]. However, we need not measure distance in the entire basis function space, only that part of it which lies on the regression model surface. And since this surface is linear in basis function space, we implicitly constrain our distance measures to the this hyperplane. Thus, given a threshold distance d_{min}, we obtain a set of points $\{(\mathbf{x}_i, y_i), i = 1, ..., k\}$ such that, for all $i = 1, ..., k$,

$$d_{\min} \geq \frac{1}{M}\left\|\Phi\left(\mathbf{x}_i\right) - \Phi\left(\mathbf{x}'\right)\right\|^2 \tag{9}$$

These points are then used to estimate $\hat{H}(y - \hat{f}(\mathbf{x}')|\Phi(\mathbf{x}'))$ by calculating an empirical cumulative distribution function (*ecdf*) [2], which is a standard function in the Matlab statistics toolbox (also known as the Kaplan-Meier cumulative distribution function).

There is a tradeoff here in choosing d_{min}. If it is too small, the *ecdf* will not be an accurate estimate of the true *cdf*. If d_{min} is too big, it will include a region that is too large, making the estimate of the error *not* point specific. To address this, we take a cross-validation approach. The property of Kaplan-Meier cdf that we exploit is that, given a confidence level of $100(1-\alpha)\%$, it returns a range of maximum and minimum *cdf* estimates. By randomly dividing the points $\{(\mathbf{x}_i, y_i), i = 1, 2, ...\}$ into two sets, we can use cross validation to decide when the first *cdf* of one set is within the $100(1-\alpha)\%$ confidence interval of the second. When d_{min} is large enough so that this is true, we are $100(1-\alpha)\%$ confident that our estimates of $\hat{H}(y - \hat{f}(\mathbf{x}')|\Phi(\mathbf{x}'))$ is accurate.

We now state the following theorem.

Theorem 2: *Assume that* $\{(\mathbf{x}_1, y_1), ..., (\mathbf{x}_N, y_N)\}$ *where* not *used to construct the regression model* $\hat{f}(\mathbf{x})$. *Assume also that* $f(\mathbf{x})$, $\hat{f}(\mathbf{x})$ *and* $\rho(\mathbf{x})$ *are define on a compact set. Then, as the number of training examples approaches infinity (*$N \rightarrow \infty$*) and* $d_{min} \rightarrow 0$, *for any specific* $\mathbf{x}'$ *generated according to the distribution* D_x, $E[|\hat{H}(y - \hat{f}(\mathbf{x}')|\Phi(\mathbf{x}')) - H(y - \hat{f}(\mathbf{x}')|\Phi(\mathbf{x}'))|] \rightarrow 0$, *where* $\hat{H}(y - \hat{f}(\mathbf{x}')|\Phi(\mathbf{x}'))$ *is estimated using the Kaplan-Meier* cdf *as defined above.*

Proof Sketch: The proof follows directly from the properties of the Kaplan-Meier *cdf* and the definition of compact set.

The importance of the above theorem is that it establishes the convergence of our method to the true point specific *cdf* noise estimates as the sample size increases.

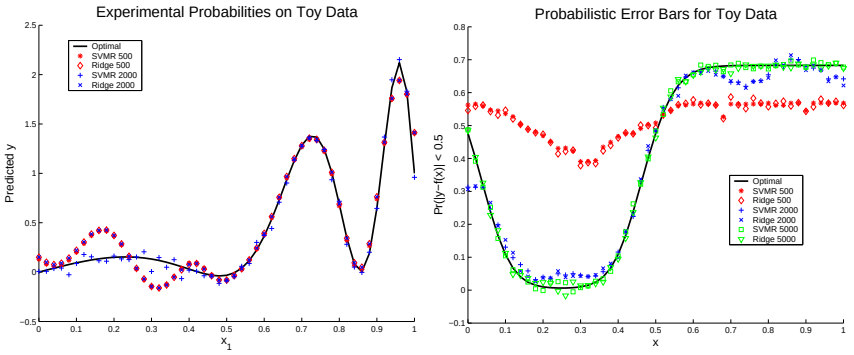
3.2 Obtaining Unbiased Data

In order to ensure that the data used to estimate $\hat{H}(y - \hat{f}(\mathbf{x}')|\Phi(\mathbf{x}'))$ is unbiased, we use a standard Q_f fold cross validation technique. We separate the data $D = \{(\mathbf{x}_1, y_1), ..., (\mathbf{x}_N, y_N)\}$ into Q_f sets of approximately equal size $T_1, ..., T_{Q_f}$. Then, for $i = 1, ..., Q_f$ we generate Q_f models $\hat{f}_1(\mathbf{x}), ..., \hat{f}_{Q_f}(\mathbf{x})$, where model $\hat{f}_i(\mathbf{x})$ is constructed using data set $\{D - T_i\}$, allowing the points in T_i to be unbiased with respect to $\hat{f}_i(\mathbf{x})$. Therefore, every point in the original set $D = \{(\mathbf{x}_1, y_1), ..., (\mathbf{x}_N, y_N)\}$ is unbiased with respect to one model $\hat{f}_i(\mathbf{x})$. By measuring the distance for d_{min} in (9) using the basis functions for which a point was *NOT* used to build the corresponding model, we obtain an unbiased set for estimating $\hat{H}(y - \hat{f}(\mathbf{x}')|\Phi(\mathbf{x}'))$.

3.3 Algorithm Summary

The final Probabilistic Regression model is defined by: 1) a single basis function model $\mathbf{a} = (a_1, ..., a_k)$, $\Phi(\mathbf{x}) = (\phi_1(\mathbf{x}), ..., \phi_k(\mathbf{x}))$ and b as defined in (4); a set of values $y_1, ..., y_n$ (see (4)) for each training point input $\mathbf{x}_1, ..., \mathbf{x}_N$ obtained via cross validation as described above; and finally a vector $(\phi_1(\mathbf{x}_i), ..., \phi_k(\mathbf{x}_i))$ for each training input. For each test point $\mathbf{x}$, we calculate $\Pr(y_1 \leq y \leq y_2|\mathbf{x}, \hat{f}(\mathbf{x}))$ as follows (note that the we use the *ecdf* function in the *matlab statistics toolbox*):

1. *Project* $\mathbf{x}$ *into basis function space.*
2. *Find* $d_{\min}$. Choose a window size $d_{\min}$ that gives $100(1-\alpha)\%$ confidence in estimates of $\hat{H}(y - \hat{f}(\mathbf{x}')|\Phi(\mathbf{x}'))$.
3. *Estimate probability and locally optimal mean.* Use (7) to estimate $\Pr(y_1 \leq y \leq y_2|\mathbf{x}, \hat{f}(\mathbf{x}))$ and (8) to estimate $\hat{y}$.



a) Mean Results b) $\Pr(0.2 \leq y \leq 0.2|x_1)$

Figure 2: Results Symmetric Toy Example

4 Experimental Results

4.1 Learning Algorithm Implementation Details

The regression model formulation proposed here requires 1) a specification of α for the $100(1-\alpha)\%$ confidence interval in estimating the empirical *cdf* noise (see Section 3.1); 2) the number of folds Q_f used to obtain unbiased samples (see Section 3.2); and, 3) the basis function learning algorithm used to construct the regression model (4) for each fold. Unless specified otherwise, we use $alpha = 0.05$ and $Q_f = 10$ in the experiments reported here.

We experimented with two types of basis function algorithms: ridge regression with Gaussian Kernels, and support vector regression. For ridge regression [3] we set the ridge parameter to $1e-6$. For support vector regression we used libSVM (www.csie.ntu.edu.tw/∼cjlin/libsvm/).

4.2 Toy Data

The toy regression example used here is the one dimensional problem shown in Figure 1. The data was generated according to:

$$f(x_1) = x_1 - \sin\left(2\pi x_1^3\right)\cos\left(2\pi x_1^3\right)\exp\left(x_1^4\right)$$

The noise term ρ is dependent on x_1 as shown in Figure 2b and was calculated as follows:

$$\rho(\mathbf{x}) = \begin{cases} \mathrm{N}\left[0.7\exp\left(\frac{-\|x_1-0.25\|^2}{0.05}\right), 0.2\right] \rightarrow \Pr(0.5) \\ \mathrm{N}\left[-0.7\exp\left(\frac{-\|x_1-0.25\|^2}{0.05}\right), 0.2\right] \rightarrow \Pr(0.5) \end{cases}$$

where $\mathrm{N}(m,\sigma)$ is a Gaussian distribution with mean m and standard deviation σ, and $\rightarrow Pr(0.5)$ means with probability 0.5 - therefore the noise term is equally likely to be above and below the mean $f(x_1)$. Given this definition of noise, the exact $\Pr(0.2 \leq y \leq 0.2|x_1)$ is plotted in Figure 2b.

We experimented with two type of basis function regression models. Both used a Gaussian kernel with $2\sigma^2 = 0.01$. The first model type was a kernel ridge regression [3] model with the ridge parameter to $1e-6$. The second was the ν-SVR algorithm [6] with $\nu = 0.5$ and $C = 1$.

The results for estimating the mean function $f(\mathbf{x})$ using 500 and 2000 training examples are presented in Figure 2a. One can see that both algorithms do fairly well, with ν-SVR based on 2000 examples doing slightly better than ridge regression. The results for predicting the $\Pr(0.2 \leq y \leq 0.2|x_1)$ are given in Figure 2b, for training set sizes of 500, 2000 and 5000. The proposed algorithm, using both ridge and ν-SVR gave poor predic-

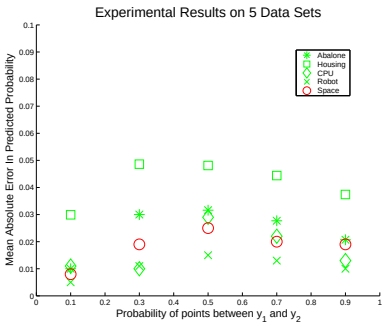


Figure 3: *Results on five datasets.*

Table 1: Regression Data Sets

Data	Number of Examples	Number of Features	Training Size	Testing Size	Q_f	Number of Random Experiments
abalone	4177	8	3133	1044	10	50
housing	505	13	455	51	10	100
cpu small	8192	12	4096	4096	5	10
robot arm	20000	12	15000	5000	2	1
space ga	3106	6	1500	1606	10	10

tions of $\Pr(0.2 \leq y \leq 0.2|x_1)$ when only 500 training samples were used. However, when 2000 training samples are used, the proposed algorithm accurately predicts the probabilities. Furthermore, with 5000 training samples the the predictions $\Pr(0.2 \leq y \leq 0.2|x_1)$ closely match the true values. Therefore, as predicted by *Theorem 2*, as the training sample increases, the approximations of $\hat{H}(y - \hat{f}(\mathbf{x}')|\Phi(\mathbf{x}'))$ improve.

4.3 Benchmark Data

We applied the proposed algorithm to 5 standard regression datasets. These are summarized in Table 1. The *housing* dataset was obtained from the UCI Machine Learning Repository (*http://www.ics.uci.edu/ mlearn/MLRepository.html*). The *abalone* and *cpu* small sets were obtained from Delve (*http://www.cs.toronto.edu/∼delve/*). The *space ga* dataset was obtained from StatLib (*http://lib.stat.cmu.edu/datasets/*). The *robot arm* dataset was originally used in [4] and contains 4 outputs - the results reported here on this dataset are averaged over these outputs. Table 1 indicates the total number of examples, the number of features, the training and test set sizes, the number of folds Q_f used to obtain unbiased samples, and the number of random tests done.

We used the ν-SVR algorithm [6] to build the regression models $\hat{f}(\mathbf{x})$, with the Gaussian Kernel. For all experiments we set $\nu = 0.5$, $C = 500$ and, following [6] the Gaussian kernel σ such that $2\sigma^2 = 0.3d$, where d is the dimension of the data as defined in 1. All inputs in the datasets were scaled to lie between 0 and 1.

To evaluate the probabilities generated by the our framework, we divided each test data set outputs into intervals bounded by y_1 and y_2, such that the observed frequencies in the first interval is 0.1, in the second interval is 0.3, in the third interval is 0.5, in the fourth interval is 0.7, and finally in the fifth interval is 0.9. These observed frequencies can be compared to the actual predicted $\Pr(y_1 \leq y \leq y_2|\mathbf{x}, \hat{f}(\mathbf{x}))$. The mean absolute difference between the predicted and observed probabilities (i.e. frequency) is shown in Figure 3. The x-axis shows the predicted probability and the y axis shows the mean absolute error in this prediction over all runs. We can see that the probability estimates quite accurate, falling within a probability of 0.05 for the small Housing Dataset, and much lower for the larger

Table 2: MSE Error Rates on Regression Data Sets

Data	Standard SVM	Locally Modified SVM
abalone	5.2	4.6
housing	9.8	9.4
cpu small	16.0	15.9
robot arm	3.2	3.1
space ga	0.012	0.011

datasets. Once more showing that more data leads to better probability estimates.

Finally, the mean squared error rates of our algorithm are given in table 2 (note that the predictions of $\hat{y}$ are made as specified in equation (8)). We can see that the proposed algorithm slightly outperforms an SVM regression model (generated using the same learning parameters) who's mean predictions have not been locally modified. This result supports *Theorem 1*, which states that local estimates of the mean can improve overall regression accuracy.

5 Conclusion

The goal of this paper is to formulate a general framework for predicting error rates in basis function regression models, which includes the widely used support vector regression formulation, as well as kernel based ridge regression. Given any user specified $y_2 > y_1$, we estimate the $\Pr(y_1 \leq y \leq y_2 | \mathbf{x}, \hat{f}(\mathbf{x}))$, which strictly depends on the input $\mathbf{x}$. Our formulation is based on empirically estimating the *point specific* cumulative distribution functions of the noise term. The observation that makes this feasible is that the regression problem is linear in basis function space, allowing us to effectively group points together for estimating the cumulative distribution function of the noise. Our approach does not make specific distribution assumptions, such as Gaussian noise. In addition, under appropriate smoothness and compactness assumptions, we can show that estimates of the cumulative distribution function of the noise converge to the true value as the learning sample size increases. Experimental results indicate that our method gives good estimates of $\Pr(y_1 \leq y \leq y_2 | \mathbf{x}, \hat{f}(\mathbf{x}))$, as well as mean squared regression errors that match those obtained by support vector regression.

References

[1] Ronan Collobert and Samy Bengio. Svmtorch: Support vector machines for large-scale regression problems. *Journal of Machine Learning Research*, 1:143–160, 2001.

[2] D.R. Cox and D. Oakes. *Analysis of Survival Data*. Chapman and Hall, London, 1984.

[3] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: data mining, inference and prediction*. Springer, 2001.

[4] M. I. Jordan and R. A. Jacobs. Hierarchical mixtures of experts and the em algorithm. *Neural Computation*, 6:181–214, 1994.

[5] John Neter, William Wasserman, and Michael H. Kutner. *Applied Linear Statistical Models*. IRVIN, Burr Ridge, Illinois, 3rd edition, 1990.

[6] B. Schölkopf, P. Bartlett, B. Smola, and R. Williamson. Shrinking the tube: A new support vector regression algorithm. In M. S. Kearns, S. A. Solla, and D. A. Cohn, editors, *Advances in Neural Information Processing Systems 11*, Cambridge, MA, 1999. MIT Press.

[7] B. Scholkopf and A.J. Smola. *Learning with Kernels*. MIT Press, 2002. 412-414.

[8] M. Tipping. The relevance vector machine. In S.A. Solla, T.K. Leen, and K.-R. Müller, editors, *Advances in Neural Information Processing Systems 12*, Cambridge, MA, 2000. MIT Press.

[9] Christopher K. I. Williams and Carl Edward Rasmussen. Gaussian processes for regression. In David S. Touretzky, Michael C. Mozer, and Michael E. Hasselmo, editors, *Advances in Neural Information Processing Systems 8*, Cambridge, MA, 1995. MIT Press.