THE UNIVERSITY *of* EDINBURGH

Edinburgh Research Explorer

# Incremental online learning in high dimensions

OPEN ACCESS

# Incremental Online Learning in High Dimensions

**Sethu Vijayakumar**
*sethu.vijayakumar@ed.ac.uk*
*School of Informatics, University of Edinburgh,*
*Edinburgh EH9 3JZ, U.K.*

**Aaron D'Souza**
*adsouza@usc.edu*
**Stefan Schaal**
*sschaal@usc.edu*
*Department of Computer Science, University of Southern California,*
*Los Angeles, CA 90089-2520, U.S.A.*

**Locally weighted projection regression (LWPR) is a new algorithm for incremental nonlinear function approximation in high-dimensional spaces with redundant and irrelevant input dimensions. At its core, it employs nonparametric regression with locally linear models. In order to stay computationally efficient and numerically robust, each local model performs the regression analysis with a small number of univariate regressions in selected directions in input space in the spirit of partial least squares regression. We discuss when and how local learning techniques can successfully work in high-dimensional spaces and review the various techniques for local dimensionality reduction before finally deriving the LWPR algorithm. The properties of LWPR are that it (1) learns rapidly with second-order learning methods based on incremental training, (2) uses statistically sound stochastic leave-one-out cross validation for learning without the need to memorize training data, (3) adjusts its weighting kernels based on only local information in order to minimize the danger of negative interference of incremental learning, (4) has a computational complexity that is linear in the number of inputs, and (5) can deal with a large number of—possibly redundant—inputs, as shown in various empirical evaluations with up to 90 dimensional data sets. For a probabilistic interpretation, predictive variance and confidence intervals are derived. To our knowledge, LWPR is the first truly incremental spatially localized learning method that can successfully and efficiently operate in very high-dimensional spaces.**

## 1 Introduction

Despite the recent progress in statistical learning, nonlinear function approximation with high-dimensional input data remains a nontrivial problem, especially in incremental and real-time formulations. There is, however, an increasing number of problem domains where both of these properties are important. Examples include the online modeling of dynamic processes observed by visual surveillance, user modeling for advanced computer interfaces and game playing, and the learning of value functions, policies, and models for learning control, particularly in the context of high-dimensional movement systems like humans or humanoid robots. An ideal algorithm for such tasks needs to avoid potential numerical problems from redundancy in the input data, eliminate irrelevant input dimensions, keep the computational complexity of learning updates low while remaining data efficient, allow online incremental learning, and, of course, achieve accurate function approximation and adequate generalization.

When looking for a learning framework to address these goals, one can identify two broad classes of function approximation methods: (1) methods that fit nonlinear functions globally, typically by input space expansions with predefined or parameterized basis functions and subsequent linear combinations of the expanded inputs; and (2) methods that fit nonlinear functions locally, usually by using spatially localized simple (e.g., low-order polynomial) models in the original input space and automatically adjusting the complexity (e.g., number of local models and their locality) to accurately account for the nonlinearities and distributions of the target function. Interestingly, the current trends in statistical learning have concentrated on methods that fall primarily in the first class of global nonlinear function approximators, for example, gaussian process regression (GPR; Williams & Rasmussen, 1996), support vector machine regression (SVMR; Smola & Schölkopf, 1998), and variational Bayes for mixture models (VBM; Ghahramani & Beal, 2000).[1] In spite of the solid theoretical foundations that these approaches possess in terms of generalization and convergence, they are not necessarily the most suitable for online learning in high-dimensional spaces. First, they require an a priori determination of the right modeling biases. For instance, in the case of GPR and SVMR, these biases involve selecting the right function space in terms of the choice of basis or kernel functions (Vijayakumar & Ogawa, 1999), and in VBM the biases are concerned with the right number of latent variables and proper initialization.[2] Second, all of these recent function approximator methods were developed

---

[1] Mixture models are actually in between global and local function approximators since they use local model fitting but employ a global optimization criterion.

[2] It must be noted that some recent work (Schölkopf, Burgess, & Smola, 1999) has started to look at model selection for SVMs and GPRs and automatic determination of number of latent models for VBM (Ghahramani & Beal, 2000).

primarily for batch data analysis and are not easily or efficiently adjusted for incrementally arriving data. For instance, in SVMR, adding a new data point can drastically change the outcome of the global optimization problem in terms of which data points actually become support vectors, such that all (or a carefully selected subset of) data have to be kept in memory for reevaluation. Thus, adding a new data point in SVMR is computationally rather expensive, a property that is also shared by GPR. VBM suffers from similar problems due to the need for storing and reevaluating data when adding new mixture components (Ueda, Nakano, Ghahramani, & Hinton, 2000). In general, it seems that most suggested Bayesian learning algorithms are computationally too expensive for real-time learning because they tend to represent the complete joint distribution of the data, albeit as a conditionally independent factored representation. As a last point, incremental approximation of functions with global methods is prone to lead to negative interference when input distributions change (Schaal & Atkeson, 1998). Such changes are, however, typical in many online learning tasks.

In contrast to the global learning methods described above, function approximation with spatially localized models is rather well suited for incremental and real-time learning, particularly in the framework of locally weighted learning (LWL; Atkeson, Moore, & Schaal, 1997). LWL methods are very useful when there is limited knowledge about the model complexity such that the model resources can be increased in a purely incremental and data-driven fashion, as demonstrated in previous work (Schaal & Atkeson, 1998). However, since these techniques allocate resources to cover the input space in a localized fashion, in general, with an increasing number of input dimensions, they encounter an exponential explosion in the number of local models required for accurate approximation—often referred to as the "curse of dimensionality" (Scott, 1992). Hence, at the outset, high-dimensional function approximation seems to be computationally infeasible for spatially localized learning.

Some efficient global learning methods with automatic resource allocation in high-dimensional spaces, however, have been employed successfully by using techniques of projection regression (PR). PR copes with high-dimensional inputs by decomposing multivariate regressions into a superposition of single-variate regressions along a few selected projections in input space. The major difficulty of PR lies in the selection of efficient projections, that is, how to achieve the best-fitting result with as few univariate regressions as possible. Among the best-known PR algorithms are projection pursuit regression (Friedman & Stutzle, 1981) and its generalization in the form of generalized additive models (Hastie & Tibshirani, 1990). Sigmoidal neural networks can equally be conceived of as a method of projection regression, in particular when new projections are added sequentially, as in cascade correlation (Fahlman & Lebiere, 1990).

In this letter, we suggest a method of extending the beneficial properties of spatially localized learning to high-dimensional function approximation

problems. The prerequisite of our approach is that the high-dimensional learning problems we address have locally low-dimensional distributions, an assumption that holds for a large class of real-world data (Tenenbaum, de Silva, & Langford, 2000; Roweis & Saul, 2000; Vlassis, Motomura, & Krose, 2002; D'Souza, Vijayakumar, & Schaal, 2001). If distributions are locally low dimensional, the allocation of local models can be restricted to these thin distributions, and only a tiny part of the entire high-dimensional space needs to be filled with local models. Thus, the curse of dimensionality of spatially localized model fitting can be avoided. Under these circumstances, an alternative method of projection regression can be derived, focusing on finding efficient local projections. Local projections can be used to accomplish local function approximation in the neighborhood of a given query point with traditional LWL approaches, thus inheriting most of the statistical properties from well-established methods (Hastie & Loader, 1993; Atkeson et al. 1997). As this letter will demonstrate, the resulting learning algorithm combines the fast, efficient, and incremental capabilities of LWL techniques while alleviating the problems faced due to high-dimensional input domains through local projections.

In the following sections, we first review approaches of how to find good local projections by looking into various schemes for performing dimensionality reduction for regression, including principal component regression, factor analysis, and partial least squares regression. Afterward, we embed the most efficient and robust of these projection algorithms in an incremental nonlinear function approximator (Vijayakumar & Schaal, 1998) capable of automatically adjusting the model complexity in a purely data-driven fashion. In several evaluations, on both synthetic and real-world data, the resulting incremental learning system demonstrates high accuracy for function fitting in very high-dimensional spaces, robustness toward irrelevant and redundant inputs, as well as low computational complexity. Comparisons will prove the competitiveness with other state-of-the-art learning systems.

## 2  Local Dimensionality Reduction for Locally Weighted Learning

Assuming that data are characterized by locally low-dimensional distributions, efficient algorithms are needed to exploit this property. We will focus on locally weighted learning (LWL) methods (Atkeson et al., 1997) because they allow us to adapt a variety of linear dimensionality-reduction techniques for the purpose of nonlinear function approximation (see section 3) and because they are easily modified for incremental learning. LWL-related methods have also found widespread application in mixture models (Jordan & Jacobs, 1994; Xu, Jordan, & Hinton, 1995; Ghahramani & Beal, 2000) such that the results of this section can contribute to this field too.

The learning problems considered here assume the standard regression model:

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

where $\mathbf{x}$ denotes the $N$-dimensional input vector, $y$ the (for simplicity) scalar output, and $\epsilon$ a mean-zero random noise term. When only a local subset of data in the vicinity of a point $\mathbf{x}_c$ is considered and the locality is chosen appropriately, a low-order polynomial can be employed to model this local subset. Due to a favorable compromise between computational complexity and quality of function approximation (Hastie & Loader, 1993), we choose linear models

$$y = \boldsymbol{\beta}^T \mathbf{x} + \epsilon. \tag{2.1}$$

A measure of locality for each data point, the weight $w_i$, is computed from a gaussian kernel,

$$w_i = \exp(-0.5(\mathbf{x}_i - \mathbf{x}_c)^T \mathbf{D}(\mathbf{x}_i - \mathbf{x}_c)), \quad \text{and} \quad \mathbf{W} \equiv \text{diag}\{w_1, \ldots, w_M\}, \tag{2.2}$$

where $\mathbf{D}$ is a positive semidefinite distance metric that determines the size and shape of the neighborhood contributing to the local model (Atkeson et al., 1997). The weights $w_i$ will enter all following algorithms to ensure spatial localization in input space. Without loss of generality, we assume zero mean of all inputs and outputs in our algorithms, which is ensured by subtracting the weighted mean $\overline{\mathbf{x}}$ or $\overline{y}$ from the data, where

$$\overline{\mathbf{x}} = \sum_{i=1}^{M} w_i \mathbf{x}_i \bigg/ \sum_{i=1}^{M} w_i, \quad \text{and} \quad \overline{y} = \sum_{i=1}^{M} w_i y_i \bigg/ \sum_{i=1}^{M} w_i, \tag{2.3}$$

and $M$ denotes the number of data points. The input data are summarized in the rows of the matrix $\mathbf{X} = [\mathbf{x}_1 \ \mathbf{x}_2, \ldots, \mathbf{x}_M]^T$, the corresponding outputs are the coefficients of the vector $\mathbf{y}$, and the corresponding weights, determined from equation 2.2, are in the diagonal matrix $\mathbf{W}$.

As candidate algorithms for local dimensionality reduction, we consider two techniques, factor analysis and partial least squares regression. Factor analysis (Everitt, 1984) is a density estimation technique that assumes that the observed data were generated from a lower-dimensional process, characterized by $k$ latent or hidden variables $\mathbf{v}$ that are all independently distributed with mean zero and unit variance. The observed variables are generated from the latent variables through the transformation matrix $\mathbf{U}$

and additive mean zero independent noise $\epsilon$ with diagonal covariance matrix $\Omega$:

$$\mathbf{z} = \mathbf{U}\mathbf{v} + \epsilon, \tag{2.4}$$

where $E\{\epsilon\epsilon^T\} = \Omega$ and $E$ denotes the expectation operator. If both $\mathbf{v}$ and $\epsilon$ are normally distributed, the parameters $\Omega$ and $\mathbf{U}$ can be obtained iteratively by the expectation-maximization algorithm (EM) (Rubin & Thayer, 1982).

Factor analysis is superset for several dimensionality-reduction algorithms. For $\mathbf{z} = \mathbf{x}$ and $E\{\epsilon\epsilon^T\} = \sigma^2\mathbf{I}$, we obtain principal component analysis in input space (Tipping & Bishop, 1999). For the purpose of regression, the lower-dimensional representation $\mathbf{v}$ would serve as a new input to the regression problem—an algorithm called principal component regression (PCR). However, it is well documented that PCR has the huge danger of eliminating low-variance input dimensions that are nevertheless crucial for the regression problem, thus leading to inferior function approximation results (Frank & Friedman, 1993; Schaal, Vijayakumar, & Atkeson, 1998).

Thus, for the purpose of regression, it is more useful to use factor analysis in joint space of input and output data:

$$\mathbf{z} = \begin{bmatrix} \mathbf{x} \\ y \end{bmatrix}, \epsilon = \begin{bmatrix} \epsilon_x \\ \epsilon_y \end{bmatrix}, E\{\epsilon\epsilon^T\} = \Omega. \tag{2.5}$$

Again, if we assume $E\{\epsilon\epsilon^T\} = \sigma^2\mathbf{I}$, we obtain the PCA solution, this time performed in joint space. PCA algorithms are appealing, as they can be solved rather efficiently. Alternatively, without additional constraints on $\Omega$ (except that it is diagonal), the most powerful factor analysis algorithm for dimensionality reduction is obtained and requires an iterative EM solution. In both joint-space formulations, the regression parameters $\boldsymbol{\beta}$ (cf. equation 2.1) can be recovered by computing the expectation of $p(y|\mathbf{x})$, which is obtained from standard manipulations of the normally distributed joint distribution $p(\mathbf{x}, \mathbf{v}, y)$ (Schaal et al., 1998). While empirical evaluation in Schaal et al. (1998) verified that the unconstrained (i.e., non-PCA) version of joint-space factor analysis performs very well for regression, it also highlighted an important problem. As a density estimation method, factor analysis crucially depends on representing the complete latent space $\mathbf{v}$ of the joint input vector $\mathbf{z}$; otherwise, performance degrades severely. Hence, even if there are input dimensions that are irrelevant for the regression, they need to be represented in the latent variable vector unless they are redundant with combinations of other inputs. This property is problematic for our goals, as we expect a large number of irrelevant inputs in high-dimensional learning problems. The inferior performance of factor analysis when the latent space is underestimated also makes it hard to apply it in constructive algorithms—those

Table 1: Locally Weighted Implementation of Partial Least Squares Regression.

---

1. Initialize: $\mathbf{X}_{res} = \mathbf{X}$, $\mathbf{y}_{res} = \mathbf{y}$

2. Repeat for $r = 1$ to $R$ (No. of projections)
    (a) $\mathbf{u}_r = \mathbf{X}_{res}^T \mathbf{W} \mathbf{y}_{res}$ where $\mathbf{W} \equiv \mathrm{diag}\{w_1, \ldots, w_M\}$ is the matrix of locality weights.
    (b) $\beta_r = \mathbf{z}_r^T \mathbf{W} \mathbf{y}_{res} / (\mathbf{z}_r^T \mathbf{W} \mathbf{z}_r)$ where $\mathbf{z}_r = \mathbf{X}_{res} \mathbf{u}_r$.
    (c) $\mathbf{y}_{res} = \mathbf{y}_{res} - \mathbf{z}_r \beta_r$.
    (d) $\mathbf{X}_{res} = \mathbf{X}_{res} - \mathbf{z}_r \mathbf{p}_r{}^T$ where $\mathbf{p}_r = \mathbf{X}_{res}^T \mathbf{W} \mathbf{z}_r / (\mathbf{z}_r^T \mathbf{W} \mathbf{z}_r)$.

---

that grow the latent space in a data-driven way until the full latent space is recovered. Since the regression results are of low quality until the full latent space is recovered, predictions of the learning system cannot be trusted until a significant amount of data has been encountered, with the open problem of how to quantify "significant."

As a surprising result of the empirical comparisons of local dimensionality-reduction techniques presented in Schaal et al. (1998), one particular algorithm, partial least squares regression (PLS) (Wold, 1975; Frank & Friedman, 1993), achieved equally good and more robust results than factor analysis for regression without any of the noted problems. PLS, a technique extensively used in chemometrics, recursively computes orthogonal projections of the input data and performs single-variable regressions along these projections on the residuals of the previous iteration step. Table 1 provides an outline of the PLS algorithm, derived here for implementing the locally weighted version (LWPLS). The key ingredient in PLS is to use the direction of maximal correlation between the residual error and the input data as the projection direction at every regression step. Additionally, PLS regresses the inputs of the previous step against the projected inputs $\mathbf{z}$ in order to ensure the orthogonality of all the projections $\mathbf{u}$ (step 2d). Actually, this additional regression could be avoided by replacing $\mathbf{p}$ with $\mathbf{u}$ in Step 2d, similar to techniques used in principal component analysis (Sanger, 1989). However, using this regression step leads to better performance of the algorithm as PLS chooses the most effective projections if the input data have a spherical distribution: in the spherical case, with only one projection, PLS will find the direction of the gradient and achieve optimal regression results. The regression step in 2d chooses the reduced input data $\mathbf{X}_{res}$ such that the resulting data vectors have minimal norms and, hence, push the distribution of $\mathbf{X}_{res}$ to become more spherical. An additional consequence of step 2d is that all the projections $\mathbf{z}_r$ become uncorrelated, that is, $\mathbf{z}_j^T \mathbf{z}_r = 0 \ \forall j \neq r$, a property that will be important in the derivations below.

Due to all these consideration, we will choose PLS as the basis for an incremental nonlinear function approximator, which, in the next sections, will be demonstrated to have appealing properties for nontrivial function fitting problems.

## 3 Locally Weighted Projection Regression

For nonlinear function approximation, the core concept of our learning system, locally weighted projection regression (LWPR), is to find approximations by means of piecewise linear models (Atkeson et al., 1997). Learning involves automatically determining the appropriate number of local models $K$, the parameters $\beta_k$ of the hyperplane in each model, and also the region of validity, called receptive field (RF), parameterized as a distance metric $\mathbf{D}_k$ in a gaussian kernel (cf. equation 2.2):

$$w_k = \exp\left(-\frac{1}{2}(\mathbf{x} - \mathbf{c}_k)^T \mathbf{D}_k (\mathbf{x} - \mathbf{c}_k)\right). \tag{3.1}$$

Given a query point $\mathbf{x}$, every linear model calculates a prediction $\hat{y}_k(\mathbf{x})$. The total output of the learning system is the normalized weighted mean of all $K$ linear models,

$$\hat{y} = \sum_{k=1}^{K} w_k \hat{y}_k \Big/ \sum_{k=1}^{K} w_k, \tag{3.2}$$

also illustrated in Figure 1. The centers $\mathbf{c}_k$ of the RFs remain fixed in order to minimize negative interference during incremental learning that could occur due to changing input distributions (Schaal & Atkeson, 1998). Local models are created on an as-needed basis as described in section 3.2. Table 2 provides a reference list of indices and symbols that are used consistently across the description of the LWPR algorithm.

**3.1 Learning with LWPR.** Despite its appealing simplicity, the piecewise linear modeling approach becomes numerically brittle and computationally too expensive in high-dimensional input spaces when using ordinary linear regression to determine the local model parameters (Schaal & Atkeson, 1998). Thus, we will use locally weighted partial least squares regression within each local model to fit the hyperplane. As a significant computational advantage, we expect that far fewer projections than the actual number of input dimensions are needed for accurate learning. The next sections describe the necessary modifications of PLS for this implementation, embed the local regression into the LWL framework, explain a method of automatic distance metric adaptation, and finish with a complete nonlinear learning scheme, called locally weighted projection regression (LWPR).

*3.1.1 Incremental Computation of Projections and Local Regression.* For incremental learning, that is, a scheme that does not explicitly store any training data, the sufficient statistics of the learning algorithm need to be accumulated in appropriate variables. Table 3 provides suitable incremental update
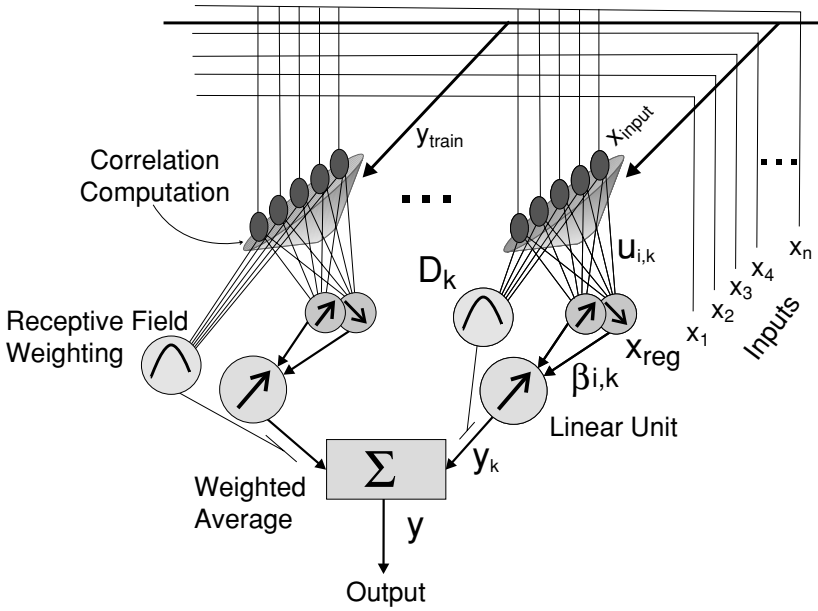
Figure 1: Information processing unit of LWPR.

rules. The variables $a_{zz,r}$, $a_{zres,r}$, and $\mathbf{a}_{xz,r}$ are sufficient statistics that enable us to perform the univariate regressions in step 2c.1.2 and step 2c.2.2, similar to recursive least squares, that is, a fast Newton-like incremental learning technique. $\lambda \in [0, 1]$ denotes a forgetting factor that allows exponential forgetting of older data in the sufficient statistics. Forgetting is necessary in incremental learning since a change of some learning parameters will affect a change in the sufficient statistics. Such forgetting factors are a standard technique in recursive system identification (Ljung & Soderstrom, 1986). It can be shown that the prediction error of step 2b corresponds to the leave-one-out cross-validation error of the current point after the regression parameters were updated with the data point. Hence, it is denoted by $e_{cv}$.

In Table 3, for $R = N$, that is, the same number of projections as the input dimensionality, the entire input space would be spanned by the projections $\mathbf{u}_r$ and the regression results would be identical to that of ordinary linear regression (Wold, 1975). However, once again, we emphasize the important properties of the local projection scheme. First, if all the input variables are statistically independent and have equal variance,[3] PLS will find the optimal

---

[3] It should be noted that we could insert one more preprocessing step in Table 3 that independently scales all inputs to unit variance. Empirically, however, we did not notice a significant improvement of the algorithm, so we omit this step for simplicity.

Table 2: Legend of Indexes and Symbols Used for LWPR.

| Notation | Affectation |
|---|---|
| $M$ | Number of training data points |
| $N$ | Input dimensionality (i.e., dim. of $\mathbf{x}$) |
| $k = (1 : K)$ | Number of local models |
| $r = (1 : R)$ | Number of local projections used by PLS |
| $\{\mathbf{x}_i, y_i\}_{i=1}^M$ | Training data |
| $\{\mathbf{z}_i\}_{i=1}^M$ | Lower-dimensional projection of input data $\mathbf{x}_i$ (by PLS) |
| $\{z_{i,r}\}_{r=1}^R$ | Elements of projected input $\mathbf{z}_i$ |
| $\mathbf{u}_r$ | $r$th projection direction, i.e., $z_{i,r} = \mathbf{x}_i^T \mathbf{u}_r$ |
| $\mathbf{p}_r$ | Regressed input space to be subtracted to maintain orthogonality of projection directions |
| $\mathbf{X}, \mathbf{Z}$ | Batch representations of input and projected data |
| $w$ | Activation of data $(\mathbf{x}, y)$ on a local model centered at $\mathbf{c}$ |
| $\mathbf{W}$ | Weight matrix $\mathbf{W} \equiv \text{diag}\{w_1, \ldots, w_M\}$ representing the activation due to all $M$ data points |
| $W^n$ | Sum of weights $w$ seen by the local model after $n$ data points |
| $\beta_r$ | $r$th component of slope of the local linear model $\boldsymbol{\beta} \equiv [\beta_1 \cdots \beta_R]^T$ |
| $a_{var,r}^n$ | Sufficient statistics for incremental computation of $r$th dimension of variable var after seeing $n$ data points. |

projection direction $\mathbf{u}_r$ in roughly a single sweep through the training data. The optimal projection direction corresponds to the gradient of the local linearization parameters of the function to be approximated. Second, choosing the projection direction from correlating the input and the output data in step 2b.1 automatically excludes irrelevant input dimensions. And third, there is no danger of numerical problems due to redundant input dimensions as the univariate regressions can easily be prevented from becoming singular.

*3.1.2 Adjusting the Shape and Size of Receptive Field.* The distance metric $\mathbf{D}$ and, hence, the locality of the receptive fields, can be learned for each local model individually by stochastic gradient descent in a penalized leave-one-out cross-validation cost function (Schaal & Atkeson, 1998),

$$J = \frac{1}{\sum_{i=1}^M w_i} \sum_{i=1}^M w_i (y_i - \hat{y}_{i,-i})^2 + \frac{\gamma}{N} \sum_{i,j=1}^N D_{ij}^2, \tag{3.3}$$

where $M$ denotes the number of data points in the training set. The first term of the cost function is the mean leave-one-out cross-validation error of the local model (indicated by the subscript $i, -i$) which ensures proper generalization (Schaal & Atkeson, 1998). The second term, the penalty term, makes sure that receptive fields cannot shrink indefinitely in case of large amounts of training data. Such shrinkage would be statistically correct for

Table 3: Incremental Locally Weighted PLS for One RF Centered at **c**.

---

**1. Initialization:** (# data points seen $n = 0$)

$\quad$ $\mathbf{x}_0^0 = \mathbf{0}$, $\beta_0^0 = 0$, $W^0 = 0$, $\mathbf{u}_r^0 = \mathbf{0}$, $\mathbf{p}_r^0 = \mathbf{0}$; $\quad r = 1 : R$

**2. Incorporating new data:** Given training point $(\mathbf{x}, y)$

$\quad$ **2a. Compute activation and update the means**

$\qquad$ 1. $w = \exp(-\frac{1}{2}(\mathbf{x} - \mathbf{c})^T \mathbf{D}(\mathbf{x} - \mathbf{c}))$; $\quad W^{n+1} = \lambda W^n + w$

$\qquad$ 2. $\mathbf{x}_0^{n+1} = (\lambda W^n \mathbf{x}_0^n + w\mathbf{x})/W^{n+1}$; $\quad \beta_0^{n+1} = (\lambda W^n \beta_0^n + wy)/W^{n+1}$

$\quad$ **2b. Compute the current prediction error**

$\qquad$ $\mathbf{x}_{res,1} = \mathbf{x} - \mathbf{x}_0^{n+1}$, $\hat{y} = \beta_0^{n+1}$

$\qquad$ Repeat for $r = 1 : R$ (# projections)

$\qquad$ 1. $z_r = \mathbf{x}_{res,r}^T \mathbf{u}_r^n / \sqrt{\mathbf{u}_r^{n^T} \mathbf{u}_r^n}$

$\qquad$ 2. $\hat{y} \leftarrow \hat{y} + \beta_r^n z_r$

$\qquad$ 3. $\mathbf{x}_{res,r+1} = \mathbf{x}_{res,r} - z_r \mathbf{p}_r^n$

$\qquad$ 4. $MSE_r^{n+1} = \lambda MSE_r^n + w\,(y - \hat{y})^2$

$\qquad$ $e_{cv} = y - \hat{y}$

$\quad$ **2c. Update the local model**

$\qquad$ $res_1 = y - \beta_0^{n+1}$

$\qquad$ For $r = 1 : R$ (# projections)

$\qquad$ **2c.1** Update the local regression and compute residuals

$\qquad\quad$ 1. $a_{zz,r}^{n+1} = \lambda\, a_{zz,r}^n + w\, z_r^2$; $\quad a_{zres,r}^{n+1} = \lambda\, a_{zres,r}^n + w\, z_r\, res_r$

$\qquad\quad$ 2. $\beta_r^{n+1} = a_{zres,r}^{n+1}/a_{zz,r}^{n+1}$

$\qquad\quad$ 3. $res_{r+1} = res_r - z_r \beta_r^{n+1}$

$\qquad\quad$ 4. $\mathbf{a}_{xz,r}^{n+1} = \lambda\, \mathbf{a}_{xz,r}^n + w\mathbf{x}_{res,r} z_r$

$\qquad$ **2c.2** Update the projection directions

$\qquad\quad$ 1. $\mathbf{u}_r^{n+1} = \lambda\, \mathbf{u}_r^n + w\mathbf{x}_{res,r}\, res_r$

$\qquad\quad$ 2. $\mathbf{p}_r^{n+1} = \mathbf{a}_{xz,r}^{n+1}/a_{zz,r}^{n+1}$

$\qquad$ $e = res_{r+1}$

**3. Predicting with novel data ($\mathbf{x}_q$):** Initialize: $y_q = \beta_0$, $\mathbf{x}_q = \mathbf{x}_q - \mathbf{x}_0$

$\quad$ Repeat for $r = 1 : R$

$\quad$ • $y_q \leftarrow y_q + \beta_r s_r$ where $s_r = \mathbf{u}_r^T \mathbf{x}_q$

$\quad$ • $\mathbf{x}_q \leftarrow \mathbf{x}_q - s_r \mathbf{p}_r^n$

---

Note: The subscript $k$ referring to the $k$th local model is omitted throughout this table since we are referring to updates in one local model or RF.

asymptotically unbiased function approximation, but it would require maintaining an ever increasing number of local models in the learning system, which is computationally too expensive. The trade-off parameter $\gamma$ can be determined either empirically or from assessments of the maximal local curvature of the function to be approximated (Schaal & Atkeson, 1997); in general, results are not very sensitive to this parameter (Schaal & Atkeson, 1998), as it primarily affects resource efficiency—when input and output data are preprocessed to have unit variance, $\gamma$ can be kept constant, for example, at $\gamma = 1e - 7$ as in all our experiments. It should be noted that due to

the local cost function in equation 3.3, learning becomes entirely localized too; no parameters from other local models are needed for updates as, for instance, in competitive learning with mixture models. Moreover, minimizing equation 3.3 can be accomplished in an incremental way without keeping data in memory (Schaal & Atkeson, 1998). This property is due to a reformulation of the leave-one-out cross-validation error as the PRESS residual error (Belsley, Kuh, & Welsch, 1980). As detailed in Schaal and Atkeson (1998) the bias-variance trade-off is thus resolved for every local model individually such that an increasing number of local models will not lead to overfitting. Indeed, it leads to better approximation results due to model averaging (see equation 3.2) in the sense of committee machines (Perrone & Cooper, 1993).

In ordinary weighted linear regression, expanding equation 3.3 with the PRESS residual error results in

$$
J = \frac{1}{\sum_{i=1}^{M} w_i} \sum_{i=1}^{M} \frac{w_i (y_i - \hat{y}_i)^2}{\left(1 - w_i \mathbf{x}_i^T \mathbf{P} \mathbf{x}_i\right)^2} + \frac{\gamma}{N} \sum_{i,j=1}^{N} D_{ij}^2, \tag{3.4}
$$

where $\mathbf{P}$ corresponds to the inverted weighted covariance matrix of the input data. Interestingly, we found that the PRESS residuals of equation 3.4 can be exactly formulated in terms of the PLS projected inputs $\mathbf{z}_i \equiv [z_{i,1} \dots z_{i,R}]^T$ (cf. Table 3) as

$$
J = \frac{1}{\sum_{i=1}^{M} w_i} \sum_{i=1}^{M} \frac{w_i (y_i - \hat{y}_i)^2}{\left(1 - w_i \mathbf{z}_i^T \mathbf{P}_z \mathbf{z}_i\right)^2} + \frac{\gamma}{N} \sum_{i,j=1}^{N} D_{ij}^2
$$

$$
\equiv \frac{1}{\sum_{i=1}^{M} w_i} \sum_{i=1}^{M} J_1 + \frac{\gamma}{N} J_2, \tag{3.5}
$$

where $\mathbf{P}_z$ corresponds to the inverse covariance matrix computed from the projected inputs $\mathbf{z}_i$ for $R = N$, that is, the $\mathbf{z}_i$'s spans the same full-rank input space[4] as $\mathbf{x}_i$'s in equation 3.4 (cf. the proof in appendix A). It can also been proved, as explained in appendix A, that $\mathbf{P}_z$ is diagonal, which greatly contributes to the computational efficiency of our update rules. Based on this cost function, the distance metric in LWPR is learned by gradient descent,

$$
\mathbf{M}^{n+1} = \mathbf{M}^n - \alpha \frac{\partial J}{\partial \mathbf{M}} \quad \text{where } \mathbf{D} = \mathbf{M}^T \mathbf{M} \text{ (for positive definiteness)} \tag{3.6}
$$

where $\mathbf{M}$ is an upper triangular matrix resulting from a Cholesky decomposition of $\mathbf{D}$. Following Schaal and Atkeson (1998), a stochastic approximation of the gradient $\frac{\partial J}{\partial \mathbf{M}}$ of equation 3.5 can be derived by keeping track

---

[4] For rank-deficient input spaces, the equivalence of equations 3.4 and 3.5 holds in the subspace spanned by $X$.

Table 4: Derivatives for Distance Metric Update.

---

*For the current data point* $\mathbf{x}$, *its PLS projection* $\mathbf{z}$ *and activation* $w$:

$$\frac{\partial J}{\partial \mathbf{M}} \approx \left( \sum_{i=1}^{M} \frac{\partial J_1}{\partial w} \right) \frac{\partial w}{\partial \mathbf{M}} + \frac{w}{W^{n+1}} \frac{\partial J_2}{\partial \mathbf{M}} \text{ (stochastic update of equation 3.5)}$$

$$\frac{\partial w}{\partial M_{kl}} = -\frac{1}{2} w (\mathbf{x} - \mathbf{c})^T \frac{\partial \mathbf{D}}{\partial M_{kl}} (\mathbf{x} - \mathbf{c}); \quad \frac{\partial J_2}{\partial M_{kl}} = 2 \frac{\gamma}{N} \sum_{i,j=1}^{N} D_{ij} \frac{\partial D_{ij}}{\partial M_{kl}}$$

$$\frac{\partial D_{ij}}{\partial M_{kl}} = M_{kj} \delta_{il} + M_{ki} \delta_{jl}; \text{ where } \delta_{ij} = 1 \text{ if } i = j \text{ else } \delta_{ij} = 0.$$

$$\sum_{i=1}^{M} \frac{\partial J_1}{\partial w} = \frac{e_{cv}^2}{W^{n+1}} - \frac{2\,e}{W^{n+1}} \mathbf{q}^T \mathbf{a}_H^n - \frac{2}{W^{n+1}} \mathbf{q}^{2^T} \mathbf{a}_G^n - \frac{a_E^{n+1}}{(W^{n+1})^2}$$

$$\text{where } \mathbf{z} = \begin{bmatrix} z_1 \\ \vdots \\ z_R \end{bmatrix}, \; \mathbf{z}^2 = \begin{bmatrix} z_1^2 \\ \vdots \\ z_R^2 \end{bmatrix}, \; \mathbf{q} = \begin{bmatrix} z_1/a_{zz,1}^{n+1} \\ \vdots \\ z_R/a_{zz,R}^{n+1} \end{bmatrix}, \; \mathbf{q}^2 = \begin{bmatrix} q_1^2 \\ \vdots \\ q_k^2 \end{bmatrix},$$

$$\mathbf{a}_H^{n+1} = \lambda \mathbf{a}_H^n + \frac{w\,e_{cv}\mathbf{z}}{(1-h)}; \; \mathbf{a}_G^{n+1} = \lambda \mathbf{a}_G^n + \frac{w^2 e_{cv}^2 \mathbf{z}^2}{(1-h)} \text{ where } h = w\mathbf{z}^T \mathbf{q}$$

$$a_E^{n+1} = \lambda a_E^n + w e_{cv}^2$$

---

Note: Refer to Table 3 for some of the variables.

of several sufficient statistics, as shown in Table 4. It should be noted that in these update laws, we treated the PLS projection direction, and hence, $\mathbf{z}$, as if it were independent of the distance metric, such that chain rules need not be taken throughout the entire PLS recursions. Empirically, this simplification did not seem to have any negative impact and reduced the update rules significantly.

**3.2 The Complete LWPR Algorithm.** All update rules can be combined in an incremental learning scheme that automatically allocates new locally linear models as needed. The concept of the final learning network is illustrated in Figure 1, and an outline of the final LWPR algorithm is shown in Table 5.

In this pseudocode, $w_{gen}$ is a threshold that determines when to create a new receptive field, as discussed in Schaal and Atkeson (1998). $w_{gen}$ is a computational efficiency parameter and not a complexity parameter as in mixture models. The closer $w_{gen}$ is set to 1, the more overlap local models will have, which is beneficial in the spirit of committee machines (cf. Schaal & Atkeson, 1998; Perrone & Cooper 1993) but more costly to compute. In general, the more overlap is permitted, the better the function-fitting results, without any danger that the increase in overlap can lead to overfitting. $\mathbf{D}_{def}$ is the initial (usually diagonal) distance metric in equation 3.1. The initial number of projections is set to $R = 2$. The algorithm has a simple

Table 5: Pseudocode of the Complete LWPR Algorithm.

---

- Initialize the LWPR with no receptive field (RF).
- For every new training sample $(\mathbf{x}, y)$:
  - For k = 1 to $K$ (number of receptive fields):
    * Calculate the activation from equation 3.1
    * Update projections and regression (see Table 3) and distance metric (see Table 4)
    * Check if number of projections needs to be increased (cf. section 3.2)
  - If no RF was activated by more than $w_{gen}$;
    * Create a new RF with $R = 2$, $\mathbf{c} = \mathbf{x}$, $\mathbf{D} = \mathbf{D}_{def}$

---

mechanism of determining whether $R$ should be increased by recursively keeping track of the mean-squared error (MSE) as a function of the number of projections included in a local model—step 2b.4 in Table 3. If the MSE at the next projection does not decrease more than a certain percentage of the previous MSE, $\frac{MSE_{r+1}}{MSE_r} > \phi$, where $\phi \in [0, 1]$, the algorithm will stop adding new projections locally. As $MSE_r$ can be interpreted as an approximation of the leave-one-out cross-validation error of each projection, this threshold criterion avoids problems due to overfitting. Due to the need to compare the $MSE$ of two successive projections, LWPR needs to be initialized with at least two projection dimensions. A comparison of these mechanisms of constructive learning with previous algorithms in the literature (e.g, Platt, 1991) can be found in Schaal and Atkeson (1998).

*3.2.1 Speed-Up for Learning from Trajectories.* If in incremental learning, training data are generated from trajectories (i.e., data are temporally correlated), it is possible to accelerate lookup and training times by taking advantage of the fact that two consecutively arriving training points are close neighbors in input space. For such cases, we added a special data structure to LWPR that allows restricting updates and lookups to only a small fraction of local models instead of exhaustively sweeping through all of them. For this purpose, each local model maintains a list of all other local models that overlap sufficiently with it. Sufficient overlap between two models $i$ and $j$ can be determined from the centers and distance metrics. The point $\mathbf{x}$ in input space that is the closest to both centers in the sense of a Mahalanobis distance is $\mathbf{x} = (\mathbf{D}_i + \mathbf{D}_j)^{-1}(\mathbf{D}_i \mathbf{c}_i + \mathbf{D}_j \mathbf{c}_j)$. Inserting this point into equation 3.1 of one of the local models gives the activation $w$ due to this point. The two local models are listed as sufficiently overlapping if $w \geq w_{gen}$ (cf. Table 5). For diagonal distance metrics, the overlap computation is linear in the number of inputs. Whenever a new data point is added to LWPR, one neighborhood relation is checked for the maximally activated RF. An appropriate counter for each local model ensures that overlap with all other local models is checked exhaustively. Given this nearest-neighbor data structure, lookup and learning can be confined to only a few RFs. For every lookup (update), the identification number of the maximally activated RF is returned.

The next lookup (update) will consider only the neighbors of this RF. It can be shown that this method is as good as an exhaustive lookup (update) strategy that excludes RFs that are activated below a certain threshold $w_{cutoff}$.

3.2.2 *Pruning of Local Models.* As in the RFWR algorithm (Schaal & Atkeson, 1998), it is possible to prune local models depending on the level of overlap between two local models or the accumulated locally weighted mean-squared error. The pruning strategy is virtually identical to that in (Schaal & Atkeson, 1998, sec. 3.14). However, due to the numerical robustness of PLS, we have noticed that the need for pruning or merging is almost nonexistent in the LWPR implementation, such that we do not expand on this possible feature of the algorithm.

3.2.3 *Computational Complexity.* For a diagonal distance metric **D** and under the assumption that the number of projections $R$ remains small and bounded, the computational complexity of one incremental update of all parameters of LWPR is linear in the number of input dimensions $N$. To the best of our knowledge, this property makes LWPR one of the computationally most efficient algorithms that have been suggested for high-dimensional function approximation. This low-computational complexity sets LWPR apart from our earlier work on the RFWR algorithm (Schaal & Atkeson, 1998), which was cubic in the number of input dimensions. We thus accomplished one of our main goals: maintaining the appealing function approximation properties of RFWR while eliminating its problems in high-dimensional learning problems.

3.2.4 *Confidence Intervals.* Under the classical probabilistic interpretation of weighted least squares (Gelman, Carlin, Stern, & Rubin, 1995), that each local model's conditional distribution is normal with heteroscedastic variances $p(y|\mathbf{x}; w_k) \sim N(\mathbf{z}_k^T \beta_k, s_k^2/w_k)$, it is possible to derive the predictive variances $\sigma_{pred,k}^2$ for a new query point $\mathbf{x}_q$ for each local model in LWPR.[5] The derivation of this measure is in analogy with ordinary linear regression (Schaal & Atkeson, 1994; Myers, 1990) and is also consistent with the Bayesian formulation of predictive variances (Gelman et al., 1995). For each individual local model, $\sigma_{pred,k}^2$ can be estimated as (refer to Tables 4 and 3 for variable definitions):

$$\sigma_{pred,k}^2 = s_k^2 \left(1 + w_k \mathbf{z}_{q,k}^T \mathbf{q}_k\right),  \tag{3.7}$$

---

[5] Note that $w_k$ is used here as an abbreviated version of $w_{\{q,k\}}$—the weight contribution due to query point **q** in model $k$—for simplicity.

where $\mathbf{z}_{q,k}$ is the projected query point $\mathbf{x}_q$ under the $k$th local model, and

$$s_k^2 \approx MSE_{k,R}^{n=M}/(M_k' - p_k'); \quad M_k' \equiv \sum_{i=1}^{M} w_{k,i} \approx W_k^{n=M}$$

$$p_k' \equiv \sum_{i=1}^{M} w_{k,i}^2 \mathbf{z}_{k,i}^T \mathbf{q}_{k,i} \approx a_{p_k'}^{n=M} \text{ with incremental update of}$$

$$a_{p_k'}^{n+1} = \lambda a_{p_k'}^n + w_k^2 \mathbf{z}_k^T \mathbf{q}_k.$$

The definition of $M'$ in terms of the sum of weights reflects the effective number of data points entering the computation of the local variance $s_k^2$ (Schaal & Atkeson, 1994) after an update of $M$ training points has been performed. The definition of $p'$, also referred to as the local degrees of freedom, is analogous to the global degrees of freedom of linear smoothers (Hastie & Tibshirani, 1990; Schaal & Atkeson, 1994).

In order to obtain a predictive variance measure for the averaging formula (equation 3.2), one could just compute the weighted average of the predictive variance in equation 3.7. While this approach is viable, it nevertheless ignores important information that can be obtained from variance across the individual predictions $\hat{y}_{q,k}$ and is thus potentially too optimistic. To remedy this issue, we postulate that from the view of combining individual $\hat{y}_{q,k}$, each contributing $y_{q,k}$ was generated from the process

$$y_{q,k} = y_q + \epsilon_1 + \epsilon_{2,k},$$

where we assume two separate noise processes: (1) one whose variance $\sigma^2$ is independent of the local model, that is, $\epsilon_1 \sim N(0, \sigma^2/w_k)$ (and accounts for the differences between the predictions of the local models) and (2) another, which is the noise process $\epsilon_{2,k} \sim N(0, \sigma_{pred,k}^2/w_k)$ of the individual local models. It can be shown (see appendix B) that equation 3.2 is a consistent way of combining prediction from multiple models under the noise model we just described and that the combined predictive variance over all models can be approximated as

$$\sigma_{pred}^2 = \frac{\sum_k w_k \, \sigma^2}{(\sum_k w_k)^2} + \frac{\sum_k w_k \, \sigma_{pred,k}^2}{(\sum_k w_k)^2}. \tag{3.8}$$

The estimate of $\sigma_{pred,k}$ is given in equation 3.7. The global variance across models can be approximated as $\sigma^2 = \sum_k w_k (\hat{y}_q - \hat{y}_{k,q})^2 / \sum_k w_k$. Inserting these values in equation 3.8, we obtain:

$$\sigma_{pred}^2 = \frac{1}{(\sum_k w_k)^2} \sum_{k=1}^{K} w_k \big[ (\hat{y}_q - \hat{y}_{k,q})^2 + s_k^2 \big( 1 + w_k \mathbf{z}_k^T \mathbf{q}_k \big) \big]. \tag{3.9}$$
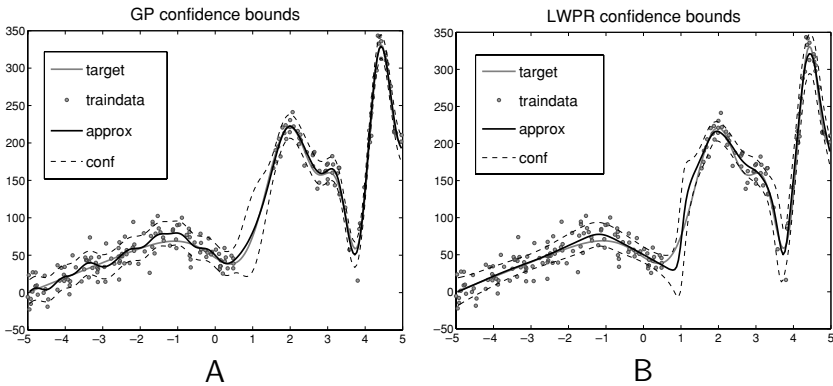
Figure 2: Function approximation with 200 noisy data points along with plots of confidence intervals for (A) gaussian process regression and (B) LWPR algorithms. Note the absence of data in the range [0.5 1.5].

A one-standard-deviation-based confidence interval would thus be

$$I_c = \hat{y}_q \pm \sigma_{pred}. \tag{3.10}$$

The variance estimate in equation 3.8 is consistent with the intuitive requirement that when only one local model contributes to the prediction, the variance is entirely attributed to the predictive variance of that single model. Moreover, a query point that does not receive a high weight from any local model will have a large confidence interval due to the small squared sum-of-weight value in the denominator. Figure 2 illustrates comparisons of confidence interval plots on a toy problem with 200 noisy data points. Data from the range [0.5 1.5] were excluded from the training set. Both gaussian process regression and LWPR show qualitatively similar confidence interval bounds and fitting results.

## 4 Empirical Evaluation

The following sections provide an evaluation of our proposed LWPR learning algorithm over a range of artificial and real-world data sets. Whenever useful and feasible, comparisons to state-of-the-art alternative learning algorithms are provided, in particular, support vector regression (SVM) and gaussian process regression (GP). SVMR and GPR were chosen due to their generally acknowledged excellent performance in nonlinear regression on finite data sets. However, it should be noted that both SVM and GP are batch learning systems, while LWPR was implemented as a fully incremental algorithm, as described in the previous sections.
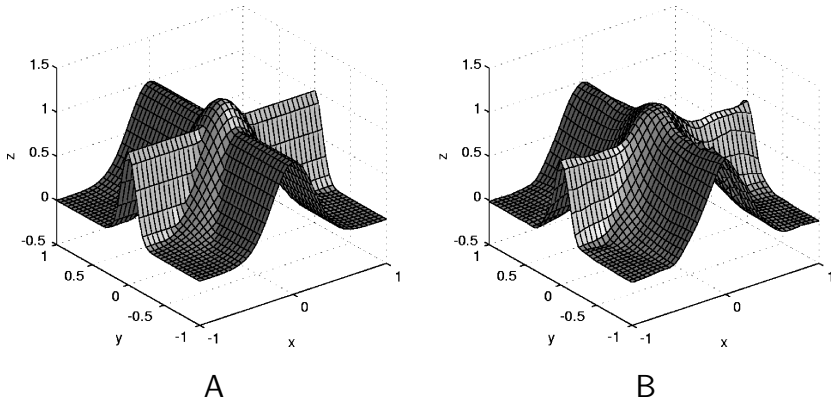
Figure 3: (A) Target and (B) learned nonlinear cross function.

**4.1 Function Approximation with Redundant and Irrelevant Data.** We implemented LWPR algorithm as outlined in section 3. In each local model, the projection regressions are performed by (locally weighted) PLS, and the distance metric **D** is learned by stochastic incremental cross validation; all learning methods employed second-order learning techniques; incremental PLS uses recursive least squares, and gradient descent in the distance metric was accelerated as described in Schaal and Atkeson (1998). In all our evaluations, an initial (diagonal) distance metric of $\mathbf{D}_{def} = 30\mathbf{I}$ was chosen, the activation threshold for adding local models was $w_{gen} = 0.2$, and the threshold for adding new projections was $\phi = 0.9$ (cf. section 3.2).

As a first test, we ran LWPR on 500 noisy training data drawn from the two-dimensional function (Cross 2D) generated from $y = \max\{\exp(-10x_1^2), \exp(-50x_2^2, 1.25\exp(-5(x_1^2 + x_2^2)))\} + \mathrm{N}(0, 0.01)$, as shown in Figure 3A. This function has a mixture of areas of rather high and rather low curvature and is an interesting test of the learning and generalization capabilities of a learning algorithm: learning models with low complexity find it hard to capture the nonlinearities accurately, while more complex models easily overfit, especially in linear regions. A second test added eight constant (i.e., redundant) dimensions to the inputs and rotated this new input space by a random 10-dimensional rotation matrix to create a 10-dimensional input space with high-rank deficiency (Cross 10D). A third test added another 10 (irrelevant) input dimensions to the inputs of the second test, each having $N(0, 0.05^2)$ gaussian noise, thus obtaining a data set with 20-dimensional input space (Cross 20D). Typical learning curves with these data sets are illustrated in Figure 4. In all three cases, LWPR reduced the normalized mean squared error (thick lines) on a noiseless test set (1681 points on a $41 \times 41$ grid in the unit-square in input space) rapidly in 10 to 20 epochs of training to less than $nMSE = 0.05$, and it converged to the
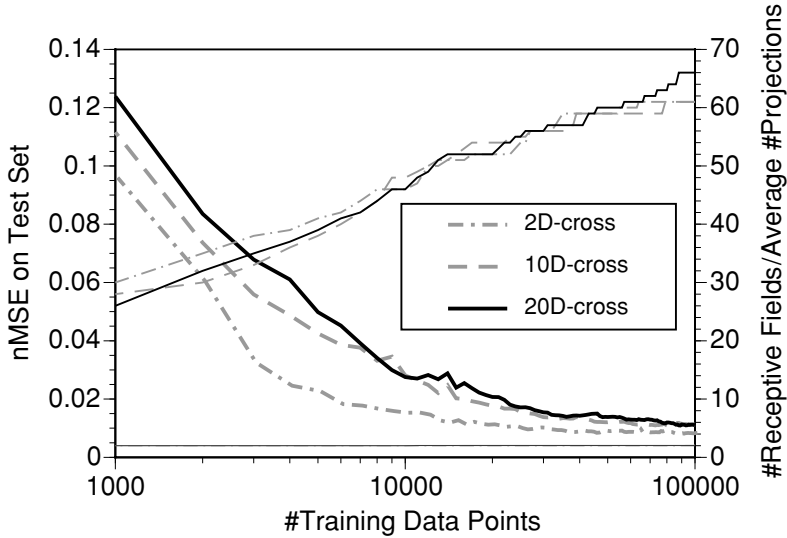
Figure 4: Learning curves for 2D, 10D, and 20D data for cross approximation.

excellent function approximation result of $nMSE = 0.015$ after 100,000 data presentations or 200 epochs.[6] Figure 5 shows the adapted distance metric, while Figure 3B illustrates the reconstruction of the original function from the 20-dimensional test data, visualized in 3D, a highly accurate approximation. The rising thin lines in Figure 4 show the number of local models that LWPR allocated during learning. The very thin lines at the bottom of the graph indicate the average number of projections that the local models allocated: the average settled at a value of around two local projections, as is appropriate for this originally two-dimensional data set. This set of tests demonstrates that LWPR is able to recover a low-dimensional nonlinear function embedded in high-dimensional space despite irrelevant and redundant dimensions and that the data efficiency of the algorithm does not degrade in higher-dimensional input spaces. The computational complexity of the algorithm increased only linearly with the number of input dimensions, as explained in section 3.

The results of this evaluations can be directly compared with our earlier work on the RFWR algorithm (Schaal & Atkeson, 1998), in particular Figures 4 and 5 of this earlier article. The learning speed and the number of allocated local models for LWPR is essentially the same as for RFWR in the 2D test set. Applying RFWR to the 10- and 20-dimensional data set

---

[6] Since LWPR is an incremental algorithm, data presentations in this case refer to repeated random-order presentations of training data from our noisy data set of size 500.
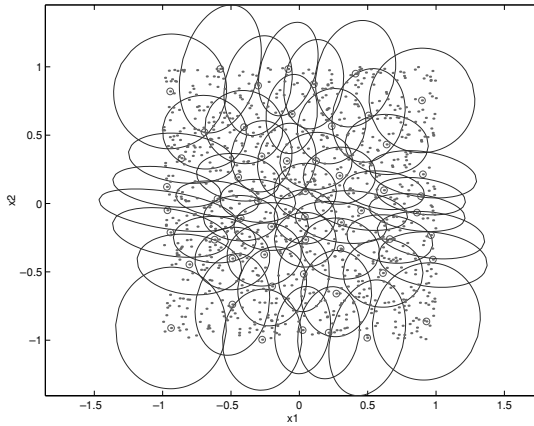
Figure 5: The automatically tuned distance metric for the cross approximation.

of this article, however, is problematic, as it requires a careful selection of initial ridge regression parameters to stabilize the highly rank-deficient full covariance matrix of the input data, and it is easy to create too much bias or too little numerical stabilization initially, which can trap the local distance metric adaptation in local minima. While the LWPR algorithm just computes about a factor 10 times longer for the 20D experiment in comparison to the 2D experiment, RFWR requires a 1000-fold increase of computation time, thus rendering this algorithm unsuitable for high-dimensional regression. In order to compare LWPR's results to other popular regression methods, we evaluated the 2D, 10D, and 20D cross data sets with gaussian process regression (GP) and support vector (SVM) regression in addition to our LWPR method. It should be noted that neither SVM nor GP methods is an incremental method, although they can be considered state-of-the-art for batch regression under relatively small numbers of training data and reasonable input dimensionality. The computational complexity of these methods is prohibitively high for real-time applications. The GP algorithm (Gibbs & MacKay, 1997) used a generic covariance function and optimized over the hyperparameters. The SVM regression was performed using a standard available package (Saunders et al., 1998) and optimized for kernel choices.

Figure 6 compares the performance of LWPR and gaussian processes for the above-mentioned data sets using 100, 300, and 500 training data points.[7] As in Figure 3 the test data set consisted of 1681 data points corresponding to the vertices of a $41 \times 41$ grid over the unit square; the corresponding output values were the exact function values. The approximation error was

---

[7] We have not plotted the results for SVM regression since it was found to consistently perform worse than GP regression for the given number of training data.
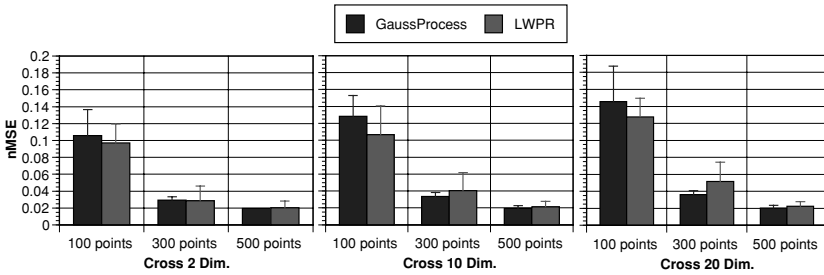
Figure 6: Normalized mean squared error comparisons between LWPR and gaussian processes for 2D, 10D, and 20D Cross data sets.

measured as a normalized weighted mean squared error, *nMSE*, that is, the weighted *MSE* on the test set normalized by the variance of the outputs of the test set. The weights were chosen as $1/\sigma_{pred,i}^2$ for each test point $\mathbf{x}_i$. Using such a weighted *nMSE* was useful to allow the algorithms to incorporate their confidence in the prediction of a query point, which is especially useful for training data sets with few data points where query points often lie far away from any training data and require strong extrapolation to form a prediction. Multiple runs on 10 randomly chosen training data sets were performed to accumulate the statistics.

As can be seen from Figure 6, the performance differences of LWPR and GP were largely statistically insignificant across training data sizes and input dimensionality. LWPR had a tendency to perform slightly better on the 100-point data sets, most likely due to its quickly decreasing confidence when significant extrapolation is required for a test point. For the 300-point data sets, GP had a minor advantage and less variance in its predictions, while for 500-point data sets, both algorithms achieved equivalent results. While GPs used all the input dimensions for predicting the output (deduced from the final converged coefficients of the covariance matrix), LWPR stopped at an average of two local projections, reflecting that it exploited the low-dimensional distribution of the data. Thus, this comparison illustrates that LWPR is a highly competitive learning algorithm in terms of its generalization capabilities and accuracy of results, in spite of its' being a truly incremental, computationally efficient and real-time implementable algorithm.

**4.2 Comparisons on Benchmark Regression Data Sets.** While LWPR is specifically geared toward real-time incremental learning in high dimensions, it can nevertheless also be employed for traditional batch data analysis. Here we compare its performance on two natural real-world benchmark data sets, again using gaussian processes and support vector regression as competitors.

Table 6: Comparison of Normalized Mean Squared Errors on Boston and Abalone Data Sets.

|         | Gaussian Process     | Support Vectors  | LWPR              |
| ------- | -------------------- | ---------------- | ----------------- |
| Boston  | $0.0806 \pm 0.0195$  | $0.1115 \pm 0.09$ | $0.0846 \pm 0.0225$ |
| Abalone | $0.4440 \pm 0.0209$  | $0.4830 \pm 0.03$ | $0.4056 \pm 0.0131$ |

The data sets we used were the Boston Housing data and the Abalone data set, both available from the UCI Machine Learning Repository (Hettich & Bay 1999). The Boston Housing data, which had 14 attributes, was split randomly (10 random splits) into disjoint sets of 404 training and 102 testing data. The Abalone data set, which had 9 attributes, was downsampled to yield 10 disjoint sets of 500 training data points and 1177 testing points.[8]

The GP used hyperparameter estimation for the open parameters of the covariance matrix, while for SVM regression, the results were obtained by employing a gaussian kernel of width 3.9 and 10 for the Boston and Abalone data sets, respectively, based on the optimized values suggested in Schölkopf, Smola, Williamson, & Bartlett (2000). Table 6 shows the comparisons of the normalized mean squared error (nMSE) achieved by GP, SVM, and LWPR on both data sets. Once again, LWPR was highly competitive on these real-world data sets, consistently outperforming SVM regression and achieving very similar *nMSE* results as GP regression.

**4.3 Sensorimotor Learning in High-Dimensional Space.** In this section, we look at the application of LWPR to real-time learning in high-dimensional spaces in a data-rich environment, an example of which is learning for robot control. In such domains, LWPR is (to the best of our knowledge) one of the only viable and practical options for principled statistical learning. The goal of learning in this evaluation is to estimate the inverse dynamics model (also referred to as an *internal model*) of the robotic system such that it can be used as a component of a feedforward controller for executing fast, accurate movements. The inverse dynamics model is a mapping from joint position, joint velocity, and joint acceleration to joint torques, a function with three times the number of degrees of freedom (DOF) as input dimensionality.

We implemented LWPR on the real-time operating system (vx-Works) for the Sarcos humanoid robot in Figure 7A, a 30 DOF system, which used its right hand to draw a lying figure 8 pattern. Out of the four parallel

---

[8] The gaussian process algorithm had problems of convergence and numerical stability for training data sizes above 500 points. However, a more comprehensive computation can be carried out by using techniques from Williams and Seeger (2001) to scale up the GP results, as pointed out by one of the reviewers.
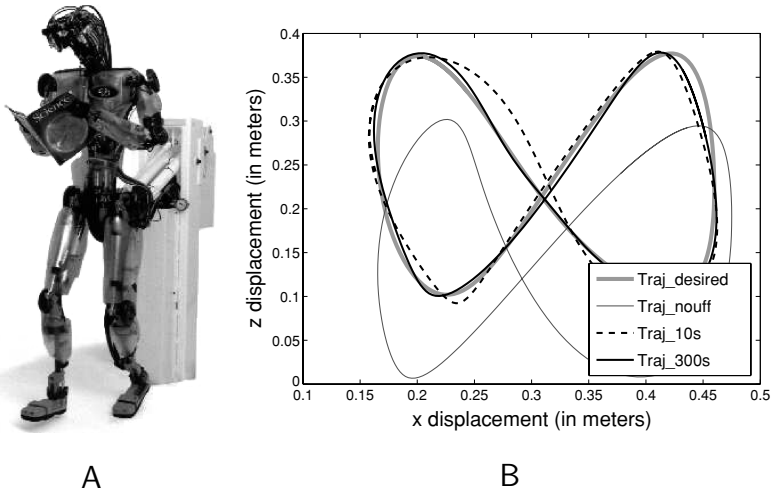
Figure 7: (A) The 30-DOF SARCOS humanoid robot. (B) Results of online learning of the inverse dynamics with LWPR on the humanoid robot.

processors of the system, one 366 Mhz PowerPC processor was completely devoted to lookup and learning with LWPR. In order to accelerate lookup and training times, the nearest-neighbor data lookup described in section 3.2.1 was used. Learning of the inverse dynamics model required learning in a 90-dimensional input space and the outputs were the 30 torque commands for each of the DOFs. Ideally, we would learn one individual LWPR model for each of the 30 output dimensions. However, as the learning of 30 parallel LWPR models would have exceeded the computational power of our 366 Mhz real-time processors, we chose to learn one single LWPR model with a 30-dimensional output vector: each projection of PLS in LWPR regressed all outputs versus the projected input data. The projection direction was chosen as the mean projection across all outputs at each projection stage of PLS. This approach is suboptimal, as it is quite unlikely that all output dimensions agree on one good projection direction; essentially, one assumes that the gradients of all outputs point roughly into the same direction. On the other hand, as D'Souza et al. (2001) demonstrated that movement data of actual physical movement systems lie on locally low-dimensional distributions, one can hope that LWPR with multiple outputs can still work successfully by simply spanning this locally low-dimensional input space with all projections.

The LWPR model was trained online while the robot performed a pseudo–randomly drifting figure 8 pattern in front of its body. Lookup proceeded at 480 Hz, while updating the learning model was achieved at about 70 Hz. After 10 seconds of training, learning was stopped, and the robot

attempted to draw a planar figure 8 in the $x - z$ plane of the robot end-effector at 2 Hz frequency for the entire pattern. The same test pattern was also performed after 300 seconds of training. Figure 7B demonstrates the result of learning. In this figure, $\text{Traj}_{desired}$ denotes the desired figure 8 pattern; $\text{Traj}_{10}$ is the LWPR learning result after 10 seconds of training; and $\text{Traj}_{300}$ is the result after 300 seconds of training. The $\text{Traj}_{nouff}$ trace demonstrates the figure 8 patterns performed without any inverse dynamics model, just using a low-gain negative feedback (proportional-derivative (PD)) controller. LWPR rapidly improves over a control system with no inverse dynamics controller; within 10 seconds of movement, the most significant inertial and gravity perturbation have been compensated. Convergence to low error tracking of the figure 8 takes slightly longer—about 300 seconds ($\text{Traj}_{300}$ in Figure 7B—but is reliably achieved. About 50 local models were created for this task. While tracking performance is not perfect, the learned inverse dynamics outperformed the model estimated by rigid body dynamics methods (An, Atkeson, & Hollerbach, 1988) significantly in terms of its average tracking error of the desired trajectory. This rigid dynamics model was estimated from about 1 hour of data collection and 30 minutes off-line processing of the data. These results are the first that demonstrate an actual implementation of real-time inverse dynamics learning on such a robot of this complexity.

*4.3.1 Online Learning for Autonomous Airplane Control.* The online learning abilities of LWPR are ideally suited to be incorporated in algorithms of provably stable adaptive control. The control theoretic development of such an approach was presented in Nakanishi, Farrell, and Schaal (2004). In essence, the problem formulation begins with a specific class of equations of motion of the form

$$\dot{\mathbf{x}} = f(\mathbf{x}) + g(\mathbf{x})\mathbf{u}, \tag{4.1}$$

where $\mathbf{x}$ denotes the state of the control system, the control inputs, and $f(\mathbf{x})$ and $g(\mathbf{x})$ are nonlinear function to approximated. A suitable control law for such a system is

$$\mathbf{u} = \hat{g}(\mathbf{x})^{-1}\left(-\hat{f}(\mathbf{x}) + \dot{\mathbf{x}}_c + \mathbf{K}(\mathbf{x}_c - \mathbf{x})\right), \tag{4.2}$$

where $\mathbf{x}_c, \dot{\mathbf{x}}_c$ are a desired reference trajectory to be tracked, and the "hat" notation indicates that these are the approximated version of the unknown function.

We applied LWPR in this control framework to learn the unknown function $f$ and $g$ for the problem of autonomous airplane control on a high-fidelity simulator. For simplicity, we considered only a planar version of the

airplane, governed by the differential equation (Stevens & Lewis, 2003):

$$\dot{V} = \frac{1}{m}(T\cos\alpha - D) - g\sin\gamma$$

$$\dot{\alpha} = -\frac{1}{mV}(L + T\sin\alpha) + \frac{g\cos\gamma}{V} + Q \qquad (4.3)$$

$$\dot{Q} = cM.$$

In these equations, $V$ denotes the forward speed of the airplane, $m$ the mass, $T$ the thrust, $\alpha$ the angle of attack, $g$ the gravity constant, $\gamma$ the flight path angle with regard to the horizontal world coordinate system axis, $Q$ the pitch rate, and $c$ an inertial constant. The complexity of these equations is hidden in $D$, $L$, and $M$, which are the unknown highly nonlinear aerodynamic lift force, drag force, and pitch moment terms, which are specific to every airplane.

While we will not go into the detail of provably stable adaptive control with LWPR in this letter and how the control law 4.2 is applied for airplane control, from the viewpoint of learning, the main components to learn are the lift and drag forces, and the pitch moment. These can be obtained by rearranging equation 4.3 to:

$$D = T\cos\alpha - (\dot{V} + g\sin\gamma)m$$
$$= f_D(\alpha, Q, V, M, \gamma, \delta_{OFL}, \delta_{OFR}, \delta_{MFL}, \delta_{MFR}, \delta_{SPL}, \delta_{SPR})$$
$$L = \left(\frac{g\cos\gamma}{V} + Q - \dot{\alpha}\right)mV - T\sin\alpha$$
$$= f_L(\alpha, Q, V, M, \gamma, \delta_{OFL}, \delta_{OFR}, \delta_{MFL}, \delta_{MFR}, \delta_{SPL}, \delta_{SPR})$$
$$M = \frac{Q}{c} = f_M(\alpha, Q, V, M, \gamma, \delta_{OFL}, \delta_{OFR}, \delta_{MFL}, \delta_{MFR}, \delta_{SPL}, \delta_{SPR}).$$

$$(4.4)$$

The $\delta$ terms denote the control surface angles of the airplane, with indices Midboard-Flap-Left/Right (MFL, MFR), Outboard-Flap-Left/Right (OFL, OFR), and left and right spoilers (SPL, SPR). All terms on the right-hand side of equation 4.4 are known, such that we have to cope with three simultaneous function approximation problems in an 11-dimensional input space, an ideal application for LWPR.

We implemented LWPR for the three functions above in a high-fidelity simulink simulation of an autonomous airplane using the adaptive control approach of Nakanishi et al. (2004). The airplane started with no initial knowledge, just the proportional controller term in equation 4.2 (the term multiplied by **K**). The task of the controller was to fly doublets—
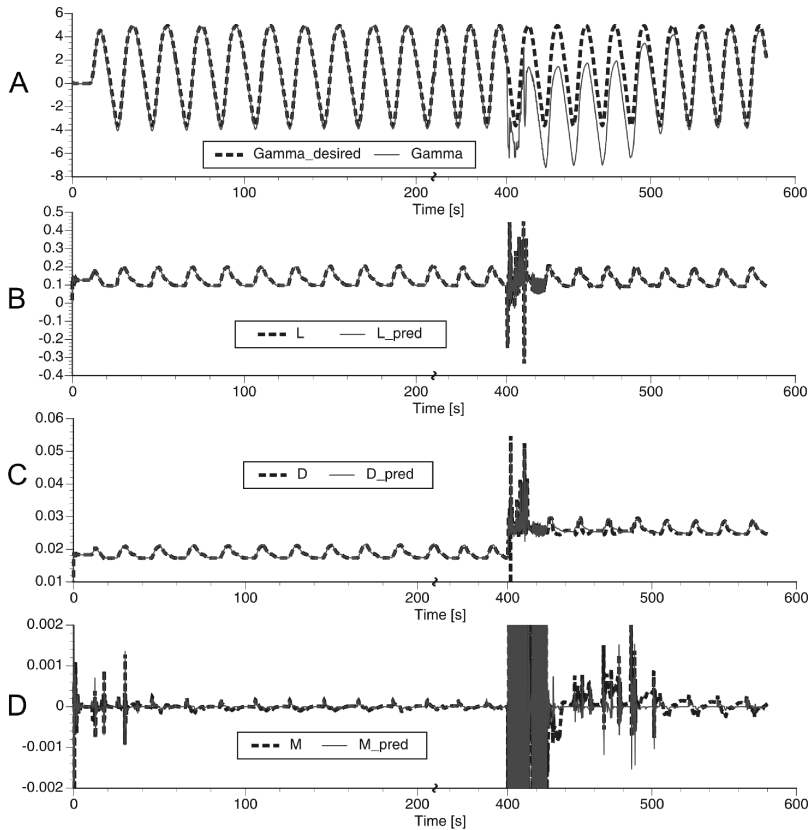
Figure 8: LWPR learning results for adaptive learning control on a simulated autonomous airplane. (A) Tracking of flight angle: $\gamma$. (B) Approximation of lift force: D. (C) Approximation of drag force: L. (D) Approximation of pitch moment: M. At 400 seconds into the flight, a failure is simulated that locks one control surface to a 17 degree angle. Note that for clarity of presentation, an axis break was inserted after 200 seconds.

up-and-down trajectories that are essentially sinusoidlike variations of the flight path angle $\gamma$.

Figure 8 demonstrates the results of this experiment. Figure 8A shows the desired trajectory in $\gamma$ and its realization by the controller. Figures 8B, 8C, and 8D illustrate the online function approximation of $D$, $L$, and $M$. As can be seen, the control of $\gamma$ achieves almost perfect tracking after just a few seconds. The function approximations of $D$ and $L$ are very accurate after a very short time. The approximation $M$ requires longer for convergence but

progresses quickly. About 10 local models were needed for learning $f_D$ and $f_L$, and about 20 local models were allocated for $f_M$.

An interesting element of Figure 8 happens after 400 seconds of flight, where we simulated a failure of the airplane mechanics by locking the MFR to a 17 degree deflection. As can be seen, the function approximators very quickly reorganize after this change, and the flight is successfully continued, although $\gamma$ tracking has some error for a while until it converges back to good tracking performance. The strong signal changes in the first seconds after the failure are due to oscillations of the control surfaces, not a problem in function approximation. Without adaptive control, the airplane would have crashed.

## 5 Discussion

Nonlinear regression with spatially localized models remains one of the most data-efficient and computationally efficient methods for incremental learning with automatic determination of the model complexity. In order to overcome the curse of dimensionality of local learning systems, this article investigated methods of linear projection regression and how to employ them in spatially localized nonlinear function approximation for high-dimensional input data with redundant and irrelevant components. Due to its robustness in such a setting, we chose partial least squares regression at the core of a novel function approximator, locally weighted projection regression (LWPR). The proposed technique was evaluated on a range of artificial and real-world data sets in up to 90-dimensional input spaces. Besides showing fast and robust learning performance due to second-order learning methods based on stochastic leave-one-out cross validation, LWPR excelled by its low computational complexity: updating each local model with a new data point remained linear in its computational cost in the number of inputs since the algorithm accomplishes good approximation results with only three to four projections irrespective of the number of input dimensions. To our knowledge, this is the first spatially localized incremental learning system that can efficiently work in high-dimensional spaces and is thus suited for online and real-time applications. In addition, LWPR compared favorably in its generalization performance with state-of-the-art batch regression regression methods like gaussian process regression and can provide qualitatively similar estimates of confidence bounds and predictive variances.

The major drawback of LWPR in its current form is the need for gradient descent to optimize the local distance metrics in each local model, and the manual tuning of a forgetting factor as required in almost all recursive learning algorithms that accumulate sufficient statistics. Future work will derive a probabilistic version of partial least squares regression that will allow a complete Bayesian treatment of locally weighted regression with locally linear models, with, we hope, no remaining open parameters for

manual adjustment. Whether a full Bayesian version of LWPR can achieve the same computational efficiency as in the current implementation, however, remains to be seen.

## Appendix A: PRESS Residuals for PLS

We prove that under the assumption that $\mathbf{x}$ lives in a reduced dimensional subspace, the PRESS residuals of equation 3.4 can indeed be replaced by the residual denoted by equation 3.5,

$$\mathbf{x}_i^T \mathbf{P} \mathbf{x}_i = \mathbf{z}_i^T \mathbf{P}_z \mathbf{z}_i, \tag{A.1}$$

where $\mathbf{P} = (\mathbf{X}^T \mathbf{X})^{\dagger}$, $\mathbf{P}_z = (\mathbf{Z}^T \mathbf{Z})^{-1}$ corresponds to the (pseudo)inverse covariance matrices and the symbol $\dagger$ represents the SVD pseudoinverse (Press, Flannery, Teukolsky, & Vetterling, 1989) that generates a solution to the inverse in the embedded lower-dimensional manifold of $\mathbf{x}$ with minimum norm solution in the sense of a Mahalanobis distance.
(Refer to Table 1 for the batch notations.)

**Part 1.** Let $\mathbf{T}$ be the transformation matrix with full rank in row space that denotes coordinate transformation from the rank-deficient space of $\mathbf{x}$ to the full rank space of $\mathbf{z}$. Then for any $\mathbf{z} = \mathbf{T}^T \mathbf{x}$ and the corresponding inverse covariance matrix $\mathbf{P}_z$, we can show that

$$\mathbf{z}_i^T \mathbf{P}_z \mathbf{z}_i = \mathbf{x}_i^T \mathbf{T}((\mathbf{XT})^T(\mathbf{XT}))^{-1} \mathbf{T}^T \mathbf{x}_i = \mathbf{x}_i^T (\mathbf{X}^T \mathbf{X})^{\dagger} \mathbf{x}_i = \mathbf{x}_i^T \mathbf{P} \mathbf{x}_i. \tag{A.2}$$

A linear transformation maintains the norm.

**Part 2.** In this part, we show that the recursive PLS projections that transform the inputs $\mathbf{x}$ to $\mathbf{z}$ can be written as a linear transformation matrix, which completes our proof. Clarifying some of the notation:[9]

$$\mathbf{X} \equiv \begin{bmatrix} \mathbf{x}_1 \\ \vdots \\ \mathbf{x}_M \end{bmatrix}, \quad \mathbf{Z} \equiv \begin{bmatrix} \mathbf{z}_1 \\ \vdots \\ \mathbf{z}_M \end{bmatrix} \equiv [\tilde{\mathbf{z}}_1 \ldots \tilde{\mathbf{z}}_R]. \tag{A.3}$$

We now look at each of the $R$ PLS projection directions and attempt to show that $\mathbf{z}_i = \mathbf{T}^T \mathbf{x}_i$ or (in a batch sense) $\mathbf{Z} = \mathbf{XT}$ by showing (for each individual projections) $r$, there exists a $\mathbf{t}_r$ such that

$$\tilde{\mathbf{z}}_r = \mathbf{X} \mathbf{t}_r, \quad where \quad \mathbf{T} = [\mathbf{t}_1 \ \mathbf{t}_2 \ \ldots \ \mathbf{t}_r]. \tag{A.4}$$

---

[9] Here, we have used $\mathbf{z}$ and $\tilde{\mathbf{z}}$ to distinguish between the row vectors and the column vectors, respectively, of the projected data matrix $\mathbf{Z}$.

For $r = 1$ (cf. Table 1),

$$\tilde{\mathbf{z}}_1 = \mathbf{X}\mathbf{X}^T\mathbf{y} = \mathbf{X}\mathbf{t}_1, \quad where \ \ \mathbf{t}_1 = \mathbf{X}^T\mathbf{y}. \tag{A.5}$$

For $r = 2$ (cf. Table 1),

$$\tilde{\mathbf{z}}_2 = \mathbf{X}_{res}\mathbf{X}_{res}^T\mathbf{y}_{res}. \tag{A.6}$$

We also know from the algorithm (cf. Table 1) that

$$\mathbf{X}_{res} = \mathbf{X} - \frac{\tilde{\mathbf{z}}_1\tilde{\mathbf{z}}_1^T\mathbf{X}}{\tilde{\mathbf{z}}_1^T\tilde{\mathbf{z}}_1} = \left(\mathbf{I} - \frac{\tilde{\mathbf{z}}_1\tilde{\mathbf{z}}_1^T}{\tilde{\mathbf{z}}_1^T\tilde{\mathbf{z}}_1}\right)\mathbf{X} = P_{\tilde{\mathbf{z}}_1}\mathbf{X} \tag{A.7}$$

$$\mathbf{y}_{res} = \mathbf{y} - \frac{\tilde{\mathbf{z}}_1\tilde{\mathbf{z}}_1^T\mathbf{y}}{\tilde{\mathbf{z}}_1^T\tilde{\mathbf{z}}_1} = \left(\mathbf{I} - \frac{\tilde{\mathbf{z}}_1\tilde{\mathbf{z}}_1^T}{\tilde{\mathbf{z}}_1^T\tilde{\mathbf{z}}_1}\right)\mathbf{y} = P_{\tilde{\mathbf{z}}_1}\mathbf{y}, \tag{A.8}$$

where $P_{\tilde{\mathbf{z}}_1}$ represents a projection operator. Using results from equations A.7 and A.8 in equation A.6,

$$\begin{aligned}\mathbf{s}_2 &= P_{\tilde{\mathbf{z}}_1}\mathbf{X}(P_{\tilde{\mathbf{z}}_1}\mathbf{X})^T P_{\tilde{\mathbf{z}}_1}\mathbf{y} \\ &= P_{\tilde{\mathbf{z}}_1}\mathbf{X}\mathbf{X}^T P_{\tilde{\mathbf{z}}_1}\mathbf{y} \ \dots \text{using property of projection operator: } P_{\tilde{\mathbf{z}}_1} = P_{\tilde{\mathbf{z}}_1}^T \\ &= P_{\tilde{\mathbf{z}}_1}P_{\tilde{\mathbf{z}}_1} = P_{\tilde{\mathbf{z}}_1}\mathbf{X}\mathbf{t}_2'. \end{aligned} \tag{A.9}$$

It can be shown easily by writing out the pseudo-inversion that there exists an operator $\mathbf{R}_2 = (\mathbf{I} - \mathbf{u}_1\mathbf{u}_1^t\mathbf{X}^T\mathbf{X}/\mathbf{u}_1^T\mathbf{X}^T\mathbf{X}\mathbf{u}_1)$ such that

$$P_{\tilde{\mathbf{z}}_1}\mathbf{X} = \mathbf{X}\mathbf{R}_2. \tag{A.10}$$

Using equations A.9 and A.10, we can write

$$\tilde{\mathbf{z}}_2 = \mathbf{X}\mathbf{t}_2 \ \ where \ \mathbf{t}_2 = \mathbf{R}_2\mathbf{t}_2' = \left(I - \frac{\mathbf{u}_1\mathbf{u}_1^t\mathbf{X}^T\mathbf{X}}{\mathbf{u}_1^T\mathbf{X}^T\mathbf{X}\mathbf{u}_1}\right)\mathbf{X}^T P_{\tilde{\mathbf{z}}_1}\mathbf{y}. \tag{A.11}$$

This operation can be carried out recursively to determine all the $\mathbf{t}_k$, showing that the PLS projections can be written as a linear transformation. This completes the proof of the validity of the modified PRESS residual of equation A.1 for PLS projections.

Also note that, $\mathbf{P}_z$ is diagonal by virtue of the fact that in the PLS algorithm, after every projection iteration, the projected components of input space $\mathbf{X}$ are subtracted before computing the next projection (Table 1d or Table 3 2b.3), ensuring the next component of $\mathbf{Z}$ will always be orthogonal to the previous ones. This property was discussed in Frank and Friedman (1993).

## Appendix B: Combined Predictive Variances

The noise model for combining prediction from individual local model is

$$y_{q,k} = y_q + \epsilon_1 + \epsilon_{2,k},$$

where $\epsilon_1 \sim N(0, \sigma^2/w_k)$ and $\epsilon_{2,k} \sim N(0, \sigma^2_{pred,k}/w_k)$. The mean prediction due to multiple local models can be written according to a heteroscedastic average,

$$\hat{y}_q = \sum_k \left( \frac{w_k}{\sigma^2 + \sigma^2_{pred,k}} y_{q,k} \right) \bigg/ \sum_k \left( \frac{w_k}{\sigma^2 + \sigma^2_{pred,k}} \right)$$

$$\approx \frac{\sum_k w_k y_{q,k}}{\sum_k w_k} \approx \frac{\sum_k w_k \hat{y}_{q,k}}{\sum_k w_k},$$

(B.1)

under the assumption that $(\sigma^2 + \sigma^2_{pred,k})$ is approximately constant for all contributing models $k$ and that $\hat{y}_{q,k}$ is an estimate over multiple noisy instances of $y_{q,k}$ that has averaged out the noise process $\epsilon_{2,k}$—exactly what happens within each local model. Thus, equation B.1 is consistent with equation 3.2 under the proposed dual noise model. The combined predictive variance can now be derived as

$$\sigma^2_{pred} = E\{y_q^2\} - (E\{y_q\})^2 = E\left\{ \left( \frac{\sum_k w_k y_{q,k}}{\sum_k w_k} \right)^2 \right\} - (E\{y_q\})^2$$

$$= \frac{1}{\left( \sum_k w_k \right)^2} E\left\{ \left( \sum_k w_k y_q \right)^2 + \left( \sum_k w_k \epsilon_1 \right)^2 + \left( \sum_k w_k \epsilon_{2,k} \right)^2 \right\} - (\hat{y}_q)^2$$

$$= \frac{1}{(\sum_k w_k)^2} E\left\{ \left( \sum_k w_k \epsilon_1 \right)^2 + \left( \sum_k w_k \epsilon_{2,k} \right)^2 \right\}.$$

(B.2)

Using the fact that $E\{x^2\} = (E\{x\})^2 + \text{var}(x)$ and noting that $\epsilon_1$ and $\epsilon_{2,k}$ have zero mean,

$$\sigma^2_{pred} = \frac{1}{\left( \sum_k w_k \right)^2} \text{var}\left( \sum_k w_k \epsilon_1 \right) + \frac{1}{\left( \sum_k w_k \right)^2} \text{var}\left( \sum_k w_k \epsilon_{2,k} \right)$$

$$= \frac{1}{\left( \sum_k w_k \right)^2} \left[ \sum_k w_k^2 \frac{\sigma^2}{w_k} \right] + \frac{1}{\left( \sum_k w_k \right)^2} \left[ \sum_k w_k^2 \frac{\sigma^2_{pred,k}}{w_k} \right]$$

$$\sigma_{pred}^2 = \frac{\sum_k w_k \sigma^2}{\left(\sum_k w_k\right)^2} + \frac{\sum_k w_k \sigma_{pred,k}^2}{\left(\sum_k w_k\right)^2}, \tag{B.3}$$

which gives the expression for the combined predictive variances.

## References

An, C. H., Atkeson, C., & Hollerbach, J. (1988). *Model based control of a robot manipulator*. Cambridge, MA: MIT Press.

Atkeson, C., Moore, A., & Schaal, S. (1997). Locally weighted learning. *Artificial Intelligence Review*, *11*(4), 76–113.

Belsley, D. A., Kuh, E., & Welsch, D. (1980). *Regression diagnostics*. New York: Wiley.

D'Souza, A., Vijayakumar, S., & Schaal, S. (2001). Are internal models of the entire body learnable? *Society for Neuroscience Abstracts*, 27.

Everitt, B. S. (1984). *An introduction to latent variable models*. London: Chapman and Hall.

Fahlman, S. E., & Lebiere, C. (1990). The cascade-correlation learning architecture. In D. S. Touretzky (ed.), *Advances in neural information processing systems, 2* (pp. 524–532). San Diego: Morgan-Kaufmann.

Frank, I., & Friedman, J. (1993). A statistical view of some chemometric tools. *Technometrics*, *35*(2), 109–135.

Friedman, J. H., & Stutzle, W. (1981). Projection pursuit regression. *Journal of America Statistical Association*, *76*, 817–823.

Gelman, A. B., Carlin, J. S., Stern, H. S., & Rubin, D. B. (1995). *Bayesian data analysis*. London: Chapman and Hall.

Ghahramani, Z., & Beal, M. (2000). Variational inference for Bayesian mixtures of factor analysers. In S. A. Solla, T. K. Leen, & K.-R. Müller (Eds.), *Advances in neural information processing systems, 12* (pp. 449–455). Cambridge, MA: MIT Press.

Gibbs, M., & Mackay, D. J. C. (1997). *Efficient implementation of gaussian processes* (Tech. Rep.). Cambridge: Cavendish Laboratory.

Hastie, T., & Loader, C. (1993). Local regression: Automatic kernel carpentry. *Statistical Science*, *8*, 120–143.

Hastie, T., & Tibshirani, R. (1990). *Generalized additive models*. London: Chapman and Hall.

Hettich, S., & Bay, S. (1999). The UCI KDD archive. London: University of California, Irvine, Department of Information and Computer Science.

Jordan, M., & Jacobs, R. (1994). Hierarchical mixture of experts and the EM algorithm. *Neural Computation*, *6*(2), 181–214.

Ljung, L., & Soderstrom, T. (1986). *Theory and practice of recursive identification*. Cambridge, MA: MIT Press.

Myers, R. H. (1990). *Classical and modern regression with applications*. Boston: PWS-Kent.

Nakanishi, J., Farrell, J. A., & Schaal, S. (2004). Composite adaptive control with locally weighted statistical learning. In *IEEE International Conference on Robotics and Automation* (pp. 2647–2652). Piscataway, NJ: IEEE.

Perrone, M. P., & Cooper, L. N. (1993). *Neural networks for speech and image processing*, London: Chapman Hall.

Platt, J. (1991). A resource-allocating network for function interpolation. *Neural Computation*, *3*(2), 213–225.

Press, W., Flannery, B., Teukolsky, S., & Vetterling, W. (1989). *Numerical recipes in C: The art of scientific computing*. Cambridge: Cambridge University Press.

Roweis, S., & Saul, L. (2000). Nonlinear dimensionality reduction by local linear embedding. *Science*, *290*, 2323–2326.

Rubin, D., & Thayer, D. (1982). EM algorithms for ML factor analysis. *Psychometrika*, *47*(1), 69–76.

Sanger, T. (1989). Optimal unsupervised learning in a single layer feedforward neural network. *Neural Networks*, *2*, 459–473.

Saunders, C., Stitson, M., Weston, J., Bottou, L., Schoelkopf, B., & Smola, A. (1998). *Support vector machine—reference manual* (Tech. Rep. TR CSD-TR-98-03). London: Department of Computer Science, Royal Holloway, University of London.

Schaal, S., & Atkeson, C. (1994). Assessing the quality of learned local models. In J. D. Cowan, G. Tesauro, & J. Alspetor (Eds.). *Advances in neural information processing systems, 6* (pp. 160–167). San Mateo, CA: Morgan-Kaufmann.

Schaal, S., & Atkeson, C. (1997). *Receptive field weighted regression* (Tech. Rep. TR-H-209). Kyoto, Japan: ATR Human Information Processing.

Schaal, S., & Atkeson, C. (1998). Constructive incremental learning from only local information. *Neural Computation*, *10*(8), 2047–2084.

Schaal, S., Vijayakumar, S., & Atkeson, C. G. (1998). Local dimensionality reduction. In M. I. Jordan, M. J. Kearns, & S. A. Solla (Eds.), *Advances in neural information processing systems*, *10*. Cambridge, MA: MIT Press.

Schölkopf, B., Burgess, C., & Smola, A. J. (1999). *Advances in kernel methods: Support vector learning*. Cambridge, MA: MIT Press.

Schölkopf, B., Smola, A. J., Williamson, R., & Bartlett, P. (2000). New support vector algorithms. *Neural Computation*, *12*(5), 1207–1245.

Scott, D. (1992). *Multivariate density estimation*. New York: Wiley.

Smola, A. J., & Schölkopf, B. (1998). *A tutorial on support vector regression* (NEURO-COLT Tech. Rep. NC-TR-98-030). London: Royal Holloway College.

Stevens, B. L., & Lewis, F. L. (2003). *Aircraft control and simulation*. New York: Wiley.

Tenenbaum, J., de Silva, V., & Langford, J. (2000). A global geometric framework for nonlinear dimensionality reduction. *Science*, *290*, 2319–2323.

Tipping, M., & Bishop, C. (1999). Probabilistic principal component analysis. *Journal of the Royal Statistical Society Series B*, *61*(3), 611–622.

Ueda, N., Nakano, R., Ghahramani, Z., & Hinton, G. (2000). SMEM algorithm for mixture models. *Neural Computation*, *12*, 2109–2128.

Vijayakumar, S., & Ogawa, H. (1999). RKHS based functional analysis for exact incremental learning. *Neurocomputing*, *29*(1–3), 85–113.

Vijayakumar, S., & Schaal, S. (1998). Local adaptive subspace regression. *Neural Processing Letters*, *7*(3), 139–149.

Vlassis, N., Motomura, Y., & Krose, B. (2002). Supervised dimensionality reduction of intrinsically low-dimensional data. *Neural Computation*, *14*, 191–215.

Williams, C. K. I., & Rasmussen, C. (1996). Gaussian processes for regression. In M. Touretzky & Hasselmo (Eds.), *Advances in neural information processing systems*, *8*. Cambridge, MA: MIT Press.

Williams, C. K. I., & Seeger, M. (2001). Using the Nystrom method to speed up kernel machines. In T. G. Dietterich, S. Becker, & Z. Ghahramani (Eds.), *Advances in neural information processing systems, 14*. Cambridge, MA: MIT Press.

Wold, H. (1975). *Perspectives in probability and statistics*. London: Chapman Hall.

Xu, L., Jordan, M., & Hinton, G. (1995). An alternative model for mixtures of experts. In G. Tesauro, D. Touretzky, & T. Leen (Eds.), *Advances in neural information processing systems, 7* (pp. 633–640). Cambridge, MA: MIT Press.