Neural Process Lett (2009) 29:109–131
DOI 10.1007/s11063-009-9098-0

Local Dimensionality Reduction for Non-Parametric Regression

Heiko Hoffmann · Stefan Schaal · Sethu Vijayakumar

Published online: 13 February 2009

Abstract Locally-weighted regression is a computationally-efficient technique for non-linear regression. However, for high-dimensional data, this technique becomes numerically brittle and computationally too expensive if many local models need to be maintained simultaneously. Thus, local linear dimensionality reduction combined with locally-weighted regression seems to be a promising solution. In this context, we review linear dimensionality-reduction methods, compare their performance on non-parametric locally-linear regression, and discuss their ability to extend to incremental learning. The considered methods belong to the following three groups: (1) reducing dimensionality only on the input data, (2) modeling the joint input-output data distribution, and (3) optimizing the correlation between projection directions and output data. Group 1 contains principal component regression (PCR); group 2 contains principal component analysis (PCA) in joint input and output space, factor analysis, and probabilistic PCA; and group 3 contains reduced rank regression (RRR) and partial least squares (PLS) regression. Among the tested methods, only group 3 managed to achieve robust performance even for a non-optimal number of components (factors or projection directions). In contrast, group 1 and 2 failed for fewer components since these methods rely on the correct estimate of the true intrinsic dimensionality. In group 3, PLS is the only method for which a computationally-efficient incremental implementation exists.

H. Hoffmann (✉) · S. Vijayakumar
IPAB, School of Informatics, University of Edinburgh, King's Buildings, Mayfield Road, Edinburgh, EH9 3JZ, UK
e-mail: heiko@clmc.usc.edu

Present Address:
H. Hoffmann
Biomedical Engineering, University of Southern California, RTH 421, 3710 S. McClintock Ave, Los Angeles, CA 90089-2905, USA

S. Schaal
Computer Science and Neuroscience, University of Southern California, RTH 401, 3710 S. McClintock Ave, Los Angeles, CA 90089-2905, USA

Thus, PLS appears to be ideally suited as a building block for a locally-weighted regressor in which projection directions are incrementally added on the fly.

Keywords Correlation · Dimensionality reduction · Factor analysis · Incremental learning · Kernel function · Locally-weighted regression · Partial least squares · Principal component analysis · Principal component regression · Reduced-rank regression

1 Introduction

Regression models the continuous relationship between two sets of variables, usually called inputs and outputs (or independent and dependent variables). The process of modeling entails finding the structure as well as the free parameters of a function such that it optimally describes a given set of input and output data. Regression is a generic and important statistical tool with a wide field of applications ranging from data mining, signal processing, chemometrics [52], and econometrics [16] to adaptive learning control and robotics [47].

A common approach to non-linear regression is to approximate an input-output relationship with a linear combination of basis functions [8]. Popular examples of this approach are neural networks [8], support vector regression [44], and Gaussian process regression [35]; the latter is also known as "kriging" [9,28]. Increasingly popular among these methods is Gaussian process regression, which offers a sound probabilistic treatment: the Gaussian process is a probability distribution over functions, resulting in little parameter tuning and confidence boundaries on output values [35].

Unfortunately, Gaussian process regression is computationally expensive: training (finding the free parameters) is $O(n^3)$, where n is the number of data points;[1] furthermore, testing (applying the function to a test point) is $O(n^2)$. A better computational complexity scaling has support vector regression, $O(n^2)$ for training [40]. However, support vector regression does not provide confidence boundaries. A probabilistic version was realized by the relevance vector machine [42], which is, however, again $O(n^3)$.

Real-time applications—like, for example, adaptive robot control—require computation speeds that are still beyond the above-mentioned regression techniques [48]. Here, a feasible alternative is locally-weighted regression [3,45–47]. Locality is introduced by weighting the data such that effectively only points in the neighborhood of the local model (with distances measured relative to the center of the local model) contribute to the regression [2,14]. Thus, the kernel function determines the weights of single data points and not the basis function of a local model as above, which is here usually linear [2]. Given the weights, a local model can be computed in $O(n)$ time by least-squares regression. Confidence boundaries can be computed by assuming homoscedastic noise for each local model [48] .

In high-dimensional space, however, confining models locally may potentially be catastrophic: the proportional volume of the neighborhood decreases exponentially with increasing dimensionality; thus, eventually this volume may not contain enough data points for a meaningful estimation of the regression coefficients—see the 'curse of dimensionality' [6]. The alternative is to use large local models, which will lead to hopeless over-smoothing. Moreover, with increasing dimensionality, typically, all data points tend to have the same distance to each other [7]; thus, spatial localization becomes meaningless.

[1] Faster approximations exist that essentially work with a reduced training data set [35].

In many real-world applications, fortunately, high-dimensional data are confined to locally-low-dimensional distributions—see Sect. 2. Hence, if we have a local regression technique that exploits these low-dimensional distributions, then, we can hope to carry over all the potential benefits of locally-weighted techniques to high-dimensional real-world data: locally-weighted regression requires inversion of the covariance matrix of the data, which is $O(d^3)$, with d equal the dimensionality of the data. Thus, reducing the dimensionality is critical.

The dimensionality of a data distribution may be reduced with global[2] non-linear techniques [4,19,36,41,51]. However, these techniques are computationally expensive: semi-definite embedding is $O(n^3)$ [51]; locally-linear embedding is $O(n^2)$ [36]; Isomap is $O(n^3)$ [41], Laplacian eigenmaps are $O(n^2)$ [4], and about the stochastic neighbor embedding, Hinton and Roweis note: "it takes several hours to find a good embedding for just 3,000 data points" [19].

If using locally-confined linear regression, the natural alternative to the above non-linear dimensionality reduction techniques is linear dimensionality reduction—separately, for each locally-confined model. The primary aim of this paper is to compare linear dimensionality-reduction techniques suited for locally-weighted regression; an aim which is slightly divergent from generic dimensionality-reduction which aims at data preservation, optimal reconstruction or visualization, for example.

Real-time applications usually require an incremental learning scheme. Such a scheme has two big advantages: first, it is memory efficient since only one training pattern needs to be stored at a time (in addition to the sufficient statistics), and second, it adapts quickly to changes in the environment without catastrophic failure in the transition phase (graceful degradation). Thus, we will discuss how the methods that we test extend to incremental learning.

To find out which dimensionality-reduction methods are most suitable for locally-weighted regression, we compare six state-of-the-art methods, which are grouped into (1) dimension reduction only on the input data, (2) modeling the joint input-output data distribution, and (3) maximizing the correlation between projection directions and output data. To group 1 belongs principal component regression (PCR); to group 2 belongs principal component analysis (PCA) in joint space, factor analysis (FA), and probabilistic PCA (PPCA) in joint space; and to group 3 belongs reduced-rank regression (RRR) and partial least squares (PLS). Where applicable, we derive corresponding weighted and incremental formulations.

Through extensive empirical comparison studies, we show that all tested dimensionality-reduction methods perform reasonably well if the number of components (factors or projection directions) matches the intrinsic dimensionality of the data distribution, but for fewer components, group 1 and 2 methods fail. The number of components, however, is typically not given beforehand. In incremental learning, components are added incrementally based on a test criterion in a data driven manner. If the number of components is less than the intrinsic dimensionality, PLS—among the algorithms that can be formulated in an incremental scheme—results in the lowest prediction error, making it an ideal candidate for adding projection directions on the fly. Furthermore, we show that when applying PLS for locally-weighted regression, the optimal distance metric (or locality neighborhood) is relatively insensitive to the choice of number of projections.

The remainder of the article is organized as follows. Section 2 provides evidence for locally-low-dimensional distributions. Section 3 introduces locally-linear regression. Section 4 explains the dimensionality-reduction methods. Section 5 evaluates these methods, first, on a synthetic data set with known structure, and second, on two real-world data sets. Section 6 discusses the results of the experiments, and Sect. 7 closes with conclusions.

[2] Here, meaning "not locally confined".

2 Evidence for Locally-Low-Dimensional Distributions

Our methodology for dimensionality reduction for regression relies on the assumption that high-dimensional data sets have locally-low-dimensional distributions, an assumption that requires some clarification. Across domains like vision, speech, motor control, climate patterns, human gene distributions, and a range of other physical and biological sciences, various researchers have shown that the true intrinsic dimensionality of high dimensional data is often very low [21,36,41,49]. We interpret these findings as evidence that the physical world has a significant amount of coherent structure that expresses itself in terms of strong correlations between different variables that describe the state of the world at a particular moment in time. For instance, in computer vision, neighboring pixels of an image of a natural scene have redundant information. Moreover, the probability distribution of natural scenes in general has been found to be highly structured such that it lends itself to a sparse encoding in terms of set of basis functions [5,33].

Another example comes from our own research on human motor control. Despite that humans can accomplish movement tasks in almost arbitrary ways—thus possibly generating arbitrary distributions of the variables that describe their movements—behavioral research has discovered regularities within and across individuals [26,38]. These regularities lead to locally-low-dimensional data distributions, as illustrated in the example in Fig. 2. In this analysis [12], we assessed the intrinsic dimensionality of data collected from full-body movements of several human subjects. The data were collected with a special full-body exoskeleton (see Fig. 1) that recorded simultaneously 35 joint angles at 100 Hz sampling frequency. Subjects performed a variety of daily-life tasks (e.g., walking, object manipulation, and reaching) until about a gigabyte of data was accumulated. Our analysis examined the local dimensionality of the joint distribution of positions, velocities, and accelerations of the collected data, i.e., a 105-dimensional data set, as would be needed as inputs to learn an inverse dynamics

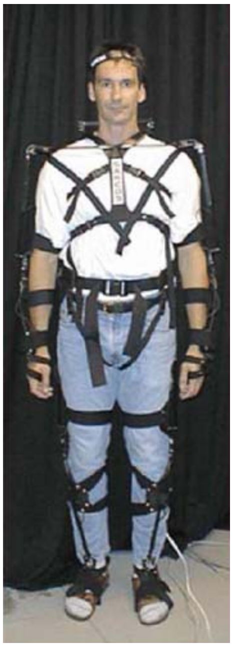
Fig. 1 Thirty-degrees-of-freedom sensuit used to capture human kinematic movement data

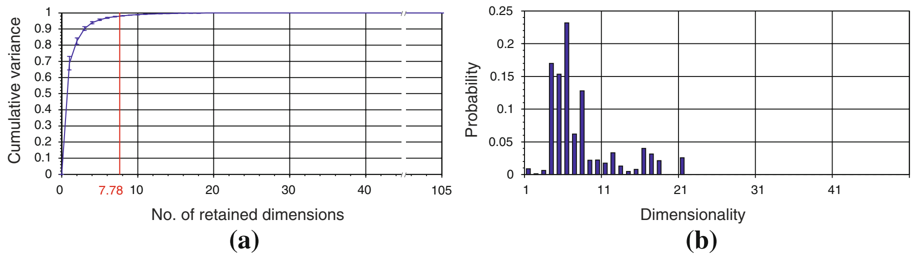

Fig. 2 Dimensionality analysis **a** The cumulative variance accounted versus the local dimensionality (averaged across all mixture models; an average of 7.78 dimensions accounted for 99% of the variance) **b** The distribution of the effective dimensionality across all mixture models

model for motor control [26]. To analyze the local dimensionality, we employed a variational Bayesian mixture of factor analyzers that automatically estimated the required number of mixture components [17]. As shown in Fig. 2(a), the local dimensionality was around 5–8 dimensions, computed based on the average number of significant latent variables per mixture component. Figure 2(b) shows the distribution of the effective dimensionality across all mixture models.

In summary, the results from our analysis and other sources in the literature show that there is a large class of high dimensional problems that can be treated locally in much lower dimensions if one can determine appropriate regions of locality and the local projections that model the corresponding low dimensional distributions. As a caveat, however, it may happen that such low dimensional distributions are embedded in additional dimensions that are irrelevant for the problem at hand but have considerable variance. In the context of regression, it will thus be important to model only those local dimensions that carry information that is important for the regression and eliminate all other dimensions, i.e., to perform local dimensionality reduction with the conditional distribution of regression in mind and not just based on input or joint input-output distributions.

3 Locally-Linear Regression

Linear regression assumes a linear relationship of an input vector $\mathbf{x}$ and an output variable y. Here, for simplicity, we consider only the case of one output variable, since, on the one hand, many problems can be decomposed into multiple mappings onto univariate outputs, and, on the other hand, the generalization to many output variables is mostly straightforward. We also assume, without loss of generality, that the mean values of $\mathbf{x}$ and y are zero. Given this simplification, the model function in linear regression is

$$y = \boldsymbol{\beta}^T \mathbf{x} + \epsilon_y. \tag{1}$$

Here, $\mathbf{x}$ is a d-dimensional input vector; y is the output value; $\boldsymbol{\beta}$ are the regression coefficients, and ϵ_y is a homoscedastic (independent of $\mathbf{x}$) noise variable.[3] Furthermore, let n be the number of training data points $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$. The coefficients $\boldsymbol{\beta}$ can be obtained by minimizing the expected squared error E,

[3] For locally-weighted regression, we can deal with hetereoscedastic data as long as the noise variance is approximately constant in each local model.

$$E = \sum_{i=1}^{n} ||y_i - \boldsymbol{\beta}^T \mathbf{x}_i||^2, \tag{2}$$

which results in the ordinary-least-squares solution

$$\boldsymbol{\beta} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y}. \tag{3}$$

The matrix $\mathbf{X}$ contains the input vectors $\mathbf{x}_i$ in its rows, and the vector $\mathbf{y}$ contains the n output values y_i.

This solution has also a probabilistic interpretation. If we assume that ϵ_y has a Gaussian distribution of variance σ^2, then the conditional probability of y given $\mathbf{x}$ can be written as $p(y|\mathbf{x}) \propto \exp(-\frac{1}{2}(y - \boldsymbol{\beta}^T \mathbf{x})^2/\sigma^2)$. Equation (3) maximizes the likelihood, $L = \prod_i p(y_i, \mathbf{x}_i)$, of the training data, given a constant prior $p(\mathbf{x})$.

In the locally-linear case, the data points $\{\mathbf{x}_i\}$ are weighted depending on their relative position from the center $\mathbf{c}$ of a local model. The most common weighting function—and the only one we will deal with here—is the Gaussian function,

$$w_i = \exp\left(-\frac{1}{2}(\mathbf{x}_i - \mathbf{c})^T \mathbf{D}(\mathbf{x}_i - \mathbf{c})\right). \tag{4}$$

The matrix $\mathbf{D}$ is a positive semi-definite distance metric that determines the locality (region of influence) of the local model. Regression methods can be adapted to the local case by multiplying each data point in the loss function (2) with the corresponding weight w_i [2],

$$E_w = \sum_{i=1}^{n} w_i ||y_i - \boldsymbol{\beta}^T \mathbf{x}_i||^2, \tag{5}$$

Thus, in the locally-weighted case, the solution (3) is replaced by

$$\boldsymbol{\beta} = (\mathbf{X}^T \mathbf{W} \mathbf{X})^{-1} \mathbf{X}^T \mathbf{W} \mathbf{y}, \tag{6}$$

where $\mathbf{W}$ is a diagonal matrix, containing the w_i's along its diagonal. Weighting the loss function with w_i is equivalent to weighting each single data point $\mathbf{x}_i$ and output y_i with $\sqrt{w_i}$ [2]. The same reformulation also holds for the dimensionality-reduction methods discussed below [39].

4 Dimensionality-Reduction Methods

Dimensionality-reduction methods reduce a d-dimensional data set to a k-dimensional set with $k < d$, such that key characteristics of the data are retained—the exact definition of "key characteristics" is going to be important, as will become apparent below.

The following notations will be used across all methods: the (mean zero) input data are stored in the $n \times d$ matrix $\mathbf{X}$, and the (mean zero) output data are in the n-dimensional vector $\mathbf{y}$. For the methods that combine input and output into a joint space, the $n \times (d + 1)$ matrix $\mathbf{Z}$ contains in its rows the vectors $\mathbf{z}_i^T = [\mathbf{x}_i^T, y_i]$.

The following subsections present the fundamentals of six methods—principal component regression, principal component analysis in joint space, factor analysis, probabilistic PCA in joint space, partial least squares, and reduced-rank regression. Each section will show how dimensionality-reduction can be used in a regression setting and will also address how the method, if applicable, extends to incremental learning.

4.1 Principal Component Regression (PCR)

Principal component regression is a widespread tool for reducing the dimensionality of the input space. Here, a principal component analysis is applied to the input data before computing ordinary least squares on the principal subspace.

The principal components are the eigenvectors of the covariance matrix $\mathbf{C} = \mathbf{X}^T\mathbf{X}$. Let $\mathbf{U}$ be a matrix that stores these eigenvectors in its columns. For regression, the input is first mapped onto the principal subspace. Then, on this subspace, we compute ordinary least squares regression. Thus, we minimize $||\mathbf{y} - \mathbf{XU}\boldsymbol{\alpha}||^2$ with respect to the reduced coefficients $\boldsymbol{\alpha}$. Therefore, the final regression coefficients are

$$\boldsymbol{\beta} = \mathbf{U}\boldsymbol{\alpha} = \mathbf{U}(\mathbf{U}^T\mathbf{X}^T\mathbf{XU})^{-1}\mathbf{U}^T\mathbf{X}^T\mathbf{y} = \mathbf{U}\boldsymbol{\Lambda}^{-1}\mathbf{U}^T\mathbf{X}^T\mathbf{y}, \tag{7}$$

where $\boldsymbol{\Lambda}$ is a diagonal matrix containing the eigenvalues of the covariance matrix $\mathbf{C}$. In the locally-weighted version, the coefficients become

$$\boldsymbol{\beta}_w = \mathbf{U}_w\boldsymbol{\Lambda}_w^{-1}\mathbf{U}_w^T\mathbf{X}^T\mathbf{W}\mathbf{y}; \tag{8}$$

here, $\mathbf{U}_w$ and $\boldsymbol{\Lambda}_w$ are extracted from the weighted covariance matrix $\mathbf{C}_w = \mathbf{X}^T\mathbf{WX}$.

To obtain an incremental version of PCR, we basically need an incremental PCA routine. Many methods exist for extracting the principal components incrementally [11,31,32,34,37]. These methods compute the estimates of the eigenvectors and eigenvalues at time step $t+1$, $\mathbf{U}(t+1)$ and $\boldsymbol{\Lambda}(t+1)$, given $\mathbf{U}(t)$, $\boldsymbol{\Lambda}(t)$, and a new data sample $(\mathbf{x}, y)$. Furthermore, the regression coefficients can be updated incrementally as $\boldsymbol{\beta}(t+1) = \boldsymbol{\beta}(t) + \eta\,(\mathbf{U}(t)\boldsymbol{\Lambda}^{-1}(t)\mathbf{U}^T(t)\,\mathbf{x}y - \boldsymbol{\beta}(t))$, where η is the learning rate.

4.2 Principal Component Analysis in Joint Space (PCAJ)

As an alternative to PCR, the principal components can be also extracted in the joint space of input and output. Thus, different from above, the eigenvectors are extracted from the covariance matrix in joint space, $\mathbf{C} = \mathbf{Z}^T\mathbf{Z}$, with $\mathbf{Z} = [\mathbf{X}\ \mathbf{y}]$. In this space, the principal subspace describes the orientation of the data distribution. For regression, PCAJ has been used to identify directions in input space that have high predictive value [50]. Here, however, we use the principal subspace directly for regression by mapping an input point as close as possible to this subspace [39].

The matrix $\mathbf{U}$ can be decomposed into $\mathbf{U}_x$ for the input space and $\mathbf{U}_y$ for the output space, $\mathbf{U}^T = [\mathbf{U}_x^T\ \mathbf{U}_y^T]$. To achieve a mapping from input to output space, first, a point on the principal subspace (given by $\mathbf{Uv}$) is obtained, whose input-space component is closest to a given input $\mathbf{x}$: the free variables $\mathbf{v}$ are obtained by minimizing $||\mathbf{x} - \mathbf{U}_x\mathbf{v}||^2$ with respect to $\mathbf{v}$. This results in $\mathbf{v} = (\mathbf{U}_x^T\mathbf{U}_x)^{-1}\mathbf{U}_x\mathbf{x}$. Finally, the output is given by $y = \mathbf{U}_y\mathbf{v}$. Therefore, the regression coefficients are

$$\boldsymbol{\beta} = \mathbf{U}_y(\mathbf{U}_x^T\mathbf{U}_x)^{-1}\mathbf{U}_x. \tag{9}$$

Using the orthonormality of $\mathbf{U}$, which implies $\mathbf{U}_x^T\mathbf{U}_x + \mathbf{U}_y^T\mathbf{U}_y = 1$, and the Woodbury matrix identity,[4] the expression for the coefficients $\boldsymbol{\beta}$ can be expressed in a computationally more efficient way:

$$\boldsymbol{\beta} = \mathbf{U}_x(\mathbf{U}_y^T - \mathbf{U}_y^T(\mathbf{U}_y\mathbf{U}_y^T - \mathbf{I})^{-1}\mathbf{U}_y\mathbf{U}_y^T). \tag{10}$$

[4] For invertible square matrices $\mathbf{A}$ and $\mathbf{C}$, the following identity holds: $(\mathbf{A}+\mathbf{UCU}^T)^{-1} = \mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{U}(\mathbf{C}^{-1} + \mathbf{U}^T\mathbf{A}^{-1}\mathbf{U})^{-1}\mathbf{U}^T\mathbf{A}^{-1}$.

For one-dimensional output, the matrix inversion is just a scalar division. In the locally-weighted version, the eigenvectors are extracted from the weighted covariance matrix $\mathbf{C}_w = \mathbf{X}^T\mathbf{W}\mathbf{X}$. To make this algorithm incremental, as for PCR, we need an incremental PCA technique, and the regression coefficients $\boldsymbol{\beta}$ can be computed using (10) based on the current estimate of $\mathbf{U}$.

4.3 Factor Analysis (FA)

Factor analysis is one of the statistically most sound methods for dimensionality reduction in linear systems. Different from PCR and PCAJ, FA is derived from the assumption that the data are generated from a linear model with $k < d$ hidden variables plus a noise term: $\mathbf{z} = \mathbf{U}\mathbf{v} + \boldsymbol{\epsilon}$ [13]. Here, $\mathbf{z}$ is a point in the joint space of input and output. The latent variables $\mathbf{v}$ are assumed to be spherically distributed, $\mathcal{N}(0, \mathbf{I})$. The noise $\boldsymbol{\epsilon}$ is assumed to be distributed according to $\mathcal{N}(0, \boldsymbol{\Omega})$, with a diagonal matrix $\boldsymbol{\Omega}$.

Under these assumptions, the unknown parameters $\mathbf{U}$ and $\boldsymbol{\Omega}$ can be obtained by maximizing the likelihood of the data $\{\mathbf{z}_i\}$. This maximization can be carried out using the expectation-maximization (EM) algorithm, as, for example, described in [18]. As result, the conditional probability $p(\mathbf{z}|\mathbf{v})$ is given, which is distributed according to $\mathcal{N}(\mathbf{U}\mathbf{v}, \boldsymbol{\Omega})$.

For regression, we are interested in the expectation value of y, $E(y)$, which equals $\boldsymbol{\beta}^T\mathbf{x}$. To obtain this value, we compute $p(\mathbf{y}|\mathbf{x})$. This computation is achieved by the following two steps. First, we decompose $p(\mathbf{z}|\mathbf{v})$ into $p(\mathbf{x}|\mathbf{v}) \sim \mathcal{N}(\mathbf{U}_x\mathbf{v}, \boldsymbol{\Omega}_x)$ and $p(y|\mathbf{v}) \sim \mathcal{N}(\mathbf{U}_y\mathbf{v}, \boldsymbol{\Omega}_y)$ by splitting $\mathbf{U}$ and $\boldsymbol{\Omega}$ into the respective components for the input and output space. Second, we use Bayes' rule to compute $p(\mathbf{v}|\mathbf{x}) = p(\mathbf{x}|\mathbf{v})p(\mathbf{v})/p(\mathbf{x})$ and marginalize $\mathbf{v}$: $p(y|\mathbf{x}) = \int p(y|\mathbf{v})p(\mathbf{v}|\mathbf{x})d\mathbf{v}$. Since $p(y|\mathbf{x})$ is Gaussian, $E(y)$ is the center of this function. The result is

$$\boldsymbol{\beta} = \mathbf{U}_y\mathbf{U}_x^T(\boldsymbol{\Omega}_x + \mathbf{U}_x\mathbf{U}_x^T)^{-1}. \tag{11}$$

To obtain a locally-weighted version, in the EM algorithm [18], each data point $\mathbf{z}_i$ is multiplied by $\sqrt{w_i}$.

FA lacks a proper incremental formulation, although it is possible to create ad hoc incremental implementations that accumulate the sufficient statistics of FA incrementally with forgetting factors. Furthermore, FA is computationally more expensive than the other methods because of the iterative EM algorithm, and FA requires a little bit of noise in the input data—otherwise, the algorithm encounters numerical singularities. On the positive side, FA is actually able to deal with noise in the inputs in a principled way, which is superior in theory to any of the other algorithms in this paper.

4.4 Probabilistic Principal Component Analysis (PPCA)

Probabilistic PCA is a special case of factor analysis under the assumption of isotropic noise [43]. This restriction allows an analytic solution, which omits the EM-algorithm. The model density $p(y, \mathbf{x})$ in joint space is described by a multivariate Gaussian, $\mathcal{N}(0, \mathbf{A}^{-1})$ with

$$\mathbf{A} = \mathbf{U}(\boldsymbol{\Lambda}^{-1} - \mathbf{I}_k/\sigma^2)\mathbf{U}^T + \mathbf{I}_d/\sigma^2, \tag{12}$$

where $\mathbf{I}_k$ is the $k \times k$ identity matrix; $\mathbf{U}$ and $\boldsymbol{\Lambda}$ are the eigenvectors and eigenvalues as obtained from a PCA in joint space, and σ is the residual variance, $\sigma^2 = \text{trace}(\mathbf{C}) - \text{trace}(\boldsymbol{\Lambda})$, using the covariance matrix $\mathbf{C}$. Different from factor analysis, here, the variance σ^2 is the same for all residual dimensions.

For regression, we use the probabilistic density $p(y, \mathbf{x})$ as in factor analysis (Sect. 4.3) and not the intersection with the principal subspace as in PCAJ (Sect. 4.2)—otherwise the results would be the same as for PCAJ. Since $p(y, \mathbf{x})$ is readily available, we choose the coefficients $\boldsymbol{\beta}$ such that the output y maximizes $p(y, \mathbf{x})$ for a given input $\mathbf{x}$ [22,23]. As stated above, $p(y, \mathbf{x}) \propto \exp(-\frac{1}{2}[\mathbf{x}^T, y]\mathbf{A}[\mathbf{x}^T, y]^T)$. Thus, we decompose $\mathbf{A}$ into the components of input and output space:

$$\mathbf{A} = \begin{pmatrix} \mathbf{A}_{xx} & \mathbf{A}_{xy} \\ \mathbf{A}_{xy}^T & \mathbf{A}_{yy} \end{pmatrix}. \tag{13}$$

The optimal output y maximizes $p(y, \mathbf{x})$; therefore, we set the derivative of $p(y, \mathbf{x})$ with respect to y equal zero. This step results in $y = \mathbf{A}_{yy}^{-1}\mathbf{A}_{xy}^T\mathbf{x}$. Thus, the regression coefficients are

$$\boldsymbol{\beta} = \mathbf{A}_{xy}\mathbf{A}_{yy}^{-1}. \tag{14}$$

For a one-dimensional output y, the inversion of $\mathbf{A}_{yy}$ is just a division by a scalar. An alternative derivation can be obtained by taking $\mathbf{A}$ as an ellipsoid and by computing the value y that results in the smallest Mahalanobis distance of $[\mathbf{x}^T, y]$ to the origin [22,23].

For the locally-weighted version, the eigenvectors and eigenvalues are extracted from the weighted covariance matrix $C_w = \mathbf{X}^T\mathbf{W}\mathbf{X}$. For incremental learning, the eigenvectors $\mathbf{U}$ and eigenvalues $\boldsymbol{\Lambda}$ can be computed as in PCAJ. In addition, an estimate for the residual variance $v_{\text{res}} = (d-k)\sigma^2$ needs to be computed, which can be obtained by iterating

$$v_{\text{res}}(t+1) = v_{\text{res}}(t-1) + \eta\left(\mathbf{z}^T\mathbf{z} - \mathbf{z}^T\mathbf{U}\mathbf{U}^T\mathbf{z} - v_{\text{res}}(t-1)\right), \tag{15}$$

where η is the learning rate as in incremental PCA. Based on the current estimates of $\mathbf{U}$, $\boldsymbol{\Lambda}$, and σ, the regression coefficients can be updated according to (12), (13), and (14).

4.5 Reduced-Rank Regression (RRR)

Reduced-rank regression [24]—a common model in econometrics [16]—is multiple regression with a rank constraint on the coefficient matrix $\mathbf{B}$, as in

$$\mathbf{Y} = \mathbf{X}\mathbf{B} + \mathbf{E}, \tag{16}$$

where $\mathbf{Y}$ is the output and $\mathbf{E}$ the error matrix. Since, in our case, we consider only a 1-dimensional output, the rank of $\mathbf{B}$ is always 1, i.e, the rank cannot be constrained anymore. Thus, here, reduced-rank regression does not do dimensionality reduction and, therefore, does not provide a computational gain over ordinary least squares, which gives the same results. Still, we put RRR into this comparison for two reasons: first, as a control showing the optimal regression performance (the same holds for ordinary least squares), and second, as a context for PLS, which may be regarded as a mix between PCR and RRR (see Sect. 4.6).

Reduced-rank regression methods extract projections of maximal correlation between input and output. These projections are extracted by maximizing the squared correlation

$$\text{corr}^2(\mathbf{X}\mathbf{u}_i, \mathbf{y}) = (\mathbf{u}_i^T\mathbf{X}^T\mathbf{y})^2/\mathbf{u}_i^T\mathbf{X}^T\mathbf{X}\mathbf{u}_i \tag{17}$$

under the constraint that the projections $\mathbf{X}\mathbf{u}_i$ are orthogonal to each other and have unit length, $\mathbf{u}_i^T\mathbf{X}^T\mathbf{X}\mathbf{u}_i = 1$ [53]. As it turns out, the projection directions $\mathbf{u}_i$ are the columns of the matrix $\mathbf{U}$ that contains the eigenvectors of $\mathbf{C} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}\mathbf{y}^T\mathbf{X}$. For regression, as in PCR, $\mathbf{X}$ is projected onto $\mathbf{U}$, and we work only with the projections. Thus, we minimize $||\mathbf{y} - \mathbf{X}\mathbf{U}\boldsymbol{\alpha}||^2$ with respect to $\boldsymbol{\alpha}$ and obtain

$$\boldsymbol{\beta} = \mathbf{U}\boldsymbol{\alpha} = \mathbf{U}(\mathbf{U}^T\mathbf{X}^T\mathbf{X}\mathbf{U})^{-1}\mathbf{U}^T\mathbf{X}^T\mathbf{y}. \tag{18}$$

In our case of a 1-dimensional output, we only have a single projection direction $\mathbf{u}$. Here, the matrix inversion in the $\boldsymbol{\beta}$ term is only a devision. The computational load, however, is in the computation of $\mathbf{C}$, which requires the inversion of $\mathbf{X}^T\mathbf{X}$ with complexity $O(d^3)$.

In the locally-weighted version, the eigenvectors are extracted from the matrix

$$C_w = (\mathbf{X}^T\mathbf{W}\mathbf{X})^{-1}\mathbf{X}^T\mathbf{W}\mathbf{y}\mathbf{y}^T\mathbf{W}\mathbf{X}. \tag{19}$$

In addition, the regression coefficients are computed as

$$\boldsymbol{\beta}_w = \mathbf{U}(\mathbf{U}^T\mathbf{X}^T\mathbf{W}\mathbf{X}\mathbf{U})^{-1}\mathbf{U}^T\mathbf{X}^T\mathbf{W}\mathbf{y}. \tag{20}$$

This method does not offer an incremental solution: all data points need to be accumulated for computing the projection matrices (compare with PLS—Sect. 4.6).

4.6 Partial Least Squares (PLS)

Partial least squares (PLS) is a regression technique extensively used in chemometrics [52,15]. However, it is less well known or less accepted in the statistical machine learning community because PLS is engineered rather than motivated by a statistical description of the training data. The algorithm starts by extracting a direction $\mathbf{u}_1$ in input space that highly correlates with the output. Then, the input is projected onto this direction, and the corresponding regression coefficient is computed. Further directions are obtained by deflation (see Algorithm 1).

PLS is purely algorithmic and not derived from an optimality criteria. However, up to several digits, PLS produces the same result as SIMPLS [10]; for one factor, the results are even identical. SIMPLS maximizes $(\mathbf{u}_i^T\mathbf{X}^T\mathbf{y})^2$ under the constraint that the projections $\mathbf{X}\mathbf{u}_i$ are orthogonal to each other, and that $\mathbf{u}_i^T\mathbf{u}_i = 1$. Since $(\mathbf{u}_i^T\mathbf{X}^T\mathbf{y})^2 = \mathrm{corr}^2(\mathbf{X}\mathbf{u}_i, \mathbf{y})\,\mathrm{var}(\mathbf{X}\mathbf{u}_i)$, PLS has therefore been considered as a mixture between RRR and PCR [1]. For spherically-distributed input data ($\mathrm{var}(\mathbf{X}\mathbf{u}_i) = \mathrm{const.}$), PLS produces the same result as RRR. Alternative methods can be constructed by tuning the objective function between $\mathrm{var}(\mathbf{X}\mathbf{u}_i)$ and $\mathrm{corr}^2(\mathbf{X}\mathbf{u}_i, \mathbf{y})$ [1]. However, these methods require an additional parameter and are thus not considered here.

Algorithm 1 Partial least squares

```
Training:
1: $\mathbf{X}_1 = \mathbf{X}$
2: $\mathbf{y}_1 = \mathbf{y}$
3: for $i = 1$ to $k$ do
4:    $\mathbf{u}_i = \mathbf{X}_i^T\mathbf{y}_i$
5:    $\mathbf{s}_i = \mathbf{X}_i\mathbf{u}_i$
6:    $\beta_i = \mathbf{s}_i^T\mathbf{y}_i/(\mathbf{s}_i^T\mathbf{s}_i)$
7:    $\mathbf{y}_{i+1} = \mathbf{y}_i - \beta_i\mathbf{s}_i$
8:    $\mathbf{p}_i = \mathbf{X}_i^T\mathbf{s}_i/(\mathbf{s}_i^T\mathbf{s}_i)$
9:    $\mathbf{X}_{i+1} = \mathbf{X}_i - \mathbf{s}_i\mathbf{p}_i^T$
10: end for
```

```
Look-up:
1: $\mathbf{x}_1 = \mathbf{x}$
2: $y_1 = 0$
3: for $i = 1$ to $k$ do
4:    $s_i = \mathbf{x}_i^T\mathbf{u}_i$
5:    $y_{i+1} = y_i + \beta_i s_i$
6:    $\mathbf{x}_{i+1} = \mathbf{x}_i - s_i\mathbf{p}_i$
7: end for
8: $y = y_{k+1}$
```

In the locally-weighted version, we need to do the following substitutions: $\mathbf{u}_i = \mathbf{X}^T\mathbf{W}\mathbf{y}_i$, $\beta_i = \mathbf{s}_i^T\mathbf{W}\mathbf{y}_i/(\mathbf{s}_i^T\mathbf{W}\mathbf{s}_i)$, and $\mathbf{p}_i = \mathbf{X}_i^T\mathbf{W}\mathbf{s}_i/(\mathbf{s}_i^T\mathbf{W}\mathbf{s}_i)$. The remaining equations in Algorithm 1 stay untouched.

Table 1 Comparison between the six tested dimensionality-reduction methods. In the complexity, n is the number of data points (either for training or testing), d their dimensionality, k the number of components, and M the number of EM steps

Dimensionality-reduction method	PCR	PCAJ	FA	PPCA	RRR	PLS
Modeling data variance	Yes	Yes	Yes	Yes	No	No
Joint input–output space	No	Yes	Yes	Yes	–	–
Maximize input–output correlation	No	No	No	No	Yes	Yes
Number of tuning parameters	0	0	0	0	0	0
Incremental version exists	Yes	Yes	No	Yes	No	Yes
Computational complexity (training)	$O(nd^2)$	$O(nd^2)$	$O(Mndk)$	$O(nd^2)$	$O(nd^2)$	$O(ndk)$
Computational complexity (incremental)	$O(ndk)$	$O(ndk)$	–	$O(ndk)$	–	$O(ndk)$
Computational complexity (testing)	$O(nd)$	$O(nd)$	$O(nd)$	$O(nd)$	$O(nd)$	$O(ndk)$

An incremental version of the training can be readily derived from Algorithm 1. The lines 1, 2, 5, 7, 9 on the left-hand side already contain the equations for single data points. For the remaining lines, an estimate of $\mathbf{u}_i$, $a_i = \mathbf{s}_i^T\mathbf{W}\mathbf{y}_i$, $b_i = \mathbf{s}_i^T\mathbf{W}\mathbf{s}_i$, and $\mathbf{q}_i = \mathbf{X}_i^T\mathbf{W}\mathbf{s}_i$ needs to be updated during each step, given a learning rate η. Algorithm 2 shows one iteration step for a new sample $(\mathbf{x}, y)$ with weight w.

Algorithm 2 Incremental partial least squares (locally weighted)

```
1: $\mathbf{x}_1 = \mathbf{x}$
2: $y_1 = y$
3: for $i = 1$ to $k$ do
4:    $s = \mathbf{u}_i(t)^T\mathbf{x}_i$
5:    $a_i(t+1) = a_i(t) + \eta(swy_i - a_i(t))$
6:    $b_i(t+1) = b_i(t) + \eta(sws - b_i(t))$
7:    $\mathbf{q}_i(t+1) = \mathbf{q}_i(t) + \eta(\mathbf{x}_i ws - \mathbf{q}_i(t))$
8:    $\mathbf{u}_i(t+1) = \mathbf{u}_i(t) + \eta(\mathbf{x}_i wy_i - \mathbf{u}_i(t))$
9:    $\beta_i = a_i(t+1)/b_i(t+1)$
10:   $y_{i+1} = y_i - \beta_i s$
11:   $\mathbf{p}_i = \mathbf{q}_i(t+1)/b_i(t+1)$
12:   $\mathbf{x}_{i+1} = \mathbf{x}_i - s\,\mathbf{p}_i$
13: end for
```

4.7 Summary

A brief summary of the presented dimensionality-reduction methods is presented in Table 1. This table further shows the computational complexity of each method, for training and testing, and, if available, for the incremental version of the algorithm. Only the dominant complexity term is shown, assuming $n > d > k$.

5 Experiments

The experiments are split in two parts. First, on a data set with known structure, we show how the dimensionality-reduction methods compare with each other for different factor numbers and kernel widths. Second, we demonstrate that key results observed in the synthetic data are also visible in two real-world data sets.

5.1 Synthetic Data

The synthetic data are sampled from an embedded manifold of either linear or non-linear structure as detailed in Sect. 5.1.1. Section 5.1.2 describes the evaluation settings. On the linear data, Sect. 5.1.3 compares the dimensionality-reduction methods for different number of components k and different noise settings. On the non-linear data, for locally-weighted regression, Sect. 5.1.4 compares the sensitivity of the various methods to the optimal kernel width for a range of projections k.

5.1.1 Data Generation

The synthetic data set was designed to assume the following four characteristics. First, the distribution of the input data is restricted to a q-dimensional subspace of arbitrary orientation to allow a comparison between the assumed number k of factors and the intrinsic dimensionality q. Second, the data are linearly transformed equivalent to duplicating data columns in $\mathbf{V}$ and rotating the resulting data points in the d-dimensional input space. Thus, this step introduces redundant dimensions. Third, the expected variance in each input and output dimension equals 1 (Appendix), which matches the standard data pre-processing of whitening the data—note, here, we do not whiten the entire data (which would result in artificially expanding the noise directions as well). Finally, the non-linear function was chosen such that locally around the origin, the same input-output relation holds as for the linear case.

We generated 5,000 training patterns and 10,000 test patterns. Each pattern consists of a d-dimensional input vector $\mathbf{x}$ and a 1-dimensional output y, which were generated using the following procedure:

A. Generate a q-dimensional vector $\mathbf{v}$ with entries distributed according to $\mathcal{N}(0, d/q)$. This variance leads to an expected variance of 1 for each dimension (Appendix). The vector $\mathbf{v}$ contains the ‘latent’ variables, and q is the intrinsic dimensionality.

B. Compute the input vector as $\mathbf{x} = \mathbf{M}\mathbf{v} + \boldsymbol{\epsilon}_x$. The matrix $\mathbf{M}$ maps $\mathbf{v}$ into a d-dimensional space, while preserving the distance relationships (thus, $\mathbf{M}^T\mathbf{M} = \mathbf{I}$). The entries of $\mathbf{M}$ were first chosen uniformly within the interval $[-1, 1]$. Then, the column vectors of $\mathbf{M}$ were orthonormalized using a Gram-Schmidt procedure. The vector $\boldsymbol{\epsilon}_x$ adds Gaussian noise.

C. Compute the output y directly from $\mathbf{v}$, either linearly, $y = \boldsymbol{\beta}^T\mathbf{v} + \epsilon_y$, or non-linearly, $y = \boldsymbol{\beta}^T \sin(\mathbf{v}) + \epsilon_y$ (to test the effect of the kernel width, Sect. 5.1.4), where ϵ_y adds Gaussian noise. The coefficients $\boldsymbol{\beta}$ were first chosen uniformly from the interval $[-1, 1]$. Then, $\boldsymbol{\beta}$ was scaled such that $E(y^2) = 1$ (see Appendix).

The dimensionality d was set to 10, and the intrinsic dimensionality q was set to 5. The noise was added only to the training patterns; the test patterns were noise free. Six different noise settings were chosen as detailed in the following:

(1) Low isotropic noise: $\forall i : p(\epsilon_x^i) = \mathcal{N}(0, 0.0001)$ and $p(\epsilon_y) = \mathcal{N}(0, 0.0001)$.
(2) High isotropic noise: $\forall i : p(\epsilon_x^i) = \mathcal{N}(0, 0.01)$ and $p(\epsilon_y) = \mathcal{N}(0, 0.01)$.
(3) Low output noise: $\forall i : \epsilon_x^i = 0$ and $p(\epsilon_y) = \mathcal{N}(0, 0.0001)$.
(4) High output noise: $\forall i : \epsilon_x^i = 0$ and $p(\epsilon_y) = \mathcal{N}(0, 0.01)$.
(5) Low output noise and irrelevant noise dimensions: same as model 3, but adding to $\mathbf{X}$ five columns filled with isotropic noise distributed according to $\mathcal{N}(0, 1)$.
(6) High output noise and irrelevant noise dimensions: analogous to model 5.

These noise settings were chosen to selectively match assumptions of the dimensionality-reduction methods (like PPCA assumes isotropic noise) and to explore typical conditions in applications: first, the input to a system is often controlled (low noise) and the corresponding output observed (noisy); second, for learning from sensory data, irrelevant noise dimensions relate to sensor values that are irrelevant for a given task.

5.1.2 Evaluation Settings

We tested the batch versions of the dimensionality-reduction methods, since batch versions are available for all methods. On the linear data with isotropic noise, we repeated the tests using the incremental versions of PCR, PCAJ, PPCA, and PLS.

The batch versions were slightly modified to avoid numerical instabilities. In the case of only output noise, the covariance matrix $\mathbf{X}^T\mathbf{X}$ is singular; it has rank q. Thus, each of the methods PCR, PCAJ, PPCA, PLS, and RRR will be numerically unstable for $k > q$. For RRR, this problem even holds for $k = 1$. To avoid that these numerical instabilities contaminate the results, the methods were slightly modified. In PCR, PPCA, and RRR, a small value (10^{-6}) was added to each diagonal element of the covariance matrix. Thus, here, RRR performs like ridge regression [20], which is ordinary least squares with this addition to the covariance matrix. We chose this addition to be small enough not to alter the results for $k \leq q$; and for $k > q$, it adds a small variance to the excess components $\mathbf{u}$ such that they are defined but negligible for the output y. In FA, 10^{-6} was added to the diagonal of the estimated noise variance $\boldsymbol{\Omega}$ to avoid divisions by too tiny values. For PLS, if $k > q$, $\mathbf{s}_i^T\mathbf{s}$ can get close to zero, and the algorithm gets unstable. This instability could be cured by setting $\beta_i = 0$ for $\mathbf{s}_i^T\mathbf{s}_i < 10^{-16}$, since the corresponding direction does not contribute to the y value. In PCAJ, for $k > q$, the first excess component points into the direction of the output; thus, the method fails since the principal subspace is orthogonal to the input space (this is a weakness of the method and not a numerical issue).

To evaluate the methods, data generation and regression were repeated for 100 runs. Normalized mean square errors (nMSE)—the mean prediction error divided by the variance of the output—were computed on the test set and averaged over all 100 runs. In each run, training and test set were the same for all methods and for all k and D values. Apart from FA, all methods have analytic solutions in the batch version. For FA, 1,000 expectation–maximization steps were iterated. RRR was evaluated only for $k = 1$ since we have only one non-zero eigenvalue. In the plots, however, we replicate the corresponding result for all k values to ease the comparison with the other methods.

In the incremental versions of PCR, PCAJ, and PPCA, we used the robust recursive least square algorithm [34] to extract the principal components incrementally. We implemented this algorithm as described in [22]. Since the orthonormality of the principal components slowly degrades, we did a Gram-Schmidt orthonormalization of the eigenvectors after each 200 learning steps. For PLS, we used the algorithm as shown in Sect. 4.6 (here, $w = 1$). Initially, the vectors $\mathbf{u}_i$ were set to random values and, then, orthonormalized. Similar to above, we set $\beta_i = 0$ whenever $s^2 < 10^{-16}$. For all incremental algorithms, we set the learning rate to $\eta = 1/t$, where t is the iteration step. This choice counterbalances forgetting and update of the estimate such that all data points have statistically the same weight (see, e.g., Sect. A.3 in [21]—note, this choice might be undesired in an incremental setting in which older data samples should be forgotten).

In the non-linear case, the data points were weighted according to (4). The center $\mathbf{c}$ was set to zero, and the distance metric $\mathbf{D}$ was set to $D\mathbf{I}_d$, where $\mathbf{I}_d$ is the d-dimensional identity matrix. For testing, the errors for each test point were multiplied with the same weight function used for training.

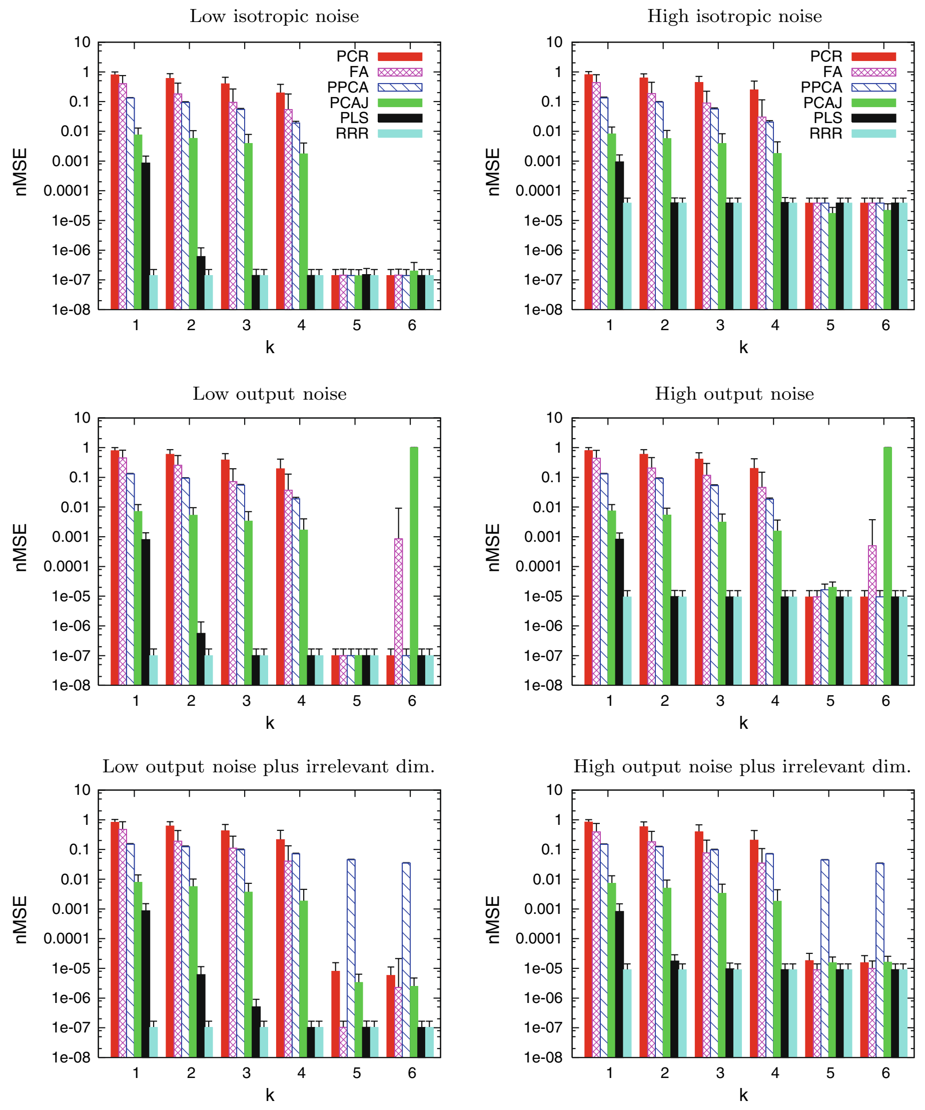

Fig. 3 Normalized mean square errors (nMSE) depending on the number of factors k. Mean values and standard deviations are shown for the six noise conditions

5.1.3 Comparison of Dimensionality Reduction Methods

On the linear data, Fig. 3 compares the dimensionality reduction methods for different number of components k and different noise settings. For the case without irrelevant noise dimension, if k matches the intrinsic dimensionality q, all six methods do almost equally well.

For $k > q$, PCAJ fails for only output noise, and for other noise settings, the prediction error of PCAJ increases with increasing k (not shown in the figure). The other methods are not impaired by using too many components or factors, apart from FA for zero noise in the input; here, FA's performance fluctuates highly.

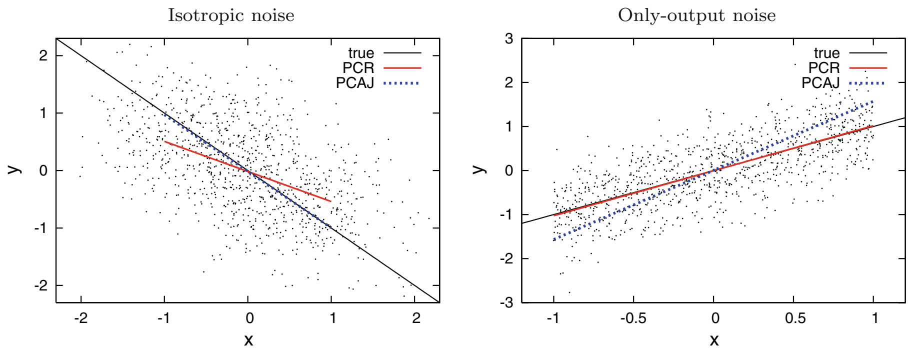

Fig. 4 Impact of the noise generation model on regression, comparing isotropic noise with noise only in the output. Here, for illustrative purpose, the latent variable v was uniformly distributed instead of Gaussian

For $k < q$, PCR, FA, PPCA, and PCAJ generate large errors. PCR does worst, because it omits a component in input space that contributes to the output value. Second worst is FA, whose model assumptions are violated. On the other hand, PLS and RRR, which consider the correlation between input and output, do much better than the remaining methods: RRR is already optimal with only one projection direction, and PLS is almost optimal with only two projection directions.

When adding irrelevant noise dimensions, PLS, RRR, and FA are almost unaffected. However, PCR, PCAJ, and PPCA, which model the variance, get distorted since the data have significant variance in the irrelevant dimensions. Worst affected is PPCA, which essentially models the data by wrapping an ellipsoid around it.

For isotropic noise and $k = q$, PCAJ is slightly better than the other methods, and for only output noise, PCAJ is slightly worse than all other methods. This difference is more distinct for higher noise. To illustrate this result, we compare PCAJ and PCR for $d = 1$ (Fig. 4; in this simple case, PCR, FA, PPCA, PLS, and RRR all yield the same result). With isotropic noise (Fig. 4 (left)), PCAJ produces the correct result (nMSE: $7.5 * 10^{-4} \pm 7.4 * 10^{-4}$), which differs from the least-squares solution, and thus, PCR fails (nMSE: 0.22 ± 0.01). For only output noise (Fig. 4 (right)), the ordinary least-squares solution is optimal (nMSE: $3.7 * 10^{-4} \pm 3.2 * 10^{-4}$), but, here, PCAJ fails (nMSE: 0.30 ± 0.02) because the direction of maximal variance is rotated into the direction of the output noise.

The incremental implementations result in a higher error rate (Fig. 5). However, we see the same pattern as observed in the batch algorithms: for $k < q$, PLS is doing better than PCR, PCAJ, and PPCA, and is even better than the batch versions of these algorithms (compare Fig. 3 with 5).

5.1.4 Locally-Weighted Regression

In this section, we evaluate locally weighted regression using the various dimensionality reduction techniques on the non-linear data set. Figure 6 compares the dimensionality reduction methods for various values of the locality measure D—see (4)—and two k-values: $k = 4$ and $k = 5$. The results are only shown for low output noise (setting 3 in Sect. 5.1.1). If k equals the intrinsic dimensionality ($q = 5$), all methods show about the same performance. The prediction error is optimal for a specific kernel width, here, $D = 12$. For $k = 4$, only PLS has the same optimal D-value and performance, which also equal the RRR results. The

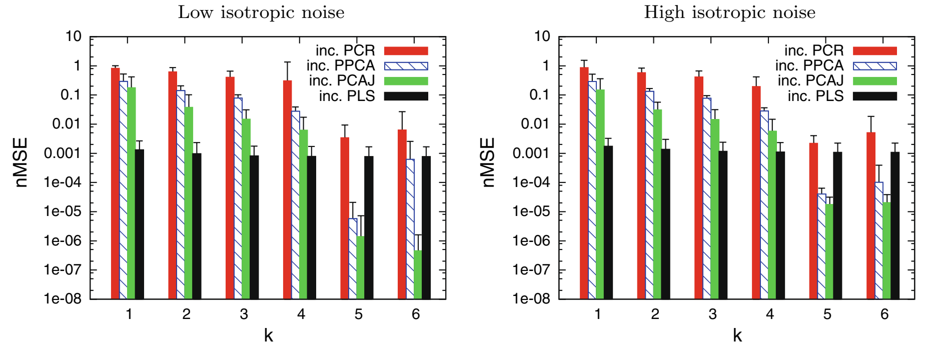

Fig. 5 Normalized mean square errors (nMSE) depending on the number of factors k. Mean values and standard deviations are shown for the incremental versions of the tested dimensionality reduction methods

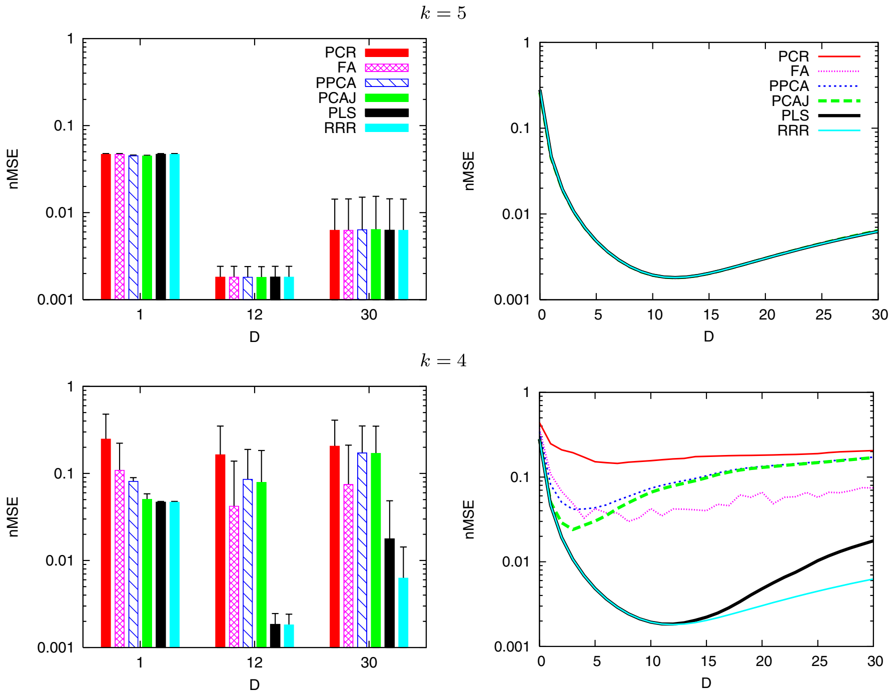

Fig. 6 Prediction errors (nMSE) depending on the parameter D (the inverse of the squared width of the weight function) for optimal ($k = 5$) and sub-optimal ($k = 4$) number of factors

methods PPCA and PCAJ have an optimal D that is shifted to a much lower value ($D = 3$), and PCR and FA's optimal D-value is blurred by the variance in the prediction error. For $k = 6$, the results are the same as for $k = 5$ for all methods except PCAJ (not shown in the figure); PCAJ fails as observed above (see Fig. 3). For $k = 1$, the optimal D-value shifts even for PLS to smaller values, that is, wider kernels (not shown in the figure).

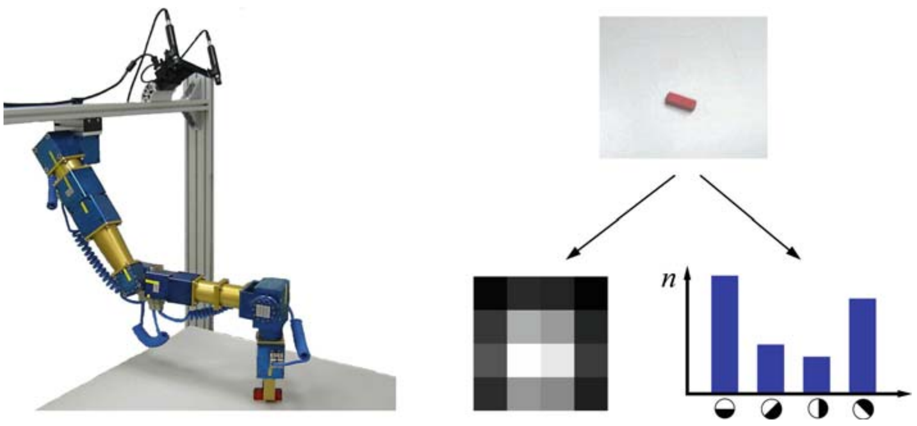

Fig. 7 Robot setup used to collect the vision-robot data (Left) and sample camera image with coarse-grained view and edge histogram (Right)

5.2 Real-World Data

The dimensionality-reduction methods were further tested on real-world data to demonstrate that the trends shown in previous results are also relevant to non-synthetic data and hence, can be used for real-world application.

5.2.1 Methods

Two data sets were used: vision-robot and census-house data. The vision-robot data were taken from a study in which a six-degrees-of-freedom robot arm learned to grasp an object (a small brick) presented visually on a table surface [23]—see Fig. 7. Each data point combines the visual information of the object with the grasping arm posture of the robot. To collect a data point, the robot put the brick on the table, recorded the corresponding arm posture, removed the arm, and took a picture of the brick. The visual information consists of a 4×4-pixels grid providing a coarse-grained view of the table surface and a histogram showing the edge distribution over four orientations within the camera image [23]. The arm posture consists of six joint angles.

The vision-robot data set is locally low-dimensional, because the brick on the table has only three degrees of freedom (two for position and one for orientation). Thus, there is a lot of redundancy in the sensor values (the coarse-grained image and the edge histogram). A disadvantage of this data set for our regression study is that several redundant arm postures exist for a given image. For a given end-effector position, the inverse kinematics of the robot arm has several solutions; i.e., several combinations of joint angles lead to the same end-effector position [25,30]. However, in the collected data, the shoulder joint showed almost no redundancy (three redundant postures could be identified and were removed). Thus, we used the angle of the shoulder joint as a target value for the 20-dimensional sensory input. Since the orientation of the brick is irrelevant for the angle of the shoulder joint, the intrinsic dimensionality of the relevant data is only two. The data set consists of 3,368 patterns; the first 2,000 were used for training, the rest for testing.

The census-house data set [27] is part of the Delve repository. The data show median house prices and demographic compositions of several survey regions, as obtained from the 1990

US Census. Here, we used a subset called ‘house-price-16L’, that contains only 16 of the demographic attributes. These attributes form the input and the corresponding house price is the output. Also this data set contains redundancy, since some of the attributes correlate with each other, for example, the number of families and the number of households. However, the intrinsic dimensionality is unknown. The pattern set consists of 22,784 data points; thereof 10,000 were used for training and the rest for testing.

Both training data sets were processed such that each attribute had zero mean and unit variance. For regression, as for the sine-function before, we used only one local model centered at the origin. The weight-function was uniform and had the same metric parameter D as before. The optimal D-value was obtained by, first, computing the regression using RRR with $k = 1$ for various D-values (in steps of 0.1), and second, choosing the D-value resulting in the lowest error ($D = 2.4$ for the vision-robot and $D = 0.3$ for the house data). Using this optimal D, the six methods were evaluated with varying number of factors k.

5.2.2 Results

For both data sets, the results show many of the characteristics as seen in Sect. 5.1.3 for the synthetic data (Fig. 8). RRR provides a kind of base-line for the optimal performance. Among the remaining methods, PLS’ regression error decreases the most quickly with increasing number of factors. FA, as in the case with irrelevant noise dimensions, converges more quickly than the remaining methods PCR, PCAJ, and PPCA. On the vision-robot data, the prediction error for FA drops sharply at the intrinsic dimensionality ($k = 2$). The error for PCR, PCAJ, and PPCA drops at a higher k-value ($k = 4$). When increasing k further, only PCAJ fails.

6 Discussion

We compared the regression performance of non-parametric dimensionality-reduction methods on synthetic and real-world data. The synthetic data were constructed such that they were restricted to a lower q-dimensional subspace. Our basic finding was that if the number k of factors, components, or projection directions is smaller than q then methods that are based on maximizing the correlation between projection directions and output (PLS and RRR) do better than methods that model the distribution of data in joint space (FA, PPCA, and PCAJ), which in turn do better than methods that reduce the dimensionality without taking the regression target into account (PCR). This finding is significant in view of the practical requirements of incrementally adding projection directions in a real-world application without significant prior knowledge of the true intrinsic dimensionality.

The data generation mechanism fulfilled all assumptions of factor analysis except for the true intrinsic dimensionality q. Thus, the breakdown of FA for incorrect number of factors demonstrates its brittleness with respect to the match between k and q; making it inherently difficult to use for a constructive, incremental algorithm. In contrast, PLS shows promising results with graceful degradation for $k < q$.

As the comparison with RRR shows, the dimensionality could be even reduced to *one* by finding a direction whose projections maximally correlate with the output—this is nothing but the direction of the true regression vector β. However, as mentioned earlier, computationally, RRR is equivalent to doing ordinary least squares on the full data set, i.e., the full covariance matrix needs to be inverted, and there are no computationally-attractive incremental implementations. Thus, PLS appears as the best compromise: the method maximizes

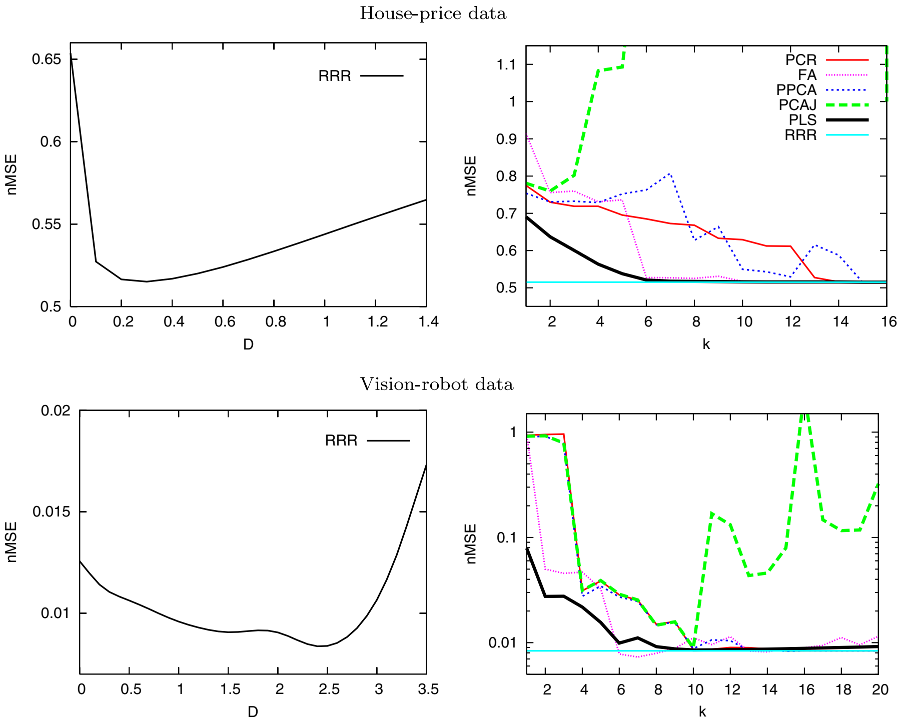

Fig. 8 Regression errors on the house-price and the vision-robot data. (Left) Dependence on the metric parameter D, using RRR with $k = 1$. (Right) For optimal D, dependence on the number of factors comparing different methods

only the correlation in the special case of spherically-distributed input data, but it provides an efficient implementation, which can be also written in an incremental way.

In a further test, we showed that this advantage of PLS for $k < q$ actually holds also in the incremental version of the algorithm. The lower performance of the incremental methods for $k > 1$ is probably due to simultaneous component estimation, which is known to accumulate error in incremental PCA [29]. For $k \geq q$, PCAJ did best. In our incremental PCAJ implementation, inconsistencies between the incrementally updated variables are probably compensated by the Gram-Schmidt orthonormalization, which was computed every 200 steps (see Sect. 5.1.2). For the other methods, this compensation probably did not work as well, because additional variables have to be updated (β for PCR and the residual variance for PPCA). Moreoever, our PLS implementation did not use any such compensation step.

Apart from the breakdown for $k < q$, all of PCAJ, PCR, FA, and PPCA showed an additional specific weakness. PCAJ deteriorates for $k > q$; particularly, for only output noise, one principal component points into the direction of this noise, and regression cannot proceed because the principal subspace is perpendicular to the input space. PCR fails dramatically if $k < q$ because part of the input that contributes to the output value is ignored. FA shows higher regression errors for $k > q$ if input dimensions have zero noise, probably, because the model does not consider zero noise variance.

PPCA fails if we add irrelevant noise dimensions because the ellipsoid associated with the eigenvectors and eigenvalues includes this noise (see Sect. 4.4). Thus, to compensate for this additional noise, as many components as noise dimensions need to be added [21]. PCR and

PCAJ do better with irrelevant noise because the data's variance equals d/q on the embedded subspace; the variance in the noise dimensions equals 1 (see Sect. 5.1.1). Here, FA was not affected, because the irrelevant noise dimensions are covered by the model assumptions. This feature may explain why FA did better than PCAJ, PCR, and PPCA on the real-world data.

Furthermore, we illustrated the influence of the noise-generation model on the regression results. Apart from PCAJ, all tested methods produce an optimal regression result for only output noise, but not for isotropic noise. The model assumptions of FA and PPCA are consistent with isotropic noise, and these methods do reproduce the underlying model correctly—the density $p(y, \mathbf{x})$ matches the distribution of the data in joint space. However, maximizing $p(y|\mathbf{x})$ for a given $\mathbf{x}$ results in the ordinary-least-squares solution, which is not optimal for noise-free test data. PCAJ also aligns its principal subspace with the data distribution for isotropic noise, but PCAJ uses this subspace directly as the result for regression, which is optimal for noise-free test data.

On non-linear data sets, with locally-linear regression, the prediction error depends on the width of the kernel function. An optimal width exists, which is a trade-off between finding a small enough value for a good linear approximation and a large enough value to avoid over-fitting (because of the noise). In incremental learning, when the number of components q changes, a robustness of the optimal width about q is desirable. This robustness was only found in PLS (see Fig. 6).

Using the real-world data, we could reproduce the main findings obtained from the synthetic data, particularly, the advantage of maximizing the correlation versus maximizing the variance. Thus, our synthetic data seems to have basic characteristics found in real applications: being lower-dimensional than the embedding space, including dimensions with irrelevant noise, being globally non-linear, and locally linear. In robotics, such characteristics are likely to occur in sensory data, for example, in visual and tactile input.

7 Conclusions

For linear regression, dimensionality-reduction methods that maximize the correlation between projection directions and output data do better than methods that model only the variance of the data distribution. The drawback of these latter methods is that they need to assume that the number of factors or components is at least as high as the intrinsic dimensionality of the data—this makes incremental addition of components problematic. Ideally, one projection direction would be sufficient, namely the direction in input space that maximally correlates with the output. The computation of this direction, however, involves calculations which are at least as expensive as the ordinary weighted least squares on the full-dimensional data set.

Fortunately, a locally weighted, incremental reformulation of partial least squares (PLS) provides an ideal dimensionality-reduction technique: PLS does well with only a few projection directions (it works with only one projection in the special case of spherically distributed input data), and since the projection directions are orthogonal, additional relevant dimensions can be added without relearning existing projection directions, e.g., as in use with online regression algorithms like locally-weighted projection regression [48].

Acknowledgements This work was funded in part by the SENSOPAC project. SENSOPAC is supported by the European Commission through the Sixth Framework Program for Research and Development. Part of this manuscript was written while H.H. was at the University of Southern California and funded by DFG grant HO 3887/1-1. We are grateful to the anonymous reviewers for their helpful comments.

Appendix

Variance of the generated data

We chose the generation of the synthetic data such that they have an expected variance of 1 in each dimension; this is not to be confused with whitening of the entire data set, i.e., normalizing to a unit sphere. The data are generated according to $\mathbf{x} = \mathbf{M}\mathbf{v}$ and $y = \boldsymbol{\beta}^T\mathbf{v}$. The latent variables v_i have the variance d/q, and the regression coefficients $\boldsymbol{\beta}$ are normalized such that $||\boldsymbol{\beta}||^2 = q/d$. Thus, the average variance of x_i is 1:

$$\frac{1}{d}\sum_{i=1}^{d} E\left(x_i^2\right) = \frac{1}{d}E\left(\sum_{i=1}^{d} x_i^2\right) = \frac{1}{d}E\left(\mathbf{x}^T\mathbf{x}\right) = \frac{1}{d}E\left(\mathbf{v}^T\mathbf{v}\right) = \frac{1}{d}\sum_{i=1}^{q} E\left(v_i^2\right) = 1. \tag{21}$$

Here, we used $\mathbf{M}^T\mathbf{M} = \mathbf{I}$ and the linearity of the expectation value E. Since the expected variance of x_i across all training runs is the same for all i, $E\left(x_i^2\right)$ matches the above averaged value. Finally, the variance of y is also 1:

$$E\left(y^2\right) = \sum_{i,j} E\left(\beta_i\beta_j v_i v_j\right) = \sum_i \beta_i^2 E\left(v_i^2\right) = ||\boldsymbol{\beta}||^2\, d/q = 1. \tag{22}$$

References

1. Abraham B, Merola G (2005) Dimensionality reduction approach to multivariate prediction. Comput Stat Data Anal 48:5–16
2. Atkeson CG, Moore AW, Schaal S (1997a) Locally weighted learning. Artif Intell Rev 11:11–73
3. Atkeson CG, Moore AW, Schaal S (1997b) Locally weighted learning for control. Artif Intell Rev 11:75–113
4. Belkin M, Niyogi P (2003) Laplacian eigenmaps for dimensionality reduction and data representation. Neural Comp 15(6):1373–1396
5. Bell A, Sejnowski T (1997) The independent components of natural scenes are edge filters. Vis Res 37(23):3327–3338
6. Bellman RE (1961) Adaptive control processes: a guided tour. Princeton University Press, Princeton, NJ
7. Beyer K, Goldstein J, Ramakrishnan R, Shaft U (1999) When is ‘nearest neighbor’ meaningful? In: Proceedings of 7th international conference on database theory, Jerusalem, Israel, pp 217–235
8. Bishop CM (2006) Pattern recognition and machine learning. Springer, New York
9. Cressie N (1993) Statistics for spatial data. Wiley, New York
10. de Jong S (1993) SIMPLS: an alternative approach to partial least squares regression. Chemometr Intell Lab Syst 18:251–263
11. Diamantaras KI, Kung SY (1996) Principal component neural networks. Wiley, New York
12. D’Souza A, Vijayakumar S, Schaal S (2001) Are internal models of the entire body learnable? Society for Neuroscience, Abstracts
13. Everitt BS (1984) An introduction to latent variable models. Chapman and Hall, London
14. Fan J, Gijbels I (1996) Local polynomial modeling and its applications. Chapman and Hall, London, UK
15. Frank IE, Friedman JH (1993) A statistical view of some chemometrics regression tools. Technometrics 35(2):109–135
16. Geweke J (1996) Bayesian reduced rank regression in econometrics. J Econometr 75(1):121–146
17. Ghahramani Z, Beal M (2000) Variational inference for bayesian mixtures of factor analysers. In: Solla S, Leen T, Müller KR (eds) Advances in neural information processing systems, vol 12. MIT Press, Cambridge, MA, pp 449–455
18. Ghahramani Z, Hinton GE (1997) The EM algorithm for mixtures of factor analyzers. Tech. Rep. CRG-TR-96-1. Department of Computer Science, University of Toronto, Canada
19. Hinton G, Roweis ST (2003) Stochastic neighbor embedding. In: Becker S, Thrun S, Obermayer K (eds) Advances in neural information processing systems, vol 15. MIT Press, Cambridge, MA, pp 857–864

20. Hoerl AE, Kennard RW (1970) Ridge regression: biased estimation for nonorthogonal problems. Technometrics 12(1):55–67
21. Hoffmann H (2005) Unsupervised learning of visuomotor associations, MPI series in biological cybernetics, vol 11. Logos Verlag Berlin, PhD thesis (2004), Bielefeld University, Germany
22. Hoffmann H, Möller R (2003) Unsupervised learning of a kinematic arm model. In: Kaynak O, Alpaydin E, Oja E, Xu L (eds) Artificial neural networks and neural information processing—ICANN/ICONIP 2003, LNCS, vol 2714. Springer, Berlin, pp 463–470
23. Hoffmann H, Schenck W, Möller R (2005) Learning visuomotor transformations for gaze-control and grasping. Biol Cybernetics 93(2):119–130
24. Izenman AJ (1975) Reduced-rank regression for the multivariate linear model. J Multivar Anal 5(2):248–264
25. Jordan MI, Rumelhart DE (1992) Forward models: supervised learning with a distal teacher. Cogn Sci 16:307–354
26. Kawato M (1999) Internal models for motor control and trajectory planning. Curr Opin Neurobiol 9:718–727
27. Kustra R (1996) Delve census-house dataset. Available at http://www.cs.toronto.edu/~delve/data/datasets.html
28. Matheron G (1963) Principles of geostatistics. Econ Geol 58(8):1246–1266
29. Möller R (2002) Interlocking of learning and orthonormalization in RRLSA. Neurocomputing 49:429–433
30. Movellan JR, McClelland JL (1993) Learning continuous probability distributions with symmetric diffusion networks. Cogn Sci 17:463–496
31. Oja E (1982) A simplified neuron model as a principal component analyzer. J Math Biol 15(3):267–273
32. Oja E (1989) Neural networks, principle components, and subspaces. Int J Neural Syst 1(1):61–68
33. Olshausen BA, Field DJ (1996) Emergence of simple-cell receptive field properties by learning a sparse code for natural images. Nature 381:607–609
34. Ouyang S, Bao Z, Liao GS (2000) Robust recursive least squares learning algorithm for principal component analysis. IEEE Trans Neural Netw 11(1):215–221
35. Rasmussen CE, Williams CKI (2006) Gaussian processes for machine learning. MIT Press, Cambridge, MA
36. Roweis ST, Saul LK (2000) Nonlinear dimensionality reduction by locally linear embedding. Science 290:2323–2326
37. Sanger TD (1989) Optimal unsupervised learning in a single-layer linear feedforward neural network. Neural Netw 2:459–473
38. Schaal S, Sternad D (2001) Origins and violations of the 2/3 power law in rhythmic 3d movements. Exp Brain Res 136:60–72
39. Schaal S, Vijayakumar S, Atkeson CG (1998) Local dimensionality reduction. In: Jordan M, Kearns M, Solla S (eds) Advances in neural information processing systems, vol 10. MIT Press, Cambridge, MA, pp 633–639
40. Schölkopf B, Smola AJ (2002) Learning with kernels. MIT Press, Cambridge, MA
41. Tenenbaum JB, de Silva V, Langford JC (2000) A global geometric framework for nonlinear dimensionality reduction. Science 290:2319–2323
42. Tipping ME (2001) Sparse bayesian learning and the relevance vector machine. J Mach Learn Res 1:211–244
43. Tipping ME, Bishop CM (1999) Probabilistic principal component analysis. J Roy Stat Soc Ser B 61(3):611–622
44. Vapnik VN (1995) The nature of statistical learning theory. Springer-Verlag, New York
45. Vijayakumar S, Schaal S (2000a) Fast and efficient incremental learning for high-dimensional movement systems. In: Proceedings of the international conference on robotics and automation, San Francisco, CA
46. Vijayakumar S, Schaal S (2000b) Locally weighted projection regression: an O(n) algorithm for incremental real time learning in high dimensional space. In: Proceedings of the 17th international conference on machine learning, Montreal, Canada, pp 1079–1086
47. Vijayakumar S, D'Souza A, Shibata T, Conradt J, Schaal S (2002) Statistical learning for humanoid robots. Auton Robots 12(1):55–69
48. Vijayakumar S, D'Souza A, Schaal S (2005) Incremental online learning in high dimensions. Neural Comput 17:2602–2634
49. Vlassis N, Motomura Y, Kröse B (2002) Supervised dimension reduction of intrinsically low-dimensional data. Neural Comput 14:191–215
50. Webster JT, Gunst RF, Mason RL (1974) Latent root regression analysis. Technometrics 16(4):513–522

51. Weinberger KQ, Sha F, Saul LK (2004) Learning a kernel matrix for nonlinear dimensionality reduction. In: Proceedings of the 21st international conference on machine learning, Washington, DC
52. Wold S, Ruhe A, Wold H, Dunn III WJ (1984) The collinearity problem in linear regression. The partial least squares (PLS) approach to generalized inverses. SIAM J Sci Stat Comput 5(3):735–743
53. van den Wollenberg AL (1977) Redundancy analysis an alternative for canonical correlation analysis. Psychometrika 42(2):207–219