Neurocomputing 131 (2014) 427–439

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Probabilistic local reconstruction for *k*-NN regression and its application to virtual metrology in semiconductor manufacturing

Seung-kyung Lee [a], Pilsung Kang [b], Sungzoon Cho [a,*]

[a] Seoul National University, 1 Gwanakro, Gwanak-gu, 151-744 Seoul, Republic of Korea
[b] Seoul National University of Science & Technology, 232 Gongneung ro, Nowon-gu, 139-743 Seoul, Republic of Korea

ARTICLE INFO

Article history:
Received 4 October 2012
Received in revised form
1 October 2013
Accepted 12 October 2013
Communicated by Dr. T. Heskes
Available online 23 October 2013

Keywords:
Locally linear reconstruction
k-NN regression
Bayesian kernel model

ABSTRACT

The "*locally linear reconstruction*" (LLR) provides a principled and *k*-insensitive way to determine the weights of *k*-nearest neighbor (*k*-NN) learning. LLR, however, does not provide a confidence interval for the *k* neighbors-based reconstruction of a query point, which is required in many real application domains. Moreover, its fixed linear structure makes the local reconstruction model unstable, resulting in performance fluctuation for regressions under different *k* values. Therefore, we propose a probabilistic local reconstruction (PLR) as an extended version of LLR in the *k*-NN regression. First, we probabilistically capture the reconstruction uncertainty by incorporating Gaussian regularization prior into the reconstruction model. This prevents over-fitting when there are no informative neighbors in the local reconstruction. We then project data into a higher dimensional feature space to capture the nonlinear relationship between neighbors and a query point when a value of *k* is large. Preliminary experimental results demonstrated that the proposed Bayesian kernel treatment improves accuracy and *k*-invariance. Moreover, from the experiment on a real virtual metrology data set in the semiconductor manufacturing, it was found that the uncertainty information on the prediction outcomes provided by PLR supports more appropriate decision making.

1. Introduction

k-Nearest neighbor (*k*-NN) learning is the most popular realization of the instance-based learning approach. Storing *N* training pairs, $\{\mathbf{x}_n \in \mathbb{R}^D, y_n\}_{n=1}^N$, each consisting of an input point (or vector) and an output value (or target value), *k*-NN learning defers learning or training until a query point $\mathbf{x}$ is given. *k*-NN learning then predicts the query's target value $y(\mathbf{x})$ by combining those of most similar *k* neighbors among the training pairs. Due to its algorithmic simplicity and high prediction performance, comparable to those of more complex models such as artificial neural networks (ANN) or support vector machines (SVM), *k*-NN learning has been successfully applied to various applications such as manufacturing [1,2], bioinformatics [3,4], time series analysis [5,6], collaborative filtering [7], and image processing [8–10].

In the regression problem whose target variable is continuous, $y \in \mathbb{R}$, the predictive function of the k-NN learning is commonly defined as a weighted sum of the target values $y_{NN(\mathbf{x},i)}$ of *k* nearest neighbors $\mathbf{x}_{NN(\mathbf{x},i)}$, $i = 1, \ldots, k$, as follows:

$$y(\mathbf{x}) = \sum_{i=1}^{k} w_{NN(\mathbf{x},i)} y_{NN(\mathbf{x},i)} = \mathbf{w}_{NN(\mathbf{x})}^T \mathbf{y}_{NN(\mathbf{x})}, \quad (1)$$

where $NN(\mathbf{x}, i)$ is the index set of the *i*th nearest neighbor of a query point $\mathbf{x}$, chosen among the training points $\{\mathbf{x}_n\}_{n=1}^N$. This weighted *k*-NN learning [11,12] has two user-specific parameters to be determined: first, the number of nearest neighbors, *k*, and second, the weights assigned to the selected neighbors, $\mathbf{w}_{NN(\mathbf{x})} = [w_{NN(\mathbf{x},1)}, \ldots, w_{NN(\mathbf{x},k)}]^T$. Specially, the performance of *k*-NN learning mainly depends on the locality control parameter, *k*, which is usually chosen by empirical cross-validation or by domain experts in practice [13].

Recently, [13] proposed locally linear reconstruction (LLR) as a principled and *k*-invariant weight kernel for *k*-NN learning. While conventional distance-based weight kernels such as inversion kernel [14,8], exponential kernel [15], and Gaussian kernel [16] under-utilize available information in *k*-nearest neighbors [13,17], LLR takes into account the "locally linear topology" around a query point, assuming that the query point can be linearly reconstructed (described) by its *k* nearest neighbors in the input space. That is, the LLR weights for *k* neighbors are collectively determined, based on their proportional contributions to the reconstruction of the query point. Compared with the distance-based kernels, the LLR tends to identify the best subset of relevant neighbors for

* Corresponding author. Tel.: +82 1 7203 6275.
E-mail addresses: sklee83@snu.ac.kr (S.-k. Lee), pskang@seoultech.ac.kr (P. Kang), zoon@snu.ac.kr (S. Cho).

0925-2312/$ - see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.neucom.2013.10.001

sufficiently large k values. Kang and Cho [13] empirically showed that the significant advantage of LLR gives rise to a prediction performance more accurate and robust to k, i.e., k-invariant than the distance-based kernels in both k-NN classifications and regressions.

The locally linear reconstruction, however, has some disadvantages as follows. First, in practice, neighbors for a query point are often not informative to describe the query. Thus, in order to providing an answer for the question, "how informative is the set of k neighbors in LLR?", it is desirable to provide a probabilistic estimate, but LLR provides a point estimate. Second, the fixed linear structure makes the local reconstruction less flexible, resulting in performance fluctuation for some regressions under different k values. A small k with only a handful of neighbors may not guide LLR correctly. With a large k, the non-linear neighborhood structure embedded in too redundant neighbors cannot be captured.

In this paper, we propose a Bayesian kernel model of LLR, named probabilistic local reconstruction (PLR) in order to overcome the limitations of LLR in k-NN regression. First, we employ an explicit probabilistic model for the local reconstruction in order to capture the reconstruction uncertainty as a form of "error-bars". Under the probabilistic viewpoint, incorporating Gaussian prior into the reconstruction allows one to explain unweighted or equally weighted k-NN prediction, i.e., contributions of all neighbors in the prediction are equal, and avoid over-fitting when there are no informative neighbors in the local reconstruction. We then adopt a kernel trick, which transforms a lower-dimensional input space into a higher-dimensional feature space without the need for explicit mapping function. It helps the model capture the non-linear neighborhood structure in a large k setting without suffering from under-fitting, and be robust to a wider range of k values. Consequently, as our main contribution, we demonstrate that the Bayesian kernel treatment in the local reconstruction problem improves the k-invariant property as well as prediction accuracy in the kernelized k-NN regression.

The rest of this paper is structured as follows. In Section 2, we reviewed the LLR kernel for k-NN learning. In Section 3, we derived our proposed model, PLR. In Section 4, with experiments on benchmark data sets, we demonstrate the usefulness of PLR in a real-world application, Virtual Metrology (VM) in the semiconductor manufacturing. In Section 5, with a conclusion, we discuss future work.

2. Related work

LLR is originated from a local manifold learning method, locally linear embedding (LLE) for dimensionality reduction on non-linear manifolds [18]. The weight kernel derived from LLE for k-NN learning is based on the reconstruction weight assigned to a neighbor $\mathbf{x}_{NN(\mathbf{x},i)}$ when reconstructing a query $\mathbf{x}$ in terms of its corresponding k neighbors. It is the reason that the kernel is named as locally linear reconstruction (LLR) [13].

With the assumption that a query in the input space can be described by a linear combination of its neighbors, the local reconstruction is the problem to capture the underlying combination or topology to minimize the difference between the query and its description. Kang and Cho [13] simply assumed that a given query $\mathbf{x}$ can be linearly reconstructed by its k nearest neighbors, and used its "locally linear topology" as the weights for the neighbors in k-NN learning.

Let $\mathbf{x}_{NN(\mathbf{x},i)}$ denote the ith nearest neighbor of a query point, and $\mathbf{X}_{NN(\mathbf{x})}=[\mathbf{x}_{NN(\mathbf{x},1)},\ldots,\mathbf{x}_{NN(\mathbf{x},k)}]$ denote the "neighbor" matrix where the ith column vector is the ith neighbor of the query point $\mathbf{x}$. Then, we can define the description, i.e., reconstructed point $\mathbf{x}_*$ of the query $\mathbf{x}$ as follows:

$$\mathbf{x}_*=\sum_{i=1}^{k} w_{NN(\mathbf{x},i)}\mathbf{x}_{NN(\mathbf{x},i)}=\mathbf{X}_{NN}\mathbf{w}, \tag{2}$$

where the weight vector $\mathbf{w}=\mathbf{w}_{NN(\mathbf{x})}=[w_{NN(\mathbf{x},1)},\ldots,w_{NN(\mathbf{x},k)}]^T$ is the linear coefficient vector characterizing the local geometry in the neighborhood of the query, and is constrained to have the sum of 1, $\sum_i w_{NN(\mathbf{x},i)}=\mathbf{w}^T\mathbf{1}_{k\times 1}=1$, where $\mathbf{1}_{k\times 1}$ is the $k\times 1$ column vector with every element being one [18]. Geographically, the reconstructed point $\mathbf{x}_*$ is interpreted as the projected point on the neighbor space, i.e., hyper-plane spanned by the k neighbors [18,13], and obtained by minimizing the reconstruction error $E(\mathbf{w};\mathbf{x})$ as follows:

$$\begin{aligned}E(\mathbf{w};\mathbf{x})&=\tfrac{1}{2}(\mathbf{x}-\mathbf{X}_{NN(\mathbf{x})}\mathbf{w})^T(\mathbf{x}-\mathbf{X}_{NN(\mathbf{x})}\mathbf{w})\\&=\tfrac{1}{2}\mathbf{x}^T\mathbf{x}-\mathbf{w}^T(\mathbf{X}_{NN(\mathbf{x})}^T\mathbf{x})+\tfrac{1}{2}\mathbf{w}^T\mathbf{C}\mathbf{w},\end{aligned} \tag{3}$$

where $\mathbf{C}=\mathbf{X}_{NN(\mathbf{x})}^T\mathbf{X}_{NN(\mathbf{x})}$ is a k-by-k local gram matrix. The explicit solution $\mathbf{w}$ is obtained as Eq. (4) by solving the constrained least squares problem, and shifting the neighbor space so that the query point is its origin, i.e., subtracting the query from all neighbors and the query itself [17]:

$$\mathbf{w}=\sum_i\tilde{\mathbf{C}}_{i,j}^{-1}\Big/\sum_{l,m}\tilde{\mathbf{C}}_{l,m}^{-1}. \tag{4}$$

where $\tilde{\mathbf{C}}$ is the local gram matrix centered on the query, i.e., $\tilde{\mathbf{C}}=(\mathbf{X}_{NN(\mathbf{x})}-\mathbf{x}\mathbf{1}_{k\times 1}^T)^T(\mathbf{X}_{NN(\mathbf{x})}-\mathbf{x}\mathbf{1}_{k\times 1}^T)$. Regularization term ($\Delta\ll 1$) for penalizing large weights is added to this local gram matrix as follows [18]:

$$\tilde{\mathbf{C}}\leftarrow\tilde{\mathbf{C}}+\Delta\mathbf{I}. \tag{5}$$

In LLR, the coefficients, i.e., weights are constrained to sum to one, but may be either positive or negative. Additionally, the coefficients may be forced to be non-negative, as in the LLR-based k-NN classification [13]. It solved the convex reconstruction [19] as the quadratic programming (QP) formulation [13]. Here, we consider only the conventional LLR without the non-negativity constraint, in that negative weights may be helpful for data points lying on the boundary of a manifold and outside the convex hull of their neighbors [19].

And we note that there are positive- or negative-weighted neighbors in LLR, unlike other weight kernels in the k-NN learning. Regarding the role of two kinds of neighbors, [13] stated that a query point is pulled by positive-weighted neighbors, while it is pushed away by negative-weighted neighbors. For consistency with the interpretation, here we use the following target predictive function in the k-NN regression:

$$y(\mathbf{x})=\bar{y}_{NN(\mathbf{x})}+\sum_{i=1}^{k} w_{NN(\mathbf{x},i)}[y_{NN(\mathbf{x},i)}-\bar{y}_{NN(\mathbf{x})}]. \tag{6}$$

The first term is an average of the target values of k neighbors ($\bar{y}_{NN(\mathbf{x})}=k^{-1}\sum_i y_{NN(\mathbf{x},i)}$). It is equivalent to the predicted output of the unweighted k-NN regression, which is the optimal output of the constant function to minimize the quadratic loss [20]. In contrast, the second term is interpreted as a correction term introduced by LLR. That is, positive-weighted neighbors in the local geometry captured by LLR pull the predicted output toward their target values, while negative-weighted neighbors push away it from their target values.

In this paper, we point out that a query and its k neighbors in LLR are geometrically assumed to lie on or close to a locally linear patch of the manifold in the input space [18]. Based on the assumption, the underlying neighborhood structure for a query is fitted to the fixed linear model. In this way, fitting its inherently fixed linear structure to the underlying local topology around a query point in the input space, may not guarantee the k-invariant

property of LLR, with the potential for over-fitting or under-fitting in some regressions. The limitation leads to our Bayesian kernel model in the locally linear reconstruction.

We would like to finish this section noting the difference between the proposed approach and similarly named approaches: kernelized k-NN and Bayesian k-NN. First, the kernelized k-NN was proposed to substitute a kernel distance metric for the original one in Hilbert space [21]. In the weighted k-NN view, the kernelized implementation can be achieved by employing some distance-based kernels as the weights for k neighbors. Although our proposed method also employs a Gaussian kernel, it utilizes the kernel in accordance with the neighborhood structure information. Second, the conventional Bayesian k-NN accounts for the uncertainty in the choice of k in k-NN prediction itself [22]. On the other hand, PLR does not directly handle (or optimize) the k parameter in Bayesian way. Rather, PLR is proposed in order to make the k-NN robust to the parameter, k, i.e., k-invariant.

3. Probabilistic local reconstruction

3.1. The local reconstruction in the probabilistic view

As the first step, we treat the locally linear reconstruction in the probabilistic view in order to capture the reconstruction uncertainty. Some query points are well-reconstructed from its neighbors, because the neighbors are sufficiently informative to describe the queries. For other queries, there may be no sufficient information to describe the queries in training points. The uncertain and inaccurate reconstructions for such query points give rise to performance deterioration in k-NN learning. Therefore, for a more reliable estimation, it is desirable to quantify the reconstruction uncertainty as a form of error-bars.

In this way, we define the reconstruction equation as a generative model for a query point $\mathbf{x}$ with an additive noise, following a Gaussian distribution with zero mean and covariance $\beta^{-1}\mathbf{I}$:

$$\mathbf{x}=\mathbf{X}_{NN(\mathbf{x})}\mathbf{w}+\varepsilon=\mathbf{x}_*+\varepsilon,\quad \varepsilon\sim N(\mathbf{0},\sigma^2\mathbf{I}=\beta^{-1}\mathbf{I}). \tag{7}$$

The additive noise assumption is introduced for explaining the difference between a query $\mathbf{x}$ and its reconstructed point $\mathbf{x}_*$. It serves as the regularization term in the non-probabilistic LLR. In the probabilistic view, the likelihood for reconstruction of the query $\mathbf{x}$, then becomes

$$p(\mathbf{x}|\mathbf{X}_{NN(\mathbf{x})},\mathbf{w})=N(\mathbf{x}|\mathbf{X}_{NN(\mathbf{x})}\mathbf{w},\beta^{-1}\mathbf{I}). \tag{8}$$

It is the probability density of the query point, given its corresponding k nearest neighbors. The explicit solution obtained by LLR is equivalent to the weight parameters that maximize the likelihood function. Rather than simply maximizing the likelihood, we incorporate Gaussian prior $p(\mathbf{w})=N(\mathbf{w}|k^{-1}\mathbf{1}_{k\times1},\mathbf{I})$ into the model, considering that the sum of weights is constrained to one, i.e., $\sum_i w_{NN(\mathbf{x},i)}=\mathbf{w}^T\mathbf{1}_{k\times1}=1$ and all neighbors equivalently contribute to reconstruction as a baseline. With the Gaussian prior over the weight vector, the weight posterior is obtained by Bayes' rule as follows:

$$p(\mathbf{w}|\mathbf{X}_{NN(\mathbf{x})},\mathbf{x})\propto p(\mathbf{x}|\mathbf{X}_{NN(\mathbf{x})},\mathbf{w})p(\mathbf{w})=N(\mathbf{w}|\mu,\Sigma). \tag{9}$$

In the weight posterior, the mean and covariance are defined as $\mu=\Sigma(\beta\mathbf{X}_{NN(\mathbf{x})}^T\mathbf{x}+k^{-1}\mathbf{1}_{k\times1})$ and $\Sigma=(\beta\mathbf{X}_{NN(\mathbf{x})}^T\mathbf{X}_{NN(\mathbf{x})}+\mathbf{I}^{-1})^{-1}=(\beta\mathbf{C}+\mathbf{I})^{-1}$ respectively. Finally, by shifting the neighbor space to the query point as its origin, our probabilistic LLR is obtained as follows:

$$\mu=\Sigma(k^{-1}\mathbf{1}_{k\times1}),\quad \Sigma=(\beta\tilde{\mathbf{C}}+\mathbf{I})^{-1}, \tag{10}$$

where $\tilde{\mathbf{C}}=\tilde{\mathbf{X}}_{NN(\mathbf{x})}^T\tilde{\mathbf{X}}_{NN(\mathbf{x})}=(\mathbf{X}_{NN(\mathbf{x})}-\mathbf{x}\mathbf{1}_{k\times1}^T)^T(\mathbf{X}_{NN(\mathbf{x})}-\mathbf{x}\mathbf{1}_{k\times1}^T)$. The posterior covariance matrix Σ is similar to the inversion of the regularized local gram matrix in Eq. (5), i.e., $(\tilde{\mathbf{C}}+\Delta\mathbf{I})^{-1}$ where $0<\Delta\ll1$ in the non-probabilistic LLR.

We note that this Bayesian treatment of LLR using the Gaussian prior is motivated by the probabilistic extension of LLR for avoiding the over-fitting problem. Introducing the Gaussian prior allows also the probabilistic LLR to explain the constant-fit, which leads to the equally weighted k-NN prediction. That is, in the probabilistic view, LLR is considered the learning process that adjusts the reconstructed projection at the average point of k neighbors (expected by the prior), to be close to the corresponding query point in the input space. If the additive noise assumption increases with few given neighbors, i.e., the parameter β approaches zero, the difference between a query $\mathbf{x}$ and its reconstructed point $\mathbf{x}_*$ increases, because the larger portion of such difference is explained by the noise assumption. It reduces the capacity of the model, making contributions of all neighbors to the reconstruction equal, i.e., $1/k$:

$$\lim_{\beta\to0}\mu=\lim_{\beta\to0}(\beta\tilde{\mathbf{C}}+\mathbf{I})^{-1}(k^{-1}\mathbf{1}_{k\times1})=k^{-1}\mathbf{1}_{k\times1}. \tag{11}$$

And the probabilistic estimate of PLR can be incorporated into the predictive variance of the k-NN regressor, providing both of the two reject options together, i.e., the reject option by the lack of training points, and the ambiguity one [23,24] by inconsistent or severely dispersed target values of k neighbors in the case of k-NN regression. Applying the weight posterior mean μ and covariance Σ to the k-NN prediction in Eq. (6), we obtained the predictive distribution for the target value y of a query point $\mathbf{x}$:

$$y=\bar{y}_{NN(\mathbf{x})}+\mu^T(\mathbf{y}_{NN(\mathbf{x})}-\bar{y}_{NN(\mathbf{x})}\mathbf{1}_{k\times1}) \tag{12}$$

$$\sigma^2=\frac{1}{k}(\mathbf{y}_{NN(\mathbf{x})}-\bar{y}_{NN(\mathbf{x})}\mathbf{1}_{k\times1})^T\Sigma(\mathbf{y}_{NN(\mathbf{x})}-\bar{y}_{NN(\mathbf{x})}\mathbf{1}_{k\times1}), \tag{13}$$

where $\mathbf{y}_{NN(\mathbf{x})}=[y_{NN(\mathbf{x},1)},\ldots,y_{NN(\mathbf{x},k)}]^T$. k^{-1} in the predictive variance is added for consistency with the variance of the k neighbors' target values. When we define a predictive variance in the equally weighted k-NN regression as follows:

$$\sigma^2=\frac{1}{k}(\mathbf{y}_{NN(\mathbf{x})}-\bar{y}_{NN(\mathbf{x})}\mathbf{1}_{k\times1})^T(\mathbf{y}_{NN(\mathbf{x})}-\bar{y}_{NN(\mathbf{x})}\mathbf{1}_{k\times1}), \tag{14}$$

the predictive variance of PLR defined in Eq. (13) is explained as the variance of neighbors' target values considering the geometric properties of a neighborhood as a form of the covariance matrix of the PLR solution $\Sigma=(\beta\tilde{\mathbf{C}}+\mathbf{I})^{-1}$. The difference between the predictive variances of the equally weighted and PLR-based k-NN regressions is shown in Fig. 1. The figure shows the predictive variances for the predictions as the one standard-deviation error-bars, based on the training points sampled in the restrictive region, i.e., $-8\le x\le8$ in the one-dimensional input space. It is intuitive that the error bars get larger for input values that are far from any training points [25], because extrapolations to extremely sparse or unknown regions in the input space become uncertain [26,27]. As shown in Fig. 1(a), the predictive variance of the equally weighted k-NN regression does not capture such extrapolation regions, since it depends only on neighbors' target values. On the other hand, the predictive variances of PLR in sparse regions with no informative training points are corrected by the posterior covariance term, and dramatically increase as shown in Fig. 1(b). In this way, the predictive variance of PLR can be interpreted as a reliance level of the k-NN prediction, taking account of extrapolation cases in the input space. Prediction of a query's target value with high predictive variance can be rejected without unreliable prediction.

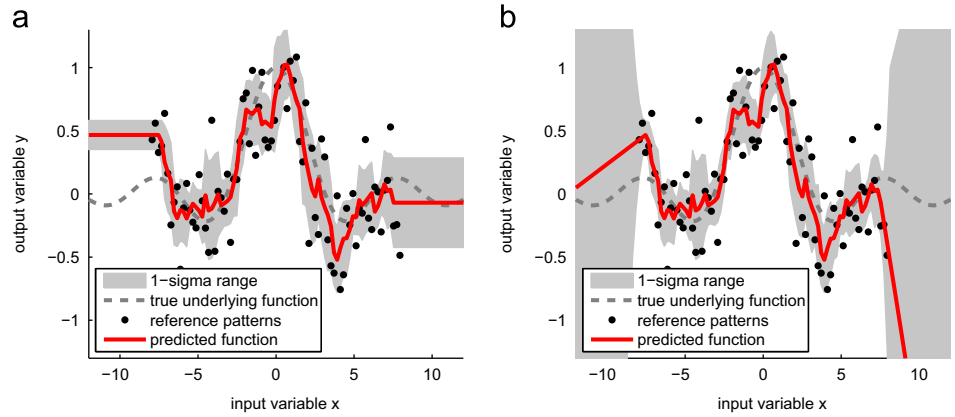

Fig. 1. Illustration of the predictive variances of (a) equally weighted and (b) PLR-based k-NN regressions (k=5 and N=200).

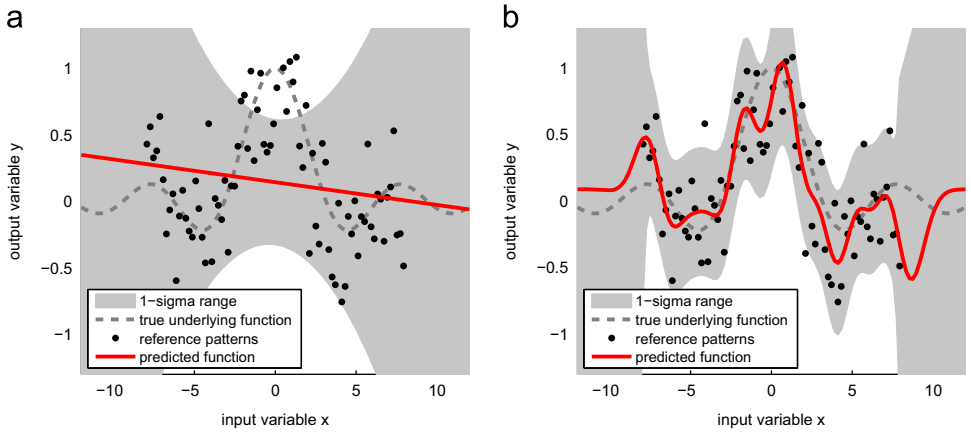

Fig. 2. Illustration of the predictive variances of two k-NN regressions using PLR with (a) linear kernel and (b) RBF kernel (k=200 and N=200).

Table 1
Description of the data sets.

Data set	Train	Test	Attribute	Origin	Time-series?
Stock	500	450	9	Luís Torgo	No
Abalone	2000	2000	8	Luís Torgo	No
Wind	3000	3500	11	StatLib	No
Computer	4000	4000	22	Delve	No
Kinematics	4000	4000	8	Luís Torgo	No
Bank	4000	4000	8	Luís Torgo	No
Pumadyn	4000	4000	8	Luís Torgo	No
Add10	4000	5000	5	Delve	No
California	5000	5000	8	Luís Torgo	No
Census	5000	5000	9	Luís Torgo	No
Gold	500	500	10	TSDL	Yes
Sunspot	1000	1500	10	TSDL	Yes
Melbmax	1000	2500	10	TSDL	Yes
Santa Fe A	5000	5000	10	TSDL	Yes
Santa Fe D	5000	5000	10	TSDL	Yes

Table 2
Parameter settings for each algorithm.

Model label	Parameter settings
Unweighted	k=3,5,10,50,100,200,300,400,500
RBF	k=3,5,10,50,100,200,300,400,500 $w_{NN(\mathbf{x},i)}=\exp\left(-\frac{1}{2l^2}\|\mathbf{x}-\mathbf{x}_{NN(\mathbf{x},i)}\|^2\right)$, l=1
LLR	k=3,5,10,50,100,200,300,400,500 $\mathbf{w}_{NN(\mathbf{x})}=\sum_i \tilde{\mathbf{C}}^{-1}_{i,j}/\sum_{l,m}\tilde{\mathbf{C}}^{-1}_{l,m}$ $\Delta=10^{-3}$
PLR-linear	k=3,5,10,50,100,200,300,400,500 $N(\mathbf{w}_{NN(\mathbf{x})}\|\mu=\Sigma(k^{-1}\mathbf{1}_{k\times 1}),\Sigma=(\beta\tilde{\mathbf{K}}+\mathbf{I})^{-1})$ $\mathbf{k}(\mathbf{x},\mathbf{x}')=\mathbf{x}^T\mathbf{x}'$, linear kernel β=10,100,1000
PLR-RBF	k=3,5,10,50,100,200,300,400,500 $N(\mathbf{w}_{NN(\mathbf{x})}\|\mu=\Sigma(k^{-1}\mathbf{1}_{k\times 1}),\Sigma=(\beta\tilde{\mathbf{K}}+\mathbf{I})^{-1})$ $\mathbf{k}(\mathbf{x},\mathbf{x}')=\exp\left(-\frac{1}{2l^2}\|\mathbf{x}-\mathbf{x}'\|^2\right)$, l=1 β=10,100,1000
LinReg	–

3.2. Projecting inputs into a feature space

As described earlier, LLE is based on the assumption that a query and its k neighbor points geometrically lie on or close to a locally linear patch of the underlying manifold in the input space [18]. Moreover, setting k too large violates the assumption [19], and may be sub-optimal in non-linear geometric structures comprising too many neighbors in the input space. Thus, in order to make LLR robust to a wider range of k values, we transform a lower-dimensional input space into a higher-dimensional feature space ($\mathbf{x}\mapsto\phi(\mathbf{x})$) and adopt a kernel function $\mathbf{k}(\mathbf{x},\mathbf{x}')=\phi(\mathbf{x})^T\phi(\mathbf{x}')$ to

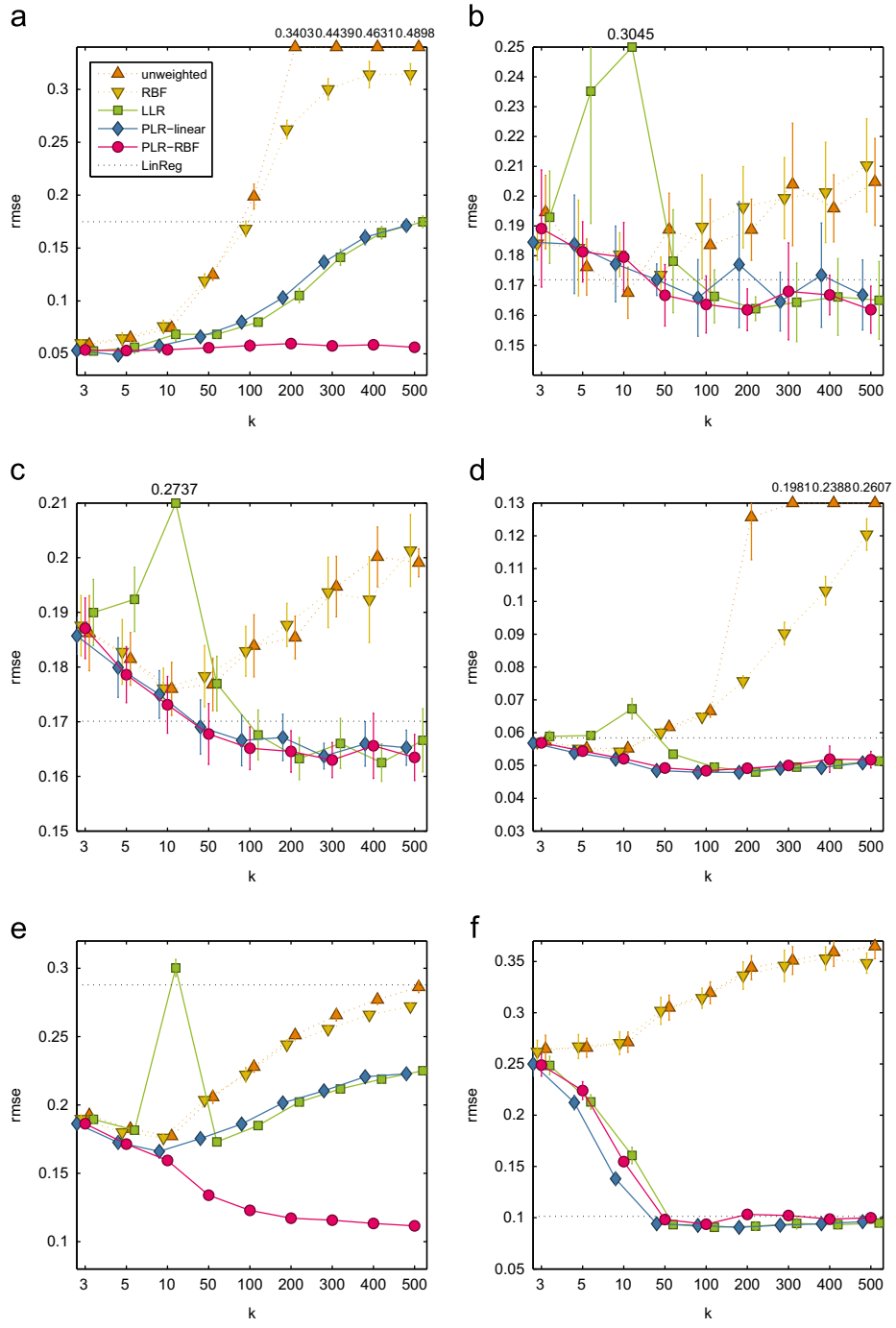

Fig. 3. The RMSE means and one-sigma ranges of all the models by k values for each data set (x-axis denotes k while y-axis denotes the RMSE). (a) `Stock`, (b) `Abalone`, (c) `Wind`, (d) `Computer`, (e) `Kinematics`, (f) `Bank`.

compute the dot product in the feature space with no explicit mapping function $\phi(\cdot)$. Applying the kernel function to the posterior mean and covariance in Eq. (10), we obtained the following PLR solution:

$$\mu = \Sigma(k^{-1}\mathbf{1}_{k\times 1}), \quad \Sigma = (\beta\tilde{\mathbf{K}} + \mathbf{I})^{-1}. \tag{15}$$

In the equation, the centered kernel matrix $\tilde{\mathbf{K}}$ is the same as the matrix of kernel Principal Component Analysis (KPCA) [28], and defined as $\tilde{\mathbf{K}} = \mathbf{K} - \mathbf{1}_{k\times 1}\mathbf{k}_*^T - \mathbf{k}_*\mathbf{1}_{k\times 1}^T + \mathbf{k}(\mathbf{x},\mathbf{x})\mathbf{1}_{k\times 1}\mathbf{1}_{k\times 1}^T$, where $\mathbf{K}$ is the kernel matrix defined as the k-by-k matrix, $\mathbf{K}_{ij} = \mathbf{k}(\mathbf{x}_{NN(\mathbf{x},i)}, \mathbf{x}_{NN(\mathbf{x},j)})$, and $\mathbf{k}_*$ is the column vector consisting of the kernel functions of k neighbors for a query point, i.e., $\mathbf{k}_* = [\mathbf{k}(\mathbf{x}, \mathbf{x}_{NN(\mathbf{x},1)}), \ldots, \mathbf{k}(\mathbf{x}, \mathbf{x}_{NN(\mathbf{x},k)})]^T$.

The kernelized expansion of LLR in our PLR model makes the model more flexible for various k settings. If the kernel function is linear, i.e., $\mathbf{k}(\mathbf{x},\mathbf{x}') = \mathbf{x}^T\mathbf{x}'$, PLR has the same model complexity as LLR. If we use other kernel functions, on the other hand, such as the radial basis function (RBF) kernel $\mathbf{k}(\mathbf{x},\mathbf{x}') = \exp(-(1/2l^2)\|\mathbf{x}-\mathbf{x}'\|^2)$, the PLR can capture non-linear geometric structures with large k settings.

For example, Fig. 2 shows the comparison of the two k-NN regressions with $k=200$ and the number of training points, $N=200$. PLR in the one-dimensional input space is simplified as representing a query point x as a weighted combination of neighbors' input values, i.e., $x = \sum_{i=1}^{k} w_{NN(x,i)} x_{NN(x,i)}$. It implies that all neighbors' contributions, i.e., weights in the reconstruction

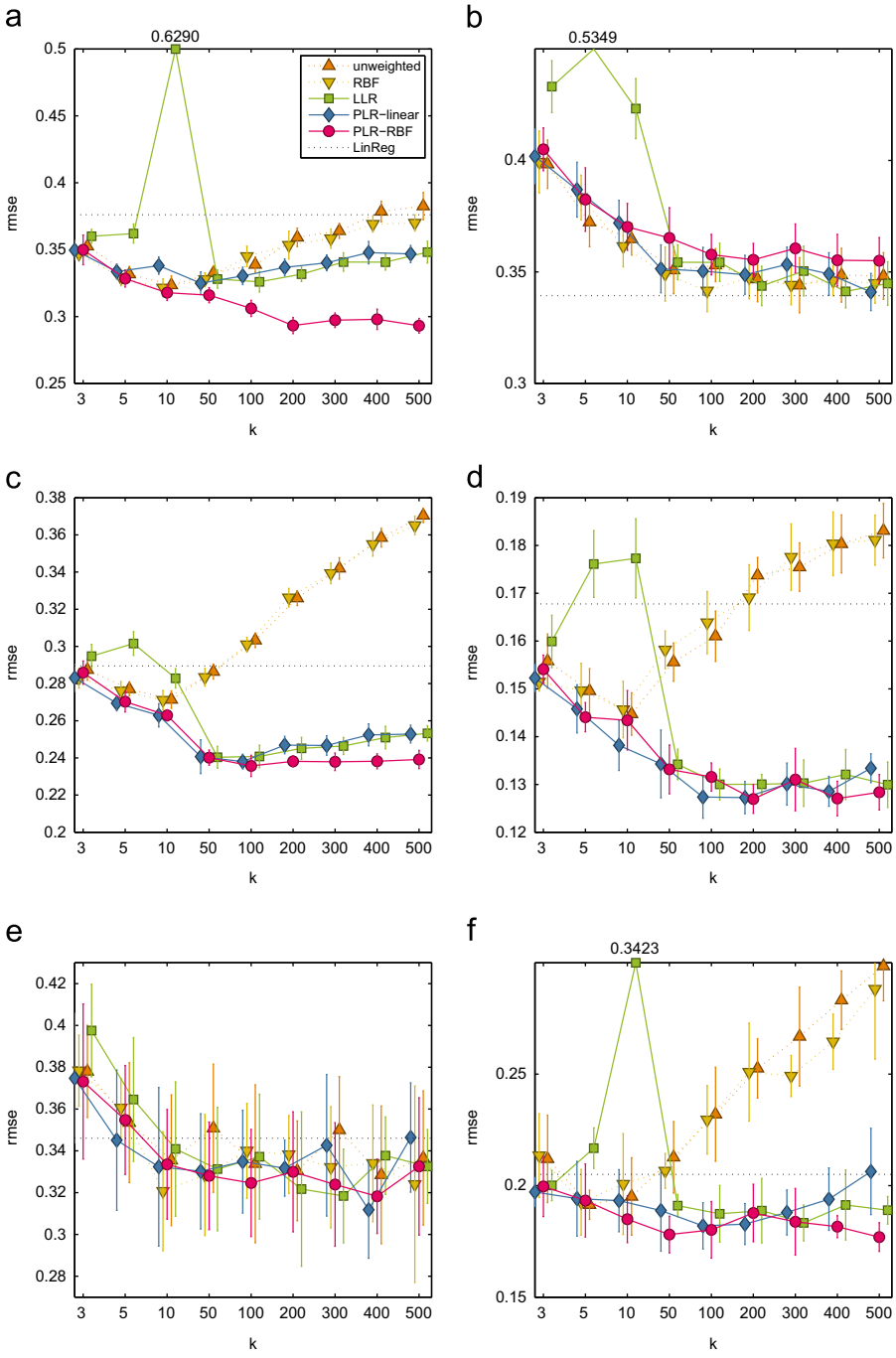

Fig. 4. The RMSE means and one-sigma ranges of all the models by k values for each data set (x-axis denotes k while y-axis denotes the RMSE). (a) `Pumadyn`, (b) `Add10`, (c) `California`, (d) `Census`, (e) `Gold`, (f) `Sunspot`.

become linearly proportional to their distances or similarities with the query value. In this way, as shown in Fig. 2, k-NN regression using this neighborhood kernel in large k settings provides a linear fit where all the input values are replaced with their corresponding target values, i.e., $y = \sum_{i=1}^{k} w_{NN(x,i)} y_{NN(x,i)}$. On the other hand, the contributions estimated by PLR in the feature space can be biased to more relevant neighbors, i.e., not linear, as shown in Fig. 2(b). In this way, since PLR in the feature space captures such non-linear or biased neighborhood structure, the kernelized PLR avoids performance deterioration in large k settings.

In summary, with the definition of the local reconstruction in the probabilistic view, we expect our local reconstruction to provide a probabilistic estimate including a confidence interval for the predicted output. Moreover, under the probabilistic view, we incorporate Gaussian regularization prior into the model and adopt a kernel trick in order to make our local reconstruction model more flexible for various k settings. In this way, we expect that the Bayesian kernel treatment in the local reconstruction improves the k-invariant property as well as prediction accuracy in the local reconstruction-based k-NN regression.

4. Experiments and results

4.1. Experiments on benchmark data sets

We conducted experiments on 15 regression tasks. The data sets used in our experiments were selected from the regression

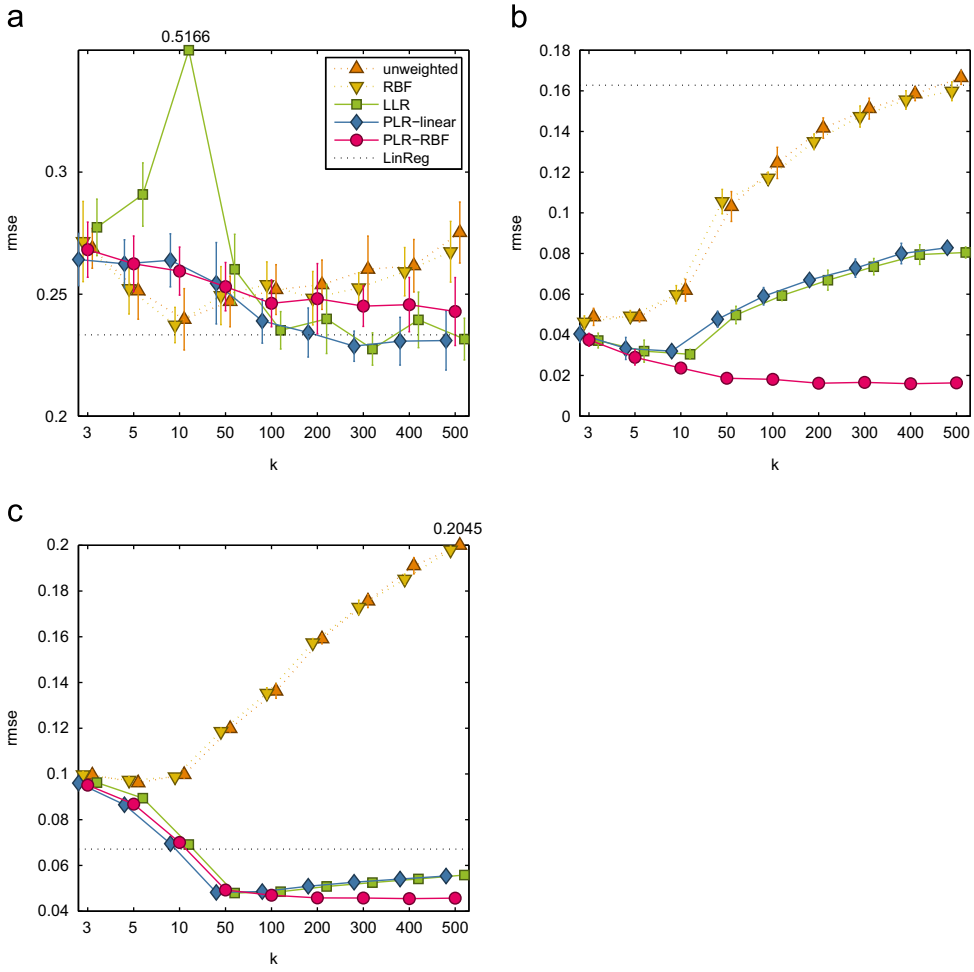

Fig. 5. The RMSE means and one-sigma ranges of all the models by k values for each data set (x-axis denotes k while y-axis denotes the RMSE). (a) `Melbmax`, (b) `Santa Fe A`, (c) `Santa Fe D`.

data sets provided by Luís Torgo,[1] Delve data sets,[2] Time Series Data Library (TSDL),[3] and Statlib.[4] The descriptions of each data set are summarized in Table 1. For the time series data sets, we re-formulated them as regression problems by defining 10 consecutive values as inputs and the following value as the output. Such re-formulation is a typical method used in other research related to regression problems [29]. In addition, we randomly selected a predefined number of data points for the training and testing ten times. The 'train' column in Table 1 denotes the number of training points for each data set, while 'test' denotes the number of data points to be predicted, i.e., query points.

For verifying the effectiveness of PLR for k-NN regression, five k-NN regression models with linear regression (labeled as 'LinReg') were used in the benchmark data sets. We tested the k-NN models in nine k values ranging from 3 to 500, validating that the models are robust to any non-optimized k values, i.e., k-invariant property. The models and corresponding parameter settings are summarized in Table 2. First, unweighted or equally weighted k-NN regression was used as a baseline, and labeled as 'unweighted' in Table 2. Compared with the 'unweighted' one, the other four k-NN regression models are kernelized versions. That is, second, the Gaussian or RBF kernel was used as one of the distance-based ones for the kernelized k-NN regression. Third, LLR defined in Eq. (4) was tested with the constant regularization term $\Delta = 10^{-3}$. Fourth, our probabilistic extension of LLR, i.e., PLR in the input space was adopted as a weight kernel of k-NN regression. We labeled it as 'PLR-linear' in that the linear kernel function $\mathbf{k}(\mathbf{x},\mathbf{x}') = \mathbf{x}^T\mathbf{x}'$ is used in the view point of the kernelized PLR defined in Eq. (15). Moreover, the parameter β in the noise assumption of PLR was selected among the three candidates, i.e., {10,100,1000} by the five-fold cross-validation for training points. Comparing it with LLR, we explored how significant the tuned noise assumption for locally linear reconstruction is in given small k settings. Finally, PLR in the feature space defined by the RBF kernel $\mathbf{k}(\mathbf{x},\mathbf{x}') = \exp(-(1/2l^2)\|\mathbf{x}-\mathbf{x}'\|^2)$ with a fixed $l=1$ was tested and labeled as 'PLR-RBF' in Table 2. As mentioned earlier, it is expected to capture the non-linear neighborhood structure formed by large k values. In this preliminary experiment, we explored how beneficial such flexible reconstruction is in regressions with large k values.

Figs. 3–5 depict the RMSE values and their one-sigma ranges of all the models by k values. LinReg, i.e., linear regression was learned from the set of training samples regardless of k values. Thus, its RMSE value is denoted as a horizontal dotted line for each benchmark data set. Note that we set the caps on y-axis for some plots, and ignore the actual scale of k values for the sake of interpretation. Moreover, Tables 3–5 summarize the RMSEs by four intervals of k values, i.e., low (small k=3, 5, 10), middle (k=50, 100, 200), high (large k=300, 400, 500), and overall with statistical tests. From the experimental results, there are some empirical observations and discussions to be noted as follows.

[1] http://www.liaad.up.pt/~ltorgo/Research/
[2] http://www.cs.toronto.edu/~delve/data/datasets.html
[3] http://robjhyndman.com/TSDL/
[4] http://lib.stat.cmu.edu/datasets/

Table 3
The RMSE values of all the *k*-NN models with four intervals (i.e., low, middle, high, and overall) of *k* values for each data set. The bold faced number represents the best *k*-NN RMSE for the corresponding data sets. The double plus, i.e., ++ indicates that the marked *k*-NN model is superior to other four *k*-NN models with significance level of 0.05, while the plus indicates that the marked model is superior to other three *k*-NN models with the same significance level. LinReg is excluded from all the additional indications.

Data set	*k*	Unweighted	RBF	LLR	PLR-linear	PLR-RBF	LinReg
Stock	Low	0.0661	0.0669	0.0591	**0.0532**$^{+}$	0.0536^{+}	–
	Middle	0.2213	0.1831	0.0845	0.0831	**0.0578**$^{++}$	–
	High	0.4656	0.3095	0.1602	0.1561	**0.0575**$^{++}$	–
	Overall	0.2510	0.1865	0.1013	0.0975	**0.0563**$^{++}$	0.1749
Abalone	Low	0.1795	0.1824	0.2442	**0.1819**	0.1833	–
	Middle	0.1870	0.1865	0.1689	0.1716	**0.1641**$^{+}$	–
	High	0.2015	0.2036	**0.1652**	0.1683	0.1656	–
	Overall	0.1893	0.1908	0.1928	0.1739^{+}	**0.1710**$^{+}$	0.1719
Wind	Low	0.1812	0.1821	0.2187	0.1802	**0.1796**	–
	Middle	0.1820	0.1830	0.1693	0.1676	**0.1658**$^{+}$	–
	High	0.1980	0.1958	0.1651	0.1650	**0.1640**	–
	Overall	0.1871	0.1870	0.1843	0.1709^{+}	**0.1698**$^{+}$	0.1701
Computer	Low	0.0559	0.0555	0.0618	**0.0542**$^{+}$	0.0545^{+}	–
	Middle	0.0847	0.0668	0.0503	**0.0481**$^{++}$	0.0489^{+}	–
	High	0.2325	0.1046	0.0504	**0.0497**$^{++}$	0.0512	–
	Overall	0.1244	0.0757	0.0542	**0.0507**$^{++}$	0.0515^{+}	0.0584
Kinematics	Low	0.1841	0.1819	0.2238	**0.1750**$^{+}$	0.1723^{+}	–
	Middle	0.2282	0.2233	0.1867	0.1876	**0.1246**$^{++}$	–
	High	0.2764	0.2645	0.2186	0.2179	**0.1135**$^{++}$	–
	Overall	0.2296	0.2232	0.2097	0.1935^{+}	**0.1368**$^{++}$	0.2829
Bank	Low	0.2673	0.2663	0.2074	**0.2001**	0.2092	–
	Middle	0.3228	0.3174	**0.0919**$^{+}$	0.0924^{+}	0.0984	–
	High	0.3582	0.3490	**0.0942**$^{+}$	0.0943^{+}	0.1003	–
	Overall	0.3161	0.3109	0.1312	**0.1289**	0.1359	0.1015

Table 4
The RMSE values of all the *k*-NN models with four intervals (i.e., low, middle, high, and overall) of *k* values for each data set. The bold faced number represents the best *k*-NN RMSE for the corresponding data sets. The double plus, i.e., ++ indicates that the marked *k*-NN model is superior to other four *k*-NN models with significance level of 0.05, while the plus indicates that the marked model is superior to other three *k*-NN models with the same significance level. LinReg is excluded from all the additional indications.

Data set	*k*	Unweighted	RBF	LLR	PLR-linear	PLR-RBF	LinReg
Pumadyn	Low	0.3359	0.3325	0.4504	0.3405	**0.3320**	–
	Middle	0.3436	0.3421	0.3286	0.3307	**0.3051**$^{++}$	–
	High	0.3750	0.3658	0.3432	0.3449	**0.2961**$^{++}$	–
	Overall	0.3515	0.3468	0.3740	0.3387^{+}	**0.3111**$^{++}$	0.3761
Add10	Low	0.3785	0.3814	0.4638	0.3869	**0.3859**	–
	Middle	0.3502	**0.3460**	0.3508	0.3502	0.3595	–
	High	0.3469	**0.3450**	0.3455	0.3478	0.3570	–
	Overall	0.3585	**0.3575**	0.3867	0.3616	0.3675	0.3394
California	Low	0.2786	0.2767	0.2930	**0.2718**$^{+}$	0.2730	–
	Middle	0.3052	0.3035	0.2421	0.2418	**0.2380**$^{++}$	–
	High	0.3570	0.3531	0.2502	0.2506	**0.2384**$^{++}$	–
	Overall	0.3136	0.3111	0.2618	0.2548^{+}	**0.2498**$^{++}$	0.2895
Census	Low	0.1500	0.1489	0.1711	**0.1454**$^{+}$	0.1472	–
	Middle	0.1635	0.1637	0.1314	**0.1296**	0.1306	–
	High	0.1796	0.1797	0.1308	0.1307	**0.1288**	–
	Overall	0.1644	0.1641	0.1444	**0.1352**$^{+}$	0.1355^{+}	0.1678
Gold	Low	0.3557	0.3532	0.3677	**0.3507**	0.3538	–
	Middle	0.3384	0.3355	0.3301	0.3322	**0.3275**	–
	High	0.3384	0.3301	0.3295	0.3336	**0.3249**	–
	Overall	0.3441	0.3396	0.3424	0.3389	**0.3354**	0.3460
Sunspot	Low	0.1995	0.2023	0.2531	0.1949	**0.1927**	–
	Middle	0.2325	0.2290	0.1891	0.1846	**0.1820**$^{+}$	–
	High	0.2828	0.2673	0.1879^{+}	0.1961	**0.1808**$^{++}$	–
	Overall	0.2383	0.2329	0.2100	0.1918^{+}	**0.1852**$^{++}$	0.2050

First, problems such as `Abalone`, `Wind`, `Bank`, `Add10`, `Gold`, `Melbmax`, and `Santa Fe C` seem to be sampled sparsely, or have noisy relationships between the inputs and target variable. It is inferred from that RMSEs of the unweighted *k*-NN model at the small *k* interval are higher than the RMSEs of LinReg. Since weight kernels of the five *k*-NN models do not depend on the target variable, the characteristic of the problems is the inherent limitation of the *k*-NN models, and may be more critical than how to estimate the

Table 5
The RMSE values of all the k-NN models with four intervals (i.e., low, middle, high, and overall) of k values for each data set. The bold faced number represents the best k-NN RMSE for the corresponding data sets. The double plus, i.e., $++$ indicates that the marked k-NN model is superior to other four k-NN models with significance level of 0.05, while the plus indicates that the marked model is superior to other three k-NN models with the same significance level. LinReg is excluded from all the additional indications.

Data set	k	Unweighted	RBF	LLR	PLR-linear	PLR-RBF	LinReg
Melbmax	Low	**0.2532**$^{+}$	0.2537^{+}	0.3616	0.2635	0.2634	–
	Middle	0.2509	0.2506	0.2451	**0.2426**$^{+}$	0.2491	–
	High	0.2657	0.2597	0.2329^{+}	**0.2301**$^{+}$	0.2446	–
	Overall	0.2566	0.2547	0.2798	**0.2454**$^{++}$	0.2524	0.2333
Santa Fe A	Low	0.0532	0.0517	0.0332	0.0352	**0.0300**$^{++}$	–
	Middle	0.1231	0.1192	0.0586	0.0578	**0.0176**$^{++}$	–
	High	0.1588	0.1542	0.0778	0.0785	**0.0163**$^{++}$	–
	Overall	0.1117	0.1084	0.0565	0.0572	**0.0213**$^{++}$	0.1627
Santa Fe D	Low	0.0985	0.0986	0.0850	0.0840	**0.0840**	–
	Middle	0.1384	0.1370	0.0491	0.0492	**0.0474**$^{++}$	–
	High	0.1904	0.1853	0.0542	0.0540	**0.0457**$^{++}$	–
	Overall	0.1424	0.1403	0.0627	0.0624	**0.0590**	0.0671

weights for k neighbors. As a result, in these seven regression problems, PLR-linear or PLR-RBF is not significantly superior to other k-NN models in terms of the overall RMSE except `Abalone`, `Wind`, and `Melbmax`, although one of their RMSEs is the best one except `Add10`. However, PLR can be useful in that it provides the reliance level of a prediction, which is demonstrated with a real-world application later.

Second, when we compared PLR-linear with LLR, RMSEs by k values are showed clear similarity in some data sets such as `Bank`, `Santa Fe A`, and `Santa Fe D`. It is due to that these two models have the same model complexity in the weight allocation problem of k-NN learning. Moreover, in the data sets, these two models are relatively more robust to k values than the unweighted and RBF k-NN models. However, there were spikes in LLR with a small k value in some regression problems. Such spikes – even higher than the RMSE of the unweighted k-NN model – imply that LLR with a handful of neighbors has the risk of the over-fitting problem and its regularization term should be carefully tuned. In contrast, PLR-linear whose parameter β is optimized, avoids the over-fitting problem, showing significantly superior performances in five (i.e., `Stock`, `Computer`, `Kinematics`, `California`, and `Census`) out of the eight less noisy problems in terms of RMSE at the small k interval in the tables.

Third, PLR-RBF outperformed both PLR-linear and LLR in large k settings. The tables show that PLR-RBF is significantly better than the two k-NN models at the large k interval in six (i.e., `Stock`, `Kinematics`, `Pumadyn`, `California`, `Sunspot` and `Santa Fe A`) out of the eight less noisy problems. We note that the unweighted k-NN model at the large k interval is equivalent to the (global) constant fit. This way, when the RMSE of the unweighted k-NN at the large k interval becomes nearly equal to the RMSE of LinReg in a regression problem, the problem is likely to have the non-linear relationship as LinReg does not work properly. In our experiment, `Kinematics`, `Pumadyn` and `Santa Fe A` seem to have such tendency, and are included in the six regression problems where PLR-RBF works well. Although PLR-RBF does not directly capture the non-linear relationships between the inputs and target variable, PLR-RBF has on the average 53% of PLR-linear's RMSE at the large k interval in the non-linear problems. That is, PLR-RBF based k-NN regression can avoid performance deterioration due to large k values. Particularly, the effect becomes salient in the non-linear regression problems, illustrated by the example in Fig. 2.

Finally, PLR was significantly superior to other three k-NN models at the overall interval of k values in 11 out of all the 15 regression problems. From this experimental result, we conclude that PLR is more k-invariant than other k-NN models. Users may choose any k values without any significant performance drop due to setting an inappropriate k value, e.g., too small or too large. In addition, the theoretical time complexity of PLR is the same as that of LLR, i.e., $O(N \log N + k^{2.376})$ due to k neighbors identification and matrix inversion.[5] However, as shown in Fig. 6, the experimental time–cost per one query point is heavier than that of LLR. Though light, PLR requires overhead time–cost. Nevertheless, in small k settings, such difference becomes small. The additional time–cost is compensated by the PLR's stable prediction performances and additional information on prediction results.

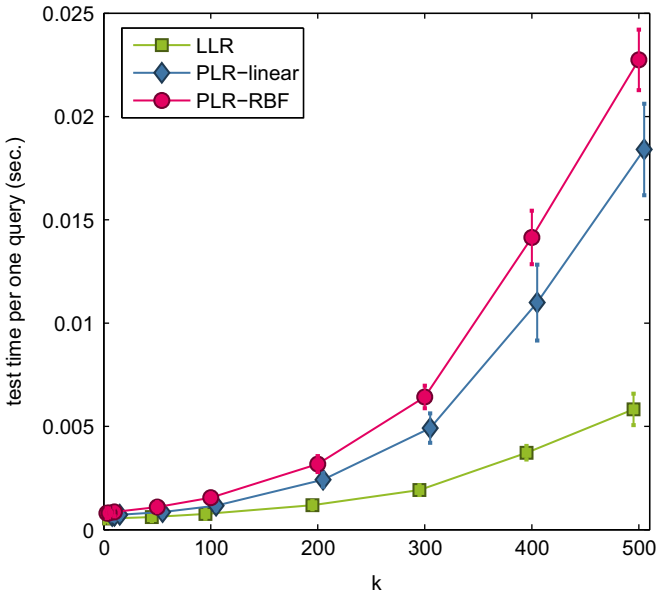

Fig. 6. The prediction time per one query point of three k-NN models. (x-axis denotes k values while y-axis denotes the prediction time per one query in second.)

4.2. Real-world application: virtual metrology

The fabrication process in the semiconductor manufacturing consists of 300–500 highly complex processing steps [1,31–33], quality of which is commonly evaluated as yield, or the fraction of total input transformed into shippable output, i.e., good chips with

[5] Since the fastest matrix inversion requires $O(n^{2.376})$ [30], we wrote the computational cost $O(n^{2.376})$ instead of $O(n^3)$ in this paper.

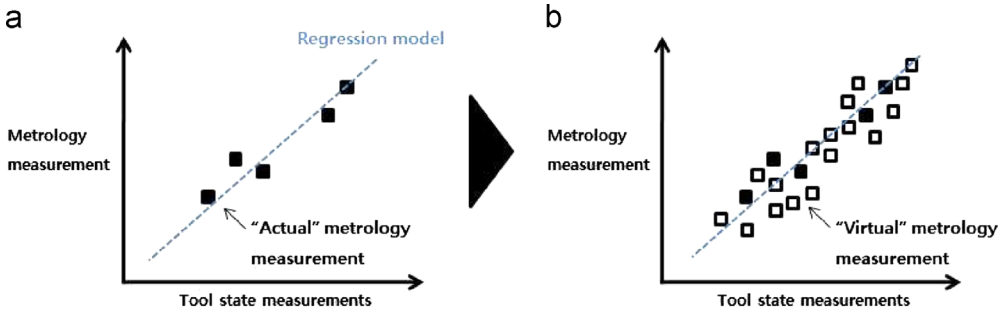

Fig. 7. Concept of virtual metrology (VM): (a) learning the relation between a tool state measurement and "actual" metrology measurement, (b) predicting "virtual" metrology measurement for wafers with no metrology measurement.

no defect [34]. Since the yield cannot be monitored until the end of the overall process [31], other measurement available in smaller time-scale and lower level is necessary for the quality management of the "on-line" process. In this way, the metrology measured at the process step level [31] becomes indispensable for sustaining yield performance at every step in conjunction with Run-to-Run (R2R) control [35,1,2]. That is, metrology measurements such as thickness, resistance, critical dimension (CD), overlay, particles, and etch rate [31] are utilized as a feedback for adjusting the tool state variables such as temperature, pressure, flow and current at a tool assigned for a processing step [31]. Due to time and cost, one wafer for one lot of 25 wafers [1,2] is actually measured [31]. In other words, the quality of every wafer cannot be assured in the lot-to-lot quality management [36,37,1,32]. Virtual metrology (VM) is defined as the technique to predict metrology measurements from tool state measurements without "actual" metrology operations [36], and commonly formulated as a regression problem [36–39,1]. As shown in Fig. 7, a regressor is built from the training pairs each consisting of tool state measurements as an input point and metrology one as an output value. Then, the metrology values of "non-measured" wafers in the metrology operation are predicted from their corresponding tool state measurements by the VM model [2].

Although the use of regression-based VM experimentally have shown accurate predictions of "actual" metrology values in many other research, process engineers in the application still have the question for the reliance levels of the virtual measurement values – actually, no one know the accuracy of the "virtual" ones. This reliability issue is related to the risk in the actual use of VM, often causing hesitation in the application [40,32]. Here, our PLR framework in k-NN regression is useful in that it provides one unified solution for error-bars of virtual measurements. It is the k-NN predictive variance incorporated with our PLR estimate, which take account of extrapolation cases without additional modules such as outlier detection models. This predictive variance is defined in Eq. (13), and provides error-bars shown in Fig. 1(b).

Our real-world VM data used for demonstration was gathered from a tool for photo-lithography processing step and its corresponding metrology operation at a Korean foundry for a period of four months. The tool state is described in terms of 117 variables, and the wafer quality is represented by one variable. As the goal is to show the effectiveness of our approach, we focused on the first 600 wafers on the time-ordered sequence of wafers measured in the metrology operation. Fig. 8(a) shows changes of actual values for the chosen metrology variable on the discrete time index $t = 1, \ldots, 600$. Fig. 8(b) shows how the tool states change after the preventive maintenances. This non-stationary property of the VM application requires regression models to be "adaptive" in the data stream view. Thus, for adaptation of our k-NN regression model, here we used Time-Weighted Forgetting (TWF) [41], one of

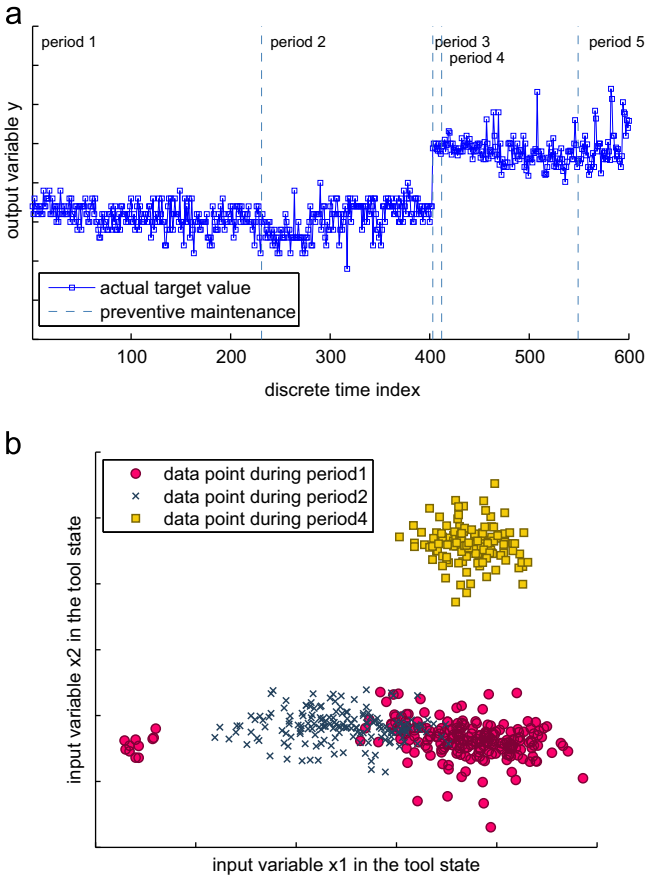

Fig. 8. VM data exploration (a) metrology measurements change after preventive maintenances, (b) tool state changes after preventive maintenances.

adaptive instance selection methods for k-NN learning.[6] TWF is implemented by the exponential weighting function. The TWF weight for a new incoming training pair is initially set to $w_{twf} = 1$. Then, the weights for each training pairs are recursively updated by the simple rule, $w_{twf} \leftarrow \gamma w_{twf}$, with $0 < \gamma < 1$. Forgetting is achieved by deleting training pairs with w_{twf} below the threshold θ. In this way, TWF is equivalent to the sliding window technique, keeping only current $N = \log \theta / \log \gamma$ training pairs with forgetting older ones [41]. Consequently, k-NN predictions with TWF are

[6] Further discussion of other adapting methods in changing environments such as our VM application is out of scope of this current work (the problem due to changing environments is commonly referred to as concept drift problem [42], and see [43,44]), and leads to future work.

always based on the current N (e.g., $N=50$ in this work) training pairs.

Fig. 9 shows the result of applying our linear PLR-based 5-NN regression with TWF to the VM data stream shown in Fig. 8. No variable selection was performed for simplification. At every lazy-learning for a wafer, each input variable was normalized by min–max scaling, and filtered if all values on the variable are identical, as in the work of [1,2]. Given k-invariance of PLR, the parameter k was set to five, to minimize the computational load and the parameter β is selected among 10, 100, 1000 by the five-fold cross-validation on training points. Our model, then, provides the predicted range of the virtual metrology value in terms of the predictive mean and variance, as shown in Fig. 1(b). The higher the predictive variance is, the larger the range is, implying that the prediction is not reliable. We compared the actual RMSE of "reliable" and "unreliable" predictions when the threshold is defined as the variance covering 95% of training points selected by TWF.

Table 6 shows the VM prediction performances of the regression models used in Section 4.1. Comparison of RMSEs of overall predictions by models confirms that the grosser features of the problem learned by simpler models, such as the unweighted k-NN, are more likely to persist than the smaller features learned by the more sophisticated models in the changing environment [45]. However, by detecting unreliable predictions due to the change, PLR provides more reliable and accurate prediction performance. In PLR-linear, unreliable predictions led to on average 20 times RMSE (Table 6). Moreover, contrary to a "reasonable" assumption that predictions near preventive maintenances are unreliable, many predictions within periods – believed to be relatively stationary – are unreliable. For example, VM predictions for wafers at the discrete time index $t=345$ and 538 are detected as unreliable ones. These predictions are not affected by training points collected during their foregone periods, because TWF is set to keep current 50 training points. Fig. 10 shows training points selected by TWF (denoted as filled circles) and forgotten ones within the same period (denoted as unfilled circles) at $t=345$ and 538, respectively. As shown in Fig. 10(a), it seems that the unreliable prediction at $t=345$ was due to process drift. The tool state was slowly shifting from right to left. With the process drift, the query point at $t=345$ is in the leftmost region. On the other hand, in Fig. 10(b), the query point at $t=538$ (denoted as a cross) seems to be simply an outlier, with no drift. Consequently, any regression models would have to extrapolate the query point from relatively "unrelated" training points. Our proposed model provides the reliability level for the prediction, which is critical for detecting and ignoring unreliable predictions.

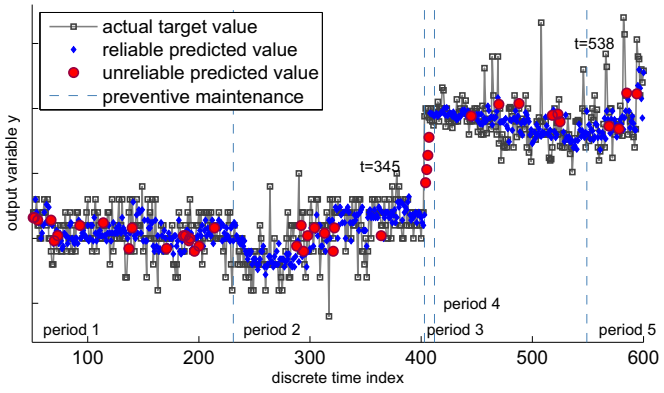

Fig. 9. VM prediction result of the linear PLR-based 5-NN regression.

5. Conclusion

In this paper, we proposed a probabilistic LLR, and extended it to the kernelized version in order to improve the prediction accuracy and to reduce the performance variation for k. It is named probabilistic local reconstruction (PLR). Based on the experimental results, the explicit probabilistic model with Gaussian regularization prior in the reconstruction phase of the PLR avoids over-fitting for a small k. Kernelization rendered the model more flexible than LLR. As a consequence, PLR resulted in higher prediction accuracies as well as lower prediction variations for the critical parameter k in k-NN learning. In addition, we

Table 6
RMSE ($\times 10^{-6}$) values for various regression models.

Predictions	Unweighted	RBF	LLR	PLR-linear	PLR-RBF	LinReg
Overall prediction	1748	1893	11,562	11,490	**1744**	32,461
Reliable prediction	–	–	–	**1605** (86%)	1722 (93%)	–
Unreliable prediction	–	–	–	30,254 (14%)	1992 (7%)	–

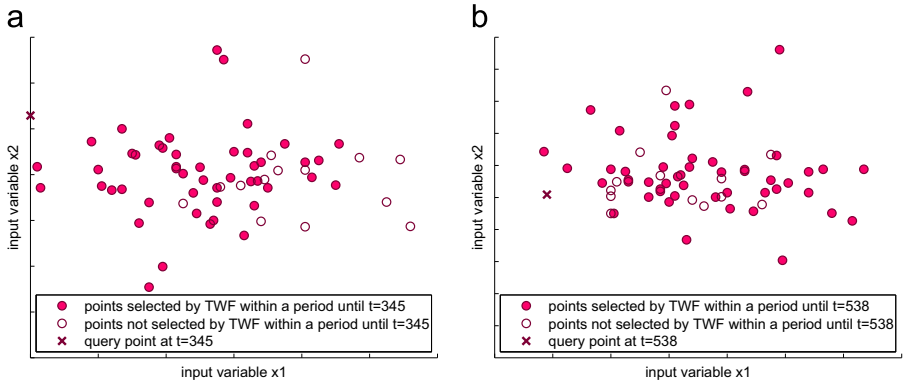

Fig. 10. Why are VM predictions unreliable at $t=345$ and $t=538$? The predictions are based on relatively "unrelated" training points.

demonstrated that our PLR-based *k*-NN regression can be useful in the VM application in the semiconductor industry. Given the reconstruction uncertainty, PLR provides an answer to the question regarding the reliance levels of the predicted virtual measurements, and is expected to reduce the risk in the actual use of VM in the industry.

There are two research issues, which would guide future work. First, it is further investigated to use the reconstruction probability density as a *k*-NN based outlier detection measure for lack of informative training points in the PLR view. We expect that the probability density of a query can be a *k*-NN concentration measure of the One-Class Neighbor Machine (OCNM) [46]. The OCNM based on the sparsity, or concentration measure based on *k* nearest neighbors, provides the decision function whether a new incoming query point is an outlier or not, and is useful in some applications such as the network anomaly detection requiring consistent adaptation of variations in the structure of normality [47].

Second, there is the issue for the use of PLR in changing environments such as our VM application. In the current work, for adaptation of *k*-NN regression, we used the TWF [41], one of the adaptive instance selection methods. However, since TWF considers temporal relevance in the instance selection, it suffers from time-varying sampling distributions [48,41]. Thus, Locally Weighted Forgetting (LWF) [41] was proposed to tackle the limitation. It considers spatial relevance, i.e., a new training point supersedes the previous similar one [41]. We expect that the reconstruction probability density of PLR to be useful in the locally weighted learning and forgetting for changing environments, meriting of further investigation.

Acknowledgments

This work was supported by the Brain Korea 21 project in 2006-2011, the Brain Korea 21 PLUS project in 2013, the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIP) (No. 2011-0030814), Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science, ICT, and Future Planning (2011-0021893) and 2013 Seoul National University Brain Fusion Program Research Grant. This work was also supported by the Engineering Research Institute of SNU.

References

[1] P. Kang, Hyoung-joo Lee, S. Cho, D. Kim, J. Park, C.-K. Park, S. Doh, A virtual metrology system for semiconductor manufacturing, Expert Syst. Appl. 36 (2009) 12554–12561.
[2] P. Kang, D. Kim, Hyoung-joo Lee, S. Doh, S. Cho, Virtual metrology for run-to-run control in semiconductor manufacturing, Expert Syst. Appl. 38 (3) (2011) 2508–2522.
[3] Y. Gan, J. Guan, S. Zhou, A pattern-based nearest neighbor search approach for promoter prediction using dna structural profiles, Bioinformatics 25 (16) (2009) 2006–2012.
[4] H. Li, X. Dai, X. Zhao, A nearest neighbor approach for automated transporter prediction and categorization from protein sequences, Bioinformatics 24 (9) (2008) 1129–1136.
[5] S. Yakowitz, Nearest-neighbor methods for time series analysis, J. Time Ser. Anal. 8 (2) (2008) 235–247.
[6] Ginés Rubio, Luis Javier Herrera, Héctor Pomares, Ignacio Rojas, Alberto Guillén, Design of specific-to-problem kernels and use of kernel weighted k-nearest neighbors for time series modelling, Neurocomputing 73 (2010) 1965–1975.
[7] M. O'mahony, N. Hurley, G. Kushmerick, N. Silvestre, Collaborative recommendation: a robustness analysis, ACM Trans. Internet Technol. 4 (4) (2003) 344–377.
[8] G. Wolberg, Digital Image Warping, IEEE Computer Society Press, Los Alamitos, CA, USA, 1990.
[9] S. Singh, J. Haddon, M. Markou, Nearest-neighbor classifiers in natural scene analysis, Pattern Recognit. 34 (8) (2001) 1601–1612.
[10] X.-Y. Jing, H.-S. Wong, D. Zhang, Face recognition based on discriminant fractional fourier feature extraction, Pattern Recognit. Lett. 27 (13) (2006) 1465–1471.
[11] S.A. Dudani, The distance-weighted k-nearest-neighbor rule, IEEE Trans. Syst. Man Cybern. SMC-6 (4) (1976) 325–327.
[12] Kozak K, M. Kozak, K. Stapor, Weighted k-nearest-neighbor techniques for high throughput screening data, Int. J. Biol. Life Sci. 1 (3) (2006) 155–160.
[13] P. Kang, S. Cho, Locally linear reconstruction for instance-based learning, Pattern Recognit. 41 (2008) 3507–3518.
[14] D. Ruprecht, H. Müller, A Framework for Generalized Scattered Data Interpolation, Technical Report No. 539, Universität Dortmund, Fachbereich Informatik, Dortmund, Germany, 1994.
[15] D.W. Aha, R.L. Goldstone, Concept learning and flexible weighting, in: Proceedings of the Fourteenth Annual Conference of the Cognitive Science Society, Erlbaum, 1992, pp. 534–539.
[16] M.P. Wand, W.R. Schucany, Gaussian-based kernels, Can. J. Stat. 18 (3) (1990) 194–204.
[17] L. Ma, M. Crawford, J. Tian, Local manifold learning-based *k*-nearest-neighbor for hyperspectral image classification, IEEE Trans. Geosci. Remote Sens. 48 (11) (2010) 4099–4109.
[18] S.T. Roweis, L.K. Saul, Nonlinear dimensionality reduction by locally linear embedding, Science 290 (2000) 2323–2326.
[19] L.K. Saul, S.T. Roweis, Think globally, fit locally: unsupervised learning of low dimensional manifolds, J. Mach. Learn. Res. 4 (2003) 119–155.
[20] L. Bottou, V. Vapnik, Local learning algorithms, Neural Comput. 4 (1992) 888–900.
[21] K. Yu, L. Ji, X. Zhang, Kernel nearest-neighbor algorithm, Neural Process. Lett. 15 (2) (2002) 147–156.
[22] C.C. Holmes, N.M. Adams, A probabilistic nearest neighbour method for statistical pattern recognition, J. R. Stat. Soc. Ser. B (Stat. Methodol.) 64 (2) (2002) 295–306.
[23] B. Dubuisson, M. Masson, A statistical decision rule with incomplete knowledge about classes, Pattern Recognit. 26 (1) (1993) 155–165.
[24] T.C.W. Landgrebe, D.M.J. Tax, P. Paclík, R.P.W. Duin, The interaction between classification and reject performance for distance-based reject-option classifiers, Pattern Recognit. Lett. 27 (8) (2006) 908–917.
[25] C.E. Rasmussen, C.K.I. Williams, Gaussian Processes for Machine Learning, MIT Press, 2006.
[26] S. Roberts, W. Penny, D. Pillot, Novelty, confidence and errors in connectionist systems, in: IEE Colloquium on Intelligent Sensors (Digest No: 1996/261), 1996, pp. 10/1–10/6.
[27] D.M.J. Tax, R.P.W. Duin, Support vector data description, Mach. Learn. 54 (1) (2004) 45–66.
[28] B. Schölkopf, A. Smola, K.-R. Müller, Nonlinear component analysis as a kernel eigenvalue problem, Neural Comput. 10 (5) (1998) 1299–1319, URL ⟨http://dx.doi.org/10.1162/089976698300017467⟩.
[29] D. Kim, S. Cho, Bootstrap based pattern selection for support vector regression, in: Proceedings of the 12th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD2008), Lecture Notes in Artificial intelligence, vol. 5012, 2008, pp. 608–615.
[30] D. Coppersmith, S. Winograd, Matrix multiplication via arithmetic progressions, J. Symb. Comput. 9 (1990) 251–280.
[31] A.-J. Su, J.-C. Jeng, H.-P. Huang, C.-C. Yu, S.-Y. Hung, C.-K. Chao, Control relevant issues in semiconductor manufacturing: overview with some new results, Control Eng. Pract. 15 (10) (2007) 1268–1279.
[32] P. Kang, D. Kim, S. kyung Lee, S. Dho, S. Cho, Estimating the reliability of virtual metrology predictions in semiconductor manufacturing: a novelty detection-based approach, J. Korean Inst. Ind. Eng. 38 (1) (2012) 46–56.
[33] C.-F. Chien, C.-Y. Hsu, P.-N. Chen, Semiconductor fault detection and classification for yield enhancement and manufacturing intelligence, Flex. Serv. Manuf. J. 25 (3) (2013) 367–388.
[34] S. Cunningham, C.J. Spanos, K. Voros, Semiconductor yield improvement: results and best practices, IEEE Trans. Semicond. Manuf. 8 (2) (1995) 103–109.
[35] Y.-J. Chang, Y. Kang, C.-L. Hsu, C.-T. Chang, T.Y. Chan, Virtual metrology technique for semiconductor manufacturing, in: International Joint Conference on Neural Networks, 2006. IJCNN '06, 2006, pp. 5289–5293.
[36] P. Chen, S. Wu, J. Lin, F. Ko, H. Lo, J. Wang, C. Yu, M. Liang, Virtual metrology: a solution for wafer to wafer advanced process control, in: IEEE International Symposium on Semiconductor Manufacturing, 2005. ISSM 2005, 2005, pp. 155–157.
[37] T.-H. Lin, M.-H. Hung, R.-C. Lin, F.-T. Cheng, A virtual metrology scheme for predicting cvd thickness in semiconductor manufacturing, in: Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006, 2006, pp. 1054–1059.
[38] J.C. Yung-Cheng, F.-T. Cheng, Application development of virtual metrology in semiconductor industry, in: 31st Annual Conference of IEEE Industrial Electronics Society, 2005. IECON 2005, 2005, 6 pp.
[39] Y.-T. Chen, H.-C. Yang, F.-T. Cheng, Multivariate simulation assessment for virtual metrology, in: Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006, 2006, pp. 1048–1053.
[40] F.-T. Cheng, Y.-T. Chen, Y.-C. Su, D.-L. Zeng, Evaluating reliance level of a virtual metrology system, IEEE Trans. Semicond. Manuf. 21 (1) (2008) 92–103.
[41] M. Salganicoff, Tolerating concept and sampling shift in lazy learning using prediction error context switching, Artif. Intell. Rev. 11 (1–5) (1997) 133–155.
[42] G. Widmer, M. Kubat, Learning in the presence of concept drift and hidden contexts, Mach. Learn. 23 (1) (1996) 69–101.

[43] A. Tsymbal, The Problem of Concept Drift: Definitions and Related Work, Technical Report, 2004.
[44] I. Zliobaite, Learning Under Concept Drift: An Overview, Technical Report, Vilnius University, 2009. URL ⟨http://arxiv.org/pdf/1010.4784.pdf⟩.
[45] D.J. Hand, Classifier technology and the illusion of progress, Stat. Sci. 21 (1) (2006) 1–15.
[46] A. Munoz, J. Moguerza, Estimation of high-density regions using one-class neighbor machines, IEEE Trans. Pattern Anal. Mach. Intell. 28 (3) (2006) 476–480.
[47] T. Ahmed, B. Oreshkin, M. Coates, Machine learning approaches to network anomaly detection, in: Proceedings of the 2nd USENIX Workshop on Tackling Computer Systems Problems with Machine Learning Techniques, SYSML'07, USENIX Association, Berkeley, CA, USA, 2007, pp. 7:1–7:6. URL ⟨http://dl.acm.org/citation.cfm?id=1361442.1361449⟩.
[48] J. Beringer, E. Hüllermeier, Efficient instance-based learning on data streams, Intell. Data Anal. 11 (6) (2007) 627–650.

Seung-kyung Lee received M.S. in 2008 and is currently a Ph.D candidate in the Department of Industrial Engineering, College of Engineering, Seoul National University, Seoul, Republic of Korea. His research interests are instance-based learning, kernel learning machines, Bayesian modeling, and online learning algorithms.

Pilsung Kang is an assistant professor of Information Technology Management (ITM) Program, Seoul National University of Science and Technology (Seoultech). His research interests include instance-based learning, kernel learning machines, novelty detection, graph-based learning, learning algorithms in class imbalance and their applications such as keystroke dynamics based authentication, fault detection in manufacturing process, and customer relationship management.

Sungzoon Cho is a professor in the Department of Industrial Engineering, College of Engineering, Seoul National University, Korea. His research interests are neural network, pattern recognition, data mining, and their applications in various areas such as response modeling and keystroke based authentication. He published over 100 papers in various journals and proceedings. He also holds a US patent and a Korean patent concerned with keystroke-based user authentication.