ABSTRACT

Probabilistic Local Reconstruction for k-Nearest Neighbor Learning

Seung-kyung Lee
Department of Industrial Engineering
The Graduate School
Seoul National University

Locally Linear Embedding (LLE) kernel, one of data-dependent kernels, assesses similarity of two data points in terms of neighboring points in the training set, thus, captures the local topology. Its use in the kernelized k-nearest neighbor (k-NN) prediction was shown to be more robust to k, i.e., "k-invariant" and accurate predictions than other conventional data-independent kernels. The LLE kernel, often named Locally Linear Reconstruction (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, fitting the fixed linear structure to the underlying local topology around a query point may not guarantee the k-invariant property with small or large k settings. Therefore, Probabilistic Local Reconstruction (PLR) as a probabilistic extension of LLR

is proposed and applied to k-NN regression. This probabilistic extension of LLR is explained as the learning process of adjusting the reconstructed point at the average point of k neighbors - expected by the prior assumption - to be close to the corresponding query point. The learning degree is controlled by the additive noise assumption of PLR. High additive noise setting makes PLR avoid the small k problem which is due to no informative neighbors in the locally linear reconstruction. Moreover, the kernelized implementation of PLR is provided to capture the non-linear or locally biased topology embedded in too redundant neighbors with a large k setting. Preliminary experimental result on some benchmark regression problems demonstrated that the proposed Bayesian kernel treatment of LLR improves accuracy and k-invariance. Further, based on the PLR kernel, more reasonable prediction reliance estimation for k-NN regression is proposed. The PLR k-NN prediction reliance takes account of both two uncertainties - reconstruction uncertainty and target uncertainty. From the experiment on a real-world Virtual Metrology (VM) problem where it is critical to provide the reliance level of predictions, it was found that the uncertainty information on the k-NN prediction outcomes provided by the PLR kernel supports more appropriate decision making by rejecting the predictions of data points which would otherwise be unreliably and incorrectly predicted.

.....

Keywords: Data mining, Kernel, Data-dependent kernel, Locally Linear Embedding, Locally Linear Reconstruction, Regression, Prediction reliance, Virtual metrology, Semiconductor manufacturing

Student Number: 2008–30307

Notation

Notations used in this dissertation.

$\mathcal{X}$	: the input space or domain of samples
N	: the number of samples for training or training points
$\mathbf{X}$	: the training points $\mathbf{X} = \{\mathbf{x}_n \in \mathcal{X} = \mathbb{R}^D\}_{n=1}^N$
$\mathcal{Z}$	: the lower-dimensional space, co-domain of a mapping function for dimensionality reduction, $\mathbf{z} : \mathcal{X} \to \mathcal{Z}$
$\mathbf{Z}$	: the training points embedded in the lower-dimensional space, $\mathbf{Z} = \{\mathbf{z}_n \in \mathcal{Z} = \mathbb{R}^d\}_{n=1}^N$, $d < D$
$\mathbf{z}(\mathbf{x})$	: the mapping function for dimensionality reduction or embedding point $\mathbf{z}(\mathbf{x}) = \mathbf{z}$ of an out-of-sample point $\mathbf{x}$
$\mathcal{Y}$	: the co-domain of a predictive function, $y : \mathcal{X} \to \mathcal{Y}$, conventionally known as the target space (1) if $\mathcal{Y} = \mathbb{R}$, the predictive function $y(\mathbf{x})$ becomes a regressor (2) if $\mathcal{Y} = \{1, ..., C\}$, where C is the number of classes, the function $y(\mathbf{x})$ becomes a classifier
$\mathbf{Y}$	: the known target values of training points, $\mathbf{Y} = \{y_n \in \mathcal{Y}\}_{n=1}^N$
$y(\mathbf{x})$	: the prediction function $y : \mathcal{X} \to \mathcal{Y}$ or predicted target value $y(\mathbf{x}) = y$ of a point $\mathbf{x}$

$\|\|\mathbf{x}_i - \mathbf{x}_j\|\|$	: Euclidean distance between $\mathbf{x}_i$ and $\mathbf{x}_j$
$\mathbf{1}_{k\times 1}$	: the $k \times 1$ column vector with every element being one
δ_{ij}	: a Kronecker delta which is one if $i = j$ and zero otherwise
$\mathbf{k}(\cdot,\cdot)$	: the kernel function $\mathbf{k} : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, $(\mathbf{x}_i, \mathbf{x}_j) \mapsto \mathbf{k}(\mathbf{x}_i, \mathbf{x}_j)$
$\phi(\cdot)$	: a mapping from a data point $\mathbf{x}$ into one in a higher-dimensional feature space $\mathbf{x} \mapsto \phi(\mathbf{x})$, which is not explicitly defined by the kernel trick, $\mathbf{k}(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j)$
$\mathbf{K}$	: the kernel matrix or Gram matrix, $\mathbf{K}_{ij} := \mathbf{k}(\mathbf{x}_i, \mathbf{x}_j)$
$nn(\mathbf{x})$	: the index set of neighbors for $\mathbf{x}$ among training points $\mathbf{X}$
$nn(\mathbf{x}, i)$	: the index of the i^{th} neighbors for $\mathbf{x}$ among $\mathbf{X}$
$\mathbf{x}_{nn(\mathbf{x},i)}$	: the i^{th} nearest neighbors of a point $\mathbf{x}$, $\mathbf{x}_{nn(\mathbf{x},i)} \in \mathbf{X}$
$\mathbf{X}_{nn(\mathbf{x})}$	: $D \times k$ "neighbor" matrix, $\mathbf{X}_{_{NN(\mathbf{x})}} = [\mathbf{x}_{nn(\mathbf{x},1)}, ..., \mathbf{x}_{nn(\mathbf{x},k)}]$
$\mathbf{k}_{lle}(\cdot,\cdot)$	: LLE kernel
$\mathbf{x}_{recon}$	: a reconstructed point of a data point $\mathbf{x}$ by LLR or PLR
$w_{llr}(\mathbf{x}, \mathbf{x}_n)$	: the LLR coefficient of $\mathbf{x}_n \in \mathbf{X}$ for reconstruction of $\mathbf{x}$ $\sum_{n=1}^{N} w_{llr}(\mathbf{x}, \mathbf{x}_n) = 1$ and $w_{llr}(\mathbf{x}, \mathbf{x}_{nn(\mathbf{x},i)}) = 0$ for $\forall i > k$
$\mathbf{w}_{llr}(\mathbf{x})$	: $k \times 1$ LLR coefficient vector, $\mathbf{w}_{llr} := \mathbf{w}_{llr}(\mathbf{x})$ for convenience $\mathbf{w}_{llr} = [w_{llr}(\mathbf{x}, \mathbf{x}_{nn(\mathbf{x},1)}), ..., w_{llr}(\mathbf{x}, \mathbf{x}_{nn(\mathbf{x},k)})]^T$ $\mathbf{w}_{llr}^T \mathbf{1}_{k\times 1} = 1$
$E(\mathbf{w}_{llr}; \mathbf{x})$	: the reconstruction error function of LLR, given $\mathbf{x}$ $E(\mathbf{w}_{llr}; \mathbf{x}) = \frac{1}{2}(\mathbf{x} - \mathbf{x}_{recon})^T (\mathbf{x} - \mathbf{x}_{recon})$
$\mathbf{W}_{llr}$	: $\mathbf{W} := \mathbf{W}_{llr}$ for convenience the $N \times N$ overlapped matrix of LLR coefficient vectors for $\forall \mathbf{x}_n \in \mathbf{X}$, $\mathbf{W}_{ij} := w_{llr}(\mathbf{x}_i, \mathbf{x}_j)$

$\mathbf{C}$	: $k \times k$ local Gram matrix, $\mathbf{C} = \mathbf{X}_{nn(\mathbf{x})}^T \mathbf{X}_{nn(\mathbf{x})}$
$\tilde{\mathbf{C}}$	: $k \times k$ local Gram matrix centered on $\mathbf{x}$, $\tilde{\mathbf{C}} = \tilde{\mathbf{X}}_{nn(\mathbf{x})}^T \tilde{\mathbf{X}}_{nn(\mathbf{x})} = (\mathbf{X}_{nn(\mathbf{x})} - \mathbf{x}\mathbf{1}_{k\times 1}^T)^T (\mathbf{X}_{nn(\mathbf{x})} - \mathbf{x}\mathbf{1}_{k\times 1}^T)$
$\Phi(\mathbf{Z}; \mathbf{W})$	: the embedding cost function of LLE, given $\mathbf{W}$
$\mathbf{w}$	: the $k \times 1$ weight vector of the weighted k-NN prediction, $\mathbf{w} = [w_1, ..., w_k]^T$
$y_{nn(\mathbf{x},i)}$	: the corresponding target values of the i^{th} neighbors
$\mathbf{y}_{nn(\mathbf{x})}$	: $k \times 1$ vector of the target values of k neighbors
$\bar{y}_{nn(\mathbf{x})}$	: an average of the target values of k neighbors, $\bar{y}_{nn(\mathbf{x})} = k^{-1} \sum_i y_{nn(\mathbf{x},i)}$
β_{recon}	: the inverse noise variance in PLR, $\beta_{recon} := \sigma_{recon}^{-2}$
ϵ_{recon}	: the reconstruction uncertainty, the additive Gaussian noise in PLR, $\epsilon_{recon} \sim N(\mathbf{0}, \beta_{recon}^{-1}\mathbf{I})$
μ	: the posterior mean for PLR coefficient vector
Σ	: the posterior covariance for PLR coefficient vector
$w_{plr}(\mathbf{x}, \mathbf{x}_n)$	: the PLR coefficient of $\mathbf{x}_n \in \mathbf{X}$ for reconstruction of $\mathbf{x}$
$\mathbf{w}_{plr}(\mathbf{x})$	: $\mathbf{w}_{plr} := \mathbf{w}_{plr}(\mathbf{x})$ for convenience the probabilistic estimate of PLR, $N(\mathbf{w}_{plr}\|\mu, \Sigma)$
γ	: a constant such that $\gamma \sum_i \mu_i = \gamma \mu^T \mathbf{1}_{k\times 1} = 1$
$\tilde{\mathbf{K}}$	: $k \times k$ centered kernel matrix,
$\mathbf{k}(\mathbf{x}, \mathbf{X}_{nn(\mathbf{x})})$	: the $k \times 1$ column vector consisting of the kernel functions of k neighbors for a query point $\mathbf{k}(\mathbf{x}, \mathbf{X}_{nn(\mathbf{x})}) = [\mathbf{k}(\mathbf{x}, \mathbf{x}_{nn(\mathbf{x},1)}), ..., \mathbf{k}(\mathbf{x}, \mathbf{x}_{nn(\mathbf{x},k)})]^T$
σ^2	: the predictive variance of a regressor
w_{TWF}	: weight for time-weighted forgetting (TWF)

η	: forgetting or decaying ratio of TWF
θ	: threshold for selecting training points in TWF
β_{target}	: the inverse noise variance of Gaussian process (GP) regression, $\beta_{target} := \sigma_{target}^{-2}$
ϵ_{target}	: the target uncertainty, the additive Gaussian noise in GP, $\epsilon_{target} \sim N(\mathbf{0}, \beta_{target}^{-1}\mathbf{I})$
$\mathbf{k}(\mathbf{x}, \mathbf{X})$	: the $N \times 1$ column vector consisting of the kernel functions of N samples for a query point

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Knowledge Discovery in Databases and Data Mining

Knowledge Discovery in Databases (KDD) refers to "the nontrivial process of identifying valid, novel, potentially useful, and ultimately understandable patterns, i.e., knowledge in data" (Fayyad et al., 1996). Here, the "pattern" or "regularity" is an expression or model for a set of data, which are too voluminous to be manually analyzed. Data mining in a narrow sense is a particular step in the KDD process and the application of specific algorithms for (semi-) automatically extracting patterns from data, e.g., fitting a model or function to data or finding more abstract description of a subset of data (Fayyad et al., 1996). On the other hand, data mining in a broad sense often refers to the application of algorithms utilized throughout the KDD process (Shmueli et al., 2011). In this way, KDD or data mining is an attempt for scaling up - currently, scaling "out" (Lin et al., 2006) - the manual analysis capabilities to handle the overwhelming volume of data (Fayyad et al., 1996).

And there are some typical data mining tasks as follows. First, clustering is the task to build - commonly "learn" - a function mapping data

points - usually represented as a vector of measurements - into groups so that "similar" points are mapped into the same group (Jain et al., 1999; Maimon and Rokach, 2010). Conventionally, for given N data points $\mathbf{X} = \{\mathbf{x}_n \in \mathcal{X} = \mathbb{R}^D\}_{n=1}^N$, the mapping function is defined as $f : \mathcal{X} \to \{1, ..., K\}$, where K is the number of clusters. It is a descriptive analysis derived from a need for understanding the salient characteristics of objects and identifying them with a type, i.e., pattern (Fayyad et al., 1996; Maimon and Rokach, 2010).

Second, dimensionality reduction or manifold learning is to learn a function mapping data points into a lower dimensional space while preserving the hidden structure of the data (Burges, 2009; Saul et al., 2006) - local topology, i.e., "similar" data points in the input space are ideally mapped to nearby points on the lower dimensional space (Hadsell et al., 2006; Kayo, 2006). When training points embedded in the lower dimensional space or manifold are denoted as $\mathbf{Z} = \{\mathbf{z}_n \in \mathcal{Z} = \mathbb{R}^d\}_{n=1}^N$, where $D > d$, the mapping function is defined as $\mathbf{z} : \mathcal{X} \to \mathcal{Z}$. It is derived from the belief that objects in our world are observed in terms of too many "extrinsic" and "coherent" measurements, but described - surprisingly - in terms of a few "intrinsic" features (Roweis and Saul, 2000). This lower dimensional description does not only lead us to better model for subsequent data mining tasks, but also contribute to an interpretation of data through visualization (Burges, 2009; Ghodsi, 2006).

Third, prediction - classification and regression - is the task to learn a function mapping data points into unknown or future values of another variable of interest - i.e., "target" variable (Fayyad et al., 1996). This learning of the predictive function is commonly referred to "supervised" learning in that we are given N data points along with corresponding and already known target values $\mathbf{Y} = \{y_n \in \mathcal{Y}\}_{n=1}^N$. The target values may be continuous, i.e., $\mathcal{Y} = \mathbb{R}$ (in regression), or discrete, i.e., class labels $\mathcal{Y} = \{1, ..., C\}$ where C is the number of classes (in classification) (Rasmussen

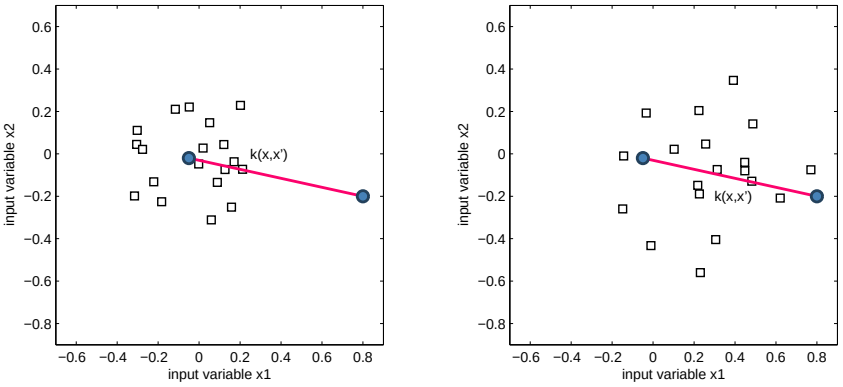
and Williams, 2006; Tipping, 2001). Then the function is inferred for the underlying relationship between the data points and corresponding target values, and defined as $y : \mathcal{X} \to \mathcal{Y}$. We expect that the function would be predictive with the smoothness assumption - i.e., when two data points are close, the corresponding target values are also close (Zhang et al., 2000)

1.2 Kernel: Conceptualizing Similarities and Differences

"Understanding our world requires conceptualizing the similarities and differences between the entities that compose it" (Tryon and Bailey, 1970). One of key questions shared among the typical data mining tasks is how to characterize similarities or dissimilarities between data points. A similarity or affinity matrix representing the pair-wise similarity is the core structure containing all necessary information for learning most data mining models. Moreover the "kernel" concept has been investigated for that, i.e., informally, it is a similarity measure between two data points, $\mathbf{x}_i$ and $\mathbf{x}_j$ (Chen et al., 2009; Scholköpf and Smola, 2002) as follows,

$$\mathbf{k} : \mathcal{X} \times \mathcal{X} \to \mathbb{R}, \quad (\mathbf{x}_i, \mathbf{x}_j) \mapsto \mathbf{k}(\mathbf{x}_i, \mathbf{x}_j). \tag{1.1}$$

Particularly, the beauty of using the concept lies in the "implicit" use of a dot-product space - usually called feature space - induced by the class of (positive definite) kernels (Chen et al., 2009; Hofmann et al., 2008; Scholköpf and Smola, 2002). By computing dot products in feature space by means of kernel functions in input space - it is the kernel trick -, we can apply simple, e.g., linear models in the "abstract" feature space to extracting complex, e.g., nonlinear patterns in "practical" and real-world problems (Hofmann et al., 2008; Schölkopf et al., 1998). In this way, a $N \times N$ similarity matrix in a dot-product space, called kernel matrix or



(a) (b)

Figure 1.1: Are the kernel values equal?

Gram matrix $\mathbf{K}$ is expressed in terms of kernels, i.e., $\mathbf{K}_{ij} := \mathbf{k}(\mathbf{x}_i, \mathbf{x}_j)$. The methods for learning models from the Gram matrix, i.e., kernel methods have been popularly researched over the last ten years (Hofmann et al., 2008), e.g., Support Vector Machine (SVM) (Vapnik, 1995) for classification and kernel Principal Component Analysis (KPCA) (Schölkopf et al., 1998) for dimensionality reduction, etc.

And the performance of the kernel methods strongly depends on the choice of appropriate kernels well-conceptualizing "similarity" between two entities in the real-world problem to be solved. There are some widely used kernels such as polynomial kernel $\mathbf{k}(\mathbf{x}_i, \mathbf{x}_j) = (c + \mathbf{x}_i^T \mathbf{x}_j)^p$ where the degree $p \in \mathbb{N}$ and $c \geq 0$, inverse distance kernel $\mathbf{k}(\mathbf{x}_i, \mathbf{x}_j) = (1 + ||\mathbf{x}_i - \mathbf{x}_j||^p)^{-1}$ (Wolberg, 1990), exponential kernel (Aha and Goldstone, 1992) and Gaussian kernel (or radial basis function, RBF kernel) $\mathbf{k}(\mathbf{x}_i, \mathbf{x}_j) = exp(-\frac{||\mathbf{x}_i - \mathbf{x}_j||^2}{2l^2})$ where the kernel width $l > 0$ (Wand and Schucany, 1990). This class of kernels is sometimes called data-independent kernel in that they depend only on $\mathbf{x}_i$ and $\mathbf{x}_j$ (Bengio et al., 2004). These kernels are simply proportional to a dot product similarity in that input space (in the case of the

polynomial kernel) or inversely proportional to a distance measure (in the case of inverse distance, exponential and Gaussian kernel). In this way, the data-independent kernels do not capture geometry of neighboring data points. Fig. 1.1 shows why the local geometric structure should be taken account of. Two points $\mathbf{x}_i$ and $\mathbf{x}_j$ denoted as filled circles in Fig. 1.1(a) and (b) are the same. Thus, in the case of data-independent kernels, the kernel values $\mathbf{k}(\mathbf{x}_i, \mathbf{x}_j)$ of the two pairs become equivalent. However, the two pairs have different local geometry, made up of data points denoted squares in the figure. It is desirable that the pair in Fig. 1.1(b) is more similar than one in Fig. 1.1(a).

On the other hand, data-dependent kernels was induced from some local manifold learning methods such as Locally Linear Embedding (LLE) (Roweis and Saul, 2000), Isomap (Tenenbaum et al., 2000) and Laplacian Eigenmaps (LE) (Belkin and Niyogi, 2003), which utilize local information for nonlinear dimensionality reduction. In kernel view, these local manifold learning methods are interpreted as performing kernel PCA (Schölkopf et al., 1998) on the Gram matrix $\mathbf{K}$ defined for each of these method (Bengio et al., 2004, 2003a; Ham et al., 2004). Each element in the gram matrix can be defined as a kernel, i.e., $\mathbf{K}_{ij} := \mathbf{k}(\mathbf{x}_i, \mathbf{x}_j)$, which is called data-dependent kernel in that the $\mathbf{k}(\mathbf{x}_i, \mathbf{x}_j)$ depends not only on $\mathbf{x}_i$ and $\mathbf{x}_j$, but also other training points $\mathbf{x} \in \mathbf{X}$ (Bengio et al., 2004). In other word, the kernel assesses similarity of two data points in terms of neighboring points in the training set (Scholköpf and Smola, 2002), thus, captures the local geometry, i.e., local topology. For example, the kernel of Isomap is related to the Dijkstra's shortest path distance, while the one of LE is related to commute times (Ham et al., 2004). Further, defining a kernel for each methods generalizes the embedding to out-of-sample points $\mathbf{x} \notin \mathbf{X}$ in terms of the kernel functions, i.e., $\mathbf{k}(\mathbf{x}, \mathbf{x}_i)$, $\mathbf{x}_i \in \mathbf{X}$ with other embedding points via Nyström formula, which is used to speed up kernel methods (Bengio et al., 2004, 2003a; Williams and Seeger, 2001) - originally, there

was no straightforward extension for out-of-samples in the manifold learning methods.

1.3 Locally Linear Embedding Kernel and k-NN Prediction

In this dissertation, the data-dependent kernel composing the Gram matrix for LLE, which is called LLE kernel (Scholköpf and Smola, 2002) is considered. LLE is a local manifold learning method for non-linear dimensionality reduction while preserving the local topology in the neighborhood of each data point (Roweis and Saul, 2000). The corresponding kernel is based on the Locally Linear Reconstruction (LLR) operation to capture the local topology on the locally linear subspaces spanned by neighboring data points (Bengio et al., 2004, 2003a; Ham et al., 2004). The advantages of LLE kernel are 1) preserving the local topology unlike the data-independent kernels - it is not free lunch, but - 2) computationally efficient, i.e., LLE kernel requires solving the constrained least squares problem, while the data-dependent kernel for Isomap requires a large dynamic programming problem (Roweis and Saul, 2000). In this way, LLE kernel has been successfully used not only in dimensionally reduction and clustering - in conjunction with clustering algorithms (Kim and Finkel, 2003; Zhou et al., 2009), but also in k-nearest neighbor (k-NN) prediction, which is conventionally called as k-NN learning (Kang and Cho, 2008; Ma et al., 2010).

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 and its corresponding target value, k-NN prediction defers learning or training until a query point $\mathbf{x}$ is given. Its predictive function $y(\mathbf{x})$, which outputs the target value of a query point, then refers target values of k neighbors selected among the training points. In this

point of view, k-NN prediction is one of the simplest implementation of the smoothness assumption. 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 (Kang et al., 2009, 2011), bioinformatics (Gan et al., 2009; Li et al., 2008), time series analysis (Rubio et al., 2010; Yakowitz, 2008), collaborative filtering (O'mahony et al., 2003), and image processing (Jing et al., 2006; Singh et al., 2001; Wolberg, 1990).

And the contribution of each neighbor to the predictive function can be weighted according to its relevance to the query point, in that some neighbors are often more relevant to a query than others in many practical cases (Dudani, 1976; Kozak et al., 2006). Some experimental results showed that such weighted approach could achieve lower error rate than equally-weighted or un-weighted one (Macleod et al., 1987; Zavrel, 1997). Thus, (weighted) k-NN learning is a variation of locally weighted learning (Kang and Cho, 2008; Mitchell, 1997). Moreover, the kernelized k-NN prediction was proposed to substitue a kernel distance metric for the conventional ones such as Euclidean distance the in input space (Yu et al., 2002). In the weighted k-NN prediction, the kernelized implementation can be achieved by employing some kernels as the weights for k nearest neighbors. Finally, using LLR kernel[1], rather than data-independent kernels in the weighted or kernelized k-NN learning was shown to be more robust to its locally parameter k - i.e., k-invariant - and improves prediction performance (Kang and Cho, 2008; Ma et al., 2010). As pointed out

[1]Kang and Cho (2008) applied the LLE kernel without solving a sparse eigenvector problem to k-nearest neighbor prediction. It is the reason that the kernel is named Locally Linear Reconstruction (LLR). In the strict sense, the LLR kernel is a special case of the LLE kernel between training points and an out-of-sample point - a query point whose target value is to be predicted in the case of k-NN prediction. In this dissertation, 'LLR kernel' and 'LLE kernel' are used accordingly.

earlier, it is due to that LLE kernel conceptualizes similarity between two data points, taking account of local topology around them.

Despite of the effectiveness of the data-dependent LLE kernel, there still remain some research issues for LLE kernel to overcome. In this dissertation, two major issues of LLE or LLR kernel in the kernelized k-NN learning - particularly, regression - are studied:

- **Flexibility**

  The fixed linear structure of LLR 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. As pointed out by (Saul and Roweis, 2003), when there are more neighbors than input dimensions ($k > D$), or when the data points are not in general position, a regularization term for penalizing large weights is required in order to be robust to noise or outliers. On the other hand, with a large k, the non-linear topology embedded in too redundant neighbors cannot be captured. The fundamental assumption of LLR - a query and its k neighbors are geometrically assumed to lie on or close to a locally linear subspaces - is violated (Saul and Roweis, 2003).

- **"How informative is the set of k neighbors in LLR?"**

  In practice, neighbors for a query point are often not informative to describe the query. It is often need to provide an answer for the question, "how informative is the set of k neighbors in LLR?" In general, classifiers or regressor only provide reliable estimates for points in regions where training points are densely sampled. On the other hand, extrapolations to extremely sparse or unknown regions in input space become uncertain (Roberts et al., 1996; Tax and Duin, 2004). Moreover, when we consider also the target uncertainty -

ambiguity by inconsistent or severely dispersed target values of k neighbors (Dubuisson and Masson, 1993; Landgrebe et al., 2006) - violating the smoothness assumption, it is desirable to provide a probabilistic estimate for incorporating with the target uncertainty represented as predictive variance of the k-NN prediction, but LLR provides a point estimate.

1.4 Contributions of this Dissertation

In this dissertation, a Bayesian kernel model of LLR, named Probabilistic Local Reconstruction (PLR) is proposed in order to handle the limitations of LLR in k-NN regression. The main contributions of this dissertation are following:

- **Improving k-invariance and accuracy (in Chapter 3)**

  The probabilistic extension of LLR (explained in Section 3.1) is proposed in order to handle the small k problem in the kernelized k-NN regression using LLR kernel. First, an explicit probabilistic model for the locally linear reconstruction is employed. Then, incorporating Gaussian prior into the probabilistic model allows one to explain equally-weighted k-NN prediction, i.e., the output of the local constant fit. In this way, unlike the non-probabilistic LLR, the probabilistic extension is explained as the learning process to adjust the reconstructed point at the average point of k neighbors - expected by the prior assumption - to be close to the corresponding query point. The interpretation intuitively explains how the PLR kernel can be robust to noise or outliers.

  And the kerenlized implementation for LLR (explained in Section 3.2) is provided - i.e., transforming a lower-dimensional input space into a higher-dimensional feature space without the need for explicit

mapping function. Fitting the 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 kernel in some regressions. The kernelized implementation helps the model capture the non-linear local topology in a large k setting, and be robust to a wider range of k values. Consequently, it is demonstrated 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.

- **Quantifying the reconstruction uncertainty (in Chapter 4)**

  The probabilistic estimate of PLR quantifies the reconstruction uncertainty due to lack of neighboring points, and provides the answer for the question "how informative the set of k neighbors in the locally linear reconstruction of a query?" It is noted that the reconstruction probability may be used as an outlier detection measure, like the reconstruction error of the non-probabilistic LLR - the Euclidean distance between a query and its reconstructed or projected point in the space spanned by its k neighbors (Kang and Cho, 2009). However, in the case of regression or classification problems, the probabilistic estimate quantifying the reconstruction uncertainty is preferred in that the reconstruction uncertainty can be incorporated with the target uncertainty as a form of "error-bars." In this way, the predictive variance based on PLR kernel takes account of both two cases - extrapolation and ambiguity cases, and provides more reasonable prediction reliance for k-NN regression (explained in Section 4.1). It is expected to be useful in regression problems where the extrapolation issue is critical.

  And one of the regression problems, Virtual Metrology (VM) in the semiconductor industry is introduced. VM is an important issue in

wafer manufacturing. Those wafers whose properties are not physically measured are virtually predicted by a regressor. In the application problem, sampling shift (Salganicoff, 1997) - commonly known as process drift in the domain - frequently occurs and raises the prediction reliance issue due to extrapolations. therefore, it is practically critical to provide the reliance level of the virtual predictions (explained in Section 4.2). It is demonstrated that PLR provides an appropriate reliance level for predictions.

The reminder of this dissertation is organized as follows. In Chapter 2, literature reviews of the data-dependent kernel, LLE kernel and its use in the kernelized k-NN regression are presented. In Chapter 3, Probabilistic Local Reconstruction is proposed with experimental results on benchmark data sets. In Chapter 4, estimation of prediction reliance using PLR is proposed, and one of its application, Virtual Metrology is introduced with experimental results on a real-world VM problem. Finally, Chapter 5 concludes this dissertation with the summary and discuss future works.

Chapter 2

Literature Review

In Section 2.1, Locally Linear Embedding (LLE) is introduced. Briefly, LLE is a local manifold learning method for learning a topology preserving mapping of data points in input space into ones in lower-dimensional space (Roweis and Saul, 2000). For that, LLE characterizes the local topology in the neighborhood of each data point by locally linear subspaces. Locally Linear Reconstruction (LLR) is the operation for capturing the "locally linear topology". It then constructs a global embedding of the manifold by solving a sparse eigenvector problem of the matrix $\tilde{\mathbf{K}}$ obtained from the local topologies of data points. Further, in Section 2.2, the interpretation of LLE in a kernel view is explained. That is, the construction of a global embedding of the manifold is interpreted as kernel PCA for a (positive definite) Gram matrix $\mathbf{K}$, expressed in terms of LLE kernels, i.e., $\mathbf{K}_{ij} := \mathbf{k}_{lle}(\mathbf{x}_i, \mathbf{x}_j)$ (Bengio et al., 2004, 2003a; Ham et al., 2004; Scholköpf and Smola, 2002). This interpretation gives rise to the generalized kernel, which is defined not only among sample points $\mathbf{x}_n \in \mathbf{X}$, but also out of samples $\mathbf{x} \notin \mathbf{X}$. Finally, as an application to supervised learning problem, the approach to use LLE kernel in k-nearest neighbor (k-NN) prediction (Kang and Cho, 2008; Ma et al., 2010) is introduced in Section 2.3.

2.1 Locally Linear Embedding for Manifold Learning

Identifying neighbors of each data point $\mathbf{x} \in \mathbf{X}$, - conventionally, k nearest neighbors in Euclidean space, $\mathbf{x}_{nn(\mathbf{x},i)} \in \mathbf{X}$, $i = 1, ..., k$ (Saul and Roweis, 2003) where $nn(\mathbf{x}, i)$ is the index of the i^{th} neighbors for $\mathbf{x}$ among the training points $\mathbf{X}$, LLE assumes that the data point is described as a linear combination of its neighbors. Let $\mathbf{X}_{nn(\mathbf{x})} = [\mathbf{x}_{nn(\mathbf{x},1)}, ..., \mathbf{x}_{nn(\mathbf{x},k)}]$ denote the "neighbor" matrix where the i^{th} column vector is the i^{th} neighbor of a data point $\mathbf{x}$. The description of $\mathbf{x}$ in terms of k neighbors, i.e., geographically, a reconstructed point $\mathbf{x}_{recon}$ on the locally linear subspace, i.e., hyper-plane spanned by k neighbors (Kang and Cho, 2008; Roweis and Saul, 2000) is defined as follows,

$$\mathbf{x}_{recon} = \sum_{i=1}^{k} w_{llr}(\mathbf{x}, \mathbf{x}_{nn(\mathbf{x},i)})\mathbf{x}_{nn(\mathbf{x},i)} = \mathbf{X}_{nn}\mathbf{w}_{llr}, \tag{2.1}$$

where $w_{llr}(\mathbf{x}, \mathbf{x}_i)$ is the LLR coefficient of $\mathbf{x}_i$ for reconstruction of $\mathbf{x}$, and 0 if $\mathbf{x}_i$ is not in the k nearest neighbors of $\mathbf{x}$, i.e., $i \notin nn(\mathbf{x})$. Then, the coefficient vector on the locally linear subspace, $\mathbf{w}_{llr} := \mathbf{w}_{llr}(\mathbf{x})$ for convenience where $\mathbf{w}_{llr}(\mathbf{x}) = [w_{llr}(\mathbf{x}, \mathbf{x}_{nn(\mathbf{x},1)}), ..., w_{llr}(\mathbf{x}, \mathbf{x}_{nn(\mathbf{x},k)})]^T$ is constrained to have the sum of one, i.e., $\sum_{i=1}^{k} w_{llr}(\mathbf{x}, \mathbf{x}_{nn(\mathbf{x},i)}) = \mathbf{w}_{llr}^T \mathbf{1}_{k\times 1} = 1$ where $\mathbf{1}_{k\times 1}$ is the $k \times 1$ column vector with every element being one (Roweis and Saul, 2000). The coefficient vector is often called "locally linear topology" to be preserved by LLE with dimensionality reduction. In the topology, neighbors of positive coefficient pull the data point, while ones of negative coefficient push it (Kang and Cho, 2008).

And Locally Linear Reconstruction (LLR) is to capture the locally linear topology to minimize the reconstruction error, which is defined as

difference between the data point $\mathbf{x}$ and its reconstructed point $\mathbf{x}_{recon}$:

$$\begin{aligned} E(\mathbf{w}_{llr};\mathbf{x}) &= \frac{1}{2}(\mathbf{x}-\mathbf{x}_{recon})^T(\mathbf{x}-\mathbf{x}_{recon}) \\ &= \frac{1}{2}(\mathbf{x}-\mathbf{X}_{nn(\mathbf{x})}\mathbf{w}_{llr})^T(\mathbf{x}-\mathbf{X}_{nn(\mathbf{x})}\mathbf{w}_{llr}) \\ &= \frac{1}{2}\mathbf{x}^T\mathbf{x} - \mathbf{w}_{llr}^T(\mathbf{X}_{nn(\mathbf{x})}^T\mathbf{x}) + \frac{1}{2}\mathbf{w}_{llr}^T\mathbf{C}\mathbf{w}_{llr}, \end{aligned} \tag{2.2}$$

where $\mathbf{C} = \mathbf{X}_{nn(\mathbf{x})}^T\mathbf{X}_{nn(\mathbf{x})}$ is a $k\times k$ local Gram matrix. The explicit solution $\mathbf{w}_{llr}$ is obtained as Eq. (2.3) by solving the constrained least squares problem and shifting the locally linear space so that the data point is its origin, i.e., subtracting the data point from all neighbors and the point itself (Ma et al., 2010).

$$\mathbf{w}_{llr} = \sum_i \tilde{\mathbf{C}}_{i,j}^{-1} / \sum_{l,m} \tilde{\mathbf{C}}_{l,m}^{-1} \tag{2.3}$$

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 (Roweis and Saul, 2000):

$$\tilde{\mathbf{C}} \leftarrow \tilde{\mathbf{C}} + \Delta\mathbf{I}. \tag{2.4}$$

Lower-dimensional points, i.e., embedding points $\mathbf{Z} = \{\mathbf{z}_n \in \mathbb{R}^d\}_{n=1}^N$ preserving the locally linear topology is obtained by minimizing the embedding cost function (Roweis and Saul, 2000) defined as follows:

$$\Phi(\mathbf{Z};\mathbf{W}) = \sum_{i=1}^N |\mathbf{z}_i - \sum_{j=1}^N \mathbf{W}_{ij}\mathbf{z}_j|^2. \tag{2.5}$$

where the element $\mathbf{W}_{ij}$ composing a $N \times N$ matrix $\mathbf{W}$ is the LLR coefficient of $\mathbf{x}_j$ for reconstruction of $\mathbf{x}_i$, i.e., $\mathbf{W}_{ij} := w_{llr}(\mathbf{x}_i, \mathbf{x}_j)$. In this way, $\sum_j \mathbf{W}_{ij} = 1$. Embedding points are constrained to $\sum_i \mathbf{z}_i = 0$

and $\frac{1}{N}\sum_i \mathbf{z}_i\mathbf{z}_i^T = \mathbf{I}$. Further, when the cost function is transformed to $\Phi(\mathbf{Z};\mathbf{W}) = \sum_{ij}\tilde{\mathbf{K}}(\mathbf{z}_i^T\mathbf{z}_j)$, where the sparse matrix $\tilde{\mathbf{K}} = (\mathbf{I}-\mathbf{W})^T(\mathbf{I}-\mathbf{W})$, the optimal embedding $\mathbf{Z} \in \mathbb{R}^d$ is obtained by computing the bottom $d+1$ eigenvectors of $\tilde{\mathbf{K}}$[1] by Rayleitz-Ritz theorem (Horn and Johnson, 1990) - it is the sparse eigen-value problem (Roweis and Saul, 2000; Saul and Roweis, 2003).

2.2 LLE in a Kernel View

In a kernel view, LLE is interpreted as kernel PCA for the positive definite matrix $\mathbf{K} := \lambda_{max}\mathbf{I} - \tilde{\mathbf{K}}$ based on LLR (Bengio et al., 2004, 2003a; Ham et al., 2004). The coordinates of the top $d+1$ eigenvectors of $\mathbf{K}$ provide the LLE embeding. Bengio et al. (2004) define the generalized kernel $\mathbf{k}(\mathbf{x}_i, \mathbf{x}_j)$ for computing the matrix $\mathbf{K}$, regardless whether data points is an out-of-sample or not. That is, for any two data points $\mathbf{x}_i$ and $\mathbf{x}_j$, the LLE kernel $\mathbf{k}_{lle}$ is defined as follows:

$$\mathbf{k}_{lle} = (\lambda_{max} - 1)\mathbf{k}' + \mathbf{k}''. \tag{2.6}$$

In the kernel definition, the out-of-sample extension term $\mathbf{k}'(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{k}'(\mathbf{x}_j, \mathbf{x}_i)$ is defined as follows:

$$\mathbf{k}'(\mathbf{x}_i, \mathbf{x}_j) = \begin{cases} \delta_{ij} & \text{if } \mathbf{x}_i, \mathbf{x}_j \in \mathbf{X} \\ 0 & \text{if } \mathbf{x}_i, \mathbf{x}_j \notin \mathbf{X} \\ w_{llr}(\mathbf{x}, \mathbf{x}_j) & \text{if } \mathbf{x}_i = \mathbf{x} \notin \mathbf{X} \text{ and } \mathbf{x}_j \in \mathbf{X} \end{cases} \tag{2.7}$$

where δ_{ij} is a Kronecker delta which is one iff $i = j$ and zero otherwise. Moreover, the within-sample term $\mathbf{k}''(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{k}''(\mathbf{x}_j, \mathbf{x}_i)$ is defined as

[1] the smallest eigenvalue is zero and the corresponding eigenvector is the uniform vector, which is discard.

follows:

$$\mathbf{k}''(\mathbf{x}_i, \mathbf{x}_j) = \begin{cases} \mathbf{W}_{ij} + \mathbf{W}_{ji} - \sum_r \mathbf{W}_{ri}\mathbf{W}_{rj} & \text{if } \mathbf{x}_i, \mathbf{x}_j \in \mathbf{X} \\ 0 & \text{if } \mathbf{x}_i \notin \mathbf{X} \text{ or } \mathbf{x}_j \notin \mathbf{X} \end{cases} \tag{2.8}$$

And for $\mathbf{x}_i, \mathbf{x}_j \in \mathbf{X}$, the LLE kernel $\mathbf{k}_{lle}(\mathbf{x}_i, \mathbf{x}_j)$ is equal to the element in the kernel matrix $\mathbf{K} = \lambda_{max}\mathbf{I} - \tilde{\mathbf{K}} = \lambda_{max}\mathbf{I} - (\mathbf{I} - \mathbf{W})^T(\mathbf{I} - \mathbf{W})$ as follows:

$$\begin{aligned} \mathbf{k}_{lle}(\mathbf{x}_i, \mathbf{x}_j) &= (\lambda_{max} - 1)\mathbf{k}'(\mathbf{x}_i, \mathbf{x}_j) + \mathbf{k}''(\mathbf{x}_i, \mathbf{x}_j) \\ &= (\lambda_{max} - 1)\delta_{ij} + \mathbf{W}_{ij} + \mathbf{W}_{ji} - \sum_r \mathbf{W}_{ri}\mathbf{W}_{rj} \\ &= \lambda_{max}\delta_{ij} - (\delta_{ij} - \mathbf{W}_{ij} - \mathbf{W}_{ji} + \sum_r \mathbf{W}_{ri}\mathbf{W}_{rj}). \end{aligned}$$

On the other hand, the LLE kernel $\mathbf{k}_{lle}(\mathbf{x}, \mathbf{x}_i)$ for $\mathbf{x} \notin \mathbf{X}$ and $\mathbf{x}_i \in \mathbf{X}$ becomes as follows:

$$\begin{aligned} \mathbf{k}_{lle}(\mathbf{x}, \mathbf{x}_i) &= (\lambda_{max} - 1)\mathbf{k}'(\mathbf{x}, \mathbf{x}_i) + \mathbf{k}''(\mathbf{x}, \mathbf{x}_i) \\ &= (\lambda_{max} - 1)w_{llr}(\mathbf{x}, \mathbf{x}_i). \end{aligned}$$

Based on Nyström formula, the embedding point $\mathbf{z}$ for an out-of-sample $\mathbf{x} \notin \mathbf{X}$ is achieved as following:

$$\mathbf{z}(\mathbf{x}) = \frac{\lambda_{max} - 1}{\lambda_{max} - \lambda_k} \sum_i \mathbf{z}_{ik}\mathbf{k}_{lle}(\mathbf{x}, \mathbf{x}_i),$$

where λ_r is the r-th lowest eigenvalue of the matrix $\tilde{\mathbf{K}}$ and $\mathbf{z}_{ir}$ is the r-th coordinate of the embedding point $\mathbf{z}_i$ (Bengio et al., 2004)[2]. Moreover, if $\lambda_{max} \to \infty$, the embedding point becomes equal to the following approxi-

[2]For the proof of the out-of-sample extension, see (Bengio et al., 2003b).

mation proposed by Saul and Roweis (2003):

$$\mathbf{z}(\mathbf{x}) = \sum_{i=1}^{N} \mathbf{z}_i \mathbf{k}_{lle}(\mathbf{x}, \mathbf{x}_i) = \sum_{i=1}^{N} \mathbf{z}_i w_{llr}(\mathbf{x}, \mathbf{x}_i). \tag{2.9}$$

Finally, we know that the LLE kernel between a out-of-sample and its neighbor is equal to the coefficient of the neighbor in the locally linear reconstruction of the out-of-sample point in terms of the neighbor and other $k-1$ neighbors. It is a data-dependent kernel.

2.3 LLE Kernel and k-NN Prediction

The appealing property of LLE kernel, i.e., capturing the locally linear topology in terms of neighboring points seems to be well-suited to k-NN prediction. In this way, some research using LLE kernels as weights in the weighted and kernelized k-NN prediction (Kang and Cho, 2008; Ma et al., 2010) were performed. Kang and Cho (2008) demonstrated that using LLE kernels as weights in the k-NN prediction shows more accurate and k-invariant prediction than using other data-independent kernels without much additional computational cost in both regression and classification problems. Particularly, they called the weight kernel as LLR, in that the kernel for one neighbor in a prediction of a query point - out-of-sample point - is based on the coefficient assigned to the neighbor in the locally linear reconstruction of the query point. As explained in Section 2.2, the weight kernel which they called LLR, shortly LLR kernel is explained as a special case of the LLE kernel for an out-of-sample point. Moreover, in the work of Ma et al. (2010), a comprehensive comparison of the data-dependent kernels induced from some local manifold learning methods such as LLE, Local Tangent Space Alignment (LTSA) (Zhang and Zha, 2005) and Laplacian Eigenmaps (LE) (Belkin and Niyogi, 2003) was conducted in the hyper-spectral image classification. In their exper-

iments, LLE kernel was also shown to have relatively more accurate and k-invariant predictions.

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

$$y(\mathbf{x}) = \sum_{i=1}^{k} w_i y_{nn(\mathbf{x},i)} = \mathbf{w}^T \mathbf{y}_{nn(\mathbf{x})}, \tag{2.10}$$

where $nn(\mathbf{x}, i)$ is the index of the i^{th} nearest neighbor of a query point $\mathbf{x}$, and $nn(\mathbf{x})$ is the index set of neighbors among the training points $\mathbf{X} = \{\mathbf{x}_n \in \mathcal{X} = \mathbb{R}^D\}_{n=1}^N$. This weighted k-NN learning (Dudani, 1976; Kozak et al., 2006) has two user-specific parameters to be determined: first, the number of nearest neighbors, k and second, the weights assigned to the selected neighbors, $\mathbf{w} = [w_1, ..., w_k]^T$. 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 (Kang and Cho, 2008).

While conventionally used distance-based weight kernels in k-NN learning, e.g., data-independent kernels such as inversion kernel (Ruprecht and Müller, 1994; Wolberg, 1990), exponential kernel (Aha and Goldstone, 1992), and Gaussian kernel (Wand and Schucany, 1990) under-utilize available information in k-nearest neighbors (Kang and Cho, 2008; Ma et al., 2010), LLR kernel takes into account the locally linear topology around a query point, assuming that the query point can be linearly reconstructed by its k nearest neighbors in the input space. That is, the LLR weights for k neighbors in k-NN learning, i.e., $w_i := \mathbf{k}_{lle}(\mathbf{x}, \mathbf{x}_{nn(\mathbf{x},i)}) = w_{llr}(\mathbf{x}, \mathbf{x}_i)$ are collectively determined, based on their proportional contributions to the reconstruction of the query point. Moreover, compared with the distance-

based kernels, LLR kernel tends to identify the best subset of relevant neighbors for sufficiently large k values.

And we note that the coefficients or weights assigned for k neighbors are obtained by LLR operation, thus, constrained to sum to one, but may be either positive or negative unlike other weight kernels in the k-NN learning. Regarding the role of two kinds of neighbors, Kang and Cho (2008) 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,

$$y(\mathbf{x}) = \bar{y}_{nn(\mathbf{x})} + \sum_{i=1}^{k} w_i [y_{nn(\mathbf{x},i)} - \bar{y}_{nn(\mathbf{x})}]. \tag{2.11}$$

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 (Bottou and Vapnik, 1992). In contrast, the second term is interpreted as a correction term introduced by weight kernels. In the case of LLR kernel, positive-weighted neighbors in the locally linear topology pull the predicted output toward their target values, while negative-weighted neighbors push away it from their target values. In addition, the coefficients may be forced to be non-negative, as in the LLR-based k-NN classification (Kang and Cho, 2008). It then solved the convex reconstruction (Saul and Roweis, 2003) as the quadratic programming (QP) formulation (Kang and Cho, 2008). Here, we consider only the conventional LLR kernel without the non-negativity constraint, in that negative weights may be helpful for a data point lying on the boundary of a manifold and outside the convex hull of their neighbors (Saul and Roweis, 2003).

And we point out that a query and its k neighbors in LLR kernel are geometrically assumed to lie on or close to a locally linear patch of the manifold in the input space (Roweis and Saul, 2000). 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 in some regressions. The limitation leads to a Bayesian kernel model of the locally linear reconstruction, introduced in Chapter 3.

Chapter 3

Probabilistic Local Reconstruction for k-NN Regression

In this chapter, a Bayesian kernel treatment of LLR kernel, named Probabilistic Local Reconstruction (PLR) is introduced. PLR kernel provides an probabilistic extension of LLR kernel in Section 3.1. Briefly, computing PLR kernel for k-NN prediction is interpreted as adjusting the reconstructed point at the average point of k neighbors - expected by the prior assumption - to be close to the corresponding query point. The degree of the adjustment is controlled by the additive noise assumption of PLR. In this way, k-NN prediction using PLR kernel can be the local constant-fit without suffering from the small k problem with a handful of neighbors. In addition, the kernelized implementation of PLR itself is introduced in Section 3.2. It helps the model capture the non-linear topology embedded in too redundant neighbors with a large k setting, thus, be robust to a wider range of k values. In preliminary experimental results in benchmark regression problems in Section 3.3, we demonstrate that the Bayesian kernel treatment of LLR kernel improves the k-invariant property as well as prediction accuracy in the kerenlized k-NN regression. Finally, we provide the summary and discussions in Section 3.4.

3.1 Probabilistic Extension of LLR

As the first step, the locally linear reconstruction is treated in a probabilistic view in order to handle the small k problem when using LLR kernel in the kernelized k-NN regression. 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, implying that the set of k neighbors are irrelevant to k-NN prediction. These uncertain and inaccurate reconstructions for such query points give rise to performance deterioration in k-NN prediction. Such problem becomes serious when the locality parameter k is set to be too small.

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

$$\mathbf{x} = \mathbf{X}_{nn(\mathbf{x})}\mathbf{w}_{plr} + \epsilon_{recon} = \mathbf{x}_{recon} + \epsilon_{recon}, \quad \epsilon_{recon} \sim N(\mathbf{0}, \beta_{recon}^{-1}\mathbf{I} = \sigma_{recon}^2\mathbf{I}). \tag{3.1}$$

The additive noise assumption is introduced for explaining the difference between a query $\mathbf{x}$ and its reconstructed point $\mathbf{x}_{recon}$. 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:

$$p(\mathbf{x}|\mathbf{X}_{nn(\mathbf{x})}, \mathbf{w}_{plr}) = N(\mathbf{x}|\mathbf{X}_{nn(\mathbf{x})}\mathbf{w}_{plr}, \beta_{recon}^{-1}\mathbf{I}). \tag{3.2}$$

It is the probability density of the query point, given its corresponding k nearest neighbors. The explicit solution obtained by LLR - minimizing the reconstruction error of a quadratic form - is equivalent to the weight parameters that maximize the likelihood function. Rather than simply maximizing the likelihood, we incorporate Gaussian prior $p(\mathbf{w}_{plr}) = N(\mathbf{w}_{plr}|k^{-1}\mathbf{1}_{k\times 1}, \mathbf{I})$ into the model, considering that the sum of

weights is constrained to one, i.e., $\sum_{i=1}^{k} w_{plr}(\mathbf{x}, \mathbf{x}_{nn(\mathbf{x},i)}) = \mathbf{w}_{plr}^T \mathbf{1}_{k\times 1} = 1$ 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,

$$p(\mathbf{w}_{plr}|\mathbf{X}_{nn(\mathbf{x})}, \mathbf{x}) \propto p(\mathbf{x}_{plr}|\mathbf{X}_{nn(\mathbf{x})}, \mathbf{w}_{plr}) p(\mathbf{w}_{plr}) = N(\mathbf{w}_{plr}|\mu, \Sigma). \quad (3.3)$$

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

$$\mu = \Sigma(k^{-1}\mathbf{1}_{k\times 1}), \quad \Sigma = (\beta_{recon}\tilde{\mathbf{C}} + \mathbf{I})^{-1}, \quad (3.4)$$

where $\tilde{\mathbf{C}} = \tilde{\mathbf{X}}_{nn(\mathbf{x})}^T\tilde{\mathbf{X}}_{nn(\mathbf{x})} = (\mathbf{X}_{nn(\mathbf{x})} - \mathbf{x}\mathbf{1}_{k\times 1}^T)^T(\mathbf{X}_{nn(\mathbf{x})} - \mathbf{x}\mathbf{1}_{k\times 1}^T)$. The posterior covariance matrix Σ is similar to the inversion of the regularized local gram matrix in Eq. (2.4), i.e., $(\tilde{\mathbf{C}} + \Delta\mathbf{I})^{-1}$ where $0 < \Delta \ll 1$ in the non-probabilistic LLR. Moreover, when the sum-to-one constraint of LLR is applied, PLR solution is adjusted with γ so that $\gamma\sum_i \mu_i = \gamma\mu^T\mathbf{1}_{k\times 1} = 1$. It is straightforward.

And it is noted that introducing the Gaussian prior allows the probabilistic LLR to explain the constant-fit, which leads to the equally-weighted k-NN prediction. In the probabilistic view, LLR is considered as the learning process to adjust the reconstructed projection, i.e., reconstructed point $\mathbf{x}_{recon}$ 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_{recon} approaches zero, the difference between a query $\mathbf{x}$ and its reconstructed point $\mathbf{x}_{recon}$ 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., $1/k$.

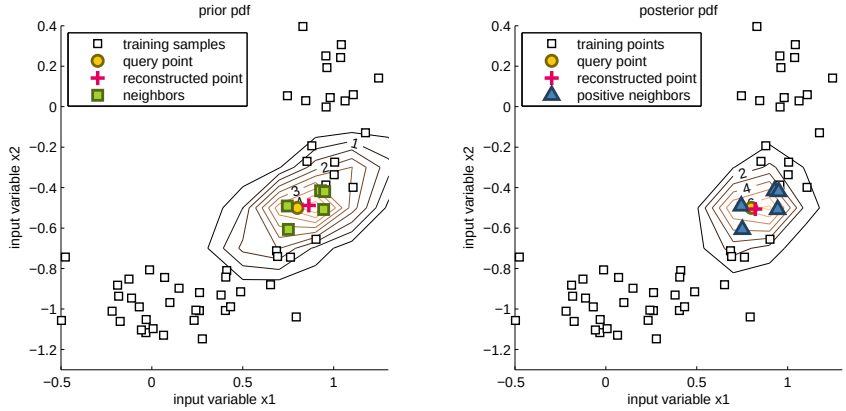
$$\lim_{\beta_{recon}\to 0}\mu = \lim_{\beta_{recon}\to 0}(\beta_{recon}\tilde{\mathbf{C}} + \mathbf{I})^{-1}(k^{-1}\mathbf{1}_{k\times 1}) = k^{-1}\mathbf{1}_{k\times 1}. \tag{3.5}$$

In order to provide the geometrical interpretation, the following posterior distribution for a reconstructed point $\mathbf{x}_{recon}$ is derived, based on the reconstruction equation shown in Eq. (3.1):

$$p(\mathbf{x}_{recon}|\mathbf{X}_{nn(\mathbf{x})}, \mathbf{x}) = N(\mathbf{x}_{recon}|\tilde{\mathbf{X}}^T_{nn(\mathbf{x})}\mu + \mathbf{x}, \tilde{\mathbf{X}}_{nn(\mathbf{x})}\Sigma\tilde{\mathbf{X}}^T_{nn(\mathbf{x})}) \tag{3.6}$$

When the weight posterior is replaced with the prior, the mean vector of the prior distribution for the reconstructed point becomes the average point of k neighbors. That is, the posterior weight obtained by the Bayes' rule in Eq. (3.4) makes the reconstructed point move from the average point of k neighbors to the query point.

For example, Fig. 3.1 and 3.2 show such changes from priors (left plots in each figure) to posteriors (right plots in each figure) for reconstructed points of two different queries denoted as filled circles. The query point in Fig. 3.1 is located inside the manifold, thus, likely to have neighbors (denoted as filled squares in Fig. 3.1(a)) informative to the locally linear reconstruction. In this case, the probable region in which the reconstructed point for the query is possibly located, becomes narrow, as shown in its posterior probability density function in Fig. 3.1(b). On the other hand, the query point in Fig. 3.2 is located outside the manifold and has relatively less informative neighbors than the former case. Nevertheless, it is noted that the reconstructed point (denoted as a cross) approaches to the corresponding query through PLR, i.e. fitting from the prior to the posterior, as shown in Fig. 3.2(a) and (b). The reconstruction error decreases. However, the region where the reconstructed point is probable rather becomes wide, as shown in Fig. 3.2(b). In this way, the posterior probability

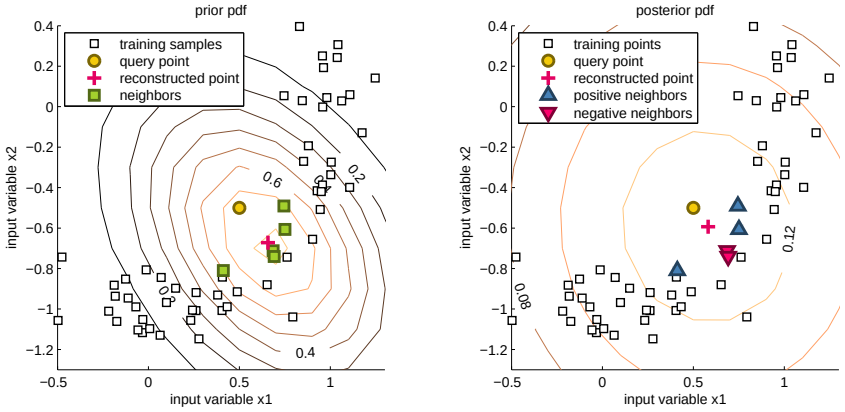


(a) (b)

Figure 3.1: The (a) prior and (b) posterior probability density for the reconstructed point of a query inside the manifold, obtained by the probabilistic view of LLR with k=5

density function for the reconstructed point shows that the query point is not surrounded by informative neighbors, which leads to error-minimized, but uncertain reconstruction of the query.

In addition, there are negative-weighted neighbors in the case of Fig. 3.2(b), while no negative-weighted ones in Fig. 3.1(b). The example reconfirms some intuitions pointed out by other research, implying the probabilistic formulation of LLR intuitively well explains the learning mechanism of LLR. First, the sign of weights does not necessarily depend on their pair-wise distances to the query, as shown in Fig. 3.2(b). It is rather determined by how far neighbors are located from hyper-plane minimizing reconstruction error. Second, as Saul and Roweis (2003) pointed out, Fig. 3.2 shows that negative weights are helpful for data points lying on the boundary of a manifold and outside the convex hull of their neighbors. It is explained by the interpretation for the role of two kinds of neighbors (Kang and Cho, 2008): the negative-weighted neighbors push away the hyper-plane to the query. In addition, that is the reason why there are no



(a) (b)

Figure 3.2: The (a) prior and (b) posterior probability density for the reconstructed point of a query outside the manifold, obtained by the probabilistic view of LLR with k=5

negative-weighted neighbors in the case of Fig. 3.1 - the query is located inside the manifold.

3.2 Kernelized Implementation of PLR

As described earlier, LLE kernel 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 (Roweis and Saul, 2000). Moreover, setting k too large violates the fundamental assumption of LLR (Saul and Roweis, 2003), 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, the lower-dimensional input space is transformed into a higher-dimensional feature space, i.e., $\mathbf{x} \mapsto \phi(\mathbf{x})$ with the kernel trick $\mathbf{k}(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j)$ computing the dot product in the feature space with no explicit mapping function $\phi(\cdot)$. Applying the kernel function to the posterior mean and

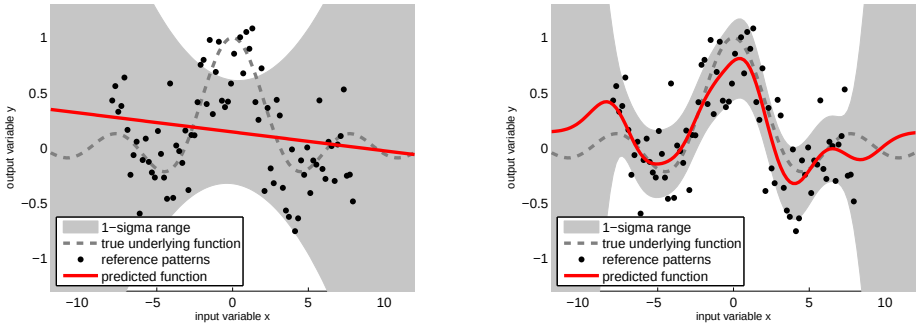
covariance in Eq. (3.4), the following PLR solution is obtained:

$$\mu = \Sigma(k^{-1}\mathbf{1}_{k\times 1}), \quad \Sigma = (\beta_{recon}\tilde{\mathbf{K}} + \mathbf{I})^{-1}. \tag{3.7}$$

In the equation, the centered kernel matrix $\tilde{\mathbf{K}}$ is the same as the matrix of kernel Principal Component Analysis (KPCA) (Schölkopf et al., 1998), and defined as $\tilde{\mathbf{K}} = \mathbf{K} - \mathbf{1}_{k\times 1}\mathbf{k}(\mathbf{x}, \mathbf{X}_{nn(\mathbf{x})})^T - \mathbf{k}(\mathbf{x}, \mathbf{X}_{nn(\mathbf{x})})\mathbf{1}_{k\times 1}^T + \mathbf{k}(\mathbf{x}, \mathbf{x})\mathbf{1}_{k\times 1}\mathbf{1}_{k\times 1}{}^T$. The matrix $\mathbf{K}$ is the $k \times k$ kernel matrix where each element is defined as $\mathbf{K}_{ij} := \mathbf{k}(\mathbf{x}_{nn(\mathbf{x},i)}, \mathbf{x}_{nn(\mathbf{x},j)})$. $\mathbf{k}(\mathbf{x}, \mathbf{X}_{nn(\mathbf{x})})$ is the column vector consisting of the kernel functions of k neighbors for a query point, i.e., $\mathbf{k}(\mathbf{x}, \mathbf{X}_{nn(\mathbf{x})}) = [\mathbf{k}(\mathbf{x}, \mathbf{x}_{nn(\mathbf{x},1)}), ..., \mathbf{k}(\mathbf{x}, \mathbf{x}_{nn(\mathbf{x},k)})]^T$.

The kernelized implementation of PLR estimates data-dependent kernels in terms of data-independent kernels as base ones. If the kernel function used as a base kernel in PLR is linear, i.e., $\mathbf{k}(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^T\mathbf{x}_j$, PLR has the same model complexity as LLR. On the other hand, if data-independent kernels such as radial basis function (RBF) kernel $\mathbf{k}(\mathbf{x}_i, \mathbf{x}_j) = exp(-\frac{||\mathbf{x}_i - \mathbf{x}_j||^2}{2l^2})$ are used as a base kernel, the PLR can capture non-linear topology embedded in too redundant neighboring points with large k settings. In this way, the kernelized implementation of PLR makes the local reconstruction more flexible for a wider range of k values. It improves the accuracy of the corresponding kernelized k-NN regression with a large k setting.

For example, Fig. 3.3 shows the comparison of the two kernelized k-NN regressions with k=200 and the number of training points, N=200. The weight kernel used in these two k-NN regressions is the PLR kernel, i.e., $w_i := w_{plr}(\mathbf{x}, \mathbf{x}_{nn(\mathbf{x},i)})$ and $\mathbf{w} := \mathbf{w}_{plr} \sim N(\mu, \Sigma)$. The predictive variances of the k-NN regressions using PLR kernel will be discussed with a real-world application in Chapter 4. The point in this sub-section is that the difference of two k-NN regressions, which is the base kernel used in PLR, i.e., linear kernel in Fig. 3.3(a) and RBF kernel in Fig. 3.3(b). In



(a) (b)

Figure 3.3: illustration of the two PLR k-NN regressions with $\beta_{recon} = 1$, (a) linear kernel and (b) RBF kernel ($l = 2$)

the case of PLR with the linear kernel, PLR is simplified as representing as a query point x as a weighted combination of neighbors' input values, i.e., $x = \sum_{i=1}^{k} w_{plr}(\mathbf{x}, \mathbf{x}_{nn(\mathbf{x},i)}) x_{nn(x,i)}$. All neighbors' contributions in the reconstruction, i.e., PLR weights become linearly proportional to their distances or similarities with the query value. In this way, the kernelized k-NN regression using the linear PLR in a large k setting, i.e., $y = \sum_{i=1}^{k} w_{plr}(\mathbf{x}, \mathbf{x}_{nn(\mathbf{x},i)}) y_{nn(x,i)}$ becomes a linear fit, as shown in Fig. 3.3. On the other hand, the k-NN weights estimated by non-linear PLR can be biased to more relevant neighbors, i.e., not linear, as shown in Fig. 3.3(b). In this way, since PLR captures such non-linear or biased topology according to its base kernel definition, the kernelized implementation of PLR avoids performance deterioration in large k settings.

3.3 Experimental Result on Benchmark Regression Problems

Experiments on 15 regression problems were conducted in order to demonstrate the k-invariant property of PLR. The data sets used in our ex-

Table 3.1: The description of benchmark regression problems

Data set	Train	Test	Attribute	Origin	Time-series?
`Stock`	500	450	9	Luís Torgo	No
`Abalone`	2,000	2,000	8	Luís Torgo	No
`Wind`	3,000	3,500	11	StatLib	No
`Computer`	4,000	4,000	22	Delve	No
`Kinematics`	4,000	4,000	8	Luís Torgo	No
`Bank`	4,000	4,000	8	Luís Torgo	No
`Pumadyn`	4,000	4,000	8	Luís Torgo	No
`Add10`	4,000	5,000	5	Delve	No
`California`	5,000	5,000	8	Luís Torgo	No
`Census`	5,000	5,000	9	Luís Torgo	No
`Gold`	500	500	10	TSDL	Yes
`Sunspot`	1,000	1,500	10	TSDL	Yes
`Melbmax`	1,000	2,500	10	TSDL	Yes
`Santa Fe A`	5,000	5,000	10	TSDL	Yes
`Santa Fe D`	5,000	5,000	10	TSDL	Yes

periments were selected from the regression data sets provided by Luís Torgo[1], Delve datasets[2], Time Series Data Library (TSDL)[3], and Statlib[4]. The descriptions of each data set are summarized in Table 3.1. The time series data sets are re-formulated 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 (Kim and Cho, 2008). In addition, a predefined number of data points for the training and testing are randomly selected ten times. The 'train' column in Table 3.1 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.

[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 3.2: The parameter settings for each regression model

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
	$w_i := \exp(-\frac{1}{2l^2}\|\|\mathbf{x} - \mathbf{x}_{nn(\mathbf{x},i)}\|\|^2)$, $l = 1$
LLR	k=3,5,10,50,100,200,300,400,500
	$\mathbf{w} := \mathbf{w}_{llr} = \sum_i \tilde{\mathbf{C}}^{-1}_{i,j} / \sum_{l,m} \tilde{\mathbf{C}}^{-1}_{l,m}$
	$\Delta = 10^{-3}$
PLR-linear	k=3,5,10,50,100,200,300,400,500
	$\mathbf{w} := N(\mathbf{w}_{plr}\|\mu = \Sigma(k^{-1}\mathbf{1}_{k\times 1}), \Sigma = (\beta_{recon}\tilde{\mathbf{K}} + \mathbf{I})^{-1})$
	$\mathbf{k}(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^T\mathbf{x}_j$, linear kernel
	β_{recon}=10,100,1000
PLR-RBF	k=3,5,10,50,100,200,300,400,500
	$\mathbf{w} := N(\mathbf{w}_{plr}\|\mu = \Sigma(k^{-1}\mathbf{1}_{k\times 1}), \Sigma = (\beta_{recon}\tilde{\mathbf{K}} + \mathbf{I})^{-1})$
	$\mathbf{k}(\mathbf{x}_i, \mathbf{x}_j) = exp(-\frac{\|\|\mathbf{x}_i - \mathbf{x}_j\|\|^2}{2l^2})$, $l = 1$
	β_{recon}=10,100,1000
LinReg	-

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. The k-NN models were tested in nine k values ranging from 3 to 500 in order to verify how the models are robust to any non-optimized k values, i.e., k-invariant. The models and corresponding parameter settings are summarized in Table 3.2. First, unweighted or equally-weighted k-NN regression was used as a baseline and labeled as 'unweighted' in Table 3.2. 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 data-independent ones for the kernelized k-NN regression. Third, LLR defined in Eq. (2.3) was tested with the constant regularization term $\Delta = 10^{-3}$. Fourth, linear PLR - PLR using the linear kernel $\mathbf{k}(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^T\mathbf{x}_j$ as a base kernel - was adopted as a weight kernel of k-NN regression and labeled as 'PLR-linear'. Moreover,

the parameter β_{recon} 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. Comparison between 'PLR-linear' and LLR shows how significant the tuned noise assumption for locally linear reconstruction is in given small k settings. Finally, non-linear PLR using the RBF kernel $\mathbf{k}(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\frac{1}{2l^2}||\mathbf{x}_i - \mathbf{x}_j||^2)$ with a fixed $l = 1$ as a base kernel was tested and labeled as 'PLR-RBF' in Table 3.2. As mentioned earlier, it is expected to capture the non-linear neighborhood structure formed by large k values. This preliminary experiment shows how useful the weight kernel based on such flexible reconstruction is in the kernelized k-NN regressions with large k values.

Figs. 3.4, 3.5 and 3.6 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. For the sake of interpretation, the caps on y-axis were set for some plots and the actual scale of k values was ignored. Moreover, Tables 3.3 and 3.4 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.

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 may be the inherent limitation of the k-NN regression and be more critical than which kernels are suitable in k-NN regression. 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 will be demonstrated with a real-world application in Chapter 4.

Second, The RMSEs of PLR-linear and LLR by k values were 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 estimating the weights for k neighbors in k-NN learning. Moreover, in the data sets, these two models were 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 performance drop and its regularization term should be carefully turned. In contrast, PLR-linear whose parameter β_{recon} is optimized, showed 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. That is, k-NN regression using PLR-RBF as a weight kernel can avoid performance deterioration due to large k values in some regressions. In addition, although PLR-RBF does not directly capture the non-linear relationships between the inputs and target variable, the effect may become salient in the non-linear regression problems, illustrated by the example in Fig. 3.3.

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 prob-

lems. This experimental result leads to the conclusion that PLR is more k-invariant than other weight kernels in k-NN regression. 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.

And 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. 3.7, 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. Moreover, the times by the number of training points N for optimizing PLR parameters in training phase are shown in Fig. 3.8. Since the parameter optimization was necessary but not performed in LLR, the train time of LLR is omitted in the figure. As shown in the figure, the train time of PLR mainly depends on N rather than k. It implies that the time for the parameter optimization can be reduced by some efficient neighbor search algorithms such as k-d tree.

3.4 Summary and Discussion

In this chapter, a Bayesian kernel treatment of LLR, named Probabilistic Local Reconstruction (PLR) is presented. The probabilistic extension of LLR provides the probabilistic interpretation of the locally linear reconstruction. It allows k neighbors the equal contribution in the reconstruction, resulting in the equally-weighted k-NN prediction without suffering from the small k problem with few informative neighboring points. In addition, its kernelized implementation rendered PLR kernel more flexible than

[5]Since the fastest matrix inversion requires $O(n^{2.376})$ (Coppersmith and Winograd, 1990), we wrote the computational cost $O(n^{2.376})$ instead of $O(n^3)$ in this dissertation.

LLR, which is suitable to k-NN regression with large k. Because the kerenlized PLR can capture the non-linear or locally-biased topology embedded in too redundant neighboring points. The preliminary experimental result demonstrated that using PLR as a weight kernel in kernelized k-NN regression achieves higher prediction accuracy as well as lower variation for the locality parameter k, i.e., k-invariant.

This chapter is finished with an interesting issue: PLR for outlier detection problem. The reconstruction probability density for a reconstructed point at a query in PLR is expected to be an outlier detection measure for lack of informative training points in the locally linear reconstruction view. Since the k-NN based outlier detection is out of the scope of this chapter, its possibility of the outlier detection itself is briefly pointed out. Fig. 3.9 shows the reconstruction probability densities at arbitrary queries in the two-dimensional input space. As shown in the figure, the probability densities form a data domain description (Tax and Duin, 2004), i.e., a boundary covering the training points. In this way, the probability density of a query in PLR view may be a k-NN concentration measure of the One-Class Neighbor Machine (OCNM, see Munoz and Moguerza (2006)). 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 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 (Ahmed et al., 2007).

And the reconstruction error similar to the reconstruction probability density, may be other candidate for outlier detection. The reconstruction error of PLR is a combination between the average point of k neighbors and the conventional reconstruction error of LLR. It is related to the hybrid novelty score proposed by Kang and Cho (2009) in the keystroke dynamics-based user authentication problem. The hybrid novelty score combines two distances - the average distance to the k neighbors and reconstruction error

of LLR.

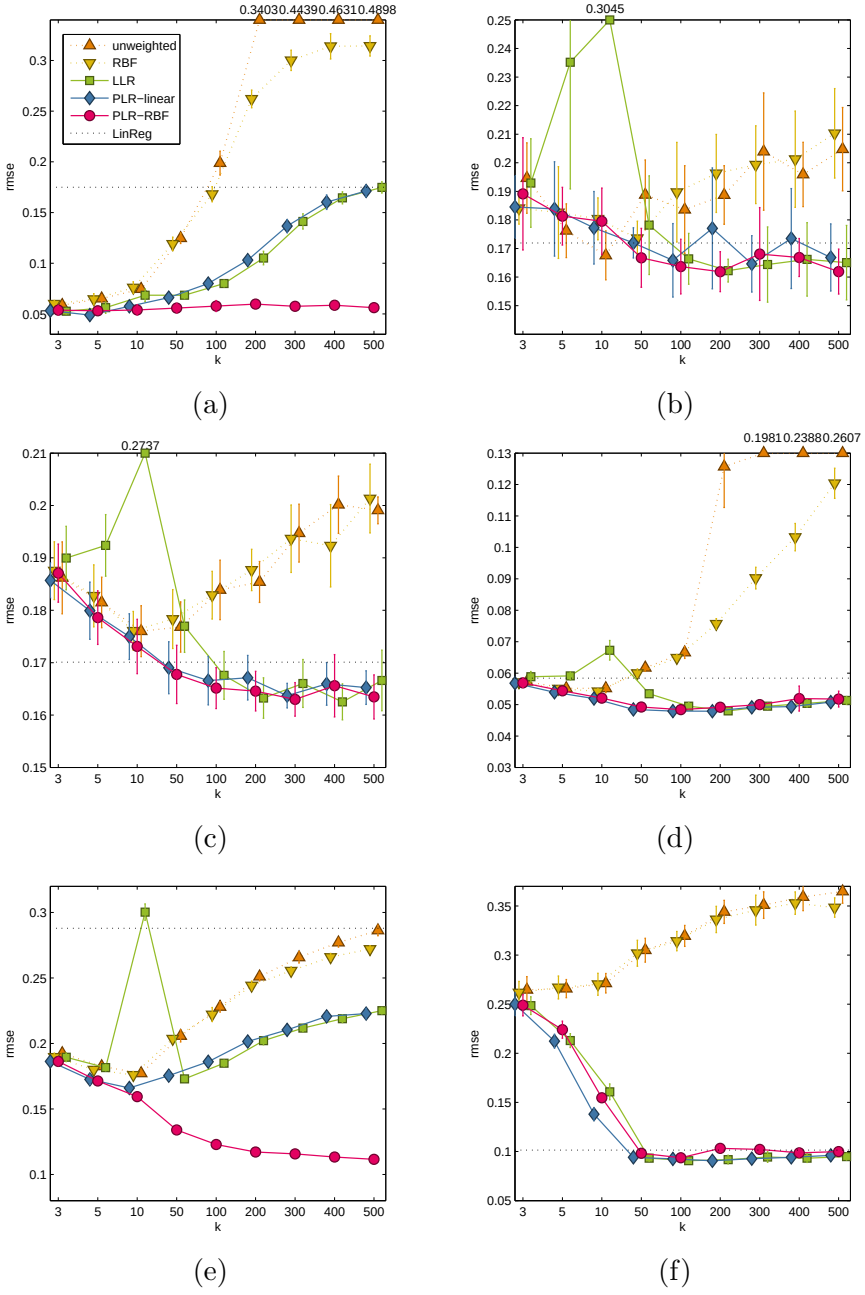


Figure 3.4: RMSE means and one-sigma ranges of all the models by k values for each data set (x-axis denotes k while y-axis denotes the RMSE). (a) `Stock` (b) `Abalone` (c) `Wind` (d) `Computer` (e) `Kinematics` (f) `Bank`.

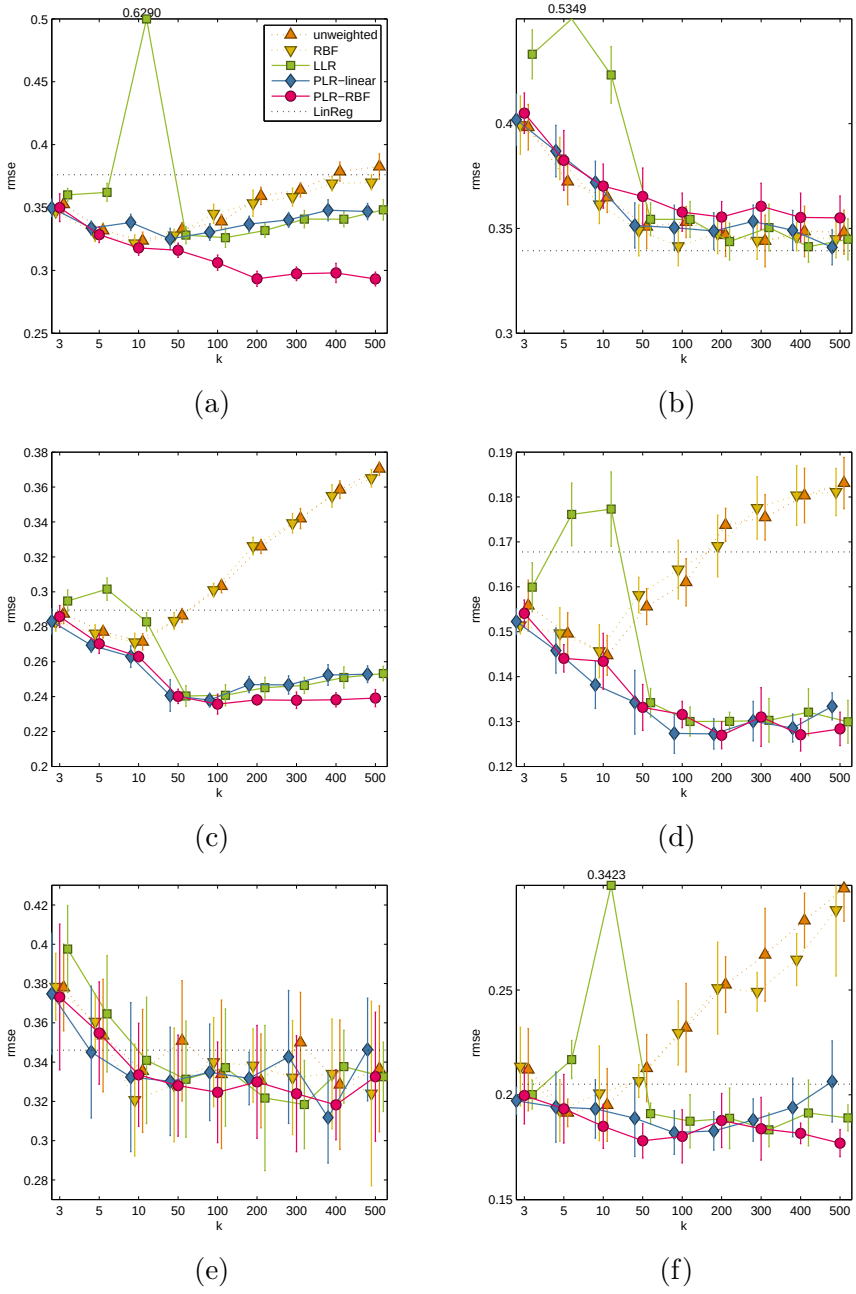


Figure 3.5: RMSE means and one-sigma ranges of all the models by k values for each data set (x-axis denotes k while y-axis denotes the RMSE). (a) `Pumadyn` (b) `Add10` (c) `California` (d) `Census` (e) `Gold` (f) `Sunspot`.

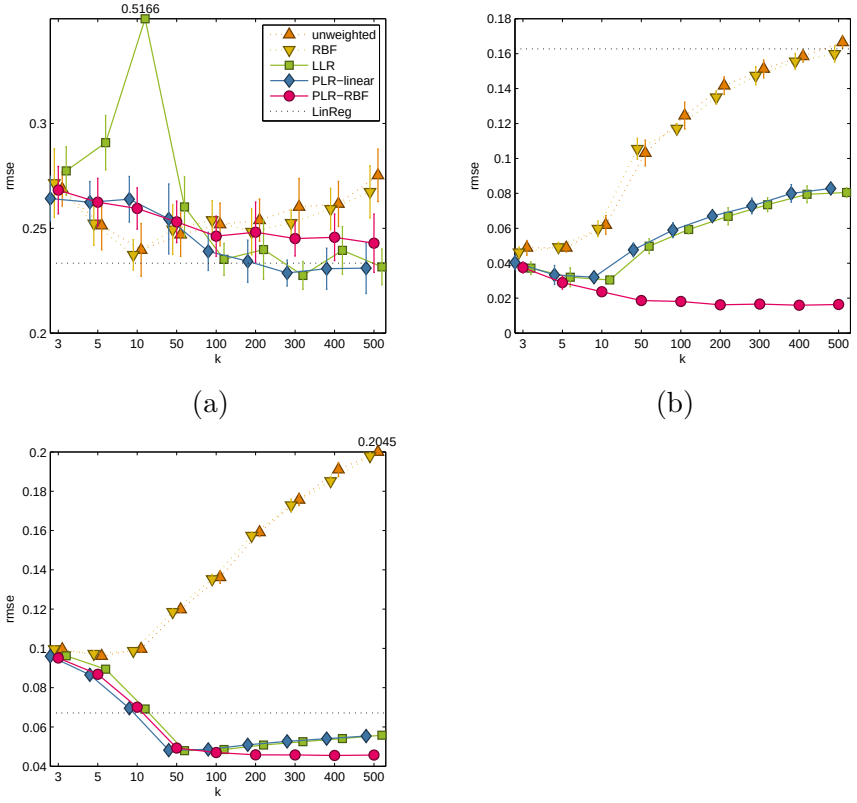


Figure 3.6: RMSE means and one-sigma ranges of all the models by k values for each data set (x-axis denotes k while y-axis denotes the RMSE). (a) `Melbmax` (b) `Santa Fe A` (c) `Santa Fe D`.

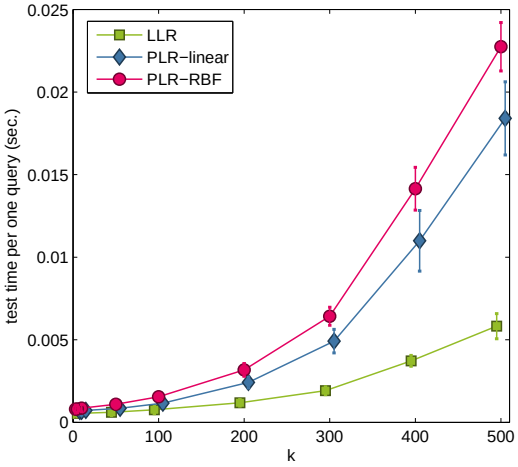


Figure 3.7: Prediction time per one query point of three k-NN models (x-axis denotes k values while y-axis denotes the prediction time per one query in second.)

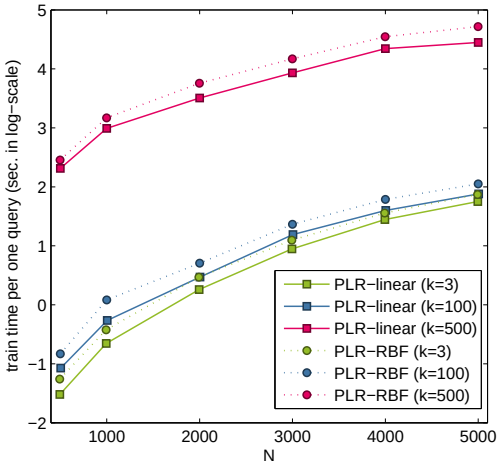


Figure 3.8: Train time of PLR-linear and PLR-RBF (x-axis denote N values while y-axis denotes the train time in the log-scale of second.)

Table 3.3: RMSEs of all the k-NN models with four intervals (low, middle, high, and overall) of k values for each data set. The bold faced number represent the best k-NN RMSE for the corresponding data sets. The the double plus ++ 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
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

Table 3.4: RMSEs of all the k-NN models with four intervals (low, middle, high, and overall) of k values for each data set. The bold faced number represent the best k-NN RMSE for the corresponding data sets. The the double plus ++ 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
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
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

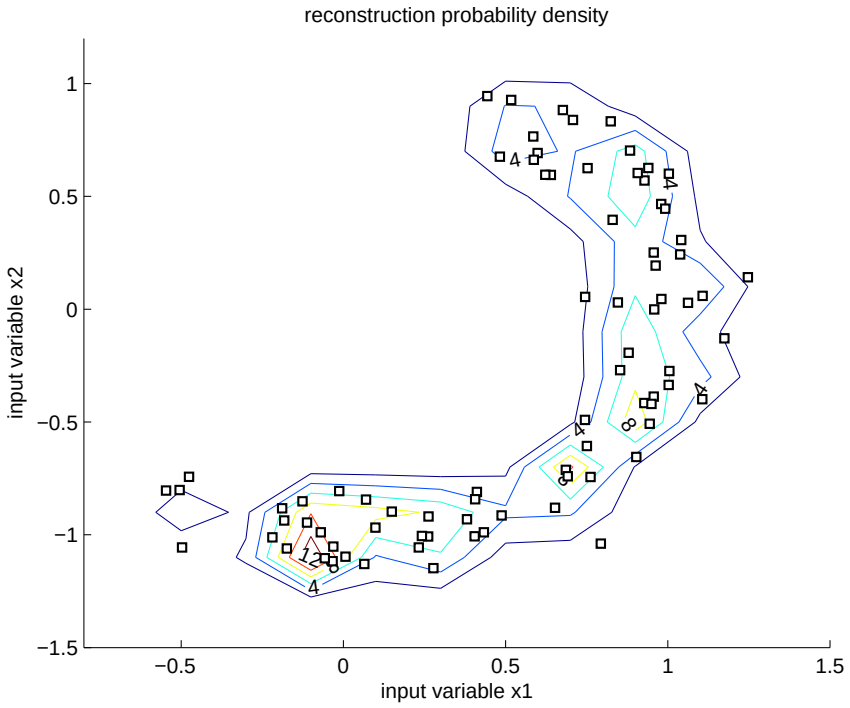


Figure 3.9: Reconstruction probability density contour of queries in two-dimensional input space, obtained by PLR with k=5

Chapter 4

Estimation of Prediction Reliance for k-NN Regression

The probabilistic extension of LLR provides a probabilistic estimate quantifying reconstruction uncertainty, implying "how informative is the set of k neighbors in the locally linear reconstruction?" Reconstructions in extremely sparse or unknown regions in input space are likely to be uncertain. Moreover, one of the advantages of the probabilistic estimate is that it can be incorporated with target uncertainty, i.e., the predictive variance of the k-NN regressor. In this way, the predictive variance incorporated with reconstruction uncertainty and its useful property with a toy example are explained in Section 4.1. It is a reasonable prediction reliance estimator of k-NN regression. Further, Virtual Metrology (VM) problem in the semiconductor industry, one of applications where it is critical to provide the reliance level of predictions, is introduced in Section 4.2. Experimental result on a real-world VM problem in Section 4.3 shows that the uncertainty information on the prediction outcomes provided by PLR supports more appropriate decision making. Finally, Section 4.4 concludes the chapter with some discussions.

4.1 PLR-based Prediction Reliance for k-NN Regression

In general, classifiers or regressor only provide reliable estimates for points in regions where training points are densely sampled. On the other hand, extrapolations to extremely sparse or unknown regions in input space become uncertain (Roberts et al., 1996; Tax and Duin, 2004). Moreover, there may be the target uncertainty, i.e., ambiguity by inconsistent or severely dispersed target values of neighbors (Dubuisson and Masson, 1993; Landgrebe et al., 2006) - violating the smoothness assumption. In the test or deployment stage, The introduction of a reject option for the two uncertain cases, i.e., rejecting the predictions of data points which would otherwise be unreliably predicted, should be considered (De Stefano et al., 2000; Landgrebe et al., 2006).

PLR kernel is a probabilistic estimate quantifying the reconstruction uncertainty, which implies lack of informative neighboring points for the reconstruction of a query point, as shown in Fig. 3.9. Moreover, the probabilistic estimate 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 (Dubuisson and Masson, 1993; Landgrebe et al., 2006) by inconsistent or severely dispersed target values of k neighbors in the case of k-NN regression. Applying PLR to kernelized k-NN regression defined in Eq. (2.11), i.e., $\mathbf{w} := \mathbf{w}_{plr} \sim N(\mu, \Sigma)$, the predictive distribution for the target value y of a query point $\mathbf{x}$ is obtained:

$$y = \bar{y}_{nn(\mathbf{x})} + \mu^T(\mathbf{y}_{nn(\mathbf{x})} - \bar{y}_{nn(\mathbf{x})}\mathbf{1}_{k\times 1}) \tag{4.1}$$

$$\sigma^2 = \frac{1}{k}(\mathbf{y}_{nn(\mathbf{x})} - \bar{y}_{nn(\mathbf{x})}\mathbf{1}_{k\times 1})^T\Sigma(\mathbf{y}_{nn(\mathbf{x})} - \bar{y}_{nn(\mathbf{x})}\mathbf{1}_{k\times 1}), \tag{4.2}$$

where $\mathbf{y}_{nn(\mathbf{x})} = [y_{nn(\mathbf{x},1)}, ..., y_{nn(\mathbf{x},k)}]^T$. k^{-1} in the predictive variance is

added for consistency with the variance of the k neighbors' target values. When the predictive variance in the equally-weighted k-NN regression is defined as follows,

$$\sigma^2 = \frac{1}{k}(\mathbf{y}_{nn(\mathbf{x})} - \bar{y}_{nn(\mathbf{x})}\mathbf{1}_{k\times 1})^T(\mathbf{y}_{nn(\mathbf{x})} - \bar{y}_{nn(\mathbf{x})}\mathbf{1}_{k\times 1}), \tag{4.3}$$

the predictive variance of PLR defined in Eq. (4.2) is explained as the variance of neighbors' target values considering the local topology of neighboring points as a form of the covariance matrix of PLR, $\Sigma = (\beta_{recon}\tilde{\mathbf{C}}+\mathbf{I})^{-1}$.

The difference between the predictive variances of the equally-weighted and PLR k-NN regressions is shown in Fig. 4.1. 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., $-8 \leq x \leq 8$ 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 (Rasmussen and Williams, 2006), because extrapolations to extremely sparse or unknown regions in the input space become uncertain (Roberts et al., 1996; Tax and Duin, 2004). As shown in Fig. 4.1(a), the predictive variance of the equally-weighted k-NN regression does not capture such extrapolation regions, since it depends on only neighbors' target values. On the other hand, the predictive variances of PLR k-NN regression in sparse regions with no informative training points increases by the reconstruction noise, $\beta_{recon} := \sigma_{recon}^{-2}$. As shown in Fig. 4.1(b), when the β_{recon} is set to nearly zero, its corresponding predictive variances are similar to that of the equally-weighted k-NN regression. It is consistent with the theoretical explanation of PLR in Chapter 3. Moreover, as β_{recon} increases, PLR becomes more sensitive to the region where there is no informative neighboring points. Thus, the predictive variances of PLR k-NN regression in sparse regions with no informative training points are corrected by the covariance term induced by PLR and dramatically

increase as shown in Fig. 4.1(c) and (d).

In this way, one of the important implications of PLR kernel is that it can be utilized as a reliance level of the k-NN prediction, taking account of extrapolation cases in the input space. It is noted that one of the well-principled regression models providing predictions' reliance levels, Gaussian process (GP) regression (Rasmussen and Williams, 2006) may also capture such extrapolated predictions. However, GP regression fundamentally expresses the target uncertainty, i.e., the uncertainty over target values in the probabilistic view. In other words, GP regression assumes an unknown underlying function $f(\mathbf{x})$ which generates noiseless data points and aims to recover the underlying function from noisy samples. GP regression is defined as $y(\mathbf{x}) = f(\mathbf{x}) + \epsilon_{target}$ where the additive Gaussian noise $\epsilon_{target} \sim N(\mathbf{0}, \beta_{target}^{-1} = \sigma_{target}^2)$. Based on the model assumption, the predictive value and variance of the simple GP regression for a data point $\mathbf{x}$ is defined as follows,

$$y = \mathbf{k}(\mathbf{x}, \mathbf{X})^T (\mathbf{K} + \sigma^2 \mathbf{I})^{-1} \mathbf{y}, \tag{4.4}$$

$$\sigma^2 = \mathbf{k}(\mathbf{x}, \mathbf{x}) - \mathbf{k}(\mathbf{x}, \mathbf{X})^T (\mathbf{K} + \beta_{target}^{-1} \mathbf{I})^{-1} \mathbf{k}(\mathbf{x}, \mathbf{X})^T. \tag{4.5}$$

In this definition, the kernel matrix K is the $N \times N$ matrix instead of $k \times k$ as PLR, and $\mathbf{k}(\mathbf{x}, \mathbf{X})$ is also $N \times 1$ column vector. Fig. 4.2 illustrates that the target uncertainty captured by GP regression. As shown in figure, GP regression may capture the extrapolated regions by its predictive variances. However, the target noise β_{target} is fundamentally induced in order to control the model complexity, not capture the extrapolated regions. Thus, the target noise affect the uncertainty over the dense region, as shown in Fig. 4.2 - the reconstruction noise of PLR does not affect the uncertainty information over the dense region. That is, it cannot capture the uncertain prediction due to extrapolations independently, unlike the PLR k-NN regression. Consequently, k-NN predictive variance using PLR

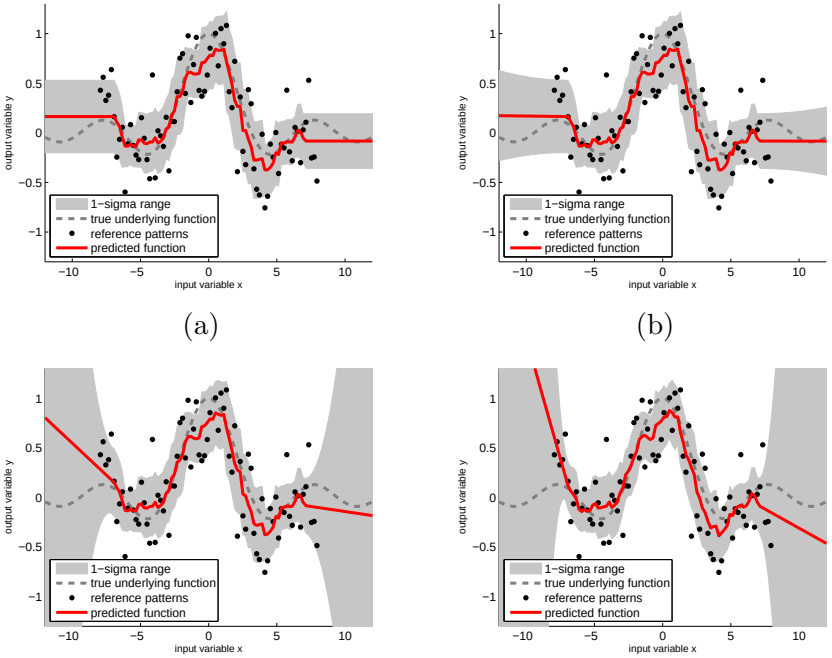
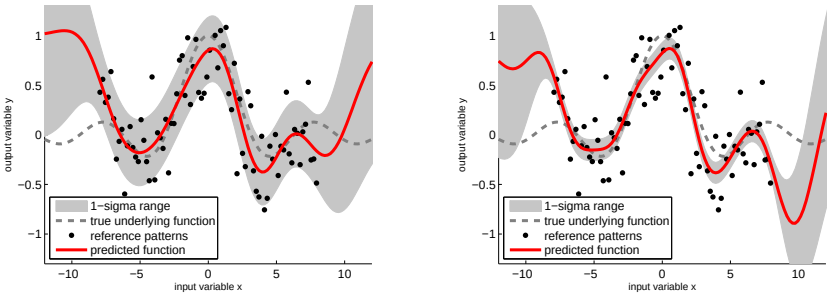


Figure 4.1: illustration of the k-NN predictive variances ($k = 10$ and $N = 200$) of (a) equally-weighted, (b) linear PLR ($\beta_{recon} = 0.001$), (c) linear PLR ($\beta_{recon} = 0.1$) and (d) linear PLR ($\beta_{recon} = 10$)

kernel is expected to be useful in various applications where there occur frequent sampling shifts (Salganicoff, 1997), which lead to the prediction reliance issue by extrapolations. Predictions of queries' target value with high PLR predictive variances may be rejected without unreliable prediction due to extrapolations as well as ambiguity, i.e., highly dispersed target values of neighboring points in k-NN regression. In this dissertation, as one of such applications, Virtual Metrology in the semiconductor industry is introduced for demonstrating the additional advantage obtained by using PLR kernel in k-NN regression.



(a) (b)

Figure 4.2: illustration of the predictive variances of Gaussian process regression with RBF kernel ($l = 2$) and (a) $\beta_{target} = 10$ and (b) $\beta_{target} = 100$

4.2 Reliance Level of Virtual Metrology in Semiconductor Manufacturing

The fabrication process in the semiconductor manufacturing consists of 300-500 highly complex processing steps (Chien et al., 2013; Kang et al., 2009, 2012; Su et al., 2007), quality of which is commonly evaluated as yield, or the fraction of total input transformed into shippable output, i.e., good chips with no defect (Cunningham et al., 1995). Since the yield cannot be monitored until the end of the overall process (Su et al., 2007) - which recently takes about three months as shown in Fig. 4.5, 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 (Su et al., 2007) becomes indispensable for sustaining yield performance at every step in conjunction with Run-to-Run (R2R) control (Chang et al., 2006; Kang et al., 2009, 2011). That is, metrology measurements such as thickness, resistance, Critical Dimension (CD), overlay, particles, etch rate, etc. (Su et al., 2007) are utilized as a feedback for adjusting the tool state variables such

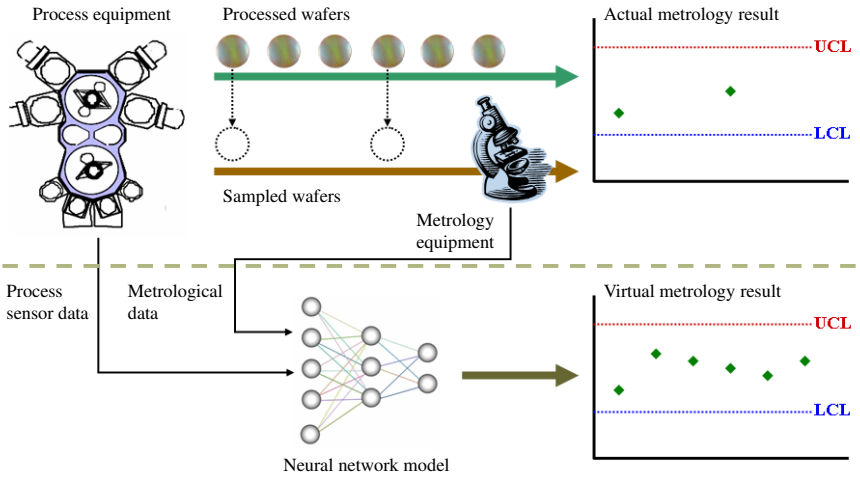


Figure 4.3: Comparison of actual metrology (upper) and virtual metrology (lower)(Kang et al., 2009)

as temperature, pressure, flow and current, etc. at a tool assigned for a processing step (Su et al., 2007). Due to time and cost, one wafer for one lot of 25 wafers (Kang et al., 2009, 2011) is actually measured (Su et al., 2007). In other words, the quality of every wafer cannot be assured in the lot-to-lot quality management (Chen et al., 2005; Kang et al., 2009, 2012; Lin et al., 2006). Virtual Metrology (VM) is defined as the technique to predict metrology measurements from tool state measurements without "actual" metrology operations (Chen et al., 2005), as shown in Fig. 4.3 and commonly formulated as a regression problem (Chen et al., 2005, 2006; Kang et al., 2009; Lin et al., 2006; Yung-Cheng and Cheng, 2005). As shown in Fig. 4.4, 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 (Kang et al., 2011).

Although the use of regression-based VM experimentally have shown

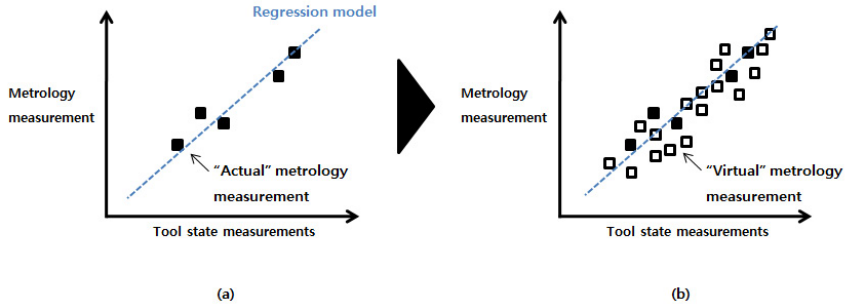


Figure 4.4: Concept of Virtual Metrology (VM): (a) learning the relation between a tool state measurement and "actual" metrology measurement, (b) predicting "virtual" metrology measurement for wafers with no metrology measurement

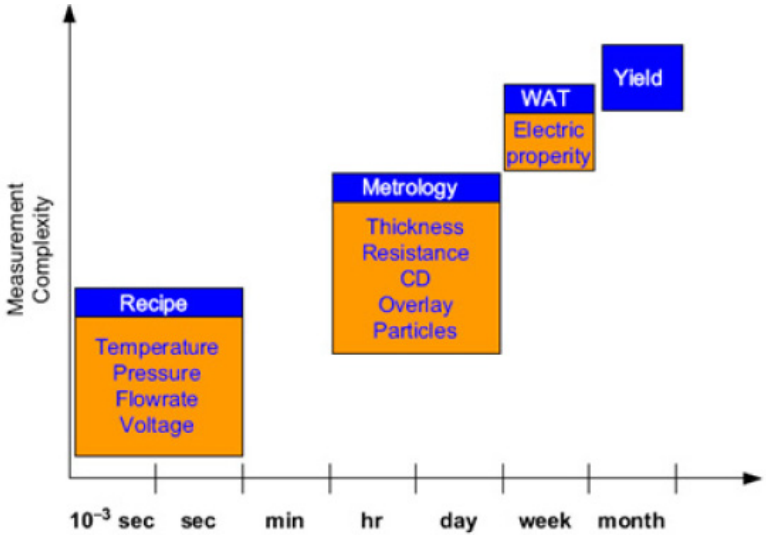


Figure 4.5: Multiple-time-scale characteristics for the process and quality measurements in semiconductor industry (Su et al., 2007)

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 (Cheng et al., 2008; Kang et al., 2012). Moreover, preventative maintenance and process drift due to daily wear and tear of tools in the fabrication process (Kang et al., 2011) changes the tool state and its relation with actual metrology measurements. It implies that a regressor built in one time may be "outdated" one in another time and extrapolate metrology measurements. It is desirable that these predictions extrapolated to "not-tested" or unknown regions in the tool state have lower reliance levels.

And in this application, some approaches for providing more reasonable reliance level of VM, taking account of process drifts have been currently researched (Cheng et al., 2008; Kang et al., 2012). Conventional approaches consider two unreliable situations separately, i.e., extrapolated one by the lack of training points, and ambiguous one by the highly dispersed target values of mutually similar training points. In this way, outlier detection models ahead of VM regressions are adopted in order to capture the extrapolated cases. On the other hand, PLR kernel in k-NN regression is not only suitable to the online prediction environment, but also provides one unified solution for error-bars of virtual measurements combining the reconstruction and target uncertainty. This predictive variance is defined in Eq. (4.2) and provides error-bars shown in Fig. 4.1(b). In the next section, some regressions estimating a prediction reliance without additional outlier detection models are compared in the real-world VM data.

4.3 Experimental Result on Real-World Virtual Metrology

The 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 usefulness of the PLR k-NN predictive variance, the simply first 600 wafers on time-rdered sequence of wafers measured in the metrology operation is focused. Fig. 4.6(a) shows changes of actual values for the chosen metrology variable on the discrete time index $t = 1, ..., 600$. Fig. 4.6(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 regression models, here Time-Weighted Forgetting (TWF) (Salganicoff, 1997), one of adaptive instance selection methods for k-NN learning[1] is used. 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, $w_{TWF} \leftarrow \eta w_{TWF}$, with $0 < \eta < 1$. 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 $N = \log\theta / \log\eta$ training pairs with forgetting older ones (Salganicoff, 1997). Consequently, regressions with TWF are always based on the current N (e.g., $N = 50$ in this work) training pairs.

Fig. 4.7 shows the result of applying the linear PLR k-NN regression with TWF to the VM data stream shown in Fig. 4.6. No variable selec-

[1] Further discussion of other adapting methods in changing environments such as the VM application is out of scope of this chapter (the problem due to changing environments is commonly referred to as concept drift problem (Widmer and Kubat, 1996), and see (Tsymbal, 2004; Zliobaite, 2009)), and leads to future work.

Table 4.1: RMSE ($\times 10^{-6}$) values for three probabilistic regressions

models	overall	reliable	unreliable
PLR-linear 5-NN	11,490	1,605(86%)	30,254(14%)
PLR-RBF 5-NN	1,744	1,722(93%)	1,992(7%)
GP	2,300	1,500(7%)	2,300(93%)

tion 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 Kang et al. (2009, 2011). Given k-invariance of PLR, the parameter k was set to five, to minimize the computational load and the parameter β_{recon} is selected among $10, 100, 1000$ by the five-fold cross-validation on training points. PLR k-NN regression, then, provides the predicted range of the virtual metrology value in terms of the predictive mean and variance, as shown in Fig. 4.1(b). The higher the predictive variance is, the larger the range is, implying the prediction is not reliable. When the threshold is defined as the variance covering 95% of training points selected by TWF, the actual RMSE of "reliable" and "unreliable" predictions are compared.

Table 4.1 shows the the VM prediction performances of GP and PLR k-NN regressions. GP regression is based on the Gaussian Process for Machine Learning (GPML) matlab toolbox[2]. Moreover, its (covariance) kernel is set to be the RBF kernel, and hyper-parameters of GP regression are optimized by its gradient descent based optimization method. For detecting unreliable predictions, the same logic with PLR is applied. Consequently, as shown in the table, by detecting unreliable predictions due to the change, PLR k-NN regression provides more reliable and accurate prediction performance than GP regression - GP regression is shown to be relatively inappropriate for detecting unreliable predictions. The result of the linear PLR k-NN regression demonstrates that there was a few (about

[2]http://www.gaussianprocess.org/gpml/code

14%) unreliable predictions, which lead to on average 20 times RMSE as shown in Table 4.1.

Further, contrary to a "reasonable" assumption or belief 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 $t = 295$ and 488 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. 4.8 shows training points selected by TWF (denoted as filled circles) and forgotten ones within the same period (denoted as unfilled circles) at $t = 295$ and 488, respectively. As shown in Fig. 4.8(a), it seems that the unreliable prediction at $t = 295$ was due to process drift. The tool state was slowly shifting from right to left. With the process drift, the query point at $t = 295$ is in the leftmost region. On the other hand, in Fig. 4.8(b), the query point at $t = 488$ (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. The PLR-based k-NN regressor provides the reliability level for the prediction, which is critical for detecting and ignoring unreliable predictions.

4.4 Summary and Discussion

In this chapter, the k-NN predictive variance based on PLR kernel quantifying both the reconstruction and target uncertainties is proposed. It is expected to be useful in various real-world applications where the prediction reliance issue due to extrapolations is critical. As one of such applications, Virtual Metrology (VM) in the semiconductor industry is introduced. The PLR-based k-NN predictive variance is utilized as the reliance level of virtual predictions. The experimental result on the real-world virtual metrol-

Table 4.2: RMSE ($\times 10^{-6}$) values for various regression models

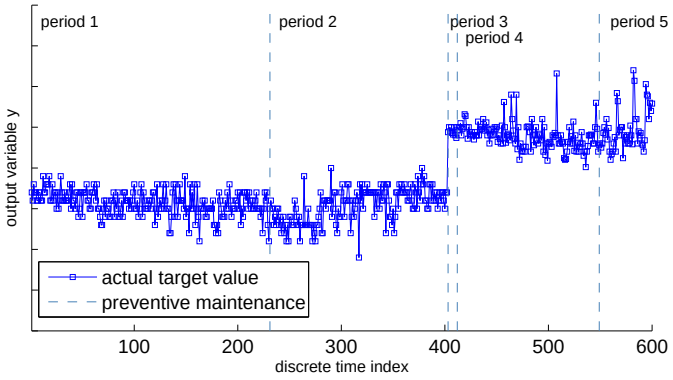
models	overall	reliable	unreliable
unweighted 5-NN	1,748	-	-
RBF 5-NN	1,893	-	-
LLR 5-NN	11,562	-	-
PLR-linear 5-NN	11,490	1,605(86%)	30,254(14%)
PLR-RBF 5-NN	1,744	1,722(93%)	1,992(7%)
LinReg	32,461	-	-
GP	2,300	1,500(7%)	2,300(93%)

ogy showed that the PLR-based k-NN regressor provides the reasonable prediction reliance, which is critical for detecting and rejecting unreliable predictions. It is expected to reduce the risk in the actual use of VM in the industry.

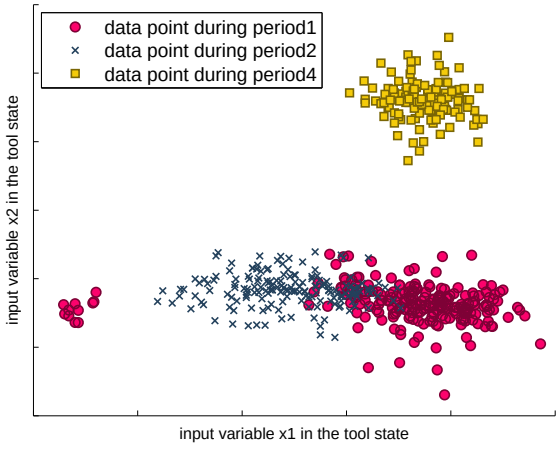
And the VM prediction performances of various regression models are additionally summarized in Table 4.2. In the table, it is noted that unweightd k-NN regression may be superior to other sophisticated ones. It is due to that the grosser features of the problem learned by simpler models are more likely to persist than the smaller features learned by the more sophisticated models in the changing environment (Hand, 2006). In other words, the prediction performance of a simler model declines less (or be more robust) than that of more complex one in the changing environment. However, PLR k-NN prediction reliance captures inconsistent regions occurred by process drifts and rejects predictions in the corresponding regions. Thus, as shown in the experimental result, PLR k-NN prediction reliance can provide an appropriate reliance level for VM predictions, i.e., high reliance levels to the wafers with small RMSEs, low levels to the wafers with large RMSEs.

In other perspective view, PLR k-NN prediction reliance may be used for adaptation of prediction models, rather than rejection of predictions. In this chapter, for adaptation of k-NN regression, TWF (Salganicoff, 1997), one of adaptive instance selection methods was used. However,

since TWF considers temporal relevance in the instance selection, it suffers from time-varying sampling distributions (Beringer and Hüllermeier, 2007; Salganicoff, 1997). Thus, Locally-Weighted Forgetting (LWF) (Salganicoff, 1997) was proposed to tackle the limitation. It considers spatial relevance, i.e., a new training point supersedes the previous similar one (Salganicoff, 1997). Furthermore, the PLR k-NN prediction reliance may be useful in the locally-weighted learning and forgetting for adapting to changing environments, meriting of further investigation.



(a)



(b)

Figure 4.6: VM data exploration (a) metrology measurements changes after preventive maintenances (b) tool state changes after preventive maintenances.

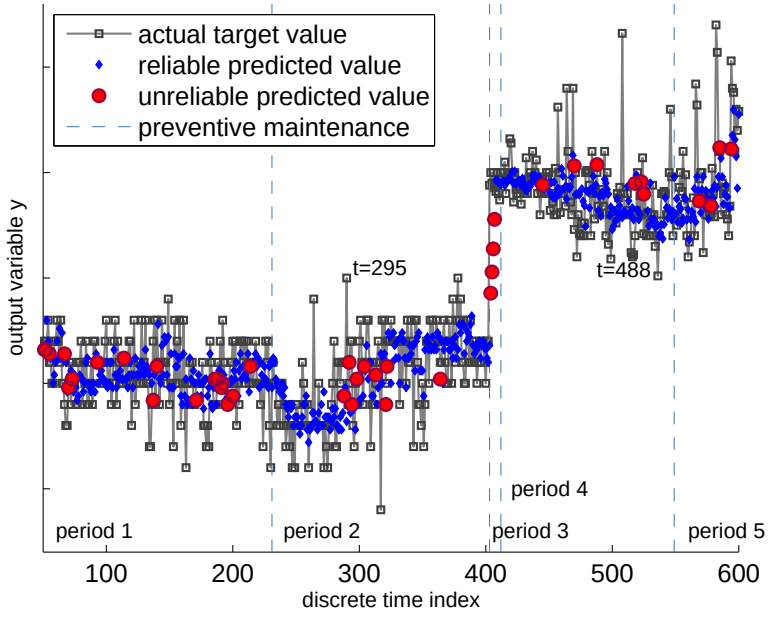
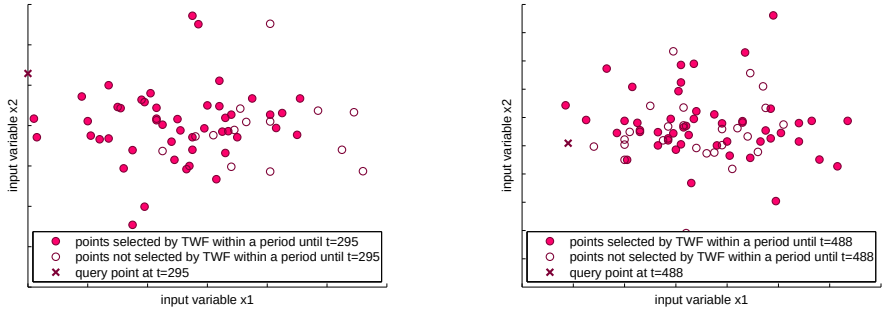


Figure 4.7: VM prediction result of the linear PLR-based 5-NN regression



(a) (b)

Figure 4.8: Why are VM predictions unreliable at $t = 295$ and $t = 488$? The predictions are based on relatively "unrelated" training points.

Chapter 5

Conclusion

5.1 Summary and Contributions

In this dissertation, the two research issues of LLR kernel, one of data-dependent kernel in kernelized k-NN regression were considered: 1) flexibility of locally linear reconstruction and 2) probabilistic quantification of the reconstruction uncertainty:

- **Flexibility of locally linear reconstruction**

  First, fitting the 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 kernel. A small k with only a handful of neighbors may not guide LLR correctly. On the other hand, with a large k, the non-linear topology embedded in too redundant neighbors cannot be captured.

- **Probabilistic quantification of the reconstruction uncertainty**

  Second, in practice, neighbors for a query point are often not informative to describe the query. It is desirable to provide a probabilistic estimate, providing the answer for the question, "how informative is the set of k neighbors in LLR?". Because the target uncertainty

should be also considered in k-NN prediction. However, LLR provides a point estimate.

And a Bayesian kernel treatment of LLR kernel, named Probabilistic Local Reconstruction (PLR) was proposed in this dissertation. There two concepts - Bayesian formulation of LLR and its additional kernelized implementation - give solutions for the aforementioned research issues:

- **Bayesian treatment of LLR**

  The probabilistic extension of LLR explains the intuitive control mechanism of the locally linear reconstruction for small k settings and captures the reconstruction uncertainty. LLR is probabilistically explained as the learning process of adjusting the reconstructed point at the average point of k neighbors - expected by the prior assumption - to be close to the corresponding query point. The learning degree is controlled by the additive noise assumption of PLR. In this way, the probabilistic extension explains how the PLR kernel can be robust to noise or outliers in its reconstruction phase. Moreover, the probabilistic estimate of PLR quantifies the reconstruction uncertainty due to lack of neighboring points and provides more reasonable prediction reliance for k-NN prediction as a form of "error-bars" by incorporated with the target uncertainty. The PLR k-NN predictive variance in Virtual Metrology (VM) problem in the semiconductor industry is shown to be useful to capture both two uncertainties - reconstruction uncertainty by extrapolation and target uncertainty by ambiguity.

- **Kernelized implementation**

  In order to capture the non-linear topology, the kerenlized implementation of PLR was provided - i.e., transforming a lower-dimensional input space into a higher-dimensional feature space without the need

for explicit mapping function. It captures the non-linear or locally biased topology embedded in too redundant neighbors with large k settings in k-NN predictions. In this way, PLR is shown to be more k-invariant and accurate than LLR in k-NN regression.

5.2 Further Research Issues

Some research issues are summarized as follows:

- **Is it practically reasonable to Gaussian noise in the local reconstruction?**

  In the proposed PLR formulation, the reconstruction error or distance between a query point and its reconstructed point in terms of its k neighboring points is assumed to follow the (isotropic) Gaussian distribution. Although the model complexity of PLR increases, other distributions such as ARD (Automatic Relevance Determination) Gaussian distribution may be more suitable to express the reconstruction distance in the input space.

- **PLR-based locally-weighted learning and forgetting**

  In this dissertation, for adaptation of k-NN regression, TWF (Salganicoff, 1997), one of adaptive instance selection methods was used. However, since TWF considers temporal relevance in the instance selection, it suffers from time-varying sampling distributions (Beringer and Hüllermeier, 2007; Salganicoff, 1997). Thus, Locally-Weighted Forgetting (LWF) (Salganicoff, 1997) was proposed to tackle the limitation. It considers spatial relevance, i.e., a new training point supersedes the previous similar one (Salganicoff, 1997). Furthermore, the reconstruction probability density of PLR seems to be useful in the locally-weighted learning and forgetting for changing environments, meriting of further investigation.

- **PLR kerenl in the semi-supervised learning problem**

  LLE, LLR and PLR assesses similarity of two data points in terms of neighboring points in the training set (Scholköpf and Smola, 2002), thus, captures the local topology. In this way, the data-dependent kernels between two data points whose target values are known, i.e., labeled points can consider the local topology of neighboring data points, regardless of they are labeled or not. This property is expected to be useful in the semi-supervised learning problem, which is introduced for avoiding over-fitting of supervised learning techniques due to the low sampling size of labeled data point when a small amount of labeled points with a large amount of unlabeled points is given.

Bibliography

Aha, D. W., Goldstone, R. L., 1992. Concept learning and flexible weighting. In: Proceedings of the 14th Annual Conference of the Cognitive Science Society. Erlbaum, pp. 534–539.

Ahmed, T., Oreshkin, B., Coates, M., 2007. 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, pp. 7:1–7:6.

Belkin, M., Niyogi, P., Jun. 2003. Laplacian eigenmaps for dimensionality reduction and data representation. Neural Computation 15 (6), 1373–1396.

Bengio, Y., Delalleau, O., Roux, N. L., Paiement, J.-F., Vincent, P., Ouimet, M., 2004. Learning eigenfunctions links spectral embedding and kernel pca. Neural Computation 16, 2004.

Bengio, Y., Paiement, J.-F., Vincent, P., 2003a. Out-of-sample extensions for lle, isomap, mds, eigenmaps, and spectral clustering. In: Advances in Neural Information Processing Systems. MIT Press, pp. 177–184.

Bengio, Y., Vincent, P., Paiement, J.-F., Delalleau, O., Ouimet, M., Roux,

N. L., 2003b. Spectral clustering and kernel pca and learning eigenfunctions. Technical report, Département d'informatique et recherche opérationnelle, Université de Montréal.

Beringer, J., Hüllermeier, E., Dec. 2007. Efficient instance-based learning on data streams. Intelligent Data Analysis 11 (6), 627–650.

Bottou, L., Vapnik, V., 1992. Local learning algorithms. Neural Computation 4, 888–900.

Burges, C. J. C., 2009. Dimension reduction: A guided tour. Foundations and Trends in Machine Learning 2 (4), 275–365.

Chang, Y.-J., Kang, Y., Hsu, C.-L., Chang, C.-T., Chan, T. Y., 2006. Virtual metrology technique for semiconductor manufacturing. In: International Joint Conference on Neural Networks (IJCNN 2006). pp. 5289–5293.

Chen, P., Wu, S., Lin, J., Ko, F., Lo, H., Wang, J., Yu, C., Liang, M., 2005. Virtual metrology: a solution for wafer to wafer advanced process control. In: IEEE International Symposium on Semiconductor Manufacturing (ISSM 2005). pp. 155–157.

Chen, Y., Garcia, E. K., Gupta, M. R., Rahimi, A., Cazzanti, L., Jun. 2009. Similarity-based classification: Concepts and algorithms. Journal of Machine Learning Research 10, 747–776.

Chen, Y.-T., Yang, H.-C., Cheng, F.-T., 2006. Multivariate simulation assessment for virtual metrology. In: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA 2006). pp. 1048–1053.

Cheng, F.-T., Chen, Y.-T., Su, Y.-C., Zeng, D.-L., 2008. Evaluating re-

liance level of a virtual metrology system. IEEE Transactions on Semiconductor Manufacturing 21 (1), 92–103.

Chien, C.-F., Hsu, C.-Y., Chen, P.-N., 2013. Semiconductor fault detection and classification for yield enhancement and manufacturing intelligence. Flexible Services and Manufacturing Journal 25 (3), 367–388.

Coppersmith, D., Winograd, S., March 1990. Matrix multiplication via arithmetic progressions. J. Symb. Comput. 9, 251–280.

Cunningham, S., Spanos, C. J., Voros, K., 1995. Semiconductor yield improvement: Results and best practices. IEEE Transactions on Semiconductor Manufacturing 8 (2), 103–109.

De Stefano, C., Sansone, C., Vento, M., 2000. To reject or not to reject: That is the question-an answer in case of neural classifiers. IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews 30 (1), 84–94.

Dubuisson, B., Masson, M., 1993. A statistical decision rule with incomplete knowledge about classes. Pattern Recognition 26 (1), 155 – 165.

Dudani, S. A., 1976. The distance-weighted k-nearest-neighbor rule. IEEE Transactions on Systems, Man and Cybernetics SMC-6 (4), 325–327.

Fayyad, U., Piatetsky-shapiro, G., Smyth, P., 1996. From data mining to knowledge discovery in databases. AI Magazine 17, 37–54.

Gan, Y., Guan, J., Zhou, S., 2009. A pattern-based nearest neighbor search approach for promoter prediction using dna structural profiles. Bioinformatics 25 (16), 2006–2012.

Ghodsi, A., 2006. Dimensionality reduction, a short tutorial. Technical report, Department of Statistics and Actuarial Science, University of Waterloo, Ontario, Canada.

Hadsell, R., Chopra, S., LeCun, Y., 2006. Dimensionality reduction by learning an invariant mapping. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition. Vol. 2. pp. 1735–1742.

Ham, J., Lee, D. D., Mika, S., Schölkopf, B., 2004. A kernel view of the dimensionality reduction of manifolds. In: Proceedings of the twenty-first international conference on Machine learning (ICML2004). ACM, New York, NY, USA, pp. 47–.

Hand, D. J., 2006. Classifier technology and the illusion of progress. Statistical Science 21 (1), 1–15.

Hofmann, T., Schölkopf, B., Smola, A. J., 2008. Kernel methods in machine learning. The Annals of Statistics 36 (3), 1171–1220.

Horn, R., Johnson, C., 1990. Matrix Analysis. Cambridge University Press.

Jain, A. K., Murty, M. N., Flynn, P. J., Sep. 1999. Data clustering: a review. ACM Computing Survey 31 (3), 264–323.

Jing, X.-Y., Wong, H.-S., Zhang, D., 2006. Face recognition based on discriminant fractional fourier feature extraction. Pattern Recognition Letters 27 (13), 1465–1471.

Kang, P., Cho, S., 2008. Locally linear reconstruction for instance-based learning. Pattern Recognition 41, 3507–3518.

Kang, P., Cho, S., Nov. 2009. A hybrid novelty score and its use in keystroke dynamics-based user authentication. Pattern Recognition 42 (11), 3115–3127.

Kang, P., joo Lee, H., Cho, S., Kim, D., Park, J., Park, C.-K., Doh, S., 2009. A virtual metrology system for semiconductor manufacturing. Expert Systems with Applications 36, 12554–12561.

Kang, P., Kim, D., joo Lee, H., Doh, S., Cho, S., 2011. Virtual metrology for run-to-run control in semiconductor manufacturing. Expert Systems with Applications 38 (3), 2508 – 2522.

Kang, P., Kim, D., kyung Lee, S., Dho, S., Cho, S., 2012. Estimating the reliability of virtual metrology predictions in semiconductor manufacturing:A novelty detection-based approach. Journal of the Korean Institute of Industrial Engineers 38 (1), 46–56.

Kayo, O., 2006. Locally linear embedding algorithm. Extensions and applications. technical report, Faculty of Technology, University of Oulu, Department of Electrical and Information Engineering, University of Oulu, Finland.

Kim, D., Cho, S., 2008. Bootstrap based pattern selection for support vector regression. In: Proceedings of the 12th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD2008). Vol. 5012 of Lecture Notes in Artificial Intelligence. pp. 608–615.

Kim, D., Finkel, L., 2003. Hyperspectral image processing using locally linear embedding. In: Proceedings of the first International IEEE EMBS Conference on Neural Engineering. pp. 316–319.

Kozak, K., Kozak, M., Stapor, K., 2006. Weighted k-nearest-neighbor techniques for high throughput screening data. International Journal of Biological and Life Sciences 1 (3), 155–160.

Landgrebe, T. C. W., Tax, D. M. J., Paclík, P., Duin, R. P. W., Jun. 2006. The interaction between classification and reject performance for distance-based reject-option classifiers. Pattern Recognition Letter 27 (8), 908–917.

Li, H., Dai, X., Zhao, X., 2008. A nearest neighbor approach for automated

transporter prediction and categorization from protein sequences. Bioinformatics 24 (9), 1129–1136.

Lin, T.-H., Hung, M.-H., Lin, R.-C., Cheng, F.-T., 2006. A virtual metrology scheme for predicting cvd thickness in semiconductor manufacturing. In: Proceedings of the IEEE International Conference on Robotics and Automation (ICRA 2006). pp. 1054–1059.

Ma, L., Crawford, M., Tian, J., 2010. Local manifold learning-based k-nearest-neighbor for hyperspectral image classification. IEEE Transactions on Geoscience and Remote Sensing 48 (11), 4099–4109.

Macleod, J., Luk, A., Titterington, D., 1987. A re-examination of the distance-weighted k-nearest neighbor classification rule. IEEE Transactions on Systems, Man and Cybernetics 17 (4), 689–696.

Maimon, O., Rokach, L., 2010. The Data Mining and Knowledge Discovery Handbook. The Kluwer International Series in Engineering and Computer Science. Springer.

Mitchell, T. M., 1997. Machine Learning, 1st Edition. McGraw-Hill, Inc., New York, NY, USA.

Munoz, A., Moguerza, J., 2006. Estimation of high-density regions using one-class neighbor machines. Pattern Analysis and Machine Intelligence, IEEE Transactions on 28 (3), 476–480.

O'mahony, M., Hurley, N., Kushmerick, N. Silvestre, G., 2003. Collaborative recommendation: A robustness analysis. ACM Transactions on Internet Technology 4 (4), 344–377.

Rasmussen, C. E., Williams, C. K. I., 2006. Gaussian processes for machine learning. MIT Press.

Roberts, S., Penny, W., Pillot, D., 1996. Novelty, confidence and errors in connectionist systems. In: IEE Colloquium on Intelligent Sensors and Fault Detection. pp. 10/1–10/6.

Roweis, S. T., Saul, L. K., 2000. Nonlinear dimensionality reduction by locally linear embedding. Science 290, 2323–2326.

Rubio, G., Herrera, L. J., Pomares, H., Rojas, I., Guillén, A., 2010. Design of specific-to-problem kernels and use of kernel weighted k-nearest neighbors for time series modelling. Neurocomputing 73, 1965–1975.

Ruprecht, D., Müller, H., 1994. A framework for generalized scattered data interpolation. Tech. rep., Universität Dortmund, Fachbereich Informatik, D-44221, Dortmund, Germany.

Salganicoff, M., 1997. Tolerating concept and sampling shift in lazy learning using prediction error context switching. Artificial Intelligence Review 11 (1-5), 133–155.

Saul, L. K., Roweis, S. T., Dec. 2003. Think globally, fit locally: unsupervised learning of low dimensional manifolds. Journal of Machine Learning Research 4, 119–155.

Saul, L. K., Weinberger, K. Q., Ham, J. H., Sha, F., Lee, D. D., 2006. Spectral methods for dimensionality reduction. Semisupervised Learning.

Schölkopf, B., Smola, A., Müller, K.-R., Jul. 1998. Nonlinear component analysis as a kernel eigenvalue problem. Neural Computation 10 (5), 1299–1319.

Scholköpf, B., Smola, A. J., 2002. Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond. MIT Press, Cambridge, MA, USA.

Shmueli, G., Patel, N., Bruce, P., 2011. Data Mining for Business Intelligence: Concepts, Techniques, and Applications in Microsoft Office Excel with XLMiner. Wiley.

Singh, S., Haddon, J., Markou, M., 2001. Nearest-neighbor classifiers in natural scene analysis. Pattern Recognition 34 (8), 1601–1612.

Su, A.-J., Jeng, J.-C., Huang, H.-P., Yu, C.-C., Hung, S.-Y., Chao, C.-K., 2007. Control relevant issues in semiconductor manufacturing: Overview with some new results. Control Engineering Practice 15 (10), 1268 – 1279.

Tax, D. M. J., Duin, R. P. W., Jan. 2004. Support vector data description. Machine Learning 54 (1), 45–66.

Tenenbaum, J. B., de Silva, V., Langford, J. C., 2000. A global geometric framework for nonlinear dimensionality reduction. Science 290, 2319–2323.

Tipping, M. E., 2001. Sparse bayesian learning and the relevance vector machine. Journal of Machine Learning Research 1, 211–244.

Tryon, R., Bailey, D., 1970. Cluster analysis. McGraw-Hill.

Tsymbal, A., 2004. The problem of concept drift: Definitions and related work. Tech. rep.

Vapnik, V. N., 1995. The Nature of Statistical Learning Theory. Springer-Verlag New York, Inc., New York, NY, USA.

Wand, M. P., Schucany, W. R., 1990. Gaussian-based kernels. The Canadian Journal of Statistics 18 (3), 194–204.

Widmer, G., Kubat, M., 1996. Learning in the presence of concept drift and hidden contexts. Machine Learning 23 (1), 69–101.

Williams, C., Seeger, M., 2001. Using the nystrom method to speed up kernel machines. In: Advances in Neural Information Processing Systems. Vol. 13. MIT Press, pp. 682–688.

Wolberg, G., 1990. Digital image warping. IEEE Computer Society Press, Los Alamitos, CA, USA.

Yakowitz, S., 2008. Nearest-neighbor methods for time series analysis. Journal of Time Series Analysis 8 (2), 235–247.

Yu, K., Ji, L., Zhang, X., 2002. Kernel nearest-neighbor algorithm. Neural Processing Letters 15 (2), 147–156.

Yung-Cheng, J. C., Cheng, F.-T., 2005. Application development of virtual metrology in semiconductor industry. In: Proceedings of the 31st Annual Conference of IEEE Industrial Electronics Society (IECON 2005).

Zavrel, J., 1997. An empirical re-examination of weighted voting for k-nn. In: Proceedings of the 7th Belgian-Dutch Conference on Machine Learning. pp. 139–148.

Zhang, B., Elkan, C., Dayal, U., Hsu, M., 2000. A model-independent measure of regression difficulty. Tech. rep., Hewlett-Packard Laboratories.

Zhang, Z., Zha, H., 2005. Principal manifolds and nonlinear dimensionality reduction via tangent space alignment. SIAM Journal of Scientific Computing 26 (1), 313–338.

Zhou, D., Li, J., Ma, W., 2009. Clustering based on lle for financial multivariate time series. In: Proceedings of the International Conference on Management and Service Science (MASS 2009). pp. 1–4.

Zliobaite, I., 2009. Learning under concept drift: An overview. Tech. rep., Vilnius University.