Localized Projection Learning

Kazuki Tsuji, Mineichi Kudo, and Akira Tanaka

Division of Computer Science,
Graduate School of Information Science and Technology,
Hokkaido University, Sapporo, 060-0814, Japan
{kazuki,mine,takira}@main.ist.hokudai.ac.jp

Abstract. It is interesting to compare different criteria of kernel machines. In this paper, the following is made: 1) to cope with the scaling problem of projection learning, we propose a dynamic localized projection learning using k nearest neighbors, 2) the localized method is compared with SVM from some viewpoints, and 3) approximate nearest neighbors are demonstrated their usefulness in such a localization. As a result, it is shown that SVM is superior to projection learning in many classification problems in its optimal setting but the setting is not easy.

1 Introduction

In pattern recognition, the design of a classifier can be made through regression. To achieve this, a dummy output y is introduced instead of the class-label output, e.g., $y = +1$ for one class and $y = -1$ for another class in two-class problems as seen in the paradigm of the support vector machines (SVMs) [1]. The goal of this type of classifiers is to estimate the target function f such as $y = f(\boldsymbol{x})$, where $\boldsymbol{x} \in \mathbf{R}^p$ is a sample with p input features and $y \in \mathbf{R}$ is the corresponding output. We try to find a good approximator/regressor $\hat{f}$ from a limited number of training sample pairs $(\boldsymbol{x}_i, y_i)$ $(i = 1, \ldots, \ell)$. According to whether there is noise or not, a regressor or an approximator becomes more appropriate than the other.

In this paper we compare two typical but criterion-different kernel machines: SVM [1] and Projection Learning [2]. The goal of this paper is described in three-fold: 1) to propose a localized projection learning to bring scalability into the projection learning, 2) to examine how well the proposed localized projection learning is competitive to the original projection learning, and 3) using the localized projection learning, to compare the projection learning and the SVM in performance and in speed.

2 Reproducing Kernel Hilbert Spaces

Reproducing kernel Hilbert spaces [3,4] is the key concept to interpret above approaches in a unified framework.

E.R. Hancock et al. (Eds.): SSPR & SPR 2010, LNCS 6218, pp. 90–99, 2010.

Definition 1. *[3] Let $\mathbf{R}^p$ be a p-dimensional real vector space and let $\mathcal{H}$ be a class of functions defined on $\mathcal{D} \subset \mathbf{R}^p$, forming a Hilbert space of real-valued functions. A function $K(\boldsymbol{x}, \tilde{\boldsymbol{x}})$ $(\boldsymbol{x}, \tilde{\boldsymbol{x}} \in \mathcal{D})$ is called the reproducing kernel if*

1. *For every $\tilde{\boldsymbol{x}} \in \mathcal{D}$, $K(\boldsymbol{x}, \tilde{\boldsymbol{x}})$ is a function of $\boldsymbol{x}$ belonging to $\mathcal{H}$.*
2. *For every $\tilde{\boldsymbol{x}} \in \mathcal{D}$ and every $f \in \mathcal{H}$, $f(\tilde{\boldsymbol{x}}) = \langle f(\boldsymbol{x}), K(\boldsymbol{x}, \tilde{\boldsymbol{x}})\rangle_{\mathcal{H}}$, where $\langle \cdot, \cdot \rangle_{\mathcal{H}}$ denotes the inner product of the Hilbert space $\mathcal{H}$.*

The Hilbert space $\mathcal{H}$ that has a reproducing kernel is called a reproducing kernel Hilbert space (RKHS). The *reproducing property* (the second condition) enables us to treat a function value by the inner product of two elements of $\mathcal{H}$.

Next, we introduce the Schatten product [5] that is a convenient tool to reveal the reproducing property of kernels.

Definition 2. *[5] Let $\mathcal{H}_1$ and $\mathcal{H}_2$ be Hilbert spaces. The Schatten product of $g \in \mathcal{H}_2$ and $h \in \mathcal{H}_1$ is defined by*

$$(g \otimes h)f = \langle f, h\rangle_{\mathcal{H}_1} g, \quad f \in \mathcal{H}_1.$$

Note that $(g \otimes h)$ is a linear operator from $\mathcal{H}_1$ onto $\mathcal{H}_2$. It is easy to show that the following relations hold for $h, v \in \mathcal{H}_1$, $g, u \in \mathcal{H}_2$.

$$(h \otimes g)^* = (g \otimes h), \quad (h \otimes g)(u \otimes v) = \langle u, g\rangle_{\mathcal{H}_2}(h \otimes v),$$

where the superscript * denotes the adjoint operator.

3 Projection Learning (PL)

Let $\{(y_i, \boldsymbol{x}_i) | i = 1, \ldots, \ell\}$ be a given training data set $\boldsymbol{x}_i \in \mathbf{R}^p, y_i \in \mathbf{R}$, satisfying

$$y_i = f(\boldsymbol{x}_i) + n_i,$$

where f denotes the true function and n_i denotes a zero-mean additive noise.

In this paper, we assume that the unknown function f belongs to the RKHS $\mathcal{H}_K$ with the kernel function K. If $f \in \mathcal{H}_K$, then by the reproducing property above equation is rewritten as

$$y_i = \langle f(\boldsymbol{x}), K(\boldsymbol{x}, \boldsymbol{x}_i)\rangle_{\mathcal{H}_K} + n_i.$$

Let $\boldsymbol{y} = [y_1, \ldots, y_\ell]'$ and $\boldsymbol{n} = [n_1, \ldots, n_\ell]'$ with the superscript $'$ denoting the transposed matrix (or vector), then applying the Schatten product yields

$$\boldsymbol{y} = \left(\sum_{k=1}^{\ell} [\boldsymbol{e}_k^{(\ell)} \otimes K(\boldsymbol{x}, \boldsymbol{x}_k)] \right) f(\boldsymbol{x}) + \boldsymbol{n}, \tag{1}$$

where $\boldsymbol{e}_k^{(\ell)}$ denotes the kth vector of the canonical basis of $\mathbf{R}^\ell$. For a convenience of description, with the sample set $\boldsymbol{X} = \{x_1, \ldots, x_\ell\}$, we write

$$A_{K,X} = \left(\sum_{k=1}^{\ell} [\boldsymbol{e}_k^{(\ell)} \otimes K(\boldsymbol{x}, \boldsymbol{x}_k)] \right).$$

Here $A_{K,X}$ is a linear operator that maps an element of $\mathcal{H}_K$ onto $\mathbf{R}^\ell$. Then Eq.(1) can be simply written by

$$\boldsymbol{y} = A_{K,X} f + \boldsymbol{n}. \tag{2}$$

That is, the result of sampling of f on $\boldsymbol{X}$ of fixed ℓ samples was contaminated by noise $\boldsymbol{n}$ and observed as $\boldsymbol{y}$.

Projection learning [2] is derived to attain the minimum squared error on $\boldsymbol{X}$ between the target function f and an estimator $\hat{f}$ measured in $\mathcal{H}$ in the case of zero-mean noise. By solving (2) without $\boldsymbol{n}$, such an optimal estimator is given as

$$\hat{f}(\cdot) = A^+_{K,X}\boldsymbol{y} = A^*_{K,X}(A_{K,X}A^*_{K,X})^+\boldsymbol{y} \tag{3}$$

$$= \sum_{k=1}^{\ell} \boldsymbol{y}' G^+_{K,X} \boldsymbol{e}^{(\ell)}_k K(\cdot, \boldsymbol{x}_k), \tag{4}$$

where $*$ is the adjoint matrix, $+$ is the Moore-Penrose generalized inverse, and $G_{K,X} = (K(\boldsymbol{x}_i, \boldsymbol{x}_j))$ is the $\ell \times \ell$ "Gram matrix."

When no error exists, Eq.(3) can be rewritten as

$$\hat{f} = A^*_{K,X}(A_{K,X}A^*_{K,X})^+ A_{K,X} f = P_{R(A^*_{K,X})} f = \sum_{k=1}^{\ell} \alpha_k K(\boldsymbol{x}, \boldsymbol{x}_k).$$

As a result, we have $\hat{f}$ as the minimizer of

$$J_{PL}(\hat{f}) = ||f - \hat{f}||^2_{\mathcal{H}_K}, \hat{f} \in \mathrm{span}\{K(\cdot, x_1), K(\cdot, x_2), \ldots, K(\cdot, x_\ell)\},$$

where $\mathrm{span}\{\cdot\}$ is the closure of linear combinations of the elements.

It is also easy to show that $\hat{f}$ is equivalent to f on a sample set $\boldsymbol{X}$, that is, with $\boldsymbol{f} = [f(\boldsymbol{x}_1), \ldots, f(\boldsymbol{x}_\ell)]'$ and $\hat{\boldsymbol{f}} = [\hat{f}(\boldsymbol{x}_1), \ldots, \hat{f}(\boldsymbol{x}_\ell)]'$, $\hat{\boldsymbol{f}} \equiv \boldsymbol{f}$ holds as long as $G^+ = G^{-1}$.

The solution by PL is optimal in the sense of minimum squared error if no error is considered. However, things change when noise is taken into account. A clear relationship for the case is given by Tanaka *et al.* [6,7]. Roughly speaking, as the number of training samples increases, the projection error is reduced but the error caused by noise is increased, so that the total approximation error $||f - \hat{f}||^2_{\mathcal{H}_k}$ is unknown on whether reduced or increased. When noise exists, from (2) and (3), our estimate becomes

$$\hat{f} = A^+_{K,X}\boldsymbol{y} = A^+_{K,X}(A_{K,X}f + \boldsymbol{n}) = A^+_{K,X}A_{K,X}f + A^+_{K,X}\boldsymbol{n} = P_{R(A^*_{K,X})}f + A^+_{K,X}\boldsymbol{n}.$$

In [6], $||P_{R(A^*_{K,X})}f||^2_{\mathcal{H}_K} = \boldsymbol{f}' G^+_{K,X}\boldsymbol{f}$ and, in [7], $||A^+_{K,X}\boldsymbol{n}||^2_{\mathcal{H}_K} = \boldsymbol{n}' G^+_{K,X}\boldsymbol{n}$ were shown. Thus, we have the following relationship:

$$||f - \hat{f}||^2_{\mathcal{H}_K} = ||f - P_{R(A^*_{K,X})}f||^2_{\mathcal{H}_K} + ||A^+_{K,X}\boldsymbol{n}||^2_{\mathcal{H}_K}$$

$$= ||f||^2_{\mathcal{H}_K} - \boldsymbol{f}' G^+_{K,X}\boldsymbol{f} + \boldsymbol{n}' G^+_{K,X}\boldsymbol{n}. \tag{5}$$

Since $G^{+}_{K,X}$ is symmetric and non-negative definite, the second and third terms increase in their absolute values as the sizes of $\boldsymbol{f}, \boldsymbol{n}, G_{K,X}$ increase with increasing samples. Therefore a trade-off arises.

4 Support Vector Machines (SVM)

Although it was originally formulated for classification as the norm minimization under a separation condition, support vector machines are also seen as a regression algorithm. Indeed the criterion in a regression form is given by

$$J_{SVM}(\hat{f}) = \frac{1}{\ell}\sum_{i=1}^{\ell} |1 - y_i \hat{f}(x_i)|_{+} + \lambda \|\hat{h}\|^2_{\mathcal{H}_K}, \tag{6}$$

where $|\cdot|_{+}$ is a function to remain the value for a positive value and zero otherwise. Here, f is decomposed as $\hat{f} = \hat{h} + \hat{c}$ and $\hat{h} \in \mathcal{H}$ and $\hat{c}$ is a constant and $P\hat{f} = \hat{h}$. Therefore, if $\{1\} \subset \mathcal{H}_K$ then this equation is rewritten as

$$J_{SVM}(\hat{f}) = \frac{1}{\ell}\sum_{i=1}^{\ell} |1 - y_i \hat{f}(x_i)|_{+} + \lambda \|P\hat{f}\|^2_{\mathcal{H}_K}, \tag{7}$$

where $P\hat{f}$ is the projection of $\hat{f}$ into the orthogonal complement of $\{1\}$ in $\mathcal{H}_K$.

It is also known that the minimizer of this criterion has the form $\hat{f}(\cdot) = \sum_{i=1}^{c} c_i K(\cdot, x_i) + c_0$. That is, $P\hat{f}$ is the projection of $\hat{f}$ into $\mathrm{span}\{K(\cdot, x_1), K(\cdot, x_2), \ldots, K(\cdot, x_\ell)\}$.

5 Localized Projection Learning (LPL)

Projection learning (4) can be expressed explicitly as

$$\begin{aligned}\hat{f}(\boldsymbol{x}) &= (K(\boldsymbol{x}, \boldsymbol{x}_1), \ldots, K(\boldsymbol{x}, \boldsymbol{x}_\ell)) \begin{pmatrix} K(\boldsymbol{x}_1, \boldsymbol{x}_1) & K(\boldsymbol{x}_1, \boldsymbol{x}_2) & \cdots & K(\boldsymbol{x}_1, \boldsymbol{x}_\ell) \\ K(\boldsymbol{x}_2, \boldsymbol{x}_1) & K(\boldsymbol{x}_2, \boldsymbol{x}_2) & \cdots & K(\boldsymbol{x}_2, \boldsymbol{x}_\ell) \\ \vdots & \vdots & \ddots & \vdots \\ K(\boldsymbol{x}_\ell, \boldsymbol{x}_1) & K(\boldsymbol{x}_\ell, \boldsymbol{x}_2) & \cdots & K(\boldsymbol{x}_\ell, \boldsymbol{x}_\ell) \end{pmatrix}^{-1} \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_\ell \end{pmatrix} \\ &= \boldsymbol{K}(\boldsymbol{x}) G^{-1}_{K,X} \boldsymbol{y}.\end{aligned}$$

In the localized version, we use data-dependent $k(\leq \ell)$ training samples for $\hat{f}$. Let N_i $(i = 1, 2, \ldots, k)$ be the ith nearest neighbor of $\boldsymbol{x}$ in $\boldsymbol{X}$. By limiting the sample set from $\boldsymbol{X}$ to $\boldsymbol{X}_{kNN}(\boldsymbol{x}) = \{\boldsymbol{x}_{N_1}, \boldsymbol{x}_{N_2}, \ldots, \boldsymbol{x}_{N_k}\}$, we have

$$\begin{aligned}\hat{f}(\boldsymbol{x}) &= (K(\boldsymbol{x}, \boldsymbol{x}_{N_1}), \ldots, K(\boldsymbol{x}, \boldsymbol{x}_{N_k})) \begin{pmatrix} K(\boldsymbol{x}_{N_1}, \boldsymbol{x}_{N_1}) & \cdots & K(\boldsymbol{x}_{N_1}, \boldsymbol{x}_{N_k}) \\ \vdots & \ddots & \vdots \\ K(\boldsymbol{x}_{N_k}, \boldsymbol{x}_{N_1}) & \cdots & K(\boldsymbol{x}_{N_k}, \boldsymbol{x}_{N_k}) \end{pmatrix}^{-1} \begin{pmatrix} y_{N_1} \\ y_{N_2} \\ \vdots \\ y_{N_k} \end{pmatrix} \\ &= \boldsymbol{K}_{kNN}(\boldsymbol{x}) G^{-1}_{K,\boldsymbol{X}_{kNN}} \boldsymbol{y}_{kNN}\end{aligned}$$

This localization is clearly effective only for kernels in which the value of $K(\boldsymbol{x}, \boldsymbol{y})$ takes near zero when $\boldsymbol{x}$ and $\boldsymbol{y}$ are far from each other in the original feature space, such as a radial basis function.

It should be noted that $\hat{f}$ changes depending on a given data $\boldsymbol{x}$. A singularity of $G_{K,X}$ often becomes a problem when ℓ is large, but the reduced $G_{K,X_{kNN}}$ becomes non-singular in most cases. Of course, the most advantage of LPL is the calculation cost compared with the original PL. Note that we have to calculate the inverse of $G_{K,X_{kNN}}$ of size $k \times k$ for every data $\boldsymbol{x}$, although the inverse of $G_{K,X}$ is possible to be pre-calculated in the original PL. However, the cost is negligible because k is ignorably small to ℓ.

Even in the performance, we might expect a little raise compared with PL. Let us consider a trade-off of Eq. (5). By choosing the k nearest neighbors of a given data, the first (approximation) term is almost kept compared with the case that all samples are used, while the second time is necessary reduced because of its smaller size. As a result, a better trade off can be expected to be realized.

6 Comparison between SVM and LPL

Now let us compare SVM, PL and LPL in their criteria to minimize. For simplicity, we assume that every training samples are correctly classified with a positive margin. We also assume $\{1\} \subset \mathcal{H}_K$.

Then, SVM minimizes

$$J_{SVM}(\hat{f}) = \|\hat{f}\|^2_{\mathcal{H}_K}, \hat{f} \in \mathrm{span}\{K(\cdot, x_1), K(\cdot, x_2), \ldots, K(\cdot, x_\ell)\}$$

under the condition of $\sum_{i=1}^{N} |1 - y_i \hat{f}(x_i)|_+ = 0$. Eventually, only support vectors $S_1, \ldots, S_t$ are necessary for minimizing

$$J_{SVM}(\hat{f}) = \|\hat{f}\|^2_{\mathcal{H}_K}, \hat{f} \in \mathrm{span}\{K(\cdot, x_{S_1}), K(\cdot, x_{S_2}), \ldots, K(\cdot, x_{S_t})\}.$$

Similarly, PL minimizes

$$J_{PL}(\hat{f}) = ||f - \hat{f}||^2_{\mathcal{H}_K}, \hat{f} \in \mathrm{span}\{K(\cdot, x_1), K(\cdot, x_2), \ldots, K(\cdot, x_\ell)\}.$$

under the condition of $\sum_{i=1}^{N} |f(x_i) - \hat{f}(x_i)| = 0$, and LPL minimizes, with k nearest neighbors,

$$J_{LPL}(\hat{f}) = ||f - \hat{f}||^2_{\mathcal{H}_K}, \hat{f} \in \mathrm{span}\{K(\cdot, x_{N_1}), K(\cdot, x_{N_2}), \ldots, K(\cdot, x_{N_k})\}.$$

under the condition of $\sum_{i=1}^{k} |f(x_{N_i}) - \hat{f}(x_{N_i})| = 0$. These criteria are shown in Fig.1 and Fig.2. The following is worth noticing:

1. SVM and PL find the estimator $\hat{f}$ in the same subspace of $\mathcal{H}_K$ when a constant is ignored.
2. Nevertheless, SVM finds the minimum norm solution in $||\hat{f}||$ and PL finds the minimum difference solution in $||f - \hat{f}||$. They take reverse directions.

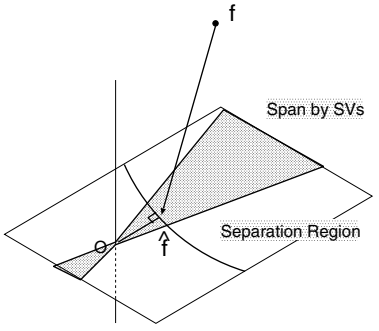


Fig. 1. Support Vector Machine

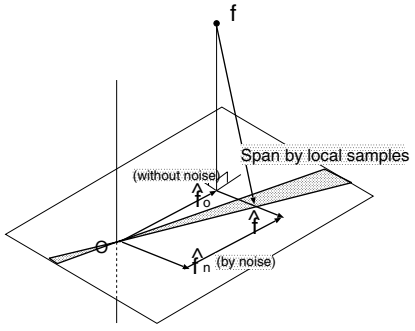


Fig. 2. Localized Projection Learning

3. LPL is expected to simulate PL well as long as k is large enough.
4. LPL has a high scalability because of the number of samples is limited to k.
5. Limiting the space by nearest neighbors in LPL works in a direction to worsen the approximation to the target but to increase the robustness against noise.
6. The necessary conditions are totally different. In SVM, the absolute value of $\hat{f}(\boldsymbol{x}_i)$ is not important as long as $y_i\hat{f}(\boldsymbol{x}_i) \geq 1$, while $\hat{f}(\boldsymbol{x}_i) = f(\boldsymbol{x}_i)$ has to be satisfied in PL. The latter is stronger than the former.

7 Approximate Nearest Neighbors

One advantage of LPL is to use k nearest neighbors for obtaining a scalability. However, when we want the exact k nearest neighbors, we need linear time both in dimensionality and in data size. Most sophisticated algorithms cannot beat this complexity in high dimensions. Therefore, for recent years, approximate nearest neighbors or probably correct nearest neighbors are gathering much attention [8,9]. In such a relaxation, the computational cost can be greatly reduced, usually sub-linear in sample size. Fortunately, in LPL, we do not necessary need the exact k nearest neighbors and suboptimal k nearest neighbors are acceptable. So, we can use such efficient techniques. We will use ANN [8] in this study. Its time complexity in search phase is $O(c_{p,\eta} \log \ell)$ with $c_{p,\eta} \leq p\lceil 1 + 6p/\eta \rceil^p$, where $\eta \geq 0$ is an approximation parameter.

8 Complexity

Time complexities of these algorithms are compared in Table 1. Usually the number t of support vectors is nearly proportional to the number ℓ of training samples. In Table 1, the number k of nearest neighbors in LPL is ignored because k is small enough compared with ℓ. The complexity of LPL comes from that of ANN to find k nearest neighbors. The other cost is quite low in LPL.

Table 1. Time complexity. Here, ℓ is the number of training samples, p is the dimensionality and t is the number of support vectors.

Phase	PL	LPL	SVM
Training	$O(\ell^2 p)$	$O(\ell p \log \ell)$	$O(\ell t p)$
Testing	$O(\ell p)$	$O(p \log \ell)$	$O(tp)$

Table 2. Statistics of datasets

Dataset	# samples (c_1, c_2)	Dim.
heart-statlog	270 (150, 120)	13
ionosphere	351 (225, 126)	34
sonar	208 (111, 97)	60
diabetes	768 (500, 268)	8
liver-disorders	345 (200, 145)	6
spambase	4601 (2788, 1813)	57

9 Experiments

We carried out experiments using one synthetic dataset and six real-life two-class datasets taken from UCI machine learning repository [10] (Table 2). The synthetic dataset is of two classes in 2-dimensional space (Fig. 3).

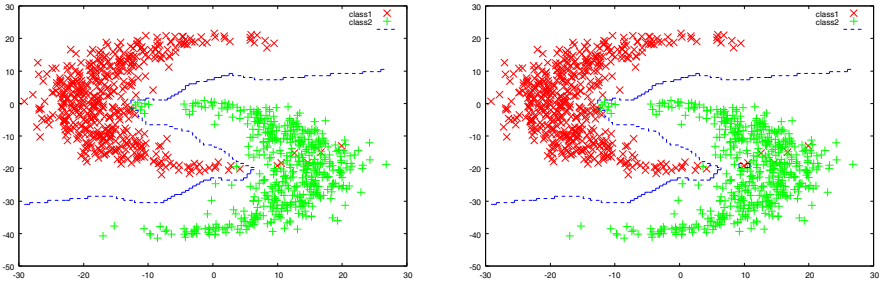
We compared PL, LPL and SVM. The SVM is the one implemented by **libsvm** [11] with a soft margin parameter $C = 1$. The kernel function is a Gaussian with a standard deviation σ. The number of nearest neighbors k in LPL was chosen to $k = \lfloor \log_{10} \ell + 1 \rfloor$ in order to simulate a consistent nature of k nearest neighbors, that is, k-NN approaches to Bayes classifier with $k = o(n)$. In ANN, we have used $\eta = 0.0$, that is, we found the exact nearest neighbors. Even if $\eta = 0.0$, the computational cost is known to be sublinear in ℓ for a low-dimensional space. The recognition rate was evaluated by 10-fold cross validation technique.

9.1 Comparison of PL and LPL

First we compared PL and LPL in performance and in time. The result for the synthetic dataset showed that the performance is almost the same in range $\sigma \in [0.01, 10]$. Indeed, there is almost no difference between the decision boundary of PL and that of LPL (Fig. 3). As for the time, LPL was faster than PL in both of training and testing phases (Fig. 4). This is consistent with the theoretical analysis in Table 1. Even in real-life datasets, this tendency was observed. As seen in Fig. 5, the recognition rates of LPL were comparative with those of PL, while the time for testing was greatly reduced as described later. Therefore, we will mainly compare LPL and SVM in the following.

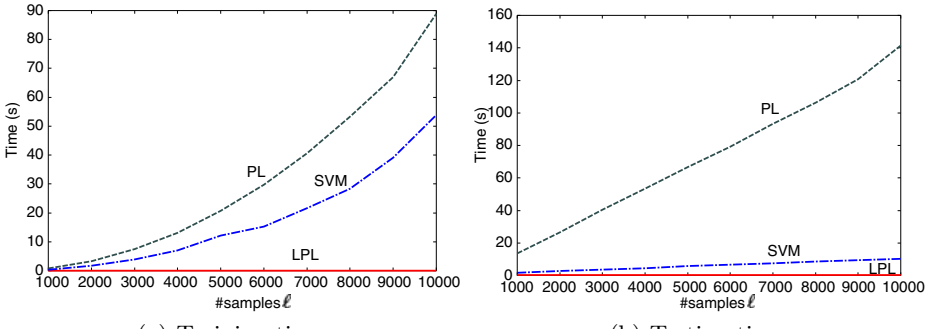
9.2 Comparison of LPL and SVM

In comparison between LPL and SVM, the difference of time complexities appeared clearly in the synthetic data (Fig. 4). LPL is the fastest among three algorithms in both of training and testing phases. Even for **spambase**, which is the largest in sample size, LPL was remarkable faster than SVM. Indeed, the training time and testing time were 0.019 and 0.027 seconds, respectively, in LPL, while 6.461 and 0.632 seconds, respectively, in SVM. For the other datasets, an obvious difference was not observed because of the short time.



(a)PL (b)LPL

Fig. 3. Decision boundaries of PL and LPL ($\sigma = 1.0$)



(a) Training time (b) Testing time

Fig. 4. Time consumed by PL and LPL on a synthetic dataset. The curve of LPL is almost identical to the horizontal axis.

In performance, we have to be careful about the values of parameters, especially σ for Gaussian kernel. We have compared LPL and SVM in the same $\sigma = 1.0$ for time comparison. However, the optimal value of σ should be different in LPL and SVM. Therefore, we chose the best value σ^* in the range of $\sigma \in [0.01, 100]$ at 39 values in a log scale.

As shown in Fig. 5, with their optimal values, SVM was superior to LPL in most cases. This maybe implies that the separation criterion (largest margin criterion) employed in SVM is better than the approximation criterion (closest criterion) for classification problems.

9.3 Robustness

Although SVM was better than LPL with the optimal σ^*, such an optimal setting is time consuming and is sometimes even difficult to be made. So we examined how sensitive LPL and SVM are against the change of σ. In Fig. 6, a result is shown for **sonar** dataset. We can see that SVM shows a high performance only in a narrow range of σ and that LPL has a larger range for it and PL follows. The difference of robustness is also confirmed from a comparison of two cases of

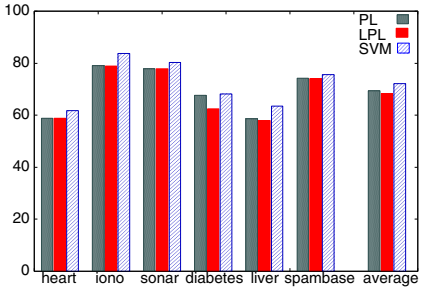


Fig. 5. Recognition rate of PL, LPL and SVM with optimal σ^*

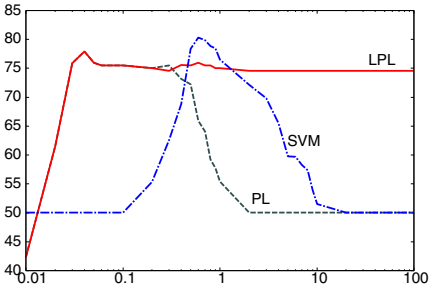
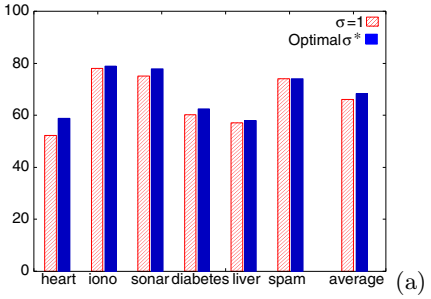
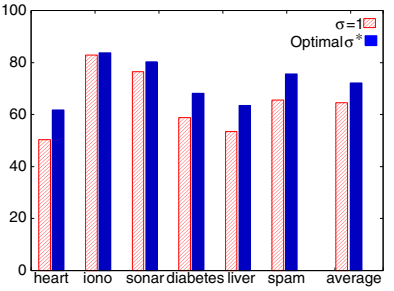


Fig. 6. Comparison of recognition rates for $\sigma \in [0.01, 100]$ on **sonar**



(a) LPL



(b) SVM

Fig. 7. Recognition rates for optimal σ^* and $\sigma = 1$

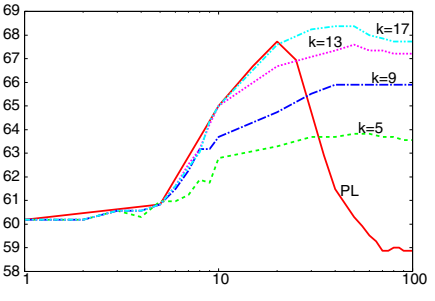


Fig. 8. Recognition rate of LPL for $k = 5, 9, 13, 17$ and $\sigma \in [1, 100]$ on **diabetes**

$\sigma = 1$ and $\sigma = \sigma^*$. In Fig. 7, we can see that the difference of LPL is smaller than that of SVM. It means that SVM is more sensitive than LPL.

The value of parameter k is also important in LPL, because the value determines the samples that affect the decision of a given sample. In **deabetes**, in which LPL was worse than SVM in performance largely, we have changed the value of k in $k = 5, 9, 13, 17$ (our default setting was $k = 3$) (Fig.8). From Fig.8, we can see that a better performance of LPL can be obtained with larger values

of k. With $k = 17$, the performance of LPL is comparable with that of SVM. As a result, the optimal choice of k would improve the performance of LPL.

10 Conclusion

In this paper, scalability was brought to projection learning (PL) by localizing it with nearest neighbors of a given sample to be classified. First, it was confirmed that the classification performance is almost maintained by this localization and that the testing speed is rather improved, as it is sublinear in the number of training samples. In comparison of performance between SVM and LPL (localized PL), SVM was a little superior to LPL with the optimally chosen parameters. This might imply that the criterion of SVM is better than that of LPL for classification problems. However, we need more investigation. On the contrary, it was revealed that LPL is faster than SVM and is more robust than SVM against the parameter change. One attractive point of LPL is that it allows us to use approximate nearest neighbors so that we can use efficient algorithms even in high-dimensional problems.

One of the future studies is to compare the performance in multi-class problems. In that case, several ways of giving dummy quantitative variables have to be considered.

References

1. Vapnik, V.N.: The Nature of Statistical Learning Theory. Springer, New York (1999)
2. Ogawa, H.: Neural Networks and Generalization Ability. IEICE Technical Report NC95-8, 57–64 (1995)
3. Aronszajn, N.: Theory of Reproducing Kernels. Transactions of the American Mathematical Society 68, 337–404 (1950)
4. Mercer, J.: Functions of Positive and Negative Type and Their Connection with The Theory of Integral Equations. Transactions of the London Philosophical Society A, 415–446 (1909)
5. Schatten, R.: Norm Ideals of Completely Continuous Operators. Springer, Berlin (1960)
6. Tanaka, A., Imai, H., Kudo, M., Miyakoshi, M.: Optimal Kernel in a Class of Kernels with an Invariant Metric. In: da Vitoria Lobo, N., Kasparis, T., Roli, F., Kwok, J.T., Georgiopoulos, M., Anagnostopoulos, G.C., Loog, M. (eds.) S+SSPR 2008. LNCS, vol. 5342, pp. 530–539. Springer, Heidelberg (2008)
7. Tanaka, A., Imai, H., Kudo, M., Miyakoshi, M.: Relationship Between Generalization Error and Training Samples in Kernel Regressors. To appear in ICPR 2010 (2010)
8. Arya, S., et al.: An optimal algorithm for approximate nearest neighbor searching fixed dimensions. Journal of the ACM 45-6, 891–923 (1998), `http://www.cs.umd.edu/~mount/ANN/`
9. Indyk, P., Motwani, R.: Approximate nearest neighbors: towards removing the curse of dimensionality. In: Proceedings of the 30th Annual ACM Symposium on Theory of Computing, pp. 604–613 (1998)
10. Asuncion, A., Newman, D.: UCI machine learning repository (2007)
11. Chang, C.C., Lin, C.J.: LIBSVM: a library for support vector machines, `http://www.csie.ntu.edu.tw/~cjlin/libsvm/`