Localized Projection Learning

Kazuki Tsuji, Mineuchi 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). 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.

The SVMs are formulated for two-class problems with $y = \pm 1$ [1]. An SVM aims to attain the maximal margin between two class sets of samples. Especially, in a "hard margin SVM", as will be shown, $\hat{f}$ is taken in a space $\mathrm{span}\{K(\cdot, x_1), K(\cdot, x_2), \ldots, K(\cdot, x_\ell)\} + \mathrm{span}\{1\}$, where $K(\cdot, \cdot)$ is a "kernel" and $\mathrm{span}\{\cdot\}$ is the closure of linear combinations of the elements.

Projection learning is a more straightforward way to find $\hat{f}$, in which the squared distances in the training sample pairs are minimized. The $\hat{f}$ is taken from a space $\mathrm{span}\{K(\cdot, x_1), K(\cdot, x_2), \ldots, K(\cdot, x_\ell)\}$ so as to realize the orthogonal projection of f into this space and to find the closest approximation in the norm.

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 localized projection learning, to compare the projection learning and the SVM in classification performance.

2 Reproducing Kernel Hilbert Spaces

Reproducing kernel Hilbert spaces [2, 3] is the key concept to interpret above approaches in a unified framework. In this section, we state some mathematical definitions concerned with it.

Definition 1. *[2] 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. The function $K(\boldsymbol{x}, \tilde{\boldsymbol{x}}),\ (\boldsymbol{x}, \tilde{\boldsymbol{x}} \in \mathcal{D})$ is called a reproducing kernel of $\mathcal{H}$, 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 $f \in \mathcal{H}$,*

$$f(\tilde{\boldsymbol{x}}) = \langle f(\boldsymbol{x}), K(\boldsymbol{x}, \tilde{\boldsymbol{x}}) \rangle_{\mathcal{H}}, \tag{1}$$

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 Eq.(1) enables us to treat a value of a function at a point in $\mathcal{D}$. Note that reproducing kernels are non-negative definite [2]:

$$\sum_{i,j=1}^{N} c_i c_j K(\boldsymbol{x}_i, \boldsymbol{x}_j) \geq 0, \tag{2}$$

for any N, $c_1, \ldots, c_N \in \mathbf{R}$, and $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N \in \mathcal{D}$. In addition, $K(\boldsymbol{x}, \tilde{\boldsymbol{x}}) = K(\tilde{\boldsymbol{x}}, \boldsymbol{x})$ for any $\boldsymbol{x}, \tilde{\boldsymbol{x}} \in \mathcal{D}$ is followed [2]. If a reproducing kernel $K(\boldsymbol{x}, \tilde{\boldsymbol{x}})$ exists, it is unique, and, conversely, every positive definite function $K(\boldsymbol{x}, \tilde{\boldsymbol{x}})$ has the unique corresponding RKHS [2].

Next, we introduce the Schatten product [4] that is a convenient tool to reveal the reproducing property of kernels.

Definition 2. *[4] 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*

$$(g \otimes h) f = \langle f, h \rangle_{\mathcal{H}_1} g, \quad f \in \mathcal{H}_1. \tag{3}$$

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 $h, v \in \mathcal{H}_1$, $g, u \in \mathcal{H}_2$.

$$(h \otimes g)^* = (g \otimes h), \quad (h \otimes g)(u \otimes v) = \langle u, g \rangle_{\mathcal{H}_2} (h \otimes v), \tag{4}$$

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 with $y_i \in \mathbf{R}$, $\boldsymbol{x}_i \in \mathbf{R}^p$, satisfying

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

where f denotes the unknown true function and n_i denotes a zero-mean additive noise. The aim of machine learning is to estimate the unknown function f by using the given training data set and statistical properties of noise.

In this paper, we assume that the unknown function f belongs to the RKHS $\mathcal{H}_K$ corresponding to a certain kernel function K. If $f \in \mathcal{H}_K$, then Eq.(3) is rewritten as

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

on the basis of the reproducing property (1) of kernels. 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 to Eq.(3) yields

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

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 $X = \{x_1, \ldots, x_\ell\}$, 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.(3) can be simply written by

$$\boldsymbol{y} = A_{K,X} f + \boldsymbol{n}. \tag{5}$$

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

Projection learning is derived to attain the minimum squared error on X between the target function f and an estimator $\hat{f}$ measured in $\mathcal{H}$ in the case of zero-mean noise. Such an optimal estimator is given by solving (5) as

$$\begin{aligned} \hat{f}(\cdot) &= A_{K,X}^{+} \boldsymbol{y} && (6)\\ &= A_{K,X}^{*} (A_{K,X} A_{K,X}^{*})^{+} \boldsymbol{y} && (7)\\ &= \sum_{k=1}^{\ell} \boldsymbol{y}' G_{K,X}^{+} \boldsymbol{e}_k^{(\ell)} K(\cdot, \boldsymbol{x}_k), && (8) \end{aligned}$$

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.(7) 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

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

It is also easy to show that $\hat{f}$ is equivalent to f on X, that is, with $\boldsymbol{f} = [f(\boldsymbol{x}_1), \ldots, f(\boldsymbol{x}_\ell)]'$,

$$\hat{\boldsymbol{f}} \equiv \boldsymbol{f}, \tag{9}$$

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.* [7, 8]. Roughly speaking, as the number of training samples increases, the approximation error $||f - P_{R(A^*_{K,X})} f||^2_{\mathcal{H}_K}$ is reduced but the error caused by noise $||P_{R(A^*_{K,X})} f - A^*_{K,X}\boldsymbol{n}||^2_{\mathcal{H}_K}$ is increased, so that $||f - \hat{f}||^2_{\mathcal{H}_K}$ is unknown on whether reduced or increased. When noise exists, from (5) and (6), our estimate becomes

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

In [7], $||P_{R(A^*_{K,X})} f||^2_{\mathcal{H}_K} = \boldsymbol{f}' G^+_{K,X} \boldsymbol{f}$ and, in [8], $||A^+_{K,X}\boldsymbol{n}||^2_{\mathcal{H}_K} = \boldsymbol{n}' G^+_{K,X}\boldsymbol{n}$ were shown. Thus, we have the following relationship:

$$\begin{aligned} ||f - \hat{f}||^2_{\mathcal{H}_K} &= ||f - P_{R(A^*_{K,X})} f||^2_{\mathcal{H}_K} + ||A^+_{K,X}\boldsymbol{n}||^2_{\mathcal{H}_K} \\ &= ||f||^2_{\mathcal{H}_K} - \boldsymbol{f}' G^+_{K,X}\boldsymbol{f} + \boldsymbol{n}' G^+_{K,X}\boldsymbol{n}. \end{aligned} \tag{10}$$

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

$$J_{SVM}(\hat{f}) = \frac{1}{\ell}\sum_{i=1}^{\ell} |1 - y_i \hat{f}(x_i)|_+ + \lambda \|\hat{h}\|^2_{\mathcal{H}_K}, \tag{11}$$

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 $h \in \mathcal{H}$ and c is a constant and $P\hat{f} = \hat{h}$. Therefore, if $\{1\} \subset \mathcal{H}_K$ then this equation is rewritten as

$$J_{SVM}(\hat{f}) = \frac{1}{n}\sum_{i=1}^{N} |1 - y_i \hat{f}(x_i)|_+ + \lambda \|P\hat{f}\|^2_{\mathcal{H}_K}, \tag{12}$$

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

$$\hat{f}(\cdot) = \sum_{i=1}^{c} c_i K(\cdot, x_i) + c_0 \tag{13}$$

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 (8) can be expressed explicitly as

$$\begin{aligned}\hat{f}(\boldsymbol{x}) &= (K(\boldsymbol{x}, \boldsymbol{x}_1), \ldots, K(\boldsymbol{x}, \boldsymbol{x}_\ell)) \begin{pmatrix} K(\boldsymbol{x}_1, \boldsymbol{x}_1) & K(\boldsymbol{x}_1, \boldsymbol{x}_2) & \cdots & K(\boldsymbol{x}_1, \boldsymbol{x}_\ell) \\ K(\boldsymbol{x}_2, \boldsymbol{x}_1) & K(\boldsymbol{x}_2, \boldsymbol{x}_2) & \cdots & K(\boldsymbol{x}_2, \boldsymbol{x}_\ell) \\ \vdots & \vdots & \ddots & \vdots \\ K(\boldsymbol{x}_\ell, \boldsymbol{x}_1) & K(\boldsymbol{x}_\ell, \boldsymbol{x}_2) & \cdots & K(\boldsymbol{x}_\ell, \boldsymbol{x}_\ell) \end{pmatrix}^{-1} \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_\ell \end{pmatrix} \\ &= \boldsymbol{K}(\boldsymbol{x}) G_{K,X}^{-1} \boldsymbol{y} \end{aligned} \tag{14}$$

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 in X to $\boldsymbol{x}$. By limiting the sample set from X to $X_{kNN}(\boldsymbol{x}) = \{\boldsymbol{x}_{N_1}, \boldsymbol{x}_{N_2}, \ldots, \boldsymbol{x}_{N_k}\}$, we have

$$\begin{aligned}\hat{f}(\boldsymbol{x}) &= (K(\boldsymbol{x}, \boldsymbol{x}_{N_1}), \ldots, K(\boldsymbol{x}, \boldsymbol{x}_{N_k})) \begin{pmatrix} K(\boldsymbol{x}_{N_1}, \boldsymbol{x}_{N_1}) & \cdots & K(\boldsymbol{x}_{N_1}, \boldsymbol{x}_{N_k}) \\ \vdots & \ddots & \vdots \\ K(\boldsymbol{x}_{N_k}, \boldsymbol{x}_{N_1}) & \cdots & K(\boldsymbol{x}_{N_k}, \boldsymbol{x}_{N_k}) \end{pmatrix}^{-1} \begin{pmatrix} y_{N_1} \\ y_{N_2} \\ \vdots \\ y_{N_k} \end{pmatrix} \\ &= \boldsymbol{K}_{kNN}(\boldsymbol{x}) G_{K,X_{kNN}}^{-1} \boldsymbol{y}_{kNN} \end{aligned} \tag{15}$$

This localization approach is 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 $G_{K,X_{kNN}}$ 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 relatively small to ℓ.

Even in the performance, we can expect a little raise compared with PL. Let us consider a trade-off of Eq. (10). By choosing the samples to be used to the k nerarest neighbors, the first (approximation) term is almost kept while the second time is necessary reduced. As a result, a better tradeoff 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

$$J_{SVM}(\hat{f}) = \|\hat{f}\|^2_{\mathcal{H}_K}, \hat{f} \in \mathrm{span}\{K(\cdot, x_1), K(\cdot, x_2), \ldots, K(\cdot, x_\ell)\} \tag{16}$$

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

$$J_{SVM}(\hat{f}) = \|\hat{f}\|^2_{\mathcal{H}_K}, \hat{f} \in \mathrm{span}\{K(\cdot, x_{S_1}), K(\cdot, x_{S_2}), \ldots, K(\cdot, x_{S_t})\}. \tag{17}$$

Similarly, PL minimizes

$$J_{PL}(\hat{f}) = ||f - \hat{f}||^2_{\mathcal{H}_K}, \hat{f} \in \mathrm{span}\{K(\cdot, x_1), K(\cdot, x_2), \ldots, K(\cdot, x_\ell)\}. \tag{18}$$

under the condition of $\sum_{i=1}^{N} |f(x_i) - \hat{f}(x_i)| = 0$ (see (9)), and LPL minimizes, with k nearest neighbors,

$$J_{LPL}(\hat{f}) = ||f - \hat{f}||^2_{\mathcal{H}_K}, \hat{f} \in \mathrm{span}\{K(\cdot, x_{N_1}), K(\cdot, x_{N_2}), \ldots, K(\cdot, x_{N_k}\}. \tag{19}$$

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.
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 actual samples is limited to k.
5. Limiting the space, by support vectors in SVM and by nearest neighbors in LPL, work in a direction to lengthen the norm of the solution and to make worse the approximation to the target.
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.

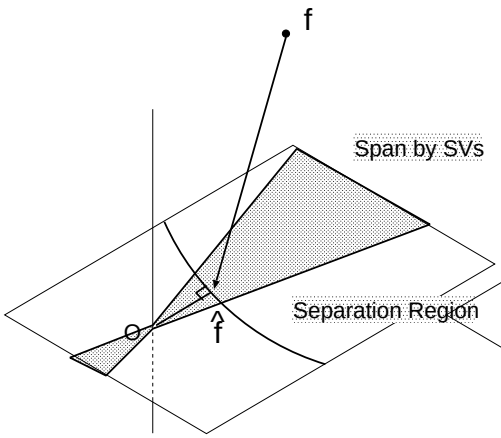

Fig. 1. Support Vector Machine

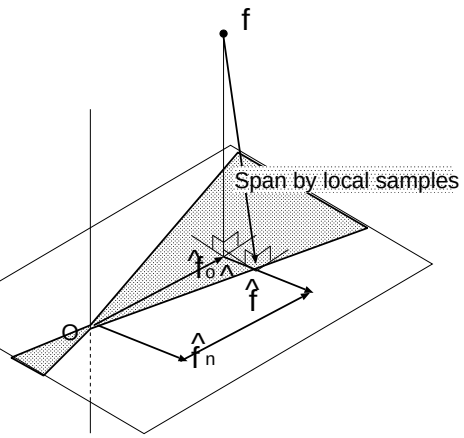

Fig. 2. Localized Projection Learning

7 Approximate Nearest Neighbors

One advantage of LPL is to use k nearest neighbors for preserving a scalability. However, when we want the exact k nearest neighbors, we cannot avoid "the curse of dimensionality", that is, we need linear time both in dimensionality and data size. Most sophisticated algorithms cannot beat this curse. Therefore, for recent years, approximate nearest neighbors or probably correct nearest neighbors are gathering much interests [9, 10]. 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 [9] in this study. Its time compexity 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. Once they are found, the other cost is quite low. However, to better choice, we can take it as small but one increasing with ℓ, for example, $k = \log \ell$. Even so, the time complexity is still less than those of the other algorithms.

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	$O(\ell p)$	$O(p \log \ell)$	$O(tp)$

9 Experiments

k nearest neighbors is implemented by ANN.SVM is implemented by Libsvm. SVM's parameter other than σ is default of Libsvm($C = 1$). k nearest neighbors's parameter k is $\log_{10} \ell + 1$.ℓ is number of sample.

9.1 Comparison of computational time

We show the results of the recognition rate by experiment using artificial data in Table 2. Graph of computational time is shown in figure 3 and 4. Used data set is two class and two dimension. Training data set increases from 1000 to 10000. kernel is Gaussian kernel and parameter is $\sigma = 1$. Because recognition rate is good on average, we used $\sigma = 1$.

Table 2. Computational time in three algorithms. In below, Tr is the training time and Te is the testing time in seconds. #SV is the number of support vectors.

#samples ℓ	PL		LPL ($k = \log_1 0\ell + 1$)		SVM		
	Tr	Te	Tr	Te	Tr	Te	#SV
1000	0.84	13.50	0.00	0.26	0.35	1.66	801
2000	3.28	26.69	0.00	0.27	1.71	2.64	1360
3000	7.50	40.39	0.00	0.28	3.94	3.61	1833
4000	13.11	53.49	0.01	0.29	6.95	4.49	2307
5000	20.69	66.75	0.00	0.26	12.12	5.71	2886
6000	29.70	79.13	0.01	0.31	15.22	6.63	3372
7000	40.66	93.38	0.01	0.28	21.68	7.47	3842
8000	53.21	106.39	0.01	0.25	28.40	8.48	4379
9000	66.91	120.71	0.01	0.30	39.15	9.40	4871
10000	88.91	141.4	0.01	0.38	53.90	10.29	5277

In the beginning, we compare between PL and LPL. In Figure3 and 4, LPL is very faster than PL.

9.2 Comparison of recognition rate

Next, we show the results of the recognition rate by six data sets(Table 3). At $\sigma = 1$, result is showed in Table5 and Figure7. At optimal σ, result is showed in Table6 and Figure8.

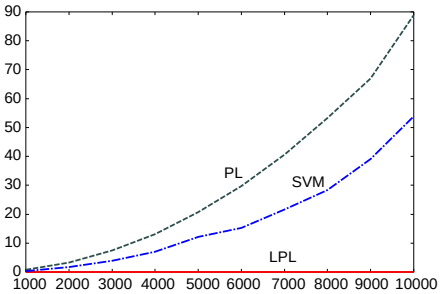

Fig. 3. Comparison of training time.

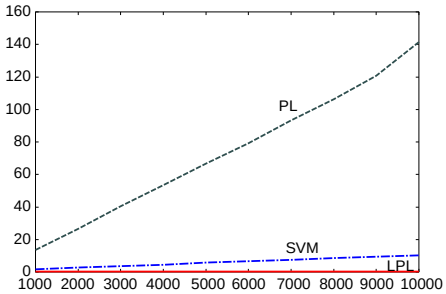

Fig. 4. Comparison of testing time.

Fig. 5. Recognition rate for $\sigma = 1$.

Data set	Algorithm		
	PL	LPL	SVM
heart-statlog	52.22	52.22	50.37
ionosphere	69.71	78	82.86
sonar	55.38	75.05	76.52
diabetes	60.18	60.18	58.74
liver-disorders	56.87	57.15	53.38
spambase	74.24	74.11	65.52
Average	61.43	66.12	64.57

Fig. 6. Recognition rate for the optimum parameter value. A value of σ^* is the optimally chosen value.

Data set	Algorithm					
	PL	σ^*	LPL	σ^*	SVM	σ^*
heart-statlog	58.89	15	58.89	25	61.85	35
ionosphere	79.14	0.3	78.86	0.4	83.71	5
sonar	77.88	0.04	77.88	0.04	80.31	0.6
diabetes	67.72	20	62.39	25	68.25	95
liver-disorders	58.62	5	58.03	0.5	63.56	80
spambase	74.24	1	74.11	1	75.65	10
Average	69.42	-	68.36	-	72.22	-

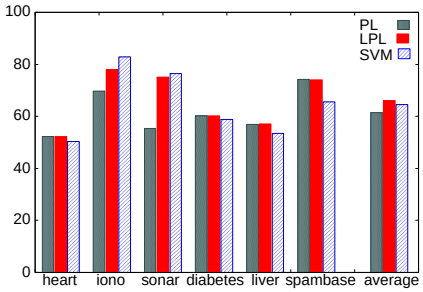

Fig. 7. recognition rate by $\sigma = 1$

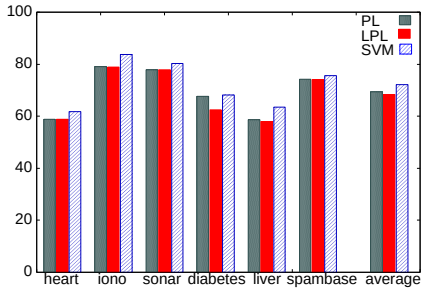

Fig. 8. recognition rate by optimum parameter

Table 3. Data set

Dataset	# samples (Class 1, Class 2)	Dimension
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

10 Discussion

Our main goal was to make clear the difference between SVM and PL in classification problems. Roughly speaking, SVM is better than PL, because a better approximation of an unknown target function does not contribute directly for better design of classifiers, when regression is used for intermediate with dummy variables. However, we noticed that PL is more robust than SVM in the setting of parameter values. This is because PL minimizes the average error but SVM minimizes the error of samples near the decision boundary. When the Bayes error is not zero, SVM is sensitive to those samples.

In either algorithms, existence of noise causes a big problem. In $y = \pm 1$ setting, a mis-labeled error produces noise $n = \pm 2$ to change the sign. Once such an error happens, both algorithms are strongly affected when training samples are not sufficient. Especially such an error more happens near the Bayes decision boundary. Then the setting of soft margin parameter and the other parameter related to chosen kernel becomes more harder than ever in SVM. On the other hand, in the case of a large sample set, LPL is expected to work better because of its robustness.

11 Conclusion

In this paper, scalability was brought to projection learning (PL) by localizing it, LPL, by 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 now sublinear in the number of training samples. In comparison of performance between SVM and LPL, SVM was a little superior to LPL. This is mainly because the criterion of SVM is better than that of LPL for classification problems. As the same time, however, it was also revealed that LPL is better than SVM in the easiness of parameter setting. One more attractive point is that LPL allows us to use approximate nearest neighbors so that we can use find nearest neighbors very efficiently even in high-dimensional problems.

One of the future study 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. Aronszajn, N.: Theory of Reproducing Kernels. Transactions of the American Mathematical Society **68** (1950) 337–404
3. Mercer, J.: Functions of Positive and Negative Type and Their Connection with The Theory of Integral Equations. Transactions of the London Philosophical Society **A** (1909) 415–446
4. Schatten, R.: Norm Ideals of Completely Continuous Operators. Springer-Verlag, Berlin (1960)
5. Ogawa, H.: Neural Networks and Generalization Ability. IEICE Technical Report **NC95-8** (1995) 57–64
6. Rao, C.R., Mitra, S.K.: Generalized Inverse of Matrices and its Applications. John Wiley & Sons (1971)
7. Tanaka, A., Imai, H., Kudo, M., Miyakoshi, M.: Optimal Kernel in a Class of Kernels with an Invariant Metric. In: Joint IAPR Internatioanl Workshops S+SSPR 2008, Lecture Notes in Computer Science, **5342**, 2008, 530–539.
8. Tanaka, A. , Imai, H., Kudo, M., Miyakoshi, M.: Relationship Between Generarization Error and Training Samples in Kernel Regressors. Proc. of 19th International Conference on Pattern Recognition (ICPR 2008), submitted.
9. Arya, S. *et al.*: An optimal algorithm for approximate nearest neighbor searching fixed dimensions. Journal of the ACM, 45-6(1998), pp. 891–923. URL:http://www.cs.umd.edu/ mount/ANN/.
10. Indyk, P. and Motwani, R.: Approximate nearest neighbors: towards removing the curse of dimensionality. Proceedings of the 30th Annual ACM Symposium on Theory of Computing, pp. 604–613, 1998.