Available online at
http://www.ibnusina.utm.my/jfs

Journal of Fundamental Sciences

Article

Accuracy test in identifying the splice site type of DNA sequences by using probabilistic neural networks and support vector machines

Djati Kerami*

Mathematics Department, Faculty of Mathematics and Science University of Indonesia, Depok – 16424, Indonesia
**To whom correspondence should be addressed. E-mail: djatikr@makara.cso.ui.ac.id*

Received 10 November 2005
http://dx.doi.org/10.11113/mjfas.v1n1.10

ABSTRACT

It has been known that Probabilistic Neural Networks as machine learning is very fast in it's computation time and give a better accuracy comparing to another type of neural networks, on solving a real-world application problem. In the recent years, Support Vector Machines has become a popular model over other machine learning. It can be analyzed theoretically and can achieve a good performance at same time. This paper will describe the use of those machines learning to solve pattern recognition problems with a preliminary case study in detecting the type of splice site on the DNA sequences, particularity on the accuracy level. The results obtained show that Support Vector Machines have a good accuracy level about 95 % comparing to Probabilistic Neural Networks with 92 % approximately.

| Probabilistic Neural Networks | Support Vector Machines | Splice sites type detection | Accuracy level |

1. Introduction

Neural Networks (NNs) and Support Vector Machines (SVMs) both are machine learning techniques that learn a model or pattern based on training data, and use the model to predict or to classify on future data. Both two machine learning models can be regarded as a tool in soft computing approach. By a soft computing technique, we try to understand a system behavior in exchange for unnecessary precision for practical problem. In addition, there are no superposition are made about preexisting analytical model. This technique is rather different with hard computing approach which is always require a precisely stated analytical model.

Among several model type of NNs, the Probabilistic Neural Networks (PNNs) introduced by Specht [1], has been known for speed advantage and the accuracy level comparing to other type of NNs. SVMs are even more recent technology than NNs [4]. In about 1979 Vladimir Vapnik has had developed the theoretical foundation for SVMs based on statistical learning approach [3,5]. After this development, SVMs are now widely used and studied as a machine learning model [6,7,8].

In this paper, those two machines learning above will be used in the domain of pattern recognition problems, in particularity, splice site type detection on the DNA sequences. This present paper is focused on the accuracy level (or generalization capability) which is regarded as one of machine performance measurement.

2. Theoretical Background

2.1. Learning To Classify

Let us start with a general notion of the learning problem that we consider in this paper. The task of classification is to find a rule, which based on external observation, assign an object to one of several classes. In the simplest case there are only two different classes. Possible formalization of this task is to estimate a function f : $R^n \rightarrow \{1,-1\}$, using input-output data pairs generated independent identically distributed according to an unknown probability distribution function P(x,y),

$$X \text{ x } Y = \{(x_1,y_1), \ldots\ldots, (x_m,y_m)\}, \text{ with every } x_i \in R^n \text{ and } y_i \in \{1,-1\}, \text{ i} = 1, \ldots, \text{m}.$$

In the training stage, we estimate *f*, minimizing the expected error (risk)

$$E(f) = \int L(f(x), y)dP(x, y)$$

where L denotes a suitably chosen loss function.

The task of classification above is done by machine learning, is this paper we use PNNs and SVMs.

2.1.1. Probabilistic Neural Networks

PNNs is a one type of Neural Networks model using in supervised learning. This model is widely known in pattern recognition problems. The initial model was introduced by Specht [1], applying estimated Parzen non parametric probability function and Bayes classification rules. The architecture introduced consist of input layer, pattern layer, summation layer, and output layer (Figure 1).

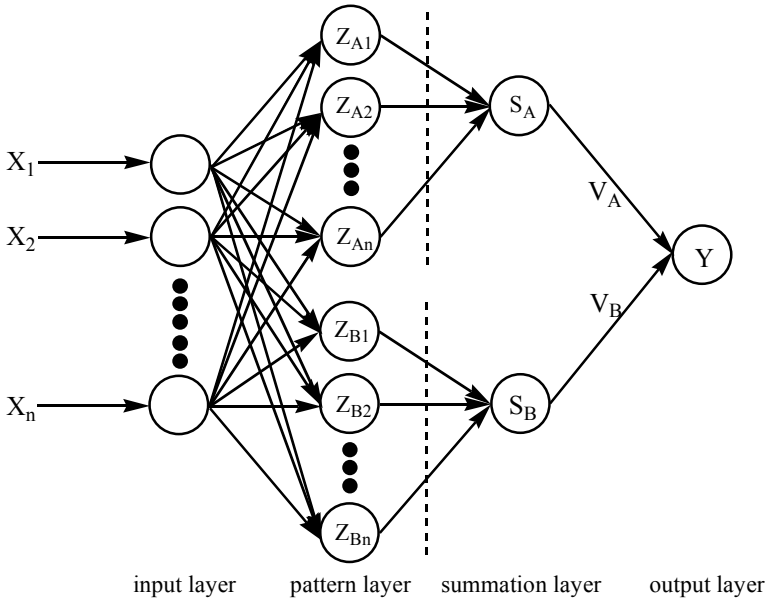

Fig. 1 PNNs architecture, with input layer, pattern layer, summation layer, and output layer.

In estimation process, PNNs use two stages, learning stage and estimation stage [2]. In learning stage, input neurons (in input layer) distribute input vector according their weight and send it to pattern neurons (in pattern layer). Then, patterns neurons transmit it to summation neuron (in summation layer) for summation purpose. In estimation stage, this summation results are connected with the decision, to verify whether the input vector is a member of any class given or not.

Suppose that the PNNs consist of two input neurons (Figure 1), four pattern neurons, and two summation neurons. Input vector x through input neurons, is sent to pattern neurons Z_{A1}, Z_{A2}, Z_{B1}, and Z_{B2}. This input to pattern neurons is denoted by $z\_in_j = x^T.w$, j=1,…,4, where w is weighted vector obtained at the learning stage. The output from pattern neurons is z_out_j which will be obtained by using activation function f as follows

$$z\_out_j = f(z\_in_j) = \exp\left(\frac{z\_in_j - 1}{\sigma^2}\right) \quad (1)$$

z_out_j will be an input for corresponding summation neuron. Hence, input to summation neuron S_A is $s\_in_A = z\_out_{A1} + z\_out_{A2}$, and input to summation neurons S_B is $s\_in_B = z\_out_{B1} + z\_out_{B2}$. Here, the connection weight between neurons input and summation neurons is 1, between summation neuron S_A and output neurons is v_A, and between summation neuron S_B and output neurons is v_B. In this case

$$v_A = \frac{\lambda_{BA} h_A}{m_A} \text{ dan } v_B = \frac{\lambda_{AB} h_B}{m_B}$$

where, m_A is the number of training data in class A, m_B is number of training data in class B, $m = m_A + m_A$ is total training data. Hence, $h_A = \frac{m_A}{m}$ is probability (a priori) of input data in class A. In the same manner, h_B is probability (a priori) of input data in class B. While λ_{AB} is a cost function to represent the fact that it may be misclassify input data is in class A instead of in class B. Similarly with λ_{BA}.
For the classification into two classes, clearly that $\lambda_{AB} = \lambda_{BA} = 1$.

Finally, input to neuron output Y represented by y_in is $\{v_A.S_{A\_out}, v_B.S_{B\_out}\}$. By using a decision criteria will be decided whether input vector is in class A or in class B. Here, $x \in$ class A if $v_A.S_{A\_out} > v_B.S_{B\_out}$. Otherwise, $x \in$ class B. That criteria is called Bayes criteria, that can be stated by

$$f(y\_in) = \begin{cases} 1 \ (x \in A), if \ v_A.S_{A\_out} > v_B.S_{B\_out} \\ -1 (x \in B), if \ v_A.S_{A\_out} \le v_B.S_{B\_out} \end{cases}$$

With further manipulation, we can state that if $x \in$ class A, then $\lambda_{BA} h_A \hat{f}_A(x) > \lambda_{AB} h_B \hat{f}_B(x)$, where $\hat{f}_A(x)$ is approximated probability density function (obtained in previous stage). Similarly, if $x \in$ class B.

2.1.2. Support Vector Machine

As a machine learning model, SVMs is a recently developed which designed for efficient multidimensional function approximation. The basic idea is to determine a classifier minimizing the empirical risk (that is, training set error) which corresponds to the generalization or test set error [5,6,7,8]. The key to understanding SVMs is to see how they introduce an optimum hyperplane to separate classes of data in the classifiers.

Given an input vector $X = \{x_1, x_2, \ldots, x_m\}$, each $x_i \in R^n$ and a target vector $Y = \{y_1, y_2, \ldots, y_m\}$, each $y_i \in \{1,-1\}$, i=1,2, .., m. Let us consider $(x_i, y_i) \in X \times Y$ as a pair of training data which is separable.

The main idea of SVMs is to determine a hyperplane <w, x> + b = 0 ($w \in R^n$, $b \in R$) separating X into two classes associated with the value of 1 or –1, with a maximum margin. In this case, the margin is a distance between two parallel hyperplanes with the middle hyperplane (Figure 1). This middle hyperplane will be considered as a *decision function* for two class classification,

$$f(x) = \text{sign}(\langle w, x\rangle + b)$$

where f(x) = 1 if $\langle w, x\rangle + b \geq 0$ and f(x) = -1 if $\langle w, x\rangle + b < 0$.

Based on Vapnik-Chervonenkis (VC) Theorem [3], the selection of maximum margin will give the best level of accuracy. To determine hyperplane equation $\langle w, x\rangle + b = 0$, we must determine first the value of w and b. In this case, we use canonical hyperplanes,

$$\langle w, x^+\rangle + b = +1 \quad \text{and} \quad \langle w, x^-\rangle + b = -1$$

where , x^+ is data which is in the class of y = +1 and the nearest to hyperplane, and x^- is data in the class of y = -1 and the nearest to hyperplane $\langle w,x\rangle + b = 1$.

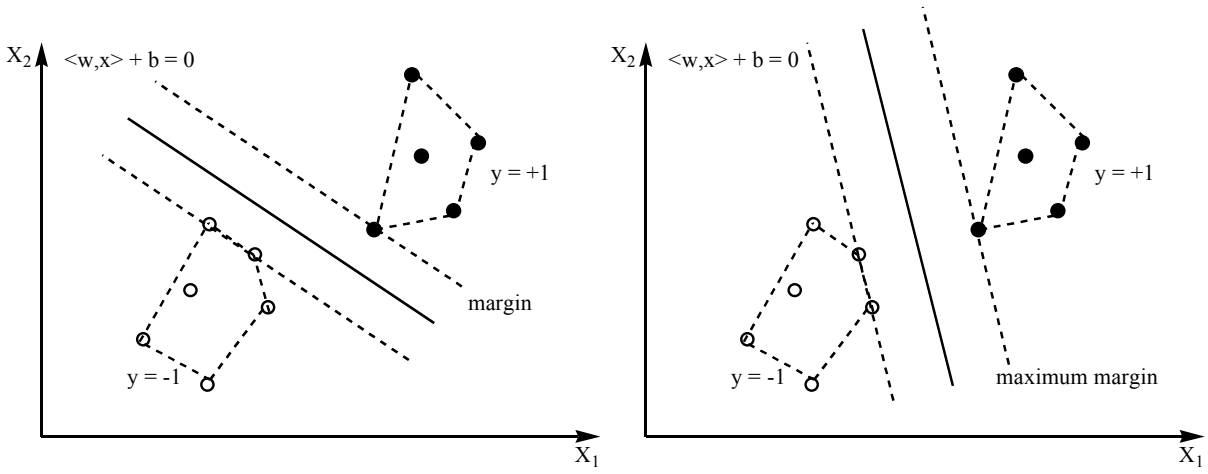

Fig. 2 Margin of hyperplane.

The width of margin γ is a distance from x^+ (or x^-) to hyperplane,

$$\gamma = \left(\frac{\left| \langle w, x^+ \rangle + b \right|}{\|w\|} \right) = \left(\frac{\left| \langle w, x^- \rangle + b \right|}{\|w\|} \right) = \left(\frac{\langle w, x^+ \rangle + b}{\|w\|} \right) = \left(-\frac{\langle w, x^- \rangle + b}{\|w\|} \right) = \frac{1}{\|w\|}$$

This implies that the maximization of margin is equivalent to the minimization of ||w|| subject to $\langle w, x_i\rangle + b \geq +1$ if $y_i = 1$ and $\langle w, x_i\rangle + b \leq -1$ if $y_i = -1$ which can be combined into one constraint

$$y_i (\langle w, x_i\rangle + b) \geq 1 \ \forall i$$

Hence, SVMs learning problem becomes the following quadratic programming problem (QP) as follows :

Minimize <w, w>
subject to y_i (<w, x_i> + b) ≥ 1, i = 1, …, m. (2)

The solution of QP problem above (i.e. w, b) give the optimum hyperplane, with maximum margin $\gamma = 1/\|w\|$. We transform (2) into dual form by introducing Lagrange multiplier $\alpha_i \geq 0$ (Karush-Kuhn-Tucker condition),

$$L(w,b,\alpha) = \frac{1}{2}\langle w, w\rangle - \sum_{i=1}^{m} \alpha_i \left[y_i \left(\langle w, x_i \rangle + b \right) - 1 \right] \tag{3}$$

By equalizing zero the derivation of L with respect to w and b, will give

$$w = \sum_{i=1}^{m} y_i \alpha_i x_i \quad \text{and} \quad 0 = \sum_{i=1}^{m} y_i \alpha_i \tag{4}$$

Substituting (4) into Lagrange function (3), will give

$$\begin{aligned} L(w,b,\alpha) &= \frac{1}{2} \sum_{i,j=1}^{m} y_i y_j \alpha_i \alpha_j \langle x_i, x_j \rangle - \sum_{i,j=1}^{m} y_i y_j \alpha_i \alpha_j \langle x_i, x_j \rangle + \sum_{i=1}^{m} \alpha_i i \\ &= \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} y_i y_j \alpha_i \alpha_j \langle x_i, x_j \rangle \end{aligned}$$

Hence, QP problem (2) becomes

$$\text{Maximize } W(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i=1}^{m} y_i y_j \alpha_i \alpha_j \langle x_i, x_j \rangle \tag{5}$$

$$\text{subject to } \begin{array}{l} \sum_{i=1}^{m} y_i \alpha_i = 0 \\ \alpha_i \geq 0, i = 1, \ldots, m \end{array}$$

The solution of (5) will give an optimum weight vector $w^* = \sum_{i=1}^{m} y_i \alpha_i^* x_i$ which define a hyperplane with maximum margin, $\gamma = 1/\|w\|$.

The optimum value of b* can be determined from the constraints of QP problem (2). Before continuing, we consider the inequality in (2). We observe that for this inequality, there will be some training vectors only for which the equality y_i (<w, w_i> + b) = 1, i = 1, …, m, holds true. Those training vectors are called *support vectors*.

2.1.3. Linear Support Vector

Learning technique described above will only work in linearly separable training data. For some problems, sometimes we have non linearly separable training data. The classifier of this data is obtained by introducing a function which map the data space to finite dimensional space (we call *feature space*). By this mapping the nonlinear training data becomes linear (Figure 3) After then, we try to seek an optimal hyperplane as described above.

By using the weight w obtained, decision function f(x) can be calculated as an inner product of training data and testing data,

$$f(x) = sign\left(\left\langle w^*, x\right\rangle + b\right) = sign\left(\sum_{i=1}^{l} \alpha_i^* y_i \left\langle x_i, x\right\rangle + b\right)$$

If we denote the function which map the data to the feature space as ϕ, then the decision function above can be denoted as follows :

$$f(x) = sign\left(\left\langle w^*, x\right\rangle + b\right) = sign\left(\sum_{i=1}^{l} \alpha_i^* y_i \left\langle \phi(x_i), \phi(x)\right\rangle + b\right) \quad (6)$$

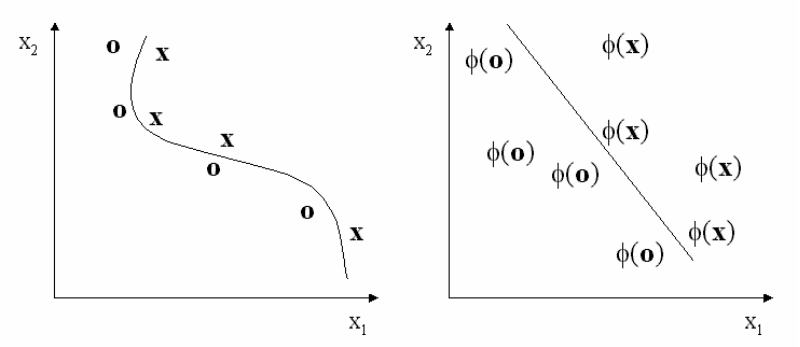

Fig. 3 Mapping to *feature space by* function ϕ.

If we want to determine inner product $\langle\phi(w_i), \phi(w)\rangle$ directly, we have to know ϕ explicitly. But there is a way for computing the inner product without using ϕ in feature space. This effective trick is done by using kernel function, $K(x_i, x) = \langle\phi(x_i), \phi(x)\rangle$ [8].

Hence, decision function (6) can be denoted as follows

$$f(x) = sign\left(\sum_{i=1}^{m} \alpha_i^* y_i K(x_i, x) + b\right)$$

There are many kernel function has been developed. In this paper we use a polynomial function,

$$K(x, x') = \langle x, x'\rangle^d,\ d \in Z^+ \quad (7)$$

2.1.4. SVMs with Soft Margin

The main problem with the optimal hyperplane described above is under assumption that there is no data training error, but normally there are errors happened. Such problem can be omitted by using a soft margin technique. By this technique, we introduce slack variable ξ in (2),

$$\text{Minimize } \langle w, w\rangle + C \sum_{i=1}^{m} \xi_i^2 \quad (8)$$

$$\text{Subject to } y_i(\langle w, x_i\rangle + b) \geq 1 - \xi_i,\ i = 1, \ldots, m$$

$$\xi_i \geq 0,\ i = 1, \ldots, m$$

By this technique, learning method would be a problem to determine an optimum hyperplane having minimum earning data error simultaneously. We call this technique by *structural risk minimization*.

By using the same procedure as above, the dual form of (8) can be denoted as

$$\text{Maximize } W(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i=1}^{m} y_i y_j \alpha_i \alpha_j K(x_i, x_j) \quad (9)$$

$$\text{Subject to } \sum_{i=1}^{m} y_i \alpha_i = 0$$
$$C \geq \alpha_i \geq 0, i = 1,..., m$$

Suppose the solution obtained is α^*, the decision function is $f(x) = \sum_{i=1}^{l} y_i \alpha_i^* K(x_i, x) + b^*$, where b^* is selected in such a way that $y_i f(x_i) = 1 \; \forall$ i with $C > \alpha^*_i > 0$. Hence, decision function sign(f(x)) is a hyperplane in feature space which implicitly defined by using $K(x_i, x)$, with margin

$$\gamma = \left(\sum_{i,i \in SV} y_i y_j \alpha_i^* \alpha_j^* K(x_i, x_j) \right)^{-1/2}$$

3. Computer Simulation

A computer simulation has been executed by using Matlab 6.5.1 running in a PC Pentium IV. In this simulation, PNNs and SVMs models are used to determine whether a DNA (Deoxyribo Nucleic Acid) sequence is a donor type or not. This is part of a process in splice site type detection on DNA sequences (compose of nucleotide Guanine (G), Adenine (A), Thymine (T), Cytosin (C). This splice site detection will be used later in protein folding (introns removal and exons binding).

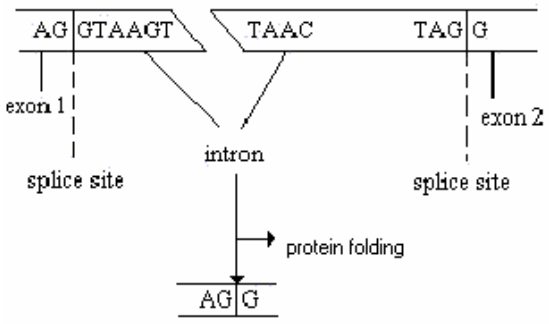

Fig. 4 Splice sites and protein folding.

3.1. Data Used For Experiment

Data in form of *DNA* sequences existed in *Splice-junction Gene Sequences Data Base*, which is part of *Molecular Biology Data Bases* available in [10]. Each sequence has 60 *base-pair length*, and each nucleotide is represented in four bits, *A*: 1000, *T*: 0100, *G*: 0010 and *C*: 0001. Hence, the input data dimension is 60 x 4 = 240 bit. Meanwhile, the output (placed in the beginning of DNA sequence) consist of 1 bit, with the value of 1 if the splice site is donor type and 0 otherwise.

Tabel 1 The accuracy level (%), (i) PNNs (ii) Linear SVMs (iii) Nonlinear SVMs.

	|T|				
	250	300	400	500	600
|L|=250 |V|=250	(i) 90.00 (ii) 96.82 (iii) 96.85	(i) 89.01 (ii) 94.93 (iii) 94.97	(i) 92.75 (ii) 94.68 (iii)94.68	(i) 92.80 (ii) 94.23 (iii)94.24	(i) 91.17 (ii) 93.86 (iii)93.89
|L|=500 |V|=250	(i) 93.20 (ii) 96.87 (iii) 96.89	(i) 93.67 (ii) 95.19 (iii)95.19	(i) 93.25 (ii) 95.92 (iii)95.95	(i) 93.20 (ii) 95.45 (iii)95.48	(i) 92.23 (ii) 95.32 (iii)95.31
|L|=600 |V|=250	(i) 95.20 (ii) 97.05 (iii)97.10	(i) 94.33 (ii) 97.07 (iii)97.06	(i) 95.20 (ii) 96.89 (iii)96.90	(i) 94.20 (ii) 96.47 (iii)96.60	(i) 93.17 (ii) 95.87 (iii)95.88

In the experiment have been done, the data above is divided into three groups, called learning (or training) data L, validation data V, and testing data T. Those three groups of data is used sequentially. The member of each group is selected randomly with cross validation technique from total data given (we used 3175 sequences). This selection technique is nearly similar to technique used by Bolat et al. [8].

In the application with PNNs model, L is used to obtain the weight of PNNs, V is used to determine the best parameter σ in (1). In the application with SVMs (here we used linear SVMs, and Non linear SVMs), L and V are used to obtain the parameter values of C in (9), d in polynomial function (7), in a similar way.

3.2. Test of Accuracy Level

In this paper we define the accuracy level (sometimes called generalization capability) as follows: with the set of testing data T = $\{(x_1,y_1), \ldots\ldots, (x_k,y_k)\}$, we verify whether output $\hat{y}_i = f(x_i)$ is equal or not to y_i as the target. The accuracy level is : [(the number of x_i where $\hat{y}_i = y_i$) / |T|] x 100 %.

By using a various size of testing data T that have been prepared before, we determine the accuracy level of PNNs and SVMs. Here, 2nd bit until 241th bit from each DNA sequences are processed by PNNs and SVMs, and the result is verified with 1st bit of the sequences, whether equal or not. The verification result gives the level of accuracy of the PNNs and SVMs

4. Results Analysis and Discussion

The implementation results are given in Table 1 below. Figure 5 shows the accuracy level of PNNs, Linear SVMs, Non Linear SVMs, for learning data size |L|=600, validation data size |V|=250, with various testing data size |T|.

From Table 1, roughly we can see that the increasing of testing data size will decrease the accuracy level. In the case of increasing of learning data, the accuracy level will increase also. Intuitively, this phenomena is quite similar to the meaning of learning.

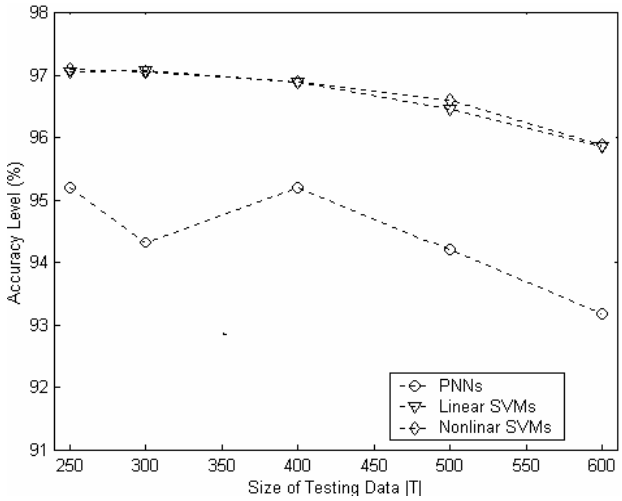

Fig. 5 Accuracy Level of PNNs, Linear SVMs, Non Linear SVMs, for |L|= |600|, |V|=250.

Table 1 shows also that SVMs have a better accuracy level than PNNs. But, there are no significant different between Linear SVMs and Nonlinear SVMs in their accuracy level. It shows that the data is linearly separable. Finally, we can see that there are slightly decreasing of accuracy level, in case of testing data size getting bigger (|T| > 400), perhaps it is caused by the computer limitation.

Roughly, those results show that SVMs have accuracy level about 95.7 % and PNNs have about 92.8 %.

5. Conclusion

Based on the result discussed above, in the case of splice site detection of DNA sequences problem, SVMs has better accuracy level than PNNs. There is no significant different between Nonlinear SVMs and Linear SVMs on their accuracy level.

6. Acknowledgments

This paper is a part of study on 'Preliminary studies of machine learning application on bioinformatics' (2002-3), supported by QUE Project - Departemen Matematika Universitas Indonesia. The QUE Project DGHE, was supported by World Bank Loan no.4193-IND 1997. The author thanks this project for the financial support.

7. References

[1] D.F. Specht, Neural Networks, 3 (1991) 108.
[2] L. Faucett, Fundamentals of Neural Network, Prentice Hall Inc, Englewood Cliffs, 1994.
[3] V.Vapnik, The Nature of Statistical Learning Theory, Springer, New York, 1995.
[4] M.A. Hearst. B. Schölkopf. S. Dumais. E. Osuna and J. Platt, IEEE Intelligent System, 1 (1998) 18.
[5] V. Vapnik, Statistical Learning Theory, John Wiley & Sons, New York, 1998.
[6] C.J.C. Burges, Data Mining and Knowledge Discovery, 2 (1998) 955.
[7] N.Cristianini, J.Shawe-Taylor, An Introduction to Support Vector Machines and other Kernel-Based Learning Methods, Cambridge Univ.Press, 2000.
[8] J.Shawe-Taylor, N.Cristianini, Kernel Methods for Pattern Analysis, Cambridge Univ.Press, 2004.
[9] B. Bolat, T. Yildirim, Electrical & Electronic Engineering, Istanbul Univ., 4 (2004) 1137.
[10] Molecular Biology Data Bases, http://www.ics.edu/~mlearn/ Mlsummary.html