IEEE Int. Conf. Neural Networks & Signal Processing
Nanjing, China, December 14-17, 2003

SPARSE TRAINING PROCEDURE FOR KERNEL NEURON

Jianhua XU[1], *Xuegong ZHANG*[2], *Yanda LI*[2]

[1] School of Mathematical and Computer Sciences, Nanjing Normal University, Nanjing 210097, Jiangsu Province, China, Email to xujianhua@email.njnu.edu.cn
[2] Department of Automation, Tsinghua University / State Key Laboratory of Intelligent Technology and Systems, Beijing 100084, China, Email to zhangxg@mail.tsinghua.edu.cn

ABSTRACT

The kernel neuron is the generalization of classical McCulloch-Pitts neuron using Mercer kernels. In order to control generalization ability and prune structure of kernel neuron, we construct a regularized risk functional including both empirical risk functional and Laplace regularization term in this paper. Based on the gradient descent method, a novel training algorithm is designed, which is referred to as the sparse training procedure for kernel neuron. Such a procedure can realize three main ideas: kernel, regularization (or large margin) and sparseness in the kernel machines (e.g. support vector machines, kernel Fisher discriminant analysis, etc.), and can deal with the nonlinear classification and regression problems effectively.

Keywords: kernel neuron, support vector machine, sparseness, regularization.

1. INTRODUCTION

In the artificial neural networks the basic element is McCulloch-Pitts neuron (or M-P neuron)[1]. Rosenblatt [2] proposed the first learnable procedure: Perceptron, which could only deal with the linearly separable cases as a simple linear classifier. In order to handle the more complicated real-world problems, many models and their training procedures are introduced, e.g. back propagation training method for multilayer perceptron [3], adaline with some nonlinear transform [4], and radial basis function net (RBF)[5].

Recently, several kernel-based machines for nonlinear problems, such as support vector machines (SVM)[6-8], kernel Fisher discriminant analysis (KFD)[9], and large margin kernel pocket algorithm [10], are gaining more and more attention in the nonlinear classifier designing. There exist three attractive concepts: kernel idea, large margin or regularization, and sparseness.

The kernel idea is an effective technique to realize the nonlinear transform implicitly. Xu et al [11] introduced the kernel neuron by generalizing M-P neuron through Mercer kernels, and constructed a simple training algorithm based on the gradient descent method. Kernel neuron and its training procedure can be considered as a unified framework for three nonlinear techniques mentioned above in the neural networks.

The regularization technique developed by Tikhonov & Arsenin [12] is to handle ill-posed problems. Such a technique has widely been used in neural networks. It has been found that adding a proper regularization term to an objective functional can result in significant improvements in net generalization [13], and also can prune the structure of nets [14]. There mainly are three usual regularization terms: the squared or Gaussian, absolute or Laplace, and normalized or Cauchy regularization terms. With respect to the effeciency of supervised learning, Saito and Nakano[13] gave a detialed comparison on the three regularizaion terms and different learning algorithms, and pointed out that the combination of the squared regularization term and the second order learning algorithm drastically improves the convergence and generalization ability. Williams [15] concluded that the Laplace regularization term is more appropriate than the Gaussian one from the viewpoint of net pruning. Ishikawa [16] used the Laplace regularizer to construct a simple but effective learning method called a structural learning with forgetting in order to pruning the forward neural nets.

In this paper, in order to improve the generalization ability and to get a sparse discriminant or regression function for kernel neuron, we add the Laplace regularization term to the original empirical risk functional defined in the paper of Xu et al [11]. Based on gradient descent approach, a training algorithm is constructed. It can be referred to as the sparse training procedure for kernel neuron, and can realize three main ideas in support vector machines. As a nonlinear technique, it can handle both nonlinear classification and regression problems effectively.

2. DEFINITION OF KERNEL NEURON

This paper is devoted to two problems: classification and regression problems, in neural networks. Let

$$\{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), ..., (\mathbf{x}_i, y_i), ..., (\mathbf{x}_l, y_l)\} \quad (1)$$

be the training set of l iid samples, where $\mathbf{x}_i \in R^n$. For the classification problem of binary classes (ω_1, ω_2), suppose

$$y_i = \begin{cases} +1, & if \ \mathbf{x}_i \in \omega_1 \\ -1, & if \ \mathbf{x}_i \in \omega_2 \end{cases} \quad (2)$$

while for the regression problem, suppose $y_i \in R, \ i = 1, ..., l$.

The classical M-P neuron is defined as

$$o(\mathbf{x}) = f((\mathbf{w} \cdot \mathbf{x}) + b) \quad (3)$$

where o is the output of neuron, $\mathbf{x}$ is the input vector, $\mathbf{w}$ and b are the weight vector and threshold respectively. Function f implies the transform function. For the M-P neuron, f is a hard limiting function, i.e. the sign function. In neural networks, f is a continuously differentiable and monotone function, e.g., sigmoid function and linear function.

In the paper of Xu, et al [11], a kernel version of M-P neuron is defined as

$$o(\mathbf{x}) = f(\sum_{i=1}^{l} \alpha_i k(\mathbf{x}_i, \mathbf{x}) + \beta) \quad (4)$$

where $\alpha_i \in R, \ i = 1,2,...l$ are the coefficients corresponding to each sample, $k(\mathbf{x}_i, \mathbf{x})$ is the kernel function satisfying Mercer conditions, e.g., polynomial kernel, RBF kernel and two layers neural net kernel [7,8].

Note that generally speaking the input-output relationship is nonlinear in kernel neuron. Only in the case when the transform function is linear and kernel function is the linear kernel (namely $k(\mathbf{x}_i, \mathbf{x}) = \mathbf{x}_i^T \mathbf{x}$), the relationship is linear, and can be considered as the equivalent form of M-P neuron.

The kernel neuron utilizes the nonlinear kernels to realize the nonlinear transform form the original input vector space (R^n) to the real number space (R).

3. SPARSE TRAINING PROCEDURE FOR KERNEL NEURON

For the kernel neuron, Xu et al [11] defined an empirical functional and constructed a training algorithm based on standard gradient descent scheme. Such a training procedure only realizes the kernel idea. Specially, it is difficult to control the generalization ability and to obtain a sparse decision function. Adding a proper regularization term in risk functional to decay the connection weights is simple way to pruning weights without complicating the learning algorithm much [14]. In kernel neuron, such a pruning implies that a sparse representation would occur, i.e., many α_i would be close to zeros. Simultaneously regularization method also can improve the generalization and convergence of training procedure.

Thus, we define a regularized risk functional consisting of empirical risk (square error summation between the actual output and desired output) and the Laplace regularization term, that is,

$$E(\boldsymbol{\alpha}, \beta) = \frac{1}{2} \sum_{j=1}^{l} [y_j - o(\mathbf{x}_j)]^2 + \mu \sum_{j=1}^{l} |\alpha_j| \quad (5)$$

where $\boldsymbol{\alpha} = [\alpha_1, ..., \alpha_l]^T$, μ is the regularization parameter. Now our goal is to construct an effective algorithm to find out the coefficient vector $\boldsymbol{\alpha}$ and threshold β that minimize risk functional (5). This can still be done by the standard gradient descent scheme. The gradient of (5) is,

$$\frac{\partial E}{\partial \alpha_m} = -\sum_{j=1}^{l} \left[y_j - f(\bar{y}_j) \right] f'(\bar{y}_j) k(\mathbf{x}_m, \mathbf{x}_j) + \mu \ \mathrm{sgn}(\alpha_m)$$
$$m = 1, 2, \cdots, l \quad (6)$$
$$\frac{\partial E}{\partial \beta} = -\sum_{j=1}^{l} \left[y_j - f(\bar{y}_j) \right] f'(\bar{y}_j)$$

where $\bar{y}_j = \sum_{i=1}^{l} \alpha_i k(\mathbf{x}_i, \mathbf{x}_j) + \beta$, $f'(\bar{y}_j)$ is the first derivative of $f(\bar{y}_j)$, and sgn is the sign function.

Like the back-propagation training algorithm in the forward neural networks, we also use single sample correction and add a momentum term in iterative procedure. Therefore, a novel iterative procedure for kernel neuron could be rewritten as:

Algorithm-1 (SKN-1):

1. Let t=0 and $\alpha_m(t), \beta(t)$ be arbitrary.
2. Pick up some sample $\mathbf{x}_j$.
3. Set t=t+1.
4. Calculate

$$\bar{y}_j(t) = \sum_{i=1}^{l} \alpha_i(t-1) k(\mathbf{x}_i, \mathbf{x}_j) + \beta(t-1) \quad \text{and} \quad f'(\bar{y}_j(t))$$

5. Calculate

$$\Delta\alpha_m(t) = \lambda_1 (y_j - f(\bar{y}_j(t))) f'(\bar{y}_j(t)) k(\mathbf{x}_m, \mathbf{x}_j)$$
$$- \lambda_2 \ \mathrm{sgn}(\alpha_m(t-1)) + \lambda_3 \Delta\alpha_m(t-1)$$
$$\Delta\beta(t) = \lambda_1 (y_j - f(\bar{y}_j(t))) f'(\bar{y}_j(t)) + \lambda_3 \Delta\beta(t-1)$$

6. Update

$$\alpha_m(t) = \alpha_m(t-1) + \Delta\alpha_m(t)$$
$$\beta(t) = \beta(t-1) + \Delta\beta(t)$$

7. If $\|\Delta\alpha_m(t)\|^2 + \Delta\beta(t)^2 < \varepsilon$ or $t \geq t_{\max}$, then stop; otherwise, go to step 2.

where m=1,…,l, λ_1 is the learning rate, $\lambda_2 = \lambda_1 \mu$, λ_3 denotes the momentum parameter, ε is a threshold to stop algorithm, and $t_{\max}$ is the maximal iterations.

Ishikawa [16] pointed out that in neural nets such a weight decay is constant in contrast to exponential decay [17] and unnecessary connections fade away. This means that a large number of parameters are close to zeros and the sparseness happens. Particularly when $\lambda_2 = \lambda_3 = 0$, this approach is the same as the simple training procedure for kernel neuron [11].

Ishikawa [16] also advised a selective pruning procedure to make only the connection weights decay whose absolute values is below a threshold θ after a training procedure listed above. Another regularized risk functional for kernel neuron could be constructed as,

$$E(\boldsymbol{\alpha}, \beta) = \frac{1}{2}\sum_{j=1}^{l}[y_j - o(\mathbf{x}_j)]^2 + \mu \sum_{\substack{j=1 \\ |\alpha_j|<\theta}}^{l} |\alpha_j| \qquad (7)$$

Such an idea can improve the goodness of fit of a model and decay the small values further. This means that a more sparse representation can occur. The novel sparse training procedure would be comprised of two steps as follow:

Algorithm-2 (SKN-2):
1. Algorithm-1 listed above.
2. Algorithm-1, but including a threshold θ for α .

In this paper, we call two training algorithms above as the sparse training procedure 1 and 2 for kernel neuron, or simply SKN-1 and SKN-2 respectively.

As in the sparse LS-SVM [18], we refer to the $\{\alpha_1, ..., \alpha_l\}$ as the spectrum of kernel neuron. The sparseness means that many components of spectrum are very close to zeros. If such components are forced to zero, the final discriminant function changes little. In this paper, a proper threshold δ is set to impose some components zeros. If $|\alpha_i| \geq \delta$, this sample or vector is still called as support vector. Now the final discriminant or regression function can be represented as

$$f(\mathbf{x}) = f(\sum_{\substack{i=1 \\ |\alpha_i|\geq\delta}}^{l} \alpha_i k(\mathbf{x}_i, \mathbf{x}) + \beta) \qquad (8)$$

In the case when we handle binary classification problem, for some new input vector $\mathbf{x}$, if $f(\mathbf{x}) > 0$, then $\mathbf{x} \in \omega_1$, otherwise $\mathbf{x} \in \omega_2$. For the regression problem, we consider $f(\mathbf{x})$ as the regression result.

4. EXPERIMENT RESULTS AND ANALYSIS

To evaluate the performance of our new training procedure, we devised three artificial data sets: a linearly separable cases, a nonlinear case with 10 misclassified samples, and a nonlinear regression.

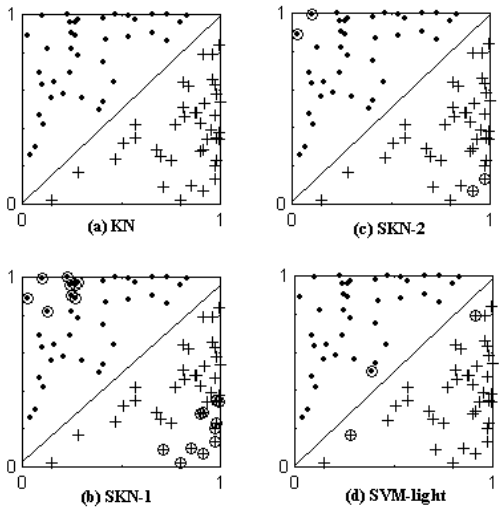

Fig.1: The separation hyperplanes from different algorithms for the linear case.

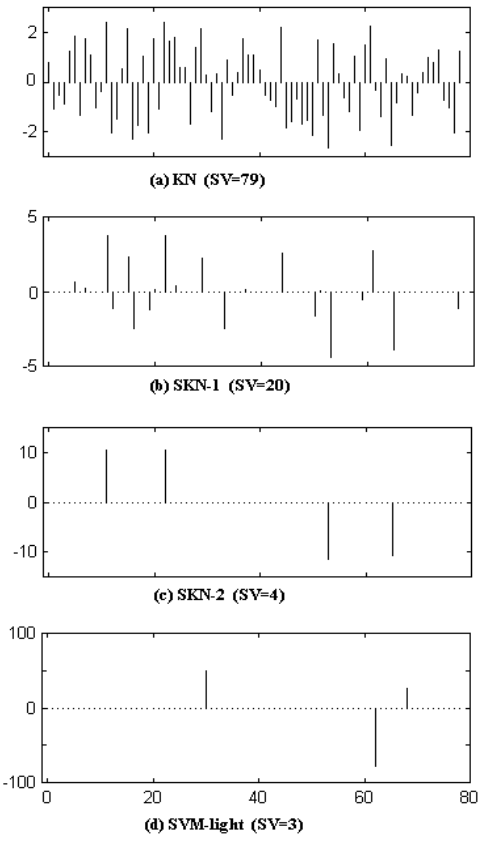

Fig.2: The spectra from different approaches for the linear case.

The example in Fig.1 is a linearly separable case, where there exist 79 samples of two classes (marked by crosses and points in the figure) which can be classified without error by several linear classifiers. Fig.1 illustrates the seperation lines obtained by using KN [11], SKN-1, SKN-2 and SVM-light method [19] with linear kernel $k(\mathbf{x},\mathbf{y}) = \mathbf{x}^T\mathbf{y}$, where the circles indicate the support vectors (or SV).

Fig.2 shows the corresponding spectra of these learning appproaches. In KN (Fig.2 (a)), the spectrum is not sparse obviously. In Fig.2 (b) and (c), the large number of components are close to zeros and the sparesness occures in the dicision function. Fig. 2 (d) illustrates the spectrum of SVM-light method [19].

For the nonlinear problem, we design an example which contains ten misclassified samples using some linear classifiers. Fig.3 shows the decision boundaries from KN, SKN-1, SKN-2 and SVM-light with the linear kernel. Note that there exists two contradiction samples, i.e. $\mathbf{x}_i = \mathbf{x}_j, y_i \neq y_j, i \neq j$. We find out that the number of support vectors from our sparse training algorithm is less than that from SVM-light (C=1000, other parameters are default values). This example demonstrates that our algorithm works well for the nonlinear problem.

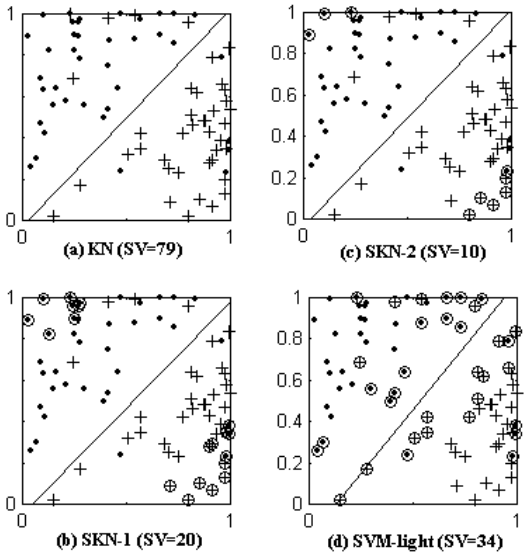

Fig.3: The separated hyperplanes from different algorithms for nonlinear case.

To the nonlinear regression problem, a function $f(x) = (1 - x + 2x^2)e^{-0.5x^2}$ is used [13]. In the experiment, a value of x is randomly generated in the range from –4 to +4, and the corresponding value of function is computed and added a guassian noise with a zero mean and a variance 0.04. The total number of training samples is 30. When a radial basis function kernnel with width 1.0 is utilized, Fig. 4(a) demonstrates the regression result from SKN-2, where the solid curve is the regression result, the dashed line true function, and the crosses are actual samples. In Fig. 4 (b), the result comes from SVM with RBF kerenl (width =0.4), C=100.0, $\varepsilon = 0.2$. In both (a) and (b), the circles denote support vectors. It is easy to see that the number of support vectros form SKN-2 is less than that form SVM, that is 6 versus 16.

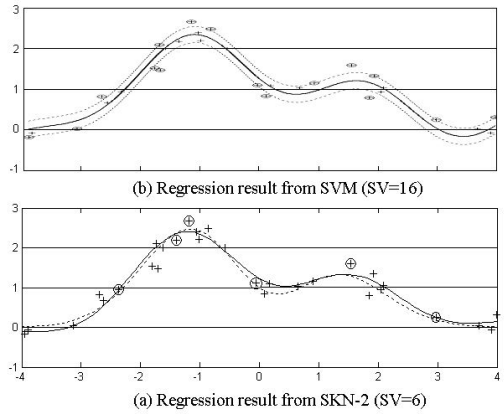

Fig.4: The nonlinear regress example with SKN-2 and SVM.

Three artificial examples above show sparse training procedure can work well for both nonlinear calssification and regress problems. It is possible that our methods can obtain more sparse function than SVM does.

5. DISCUSSIONS AND CONCLUSIONS

The kernel neuron is the nonlinear generalization of neuron with kernels. In order to control the generalization ability and obtain the sparse decision function, two regularized risk functionals are defined for kernel neuron, which consist of the emperical risk fuctional and Laplace regularization term. Based on the gradient descent scheme, two sparse training procedures are developed, i.e. SKN-1 and SKN-2. The new methods can be regarded as a new general-purpose nonlinear learning machine, since they can be applied for both nonlinear pattern recognition and regression problems. Experiments on artifical data sets show that they work well on both linearly seperatable and non-seperatable data, and also on regression problem.

For three usual kinds of kernel functions, our kernel neuron and its sparse training procedures can implement the similar performances of multi-layer perceptrons (the kernel of two layer neural networks), radial basis function net (RBF kernel) and the adaline with nonlinear

preprocessors (ploynomial kernel). Furthermore SKN protects us from designing hidden layer node, clustering the centers and constructing the polynomial transfrom, etc.

6. ACKNOWLEDGEMENTS

This work is supported by National Natural Science Foundation of China, project No. 60275007.

7. REFERENCES

[1] W. S. McCulloch, W. Pitts, "A logical calculus of the ideas immanent in nervous activity", Bulletin of Mathematical Biophysics 5, 115-131, 1943.

[2] F. Rosenblatt, "The perceptron: probabilistic model for information storage and organization in the brain", Psychological Review, 65(6), 386-408, 1958.

[3] D. E. Rumelhart, G. E. Hinton, R. J. Williams, "Learning representations by back-propagating errors", Nature, 323(9), 533-536, 1986.

[4] D.F. Specht, "Generation of polynomial discriminant function for pattern recognition", IEEE Transactions on Electronic Computer, EC-16, 308-319, 1967.

[5] S. Theodoridis, K. Koutroumbas, *Pattern Recognition*, Academic Press, San Diego, 1999.

[6] C. Cortes, V. N. Vapnik, "Support vector networks", Machine Learning, 20(3), 273-297, 1995.

[7] V. N. Vapnik, *Statistical Learning Theory*, Wiley, New York, 1998.

[8] V. N. Vapnik, *The Nature of Statistical Learning Theory* (2nd ed.), Springer-Verlag, New York, 1999.

[9] S. Mika, G. Ratsch, J. Weston, B. Scholkopf, K-R. Muller, "Fisher discriminant analysis with kernels", Proceedings of Neural Networks for Signal Processing IX, 41-48, IEEE Press, New York, 1999.

[10] J. Xu, X. Zhang, Y. Li, "Large margin kernel pocket algorithm", Proceedings of IJCNN2001, 1480-1485, Washington DC, IEEE Press, 2001.

[11] J. Xu, X. Zhang, Y. Li, "Kernel Neuron and its Training Algorithm", Proceedings of 8th International Conference on Neural Information Processing, Vol. 2, 861-866, Shanghai China, Fudan University Press, 2001.

[12] A. N. Tikhonov, V. Y. Arsenin, *Solution of Ill-posed Problem*, W. H. Wanston, Washington DC, 1977.

[13] K. Saito, R. Nakano, "Second order learning algorithm with squared penalty term", Neural Computation, 12(3), 709-729, 2000.

[14] R. Reed, "Pruning algorithms -- a survey", IEEE Transactions On Neural Networks, 4(5), 740-747, 1993.

[15] P. M. Williams, "Bayesian Regularization and pruning using a Laplace prior", Neural Computation, 7(1), 117-143, 1995.

[16] M. Ishikawa, "Structural learning with forgetting", Neural Networks, 9(3), 509-521, 1992.

[17] D.C. Pault, S. J. Nowlan, G. E. Hinton, "Experiments on learning by back propagation", Technical Report, CMU-CS-86-126, Carnegie-Mallon University, 1986.

[18] J. A. K. Suykens, L. Lukas, J. Vandewalle, "Sparse least squares support vector machines classifiers", In 8th European Symposium on Artificial Neural Networks (ESANN'2000), 37-42, 2000.

[19] T. Joachims, "Making large-scale SVM learning practical", Advances in Kernel Methods—Support Vector Learning, Scholkopf B, Burges C, and Smola A (ed), 169-184, Cambridge MA, MIT Press, 1999.