表現能力と汎化性能の高い人工神経回路網に関する研究
On training of embedding functions in SVM

研究代表者　研　究　員　**趙　　晋輝**（中央大学理工学部）
共同研究者　研　究　員　**辻井　重男**（中央大学理工学部）
共同研究者　共同研究員　**星野　美保**（ＮＥＣ(株)）

1 Introduction

Support vector machines combined with kernel techniques are very successful in various applications such as classification and regression, etc with high capability of generalization [1][2]. On the other hand, following problems can be observed in formulation of such kernel-based support vector machines. In the first place, it assumed that the data in the feature space are linearly separable, which however, can not be tested a priori. In fact, there are always the possibility for outliers and current treatments of them such as using soft margin etc. at their best only to reduce the damage rather than to solve the problem from the begining.

In the second place, with the fixed kernels or embedding functions from the input space to the feature space, even when the data are linearly separable in the feature space, it could and usually need a very high dimesional feature space, or a large number of such embedding functions. This causes a huge amount of computational and memory cost, the major difficulty in implemtation of support vector machines. Besides, the kernels in infinite dimensional functional spaces are usually designed to behave asymptotically optimal but it seems to be much harder to find the optimal approximation in finite dimension.

In this paper, to overcome the above difficulties, we show an approach of support vector machines different from kernel-based techneque. In particular, instead of using the fixed kernel to determine the embedding functions or feature maps, we train the embedding functions adaptively in order to achieve linearly separability automatically therefore solve the outlier problem completely. A nonlinear function is used in these cost functions to train only mis-classfied data but leave the correctly classified ones remain intact. Besides, this training could also make it possible to find optimal embeddings or feature maps from the input space to a low dimensional feature space, then reduce implementation cost. Furthermore, to maximize the margin as large as one wishes by training of the embedding functions.

Starting from observation on the cost funciton presently for support vector machines, two new cost functions are presented and their properties are discussed. Then algorithms to train the weight coefficients and the parameters in the embedding functions are shown. These algorithms are then applied to support vector machines with RBF kernels.

2 Observation on cost-function

Let $\boldsymbol{w}$ be the weight vector of the SVM, b the threshold, $\{\boldsymbol{x}_i, d_i\}$ the pairs of inputs and desired output as training data, $\boldsymbol{\phi}(\cdot)$ the feature maps or embedding functions from the input space to the feature space.

In order to train the embedding functions or the feature maps from the input space to the feature space, we first try to use the standard Lagrange multiplier cost function of SVM, but assume that the embedding function $\boldsymbol{\phi} = \boldsymbol{\phi}_{\boldsymbol{\theta}}$ is parametrized by $\boldsymbol{\theta}$.

$$L(\boldsymbol{w}, b, \boldsymbol{\alpha}) = \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} - \sum_{i=1}^{n} \alpha_i \left(d_i(\boldsymbol{w}^T\boldsymbol{\phi}_\theta(\boldsymbol{x}_i) + b) - 1\right)$$

but we are going to training the embedding $\boldsymbol{\phi}_\theta$ and the weight $\boldsymbol{w}$ at the same time. The gradients

$$\frac{\partial L}{\partial \boldsymbol{w}} = \boldsymbol{w} - \sum_{i=1}^{n} \alpha_i d_i \boldsymbol{\phi}(\boldsymbol{x}_i)$$

$$\frac{\partial L}{\partial \boldsymbol{\theta}} = -\sum_{i=1}^{n} \alpha_i d_i \boldsymbol{w} \frac{\partial \boldsymbol{\phi}(\boldsymbol{x}_i)}{\partial \boldsymbol{\theta}}$$

can be used to training the weights and embedding functions as well. e.g.

$$\boldsymbol{\theta}(n+1) = \boldsymbol{\theta}(n) + \mu \sum_{i=1}^{n} \alpha_i d_i \boldsymbol{w} \frac{\partial \boldsymbol{\phi}(\boldsymbol{x}_i)}{\partial \boldsymbol{\theta}}$$

However, a big problem, as in the primal problem of support vector machine is that the exact values of the

Lagrange multipliers α_i are unknown here so one can not determine the values of the gradients.

This problem was successfully overcomed by switching to the dual problem of which the dual cost function is castedl purely in the multipliers as unknown. Unfortunately this trick does not work for our case since the dual cost function will, as the primal one, still contain the parameter $\boldsymbol{\theta}$ of embedding functions.

Another problem is that the errors even when

$$d_i(\boldsymbol{w}^T\boldsymbol{\phi}(\boldsymbol{x}_i)+b) > 1$$

or the correctly classifield data are taken into it and also used in training. Although theoretically one can disgard the data with zero mulpliers and only consider the support vectors or the data on the support hyperplane, the information of multipliers virturally equals the solution of SVM itself. Thus, in practice one has to consider the possibility of all data to be support vectors and this cost function simply drive all data onto the support hyperplanes.

In the following section, we will show new cost functions which overcome these problems

3 New cost functions

We return to the original formulation of SVM:

$$\min\ f(\boldsymbol{w}) = \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} \tag{1}$$

$$\text{subject to}\quad d_i(\boldsymbol{w}^T\boldsymbol{\phi}(\boldsymbol{x}_i)+b) \geq 1, i = 1, ..., N \tag{2}$$

Let $b = w_0$, one may redefine the discrimination function to absort the threshold b into $\boldsymbol{w}$, and use the same $\boldsymbol{w}$ to denote the new weight vector:

$$\boldsymbol{w} = (b, w_1, ..., w_M)^T = (w_0, w_1, ..., w_M)^T$$

and also $\phi_0 = 1$ so let $\boldsymbol{y} = \boldsymbol{\phi}(\boldsymbol{x})$ be the embedding function from input space to feature space.

$$\begin{aligned}\boldsymbol{y} &= (1, \boldsymbol{\phi}^T(\boldsymbol{x}))^T \\ &= (1, \phi_1(\boldsymbol{x}), ..., \phi_j(\boldsymbol{x}), ..., \phi_M(\boldsymbol{x})))^T \\ &= (1, y_1, ..., y_M)^T.\end{aligned}$$

In particular, the embedding functions $y_j = \phi_j$ are parameterized by the parameter vectors $\boldsymbol{\theta}_j$ i.e.

$$y_j(\boldsymbol{x}) = y_{\boldsymbol{\theta}_j}(\boldsymbol{x})$$

Let $w_0 = b$, then one has

$$g(\boldsymbol{y}) = (w_1, ..., w_M)\boldsymbol{\phi}(\boldsymbol{x}) + b \tag{3}$$

$$= (b, w_1, ..., w_M)\begin{pmatrix} 1 \\ \boldsymbol{\phi} \end{pmatrix} = \boldsymbol{w}^T\boldsymbol{y} \tag{4}$$

Denote $\boldsymbol{y}_i = \boldsymbol{\phi}(\boldsymbol{x}_i), i = 1, ..., N,$

$$\begin{aligned}\boldsymbol{y}_i &= (\phi_1(\boldsymbol{x}_i), ..., \phi_j(\boldsymbol{x}_i), ..., \phi_M(\boldsymbol{x}_i))^T \\ &= (y_1(\boldsymbol{x}_i), ..., y_j(\boldsymbol{x}_i), ..., y_M(\boldsymbol{x}_i))^T\end{aligned}$$

The SVM can separate correctly $\boldsymbol{y}_i = \phi(\boldsymbol{x}_i)$ if

$$d_i g(\boldsymbol{y}_i) = d_i\boldsymbol{w}^T\boldsymbol{y}_i > 0$$

However, in order to formulate the maximizing margin problem as the minimizing weight vector prolem, one needs to stick with the scaling assumption or use instead the condition

$$d_i g(\boldsymbol{y}_i) = d_i\boldsymbol{w}^T\boldsymbol{y}_i \geq 1$$

Even though these two statements above are in fact equivalent, upto a particular scaling of $\boldsymbol{w}$, this scaling here is critical and only the weight vector under this scaling can be used in the SVM cost function.

The SVM then failed to separate the $\boldsymbol{y}_i$ only when

$$d_i g(\boldsymbol{y}_i) = d_i\boldsymbol{w}^T\boldsymbol{y}_i < 1.$$

Define a step function

$$s(x) = \begin{cases} x & x \geq 0 \\ 0 & x < 0 \end{cases}$$

The recognition error can then be evaluated by the value of $s(1 - d_i g(\boldsymbol{y}_i))$ for the scaled $\boldsymbol{w}$.

We will use this latter term in the new cost functions. Let $\Theta = (\boldsymbol{\theta}_j, j = 1, ..., M)$

$$\begin{aligned}J_1(\boldsymbol{w}, \Theta) &= \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + \sum_i s(1 - d_i g(\boldsymbol{y}_i)) \\ &= \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + \sum_{i, d_i g(\boldsymbol{y}_i)<1} (1 - d_i g(\boldsymbol{y}_i)) \\ &= \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + \sum_{i, d_i \boldsymbol{w}^T\boldsymbol{y}_i<1} (1 - d_i\boldsymbol{w}^T\boldsymbol{y}_i)\end{aligned}$$

This cost function is continuous and smooth (linear) where the step function s is nonzero. All we need to calculate is when then step function s to be nonzero.

One can also define another cost function

$$\begin{aligned} J_2(\boldsymbol{w},\Theta) &= \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + \sum_i s^2(1-d_i g(\boldsymbol{y}_i)) \\ &= \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + \sum_{i,d_i g(\boldsymbol{y}_i)<1} (1-d_i g(\boldsymbol{y}_i))^2 \\ &= \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + (d_i\boldsymbol{w}^T\phi_\theta(\boldsymbol{x}_i))^2 \sum_{i,d_i\boldsymbol{w}^T\boldsymbol{y}_i<1} (1-d_i\boldsymbol{w}^T\boldsymbol{y}_i)^2 \end{aligned}$$

which is continously differentiable. This is enough to gradient algorithms. Besides, it has larger penalty for large recognition error but insensitive to small error.

Another important feature of these cost functions is that the minimal points of (1),(2) are contained in the minimal points of J_1 and J_2. This can be observed from the fact that on the minimal points of J_1 and J_2, the second terms of them are zero, which means the margins achieved a minimal while all data are correctly classified. Moreover, they do not drive all data onto the support hyperplanes.

4 Training feature map of SVM

As we assumed that the embedding function $\boldsymbol{y} = \boldsymbol{\phi}(\boldsymbol{x})$ is parametrized by parameter vector $\Theta = (\boldsymbol{\theta}_j, j = 1, ..., M)$. We will show training algorithms for both $\boldsymbol{w}$ and $\boldsymbol{\theta}_j$ using the two new cost functions.

4.1 Training using cost function J_1

The training algorithm uses the gradient of the cost function.

$$\begin{aligned} &\frac{\partial}{\partial \boldsymbol{w}} J_1(\boldsymbol{w},\Theta) \\ &= \frac{\partial}{\partial \boldsymbol{w}} \left(\frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + \sum_{i,d_i\boldsymbol{w}^T\boldsymbol{y}_i<1} 1 - d_i\boldsymbol{w}^T\boldsymbol{y}_i \right) \\ &= \boldsymbol{w}^T - \sum_{i,d_i\boldsymbol{w}^T\boldsymbol{y}_i<1} d_i\boldsymbol{y}_i^T \end{aligned}$$

$$\begin{aligned} &\frac{\partial}{\partial \boldsymbol{\theta}_j} J_1(\boldsymbol{w},\Theta) \\ &= \frac{\partial}{\partial \boldsymbol{\theta}_j} \left(\frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + \sum_{i,d_i\boldsymbol{w}^T\boldsymbol{y}_i<1} 1 - d_i\boldsymbol{w}^T\boldsymbol{y}_i \right) \\ &= \begin{cases} -d_i w_j \frac{\partial y_j(\boldsymbol{x}_i)}{\partial \boldsymbol{\theta}_j} & \text{if } d_i\boldsymbol{w}^T\boldsymbol{y}_i < 1 \\ 0 & \text{if } d_i\boldsymbol{w}^T\boldsymbol{y}_i \geq 1 \end{cases} \end{aligned}$$

4.2 Training using cost function J_2

The cost function J_2 is a quadratic function of $\boldsymbol{w}$. The gradient is

$$\begin{aligned} &\frac{\partial}{\partial \boldsymbol{w}} J_2(\boldsymbol{w},\Theta) \\ &= \frac{\partial}{\partial \boldsymbol{w}} \left(\frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + \sum_{i,d_i\boldsymbol{w}^T\boldsymbol{y}_i<1} (1-d_i\boldsymbol{w}^T\boldsymbol{y}_i)^2 \right) \\ &= \boldsymbol{w}^T + \sum_{i,d_i\boldsymbol{w}^T\boldsymbol{y}_i<1} -2d_i(1-d_i\boldsymbol{w}^T\boldsymbol{y}_i)\boldsymbol{y}_i^T \end{aligned}$$

$$\begin{aligned} &\frac{\partial}{\partial \boldsymbol{\theta}_j} J_2(\boldsymbol{w},\Theta) \\ &= \frac{\partial}{\partial \boldsymbol{\theta}_j} \left(\frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + \sum_{i,d_i\boldsymbol{w}^T\boldsymbol{y}_i<1} (1-d_i\boldsymbol{w}^T\boldsymbol{y}_i)^2 \right) \\ &= \begin{cases} -d_i w_j (1-d_i\boldsymbol{w}^T\boldsymbol{y}_i) \frac{\partial y_j(\boldsymbol{x}_i)}{\partial \boldsymbol{\theta}_j} & \text{if } d_i\boldsymbol{w}^T\boldsymbol{y}_i < 1 \\ 0 & \text{if } d_i\boldsymbol{w}^T\boldsymbol{y}_i \geq 1 \end{cases} \end{aligned}$$

5 Training RBF SVM

The RBF networks with the most general form as

$$y_i(\boldsymbol{x}) = \phi(\boldsymbol{x}_i) = \exp\{-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{c}_i)^T\Sigma_i(\boldsymbol{x}-\boldsymbol{c}_i)\}$$

where the center is $\boldsymbol{c}_i = \boldsymbol{x}_i + \boldsymbol{a}_i$ Even most RBF kernel SVM did not include $\boldsymbol{a}_i$ in the center, Mercer kernel also, although we do not need it to be a prefixed kernel.

In fact, it is vitally important to include $\boldsymbol{a}_i$ in the centers here since we wish to find optimal embedding functions which can asign the input $\boldsymbol{x}_i$ a new position by $\boldsymbol{c}_i = \boldsymbol{x}_i + \boldsymbol{a}_i$ in the feature space so that both linearlly separablity and the maximal margin can be achieved.

In the training algorithm of embedding functions of RBF networks, the gradients with respect to the center and the covariance matrix can be obtained as follows.

$$\frac{\partial y_i}{\partial \boldsymbol{a}_i} = y_i(\boldsymbol{x}-\boldsymbol{c}_i)^T\Sigma_i.$$

Let $\Sigma = (\sigma_{lk}), \boldsymbol{x} = (x_l)^T, \boldsymbol{c}_i = (c_{ik})^T$,

$$\frac{\partial y_i(\boldsymbol{x})}{\partial \sigma_{lk}} = -\frac{y_i(\boldsymbol{x})}{2}(x_l - c_{il})^T(x_k - c_{ik})$$

or $$\frac{\partial y_i(\boldsymbol{x})}{\partial \Sigma} = -\frac{y_i(\boldsymbol{x})}{2}(\boldsymbol{x}-\boldsymbol{c}_i)^T(\boldsymbol{x}-\boldsymbol{c}_i)^T.$$

6 Simulation

The proposed algorithm is applied to train the feature map of a RBF SVM using data of Australian credit card problem and Heart disease problem. As shown in Fig. 1 a RBF with fixed embedding functions in practice are rarely able to linearly separate data even in a high- dimensional feature space, not mention the maximization of margin. However, using the proposed algorithm both

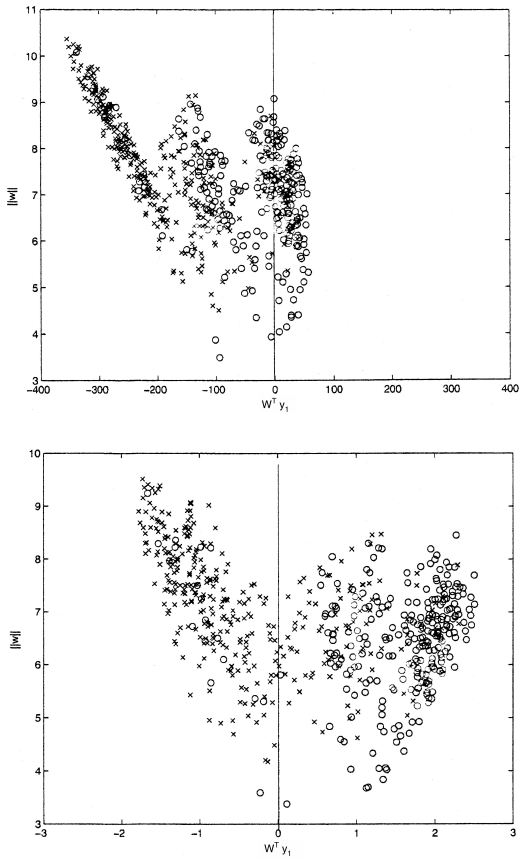

図 1　RBF SVM without and with feature map training

linearly seperatability and margin are improved significantly, which mean improvement of recognition rate and generalization perfermonce.

7 Further extensions

This training strategy of embedding functions of feature maps can also be applied to support vector machines with different kernel or even not kernel-based networks, e.g. single or multilayered percetrons, of multilayered RBF networks [4] and a more general network with a new topology called pyramid networks [3].

参 考 文 献

[1] R. Herbrich, “Learning kernel classifier” The MIT Press, 2001.

[2] B. Scholkopf and A. J. Smola, “Learning with kernels-support vector machines, regulation, optimization, and behyond”, The MIT Press, 2002.

[3] J. Chao, M. Hoshino, T. Kitamura, T. Masuda, “A New Pyramid Network and Its Generalization Performance”, Proceedings of IJCNN ’01: 2001 International Joint Conference on Neural Networks, pp.2811–2816, 2001

[4] J. Chao, M. Hoshino, T. Kitamura, T. Masuda, “A Multilayer RBF Network and Its Supervised Learning”, Proceedings of IJCNN ’01 : 2001 International Joint Conference on Neural Networks, pp.1995–2816, 2001

[5] J. Chao, W.Ratanasuwan, S.Tsujii, “Globally Converging Learning of Multilayer Perceptrons”, Proc. of Int. Joint conf. on Neural Networks, Vol.2, p.1079–1089, 1991.