

Support Vector Machines for Direction of Arrival Estimation

Judd A. Rohwer* and Chaouki T. Abdallah†

1 Abstract

Machine learning research has largely been devoted to binary and multiclass problems relating to data mining, text categorization, and pattern/facial recognition. Recently, popular machine learning algorithms have successfully been applied to wireless communication problems, notably spread spectrum receiver design, channel equalization, and adaptive beamforming with direction of arrival estimation (DOA). Various neural network algorithms have been widely applied to these three communication topics. New advanced learning techniques, such as support vector machine (SVM) have been applied, in the binary case, to receiver design and channel equalization. This paper presents a multiclass implementation of SVMs for DOA estimation and adaptive beamforming, an important component of code division multiple access (CDMA) communication systems.

2 Introduction

Machine learning techniques have been applied to various problems relating to cellular communications. In our research we present a machine learning based approach for DOA estimation in a CDMA communication system [1]. The DOA estimates are used in adaptive beamforming for interference suppression, a critical component in cellular systems. Interference suppression reduces the multiple access interference (MAI) which lowers the required transmit power. The interference suppression capability directly influences the cellular system capacity, i.e., the number of active mobile subscribers per cell.

Beamforming, tracking, and DOA estimation are current research topics with various technical approaches. Least mean square estimation, Kalman filtering, and neural networks [2],[3],[4], have been successfully applied to these problems.

*Sandia National Laboratories P.O. Box 5800, MS-0986 Albuquerque, NM 87185-0986: jarohwe@sandia.gov

†C.T. Abdallah is with the Department of Electrical and Computer Engineering, University of New Mexico, Albuquerque, NM 87131, USA; chaouki@eece.unm.edu

⁰Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy under Contract DE-AC04-94AL85000.

Many approaches have been developed for calculating the DOA; three techniques based on signal subspace decomposition are ESPRIT, MUSIC, and Root-MUSIC [1].

Adaptive antenna arrays are critical components of a wireless communication system. System designers utilize state of the art signal processing techniques for adaptive beamforming and interference suppression. Research in this area shows that it is possible to direct the maximum antenna gain towards the desired signal and to effectively place nulls in the direction of interfering users. By adaptively updating the antenna array weights the desired signal can be tracked, keeping continuous coverage and maintaining the largest possible signal-to-interference (SIR) ratio. The adaptive antenna arrays must be able to process multiple DOAs of the desired signal.

Neural networks have been successfully applied to the problem of DOA estimation and adaptive beamforming in [4], [5], [6]. New machine learning techniques, such as support vector machines (SVM) and boosting, perform exceptionally well in multiclass problems and new optimization techniques are published regularly. These new machine learning techniques have the potential to exceed the performance of the neural network algorithms relating to communication applications. Specifically the techniques applied DOA estimation and adaptive beamforming will increase the speed of convergence to the desired antenna array weights and decrease the time necessary for system training.

In our research machine learning is applied to two areas of adaptive antenna arrays; the estimation of the array weight vectors, $\widehat{\mathbf{W}}_{i,r}$, and the DOA estimates, $\widehat{\boldsymbol{\theta}}_r$. The goal is to minimize the interference which in turn will improve the quality of service (QoS) and increase cellular system capacity. To reduce the interference power the adaptive array system focuses the beam towards the desired user and effectively places nulls in the direction of the interferers. In a CDMA system the beamwidth, maximum directivity, and nulls do not require extreme accuracy, as would be required for frequency division multiple access (FDMA) systems. The machine learning methods presented in this paper include subspace based estimation applied to the sample covariance matrix of the received signal. The optimization techniques use both training data and received data to generate the DOA estimates and antenna weight vectors. The end result is an efficient machine learning approach to finding the DOA estimates which are applied to a Rake receiver based CDMA cellular architecture [1]. This cellular base station design with Rake receivers maximizes the received SIR, reduces multiple access interference (MIA), and thus reduces the required transmit power from the mobile subscriber. If a system includes R Rake fingers then the machine learning algorithms must estimate the L dominated signal paths for each Rake finger.

Many computational techniques exist for working through limitations of DOA estimation techniques, but currently no techniques exist for a system level approach to accurately estimating the DOAs at the base station. A number of limitations relating to popular DOA estimation techniques are: 1) the signal subspace dimension is not known, many papers assume that it is. The differ-

ences in eigenvalues between the covariance matrix and the sample covariance matrix add to the uncertainty, 2) searching all possible angles to determine the maximum response of the MUSIC algorithm, 3) evaluating the Root-MUSIC polynomial on the unit circle, 4) multiple eigen decompositions for ESPRIT, 5) computational complexity for maximum likelihood method.

This paper is organized as follows. Section 3 presents the system models for an adaptive antenna array CDMA systems. A review of machine learning is presented in Section 4. In Section 5 we present background information on binary and multiclass SVMs. Finally, Section 6 presents a multiclass SVM algorithm for DOA estimation and simulation results.

3 System Models

This section includes an overview of system models for the received signal and adaptive antenna arrays designs. All notation is described below and is consistently used throughout the paper.

3.0.1 Received Signal at Antenna Array output

The signal from antenna array A , detected at the adaptive array processor is

$$\mathbf{x}_A(t) = \sum_{j=1}^J \sum_{l=1}^L \sqrt{p_{t_j}(t) g_{ji}(t) \bar{\mathbf{a}}(\theta_l) q_j^l} s_j(t - \tau_j) \cos(w_c(t - \tau_j)) + n_j(t). \quad (1)$$

$\mathbf{x}_A(t)$ is the signal vector from antenna array A

$p_{t_j}(t)$ is the transmit signal power from mobile j .

$g_{ji}(t)$ is the link gain from base j to mobile i .

q_j^l is the attenuation due to shadowing from path l .

$\bar{\mathbf{a}}(\theta_l) = [1 \ e^{-jk_l} \ \dots \ e^{-j(D-1)k_l}]^T$ is the $D \times 1$ steering vector.

$k_l = \frac{\nu w_o}{c} \sin \theta_l$

θ_l is the direction of arrival of the l signal.

D is the number of elements in the array.

$s_j(t - \tau_j)$ is the CDMA spreading code.

τ_j is the propagation delay from mobile j to the base station.

$n_j(t)$ is the additive white Gaussian noise for mobile j 's received signal.

J is the number of received signals; One desired and $J - 1$ interfering signals.

L is the number of transmission paths.

The spreading code is generated by

$$s_j(t - \tau_j) = \sum_{n=1}^{S_c} b_j(n) c_j(t - nT). \quad (2)$$

$b_j(t)$ is the baseband signal of mobile j .

$c_j(t)$ is the CDMA spreading code of mobile j .

S_c is the number of chips in the spreading code.

T is the time period of one chip.

To ease the complexity of the notation the terms relative to the multiple paths are combined as

$$\mathbf{z}_j = \sum_{l=1}^L \bar{\mathbf{a}}(\theta_l) q_j^l. \quad (3)$$

In [7] \mathbf{z}_j is defined as the spatial signature of the antenna array to the j^{th} source. The signal vector from the antenna array is rewritten as

$$\mathbf{x}_A(t) = \sum_{j=1}^J \sqrt{p_{t_j}(t) g_{j_i}(t)} \mathbf{z}_j s_j(t - \tau_j) \cos(w_c(t - \tau_j)) + n_j(t). \quad (4)$$

3.0.2 Received Signal at Rake Receivers

The received signal at the input to the Rake receiver is

$$x_r(t) = \mathbf{W}_{i,r} \mathbf{x}_A(t). \quad (5)$$

$\mathbf{W}_{i,r}$ is the antenna array weight vector for signal/mobile i at Rake finger r , $0 < r \leq R$, R is the number of Rake fingers in the system. Refer to Figure 1. From equation (1) the carrier and spreading sequence are removed by the

Figure 1: Rake Receiver with Adaptive Antenna Array

operations within the Rake receiver: mixing, lowpass filtering, and correlation. The average output power from the Rake finger, r , is shown in equation (6). Note that the received continuous time signal is sampled and is represented as a discrete signal.

$$\Phi_{r,r} = \mathbb{E} \{ \mathbf{W}_{i,r}^H x_r(n) x_r^H(n) \mathbf{W}_{i,r} \} \quad (6)$$

$$= \mathbf{W}_{i,r}^H \mathbb{E} \{ x_r(n) x_r^H(n) \} \mathbf{W}_{i,r} \quad (7)$$

$$= \mathbf{W}_{i,r}^H \phi_{r,r}^2 \mathbf{W}_{i,r} \quad (8)$$

$\mathbf{W}_{i,r}^H$ is the Hermitian transpose of $\mathbf{W}_{i,r}$, $x_r^H(n)$ is the Hermitian transpose of $x_r(n)$. In equation (8) $\phi_{r,r}^2$ is the autocorrelation of the received signal and is defined below.

$$\phi_{r,r}^2 = \phi_{s,r}^2 + \phi_{I+N,r}^2 \quad (9)$$

$$\phi_{s,r}^2 = p_{t_i} g_{ii} \mathbf{z}_i \mathbf{z}_i^H \quad (10)$$

$$\phi_{I+N,r}^2 = \sum_{j \neq i}^J p_{t_j} g_{ji} \mathbf{z}_j \mathbf{z}_j^H + n_i \mathbf{I} \quad (11)$$

The received power is rewritten in equations (12) - (15).

$$\Phi_{r,r} = \mathbf{W}_{i,r}^H (\phi_{s,r}^2 + \phi_{I+N,r}^2) \mathbf{W}_{i,r} \quad (12)$$

$$= \mathbf{W}_{i,r}^H \phi_{s,r}^2 \mathbf{W}_{i,r} + \mathbf{W}_{i,r}^H \phi_{I+N,r}^2 \mathbf{W}_{i,r} \quad (13)$$

$$= p_{t_i} g_{ii}^H \mathbf{W}_{i,r}^H \mathbf{z}_i \mathbf{z}_i^H \mathbf{W}_{i,r} + \sum_{j \neq i}^J p_{t_j} g_{ji}^H \mathbf{W}_{i,r}^H \mathbf{z}_j \mathbf{z}_j^H \mathbf{W}_{i,r} + n_i \mathbf{W}_{i,r}^H \mathbf{W}_{i,r} \quad (14)$$

$$\Phi_{r,r} = \Phi_{s,r} + \Phi_{I+N,r} \quad (15)$$

From the above equations the SIR at the output of each Rake receiver is

$$SIR_{i,r} = \gamma_{i,r} = \frac{\Phi_{s,r}}{\Phi_{I+N,r}} = \frac{p_{t_i} g_{ii} \mathbf{W}_{i,r}^H \mathbf{z}_i \mathbf{z}_i^H \mathbf{W}_{i,r}}{\sum_{j \neq i}^J p_{t_j} g_{ji} \mathbf{W}_{i,r}^H \mathbf{z}_j \mathbf{z}_j^H \mathbf{W}_{i,r} + n_i \mathbf{W}_{i,r}^H \mathbf{W}_{i,r}}. \quad (16)$$

The composite SIR_i is taken at the output of the diversity combiner, in general for an equal gain combiner $SIR_i = \sum_{r=1}^R SIR_{i,r}$. Refer to [1] for information on various diversity combining techniques. The goal of adaptive beamforming is to maximize the power of the received signal, $\Phi_{s,r}$, and minimize the interference and noise power, $\Phi_{I+N,r}$, by finding the optimum antenna array weights $\mathbf{W}_{i,r}$ for each rake receiver.

3.0.3 Machine Learning for DOA Estimation

From the system model in Section 3.0.2 the solution for the optimal weight vector [7], [8], is

$$\widehat{\mathbf{W}}_{i,r} = \frac{\Phi_{I+N,r}^{-1} \mathbf{z}_i}{\mathbf{z}_i^H \Phi_{I+N,r}^{-1} \mathbf{z}_i} \quad (17)$$

With principal component analysis and machine learning we can estimate the interference, $\Phi_{I+N,r}$, and the array response, $\mathbf{z}_j = \sum_{l=1}^L \bar{\mathbf{a}}(\theta_l) q_j^l$.

To estimate the array response, \mathbf{z}_i , we must know $\bar{\mathbf{a}}(\theta_l)$ and q_i^l . The continuous pilot signal, included in cdma2000, can be used in estimating q_i^l . This must be done for each resolvable path, i.e., $q_i = [q_i^1, q_i^2, \dots, q_i^L]$. Estimating $\bar{\mathbf{A}}(\theta) = [\bar{\mathbf{a}}(\theta_1), \bar{\mathbf{a}}(\theta_2), \dots, \bar{\mathbf{a}}(\theta_L)]$ requires information on the DOA.

The process of DOA estimation is to monitor the outputs of D antenna elements and predict the angle of arrival of L signals, $L < D$. The output matrix from the antenna elements is

$$\begin{aligned}\bar{\mathbf{A}} &= [\bar{\mathbf{a}}(\theta_1) \quad \bar{\mathbf{a}}(\theta_2) \quad \dots \quad \bar{\mathbf{a}}(\theta_L)] & (18) \\ \bar{\mathbf{a}}(\theta_l) &= [1 \quad e^{-jk_l} \quad e^{-j2k_l} \quad \dots \quad e^{-j(D-1)k_l}]^T, & (19)\end{aligned}$$

and the vector of incident signals is $\boldsymbol{\theta}_r = [\theta_1, \theta_2, \dots, \theta_L]$. With a training process the learning algorithms generate DOA estimates, $\hat{\boldsymbol{\theta}}_r = [\hat{\theta}_1, \hat{\theta}_2, \dots, \hat{\theta}_L]$, based on the responses from the antenna elements, $\bar{\mathbf{a}}(\theta_l)$.

4 Machine Learning Background

Machine learning has already made an impact in the analysis and design of communication systems. Neural Networks are applied to numerous problems, ranging from adaptive antenna arrays [4], multiuser receiver design [9],[10], interference suppression [11], and power prediction [12]. Designs with SVMs are starting to appear in the journals [13],[14]. Boosting algorithms have been applied to standard classification problems, such as text and image classification, but have yet to be applied to specific communication problems.

Machine learning is the process of observing input data and applying classification rules to generate a binary or multiclass label to the output. In the binary case a classification function is estimated using input/output training pairs with unknown probability distribution, $P(\mathbf{x}, y)$, \mathbf{x} is sample vector of observations, and

$$\begin{aligned}(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_n, y_n) &\in \mathbb{R}^N \times Y, & (20) \\ y_i &= \{-1, +1\}. & (21)\end{aligned}$$

The estimated classification function maps the input to a binary output, $f : \mathbb{R}^N \rightarrow \{-1, +1\}$. The system is first trained with the input/output data pairs then the test data, from the same probability distribution $P(\mathbf{x}, y)$, is applied to the classification function. The binary output label, $+1$, is generated if $f(\mathbf{x}) \geq 0$, likewise -1 is the output label if $f(\mathbf{x}) < 0$. For the multiclass case $Y \in \mathbb{R}^G$ where Y is a finite set of real numbers and G is the size of the multiclass label set. The objective is to estimate the function which maps the input data to a finite set of output labels $f : \mathbb{R}^N \rightarrow \mathbb{G}(\mathbb{R}^N) \in \mathbb{R}^G$

Estimating the classification function is approached by minimizing the expected risk [15]. The risk is defined as

$$R[f] = \int L(f(\mathbf{x}), y) dP(\mathbf{x}, y). \quad (22)$$

The loss function, L , is explained in [15]. Since the probability distribution of the input data is unknown the classification function must be estimated. The

estimation process is based on empirical risk minimization.

$$R_{emp}[f] = \frac{1}{n} \sum_{i=1}^l (f(\mathbf{x}_i), y_i) \quad (23)$$

By setting conditions of the minimization routine the empirical risk converges towards the expected risk. Minimization routines must be carefully selected so the generalization (test) error closely tracks the training error. Overfitting occurs when small sample (test) sizes generate large deviations in the empirical risk; the problem of overfitting can be minimized by restricting the complexity of the function class from which the classification function is chosen. For a detailed synopsis of this area of research review the Vapnik-Chervonenkis (VC) theory and structural risk minimization (SRM) [16].

The general process of SVM algorithms is to project the input space to a higher feature space via a nonlinear mapping,

$$\Gamma : \mathbb{R}^N \rightarrow F \quad (24)$$

$$\mathbf{x} \mapsto \Gamma(\mathbf{x}). \quad (25)$$

The input data $\mathbf{x}_1, \dots, \mathbf{x}_n \in \mathbb{R}^N$ is mapped into a new feature space F which could have a much higher dimensionality. The data in the new feature space is then applied to the desired machine learning algorithm. For the binary case the input/output pairs in new feature space are described as

$$(\Gamma(\mathbf{x}_1), y_1), \dots, (\Gamma(\mathbf{x}_n), y_n) \in F \times Y. \quad (26)$$

The background of machine learning shows that the dimension of the feature space is not as important as the complexity of the classification functions. For example in the input space the input/output pairs may be represented with a nonlinear function, but in a higher dimension feature space the input/output pairs may be separated with a linear hyperplane.

4.1 Kernel Functions

Kernel functions are used to compute the scalar dot products of the input/output pairs in the feature space F . Without kernel functions it might be impossible to perform scalar operations in the higher dimensional feature space. Essentially, an algorithm in the input space can be applied to the data in the feature space.

$$\Gamma(\mathbf{x}) \cdot \Gamma(\mathbf{y}) = \mathbb{k}(\mathbf{x}, \mathbf{y}) \quad (27)$$

Therefore a linear algorithm in the feature space corresponds to a nonlinear algorithm, such as the classification functions, in the input space. This allows a decision rule to be applied to the inner product of training points and test points in the feature space.

Three popular kernel functions are the linear kernel, polynomial kernel, radial basis function (RBF), and multilayer perceptrons (MLP).

$$\text{linear} \quad \rightarrow \quad \mathbb{k}(\mathbf{x}, \mathbf{y}) = \mathbf{x} \cdot \mathbf{y} \quad (28)$$

$$\text{polynomial of degree } \mathbf{d} \quad \rightarrow \quad \mathbb{k}(\mathbf{x}, \mathbf{y}) = ((\mathbf{x} \cdot \mathbf{y}) + \boldsymbol{\theta})^{\mathbf{d}} \quad (29)$$

$$\text{RBF} \quad \rightarrow \quad \mathbb{k}(\mathbf{x}, \mathbf{y}) = \exp\left(\frac{-\|\mathbf{x} - \mathbf{y}\|^2}{\sigma^2}\right) \quad (30)$$

$$\text{MLP} \quad \rightarrow \quad \mathbb{k}(\mathbf{x}, \mathbf{y}) = \tanh(\kappa(\mathbf{x} \cdot \mathbf{y}) + \boldsymbol{\theta}) \quad (31)$$

The performance of each kernel function varies with the characteristics of the input data. Refer to [17] for more information on feature spaces and kernel methods.

5 Support Vector Machines - Background

SVMs were originally designed for the binary classification problem. A variety of approaches are currently being developed to tackle the problem of applying SVMs to multiclass problems. Much like all machine learning algorithms SVMs find a classification function that separates the hyperplane with the largest margin. This is the difference between all machine learning algorithms, the mathematical operations involved in calculating the optimal separating hyperplane. The SVM maps an inner product of the input space into a higher dimensional feature space via a kernel operation. The projected data does not have the full dimensionality of the feature space since the mapping process is to a non-unique generalized surface [14]. The data points near the optimal hyperplane are the “support vectors” and serve as the basis of the feature space. Therefore SVMs are a nonparametric machine learning algorithm with the capability of controlling the capacity through the support vectors.

5.1 Binary Classification

In binary classification system the machine learning algorithm produces estimates with a hyperplane separation, i.e., $y_i \in [-1, 1]$ represents the classification “label” of the input vector \mathbf{x} . The input sequence and a set of training labels are represented as $\{\mathbf{x}_k, y_k\}_{k=1}^K$, $y_k = \{-1, +1\}$. If the two classes are linearly separable in the input space then the the hyperplane is defined as $\mathbf{w}\mathbf{x} + b = 0$, \mathbf{w} is a vector of weights and b is a bias term, if the input space is projected to a higher dimensional feature space then the hyperplane becomes $\mathbf{w}\boldsymbol{\Gamma}(\mathbf{x}) + b = 0$. The nonlinear function $\boldsymbol{\Gamma}(\cdot) : \mathbb{R}^N \rightarrow \mathbb{R}^{N'}$ maps the input space to the feature space.

5.1.1 Support Vector Machines

The SVM algorithm is based on the assumption [18] that

$$\mathbf{w}^T \mathbf{\Gamma}(\mathbf{x}_k) + b \geq 1, \text{ if } y_k = +1, \quad (32)$$

$$\mathbf{w}^T \mathbf{\Gamma}(\mathbf{x}_k) + b \leq -1, \text{ if } y_k = -1. \quad (33)$$

This formulation is restated as $y_k [\mathbf{w}^T \mathbf{\Gamma}(\mathbf{x}_k) + b] \geq 1, k = 1, \dots, K$.

The SVM optimization is defined as

$$\min_{\mathbf{w}, b, \phi} \mathcal{L}(\mathbf{w}, \phi) = \frac{1}{2} \|\mathbf{w}\|^2 + c \sum_{k=1}^K \phi_k, \text{ with constraints} \quad (34)$$

$$y_k [\mathbf{w}^T \mathbf{\Gamma}(\mathbf{x}_k) + b] \geq 1 - \phi_k, \quad k = 1, \dots, K \quad (35)$$

$$\phi_k \geq 0, \quad k = 1, \dots, K. \quad (36)$$

Misclassifications, due to overlapping distributions, are accounted for with the slack variables ϕ_k , c is a tuning parameter. The margin between the hyperplane and the data points in the feature space is maximize when \mathbf{w} is minimized. The solution to the optimization problem in (34) is given by the saddle point of the Lagrangian function,

$$\mathcal{Z}(\mathbf{w}, b, \phi, \alpha, \varepsilon) = \mathcal{L}(\mathbf{w}, \phi) - \sum_{k=1}^K \alpha_k \{y_k [\mathbf{w}^T \mathbf{\Gamma}(\mathbf{x}_k) + b] - 1 + \phi_k\} - \sum_{k=1}^K \varepsilon_k \phi_k \quad (37)$$

where $\alpha_k \geq 0$ and $\varepsilon_k \geq 0$ are Lagrangian multipliers. By computing

$$\max_{\alpha, \varepsilon} \min_{\mathbf{w}, b, \phi} \mathcal{Z}(\mathbf{w}, b, \phi, \alpha, \varepsilon), \quad (38)$$

the dual Lagrangian problem is developed; differentiating with respect to \mathbf{w}, b, ϕ , [17], [18] leads to

$$\frac{d\mathcal{Z}}{d\mathbf{w}} = 0, \quad \mathbf{w} = \sum_{k=1}^K \alpha_k y_k \mathbf{\Gamma}(\mathbf{x}_k) \quad (39)$$

$$\frac{d\mathcal{Z}}{db} = 0, \quad \sum_{k=1}^K \alpha_k y_k = 0 \quad (40)$$

$$\frac{d\mathcal{Z}}{d\phi} = 0, \quad 0 \leq \alpha_k \leq c, \quad k = 1, \dots, K. \quad (41)$$

The classic quadratic programming problem is developed by replacing \mathbf{w} in the Lagrangian:

$$\max_{\alpha} \Omega(\alpha) = -\frac{1}{2} \sum_{k,j=1}^K \alpha_k \alpha_j y_k y_j \mathbb{k}(\mathbf{x}_k, \mathbf{x}_j) + \sum_{k=1}^K \alpha_k, \text{ such that} \quad (42)$$

$$\sum_{k=1}^K \alpha_k y_k = 0, \quad 0 \leq \alpha_k \leq c, \quad k = 1, \dots, K. \quad (43)$$

The kernel function, described in Section 4.1, is $\mathbb{k}(\mathbf{x}_k, \mathbf{x}_j) = \Gamma(\mathbf{x}_k) \cdot \Gamma(\mathbf{x}_j)$. The kernel should be chosen in order to eliminate the need to calculate \mathbf{w} and $\Gamma(\mathbf{x})$. From the developments above the nonlinear SVM is defined as

$$y(\mathbf{x}) = \text{sign} \left[\sum_{k=1}^K \alpha_k y_k \mathbb{k}(\mathbf{x}, \mathbf{x}_k) + b \right]. \quad (44)$$

The non-zero α'_k s are “support values” and the corresponding data points are the “support vectors”. The support vectors are located close to the hyperplane boundary. The test data \mathbf{x} is projected onto the training vectors \mathbf{x}_k . The summation of the multivariable product of the support values, binary labels and the feature space projection products the corresponding test label. This SVM binary classification algorithm is used to produce larger multiclass classification algorithms.

5.1.2 Sequential Minimal Optimization (SMO)

Numerous algorithms have been developed for training SVMs. The SMO algorithm is an optimization routine for the quadratic programming (QP) problem, equation (42), where the large scale QP problem is decomposed into smaller, manageable QP problems. The small QP problems are solved analytically versus large scale QP optimization routines. This approach promotes the sparseness of the data sets, created by the zero-valued support vectors. SMOs have minimum training time which is linear with respect to the training set size, the maximum training time is quadratic with the training set size. This can be orders of magnitude less than other training algorithms presented in current research, such as the projected conjugate gradient (PCG) chunking method and SVM^{light} [19],[20]. The SMO algorithm reduces the complexity and improves the training time of SVMs, all of which makes SVMs more user friendly and attractive to many applications. Refer to [19] for pseudo code of SMO algorithm.

5.1.3 Least Squares SVM

Suykens, et.al., [21] introduced a least squares SVM (LS-SVM) which is based on the Vapnik SVM classifier discussed in Section 5.1.1 and repeated below in equation (45).

$$y(\mathbf{x}) = \text{sign} \left[\sum_{k=1}^K \alpha_k y_k \mathbb{k}(\mathbf{x}, \mathbf{x}_k) + b \right] \quad (45)$$

The LS-SVM classifier is generated from the optimization problem:

$$\min_{\mathbf{w}, b, \phi} \mathcal{L}_{LS}(\mathbf{w}, b, \phi) = \frac{1}{2} \|\mathbf{w}\|^2 + \frac{1}{2} \psi \sum_{k=1}^K \phi_k^2, \text{ with constraints} \quad (46)$$

$$y_k [\mathbf{w}^T \Gamma(\mathbf{x}_k) + b] \geq 1 - \phi_k, \quad k = 1, \dots, K \quad (47)$$

The Lagrangian of equation (46) is defined as

$$\mathcal{Z}_{LS}(\mathbf{w}, b, \phi, \alpha) = \mathcal{L}_{LS}(\mathbf{w}, b, \phi) - \sum_{k=1}^K \alpha_k \{y_k [\mathbf{w}^T \mathbf{\Gamma}(\mathbf{x}_k) + b] - 1 + \phi_k\} \quad (48)$$

where α_k are Lagrangian multipliers that can either be positive or negative. The conditions of optimality are similar to those in equations (39) – (41), but have updated for the LS approach.

$$\frac{d\mathcal{Z}_{LS}}{d\mathbf{w}} = 0, \quad \mathbf{w} = \sum_{k=1}^K \alpha_k y_k \mathbf{\Gamma}(\mathbf{x}_k) \quad (49)$$

$$\frac{d\mathcal{Z}_{LS}}{db} = 0, \quad \sum_{k=1}^K \alpha_k y_k = 0 \quad (50)$$

$$\frac{d\mathcal{Z}_{LS}}{d\phi} = 0, \quad \alpha_k = \psi \phi_k \quad (51)$$

$$\frac{d\mathcal{Z}_{LS}}{d\alpha_k} = 0, \quad y_k [\mathbf{w}^T \mathbf{\Gamma}(\mathbf{x}_k) + b] - 1 + \phi_k = 0 \quad (52)$$

These conditions can be written as a linear system [21].

$$\begin{bmatrix} I & 0 & 0 & -Z^T \\ 0 & 0 & 0 & -Y^T \\ 0 & 0 & \psi I & -I \\ Z & Y & I & 0 \end{bmatrix} \begin{bmatrix} \mathbf{w} \\ b \\ \phi \\ \alpha \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \vec{1} \end{bmatrix} \quad (53)$$

$$Z = [\mathbf{\Gamma}(\mathbf{x}_1)^T y_1, \dots, \mathbf{\Gamma}(\mathbf{x}_K)^T y_K] \quad (54)$$

$$Y = [y_1, \dots, y_K], \quad \vec{1} = [1, \dots, 1] \quad (55)$$

$$\phi = [\phi_1, \dots, \phi_K], \quad \alpha = [\alpha_1, \dots, \alpha_K] \quad (56)$$

By eliminating weight vector \mathbf{w} and the slack variable ϕ the linear system is reduced to:

$$\begin{bmatrix} 0 & Y^T \\ Y & ZZ^T + \psi^{-1}I \end{bmatrix} \begin{bmatrix} b \\ \alpha \end{bmatrix} = \begin{bmatrix} 0 \\ \vec{1} \end{bmatrix} \quad (57)$$

In the linear systems defined in (53)–(57) the support values α_k are proportional to the errors at the data points. In the standard SVM case many of these support values are zero. Because most, if not all, of the support values are non-zero, finite, the LS-SVM case has a support value spectrum. Since the matrix for the LS-SVM linear system is $(K + 1) \times (K + 1)$ an iterative solution is required. In [21] a conjugate gradient method is proposed for solving b and α , which are required for the SVM classifier in equation (45). Pseudo algorithms for the implementation of the LS-SVM algorithm are also presented.

5.2 Multiclass Classification

There exist many SVM approaches to multiclass classification problem. Two primary techniques are one-vs-one and one-vs-rest. One-vs-one applies SVMs to selected pairs of classes. For P distinct classes there are $\frac{P(P-1)}{2}$ hyperplanes that separate the classes with maximum margin. The one-vs-rest SVM technique generates P hyperplanes that separate each distinct class from the ensemble of the rest. In this paper we only consider the one-vs-one multiclass SVM.

For the multiclass problem the machine learning algorithm produces estimates with multiple hyperplane separations. The set of input vectors and training labels is defined as $\{\mathbf{x}_n, y_n^g\}_{n=1, g=1}^{n=N, g=G}$, $\mathbf{x}_n \in \mathbb{R}^N, n = 1, \dots, N$, $y_i \in \{1, \dots, G\}$, n is the index of the training pattern and G is the number of classes.

One-vs-one multiclass classification is based on the binary SVMs discussed in Sections 5.1 and 5.1.3. In the training phase the margins for $\frac{P(P-1)}{2}$ hyperplanes are constructed. The basic approach uses a tree structure to compare the test data to each of the $\frac{P(P-1)}{2}$ hyperplanes. Through a series of elimination steps the best label is assigned to the input data. The Decision Directed Acyclic Graph (DDAG) and DAGSVM are specific techniques for one-vs-one multiclass classification; a review of each is included below.

5.2.1 DDAG and DAGSVM

Platt, et.al., [22] introduced the DDAG, a VC analysis of the margins, and the development of the DAGSVM algorithm. The two techniques are based on $\frac{P(P-1)}{2}$ classifiers for a P class problem, one classifier for each pair of classes. The DAGSVM includes an efficient one-vs-one SVM implementation that allows for faster training than the standard one-vs-one algorithm and the one-vs-rest approach.

The DDAG algorithm includes $\frac{P(P-1)}{2}$ nodes, each associated with a one-vs-one classifier and its respective hyperplane. The test error of the DDAG depends on the number of classes, P , and the margins at each node. In [22] it is proved that maximizing the margins at each node of the DDAG will minimize the generalization error, independently of the dimension of the input space. Likewise, the input data is projected to a higher dimension feature space using appropriate kernel functions.

The DAGSVM algorithm is based on the DDAG architecture with each node containing a binary SVM classifier of the i^{th} and j^{th} classes. The training time of the DAGSVM classifier is equivalent to the standard one-vs-one SVMs. The performance benefit of the DAGSVM is realized when the i^{th} classifier is selected at the i^{th}/j^{th} node and the j^{th} class is eliminated. Thus any other class pairs containing the j^{th} class are removed from the remaining SVM operations and the j^{th} class is not a candidate for the output label. Refer to Figure 2 for a diagram of the DAGSVM approach. An analysis of the training times for one-vs-rest, one-vs-one, and the DAGSVM with SMO are presented in [22].

Figure 2: DAGSVM for Four Classes

6 SVMs and DOA Estimation

Interference suppression, with adaptive antenna arrays, requires estimating the optimal antenna weight vector (17) and the DOAs of the dominant signal paths (18). Eigen decomposition or principal component analysis (PCA) can be used to generate the interference and noise power estimates, $\Phi_{I+N,r}$, and the eigenvectors of the signal covariance matrix. In this paper we propose a multiclass SVM algorithm trained with projection vectors generated from the signal subspace eigenvectors and the sample covariance matrix. The output labels from the SVM system are the DOA estimates.

Two one-vs-one multiclass SVM techniques for DOA estimation are presented in this section, DAGSVM with SMO and LS-SVM. Each algorithm is trained for P DOA classes. The number of classes is dependent upon on the antenna sectoring and required resolution. For a CDMA system the desired interference suppression dictates the fixed beamwidth of X degrees. CDMA offers this flexibility since the all mobiles use the same carrier frequency. For FDMA systems a narrow beamwidth is desired, since frequency reuse determines the capacity of a cellular system.

6.1 Preprocessing

The signal subspace eigenvectors of the received signal covariance matrix are required for accurate DOA estimation. For a CDMA system with adaptive

antenna arrays, refer to Section 3.0.2, the covariance matrix of the received signal is

$$\bar{\mathbf{R}}_{rr} = \mathbb{E} [\mathbf{x}_r \mathbf{x}_r^H], \quad (58)$$

where $\mathbf{x}_r \in C^n$ is a complex random vector process.

In our machine learning based DOA estimation algorithm the principal eigenvectors must be calculated. Eigen decomposition (ED) is the standard computational approach for calculating the eigenvalues and eigenvectors of a the covariance matrix. ED is a computationally intense technique, faster algorithms, such as PASTd [23] with $4NL+O(L)$ computations, have been developed for real time processing applications; L is the dimension of the desired signal subspace and N is the dimension of the input vector.

For a machine learning based approach to DOA estimation the output of the Rake receiver is used to calculate the sample covariance matrix $\hat{\mathbf{R}}_{rr}$,

$$\hat{\mathbf{R}}_{rr} = \frac{1}{M} \sum_{k=K-M+1}^K \mathbf{x}_r(k) \mathbf{x}_r^H(k) \quad (59)$$

The dimension of the observation matrix is $D \times M$ and the dimension of the sample covariance matrix is $D \times D$. D is the number of antenna elements and M is ideal sample size (window length) which must be determined through testing.

6.1.1 Algorithms for DOA Estimation

Two primary, classic methods for subspace based DOA estimation exist in literature, Multiple Signal Classification (MUSIC) [24] and Estimation of Signal Parameters Via Rotational Invariance Techniques (ESPRIT) [25]. The MUSIC algorithm is based on the noise subspace and ESPRIT is based on the signal subspace. There are implementation issues involved with each approach; accurate array characterization is required for MUSIC and multiple eigen decompositions are required for ESPRIT.

Figure 3 includes a plot showing DOA estimation results with the MUSIC algorithm. DOA estimates for ten subspace updates are plotted. The antenna array consists of eight elements and the input signal contains one distinct signals with a DOA at 25° . These results serve as a benchmark for the machine learning based DOA estimation. For the proposed machine learning technique there is a trade-off between the accuracy of the DOA estimation and antenna array beamwidth. An increase in DOA estimation accuracy translates into a smaller beamwidth and a reduction in MAI. Therefore the accuracy in DOA estimation directly influences the minimum required power transmitted by the mobile. There should be a balance between computing effort and reduction in MAI.

6.2 Support Vectors for Multiclassification of DOAs

The multiclass SVM implemented with a DDAG is the primary technique presented for DOA estimation. The DDAG tree is initialized for P classes with

Figure 3: DOA Estimation with MUSIC and Eigendecomposition.

$\frac{P(P-1)}{2}$ nodes. Therefore $\frac{P(P-1)}{2}$ one-vs-one SVMs are trained to generate the hyperplanes with maximum margin at each node. For each class the training vectors, \mathbf{x}_n , are generated from the eigenvectors spanning the signal subspace.

6.2.1 LS-SVM DDAG with Projection Vectors

The proposed technique for machine learning based DOA estimation is the LS-SVM algorithm applied to the DDAG decision tree. Each DOA class consists of a space of DOAs determined by the desired resolution. For example, if there are P desired classes and a 90° antenna sector, then each DOA class would include a DOA region of $X = \frac{90^\circ}{P}$ degrees.

The LS-SVM approach to DOA estimation uses projection vectors generated from the projection of $\hat{\mathbf{R}}_{rr}$ onto the primary eigenvector of the signal subspace. In the training phase the hyperplanes at each DDAG node are constructed with these projection vectors. In the testing phase $\hat{\mathbf{R}}_{rr}$ is generated from the received signal $\mathbf{x}_r(k)$, refer to (59). Then the projection coefficients for the i^{th}/j^{th} node of the DDAG are computed with dot products of $\hat{\mathbf{R}}_{rr}$ and the i^{th}/j^{th} training eigenvectors. This new set of projection vectors is testing with the i^{th}/j^{th} hyperplane generated during the training phase. The DOA labels are then assigned based on the DDAG evaluation path. A similar projection coefficient technique has been successfully applied to a multiclass SVM facial recognition problem presented in [26].

6.2.2 LS-SVM DDAG based DOA Estimation Algorithm

The following routine is applied to each Rake finger, if there are R fingers in the Rake receiver, then there will be R parallel implementations.

- Preprocessing
 1. Generate the $D \times N$ training signal vectors for the P SVM classes, D is the number of antenna elements, N is the number of samples.
 2. Generate the P sample covariance matrices, \mathbb{M} , with M samples from the $D \times N$ data vector
 3. Calculate the signal eigenvector, \mathbb{S} , from each of the P sample covariance matrices.
 4. Calculate the $D \times 1$ projection vectors, $\mathbb{M} \times \mathbb{S}$, for each of the P classes. The ensemble of projection vectors consists of $\frac{N}{M}$ samples
 5. Store the projection vectors for the training phase and the eigenvectors for the testing phase.
- LS-SVM training
 1. With the P projection vectors train the $\frac{P(P-1)}{2}$ nodes with the one-vs-one LS-SVM algorithm. .
 2. Store the LS-SVM variables, α_k and b from equation (45), which define the hyperplane separation for each DDAG node.
- LS-SVM testing for the i^{th}/j^{th} node DDAG node
 1. Acquire $D \times N$ input signal from antenna array, this signal has an unknown DOA.
 2. Generate the P sample covariance matrices with M samples from the $D \times N$ data vector.
 3. Calculate two $D \times 1$ projection vectors with the i^{th} and j^{th} eigenvectors from the preprocessing steps.
 4. Test both projection vectors against the LS-SVM hyperplane for the i^{th}/j^{th} node. This requires two separate LS-SVM testing cycles, one with the projection vector from the i^{th} eigenvector and one with the projection vector from the j^{th} eigenvector.
 5. Calculate the average value of the two LS-SVM output vectors (labels). Select the value which is closest to the correct label, 0/1.
 6. Repeat process for the next DDAG node in the evaluation path or declare the final DOA label.

6.3 Simulation Results

Two simulation plots are included below. Each simulation consists of a four class LS-SVM DDAG system. Figure 4 shows results for a ten degree range per class. Figure 5 shows results for a one degree range per class. Testing shows that the LS-SVM DDAG system accurately classifies the DOAs for any desired number of classes and DOA separations from one degree to fifteen degrees.

The antenna array includes eight elements, therefore the training and test signals were 8×1 vectors. The training and test signals are the complex outputs from the antenna array. The received complex signal is modeled with a zero mean normal distribution with unit variance; the additive noise includes a zero mean distribution with a 0.2 variance. The DOAs for the set of test signals are unknown to the system. Both the training and test signals consisted of 1500 samples and the window length of the sample covariance matrix was set to five. Therefore the training and test sets were composed of 300 samples of each 8×1 projection vector.

The system training consists of six DDAG nodes for the four DOA classes. To completely test the LS-SVM DDAG system's capabilities the simulation were automated to test a wide range of DOAs. The DOA test set consisting of signals ranging from three degrees before the first DOA class to three degrees after the last DOA class. Thus there were forty-six test signals for Figure 4 and fourteen test signals for Figure 5.

As can be seen from the two plots the LS-SVM DDAG DOA estimation algorithm is extremely accurate. No misclassifications were logged. By testing two projection vectors at each node there is a possibility of classifying the DOA between two DOA labels, effectively increasing the resolution of the system. This condition exists if the average value of the two LS-SVM output vectors are equal, or within a set range. This occurs at the 10 degree point in Figure 4.

6.4 Multilabel Capability for Multiple DOAs

The machine learning algorithm must generate multiclass labels, $y_i \in \chi$, where $\chi \in [-90, 90]$ is a set of real numbers that represent an appropriate range of expected DOA values, and multiple labels $y_i, i = 1 \dots L$ for L dominant signal paths. If antenna sectoring is used in the cellular system the multiclass labels are from the set $\chi \in [S_i]$, where S_i is field of view for the i^{th} sectors.

In DOA estimation for CDMA cellular systems there can be multiple DOAs for a given signal. This results from multipath effects induced by the environment. The Rake receiver design includes independent receivers that track signals within a specific time delay. This design reduces the number of DOAs tracked for a given Rake receiver finger. The machine learning system must be able to discriminate between a small number of independent DOAs that include signal components with similar time delays. With this constraint the machine learning algorithm must be a multiclass system and able to process multiple labels.

The multiclass LS-SVM with output encoding is another possible approach

Figure 4: LS-SVM for DOA estimation, four classes with 10 degree separation between each.

to DOA estimation. The interesting aspect of the multiclass LS-SVM is the concept of using encoding techniques to reduce the number of classes. If there are P classes then we could train $G = \log_2(P)$ LS-SVM classes, which greatly reduces the training and testing time.

7 Conclusion

Interference suppression for a CDMA communication system promotes lower mobile transmit power and higher system capacity. Due to CDMA design the DOA estimates for interference suppression do not require the same level of accuracy that is required for FDMA or TDMA systems. A reduction in beamwidth automatically reduces the MAI, the degree of MAI reduction is commensurate with the reduction in beamwidth. Therefore there is a trade-off between computational complexity and MAI reduction. In addition antenna sectoring will also reduce the computational requirements by reducing the number of classes per SVM system. Likewise the application of Rake receivers in the communication system reduces the number of DOA per each Rake finger, again reducing the number of multilabels in a multiclass SVM system.

In this paper we presented a machine learning architecture for DOA estimation as applied to a CDMA cellular system. In addition we presented an overview of a multiclass SVM learning method and successful implementation

Figure 5: LS-SVM for DOA estimation, four classes with 1 degree separation between each.

of a multiclass LS-SVM DDAG system for DOA estimation. Initial simulation results show a high degree of accuracy, as related to the DOA classes and prove that the LS-SVM DDAG system has a wide range of performance capabilities. Future work will investigate the performance of the LS-SVM DDAG system for a multiclass, multilabel DOA estimation problem and more complex communication channels will be included in the simulations.

References

- [1] Joseph C. Liberti, Jr. and Theodore S. Rappaport, *Smart Antennas for Wireless Communications: IS-95 and Third Generation CDMA Applications*, Prentice Hall, Upper Saddle River, NJ, 1999.
- [2] J.H. Winters, "Signal Acquisition and Tracking with Adaptive Arrays in the Digital Mobile Radio System IS-54 with Flat Fading," *IEEE Transactions on Vehicular Technology*, Vol. 42, No. 4, 377-384, November 1993.
- [3] Z. Raida, "Steering an Adaptive Antenna Array by the Simplified Kalman Filter," *IEEE Transactions on Antennas and Propagation*, Vol. 43, No. 6, 627-629, June 1995.
- [4] Ahmed H. El Zooghby, Christos G. Christodoulou, and Michael Georgiopoulos, "A Neural Network-Based Smart Antenna For Multiple Source

- Tracking”, IEEE Transactions On Antennas and Propagation, vol. 48, no. 5, pp. 768-776, May 2000
- [5] Ahmed H. El Zooghby, Christos G. Christodoulou, and Michael Georgiopoulos, “Performance of Radial-Basis Function Networks for Direction of Arrival Estimation with Antenna Arrays”, IEEE Transactions On Antennas and Propagation, vol. 45, no. 11, pp. 1611-1617, November 1997
- [6] Ahmed H. El Zooghby, Christos G. Christodoulou, and Michael Georgiopoulos, “Neural Network-Based Adaptive Beamforming for One- and Two- Dimensional Antenna Arrays”, IEEE Transactions On Antennas and Propagation, vol. 46, no. 12, pp. 1891-1893, December 1998
- [7] Farrokh Rashid-Farrokhi, Leandros Tassiulas, K.J. Ray Liu, “Joint Optimum Power Control and Beamforming in Wireless Networks Using Antenna Arrays”, IEEE Transactions On Communications, vol. 46, no. 10, pp. 1313-1324, October 1998.
- [8] Rober A. Monzingo and Thomas W. Miller, *Introduction to Adaptive Antenna Arrays*, Wiley, New York, 1980.
- [9] Teong Chee Chuah, Bayan S. Sharif, and Oliver R. Hinton, “Robust Adaptive Spread-Spectrum Receiver with Neural-Net Preprocessing in Non-Gaussian Noise”, IEEE Transactions On Neural Networks, vol. 12, no. 3, pp. 546-558, May 2001.
- [10] David C. Chen and Bing J. Sheu, “A Compact Neural-Network-Based CDMA Receiver”, IEEE Transactions On Circuits and Systems II: Analog and Digital Signal Processing, vol. 45, no. 3, pp. 384-387, March 1998.
- [11] Kaushik Das and Salvatore D. Morgera, “Adaptive Interference Cancellation for DS-CDMA Systems Using Neural Network Techniques”, IEEE Journal On Selected Areas In Communications, vol. 16, no. 9, pp.1774-1784, December 1998.
- [12] Xiao Ming Gao, Xiao Zhi Gao, Jarno M. A. Tanskanen, and Seppo J. Ovaska, “Power Prediction in Mobile Communication Systems Using an Optimal Neural-Network Structure”, IEEE Transactions On Neural Networks, vol. 8, no. 6, pp. 1446-1455, November 1997.
- [13] S. Chen, A.K. Samingan, and L. Hanzo, “Support Vector Machine Multiuser Receiver for DS-CDMA Signals in Multipath Channels”, IEEE Transactions On Neural Networks, vol. 12, no. 3, pp. 604-611, May 2001.
- [14] Daniel J. Sebald and James A. Bucklew, “Support Vector Machine Techniques for Nonlinear Equalization”, IEEE Transactions On Signal Processing, vol. 48, no. 11, pp. 3217-3226, November 2000.

- [15] Klaus-Robert Muller, Sebastian Mika, Gunnar Ratsch, Koji Tsuda, and Bernard Scholkopf, “An Introduction to Kernel-Based Learning Algorithms”, *IEEE Transactions on Neural Networks*, vol. 12, no. 2, pp. 181-201, March 2001.
- [16] Vladimir N. Vapnik, *The Nature of Statistical Learning Theory*, Springer-Verlag, New York, 1995.
- [17] Nello Christianini and John Shawe-Taylor, *An Introduction to Support Vector Machines*, Cambridge University Press, New York, 2000.
- [18] Johan A.K. Suykens, “Support Vector Machines: A Nonlinear Modelling and Control Perspective”, *European Journal of Control*, vol 7, pp. 311-327, 2001
- [19] John C. Platt, “Fast Training of Support Vector Machines using Sequential Minimal Optimization”, in *Advances in Kernel Methods-Support Vector Learning*, B. Scholköpf, C.J.C. Burges, and A.J. Smola, Eds, pp 185-208, Cambridge, MA, MIT Press, 1999.
- [20] John C. Platt, “Using Analytic QP and Sparseness to Speed Training of Support Vector Machines”, in *Advances in Neural Information Processing Systems*, vol. 11, Cambridge, MA, MIT Press, 1999.
- [21] J.A.K. Suykens, L. Lukas, P. Van Dooren, B. DeMoor, and J. Vandewalle, “Least Squares Support Vector Machine Classifiers: a Large Scale Algorithm”, *ECCTD’99 European Conf. on Circuit Theory and Design*, pp. 839-842, August 1999.
- [22] John C. Platt, Nello Christianini, and John Shawe-Taylor, “Large Margin DAGs for Multiclass Classification”, in *Advances in Neural Information Processing Systems*, vol. 12, pp. 547-553, Cambridge, MA, MIT Press, 2000.
- [23] Bin Yang, “Projection Approximation Subspace Tracking”, *IEEE Transactions on Signal Processing*, vol. 43, no. 1, pp. 95-107, January 1995.
- [24] R.O. Schmidt, “Multiple Emitter Location and Signal Parameter Estimation”, *IEEE Transactions on Antennas and Propagation*, AP-34, pp. 276-280, March 1986.
- [25] Richard H. Roy, and Thomas Kailath, “ESPRIT-Estimation of Signal Parameters Via Rotational Invariance Techniques”, *IEEE Transactions On Acoustics, Speech, and Signal Processing*, vol. 37, no. 7, pp. 984-995, July 1989.
- [26] Guodong Guo, Stan Z. Li, and Kap Luk Chan, “Support Vector Machines for Face Recognition”, *Image and Vision Computing*, vol 19, pp. 631-638, 2001.