PRIMAL-DUAL OPTIMIZATION METHODS IN NEURAL NETWORKS AND SUPPORT VECTOR MACHINES TRAINING

Theodore B. Trafalis
School of Industrial Engineering
University of Oklahoma
202 West Boyd, Room 124, Norman, OK 73019
E-mail: trafalis@mailhost.ecn.ou.edu

May 7, 1999

ABSTRACT

Recently a lot of attention has been given to applications of mathematical programming to machine learning and neural networks. In this tutorial we investigate the use of Interior Point Methods (IPMs) to Support Vector Machines (SVMs) and Artificial Neural Networks (ANNs) training. The training of ANNs is a highly nonconvex optimization problem in contrast to the SVMs training problem which is a convex optimization problem. Specifically, training a SVM is equivalent to solving a linearly constrained quadratic programming (QP) problem in a number of variables equal to the number of data points. This problem becomes quite challenging when the size of the data becomes of the order of some thousands. IPMs have been shown quite promising for solving large scale linear and quadratic programming problems. We focus on primal-dual IPMs applied to SVMs and neural networks and investigate the problem of reducing its computational complexity. We also develop a new class of incremental nonlinear primal-dual techniques for artificial neural training and provide preliminary experimental results for financial forecasting problems.

INTRODUCTION

The objective of this paper is to provide an introductory tutorial on the basic ideas of primal-dual Interior Point Methods (IPMs) [45, 44] as applied to Artificial Neural Networks (ANNs) and Support Vector Machines (SVMs) learning. Other IPMs applied to ANNs training are discussed in [52, 50, 48, 56]. IPMs [44, 66] have been very successful in solving large scale problems in linear programming. Numerous algorithms have been developed for this purpose. The primal-dual algorithm [28] and its variants are among the most efficient techniques for solving linear programming problems. In the general nonlinear case, several algorithms have also been proposed [67, 68, 27, 14, 7, 8, 17]. The nonlinear primal-dual algorithms attempt to solve the first order Karush-Kuhn-Tucker (KKT) (or perturbed) optimality conditions. In the case of a convex optimization problem, the system of optimality conditions is solved using Newton's method. Efficient factorization techniques are used to solve the resulting linear system of equations and when the problem is nonconvex, the Jacobian of the system is sometimes replaced by a Quasi-Newton approximation update [67].

In engineering applications, where problems often possess a special structure such as in least squares problems, one would like to exploit those special structures to improve the computational complexity of the algorithm or reduce the memory storage requirements. For instance, in least squares problems, the objective function can be decomposed into a sum of elementary functions that relate to independent data blocks [1]. Bertsekas [4, 5, 3] has shown how one can exploit this structure to develop incremental

[1] Davidon [11] proposed a similar algorithm without any proof of convergence.

algorithms that process the data block one at a time. This is very useful in processing data from online applications. It is also beneficial to use such incremental algorithms for very large data sets because not all the data are required to be stored and processed at the same time. We discuss an incremental version [10, 49] of the nonlinear primal-dual algorithm developed by El-Bakry et al. [14] and Yamashita [67]. Until recently, incremental techniques were limited to unconstrained optimization problems [4, 5, 57, 11] with the exception of [10, 49]. In [10, 49] recent developments in constrained nonlinear optimization and incremental concepts from Extended Kalman Filtering [3] are combined.

The ANNs training problem can be seen as a nonlinear least squares problem [20]. The well-known backpropagation algorithm [64] takes advantage of this structure to train ANNs online. The backpropagation algorithm is a gradient descent technique and it converges rather slowly to a local minimum of the training error surface. One would like to apply the recent state of the art optimization techniques such as interior point methods to train ANNs. For work along this line see [56]. Here, we discuss a nonlinear primal-dual algorithm to train ANNs. The ANN training problem is written as a general nonconvex nonlinear optimization problem with linear constraints. The constraints on the weights are chosen as to avoid saturation of the neurons outputs [2]. By processing the data incrementally, the technique reduces the memory storage requirements for a single iteration and the computational complexity of each iteration is also reduced.

The second part of the paper discusses applications of IPMs to SVMs training. Although the subject SVMs can be said to have started in the late seventies it is only now receiving increasing attention in the machine learning [9] and computer vision research communities [35]. A basic reason is that the training of SVMs is slow, especially for large problems.

Recently a lot of attention has been given to applications of mathematical programming to machine learning and neural networks [31, 30, 55, 48]. There are several challenges for mathematical programming [6]. We consider the issue of scaling to large sets of constraints and variables.

Training a SVM is equivalent to solving a linearly constrained quadratic programming (QP) problem in a number of variables equal to the number of data points [62]. This problem becomes quite challenging when the size of the data becomes of the order of some thousands. IPMs have been shown quite promising for solving large scale linear and quadratic programming problems [44, 28]. Since the training of SVMs can be cast as a quadratic programming problem it seems natural to investigate the use of IPMs for training SVMs. IPMs have been already used by the author for training artificial neural networks (ANNs) with good results [52, 51, 46, 47, 53, 54, 50, 55, 48, 56, 49]. The training of ANNs is a highly nonconvex optimization problem in contrast to the SVMs training problem which is a convex optimization problem. Therefore IPMs can be used quite effectively in SVM training.

There are several classes of IPMS from which to choose to solve the SVMs QP problem [45, 44]. Some preliminary work using primal-dual methods has been investigated by Smola [40]. He solved moderately sized problems up to approximately 3,000 samples. For more data points it is more appropriate to use column generation interior point techniques [45].

This paper is organized as follows. The next three sections present a short overview of IPMs in mathematical programming, the general nonlinear primal-dual technique, and incremental primal-dual algorithm, respectively. Then, we apply the primal-dual technique to ANN training and to financial forecasting. In addition, we give a short overview of the basics in SVMs and the primal-dual approach applied to SVM training is presented. We also discuss interior point column generation techniques and future research issues.

IPMS IN MATHEMATICAL PROGRAMMING

Ideas such as the ellipsoid method [26] or barrier methods [15] from the 60's and 70's are at the origin of the interior point methods for linear programming. However, before 1984, there was no practical implementation of these ideas. In 1984, Karmarkar presented a projective algorithm [23] for linear programming. The algorithm theoretically outperformed the simplex method for large scale problems. Soon, variants such as the affine scaling algorithm [59] and the method of analytical centers [42] were

[2] This is a common problem in ANN training. The weights tend to reach large values that lead to saturated outputs [63].

developed and lead to even better computational complexity than the original projective algorithm. The primal-dual method for linear programming [32, 28] was developed in 1986. The primal-dual method is now considered as the most effective interior point method for linear programming. It leads to very nice complexity results and it also shows a very elegant and attractive structure that is at the origin of what is now known as the path-following methods. From a theoretical point of view, researchers realized that there was a strong connection between these algorithms and the barrier methods [15].

Later, extensions of interior point methods to the convex quadratic programming problem were investigated. Algorithms such as the trust region technique [43, 16] or the primal-dual method for convex quadratic problems [33] were proposed. A trust region technique for nonconvex quadratic programming problem was later developed [69]. It solves a ball constrained convex quadratic program using an affine scaling based method at each iteration.

In the past few years, extensive work has been done in the area of positive semi-definite programming (PDP) [58]. The problem is to minimize a linear function with respect to the variable x subject to the constraint that a matrix $F(x)$ is positive semi-definite. It can be shown that linear and quadratic programming are special cases of PDP. Many problems that arise in engineering can be written as PDP problems. Because PDP unifies linear, quadratic and convex optimization problems in general, it has become a very appealing research direction in the area of optimization. The reader should refer to [58] for an extensive review of PDP. A more extensive literature review can be found in the web site http://www-unix.mcs.anl.gov/otc/InteriorPoint/.

THE NONLINEAR PRIMAL-DUAL TECHNIQUE

Consider the following general nonlinear programming problem

$$\begin{array}{rcl} min & f(x) & \\ s.t. & & \\ h(x) & = & 0 \\ g(x) & \geq & 0 \end{array} \tag{1}$$

where $f: \Re^n \rightarrow \Re$, $h: \Re^n \rightarrow \Re^m$, and $g: \Re^n \rightarrow \Re^p$. The Lagrangian associated with Problem 1 is

$$L(x, y, z) = f(x) + y^T h(x) - z^T g(x) \tag{2}$$

where $y \in \Re^m$ and $z \in \Re^p$ are the vectors of Lagrange multipliers for the equalities h and the inequalities g.

The Karush-Kuhn-Tucker optimality conditions can be written as follows

$$\begin{array}{rcl} \nabla_x L(x, y, z) & = & 0 \\ h(x) & = & 0 \\ g(x) & \geq & 0 \\ Zg(x) & = & 0 \\ z & \geq & 0 \end{array} \tag{3}$$

where $\nabla_x L(x, y, z) = \nabla f(x) + \nabla h(x) y - \nabla g(x) z$ and Z is a diagonal matrix formed with the coordinates of the vector z ($Z = diag\{z\}$).

We introduce slack variables for the inequality constraints. Let s be the vector of slack variables and S be the associated diagonal matrix. We also define e as the unit vector ($e \in \Re^p$, $e(i) = 1, i = 1, \ldots, p$). The KKT system of equations 3 can be reduced to the following compact form

$$F_\mu(x, y, s, z) = \begin{pmatrix} \nabla_x L(x, y, z) \\ h(x) \\ g(x) - s \\ ZSe \end{pmatrix} = 0 \ (s, z) \geq 0. \tag{4}$$

We perturb the complementary slackness condition as follows

$$ZSe \;=\; \mu e \tag{5}$$

where $\mu > 0$ is the perturbation parameter and will be decreased at each iteration. The perturbation of the KKT conditions can be seen as a way to ensure adherence to the central path in order to achieve global convergence (see [14]).

In the following, we will use the perturbed KKT system 6 instead of the original KKT system 4. It can be written in the form

$$F_\mu(x,y,s,z) = \begin{pmatrix} \nabla_x L(x,y,z) \\ h(x) \\ g(x) - s \\ ZSe - \mu e \end{pmatrix} = 0 \;\; (s,z) \geq 0. \tag{6}$$

Next, we recall Newton's method for solving the nonlinear system of equations 6.

Let $v_k = (x_k, y_k, s_k, z_k)$ be a current Newton iterate.

The Newton correction $\Delta v_k = (\Delta x_k, \Delta y_k, \Delta s_k, \Delta z_k)$ is the solution of the following linear system

$$J(v_k)\Delta v_k \;=\; -F_\mu(v_k) \tag{7}$$

where $J(v_k) = F'_\mu(v_k)$ and $J(v_k)$ is given by

$$J(v_k) = \begin{pmatrix} \nabla_x^2 L(x,y,z) & \nabla h(x) & 0 & -\nabla g(x) \\ \nabla h(x)^T & 0 & 0 & 0 \\ \nabla g(x)^T & 0 & -I & 0 \\ 0 & 0 & Z & S \end{pmatrix}. \tag{8}$$

In the general case, the matrix J is indefinite and the non-singularity of J is not always insured. If singularity happens, it is due to the fact that the reduced hessian of the Lagrangian $\nabla_x^2 L(x,y,z)$ is non singular . In [10] requirements about linear independence of the constraint normals and strict complementarity are discussed which also enter the conditions for singularity of J. For problems that do not generate singularity $\nabla_x^2 L(x,y,z)$ is calculated by central differences [10]. For problems that are nonconvex, $\nabla_x^2 L(x,y,z)$ will be approximated by a positive definite matrix using a Quasi-Newton update formula [41]. This approximation of the hessian not only helps in convexifying the problem but also in improving computational complexity. The central differences involved in calculating the hessian are rather tedious and time consuming because a large amount of function evaluations is required. However, a Quasi-Newton update is developed that requires only first derivatives, therefore the number of function evaluations is considerably reduced. The main application of primal-dual methods has been in instances where second derivatives are available. In cases where the second derivatives are not available central-differece approximations can be applied. Generally primal-dual algorithms, applied for example on a minimization problem, may converge to a local maximum or even worse to a saddle point since the first order optimality conditions are also satisfied in those points. To avoid such cases a merit function is incorporated within the primal-dual interior point algorithm. This is achieved by using an Armijo rule to determine the step-size, which guarantees the monotonic decrease of the merit function [10].

Next, we describe the Quasi-Newton update used in approximating $\nabla_x^2 L(x,y,z)$.

Let $H^{(k)}$ denote the approximation of $\nabla_x^2 L(x,y,z)$ at iteration k. We will use the following approximation

$$H^{(k+1)} = \lambda H^{(k)} \;+\; \nabla_x L(x,y,z)(\nabla_x L(x,y,z))^T. \tag{9}$$

where $0 < \lambda \leq 1$. Using the Sherman-Morrisson-Woodbury formula [13], the inverse of $H^{(k+1)}$ denoted by $P^{(k+1)}$ can be expressed as follows

$$P^{(k+1)} = \frac{P^{(k)} - P^{(k)}\nabla_x L \left((\nabla_x L)^T P^{(k)} \nabla_x L \;+\; \lambda I\right)^{-1} (\nabla_x L)^T P^{(k)}}{\lambda} \tag{10}$$

where $\nabla_x L = \nabla_x L(x,y,z)$.

This update of the Hessian is based on the Recursive Prediction Error Method (RPEM) [41]. It is similar to the Quasi-Newton update of Davidon [11]. In numerical analysis, the BFGS update is often used to approximate the Hessian.

We perform linesearches in each Newton direction. Let $\alpha_k = (\alpha_x, \alpha_y, \alpha_s, \alpha_z)$ be the vector of resulting optimal steplengths. Therefore we can summarize the move in the primal and dual spaces by the following

$$v_{k+1} = v_k + \Lambda_k \Delta v_k \,, \tag{11}$$

where $\Lambda_k = diag\{\alpha_k\}$ is the diagonal matrix composed of the coordinates of α_k.

The algorithm performs multiple moves as described above until a stopping criterion is met. The stopping criterion is chosen to be the following

$$\|F_\mu(v_k)\| < \epsilon. \tag{12}$$

This means that we will find an ϵ-approximate solution of the KKT optimality conditions. In practice, this approximate solution is usually satisfactory. The choice of ϵ depends on the accuracy needed for the specific practical application.

Next, we give a generic primal-dual algorithm that emphasizes and summarizes the major steps that we have elaborated above.

Nonlinear Primal-Dual Algorithm (NLPD)

Step 1 Set k=0.

Step 2 Start with $v_k = (x_k, y_k, s_k, z_k)$ where $(s_k, z_k) > 0$ and initialize $\mu_k > 0$.

Step 3 Check stopping criterion in Eq. 12

- if Eq. 12 is satisfied go to Step 8.
- if Eq. 12 is not satisfied go to Step 4.

Step 4 Solve the system of linear equations 7 for Δv_k.

Step 5 Perform the linesearches to determine α_k.

Step 6 Move to next point $v_{k+1} = v_k + \Lambda_k \Delta v_k$.

Step 7 Set $k = k + 1$, update μ_k and go to Step 3.

Step 8 Stop with the optimal point $v^* = v_k$.

Factorizing the Jacobian Matrix

Recall that the main step of the primal-dual algorithm is to solve the following linear system of equations

$$J(v_k)\Delta v_k \;=\; -F_\mu(v_k), \tag{13}$$

where $J(v_k) = F'_\mu(v_k)$ and $J(v_k)$ is given by

$$J(v_k) = \begin{pmatrix} \nabla^2_x L(x,y,z) & \nabla h(x) & 0 & -\nabla g(x) \\ \nabla h(x)^T & 0 & 0 & 0 \\ \nabla g(x)^T & 0 & -I & 0 \\ 0 & 0 & Z & S \end{pmatrix}. \tag{14}$$

By susbstituting some terms in 13, we will see that we can reduce the dimension of the system to be inverted from $(n+m+2p)\times(n+m+2p)$ to $(n+m)\times(n+m)$ and that the reduced system is symmetric. The symmetry of the system is an important property. It permits the use of factorization techniques

such as the Cholesky factorization or the Bunch and Parlett symmetric indefinite factorization. The reduction of 13 to a smaller symmetric system is described next.

13 can be written as

$$\left\{\begin{array}{lllll} \nabla_x^2 L(x,y,z)dx & +\nabla h(x)dy & -\nabla g(x)dz & = & -f_1 \\ \nabla h(x)^T dx & & & = & -f_2 \\ \nabla g(x)^T dx & -ds & & = & -f_3 \\ & Zds & +Sdz & = & -f_4 \end{array}\right. \quad (15)$$

where

$$F_\mu(x,y,s,z) = \begin{pmatrix} \nabla_x L(x,y,z) \\ h(x) \\ g(x) - s \\ ZSe - \mu e \end{pmatrix} = \begin{pmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \end{pmatrix} \quad (16)$$

From 15, we have

$$dz = -S^{-1} f_4 - S^{-1} Z ds \quad (17)$$

and

$$ds = \nabla g(x)^T dx + f_3 \quad (18)$$

therefore,

$$dz = -S^{-1} f_4 - S^{-1} Z \nabla g(x)^T dx \quad (19)$$

Substituting the expression of dz back in the first equation of 15, we obtain

$$\nabla_x^2(x,y,z)dx + \nabla h(x)dy + \nabla g(x) S^{-1} f_4 + \nabla g(x) S^{-1} Z \nabla g(x)^T dx = -f_1 \quad (20)$$

Hence 15 is equivalent to

$$\left\{\begin{array}{lll} \left[\nabla_x^2 L(x,y,z) + \nabla g(x) S^{-1} Z \nabla g(x)^T\right] dx & +\nabla h(x)dy & = \quad -f_1 - \nabla g(x) S^{-1} f_4 \\ \nabla h(x)^T dx & & = \quad -f_2 \end{array}\right. \quad (21)$$

and

$$ds = \nabla g(x)^T dx + f_3 \quad (22)$$

$$dz = -S^{-1} f_4 - S^{-1} Z \nabla g(x)^T dx \quad (23)$$

As we can see, the system 21 is symmetric and of dimension $(n+m) \times (n+m)$. To gain memory space and to speed up the algorithm, we use fast factorization techniques for inverting 21.

The Update of μ

The update of μ remains one of the most critical issue in primal-dual techniques. It has great effect on the behavior of the algorithm. It is well understood that it corresponds to the concept of adherence to the central path and therefore to the convergence of the algorithm to the optimal point. Even though the role of μ is clear, the choice of the update formula for μ remains uncertain. In the linear programming case, μ is usually chosen as proportional to the duality gap so that reducing μ ensures a reduction in the duality gap at each iteration and eventually it becomes zero at optimality. Experiments have shown that this update is sucessful. In the nonlinear case, a similar update is used. The justification for this choice is not as straight forward as in linear programming. Attempts to explain it have already been proposed [14, 67].

We use the following update formula

$$\mu_k = \sigma_k \frac{s_k^T z_k}{p} \quad (24)$$

where σ_k is a parameter of the algorithm. El-Bakry et al. [14] have proposed the following values for σ_k

$$\sigma_k = 100 s_k^T z_k \quad \text{if} \quad 100 s_k^T z_k < 0.2$$
$$\sigma_k = 0.2 \quad \text{if} \quad 100 s_k^T z_k > 0.2$$

The Linesearches

Recall the following notations

$$v = \begin{pmatrix} x \\ y \\ s \\ z \end{pmatrix} \quad dv = \begin{pmatrix} d_x \\ d_y \\ d_s \\ d_z \end{pmatrix}$$

We start by taking a full Newton step as follows

$$\begin{aligned} x_{k+1} &= x_k + \alpha_p d_x \\ y_{k+1} &= y_k + \alpha_d d_y \\ s_{k+1} &= s_k + \alpha_s d_s \\ z_{k+1} &= z_k + \alpha_z d_z \end{aligned} \tag{25}$$

where

$$\begin{aligned} \alpha_p &= 1 \\ \alpha_d &= 1 \\ \alpha_s &= \frac{-1}{\min_k\{\frac{d_{s_k}}{s_k}, -1\}} \\ \alpha_z &= \frac{-1}{\min_k\{\frac{d_{z_k}}{z_k}, -1\}} \end{aligned} \tag{26}$$

α_s and α_z are chosen so as to ensure that s_{k+1} and z_{k+1} remain positive after the move.

We define Φ as follows

$$\Phi(v) = \|F(v)\|^2 = F(v)^T F(v) \tag{27}$$

to be used as the merit function for the linesearch. We follow the Armijo rule for backtracking. The backtracking avoids taking steps that lead to small decrease in Φ. Refer to Dennis and Schnabel's book [13] for more details on backtracking and step taking rules. The Armijo condition can be expressed as follows:

Find the smallest integer i such that

$$\Phi(v + \overline{\alpha} 2^{-i} dv) - \Phi(v) \leq \overline{\alpha} 2^{-i} dv^T \nabla\Phi(v) \tag{28}$$

where $\overline{\alpha} = \min\{\gamma\alpha_{\max}, 1\}$, $\alpha_{\max} = \min\{\alpha_p, \alpha_d, \alpha_s, \alpha_z\}$ and γ is a parameter of the algorithm. A typical value for γ is 10^{-4}.

The Stopping Criterion

As mentioned before, the algorithm stops when the KKT conditions are satisfied with an accuracy of ϵ. Algebraically, the stopping rule is as follows

$$\|F_\mu(v_k)\| < \epsilon \tag{29}$$

In our implementation, we will rather use a normalized version of the criterion as shown below

$$\frac{\|F_\mu(v_k)\|_2^2}{1 + \|v_k\|_2^2} < \epsilon \tag{30}$$

ϵ is problem dependent and is given as a parameter of the algorithm.

AN INCREMENTAL PRIMAL-DUAL ALGORITHM FOR PROBLEMS WITH SPECIAL STRUCTURE

In [10] a primal-dual algorithm is proposed that modifies the generic primal-dual technique for problems that have the following form

$$min \quad f(x) = \textstyle\sum_{l=1}^{L} f_l(x) \tag{31}$$

$$
\begin{array}{lcl}
s.t. & & \\
h_l(x) & = & 0 \\
g_l(x) & \geq & 0 \ for \ l = 1, \ldots, L
\end{array}
$$

This type of problems is very common in engineering applications. For example, constrained least square problems are one of the special instances of 31. Our interest is in the artificial neural network training problem that has also this specific structure. In this section, we develop the incremental primal-dual technique for the general problem 31 and in the next section we describe how it is applied to the neural network training problem. In the neural network community, the incremental update of the variables is known as *online training* [20]. In general, the problems for which incremental methods are suitable are problems that deal with a large amount of data and the objective function f is a sum of errors terms between estimated and measured values for a specific problem. If the given problem has L data points, the incremental technique decomposes the problem into L subproblems. The update of the variables is performed by increments. Each increment corresponds to a data point.

One benefit from using an incremental idea rather than a classical method is a memory space gain. The memory space required by the method depends only on the size of the subproblems whereas in the traditional methods one must store information from all the data points simultaneously, which can lead to memory overflows when dealing with large scale problems. Another benefit from using an incremental technique is that the data are fed several times to the system and in the case of neural network training for example, cycles of data feeds increases the chance to find a global optimum of the error function. Also, in online applications, where all the data are not available at one time but only on a one-by-one basis, incremental methods are needed.

Incremental gradient techniques have been investigated recently [4, 5, 57]. They were first proposed by Davidon [11].

The incremental primal-dual algorithm starts from a point v_1^0 and generates the sequence $(v_1^t, \ldots, v_{L+1}^t)_{t=0,1,\ldots}$ where

$$
\begin{array}{lcl}
v_{l+1}^t & = & v_l^t \ + \Lambda_l^t \Delta v_l^t \ \ l = 1, \ldots, L \\
v_1^{t+1} & = & v_{L+1}^t
\end{array}
$$

and Δv_l^t is calculated by performing one Newton's step towards the solution of the following subproblem

$$
\begin{array}{ll}
min & f_l(x) \\
s.t. & \\
 & h_l(x) = 0 \\
 & g_l(x) \geq 0.
\end{array}
$$

Specifically, Δv_l^t is the solution of the linearized system of the KKT conditions

$$
J(v_l^t)\Delta v_l^t \ = \ -F_\mu^l(v_l^t) \tag{33}
$$

where $v_l^t = (x_l^t, y_l^t, s_l^t, z_l^t)$,

$$
F_\mu^l(v_l^t) = \begin{pmatrix} \nabla_x L_l(x_l^t, y_l^t, z_l^t) \\ h_l(x_l^t) \\ g_l(x_l^t) - s_l^t \\ ZSe - \mu e \end{pmatrix} = 0 \ (s_l^t, z_l^t) \geq 0, \tag{34}
$$

$$
L_l(x_l^t, y_l^t, z_l^t) = f_l(x_l^t) + (y_l^t)^T h_l(x_l^t) - (z_l^t)^T g_l(x_l^t), \tag{35}
$$

$$
J(v_l^t) = \begin{pmatrix} \nabla_x^2 L_l(x_l^t, y_l^t, z_l^t) & \nabla h_l(x_l^t) & 0 & -\nabla g_l(x_l^t) \\ \nabla h_l(x_l^t)^T & 0 & 0 & 0 \\ \nabla g_l(x_l^t)^T & 0 & -I & 0 \\ 0 & 0 & Z & S \end{pmatrix} \tag{36}
$$

and Δ_l^t is the vector of step lengthes calculated through linesearches. The update of μ is performed after all the increments of one phase have been executed.

This method of incrementing the update of the variables can be seen as performing a Newton's step that is the geometric sum of all the Newton's directions corresponding to each one of the data points l, $l = 1, \ldots, L$. Mathematically, it can be seen as follows

$$\begin{aligned} v_1^t &= v_{start} && (37)\\ v_2^t &= v_1^t + \Lambda_1^t \Delta v_1^t && (38)\\ v_3^t &= v_2^t + \Lambda_2^t \Delta v_2^t && (39)\\ v_4^t &= v_3^t + \Lambda_3^t \Delta v_3^t && (40)\\ &\vdots && (41)\\ v_L^t &= v_{L-1}^t + \Lambda_{L-1}^t \Delta v_{L-1}^t && (42)\\ v_{L+1}^t &= v_L^t + \Lambda_L^t \Delta v_L^t. && (43) \end{aligned}$$

Summing right and left hand sides in Equations 37 through 43 and cancelling identical terms on both sides, we obtain

$$v_{L+1}^t = v_{start} + d^t \tag{44}$$

where $d^t = \sum_{l=1}^t \Delta_l^t$ and v_{start} is some arbitrary starting value.

d^t represents the geometric sum of all the Newton's directions generated by each of the data point individually.

Next, we give the main steps of the incremental primal-dual algorithm for nonlinear programming.

Incremental Nonlinear Primal-Dual Algorithm (INC-NLPD)

Step 1 Set $t = 0$, $l = 1$.

Step 2 Start with $v_l^t = (x_l^t, y_l^t, s_l^t, z_l^t)$ where $(s_l^t, z_l^t) > 0$ and initialize $\mu > 0$.

Step 3 If stopping criterion is satisfied, go to Step 10, otherwise go to Step 4.

Step 4 while $(l \leq L + 1)$ do Step 5, 6, 7 and 8, otherwise go to Step 9.

Step 5 Solve the system of linear equations 33 for Δv_l^t.

Step 6 Perform the linesearches to determine Λ_l^t.

Step 7 Move to next point $v_{l+1}^t = v_l^t + \Lambda_l^t \Delta v_l^t$.

Step 8 Set $l = l + 1$ and go to Step 4.

Step 9 Set $t = t + 1$, $v_1^t = v_{L+1}^{t-1}$, update μ and go to Step 3.

Step 10 Stop with the optimal point $v^* = v_l^t$.

Note that in Step 3 the stopping criterion can not be defined as in the general NLPD algorithm because the norm of the KKT conditions is different for each one of the increment l. Usually, in practice the stopping criterion used is a measure of accuracy for the given application. For example in the case of artificial neural network training, the algorithm stops when the training error becomes smaller than a preset threshold.

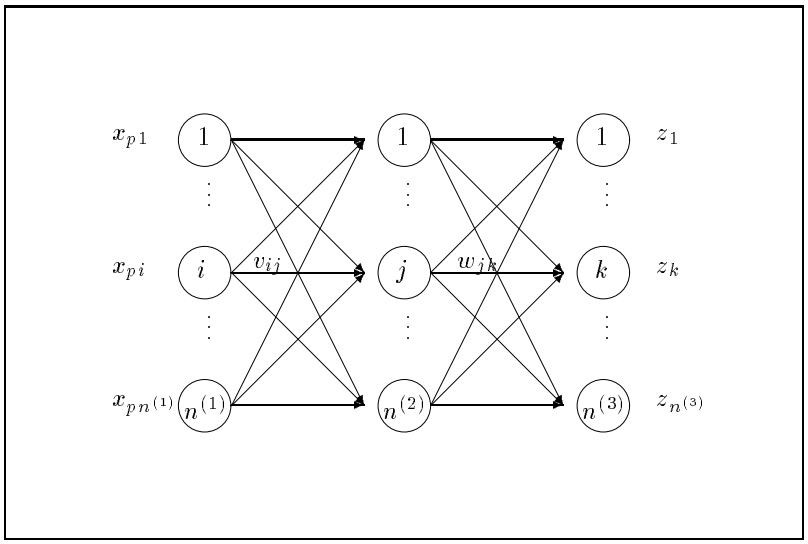

Figure 1: Multilayer Artificial Neural Network

Since the above approach generally locates local minima a stochastic variation of the NLPD is proposed in [10]. The main idea is that of perturbing the steps of the local search procedure (NLPD) by a random noise, whose aim is to help in escaping from the region of attraction of local, but not global, minima. The magnitude of the noise is decreased as the iteration counter increases. Moreover, according to a simulated annealing approach, nonimproving moves are accepted with probability defined by the so called Metropolis acceptance function. The next section describes the application of the above technique to the training of artificial neural networks.

APPLICATION TO ANNS TRAINING

Problem Statement

We consider feedforward artificial neural networks. All the connections between neurons are forward and there are no connections between neurons belonging to the same layer. We are interested in the case of multilayer ANNs. Typically, the ANN will have one input layer, one hidden layer and one output layer. The training methods that we will develop can be easily adapted to the case where multiple hidden layers are present. Figure 1 illustrates the ANN architecture used in this research.

The problem to be solved is known as supervised ANN training. Given a set of input patterns and their corresponding desired outputs, the optimal set of connection weights between neurons that achieved a minimum error between the desired outputs and the actual outputs of the network is to be found. The network is said to be trained when the optimal connection weights are found.

The classical methodologies to train such ANNs are based on the backpropagation concept first presented by Werbos [64]. The idea is to iteratively update the weights, starting from the output layer where complete information is given, and backpropagate the information to the layers that precede it. The update of the weights is derived from the gradient descent technique. Variants that used Quasi-Newton techniques have also been proposed [1]. The backpropagation algorithm (BP) has given very good results and it is still used as a benchmark algorithm for its robustness. However, BP leads to long training times and its use in online applications that require fast training is therefore inappropriate.

We propose to develop a new approach for supervised training of ANNs. We describe the training problem as a general nonconvex constrained minimization problem with linear constraints. To perform the online training of the ANN, we will use the incremental algorithm presented earlier.

The Error Minimization Problem

The following notations are to be used next

- P - Number of input patterns.
- $n^{(1)}$ - Number of input neurons.
- $n^{(2)}$ - Number of hidden neurons.
- $n^{(3)}$ - Number of output neurons.
- f_a - Common activation function for all the neurons.
- $X_p = (x_{p1}, x_{p2}, \ldots, x_{pn^{(1)}})^T$ - Input vector corresponding to pattern p.
- $D_p = (d_{p1}, d_{p2}, \ldots, d_{pn^{(3)}})^T$ - Desired output vector corresponding to pattern p.
- v_{ij} - Weight on the connection from i^{th} input neuron to j^{th} hidden neuron.
- w_{jk} - Weight on the connection from j^{th} hidden neuron to k^{th} output neuron.
- $y = (y_1, y_2, \ldots, y_{n^{(2)}})^T$ - Output vector of the hidden layer.
- $z = (z_1, z_2, \ldots, z_{n^{(3)}})^T$ - Output vector of output layer.

We have the following

$$y_j = f_a(\sum_{i=1}^{n^{(1)}} v_{ij} x_{pi}) \text{ for all } j = 1, \ldots, n^{(2)}. \tag{45}$$

and,

$$z_k = f_a(\sum_{j=1}^{n^{(2)}} w_{jk} y_j) \text{ for all } k = 1, \ldots, n^{(3)}. \tag{46}$$

The activation functions for all the neurons are assumed to be identical. A typical choice for the activation function is the hyperbolic tangent defined as

$$f_a(x) = tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}. \tag{47}$$

When the weights become too large, the activation function is working in its flat regions and the neurons have difficulties to fire. This phenomenon is known as *network paralysis* [63]. To ensure that the neurons are not saturated and that they are able to fire at all times, we would like to remain in the region of the activation function where it has a linear behavior. With this in mind, rather than imposing box constraints on the weights, we will constrain the weights between the input and hidden layers as follows

$$-\overline{\alpha} \leq f_a(\sum_{i=1}^{n^{(1)}} v_{ij} x_{pi}) \leq \overline{\alpha} \; for \; j = 1, \ldots, n^{(2)}. \tag{48}$$

Since f_a is symmetric around 0 and one-to-one mapping in its domain of definition, the above inequality is equivalent to

$$-f_a^{-1}(\overline{\alpha}) \leq \sum_{i=1}^{n^{(1)}} v_{ij} x_{pi} \leq f_a^{-1}(\overline{\alpha}) \; for \; j = 1, \ldots, n^{(2)} \tag{49}$$

where $\overline{\alpha}$ is a preset value, typically $0.7 \leq \overline{\alpha} \leq 0.9$. By this selection, we avoid the flat areas of the error function.

Therefore, the training problem can be written as the following error minimization problem

$$
\begin{aligned}
&min && E(v,w)\\
&s.t. &&\\
&&& for\ p = 1,\ldots,P\\
&&& -f_a^{-1}(\overline{\alpha}) \leq \textstyle\sum_{i=1}^{n^{(1)}} v_{i1}x_{pi} \leq f_a^{-1}(\overline{\alpha})\\
&&& -f_a^{-1}(\overline{\alpha}) \leq \textstyle\sum_{i=1}^{n^{(1)}} v_{i2}x_{pi} \leq f_a^{-1}(\overline{\alpha})\\
&&& \vdots\\
&&& -f_a^{-1}(\overline{\alpha}) \leq \textstyle\sum_{i=1}^{n^{(1)}} v_{in^{(2)}}x_{pi} \leq f_a^{-1}(\overline{\alpha})
\end{aligned}
$$

where $E(v,w) = \frac{1}{2}\sum_{p=1}^{P}\sum_{k=1}^{n^{(3)}}(z_k - d_{pk})^2$. We see that this formulation possesses the special structure required by the incremental primal-dual technique developed in the previous section. In previous work [53, 56] box constraints were used on the weights. If box constraints were to be used on the weights, one would have $2[n^{(2)}(n^{(1)}n^{(3)})]$ box constraints whereas now we have $2n^{(2)}$ weighted-sum constraints. Note that at each step of the incremental algorithm, we only consider the constraints corresponding to the current pattern. The information from the previous constraints is stored in the incremental direction of move. This type of training is known as online training of the data. The pattern are fed to the system one at a time.

Application to Financial Forecasting

In finance, one would like to predict the behavior of some market indicators so as to increase the return on investment. The problem of financial forecasting is complicated. Its complexity is due to the number of factors that can influence the behavior of the market. Usually, experts isolate a number of significant factors that are either derived from historical data or estimated subjectively from experience. Recently, a different approach has been shown to be very successful. The idea is to use machine learning to detect patterns in the financial history of a specific market and try to identify these patterns in the present and future data. Neural networks have been shown to be very accurate for this specific application [65, 37, 38, 12].

We present computational results for two types of financial data. The first database contains the values of return rate for the BOEING stock price over time (days). The second database is a record of the exchange rate from the Swiss Franc to US dollars. It has been recorded every minute. We propose to use a 3 layers neural network architecture to learn the data. We use data from the 5 previous time steps as input, and the next data point as output. Therefore the input layer has 5 input nodes, the output layer has one output node and we use 5 nodes for the hidden layer. Our experiments have shown that 5 hidden nodes gave the best architecture for this application. Figure 2 describes the precise ANN architecture.

We train the ANN using 5 different methodologies as listed below

- A backpropagation algorithm. It is a gradient descent technique that uses information of the output layer to compute gradients at the hidden layer.
- A Recursive Prediction Error Method (RPEM)[53]. It is Quasi-Newton modification of the backpropagation algorithm where the update of the approximation of the hessian matrix follows the RPEM update [41].
- A logarithmic barrier method (Log. Barrier)[53, 56]. The algorithm embeds the box constraints on the weights as a logarithmic penalty term into the objective function. It is an interior point method that uses the backpropagation framework to compute the gradients.
- A Stochastic logarithmic barrier method (Stoc. Log. Barrier)[56]. It is a stochastic version of the previous algorithm. It inflicts random perturbation on the search directions to seek better local minima and hopefully global minima.

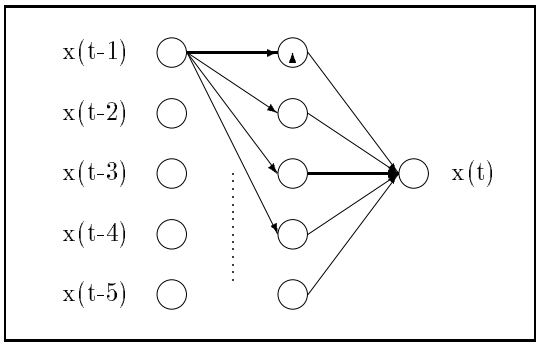

Figure 2: Artificial Neural Network Architecture for Financial Forecasting

Table 1: Training Results for Boeing Stock Price Return

Algorithm	Mean of Training Error	Variance of Training Error
Backpropagation	0.131106	0.000312879
RPEM	0.605977	0.495492
Log. Barrier	3.6606	63.9709
Stoc. Log. Barrier	0.032735	3.07687E-05
INCNLPD	0.05999418	6.13E-12

- The incremental nonlinear primal-dual algorithm (INCNLPD) developed above.

Tables 1 and 2 report the means and variances of the training error for ten runs of each algorithm. The training is performed over 40 days, and prediction is performed from day 40 to day 58. For the Swiss Franc Exchange database the training is performed over 100 patterns and the next 100 patterns are used to evaluate the prediction. From the tables, it is clear that INCNLPD outperforms the backpropagation algorithm. The backpropagation technique stops with local minima that often are not good enough to train the data. This is due to the fact that the backpropagation is a gradient descent technique that only uses local information on the error surface while INCNLPD uses primal and dual information as well as perturbation on the KKT conditions to seek good local minima. INCNLPD performs as good as the Stoc. Log. Barrier method. The testing phases are very similar for both algorithms. This is very encouraging because the Stoc. Log. Barrier algorithm was shown to be very effective for function approximation [53]. INCNLPD seems to be more robust than RPEM or Log. Barrier methods. It gives more consistent results. For the Swiss Franc database, the Log. Barrier was not able to train the data. We have also train the same ANN architecture with the *NevProp* implementation of the fast backpropagation algorithm [34]. For the BOEING database, the mean training error achieved for 10 runs was of 0.31555 with zero variance and for the Swiss Franc database, the mean training error was of 0.00326 with also zero variance. Comparing these results with those of Tables 1 and 2, INCNLPD is also much better.

APPLICATIONS OF IPMS TO SUPPORT VECTOR MACHINES

Recently a lot of attention has been given to applications of mathematical programming to machine learning and neural networks. In this section we investigate the use of IPMs to SVMs training. Empha-

Table 2: Training Results for Swiss Franc Exchange Rate

Algorithm	Mean of Training Error	Variance of Training Error
Backpropagation	0.000821972	3.84808E-12
RPEM	7.45659E-05	6.56031E-10
Log. Barrier	9.54193	11.2404
Stoc. Log. Barrier	0.000189679	4.17968E-09
INCNLPD	0.00010937982	3.9536324E-11

sis will be given to primal-dual IPMs. Training a SVM is equivalent to solving a linearly constrained quadratic programming (QP) problem in a number of variables equal to the number of data points. This problem becomes quite challenging when the size of the data becomes of the order of some thousands. IPMs have been shown quite promising for solving large scale quadratic programming problems. Therefore it seems natural to investigate the use of IPMs for training SVMs.

SVMs for Pattern Recognition

SVMs were invented by Vapnik [62]. In its simplest form an SVM is a hyperplane that separates a set A from a set B with maximum margin. The set A can be considered as a set of positive examples and the set B as a set of negative examples.

At first we start with the simplest case: linear machines trained on separable data. Let the following noiseless data are given $\{x_i, y_i\}\ ,\ i = 1, ..., l,\ y_i \in \{-1, 1\}\ ,\ x_i \in R^d$. We suppose that we can find some hyperplane (separating hyperplane) which separates the positive from the negative examples. Let d_+ be the shortest distance from the separating hyperplane to the closest positive example. Similarly let d_- be the shortest distance to the closest negative example. We define the margin of a separating hyperplane to be $d_+ + d_-$. In the linear separable case the SVM training algorithm looks for the separating hyperplane with the largest possible margin. Intuitively this hyperplane will provide the smallest generalization error among the infinite hyperplanes which separate the data linearly. In order to find the pair of hyperplanes which gives the maximum margin we consider the following optimization problem:

$$\min \|w\|^2 \quad s.t \quad y_i(x_i w + b) - 1 \geq 0 \quad \forall i. \tag{50}$$

In the case of non separable data Vapnik [62] has shown that the training of a SVM leads to the following quadratic optimization problem

$$\min \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j x_i . x_j - \sum_i \alpha_i \tag{51}$$

subject to:

$$0 \leq \alpha_i \leq C, \tag{52}$$

$$\sum_i \alpha_i y_i = 0, \tag{53}$$

where C is a constant selected by the user. The vector p defining the separating hyperplane is given as in the linear separable case by

$$p = \sum_{i=1}^{N} \alpha_i y_i x_i, \tag{54}$$

where N is the cardinality of the set of support vectors. A support vector is defined as a training vector for which $\alpha_i > 0$.

Non-linear SVMs

Now we consider the case where separating surface is a nonlinear function of the data. In order to generalize in this case we observe that in the objective function of linear SVMs the data appear in the form of dot products $x_i.x_j$. Then we map the data in a higher dimensional euclidean space through a non-linear map which we call Ψ. More specifically:

$$\Psi : R^d \rightarrow X. \tag{55}$$

The space X is called the feature space. In the feature space we consider a linear classification problem. By considering a dot product in the transformed space X we can generalize the linear SVMs into nonlinear SVMs. Specifically we define the dot product in X through a kernel function K. More specifically we will have $K(x_i, x_j) = \Psi(x_i).\Psi(x_j)$. Now we have to solve the same quadratic programming problem as in the previous section replacing the $x_i.x_j$ with $K(x_i, x_j) = \Psi(x_i).\Psi(x_j)$. The final classifier has the following nonlinear form:

$$f(x) = (\sum_{i=1}^{l} \alpha_i y_i K(x, x_i) + b) \tag{56}$$

Training a SVM involves the solution of a quadratic programming problem in l variables, where l is the number of support vectors or the number of data points. When the number of data points is large (greater than 2,000) the QP problem becomes difficult to solve with the standard QP approaches. There are interesting engineering applications (object character recognition, face detection e.t.c) [35] where 50,000 or more data points are not uncommon. Then decomposition can be applied [36].

Reformulating the Problem

Traditionally, SVMs are derived by first deriving the constrained optimization problem of finding the optimal separating hyperplane that has maximum margin (the primal problem), and then studying its dual problem.However, it is also possible to solve the primal formulation of the problem using only the kernel K. In fact, one can show that the primal problem, which is the dual of the dual problem can also be written as:

$$\min imize \ \frac{1}{2}\alpha^T Q \alpha + C \sum_{i=1}^{n} \xi_i$$

$$subject \ to:$$

$$y_i(\sum_{j=1}^{n} \alpha_j y_j K(x_i, x_j) + b) \geq 1 - \xi_i \quad i = 1, ..., n$$

$$\xi_i, \alpha_i \geq 0 \qquad i = 1, ..., n \ and \ b \ is \ free$$

This problem formulation, which from now on we refer to as the primal reformulation, has the same initial structure as the original primal formulation, but incorporates the kernel K and has a natural and clear interpretation (see Osuna and Girosi [39]). The above formulation provides a nice framework to apply column generation interior point techniques which will be described later.

SVMs for Regression

Recently SVMs have been used for regression. Next we extend the SVM approach to regression problems developing the basic ideas [39]. Suppose we have given a set of noiseless training data $(x_1, y_1), ..., (x_l, y_l) \subseteq X \times R$, where X denotes the space of input patterns and $X \subseteq R^d$. Usually the set X contains points with components describing attributes for a specific application. For applications in finance these might be exchange rates and some econometric indicators. Our goal is to find a function f that has a deviation

of at most ϵ from the desired targets y_i and is as flat as possible. In order to explain the ideas we start with the simple case where we approximate the data with linear functions

$$f(x) = w^T x + b. \tag{57}$$

The flatness in this case means that the length of w is small. Therefore we consider the following convex optimization problem:

$$\min \frac{1}{2} \|w\|^2 \; s.t. \quad \begin{cases} y_i - w^T x_i - b \leq \varepsilon \\ -y_i + w^T x_i + b \leq \varepsilon \end{cases} \tag{58}$$

Here we make the assumption that the constrained optimization problem is feasible. In order to work with infeasible constraints we introduce slack variables ξ_i and ξ_i^* respectively. Then we arrive at the following formulation ([62])

$$\min \frac{1}{2} \|w\|^2 + c \sum_{i=1}^{l} (\xi_i + \xi_i^*) s.t \begin{cases} y_i - w^T x_i - b \leq \varepsilon + \xi_i \\ -y_i - w^T x_i + b \leq \varepsilon + \xi_i^* \\ \xi_i, \xi_i^* \geq 0 \end{cases} \tag{59}$$

By considering the Wolfe dual of the above optimization problem and mapping the data x into a high dimensional space F (equipped with a dot product) through a nonlinear mapping Φ as in the previous section we have the following quadratic optimization problem:

$$\max W(\alpha, \alpha^*) = -\varepsilon \sum_{i=1}^{l} (\alpha_i^* + \alpha_i) + \sum_{i=1}^{l} (\alpha_i^* - \alpha_i) y_i$$

$$-\frac{1}{2} \sum_{i,j=1}^{l} (\alpha_i^* - \alpha_i)(\alpha_j^* - \alpha_j) k(x_i, x_j)$$

$$subject\ to\ 0 \leq \alpha_i, \alpha_i^* \leq C,\ and\ \sum_{i=1}^{l} (\alpha_i - \alpha_i^*) = 0,$$

where C, ϵ are chosen by the user. Then the regression estimator takes the following form

$$f(x) = \sum_{i=1}^{l} (\alpha_i^* - \alpha_i) k(x, x_i) + b, \tag{60}$$

where α_i^*, α_i are the optimal solutions of the above optimization problem. The dot product is defined through a kernel function as follows:

$$k(x_i, x_j) = \Phi(x_i).\Phi(x_j), \tag{61}$$

where k is a symmetric function satisfying the conditions in Mercer's theorem [39, 40]. Next we investigate a QP primal-dual formulation of the SVMs training problem.Our presentation follows [40].

0.1 Quadratic Programming Primal-Dual Formulation

The SVMs training problem in its dual form is a special case of the following general QP problem:

$$\min \frac{1}{2} q(\alpha) + c.\alpha \tag{62}$$

$$s.t \quad A\alpha = b \qquad l \leq \alpha \leq u. \tag{63}$$

By adding slack variables for the box constraints we have the following formulation:

$$\min \frac{1}{2} q(\alpha) + c.\alpha$$

$$s.t \;\; A\alpha = b, \;\; \alpha - g = l, \;\; \alpha + t = u$$

$$g, t \geq 0, \;\; \alpha \;\; free$$

The Wolfe dual of the above problem can be written as follows

$$\max imize \frac{1}{2}(q(\alpha) - \nabla_\alpha q(\alpha).\alpha) + b^T y + l^T z - u^T s$$

$$subject \; to \;\; \frac{1}{2}\nabla_\alpha q(\alpha) + c - (Ay)^T + s = z$$

$$s, z \geq 0, \;\; y \;\; free.$$

A necessary condition for optimality is that the primal/dual variables satisfy both the feasibility conditions and the complementary slackness conditions,

$$\left.\begin{array}{l} g_i z_i = 0 \\ s_i t_i = 0 \end{array}\right\} \quad for \; all \; i \in \{1, ..., l\} \tag{64}$$

The basic idea of IPMs path following algorithms as described in section 3 is to solve the perturbed KKT conditions iteratively. The duality gap between primal and dual objective function monitors the progress of the algorithm towards the optimal solution. More specifically, one considers the perturbed complementary slackness conditions:

$$\left.\begin{array}{l} g_i z_i = \mu \\ s_i t_i = \mu \end{array}\right\} \quad for \; all \; i \in \{1, ..., l\} \tag{65}$$

Then by linearizing the perturbed KKT system we get the following:

$$\begin{array}{l} A(\alpha + \Delta\alpha) = b \\ \alpha + \Delta\alpha - g - \Delta g = l \\ \alpha + \Delta\alpha + t + \Delta t = u \\ c + \frac{1}{2}\nabla_\alpha q(\alpha) + \frac{1}{2}\nabla^2_\alpha q(\alpha)\Delta\alpha - (A(y + \Delta y))^T + s + \Delta s = z + \Delta z \\ (g_i + \Delta g_i)(z_i + \Delta z_i) = \mu \\ (s_i + \Delta s_i)(t_i + \Delta t_i) = \mu \end{array} \tag{66}$$

The method is described in Vanderbei [60]. Then one gets

$$\begin{array}{l} A\Delta\alpha = b - A\alpha =: \rho \\ \Delta\alpha - \Delta g = l - \alpha + g =: \nu \\ \Delta\alpha + \Delta t = u - \alpha - \tau =: \tau \\ (A\Delta y)^T + \Delta z - \Delta s - \frac{1}{2}\nabla^2_\alpha q(\alpha)\Delta\alpha = c - (Ay)^T + \frac{1}{2}\nabla_\alpha q(\alpha) + s - z =: \sigma \\ g^{-1} z \Delta g + \Delta z = \mu g^{-1} - z - g^{-1}\Delta g \Delta z =: \gamma_z \\ t^{-1} s \Delta t + \Delta s = \mu t^{-1} - s - t^{-1}\Delta t \Delta s =: \gamma_s \end{array} \tag{67}$$

where

$$g^{-1} = (1/g_1, ..., 1/g_n). \tag{68}$$

Similarly for t^{-1}. The product of two vectors is defined as the componentwise product of vectors. Solving the above system of equations we can formulate the reduced KKT-system:

$$\begin{bmatrix} -(\frac{1}{2}\nabla^2_\alpha q(\alpha) + g^{-1}z + t^{-1}s & A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} \Delta\alpha \\ \Delta y \end{bmatrix} = \begin{bmatrix} \sigma - g^{-1}z\hat{\nu} - t^{-1}s\hat{\tau} \\ \rho \end{bmatrix} \tag{69}$$

In order to solve the above system we use the standard Cholesky factorization. There are several issues to consider. One is the selection of the parameter μ. Vanderbei ([60]) suggests the following heuristic:

$$\mu = \frac{g^T z + s^T t}{2n} \left(\frac{\xi - 1}{\xi + 10}\right)^2 \tag{70}$$

The logic behind this heuristic is to use an average of the duality gap for μ and then decrease it rapidly if we are far from the boundaries of the positive orthant. In recent work [24, 25] a new primal-dual algorithm is developed based on algebraic differential equations in which an analytic approach of computing the parameter μ is given solving an ordinary differential equation. We have also to choose a good starting point for our algorithm. Vanderbei considers the following problem

$$\begin{bmatrix} -(\frac{1}{2}\nabla^2_\alpha q(\alpha)+1) & A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} \alpha \\ y \end{bmatrix} = \begin{bmatrix} c \\ b \end{bmatrix} \tag{71}$$

, to find a good starting point for its algorithm. In order to guarantee positivity of the variables, following section 3, we have

$$\begin{array}{l} x = \max(x, u/100) \\ g = \min(\alpha - l, u) \\ t = \min(u - \alpha, u) \\ z = \min\left(\max\left(\frac{1}{2}\nabla_\alpha q(\alpha) + c - (Ay)^T, 0\right)\right) + u/100, u) \\ s = \min\left(\max\left(-\frac{1}{2}\nabla_\alpha q(\alpha) - c + (Ay)^T, 0\right)\right) + u/100, u) \end{array} \tag{72}$$

The above approach can be applied both to pattern recognition and regression. More specifically in the case of classification we have the following

$$q(\alpha) = \sum_{i,j=0}^{m} \alpha_i \alpha_j y_i y_j k(x_i, x_j). \tag{73}$$

Therefore

$$\begin{array}{l} \frac{\partial q(\alpha)}{\partial \alpha_i} = y_i \sum\limits_{j=1}^{m} \alpha_j y_j k(x_i, x_j) \\ \nabla^2_{\alpha_i, \alpha_j} q(\alpha) = y_i y_j k(x_i, x_j) \end{array}. \tag{74}$$

In this case since the Hessian is a dense matrix we have to use Cholesky factorization. In the case of regression one has to consider the following Hessian

$$N = \begin{bmatrix} K + D & -K \\ -K & K + D \end{bmatrix}$$

where D, D' are diagonal matrices. By using an orthogonal transformation we can transform the matrix N into the following form

$$O^T N O = \begin{bmatrix} 2K + \frac{D+D}{2} & \frac{D-D}{2} \\ \frac{D-D}{2} & \frac{D+D}{2} \end{bmatrix},$$

which can be inverted in $O(m^2)$ operations compared to $O(m^3)$for standard diagonalization methods for arbitrary symmetric matrices. In his recent work Smola [40] has shown the above approach to be quite effective with moderately sized problems (up to 3,000 training data sets). The derived solutions are precise in the sense that the duality gap is of the relative size of 10^-7.

On large data sets there are several problems due to memory and CPU limitations to compute and keep in memory the matrix $k(x_i, x_j)$. The quadratic form involves a matrix that has a number of elements equal to the square of the number of training examples. This matrix cannot fit easily into the memory of a standard computer (e.g, 128 Megabytes are needed if there are more than 4000 training examples).Therefore one has to investigate more efficient ways to solve the resulting quadratic optimization problems. Vapnik [61] suggested a decomposition method to solve the SVM QP, called "chunking." The idea of the chunking algorithm is that the value of the quadratic form is the same if you remove the rows and columns of the matrix that correspond to zero Lagrange multipliers α_i. Hence, the QP problem can be decomposed into a series of smaller QP problems. The objective is to identify all of the non zero α_is and discard all of the zero α_is. Each QP subproblem is initialized with the results of the previous subproblem. Finally, the entire set of non zero α_is has been identified, solving the original QP problem. Chunking reduces the size of the matrix from the number of training examples squared to approximately the number support vectors squared. However, chunking still may

not handle large scale training problems in the case of large number of support vectors. In order to solve this problem Kaufman [39] suggested sophisticated data structures in the QP method. Osuna et al.[36] suggest keeping a constant size matrix for every QP sub problem by adding one example and subtracting one example at every step. Recently Platt [39] developed the Sequential Minimal Optimization (SMO) method that is a simple algorithm that quickly solves the SVM QP problem without any extra matrix storage and without invoking an iterative numerical routine for each sub problem. SMO decomposes the overall QP problem into QP sub problems similar to Osuna's method. A more extensive literature review on SVMs can be found on the web site http://svm.first.gmd.de/.

INTERIOR POINT COLUMN GENERATION (IPCG) TECHNIQUES

The IPCG algorithm has been developed in optimization theory to solve the so called convex feasibility problem. There, the objective is to find a feasible point in a convex set $S \subset R^M$, where we assume S contains an interior point. The convex set S may be bounded by inequalities of arbitrary shape. Their number may be finite or infinite and they may be defined implicitly. The convex feasibility problem includes as a special case the linear programming problem, the linear feasibility problem and the convex quadratic programming problem with quadratic constraints [19]. One of the key features is that inequality constraints that define the convex feasible region) are considered one at a time. Therefore IPCG algorithms can be quite beneficial in solving the SVM problem either in its primal form or its dual form.

The IPCG algorithm works as follows. We start with a sufficiently large "search region" Ω_0, such that $S \subset \Omega_0$. The algorithm iteratively cuts a portion away from the search region, thereby generating the sequence of shrinking sets:

$$\Omega_0 \supset \Omega_1 \supset \Omega_2 \supset .... \supset \Omega_k \supset \Omega_{k+1} \supset S. \tag{75}$$

At each iteration k, the "analytic center" y_k (the definition of this center will be given shortly) of Ω_k, is tested for feasibility with respect to S. If $y_k \in S$, then the algorithm stops. Otherwise, we have that $y_k \notin S$ and a cut is generated to form Ω_{k+1},

$$\Omega_{k+1} = \Omega_k \cap \left\{ y \in R^M : c_{k+1} - a_{k+1}^T y - \frac{1}{2} y^T Q_{k+1} y \geq 0 \right\}, \tag{76}$$

where c_{k+1} is a scalar, a_{k+1} is an $Mx1$ vector, and Q_{k+l} is an MxM symmetric positive semidefinite matrix (in the case of quadratic constraints). Cuts are always placed through Q_k in a way such $S \subset \Omega_{k+1} \subset \Omega_k$. The success of IPCG method depends critically on the choice of the center for each Ω_k. Intuitively we would like y_k to be located at the geometric center of Ω_k. Unfortunately the geometric center is difficult to compute [22]. In order to achieve overall computational efficiency the analytical center is used which depends on the analytic representation of Ω_k. It is defined as follows. To Ω_k we correspond the following potential function

$$\phi_k(y) = -\sum_{j=1}^{n_k} \log s_j, \tag{77}$$

where the slacks s_j are measures of the distance from y to each of the boundaries of Ω_k which is defined by n_k inequalities. Specifically,

$$s_j = c_j - a_j^T y - \frac{1}{2} y^T Q_j y, \quad j = 1, ..., n_k \tag{78}$$

The minimizer of ϕ_k is called the analytical center of Ω_k [42]. In [29] we use build-up and down strategies [21] in the IPCG framework which can be considered as an interior point version of Osuna's decomposition methods. There are several challenges like strategies which drop constraints that do not correspond to support vectors.

CONCLUSIONS

We have developed primal-dual algorithms for ANNs and SVMs training. A new class of incremental primal-dual techniques for problems with special decomposition properties was also discussed. The technique performs successive increments for each decomposition term of the objective function. The method is particularly beneficial for online applications and problems that have a large amount of data. We have shown that the technique can be nicely applied to artificial neural network training and we provided preliminary computational results. Convergence of the above algorithm is discussed in [10].

Primal-dual path following algorithms are relatively fast for medium sized problems. They are also robust in terms of accurate solutions, especially in the case of SVMs training where they compute global optimal solutions. Unfortunately they require computation and inversion of the kernel matrix and can become very expensive. Chunking and decomposition techniques of IPMs, like IPM column generation techniques can be beneficial in improving the complexity of those algorithms. This is the subject of a forthcoming paper [29].

References

[1] Battiti, R. (1992), "First and Second-Order Methods for Learning Between Steepest Descent and Newton's Method",*Neural Computation*, Vol. 4, 141–166.

[2] Bazaraa, M.Z., Sherali, H.D., and Shetty, C.M. (1993), *Nonlinear Programming: Theory and Algorithms*. John Wiley & Sons New York.

[3] Bertsekas, D.P. (1982), *Constrained Optimization and Lagrange Multiplier Methods*, Academic Press, San Diego, CA.

[4] Bertsekas, D.P. (1995), "Incremental Least Squares Methods and the Extended Kalman Filter", *Technical Report*, Department of Electrical Engineering and Computer Science, M.I.T., Cambridge, MA.

[5] Bertsekas, D.P. (1996), "A New Class of Incremental Gradient Methods for Least Squares Problems", *Technical Report*, Department of Electrical Engineering and Computer Science, M.I.T., Cambridge, MA.

[6] Bradley, P.S., Fayyad, U.M., and O.L. Mangasarian, (1998), "Data Mining: Overview and Optimization Opportunities, Mathematical Programming" *Technical Report 98-01*, University of Wisconsin Madison.

[7] Breitfeld, M. and Shanno, D. (1994), " Preliminary Computational Experience with Modified Log-Barrier Functions for Large-Scale Nonlinear programming ", *Large Scale Optimization State of the Art*. Hager, Hearn, and Pardalos, Editors. Kluwer Academic Publishers, 45-67.

[8] Byrd, R.H., Gilbert, J.C., and Nocedal, J. (1996), "A Trust Region Method Based on Interior Point Techniques for Nonlinear Programming", *Technical Report*, INRIA Rocquencourt, Le Chesnay, France.

[9] Cortes, C. and Vapnik, V., (1995) "Support Vector Networks", *Machine Learning* 20, 273 –297.

[10] Couellan, N.P. (1997), *Primal-Dual Techniques for Nonlinear Programming and Applications to Artificial Neural Network Training*, Ph.D. Dissertation, School of Industrial Engineering, University of Oklahoma, Norman, OK.

[11] Davidon, W.C. (1976), "New Least-Square Algorithms", *Journal of Optimization Theory and Applications*, Vol. 18, no. 2, 187-197.

[12] Deboeck, G.J. (1994), *Trading On the Edge*, John Wiley and Sons, New York.

[13] Dennis, J.E. and Schnabel, R.B. (1996), *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*, Classics in Applied Mathematics, SIAM, Philadelphia.

[14] El-Bakry, A. S., Tapia, R. A., Tsuchiya, T., and Zhang, Y. (1992),"On the Formulation and Theory of the Primal-Dual Newton Interior Point Method for Nonlinear Programming", *Technical Report*, Department of Computational and Applied Mathematics, Rice University, Houston, Texas, Revised 1995.

[15] Fiacco, A.V. and McCormick, G.P. (1968), *Nonlinear Programming: Sequential Unconstrained Minimization Techniques*, John Wiley, New York.

[16] Flippo, O.E. and Jansen, B. (1992), "Duality and Sensitivity in Quadratic Optimization Over a Sphere", Technical Report 92-65. Falculty of Technical Mathematics and Informatics, Delft University of Technology.

[17] Forsgren, A. and Gill, P.E. (1996), "Primal-Dual Interior Methods for Nonconvex Nonlinear Programming", *Technical Report NA-3*, UCSD Department of Mathematics, Royal Institute of Technology, Stockholm, Sweden.

[18] Girosi, F. (1998) "An Equivalence Between Sparse Approximation and Support Vector Machines", *Neural Computation*, 10(6), 1455-1480.

[19] Goffin, J.L., Luo, Z.Q., and Ye,Y. (1994), "On the Complexity of a Column Generation Algorithm for Convex or Quasi-convex Feasibility Problems" *Large Scale Optimization: State of the Art*, W.W Hager, D.W.Hearn and P.M. Pardalos,eds., Kluwer Academic Publishers, 182-189.

[20] Haykin, S. (1994), *Neural Networks: A Comprehensive Foundation*, MacMillan Publishing Company, N.Y.

[21] D.den Hertog, Kaliski, J., Roos, C., and Terlaky, T. (1995) "A Logarithmic Barrier Cutting Plane Method for Convex Programming", *Annals of Operations Research*, 58, 69-98.

[22] Kaiser, M.J., Morin, T.L., and Trafalis, T.B. (1991) "Centers and Invariant Points of Convex Bodies," *Applied Geometry and Discrete Mathematics: The Victor Klee Festschrift*, eds. P. Gritzmann and B. Sturmfels, AMS DIMACS Series in Discrete Mathematics and Theoretical Computer Science, American Mathematical Society, Providence, RI, 367-385.

[23] Karmarkar, N. (1984), "A New Polynomial-Time Algorithm for Linear Programming", *Combinatorica*, Vol. 4 (4), 373-395.

[24] Kasap, S. (1999), *Differential-Algebraic Equations in Interior Point Optimization Methods: A New Approach to the Parameterization of the Central Trajectory*, Ph.D. Dissertation in Progress, School of Industrial Engineering, University of Oklahoma, Norman, OK.

[25] Kasap, S. and Trafalis, T.B. (1999), *A New Approach to the Parameterization of the Central Trajectory in the Primal-Dual Interior Point Methods for Linear and Quadratic Programming Problems*, Working Paper, School of Industrial Engineering, University of Oklahoma, Norman, OK.

[26] Khachyian, L.G. (1979), "A Polynomial Algorithm in Linear Programming" (in Russian), *Doklady Akademiia Nauk USSR*, Vol. 224, 1093-1096.

[27] Lasdon, L.S., Plummer, J., and Yu, G. (1995), "Interior Point Algorithms for General Nonlinear Programs", *ORSA Journal on Computing*, Vol. 7 (3), 321-332.

[28] Lustig, I.J., Marsten, R.E., and Shanno, D.F. (1994), "Interior Point Methods for Linear Programming: Computational State of the Art", *ORSA Journal on Computing*, Vol. 6 (1), Winter 1994, 1-14.

[29] Malyscheff, A. and Trafalis, T.B. (1999), *Interior Point Method Column Generation Techniques Applied to Support Vector Machine Learning*, Working Paper, School of Industrial Engineering, University of Oklahoma, Norman, OK.

[30] Mangasarian, O.L. (1993), "Mathematical Programming in Neural Networks", *ORSA Journal on Computing* 5, 349–360.

[31] Mangasarian, O.L. (1996), "Mathematical Programming in Machine Learning", in *Nonlinear Optimization and Applications*, G. Di Pillo and F. Giannessi, editors, *Proceedings of Nonlinear Optimization and Applications*, Workshop, Erice, June 1995, Plenum Press, New York 1996, 283–295.

[32] Megiddo, N. (1986), "Pathways to the Optimal Set in Linear Programming", *Progress in Mathematical Programming*, ed. N. Megiddo, Springer-Verlag, New-York, 131-158.

[33] Monteiro, R.C. and Adler, I. (1989), "Interior Path Following Priaml-Dual Algorithms, Part II: Convex Quadratic Programming", *Mathematical Programming*, Vol. 44, 43-66.

[34] *NevProp*, University of Nevada at Reno, FTP://unssun.scs.unr.edu/pub/goodman/nevpropdir

[35] Osuna, E., Freund, R., and Girosi, F. (1997) "Training Support Vector Machines: An Application to Face Detection", *Proc. Computer Vision and Pattern Recognition '97*, 130– 136.

[36] Osuna, E., Freund, R. and Girosi, F. (1997) "Improved Training Algorithm for Support Vector Machines", *Proc. IEEE NNSP '97.*

[37] Refenes, A.P. (1995), *Neural Networks in the Capital Markets*, John Wiley and Sons, New York.

[38] Refenes, A.P., Abu-Mostafa, Y., Moody, J., and Weigend, A. (1996), "Neural Networks in Financial Engineering", *Series: Progress in Neural Processing*, Vol. 2, *Proceedings of the 3rd International Conference in Neural Networks in Capital Markets*, World Scientific Publishing Company, NJ.

[39] Scholkopf, B., Burges, C.J.C., and Smola, A.J. (1999), *Advances in Kernel Methods: Support Vector Learning*, The MIT Press, Cambridge, Massachusetts.

[40] Smola, A. J., (1998) *Learning with Kernels*, PhD Thesis.

[41] Söderstrom, T. and Stoica, P. (1989), *System Identification* , Prentice Hall International (UK), Englewood Cliffs, NJ.

[42] Sonnevend, G. (1985), "An Analytical Center for Polyhedrons and then New Classes of Global Algorithms for Linear (Smooth, Convex) Programming", *Proceedings of the 12th IFIP Conference on System Modeling and Optimizations*, Budapest, *Lectures Notes in Control Information Sciences*, Springer-Verlag, New-York, Vol. 84, 866-876.

[43] Sorensen, D.C. (1982), "Newton's Method with a Model Trust Region Modification", *SIAM Journal on Numerical Analysis*, Vol. 19, 409-426.

[44] Terlaky, T. (1996), *Interior Point Methods in Mathematical Programming*, Kluwer Academic Publishers.

[45] Trafalis, T.B. (1999), *Interior Point Optimization Methods: Theory and Applications*, Kluwer Academic Publishers, forthcoming.

[46] Trafalis, T.B. and Couellan, N.P. (1994), "Neural Networks Training via a Primal-Dual Interior Point Method for Linear Programming", *Proceedings of WCNN*, Vol. II, 798–803.

[47] Trafalis, T.B. and Couellan, N.P. (1996), "Neural Networks Training via an Affine Scaling Quadratic Optimization Algorithm", *Neural Networks* 9, 475–481.

[48] Trafalis, T.B. and Couellan, N.P. (1996), "Neural Network Training via Quadratic Programming", *Interfaces in Computer Science and Operations Research: Advances in Metaheuristics, Optimization and Stochastic Modelling*, R.S. Barr, R.V. Helgason and J.L. Kennington (eds.), Kluwer Academic Publishers, 123–139.

[49] Trafalis, T.B. and Couellan, N.P. (1998), "An Incremental Nonlinear Primal-Dual Algorithm and Applications to Artificial Neural Networks Training", *Proceedings of LSS'98*, preprints, Volume II, Elsevier, 1053–1059.

[50] Trafalis, T.B., Couellan, N.P., and Tutunji, T. (1996), "Interior Point Methods for Supervised Training of Artificial Neural Networks with Bounded Weights", In: *Lecture Notes in Economics and Mathematical Systems*, 450 (ed.)P.M. Pardalos, D. Hearn and W. Hager, Springer Verlag,441–470.

[51] Trafalis, T.B. and Sieger, D.B. (1993), "Training of Multilayer Feedforward Artificial Neural Networks by a Logarithmic Barrier Function Adaptation Scheme", In: *Intelligent Engineering Systems Through Artificial Neural Networks*, (ed.) C.H. Dagli, L.I. Burke, B.R. Fernandez and J. Ghosh, Vol. 3, ASME Press, 167–173.

[52] Trafalis, T.B., Terlaky, T., Warners, J.P., Quist, A.J., and Roos, C. (1996), "Unsupervised Neural Network Training Via a Potential Reduction Approach" *Technical Report*, Faculty of Technical Mathematics and Informatics, Delft University of Technology, Delft, Netherlands.

[53] Trafalis, T.B. and Tutunji, T. (1994), "A Quasi-Newton Barrier Function Algorithm for Artificial Network Training with Bounded Weights", In: *Intelligent Engineering Systems Through Artificial Neural Networks*, (ed.) C.H. Dagli, L.I. Burke, B.R. Fernandez and J. Ghosh, Vol. 4, ASME Press, 161–173.

[54] Trafalis, T.B. and Tutunji, T. (1995), "A Stochastic Logarithmic Barrier Function Algorithm for Neural Network Training", In: *Intelligent Engineering Systems Through Artificial Neural Networks*, (ed.) C.H. Dagli, L.I. Burke, B.R. Fernandez and J. Ghosh, Vol. 5, ASME Press, 167–172.

[55] Trafalis, T.B. and Tutunji, T. (1997), "Deterministic and Stochastic Logarithmic Barrier Function Methods for Neural Network Training", *Parallel Computing in Optimization*, A. Migdalas, P.M. Pardalos and S. Storoy, (eds.), Kluwer Academic Publishers, Chapter 13.

[56] Trafalis, T.B. and Tutunji, T. (1998), "Barrier and Stochastic Barrier Newton-Type Methods for Training Feedforward Neural Networks with Bounded Weights", *International Journal of Smart Engineering System Design*, vol. 1, 241–254.

[57] Tseng, P. (1995), "Incremental Gradient(-Projection) Method with Momentum Term and Adaptive Stepsize Rule", *Technical Report*, Department of Mathematics, University of Washington, Seattle, WA.

[58] Vandenberghe, L. and Boyd, S. (1994), "Positive Definite Programming", *Technical Report*, Information Systems Laboratory, Department of Electrical Engineering, Stanford University, Stanford, CA.

[59] Vanderbei, R.J., Meketon, M.S., and Freedman, B.A. (1986), "A Modification of Karmarkar's Linear Programming Algorithm", *Algorithmica*, Vol. 1, 395-407.

[60] Vanderbei, R.J. (1998), "LOQO an Interior Point Code for Quadratic Programming", *Technical Report*, Statistics and Operations Research, Princeton University, SOQ-94-15, Revised: November 30, 1998.

[61] Vapnik, V. (1982) *Estimation of Dependence Based on Empirical Data*, Springer Verlag.

[62] Vapnik, V. (1995) *The Nature of Statistical Learning Theory*, Springer Verlag.

[63] Wasserman, P.D. (1989), *Neural Computing: Theory and Practice*, NY: Van Nostrand Reinhold.

[64] Werbos, P. (1974), *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences*, PhD. Thesis, Committee on Applied Mathematics, Harvard University, Cambridge, MA. Reprinted in P. Werbos, *The Roots of Backpropagation: From Ordered Derivatives to Neural Networks and Political Forecasting*, NY: Wiley (1993).

[65] White, H., (1993), "Economic Prediction Using Neural Networks: The Case of IBM Daily Stock Returns", in Trippi, R. R. and Turban E. (Eds.), *Neural Networks in Finance and Investing*, pp. 315-328.

[66] Wright, S. (1996), *Primal Dual Interior Point Methods*, SIAM.

[67] Yamashita, H. (1992), "A Globally Convergent Primal-Dual Interior Point Method for Constrained Optimization", *Technical Report*, Mathematical System Institute, Inc., Shinjuku, Tokyo, Japan.

[68] Yamashita, H. (1994), "A Primal-Dual Interior Point Trust Region Method for Large Scale Constrained Optimization", *Technical Report*, Mathematical System Institute, Inc., Shinjuku, Tokyo, Japan.

[69] Ye Y. (1992), "On Affine Scaling Algorithms for Nonconvex Quadratic Programming", *Mathematical Programming*, Vol. 56, 285-300.