Interior Point Implementations of Alternating Minimization Training

Technical Report of the ISIS Group
at the University of Notre Dame
ISIS-94-008
June, 94

Michael Lemmon and Peter T. Szymanski
Department of Electrical Engineering
University of Notre Dame
Notre Dame, IN 46556

Interdisciplinary Studies of Intelligent Systems

Interior Point Implementations of Alternating Minimization Training

Michael Lemmon and Peter T. Szymanski

Dept. of Electrical Engineering

University of Notre Dame

Notre Dame, IN 46556

June 8, 1994

Abstract

This paper presents an alternating minimization algorithm used to train radial basis function networks. The algorithm is a modification of an interior point method used in solving primal linear programs. The resulting algorithm is shown to have a convergence rate on the order of $\sqrt{n}L$ iterations where n is a measure of the network size and L is a measure of the resulting solution's accuracy.

1 Introduction

In recent years, considerable research has investigated the use of alternating minimization (AM) techniques in the supervised training of radial basis function networks. AM techniques were first introduced in soft-competitive learning algorithms (Nowlan 1990). This training procedure was later shown to be closely related to Expectation-Maximization algorithms used by the statistical estimation community (Jordan and Jacobs 1993). Alternating minimizations search for optimal network weights by breaking the search into two distinct minimization problems. A given network performance functional is extremalized first with respect to one set of network weights and then with respect to the remaining weights. These learning procedures have found applications in the training of local expert systems (Jacobs, *et al.* 1991), and in Boltzmann machine training (Byrne 1992). More recently, convergence rates have been derived by viewing the AM method as a variable metric algorithm (Jordan and Xu 1993).

This paper examines alternating minimization as a perturbed linear programming (LP) problem. Recent advances in the application of barrier function methods to LP problems have resulted in the development of "path following" or "interior point" (IP) algorithms (Gonzaga 1992). These algorithms are characterized by fast convergence with low computational complexity. This paper shows how an IP algorithm associated with a primal LP problem can be modified into an alternating minimization training procedure. The principal result of this paper is a bound on how aggressively one leg of the alternating minimization can be attacked so that the resulting algorithm still has the sublinear convergence rates characteristic of its LP counterpart. Section 2 provides a formal problem statement. Section 3 describes the alternating minimization training algorithm. Section 4 proves the principal results of this paper. Section 5 summarizes preliminary simulation experiments supporting the paper's analyses.

2 Problem Statement

Consider a training set of R input-output pairs, $\mathcal{T} = (\mathbf{z}_i, y_i) \in \Re^N \times \Re$ for $i = 1, \ldots, R$. Consider the following stochastic approximator, which computes a *predicted output*, $\hat{y}$, given the input vector $\mathbf{z}$. The prediction is computed according to the following formula

$$\hat{y} = \hat{\theta}^T \mathbf{z} \tag{1}$$

where $\hat{\theta} \in \Re^N$ is a conditional random vector drawn from a finite set of M N-dimensional parameter vectors

$$\Theta = \{\theta_1, \ldots, \theta_M\} \tag{2}$$

It will be convenient to let $\hat{y}_m(\mathbf{z}) = \theta_m^T \mathbf{z}$ denote the predicted output associated with the mth parameter vector in Θ. The conditional probability mass function for the random vector $\hat{\theta}$ is conditioned on the input $\mathbf{z}$ and is given as follows,

$$\begin{aligned} Q(\hat{\theta}; \theta_m | \mathbf{z}) &= Q_{m|\mathbf{z}} && (3) \\ &= \text{Probability that } \hat{\theta} \text{ equals } \theta_m \text{ given input vector } \mathbf{z} && (4) \end{aligned}$$

for $m = 1, \ldots, M$ and $\mathbf{z} \in \Re^N$. The set of gating probabilities will be denoted as Q.

The preceding stochastic approximator can be viewed as a neural network. The network consists of $M + 1$ neurons. M of the neurons are *agent neurons* while the other neuron is referred to as a *gating neuron.* The agent neurons receive inputs from the vector $\mathbf{z}$. The mth agent neuron is parameterized by the vector $\theta_m \in \Theta$ ($m = 1, \ldots, M$). The output of the mth agent neuron in response to the applied input vector $\mathbf{z}$ is given by $\hat{y}_m(\mathbf{z}) = \theta_m^T \mathbf{z}$. The gating agent is parameterized by the set of conditional probabilities, $Q_{m|\mathbf{z}}$, $m = 1, \ldots, M$ and $\mathbf{z} \in \Re^N$. The gating neuron receives its input from the agent neurons' outputs and from the input, $\mathbf{z}$. The gating agent then selects the entire network's output to be the mth agent neuron's output. The selection is made in a probabilistic manner so that the probability of agent neuron m being selected given input $\mathbf{z}$ is equal to $Q_{m|\mathbf{z}}$.

The network will be said to be "optimal" with respect to a training set, $\mathcal{T}$, if the mean square output error is minimized. Let the squared output error of the mth neuron be expressed as $d_m(\mathbf{z}) = (\hat{y}_m(\mathbf{z}) - y(\mathbf{z}))^2$, the averaged squared output error is given by

$$\bar{d}(Q, \Theta) = \sum_{m,\mathbf{z}} p_{\mathbf{z}} Q_{m|\mathbf{z}} (\theta_m^T \mathbf{z} - y(\mathbf{z}))^2 \tag{5}$$

Since the gating agent weights represent probabilities, the minimization of $\bar{d}(Q, \Theta)$ is constrained so that Q is a valid conditional probability mass function.

Network training can therefore be viewed as a constrained optimization problem. In particular, this optimization problem can be expressed in a form very similar to conventional linear programming problems. The following notational conventions are adopted to highlight this connection. Let $\mathbf{x} \in \Re^{MR}$ be the gating neuron's weight vector. It is therefore assumed that the components of $\mathbf{x}$ correspond to distinct $Q_{m|\mathbf{z}}$'s. Let $\theta_m \in \Re^N$ denote the mth model neuron's weight vector and define the *cost* vector conditioned on parameter vector Θ as

$$\mathbf{c}(\Theta) = (c_{11}, \cdots, c_{1R}, c_{21}, \cdots, c_{2R}, \cdots, c_{M1}, \cdots, c_{MR})^T \tag{6}$$

where $c_{mr} = p_{\mathbf{z}_r}(\theta_m^T \mathbf{z}_r - y(\mathbf{z}_r))^2$, $m = 1, \cdots, M$ and $r = 1, \cdots, R$. With this notation, the network training problem can be stated as follows,

$$\begin{array}{rc} \text{minimize} & \mathbf{c}^T(\Theta)\mathbf{x} \\ \text{with respect to} & \mathbf{x}, \Theta \\ \text{subject to} & A\mathbf{x} = \mathbf{b} \\ & x_i \geq 0, i = 1, \cdots, MR \end{array} \tag{7}$$

where $\mathbf{b} = (1, \cdots, 1)^T \in \Re^R$ and $A = [I_{R\times R} \cdots I_{R\times R}] \in \Re^{R\times MR}$.

One approach for solving this problem is to break up the optimization into two steps. The first step involves minimizing the above cost functional with respect to $\mathbf{x}$ assuming a fixed

Θ. Since $\mathbf{x}$ is a vector of $Q_{m|\mathbf{z}}$'s, this is the Q-update of the algorithm. The second leg of the algorithm minimizes the functional with respect to Θ assuming fixed $\mathbf{x}$. This leg is called the Θ-update. Because the proposed optimization alternates between two different subsets of weights, this training procedure is often referred to as *alternating minimization*. Note that the Q-update is a linear programming problem while the Θ-update is a quadratic programming problem. Consequently, the AM training procedure can be viewed as a perturbed linear programming problem.

3 Proposed Training Algorithm

The preceding section noted that network training can be viewed as a perturbed linear programming problem. This observation is significant for there exist very efficient LP solvers based on the barrier function technique used in non-linear optimization. Recent advances in path following or interior point (IP) methods have developed LP solvers which exhibit convergence rates which scale in a sublinear way with problem size (Gonzaga 1992). This section introduces a modification of a primal IP algorithm that can be used for neural network training. The proposed modification is later shown to preserve the computational efficiency enjoyed by its LP counterpart.

To see how such a modification might arise, we first need to examine path following LP solvers. Consider the following LP problem.

$$\begin{array}{rc} \text{minimize} & \mathbf{c}^T\mathbf{x} \\ \text{with respect to} & \mathbf{x} \\ \text{subject to} & A\mathbf{x} = \mathbf{b} \\ & x_i \geq 0, i = 1, \cdots, n \end{array} \tag{8}$$

This problem can be solved by solving a sequence of augmented optimization problems arising from the primal parameterization of the LP problem.

$$\begin{array}{rc}\text{minimize} & \alpha^{(k)}\mathbf{c}^T\mathbf{x}^{(k)} - \sum_i \log x_i^{(k)} \\ \text{with respect to} & \mathbf{x}^{(k)} \in \Re^n \\ \text{subject to} & A\mathbf{x}^{(k)} = \mathbf{b} \\ & x_i^{(k)} \geq 0, i = 1, \cdots, n \end{array} \tag{9}$$

where $\alpha^{(k)} \geq 0$ $(k = 1, \cdots, K)$ is a finite length, monotone increasing sequence of real numbers. $\mathbf{x}^{(k),*}$ denotes the solution for the kth optimization problem in the sequence and is referred to as a *central point.* The locus of all points, $\mathbf{x}^{(k),*}$ where $\alpha^{(k)} \geq 0$ is called the *central path.* The augmented problem takes the original LP cost vector and adds a logarithmic barrier which keeps the central point away from the boundaries of the feasible set. As α increases, the effect of the barrier is decreased, thereby allowing the central point to approach the LP problem's solution in a controlled manner.

Path following (IP) methods solve the LP problem by approximately solving the sequence of augmented problems shown in (9). The parameter sequence, $\alpha^{(0)}, \alpha^{(1)}, \cdots, \alpha^{(K)}$, is chosen to be a monotone increasing sequence so that the central points, $\mathbf{x}^{(k),*}$, of each augmented optimization approach the LP solution in a monotone manner. It has been shown that for specific choices of the α sequence, that the sequence of approximate central points will converge to an ϵ-neighborhod of the LP solution after a finite number of iterations. For primal IP algorithms, the required condition is that successive approximations of the central points lie within the quadratic region of convergence for a scaling steepest descent (SSD) algorithm (Gonzaga 1992). A single SSD iteration is shown below

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - XP_{AX}X\nabla f_\alpha(\mathbf{x}) \tag{10}$$

where $P_A = I - A^T(AA^T)^{-1}A$, $X = \text{diag}(x_1^{(k)}, x_2^{(k)}, \cdots, x_n^{(k)})$, and $f_\alpha(\mathbf{x}^{(k)}) = \alpha\mathbf{c}^T\mathbf{x}^{(k)} - \sum_i \log x_i^{(k)}$. In particular, it has been shown that if the kth approximating solution is sufficiently close to the kth central point and if $\alpha^{(k+1)} = \alpha^{(k)}/\beta$ where $1/\beta \leq (1 + \nu/\sqrt{n})$ and

$\nu \leq 0.1$, then the "closeness" to the $(k+1)$st central point is guaranteed and the resulting algorithm will converge on the order $O(\sqrt{n}L)$ iterations where $L = n + p + 1$ specifies the size of the LP problem and p is the total number of bits used to represent the data in A, $\mathbf{b}$, and $\mathbf{c}$.

The preceding discussion reveals that a key component to a path following method's computational efficiency lies in controlling the iteration so that successive central points lie within the SSD algorithm's region of quadratic convergence. If we are to successfully extend such methods to (7), then this "closeness" of successive solutions must be preserved by the Θ-update of the algorithm. Due to the quadratic nature of the Θ-update, this minimization can be done exactly using a single Newton-Raphson iteration. Let Θ^* denote Θ-update's minimzer. If we update Θ to Θ^*, it is quite possible that the cost vector, $\mathbf{c}(\Theta)$, will be rotated in such a way that the current solution, $\mathbf{x}^{(k)}$, no longer lies in the quadratic region of convergence. Therefore, if we are to preserve the IP method's computational efficiency it will be necessary to be less "aggressive" in the Θ-update. In particular, this paper proposes the following Θ-update

$$\theta_m^{(k+1)} = (1-\gamma)\theta_m^{(k)} + \gamma\theta_m^{(k+1),*} \tag{11}$$

where $\theta_m^{(k)}$ is the mth parameter vector at time k, $\theta^{(k+1),*}$ is the minimizing parameter vector at time $k+1$ given by

$$\theta_m^{(k),*} = \left[\sum_{\mathbf{z}} p_{\mathbf{z}} Q_{m|\mathbf{z}}^{(k)} \mathbf{z}\mathbf{z}^T\right]^{-1} \sum_{\mathbf{z}} p_{\mathbf{z}} Q_{m|\mathbf{z}}^{(k)} y(\mathbf{z})\mathbf{z}. \tag{12}$$

and $0 < \gamma < 1$ controls the size of the update.

The preceding discussion suggests that a slight modification of the IP algorithm can be used to solve our network training problem. The actual form of the algorithm suggested by the above discussion is shown below.

Algorithm 3.1 (Proposed Training Algorithm)

```
Initialize
    k = 0.
    Choose x_i^(k), θ_m^(k), and α^(k) for i = 1,···,(MR), and for m = 1,···,M.
repeat
    α^(k+1) = α^(k)/β, where β ∈ (0,1).
    Q-update:
        x_0 = x^(k)
        for i = 0,···,P−1
            x_{i+1} = ScalingSteepestDescent(x_i, α^(k+1), Θ^(k))
        x^(k+1) = x_P
    Θ-update: Let
                θ_m^(k+1) = (1−γ)θ_m^(k) + γθ_m^(k+1),*
      where γ ≤ γ(x^(k+1), α^(k+1), c(Θ^(k))) and θ_m^(k+1),* is the minimizing θ_m.
    k = k+1
until(Δ < ε)
```

4 Theoretical Result

This section provides a bound on the parameter, γ, controlling the AM algorithm's Θ-udpate so that successive $\mathbf{x}^{(k)}$ vector solutions lie within the SSD algorithm's region of quadratic convergence.

Theorem 4.1 *Let $\theta_m^{(k)}$ and $\theta_m^{(k),*}$ be the current and minimizing parameter vectors at time k. Let $\mathbf{c}(\Theta^{(k)}) = \mathbf{c}^{(k)}$ and $\mathbf{c}(\Theta^{(k),*}) = \mathbf{c}^{(k),*}$. Let $\delta(\mathbf{x},\alpha,\mathbf{c}) = ||P_{AX}X\nabla_{\mathbf{x}} f_\alpha(\mathbf{x},\Theta)||$, where $f_\alpha(\mathbf{x},\Theta) = \alpha\mathbf{c}(\Theta)^T\mathbf{x} - \sum_i \log x_i$. Assume that $\delta(\mathbf{x}^{(k+1)},\alpha^{(k+1)},\mathbf{c}^{(k)}) = \delta_1 < 0.5$ and let $\theta_m^{(k+1)} = (1-\gamma)\theta_m^{(k)} + \gamma\theta_m^{(k+1),*}$. If γ is chosen as*

$$\gamma \leq \frac{\delta_2 - \delta_1}{\alpha||\mathbf{c}^{(k+1),*} - \mathbf{c}^{(k)}||} \tag{13}$$

where $\delta_1 < \delta_2 = 0.5$, then $\delta(\mathbf{x}^{(k+1)},\alpha^{(k+1)},\mathbf{c}^{(k+1)}) \leq \delta_2 = 0.5$.

Proof: We are interested in defining γ such that $\delta(\mathbf{x}^{(k+1)},\alpha^{(k+1)},\mathbf{c}^{(k+1)}) \leq 0.5$. Let $\mathbf{h}(\mathbf{x},\alpha,\mathbf{c}(\Theta)) = P_{AX}X\nabla_{\mathbf{x}} f_\alpha(\mathbf{x},\Theta)$ and let $\mathbf{h}_1 = \mathbf{h}(\mathbf{x}^{(k+1)},\alpha^{(k+1)},\mathbf{c}^{(k)})$ and $\mathbf{h}_2 = \mathbf{h}(\mathbf{x}^{(k+1)},\alpha^{(k+1)},\mathbf{c}^{(k+1)})$. Using the triangle inequality, it can be shown that

$$||\mathbf{h}_2|| \leq ||\mathbf{h}_2 - \mathbf{h}_1|| + ||\mathbf{h}_1||.$$

Since, $||\mathbf{h}_1|| \leq \delta_1$ and $\nabla f_\alpha(\mathbf{x}) = \alpha \mathbf{c}(\Theta) + \mathbf{x}^{-1}$ where $\mathbf{x}^{-1} = (x_1^{-1}, \cdots, x_n^{-1})^T$, then

$$||\mathbf{h}_2|| \leq ||\alpha P_{AX} X(\mathbf{c}^{(k+1)} - \mathbf{c}^{(k)})|| + \delta_1.$$

Convexity arguments produce $(\mathbf{c}^{(k+1)} - \mathbf{c}^{(k)}) \leq \gamma(\mathbf{c}^{(k+1),*} - \mathbf{c}^{(k)})$, resulting in

$$||\mathbf{h}_2|| \leq ||\alpha\gamma P_{AX} X(\mathbf{c}^{(k+1),*} - \mathbf{c}^{(k)})|| + \delta_1.$$

Since P_{AX} is an orthogonal projection $(||P_{AX}\mathbf{x}|| \leq ||\mathbf{x}||)$, $||A\mathbf{x}|| \leq ||A||\,||\mathbf{x}||$, and $||X||$ is the largest eigenvalue of X and $||X|| \leq 1$, then

$$||\mathbf{h}_2|| \leq \alpha\gamma ||\mathbf{c}^{(k+1),*} - \mathbf{c}^{(k)}|| + \delta_1.$$

Plugging in the value of γ from (13) yields

$$||\mathbf{h}_2|| \leq \left[\frac{\alpha||\mathbf{c}^{(k+1),*} - \mathbf{c}^{(k)}||(\delta_2 - \delta_1)}{\alpha||\mathbf{c}^{(k+1),*} - \mathbf{c}^{(k)}||}\right] + \delta_1.$$

Simplifying produces the desired result that $||\mathbf{h}_2|| \leq \delta_2 \leq 0.5$. □

Theorem 4.1 shows that the non-linear optimization can be embedded within the steps of the path following algorithm without it taking solutions too far from successive central points. Theorem 4.2 uses this result to show that the algorithm maintains the condition that $\delta(\mathbf{x}^{(k)}, \alpha^{(k)}, \mathbf{c}(\Theta^{(k)})) \leq 0.5$ for successive $\mathbf{x}^{(k)}$'s and $\theta_m^{(k)}$'s.

Theorem 4.2 *Let $\delta(\mathbf{x}^{(k)}, \alpha^{(k)}, \mathbf{c}(\Theta^{(k)})) \leq 0.5$ and let $\alpha^{(k+1)} = \alpha^{(k)}/\beta$ where $1/\beta \leq (1 + \nu/\sqrt{n})$ and $\nu \leq 0.1$. Let $\theta_m^{(k+1)} = (1-\gamma)\theta_m^{(k)} + \gamma\theta_m^{(k+1),*}$ where γ is from (13). Then, one iteration of Algorithm 3.1 with two SSD iterations produces $\mathbf{x}^{(k)}$ and $\Theta^{(k)}$, such that $\delta(\mathbf{x}^{(k+1)}, \alpha^{(k+1)}, \mathbf{c}(\Theta^{(k+1)})) \leq 0.5$.*

Proof: By assumption, $\delta(\mathbf{x}^{(k)}, \alpha^{(k)}, \mathbf{c}(\Theta^{(k)})) \leq 0.5$, meaning the current solution is near the current central point. Two SSD iterations produce $\mathbf{x}^{(k+1)}$ where $\delta(\mathbf{x}^{(k+1)}, \alpha^{(k+1)}, \mathbf{c}(\Theta^{(k)})) \leq$

$\delta_1 = 0.2401$ (Szymanski and Lemmon 1994). Using this value for δ_1 in Theorem 4.1, a valid γ can be computed using (13) where $\delta_2 = 0.5$, resulting in $\Theta^{(k+1)}$ such that

$$\delta(\mathbf{x}^{(k+1)}, \alpha^{(k+1)}, \mathbf{c}(\Theta^{(k+1)})) \leq 0.5.$$

$\square$

The convergence rate of the algorithm is established by using the LP problem's duality gap under the assertion that $\gamma \to 1$. The duality gap is the difference between the current solution for a primal and dual formulation of the LP problem. Path following algorithms allow the duality gap to be expressed as follows

$$\Delta(\alpha^{(k)}) = \frac{n + 0.5\sqrt{n}}{\alpha^{(k)}}. \tag{14}$$

and thus provide a convenient stopping criterion for the algorithm. Note that $\alpha^{(k)} = \alpha^{(0)}/\beta^k$ where $\beta \leq (1 + \nu/\sqrt{n})$. This implies that $\Delta^{(k)} = \beta^k \Delta^{(0)} \leq \beta^k 2^L$. If k is chosen so that $\beta^k 2^L \leq 2^{-L}$, then $\Delta^{(k)} \leq 2^{-L}$ which implies that $k \geq 2L/\log(1/\beta)$. Inserting our choice of β one finds that $k \geq (2\sqrt{n}L/\nu) + 2L$. The preceding argument establishes that the LP part of the algorithm will converge in $O(\sqrt{n}L)$ iterations where in our problem $n = MR$. In other words, the training procedure's training time scales in a *sublinear* manner with network size. Note that if γ goes to unity as $\alpha^{(k)}$ increases, then Θ will also have to converge to Θ^*. Methods for ensuring that $\gamma \to 1$ are currently under study. Simulation experiments in Section 5 suggest that slowing down the Q-update, which is done by reducing the value of ν, allows sufficient freedom for γ to go to unity.

5 Simulation Example

Various simulations were run on a time series prediction task to test the modified IP algorithm. The time series

$$y(t) = \sin(\pi t) - \sin(2\pi t) + \sin(3\pi t) - \sin(\pi t/2)$$

was used for the experiment and can be accurately predicted by an 8^{th} order linear filter. A sampling frequency of 20 Hz was used to produce 81 samples over 4 seconds of data. Figure 1 depicts the function and the sampled points.

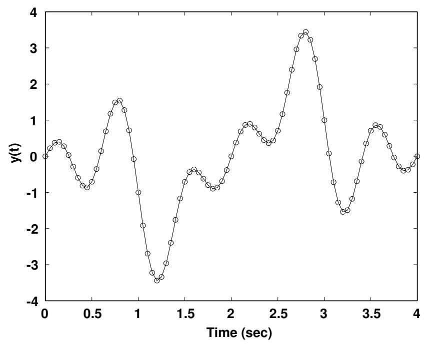

Figure 1: Time Series

Experiments were run to determine how a number of lower order systems could approximate this higher order system. The goals of the experiments were to test the convergence time of the algorithm, qualitatively examine the effects of the parameter, ν, on the convergence of the model parameter vectors to locally optimal values, and to examine how the use of the lower order models affects the approximation.

The first set of results describes the convergence of the algorithm. The experiments consisted of 100 randomly chosen starting Θ's for each of a number of different networks ranging in size from $M = 2$ to 25 model agents. This corresponds to an LP problem dimension of $n = 66$ to 825. The number of iterations for the duality gap to fall below a prespecified tolerance of $\epsilon = 10^{-4}$ was computed using knowledge of the starting duality gap and the parameter, β. The results are shown in Figure 2. This shows that as n grows the convergence rate tends to the predicted theoretical bound.

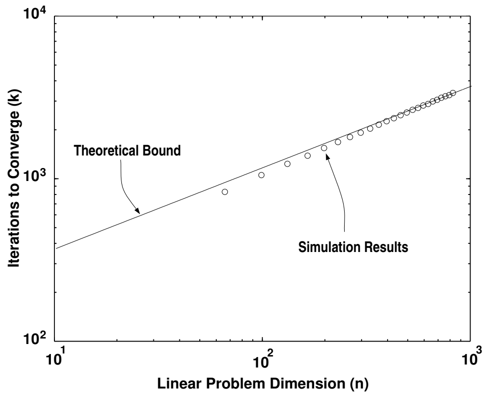

Figure 2: Convergence Rate as a function of n

Additional simulations demonstrate that the choice of ν, which controls the distance between successive central points, affects γ and the convergence of the Θ parameters. In the left panel of Figure 3 with $\nu = 0.1$, the Θ-update stalls as $\gamma \to 0$ as k increases indicating lack of convergence of the θ_m's to a locally optimal solution. The right panel of Figure 3 with $\nu = 0.05$ depicts convergence of $\gamma \to 1$ as k increases, indicating convergence of $\Theta^{(k)}$ to a locally optimal solution. These results indicate that taking too big a step in the Q-update may not allow for the convergence of the Θ's to locally optimal Θ's. An appropriate bound on ν, which controls the step size, is required. Future work will address a schedule for ν that guarantees convergence of the Θ's much the same as the choice of γ guarantees the convergence of the Q's.

The last set of results demonstrates the effects of using multiple models on the distribution of the data points among the set of models. Figure 4 depicts the distribution of the data points among three different model agents. Each data point is represented by a $*$, $+$, or a $\circ$ indicating that it was mapped to model 1, 2, or 3, respectively. The training resulted

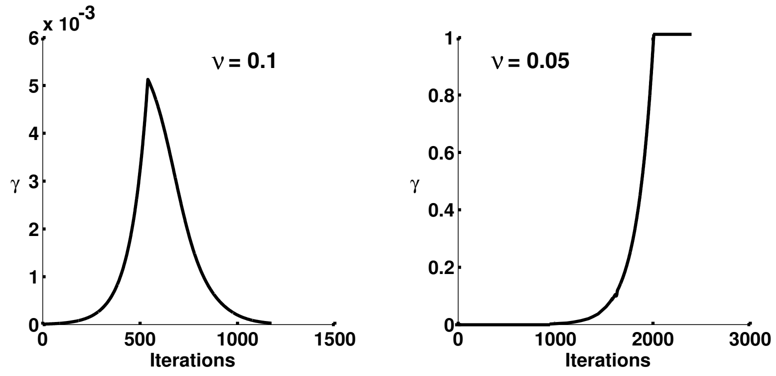

Figure 3: Convergence of γ

in a distinct grouping of data points which had similar optimal parameter vectors. It is interesting to note that the modeling maps points to the model that best approximates them and does not constrain the groups to be completely contiguous.

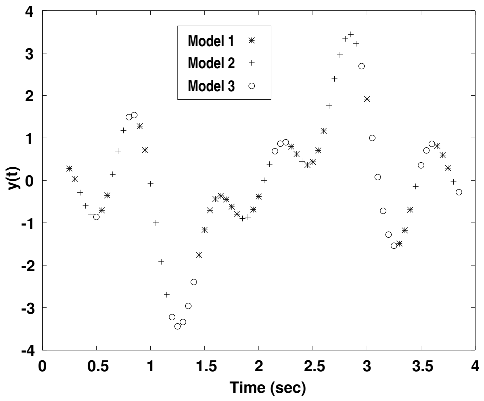

Figure 4: Distribution of Data Points among Models

6 Summary

This paper has presented an alternating minimization algorithm which can be proven to converge on the order of $\sqrt{n}L$ iterations. The work has established a means by which interior point methods can be applied to neural network training in a way which preserves the computational efficiency of interior point LP solvers. The algorithm presented here represents a small-step version of the interior point method. An interesting area of future research concerns large-step versions and their relationship to other alternating minimization procedures based on Expectation-Maximization methods. Relationships to a number of different areas exist including Expectation-Maximization algorithms (Dempster, Laird, and Rubin 1977), soft competition (Nowlan 1990), and adaptive mixtures of experts (Jacobs, *et al.* 1991, Jordan and Jacobs 1993, Jordan and Xu 1993).

References

Bazaraa, M.S., Sherali, H.D., and Shetty, C.M. 1993. *Nonlinear Programming: Theory and Algorithms.* John Wiley and Sons, Inc., New York.

Byrne, W. 1992. Alternating minimization and Boltzmann machine learning. *IEEE Transactions on Neural Networks*, 3(4):612–620.

Dempster, A.P., Laird, N.M., and Rubin, D.B. 1977. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society*, 39:1–38.

Gonzaga, C.C. 1992. Path-following methods for linear programming. *SIAM Review*, 34(2):167–224.

Jacobs, R.A. and Jordan, M.I. 1991. A modular connectionist architecture for learning piecewise control strategies. In *Proceedings of the American Control Conference*, volume 2, pages 1597–1602, Boston, Massachusetts.

Jacobs, R.A., Jordan, M.I., Nowlan, S.J., and Hinton, G.E. 1991. Adaptive mixtures of local experts. *Neural Computation*, 3:79–87.

Jordan, M.I. and Jacobs, R.A. 1993. Hierarchical mixtures of experts and the EM algorithm. Technical Report 9301, MIT Computational Cognitive Science.

Jordan, M.I. and Xu, L. 1993. Convergence results for the EM approach to mixtures of experts architectures. Technical Report 9303, MIT Computational Cognitive Science.

Nowlan, S.J. 1990. Maximum likelihood competitive learning. In *Advances in Neural Information Processing Systems 2*, pages 574–582. Morgan Kaufmann Publishers, Inc., San Mateo, California.

Szymanski, P.T. and Lemmon, M.D. 1994. A modified interior point method for alternating minimizations. Technical Report ISIS-94-005, Department of Electrical Engineering, University of Notre Dame.